Multiuser Geographic Information Systems with ArcInfo™ 8

An ESRI® Technical Paper • April 2000

ESRI 380 New York St., Redlands, CA 92373-8100, USA • TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB www.esri.com

Multiuser Geographic Information Systems with ArcInfo 8

An ESRI Technical Paper

Contents **Page**

Multiuser Geographic Information Systems with ArcInfo 8

Introduction

The ArcInfo™ geodatabase model is implemented on standard relational databases with the ArcSDE™ (formerly Spatial Database Engine™ [SDE®] software) application server. ArcSDE defines an open interface to database systems for our users. It allows ArcInfo to manage geographic information on a variety of different database platforms including Oracle®, SQL Server™, DB2™, and others.

Implementing a geodatabase with the ArcSDE application server allows you to provide access to multiple users for both query and update of your geographic data. ArcSDE and ArcInfo support long transactions through versioning. Versioning lets users simultaneously create multiple, persistent representations of the database without data replication. Since all changes are recorded in delta tables, users can edit the same features or rows without explicitly applying locks to prohibit other users from modifying the same data.

Successfully implementing a multiuser geographic information system (GIS) with ArcInfo and ArcSDE starts at the very beginning with data model design and database tuning. How the data is stored in the database, the applications that access it, and the client and server hardware configurations are all key factors to a successful multiuser GIS system. This technical paper will provide guidelines for each of these factors and discuss each step of the process to help you implement a successful multiuser GIS system with ArcInfo and ArcSDE. It will also present the results of a case study where a multiuser system is developed for a gas database from the ground up.

Implementing Your GIS System

This section provides broad guidelines as to what to do and what to avoid when implementing your multiuser systems. There are good practices to follow through each phase of the implementation process to achieve a high level of performance and scalability. These points are presented in an ordered fashion beginning with tuning the database where the data is physically located, up to configuring the application you use to work with the data.

Creating Your Oracle Database

Creating an Oracle instance and database requires an understanding of how your applications and users will be accessing and maintaining your data. Simply installing Oracle and creating your geodatabase will not suffice and will lead to extreme performance bottlenecks. You must start by designing how your Oracle instance and database will be distributed across your hardware's physical devices, how many resources will be available based on your available hardware, and your database's backup and recovery strategy.

For additional information see the ArcSDE "Configuring and Tuning Oracle" documentation or your Oracle documentation. The following discussion is merely an introduction to the terminology and concepts. Please always reference Oracle's documentation for comprehensive descriptions and examples.

The first step in creating your Oracle database is to determine the optimal distribution of the instance's control files, redo log files, rollback tablespaces, system tablespace, user tablespaces, temporary tablespaces, and archive files. Each of these is briefly described below.

- Control files maintain the structure of the database. The control files are required to mount, open, and access the database and are used to identify all database files and log files. The control files must always be available for writing by the database; the recommended configuration is a minimum of two control files located on different disks.

- Redo log files record each change made to the database and are used for recovery. There must be a minimum of two redo log files.

- Rollback tablespaces store rollback segments that maintain the data's "before" image of uncommitted transactions. The rollback segments provide read consistency for queries started before a transaction is committed. Databases with frequent updates will experience a high degree of disk I/O to the rollback data files.

- The database will contain one system tablespace used by Oracle for the data dictionary. The data dictionary contains metadata for each object in the database and is accessed each time a SQL statement is parsed to validate an object's existence and privileges.

- User tablespaces are the storage containers for your database's tables and indexes. Based on the size of your database (the amount of data you are maintaining), your available hardware, number of users, and your applications will determine how many tablespaces may be required.

- Temporary tablespaces are used for building indexes, performing sort operations, and generating statistics.

- Archive files are the copied redo log files, created following a log file switch and used for media recovery. Your database's transaction activity and size of the redo log files will determine the frequency with which archive files are created.

How you configure each of these database objects will influence how your database performs. For example, if you place a heavily accessed tablespace on the same device as your database's redo log files, you may suffer severe performance degradation when your applications are updating the data.

Once you have created your database you will need to customize the instance's initialization parameters in the init.ora file. These parameters define the characteristics of the instance such as the memory usage. For a detailed discussion of the Oracle initialization parameters refer to Oracle's documentation "Oracle8 Server Tuning."

To access your database you will need to create database users. Your users will access the database through applications by providing a user name and password to establish a connection. Each user should have a default tablespace and temporary tablespace assigned when the account is created. The default tablespace is used as the default storage container if the tablespace is not specified when new tables or indexes are created. The temporary tablespace is used when a SQL statement is issued that requires the use of a temporary segment. Therefore, how your users will be accessing the database will influence your decision as to which default and temporary tablespaces are assigned to each user.

Once your users have been created it may be wise to manage database and system resources by assigning profiles. A profile is a named set of resource limits. If you enable resources, Oracle will limit a user's use of the database based on the limits you set and assign for a given profile. This will ensure that no user will mistakenly perform an operation that might consume a significant amount of resources that will then affect the database's performance for all other users.

Just as important as configuring your Oracle database is choosing and implementing a backup and recovery strategy. You should decide on your strategy based on what your organization deems acceptable with respect to data loss from disk failure, point-in-time recovery to correct invalid transactions, and the availability of the database (if you are a twenty-four hours a day, seven days per week organization).

Some organizations elect to implement a redundant arrays of independent disks (RAID) system as a mechanism to provide data redundancy in case of disk failure. RAID, in part, is the striping of data across several physical drives and can be implemented in any of several levels, ranging from 0 to 6. Levels 1 and 5 are most commonly used. But choosing which level to implement and how can be a very complex decision that must balance performance, cost, and the availability of data.

If disk space is not an issue, RAID level 1 is the most beneficial providing full data redundancy. Query and write performance are not impacted and are no different than usual. For all other levels, writing to the file system is slower and reads may typically be faster. Whatever level is implemented, it is transparent to your Oracle database. All the RAID features are handled by the operating system.

If you do implement a RAID system, Oracle strongly recommends not placing your instances redo logs on a RAID controlled device. Because the redo log files are accessed sequentially by block, not randomly, in the case of a tablespace's data file, performance will degrade. Also, temporary tablespaces should not be located on a level 5 device because of the performance associated with an Oracle sort operation on disk.

No matter what strategy you elect to implement, it needs to be thoroughly tested in an environment before and after you implement your production database.

Recommendations

Always keep in mind how each database object interacts with the other database objects and attempt to properly distribute these objects across multiple hardware devices. At all costs, avoid placing any additional database files on the same device as the redo log files. This is the area of greatest contention. It is also recommended to distribute redo log files or groups on multiple devices. This will avoid disk contention while the archive process is reading the redo log file, if archiving is enabled. As well, the archive destination should be located on a device other than the device in which the current redo log file is located when being archived.

When creating database users, be sure to specify a default and temporary tablespace. If not, Oracle will assign the SYSTEM tablespace for the default and temporary tablespace. This can impact database performance. Also take advantage of assigning profiles to users. This will ensure that no processes can overextend the database's and hardware's resources, which could significantly impact all other users of the database.

There are a number of further recommendations that could be made, but they are often dependent on the environment and application. Those mentioned above are just a few. The case study at the end of this document contains additional recommendations and tips to consider when you implement your database.

Designing Your Geodatabase

The schema design phase of your geodatabase implementation is critical to the ultimate performance of your GIS system. Geodatabases can contain feature data sets, geometric networks, relationship classes, feature-linked annotation, and other specialized abstractions. Careful consideration of the cost, limitations, and true purpose of each of these abstractions is required to avoid creating a geodatabase that may not perform well in either a single- or multiuser environment.

The following is a description of common geodatabase modeling errors, why they should be avoided, and suggestions for modeling your database design to avoid them.

Feature Data Sets

Feature data sets exist in the geodatabase to define a scope for a spatial reference. All feature classes that participate in topological relationships with one another (e.g., a geometric network) must have the same spatial reference. Feature data sets are a way to group feature classes with the same spatial reference so that they can participate in topological relationships with each other.

To most users, feature data sets also have a natural organizational quality, much like a folder on a file system. Since for many GIS applications the majority of the data has the same spatial reference, the temptation to group large numbers of feature classes into feature data sets is irresistible.

Feature data sets, however, are not free. When you open a feature class contained in a feature data set to look at its properties or draw or query it in ArcCatalog™, ArcMap™, or a custom application, all of the other feature classes in that feature data set are also opened. This is done because updates to a feature class in a feature data set can

potentially ripple to other feature classes in the feature data set that participate in topological relationships. The effect of this is that feature data sets with a large number of feature classes will be slow to open for the first time.

For example, when browsing such a feature data set in ArcCatalog, there is no perceptible difference in performance until you either open the properties dialog or try to preview the contents of one of the feature classes. There will be a time lag while ArcCatalog opens all of the feature classes in the feature data set. Once all are open, the properties dialog appears, or the data appears. Performing the same action on another feature class will not cost you the same penalty since the feature data set is already opened.

The same effect is realized when adding a feature class from a feature data set to your map in ArcMap or when opening an ArcMap document that contains a feature class from a feature data set.

Recommendations

Use feature data sets to group classes that have topological relationships with each other (i.e., geometric networks and shared boundaries). Beyond that, don't overload feature data sets with lots of feature classes. Having stand-alone feature classes (at the database level) is perfectly acceptable. If you do want to group your classes into feature data sets, try to group feature classes based on those that are used together in your applications.

Geometric Networks

Geometric networks can be comprised of a number of edge and junction feature classes. When editing geometric networks in ArcMap, topological relationships between features and between feature classes are maintained while editing on the fly. The benefit of this model is that there is no need to perform a postediting process to build topology. The downside of this is it does impose a cost on the time it takes to add or modify features in network feature classes.

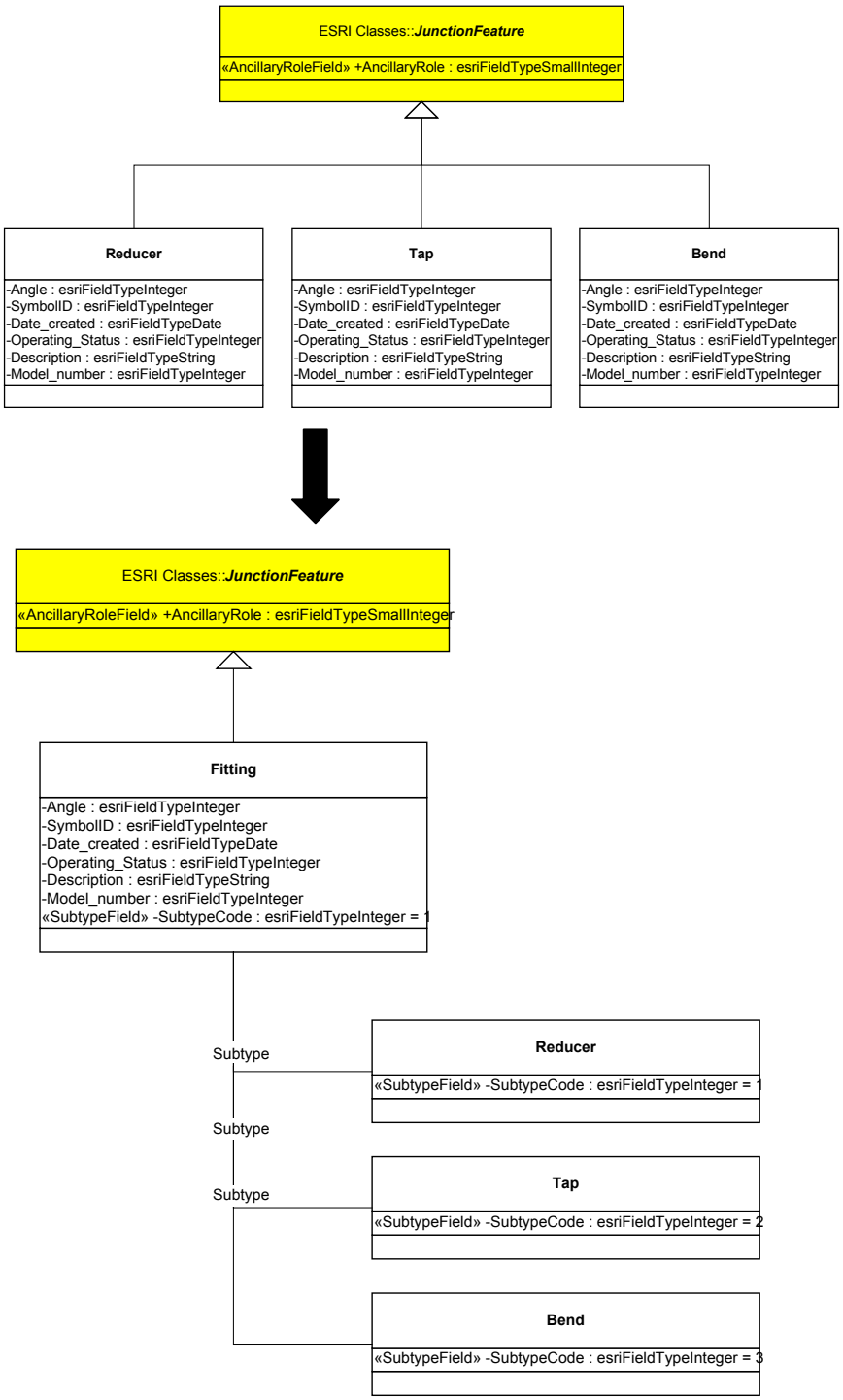
Topological connectivity in a network feature class is based on geometric coincidence. If a junction is added along an edge, they will become topologically connected to one another. When a new feature is added to a network feature class, this geometric coincidence must be discovered. So each feature class in the network must be analyzed by performing a spatial query against each network class to determine if the new feature is coincident with other network features at any point, and if so, then the user must establish the connectivity.

If you do not use the edit cache when editing your network features, as there are more feature classes in the network, the performance will be correspondingly slower for adding new features, connecting, and moving existing features. For more information on the edit cache, see the Tuning Your Application section in this document.

Recommendations

Try to reduce the number of feature classes you have in your geometric network by lumping feature classes together using subtypes. If your feature classes carry different attributes, you can use relationships to manage subtype-specific attributes in different tables in the database, or you can keep all the attributes in the same table using nulls for those that do not apply to a particular subtype.

J-8457



Reduce the number of classes in your model by lumping them together using subtypes. In this example, three classes, Reducer, Tap, and Bend, were lumped into a single feature class—Fitting—using subtypes.

When discovering connectivity, a separate spatial query must be executed on the server for each feature class in the network. If you use the edit cache while editing the network, these spatial queries do not need to go against the server and are much faster. You will not pay as much of a penalty for having a large number of feature classes in your network if you use the edit cache.

Relationship Classes and Feature-Linked Annotation

Relationship classes allow you to maintain associations between objects in your geodatabase. These relationships can be simple and passive, or they can be composite. Composite relationships imply parent/child or composition and therefore have behavior, which is triggered through changes to objects on one side of the relationship to objects on the other side.

When editing composite features, edits such as move, rotate, and delete are cascaded to the related objects through the relationship class. There is a cost when navigating these relationships. The cost is minimized when indexes are maintained for the primary and foreign keys for the relationship class.

Feature-linked annotation maintains the link between a feature and its annotation element through a composite relationship. Feature-linked annotation has additional behaviors to a typical composite relationship in that every time a new feature is created, a new annotation feature is also created in any related annotation classes, unless this behavior was disabled when the annotation class was created.

It is important to realize that if you edit a feature class that participates in a relationship class with messaging and the related class is not in the map, that map will open so that it can respond to the message (by moving, deleting itself, or performing some custom behavior).

With many traditional ArcInfo coverage data models, the feature attribute table contained as few items as possible, and many of the attributes for a feature class were contained in a related table. This can be done with geodatabase feature classes; however, navigating a relationship in the geodatabase is a more costly operation than navigating relates in INFO. In the INFO environment, it was common to store the symbology for a feature in an external related table called a lookup table. This can still be done in the geodatabase using relationship classes; however, for performance considerations, it is recommended that symbology information be stored in the feature class itself.

Recommendations

In general, when working with a class in ArcMap, have all related classes, including feature-linked annotation, also in the map. This way, the related classes are opened once when they are added to ArcMap. If they are not in the map, then each time you access related objects the class must be opened.

You can symbolize features based on a field in a related class; however, there is a cost to navigating relationships. For large data, symbolizing this way will be slow, even with indexes on the primary and foreign keys. Try to keep attributes for symbolization on the feature class table to avoid this performance hit.

Loading Your Data—The Initial Load

Once you have designed your geodatabase and tuned the relational database management system (RDBMS), you may have some existing data to load into your geodatabase. You need to be aware of a number of things that can impact performance when loading data. This section discusses how data loading can affect performance and gives examples for data loading strategies.

Data Conversion

If you are not using Computer-Aided Software Engineering (CASE) tools to generate your database schema, loading data into the geodatabase is a straightforward process. Use the data conversion tools in either ArcCatalog or ArcToolbox™ software to convert your coverage and shapefile data into feature classes, then build any required networks and register your data as versioned.

Alternatively, you can use ArcCatalog to create your feature classes, use the Simple Data Loader to load your shapefile and coverage data into those feature classes, and then build any networks and version the data. Try to do as much data conversion and data loading as possible before building any networks.

Using CASE Tools and Loading Data

Geodatabase design tools in ArcInfo allow you to design your database schema using Unified Modeling Language (UML) and CASE tools. The schema generation tools in ArcCatalog translate your UML design into an empty geodatabase schema, that is, a set of feature classes, tables, relationship classes, geometric networks, and rules. The different strategies for loading your existing data into that schema, and the performance considerations of each, are the subject of this section.

Designing your database schema using UML and CASE tools provides a number of data loading considerations. When generating schema using CASE tools and loading data, there are two strategies you can use.

Strategy 1: Using the Simple Data Loader

1. Use the Schema Generation wizard to create the empty geodatabase schema from your UML model in your database.

2. Delete any networks that were created. This will also delete any associated connectivity rules and class extensions.

3. Load all of your data into your database using the Simple Data Loader in ArcCatalog.

4. Build any required networks using the Build Geometric Network wizard in ArcCatalog or ArcToolbox.

5. Use the Schema Generation wizard to reapply the UML to the existing data to re-create the network connectivity rules and assign any class extensions.

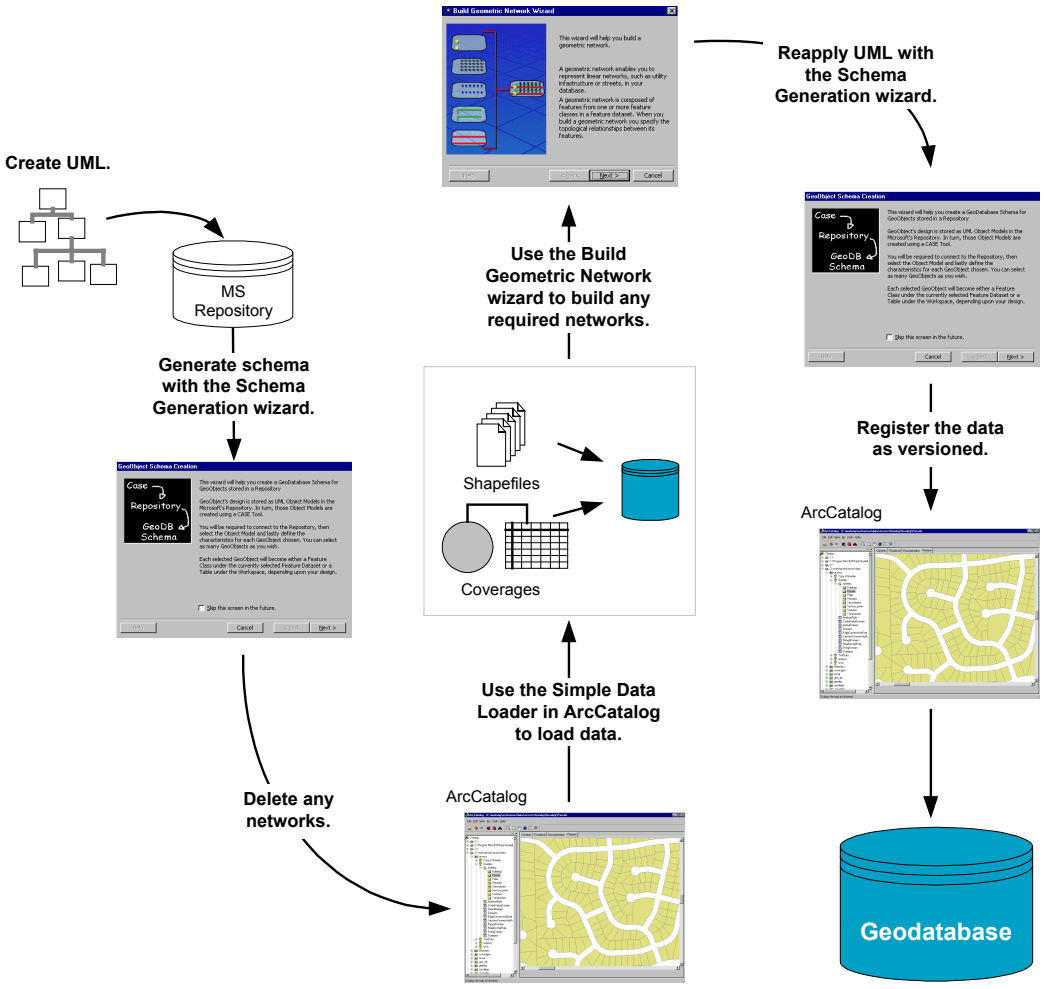
6. Register your data as versioned.

This strategy has a number of advantages. By loading all of your data with no network, the data will load much faster. Since the data is not versioned, all of the data will be loaded directly into the base tables, and a database compress will not be required. When

J-8457

you delete the network in step 2, it will automatically delete all connectivity rules associated with that network and revert all of its participant feature classes to simple. By reapplying the model after the network is built, your connectivity rules are reapplied and any class extensions described by the model are also reassociated with their corresponding classes.

The only limitation of this method is when you have classes with custom objects that have custom object creation behavior. This custom creation behavior would not be executed. In these cases, you may want to do a combination of the first and second method (see below): load all noncustom features, build networks, apply your model, which will create the custom object classes, then version your data and use the object loader to populate the custom classes. Do not forget to run compress.



Data Loading Strategy 1: Using the Simple Data Loader

Strategy 2: Using the Object Loader

1. Use the Schema Generation wizard to create the empty geodatabase schema in your database.

2. Use the Simple Data Loader in ArcCatalog to load your existing data into your simple feature classes and tables.

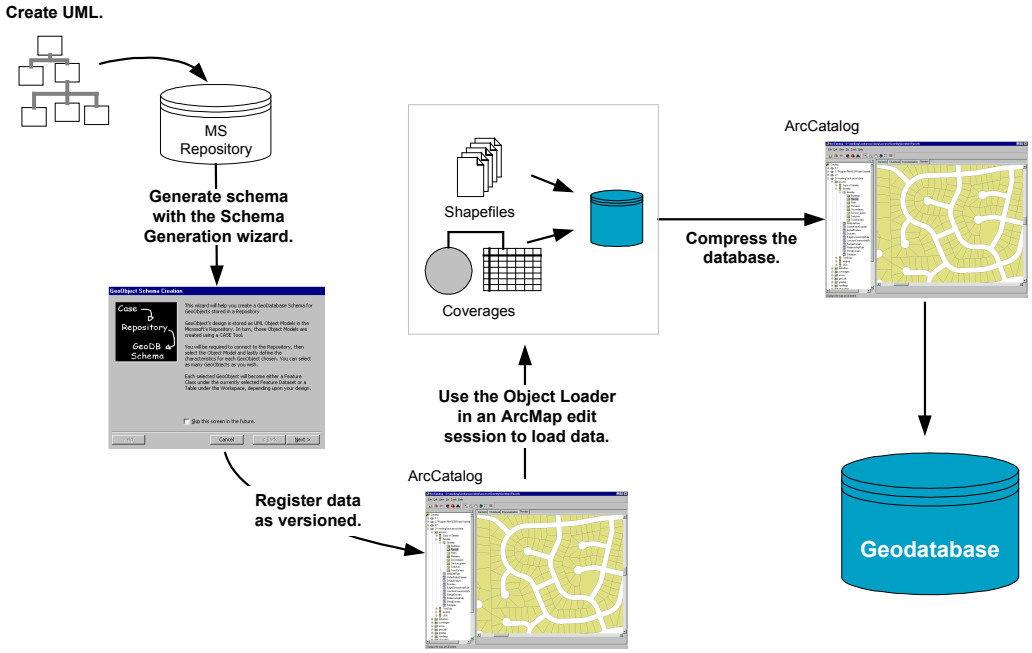
3. Register your data as versioned.

4. Use the Object Loader in ArcMap to load your existing data into your network feature classes. This step automatically builds network topology within an edit session.

5. Run compress to compress the database (you should always run compress after large data loads into versioned classes).

6. Use the ArcSDE utility `sdetable -o update_dbms_stats` to update the database statistics for each feature class that you loaded data into.

This strategy has a number of disadvantages. First, loading data into network feature classes is a slow process—several seconds per feature. Once all of the data is loaded, since it is versioned, it will be in the delta tables, not the base tables for the feature classes. If you use this method to load your data, once it is loaded you should run compress on your database to push all the records from the delta tables to the base tables. Having your data in the base tables will result in better query speed than if you have large amounts of data in your delta tables. For more details on compressing your database to improve performance, see Compressing Your Database later in this document.



Data Loading Strategy 2: Using the Object Loader

In general, it is better to do as much data loading and network building as possible before versioning your data and applying your UML model to generate subtypes, rules, and relationships.

Loading Annotation

Annotation can be created in bulk in the geodatabase by converting labels to annotation in ArcMap or by converting coverage annotation by loading it into an annotation feature class.

In general, creating annotation in bulk using either of these methods should be done before you version your data. This way, all of the annotation features are added directly to the base tables. If you load your annotation into versioned annotation classes, it requires you to be editing, and all of the new annotation features are added to the delta tables for the version you are working with. If you do create a large amount of annotation this way into a versioned annotation feature class, you should run compress on the database to move those features into the base tables for the feature class.

If you are creating annotation for network features by converting labels to annotation in ArcMap, you should do so after you build the network. If you snap features when building your network, features will be moved. For performance, the moving of features is done at a level that does not trigger object behavior. This means the moving of features will not trigger the linked annotation to move with it, which can result in feature-linked annotation not being in the correct place relative to its related feature.

Preprocessing the Data Using SQL Prior to Versioning

After loading a large amount of data into your geodatabase, it may be the case that you want to bulk update some values for some of the attributes. You can do this by versioning the data, then using ArcMap and the Editor to perform the update. The problem with this approach is that all of the updated features will be in the delta tables, and you will want to compress your database to move them into the base tables.

Another approach is to identify these kinds of bulk attribute updates that can be done using SQL before the data is versioned. Using this approach means that these bulk updates are done before the database is versioned, and all of the features remain in the base tables.

There are some rules that apply to performing this kind of operation. It is important to understand your data model so that the attributes you update do not affect other objects in the database through relationships or other behavior. If you use SQL to do this kind of operation without this understanding, it may result in data corruption. For example, if you use SQL to modify the attributes of a feature from which text is derived for feature-linked annotation, the annotation features will not be messaged to update themselves, so the annotation and feature will be out of sync.

The following is a list of some important guidelines when doing this kind of operation.

- Never update records in SQL after your data has been versioned.
- When updating data using SQL, do not modify attributes that through geodatabase behavior affect other objects in the database.

- Never update the ObjectID field with SQL.
- Never update the Enabled or AncillaryRole field for a network feature class using SQL. When this field is updated through ArcInfo, it results in changes to the geometric network topology tables that SQL will not trigger.

Compressing Your Database

As your ArcSDE geodatabase is edited over time, the number of database states and rows in each feature class's delta tables will increase. Some database states may no longer be referenced by a version, but the edits or rows associated with those database states may still exist in the delta tables. Unfortunately, all queries will become slower as the number of rows increase in the delta tables, just as any query would as the number of rows increased in the table.

Running the compress command from ArcCatalog will remove all database states that are no longer referenced by a version and move all the rows in the delta tables, which are common to all versions, to the base table. By reducing the number of rows in the delta tables, performance will improve. The compress command should be run periodically throughout the lifetime of a geodatabase. However, keep in mind that compress can only be executed by the ArcSDE administrator when no users are connected to the database.

To achieve the maximum benefit when running the compress command, you will need to first reconcile, post, and delete each version with the DEFAULT version prior to executing the command. Sometimes this may not be a reasonable option based on your organization's work flow process. At minimum, to improve performance simply reconcile each version with the DEFAULT version and then perform the compress command; this will move all the edits in the delta tables to the base tables that reference the DEFAULT version.

Remember the compress command can still be executed without first reconciling, posting, and deleting each version, but the benefits may not be as noticeable. Finally, after performing the compress it is extremely important to update the database statistics for each feature class by running the ArcSDE utility `sdetable -o update_dbms_stats`.

Loading Your Data—Appending Data to a Geodatabase

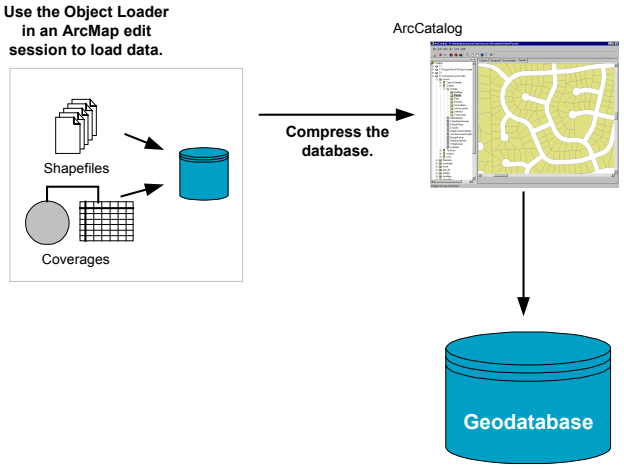
Once your database is built and your data is loaded, at some point in time you may want to append another area of data to your existing database. If you have no network data in your geodatabase, the process is straightforward.

1. Use the Object Loader in ArcMap to load the new data into your feature classes.
2. Run compress to compress the database (you should always run compress after large data loads into versioned classes).
3. Use the ArcSDE utility `sdetable -o update_dbms_stats` to update the database statistics for each feature class that you loaded data into.

All object behavior is executed such as creating linked annotation for feature classes that have a linked annotation class.

If you are appending data to network feature classes, then the above method will work, but since the entire network cannot be cached when using the Object Loader to load data into network feature classes it will be a very slow process—up to eight to twelve seconds per feature, depending on the number of feature classes in the network. For example, loading 1,000 new features into a network using this method may take as long as three hours. This method may be slow, but it is reliable and executes any necessary object behavior.

While the Object Loader wizard in ArcMap allows only one target feature class per loading operation, you can write custom scripts using Visual Basic, Visual Basic for Applications, or C++ to load multiple sources into multiple targets. To see a sample of how to do this, go to ArcOnline (www.esri.com/usersupport/arconline).



Appending Data Using the Object Loader

This method will work fine for simple features. If you are appending a large amount of data into a network (more than a few thousand features), then this method will be too time-consuming to be practical. There is an alternative method to appending data that is faster (in the case of network data).

This method essentially involves unregistering the data as versioned and dropping the network for the duration of the data loading operation. The potentially undesirable point of this method is the fact that the data must be unregistered as versioned. When unregistering data as versioned, all edits that have been made on the data that is not in the base table will be lost. The following outlines the steps to take for this method to ensure no data will be lost.

1. For each outstanding version in the database:

   - Reconcile and post the version against DEFAULT.
   - Delete the version.

2. Run compress to compress the database.

3. Unregister the data as versioned. NOTE: If you have not completed steps 1 and 2 before unregistering your data as versioned, then you will lose any edits that those versions contain.

4. Delete the geometric network.

5. Use the Simple Data Loader in ArcCatalog to load the new data to your existing feature classes.

6. Rebuild the geometric network using the Build Geometric Network wizard in ArcCatalog or ArcToolbox.

7. If you created your geodatabase schema using CASE tools, use the Schema Generation wizard to reapply the UML to the existing data to re-create the network connectivity rules and assign any class extensions. If you are not using CASE tools, then you will need to use ArcCatalog to re-create your connectivity rules.

8. Register your data as versioned and continue with production. Registering the data as versioned automatically updates the database statistics for the feature classes.

There are a number of limitations to this method that may make it necessary for you to use the first method.

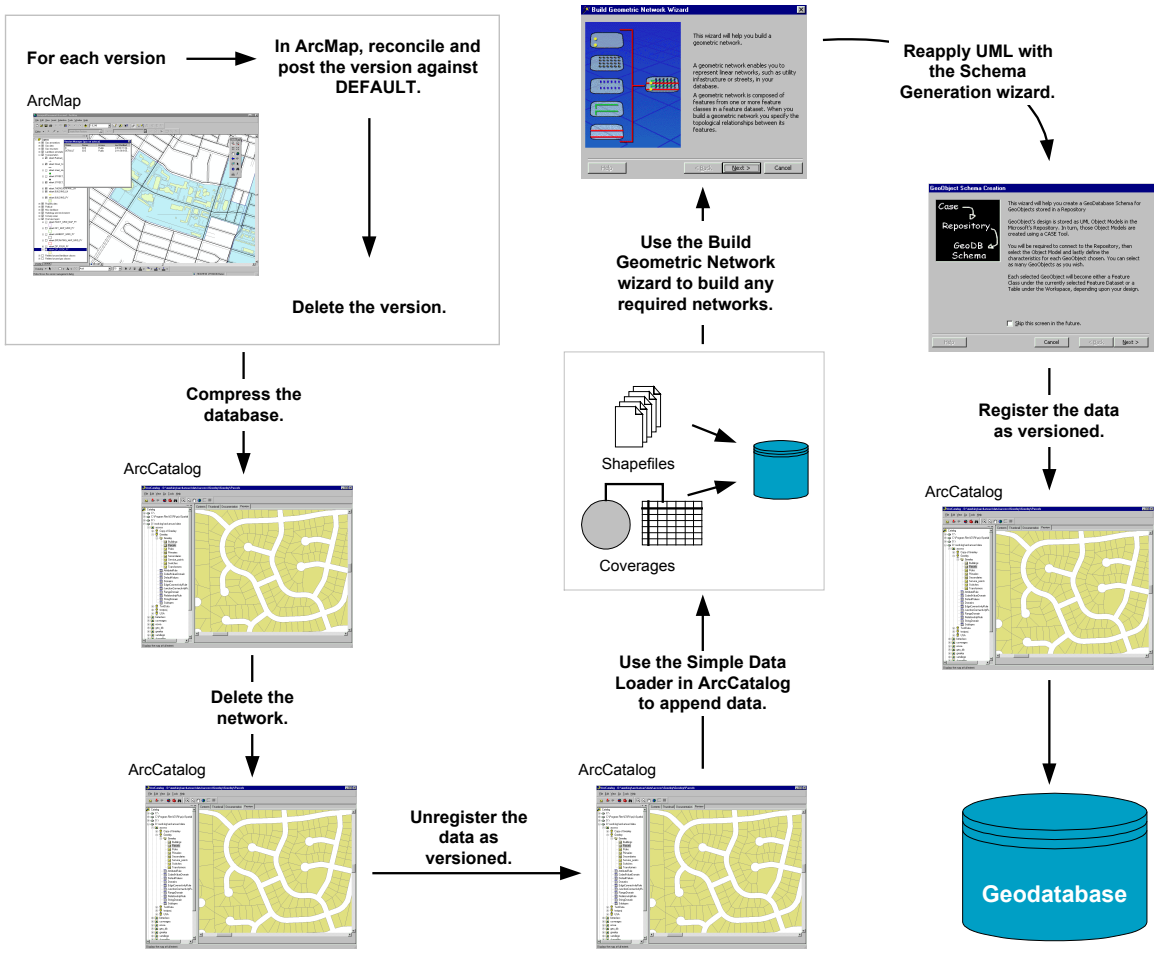
- If your network has any complex junction features with connection points and custom topology you cannot use this method since the process of batch rebuilding the network will not re-create the custom topology.

- If you have disabled any network features, they will be enabled again.

- If you have disconnected any network feature from the network, the process of rebuilding the network will reconnect all of those features.

- If any of your network feature classes have feature-linked annotation, you cannot use the Simple Data Loader to load features into it. In this case you must use the Object Loader so that all annotation features will be created as new network features are created.

J-8457



Appending Data Using the Simple Data Loader

Tuning Your Application

A good enterprise geodatabase system does not stop at the data model and database tuning. How you work with your data will play a large role in determining how well your system performs. Keep in mind that your map layers and ArcMap documents can be considered just as important a part of your geodatabase as the data and the database itself. There are many dos and don'ts associated with working with the geodatabase. This section outlines what you can and should not do when designing your applications.

Map Documents and Layer Files

Map documents and layer files are an important part of the geodatabase. You can use them to provide ArcMap environments that perform well and prevent users from performing operations against the database that will take a lot of serverside resources and bog the system down, as follows:

- *Scale suppression and overview layers.* Use scale suppression in both your ArcMap documents and layer files such that layers only draw at scales where their information can be visualized in a meaningful way. Zooming out to a scale and drawing 500,000 features is very costly on both the server and client and does not

have much benefit. This is especially important with annotation. Drawing annotation is a very costly process, and annotation tends not to be visually useful unless zoomed into a large scale.

You should also have an overview layer that has a small number of features and can be easily viewed at the full scale of the map such as a reference grid.

- *The right classes in the right documents.* Often there are a number of different types of users who do different jobs in an enterprise GIS system. Some may deal heavily with one kind of data and may never work with some of the classes in the database. Create map documents that contain the classes from the database that are required for that particular user or application. The fewer classes in the map, the less number of queries required against the server every time a pan, zoom, identify, and so forth, is performed.

  Keep in mind that when working with a class in ArcMap, any related classes should also be included in the map when navigating relationships and editing composite objects. For each related class that is not in the map, each time the relationship is navigated, either by an edit or a query, the related class needs to be opened.

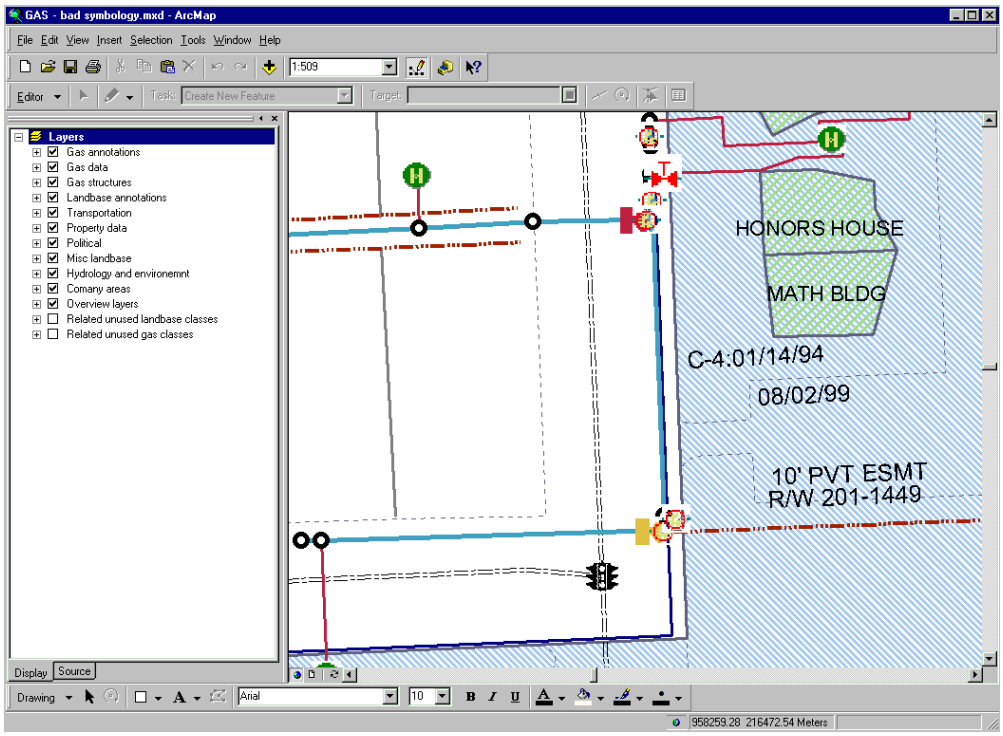
  Using the shared-edge editing tool to edit features in different feature classes with coincident boundaries can be very slow if you do not have the right classes in the map. This tool uses geometric coincidence to discover these shared-edge relationships on the fly. It will spatially search all of the feature classes in the same feature data set for this coincidence. If some of the feature classes in the data set are not in the map, they will be opened every time one of these spatial queries is executed. So when using the shared-edge editing tool, make sure that all of the feature classes in the feature data set are included in the map.

- *Picking the right symbology.* ArcInfo includes a rich collection of symbology with symbols for various industry domains. Some of these symbols are simple single-layer symbols, while others are a more complex mixture of a number of layers. The time it takes to draw features is related to the style of symbology you choose to draw them with.

  It is important to realize the function that each user in your organization is doing when choosing symbology. For example, users who read the data in your geodatabase for the purpose of making maps and performing visual analysis, such as tracing network features, will require maps with sophisticated symbology. Those users who maintain and update the data may not require such complex symbology to get their work done.

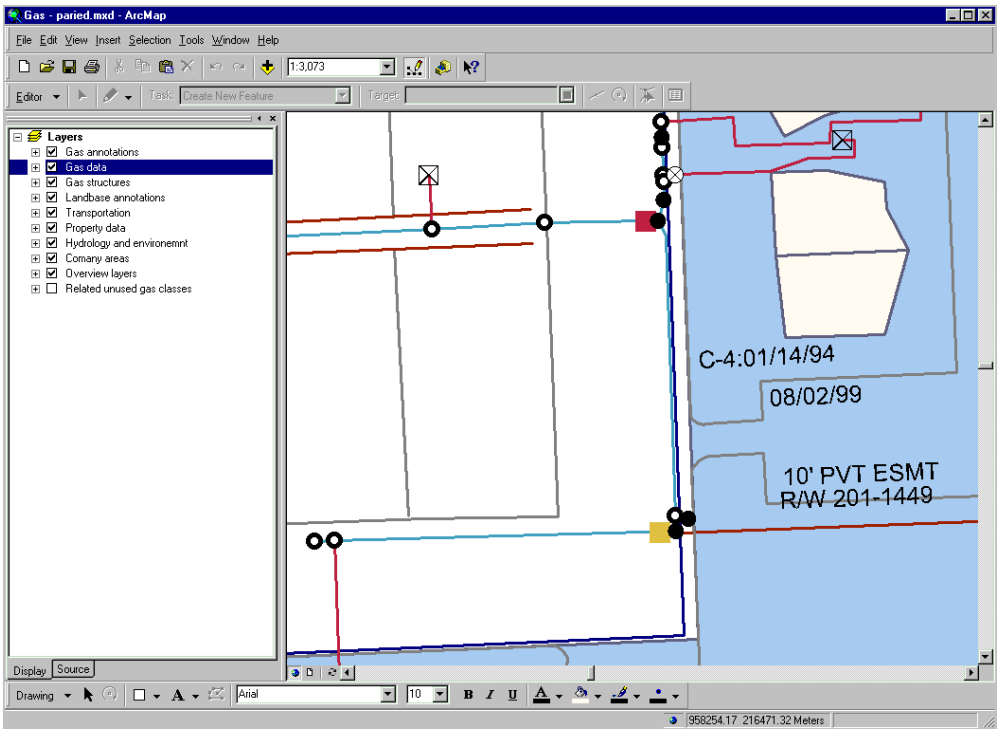
  Keep your symbology for your users (especially your editors) simple unless complex symbology is required. Part of the time it takes for a user to edit features in the geodatabase includes the redraw time of those areas of the map that are affected by the edit. If you use symbology that is unnecessarily complex, the amount of time it takes for an editor to perform a simple edit may be significantly longer because the redraw of the affected area is significantly longer.

J-8457



Complex symbology that is required for making maps may slow down the editing process.

J-8457



Simplifying your map documents by using simple symbology and reducing the number of backdrop layers can significantly improve performance.

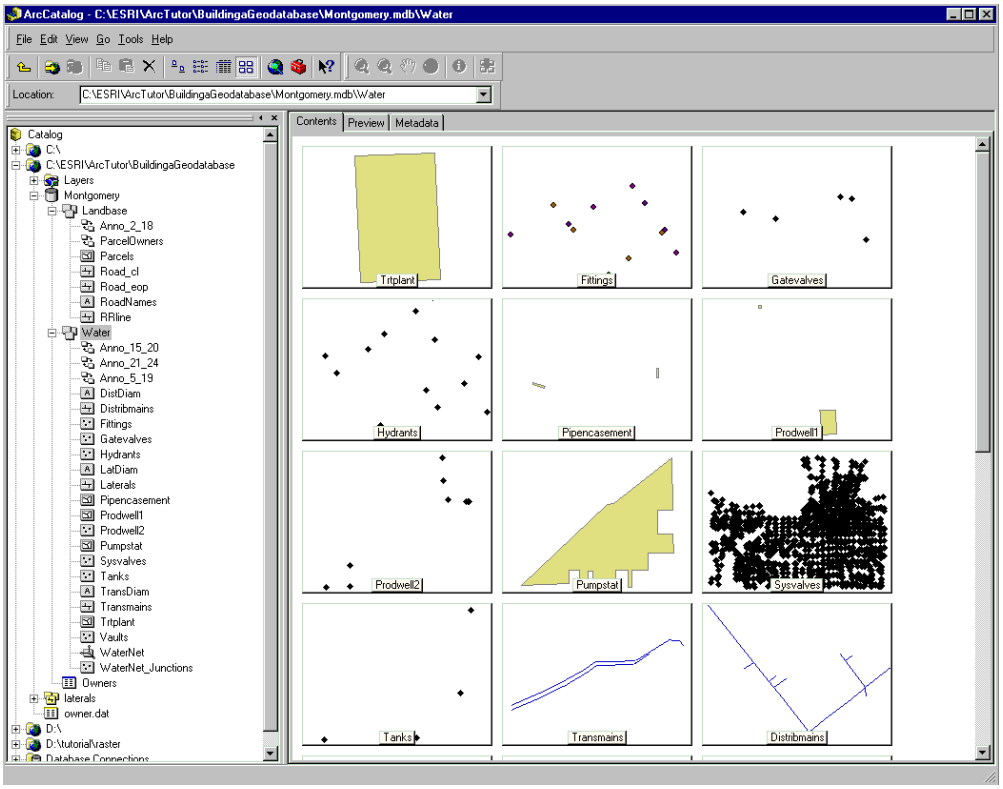
You will need to make decisions about how important are complex symbology and the number of feature classes in your map versus what performance trade-off you make to accommodate them. For example, if your editing application calls for complex symbology and numerous backdrop layers, you will need to consider whether the impact on performance is significant enough to reduce the number of layers and the amount of complex symbology you use.

The example above shows two map documents based on the same editing application. The first document has both the gas network data and the land base data and is symbolized with complex symbology. The second document has eighteen fewer land base feature classes (including eight fewer annotation classes) and is symbolized with simpler symbology. The same kinds of edit operations were on average three to five times faster when using the second document on the same database.

Working with the Data

- *Browsing data.* When using ArcCatalog to browse the contents of your geodatabase, create thumbnails for your feature classes. Using thumbnails allows you to have a visual representation of your data without having to actually open the feature class, execute a query against the server, and draw the data. This also prevents you from unnecessarily drawing large amounts of data at full scale, which would impose a heavy load on the server.

J-8457



Create thumbnails for your data so you can easily browse its contents without sending queries to the database.

- *Identifying features.* The simplest way to get information about a particular feature in your map is to use the Identify tool. However, using this tool can be very slow if you have a large number of layers in your map. When you click your map with the Identify tool, it performs a spatial query for each active layer in the map to return any features that intersect that point. These spatial queries are expensive for the server, so the response time for the Identify tool will be longer the more feature classes you have active in the map.

- *Using the edit cache.* When editing data in a geodatabase, be it network data or otherwise, one of the most important tools to use to improve performance is the edit cache. The edit cache will do what its name implies, which is cache data from the server on the client. This caching of data cuts down on the number of queries that the client needs to execute on the server during editing.

  Since editing network features requires a number of spatial queries against the server to discover geometric coincidence, when using the edit cache while editing the network, these spatial queries do not need to go against the server, are much faster, and are not as much of a load on the server. In general, when editing data in a geodatabase, especially network data, you should always use the edit cache. Using the edit cache can make your edits five times faster than without the cache.

The edit cache cannot only make editing faster, but also drawing, selecting, and identifying features that are cached. Spatial information is cached only so any attribute queries or navigating to related objects outside the cache will not benefit from the cache. You will pay the price for using the edit cache when performing undo operations while editing. Undo is significantly slower when using the edit cache.

The user must build the edit cache by clicking the Build Edit Cache command on the Editor menu. This command will cache the features in the current map extent. If you pan or zoom out of that area, you must rebuild the edit cache. The edit cache will also be cleared any time an edit operation is aborted for any reason, you save your edits, or you stop editing. In general, if during the course of editing everything suddenly slows down, you probably need to rebuild your edit cache.

- *Different accounts for different users.* Each user of the geodatabase should have their own separate database account. When you select a large number of features in an ArcSDE feature class with ArcMap, the selection is logged in a table in the database. So the more users connected to the database as the same user account, the more that this table is written to. Having a different database account for each of these users means their selections will be written to their own log tables and will not be contending with other users for writing to a common log table.

  When granting privileges to your various database users on your data, provide only the level of access rights they require to do their work. This becomes important when using the edit cache. When you build an edit cache, all of the features in the map that you have edit privileges to are cached, regardless of whether you intend to edit them or not. Restricting editing privileges for users to only those classes that they edit can significantly cut down the time it takes to build the edit cache if there are a number of addition classes in the map.

Diagnosing Your Database

You have followed the recommendations outlined in both the ArcInfo documentation and this technical paper, but your performance is still unsatisfactory. This section will discuss some of the diagnostic tools you can use to analyze your database performance and help to identify bottlenecks.

Diagnosing Performance Problems in Oracle—Where to Look?

The hard part about the database tuning process is to identify the performance problems and where to look for bottlenecks. Performance problems can be caused by one or all of the following (listed from most significant to least significant):

- *Application/Program tuning*. Usually badly tuned queries. Isolate the problem queries and try to optimize them if possible.

- *Database design and indexing*. Look for missing indexes and avoid overindexing, which can cause bad performance with inserts. Check if your tables have been analyzed and have updated statistics. Use the `sdetable` administration tool or Oracle's ANALYZE statement to analyze tables and indexes.

- *Database tuning*. Spread your data on several disks and devices to avoid I/O disk contention. Tune Oracle System Global Area (SGA) and use the performance monitor within the Enterprise Manager to check for memory usage, or write SQL scripts that do the same. The four most significant parameters to tune are DB_BLOCK_BUFFERS, SHARED_POOL_SIZE, SORT_AREA_SIZE, and DB_BLOCK_SIZE.

- *OS tuning*. Check for CPU usage and disk I/O problems, and monitor the system load and the network traffic.

ArcInfo 8 and ArcSDE 8 queries are optimized with database hints, but sometimes those may fail. Most causes for failure are incorrect statistics on the tables and indexes. Those are easily fixed by recomputing the statistics with the ANALYZE SQL statement. Missing indexes, overindexing, or a badly configured SGA might cause other failures. The following section lists how to identify these problems and resolve them.

Tools for Performance Analysis

There are several tools available to assist in tuning your database and rating your system. Tuning a database is not a simple task and is an ongoing process. If you have access to Oracle's Enterprise Manager and the performance monitor (Windows NT), then it can do the substantial part of the work. You can also use the provided SQL scripts or the tools suggested here (UNIX and Windows NT) to analyze the performance and identify bottlenecks.

Database Tuning

To determine if the SGA is correctly configured for your database and whether you have disk I/O bottlenecks, you can check the following:

- *Buffer cache hit ratio*. Lists the ratio of data in memory versus data read from disk for your system. Should be above 95 percent. If not, then tune your DB_BLOCK_BUFFERS.

```
select
 sum(decode (name, 'consistent gets',value, 0)) "Consis Gets",
 sum(decode (name, 'db block gets',value, 0)) "DB Blk Gets",
 sum(decode (name, 'physical reads', value, 0)) "Phys Reads",
 (sum(decode(name, 'consistent gets', value, 0)) +
 sum(decode(name, 'db block gets', value, 0)) -
 sum(decode(name, 'physical reads', value, 0))) /
 (sum(decode(name, 'consistent gets', value, 0)) +
 sum(decode(name, 'db block gets', value, 0)) ) * 100 "Hit Ratio"
from v$sysstat;

SQL> @hitrate;

Consis Gets DB Blk Gets Phys Reads  Hit Ratio
----------- ----------- ---------- ----------
    1387942      121840      10926 99.2763194
```

- *Data dictionary cache hit ratio*. Displays the memory allocated for accessing Oracle's underlying tables. Should be above 95 percent. If not, then tune your SHARED_POOL_SIZE.

```
select (1-(sum(getmisses)/sum(gets))) * 100 "Hit Ratio" from
v$rowcache;

SQL> @hitratio;

 Hit Ratio
----------
99.9066344
```

- *Memory sort hit ratio*. Displays the sorts performed in memory versus sorting from temporary segments on disk. Should be 99 percent or above. If not, increase the size of the SORT_AREA_SIZE and SORT_AREA_RETAINED_SIZE. Note that setting the value of SORT_AREA_SIZE too high can cause the OS to run out of memory since it is set per user process.

```
select a.value "Disk Sorts", b.value "Memory Sorts",
round(100*(b.value/decode((a.value+b.value),
0,1,(a.value+b.value))),2) "Pct Memory Sorts"
from v$sysstat a, v$sysstat b
where a.name = 'sorts (disk)'
and b.name = 'sorts (memory)';

SQL> @hitrate;

Disk Sorts Memory Sorts Pct Memory Sorts
---------- ------------ ----------------
         0        81086              100
```

- *I/O disk contention*. Use the FILE I/O manager within the Enterprise Manager to determine if reads and writes are evenly distributed. You can also get this information by running the following script. Try to balance your I/O by moving data files between disks.

```
drop table tot_read_writes;
create table tot_read_writes as select sum(phyrds) phys_reads, sum
(phywrts)phys_wrts from v$filestat;
col name format a30
col phyreads format 999,999,999
col phywrts format 999,999,999
col read_pct format 999.99
col write_pct format 999.99
select name, phyrds, phyrds * 100 / trw.phys_reads read_pct, phywrts,
phywrts * 100 / trw.phys_wrts write_pct from tot_read_writes trw,
v$datafile df,
```

J-8457

```
        v$filestat fs where df.file# = fs.file# order by phyrds desc;

SQL> @diskperc;

12 rows selected.

NAME                              PHYRDS READ_PCT      PHYWRTS WRITE_PCT
-------------------------------  --------- --------   ---------- ---------
/azteca10/oradata/feature01.dbf       2,461    45.61           1      6.67

/azteca9/oradata/business01.dbf         909    16.85           0       .00

/azteca4/oradata/system01.dbf           678    12.56           2     13.33

/azteca13/oradata/delta01.dbf           328     6.08           1      6.67

/azteca12/oradata/indxes01.dbf          196     3.63           0       .00

/azteca1/oradata/sde01.dbf               95     1.76           1      6.67

/azteca11/oradata/spatial01.dbf          48      .89           1      6.67

/azteca7/oradata/users01.dbf             17      .32           6     40.00

/azteca5/oradata/rbs01.dbf               14      .26           1      6.67

/azteca6/oradata/rbs02.dbf               13      .24           1      6.67
```

Application Tuning

The next questions to answer involve the application you are running against your database. The following describes how you can discover where your problem queries are.

Oracle's TRACE, TKPROF, and EXPLAIN PLAN provide timed statistics and execution explanations on your queries. Before enabling tracing you need to run a system script that creates an explain table in the user schema. The name of the script is UTLXPLAN.SQL. It is located in $ORACLE_HOME/rdbms/admin.

```
SQL> connect gdb/gdb;
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan;
```

Tracing Oracle can be explicitly set for the database instance or for individual sessions. To enable SQL tracing for the instance, set the following parameters in the init.ora file:

```
SQL_TRACE = TRUE
TIMED_STATISTICS = TRUE
```

To enable tracing for the current session:

```
SQL> ALTER SESSION SET TRACE = TRUE;
SQL> ALTER SESSION SET TIMED_STATISTICS = TRUE;
```

To enable someone else's session (in this case the user GDB):

```
SELECT SID,SERIAL#,OSUSER FROM V$SESSION WHERE OSUSER = 'GDB';

SIDSERIALOSUSER
924GDB

SQL> EXECUTE DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION(9,24,TRUE)
```

Tracing can even be enabled from the application if it can execute SQL statements. Here is an example of how to enable it from a VBA script in ArcMap.

```
Dim pDoc As IMxDocument
Set pDoc = ThisDocument

Dim pFeatLyr As IFeatureLayer
Dim pFeatCls As IFeatureClass
Dim pDS As IData set
Dim pWS As IWorkspace

Set pFeatLyr = pdoc.SelectedLayer
Set pFeatCls = pFeatLyr.FeatureClass
Set pDS = pFeatCls
Set pWS = pDS.Workspace

pWS.ExecuteSQL ("alter session set sql_trace = true")
pWS.ExecuteSQL ("alter session set timed_statistics = true")
```

Use the TKPROF Oracle utility to translate the TRACE output file to a readable format. You can run TKPROF against a previously created TRACE file or while the application creating the TRACE file is still running. Use the EXPLAIN=<username>/<password> option to get the execution path.

TRACE/TKPROF Example: Analyzing a selection problem from ArcMap. Trace selection before and after the table has been ANALYZED and the statistics computed.

Scenario 1: Table Not Analyzed

```
SELECT /*+ ORDERED INDEX(GDB.S156 S156_IX1) INDEX(GDB.F156 F156_UK1)
  INDEX(GAS_MAIN A156_IX1) */  OBJECTID,  SHAPE, ARC_TAG,  SPAN_TAG, NTAG,
OLDARC_TAG, OLDSPAN_TA,  ARC_TYPE,  ARC_SUBTYP,  CKT_ID,  PHASE,   FEATURE_ST,
SYMBOL,  PSHT_SYMBO,  SSHT_SYMBO,  GROUP_ID,  VER_ID, VER_END, VOLTAGE,  PHS,
CAT,  SUBCAT,  DB_TAG,  DUCT_POS,
```

J-8457

```
  WO_NO,  WO_DATE,  URD_COND,  COND_TY,  COND_DESC,  OWNER,  ERR_FLAG,
  ERR_DESC,  SVOLTS,  TRN_TAG,  ID, SHAPE.LEN  ,S_.eminx,S_.eminy,
  S_.emaxx,S_.emaxy, SHAPE.fid,SHAPE.numofpts,SHAPE.entity,SHAPE.points,
  SHAPE.rowid
FROM (SELECT /*+ INDEX(SP_ S156_IX1) */ DISTINCT sp_fid,eminx,
  eminy,emaxx,emaxy FROM GDB.S156 SP_  WHERE SP_.gx >= 55 AND SP_.gx <= 55
  AND SP_.gy >= 52 AND SP_.gy <= 52 AND SP_.eminx <= 1107356032 AND SP_.eminy
  <= 1042069900 AND SP_.emaxx >= 1106175865 AND SP_.emaxy >= 1040260311 ) S_,
   GDB.GAS_MAIN, GDB.F156 SHAPE WHERE S_.sp_fid = SHAPE.fid AND
  S_.sp_fid = GDB.GAS_MAIN.SHAPE  AND  SHAPE.FID  =
  GDB.GAS_MAIN.SHAPE


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      2      3.10       5.01          0          0         29           0
Fetch        1      6.24      10.07        116        233       2281           3
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      9.34      15.08        116        233       2310           3

Misses in library cache during parse:  1
Optimizer goal:  CHOOSE
Parsing user id:  54  (GDB)


       Rows     Execution Plan
       -------  ---------------------------------------------------
             0  SELECT STATEMENT   GOAL:  CHOOSE
             3   MERGE JOIN
             4    SORT (JOIN)
             3     NESTED LOOPS
             4      VIEW
             4       SORT (UNIQUE)
             3        INDEX (RANGE SCAN) OF 'S156_IX1' (UNIQUE)
             3      TABLE ACCESS (BY INDEX ROWID) OF 'GAS_MAIN'
             6       INDEX (UNIQUE SCAN) OF 'A156_IX1' (UNIQUE)
             3    SORT (JOIN)
         14940     TABLE ACCESS (FULL) OF 'F156'
```

The output shows a FULL TABLESCAN of the ArcSDE feature table for the layer GAS_MAIN. There is also no sign of the table being ANALYZED, so that is a probable cause for the query to fail. Running the same query after the table and indexes have been ANALYZED (and new statistics computed) gives the following result:

Scenario 2: Table Analyzed

```
azteca% sdetable -o update_dbms_stats -t gas_main -m COMPUTE -i esri_sde -
s azteca -u gdb -p gdb

ArcSDE   8.0.2    Build 334 Tue Mar 10 22:30:52 PST 2000
Attribute        Administration Utility
--------------------------------------------------
Table statistics for table gas_main and all support tables updated.
```

```
SELECT /*+ ORDERED INDEX(GDB.S156 S156_IX1) INDEX(GDB.F156 F156_UK1)
  INDEX(GAS_MAIN A156_IX1) */  OBJECTID,  SHAPE, ARC_TAG,  SPAN_TAG, NTAG, OLDARC_TAG,
OLDSPAN_TA,  ARC_TYPE,  ARC_SUBTYP,  CKT_ID,  PHASE,   FEATURE_ST,  SYMBOL,
PSHT_SYMBO,  SSHT_SYMBO,  GROUP_ID,  VER_ID, VER_END, VOLTAGE,  PHS,  CAT,  SUBCAT,
DB_TAG,  DUCT_POS,
  WO_NO,  WO_DATE,  URD_COND,  COND_TY,  COND_DESC,  OWNER,  ERR_FLAG,
  ERR_DESC,  SVOLTS,  TRN_TAG,  ID, SHAPE.LEN  ,S_.eminx,S_.eminy,
  S_.emaxx,S_.emaxy, SHAPE.fid,SHAPE.numofpts,SHAPE.entity,SHAPE.points,
  SHAPE.rowid
FROM (SELECT /*+ INDEX(SP_ S156_IX1) */ DISTINCT sp_fid,eminx,
  eminy,emaxx,emaxy FROM GDB.S156 SP_  WHERE SP_.gx >= 55 AND SP_.gx <= 55
  AND SP_.gy >= 52 AND SP_.gy <= 52 AND SP_.eminx <= 1107356032 AND SP_.eminy
  <= 1042069900 AND SP_.emaxx >= 1106175865 AND SP_.emaxy >= 1040260311 ) S_,
   GDB.GAS_MAIN, GDB.F156 SHAPE WHERE S_.sp_fid = SHAPE.fid AND
  S_.sp_fid = GDB.GAS_MAIN.SHAPE  AND  SHAPE.FID =  GDB.GAS_MAIN.SHAPE
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.03	0.03	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	0.00	0.00	0	21	0	3
total	3	0.03	0.03	0	21	0	3

```
Misses in library cache during parse:  1
Optimizer goal:  CHOOSE
Parsing user id:  54  (GDB)

Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL:  CHOOSE
      3   NESTED LOOPS
      4    NESTED LOOPS
      4     VIEW
      4      SORT (UNIQUE)
      3       INDEX (RANGE SCAN) OF 'S156_IX1' (UNIQUE)
      6     TABLE ACCESS   GOAL:  ANALYZED (BY INDEX ROWID) OF
                'GAS_MAIN'
```

```
6      INDEX   GOAL:  ANALYZED (UNIQUE SCAN) OF 'A156_IX1' (UNIQUE)

3    TABLE ACCESS   GOAL:  ANALYZED (BY INDEX ROWID) OF 'F156'
6     INDEX   GOAL:  ANALYZED (UNIQUE SCAN) OF 'F156_UK1' (UNIQUE)
```

OS Tuning

There are a number of OS tools you can use to determine a problem with the ArcSDE server's memory, CPUs, and disk configuration.

On UNIX, the virtual memory consists of the RAM, disks, and swap space. When your server runs out of physical memory, it has to page or swap. Paging is moving a page of memory to the swap device; swapping is transferring an entire process from RAM to a swap device. Both paging and swapping result in performance degradation.

Part of the tuning process is tuning the OS and figuring out CPU, memory, and disk I/O contentions. For Windows NT there is the task manager and the performance monitor. The performance monitor has more features and can monitor memory, processes, disks, and so forth. On UNIX, there are few tools that can be used to find performance problems. Use the tools to check how loaded your system is and the usage of the virtual memory.

Memory and CPU

The sar command can identify many different performance issues using the different switches. Using the –u switch lists out the CPU usage.

```
azteca% sar -u 2 2

SunOS azteca 5.6 Generic_105181-09 sun4u    03/10/00

16:10:37    %usr    %sys    %wio   %idle
16:10:39       1       1       6      92
16:10:41       0       1       5      93

Average        1       1       6      92
```

A low %idle time means high load. Look for CPU intensive processes running on the system. Use the top utility to list the "most CPU expensive" processes. A high %iowait indicates disk contention problems.

```
load averages:  0.18,  0.20,  0.21
16:19:39
241 processes:  230 sleeping, 10 stopped, 1 on cpu
CPU states:  89.8% idle,  1.0% user,  2.5% kernel,  6.7% iowait,  0.0%
swap
Memory:  2048M real, 34M free, 170M swap in use, 854M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
26486 sde        1   0    0 2120K 1672K cpu1    0:03  1.34% top
  649 smtp       1  58    0   21M   16M sleep  47.2H  0.76% esd
  314 smtp       6   0    0 3536K 1672K sleep 183:55  0.19% automountd
```

```
  648 smtp       1  58    0 6120K 2472K sleep 217:52  0.08% jre
  892 smtp       1  58    0 1600K  616K sleep  27:28  0.04% IPXd
  260 smtp       1  58    0 2376K 1264K sleep  76:29  0.04% ypserv
  651 smtp       1  58    0 9608K 6112K sleep 104:11  0.02% esd
26526 smtp       1  58    0 1560K 1200K sleep   0:00  0.02% in.telnetd
  406 smtp       5  58    0 2296K  832K sleep  29:02  0.02% vold
20751 smtp       4  52    2   12M 2352K sleep   0:20  0.02% arcrqmgr
29562 oracle    11  58    0  167M  150M sleep 156:55  0.01% oracle
  652 kentlroo   1  58    0 7048K 4936K sleep  90:07  0.01% esd
  394 kentlroo   1  58    0  904K  408K sleep   0:53  0.01% utmpd
  459 kentlroo   1  58    0 1592K  744K sleep   0:57  0.01% nfsd
15846 sde804     1  58    0 2128K 1072K sleep   7:14  0.01% ESRI
```

Memory status can also be obtained by using vmstat.

```
procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr s0 s2 s2 s2   in   sy   cs us sy id
 9 0 0 2649056 32120  1   5  6 10 10  0  0  1  0  0  0 1160 25120 4794 93 6  1
```

The above result shows a heavy loaded CPU with 1 percent idle time and nine processes waiting (r) to be run.

mpstat reports per-processor statistics and lists the activities on each of the processors on a multi-CPU system.

```
azteca% mpstat 5 2
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   23   1  854    33   32   99    1    4    5    0   650    4   3  17  77
  1   22   2  905     3    1  105    1    5    5    0   813    4   3  17  76
  4   21   0  762   422  214   86    1    4    7    0   517    3   4  17  76
  5   20   1  904    35   33   99    1    4    6    0   988    4   3  17  76
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   29   0  100    15   14   81    0    4    3    0   139    0   1  10  89
  1    2   0   86     2    1   77    1    3    3    0   982    3   1  10  86
  4    0   0  173   303  103  109    0    5    3    0   110    0   1  10  88
  5   69   0 4367    11   10   62    0    4    3    0   139    1   4   9  85
```

If your CPU is heavily used then you will see some performance degradation. The work-around could be to

- Get faster processors.
- Get more CPUs.

Heavy CPU usage can also be caused by I/O contentions and bad queries that go to disk for full tablescans. Adding more CPUs will not solve the problem, and you also need to analyze if there is an I/O problem on your system.

Disk I/O

The final step when analyzing the server is to check the I/O and locate potential bottlenecks.

sar with –d switch lists disk activities. Check for the %busy values and how the load balance is between disks.

```
azteca% sar -d 5 2

SunOS azteca 5.6 Generic_105181-09 sun4u    01/26/00

16:41:39   device        %busy   avque   r+w/s  blks/s  avwait  avserv

ssd9,e           0        0.0       0       0     0.0     0.0
ssd10           10        0.1       7     119     0.0    16.3
ssd10,c          0        0.0       0       0     0.0     0.0
ssd10,d          0        0.0       0       0     0.0     0.0
ssd10,e         10        0.1       7     119     0.0    16.3
ssd12           11        0.5       5      81     0.0    89.5
ssd12,c          0        0.0       0       0     0.0     0.0
ssd12,d          0        0.0       0       0     0.0     0.0
ssd12,e         11        0.5       5      81     0.0    89.5
ssd13            3        0.0       2      26     0.0    16.6
```

If you are experiencing disk I/O contentions, then check if your data files are evenly distributed across devices. Use multiple "small" disks devices instead of using a single large volume, and distribute the data files/tablespaces across devices.

Case Study—A Gas Geodatabase

For our case study we took a gas and land base database that was designed in UML, for which we had data for a number of the feature classes stored in coverages and INFO tables. The challenge of this case study was to take the database design and existing data and build a geodatabase in Oracle, then create application documents to work with the geodatabase for editing and viewing.

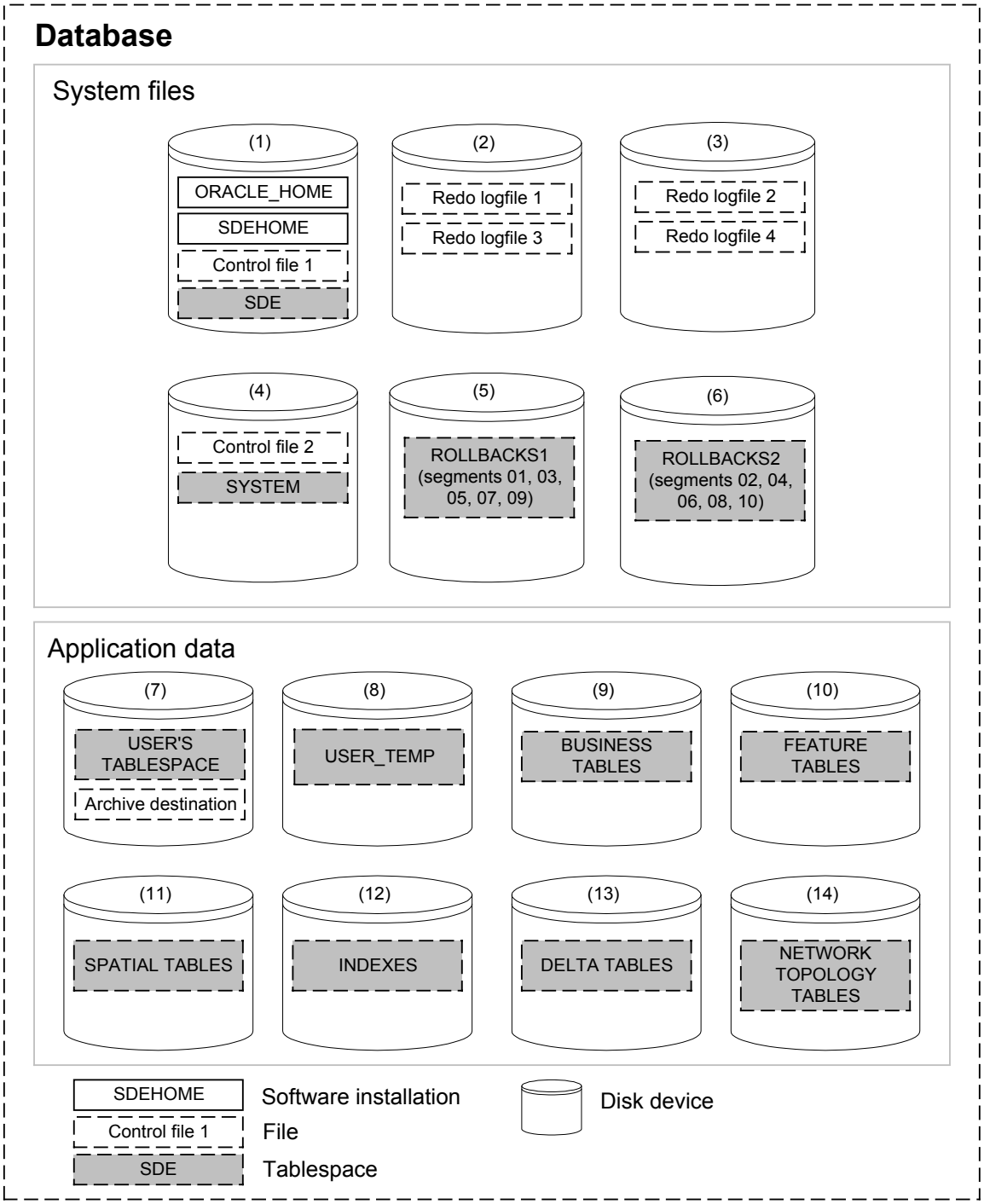
Building the Geodatabase

The first step is to create the Oracle instance and database. Creating the instance and database requires understanding how you will be distributing the instance's control files, redo log files, rollback segments, system tablespace, and all the user's tablespaces for ArcSDE and the geodatabase. But first, let's look at our organization's hardware since this is the determining factor as to how we will design and lay out our Oracle database configuration.

Our server is a Solaris Enterprise 4000 with four CPUs, 2 GB of memory, 2 GB of swap space, and fourteen 9 GB disk drives. The high number of disk drives facilitates distributing our Oracle instance and tablespaces across multiple devices to reduce disk I/O contention.

The following diagram shows how we intend to distribute our instance and tablespaces when we create the database.

J-8457



Disk (1) will contain Oracle and ArcSDE executables, the first of two control files, and the SDE tablespace used for our geodatabase's metadata.

Disk (2) and disk (3) will contain the redo log files. The redo log files are interleaved between each disk to avoid I/O contention when the archive process starts reading the log file after each log file switch.

Disk (4) will contain the second control file, for backup in case disk (1) fails, and the database's system tablespace. Placing these two files on the same device should not create much disk I/O contention.

Disk (5) and disk (6) will contain the rollback tablespaces each with five segments. Notice we will also interleave our rollback segments between each tablespace to help distribute each transaction between devices.

Disk (7) will contain the user's tablespace that is used for log files for large selection sets and the archive destination. We placed these two files together because of the infrequency of activity. User activity is solely based on your application's use of log files. The archive process will only create an archive file after each log file switch, which should not be that frequent.

Disk (8) will contain the user's temporary tablespace. Keep in mind the temporary tablespace is used by Oracle to build indexes, generate statistics, and perform sort operations when additional buffers are required. This can be an active tablespace based on version queries performing frequent sort operations when needed.

Disks (9) through (14) will contain all of the database's tablespaces for storing business tables, feature tables, spatial tables, indexes, version delta tables, and network topology tables. If we were fortunate enough to have additional devices we could create multiple tablespaces for distributing our tables, but unfortunately, the case study is limited to only fourteen disk drives.

Now that we have determined how we want to distribute our files across our disks, let's install Oracle and build the database. This document is not intended to demonstrate the necessary steps to install Oracle—please reference Oracle's documentation. Also, our instance's init.ora file can be found in the appendix with parameter descriptions.

You have the option to create your database either by using Oracle's utilities and scripts or by the command line in Server Manager. We have provided the command line syntax to assist in following the procedures we performed in creating our database.

Begin by starting Server Manager at a command prompt and then proceed in creating the database.

```
azteca% svrmgrl

Oracle Server Manager Release 3.1.6.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation.  All Rights Reserved.
```

```
Oracle8i Enterprise Edition Release 8.1.6.0.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production

SVRMGR>
CONNECT INTERNAL
STARTUP NOMOUNT PFILE =
/azteca1/ora816/admin/case_study/pfile_init816.ora.
CREATE DATABASE "case_study" CONTROLFILE REUSE
   MAXDATAFILES 40
   MAXINSTANCES 1
   MAXLOGFILES 32
   CHARACTER SET US7ASCII
   NATIONAL CHARACTER SET US7ASCII
DATAFILE '/azteca2/oradata/case_study/system01.dbf' SIZE 200M REUSE
LOGFILE '/azteca3/oradata/case_study/rdo1.dbf' SIZE 250M REUSE,
'/azteca4/oradata/case_study/rdo2.dbf' SIZE 250M REUSE,
'/azteca3/oradata/case_study/rdo3.dbf' SIZE 250M REUSE,
'/azteca4/oradata/case_study/rdo4.dbf' SIZE 250M REUSE;
```

Certain Oracle SQL script files must be run in order to create data dictionary functions. They can be run during or after database creation.

```
@$ORACLE_HOME/rdbms/admin/catalog.sql
```

Next, use the CREATE ROLLBACK SEGMENT command to create your initial rollback segment and the CREATE TABLESPACE command for your rollback tablespaces.

```
CREATE ROLLBACK SEGMENT r0 TABLESPACE SYSTEM
STORAGE (INITIAL 16K NEXT 16K MINEXTENTS 2 MAXEXTENTS 20);
ALTER ROLLBACK SEGMENT r0 ONLINE;

CREATE TABLESPACE rbs1 DATAFILE '/azteca5/oradata/case_study/rbs101.dbf'
SIZE 50M REUSE DEFAULT STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 1
MAXEXTENTS 100 PCTINCREASE 0);

CREATE TABLESPACE rbs2 DATAFILE '/azteca6/oradata/case_study/rbs201.dbf'
SIZE 50M REUSE DEFAULT STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 1
MAXEXTENTS 100 PCTINCREASE 0);
```

Next, create the temporary tablespace and rollback segments. Always set the temporary tablespace's initial extent value as two times the size of the SORT_AREA_SIZE parameter. The reasoning is if the sort operation is not going to fit in memory (the value of SORT_AREA_SIZE), then the sort will need to allocate the necessary extents in the temporary tablespace to perform the operation. Therefore, the initial extent should be a minimum of twice the value of SORT_AREA_SIZE to minimize dynamic extension

during the sort operation. Also, make sure to interweave the rollback segments between the two rollback tablespaces.

```
CREATE TEMPORARY TABLESPACE temp TEMPFILE
'/azteca8/oradata/case_study/temp01.dbf'SIZE 2M AUTOEXTEND ON NEXT 1M
MAXSIZE UNLIMITED;
CREATE ROLLBACK SEGMENT r01 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs1;
CREATE ROLLBACK SEGMENT r02 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs2;
CREATE ROLLBACK SEGMENT r03 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs1;
CREATE ROLLBACK SEGMENT r04 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs2;
CREATE ROLLBACK SEGMENT r05 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs1;
CREATE ROLLBACK SEGMENT r06 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs2;
CREATE ROLLBACK SEGMENT r07 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs1;
CREATE ROLLBACK SEGMENT r08 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs2;
CREATE ROLLBACK SEGMENT r09 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs1;
CREATE ROLLBACK SEGMENT r010 STORAGE (INITIAL 256K NEXT 256K MINEXTENTS 20
MAXEXTENTS 100) TABLESPACE rbs2;
```

Now bring each rollback segment online, drop the initial rollback segment r0, and alter the SYS and SYSTEM users temporary tablespace.

```
ALTER ROLLBACK SEGMENT r01 ONLINE;
ALTER ROLLBACK SEGMENT r02 ONLINE;
ALTER ROLLBACK SEGMENT r03 ONLINE;
ALTER ROLLBACK SEGMENT r04 ONLINE;
ALTER ROLLBACK SEGMENT r05 ONLINE;
ALTER ROLLBACK SEGMENT r06 ONLINE;
ALTER ROLLBACK SEGMENT r07 ONLINE;
ALTER ROLLBACK SEGMENT r08 ONLINE;
ALTER ROLLBACK SEGMENT r09 ONLINE;
ALTER ROLLBACK SEGMENT r010 ONLINE;

ALTER ROLLBACK SEGMENT r0 OFFLINE;
DROP ROLLBACK SEGMENT r0;

ALTER USER sys TEMPORARY TABLESPACE temp;
ALTER USER system TEMPORARY TABLESPACE temp;
```

Next, run the following database scripts.

```
@$ORACLE_HOME/rdbms/admin/catproc.sql
@$ORACLE_HOME/rdbms/admin/caths.sql
@$ORACLE_HOME/rdbms/admin/otrcsvr.sql

CONNECT system/manager
@/azteca1/ora815/sqlplus/admin/pupbld.sql
```

Create the SDE tablespace and SDE user, required for ArcSDE, and the remaining database tablespaces.

```
CREATE TABLESPACE sde DATAFILE '/azteca1/oradata/case_study/sde01.dbf' SIZE 40M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);
CREATE USER sde IDENTIFIED BY sde DEFAULT TABLESPACE sde TEMPORARY TABLESPACE temp QUOTA
UNLIMITED ON sde QUOTA UNLIMITED ON temp;

GRANT CONNECT, RESOURCE TO sde;

CREATE TABLESPACE users DATAFILE '/azteca7/oradata/case_study/users01.dbf' SIZE 2048M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

CREATE TABLESPACE business DATAFILE '/azteca9/oradata/case_study/business01.dbf' SIZE 2048M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

CREATE TABLESPACE feature DATAFILE '/azteca10/oradata/case_study/feature01.dbf' SIZE 2048M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

CREATE TABLESPACE spatial DATAFILE '/azteca11/oradata/case_study/spatial01.dbf' SIZE 2048M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

CREATE TABLESPACE indxes DATAFILE '/azteca12/oradata/case_study/indxes01.dbf' SIZE 2048M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

CREATE TABLESPACE delta DATAFILE '/azteca13/oradata/case_study/delta01.dbf' SIZE 2048M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

CREATE TABLESPACE network DATAFILE '/azteca114/oradata/case_study/network01.dbf' SIZE 2048M
DEFAULT STORAGE (INITIAL 512K NEXT 512K MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);
```

Then proceed to create each database user and grant connection privileges.

```
CREATE USER andy IDENTIFIED BY freeman DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp QUOTA
UNLIMITED ON users QUOTA UNLIMITED ON temp;

GRANT CONNECT, RESOURCE TO andy;

CREATE USER …
```

Next, install the ArcSDE application server. Please reference your ArcSDE installation documentation for additional information. When starting the ArcSDE instance for the first time, it will create several tables for managing your spatial data. All the tables will be created in the SDE user's default tablespace. In our case, the SDE tablespace located on disk (1).

```
sdemon –o start –p sde
```

ArcSDE will also install several stored procedures in the database. To improve performance Oracle recommends pinning stored procedures in the shared pool to avoid being paged out over time. This will avoid the shared pool reloading the object into memory and parsing the procedure. You will first need to execute the $ORACLE_HOME/rdbms/admin/dbmspool.sql as the SYS user. Next, pin each of the SDE user packages.

```
execute dbms_shared_pool.keep ('SDE.LAYERS_UTIL');
execute dbms_shared_pool.keep ('SDE.LOCATOR_UTIL');
execute dbms_shared_pool.keep ('SDE.METADATA_UTIL');
execute dbms_shared_pool.keep ('SDE.RASTERCOLUMNS_UTIL');
execute dbms_shared_pool.keep ('SDE.REGISTRY_UTIL');
execute dbms_shared_pool.keep ('SDE.SDE_UTIL');
execute dbms_shared_pool.keep ('SDE.SREF_UTIL');
execute dbms_shared_pool.keep ('SDE.VERSION_UTIL');
```

Once the ArcSDE instance has started, you will need to create the geodatabase metadata tables by running the gdbs executable. But first, prior to running the command, you will need to modify the dbtune.sde file.

The dbtune.sde file is a configuration file located in the $SDEHOME/etc directory on UNIX servers and in the %SDEHOME%\etc folder on Windows NT servers. It is a configuration file that contains Oracle table and index creation parameters. These parameters allow the ArcSDE service to communicate to the Oracle server such things as the tablespace a table or index will be created in, the size of its initial and next extent, and other parameters that can be set on either the CREATE TABLE or CREATE INDEX statements.

To better organize our database objects, we want to create the gdbs metadata tables in the SDE tablespace. Therefore, edit the dbtune.sde file and set the parameters for the A_TBLSP and INDEX_TABLESPACE to SDE and accept the default values for the remaining parameters. Then run the $SDEHOME/tools/gdbs command.

```
Usage:  gdbs <CREATE | ADD | REMOVE | UPGRADE> <server> <instance> <SDE
DBA password> [database]

azteca% gdbs create azteca esri_sde sde
```

Now that the SDE tablespace and geodatabase are created, we can continue to create our schema and load the existing coverage data. Using the Schema Generation wizard in ArcCatalog we created the schema for our UML data model design in the geodatabase. The wizard allows you to specify a dbtune file keyword for each feature class and table that it creates. In this case, we used the DEFAULTS dbtune file keyword for each of the classes and tables. But prior to running the wizard we have to modify the dbtune.sde file. This time change the A_TBLSP parameter to business, F_TBLSP to feature, S_TBLSP to spatial, and INDEX_TABLESPACE to indexes and accept the default values for the remaining parameters. When each feature class is created, the business table will be located in the business tablespace, the feature table will be located in the feature tablespace, and so forth. This will help distribute disk I/O across multiple devices when the data is accessed by the application.

Next, we used ArcCatalog to create all the annotation classes for our database. And since we have existing coverage data that needs to be loaded into the new database schema, we deleted the two geometric networks that were originally created by the schema wizard. By deleting the geometric networks we can use the Simple Data Loader to import the coverage data.

After importing each coverage into our geodatabase feature classes, we need to verify in SQL*Plus the number of segment extents created for each object. This is a common activity database administrators should perform to reduce the number of extents on disk. Accessing a table in one contiguous extent is faster then accessing a table spanned across multiple extents. The following SQL statement identifies the database objects (tables or indexes) that have multiple extents.

```
SELECT SEGMENT_NAME "TABLE", TABLESPACE_NAME "TABLESPACE", INITIAL_EXTENT,
NEXT_EXTENT, EXTENTS FROM USER_SEGMENTS;

TABLE                     TABLESPACE     INITIAL_EXTENT NEXT_EXTENT    EXTENTS
------------------------- -------------- -------------- ------------ ----------
GAS_MAINS                 BUSINESS              41779200      131072          2
FLOW_CONTROL_DEVICES      BUSINESS               2949120      131072          1 VALVES
BUSINESS                  4751360                 131072           1
F46                       FEATURE                2097152      131072          2
S46                       SPATIAL                1146880      131072          1
```

If you discover that you have several objects that contain an excess number of extents, you should fix this problem prior to using your database to avoid poor performance. To correct the problem you have two options: either delete the layer in ArcCatalog and modify the dbtune.sde file parameters for the initial and next extent and reimport the data, or use Oracle's Export/Import utility. Using Oracle's utility may be the better option, guaranteeing success as opposed to trial and error setting the initial and next extent parameters.

When using Oracle's Export/Import utility make sure to export the tables with the compress option. Export will also export all the table's indexes. Then you will have to

drop the table prior to importing the export file. During the import process Oracle will compress the data into one extent.

Now that we have imported all of our coverage data and cleaned up the number of object extents we can manually build the geometric networks and reapply our UML.

If you have any geometric networks and your database design includes a tablespace for the network topology tables, modify the dbtune.sde file's A_TBLSP parameter value to the name of the tablespace. When the geometric network is built, the topology tables created in the database will be located in the specified tablespace. This can potentially help alleviate any disk I/O problems between applications accessing the geometric network and other objects located in different tablespaces and on different disks.

Before registering any feature classes or feature data sets as versioned it is also recommended that you perform any batch updates directly to the tables using SQL. For example, one instance in which you may need to do this is setting feature-linked annotation. If you have used the Convert Coverage Annotation tool in ArcMap you may need to set the foreign key value for the featureid field in the annotation class (see the appendix for a SQL example).

When you are finally ready to register your data as versioned, you will have to once again change the storage parameters in the dbtune.sde file. When a feature class or feature data set is registered as versioned, additional delta tables are created. Change the A_TBLSP parameter value to the name of the tablespace that will store the delta tables.

If your organization is not dependent on your spatial features area and length values or you manage these values as additional attribute columns on the business table, drop these indexes from the feature table. The indexes will be named F<#>_AREA_IX2 and F<#>_LEN_IX3, where # is the feature table identifier. Having fewer indexes will increase performance when creating new features. This is a general rule that applies to all tables—the fewer the indexes, the faster the creation will be for new rows in the table.

```
DROP INDEX F84_AREA_IX2;
DROP INDEX F84_LEN_IX3;
```

The next step is to create thumbnails for browsing in ArcCatalog. Preview each feature class and zoom into an area containing a representative set of features, and create thumbnails. These thumbnails are stored as part of the metadata for that feature class in the database.

Tuning Our Application

For the purposes of keeping this simple, let's assume that there are three types of users for our system: those that edit gas network features, those that edit land base features, and those that edit no features but simply view and query the data. Each user will have different permission levels on the various data sets within the geodatabase, and each will require different map documents to do their work.

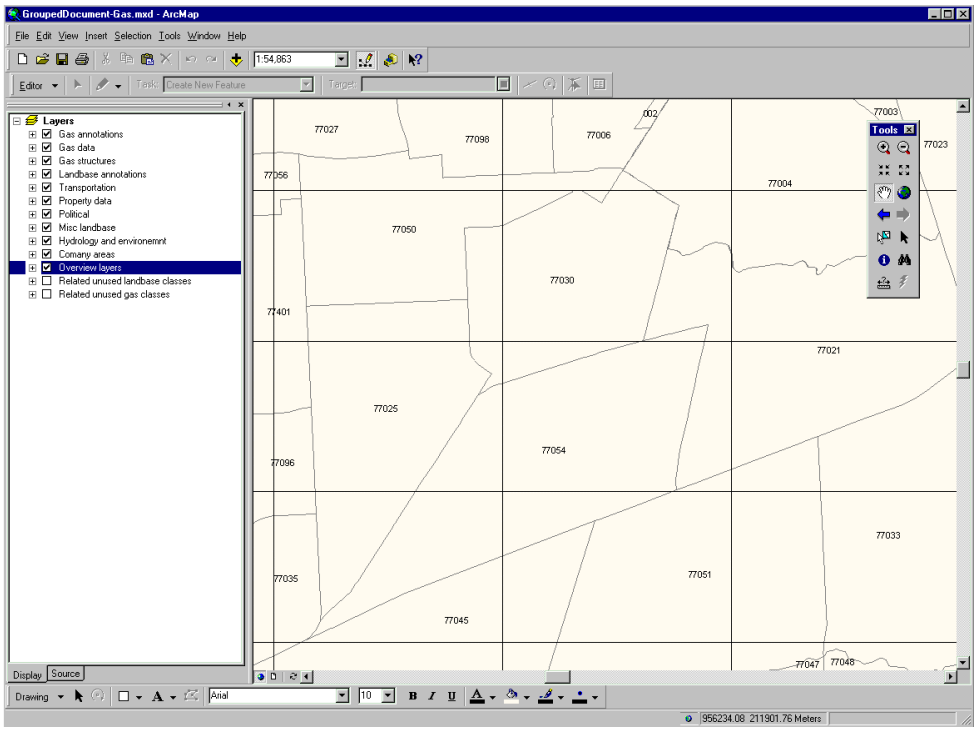
When setting permissions, we gave users SELECT/UPDATE/INSERT/DELETE access to only those data sets that they are editing and SELECT permissions to those other data

sets that they require for backdrops, and so forth, while they are editing. Users who do not do any editing require only SELECT permissions on each data set that they need read access to.

Setting these different permissions for editing will make building the edit cache, which is critical for performance, much quicker. Since only those features that are editable are cached, the unnecessary caching of features that are not going to be edited is eliminated.

Once the permissions were set up as described, we set out to make our map documents and layer files. Each map document contained only those feature classes that the user required for their work, plus any other related classes. For example, the gas network editing users had all of the gas feature classes and annotation classes in the map. In addition, they had a collection of the land base data for use as reference data.

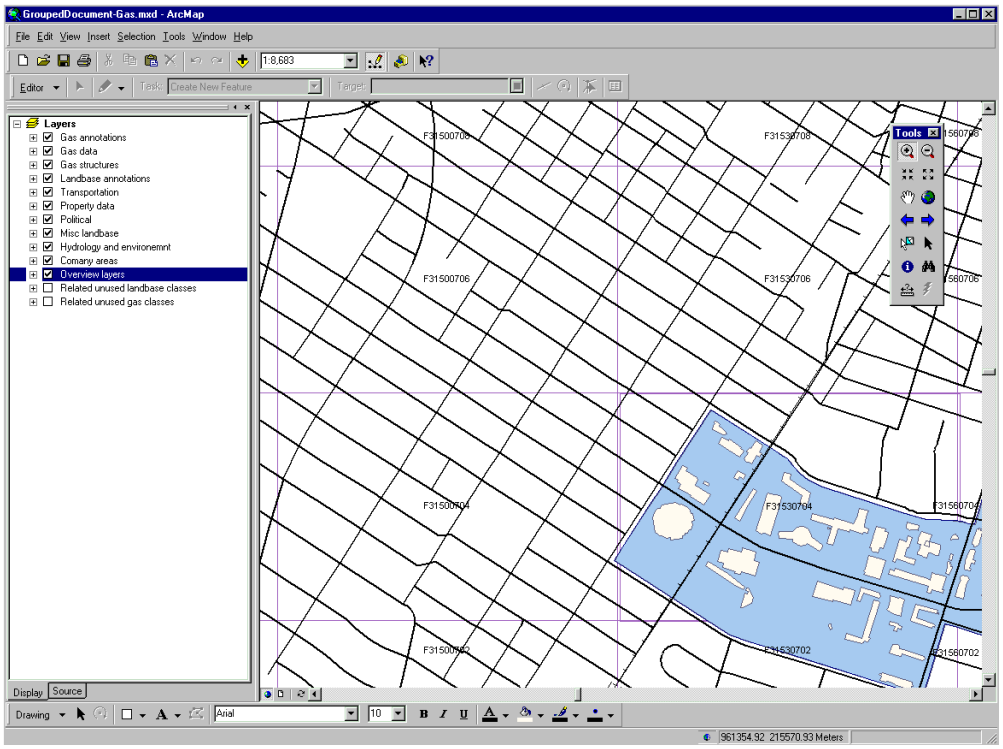
Overview feature classes were identified for each map document. These consisted of ZIP Code boundaries and map grids for the study area. The ZIP Codes are the only features that draw at full scale.



Overview layers that draw are used for large-scale drawing. This keeps the load on the server down for panning and zooming around your entire data area.

Each other feature class is given a minimum scale drawing threshold, such that they only draw when their data is meaningful. Maximum thresholds were also set for feature classes with very large features, such that when zoomed into a scale that was smaller than a single feature in a class, it was no longer drawn (and therefore no longer queried against).

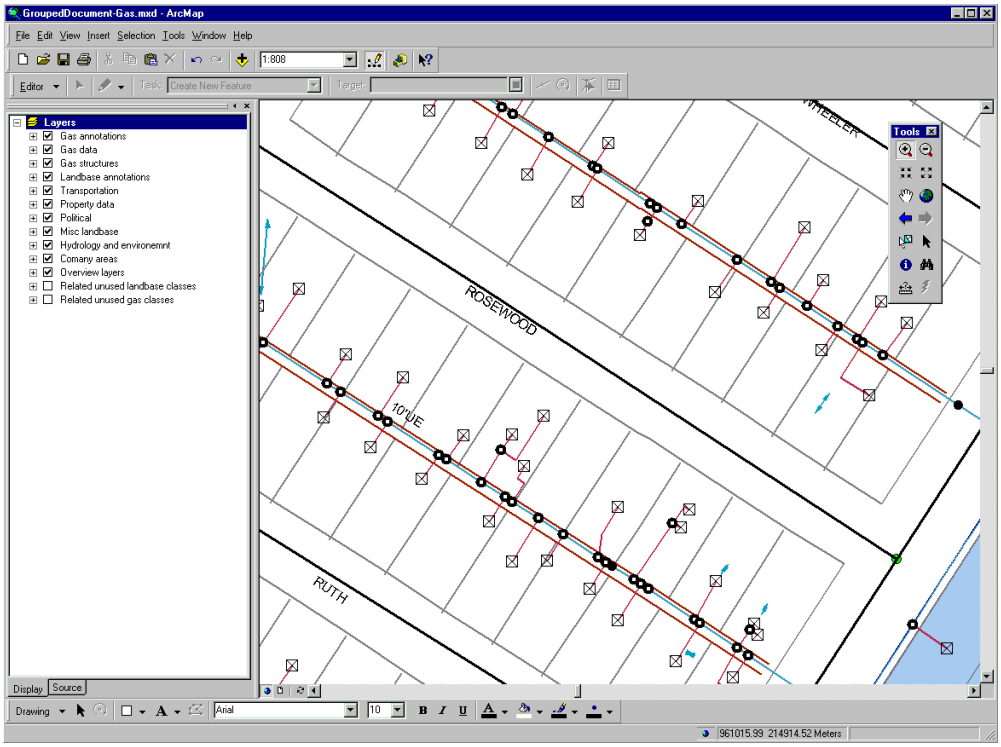
J-8457



Scale suppression is set to turn off large feature layers and turn on finer detail reference layers as we zoom in farther and our browsing becomes more detailed.

For the most detailed data, such as the gas network fittings and annotation, these feature classes were given scale suppression such that they only draw when zoomed in at a scale where users would actually edit the network. All annotation feature classes were given scale suppression to a point where their text could actually be read.

J-8457



Our most detailed data and annotation is suppressed until we are at a scale where they can be visualized and worked with in a meaningful way.

The result of these efforts to make application-intelligent map documents is that navigating around the data was much quicker and required only necessary data to be returned from the server and drawn in the map. The benefits were twofold: fewer features coming back from the server meant faster drawing times at all scales, and fewer feature classes being queried and less data being read on the server meant less server load for each user.

The time it takes to open these map documents is directly related to the number of classes that need to be opened, based on what is in your map. If your map contains thirty feature classes, then thirty feature classes need to be opened, in addition to any other feature classes that are in feature data sets with those classes that are not in the map. By organizing our feature classes into functional use feature data sets we avoided opening extra classes that we did not need for our application.

Editing the data was always done with the edit cache. This improved the editing performance of the network data six times over editing without the cache. Caching also made redrawing areas within the cache considerably faster than with no features cached. Short editing transactions against DEFAULT did not impose a large load on the database when edits were saved. However, after creating new versions for longer transactions, the reconcile process, especially in the presence of network data, did impose a heavy load on the server. For that reason, reconcile and posting versions back to DEFAULT with a

large number of edits was done when there were not a large number of users on the system.

Conclusion

Implementing a multiuser GIS system with ArcInfo and ArcSDE requires planning and experimentation to be successful. Each stage of the process, from the data model to database tuning and application design, has a significant impact on how well your system performs.

When modeling your geodatabase, keep in mind the cost of things like geometric networks and relationship classes to ensure that you are not starting with a design that will not perform well regardless of how your database is tuned. Also, when working with your geodatabase, use map documents that will not allow your users to accidentally perform a heavyweight operation that loads down the server and does not serve much value.

Following the guidelines contained in this document will help you to create a successful multiuser GIS system with ArcInfo and the geodatabase.

References

Building a Geodatabase, Andrew MacDonald, 1999 ESRI Press

Using ArcMap, Michael Minami, Michelle Sakala, Jennifer Wrightsell, 1999 ESRI Press

Modeling Our World, Michael Zeiler, 1999 ESRI Press

Oracle online documentation

Appendix A

The following SQL script can be run after importing data and coverage annotation to relink the features and their feature-linked annotation features. Note that this script should only be run after the features and the annotation are loaded into the geodatabase and before the data is registered as versioned.

```
-- PL/SQL Script to set feature linked annotation
-- Requires setting the Origin Feature Class,
-- target Annotation Class and the fields
-- used to join the classes.  You may have to
-- change your origin feature class row_id
-- field, if it is not OBJECTID.

-- Commit interval set to 1000.

-- Run this script ONLY prior to registering
-- the classes as versioned.

DECLARE
CURSOR fetch_oid IS
 SELECT a.objectid "A_OID", b.objectid "B_OID" FROM <feature class> a,
<annotation class> b  WHERE a.<feature class join field> = b.<annotation
class join field> FOR UPDATE;
```

```
ctr NUMBER := 0;

BEGIN
FOR update_loop IN fetch_oid LOOP
 UPDATE <annotation class> SET featureid = update_loop.a_oid WHERE
objected = update_loop.b_oid;

IF ctr = 1000 THEN
 COMMIT;
 ctr := 0;
END IF;

ctr := ctr + 1;
END LOOP;

COMMIT;
END;
/
```

Appendix B

The attached init.ora file is the parameter values we set for our case study. Descriptions for some of the parameters will provide assistance in determining the values you may consider when modifying your instances file. Please reference Oracle's documentation for specific descriptions and default values for each parameter.

INSTANCE_NAME = gdb816

SERVICE_NAMES = gdb

CONTROL_FILES = ("/azteca1/oradata/gdb/control01.ctl",
"/azteca4/oradata/gdb/control02.ctl")

DB_BLOCK_BUFFERS = 65000
Specifies the number of database buffers available in the buffer cache. It is one of the primary parameters that contribute to the total memory requirements of the SGA on the instance.

SHARED_POOL_SIZE = 204800000
The shared pool stores shared SQL statements, PL/SQL blocks, packages, procedures, functions, triggers, and the data dictionary cache. Increasing the size of the shared pool will increase performance for a large multiuser system.

SHARED_POOL_RESERVED_SIZE = 20480000
This parameter can be used to avoid performance degradation in the shared pool when pool fragmentation forces Oracle to search for and free chunks of unused pool to satisfy the current request. Oracle documentation recommends 10 percent of the shared pool be reserved; the default size is 5 percent.

J-8457

LOG_CHECKPOINT_INTERVAL = 0

LOG_CHECKPOINT_TIMEOUT = 0

PROCESSES = 215

LOG_BUFFER = 512000
A larger value in a busy system decreases the amount of disk I/O to the log files. Its maximum value is calculated by 128K * CPU_COUNT.

DB_WRITER_PROCESSES = 4
Specifies the initial number of database writer processes for an instance.

OPEN_CURSORS = 100

SORT_AREA_SIZE = 1024000
Maximum amount of memory used before the sort operation occurs on disk (1M).

SORT_AREA_RETAINED_SIZE = 512000
Specifies the maximum amount of memory to retain in the User Global Area (UGA) after performing a sort operation.

MAX_DUMP_FILE_SIZE = 5120000
Limits trace file size to 5M each.

LOG_ARCHIVE_START = false

LOG_ARCHIVE_DEST_1 = "location=/azteca7/oradata/gdb/arch"

LOG_ARCHIVE_FORMAT = %t_%s.arc

ROLLBACK_SEGMENTS = (r01, r02, r03, r04, r05, r06, r07, r08, r09, r010)

BACKGROUND_DUMP_DEST = /azteca1/ora815/admin/gdb/bdump

CORE_DUMP_DEST = /azteca1/ora815/admin/gdb/cdump

USER_DUMP_DEST = /azteca1/ora815/admin/gdb/udump

SQL_TRACE = false
We recommend keeping SQL_TRACE disabled. If you need to perform a trace, enable it only for a specific session.

DB_BLOCK_SIZE = 16384
16K block size

REMOTE_LOGIN_PASSWORDFILE = none

OS_AUTHENT_PREFIX = ""

COMPATIABLE = "8.1.6"

TIMED_STATISTICS = true
Enable TIMED_STATISTICS to monitor and maintain database statistics.

PRE_PAGE_SGA = true
Prepaging the SGA will ensure the entire SGA memory is contiguous when the instance is started.

RESOURCE_LIMIT = true
Enabling RESOURCE_LIMIT sets the enforcement of resource limits in profiles assigned to database users.

SESSION_CACHED_CURSORS = 200

REPLICATION_DEPENDENCY_TRACKING = false
You should not disable this value unless you are positive that no read/write operations will occur to replicated tables. When disabled, read/write operations to the database will run faster.