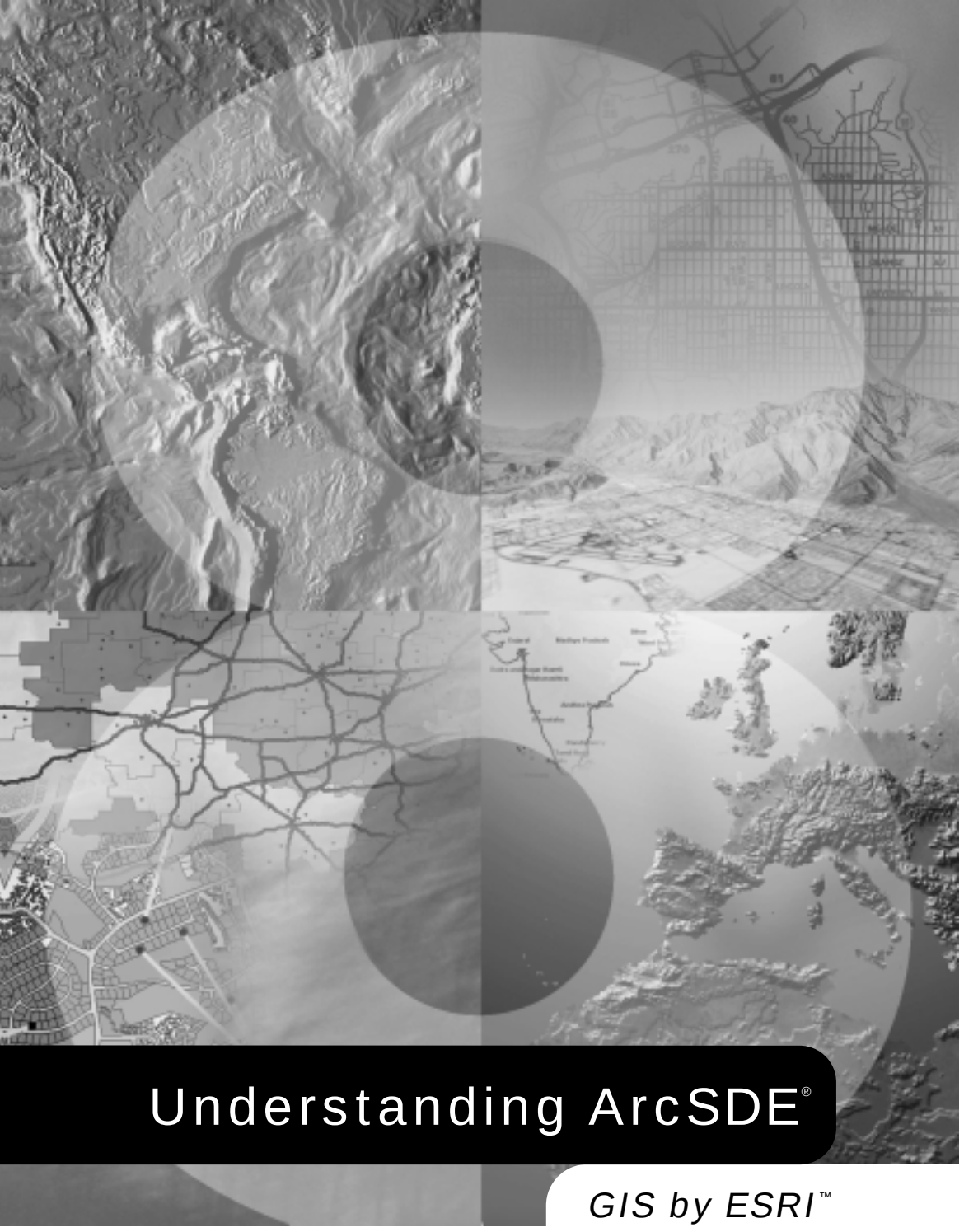


# Understanding ArcSDE®

*GIS by ESRI™*

# Contents


INTRODUCING ArcSDE ..... 1

Welcome to ArcSDE ..... 2

ArcSDE benefits ..... 3

ArcSDE: a key part of ArcGIS ..... 5

Additional applications using the ArcSDE gateway ..... 8

The ArcSDE developer's API ..... 9

A message to SDE 3 users ..... 10

GEODATABASES ..... 11

What is a geodatabase? ..... 12

The geodatabase repository ..... 13

Vector data in a geodatabase ..... 14

Raster data in a geodatabase ..... 17

Other data in a geodatabase ..... 20

Working with geodatabases ..... 21

Implementing geodatabases ..... 23

DATA STORAGE ..... 25

Data storage with ArcSDE ..... 26

Organizing features ..... 27

Types of features ..... 28

Feature storage ..... 29

Geometry storage options ..... 31

Spatial indexing ..... 32

Relational access and object relational access ..... 33

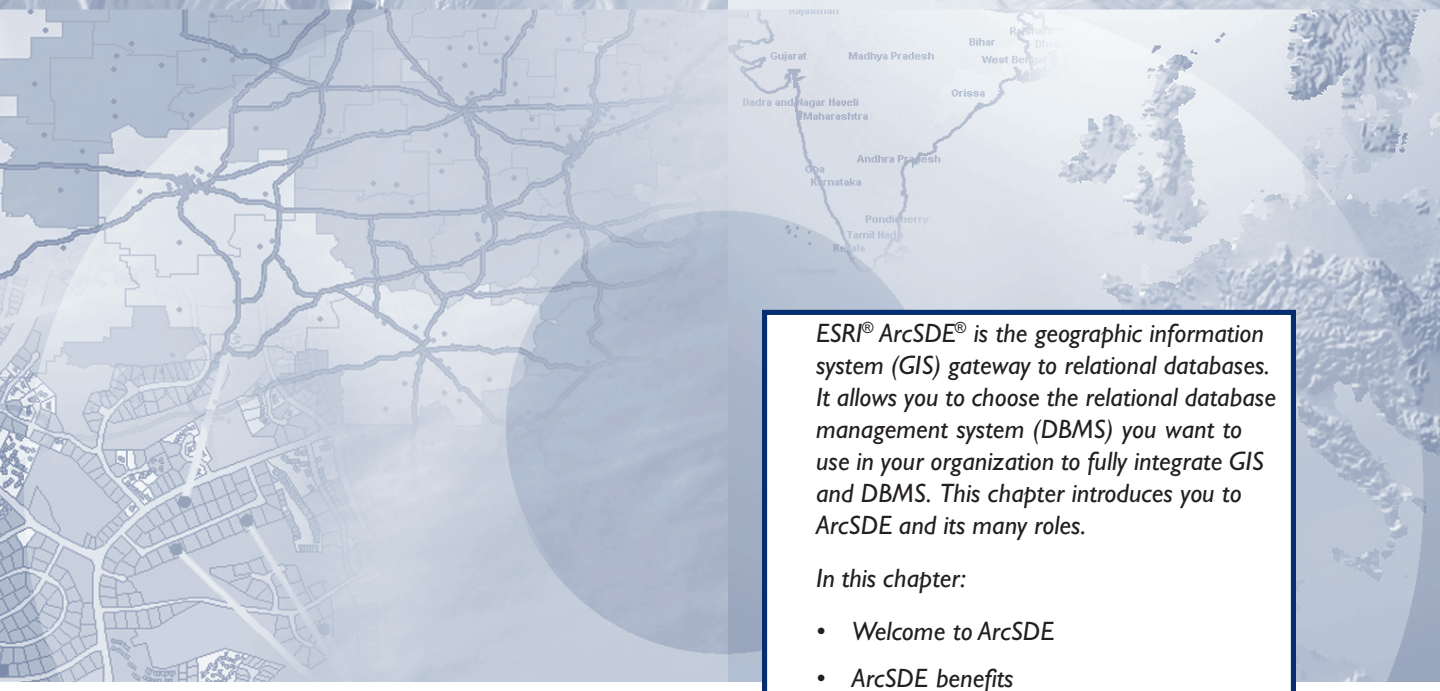
Raster data storage ..... 35

THE ARCSDE ARCHITECTURE ..... 37
What is an application server? ..... 38
Direct connections ..... 40
Connecting with a direct connect driver ..... 41
Licensing ..... 42

ARCSDE FOR COVERAGES ..... 45
Introducing ArcSDE for Coverages ..... 46
ArcSDE for Coverages ..... 47

WHERE TO GO NEXT ..... 49
Administrators ..... 50
Developers ..... 51
End users ..... 52
ArcSDE resources ..... 53

# 1 Introducing ArcSDE




*ESRI® ArcSDE® is the geographic information system (GIS) gateway to relational databases. It allows you to choose the relational database management system (DBMS) you want to use in your organization to fully integrate GIS and DBMS. This chapter introduces you to ArcSDE and its many roles.*

*In this chapter:*

- *Welcome to ArcSDE*
- *ArcSDE benefits*
- *ArcSDE: A key part of ArcGIS*
- *Additional applications using the ArcSDE gateway*
- *The ArcSDE developer's API*
- *A message to SDE® 3 users*

Welcome to ArcSDE, the GIS gateway to your DBMS.

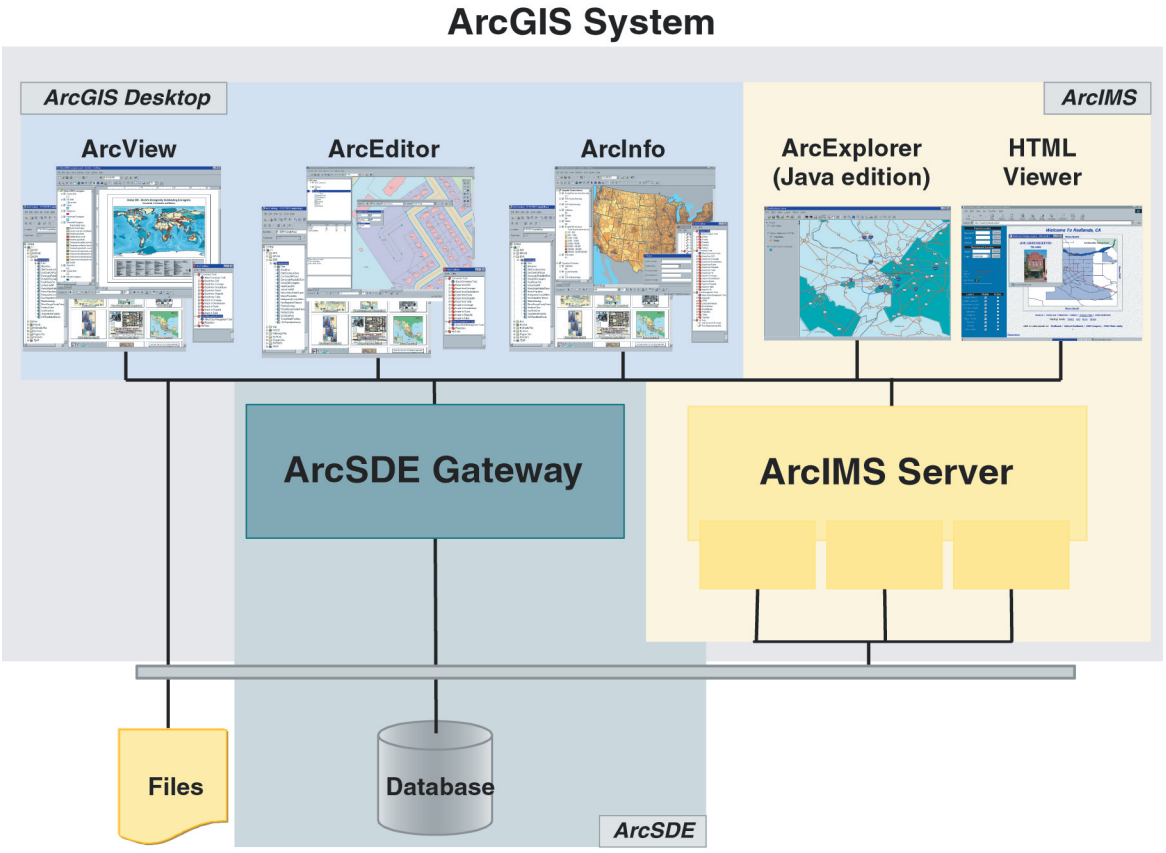
ArcSDE is the tool that allows you to store and manage spatial data in your chosen DBMS. ArcSDE is open; it works with a variety of different databases—including Oracle®, Informix®, IBM® DB2®, and Microsoft® SQL Server™—that scale from work groups to large enterprise databases.

ArcSDE plays a fundamental role in a multiuser GIS. With ArcSDE, your ArcGIS™ software (ArcInfo™, ArcEditor™, ArcView®, and ArcIMS™) can work directly with spatial data managed in your DBMS. ArcSDE also works as an application server, delivering spatial data to many kinds of applications and serving spatial data across the Internet.

ArcSDE provides the gateway between the GIS and the DBMS to share and manage your spatial data as tables. In a heterogeneous database environment, where a number of different departmental or personal database systems are used, ArcSDE provides a common model for geographic information. This allows you to take full advantage of the facilities your DBMS has to offer for integrating your GIS information with the rest of your organization's data holdings.

In this book, you will begin to learn about three primary roles of ArcSDE in multiuser GIS:

- DBMS Gateway supporting many databases
- As a high-performance application server to deliver spatial data to many users and applications
- Advanced editing services for long transactions and versioning



Multiuser GIS requires a database and GIS tools to work with the spatial data. Many types and sizes of spatial databases can be built, but typically, as your organization evolves, your spatial database will grow in size and number of users. As your spatial database grows, you may find it valuable to share it more widely within your organization or outside of your organization on the World Wide Web.

In light of the trend toward large, widely shared spatial databases, ArcSDE provides a number of key benefits including the following:

### Database connection configuration options

ArcSDE allows you to distribute the GIS application processing between the database server, the client, and the ArcSDE application server. You may also configure your system to use what are called "direct connections". The application can connect directly to the DBMS without using the ArcSDE application server. Connecting to the ArcSDE application server provides performance advantages as well as providing a mechanism for supporting serverside applications. Direct connections provide an easier way to achieve DBMS connectivity with less administration. You can, for example, directly connect to an Oracle Spatial database without requiring the ArcSDE application server. Direct connections are also an advantage in configurations requiring failover support. The connection configuration options allow you to define the system that best supports your needs.

### Spatial data representation

The spatial data representation is built on standard data types in the DBMS. In cases where the DBMS has extended spatial data types, ArcSDE can access and use these types for managing feature geometry. When necessary, ArcSDE provides the mechanism for managing feature geometry. These storage methods provide a fast and compact representation for spatial data.

### Database portability

With ArcSDE, you can move data from one DBMS to another without loss of information through ArcSDE data export and import capabilities. This is especially important if your enterprise has a heterogeneous database environment, possibly including work group or personal database systems. This capability is available for even the most advanced geodatabase designs.

### Application portability

ArcSDE defines a logical model for spatial data, implemented on top of a physical database representation. With little or no change, applications developed with the ArcSDE application programming interface (API) will run on different physical schemas within a DBMS. This lets the GIS database administrator choose the best database schema for each individual dataset to meet the requirements of an application. For example, one dataset can be maintained and edited using a binary schema (stored as LONG RAW in Oracle), while another dataset might be stored as Oracle Spatial geometry types. How the data is stored is transparent to the end-user application.

### Data integrity

ArcSDE manages the integrity of the point, line, and polygon information added to the database and won't allow ill-formed feature geometry to be inserted (for example, polygon boundaries must be closed). In addition, you can use the ArcSDE gateway with ArcInfo and ArcEditor to implement additional integrity constraints on the data model that aren't practical to implement in the DBMS itself. For example, you can add connectivity rules for utility networks.

### Application programming interface

ArcSDE provides open, high-level C and Java™ APIs for querying and processing spatial information. These APIs provide GIS functions for advanced application development. When the host DBMS provides extended spatial types (for example, Oracle Spatial, Informix Spatial DataBlade®, and IBM's DB2 Spatial Extender), a Structured Query Language (SQL) API is also available for the DBMS to work with the geometry columns in the DBMS.

ArcGIS provides a Microsoft Component Object Model (COM) API called ArcObjects™, containing a Geodata Access subsystem, which is compliant with the Open GIS Consortium (OpenGIS®) simple feature standard.

### Database and application development costs

You can significantly reduce the cost of building and maintaining a spatial database by using ArcSDE with ArcGIS. ArcGIS provides many tools and datasets to help you quickly implement your application. Other ESRI software can use the ArcSDE gateway—so you can choose the most appropriate tools for your application.

### Applications and development tools

ArcSDE is the standard interface that allows direct access to spatial databases from ESRI's GIS software—ArcInfo, ArcEditor, ArcView, and ArcIMS. These applications and their integrated developer tools provide a comprehensive framework for creating, managing, and using spatial information. ArcSDE also supports direct interfaces from AutoCAD® and MicroStation® to spatial databases. In addition, a significant third-party developer community exists for ArcSDE. Visit the ESRI ArcSDE Web site at www.esri.com to learn more about partner applications.

ArcGIS is an integrated family of software consisting of three key parts:

- ArcGIS Desktop, an integrated suite of advanced GIS applications available as ArcView, ArcEditor, and ArcInfo
- ArcIMS, Internet-based GIS for distributing data, maps, and services
- ArcSDE, the gateway for managing GIS data in a DBMS

ArcGIS uses an object-relational data model called a geodatabase for representing geographic information in a DBMS. Geodatabases model not only spatial features but also their behavior, rules, and relationships with other feature classes and objects in the geodatabase. This integration of behavior with features allows you to create more sophisticated, advanced GIS models. Once you've defined behavior for features, it is available in each of the ArcGIS desktop applications—ArcCatalog™, ArcMap™, and ArcToolbox™—as well as in ArcObjects, the object-oriented developer components for ArcGIS.

A geodatabase is implemented using standard DBMS technology and can scale from small personal geodatabases maintained by the Microsoft Jet Engine to multiuser databases shared in a work group or across an enterprise.

Since building and managing a shared geodatabase for multiple users require a GIS and a DBMS, ArcGIS Desktop and ArcSDE are delivered together as a single solution. You use ArcGIS Desktop with ArcSDE to implement your database design, specify feature behavior, add and edit geodatabase contents, and manage geodatabases in a multiuser setting.

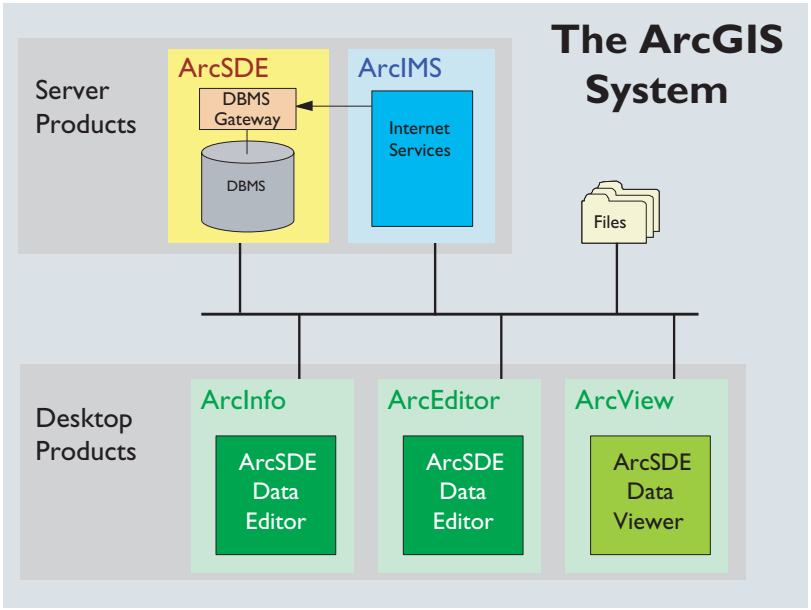
Three types of ArcGIS Desktop seats are available to use with ArcSDE and a multiuser geodatabase:

1. ArcView for mapping, map analysis, and geodatabase use.

2. ArcEditor for building and maintaining a multiuser geodatabase with its advanced editing and version management.

3. ArcInfo for data loading, geoprocessing, and other advanced tasks.

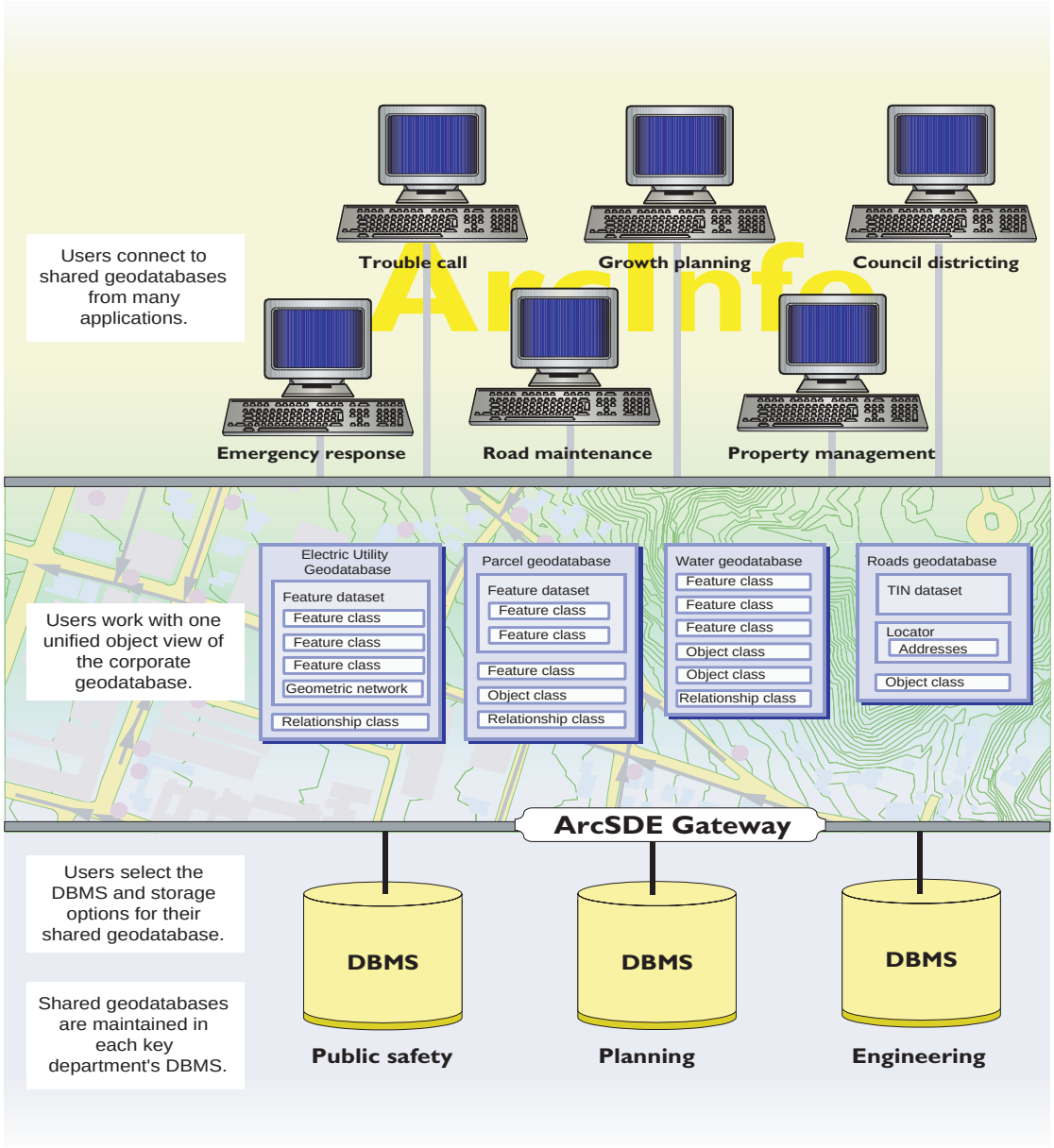
Each of these seats provides more and more powerful capabilities for building and using large multiuser geodatabases.




*The ArcGIS Desktop products—ArcInfo, ArcEditor, and ArcView—as well as the Internet GIS technology—ArcIMS—are designed to take maximum advantage of ArcSDE and DBMS technology.*

The combination of ArcGIS Desktop and ArcSDE provides a complete and powerful enterprise GIS solution. Together, they allow you to:

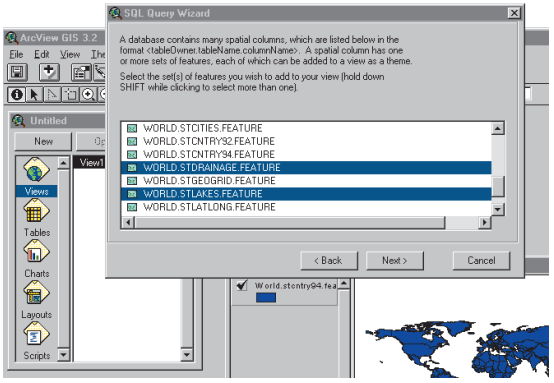
- Directly edit spatial information in the DBMS.
- Manage the edits of multiple users on the same geodatabase with long transactions and alternate versions.
- Manage features in a geometric network. Connect and edit features as a graph of related objects and perform network queries such as tracing and finding the shortest path between locations.
- Manage features with integrated topology and shared geometry. Define and edit features that have been integrated in a planar topology and maintain their integrity.
- Associate editing and mapping rules with features in the geodatabase. You can define and manage associations (or relationships) between different feature classes or objects such as between parcels and owners or between lot lines and lot line annotation. ArcMap, the ArcGIS editing and mapping application, exploits and maintains these relationships.
- Manage georeferenced imagery and other raster datasets in the geodatabase.
- Convert spatial information between a variety of different file formats and the geodatabase.
- Use address matching and dynamic segmentation to associate database records with geographic locations for mapping and spatial analysis.
- Write geodatabase applications by either customizing or extending the standard ArcGIS application framework or by using ArcObjects, the extensive COM-based development library for ArcInfo.


Users connect to shared geodatabases from many applications.
Trouble call
Growth planning
Council districting
ArcInfo
Emergency response
Road maintenance
Property management
Users work with one unified object view of the corporate geodatabase.
Electric Utility Geodatabase
Feature dataset
Feature class
Feature class
Feature class
Geometric network
Relationship class
Parcel geodatabase
Feature dataset
Feature class
Feature class
Feature class
Object class
Relationship class
Water geodatabase
Feature class
Feature class
Feature class
Object class
Object class
Relationship class
Roads geodatabase
TIN dataset
Locator
Addresses
Object class
ArcSDE Gateway
Users select the DBMS and storage options for their shared geodatabase.
Shared geodatabases are maintained in each key department's DBMS.
DBMS
DBMS
DBMS
Public safety
Planning
Engineering

The high-performance ArcSDE gateway lets you serve the contents of geodatabases to many people who may use a wide variety of applications to access the information. It allows you to centrally manage and share your organization's large, multiuser geodatabase or geodatabases. In a typical client/server configuration, the ArcSDE server is located with the centralized database on the network. Desktop applications establish connections over the network to work with the contents of the geodatabase. Geodatabases can also be served over the Internet using ArcIMS.

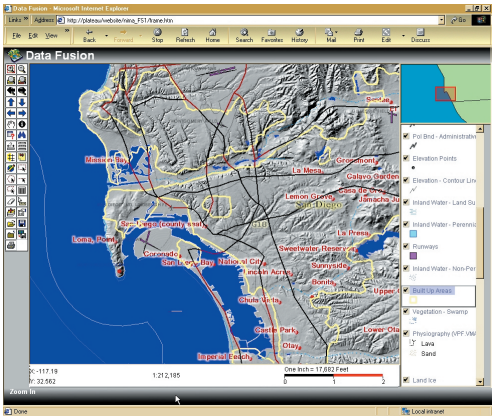
You can use a wide variety of GIS and computer-aided design (CAD) applications to work with geodatabases through the ArcSDE gateway.

ArcView GIS 3, the world's most popular GIS, can access and work with geodatabases through ArcSDE. Custom applications are built using Avenue™, ArcView GIS 3's object-oriented programming environment.

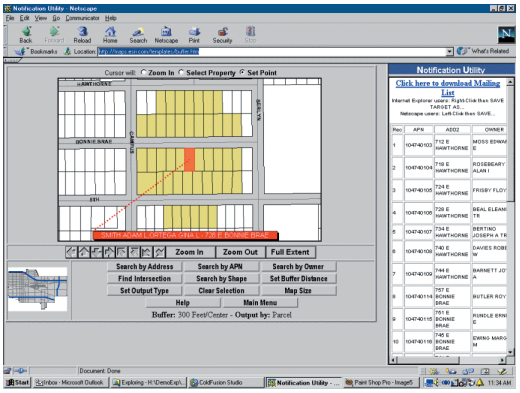



*ArcView GIS 3*

ArcIMS provides distributed GIS for the Internet, adding services and applications that allow your GIS to be distributed across the World Wide Web. ArcIMS can include ArcSDE as part of its configuration for shared geodatabase access.




*ArcIMS 3 Custom Java Viewer*



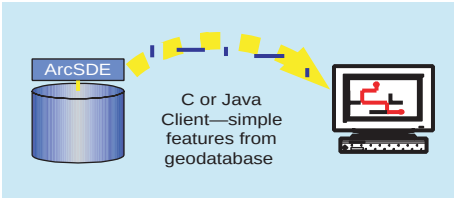

*ArcIMS 3 HTML Viewer*

ArcSDE comes with high-level APIs for querying and working with information in geodatabases. These include:

- ArcSDE client API for C and Java developers
- COM API (ArcObjects) for use with ArcInfo, ArcEditor, and ArcView

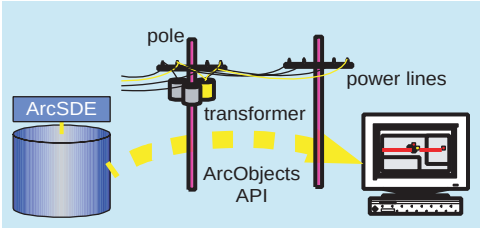
### ArcSDE Client API

Available for both C and Java, the ArcSDE client API provides many advanced GIS functions. All ESRI client applications that work with the ArcSDE gateway use the C API. The ArcSDE client API provides a relational view of simple spatial features for these applications. These APIs allow developers to build custom applications to work with any DBMS supported by ArcSDE. The client API is most appropriate for building focused, mission-critical applications—for example, in emergency response and customer care. ArcSDE has a significant third-party developer community providing application solutions for many industries.



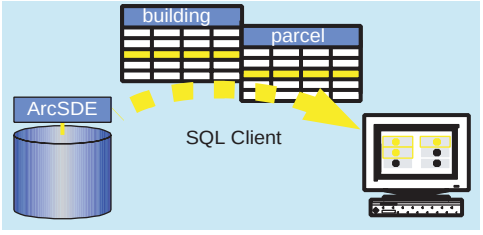

### ArcObjects

ArcObjects is the COM developer's API for ArcGIS Desktop. It provides the ability to access and work with the contents of geodatabases as objects with advanced GIS behavior and relationships.




### SQL

There are SQL APIs available for working with geodatabases. The SQL interface provided by your DBMS can be used to work with the contents of geodatabases. Since geodatabases use standard DBMS columns, the DBMS's SQL API is used. For a spatially enabled DBMS (i.e., Informix, IBM DB2, and Oracle Spatial), special geometry types extend the types supported in the DBMS and in their SQL implementations, allowing you to perform spatial queries directly in the geodatabase. This allows query access to feature geometry using DBMS applications.




For example, you could use an SQL select statement to perform a spatial join.

ArcSDE 8.1 is a major upgrade to Spatial Database Engine™ (SDE) 3. In addition to adding significant extensions to the SDE 3 data model, ArcSDE is tightly integrated with ArcInfo, ArcEditor, and ArcView. While it is important for you to begin to understand these changes, another major goal of ArcSDE is to continue to support all of the existing capabilities in SDE 3 so that migrating your applications to ArcSDE is not difficult. This section provides an overview of many of the changes to help you understand how to migrate your existing SDE 3 systems to ArcSDE.

If you have been using SDE 3 in a production environment, you may be concerned about migrating to ArcSDE. What's involved? First, ArcSDE provides all the existing capabilities of SDE 3 and will continue to serve its traditional role as an application server for SDE 3 databases in your DBMS, without requiring data to be reloaded. All of the currently supported SDE 3 client applications (such as ArcView GIS 3 and MapObjects®) continue to be supported by ArcSDE at the same level as by SDE 3. No changes are required in existing ESRI applications to migrate from SDE 3 to ArcSDE. Custom applications built from the C API will need to recompile and relink with ArcSDE client libraries before being able to work with ArcSDE.

The transition to ArcSDE is simpler in cases where your database schema does not change. However, because of the new extensions to the spatial data model and the tight integration with ArcInfo, you should consider migrating your SDE 3 system to ArcSDE to take advantage of these new capabilities. Several migration strategies are presented in the *What is ArcGIS?* booklet.

### From SDE 3 layers to geodatabases

Perhaps the most fundamental change in ArcSDE is the extension to the spatial data model. In earlier releases of SDE, the spatial data model organized features into independent feature classes called SDE layers. Each individual SDE layer is a collection of features organized into one or more feature (entity) types as a feature table in your DBMS. When you needed to model relationships between feature classes, it was your responsibility to build feature behavior and relationships in your application.

In ArcSDE, spatial data in a database is called a geodatabase. In conjunction with ArcInfo, the geodatabase data model capabilities have been significantly enhanced to support advanced vector data models, raster datasets, and address-based datasets. While geodatabases are stored in relational tables, they are used as objects in ArcInfo. Feature behavior is easily modeled in ArcInfo, so advanced applications are much easier to implement.

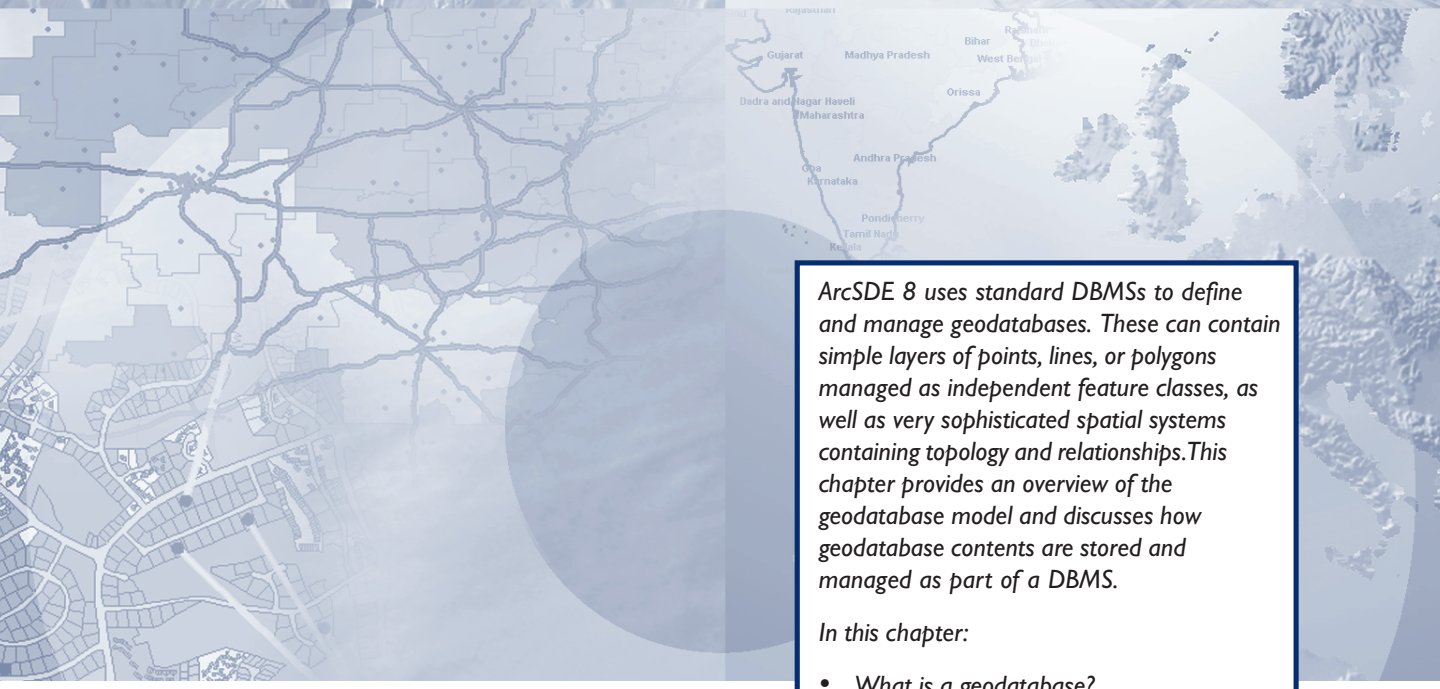
Geodatabases for ArcSDE are flexible enough to support both existing SDE 3 layer data models and more advanced database designs. With ArcSDE, independent layers can still be designed and managed as they are in SDE 3; they are referred to as standalone feature classes. In addition, sets of related feature classes can be managed together in an integrated feature dataset. Feature datasets are somewhat analogous to the file-based coverage format in ArcInfo—they can contain multiple feature classes and can be used to represent more complex spatial models than independent SDE layers. For example, topological relationships can be used to model networks and shared geometry between feature classes using feature datasets. This added ability to model systems of related features in feature datasets is a very significant enhancement.

Features in geodatabases have been extended beyond features in SDE layers in two other areas: geometry and behavior. In geodatabases, feature geometry has been extended to include parametrically defined edges such as circular curves and Bézier curves. These allow more sophisticated and accurate representations of the built environment. Additionally, features in geodatabases can now have behavior. Behavior can be implemented in geodatabases using validation rules, relationships, and attribute domains.

Another key advantage of ArcSDE over SDE 3 is the ArcInfo support for concurrent editing of data by multiple users using versioned geodatabases. The ability to create versions of the geodatabase allows more flexibility in creating, maintaining, and managing a geodatabase. ArcInfo and ArcSDE also provide the opportunity to create versions that simulate alternative design scenarios.

Chapter 2, 'Geodatabases', describes in more detail how geodatabases extend the fundamental SDE 3 data model.

# 2 Geodatabases




*ArcSDE 8 uses standard DBMSs to define and manage geodatabases. These can contain simple layers of points, lines, or polygons managed as independent feature classes, as well as very sophisticated spatial systems containing topology and relationships.This chapter provides an overview of the geodatabase model and discusses how geodatabase contents are stored and managed as part of a DBMS.*

*In this chapter:*

- *What is a geodatabase?*
- *The geodatabase repository*
- *Vector data in a geodatabase*
- *Raster data in a geodatabase*
- *Other data in a geodatabase*
- *Working with geodatabases*
- *Implementing geodatabases*

A geodatabase is a repository of your spatial data inside a DBMS. It contains all of your vector data, raster data, tables, and other GIS objects. The term geodatabase is short for geographic database, a relational database containing geographic information.

Geodatabases come in many sizes and have any number of users. They can scale from small, single-user databases to large work group and enterprise geodatabases used simultaneously by many users. ArcSDE lets you implement a multiuser geodatabase of any size in the DBMS of your choice—Oracle, Microsoft SQL Server, IBM DB2, and Informix.

The geodatabase model supports an object-relational vector data model. In this model, entities are represented as objects with properties, behavior, and relationships. Support for a variety of different geographic object types is built into the system.

These object types include simple objects, geographic features (objects with location), networks and topology (objects having spatial relationships with other features), annotation features, and other more specialized feature types.

The geodatabase model lets you define relationships between objects, together with rules for maintaining their referential integrity.

The simplest geodatabase is one that contains a number of independent feature layers. Each feature layer typically contains points, lines, polygons, or annotation. This is analogous to how SDE layers are implemented in SDE 3 and to ArcView shapefiles. In these data models, each layer contains a single, standalone feature class.

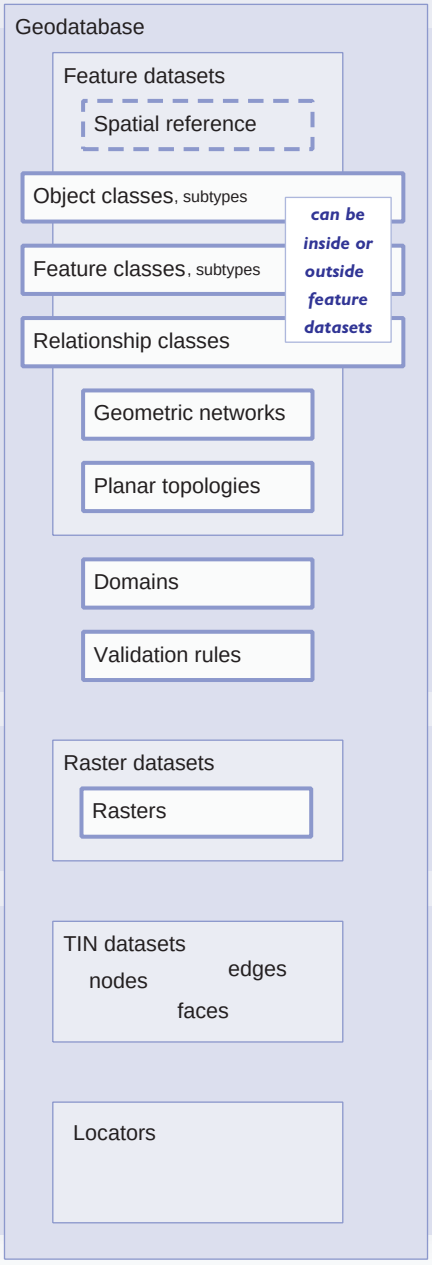
The goal of this chapter is to briefly introduce you to geodatabases and how they can be used to model geography—from very simple data models containing a few independent feature classes to sophisticated spatial systems containing topology and other complex relationships among the spatial objects in the database. This chapter provides a foundation for exploring geodatabase design, which is discussed in detail in the book *Modeling Our World—The ESRI Guide to Geodatabase Design*.

| | DBMS | Client/Server | Objects | Long Transactions* | Editors | C or Java API | Raster | Size |
|---|---|---|---|---|---|---|---|---|
| Multiuser Geodatabases | Oracle, Microsoft SQL Server, Informix, IBM DB2 | Yes | Yes | Yes | 1 or more | Yes | Yes | Unlimited |
| Personal Geodatabase | Microsoft Jet | No | Yes | No | 1 only | No | No | Up to 2 Gb |

*Database transactions spanning multiple edit sessions

*Geodatabases can be multiuser, implemented with ArcSDE in the DBMS of your choice, or smaller personal geodatabases implemented in Microsoft Jet Engine databases. The 2 Gb limit in personal geodatabases is the MS Access MDB file size limit.*

# Inside a geodatabase




All feature classes in a feature dataset share a common coordinate system. Because the feature dataset is the container of topological associations, it is important to guarantee a common spatial reference.


A feature dataset contains objects and features and the relationships among them. An object is a nonspatial entity, and a feature is a spatial entity. A relationship links two entities.

Objects of the same kind are stored in an object class. Features of the same kind and with the same type of geometric shape are stored in a feature class.

A relationship class stores relationships between entities in two object or feature classes.

Geometric networks model linear systems such as utility networks and transportation networks. They support a rich set of network tracing and solving functions.

Planar topologies model systems of line and area features as a continuous coverage of an area. Planar topologies allow features to share common boundaries such as counties that share an outer boundary with a state.

Domains are sets of valid attribute values for object attributes. They can be textual or numeric.

Validation rules enforce data integrity through domains, relationship rules, and connectivity rules.

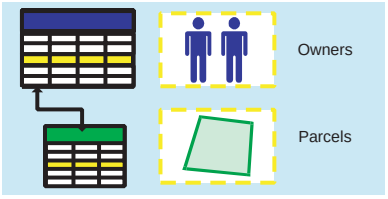
Raster datasets can represent an imaged map, a surface, an environmental attribute sampled on a grid, or photographs of objects referenced to features. Some raster data is collected in bands that commonly represent different spectral ranges of camera filters.

TIN datasets are triangulations of sets of irregularly located points with z-values (elevations) sampled from a surface. TINs are most often used to model the earth's surface but are also used to study the distribution of a continuous environmental factor such as chemical concentration. At the 8.1 release TINs are stored in coverage workspaces.

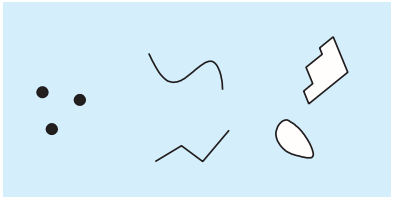

Many databases contain alphanumeric descriptions of locations such as addresses. These descriptions can be stored in tables in a geodatabase. A locator defines a process for converting alphanumeric descriptions to geographic features in the geodatabase.

Vectors are a versatile and frequently used geographic data representation, well-suited for representing features with discrete boundaries such as wells, streets, rivers, states, and parcels. Typically, features are spatially represented as points, lines, or polygons. Here is a summary of the vector contents in a geodatabase.

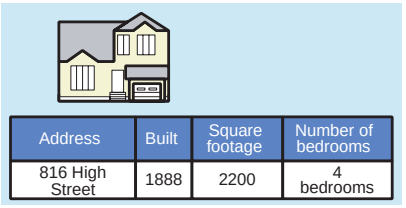
- Object class. A database table with which you can associate behavior. Rows in a table are object instances that have special behavior in the GIS. An example of an object class is "owners" of "land parcels". You can establish a relationship between the polygon features for land parcels and the object class of owners.




- Feature class. A collection of features of the same type. A feature is simply an object that has a location stored as one of its properties or fields in the row. A feature's geometry type is commonly point, line, polygon, or annotation. Examples of feature classes are streams, counties, and census tracts. Feature classes can be independent of one another or can be related to other feature classes. When related to one another, feature classes are organized together in a feature dataset, which you will read more about below. For readers familiar with SDE 3 databases, SDE layers are analogous to standalone feature classes. In fact, existing SDE 3 layers present themselves in a geodatabase as standalone feature classes.

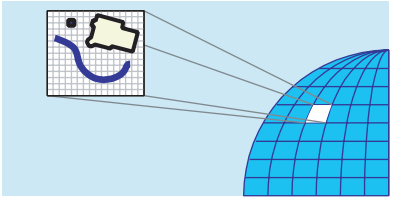


- Feature attributes. Properties stored as fields in a feature class table. Attributes define standard and custom properties of features and can be numeric, textual, or descriptive identifiers.

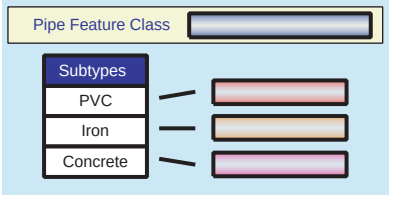


| Address | Built | Square footage | Number of bedrooms |
|---|---|---|---|
| 816 High Street | 1888 | 2200 | 4 bedrooms |

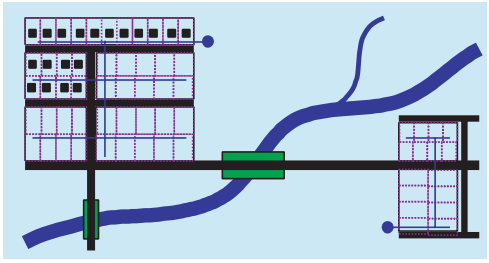
- Spatial reference. The earth-based coordinate system in which the dataset is represented. This specifies the dataset's real-world location. The spatial reference includes properties such as the map projection, the datum, the allowable range for the coordinates (for example, the range for x,y or x,y,z), and so on.



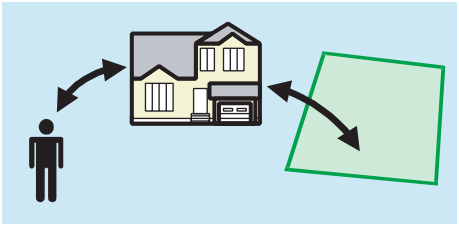
- Subtypes. A set of classes for the members of a feature class. A feature class may contain features that have the same general behavior and properties but that have a different meaning or role in the data model. For example, while it is useful to distinguish iron pipes from PVC pipes and the role each plays in your data model, it may be more appropriate to design a single "pipes" feature class and to distinguish the different types of pipe as subtypes.



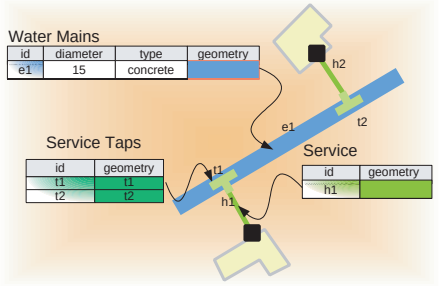
- Feature dataset. A collection of feature classes with the same spatial reference. The feature classes in a feature dataset may be organized into networks or planar topologies. If you are familiar with ArcInfo, feature datasets are analogous to coverages in that they are collections of related feature classes; however, feature datasets are less restrictive and more functional than coverages. Feature datasets are vital when your GIS must model a system of spatially related features such as facilities networks, roads, environmental layers (such as soils, surface topography, and vegetation), census geography, and so on.



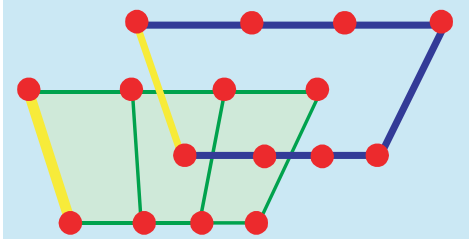
- Relationships. An association between two objects. A relationship lets you work with features and related rows in tables as well as relationships between rows. Relationships are organized into relationship classes. A relationship class defines a set of relationship instances between two feature classes or object classes. For example, feature-based annotation can be modeled using relationships. You can define what happens to the annotation if the feature is moved, deleted, or if its attribute values change.



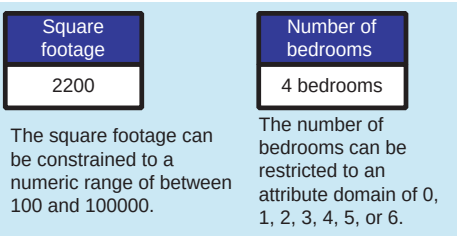
- Geometric networks. A user-defined collection of feature classes that form part of a connected network of edges, junctions, and turns. You define the set of feature classes that participate in a geometric network and the role for each feature class (for example, as edges or junctions), and organize these feature classes into a feature dataset. For example, in a water network, valves and meters play the role of junctions, while mains and service lines have roles as edges.



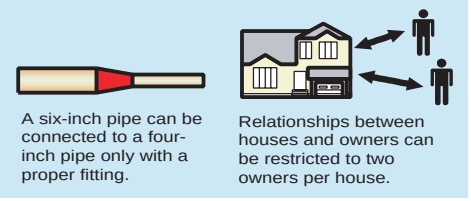

- Planar topologies. A user-defined collection of feature classes that share geometry. A planar topology allows you to have one common set of lines to represent the geometry of a number of feature classes. For example, feature classes, such as soil types, vegetation, terrain, and water, can share common polygon boundaries. Using topology editing tools, updating the geometry of one feature class automatically updates all feature classes in the planar topology sharing the boundary. Feature classes that participate in a planar topology are organized into the same feature dataset.



- Domains. Define the valid values for attributes as a range or a set of values. Domains can be used to validate any attribute in the geodatabase.

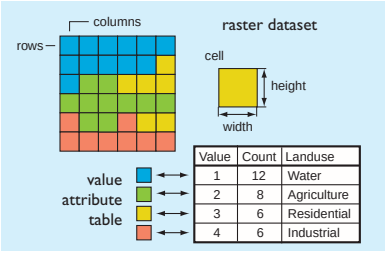



- Validation rules. One or more constraints on the attribute values, topology, or placement of features to enforce the behavioral integrity of your features. For example, connectivity rules define constraints about how features are interconnected in networks.

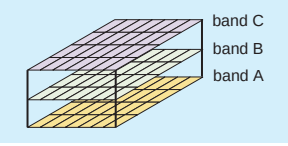


Geodatabases can contain raster datasets, which divide space into uniform cells or pixels. A raster dataset stores a two-dimensional matrix with sampled values for each cell. Each cell has the same width and height. The location (geographic coordinate) of the upper-left corner of the grid, together with the cell size and the number of rows and columns, defines the spatial extent of the raster dataset.

The cells in a raster dataset can depict a variety of data such as the reflectance of light for part of the spectrum in a satellite image, a color value for a photograph, a thematic attribute such as vegetation type, a surface value, or elevation.




A raster dataset contains one or more layers called bands. For example, a color image has three bands (red, green, and blue), a digital elevation model (DEM) has one band (holding elevation values), and a multispectral image may have many bands.




Each raster band contains the actual cell values, as well as key properties, such as:

- Statistics (minimum, maximum, and mean cell values)
- Histogram of cell values
- Value attribute table (optional additional attribute information about various cell values)
- Default color map for raster display (optional)

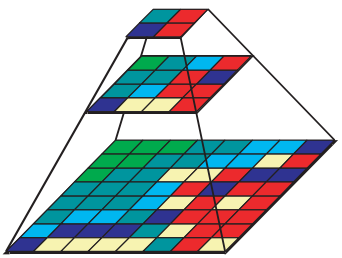
### Very large raster support

Raster datasets in geodatabases can be very large and cover large geographic areas with fine detail. The geodatabase automatically partitions very large raster datasets into tiles and can compress them for efficient access and storage. You can use this method to build very large raster datasets in the geodatabase. When loading raster datasets, you can merge smaller tiles into a mosaic covering the entire image extent, thus creating a large seamless mosaic.

Since raster datasets can cover very large areas, you will often want to work with a subset of the entire database layer when doing analysis. When working with large raster datasets, you can specify the visible extent of the raster to minimize retrieving unwanted data from the server. For display purposes, retrievals are automatically limited to just the data that fits in the current map extent.

### Resampled pyramids increase drawing performance

You can build "pyramids" for a raster dataset in a geodatabase. A pyramid is a series of reduced resolution representations of the dataset. Each level is a resampled representation of the raster at a coarser spatial resolution. Pyramids are used to improve the display performance of rasters when you are not working with the pixel information at full resolution—for example, when zoomed out on a map. Pyramids contain a number of layers, each resampled at a more generalized level.



*Three levels of a pyramid. Each raster is resampled with a larger cell size than the cell size of the raster beneath it in the pyramid.*

When working with an image that has pyramids, ArcGIS automatically determines the most appropriate pyramid level to use each time the raster dataset is drawn. For example, drawing a very large extent uses the most generalized layer in a pyramid, while the most detailed pyramid layer is drawn when you are zoomed in very close.

Image pyramids add some extra storage to a raster dataset, depending on the number of levels in your pyramid. Typical percentage increases are 8 percent. However, the improved drawing performance is dramatic, particularly with datasets larger than 100 megabytes.

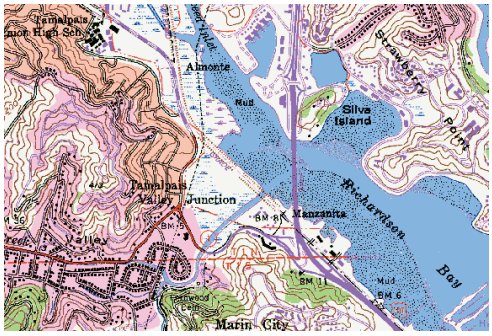
### Supported raster formats

All raster formats supported by ArcGIS can be used to create raster datasets in your ArcSDE geodatabase. Some of the raster types that can be imported to a geodatabase include:

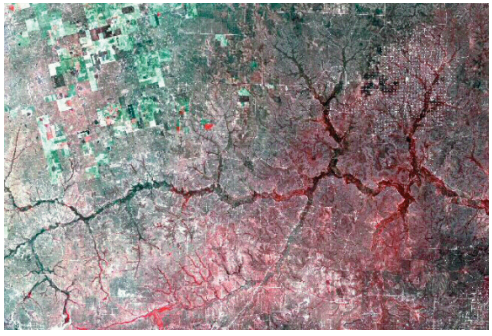
- Grids (including Grid stacks)
- TIFF
- ERDAS® IMAGINE® Images
- ERDAS .raw, .lan, and .gis files
- ER Mapper® images
- BIL/BIP/BSQ band interleaved and sequential files
- MrSID™ encoded images (not encoded in geodatabase)
- BMP
- JPEG
- GIF
- ADRG
- PNG
- CIB
- CADRG
- DTED Level 1 & 2
- NITF

Rasters can represent several kinds of gridded information in a unified model:

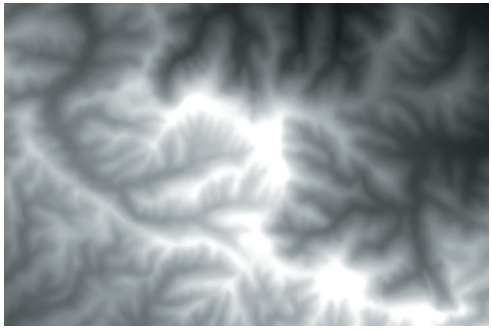
- Scanned map sheets




- Multispectral imagery

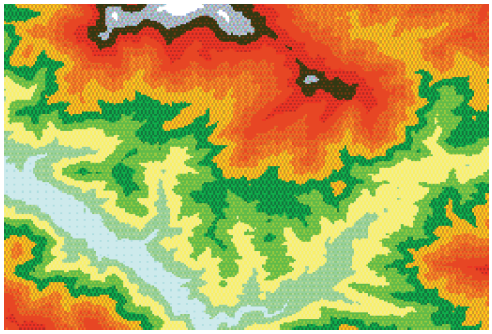


- Digital elevation models



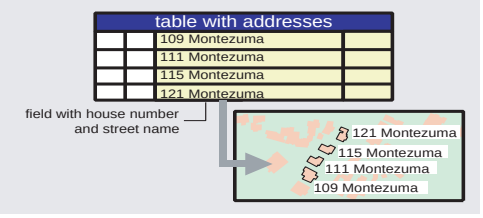
- Categorical grids (for example, land use grid)

- Continuous surfaces (for example, distance from streams, population concentrations)

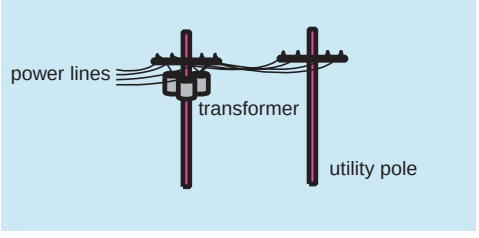


All kinds of spatial data can be managed in a geodatabase. Many spatial data types can be represented and managed with geodatabases including the following:

- Locators. Addresses are among the most common descriptions of locations and can identify places or features on a map. Other types of alphanumeric descriptions of locations are also possible. Locators define processes for creating features that are described by these alphanumeric descriptions of locations. The geodatabase can manage locators and the relationships between these alphanumeric descriptions and the features that they describe.




- User-defined custom features. Geodatabase objects are extensible. Developers can add their own custom features when necessary using the Geodata Access objects in the COM-based development environment named ArcObjects.




TINs are not stored in geodatabases at the 8.1 release, but TIN support is planned for a future release.

GIS has always been a technology with advanced data models to represent spatial systems. A key advantage of geodatabases is that they can be used to create integrated models, using objects and relationships, much more effectively than other GIS data models. Previously, much of the system behavior and the relationships between entities in the spatial system had to be coded in your GIS applications. With geodatabases this behavior is part of your geodatabase design, so application deployment is much simpler and more efficient.

### Modeling feature datasets

A significant task in designing a geodatabase involves defining the entities in the geodatabase and how they will be represented as features, rasters, attributes, and so on. In many, if not most, GIS implementations, it is important to have the ability to define spatial data as a system of objects and relationships. Indeed, part of the power of GIS is the ability to model and represent the relationships among these objects.

For example, a street network consists of connected street segments. "Main Street" could be modeled as a series of connected segments that make up the street sections along each block. Address ranges are associated with each street segment so that locations can be found along each street segment. Intersections define where streets connect and can have special properties. They might have turn restrictions or stoplights. They might be overpasses or underpasses, and so on. In addition, streets can be related to other features such as blocks. The street segments define the boundary of the block they enclose. Groups of blocks can be collected into districts such as census tracts, zoning classes, and other administrative units.

A sample geodatabase design might model this as a set of feature classes in one feature dataset:

- Streets
- Blocks
- Block groups
- Census tracts

All of these would participate in a common planar topology in the feature dataset.

In summary, the reasons to organize feature classes in a feature dataset are numerous, but three common reasons are:

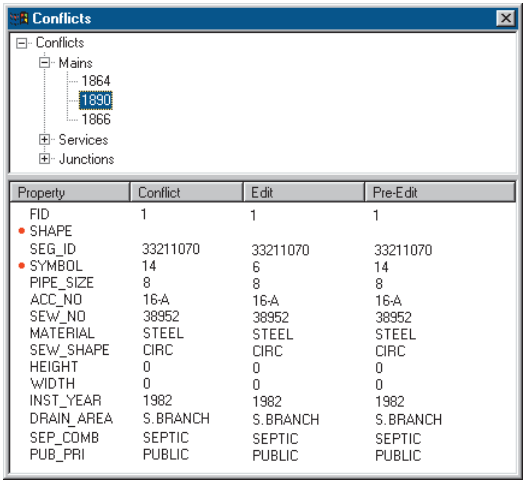
- Thematic representation. The set of feature classes represents the same phenomenon you wish to model. For example, you might have three feature classes for water bodies in a feature dataset: one each for points, lines, and polygon feature representations.
- Geometric networks. The feature classes play roles as junctions, edges, and turns in a geometric network, for example, pipes, valves, pumps, and feeders in a water network or streets, intersections, and turns in an urban road network.
- Planar topologies. Feature classes can share geometry with other feature classes. When you update the boundary of one feature, you want the shared boundaries of other features to be updated as well.

### Long transactions and alternate versions

A key feature in any multiuser database is the ability to manage concurrent access to the data. Multiuser read and write access is crucial. A DBMS provides this functionality on tables. What the DBMS does not provide, however, is the concept of a long transaction for GIS editing or for representing "what if" scenarios. ArcSDE with ArcGIS provides a way to support these with versioning.

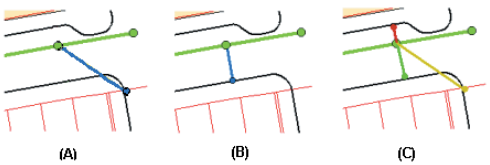
The primary role of versioning is to simplify the editing experience. Many GIS edits require more than just a few minutes' time to complete, and some editing tasks require hours, days, or even months to complete. Long transaction editing is supported by creating versions. Versioning lets users simultaneously create multiple, persistent representations of the database without making copies of the data. Multiple users can simultaneously edit the same features or rows without explicitly applying locks to prohibit other users from modifying the same data.

At the end of an edit session, or when reconciling different versions, the edited features are merged into the target version of the geodatabase. When the same feature has been edited in different versions, a conflict is detected. The conflict resolution dialog box in the ArcMap Editor helps to resolve conflicts, and the affected features are shown on the map.




| Property | Conflict | Edit | Pre-Edit |
|---|---|---|---|
| FID | 1 | 1 | 1 |
| • SHAPE | | | |
| SEG_ID | 33211070 | 33211070 | 33211070 |
| • SYMBOL | 14 | 6 | 14 |
| PIPE_SIZE | 8 | 8 | 8 |
| ACC_NO | 16-A | 16-A | 16-A |
| SEW_NO | 38952 | 38952 | 38952 |
| MATERIAL | STEEL | STEEL | STEEL |
| SEW_SHAPE | CIRC | CIRC | CIRC |
| HEIGHT | 0 | 0 | 0 |
| WIDTH | 0 | 0 | 0 |
| INST_YEAR | 1982 | 1982 | 1982 |
| DRAIN_AREA | S.BRANCH | S.BRANCH | S.BRANCH |
| SEP_COMB | SEPTIC | SEPTIC | SEPTIC |
| PUB_PRI | PUBLIC | PUBLIC | PUBLIC |

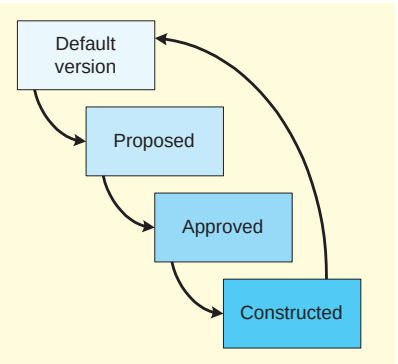
*This conflict resolution dialog box shows three feature classes with conflicts and a feature with each of its version's attributes.*




*The feature in blue as it existed prior to editing (A), the feature after being modified (B), and three representations during conflict resolution (C).*

All versioning capabilities are accessed and used through ArcGIS Desktop. Through versioning, ArcInfo and ArcEditor allow multiuser editing, long transaction support, and history management of ArcSDE geodatabases. ArcView can connect to, view, and use different versions of the database but cannot edit them.

All organizations that use ArcSDE and maintain their data with ArcInfo or ArcEditor will use versioning. The geodatabase and versioning provide organizations with advanced data storage techniques that help to organize work flow in organizations where spatial data is used. Engineers can generate design alternatives using the geodatabase. Spatial analysts can perform complex "what if" scenarios without affecting the default or "as-built" representation in the database. And database administrators can create historical "snapshots" of the geodatabase for archival and database recovery.




*A common work flow process evolving through each stage of a project.*

A common task that uses versioning is geodatabase maintenance. Most users work with and view the default version of the geodatabase. Administrators and editors maintain the database through other versions, managing their edits with the default version when appropriate.

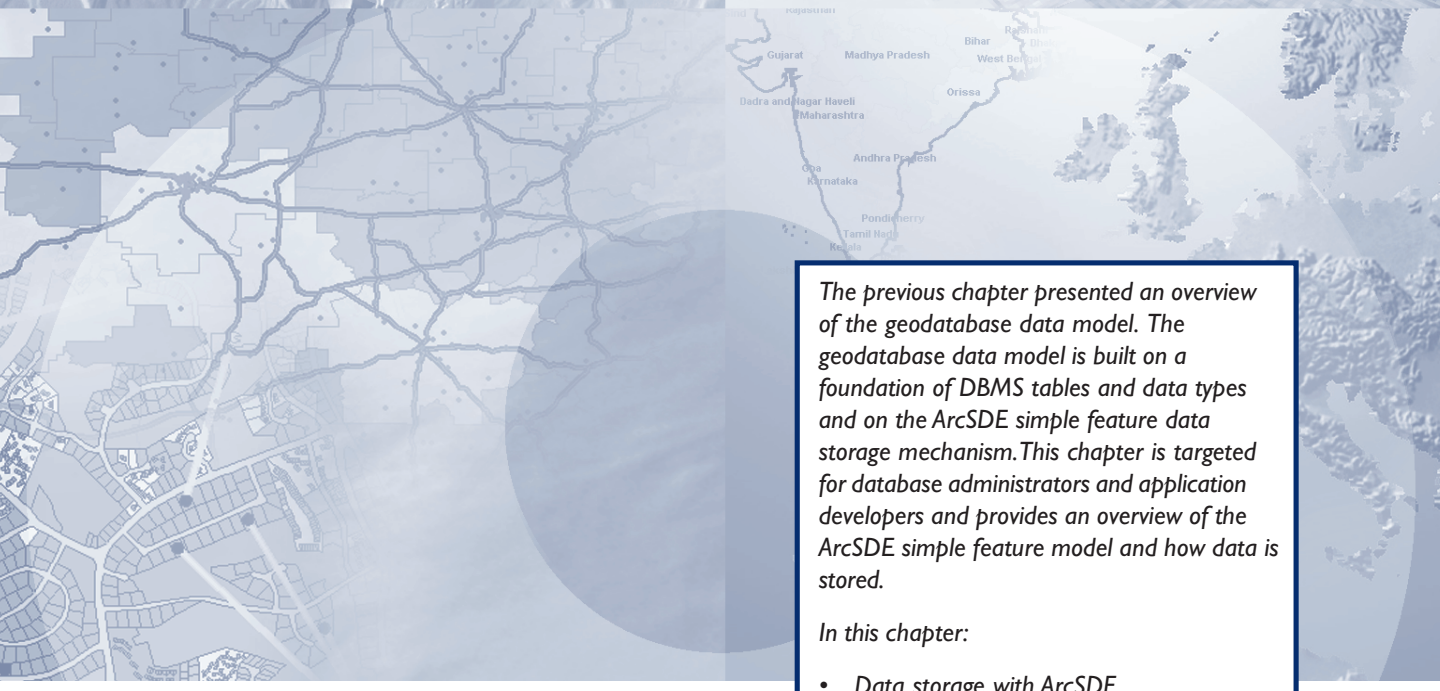
To implement a multiuser geodatabase, you can use ArcInfo or ArcEditor with ArcSDE. Once you decide what data you want to manage, you design the geodatabase schema and implement it using ArcCatalog. The book *Modeling Our World,* included with ArcGIS, is the geodatabase design guide. It explains in detail many of the concepts introduced in this chapter and will help you make key design decisions about your geodatabase implementation.

It is worth noting that others may have attempted the same data modeling exercise. An industry-specific data model guide might be available from ESRI or another party to help you with your specific database design goals. These guides provide a database "template" in which the fundamental objects and schema have been predefined. ESRI provides a number of these designs in application extensions such as ArcGIS Water and Facilities Data Model and the Parcel Data Model. For further information on these extensions, contact your ESRI sales representative.

An excellent guide for implementing geodatabases, *Building a Geodatabase,* is available in the ArcGIS package.

Another very important aspect of building a geodatabase is to fine-tune the DBMS implementation. The physical design of tables and where you put them is critical for optimum performance from your database. There is no perfect cookbook for how to tune a database, but some important guidelines are provided in the configuration and tuning guides for each supported DBMS. Please find a PDF file, config_tuning_guide_<your dbms>.pdf, in the documentation folder of your install location and the CD–ROM installation media.

# 3 Data storage




*The previous chapter presented an overview of the geodatabase data model. The geodatabase data model is built on a foundation of DBMS tables and data types and on the ArcSDE simple feature data storage mechanism. This chapter is targeted for database administrators and application developers and provides an overview of the ArcSDE simple feature model and how data is stored.*

*In this chapter:*

- *Data storage with ArcSDE*
- *Organizing features*
- *Types of features*
- *Feature storage*
- *Geometry storage options*
- *Spatial indexing*
- *Relational access and object relational access*
- *Raster data storage*

In the previous chapter, you learned about the ArcGIS geodatabase. The geodatabase is a data model built on a foundation of simple features. This chapter reviews the basics of the ArcSDE simple feature model and how the data is physically stored in the DBMS.

The first thing you should know is that data is never stored in ArcSDE. Data is stored in a DBMS. ArcSDE is the tool that allows you to put it there and to use it in GIS applications.

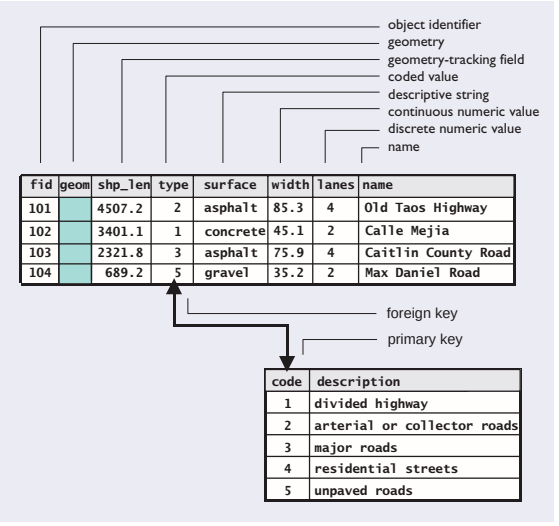
Secondly, all data is stored in standard DBMS tables using data types available for the host database. ArcSDE does not supersede or replace any existing DBMS functionality. Instead, it uses and complements the base DBMS capability by adding a spatial component to the database. A key part of using ArcSDE is tuning the host DBMS. ArcSDE performance hinges largely on how well the host database is tuned.

### Key DBMS concepts

In a relational database management system model, data is stored in tables consisting of rows and columns. The cell defined by the intersection of a row and column is called a field; the data contained in the field is a value. A row represents a particular occurrence, or instance, of a feature, while the columns contain the attributes of the feature such as owner name for a parcel. Attributes can have many types such as dates, text strings, or numbers. A geometric shape of a feature is another type of value, stored in a column that defines an abstract geometric data type.

SQL provides an interface to relational tables that allows you to select rows based on the values contained in the fields. An SQL statement can range from very simple to very complex, allowing you to compose virtually any type of query from basic column types.

A query may return columns from any number of tables by joining the tables together on key columns. A primary key (can be one or more columns) uniquely identifies rows in a table. The same column or columns, duplicated in another table, is called the foreign key. These keys allow tables to be joined.




| fid | geom | shp_len | type | surface | width | lanes | name |
|---|---|---|---|---|---|---|---|
| 101 | | 4507.2 | 2 | asphalt | 85.3 | 4 | Old Taos Highway |
| 102 | | 3401.1 | 1 | concrete | 45.1 | 2 | Calle Mejia |
| 103 | | 2321.8 | 3 | asphalt | 75.9 | 4 | Caitlin County Road |
| 104 | | 689.2 | 5 | gravel | 35.2 | 2 | Max Daniel Road |

| code | description |
|---|---|
| 1 | divided highway |
| 2 | arterial or collector roads |
| 3 | major roads |
| 4 | residential streets |
| 5 | unpaved roads |

The result of a query is a set of rows meeting the criteria established by the SQL statement. This set is called a cursor. An application can reference a number of active cursors. The application steps through a cursor, looking at each individual row. As each successive row is requested, the appropriate values of each field are made available to the application.

ArcSDE extends SQL by providing tools to work with spatial data. You can also use standard SQL in the ArcSDE software's API to perform attribute-only queries.

### Organizing geographic features

ArcSDE organizes features in feature classes. A feature class is a collection of one or more features of one geometric type and is synonymous with the older term "layer", used with SDE 3.x. A feature is a geometric representation of a spatial object (e.g., a road), defined as a sequence of one or more x,y coordinates and the attributes for that geometry. Features are stored so that one row in a table equals one feature.

| Logical elements | Database elements |
|---|---|
| Object | Row |
| Attribute | Column, Field |
| Class | Table |

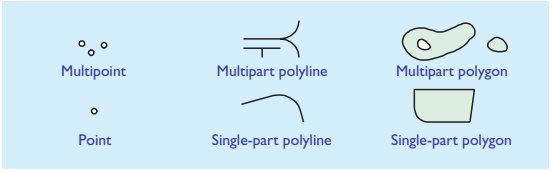
Users may typically think of a feature class as one single table. However, ArcSDE implements a feature class as one or more tables, depending on the DBMS and column type used for storing the geometry.

When you enable versioning on a feature class, two additional tables are added to the layer's table schema. Geometry storage options are database dependent and are reviewed later in this chapter.

ArcSDE doesn't change existing DBMSs or affect current applications. It simply adds a spatial column to tables and provides tools (an API) for a client application, such as ArcMap, to manage and access the geometry data referenced by that column.

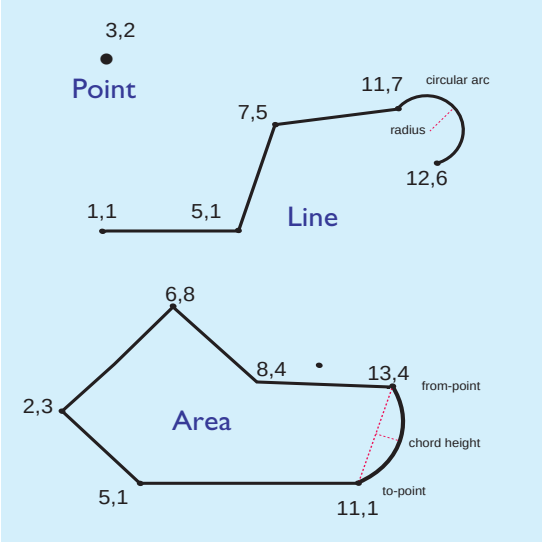
When you add a spatial column to a table (sometimes referred to as a business table), you spatially enable it. The ArcSDE software manages spatially enabled tables by storing information such as the name of a feature class, its owner, x,y extent, type of geometry allowed in the layer, and many other pieces of information in ArcSDE metatables. These ArcSDE metatables are stored in the host DBMS and are created when ArcSDE is installed. They are owned, managed, and populated exclusively by ArcSDE.

We've talked about features in a table. So what geometry types are supported with ArcSDE? The following graphic illustrates the feature geometries you may store in your DBMS with ArcSDE.




### Basic feature storage

ArcSDE stores geometric shapes as x,y coordinates and true curves. Points are recorded as a single x,y coordinate, lines as a series of ordered x,y coordinates and curves, and areas as sets of lines composed of x,y coordinates and curves that have the same starting and ending point.



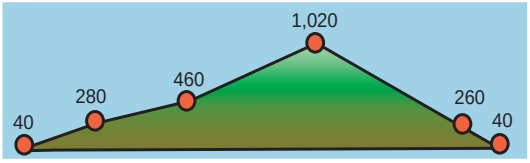

*ArcSDE stores a list of x,y coordinates that define the location and shape of each geographic feature.*

Every geometry type in ArcSDE has a set of strict verification rules that determine whether a geometry is geometrically correct before it is stored. Verification rules for each shape type are described in the ArcSDE Developer Help (in the documentation folder of your ArcSDE installation).

These simple features that ArcSDE stores in a DBMS are the building blocks on which the ArcGIS geodatabase is built.

### Z values

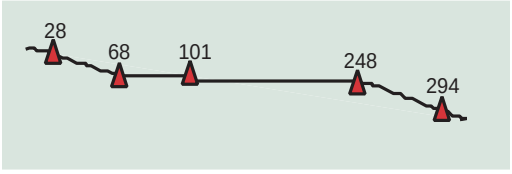
ArcSDE allows you to add z values to the x,y coordinates. Z values can represent height or depth. Geometry may be either two-dimensional (x,y) or three-dimensional (x,y,z).




*Z values, such as elevations or depths, can be stored in a feature's geometry. In this case, points on a mountain have Z values containing their elevation, in addition to their x,y position.*

### Measures

Measures represent a distance, time, address, or some other event at given points along a feature. You can add measure values to any geometry type. Identifying an event by road name and distance from a known location (e.g., Highway 20, kilometer 38) is a common method for locating highway information such as sign or accident locations, exit ramps, pavement quality, and speed zones.



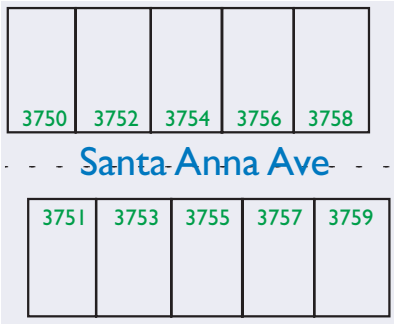

*Measures on a simple line shape mark traffic accidents on a highway.*

Measure values are independent of a geometry's coordinate system. The x,y coordinate of a point could be (529482, 5109382) with a measure value of 248. Although many applications use measures to represent increasing linear distances along a line (see the graphic above), measure values can arbitrarily increase, remain constant, or decrease.

Like Z values, geometry can contain a measure value (x,y,m). You could also have four-dimensional geometry with Z and M values stored with geometry (x,y,z,m).

### ArcSDE annotation

Annotation is text that labels features for cartographic display. It helps identify places and features. For example, a road map can have labels that identify the names of the roads, the distances between intersections, and so on.




*The street name and the addresses are attached to the street and parcel features, respectively.*

Annotation is a feature attribute. ArcSDE stores annotation as a feature attribute to ensure a direct link between the text and the feature it labels. Annotation properties are stored in a single BLOB (binary) column as part of the feature table, or they can be stored in one or more related tables. See the ArcSDE Developer Help for more details on annotation.

Annotation is usually thought of as text attached to a feature or coordinates on a map. Maps often include other text to label nongeographic map components such as key legends, map titles, North arrows, and scale bars. This text has no geographic coordinates and isn't stored by ArcSDE.

ArcSDE annotation is not the same as ArcGIS geodatabase annotation. Please refer to *Building a Geodatabase* to create and maintain geodatabase annotation. ArcGIS Desktop applications can view ArcSDE annotation but cannot edit it. If you need to edit your annotation with ArcEditor, for example, you will need to convert your ArcSDE annotation to the ArcGIS geodatabase annotation format.

### CAD data

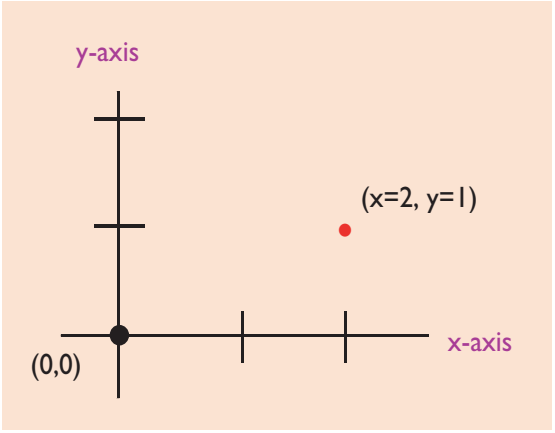
ArcSDE layers can also contain CAD entities, with the aid of ArcSDE CAD Client. When a CAD entity is stored in a layer, an ArcSDE feature representing the entity is also stored. Only ArcSDE CAD Client can work with CAD entities, but other ArcSDE client applications can use the corresponding features. Refer to Using ArcSDE CAD Client (usingcadclient.pdf in the documentation folder) for more information.

### Coordinate reference

Each ArcSDE feature class contains coordinate reference information for your data. The coordinate reference includes the coordinate system and the information needed to convert from real-world coordinates to internal ArcSDE storage values. ArcSDE stores its coordinates as positive integer values internally because they take less room to store in the database and speed calculations.

The position of spatial data is defined by its coordinate system, usually projected (planar) or geographic.

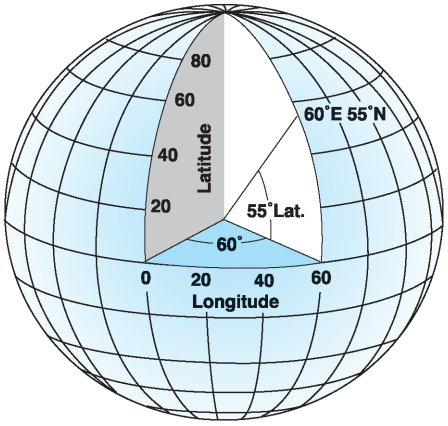
The projected coordinate system, usually measured in meters or feet, defines locations on a two-dimensional plane using two axes: the x-axis, representing east–west; and the y-axis, representing north–south. They intersect at the origin (0,0). Locations are defined relative to the origin.




Points above the x-axis or to the right of the y-axis have positive values. Points below or to the left are negative.

Measures and z-coordinates are referenced independently of the x,y coordinate system, allowing you to assign to them whatever values you wish.

The geographic coordinate system defines locations on a spheroid, a three-dimensional surface. When storing geographic coordinates, longitude values correspond to x, while latitude values correspond to y.




*Understanding Map Projections*, one of the books in the ArcGIS documentation set, is a crucial book for anyone needing an indepth understanding of coordinate reference.

## Geometry storage options

ArcSDE manages the physical storage of geometry for features using standard data types provided by the host DBMS. Some DBMSs have spatial data types, while others provide standard binary or binary large object (BLOB) storage types.

ArcSDE geometry storage depends on the DBMS you use. The options for each DBMS are:

- ArcSDE compressed binary. This is stored as "long raw" in Oracle and as type "image" in SQL Server.
- Oracle Spatial normalized schema (relational model). In the normalized geometry schema, the coordinate values for a geometry are stored as DBMS numeric data types in a separate geometry table. Access to a geometry is through a foreign key—the Geometry ID, or GID. Only available if using Oracle, this data storage schema can result in multiple rows of data for a single geometry, depending on how many x,y points define the geometry (for example, the coast of Norway might occupy many rows). This implementation conforms to the normalized geometry model defined by the OpenGIS Simple Features Specification for SQL.
- Oracle Spatial geometry type. In this object-relational model, Oracle 8*i* extends the database model to include an SDO_GEOMETRY object in the Oracle DBMS.
- Oracle LOB data. LOB data is stored as type BLOB and is supported primarily for Oracle Replication Services.
- Spatial types. A spatial type embeds support for GIS feature geometry into the DBMS kernel. Some DBMSs that support spatial types comply with the OpenGIS SQL specification for user-defined types (UDTs) and the ISO SQL Multimedia Spatial Standard. These standards define columns capable of storing spatial data, such as the location of a landmark, a street, or a parcel of land. Use of these spatial types integrates geometry and nonspatial attributes, providing a single point of access inside the DBMS through an SQL API. Informix and IBM DB2 support spatial types.

ArcSDE client applications see the data as feature layers, regardless of the geometry storage type. In the case of Oracle, you have the option to choose the storage methods for any feature class. You may choose to store a point layer as Oracle Spatial geometry types and a polygon layer as ArcSDE compressed binary. The decision on how to store your geometry should be based on the DBMS you use and the requirements specific to your implementation.

Details on these geometry storage types and how you can define the storage type before loading data can be found in the respective configuration and tuning guides for each DBMS. Please check in your documentation folder for a PDF file of the form config_tuning_guide_<your dbms>.pdf.

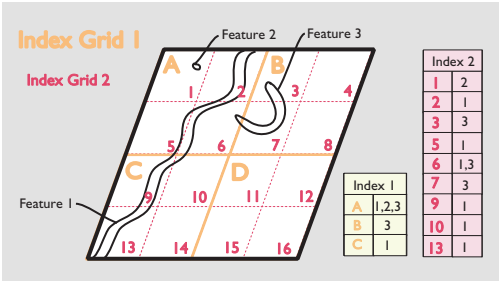
| DBMS | GEOMETRY STORAGE | COLUMN TYPE |
|---|---|---|
| Oracle | ArcSDE Compressed Binary | Long Raw or LOB |
| | LOB | BLOB |
| | Oracle Spatial—Normalized Schema | Number |
| | Oracle Spatial—Geometry Type | SDO_Geometry |
| Microsoft SQL Server | ArcSDE Compressed Binary | Image |
| Informix | Spatial DataBlade—Geometry Object | ST_Geometry* |
| IBM DB2 | Spatial Extender—Geometry Object | ST_Geometry* |

*Summary of the geometry storage and column type options available by DBMS.*

**ST_GEOMETRY is a superclass of several subclasses (e.g., ST_polygon).*

ArcSDE uses a continuous data model; it doesn't tile or split your data. For example, you could store all the parcels for the United States in one continuous feature class. Each parcel would be stored as a complete polygon, quickly retrieved by a single disk access.

To support the use of databases with millions of spatial records, ArcSDE spatially indexes features in each feature class for rapid search and retrieval. ArcSDE builds a spatial index by applying a grid to the feature class. It records features that fall within each grid cell in an index table (the S table of the feature class schema). A feature that falls in more than one cell is listed in each. Grid cells with no data are not included in the table.




*The feature class is overlaid by grid cells to create the spatial index.*

A layer of data may have up to three index grids of different resolutions, although in practice a single grid is usually sufficient.

### Other spatial indexing methods

There is more than one way to create a spatial index. The ArcSDE spatial indexing does not apply to all databases or geometry storage methods. Oracle Spatial and Informix use other methods for spatial indexing. Please refer to the configuration and tuning guide for your database for more information.

Spatial indexing is an advanced, but important, topic. Other performance variables notwithstanding, a poorly defined spatial index can make data retrieval unacceptably slow. The ArcSDE spatial indexing method and some guidelines on how to determine your spatial index (grid size) are described in detail in the configuration and tuning guide for your database (config_tuning_guide_<your dbms>.pdf) in the documentation folder of your software install location.

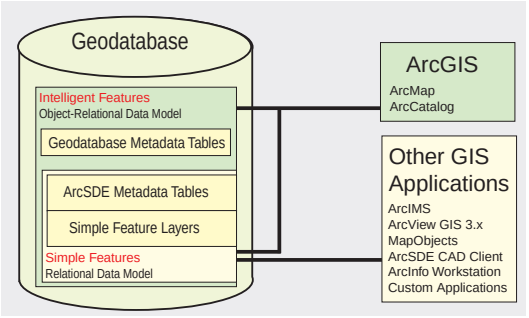
| DBMS | GEOMETRY STORAGE | SPATIAL INDEXING METHOD |
|---|---|---|
| Oracle | ArcSDE Compressed Binary | ArcSDE grid |
| | LOB | ArcSDE grid |
| | Oracle Spatial—Normalized Schema | SDO |
| | Oracle Spatial—Geometry Type | RTREE |
| Microsoft SQL Server | ArcSDE Compressed Binary | ArcSDE grid |
| Informix | Spatial DataBlade—Geometry Object | RTREE |
| IBM DB2 | Spatial Extender—Geometry Object | ArcSDE grid |

*Summary of the geometry storage and spatial indexing methods available by DBMS.*

There is a great deal of metadata that must be maintained about spatial data in a DBMS. ArcSDE and the ArcGIS geodatabase have a set of metatables for managing this spatial data. However, these metatables should never be managed or edited directly. Only ArcSDE populates and manages the ArcSDE tables, and only ArcGIS Desktop manages the geodatabase tables. End users never need to be aware that these tables exist. More information about these tables may be found in the ArcSDE Developer Help and is provided for database administrators (DBAs).

### Relational view and object relational view

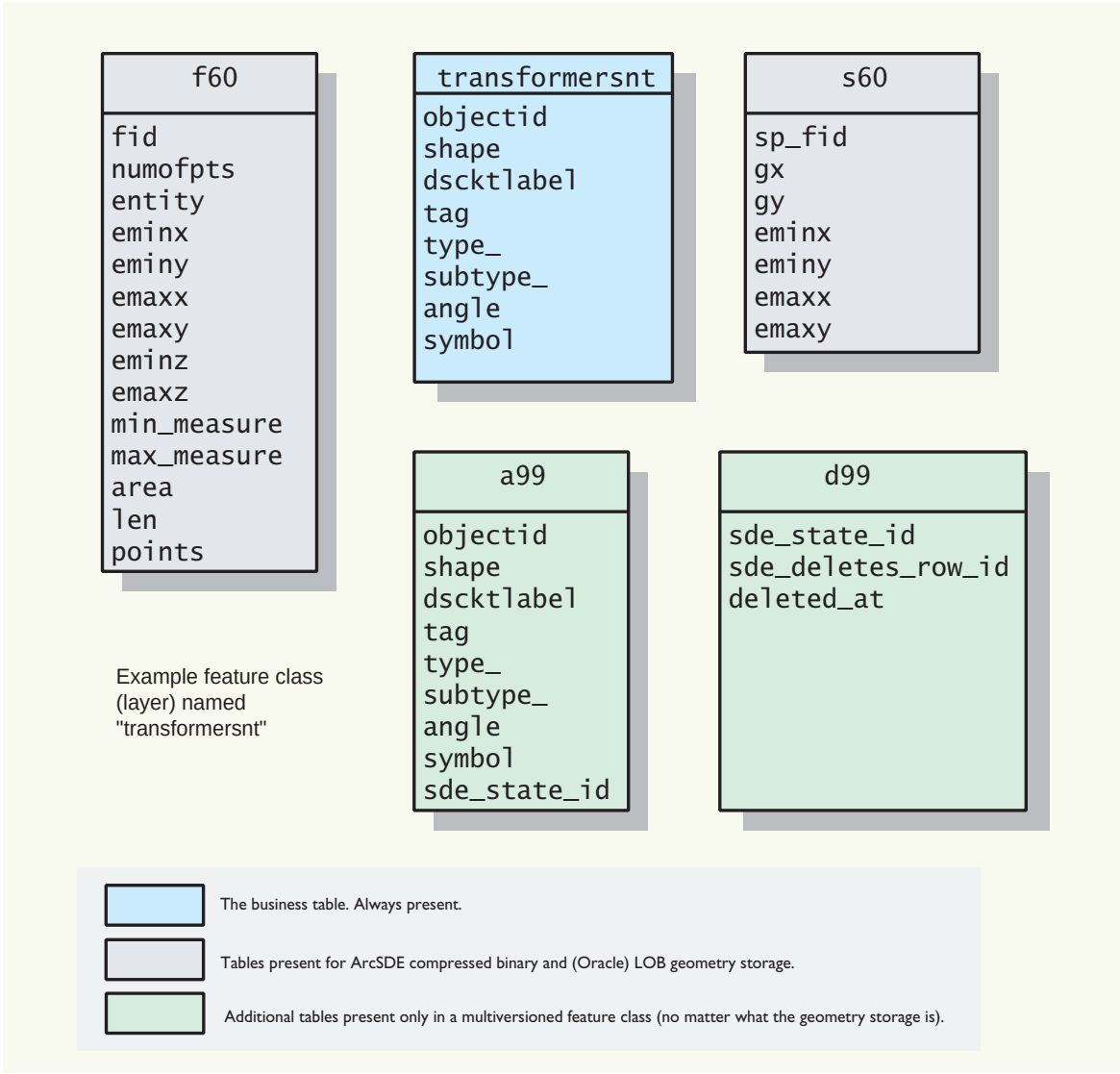
You might logically think of the ArcSDE and ArcGIS geodatabase system tables as one set of metatables. However, there is an important distinction, based on the type of access an application has to the data. Some applications have relational access to data in a geodatabase, while others have object relational access. Relational access is access to the simple features of ArcSDE. Object relational access is access to the simple features plus the intelligence for the features stored in the geodatabase metatables.




*Relational access and object relational access depend on whether the client application has access to the geodatabase metatables.*

The geodatabase metatables are accessible and used only by applications with object relational access—the ArcGIS Desktop products. This means that applications like ArcView GIS 3.x, MapObjects 2.x, ArcInfo 8 Workstation (e.g., ArcPlot™), ArcSDE CAD Client, and third party applications built with the C or Java APIs cannot take advantage of the data stored in those geodatabase metatables. However, these applications do have access to the underlying simple features.

It is also important to note the ArcGIS family of products can use the simple features without having to populate any of the geodatabase metatables first. This use is limited to basic display and query functionality.




*Example schema of a single feature class for geometry. The color scheme indicates which tables are present for different types of geometry storage. The "f" table contains the geometry (the actual x,y values) of features. The "s" table is the spatial index table for Microsoft SQL Server, Oracle with LOB, and ArcSDE compressed binary storage. Oracle Spatial-, Informix-, and DB2-based feature classes do not have the "f" or the "s" table. All multiversioned feature classes (layers) will have an "a" table for data additions and a "d" table for all deletions. The "60" in the table names is the internal layer ID, and the "99" is the internal registration ID.*

Raster data represents a significant portion of the total data used in a GIS. ArcSDE provides support for rasters in a number of different formats.

ArcSDE handles raster data very much like vector data. When a business table is created with a column of raster type, ArcSDE will reference this image column as an image (raster) dataset. Image datasets can have multiple images, but georeferenced images cannot be combined in a single image layer with nongeoreferenced images. Information about the raster column is maintained in one of the ArcSDE system tables called raster_columns.

Here's a brief description of the tables in raster layer schema.

For each raster type column in a business table, ArcSDE will automatically create four additional tables. They are:

1. Metadata table for raster (SDE_RAS_<id#>)

2. Metadata table for raster band (SDE_BND_<id#>)

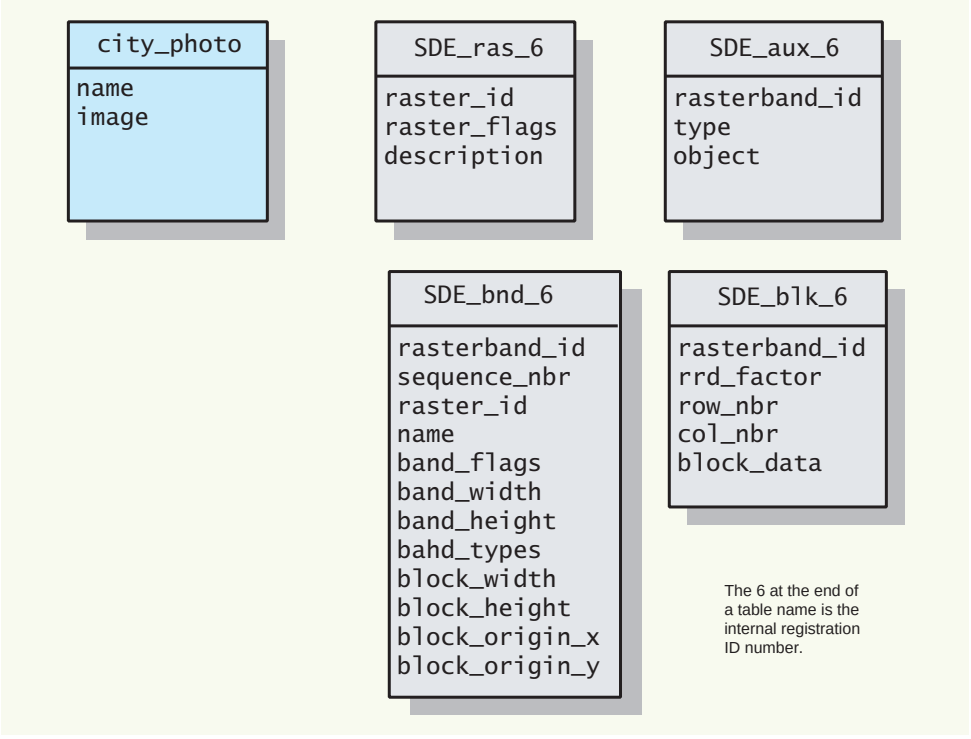
3. Auxiliary table for raster band (SDE_AUX_<id#>)

4. Block table (SDE_BLK_<id#>)

The first two metadata tables are used to store information about a raster and a raster band such as the image dimensions and the pixel depth.

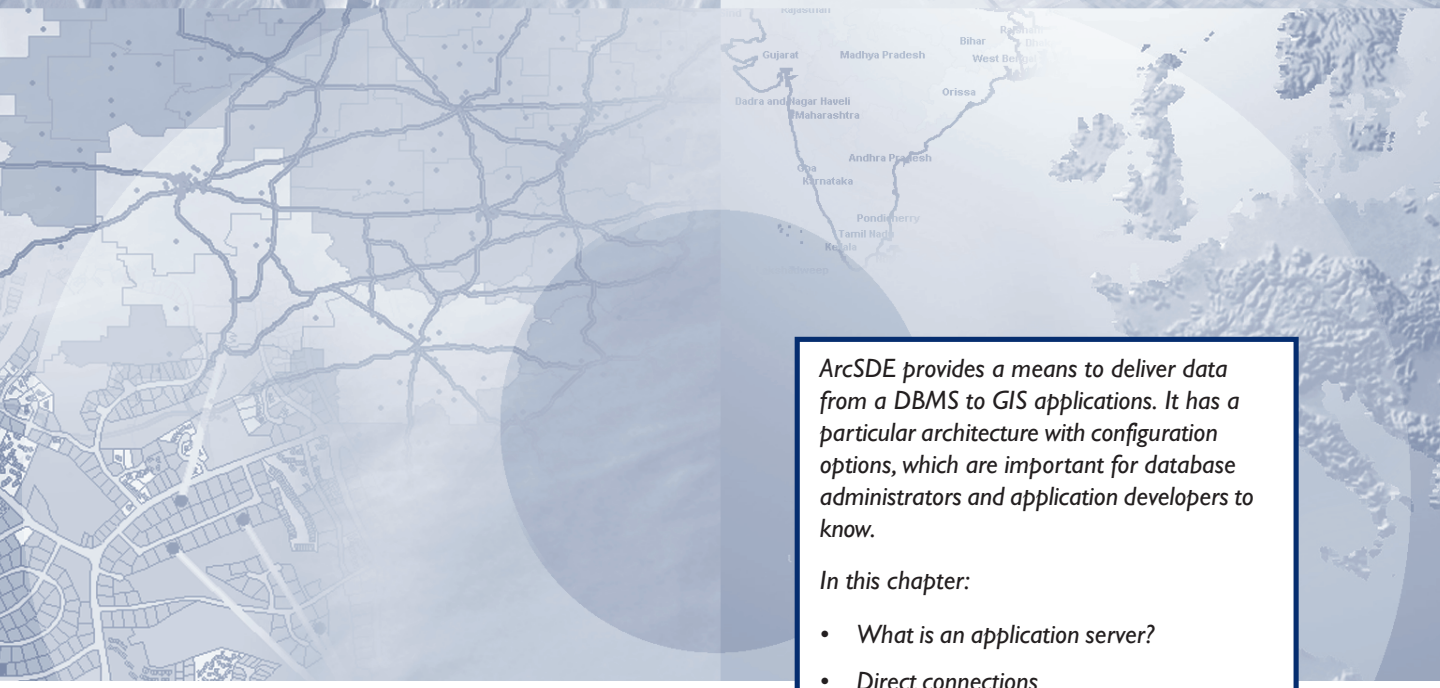
The auxiliary table is used to save additional information about a raster band such as a colormap and raster statistics. The block table is where the actual pixel blocks are stored.

The best way to put raster data into your DBMS is with the ArcCatalog Raster to Geodatabase Wizard.




*Tables that define an ArcSDE raster layer*

# 4 The ArcSDE architecture




*ArcSDE provides a means to deliver data from a DBMS to GIS applications. It has a particular architecture with configuration options, which are important for database administrators and application developers to know.*

*In this chapter:*

- *What is an application server?*
- *Direct connections*
- *Connecting with a direct connect driver*
- *Licensing*

Architecturally speaking, there are two basic configurations you can use with ArcSDE implementations. You may choose a three-tier architecture, which uses an application server; a two-tier architecture, which uses what are called direct connect drivers; or a combination of the two.

### The application server

ArcSDE is built with client/server technology—a client application sends requests to the server. In turn, the server receives the request, generates results, and delivers them to the client.

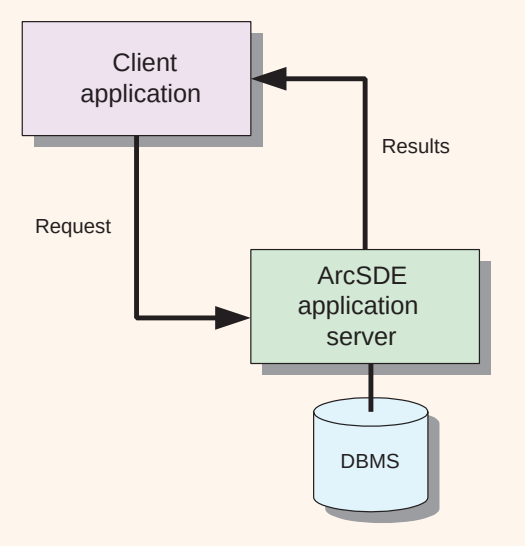
The ArcSDE server disseminates spatial data based on highly efficient spatial search functions, provides geometric data validation, and works within heterogeneous hardware and network environments. Data can be delivered to any client from any server anywhere on a network.

In a typical configuration, an ArcSDE application server resides with your relational database on a server platform. The ArcSDE application server performs spatial searches and sends data that meets the search criteria to the client. For example, a common query handled by the ArcSDE application server is to retrieve all the features in a particular map extent to be drawn in the display window.

ArcSDE sends data to the client using "data buffering". Buffering is the process of collecting large chunks of data and sending it all to the client application, rather than sending one record at a time. Processing and buffering data on the server is much more efficient than sending all of the data across the network and having the client determine which data to send to the end-user application. This becomes critical when applications are simultaneously using thousands of records in the database.

ArcSDE uses cooperative processing, which means that data processing occurs on both the client application machine and the server, depending on which is faster. Some functions require no communication with the server. CPU-intensive tasks such as polygon overlay and clipping are best performed by the client application to avoid excessive demands on both the server and the available bandwidth.

The computer network connects many clients to the server. The network must support TCP/IP. It can be a low-speed wide area network (WAN) or a fast local area network (LAN). Network file system mounts are not required for data transfer between the server and the client. This is an important performance and administrative benefit.

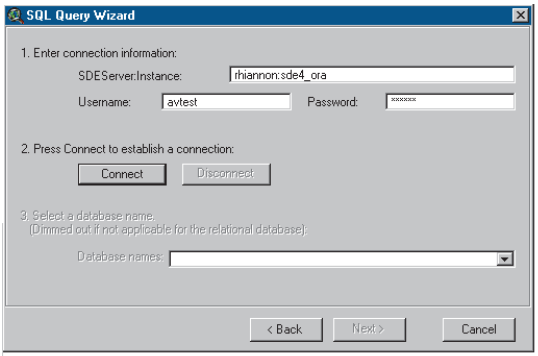



*The ArcSDE server handles simultaneous requests from multiple users to update and retrieve information in a geodatabase.*

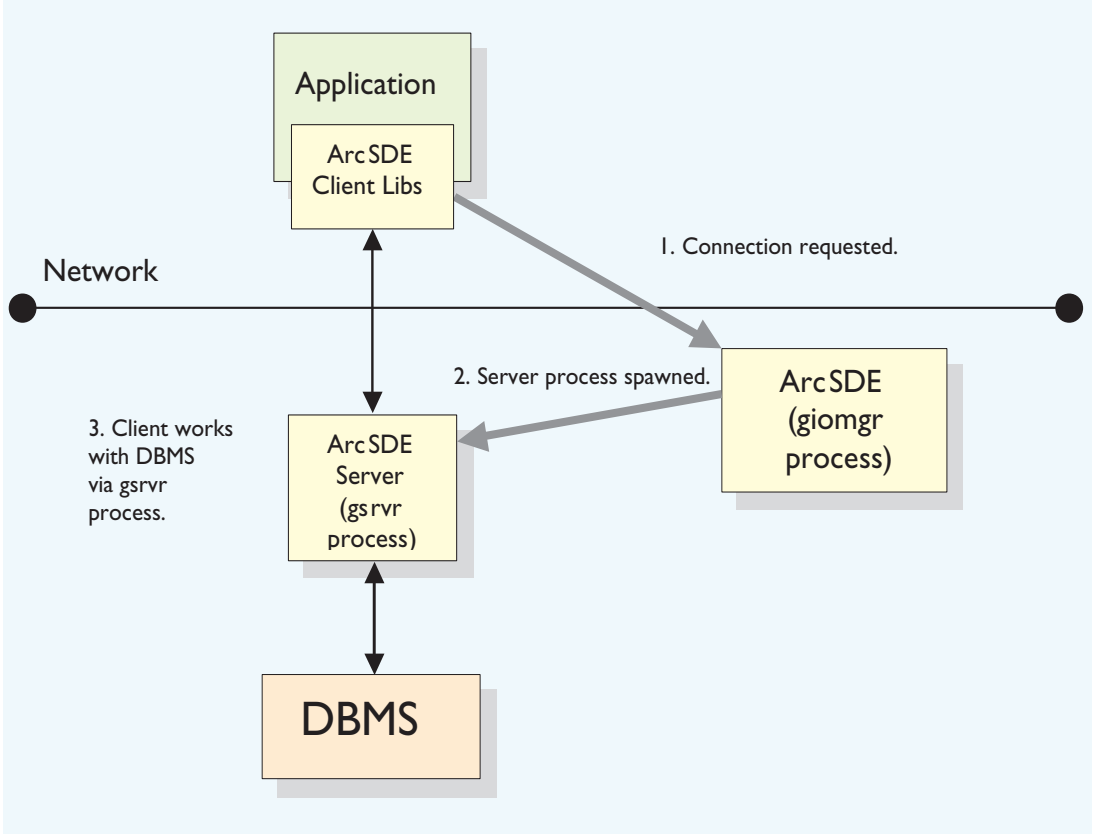
### The application service connection process

Geodatabase security is managed via the DBMS. Therefore, accessing data requires a "connection" to the host DBMS. A user must enter connection information including a username and password. This connection information, passed on to the ArcSDE server, is used to log in to the DBMS. If the login connection is successful, the user can then begin working with the geodatabase.

The graphic below illustrates the basic connection process. The giomgr process is the "service" that a user connects to. Creating and managing services is described in *Managing ArcSDE Services*. Typically, the ArcSDE service runs on the same machine as the DBMS server. The gsrvr process always runs on the same machine as the giomgr process. Each connection request starts a gsrvr process. Each gsrvr process will use a DBMS connection.




*The connection wizard in ArcView GIS shows a user named "avtest" connecting to a service named "sde4_ora" on a server machine named "rhiannon". The database input field is for connecting to specific DBMSs that require a database name (e.g., Microsoft SQL Server).*



Prior to the 8.1 release, ArcSDE only had one basic architecture: a three-tiered one. The three tiers consisted of a client, the ArcSDE application server, and the host DBMS. The client applications had to connect to the ArcSDE application server, which then connected to the host DBMS. With 8.1, ArcSDE can also be used without the application server. Essentially, there is a two-tier architecture. The two-tier and three-tier systems are designed to work independently, or together, giving the DBA flexibility in system design. When using the two-tier direct connection architecture, you will use what are called "direct connect drivers".

Using the two-tier architecture, an application connects directly to the database, without using the ArcSDE application server. The ArcSDE client and server functionality are executed on the same machine as the application—there is no separate ArcSDE server process running anywhere.

The direct connection configuration is often easier to install and administer. There is no need for installing and administering the ArcSDE application server. This configuration allows for increased scalability because it off-loads work from the server to each connected client.

The two-tier architecture provides additional flexibility in configuring systems:

• The direct connect database driver doesn't require the administration of the ArcSDE server process. There is no intermediate giomgr process to set up/configure or start up. Setting up a single-user database using Personal Oracle or the Microsoft Database Engine (MSDE), for example, is easier.

• The direct connect database driver provides additional options for scaling a system. The direct connect architecture moves the ArcSDE server functionality to the desktop. This removes the ArcSDE load from the database server and will allow additional resources to be freed up for the DBMS, which means you get better scalability on the database server.

• The direct connection configuration can be very useful in "hot-failover" environments. For example, it's easier to configure ArcSDE with Oracle Parallel Server with the direct connect driver.

**Are there any reasons why you wouldn't want to use the two-tier direct connect architecture?**

The following reasons might apply:

• Increased network traffic. This can happen if the spatial filter selectivity is low—for example, finding a very small number of features adjacent to a long sinuous feature that crosses a large area. Increased network traffic can result in poorer overall performance, but the severity of the degradation depends on the bandwidth of your network.

• You do not want the administrative overhead of networking software (e.g., Net8 for Oracle).

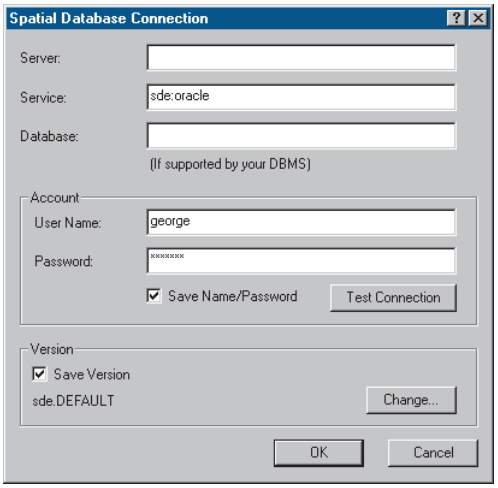
• If you are building a thin client, executing the ArcSDE server process on the client machine may not be desirable.

• If you have a low-end desktop computer with limited memory and/or chip speed, you may need to use the application server to move that server functionality off of your client.

The application server configuration offers many performance advantages and is generally faster in a mixed UNIX®/Windows® network because of the intelligent ArcSDE client/server architecture.

Regardless of whether you configure your system with the application server, direct connect drivers, or a mixture of the two, the same client functionality is available to any 8.1 client application including those built with the C API. All ESRI client products are delivered with direct connect drivers.

Using the direct connect drivers requires some setup before users can begin connecting. The same ArcSDE user, ArcSDE system tables, and geodatabase system tables used by the application server configuration must also exist for direct connection configurations. Your administrator must set these up prior to any ArcSDE connections. If you are using Oracle, you must also have the Net8 Client installed on your client machine. Please read the ArcSDE installation guide closely for setup information as well as supported databases and hardware operating systems. Please refer to Oracle documentation for Net8 installation and configuration.



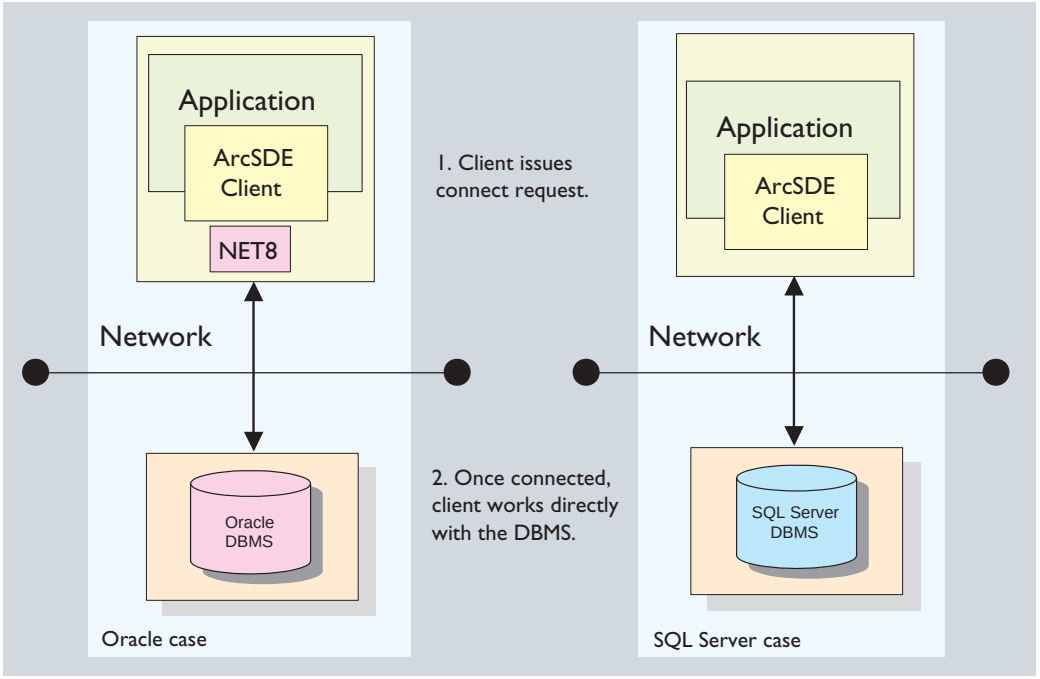

*Here's an example from ArcGIS for an Oracle direct connection. The service value must be sde:oracle, and the Net8 service name is added to the password. The server field can be left blank.*

Client applications specify whether they are connecting to the direct connect driver or the application server via the connection syntax. For example, the service value is

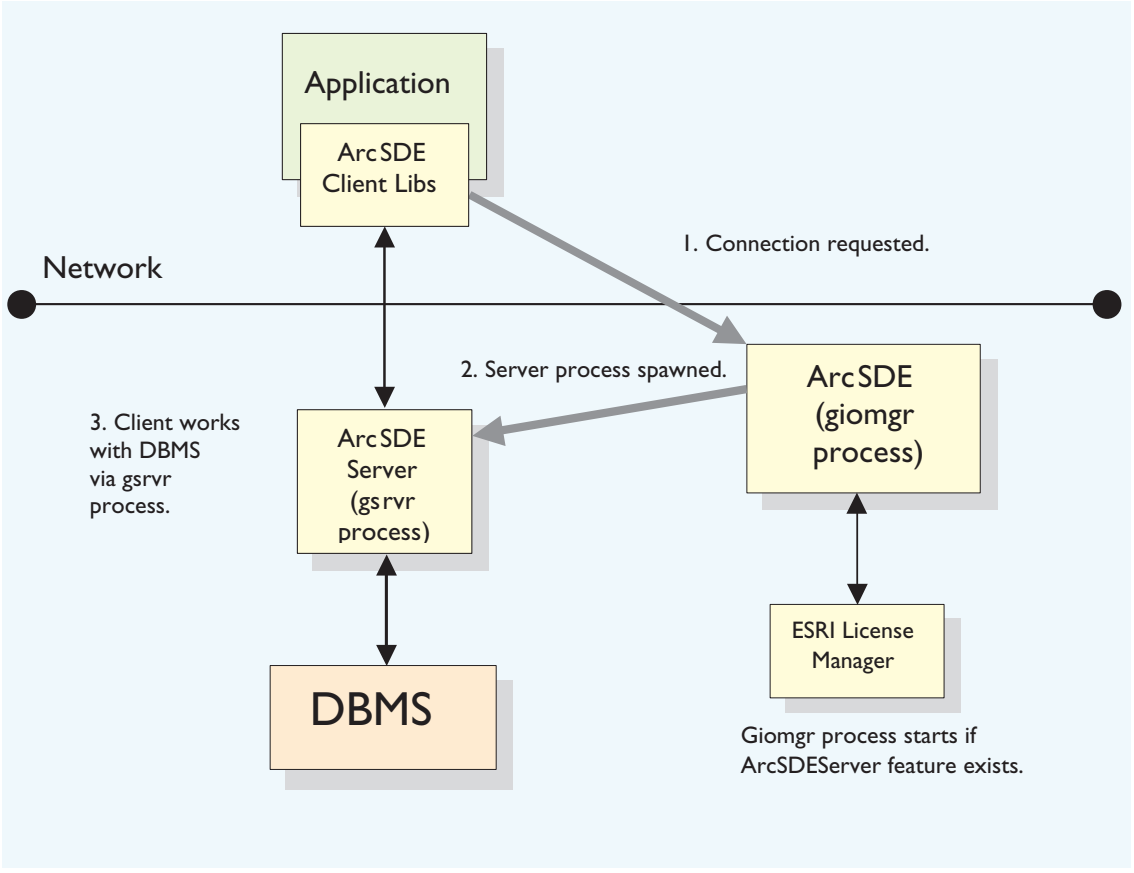
```
sde:oracle
```

for the Oracle driver; no server name is required. In the case of Oracle, you'll also have to provide the Net8 service name.

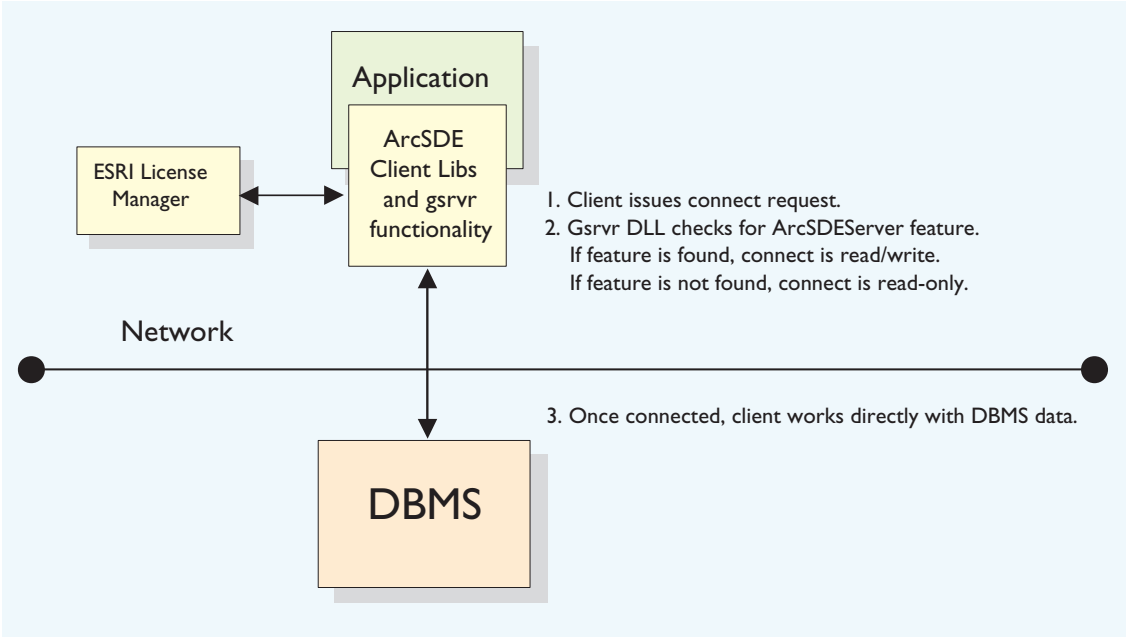
Below is a graphic illustrating the two-tier architecture. The gsrvr functionality is on the client in the form of a dynamically linked library. The connection is still via a DBMS account, and DBMS security is still used. Once connected, the client works directly with the DBMS without an intermediate application server process.



ArcSDE is a licensed product. Your administrator will need to install a license manager and get a keycode for ArcSDE to work. A license is required for both the application server and the direct connect driver. However, in the case of the direct connect driver, if no license is found, the application is granted a read-only connection no matter what functionality the application provides. Please refer to the LicenseManagerGuide.htm file found in your documentation folder for more information on all licensing.

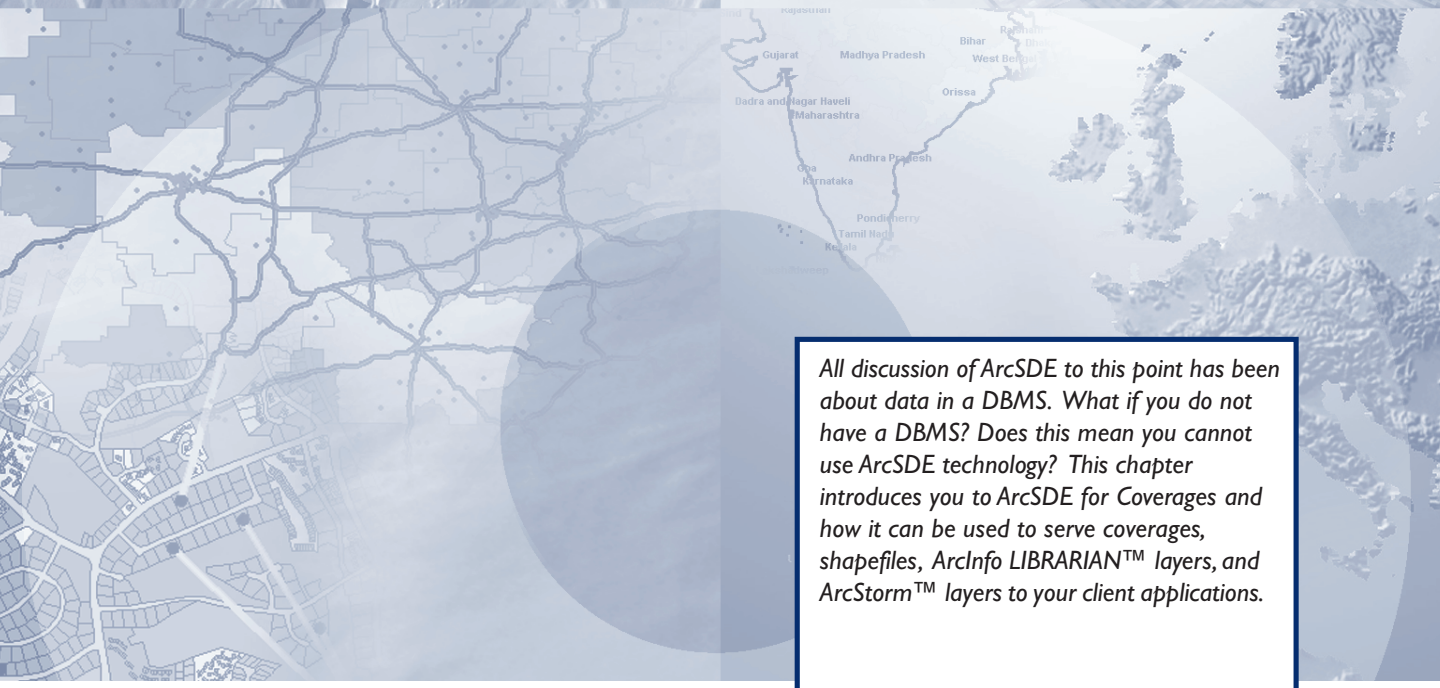



*This illustration is of a three-tier architecture. The application server starts when a license is found. The ArcSDE license is an "enabling" license. This means that once licensed, any number of application servers and/or user connections can be made. The license manager does not count them.*




*This illustration is of a two-tier architecture. The client connection looks for an ArcSDE server license. If it doesn't find one, the connection is read-only no matter what editing functionality the application may provide or what the underlying DBMS permissions may be.*

# 5 ArcSDE for Coverages




*All discussion of ArcSDE to this point has been about data in a DBMS. What if you do not have a DBMS? Does this mean you cannot use ArcSDE technology? This chapter introduces you to ArcSDE for Coverages and how it can be used to serve coverages, shapefiles, ArcInfo LIBRARIAN™ layers, and ArcStorm™ layers to your client applications.*

While ArcSDE provides the gateway to numerous commercial relational DBMSs, another available ArcSDE application server serves file-based spatial datasets. This read-only server, called ArcSDE for Coverages, serves the following file-based vector datasets:

- ArcInfo coverages
- ESRI shapefiles
- ArcInfo LIBRARIAN layers
- ArcStorm layers

Many sites have file-based data on UNIX file servers and use Windows client applications. Mounting disks on each Windows machine is administrative overhead, and performance may be poor. ArcSDE for Coverages allows network file system (NFS)-less access of the UNIX file server data, and it is often significantly faster than NFS access.

ArcSDE for Coverages is crucial to providing access to file-based spatial datasets for all ArcSDE client applications. MapObjects and ArcIMS can only access ArcInfo LIBRARIAN and ArcStorm data via ArcSDE for Coverages. In addition, ArcIMS depends on ArcSDE for Coverages to provide access to ArcInfo coverage data.

ArcSDE for Coverages can also be part of your migration strategy from file-based data sources to geodatabases. The most direct way to put ArcInfo LIBRARIAN and ArcStorm layers into feature datasets of the geodatabase is by using the ArcSDE for Coverages server. An ArcSDE for Coverages layer can be directly loaded into a geodatabase using ArcCatalog. Alternatively, you can also use the ArcSDE command cov2sde to put ArcInfo LIBRARIAN or ArcStorm data into standalone feature classes in the geodatabase. However, further work is required to migrate these standalone feature classes into a feature dataset. Manually creating an intermediate coverage from an ArcInfo LIBRARIAN or ArcStorm layer and then loading that coverage into the geodatabase may take considerable time and disk space.

Customized applications built for ArcSDE for Coverages can still be used if you migrate your data to a DBMS version of ArcSDE, although attribute queries will need to be rewritten in SQL for your DBMS.

### How does ArcSDE for Coverages work?

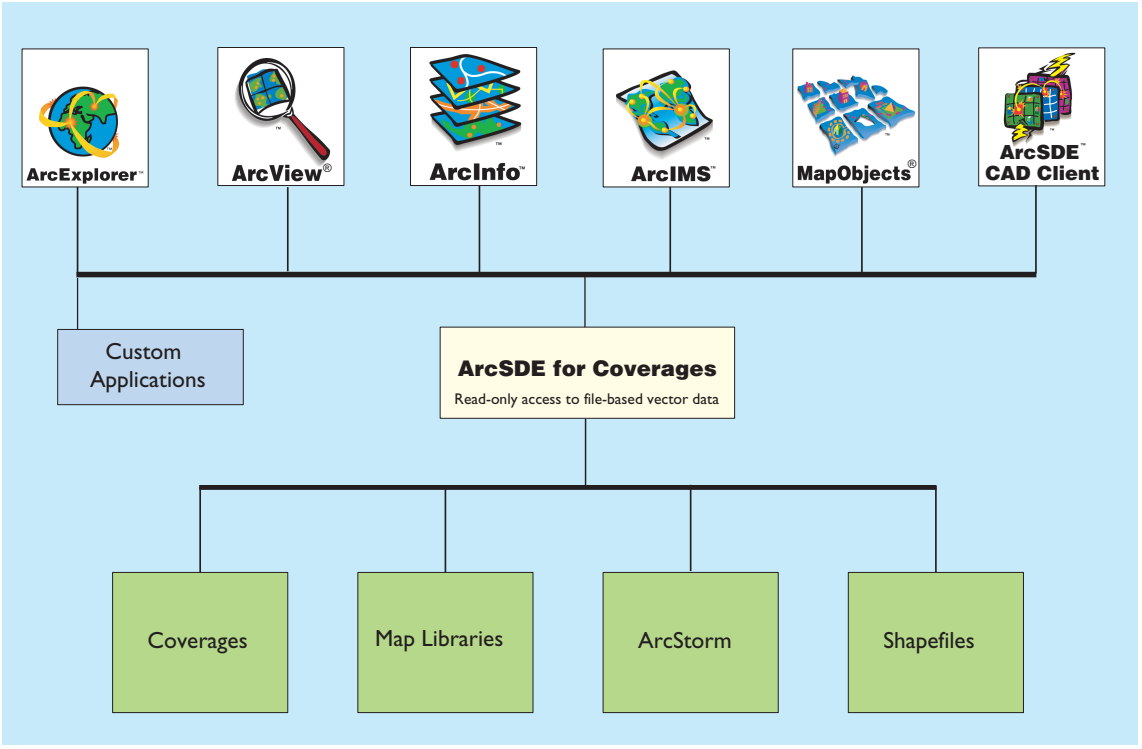
Your data is not copied, loaded, or otherwise moved anywhere. When your ArcSDE for Coverages server is running, there is a utility provided to "register" data with the ArcSDE server. Essentially, you tell the server where the data is, provide a layer name, and set a few properties. At that point, the data can be served to a client application. You can also set up relates to attribute data in a DBMS.

In the case of data split by tiles (ArcInfo LIBRARIAN and ArcStorm line and polygon data), the data is "sewn together" on the server if there is a unique ID for each feature. Once a whole feature is assembled, it is sent to the client.

Data transfer is the same as with the DBMS version of ArcSDE. Data is delivered to clients via TCP/IP. This means you need to define a service with a specific name and TCP/IP port number in your services file.

ArcSDE for Coverages is unlicensed and provided free of charge with each ArcGIS and ArcIMS release. This makes it an inexpensive way to begin working with ArcSDE technology.

For more information about setting up an ArcSDE for Coverages server, refer to the book *ArcSDE for Coverages Administration Guide*, now in PDF format (arcsde_cov_admin_guide.pdf), located in the documentation folder of the ArcSDE 8.1 for Coverages CD.




*ArcSDE for Coverages acts as a read-only TCP/IP gateway to your file-based data.*

# 6 Where to go next

*Now that you've read about ArcSDE, what's your next step? Many different kinds of people use ArcSDE. This chapter discusses the various types of work you can do with ArcSDE and what documents you should consult for more information. Four types of people interact with ArcSDE, each with a different role: database administrators, GIS administrators, application developers, and end users.*

*In this chapter:*

- *Administrators*
- *Developers*
- *End users*
- *ArcSDE resources*

To implement a multiuser GIS, two key roles must be filled: the database administrator and the GIS administrator. Sometimes these are the same person; sometimes multiple people are involved.

### The database administrator

The database administrator is responsible for managing the data being served by ArcSDE. When using the ArcSDE application server, the DBA also manages that server and its interface to an organization's relational database. The goal of *Managing ArcSDE Services,* along with the ArcSDE installation guide (Install.htm link) and ArcSDE configuration and tuning guide for <your DBMS> (config_tuning_guide_<your dbms>.pdf), is to help database administrators perform a number of key tasks and procedures including:

- Installing the ArcSDE application server
- Configuring the ArcSDE application server for the network
- Starting, stopping, and monitoring ArcSDE server processes
- Tuning the relational database and optimizing ArcSDE performance on the network
- Setting up ArcSDE connections and user access to the database

The ArcSDE application server is often installed on large multiuser computer networks. Managing an ArcGIS system installation is another task for database administrators and requires a good understanding of the key parts of the system. The DBMS and data are located on a UNIX or Windows NT® server. The ArcSDE application server is also typically located with your relational database, while the ESRI client applications are distributed on many machines across your network. All of the parts of the ArcGIS software system use a license manager to control access to its various applications. To effectively manage this system, you should refer to the ArcGIS Desktop install guide, the ArcInfo Workstation install guide (Install.htm link), and the LicenseManagerGuide.htm on the ArcInfo CD.

### The GIS administrator

The GIS administrator is responsible for designing and implementing an organization's geodatabase. The GIS administrator models a spatial system of features, surfaces, images, and other geographic information to design a geodatabase that supports the work of the organization. This is a key role for a successful multiuser GIS.

It's important to understand the spatial system to be modeled—what its objects are and the relationships between the various objects. For example, designing a geodatabase for land records requires an understanding of the relationships between parcels, surveys, ownership, and important administrative units such as zoning and easements. It also requires a complete understanding of geodatabase concepts and the best practices and methods for GIS database design.

The GIS administrator must be knowledgeable about geodatabase structures and how to use them to represent a real-world system. The GIS administrator must also understand how to apply computer tools, such as UML models and the ArcGIS system, to effectively implement a geodatabase design.

Two key books are included in the ArcGIS 8 package to help you accomplish these tasks:

- *Modeling Our World—The ESRI Guide to Geodatabase Design* introduces you to geodatabases and their contents. It can help you make decisions about the best strategies to employ in your geodatabase design. It describes in detail the objects and various parts of a geodatabase. *Modeling Our World* will help you with the design process and the key decisions you'll need to make along the way.
- *Building a Geodatabase* is the GIS administrator's guide to implementing a specific geodatabase design. Once you have completed your physical database design, this book contains all the necessary guidance to implement your geodatabase schema. It provides information on using ArcCatalog for these tasks and for using UML. UML is an excellent tool for object modeling and design. It is very useful in implementing an integrated design for an entire business database including the geodatabase components.

If you haven't done so already, you might also want to read *Getting Started with ArcGIS*. It gets you started using the software immediately and refers to further ArcGIS documentation. *Getting Started with ArcGIS* contains an overview of the ArcGIS software system and has a series of exercises that lead you through a small GIS project.

Application developers have a number of opportunities to work with geodatabases. In the ArcGIS system, you can build custom applications or simply adjust the look and feel of existing applications with Visual Basic® for Applications (VBA) within the ArcMap and ArcCatalog applications. ArcInfo 8 and ArcView 8.1 also ship with ArcObjects—the complete set of COM-based components for the ArcInfo and ArcView 8.1 software. The ArcInfo superset of COM objects contains hundreds of COM classes with thousands of methods. *Exploring ArcObjects* is your guide to begin to work with ArcObjects.

Other ESRI software such as ArcView GIS 3 and MapObjects 2 can use the ArcSDE application server directly. In these systems, you can create custom applications that access geodatabases through ArcSDE. Each software provides its own ArcSDE programmer's interface that you can take advantage of as a developer. The ArcView GIS 3 object-oriented programming language, Avenue, can access ArcSDE functionality in addition to customizing the look of ArcView GIS 3. The subsystem that accesses and works with the ArcSDE application server is the Database Access extension. Refer to the book *Using Avenue* in the ArcView GIS package for additional guidance.

You can also develop applications with MapObjects using VB, Visual C++®, and other COM-based developer tools. A part of the MapObjects 2 developer components describes building applications that access and work with ArcSDE. Two MapObjects user guides have more information on building MapObjects applications that use the ArcSDE application server: *Developing Applications with MapObjects* and *MapObjects Programmer's Reference*.

With the ArcSDE CAD Client extension, you can work with true CAD entities in a geodatabase. The ArcSDE CAD Client extension has an extensive C language API that has complete access to all ArcSDE functionality. For more information, see Using ArcSDE CAD Client, included as a PDF file (usingcadclient.pdf) in the documentation folder on the ArcSDE 8.1 Client CD.

The ArcSDE application server provides an open client interface to work with the contents of geodatabases as simple features. The ArcSDE simple feature API allows non-ESRI software applications to access and work with geodatabases. The ArcSDE client API is available in both C and Java. Refer to the ArcSDE Developer Help system located in the documentation folder on the ArcSDE CD.

Typically, end users are those using a client application such as ArcMap. Most end users do not know or even need to know they're using ArcSDE to access data in a DBMS. Working with a multiuser geodatabase is just like working with any other geographic data—use ArcCatalog to explore and work with data holdings and use ArcMap to map, edit, and analyze data from the geodatabase.

If you haven't done so already, you might want to read *Getting Started with ArcGIS*. It gets you started using the software immediately and refers to further ArcGIS documentation. *Getting Started with ArcGIS* contains an overview of the ArcGIS software system and has a series of exercises that lead you through a small GIS project.

To learn more about using shared geodatabases, you should refer to *Using ArcCatalog* and *Using ArcMap* in the ArcInfo user documentation set.

This book is designed to provide an introduction to ArcSDE and a brief explanation of its roles and benefits. For more detailed information about ArcSDE, you are encouraged to explore the additional resources available. These are listed and described below.

### ArcSDE 8

ArcSDE installation guide: An HTML file that contains detailed installation instructions. The file is called Install.htm and is located in the documentation directory of the ArcSDE CD.

*Managing ArcSDE Services*: Available as a printed book.

ArcSDE configuration and tuning guide for <your DBMS>: This is one of the DBMS-specific PDF files (config_tuning_guide_<your dbms>.pdf) in the documentation folder on the ArcSDE CD.

ArcSDE Developer Help: Contains developer information on C and Java and command references for ArcSDE administration commands, located in the documentation folder on the ArcSDE CD.

### ArcSDE CAD Client 2

Using ArcSDE CAD Client: Available as a PDF file (usingcadclient.pdf) in the documentation folder on the ArcSDE 8.1 Client CD.