



# 1

# Object modeling and geodatabases

A geographic data model is a representation of the real world that can be used in a GIS to produce maps, perform interactive queries, and execute analysis.

Contemporary developments in database and software technology is enabling a new generation of geographic data models. These are the topics in this chapter:

- Modeling objects with GIS
- The progress of geographic data models
- The geodatabase, store of geographic data
- Features in an object-oriented data model
- Serving and accessing geographic data
- Building data models
- Viewing and designing geodatabases
- Guide to reading UML object diagrams
- Technology trends

## DRAFT

MODELING OUR WORLD  
THE ESRI GUIDE TO  
GEODATABASE DESIGN

ARCINFO 8.0  
PRE-RELEASE

JULY 23, 1999

COPYRIGHT © 1999  
ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE, INC.  
ALL RIGHTS RESERVED.

The purpose of a Geographic Information System (GIS) is to provide a spatial framework to support decisions for the intelligent use of Earth's resources and to manage the man-made environment.

Most often, a GIS presents information in the form of maps and symbols. Looking at a map gives you the knowledge of where things are, what they are, how they can be reached through roads or other transport, and what things are adjacent and nearby. A GIS can also disseminate information through an interactive session with maps on a personal computer. This interaction reveals information that is not apparent on a printed map.

For example, you can query all known attributes of a feature, create a list of all things connected from one point on a network to another, and perform simulations to gauge qualities such as water flow, travel time, or dispersion of pollutants.

The information display and analysis you wish to support depends upon how you model geographic objects from the world.

### MANY WAYS TO MODEL A SYSTEM

Our interaction with objects in the world is diverse and you can model them in many ways.

Consider one example, rivers. Rivers are interesting because they are natural features, they are used for transportation, they delimit political or administrative areas, and they are an important feature in the shape of a surface. Here are a few of the many ways you can think about modeling rivers in a GIS:



- As a set of lines that form a network. Each section of line has flow direction, volume, and other attributes of a river. You can apply a linear network model to analyze hydrographic flow or ship traffic.



- As a border between two areas. A river can delimit political areas such as provinces or counties, or can be a barrier for natural regions such as wildlife habitats.



- As an areal feature with an accurate representation of its banks, braids, and navigable channels on the river.



- As a sinuous line forming a trough in a surface model. From the river's path through a surface, you can calculate its profile and rate of descent, the watershed it drains, and its flooding potential for a prescribed rainfall.

### MAP USE GUIDES THE DATA MODEL

It's clear that even a common type of geographic feature such as a river can be represented in a GIS in a variety of ways. No model is intrinsically superior; the type of map you want to create and the context of the problems to be solved will guide which model is best.



# representations of geography

## features

Features are discrete objects on a map. Small objects are represented as points, long objects with lines and broad objects with polygons.

## network

A network is a set of features that participate in a linear system such as utility network, stream network, or road network. Networks are well suited for tracing analysis.

## location

227 East Palace Avenue

The geodatabase stores locators and addresses. A locator interpolates a location from an address using local postal conventions. You can find a geographic location for any address.

## surface

The earth's surface can be kept in a geodatabase in several forms; as a triangulated irregular network (TIN), as elevation values on pixels in a raster, or as contour lines.

## image

Rasters (images) are an efficient technology to capture large amounts of imaged data. They provide an informative background display to feature layers on a map

A geographic data model is an abstraction of the real world that employs a set of data objects to support map display, query, editing, and analysis.

ArcInfo 8 introduces a new object-oriented data model—the geodatabase data model—with the benefit of representing natural behaviors and relationships of features. To understand the impact of this new model, it is instructive to review three generations of geographic data models.

## I THE CAD DATA MODEL

The very first computerized mapping systems drew vector maps with lines displayed on cathode ray tubes and raster maps using overprinted characters on line printers. From this genesis, the 1960's and 70's saw the refinement of graphics hardware and mapping software that could render maps with reasonable cartographic fidelity.

In this era, maps were typically created with general purpose CAD (Computer Aided Drafting) software. The CAD data model stored geographic data in binary file formats with representations for points, lines, and areas. Scant information about attributes was kept in these files; map layers and annotation labels were the primary representation of attributes.

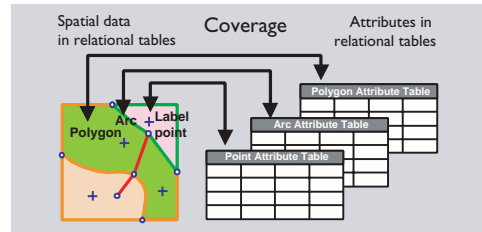
## II THE COVERAGE DATA MODEL

In 1981, ESRI introduced its first commercial GIS software, ArcInfo, which implemented a second generation geographic data model, the coverage data model (also known as the georelational data model). This model has two key facets:

- Spatial data is combined with attribute data. The spatial data is stored in indexed binary files which are optimized for display and access. The attribute data is stored in tables with a number of rows equal to the number of features in the binary tables and joined by a common identifier.
- Topological relationships between vector features can be stored. This means that the spatial data record for a line contains information about which nodes delimit that line, and by inference, which lines are connected, and also which polygons are on its right and left side.

The major advance of the coverage data model was

the ability for the user to customize feature tables; not only could fields be added, but database relates could be set up to external database tables.



Due to the performance limitations of computer hardware and database software of the time, it was not practical to store spatial data directly in a relational database. Rather, the coverage data model combined spatial data in indexed binary files with attribute data in tables.

Despite this compromise of partitioning spatial and attribute data, the coverage data model has become the dominant data model in GIS. This has been for good reason—the coverage data model made high-performance GIS possible and stored topology facilitated improved geographic analysis and more accurate data entry.

### Limitations of the coverage data model

Yet, the coverage data model has an important shortcoming—features are aggregated into homogeneous collections of points, lines, and polygons with generic behavior. The behavior of a line representing a road is identical to the behavior of a line representing a stream.

The generic behavior supported by the coverage data model enforces the topological integrity of a dataset. An example is if you add a line across a polygon, it is automatically split into two polygons.

But it is desirable to also support the special behaviors of streams, roads, and other real-world objects. An example is that a stream flows in one direction and when two stream segments combine, the flow of the downstream segment is the addition of the two upstream flows. Another example is that when two roads cross, there ought to be a traffic intersection at their junction unless one of the roads is an overpass or underpass.

## Customizing features in coverages

With the coverage data model, ArcInfo application developers had some notable success in adding this type of behavior to features through macro code written in the Arc Macro Language (AML). Many successful, large-scale, industry-specific applications were built.

However, as applications became more complex, it became apparent there needed to be a better way to associate behavior with features. The problem was that the developer had the task of keeping the application code in synchronicity with feature classes—no easy task. The time had come for a new geographic data model with an infrastructure to tightly couple behavior with features.

## III THE GEODATABASE DATA MODEL

ArcInfo 8 introduces a new object-oriented data model called the *geodatabase data model*. The defining purpose of this new data model is to let you make the features in your GIS datasets smarter by endowing them with natural behaviors and to allow any sort of relationship to be defined among features.

The geodatabase data model brings a physical data model closer to its logical data model. The data objects in a geodatabase are mostly the same objects you would define in a logical data model, such as owners, buildings, parcels, and roads.

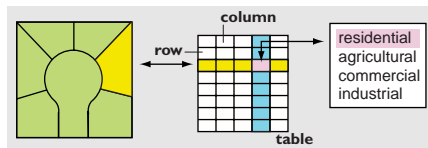
Further, the geodatabase data model lets you implement the majority of custom behavior without writing any code. Most behavior is implemented through domains, validation rules, and other functions of the framework provided in ArcInfo. Writing software code is only necessary for the more specialized behaviors of features.

## SCENARIOS OF OBJECT INTERACTIONS

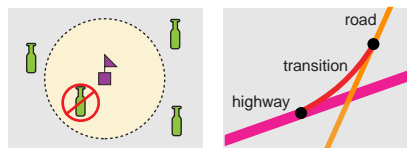
To get a sense of why an object-oriented data model is important, here are scenarios that illustrate common tasks you might perform with features. From these scenarios, we'll sift out the benefits of an object-oriented data model and then review some specific characteristics of the geodatabase data model.

## Adding and editing features

When you add geographic features to your GIS database, you want to ensure that features are placed correctly according to rules such as these:



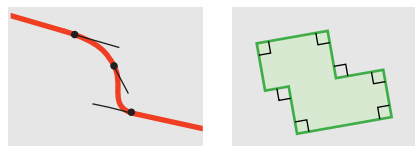
- *That the values that you assign to an attribute falls within a prescribed set of permissible values.* A parcel of land may only have certain land uses such as 'residential', 'agricultural', or 'industrial'.



- *That a feature can be placed adjacent or connected to another feature only if certain constraints are met.* Placing a liquor store near a school is not permitted by law. Or, a city road cannot be connected to an highway without a transition segment such as an on-ramp.



- *That collections of certain features conform to their natural spatial arrangement.* A stream system should always flow downhill. Flow down from a junction is the sum of flows upstream.

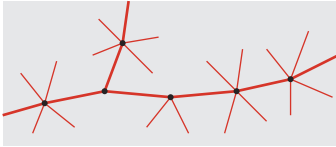


- *That the geometry of a feature follows its logical placement.* The lines and curves that make up a road should be tangent. And, building corners most often form right angles.

## Relationships among features

All objects in the world are entangled in relationships with other objects. From the perspective of a GIS, these relationships can be considered to fall within three general categories; topological, spatial, and general.

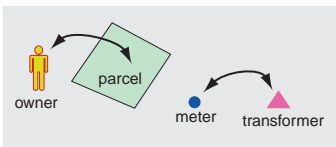
These are some examples of each of these types of relationships:



- When you edit features in an electric utility system, you want to be sure that the ends of primary and secondary lines connect exactly and that you are able to perform tracing analysis on that electric network. A set of topological relationships are defined for you when loading or editing features within a connected system.



- When you work with a map with buildings, blocks, and school districts, you might want to determine which block contains a particular building, the set of all buildings within a school district, and which blocks contain no buildings. A fundamental function of a GIS is to determine whether a feature is inside, touching, outside, or overlapping another feature. Spatial relationships are inferred from the geometry of features.

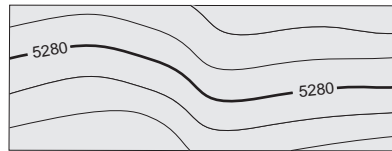


- Some objects have relationships which are not present on a map. A parcel has a relationship to an owner, but the owner is not a feature on a

map. A general relationship connects the parcel and the owner. Some features on a map have relationships, but their spatial relationship is ambiguous. A utility meter is in the general vicinity of an electric transformer, but it is not touching the transformer. The meter and the transformer might not be reliably related by their spatial proximity in crowded areas, so a general relationship ties the two features together.

## Cartographic display

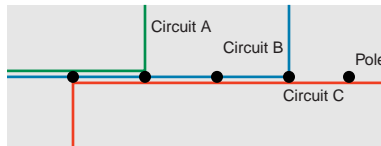
Most of the time, you will draw features on a map with pre-defined symbols, but sometimes you want more control over how your features are drawn. These are some specialized drawing behaviors:



- When you display a contour line, you want its elevation annotated along a flat section of the contour, at an average interval such as four inches, and not obscuring other features.



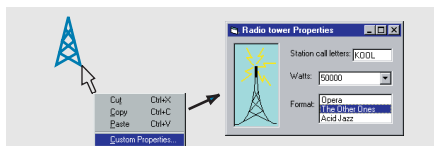
- When you draw roads on a detailed map, you would like the road drawn as parallel lines with clean intersections wherever there is a road intersection.



- When multiple electrical wires are physically mounted on the same set of utility poles, but you would like to depict them as spread in a set of parallel lines with a standard offset in map units.

## Interactive analysis

Dynamic map displays invite the user to touch features and find properties, relationships, and to launch analyses. These are examples of some tasks you may want to support upon selected features:



- Touch a feature on a map display and invoke a form to query and update its properties.



- Select a part of an electric network where line maintenance is planned, find all affected downstream customers, and make a mailing list to notify them.

## BENEFITS OF THE GEODATABASE DATA MODEL

The common thread throughout these scenarios is that it is very useful to apply object-oriented data modeling to features. Object-oriented data modeling lets you characterize features more naturally by letting you define your own types of objects, by defining topological, spatial, and general relationships, and by capturing how these objects interact with other objects. Some of the benefits of the geodatabase data model are:

- *A uniform repository of geographic data.* All of your geographic data can be stored and centrally managed in one database.
- *Data entry and editing is more accurate.* Fewer mistakes are made because most of them can be prevented by intelligent validation behavior. For many users, this alone is a compelling reason to adopt the geodatabase data model.
- *Users work with more intuitive data objects.* Properly designed, a geodatabase contains data objects that corresponds to the user's model of

data. Instead of generic points, lines, and areas, the user works with their objects of interest such as transformers, roads, and lakes.

- *Features have a richer context.* With topological associations, spatial representation, and general relationships, you not only define a feature's qualities, but its context with other features. This lets you specify what happens to features when a related feature is moved, changed, or deleted. This context also lets you locate and inspect a feature that is related to another.
- *Better maps can be made.* You have more control over how features are drawn and you can add intelligent drawing behavior. You can apply sophisticated drawing methods directly in ArcInfo's mapping application, ArcMap. Highly specialized drawing methods can be executed through writing software code.
- *Features on a map display are dynamic.* When you work with features in ArcInfo, they can respond to changes in neighboring features. You can also associate custom queries or analytic tools with features.
- *Shapes of features are better defined.* The geodatabase data model lets you use define the shapes of features using straight lines, circular curves, elliptical curves, and Bezier splines.
- *Sets of features are continuous.* By their design, geodatabases can accommodate very large sets of features without tiles or other spatial partitions.
- *Many users can edit geographic data simultaneously.* The geodatabase data model supports work flows where many people can edit features in a local area, and then reconcile any differences that emerge.

To be sure, you can realize some of these benefits without an object-oriented data model, but you would be at a disadvantage—you would need to write external code that would be loosely coupled to features and prone to complexity and error. A principal advantage of the geodatabase data model is that you have a framework that makes it as easy as possible to create intelligent features that mimic the interactions and behaviors of real world objects.

A geodatabase can contain four representations of geographic data:

- vector data for representing features,
- raster data for representing images, gridded thematic data, and surfaces,
- triangulated irregular networks (TINs) for representing surfaces, and
- locators and addresses for finding a geographic position from an address.

A geodatabase stores all of these representations of geographic data in a commercial relational database. This lets geographic data be administered centrally by information technology professionals and allows ArcInfo to leverage and keep pace with developments in database technology.

### REPRESENTING FEATURES WITH VECTORS

Many of the features in the world have well defined shapes. Vector data represents the shapes of features precisely and compactly as an ordered set of coordinates with associated attributes. This representation supports geometric operations such as calculating length and area, identifying overlaps and intersections, and finding other features which are adjacent or nearby.

Vector data can be classified by dimension:

- Points are zero-dimensional shapes represent geographic features too small to be depicted as lines or areas. Points are stored as a single x,y coordinate with attributes.
- Lines are one-dimensional shapes that represent geographic features too narrow to depict are areas. Lines are stored as a series of ordered x,y coordinates with attributes. The segments of a line can be straight, circular, elliptical, or splined.
- Areas are two-dimensional shapes that represent broad geographic features that are stored as a series of segments that enclose an area. These segments form a set of closed areas.

Another type of vector data is annotation. These are descriptive labels that are associated with features to display names and attributes.

Vector data in a geodatabase has a structure that directs the storage of features by their dimension and relationships. A feature dataset is the container of spatial entities (features), non-spatial entities (objects), and the relationships between them. Topological associations are represented with geometric networks and planar graphs.

A geodatabase also stores validation rules and domains to ensure that when features are created or updated, their attributes remain valid in the context of related features and objects.

### REPRESENTING GRIDDED DATA WITH RASTERS

Much of the data collected in a geodatabase is in gridded form. This is because cameras and imaging systems record data as pixel values in a two-dimensional grid, or raster.

A pixel is a cell element of a raster and its values can depict a variety of data. A pixel can store the reflectance of light for part of the spectrum, a color value for a photograph, a thematic attribute such as vegetative type, or surface value, or elevation.

### REPRESENTING SURFACES WITH TINs

A triangulated irregular network (TIN) is a model of a surface. A geodatabase stores TINs as an integrated set of nodes with elevations and triangles with edges through which an elevation (or z value) can be interpolated for any point within the geographic extent of a TIN.

TINs enable surface analysis such as watershed studies, visibility of a surface from an observation point, and delineation of surface features such as ridges, streams and peaks. TINs can also depict the physical relief of terrain.

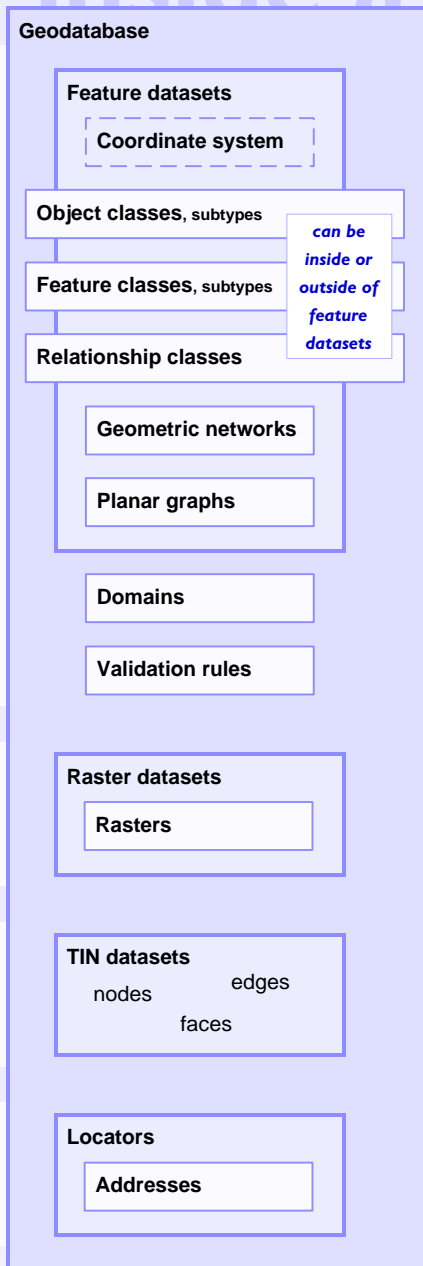
*Note: In ArcInfo 8.0, a geodatabase does not yet store TINs. For the interim, TINs can be stored in coverage workspaces.*

### FINDING ADDRESSES WITH LOCATORS

Perhaps the most common geographic task is to locate a place by an address. A geodatabase can store locators and addresses. Locators are methods that apply national postal conventions to convert an address to a position. You can interact with these points as any other point feature on the map.



# 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 non-spatial 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 graphs model systems of line and area features as a continuous coverage of an area. Planar graphs allows features to share common boundaries, such as counties sharing 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 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 which 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.



Corporate and agency databases have many records with addresses. These addresses can be located within a geodatabase. A locator is a method to convert an address to a geographic position. The found locations are displayed as features on the map.

What differentiates ArcInfo 8 from antecedent releases is that object-oriented methodology is applied to geographic data modeling. A developer interacts with data objects through a framework of object-oriented software classes called the *geodatabase data access objects*.

There are three key hallmarks of object-orientation: polymorphism, encapsulation, and inheritance.

- *Polymorphism* means that the behaviors (or methods) of an object class can adapt to variations of objects. An example of this is that the core behavior of features, such as draw, add or delete operations, is mostly the same whether the features reside in a geodatabase, coverage, or shapefile.
- *Encapsulation* means that an object is accessed only through a well-defined set of software methods, organized into software interfaces. The geodatabase data access objects mask the internal details of data objects and provide a standard programming interface.
- *Inheritance* means that an object class can be defined to include the behavior of another object class and have additional behaviors. You can create custom feature types in ArcInfo and inherit the behavior of standard features. For example, a transformer object can be extended (or sub-typed) from a standard ArcInfo feature type such as a simple junction feature.

### UNIFIED DATA MODEL

The geodatabase data access objects is a software technology that provides uniform access to geographic data from several data sources such as geodatabases, coverages, and shapefiles.

ArcInfo developers interact with geographic data through a set of data objects, such as datasets, tables, feature classes, rows, objects, and features. These objects comprise a common and consistent view of geographic data.

Because of this unified data model, the ArcInfo user can work with geodatabases, coverages, and shapefiles in the same way. The unified data model simplifies how users work with data by emphasizing the common characteristics of data.

### EXTENSIBLE FEATURES

An important aspect of a geodatabase is that you can optionally create custom features such as transformers and roads, instead of points and lines.

To the ArcInfo user, this means that a transformer or road has all of the standard display, query, and edit behavior of standard point features and line features, but with additional behaviors. You can specify that a transformer must be drawn touching a power pole and perpendicular to the electric line through the pole. Or, when a road is edited, all of its segments must be tangent.

A data modeler can use standard feature types to implement a rich data model. For advanced applications, a developer can extend the standard feature types and create custom features using the object-oriented technique of type inheritance.

Any custom feature that you create enjoys the same performance and functionality as the standard feature types provided by ArcInfo. This offers limitless opportunities for sophisticated application development.

### FEATURES AND OBJECT-ORIENTATION

Features in a geodatabase are implemented as a set of relational tables. Some of these tables represent collections of features. Other tables represent relationships between features, validation rules, and attribute domains.

ArcInfo manages the structure and integrity of these tables and presents an object-oriented geographic data through the geographic data access objects.

All users and most developers will neither know nor care about the details of the internal structure of a geodatabase. The ArcCatalog application is your user interface to establish, modify, and refine the structure of your geodatabase.

The object view of data lets you focus your efforts on building a geographic data model and hides most of the physical database structure of the geodatabase. For more information on the physical implementation of geodatabases, read the [SDE conceptual guide].

# features in a geodatabase

## unified data model

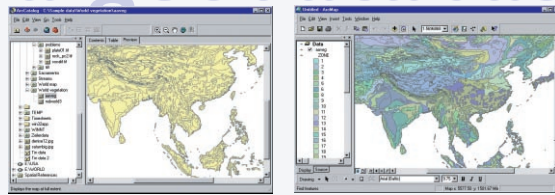
ArcInfo is versatile at displaying and analyzing geographic features. ArcInfo supports a number of data sources, among them geodatabases, coverages, and shapefiles.

The *geodatabase data access objects* are a programming interface that largely hides any differences among feature types from geodatabases, coverages, and shapefiles.

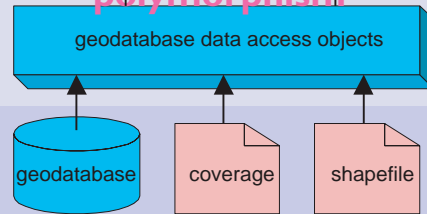
ArcInfo applications

data components

data sources



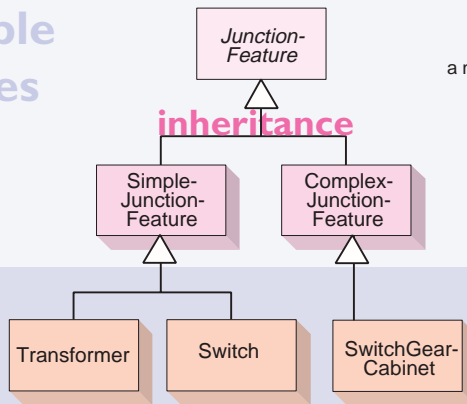
polymorphism



## extensible features

standard feature types

custom feature types



The geodatabase data access objects include a number of software components that represent the types of features that are ready for use. Shown here are some of the network feature types. These have intrinsic behavior that guarantee the topological integrity of features in a geometric network. Most data modelers use standard feature types without extending them through custom programming.

These are some custom features which have been extended from the standard feature types. They implement specialized behaviors for custom applications developed by data modelers and programmers.

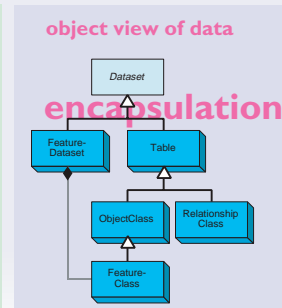
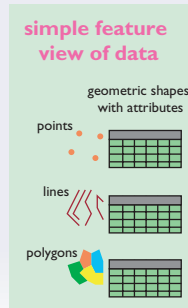
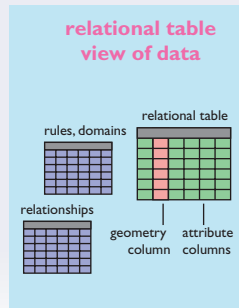
## data access

Data can be viewed in three ways.

The relational table view of data exposes the internal details of the physical storage as database tables.

The simple feature view presents data in the form of features without the structure of topology and relationships.

The object view of data encapsulates the internal details and presents a higher level of structure that is closer to the user's conceptual model of data.



ArcInfo accesses geographic data served through ArcSDE, the Arc Spatial Database Engine. ArcSDE is the software technology that enables you to create geodatabases that range from small to very large sets of geographic data and provides an open interface to the relational database of your choice.

### HOW A GEODATABASE EXTENDS A DATABASE

These are some of the facets of a geodatabase that enhance relational database technology:

- A geodatabase can represent geographic data in four manifestations: discrete objects as vector features, continuous phenomena as rasters, surfaces as TINs, and references to places as locators and addresses.
- A geodatabase stores shapes of features and ArcInfo provides functions for performing spatial operations such as finding objects that are nearby, touch, or intersect. A geodatabase has a framework for defining and managing the geographic coordinate system for a set of data.
- A geodatabase can model topologically integrated sets of features such as transportation or utility networks and subdivisions of land based on natural resources or land ownership.
- A geodatabase can define general and arbitrary relationships between objects and features.
- A geodatabase can enforce the integrity of attributes through domains and validation rules.
- A geodatabase can bind the natural behavior of features to the tables which store features.
- A geodatabase can present multiple versions so that many users can edit the same data.

### PERSONAL AND MULTI-USER GEODATABASES

Geodatabases come in two variants—personal and multi-user.

Personal geodatabase support is built into ArcInfo and is suitable for project-oriented GIS. A personal geodatabase is implemented as a Microsoft Access database. When you install ArcInfo, Microsoft Jet is also installed, which provides the services for ArcInfo to create and update Access databases. You do not need to separately install Microsoft Access.

For large enterprises, you can deploy multi-user geodatabases with ArcSDE—the multi-user data access extension to ArcInfo. ArcSDE is installed on a data server that administers your organization's relational database. Through a TCP/IP network, ArcSDE serves geodatabases to the ArcInfo applications running on personal computers. ArcSDE can be deployed on Windows NT or UNIX.

ArcSDE allows remote access to geographic data and allows many users to view and edit the same geographic data. ArcSDE is centrally tuned and managed by your database administrator.

### AN OPEN AND SCALABLE DATA SERVER

ArcInfo allows you to configure and deploy small to very large geodatabases. If you are working with moderately sized datasets, you can deploy personal geodatabases in ArcCatalog. This configuration yields good performance for datasets up to approximately 250,000 objects and supports one editor and several simultaneous viewers.

For more demanding datasets and to support many concurrent editors, you can deploy the ArcSDE extension to ArcInfo on the relational database best suited for your organization.

These are some reasons to add the ArcSDE extension to your ArcInfo installation:

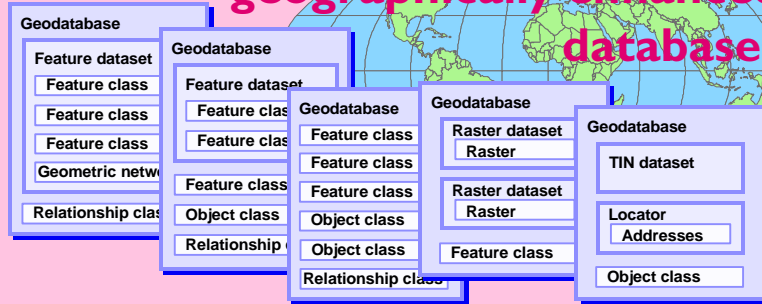
- You have limitless flexibility in scaling databases.
- You can deploy the relational database of your choice.
- You can serve geographic data from UNIX or Windows NT.
- You can serve data to other applications such as MapObjects, ArcIMS (Arc Internet Map Server), ArcView, and CAD client applications.
- You can centrally store and administer geodatabases.
- You can build Open GIS Consortium (OGC) compliant applications.
- You can build SQL applications to access the tables and rows in a geodatabase.



# open data framework

## geographically enhanced databases

A geodatabase is an instance of a relational or object-relational database that has been enhanced by adding geographic data storage, referential integrity constraints, map display, feature editing, and analysis functions.



Geodatabases on any supported relational database operate identically.

### Personal geodatabase

Personal geodatabase support is built into ArcInfo and provides access to local data.

### ArcSDE

ArcSDE is a technology that uses the native data types and operators in a relational or object-relational database and extends them to provide the complete functionality of a geodatabase.

ArcSDE is the multi-user extension to ArcInfo

## relational databases

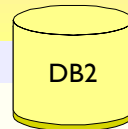


### project GIS

A personal geodatabase is directed towards personal or small work-group use. It can handle small to moderately sized datasets.

Personal geodatabases are implemented on the Microsoft Jet engine which stores data as Access databases.

A personal geodatabase has all the functionality of a geodatabase served through ArcSDE, except for versioning.



### enterprise GIS

A geodatabase served through ArcSDE can manage very large sets of geographic data and serve large numbers of viewers and editors. Geographic data is accessed from a data server on a network. This GIS data is centrally administered in large databases and integrates well with other corporate data. These databases require a system administrator for permissions, tuning, and optimization.

ArcSDE operates on any leading relational database. The ArcInfo developer can interact with a geodatabase through the geodatabase data access objects. A developer can access an ArcSDE geodatabase outside of ArcInfo through a C API (application programmer interface) or a SQL API.

To model work flow processes, a geodatabase served through ArcSDE supports long transactions and version management. A versioned geodatabase allows many editors to work concurrently and includes a framework for resolving edit conflicts.

A developer can access data in a geodatabase at three basic levels:

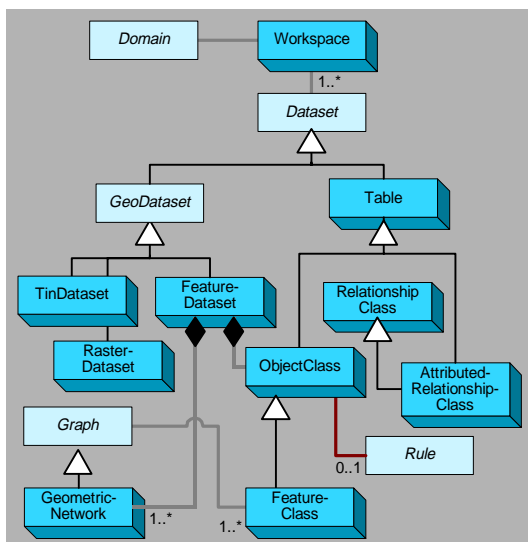
- Through the geodatabase data access objects, a subset of ArcObjects, the software components on which the desktop ArcInfo applications are built.
- As simple non-topological features through the ArcSDE API that complies with the Open GIS Consortium simple feature specification.
- As raw rows, columns, and tables through the native SQL interface of the relational database.

### ACCESSING DATA THROUGH ARCOBJECTS

The richest level of accessing geographic data is through the geodatabase data access objects. At this level, the full structure of a geodatabase is revealed; topology, relationships, integrity rules, behavior, as well as raster, surface, and location representations.

You can programmatically access data through ArcObjects using Visual Basic for Applications (VBA) inside ArcInfo or with Visual C++ or any other COM compliant development environment.

This is a simplified UML diagram of a portion of the geodatabase data access objects in ArcObjects. This model is discussed further in Chapter 4, *The structure of geographic data*.



### ACCESSING DATA AS SIMPLE FEATURES

For many spatial applications, it is sufficient and desirable to access geographic data in the form of simple non-topological features. This approach is especially suitable for building integrated applications for which geographic data is a vital component, but perhaps not the focus. Examples include facilities management and traffic analysis.

ArcSDE exposes a simple feature application programmer interface (API) in C and Java that is compliant with the Open GIS Consortium (OGC) simple features specification.

OGC is an organization that includes all of the leading spatial data vendors and has the purpose of developing standard software interfaces for the free exchange of spatial information among heterogeneous geographic information systems.

Organizations that have geographic data in various formats on a networks can build applications that integrate this data in the form of simple features.

ESRI is a leading contributor to the OGC technical specifications and is committed to the open interchange of geographic data.

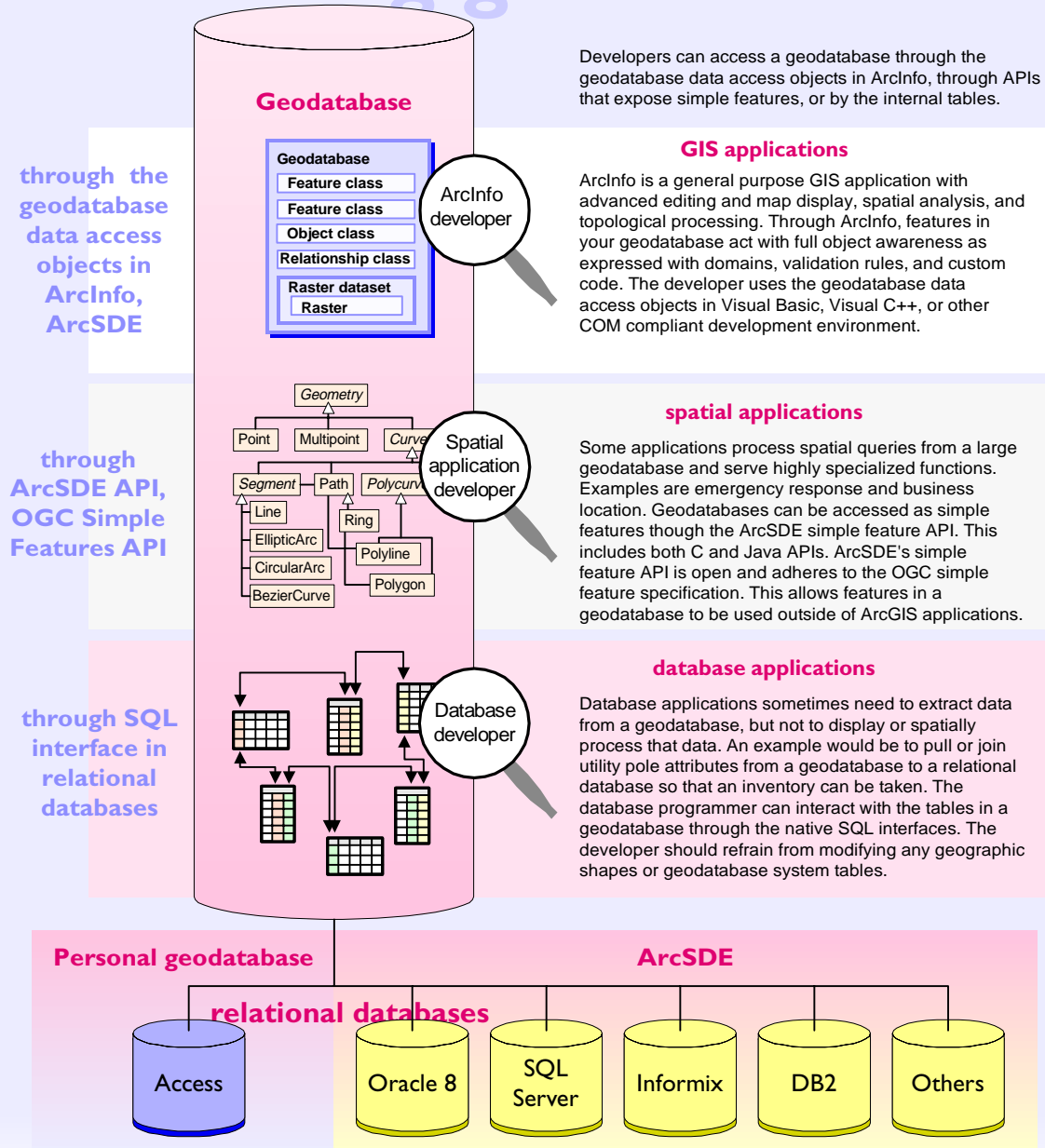
### ACCESSING DATA THROUGH SQL

A geographic information system is a rich repository of data about natural features or facilities such as transportation or utility networks. While this data is collected and managed as a geodatabase, external database applications can effectively access and share this data for non-geographic use.

Using the native SQL interfaces of your relational database, you can build applications to mine data from your geodatabases and use them for tasks such as managing inventory, processing work orders, or statistical analysis.

In this view, a geodatabase is a set of tables, rows, and columns. Through the SQL interfaces, you can see the internal database structure of a geodatabase, which includes metadata tables for objects such as networks. This structure is not directly visible in ArcInfo and is managed through the user interface of ArcCatalog. You can selectively update attributes of rows that represent features, but you should take care not to corrupt the structure of the geodatabase.

# accessing geodatabases



Designing a geodatabase is fundamentally the same as designing any database. That's because a geodatabase is an instance of a relational database—albeit one that contains a structure for representing geographic data.

The geodatabase extends, yet simplifies, the design process by presenting an object-oriented data structure that expresses the spatial and topological relationships of geographic features. Part of this structure is a special facility for representing sets of objects as integrated systems—such as stream and road networks or sets of land parcels. This structure on a set of features is called topology.

The geodatabase data model is the bridge between people's cognitive perception of the objects surrounding them in the world and how those objects are stored in relational databases.

### GEODATABASE DESIGN

Traditional relational database design spans two basic steps—the articulation of a logical data model and the physical implementation of database models (or schemas).

The logical data model captures the user's view of data and the database model implements the data model within the framework of relational database technology.

#### Designing a logical data model

The key task in building a logical data model is to precisely define the set of objects of interest and to identify the relationships between them.

Some examples of objects you might consider are streets, parcels, owners, and buildings. Some examples of their relationships are 'located at', 'owned by', and 'is part of'.

Once you have an initial logical data model, you can validate it against the user's requirements for entering, updating, and accessing data and by testing it against the organization's practices and procedures (or, business rules).

It is especially important to involve representatives from each prospective user group. A logical data model built for a subset of users is guaranteed to

have deficiencies for overlooked users.

Building a logical data model is an iterative process and is an art that is acquired through experience. There is no single 'correct' model, but there are good models and bad models. It's difficult to determine precisely when a model is correct and complete, but an indication that you are drawing close is when you can answer these questions in the affirmative:

- Does the logical data model represent all data without duplication?
- Does the logical data model support an organization's business rules?
- Does the logical data model accommodate different views of data for distinct groups of users?

### Representing logical data models

In the past, logical data models were often drawn in what are known as 'entity-relationship diagrams'. A number of leading object-oriented modelers put forward various design methodologies and diagram notations.

These methodologies emphasized different aspects such as data flow or use-case scenarios, but a problem with entity-relationship diagrams is that their appearance varied with the design methodology.

More recently, most object-oriented modelers have adopted the Unified Modeling Language (UML), which is a standard notation for expressing object models and is endorsed by leading software and database companies.

It is important to note that UML is not a design methodology, but rather a diagrammatic notation. With UML, you can adopt the object-oriented design methodology of your choosing and then express the model in a standard way with UML.

This book uses UML for drawing ArcInfo's object model, called ArcObjects, and for drawing the custom object models you can create in a geodatabase.



### Implementing a physical database model

A physical database model is built from the logical data model. Typically, a specialist in relational databases receives the logical data model from the data modeler and uses the database administration tools to define the database schema and create new databases ready for data transfer and entry.

The physical database design has some similarity to the logical data model, but there are differences. Classes of objects may be split or joined when implemented in tables. Rules and relationships can be expressed in several ways.

An important benefit of the geodatabase is that it is a physical implementation of data, but it lets you structure your data in a fashion that is close to the logical data model.

### Elements of the logical and database models

These are the basic elements of the logical data model and their corresponding database elements.

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

A logical data model is an abstraction of the objects that we encounter in a particular application. This abstraction is converted into database elements.

An object represents an entity such as a house, lake, or customer. An object is stored as a row.

An object has a set of attributes. Attributes characterize qualities of an object, such as its name, a measure, a classification, or an identifier (or key) to another object. Attributes are stored in a database in columns (or fields).

A class is a set of similar objects. Each object in a class has the same set of attributes. A class is stored in a database as a table. The rows and columns in a table form a two-dimensional matrix.

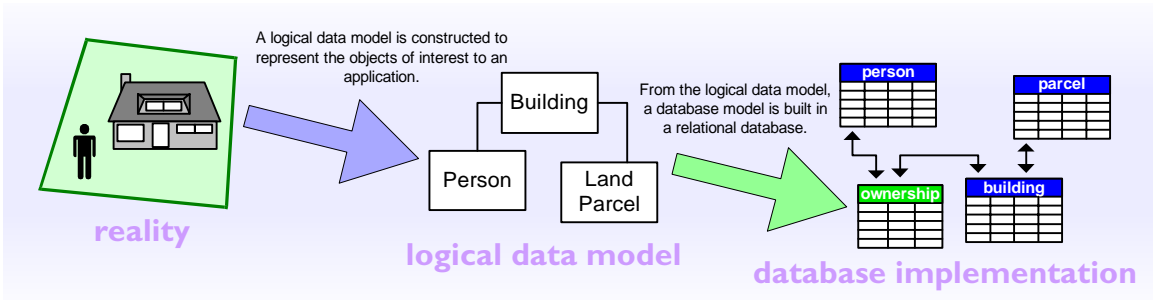
### Handling complex data

Relational databases enjoy their commercial dominance because they implement a simple, elegant and well understood theory. This simplicity is at once a strength and a weakness—it is conceptually straight-forward to build relational databases, but difficult to model complex data.

Geographic databases contain complex data. The shapes of line and area features are structured sets of coordinates that cannot be well represented with standard atomic field types such as integer, real, string. Further, features are gathered into systems that have explicit topological relationships, implicit spatial relationships, or general relationships.

The relational database is the foundation for a geodatabase. A key purpose of the geodatabase is to handle the complexity of geographic data with a uniform data model independent of the relational database underneath.

Chapter 12, *Geodatabase design guide*, returns to these topics in the context of designing and building geodatabases.



## GUIDELINES FOR GEODATABASE DESIGN

The structure of a geodatabase—feature datasets, feature classes, topological groupings, relationships, and other elements—lets you design geographic databases that are close to their logical data models. For a data modeler, this is the essential reason for the introduction of geodatabases into ArcInfo 8.

These are the basic steps to design a geodatabase:

- *Model the user's view of data.* Perform interviews with users, understand an organization's structure, and analyze the business requirements.
- *Define objects and relationships.* Build the logical data model with the set of objects and how they are related to one another.

- *Select geographic representation.* Determine whether vector, raster, surface, or location representation is best for the data of interest.
- *Match to geodatabase elements.* Fit the objects in the logical data model into the elements of a geodatabase.
- *Organize geodatabase structure.* Build the structure of a geodatabase with consideration of thematic groupings, topological associations, and department responsibility of data.

This topic is discussed in greater detail in Chapter 12, *Geodatabase design guide*.

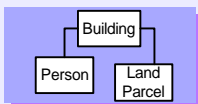
# steps to building a geodatabase

1  
*model the user's view of data*



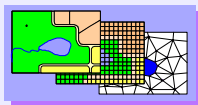
identify organizational functions  
determine data needed to support functions  
organize data into logical groupings

2  
*define objects and relationships*



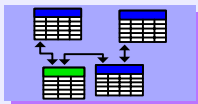
identify and describe objects  
specify relationships between objects  
document model in diagram

3  
*select geographic representation*



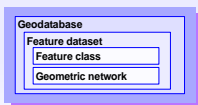
represent discrete features with points, lines, areas  
characterize continuous phenomena with rasters  
model surfaces with TINs or rasters

4  
*match to geodatabase elements*



determine geometry type of discrete features  
specify relationships between features  
implement attribute types for objects

5  
*organize geodatabase structure*



organize systems of features  
define topological associations  
assign coordinate systems  
define relationships and rules

You can approach ArcInfo in two ways; as a user of applications such as ArcMap and ArcCatalog or as a software developer building custom applications.

Data modelers straddle these two worlds—you use the applications for most of your work in creating geodatabases, but you will sometimes write software code, especially if you are trying to create rich data models that support powerful applications.

One aim of this book is to present the important data modeling concepts both as they are applied in the ArcInfo applications and the ArcInfo software components, which are called ArcObjects.

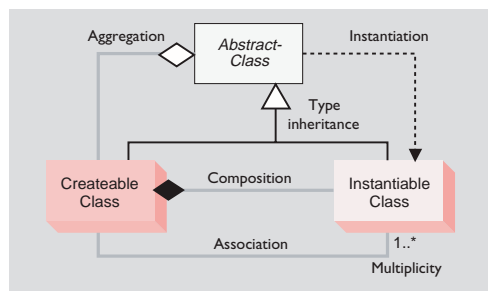
A pattern through this book is to first present the concepts for a topic as you experience it through the ArcInfo application. Next, that topic is summarized with an annotated diagram of the relevant section of the ArcInfo object model diagram.

An example of this presentation is that the topic of the structure of geodatabases, feature datasets, and feature classes is first discussed with the user's perspective of the catalog. Next, the programmer's perspective is summarized with a diagram of part of the geodatabase data components.

These two views have similarities, but subtle differences. A user interface sometimes hides details about software components that are important to the programmer. One goal of this book is to give you the insight to bridge the user and developer perspectives.

## Reading the class diagrams

This is the key for the object model diagrams you will find throughout this book.



This notation is based on the UML (Unified Modeling Language) notation, an industry diagramming standard for object-oriented analysis and design.

The object model diagrams are an important supplement to the information you receive in object browsers. The development environment, Visual Basic or other, lists all of the many classes and members, but does not hint at the structure of those classes. These diagrams complete your understanding of the ArcInfo components.

This book uses UML to document the ArcInfo software components, ArcObjects, and to illustrate custom data models that you can build.

## Classes and objects

There are three types of classes shown in the UML diagrams—abstract classes, createable classes, and instantiable classes.

An *abstract class* cannot be used to create new objects, but it is a specification for subclasses. An example is that a 'line' could be an abstract class for 'primary line' and 'secondary line' classes.

A *createable class* represents objects that you can directly create using the object declaration syntax in your development environment. In Visual Basic, this is written with the *Dim As New <object>* or *CreateObject(<object>)* syntax.

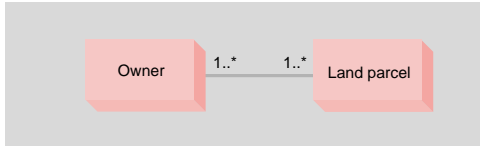
An *instantiable class* cannot directly create new objects, but objects of this class can be created as a property of another class or created by functions from another class.

In the Visual Basic object browser, you can inspect all of the ArcInfo createable and instantiable classes, but not the abstract classes.

## Relationships

Among abstract classes, createable classes, and instantiable classes, there are several types of class relationships possible.

*Associations* represent relationships between classes. They have defined multiplicities at both ends.



In this diagram, an owner can own one or many land parcels and a land parcel can be owned by one or many owners.

A *Multiplicity* is a constraint on the number of objects that can be associated with another object. This is the notation for multiplicities:

1 - One and only one. Showing this multiplicity is optional; if none is shown, '1' is implied.

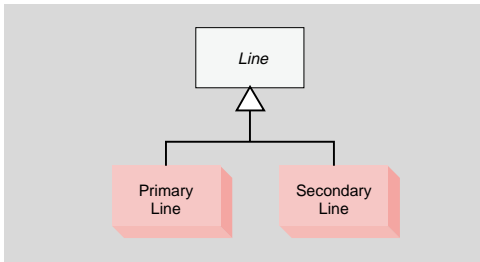
0..1 - Zero or one

M..N - From M to N (positive integers)

\* or 0..\* - From zero to any positive integer

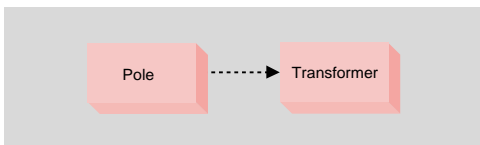
1..\* - From one to any positive integer

*Type inheritance* defines specialized classes which share properties and methods with the superclass and have additional properties and methods.



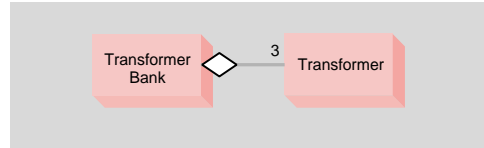
This diagram shows that a primary line (createable class) and secondary line (createable class) are types of a line (abstract class).

*Instantiation* specifies that one object from one class has a method with which it creates an object from another class.



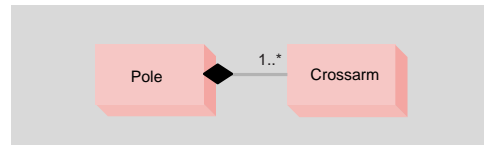
A pole object might have a method to create a transformer object..

*Aggregation* is an asymmetric association in which an object from one class is considered to be a 'whole' and objects from the other class are considered 'parts'.



A transformer bank has exactly three transformers. In this design, transformers can be associated with a transformer bank, but may also exist after the transformer bank is removed.

*Composition* is a stronger form of aggregation in which objects from the 'whole' class control the lifetime of objects from the 'part' class.



A pole contains one or many crossarms. In this design, a crossarm cannot be recycled when the pole is removed. The pole object controls the lifetime of the crossarm object.

## Expressing models with diagram notation

If you are unaccustomed to this type of diagram notation, practice reading the examples above and conceive of your own examples. Before long, you'll read these diagrams with ease.

You'll find that it is worth your effort to understand this notation. It describes object models in a concise and expressive way and will facilitate your conceptual understanding of the ArcInfo software components.

Understanding this notation is also critical if you create custom features by extending the geodatabase data access objects. With ArcCatalog, you can launch a CASE environment to create custom data models with a visual user interface. This interface is based on manipulating graphical symbols from the UML notation.



A geographic information system is at its core a database management system enhanced to store, index, and display geographic data.

ArcInfo 8 is a significant release of new GIS technology that exploits several important technology trends just as they have become ready for commercial implementation. These trends collectively realize the vision of GIS as a geographically-enabled database.

The timing of ArcInfo 8 is fortuitous as it occurs during the convergence of several critical developments in software and database technology. These are the principal trends that shape the technological framework of ArcInfo 8.

### **Spatial data and databases**

When the coverage data model was first implemented, practical considerations led to the spatial component of geographic data being contained in binary files with unique identifiers to rows in relational database tables that stored feature attributes.

With performance and functional advances in database technology, it is now possible and advantageous to store all spatial data directly within the same database tables as attribute data.

The gain from storing spatial data directly within commercial databases is improved data administration, the utilization of data access and management services, and closer integration with the other databases that an organization manages.

Moreover, the ArcInfo user can select from any of the industry-leading relational databases to host their geographic databases.

### **User interface**

Applications developed for Microsoft Windows have set a new standard for ease-of-use and consistency. Users have become accustomed to expected behaviors for mouse interaction, menus, dialogs, and the like. These user interface standards have made powerful applications accessible and usable by people who are not computer experts.

ArcInfo 8 thoroughly implements the Windows

standards for user interface and stands as a new milestone in making GIS software easier to use.

### **Software component architecture**

Modern software is built on software component architectures, examples of which are Microsoft COM (Component Object Model), CORBA (Common Object Request Broker Architecture), and Java RMI (Remote Method Invocation).

The idea behind components is to divide software functionality into discrete, independent pieces that can be developed, tested, and combined into programs. By their design, components can be used to build any number of applications without modification. This is a high level of software re-use.

The benefit of software component architectures is better software quality, better performance, and the ability to update software versions without affecting other installed software.

ArcInfo 8 is built on the Microsoft COM architecture because it is the most robust and reliable component framework for desktop applications.

### **Programming environment**

Visual programming environments such as Visual Basic have become the norm for application development.

The benefits of using these languages are the large pool of experienced programmers and the richness of these environments. It is no longer necessary or desirable to use proprietary macro languages.

ArcInfo 8 uses Visual Basic for Applications as its embedded macro language for customizing its applications, ArcMap and ArcCatalog. Other COM compliant languages such as Visual C++ can be used to extend the geodatabase data model.

### **Trends in summary**

The common theme of these technology trends is open standards and interoperability.

The benefit of implementing these trends is to leverage technology from other segments which enables ESRI to concentrate its research and development on core GIS functionality.