7 Geographic Data Modeling

LEARNING OBJECTIVES

After studying this chapter you will be able to:

- Define what geographic data models are and discuss their importance.
- Understand how to undertake GI data modeling.
- Outline the main geographic models used in GI databases and their strengths and weaknesses.
- Understand key topology concepts and why topology is useful for data validation, analysis, and editing.
- Read data model notation.
- Describe how to model the world and create a useful geographic database.

This chapter discusses the technical issues involved in modeling the real world in a geographic information (GI) database. It describes the process of data modeling and the various data models that have been used. A data model is a set of constructs for describing and representing parts of the real world in a digital computer system. Data models are vitally important because they control the way that data are stored and have a major impact on the type of analytical operations that can be performed. Early GI databases were based on extended CAD, simple graphical, and image data models. In the 1980s and 1990s, the hybrid georelational model came to dominate. More recently, major software systems have been developed on more advanced and standards-based geographic object models that include elements of all earlier models.

7.1 Introduction

This chapter builds on the material on geographic representation presented in Chapter 3. By way of introduction, it should be noted that the terms *representation* and *model* overlap considerably (we will return to a more detailed discussion of models in Chapter 15). *Representation* is typically used in conceptual and scientific discussions, whereas *model* is used in practical and database circles and is preferred in this chapter. The focus here is on how geographic reality is modeled (that is to say, abstracted or simplified) in GI databases, with particular emphasis on the different types of data models that have been developed. A data model is an essential ingredient of any operational GI system and, as the discussion will show, has important implications for the types of operations that can be performed and the results that can be obtained.

7.1.1 Data Model Overview

The heart of any GI database is the data model, which is a set of constructs for representing objects and processes in the digital environment of the computer (Figure 7.1). People (GI system users) interact with operational GI systems in order to undertake tasks such as making maps, querying databases, and performing site suitability analyses. Because the types of analyses that can be undertaken are strongly influenced by the way the real world is modeled, decisions about the type of data model to be adopted are vital to the success of a GI project.

A data model is a set of constructs for describing and representing selected aspects of the real world in a computer.

As described in Chapters 2 and 3, geographic reality is continuous and of seemingly infinite complexity, but computers are finite, are comparatively

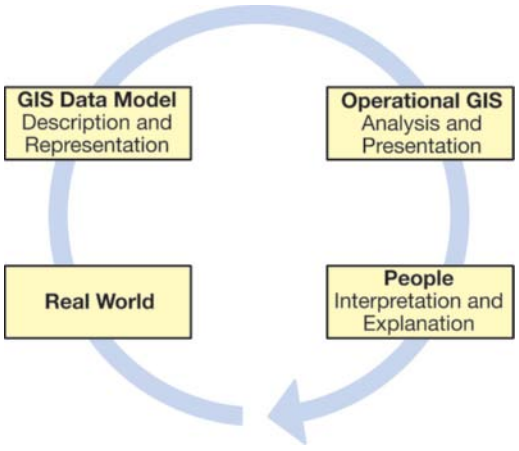


Figure 7.1 The role of a data model in GI systems.

simple, and can only work with digital data. Therefore, difficult choices have to be made about what things are modeled in a GI database and how they are represented. Because different types of people use GI systems for different purposes, and the phenomena these people study have different characteristics, there is no single type of all-encompassing data model that is best for all circumstances. Modern GI systems are able to incorporate multiple data models and so can be applied to a wide range of different application areas.

7.1.2 Levels of Data Model Abstraction

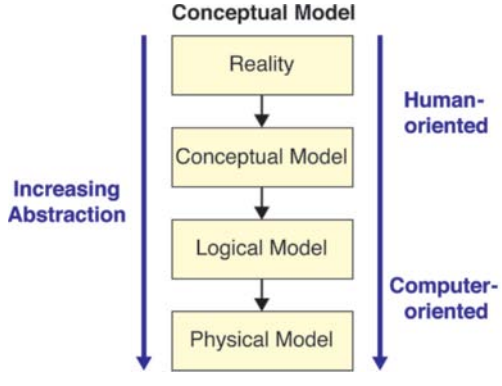
When representing the real world in a computer, it is helpful to think in terms of four different levels of abstraction (levels of generalization or simplification); these are shown in Figure 7.2. First, *reality* is made up of real-world phenomena (buildings, streets, wells, lakes, people, etc.) and includes all aspects that may or may not be perceived by individuals or deemed relevant to a particular application. Second, the *conceptual model* is a human-oriented, often partially structured, model of selected objects and processes that are thought relevant to a particular problem domain. Third, the *logical model* is an implementation-oriented representation of reality that is often expressed in the form of diagrams and lists. Last, the *physical model* portrays the actual implementation in a GI system and often comprises tables stored as files or databases (see Chapter 9). Use of the term *physical* here is actually misleading because the models are not physical and only exist digitally in computers, but this is the generally accepted use of the term. Relating back to the discussion of uncertainty in Chapter 5 (Figure 5.1), the conceptual and logical models are found beyond the U1 filter, and the physical model is that on which analysis may be performed (beyond the U2 filter).

In data modeling, users and system developers participate in a process that successively engages with each of these levels. The first phase of modeling begins with the definition of the main types of objects to be represented in the GI database and concludes with a conceptual description of the main types of objects and relationships between them. Once this phase is complete, further work will lead to the creation of diagrams and lists describing the names of objects and what they look like, their behavior (how they act), and the type of interaction between objects. This type of logical data model is very valuable for defining what a GI system will be able to do and the type of domain over which it will extend. Logical models are implementation-independent and in theory can be created in any GI system with appropriate capabilities. The final data modeling phase involves creating a model showing how the objects under study can be digitally implemented in a GI database. Physical models describe the exact files or database tables used to store data, the relationships between object types, and the precise operations that can be performed. For more details about the practical steps involved in data modeling, see Sections 7.3 and 7.4.

A data model provides system developers and users with a common understanding and reference point. For developers, a data model is the means to represent an application domain in terms that may be translated into a design and then implemented in a system. For users, it provides a description of the structure of the system, independent of specific items of data or details of the particular application. A data model controls the types of things that a GI system can handle and the range of operations that can be performed on them.

The discussion of geographic representation in Chapter 3 introduced discrete objects and fields, the two fundamental conceptual models for representing real-world things geographically. In the same chapter the raster and vector logical models were also

Figure 7.2 Levels of GI data model abstraction.



Source: Esri

(A) (B)

Figure 7.3 Different representational models of the same area in Colorado, USA: (A) aerial photograph; and (B) vector objects, some digitized from the photograph.

introduced. Figure 7.3 shows two representations of the same area in a GI database, one raster and the other vector. Notice the difference in the objects represented. The roads and buildings can be clearly seen on both figures, but the vector representation (Figure 7.3B) also shows the lot (property) boundaries and water mains and valves. Cars and other surface texture ephemera can be observed on the raster representation (Figure 7.3A). The next sections in this chapter focus on the logical and physical representation of raster, vector, and related models in GI systems.

7.2 GI Data Models

In the past half-century, many GI data models have been developed and deployed in GI systems. The key types of geographic data models and their main areas of application are listed in Table 7.1. All are based in some way on the conceptual discrete object/field and logical vector/raster geographic data models. All GI systems include a core data model that is built on one or more of these GI data models. In practice, any modern comprehensive GI system supports at least some elements of all these models. As discussed earlier, the GI core system data model is the means to represent geographic aspects of the real world and defines the type of geographic operations that can be performed. It is the responsibility of the implementation team to populate this generic model with information about a particular problem (e.g., utility outage management, military mapping, or natural resource planning). Some GI software packages come with a fixed data model, whereas others have models that can be extended by adding new object types and relationships. Those that can easily be extended are better able to model the richness of geographic domains and in general are the easiest to use and the most productive systems.

Table 7.1 Some commonly used geographic data models.

Data model	Example application
Computer-aided design (CAD)	Automating engineering design and drafting
Graphical (nontopological)	Simple mapping and graphic arts
Image	Image processing and simple grid analysis
Raster/grid	Spatial analysis and modeling, especially in environmental and natural resources applications
Vector/georelational topological	Many operations on geometric features in cartography, socioeconomic, and resource analysis and modeling
Network	Network analysis in transportation and utilities
Triangulated irregular network (TIN)	Surface/terrain visualization, analysis and modeling
Object	Many operations on all types of entities (raster/vector/TIN, etc.) in all types of applications

When modeling the real world for representation inside a GI database, it is convenient to group entities of the same geometric type together (for example, all point entities such as lights, garbage cans, or dumpsters might be stored together). A collection of entities of the same geometric type (dimensionality) is referred to as a class or layer. It should also be noted that the term *layer* is quite widely used in GI systems as a general term for a specific dataset. It is derived from the process of entering different types of data into a GI system from paper maps, which was undertaken one plate at a time. (All entities of the same type were represented in the same color and, using printing technology, were reproduced together on film or printing plates.) Grouping entities of the same geographic type together makes the storage of geographic databases more efficient (for further discussion, see Section 9.3). It also makes it much easier to implement rules for validating edit operations (for example, the addition of a new building or census administrative area) and for building relationships between entities. All the data models discussed in this chapter use layers in some way to handle geographic entities.

A layer is a collection of geographic entities of the same geometric type (e.g., points, lines, or areas). Grouped layers may combine layers of different geometric types.

7.2.1 CAD, Graphical, and Image Data Models

The earliest GI systems were based on very simple data models derived from work in the fields of CAD (computer-aided design and drafting), computer cartography, and image analysis. In a CAD system, real-world entities are represented symbolically as simple point, line, and area vectors. This basic CAD data model never became widely popular for GI because of three severe problems for most applications at geographic scales. First, because CAD models typically use local drawing coordinates instead of real-world coordinates for representing objects, they are of little use for map-centric applications. Second, because individual objects do not have unique identifiers, it is difficult to tag them with attributes. As the following discussion shows, this is a key requirement for GI applications. Third, because CAD data models focus on graphical representation of objects, they cannot store details of any relationships between objects (e.g., topology or networks), yet this type of information is essential in many spatial analytical operations.

A second type of simple GI geometry model was derived from work in the field of computer cartography. The main requirement for this field in the 1960s was the automated reproduction of paper topographic maps and the creation of simple thematic maps. Techniques were developed to digitize maps and store them in a computer for subsequent plotting and printing. All paper map entities were stored as points, lines, and areas, with annotation used for place-names. Like CAD systems, there was no requirement to tag objects with attributes or to work with object relationships.

At about the same time that CAD and computer cartography systems were being developed, a third type of data model emerged in the field of image processing. Because the main data sources for geographic image processing are scanned aerial photographs and digital satellite images, it was natural that these systems would use rasters or grids to represent the patterning of real-world objects on the Earth's surface. The image data model is also well suited to working with pictures of real-world objects, such as photographs of water valves and scanned building floor plans that are held as attributes of geographically referenced entities in a database (Figure 7.4).

Despite their many limitations, some of which have been overcome through data model extensions, GI systems still exist based on these simple data models, although the number is in significant decline. This is partly for historical reasons—the GI system may have been built before newer, more advanced models became available—but also because of lack of knowledge about the newer approaches described in this chapter.

7.2.2 Raster Data Model

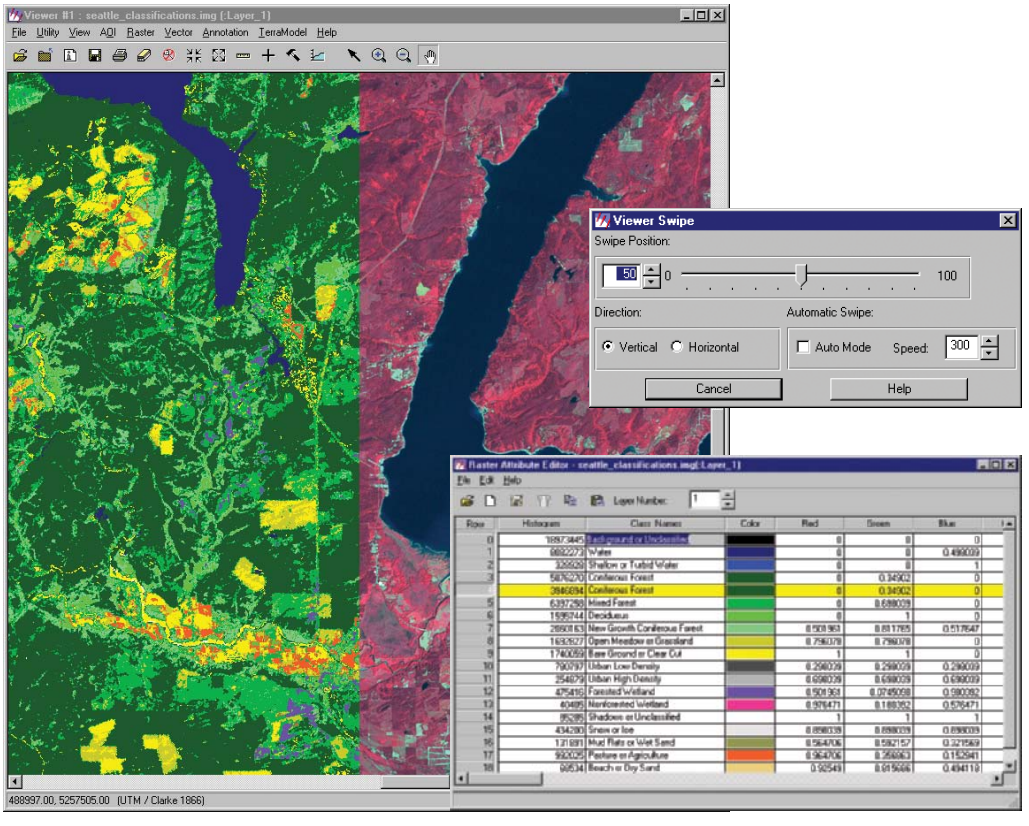
The raster data model uses an array of cells, or pixels, to represent real-world objects (see Figure 3.10). The cells can hold attribute values based on one of several encoding schemes, including categories, and integer and floating-point numbers (see Box 3.2 for details). In the simplest case a binary representation is used (for example, presence or absence of vegetation), but in more advanced cases floating-point values are preferred (for example, height of terrain above sea level in meters). In some systems, multiple attributes can be stored for each cell in a type of value attribute table where each column is an attribute and each row either a pixel or a pixel class (Figure 7.5).

Raster data are usually stored as an array of grid values, with metadata (data about data: see Section 10.2) about the array held in a file header. Typical metadata include the geographic coordinate of the upper-left corner of the grid, the cell size, the number of row and column elements, and the projection. The data array itself is usually stored as a compressed file or record in a

Source: Esri

Figure 7.4 A photographic image used as a building object attribute in an electric facility system.

Figure 7.5 Raster data of the Olympic Peninsula, Washington State, with associated value attribute table. Bands 4,3,2 from Landsat 5 satellite with land cover classification overlaid.



Screenshot courtesy: ERDAS Inc.; data courtesy: USGS

database management system (see Section 9.3). Techniques for compressing rasters are described in Box 7.1.

Datasets encoded using the raster data model are particularly useful as a backdrop map display because they look like conventional maps and can communicate a lot of information quickly. They are also widely used for analytical applications such as disease dispersion modeling, surface-water flow analysis, and store location modeling.

7.2.3 Vector Data Model

The raster data model discussed earlier is most commonly associated with the field conceptual data

Technical Box 7.1

Technical Raster Compression Techniques

Although the raster data model has many uses in GIS, one of the main operational problems associated with it is the sheer amount of raw data that must be stored. To improve storage efficiency, many types of raster compression techniques have been developed such as run-length encoding, block encoding, wavelet compression, and quadtrees (see Section 9.7.2.2 for another use of quadtrees as a means of indexing geographic data). Table 7.2 presents a comparison of file sizes and compression rates for three compression techniques based on the image in Figure 7.6. It can be seen that even the

Table 7.2 Comparison of file sizes and compression rates for selected raster compression techniques (using image shown in Figure 7.6).

Compression technique	File size (MB)	Compression rate
Uncompressed original	80.5	—
Run-length	17.7	5.1
Wavelet	2.3	38.3

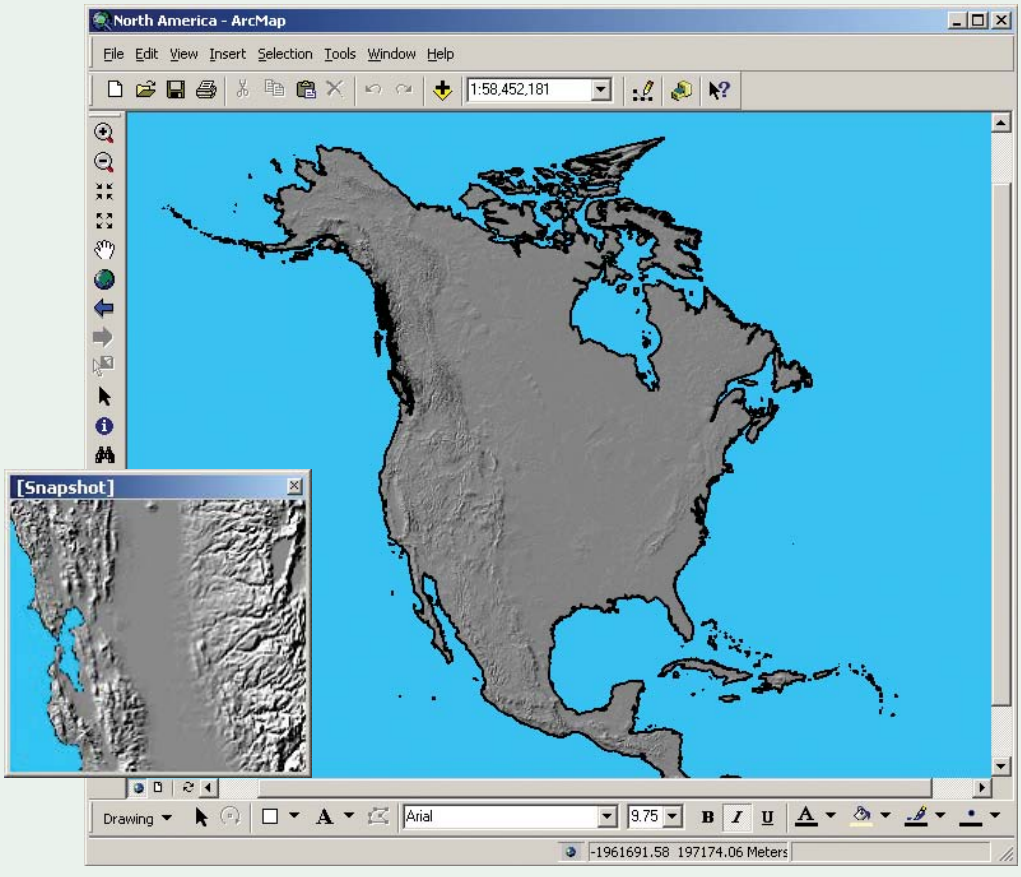


Figure 7.6 Shaded digital elevation model of North America used for comparison of image compression techniques in Table 7.2. Original image is 8,726 by 10,618 pixels, 8 bits per pixel. The inset shows part of the image at a zoom factor of 1,000 for the San Francisco Bay area.

comparatively simple run-length encoding technique compresses the file size by a factor of 5. The more sophisticated wavelet compression technique results in a compression rate of almost 40, reducing the file from 80.5 to 2.3 MB.

Run-Length Encoding
Run-length encoding is perhaps the simplest compression method and is very widely used. It involves encoding adjacent row cells that have the same value, with a pair of values indicating the number of cells with the same value and the actual value.

Block Encoding
Block encoding is a two-dimensional version of run-length encoding in which areas of common cell values are represented with a single value. An array is defined as a series of square blocks of the largest size possible. Recursively, the array is divided using blocks of smaller and smaller sizes. It is sometimes described as a quadtree data structure (see also Section 9.7.2.2).

Wavelet
Wavelet compression techniques invoke principles similar to those discussed in the treatment of fractals (see Section 2.8). They remove information by recursively examining patterns in datasets at different scales, always trying to reproduce a faithful representation of the original. A useful by-product of this for geographic applications is that wavelet-compressed raster layers can be quickly viewed at different scales with appropriate amounts of detail. The JPEG 2000 standard defines wavelet image compression.

Run-length and block encoding both result in lossless compression of raster layers; that is, a layer can be compressed and decompressed without degradation of information. In contrast, the wavelet compression technique is *lossy* because information is irrevocably discarded during compression. Although wavelet compression results in very high compression ratios, because information is lost its use is limited to applications that do not need to use the raw digital numbers for processing or analysis. It is not appropriate for compressing DEMs, for example, but many organizations use it to compress scanned maps and aerial photographs when access to the original data is not necessary.

model. The vector data model, on the other hand, is closely linked with the discrete object view. Both of these conceptual perspectives were introduced in Section 3.5. The vector data model is used in GI databases because of the precise nature of its representation method, its storage efficiency, the quality of its cartographic output, and the wide availability of functional tools for operations like map projection, overlay processing, and cartographic analysis.

In the vector data model, each object in the real world is first classified into a geometric type; in the 2-D case it is point, line, or area (Figure 7.7). Points (e.g., wells, soil pits, and retail stores) are recoded as single coordinate pairs, lines (e.g., roads, streams, and geologic faults) as a series of ordered coordinate pairs (also called polylines—Section 3.6.2), and areas (e.g., census tracts, soil areas, and oil license zones) as one or more line segments that close to form a polygonal area. The coordinates that define the geometry of each object may have 2, 3, or 4 dimensions: 2 (x, y: row and column, or latitude and longitude), 3 (x, y, z: the addition of a height value), or 4 (x, y, z, m: the addition of another value to represent time or some other property—perhaps the offset of road signs from a road centerline, or an attribute).

For completeness, it should also be said that in some data models linear features can be represented not only as a series of ordered coordinates, but also as curves defined by a mathematical function (e.g., a spline or Bézier curve). These are particularly useful for representing built-environment entities like road curbs and some buildings.

7.2.3.1 Simple Features

Geographic entities encoded using the vector data model are usually called features, and this will be the convention adopted here. Features of the same geometric type are stored in a GI database as a feature class, or when speaking about the physical (database) representation the term *feature table* is preferred. Here each feature occupies a row, and each property of the feature occupies a column. GI databases commonly deal with two types of feature: simple and topological. The structure of simple feature polyline and polygon datasets is sometimes called *spaghetti* because, like a plate of cooked spaghetti, lines (strands of spaghetti) and polygons (spaghetti hoops) can overlap, and there are no stored relationships between any of the objects.

Features are vector objects of type point, polyline, or polygon.

Simple feature datasets are useful in GI applications because they are easy to create and store, and because they can be retrieved and rendered on

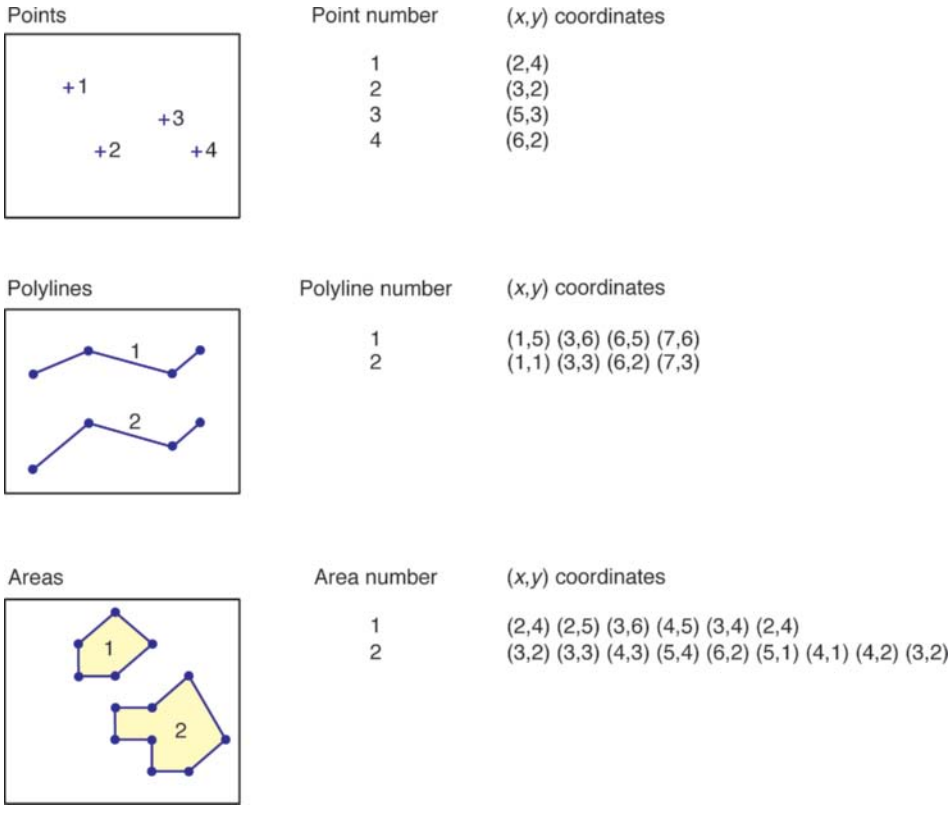


Figure 7.7 Representation of point, line, and area objects using the vector data model.

screen very quickly. However, because simple features lack more advanced data structure characteristics, such as topology (see next section), operations such as shortest-path network analysis and querying polygon adjacency cannot be performed without additional calculations.

7.2.3.2 Topological Features

Topological features are essentially simple features structured using topological rules. Topology is the mathematics and science of geometrical relationships. Topological relationships are nonmetric (qualitative) properties of geographic objects that remain constant when the geographic space of objects is distorted. For example, when a map is stretched, properties such as distance and angle change, whereas topological properties such as adjacency and containment do not change. Topological structuring of vector layers introduces some interesting and very useful properties, especially for polyline (also called 1-cell, arc, edge, line, and link) and area (also called 2-cell, polygon, and face) data. Topological structuring of line layers forces all line ends that are within a user-defined distance to be snapped together so that they are given exactly the same coordinate value. A node is placed wherever the ends of lines meet or cross. Following on from the earlier analogy, this type of data model is sometimes referred to as spaghetti with meatballs (the nodes being the meatballs on the spaghetti lines). Topology is important in GI systems because of its role in data validation, modeling integrated feature behavior, editing, and query optimization.

> **Topology is the science and mathematics of relationships used to validate the geometry of vector entities and for operations such as network tracing and tests of polygon adjacency.**

Data validation Many of the geographic data collected from basic digitizing, field data collection devices, photogrammetric workstations, and CAD systems comprise simple features of type point, polyline, or polygon, with limited structural intelligence (for example, no topology). Testing the topological integrity of a dataset is a useful way to validate the geometric quality of the data and to assess its suitability for geographic analysis. Some useful data validation topology tests include the following:

- Network connectivity—do all network elements connect to form a graph (i.e., are all the pipes in a water network joined to form a wastewater system)? Network elements that connect must be "snapped" together (that is, given the same coordinate value) at junctions (intersections).

- Line intersection—are there junctions at intersecting polylines, but not at crossing polylines? It is, for example, perfectly possible for roads to cross in planimetric 2-D view, but not intersect in 3-D (for example, at a bridge or underpass).
- Overlap—do adjacent polygons overlap? In many applications (e.g., land ownership), it is important to build databases free from overlaps and gaps so that ownership is unambiguous.
- Duplicate lines—are there multiple copies of network elements or polygons? Duplicate polylines often occur during data capture. During the topological creation process, it is necessary to detect and remove duplicate polylines to ensure that topology can be built for a dataset.

Modeling the integrated behavior of different feature types In the real world, many objects share common locations and partial identities. For example, water distribution areas often coincide with water catchments, electric distribution areas often share common boundaries with building subdivisions, and multiple telecommunications fibers are often run down the same conduit. The data modeling challenge is how to model such integrated features effectively in a GI database. These situations can be modeled as either objects with shared geometries or as objects with separate geometries integrated for editing, analysis, and representation. There are advantages and disadvantages to both approaches. In the former, there are fewer coordinates to store, and feature integrity is inherent in the data. In the case of the latter, multiple objects with separate geometries are easier to implement in commercially available database management systems. If one feature is moved during editing, then logically both features should move. This is achieved by storing both objects separately in the database, each with its own geometry, but integrating them inside the GI editor application so that they are treated as single features. When the geometry of one is moved, the other automatically moves with it.

Editing productivity Topology improves editor productivity by simplifying the editing process and providing additional capabilities to manipulate feature geometries. Editing requires both topological data structuring and a set of topologically aware tools. The productivity of editors can be improved in several ways:

- Topology provides editors with the ability to manipulate common, shared polylines and nodes as single geometric objects to ensure that no differences are introduced into the common geometries.
- Rubberbanding is the process of moving a node, polyline, or polygon boundary and receiving interactive feedback on-screen about the location of all topologically connected geometry.
- Snapping (e.g., forcing the end of two polyline features to have the same coordinate) is a useful technique to both speed up editing and maintain a high standard of data quality.
- Autoclosure is the process of completing a polygon by snapping the last point to the first digitized point.
- Tracing is a type of network analysis technique that is used, especially in utility applications, to test the connectivity of linear features (e.g., is the newly designed service going to receive power?).

Optimizing queries Many GI system queries can be optimized by precomputing and storing information about topological relationships. Some common examples include the following:

- Network tracing (e.g., find all connected water pipes and fittings)
- Polygon adjacency (e.g., determine who owns the parcels adjoining those owned by a specific owner)
- Containment (e.g., find out which manholes lie within the pavement area of a given street)
- Intersection (e.g., examine which census tracts intersect with a set of health areas)

The remainder of this section focuses on a conceptual understanding of GI topology, emphasizing the polygon case. The next section considers the network case; the relative merits and implementations of two approaches to GI topology are discussed in Section 9.7.1. These two implementation approaches differ because in one case relationships are batch built and stored along with feature geometry, and in the other relationships are calculated interactively when they are needed.

Conceptually speaking, in a topologically structured polygon data layer, each polygon is defined as a collection of polylines that in turn are made up of an ordered list of coordinates (vertices). Figure 7.8 shows an example of a polygon dataset comprising six polygons (including the "outside world": Polygon 1). A number in a circle identifies a polygon. The lines that make up the polygons are shown in the area-polyline list. For example, Polygon 2 can be assembled from lines 4, 6, 7, 10, and 8. In this particular implementation example, the 0 before the 8 is used to indicate that Line 8 actually defines an "island" inside Polygon 2. The list of coordinates for each line is also shown in Figure 7.8. For example, Line 6 begins and ends with coordinates (7,4) and (6,3)—other coordinates have been omitted for brevity. A line may appear in the area-polyline list more than once (for example, Line 6 is used in the definition of both Polygons 2 and 5), but the actual coordinates for each polyline are only stored once in the polyline-coordinate

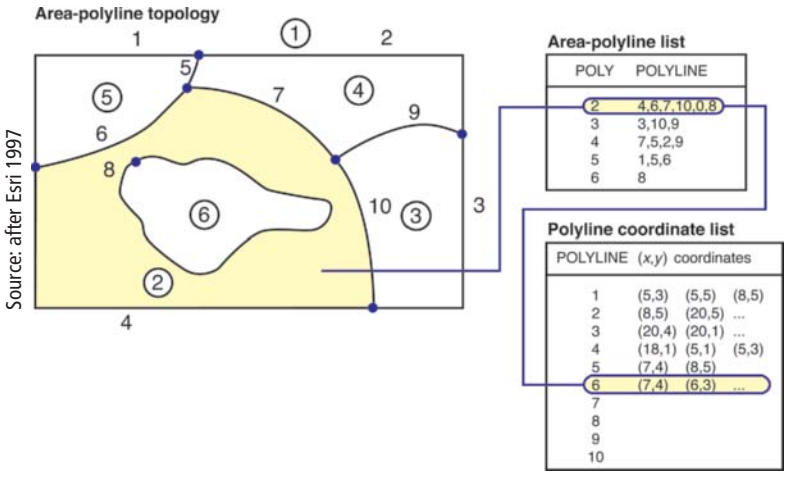


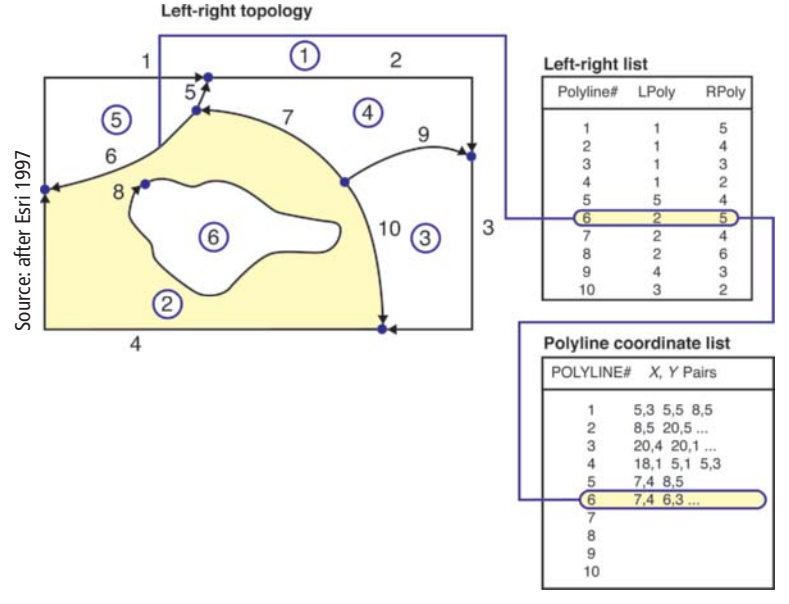
Figure 7.8 A topologically structured polygon (area) data layer. The polygons are made up of the polylines shown in the area polyline list. The lines are made up of the coordinates shown in the line coordinate list.

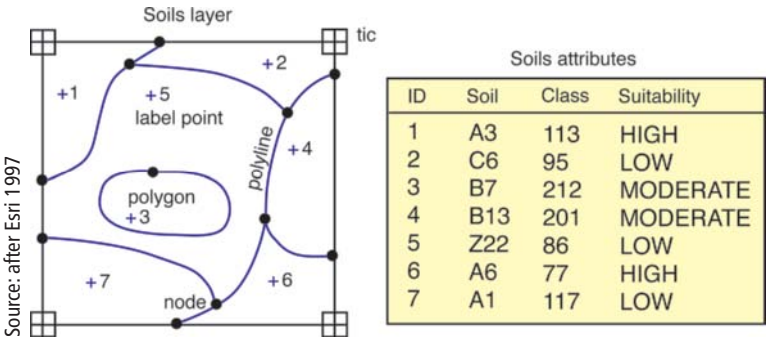
list. Storing common boundaries between adjacent polygons avoids the potential problems of gaps (slivers) or overlaps between adjacent polygons. It has the added bonus that there are fewer coordinates in a topologically structured polygon feature layer compared with a simple feature layer representation of the same entities. The downside, however, is that drawing a polygon requires that multiple polylines must be retrieved from the database and then assembled into a boundary. This process can be time consuming when repeated for each polygon in a large dataset.

Planar enforcement is a very important property of topologically structured polygons. In simple terms, planar enforcement means that all the space on a map must be filled and that any point must fall in one polygon alone; that is, polygons must not overlap. Planar enforcement implies that the phenomenon being represented is conceptualized as a field.

The contiguity (adjacency) relationship between polygons is also defined during the process of topological structuring. This information is used to define the polygons on the left- and right-hand sides of each polyline, in the direction defined by the list of coordinates (Figure 7.9). In Figure 7.9, Polygon 2 is on the left of Polyline 6 and Polygon 5 is on the right (polygon identifiers are in circles). Thus from a simple

Figure 7.9 The contiguity of a topologically structured polygon data layer. For each polyline the left and right polygons are stored with the geometry data.





Soils attributes

ID	Soil	Class	Suitability
1	A3	113	HIGH
2	C6	95	LOW
3	B7	212	MODERATE
4	B13	201	MODERATE
5	Z22	86	LOW
6	A6	77	HIGH
7	A1	117	LOW

Figure 7.10 An example of a georelational polygon dataset. Each of the polygons is linked to a row in an RDBMS table. The table has multiple attributes, one in each column.

look-up operation, we can deduce that Polygons 2 and 5 are adjacent.

Software systems based on the vector topological data model have become popular over the years. A special case of the vector topological model is the *georelational* model. In this model derivative, the feature geometries and associated topological information are stored in regular computer files, whereas the associated attribute information is held in relational database management system (RDBMS) tables. The GI system maintains the intimate linkage between the geometry, topology, and attribute information. This hybrid data management solution was developed to take advantage of RDBMS to store and manipulate attribute information. Geometry and topology were not placed in RDBMSs because, until a decade ago, RDBMSs were unable to store and retrieve geographic data efficiently. Figure 7.10 is an example of a georelational model as implemented in Esri's original ARC/INFO coverage polygon dataset. It shows file-based geometry and topology information linked to attributes in an RDBMS table. The ID (identifier) of the polygon, the label point, is linked (related or joined) to the ID column in the attribute table. Thus, in this soils dataset Polygon 3 is soil B7, of Class 212, and its suitability is MODERATE.

The topological feature geographic data model has been extensively used in GI applications over the last 25 years, especially in government and natural resources applications based on polygon representations. Typical government applications include cadastral management, tax assessment, land/property parcel management, land-use zoning, planning, and building control. In the areas of natural resources and environment, key applications include site suitability analysis, integrated land use modeling, license mapping, natural resource management, and conservation. Tax appraisal is an example of a GI application based on the topological feature data model; the topological data model is chosen in order to avoid overlaps and gaps in tax parcels (polygons), to ensure that all parcel boundaries closed (were validated), and to store data in an efficient way. This is despite the fact that there is an overhead in creating and maintaining parcel topology, as well as degradation in draw and query performance for large databases.

7.2.3.3 Network Data Model

The network data model is a special type of the topological feature model. It is discussed here separately because it raises several new issues and has been widely applied in GI projects.

Networks can be used to model the flow of goods and services. There are two primary types of networks: *radial* and *looped*. In radial or tree networks, flow always has an upstream and downstream direction. Stream and storm drainage systems are examples of radial networks. In looped networks, self-intersections are common occurrences. Water distribution networks are looped by design to ensure that service interruptions affect the fewest customers.

In GI databases, networks are modeled as points (for example, street intersections, fuses, switches, water valves, and the confluence of stream reaches, usually referred to as nodes in topological models) and lines (for example, streets, transmission lines, pipes, and stream reaches). Network topological relationships define how lines connect with each other at nodes. For the purpose of network analysis, it is also useful to define rules about how flows can move through a network. For example, in a sewer network, flow is directional from a customer (source) to a treatment plant (sink), but in a pressurized gas network flow can be in any direction. The rate of flow is modeled as impedances (weights) on the nodes and lines. Figure 7.11 shows an example of a street network. The network comprises a collection of nodes (types of street intersection) and lines (types of street), as well as the topological relationships between them. The topological information makes it possible, for

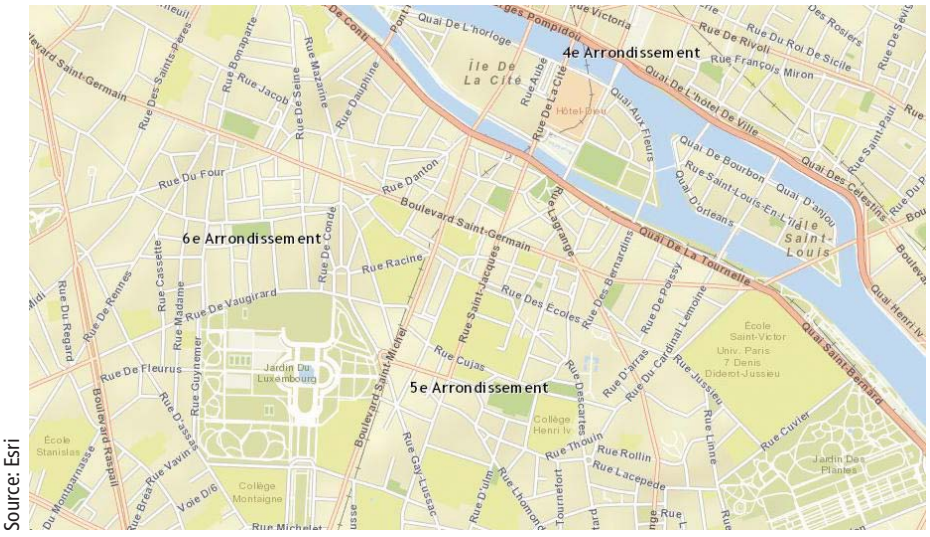


Figure 7.11 An example of a street network.

example, to trace the flow of traffic through the network and to examine the impact of street closures. An impedance defined on the intersections and streets determines the speed at which traffic flows. Typically, the rate of flow is proportional to the street speed limit, the number of lanes, and the timing of stoplights at intersections. Although this example relates to streets, the same basic principles also apply to, for example, electric, water, and railroad networks.

In georelational implementations of the topological network feature model, the geometry and topology information is typically held in ordinary computer files, and the attributes are stored in a linked database. The GI software tools are responsible for creating and maintaining the topological information each time a change in the feature geometry takes place. In more modern object models the geometry, attributes, and topology may be stored together in a DBMS, or topology may be computed on the fly.

Many applications utilize networks. Prominent examples include calculating power load drops over an electricity network, routing emergency response vehicles over a street network, optimizing the route of mail deliveries over a street network, and tracing pollution upstream to a source over a stream network.

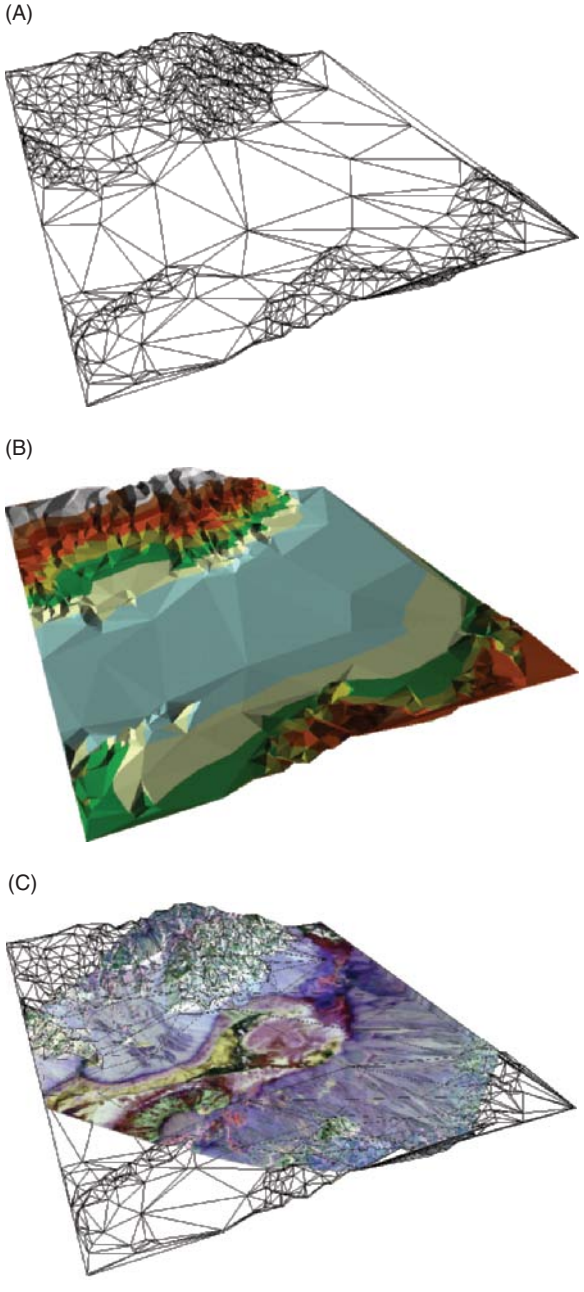
Network data models are also used to support another data model variant called *linear referencing* (see Section 4.5). The basic principle of linear referencing is quite simple. Instead of recording the locations of geographic entities as explicit *x*, *y*, *z* coordinates, they are recorded as distances along a network (called a route system) from a point of origin. This is a very efficient way of storing information such as road pavement (surface) wear characteristics (e.g., the location of potholes and degraded asphalt), geological seismic data (e.g., shockwave measurements at sensors along seismic lines), and pipeline corrosion data. An interesting aspect of this is that a two-dimensional network is reduced to a one-dimensional linear route list. The location of each entity (often called an event) is simply a distance along the route from the origin. Offsets are also often stored to indicate the distance from a network centerline. For example, when recording the surface characteristics of a multicarriageway road, several readings may be taken for each carriageway at the same linear distance along the route. The offset value will allow the data to be related to the correct carriageway. Dynamic segmentation is a special type of linear referencing. The term derives from the fact that event data values are held separately from the actual network route in database tables (still as linear distances and offsets) and then dynamically added to the route (segmented) each time the user queries the database. This approach is especially useful in situations in which the event data change frequently and need to be stored in a database due to access from other applications (e.g., traffic volumes or rate of pipe corrosion).

7.2.3.4 TIN Data Model

The geographic data models discussed so far have concentrated on one- and two-dimensional data. There are several ways to model three-dimensional data, such as terrain models, sales cost surfaces, and geological strata. The term *2.5-D* is sometimes used to describe surface structures because they have dimensional properties between 2-D and 3-D. A true 3-D structure will contain multiple *z* values at the same (*x*,*y*) location and thus is able to model overhangs and tunnels, as well as support accurate volumetric calculations such as cut and fill (a term derived from civil-engineering applications that describes cutting earth from high areas and placing it in low areas to construct a flat surface, as is required in,

for example, railroad construction). Both grids and triangulated irregular networks (TINs) are used to create and represent surfaces in GI databases. A regular grid surface is really a type of raster dataset, as discussed earlier in Section 7.2.2. Each grid cell stores the height of the surface at a given location. The TIN structure, as the name suggests, represents a surface as contiguous nonoverlapping triangular elements (Figure 7.12). A TIN is created from a set of points with *x*, *y*, and *z* coordinate values. A key advantage of the TIN structure is that the density of sampled points, and therefore the size of triangles, can be adjusted to reflect the relief of the surface being modeled, with more points sampled in areas of variable relief (see Section 2.4). TIN surfaces can be created by performing what is called a Delaunay triangulation (Figure 7.13; Section 13.3.6.1).

Figure 7.12 TIN surface of Death Valley, California: (A) "wireframe" showing all triangles; (B) shaded by elevation; and (C) draped with satellite image.

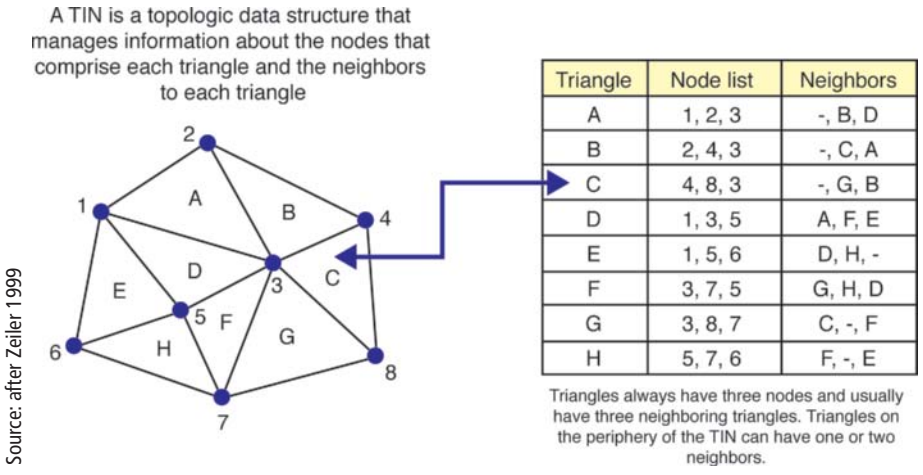


A TIN is a topological data structure that manages information about the nodes comprising each triangle and the neighbors of each triangle. Figure 7.13 shows the topology of a simple TIN. As with other topological data structures, information about a TIN may be conveniently stored in a file or database table, or computed on the fly. TINs offer many advantages for surface analysis. First, they incorporate the original sample points, providing a useful check on the accuracy of the model. Second, the variable density of triangles means that a TIN is an efficient way of storing surface representations such as terrains that have substantial variations in topography. Third, the data structure makes it easy to calculate elevation, slope, aspect, and line-of-sight between points. The combination of these factors has led to the widespread use of the TIN data structure in applications such as volumetric calculations for roadway design, drainage studies for land development, and visualization of urban forms. Figure 7.14 shows two example applications of TINs. Figure 7.14A is a shaded landslide risk TIN of the Pisa district in Italy, with building objects draped on top to give a sense of landscape. Figure 7.14B is a TIN of the Yangtse River, China, greatly exaggerated in the *z* dimension. It shows how TINs draped with images can provide photorealistic views of landscapes.

Like all 2.5-D and 3-D models, TINs are only as good as the input sample data (see Section 2.4). They are especially susceptible to extreme high and low values because there is no smoothing of the original data. Other limitations of TINs include their inability to deal with discontinuity of slope across triangle boundaries, the difficulty of calculating optimum routes, and the need to ensure that peaks, pits, ridges, and channels are captured if a drainage network TIN is to be accurate.

7.2.4 Object Data Model

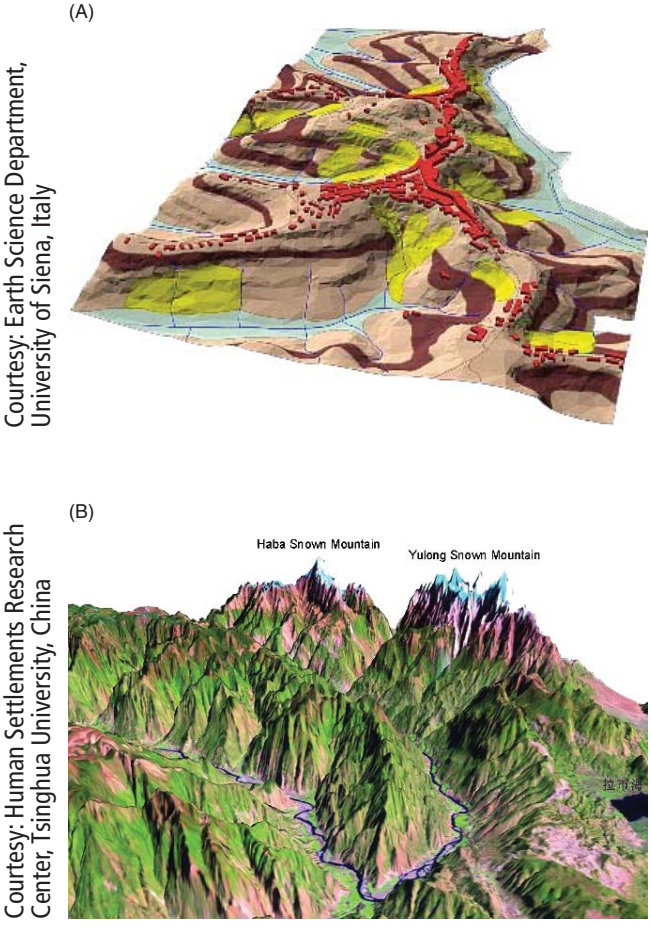
All the geographic data models described so far are geometry-centric; that is, they model the world as collections of points, lines, areas, TINs, or rasters. Any operations to be performed on the geometry (and, in some cases, associated topology) are created as procedures (programs or scripts) that are separated from the data. Unfortunately, this approach can present several limitations for modeling geographic systems. All but the simplest of geographic systems



Triangle	Node list	Neighbors
A	1, 2, 3	-, B, D
B	2, 4, 3	-, C, A
C	4, 8, 3	-, G, B
D	1, 3, 5	A, F, E
E	1, 5, 6	D, H, -
F	3, 7, 5	G, H, D
G	3, 8, 7	C, -, F
H	5, 7, 6	F, -, E

Figure 7.13 The topology of a TIN.

Figure 7.14 Examples of applications that use the TIN data model: (A) Landslide risk map for Pisa, Italy; (B) Yangtse River, China.



contain many entities with large numbers of properties, complex relationships, and sophisticated behavior. Modeling such entities as simple geometry types is overly simplistic and does not easily support the sophisticated characteristics required for modern analysis. In addition, separating the state of an entity (attributes or properties defining what it *is*) from the behavior of an entity (methods defining what it *does*) makes software and database development tedious, time-consuming, and error prone. To try to address these problems, geographic object data models were developed; they allow the full richness of geographic systems to be modeled in an integrated way in GI databases.

The central focus of a GI object data model is the collection of geographic objects and the relationships between the objects (see Box 7.2). Each geographic object is an integrated package of geometry, properties, and methods. In the object data model, geometry is treated like any other attribute of the object and not as its primary characteristic (although clearly from an applications perspective it is often the major property of interest). Geographic objects of the same type are grouped together as object classes, with individual objects in the class referred to as "instances." In many GI systems each object class is stored physically as a database table, with each row an object and each object property a column. The methods that apply are attached to the object instances when they are created in memory for use in the application.

An object is the basic atomic unit in an object data model and comprises all the properties that define the state of an object, together with the methods that define its behavior.

Technical Box 7.2

Object-Oriented Concepts

An ***object*** is a self-contained package of information describing the characteristics and capabilities of an entity (a thing of interest) under study. An interaction between two objects is called a ***relationship***. In a geographic object data model, the real world is modeled as a collection of objects and the relationships between the objects. Each entity in the real world to be included in the GI database is an object. A collection of objects of the same type is called a ***class***. In fact, classes are a more central concept than objects from the implementation point of view because many of the object-oriented characteristics are built at the class level. A class can be thought of as a template for objects. When creating an object data model the designer specifies classes and the relationships between classes. Only when the data model is used to create a database are objects (instances or examples of classes) actually created.

Examples of objects include oil wells, soil bodies, stream catchments, and aircraft flight paths. In the case of an oil-well class, each oil-well object might include ***properties*** defining its ***state***—annual production, owner name, date of construction, and type of geometry used for representation at a given scale (perhaps a point on a coarse-scale map and a polygon on a fine-scale one). The oil-well class could have connectivity relationships with a pipeline class that represents the pipeline used to transfer oil to a refinery. There could also be a relationship defining the fact that each well must be located on a drilling platform. Finally, each oil-well object might also have ***methods*** defining the ***behavior*** or what it can do. Example behavior might include how objects draw themselves on a computer screen, how objects can be created and deleted, and editing rules about how oil wells snap to pipelines.

Three key facets of object data models make them especially good for modeling geographic systems: encapsulation, inheritance, and polymorphism.

Encapsulation describes the fact that each object packages together a description of its state and behavior. The state of an object can be thought of as its properties or attributes (e.g., for a forest object, it could be the dominant tree type, average tree age, and soil pH). The behavior is the sum of the methods or operations that can be performed on an object (for a forest object these could be create, delete, draw, query, split, and merge). For example, when splitting a forest polygon into two parts, perhaps following a part sale, it is useful to get the GI system to automatically calculate the areas of the two new parts. Combining the state and behavior of an object together in a single package is a natural way to think of geographic entities and a useful way to support the reuse of objects.

Inheritance is the ability to reuse some or all of the characteristics of one object in another object. For example, in a gas facility system a new type of gas valve could easily be created by overwriting or adding a few properties or methods to a similar existing type of valve. Inheritance provides an efficient way to create models of geographic systems by reusing objects and also a mechanism to extend models easily. New object classes can be built to reuse parts of one or more existing object classes and add some new unique properties and methods. The example described in Section 7.3 shows how inheritance and other object characteristics can be used in practice.

Polymorphism describes the process whereby each object has its own specific implementation for operations like draw, create, and delete. One example of the benefit of polymorphism is that a geographic database can have a generic object-creation component that issues requests to be processed in a specific way by each type of object class. A utility system's editor software can send a generic create request to all objects (e.g., gas pipes, valves, and service lines), each of which has specific create algorithms. If a new object class is added to the system (e.g., landbase), then this mechanism will work because the new class is responsible for implementing the create method. Polymorphism is essential for isolating parts of software as self-contained components (see Chapter 6).

All geographic objects have some type of relationship to other objects in the same object class and, possibly, to objects in other object classes. Some of these relationships are inherent in the class definition (for example, some GI systems remove overlapping polygons), whereas other interclass relationships are user-definable. Three types of relationships are commonly used in geographic object data models: topological, geographic, and general.

A class is a template for creating objects.

Generally, topological relationships are built into the class definition. For example, modeling real-world entities as a network class will cause network

topology to be built for the nodes and lines participating in the network. Similarly, real-world entities modeled as topological polygon classes will be structured using the node–polyline model described in Section 7.2.3.2.

Geographic relationships between object classes are based on geographic operators (such as overlap, adjacency, inside, and touching) that determine the interactions between objects. In a model of an agricultural system, for example, it might be useful to ensure that all farm buildings are within a farm boundary using a test for geographic containment.

General relationships are useful to define other types of relationships between objects. In a tax assessment system, for example, it is advantageous to define a relationship between land parcels (polygons) and ownership data that is stored in an associated DBMS table. Similarly, an electric distribution system relating light poles (points) to text strings (called annotation) allows depiction of pole height and material of construction on a map display. This type of information is very valuable for creating work orders (requests for change) that alter the facilities. Establishing relationships between objects in this way is useful because if one object is moved, then the other will move as well, or if one is deleted, then the other is also deleted. This makes maintaining databases much easier and safer.

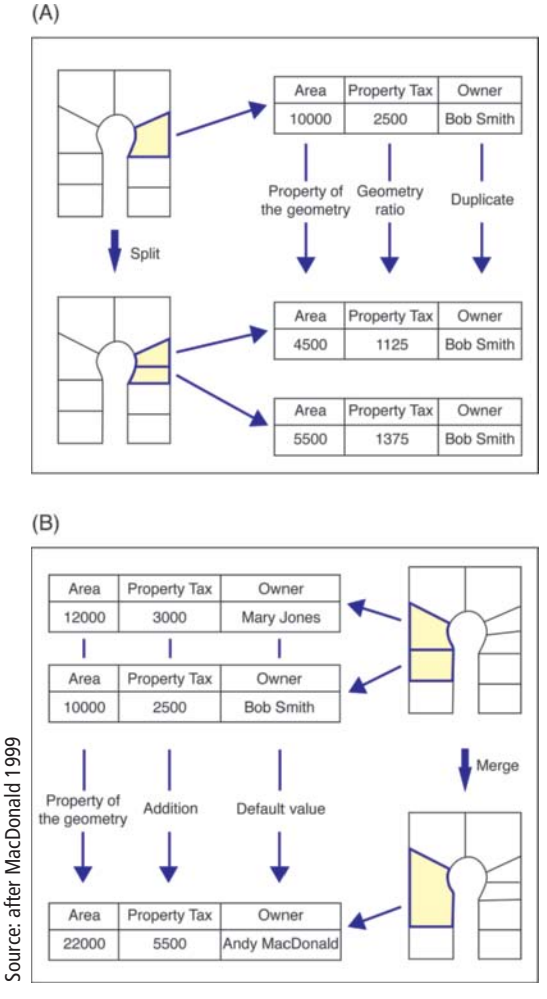
In addition to supporting relationships between objects (strictly speaking, between object classes), object data models also allow several types of rules to be defined. Rules are a valuable means of maintaining database integrity during editing tasks. The most popular types of rules used in object data models are attribute, connectivity, relationship, and geographic.

Attribute rules are used to define the possible attribute values that can be entered for any object. Both range and coded-value attribute rules are widely employed. A range attribute rule defines the range of valid values that can be entered. Examples of range rules include the following: highway traffic speed must be in the range 25–70 miles per hour; forest compartment average tree height must be in the range of 0–50 meters. Coded attribute rules are used for categorical data types. For example, land use must be of type commercial, residential, park, or other; or pipe material must be of type steel, copper, lead, or concrete.

Connectivity rules are based on the specification of valid combinations of features, derived from the geometry, topology, and attribute properties. For example, in an electric distribution system a 28.8-kV conductor can only connect to a 14.4-kV conductor via a transformer. Similarly, in a gas distribution system it should not be possible to add pipes with free ends (that is, with no fitting or cap) to a database.

Geographic rules define what happens to the properties of objects when an editor splits or merges them (Figure 7.15). In the case of a land parcel split following the sale of part of the parcel, it is useful to define rules to determine the impact on properties such as area, land-use code, and owner. In this example, the original parcel area value should be divided in proportion to the size of the two new parcels, the land-use code should be transferred to both parcels, and the owner name should remain for one parcel, but a new one should be added for the part that was sold off. In the case of a merge of two adjacent water pipes, decisions need to be made about what happens to attributes such as material, length, and corrosion rate. In this example, the two pipe materials should be the same, the lengths should be summed, and the new corrosion rate determined by a weighted average of both pipes.

Figure 7.15 Example of split and merge rules for parcel objects: (A) split; and (B) merge.



Source: after MacDonald 1999

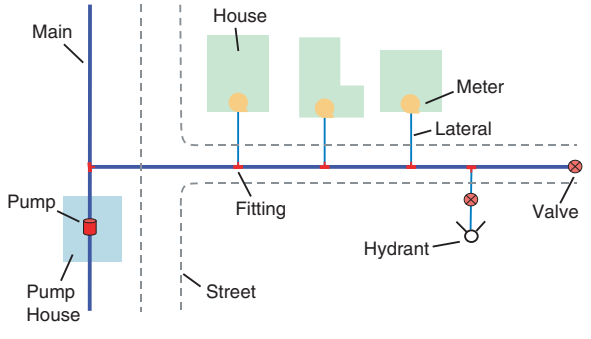
7.3 Example of a Water-Facility Object Data Model

This section presents an example of a geographic object model and discusses how many of the concepts introduced earlier in this chapter are used in practice. The example selected is that of an urban water facility model. The types of issues raised in this example apply to all geographic object models, although of course the actual objects, object classes, and relationships under consideration will differ. The role of data modeling, as discussed in Section 7.1, is to represent the key aspects of the real world inside the digital computer for management, analysis, and display purposes.

Figure 7.16 is a diagram of part of a water distribution system, a type of pressurized network controlled by several devices. A pump is responsible for moving water through pipes (mains and laterals) connected together by fittings. Meters measure the rate of water consumption at houses. Valves and hydrants control the flow of water.

The goal of the example object model is to support asset management, mapping, and network analysis applications. Based on this goal it is useful to classify the objects into two types: the landbase and the water facilities. Landbase is a general term for objects such as houses and streets that provide geographic context but are not used in network analysis. The landbase object types are Pump House, House, and Street. The water-facilities object types are Main, Lateral (a smaller type of *WaterLine*), Fitting (water line connectors), Meter, Valve, and Hydrant. All these object types need to be modeled as a network in order to support network analysis operations such as network isolation traces and flow prediction. A network isolation trace is used to find all parts of a network that are unconnected (isolated). By using the topological connectivity of the network and information about whether pipes and fittings support water flow, it is possible to determine connectivity. Flow prediction is used to estimate the flow of water through

Figure 7.16 Water distribution system water-facility object types and geographic relationships.



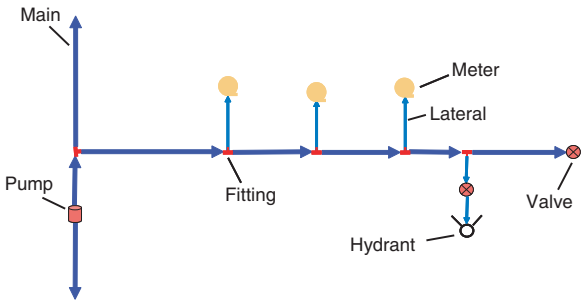
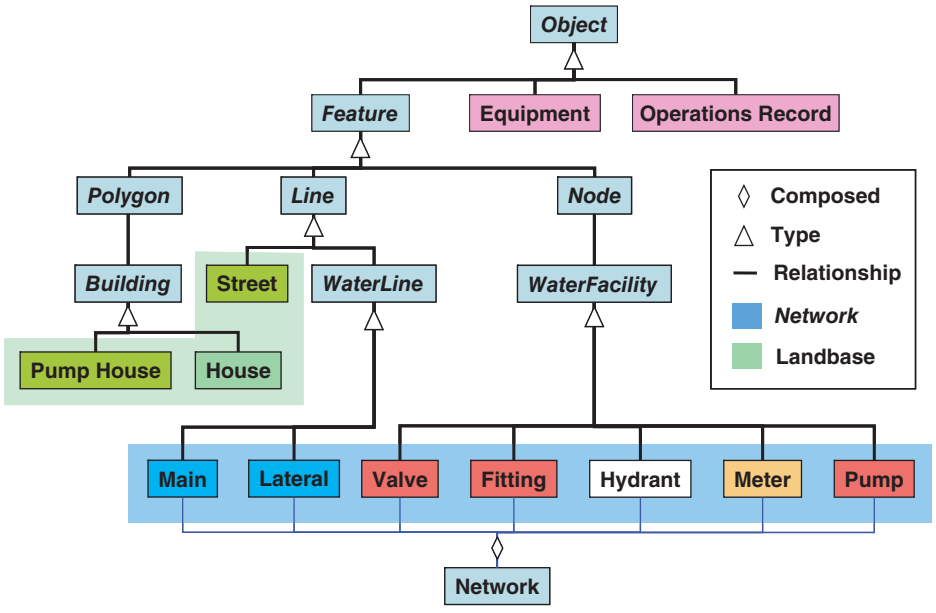


Figure 7.17 Water distribution system network.

the network based on network connectivity and data about water availability and consumption. Figure 7.17 shows all the main object types and the implicit geographic relationships to be incorporated into the model. The arrows indicate the direction of flow in the network. When digitizing this network using a GI editor, it is useful to specify topological connectivity and attribute rules to control how objects can be connected (see Section 7.2.3.3). Before this network can be used for analysis, it will also be necessary to add flow impedances to each link (for example, pipe diameter).

Having identified the main object types, the next step is to decide how objects relate to each other and the most efficient way to implement them. Figure 7.18 shows one possible object model that uses a popular object modeling language called the Unified Modeling Language (UML) to show objects and the relationships between them. Some additional color-coding has been added to help interpret the model. In UML models each box is an object class, and the lines define how one class reuses (inherits) part of the class above it in a hierarchy. Object class names in an italic font are abstract classes; those with regular font names are used to create (instantiate) actual object instances. Abstract classes do not have instances and exist for reasons of efficiency. It is sometimes useful to have a class that implements some capabilities once, so that several other classes can then be reused. For example, Main and Lateral are both types of *Line*, as is Street. Because Main and Lateral share several things in common—such as ConstructionMaterial, Diameter, and InstallDate properties, and connectivity and draw behavior—it is efficient to implement these in a separate abstract class, called *WaterLine*. The triangles indicate that one class is a type of another class. For example, Pump House and House are types of *Building*, and Street and *WaterLine* are types of *Line*. The diamonds indicate composition. For example, a network is composed of a collection of *Line* and *Node* objects. In the water-facility object model, object classes without any geometry are colored pink. The Equipment and Operations-Record object classes have their location determined by being associated with other objects (e.g., valves and

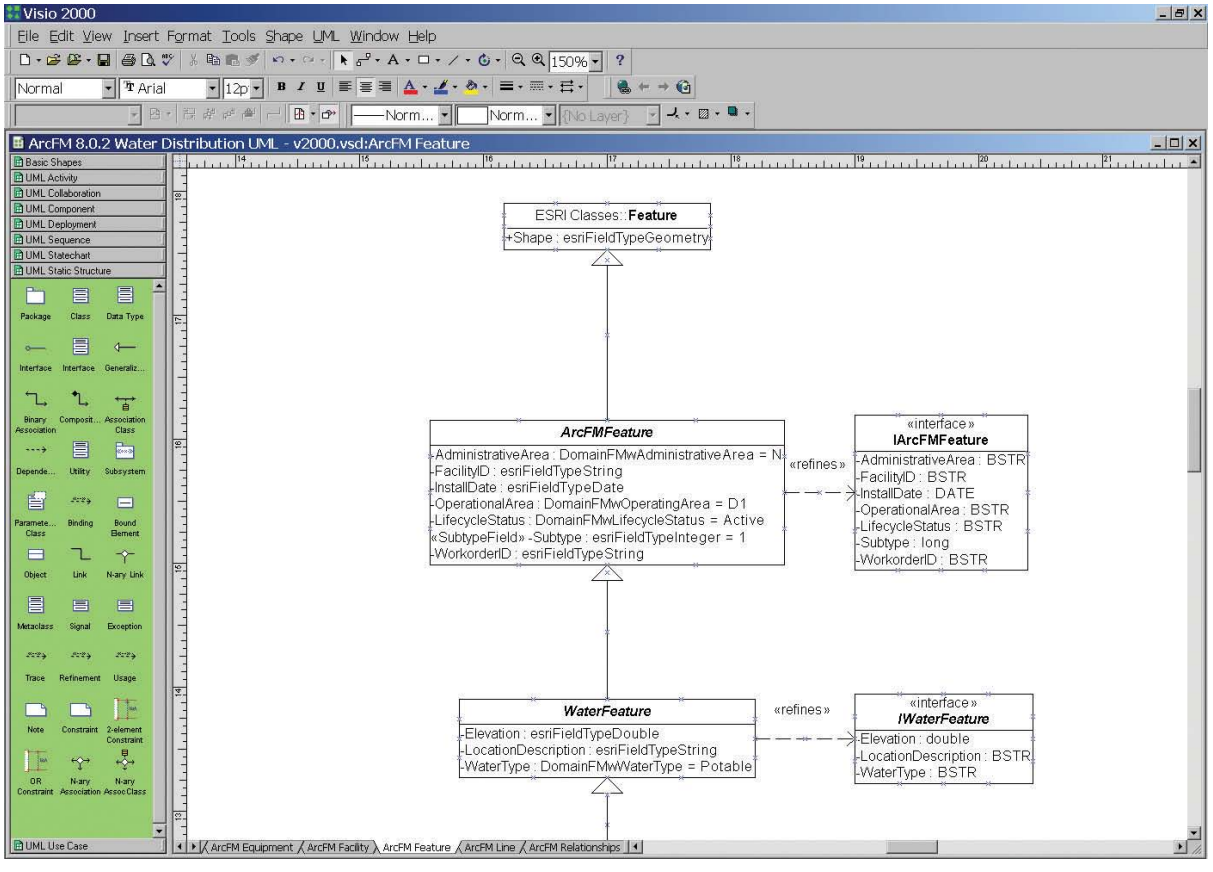
Figure 7.18 A water-facility object model.



mains). The Equipment and OperationsRecord classes are useful places to store properties common to many facilities, such as EquipmentID, InstallDate, ModelNumber, and SerialNumber.

Once this logical geographic object model has been created, it can be used to generate a physical data model. One way to do this is to create the model using a computer-aided software engineering (CASE) tool. A CASE tool is a software application that has graphical tools that draw and specify a logical model (Figure 7.19). A further advantage of a CASE tool is that physical models can be generated directly from

Figure 7.19 An example of a CASE tool (Microsoft Visio). The UML model is for a utility water system.



the logical models, including all the database tables and much of the supporting code for implementing behavior. Once a database structure (schema) has been created, it can be populated with objects and the intended applications put into operation.

7.4 Geographic Data Modeling in Practice

Geographic analysis is only as good as the geographic database on which it is based, and a geographic database is only as good as the geographic data model from which it is derived. Geographic data modeling begins with a clear definition of the project goals and progresses through an understanding of user requirements, a definition of the objects and relationships, formulation of a logical model, and then creation of a physical model. These steps are a prelude to database creation and, finally, database use.

No step in data modeling is more important than understanding the purpose of the data-modeling exercise. This understanding can be gained by collecting user requirements from the main users. Initially, user requirements will be vague and ill-defined, but over time they will become clearer. Project goals and user requirements should be precisely specified in a list or narrative. Peter Batty's career and experience of GI data modeling is highlighted in Box 7.3.

Formulation of a logical model necessitates identification of the objects and relationships to be modeled. Both the attributes and behavior of objects are required for an object model. A useful graphic

Biographical Box 7.3

Peter Batty: Geospatial Data Modeler

Peter Batty has served as Chief Technology Officer at several major geospatial software companies and has been involved in a number of innovations in the industry.

He studied at Oxford University, receiving a Bachelor's degree in Mathematics and a Master's degree in Computation. He began work for IBM in 1986 on their Geographic Facilities Information System (GFIS), a pioneering GI system focused on electric and gas utilities. Peter was an early advocate of using standard relational databases for GIS. He wrote an influential article, "Exploiting Relational Database Technology in GIS," in 1990[1], which was referenced by Oracle at the initial launch of Oracle Spatial (then Oracle MM) in 1995.

In 1992 Peter went to work for Smallworld Systems, at the time a small startup in the UK, which grew to become the market leader for GI systems in utilities and telecommunications during the 1990s. Peter was the first employee to move to the United States when Smallworld began operating there in 1993, and he held various technical roles before becoming Chief Technology Officer.

Smallworld introduced several innovations that had a significant impact on the industry, most notably the handling of long transactions using a new approach called version management. This eventually became a standard approach in the industry, being adopted in some form by Esri, Intergraph, and Oracle, among others. Long transactions are fundamental to handling the design and planning process in utilities.

The Smallworld data model introduced a number of new ideas that were especially good for modeling complex networks. One of these was the ability for a feature to have multiple spatial attributes, rather than a feature having just a single spatial type, as is still common today. Another was the notion of "internal worlds" that schematically represented the network connections inside a complex object such

Courtesy: Peter Batty

Figure 7.20 Peter Batty

as a switchgear or substation, in a separate coordinate space from the main "geographic world".

In 1997–98, Peter led the development of PowerOn, a new product for electric outage management built on top of Smallworld, which quickly became the outage management market leader. A key innovation in PowerOn was a "tree key" hierarchical indexing scheme that enabled very fast updates of large numbers of electrical devices when power switched on or off.

In 2000, Smallworld was acquired by General Electric, and in 2002 Peter left to cofound a company called Ten Sails, now known as Ubisense. From 2005 to 2007 he left to serve as CTO of Intergraph, at that time the second-largest geospatial software company. In 2007 he founded a startup called Spatial Networking that worked on future location and social networking. He returned to Ubisense in 2010, and his current work is focused on applying the new generation of Web mapping systems, including Google Maps and a number of open-source software packages, to enterprise utility and telecom applications.

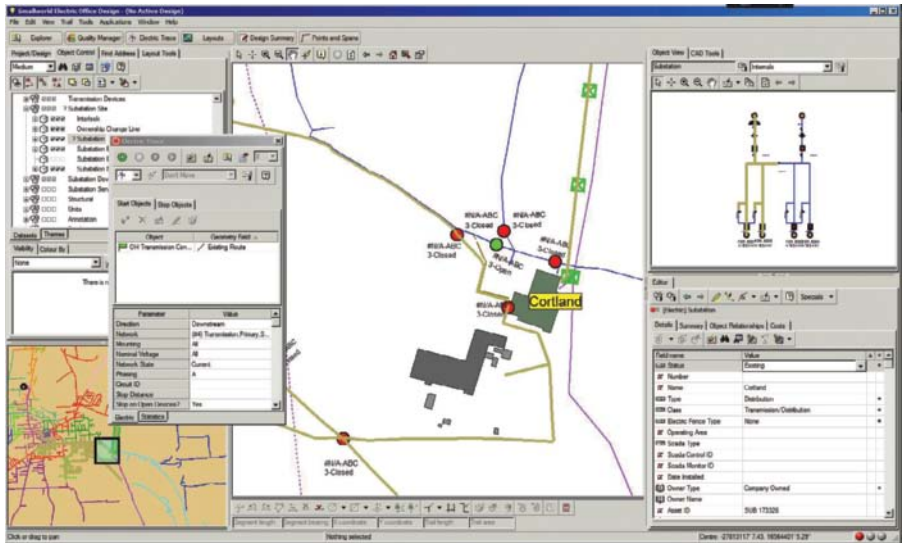
Peter has in recent years become quite involved with Open Source geospatial software. In 2011 he chaired the FOSS4G conference in Denver, the leading global conference for open-source geospatial software. This was the largest FOSS4G event to date, attracting 900 attendees from 42 countries. He served on the board of OSGeo, the Open Source Geospatial Foundation, from 2011 to 2013. Previously he also served on the board of the Geospatial Information and Technology Association (GITA) for a total of 5 years.

[1] www.ebatty.com/exploiting_rdb_in_gis.htm (originally published in 1990, republished in Computers and Geosciences in 1992)

tool for creating logical data models is a CASE tool, and a useful language for specifying models is UML. It is not essential that all objects and relationships be identified at the first attempt because logical models can be refined over time. The key objects and relationships for a water distribution system object model are shown in Figure 7.18.

Once an implementation-independent logical model has been created, this model can be turned into a system-dependent physical model. A physical model

Figure 7.21 Smallworld "internal worlds" screenshot. This screenshot shows the concept of "internal worlds" that can be used to model complex network devices in Smallworld. The main map shows an electric substation, represented by a polygon geometry. This contains a number of switches and other electric devices that need to be independently modeled in the GI system. The top right window shows a schematic representation of the devices inside the substation. These are stored in an independent "world" with its own (nongeographic) coordinate system. This world is owned by the substation. The highlighted cables on the map show an electric network trace that starts at the top right, then when it reaches the substation it jumps into the internal world and continues tracing there, looking at which switches are open and closed, then it jumps back out to the geographic world and continues tracing along two of the circuits that leave the substation.



Courtesy: Peter Batty

will result in an empty database schema—a collection of database tables and the relationships between them. Sometimes, for performance optimization reasons or because of changing requirements, it is necessary to alter the physical data model. Even at this relatively late stage in the process, flexibility is still necessary.

It is important to realize that there is no such thing as the correct geographic data model. Every problem can be represented with many possible data models. Each data model is designed with a specific purpose in mind and is suboptimal for other purposes. A classic dilemma is whether to define a general-purpose data model that has wide applicability, but that can, potentially, be complex and inefficient, or to focus on a narrower highly optimized model that will yield better performance. A small prototype can often help resolve some of these issues.

Geographic data modeling is both an art and a science. It requires a scientific understanding of the key geographic characteristics of real-world systems, including the state and behavior of objects, and the relationships between them. Geographic data models are of critical importance because they have a controlling influence over the type of data that can be represented and the operations that can be performed. As we have seen, object models are the best type of data model for representing rich object types and relationships in facility systems, whereas simple feature models are sufficient for elementary applications such as a map of the body. In a similar vein, so to speak, raster models are good for data represented as fields such as soils, vegetation, pollution, and population counts.

Questions for Further Study

1. Figure 7.22 is an oblique aerial photograph of an area of northern Italy. Take 10 minutes to list the main five object classes (including their attributes and behaviors) and the relationships between the classes that you can see in this picture that would be appropriate for a facilities information management system study.
2. Why is it useful to include the conceptual, logical, and physical levels in geographic data modeling?
3. Describe, with examples, five key differences between the topological vector and raster geographic data models. It may be useful to consult Figure 7.3 and Chapters 2 and 3.
4. Review the terms *encapsulation, inheritance,* and *polymorphism*, and explain with geographic examples why they make object data models superior for representing geographic systems.

Figure 7.22 Oblique view northern Italy

Further Reading

Arctur, D. and Zeiler, M. 2004. *Designing Geodatabases: Case Studies in GIS Data Modeling*. Redlands, CA: Esri Press.

Esri. 1999. *Understanding GIS: The ArcInfo Method*. Redlands, CA: Esri Press.

Perencsik, A., Woo, S., Booth, B., Crosier S., Clark, J., and MacDonald, A. 2004. *Building a Geodatabase*. Redlands, CA: Esri Press.

Worboys, M. F. and Duckham, M. 2004. *GIS: A Computing Perspective* (2nd ed.). Boca Raton, FL: CRC Press.

Zeiler, M. and Murphy, J. 2010. *Modeling Our World: The Esri Guide to Geodatabase Concepts* (2nd ed). Redlands, CA: Esri Press.