9.1 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* describe essentially the same thing. Representation is more widely used to describe the conceptual and scientific issues, whereas model is used in practical and database circles. In this chapter the term ''model'' will be used, given the practical approach. This chapter focuses on how geographic reality is modeled (abstracted or simplified) in a GIS, with particular emphasis on choosing one particular style of data model over another.

9.1.1 What is a data model?

The heart of any GIS is the data model, which is a set of constructs for representing objects and processes in the digital environment of the computer (Figure 9.1). People (GIS users) interact with operational GIS in order to perform tasks like 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 model to be adopted are vital to the success of a GIS 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 Chapter 3, geographic reality is infinitely complex, but computers are finite. Therefore, difficult choices have to be made about what and how things are modeled using GIS. Because different types of people use GIS for different purposes, and the type of phenomena people study have different characteristics, there is no single type of GIS data model that is best for all circumstances.

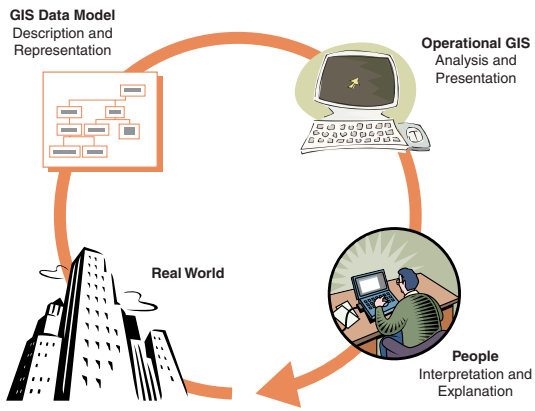


Figure 9.1 The role of a data model in GIS

9.1.2 Levels of data model abstraction

When representing the real world in a computer, it is helpful to think in terms of the four different levels of abstraction (levels of generalization or simplification) that are shown in Figure 9.2. 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. Fourth, the *physical model* portrays the actual application in a GIS, and often comprises tables stored as files or databases (see Chapter 11). Use of the term physical here is actually misleading because the models are not physical, they only exist digitally in computers.

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 definition of the main types of objects to be represented in the GIS and concludes with a conceptual description of the main types of objects and relationships between

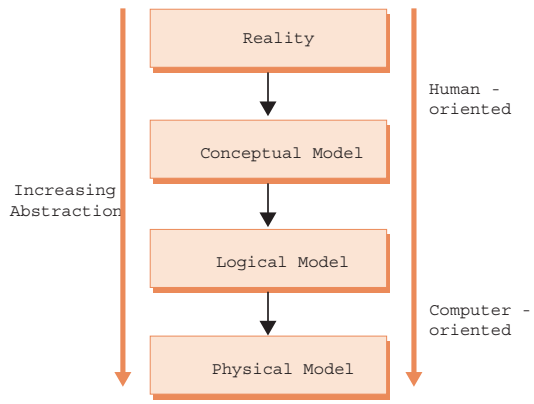


Figure 9.2 Levels of abstraction relevant to GIS data models

them. Once this phase is complete, further work will lead to the creation of diagrams and lists describing the names of objects, their behavior, and the type of interaction between objects. This type of logical data model is very valuable for defining what a GIS will do and the type of domain over which it will extend. Logical models are implementation independent, and can be created in any GIS with appropriate capabilities. The final data modeling phase involves creating a model showing how the objects under study can be digitally implemented in a GIS. 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 9.3 and 9.4.

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 implementation of 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 (Worboys 1995).

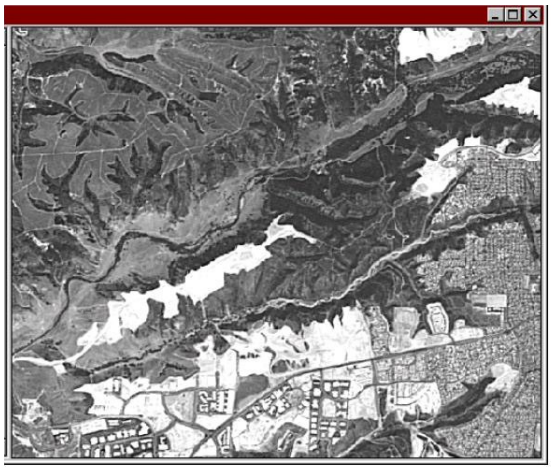
The discussion of geographic representation in Chapter 3 introduced discrete objects and fields, the two fundamental conceptual models for representing real-world objects geographically. In the same chapter the raster and vector logical models were also introduced. Figure 9.3 shows two representations of raster and vector objects in a GIS. Notice the difference in the objects represented. Major roads, in false color green, and areas cleared of vegetation in false color red, are more clearly visible in the vector representation, whereas smaller roads and built-up areas can best be seen on the scanned raster aerial photograph. The next sections in this chapter focus on the logical and physical representation of raster, vector, and related models in GIS software systems.

9.2 GIS Data Models

In the past half-century many GIS data models have been developed and deployed in GIS software systems. The key types of geographic data models and their main areas of application are listed in Table 9.1. All are based in some way on the conceptual discrete object/field and logical vector/raster geographic data models.

All GIS software systems include a core data model that is built on one or more of the GIS data models described below. As discussed earlier, the 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 GIS 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 GIS software packages come with a fixed data model, while others have models that can be easily extended. Those that can easily be extended are

(A)



(B)

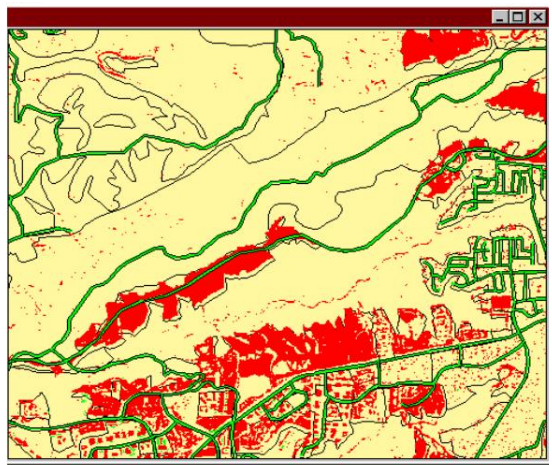


Figure 9.3 Representations of the same area (San Diego, California): (A) panchromatic SPOT raster satellite image collected in 1990 at 10 m resolution (Courtesy: ERDAS); (B) vector objects digitized from the image (Courtesy: ERDAS)

Table 9.1 Geographic data models used in GIS

Data Model	Application
Computer-aided design (CAD)	Engineering design
Graphical (non-topologic)	Simple mapping
Image	Image processing and simple grid analysis
Raster/grid	Spatial analysis and modeling especially in environmental and natural resources applications
Vector/geo-relational topologic	Many operations on vector geometric features in cartography, socio-economic and resource analysis, and modeling
Network	Network analysis in transportation, hydrology, and automated mapping/facilities management (AM/FM)
Triangulated irregular network (TIN)	Surface/terrain analysis and modeling
Object	Many operations on all types of entities in all types of applications

better able to model the richness of geographic domains, and in general are the easiest to use and are the most productive systems.

When modeling the real world for representation inside a GIS it is convenient to group entities of the same geometric type together (for example, all point entities such as lights, garbage cans, dumpsters, etc. 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 GIS as a general term for a distinct dataset. It derived from the process of entering different types of data into a GIS 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 of this see Section 11.3). It also makes it much easier to implement rules for validating edit operations (e.g. addition of a new building or census administrative area) and for building relationships between entities. All of the data models discussed below 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 polygons). Grouped layers may combine layers of different geometric types.

9.2.1 CAD, graphical, and image GIS data models

The earliest GIS were based on very simple models derived from work in the fields of CAD (computer-aided design), computer cartography, and image analysis. In a CAD system real-world entities are represented symbolically as simple point, line, and polygon vectors. This basic CAD data model never became widely popular in GIS 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 GIS applications. Third, because CAD data models are focused on graphical representation of objects they do not store details of any relationships between objects (e.g. topology), the type of information essential in many spatial analytical operations.

A second type of simple GIS 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 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 polygons, with annotation used for placenames. 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 source for geographic image processing is 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

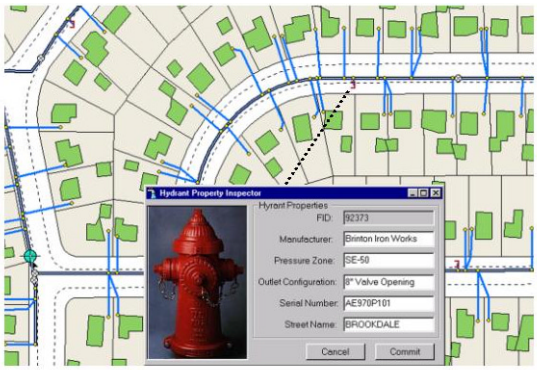


Figure 9.4 An image of a hydrant used as an object attribute in a water facility system

objects on the Earth 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 9.4).

In spite of their many limitations GIS still exist based on these simple data models. This is partly for historical reasons – the GIS may have been purchased before newer, more advanced models became available – but also because of lack of knowledge about the newer approaches described below.

9.2.2 Raster data model

The raster data model uses an array of cells, or pixels, to represent real-world objects (Figure 3.7). The cells can hold any attribute values based on one of several encoding schemes including categories, and integer and floating point numbers (see Box 3.3 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 9.5).

Raster data are usually stored as an array of grid values, with metadata about the array held in a file *header*. Typical metadata include the geographic coordinate of the upper-left corner of

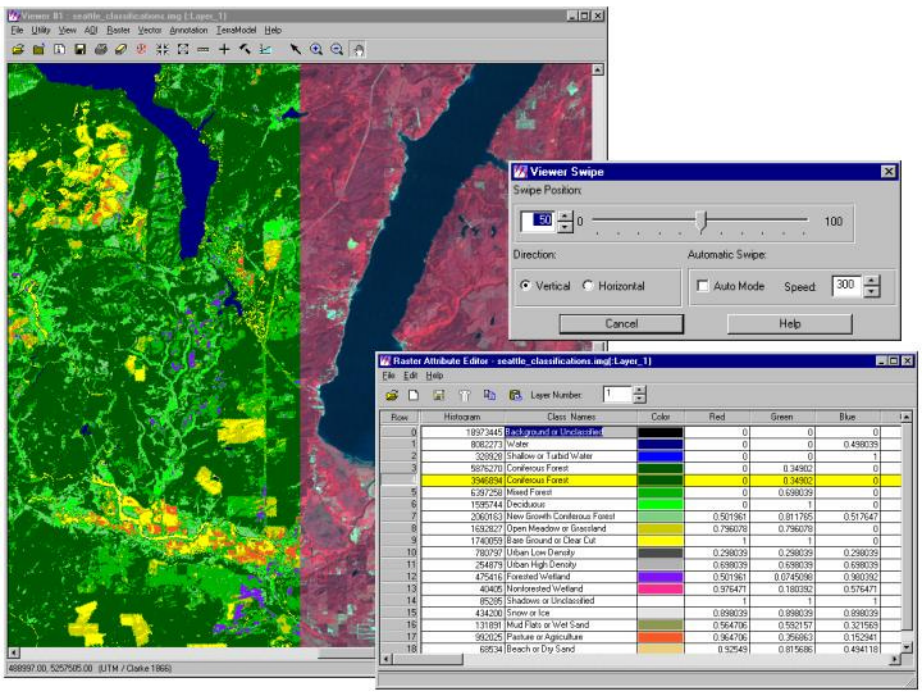


Figure 9.5 Raster data of the Olympic Peninsula, Washington State, USA, with associated value attribute table. Bands 4, 3, 2 from Landsat 5 satellite with land cover classification overlain (ERDAS Imagine: screenshot courtesy of ERDAS Inc.; data courtesy of EROS Data Center)

Box 9.1 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 technique have been developed such as run-length encoding, block encoding, wavelet compression, and quadtrees (see Section 11.7.2.2 for another use of quadtrees as a means to index geographic data). Table 9.2 presents a comparison of file sizes and compression rates for three compression techniques based on the image in Figure 9.6. It can be seen that even the comparatively simple run-length encoding technique (Figure 3.11) 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 90.5 to 2.3 MB.

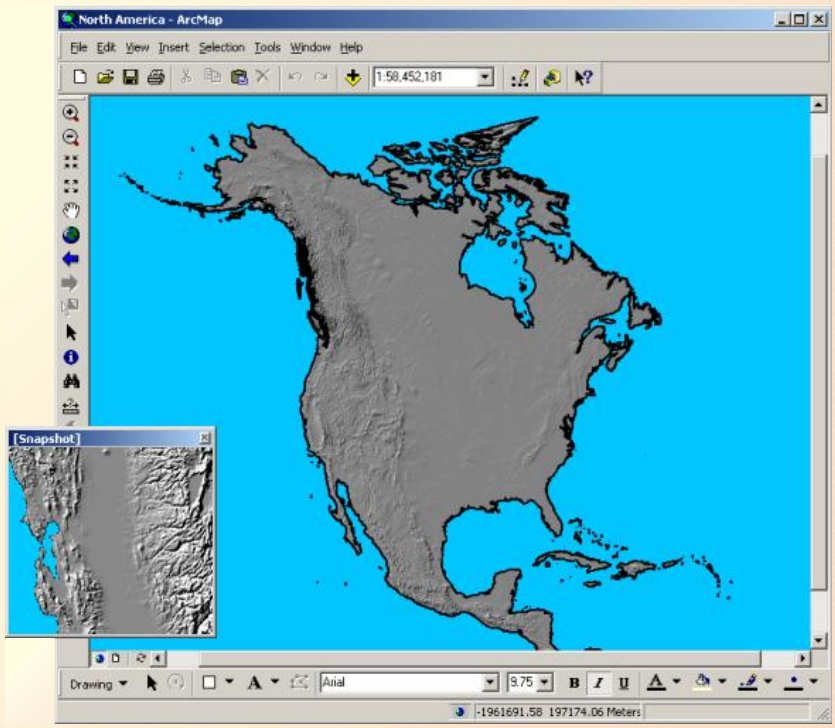


Figure 9.6 Shaded digital elevation model of North America used for comparison of image compression techniques in Table 9.2. Original image is 8726 by 10619 pixels, 8 bits per pixel. The inset shows part of the image at a zoom factor of 1000 for the San Francisco Bay area (Source: ESRI)

Table 9.2 Comparison of file sizes and compression rates for raster compression techniques (using image shown in Figure 9.6)

Compression technique	File size (MB)	Compression rate
Uncompressed original	90.5	
Run-length	17.7	5.1
Wavelet	2.3	39.3

Box 9.1 Continued

Run-length encoding
Run-length encoding is perhaps the simplest compression method, and for this reason it has already been used to introduce the concept of raster data compression (see Section 3.6.2 and Figure 3.11). It involves encoding 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 little like a two-dimensional version of run-length encoding in which the 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 size. It is sometimes described as a *quadtree* data structure.

Wavelet
Wavelet compression techniques invoke principles similar to those discussed in our treatment of fractals, as discussed in Section 5.8. They remove information by recursively examining patterns in datasets at different scales. 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. MrSID (Multiresolution Seamless Image Database) from LizardTech is an example of a wavelet compression technique that is widely used in geographic applications, especially for compressing aerial photographs. Similar wavelet compression algorithms are available from other public and private sources.

Run-length and block encoding both result in lossless compression of raster layers, that is, a layer can be compressed and decompressed without loss of detail. In contrast, the MrSID wavelet compression technique is *lossy* since information is irrevocably discarded during compression. Although MrSID 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.

the grid, the cell size, and the number of row and column elements. The array itself is usually stored as a compressed file or as a record in a database management system (see Section 11.3). Techniques are described in Box 9.1 (see also Section 3.6.2 for the general principles involved).

Data 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.

9.2.3 Vector data model

The raster data model discussed above is most commonly associated with the field conceptual data 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. To date, the vector data model has been very widely implemented in GIS. This is because of the precise nature of its representation method, its storage efficiency, the quality of its cartographic output, and the availability of functional tools for operations like map projection, overlay, and analysis.

In the vector data model each object in the real world is first classified into a geometric type: point, line, or polygon (Figure 9.7). 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, and polygons (e.g. census tracts, soil areas, and oil license areas) as one or more line segments that close to form an 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).

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

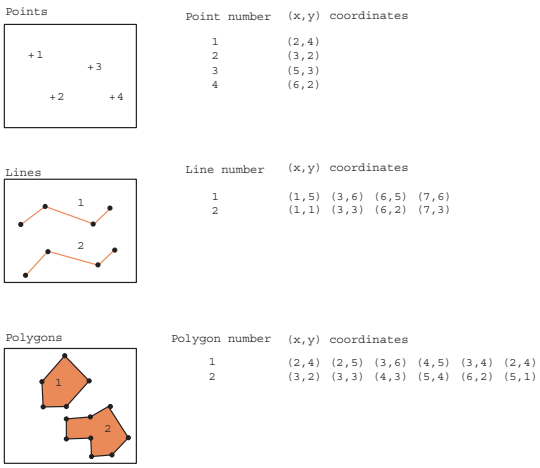


Figure 9.7 Representation of point, line, and polygon objects using the vector data model

particularly useful for repre-senting artificial entities like road curbs and some buildings.

9.2.3.1 Simple features

Geographic entities encoded using the vector data model are often called features and this will be the convention adopted here. Features of the same geometric type are stored in a geographic 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. GIS commonly deal with two types of feature: simple and topologic. The structure of simple feature line and polygon datasets is sometimes called spaghetti because, like a plate of cooked *spaghetti*, lines and polygons (strands of spaghetti) can overlap and there are no relationships between any of the objects.

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

Simple feature datasets are useful in GIS applications because they are easy to create and store, and because they can be retrieved and rendered on screen very quickly. On the other hand the lack of any connectivity relationships between the features means that operations like shortest-path network analysis and polygon adjacency cannot be performed without additional time-consuming calculations. Simple feature polygon layers are also inefficient for modeling phenomena conceptualized as fields because the boundary between adjacent features (for example, two potato fields) must be digitized and stored twice. They are also inflexible, as it is not easy to dissolve common boundaries when amalgamating zones. Furthermore, because simple feature polygons can overlap this limits their use in certain applications. For example, in land ownership applications land parcels cannot overlap, for otherwise two people could own the same piece of land (land ownership is conceptualized as a field that necessarily has one owner at every point)!

9.2.3.2 Topologic features

Topologic features are essentially simple features structured using topologic rules. Topology is the science and mathematics of relationships. In GIS topology is used to validate the geometry of vector datasets (e.g. check that polygons close and that all lines in a network are joined together), and for certain types of operations (e.g. network tracing and tests of polygon adjacency). Topologic structuring of vector layers introduces some interesting and very useful properties, especially for line (also called 1-cell, arc, edge, and link) and polygon (also called 2-cell, area, 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 (see also Section 6.3.3). A node is placed wherever the ends of lines meet. 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 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.

In a topologically structured polygon data layer each polygon is defined as a collection of lines, that in turn are made up of an ordered list of coordinates (vertices). The list of lines that make up a polygon is stored with the geometry data. Figure 9.8 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 polygon-line 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

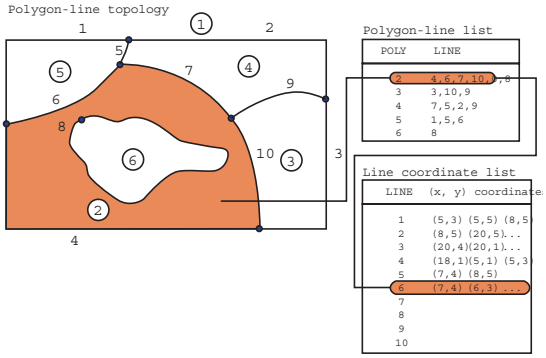


Figure 9.8 A topologically structured polygon data layer. The polygons are made up of the lines shown in the polygon-line list. The lines are made up of the coordinates shown in the line coordinate list (Source: after ESRI 1997)

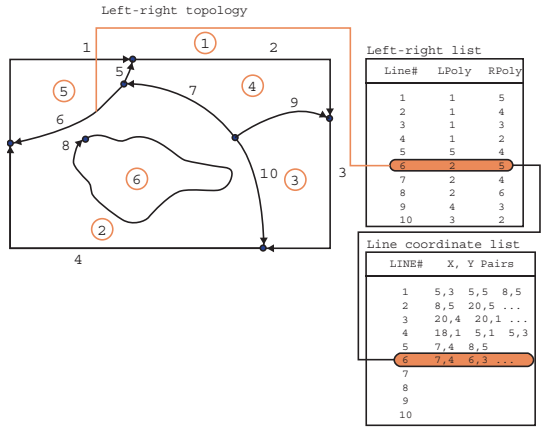


Figure 9.9 The contiguity of a topologically structured polygon data layer. For each line the left and right polygon is stored with the geometry data (Source: after ESRI 1997)

2. The list of coordinates for each line is also shown in Figure 9.8. For example, line 5 begins with coordinates 7,4 and 6,3 — other coordinates have been omitted for brevity. A line may appear in the polygon-line 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 line are only stored once in the line-coordinate 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 lines must be retrieved from the database and the boundary assembled, a process that can be time consuming when repeated for each polygon in a large dataset.

Moving on from the one-dimensional case of lines to the two-dimensional case of polygons requires introduction of the concept of planar enforcement. 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 is conceptualized as a field.

The contiguity (adjacency) relationship between polygons is also defined during the process of topologic structuring. This information is usually stored as a list of the polygons on the left- and right-hand side of each line, in the direction defined by the list of coordinates (Figure 9.9). In Figure 9.9 polygon 2 is on the left of line 6 and polygon 5 is on the right. Thus we can deduce from a simple look-up operation that polygons 2 and 5 are adjacent.

Software systems based on the vector topologic feature data model, also called the geo-relational model, have become popular over the years. The term *geo-relational* derives from the way the topologic feature model is implemented. In this model, the geometry and associated topologic information is stored in regular computer files, whereas the associated attribute information is held in relational database management system (RDBMS) tables. The GIS software 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 RDBMS because, until relatively recently, RDBMS were unable to store and retrieve geographic data efficiently. Figure 9.10 is an

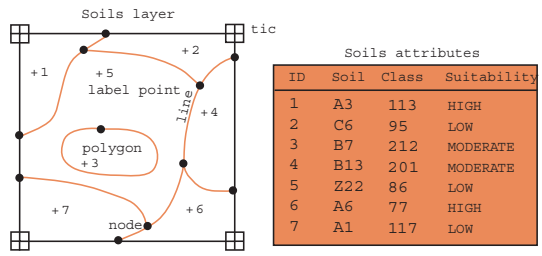


Figure 9.10 An example of a geo-relational polygon dataset. Each of the polygons is linked to a row in an RDBMS table. The table has multiple attributes, one in each column (Source: after ESRI 1997)

example of a geo-relational or coverage polygon dataset showing 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 (see also Chapter 11). Thus in this soils dataset polygon 3 is soil B7, of class 212, and its suitability is moderate.

The topologic feature geographic data model has been extensively used in GIS applications over the last 20 years, especially in government and natural resources applications based on polygon representations (see Sections 2.3.2 and 2.3.4). Typical government applications include: cadastral management, tax assessment, parcel management, 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.

The tax appraisal case study discussed in Section 2.3.2.2 is an example of a GIS based on the topologic feature data model. Because the developers of this system wanted to avoid overlaps and gaps in tax parcels (polygons), to ensure that all parcel boundaries closed (were validated), and that they were stored in an efficient way, they chose this model. This is in spite of the fact that there is an overhead in creating and maintaining parcel topology, as well as a slight degradation in draw and query performance for large databases.

9.2.3.3 Network data model

The network data model is really a special type of topologic feature model. It is discussed here separately because it raises several new issues and has been widely applied in GIS studies.

Networks can be used to model the flow of goods and services. There are two primary types

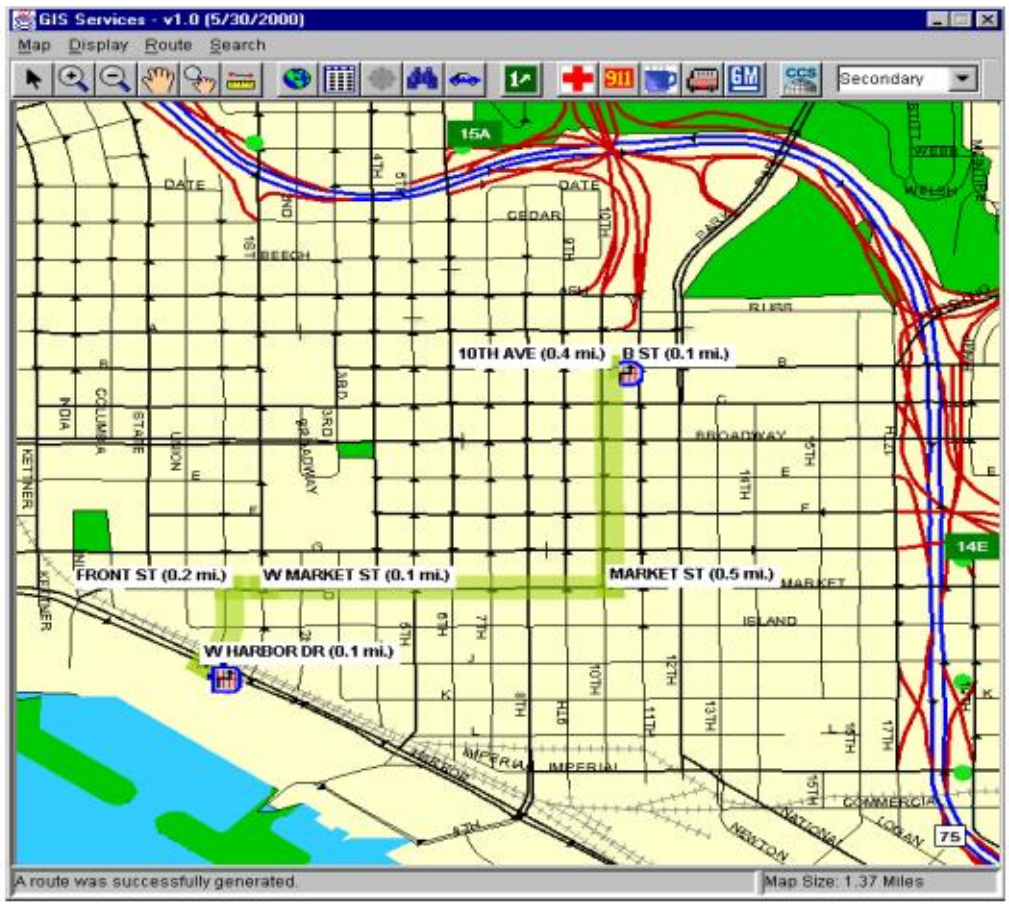


Figure 9.11 An example of a street network

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 GIS software systems 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 topologic models), and lines (for example, streets, transmission lines, pipes, and stream reaches). Network topologic 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 on the nodes and lines. Figure 9.11 shows an example of a street network. The network comprises a collection of nodes (types of street intersection) and lines (types of street). The topologic relationship between the features is maintained in a connectivity table. By consulting information stored in the connectivity table it is possible to trace the flow of traffic through the network and to examine the impact of street closures. The impedance on the intersections and streets determines the speed at which traffic flows. Typically, the rate of flow is proportional to the street speed limit and 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 geo-relational implementations of the topologic feature model, the geometry and topologic information is typically held in ordinary computer files and the attributes in a linked database. The GIS software tools are responsible for creating and maintaining the topologic information each time there is a change in the feature geometry.

There are many applications that 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* (Section 4.4). The basic principle of linear referencing is quite simple. Instead of recording the location of geographic entities as explicit *x*, *y*, *z* coordinates, they are stored 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 pot holes 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 multi-carriageway 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). For further discussion of linear referencing see Section 4.4.

9.2.3.4 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 geologic strata. Strictly speaking, surfaces are said to be 2.5D structures. A true 3D structure will contain multiple *z* values at the same *x*, *y* location and thus is able to support volumetric calculations like 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 railroad construction). For further discussion of 2.5D see Box 3.5. Both grids and triangulated irregular networks (TINs) are used to create and represent surfaces in GIS. A regular grid surface is really a type of raster dataset as discussed earlier in Section 9.2.2. 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 non-overlapping triangular elements (Figure 9.12). A TIN is created from a set of mass points, that is, 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 5.4). TIN surfaces are frequently created by performing what is called a *Delaunay triangulation* of the points to create a series of triangular areas (also called faces) that touch their neighbors at each edge.

A TIN is a topologic data structure that manages information about the nodes that comprise each triangle and the neighbors of each triangle. Figure 9.13 shows the topology of a simple TIN. As with other topologic data structures, information about a TIN may be conveniently stored in a file or database table.

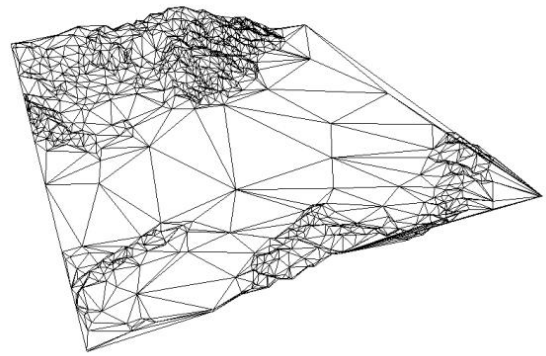
TINs offer many advantages for surface analysis. First, they incorporate the original sample points, providing a useful check on the accuracy of the model. Second, the variable density of triangles means that a TIN is an efficient way of storing surface representations. Third, the data structure makes 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 9.14 shows two example applications of TINs. Figure 9.14(A) is a shaded landslide risk TIN of Pisa district, Italy with building objects draped on top to give a sense of landscape. Figure 9.14(B) is a TIN of the Yangtse River, China greatly exaggerated in the *z* dimension. It shows how TINs draped with images can provide photo-realistic views of landscapes.

Like all 2.5D and 3D models TINs are only as good as the input sample data. They are especially susceptible to extreme high and low values because there is no smoothing of original data. Given that many DEMs are provided as grid data structures, the usage of TINs is not as high as it might be.

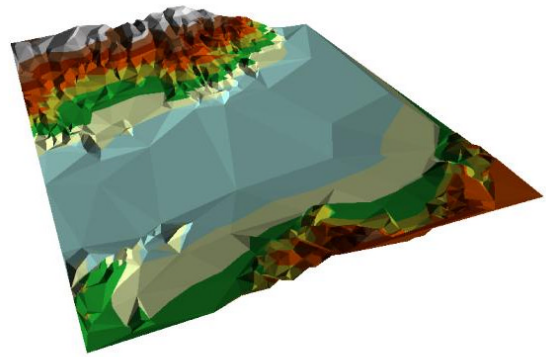
9.2.4 Object data model

All the geographic data models described so far are geometry-centric, that is they model the world as collections of points, lines, and polygons. Any operations to be performed on the geometry (and, in some cases, associated topology) are created as separate procedures (programs or scripts).

(A)



(B)



(C)

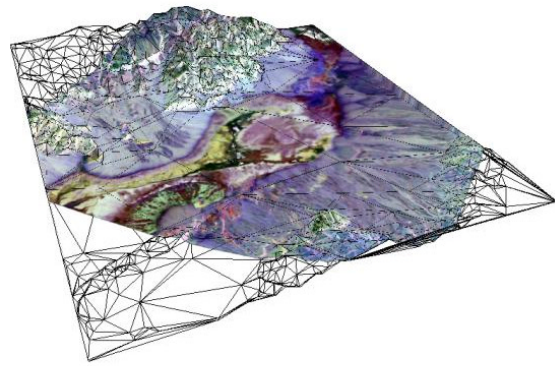
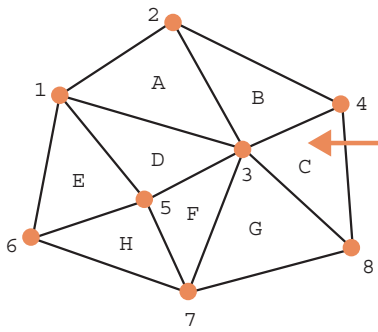


Figure 9.12 TIN surface of Death Valley, California: (A) "wireframe" showing all triangles; (B) shaded by elevation; (C) draped with satellite image

A TIN is a topologic data structure that manages information about the nodes that comprise each triangle and the neighbors to each triangle



Triangle	Node list	Neighbours
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

Triangles always have three nodes and usually have three neighboring triangles. Triangles on the periphery of the TIN can have one or two neighbors.

Figure 9.13 The topology of a TIN (Source: after Zeiler 1999)

Unfortunately, this approach can present several limitations for modeling geographic systems. All but the simplest of geographic systems contain many entities with large numbers of properties, complex relationships, and sophisticated behavior. Modeling such entities as simple point, line, and polygon geometry types is overly simplistic and does not easily support the sophisticated characteristics required for modern analysis. Additionally, 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. These allow the full richness of geographic systems to be modeled in an integrated way in a GIS.

The central focus of an object data model is the collection of geographic objects and the relationships between the objects (see Box 9.3). Each geographic object is an integrated package of geometry, properties, and methods. As such it is a software representation of the extent and function of the *geographic individuals* described in Chapter 5. In the object data model geometry is treated like any other attribute of the object and not as its primary characteristic. Geographic objects of the same type are grouped together as object classes, with individual objects in the class referred to as instances. In modern GIS software systems each object class is stored in the form of a database table, with each row an object and each 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 an object's behavior.

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, a topologic polygon dataset will not have overlapping polygons) while other interclass relationships are user-definable. Three types of relationships are commonly used in geographic object data models: topologic, geographic, and general.

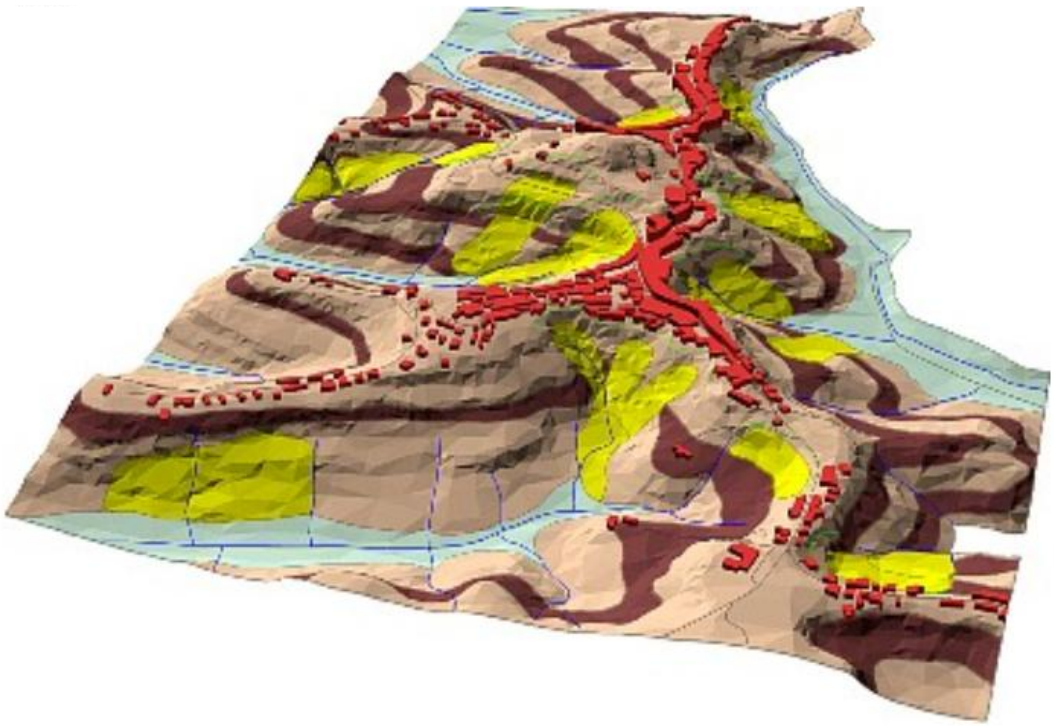
A class is a template for creating objects.

Topologic relationships are generally 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 topologic polygon classes will be structured using the node–line model described in Section 9.2.3.2.

Geographic relationships between object classes are based on geographic operators (such as overlap, adjacency, inside, and touching) that determine the interaction 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.

Geographic relationships are useful to define other types of relationship between objects. In a

(A)



(B)

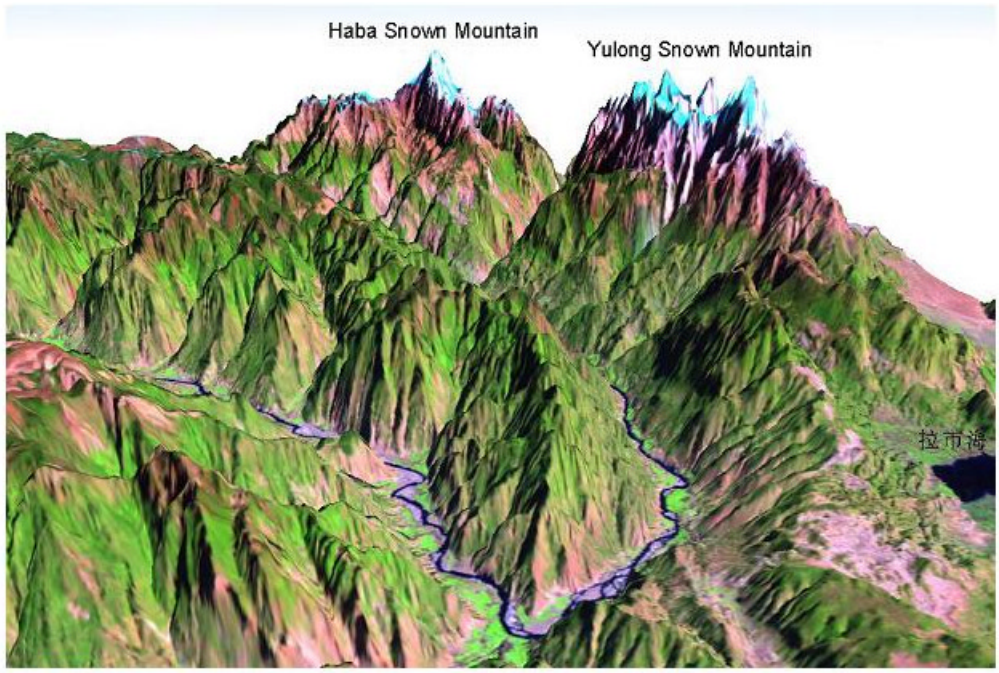


Figure 9.14 Examples of applications that use the TIN data model: (A) Landslide risk map for Pisa, Italy (Courtesy: Earth Science Department, University of Siena, Italy); (B) Yangtse River, China (Courtesy: Human Settlements Research Center, Tsinghua University, China)

Box 9.2 Object-oriented concepts in GIS

An object is a self-contained package of information describing the characteristics and capabilities of an entity under study. 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 GIS is an object. A collection of objects of the same type is called a *class*. In actual fact classes are a more central concept than objects from the implementation point of view. A class can be thought of as a template for objects. When creating an object data model the data model 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's 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 small scale map and a polygon on a large-scale one). The 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 well object might also have methods or behavior defining what it can do. Example behavior might include how to draw itself on a computer screen, how to create and delete itself, and editing rules about how the well snaps to pipelines.

There are three key facets of object data models that 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 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 GIS 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 re-use of objects.

Inheritance is the ability to re-use 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 9.3 below 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 (e.g. gas pipes, valves, and service lines). This mechanism will work for a new object class because the new class is responsible for implementing the object creation method.

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 because they enforce validation constraints. 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: highway traffic speed must be in the range 25–70 miles (40–120 km) per hour; forest compartment average tree height must be in the range 0–50 meters. Coded attribute rules are used for categorical data types. Examples include: land use must be of type commercial, residential, park, or other; or pipe material must be of type steel, copper, lead, or concrete.

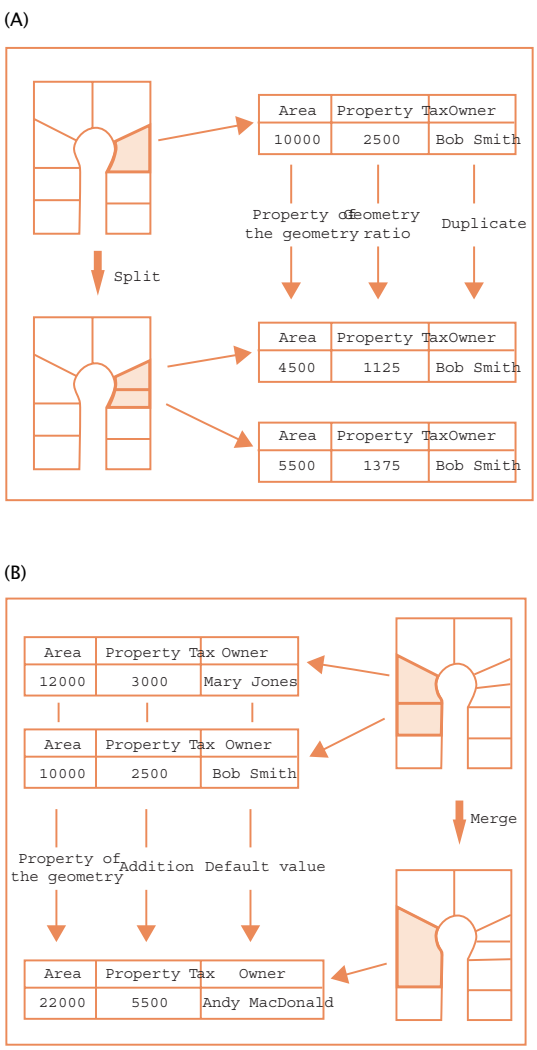


Figure 9.15 Example of split and merge rules for parcel objects: (A) split; (B) merge (Source: after MacDonald 1999)

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).

Geographic rules define what happens to the properties of objects when an editor splits or merges them (Figure 9.15). 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 like 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 like 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.

9.3 Example of a Water Facility Object Data Model

The goal of this section is to describe an example of a geographic object model. It will discuss 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 9.1, is to represent the key aspects of the real world inside the digital computer for management, analysis, and display purposes.

Figure 9.16 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 purpose of the object model is to support asset management, mapping, and network analysis applications. Based on this it is useful to classify the objects into two types: the landbase

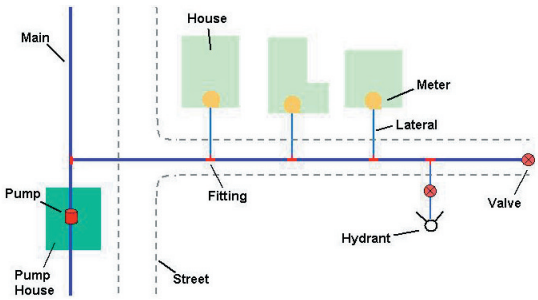


Figure 9.16 Water distribution system water facility object types and geographic relationships

and the water facilities. Landbase is a general term for objects like houses and streets that provide geographic context, but are not used in network analysis. The landbase object types are PumpHouse, 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 of these object types need to be modeled as a network in order to support network analysis operations like network isolation traces and flow prediction. A network isolation trace is used to find all parts of a network that are unconnected (isolated). Using the topologic 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 the network based on network connectivity and data about water availability and consumption. Figure 9.17 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 GIS editor it will be useful to specify topologic connectivity and attribute rules to control how objects can be connected (see Section 9.2.3.3 above). Before this network can be used for analysis it will also be necessary to add flow impedances to each link (for example, pipe diameter).

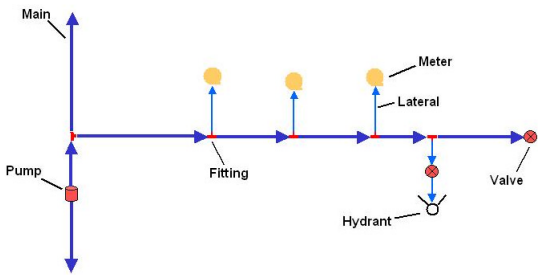


Figure 9.17 Water distribution system network

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 9.18 shows one possible object model that uses 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. UML models are a type of tree structure where each box is an object class and the lines define how one class reuses (inherits) part of the class above it in the tree.

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 exist for efficiency reasons. 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, PumpHouse 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 OperationsRecord object classes have their location determined by being associated with other objects (e.g. valves and 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 to draw and specify a logical model (Figure 9.19). A further advantage of a CASE tool is that physical models can be generated directly from the logical models, including all the database tables and much of the supporting code for implementing behavior (Figure 9.20). Once a database structure (schema) has been created, it can be populated with objects and the intended applications put into operation.

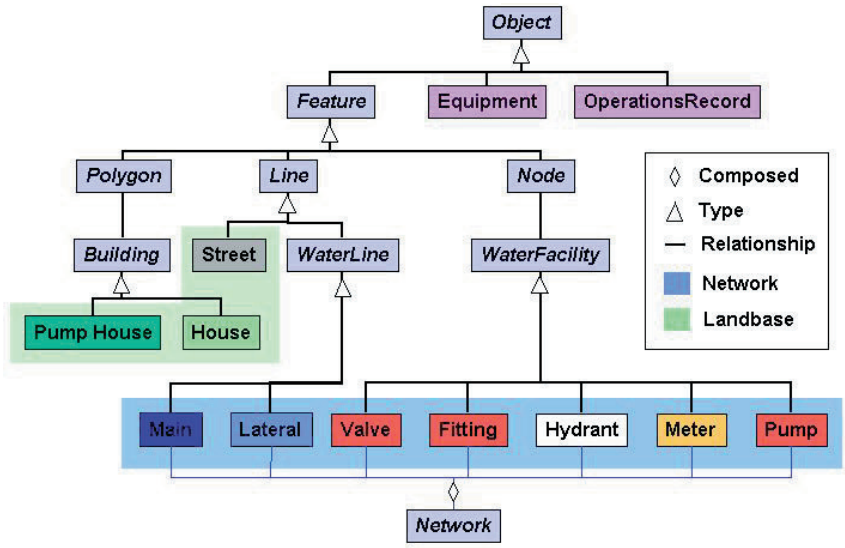


Figure 9.18 A water facility object model

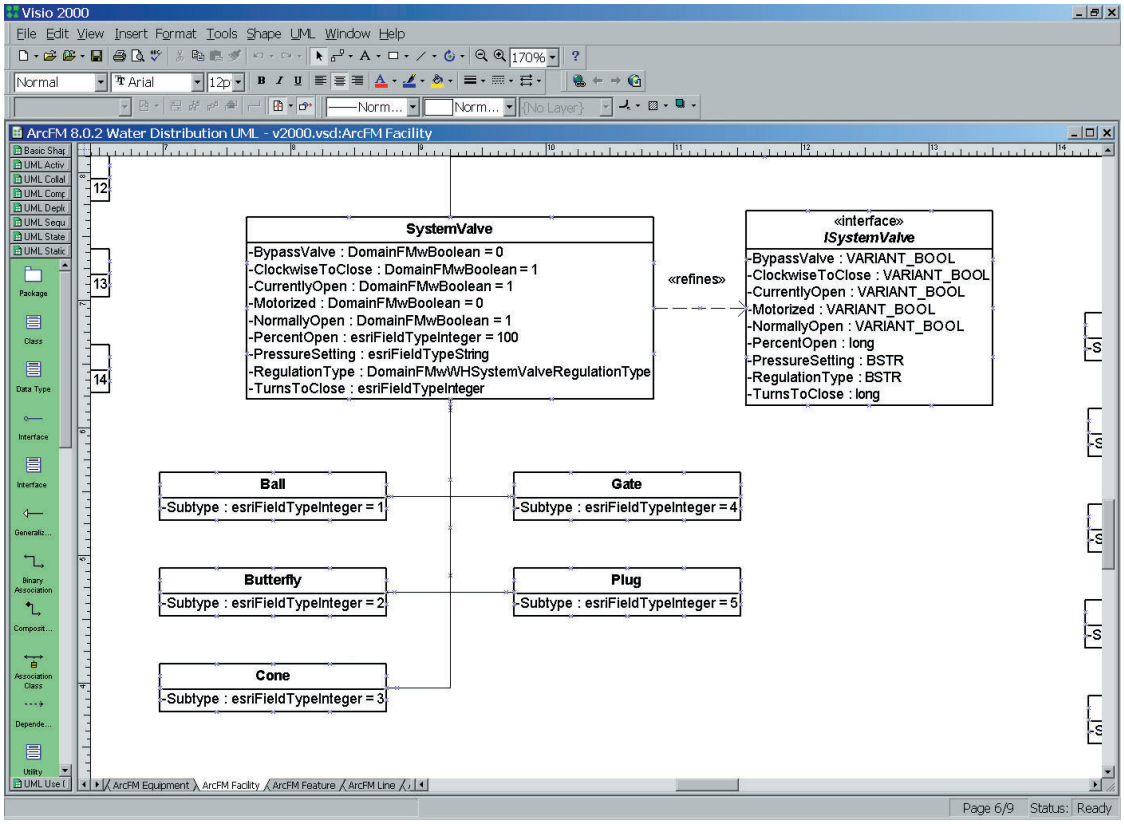


Figure 9.19 An example of a CASE tool (Microsoft Visio). The UML model is for a utility water system

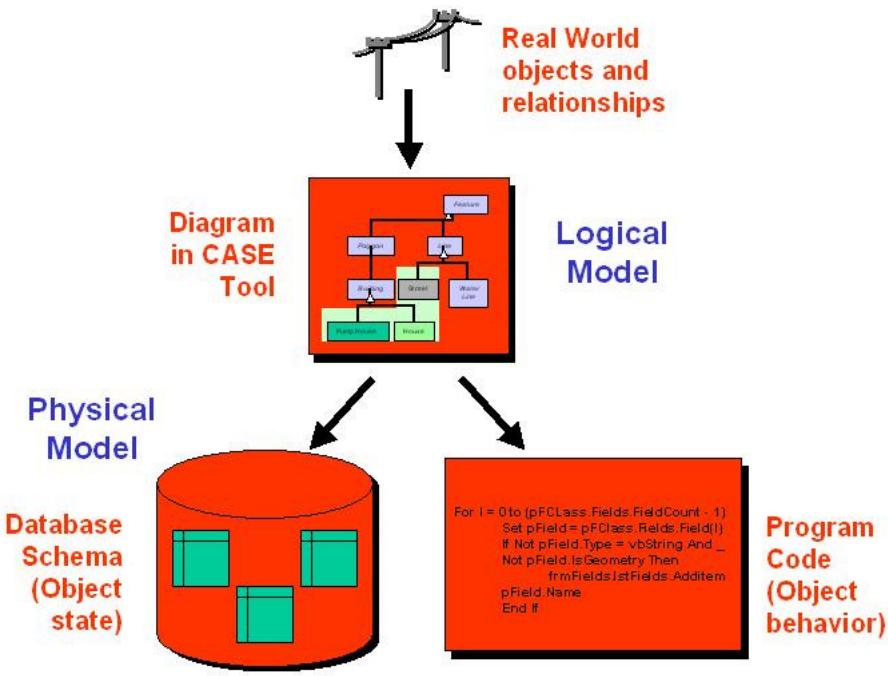


Figure 9.20 Use of a CASE tool in geographic data modeling (Source: ESRI)

9.4 Geographic Data Modeling in Practice

Geographic information science 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.

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 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 the water distribution system object model are shown in Figure 9.18 above.

Once an implementation-independent logical model has been created, this model can be turned into a system-dependent physical model. A physical model 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 sub-optimal 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. 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 very elementary applications such as a map of the body. In a similar vein raster models are good for field-based data such as soils, vegetation, pollution, and population counts. There is further discussion on geographic data modeling in the next chapter.

Questions for Further Study

1. Figure 9.21 is an oblique aerial photograph of part of the city of Kfar-Saba, Israel. Take 10 minutes to list all the object classes (including their attributes and behavior) and the relationships between the classes that you can see in this picture that would be appropriate for a city information system study.
2. Why are there so many GIS data models?
3. Why is it useful to include the conceptual, logical, and physical levels in geographic data modeling?
4. Describe five key differences between the topologic vector and raster geographic data models. It may be useful to consult Figure 9.3 and Chapter 3.
5. Review the terms encapsulation, inheritance, and polymorphism and explain with geographic examples why they make object data models superior for representing geographic systems.

Online Resources

NCGIA Core Curricula (www.ncgia.ucsb.edu/pubs/core.html

Core Curriculum in GIScience, Sections 2.3.1 (Information Organization and Data Structure, Albert Yeung), 2.3.2 (Non-Spatial Database Models, Thomas Meyer), 2.4.1 (Rasters, Michael Goodchild), 2.4.2 (TINs), 2.4.3 (Quadtrees and Scan Orders, Michael Goodchild), 2.6 (Representing Networks, Benjamin Zhan), and 2.9.1 (Transportation Networks, Val Noronha)

Core Curriculum in GIS, 1990, Units 4 (The Raster GIS), 11 (Spatial Objects and Database Models), 12 (Relationships among Spatial Objects), 13 (The Vector or Object GIS), 21 (The Raster/Vector Database Debate), 30 (Storage of Complex Objects), 31 (Efficient Storage of Lines – Chain Codes), 35 (Raster Storage), 36 (Hierarchical Data Structures), 37 (Quadtree Algorithms and Spatial Indexes), 38 (Digital Elevation Models), 39 (The TIN Model),

Figure 9.21 Oblique aerial view of Kfar-Saba, Israel (Courtesy: ESRI)

42 (Temporal and Three-Dimensional Representations), 43 (Database Concepts I), and 44 (Database Concepts II)

ESRI Virtual Campus course on ESRI software (campus.esri.com):
Introduction to ArcView GIS, by ESRI (especially the ''Getting Data into ArcView'' lesson of the Module ''Basics of ArcView'': see also Einführung in ArcView GIS by ESRI and Josef Strobl, and Introdução ao ArcView GIS by ESRI and Mirna Cortopassi Lobo)
Understanding Geographic Data by David DiBiase

General:
www.infogoal.com/dmc/dmcdmd.htm General data modeling resources
www.utexas.edu/cc/database/datamodeling/ Introduction to data modeling
www.fairview-industries.com/intro.htm Example of cadastral data modeling
www.rational.com/university/index.jsp Rational University – e-University from a commercial vendor
www.innovativegis.com/education/primer/concepts.html Introduction to GIS data models

Reference Links

Maguire D J, Goodchild M F, and Rhind D W (eds) 1991 *Geographical Information Systems: Principles and applications*. Harlow, UK: Longman (Text available online from 'Links to Big Book 1' at www.wiley.com/gis and www.wiley.co.uk/gis).
Chapter 16, High-level spatial data structures for GIS (Egenhofer M J, Herring J)
Chapter 19, Digital terrain modeling (Weibel R, Heller M)

Longley P A, Goodchild M F, Maguire D J, and Rhind D W (eds) 1999 *Geographical Information Systems: Principles, techniques, management and applications*. New York: John Wiley.
Chapter 25, GIS customization (Maguire, D J)
Chapter 26, Relational databases and beyond (Warboys M F)
Chapter 29, Principles of spatial database design and analysis (Bédard, Y)
Chapter 36: Spatial tessellations (Boots, B)

References

ESRI 1997 *Understanding GIS: the ArcInfo Method*. Redlands, California: ESRI Press.

ESRI 2000 *Inside ArcFM Water.* Redlands, California: ESRI Press.

Heywood I, Cornelius S, and Carver S 1998 *An Introduction to Geographical Information Systems*. Harlow: Longman.

MacDonald A 1999 *Building a Geodatabase*. Redlands, California: ESRI Press.

Worboys M F 1995 *GIS: A computing perspective*. London: Taylor and Francis.

Zeiler M 1999 *Modeling Our World: The ESRI guide to geodatabase design*. Redlands, California: ESRI Press.