9

Attribute Data Structures

9.1 SEQUENCE OF DEVELOPMENT

In the preceding chapters we examined physical data structures, the actual arrangement of digital cartographic objects in computer files. We also examined logical data structures, the conceptual links between digital cartographic objects and their physical data structures. Since the advent of geographic information systems (GISs), much attention has been devoted to how effective the physical-to-logical data structure link is, especially when using these structures for interactive query and analysis. These additional functions (additional, that is, to computer cartographic systems) require the ability to query thematic data using geographic properties. More strictly we query the cartographic objects representing the geographic phenomena by their attributes reflecting geographic properties. In addition, these queries require the retrieval of data from many different storage locations, usually many files with different sizes and formats.

Advanced analytical operations, the sort supported in GISs, require the management of attribute data to support query and analysis by cartographic data structures. The map and the attribute data physically reside in separate files or are separated within the computer's memory. The problem of managing these files is one of database management, a field well represented within computer science. If we look at the sequence of how database management systems developed, we find that there are distinctive trends in the historic development of attribute data structures. Although these trends are more important to GISs than to computer cartography, they are of relevance to analytical cartography and therefore will be discussed here.

The first generation of GISs and computer cartographic systems did not really support query and analysis interactively. These early systems had sophisticated map data structures, but were usually grid or entity-by-entity based, and often included the thematic data as part of the cartographic data structure. Because the purpose of these systems was to produce a map, the independence of the data was sacrificed. For example, to convert choropleth data from numbers to percentages would usually involve reentering the thematic data, perhaps as a set of parameters on "cards" for batch processing. The multifile problem was avoided by merging the data into a single file containing the data structure.

Second-generation systems were still entity-by-entity structured, but they were topological systems. Using these systems, we could expand the basic entity of a point to build lines and then lines to build polygons. Polygons in a topological system could consist of a chain list or a point list, but we also have as part of the data structure information about which chains are linked to which other chains. By keeping track of forward and reverse linkages so that we can reassemble our polygons using the topology, if we need to fill them or verify them, we are actually storing attribute data as raw geographic properties, in particular connectivity and contiguity. If we need to know which lines are connected to other lines, we can find that out from our data structure.

In these systems, different parts of the structure could be stored in different files. As a result, the data structure is more flexible. The thematic data, however, often remained as part of the map information. For example, the DIME files consisted of single sets of records, with index numbers for the polygons. The user of the DIME files had to manually merge the census variables with the map data to produce computer-generated maps. The only physical link between the topological and the attribute data was the census tract or block number, which happened to be the same in each of two independent data sets. Although this arrangement was closer to supporting query and analysis, the data structure itself was not the means by which query was performed.

Third-generation systems combine map and attribute data together, usually within a GIS. Third-generation systems are distinctive because they are capable of performing separate data management, that is, performing transformations on the thematic data directly without reference to the geography and across multiple files. These systems have been able to integrate the capabilities of database management systems, such as selective retrieval, fast sorting, and operations on multiple data attributes, into systems designed for the display and analysis of geographic data. These systems use the concept of the data model and implement one of three basic models common in database theory.

The first two of these data models, the hierarchical and network models, have now been largely superceded, but are important historically and so are considered in section 9.2. The third of these models, the relational model, is itself a movement to a fourth stage of attribute data structure development. As we will see, it is this data model that has allowed the development of GISs capable of performing real-time query and analysis by geographic properties and their pertinent cartographic objects. It is, therefore, worth examining the basic data models before returning to an examination of some current attribute data structures found in contemporary GISs.

9.2 ATTRIBUTE DATA MODELS

A data model can be defined at any one of three levels. The data to be modeled correspond to the geographic phenomena or entities as they exist in the real world. Because they are data, they correspond to a set of measurements of real-world phenomena. The three levels at which data models have been designed are (1) the conceptual level, at which level the model is independent of the specific systems or program data structures used to organize or manage the data; (2) the logical level, which corresponds to the data structure, or logical organization of the components of the model and in which the relationship between components are explicitly defined; and (3) the physical level, at which a set of rules exist which specify how the machine implements the data structure within a computing environment (Peuquet, 1984).

This division corresponds closely to the terms entity, object, and data structure used in this text, although the data structure has been shown to contain both a logical (design) and a physical (file or programming language) structure of its own. The latter division is largely artificial, to understand the data structures in use for spatial data it is necessary to implement them, or at least understand their implementations. For attribute models, however, it is less critical to understand the physical implementation. The discussion here, therefore, is restricted to the conceptual data model level in Peuquet's hierarchy.

9.2.1 The Hierarchical Data Model

As discussed previously, in the management of attribute data, the equivalent of the term *cartographic data structure* is the term *data model.* Data models were developed from database management theory, part of computer science. The first and simplest of the data models is the hierarchical model. The hierarchical way to structure data can refer to many different attributes, not just those of geographic data. Hierarchical data management systems form the basis of many commercially available systems and have influenced file structures and our thinking about organizing data generally.

To use a geographical example, though, the top of a hierarchy is called the *root* (from the root of a tree) and for geography the root could be the WORLD (Figure 9.1). We know that there are seven CONTINENTS, and one of those continents is North America, which consists of several COUNTRIES, one of which is the United States of America. The United States consists of 50 STATES, one of which is New York, and New York consists of COUNTIES, one of which is New York (same name, but different level in the hierarchy). New York County, otherwise known as Manhattan, contains DISTRICTS, one of which is the Upper East Side, where we can find the LOCATION of Hunter College.

This is a hierarchical structure. Each piece of the structure has links to "children" and a single "parent" through which all searching must pass. A full reference to a particular element, such as a cartographic entity like Hunter College, a unit upon which we are going to hang data, gives its path down the "tree." An atlas is usually structured this way, with all maps for Asia bound together and separated country by country. At the lowest level of the tree, if we think of the tree as inverted, we find the "leaves" or records. For cartographic data the records are most commonly cartographic objects, and we assume that the map database contains graphic primitives associated with the cartographic objects. In addition, normally for each entity we have attribute information.

Attributes can store nominal, ordinal, interval, or ratio data, as well as text or complex characteristics. Attributes and entities have both a physical and a logical structure. In database management systems, the physical structure is often hidden. Logically, the attributes and entities can be thought of as occupying a flat file.

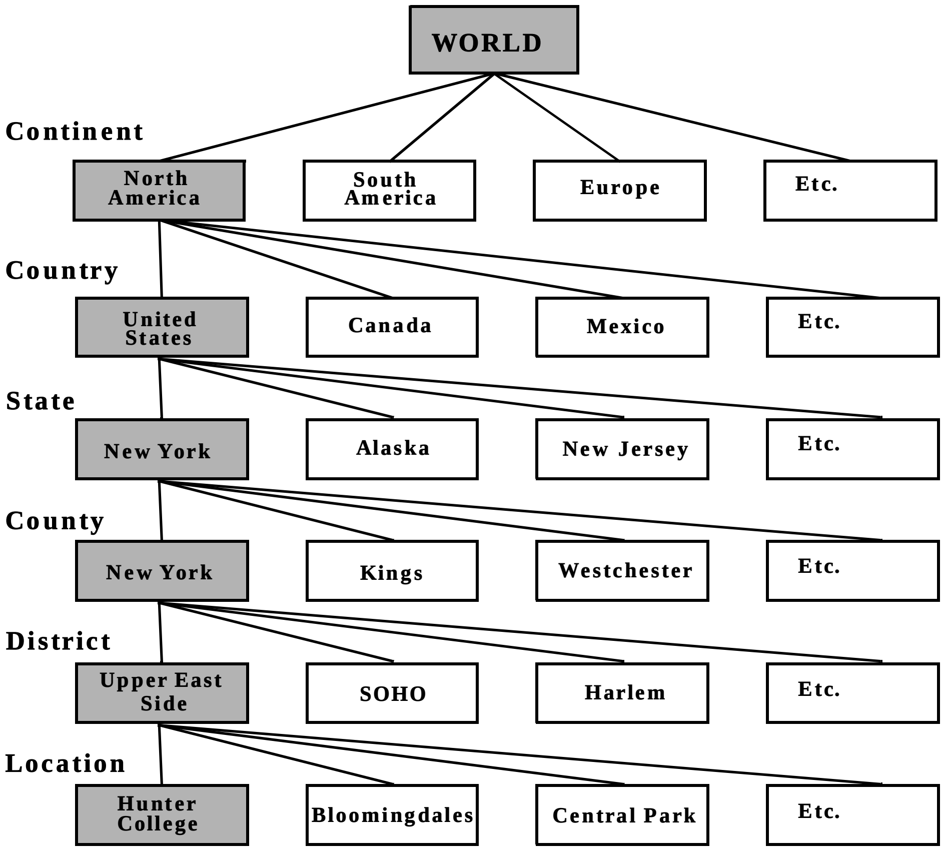

Figure 9.1 A geographic hierarchy.

A flat file (Figure 9.2) resembles a spread-sheet table or matrix, with rows of records and columns of attributes. A flat file corresponds to a particular level of the hierarchy. For example, a flat file for New York State could contain counties as records, with each county having another flat file further down the hierarchy. The census distribution tapes and the city and county data book are good examples of flat files. Usually, all the communities are listed as rows, and hundreds and hundreds of data values for different attributes are listed as columns.

In the hierarchical data model, we organize the links between the logical and the physical aspects of the data in a tree structure. Although this system is often adequate, several problems can result. A major problem arises from cartographic objects that belong to multiple hierarchies. For example, police districts, school districts, census tracts, voting districts, and congressional districts rarely coincide, all being part of different hierarchies. Yet a single cartographic entity falls into all these areas. Also, hierarchies are

Attributes

Entities

Mountain	Elevation	Country	Second Country	Etc
Everest	8847	Nepal	China	
K2	8611	China	Pakistan	
Annapurna	8078	Nepal	——	
Dhaulagiri	8172	Nepal	——	
Gasherbrum	8068	China	Pakistan	
Kanchenjunga	8598	India	Nepal	
Makalu	8481	China	Nepal	
Nanga Parbat	8123	Pakistan	——	
Xixabangma Feng	8012	China	——	

Figure 9.2 A flat file.

rarely spatially mutually exclusive, leaving gaps in some areas and double covering other areas. For example, the boundary between New York and New Jersey is marked at the halfway point on the George Washington Bridge. If the bridge is a single cartographic object in a cartographic database, which state is it included under? Also, while cities are usually parts of counties, New York City contains five counties, an inversion of the usual direction of the hierarchy.

A simple geographic type of query of a hierarchically structured attribute data base might be: How many people live in Manhattan community district number 4? Does it cross any census tract boundaries? An example of hierarchical cartographic information is contour lines on a map. The 100-meter contour line presumably contains all points with elevations greater than 100 meters, unless annotated as a closed depression. Hierarchical space partitioning may therefore be suitable for map data but unsuitable for all but the most rigidly encoded attribute data. The method also involves duplication and therefore is not highly storage efficient.

9.2.2 The Network Data Model

The second data model that has been used in database management theory is the network model. In the network model, we allow relationships between entities. Normally, these are not complex relationships. For example, we might allow an entity, a particular block in Manhattan, to belong to a certain congressional district, a census tract, and a school district. Here, the entity is a block and the relationship is a link to an area.

In computer science, a particularly powerful data structure is the linked list. In a linked list, each member of a list contains a pointer to the next member of the list. We saw in Chapter 5, for example, in the USGS's GIRAS files, that the polygon files simply contain sequential pointers to records in the arc files, which in turn contain pointers to the coordinate files. This is the linked list concept, and this concept is behind the network structure.

Links between objects in this structure form networks. Geographically, the pointers carry associations such as ". . . is a member of." Using the parent-child analog above, network systems allow children to have multiple parents, and even allow a child to be its own grandparent. Thus we can consider the hierarchical to be a subset of the network model.

We remove much of the structure of the hierarchy altogether and let our most common, or the highest resolution entity, become the basis for our model. Although this is true geographically, in a attribute context it does not matter if we do not know the exact extent of a particular area, as long as we store pointers from that entity to everything that incorporates it. This model has won a fair degree of acceptance as a data structure for geographic information, although the query ability of these systems may be as limited as that of hierarchical systems (Aronson, 1987).

9.2.3 The Relational Data Model

Returning to the context mentioned above, of levels of development of data management within cartographic and geographic analytical systems, we have now reached the fourth-generation system. In some fourth-generation systems, the data model is termed the *relational model*. The basic unit within a relational system is the table. Within each table we store the name of an entity, the type of that entity, and the attributes associated with that entity. Each table closely resembles the flat file mentioned previously.

One of the attributes associated with a cartographic object, a line, for example, would be the points it contains. Within a relational structure, the points would be stored sequentially in a separate file, because normally relational systems do not use ordering of records. The link between the files is by a *key* or index. The index could be a line identifier. So, for example, in a DLG file we could have a particular line representing a part of an interstate highway between two junctions. On the map, the line would be symbolized as a red double line with a black border, a fact that could be stored along with the line attributes.

The attributes of the line, however, need not be stored with the line. The line could be numbered with a key, and the corresponding key link to another file could contain the attributes of the line, for example, that it is an interstate highway, the type of road surface, the volume of traffic, and so forth. An additional attribute may also be a further index, for example, links (keys) to a file of areas, such as counties, through which the road passes. At this level, links can be established freely between any of the objects, and only the minimum number of files is required to store the data.

A graphic data structure with relational characteristics is very versatile. We can come into this file structure anywhere, at any point, and retrieve useful information. For the interstate highway, we may have a single arc record and wish to make a query at the arc level. One thing we might be interested in is what arcs meet this particular arc at the end of this segment. A route-finding application, for example, may need to provide in-vehicle instructions based on the connecting highways. All we need do is follow the segment along to the end, sort the arc attribute file, and from that retrieve the polygon identifiers that contain these arcs, and so find the polygons that include a particular location. Thus the in-vehicle system could notify the driver as she or he passes from town to town, city to city, or county to county.

The relational structure involves some duplication of information between data files. Also, we end up doing a tremendous amount of indexing. The common element within relational structures is the key, sometimes called the index, which is a unique identifier shared between two files. In relational database management systems complete software systems take care of retrieval of records based on their attributes and even contain systems called data definition languages for setting up new tables and entering data into the tables. Many such systems are now commercially available, and several have been used effectively for geographic data. Relational database management systems also minimize data volume by a process called weeding. As we build new files, and it is very easy to build files out of existing files in relational systems, the system automatically looks for duplication of keys within a file and eliminates redundant records.

Query within a relational system allows some very powerful and flexible searches such as the joining of data sets to provide temporary sets useful for a particular query. Aronson (1987) noted that the relational structure is most amenable to geographic data because of its simplicity, flexibility, efficiency of storage, and its nonprocedural nature, that is, its free-form query. The relational data model emerged as “the dominant commercial data management tool of the eighties” (Aronson, 1987).

9.3 SOME ATTRIBUTE DATA MODELS AND STRUCTURES

9.3.1 The Hypergraph Structure

The way the relational data model has found its way into GISs is generally in its extended form as the *entity-relationship model,* which is now the basis of several successful GIS systems, such as the ARC/INFO system (Morehouse, 1985). To some degree, the entity-relationship concept is similar to an idea by Francois Bouille, a French computer scientist who designed the hypergraph approach and successfully introduced the idea into the English language literature on GIS. The approach suggested that a body of theory from computer science and relational database management theory could be linked by two mathematical subfields, set theory and topology, then applied to geographical data.

Bouille proposed in 1978 the four fundamental geographic concepts of objects, class, attributes, and relations (Bouille, 1978). We can think of these as cartographic objects, dimension, attributes, and linkages. In addition, Bouille used abstract data types: object class, attribute of object, attribute of class, relationships between objects, and relationships between classes. Under this system, it was possible to develop an *n*-dimensional diagram that explicitly stated all the possible links between the objects. Its expression in topological space was a concept borrowed from the mathematical field of topology, called a hypergraph.

A *hypergraph* is a network consisting of nodes and edges. The hypergraph provided the mathematical basis for making relations link objects, classes, and their attributes. The entity-relationship model uses similar ideas, as seen below. There are examples of implementations of the hypergraph structure, to road networks, for example (Rugg, 1983). The idea led to considerable activity in publishing and research in the area of relational database modeling for spatial data.

9.3.2 The Entity-Relationship Structure

Nyerges introduced into GIS and cartography a model developed by Chen (1976), which Chen termed the *entity-relationship model* (Nyerges, 1980). A very similar model was proposed in the same year by the computer scientists Shapiro and Haralick (1980) and placed into actual use by 1982. This represents a very short gap between research and implementation of new systems, a sign of a vigorous research frontier.

Chen noted that his model was more powerful than the simple relational model and that it had all the capabilities of both the network and the relational models. The entity-relationship model has become the standard for many recently developed GISs. In many respects, this is the "single super-flexible structure forming the basis of geographic data management, display and analysis" (Clarke, 1986). An extension of the entity-relationship design is the relational model with non–fixed-length sets of attributes. In computer science, such data management has come to be called object-oriented, and a number of systems use the approach. So far, only a few GISs use this method, although the concept is pertinent to many previous designs.

9.3.3 Object-Oriented Models

A considerable amount of interest has been devoted recently to the application of the tools of object-oriented computer programming (OOP) to data bases. Object-oriented programs use extensions of the programming language to support *classes.* A class is somewhat like a structure, that is, it can contain multiple pieces of information about a single object. Data elements that fit into classes are called *objects,* hence the name object-oriented. A class specification within OOP represents a sort of template, a generic set of attributes and features that an object can possess.

More sophisticated capabilities are supported by OOP systems, however. A class of objects can have a set of attributes, measures, algorithms, or operations that apply only if the object meets its formal specifications. For example, to compute an intersection point between two line segments, we must have two line objects that are not parallel or coincident, and that meet at a point. If these conditions are met, the "rules" or the algorithm for line intersection can be thought of as "applying" to that object. Passing a template from one object to another, therefore, passes on all the rules also. This is called *inheritance* (Figure 9.3).

Thus complex objects can be made from simpler objects by inheriting their specifications, or new instances of objects can be created by copying the template from another object. The particular set of measures, conditions, and rules that apply to an object can be stored, manipulated, and even passed along with the object to other parts of the program,

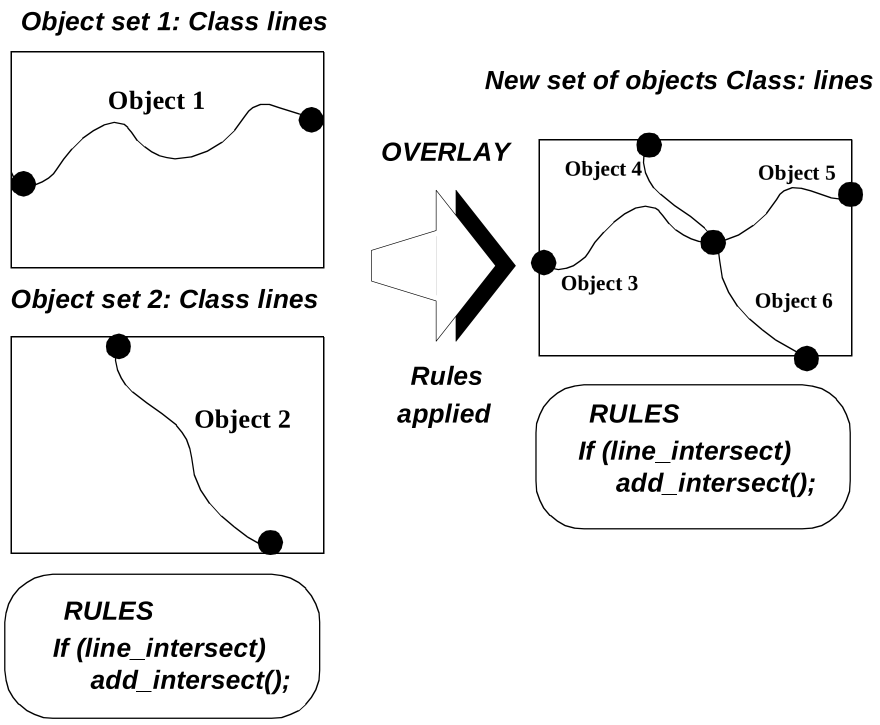

Figure 9.3 Lines as objects showing inheritance.

a condition known as *encapsulation*. The user can control how much of the encapsulated data need be bundled with the object and never revealed and how much needs to move along when the object is modified or used elsewhere in the data base.

A particular object, placed into the template is termed an *instance* of that object. In Figure 9.3, two instances of an object of the class line exist and are encapsulated with their set of intersection rules. An overlay operation then creates a new set of instances of the class line, generated by breaking the objects according to the rules. Each of the new objects inherits the class line and the rules associated with it.

Once a full set of generic objects is defined, each object can simply be treated as one data entry in a very simple data base. The use of object-oriented databases, therefore, allows a whole range of checking, automatic recomputation, reordering and so forth that can be *hidden* from the user. The user of a database, for example, need not know that a set of lines and polygons stored in the data base “know” how to insert new segment points when intersection occurs in the overlay process. The user can simply perform the database operation to do the overlay, without being concerned about each step along the way.

The costs of using object-oriented systems are twofold. First, before beginning, the user must spend time building a full set of objects and relations between objects for use with the data. This is very difficult, and few cartographers to date have attempted this other than in specific contexts, despite that spatial objects and their relations are apparently well defined and understood. In addition, the full overhead of a full set of operations and relations is attached to every data capsule, generating the need for powerful systems to implement the methods. On the other hand, once these object specifications are built, they can be reused again and again for different projects, data, and so forth.

A new generation of software is now appearing that uses object-oriented programming and object-oriented databases for cartographic purposes. A distinct advantage is the full encapsulation of all the data and rules about objects such as map features, giving considerable analytical flexibility to these systems. The systems need no longer make distinctions between graphics, attributes, and algorithms as far as cartographic data are concerned. The impact of this fact, that software and algorithms can become as generic as the data structures used for map data, will be quite liberating for analytical and computer cartography. The powerful extensions to the C programming language in C++ have become the major tools with which these systems are being constructed.

9.3.4 Hybrid Structures

Roger Tomlinson in 1978 argued that geographic data will never conform to the computer science data models and that we should not try to force geographic data into structures that were designed for handling attribute data. Tomlinson noted that in mapping and analysis we have two different goals: In mapping, all we really need to do is retrieve the entities and the geographic locational attributes associated with those entities. In analysis we very often want to retrieve both the entities and the relationships between the entities. Tomlinson said that it might be more appropriate to develop entirely new data models from a geographic viewpoint. A “geographic” data model has yet to be developed.

As far as cartography is concerned, such a model may never need to exist. Only in GISs do we need to embed more information than we need to depict graphically a map as a set of symbols, and to support cartographic analysis. A “geographic data model” may not be the answer as far as GIS is concerned because no matter how complex the data structure, there are always queries that reveal weakness in the data structure for a particular application.

An alternative is to look at some other approaches that have less flexibility. One approach has been to propose systems with “artificial intelligence,” or at least with “expert knowledge,” which interact in a more effective manner with the GIS user and structure the data accordingly (for a survey, see Robinson and Frank, 1987).

Finally, we could use the existing approach, which seems to be to build a new or modified data structure for each new application. Although the off-the-shelf nature of database management systems has made this increasingly easy over the last few years, it is the nature of the map to attribute data link that is of most importance. Systems that support real geographic query allow the user to interact with the data through the map rather than though a menu or command-line type of interface. A cartographic “window” into the data is essential, as is the ability to view and manipulate cartographic symbols while managing the data.

Clearly, attribute data structures are essential to GISs. On a continuum, they are also vital to analytical cartography, where the management of attributes is critical, but less so for computer cartography, where it is sufficient simply to retrieve the correct attribute. It is appropriate, therefore, to conclude this part on the representation of cartographic data by examining the utility of the attribute data models for analytical and computer cartography.

9.4 DATA STRUCTURES AND MODELS FOR ANALYTICAL AND COMPUTER CARTOGRAPHY

Much of the previous discussion has been appropriate for GISs. What about analytical and computer cartography? Automated cartographic systems usually need only to retrieve entities and display them. The most important attributes are locational, and the need for complex statistical operations is minimal. For many map applications, we would be quite happy with a very simple attribute data structure as far as symbolization is concerned. Because more sophisticated cartographic applications such as dynamic maps and vehicle navigation systems require some kind of attribute structure, the two have to be irreversibly linked. Much of the structure of a map is almost totally independent of the locational attributes that a map portrays.

Analytical cartography, and also computer cartography, should concern itself with the management of attribute as well as map data. Although the map is the primary vehicle for the communication of geographic properties and distributions, it is the effectiveness of the analytical operation, the combination of data structure and cartographic transformation, that increasingly influences which map the map reader has access to. A system that produces the wrong map, or a suboptimal map, for the task at hand may be as poor a piece of cartographic software as that which uses poor design or inappropriate symbols.

The power of both map and attribute data structures, as far as analytical and computer cartography is concerned, is in how readily they support cartographic transformations, a theme discussed in Part III. For many reasons, cartographic transformations, such as particular symbolization methods, require data to be in a specific data structure. In this case, we can add to the power of our structure by choosing those that allow us to transform between structures with minimal error and data loss, or at least error of a known magnitude and distribution.

Chapter 10 introduces the cartographic transformation, and Chapter 11 extends the idea to cover the four major mapping transformations. A particularly important chapter, given the previous discussion, is Chapter 11, in which transformations between structures are considered in more detail.

9.5 REFERENCES

Aronson, P. (1987). "Attribute Handling for Geographic Information Systems." *Proceedings, AUTOCARTO 8,* Eighth International Symposium on Computer-Assisted Cartography, Baltimore, March 29-April 3, pp. 346–355.

Bouille, F. (1978). "Structuring Cartographic Data and Spatial Processes with the Hypergraph-based Data Structure," in *First International Symposium on Topological Data Structures for GIS,* edited by G. Dutton. Cambridge, MA: Harvard University. Laboratory for Computer Graphics and Spatial Analysis.

Chen, P. P.-S. (1976). “The Entity-Relationship Model—Toward a Unified View of Data.” *ACM Transactions on Database Systems,* vol. 1, no. 1, pp. 9–36.

Clarke, K. C. (1986). “Recent Trends in Geographic Information System Research.” *Geo-Processing,* vol. 3, pp. 1–15.

Morehouse, S. (1985). “ARC/INFO: A Geo-relational Model for Spatial Information”, *Proceedings, AUTOCARTO 7,* Seventh International Symposium on Computer-Assisted Cartography, Washington, DC, March 11–14, pp. 388–397.

Nyerges, T. L. (1980). “Representing Spatial Properties in Cartographic Databases”, *Proceedings, ACSM Technical Meeting,* St. Louis, pp. 29–41.

Peuquet, D. J. (1984). “A Conceptual Framework and Comparison of Spatial Data Models.” *Cartographica,* vol. 21, no. 4, pp. 66–113.

Robinson, V. B., and A. Frank (1987). “Expert Systems Applied to Problems in Geographic Information Systems: Introduction, Review, and Prospects.” *Proceedings, AUTOCARTO 8,* Eighth International Symposium on Computer-Assisted Cartography, Baltimore, March 29–April 3, pp. 510–519.

Rugg, R. D. (1983). “Building a Hypergraph-based Data Structure: The Example of Census Geography and the Road System.” *Proceedings, AUTOCARTO 6,* Sixth International Symposium on Computer-Assisted Cartography, Ottawa, October 16–21, vol. 2, pp. 211–220.

Shapiro L. G., and R. M. Haralick (1980). “A Spatial Data Structure.” *Geo-Processing,* vol. 1, no. 3, pp. 313–338.

Tomlinson, R. F. (1978). “Difficulties Iinherent in Organizing Earth Data in a Storage Form Suitable for Query.” *Proceedings, AUTOCARTO 3,* Third International Symposium on Computer-Assisted Cartography, San Francisco, January 16–20, pp. 181–201.