

CHAPTER 5

What Is Where?

- 5.1 BASIC DATABASE MANAGEMENT
- 5.2 SEARCHES BY ATTRIBUTE
- 5.3 SEARCHES BY GEOGRAPHY
- 5.4 THE QUERY INTERFACE
- 5.5 STUDY GUIDE
- 5.6 EXERCISES
- 5.7 REFERENCES
- 5.8 KEY TERMS AND DEFINITIONS

*If you don't know where you're going,
you'll end up somewhere else.*

—Yogi Berra.

*Of all the gin joints in all the towns in
all the world, she walks into mine.*

—Julius J. Epstein, *Casablanca*
(1942 film)

*All military commanders ... should
thoroughly store up [in their minds]
the location of ways in and out of
the terrain.... This is the constant value
of maps.*

—Di Tu, *On Maps*, 227 B.C., from
the *Guanzi*, Political, Economic and
Philosophical Essays from Early China,
translation by W. Allyn Rickett.



Philosophical GIS: Not the most respected subfield of geography.

5.1 BASIC DATABASE MANAGEMENT

A GIS can answer the two questions: “what?” and “where?” More important, a GIS answers the question “What is where?” The *where* component relates to the map behind all GIS activities. The *what* relates to the features, their size, geographical properties and,

above all else, their attributes. Getting this information is what the toolbox definition of GIS in Chapter 1 meant by *retrieval*.

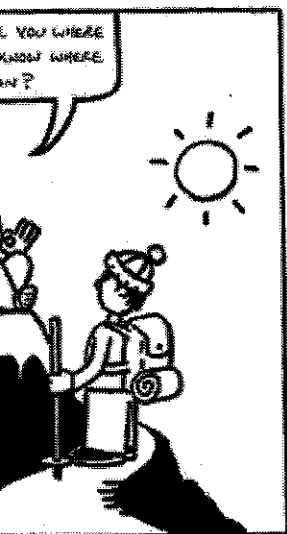
These are not trivial questions. Other forms of data organization often fall apart when dealing with "where." The telephone book, for example, a list organized alphabetically by last name, gives only relative locations (street addresses) and a house number. An entirely new directory is necessary for each new district, and to retrieve the telephone number of a friend in another town, perhaps just across the river, becomes a major problem because you require a different telephone directory than the one covering your own neighborhood.

The properties of geographic search, finding all the phone numbers of people on a single city block, for example, are not available easily to the user of a telephone directory. The secret to *data retrieval*, the ability to gain access to a record and its attributes on demand, is in data organization. In Chapter 4 we examined the various ways that the graphic part of a GIS can be structured for storage inside the computer. Which structure any given GIS uses, and how the map is encoded into that structure, fully determine how easy it is to find a record and extract its values for use. Once again, the attribute and the map data have different means of access. At the most simplistic level, the GIS is a computer program that accesses data stored in files. Obviously, with a great deal of data, the ability to access the files quickly is important if the user is to receive interactive control of the GIS.

At the logical level, access requires a *data model*, a theoretical construct that becomes the key for unlocking the data's door. In medieval times, when much information had to be memorized, monks would train their powers by spending hours committing to memory the parts of a specific cathedral, capturing a mental picture of the interior and exterior so that as they later learned text, such as the contents of the Bible, they could mentally "place" a chapter, a verse, a line, or even a word in a location that they had already memorized (Figure 5.1). Reciting that verse, then, became the memorization in reverse. The monk would mentally walk to the right place in the cathedral, and the words would be stored there. The cathedral, not in substance, but as a visual memory, became the data model. Without such a data model, data cannot be searched or extracted and therefore become worthless.

We can define a data model, then, as a logical construct for the storage and retrieval of information. It is the computer's way of memorizing all the GIS data that we need to use. This is different from the data structures we examined in Chapter 4, because these deal primarily with how the data are physically stored in files on the computer system. As we have seen, this means that a GIS must have at least two data models, and that the two must have a bridge or link between them to tie the attributes and the geography together. These are the *map data model* and the *attribute data model*. In Chapter 4 we illustrated some of the storage and organization aspects of map data models. In this chapter, we deal with the attribute models and then look closely at how the models help in locating and then extracting data from both map and attribute databases.

The database management system's (DBMS's) heritage is from within computer science, but the user community is as broad as that of GIS, literally millions of firms, accountants, colleges and universities, banks, and so forth that need to keep and organize records by computer. The earliest database management systems date from the efforts of the early 1970s, when large mainframe computers were used, data-entry was



not respected subfield of geography.

?" More important, a GIS
at relates to the map behind
geographical properties and,

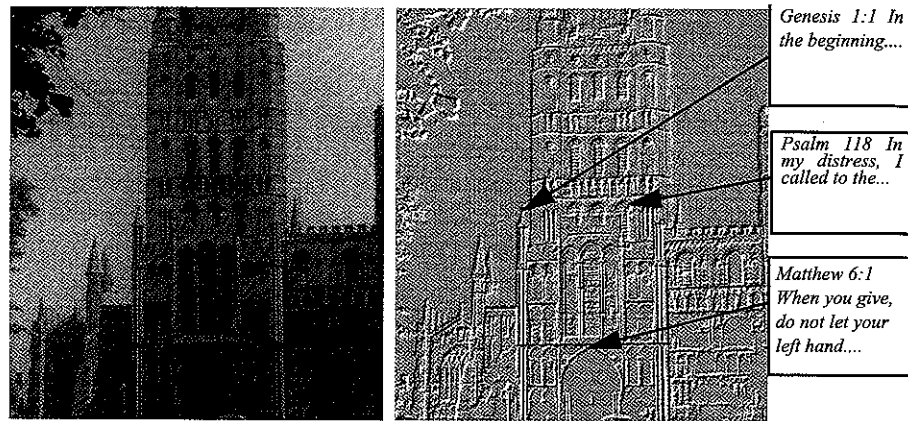


FIGURE 5.1: Medieval monks used the cathedral as a data model for storing documents such as the Bible. The data model in GIS is a logical construct for the storage and retrieval of spatial information. (Photo of Ely Cathedral, UK.)


by key punch and punched cards, and the technology was called *automatic data processing*.

Database management went through its own revolutions due to the technological trends we have already discussed from the GIS viewpoint: the microcomputer, the workstation, the network, low-cost bulk storage, interactive and graphical user interfaces, and so on. Database management, however, has also been significantly influenced by the intellectual breakthroughs that led to radical changes in the way that attribute data can be stored in files. The latest revolution, the object-oriented database system, is now under way and is discussed in Chapter 10.

The parts of a DBMS have remained fairly consistent over time, regardless of how the attribute data are actually placed into files. The *data definition language* is that part of the DBMS that allows the user to set up a new database, to specify how many attributes there will be, what the types and lengths or numerical ranges of each attribute will be, and how much editing the user is allowed to do. This establishes the *data dictionary*, a catalog of all of the attributes with their legal values and ranges. Every DBMS has the ability to examine the data dictionary, and the data dictionary itself is a critical piece of metadata (data about the data) that is often required when the database has to move between systems.

The most basic management function is *data entry*, and because most entry of attribute data is fairly monotonous and may be by transcription from paper records, the DBMS's data-entry system should be able to enforce the ranges and limits entered into the data dictionary by the data definition language. For example, if an attribute is to contain a percentage, and the data-entry person types a value of "110", the DBMS should refuse to accept the value and alert the person.

All data entry is subject to error, and the first step after entry should be *verification*. This is often done by producing a report or printed copy of the data in a standard form that can be checked against the original. Even if the data are error free, most databases must be *updated* to reflect change. Deletion, insertion, and modification of records, or



Genesis 1:1 In
the beginning...

Psalms 118 In
my distress, I
called to the...

Matthew 6:1
When you give,
do not let your
left hand....

documents such as the Bible.
tial information. (Photo of Ely

called *automatic data*

due to the technological
microcomputer, the work-
technical user interfaces, and
cantly influenced by the
that attribute data can be
se system, is now under

r time, regardless of how
n language is that part of
cify how many attributes
of each attribute will be,
es the *data dictionary*, a
es. Every DBMS has the
itself is a critical piece
he database has to move

d because most entry of
ion from paper records,
anges and limits entered
xample, if an attribute is
ue of "110", the DBMS

ry should be *verification*.
e data in a standard form
error free, most databases
modification of records, or

sometimes changes to the data dictionary itself, must be made frequently. This is done by using the data maintenance part of the DBMS. Care must be taken when updates are made, because changes create a new updated version of the entire database. Sometimes modifications are done in batches and a new version of the database "released," reflecting a whole suite of changes, perhaps to reflect the calendar year.

With the preceding tasks complete, the DBMS can then be used to perform its more advanced functions. These are the *sorting*, *reordering*, *subsetting*, and *searching* functions. For example, a database of student records could be sorted by grade-point average to find all students below a "C" average. A database of students containing zip codes in their address could be reorganized by zip code to make mailing easier for the mailroom staff. Subsetting involves using the *query language*, that part of the DBMS that allows the user to interact with the data to perform these tasks, to create a new data set that meets certain search criteria, such as all students who have more than 100 credits toward their degree. Finally, a *search* for a specific record is often needed. At a public terminal, for example, students may be allowed to type in their Social Security numbers to allow them to examine the grades they got in a given semester. All of these functions are part of regular DBMS and are performed differently in each different system. Nevertheless, all of these functions are common to both DBMSs and GISs.

The first-generation DBMSs used a hierarchical structure for their file organization. For example, a university might contain a division or school, the school might contain a department for a single discipline such as Geography, and a department may have a group of students who are majors in that department (Figure 5.2). All students who are declared majors must be assigned to a department, each department belongs to one division, and the entire university consists of a group of divisions. The file structure of a hierarchical system can be organized in the same way. A top-level directory could contain a list of divisions and a set of divisional directories. Changing down to the next level, the divisional level, reveals a list of departments and departmental directories. Down in a departmental directory lie the files containing the student records, one student per record, with attributes such as the student's name and address, year of expected graduation, and grades in various classes. The first generation of database managers used exactly this type of organization of files and records as its data model.

Life is not as simple as the hierarchical model would like it to be, however. In many cases, relationships between records overlap. This is even more so for geographic data, as we will see below. For now, consider a simple hierarchy for the political administration of a state. The order could go state-county-city-district. For much of the country, this would work. In New York City, however, the city of New York consists of the five boroughs, each of them a county in its own right as far as the state of New York is concerned. So the simple hierarchy model already fails, because the hierarchy is literally stood on its head in one case. The data model would require that the county file contain records for cities, but the reality would be the opposite, that the city should contain the county.

Another complex case would be multiple membership. Consider a single house that falls into a fire district, a police district, a school district, a voting district, and a census tract. Five databases, each structured using a different hierarchy, would have five completely independent ways of getting to the attributes for the single house. Although

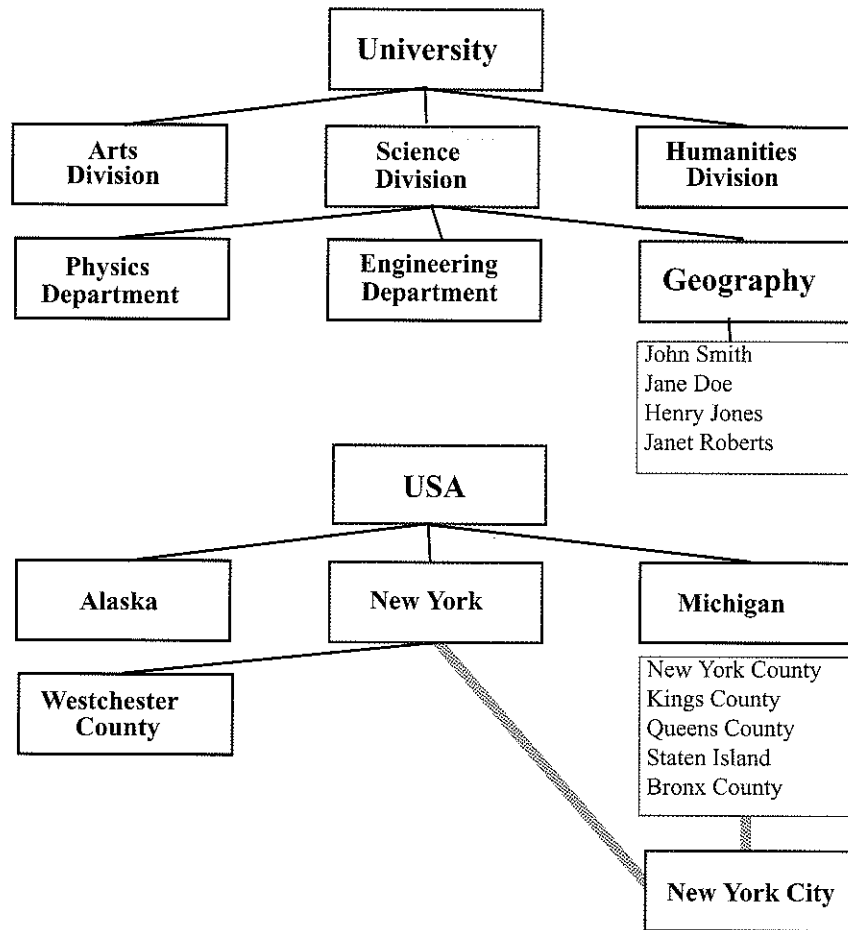


FIGURE 5.2: Hierarchical structures for information. The university example is a simple hierarchy. The nation/state/country/city example is an "inverted" hierarchy. The hierarchical file organization or data model would not work in this case.

each database might store different kinds of information, at some stage it might be useful to assemble together all of the data to be used. Under a hierarchical structure, this cannot be done.

The revolution in DBMS that untangled this database logjam was the use of relational database management systems. Beginning with a set of theoretical breakthroughs in the 1970s, relational DBMSs swept the field and replaced almost all existing systems during the 1980s. They remain the dominant form of DBMS today. The relational model is rather simple, and from the user's standpoint is an extension of the flat file model (Figure 5.3). The major difference is that a database can consist of several flat files, and each can contain different attributes associated with a record. Using the example of the house above, the house can now be a single record in several databases, none of which requires a hierarchy. If the hierarchy still makes sense, we can keep separate files by district or use codes such as zip codes to reveal districts.

5.2 SEARCHING

Most G
ager, o
attribu
include
all reco
to be o
genera
then th

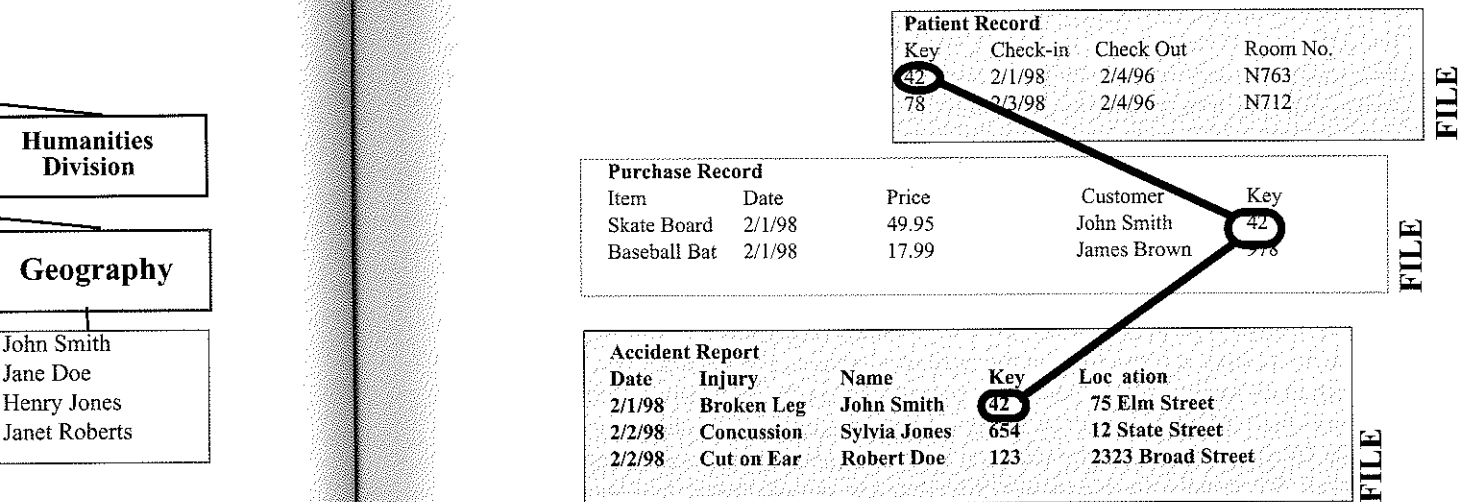


FIGURE 5.3: In a relational database, files with different structures are linked by keys. Records with a common key relate to one feature within the GIS.

Critical to each part of a relational database is a special attribute that serves as a marker rather than a regular attribute. We could assign to every record a unique identifier, for example, or time and date stamp a transaction record to make it different from all others. This "key" attribute can then serve as a link between the flat files. Because the key is unique it allows us to extract various attributes and records from one database and others from another as required. We can store the data in a group of linked files, each of which can be used, have data entered, and be edited and updated and searched separately and without affecting the others.

The relational database manager contained a new set of data management commands that allows the keys and links to be exploited. These typically include such actions as *relate*, to select from two flat files that have a common key attribute, and *join*, to take the relate output and merge them into a single database. So the relational data model, while permitting records and attributes to be separated into different files for storage and maintenance, also allows the user to assemble any combination of attributes and records, as long as they are linked by a key attribute. A *join* can create many unneeded sub-records for a single feature, such as multiple records with different dates, so care should be exercised when joining databases.

5.2 SEARCHES BY ATTRIBUTE

Most GIS systems include as part of the package a fairly basic relational database manager, or simply build on the existing capabilities of a database system. Searches by attribute then are controlled by the capability of the database manager. All DBMSs include functions for basic data display; that is, show all attributes in a database, show all records with their attributes, and show all existing databases. Most also allow records to be output in a standard form, with a particular page layout and style, called a *report generator*. If we need a paper copy of the database, perhaps for checking and verification, then the report generator is used.

As far as actually doing retrieval is concerned, the DBMS must support functions that fall into the category of *query*. As we have seen, a DBMS should allow sufficient data query that any record can be isolated and any subset required for mapping found easily. We may also sometimes wish to reorder or renumber an attribute.

A *find* is the most basic attribute search. Find is usually intended to get a single record. We might find record 15, for example. Finding can be by *search* or by *browse*. Browse searches record by record, displaying each, until the user finds the one needed. Sorting can sort alphabetically for a field, or numerically for a number. Note that a sort may or may not deal with missing values, and where it places them may be significant.

A *restrict* operation allows the user to retrieve a subset of the total number of records by placing a restriction on the attributes' values. For example, we could restrict a search to all records with a date more recent than 1/1/99, or to cities with a population of more than 100,000 people. A *select* operation allows us to choose what attributes will be taken out from another database to form a new database with fewer "selected" attributes. We usually do this to *join* these records and attributes onto another database in the relational system. As we will see in Chapter 6, a *compute* operation allows us to compute a value for an attribute, to assign a value, or to do mathematical operations between attributes—divide one by another, for example. We can also usually *renumber* an attribute, that is, change the values to our specifications. We might want to find all percentages in an attribute and change them to a zero if they fall below 50% or a one if they are greater, so that we can do a binary combination with another.

For example, in a database of state populations and areas called states, we could use compute to create a new attribute called population_density.

```
compute in states population_density = population / area
<50 records in result>
```

This creates a new attribute, which we can recode into high (3), medium (2), and low (1):

```
restrict in states where population_density > 1000
<20 records selected in result>
```

```
recode population_density = 3
<20 values recoded in result>
```

```
join result with states replace
<20 records changed in state>
```

```
restrict in states where population_density > 100
<12 records in result>
```

```
recode population_density = 2
<12 values changed in result>
```

```
join result with states replace
<12 records changed>
```

5.3 SEARCH

```

compute in states where population_density! = 3 or
  2 population_density = 1
<18 records changed>

```

The recoding is now complete. We can now sort by the new recoded value.

```

list attribute in state population_density

<In database 'state' attribute values for
  'population_density'>
<1 18 records>
<2 12 records>
<3 20 records>
<no missing values>

sort result by population_density
<50 records in result>

replace state with result
<50 records changed>

```

In this exchange, note that commands are given one line at a time and often must be used in combination to get the desired effect. Note also that most database systems work by performing their operation on a temporary working set (called *result* in the example) that must be placed back into an existing database when necessary. Many DBMSs use menus, variations on different query languages, and different keywords and commands to accomplish the same results.

Before we leave searching by attribute, consider the problems of using these tools to do even a simple search by distance. Merely to find the distance of each record from one point, we would have to do two subtractions, a multiplication, and a squaring, and then a summation and a square-root operation. Obviously, these database tools, although of immense use, are going to be only humble assistants in our geographical searching needs.

5.3 SEARCHES BY GEOGRAPHY

When we considered searching attributes, we looked at the following search and retrieval commands: *show attributes*, *show records*, *generate a report*, *find*, *browse*, *sort*, *recode*, *restrict*, and *compute*. Moving over to the spatial data within a GIS, some of the operations possible are just spatial equivalents of these, while some are more complex. We discuss the simple retrievals first.

In the map database our records are, instead, features. There are some special attributes specific to the spatial data, and those relate to the coordinates and their measures, plus the characteristics of the lines and polygons. Showing attributes, then, consists of examining the new spatial attributes, such as the actual coordinates themselves, the lengths of arcs, and the areas of polygons. Note that these are already valuable. They could have been the source of the areas used in the population density calculations in the example in Section 5.2. Using the attribute search functions on these attributes now has

spatial consequences. For example, we could find all arcs greater than a certain length, or polygons more than 1 hectare.

Show all records in a spatial sense becomes either show all attributes or display all features on a map. Generating a map, which allows us to search for information visually, is a spatial retrieval operation as far as the GIS is concerned. If we wish to generate a report, the spatial equivalent would be to produce a finished map to cartographic standards, including labels, metadata, legend, and so on.

Browsing works on a map by highlighting. We may color-code a specific feature or features. Some GISs allow a displayed single feature to be blinked on and off for visual effect. Finding becomes what many GIS packages call *identify* or *locate*; that is, use a pointing device of some kind such as a mouse to point to a feature. Indicating its selection successfully can then retrieve that feature's attributes from the attribute database (Figure 5.4).

This spatial searching, browsing the map and picking features, like turning over stones at the beach to search for crabs, is a very powerful GIS capability indeed, especially if the GIS is on a portable computer and the feature located is the one you are standing in front of as located by your portable GPS receiver. On the map we can also search by indicating a single feature, all features within a rectangle that we drag out, or all features within an irregular area that we sketch on the screen with a drawing tool.

Sorting has less spatial meaning and is usually given a GIS context by examining the spatial pattern that results from a sort by attribute. For example, we could sort states in the state database used in Section 5.2 by their per capita income, then display on the

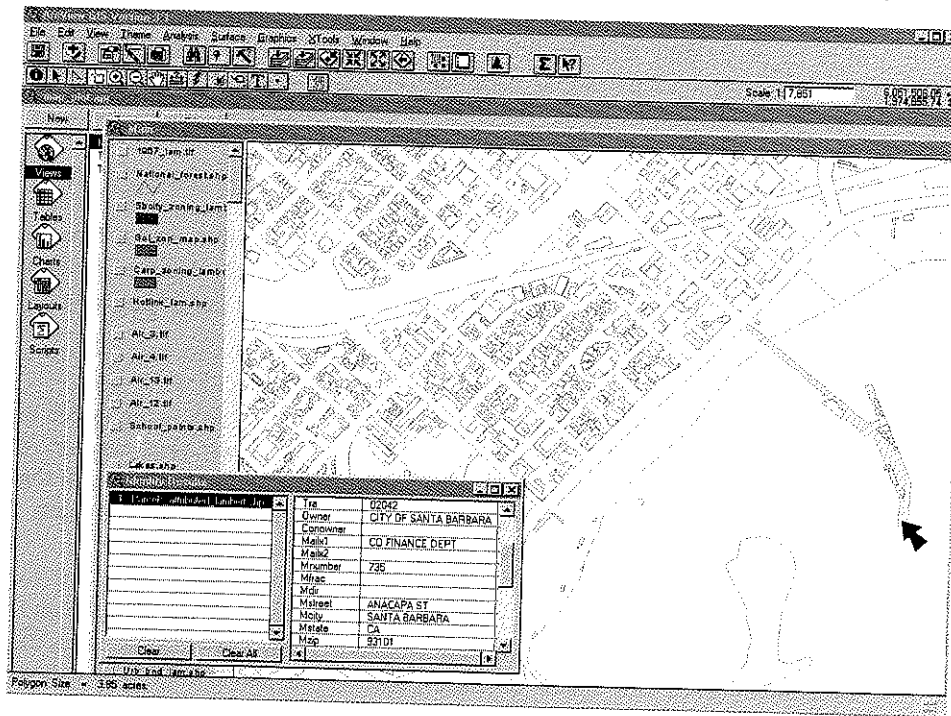


FIGURE 5.4: Spatial search in a GIS using the ArcView identify tool to list the attributes for a polygon chosen by pointing. Parcel and building data are for the City of Santa Barbara. (Courtesy of Ryan Aubry.)

r than a certain length,

attributes or display all
or information visually,
If we wish to generate
d map to cartographic

-code a specific feature
blinked on and off for
entify or locate; that is,
a feature. Indicating its
in the attribute database

tures, like turning over
bility indeed, especially
ne one you are standing
o we can also search by
drag out, or all features
wing tool.

S context by examining
ole, we could sort states
me, then display on the



t the attributes for a polygon
(Courtesy of Ryan Aubry.)

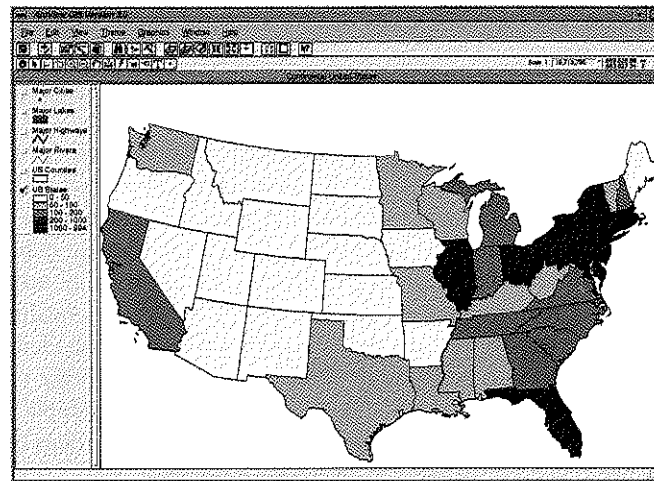


FIGURE 5.5: Recoding (classifying) the attribute values to make a shaded choropleth map showing coterminous United States population density in 1990 using ArcView GIS 3.0.

map the highest and lowest states. Similar to this is recoding by attribute. In the example in Section 5.2, we converted the density values into high, medium, and low values by recoding. We could use this recoded attribute directly to make a choropleth or shade-tone map of the population density, as in Chapter 7 (Figure 5.5).

Operations that perform attribute manipulations, recode, and compute in a spatial sense merely change which features are displayed and how. For example, we could compute and display a new attribute, such as population density in Section 5.2. However, each operation has an equivalent spatial operation. Recoding features spatially, that is, changing the scope of their attributes, is equivalent to a spatial merge. Removing isolated pixels by assigning them to their enclosing or most dominant neighboring region is an example.

Compute can be given a spatial context as distance, length, area, or volume computations and transformations. For example, we could generate a new map that contains the distance to the nearest point feature, the cumulative downstream distance along a stream, or the travel distance along a road network. These new maps could be displayed or used in conjunction with the GIS layers to perform more complex operations, just as we combined the attribute *query commands* in the preceding section to achieve an end result.

Select and join are the remaining operations. *Select* means to extract specific attributes and to reduce the width of the database. By first selecting, we could use only certain themes or layers in a GIS retrieval operation, or we could change the map scale or extent. Picking a subregion, merging quadrangles into a county, aggregating land cover categories from level 2 to level 1 (from seven classes to one class for urban land, for example), picking only major rivers from a full stream network, or generalizing lines on the map for depiction at a broader scale are all examples of a geographic select.

The form of select used most frequently in GIS operations is the buffer operation, that is, to select only those parts of a map or those features that lie within a certain distance of a point, a set of points, a line, or an area. Examples are to restrict a search for

malaria victims in a West African nation to those villages more than 10 kilometers from a health clinic, or to limit our search for a summer cottage to one within 200 meters of a lake. Buffers around points form circular areas, around lines they form “worms,” and around polygons they form larger regions (Figure 5.6).

A *join* operation is the cross-construction of a database by merging attributes across flat files. In the geographic sense, this is termed a map *overlay*. In a map overlay, a new map is created that shares the space division of both source maps (Figure 5.7). Every new polygon created on the map has a new attribute record in an expanded attribute database associated with the map overlay. This means that the overlay map is searchable by either of the sets of regions used to create it. Examples are city health districts overlain with zip codes, so that we can use data assembled on health, accidents, and so on, with data from mailing lists on population, ethnicity, income, and other variables. By joining the map layers with a map overlay, we can compute, for example, the average income of people suffering from heart disease, or determine by age groups those people at risk from certain age-related diseases (Figure 5.8).

The power of the attribute database manager came from the use of multiple operations in sequence. The same is true of GIS retrieval operations. For example, we could take the overlain health district data in Figure 5.8 and again multiply it with a computed distance map from hospitals represented as points, perhaps geocoded from the

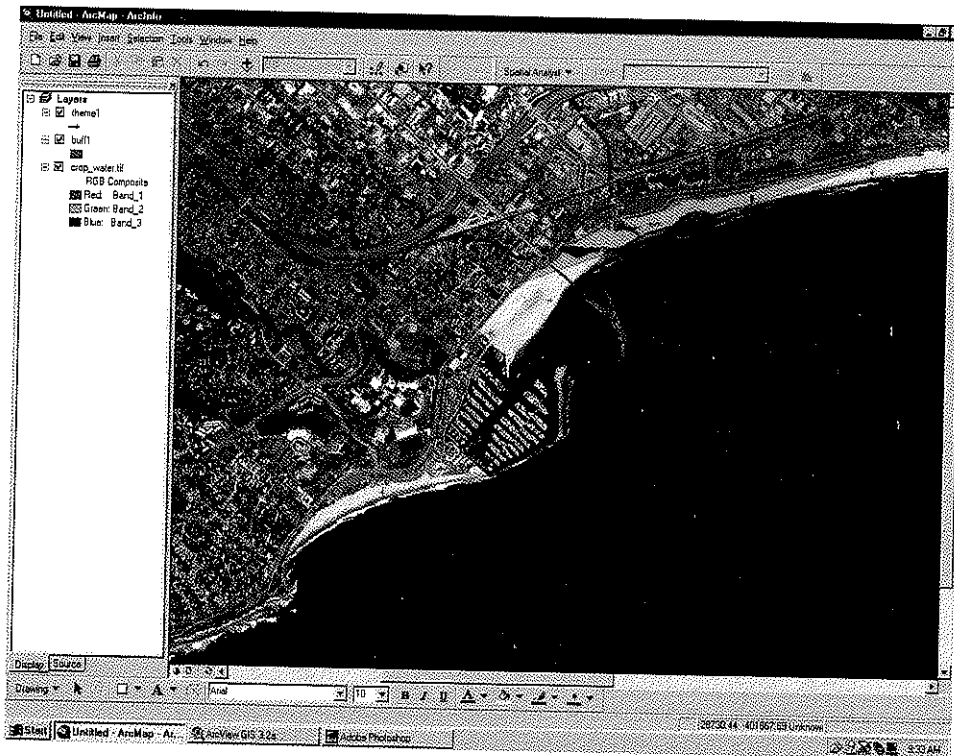


FIGURE 5.6: An example of GIS buffering as a retrieval mechanism. One thousand foot buffer from the railroad tracks in downtown Santa Barbara, shown in transparent red. ArcInfo 8 image by Ryan Aubry.

FIGURE
data an

FIGURE
The GIS
(Map by

than 10 kilometers from
the within 200 meters of
they form "worms," and

merging attributes across
in a map overlay, a new
(Figure 5.7). Every new
attribute database
p is searchable by either
h districts overlain with
ts, and so on, with data
variables. By joining the
the average income of
those people at risk from

the use of multiple oper-
For example, we could
multiply it with a com-
aps geocoded from the



thousand foot buffer from the
3 image by Ryan Aubry.

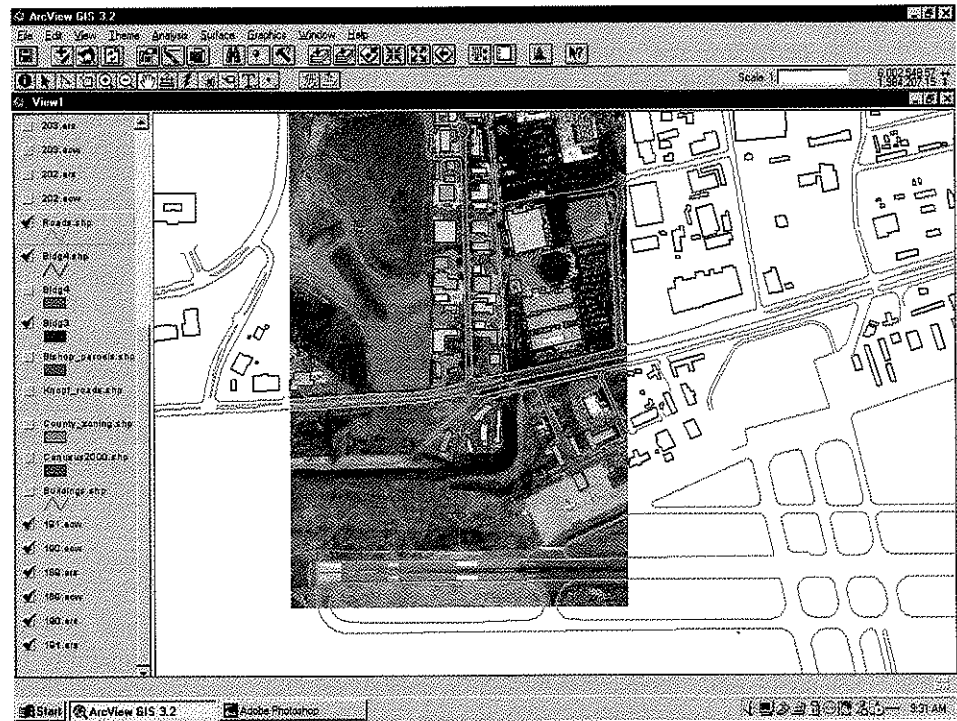


FIGURE 5.7: Map overlay unites different map layers by giving them a common reference base. Here road data and building footprints are overlain onto a high resolution air photo of Goleta, CA.



FIGURE 5.8: Map overlay. Example shows health districts and zip codes overlain for parts of New York city. The GIS can combine the two, so that attributes from one database can be cross referenced with the other. (Map by Barbara Tempalski.)

Yellow Pages of the telephone book. This would allow us to display a map showing, perhaps in red, where large numbers of people live long distances from hospitals. Some geographic queries don't fit a clear definition of attribute query operations. A map display can be zoomed into and out of to change the equivalent resolution with comparative ease. Some geographic queries can search by geographic properties and topology.

An entire suite of geographic searches are searches and tests by relations of points, lines, and areas. For example, we can select all points enclosed within one or more regions. Join then allows us to assign attributes from the point features to the area features, say weather statistics from weather stations, to administrative districts. Typical GIS searches are point in polygon, line in polygon, and point distance to a line. If the points are oil storage tanks and the lines are rivers, this can have great analytical value. We can also weight layers in an overlay, perhaps building a composite layer called *land suitability* in the same way that early planners did with overlay mapping. Another popular GIS layer to construct by weighting is a cost surface, built from a combination of layers and distance calculations. Low spots on this map may be good for business locations.

Finally, some highly specific geographic computations are possible. Examples are line-of-sight calculations, computing a viewshed or region that is visible from one place on a map using a base of terrain; maps showing slope and aspect or direction of slope, which might be useful in assessing development suitability or flooding potential; traffic volumes measured on a street network, used to predict traffic jams; or maps showing merged outputs of models or predictions with data, to predict earthquake hazards.

5.4 THE QUERY INTERFACE

Both database management and geographic information management share the fact that the user must somehow interact with the data in an appropriate way. The first generation of DBMS and GIS both used only batch-type interaction with the data, usually closely linked to working with the operating system, the physical management of disk, and so on. This type of interaction dates from the punched card, in that all processes had to be thought out in advance and a file (or stack of cards) produced that could execute the different commands one at a time.

When interactive computing became commonplace, the command line as a query vehicle for data query took over. Commands were typed into the computer one at a time, under the control of the DBMS itself, and the software responded by performing the computations one at a time while the user waited for the command to be completed. Many GISs still use this type of interaction, or permit it to allow the use of macros. *Macros* are files containing commands to be executed one at a time. If an error is detected in a macro, the execution can be stopped and the file modified to correct the mistake.

In the typical form, a command consists of a keyword for the operation such as IMPORT, OVERLAY, SELECT, and so on, and a set of optional or required parameters. Parameters may be file names, numerical values associated with the task, names of options, or any other pertinent value. Many GIS packages will supply defaults for any parameters left out, and most respond to the command without parameters by giving a list of parameters. The largest GIS packages can have hundreds or even thousands of commands, giving more than one way to do a particular task by different routes.

5.5 STUDY

5.5

CHA

Bas

Most GIS packages now are fully integrated with the WIMP (windows, icons, menus, and pointers) interface specified by the operating system, such as Windows or X-Windows. Choices are now most commonly made by menu, with message windows popping up for the user to provide essential parameters when they are required. Values can also sometimes be set by sliders, widgets, and by screen tools such as dials, choice lists, and buttons.

Another fairly recent trend is that most GISs also contain a language or macro tool for automating repetitive tasks. Examples are ArcView's Avenue and Visual Basic, MapInfo's MapBasic, and Arc/Info's AML. In some cases, these languages can interact with the graphical user interface (GUI) tools, presenting a choice as a menu, for example. Thus any GIS user can now become a programmer, establishing his or her own particular task as a query tool for all other users. In large GIS operations when training and employee time are limited, GIS analysts are often employed to automate many other GIS tasks, such as routine queries or database editing.

Finally, there have been efforts to develop a suite of database interaction commands that all users can assume as standard interface to relational databases. The result, the Structured Query Language (SQL), has found a broad acceptance as a much used tool in regular database management, although less use in GIS. Some have argued that almost all GIS operations are possible in SQL, while others have sought to extend its capabilities into the spatial domain. Given the differences between DBMSs, this is a welcome effort.

Most GISs have often lagged somewhat in their attribute database capabilities behind commercial systems for more routine data applications. The more recent interfaces, however, and the broad support for macros and windowing systems have led to a fair amount of convergence in the "look and feel" of GISs, although we are still a long way away from all GISs functioning in much the same way. Given the rich variety of GIS applications, and the rapidly expanding nature of the field, the standardization of GIS operations and query mechanisms is still many years away.

5.5 STUDY GUIDE

5.5.1 Summary

CHAPTER 5: What Is Where?

Basic Database Management (5.1)

- A GIS can answer the question: What is where?
- Retrieval is the ability of the DBMS or GIS to get back on demand data that were previously stored.
- Geographic search is the secret to GIS data retrieval.
- Many forms of data organization are incapable of geographic search.
- GIS systems have embedded DBMSs, or link to a commercial DBMS.
- A data model is a logical construct for the storage and retrieval of information.
- GIS map data structures are map data models.
- Attribute data models are needed for the DBMS.
- The origin of DBMS data models is in computer science.
- A DBMS contains:
 - A data definition language
 - A data dictionary

- A data-entry module
 - A data update module
 - A report generator
 - A query language
- DBMS queries are sorting, renumbering, subsetting, and searching.
 - The query language is the user interface for searching.
 - Historically, databases were structured hierarchically.
 - Most current DBMSs use the relational model.
 - Relational databases are based on multiple flat files for records, with dissimilar attribute structures, connected by a common key attribute.

Searches by Attribute (5.2)

- Examples of searches by attribute are find and browse.
- Examples of data reorganization are select, renumber, and sort.
- Compute allows the creation of new attributes based on calculated values.
- Attribute queries are not very useful for geographic search.

Searches by Geography (5.3)

- In a map database the records are features.
- The spatial equivalent of a find is locate; the GIS highlights the result.
- Spatial equivalents of the DBMS queries result in locating sets of features or building new GIS layers.
- Buffering is a spatial retrieval around points, lines, or areas based on distance.
- Overlay is a spatial retrieval operation that is equivalent to an attribute join.
- Combinations of spatial and attribute queries can build some complex and powerful GIS operations, such as weighting.

The Query Interface (5.4)

- GIS query is usually by command line, batch, or macro.
- Most GIS packages use the GUI of the computer's operating system to support both a menu-type query interface and a macro or programming language.
- SQL is a standard interface to relational databases and is supported by many GISs.

5.5.2 Study Questions

Basic Database Management

Make a table listing the component parts of a DBMS. What particular task or tasks does each section of the DBMS do? Add a column to the table giving a brief summary of the role of that component. For example, the data-entry module could have the entry, "Permits user to enter attribute data into the database." Add another column that lists parallels between attribute databases and map databases, such as "report generation" and "map display."

5.6 EXERCISES

1.

2.

3.

4.

5.7 REFERENCES

Ber
C
Bur
n
ESE
Peu
g

Searches by Attribute

List and define each of the tools that the user of a DBMS has available for searching by attribute. What are the differences between the types of search; for example, *find* versus *browse*?

Searches by Geography

What geographic retrieval tools are available for each of the following: point features, line features, and area features? How can these be used in combination to make complex queries to the GIS?

The Query Interface

What are the major types of user interfaces that the GIS user could face as he or she moves from one GIS software package to another? What are the advantages and disadvantages of each? Which would you prefer as a new GIS user?

5.6 EXERCISES

1. Using the DBMS within your GIS and a sample data set, work through each of the following attribute-only queries: *find*, *sort*, *browse*, *restrict*, *select*, and *compute*. In each case, use the report generator to print a clean copy of the result, printing only a few records. Write a one-paragraph set of instructions for a novice to do the same thing.
2. Use the geographic search commands within your GIS and a sample data set to do the following spatial queries: *locate*, *highlight*, and *buffer*. In every case, generate a simple but well-designed map to show the outcome. How many records does each of the commands isolate, and why?
3. Using two different maps that are coregistered within your GIS at the same scale and on the same projection, do a map overlay. Use the report generator of the DBMS to list out the attributes of each separate, and then the composite set of "most common geographic units." How many records are in each data set? Did the overlay require you to deal with sliver polygons? How would your GIS deal with any very small polygons that might have been created? Calculate a per polygon time that the operation took, using a stopwatch as a timer if your computer does not print execution times.
4. If you have access to two GIS systems, examine the manuals and the software to see what the user is expected to do to conduct (a) a spatial "locate" and (b) a DBMS-style list of the names of the attributes, and a list of the first few records' attribute values. What sorts of steps are needed? How intuitive are the steps? What sort of user interface does the GIS require you to use (e.g., menus, WIMP, command line, etc.).

5.7 REFERENCES

- Berry, J. K. (1993) *Beyond Mapping: Concepts, Algorithms and Issues in GIS*. Fort Collins, CO: GIS World.
- Burrough, P. A. (1986) *Principles of Geographical Information Systems for Land Resources Assessment*. Oxford: Clarendon Press.
- ESRI (1995) *Understanding GIS: The Arc/Info Method*. New York: Wiley.
- Peuquet, D. J. (1984). "A conceptual framework and comparison of spatial data models." *Cartographica*, vol. 21, no. 4, pp. 66–113.

Urban Geographic Information Systems. New York: McGraw-Hill.
ective. London: Taylor and Francis.

measurement or value for a feature. Attributes
they can be dates, standardized values, or field
ch data are collected and organized. A column

to the computer from a file rather than directly
e.

peated examination of records until a suitable

numerical data (but not simply "counts") for a
ata into classes and (2) shading each class on

at uses the numerical values of one or more
y attribute created by the command.

DBMS that allows the user to set up a new
s there will be, what the types and lengths or
be, and how much editing the user is allowed

tes for a data set, along with all the constraints
data definition phase. Can include the range
and missing values, and the legal width of

bers into a computer, usually attribute data.
nd or acquired through networks, from CD-
from a GPS receiver, from data loggers, and

on of data for use in an information system.
e by computer.

Part of a GIS, the set of tools that allow the
attribute data.

ction provided for the user by the GIS without

t of a landscape.

n of numbers. The numbers are organized into
es, records as rows, and attributes as columns.
e location on the storage mechanism of a

tended to locate a single record or a set of
of their attributes.

GIS that uses spatial properties as its basis.

model based on sets of fully enclosed subsets

highlight: A way
result of a quer
identify: To find
pointing device
join: To merge bo
key attribute: A
throughout the
locate: See **identi**
macro: A comma
then submitted
menu: A compon
choices from a
overlay: A GIS op
on the basis of
parameter: A nu
submitting a co
query: A question
system or GIS.
query language:
relate: A DBMS
ture them accor
relational model:
attribute structu
renumbering: Us
report generator:
of all the values
restrict: Part of th
selected out of
retrieval: The abil
memory records
search: Any datab
select: A DBMS o
sort: To place the
SQL (Structured
base managemen
subsetting: Extra
update: Any repla
verification: A pro
against their cor

Huxhold, W. E. (1991) *An Introduction to Urban Geographic Information Systems*. New York: Oxford University Press.

Warboys, M. F. (1995) *GIS A Computing Perspective*. London: Taylor and Francis.

5.8 KEY TERMS AND DEFINITIONS

attribute: A numerical entry that reflects a measurement or value for a feature. Attributes can be labels, categories, or numbers; they can be dates, standardized values, or field or other measurements. An item for which data are collected and organized. A column in a table or data file.

batch: Submission of a set of commands to the computer from a file rather than directly from the user as an interactive exchange.

browse: A method of search involving repeated examination of records until a suitable one is found.

choropleth map: A map that shows numerical data (but not simply "counts") for a group of regions by (1) classifying the data into classes and (2) shading each class on the map.

compute: Data management command that uses the numerical values of one or more attributes to calculate the value of a new attribute created by the command.

data definition language: The part of the DBMS that allows the user to set up a new database, to specify how many attributes there will be, what the types and lengths or numerical ranges of each attribute will be, and how much editing the user is allowed to do.

data dictionary: A catalog of all the attributes for a data set, along with all the constraints placed on the attribute values during the data definition phase. Can include the range and type of values, category lists, legal and missing values, and the legal width of the field.

data entry: The process of entering numbers into a computer, usually attribute data. Although most data are entered by hand or acquired through networks, from CD-ROMs, and so on, field data can come from a GPS receiver, from data loggers, and even by typing at the keyboard.

data model: A logical means of organization of data for use in an information system.

database: Any collection of data accessible by computer.

DBMS (database management system): Part of a GIS, the set of tools that allow the manipulation and use of files containing attribute data.

default: The value of a parameter or a selection provided for the user by the GIS without user modification.

feature: A single entity that composes part of a landscape.

flat file: A simple model for the organization of numbers. The numbers are organized into a table, with values for variables as entries, records as rows, and attributes as columns.

file: Data logically stored together at one location on the storage mechanism of a computer.

find: A database management operation intended to locate a single record or a set of records or features based on the values of their attributes.

geographic search: A find operation in a GIS that uses spatial properties as its basis.

hierarchical data model: An attribute data model based on sets of fully enclosed subsets and many layers.

- highlight:** A way of indicating to the GIS user a feature or element that is the successful result of a query.
- identify:** To find a spatial feature by pointing to it interactively on the map with a pointing device such as a mouse.
- join:** To merge both records and attributes for unrelated but overlapping databases.
- key attribute:** A unique identifier for related records that can serve as a common thread throughout the files in a relational database.
- locate:** See **identify**.
- macro:** A command language interface allowing a "program" to be written, edited, and then submitted to the GIS user interface.
- menu:** A component of a user interface that allows the user to make selections and choices from a preset list.
- overlay:** A GIS operation in which layers with a common, registered map base are joined on the basis of their occupation of space.
- parameter:** A number, value, text string, or other value required as the consequence of submitting a command to the GIS.
- query:** A question, especially if asked of a database by a user via a database management system or GIS.
- query language:** The part of a DBMS that allows the user to submit queries to a database.
- relate:** A DBMS operation that merges databases through their key attributes to restructure them according to a user's query rather than as they are stored physically.
- relational model:** A data model based on multiple flat files for records, with dissimilar attribute structures, connected by a common key attribute.
- renumbering:** Use of the DBMS to change the ordering or ranges of attributes.
- report generator:** The part of a database management system that can produce a listing of all the values of attributes for all records in a database.
- restrict:** Part of the query language of a DBMS that allows a subset of attributes to be selected out of the flat file.
- retrieval:** The ability of a database management system or GIS to get back from computer memory records that were stored there previously.
- search:** Any database query that results in successful retrieval of records.
- select:** A DBMS command designed to extract a subset of the records in a database.
- sort:** To place the records within an attribute in sequence according to their value.
- SQL (Structured Query Language):** A standard language interface to relational database management systems.
- subsetting:** Extracting a part of a data set.
- update:** Any replacement of all or part of a data set with new or corrected data.
- verification:** A procedure for checking the values of attributes for all records in a database against their correct values.