Topological

To get an intuitive feeling of what a topological space is, imagine two polygons that touch (meet) each other and are drawn on a rubber sheet. Now if we deform the rubber sheet by stretching or bending but not cutting or folding it, the adjacency of the polygons remains intact. *Meet* is an example of a topological property, and the study of transformations (deformations) that preserve topological properties is called topology. Consider the map of the world with political boundaries of countries. The neighboring countries *meet* each other, whether the map is drawn on a sphere or on a flat space. The area of a polygon is clearly not a topological property. In fact, the relative areas of different countries are often not preserved in many maps. Areas of countries near the equator are reduced relative to the areas of countries near the poles in many planar maps. In Table 2.1 we list some common topological and nontopological properties [Worboys, 1995].

From a spatial/geographic database point of view, topological relationships such as *meet, within,* and *overlap* are most likely to be queried by a user of a spatial database management system. Is a given land parcel adjacent to a hazardous waste site? Does the river floodplain overlap a proposed highway network? All these are examples of topological relationships. Two common types of topological queries in spatial databases are:

- Find all objects that have a topological relation R to a given object.
- What is the topological relation between objects A and B?

The binary topological relationship between two objects, A and B, in a plane $\Re^2$ is based upon the intersection of A's interior (A°), boundary (∂A), and exterior (A^-) with B's interior (B°), boundary (∂B) and exterior (B^-) [Egenhofer, 1994]. The *nine-intersection* matrix between the six object parts defines a topological relationship and can be concisely represented using the following matrix:

$$\Gamma_9(A, B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

By considering the values empty (0) and nonempty (1), one can distinguish between $2^9 = 512$ binary topological relationships. For a two-dimensional region embedded in $\Re^2$, eight relations can be realized, and they provide mutually exclusive complete coverage. These relations are *disjoint, meet, overlap, equal, contains, inside, covers,* and *covered by*.

Figure 2.3 shows how topological relations can be represented using the nine-intersection matrix. For example, the *disjoint* relationship is described in the nine-intersection model by the boolean matrix shown in the top-left corner of Figure 2.3. The zero entries indicate that *interior (A)* has no points in common with either *interior (B)* or *boundary (B)*. Similarly *interior (B)* has no point in common with *boundary (A)*. Similarly, *boundary (A)* and *boundary (B)* have no points in common.

The topological relationships on other pairs of spatial data types, for example *(point, surface), (point, curve)* can be defined in a similar manner. A point can be inside, outside, or on the boundary of a surface as listed in Table 2.1. A curve may *cross* the interior of a *surface* or may *meet* or be *disjoint* from a surface. A point may be an endpoint of a

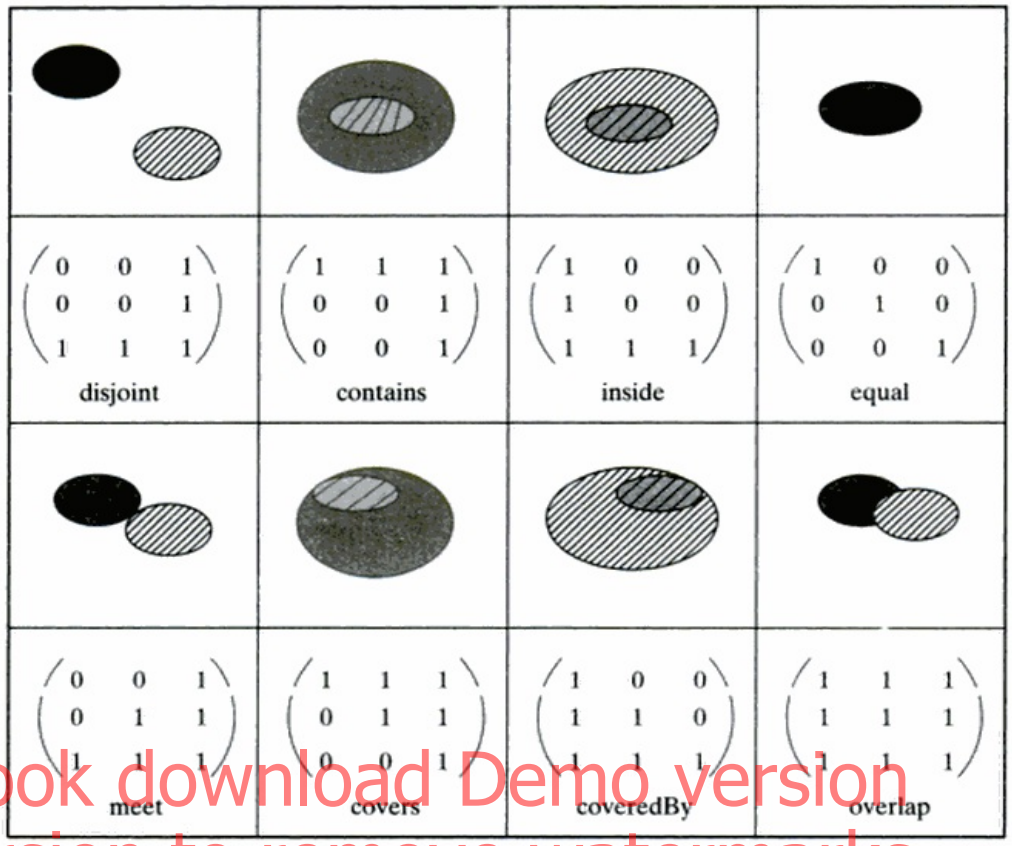


FIGURE 2.3. The nine-intersection model [Egenhofer et al., 1989].

Metric Space

Mathematically speaking, a set X is called a metric space if for any pair of points x and y of X, there is an associated real number $d(x, y)$, called the distance (also called a metric) from x to y, with the following properties:

1. $d(x, y) \geq 0$ and $d(x, x) = 0$
2. $d(x, y) = d(y, x)$
3. $d(x, y) \leq d(x, z) + d(z, y)$

for all x, y, z in X. Any function that satisfies these properties is called a metric on X.

In metric spaces the notion of distance is well defined. A distance function can be used to induce a topology on the space, and therefore every metric space is also a topological space. In network or graph settings, metric spaces play a crucial role. Optimal distance and shortest travel-time queries are ideally handled in a metric space setting.

Euclidean

Let R be the field of real numbers. A *vector space* V over R is a nonempty set V of objects v called *vectors*, together with two operations:

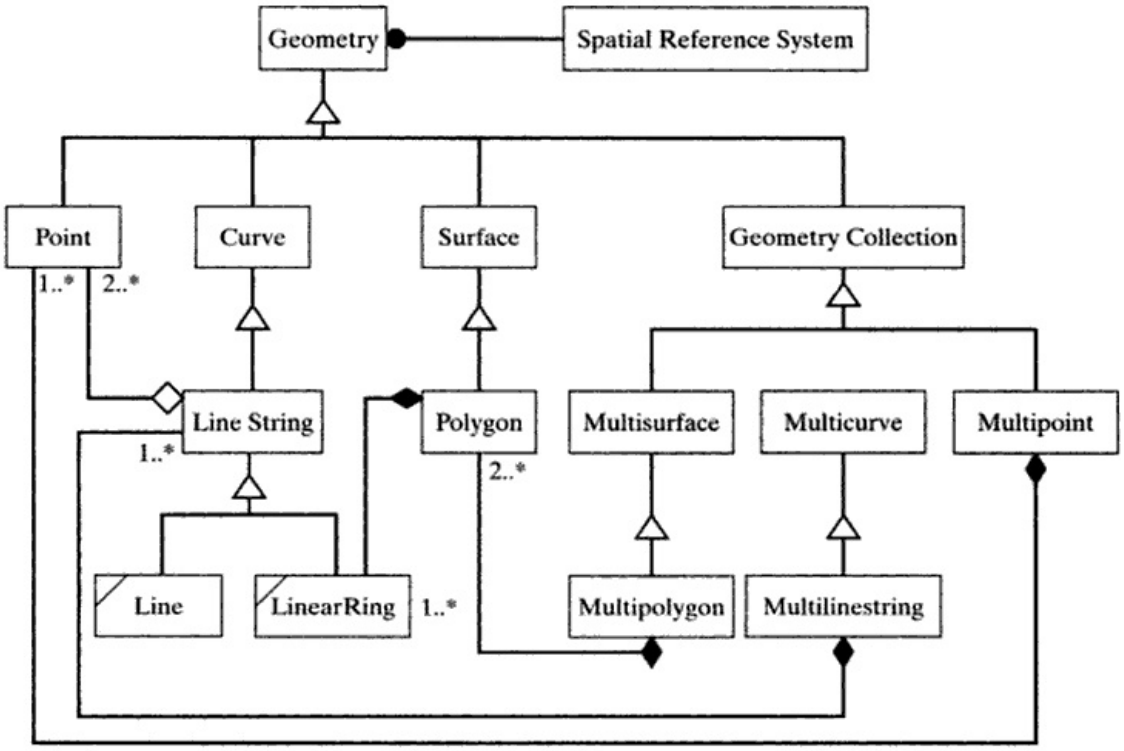


FIGURE 2.2. An OGIS proposal for building blocks of spatial geometry in UML notation [OGIS, 1999].

"geometric-intersection." For example, if one takes a geometric-difference of boundaries of Canada and Quebec, the result is a "multisurface" even if Canada and Quebec were of "surface" spatial data type. This property is useful to support multistep querying and data processing.

2.1.4 Operations on Spatial Objects

How do spatial objects interact with each other? In field-based modeling, the field function principally determines the type of operations allowed. After all, it would not be possible to apply a gradient operation on a nonsmooth field! In object-based modeling, the underlying embedding space determines the relationships that can exist between objects. We now describe a typology of the embedding space and associated relationships.

Set-Oriented

The simplest and most general of all embedding spaces is called set-oriented. Common relationships allowable in this setting are the usual set-based relationships of union, intersection, containment, and membership. Hierarchical relationships such as a forest-stand contained in a forest, a forest contained in a state park, and a state park contained within a state are adequately modeled by set theory.

Zonal Operations

Zonal operations are naturally associated with aggregate operators or the integration function, $\int dx\,dy$ in calculus. In the `forest` example, the step function, which maps the forest onto the set of attributes {Oak, Fir, Pine}, also partitions the underlying space into three polygons, or *zones*. An operation that calculates the average height of the trees for each species is a zonal operation.

2.1.2 Object-Based Models

In object-based modeling, the focus is to abstract spatial information into distinct, identifiable, and relevant things, or entities, called *objects*. For example, we can characterize a park by the forest-stands, rivers, lakes, and roads that are contained in it. All these entities are clearly distinct and identifiable. Whether they are relevant or not depends on the application we are trying to model. Each object has a set of attributes that characterize it. The key difference between spatial objects and the objects/entities that are ubiquitous in traditional database modeling is that the attributes of spatial objects diverge into two distinct categories: spatial and nonspatial. It is through its spatial attributes that an object interacts with the underlying embedding space. In the `Forest` example, the forest-stand representing Fir is a spatial object, with the polygon that represents its spatial extent being its spatial attribute and the name Fir its nonspatial, alphanumeric attribute. A spatial object can have more than one spatial attribute representing, for instance, many levels of generalizations. A road on a map could be represented as a line or a polygon, depending on the scale of the map.

2.1.3 Spatial Data Types

A key issue in object-based models of spatial information is the choice of a basic set of spatial data types required to model common shapes on maps. Many proposals have been made over the years. A consensus is slowly emerging in terms of the OGIS standard [OGIS, 1999]. Figure 2.2 shows the fundamental building blocks of two-dimensional spatial geometry and their interrelationships in UML notation. We give a brief description of the UML notation in Section 2.4.

The most general shape is represented by "geometry" described via a "spatial representation system," which is a coordinate system like latitude/longitude or some other consensus framework. The "geometry" is subdivided into four categories, namely, *point*, *curve*, *surface*, and *geometry collection*. *Point* describes the shape of a zero-dimensional object, for example, city-centers in a map of the world. *Curve* describes the shapes of one-dimensional objects, for example, rivers in the map of a world. The *curve* objects are often approximated by a *linestring*, which is represented by two or more points. The simplest *linestring* is a straight line joining two or more *points*. The category *surface* describes the shape of two-dimensional objects, for example, countries on a map of the world. A *surface* is often modeled by a *polygon*. *Geometry collection* represents complex shapes, such as, collection of oil wells, a group of islands and so on. *Geometry collection* in turn is of three types, namely, *multipoint*, *multicurve*, and *multisurface*. The *geometry collection* spatial data types provide a "closure" property to OGIS spatial data types under geometric operations such as "geometric-union," "geometric-difference," or

attribute domain was the set {fir, oak, pine}. For the special case when the functions are single-valued and the underlying space is the Euclidean plane, fields are naturally viewed as either surfaces or isolines, which are the locus of points that have the same attribute value. The third important component of field-based modeling is specification of the operations on the field functions.

Relationships and interactions between the different fields are specified by *field operations*. Field operations map a subset of fields onto other fields. Examples of field operations are union, $+$, and composition, $\circ$:

$$f + g : x \rightarrow f(x) + g(x)$$
$$f \circ g : x \rightarrow f(g(x))$$

Field operations can be classified into three categories [Worboys, 1995]: *local, focal,* and *zonal*.

Local Operations

For a local operation, the value of the new field at a given location in the spatial framework depends only on the value of the input field at that location. For example, consider an idealized `State-Park` which is completely partitioned into trees, lakes, and meadows. Define a function f and g as:

$$f(x) = \begin{cases} 1 & \text{if } x = \text{“tree”} \\ 0 & \text{otherwise} \end{cases}$$

and

$$g(x) = \begin{cases} 1 & \text{if } x = \text{“lake”} \\ 0 & \text{otherwise} \end{cases}$$

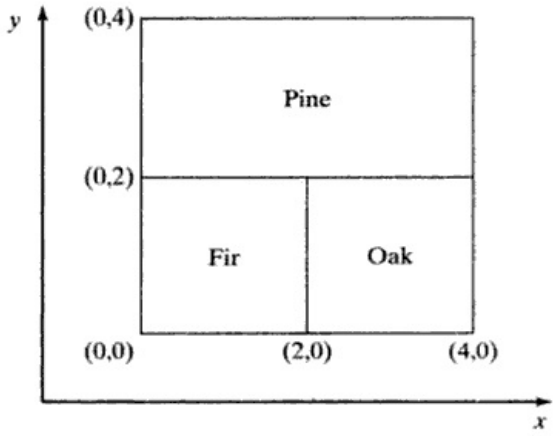
Then $f + g$, that is, the union of f and g is defined as

$$(f + g)(x) = \begin{cases} 1 & \text{if } x = \text{“tree” or “lake”} \\ 0 & \text{otherwise} \end{cases}$$

This is an example of a local operation.

Focal Operations

For a focal operation, the value of the resulting field at a given location depends on the values that the input field assumes in a small neighborhood of the location. The *limit* operation in calculus is an example of a focal operation. Let $E(x, y)$ be the elevation field of the `State-Park`. That is, E at (x, y) gives the value of the elevation at the location (x, y) in the spatial framework F. Then the operation that calculates the gradient of the elevation field, $\nabla \cdot E(x, y)$, is a focal operation, because the value of the gradient at (x, y) depends on the value of the elevation field in a “small” neighborhood of (x, y). Here we are assuming that the field varies smoothly and does not experience sharp jumps, as the step function did in the **Forest** example.



(a)

Object Viewpoint of Forest Stands

Area-ID	Dominant Tree Species	Area/Boundary
FS1	Pine	[(0,2),(4,2),(4,4),(0,4)]
FS2	Fir	[(0,0),(2,0),(2,2),(0,2)]
FS3	Oak	[(2,0),(4,0),(4,2),(2,2)]

(b)

Field Viewpoint of Forest Stands

$$f(x,y) = \begin{cases} \text{"Pine,"} & 2 \leq x \leq 4\,;\, 2 < y \leq 4 \\ \text{"Fir,"} & 0 \leq x \leq 2;\, 0 \leq y \leq 2 \\ \text{"Oak,"} & 2 < x \leq 4;\, 0 \leq y \leq 2 \end{cases}$$

(c)

FIGURE 2.1. The object-field dichotomy. (a) A map showing three forest stands: *Pine, Fir, Oak*. (b) An *object* viewpoint representing the map as a collection of three objects. Each object has a unique identifier, a dominant tree species, and an area. The boundary of the area, a polygon, is specified by the coordinates. (c) A *field* viewpoint, where each point is mapped to the value of the dominant tree species.

2.1.1 Field-Based Model

Defining the field model for a spatial application requires that we determine three components: a *spatial framework*, *field functions*, and a set of relevant *field operations* [Worboys, 1995].

A spatial framework F is a finite grid imposed on the underlying space. All measurements are then based on this framework. The best-known example of a spatial framework is the system of latitude and longitude to reference the earth's surface. A spatial framework is a finite structure, and errors introduced due to discretization are unavoidable. A finite set of n computable functions or simple fields $\{f_i,\ 1 \leq i \leq n\}$,

$$f_i : \text{Spatial framework} \rightarrow \text{Attribute domain}(A_i)$$

map the spatial framework F onto different attribute domains A_i. The choice of the different field functions and the attribute domains is determined by the spatial application at hand. In the `Forest` example we had a single-field function, a step function, and the

CHAPTER 2

Spatial Concepts and Data Models

This chapter presents techniques related to the modeling of spatial database applications. GIS is arguably the most popular spatial database application, and our discussion reflects this fact. Besides GIS, other applications with a definite spatial or geometric component, such as CAD and astronomy, can also benefit from the techniques described in this chapter.

Traditional database concerns have typically been dominated by business and administrative applications. There the focus had been to efficiently (and securely) process large numbers of relatively simple transactions. Falling prices of data-capturing devices such as the Global Positioning System (GPS), the easy availability of satellite and cartographic data over the Internet, and the increasing power of desktop computing have redefined the functionality of a database. A DBMS is no longer considered a distinct, isolated repository of data but an active component of a multisystem computing environment. In fact, the trend is to shift computationally intensive tasks directly to the DBMS. A GIS is a premier example of this trend.

An SDBMS integrates the special needs of modeling spatial data into the system. This is a nontrivial task because

- Spatial data is relatively more complex compared with traditional business data, and the old database constructs are not adequate for handling it.
- Database design and implementation are typically handled by computer scientists, while the handling of spatial data falls in the domain of geographers, environmentalists, and other physical scientists. Traditionally, these disciplines have evolved along different trajectories.

The remainder of the chapter is organized as follows. In Section 2.1 we describe the different models for processing spatial information. In Section 2.2 we provide general database design and modeling principles. Section 2.3 is devoted to the extensions of the entity-relationship (ER) model for SDBs, and Section 2.4 is devoted to the Unified

Modeling Language (UML), a conceptual model primarily designed for object-oriented databases. We close with a summary of the chapter.

2.1 MODELS OF SPATIAL INFORMATION

We introduce an example, `State-Park` SDB which will be used to illustrate the different concepts in spatial data modeling. A State-Park SDB consists of `forests` which are a collections of `forest-stands` that correspond to different species of trees. A State-Park is accessed by `roads` and has a `manager`. There are `fire-Stations` within a state-park which are exclusively responsible for monitoring and managing fires in the State-Park. The State-Park is also dotted with `facilities` such as camping groups and offices. Finally, there are `rivers` which pass through the state park and also supply water to the different facilities.

Models of spatial information are usually grouped into two broad categories: *field* and *object*. We illustrate this dichotomy with the help of an example shown in Figure 2.1. Consider an idealized forest partitioned into homogeneous areas, each with one (dominant) tree species. In this example there are three species, namely, `Fir`, `Oak`, and `Pine`.

There are two complementary ways of modeling the forest. In the *functional* viewpoint, the forest is modeled as a function where the *domain* is the underlying geographic space of the forest and the *range* is a set consisting of three elements—the names of the tree species. The function, say f, maps each point of the space occupied by the forest into exactly one element of the range. The function f is a step-function—constant where trees are alike and changing to a different value where the tree species changes. In GIS the functional model is called the *Field* model. In Figure 2.1(c), the field representation of the forest is shown using a piece-wise function. A different field representation can be defined on a grid with each cell specifying the name of the dominant tree species. This second representation may be preferred if the boundaries of forest-stand are highly irregular. Other field representation include iso-lines showing contours with fixed values of a physical variable, for example, temperature, pressure.

Now consider the places where the function f changes values. In an idealized setting where the demarcation between the tree species is clearly defined, we should get the boundaries of polygons. Each polygon can be assigned a unique identifier and a nonspatial attribute—the name of the tree species. Thus the forest can be modeled as a collection of polygons (i.e., forest-stands) each corresponding to a tree species. This is the *Object* viewpoint and is shown in Figure 2.1(b).

The decision to model a spatial application using a field or an object paradigm is based on the requirements of the application and tradition. Amorphous phenomena, for example, fire, flood, hazardous spills, are naturally modeled as fields the boundaries are fluid. Other spatial phenomena can also be modeled using fields. Field models are often used to model continuous spatial trends such as elevation, temperature, and soil variation. In fact, in the area of remote-sensing, where the earth's surface is mapped from satellite- and aircraft-based sensors, the field model is dominant. Object models are more common in modeling transportation networks (e.g., roads), land parcels for property tax, and legal ownership-related applications. It has been postulated that the object model may have originated from societal needs to demarcate ownerships viewing land as property [Couclelis, 1992].

1.3.2 Spatial data models are covered in [Egenhofer, 1991b; Worboys, 1995] and [Laurini and Thompson, 1992; Price et al., 2000].

1.3.3 Extension of SQL for SDBs is discussed in [Egenhofer, 1994]. The OGIS specification for extending SQL for geospatial applications is described in [OpenGIS, 1998].

1.3.4 Representation of spatial data is covered extensively in [Laurini and Thompson, 1992; Worboys, 1995]. The filter-refine paradigm is discussed in [Chrisman, 1997].

1.3.5 The standard reference in spatial indexing is [Samet, 1990]. An up-to-date overview is in [Gaede and Gunther, 1998].

1.3.8 Issues related to the architectural design of SDBs are discussed in [Adam and Gangopadhyay, 1997; Worboys, 1995].

EXERCISES

Discussion

1. Discuss the differences between spatial and nonspatial data.
2. Geographic applications are a common source of spatial data. List at least four other important sources of spatial data.
3. What are the advantages of storing spatial data in a DBMS as opposed to a file system?
4. How can object-relational databases be used to implement an SDBMS?
5. List the differences and similarities between *spatial*, *CAD*, and *image* databases?
6. The interplay between the vector and raster data models has often been compared with the wave-particle duality in physics. Discuss.
7. Cognitive maps are described as "internal representations of the world and its spatial properties stored in memory." Do humans represent spatial objects as discrete entities in their minds? Is there a built-in bias against raster and in favor of vector representation?
8. Sorting is a popular method to access "traditional data" rapidly. Why is it difficult to sort spatial data?
9. Predicting global climate change has been identified as a premier challenge for the scientific community. How can SDBMS contribute in this endeavor?
10. E-commerce retailers reach customers all over the world via the Internet from a single facility. Some of them claim that geography and location are irrelevant in the Internet age. Do you agree? Justify your answer.
11. Define location-based and location independent services. Provide examples.
12. XML is emerging as a popular data interchange model on internet applications. Discusses the research impact of XML on spatial databases.
13. Compare and contrast:
    - **(i)** GIS vs. SDBMS
    - **(ii)** OODBMS vs. ORDBMS
    - **(iii)** GI Systems vs. GI Services
    - **(iv)** Data model vs. Query language
    - **(v)** Query processing vs. File organization and indices
    - **(vi)** Main memory vs. Disk
    - **(vii)** Querying vs. Data mining
14. Define spatial databases. List a few applications of spatial databases beyond those listed in this chapter.

1. By designing algorithms that assume that the data sets are large or by designing general scaling tools that can be coupled with data-mining algorithms
2. By extending SQL with "mining" functionality, so that mining tools are callable from within SQL

Estimates showing that up to 80 percent of the data in digital form is actually spatial data have spawned efforts to invent mining techniques that take the special nature of spatial data into consideration. For example, random sampling is a common technique applied in data mining to cut down on the size of the data set being analyzed without significant loss of information. Traditional methods of sampling are not effective in the case of spatial data because of *spatial autocorrelation*. Thus techniques from spatial statistics should be incorporated into mining algorithms to deal with spatial data.

In GIS, data mining is not new. Map generalization and classification of remote sensing images are relatively mature fields and are clearly related to data mining. Famous historical examples of spatial data mining include identification of the contaminated water source responsible for the 1854 cholera epidemic in London [Griffith, 1999], as well as identification of fluoride in drinking water for good dental health. In Chapter 7 we make an effort to bridge this gap.

1.7 SUMMARY

Spatial data management is of use in many disciplines, including geography, remote sensing, urban planning, and natural resource management. Spatial database management plays an important role in the solution of grand challenge scientific problems such as global climate change and genomics.

There are three classes of users who can benefit from spatial database management systems. Business users are interested in using spatial data to augment other forms of information while deciding about marketing campaigns, distribution centers, and retail locations. For a business user, spatial data is an important source of secondary information.

For scientific users who specialize in the study of the environment, natural resources, and geography, spatial data is of paramount importance. Users in this group have been using GIS products for the analysis of spatial data, but with the size of data sets growing rapidly, there is a need for specialized data management techniques that address the distinguishing properties of spatial data.

Finally there is a third class of users who want to use spatial data to personalize their experience and interaction especially on the worldwide Web. For example, queries on a search engine can produce results that are spatially localized and more meaningful.

BIBLIOGRAPHIC NOTES

1.1 For a history of database technology and a general introduction to DBMS, see [Ramakrishnan, 1998; Elmasri and Navathe, 2000; Silberschatz et al., 1997]. For an overview of the shortcomings of RDBMS in handling spatial data and a general introduction to object-relational DBMS, see [Stonebraker and Moore, 1997].

1.2 For an overview of SDBs, see [Scholl et al., 2001; Shekhar et al., 1999a] and the special issue of very large databases (VLDB) journal and the overview in [Guting, 1994a] in particular.

1.3 For integrating a SDB into off-the-shelf databases systems for CAD applications, see [Kriegel et al., 2001]. For an example of SDBMS, see [de La Beaujardiere et al., 2000].

ISBN 978-81-317-2628-0

First Impression, 2009

This edition is manufactured in India and is authorized for sale only in India, Bangladesh, Bhutan, Pakistan, Nepal, Sri Lanka and the Maldives. Circulation of this edition outside of these territories is UNAUTHORIZED.

Published by Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education in South Asia.

Head Office: 482, F.I.E., Patparganj, Delhi 110 092, India.
Registered Office: 14 Local Shopping Centre, Panchsheel Park, New Delhi 110 017, India.

Printed in India by Sai Print O Pack.

Spatial Databases: A Tour

Shashi Shekhar
Sanjay Chawla

An Alan R. Apt Book

PEARSON
Education

Contents

List of Figures

List of Tables

This book evolved from the class notes of a graduate course on Scientific Databases (Csci 8705) in University of Minnesota. Researchers and students both within and outside the Computer Science Department found the course very useful and applicable to their work. Despite the good response and high level of interest in the topic, no textbook available in the market was able to meet the interdisciplinary needs of the audience. A recent book by [Scholl et al., 2001] focuses on traditional topics related to query languages and access methods while leaving out current topics such as spatial networks (e.g., road maps) and data mining for spatial patterns. A monograph by [Adam and Gangopadhyay, 1997] catalogs research papers on database issues in GIS, with little reference to the industrial state-of-the-art. Another [Worboys, 1995] also focuses on GIS and has only two chapters devoted to database issues. Many of these books neither use industry standards, e.g., OGIS, nor provide adequate instructional support, example, questions and problems at the end of each chapter to allow students to assess their understanding of the main concepts. Not surprisingly, our colleagues in academia working in databases, parallel computing, multimedia information, civil and mechanical engineering, and forestry have expressed a strong desire for a comprehensive text on spatial databases. Industry professionals involved in software development for GIS and CAD/CAM have also made several requests for information on spatial databases in a collected form.

As a first step toward developing this book, we completed a survey paper, "Spatial Databases: Accomplishments and Research Needs," for IEEE Transactions on Knowledge and Data Engineering (Jan. 1999). We noticed that the research literature in computer science was skewed with numerous publications on some topics (e.g., spatial indexes, spatial-join algorithms) and relatively scarce publications on many other important topics (e.g., conceptual modeling of spatial data). We looked to GIS industry as well as GIS researchers outside computer science for ideas in these areas for the relevant chapters in the book.

Being on sabbatical for the academic year 1997–1998 facilitated initial work on the book. We completed a draft of many chapters by expanding on the course notes of CSci 8705. The first draft of the book was used in database courses at the University of Minnesota. Subsequently the book was revised using feedback from reviewers, colleagues, and students.

We believe the following features are unique aspects of this book:

- The aim of this book is to provide a comprehensive overview of spatial databases management systems. It covers current topics, example, spatial networks and spatial data mining in addition to traditional topics, example, query languages, indexing, query processing.
- A set of questions and problems is provided in each chapter to allow readers to test understanding of core concepts, to apply core concepts in new application domains, and to think beyond the material presented in the chapter. Additional instructional aids (e.g., laboratories, lecture notes) are planned on the book Web site.
- A concerted effort has been made to "look beyond GIS." Techniques from SDBMS are finding applications in many diverse areas, including multimedia information, CAD/CAM, astronomy, meteorology, molecular biology, and computational mechanics.

spatial datablades. We are thankful to Alan Apt and his wonderful staff at Prentice Hall for helping with the process of book-writing with constant encouragement. The presentation improved a great deal from the comments from the anonymous reviewers and we are grateful to them.

Our research related to spatial databases has been supported by many organizations including the United Nations Development Programme, the National Science Foundation, the National Aeronautics and Space Agency, the Army Research Laboratories, the U.S. Department of Agriculture, the Federal Highway Administration, the US Department of Transportation, the Minnesota Department of Transportation, the Center for Urban and Regional Affairs, and the Computing Devices International. Many of the research project resulted in surveys of research literature, as well as in development of innovative techniques for problems related to spatial databases.

Special thanks to the members of spatial database research group in the Computer Science Department at the University of Minnesota. They contributed in many different ways including literature surveys, development of examples and figures, and insights into various methods, as well in developing proper problem formulations and innovative solution methods. We are extremely grateful to Vatsavai Ranga Raju for careful review and multiple improvements to the earlier drafts. We also thank the students in different offerings of Csci 8701, and Csci 8705 for working with the earlier drafts of the books and providing helpful suggestions towards revising the material.

We benefitted from discussions with many other people over the years. They include Marvin Bauer, Yvan Bedard, Paul Bolstad, Nick Bourbakis, Thomas Burk, John Carlis, Jai Chakrapani, Vladimir Cherkassky, Douglas Chub, William Craig, Max Donath, Phil Emmerman, Max Egenhofer, Michael Goodchild, Ralf Hartmut Gueting, Oliver Gunther, John Gurney, Jia-Wei Han, Ravi Janardan, George Karypis, Hans-Peter Kriegel, Robert McMaster, Robert Pierre, Shamkant Navathe, Raymond Ng, Hanan Samet, Paul Schrater, Jaideep Srivastava, Benjamin Wah, Kyu-Young Whang, and Michael Worboys.

- In each chapter we have tried to bring out the object-relational database framework, which is a clear trend in commercial database applications. This framework allows spatial databases to reuse relational database facilities when possible, while extending relational database facilities as needed.
- We will use spatial data types and operations specified by standards (e.g., OGIS) to illustrate common spatial database queries. These standards can be incorporated into an object-relational query language such as SQL-3.
- Self-contained treatment without assuming pre-requisite knowledge of GIS or Databases.
- Complete coverage of spatial networks including modeling, querying and storage methods.
- Detailed discussion of issues related to Spatial Data Mining.
- Balance between cutting-edge research and commercial trends.
- Easy to understand examples from common knowledge application domains.

CHAPTER ORGANIZATION

The book is divided into eight chapters, each one an important subarea within spatial databases. We introduce the field of spatial databases in Chapter 1. In Chapter 2, we focus on spatial data models and introduce the field versus object dichotomy and its implications for database design. Chapter 3 discusses the necessary enhancements required to make traditional query languages compatible with spatial databases. We provide an extensive discussion of various proposals to extend SQL with spatial capabilities. Spatial databases deal with extraordinarily large amounts of data, and it is essential for DBMS to provide sophisticated storage, compression, and indexing methods to enhance the performance of query processing. Spatial data storage and indexing schemes are covered in Chapter 4. From query languages and indexing, we move on to query processing and optimization in Chapter 5. Here we discover that many standard techniques from traditional databases have to be abandoned or drastically modified in order to be applicable in a spatial context. We also introduce the filter-refine paradigm for spatial query processing. In Chapter 6, we show how spatial database technology is being applied to spatial networks. In this chapter we also cover network data models and query languages. Chapter 7 covers the emerging field of spatial data mining. In this chapter we expose the readers to the concept of spatial dependency that is prevalent in spatial data sets, and show how this can be modeled and incorporated into data mining process. Finally in Chapter 8, we discuss emerging trends in spatial databases.

ACKNOWLEDGMENTS

We have received help from many people and we are extremely grateful to them. This book would not have started without encouragement from Professor Vipin Kumar, Computer Science Department, University of Minnesota; and Dr. Jack Dangermond, President, Environmental Systems Research Institute (ESRI). This book benefitted a great deal from access to ESRI researchers and products. We are also grateful to Dr. Siva Ravada (Oracle Corporation) and Dr. Robert Uleman (Illustra Inc.) for help with understanding of their

Foreword

Ever since Cro-Magnon hunters drew pictures of track lines and tallies thought to depict migration routes on the walls of caves near Lascaux, France, 35,000 years ago, people have been interested in graphics linked to geographic information. Today, such geographic information systems (GISs) are used in applications ranging from tracking the migration routes of caribou and polar bears, identifying the effects of oil development on animal life, aiding farmers to minimize the use of pesticide in their farms, helping corporate supply managers to predict the best places to build distribution warehouses, to relating information about rainfall and aerial photographs to wetlands drying up at certain times of the year.

A GIS, in the strictest sense, is a computer system for assembling, storing, manipulating, and displaying data with respect to their locations. However, modern GISs often assimilate data from multiple disparate sources in many different forms in order to answer queries and help analyze information. In a broad sense, a GIS not only converts and stores geographical information in digital form for analysis, but must also collects, transforms, aggregates, indexes, links, and mines related spatial databases. A modern GIS makes it possible to integrate information that is difficult to associate through any other means and combines mapped variables to build and analyze new variables.

With this broad perspective in mind, Shekhar and Chawla have done a marvelous job in presenting the fundamentals and trends in geographical information processing. The core of the book is a tour de force sequence of concepts and methods, progressively explaining models, languages and algorithms until we distinguish branches from trees and trees from forests. The authors not only explain the concepts but illustrate them well by numerous examples. They have emphasized the many nontrivial issues in integrating spatial data into traditional databases, ranging from deep ontological questions about the modeling of space to the important issues about file management. Each chapter is further supplemented by many thought-provoking exercises that aid readers in better understanding of the concepts and algorithms presented. The book ends with an excellent exposition of spatial data mining and future trends in spatial databases that helps readers appreciate emerging research issues.

This book is suitable as a textbook for an interdisciplinary course on geographic information systems, as well as a handy reference for people working in the area. Readers should find it easy to understand and apply the concepts and algorithms learned, even without any formal training in databases. Many disciplines will benefit from techniques learned in this book, leading to wider applications of the technology throughout government, business, and industry.

As the reader of the first books in this area, and I am confident that you will benefit by what you learn in this exciting and rewarding area.

Prof. Benjamin Wah
University of Illinois at Urbana-Champaign
Department of Electrical and Computer Engineering
1101 W. Springfield Avenue
Urbana, IL 61801
President, IEEE Computer Society (2001)
March 2002

Preface

Over the years it has become evident in many areas of computer applications that the functionality of database management systems has to be enlarged to include spatially referenced data. The study of spatial database management systems (SDBMS) is a step in the direction of providing models and algorithms for the efficient handling of data related to space.

Spatial databases have now been an active area of research for over two decades. Their results, example, spatial multidimensional indexing, are being used in many different areas. The principle impetus for research in SDBMs comes from the needs of existing applications such as geographical information systems (GIS) and computer-aided design (CAD), as well as potential applications such as multimedia information systems, data warehousing, and NASA's earth observation system. These spatial applications have over one million existing users.

Major players in the commercial database industry have products specifically designed to handle spatial data. These products include the spatial data engine (SDE), by Environment Systems Research Institute (ESRI); as well as spatial datablades for object-relational database servers from many vendors including Intergraph, Autodesk, Oracle, IBM and Informix. Research prototypes include Postgres, Geo2, and Paradise. The functionality provided by these systems includes a set of spatial data types such as the point, line, and polygon, and a set of spatial operations, including intersection, enclosure, and distance. An industrywide standard set of spatial data types and operations has been developed by the Open Geographic Information Systems (OGIS) consortium. The spatial types and operations can be made a part of an object-relational query language such as SQL3. The performance enhancement provided by these systems includes a multidimensional spatial index and algorithms for spatial access methods, spatial range queries, and spatial joins.

The integration of spatial data into traditional databases amounts to resolving many nontrivial issues at various levels. They range from deep ontological questions about the modeling of space, for example, whether it should be field based or object based, thus paralleling the wave-particle duality in physics to more mundane but important issues about file management. These diverse topics make research in SDBMS truly interdisciplinary.

Let us use the example of a country dataset to highlight the special needs of spatial databases. A country has at least one nonspatial datum, its name, and one spatial datum, its boundary. There is no ambiguity about storing or representing its name, but unfortunately it is not true for its boundary. Assuming that the boundary is represented as a collection of straight lines, we need to include a spatial data type *line* and the companion types *point* and *region* in the database system to facilitate spatial queries on the object *country*. These new data types need to be manipulated and composed according to some fixed rules leading to the creation of a spatial algebra. Because spatial data is inherently visual and usually voluminous, database systems have to be augmented to provide visual query processing and special spatial indexing . Other important database issues such as concurrency control, bulk loading, storage, and security have to be revisited and fine-tuned to build an effective spatial database management system.

Foreword

Spatial information—that is, information about objects located in a spatial frame such as the Earth's surface—has long been recognized as presenting special problems for computing. As early as 1972 the term *spatial data handling* was being used to refer to the activities of a small but dynamic community of researchers who were committed to exploiting electronic data processing in order to increase productivity in such areas as map compilation and editing; map measurement; and spatial data analysis. Spatial information is rich in high-level structures, and although each of the classical models of database management that began to emerge in the 1960s had something to offer to this application area, neither the relational model nor object-oriented modeling provide a perfect fit. The relational model handles topological relationships well, but lacks the means to represent complex hierarchical relationships that span spatial scales; while object-oriented models handle both topological and hierarchical relationships, but have difficulty dealing with phenomena that are essentially continuous in space.

Books that elucidate the complex story of spatial databases are few and far between, and thus this book is especially welcome. It covers the entire field, from representation through query to analysis, in a style that is clear, logical, and rigorous. Especially welcome is the chapter on data mining, which addresses both traditional spatial data analysis and also new techniques that have been developed in the past few years to take advantage of today's high-speed computing in processes of automated search for anomalies and patterns in very large spatial databases. It is intended for computer science students, as reflected in the sequence of topics and the style of presentation, but will also be useful for students from other disciplines looking for a more rigorous and fundamental approach than is provided by most textbooks on geographic information systems (GIS).

The importance of spatial databases is growing rapidly, partly as a result of the recognition that their applications extend well beyond the traditional domain of GIS. Location and time are powerful ways of identifying and characterizing information, because many data sets have *footprints* in space and time. This is obviously true of maps and Earth images, but is also true of many reports, books, photographs, and other types of information. Thus location is a powerful basis for search, and for finding relevant information in distributed resources such as the World Wide Web. There is growing recognition that space (and time) provides important ways of integrating information that go far beyond the traditional domains of spatial databases and GIS. The last chapter of the book discusses some of these, and gives a sense of why many believe that the importance of spatial databases is likely to continue to grow rapidly over the coming years.

Michael F. Goodchild
National Center for Geographic Information and Analysis,
and Department of Geography,
University of California, Santa Barbara, CA 93106-4060, USA

PEARSON Education

Spatial Databases

A TOUR

Shashi Shekhar · Sanjay Chawla

CHAPTER 1

Introduction to Spatial Databases

1.1 OVERVIEW

We are in the midst of an information revolution. The raw material (data) which is powering this controlled upheaval is not found below the earth's surface where it has taken million of years to form but is being gathered constantly via sensors and other data-gathering devices. For example, NASA's Earth Observing System (EOS) generates one terabyte of data every day.

Satellite images are one prominent example of spatial data. To extract information from a satellite image, the data has to be processed with respect to a spatial frame of reference, possibly the earth's surface. But satellites are not the only source of spatial data, and the earth's surface is not the only frame of reference. A silicon chip can be, and often is, a frame of reference. In medical imaging the human body acts as a spatial frame of reference. In fact, even a supermarket transaction is an example of spatial data if, for example, a zip code is included. Queries, or commands, posed on spatial data are called spatial queries. For example, the query "What are the names of all bookstores with more than ten thousand titles?" is an example of a nonspatial query. On the other hand, the query, "What are the names of all bookstores within ten miles of the Minneapolis downtown?" is an example of a spatial query. This book is an exposition of efficient techniques for the storage, management, and retrieval of spatial data.

Today, data is housed in and managed via a database management system (DBMS). Databases and the software that manages them are the silent success story of the information age. They have slowly permeated all aspects of daily living, and modern society would come to a halt without them. Despite their spectacular success, the prevalent view is that a majority of the DBMSs in existence today are either incapable of managing spatial data or are not user-friendly when doing so. Now, why is that? The traditional

role of a DBMS has been that of a simple but effective warehouse of business and accounting data. Information about employees, suppliers, customers, and products can be safely stored and efficiently retrieved through a DBMS. The set of likely queries is limited, and the database is organized to efficiently answer these queries. From the business world, the DBMS made a painless migration into government agencies and academic administrations.

Data residing in these mammoth databases is simple, consisting of numbers, names, addresses, product descriptions, and so on. These DBMSs are very efficient for the tasks they were designed for. For example, a query such as "List the top ten customers, in terms of sales, in the year 1998" will be very efficiently answered by a DBMS even if it has to scan through a very large customer database. Such commands are conventionally called "queries" although they are not questions. The database will not scan through all the customers; it will use an index, as you would do with this book, to narrow down the search. On the other hand, a relatively simple query such as "List all the customers who reside within fifty miles of the company headquarters" will confound the database. To process this query, the database will have to transform the company headquarters and customer addresses into a suitable reference system, possibly latitude and longitude, in which distances can be computed and compared. Then the database will have to scan through the entire customer list, compute the distance between the company and the customer, and if this distance is less than fifty miles, save the customer's name. It will not be able to use an index to narrow down the search, because traditional indices are incapable of ordering multidimensional coordinate data. A simple and legitimate business query thus can send a DBMS into a hopeless tailspin. Therefore the need for databases tailored for handling spatial data and spatial queries is immediate.

1.2 WHO CAN BENEFIT FROM SPATIAL DATA MANAGEMENT?

Professionals from all walks of life have to deal with the management and analysis of spatial data. Below we give a small but diverse list of professionals and one example of a spatial query relevant to the work of each.

Mobile phone user: Where is the nearest gas station? Is there a pet-food vendor on my way home?

Army field commander: Has there been any significant enemy troop movement since last night?

Insurance risk manager: Which houses on the Mississippi River are most likely to be affected by the next great flood?

Medical doctor: Based on this patient's Magnetic Resonance Imaging (MRI), have we treated somebody with a similar condition?

Molecular biologist: Is the topology of the amino acid biosynthesis gene in the genome found in any other sequence feature map in the database?

Astronomer: Find all blue galaxies within two arcmin of quasars.

Climatologist: How can I test and verify my new global warming model?

a GIS can list neighboring countries of a given country (e.g., France) given the political boundaries of all countries. However it will be fairly tedious to answer set queries such as, *list the countries with the highest number of neighboring countries* or *list countries which are completely surrounded by another country.* Set-based queries can be answered in an SDBMS, as we show in Chapter 3.

SDBMSs are also designed to handle very large amounts of spatial data stored on secondary devices (e.g., magnetic disks, CD-ROM, jukeboxes, etc.), using specialized indices and query-processing techniques. Finally, SDBMSs inherit the traditional DBMS functionality of providing a concurrency-control mechanism to allow multiple users to simultaneously access shared spatial data, while preserving the consistency of that data.

A GIS can be built as the front-end of an SDBMS. Before a GIS can carry out any analysis of spatial data, it accesses that data from an SDBMS. Thus an efficient SDBMS can greatly increase the efficiency and productivity of a GIS.

1.4 THREE CLASSES OF USERS FOR SPATIAL DATABASES

The use and management of spatial data to enhance productivity has now been embraced by scientists, administrators, and business professionals alike. Equally important, the concept that spatial or geographic or geospatial data has to be treated differently, compared with other forms of data, has slowly permeated the thinking and planning of all major database vendors. As a result, specialized spatial products have appeared in the marketplace to enhance the spatial capabilities of generic DBMSs. For example, spatial attachments, with esoteric names such as *cartridge*, *datablade*, or more benign names such as *spatial option* have been introduced by Oracle, Informix, and IBM, respectively.

From a major database vendor perspective, there is a demand for specialized products for the management of spatial data, but this is obviously not the only form of data that businesses use. In fact, it is not the only form of specialized data. For example, besides spatial attachments, database vendors have released attachments for *temporal*, *visual*, and other multimedia forms of data.

On the other hand, GIS vendors have targeted users who are exclusively focused on the analysis of spatial data. This market by definition is relatively narrow and consists of specialists in the scientific community and/or government departments. Users of GIS typically work in an isolated environment vis-à-vis other information technology users and access specialized databases specifically designed for them. In order to manage the ever-increasing volume of spatial (and nonspatial) data and link to commercial databases, GIS vendors have introduced middleware products, such as ESRI's Spatial Data Engine. The focus in the GIS community has evolved in the last decade, as shown in Figure 1.1. GIS, which began as a software system for representing geographic information in a layered fashion, attained the status of Geographic Information (GI) science by focusing on the *algebra* of map and spatial operations. With the prominence of the personal computing, the focus has again shifted to providing geographic or spatial services over the personal computers. The prime examples are the spatially sensitive search engines and map facilities in MS Office 2000 to augment the classical spreadsheet and database tools.

With the advent of the Internet, there is another group of users who want to use spatial data but only at a very high and user friendly level. For example, one of the more popular sites on the Internet provides direction maps to visitors. Another site provides a spatially sensitive search engine. Queries such as "Find all Mexican restaurants in downtown Minneapolis" can be answered by such search engines. Another promising

Pharmaceutical Researcher: Which molecules can dock with a given molecule based on geometric shapes?

Sports: Which seats in a baseball stadium provide best view of pitcher and hitter? Where should TV camera be mounted?

Corporate supply manager: Given trends about our future customer profile, which are the best places to build distribution warehouses and retail stores?

Transport specialist: How should the road network be expanded to minimize traffic congestion?

Urban sprawl specialist: Is new urban land development leading to the loss of rich agricultural land.

Ski resort owner: Which mountains on our property are ideal for a beginner's ski run?

Farmer: How can I minimize the use of pesticide on my farm?

Golf entrepreneur: Where do I build a new golf course which will maximize profit given the constraints of weather, Environmental Protection Agency pesticide regulations, the Endangered Species Act, property prices, and the neighborhood demographic profile.

Emergency service: Where is the person calling for help located? What is the best route to reach her?

1.3 GIS AND SDBMS

The Geographic Information System (GIS) is the principal technology motivating interest in Spatial Database Management Systems (SDBMSs). GIS provides a convenient mechanism for the analysis and visualization of geographic data. Geographic data is spatial data whose underlying frame of reference is the earth's surface. The GIS provides a rich set of analysis functions which allow a user to affect powerful transformations on geographic data. The rich array of techniques that geographers have packed into the GIS are the reasons behind its phenomenonal growth and multidisciplinary applications. Table 1.1 lists a small sample of common GIS operations.

A GIS provides a rich set of operations over few objects and layers, whereas an SDBMS provides simpler operations on a set of objects and sets of layers. For example,

TABLE 1.1: List of Common GIS Analysis Operations [Albrecht, 1998]

Search	Thematic search, search by region, (re-)classification
Location analysis	Buffer, corridor, overlay
Terrain analysis	Slope/aspect, catchment, drainage network
Flow analysis	Connectivity, shortest path
Distribution	Change detection, proximity, nearest neighbor
Spatial analysis/Statistics	Pattern, centrality, autocorrelation, indices of similarity, topology: hole description
Measurements	Distance, perimeter, shape, adjacency, direction

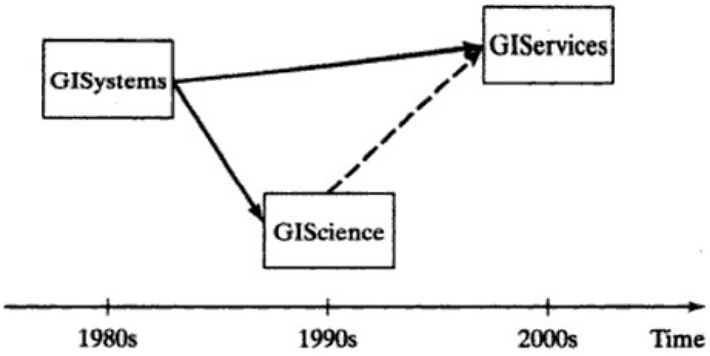


FIGURE 1.1. The evolution of the abbreviation GIS over the last two decades. In the 1980s GIS was Geographic Information System; in the 1990s, Geographic Information Science was the preferred phrase, and now the trend is toward Geographic Information Services.

use of spatial technologies is related to the location determination of mobile phones. The U.S. federal government has mandated that by October 2001, the location of all cell phones must be traceable to an accuracy of 125 meters, 67 percent of the time. Thus, although it is often remarked that the Internet revolution has "killed" the concept of geography, paradoxically, the demand for spatial technologies is continuously growing.

The recent increase in the use of personal digital assistants (PDAs), mobile phones, and two-way pagers has opened up several opportunities for new applications. Example applications are mobile workforces and location-based services. As the name suggests, the mobile workforce works from remote locations—at clients' locations, branch offices, and remote field locations. These workforces typically download a segment of data needed for a particular task, work with that segment at a remote location, and update (synchronize) their modifications with the master database at the end of the day. An important aspect of this scenario is that the client hoards data and works locally on the data in a disconnected fashion. An important trend in recent years is the availability of location-based services, that is, the services that actually change depending on the geographic location of the individual user. With the availability of global positioning systems (GPS), it is easy to precisely determine the location of any client/user and offer appropriate solutions based on the geographic location of the user. Example applications of location-based services are to locate a customer, find the nearest pizza place with a gas station on the way, display a road map for a new city with tourist locations highlighted, or location sensitive transactions and alerts. The impact and importance of the location-based services led the Open GIS Consortium (OGC) to initiate Open Location Services (OpenLS) with a vision to integrate geospatial data and geoprocessing resources into the location services and telecommunication infrastructure.

AN EXAMPLE OF AN SDBMS APPLICATION

Earlier we gave an example of a simple spatial query to illustrate the shortcomings of traditional databases. We now give a more detailed example. Figure 1.2 shows a Landsat Thematic Mapper (TM) image of Ramsey County, Minnesota. Overlaid on the image are the boundaries (thick black lines) of census blocks and the locations

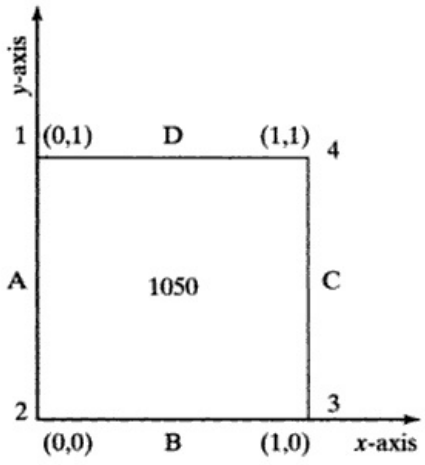
One *natural* way of storing information about the census blocks, for example, their name, geographic area, population, and boundaries, is to create the following table in the database:

```
create table census_blocks (
                                name        string ,
                                area        float,
                                population  number,
                                boundary    polyline );
```

In a (relational) database, all objects, entities, and concepts that have a distinct identity are represented as relations or tables. A relation is defined by a name and a list of distinguishing attributes that characterize the relation. All instances of the entity are stored as tuples in the table. In the preceding code fragment, we have created a table (relation) named *census_block*, which has four attributes: name, area, population, and boundary. At table creation time, the types of attributes have to be specified, and they are string, float, number, and polyline. Polyline is a datatype to represent a sequence of straight lines.

Figure 1.3 shows a hypothetical census block and how information about it can be stored in a table. Unfortunately, such a table is not natural for a traditional relational database because polyline is not a built-in datatype. One way to circumvent this problem is to create a collection of tables with overlapping attributes, as shown in Figure 1.4. Another way is to use a stored procedure. For a novice user, these implementations are quite complex. The key point is that the census block data cannot be naturally mapped onto a relational database. We need more constructs to handle spatial information in order to reduce the semantic gap between the user's view of spatial data and the database implementation. Such facilities are offered by the object-oriented software paradigm.

The object-oriented software paradigm is based on the principles of user-defined datatypes, along with inheritance and polymorphism. The popularity of such languages as C++, Java, and Visual Basic is an indicator that object-oriented concepts are firmly



Census_blocks

Name	Area	Population	Boundary
1050	1	1839	Polyline((0,0),(0,1),(1,1),(1,0))

FIGURE 1.3. Census blocks with boundary ID:1050.

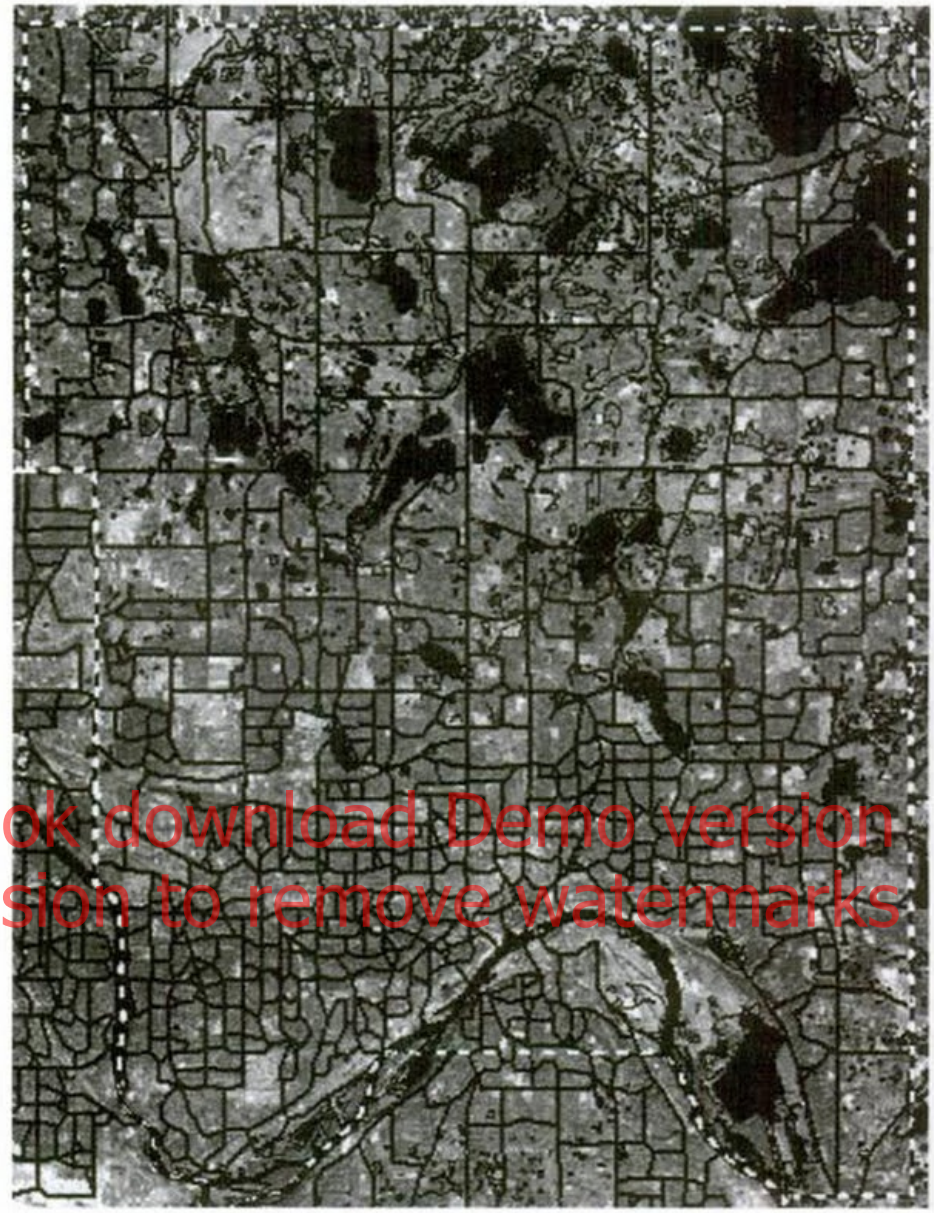


FIGURE 1.2. Landsat image of Ramsey County, Minnesota, with spatial layers of information superimposed.

of wetlands (thin black lines). From the image we can easily identify several lakes (dark patches in north) and the Mississippi river (south). This county covers about 156 square miles and covers most of the urban and suburban regions of St. Paul. This figure was created in ArcView, a popular GIS software program. A typical spatial database consists of several images and vector layers like land parcels, transportation, ecological regions, soils, etc. Here we have four layers: the basic image, a layer of census blocks, another for wetlands, and finally the layer of county boundaries (white dashed lines).

Census_blocks

Name	Area	Population	boundary-ID
340	1	1839	1050

Polygon

boundary-ID	edge-name
1050	A
1050	B
1050	C
1050	D

Edge

edge-name	endpoint
A	1
A	2
B	2
B	3
C	3
C	4
D	4
D	1

Point

endpoint	*x*-coor	*y*-coor
1	0	1
2	0	0
3	1	0
4	1	1

FIGURE 1.4. Four tables required in a relational database with overlapping attributes to accommodate the polyline datatype.

established in the software industry. It would seem that our land parcel problem is a natural application of object-oriented design: Declare a class polyline and another class land_parcel with attribute *address*, which is a string type, and another attribute *boundary* which is of the type *polyline*. We do not even need an attribute *area* because we can define a method *area* in the polyline class which will compute the area of any land parcel on demand. So will that solve the problem? Are object-oriented databases (OODBMS) the answer? Well, not quite.

The debate between relational versus object-oriented within the database community parallels the debate between vector versus raster in GISs. The introduction of abstract data types (ADTs) clearly adds flexibility to a DBMS, but there are two constraints peculiar to databases that need to be resolved before ADTs can be fully integrated into DBMSs.

- Market adoption of OODBMS products has been limited, despite the availability of such products for several years. This reduces the financial resources and engineering efforts to performance-tune OODBMS products. As a result, many GIS users will use systems other than OODBMS to manage their spatial data in the near future.
- SQL is the lingua franca of the database world, and it is tightly coupled with the relational database model. SQL is a declarative language, that is, the user only

specifies the desired result rather than the means of production. For example, in SQL the query "*Find all land parcels adjacent to MY_HOUSE.*" should be able to be specified as follows:

```
SELECT   M.address
FROM     land_parcel L, M
WHERE    Adjacent(L,M) AND
         L.address = 'MYHOUSE'
```

It is the responsibility of the DBMS to implement the operations specified in the query. In particular, the function *Adjacent*(L, M) should be callable from within SQL. The popular standard, SQL-92, supports user-defined functions and SQL-3/SQL 1999; the next revision supports ADTs and a host of data structures such as lists, sets, arrays, and bags. Relational databases that incorporate ADTs and other principles of object-oriented design are called object relational database management systems (OR-DBMS). The historical evolution of database technology is shown in Figure 1.5.

The current generation of OR-DBMSs offers a modular approach to ADTs. An ADT can be built into or deleted from the system without affecting the rest of the system. Although this "plug-in" approach opens up the DBMS for enhanced functionality, there is very little built-in support for the optimization of operations. Our focus is to specialize

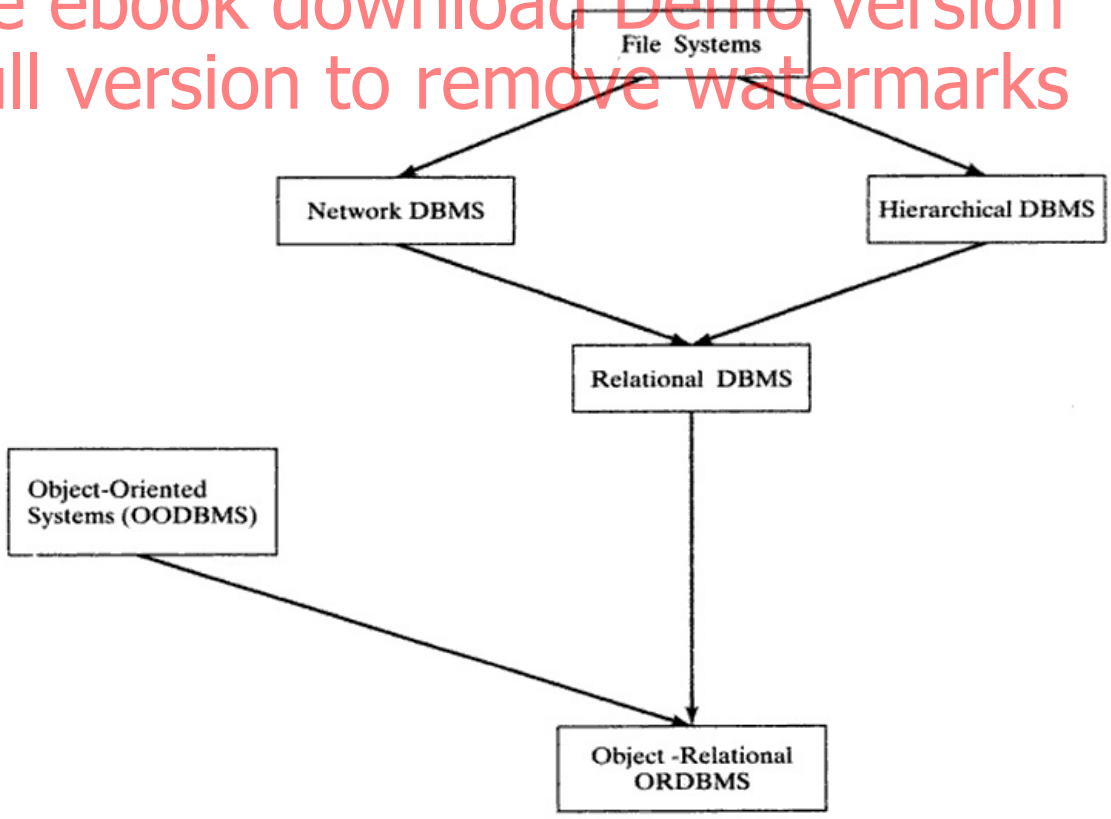


FIGURE 1.5. Evolution of databases. [Khoshafian and Baker, 1998]

an OR-DBMS to meet the requirements of spatial data. By doing so, we can extrapolate spatial domain knowledge to improve the overall efficiency of the system. We are now ready to give a definition of SDBMS for setting the scope of the book.

1. An SDBMS is a software module that can work with an underlying database management system, for example, OR-DBMS, OODBMS.
2. SDBMSs support multiple spatial data models, commensurate spatial abstract data types (ADTs), and a query language from which these ADTs are callable.
3. SDBMSs support spatial indexing, efficient algorithms for spatial operations, and domain-specific rules for query optimization.

Figure 1.6 shows a representation of an architecture to build an SDBMS on top of an OR-DBMS. This is a three-layer architecture. The top layer (from left to right) is the spatial application, such as GIS, MMIS (multimedia information system), or CAD (computer-aided design). This application layer does not interact directly with the OR-DBMS but goes through a middle layer which we have labeled spatial database (SDB). The middle layer is where most of the available spatial domain knowledge is encapsulated, and this layer is "plugged" into the OR-DBMS. No wonder commercial OR-DBMS

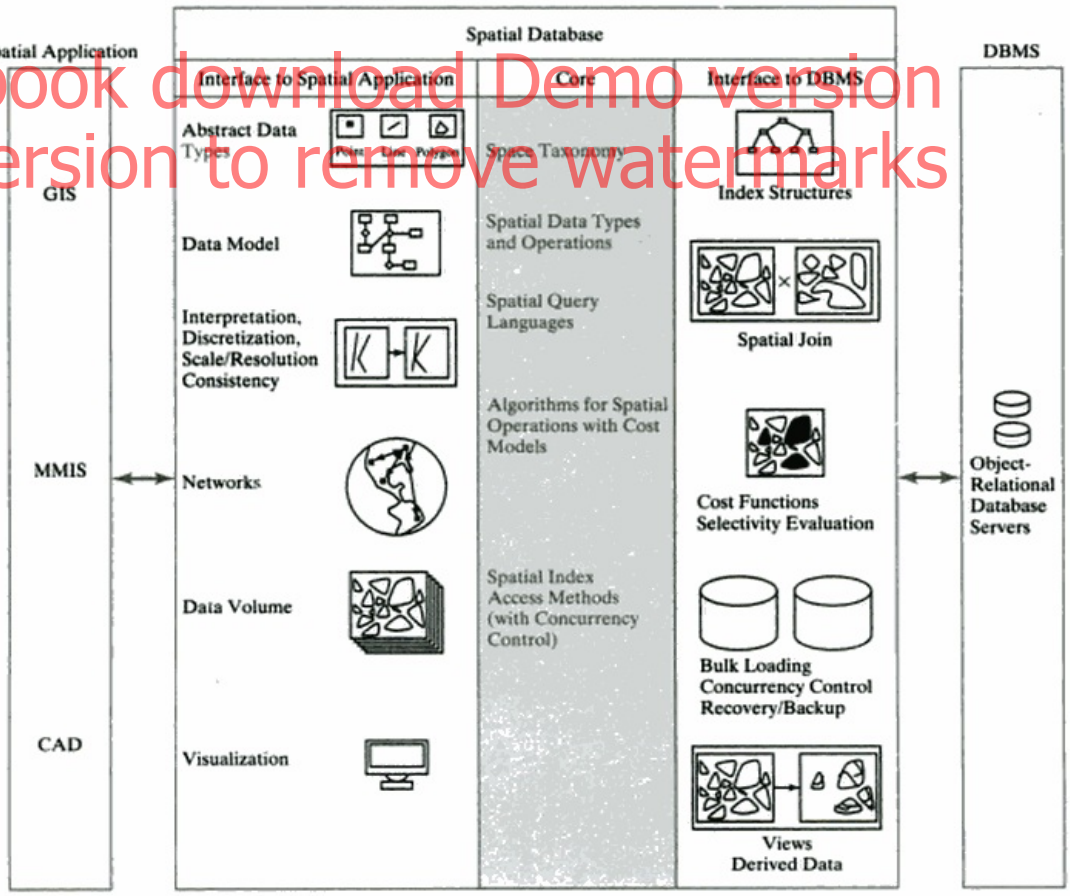


FIGURE 1.6. Three-layer architecture.

products have names such as Spatial Data Blade (Illustra), Spatial Data Cartridge (Oracle), and Spatial Data Engine (ESRI).

We close this section by recalling the core features that are essential for any DBMS to support but are implicit to a casual user:

1. *Persistence:* The ability to handle both transient and persistent data. Although transient data is lost after a program terminates, persistent data not only transcends program invocations, but also survives system and media crashes. Further, the DBMS ensures that a smooth recovery takes place after a crash. In database management systems, the state of the persistent object undergoes frequent changes, and it is sometimes desirable to have access to the previous data states.
2. *Transactions:* Transactions map a database from one consistent state to another. This mapping is atomic (i.e., it is executed completely or aborted). Typically, many transactions are executed concurrently, and the DBMS imposes a serializable order of execution. Consistency in a database is accomplished through the use of integrity constraints. All database states must satisfy these constraints to be deemed consistent. Furthermore, to maintain the security of the database, the scope of the transactions is dependent on the user's access privileges.

1.6 A STROLL THROUGH SPATIAL DATABASES

We closed the previous section by introducing a three-layer architecture to build a SDBMS on top of an OR-DBMS. We noted that most of the spatial domain knowledge is encapsulated in the middle layer. In this section, we briefly describe some of the core functionalities of the middle layer. This section can also be considered as a detailed overview of the rest of the book, or a "quick tutorial" on SDBMSs.

1.6.1 Space Taxonomy and Data Models

Space is indeed the final frontier, not only in terms of travel but also in its inability to be captured by a concise description. Consider the following refrain echoed by hapless drivers all over: "I don't remember how far Jack's house is. Once I am nearby, I might recall on which side of the street it lies, but I am certain that it is adjacent to a park." This sentence gives a glimpse of how our brain (mind) structures geographic space. We are terrible in estimating distances, maybe only slightly better in retaining direction and orientation, but (we are) fairly good when it comes to remembering *topological* relationships such as *adjacent*, *connected*, and *inside*.

Topology, a branch of mathematics, is exclusively devoted to the study of relationships which do not change due to elastic deformation of underlying space. For example, if there are two rectangles, one inside the other, or both adjacent to each other, drawn on a rubber sheet, and if it is stretched, twisted, or shrunk, the named relationships between the two rectangles will not change! Another clue about how our mind organizes space is to examine the language we speak: The shape of an object is a major determinant in an object's description. Is that the reason why we have trouble accepting a whale as a mammal and a sea horse as a fish? Objects are described by nouns, and we have as many nouns as different shapes. On the other hand, the spatial relationship between objects are described by prepositions and encode very weak descriptions of their shape. In the "Coffman Union is to the southeast of Vincent Hall," the shapes of the buildings

TABLE 1.2: Different Types of Spaces with Example Operations

Topological	Adjacent
Network	Shortest-path
Directional	North-of
Euclidean	Distance

play almost no role in the relationship "southeast." We could often replace the buildings with coarse rectangles without affecting the relationship.

Space taxonomy refers to the multitude of descriptions that are available to organize space: topological, network, directional, and Euclidean. Depending on why we are interested in modeling space in the first place, we can choose an appropriate spatial description. Table 1.2 shows one example of a spatial operation associated with a different model of space. It is important to realize that no single description (model) of space can answer all queries.

A data model is a rule or set of rules to identify and represent objects referenced by space. Minnesota is the "land of ten thousand lakes." How can these lakes be represented? An intuitive, direct way is to represent each lake as a two-dimensional region. Similarly a stream, depending on the scale, can be represented as a one-dimensional curved line, and a well-site by a zero-dimensional point. This is the *object* model. The object model is ideal for representing such non-amorphous spatial entities as lakes, road networks, and cities. The object model is conceptual; it is mapped onto the computer using the *vector* data structure. A vector data structure maps regions into polygons, lines into polylines, and points to points.

The *field* model is often used to represent continuous or amorphous concepts for example, the temperature or the clouds field. A field is a function that maps the underlying reference frame into an attribute domain. For temperature, popular attribute domains are Celsius and Fahrenheit. The *raster* data structure implements the *field* model on a computer. A raster data structure is a uniform grid imposed on the underlying space. Because field values are spatially autocorrelated (they are continuous), the value of each cell is typically the average of all the field points that lie within the cell. Other popular data structures for fields are TIN (triangulated irregular network), contour lines, and point grids. (More on this in Chapters 2 and 8.)

1.6.2 Query Language

From our discussion so far, it is obvious that the current functionality of relational query language (e.g., SQL2) has to be extended if it is to be a natural spatial query language. In particular, the ability to specify spatial ADT attributes and methods from within SQL is crucial. The industrywide endeavor to formulate a standard to extend SQL goes by the name SQL-3, the popular standard being SQL-2. SQL-3, provides support for ADTs and other data structures. It specifies the syntax and the semantics, giving freedom to vendors to customize their implementation.

For a spatial extension of SQL, a standard has already been agreed upon. The OGIS (Open GIS) consortium, led by major GIS and database vendors, has proposed a specification for incorporating 2D geospatial ADTs in SQL. The ADTs proposed are based on

the *object* model and include operations for specifying topological and spatial analysis operations. In Chapter 2, we give a detailed description of the OGIS standard, along with an example to illustrate how spatial queries can be performed from within SQL.

1.6.3 Query Processing

As mentioned before, a database user interacts with the database using a declarative query language such as SQL. The user only specifies the result desired and not the algorithm to retrieve the result. The DBMS must automatically implement a plan to efficiently execute the query. Query processing refers to the sequence of steps that the DBMS will initiate to process the query.

Queries can be broadly divided into two categories: single-scan queries and multi-scan queries. In a single-scan query, a record (tuple) in the table (relation) being queried has to be accessed at most once. Thus the worst-case scenario, in terms of time, is that each record in the table will be accessed and processed to verify whether it meets the query criterion. The first spatial query we introduced in this chapter—*"List the names of all bookstores which are within ten miles of the Minneapolis downtown"*—is an example of a single-scan query. The result of this query will be all bookstores that intersect a circle of radius ten miles centered on the courthouse. This query is also an example of a *spatial-range* query, where the range refers to the query region. Here the query region is the circle of radius ten miles. If the query region is a rectangle, the spatial range query is often referred to as a *window* query.

A join query is a prototype example of a multiscan query. To answer a join query, the DBMS has to retrieve and combine two tables in the databases. If more than two tables are required to process the query, then the tables may be processed in pairs. The two tables are combined, or "joined," on a common attribute. Because a record in one table can be associated with more than one record in the second table, records may have to be accessed more than once to complete the join. In the context of SDBs, when the joining attributes are spatial in nature, the query is referred to as a *spatial-join* query. We now give an example to illustrate the difference between a nonspatial join and a spatial join.

Figure 1.7 illustrates the difference between a nonspatial and a spatial join. Consider the two relations shown in the figure: senator and business. The senator relation is characterized by three attributes: the name of the senator, his or her social security number, the senator's gender, and the district that the senator represents. The *district* is a spatial attribute and is represented by a polygon. The other relation, business, lists information about all the businesses: the business name, its owner, the owner's social security number, and the location of the business. The *location* is a spatial attribute represented as a point location. Consider the following query: *"Find the name of all female senators who own a business."* This example includes a nonspatial join, where the joining attributes are the social security number of the senator relation and the social security number of the business owner. In SQL, this query will be written as follows:

```
SELECT   S.name
FROM     Senator S, Business B
WHERE    S.soc-sec = B.soc-sec AND
         S.gender = 'Female'
```

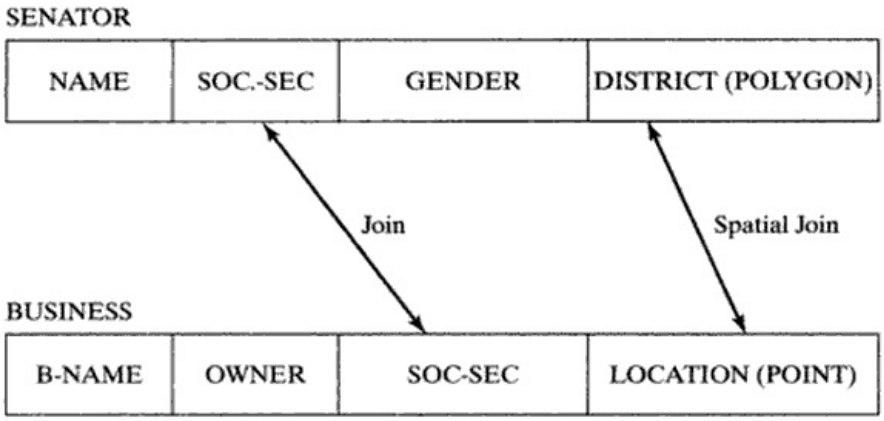


FIGURE 1.7. Two relations to illustrate the difference between join and spatial join.

Now consider the query, "*Find all senators who serve a district of area greater than 300 square miles and who own a business within the district.*" This query includes a spatial-join operation on the attributes *location* and *district*. Thus, while in a nonspatial join the joining attributes must be of the same type, in a spatial join the attributes can be of different types: point and polygon, in this case. This is how the query will be expressed in SQL:

```
SELECT   S.name
FROM     Senator S, Business B
WHERE    S.district.Area() > 300 AND
         Within(B.location, S.district)
```


An SDBMS processes range queries using the filter-refine paradigm. This is a two-step process. In the first step, the objects to be queried are represented by their minimum bounding rectangles (MBRs). The rationale is that it is easier (computationally cheaper) to compute the intersection between a query region and a rectangle rather than between the query region and an arbitrary, irregularly shaped, spatial object. If the query region is a rectangle, then at most four computations are needed to determine whether the two rectangles intersect. This is called the filter-step, because many candidates are eliminated by this step. The result of the filter-step contains the candidates that satisfy the original query. The second step is to process the result of the filter-step using exact geometries. This is a computationally expensive process, but the input set for this step, thanks to the filter-step, has low cardinality. Figure 1.8 shows an example of the filter-refine strategy.

We now describe, with the help of an example, an algorithm to process the filter-step of a spatial-join query. This algorithm is based on the *plane sweep* technique, which is extensively used in computational geometry to compute the intersections of geometric objects.

The filter step of many spatial-join queries can be reduced to the problem of determining all pairs of intersecting rectangles. Consider two sets of rectangles (Figure 1.9[b]) $R = \{R_1, R_2, R_3, R_4\}$ and $S = \{S_1, S_2, S_3\}$, which represent the MBRs of the spatial attributes of two tables involved in a join. Each rectangle T can be identified by its lower-left corner, $(T.xl, T.yl)$ and its upper-right corner $(T.xu, T.yu)$, as shown in Figure 1.9(a). We sort all the rectangles in R and S according to the x values of their

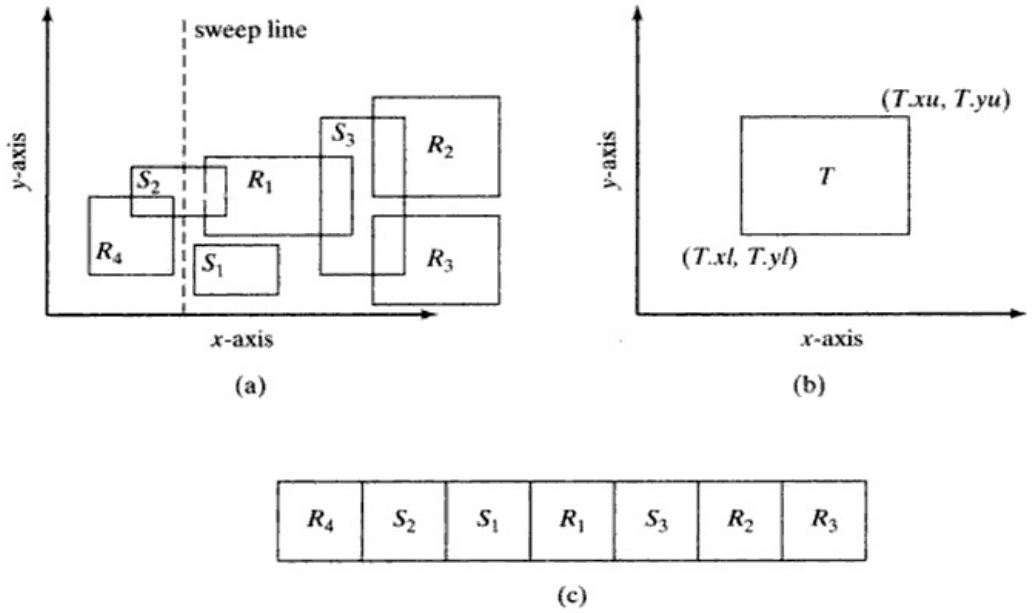


FIGURE 1.9. Determining pairs of intersecting rectangles: (a) two sets of rectangles: R and S; (b) a rectangle T with its lower-left and upper-right corners marked; and (c) the sorted set of rectangles with their joins. Note the filtering nature of the plane sweep algorithm. In the example, twelve possible rectangle pairs could be joined. The filtering step reduced the number of possibilities to five. An exact geometry test can be used to check which of the five pairs of objects satisfy the query predicate [Beckmann et al., 1990].

1.6.4 File Organization and Indices

DBMSs have been designed to handle very large amounts of data. This translates into a fundamental difference in how algorithms are designed in a GIS data analysis versus a database environment. In the former, the main focus is to minimize the computation time of an algorithm, assuming that the entire data set resides in main memory. In the latter, emphasis is placed on minimizing the sum of the computation and the I/O (input/output) time. The I/O time is the time required to transfer data from a disk (hard drive) to the main memory. This is because, despite falling prices, the main memory is not large enough to accommodate all the data for many large applications. This difference is conceptualized in the way one perceives the fundamental design of the computer. For many programmers, the computer essentially consists of two components: the CPU (central processing unit) and an infinite amount of main memory (Figure 1.10[a]). On the other hand, for many DBMS designers, the computer has three parts: the CPU, finite main memory, and infinite disk space (Figure 1.10[b]). Although a CPU can directly access data in the main memory, to access data on the disk, it first has to be transferred into the main memory. The difference in time between accessing data from random access memory (RAM) and disk is unbelievably large: a factor of a hundred thousand (year 2000). This ratio is getting worse every year because the CPU and memory are getting faster at a higher rate than disks and secondary storage devices.

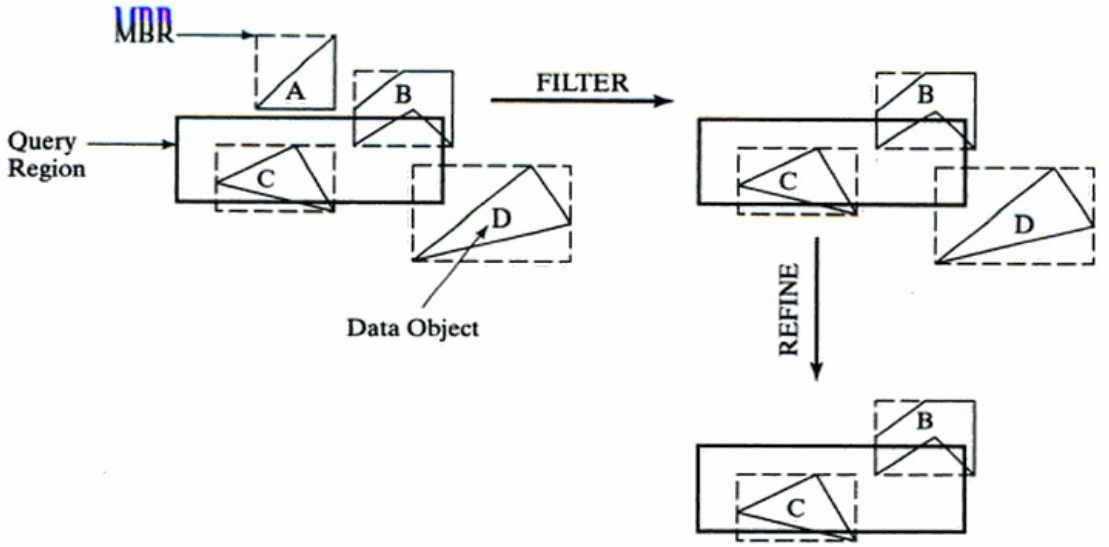


FIGURE 1.8. The filter-refine strategy for reducing computation time.

lower-left corners (i.e., on $T.xl$). The sorted rectangles are shown in Figure 1.9(c). We collect all of the (sorted) rectangles in R and S in the set $R \cup S$ and proceed as follows:

1. Move a *sweep line*, for example, a line perpendicular to the x-axis, from left to right, and stop at the first entry in $R \cup S$. This is the rectangle T with the smallest $T.xl$ value. In our example (see Figure 1.9(a)) this is the rectangle $R4$.
2. Search through the sorted rectangles of S until arriving at the first rectangle S^f such that $S^f.xl > T.xu$. Clearly, the relation $[T.xl, T.xu] \cap [S^j.xl, S^j.xu]$ holds (nonempty) for all $1 \leq j < f$. In our example S^f is S^1. Note that the superscript is in the order of array index in Figure 1.9(c), i.e., $S^1 = S_2, S^2 = S_1, S^3 = S_3$. Thus S_2 is a candidate rectangle that may overlap R_4. This will be confirmed in the next step.
3. If the relation $[T.yl, T.yu] \cap [S^j.yl, S^j.yu]$ holds for any $1 \leq j \leq f$, then the rectangle S^j intersects T. Thus this step confirms that R_4 and S_2 indeed overlap, and $< R_4, S_2 >$ are part of the join result. Record all this information, and remove the rectangle T from the set $R \cup S$. R_4 is removed from the set $R \cup S$ because it cannot participate in any more pairs in the resulting set.
4. Move the sweep line across the set $R \cup S$ until it reaches the next rectangle entry. This is rectangle S_2 in our example. Now proceed as in steps 2 and 3.
5. Stop when the $R \cup S = \emptyset$.

The filter step of the spatial-join algorithm results in $<R_4, S_2>$, $<R_1, S_2>$, $<R_1, S_3>$, $<R_2, S_3>$, $<R_3, S_3>$ as candidate pairs for the refinement step, based on the exact geometries of the objects. The refinement step may eliminate a few pairs from the final result if exact geometry computation shows that there is no overlap. The main purpose of the filter step is to reduce the computation cost of exact geometry computation by eliminating as many pairs as possible. Chapters 5 and 7 present more details of disk-based algorithms for processing spatial queries.

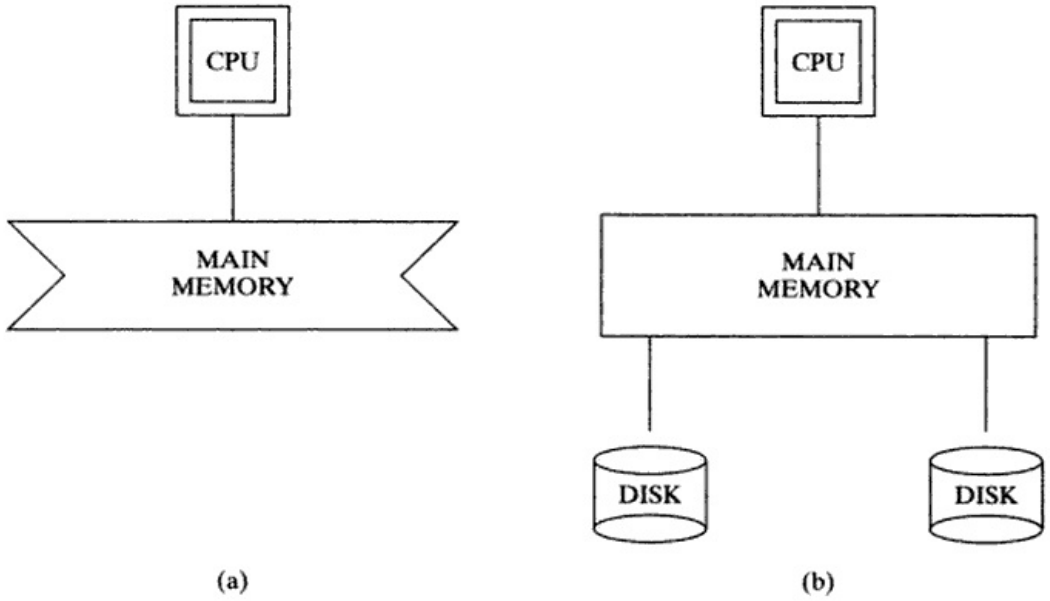


FIGURE 1.10. (a) Programmer's viewpoint; (b) DBMS designer's viewpoint.

At this moment (details in Chapter 4), it is worthwhile to view the secondary storage device (hard drive) as a book. The smallest unit of transfer between the disk and main memory is a page, and the records of tables are like structured lines of text on the page. A query issued by a user essentially causes a search about a few selected lines embedded in pages throughout the book. Some pages can reside in the main memory, and pages can be fetched only one at a time. To accelerate the search, the database uses an index. Thus in order to search for a line on a page, the DBMS can fetch all of the pages spanned by a table and scan them line by line until the desired record is found. The other option is to search in the index for a desired *key* word and then go directly to the page specified in the index. The index entries in a book are sorted in alphabetical order. Similarly, if the index is built on numbers, like the social security number, then they can be numerically ordered.

Spatial data can be sorted and ordered, but that comes at the loss of spatial proximity. What do we mean here? Consider Figure 1.11(a). Here we have a grid that is row-ordered. Now consider the neighbors of 4. On the grid the neighbors of 4 are 3, 7, and 8, but if stored in the way it is sorted, its neighbors will be 3 and 5. Thus, sorting has destroyed the original neighborhood relationship. Much research has been done to find better ways of ordering multidimensional data. Figure 1.11(b) shows another method. It is worth pointing out that no total ordering can preserve spatial proximity completely.

The *B-tree* (for balanced) index structure is arguably the most popular index employed by a relational DBMS. It is sometimes stated that the B-tree index structure is a major reason for the almost universal acceptance of relational database technology. The B-tree implementation crucially depends on the existence of an order in the indexing field. Figure 1.12 shows the crucial difference between a binary tree and a B-tree. Each node of a B-tree corresponds to a page of the disk. The number of entries in each node

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(a)

7	8	14	16
5	6	13	15
2	4	10	12
1	3	9	11

(b)

FIGURE 1.11. Different ways of ordering multidimensional data: (a) row order; (b) Z-order. If a line is drawn following the numbers in ascending order, the Z pattern will become obvious.

depends on the characteristics of the indexing field and the size of the disk page. If a disk page holds m keys, then the height of a B-tree is only $O(\log_m(n))$, where n is the number of total records. For trillon (10^{12}) records, one only requires a B-tree of height 6 for $m = 100$. Thus a record for a given key value can be retrieved in about half a dozen disk accesses even from such a large collection of records.

Because a natural order does not exist in multidimensional space, the B-tree cannot be used directly to create an index of spatial objects. The combination of spatial-order and B-tree is a common way to get around the lack of a natural order on spatial objects. Many commercial systems have adopted such an approach.

The R-tree data structure was one of the first index specifically designed to handle multidimensional extended objects. It essentially modifies the ideas of the B-tree to accommodate extended spatial objects. Figure 1.13 shows an example of how the R-tree organizes extended objects. More details follow in Chapter 4.

1.6.5 Query Optimization

One of the major strengths of relational database technology is how it efficiently executes a query by generating a sophisticated query evaluation plan. To explain this we revisit the query "*Find the names of all female senators who own a business.*" This query is actually a composition of two subqueries: a selection query and a join query. "*Name all*

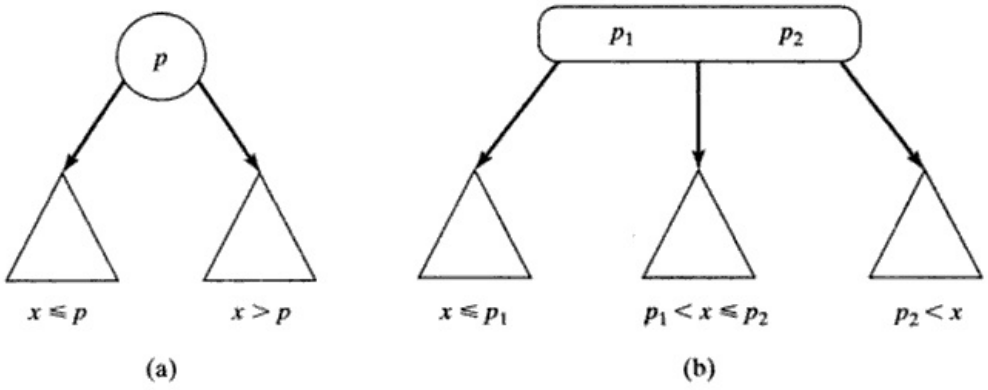


FIGURE 1.12. (a) Binary tree; (b) B-tree.

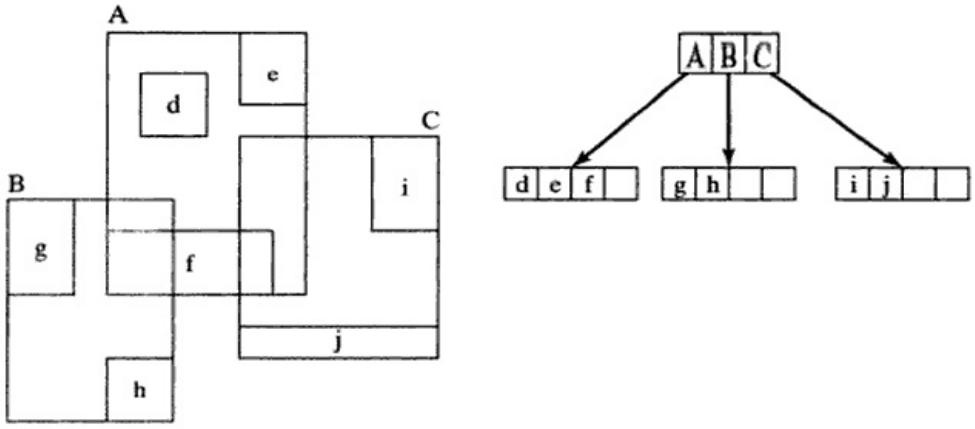


FIGURE 1.13. R-tree extends B-tree for spatial objects.

female senators" is a selection query because we select all of the female senators from a list of all senators. The query *"Find all senators who own a business"* is a join query because we combine two tables to process the query. The question is, in which order should these subqueries be processed: select before join or join before select? Remember a join is a multiscan query, and select is a single-scan query, so it is more important for a join operation to work on a smaller size table than the select operation. Now it is obvious that select should be done before join.

Consider the spatial query we have discussed before: *"Find all senators who serve a district of area greater than 300 square miles and who own a business within the district."* This query too is a composition of two subqueries: a range query and a spatial-join query. The range subquery is *"Name all senators who serve a district of area greater than 300 square miles."* The spatial-join query is *"Find all senators who own a business located in the district they serve."* Again, a range query is a single-scan and a join is a multiscan query, but there is one important difference. The single-scan query has to employ a potentially expensive function, *S.district.Area()*, to compute the area of each district. Of course, the join function also involves a computationally expensive function, *Within(B.location,S.district)*, but the point is that it is very difficult to assess how to order the processing of in spatial subqueries. This important step is taken up in Chapters 5 and 7.

1.6.6 Data Mining

Data mining, the systematic search for potentially useful information embedded in digital data, is now a hot topic of research inside and outside academia. Major corporations and government agencies have realized that the large amount of data that they have accumulated over the years, and are still gathering, can be systematically explored to provide a new perspective on their operations. Thus, data that was earlier considered a burden in terms of storage and management has suddenly become a source of wealth and profits. The story, now almost of mythic proportions, of how it was discovered by using data-mining tools that people who buy diapers in the afternoon are likely to buy beer too has fueled a massive effort to design and invent efficient tools for mining data. Until now most of the effort in data mining has focused on algorithm design and analysis. A database specialist can leverage his or her expertise on two fronts:

TABLE 2.1: Examples of Topological and Nontopological Operations

Topological	
endpoint(point, arc)	A point at the end of an arc.
simple-nonself-intersection (arc)	A nonself-intersecting arc does not cross itself.
on-boundary (point, region)	Vancouver is on the boundary of Canada and the United States.
inside (point, region)	Minneapolis is inside of Minnesota state.
outside (point, region)	Madison is outside of Minnesota state.
open (region)	Interior of Canada is an open region (excludes all its boundaries).
close (region)	Carleton County is a closed region (includes all its boundaries).
connected (region)	Switzerland is a connected region, whereas Japan is not (given any two points in the area, it is possible to follow a path from one point to the other such that the path is entirely within the area).
inside (point, loop)	A point is within the loop.
crosses (arc, region)	A road (arc) passes through a forest (region).
touches (region, region)	Minnesota (region) is a neighboring state of Wisconsin (region).
touches (arc, region)	Interstate freeway 90 (arc) passes by Lake Michigan (region).
overlap (region, region)	Land cover (region) overlaps with land use (region).
Nontopological	
Euclidean-distance (point, point)	Distance between two points.
direction (point, point)	Madison city is east of Minneapolis city.
length (arc)	Length of a unit vector is one unit.
perimeter (area)	Perimeter of a unit square is 4 units.
area (region)	Area of unit square is one square unit.

curve, interior point of a curve, or off the curve as listed in Table 2.1. The relationships between curve are more complex and still an area of research [Clementini et al., 1993].

Directional

Directional relationships can be of three types, namely, absolute, object-relative, or viewer based. Absolute direction relationships are defined in the context to a global reference system, for example, north, south, east, west, northeast, and so on. Object-relative directions are defined, using the orientation of a give object. Example relationships include left, right, front, behind, above, below, and so on. Viewer relative directions are defined with respect to a specially designated reference object called the viewer.

example is useful in designing new drugs (pharmaceutical) by finding molecules which can dock in certain areas of other molecules. Another family of spatial operations is based on visibility relationships to address queries like list the locations on a golf course to get best view of hole 9. We hope to see consensus mathematical frameworks for such families of operations in the near future.

2.1.6 Mapping Spatial Objects into Java

Having defined the spatial hierarchy (see Figure 2.2 and the corresponding array of spatial relationships), we can easily program these notions into an object-oriented programming language such as Java.

We give an example of how a specific query can be written in Java. The query is, *"Find all tourist offices within 10 miles of the Maple Campground."*

The java class `Facility` and `FacilitySet` are used to model the locations of tourist offices as well as the campgrounds. A separate class `Query` is used to model the distance computation and the query itself. The `Facility` class has three attributes, namely, *name*, *type*, and *location*. The *type* attribute is used to distinguish between tourist offices and campgrounds. The location of facilities is *point* spatial datatype. This program assumes that the spatial datatype *point* is available as part of a spatial library (e.g., SDE from ESRI). The library also provides a *distance* function on the *point* spatial datatypes. There are three methods in `FacilityClass`. The method *facility* is a constructor used to initialize new objects. Method *getName* is used to extract the *name* attribute and method; *withinDistance* is used to test if another facility is within a given distance.

Class `FacilitySet` is used to model a collection of facilities. For example, it can model a set of tourist offices. It has two attributes, *maxSize* and *facilityTable*. maxSize designates the maximum size of the collection of facilities. The facilityTable is used to store information about each facility. There is only one method in `FacilitySet`. This method is used to read the information about various facilities from a file to initialize the attributes of an instance of `FacilitySet`.

The class `Query` implements a query to extract all the tourist offices within 10 miles of the Maple campground. This class has one method, named *main*, which loops through the list of tourist offices checking the distance to the Maple campground.

The key point of this discussion of the Java program is to show that the spatial datatypes and operations can be used from a host language different from SQL, which is used in the rest of the book. The second point is to compare the amount of programming effort required by spatial querying. SQL will greatly reduce the amount of code needed to specify simple spatial queries. It will also reduce the burden of performance tuning and data-structure selection. We encourage readers to compare SQL with Java in the coming chapters to revisit these points.

```
import java.lang.*;
import java.io.*;
import java.util.*;


/* assume class Point is given, which contains two attributes: x and y with
double type, and also some member functions, including distance (Point) */

/* Assume the original data is stored in file ''facilityFile''.
```

1. Addition: $u + v \in V$ for all $u\, v \in V$
2. Product: $\alpha u \in V$ for all $\alpha \in R, v \in V$.

Besides the existence of a special vector 0, there are other axioms that the two operations addition and product need to satisfy. For a complete discussion of vector space see [Blyth and Robertson, 1999].

If there exists a (minimal) finite set of vectors $\{e_1, e_2, \ldots, e_n\}$ such that any $v \in V$ can be expressed as a linear combination of the $e_i's$, that is, there exists $\alpha_1, \ldots \alpha_n \in R$ such that:

$$v = \alpha_1 e_1 + \ldots + \alpha_n e_n$$

then the vector space is finite dimensional. In a three-dimensional space, the $e_i's$ correspond to the familiar x, y, z coordinate axis. If we add the notion of inner product (angle) to vector space, we get a Euclidean space. In a Euclidean space setting, all spatial relationships, including set, topological, metric, and directional (north/south), can be defined.

2.1.5 Dynamic Spatial Operations

Most of the operations that we have discussed so far have been *static*, in the sense that the operands are not affected by the application of the operation. For example, calculating the length of a curve has no effect on the curve itself. *Dynamic* operations alter the objects upon which the operations act. The three fundamental dynamic operations are create, destroy, and update. All dynamic operations are variations upon one of these themes [Worboys, 1995]. In Table 2.2 some example of the create and update operations are listed.

An example of the *merge* operation is the "map-reclassification" operation supported by many GIS software products. Consider reclassification of the map of countries on the basis of the majority religion practiced in the countries, for example, *none, Islam, Christianity, Buddhism, and Hinduism*. The boundaries between neighboring countries with the same majority religion are removed via the *merge* operation in order to reclassify the maps. Similar examples of other dynamic spatial operations can be found in cartographic projections and map editing features in a GIS.

Additional classes of operations on spatial object exists. For example, operations can be defined on the shape of an extended object to ask queries, such as, which objects are squarish in shape? Which pair of objects fit like pieces of a jig-saw puzzle. The last

TABLE 2.2: Representative Example of Dynamic Spatial Operations [Worboys, 1995].

Create	reproduce (X)	Produces an exact replica of X.
	generate (X)	Generates an object that depends on but does not replicate X.
	split (X)	Creates objects whose aggregation is X.
	merge (X,Y,Z)	X, Y, and Z are merged to create a single object.
Update	translate	Shifts the position of the object in the plane.
	rotate	Changes the orientation without changing the shape.
	scale	The object size is changed, but the shape remains the same.
	reflect	The mirror reflection of an object in a straight line.
	shear	Deforms the object in a predefined fashion.

```
Each line in the file represents a facility; use @@ as its delimiter, e.g.
   Maple @@ campground @@ 2.0 @@ 3.0
   Office @@ Tourist-Office @@ 6.0 @@ 8.9

public class Facility {
   protected String name;
   protected String type;
   protected Point  location;

   public Facility (String name, String type, Point location) {
     this.name = name;
     this.type = type;
     this.location = location;
   }

   public String getName() {
     return name;
   }

   public boolean withinDistance(Facility f, double d) {
     if (this.location.distance(f.location) < d)
       return true;
     else
       return false;
   }
}
public class FacilitySet {
   const maxSize = 50;
   protected Facility[maxSize] facilityTable;

   /* read from file filename and initialize the facility table */
   public FacilitySet(String filename) {
     BufferedReader in = new BufferedReader (new FileReader(filename));
     String inline;
     StringTokenizer strLine;
     int i=0 ;
     String token;

     while ((inline = in.readLine())!= null) {
       strLine = new StringTokenizer(inline, "@@");

       /* read name */
      token = strLine.nextToken();
      FacilityTable[i].name = token;

      /* read type */
      token = strLine.nextToken();
      FacilityTable[i].type = token;

      /* read x coordinate */
```

```
        token = strLine.nextToken();
        FacilityTable[i].location.x = Double.valueOf(token).doubleValue();

        /* read x coordinate */
        String type token = strLine.nextToken();
        FacilityTable[i++].location.y = Double.valueOf(token).doubleValue();

    }
  }
}

public class FacilityDemo {

   public static void main(String[] args) {

     Facility f = new Facility("Maple", "Campground", Point(2.0,4.0));
     Facility[] fTable = new FacilitySet("facilityFile");
     String[] resultTable = new string[fTable.length];

     int j=0;
     for (int i=0; i < fTable.length; i++) {
       if (f.withinDistance(fTable[i], 2.0)
         and fTable[i].type = "Tourist-Office")
         resultTable[j++] = fTable[i].name;
     }
   }
}
```


2.2 THREE-STEP DATABASE DESIGN

So far in this chapter we have described two models of spatial information: the object model and the field model. Our view of these models was from the perspective of data modeling involving concepts intrinsic to the spatial domain. We now introduce classical data modeling from the perspective of database design. Our ultimate goal is to "blend" the two families of concepts.

Database applications are modeled using a three-step [Elmasri and Navathe, 2000] design process. In the first step, all the available information related to the application is organized, using a high-level *conceptual data model*. At the conceptual level, the focus is on the datatypes of the application, their relationships and constraints. The actual implementation details are left out at this step of the design process. Plain text combined with simple but consistent graphic notation is often used to express the conceptual data model. The Entity Relationship (ER) model is one of the most prevalent of all conceptual design tools.

The second step, also called the logical modeling phase, is related to the actual implementation of the conceptual data model in a commercial DBMS. Data in a commercial database is organized using an implementation model. Examples of implementation models are hierarchical, network, and relational. The relational model is one of the most widely implemented models in current commercial databases. In the relational model, the datatypes, relationships, and constraints are all modeled as `Relations`. Tightly coupled with the relational model is the formal query language relational algebra (RA). The RA consists of

simple operations that allow quering of the data organized as relations. RA is described in detail in Chapter 3.

The relational model does not fit the need of spatial data, as explained by [Herring, 1991]:

> Relational algebra characterizes the querying capabilities of relational databases. A relational database can answer any query expressible in relational algebra, which is universally accepted model of traditional applications of relational databases.
>
> In contrast, there is no universally accepted mathematical model of geographic information, making it difficult to design spatial query languages and spatial databases. In addition, there is a considerable semantic gap between GIS and relational databases, leading to complexities and inconveniences.

Chapter 7 provides a specific example to show that RA is not capable of expressing the *transitive closure*, an important graph operation, without additional assumptions. The transitive closure operation is closely linked to the *reclassification* operation [Delis et al., 1994], discussed in Section 2.1.5.

The third and final step, modeling of the physical design, deals with the nuts and bolts of the actual computer implementation of the database applications. Issues related to the storage, indexing, and memory management are handled at this level and will be the subject of discussion in subsequent chapters. We now introduce the ER model.

2.2.1 The ER Model

The first step in database design is to come up with the conceptual model of the "miniworld." The purpose of the conceptual model is to represent the miniworld in a manner devoid of computer metaphors. The motivation is to separate out the concepts of the application from the implementation detail. There are many design tools available for conceptual data modeling, but the ER model is one of the most popular. The ER model integrates seamlessly with the relational data model, which in turn is one of the most prevalent logical models, the second step in the three-step design paradigm. Besides the ER model and propelled by the success of the object-oriented design methodology, the UML is another popular conceptual modeling tool. We discuss the UML in some detail in the next section, but here we model the *State-Park* example using the ER model.

Entities and Attributes

In the ER model, the miniworld is partitioned into *entities* which are characterized by *attributes* and interrelated via *relationships*. Entities are things or objects that have an independent physical or conceptual existence. In the *State-Park* example, `FOREST`, `RIVER`, `FOREST-STAND`, `ROAD`, and `FIRE-STATION` are all examples of entities.

Entities are characterized by *attributes*. For example, *name* is an attribute of the entity `FOREST`. An attribute (or a set of attributes) that uniquely identifies instances of an entity is called the *key*. In our example the *name* attribute of the entity `ROAD` is a key, assuming it is not possible that two different roads have the same name. All instances of `ROAD` in our example database have unique names. Though not a conceptual design issue, a mechanism must exist in the DBMS to enforce this constraint.

Attributes can be single valued or multivalued. *Species* is a single-valued attribute of `FOREST-STAND`. We now explain the existence of a multivalued attribute in the context

of our example. The FACILITY entity has an attribute Pointid, which is a unique identification for the spatial location of instances of the entity. We are assuming that the **map** scale predicates instances of FACILITY to be represented as points. It is possible that a given facility spans two distinct point locations, in which case the Pointid attribute is multivalued. A similar argument holds for other entities.

Suppose we want to store information about the *elevation* of a FOREST. Because *elevation* can change values within the FOREST entity, we model it as a multivalued attribute, since field data type is not available.

Relationships

Besides entities and attributes, the third construct in the ER model is the *relationship*. Entities interact or connect with each other through relationships. We have already noted spatial relationships in the previous section; here we focus on the general concept of relationships. Though many entities can participate in a given relationship, we will deal only with *binary* relationships, that is, those between two entities. There are three kinds of relationships based on cardinality constraints: one-one, many-one, and many-many.

One-One (1:1)
In a one-one relationship, each instance of one entity can relate to only one instance of the other participating entity. For example, the relationship *manages* between the entities MANAGER and FOREST is a one-one relationship: a FOREST can only have one MANAGER, and a MANAGER can manage only one FOREST.

Many-One (M:1)
A many-one relationship can potentially connect many instances of one entity with one instance of the other entity participating in the relationship. The *belongs_to* is a many-one relationship between the entities FACILITY and FOREST, assuming each facility belongs to only one forest but many facilities can belong to a forest.

Many-Many (M:N)
Sometimes many instances of one entity can be related to many instances of the other participating entity. The relationship supplies_water_to which connects the entities RIVER and FACILITY is such a relationship. Sometimes relationships can have attributes too. *supplies_water_to* has an attribute *Volume* which keeps track of the quantity of water supplied by a river to a facility.

ER Diagram

Associated with the ER model is the ER diagram, which gives a graphic representation to the conceptual model. In the ER diagram, entities are represented as *boxes*, attributes as *ovals* connected to the boxes with straight lines, and relationships as **diamond** boxes. The *cardinality* of the relationship, whether it is 1:1, M:1, or M:N, is shown around the diamonds. The key attribute is underlined, and multivalued attributes are represented as *double-ovals*. The ER diagram of the *State-Park*example is shown in Figure 2.4. There are seven entities, namely, FOREST-STAND, RIVER, ROAD, FACILITY, FOREST, FIRE-STATION, and MANAGER. The attributes of entity FOREST include *name*, *elevation*, and *polygonid*. *Name* is a unique id, that is, each forest has a unique name. The diagram

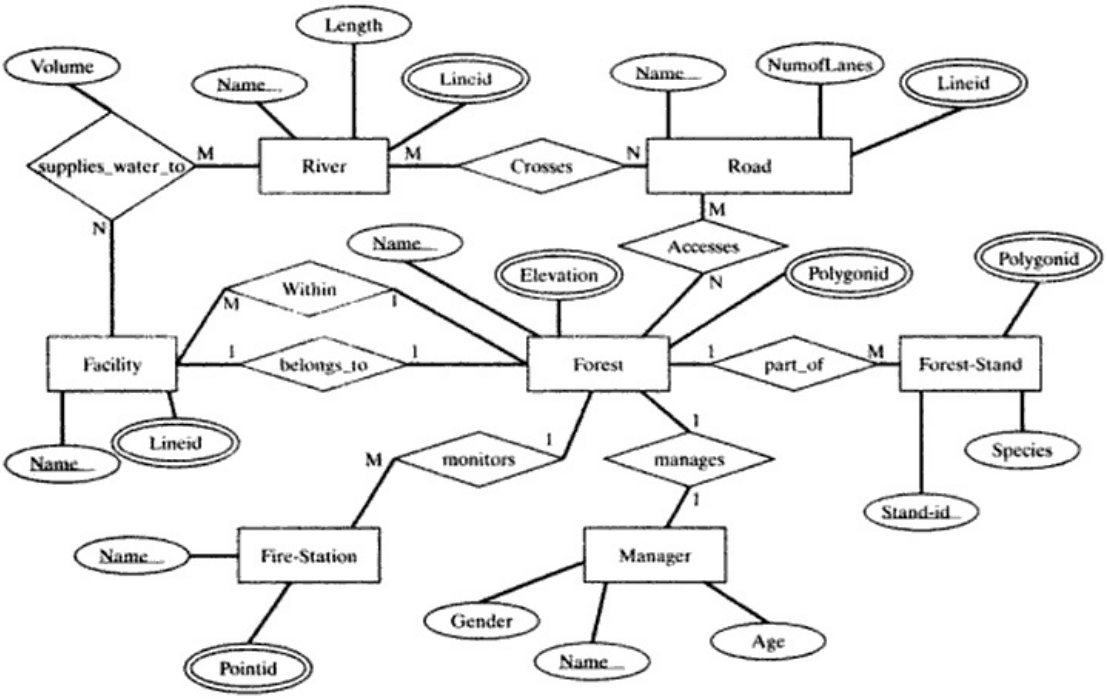


FIGURE 2.4. An ER diagram for the *State-Park* example.

also shows eight relationships. Entity FOREST participates in six relationships. Entity FIRE-STATION participates in only one relationship, namely *monitors*. The cardinality constraints show that each fire station monitors one forest, but many fire stations can monitor a forest. Some of the relationships are spatial in nature. These include *crosses*, *within*, *part-of*. Many other spatial relationships are implicit in this diagram. For example, a river *crosses* a road is shown explicitly, but a river *crosses* a forest is implicit.

2.2.2 The Relational Model

The relational model to represent data was introduced by Codd in 1970. Since then, it has become one of the most popular logical data models. The popularity and power of this model is a consequence of the simplicity of its structure. We explain the terminology of the relational model in the context of the *State-Park* example. Suppose we wanted to organize available data of all the forests in state parks. Then we could organize the information in the form of a table, namely the table `Forest`, and list the array of the available information in columns. For the `Forest` table, the associated data consists of three things: the *Name* of the Forest, its *Elevation*, and spatial *Geometry*.

The table is called a *relation*, and the columns, *attributes*. Each different instance of `Forest` will be identified with a row in the table. A row is called a *tuple*, and the order in which the rows and columns appear in the table is unimportant. Thus a relation is an unordered collection of tuples. Together the table and column names constitute the *relation schema*, and a collection of rows or tuples is called a *relational instance*. The number of columns is called the degree of the relation. The `Forest` is a relation of degree three. Similarly, data about, for example, the different forest stands and rivers can be organized in separate tables.

What are the allowable values of the attributes? In traditional database applications the datatypes of the attributes, called `domains`, are limited, consisting of integers, floats, character strings, dates, and a few other domains. Furthermore, there is no provision for user-defined data types. In the `Forest` table, the attribute *name* fits nicely into this limited set but *elevation* and *geometry* do not. This is one of the reasons that traditional relational database technology is hard pressed to meet the requirements for SDBs. Despite that, we will show how spatial information can be mapped into a relational data model. In the object-relational data model, it would be justified to assume that we have added new basic domains or datatypes as specified in the OGIS standard. This is exactly what we will do in the next section.

Certain constraints on the relational schema must be maintained to ensure the logical consistency of the data. They are the *key*, *entity integrity*, and *referential integrity* constraints. The key constraint specifies that every relation must have a *primary key*. A key is a subset of the attributes of the relation whose values are unique across the tuples in a relation. There may be many keys in a relation, and the one that is used to identify the tuples in a relation is called the primary key. The entity integrity constraint states that no primary key can be null. This constraint is obvious; it would be impossible to uniquely identify tuples with a null value for its primary key. Logically consistent relationships between the different relations are maintained through the enforcement of referential integrity constraints. To explain the mechanism of referential integrity, we first describe the notion of a *foreign key*. A foreign key is a set of attributes in a relation which is duplicated in another relation. The referential integrity constraint stipulates that the value of the attributes of a foreign key either must appear as a value in the primary key of another relation or must be null. Thus a relation refers to another relation if it contains foreign keys.

2.2.3 Mapping the ER Model to the Relational Model

Many software packages, also called CASE tools, can translate an ER model to a relational schema. Example software packages include ERwin, Oracle Designer 2000, Rational Rose, and so on. This translation facility allows database designers to work with conceptual data models, focusing on the needs of application domain. If it were not for the spatial attributes, the ER model could be mapped seamlessly and intuitively into the relational model. There are six basic steps:

1. Map each entity into a separate relation. The attributes of the entity are mapped into the attributes of the relation. Similarly, the key of the entity is the primary key of the relation. The relational schema for ER diagram of Figure 2.4 is shown in Figure 2.5.
2. For relationships whose cardinality is 1:1, we place the key attribute of any one of the entities as a foreign key in the other relation. For example, the relation `Manager` has a foreign key attribute corresponding to the primary key, the *name*, of the `Forest`.
3. If the cardinality of the relationship is M:1, then place the primary key of $1-side$ relation as a foreign key in the relation of $M-side$. For example, the relation `Fire-Station` has a foreign key which is the primary key of the relation `Forest`.
4. Relationships with the cardinality M:N have to be handled in a distinct manner. Each M:N relationship is mapped onto a *new* relation. The name of the relation is

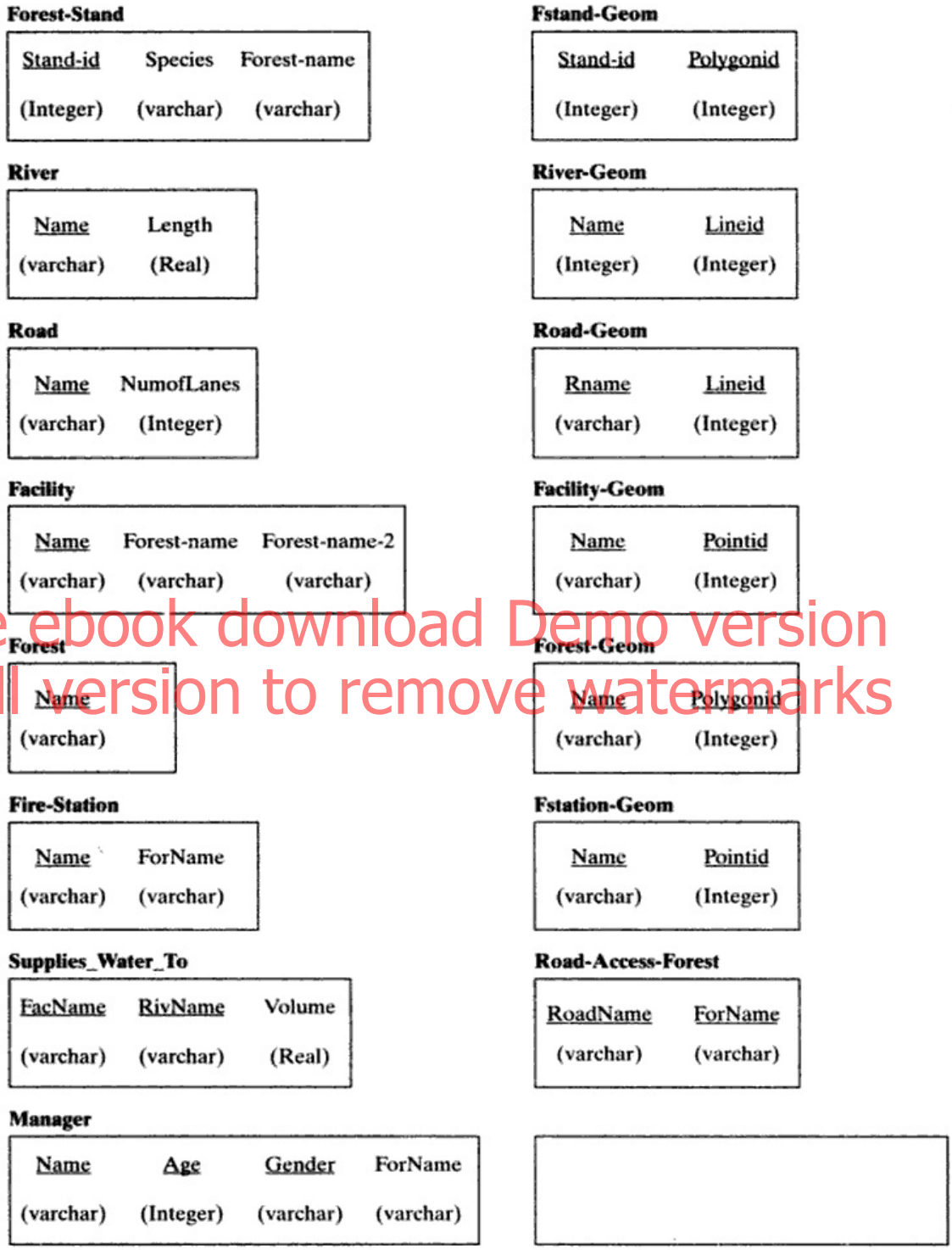


FIGURE 2.5. Relational schema for the State-Park example.

the name of the relationship, and the primary key of the relation consists of the pair of primary keys of the participating entities. If the relationship has any attributes then it becomes an attribute of the new relation. `Supplies_Water_To` is an example of an *M*:*N* relationship between the entities `Facility` and `River`. The river name and the facility name constitute the primary key, and the attribute *volume* becomes an attribute in the new table. Note that the M:N spatial relationship `Road Crosses River` results in a new table `Road-Crosses-River`.

5. For a multivalued attribute, a new relation is created that has two columns: one corresponding to the multivalued attribute and the other corresponding to the key of the entity that owns the multivalued attribute. Together the multivalued attribute and the key of the entity constitute the primary key of the new relation. For example, the `Forest-Stand` entity has a multivalued attribute *polygonid* which is an identification number (integer) for the geometric location of the city. *Pointid* is a multivalued attribute because a forest-stand may span two disjoint locations (e.g., a road may cut through a forest stand). We therefore have the relation `Fstand-Geom`. Similarly, we have relations `Forest-Geom`, `River-Geom`, `Road-Geom`, `Facility-Geom`, and `Fstation-Geom`.
6. The *elevation* attribute needs to be handled in a distinct manner. First, as we have pointed out, *elevation* is a multivalued attribute. Therefore we clearly need a new relation namely, `Elevation`. The attributes of this new relation are *Forest_Name*, *Elevation*, and *Pointid* as shown in Figure 2.6. The *elevation* attribute captures the height of the forest at the *pointid* location. In this table all three attributes constitute the primary key.

Spatial Tables

The spatial attributes and the space-varying attributes in the ER diagram have to be handled in a special way in the relational model. New domains, for example, spatial objects, are represented as new relations. The primary key for these relations is used as the foreign key in relations representing entities containing attributes typed using these domains. As described earlier, *pointid, lineid,* and *polygonid* are new domains and can be modeled as separate relations. Corresponding to each one of these attributes, there is a relation: `Point`, `Line`, and `Polygon` (see Figure 2.6).

Polygon

Polygonid	Seq-no	Pointid
(Integer)	(Integer)	(Integer)

Line

Lineid	Seq-no	Pointid
(Integer)	(Integer)	(Integer)

Point

Pointid	Latitude	Longitude
(Integer)	(Real)	(Real)

Elevation

Forest-name	Pointid (F.K.)	Elevation
(varchar)	(Integer)	(Real)

FIGURE 2.6. Schema for point, line, polygon, and elevation.

1. The `Point` table has three attributes: *pointid*, *latitude*, and *longitude*. Though there are many other reference systems, the latitude-longitude system is the most familiar, and all other reference systems can be derived from it.
2. A finite straight line can be defined in terms of two points. Therefore the *pointid* attribute in the `Line` table is a foreign key to the *point* table. The *seq-no* refers to the sequence number of the points that constitute a line identified by *lineid*.
3. The `Polygon` table is like the `Line` table with the constraint that the first and last sequence numbers refer to the same *pointid*.

2.3 TRENDS: EXTENDING THE ER MODEL WITH SPATIAL CONCEPTS

This translation of spatial attributes in an ER diagram to spatial table does not take full advantage of the spatial data types described in Section 2.1.3. It simply treats spatial attributes as any other nonspatial attributes. We now describe the emerging trend toward extending ER diagrams with pictograms to provide special treatment to spatial data types. This reduces clutter in ER diagrams, as well as in the resulting relational schema, while improving the quality of spatial modeling. The spatial relationships, for example, Road-Crosses-River can be omitted from the ER diagram and made implicit. The relations representing multivalued spatial attributes and M:N spatial relationships may not be needed in a relational schema.

As the earlier discussion shows, the ER model is unable, at least intuitively, to capture special semantics inherent in spatial modeling. Specifically, the shortcomings of the ER model are:

1. The ER model was originally designed with an implicit assumption of an object-based model. Therefore a field model cannot be naturally mapped using the ER model.
2. While in a traditional ER model, relations between entities are derived from the application under consideration, in spatial modeling there are always inherent relationships between spatial objects. For example, all the topological relationships discussed earlier are valid instances of relationships between two spatial entities. How should they be incorporated into the ER model without cluttering the diagram?
3. The type of entity used to model a spatial object depends on the scale of the "map." A city can be mapped as a point or a polygon depending on the resolution of the map. How should multiple representations of the same object be represented in a conceptual model?

2.3.1 Extending the ER Model with Pictograms

Many extensions have been proposed to the ER model to make the conceptual modeling of spatial applications easier and more intuitive. The idea is to add constructs that capture and convey the semantics of spatial reasoning and at the same time keep the graphical representation simple. Using *pictograms* to annotate and extend ER diagrams has been recently proposed.

Spatial relationships, including topological, directional, and metric relationships, are implicit between any two entities that have a spatial component. For example, it is natural to consider the topological relationship–*cross* between a `Forest` and `River` entity.

Including the *cross* relationship in an ER diagram does not convey added information about the structure of the application being modeled.

We will show how pictograms can be used to convey the spatial datatypes, the scale, and the implicit relationships of spatial entities. We will present the pictogram enhancement in the Bachus-Naur form (BNF) grammar notation, though it is not necessary to be familiar with the BNF notation to understand what follows. Information about the grammar notation can be found in any standard computer science text on compilers as well as many books on English grammar.

Entity Pictograms

Pictogram

A pictogram is a miniature representation of the object inserted inside of a box. These iconic representations are used to extend ER diagrams and are inserted at appropriate places inside the entity boxes. A pictogram can be a basic shape or a user-defined shape.

⟨ Pictogram ⟩ → [⟨ Shape ⟩]
→ [*]
→ [!]

Grammar (for Pictogram)

Shape

Shape is the basic graphical element in the pictogram, which represents the elements in the spatial data model. A model element can be a basic shape, a multishape, a derived shape, or an alternate shape. Many of the objects have simple basic shapes.

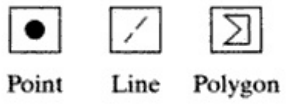
⟨ Shape ⟩ → ⟨ Basic Shape ⟩
→ ⟨ Multi-Shape ⟩
→ ⟨ Derived Shape ⟩
→ ⟨ Alternate Shape ⟩

Grammar (for Shape)

Basic Shape

In a vector model, the basic elements are the point, line, and polygon. In a typical application, a majority of the spatial entities are represented as basic shapes. In the forest example, we had represented a facility as a point (0-D), a river or road network as lines (1-D), and forest areas as polygons (2-D).

⟨ Basic Shape ⟩ → ●
→ /
→ ∑



Grammar (for Basic Shape) *Pictograms for Basic Shapes*

Spatial Databases

A TOUR

Shashi Shekhar • Sanjay Chawla

This inter-disciplinary comprehensive text shows how to manage spatial data in GIS, CAD, and multimedia systems and their application domains. This book helps readers master various stages of traditional spatial database design and implementation, introducing conceptual models (e.g. pictogram-enhanced ERD), explaining query languages (e.g. OGIS, SQL3), and finally describing efficient implementations using query optimization algorithms as well as spatial storage and indexing methods (e.g. R-trees). For students or professionals, this book balances cutting-edge research and practice to provide many firsts.

OUTSTANDING FEATURES

- ★ Comprehensive yet concise overview of Spatial Databases for students and practitioners
- ★ Self-contained treatment without assuming pre-requisite knowledge of GIS or Databases
- ★ Extensive use of industry standards, such as OGIS spatial data types and operations
- ★ Complete coverage of spatial networks including modeling, querying and storage methods
- ★ Detailed discussion of issues related to Spatial Data Mining
- ★ Balance between cutting-edge research and commercial trends
- ★ Variety of exercises at the end of each chapter
- ★ Easy-to-understand examples from common knowledge application-domains

Shashi Shekhar, an authority on spatial databases, is a professor and the head of the Spatial Database Research Group in the Department of Computer Science at the University of Minnesota. Shashi has published numerous articles and has advised many organizations on spatial database issues. He holds a Ph.D. degree in Computer Science from the University of California, Berkeley.

Sanjay Chawla is a Senior Technical Instructor with Vignette Corporation in Waltham, Massachusetts. He holds a Ph.D. degree from the University of Tennessee.

PEARSON Education

This edition is manufactured in India and is authorized for sale only in India, Bangladesh, Bhutan, Pakistan, Nepal, Sri Lanka and the Maldives.

ISBN 978-81-317-2628-0

9 788131 726280