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) generate one terabyte of data every day.

Satellite images are one prominent example of spatial data. To extract information from a satellite image, it 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 non-spatial 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 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 which manages them (DBMS) 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.

DRAFT

Data residing in these mammoth databases is simple, consisting of numbers, names, addresses, product descriptions, etc. These DBMSs are very efficient for the tasks they were designed for. For example, a query like "List the top ten customers, in terms of sales, in the year 1998" will be very efficiently answered by a DBMS even if the database 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 are most likely to be affected in the next great flood on the Mississippi?

Medical doctor: Based on this patient's 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.

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

DRAFT

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

Urban sprawl specialist: New urban land development should not be at 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: We want to build a new golf course which will maximize profit given the constraints of weather, EPA pesticide regulations, the Endangered Species Act, property prices, and the neighborhood demographic profile.

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

1.3 GIS and SDBMS

The Geographic Information System (GIS) is the principal technology motivating interest in Spatial Database Management Systems (SDBMS). GIS provides a convenient mechanism for the analysis and visualization of geographic data. Geographic data is spatial data where the underlying frame of reference is the Earth's surface. The GIS provides a rich set of analysis functions which allows a user to affect powerful transformations on geographic data. The rich array of techniques that geographers have packed into the GIS are the reason 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 set of objects and sets of layers. For example, 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 like, *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 will show in chapter three.

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.

DRAFT

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

Table 1.1: List of common GIS analysis operations [Albrecht, 1998]

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 to 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 database management systems. For example, spatial attachments, with esoteric names like *cartridge, datablade,* or more benign names like *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-a-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, like ESRI's Spatial Data Engine. The focus in the GIS community has evolved in the last decade, as shown in Figure 1.1. GIS, from being a software system for representing geographic information in a layered fashion, attained the status of GI science by focusing on the *algebra* of map and spatial operations. With the prominence of the Internet, the focus has again shifted to providing geographic or spatial services over the web. 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. For example, queries like "Find all Mexican restaurants in downtown

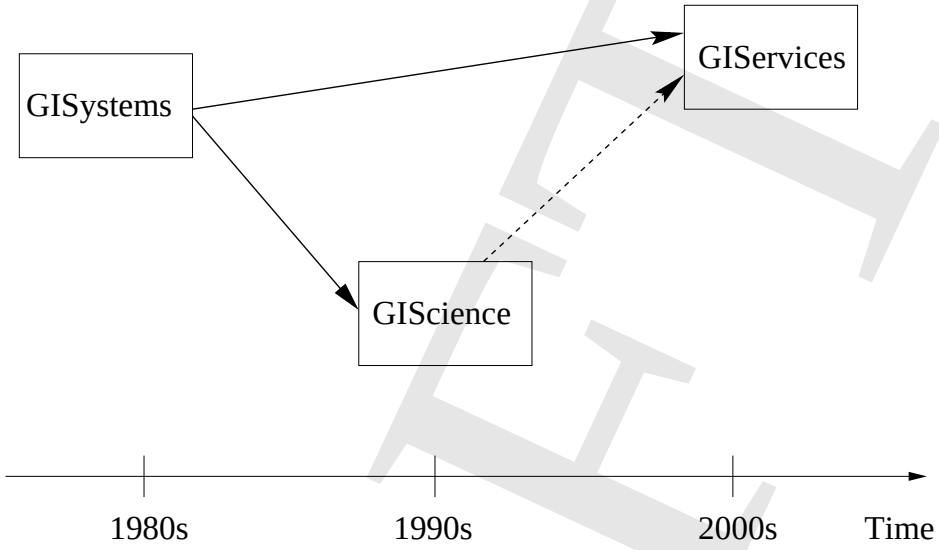


Figure 1.1: The evolution of the abbreviation GIS over the last two decades. In the 1980's GIS was Geographic Information System; in the 1990's, Geographic Information Science was the preferred phrase, and now the trend is toward Geographic Information Services.

Minneapolis" can be answered by such search engines. Another promising use of spatial technologies is related to the location determination of mobile-phones. The United States 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, on one hand, it is often remarked that the Internet revolution has "killed" the concept of geography; on the other hand, paradoxically, the demand for spatial technologies is continuously growing.

With recent increase in use of personal digital assistants (PDAs), mobile phones, two-way pagers have opened up several opportunities for new applications. Example applications are mobile workforces, location based services. Mobile workforce - as the name suggests, work from remote locations - at clients locations, branch offices, remote field locations. These workforce typically download a segment of data need for a particular task, work with that segment at 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: find a customer, find nearest pizza place with a gas station on the way, road map for a new city with tourist locations highlighted, location sensitive transactions and alerts. The impact and importance of the location based services led OGC to initiate OpenLS with a vision to integrate geospatial data and geoprocessing resources into the location services and telecommunication infrastructure.

1.5 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 location of wetlands (think black lines). From the image we can easily identify several lakes (north) and the Missisipi river (south). This county covers about 156 square miles and covers most of the urban and suburban regions of Minneapolis. 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. For simplicity we have shown only three vector layers. Information in a GIS is usually compiled in the form of layers. 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).

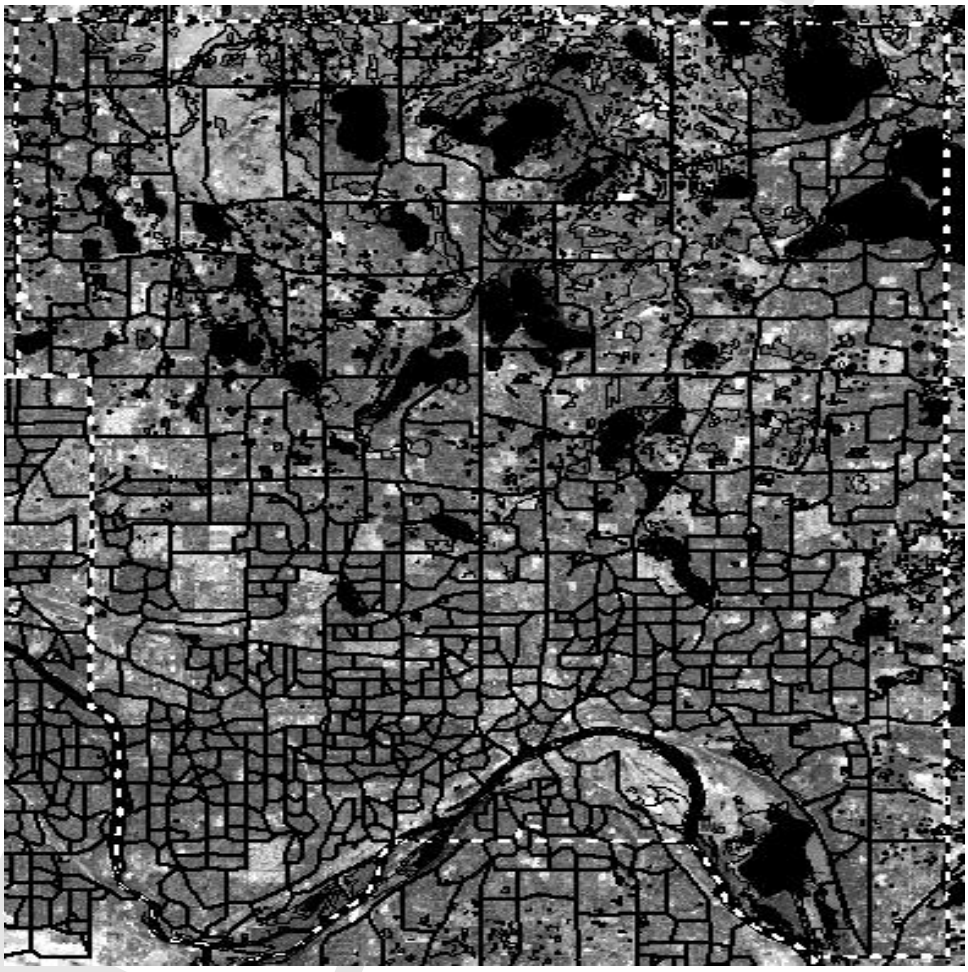
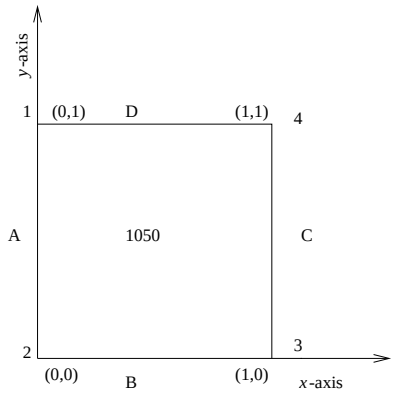


Figure 1.2: Landsat image of Ramsey county, Minnesota, with spatial layers of information superimposed

One *natural* way of storing information about the census blocks, for example, their name,



Census_blocks

Name	Area	Population	Boundary
340	1.58	1839	Poly((5,8),(2,10),...(18,12),(5,8))

Figure 1.3: Census blocks with boundary ID:1050

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 which have a distinct identity are represented as relations or tables. A relation is defined by a name and a list of distinguishing attributes which characterize the relation. All instances of the relation 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 here 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 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 blocks 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 object-oriented software paradiam.

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

Census_blocks

Name	Area	Population	boundary-ID
340	1.58	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.

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 vs. object-oriented within the database community parallels the debate between vector vs. 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 i.e., 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:

DRAFT

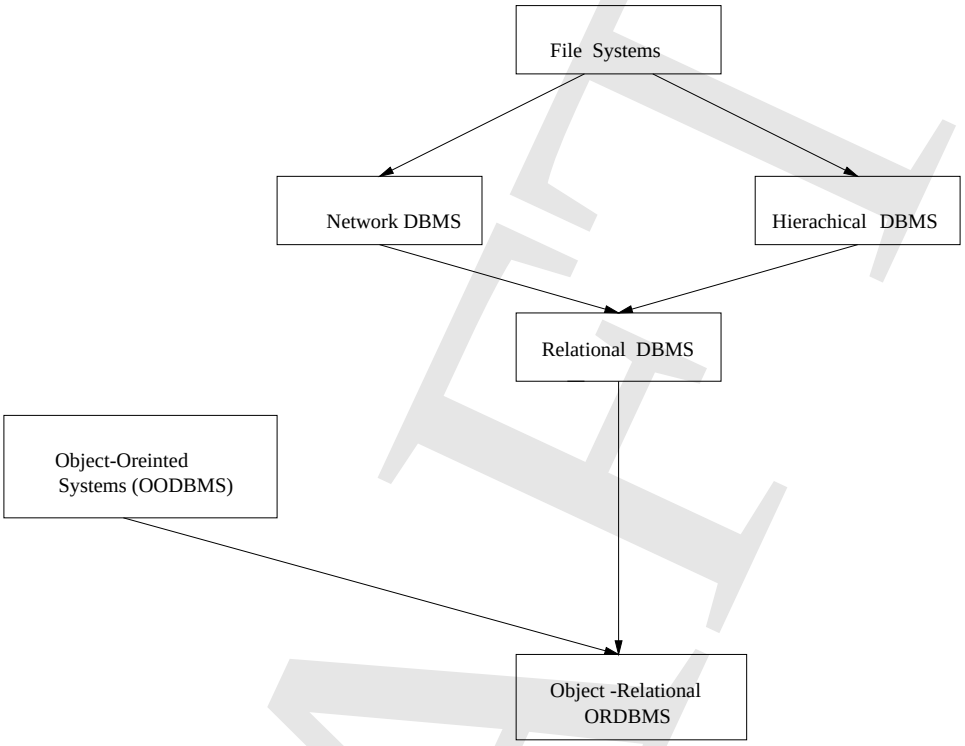


Figure 1.5: Evolution of databases [Khoshafian and Baker, 1998]

```
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 current standard, SQL-92, supports user-defined functions, and SQL-3 , the next revision will support ADTs and a host of data structures such as lists, sets, arrays, and bags. Relational databases which 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. While 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 will be to specialize 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. A spatial database management system is a software module that can work with an underlying database management system, e.g., OR-DBMS, OODBMS.

DRAFT

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.

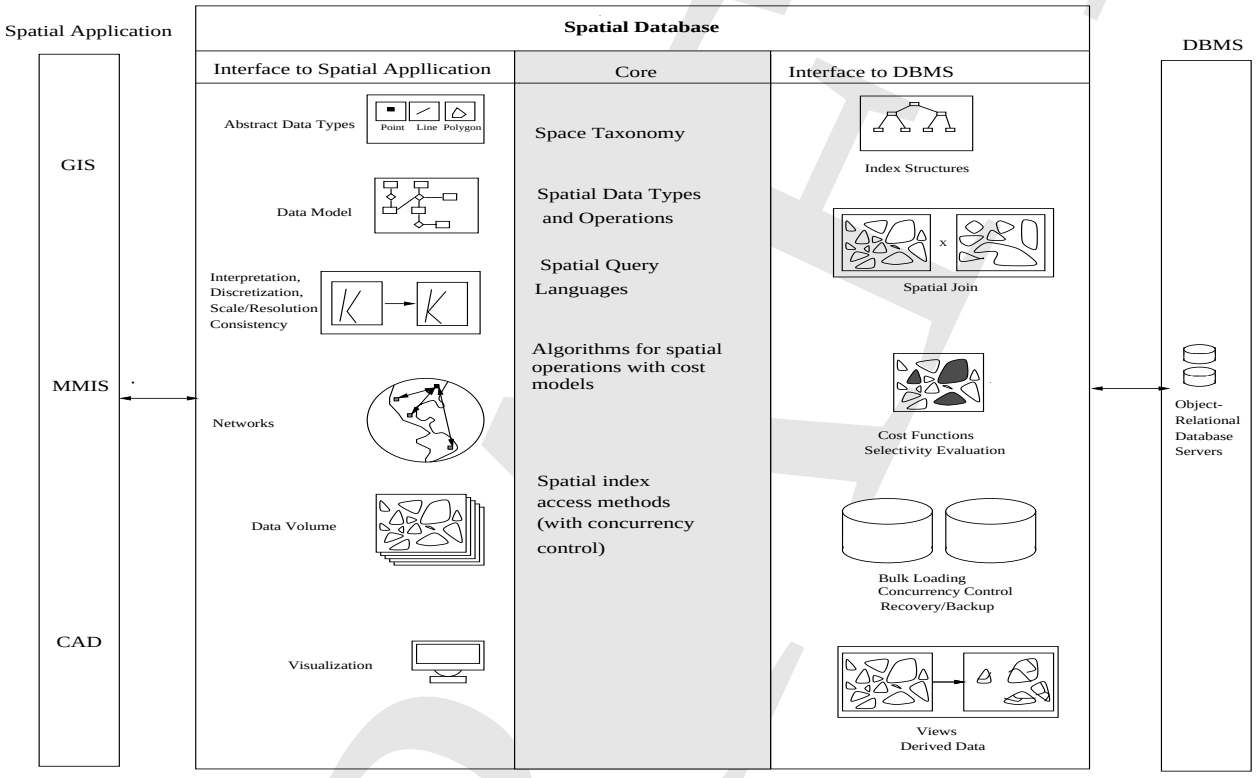


Figure 1.6: Three-layer architecture

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". The middle layer is where most of available spatial domain knowledge is encapsulated, and this layer is "plugged" into the OR-DBMS. No wonder commercial OR-DBMS products have names like 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 invisible to a casual user:

1. *Persistence:* The ability to handle both transient and persistent data. While 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 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 bad in estimating distances, maybe only slightly better in retaining direction and orientation, but (we are) fairly good when it comes to remembering *topological* relationships like *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. "Coffman Union is to the southeast of Vincent Hall." The shapes of the buildings play almost no role in the relationship "southeast". We could easily 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 upon why we are inter-

ested 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 universal description (model) of space which can answer all queries.

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

Table 1.2: Different types of spaces with an example operation

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 upon 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 discontinuous spatial entities like 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 entities, for example, the temperature field. A field is a function which 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. Since field values are spatially autocorrelated (they are continuous), the value of each cell is typically the average of all of 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 chapter 2 and 8).

1.6.2 Query Language

From our discussion so far, it is obvious that the current functionality of SQL 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 industry-wide endeavor to formulate a standard to extend SQL goes by the name SQL-3, the current standard being SQL-2. SQL-3, though still in draft form, will provide support for ADTs and other data structures. The draft will only specify 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 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 will give a detailed description of the OGIS standard, along with an example to illustrate how GIS tasks 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 multiscan 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 accessed more than once to complete the join. In the context of spatial databases, 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.

Consider the two relations showed in Figure 1.7. We have two relations: 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 owners 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:

DRAFT

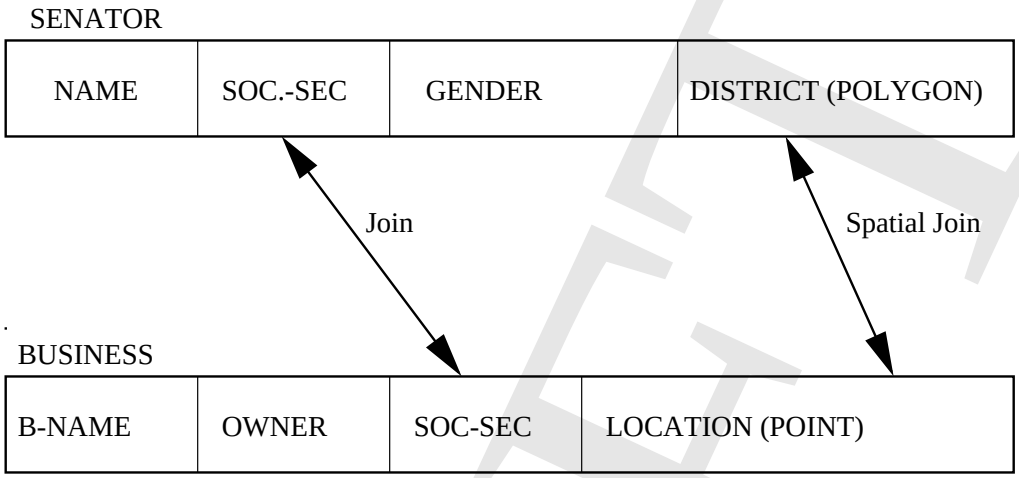


Figure 1.7: Two relations to illustrate the difference between join and spatial join

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

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
        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.

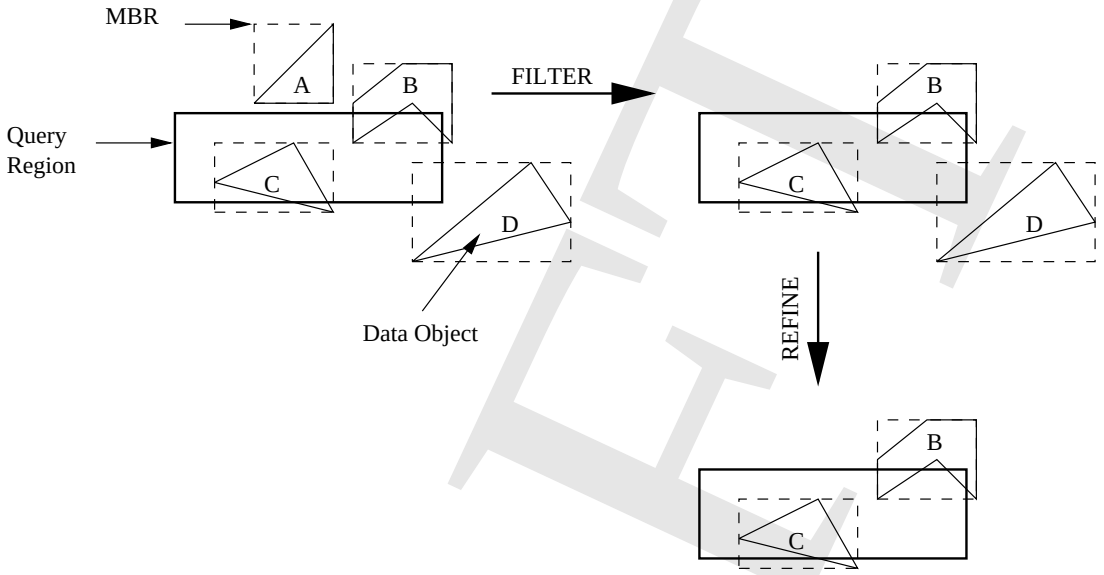


Figure 1.8: The filter-refine strategy for reducing computation time

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 1.9b) $R = \{R_1, R_2, R_3, R_4\}$ 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 1.9a. We sort all the rectangles in R and S according to the x values of their lower-left corners (i.e., on $T.xl$). The sorted rectangles are shown in the first row of Figure 1.9c. 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 1.9a) 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. 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 of 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. Chapter 5 and 7 will present more details of disk-based algorithms for processing spatial queries.

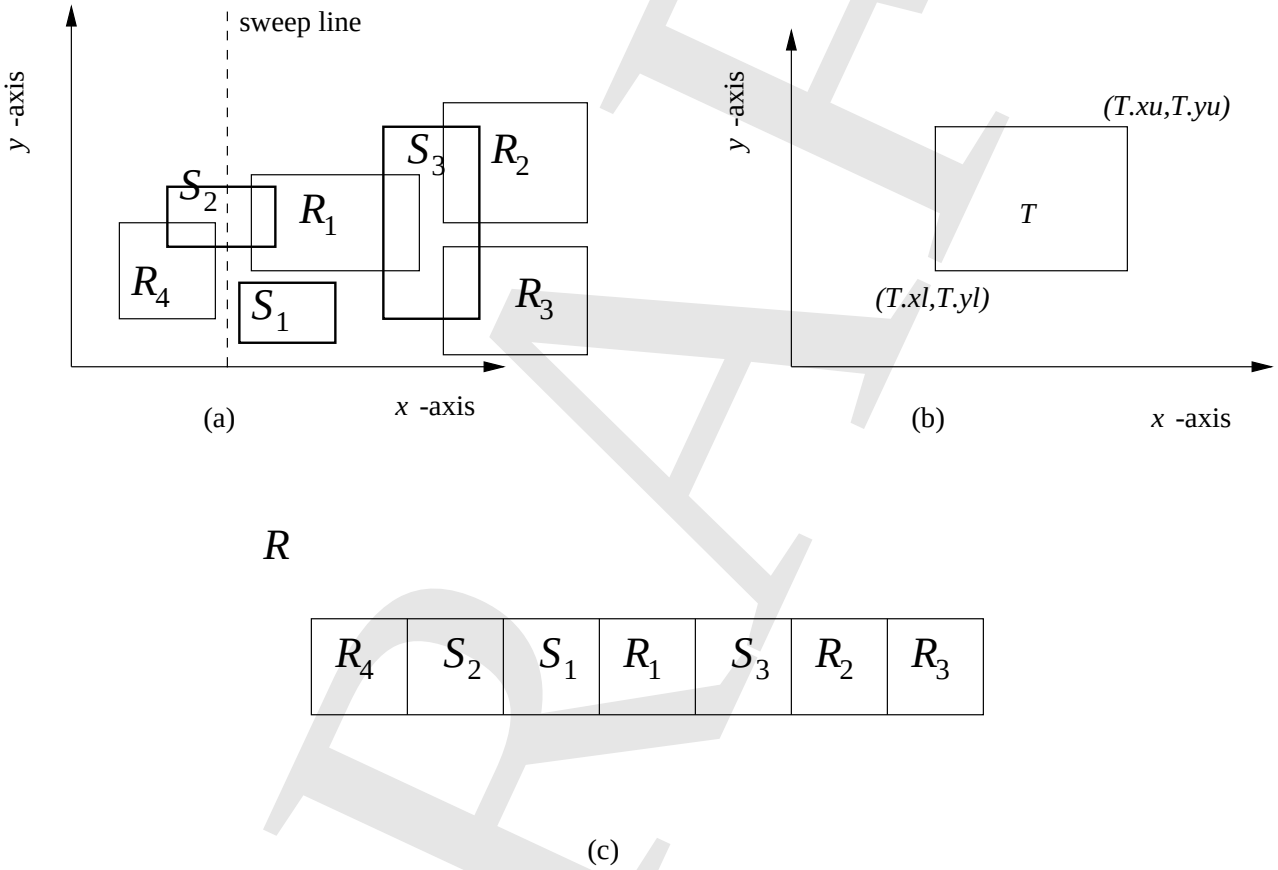


Figure 1.9: Determining pairs of intersecting rectangles. (a) Two sets of rectangles: R and S. (b) A rectangle T with its lower-left and upper-right corners marked. (c) The sorted set of rectangles with their joins. Note the filtering nature of the plane sweep algorithm. In the example, twelve possible 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

Database management systems 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 vs. 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 proportional to 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 to accommodate all of 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 1.10a). On the other hand, for many DBMS designers, the computer has three parts: the CPU, finite main memory, and infinite disk space (Figure 1.10b). While 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 RAM and disk is unbelievably large: a factor of a hundred thousand (year 2000). This ratio is getting worse every year, since the CPU and memory are getting faster at a higher rate than disks and secondary storage devices.

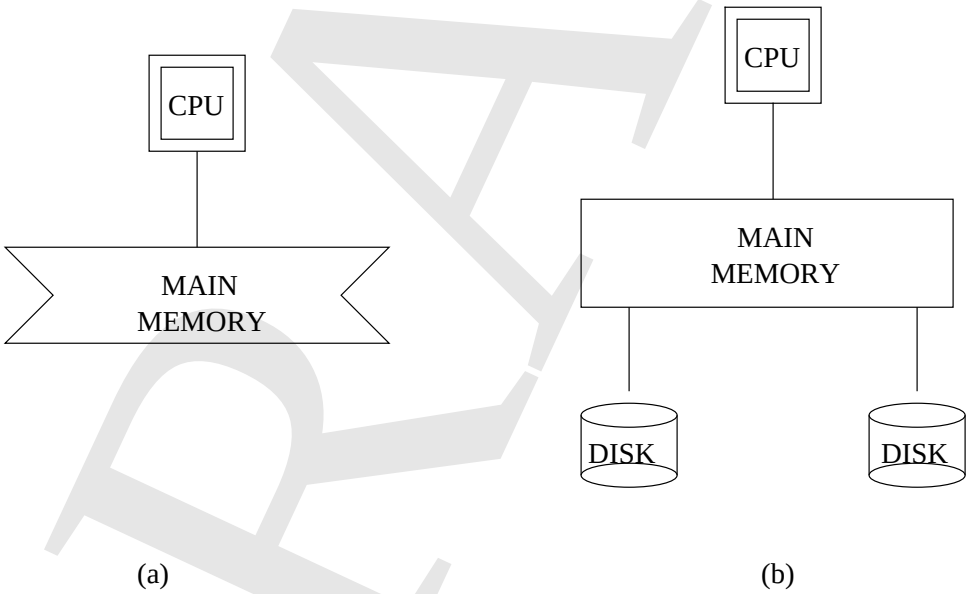


Figure 1.10: (a) Programmer's viewpoint; (b) DBMS designer's viewpoint.

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 1.11a. Here we have a grid which 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 1.11b shows another method. It is worth pointing out that no total ordering can preserve spatial proximity completely.

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.

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 upon the existence of an order in the indexing field. We will discuss B-trees in detail in chapter 4, but Figure 1.12 shows the crucial difference between a binary tree and B-tree. Each node of a B-tree corresponds to a page of the disk. The number of entries in each node depends on the characteristics of the indexing field and the size of the disk page. If the disk can hold m keys, then (we will prove this in chapter 4) 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 siz to create the index. This can drastically cut down on the total number of disk page accesses to process a query.

Since 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 Z-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 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.

DRAFT

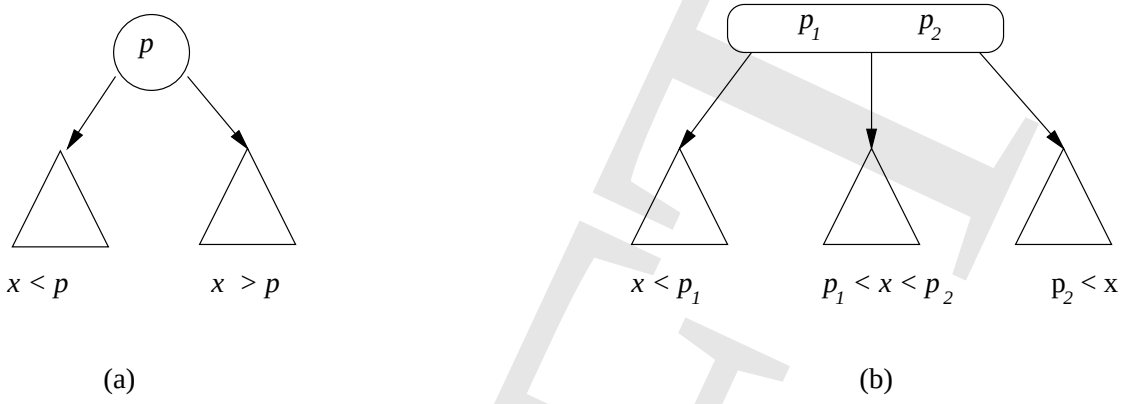


Figure 1.12: (a) Binary tree; (b) B-tree

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 female senators"* is a selection query because we select all of the female senators from 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. So now it is obvious that select should be done before join.

Now 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 subqueries. This important is taken up in Chapters 4 and 7.

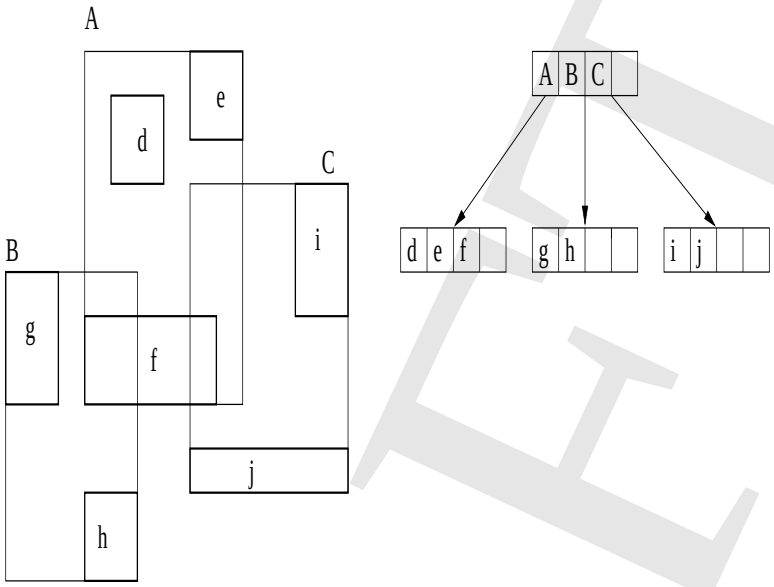


Figure 1.13: R-tree extends B-tree for spatial objects

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 using data-mining tools it was discovered 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:

- By designing algorithms which assume that the data sets are large, or by designing general scaling tools that can be coupled with data-mining algorithms.
- 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.

DRAFT

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 contaminated water source responsible for 1854 cholera epidemic in London [Griffith, 1999], and as well identification of fluoride in drinking water to good dental health. In Chapter 7 we will make an effort to bridge this gap.

DRAFT

1.7 Summary

Spatial database management systems(SDBMS) play a prominent role in the management of query of spatial data. 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 grand scientific problems like 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 size of data sets growing rapidly there is a need for specialized data management techniques which 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 World Wide Web. For example, queries on a search engine can produce results which are spatially localized and more meaningful.

DRAFT

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 spatial databases, see the special issue of very larger data bases (VLDB) journal and the overview in [Guting, 1994a] in particular.

1.3 For integrating a spatial database into off-the-shelf databases systems for CAD applications, see [Kriegel et al., 2001]. For an example of Spatial Database Management System, see [de La Beaujardiere et al., 2000].

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

1.3.3 Extension of SQL for spatial databases 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 spatial databases are discussed in [Adam and Gangopadhyay, 1997, Worboys, 1995].

DRAFT

Exercises

Discussion

1. Discuss the differences between spatial and non-spatial 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 integrating 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 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 services. Provide examples.

DRAFT