Geodata Modeling and Query in Geographic Information Systems

Technical Report

Nabil R. Adam
Rutgers University, CIMIC

October 1996

Contents

Part I

Introduction

Geographic information systems (GIS) deal with collecting, modeling, managing, analyzing, and integrating spatial (locational) and non-spatial (attribute) data required for geographic applications. Examples of spatial data are digital maps, administrative boundaries, road networks, and those of non-spatial data are census counts, land elevations and soil characteristics.

GIS shares common areas with a number of other disciplines such as computer-aided design, computer cartography, database management, and remote sensing. None of these disciplines however, can by themselves fully meet the requirements of a GIS application. Examples of such requirements include: the ability to use locational data to produce high quality plots, perform complex operations such as network analysis, enable spatial searching and overlay operations, support spatial analysis and modeling, and provide data management functions such as efficient storage, retrieval, and modification of large datasets; independence, integrity, and security of data; and concurrent access to multiple users. It is on the data management issues that we devote our discussions in this monograph.

Traditionally, database management technology have been developed for business applications. Such applications require, among other things, capturing the data requirements of high-level business functions and developing machine-level implementations; supporting multiple views of data and yet providing integration that would minimize redundancy and maintain data integrity and security; providing a high-level language for data definition and manipulation; allowing concurrent access to multiple users; and processing user transactions in an efficient manner. The demands on database management systems have been for speed, reliability, efficiency, cost effectiveness, and user-friendliness. Significant progress have been made in all of these areas over the last two decades to the point that many generalized database platforms are now available for developing data intensive applications that run in real-time. While continuous improvement is still being made at a very fast-paced and competitive rate, new application areas such as computer aided design, image processing, VLSI design, and GIS have been identified by many [47, 74] as the next generation of database applications.

These new application areas pose serious challenges to the currently available database technology. At the core of these challenges is the nature of data that is manipulated. In traditional database applications, the database objects do not have any spatial dimension, and as such, can be thought of as point data in a multi-dimensional space. For example, each instance of an entity EMPLOYEE will have a unique value corresponding to every attribute such as employee_id, employee_name, employee_address and so on. Thus, every Employee instance can be thought of as a point in a multi-dimensional space where each dimension is represented by an attribute. Furthermore, all operations on such data are one-dimensional. Thus, users may retrieve all entities satisfying one or more constraints. Examples of such constraints include employees with addresses in a certain area code, or salaries within a certain range. Even though constraints can be specified on multiple attributes (dimensions), the search for such data is essentially orthogonal across these dimensions.

In contrast with the traditional database applications, GIS applications require both spatial and non-spatial objects as data. Unlike a non-spatial object, a spatial object may have dimensions such as length (for lines), area (for surfaces), and volume (for solids). Furthermore, spatial objects have locations identified by a coordinate system. The locational property of spatial objects gives rise to spatial search even for zero-dimensional objects such as points. For instance, a user may request to retrieve all points within a specified radius from a point given as the center. Such a query will require a two dimensional search. If the information on spatial proximity among all point objects is not preserved, the above query will require an exhaustive comparison of the coordinates of the center with that of all other point objects in storage, resulting in a prohibitively expensive query execution plan. The information on spatial proximity will be lost if the point objects are represented in the same way as traditional database objects (for instance, as rows in a relation).

Additional requirements arise with higher dimensional objects. Examples include spatial relationships such as intersection between linear objects, overlap, containment, and shared boundaries between areal objects, incidence relationship between a point and a line, and containment and distance relationship between a point and an area. Simple geometric objects such as line and area can be combined into larger objects. Such objects can be further classified into compound, where the constituent objects are similar, or complex, where the constituent objects are dissimilar [54]. Examples of compound objects are *dyad* (pair of point domains), *network* (collection of curves), *lattice* (collection of points), and *tessellations* (collection of areas). Complex objects can be decomposed into a finite number of constituent domains of different types. An example of a complex object is a spatial unit consisting of a land parcel, a house, and utility network.

In the above discussion we took the *object-based* approach of dealing with geographic data. Another approach used frequently in developing a GIS, called the *field-based* approach, uses a complementary view of spatial information. Instead of associating attributes with individual spatial objects, it addresses the variation of the individual attributes across a spatial domain. This gives rise to the layer-based organization of data, where each layer represents the spatial variation of a specific attribute. Such representations impose additional functional requirements such as polygon overlay and reclassification, which cannot be supported using existing database technology [28, 47].

Since both non-spatial and spatial data are used in a GIS, it requires a seamless integration between spatial data such as extent, location, and orientation, and attribute data such as ownership and valuation of land parcels. Users must be able to retrieve spatial data given set of attribute values and retrieve a set of attribute values for a specified spatial object. Modifications of spatial data such as polygons representing land parcels as well as their attribute values must be supported. This would require maintaining integrity of data if an update is performed. For example, in a GIS for land information, if the size of a land parcel is changed, it would affect that of its neighboring land parcels, and the attribute

data must be changed for all land parcels that are affected by the changes made. Integration of spatial and non-spatial data, and maintaining data integrity for both types poses a significant challenge to existing database technology that has been primarily designed for non-spatial data.

Other challenges include dealing with spatio-temporal data, providing multi-valued logic capabilities, and handling extremely long transactions [15, 47].

The rest of the book is organized as follows: Chapter 2 contains an analysis of the data requirements in various GIS applications. GIS applications are categorized into *object-based* and *field-based* applications, depending on the nature of their spatial information requirements. Field-based applications deal with the spatial distribution of data, whereas object-based applications deal with objects with geospatial references. We show how the data requirements of the object-based applications differ from those of the field-based applications. Within each group, further classification is made by functional areas and the spatial data modeling techniques that are predominantly used. We thus develop a taxonomy of GIS applications based on the nature of their information requirements.

Chapter 3 starts with the operations required for various field and object-based applications. We discuss the differences in these operations and illustrate how each operation is supported more naturally by either representing the application domain as a spatial distribution of certain attributes (fields) or as a number of discrete objects plotted in an Euclidean space. There are fundamental differences between the organizations of spatial data in the field and object based representations. For instance, in field-based applications, the spatial distribution of an attribute is captured by dividing the space into tessellations and studying the spatial variation of attribute values across tessellations. Object-based applications, on the other hand, need to manipulate the spatial extents of geographic objects. Different representations of spatial data give rise to inter-operability issues: how to convert data between different methods. We describe spatial organization methods based on the discretization of space using regular and irregular tessellations, and those for representing the topologies of spatial objects. We describe the methods developed for intra and inter-format conversion methods. We then discuss the issue of integrating the two methods in a way that would facilitate spatial data organization in applications requiring both types of data organization.

In chapter 4 we discuss the usefulness of using database technology for GIS applications. We discuss the various strengths and weaknesses of database technology in dealing with spatial data. We also discuss the various database architectures that have been proposed to deal with applications such as GIS. We perform a requirements analysis by evaluating the needs of GIS applications and then identify the functionalities that can be provided by a database platform. This exercise is geared towards determining the feasibility of a generalized database platform for supporting GIS applications across multiple domains.

Next (chapter 5) we take a top-down view of developing a spatial database for GIS applications. We start with a discussion on the various attempts made in spatial data modeling. The data models are divided into two categories:

application-dependent, and application-independent. We further categorize existing data models into underlying paradigms on which they are based. Two such paradigms are extensions of the entity-relationship and object-oriented models. The data models are evaluated by their ability to support the spatial operations required by both field and object-based applications. In addition, we suggest ways of combining existing data models by drawing upon the strengths and eliminating the weaknesses of each.

Chapter 6 addresses the topic of spatial query processing. We categorize spatial queries into the type of data manipulated, the type of operations performed, and the language in which the queries can be expressed. For each of these, we discuss the studies done in the literature, their shortcomings, and ways of improving them. We also address the issue of optimization of spatial queries, by studying the effect of processing strategy on performance, and possible ways of restructuring to improve operational efficiency.

We next (chapter 7) describe the physical database design issues for storing and retrieving spatial data. Efficient storage and retrieval of data is accomplished by indexing. We discuss the various spatial indexing methods that have been developed for different data types such as point, line segment, rectangle, and volume data. We then discuss what extensions are required to incorporate spatial indexing capabilities in relational databases.

In conclusion, we discuss the open research issues in each of the above areas, and provide future research ideas for possible improvements of the existing methodologies.

Part II

GIS Applications

The purpose of this chapter is to describe a range of GIS applications and identify their data and functional requirements. Due to the wide variety of GIS application domains, different approaches have been taken in representing spatial and non-spatial data. As we have already mentioned in Chapter I, two major types of representation methods for spatial data are called *field* and *object* based representations. Field-based representation methods capture the spatial variation of attribute data. An example of field-based representation is that of the variation of soil characteristics (such as water holding capacity and porosity) over a given landscape. Object-based representation methods, on the other hand, associate attribute data with individual objects, and are suitable for applications requiring the relationships among objects that have both spatial and non-spatial attributes. Examples of spatial objects are property parcels and electoral districts (for land resource management), plant locations (for utilities), point locations of distribution channels (for market analysis), and city landmarks (for car navigation systems).

It is clear that the suitability of a field or object-based representation method depends on the data requirement of an application. Hence we divided GIS applications into field-based and object-based applications depending on the predominance of the type of representation method used. Field-based applications are further subdivided into those dealing with single-factor maps and cartographic functions and others that deal with topographic modeling of terrains. Object-based applications, on the other hand, have been used in a variety of application domains, and hence we divide them by functional areas. The classification of the GIS applications are shown in Figure 0.1.

0.1 Field-based applications

0.1.1 Cartographic Modeling

Cartographic modeling is a method used for manipulating attribute data associated with geographic units depicted in maps [73, 6, 72]. In cartographic modeling, data for a specific study area is organized in the form of *map layers*. The study area is divided into cartographic units defined by Cartesian coordinates. Each map layer is a two dimensional description of the study area in terms of unique attributes of individual locations. Examples of attributes include geological formations, soil types, vegetation, roads, land use patterns, and political boundaries. Locations exhibiting the same characteristic form a *zone*. For example, a map layer could be used to describe the vegetation of a study area by dividing it into four zones: non-vegetation, forest, field, and wetland [73]. Each zone has an associated label and value. The orientation of the layer is specified in terms of its deviation from the north. The smallest cartographic unit determines the *resolution* of the map layer.

Cartographic modeling includes a set of data transformation functions that can perform operations on map layers. Examples of such operations include combining map layers to form new ones, reclassifying zones, and simulating dis-

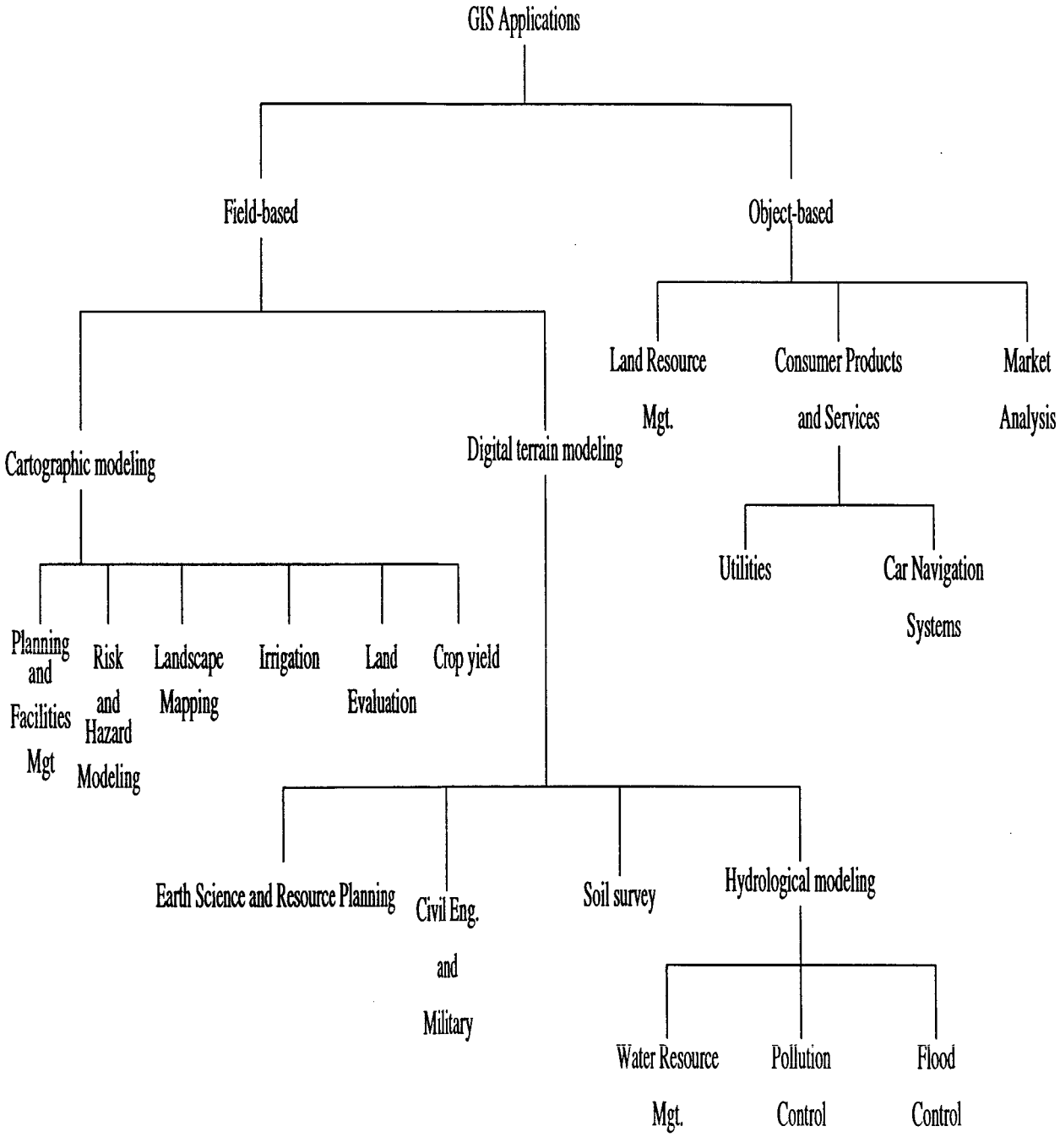


Figure 0.1 Classification of GIS Applications

persions. A sequence of primitive operations can be combined into a complex *procedure* that can model complex phenomena such as soil erosion. The cartographic data transformation operations can be classified into *local, zonal, focal,* and *incremental* operations. Local operations are mathematical functions (such as ratings of the altitudes of locations in an area) that take existing values of a particular attribute of a location and return aggregate measures by applying statistical (such as maximum, minimum, and mean), mathematical (such as product, ratio, and root), and trigonometric (such as arcsine, cosine, and tangent) functions on them. Zonal operations calculate values across locations that occur within the same zone on other specified layers.

Neighborhood operations are those that calculate new values by applying a function to the values of immediate or extended neighborhoods. Such operations include focal operations such as *FocalRating, FocalSum,* and *FocalMean.* Incremental operations deal with the geometry of cartographic forms using immediate neighborhood values. These forms could be *lineal* (one dimensional), *areal* (two dimensional), and *surficial* (three dimensional) features such as *IncrementalGradient* and *IncrementalDrainage.*

Data manipulation in cartographic modeling is primarily based on locational data. Rather than associating locational information with "entities" or "objects", location is used as the primary unit of data and attributes are associated with it. Location based manipulation of data requires functions for generating new attribute values from existing ones. For example, in soil surveys[53], soil profile data are collected from auger borings and soil pits in various soil units. These data are then used to interpolate single-attribute surfaces, which can be used to generate new map layers [14, 12, 13].

Cartographic modeling requires data preparation capabilities such as line digitizing, video scanning, aerial image enhancement, and cartographic reprojection, data presentation such as map drawing and visual simulation, and programming such as device control and error handling.

Example applications:

- **Planning and facilities management:** Planning and facilities management refers to municipal planning and management activities such as school and political districting, facility siting and management, traffic analysis, and administration of public facilities [78]. These applications require various cartographic tools such as overlays, polygonization or creating new districts from existing maps, and matching attribute data such as census data with the corresponding spatial units in maps [25].

- **Risk and hazard modeling:** Hazardous wastes are created by industrial wastes and chemicals, mining, and smelting of pollutants such as zinc, lead, and other chemicals. Cartographic modeling has been applied to hazardous waste removal projects [76]. Such projects consist of two parts: analyzing risks on public health by contaminants in environmental media such as soil, ground and surface water, and air; and forming a remedial

strategy. The first step requires determining pollution levels in soil samples, modeling pollutant transport through ground water and water supply systems (which is referred to as hydrological modeling, discussed later in this chapter), identifying residential locations, and collecting health census data. Cartographic modeling techniques such as reclassification are used for developing maps for high-risk groups from census data. Water system layouts are overlaid with contamination areas to develop pollution concentration areas. Pollutant concentration maps are overlayed with residential location to develop maps of exposed locations. The second step is to develop and implement a remedial strategy. Cartographic modeling is used in optimizing remediation, monitoring progress, and providing logistical support for future remediation.

- **Miscellaneous applications:**

  Other applications using cartographic modeling include irrigation [56] landscape mapping [16], land evaluation [29], and crop yield models [30]. In these applications, geographic data is organized in map layers [73, 6, 72]. Cartographic operations such as reclassification, overlaying, and neighborhood functions are used in these applications.

We summarize in the following the functional and spatial database requirements for cartographic modeling.

Functional requirements:

- **Overlaying:** Cartographic modeling requires *overlaying* of individual map layers to form new ones. Different map layers such as slope, infiltration, and water holding capacity may be overlaid to form new layers. Such overlay operations require combining attribute data in a manner specified by the user, and then generating and storing maps for the derived attribute values. Another example of overlaying operation is the digital integration of geoscientific maps used in applications for geological survey [9, 1]. In these applications, several factors are combined using Bayesian methods to develop combined probability estimates for occurrence of mineral deposits in geographical survey areas. The input factors are represented as input map layers and are assigned "weights of evidence". This method has been used in identifying gold deposits [2, 10, 11], and seismic epicenters [9].

  Overlay operations can produce combined values from input maps on a point-by-point basis (called location-specific overlaying), or assign the same value to entire thematic regions (called category-wide compositing) [7]. Since input maps can be qualitative (such as soil types and land use) as well as quantitative (such as income and age for demographic maps), a number of different types of functions are required to combine values from input maps into meaningful combined data.

- **Reclassification:** As the name suggests, reclassification refers to regrouping or "purposeful recoloring" of maps [7]. Such methods are used to

generate new information of qualitative or quantitative nature from existing data. Examples include generating a binary map from a multi-color coded map to emphasize a particular category, generating a map of contour lines from a map of elevation data, and travel-time isochronic (equidistant) maps, for a given center, from a map showing political boundaries.

- **Distance measurement:** Another functionality required in cartographic modeling is measuring distance between locations in maps. Distance measurement often requires dealing with constraints such as *relative* and *absolute* barriers [7]. Relative barriers refer to restrictions that increase travel time/cost. An example of a relative barrier is speed limits in streets that can increase travel time. Absolute barriers refer to complete restrictions to passage and are equivalent to "infinite distances". An example of an absolute barrier is a one way street for an automobile entering from the wrong end.

Spatial database issues:

- **Data capture:** The first step in developing a spatial database for cartographic modeling is to capture map data into digital form. Some of the issues related to data capture that complicate the development of a digital map database include: the characteristics of the source map, cartographic license, map scale, coordinate systems used, and the method of data capture [50].

  Source map characteristics refer to the scale and projection of the source map which determines the nature of the digitized map and hence the cartographic operations allowable. Cartographic license refers to alterations, such as displacing objects that overlap, and eliminating details from the source map that are judged to be insignificant (also called generalization), Although such practices are acceptable in manual cartographic operations, they can introduce locational errors in spatial databases, because of the much higher level of precision with which map information is stored, once they are digitized.

  The degree of precision of digitized maps being higher than that of the source maps, disparities can be introduced in the process of digitizing source maps. For example, a line that represents a ground feature (such as a river) in the source map may be much thicker than its digital counterpart. This results in erroneous representation of geographic features such as rivers, roads, and boundaries. Also, since different source maps may have been drawn to different scales, such maps cannot be overlayed as their differences in scales will introduce errors.

  An alternative to digitizing hardcopy maps manually or through automated scanning methods is to capture spatial data from remote sensors in satellites such as Landsat, NOAA AVHRR (Advanced Very High Resolution Radiometer), and SPOT HRV (High Resolution Visible Range Instru-

ment) [50]. Data from remote sensors are free from interpretive bias and are of higher degree of accuracy.

- **Data integration:** Data integration refers to combining data received from different sources as well as linking data of different types. As we have shown before, different sources are used for cartographic data collection. Since several coordinate systems such as the Universal Transverse Mercator, Global Positioning System, and local cadastral system are used in identifying locations, integrating information from different source maps requires transformation of the coordinate system of the source map to the one adopted for the specific spatial database in use.

  Another aspect of data integration is creating and maintaining link between spatial data and attribute data associated with it. For example, a map layer containing demographic information will require a link between the polygonal units in the map that represent different political boundaries with the demographic information relevant to them. In addition to establishing such linkage, it is also necessary to maintain it if changes such as redefinition of political boundaries should occur.

0.1.2 Digital Terrain Modeling:

Digital terrain modeling (DTM) refers to the digital representation of a portion of the earth's surface, where land elevations at sample points are linked together to form a simply connected surface model [77, 68]. Although usually used as an elevation model, the DTM can also be used as other single-valued surface models such as air temperature and population density over a geographical area by simply substituting elevation with the corresponding parameter.

DTM combines the various elevation data taken at the sample points into a collective *data structure*. Two of the most common data structures are the *rectangular grid* (also called the elevation matrix), and the *Triangulated Irregular Network* (TIN). Rectangular grids are two-dimensional array structures and hence suitable for computerization. However, the static dimensions of the structure makes it impossible to vary the number of sample points with the complexity of the terrain (i.e., more sample points for rough terrains). TINs, on the other hand, model the surface by connecting the sample points as vertices of triangles. Thus, the point density can be varied with the roughness of the terrain. TINs can model a terrain more accurately with smaller number of data points than rectangular grids. However, terrain modeling algorithms are simpler for the rectangular grids structure than for TINs. DTM systems, therefore, are required to be able to export/import between these data structures and refine a DTM.

DTMs contain valuable information that are used as input in many GIS applications such as studying soil erosion, environmental impact, and hydrological runoff simulations. Analysis of DTM data are categorized into general *geomorphometric* and specific geomorphometric analyses respectively, depending on whether the characteristics under study are applicable to any continuous

rough surface or specific landforms. General geophormometry involves extraction of slope values from a DTM. Slope values include *gradient*, which is the maximum rate of change in altitude; *aspect*, which indicates the compass direction of the gradient; *profile convexity*, which is the rate of change of gradient; *plan convexity*, or the convexity of contours; and other parameters. Specific geomorphology involves extraction of terrain features related to surface hydrology such as drainage channels and basins, and topology of channel networks, and descriptive attributes such as mean slope and basin area.

Example applications:

The range of applications using digital terrain modeling is vast and varied. The main application areas, however, have been classified into five categories [77] that include: surveying and photogrammetry, civil engineering, planning and resource management, earth sciences, and military applications. Photogrammetry and surveying applications have a relatively limited use of DTM in that there is little emphasis on analytical techniques and more on terrain data capture, quality assessment, and generation of topological maps for other applications. Applications in Earth science and planning and resource management areas require similar functionalities and hence we group them together. Similarly, we group civil and military applications together due to their similarities in data and functional requirements.

- **Earth science and resource planning applications:** Earth science applications include geological, glaciological,and hydrological (which we treat as a separate group because of its significance) applications that use DTM techniques for studying terrain discontinuities such as drainage basins and network divides, as well as geological interpretations.

  Resource planning and management includes applications such as urban and environmental planning, agriculture, and forestry. DTM functions are used for pollution dispersion models for industrial site selection, harvesting strategy, and crop suitability. The DTM functions used include data verification, interpretation, visualization, and support of TIN and raster structures.

- **Civil engineering and military applications:** Civil engineering application include road and airfield design, and site planning for dam, reservoirs, and open cast mining. The DTM functions required by these applications include visibility analysis, relief shadow analysis, profile computation, and volumetric computation.

  Military applications include battlefield management site planning, and vehicle trafficability analysis. These applications use DTM functions such as intervisibility analysis and advanced visualization functions for flight simulation.

- **Soil survey:** Soil data is gathered for developing crop yield models, performing studies on experimental agricultural farms, soil pollution, and

fertilizer placement. Soil data includes quantitative and qualitative descriptions of location, geological formation, soil profiles such as thickness, color, and porosity. The area under study is divided into polygonal units by drawing boundaries around areas with similar soil characteristics. Soil data can be gathered only at sample points, called soil profiles, whose coordinates are stored along with attribute data at the soil profiles. A unit polygon represents an area of similar soil characteristics. A soil unit consists of a number of non-contiguous units polygons having the same soil characteristics. Soil attributes vary abruptly and at short ranges. An interpolation method called *kriging* is used to optimize the sample spacing and simulate the spatial variation of soil characteristics. Functionalities used in soil soil survey applications include reclassifying soil polygons by attributes such as texture class, color, and thickness, overlaying with other coverages such as land use and climatic conditions to form composite overlays that are used to determine new characteristics such as soil suitability and pollution.

- **Hydrological modeling:** Hydrological modeling refers to the study of the flow of water above and below the ground surface. Hydrological modeling starts with studying the discharge of rain water through runoff, groundwater discharge, storage, filtration, and evaporation. Hydraulic parameters such as flow velocity, elevation, and flux; and physical and chemical processes such as degradation, adsorption and oxidation are used to determine the transport of both insoluble substances such as oils and sediments, as well as dissolved toxic chemicals and nontoxic biological constituents [58].

  Terrain and land use data are used in modeling fluid flow in hydrological modeling. These data, in turn, are obtained from grid, TIN, and contour models. The topological representation of the flow domain are used as input to two dimensional finite element and finite difference methods for flow computation.

  Hydrological modeling is used in applications such as water resource management [52], pollution control, and flood control [58].

Functional requirements:

- **Interpolation:**

  An important functionality requirement of DTM is *interpolation*, which refers to the process of deriving elevation data for points at which no data sample have been taken. Interpolation includes single-point computation, computation of a rectangular grid, computation along a contour, and resampling (densification or coarsening) of rectangular grids. Most of the interpolation methods are based on *triangulation*, a method that uses a TIN structure for interpolating the elevation of a point inside the triangle using a linear or polynomial function of the elevations of the vertices of the triangle. Interpolation over a large area is carried out by dividing the

area into smaller regions (a process called *tiling*) and subsequently *joining* sub-areas into larger ones. Other types of interpolation use digitized contour data for linear or polynomial interpolation. Various algorithms exist for such interpolation methods.

- **Visualization:** Visualization refers to the graphical display of terrain information and DTM operations. The major visualization techniques include contouring, hillshading, generation of orthophotos, and perspective display. Contours (also called isolines) refer to spatial units of lines or arcs of equal attribute values (such as altitude) [54]. DTM interpolation methods are used for constructing of contour lines. Visualization methods for contour lines include raster contouring based on individual grid cells, inclined contours, and shaded contours, both of which produce three-dimensional displays.

  Hillshading refers to a method of illumination used for qualitative relief depiction, that uses varied light intensities for individual DTM facets [77].

  Orthographic displays or orthophotos are developed by overlapping areas and eliminating image distortions. Such displays present an undistorted image of all parts of terrain data, and can include contours, hillshading, results of interpretation, and DTM components.

  Perspective displays produce three dimensional images of DTM facets using perspective projection methods based on computer graphics [77]. These methods provide visual interpretation and analysis of terrain data by overlapping cartographic data and other three dimensional objects with terrain data, and providing animated scene renderings used in flight simulation and computer generated films.

Spatial database issues:

- **Data capture:** Methods of data capture for DTM includes *ground surveys*, *photogrammetric* sampling methods for data collection, and digitizing cartographic documents using manual, semi, or fully automated methods. Among these, ground surveys are the most accurate but also the most time consuming means of collecting digital terrain data.

- **Interpretation:** Digital terrain modeling includes various operations on terrain data such as *editing*, filtering, joining and merging, and converting between terrain models. Editing a DTM involves selecting a part of a terrain and modifying selected properties of individual elements (such as changing heights). Filtering a DTM involves *smoothing*, which involves reducing details or *enhancing*, which involves adding details. Joining involves combining adjacent models and can be applied to grids with corresponding resolution and orientation, or to TINs by patching (also called *zipping*) their borders. Merging implies combining overlapping models and may require resolving conflict among attribute data. Conversion of terrain

models are needed because of the existence of a number of such models such as the grid model, the TIN, and the contour model.

0.2 Object-based applications

0.2.1 Land resource management:

Applications in this category include urban studies [49], development and implementation of national guidelines for land management [20] management of natural resources [5, 8], evaluation of tourism facilities and impact on local economy, and managing census of housing and population data.

A land information system (LIS) is a computerized tool that facilitates, in a systematic manner, the collection, storage, and dissemination of data related to land resources. LISs have been used in national policy making, development and implementation of national guidelines for land management, management of natural resources, evaluation of tourism facilities and impact on local economy, and managing census of housing and population data.

An LIS consists of textual (attribute) data such as land ownership and valuation, and spatial (graphical) data such as road and utility networks, that are linked through unique identifiers assigned to spatial units such as land parcels and resource polygons.

A major bottleneck in developing LISs is converting traditional spatial data into computerized forms (such as digitizing maps) because of the volume of such data available and the lack of cost effective and fast data conversion equipment ([24]). Attribute data definition can sometimes have inherent ambiguities. For example, the definition of a land parcel may be different for legal and tax purposes. Surveying and mapping land parcels are also costly and time consuming.

The following are examples of some of the functionalities that are required by an automated LIS:

Functional requirements:

- **Map analysis:** Map analysis is performed for various purposes such as identifying conservation areas, determining accessibility to reads for areas under development, supporting ecological research, and estimating land use [6].

- **Conflict resolution:** Since land development may be done for multiple purposes such as residential development, tourism development, and ecological conservation, conflict resolution models are used to determine the best possible use [6]. Examples of such models include "hierarchical dominance", "multiple use", and "trade off", that are based on assumed preference on land use, identifying and accommodating compatible uses, and selecting the best uses on a parcel by parcel basis.

Spatial database issues:

- **Creating and maintaining spatial objects:** The primary objects for land information systems are property parcels, environmental and resource polygons, road and utility networks, topographic data, and census and electoral polygons. The creation, retrieval, and maintenance of these spatial objects is fundamental to an LIS.
- **Integrating attribute and spatial data:** An LIS would require the maintenance of spatial data such as spatial extent, location, and orientation, and attribute data such as ownership and valuation of land parcels. Users must be able to retrieve land parcels given set of attribute values and retrieve a given set of attribute values for a specified land parcel. Modifications of land parcels as well as their attribute values must be supported. This would require maintaining integrity of data if an update is performed. For example, if the size of a land parcel is changed, it would affect that of its neighboring land parcels, and the attribute data must be changed for all land parcels that are affected by the changes made.
- **Ambiguities:** Since the definition of a land parcel may vary according to its usage, there must be a systematic way of storing and retrieving multiple values for the attribute data for a given land parcel.
- **Visual inspection:** Users must be able to issue data retrieval as well as update requests visually. This would require the development of a visual query language that supports a set of predefined visual operators. A systematic way of developing such operators and a language for data manipulation is required.

0.2.2 Consumer products and services:

Examples of these applications include utility networks [35, 57] and consumer products such as car navigation systems [21, 80].

Utilities:

Utility companies provide services such as gas, water, electricity, telecommunication, cable television, and underground waste disposal to private and commercial customers. All of these services require pipeline and/or cable networks for the region that is serviced. Examples of data kept for utilities include plant location, data about pipelines such as diameter, material, maximum pressure, and data about transmission lines for electricity such as operational voltage, material and insulation for electrical wiring. These records are associated with a base map of the serviced area for maintenance and enhancement purposes. A significant amount of utility work involves expansion of the utility network and laying out of new plants for new developments. Utilities require to keep map-based records of distribution networks for planning expansion, forecasting demands, and locating plants for maintenance purposes.

A spatial database for utilities would be required to provide its users the ability to design and maintain a utilities network. We discuss in the following some of the functionalities and database requirements for such systems:

Functional requirements:

- **Spatial manipulation:** A GIS for utilities require manipulation of spatial information for purposes such as network design to manipulate cables, pipelines, and electrical lines. Other examples include tracing the position of a fault in a network, and locating free duct space for installing new cables. Such query capabilities would require establishing a language for specifying the inputs and performing operations such as intersection between input and existing lines segments, measuring spatial orientations, and extents.
- **Historical information:** Digital maps for past, present, and future developmental plans are required for plant maintenance and planning purposes such as plant replacement and engineering inquiries [57].

Spatial database issues:

- **Hypothetical queries:** An important aspect in designing a utilities network is to predict the behavior of the network under certain conditions such as overload. Thus users must be able to issue hypothetical queries to simulate conditions and observe the reaction of the existing network. Such activities can be performed for planning as well as monitoring purposes.
- *Database links:* Monitoring the loading patterns of networks can be achieved through links established between the actual network and a computer database. Automated control systems can be developed with programmable logic controllers to monitor the network loading patterns.

Car navigation

Car navigation systems [21, 80] include computer controlled equipment for land-based navigation activities such as determining locations in a map, displaying maps and generating textual and verbal route instructions, and providing guidance towards destinations. Location determination is accomplished with wheel rotation sensors and solid state compasses that measure the distance traveled from a starting position. Change in direction is measured from the difference in distance traveled on two opposing wheels. Radio location sensors are another type of sensors that consist of on-board receivers and computers that calculate the location of the vehicle by receiving signals from transmitters with known location and signal propagation times. All of these sensors suffer from errors due to sensor noise, bias, measurement anomalies, and distortion and blockage of signals in urbanized areas. One method of reducing these errors is *map matching*, which integrates navigational data collected with the roads and intersections depicted in the map.

Road maps are displayed on CRTs mounted in the dashboard. An important consideration in map display is that the map should be oriented such that the top of the screen corresponds to the heading direction, so that it is non-distracting and easy to use for the driver. Another aspect of map display is scaling, where the road maps may be displayed in different scales to suit the need of the driver.

Destination finding involves searching through a map database for a user specified destination such as the nearest restaurant, and displaying it in the map. Since keyboard input is not feasible for the driver, simplified input mechanisms such as buttons are used. Pathfinding is another use for car navigation systems which require the selection of the *best* path to a destination. This requires various information including road networks, traffic flow conditions, and turn restrictions.

Digital representation of maps can range from bit-mapped images, which can only support display but not destination or pathfinding, to vector-based encoding, which can support pathfinding, to geometrical encoding that support map matching and display. Data requirements include attribute data, such as name and address, map displays, and topological data for pathfinding. Map displays require continuous orientation of the display with the vehicle movement, which is memory and processor intensive.

Some of the functional and database requirements that car navigation systems require are discussed below.

Functional requirements:

- **Map display:** The location of the car in motion needs to be continuously displayed, which requires real-time linkage between the tire rotations, map database, and map display.

- **Specialized operations:** Destination and path finding are examples of specialized operations that are performed in car navigation systems. Destination finding would require identifying and displaying the destination point on a map, using spatial indexing methods. Proper links must be established between path the finding algorithm, map database, and attribute data such as traffic volume and turn restrictions.

Spatial database issues:

- **Comparing and correcting spatial data:** Many car navigation systems use map matching to reduce errors in data collected through dead reckoning systems. Using a spatial database would enable such systems to efficiently maintain and retrieve spatial data. However, such a database would require the capabilities to automatically load data from wheel sensors, compare them with the map database, and make suitable adjustments to correctly locate the position of the vehicle.

- **Real-time update of information:** Since car navigation systems deal with dynamic data such as traffic flow, real-time update operations may

need to be performed. On the other hand, statistical information such as typical traffic volumes during specific hours may be used instead of actual data, which may need to be dynamically modified in special situations such as accidents on a certain route.

0.2.3 Market analysis

Marketing deals with identifying potential customers, reaching customers through advertising and sales promotion, and maximizing sales through distribution channels [4, 27, 51]. Market analysis involves identifying customer bases, analyzing sales patterns, matching potential customer profiles with product characteristics, and optimizing the locations of branches to reach customers effectively. The primary need of a GIS in market analysis is derived from the geographically and temporally dynamic nature of the demand and supply of products and services. For example, customer demands constantly change due to the arrival of new resident and commercial customer groups. Similarly, supply of products and services change due to the arrival of new businesses offering various mix of merchandise. The dynamic nature of the demand and supply of products and services makes it necessary to constantly revise competitive strategies to reach customers through advertising and distribution. GIS have many applications in market analysis. For example, in order to establish new site locations for retail stores, catchment areas have to be identified. Increasingly, travel time isochronic displays in maps, which represent areas that are equidistant in terms of travel time from the store site, are used to supplement survey-based methods for identifying potential catchment areas. In addition, locations of competing stores, pricing, and product preferences of customers are other factors that influence site location decisions. The GIS functionalities required for market analysis include database management functions, cartographic operations, and spatial analyses.

Some of the functionalities and database requirements for marketing applications are discussed below:

Functional requirements:

- **Different resolution level displays:** Data displays at different levels of resolution are necessary for market analysis. For example, service areas may be needed to be displayed at the state level, service locations may be needed at the county level, and customer addresses may be needed at the street level. If data at different levels of resolution are stored separately, the inclusion relationship between spatial data (such as between states an counties) must be maintained. To continue with the example, a user should be permitted to request for customer address information from a state or county level map, which would imply that the spatial relationship between state/county and street must be maintained in the database.

- **Forecasting:** Various forecasting methods are used in determining sales

for specific products and services. Such methods make use of the information about local market demands and competitor information and include simple and complex models such as "fair share", multiple regression, and spatial interaction models [4].

Spatial database issues:

- **Linking attribute and spatial data:** Market analysis requires generation and display of maps based on attribute data. For example, a user may request for a map that displays the service areas covered by an existing utility company. Such requests would require an efficient linkage between the attribute data and their spatial counterparts stored in the database.
- **Branch location analysis:** Such analyses are important for planning purposes. For example, in order to analyze the suitability of a store location, a user may require to see travel time isochrones around a hypothetical point. Such queries would require displaying isochrones on a real-time basis on a map, which in turn would require retrieving spatial data (the shape, size, and orientation of the isochrones) from attribute data (travel time distance).

Part III

Representation and Manipulation of Spatial Data

In this chapter we present a number of methods for spatial data representation and manipulation. In keeping with the classification scheme of application domains into field and object-based applications, we divide these methods into field and object-based representation and manipulation methods. In field-based representation, the geographic space is partitioned (tessellated) into cells that are disjoint and that colectively cover the entire geographic region under study. Geographic objects are embedded in the space, and are described as well as manipulated in terms of the individual cells. For example, in this representation scheme, a lake would be described by the cells that cover its interior, rather than the polyline constituting its boundary. In this sense, the tessellation model is area oriented where the emphasis is on the content of the area rather than its boundary. Tessellation models can be further subdivided into fixed and variable spatial resolution models [54]. Fixed spatial resolution models use a geographic data structure called *raster*, which consists of polygonal units of equal size, These units are used to describe geographical features. Variable spatial resolution models use units of variable sizes at a given level of resolution.

In the object-based representation methods, individual objects are represented by some geometric counterpart. The explicit representation of the spatial extents (such as polylines representing the boundary of a lake) using directed line segments have given rise to the term *vector model.* The vector model describes geographical features in two or three dimensional coordinate systems. In this representation scheme, a polygon is represented by its boundary (which is often discretized into a number of straight line segments). In this sense the vector model is boundary-oriented. This model is more suitable for applications requiring high quality cartographic operations, coordinate geometry, and networks.

The rest of the chapter is organized as follows: we describe the representation of field-based and object-based data in Sections 0.3 and 0.4 respectively. Section 0.5 discusses the various methods of converting data from one representation to another. Lastly, Section 0.6 discusses some of the issues that need to be addressed in spatial databases with respect to data representation and manipulation.

0.3 Field-based representation and manipulation

0.3.1 Representation of field data

Fixed spatial resolution model

As mentioned before, the fixed spatial resolution model uses a data structure called the raster [71], which is a cellular or grid structure that models a specific data layer for a geographic area by decomposing it into units of equal size represented by a raster cell. Each cell in a raster structure has two associated values: a positional value that marks its identity, and an attribute value of the underlying geographic area it represents. Thus, the raster structure is a scaled representation of a geographic area where the scale is the ratio of the size of

a cell to the geographic unit it represents. Thus, if each side of a cell is one centimeter in length and represents a stretch of one hundred meters on earth surface, then the scale of representation is 1:10000. Since each cell has one associated attribute value, an inherent assumption is that the attribute value is uniform over the geographic unit it represents. In reality this is hardly ever the case because attribute values such as land elevation, cover, and use, constantly change. In variations of the raster model, the average, maximum, and minimum values are also stored for each cell.

In raster structures the cells could be rectangular, square, triangular, hexagonal, or any other geometric shape. All of these geometric shapes that completely cover the surface are called *tessellations*. Each of these tessellations have their own advantages and disadvantages. For example, hexagonal tessellations have the advantage that all adjoining cells are equidistant from any cell in the structure, which enhances searching through the data in spatial applications. This is not true in the case of square or rectangular tessellations, because the diagonal cells are more distant that the ones on the sides or those above and below. One major advantage that square tessellations have over others is that such a raster structure can be recursively decomposed from less to high resolution models, where resolution refers to the level of detail captured by a particular structure. For example, a given geographic area can be divided into four quadrants, and each of these quadrants can be successively decomposed into four smaller quadrants. In this type of decomposition the shape of the cells remain the same, while the same geographic area can be modeled at different resolutions, which gives rise to a hierarchical raster structure. Since tessellations of other shapes do not have the property of retaining the same shape if they are subdivided into smaller cells, a hierarchical raster structure is more easily obtainable using the square tessellation model.

Variable Spatial Resolution Structure

The variable spatial resolution structure allows selective decomposition of a geographic area. If the area is decomposed into four quadrants at each level of resolution, the structure can be represented by a *quad-tree*. A quad-tree is a hierarchical data structure where each level is successively decomposed into four quadrants to form the layer at the next level. As in the raster structure, attribute values are associated with each cell in a quad-tree. It is a variable resolution model since at a given resolution level a cell is further decomposed only if there is internal variability of attribute values within that cell. When the presence or absence of a given attribute value, such as land use, is stored, it is called a binary incidence representation. However, in addition to attribute values, if points, edges of line segments, and vertexes of polygons occur within a cell, their coordinate values are also stored. Each cell can be assigned a unique key value based on its location. One such key assignment is based on on space filling curves such as the Peano curve.

An example of a variable resolution data structure is shown in Figure 0.2, where 0.2(a) represents a partial map of a fictitious golf course, where the vertices

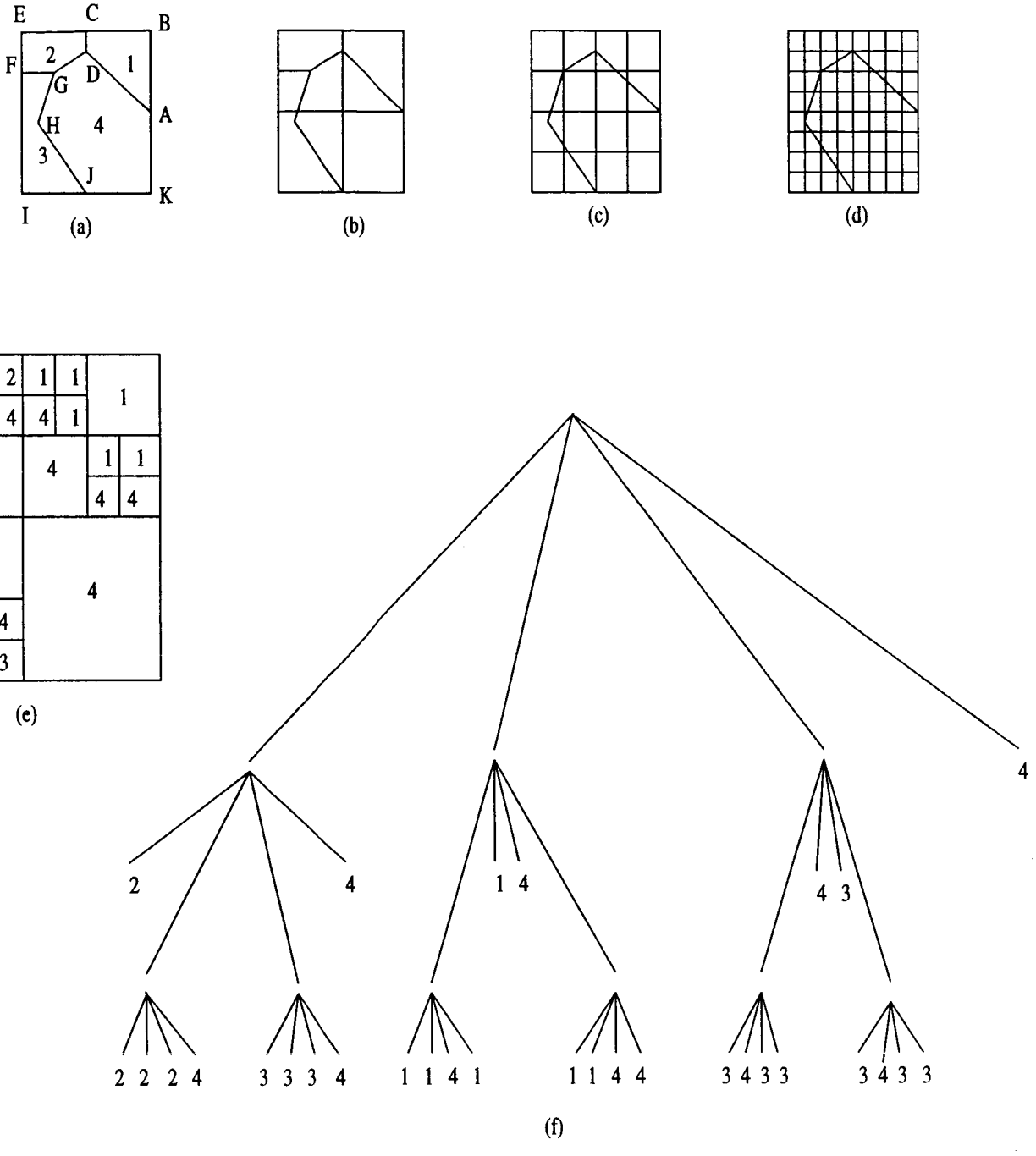


Figure 0.2 A partial map of a golf course

are shown with letters A through K. Figures 0.2(b) - 0.2(d) represent levels 1 to 3 of the hierarchical raster structure corresponding to the map. The golf course is divided into four types of land covers: tee, bunker, rough, and fairway, which are numbered 1, 2, 3, and 4 respectively. The map of the golf course is divided into four quadrants in Figure 0.2(b): north-west (NW), north-east (NE), south-west (SW), and south-east (SE), and each of these quadrants are further subdivided into four quadrants in Figure 0.2(c). Thus, there are 16 cells in the raster structure in Figure 0.2(c). Each of the cells in Figure 0.2(c) are further subdivided into four quadrants, giving rise to 64 cells in the raster structure in Figure 0.2(d). Figure 0.2(e) is the *maximal block representation* of the raster structure depicted in Figure 0.2(d) and eliminates all redundant information. For example, all cells in the SE quadrant in Figure 0.2(d) are in the fairway. Therefore, there is no need to further subdivide this quadrant. Hence, the SE quadrant in Figure 0.2(e) is not subdivided any further and the entire quadrant is marked as 4 which represents the land cover type fairway. The other three quadrants in Figure 0.2(e) are further subdivided into four smaller quadrants as each of these areas do not have a uniform land cover type. The NW and SE sub-quadrants of the NW quadrant do not need to be divided further. Similarly, the NE and SW sub-quadrants of the NE and SW quadrants need no further subdivision. The quad-tree structure for Figure 0.2(e) is shown in Figure 0.2(f). Thus, the raster structure in Figure 0.2(e) stores the same amount of information as that in Figure 0.2(d). However, the number of cells in Figure 0.2(e) is 31, as opposed to 64 in Figure 0.2(d), which is about a 50% reduction in terms of storage requirement.

0.3.2 Manipulating field data

We divide the operations on field data into two parts: partitioning the space, and analyzing field data. Each of these topics are discussed in detail in the sections below.

Partitioning the space:

A number of applications in GIS require the partitioning of space using tessellations. Examples of such applications include interpolation of data from collected sample points for digital terrain modeling, determining nearest neighbors, and performing locational optimization such as selecting sites for business applications and optimizing transportation networks. The solution to all these problems stems from the fundamental question that given a set of point locations (sites) in a continuous space, how to associate every other point in that space to its nearest site (or sites, if more than one exist) [62]. Such an association results in an exhaustive partitioning of the space into a network called the *Voronoi diagram*. The region of space associated with each site is called *Voronoi polygon*, with its constituent vertices and edges being *Voronoi vertices* and *Voronoi edges* respectively.

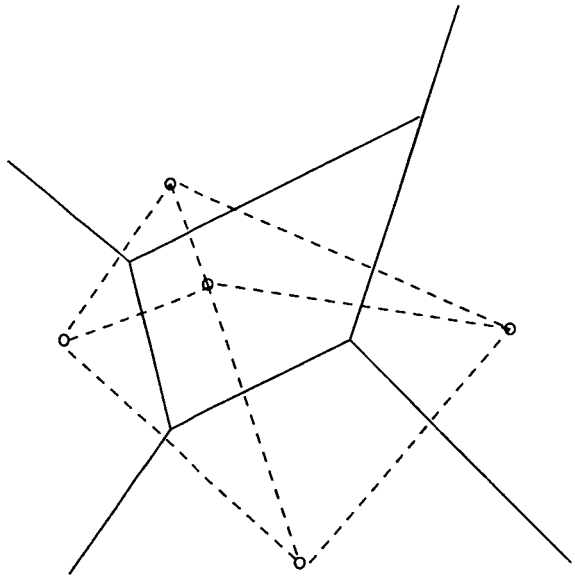


Figure 0.3 Illustration of the Voronoi diagram and Delaunay triangulation

The dual of an n dimensional Voronoi diagram, called the *Delaunay tessellation*, can be constructed by joining the points from its $n-1$ dimensional faces. In the two-dimensional Euclidean plane, the straight line dual of the Voronoi diagram results into a triangulation (called the Delaunay triangulation) of the convex boundary of the sites. Figure 0.3 is an illustration of the Voronoi diagram and Delaunay triangulation for the given set of points shown as empty circles. In Figure 0.3, the Voronoi diagram is shown with solid lines and the Delaunay triangulation is shown with dashed lines.

We discuss below algorithms for developing Voronoi diagrams for a two dimensional Euclidean plane.

1. **Incremental method:** This method starts with the Voronoi diagram $V(P_n)$ for a given set P_n of n points and develops new edges as new points are added to it. The basic idea is to construct the Voronoi polygon for a new point p_{n+1} that is added to the set of points (P). The steps are detailed below:

   (a) The first step is to identify the point p_i whose Voronoi polygon contains the point p_{n+1}.

   (b) Next, a perpendicular bisector to the segment $p_{n+1}p_i$ is drawn which intersects the $V(P_n)$ at q_i and q_{i+1} respectively.

   (c) Assuming that q_iq_{i+1} is the oriented counterclockwise, it enters the adjacent Voronoi polygon at point q_{i+1}. A perpendicular bisector to the segment $p_{n+1}p_{i+1}$ is drawn.

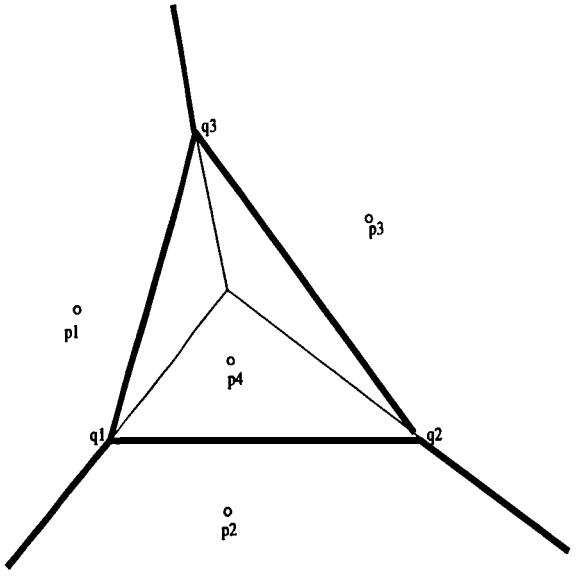


Figure 0.4 Illustration of the incremental method

(d) Steps 1b and 1c are repeated until the point q_i is revisited.

(e) At this point, the Voronoi polygon for the new point p_{n+1} is obtained.

(f) The new Voronoi diagram $V(P_{n+1})$ is obtained by removing the substructure inside the Voronoi polygon for p_{n+1}.

The method is illustrated in Figure 0.4, where the initial set of points contained p_1, p_2, and p_3. As point p_4 is added to the set, p_2 is chosen as the starting point since the Voronoi polygon of p_2 contained p_4, and is therefore the closest to p_4. Next, the perpendicular bisector p_4p_2 is drawn which intersects the Voronoi diagram at points q_1 and q_2 respectively. Similarly, perpendicular bisectors of p_4p_3 and p_4p_1 are drawn which intersect the Voronoi polygon of p_3 at q_2 and q_3, and that of p_1 at q_3 and q_1 respectively. The new Voronoi diagram is obtained by removing the substructure, which is shown in ordinary lines. The new Voronoi diagram is shown in bold lines.

Several important factors must be considered in implementing this algorithm. Firstly, the point p_{n+1} must not lie on the convex hull of P_n, which will result in an infinite Voronoi polygon for p_{n+1}.. Secondly, the point p_i should be the nearest neighbor to p_{n+1}. Thirdly, the size of the substructure should be kept small. The second and third factors will adversely affect the execution time. All of the above cases are discussed in detail in [62]. This algorithm has average and worst case time complexities of $O(n)$ and $O(n^2)$ respectively.

2. **Divide and conquer:** This method assumes that no two points in the

given set of points P_n have the same y coordinate. The steps in the algorithm are as follows:

(a) If the number of points in P_n is two, the Voronoi diagram is simply their perpendicular bisector. If the number of points is three, the Voronoi diagram consists of the perpendicular bisectors for each pair of points (p_1p_2, p_2p_3, p_3p_1) which meet the center of the circumcircle the traingle whose vetrices are at p_1, p_2, and p_3.

(b) If the number of points in P_n is greater than three, P_n is divided into two equal sets P_{n_1} and P_{n_2} such that the x coordinates of all points in P_{n_1} are less than that of P_{n_2} each set P_{n_1} and P_{n_2}.

(c) Steps 2a and 2b are repeated for each of the sets P_{n_1} and P_{n_2}.

(d) The Voronoi diagram for P_{n_1} and P_{n_2} are merged to obtain the Voronoi diagram for P_n.

The above algorithm has the worst case time complexity of $O(nlogn)$.

Since the Delaunay triangulation is the dual of the corresponding Voronoi diagram, it can be obtained by joining the vertices whose Voronoi polygons share a common edge. Data structures such as the winged-edge and quad-edge structures (described later in this chapter) can be used for obtaining the Delaunay triangulation for a given Voronoi diagram.

Analysis of field data:

1. **Spatial filtering:** Spatial filtering refers to the process of identifying patterns in attribute data over a geographical area. Two types of patterns are of interest in this context. The first deals with general trends in the spatial orientation of attribute data. Examples in this category include average rainfall, average household income, and total number of spieces in a geographical area. The other type of spatial pattern include detection of edges in the spatial distribution of different attribute values. Thus, while the former deals with smoothing the data by eliminating noise in the forms of individual variability in data that do not correspond to general trends, the latter highlights individual variabilities in order to bring out the differences.

   As an example of smoothing, consider a raster data set consisting of $N \times N$ cells. This data set can be aggregated into $k \times k$ groups where $k = N/n$. Then each of the k groups will consist of an $n \times n$ grid. The attribute value of each k group can then be equal to the mean of the attribute values of the $n \times n$ cells. Such a process gives rise to a new data layer that represents the general trend or pattern of data by smoothing out individual variations. This process can be repeated for different window sizes to identify the general trends at varied degrees of resolution.

In direct contrast to smoothing is the process of *high pass filter* that enhances the contrast in the data value with respect to those in the neighborhood. In high pass filter in an $n \times n$ grid, the data value is calculated by the following formula: $n^2 \times v_c - \Sigma_{i=1}^{n^2} v_i$, where v_c is the value at the centroid of the region, and v_i represents the data value at cell i, calculating from the top left corner in a row or column major system. In this process, the value at the centroid is inflated if it is substantially different from those of its neighbors, and deflated otherwise. Thus, areas where significant variability of data values exist are sharply contrasted with those where such variations do not exist.

2. **Spatial transformation:** Spatial transformation deals with transforming data according to a specified spatial orientation such as direction. Examples of spatial transformation include *texture transformation* which is similar to the high pass filter except that delimited regions are demarcated with different textures.

   Another transformation commonly used is *slope analysis*, in which the slope of a terrain is determined by first calculating the local slope in an $n \times n$ portion of elevation data in the raster format and then extending the analysis to larger areas by using the local means of each $n \times n$ regions.

   Related to slope analysis is *aspect analysis*, which refers to the process of determining the *direction*, rather than the value of the slope. Aspect is the horizontal component of the vector that is perpendicular to the surface. Since aspect data is categorized into a set of predetermined directions, such data is in the ordinal scale.

   Slope analysis is often done along a specified path. In raster data, elevation data for the neighborhood cells in the specified direction are used to calculate the slope. Often calculating the slope requires interpolation of the elevation data in the adjacent cells to obtain the slope along along the specified path. For vector data such calculations are done by interpolating elevation data from contour lines.

0.4 Object-based representation and manipulation

In object-based representation methods, a finite number of spatially referenced objects are plotted on an embedding space. The objects are described in terms of their static, behavioral, and structural properties [82]. The static properties describe attributes such as name and population (of a city) related to the spatial object. Behavioral properties refer to methods for manipulating the object, such as plotting a representation (a point) of the object (a city) at a certain scale (a country map). Structural properties refer to the spatial properties such as location and extent of the object with respect to the embedding space. It is to the structural properties that most of the discussion in this chapter is devoted.

Polygon	Line segment
1	AB, BC, CD, DA
2	CD, CE, EF, FG, GD
3	FG, FI, IJ, JH, HG
4	JK, KA, DA, GD, HG, JH

Table 0.1 Example of spaghetti structure

Geographic objects are represented by their geometric counterparts. For example, points can be used to represent oil wells, lines to represent roads, rivers, and boundaries, polygons to represent enclosed areas such as land parcels, and blocks to represent three dimensional objects such as buildings and aquifers. Thus, point, line, polygon, and block are the four fundamental geometric units that are used to represent geographic objects [54]. Examples of static properties are spot heights for point objects, names for rivers and roads represented by line objects, county names and elevation layers represented by areal objects, and subsurface depths for aquifers represented by block objects. These geometric units can be combined to form larger objects. If the same type of simple geometric units are used to form a larger object, then it is called a *compound object.* Examples of compound objects are dyads, networks, and lattices. *Complex objects*, on the other hand, are those made up of simple units of different types. An example of such an object is a county consisting of land parcels (polygons), roads (lines), and utility networks.

0.4.1 Representation of spatial objects

Several methods have been suggested in the literature (e.g., [54, 61, 66, 67, 71, 82]) including *spaghetti*, *arc node structure*, *doubly connected edge list*, *winged-edge*, and *quad-edge* representations. We discuss each of these briefly below:

1. **Spaghetti:** The spaghetti structures are used for two dimensional representations of spatial objects including polylines (collection of straight line segments), mixtilines (collection of straight lines and arcs), polygons, collection of polygons, and polyhedra. Polylines are divided into the constituent straight line segments, arcs are discretized into the constituent straight line segments, polygons are represented by arcs constituting their boundaries, and polyhedra are decomposed into their facets, which are represented as polygons.

   The spaghetti representation of the golf course map of Figure 0.2(a) is shown in Table 0.1. The map is divided into polygons and line segments. The polygons are numbered from 1-4, according to the land cover they represent, and corresponding to each polygon the arcs that mark its boundaries are recorded. Attribute data such as "perimeter", "area", and "type", in addition to locational data, can be associated with spaghetti structures. Representations using spaghetti structures lose topological relationships

Arc	Start node	End node	Left area	Right area
AB	A	B	1	X
BC	B	C	1	X
CD	C	D	1	2
DA	D	A	1	4
GD	G	D	2	4

Table 0.2 The arc node structure

such as distance because the spatial data is broken down into the simple constituent parts [48, 82].

2. **Arc node structure:** The arc node structure [67] is an extension of the spaghetti structure that uses arcs and nodes to add the adjacency relationship of contiguous polygons. The arc node structure stores line segments as directed arcs, the start and end points of each arc as nodes, and the adjacent areas to the left and/or right of each arc. The arcs can only intersect at their end points and have an area to their right and left hand sides. Thus outside arcs (as opposed to those representing interior boundaries) have an external region associated with them. Table 0.2 represents an arc node structure of an area. The area external to the the golf course is demarcated by the letter "X".

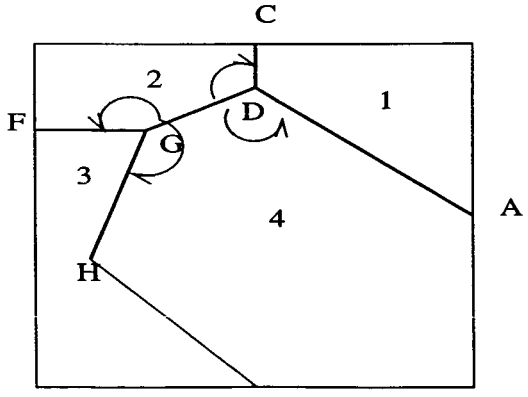
3. **Doubly connected edge list:** The doubly connected edge list (DCEL) [61, 66, 82] adds information about the planar orientation of the arcs around the vertices that constitute their end points and the areas that are situated to their left and right hand sides. An example of a DCEL is shown in Table 0.3 that associates each arc with its start and end points, faces (areas) to its left and right, and two pointers pointing to the previous and following edges respectively.

4. **Winged-edge representation:** The winged-edge representation [3, 79], used extensively in computer graphics and geometric modeling applications, adds further topological information to the DCEL structure. It associates with each edge the start and end vertex points, pointers to the left and right faces adjacent to the edge, and pointers to the edges that are incident to the two vertices in clockwise and counterclockwise manners. This representation is particularly useful in representing the boundaries of polyhedra. An example of the winged-edge representation is shown in Figure 0.5.

5. **Quad-edge representation:** The quad-edge representation [38] is the most general in its ability to model surfaces with or without boundaries and orientation. As the name suggests, four pointers are associated with each edge: two pointing to its start and end vertices, and two pointing to its adjacent faces. The quad-edge representation automatically encodes the dual of the surface that is being represented. Hence obtaining the dual of

Arc	Start node	End node	Left area	Right area	Prev. arc	Following arc
AB	A	B	1	X	KA	BC
BC	B	C	1	X	AB	CD
CD	C	D	1	2	BC	DA
DA	D	A	1	4	CD	AB
DG	D	G	4	2	AD	GH
GH	G	H	4	3	DG	HJ
HJ	H	J	4	3	GH	JK
JK	J	K	4	X	HJ	KA
KA	K	A	4	X	JK	AD
AD	A	D	4	1	AK	DG
CE	C	E	2	X	DC	EF
EF	E	F	2	X	CE	FG
FG	F	G	2	3	EF	GD
GD	G	D	2	4	FG	DC
DC	D	C	2	1	GD	CE
FI	F	I	3	X	GF	IJ
IJ	I	J	3	X	FI	JH
JH	J	H	3	4	IJ	HG
HG	H	G	3	4	JH	GF
GF	G	F	3	2	HG	FI

Table 0.3 The DCEL structure



Start Node: G
End Node: D
Left face: 2
Right face: 4
Edge incident clockwise at G: HG
Edge incident counterclockwise at G: FG
Edge incident clockwise at D: DC
Edge incident counterclockwise at D: DA

Figure 0.5 The winged-edge structure for arc GD

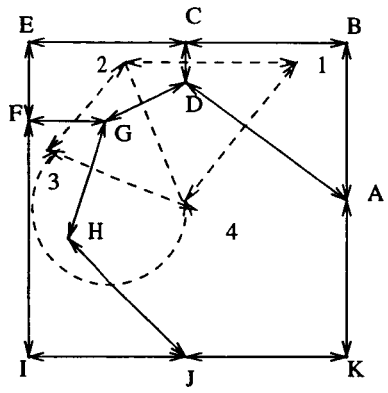


Figure 0.6 The quad-edge structure

a surface from its quad-edge representation requires no extra computation. The map in Figure 0.2, its dual, and quad-edge representation are shown in Figure 0.6, where the dual is shown by dashed lines. For simplicity of the figure, we have not shown the duals corresponding to the boundary segments AB, BC, CE, EF, FI, IJ, JK, and KA.

0.4.2 Manipulating spatial objects

The operations on spatial data has been categorized into *static* and *dynamic* operations with the difference that static operations do not change the objects given as the operands, while dynamic operations change the operand objects [82].

Static operations have been further classified into *general*, *set-oriented*, *topological*, and *Euclidean* [82]. The only general operation is *equal* that evaluates the two spatial operands and returns a boolean value depending on whether the two operands are the same object.

Dynamic operations change the operands as a result of the operation. Examples of dynamic operation include *generation* of new objects, and transforming existing objects by operations such as *split*, *merge*, *scale*, and *rotate*.

We discuss below some of the important algorithms used for manipulating spatial objects discretizing, topological, set-oriented, and Euclidean algorithms.

Discretizing algorithms:

Due to increased efficiency, spatial objects having geographic coordinate system are manipulated using their geometric representations in an Euclidean plane [82]. The planar references can then be mapped to the terrestrial references and *vice versa*. Discretizing algorithms are used to represent geographic objects in a discrete Euclidean plane, where every point in the plane has integer coordinates. Discretization, however, may introduce errors. For example, every point in a

line segment which is not parallel to either axis cannot have integer coordinates. Thus, the point of intersection of two such line segments will have to be approximated with a point having integer coordinates. The Greene-Yao algorithm is used to establish an upper bound on such errors [37]. Other discretization algorithms include approximation of arcs by polylines used in cartographic generalization [82].

Topological algorithms:

Topological operations include *boundary*, *interior*, *overlap*, *cover*, and *connectedness*. These operations are homeomorphic in that their results are preserved after topological transformations of the plane. A convex polygon is one that has no vertex with an interior angle less than or equal to π.

1. **Boundary of convex polygons:** The problem dealt with here is to obtain the boundary of a convex polygon from a given set (N with cardinality n) of points that are either interior to it or lie on its boundary.

   One approach in solving this problem is to determine the *extreme points*, which are vertices (i.e., interior angles are less than π). The extreme points can be identified by isolating them from the *interior points*. A point p is an interior point if and only if it lies inside a closed triangle (which includes the sides) whose vertices are points in the given set N. All interior points can thus be obtained by testing them against sets of triples from the set N. This algorithm is very inefficient having $O(n^4)$ complexity.

   A second approach is to determine the *extreme edges* of the polygon. Since every interior point will lie on or to the left of an extreme edge oriented counterclockwise, checking every point p_i against all edges p_jp_k such that $i \neq j \neq k$ and $i, j, k \in N$, will yield all such edges. This algorithm has a complexity of $O(n^3)$.

   Another solution to this problem is the *gift wrapping* algorithm [17] that improves the complexity to $O(n^2)$ by eliminating the need to check each point against all edges. Instead, it starts with an extreme point p_i (identified by its extreme x or y coordinate) and identifies another extreme point p_j ($i \neq j$) such that the segment p_ip_j creates the smallest counterclockwise angle θ with the vertical axis (if p_i has an extreme x coordinate) or horizontal axis (if p_i has an extreme y coordinate). The next step is to find another point p_k ($k \neq j$) such that the edge p_jp_k makes the smallest counterclockwise angle with p_ip_j. The algorithm stops when p_i is revisited.

   A more efficient algorithm with complexity $O(nlogn)$, suggested in [36], starts with the extreme point p_0 that has the smallest y and largest x coordinates and sorts all other points $(p_1, \ldots, p_{n-1})$ by the angles created by $p_0p_1, p_0p_2, \ldots, p_0p_{n-1}$ with the horizontal axis. It then creates a stack with p_0 and p_1 pushed into it. If $p_0p_1p_2$ creates an internal angle less than π, then p_2 is popped because it is not an extreme point. In that case, the points $p_op_1p_3$ are checked. Otherwise the triple $p_1p_2p_3$ are checked. The

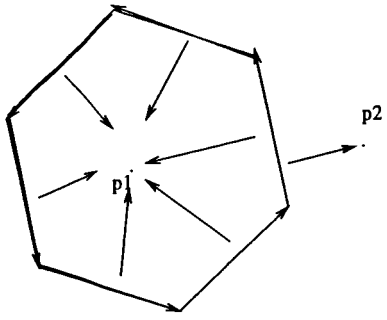


Figure 0.7 Point in convex polygons

boundary is obtained by repeatedly checking this condition for all triples of consecutive points and pushing the appropriate points into the stack. The algorithm stops when p_0 is revisited.

2. **Point in polygon:** The purpose of this algorithm is to determine whether a given point lies inside a given polygon. There are numerous applications of this algorithm including LIS, car navigation systems, and business applications. Different algorithms are required for *convex* and *nonconvex* polygons. For a convex polygon P with n vertices, the point p is inside P if and only if p is on the left hand side of all directed edges oriented counterclockwise [64, 82]. This is illustrated in Figure 0.7, where points p_1 and p_2 are inside and outside of polygon P, whose boundary is shown with directed arcs in the figure. The arrows, emanating from the arcs that form the boundary of P and are directed towards p_1 and p_2, are intended to show that p_1 is always on the left-hand side of the arcs forming the boundary of the polygon. Point p_2, on the other hand, is clearly on the right-hand side of the arc shown on the boundary of the polygon P. Thus, p_1 and p_2 are on the inside and outside of the polygon respectively.

   We describe below two popular algorithms for solving the point in polygon problem for nonconvex polygons. These include *ray crossing* (also known as the semi-line algorithm) and *winding numbers*.

   Ray crossing: In order to determine if a point p is inside a polygon P, the ray crossing algorithm requires a ray extended from p to infinity in an arbitrary direction. If the ray intersects the boundary of P an odd number of times, point p is *inside* polygon P, otherwise it is *outside* polygon P. Several situations are treated as special cases in this algorithm. This includes the following:

   (a) The ray crosses P at a vertex.

   (b) The ray is collinear with an edge of P.

(c) The point p lies on the boundary of P.

All of the above cases are handled by counting a crossing as an intersection if and only if one endpoint of the edge with which the ray crosses is strictly above the intersection and the other end point is on or below the point of intersection. Some limitations of the above algorithm are mentioned in [64] but it is generally acceptable for most applications.

Winding numbers: The method of winding number calculates the total signed angular turn (w) made by an individual located at point p and always facing toward another point traversing counterclockwise and making a complete turn along the boundary of polygon P. The winding number is calculated by $w \div 2\pi$. If the winding number is 1, which means that a complete rotation has been made, then the point p is inside polygon P. In any other case the point p is outside polygon P.

Set-oriented algorithms:

Set-oriented operations include standard set theoretic operations such as *equality, membership, subset, disjoint, intersection,* and *union.*

1. **Segment intersection:** The segment intersection problem deals with determining if two line segments intersect and if so, the point of their intersection. An obvious solution to this problem is to solve the two slope-intercept equations for the segments simultaneously. A drawback to this approach involves special cases such as vertical segments. A more general approach for solving this problem is to use a parametric representation of the two segments as vectors. Thus, a line segment ab can be represented as a vector $B - A$, where $A(x_A, y_A)$ and $B(x_B, y_B)$ are the two end points. Any point on AB can then be represented as $A + \alpha(B - A)$, where $\alpha \in [0, 1]$. Thus, when $\alpha = 0$ we get the point A, and with $\alpha = 1$ we get the point B. Similarly, any point on a second line segment CD with two end points $C(x_C, y_C)$ and $D(x_D, y_D)$ will have the parametric representation $C + \beta(D - C)$. The point of intersection of AB and CD can then be represented as the vector equation $A + \alpha(B - A) = C + \beta(D - C)$, which is equivalent to two equations as follows:

$$x_A + \alpha(x_B - x_A) = x_C + \beta(x_D - x_C) \tag{0.1}$$

$$y_A + \alpha(y_B - y_A) = y_C + \beta(y_D - y_C) \tag{0.2}$$

The solution to simultaneous equations 0.1 and 0.2 yields the following results:

$$\alpha = \frac{x_A(y_D - y_C) + x_C(y_A - y_D) + x_D(y_C - y_A)}{x_A(y_D - y_C) + x_B(y_C - Y_D) + x_C(y_A - y_B) + x_D(y_B - y_A)} \tag{0.3}$$

$$\beta = -\frac{x_A(y_C - y_B) + x_B(y_A - y_C) + x_C(y_B - y_A)}{x_A(y_D - y_C) + x_B(y_C - Y_D) + x_C(y_A - y_B) + x_D(y_B - y_A)} \tag{0.4}$$

2. **Polygon intersection:** The polygon intersection problem deals with finding the intersection of two polygons P and Q having m and n edges respectively. Applications of the polygon intersection problem can be found in many GIS domains such as map overlaying, utilities management, planning and facilities management as well as non GIS domains such as VLSI. Several linear time ($O(m+n)$) algorithms for the intersection of convex polygons have been developed. It has been shown (e.g., [66]) that the intersection produces a convex polygon with a maximum of $m+n$ edges. Here we describe one of the most elegant algorithms described in [65, 64, 66]. The algorithm works by having two directed (counterclockwise) edges $p_{i-1}p_i$ and $q_{j-1}q_j$ of the two polygons P and Q "chase" each other along the boundaries such that they always meet at the intersections. The intersection points form the vertices of the polygon which is the result of the intersection of P and Q. In order to ensure that the two chasing edges always meet at the intersections of P and Q, the following rules are applied:

   (a) If $p_{i-1}p_i \times q_{j-1}q_j > 0$ and $q_j \in L(p_{i-1}p_i)$ then increment i.
   (b) If $p_{i-1}p_i \times q_{j-1}q_j > 0$ and $q_j \notin L(p_{i-1}p_i)$ then increment j.
   (c) If $p_{i-1}p_i \times q_{j-1}q_j < 0$ and $p_i \in L(q_{j-1}q_j)$ then increment j.
   (d) If $p_{i-1}p_i \times q_{j-1}q_j < 0$ and $p_i \notin L(q_{j-1}q_j)$ then increment i.

   In the above rules, $\times$ refers to the cross product of the two vectors given as operands, and L refers to the closed half plane to the left of the vector given as its parameter. There are three special cases that arise if no intersection points are found between P and Q. These are shown below:

   (a) If $p_i \in Q$ then $P \subseteq Q$.
   (b) If $q_j \in P$ then $Q \subseteq P$.
   (c) Otherwise $P \cap Q = \emptyset$.

   An example of the application of the above algorithm is shown in Figure 0.8.

Euclidean algorithms

Euclidean operations include *distance*, *length*, *area*, *perimeter*, and *centroid* that take spatial objects such as points, lines, and areas as input and return a real number as a result of the operation.

The area of a convex polygon with vertices $p_0, p_1, \ldots, p_{n-1}$ can be calculated by the sum of the triangles created by $p_0p_1p_2, p_0, p_2, p_3, \ldots, p_0p_{n-2}p_{n-1}$. Thus, the area of the polygon $p_0p_1p_2p_3p_4p_5$, shown in Figure 0.9, is equal to the sum of the areas of triangles $p_0p_1p_2$, $p_0p_2p_3$, $p_0p_3p_4$, and $p_0p_4p_5$.

The centroid of a polygon P with vertices $p_0, \ldots, p_{n-1}$ is the mean $(p_0p_1 + p_1p_2 + \ldots + p_{n-1}p_0)/n$. Other measurements such as length, and perimeter can be computed using simple concepts from coordinate geometry.

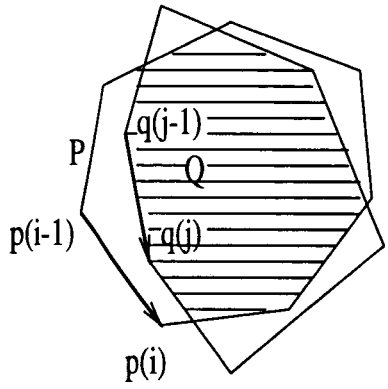


Figure 0.8 Intersection of convex polygons

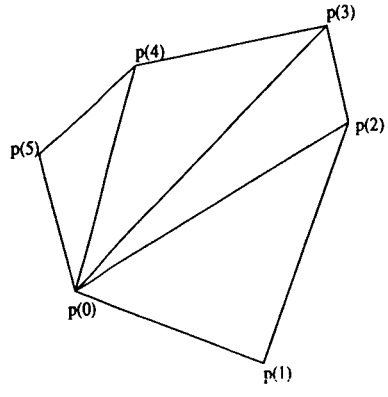


Figure 0.9 Area of convex polygons

Another operation in this context is proximity analysis, that deals with the determination of the distance of an object from another object, as well as identification of the existence of objects within a specified boundary. If the objects under question are points, then the distance measure is simply the length of an straight line joining them, unless there are restrictions on the traversal path from the source to the destination. An example of such a restriction is the layout of the road network between two locations in a city, in which case the distance may not be the straight line distance between them. The distance measure becomes even more complicated if it is measured in terms of the shortest traversal time in a rush hour with varying congestion patterns along different routes.

The distance measure between two polygons, unlike points, can have multiple interpretations. It could, for example, be measured by the distance between their centroids or that between their edges. The former could be relevant for determining the average distance between the downtown areas of two cities while the latter could be required to measure the distance between an irrigation canal and the edge of a field [71].

Searching for objects in the vicinity of another has different implications based on the shape of the source object in question. Thus, for a point source, the vicinity could be specified by the circle of a given radius, where the point is at the center of the circle. For a line segment, the vicinity could be specified by a region enclosed by two lines running at a specified distance, in parallel and on opposite sides of the source line segment.

0.5 Data Conversion

Data conversion refers to the process of converting data from one format to another. Such conversion is necessary due to the suitability of different data formats for different purposes. For example, the input to a GIS may come from digitized data in the raster format and the data format used in the GIS may be in vector format. Moreover, there are different types of raster and vector formats for data structures, which makes it necessary to have intra format data conversion procedures. Thus, there are four sets of conversion methods: raster to raster, raster to vector, vector to raster, and vector to vector.

0.5.1 Intra-Format conversion

In this section we describe raster to raster and vector to vector data conversion methods. Raster formats differ in the way they are stored. Three commonly used storage strategies for data in raster formats include *band sequential* (BSQ), *band interleaved by pixel* (BIP), and *band interleaved by line* (BIL). The purpose of all these strategies is to efficiently store raster structure data for different thematic data layers. For example, a given geographic study area may have three thematic data layers: land elevation, rainfall, and slope. Each of these layers will be stored in a separate raster structure with the same resolution. These individual raster structures can be stored in different files, which is called

the BSQ strategy, or the pixels (raster cells) can be stored sequentially with all data values stored after each pixel, which is the BIP strategy, or the pixels stored in separate lines with data values interleaved between them, which is the referred to as the BIP strategy. Conversion between raster formats requires reorganization of the raster cells and their corresponding data values, which is a relatively simple operation as compared to inter format conversion.

Vector to vector conversion is necessitated due to the usage of different types of vector formats. Data structures in the vector format can be classified into two primary categories. In the first category is the *whole polygon* method in which the polygons are stored in terms of the coordinates of the vertices. In this method, the nodes and the arcs are implicitly representation. In contrast, other methods, including DIME (Dual Independent map Encoding) , arc-node, and relational structures, are variations of the arc node structure, where nodes, arcs, and polygons are stored in tabular formats.

0.5.2 Inter-Format Conversion Methods

Inter-format data conversion consists of two types: raster to vector and vector to raster. In raster to vector conversion, the input data is in raster format and the output is in the vector format.

Raster to vector conversion: Converting a raster structure to the corresponding vector structure must start with some assumptions about the raster and vector representation of data. In a binary image raster format, if a point occurs inside a cell, the cell value is"1", otherwise it is "0", which implies that the position of the point inside the cell is lost. Similarly, if a line passes through a cell, the cell value is "1", otherwise it is "0". Here the assumption is that the line always passes through the center of the cell. A second assumption that needs to be made about a raster representation is about cell connectivity. There are two possible ways to connect adjoining pixels: only orthogonally, and orthogonally and diagonally, referred to as "4-connected" and "8-connected" respectively. In the 4-connected neighborhood approach, pixels that are located in one of the four positions with respect to a given pixel are considered adjoining: directly above, directly below, to the left, and to the right. In the 8-connected approach, adjoining cells that are diagonally located are also included in determining adjacency. For vector data structure that will be derived from a raster structure, the assumptions are related to the possible orientations of the vector and its thickness. Thus, an orthogonal vector is oriented vertically or horizontally to the axes of the raster structure, whereas a nonorthogonal vector may be oriented at an arbitrary angle to the raster axes. With respect to thickness, a vector could be assumed to have a unit width (as in the case of polygons) or variable, non-unit thickness (as in the cases of images and pictures).

All of the methods we discuss here assume binary raster structure as input data. An orthogonal vector (a vector parallel to either the x or the y axis)

with unit width can be converted from its corresponding raster structure using the 4-connected approach, simply by joining adjacent pixels by unit vectors. Polygons with such vectors can similarly be retrieved with no distortion. To extract a nonorthogonal vector with unit width from the corresponding raster structure, the 4-connected approach results in distortion, because such a vector can only be represented by horizontal and vertical line segments, which lie in the general direction of the vector as fragmented line segments. Consequently, a polygon with several vectors result in considerably amount of distortion.

An 8-connected approach to a nonorthogonal unit vector results in a less distorted image, but may introduce additional lines that were not present in the original vector. For example, portions of the vector that pass off the center of a pixel may be represented as steps. Nonorthogonal polygons may have additional lines at vertices, because pixels near a vertex will be connected by diagonal as well as horizontal lines.

Vectors with unit width are uncommon except in geometric shapes. Examples of vectors with variable thickness are lines in images. To convert a raster representation of a line with variable thickness into a corresponding vector representation, a method called ***skeletonizing*** is used. The purpose of skeletonizing is to extract the skeletal raster image of the vector. A skeletal raster image can be obtained by successive *peeling* of the raster image where each pass produces a thinner image. The process stops when the raster image of a unit width vector is left. Another method for skeletonizing a polygonal raster image with variable thickness is to increase the gaps between vectors in the polygon. A third approach determines the center of the vectors, and retains the pixels that are farthest from the outside edges of the image.

Vector to raster conversion: Converting a vector to a raster structure involves overlaying the vector on a raster array, and identifying the pixels through which the vector passes. Pixels through which the vector passes can then be darkened to produce the raster image. This approach, however, often produces a stair-step distortion for vectors that are at an angle to the vertical and horizontal axes of the raster array. This distortion, called ***aliasing*** occurs due to the fact that such a vector can pass through two horizontally or vertically adjacent pixels. The distortion due to aliasing can be reduced by gray scaling the pixels according to some measure of coverage of the pixel by the vector. An obvious choice is the length of the vector that passes through the pixel measured as a fraction of the diagonal of the pixel. Polygonal structures represented in the vector format can be converted to the corresponding raster structure by converting the vectors of the polygon, and representing the interior by some pre determined value. Point data from vector format are converted simply by assigning the value of a feature at that point (such as land elevation) to the pixel whose center is nearest to the point in the vector format. Since raster cells cover

a definite area, the implicit assumption is that the entire area will have the same feature value, which is not always realistic. A further limitation occurs when the pixel can be assigned more than one value if more than one point are equidistant from the pixel center.

0.6 Spatial Database Issues

The choice of the data format used for storage of spatial data can have an impact on the indexing method to be chosen for efficient retrieval. At the same time, different data structures facilitate different types of operations, hence affecting modeling and querying capabilities. For example, raster structures are more conducive to area-oriented operations, while vector structures are more boundary-oriented. Therefore, from a spatial database standpoint, it is important to determine the suitability of the data structure for the specific application at hand, and if possible, develop methods for supporting both types of data structures. We thus propose the following research issues in the context of spatial databases.

- **Database architecture:** As discussed in the previous sections, numerous methods have been proposed for representing and manipulating spatial data, which are not needed in traditional database systems that are designed for dealing with only alphanumeric data. One obvious solution is to separate the spatial data from non-spatial data, which are manipulated separately a geometric and a traditional database engine respectively. A fundamental problem with this approach is maintaining data integrity: if a change is made in the spatial data, how would that change be reflected in its non-spatial counterpart, and vice versa. Integrating spatial and non-spatial data introduces questions such as appropriate modeling techniques that are suitable for both type of data. We explore these issues further in the next chapter.

- **Classification of application areas by suitability of data structure:** A spatial database may offer different data structures to the application designer who can choose the appropriate data structure according to its applicability to the specific domain. The underlying assumption is that different application areas will be predisposed to certain types of data structure. In support of this view, we refer to the discussion in Chapter II, systems for car navigation, market analysis, and utilities mostly require boundary oriented data and functions, whereas DTM and cartographic modeling, land information systems, and soil surveys require more area-oriented data.

- **Hybrid structure:** Most applications, however, require both boundary-oriented as well as area-oriented data, although the requirement of one type of data might be more prevalent than that of the other. For example, map display, which is a boundary-oriented function, is common to all GIS

applications including those which predominantly require area-oriented operations. Hence spatial databases should enable an application to support the storage of both boundary and area-oriented data using different data structures. Since the boundary and area-oriented data are interrelated, such a separation would also require appropriate linkage between the data in order to ensure database integrity for update transactions.

Conversion between vector and raster formats is another issue. Although such conversion is possible, as discussed in a later chapter, real-time data conversion while processing a query is not feasible due to its adverse effect on response time.

- **Object orientation:** Since the boundary and area-oriented data are often related to the same underlying object (such as a land parcel, land cover, or utility network), the object oriented paradigm can provide a suitable modeling framework, where the boundary and area oriented data can be manipulated using suitable methods encapsulated in the object definition. The OO model also enables reusing methods and data through inheritance, and representing geographic containment relationship among objects at different resolutions.

Part IV

Data Modeling

The purpose of this chapter is to present a critical review of existing data models for GIS applications and identify their strengths and weaknesses. We then suggest future research issues including ideas to extend these existing data models. In Section 0.7,we present a discussion of the data models that have been proposed for traditional database applications, including the relational, extended entity-relationship (EER), and object-oriented data model. discuss the data models that have been proposed In Sections 0.8, 0.9, and 0.10, we discuss efforts to extend the traditional data models in order to incorporate functionalities required in GIS applications. In Section 0.11, we discuss extensible data models that have been proposed for GIS applications. Section 0.12 includes discussion about other GIS data models that are not fully described by any of the previous categories. These models include the topological model, Realm, Open Geodata Interoperability Specification (OGIS), and Computational Modeling Syetem (CMS). Section 0.13 addresses future research issues on data modeling for GIS applications.

0.7 Data models for traditional applications

The design of databases for traditional business applications have been based on the top-down, three-schema architecture [33]. The top-level schema, often referred to as the conceptual data model, is a high-level data model developed by determining the database requirements in requirements analysis phase. The conceptual data model is independent of the database management system (DBMS) used for system implementation. The second-level schema, referred to as the logical data model, is a translation of the conceptual data model into a data structure that is consistent with the DBMS platform selected for implementation. At the lowest level, the internal details of file organization and access paths for data are designed for the DBMS specifically chosen for the implementation. The internal schema is often referred to as the physical data model.

0.7.1 Conceptual data model

One of the most widely used conceptual data model is the entity-relationship (ER) model [18]. The ER model is used to describe, using a high-level user perception of data, the application-specific database structures which are necessary for supporting data retrieval and update transactions. The basic components of the ER model are *entity*, *relationship*, and *attribute*. Entities are objects or concepts that are of interest to the enterprise, and have independent existence. In defining entities and relationships, the ER model differentiates between *type* and *instance* to represent a generic set and a specific member of the set respectively. Entities that existantially dependent upon other entites are called *weak entities*. Relationships represent associations among entities. The degree of a relationship is determined by the number of participating entities, and can range from unary (recursive), binary, to n-ary. The structural constraints in a relationship are divided into cardinality and participation constraints. The cardinality ratio

of an entity (in a relationship) determines its maximum (for maximum cardinality) and minimum (for minimum cardinality) number of possible occurances in the relationship. The participation constraint of an entity in a relationship determines whether the entity must participate (minimal cardinality greater then zero) of may or may not participate (minimal cardinality equal to zero) in it. The attribute of an entity (relationship) defines a property of that entity (relationship), and are assigned values from a domain associated with it. Attributes are further divided by their structure (simple versus composite), value (single versus multi-valued), and storage (stored versus derived). If an attribute can be used for unique identification of the entity or relationship instances, then it is called a *candidate key.* A *primary key* is selected from a group of candidate keys. Similarly, a composite key can be formed by a group of attributes.

The ER model suffers from a number of problems such as *connection traps* [23], and more significantly, lacking sufficient constructs to develop complex applications such as computer-aided design/manufacturing, and GIS. The initial ER model has been extended to the entended/enhanced ER model [23, 33] by incorporating additional modeling constructs such as superclass/subclass relationship among entities, specialization/generalization of entities, and attribute inhertance by subclass entities. Thus, a subclass entity type is a member of a superclass entity, can inherit some of its parent entity, and can have its own set of subclass entities. This gives rise to a *type hierarchy* among subclasses and superclasses, which can be viewed from top-down as a specialization hiererchy, and from bottom-up as a generalization hierarchy. Structural constraints such as *disjoint* and *participation* have been defined over the type hierarchy. If a type hierarchy is disjoint, then an entity can be the member of only one of the subclasses of a given superclass. The participation constraint of a type hierarchy is total, if all entities of a superclass must belong to one of its subclasses. We describe the efforts made for extending the ER model to GIS application is Section 0.8.

0.7.2 Logical data model

Several logical data models, including the *relational*, *hierarchical*, and *network* models, have been proposed and used in traditional database applications. Among these, the relational model has become the dominant model for database applications [23]. The only data structure used in the relational model is the *relation.* Conceptually, a relation is a table consiting of rows and columns, where the columns consist of the attributes and the rows represent the individual records (tuples). Mathematically a relation is a subset of the cartesian product $D_1 \times D_2 \times \ldots \times D_n$, where $D_i (i = 1 \ldots n)$ is the set of allowable values (domain) for attribute i.

In the relational data model, any group of one or more attributes whose values can uniquely identify every tuple in the relation, forms a *candidate key.* One such group is chosen as the *primary key.* A set {A} of one or more attributes in a relation S forms the *foreign key* to another relation T in which {A} is a candidate key. Several integrity constraints are applied in the construction of a relation, in

order to ensure the accuracy of the data stored. The *entity integrity* constraint requires that none of the atribute forming the primary key of a relation can have a *null* value. The *referential integrity* consitaint requires that every value of a foriegn key must match a value of the correspnding candidate key. Additionally, business rules can be enforced as integrity constraints.

The relational model also provides languages for data manipulation. The basic relatinal languages defined in [22] are *relational algebra* and *relational calculus*. Relational agebra is a pocedural language that provides several operators whose inputs and outputs are both relations. The five basic operations are *selection*, *projection*, *cartesian product*, *union*, and *difference*, which have been used to develop additional operators such as *join*, *intersection*, and *division*. Relational calculus is based on *first-order predicate calculus* and has taken the forms of *tuple-oriented* calculus and *domain-oriented* calculus. The tuple-oriented calculus uses tuple variables that range over relations, and processes a query by retrieving tuples that satisfy a given predicate. A predicate is a well-formed formula that specifies conditions using comparison operators and free (tuple) and bound variables as operands. Bound variables are defined using *existential/universal quantifiers*. The domain-oriented calculus differs from the tuple-oriented calculus by using the domains of the variables instead of tuples of relations in specifying the attribute values. Tuples are then retrieved from relations by matching the attribute values with those of the domain variables specified.

0.7.3 The Object-oriented Data Model

The object-oriented model introduced *abstraction* and *encapsulation* in data modeling.

In this section we provide a description of the object-oriented (OO) data model in terms of class and type definitions, structural semantics, object behavior, communication between objects, schemas, query language, and object-oriented calculus.

1. Types and class definitions: A fundamental element of the OO model is the object, which has a unique identity denoted by an object identifier or OID, and which consists of a pair (identifier, value). Objects are grouped in classes. All objects belonging to the same class have values of the same type. Types are defined over a given set of classes. Thus, for a set of class C, the family of types include (a) atomic (primitive) types with pairwise disjoint domains, integer, string, boolean, float, (b) the class names in C, (c) a set of types {t}, and (d) a tuple of types $[A_1 : t_1, \ldots, A_n : t_n]$, where $A_1, \ldots, A_n$ are attribute names. A special type *any* is defined such that it does not occur inside another type. The set of types over the class C together with the type any is called **types** (C). In addition, four sets are defined: *dom*, which is a disjoint union of the domains of the primitive types, *obj*, which is an infinite set $\{o_1, o_2, \ldots\}$ of object identifiers (OIDs),

class, which is the set of class names, and *att*, which is a set of attribute names.

2. Class hierarchy: is defined as a triple (set of class names: C, a mapping function from C to **types** (C): s , and a partial order on class names C: $\prec$).

3. Type hierarchy: is defined as the smallest partial order on **types**(C) such that:

   (a) If c is a subclass of c', then c is a subtype of c' : $c \prec c'$ $c \leq c'$.

   (b) A tuple type $[A_1 : t_1, \ldots, A_m : t_m]$ is a subtype of $[A_1 : t'_1, \ldots, A_n : t'_n]$, if $t_i \leq t'_i$, for each $i \in [1, n]$, and $n \leq m$.

   (c) If a type t is a subtype of t', then the set $\{t\}$ is a subtype of $\{t'\}$.

   (d) All types are subtypes of *any*.

4. Structural semantics: A function π maps a class c from the set C to an OID from a disjoint fininte set of OIDs. The disjoint extension of c, $\pi(c) = \cup\{\pi(c') \mid c' \in C$, and $c \prec c'\}$. In other words, if an object is a member of a class c', then it is also a member of a class c, any superclass of c'.

   (a) Domain inclusion semantics: $ti \in t'_i \rightarrow dom(t_i) \in dom(t'_i)$. In other words, the domain of a type includes that of its subtype(s).

   (b) Disjointness of the ISA hierarchy: Classes that do not have any common subclass in the ISA hierarchy have disjoint extensions. This implies that multiple inheritence is not allowed. While this simplifies associating objects to classes, it is limited in modeling power.

   Example:

   i. ACTOR_DIRECTOR $\leq$ DIRECTOR $\leq$ PERSON
   ii. ACTOR $\leq$ PERSON
   iii. ACTOR_DIRECTOR $\not\leq$ ACTOR

5. Method: defines object behavior. A method consists of a name, a signature, and an implementation. The name of a method is taken from an infinite set of method names. The signature of a method is defined for a class and it represents a mapping from the class type to another type.

   Example: get_name: Person $\rightarrow$ **string**

   Methods for a class may be inherited by all of its subclasses. If a subclass c' does not have an alternative definition for a method m that has been defined by one of its superclasses c, the definition and implementation of m will be identical for both c and c'. For example, the method

   get_name: Actor $\rightarrow$ string

   is identical in definition and implementation to that defined for Person, which is a superclass of Actor.

(a) Method resolution: refers to the determination of the correct method m for a class c.

(b) Dynamic (late or value-dependent) binding: refers to the selection of the method for an object with a given OID. The method selected corresponds to the most specific class to which the object belongs.

(c) Static (context-dependent) binding: refers to the selection of a method based on type. The OO language C++ uses static binding with the option to select dynamic binding using the keyword **virtual**.

6. Message: refers to the call to a method by a *receiver* object. The selection of the method depends on the receiver. The set of methods applicable to an object is called the interface of the object.

7. Encapsulation: refers to the fact that an object can be accessed only via its interface.

8. Well-formedness: A set of method signatures M is well-formed if it obeys the criteria of *unambiguity* and *covariance*. Unambiguity implies that if class c has superclasses c' and c'', both of which have a method definition m, then m must also be defined either in c or in another class which is a superclass of c and subclass of both c' and c''. Covariance requires that if a method is defined in a subclass as well as in its superclass, the argument and result types for the method in the subclass are refinements of those defined on the superclass.

9. Schemas and instances in the OO model:

   (a) Schema: defines data structure, classes and associated types, the ISA hierarchy, and method signatures. The data structure includes description of value names and their appropriate types.

   (b) Instance: assigns values of appropriate type to the names.

10. Query language for OODB: Queries are different from methods in that the scope of a query is the whole database as opposed to a method whose scopes is restricted to a specific class in which it is defined.

11. Object-oriented calculus: consists of *atomic variables* that range over the sort **dom**; *terms* that can be atomic elements, variables or expressions of the form $x.A$, where x is a tuple variable and A is an attribute of x; *positive literals* involving relation, equality, membership, and inclusion operations on terms; *formulas* defined from atomic formulas using quantifiers such as $\wedge$, $\vee$, $\forall$, and $\exists$; and *queries* which are expressions of the form $\{x \mid f\}$ where f is a formula with exactly one free variable x.

12. Equality of objects: objects can be tested for equality by comparing their OIDs. Alternatively, two objects can be compared for eqality by testing their values. Thus, two objects can be considered equal even if they have

different OIDs, provided they have identical values. This is known as *value equality*. Another form of equality, called *deep equality*, if the trees (expansions) obtained by recursively replacing each object with its value are equal.

13. A method name can be incorporated in a query without any consideration for its implementation. For example:

    $\{y \mid \exists(x \in Persons\_I\_like \wedge y = get\_name(x))\}$

14. Object creation: this is accomplished with the operator **new**, that takes a set of values as input and generates a new OID for each value in the set.

0.8 GIS extensions of the entity-relationship model

This approach (e.g., [54]) extends the entity-relationship (ER) model [18] to deal with spatial data, by representing spatial objects as *entities*, and capturing their structural and topological relationships with other spatial objects through *relationships*. The actual representation of a spatal object varies according to the underlying data representation methods used for it. In object-based representation, the spatial objects are represented in terms of the constituent point, line, and area objects. Examples of line-oriented objects include straight line segments, polylines that consist of multiple line segments, and mixtilines that include straight lines as well as curve segments such as portions of circle, parabola, and spline. Each spatial object, represented as an entity, is related to its constituent objects through structural relationships of varying cardinalities. For example, a mixtiline is a collection of component line and curve segments, each of which has a many-to-one relationship with the parent mixtiline. Each endpoint has a two-to-one relationship with the parent line segment. Each areal object has a one-to-one relationship to its boundary (which could be a mixtiline or a polyline). Each volume object is related to its faces (which are areal objects) through one-to-many relationships.

Similarly, fixed and variable resolution structures can be represented by the ER model. Thus, a spatial object, its constituent quadrants, vertexes, and edges are represented as entities. The constituent parts are associated with both the quadrants in which they lie and their parent objects through structural relationships. This method is extended to model terrains, simple and complex polyhedra, blocks, land parcels, and road networks.

0.9 GIS extensions of the relational data model

0.9.1 GRDM

The GeoRelational data model (GRDM), described in [47], extends the relational data model by incorporating modeling constructs required in GIS applications.

It defines five constructs: *layers*, *relations*, *virtual layers*, *object classes*, and *integrity constraints*. Layers and relations are used to define the logical schema, and virtual layers and object classes are used as user views. A *layer* corresponds to a thematic data layer that describes geographic properties by associating a two dimensional space with a set of attributes. In terms of relational database terminology, it is equivalent to a relatoin whose key is a set of geometric features such as points, lines, and regions. A *relation* is used for representing non-geographic entities. A *virtual layer* is used for representing information computable from existing layers. The computations are defined to support operations in thematic cartography, and include *attribute derivation*, *geometric computation*, *overlaying*, and *reclassification*. Attribute derivation allows generation of new attributes from existing ones, such as population density from population figures. Geometric computation refers to developing new features from existing ones, such as buffer zones of lines and regions. Overlaying refers to the development of a new layer by intersecting the geometric features of two input layers. Reclassification allows generating new layers by integrating adjacent features having same attribute values. Object classes are equivalent to spatial joins, and are defined on one or more layers by projecting the relevant attributes from each. Object classes are also used for defining subtypes of existing object classes. Integrity constraints are first order logic statements with variables ranging over entities and predicates containing arithmetic, string, comparison, spatial, and topological operators. These are used to process queries by selecting object classes that satisfy some specified constraints.

Limitations of the GRDM include inability to handle temporal data, and the lack of integration of spatial and relational data. The implementation is thus left to existing GIS platforms that handle layers and relations separately.

0.9.2 Extended relational model for GIS

Another extension of the relational model to the GIS domain is described in [34] that defines spatial relations in an extended relational framework. It assumes that the set of spatial features of interest REG is a subset of a universal region R. This leads to the closure property of REG for the operators union, intersection, subtraction, and complementation. The model follows a weak data typing for REG so that it includes points and curves as well that may result from spatial operations over regions that are members of REG.

Spatial information is incorporated into the model at the level of the assignment of attribute values. Functions are used to assign attribute values to spatial regions. Thus, a function is defined as a mapping from a spatial region to the attribute domain. Spatial operators are defined over regions as returning a set of points that satisfy the conditions expressed in the operation.

A spatial tuple consists of a series of value assignments to regions with the same spatial domain. A spatial relation is a non-empty, finite set of tuples. The key attribute is defined as the union of several regions having the same value (such as county name). Each of these regions may have separate values for other attributes (such as crop type). As for traditional relations, the key attrbute has

a unique value for each tuple. Relations can be restructured by by reassigning its key attribute and forming a weakly equivalent relation. For example, a relation with *county name* as its key atribute and *crop type* as a non-key attribute can be restructured to another relation with *crop type* as the key attribute and *county name* as the non-key attribute. This would result in restructuring the tuples and forming a new relation with the same information content as the old one.

One limitation of the model is that spatial operations are left to the implementation level, which reduces its effectiveness as a high-level model. Another limitation is the large amount of information redundancy in spatial relations, which is also left to be resolved at the implementation level.

Other significant work on extending relational database technology to spatial domains include the extension of the Starburst project [55] and the Sequioa 2000 project [40].

0.10 GIS extensions of the object-oriented data model

The object-oriented data model, with its class hierarchy, methods, and the inheritence mechanism for attributes and methods provides a conceptualization that is well-suited for GIS applications. As a result, a lot of research (e.g., [42, 60, 26, 39, 19, 75] has been done on extending the object-oriented data model to GIS applications.

Bi-Level Ojcet-oriented GIS Data Model

A extended object-oriented data model for GIS applications is described in [19] that facilitates data modeling at two levels: geographic and geometric. The geographic level deals with actual physical features such as countries, zones, and roads. At the geometric level, the data model deals with the spatial representations of the geographic counterparts. Three main constructs are used at each level: object, object class, and functions. Objects are instances of object classes, and divided into two categories: ***primitive*** and ***non-primitive***. Primitive objects are of system built-in data types such as integer, real, string, and boolean. Non-primitive objects are of derived data types such as paths, physical regions, and themes at the geographic level, and points, lines, and polygons at the geometric level. Objects can be created initially or be query generated. Objects can be transient or persistent, depending on whether they are exist temporarily and lost afterwards, or stored explicitly in the database respectively. Object classes are used to organize objects by *ISA* and *PART-OF* hierarchies. The PART-OF hierarchy models the aggregation of component objects of a parent object and represents containment relationship. The componnet objects may or may not be spatially overlapped. For example, an *airport* object can have *terminal*, *runway*, and *controltower* as component objects, which are non-overlapping. On the other hand, the object *country* can have as component objects, *province* (classified by political boundaries) and *region* (classified by economic functions), which

are spatially overlapping. The PART-OF hierarchy allows upward propagation of attributes in two ways: *selective upward aggregation* and *upward containment transitivity*. An example of the use of selective upward aggregation is the calculation of the population of a country as the sum of those of its component provinces. An example of upward containment transitivity is that if the object *city* is a component object of *province*, which in turn, is a component object of *country*, then *city* is a component object of *country*.

The *ISA* hierarchy models the subclass/superclass relationship with attribute inheritence. Multiple inheritence is allowed. Only the leaf nodes of an ISA hierarchy of geographic objects have direct representations at the geometric level.

Functions are used for data manipulation at both geographic and geometric levels, for query processing. Functions for manipulation of geographic objects include *semantic retrieval*, *set*, *aggregate*, and *superfunctions*. Semantic retrieval include tools for manipulation of spatial data for information such as *Adjacency*, *Area*, *Complement*, and *Intersection*. Retrieval functions are used for retrieving attribute values such as population of a city, and ownership of a plot of land. Set functoins are used for set-oriented operations such as *Union*, and *Distinct*. Examples of aggregate functions include *Minimum*, *Maximum*, and *Average*. Superfunctions incorporate the extensibility of the data model beyond the system-defined functions. These are programmer-defined procedures that may use the pre-defined functions. Examples of superfunctions are *Closest* and *ShortestRoute*. Geometric functions are used for manipulation og geometric objects and have been categorized under five groups: *overlap* (of polygons), *containment* (of polygons, lines, and points), *component* (such as the line segments defining a polygon), *geometric object computation* (such as calculating the centriod of a polygon), and *arithmetic computation* of attributes of geometric objects (such as the area of a polygon).

GraphDB

Many GIS applications require modeling and querying network data. Examples include transportation network such as roadways, subway systems, utility networks including pipelines for water, gas, electricity, and sewarage systems, and natural resource planning for river networks. A major feature to be modeled in network design is connectivity. GraphDB [42] describes a method for modeling public transportation networks for bus, tram, and train lines and schedules using a graph structure. The network is modeled at three levels: physical network, lines, and schedules. Each of these levels are modeled using three types of classes: *simple class*, *link class*, and *path class*. Simple classes have attributes with values as data types such as integer or string, and object types such as a reference to another object. Link classes represent edges and are simple classes with two references to *source* and target objects. Path classes are simple classes that have a list of references to node and edge objects that form a path over a graph.

The physical network models switches (vertices) as simple class, arcs as link class joining two vertices, and physical routes as path class as list of arcs in the

network. The second level introduces stations or stops as simple class, connection as link class, and line as path class. The third level depicts time schedules in terms of simple classes *arrival* and *departure*, link classes *travel*, *stay*, *change*, and *wait*, and path class *trip*. The third level of the data model utilizes the constructs defined in the second and first levels, and the second level makes uses the modeling constructs defined in the first level of the data model.

The GraphDB data model uses the geometric constructs *point*, *line*, and *region* defined in ROSE algebra.

Example of queries supported are:

1. List all departures from station X in terms of time, type and train number, end station, and arrival ime.
2. How many countries must be travelled (by land) to go from country X to Y?
3. List all direct connections from station X to station Y with travel time and distance.
4. List all cities, villages, and rivers in state X.
5. Find the shortest path from point X to Y, avoiding region Z.

0.11 Extensible data models for GIS applications

An extensible database management is one that supports the incorporation of new features. Examples of the type of extensions required include new data types, new operations on the data, new operators for the query language, and new access and storage methods [45]. Several extensible DBMSs, such as PROBE [63], DASDBS [81, 69], Starburst [46, 55], and GRAL [41] have been proposed for GIS applications.

0.11.1 The PROBE Data Model

The PROBE data model (PDM) [63] uses two basic modeling constructs: entities and functions. Entities are objects with unique identities. Entities with similar characteristics are grouped together as entity types. Examples of entities include geographic objects such as *city*, *land-division*, geometric objects such as *point-feature* and *polygon*, and non-spatial objects such as *owner* of a land parcel. Entities can be grouped into *generalization hierarchies* with subtype/supertype relationships among objects. Geometric and geographic objects are intermingled in the generalization hierarchy. For example, the geometric object *point-feature* can have the geometric object *city* as its subtype. Functions are used for modeling the properties of entities (such as population of a city), relationships among entities (such as the ownership relationship between a land-parcel and its owner), and operations (such as overlaying two layers) that can be performed on entities. Functions are divided into two categories: *computed*, whose values are obtained

through some procedure, and *stored*, whose values are obtained by a conventional database search. Spatio-temporal objects are modeled in PDM in terms of *POINT-SET*s. A point-set is the set of points occupied by a spatial object. Several operations are defined on point-sets that are divided into *point-set* operations and *structural* operations. Point-set operations include *spatial selection*, *overlay*, and *set operations*. Spatial selection involves selecting point-sets that satisfy predicates such as *empty*, *intersect*, and *contains*. The overlay operation is defined in terms of a *uniform space*, which is a maximal subspace, every point of which is contained by the same set of objects. Uniform regions give rise to the finest partitioning of a space. The overlay operator returns the spatial objects for all uniform regions of a space. Polygon overlay is implemented by applying union of the spatial objects and then applying the overlay operator on the resulting space. Structural operations are concerned with the hierarchical structure of the space, and incude object set operations such as union, intersection, and difference of spatial objects, as well as operations that restructure the hierarchy of spatial objects. Those belonging to the later category include *expand* and *reduce*. The expand operator increases the set of immediate objects by copying lower level objects to the set of immediate objects for a particular node in the object tree structure. The reduce operator is the inverse of the expand operator: it removes all immediate children nodes that are also contained by by some other child object. It also supports recursive traversals of the hierarchical structure of the contains relationship.

0.11.2 DASDBS

The DASDBS (DArmStadt DataBase System), reported in [81] is an extensible database system, designed for non-traditional applications such as office automation systems and GIS. The basic components include *complex record system* (CRS) and *access manager* (AM). The complex record system is an application-independent storage system that stores hierarchically structured data. It consists of a high-level user interface and an efficient storage manager. The access manager maps application objects to CRS objects, and allows the definition of new externally defined data types (EDT). The interface between data types defined in application programs and those in internal database representation. This necessitated the definition of two types of conversion routines: *IN*, which converts user-defined types to database types, and *OUT*, which converts database types to user-defined types. User-defined procedures are also allowed. When a user-defined procedure is invoked, the database parameters passed to it are first converted to user-defined types. A key component of the interface is the EDT preprocessor, which analyzes the type declarations in a user-defined application program. It also creates links between application programs and the database.

0.11.3 Starburst

Starburst [45, 46, 55] is an extensible database management system based on the relational data model. The primary components consists of a quey proces-

sor, called Corona, and a data manager, called Core. Two types of extensions are allowed in Starburst: *storage methods* and *attachments*. Storage methods allow alternative methods for implementing tables. Users are allowed to use a default implementation provided by the system, or specify their own methods of implementation. Access paths, integrity constraints, and triggers can be defined on tables using attachments. Starburst can model and capture disparate types of data, such as thematic map layers, topological data, and attribute data. Furthermore, attribute and geographic data are integrated in the same schema, which leads to the usage of spatial and standard operators in expressing queries. User-defined types such as complex object (using a user-defined type facility called Polyglot), user-defined functions, and production rules such as specifying a closed loop on a set of line segments are allowed. Starburst also provides data abstraction facilities to link application programs written as external packages.

0.12 Other data models for GIS applications

0.12.1 Topological model

Discretizing spatial data has been reconginzed as a problem in modeling geographic data. The problem stems from the fact that the geographic space is continuous, and yet only discrete representations are possible in computational representation and manipulation. This can result in the introduction of topological errors in the representation and computation of spatial data. This problem has been traditionally dealt with by limiting such error propagation (for example, in the Greene-Yao algorithm [37]). In order to address this problem, a data model, based on combinatorial topology, has been proposed in [32]. Spatial objects are classified by their dimensions. For each dimension, the mininal object is called a *simplex*. A simplex of dimension n consists of $(n + 1)$ numbers of $(n - 1)$ simplices. For example, the simplex in two dimension is the triangle, which consists of three 1-simplices (line segments). In this model a 0-simplex is a node and a 1-simplex is an edge. An n-simplex also consists of a number of faces whose dimensions range from 0 through n. Thus the faces of a triangle are: three 0-simplices (vertices), three 1-simplices (edges), and one 2-simplex (the triangle itself). Simplices also have orientation. Thus, the 1-simplex (line segment) has directions, and the 2-simplex (triangle) has counterclockwise (or clockwise) orientation. A finite set of simplices can form a *simplical complex*, if and only if intersection of any two simplices in the set either forms a simplex or a face of both the simplices. The *boundary* of a simplex is an ordered set of all of its faces. The boundary of a simplical complex is the sum of that of its constituent simplices (edges). Since edges have orientations, the sum of two edges with opposite orientations is zero. This enables the computation of the boundary of a simplical complex as the sum of the boundaries of its constituent simplices. Those portions of the boundaries (of the constituent simplices) that fall in its (the simplical complex's) interior cancel out in the addition process. The *co-boundary* of a simplex is defined as the set of all simplices that share

it as their common boundary. Thus, the co-boundary of an n-simplex can be obtained by intersecting it with the boundaries of the $(n+1)$-simplices. If the intersection returns a non-empty set, then it is a co-boundary. The model results in a complete partition of the space, such as a TIN in a two-dimensional space. Such a complete partition is obtained by two completeness axioms. The first axiom, called the *completeness of incidence*, requires that the intersection of two n-simplices is either empty, or a common face. The second axiom, called the *completeness of inclusion*, requires that every n-simplex is a face of an (n+1)-simplex. The model also proposes algorithms for insertion of nodes, lines, and polygons.

0.12.2 Realm

The purpose of the Realm data model [43, 44] is to provide a formal definition of spatial data types (SDTs) for GIS applications. Several criteria are used in developing the SDTs including *generality*, *rigorous definition*, *finite resolution*, *treatment of geometric Consistency*, and *general object model interface*.

Generality ensures closure properties of SDTs for the spatial operations defined over them. Rigorous definition requires unambigous definitons of SDTs and the functions and operations.

Finite resolution refers to the fundamental problem of the finiteness of storage and processing of numerical data in computerized implementation of GIS applications. Since such applications deal with data from continuous surfaces, such limitations can introduce numerical and topological errors.

Geometric consistency refers to maintaining the geometric constraints in the spatial relationships among geographic objects. For example, the common boundary between two adjacent geographic objects should be preserved in the SDTs used in defining them.

General object model interface refers to the high-level interfaces that integrate SDTs to database management systems. Such interfaces are required for development of GIS (and other) applications. The definitions of the SDTs are, however, independent of the implementation of the DBMS.

The basic building block consists of a finite, graphical data structure called realm, which forms the basis for the definition of other spatial data types. The model is described using three layers: geometric primitives are defined in the bottom layer, certain structures are presented in the next layer, and the spatial data types are defined in the top layer. Each of these are described below.

1. The bottom layer describes a finite, discrete two-dimesional space $N \times N$, in terms points, line segments, and functions that describe spatial relationships among the points and line segments in this space. A point (called an $N-point$) is represented by a pair of coordinates $(x, y) \in N \times N$. A line segment (called an $N-segment$) is described in terms a pair of N-points (p, q). The predicates in this layer take point coordinates as integer parameters and return boolean or integer values. Predicates on points consist of $=$, *on*, and *in*, where $=$ checks if two points have the same coordinates, *on*

tests whether the point lies on the line segment specified, and *in* returns *true* if the point is one of the end-points of the line segment. Predicates on line segments consist of $=$, *intersect*, *parallel*, *overlap*, *aligned*, *meet*, and *disjoint*, where two lines *meet* if they have exactly one end-point in common, *overlap* if they are collinear and share a common segment, *aligned* if they are collinear but do not share a common segment, and *disjoint* if they are neither equal nor meet or aligned. *Intersect* and *parallel* have their usual meanings.

In Realm, the database stores a spatial object along with the segments and points that constitute its spatial components. The spatial objects are linked with its spatial components through a set of pointers called *realm object identifiers* (ROID). Another set of pointers, called *spatial component identifiers* (SCID) link the components with the spatial objects. The two-way links provide fast access and update of spatial data. The basic operations consists inserting and deleting Realm objects into a Realm representation that consists of several types: *Point* refers to the set of points(P_N),*Segment* refers to a set of line segments(S_N), *RealmObject* represents a union of P_N and S_N, two sets of ROIDs and SCIDs, and *Rectangle*, that refers to the set of rectangles forming the underlying representation. Two additional types *Bool* and *Integer* refer to the sets of boolean and integer values permitted as results of functions and predicates.

The operations performed in realm are grouped into three categories: updating Realm objects, linking between realm and database objects, and identify realm objects based on a specified criterion.

Update operations include inserting and deleting points and segments. Realm requires line segments to intersect only at realm points, and in such cases two intersecting line segments are broken up into four segments with each having an end point at the point of intersection. If two segments intersect at a point which is not a realm point, the segments are redrawn to make them pass through the nearest realm point. This requirement ensures that all operations can be performed by integer arithmetic. In order to to limit the drift of line segments caused by this requirement, an *envelop* is defined for each line segment (using the Greene-Yao algorithm [37]) as a set of grid points that lie on, and immediately above or below the line segment. Thus, inserting a new line segment that intersects with an existing segment require redrawing according to the above critarion. For insertion of new points the concept of envelop of a line segment is extended to define a "proper envelop" as a subset of the envelop that does not contain any of the end points of the line segment. If a new point falls in the "proper envelop" of a line segment, the line is redrawn to include it as part of the line segment.

Operations that deal with the Linking of realm objects with their corresponding spatial components stored in the database, consist of *register*, *unregister*, *GetSCIDs*, and *GetRealmObject*. Register links a realm object

with its spatial components through its *roid*. Unregister removes such links. *GetSCIDs* returns the *SCIDs* of the spatial components for a specified realm object, and *GetRealmObject* returns its underlying geometry.

Two operations that are used to identify realm objects include *window*, and *identify*. *Window* takes a realm and a rectangular space as input and returns all real objects that either fall inside or intersect with it. *Identfy* takes a realm, an *N-point*, and a specified distnace as input parameters, and returns all realm objects that fall within the specified distance from the given *N-point*.

2. At the next level, several structures and their inter-relationships are defined.The structures include *R-cycle*, *R-face*, *R-unit*, and *R-block*. An *R-cycle* is a bounded polygon whose edges are *R-segments* (segments defined in the realm structure). An *R-point* (a point defined in the realm structure) can be as *in*, *on*, or *out* an *R-cycle*. Several topological relationships are among between *R-points*, *R-segments*, and *R-cycles*. Regions with holes are defined using two additional constructs: *R-faces* and *R-unit*. An *R-face* is an *R-cycle* that encloses a set of disjoint *R-cycles*. Thus, an *R-face* corresponds to a region with holes. An *R-unit* is dfined as the minimal *R-face*. An *R-block* is a set of line segments in which every segment is connected (either directly or through some other segments in the set) with every other segment in the set.

3. The basic spatial data types include *points*, *lines*, and *regions*, that consist of sets of *R-points*, *R-segments*, and *R-blocks* respectively. The *lines* and *regions* are furher defined in terms of the constituent (pairwise disjoint) *R-blocks* and *R-faces* (that do not have any shared boundaries) respectively. Several set-oriented operators such as *union*, *intersection*, and *difference*, and topological operators such as *disjoint*, *adjacent*, *inside* are defined over the spatial data types.

0.12.3 Open geodata interoperability specification

The purpose of the OGIS geodata model [59] is to develop a common language for building data and communicating models among cooperative GIS applications. The basic strategy is to develop building blocks for representing subsets of space and time that are required for GIS applications, using a set of well-known primitive types such as integer and real, and common aggregate types such as list and tuple. New data types are recursively or directly constructed by factories or constructor interfaces, that are defined as part of every type definition. The OGIS model defines a type as a well-defined set of programming interfaces that describes its behavior without specifying storage mechanism or method of implementation. A special type, called the root type, is defined as the highest level in the type hierarchy from which all subtypes are derived. A class is a set of algorithms that implement the behavior of a type. A type constructor contains

protocols for developing new types. An object is an instance of a class of which it is a member.

The OGIS model defines spatial and temporal interfaces. These include interval that specifies the lower bound, upper bound, and inclusion of a value in the interval; sorted lists and sorted lists of intervals; and curve of an arbitrary type that can be used in many aspects of geographically related geometries. The model also specifies geodata definitions of spatial, temporal, and spatial-temporal domains, feature, coverage, and attributes. Geometric structures are derived from simple structures, and are categorized into simplex, simplical complex, and cell types.

Geoprocessing functions can perform operations on geographical data. These functions may be defined in the database, developed by users, or provided by outside vendors. OGIS specifies a multi-method object model that allow operations to be performed on more than one object class.

0.12.4 Computational modeling system

The purpose of the computational modeling system (CMS) [70] is to develop a computational modeling environment and support the representation, evaluation, and manipulation of scientific concepts in the modeling process. The symbolic representation of a phenomenon is characterized by a set of concepts, a corresponding set of symbolic representations, and a set of operations that manipulate them. An interpretation maps the phenomenon to its symbolic representation. The operations transform one set of representation into another. Representations of the concepts are formalized with representational structures (R-structures). An R-structure is a triple consisting of a representational domain (D), a set of transformations (T), and a finite set of representations that are given explicit forms and have particular significance in the application domain (I). The representational domain (D) is a set containing all instances of some concept, and consists of informtive representations that contain computable representations (such as area and centroid for a polygon), and nominal representations that are arbitrary assignment of names. A transformation (T) represents mapping between R-structures, and includes equality relationships, intersection relationships, and spatial projections. The set of explicit instances (I) include all instances of an R-structure to which frequent references are made for modeling purposes. New R-structures can be constructed from R-structures that have already been defined, using a set of aggregate constructors such as *tuple constructors, set constructors, sequence constructors,* and *set constructors.* The constructors enable creation of new R-structures from previously defoned R-structures. A set of constraints determine which elements are selected from the domain of previously defined R-structures in the construction process. Super-domain/sub-domain relationships are also supported. *Equivalence classes* define distinct but equivalent representations of a given concepts. For example, a polygon may be represented as sequences of points, sequences of line segments, or trees of half planes. A computational modeling language (CML) is suggested that can support the creation, access and manipulation of R-structures. Thus,

CMS provides a unifying framework for integrating distributed modeling environment including DBMS and mathematical softwares. The CMS has been applied to an earth science application and a system called Amazonia has been implemented.

An example of the process of data modeling using the framework of the conceptual modeling system is discussed in [70], in the context of hydrological modeling. The purpose of the model is to study the effect of land surface characteristics and rainfall on the flow of water in a river basin. The land surface characteristics are obtained from digital elevation models (DEMs) providing the land surface elevations at points on a coordinate system. The flow and direction of water is determined by factors such as surface slope, channel segment, and drainage basin, which are derived from the DEMs using some "tranformation functions". The flow of surface water is characterized by surface flow vectors in terms of time, location, direction, and magnitude, and represented by "surface flow models". The flow models are used to generate information such as hydrographs, which are then iteratively refined by comparing them with observed hydrographs.

0.13 Data model for GIS applications: future research issues

Many of the current researh efforts have been focused on developing models for specific applications. In this respect, these efforts are quite different from data models for business data processing that are generic and applicable across application areas. This may be attributed to the varied nature of the data requirements across GIS applications, the complexity of dealing with spatial and attribute data, and relatively shorter history of research. However, it is still a worthwhile effort to determine the generic nature of data requirements across applications. One approach that has been taken to serve application developers across various application areas is to provide with tools that allow new data types and functionalities to be added to an existing data model. Models in this category have been termed *extensible models*, and we have discussed some of these models in Section 0.11. Below, we discuss some of the issues related to the development of extensible data models and then other issues that need to be addressed across GIS applications.

- Extensibility: Extensibility refers to the ability to add new features and integrating them to those of an existing system. Extensibility is required at various levels including user interface, query language, and storage methods [45]. At the user interface level, user must be allowed to define new data types and operations on the data. At the query language level, extensibility requires new language extensions for expressing operations involving spatial relationships, and efficient access methods for spatial data resulting from such operations. At the data storage level, introduction of new storage structures as well as storage media must be allowed. Furthermore, it is

important to be able to integrate new data types and operations with the existing ones seamlessly. Also important is the ability to protect against extensions that are in the "wild write" category [45], which refers to improper extensions that contradict others and thus nullify the effect of each other. It is important to have an integrity mechanism that would detect and protect the model against such extensions. At the user interface level, users must be provided with design tools that can be used to extend an existing model, and also detect the changes necessary in other parts of the system (such as access methods) as a result of the extension.

- Topological relationships among geographic entities: Topological relationships are those that are not affected under homeomorphic transformations of the spatial objects, such as rotation, translation, or scaling. Examples of topological relationships include disjoint, meet, equal, inside, covers, contains, and overlaps [31]. Methods of deriving new topological relations from existing ones must be included in the developing data models for GIS applications.

- Different data formats: As we have discussed previously, many different representations have been used in GIS applications. The two broad classifications are the object-based and field-based representations. Under these two categories, numerous variations also exist. Hence, the data model must provide interfaces to a variety of data formats and probide interoperability among them.

- Using geometric algorithms to process geographic data: We have studied and described a variety of geometric algorithms that are used to manipulate geographic objects that are represented by their geometric counterparts. Modeling spatial data for GIS applications can be done using a bi-level model (as in [19]) or by allowing interleaved definitions of geographic and geometric data (as in [63]). Geographic data can be represented by different geometric counterparts in a context-dependent manner. For example, a city may be represented as a point in a large-scale map, and as a polygon in a small-scale map. The geographic data model must be integrated with the geometric model in a seamless manner, such that the appropriate algorithms are automatically invoked in a contex-sensitive manner.

Bibliography

[1] F. P. Agterberg. Computer Programs for Mineral Exploration. *Science*, 245:76–81, 1989.

[2] F. P. Agterberg, G. F. Bonham-Carter, and D. F. Wright. Statistical Pattern Integration for Mineral Exploration. In Gaal. G., editor, *Computer Applications in Resource Exploration*, pages 1–22. Pergamon Press, Oxford, 1990.

[3] B. G. Baumgart. A Polyhedron Representation for Computer Vision. *Proceedings of the AFIPS National Conference*, 44:589–596, 1975.

[4] J. R. Beaumont. GIS and Market Analysis. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 139–151. Longman Scientific and Technical, New York, 1991.

[5] J. K. Berry. The Use of a Geographic Information System for Storm Runoff Prediction from Small Urban Watersheds. *Environmental Management Journal*, 11(1):21–27, 1987.

[6] J. K. Berry. GIS in Island Resource Planning: A Case Study in Map Analysis. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 285–295. Longman Scientific and Technical, New York, 1991.

[7] J. K. Berry. Cartographic Modeling: The Analytical Capabilities of GIS. In *Environmental Modeling with GIS*, pages 58–74. Oxford University Press New York, 1993.

[8] J. K. Berry and Berry J. K. Assessing Spatial Impacts of Land Use Plans. *International Journal of Environmental Management*, 27:1–9, 1988.

[9] G. F. Bonham-Carter. Integration of Geoscientific Data Using GIS. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 171–184. Longman Scientific and Technical, New York, 1991.

[10] G. F. Bonham-Carter, F. P. Agterberg, and Wright D. F. Integration of Geological Data Sets for Gold Exploration in Nova Scotia. *Photogrammetric Engineering and Remote Sensing*, 54(11):1585–1592, 1988.

[11] G. F. Bonham-Carter, F. P. Agterberg, and D. F. Wright. Weights of Evidence Modelling: A New Approach to Mapping Mineral Potential. *Geological Survey of Canada Paper*, 98-9:171–183, 1990.

[12] J. Bouma. Land Qualities in Space and Time. In *Proceedings of a Symposium Organised by the International Society of Soil Science (ISSS)*, pages 3–14, 1989.

[13] J. Bouma. Using Soil Data for Quantitative Land Evaluation. In *Advances in Soil Science*, volume 9, pages 177–213. Springer-Verlag, 1989.

[14] P. A. Burrough. Soil Information Systems. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 153–169. Longman Scientific and Technical, New York, 1991.

[15] P. A. Burrough and A. U. Frank. Guest Editorial: Concepts and Paradigms in Spatial Information: are Current Geographical Information Systems Truly Generic? *International Journal of Geographical Information Systems*, 9(2):101–116, 1996.

[16] P. A. Burrough and A. A. de Veer. Automated Production of Landscape Maps for Physical Planning in the Netherlands. *Landscape Planning*, 11:205–226, 1984.

[17] D. R. Chand and S. S. Kapur. An Algorithm for Convex Polytypes. *Journal of the ACM*, 17(1):78–86, 1970.

[18] P. P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *Communications of the ACM*, 13(6):377–387, 1976.

[19] A. Choi and W. S. Luk. Using an object-oriented database system to construct a spatial database kernel for GIS applications. *Computer Systems and Engineering*, 7(2):100–121, 1992.

[20] R. Chorley and R. Buxton. The government setting of gis in the uinted kingdom. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, pages 67–79. Longman Scientific and Technical, New York, 1991.

[21] H. Claussen, J. Siebold, L. Heres, and P. Lahaije. A Proposed Standard for Digital Road Maps to be used in Car Navigation. In Reekie D. H. M., Case E. R., and J. Tsai, editors, *Vehicle Navigation & Information Systems Conference, Toronto, IEEE*, pages 324–330. 1989.

[22] E. F. Codd. A Relational Model of Data for Large Shared Data Banls. *Communications of the ACM*, 13(6):377–387, 1970.

[23] T. Connolly, C. Begg, and A. Strachan. *Database Systems: A Practical Approach to Design, Implementation and Management.* Addison-Wesley, 1996.

[24] P. F. Dale. Land Information Systems. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 85–99. Longman Scientific and Technical, New York, 1991.

[25] J. Dangermond and C. Freedman. Findings Regarding a Conceptual Model of a Municipal Database and Implications for Software Design. *Geo-Processing*, 3:31–49, 1986.

[26] R. David, I. Ranyal, G. Schorter, and V. Mansart. GeO2: Why Objects in a Geographical DBMS. In D. Abel and B. C. Ooi, editors, *Advances in Spatial Databases (Proceedings of the Third Symposium on Databases) volume 692 of Lecture Notes in Computer Science*, pages 264–276. Springer-Verlag, 1993.

[27] T. H. Davnport and M. Hammer. How Executives can shape their Company's Information Systems. *Harvard Business Review*, 67(1):130–134, 1989.

[28] V. Delis, T. Hadzilacos, and N. Tryfona. An Introduction to Layer Algebra. In *Advances in GIS Research: Proceedings of the 6th International Symposium on Spatial Data Handling*, 1994.

[29] P. M. Driessen. Quantified Land Evaluation: Consistency in Time and Space. In *Land Qualities in Space and Time. Proceedings of a Symposium organized by the International Society of Soil Sciences (ISSS), Wageningen*, pages 3–14. 1989.

[30] J. B. Dumaski and C. Onofrei. Crop Yield Models for Agricultural Land Evaluation. *Soil Use and Management*, 5:181–187, 1989.

[31] M. J. Egenhofer. Reasoning about Binary Topological Relations. In O. Gunther and H J Schek, editors, *Advances in Spatial Databases, Second Symposium SSD '91*, pages 143–160, 1991.

[32] M. J. Egenhofer, A. U. Frank, and J. P. Jakson. A Topological Data Model for Spatial Databases. In A. Buchman, O. Gunther, T. R. Smith, and Y. F. Wang, editors, *Design and Implementation of Large Spatial Databases, First Symposium SSD '89*, pages 271–286, 1989.

[33] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* Benjamin/Cummings Publishing Company, second edition, 1994.

[34] S. K. Gadia and V. Chopra. A Relational Model and SQL-like Query Language for Spatial Databases. In N. R. Adam and B. K. Bhargava, editors, *Advanced Database Systems, Lecture Notes in Computer Science 759.* Springer-Verlag, 1993.

[35] J. Gateaud. The Use of Cartography Databases in Multi-purpose Utility Applications- an Experience Report. In *Proceeedings of AM/FM International -European Division Conference Montreux*, pages 15–18, 1988.

[36] R. L. Graham. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters*, 1:132–133, 1972.

[37] D. Greene and F. Yao. Finite Resolution Computational Geometry. In *Proceedings of the 27th IEEE Symposium on the Foundations of Computer Science*, pages 143–152, 1986.

[38] L. Guibas and J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[39] O. Gunther and W.-F. Riekart. The Design of GODOT: An Object-Oriented Geographic Information System. *Bulletin of the Technical Committee on Data Engineering, Special Issue on Geographical Information Systems*, 16(3):4–9, September 1993.

[40] S. Guptil and M. Stonebraker. The Sequoia 2000 Approach to Managing Large Spatial Object Databases. In D. Cowen, editor, *Fifth International Symposium on Spatial Data Handling*, pages 642–651, 1992.

[41] R. H. Guting. Gral: An Extensible Relational Database System for Geometric Applications. In P. G. Apers and G. Wiederhold, editors, *15th International Conference on Very Large Databases*, 1989.

[42] R. H. Guting. An Introdution to Spatial Database Systems. *VLDB Journal*, 3(4):357–399, October 1994.

[43] R. H. Guting and M. Scheneider. Realms: A Foundation for Spatial Data Types in database Systems. Technical Report NR. 134-11/1994, 1992.

[44] R. H. Guting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. Technical Report NR. 141-3/1993, 1993.

[45] L. M. Haas and W. F. Cody. Exploiting Extensible DBMS in Integrated Geographic Information Systems. In *Advances in Spatial Databases, 2nd Symposium, SSD '91 and Lecture Notes in Computer Science 255.* Springer-Verlag, 1991.

[46] L. M. Haas, W. Lohman, J. McPherson, P. F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. Carey, and E. Shekita. Starburst Mid Flight: As the Dust Clears. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):143–160, 1990.

[47] T. Hadzilacos and N. Tryfona. Logical Data Modelling for Geographical Applications. *International Journal of Geographical Information Systems*, 10(2):179–203, 1996.

[48] E. G. Hoel and H. Samet. A qualitative comparison study of of data structures for large line segment databases. In *Proceedings of the ACM SIGMOD Conference*, 1992.

[49] L. Holstein. Lis problems and issues in urban areas. In *Proceedings from the FIG Land Information Systems Workshop, Bali, Indonesia*, pages 53–59, 1988.

[50] K. K. Kemp. Spatial Databases: Sources and Issues. In *Environmental Modeling with GIS*, pages 363–371. Oxford University Press New York, 1993.

[51] P. Kotler. *Marketing Management: Analysis, Planning, Implementation, and Control.* Prentice-Hall, Englewood Cliffs, 1988.

[52] K. Kovar and H. P. Nachtnebel. International Association of Hydrological Sciences, Oxfordshire, April 1993.

[53] J. van Kuilenburgh, J. J. de. Gruijter, B. A. Marsman, and J. Bouma. Accuracy of Spatial Interpolation between Point Data on Soil Mositure Capacity, Compared with Estimates from Mapping Units. *Geoderma*, 27:311–325, 1982.

[54] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems.* Academic Press, 1992.

[55] G. Lohman, B. Lindsay, H. Pirahesh, and K. B. Scheifer. Extensions to Starburst: Objects, Types, Functions, and Rules. *Communications of the ACM*, 34:94–109, 1991.

[56] T. R. Loveland and B. Ramey. Applications of US Geological Survey Digital Cartographic Products, 1979-1983. *US Geological Survey Bulletin 1583*, 1986.

[57] R. P. Mahoney. GIS and Utilities. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 101–114. Longman Scientific and Technical, New York, 1991.

[58] D. R. Maidment. GIS and Hydrologic Modeling. In *Environmental Modeling with GIS*, pages 147–167. Oxford University Press New York, 1993.

[59] The OGIS Project Members. The open geodata interoperability specification. draft base document-ogis project document 94-025r1. Technical report, The OGIS Project, October 1994.

[60] P. Milne, S. Milton, and J. L. Smith. Geographical Object-oriented Databases: A Case Study. *International Journal of Geograhic Information Systems*, 7:39–56, 1993.

[61] D. E. Muller and F. P. Preparata. Finding the Intersection of Two Convex Polyhedra. *Theoretical Computer Science*, 7(2):217–236, 1978.

[62] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley and Sons, 1992.

[63] J. A. Orenstien and F. A. Manola. PROBE Spatial Data Modeling and Query Processing in an Image Database Application. *IEEE Transactions on Software Engineering*, 14(5):611–629, May 1988.

[64] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1994.

[65] J. O'Rourke, C. B. Chien, T. Olson, and D. Naddor. A New Linear Algorithm for Intersecting Convex Polygons. *Computer Graphics and Image Processing*, 19:384–391, 1982.

[66] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[67] T. K. Puecker and N. Chrisman. Cartographic Data Structures. *The American Cartographer*, 2(1):55–69, 1975.

[68] J. F. Raper and B. Kelik. Three-Dimensional GIS. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, pages 299–317. Longman Scientific and Technical, New York, 1991.

[69] H.-J. Schek. The DASDBS Project: Objectives, Experiences, and Future Propects. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), 1990.

[70] T.R. Smith and et al. Computational modeling systems. *Information Systems*, 19(4):1–27, 1994.

[71] J. Star and J. Estes. *Geographic Information Systems: An Introduction.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1989.

[72] C. D. Tomlin. *Geographic Information Systems and Cartographic Modelling.* Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[73] C. D. Tomlin. Cartographic Modelling. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, pages 361–374. Longman Scientific and Technical, New York, 1991.

[74] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, Rockville, M.D., 1988.

[75] P. van Oosterom and Jan van den Bos. An Object-Oriented Approach to the Design of Geographic Information Systems. In *Design and Implementation of Large Spatial Databases: First Symposium SSD '89 and 409 of Lecture Notes in Computer Science.* Springer-Verlag, 1989.

[76] M. von Braun. The Use of GIS in Assessing Exposure and Remedial Alternatives at Superfund Sites. In *Environmental Modeling with GIS*, pages 339–347. Oxford University Press New York, 1993.

[77] R. Weibel and M. Heller. Digital Terrain Modelling. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 1, pages 269–297. Longman Scientific and Technical, New York, 1991.

[78] R. Weibel and M. Heller. Integrated Planning Information Systems. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 297–310. Longman Scientific and Technical, New York, 1991.

[79] K. Weiler. Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. *Computer Graphics Applications*, 5(1):21–40, 1985.

[80] M. White. Car Navigation Systems. In D. J. Maguire, M. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, volume 2, pages 115–125. Longman Scientific and Technical, New York, 1991.

[81] A. Wolf. The DASDBS GEO-Kernel: Concepts, Experiences, and the Second Step. In O. Gunther and T. Smith, editors, *Design and Implementation of large Spatial Databases, proceedings of SSD'89, Santa Barbara, CA, Lecture Notes in Computer Science, No. 409*, pages 67–88. Springer-Verlag, 1989.

[82] M. Worboys. *GIS: A Computing Perspective.* Taylor & Francis Inc., 1995.

NASA

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle
Geodata Modeling and Query in Georgraphic Information Systems

5. Report Date

6. Performing Organization Code

7. Author(s)
Nabil Adam

8. Performing Organization Report No.

10. Work Unit No.

9. Performing Organization Name and Address
Rutgers University
180 University Avenue
Newark, New Jersey 07102

11. Contract or Grant No. NAS5-32337
USRA subcontract No. 5555-46

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, DC 20546-0001
NASA Goddard Space Flight Center
Greenbelt, MD 20771

13. Type of Report and Period Covered Final
August 1995 - July 1996

14. Sponsoring Agency Code

15. Supplementary Notes
This work was performed under a subcontract issued by
Universities Space Research Association
10227 Wincopin Circle, Suite 212
Columbia, MD 21044 Task 56

16. Abstract

This report discussed in seven chapters the challenges of Geographic Informtion Systems which deals collecting, modeling, managing, analyzing, and integrating spatial (locational) and non-spatial (attribute) data required for geographic applications. Other challenges include dealing with spatio-temporal data, providing multivalued logic capabilities, and handling extremely long transactions.

The approaches in the report included the object-based and field-based which uses a complementary view of spatial information. Instead of assoicating attributes with individual spatial objects, it addresses the variation of the individual attributes across a spatial domain. This gives rise to the layer-based organization of data, where each layer represents the spatial variation of a specific attribute.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
spatial data	Unclassified--Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	1	

NASA Form 1626 Oct 86