National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

University of Alberta

A COGNITIVE MAP MODEL FOR GEOGRAPHIC INFORMATION SYSTEMS

by

Xiaoyou Zhou

A thesis
submitted to the Supervisory Committee
in partial fulfillment of the requirements for the degree
of Doctor of Philosophy

Department of Computing Science

Edmonton, Alberta
Spring 1994

ISBN 0-612-11433-3

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Xiaoyou Zhou

TITLE OF THESIS: A Cognitive Map Model for Geographic Information Systems

DEGREE FOR WHICH THIS THESIS WAS PRESENTED: Doctor of Philosophy

YEAR THIS DEGREE GRANTED: 1994

(Signed) .....

Permanent Address:
Apt#10, 10651-105 Street
Edmonton, Alberta, Canada T5H 2X1

Dated 30 March, 1994

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **A Cognitive Map Model for Geographic Information Systems** submitted by **Xiaoyou Zhou** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy.**

Wayne A. Davis
.....
W. A. Davis

.....
R. Goebel

.....
P. G. Sorenson

.....
S. Cabay

.....
A. E. Peterson

.....(chair only)
U. Maydell

.....
for T. K. Poiker

Date March 24, 1994

To My Wife, Zhenrong Duan

ABSTRACT

This thesis is concerned with the problems of storage, manipulation and retrieval of geographic data. The objective of this research is to introduce a cognitive map model for the representation of spatial data in geographic information systems (GISs). The salient feature of the proposed model is such that it is a result of the intergration of three areas: database management systems, artificial intelligence, and statistics. In this thesis, methods for knowledge representation, data management, uncertainty handling that traditionally belong to separate areas are applied in a complementary manner to enhance the capabilities of a GIS. This feature provides the proposed model with significant advancements in GIS technology in three areas: 1) the model provides a more expressive language for modeling various geographic concepts, including shape, location, place, complex objects, object interrelationships, topological relationships and metric relationships; 2) the model supports the searching of complex geographic data according to both attribute-based and location-based criteria in a flexible and effective manner; 3) the model also serves as an integrated approach to spatial data modeling that effectively handles data errors.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Wayne A. Davis, for his guidance through the research and preparation of this thesis. Throughout my stay at University of Alberta, he has served as my supervisor for both my Masters and Ph.D programs. His insightful comments on my Masters thesis inspired early ideas of this thesis. His profound knowledge in the field and his unfailing confidence in my motivation and capability allowed me to pursuit this challenging and demanding research with persistence and determination. His constant striving for creative and rigorous scholarship influenced many aspects of my professional life during this endeavor. For many years, Dr. Davis taught me virtually everything. He taught me how to separate the challenges that lead nowhere from those which lead somewhere and how to prevent feelings that would divert me from pursuing my research with usual optimism and zeal. He showed me the importance of moral principles - characteristics such as sincerity, honesty and straightforwardness in one's daily living patterns. He sacrificed many precious weekends improving my writing and public speaking skills. He often included me in many interesting activities which allowed me to learn different cultures. Today, I look back with a tear in my eye and gratitude in my heart. For all he has done for me, I cannot find enough to express my thanks.

A large portion of this research was done away from Alberta, and I would like to thank the members of my supervisory committee, Drs. P. Sorenson and R. Goebel, for making this possible. I especially appreciate their encouragement, advice and constructive criticism which helped me in dealing with all kinds of problems faced in both personal and professional life. I also would like to acknowledge the support of management of facilities at Acadia University, Wolfville, Nova Scotia, especially to Drs. L. Oliver, E. Eberbach and P. Proszynski for providing the flexible work schedule and unlimited access to all resources at their disposal. Many thanks also to Edith Drummond for patiently and expertly handling the many administrative problems and chores that go with a distant student.

I wish to thank Professors S. Cabay, A. E. Peterson, U. M. Maydell and my external examiner Dr. T. K. Poiker for serving on my examining committee. Their careful reading caught many mistakes in the thesis draft, both technical and stylistic in nature. Special thanks go to Professor U. M. Maydell for her time spent with me for tightening the prose and improving the format of my oral presentation.

I would also like to express my appreciation to the Department of Computing Science for technical and financial support.

Finally, I also would like to thank my wife, Zhenrong. Her never-failing encouragement and understanding swept aside all my worries and doubts. Without her, this work would surely have remained in progress. Moreover, the joyful moments I spent with my son, Michael have always been some of the most pleasant times I ever had.

Table of Contents

Figures

Chapter 1

Introduction

1.1 Multipurpose Environmental Database Systems (MEDS): A New Challenge

Effective management of our environment is a daunting task that we face as individuals, in government, and as a society. The inability to regulate our impact on the natural world could ultimately lead to economic and social, and ecological collapse. There are too many areas of the world where our failure to assess the impact of human activities has led to visible and perhaps irreparable damage. The horrific ecological problems of the former Soviet Union, the destruction of the world's tropical rain forest in Thailand, Indonesia and the Philippines present us with shocking examples of where we may be heading. In Canada, our western forests are under tremendous development pressures. The Eastern cod fishery is at the point of collapse. How are we to deal with these enormous problems?

Environmental and resource issues cannot be separated from social and political ones. The existing infrastructures of our governments typically reflect an institutionalization of past technologies and problem understanding. Today, departments and ministries exist to deal with forests and mines, and separate organizations to deal with the protection of the environment.

The arbitrary partitioning of our institutions is not, however, reflected in the external world. Nature is an integrated system. Everything depends in some manner on everything else. We cannot attempt to look at water quality without looking at the impact of commercial forestry or agriculture or industrial development. Therefore, we must come to grips with a system that is able to gather, manage and exploit disparate information resources, and to apply them to decision-making for natural resource, environmental and economic planning.

Over the past twenty years, two competing trends have developed: an increase in the requirements of spatial data applications and the advancement of geographic information system (GIS) technology. On the one hand, a growing number of federal and international agencies have been attempted to build very large, integrated geographic information systems to store different

types of spatial data as the basic analytical tool within their respective organizations [Abel92, Anto87, AM83, Baum88, Chang77, CK81, LM83, NS93, Oren86]. On the other hand, with the extension of the scope of GIS applications, spatial data has accumulated at an increasing rate. Not only has this expanded the range of available spatial data in digital format but also has necessitated new requirements for geographic information systems.

The following five aspects which are directly related to a Multipurpose Environmental Database System (MEDS) are identified:

(1) Voluminous: Spatial data files normally contain very large amounts of data which must be stored in secondary storage with aspects in main memory as needed. The World Digital Database for Environment Sciences (WDDES) [Bick88], the European CORINE project [BM88], and the World Soil and Terrain database (SOTER) [Baug88] are all examples of MEDSs. These databases normally contain millions of spatial objects compared to the thousands of objects commonly processed in current GIS systems.

(2) Multi-typed: The data contains many diverse data types, such as real-valued functions, vector-valued functions, categorical-valued functions, and functions taking general symbolic values.

(3) Multi-layered: Many layers of different data types, interrelated by complex (implicit) relationships, must be accommodated. A typical list of coverages might be: administrative base maps, soils, rainfall, wind velocity, length of growing season, hydrology, geology, mineral and petroleum deposits, vegetation cover, land use, terrain, population density, wildlife, parks and reserves, transportation and waterbody.

(4) Multi-dimensional nature: Spatial data is applicable to a multi-dimensional space.

(5) Uncertain nature: The data in digital format can be incomplete, imprecise and error-prone, and the number of possible spatial interrelationships is very large.

To meet all these characteristics satisfactorily, a MEDS must be equipped with the following features:

Knowledge representation. The representation must accommodate both spatial and aspatial attributes. In addition, it should be able to capture the uncertainty feature of geographic data, and to efficiently support various operations. A systematic method must also be found to encode the model in terms of a data structure.

Efficient Retrieve: An efficient search scheme must be found that can extract the necessary information from the presentation according to both spatial and aspatial criteria.

Reasoning: A specification of the reasoning mechanism that is used to derive implicit relationships recorded in the knowledge base. This is the main differentiating characteristics of a knowledge base system from a database system. Reasoning may be by means of *deduction* and *abduction* techniques.

Assimilation (Learning): Given an accurate representation of some geographic knowledge, and an accurate geographic fact or description, the representation must be modifiable to include new facts or descriptions.

1.2 Existing Problems

Current GIS technology suffers from a lack of versatility both for individual systems to accommodate a broader range of applications, as well as for the incorporation of differing types of spatial data from a variety of sources [Abler87, Egen92, Oren86, Peuq84].

It is appropriate to view the problems of current GIS technology from the point of view of a data modeling process. The concept of spatial data can be viewed from a number of levels. These levels progress from reality, through an abstract user-oriented information structure, to a machine-oriented storage structure. There are four realms of interest in the philosophy of information. These realms are: reality, cognitive model, data model, and structure model [Davis86, Peuq84]: **Reality,** the real world, includes all aspects of actual objects that may or may not be perceived by individuals. **Cognitive model** is a model developed as the result of a process in which individuals selectively perceive and mentally construct their conception of the real world

of living beings, objects, places, events, and their surrogates. This model is assumed to be partly subjective since it depends not only on reality, but also on the observer's perception and knowledge of the real world. **Data model** is an abstraction of the cognitive model which incorporates only those properties thought to be relevant to the applications at hand. In general, a data model has four components [Ullman80]: 1) a data space - a collection of elements and relationships among the elements, 2) a collection of type definition constraints to be imposed on the data space, 3) a set of manipulation operators supporting the creation, deletion, and modification of elements, and 4) a language used to identify and select elements from the database. A data model forms a *physical model* ready for communication via a certain *encoding schemes* like sounds, drawings, and text. **Structure model** is the recording of data in computer code in terms of data structures and file structures.

The last three views of data correspond to the major steps involved in database design and implementation. The overall process is one of progressively refining the users' view of data. At a high level of abstraction, the data model provides users with a format of data that is easier to understand and deal with, leaving out significant details. Whereas, at a lower level of abstraction, the structure model provides the actual representation of the data.

Historically, geographic information systems and database management systems were two separate areas. On the one hand, current state-of-the-art database technology has been developed with the primary purpose of supporting business applications. Consequently, the technology addresses the requirements of hardly any geographic information applications. On the other hand, throughout the development of GISs, the subject of an efficient structural model has been a central topic. While progress has been made in this direction, little attention has been paid to the development of an adequate data model [EF89, Eg92]. It has been realized [AM83, Baum88, Gupta 91, Hass91, MO87, More85] that database management is an essential component of a GIS. To date, most GIS projects have been focused on combining existing database technology with GISs.

Geographic information systems have relegated database technology to a paradigm of spa-

tial data processing. The main problem brought about by the reorientation is the lack of systematic and accepted data models for spatial feature modeling and processing. Another striking result of this reorientation has been an increased interest in the interrelated issues of geo-data uncertainty, geometric and attribute error, database consistency and geographic generalization [Abler87, Abel92, Good92b]. Today, the characteristics of spatial data in a typical MEDS have presented a challenge to GIS developers in their search for suitable data models. To be successful, a new generation of GIS is expected to emerge.

1.3 Cognitive Map Model for GIS

To respond to this challenge, this thesis develops a novel data model based on the notation of a *cognitive map*.

In the area of cognitive perception, the terminology "cognitive map" and "cognitive model" are used interchangeably [Hart73, Kuip78]. Cognitive map was originally proposed as a technical term in neurophysiology, referring to a particular brain structure (often located in the hippocampus). A comparable definition of cognitive map was given by Downs and Stea [DS73], who stated that a cognitive map is the product of a series of psychological transformations by which an individual acquires, codes, stores, recalls, and decodes information about relative locations and attributes related to phenomena in a spatial environment.

With the development of computer technology, the term "cognitive map" has been used in a more general sense, and the information associated with a cognitive map differs subtly but significantly from that of a cognitive model. For many years, a cognitive map has been the subject of study from many different points of view under the context of artificial intelligence, computer vision, and geographic information systems [Davis86, MF89, NS93]. In these areas, the term "cognitive map" is actually referred to as a physical data model. Nevertheless, as the word "cognitive" implies, the most basic characteristics of a cognitive map is that it is an abstraction of reality at a level closer to the cognitive model.

The distinct feature of this new model is that it involves integrated techniques from different

areas (i.e., AI, DBMS, statistics, etc.). More importantly, many of the techniques can be applied in a complementary manner such that the resulting combination would yield a system with data volume capacities and data processing capabilities well beyond those which can be provided by existing GIS approaches.

1.4 Outline of the Thesis

This thesis is organized in seven chapters. The first chapter provides a general introduction to the nature of a cognitive map. The second chapter provides a comprehensive survey on the current state-of-the-art GIS technologies followed by the analysis of shortcomings of the various systems. Chapter 3 presents a cognitive map model for GIS. This model intended to provide a unified framework for spatial data modelings in GIS. Chapter 4 presents the corresponding query languages of proposed model. Chapter 5 discusses some issues of database implementation of the model. Chapter 6 analyzes the performance of the new model. Chapter 7 summarizes the main results of the proposed model and discusses problems for future research. left to be done. Finally, to highlight the underlying concepts of a cognitive map model for GISs, the problem of system analysis and design of a MEDS will be used as a running example throughout the thesis.

Chapter 2

Review of Previous Systems

During the last two decades, geographic information systems have been used to systematically collect geographic map information for digital data processing on both international and national levels. The development of geographic information systems can be divided into three stages: file systems, database systems, and knowledge based geographic information systems.

2.1 File Management for GIS (FMGIS)

Early GISs consisted of a catalog of files, containing either vector or tessellated data. Since the information content of large area geographic maps are very high and the resulting data files are extremely voluminous, the use of adequate data structures is a major concern in these systems.

The Canada Geographic Information System (CGIS) [CM89], which originated in the 1969's, was the first significant development is GIS for the storage and analysis of resource related thematic data as well as production of maps. The main data division of CGIS is the theme (coverage) which is designed to cover all of Canada. The data structure for each theme has two components: the Image Data Set (IDS) and the Descriptor Data Set (DDS). The former stores spatial information for polygons while the later encodes attribute data for each polygon. The IDS consists of a collection of *frames*, which are squares in the geodetic coordinate system. Larger frames are constructed by hierarchically combining groups of four smaller frames in a manner similar to quadtrees. Moreover, within each IDS frame, a vector structure is used to encode entities such as polygons, chains, and points. Therefore, the IDS is a hybrid model of vector and raster structures. On the other hand, attribute data for polygons, including thematic attribute, area and centroid, and a list of the IDS frames which it intersects, is stored separately in the DDS. The DDS, however, is ordered by polygon number which implies that CGIS requires an exhaustive search for attribute-based information.

Another typical example in this category is referred to in [Cook78], where Cook described a GIS closely resembling CGIS. The main concept of Cook's structure is a quadtree. To be specific,

Cook's structure uses fixed capacity tiles, called *buckets*. Bucket overflow is handled by subdivision similar to a quadtree. A significant difference between Cook's system and CGIS is that the former efficiently supports the merge of multiple themes within one structure whereas CGIS uses one structure per theme.

2.2 Database Management for GIS (DBMGIS)

In DBMGISs, vector and/or raster primitives of the spatial data domain are represented by database records (network), segments (hierarchical) or tuples (relational). Additional operators (i.e., not usually found in querying languages) are added to provide basic spatial capabilities such as computation of distance [AM83, LM83].

ARC/INFO is a product by the Environmental Systems Research Institute (ESRI) [AM83]. Like CGIS, both raster and vector structures exist in ARC/INFO. However, in ARC/INFO, thematic data is stored in the form of feature attribute tables and managed by a relational database management system. ARC/INFO provides management software called LIBRARIAN which can be used to organize spatial data, by both attributes and locations, into a map library. Attribute information associated with spatial data may be extracted, using the *Selection* operation, i.e, specifying the attribute, whereas the *Thematic* overlap, the merge of coverages, is achieved by means of the *Join* operation. LIBRARIAN uses a collection of *tiles* to partition the area of interest. Tiles may be of any shape. A spatial index, called the *index coverage*, is maintained for access to the tile structure. The *Windowing* operation is performed by defining a window as a polygon in selection coverage. LIBRARIAN facilitates windowing on a collection of tiles in which the window may span one or more tiles.

Many query languages for classical models have been developed. There are two broad classes of query languages: 1) algebraic languages, such as Information System Base Language (ISBL) [Ullm80], and 2) predicate calculus languages, e.g., SQUARE and SQL for the system R DBMS, QUEL for INGRES, and Query-by-Example (QBE) [Motro90]. The calculus is favoured for user specification for queries since the user merely specifies the criteria the query result must satisfy. An algebraic expression is the opposite in that it specifies how the result is obtained by

identifying an ordering of data manipulation operations.

There are many other GISs [Baum88, Bick88, Bun87, CK81] that use the traditional database model as their implementation paradigm. The differences between them depend on several factors such: as how the structure model is implemented in terms of vector and/or raster data, the choice of file scheme such as a B-tree or hashing to store data items, and the way in which spatial operations are associated with the underlying structure models.

It is well known [Alba90, Ban88, LM83, WH87] that classical database management systems have some difficulty supporting nontraditional applications such as engineering applications for CAD/CAM data, geographical data, and images. Thus, computer scientists have been developing several new models.

In one direction, research on the extension of traditional data models to accommodate complex objects has occurred. The simplest extension, such as reported in [Bun87], constructs a set of pre-defined spatial operators in the relational DBMS. For instance, a QBE-like language can be employed to handle nonspatial parts of queries while a set of built-in spatial operators handles the spatial aspects. A more general approach provides a DBMS that can be customized by the addition of abstract data types (ADT). INGRES [Ghosh88] has also been extended to allow the addition of ADTs. In addition, support for a new access method is provided in AgriDB [Gadia93], Ariel [Haas91], and Pardes [Etz93]. Finally, relational extensions of the standard relational model, such as RT/M and the nested model, try to strike a balance between generalizing data modeling and the data manipulation parts of the relational model.

Many extended SQLs with various features have been reported in the literature: CQL++ [Jaga93] is an SQL-like language designed to define, access and manipulate a C++ based extended relational database system; Cymbal [Greer92] is a multi-paradigm language which is able to synthesize procedural commands with the declarative constructs of symbolic logic, set theory, and SQL; ParaSQL [Gadia93] is also an SQL like query language for parametric data, of which spatial, temporal, spatio-temporal, belief and ordinary data are special cases. For other languages in this class refer to [Gupta 91, Lam91].

Despite continuous efforts to improve the SQL standard (ANSI 1991), and numerous attempts to extend SQL [Egen92, Gadia93], assumptions that limit generality are hard-wired into traditional DBMSs. As the result, the SQL-based languages are less than adequate in dealing with complex objects. First, since these languages rely on somewhat artificial joins to decompose one request into a number of lower-level ones, and re-link the tuples in the application program data space, considerable data transformation and data construction/decomposition must occur in data transfers involving the underlying DBMS. This causes the so called "impedance mismatch" problem [Ban88, Kifer89, Loric91]. Secondly, the ADT approach described above gives little support to users that need to manipulate spatial objects with ease (i.e., finding spatial relationships of objects of a plane, creating a new class of objects from existing objects based on a combination of their spatial and nonspatial properties [Oren86]). Finally, the SQL-based query facilities lack the ability to take a more active role in deducing relationships rather than being just a passive repository of data [Egen92]. Although classical query languages can be extended to incorporate more real world knowledge, the main weakness is the reliance on a given conceptual schema and the focus on syntactic aspects. The conceptual design process and the semantics being modeled are largely ignored. For example, an answer to a query about a label on a map may be "This is the mileage along Highway #1 from Edmonton to Calgary", and the query " What are possible soil classifications?" refers to the domain of an attribute, soil type. These queries belong to a class of queries called *meta-data queries*. Another important kind of query is called a *knowledge query*. Knowledge queries refer to queries that explain the reasoning process that underlies a particular query language. Spatial relationships are typically derived from representation in a spatial data model, rather than being explicitly recorded. The derivation is based on rules that formalize the criteria for the individual relationships. Spatial information systems that allow users to tailor their spatial predicates must also include facilities to let them inquire about the rules used. For instance, the adjacency between two parcels may be defined such that they share at least one common boundary, but have no common interior. Users may want to know "Why were the two objects identified as neighbors?". Such knowledge queries are of particular interest when users

modifying the database want to obtain information before making a change.

2.3 Knowledge Based Models for GIS (KBGIS)

A number of knowledge based models have also been described in the literature. These can be divided into three categories: 1) Prolog-like deductive models, 2) object-oriented models, and 3) approximate models. All these models carry out some elements of the cognitive map.

2.3.1 Prolog-like Deductive Models

The systems in this category are all variations based on the same theme of coupling logic programming languages with traditional database management systems. In such designs, logic is used to obtain a proper foundation for modeling database semantics, and factual data is stored as a database and maintained by a DBMS. Logic programming languages serve as an interface between users and the database. Knowledge about the database schema is represented as a set of clauses. Rules used in a query, but not defined in Prolog, will be retrieved from the database automatically. Among others, some of the important *logic based models* are Datalog, LDM, LDL [KW89], and Prolog-INGRES [Ghosh88]. Deductive database models have gained acceptance in GIS applications. The following shows some GIS projects in this category.

ORBI [Pere82] is an example of a early knowledge based geographic information system. It was developed to keep track of environmental resources of Portugal. ORBI consists of four modules: 1) a natural language parser for Portuguese, 2) a menu handler for fixed-format input, 3) a deduction facility that uses Prolog for geographic reasoning, and 4) an explanation facility that explains what is in the database, the kind of deductions that are possible, and what kinds of vocabulary and syntax may be used.

LOBSTER [Frank84], like ORBI, is based on a logic programming paradigm. It serves as an intelligent user interface to a spatial database system, using a network model. LOBSTER demonstrates the significant impact on the flexibilities gained by using a Prolog-like language in building the user interface.

Wang shows [Wang90] that much geographical data can be generalized and formalized into

meaningful conceptual graphs which can then be included in a Prolog paradigm. Wang's results show that deductive database technology can be gracefully tailored in a GIS.

2.3.2 Approximate Models

The salient feature of the systems in this category is that they use concepts from either fuzzy logic or statistical theory to explicitly represent and manipulate spatial uncertainty. Key work in this category has been carried out by D. McDermott [McD78], E. Davis [Davis86], and Goodchild [Good92].

D. McDermott and E. Davis developed the SPAM [McD78] program as a autonomous module of an arbitrary geographic reasoning system. The system consists of three modules: an assimilator, which assimilates a sequence of geographic facts into a cognitive map; a quantity retriever, which answers user queries about specific objects using the information in the cognitive map; and an object retriever, which enumerates all objects with specified properties.

SPAM made three contributions in dealing with spatial uncertainties. First, the use of an object coordinate reference system for positional representation. For this purpose, each object is associated with a frame of reference by three parameters: origin, scale, and orientation. Furthermore, a pair of frame references is related by three relations: the coordinates in one frame of reference for the origin of another frame; the difference between the coordinates of the two frames; and the ratio between the scales of the two frames. Secondly, shapes in SPAM are described in terms of hierarchies of "circyls". Circyls are characterized by length, cylinder radius, and endcap radii. This provides a whole spectrum of object approximation from coarse to fine. Finally, fuzzy ranges are used to capture the uncertainty in the model. Figure 2.1 shows a map that records the following facts: "the coordinates of the origin of object A in the reference frame of B lie in [2.5, 3.5], [1.0, 2.5]"; "the orientation of A is between 30^o and 60^o counterclockwise from the orientation of B"; and the scale of the reference frame of A is between 0.5 and 1.0 times that of B".

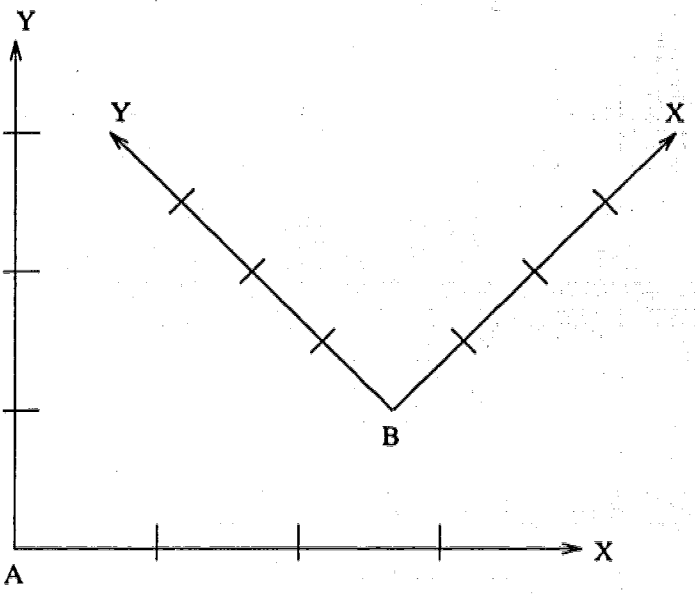


Figure 2.1 Relationship between Two Object Coordinates in SPAM

Figure 2.2 shows a hierarchical representation of a human arm.

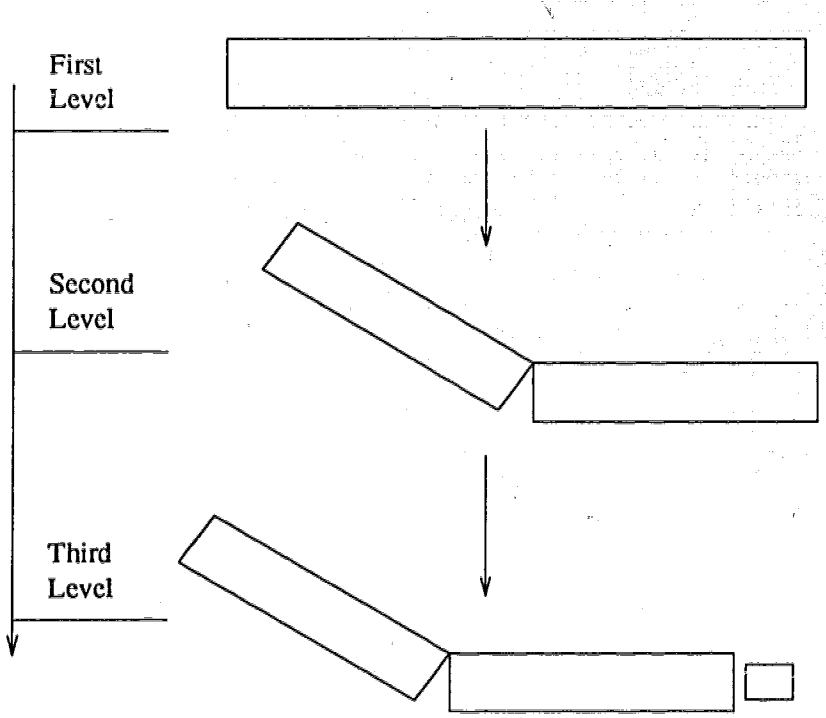
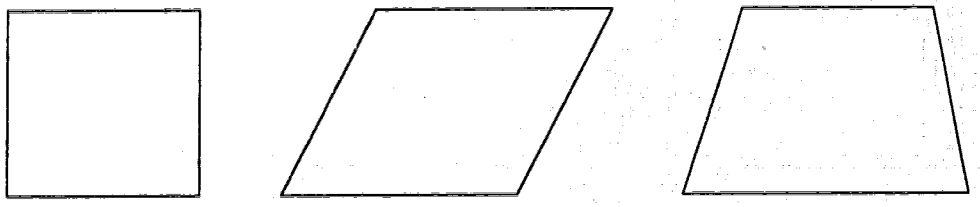


Figure 2.2 Hierarchical Representation of a Human Arm in SPAM

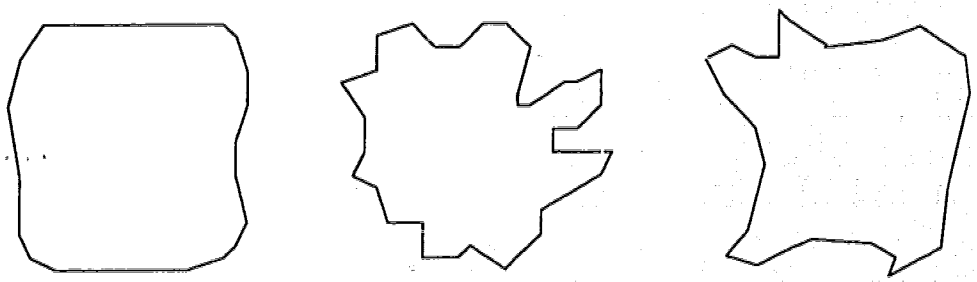
Assimilation is done in one of two ways: fuzzy restriction and remapping. Fuzzy restriction results in new and tighter bounds on one or more of the fuzzy ranges, as the new information

arrives. If fuzzy restriction fails, then the system must restructure new frames of reference using remapping.

E. Davis developed MERCATOR as an extension of SPAM [Davis86]. In comparison with SPAM, MERCATOR's representation is more expressive, more flexible, and richer in semantics. For example, instead of using a raster structure as in SPAM, MERCATOR adopts a vector model in its implementation. Consequently, almost all geographic reasonings involve object boundaries rather than their interiors. Boundaries are much easier to deal with and reasoning about boundaries proves to be adequate for a large number of inferences in cognitive maps [Davis86]. Furthermore, using a vector structure model, MERCATOR is able to give formal semantics of objects in cognitive maps [Davis86]. Finally, MERCATOR uses both fuzzy range and grain-size in measuring uncertainties: fuzzy range measures uncertainty of dimensions and grain-size measures uncertainty of shape, see Figure 2.3.



b) Shape Variation Allowed by Fuzzy Measurements



a) Shape Variation Allowed by Grain-size

Figure 2.3 Fuzzy Range and Grain Size for Shape Modeling

On the other hand, SPAM has a number of good features which were not included in MERCATOR such as the use of a frame of reference in object representation, the ability to reorganize the cognitive map in order to group objects found to be physically close, and to infer and answer queries for an arbitrary domain.

Numerous methods for uncertainty reduction and absorption have been developed, however, a systematic survey in this area is beyond the scope of this thesis.

2.3.3 Object-oriented Models

The salient features of an object-oriented model [Ban88, Jaga93] can be summarized as follows:

- The notion of complex objects: Spatial data in a MEDS is characterized by the complexity of the objects or entities that are modeled. A simple example to illustrate this point is to consider an application in the cartographic domain where entities such as districts, towns, roads, and buildings are modeled. A town is a highly structured object that may consist of networks of roads and railways represented as linear features, and parks and buildings represented as areal features.
- Encapsulation: Encapsulation is the clear separation between the external behavioral semantics of objects and the internal implementation. This black box specification is defined in terms of the behavior of the operations. An operation is performed by a method. A GIS that incorporates encapsulation provides flexibility in replacing existing methods with advanced methods - as long as the external interface remains the same, any code should continue to work.
- Polymorphism: This refers to the ability to adapt the same method to objects from different classes. Take a domain of composite objects for example - a water-body which has component objects such as: lake, river, channel, reservoir, and so forth. It is necessary to be able to search spatially for many different types of objects based on location. The query "find the polluted water body closest to a specific site" requires that the computation of spatial dis-

tance between a location (the site) to different object classes (i.e., curves and polygons). Both use the every same method.

- Identity: An object identity provides a way of denoting the object independently of its behavior or state.
- Type: Typing defines the protocol of a group of similar objects which are instances of type. It also provides means of defining semantic integrity constraints for a database.
- Inheritance: Objects are grouped into classes and objects of the same class have common operations. Moreover, these objects are organized into a hierarchy. Inheritance is a powerful modeling tool enabling capture of relationships between objects and factoring out shared specifications and implementations in applications.

There are many object-oriented systems in GISs include TIGRIS [Marx86], PROBE [MO87], GemStone [Cam90], and KBGIS [SP84]. Among others, of particular interest, the Knowledge Based Geographic Information System (KBGIS) will be reviewed.

KBGIS [SP84] is a prototype system designed to answer queries about spatial objects stored in implicit form in large multilayered spatial databases. KBGIS is based upon a pair of hierarchical data structures, one spatially-oriented and the other object-oriented. First of all, data is organized spatially in the form of a "forest" of quadtrees, i.e., each separate theme is arranged in a separate quadtree, and all quadtrees are spatially registed. Secondly, a taxonomy of data objects is recorded by an and-or tree structure.

The names and properties of classes of data objects are stored at each node in the and-or tree. There are two kinds of links. The father-son links shown graphically with a cross-link bar indicates "and" relations. For example, "eroding orchard" in Figure 2.4 is composed of the combined set of "orchard" and "slope". The father-son links without a cross-link bar, as shown graphically in Figure 2.4, are "or" relations. This indicates, for example, that either a mature or an immature orchard can be classed in a more general sense as an "orchard".

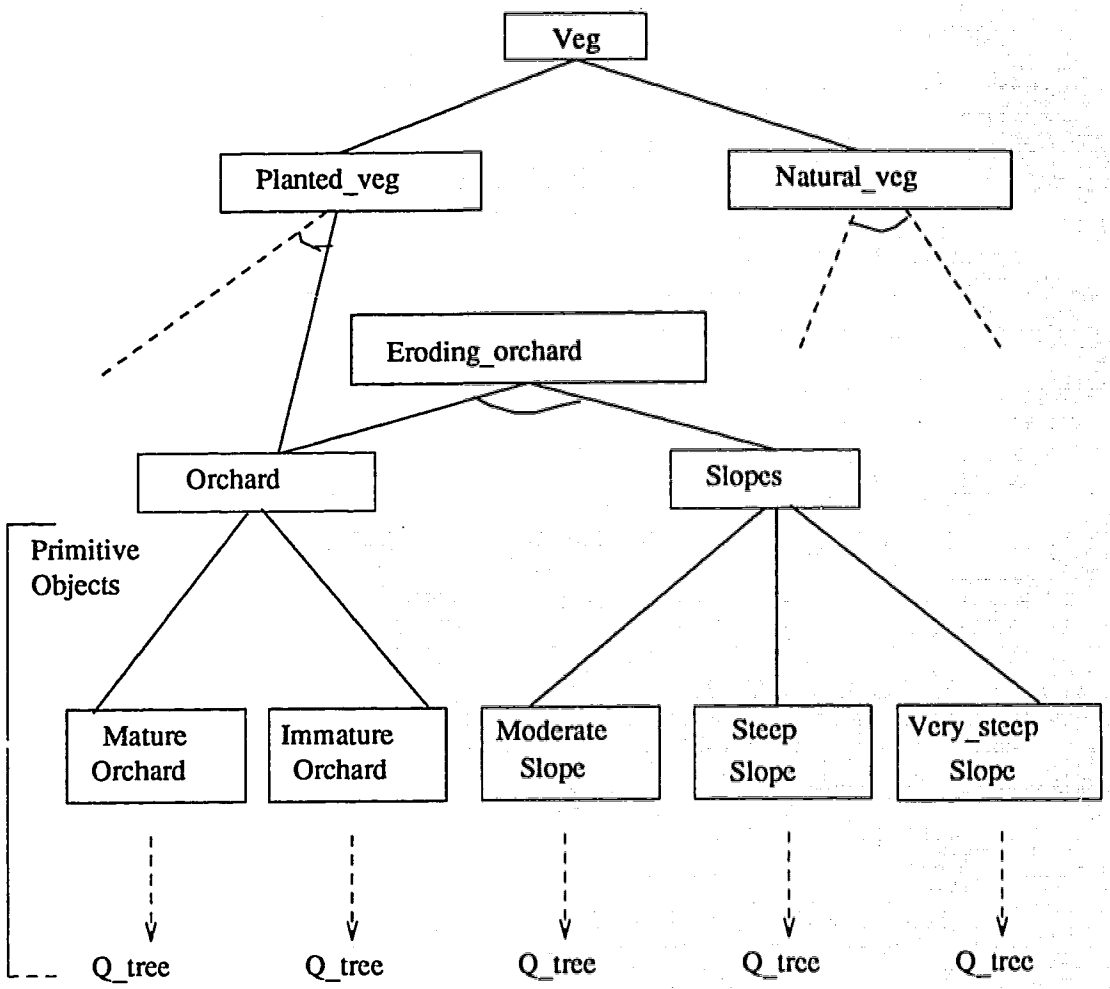


Figure 2.4 A Sample Object Tree

Although KBGIS is an early object-oriented GIS, it includes the most salient features of the object-oriented paradigm. More importantly, to improve the performance of various spatial queries, the system features some spatially-oriented components. For this reason, a large research project based on KBGIS is currently under development at the National Science and Space Centre in the United States [Cam90].

2.3.4 Problems with KBGIS

It is widely recognized [Allen90, Egen88, Frank91, Gadia93] that the current state-of-the-art KBGIS technology exhibits a number of deficiencies in handling spatial data. The underlying cause of these current shortcomings is that current systems do not lend themselves conveniently

or satisfactorily to take into account all of the characteristics of spatial data. For example, both deductive database schemes and approximate schemes normally use a flat data model and do not support the notion of complex object and data abstraction. On the other hand, despite many appealing advantages, an object-oriented paradigm is not, however, without limitations. The following problems are identified.

Object Orientation Vs. Spatial Orientation

In reality, geographic objects are complex phenomena which typically contain both spatial and aspatial features. For instance, many complex classes of spatial objects can be decomposed into smaller ones, using two very common and useful properties of objects. Both of these properties concern relationships between objects: ISA relationship ($\leq_a$) - the relationship between objects in a hierarchical taxonomy. For example, "dog ISA pet, pet ISA animal, animal ISA livingthing, etc.". ISPART relationship ($\leq_s$) - the relationship between objects that are made up of a set of components, each of which is made up of a set of components, and so forth. For example, "Canada ISPART Earth, Alberta ISPART Canada, Edmonton ISPART Alberta, etc.".

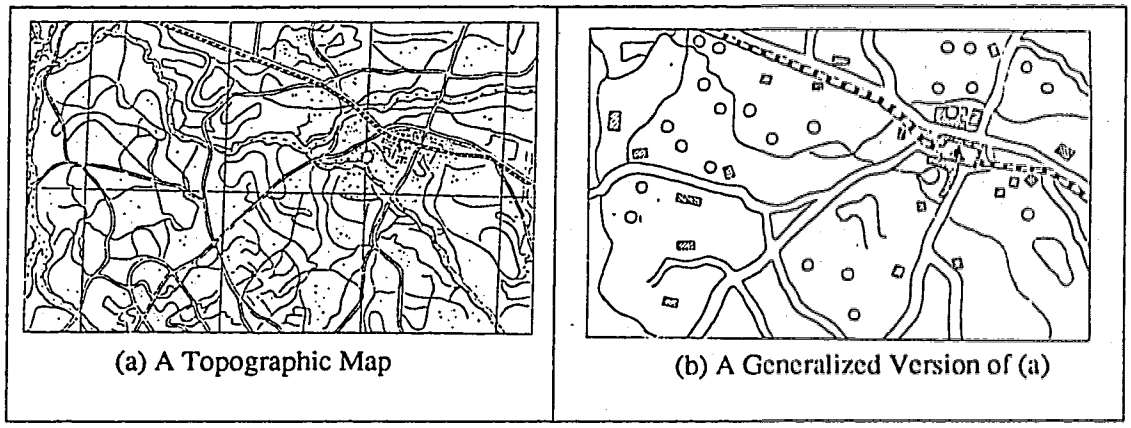
In essence, a spatial database must provide a method of accessing data based on geographical location. The spatial access method is somewhat distinct, as it permits operations to be defined that are not included in standard object-oriented systems. For example, it is possible to retrieve and to aggregate geographical information associated with an arbitrary, user-defined polygon, an operation that is not defined in standard object-oriented systems. A spatially-oriented system must also address the issues of spatial dependence, the propensity for nearby locations to influence each other and to posses similar attributes. Spatial dependence calls for better methods for dealing with the world as a set of overlapping continua, instead of forcing the world into the mold of rigidly bounded objects.

Therefore, the whole issue is not just a matter of introducing two different kinds of "inheritance" links to the model ($\leq_a$ and $\leq_s$). In a sense the problem lies much deeper. In order to achieve both inferential and acquisitional efficiencies, a GIS must provide support for both object-oriented and spatially-oriented views of an object and still retain the vital fact that they are

all representations of the same object.

Data Abstraction

Data abstraction has two aspects: attribute generalization and geometric generalization. In object-oriented paradigms, the issue of geometric generalization is largely ignored. To understand the problem, look at a typical topographic map and a generalized version, see Figure 2.5.



(a) A Topographic Map

(b) A Generalized Version of (a)

Figure 2.5 An Example of Geometric Generalization

All kinds of transformations have taken place. Rail and main roads have been widened, streets and smaller roads have been eliminated; individual features have been disappeared or have been regrouped into geometric square sided symbol simulating built up areas, a few landmarks are retained and are highly symbolized; some selected land use areas are circumscribed by polygons and enhanced whereas others are simply denoted by a graphical symbol, etc. Most striking is the change in ratio between open space and space utilized by humans through roads and built up areas. Open space shrinks, whereas, human space increases significantly, due to the considerable exaggeration of graphic representation of roads, farms and settlements.

Geometric generalization is a key aspect of the mapping process, cartographic generalization. Without such a tool, a model can present a serious bottleneck in query processings since spatial search must be performed at low description levels.

The Use of Statistics

Statistic modeling is one of the most important components of geographical science. In common usage statistics often means 'facts and figures'. For example, health statistics or education statistics refers to a branch of applied mathematics concerned with the interpretation of numerical information. As far as professional geographers are concerned, the following three types of functions are essential [Good92, NS93]: **Description** - Descriptive statistics provides three types of characteristics, namely, *central tendency, dispersion,* and *the shape of the frequency,* to summarize and measure large sets of data. **Inference** - most geographic information systems have to deal with data obtained from samples, rather than with all the data about a particular situation. Inferential statistics enable the system, within certain strictly defined limits, to make statements about characteristics of a population based only on data collected from a sample. **Significance** - refers to a set of tests (i.e., the Kolmogorov-Smirnov test, student's T test, etc.) that decides whether an observed difference or relationship between two sets of sample data is significant.

GISs must come to terms with the 'information explosion'. The amount of information available, particularly numerical data, is accelerating [Baum88, Frank91]. If a user is to make use of this mass of information ways are needed to summarize information to get a concise measure of their characteristics. Both descriptive statistics and object-oriented formalism can help to fulfill this aim. The question is how these two components can be combined to yield a class of powerful, easy-to-use, and efficient data aggregation features.

Another discrepancy of object-oriented models is that they miss a mechanism for uncertainty handling that is a necessary mechanism for geographic data modeling applications. As a result, the knowledge about representation of uncertainty information, methods for combining uncertainty information, and the drawing of inference using uncertainty information is often embedded in the application program. This is analogous to the "impedance mismatch" problem in conventional database systems.

Chapter 3

From Object-oriented Model to Cognitive Map Model

Previous discussion reveals that a GIS requires that features of databases, artificial intelligence and statistics be combined in a coherent, consistent fashion for spatial data modeling. It is the aim of this thesis to develop the framework which simultaneously addresses all these areas by employing a highly flexible and robust model for spatial data representation.

In this chapter, a new model for GIS is developed. Section 3.1 introduces some basic components of the new model. Section 3.2 introduces a four level semantic hierarchy for complex geographic data modeling. Sections 3.3 and 3.4 define some data dependency relationships and database schema of the new model. Finally, Section 3.5 summarizes the results.

3.1 System Analysis and Design for MEDSs

We begin by defining what is meant by spatial data in general and then describe the type of data that this thesis focuses on: thematic data. Spatial data can be defined as any data concerning phenomena spatially distributed in one or more dimensions [Peuq84]. This is a large class of data and is often referred to as pictorial data (e.g., [CK81]). This thesis is concerned with spatial data pertaining to the surface of the Earth which is commonly represented by two-dimensional models known as thematic data.

There are two important components of thematic data: *location data* and *attribute data*. Location data includes *points*, *curves*, and *regions*. A point is defined as a coordinate, a curve is defined as a sequence of points, and a region is defined as a polygon with finite number of holes.

Attribute data, on the other hand, is the non-positional, descriptive information associated with location data on thematic maps. In a MEDS domain, location data and attribute data are typically interrelated by complex (implicit) relations (i.e., an entity can have several attributes associated with it and vice versa). Nevertheless, the simplest correspondences between location data and attribute data are one-to-one relationships. Among others, but of particular interest, the term *thematic coverage* is used to describe some qualitative or quantitative phenomenon (theme)

across an area of interest. Generally, thematic coverage partitions an area of interest into a collection of regions with each region corresponding to an attribute value.

Thematic coverage must deal with positional issues and attribute issues at the same time. Therefore, an important consideration is which component takes logical precedence: the spatial or the attribute. In the first case, the positional description of the domain objects precedes any attributes assigned. The resulting map is called a *socioeconomic coverage*. Examples of socioeconomic coverage include administrative base maps and census division coverages. In the second case, some system classification logically precedes the map. The map results from assigning each portion of an area into a class. The resulting map is called a *categorical coverage*. Examples of categorical coverage include: vegetation cover, soils, rainfall, etc.

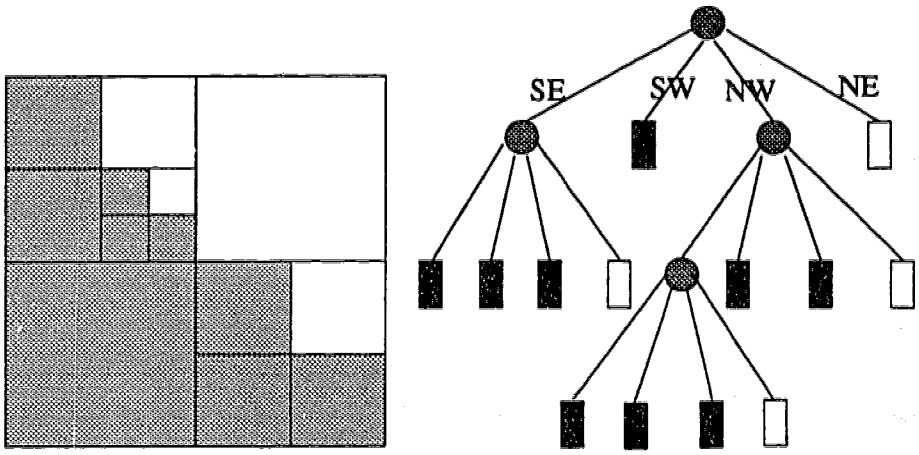
In the following, the issues of thematic data representation and manipulation under the context of a MEDB will be addressed. For this purpose, the system analysis process is divided into four levels. These levels progress from a simple thematic feature to a whole host of complex spatial objects and spatial relations. At each level, the corresponding semantics for thematic data representation and manipulation will be developed.

3.1.1 Primitive Thematic Data Representations and Manipulations

One important model, called the *raster data model*, for thematic data representation is based on the idea of subdivision or tessellation of the plane into cells. Each cell in a tessellation represents and defines a region that can be differentiated from an adjacent region (i.e., each cell identifies a location on the plane). A common term for the smallest cell of a tessellation is *pixel*.

The term quadtree has come to mean a class of nested tessellation models whose common property is that they are based on the principle of recursive decomposition of space [Samet90]. Quadtree structures are differentiated on the basis of the type of data represented and on the principle guiding the decomposition process. Quadtrees have been proposed for the representation of point, curve and region data. The most studied quadtree approach to region representation is the *region quadtree*, which is based on the successive decomposition of the image space into four

quadrants of equal size. Quadrant subdivision continues until homogeneous subquadrants are obtained. In the case of a binary image, all the leaf nodes are either BLACK or WHITE and all interior nodes are said to be GRAY. An example of a region quadtree is found in Figure 3.1.



(a) A Region (b) the Quadtree Representation of (a)

Figure 3.1 An Example of Quadtree Region Representation

Other raster data structures include Field trees [Frank90], Run Length Encoding [Van88], Cell Methods [Samet90a] and Grid Files [DH86, See91]. The primary differences among structures are the shape of the cell used, whether the tessellation is regular or irregular, whether the tessellation is flat or nested, and whether the structure is organized in main memory or in external memory.

Data manipulations at the primitive level can be divided into two categories: *location based queries* and *attribute based queries*. Examples of location based queries include set operations, thematic overlay and windowing. In Figure 3.2, these operations are illustrated using the thematic coverages *c1* (given in Figure 3.2.a), *c2* (given in Figure 3.2.b), and the window *w*. Figures 3.2d,e,f give the results of the intersection, union, and difference operations, respectively, involving A from *c*1 and C from *c*2, Figure 3.2.g shows the results of the thematic overlay between *c*1 and *c*2, and Figure 3.2.h depicts the results of extracting the window *w* from the thematic coverage *c*1.

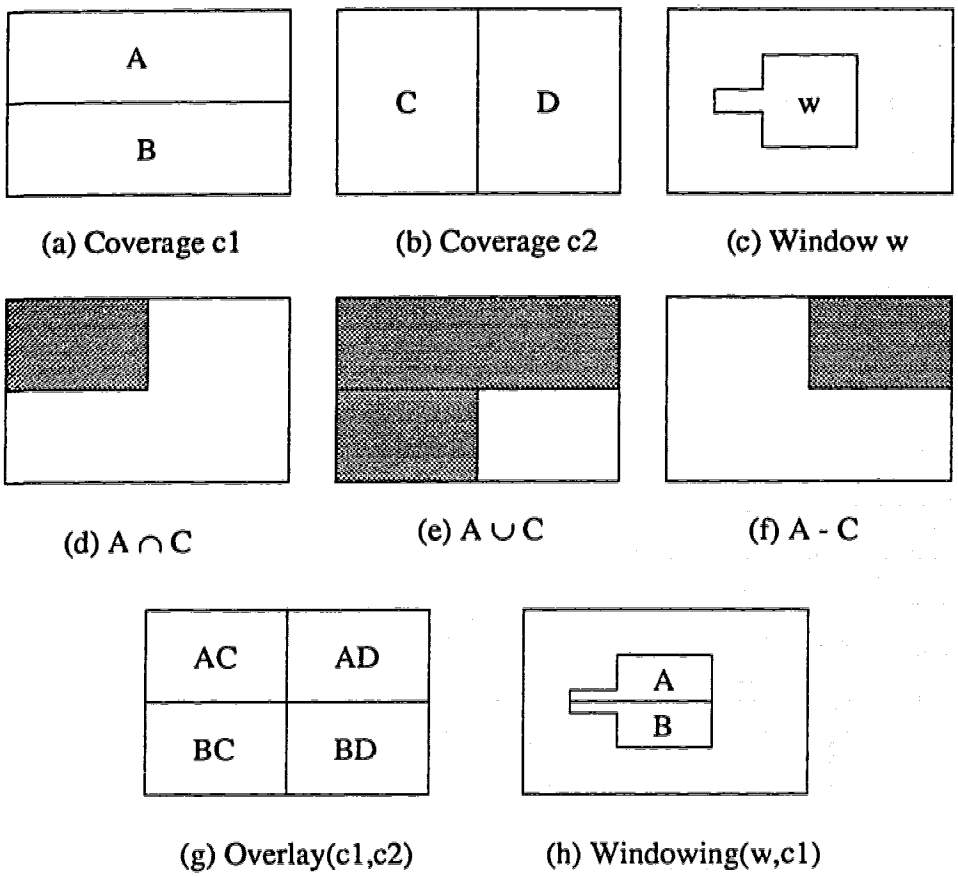


Figure 3.2 Examples of Location Based Operations

Attribute-based operations are characterized by a search for specific attribute values within a thematic coverage. Queries such as "find all flood plains on the map" belong to this class.

It turns out that raster data structures are well-suited to location based operations, as well as for determining various other region properties [DX86, Maff87, Samet90]. To facilitate attribute based operations, it is necessary to store attribute values as well. The first method associates attribute values with every cell. For instance, to represent different attribute values of a region, the leaf nodes of a quadtree contain more than two colors. The problem with this approach is that cells of the structure are not clustered according to attribute values. As the result, the whole structure is just a miscellaneous collection of the primitives from the objects in the study area [Maff87,Zhou88]. Therefore, brute force searches are unavoidable.

One way to overcome this problem is to list, for each attribute value, all of the cells of that

value. This method constitutes an inverted list of the structure. In this way, cells can be sorted and clustered according to attribute values. At first glance, this method appears to be an obvious improvement over the first method. However, one cannot achieve such an improvement without giving up something: the inverted list consumes considerable storage since all locations must be explicitly recorded. For this reason, the inverted list method is not used in practice.

To save additional storage space, a common method is to store object boundaries rather than their interiors. This method leads to a class of data structures called *vector data structures*. The distinct feature to all vector data structures is the representation of points by 2-d coordinates. Curves are represented by sequences of points and regions are constructed by considering closed sequences of points. This class of data structures are very compact in terms of storage. The disadvantage is the difficulty of performing location-based operations such as intersection and union for regions [DX86, Maff87, Van88].

The tradeoffs between raster structures and vector structures have long been the motivating force for the development of data structures that attempt to capitalize on the advantages of each model while minimizing the disadvantages.

The third model is a hybrid of the location-based and attribute-based model [VC88, CV90]. A hybrid model consists of a tessellation structure which subdivides a vector encoded thematic data set into a set of adjacent cells. The tessellation components serves both as a partition of and as an index to the vector data within each cell. A very important characteristic of the hybrid model is the resolution threshold that governs the quadtree decomposition. The type of threshold and its particular value determine several parameters describing the resulting data structure. These parameters are the size and depth of the tessellation component as well as the tile capacity of the vector component (i.e., the amount of data in each tile). These parameters directly affect the efficiency of the hybrid model for performing location-based and attribute-based operations. The strength of the hybrid model is that it permits the classification of almost all existing spatial data structures. Limiting cases of the hybrid model correspond to tessellation and vector models: At one extreme, the decomposition yields a single node in the tessellation component. This

represents a single tile that covers the entire area of interest and the hybrid structure is basically a vector representation. At the other extreme, by selecting the maximum resolution of a tessellation as the pixel level, the decomposition results in a tessellation model. Intermediate formulations of the hybrid model yield most of the specific hybrid data structures developed in the literature. Cabay and Vanzella [CV90] compared the costs associated with vector and raster organizations for performing some common operations on spatial data sets, with particular emphasis on very large thematic data sets. Their results indicate that realizations of certain intermediate formulations can provide superior practical alternative for representing large thematic data sets. The choice of data structures is, therefore, highly domain dependent.

Davis [Davis84] surveys a number of data formats for raster and vector data. He makes the important pragmatic point that spatial data should be stored in a format which closely resembles its source format. This is important for two reasons. Conversion between raster and vector can reduce the accuracy of the data. Furthermore, organizations maintaining spatial data may not want to convert their data. The second situation suggests that, for practical considerations, the question of raster versus vector is not always an issue; that is, there is often no choice about the representation type. It should be noted that in a MEDS environment, data in question are likely to be collected and maintained in a variety of different organizations. Wildlife population data, for example, may be maintained by an environmental protection department or agency, while transportation permits are more likely handled through the resource regulatory agency. Integration of various data formats needs to go beyond simply providing access to different data types. It will require a complete new look at the user interface, not only in terms of visualization and presentation but, more importantly, in terms of how user queries are integrated with computation.

To achieve this objective, the principle of *information hiding* [Ban88] must be practiced. The principle of information hiding declares that information not necessary for the user to know may be hidden from their view. For example, to develop a effective forest management application, it may be necessary to access forest cover, forest depletions, topography, climate, wildlife habitat, transportation, wildlife populations, existing forest activities, as well as land uses. It is

desirable an application be able to allow for the interaction of different data that are partly based on common, shared information, and partly on private information, without having to know how primitive data are represented and handled internally.

From a system design point of view, information hiding means that the internal data structures together with the details of procedures to be concealed in a separate module.

3.1.2 Thematic Data Error Representations and Manipulations

Research on error in GIS has two objectives, both of substantial practical significance: first, to minimize error in products, and second, to develop models of error which can be used to compute measures of uncertainty. The first is needed to maintain data consistency in domains where things are clouded by artifacts, and the second is needed to improve data accuracy through assimilation. Needless to say the second approach is a desirable feature of MEDSs.

Thematic data error can be divided into two classes: *geometric errors* and *attribute errors*. Geometric errors are characterized by accuracies in point coordinates. Attribute errors, however, are of two kinds: measurement errors (i.e., distortion in attribute values) and distribution uncertainties (i.e., an unknown distribution of a multivariate variable).

In practice, discrepancies between the data model and ground truth may occur in various forms. Examples include feature misclassification, dangling edges, self-crossing contours, etc. The most common form of error in overlaid maps is called a "sliver". Sliver polygons result when two independent distortions of the same polygon boundary are overlaid, and in large databases the number of spurious polygons so created can easily overwhelm the system [Good92b]. It is common in many systems to include code to remove slivers after overlay, using rules based on area and perhaps shape. However, such algorithms have no sound basis in thematic data handling.

In the following, the issues of thematic data error handlings will be addressed.

3.1.2.1 Geometric Error Representation and Manipulation

Distortion in lines has been described through the concept of an epsilon band by Perkal [Perk56, Blak84, Chris87]; in its simplest version the true position of the line is believed to lie

within a band of width epsilon about the observed line.

Chrisman [Chris87] has investigated the use of the model for distinguishing between sliver polygons and real polygons in topological overlay algorithms. Chrisman found that the digitization process tends to produce a bimodal distribution. This suggests that more suitable model would be some continuous distribution with asymptotic tails centred on the true line. Such a model, however, is of little use in handling errors from multiple sources.

3.1.2.1.1 Semantic Constraints for Geometric Error Handling

The major discrepancy of the traditional epsilon band approach is that the semantic constraints of the data are not utilized for error handlings. In general, there are two kinds of semantic constraints: *metric constraints* and *topological constraints*.

Metric constraints refer to constraints on the measurements of locations. For example, the position of one object with respect to the position of another object can be expressed in terms of the relative coordinate from one point to another point. Moreover, the orientation of one object with respect another object can be expressed in terms of the intersection angle between two edges. Since the underlying assumption of the data model is an imperfect world, lengths and orientations cannot be specified precisely. There is no alternative but to use ranges if the system is to tolerate inaccuracy. The correct solution is to express lengths and orientations in relative terms and to specify ranges in which they lie. For instance, the length of edge(A,B) is between 25.0 and 24.6; its orientation is between -0.5 to 1.2 degrees, etc. In general, the more constraints expressed in relative terms, the more shape information about objects is derivable. For example, instead of saying "the length of edge(A,B) is between 25.0 and 24.6 and the direction from A to B is between -0.5 to 1.2 in the absolute scale", one may wish to express local constraints in terms like "edge(A,B) is between 1.2 and 1.3 times as long as edge(C,D) and the direction from A to B is between 0.5 to 0.6 counter-clockwise of the direction from C to D". Using such facts, it is possible to conclude that ABCD is a rectangle.

Topological constraints, on the other hand, represent relations among objects such as con-

nectivities, containments, adjacencies, etc. The distinct feature of these constraints is that they are invariant under topological transformations, such as translation, scaling, and rotation.

From the theoretic point of view, both metric constraints and topological constraints are redundant and could be derived from the geometries. Their inclusion in the geographical model is nevertheless necessary for the purpose of integrity. For example, explicitly capturing adjacency relationships in the model compensates for inaccuracies in the coordinate representation.

3.1.2.1.2 Geometric Error Reduction

Geometric error reduction refers to the ability to improve spatial data representation using constraints. Geometric error reduction is a cornerstone for integration and assimilation since it would obviate the inconsistencies associated with the data caused by inaccuracy.

Definition 3.1.1: Let C be a constraint on objects $X_1,...,X_k$, and let S_i be the fuzzy range for X_i. Then geometric error reduction can be defined by:

$$Refine(C, X_j) = \{a_j \in S_j | \exists a_i\ S_i,\ i=1,...,k, i \neq j, C(a_1, \ldots, a_j, ..., a_k)\}.$$

Example 3.1: Consider two locations X1 and X2 in a one dimensional structure, let the domain of X1 and X2 be [0,7] (e.g., S1) and [0,14] (e.g., S2), respectively. The following three metric constraints are assumed:

C1: the distance between the origin and X2 is actually zero to seven feet;

C2: X1 is at least two feet from the origin;

C3: X2 is at least two feet from X1.

The following illustrates the results of geometric reduction process: The computation of Refine(C1, X2) yields S2=[0,7], the computation of Refine(C2, X1) yields S1=[2,7], the computation of Refine(C3, X2) yields S2=[4,7], and the computation of Refine(C3, X1) yields S1=[2,5].

In reality, many geometric error reductions require both topological constraints and metric constraints. The following example depicts such a situation:

Example 3.2: Assume the following constraints are adopted in the previous example:

C1: X1 coincides with X2 (i.e., a topological constraint);

C2: X1 is at most seven feet apart from the origin (i.e., a metric constraint).

The first constraint implies that S1 must be the same as S2; whereas, the second constraint concludes that the relation S1=S2=[0,7] must be true.

The above examples give us three important implications: First, topological constraints can be used to refine or correct the geometric descriptions effectively. Secondly, metric constraints do not conveniently lend themselves to structural constraint representation (e.g., it is impossible to define point coincidence, spatial object inclusion relations on the base of fuzzy range). Lastly, both topological constraints and metric constraints are necessary.

3.1.2.2 Attribute Error Representation and Manipulation

As mentioned previously, attribute errors include measurement errors and distribution uncertainties. Since measurement errors can be treated using methods similar to the epsilon band approach [Christ82], it is important to focus on distribution uncertainties.

3.1.2.2.1 Attribute Uncertainty Representation

Both statistical theory and evidence theory [KL86, CL88] are suitable tools for representing the above data types. Although evidence theory has a number of attractive features (i.e., ignorance representation, subset deduction, etc), it suffers from inefficiency in terms of both computational complexity and storage requirements. Statistics is a fundamental concept in GIS. All inferential statistics and tests of significance involve the calculation of statistics, either directly or indirectly (i.e., *a posteriori* probability). For this reason, the later formalism will be adopted:

Definition 3.1.2: Given an object O, then the property of the object, $P<O>$, is a 5-tuple $(A_o, \Delta_o, dom_o, T_o, p_o)$, which is defined recursively as follows:

- $A_o = \{a_1,...,a_n\}$, where $a_i \in A$, $i = 1,...,n$, is a set of attributes;
- $\Delta_o = \{D^o_{a_1},...,D^o_{a_n}\}$ is a non-empty set of sets of domains;

- dom_o: $A_o \rightarrow \Delta_o$ is a function that associates a domain with each attribute;
- $T_o = \underset{a \in A_o}{\times} dom_o(a)$ is a set of tuples;
- p_o: $T_o \rightarrow [0,1]$ is the statistical function associated with T_o (i.e., $\sum_{t \in T_o} p_o(t) = 1$).

For example, data input from a forest inventory often consist of classified stands of timber, defined by polygonal boundaries and homogeneous attributes. Unfortunately, the assumption that the homogeneous patch with a precise boundary is almost always an abstraction of reality. To compute an estimate of marketable timber, the attributes and area of the stand will be input to a set of yield tables. Several methods of classification, such as discriminant analysis, are capable of yielding membership probability vectors for each portion of a region, $\{p_{i1}, p_{i2}, \ldots, p_{im}\}$, where p_{ij} denotes the probability that cell i is a member of class j given its spectral response.

3.1.2.2.2 Attribute Error Measurements

An important concept associated with objects is that of information. By using Shannon's information theory [KL86], it is possible to establish a number of properties of a database.

Definition 3.1.3 Given an object O, then the entropy $H(O)$ of O is defined by

$$H(O) = - \sum_{t \in T_o} p_o(t) \log p_o(t).$$

In GISs, features such as moist forest, wet forest and rain forest are not mutually exclusive, and they are likely highly dependent on attributes such as precipitation and mean temperature. To capture these type of dependencies, the following definitions were originally proposed by Shannon [KL86]:

Definition 3.1.4: Given X, $Y \subseteq A_o$, then the entropy of $X \cup Y$ is defined as

$$H(X \cup Y) = - \sum_{x \in X, y \in Y} p(x,y) \log p(x,y)$$

where $p(x,y)$ is the joint statistics of x and y.

Definition 3.1.5: Given X, $Y \subseteq A_o$, then the entropy of Y, given X, is defined as

$$H(Y/X) = - \sum_{x \in X, y \in Y} p(x,y) logp(y/x),$$

Cavallo and Klir [CK89] show that entropy and conditional entropy are related by

$$H(X \cup Y) = H(X) + H(Y/X) = H(Y) + H(X/Y).$$

Clearly, the entropy method can be used to measure the distribution uncertainties associated with objects and their relations. Furthermore, a well-known technique called *the standard deviation* [KL86] could be adopted to describe the measurement errors.

3.1.2.2.3 Attribute Error Manipulations

This section establishes a foundation of a novel spatial statistic approach for geographic data handling. The following three important statistic operators will be introduced.

Projection Operator

A projection operator can be used to derive distributions from a multivariate variable. The formal definition of this operator is as follows:

Definition 3.1.6: Given an object O, and $X \subseteq A_o$, then the projection of set A_o onto set X results in the property $P = (X, \Delta_o, dom_o, \pi_X T_o, \pi_X p_o)$, such that $\pi_X T_o = \times_{v \in X} dom_o(v)$, and $\pi_X p_o(b) = \sum_{a \in B(b)} p_o(a)$ are satisfied, where b is a member of $\pi_X T_o$ and $B(b) = \{<a_1,...,a_n> | <a_1,...,a_n> \in T_o \text{ and } b \in \{a_1,...,a_n\}\}$.

Example 3.3: In order to determine the population distribution of palm trees and papaya trees in a region, three independent surveys were conducted. Assume *Pro.* is the population distribution of a tuple. The following figure shows the original population distribution table together with a projected table with respect to attribute Palm:

Pro.	Palm	Papaya
0.5	1988	3296
0.3	1456	2843
0.2	1988	3560

(a) a population distribution table

Projection →

Pro.	Palm
0.7	1988
0.3	1456

(b) a projection of (a)

Figure 3.3: An Example of Projection Operation

Generalization

Generalization is a process of making entity classes less specific by suppressing characteristics that describe that class [Good92a]. The following, without loss of generality, defines the semantics of a generalization operator for two objects:

Definition 3.1.7: Given object X and Y, then the generalization of object X and object Y, Z, denoted by $Z = X \Delta Y$, is defined by $P<Z> = (A_z, \Delta_z, dom_z, T_z, p_z)$, such that $A_z = A_x \cap A_y = \{a_1,...,a_n\}$; $\Delta_z = \{D^z_{a_1}, \ldots, D^z_{a_n}\}$, where $D^z_{a_i} = D^x_{a_i} \cap D^y_{a_i}$, $i=1,...,n$; $dom_z: A_z \rightarrow \Delta_z$; $T_c = \times_{v \in A_z} dom_z(v)$ such that $dom_z(v) = D^z_v$; $p_z = r_x \pi_{A_x} p_x + r_y \pi_{A_y} p_y$ where r_x and r_y are the relative weights of the samples associated with X and Y, respectively.

Example 3.4 Examine a small case. Suppose we are interested in finding the number of trees in a region defined by an arbitrary window, say W, see Figure 3.4. Since this information is not directly stored in the database, it must be derived from regions R1, R2, and R3.

To solve this problem, we must figure out the distribution function (i.e., the number of trees per square feet) of the region, W. According to the principle of spatial dependency, the property of this region tends to posses similar attributes as its neighbors. In other words, the distribution function of the region, p, can be calculated using the formula: $p = r_1 p_1 + r_2 p_2 + r_3 p_3$, where r_1, r_2 and r_3 are the relative weights, and p_1, p_2 and p_3 are distribution functions for regions R1, R2, and R3, respectively.

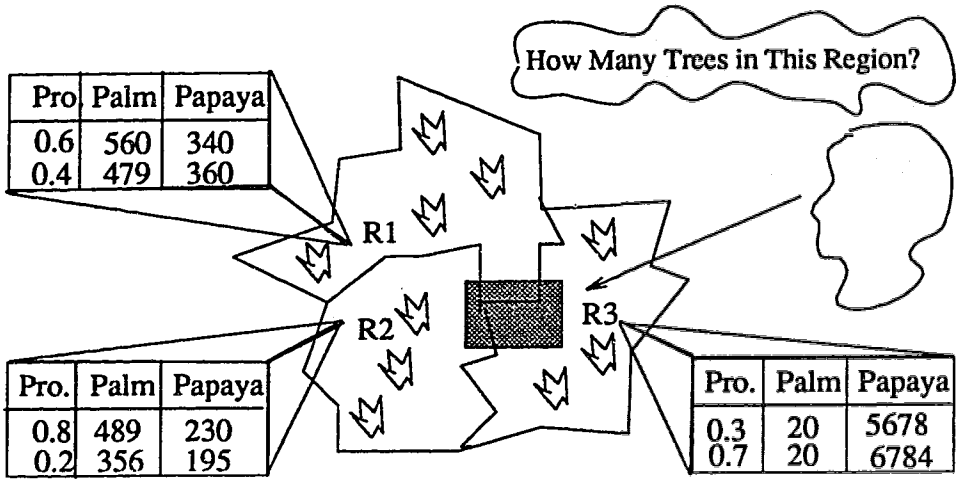


Figure 3.4 An Example of Geographic Object Generalization

The question that remains is how to determine r_i, $i=1,2,3$. A simple approach could be r_i = Area(Intersect(W,R_i))/Area(R_i), $i=1,2,3$. A careful study, however, will reveal that region R_3 is likely to be an papaya orchard. If the boundary of this orchard extended to region R_1 and R_2, then the given region should also be an orchard. Hence, in general, domain specific knowledge is usually required during the generalization process.[1]

Unlike geometric generalization, attribute generalization refers to the capability to get a concise measure of the characteristics of a set of relevant spatial objects. This is analogous to the super-class concept in an object-oriented paradigm. The major difference here is that in an object-oriented model a super-class is formed by clustering objects having the same features with respect to class, whereas, in the case of attribute generalization the above requirement is no longer necessary. Nevertheless, due to the property of spatial dependency, the underlying spatial objects are assumed to possess similar features. This situation legitimatizes the need for attribute generalization.

1. It should be pointed out that neither object-oriented systems nor the extended relational DBMSs support this type of queries.

Integration

Integration, on the other hand, is a very useful operator for reasoning with a multiple source situation. To illustrate this, consider another example: Assume that two remote sensing images of an area were obtained under the same condition, and a discriminant analysis was applied yielding two membership probability functions: T1 and T2, see Figure 3.5.

The task is to combine the evidence from both sources. This can be done in three steps:

1) Apply projection operator on T1 and T2, with respect to attributes {Moist Tundra, Moist Forest}, yielding T3 and T4, respectively.

2) Combine T3 and T4, yielding T5.

3) Construct a distribution function p with attribute vector <Dry Scrub, Moist Forest, Moist Tundra, Wet Tundra>, such that the projection of p with respect to {Moist Tundra, Moist Forest} results in the same distribution as that of T5.

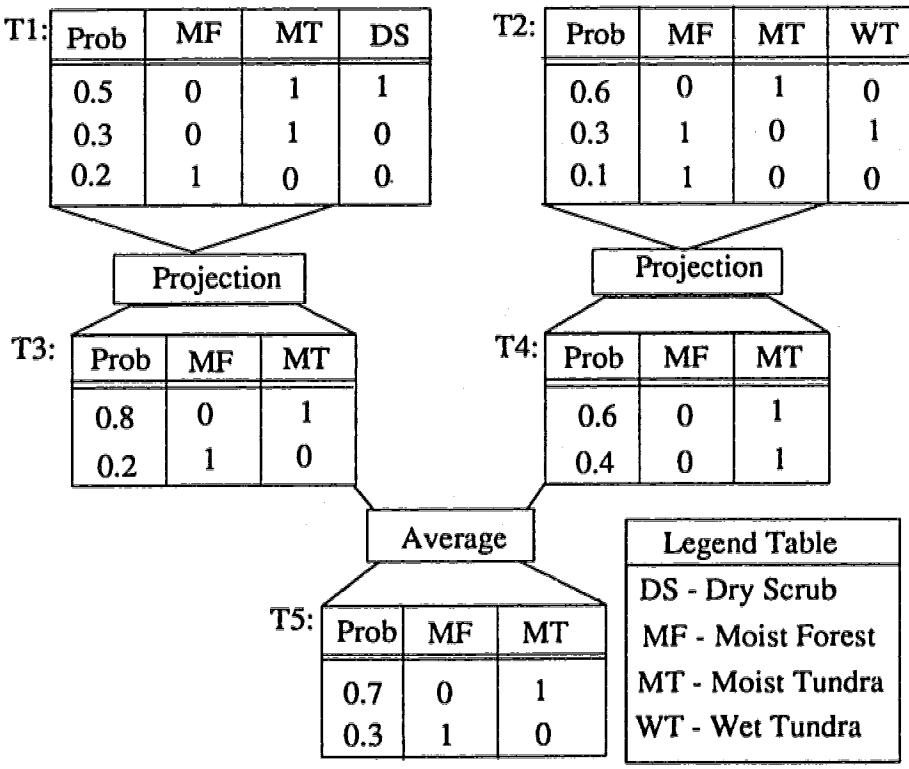


Figure 3.5: An Example of Data Integration

The last step, however, has no unique solution. From the information point of view, a solution that has the least biased is needed. To be the least biased is to minimize the amount of information contained in the distribution. Among the distributions satisfying the constraints there is a unique one [CP87], the maximum entropy distribution, that has minimum information. Hence maximum entropy inference selects the right distribution. The maximum entropy method represents a permissible strategy in estimating the uncertainty associated with the resultant object.

In summary, an integration operator can be defined as follows:

Definition 3.1.8: Given object X and Y, then the integration of object X and object Y, Z, denoted by $Z = X \nabla Y$, is defined by property $P<Z> = (A_z, \Delta_z, dom_z, T_z, p_z)$, such that $A_z = A_x \cup A_y = \{a_1,\ldots,a_n\}$; $\Delta_z = \{D^z_{a_1},\ldots,D^z_{a_n}\}$, where $D^z_{a_i} = D^x_{a_i} \cup D^y_{a_i}$, $i = 1,\ldots,n$; $dom_z: A_z \rightarrow \Delta_z$ such that $dom_z(v) = D^z_v$, for every v in set A_z; $T_z = \times_{v \in A_z} dom_z(v)$; where the distribution function p_z is determined using the maximum entropy method, i.e., $H(p_z) = \max\{H(p) \mid \pi_R p(t) = w_x \pi_R p_x(t) + w_y \pi_R p_y(t)\}$, $(\forall t)\ t \in T_z(t)$, $R = A_x \cap A_y$ is the set of shared attributes between X and Y, and w_x and w_y are the weights of conceptual dependency corresponding to object X and Y, respectively.

3.1.3 Concept of Spatial Orientation

The purpose of this section is to develop a full-fledged model that is both object-oriented and spatially-oriented.

For many years, the phrase "spatial orientation" has been used as a general term referring the capability of a data model for storage, manipulation, and retrieval of spatial data according to location [Anto87, MF89, Good92a, NS93]. Nevertheless, there is no clear definition in the GIS community on what precisely constitute a spatially-oriented model. To overcome this problem, the following concepts are identified that should be included in systems that can be categorized as truly spatially-oriented. Each concept will be elaborated and justified at all levels, according to the requirements of spatial data representations and manipulations.

1) Notion of Complex Spatial Objects

Assume a two dimensional space R^2, the following concepts for complex spatial objects will be proposed:

- Notion of Spatial Entity: A spatial entity is a triple $p = (n, t, f)$, where n is an id-term, $t \in \{point, curve, region\}$ is the type of the entity, and f is a mapping from n into R^2. A domain object is called a *spatial object*, if it is also a spatial entity.
- Composite Object: A composite object is an object with a hierarchy of component objects. A composite object can be constructed using grouping. The concept of composite object is analogous to the concept of object class. But there are some differences. The most important difference is that rather than specify a set of objects as having the same characteristics, a composite object puts a set of objects together according to their geographic significances in terms of search, inference and recognition. For instance, a typical national map may contain a collection of cities, borders of provinces, and a set of landmarks such as major rivers, islands, etc. It is neither necessary nor possible to include every object in the country.
- Aggregation: Aggregation is a construct which enables types to be amalgamated into a higher-order type, the attributes of whose objects are a aggregation of the objects of the constituent types. There are many aggregate functions such as generalization, integration, SUM, Mean, Ave, Max, Min, etc. For example, the wildlife population in North American is the sum of the wildlife populations in Canada, Mexico and U.S.A.
- Dynamic Spatial Object: A dynamic object is a spatial entity associated with an object type. The attributes of the object are aggregations of the attributes of objects of the type. Example 3.4 depicts a situation where a dynamic object is constructed by means of generalization. In general, a dynamic object can be either a user-defined window or a spatial entity created by a query process. Dynamic objects provides a window to view the world as a set of overlapping continua, instead of forcing the world into a mold of rigidly bounded objects.

2) Notion of Spatial Encapsulation and Inheritance

Spatial encapsulation has two meanings: 1) the concealment of internal spatial data structures together with details of procedures for manipulating them; 2) the separation of spatial components from nonspatial components. The first requirement has been well justified in Section 3.1.1. The second requirement will now be discussed.

Spatial features are usually referred to as shape, location, topological properties, and geometric properties of spatial objects. They play an important role in object classification, recognition, visualization, and spatial search. Moreover, the spatial relationships among entities and their relationship to cultural and terrain features represent important contextual knowledge for reasoning in these domains. There is no doubt that the way these features are organized crucially affects the effectiveness and the efficiency of the system.

Many prototype systems have been proposed to store spatial data along with nonspatial data [Anto87, Bun87, Haas91, Oren86]. A common feature of these systems is that they adopt an encoding scheme (e.g., the Morton sequence [DX86, Peuq84]) that results in generating numerical key values. These values are then stored in record attributes in the same way as nonspatial attributes. By using this approach, a spatial operation is viewed as a two step process: first, retrieve the spatial data and then operate on it. Since spatial data is stored with nonspatial data in a database, much time is wasted on the retrieval step. Both spatial and nonspatial data are retrieved although only the spatial portion in needed. This has led many researchers to try to enhance the performance of the retrieval step [Gutt84, RF88, KM90, More85]. In particular, Faloutsos [RF88] performed experiments in order to find the coding with the least retrieval time for range query. These experiments clearly indicate that the separation of geometric descriptions from other attributes is encouraged for large database systems. GISs such as ARC/INFO [More85] alleviate this problem by storing the nonspatial data in tuple form and storing the geometric entities, points, lines and areas in non-relational files in a special internal format. Links are maintained between the geometric data sets and the relational data by unique system generated feature identifiers.

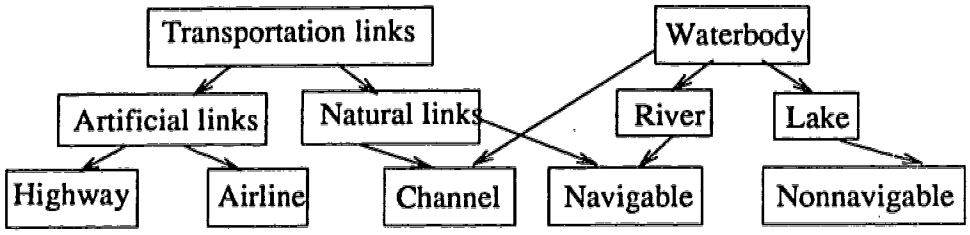
Clearly, if spatial data is structurally separated from nonspatial data, while maintaining appropriate links between the two, then the spatial data retrieval bandwidth can be much higher. The current research places a greater emphasis on the notion of separating the structure of spatial data from nonspatial data in a sense that it goes beyond the basic geometric entities.

For this purpose, the domain attribute set *A* is partitioned into two separate sub-sets: spatial attribute set and an aspatial attribute set. A spatial attribute is a location attribute, in the sense that it is inheritable from an "ISPART" relationship; whereas an aspatial attribute is a conceptual attribute in the sense that it is inheritable from an "ISA" relationship. This separation is important due to the fact that while many nonspatial properties are "ISA" inheritable, many geographic attributes, i.e., climate, soils, geology, cultural and terrain features, are functions of regions.

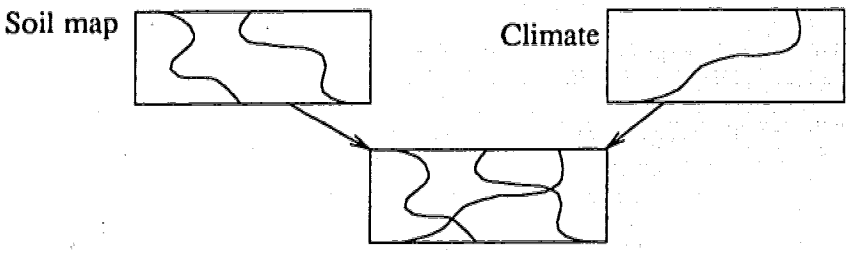
At first glance, the above separation is merely introducing another type of object hierarchy that can also be modeled by an existing object-oriented system. This analysis, however, is incorrect, since the semantics of spatial inheritance induced by the "ISPART" relation permit location based queries. Example 3.3 illustrates a situation where a window *W*(i.e., a dynamic spatial object) overlaps three regions R1, R2 and R3, and since the distribution functions are based on the regional property, *W* is able to inherit the properties from R1, R2 and R3. Such an operation would be meaningless should the attribute in question be a nonspatial feature. Therefore, it is important to distinguish the difference between the concept of spatial inheritance and the concept of object inheritance, although the two issues are closely related.

Another important consideration is how to model the hierarchical taxonomy. Some existing object-oriented systems adopt a tree structure due to its simplicity. In a typical MEDS application, a tree structure of hierarchical taxonomy is less than adequate. For instance, to develop effective agricultural management policies, we need access to transportation, waterbodies, climate, soils, etc. The above relations induce a dual lattice structure, which will be termed *the conceptual lattice*, and *the spatial lattice*, respectively. The structural feature of the model is the dualism of functions relating to two primary types of operations: object-oriented operations and spatial-oriented operations. Because of this duality many equivalent procedurals can be imple-

mented in both portions of the system. The following figure shows two simple multiple inheritance relations.



(a) A 'ISA' Multiple Inheritance Relation



(b) A 'ISPART' Multiple Inheritance Relation

Figure 3.6 Examples of Multiple Inheritance Relations

3) Notion of Spatial Data Generalization

Spatial data generalization refers to the ability to interpret spatial data at different levels with various degree of precision. In general, geometric generalization can be done in three ways: *the approximate geometry method, aggregation method,* and *the legend method.*

The first method uses an epsilon band that corresponds to the scale of measurement to approximate an object. This method entails more compact (i.e., the number of vertices can be reduced, etc.) and simplier (i.e., a long narrow region such as a river can be reduced to a line, etc.) feature representation. The second method aggregates objects having similar features into a higher order class. To this end, criteria such as the error measurements must be considered at all levels. The last method uses object legends (i.e., a collection of icons) for spatial object representation. The advantage of this scheme is two-fold: it highlights the underlying objects on the one hand, and it enables a system to bypass the complex geometry of the objects on the other hand.

Figure 2.5 provides an example of spatial data generalization: First, using the approximate geometry method, river features in Figure 2.5.a are reduced to line features in Figure 2.5.b. Secondly, using the aggregation method and the legend method, land use areas in Figure 2.5.a are either circumscribed by polygons, or denoted by icons in Figure 2.5.b.

Spatial data generalization allows us to 'vicariously experience' the geometry of the underlying spatial objects in a meanful way. In particular, the notion not only improves efficiencies for search and inference but also supports the human style of geographic analysis since more often than not a situation is first viewed from a greater distance before the region or object of interest is investigated closely. This leads to the concept of *metric space* which will be introduced next:

Definition 3.1.9: A metric space is a triple (C,ε,D), where C is a composite object, called *the context*, ε is the measure of imprecision (i.e., the scale parameter), and D is a function which calculates the distance between two locations.

The semantics at the metric level usually contains four components:

- A mapping function which maps a subset of objects into one of the metric spaces.
- A schema which selects a geometric interpretation for each object with respect to each metric space.
- A set of semantic constraints regarding the completeness and coherence of the metric space.
- A description of contexts in terms of fixed entities, such as points, paths, landmarks, and regions, linked by metric relations such as relative distance, angle, etc. In addition, each entity is associated with an unique object identifier.

3.2 Semantic Hierarchy of Spatial Objects

In the last section, a number of key decisions for designing a MEDS were identified. The question remains is how can these requirements can be fit together efficiently to form a general framework for geographical data modeling.

It is well known that human problem solving in a large-scale space is highly efficient and

robust [Hart73]. The psychological literature dealing with the development of spatial mapping and navigation [Hart73, Kuip78] provides a significant number of useful constraints on the structure of human knowledge representation. There are reasons to assume that the functioning of the human central nervous system has been optimally adapted to processing and condensing information with the help of hierarchical organization and storage principles. Furthermore, these principles optimize procedures for information processing. It is worth noting that in the area of robotics, various cognitive models for navigation and mapping in a large-scale space have been developed [KL88]. Inspired by these results and based on previous work [Zhou88], a four-level model of semantic hierarchy for the description of spatial objects is proposed: **Primitive level**: In this level, spatial objects are decomposed into a set of structure primitives. The structure models provide a description of the low-level properties (i.e., geometric properties, locations, etc.) in terms of the structure primitives. **Attribute level**: A description of the properties of objects in the environment in terms of attributes and their values. **Object level**: This level contains two parts. The first part, called *the conceptual level*, is a description of aspatial objects and their interrelationships, whereas, the second part, called *the topological level*, is a description of the environment in terms of spatial objects linked by topological relations such as connectivity, containment, and other spatial prepositions (i.e., in front of, behind, left of, beside, above, etc.). Moreover, a conceptual schema is used for the description of object type, and type inheritance for both conceptual and topological levels. **Metric level**: A description of the environment in terms of location of the spatial objects.

The cognitive map model adopts several fundamental principles such as modularity and specialization from software engineering. Since grouping all kinds of functionalities into a single unit causes problems such as overhead and complexity, the cognitive map model specializes functions into different parts. A realization of such a model consists of a general inference mechanism and a collection of function units, each of which is specialized to a specific class of data representation and manipulation. The purpose of the inference mechanism is two-fold: 1) to utilize domain knowledge to focus the attention of data processing in the most promising direction (i.e.,

under many circumstances, queries and reasoning may have both object-based and spatial-based components, the inference mechanism must decide which component should be considered first), and 2) to enforce semantic constraints between layers. For example, although both metric constraints and topological constraints are locational constraints, they belong to two different layers: the metric constraints belong to the metric level, whereas, the topological constraints belong to the topological level.

The relationships between different levels are illustrated in Figure 3.7.

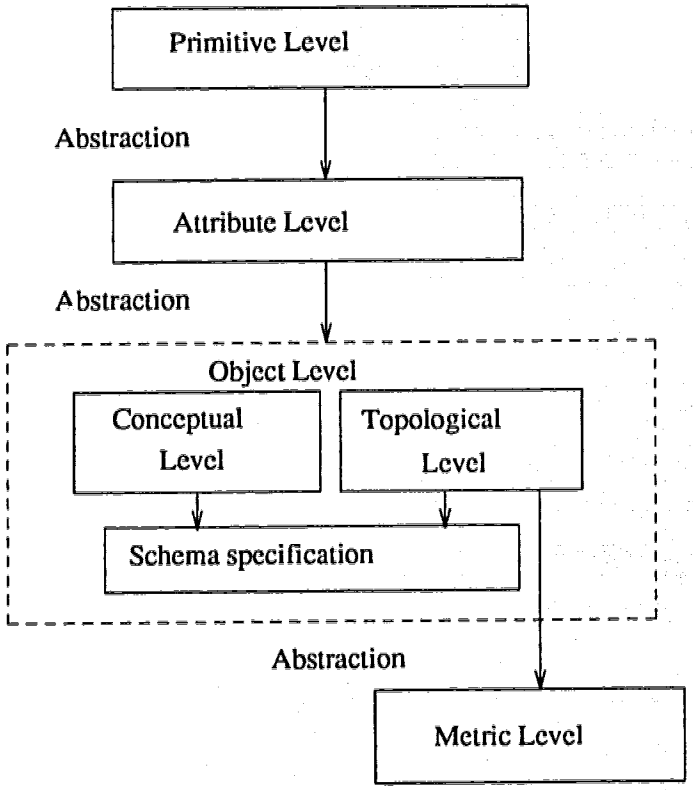


Figure 3.7 Semantic Hierarchy of Spatial Objects

3.3 Data Dependencies

It is well known that anomalies can not only be resulted by malicious treatments of data errors but also be resulted by malicious use of insertion, deletion and update in the database. To prevent this situation, a set of structural constraints, that can be enforced to preserve data integrity in the events of insertion, deletion and update, must be derived. To this end, the notion of

functional dependency and multivalued dependency will be generalized to describe the database dependencies for the new model.

3.3.1 The Attribute Level

In many cases attributes are implementable by means of tuples. The attribute level data dependency provides *a priori* knowledge of constraints on the permissible set of tuples of an object. Since the property of an object is necessarily a tuple, the data dependency for the attribute level is the same as that for a statistical counterpart of a relational database [CP87, CK89]:

Definition 3.3.1: Given an object O, let $X, Y \subseteq A_o$, then Y is said to be functionally dependent on X, denoted as FD: $X \rightarrow Y$, iff it is the case that if the tuples agree on attributes X, then they also agree on attribute Y.

By virtue of the property of conditional entropy $H(Y/X)$, the FD: $X \rightarrow Y$ can be expressed by means of the following two equivalent entropy equations [CK9]: $H(Y/X) = 0$ and $H(X \cup Y) = H(X)$.

Definition 3.3.2: Given an object O, let X and Y be disjoint subsets of A_o, and let $Z = A_o - (X \cup Y)$. Then A_o satisfies the multivalued dependency MVD: $X \twoheadrightarrow Y$ if, for any two tuples t_1 and t_2 with $\pi_X(t_1) = \pi_X(t_2)$, there exists a tuple t_3 such that $\pi_X(t_3) = \pi_X(t_1)$, $\pi_Y(t_3) = \pi_Y(t_1)$, and $\pi_Z(t_3) = \pi_Z(t_2)$.

Intuitively, $X \twoheadrightarrow Y$ means that given values for the attributes of X there is a set of zero or more associated values for the attributes of Y, and this set of Y-values is not connected in any way to values of the attributes in $R - X - Y$ [CK89]. In terms of entropy, Cavallo and Klir [CK89] show that $X \twoheadrightarrow Y$ iff $H(Y/X) = H(Y|R-(Y-X))$.

3.3.2 The Object Level

This section introduces the concepts of data dependency at the object level. The purpose of formulating these concepts is to further attenuate data redundancy and enhance data reliability.

There are two types of dependencies: *inheritance dependency* and *propagation dependency*. Inheritance dependency refers to how attributes in subclasses depend on their superclass in the object hierarchy, whereas, propagation dependency refers to how attributes of a superobject depend on its subobjects.

The difference between the concept of inheritance dependency and the concept of propagation dependency is the abstraction level in which an attribute is defined. If an attribute is originally defined at a superclass, then the semantics of data dependency implies an inheritance relationship from the superclass to its subclasses. On the other hand, if the attribute is originally associated with the subclasses, then the semantics of data dependency implies a propagation relationship from the subclasses to their superclass. There are many aggregation dependencies, i.e., the sum or union of values of the components, the greatest or the smallest value of the components, and the average and the weighted average value of the components, etc. The following illustrates an example of such dependency:

$$Canada.Population = 25{,}000{,}000.$$

$$USA.Population = 211{,}390{,}000.$$

$$Mexico.Population = 54{,}300{,}000.$$

$$North-American.Population \rightarrow bysummary.$$

Both dependencies guarantee consistency because data is only stored once and derived from there. Updates only need to consider the fundamental properties, whereas, the derived properties can be implicitly updated.

3.4 DataBase Schema

In the following, a definition of spatial database schema based on the concept of semantic hierarchy will be given:

Definition 3.4.1 *Conceptual Schema:* A conceptual schema is a 6-tuple $(ID, R, T, MN, Map_{id}, Map_{mn})$ where

- ID is a set of object identities.

- R is a set of object relations.
- T is set of type names organized in hierarchical taxonomies.
- Map_{id} is a function mapping from a subset of ID onto an element of T.
- MN is a partial ordered domain of method names with respect to four levels of semantic hierarchy.
- Map_{mn} is a function mapping from a subset of MN onto an element of T.

In other words, a conceptual schema defines two things: The semantics associated with objects (as is specified by typing functions and object relations) and the notion of type inheritance (as is defined by the partial order relationships). For example, both papaya and palm are tropical trees (i.e., an object type), a tropical tree is also a tree (i.e., an hierarchical taxonomies of object types), and every tree consists of a set of attributes such as height, age, etc. (i.e., methods associated with an object type).

Definition 3.4.2 *Geometric Schema*: A geometric schema is a triple (ID, S, $Maps$), where

- ID is a set of object identities.
- S is a set of metric spaces.
- Map_s is a mapping from a subset of ID into an element of S.

Unlike the conceptual schema, a geometric schema is purely spatially-oriented in a sense that it defines a collection of cognitive maps at different level of abstractions. On the other hand, the two schemata are tightly coupled in terms of object identities to achieve consistency and efficiency for both object-oriented and spatially-oriented representation.

The following defines the notion of spatial database:

Definition 3.4.3 *Spatial Database*: A spatial database is a quadruple $sdb = (D, S_c, S_g, M)$ where D is the set of basic domains, S_c is a conceptual schema, S_g is a geometric schema, and M is the set of applicable methods.

3.5 Conclusion

The following summarizes the relationships between the new model and previous models:

A typical early GIS represents spatial data as a collection of thematic layers, each stored as either a vector or a raster data structure. This representation corresponds to the primitive level in the cognitive map model. According to the principles in [Ullm80], such a representation is awkward (i.e., being very low degree of data integrity, consistency, and independence).

In DBMGISs, vector structure and/or raster structure primitives of the domain spatial data together with other attributes are represented by database records. This is analogous to two levels of the new model: the attribute level and the primitive level, except there is no distinction between structure primitives and other attributes. Again, these models are considered to be unsuitable for GIS due to the lack of semantic support of this class of data models.

In object-oriented models, the object level starts to play an important role. The important aspects brought by this technique is that the semantics of data relationship can be exploited to enhance the power of data representation. However, there is still no distinction between geometric properties and attribute properties.

In cognitive map models, more semantic structures are exploited to model both spatial and aspatial aspects of the underlying spatial phenomena.

The first important characteristics of the new model is the separation of the metric level from the topological level. In general, geometric uncertainty can be modeled at different abstraction levels. Because of this separation the model is able to represent and to reason with geometric imprecision effectively. The topologic level gives generality in terms of geometric entities, whereas, the metric level describes imprecisions in terms of location and relative positions of various entities. The two levels work together to achieve both data consistency and data accuracy. It should be pointed out that the importance of separating these two layer is justified in [DS73, Kuip78], and there are many other systems that followed this design principle [Davis86, KL88].

The second important feature is the separation of conceptual level from that of topological

level. In this way, the cognitive map model avoids the dilemma of object-orientation and spatial-orientation. In particular, the conceptual and attribute levels are designed to answer queries concerning the location of specific objects and to perform conceptual reasoning about these objects; whereas the metric, topological, attribute and primitive levels are designed to answer queries concerning the objects at specific locations and to perform spatial reasoning. Under many circumstances, queries and reasoning may have both object-based and spatially-based components. The two portions are therefore interrelated and designed to work together.

The third important feature is the ability to model attribute uncertainty through the use of attribute level. One of the five high-priority topics for research by the National Centre for Geographic Information and Analysis (NCGIA) is "new modes and methods of spatial analysis" [Abler87]. Abler goes on to argue the need of "non-traditional statistics" to capture the nature of attribute error and error propagation. He states:

> "In the long run we shall have to restructure our thinking about geographical phenomena and geographic inference. Geographical thinking and many conceptual tools are based on monovariate percentagism; the calculations of percentages for arbitrary statistical area such as countries or enumeration districts. GIS data will be 'pointillistic'. Maximal disaggregated data will make it possible to portray and think about several variables simultaneously, each recorded at a detailed level of precision."

In the proposed model, novel spatial statistics for implementing such an idea is developed. At the lower attribute level, statistical theory is adopted for multivariate geographic data representation; At the higher object level, many descriptive statistics are tailored into the object-oriented paradigm gracefully. More importantly, the object level can be used to refine the attribute data representation to better characterize the realty as it is reflected in the geographic data encountered.

The last important notion is the general inference mechanism. From the software engineering point of view, the provision of the general inference mechanism not only supports the specialization and modularity design principle of the new model but also provides a means of navigating knowledge inferencing and data processing.

Chapter 4

Querying Languages

The purpose of this chapter is two fold. The first is to develop a formalism for various data operations, including searching and reasoning. The second is to establish a foundation upon which structure models together with algorithms will be developed in subsequent chapters.

4.1 Recent Developments on Querying Languages for GISs

A query language is the user's tool to select data of interest, and traditional query languages such as SQL emphasize this issue by providing complex methods for data retrieval, and formulation of logical constraints upon data. Nevertheless, as analyzed previously, traditional query languages have been found to be inadequate to meet the demands of GIS applications, and various attempts have been made to solve this problem. As a result, three types of query languages have been developed: object-oriented, deductive language and extended SQL. These language classes are briefly reviewed.

Object-Oriented Approach

In the object-oriented approach [Marx86, MO87, SP84] information is highly structured by the introduction of classes and inheritance concepts. Data encapsulation and polymorphism provide some desirable manipulation facilities that are independent of the physical and logical data representation. Example: List the towns of King's county.

The data model defined in [MO86] is based on entity:

```
Type Land_division is entity
    Name(Land_division) -> String
    Area(Land_division) -> Polygon
```

and sub_type:

```
Type Area_life is Land_division
```

The above query can either be expressed using a geometric operator OB_INCLUDE (i.e., inclusion of objects):

OB_Include(Select(Area_life), Select(Land_division, Name = "King"))

or be expressed using a navigational operator:

? Area_life.Name > Land_division.Name = "King"

An object-oriented query language uses concepts that more easily describe a user's understanding of the structure of the data than does an SQL-based language. An SQL-based language requires the user to consider all data more or less in tabular form whereas the object-oriented languages allow for the consideration of complex objects as individual units with attributes and relationships between them.

Current object-oriented query languages have the following two major drawbacks. First, they do not usually support a declarative query interface (i.e., a rule-based deductive query interface). Secondly, they are not spatially-oriented in a sense that they force users to view the world as a collection of rigidly bounded objects rather than a set of overlapping continua.

Deductive Language Approach

These languages are based on knowledge representation systems. As an example, NAMEX [CJ90] implements a Prolog interface to a spatial database storing vector and raster cartographic features, along with their associated names. Database search and retrieval is performed by a set of functions implemented in a procedural language and called by Prolog predicates. In NAMEX, the cartographic task of selecting names and label configurations and avoiding conflict are expressed as Prolog rules. Many name placement rules depend upon the label's proximity to other features (i.e., settlement labels in crowded areas of the map are usually placed close to their own settlements to avoid ambiguity). This may be achieved by halving the normal separation between a label and its settlement if the settlement is within, say, 4km of another settlement. The following rule can be used to specify this condition:

```
determine_prox(Fsn,Prox,New_prox):-get_point_coords(Fsn,East,North),
                                   mask_out_current_feature,
                                   raster_circle(East,North,4000,_,Feature_list),
                                   determine_point_prox(Feature_list,Prox,New_prox).
```

A declarative query interface allows a user to pose queries at a much higher level than primitive representations of geometric objects, which releases the user's burden of understanding and programming of low-level spatial data structures. However, it is often awkward to adopt a pure deductive approach in spatial databases because many spatial primitives are computation-intensive units, which are best defined by a set of geometric-oriented computational procedures. Also, since Prolog-based deductive languages [Pere82, Frank84, CJ90] are based on a flat data model, they do not support the notion of a complex object.

Extended-SQL Approach

These languages [Gupta91, Haas91, Lorie91, NS93, WH87] are based on an existing SQL-like languages. Manipulation of geometrical information is performed using specialized operators. A query can also be expressed using a QBE (Query By Example) philosophy [Gupta91]. As an example, XSQL/2 [Lorie91] or Geo-SAL [SZ91] operators are based on three standard geometrical information: point, line and area. Nevertheless, with the XSQL/2 approach, physical data representation is independent of the data model by the definition of geometrical elements as Abstract Data Types (ADT). An ADT is an encapsulation of data structures and manipulation operators to hide all the implementation details from the user. Data manipulation is made through the use of a set of defined primitives. Example: Retrieve the map "Bay area" with its countries and cities.

Such a XSQL/2 query is:

```
Declare Cursor c For
    xmap AS SELECT * FROM map WHERE name='Bay area'
    xcountry AS SELECT * FROM country WHERE map IN xmap.id
    xcity AS SELECT * FROM city WHERE country IN xcountry.id
END
```

In XSQL/2 internal references are used to define the tuples that belong to a particular base object. For instance, a tuple in *map* is a root tuple. Some tuples in *country* are related to that map

tuple by internal references. Similarly, some *city* tuples are linked to these country tuples, also by internal references. The root and the set of tuples that are directly or indirectly related to *map* by internal references form a base complex object. In addition to ordinary tuples, XSQL/2 maintains a set of inverse relations to improve the efficiency of internal access to the base complex object. In other words, to navigate from a *map* tuple to all related *country* tuples, from a *country* tuple to all related *city* tuples, etc. Moreover, all tuples for a single map are stored together.

The main advantage is the fact that the philosophy of an object-oriented DBMS is respected. The use of an internal reference allows specification of complex objects, and the introduction of the ADT concept allows modification of the data representation without any effect on the relations defined between these objects.

The main drawback concerns the underlying assumptions that are hard-wired in the systems of any SQL-based language. For instance, since these languages rely on tuples, considerable data transformation and data construction/decomposition must occur in data transfer between an application program and the underlying DBMS. Moreover, although the traditional query languages can be extended to incorporate more real world knowledge, its main weakness are the reliance on a given conceptual schema and focus on syntactic aspects. The conceptual design process and semantics being modeled are largely ignored. On the other hand, the nature of many geographic queries involves deep reasoning. Section 2.2 provides a few examples in which the SQL-based languages are shown to be less than adequate.

In summary, the current state-of-the-art GIS query languages address important aspects of the characteristics of spatial data. However, using an existing query language does not fulfill all database requirements of a geographical system. Available object-oriented query languages provide advanced modeling capabilities, efficient navigation, and extensibility. They lack spatial-oriented access methods. Available deductive query languages provide a declarative query interface, but lack a complete object modeling capability. Available extended SQL query languages provide assertional language, modeling capabilities for complex objects. They support reasoning and navigation poorly.

4.2 Towards a Cognitive Map Approach for Geographic Query Languages

A query language is grounded in the data model on which it is based. In accordance with data modeling philosophy, the design of a language should be guided by what is needed to be modeled, rather than the other way around. To this end, the primary purpose of the proposed query language is to capture what are believed to be the essential features of a cognitive map model.

4.2.1 Representation of Hierarchical Taxonomy of Objects

The new language must come to grips with the concept of object representations at all levels, including object, object class, object type and type inheritance.

Object Representation

Objects in the proposed model can be divided into three categories: *constants*, *static objects* and *dynamic objects*. The first category consists of all persistent objects stored in the database, whereas, the second category consists of dynamic objects defined by users.

A constant is a value that explicitly appears within a query language. The most common constants by data-type classification are *integer constants* like 20001 and -51; *real constants* like -9.99 and 6.798E32; *character constants* like "A", "*", and "9".

A static object is a structured entity stored in the database. Each object is a triple <*id_term, state, behaviour*>. An object has an existence that can be uniquely identified by its identifier *id_term*. The state of an object is the set of values of its attributes. The behaviour of an object is the set of its methods which operate on either the object's id_term or the object's state.

Although an object identifier is invisible to users, its interface (an object variable) is visible to users. In fact, in the proposed language, objects and object classes are manipulated entirely through this interface. It is important, however, to distinguish identity from equality. If x and y are variables bound to objects, then x and y are identical if they are bound to the same object - a modification through x will be visible through y. Equality, on the other hand, is based on a comparison of object state.

A dynamic object is created either by a query process (i.e., overlay, union, intersection, aggregation, etc) or by users. The distinct feature of these objects is that they are not explicitly stored in the database. To achieve a uniformity, in the proposed model, a dynamic object is also perceived as a triple *<id_term, state, behaviour>*, where *id_term* is a unique system generated identifier for the object, the state of a dynamic object is derived from the database, and the behavior of a dynamic object is implied by typing inheritance.

Object Class Representation

An object class normally contains more than one object. There are two different ways to represent an object class: *enumeration method* and *object variable method.*

Using the first method, a user simply enumerates all the objects in the class within a pair of curely brackets. For instance, {Edmonton, Calgary, Montreal, Halifax} constitutes a class of cities. This method is suitable for small object class in which all members are known.

An object class can also be denoted by notion {*obj_var*}, where *obj_var* is an object variable which is usually constrained by either the semantics of the querying language or the underlying database. A user may pose a query to obtain an object class which contains all cities in the province. Such an object class can be specified using $\{x\}$, where x is an object variable that is bounded by the underlying database.

Type and Type Inheritance

The proposed cognitive map model necessitates two lattice structures: *the conceptual lattice* and *the spatial lattice* induced by "ISA" and "ISPART" relationships, respectively. Although these two lattices exhibit some functional dualities, they are not orthogonal in all aspects. Therefore, it is necessary to specify both structures at the query language level. For this reason, notions $:_c$ and $:_s$ will be adopted to represent the conceptual lattice and the spatial lattice, respectively.

- The syntactic unit *c_type* $:_c$ *item* denotes the fact that *item* is an instance of type *c_type* and the property of *item* is "ISA" inheritable, where *item* is either an object or a subtype. For example, the assertion "*Highway* $:_c$*Trans_#1*" represents the fact "Trans_#1 is a

Highway", whereas, the assertion "*Artificial_links* $:_c$ *Highway*" represents the fact "Highways are Artificial_links", etc.

- The syntactic unit *s_type* $:_s$ *item* denotes the fact that *item* is an instance of type *s_type* and the property of *item* is "ISPART" inheritable, where *item* is either an object or a subtype. For example, the assertion "*Alberta* $:_s$ *Edmonton*" represents the fact "Edmonton is part of Alberta", whereas, the assertion "*Country* $:_s$ *Province*" represents the fact "a Province is part of Country", etc.

4.2.2 Representation of Complex Object

Complex geographic object manipulations may be classified in three categories:

- *Simple object access* is necessary for insert, delete, and update operations, as well as for exact-match or range retrieval.
- *Object navigation* follows object inter-relationships, either factual or spatial, from a given object to related objects. An example of such a query is: "Retrieve the name of the mayor of the capital city of the province of Alberta".
- *Set object navigation* enables the retrieval of objects based on common properties. These properties being either expressed through some sort of predicate expression, or computed with an algebra of set-operators. Queries like: "Retrieve the names of the mayors of the towns where a river with more than one dam has its source" fall within this category.

Clearly, geographical data query requires that the querying language be able to explore the internal structure of an complex object for both assertion and navigation purposes. Moreover, methods and frequently used procedures must be incorporated into the language gracefully. To this end, the following aspects are considered:

4.2.2.1 Comparators

Comparators play a key role in specifying query constraints. In general, a comparator is a mapping θ: $<O_1, O_2> \rightarrow \{False, True\}$, where O_i are objects. There are two kinds of compara-

tors: *value-based comparators* and *set-based comparators*. A value-based comparator is one of the comparators (i.e., =, <, >, etc) for comparing two basic objects, whereas, a set-based comparator is one of the comparators (i.e., =, $\in$, $\subseteq$, etc) for comparing two set objects.

The following syntax formula specifies query constraints based on attributes.

$$<obj\_var>[<attr\_name_1> cp_1 <value_1>,...,<attr\_name_k> cp_k <value_k>]$$

where $<obj\_var>$ is an object variable, $<attr\_name_i>$, $i = 1..k$, are attribute names, $<value_i>$, $i = 1..k$, are either user specified values or values obtained from previous queries, and cp_i, $i = 1..k$, are comparators.

4.2.2.2 Navigation Operators

In referencing objects and their attributes, there are two perspectives.

- *Forest* perspective: Reference one, or more than one, attribute at a time
- *Path* perspective: Reference one, or more than one, level at a time.

The syntax form for *forest referencing* is defined as follows:

$$<obj\_var_0>[<attr\_name_1> \rightarrow <obj\_var_1>,...,<attr\_name_k> \rightarrow <obj\_var_k>]$$

where $<obj\_var_i>$, $i = 0..k$, are object variables and $<attr\_name_j>$, $j = 1..k$, are attribute names. As an example the clause $edm[mayor \rightarrow X, population \rightarrow Y]$ states: "find the mayor and the population of the city Edmonton", where *edm* is an id_term, *mayor* and *population* are attribute names, and *X* and *Y* are object variables whose types are *person* and *integer*, respectively.

The syntax form for *path referencing* is defined as follows:

$$<obj\_var_0>.<attr\_name_1>.<attr\_name_2>....<attr\_name_k> \rightarrow <obj\_var_1>$$

where $<obj\_var_i>$, $i = 0, 1$, are object variables, and $<attr\_name_j>$, $j = 1..k$, are attribute names. Using *path referencing*, the query: "find the name of the mayor of the city Edmonton" can be expressed by $edm.mayor.name \rightarrow X$, where *edm* is an id_term, *mayor* and *name* are attributes, and *X* is an object variable whose type is *string*.

In practice, which referencing approach is used depends on our needs and on what particular data are at the hand. For instance, the query: "find the name of the mayor and the population of

the city Edmonton" may be denoted as $edm[mayor.name \rightarrow X,\ population \rightarrow Y]$.

4.2.2.3 Methods and Attributes

According to the philosophy of object-oriented DBMS, the behavior of an object is the set of its methods which operate on the object's state. In general, a method is a mapping: $m : <obj\_var, para_1, \ldots, para_k> \rightarrow T$, where obj_var is an id_term, $para_i$, $i = 1..k$, are parameters, T is a target domain, and m is a function mapping from obj_var to T. Although methods are similar to ordinary procedures and functions, they should be viewed as containing an implicit parameter representing the object itself. On the other hand, it is often more convenient to view attributes as methods with single parameters in a sense that both methods and attributes are bounded to the same object. The only difference between them is that the former describes the dynamic aspects of an object, whereas the later specifies the static aspects of an object.

It is, therefore, desirable to have a uniform representation for both attributes and methods. Moreover, attributes and methods specifying the same object should be clustered together.

The following syntax formula specifies query constraints based on methods.

$$<obj\_var>[<m_1(p\_list_1)>\ cp_1\ <value_1>,\ldots,<m_k(p\_list_k)>\ cp_k\ <value_k>]$$

where $<obj\_var>$ is an id_term, m_i, $i = 1..k$, are method names, $<value_i>$, $i = 1..k$, are either user specified values or values obtained from previous queries, cp_i, $i = 1..k$, are comparators, and p_list_i, $i = 1..k$, are parameter lists. For example, the following query is requesting the annual average water discharge from lake Marion into the Santee river, and the area of lake Marion: $lake :_c marion[discharge(santee) \rightarrow X,\ area \rightarrow Y]$, where *marion* and *santee* are id_terms, *discharge* is a method, *area* is either an attribute (i.e., its value is explicitly stored) or a method (i.e., its value is calculated from the internal representation), and X and Y are object variables.

4.2.2.4 Procedure Considerations

The procedural primitives are stored in a system library which consists of two portions: *system-built-in procedures* and *application specific procedures.*

System-built-in primitives in the cognitive map model are divided into three groups:

attribute manipulations, *geometry manipulations*, and *user interface primitives*.

4.2.2.4.1 Attribute Manipulations

Attribute manipulation can be divided into the following classes:

(1) *Attribute error estimations*, that include functions for evaluating measurement errors and distribution uncertainties: *a_m_error*, *d_uncer*, etc.;

(2) *Attribute error manipulations*, that consist of procedures for error reduction, attribute projection, generalization and integration: *a_error_r*, *a_proj*, *a_general*, *a_integr*, etc.;

(3) *Set operations* such as intersection, union, difference, and complement: *a_inter*, *a_union*, *a_diff*, *a_comp*, etc.;

(4) *Aggregation functions and operators*: Aggregation functions (i.e., *sum*, *ave*, *max*, *min*, *member*, *population* and *distribution*, etc.) provide ways of summarizing information to get concise measures of their characteristics. Whereas, aggregation operators cluster individual objects into an object class according to either measurement errors (i.e., the standard deviation) or distribution uncertainties: *cluster_m*, *cluster_d*, etc.;

(5) *Attribute data manipulation operators* such as search, update, insertion and deletion: *a_search*, *a_update*, *a_insert* and *a_delete*, etc.;

4.2.2.4.2 Geometry Manipulations

Geometric manipulations procedures include the following aspects:

(1) *primitive logical predicates*, which describe the relationship of geometric objects: *g_overlapped*, *g_inside*, *g_adjacent*, etc.;

(2) *Geometric error handlings*, which include procedures for geometric error measurement, geometric error reduction, and geometric generalization: *g_m_error*, *g_error_r*, *g_general*, etc.;

(3) *Geometric data manipulations*, which include search, update, insertion, deletion, union, intersection, difference, overlay: *g_search*, *g_update*, *g_insert*, *g_delete*, *g_union*,

g_inter, g_diff, g_overlay, etc.;

(4) *Quantitative feature evaluations*, which compute the quantitative features of geo_objects: *g_distance, g_perimeter, g_area*, etc.

4.2.2.4.3 User Interface Consideration

A user interface consists of two parts: *input specifications* and *output specifications*.

There are two basic input specification functions: *cursor* and *window*. The first function picks the current position of the cursor on the screen to form a point entity, whereas, the second function returns a window selected by the user as a region entity.

Output specifications, on the other hand, contains procedures for: selecting colors, legends, display patterns, graphical representations, and output device management, etc.

4.2.2.5 Simple Arithmetic

Arithmetic primitives are essential for specifying many nontrivial queries. For example, the query: "find a river whose annual average discharge is at least 5000 cfs more than the Santee river's" requires an arithmetic operation (i.e., +) at the query language level.

For this purpose, it is assumed that arithmetic terms be usually written in the infix notation. However, similar to Prolog [Frank86, CJ90], the query language cannot tell when to consider an arithmetic term as a term itself or when to evaluate it. To force the evaluation of an arithmetic term, a new operation is required: the *is* built-in predicate. The *is* operator takes an arithmetic expression as its right operand and a variable as its left operand. All variables in the expression must be already instantiated, but the left-side variable cannot be already instantiated. For example, in: "X is Y/89 - Z" if Y and Z are instantiated, but X is not, then this clause will cause X to be instantiated with the value of the expression. When this happens, the clause is satisfied.

Clearly, the semantics of the *is* operator in the proposed language is similar to that of Prolog. The difference between the two, however, will be addressed later.

4.2.2.6 A Syntax Definition for the Proposed Query Language

The logic programming language approach offers an attractive means of solving data handling problems which require inference based on spatial knowledge. In the context of spatial data handling, a deductive query language can describe explicitly existing spatial data, in terms of location, relationships and attributes, along with rules and other facts which express how new spatial relationships may be derived from the existing ones.

4.2.2.6.1 The Definition of Terms

The language of the proposed spatial object calculus consists of a set of formulas constructed out of the alphabet symbols. Formulas are built from either a simple term or a complex term:

- **Simple Term**: A simple term is of the form $(\tau :_c X)$, $(\tau :_s X)$ $(\tau :_c C)$, $(\tau :_s C)$, $(\tau :_c f(t_1, \ldots, t_n))$, or $(\tau :_s f(t_1, \ldots, t_n))$, where t is a type which is optional, X is a variable, c is a constant, f is either an n-ary function symbol or a simple arithmetic clause, t_i $(1 \leq i \leq n)$ are variables.
- **Complex Term**: A complex term is of the form $t[l_1\ \theta_1, e_1, \ldots, l_n\ \theta_n, e_n]$ $(n \geq 1)$, where t is an object variable, l_i $(1 \leq i \leq n)$ is either an attribute or a method, e_i $(1 \leq i \leq n)$ is either a variable, a term or a collection of terms of the form $\{l_1^i, \ldots, t_{n_i}^i\}$ in which $t_1^i, \ldots, t_{n_i}^i$ are all terms, θ_i is either a comparator, the function mapping system "$\rightarrow$", or the is operator *is*.

Intuitively, a term like $\tau : t[l_1 = t_1, \ldots, l_n = t_n]$ represents an object of type τ, whose identity is t, with certain properties indicated by $t.l_i = t_i$ $(1 \leq i \leq n)$.

4.2.2.6.2 Rule Statements

Similar to Prolog, the basic form of rule statements corresponds to headed clauses. This form can be related to a known theorem in mathematics from which a conclusion can be drawn if the set of given conditions is satisfied. The right side is the antecedent, or IF part, and the left side

is the consequent, or THEN part. If the antecedent of a statement is true, then the consequent of the statement must also be true. The general form of the rule clause is:

$$<term\_0> \leftarrow <term\_1>,<term\_2>,\ldots,<term\_k>$$

which can be read as: "*term_0* can be concluded if *term_i*, $i = 1..k$, are true or can be made to be true by some instantiation of its variables". For example, the domain specific knowledge: "A floodplain is a unused river side space" can be expressed as follows:

$$floodplain :_s X \leftarrow region :_c X[landuse \rightarrow unused], river :_c Y, adjacent(Y,X).$$

4.2.2.6.3 Database and Knowledgebase Coupling

Query and reasoning traditionally belong to two separate areas: data base systems and knowledge base systems. However, it has been widely realized by both researchers and practitioners that the development of large effective knowledge bases is difficult due to the fact that knowledge-based systems provide limited capacity to supply and maintain factual data. A database system can help to overcome this difficulty. Moreover, it is often awkward to adopt a pure deductive approach in spatial database because many spatial primitives are computation-intensive units, which are best defined by a set of geometric-oriented computational procedures. This leads to the concept of database and knowledge base coupling.

Database and knowledge base coupling can be achieved by incorporating procedures written in a nonlogic language, such as *C* or *Ada*. The proposed query language allows two kinds of couplings: *the tightly couplied method* and *the loosely couplied method.*

The first approach is based on the idea of considering nonground labels as "computed functions" which can be instantiated using system-built-in procedures. For example, the logic term *adjacent*(X, Y) may be computed by calling the system-built-in function *g_adjacent*(X, Y) using clause: *adjacent*(X, Y) $\leftarrow$ *g_adjacent*(X, Y)

The second approach, on the other hand, uses the *is* operator to instantiate a variable. For instance, the clause "*w is window*", will cause *w* to be instantiated and returns the identifier of the current user_created dynamic object: *window*.

4.3 Query Examples

In developing the above querying language, the motivation is derived from the desire to capture in a logically clean way a number of scenarios whose most salient features are depicted in the following examples.

4.3.1 Conventional Queries

As previously mentioned, conventional queries can be divided into two categories: *location based queries* and *attribute based queries*. The proposed language embodies an object-oriented declarative query interface that can be utilized to specify both types of queries easily. To justify this, the following two query examples will be considered.

The first query is: "find all the pested forest within the chosen window". This is a typical location based query. Such a query can be simply specified by:

$$Forest :_c X[pested \rightarrow true],\ g\_inside(X, W),\ W\ is\ window.$$

where *Forest* is a type name, *pested* is a logical predicate, X and W are object variables, and *g_inside* and *window* are system-built-in functions.

The second query is drawn from an ecosystem for forest pest control. It is well known that forests are a major ecological habitat. Species composition, productivity and growth, nutrient cycling and organic export have been found to be related closely to weather conditions such as temperature, humidity, etc. The following query requests information about forest pests. Particularly, for each such pest, it requests the name of the pest together with the names and breeding conditions of all its natural enemies.

$$pest:_c X[name \rightarrow Y, n\_enemy \rightarrow \{Z[name \rightarrow W, breed\_cnd \rightarrow C[temp \rightarrow T, humidity \rightarrow H]\}].$$

4.3.2 Dynamic Object Queries

The proposed query language supports the notion of dynamic object in the sense that it allows users to pose queries about an area (or a location) of interest. In general, a successful query is subject to three conditions: First, the user needs to select an area (or a location) of interest. Secondly, the chosen area must be bound to a data type (the default type for a location is *point*,

whereas the default type for an area is *region*). Lastly, queries must be restricted either to geometric features or to spatial attributes.

Case 1: Examine a query illustrated in Example 3.4. Assume, without loss of generality, that regions *R1*, *R2*, and *R3* are bound to type *Forest* (i.e., $Forest :_c \{R1, R2, R3\}$), and *population_distribution* is a spatial attribute of type *Forest*. Then the query: "How many trees in the shaded rectangular area?" can be specified as follows:

$$Forest :_s W[ave(population\_distribution) \rightarrow Y],\ W\ is\ window$$

Answer: $Y = <45, 356>$ (i.e., Palm = 45, Papaya = 356)

In this example, the intended area W is created using the system-built-in function *window*, and bound to type *Forest*. The attribute *population_distribution* is implicitly derived from *R1*, *R2* and *R3*, whereas, the expected value of the population distribution of the region is obtained using the aggregate function *ave*.

Case 2: Assume that attributes *age* and *children* are nonspatial attributes of type *Parents*. Then the following query is invalid:

$$Parents :_c W[age \rightarrow Y,\ children \rightarrow \{X\}],\ W\ is\ window$$

Answer: $Y = Undefined$, $X = Undefined$.

It should be pointed out, however, that subject to type compatibilities, the proposed language does not preclude the possibility for users to associate values to any attributes of an object. For instance, the clause "$camp :_c P[name = 'Youth\ Summer\ Camp', enrollment = 50]$, *P is cursor*" defines a new camping site on a map, using the system-built-in function *cursor*.

4.3.3 Meta-Data Queries

As analyzed in chapter 2, meta-data queries are essential to geo-data interpretation and understanding. A salient feature of this type of query is that they necessitate reasoning information about the type of a given object. For instance, the queries "What is R1?" and "What are the possible pest classifications in R1?" can only be answered by considering the type definition of

region $R1$. In fact, the above queries may be expressed by using clauses such as $type(R1)$ and $type(R1)[dom(pest) \rightarrow Y]$, respectively.

Since the proposed language is strongly typed (all objects, attributes and parameters in a procedure have types), precompilation of the query language is possible. The process of precompilation is divided into two phases: 1) check the consistency of types of the same variables in the objects, attributes, procedures and predicates. 2) merge the types of the same objects to derive the minimal set of compatible types. The result of the second phase can be used to support meta-data queries.

For this purpose, a partial order relation < among all data types is defined: given two types *type1* and *type2*, *type1* < *type2* if *type1* is subsumed by *type2*, that is type1 is compatible with and more restrictive than *type2*. Clearly, the partial order relation < can be derived by either type inheritance relations or rule statements. For instance, the primitive data types of geo-primitives are *point*, *curve* and *region*. Hence, $R1$ is necessarily a *region*. On the other hand, $R1$ is also classified as *Forest* by the DBMS. According to the principle of type inheritance (i.e., $region :_c Forest$), the correct answer to the first query is $type(R1) = Forest$.

4.3.4 Knowledge Based Queries

Many geographic concepts and processes can often be expressed in terms of sets of rules. Deductive languages hold potential for representing these concepts and processes at a high level, since these languages are rule-based. Unfortunately, it is often awkward to adopt a purely deductive approach in spatial databases due to its limited capability in supplying and maintaining factual data, and its weakness in dealing with computation-intensive units. The proposed query language intends to overcome these weaknesses by integrating the power of object-oriented spatial databases with the methodologies of deductive databases. First, it supports a declarative high level query interface by defining many new properties using deductive rules. Secondly, factual data and complex structures are stored and maintained in an object-oriented spatial database. As the result, many complex reasoning tasks (i.e., object inheritance, spatial inheritance, etc.) can be directly performed within the DBMS. Thirdly, database manipulation primitives and some spatial

properties are defined by procedures implemented in the procedural language *C*. These procedures are either called by the proposed query language or invoked during the course of query processing.

In the query processing, each input parameter of a procedure must be instantiated before the procedure is called. The parameter in a procedure can be instantiated by query constants, database accessing, or the process of computing information from other procedures or terms.

To explain knowledge based query processing in more detail. Examine the database illustrated in Figure 4.1, that consists of two regions *A* and *B*, and a river *R*. Assume that both *A* and *B* are classified as unused-spaces, and their elevations are 100 feet and 95 feet, respectively.

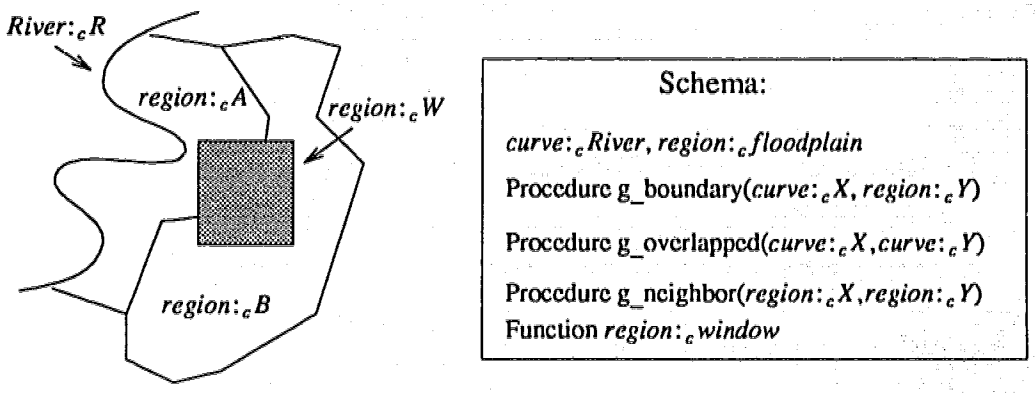


Figure 4.1 An Example of Knowledge Based Query

Assume that the following rules are adopted to determine if a region is part of a floodplain:

1. $floodplain :_s X \leftarrow X[landuse \rightarrow 'unused', elevation < 150],$
   $river :_c Y, adjacent(Y,X)$
2. $floodplain :_s X \leftarrow floodplain :_s Z, adjacent(X,Z)$
   $X.elevation \leq Z.elevation, X[landuse \rightarrow 'unused']$
3. $adjacent(X,Y) \leftarrow g\_boundary(Y,Z), g\_overlapped(Z,X)$
4. $adjacent(X,Y) \leftarrow g\_neighbor(X,Y)$

The first rule indicates that *X* is part of a floodplain if *X* is adjacent to a river and *X* is a piece of unused land with elevation less than 150. The second rule claims that *X* is part of a flood-

plain if X is adjacent to a piece of floodplain and its elevation is less than its neighbor's. The third rule states that a curve feature is said to be adjacent to a region feature if the former feature overlaps the boundary of the later feature. The fourth rule concludes that two regions X and Y are considered to be adjacent to each other if the condition $g\_neighbor(X,Y)$ is satisfied.

Suppose that it is necessary to find if window W in Figure 4.1 is part of a floodplain (i.e., $floodplain :_s W$). The query processor is driven by a resolution process similar to the standard deductive approach such as in [Kifer89] except that the processor must decide whether the current term is a procedural call, a database access unit, or a deductive term. In this case, the query processor detects that the term "$floodplain :_s W$" is not a procedure call, using the system procedure lookup table. On the other hand, since W is a dynamic object and $floodplain$ is necessarily a region, assertion "$floodplain :_s W$" must be derived from the domain of regional objects. Therefore, a spatial search based on the location of W is invoked. According to Figure 4.1, this search process returns two objects A and B. In other words, in order to show "$floodplain :_s W$", one must first prove that "$floodplain :_s A$" and "$floodplain :_s B$" are true. Again, since both assertions are not facts, attempts aimed at trying to verifying these facts by accessing data are doomed failure. Hence, the query processor resorts to deducing these assertions. Clearly, according to rules 1-4, after a member of steps, the query processor is able to conclude that W is part of a floodplain. Note that during the resolution process in verifying "$floodplain :_s A$" and "$floodplain :_s B$", both system-built-in procedure calls (i.e., $g\_neighbor$, $g\_boundary$, $g\_overlapped$, etc.) and database accesses (i.e., $A[landuse \rightarrow 'unused', elevation < 150]$) are involved.

4.3.5 Geo-Data Generalization Queries

In an interactive information system the interface must be user-friendly and the required operations must also be performed fast. In the case of a GIS, this implies that the user should be able to look at the data in several levels of detail. There are several reasons for this:

- If too much information is presented to the user at one time, it will be difficult for the user to perceive the relevant information. In fact, it is easy to make a GIS produce reams of output that will be of little use and can obscure important results.

- Unnecessary detail will slow down the display of information. This is especially true for sub-pixel size drawing primitives, because these are hardly visible and thus time is wasted.

In many cases, however, geographic information can be perceived succinctly without having to miss important information. To achieve this goal, the following measures are taken:

1) User friendly interfaces are included to highlight important information while filtering out unnecessary detail. These include the ability for the user to select various metric spaces, legends, colors and patterns. In particular, at a higher level metric space, many unnecessary details can be omitted, i.e., rivers may be perceived as curves, cities may be perceived as point icons, etc.

2) The proposed query language takes advantage of both object-oriented features and the declarative query interface. It is capable of modeling features such as sets, and the class/subclass hierarchy. It should also allow users to define concepts and pose queries at higher levels. Such a language results in releasing the user's burden of understanding and programming of low level primitives.

3) The proposed query language provides ways for users to summarize information to obtain concise measures of their characteristics. This is achieved by either calling aggregation functions (i.e., *sum*, *ave*, min, etc.) or by applying aggregate operators (i.e., *cluster_m*, *cluster_d*, etc.).

The following example is drawn from a typical flood protection application. To develop a flood protection plan, it is necessary to evaluate flood risk and the expected value of damage to the properties of an area resulting from a flood. In all cases it is necessary to be concerned with the discharge pattern of river systems at different areas. The discharge data, on the other hand, are collected at various gauging stations. Since there are several hundred gauging stations in the country, it is desirable to be able to get concise measures of their characteristics. To this end, those gauging stations with similar discharge patterns are clustered together using the following clause:

gauging_station $:_c X[$*category is m_cluster*$(Y,0.3)$, *Bargraph is select*(Y), *discharge* $\rightarrow Y]$

where *m_cluster*(*Y*,0.3) indicates that objects are grouped together subject to the constraint that the standard deviation of any object with respect to attribute value *Y* is less than 0.3, *Bargraph* is a user interface procedure which results in a bargraph to be displayed, and *discharge* is an attribute of type *gauging_station*.

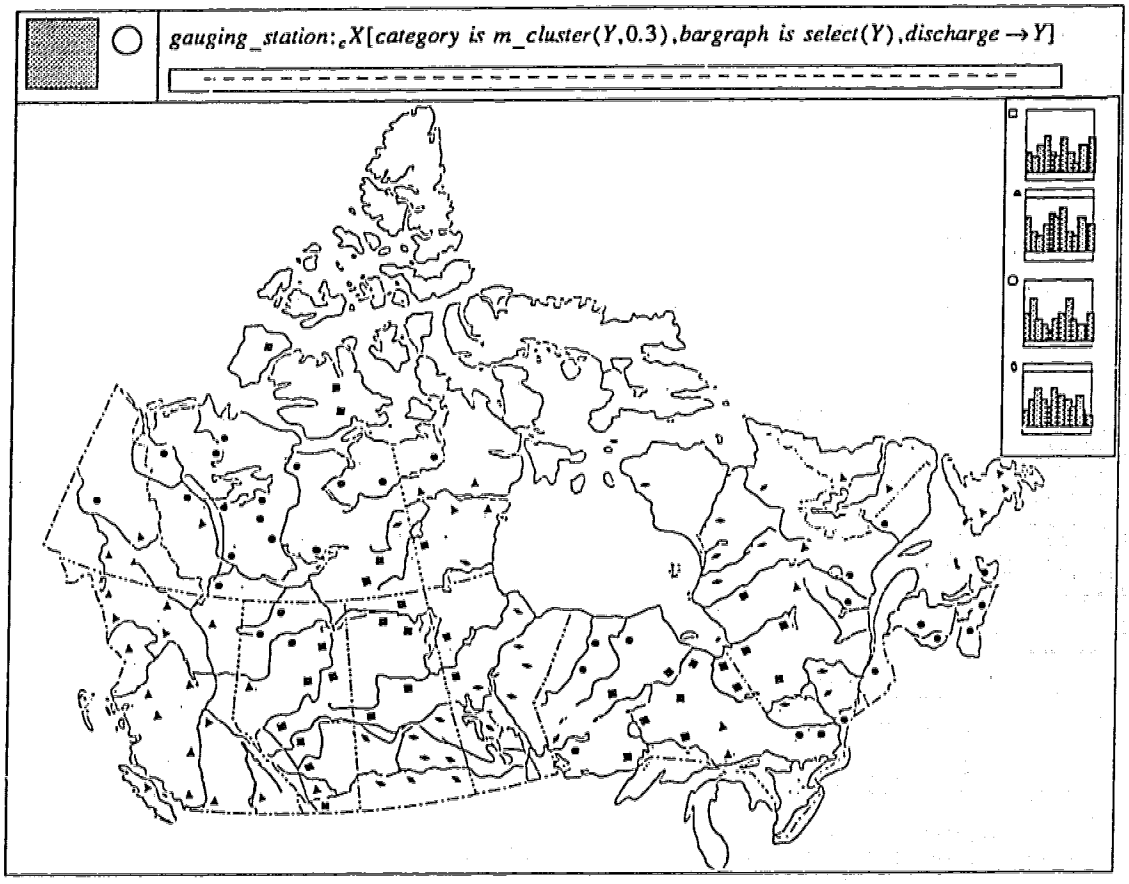


Figure 4.2: An Example of Aggregate Response

As the result, gauging stations with similar characteristics are clustered together. Moreover, since water discharge has much to do with geography and climate, spatial correlation among gauging stations is evident.

4.4 Conclusion

The proposed language provides direct support for the basic features of the cognitive map model including the semantic aspects of spatial data, the notion of spatial-orientation, and the ability to specify and manipulate uncertainties. The following summarizes some of the major

advantages of the proposed calculus:

Semantic Data Modeling: The proposed language authentically reflects the semantics of the underlying data model. First, in the proposed language, a term will have two meanings since it can be used both for denoting a class of objects and for indicating whether the denoted object satisfies certain properties. Secondly, the relationships between an object and its attributes are naturally specified by their syntactic position within the corresponding term, whereas, the metric relationships and the other object relationships can be specified as predicates. Many query examples show that the proposed language uses notions that closely match the user's understanding of the complex structure of the data. In particular, concepts such as sets, attributes, predicates, complex relations can be expressed in a clean and uniform setting. Moreover, the proposed language not only expresses complex spatial objects with ease but also supports the manipulation of spatial objects at a level of abstraction appropriate for the processing involved.

User and Programming Interface: The proposed language provides efficient and effective communication between a user and the query language, and between the query language and a programming language. First, the proposed query language enables users to manipulate the information about features of spatial object at a high level of abstraction. It also allows the specification of spatial relationships as search criteria, and supports the dynamic view of spatial phenomena in terms of windows and object constructors. Secondly, the proposed language is a high level and logic-like language that can be embedded naturally in programming language components.

Object and Spatial Reasoning: Since the semantics of the data are explicitly built into both the data and the query language, many reasoning tasks can be directly performed within the database management systems. For instance, forward chaining, backward chaining, inheritance, nonmonotonic inference become possible. Moreover, the language also lends itself conveniently to many spatial reasoning problems: 1) the computation of both absolute and relative metrics between arbitrary combination of point, line and region features, 2) spatial windowing, i.e., computation of the distribution of both spatial objects and spatial features for an arbitrary region, and 3) path

strategies such as nearest neighbor path development, etc.

Uncertainty Modeling: Perhaps the strongest endorsement of this language is that rather than rejecting uncertain data, and attempting to ignore uncertainties, the proposed language keeps them as uncertain data and processes them as uncertain information. For this purpose, a measure of reliability has been defined in the previous chapter, and this measure of reliability is explicitly incorporated into the proposed language.

Database and Knowledgebase Coupling: The proposed data model encourages such a cooperation in following ways:

- Data semantics are coupled with the data and used by database systems, i.e., semantic hierarchy, object and spatial orientation, uncertainty. As the result, many complex reasoning tasks can be directly performed within the database management system.
- The data model provides a common scheme for both data and knowledge representation, as well as a means of knowledge inferencing and data processing. In contrast to traditional logic which suffers from a voracious demand for data and encounters a combinatorial explosion before reaching a conclusion, the proposed calculus is based on an object-oriented logic which is capable of selectively modeling certain higher-order features such as sets, class/subclass hierarchy, and abstract data type. These features are considered to be crucial for any logic to be useful in nontrivial problem domains. For instance, in the proposed logic, object identities are commonly used. In this way, deductions can be performed at higher level of abstractions without having to consult the underlying database management systems most of the time. On the other hand, the database management system can provide support for reasoning with factual data and formulating explanations as necessary.

Chapter 5

Physical Data Organization

In a DBMS which consists of both primary and secondary memory, data processing time depends on how efficient the underlying model is implemented in terms of two-levels of structure models: the primary memory data structure and the secondary memory data structure. This chapter consists of three parts. First, in Section 5.1, a set of generic spatial operations pertaining to the cognitive map model is provided. Secondly, in Section 5.2, data structures at all levels are presented. Lastly, Section 5.3 concludes with the main contributions.

5.1 The Basic Spatial Operations

Many existing GIS designs and implementations are based on the so called *the single cohesive approach* [AS90]. In the single cohesive environment, a GIS is considered to be a spatial processor (SP) operating on a spatial database. Users and applications get information by passing a request to the spatial processor, which navigates the database to find the answer, which is then returned to the application. In this way, the details of the database implementation are hidden from the application, and other useful functions like concurrency control and crash recovery can be handled. By using this approach, a spatial operation is perceived as a two step process: first, retrieve the spatial data and second operate on it. In particular, query processors simply retrieve geographic data using spatial and attribute keys, leaving more complex geographic modelling and cartographic tasks for the application programmer.

The principal drawback of this approach is the difficulty of application development. If the problem cannot be solved by the query processor, then an application program must be written that extracts the relevant information and solves the geoprocessing problem itself. To be useful to the user, however, such a query operator must be very sophisticated - knowing thematic overlay, set operations, attribute modelling, etc. Since grouping of all kinds of functionalities into a single unit causes problems such as overhead and complexity, a system built in this manner is difficult to maintain, grow and change.

To overcome this problem, the techniques of modular software design and development have been adapted in the cognitive map model. Using this approach, all software development is organized around the concept of a module. A module is simply a collection of routines that work together to define a data structure or to perform some function. Modules are entirely self-contained - the code within one module interacts only with the code outside the module by well defined function calls. Given the definition of objects in the cognitive map model at a variety of levels - the metric level, the object level, the attribute level and the primitive level. The cognitive map model can be defined as the set of appropriate and useful tools that operate on these objects. Complex query processors can be created by combining predefined generic operations.

The basic operations associated with each level can be summarized as follows:

- *Metric level*: 1) The pick operation - The user selects a point $P = (x,y)$ on a map displayed on the screen with an input device such as a mouse or tablet. He wants to know which object selected; 2) the windowing operation - The user wants to select all objects within a given window or region.
- *Object level*: This level provides support for queries and operations related to the following concepts: 1) the object/topological relations - Queries in a spatial database are often based on the relationships among spatial objects. For example, in geographical applications typical spatial queries are "Retrieve all cities adjacent to the Santee river" or "Find all paths from city A to city B"; 2) the object types - Users may wish to select all instances of an object type (e.g., "list all floodplains on the map"), to pose queries concerning type inheritance (e.g., "find all line features which are both rivers and transportation links"), or to find the specific type of an object (e.g., "what is the meaning of an integer on the map?").
- *Attribute level*: Complex object queries and operations based on attribute constraints.
- *Primitive level*: Location-based queries together with a variety of location-based operations such as overlay, intersection, union, difference, etc.

5.2 Design of Efficient Structure Models for the Cognitive Map Model

If the semantics of a data model admit a decomposition into several components, then it is possible to focus attention on developing data structures that are best suited to each components. To this end, enough detail is specified to analyze the behavior of the operations and make appropriate choices as dictated by the problem at hand. In particular, two issues will be considered: The first issue concerns the problem of access to objects according to spatial and/or attribute constraints, whereas, the second issue deals with the retrieval of relevant information from objects and the performance of various operations on them.

5.2.1 A Guideline for Selecting Application Specific Structure Model

For a given application there exists an optimal database implementation. Such a scheme can be defined as follows:

Definition 5.1: Given a sequence of range queries $Q = \{q_1, \ldots, q_n\}$ and a set of all possible structure models, say S, for a spatial database. Then $s \in S$ is said to be an optimal structure model with respect to the windowing property iff

$$\sum_{i=1}^{n} w_i t(q_i,s) f_i = \min\left\{T \mid T = \sum_{i=1}^{n} w_i t(q_i,s') f_i, (\forall s') s' \in S\right\},$$

where w_i is the weight factor assigned to query q_i, $t(q_i,s')$ is the processing time for query q_i in structure model s', and f_i is the frequency of query q_i.

The above definition is purely formal. It can, however, help us to understand the implications for designing an effective structure model. First, although in practice the frequency of a query is typically unknown before hand, it can usually be estimated using simulations. Secondly, according to [VC89], various structure models can be classified using the notion of hybrid data models. Thirdly, application specific queries are all variations of three fundamental queries: 1) location-based queries (i.e., given a location, what are its attributes?); 2) attribute-based queries (i.e., given an attribute, what are its locations?); 3) hybrid queries - These queries are both

attribute-based and location-based. For instance, thematic overlay is a typical hybrid query. Thematic overlay involve the complete traversal of the input structure representing two or more themes. Each region of one theme is overlayed by all the regions of the other theme which overlap the same area, see Figure 3.2.g.

Consider a general hybrid data model for the representation of thematic data. The hybrid model is based on the method of divide-and-conquer applied to a vector encoded data structure. The hybrid model is described by introducing a tessellation to subdivide an area of interest into a set of adjacent cells. This tessellation component of the model serves both as a partition of and as an index to the vector data. Within each cell, the thematic data is still represented by the vector component. The subdivision of the tessellation component is allowed to continue to any level of resolution. If the resolution is that of pixel size cells, then one limiting case (i.e., the maximum allowable resolution) of the model is achieved, namely, a raster structure. At the other extreme, if the resolution is limited to a single cell covering the entire area, then the other limiting case is achieved (i.e., the minimum allowable resolution), that is, a vector data structure.

Cabay and Vanzella compared different structures for performing some common operations [CV90]. Their results can be summarized as follows: First, the structure offering the best attribute-based performance is vector. The cost with the hybrid structure is somewhat more costly since there is the potential for processing of irrelevant vector data. Secondly, the structure offering the best location-based performance is tessellation. The cost with the hybrid representation is slightly more expensive. Finally, hybrid queries such as the thematic overlay operation require retrieval and processing of entire themes regardless of the representation used. The hybrid model combines the advantages of both tessellation and vector models. Retrieval cost is kept to linear like a tessellation method, but the amount of data involved is substantially less. The cost of processing vector data is improved over a totally vector approach [VC89].

On the other hand, it is also equally true that the resolution parameter of a hybrid structure provides us with a measurement on the performance of various operations: The higher the resolution threshold value, the better the model towards to location-based queries. On the contrary, the

lower the resolution threshold value, the better the model towards to attribute-based queries. Hence, the problem of selecting an optimal structure model is reduced to the problem of finding an appropriate resolution threshold value for a given application. From a practical standpoint, gaining anything from divide-and-conquer means that a suitable intermediate formulation of the hybrid model must be determined. A specific intermediate formulation of the hybrid model depends on the nature of the application, especially the type and quantity of data and the types of operations applied to the data.

Assume, without loss of generality, that the range of the resolution parameter is between 0 and 1 inclusive (i.e., [0,1]). Let $r(q_i)$ be the best threshold value for query q_i, and r_o be the best threshold value for a given application, then r_o can be determined using the formula below:

$$r_o = \frac{\sum_{i=1}^{n} w_i r(q_i) f_i}{\sum_{i=1}^{n} f_i}$$

In practice, the value of $r(q_i)$, $i = 1,\ldots,n$, can be determined as follows:

$$r(q_i) = \begin{cases} 1 & \text{if } q_i \text{ is a } \textit{location-based query} \\ k & \textit{where } 0<k<1, \text{ if } q_i \text{ is a } \textit{hybrid query} \\ 0 & \text{if } q_i \text{ is an } \textit{attribute-based query} \end{cases}$$

where k can be obtained by simulations on actual data.

5.2.2 Structure Model Considerations for Expert System Applications

The previous discussion reveals that there exist no single structure model that is efficient for all applications. To be specific, consider large expert systems in which huge amount of factual data is kept in a DBMS and deductions are performed at high level, using rules. The task is to select a structure model that is best suitable to these applications.

A distinct feature of an expert system is that deductive queries are posed extensively. The fact that all resolution algorithms are rooted in the so called *Constraint Satisfaction Problem* [Kifer89] decides that deductive queries are NP hard. That is the worst case time bound for a

deductive query is deemed to be nonpolynomial.

To overcome this problem, an object-oriented deductive query language was developed in the previous chapter. The advantage of this approach is that it enables a system to focus its attention on entire objects without also having to consider all the other facts it knows. Moreover, the system is able to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanism in the most promising directions. This is important since straightforward approaches tend to lead to combinatorial explosion [Kifer89].

Aside from query language implementation, one must also consider the underlying data structures. We claim that a attribute based model is a suitable choice for such an application. The justifications are:

1) Although the proposed language allows users to perform location-based operations inside of a deductive query, majority of ground terms are attribute-based (i.e., given an attribute, find its locations). This is because that most location-based queries are interactive in nature, whereas, a typical deductive query is usually an automatic reasoning process with no or very little human interaction.

2) The proposed language builds its foundation on the notion of object-orientation. To support this notion, at the physical level, one should model the world as it really is - the world contains individual objects, each of which has several properties, including location of the object and relationships to the other objects. It is necessary to store all attributes together to form a single complex object. Moreover, it is desirable to have a simple and clean representation for location features. The attribute based models fit the bill [CV90].

3) Many deductive queries must draw inferences based on either topological or geometrical properties of the underlying objects. Most of these properties must be computed from the data representation. An attribute based representation compares favorably with either a totally tessellation or a hybrid approach to computing these properties [CV90, Marx86, Cam90, Van89].

The disadvantage of selecting an attribute based model is also evident - the representation is

inefficient in supporting both location-based and hybrid queries. To alleviate the problem, an index file must be constructed on top of an attribute based organization.

5.2.3 Indexed Files

The problem of how to organize data in both main and external memory so as to facilitate subsequent retrieval of the data is crucial in most large databases where the size of the data precludes the possibility of keeping all the data in main memory. In geographic databases, this problem is even more complicated due to the spatial nature of the queries that need to be handled. Spatial queries are inherently different than ordinary data processing queries in that retrieval by both location and attribute is usually required.

In the following, two techniques, namely, *approximate geometric enclosure* and *approximate conceptual enclosure*, for constructing indexed files for spatially-based query and attribute-based query will be proposed:

5.2.3.1 Approximate Geometry Enclosure

There are two basic spatial queries. The first is known as the *point-in-object* problem which states that given an object and a point, determine whether or not the point is contained in the object. The second one is often called the *windowing operation*. In general, a window is defined as a polygon. The windowing operation consists of extracting a portion of the objects in the system contained in a window. This operation is also known as *range search* or *clipping*.

To be effective, a database implementation is required to have a "localization" or "windowing" property. For instance, it is not possible to store all potential points that may be requested for a point-in-object problem. Instead, a point-in-object problem is solved by referring to the information in the neighborhood of the given point. Clearly, an optimal structure model must take several factors into account, including the types of queries, the system organization, and the size of the database. In particular, for very large database systems such as a typical GIS, important search time reduction advantages can be achieved if and only if 1) all the information required to solve a given problem is stored physically together, 2) only those parts of the data that are

spatially relevant to the search request reside in main memory.

Intuitively, it is advantageous to group a set of objects which are spatially close to one another to form a single object class. If the object class falls within a range query, then all objects in this class are qualified objects, otherwise, if the object class is away from the query, then none of the objects in the class are candidates. Unfortunately, this simple idea may not work well due to the fact that the complexity of the spatial embedding function is increased accordingly. To overcome this problem, the concept of approximate geometric enclosure is proposed.

Definition 5.2: Given a spatial object O in a k-dimensional space R^k, let f_s be the spatial embedding function of O, i.e., $f_s: O \rightarrow R^k$ and function g: $O \rightarrow R^k$ be an approximate geometry of f_s. Then g is called an approximate geometric enclosure of f_s if $f_s(O) \subseteq g(O)$ is satisfied.

Let $W \subseteq R^k$ and g be an approximate geometric enclosure of f_s, then the following properties are true:

1) $W \cap g(O) = \varnothing \rightarrow W \cap f_s(O) = \varnothing$.

2) $g(O) \subseteq W \rightarrow f_s(O) \subseteq W$.

Since a query with g is made simpler, both above properties are found to be useful in spatial search. The first property can be used to narrow the search space, whereas the second property can be used to obtain the results of the search quickly.

The following defines the approximate geometric enclosure for an object class:

Definition 5.3: Assume that an object class O is formed by grouping a set of objects $\{O_1,...,O_n\}$, and each object O_i, $i = 1,...,n$, is associated with an approximate geometric enclosure function g_i. Then a function g is called an approximate geometric enclosure of O iff g is an approximate geometric enclosure of $\bigcup_{i=1}^{n} g_i$.

It is now clear that when $O_1,...,O_n$ are spatially close to one another, then object O together with g favors both windowing operations and point-in-object queries. To be effective, in the cognitive map model, the approximate geometric enclosure technique is used at different levels of

abstractions: the attribute level (approximate geometric enclosure for objects), the object level (approximate geometric enclosure for object types), and the metric level (approximate geometric enclosure for object classes).

There are numerous ways of constructing approximate geometric enclosures, the simplest method is the use of the minimum bounding rectangle for each object or object class. For example, let an object class O contain three objects O_1, O_2, and O_3. Then the corresponding approximate geometric enclosure for O can be defined as g, where g is the minimum bounding rectangle containing g_1, g_2 and g_3, which are the minimum bounding rectangles for O_1, O_2 and O_3 respectively. Figure 5.1. illustrates this concept.

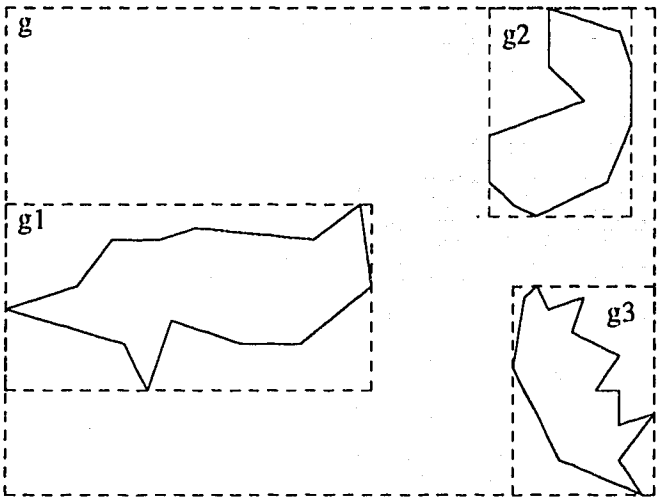


Figure 5.1 Approximate Geometric Enclosure for Spatial Objects

In general, other techniques such as the minimum bounding convex polygon can be use to construct an approximate geometric enclosure. For instance, the use of a collection of triangles in approximating 3-dimensional data has been used successfully in computer graphics, solid modeling and terrain data representations. The advantage of the minimum bounding rectangle approach is its simplicity and uniformity. On the one hand, spatial search with the representation is quite simple and can be performed efficiently. On the other hand, the user specified window and all the minimum bounding rectangles in the system are the same with respect to shape and orientation. This uniformity is quite useful in developing both efficient encoding schemes and search algo-

rithms.

5.2.3.2 Approximate Conceptual Enclosure

Attribute-based queries are characterized by search for objects with specific attribute values. A major factor towards query efficiency is the complexity of the property description. To overcome this drawback, a technique called an approximate conceptual enclosure is introduced next.

Definition 5.4: Given an object O, let $P<O> = [a_1:O_1,...,a_n:O_n]$ be the property of O. Then $P'<O> = [a'_1:O'_1,\ldots,a'_m:O'_m]$ is said to be an approximate conceptual enclosure of $P<O>$ iff either $a'_i:O'_i \leftarrow a_1:O_1 \cap,\ldots,\cap a_n:O_n$ is satisfied or $(\exists j)\, j \in \{1,...,n\}$ such that $a'_i:O'_i$ is an approximate conceptual enclosure of $P<O_j>$, for $i=1,...,m$.

In addition, $P'<O>$ must satisfy the following constraints:

- $P'<O>$ is less complicated than $P<O>$;
- $P'<O>$ lends itself conveniently to encoding in terms of data structures.

The following introduces three methods for constructing an approximate conceptual enclosure: the projection method, the flattened model method, and the semantic constraint method. As a running example in this section, the following data type is assumed:

CROPS *CD*: → **VEGETATION** with
|*name* : STRING(25)|
|*cat* : 1..7|
|*production* : INTEGER|
|*grow_in* : PROVINCE|
|*gross_profits* : INTEGER|

PROVINCE *SD*: → **COUNTRY** with
|*pname* : STRING(25)|
|*capital* : CITY|
Semantics constraint:[2]
$(\forall X)\, X \in CROPS,\ f_1 < X.gross\_profits < f_2.$

Figure 5.2 A Schema Specification

2. f_1 and f_2 are functions with parameters such as *X.cat*, *X.production*, etc.

The Projection Method

Lemma 5.1: Let $P<O> = [a_1:O_1,....,a_n:O_n]$ be the property of O and $X \subseteq \{a_1,....,a_n\}$. Then $\pi_X P<O>$ is an approximate conceptual enclosure for $P<O>$.

Proof: Directly derivable from definitions 3.3.4 and 5.4.

Q.E.D.

For a particular domain of applications, it is believed that queries will be conducted with respect to a subset of the whole attribute set. For instance, the attributes which might be used for searching a crop record can be: *name*, *cat*, *production*, etc., while other attributes are unlikely to be used as search criteria.

The selection of subset X is domain dependent. The following provides general guidance in determining X: If projection is performed on a single attribute then query is most efficient for exact match and range queries on that clustering attribute. On the other hand, if queries are based on inverted attributes rather than projected attributes then there is considerable degradation in performance. Furthermore, if projection is performed on more than one attribute, the query performance gets better as more attributes are chosen for search criteria.

The Flattened Model Method

Attributes in a complex object are nested in nature. In general, queries consult specific components of the properties and their atomic attributes at different nest levels.

Use the CROPS object class as an example, see Figure 5.2. Suppose attribute *pname* is also frequently used as a search criterion, then the clustering of objects with respect to attribute *pname* is preferable. In other words, the approximate conceptual enclosure for this example can be defined as $P'<CROPS> = [pname:O_1]$ which transforms a nested relation into a simpler and flat relation.

The Semantic Constraint Method

Both the projection method and the flattened model method rely on the syntactic knowledge about the structure of complex objects in building indices. The semantic constraint method, on the other hand, tries to exploit the knowledge about the relations, domains of their instances, and various constraints associated with them.

To illustrate the main idea, assume that a frequent query has the following form:

Select *name* From CROPS with *cat* = *X*, *production* = *Y*.

A naive index scheme calls for multiattribute clustering of the object class CROPS with respect to attributes *cat* and *production*. This can be costly from the implementation point of view. However, using semantic constraint in Figure 5.2, it is possible to build an efficient index with respect to the attribute *gross_profits*.

In summary, as indicated by the length and breadth of the consideration of approximate geometry enclosure and approximate conceptual enclosure, both methodologies are among the most important aspects of indexed access to a large member of complex spatial objects on disk. The concepts of secondary indexes become even more powerful later, as they are tailored into the specific structural components in next section.

5.2.4 Structure Models for the Cognitive Map Model

Due to the characteristics of spatial data, structure models for geographic information systems have a tendency of being awkward. To overcome the problem, there is a clear separation of the notions of metric space, object type, object, and attribute in terms of implementation. The advantage of such scheme is that it enables the system to focus its attention to high level entities, so that lower level structures may stay in the background in a fundamental way.

5.2.4.1 Structure Models for the Metric Level

A number of file structures has been proposed for handling multi-dimensional spatial data, among which there are cell methods [DH86], cell trees [Gunt89], quadtrees [Samet90a], k-d-B trees [DH86], field trees [Frank90], R-trees [Gutt84] and Buddy-trees [See91]. The following

presents a detailed discussion and a systematic comparison of these methods.

5.2.4.1.1 The R-trees

A proper file structure, which is the result of applying the approximate geometric enclosure technique at a higher level of abstraction, is called an R-tree [Gutt84, Beck90]. An R-tree is essentially an index based on spatial location. Its capabilities in dealing with advanced queries, including dynamic computation of the spatial relationship between objects, paging and I/O buffering, labels it as an excellent index scheme for high level spatial data retrieve.

The underlying assumption of an R-tree is that there is a spatial database which consists of a collection of tuples representing spatial objects, and each tuple has a unique identifier which can be used to retrieve it.

An R-tree is composed of leaf nodes and nonleaf nodes with a distinct node called the root. As defined by Guttman [Gutt84], leaf nodes of the R-tree contain entries of the form:

$$(I, tuple-identifier),$$

where tuple-identifier refers to a tuple in the database and I is an n-dimensional rectangle which is the bounding box of the spatial object index:

$$I=(I_0, I_1, \ldots, I_{n-1})$$

Here n is the dimension and I_i is the closed bounded interval [a,b] describing the extent of the object along dimension i.

On the other hand, non-leaf nodes contain entries of the form:

$$(I, child-pointer)$$

Where $child-pointer$ is the address of the successor node in the next level of the R-tree and I is the minimal rectangle which bounds all rectangles in the descendent node's entries.

Let M be the maximum number of entries that fit in one node and let $m \leq M/2$ be a parameter specifying the minimum number of entries in the node. Nodes in an R-tree corresponding to disk pages of reasonable size having values of M that produce good performance so that a spatial search needs to visit only a small number of nodes.

To make an R-tree into a dynamic structure that needs no periodic reorganization and to utilize the spatial relationship efficiently, the following requirements are imposed:

(1) Every node contains between m and M index records unless it is the root.

(2) For each index record $(I, tuple-identifier)$ in a leaf node, I is the smallest rectangle that contains the n-dimensional data object represented by the indicated tuple.

(3) For each entry $(I, child-pointer)$ in a non-leaf node, I is the smallest rectangle that contains the rectangles in the child node.

(4) The root node has at least two children unless it is a leaf.

(5) All leaves appear on the same level.

It turns out that the spatial lattice can be gracefully adapted in an R-tree, see Figure 5.3. It is not surprising since both an R-tree and the spatial lattice preserves locality. In fact, the adaptation of the spatial lattice in an R-tree shows a number of desirable features: First, objects that are near to each other on the study area are near to each other in the file with a high probability of being stored in the same physical disk block. Secondly, objects from the same physical disk block are likely to form a "sub-lattice". Furthermore, both near neighbor finding and spatial hierarchical reasoning can be accomplished by searching upper level nodes in the R-tree.

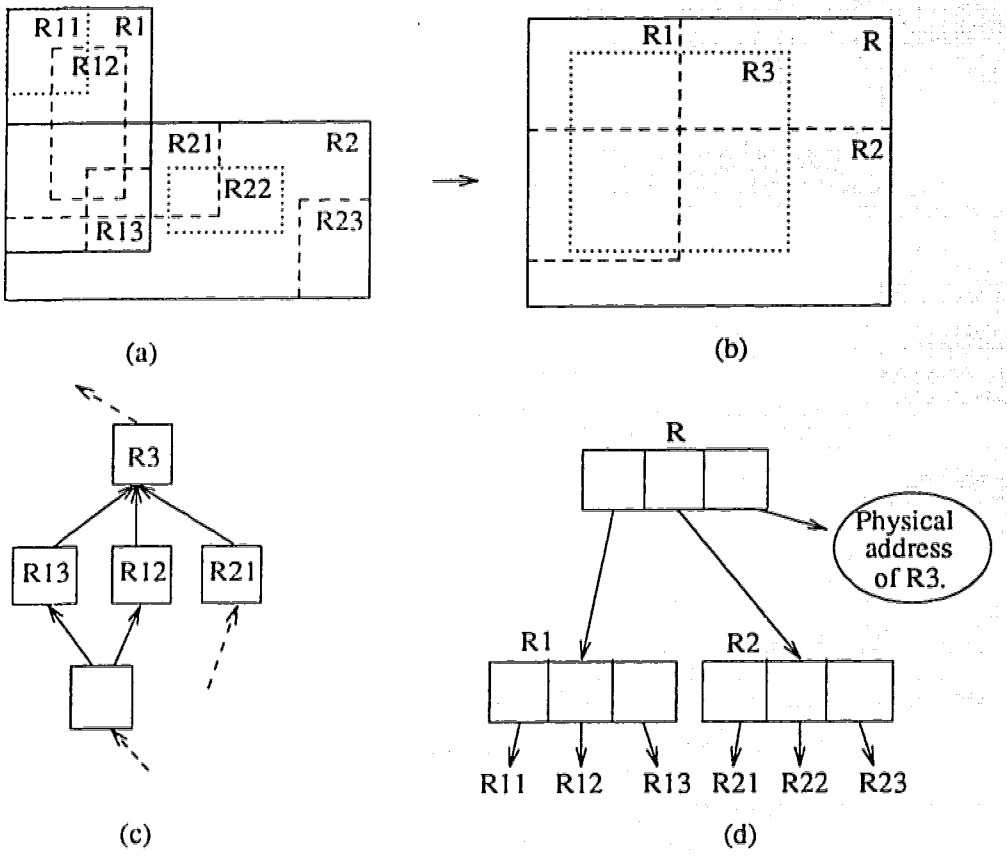


Figure 5.3 Adaptation of the Spatial Lattice into An R-Tree

(a) A Possible Configuration Among R-Tree Nodes at the Lower Level of the Tree.

(b) A Possible Configuration Among R-Tree Nodes at the Higher Level of the Tree.

(c) A Possible Sub-Lattice Embedded into the R-Tree.

(d) The Corresponding R-tree.

The other structures in this category include the R^+-trees [Beck90] and R^*-trees [Gre89]. A survey of the algorithms and comprehensive comparison of the different R-trees can be found in [Beck90]. The paper concluded that an R^*-tree performs essentially better than any other R-tree structure. It should be pointed out that the data structure of the R^*-tree is the same as for any other R-tree. However, the R^*-tree has completely different insertion and splitting algorithms.

5.2.4.1.2 The Buddy-trees

The buddy-tree organizes records with a tree-based data structure whose internal and leaf nodes are called directory and data pages, respectively. Data pages contain only data records whereas the directory pages contain so-called directory entires. A directory entry comprises a pointer to a subtree and a rectangle that covers all the records in the corresponding subtree.

A buddy-tree can be implemented using three techniques: clipping, overlapping regions, and transformation. Overall, the technique of overlapping regions is the most efficient one. In the following, the implementation of overlapping regions on top of the buddy-tree will be reviewed:

To build up the structure, the centres of the rectangles are used as 2-dimensional keys. The rectangles are inserted in a 2-dimensional buddy-tree. There is one basic difference to the buddy-tree used as a pure point access method. The directory rectangles stored in the leaves of the kd-trie are not the minimum bounding rectangles of the centres, but of the whole rectangles stored in the corresponding subtree. In addition, two kinds of rectangles associated with the structure: the R-rectangles and the B-rectangles. The R-rectangles are the minimum bounding rectangles for the whole data rectangles stored in the corresponding subtree, whereas, the B-rectangles are the minimum bounding rectangles of the centres. The R-rectangles in a page may have a common intersection. However, the B-rectangles are disjoint.

Spatial queries are performed without making use of the B-rectangles. Instead, queries require only the R-rectangles. For instance, for a point query, simply follow the pointer to the subtree whenever the corresponding R-rectangle covers the point. The B-rectangles, on the other hand, are exclusively used for insertion. A rectangle is inserted in a page when the centre of the rectangle is covered by the B-rectangle of the page.

The Choice between an R-tree and a Buddy-tree

The advantage of overlapping regions implemented on a buddy-tree is that insertions and deletions are supported very efficiently. In general, an insertion first assumes an unsuccessful exact match query. This query is restricted to a single path for the buddy-tree, while the R-tree

has to traverse almost completely several paths of the directory. A comprehensive experimental comparison [See91] based on data over six distributions demonstrated that for small query regions the buddy-tree with overlapping regions outperforms the R^*-tree. With increasing query region size the R^*-tree becomes better. The paper concludes that the buddy-tree with the technique of overlapping regions offers a similar retrieval performance and essentially better dynamic behavior (i.e., less cost for insertions and deletions) than the R^*-tree.

Clearly, a buddy-tree with overlapping regions is more suitable in the domain of CAD/CAM systems, due to the fact that insertions and deletions occur frequently in these systems. However, in the area of large geographic information systems, the majority of operations are information retrievals. The choice for the R-tree as our implementation paradigm is not by accident at all. The justification is:

[1] The storage utilization of the R-tree is superior to the buddy-tree in all cases [See91]. Since geographical data files are extremely voluminous, a file scheme which offers better storage utilization is desirable. In particular, a typical environmental database may contain millions of spatial objects. For instance, the Canada Geographic Information System which originated in the 1960's [CM89], has an accumulated data bank containing some 20 million polygons, and routinely processes seamless databases exceeding 100,000 polygons. This indicates that disk space should not be taken for granted when chosing a file scheme.

[2] Although all experimental results shown in [See91] indicate that the buddy-tree with overlapping regions offers a similar retrieval performance than an R^*-tree, the results can be biased for two reasons. First, although the height of a buddy-tree is still logarithmically bounded [See91], the structure is not height balanced. This implies that the worst case performance of a buddy-tree can be poor despite its excellent average case performance. On the other hand, an R-tree is height balanced. Thus, R-trees are more efficient than buddy-trees in handling interactive queries. Secondly, an R-tree preserves locality of domain spatial objects in a better fashion than a buddy-tree in the sense that the former clusters objects according to both shape and size information, whereas, the latter clusters objects according

to the location of the centres of the objects. Therefore, in applications, such as a typical GIS data file, in which there exist high variance in the size and the shape of the domain objects, a buddy-tree does not offer a fair clustering heuristic.

5.2.4.2 Structure Models for the Object Level

In general, tuple, tree, lattice and network structures are necessary for modeling various object relations. According to [Frank91], in practice, the number of different data types in a GIS is in the order of 100 to 1000 and objects in a GIS are often classified in groups and subgroups with inheritance chains that are 2 to 10 classes deep. Hence, for a large and complicated domain, the use of object identifiers for encoding an object relation is highly recommended. This is because object identities introduce a level of indirection of object relations through a small index which is RAM resident. It is important for efficient updates, deductions, and reorganization since references do not involve physical pointers which would cause disk accesses. Therefore, object relations should be recorded as a system file and be loaded into main memory during each session.

Data structures together with efficient algorithms for representing and navigating tuple, tree, lattice and network are well developed. By using the proposed structure model, it is relatively easy to admit existing methods to the system. Typing and database schema may be perceived as a special case of object relations. As the result, all type of queries pertinent to this layer can be handled.

It should be pointed out, however, that many topological and object relations are not explicitly recorded. They must be derived from a lower level of the representations. To solve this problem, it is necessary to build an interface between different layers.

5.2.4.3 Structure Models for the Attribute Level

The purpose of the attribute level is two-fold: On the one hand, it must support the retrieval of spatial objects using attribute constraints; on the other hand, it must support the extraction of relevant information from the qualifying objects.

To support the first aspect, index structures are used that enable users to access the objects very efficiently. Since the underlying assumption of a GIS is the presence of a large number of complex spatial objects, the use of approximate conceptual enclosure is desirable. In the following, the structure models for referencing objects and their attributes are provided.

At the attribute level, objects are represented as a set of nested relations. Recently, new database management systems supporting nested relations have been designed, and a variety of storage models have been reported in the literature [KM90]. These models can be classified into four storage models, namely, the *Normalized Storage Model*, the *Partial Normalized Storage Model*, the *Decomposed Storage Model*, and the *Flattened Storage Model*. Each storage model has its advantages together with its disadvantages. However, generally speaking, if most queries manipulate entire the object, then the Flattened Storage Model is preferred; if most queries manipulate internal relations, then the Decomposed Storage Model or the Normalized Storage Model is preferred; if most queries manipulate specific individual components of relation tuples and their atomic attributes at different nesting levels, then the Partial Normalized Storage Model is a good choice. For detailed discussion, the interested readers are referred to [KM90].

5.2.4.4 Structure Models for the Primitive Level

Natural geographic objects tend to be convoluted and irregular. They subsequently are not well-defined in terms of location, shape, and topological relations. To solve these hard problems, it is usually necessary to decompose the objects into a set of primitives such as points, lines, polygons, pixels, quadrants, etc. A variety of data structures have been developed for this level, including quadtree, raster, chain code, topologic model, POLYVRT, etc [Peuq84, Van86]. These structure models can be classified into two classes: the vector model and the raster model (for a survey on basics the reader is referred to [Peuq84, Van86]). The choice of data structure for this level is, however, domain dependent. In general, a file structure such as a linear quadtree with a B-tree or hashing scheme incorporated is considered to be appropriate for large databases [DH86, Zhou88].

5.3 Conclusion

This chapter has identified a number of basic spatial operations in relation to the semantics of the proposed model. Two important general purpose methodologies for indexing spatial objects according to spatial constraints and attribute constraints have been elaborated. Structure models for navigation of objects according to object relations and attributes have also been discussed.

The distinct feature of the proposed model is that it has an extensive set of tools which can operate on this data model. Users can interface with the system either at the basic tool level or through applications and interfaces layered on top of these tools. The primary goals in the development of this model as a software system have been generality and extendibility. All modules are self-contained: the code within one module only interacts with code outside the module by well defined function calls.

Chapter 6

System Performance Analysis

"Performance" refers to how well a system works. It is based on measurable factors within the system (in terms of system resources) and on functional factors (in terms of human resources). The performance assessment of commercial GISs has traditionally taken the form of application-specific benchmark tests commissioned by individual user agencies. This method is inadequate to evaluate future GISs since the testing methodology does not allow users to predict performance levels in terms of functional factors. Literature about performance evaluation acknowledges that human resource expenditures are at least as important as computer resource expenditures in evaluating system performance [ST92]. Traditional approaches deliberately exclude the functional factors from formal study because they do not lend themselves to quantitative measurement. To overcome this limitation, a different approach will be adopted to justify the proposed model. In particular, the following issues will be discussed:

- Efficiency of principal analytical algorithms such as query and update.
- Storage requirement for the new scheme.
- Functional factors such as ease to use, reliability and extendibility. These factors cannot be measured at all; they are simply verification that the system possesses specified features.

6.1 Query Performance

Search is a cornerstone for both query and update operations. For this reason, a novel search algorithm based on the cognitive map model is developed. The distinct feature of the algorithm is that it not only takes uncertainty into account, but also handles both object-oriented and spatially-oriented search criteria.

The procedure takes as input six parameters W, T, C_o, C_a, U, and P where W is a set of spatial windows, T is the type of the objects to be searched for, C_o is a set of object constraints, C_a is a set of attribute constraints, U is the uncertainty associated with the resultant objects, and P is the set of properties that must be returned, with respect to qualifying objects.

```
Procedure Search (W, T, C_o, C_a, U, P)
begin
                                                    {metric and object levels.}
   For given W, and the metric space l;
   R_w:=Window(l, W, T); R:={};
                                                    {object relations}
   R_w:=Uncertainty-Reduction(R_w, U);
   while R_w is not empty do
      r:=head(R_w); R_w:=tail(R_w);
      If Satisfies(r, C_o) then R:=R|{r};
   end-of-while;
                                                    {attribute level.}
   C_n:={};
   while neither C_a nor R is empty do
      Con:=head(C_a); C_a:=tail(C_a);
      If there is no index file for Con then .[3]
            C_n:=C_n|{Con}
            else If R is not empty then
               begin
                  If Con is a spatial constraint then
                     R:=Explore(R, s-lattice, Con)
                  else
                     R:=Explore(R, a-lattice, Con);
               end
      end-of-while
      return(Search-lattice(R, C_n, P));
end

Procedure Explore(O, L, C)
begin
   if L = 's-lattice' then select the spatial lattice
   else select the aspatial lattice;
   choose an index file, I, for C
      S:=search-index(I, C); R:={};
      while O is not empty do
         o:=head(O); O:=tail(O);
         If there exist r ∈ S such that o ≤_a r (or ≤_s) then R:=R|{o};
      end-of-while
end.
```

The following five subroutines are invoked in the procedure **Search**: Function **Window** retrieves objects within a window. For a given set of objects, function **Uncertainty-Reduction** returns the set of subobjects that satisfies the prespecified uncertainty requirement. Function **Satisfies** tests whether a given object satisfies certain constraints. Functions **Explore** and **Search-lattice** test whether a given object satisfies certain attribute constraints by searching the corresponding index structure and storage structure respectively. In addition, three list operators

3. A heuristic function such as the one proposed in [SP84] may be applied to here as well.

are employed: function *head* is used to obtain the first element of a list; function *tail* returns a sublist of a list by deleting the first element of the list; function | combines two lists into one.

6.1.1 Time Efficiency

There are two types of searches: main memory based search and external memory based search. To distinguish different types of operations, O_e and O_m will be used to represent the cost of disk-based searches and the cost of main memory based searches respectively.

There are three basic types of queries: the simple query, the range query, and the complex query. For a simple query, either the location or the value of a single key of an object is specified as a search criteria. A range query involves specifying a range of values or addresses of a set of objects. A complex query is a combination of simple and range queries using various logical operators (e.g., AND, OR, NOT, etc). For the purpose of simplicity, it is assumed that any complex query consists of at most one location based component.

6.1.1.1 Cost Functions for Simple Queries

Assume that simple queries return a set of object identifiers. Then the following two possible cases are analyzed:

Case 1: Location-Based Queries

Suppose N is the total number of objects, c is the capacity of the data pages (data records per page), f is the fanout of the internal node of the tree, s is the average size of the data objects, assuming the size of the screen is 1. Then the theoretical analysis in [Falo87] shows that the time complexity for retrieving a single object, using a typical R-tree structure is bounded by $O_e(log_f \frac{N(1+s)}{c(1+s)-s(1+N)})$.

Case 2: Attribute-Based Queries

Suppose that a B-tree file scheme is adopted. Let N be the maximum number of objects in an index file, c be the capacity of the data pages (data records per page), f be the fanout of an

internal node.

Without loss of generality, assume a fully-packed B-tree (i.e., every data page contain c entries, each internal node has f sons). It is not difficult to show that the height of the B-tree is bounded by $O_e(log_f M/c)$. Therefore, the worst case time complexity for indexing a single object is bounded by $O_e(log_f M/c)$.

In summary, since the proposed model consists of two complementary index structures, one for the object and the other for the spatial portion of the system, it facilitates both location based and object based simple queries.

6.1.1.1 Cost Functions for Complex Queries

The procedure is logically divided into three phases: metric and spatial level search, object level search, and attribute level search. Hence, the time complexity of the procedure is determined by the cost of execution time of these three phases.

The Cost of Phase 1:

Based on the previous assumption, a location-based component of a complex query can either be a simple query or a range query (i.e., windowing operation). The cost of a location-based simple query as indicated previously is given in [Falo87]. In the following, the cost of windowing operations will be analyzed. For this purpose, the formalism in [Falo87] will be followed, and subsequently extended.

As explained in Chapter 5, in an R-tree level, domain objects are encoded as a collection of rectangles aligned with the axes. An easy way to represent these boxes is to consider them as points in a 4-dimensional space. For a box aligned with the axes, four coordinates are enough to uniquely determine it (the x and y coordinates of the lower-left and upper-right corners).

Since 4-D spaces are difficult to illustrate, line segments (1-D objects) instead of boxes (2-D objects) will be considered. In this case, each line segment x is uniquely determined by the coordinates of its start and end points (i.e., $<x_s, x_e>$). Hence, it can be represented as a point in a two dimensional space.

Assume that the "screen" for these 1-D objects is a line segment which starts at 0 and ends at 1. Figure 6.1 shows some line segments and their 2-D representation.

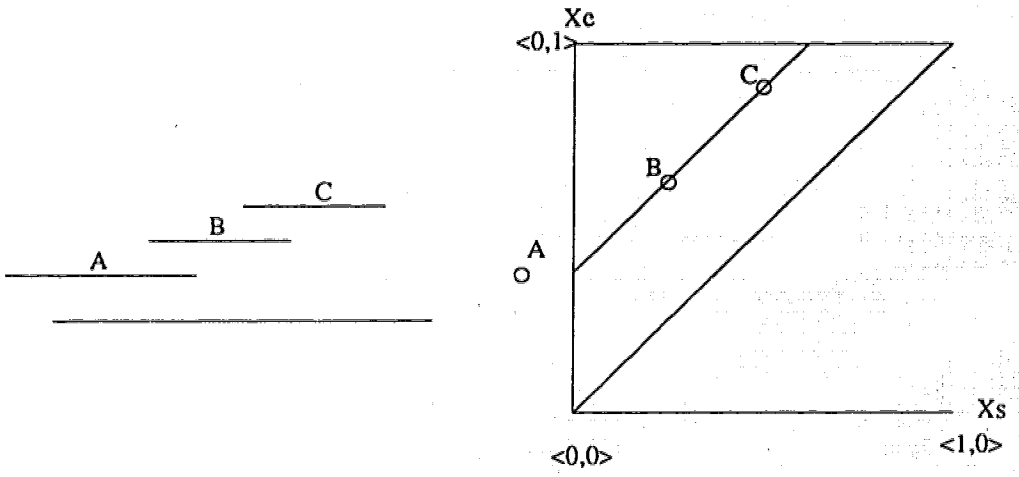


Figure 6.1: Line Segments and Their Representations in 2-D Space.

Some important observations, with respect to the transformed space:

1) There are no points below the diagonal (i.e., assume $x_s \leq x_e$);

2) Line segments of equal size, like B and C, are represented by points that lie on a line parallel to the diagonal;

3) Line segments not entirely within the screen, such as A, are allowed.

To make the analysis tractable, assume that line segments are of equal size and are uniformly distributed.

Given N segments (domain segments) of size s, uniformly distributed on the screen, a line segment (query segment) of size q intersects with $intsec(N, s, q)$ segments, Falousos et. al. showed [Falo87] that

$$intsec(N, s, q) = \frac{q+s}{1+s}(N+1).$$

By the same token, the following lemmas are established:

Lemma 6.1: Given N segments of size s, uniformly distributed on the screen, the overlaps among

the segments are $O_v(s,N)$ segments, where

$$O_v(s,N) = \frac{s(N+1)}{1+s}.$$

Moreover, assume that an R-tree is constructed from the above N segments, then the inequality $c > O_v(s,N)$ is true.

Proof: By definition, overlap is the average number of segments that intersect a single point. Hence,

$$O_v(s,N) = intsec(N, s, 0) = \frac{s(N+1)}{1+s}.$$

On the other hand, if an R-tree is built from the above N segments, then a correlation between parameters s and N must be imposed. In particular, at the leaf level, each data page contains c entries and every object that intersects a given page must be inserted as an entry of that page. In other words, assume the size of a data page is d, then constraint $c \geq intsec(N, s, d) \geq O_v(s, N)$ must be satisfied.

Q.E.D

Lemma 6.2: Given N segments (domain segments) of size s, uniformly distributed on the screen. Assume a line segment (query segment) of size q such that $q > s$, then q covers with $Cover(N, s, q)$ domain segments, where

$$Cover(N, s, q) = \frac{q-s}{1+s}(N+1).$$

Proof: Consider Figure 6.2, projecting the line AB on the horizontal axis, the line AB' of size $1 + s$ is obtained. The query region intersects the line AB on a segment CD, whose projection C'D' is of size $q - s$. The fraction of the line segment falls within the query region is $length(CD)/length(AB) = (q - s)/(1 + s)$. Since the line AB is divided into N+1 equal intervals by circles, the line CD will contain on the average $(q - s)(N + 1)/(1 + s)$ circles, which is exactly the number of segments that the query segment q will cover.

Q.E.D

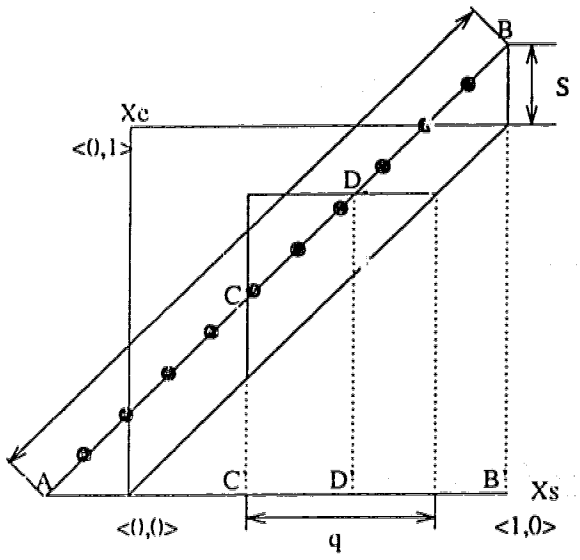


Figure 6.2 Line Segments Covered by a Query Segment.

The next result, which is attributed to Falousos et al. [Falo87], lays a foundation for the analysis. Hence, it will be restated in terms of **Lemma 6.3** as follows:

Lemma 6.3: Assume a full R-tree (i.e., every data page contains c entries, each internal node has f sons) is constructed from a set of N boxes of size s, uniformly distributed on the screen. Let h be the height of the R-tree, then $h = \log_f N - \log_f(c - O_v(s, N))$.

Now, the formal analysis of windowing operations for an R-tree will be given.

Theorem 6.1: Assume a windowing operation is performed on a full R-tree constructed from a set of N boxes of size s uniformly distributed on the screen. Let h be the height of the tree and q be the size of the query rectangle. Then the total number of disk accesses with respect to the operation is $Win_d(q, s, N)$, where $Win_d(q, s, ,N) = h + q(f^h - 1)/(f-1)$, or

$$Win_d(q, s, N) = \log_f \frac{N}{c - O_v(s, N)} + q\frac{N - c + O_v(s, N)}{(c - O_v(s, N))(f-1)}.$$

Proof: Since the tree is full, at the ith level, $i = 0..h-1$, there are exactly f^i pages that divide the screen into f^i intervals. Due to the uniformity assumption, each interval will be of size f^{-i}.

According to [Gutt84], windowing on an R-tree is as follows. A node, starting with the root, is brought into internal memory and searched for the given query window among all entries. If the node is a leaf node, then each entry corresponds to an object on the screen. Therefore, all desired objects (i.e., all entries contained in the given query window) with respect to the node are located. If the node is an internal node, then each entry corresponds to a sub R-tree. Hence, all potential sub-trees must be searched further.

The forthcoming analysis can be more easily derived by applying the following simple observation of the above search procedure:

The roots of the potential sub-trees associated with the entries that intersect the given query window at level i, where i=0,..,h-1, must be brought into internal memory.

The total number of disk accesses can be calculated as follows:

$$
\begin{aligned}
Win_d(q, s, N) &= \sum_{i=0}^{h-1} intsec(f^i, f^{-1}, q) \\
&= \sum_{i=0}^{h-1} \frac{f^{-i}+q}{1+f^{-i}}(f^i+1) \\
&= \sum_{i=0}^{h-1} (1+qf^i) \\
&= h + q\frac{f^h-1}{f-1}.
\end{aligned}
$$

In addition, according **Lemma 6.3**, $h = \log_f N - \log_f(c - O_v(s,N))$ must also be satisfied. The result of Theorem 7.1, therefore, follows immediately from the above derivations.

Q.E.D

Theorem 6.2: Assume a windowing operation is performed on a full R-tree constructed from a set of N boxes of size s uniformly distributed on the screen. Let q be the size of the query rectangle. Then the total number of objects retrieved by the windowing operation on the average is $Win_o(q, s, N)$, where $Win_o(q, s, N) = (q - s)(N+1)/(1 + s)$.

Proof: Since an R-tree guarantees the correctness and the completeness of windowing operations,

the above mentioned windowing operation returns all objects spatially contained in the query window q. Moreover, in case of a uniform distribution of N objects of size s, The total number of objects retrieved by the windowing operation on the average is $(q - s)(N + 1)/(1 + s)$, according to **Lemma 6.2.**

Q.E.D.

Corollary 6.1: Assume a windowing operation is performed on a full R-tree constructed from a set of N boxes of size s uniformly distributed on the screen. Then for a sufficiently large value of N, the cost of a windowing operation is less than 1 disk access per qualifying object. Furthermore, the cost of the operation in terms of the number of disk access per qualifying object decreases as the total number of the objects retrieved by the operation increases.

Proof: It is not difficult to show that given two arbitrary constants A and B, where $A > 0$, there exists a positive number ε such that if $N > \varepsilon$, then $A(N+1)+B \geq \log_f N$ is satisfied.

Assume h is the height of the R-tree, and q is the size of the query rectangle. Let $A = \frac{q}{2(f-1)}$ and $B = -\frac{c}{f-1}$. Then the following inequality exist:

$$
\begin{aligned}
h &\leq \log_f N \quad (By\ \ Lemma\ \ 6.1) \\
&\leq \frac{q(N+1)}{2(f-1)} - \frac{c}{f-1} \\
&\leq \frac{q(N+1)}{(1+s)(f-1)} - \frac{O_v(s,N)}{f-1} \quad (By\ \ Lemma\ \ 6.1) \\
&\leq \frac{q(N+1)}{(1+s)(f-1)} - \frac{s(N+1)}{(1+s)(f-1)} \\
&\leq \frac{(q-s)(N+1)}{(1+s)(f-1)} = \frac{Win_o(q,\ s,\ N)}{f-1}.
\end{aligned}
$$

On the other hand, it is straightforward to show that $f^h - 1 < N + 1$.

Combining the above results, for a sufficiently large N value, the cost of a windowing operation in terms of the number of disk accesses per qualifying object can be estimated as follows:

$$\frac{Win_d(q, s, N)}{Win_o(q, s, N)} = \frac{h}{Win_o(q, s, N)} + \frac{q(f^h - 1)}{(f-1)Win_o(q, s, N)}$$

$$\leq \frac{1}{f-1} + \frac{q(1+s)}{(f-1)(q-s)}$$

$$\leq \frac{1}{f-1} + \frac{2q}{(f-1)(q-s)}$$

$$\leq \frac{3}{f-1}.$$

Since in general $f > 3$ must be satisfied for an R-tree, the number of disk accesses per qualifying object is less than 1. In particular, if f is greater than 30, then number of disk accesses per qualifying record is less than 0.1.

To prove the second property, let $r = (q - s)/(1 + s)$, then according to **Lemma 6.2**, r is the ratio of the number of qualifying objects to $N + 1$. Assume also that the height of the R-tree is h, then according to Theorems 7.1-2, the following is true:

$$\frac{Win_d(q, s, N)}{Win_o(q, s, N)} = \frac{h}{r(N+1)} + \frac{q(f^h - 1)}{r(N+1)(f-1)}$$

$$= \frac{h}{r(N+1)} + \frac{r(1+s)+s}{r} \frac{f^h - 1}{(N+1)(f-1)}.$$

Since parameters h, s, f and N are independent to r, it is not very difficult to see that the cost of a windowing operation decreases as the value of r increases.

Q.E.D

Validity of the Analytical Results

The above analysis represents an optimistical strategy in estimating the lower time bound of windowing operations on R-trees in a sense that it is based on two rigid assumptions:

1) The R-tree in hand must be fully packed in which every data page contains c entires, each internal node has f children;

2) Domain objects must have a standard size and be uniformly distributed on e map.

The first requirement, however, can be lifted. According to the definition of an R-tree (see Section 5.2.2.11), the fanout of an internal node is bounded by the constraint $m \leq f \leq M$. Since a

minimum bounding rectangle can be represented by a limited number of bytes, it is reasonable to assume that $f \geq m > 30$. More importantly, in [Zhou88], it is proved that the average storage utilization of an R-tree is close to 0.69315, which is comparable to that of a B-tree.

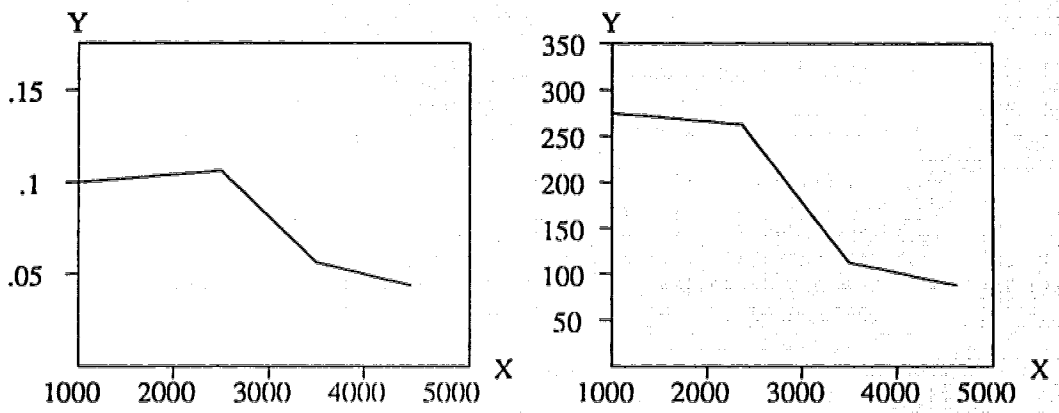
The second assumption, however, is more problematic. R-tree based schemes tend to experience major performance problems when the spatial database contains objects whose size is large relative to the total size of the data space. Each data object has to be assigned to exactly one page whose corresponding region has to be extended to cover the object. Large objects lead to large extensions and therefore lead to large overlaps between regions that may become large enough to render the index inefficient. As a result it could end up searching a significant portion of the whole index for a single point query. Furthermore, since spatial data is not usually uniformly distributed, data pages have a tendency to form irregular sizes. The price paid for the nonuniform distribution is that in order to determine the area covered by a particular object, it may be necessary to retrieve all the potential entries it occupies.

In considering the performance of an R-tree, the concepts of coverage and overlap are important. Coverage is defined as the total area of all minimum bounding rectangles of all leaf nodes, and overlap is defined as the total area contained within two or more leaf nodes.

Obviously, efficient searching demands that both overlap and coverage be minimized, although overlap seems to be the more critical of the two issues. A variation of an R-tree that avoids overlap at the expense of space is called an R^+-tree. The main difference is that rectangles can be split into smaller sub-rectangles in order to avoid overlap among minimum bounding rectangles. In some case in which a given rectangle covering a spatial object at the leaf level overlaps with another rectangle, we decompose it into a collection of non-overlapping sub-rectangles whose union makes up the original rectangle. All the pointers of sub-rectangles point to the same object. Moreover, these sub-rectangles can be judiciously chosen so that no bounding rectangle at any level need be enlarged. The same sub-splitting is propagated up to the non-leaf nodes, thus overlap is forced to stay zero.

Despite of continuous efforts [Falo87, Beck90] to improve the R-trees, spatial access

methods based on R-trees have a tendency to become awkward should the domain objects be nonuniform distributed with irregular sizes. Fortunately, various experiments results based on real data indicates that the chance of worst case happening is very rare. In particular, the importance of using an R-tree in spatial search has been studied in the past [Gutt84, Falo87, Beck90, Seeg91]. The following figure [Gutt84] shows how performance varies in terms of the number of records.



X= the number of records.
Y= pages touched per qualifying record.

X= the number of records.
Y= CPU usec. per qualifying record.

Figure 6.3 Search Performance vs. Amount of Data

The simulation was performed in C under Unix on a Vax 11/780 computer. Each search retrieved between 3% and 6% of the data. The simulation results show that as the amount of data retrieved in each search increases, the cost of processing higher tree nodes becomes less significant. The low CPU cost per qualifying record, less than 150 microseconds for larger amounts of data, shows that the index is quite effective in narrowing searches to small subtrees.

The Cost of Phase 2:

In the proposed model, most simple object relations can be represented as a pair of object identities, whereas more complex relations such as hierarchy taxonomy can be represented as a lattice or graph structure in terms of object identities. It is reasonable to assume that a lattice structure can be explored with d disk accesses, where d is a small constant. In particular, from the experiences in the past, lattice structures had been used extensively in a prototype cognitive map

system developed during the course of this research to model various geographic entities such as water bodies, transportation systems, river systems, geographic area coverages, etc. In all cases, the resultant structures were small enough to be main memory resident. This permits efficient search algorithms for various application programs since references do not involve physical pointers which would cause disk accesses.

For simple object constraints, a straightforward look-up table can be employed which usually takes $O_m(1)$ per test. In particular, uncertainty reduction may necessitates the expansion of the taxonomy of object hierarchy to a lower abstraction level. Since the operation can be easily performed in main memory, the cost of the operation is bounded by $O_m(1)$ per expansion.

For complex relations such as "and-or" relations, various efficient search algorithms from AI such as A^* (situation assessment, pathfinding), AO^* (e.g., problem solving, complex spatial relations) and minmax search (i.e., battlefield sensor fusion) can be applied.

The Cost of Phase 3:

Suppose that a B-tree file scheme is adopted for index files, and λ is the number of objects obtained from searching indice. Let N be the maximum number of objects in an index file, n_a be the number of attributes searched through index files, c be the capacity of the data pages (data records per page), f be the fanout of an internal node, α be the cost of merging two layers, L be the number of thematic layers that have to be accessed by the query.

Phase 3 consists of two sub-phases: the indexing phase and the retrieval phase:

The indexing phase uses procedure **Explore** to search relevant index files pertinent to the query. Since B-tree schemes are adopted, procedure **Explore** takes $O_e(log_f N/c)$ steps. The total cost of searching index files is, therefore, bounded by $O_e(n_a \log_f N/c)$.

The retrieval phase, on the other hand, involves searching storage structures. References to storage structures are necessary for two reasons: First, index files usually do not provide sufficient support to search all attributes; secondly, the resultant objects together with their properties must be returned.

The first step is to map object identities onto their master layers. The cost of this step is $O_e(\lambda)$. Again, since many attributes are associated with the master layer, attribute constraints can be applied to further narrow the search space from λ to λ_1, where $\lambda_1 \leq \lambda$. The second step is to retrieves those attributes defined at different layers and merge them with the master layer. The cost of this step is bounded by $O_e(\lambda_1 \alpha(L-1))$.

In summary, the total cost of phase 3 is bounded by $O_e(n_a \log_f N/c + \lambda\alpha(L-1) + \lambda)$.

Q.E.D.

Clearly, in the event where a single object must be retrieved (i.e., $\lambda = 1$), the proposed scheme is at least as efficient as any existing GIS. This is because optimal time bounds are derivable from the cost function.

On the other hand, in the event that a collection of objects must be retrieved, the proposed system outperforms existing GISs. To justify this claim, it is necessary to analyze the dominant parameters associated with the cost function.

7.1.1.3 Parameters of Cost Functions

The overall cost of a query is the total of the cost in each of the three phases. Since d and n_a are small constants, the dominant factors in the overall cost function are L, λ, and α.

Parameter L:

In the worst case, L equals to the number of attributes to be searched (i.e., each attribute is stored as a separate layer) For a nontrivial application domain, L can be very large. In particular, if every attribute is required, then L equals the total number of attributes in the system. On the other hand, in the best case, L equals 1 (i.e., all desired properties are stored in a single layer).

Note that the value of L can be simply minimized by merging all attributes into a single integrated layer. This approach yields optimal performance with respect to the cost of data integration. However, it has the tendency of indiscriminately decomposing objects into many small regions. The disadvantage is that it is too primitive and complex post processing of objects

from many small regions is often required [Gutt84].

In the proposed cognitive map model, themes are clustered at object level using a lattice model (i.e., attributes pertinent to the same region can be stored in a single layer). Such an approach aims at minimizing the cost for both data integration and post processing.

Parameter λ:

The optimal value for λ is *m*, where *m* is the number of the resultant objects. In general λ is greater than or equal to *m*. To minimize λ, the following measures are taken:

Measure 1: The Use of Index Files

One of the major contributions of the new model is that it explores the search space reduction potential of an integrated spatial representation that provides a 'window' into an object-oriented paradigm. Both an R-tree and a B-tree are adopted to improve search efficiency by narrowing search spaces.

Measure 2: Aggregate Response

Conventional responses in database systems, usually given as list of atomic objects, although sufficient to serve the purpose of conveying information, do not necessarily provide efficient and effective communication between a user and the system. An aggregate response is a list of object classes at higher abstraction levels (i.e., all vegetation except immature orchards, all transportation links except channels, etc.). The purpose of aggregate response is two-fold. The first is to improve the throughput of the system by providing concise information rather than enumerating individual objects. The second is to improve the quality of communication between the users and the system by providing expressions at a level that is easy to comprehend.

Parameter α:

Merging geographical data involves spatial search at the primitive level. Previous work in [Falo87] pointed out that in some applications, many spatial queries such as point in polygon, or

polygon in polygon may also require a search to be conducted at the primitive level.

Spatial search at the primitive level severely obstructs the efficiency of most existing GISs due to the huge amount of primitive data in these systems. The proposed model improves the efficiency of spatial search at the primitive level in two ways. Firstly, the number of searches that have to be conducted at the primitive level are significantly reduced. This is because most of the irrelevant objects can be discarded at the R-tree level without having to consult the underlying primitives [Gutt84]. In particular, experiments with a prototype cognitive map system developed during the course of this research indicated that among 100 various queries, only 13 irrelevant objects were explored. Secondly, in the proposed model, spatial objects are first systematically mapped onto a metric space, then an R-tree is constructed for each metric space. If both data abstraction and geometric generalization processes are applied in the first step, then a much smaller and simplier primitive structure results. Empirical results with real data indicates that a significant reduction in data volume is achievable using the geometric generalization process alone. In this way the number of primitives can be drastically reduced by performing spatial search at a higher metric level. For example, using a 1:15,000,000 scale of precision, the generalization of the world map data from the data bank at the University of Alberta yields a 75%-85% deduction in data volume.

6.1.2 Storage Requirements

At first glance, the proposed model calls for a large amount of extra space for incorporating uncertainties. Careful analysis, however, will indicate that the proposed model will not result in a significant increase in storage requirements due to its data sharing capability.

First of all, the hierarchical nature of the data model implies superior performance in terms of the space requirement with respect to attribute data. This is because the organization made it easy to distribute properties so that those being shared in the hierarchy are at the appropriate place for covering the maximal subset of nodes sharing them.

Secondly, as previously described, geographic objects tend to be poorly defined and their

boundaries tend to be convoluted and irregular. Hence, they do not lend themselves to a compact definition and quickly become extremely large. The issue of geometric data sharing is important.

In general, the space requirement for geometric descriptions depends on a few factors such as the domain of applications, the level of data integration, and the underlying encoding schemes. From the data model point of view, the level of data integration is a major concern.

There are two opposite ends of a broad spectrum of possible data-base schemes involving various levels of integration: the separated layer scheme and the fully integrated scheme.

At one extreme [MO87], data is represented as a collection of thematic layers, merging only at search time. This method is inefficient in terms of storage requirements due to the potential of a large amount of redundant data with respect to geometric descriptions. In addition, since thematic overlay involves many complex geometric computations, including inconsistency detection, elimination and prevention, the model is also inefficient in terms of computational complexity.

At the other extreme, on the other hand, all layers are merged into a single integrated structure. The disadvantage of this approach is that it tends to indiscriminately decompose objects into many small regions. Hence, storage efficiency deteriorates. Although the scheme avoids the issue of data integration, the absence of the ability to index spatial objects at a higher abstraction level is a significant disadvantage. For instance, the problem of finding all objects contained in a given rectangular area requires an elaborate reconstruction of objects from many small regions. As a result, this approach can represent a serious bottleneck in query processing.

The proposed model tries to strike a balance between these two extremes. Firstly, with this model, themes that tend to share a great deal of their geometry can be integrated at a master data-base layer in order to prevent high initial access time for a merge process on one hand, and to have a more compact representation on the other hand. Secondly, themes which show little spatial correlation can be represented at different layers to avoid producing many small regions and to improve the efficiency of high level retrieve processes.

6.2 Update Performance

Update in a cognitive map model can take place at different levels of abstraction. The notion of semantic hierarchy together with the representation of spatial objects through object identities allows the system to have data independence, integrity and uniformity. Firstly, and most obviously, the physical scheme can be changed by a database administrator without altering the data model. The advantage of this data independence is that it allows "tuning" of the physical database for efficiency while permitting application programs to run as if no change had been made. Secondly, at the object level, it permits efficient updates and reorganization since references do not involve physical pointers that would cause disk accesses. Thirdly, the model is able to improve its representation at different levels to include new factors and descriptions by means of assimilation, see sections 6.3-4 for a detailed discussion.

The time complexity of update at the attribute level is the same as that of **Search**. This is because the model guarantees consistency as data is only stored once and derived from there, i.e., no derived properties need to be updated explicitly, but only the fundamental properties.

6.2.1 Functional Factor 1: Advanced Data Integration

There are three scenarios that must be considered with respect to thematic layer integration.

1) Data integration between two socioeconomic layers.
2) Data integration between one socioeconomic layer and one categorical layer.
3) Data integration between two categorical layers.

6.2.1.1 Problems with Existing Integration Methods

Traditionally, data integration in GIS is done by means of thematic overlay. A layer is defined as a polygon network with each polygon associated with an unique attribute value. The thematic overlay operation involves two layers of polygon networks and consists of applying the intersection operation to every possible pair of polygons, one from each network.

Thematic overlay provides a natural way to model geographic phenomena at low abstraction

levels. By overlaying many different thematic layers, the system obtains the set of subobjects of all the objects defined in the original layers, i.e., the highest lower bound of the sublattice structure induced by these layers. There is no doubt that thematic overlay plays an important role in geographic data analysis. It is also one of the most important topics during the course of the development of GIS technologies.

It has been widely accepted that the existing thematic overlay algorithms for data integration have the following two limitations: Firstly, they are unable to integrate data at a level that matches the human's spatial reasoning style. This is because thematic overlay tends to decompose polygon networks rather than the other way around. Secondly, both attribute and geometric errors are largely ignored. In realty, geographic data are subject to a high degree of both geometric and attribute errors. Moreover, in practice, these two types of errors are difficult to disentangle. For instance, a most common form of error in an overlaid layer is called a *sliver*. A sliver occurs when a boundary between two layers is represented slightly differently in two source maps of the overlaid layer. This causes two problems: 1) a small unintended zone is created; 2) database integerity is violated (i.e., contours may cross one another).

Various models for modeling sliver errors have been reported in the literature [Good92b]. These models are all variations of the epsilon band model. Using these models, a line on one map which lies within the epsilon band of a line on the other map is assumed to represent the same line on the ground. By snapping these two lines together any associated slivers can be removed.

The above approach suffers from two major problems: First, since a sliver error in a categorical layer is likely to be caused by misclassification, the epsilon band approach does not provide a sound base for error modelings. Secondly, although snapping can be used to remove inconsistencies, it is too primitive. For instance, line A can be found to lie within line B's band, indicating A and B are the same; A can lie in line C's band, indicating A and C are the same, but line C can lie outside line B's band. In this case, it is easy to generate new inconsistencies.

6.2.1.2 Advanced Integration Capabilities

The proposed model takes geometric error, attribute error, and data abstraction into its account. Therefore, it supports many advanced integration capabilities:

High Level Data Integration

The proposed model provides users with the capability to integrate some thematic layers onto a master layer without destroying the geometry structure of the master layer. This feature is often desired when geographic information encoded on many thematic layers (e.g., categorical layers) must be censused and aggregated into a collection of prespecified divisions (i.e., a socioeconomic layer). In this case, each object in the master layer serves as a spatial window, within which properties defined at other layers are integrated using the operator ∇. Moreover, by mapping source layers onto the desired metric space, the geometric uncertainties can usually be either reduced or absorbed. This will greatly improve the efficiency of the various algorithms.

Thematic Overlay Between Socioeconomic Layers

As previously described, sliver removal is a key issue in thematic overlay. One way to get around the problem associated with the naive snapping method for sliver removal is to record the sliver as a statistical data type with more than one possible outcomes. The disadvantage of this method is that it is too conservative in a sense that it never deletes any trash data. This may lead to a disaster as more and more curves join the team after many overlay operations.

It turns out that in the proposed model, topological constraints, metric constraints and statistical data types can be tailored into the epsilon band approach gracefully for capturing the data uncertainties and eliminating data inconsistencies. To justify this claim, the following algorithm is provided to handle the sliver problem:

Assume that socioeconomic layers are encoded in terms of POLYVRT structure. That is each layer consists of a collection of polygons, and each polygon in turn consists of a collection of curves. Moreover, a curve of length n is defined by a sequence $< v_s, p_1, \ldots, p_{n-1}, v_e >$, where v_s and v_e are vertices of the polygon, and $p_1, \ldots, p_e$ are points of the curve.

Assume that a sliver is detected between v_s and v_e. In other words, two alternative curves (i.e., $c_1 = <v_s, p_1, \ldots, p_i, v_e>$ and $c_2 = <v_s, q_1, \ldots, q_j, v_e>$) are found. Then the following three cases are handled as follows:

1) c_1*'s epsilon band lies inside* c_2*'s*. Remove c_2 from the map;

2) c_2*'s epsilon band lies inside* c_1*'s*. Remove c_1 from the map;

3) *Otherwise.* Apply both metric constraints and topological constraints to refine the measurements of the curves and to eliminate inconsistent curves. Record remaining curves as a statistical data type with many possible outcomes. Each takes the odds of the relative accuracy of the corresponding curve.

Figure 6.4 illustrates a situation in which traditional overlay algorithms indiscriminately remove sliver polygons, which results in disappearance of an entire region from the map. This problem is avoidable using the proposed scheme.

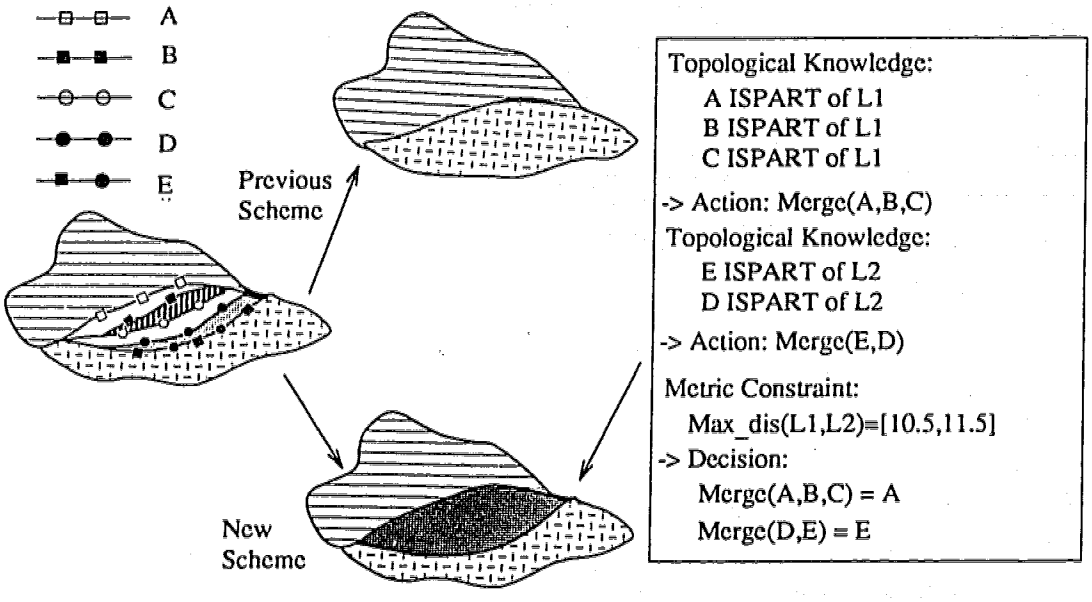


Figure 6.4: An Example of Geometric Error Handling

The following advantages are witnessed in the proposed algorithm: 1) it assimilates whatever is more accurate from the source layers; 2) it avoids the problem of unsafe snapping by using statistical data types for uncertainty representation; 3) it allows the curves to adjust their shapes

dynamically, according to new information.

Both metric constraints and topological constraints are necessary for resolving inconsistencies. Firstly, metric constraints must be used to refine the measurements of the some domain objects. Then topological constraints must be applied to refine the measurements of the other objects and to eliminate system inconsistencies.

Thematic Overlay Between Categorical Layers

In the area of attribute error handling, the statistical nature of the new model lends itself conveniently to modeling, quantifying and reasoning attribute uncertainties. As an example, the following algorithms will be proposed to handle the sliver problem with respect to overlaying of two categorical layers.

Assume the input of a thematic layer is encoded as a set of linear quadtree nodes ordered in ascending key order, with each node stored as a record consisting of four fields. The first two fields, termed KEY and RES, contain the key and the resolution parameter of the node, respectively. The third field, termed ID, identifies the class containing the node. The fourth field, termed PRO, is the probabilistic distribution function.

Let L_1 be a list of nodes representing a layer, and L_2 be a list of nodes representing another layer. The overlay of these two layers is obtained by the following procedure termed **Overlay**. Function Id-generator creates a new identifier for a new class in the overlaid map; function ∇ integrates the knowledge about the distribution of spatial phenomena from two different sources to yield a more authentic probabilistic distribution function about the reality. Function **Contains**(Q_1, Q_2) is used to determine whether node Q_2 is spatially contained in node Q_1.[4]

```
Function NewNode(Q1, Q2)
begin
  New-ID:=Id-generator(Q1.ID, Q2.ID);
  New-PRO:=∇(Q1.PRO, Q2.PRO);
  return <Q1.KEY, Q1.RES, New-ID, New-PRO>;
end.
```

4. In other words, whether the condition $Q_1.RES \geq Q_2.RES$ and $Q_1.KEY \leq Q_2.KEY \leq Q_1.KEY + 4^{**}Q_1.RES$ is satisfied, see [DW86].

```
Procedure Overlay(L1, L2)
begin
  R-list:={};
  Q1:=head(L1); Q2:=head(L2);
  while (L1 ≠ NIL and L2 ≠ NIL) do
    Case
      Q1.KEY = Q2.KEY and Q1.RES = Q2.RES:
        append NewNode(Q1, Q2) to R-list;
                                          {forward both lists.}
        L1:=tail(L1); Q1:=head(L1);
        L2:=tail(L1); Q2:=head(L2);
      Contains(Q2, Q1):
        append Newnode(Q1, Q2) to R-list;
                                          {forward list L1.}
        L1:=tail(L1); Q1:=head(L1);
      Contains(Q1, Q2):
        append Newnode(Q2, Q1) to R-list;
                                          {forward list L2.}
        L2:=tail(L2); Q2:=head(L2);
      Otherwise:
        if Q1.KEY < Q2.KEY then L1:=tail(L1); Q1:=head(L1);
        else L2:=tail(L2); Q2:=head(L2);
    end-of case;
  return(R-list);
end.
```

The following steps are recommended to handle the sliver polygon problem:

1) **Recognizing Slivers**: A sliver polygon can be determined by examining the shape of each connected component of the resultant linear quadtrees. Since a sliver polygon is small and narrow, it can be recognized from the quadtree description of the region, using features such as the area, the number of nodes, the maximum and minimum distance, etc.

2) **Sliver Removal**: Since the knowledge of the distribution of geographic phenomena is directly associated with each region, sliver removal is easy. A simple approach can assign the whole region to one of its neighbors, using a strategy such as maximum likelihood. More sophisticated methods can be applied to assign some portion of the region to one class and some other portion of the region to another class. The advantage of these methods is that they tend to yield a more accurate description of class boundaries. The disadvantage of these methods is the complexity of the classification algorithm. Since conventional algorithms treat each block independently, they have the tendency to yield many small fragments. Hence, spatial convolution operators must be applied to smooth the image. It is believed that the simple approach usually results a reasonable good approximation of the

real situation. More importantly, since the knowledge about the feature distribution is directly associated with each node, the structure is able to adapt its shape gracefully according to the new information. This is in sharp contrast to the epsilon band approach for geometric error handling.

The following analysis the time complexity of the proposed algorithm.

Suppose both L_1 and L_2 are main memory resident, the following statement is true:[5]

Theorem 6.3: The time complexity of overlay is bounded by $O_m(n(N_1 + N_2))$, where n is the resolution of the image, N_1 and N_2 are the number of blocks in L_1 and L_2 respectively.

Proof: Obviously, the while loop takes no more than $N_1 + N_2$ iterations. Again function **NewNode** costs a constant amount of execution time. Hence, the time complexity of procedure **Overlay** is bounded by $O_m(N_1 + N_2)$.

On the other hand, the time complexity of post-processing can be divided into four stages: labeling of connected components, statistical distribution calculation, sliver recognition and sliver removal. Since there can be no more than $N_1 + N_2$ nodes in the overlaid layer, the first stage takes on more than $n(N_1 + N_2)$ steps, see [DW86]. The second stage calculates the distribution function for each connected region that also requires less than $(N_1 + N_2)$ steps. The third stage involves the calculation of various geometric properties of a region, i.e., area, number of quadtree blocks, minimum/maximum distance, etc. According to the results in [DW86], these calculations are bounded by $O_m(n(N_1 + N_2))$. The fourth stage removes sliver polygons by assigning them to their neighboring classes. A typical classification process, i.e., the maximum likelihood approach, takes $O_m(S)$ steps, where $S < N_1 + N_2$ is the number of slivers in the overlaid layer.

In conclusion, thematic overlay takes $O_m(n(N_1 + N_2))$ steps.

Q.E.D.

Figure 6.5 illustrates the difference between a traditional overlay algorithm and the

5. Note: Otherwise, using a $B^+ - tree$, it is easy to show that the time complexity of overlay can be bounded by $O_e(n(N_1 + N_2)/f))$, where f is the fanout of a node, see [DW86].

proposed algorithm: the perfunctoriness of the former method results in a misclassification in the feature space, whereas, the assimilation of the latter method yields a more accurate classification in the feature space.

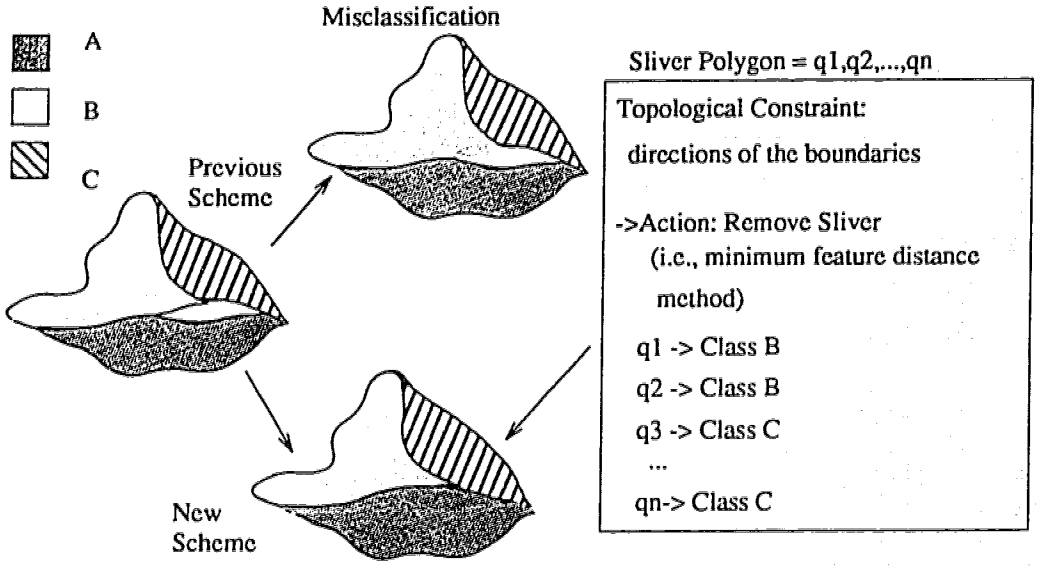


Figure 6.5: An Example of Attribute Error Handling

Functional Factor 2: Reasoning and Learning

The proposed model supports many desirable reasoning and learning capabilities. In particular, the issue of both spatial and attribute reasonings was previously studied in Chapter 4. In the following, the issue of learning will be addressed.

To date, two learning algorithms, inductive learning and deductive learning, have been reported [Pueq84]. Inductive learning is user driven and is accomplished by the user supplying positive and/or negative examples of new characteristics or relations. Deductive learning is accomplished by the storage and accumulation of information gained from previous queries. These algorithms can be easily extended in the new model. On the other hand, a major impediment to the progress of development of advanced geography learning procedures has been the lack of a suitable model for handling geographic errors. For instance, most existing GISs, except SPAM and MERCATOR, use Boolean logic or normal arithmetic for various operations. The use

of this logic implicitly assumes that both attributes and geometries are certain and precise and their combination leads to a single inferred conclusion. Without knowledge of inaccuracies inherent in the information contained in a GIS and the ways error can be multiplied by its most attractive capabilities, the system will run considerable risk of making erroneous distinctions. On the other hand, the main weakness of SPAM and MERCATOR for geographic data modeling in general is that they treat geographic errors strictly as a geometric issue. Thus, the attribute uncertainty as well as the taxonomy of complex objects is largely ignored.

Since the proposed model is both spatially-oriented and object-oriented, it opens a new avenue to prosperity of developing advanced geographic learning procedures. In the following, a new learning approach will be presented:

6.2.2.1 Maximum Entropy Approach

Maximum entropy inference is a method for updating a statistical distribution in the light of new information. The use of this approach is commonly justified by an information-theoretic argument: Given a set of statistical constraints, find the statistical distribution that satisfies the constraints but is also the least biased of all such distribution. To be the least biased is to minimize the amount of information contained in the distribution. Among the distributions satisfying the constraints there is a unique one, the maximum entropy distribution, that has minimum information. Hence maximum entropy inference selects the right distribution.

The maximum entropy approach is an important learning method adopted in the proposed model. In the previous chapter, the advantage of this approach for data integration was presented. In this section, this method will be used to calculate the most conservative, or unbiased, estimate of the probabilities given the constraints of the data model.

Given an object O, let $Q = \{q_1,\ldots,q_n\}$ be the actual, unknown probabilities of the attribute values $\{v_1,\ldots,v_n\}$ of O. Suppose that $P = \{p_1,\ldots,p_n\}$ is an initial estimate of Q, and suppose that new information about Q is obtained in the form of constraints that restrict P. A typical example of such constraints is a set of known expected values as follows:

$$\sum_{i=1}^{n} q_i f_{ki} = \overline{f_k}, \quad k=1,\ldots,m. \tag{6.1}$$

In order to determine the most conservative Q values for the statistical space consistent with the constraints, one must maximize the entropy, H, over the possible P values, where

$$H = -\sum_{i=n} p_i \log p_i.$$

This constrained optimization can be solved using Lagrange multipliers by forming the function H' that incorporates the constraints as follows:

$$H' = H + \sum_{k=1}^{m} \lambda_i (\overline{f_k} - \sum_{i=1}^{n} p_i \log p_i)$$

where λ_k, $k=1,\ldots,m$, are Lagrange multipliers to be determined, along with the value of p_i, $i=1,\ldots,n$ at the extreme point imposed by (6.1) and $\sum_{i=1}^{n} q_i = 1$ [KL86].

6.2.2.2 The Justifications of Maximum Entropy Inference

Although AI emphasizes symbolic processing, numerical information processing will always have an important role, particularly where uncertain information is involved. In this context, maximum entropy inference should have a role to play. In particular, there are two areas where maximum entropy inference are shown to be more appropriate.

The first area arises from maximum entropy's properties as an information measure - it should be useful as a means of quantifying information gains and losses within probabilistic inference procedures. A related area of possible application is the problem of searching large statistical state spaces, where it is necessary to exploit partial information about some optimal path. To solve this problem efficiently, it is necessary to use information measures to guide a heuristic search.

The second area arises from maximum entropy's properties as a method for minimum-information updating - it should be useful as a means of data integration and revision in the light of new information. According to the philosophy of maximum entropy inference, maximum relative inference must be considered as a way of passing from one statistical distribution to another.

In this sense, maximum entropy inference addresses only the dynamic problem and not the static one. The strength of maximum entropy inference relies on the fact that it builds its foundation on a dynamic environment. It also does not require the assumption of independent evidence. Since geographic data are highly dynamic and geographic phenomena are interrelated by complex relationships, the importance of this approach to GIS is evident.

Some of the putative rivals of maximum entropy inference such as Bayesian inference do not challenge it directly because they primarily address the static problem. The max-min rules for combining certainty factors in the inference system of MYCIN [KL86], for example, are really rivals of standard probability theory, not of maximum entropy. To the extent that these nonstandard theories address the dynamic component, the divergence from maximum entropy is often quite small. In the inference system of MYCIN, for example, when the evidence is known with certainty, updating reduces to conditionalization on the evidence (i.e., Bayesian inference). When the evidence is not certain the certainty factors of the hypothesis on the basis of the evidence is multiplied by the certainty factor of the evidence to yield the final degree of belief in the hypothesis, that gives a rough approximation to the result of maximum entropy inference. Furthermore, many static-based inference approaches including Bayesian inference require the assumption of independent evidence.

6.3 Potential Problems of the Proposed Model

The proposed model, however, is by no means a panacea in general. It also introduces some overhead such as the extra storage requirements and the tendency of being awkward, etc., as previously analyzed. And with some thought more problems will undoubtably appear.

6.3.1 Structural Considerations

The most controversial element is the linkage between spatial components and aspatial components. The proposed approach is to split the two at the spatial-to-aspatial juncture. Such a design principle is adequate for large and complex relations. However, it also creates a gap that must be hurdled every time a combined spatial and aspatial query is executed. In fact, for simple

and small relations, this approach may do more harm than good. Therefore, database schemas that result in indiscriminate partition of these two components at all levels for all relations should be discouraged. To achieve the best performance, it is necessary to carefully choose when and where a separated component is required.

The second important consideration is the explicit structure versus the implicit structure tradeoff. No one disagrees that both topological structures and geometric structures are boons to spatial analysis, but there is a controversy as to the manner in which the system maintains such structures. The proposed model allows two logical alternatives: the explicit structure approach (the structure is precomputed and explicitly stored) and the implicit structure approach (the structure must be computed on the fly). The first approach improves the search efficiency at the expense of both extra storage and maintenance cost, whereas, the second approach is less efficient in the query process, however, no extra storage and maintenance is needed. In practice, the proposed model can suffer from cumbersomeness if the two issues are not carefully dealt with. For instance, it is neither possible to derive all topological relations from primitive representations that admit data errors, nor feasible to store all potential relationships and scales.

Finally, although the use of a multi-variant logic at the attribute level provides a more flexible environment, it necessitates extra storage and sophisticated processing. Since, in practice, many attributes are single-valued, it is suggested that a hybrid scheme be adopted.

6.3.2 Limitation on Reasonings

Reasoning in the proposed model is mainly focused on combining multiple sources of collected data into a single environment appropriate for further manipulation, and performing inference in terms of statistical distributions. The merge process is able to recognize when two sources are in fact different representations of the same feature, separated by the collection method, statistically reasonable error, and to combine these multiple representations in a statistically valid compromise. The inference process is able to reason and quantify information within a multiple inheritance system, and to improve data representation in the light of new information.

The characteristics of spatial data necessitates that inference problems be posed a variety of uncertain and fuzzy knowledge. A number of logics for handling these problems have been proposed, including *statistical logic*, *default logic*, and *fuzzy logic*. The proposed model is based on the notion of statistic logic. It does not provide a complete answer to the inference problems, neither are either of the last two methods. But situations will be elaborated in which either default logic or fuzzy logic can serve as useful tools.

6.3.2.1 Default Logic Reasoning

Rarely does a system have at its disposal all the information that would be useful. But often when such information is lacking, there are sensible guesses that can be made, as long as no contradictory evidence is present. The construction of these guesses is known as *default reasoning*. Default reasoning is extremely important in an object-oriented paradigm because it allows a concise statement of the properties of the objects in the object hierarchy. Properties can be associated with the most general object for which they are valid. Fairly simple inference mechanisms can then be used to derive those properties for more specific objects. In a wildlife management application for example, concise statements can be made using general rules such as "a bird flies" and "mammals bear children", etc. These statements, like any other properties, can be explicitly stored in and derived from an ISA hierarchy.

Unfortunately, sometimes objects in hierarchies do not inherit all of the properties specified by their ancestors. There is, however, an exception to every rule. For example, although most mammals bear their young live, the platypuses lay eggs. Similarly, although the majority of birds fly, penguins do not fly. The most common approach in dealing with exception uses an inferential distance method. With this approach, the search for a particular piece of information in an ISA hierarchy must begin at the level representing the most specific concept involved and only move up the chain toward more general concepts if the specific information is absent. In other words, properties are inherited from their "closest" ancestor in the ISA hierarchy.

In a multiple inheritance system, the concept of closest ancestor is often meaningless. Conflicts arising from derivations must be resolved judiciously. The proposed model misses a

mechanism for handling default reasonings with exceptions. As the result, the task of drawing inference using default reasoning is left to the application program. On the other hand, since object hierarchies are embedded in the database management system, it is more appropriate to perform default reasoning inside the database system.

6.3.2.2 Fuzzy Logic Reasoning

Many spatial relationships are intrinsically fuzzy. Distance, direction, and connection are common geographic concepts that can be stated in imprecise linguistic variables such as "short", "north of", and "near", etc. These concepts may not be adequately represented by statistical logic as defined in the proposed model. They are better expressed as asymptotic membership functions using fuzzy logic. Previous work in [Robin87] demonstrated the significant impact on storing and manipulating spatial objects that is gained by using fuzzy logic. It is expected that the proposed model can be extended to handle fuzzy concepts as well. The major conceptual change necessary for applying statistical theory to typical fuzzy situations is to interpret probabilities as a measure of belief in the relevant proposition rather than a long run frequency. The fuzzy approach seemed to arise because the frequency interpretation unnecessarily restricts the domain of applicability of statistics. To this end, a formal theory for generalization, deduction and uncertainty manipulation with respect to both statistical data and fuzzy data is mandatory. The proposed model, however, as in its current form does not naturally fit in a fuzzy logic framework.

6.3.3 Drawbacks of R-trees

Although the R-tree based index scheme adopted in the proposed model has many appealing advantages, it is not without problems. For instance, most systems require concurrent access to a file. Concurrency control is complicated in an R-tree based structure because the root is a bottleneck shared by all access paths. If a process has the potential of modifying the data structure near the root (such as insertion or deletion in an R-tree), other processes may be slowed down by the adherence to locking protocols even if they access disjoint data. Another problem with the structure is deterioration in performance during and after update. To prevent performance

deterioration, any modification made to the primitives of an object necessitates a reinsertion in the R-tree; furthermore, the deletion of an object in the region may involve many reinsertions to take care of underfull nodes in the R-tree. Thus, the maintenance of an R-tree can be difficult. As a result, the scheme is detrimental to the performance of a number of primitive operations such as intersection and overlay.

6.4 Conclusion

This chapter has justified the performance of the new model in terms of time efficiency, space requirements, and functionalities. First of all, efficient principal algorithms such as query and update have been developed to support the proposed model. Both analytical and simulation results demonstrate that the new model is quite efficient and suitable for large geographic data processing. Secondly, the proposed model has been shown to be quite efficient in terms of space requirements as well. Particularly, its ability of providing a high degree of data sharing, with respect to both attribute and geometric descriptions, labels it an excellent scheme for geographic data applications. Finally, the model also facilitates a number of desirable functionalities such as data integration, reasoning, learning, and system development.

Chapter 7

Conclusion

7.1 Contributions

This thesis is concerned with the organization of spatial data for geographic information systems. To this end, a comprehensive survey of state-of-the-art GIS technologies has been presented. From the survey, problems associated with various approaches are identified. In particular, throughout the discussion, the lack of systematic and accepted data models for spatial data handling, including retrieval, reasoning, and adaptive capabilities, is emphasized.

Spatial data handling has much to do with the representation of geographic knowledge. Search efficiency depends in large measure on the organization of the database. Spatial reasoning calls for the use of a DBMS that is integrated both spatially and semantically in its organization. Learning and efficient problem solving lies heavily on the accumulation and interpretation of geographic knowledge. The goal of this thesis is to develop a novel model, called a cognitive map model, for GIS. The focus of this research includes aspects of the developments of database management, artificial intelligence (AI), and uncertainty which promise new advancements in geographic information systems. On the one hand, the nature of both A.I. and uncertainty encourage the representation and gradual accumulation of geographic knowledge. On the other hand, the provision of a DBMS not only makes use of geographic data more efficiently (in terms of both time and space complexity) but also more effectively (in terms reliability and sharability). This thesis bears witness to the cross-fertilization of these two components. As a result, the following three major contributions are made:

- **The Notion of Spatial-orientation.** Spatial-orientation views the geographical world in a manner similar to the way humans view the physical world. Its philosophy and implementation will have a long term affect on the GISs development. In particular, the concept of dynamic spatial objects, spatial dependency, spatial inheritance, and generalization, as well as the importance of the separation of spatial components from nonspatial components will undoubtly play an important role in future GISs.

- **Uncertainty and Error Modeling and Processing.** The thesis developed a novel approach for geo-data error and uncertainty handlings. The strength of this model is that it copes with various complicated geo-data inconsistencies (such as dangling edges, self-crossing contours, and sliver polygons, etc) that labels it as an excellent tool for GISs. More importantly, the distinct feature of the proposed model is its learning capabilities. Two learning methods have been proposed to support this claim: the constraint based learning and the maximum entropy method.
- **The Four-level Semantic Hierarchy.** The proposed semantic structure integrates aspatial feature modeling, spatial feature modeling and geo-data error and uncertainty handling into a general framework for geographical data modeling and processing.

7.1.1 Justification of Coherency

In the proposed model, several important concepts, such as object-orientation and spatial-orientation, semantic inheritance, uncertainty information modeling, and database management, all fit together in a clean theory of the cognitive map.

First of all, intrinsic advantages of four levels of semantic hierarchy are emphasized. The structure not only conveniently lends itself to database implementation, but also provides the cognitive map with rich semantics for modeling various geographic concepts, including shape, location, place, object, interrelationship including: topological, object and metric. Moreover, it enables the specification of complex application-specific operations and semantic integrity and consistency constraints at different levels of abstraction, and supports a complex reasoning processes metaphorically similar to human problem solving approaches. At each of these levels, it has something to contribute: At the first level (the metric level), a hierarchically organized spatial organization provides for efficient search space reduction by supporting both top-down and spatially-windowed search. At the second level (the object level), a friendly user-oriented interface offers the system with both spatially-oriented and object-oriented views of the data. At the third level (the attribute level), the notion of random observable is used to represent spatial objects. In this way, the model takes advantage of ideas from both the theory of uncertainty and

the theory of relational databases to achieve both flexibility and reliability in its representation. At the fourth level (the primitive level), efficient data structures and file structures are adopted to represent shape and location aspects of the spatial objects.

Secondly, the separation of the object level into two parts, the conceptual level and the topological level, is advocated. In this way, the model avoids the dilemma of object-orientation vs. spatial-orientation. This is because the conceptual level together with the attribute level offers the ability to describe, organize, search and reason information about objects independent of the location of the objects. Whereas, the topological level, together with the primitive level, the attribute level, and the metric level, supports various spatially-oriented operations efficiently by taking advantage of semantic hierarchy of the spatial object organization.

7.1.2 Justification of Consistency

Inconsistency refers to anomalies between the real world and the data model. In a database management system, inconsistencies come either directly from the source data or from various crude and careless use of data manipulations such as insertion, deletion and update. Generally speaking, there are two types of anomalies: structural anomalies and measurement anomalies. The proposed model not only finds facilities to prevent structural anomalies from being stored in a database, but also finds mechanisms to accommodate and constrain measurements anomalies.

7.1.2.1 Measurement Anomalies

A distinct feature of the proposed model is its tolerance of data errors. Two related error models are defined in the thesis: the first is an epsilon band error model and the second is a statistical error model. More importantly, although measurement anomalies are allowed in the proposed model, they are bounded at various abstraction levels in the sense that certain kinds consistent constraints, (i.e., the scale of precision, the geometric constraints, and the attribute constraints) can be checked by the DBMS.

7.1.2.2 Structural Anomalies

Structural anomalies can be resulted in two ways: 1) malicious use of insertion, deletion, and update in the database; 2) malicious treatments of measurement anomalies that eventually lead to structural anomalies. In the proposed model, the issue of protecting the database against mis-use is well studied. In particular, the following measures are taken:

1) Functional dependency - a set of structural constraints that can be enforced to preserve data integrity in the events of insertion, deletion, and update operations;

2) Semantic constraints on measurements - a set of constraints that restrict either at the metric level or at the topological level. Semantic constraints on measurements are used not only for integrity preservation but also for learning as well.

7.1.3 Justifications of Efficiency

The primary objective of this thesis is to develop an efficient data model in terms of both time and space requirements. In reality, however, any scheme is the result of a tradeoff between time and space and the new scheme is no exception.

7.1.3.1 Time Efficiency

The following three measures have been adopted to improve the time efficiency:

7.1.3.1.1 Structural Considerations

Cognitive map model places great emphasis on the notion of separating the structure of spatial data from nonspatial data, while maintaining appropriate links between the two. This approach provides the model an important focus-of-attention capability on the one hand, and increases the retrieval bandwidth on the other hand.

Another major contribution is that it explores the use of various index files of an integrated spatial representation that provides a 'window' into an object-oriented paradigm. Both an R-tree and a B-tree are adopted to improve search efficiency by narrowing search spaces.

7.1.3.1.2 A Lattice Model for Spatial Objects

The majority of geographic data are map coverages with one feature per theme. Nevertheless, most of the queries involve multiple features. There are two opposite ends of a broad spectrum of possible data-base schemes involving various levels of integration: the separated layer scheme and the fully integrated scheme.

At one extreme, data is represented as a collection of thematic layers, merging only at search time. Since integration of thematic layers involves many complex geometric computations, the model is inefficient.

At the other extreme, on the other hand, all layers are merged into a single geometric structure. The disadvantage of this approach is that it tends to indiscriminately decompose objects into many small regions. Although the scheme avoids the issue of data integration, the absence of the ability to index spatial objects at a higher level of abstraction is a significant disadvantage. For instance, the problem of finding all objects contained in a given rectangular area requires an elaborate reconstruction process of objects from many small regions.

A lattice model for spatial objects tries to strike a balance between the above mentioned two extremes. With this model, themes that tend to share a great deal of geometry can be integrated at a master database layer to prevent a high initial access time for a merge process; themes which show little spatial correlation can be represented at different layers to avoid producing many small regions and to improve the efficiency of the high level retrieve processes.

7.1.3.1.3 Aggregate Response

Conventional responses in database systems, usually given as a list of atomic objects, although sufficient to serve the purpose of conveying information, do not necessarily provide efficient and effective communications between a user and the system.

In the proposed model, the presence of a class hierarchy and uncertainty, an integral part of the proposed model, allows answers to be expressed implicitly in terms of classes and instances. This enables answers to be provided at various levels with different accuracies.

7.1.3.1.4 Geometric Generalization

Merging geographical data involving spatial search occurs at the primitive level. Spatial search at the primitive level severely obstructs the efficiency of most existing GISs due to the huge amount of primitive data in these systems. To overcome this problem, in the proposed model, spatial objects are first systematically mapped into a metric space. Furthermore, both data abstraction and geometric generalization processes are applied to this step. As the result, a much smaller and simplier primitive structure can be obtained.

7.1.3.2 Storage Requirements

Clearly, extra storage expenditures are necessary for incorporating statistical data values, epsilon-bands, geometric entities with different scale and precision. Nevertheless, the tradeoff between time efficiency and storage requirements is cost effective.

The justifications for the storage requirements are as follows:

1) The extra space expenditure is considered to be reasonable due to the fact that geometric generalization is capable of drastically decreasing query processing time at a moderate storage cost.

2) The model lends itself to geometric data sharing. By geometric data sharing, is meant that features with the same geometric description share the same storage structure. A lattice model suffices for the needs of geographic data sharing. With this model, knowledge about where, and to what extend, an integrated layer is needed and the relationship between different layers can be directly encoded.

7.2 Future Work

There are many problems remain to be done. The following will briefly mention a few of the areas that the current work will have potential impact on:

Heterogeneous System: Geographical information systems should be built on a distributed base. Distributed GISs have several potential advantages, e.g., extensibility, availability, and

performance. There are however, several difficult problems need to be solved before the advantages of distributed systems can be realized. The proposed model is a mixed blessing in this regard. On the one hand, the model has the potential of improving both processing efficiency and communication throughput of the network. On the other hand, currently, there are many geographical information systems based on traditional data models. These represent an extremely valuable resource and a major investment. Therefore, in the years to come these systems will keep increasing their share of the market. There are many difficulties associated with a distributed database system consisting of heterogeneous subsystems (i.e., data consistency, data format, etc) To solve these problems, a technique called *an SQL gateway* is often adopted. Unfortunately, there is a general lack of standards for exchange data between an object-oriented database and a traditional model. With the cognitive model, the problem seems to be even more critical. Hence, more research is clearly needed in this area.

Heterogeneous Data Handling: As mentioned previously, one of the important characteristics of spatial data is that it contains many diverse data types. Although the proposed statistical logic is able to model statistical data types, fuzzy inference and default reasoning are sometimes shown to be more appropriate. For instance, it is important to point out that many spatial relationships are intrinsically fuzzy. Distance, direction, and connection are common geographic concepts that can be stated as imprecise linguistic variables such as "short", "north of", and "near" etc. In addition, all geographical information that is spatial in nature is obtained from physical measurement. All physical measurements are by their very nature imprecise. These concepts may not be adequately represented by either threshold values or statistical distributions. They may be better expressed as asymptotic membership functions using fuzzy logic. Previous work in [Robin87] demonstrated the significant impact on storing and manipulating spatial data gained by using fuzzy logic. It is expected that the proposed model can be extended to handle fuzzy concepts as well. To this end, a formal theory for generalization, deduction and uncertainty manipulation with respect to both statistical data types and fuzzy data types is mandatory. This opens a new avenue for further research. Moreover, to support default reasoning, a nonmonotonic logic based on multiple

inheritance systems must be implemented.

Spatial Data Modeling: Spatial data modeling is one of the toughest problems in GIS. To combat this problem, the notion of semantic hierarchy of spatial objects is proposed. It should be made clear that currently, there exists a paucity of research that addresses digital generalization in a holistic manner. To this end, the concept of approximate geometric enclosure needs to be generalized to deal with point and curve, and efficient algorithms should be developed that can produce an approximate geometry of an object at any prespecified accuracy. In addition, the algorithm should be able to look at the interrelationships between the conditions that indicate a need for its application, the objectives or goals of the process, as well as the specific spatial and attribute transformations required to effect the changes.

Bibliography

[Abel 84] D. J. Abel, "A B+ tree Structure for Large Quadtrees", *Computer Vision, Graphics, and Image Processing*, Vol. 27, 1984, pp. 19-31.

[Abel 92] D. Abel, et. al.,"Environmental Decision Support System Project: an Exploration of Alternative Architectures for GISs", Int'l. Journal of GISs, Vol.6, No.3, 1992, pp.247-256.

[Abler 87] R. F. Abler, "The National Science Foundation National Center for Geographic Information and Analysis", *Int'l. Journal on Geographic Information Systems*, Vol.1, No. 4, 1987, pp. 303-326.

[Allen 90] K. Allen et. al., "Interpreting Space: GIS and Archaeology" S. Green and E. Zubrow (eds.). London: Taylor and Francis, 1990.

[Alba 90] A. Albano, et. al., "A Relationship Mechanism for a Strongly Typed Object-Oriented Database Programming Language", Proc. of the 17th Int'l Conf. On Very Large DataBases, 1991, pp. 565-576.

[Anto 87] R. Antony, "Spatial Reasoning Using an Object-Oriented Spatial DBMS", *In Workshop on Spatial Reasoning and Multisensor Fusion*, Los Angles, USA, Mar. 1987, pp. 42-51.

[AM 83] P. Aronson and S. Morehouse, "The ARC/INFO Map Library: A Design for a Digital Geographic Database", *Proc. Auto-Carto 6*, Ottawa, Canada, 1983, pp. 372-382.

[Ban 88] F. Bancilhon, "Object-Oriented Database Systems", *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1988, pp. 152-162.*

[Baum 88] M. Baumgardner, "A Global Scale Soils and Terrain Database", Building Databases for Global Sciences, H. Mounsey and R. Tomlinson (eds), Taylor and Francis, Landon, pp.172-180.

[Beck 90] N. Beckmann et. al., "The R^*-tree: An Efficient and Robust Access Method for Points and Rectangles", *ACM SIGMOD Conference on Management of Data*, Atlantic, NJ, USA, May 1990, pp.322-331.

[Bick 88] D. Bickmore,"World Digital Database for Environmental Sciences", Building Databases for Global Sciences, H. Mounsey and R. Tomlinson (eds), Taylor and Francis, Landon, pp.181-191.

[Blak 84] M. Blakemore, "Generalization and Error in Spatial Databases", Cartographica, Vol.21, 1984, pp.131-139.

[Brig 88] D Briggs, "CORINE: An Environmental Information System for the European Communiy", European Environmental Review, Vol.2, No.1,1988, pp.29-34.

[Bun 87] M. S. Bundock, "An Integrated DBMS Approach to Geographical Information Systems", *Proc. Auto Carto 8*, Baltimore, Maryland, 1987, pp. 292-301.

[CV 90] S. Cabay and L. Vanzella, "A Comparison of Partitioned and Seamless Organizations of Spatial Data", Proc. of GIS'90 Symposium, Vancouver, British Columbia, Mar. 1990, pp.423-429.

[Cam 90] W. Campbell, et. al.,"Intelligent Information Fusion for Spatial Data Management", Proc. of 4th Int'l. Symposium on Spatial Data Handling, Vol.2, Zurich, Switzerland, 1990, pp.567-577.

[CP 87] R. Cavallo and M. Pittarelli, "The Theory of Probabilistic Databases", *ACM SIGMOD Conference on Management of Data*, Brighton, 1987, pp. 71-81.

[CK 89] R. Cavallo and G. Klir, "Reconstructability Analysis: Evaluation of Reconstruction Hypothesis", *Int. J. of General Systems*, Vol. 7, No. 1, 1989, pp. 7-32.

[Chang 77] S. K. Chang, et. al., "A Relational Database System for Pictures", *Proc. IEEE Workshop on Picture Data Description and Management*, Chicago, Illinois, 1977, pp. 142-149.

[CK 81] S. K. Chang and T. L. Kunii, "Picture Data-Base Systems", *IEEE Computer*, Vol. 14, 1981, pp. 13-21.

[Chris 87] N. Chrisman, "The Accuracy of Map Overlays: a Reassessment", Landscape and Urban Planning, Vol.14, 1987, pp.427-439.

[Comer 79] D. Comer, "The Ubiquitous B-tree", *ACM Computing Surveys*, Vol. 11, 1979, pp. 121-137.

[Cook 78] B. Cook, "The Structural and Algorithmic Basis of a Geographic DataBase", Proc. First Int'l Advanced Study Symposium on Topological Data Structures for GISs", Vol.4, 1978, Harvard University, pp.507-522.

[CM 89] I. Crain and C. MacDonald,"The Canada Geographic Information Systems for Natural Environmental Reporting", Proc. CISM, Ottawa, 1989, pp.53-60.

[Davis 86] E. Davis, *Representing and Acquiring Geographic Knowledge*, Pitman London, Morgan Kaufmann Publishers Inc., 1986.

[Davis 84] W. A. Davis, "Data Formats for Geographic Databases", *Final Report for Alberta Energy and Natural Resources*, University of Alberta, Edmonton, Canada, 1984.

[DH 86] W. A. Davis and C. H. Hwang, "Organizing and Indexing of Spatial Data", *Proc. Second International Symposium on Spatial Data Handling*, Seattle, Washington, 1986, pp. 5-14.

[DX 86] W. A. Davis and Xiaoning Wang, "Connected Component Labeling Using Modified Linear Quadtrees", *Proceedings Graphics Interface'86*, Vancourver, May 1986, pp.235-240.

[DS 73] R. M. Downs and David Stea, *Cognitive Maps and Spatial Behavior: Process and Products*, Aldine Publishing Co., 1973.

[EF 89] M. Egenhofer and A. Frank, "Object-Oriented Modeling in GIS: Inheritance and Propagation", *Proceedings of Nineth International Symposium on Computer-Assisted Cartography*, Baltimore, Maryland, Apr. 1989, pp. 588-598.

[Egen 88] M. Egenhofer and A. Frank, "Towards a Spatial Query Language: User Interface Consideration", *Proc. 15th International Conference on Very Large Databases*, Los Angles U.S.A., Aug. 1988, pp. 124-133.

[Egen 92] M. Egenhofer,"Why not SQL?", Int'l. Journal of GISs, Vol.6, No.2, 1992, pp.71-85.

[Etz 93] O. Etzion,"PARDES: A Data-Driven Oriented Active Database Model", ACM SIGMOD Record, Vol.22, No.1, March 1993, pp.7-14.

[Falo 87] C. Falousos, et. al., "Analysis of Object Oriented Spatial Access Methods", Proc. ACM SIGMOD Conference on Management of Data, San Francisco, May 1987, pp.426-439.

[Frank 84] A. U. Frank, "Extending a Network Database with Prolog", First Int'l Workshop on Expert Database Systems, Kiawah Island, SC, 1984, pp.121-130.

[Frank 91] A. U. Frank, "Properties of Geographic Data: Requirements for Spatial Access Methods", Proc. of 2nd Symposium, SDD'91, Zurich, Switzerland, Aug. 1991, pp.225-236.

[Gadia 93] S. K. Gadia, "Parametric Databases: Seamless Integration of Spatial, Temporal, Belief and Ordinary Data", ACM SIGMOD RECORD, Vol.22, No.1, March 1993, pp.15-20.

[Ghosh 88] S. Ghosh, et. al., "Implementation of a Prolog-Ingres Interface", ACM SIGMOD Record,

Vol.17, No.2, June 1988, pp.77-88.

[Good 92a] M. Goodchild, et. al.,"Integrating GIS and Spatial Data Analysis: Problems and Possibilities", Int'l. Journal of GISs, Vol.6, No.5, 1992, pp.407-423.

[Good 92b] M. Goodchild et. al.,"Development and Test of an Error Model for Categorical Data", Int'l. Journal of GISs, Vol.6, No.2, 1992, pp.87-104.

[Greer 92] R. Greer,"DataShare and Fourth Generation Language Cymbal", Proc. AT&T Database, Sep. 1992, pp.12-21.

[Gupta 91] A. Gupta, "An Extended Object_Oriented Data Model for Large Image Bases", Proc. of 2nd Symposium, SDD'91, Zurich, Switzerland, August 1991, pp.45-62.

[Gunt 89] O. Gunther, "The Design of the Cell Tree: An Object Oriented Index Structure for Geometric Databases", Proc. IEEE 5th Int'l on Data Engineering, Los Angeles,1989, pp.598-605.

[Gutt 84] A Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching", *Proc. ACM SIGMOD Conference on Management of Data*, Boston, Mass., 1984, pp. 47-57.

[Hart 73] R. A. Hart and G. T. Moore, "The Development of Spatial Cognition: A Review", *in R. M. Downs and D. Stea (eds) Image and Environment*, Aldine Publishing Company, 1973.

[Haas 91] L. Haas and W. Cody, "Exploiting Extensible DBMS in Integrated Geographic Information Systems", Proc. of 2nd Symposium, SSD'91, Zurich, Switzerland, Aug. 1991, pp. 423-450.

[Jaga 93] H. Jagadish,"Database Research at AT&T Bell Laboratories", ACM SIGMOD Record, Vol.22, No.1, March 1993, pp.82-88.

[Kain 88] W. Kainz, "Application of Lattice Theory to Geography", Proceedings of International symposium on Spatial Data Handling, Sydney, Australia, Aug. 1988, pp. 135-142.

[KL 86] L. N. Kanal and J. F. Lemmer, *Uncertainty In Artificial Intelligence*, Elsevier Science Publishers B. V., North-Holland, 1986.

[KM 90] A. Kemper and G MoerKotte, "Access Support in Object Bases", *ACM SIGMOD Conference on Management of Data*, Atlantic City, NJ, USA, May 1990, pp.364-374.

[Kifer 89] M. Kifer, "F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and

Scheme", *Proc. ACM SIGMOD Conference on Management of Data*, 1989, pp. 134-146.

[KW 89] M. Kifer and J. Wu, "A Logic for Object-Oriented Programming", *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Mar. 1989, pp.379-393.

[Kim 89] W. Kim, "A Model of Queries for Object-Oriented Databases", *Proceedings of the 15th International Conference on Very Large Databases*, Amsterdam, The Netherland, Aug. 1989, pp.3-14.

[Kuip 78] B. Kuipers, "Modeling Spatial Knowledge", *Cognitive Science*, Vol. 2, 1978, pp. 129-153.

[KL 88] B. Kuipers and T. Levitt, "Navigation and Mapping in Large-Scale Space", *AI Magazine*, Summer, 1988, pp. 25-43.

[Kull 59] S. Kullback, *Information Theory and Statistics*, Wiley, New York, 1959.

[Lamb 91] C. Lamb, et. al.,"The ObjectStore Database System", Comm. of ACM Vol.34 No.10, Oct. 1991, pp.51-63.

[LM 83] R. A. Lorie and A. Meirer, *Using A Relational DBMS for Geographical Databases*, Computer Science Research Report 3848 (43915), IBM Research Laboratory, San Jose, CA, 1983.

[Lorie 91] R. A. Lorie,"The Use of A Complex Object Language in Geographic Data Management", Proc. of 2nd Symposium, SDD'91, Zurich, Switzerland, August, 1991, pp.319-337.

[Maff 87] G. Maffini, "Raster Versus Vector Data Encoding and Handling: A Commentary", *Photogrammetric Engineering and Remote Sensing*, Vol. 53, 1987, pp. 1397-1398.

[MM 89] F. M. Malvestuto and M. Moscarini, "Aggregate Evaluability in Statistical Databases", *Proceedings of the 15th International Conference on Very Large DataBases*, Amsterdam, The Netherlands, Aug. 1989, pp.279-286.

[McD 78] "Assimilation of New Information by Natural Language Understanding", Cognitive Science, Vol.2, No3, 1978, pp.120-134.

[MO 87] F. Manola and J. A. Orenstein, "Geographic Information Processing in the Probe Database System", *Proc. 8th International Conference on Computer-Assisted Cartography*, Baltimore, Maryland, Mar. 1987, pp. 316-325.

[MF 89] D. M. Mark and A. U. Frank, "Concepts of Space and Spatial Language", *Proceedings of Nineth*

International Symposium on Computer-Assisted Cartography, Baltimore, Maryland, Apr. 1989, pp. 538-556.

[Marx 86] R. W. Marx, "The TIGER System: Automating the Geographic Structure of the United States Census", *Government Publications Review*, Vol. 13, 1986, pp. 181-201.

[More 85] S. Morehouse, "ARC/INFO: A Geo-Relational Model for Spatial Information", *Proc. Auto Carto 7*, Washington, D.C., 1985, pp. 388-397.

[Motro 90] A. Motro and Q. Yuan, "Querying Database Knowledge",*ACM SIGMOD Conference on Management of Data*, Atlantic City, NJ, USA, May 1990, pp.173-183.

[NS 93] P. Nijkamp and H. Scholten, "Spatial Information Systems: Design, Modeling and Use in Planning", Int'l. Journal of GISs, Vol.7, No.1, 1993, pp.85-96.

[Oren 86] J. A. Orenstein, "Spatial Query Processing in an Object-Oriented Database System", *Proc. ACM SIGMOD Conference on Management of Data*, Washington, D.C., 1986, pp. 326-336.

[Oliv 89] A. Olive, "On the Design and Implementation of Information Systems from Deductive Conceptual Models", *Proceedings of the 15th International Conference on Very Large DataBases*, Amsterdam, The Netherland, Aug. 1989, pp.3-14.

[Paul 85] C. R. Paul, *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach*, Pitman Publishing Inc., 1985.

[Peuq 84] D. J. Peuquet, "Data Structures for a Knowledge-Based Geographic Information System", *Proc. International Symposium on Spatial Data Handling*, Vol. 1, 1984, pp. 372-391.

[Peuq 84] D. J. Peuquet, "A Conceptual Framework and Comparison of Spatial Data Models", *Cartographica*, Vol. 21, 1984, pp.66-113.

[Perk 56] J. Perkal, "On epsilon length", Bulletin de l'Academie Polonaise des Sciences, Vol.4, 1956, pp.399-403.

[Pere 82] M. Pereira et. al., "ORBI - An Expert System for Environmental Resource Evaluation through Natural Language", Report FCT/DI-3/82, 1982.

[Rad 91] F. Radermacher, "The Importance of MetaKnowledge for Environmental Information Systems",

Proc. of 2nd Symposium SDD'91, Zurich, Switzerland, Aug. 1991, pp.35-44.

[Robin 87] V. B. Robinson, "A Model of Error For Choropleth Maps, with Applications to Geographic Information Systems", *Auto Carto'8*, Baltimore, Maryland, Mar. 1987, pp. 165-174.

[Samet 90a] H. Samet, "Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS", Addison-Wesley, Reading, MA, 1990.

[Samet 90b] H. Samet, "The Design and Analysis of Spatial Data Structures", Addison-Wesley, Reading, MA, 1990.

[Samet 87c] H. Samet, et. al., "Recent Developments in Linear Quadtree-Based Geographic Information Systems", *Image and Vision Computing*, Vol. 5, 1987, pp. 187-197.

[See 88] B. Seeger and H. P. Krilgel, "Techniques for Design and Implementation of Efficient Spatial Access Methods", *Proc. 15th International Conference on Very Large Databases*, Los Angles, U.S.A., Aug. 1988, pp. 124-133.

[See 91] B. Seeger, "Performance Comparison of Segment Access Methods Implemented on Top of the Buddy-Tree", Proc. 2nd Symposium, SSD'91, Zurich, Switzerland, August 1991, pp.277-298.

[SB 91] H. Siegelmann and B. Badrinath, "Integrating Implicit Answers with Object-Oriented Queries", Proc. of 7th Int'l Conf. on Very Large DataBases, Barcelona, Spain,1991, pp.15-24.

[Shan 48] C. E. Shannon, "The Mathematical Theory of Communication", *Bell System Technical Journal*, Vol. 27, 1948.

[SM 88] C. D. Shum and R. Muntz, "An Information-Theoretic Study on Aggregate Responses", *Proc. 14th International Conference on Very Large Databases*, Los Angles, California, Aug. 1988, pp. 479-490.

[ST 92] D. Smith and R. Tomlinson, "Assessing Costs and Benefits of GISs: Methodological and Implementation Issues", Int'l. Journal of GISs, Vol.6, No.3, 1992, pp.247-256.

[SP 84] T. R. Smith and M. Pazner, "Knowledge-Based Control of Search and Learning in a Large-Scale GIS", *Proc. International Symposium on Spatial Data Handling*, Vol. 2, 1984, pp. 498-519.

[Smith 87] R. Smith, et. al., "A Stochastic Map for Uncertain Spatial Relationships", *Workshop on Spatial Reasoning and Multisensor Fusion*, Los Angles, USA, Mar. 1987, pp. 105-114.

[SZ 91] P. Svensson and H. Zhexue, "Geo-SAL: A Query Language for Spatial Data Analysis" Proc. 2nd Symposium, SDD'91, Zurich, Switzerland, August 1991, pp.119-142.

[Stra 90] D. Straube, *Query Optimization in Object-oriented Database Systems*, Ph.D. thesis, University of Alberta, Edmonton, Alberta, Canada, 1990.

[TS 80] P. W. Thorndyke and C. Stasz, "Knowledge Acquisition from Maps", *Cognitive Psychology*, Vol. 12, No. 1, 1980, pp. 137-175.

[Ullm 80] J. D. Ullman, *Principles of Database Systems*, Rockville, MD: Computer Science Press, 1980.

[Van 88] L. Vanzella, *Classification of Data Structures for Thematic Data*, M.Sc. Thesis, Dept. of Computing Science, University of Alberta, Alberta, Canada, 1988.

[VC 88] L. Vanzella and S. Cabay, "Hybrid Spatial Data Structures", Proc. Third International GIS/LIS Conference, San Antonio, Texas, pp. 360-372, 1988.

[Vieu 91] B. Vieux, "Geographic Information Systems and Non-Point Source Water Quality and Quantity Modelling", Hydrological Processes, Vol. 5, No. 7, 1991, pp. 101-113.

[Wang 90] Z. Wang, "A Representation Schema for Cartographic Information", Proc. 4th Int'l Symposium on Spatial Data Handling, Vol.2, Zurich, Switzerland, 1990, pp.782-791.

[WH 87] T. Waugh and R. Healey,"The GeoView Design a Relational Database Approach to Geographical Data Handling", Int'l. Journal of GISs, Vol.1, No.2, 1987, pp.101-118

[Wim 92] G. Wim,"the Vector to Raster Conversion: (mis)Use in GISs", Int'l. Journal of GISs, Vol.6, No.2, 1992, pp.159-170.

[Wor 90] M. Worboys, "The Role of Modal Logics in the Description of Knowledge in a GIS", Cognitive and Linguistic Aspects of Geographic Space. NATO ASI, 1990

[Zhou 90] X. Y. Zhou, *Heuristic Relaxation Approach for 3D Object Recognition*, Proceedings of IEEE international Conference on Robotics and Automation, Cincinnati, Ohio, U.S.A. May 1990, pp. 1852-1857.

[Zhou 88] X. Y. Zhou, *A Hybrid Structure for the Representation of Spatial Data*, M.Sc. Thesis, Dept. of Computing Science, University of Alberta, Alberta, Canada, 1988.