A PROGRESSIVE REFINEMENT APPROACH TO SPATIAL DATA MINING

by

Krzysztof Koperski
M.Sc., Warsaw University of Technology, 1988.

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
April 1999

APPROVAL

Name: Krzysztof Koperski

Degree: Doctor of Philosophy

Title of thesis: A Progressive Refinement Approach to Spatial Data Mining

Examining Committee: Dr. Robert Hadley
Chair

Dr. Jiawei Han
Senior Supervisor

Dr. Tiko Kameda
Supervisor

Dr. Tom Poiker
SFU Examiner

Dr. Max Egenhofer
External Examiner, University of Maine

Date Approved:

Abstract

Spatial data mining, i.e., mining knowledge from large amounts of spatial data, is a demanding field since huge amounts of spatial data have been collected in various applications, ranging from remote sensing to geographical information systems (GIS), computer cartography, environmental assessment and planning. The collected data far exceed people's ability to analyze it. Thus, new and efficient methods are needed to discover knowledge from large spatial databases.

The goal of this thesis is to analyze methods for mining of spatial data, and to determine environments in which efficient spatial data mining methods can be implemented. In the spatial data mining process, we use (1) non-spatial properties of the spatial objects and (2) attributes, predicates and functions describing spatial relations between described objects and other features located in the spatial proximity of the described objects. The descriptions are generalized, transformed into predicates, and the discovered knowledge is presented using multiple levels of concepts.

We introduce the concept of spatial association rules and present efficient algorithms for mining spatial associations and for the classification of objects stored in geographic information databases. A spatial association rule describes the implication of one or a set of features (or predicates) by another set of features in spatial databases. For example, the rule "*80% of the large cities in Canada are close to the Canada-U.S. border*" is a spatial association rule. A spatial classification process is a process that assigns a set of spatial objects into a number of given classes based on a set of spatial and non-spatial features (predicates). For example, the rule "*if a store is inside a large mall and is close to highways, then it brings high profits*" is a spatial classification rule.

The developed algorithms are based on the progressive refinement approach. This approach allows for efficient discovery of knowledge in large spatial databases. A complete set

of spatial association rules can be discovered, and accurate decision trees can be constructed, using the progressive refinement approach. Theoretical analysis and experimental results demonstrate the efficiency of the algorithms. The completeness of the set of discovered spatial association rules is shown through the theoretical analysis and the experiments show that the proposed spatial classification algorithm allows for better accuracy of classification than the algorithm proposed in the previous work [37].

An important proposed optimization technique used in the progressive refinement approach, is that the search for patterns at high conceptual levels may apply efficient spatial computation algorithms at a relatively coarse resolution. Only the candidate spatial predicates, which are worth detailed examination will be computed by refined spatial computation techniques. Such a multi-level approach saves computation effort because it is very expensive to perform detailed spatial computation for all the possible spatial relationships.

The results of the research have been incorporated into the spatial data mining system prototype, *GeoMiner*. *GeoMiner* includes five spatial data mining modules: characterizer, comparator, associator, cluster analyzer, and classifier. The SAND (Spatial And Nonspatial Data) architecture has been applied in the modeling of spatial databases. The *GeoMiner* system includes the spatial data cube construction module, the spatial on-line analytical processing (OLAP) module, and spatial data mining modules. A spatial data mining language, GMQL (Geo-Mining Query Language), is designed and implemented as an extension to Spatial SQL, for spatial data mining. Moreover, an interactive, user-friendly data mining interface has been constructed and tools have been implemented for visualization of discovered spatial knowledge.

make every effort to add to your faith goodness;
and to goodness, knowledge;
and to knowledge, self-control;
and to self-control, perseverance;
and to perseverance, godliness;
and to godliness, brotherly kindness;
and to brotherly kindness, love.
For if you possess these qualities in increasing measure,
they will keep you from being ineffective and unproductive
in your knowledge of our Lord Jesus Christ.

St. Peter on progressive refinement in 2 Peter 1:5-8

I have not stopped giving thanks for you, remembering you in my prayers.
Ephesians 1:16[1]

Acknowledgments

First, I thank Him through Whom all things are possible, I thank our Lord Jesus Christ. My work would not be possible without the gifts God gave me. This work was not by might nor by power, but by His Spirit[2] and to Him be the glory.

I wish to express my gratitude for the continuing support and guidance of my senior supervisor Dr. Jiawei Han. I thank him for his encouragement, confidence and many fruitful meetings we had. His advice and example of perseverance and hard work guided me though my research. He is a person who always finds time in his busy schedule for discussions with me, who has shown me the ropes of scientific work and whose directions I will use in my further career.

I am thankful to my supervisor Dr. Tiko Kameda, for his comments and advice that enabled me to make improvements to this thesis. I would like to thank Dr. Max Egenhofer and Dr. Tom Poiker for agreeing to serve as external examiners and for their remarks that helped me to better present the results of my research and improve the quality of the thesis.

My gratitude and appreciation goes to my friends from the Intelligent Database Systems Laboratory, for many discussions, help and support I have received from them. In particular I would like to thank Nebojša Stefanović, with whom I worked on the *GeoMiner* project; who worked together with me on many issues of spatial data mining and spatial OLAP, who proofread this thesis giving great feedback, and who shared many joyful moments during my years at SFU. I am grateful to other members of the team who made the current implementation of *GeoMiner* possible: Jenny Y. Tam for the design and implementation of DBMiner

[2]Zechariah 4:6.

OLAP module, based on which the GeoMiner's OLAP engine is build; Qing Chen and Wan Gong for the development of the classification module; Yijun Lu for the development of the YACC parser of the GeoMiner; and to Yongjian Fu for the initial development of the multi-level association rule mining module. Many thanks go to Anthony K.H. Tung and Kim Haas for the proofreading of this thesis and their remarks that enhanced the presentation.

I am grateful to Dr. Sławomir Pilarski for his guidance in the early stages of my stay in Canada. I would like to thank the staff and faculty of the School of Computing Science for all the help they provided.

I am very grateful to my family in Poland, for their support and love. I hope they will be proud of my achievements, as proud as I am of them. I thank my spiritual family, my brothers and sisters in Christ for their prayers, continuous encouragement, love and the patience they showed me.

This work was supported in part by the research grant NSERC-OGP003723 from the Natural Sciences and Engineering Research Council of Canada, NCE/IRIS research grant from the Networks of Centres of Excellence of Canada, SFU Graduate Fellowships, SFU President's Ph.D. Stipend, BC Science Council GREAT Award, SFU Faculty of Applied Science Graduate Fellowship, and by Ebco/Epich Scholarships in Expert Systems funded by OPUS Building Corporation, Global (West) Wholesalers Ltd., and A.B.C. Recycling Ltd.

Contents

List of Tables

List of Figures

He brought me out into a spacious place.

Psalm 18:19

Chapter 1

Introduction

Huge amounts of data have been collected through the advances in database technologies and data collection techniques. This explosive growth of data creates the necessity of automated knowledge/information discovery from data, which leads to a promising emerging field, called *data mining* or *knowledge discovery in databases* (KDD) [46, 117]. Knowledge discovery in databases can be defined as the *nontrivial extraction of implicit, previously unknown, and potentially useful information from data* [50]. Data mining represents the integration of several fields, including machine learning, database systems, data visualization, statistics, and information theory.

Although there have been many studies of data mining in relational and transaction databases [3, 46, 61, 117], data mining is in great demand in other applicative databases, including spatial databases, temporal databases, object-oriented databases, and multimedia databases. The focus of this thesis is on the methods of spatial data mining. *Spatial data mining*, or *knowledge discovery in spatial database*, refers to *the extraction of implicit knowledge, spatial relations, or other patterns not explicitly stored in spatial databases* [88].

Previous works in machine learning [48, 104, 107], database systems [137], and statistics [26, 49, 79, 128] laid the foundation for research into knowledge discovery in databases. Advances in the research on spatial data structures and spatial databases enable the efficient querying of large spatial databases [19, 34, 59]. These advances, in combination with the research into spatial reasoning [33, 59], the advances in data mining in relational databases [2, 46] and data warehousing [55] promote the research into spatial data mining [89, 91]. The huge amount of spatial data and the complexity of spatial data types and spatial access methods make the efficiency of spatial data mining algorithms an important research challenge.

1.1 Motivation

A huge amount of spatial data collected by remote sensing systems, regional sales systems, e-commerce, and other data collection tools, makes it crucial to develop tools for discovery of interesting knowledge from large spatial databases. In addition to the information traditionally stored in spatial databases, many relational databases contain spatial information as well. For example, relational data may contain the addresses of the residences of customers or the locations of stores. Such addresses can be geocoded to extract spatial coordinates, which can be stored in a spatial database, together with other spatial data. Through the described geocoding process, the amount of collected spatial data can increase even further. This amount of data has surpassed the capacity of traditional data analysis methods. Standard GISs are very slow in geographic analysis, especially on larger (national and state scale) data intensive business applications [96]. Therefore, it is important to explore new methods for mining knowledge related to both non-spatial and spatial objects in large databases.

Most of the methods for spatial data analysis developed prior to the early 1990s were dealing with numerical data and were using a statistical approach. Statistical analysis, however, may not be a very efficient way to analyze large amounts of data that is spatially interrelated. Geographic data consist of spatial objects and non-spatial descriptions of these objects. Spatial data can be described using two properties, geometric and topological. For example, geometric properties include spatial location, area, perimeter, etc., whereas topological properties include adjacency (object A is a *neighbor of* object B), and inclusion (object A is *inside* object B), and others. Spatial data are often organized by spatial indexing structures and retrieved by spatial access methods. These distinct features of spatial databases create problems when knowledge discovery methods for relational data are applied to spatial data. The data mining methods for spatial data should use spatial relationships and spatial access methods to effectively and efficiently discover knowledge. Therefore, the development of new algorithms should take into account spatial properties of the data and make use of spatial data structures and spatial query processing.

Spatial data mining methods can be applied to extract interesting knowledge from large spatial databases. In particular, these methods can be used for understanding spatial data, discovery of relationships between spatial and non-spatial data, construction of spatial knowledge-bases, query optimization, data reorganization in spatial databases, and capturing the general characteristics in a simple and a concise manner. This has wide applications in GIS, remote sensing, image databases exploration, medical imaging, robot navigation, and other areas where spatial data are used. In the last few years a growing interest in practical applications of spatial data mining could be observed in the areas such as marine ecology [81], space exploration [44], remote sensing [29], traffic analysis [147], and climate research [130]. With the further development of spatial data mining methods, the number of applications should increase even further.

To present the possible applications of our study in greater detail we examine the following application examples:

Example 1.1 Analysis of crime sites
A large number of robberies is committed in a large metropolitan area. A criminologist who analyzes patterns of these robberies may visualize crime sites using a GIS system and use maps presenting locations of other objects, to find patterns. The analysis is performed using a large number of queries and data transformations to produce buffer zones around the crime sites. A statistical package may be used to analyze relationships. □

Example 1.2 Real estate investment analysis
A financial analyst researches the price changes of the houses in a local market. A regression model is constructed based on the characteristics, which were extracted using a number of GIS operations. Such a model is used to predict the changes of the prices of new investments. Also, general statistics are produced to present the influence of different variables on the price. □

Example 1.3 Analysis of the regional sales
A market researcher analyzes sales of a national company base on the lines of products, time, the area of sales and profits. An OLAP system is employed and crosstables are used for visualization. Areas of sales are presented in the form of strings. The spatial map is produced after importing data to the GIS and producing generalized objects. If a GIS system is employed many queries have to be submitted to enable analysis on multiple concept levels. □

As one can see the current GIS can provide environment for the analysis of data using statistical methods or through the interactive process of performing a large number of spatial operations and building models based on the results of these operations. The query language support for more complex operations is weak, and the interface for the data analysis on multiple level is much more complex than the "point and click" approach used in commercial OLAP systems.

Recent progress in other data mining areas such as association rules or classification, as well as the development of On-Line Analytical Processing (OLAP) techniques bring opportunities to extend these methods toward mining of spatial data. This opportunity poses challenges and creates a chance for the advancement of mining knowledge from spatial data.

The goal of this thesis is to analyze efficient methods for mining spatial associations and spatial classification. The developed algorithms are based on the progressive refinement approach. This approach allows for efficient discovery of knowledge in large spatial databases. A complete set of spatial association rules can be discovered and highly accurate decision trees can be constructed using progressive refinement approach. Theoretical analysis and experimental results demonstrate the efficiency of the algorithms. The completeness of the set of discovered spatial association rules is shown through the theoretical analysis and the experiments show that the proposed spatial classification algorithm allows for better accuracy of classification than the algorithm proposed in the previous work [37].

1.2 Scope of the Thesis

Data mining represents the integration of several fields, including machine learning, database systems, data visualization, statistics, and information theory. The *knowledge discovery in databases* is a complex process, which covers many interrelated steps. These steps are connected through the feedback loop based on the results obtained in the discovery process. A user ultimately controls the parameters of the process. Some of the steps can be outlined as the following[1]:

- Learning the application domain, which may include studying prior knowledge, analyzing the goals of the mining process, and building concept hierarchies.

[1]Adapted from [47]

- **Data cleaning and data preprocessing**, which includes removal of noise and outliers if appropriate, unifying different formats of the data and different data sources, handling missing data fields, and sampling.
- **Construction of data warehouses** for On-Line Analytical Processing (OLAP), which includes aggregating the data and materializing views.
- **Choosing the data mining algorithm** that the data will be analyzed with. It includes the decision on the purpose of the model and rules derived by the data mining process. The algorithms for data mining may include clustering, classification, association, predication, etc.
- **Selection of the data items, dimensions, attributes, and measures** used in the mining process.
- **Data reduction and projection**, which includes finding good data representation, reduction of data dimensions, focusing on relevant items of data and relevant attributes.
- **Pattern extraction**, which includes finding rules, patterns, and models through data mining algorithms.
- **Interestingness analysis**, which might filter out the uninteresting patterns.
- **Visualization of the discovered knowledge** through tables, rules, graphs, charts, maps, diagrams, etc.
- **Application of the discovered knowledge** through documenting it, reporting it to the users, taking actions based on the discovered rules, and resolving conflicts with previously believed information.

The main research effort in knowledge discovery has been in the analysis of data mining algorithms for the purpose of pattern extraction from stored data. Data mining algorithms can be classified as follows:

- **Summarization (Characterization)** - provides a compact description of the data. It may include generalized rules and tables, visualization of the data, and the basic statistical functions, such as average or deviation. Many basic OLAP techniques, such as roll-up, drill-down, and slice, allow for interactive exploratory analysis of the summarized data. For example, one can summarize sales based on the location of the stores, time of the sales, and the items sold.

- **Clustering** - maps a data item into one of the clusters (classes) where the items in the same cluster are similar and the items in different clusters are dissimilar. For example, one can divide customers into ten separate groups and apply different marketing strategies to these different clusters.
- **Classification** - maps a data item into one of the predefined categorical classes. For example, the task of discriminating between good and bad customers can be based on classification results.
- **Prediction** - maps a data item onto a numerical variable. The linear and nonlinear **regression** models are very often used for the predication purposes. For example, estimation of the purchases to be done by a customer in the next year can be performed by prediction analysis.
- **Association analysis** - finds the links between data items based on their common occurrence. For example, the analysis of the items bought together in a single transaction may be based on association algorithm.
- **Deviation analysis** - finds deviations from the expected values such as the outliers in a class of objects. For example, the task of finding unexpected credit charge operations can be performed by deviation analysis.
- **Dependency modeling** - describes significant dependencies between variables. For example, Bayesian networks can be used to model the states in the network for the purpose of analyzing the network failures.
- **Time-series analysis** - finds the similarities in sequential data. For example, the analysis of the stock market trends can be done by time-series processing.

In this thesis we propose solutions for some of the spatial knowledge discovery steps and we focus on two spatial data mining algorithms: spatial association and spatial classification. The research has led to the construction of the spatial data mining prototype system *GeoMiner*.

The steps of the knowledge discovery process on which we will concentrate are the following:

- For the purpose of the selection of data mining task, data items, dimensions and measures GMQL – Geo-Mining Query Language is constructed.
- We approach data reduction and projection by proposing a transformation of spatial data using multilevel spatial predicates.
- The proposed spatial association and spatial classification algorithms allow pattern extraction from spatial data.
- The thresholds used in these algorithms allow the filtering out of the uninteresting patterns.
- Some of the knowledge visualization methods are incorporated into the *GeoMiner* spatial data mining prototype.

Data capturing and data cleaning are not analyzed in this thesis in detail, but our implementation of the spatial data mining prototype is based on a commercial GIS MapInfo Professional, which should enable easier input and preprocessing of data. The spatial association and spatial classification algorithms were implemented and a number of experiments were performed with these algorithms. Moreover, the *GeoMiner* prototype also incorporates some of other data mining algorithms such as clustering and summarization.

1.3 Approach

In our study of spatial data mining methods is based on the following principles which are presented together with their rationales:

- Spatial predicates should be used in the mining process.

  Most of the GISs use a quantitative approach for the storage and analysis of spatial data. They work with models built in Cartesian coordinate space. However, it has been recognized that a quantitative approach is not the best representation of the human spatial reasoning process [94]. By using a qualitative representation, such as spatial predicates, we can construct algorithms that are more efficient and a closer match to human analysis. We propose an approach to spatial data mining in which we use (1) non-spatial properties of the spatial objects, and (2) attributes, predicates, and functions representing spatial relations between described objects and other features located in their spatial proximity. The descriptions are generalized using concept hierarchies and presented in the form of predicates. Such predicates present qualitative information and they can be easily understood by humans and used in the representation of the discovered knowledge. For example, one can use the spatial predicate *close_to(school, park)* to present the information that the distance between a school and a park is below a distance threshold. If the quantitative approach was used, the stored information would say that the distance between the particular school and the particular park is exactly 345.2 meters. The same predicate can describe the relationship between many objects and be quite meaningful.

- Knowledge should be presented on different levels of concept hierarchies.

  We believe that the advancement of OLAP tools is partially related to the ability of the OLAP systems to provide multi-level, multidimensional presentation of data stored in large data warehouses [22]. The existing OLAP systems provide mainly tools for summarization and visualization of generalized data. Therefore a new approach to data mining was proposed. This new approach called On-Line Analytical Mining (OLAM), which integrates OLAP and data mining presents a promising direction for mining large databases and data warehouses [71]. The users can interactively adjust the level of generalization by going up or down the concept hierarchies. If a user wants more detailed information about different types of schools he or she can drill-down and use predicates such as *close_to(high_school, park)*. Such progressive "zoom in" and "zoom out" operations can individualize the mining process for different purposes. For example, a higher level user (e.g., CEO) may concentrate on general information, while other users can look for details. The spatial data mining prototype should integrate OLAP module with spatial data mining to provide tools for multi-level analysis of spatial databases.

- Spatial knowledge discovery algorithms should use spatial access methods and spatial query processing algorithms to efficiently mine spatial data.

  Spatial operations such as spatial join [53], that are necessary to compute spatial predicates, are computationally very expensive. Advances in the research on spatial data structures and spatial databases enable the efficient querying of large spatial databases [19, 34, 59]. Results of this research should be used to accelerate extraction of the relevant features. To reduce the cost of these operations, we propose the progressive refinement of the scope of the relevant predicates. Such an approach is similar to the progressive refinement approach for spatial queries. In the first *filter* step(s), approximations of the spatial object such as Minimum Bounding Rectangles (MBRs), are used for computations. The properties of the data mining algorithms, sampling, and machine learning methods are used to select the promising patterns. Then, in the *refinement* step the fine computations are performed only for the promising patterns.

- A user should be able to control mining parameters using query language and GUI.

  As described in Section 1.2 the data mining process is centered toward a user who controls the parameters of the process. For the presentation of the mining query to the system a mining query language should be constructed. The mining language should allow for allow for the specification of (1) the type of data mining algorithm, (2) the set(s) of data for the mining process, and (3) thresholds to filter out uninteresting rules. The language should have a form similar to Spatial SQL [34] and allows for the usage of spatial relationships for the selection of data. The graphical feedback is very important in the analysis of the results [80]. The user interface should enable presentation of the results in multiple forms of outputs, including generalized maps, generalized relations, cross-tabulation, multiple level rules, charts. The mining on different levels of concept hierarchies should be possible using simple mouse operations.

The progressive refinement process of data mining, using spatial predicates, to be presented in detail in this thesis is shown in Figure 1.1.

During the first stage of the process (A in Figure 1.1) the query written in GMQL is analyzed. The proposed language is based on DMQL (Data Mining Query Language) – a query language for mining relational data, that was developed in the Intelligent Database Research lab [66].

During the second stage of the process (B in Figure 1.1) the coarse representatives of the predicates specified in the GMQL query are computed. For example, if one wants to use *close_to* predicates, one can start with "generalized close_to" (*g_close_to*) predicates. The coarse predicate *g_close_to(school, park)* presents the information that the distance between MBRs of a school and a park is below the distance threshold. The coarse predicates can be computed much faster than the fine predicates. The coarse data is generalized using the OLAP module. Then, the relevance of the predicates to the data mining task is analyzed. Only the predicates that are possibly relevant and promising are computed on the fine

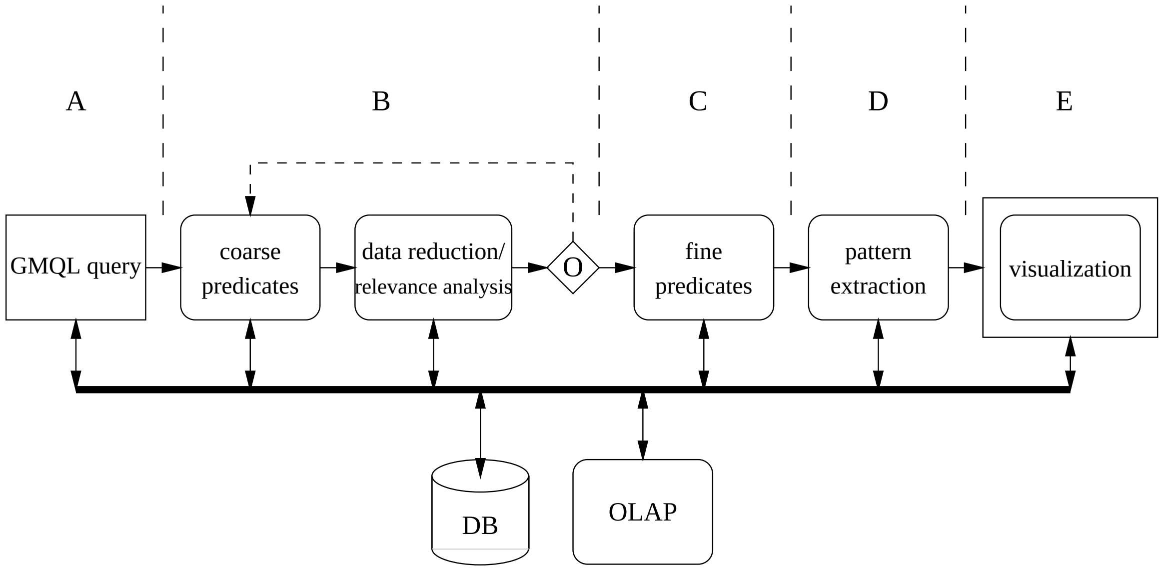

Figure 1.1: The process of spatial data mining.

level during the execution of the next stage of the process. Usually, not all predicates are relevant to the mining task. Thus, by computing irrelevant predicates only on the coarse level and not on the fine level, one may save a lot of the processing time. The progressive refinement process can be even further extended if other methods of relevance analysis are incorporated like the one using the approximations other than MBR (loop (O) in Figure 1.1). This way we can add additional steps that would sequentially narrow the scope of the relevant predicates, functions, and attributes.

In the next stage (C in Figure 1.1) the promising fine predicates are computed and generalized using the attribute-oriented method [61].

During the execution of stage D, spatial patterns are extracted based on promising fine predicates. In this thesis, we analyze two data mining tasks: spatial association and spatial classification.

Finally (E in Figure 1.1), the results of the mining process are presented to the user. The Graphical User Interface (GUI) of our spatial data mining prototype *GeoMiner* allows a user to present mining results in many ways.

The spatial database and OLAP module can be used in every step of the discovery process for data storage/retrieval and data generalization. The relational data cube structure can be used in the generalization process. In our spatial data mining system prototype *GeoMiner*, we integrated a DBMiner's data cube [67] into the OLAP unit for the task of the generalization of spatial predicates.

1.4 Organization of the Thesis

The remainder of this thesis is organized in the following way:

In Chapter 2 we present an overview of progressive refinement methods for spatial query processing and we discuss spatial data mining methods. In Chapter 3, we introduce the concept of spatial association rules, and we present a method for mining such rules. In Chapter 4 we analyze the spatial classification process. A new algorithm for spatial data classification is introduced and it is compared with the algorithm presented in [37]. In Chapter 5 we present the Geo Mining Query Language (GMQL). Finally, we describe the *GeoMiner* spatial data mining prototype in Chapter 6 and a new architecture for the generalization of spatial data is presented. The examples of the spatial data mining results are included. The thesis concludes in Chapter 7, which also presents some future research problems.

Your beginnings will seem humble, so prosperous will your future be.

Job 8:7

Chapter 2

Relevant Work

2.1 Elements of Spatial Data Mining

RULES: Various kinds of rules can be discovered from databases in general. For example, characteristic rules, comparison rules, association rules, or deviation and evolution rules can be mined. A *spatial characteristic rule* is a general description of spatial data. For example, a rule describing the general price range of houses in various geographic regions in a city is a spatial characteristic rule. A *spatial comparison rule* is a general description of the features discriminating or contrasting a class of spatial data from other class(es), like the comparison of price ranges of houses in different geographic regions. Finally, a *spatial association rule* is a rule, which describes the implication of a feature or a set of features by another set of features in spatial databases. For example, a rule associating the price range of the houses with nearby spatial features, like beaches, is a spatial association rule. The rules, together with other methods of knowledge representation are discussed in detail in the rest of the thesis.

THEMATIC MAPS: Thematic maps present the spatial distribution of a single or a few attributes. This differs from base or reference maps where the main objective is to present the position of objects in relation to other spatial objects. Thematic maps may be used for discovering different rules. For example, we may want to look at a *temperature* map while analyzing the general weather pattern of a geographic region. There are two ways to represent maps: *raster* and *vector*. Raster thematic maps have pixels associated with attribute values. For example, a map may have the altitude of the spatial objects coded as the intensity of the pixel (or the color). In a vector representation, a spatial object is represented by its geometry, which is most commonly the boundary representation, along with the thematic attributes. For example, a census tract may be represented by the boundary points and corresponding population value.

IMAGE DATABASES: These are special kinds of spatial databases where data almost entirely consist of images or pictures. Image databases are used in remote sensing, medical imaging, etc. They are usually stored in the form of grid arrays representing the image intensity in one or more spectral ranges.

In this thesis we concentrate on the mining from the databases using vectors for the representations of spatial objects.

Algorithms for spatial data mining involve the use of spatial operations such as spatial joins, map overlays, and nearest neighbor queries. Therefore, efficient Spatial Access Methods (SAM) and data structures for such computations are important issues in spatial data mining [59]. We briefly introduce some of the prominent spatial data structures and spatial computations.

SPATIAL DATA STRUCTURES: Spatial data consist of points, lines, polygons, etc. In order to build indices for these data, multidimensional trees have been proposed. These include quad-trees [125], kd-trees [13] (Figure 2.1), R-trees [60] (Figure 2.2), and R*-trees [11]. A spatial index structure organizes objects into buckets, where each bucket is associated with a bucket region, i.e., the region containing all objects stored in a bucket. Bucket regions are usually rectangles that may overlap. The nodes in the tree are organized into a hierarchical structure where the parent nodes describe regions containing all objects stored in the child nodes. One of the prominent SAMs which has been discussed a lot in the literature is the R-tree [60] and its modification the R*-tree [11]. Objects stored in R-trees are approximated by MBRs. An R-tree stores in every node a set of rectangles. At the leaves, pointers to the representation of object's boundaries and object's MBRs are stored. At the internal nodes each rectangle is associated with a pointer to a child and represents the MBR of all rectangles stored in the child. An excellent survey of spatial data structures is presented in [53].

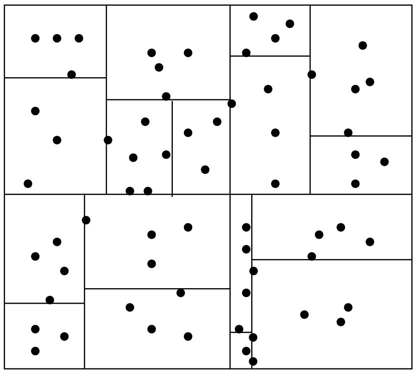

Figure 2.1: Sample kd-tree.

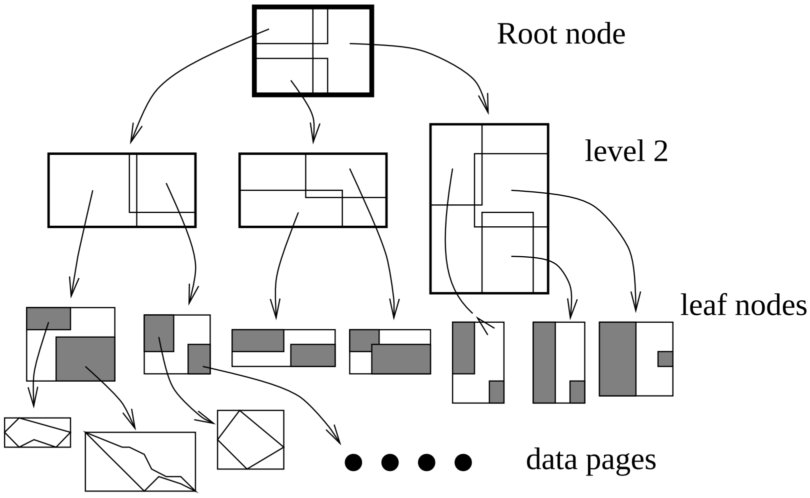

Figure 2.2: Sample R-tree

SPATIAL QUERIES: The spatial operations supported by geographical databases include queries like:

- **Point-in-polygon query**: Find all objects containing a point.
- **Window query**: Find all objects within a query rectangle.
- **Nearest neighbor query**: Find the object closest to a query object.
- **Distance scan**: Enumerate the objects in increasing distance from a query object.
- **Spatial join**: Find all pairs of objects that satisfy join condition, which may be intersection, containment, equality or others.
- **Overlay of two maps**: Find the map resulting from the overlap of two maps.
- **Buffer query**: Determine the buffer around a query object.
- **Path query**: Determine a path between two points.

Spatial access methods and computational geometry algorithms are used to efficiently process these queries.

SPATIAL QUERY PROCESSING: Optimization strategies for spatial query processing are outlined in Aref and Samet [9]. The authors proposed an architecture for spatial database called SAND (spatial and non-spatial data), which is a model of the extended relational database with spatial operations [8]. This architecture provides both spatial and non-spatial components of a spatial database to participate in query processing and optimization. The spatial query processing methods use spatial data structures and computational geometry algorithms for the efficient processing of spatial operations. Some of these methods are described in the next section.

2.2 Progressive Refinement in Spatial Query Processing

A fundamental idea for spatial query processing is the use of approximations. The approximations can be continuous or grid [59]. A continuous approximation is based directly on the spatial coordinates of the approximated object. For example, a *Minimum Bounding Rectangle* (MBR) is a continuous approximation. Grid approximations divide space into cells by a regular grid and the spatial object is represented by the set of cells it intersects.

These approximations are used for effective spatial query processing. A spatial query is executed in a sequence of *filter* and *refinement* steps. The *filter* step uses spatial access methods to restrict the search space. Spatial index methods are usually not able to directly give the exact results of a query, but they are used to present a set of candidates that possibly satisfy the conditions of the spatial query. For example, MBRs can be used to select a number of objects that may be the nearest neighbors of the query objects. Or, for spatial join query pairs of objects, whose MBRs satisfy a join condition, are returned because only these pairs of objects can potentially be in the set of the results of spatial join.

The *refinement* step is used to process candidate objects (or object pairs in the case of spatial join). In this step the exact geometric representations of the objects are used for the computations. As such exact representations may be large, the final processing might be quite expensive to accomplish. However, this step deals with a smaller number of candidates and, therefore the total processing cost might be minimized.

Recently, Brinkhoff *et al.* further extended the processing of spatial join by introducing an additional *geometric filter* that uses approximations other than MBRs [19]. The multi-step processing of spatial joins is represented in Figure 2.3[1]. The first step finds the set of candidate pairs, e.g., pairs of objects whose MBRs intersect. Spatial access methods using spatial data structures like R-trees, or geometric algorithms using plane sweep methods, are used to find all candidate pairs efficiently. One can notice, that the first step using MBRs cannot find the exact hits and may produce a large number of false hits.

The *geometric filter* is added to improve the efficiency of the processing. This filter produces hits (e.g., pairs of objects satisfying the join condition), false hits (e.g., pairs of objects that cannot satisfy the join condition) and a reduced set of candidates (e.g., pairs of objects possibly satisfying join condition). To produce hits, progressive approximations are used. A polygonal object is progressively approximated if the point set of approximation is a subset of the point set of the object [19]. *Maximum Enclosed Rectangles* (MERs) or *Maximum Enclosed Circle* (MECs) are progressive approximations. Thus, if the MECs for two objects intersect, the exact geometrical representations of the objects also intersect and a hit is detected. For the detection of false hits, approximations that are more detailed than MBRs are processed. Structures like *Minimum Bounding Circle, Minimum Bounding Ellipse, Rotated Minimum Bounding Rectangle, Minimum Bounding 4-Corner, Minimum*

[1] Adapted from [19].

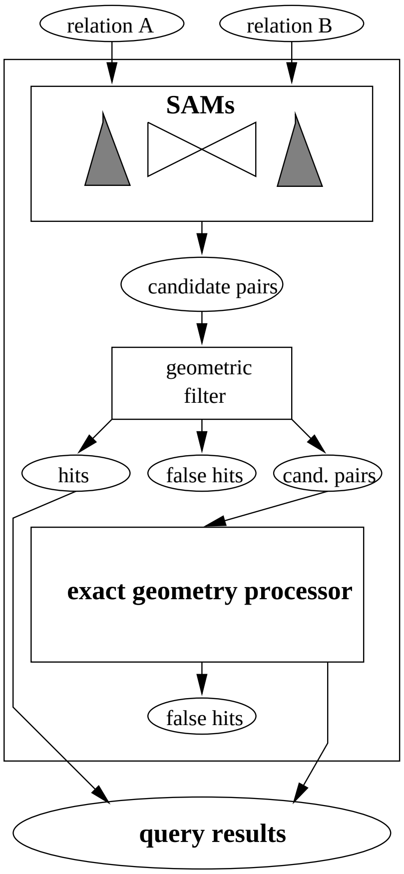

Figure 2.3: Multistep join processing

Bounding 5-Corner, and *Convex Hull* were analyzed [19] and it was found that the last two approximations are especially useful in detecting false hits. Minimum Bounding 5-Corner needs small storage and is able to eliminate about 66% of the non-intersecting object pairs delivered from the first filter step.

Finally, the exact geometry processor is used to process the reduced set of candidate pairs.

2.3 Spatial Data Mining Methods

With the advent of data mining, researchers proposed various methods for discovering knowledge from large databases. Most of them concentrate on relational or transaction databases.

These methods strive to combine already mature areas like machine learning, databases and statistics. Studies [2, 61, 117] laid a foundation for spatial data mining. The machine learning techniques *learning from examples* and *generalization and specialization* are widely used in spatial data mining. It did not take long before the statistical cluster analysis technique was modified for the use in spatial data mining [109]. Also other methods were extended toward knowledge discovery in spatial databases.

The algorithms for spatial data mining include generalization-based methods for mining spatial characteristic and discriminant rules [61, 99, 109], clustering algorithms utilizing spatial data structures [35, 36, 126, 127, 129, 142], the aggregate proximity technique for finding characteristics of spatial clusters [85], the spatial trend detection technique based on neighborhood indices [37], etc. In the following subsections, we present an overview of data mining architecture and describe a number of spatial data mining algorithms.

2.3.1 Data Mining Architecture

Various architectures (models) have been proposed for data mining. They include Han's architecture for general data mining prototype DBLEARN/DBMINER [64], Holsheimer *et al.*'s parallel architecture [73], and Matheus *et al.*'s multicomponent architecture [103]. Matheus *et al.*'s architecture seems to be very general and has been extended for spatial data mining [35]. This architecture, which is similar to other data mining architectures, is presented in Figure 2.4. In this architecture, a user may control every step of the mining process. Background knowledge, like spatial and non-spatial concept hierarchies, or information about database, is stored in a knowledge base. Data is fetched from the storage using the *DB interface* which enables optimization of the queries. Spatial data index structures, like R-trees, may be used for efficient processing. The *Focusing Component* decides which parts of data are useful for pattern recognition. For example, it may decide that only some attributes are relevant to the knowledge discovery task, or it may extract objects whose usage promises good results. Rules and patterns are discovered by the *Pattern Extraction* module. This module may use statistical, machine learning, and data mining techniques in conjunction with computational geometry algorithms to perform the task of finding rules and relations. The interestingness and significance of these patterns is then processed by the *Evaluation* module to possibly eliminate obvious and redundant knowledge. These four components may interact between themselves through the *Controller*.

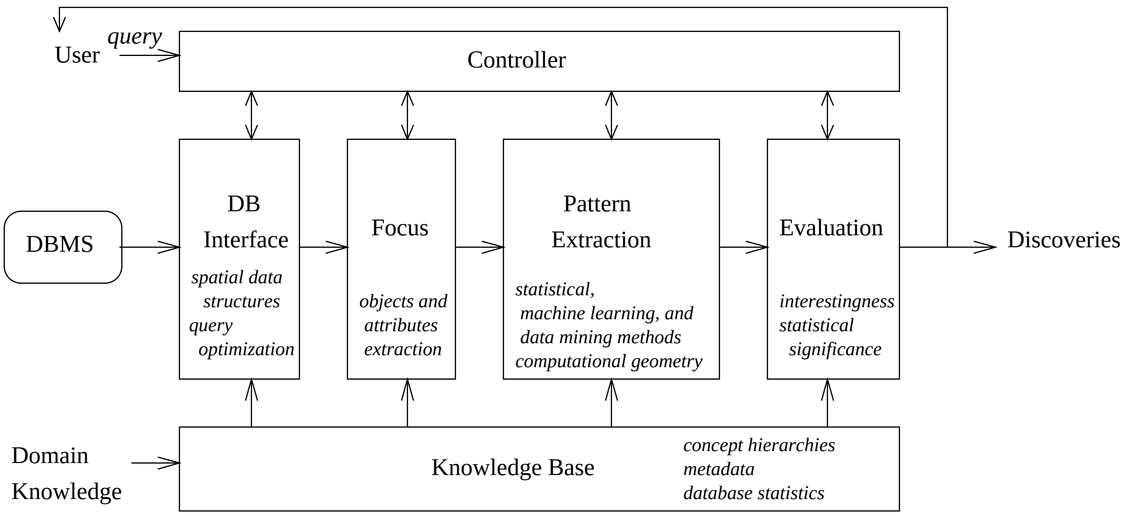

Figure 2.4: An architecture for a KDD system

2.3.2 Statistical Spatial Analysis

Statistical spatial analysis [49, 128] has been the most common approach for analyzing spatial data. Statistical analysis is a well-studied area and therefore a large number of algorithms exists, including various optimization techniques. It handles numerical data very well and usually comes up with realistic models of spatial phenomena. The major disadvantage of this approach is the assumption of statistical independence among the spatially distributed data. This causes problems, as many spatial data are in fact interrelated, i.e., spatial objects are influenced by their neighboring objects. Kriging or regression models with spatially lagged forms of the dependent variables can be used to alleviate this problem to some extent. Unfortunately, it makes the whole modeling process more complicated and can only be done by experts with a fair amount of domain knowledge and statistical expertise. In other words, it is not the kind of technique that we want to present to the end users for the analysis of spatial data. Furthermore, the statistical approach cannot model nonlinear rules very well, and symbolic values, like names, are handled poorly. Statistical methods do not work well with incomplete or inconclusive data either. Another problem related to statistical spatial analysis is the expensive computation of the results. Openshaw [115] presents a discussion on common spatial analysis methods, and bibliography on this topic.

2.3.3 Generalization-Based Knowledge Discovery

One of the widely used techniques in machine learning is *learning from examples* [104]. This method is often combined with the generalization method [107]. This approach cannot be directly adopted for large spatial databases because: 1) the algorithms have a running time that is exponential in the number of examples; and 2) it does not handle noise and inconsistent data very well. Han *et al.* modified these techniques and gave an attribute-oriented (as opposed to the tuple-oriented in machine learning algorithms) induction algorithm to mine knowledge from large relational databases [61, 65]. Later, Lu *et al.* extended this technique to spatial databases [99]. Thus, the assumptions that are made for relational databases are also carried to spatial data mining.

The generalization-based knowledge discovery methods require the existence of background knowledge in the form of concept hierarchies. In the case of spatial databases, there can be two kinds of concept hierarchies: non-spatial and spatial. Concept hierarchies can be explicitly given by the experts, or in some cases they can be generated automatically by data analysis [62]. An example of a concept hierarchy for *agricultural land use* is shown in Figure 2.5. As we ascend the concept tree, information becomes more and more general, but still remains consistent with the lower concept levels. For example, in Figure 2.5 both *jasmine* and *basmati* can be generalized to the concept *rice* which in turn can be generalized to concept *grains*, which also includes *wheat*. One can notice that such generalization process differs from the cartographic map generalization process, which involves the generalization of the symbolic representations of the objects [20].

A similar hierarchy may exist for spatial data. For example, in a generalization process, regions representing counties can be merged to states, and states can be merged to larger areas. Attribute-oriented induction is performed by climbing the generalization hierarchies and summarizing the general relationships between spatial and non-spatial data at higher concept levels. It can be done on non-spatial data by (a) climbing the concept hierarchy when attribute values in a tuple are changed to the generalized values, (b) removing attributes when further generalization is impossible and there are too many different values for an attribute, and (c) merging identical tuples. Induction is continued until values for every attribute are generalized to the desired level. The desired level is reached when the number of different values for the attribute in the generalized table is no greater than the *generalization threshold* for this attribute. During the process of merging identical tuples the

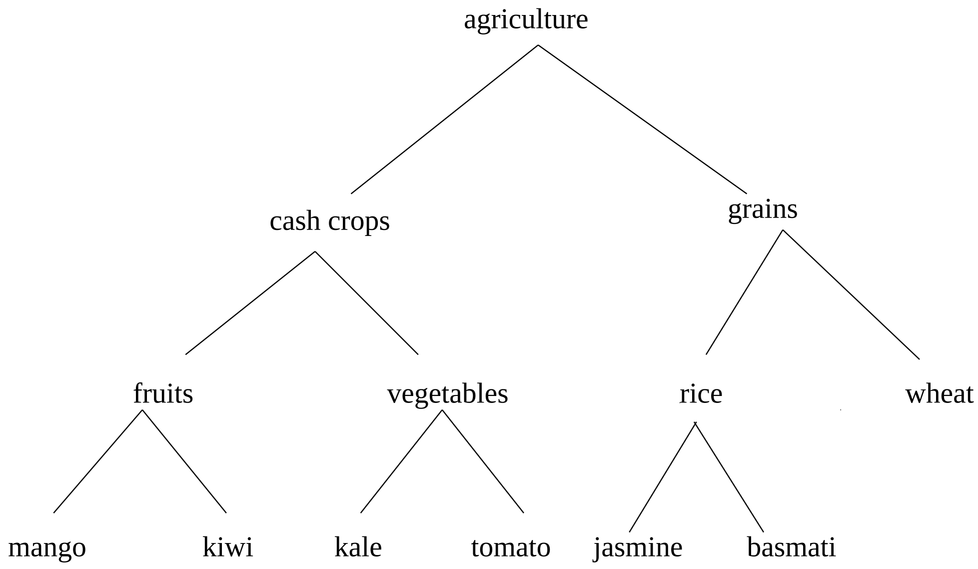

Figure 2.5: Example of agricultural land use concept hierarchy

number of merged tuples is stored in an additional attribute *count* to enable the quantitative presentation of acquired knowledge. Other aggregate values for merged tuples may be stored as well. Lu *et al.* presented two generalization based algorithms, *spatial-data-dominant* and *non-spatial-data-dominant* generalizations [99]. Both algorithms assume that the rules to be mined are general data characteristics and that the discovery process is initiated by the user who provides a learning request (query) explicitly, in a syntax similar to SQL. We will briefly describe both algorithms as follows:

SPATIAL-DATA-DOMINANT GENERALIZATION: In the first step all data described in the query are collected. Given the spatial data hierarchy, generalization can be performed first on the spatial data by merging the spatial regions according to the description stored in the concept hierarchy. Generalization of the spatial objects continues until the spatial *generalization threshold* is reached. The spatial generalization threshold is reached when the number of regions is no greater than the threshold value. After the spatial generalization process, non-spatial data are retrieved and analyzed for each of the spatial objects using the attribute-oriented induction technique as described above. An example of a query from [99] and the result of the execution of the spatial-data-dominant generalization algorithm are

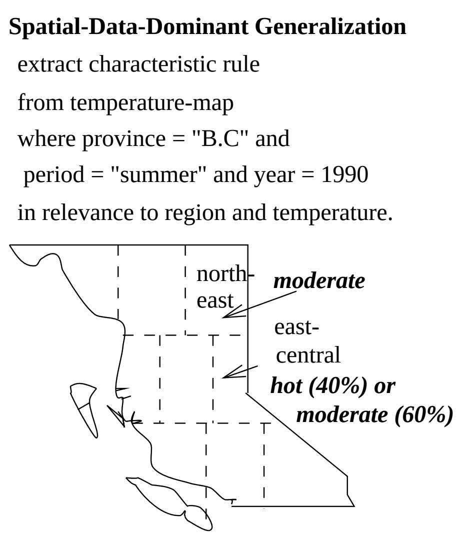

Figure 2.6: Example of the queries and the results of the execution of the spatial and non-spatial data-dominant generalization methods

presented in Figure 2.6. In this example, temperature in the range [20, 27) is generalized to *moderate*, and temperature in the range [27, ∞) is generalized to *hot*. The answer to the query is the description of all regions using a disjunction of a few predicates, which characterize each of the generalized regions. Temperature measured in the east-central region of British Columbia is in the range [22, 30]. Thus, in our example the description of the temperature weather pattern in this region is hot or moderate. The computational complexity of the algorithm is $\mathcal{O}(NlogN)$, where N is the number of spatial objects.

NON-SPATIAL-DATA-DOMINANT GENERALIZATION:
This method also starts with collecting all data relevant to the user query. In the example presented in Figure 2.6 the DB interface extracts the precipitation data relevant to the province and time period specified in the query. In the second step the algorithm performs attribute-oriented induction on the non-spatial attributes, generalizing them to a higher (more general) concept level. For example, the precipitation value in the range (10 in., 15 in.] can be generalized to the concept *wet*. The *generalization threshold* is used to determine whether to continue or stop the generalization process. In this step the pointers to spatial objects are collected as a set and put with the generalized non-spatial data. In

the third and the last step of the algorithm, neighboring areas with the same generalized attributes are merged together based on the spatial function *adjacent_to*. For example, if in one area the precipitation value was 17 in., and in neighboring area it was 18 in. both precipitation values are generalized to the concept *very wet* and both areas are merged. Approximation can be used to ignore small regions with different non-spatial description. For example, if the majority of land area can be described as *industrial*, but a few gas stations exist in this area the whole area can be described as an *industrial* one. The result of the query may be presented in the form of a map with a small number of regions with high level descriptions as it is shown in Figure 2.6. The computational complexity of this algorithm is also $\mathcal{O}(N log N)$, where N is the number of spatial objects.

We presented two generalization-based algorithms that assumed the concept hierarchies to be given or generated automatically. However, as pointed out before, there may be cases where such hierarchies are not present *a priori*. Another problem with previous algorithms is that the spatial components of the databases are explored by merging regions at lower levels of the concept hierarchy to form region(s) at higher levels of the hierarchy. Both of these facts suggest that the quality and the interestingness of the mined characteristic rules is going to be much dependent upon the given concept hierarchy(ies). In many cases such hierarchies are given by the experts and they may be not entirely appropriate. Therefore, we would like to find algorithms that do not need to use these hierarchies. We will describe some of the algorithms not depending on spatial concept hierarchies in the next subsection.

2.3.4 Methods Using Clustering

Cluster analysis is a branch of statistics that has been studied extensively for many years. The main advantage of using this technique is that interesting structures or clusters can be found directly from the data without using any background knowledge, like concept hierarchies. A similar approach in machine learning is known as *unsupervised learning*. We can exploit the results of research on clustering techniques in the spatial data mining process as proposed in [109].

Clustering algorithms used in statistics, like hierarchical, **PAM** (Partitioning Around Medoids) or **CLARA** (Clustering LARge Applications) [79], are reported to be inefficient from the computational complexity point of view. As for the efficiency concern, a new algorithm, called **CLARANS** (Clustering Large Applications based upon RANdomized Search), was developed for cluster analysis. Experimental evidence showed that CLARANS outperforms PAM and CLARA. Ng and Han used CLARANS in spatial data mining algorithms, SD(CLARANS) and NSD(CLARANS). First, we will briefly describe the three cluster analysis algorithms.

The PAM algorithm was developed by Kaufman and Rousseeuw [79]. Assuming that there are N objects, PAM finds k clusters by first finding a representative object for each cluster. Such a representative, which is the most centrally located point in a cluster, is called a *medoid*. After selecting k *medoids*, the algorithm repeatedly tries to make a better choice of medoids analyzing all possible pairs of objects such that one object is a medoid and the other is not. The measure of clustering quality is calculated for each such combination. The best choice of points in one iteration is chosen as the medoids for the next iteration. The cost of a single iteration is $\mathcal{O}(k(N-k)^2)$. It is therefore computationally inefficient for large values of N and k.

The CLARA algorithm was proposed by Kaufman and Rousseeuw [79] as well. The difference between the PAM and CLARA algorithms is that the latter one is based upon *sampling*. Only a small portion of the real data is chosen as a representative of the data and *medoids* are chosen from this sample using PAM. The idea is that if the sample is selected in a fairly random manner, then it correctly represents the whole data set and therefore, the representative objects (medoids) chosen, will be similar as if chosen from the whole data set. CLARA draws multiple samples and outputs the best clustering out of these samples. As expected, CLARA can deal with larger data sets than PAM. The complexity of each iteration now becomes $\mathcal{O}(kS^2 + k(N-k))$, where S is the size of the sample. The authors indicated through their experimental results that samples of size *40+2k* give good results.

It is easy to realize that PAM searches for the best k medoids among a given data set whereas CLARA searches for the best k medoids among the selected sample of the data set. Suppose that object O_i is one of the medoids in the best k medoids. Thus, if during sampling O_i is not selected, then CLARA will never find the best clustering. This is exactly the tradeoff for efficiency. Ng and Han [109] proposed CLARANS algorithm which tries to mix both PAM and CLARA by searching only the subset of the data set and it does not confine itself to any sample at any given time. While CLARA has a fixed sample at every stage of the search, CLARANS draws a sample with some randomness in each step of the search. The clustering process can be presented as searching a graph where every node is a potential solution, i.e., a set of k medoids. The clustering obtained after replacing a single medoid is called the *neighbor* of the current clustering. The number of neighbors to be randomly tried is restricted by the parameter *maxneighbor*. If a better neighbor is found CLARANS moves to the neighbor's node and the process starts again; otherwise, the current clustering produces a local optimum. If the local optimum is found CLARANS starts with a new, randomly selected node in search for a new local optimum. The number of local optima to be searched is also bounded by the parameter *numlocal*. It should be mentioned that CLARANS can be used to find the most natural number of clusters k_{nat}. The authors adopted a heuristic of determining k_{nat}, which uses *silhouette coefficients*[2], introduced by Kaufman and Rousseeuw [79]. CLARANS also enables the detection of outliers, e.g., points that do not belong to any cluster. CLARANS has been experimentally shown to be more efficient than both PAM and CLARA. The authors claim that the computational complexity of every iteration in CLARANS is basically linearly proportional to the number of objects. This claim has been supported by Ester *et al.* in [35]. Unfortunately, if the natural number of clusters is unknown, then all possible numbers of clusters have to be analyzed and the complexity grows to $\mathcal{O}(N^2)$.

Based upon CLARANS, two spatial data mining algorithms were developed in a fashion similar to the algorithms discussed earlier in this section: *spatial dominant approach*, SD(CLARANS) and *non-spatial dominant approach*, NSD(CLARANS). Both algorithms assume that the user specifies the type of the rule to be mined and relevant data through a learning request in a similar way as in the experimental database mining prototype, DBLearn [61].

Algorithm SD(CLARANS)

In this spatial dominant approach, spatial components of the relevant data items are collected and clustered using CLARANS. Then, the algorithm performs an attribute-oriented induction on non-spatial description of objects in each cluster. The result of the query presents high-level non-spatial description of objects in every cluster. For example, one can find that in Vancouver expensive housing units are clustered in 3 clusters. In the downtown cluster there are mainly expensive condos; mansions and single houses are located in the waterfront cluster; and the third cluster consists mainly of single houses.

[2]It is a property of an object that specifies how much the object truly belongs to the cluster.

Algorithm NSD(CLARANS)

This non-spatial dominant approach first applies non-spatial generalizations. Attribute-oriented generalization is performed on the non-spatial attributes and produces a number of generalized tuples. For example, the descriptions of expensive housing units can be generalized to single houses, mansions and condominiums. Then, for each such generalized tuple, all spatial components are collected and clustered using CLARANS to find k_{nat} clusters. In the final step, the clusters obtained in this manner are checked to see if they overlap with clusters describing other types of objects. If so, then the clusters are merged, and the corresponding generalized non-spatial descriptions of tuples are merged as well.

Depending upon the rules or the form of knowledge that the user wants to discover, it may be better to choose one or the other of the above two algorithms. Usually SD(CLARANS) is more efficient than NSD(CLARANS). But, when the distribution of points is mainly determined by their non-spatial attributes NSD(CLARANS) may have an edge.

The efficiency of the CLARANS algorithm was improved by Estivill-Castro and Murray in [40]. The authors also proposed a method to find correlations between the locations of the medoids of the clusters and the locations of other types of objects in the database like schools, subway stations, etc.

CLARANS in Large Spatial Databases: Focusing Methods

Ester *et al.* in [35] pointed out some of the drawbacks of the CLARANS clustering algorithm [109]. First of all, CLARANS assumes that the objects to be clustered are all stored in the main memory. This assumption may not be valid for large databases and that is why disk-based methods could be required. Secondly, the efficiency of the algorithm can be substantially improved by modifying the *focusing* component of the algorithm (see architecture in Figure 2.4).

The first drawback is alleviated by integrating CLARANS with efficient spatial access methods, like R*-tree. R*-tree supports the focusing techniques that Ester *et al.* proposed to reduce the cost of computations. The authors showed that the most computationally expensive step of CLARANS is calculating the total distances between the two clusterings. Thus, they proposed two approaches to reduce the cost of this step.

The first approach is to reduce the number of objects to consider. This can be done using a sample (see [113] for some sampling algorithms). However, bad sampling may lead to poor quality of clustering. A *centroid query* returns the most central object of a leaf node of the R*-tree where neighboring points are stored. Only these objects are used to compute the medoids of the clusters. Thus, the number of objects taken for consideration is reduced. This technique is called *focusing on representative objects.* The drawback is that some objects, which may be better medoids, are not considered, but the sample is drawn in a way, which still ensures a good quality of clustering.

The other technique to reduce the computations is to restrict the access to certain objects that do not actually contribute to the computation. The authors further gave two different focusing techniques, which try to exploit this approach: *focus on relevant clusters*, and *focus on a cluster.* Using the R*-tree structure the authors proposed a way of performing computation only on pairs of objects that can improve the quality of clustering instead of checking all pairs of objects as it is done in CLARANS algorithm.

Ester *et al.* applied the focusing on representative objects to a large protein database to find the segmentation of protein surfaces so as to facilitate the so-called *docking queries.* They reported that when the focusing technique was used the effectiveness decreased just from 1.5% to 3.2% whereas the efficiency increased by a factor of 50, which was the number of points stored in a disk page. The measure of effectiveness used is the average distance of the resulting clustering, whereas the measure of efficiency used is the CPU time.

Clustering Features and CF trees

R-trees are not always available and their construction may be time consuming. Zhang *et al.* presented another algorithm – **BIRCH** (Balanced Iterative Reducing and Clustering) – for clustering large sets of points [148]. The method they presented is the incremental one with possibility of adjustment of memory requirements to the size of memory that is available. The authors used concepts called *Clustering Feature* and *CF tree.*

A *Clustering Feature* CF is the triple summarizing information about subclusters of points. Given $\{\vec{X_i}\}$, i.e., the set of N d-dimensional points in the subcluster, CF for this subcluster is defined as

$$\mathrm{CF} = (N, \vec{LS}, SS)$$

where N is the number of points in the subcluster, $\vec{LS}$ is the linear sum on N points, i.e., $\sum_{i=1}^{N} \vec{X_i}$, and SS is the square sum of data points, i.e., $\sum_{i=1}^{N} \vec{X_i}^2$. The *Clustering Features* are sufficient for computing clusters and they constitute an efficient information storage method as they summarize information about the subclusters of points instead of storing all points.

A *CF tree* is a balanced tree with two parameters: branching factor B and threshold T. The branching factor specifies maximum number of children. The threshold parameter specifies the maximum diameter of subclusters stored at the leaf nodes. By changing the threshold value we can change the size of the tree. The non-leaf nodes store sums of their children's *CF*s, and thus, they summarize the information about their children. The *CF tree* is built dynamically as data points are inserted. Thus, the method is an incremental one. A point is inserted to the closest leaf entry (subcluster). If the diameter of the subcluster stored in the leaf node after insertion is larger than the threshold value, then the leaf node, and possibly other nodes are split. After the insertion of the new point the information about it is passed towards the root of the tree. One can change the size of the *CF tree* by changing the threshold. If the size of the memory that is needed for storing the *CF tree* is larger than the size of the main memory, then a larger value of threshold is specified and the *CF tree* is rebuilt. The rebuild process is performed by building a new tree from the leaf nodes of the old tree. Thus, the tree is rebuilt without the necessity of reading all the points. Therefore, for building the tree, data has to be read just once. The authors also present some heuristics for dealing with outliers and methods for improving the quality of CF trees, by additional scans of the data.

Any clustering algorithm, including CLARANS may be used with CF trees. The CPU and I/O costs of the BIRCH algorithm are $\mathcal{O}(N)$. The authors performed a number of experiments, which showed linear scalability of the algorithm with respect to the number of points, insensibility to the input order, and good quality of the clustering.

Multiple representatives of a cluster

Traditional clustering algorithms like k-mean, or k-medoids approaches suffer from the fact that a cluster is represented by just one point. This leads to problems if a shape of a cluster is a non-spherical one, or if there is a large difference in the sizes of the clusters. To deal with this problem Guha *et al.* proposed to use multiple representatives for a cluster [58]. The clustering algorithm - **CURE** (Clustering Using Representatives) - chooses a set of well-scattered points to represent a cluster. The well-scattered points are as far from each other as possible. To avoid a large influence of points on the boundary of the clusters the representative points are moved toward the center of the cluster. The algorithm starts with a random sample of the points and it partitions the sample. The kd-tree (Figure 2.1) is used for data storage. For each partition the points are hierarchically merged, forming clusters represented by p representatives. Finally, the clusters from the partitions are merged and all data is labeled on the disk. The experiments show that the algorithm handles data with a large difference in size of the clusters better than BIRCH algorithm, and it is able to deal with small "bridges" between the clusters better than the Minimum Spanning Tree method. It also handles outliers well and has a good efficiency rate with large databases.

Density-Based Clustering

Figure 2.7 shows the results of a medoid-based clustering method (CLARANS) on (1) clusters with significantly different diameters that are located close to one another (database 1); (2) concave clusters (database 2); and (3) clusters with noise (database 3). One may observe that the result of the clustering is different from the intuitively expected one. Ester *et al.* described density-based clustering method **DBSCAN**, which can deal with these problems, and additionally does not require the number of clusters as the input parameter [36]. According to the density-based definition of a cluster, for every point in a cluster there exist at least *MinPts* points in the same cluster within a distance less than the distance threshold *Eps* (points are densely located). The algorithm makes use of a spatial data structure (R*-tree) to locate points within *Eps* distance from the core points of the clusters. The average time complexity of the DBSCAN algorithm is $\mathcal{O}(NlogN)$. During the experiments DBSCAN outperformed CLARANS by a factor of between 250 and 1900 which grows with the size of the database. The DBSCAN discovered clusters which match intuitive cluster definition, as it is shown in Figure 2.8. The authors also proposed techniques for determining the parameters *MinPts* and *Eps*.

Xu *et al.* proposed an algorithm **DBCLASD** (Distribution Based Clustering of LArge Spatial Databases) [145] that eliminates the need for *MinPts* and *Eps* parameters. The algorithm is based on the assumption that points inside a cluster are randomly distributed. Such distribution implies the distribution of the distance to nearest neighbor points. The clusters grow in the way similar to DBSCAN expansion. The growth of the clusters continues as long as the distribution of points matches random distribution.

Schikuta also uses the concept of density based clustering to find the patterns in large databases. Two data structures were investigated: **Grid** structure [126] and **BANG** structure [127], with the second structure yielding better results. The algorithm starts with the

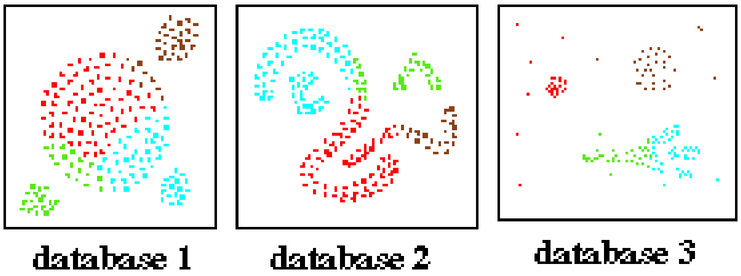

Figure 2.7: Clusters discovered by CLARANS

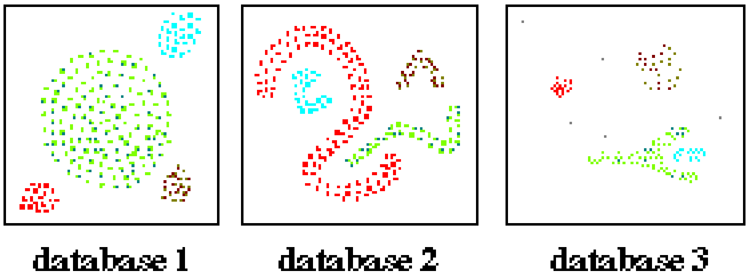

Figure 2.8: Clusters discovered by DBSCAN

construction of an index structure (Grid or BANG), which is similar to the kd-tree structure (Figure 2.1). Each data structure partitions data space into non-empty, disjoint, rectilinear blocks, adapting to the distribution of the points. In the densely populated regions, blocks are smaller than the blocks in sparsely populated regions. The blocks are sorted according to their density, i.e., the ratio of the number of points in the block to the area of the block. The blocks with high density are selected as centers of clusters and the clusters are expanded by adding neighboring blocks with the density that is not greater than the density of the currently analyzed block.

Wang *et al.* proposed another data structure that can be used for density based clustering [142]. **STING** structure is similar to a quadtree. Each node summarizes information about its child nodes including the number of points within the node and information about numerical attributes describing points. The descriptions of numerical attributes include mean, standard deviation, maximum, minimum, and the type of the distribution. The data structure assists in answering different types of questions. For example, one can find maximum connected regions, which satisfy density condition and have particular properties for non-spatial attribute. The properties of the non-spatial attributes can be specified with certain confidence. For example, one may find areas where, with 90% confidence, at least 80% of houses are in the price range between $300,000 and $500,000. The query processing starts with the top level and all children that may possibly satisfy the query condition are analyzed. At the bottom level the neighbors of the identified regions are examined and merged with the discovered clusters, if they satisfy query conditions.

Sheikholeslami *et al.* proposed **WaveCluster** algorithm, which also uses the notion of density [129]. The points are put into a grid and the density of points is calculated. The wavelet transformation is performed on the density function and high frequency signal is filtered out to remove outliers. This process may be repeated to find clusters at coarser levels. The algorithm finds the connected components (clusters) and assigns points to them.

2.3.5 Characterization of Clusters

We discussed clustering algorithms in the previous section. These algorithms are effective and efficient methods to find *where* the clusters in the spatial database are, i.e., to partition data into clusters. However, these algorithms do not answer the question *why* the clusters are there. Ng and Knorr in [85, 86] presented studies motivated by this question. This question can be rephrased as "what are the characteristics of the clusters in terms of the features that are close to them". Then we can find features (objects) that are common to the number of clusters or features that are different for the clusters [86]. We can try to answer the same question characterizing clusters not only using objects like schools, hospitals, etc., but also using thematic maps like census coverage [85].

Aggregate Proximity and Algorithm CRH

The aggregate proximity is the measure of closeness of the set of points in the cluster to a feature as opposed to the distance between a cluster boundary and the boundary of a feature. One of the problems is how to measure the aggregate proximity, because statements like *90% of the houses in a cluster are close to the feature F* are more informative and interesting than statements like *one house on the boundary of the cluster is close to a certain feature F.*

One may ask why the authors are not simply using the k nearest neighbor searches with data structures like k-d trees, R-trees and its variants, Voronoi diagrams[3], etc. It turns out that such structures are unable to perform the search needed to find aggregate proximity in an efficient way. For example, the distance between the cluster and a feature is measured as the distance between the boundaries, not between the points, like centroids. Furthermore, the costs of building and maintaining the indices are prohibitive, given the fact that such indices may not be used frequently. Therefore, the authors propose the use of computational geometry concepts [119] to find out the characteristics of a given cluster in terms of the features close to it. The authors described the algorithm **CRH** (where C is for encompassing circle, R for isothetic rectangle, and H for convex hull[4]), which uses such concepts as filters to reduce the candidate features at multiple levels. In short, they collect a large number of features from multiple maps and feed them along with the cluster to the algorithm CRH and discover knowledge about spatial relationships, as shown in Figure 2.9.

Knorr and Ng evaluated various computational geometry algorithms for distance computation, and shape descriptions and overlap computations. Taking into account the problem of data distribution in a cluster and various sizes and shapes of the features, the authors chose a technique for computing the distance between a cluster point and feature boundary. For the shape description, the authors chosen minimum bounding structures. They used

[3]Voronoi diagram of a set of points S is a partition of a space by Voronoi edges which consist of points that are "equidistant" to the closest points from a set S.

[4]Convex hull is the minimal, simple closed curve of a set of points such that a line connecting any two points of the set always lies on the interior of the boundary of convex hull.

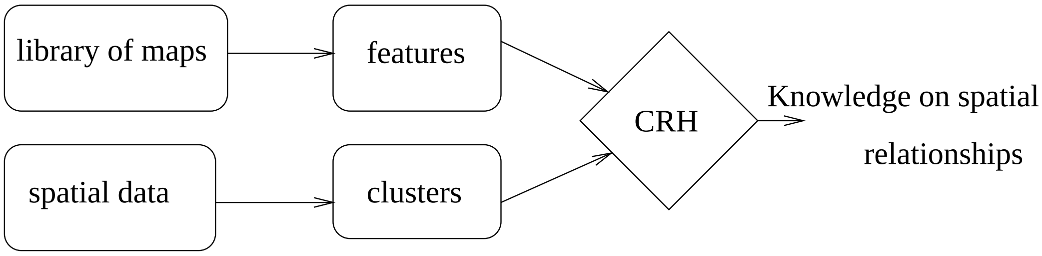

Figure 2.9: Using CRH for knowledge discovery in spatial databases

these structures to develop a multiple filtering approach, with the filters set up in an increasing order of accuracy but decreasing order of efficiency. That is, filters that are applied earlier are more efficient but coarser than the later ones.

The algorithm CRH first applies the encompassing circle filter to the large number of features. The most promising features are passed to the isothetic rectangle filter. These two filters eliminate a large number of features and only a small number of features is passed to the final convex hull filter. Then, the CRH algorithm calculates the aggregate proximity of points in the cluster to the convex boundary of each feature, upon which the features are ranked. In addition, each filter has its own threshold, which is the minimum number of features to pass on to the next filter. When the number of found features lies below the threshold, the cluster is enlarged to encompass more features to pass the threshold limit. Shape enlargement can be achieved by the *linear* policy (enlarge the shape by constant distance), or by the *bisection* policy. Bisection policy performs enlargement or diminution of the area by a distance which decreases logarithmically. This policy checks if enough features are in the area of shape and enlarges or decreases the area according to the need.

The problem with this method is that a feature may have to be tested for overlap with a cluster many times. The technique, which the authors call *memoization*, can be used to avoid multiple computations by storing the distance between each feature and the cluster the first time the intersection test is performed. Depending upon the shapes, circles, rectangles or convex hulls, the minimum distance between the circumferences, the boundaries of the rectangles, or the boundaries of the polygons respectively, are stored. Finally, the algorithm reports the features with the smallest aggregate proximities showing minimum and maximum distances of points in the cluster to the feature, average distance, and percentages of points located in the distance less than specified thresholds.

The experiments show that the algorithm CRH has the response time of less than two seconds for processing 50,000 features. Furthermore, it is empirically shown to be scalable and the *memoization* policy is found to be the most consistent and efficient of all the shape enlargement policies.

Finding Commonalities and Maximal Discriminators between Clusters

After finding features (objects), which are close to certain clusters we can find features which are common to a number of clusters or features that discriminate among clusters. Because comparison on the level of single objects may not reveal common features of the objects, generalization techniques were adopted to solve this problem [86]. The list of k top features is taken for each cluster for the analysis. Because clusters may be close to different objects, for example, different schools, the descriptions of objects have to be generalized. For example, "St. George's College" can be generalized to "private boys' school". Finally, the description of the commonality may be presented as a simple rule like "all clusters are close to private schools and parks". Furthermore, one can rank features according to the measure of "degree of commonness".

In a similar way, Knorr and Ng proposed an algorithm for finding features that discriminate clusters. For example, the algorithm may find that clusters of expensive houses are close to golf courses, while clusters of low price houses are not close to golf courses.

This technique can be extended for finding similarities and differences between clusters using thematic maps [111]. The authors developed measures for analysis of maps with nominal values, like political parties and ordinal values, like income.

2.3.6 Neighborhood Indices

The information about the neighborhood of the spatial objects is frequently used in spatial data mining. For example, density-based clustering algorithms like DBSCAN are based on information about the objects located within *Eps* distance from the currently analyzed point. For the efficient support of the spatial data mining algorithms one should analyze typical operations and data structures that can support different types of knowledge discovery queries. Ester *et al.* proposed the concepts of neighborhood graphs, neighborhood paths and neighborhood indices [37]. The concept of neighborhood can cover different types of spatial relations, including topological relations (e.g., *meet, overlap*, etc.), distance relations (e.g., *close_to*), and direction relations (e.g., *north, west, under*, etc.). The *neighborhood graph* for a relation *neighbor* is a graph where nodes are objects in the database and edges between nodes n_1 and n_2 represent the fact that the relationship $neighbor(n_1, n_2)$ holds. The *neighborhood path* in the neighborhood graph G is a set of nodes directly connected through the edges of the graph. The authors present basic operations for the construction of graphs and for browsing and expansion of paths. The *neighborhood index* is a data structure that allows for the efficient execution of the above operations. It stores all neighbors for the objects in a database.

The presented concepts can be applied for density based clustering, discovery of spatial trends as described in the next subsection, analysis of spatial associations, described in Chapter 3 and construction of decision trees, described in Chapter 4.

2.3.7 Discovery of Spatial Trends

Ester *et al.* analyze spatial trends in [37, 38, 39]. They define spatial trend as "a pattern of change of some non-spatial attribute in the neighborhood of some database object". For example, "when moving away from downtown Warsaw the price of land decreases" is a spatial trend. Two types of trends are analyzed: *global trends* and *local trends*. The first type of trend shows a pattern in all neighborhood directions of the spatial object, while local trends exist only in a particular direction, e.g., along a highway. The neighbors of the regions are progressively analyzed and regression analysis is performed through the extension of neighborhood paths to find possible correlations between distance from the central object and values of the non-spatial attributes. The algorithm starts with the set of directly connected neighbors and, if there is a trend, it analyzes neighbors of the neighbors, i.e., second level neighbors, etc. It ends execution if no trend is detected or if the maximum level of neighbors (the length of neighborhood path) is reached. The neighborhood indices may be used to accelerate the execution process.

2.3.8 Mining in Image and Raster Databases

Knowledge mining from Image and Raster Databases can be viewed as a case of spatial data mining. Data mining in image databases may be seen as similar to image processing. However, data mining studies very large amounts of data, while image processing usually concentrates on analysis of single or a few images. There have been studies, led by Fayyad *et al.* on the automatic recognition and categorization of astronomical objects [43, 44, 132]. These studies are considered as one of the first published studies presenting the results of the analysis of very large databases. At that time there were no such results published even for relational databases.

Fayyad *et al.* presented a system [44] for identifying volcanoes on the surface of Venus from images transmitted by the Magellan spacecraft. The Magellan transmitted more than 30,000 high-resolution synthetic aperture radar images of the surface of Venus from different angles. The system is composed of three basic components: *data focusing*, *feature extraction*, and *classification learning*. Like all other data focusing techniques, the first component increases the overall efficiency of the system by first identifying the portion of the image being analyzed that is most likely to contain a volcano. This is achieved by comparing the intensity of the central pixel of a region to the estimated mean background intensity of its neighborhood pixels. The second component of the system extracts interesting features from the data. Standard methods used in pattern recognition like edge detection or Hough transform, deal poorly with the variability and noise presented in the case of natural data. Since it is difficult to find attributes describing volcanoes exactly, matrices containing volcanoes' images were decomposed to eigenvectors Eigenvalues were treated as attributes describing volcanoes. Then the final task, which is performed by the rest of the system, is to discriminate between volcanoes and other objects looking like volcanoes. Such "false alarms" are caused by objects on the surface of Venus that create deviations of image intensity [12]. The final component of the system uses training examples provided by the experts to create a classifier that can discriminate between volcanoes and "false alarms". The decision tree method [121] was used for this task. The incidence angle of the synthetic aperture radar to the planet instrument strongly influenced images of volcanoes. Thus, the images were normalized according to this angle. The obtained accuracy was about 80%.

In general, it is difficult for experts to provide classifications with 100% certainty and false classifications can produce large errors during classification because they are treated as negative examples. Smyth *et al.* in [132] discussed such issues, using the above problem as a case study. The paper's main contribution is the modeling and treatment of subjective label information given by the experts using probabilistic models. This research is important because it concludes that it is possible for the knowledge discovery methods to be modified to handle the lack of absolute ground truth.

In another study [43] - the Second Palomar Observatory Sky Survey (POSS-II) - decision tree methods were also used for the classification of galaxies, stars and other stellar objects. About 3 TB of sky images were analyzed. Data images were preprocessed by the low-level image processing system FOCAS, which selected objects and produced basic attributes like: magnitudes, areas, intensity, image moments, ellipticity, orientation, etc. Astronomers classified objects in the training data set. Based on this classification, about ten training sets for decision tree algorithm were constructed. From the trees obtained by the learning algorithm, a minimal set of robust, general and correct rules was found. If no additional attributes describing features of a single image plate were used, the accuracy was around 75%. Thus, additional attributes were defined to reach a higher level of accuracy in every image. "Sure-stars" were detected in every image for the purpose of finding image resolution. To gain efficiency, this process was also automated. Using "sure-stars", two additional attributes for every image plate were computed: resolution scale and resolution fraction. These two attributes were used for normalization of attributes describing objects produced by FOCAS. Other attributes, like background level or average intensity were also used to normalize plates. After the normalization, the classification accuracy increased to around 94%. About 5×10^8 objects were classified. Obtained resolution was one magnitude better than the previous astronomical studies and it was possible to classify objects with images too faint to be classified by astronomers. The performance of decision tree methods was compared with neural networks. The tested neural network algorithms were (a) traditional backpropagation, (b) conjugate gradient optimization, and (c) variable metric optimization. The last two algorithms use numerical optimization methods to compute network weights. Different networks were tested. The performance was fairly unstable with accuracy varying from 30% to 95%. An additional drawback of neural networks was the requirement to specify internal parameters such as the number of hidden layers or size. For future investigation, testing of unsupervised clustering techniques is planned.

The above studies showed the problems related to differences between images. The necessity of "normalization" of plates was shown to improve intra- and inter- plate classification.

Stolorz and Dean [136] proposed a system for detecting earthquakes from space. The authors combined methods of statistical interference, massively parallel computing and global optimization to build the system that analyze tectonic activities with subpixel resolution over a large area. The system maps the direction and amplitude of ground displacement with 1-meter precision from remotely sensed images of 10 meters resolution. The system enabled discovery of tectonic activities away from primary faults.

Other examples of raster database mining are Stolorz *et al.*'s [135] and Shek *et al.*'s [130] studies of fast spatio-temporal data mining from geophysical data sets. The authors described the distributed parallel querying and analysis environment called CONQUEST (CONtent-based QUErying in Space and Time). CONQUEST can be distinguished from previously described database mining tools as it also takes into account the temporal components of the datasets and is designed to take advantage of parallel and distributed processing. The main bottleneck during the analysis of very large raster datasets is the extraction of the instances of certain geophysical phenomena like cyclones. To deal with this problem CONQUEST data model defines an algebra, which allows transformation techniques similar to those used in many relational DBMSs, to be applied to geoscientific data. Users may use these operators to write queries for the extraction of the specific phenomena. Moreover, in order to improve query performance, CONQUEST supports pipeline processing, partitioning, and multicasting.

CONQUEST was tested on large climatic datasets[5] to detect cyclones, blocking features, and upward energy propagation. The authors used heuristic rules based on signal processing methods for the extraction of characteristic weather phenomena. Different task decomposition methods were used to facilitate the distribution of work among a group of machines. In the case of cyclone detection, the optimal solution was the decomposition into separate temporal slices. The decomposition in the temporal dimension is not always the best solution, especially when the state plays an important role in the detection of characteristic features. The spatial decomposition, which assigns different blocks of grid points to different machines, was proven to be optimal for the detection of blocking features. The analysis of 2 GB of raw data for the extraction of upward wave propagation events took 143 minutes on one machine and 40 minutes on four machines. As a result, 8610 instances of propagation events were found. After detection of weather phenomena, the authors used a clustering algorithm for the detection of shared spatial features [135]. They also used different visualization techniques like bar charts to present the results of the mining process [130]. The goal of the authors is the building of a system that combines easy formulated queries with fast parallel execution and visualization of results for refinements of the queries. The study

[5]The datasets were chosen so that they were free of incomplete, noisy and contradictory data.

shows that very strong interaction between scientists and the knowledge discovery system is necessary. The users define features of the geoscientific phenomena based on their expert knowledge and the system extracts the instances of such phenomena. Further analysis is then performed on these instances.

2.3.9 Other Methods

The problem introduced by Fayyad *et al.* in [43, 44], has been followed up by other researchers as well. One interesting study was done by Bell *et al.* [12], who proposed a method for knowledge discovery in spatial databases based upon evidence theory [57]. The authors took the image database mining problem described above as a case study. In this study [12] the authors described an extension of general framework for database mining in relational databases based on evidential theory [5] to mine knowledge from spatial databases.

Evidential reasoning [57] is a generalization of conventional probability, in the sense that it does not make any assumptions about the independence of data being analyzed. Therefore, evidential reasoning may be a better choice than using a probabilistic model like the Bayesian method to model the data, like Venus pictures, where pixels may be interrelated. Evidential theory provides a method to combine evidences gathered from different sources to produce a single measure of uncertainty. Thus, it is claimed to be a better method to reason about spatial data in the presence of uncertainty. The combination of evidences is done using a technique based upon Dempster-Shafer theory. Informally, this theory can be regarded as a generalization of the conventional probability theory, where the probabilities are fixed and known in advance, to the case where only the upper and lower bounds on probabilities are available [57]. Bell *et al.* gave an example of how this method can be applied to image databases [12].

Major *et al.* used IXL^{TM} a commercial tool for mining of the tropical storm database [101]. The goal was to predict if hurricanes could reach the U.S. territory. Data describing hurricane paths were decomposed to observations at points. These observations were stored in a traditional relational database. Attributes like position of the hurricane, speed, direction, angle to the coast, etc., were used. Since multiple tuples describing the single hurricane in different points were stored, some data were interdependent. This interdependency of data causes problems because the algorithm which was used assumes independence of data. The best rules according to different criteria like performance, novelty, significance and simplicity were selected from rules derived by IXL. The GIS system was used to support the selection of the best rules. This study shows the necessity of extension of traditional data mining techniques toward spatial data mining for better analysis of complex spatial phenomena and spatial objects.

2.4 Data Warehousing and OLAP

In recent years one can observe the growth of new data repositories: data marts and data warehouses. The **data warehouse** is maintained separately from an organization's operational databases and it provides a solid platform for the storage of consolidated, historical data. It can be defined as "Subject-oriented, integrated, time-invariant, and nonvolatile collection of data in support of management's decision-making process" [76]. In the process of construction of a data warehouse, the snapshot of data from different repositories in the company is combined together and stored in a single place to enable fast access and manipulation for the decision support process. The process of data manipulation to facilitate decision support is known as **on-line analytical processing (OLAP)**. The basic operations and data structures supporting OLAP are different from operations and structures that support daily business operations in the process of **on-line transaction processing (OLTP)**. The OLTP operations typically enable read or update of a few records by many concurrently running transactions, while OLAP enables fast manipulation of large amounts of data by usually a small number of concurrent users.

2.4.1 Multidimensional Data Storage and Manipulation

The business data can be described in terms of dimensions and measures. For example, a sales transaction in a chain of stores can be characterized by store, customer, time, and product dimensions. It can be measured by the quantity of the products bought in the transaction, the profit from the transaction and the total amount of sale. One of the most common data schemas used in **data warehouse servers** is **star schema** presented in Figure 2.10. The star schema consists of a single **fact table** and additional **dimension tables** [82]. An entry in a fact table may represent a single transaction. It contains foreign keys pointing to the dimension tables and numerical values for the measures. The dimension tables contain hierarchical description of the dimension values. For example, *Time* can be described by *time_of_the_day, day, month,* and *year*. Dimension tables are usually left not normalized in order to enable faster data access.

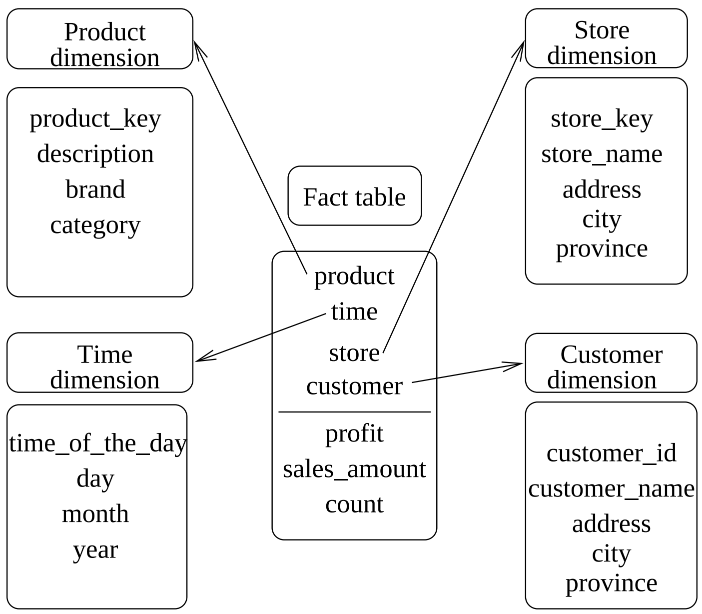

Figure 2.10: An example of star schema of a data warehouse.

The data from a data warehouse server is processed by the **OLAP server** which provides multidimensional view of aggregated data to the *clients*. Clients can be query and reporting tools, analysis and/or data mining tools, etc. The typical OLAP operations are:

- roll-up (increase the level of abstraction). For example, instead of showing data by months one can present data by years.
- drill-down (decrease the level of abstraction, or show more details). For example, instead of showing data by months one can present data by days.
- slice and dice (selection and projection). For example, one can slice data to present sales only for 1998.
- pivot (reorient the multidimensional view of data). For example, instead of presenting a cross table with rows describing years and columns describing regions one can switch columns and rows so that columns would describe years and rows would describe regions.

There are two typical approaches to the construction of OLAP servers: **Relational OLAP** (ROLAP) and **Multidimensional OLAP** (MOLAP). The first approach uses extensions to SQL to efficiently implement multidimensional data and operations. The second approach stores data in special data structures like arrays and cubes and it implements OLAP operations over these structures.

In the process of decision-making business users are usually more interested in general trends than in individual values. Thus, the process of efficient computation of multiple GROUP-BYs and aggregates is very important. One of the most common multidimensional data structures - **data cube** - tries to address this issue. The dimensions form the axes of the cube while the values of the measures are stored in the cube cells. Gray *et al.* proposed a CUBE operator that enables construction of data cubes, which also store aggregate values for the dimensions [55]. An extra value ALL is added in each dimension to contain these aggregate values. This way all groupby aggregations can be presented in a data cube.

The problem with this approach is that data cubes presenting all dimensions on all possible levels of concepts would generate a huge amount of data that would be impossible to store, even on large data servers [72]. Therefore, the selection of data aggregations that should be materialized is very important. Harinarayan *et al.* in [72] proposed a greedy algorithm for the selection of aggregates that are the best to precompute.

2.4.2 Spatial Data Warehousing

Recently Stefanović *et al.* proposed a spatial data warehouse model, which consists of both spatial and non-spatial dimensions and measures [70, 134]. A *spatial measure* is a measure, which contains a collection of pointers to spatial objects. For example, during the process of generalization the regions with the same temperature and precipitation are grouped together. The regions in such measure can be merged. A computed measure can be also used as a dimension in a data cube. In the process of materialization of spatial data cubes, it is important to precompute some spatial OLAP results, such as merge of spatially connected regions. This is essential not only for fast response in spatial OLAP, but also, and often more importantly, for sophisticated, multidimensional spatial data analysis and spatial data mining, such as spatial association, classification, etc. [68]. Spatial OLAP often generates a large number of new spatial objects, which take huge amounts of storage space if they are all materialized. Thus, one has to consider first to materialize the frequently used and shared spatial objects. The work proposes method for spatial cube construction, called *object-based selective materialization* of spatial aggregations [70, 134]. Instead in computing all complete spatial aggregates in a spatial data cube, only the results of the merges of the most frequently combined objects are materialized. Then, when particular spatial aggregates have to be materialized, instead of using a large number of spatial objects the OLAP server uses these partially precomputed results to compute spatial aggregates. Several algorithms are proposed for object-based selective materialization of spatial data cubes, and the performance study has demonstrated the effectiveness of these techniques.

2.5 Summary

The methods for knowledge discovery in relational databases applied to spatial data mining do not yield good results due to a different nature of spatial data [101]. Therefore, new approaches have been tried. Some of these studies were summarized in this chapter. Due to the large size of spatial databases and the complexity of spatial data types the efficiency of the mining algorithms is very important. The spatial data mining algorithms try to achieve good efficiency through:

- Sampling (used mainly for clustering) [58, 35],
- Summarizing information about underling groups of points for clustering [126, 127, 129, 142, 148],
- Partitioning and parallel processing [58, 135, 130, 136],
- Use of spatial approximations [86], and
- Use of spatial data structures like R*-trees [35, 36, 145], neighborhood indices [37], and kd-trees [58].

To deal with complexity of spatial data the researchers tried:

- Generalization methods [86, 99],
- Use of spatial relationships [37], and
- Spatial features extraction [43, 44, 130, 135].

A wide range of pattern extraction methods, from summarization, through clustering, classification of image data to time change analysis were researched. One can observe that the majority of the research in spatial data mining concentrated on clustering and on the exploration of image databases. In recent years algorithms for finding associations and for classification have attracted a lot of attention in the data mining community. These two methods had not really been extended into the field of the exploration of spatial data. We analyze these two areas in the next chapters.

Live in harmony with one another. Do not be proud,
but be willing to associate with people of low position.
Romans 12:16

Chapter 3

Spatial Association Rules

The mining of association rules from transaction databases has attracted a great deal of attention in data mining research [2, 3, 51, 63, 77, 83, 110, 131, 140]. However, there has been a lack of study on mining *spatial association rules*. In this chapter, we extend the techniques for mining association rules in transaction-based databases to mining spatial association rules.

An *association rule* is a general form of dependency rule and is defined on transaction-based databases [3]. It is in the form of "$W \rightarrow B$", explained as, "*if a pattern W appears in a transaction, then the pattern B tends to hold in the same transaction*", where W and B are a sets of attribute values. Formally, the rule "$W \rightarrow B$" in a set $\mathcal{D}$ holds with the confidence c if c% of transactions in $\mathcal{D}$ that contain pattern W also contain pattern B. For example, in a shopping transaction database one may find a rule like "*butter* $\rightarrow$ *bread* (90%)", which means that *90% of customers who buy butter also purchase bread.* Moreover, to ensure that such rules are interesting enough to cover frequently encountered patterns in a database, the concept of the *support* of a rule "$W \rightarrow B$" is introduced, which is defined as the ratio between the patterns of W and B occurring together in the transactions vs. the total number of transactions in the database. Efficient algorithms for the discovery of this kind of rule in transaction-based databases have been studied [2, 3].

Generalization-based spatial data mining methods [99, 109], discussed in Chapter 2, discover spatial and non-spatial relationships at a general concept level, where spatial objects are expressed as merged spatial regions [99] or clustered spatial points [109]. However, these methods cannot discover rules reflecting the structure of spatial objects and spatial/spatial or spatial/non-spatial relationships that contain spatial predicates, such as *adjacent_to, inside, close_to,* and *intersects*. Spatial association rules use spatial and non-spatial predicates to present relationships between objects.

A spatial association rule is a rule of the form "$A \rightarrow B$", where A and B are sets of predicates, some of which are spatial ones [88]. In a large database many association relationships may exist but some may occur rarely or may not hold in most cases. To focus our study on the patterns that are relatively "strong", i.e., the patterns that occur frequently and hold in most cases, the concepts of *minimum support* and *minimum confidence* are introduced [2, 3]. Informally, the **support** of a pattern A in a set of spatial objects $\mathcal{S}$ is the probability that a member of $\mathcal{S}$ satisfies pattern A; and the **confidence** of $A \rightarrow B$ is the probability that pattern B occurs, if pattern A occurs. A user or an expert may specify thresholds to confine the discovered rules to be the *strong* ones, that is, using the patterns that occur frequently and the rules that demonstrate relatively strong implication relationships.

In spatial databases, certain topological relationships hold at all times [33]. Such topological relationships can be viewed as spatial association rules with 100% confidence. For example, the containment relationship expressed in (3.1) is one such association rule.

$$contains(X, Y) \wedge contains(Y, Z) \quad \rightarrow \quad contains(X, Z). \tag{3.1}$$

However, such rules are usually domain-independent, and therefore do not carry much semantic information about specific database contents. An interesting spatial association relationship may not always hold for all the data, but may disclose some important spatial or topological features in the database. For example, one may find that 92% of cities in British Columbia (bc) and adjacent to water, are close to the U.S.A., as shown in (3.2), which associates predicates *is_a*, *in*, and *adjacent_to* with spatial predicate *close_to*.

$$is\_a(X, city) \wedge in(X, bc) \wedge adjacent\_to(X, water) \quad \rightarrow \quad close\_to(X, us).\ (92\%) \tag{3.2}$$

Although such rules are usually not 100% true, they carry some nontrivial knowledge about spatial associations, and thus it is interesting to "mine" (i.e., "discover") them from large spatial databases.

In this chapter, efficient methods for mining spatial association rules are studied, with a top-down, progressively deepening search technique proposed. The technique first searches at a high concept level for frequent patterns at a coarse resolution. Then, only for those frequent patterns, it deepens the search to lower concept levels (i.e., their lower level descendants). Such a deepening search process continues until no frequent patterns can be found. An important optimization technique is that the search for frequent patterns at high concept levels may apply efficient spatial computation algorithms at a coarse resolution (such as coarse generalized close_to, $coarse\_g\_close\_to$), using approximate spatial computation algorithms, such as R-trees or plane-sweep techniques operating on Minimum Bounding Rectangles (MBRs). Only candidate spatial predicates that are worth detailed examination will be computed by refined spatial techniques possibly leading to detailed predicates such as intersects, and contains. Such a multiple-level approach saves much computation, because it is very expensive to perform detailed spatial computation for all the possible spatial association relationships.

The remainder of this chapter is organized as follows. In Section 3.1, the concept of the spatial association rule and its data mining methods are outlined. In Section 3.2, an algorithm for the discovery of spatial association rules is presented. Experimental analysis is presented in Section 3.3. In Section 3.4 we discuss the advantages of the algorithm and its possible extensions. The study is summarized in Section 3.5.

3.1 Concepts

Spatial association rules represent spatial and non-spatial relationships between objects on a map. For example, the following rules are spatial association rules.

- Non-spatial consequent with spatial and non-spatial antecedents.

$$is\_a(X, house) \land close\_to(X, beach) \quad \rightarrow \quad is\_expensive(X).$$

- Spatial consequent with non-spatial antecedent.

$$is\_a(X, gas\_station) \quad \rightarrow \quad close\_to(X, highway).$$

Various kinds of spatial predicates can be involved in spatial association rules. They may represent topological relationships [33] between spatial objects, such as *disjoint, intersects, inside/contains, adjacent_to, covers/covered_by, equal*. They may also represent spatial orientation or ordering, such as *left, right, north, east*, or contain some distance information, such as *close_to, far_from*, etc. These predicates are used in the mining process together with *g_close_to* and *coarse_g_close_to* predicates which are defined as follows.

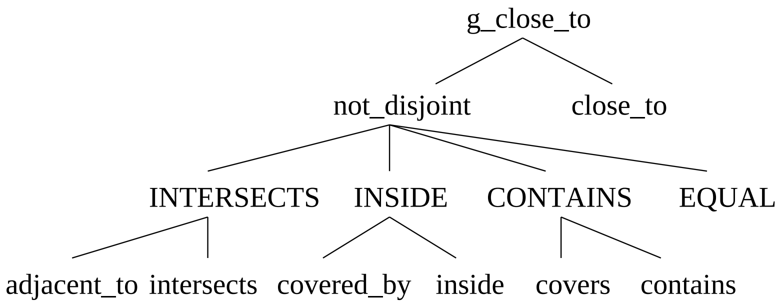

Figure 3.1: The hierarchy of topological relations.

Definition 3.1 The spatial predicate **g_close_to(X, Y)** is satisfied by objects X of type *x_type* and Y of type *y_type* if X and Y are located within the distance threshold d specified for the objects of types *x_type* and *y_type*. □

Definition 3.2 The coarse spatial predicate **coarse_g_close_to(X, Y)** is satisfied by objects X of type *x_type* and Y of type *y_type* if the Minimum Bounding Rectangles for the objects X and Y are located within the distance threshold d specified for the objects of types *x_type* and *y_type*. □

Spatial predicates (topological relations) can also be arranged into a hierarchy. For example, *g_close_to* may cover a set of refined spatial relationships, such as *intersects, inside, contains, equal*, and *close_to*, as shown in Figure 3.1.

For the systematic study of the mining of spatial association rules, we first introduce some preliminary concepts.

Definition 3.3 A **spatial association rule** is a rule in the form of

$$P_1 \wedge \ldots \wedge P_m \quad \rightarrow \quad Q_1 \wedge \ldots \wedge Q_n. \quad (s\%,\ c\%) \tag{3.3}$$

where at least one of the predicates $P_1, \ldots, P_m, Q_1, \ldots, Q_n$ is a spatial predicate, s% is the *support* of the rule, and c% is the *confidence* of the rule. The **support** of a conjunction of predicates, $P = P_1 \wedge \ldots \wedge P_k$, in a set $\mathcal{S}$, denoted as $\sigma(P/\mathcal{S})$, is the number of objects in $\mathcal{S}$ that satisfy P, versus the cardinality (i.e., the total number of objects) of $\mathcal{S}$. The

confidence[1] of a rule $P \rightarrow Q$ in $\mathcal{S}$, $\varphi(P \rightarrow Q/\mathcal{S})$, is the ratio of $\sigma(P \wedge Q/\mathcal{S})$ over $\sigma(P/\mathcal{S})$. A single predicate is called a **1-predicate**. A conjunction of k single predicates is called a **k-predicate**. □

The definition of support and confidence can be extended to adapt quantitative measurements, such as using a weight representing the area of a region object. A weight may be proportional to the area of a region that satisfies the predicate, and the support can be computed as a sum of areas of regions that satisfy a predicate, divided by the total area of regions from the set S. Moreover, fuzzy logic may also be used in the computation of confidence and support of the rules. For example, to compute the support of a fuzzy predicate *close_to(a, b)*, different weights can be used according to the distance between object a and object b.

Most people will only be interested in the patterns that occur relatively frequently (i.e., with *wide support*) and the rules that have strong implications (i.e., with *high confidence*). The rules with large values of support and high confidence are *strong rules*.

Since most people are interested in rules with wide support and high confidence, two kinds of thresholds – *minimum support* and *minimum confidence* – are introduced. Since many predicates and concepts may have strong association relationships at a relatively high concept level, the thresholds should be defined at different concept levels. For example, it is difficult to find strong association patterns between a *particular house* and a *particular beach*. However, there may be strong associations between many *expensive houses* and *luxurious beaches*. Therefore, it is expected that many spatial association rules are expressed at a relatively high concept level.

Definition 3.4 A set of predicates P is **frequent** in set $\mathcal{S}$, at level k, if the support of P, is no less than its minimum support threshold σ'_k for level k, and all ancestors of P from the concept hierarchy, are frequent at their corresponding levels. The confidence of a rule $P \rightarrow Q/\mathcal{S}$ is **high** at level k if its confidence is no less than its corresponding minimum confidence threshold φ'_k. □

[1] The concept of confidence of the rule provides the estimator of conditional probability $Pr(Q \mid P)$, i.e., the probability that Q is satisfied by a randomly chosen object that satisfies P. This concept should not be confused with the concept of *confidence interval* used in statistics. The confidence interval is the interval, that with a certain probability, a value of statistical variable falls into.

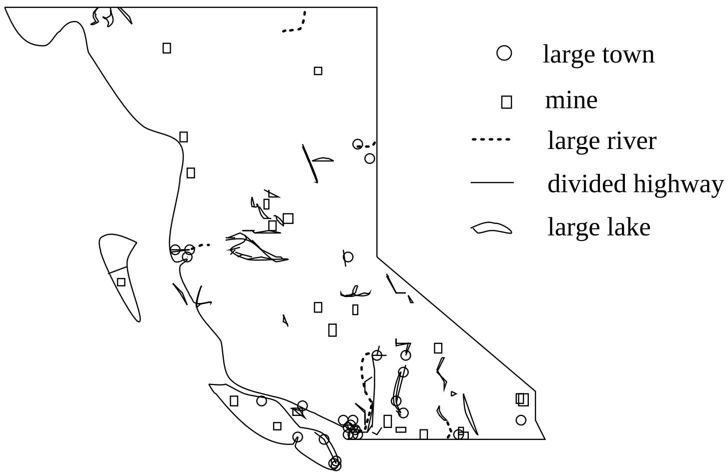

Figure 3.2: The map of BC.

Definition 3.5 A rule $P \rightarrow Q/\mathcal{S}$ is **strong** if predicate $P \wedge Q$ is *frequent* in set $\mathcal{S}$ and the *confidence* of $P \rightarrow Q/\mathcal{S}$ is high. □

Based on these definitions, an example is presented to explain the process of mining strong spatial association rules in large databases. To facilitate the specification of the primitives for spatial data mining, an SQL-like spatial data mining query interface GMQL, which is designed based on Spatial SQL [34], has been specified for an experimental spatial data mining system prototype, GeoMiner.

Example 3.1 Let the spatial database, to be studied, adopt an extended-relational data model and the SAND (spatial-and-non-spatial database) architecture [9]. That is, it consists of a set of spatial objects and a relational database describing non-spatial properties of these objects.

Our study of spatial association relationships is confined to British Columbia, a province in Canada, whose map is presented in Figure 3.2, with the following database relations for organizing and representing spatial objects.

1. $town(town\_name, town\_type, population, geo, \ldots)$.

2. $road(road\_name, road\_type, geo, \ldots)$.

3. $water(water\_name, water\_type, geo, \ldots)$.

4. $mine(mine\_name, mine\_type, geo, \ldots)$.

5. $boundary(type, admin\_region\_1, admin\_region\_2, geo, \ldots)$.

In the above relational schema, the attribute *geo* represents a spatial object (a point, a line, or an area) whose spatial pointer is stored in a tuple of the relation and points to a geographic map. The attribute *type* of a relation is used to categorize the types of spatial objects in the relation. For example, the types for *road* could be {*national highway, local highway, street, back_lane*}, and the types for *water* could be {*ocean, sea, inlet, lake, river, bay, creek*}. The *boundary* relation specifies the boundary between two administrative regions, such as B.C. and U.S.A.. Additional fields may contain other pieces of information, such as the area of a lake and the flow of a river.

Suppose a user is interested in finding the strong association relationships using spatial predicates, describing topological relationships between large towns and other nearby objects including mines, country boundary, water (sea, lake, or river) and major highways. The DMQL query is presented below (the details of DMQL syntax are presented in Chapter 5).

MINE SPATIAL ASSOCIATIONS DESCRIBING "large_towns"
WITH RESPECT TO topology(T.geo, R.geo), R.name,
  topology(T.geo, W.geo), W.name, topology(T.geo, M.geo), M.name,
  topology(T.geo, B.geo), B.admin_region_2
FROM town T, road R, water W, mines M, boundary B
WHERE T.type = "large" **AND** distance(T.geo, R.geo) < "5 km" **AND**
  distance(T.geo, W.geo) < "5 km" **AND** distance(T.geo, M.geo) < "30 km" **AND**
  distance(T.geo, B.geo) < "80 km" **AND** R.type = "divided_highway"
  AND W.type **IN** {sea, ocean, large_lake, large_river}

In this query, the topological predicates *intersects, adjacent_to, contains, close_to* shown in Figure 3.1 are used for the description of BC towns.

Moreover, the *close_to* predicate is a condition-dependent relation and is defined by a set of knowledge rules. For example, Rule (3.4) states if X is a town and Y is a country, then X is close to Y if their distance is within 80 kilometers. However, "close_to" between a town and a road will be defined by a smaller distance as shown in Rule 3.5.

$$close\_to(X,Y) \leftarrow is\_a(X, town) \wedge is\_a(Y, country) \wedge dist(X,Y) < 80\ km. \tag{3.4}$$

$$close\_to(X,Y) \leftarrow is\_a(X, town) \wedge is\_a(Y, road) \wedge dist(X,Y) < 5\ km. \tag{3.5}$$

The distance thresholds for "close_to" relations are specified in the **WHERE** clause of the GeoMiner query. For example, the distance threshold for the relation *close_to(town, road)* can be specified by the distance(Town.geo, Road.geo) $<$ "5 km" condition.

Furthermore, we assume in the B.C. map, admin_region_1 always contains a region in B.C., and thus "U.S.A." or its states must be in "B.admin_region_2". Since there is no constraint on the relation "mine", it essentially means, M.type = "ANY", which is therefore omitted in the query.

To facilitate mining multiple-level association rules and efficient processing, concept hierarchies are provided for both data and spatial predicates.

A set of hierarchies for data relations is defined as follows.

- A concept hierarchy for *towns_name*:

  (town (large_town (big_city(Vancouver, Victoria, ...)),
  (medium_size_city(Prince George, ...)),
  small_town (...))).

- A concept hierarchy for *water_name*:

  (water (sea (strait (Georgia_Strait, ...), inlet (...), ...),
  river (large_river (Fraser_River, ...), ...),
  lake (large_lake (Okanagan_Lake, ...), ...), ...), ...)

- A concept hierarchy for *road_name*:

  (road (national_highway (route1, ...),
  provincial_highway (highway_7, ...),
  city_drive (Hastings St., Kingsway, ...),
  city_street (E_1st Ave., ...), ...), ...)

- A concept hierarchy for *boundary*:

  (boundary (us_boundary, alberta_boundary, yukon_boundary)).

□

3.2 A Method for Mining Spatial Association Rules

3.2.1 An Example of Mining Spatial Association Rules

Example 3.2 We examine how the data mining query posed in Example 3.1 is processed, which illustrates the method for mining spatial association rules.

Firstly, the set of relevant data is retrieved by execution of the data retrieval methods [9] of the data mining query, which extracts the following data sets: (1) towns: only large towns; (2) roads: only divided highways[2]; (3) water: only seas, oceans, large lakes and large rivers; (4) mines: any mines; and (5) boundary: any boundary.

Secondly, the "coarse generalized close_to" (*coarse_g_close_to*) relationship between large towns and the other four sets of entities is computed at a relatively coarse resolution level using a less expensive spatial algorithm such as the MBR data structure and a plane sweeping algorithm [119], or R*-trees and other approximations [19]. The derived spatial predicates are collected in a "coarse_g_close_to" table (Table 3.1), which follows an extended relational model: each slot of the table may contain a set of entries. The support of each top-level predicate is then computed and objects related to the large towns, through predicates, whose support is below the minimum support threshold, are removed from the table. In our example entries for all "mines" are deleted.

Town	Water	Road	Boundary	Mine
Victoria	Juan_de_Fuca_Strait	highway_1, highway_17	US	
Saanich	Juan_de_Fuca_Strait	highway_1, highway_17	US	
Prince_George		highway_97		
Pentincton	Okanagan_Lake	highway_97	US	Alalla
...	...	...	...	...

Table 3.1: The computed "coarse_g_close_to" relation.

Notice that from the computed *coarse_g_close_to* relation, frequent predicates can be discovered at different concept levels and the spatial association rules can be presented accordingly. For example, the following two spatial association rules can be discovered from

[2] Not all the segments of national and provincial highways in Canada are divided ones, our computation only counts the divided ones. Also, "provincial divided highway" is abbreviated to "provincial highway" in later presentations.

this relation.

$$is\_a(X, large\_town) \ \rightarrow \ coarse\_g\_close\_to(X, water). \ \ (80\%, \ 80\%)$$

$$is\_a(X, large\_town) \wedge coarse\_g\_close\_to(X, water) \ \rightarrow \ coarse\_g\_close\_to(X, boundary). (40\%, 92\%)$$

The detailed computation process is not presented here since it is similar to mining association rules for exact spatial relationships to be presented below.

Since many people may not be satisfied with approximate spatial relationships, such as *coarse_g_close_to*, more detailed spatial computation often needs to be performed, to find the refined (or precise) spatial relationships in the spatial predicate hierarchy. Thus, we have the following steps.

Refined computation is performed on the frequent predicate sets, i.e., those retained in the *coarse_g_close_to* table. Each *coarse_g_close_to* predicate is replaced by one or many concrete predicate(s) such as *intersect, adjacent_to, close_to, inside*, etc. Such a process results in Table 3.2 in which each entry of Table 3.1 is replaced by one or a set of precise spatial predicates.

Town	Water	Road	Boundary
Victoria	⟨adjacent_to, J.Fuca_Strait⟩	⟨intersects, highway_1⟩, ⟨intersects, highway_17⟩	⟨close_to, US⟩
Saanich	⟨adjacent_to, J.Fuca_Strait⟩	⟨intersects, highway_1⟩, ⟨close_to, highway_17⟩	⟨close_to, US⟩
Prince_George		⟨intersects, highway_97⟩	
Pentincton	⟨adjacent_to,Okanagan_Lake⟩	⟨intersects, highway_97⟩	⟨close_to, US⟩
...	...	...	...

Table 3.2: Detailed spatial relationships for frequent item sets.

Table 3.2 forms a base for the computation of detailed spatial relationships at multiple concept levels. The level-by-level detailed computation of frequent predicates and the corresponding association rules is carried out as follows.

The computation starts at the top-most concept level of each concept hierarchy and computes frequent predicates at this level. For example, for each row of Table 3.2 (i.e., each large town), if the *water* attribute is nonempty, then the count of water is incremented by one. Such count accumulation forms 1-predicate rows (with $k = 1$) of Table 3.3 where the support count is registered. If the count (support) of a row is smaller than the minimum support threshold, the row is removed from the table. For example, if the minimum support is set to 50% at level 1, then the rows with count less than 20 are removed from the table. The 2-predicate rows (i.e., $k = 2$) are formed by the pair-wise combination of the frequent 1-predicates. The counts of 2-predicate rows are accumulated by checking against Table 3.2. The rows with the count smaller than the minimum support will be removed. Similarly, the 3-predicates are computed. Thus, the computation of frequent k-predicates results in Table 3.3.

k	frequent k-predicate set	count
1	⟨adjacent_to, water⟩	32
1	⟨intersects, road⟩	29
1	⟨close_to, road⟩	29
1	⟨close_to, boundary⟩	28
2	⟨adjacent_to, water⟩, ⟨intersects, road⟩	25
2	⟨adjacent_to, water⟩, ⟨close_to, boundary⟩	23
2	⟨close_to, boundary⟩, ⟨intersects, road⟩	26
3	⟨adjacent_to, water⟩, ⟨close_to, boundary⟩, ⟨intersects, road⟩	22

Table 3.3: Frequent k-predicate sets at the top concept level (for 40 large towns in B.C.).

Spatial association rules can be extracted directly from Table 3.3. For example, since the predicate "⟨intersects, road⟩" has a support count of 29, and the predicate "⟨adjacent_to, water⟩, ⟨intersects, road⟩" has a support count of 25 (i.e., the support of the rule is $25/40 \doteq 62\%$), and the confidence can be calculated as $25/29 \doteq 86\%$, we have the association rule (3.6).

$$is\_a(X, large\_town) \wedge intersects(X, road) \quad \rightarrow \quad adjacent\_to(X, water). \ (62\%,\ 86\%) \tag{3.6}$$

Notice that a predicate $is\_a(X, large\_town)$ is added in the antecedent of the rule since the rule is related only to large_town (which is taken as the base in the support computation).

Similarly, one may derive another rule (3.7). However, if the minimum confidence threshold was set to 75%, this rule (with only 72% confidence) would have been removed from the list of the association rules to be generated.

$$is\_a(X,\ large\_town) \wedge adjacent\_to(X,\ water) \rightarrow close\_to(X,\ boundary). \ (57.5\%, 72\%) \tag{3.7}$$

After mining rules at the highest levels of concept hierarchies, frequent k-predicates can be computed in the same way at the lower concept levels, which results in Tables 3.4 and 3.5. Notice that, at the lower levels, usually the minimum support and possibly the minimum confidence may need to be reduced in order to derive enough interesting rules. For example, the minimum support of level 2 is set to 25% and thus, the row with a support count of 10 is included in Table 3.4, whereas the minimum support of level 3 is set to 15% and therefore, the row with a support count of 7 is included in Table 3.5. The analyzed concepts exist on the same levels of the hierarchies. For example, *sea* is the concept from the second level of the *water* hierarchy, and *provincial_highway* is the concept from the second level of the *road* hierarchy. If the number of levels in concept hierarchies differs, then the lowest concepts from the "shallow" hierarchies are used together with the concepts found on the lower levels of the "deep" hierarchies. In our example the *boundary* hierarchy has only two levels and the concept *us_boundary* is used both at the second level and at the third level.

k	frequent k-predicate set	count
1	⟨adjacent_to, sea⟩	21
1	⟨adjacent_to, large_river⟩	11
1	⟨close_to, us_boundary⟩	28
1	⟨intersects, provincial_highway⟩	21
1	⟨close_to, provincial_highway⟩	24
2	⟨adjacent_to, sea⟩, ⟨close_to, us_boundary⟩	15
2	⟨close_to, us_boundary⟩, ⟨intersects, provincial_highway⟩	19
2	⟨adjacent_to, sea⟩, ⟨close_to, provincial_highway⟩	11
2	⟨close_to, us_boundary⟩, ⟨close_to, provincial_highway⟩	22
3	⟨adjacent_to, sea⟩, ⟨close_to, us_boundary⟩, ⟨close_to, provincial_highway⟩	10

Table 3.4: Frequent k-predicate sets at the second level (for 40 large towns in B.C.).

k	frequent k-predicate set	count
1	⟨adjacent_to, georgia strait⟩	9
1	⟨adjacent_to, fraser_river⟩	10
1	⟨close_to, us_boundary⟩	28
2	⟨adjacent_to, georgia_strait⟩, ⟨close_to, us_boundary⟩	7

Table 3.5: Frequent k-predicate sets at the third level (for 40 large towns in B.C.).

Similarly, spatial association rules can be derived directly from the frequent k-predicate set tables at levels 2 and 3. For example, Rule (3.8) is found at level 2, and Rule (3.9) is found at level 3.

$$is\_a(X, large\_town) \rightarrow adjacent\_to(X, sea). \ (52.5\%,\ 52.5\%) \tag{3.8}$$

$$\begin{aligned} is\_a(X, large\_town) \quad \wedge \quad & adjacent\_to(X, georgia\_strait) \rightarrow \\ & close\_to(X, us\_boundary). \ (17.5\%,\ 78\%) \end{aligned} \tag{3.9}$$

Notice that only the descendants of the frequent 1-predicates are examined at a lower concept level. For example, the number of large towns adjacent to a lake is small and thus "⟨adjacent_to, lake ⟩" is not represented in Table 3.4. Then the predicates such as "⟨adjacent_to, okanagan_lake ⟩" will not even be considered at the third level. The mining process stops at the lowest level of the hierarchies or when an empty frequent 1-predicate set is derived.

As an alternative to the problem, towns may also be further partitioned into *big_cities* (such as towns with a population larger than 100,000 people), *medium_size_cities*, etc., and rules like rule (3.10) can be derived by a similar mining process.

$$is\_a(X, big\_city) \wedge adjacent\_to(X, sea) \rightarrow close\_to(X, us\_boundary). \ (9\%,\ 100\%) \tag{3.10}$$

3.2.2 An Algorithm for Mining Spatial Association Rules

The above rule mining process can be summarized in the following algorithm.

Algorithm 3.1 Mining the spatial association rules defined by Definition 3.3 in a large spatial database.

Input: The input consists of a spatial database, a mining query, and a set of thresholds as follows:

1. A database, that consists of three parts (1) a spatial database, *SDB*, containing a set of spatial objects, (2) a relational database, *RDB*, describing non-spatial properties of spatial objects, and (3) a set of concept hierarchies;
2. A query, that consists of (1) a reference set $\mathcal{S}$ of described objects, (2) task-relevant sets of spatial objects $C_1, ..., C_n$ that are used for description of the objects from $\mathcal{S}$, and (3) a set of task-relevant spatial relations; and

3. Two thresholds for each level l of description: minimum support ($minsup[l]$) and minimum confidence ($minconf[l]$).

Output: Strong multiple-level spatial association rules for the relevant sets of objects and relations.

Method: Mining spatial association rules proceeds as follows:

Step 1: *Task_relevant_DB* := extract_task_relevant_objects(*SDB, RDB*);

Step 2: *Coarse_predicate_DB* :=
SAM_coarse_computation(*Task_relevant_DB*);

Step 3: *Frequent_coarse_predicate_DB* :=
filtering_with_minimum_support(*Coarse_predicate_DB*);

Step 4: *Fine_predicate_DB* :=
refined_spatial_computation(*Frequent_coarse_predicate_DB*);

Step 5: Find_frequent_predicates_and_mine_rules(*Fine_predicate_DB*);

Explanation of the detailed steps of the algorithm.

Step 1 is accomplished by the execution of a spatial query, which is constructed based on GMQL query. All task-relevant objects are collected into one database: *Task_relevant_DB*.

Step 2 is accomplished by execution of an efficient spatial access method at a Minimum Bounding Rectangle resolution level. For example, R-trees [18] or fast MBR techniques and plane-sweep algorithms [119] can be applied to extract the objects that are approximately close to each other, i.e., objects whose MBRs are within a specified distance threshold. This operation corresponds to computing *coarse_g_close_to* predicates for the *Task_relevant_DB*. The efficiency of this method is analyzed in the next subsection. Predicates describing spatial relations between objects are stored in an extended relational database, called *Coarse_predicate_DB*, which allows an attribute value to be either a single value or a set of values (i.e., in non-first-normal form).

Step 3 computes the support for each predicate in *Coarse_predicate_DB*, (and registers them in a predicate-support table), and filters out those entries whose support is below the minimum support threshold, at the top level. This filtering process results in a database *Frequent_coarse_predicate_DB*, which contains all frequent 1-predicates. Notice that spatial association rules can also be generated at this resolution level, if desired. Since this process is similar to the process of Step 5, the detailed processing of Step 3 is not presented here.

Step 4 is accomplished by execution of an efficient spatial computation algorithm [19] at a fine resolution level on *Frequent_coarse_predicate_DB* obtained in Step 3. Notice that although such computation is performed for the interesting portion of the spatial database, the computation is only on those pairs that have passed the corresponding spatial testing at a coarse resolution level. Thus, the number of object pairs that need to be computed at this level is substantially smaller than the number of pairs computed at a coarse level. Moreover, as an optimization technique, the support count of an approximate predicate in *Frequent_coarse_predicate_DB* can be used to predict whether there is still a chance for a predicate at a fine level to pass the minimum support threshold. For example, if the current support for predicate P plus the remaining number of support for its corresponding predicate *coarse_P* is less than the minimum support threshold, no further test of P is necessary in the remaining processing.

Step 5 computes the frequent k-predicates for all the k's and generates the strong association rules at multiple concept levels. This step is essential for mining multiple-level association rules, and thus, it is examined in detail.

This step is outlined as follows. First, obtain frequent k-predicates (for all the k's) at a top concept level. Second, for the frequent 1-predicates at level 1, get their corresponding frequent 1-predicates at level 2, and then get all frequent k-predicates at this level. This process repeats until an empty frequent 1-predicate set is returned or bottom level is reached in all used hierarchies. A detailed study of such a progressive deepening process for mining multiple-level association rules in a transaction-based (but not spatial) database is presented in [63].

At each level, the computation of frequent k-predicates for all k's proceeds from computing frequent-1 predicates, then frequent-2 predicates (using the pair-wise combinations of frequent 1-predicates as the candidate set), frequent-3 predicates (using the combinations of frequent 2-predicates as the candidate set), and so on, until an empty candidate set, or an empty computed k-predicate set, is obtained. Such a process of computing frequent k-predicate sets (called frequent k-itemsets in [3]) using previously computed $(k-1)$-predicate sets in a transaction-based database is studied in [3], and is called *Algorithm Apriori.*

Notice that this k-predicate sets computation algorithm is fairly efficient, since it generates candidate k-predicate sets by full exploration of the combination of $(k-1)$-predicate sets before testing the k-predicate pairs against the predicate database. For example, Table 3.4 contains frequent 2-predicates "⟨adjacent_to, sea⟩, ⟨close_to, us_boundary⟩" and "⟨close_to, us_boundary⟩, ⟨intersects, provincial_highway⟩" but does not contain "⟨adjacent_to, sea⟩, ⟨intersects, provincial_highway⟩". Therefore, a candidate 3-predicate "⟨adjacent_to, sea⟩, ⟨close_to, us_boundary⟩, ⟨intersects, provincial_highway⟩" cannot be formed. Thus, the effort of testing such a 3-predicate against the predicate database can be saved.

After finding frequent k-predicates, the set of association rules for each level l can be derived based on the minimum confidence at this level, $minconf[l]$. This is performed as follow [3]. For every frequent n-predicate A, if m-predicate B is not a subset of A, the rule "$A \rightarrow B$" is added into the result of the query if $support(A \wedge B)/support(A) \geq minconf[l]$.

The process is summarized in the following procedure, where $\mathcal{LL}[l]$ is the frequent predicate set table at level l, and $\mathcal{L}[l,k]$ is the frequent k-predicate set table at level l. The syntax of the procedure is similar to C and Pascal.

procedure find_frequent_predicates_and_mine_rules(DB);
  for ($l := 1$; $\mathcal{L}[l,1] \neq \emptyset$ and $l < max\_level$; l++) do begin
    $\mathcal{L}[l,1] := get\_frequent\_1\_predicate\_sets(DB, l)$;
    for ($k := 2$; $\mathcal{L}[l,k-1] \neq \emptyset$; k++) do begin
      $P_k := get\_candidate\_set(\mathcal{L}[l,k-1])$;
      foreach object s in S do begin
        $P_s := get\_subsets(P_k, s)$; {Candidates satisfied by s}
        foreach candidate $p \in P_s$ do $p.support++$;
      end;
      $\mathcal{L}[l,k] := \{p \in P_k \mid p.support \geq minsup[l]\}$;
    end;
    $\mathcal{LL}[l] := \bigcup_k \mathcal{L}[l,k]$;
    output := generate_association_rules($\mathcal{LL}[l]$);
  end
end $\square$

In this procedure, line (2) shows that the mining of the association rules is performed level-by-level, starting from the top-most level, until either the frequent 1-predicate set table is empty or it reaches the lowest concept level. For each level l, line (3) computes the frequent 1-predicate sets and puts it into table $\mathcal{L}[l,1]$. Lines (4)-(11) compute the frequent k-predicate sets $\mathcal{L}[l,k]$ for all $k>1$ at the level l progressively, essentially using the Apriori algorithm [3], as we discussed above. Line (12) collects all the frequent k predicate at each level l into one table $\mathcal{LL}[l]$, and finally line (13) generates the spatial association rules at each concept level from the frequent predicate table $\mathcal{LL}[l]$. □

The generated rules may need to be examined by human experts or pass through some automatic rule quality testing program [118] in order to filter out some obvious or redundant rules, and output only those fairly new and interesting ones to the users.

Correctness of the algorithm.

The Algorithm 3.1 discovers the correct and complete set of association rules given by the Definition 3.3.

Step 1 is a query processing process, which extracts all data relevant to the spatial data mining process, based on the completeness and correctness of query processing. Step 2 applies a coarse spatial computation method, which computes the whole set of relevant data and thus still ensures its completeness and correctness. Step 3 filters out those 1-predicates whose support is smaller than the minimum support threshold. Obviously, predicates filtered out are those, which have no chance to generate rules with support reaching the minimum support. Step 4 applies a fine spatial computation method, which computes predicates from the set of derived coarse predicates, and thus, still ensures the completeness and correctness based on the nature of the spatial computation methods. Finally, Step 5 ensures to find the complete set of association rules, at multiple concept levels, based on the previous studies at mining multiple-level association rules in transaction-based databases [3, 63]. Therefore, the algorithm discovers the correct and complete set of association rules.

3.3 Performance Evaluation

We have the following theorem for the efficiency of the Algorithm 3.1.

Theorem 3.1 *Let the maximal cost for computing each spatial predicate at fine resolution level be C_f, and the number of objects in the database be n. The worst-case time complexity* *of Steps 2-5 of Algorithm 3.1 is* $\mathcal{O}(n \times logn + m + C_f \times m + C_{non-spatial})$*, where* m *is the number of predicates computed in the relevant spatial data sets on the coarse level and* $C_{non-spatial}$ *is the total cost of rule mining in the predicate databases.*

Proof

Step 1 applies a spatial database query processing method, whose computational complexity has been excluded from the total cost of the computation, according to the statement of the theorem.

Step 2 involves the computation of the largest set of spatial predicates, since each pair of objects needs to be checked to see whether it potentially and approximately satisfies the predicate to be coarsely computed. In this step the intersections between pairs of MBRs of n objects are computed. Bentley and Wood [14] proved that an optimal worst case algorithm for reporting intersections of rectangles has the time complexity $\mathcal{O}(n \times logn + m)$, where n is the number of rectangles, and m is the number of intersecting pairs of rectangles which result in coarse predicates. One may notice that in the worst case, every object can be related to each other object in the database which would result in $\mathcal{O}(n^2)$ predicates.

Step 4 involves the computation of the spatial predicates at a refined level. More detailed spatial computation algorithms will be applied at this stage. Since there are no more than m predicates to be finely computed in the relevant data sets, and the worst-case cost of computing each spatial predicate at a fine resolution level is C_f, the maximum processing cost at this step should be $\mathcal{O}(C_f \times m)$. In the worst-case, if all possibly existing coarse predicates are satisfied and pass support threshold test in Step 3 of the algorithm, then $\mathcal{O}(n^2)$ of fine predicates have to be computed.

According to the algorithm, the computation of support counts, threshold testing, and rule generation will not involve further spatial computation. Thus, the total computation cost for Steps 3 and 5 will be $\mathcal{O}(C_{non-spatial})$, where $C_{non-spatial}$ is the total cost of rule mining in a non-spatial predicate database.

Adding all costs together, we have the formula presented in the theorem. □

For a detailed look at the cost of Step 5, one may find that for each level l, the support for every 1-predicate is computed by traversing through the computed refined predicate database once (no spatial computation is involved!), accumulating the count in a 1-predicate table ($k = 1$). Then, the frequent 1-predicate set can be obtained by examining the 1-predicate table. The frequent 1-predicate set can be used to get candidate 2-predicate sets,

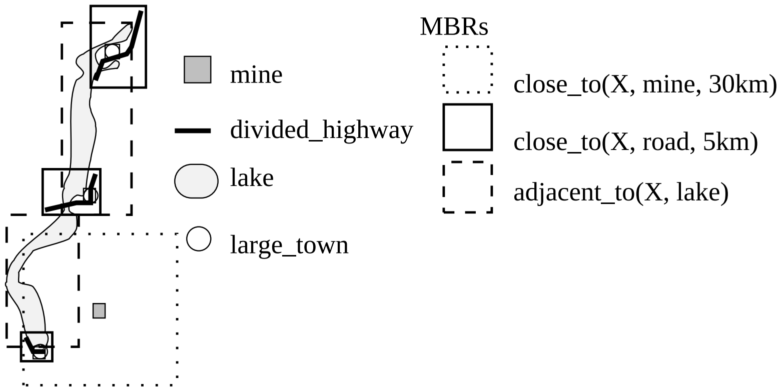

Figure 3.3: MBRs for some coarse spatial relations.

etc. Adding all such table scans together for each level l, the computed refined predicate database needs to be scanned in total p times, where $p = \sum_l (k_l + 1)$, and k_l is the maximum k such that the k-predicate table $\mathcal{L}[l, k]$ is nonempty at level l.

Performed experiments showed that the cost of mining association rules in a predicate database is not a bottleneck, i.e., this cost is not the main contributor to the total cost of mining spatial association rules (see Table 3.7).

It is important to explain the progressive refinement optimization technique in a little more detail. To avoid checking the predicates which will not be used later in the fine computation, approximate computation can be performed at a coarse resolution level. To accelerate this process, every object can be described using its MBR and coarse predicates can be derived using the R-tree technique for spatial join [18] or the plane sweep technique [119]. Large objects like the lake in Figure 3.3 can be decomposed into smaller ones. The method checks for *not_disjoint* relation between rectangles related to the relations such as *close_to, inside, adjacent_to* and rectangles bounding the described objects. It can be done during a single pass over all rectangles sorted according to the minimal x coordinates. Possible relations are stored in the extended relational database *Coarse_predicate_DB*. Then only pairs of objects with intersecting MBRs related to predicates need to be checked in detail, using more expensive refined spatial computations.

Furthermore, for faster computations one may use the generalized and approximated data. For example, sinuousity of lines can be reduced, and small regions can be converted to points, etc.

Execution time of the above mining algorithm can be estimated using the results of spatial join computations based on real data [18, 19] and on our experience on mining multilevel association rules [63]. The time of finding multiple level association rules using Algorithm 3.1 is presented by Equation 3.11. Table 3.6 lists some parameters used in the cost analysis. The component $C_c^{'} \times F_{dr} \times N \times log(N)$ of this equation presents the execution time of step 2 of the algorithm; $C_{filter} \times d \times F_{dr} \times N$ is the time of filtering infrequent coarse predicates; $F_{ratio} \times d \times F_{dr} \times N \times C_f$ presents the execution time of finding fine predicates; and $F_{ratio} \times d \times F_{dr} \times N \times N_{nsp}$ presents mining association rules from the set of fine predicates. Curve "*coarse+filter+fine*" in Figure 3.4 shows the execution time of algorithm 3.1. In the case where the filtering in Step 3 of the algorithm is not used it takes t_2 time (Equation 3.12) to mine rules as shown by curve "*coarse+fine*" in Figure 3.4. Execution time of naive algorithm, when no tree structure is used for finding coarse predicates, can be computed using Equation 3.13. This time is presented by curve "*naive+filter+fine*" in Figure 3.4. Estimated times shown in Figure 3.4 indicate a substantial improvement of performance in large databases when tree structure is used to compute coarse predicates. It also shows a large acceleration of computation process by filtering out coarse predicates not leading to frequent predicates, which avoids fine computations on such predicates.

$$\begin{aligned}
t_1 &= C_c^{'} \times F_{dr} \times N \times log(N) + C_{filter} \times d \times F_{dr} \times N + F_{ratio} \times d \times F_{dr} \times N \times C_f \\
&\quad + F_{ratio} \times d \times F_{dr} \times N \times N_{nsp} && (3.11) \\
t_2 &= C_c^{'} \times F_{dr} \times N \times log(N) + d \times F_{dr} \times N \times C_f + d \times F_{dr} \times N \times N_{nsp} && (3.12) \\
t_3 &= C_c^{''} \times F_{dr} \times N \times N + C_f \times F_{ratio} \times N_c + F_{ratio} \times d \times F_{dr} \times N \times C_f \\
&\quad + F_{ratio} \times d \times F_{dr} \times N \times N_{nsp} && (3.13)
\end{aligned}$$

We performed a number of experiments with Washington State data from the TIGER[3] dataset [141]. MapInfo Professional 4.1[4] Geographic Information System [102] was used as a spatial database for the experiments. A number of queries were executed. Spatial association rules for objects such as rivers, local parks, churches, schools, etc., were discovered.

Figure 3.5 compares the relative times of the query execution describing local parks when the filter is used ("*coarse+filter+fine*" curve) to the time when the filter is switched off and all fine predicates are computed ("*coarse+fine*" curve). The time of computations with the

[3] TIGER is the registered trademark of U.S. Census Bureau.

[4] MapInfo Professional is the registered trademark of MapInfo Corporation.

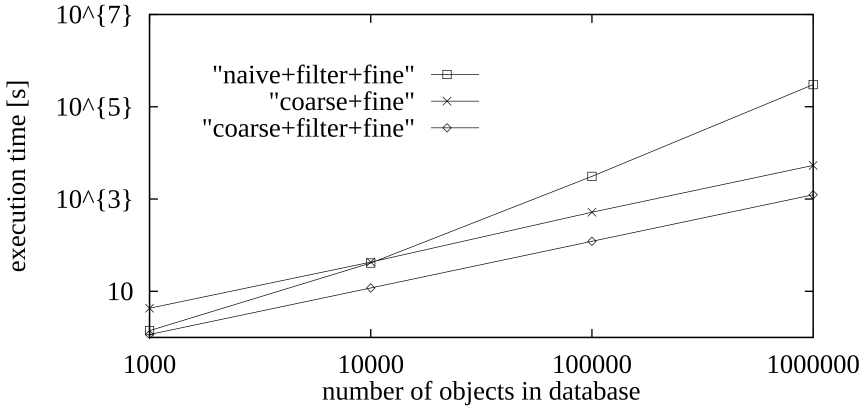

Figure 3.4: Execution time.

Name	Value	Meaning
N		number of objects in the database
F_{dr}	0.01	percentage of objects being described
d	8	average number of predicates describing an object
$C_c^{'}$	1 ms	constant for finding one fine coarse predicate using tree structure
$C_c^{''}$	0.03 ms	constant for finding one fine coarse predicate using naive algorithm
C_f	50 ms	constant for computing one fine predicate
C_{nsp}	1.5 ms	constant for finding association rules in a predicates database
C_{filter}	0.5 ms	constant for filtering out predicates in step 3 of the algorithm 3.1
F_{ratio}	0.25	ratio of coarse predicate possibly leading to frequent predicates

Table 3.6: Database parameters.

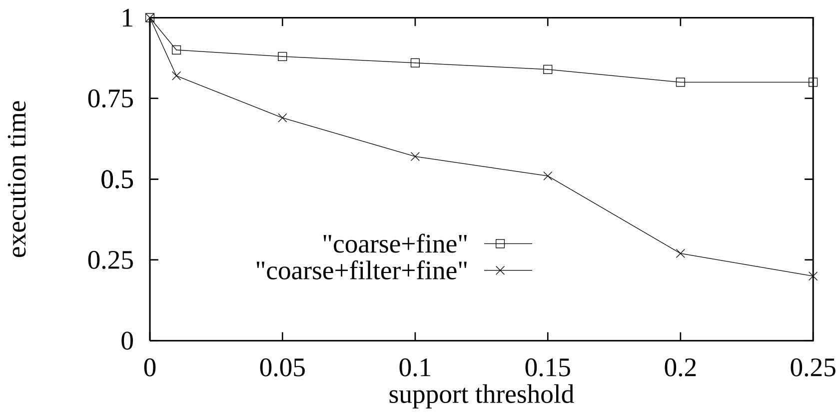

Figure 3.5: Execution time.

filter switched off decreases with increasing support threshold because the time for mining association rules from the set of fine predicates (Step 5 of the Algorithm 3.1) decreases with the support threshold as there are less frequent k-predicates to process. The time of computations with a filter decreases even faster with the increasing support threshold. This happens because, with the increasing support threshold, more predicates are filtered out on the coarse level and less fine computations are needed.

In our queries we varied distance thresholds for the *g_close_to* predicate used for the description and we also varied support threshold values. The query execution time depended on different factors such as: distance threshold (time was increasing with increased distance), the size and the complexity of described objects (time was increasing with increased size and complexity of objects), and the support threshold (time was decreasing with increased support threshold). The time for coarse predicate finding and filtration was 5% to 45% of the total query running time and time for the mining of association rules from the database of fine predicates was 3% to 37% of the total query running time. When the running time of the query was long, due to the large size of the described objects, the query execution time was dominated by the computations of the fine predicates which contributed up to 90% of the total time. When described objects were simple ones, like for example points, then the time for the computations of the fine predicates made about 40% of the total time.

Table 3.7 compares the execution times for different types of objects when the support threshold was set to 20%. The experiments were performed on a Pentium 300MHz machine.

described objects	buffer size	no_filter total [s]	coarse & filter [s]	fine [s]	rules [s]	with_filter total [s]	time reduction	comments
churches	1 km	11	3	3	2	8	27%	point
	2 km	20	6	4	6	16	20%	objects
schools	1 km	18	2	3	1	8	55%	point
	2 km	30	12	6	6	14	20%	objects
cemeteries	1 km	18	2	5	2	9	50%	point
	2 km	24	4	7	3	14	42%	objects
parks	1 km	39	3	9	1	13	67%	small region
	2 km	54	6	26	7	39	28%	objects
rivers	1 km	136	8	73	2	83	39%	large region objects

Table 3.7: Execution times for support threshold = 20%.

The values in "*no_filter total*" column present times for mining association rules when no filtration based on coarse predicates is performed. The values in "*with_filter total*" column present times for mining association rules when filtration based on coarse predicates is used. The values in the "*coarse & filter*" column present times for mining coarse predicates and their filtration. The values in the "*fine*" column present times for mining fine predicates after the filtration of predicates. The values in the "*rules*" column present times for mining rules from the database of fine predicates.

3.4 Discussion

In this section we present the major strengths of our method for mining of spatial association rules, and we discuss some of the possible modifications and extensions of the progressive refinement approach for mining spatial association rules.

3.4.1 Major Strengths of the Method

The data mining method for discovery of spatial association rules presented in previous section has the following major strengths:

- Focused data mining guided by user's query.

  The data mining process is directed by a user's query, which specifies the relevant objects and spatial association relationships to be explored. This not only leads to desirable results, but it also confines the mining process to a relatively small set of data and rules for efficient processing.

- Approximate spatial computation: Substantial reduction of the candidate set.

  Less costly, but approximate spatial computation is performed at an abstraction level first on a large set of data, which substantially reduces the set of candidate data to be examined in the future.

- Detailed spatial computation: Performed once and used for knowledge mining at multiple levels.

  The computation of support counts at each level can be performed by scanning through the same computed spatial predicate table.

- Optimizations on computation of k-predicate sets and on multiple-level mining.

  These two optimization techniques are shared with the techniques for mining other (i.e., non-spatial) multiple-level association rules [63]. First, it uses the $(k-1)$-predicate sets to derive the candidate k-predicate sets at each level, which is similar to the *Apriori* algorithm developed in [3]. Second, it starts at the top-most concept level and applies a progressive deepening technique to examine, at a lower level, only the descendants of the frequent 1-predicates, which is similar to the technique developed in [63].

- Easy modification of the algorithm for parallel and distributed processing.

  We also implemented the parallel version of the algorithm. The results were presented in [90]. They show that the algorithm performs well even when spatial indices are not available. The algorithm can also be used for mining in distributed spatial databases [1] where the thematic maps are located in different places. In this case, each site would create its own predicate database for its theme and these predicate databases could be processed by a query coordinator. The main data transfer cost would be related to the shipment of the described objects together with the shipment of frequent predicate DBs.

- User-controlled interactive mining.

  Users may control, usually via a graphical user interface (GUI), minimum support and minimum confidence thresholds at each abstraction level interactively based on the currently returned mining results.

3.4.2 Further Progressive Refinement

Algorithm 3.1 may be improved by using the idea of an additional geometric filter for the filtration of frequent predicates. This idea was introduced by Brinkhoff *et al.* for multi-step spatial join processing [19]. It was summarized in Section 2.2 of this thesis.

The modernized algorithm may be presented as follows.

Algorithm 3.2 Mining the spatial association rules defined by Definition 3.3, where the g_close_to predicates are used as spatial predicates using spatial access method and additional geometric filter.

Input: Same as in Algorithm 3.1.

Output: Strong multiple-level spatial association rules for the relevant sets of objects and g_close_to predicates.

Method: Mining spatial association rules proceeds as follows.

Step 1: *Task_relevant_DB* := extract_task_relevant_objects(*SDB, RDB*);
Step 2: *Coarse_predicate_DB* :=
SAM_coarse_computation(*Task_relevant_DB*);
Step 3: *Frequent_coarse_predicate_DB* :=
filtering_with_minimum_support(*Coarse_predicate_DB*);
Step 4: *Geo_filtered_coarse_predicate_DB* :=
geo_filter_spatial_computation(*Frequent_coarse_predicate_DB*);
Step 5: *Frequent_geo_filtered_coarse_predicate_DB* :=
filtering_with_minimum_support(*Geo_filtered_coarse_predicate_DB*);
Step 6: *Fine_predicate_DB* :=
refined_spatial_computation(*Frequent_geo_filtered_coarse_predicate_DB*);
Step 7: Find_frequent_predicates_and_mine_rules(*Fine_predicate_DB*);

Explanation of the algorithm.

The first three steps of the Algorithm 3.2 are the same as Steps (1)–(3) of the Algorithm 3.1.

Step 4 uses progressive and conservative geometric approximations for further filtration of predicates that are not frequent. In this step, progressive approximations such as *Maximum Enclosed Rectangle* (MER) or *Maximum Enclosed Circle* (MEC) are used to produce hits (i.e., objects which are within specified distance thresholds). Such hits are equivalent to fine g_close_to predicates and thus, they are marked in *Geo_filtered_coarse_predicate_DB* to avoid further processing by the *refined_spatial_computation* procedure. Conservative approximations such as Minimum Bounding 5-Corner (MB5C) are used to eliminate false hits that were not detected by the spatial access method used to perform spatial join in Step 2. They produce *Geo_filtered_coarse_predicates* (i.e., objects whose MB5C approximations are within specified distance thresholds). The experiments in [19] showed that by using MB5C, over 60% of previously unidentified false hits were detected.

Step 5 of the algorithm computes the support for the second level coarse predicates in *Geo_filtered_coarse_predicate_DB* and eliminates predicates with supports smaller than *minsup[1]*.

Step 6 uses the exact geometry processor to check if unmarked coarse predicates in *Frequent_geo_filtered_coarse_predicate_DB* satisfy the conditions of fine predicates. Such fine predicates are stored in *Fine_predicate_DB* together with fine predicates computed using progressive approximations in Step 4 and satisfying minimum support condition checked in Step 5.

Step 7 computes frequent k-predicates and mines spatial association rules in the same manner as Step 5 of the Algorithm 3.1.

Discussion of the algorithm.

The additional filtering step allows for the further elimination of predicates that have no chance to satisfy the minimum support threshold on the fine resolution level. This way the most computationally expensive operations, to find fine predicates, are limited to the predicates that satisfy the support threshold on coarser levels of resolution. Moreover, the progressive approximations allow for the faster computations of fine predicates. The work in [19] shows that by adding the additional geometry filter, the cost of spatial computations may be reduced by 40%. In the process of spatial association rule mining, using the Algorithm 3.2, there is an additional filtration cost on the level of *Geo_filtered_coarse_predicates*. This cost should be alleviated by a smaller number of predicates that have to be computed on the fine level after the filtration. According to [19], over 60% of false hits might be detected on a coarse level. □

3.4.3 Meta-rule Spatial Association Rules

In the process of mining association rules, despite the use of support and confidence thresholds, many rules in a form not interesting to the user, may be discovered. *Meta-rule guided mining* is an interactive approach that introduces additional constraints on the form of resulting rules.

Previous studies on metarule mining include [83] that uses metarules for filtering interesting association rules, [131] that uses metarules to guide mining of Bayesian data clustering rules, [51] that uses a relation table-based structure for mining multilevel association rules, and [77] that uses data cubes for metarule guided mining of association rules.

Definition 3.6 A **metarule** is a rule template in the form of

$$P_1 \wedge \ldots \wedge P_m \quad \rightarrow \quad Q_1 \wedge \ldots \wedge Q_n. \tag{3.14}$$

Where P_i (for $i = 1, \ldots, m$) and Q_j (for $j = 1, \ldots, n$) are either instantiated predicates or predicate variables [51]. □

For example, Rule 3.15

$$is\_a(X, city) \wedge close\_to(X, Y) \wedge R(Y, D) \wedge intersects(X, Z) \rightarrow Q(Z, C). \tag{3.15}$$

is a metarule.

Definition 3.7 A spatial association rule R complies with a metarule M_R if R can be unified with M_R. □

With Metarule 3.15 a rule

$$is\_a(X, city) \wedge close\_to(X, Y) \wedge is\_a(Y, airport) \wedge intersects(X, Z) \rightarrow type(Z, divided\_highway).$$

complies.

In the Algorithms 3.1 and 3.2 the process of filtering predicates that cannot be frequent was based on 1-predicates. In the process of meta-rule guided mining with templates from the Definition 3.6, m+n-predicates are used. Therefore, we have the following theorem which results may be used to speed-up the mining process.

Theorem 3.2 *If a fine spatial predicate S_i is used in a metarule M_R, which is formed based on the set P_k of frequent k-predicates, then the coarse version of P_k is frequent. (In the coarse version of the set of k-predicates P_k, the coarse spatial predicates are used instead of the corresponding fine ones)*

Proof

We know that all objects that satisfy fine predicates also must satisfy coarse predicates. Thus, the number of objects involved in coarse k-predicates is at least equal to the number of objects involved in fine k-predicates, and, if a fine k-predicate is frequent, then the corresponding coarse k-predicates must also be frequent. □

The experiments performed with GeoMiner show that the number of frequent k-predicates decreases quickly when values of k increase. The number of distinct spatial predicates used in the frequent k-predicates also decreases, when values of k increase. Using this observation, Algorithms 3.1 and 3.2 can be modified to filter out even larger number of spatial predicates that cannot be used in the meta-rule guided mining. Based on the meta-rule template, specified according to the Definition 3.6 and Theorem 3.2, coarse 1-predicates that are not used in the construction of frequent coarse m+n-predicates cannot be used in the construction of frequent fine m+n-predicates. Thus, such predicates would not be used in rules complying to meta-rules and do not have to be computed on fine resolution level.

Multi-set Spatial Metarules

One may notice that Metarule 3.15 contains spatial predicates that associate only a single set of objects (e.g., cities) with other sets of objects (e.g., airports and roads). We call such metarule a *single-set spatial metarule* in the manner similar to single variable metarule [51]. It might be interesting to investigate multi-set spatial metarules, i.e., metarules that use spatial predicates to mutually associate many sets of spatial objects. For example, Metarule 3.16 is such a metarule.

$$is\_a(X, airport) \wedge close\_to(X, Y) \wedge R(Y, D) \wedge intersects(Y, Z) \rightarrow Q(Z, C). \qquad (3.16)$$

In this metarule spatial predicate *close_to* associates *airports* with objects from set Y which are in turn related to objects from set Z, by spatial relationship *intersects*. Formally, we have the following definitions for single-set spatial metarules and multi-set spatial metarules.

Definition 3.8 A **single-set spatial metarule** is a rule template in the form of

$$P_1 \wedge \ldots \wedge P_m \wedge S_1(d, c_1) \ldots \wedge S_k(d, c_k) \rightarrow P_{m+1} \wedge \ldots \wedge P_n \wedge S_{k+1}(d, c_{k+1}) \ldots \wedge S_l(d, c_l). \quad (3.17)$$

Where $S_1, \ldots, S_l$ are instantiated spatial predicates or variable spatial predicates describing spatial relationships between a single set of spatial objects D with other sets of objects $C_1, \ldots, C_l$; and $P_1, \ldots, P_n$ are instantiated non-spatial predicates or variable non-spatial predicates describing non-spatial properties of objects from sets $C_1, \ldots, C_l$, and D. The support is measured in relation to the number of objects in class D. □

Definition 3.9 A **multi-set spatial metarule** is a rule template in the form of

$$P_1 \wedge \ldots \wedge P_p \wedge S_1(d, c_1) \ldots \wedge S_k(c_i, c_j) \rightarrow P_{p+1} \wedge \ldots \wedge P_r \wedge S_{k+1}(c_l, c_m) \wedge \ldots \wedge S_l(c_n, c_o). \quad (3.18)$$

Where $S_1, \ldots, S_l$ are instantiated spatial predicates or variable spatial predicates describing spatial relationships between sets of spatial objects $D, C_1, \ldots, C_o$; and $P_1, \ldots, P_r$ are instantiated non-spatial predicates or variable non-spatial predicates describing non-spatial properties of objects from sets $C_1, \ldots, C_l$, and D. Furthermore, the objects from sets $C_1, \ldots, C_o$, and D are mutually related via spatial predicates, i.e. if one constructs a graph in which nodes correspond to sets of objects and edges to spatial predicates associating two sets of objects, such a graph would be connected. The support is measured in relation to the number of objects in class D. □

Further investigation of the algorithms for mining spatial association rules is necessary to accommodate efficient discovery of multi-set spatial metarules.

Constraint Association Rules

Recently the concept of constraint association rules was introduced in [110]. A *constraint association query* is defined to be a query in the form: $\{(S_1, S_2) \mid \mathcal{C}\}$ where S_1 and S_2 are sets of items (or predicates) used for the head and body of the rule, and $\mathcal{C}$ is the set of conditions specified on S_1 and S_2. A similar concept was used in query flocks proposed in [140]. Ng *et al.* introduced two types of constraints, *anti-monotonicity* and *succinctness*, that are critical to the pruning process [110]. The analysis of the possibility of pushing these constraints into coarse predicates should be performed in the future.

3.4.4 Statistical Spatial Association Relationships

After the discovery of interesting strong spatial association rules, a user can ask questions about the statistical strength of rules. Such questions would be equivalent to performing a *confirmatory spatial data analysis* (CSDA) after an *exploratory spatial data analysis* (ESDA) process [7] of mining strong spatial association rules.

Spatial autocorrelation [54] is widely used in *exploratory spatial data analysis* [7]. The techniques such as Moran's I, Geary's or Gamma indices are used to detect if spatially distributed values of a non-spatial attribute are dependent on neighboring regions, or if such distribution is random.

The mentioned techniques are mostly used to find dependencies between neighboring regions on a map and they are based on a single attribute. These dependencies would relate to associations such as "*pollution(X, high)* $\wedge$ *meet(X, Y)* $\rightarrow$ *pollution(Y, high)*". In the case of spatial association, objects such as points and lines can also be used in the construction of the rules. The measures of spatial association were extended to associations between points [97, 138], and associations between points and other objects [15]. Lotwick and Silverman [97] proposed summary statistics and distribution theory of the spatial association estimators for the analysis of mutually independent stationary Poisson point processes. They also suggest Monte Carlo tests for the independence of two spatial point processes in a rectangular region. Berman [15] proposed a method for testing if a point process and another spatial process are independent. The method uses different tests under the Poisson assumption and the Monte Carlo approach when the Poisson assumption is relaxed.

Brin *et al.* proposed an extension of association rules toward finding correlation [17]. The authors measure correlation using the chi-square test for correlation from classical statistics. This approach allows users to find not only the presence of particular relationships, but also the absence of patterns and provides users with the statistical significance of the discovered patterns. The chi-square statistic, has however, limitations as it rests on the normal approximation to binomial distribution.

It would be interesting to statistically validate spatial association rules discovered using the algorithms presented in this chapter. The extension of the current work, using methods similar to the approach presented in [17], is an interesting research topic. However, it seems that because of the lack of general statistical measures of spatial associations, the validation of the discovered patterns would involve costly Monte Carlo simulations.

3.4.5 Other Issues

Mining Cross-Level Spatial Association Rules

We may relax the restriction of mining strong spatial associations among the predicates at the *same levels* of concept hierarchies to allow the discovery of "*cross-level*" associations. This relaxation may lead to the discovery of rules such as "*is_a(X, large_town)* $\wedge$ *adjacent_to(X, georgia_strait)* $\rightarrow$ *close_to(X, provincial_highway).* (20%, 70%)", where the concept *georgia_strait* is from the third level of *water* hierarchy and *provincial_highway* is the concept from the second level of *road* hierarchy. This relaxation is similar to association mining when hierarchies with different depths are used, such as *boundary* (with depth 2) and *water* (with depth 3) hierarchies in Example 3.2. As in the case of different depths of the hierarchies the predicates using concept from different levels would be stored in the predicate database. The algorithm for mining cross-level association rules in transaction databases was presented in [52]. This algorithm may be directly adopted to mining in the database of spatial predicates. Readers should notice that if the rules were mined on all combinations of levels of concept hierarchies, a huge amount of rules could be discovered. Therefore, in OLAM approach [71] a user usually specifies the levels which should be analyzed. A user guides the whole mining process, and only rules on the levels specified by him/her are presented. Such a focused approach is practically necessary in order to get meaningful results in a reasonable time.

Search for Outliers

Spatial association rules show characteristics of objects that are widely represented in the database. If a researcher is interested in finding the descriptions of the patterns that characterize only a small number of objects, such as outliers, then the discovered association rules cannot be used directly to describe these objects. To analyze outliers other data mining techniques, such as deviation analysis, are used.

There has been a significant amount of work on outliers in the field of statistics [10]. Most of the statistical tests are based on a single attribute. Knorr and Ng [87] proposed efficient algorithms for finding distance-based outliers in large multidimensional datasets. Such methods should be analyzed with regard to spatial data. The methods used for the characterization of spatial objects proposed in [39] may be explored in the direction of finding outliers. Also, spatial association rules with a very high support can be used to analyze objects that do not satisfy such rules. For example, if there is a rule with the confidence 99.8%, then one can analyze the properties of objects that do not satisfy this rule.

The statistical spatial autocorrelation techniques presented in the Subsection 3.4.4 are used to find an indicator of global spatial association relationships or to determine the existence of global spatial autocorrelation patterns based on values of one attribute. When analysts are interested in local "hot spots" in spatial distribution of an attribute, then local indicators of spatial association such as LISA [6], or G_i and G_i^* statistics [27] come to help with the identification of local pockets of non-stationarity, i.e., associations around single objects, or assess if the global pattern is reflected uniformly throughout the dataset [7]. Such small areas can exist in the data even though a single global measure of spatial autocorrelation indicates that the process is random. [143] presents an overview of other statistical methods for finding outliers in spatial data.

Measures of Rules Interestingness

When the process of mining spatial association rules proceeds to deeper levels of concept hierarchies, the values of support and confidence change. While the values of support do not increase, values of confidence may increase, remain the same, or decrease. It is interesting to analyze the behavior of these changes. Srikant and Agrawal [133] introduced the concept of *interestingness* of rules to find rules with confidence and support significantly different from expected values. This concept of interestingness is based on the comparison of a rule to the rules containing ancestor itemsets. The authors call an itemset $\hat{Z}$ an *ancestor* of the itemset Z, when we can get $\hat{Z}$ by replacing items with their ancestors from the concept hierarchies and both $\hat{Z}$ and Z have the same number of items.

The rule $X \rightarrow Y$ is *R-interesting* w.r.t. its ancestor $\hat{X} \rightarrow \hat{Y}$ if the support of the rule $X \rightarrow Y$ is more than R times the expected support ,based on $\hat{X} \rightarrow \hat{Y}$, or the confidence is more than R times the expected confidence, based on $\hat{X} \rightarrow \hat{Y}$, or $X \rightarrow Y$ has no ancestors.

For example, we may have a situation that 20% of cemeteries are close to religious sites, 30% are close to parks, and 16% are close to both religious sites and parks. In this case we have the following rule:

$$is\_a(X, cemetery) \wedge intersects(X, religious\_site) \rightarrow adjacent\_to(X, park).\ (16\%,\ 80\%) \quad (3.19)$$

If 10% of cemeteries are close to mosques and 15% of cemeteries close to small parks, then we may expect the rule:

$$is\_a(X, cemetery) \wedge intersects(X, mosque) \quad \rightarrow \quad adjacent\_to(X, small\_park).\ (4\%,\ 40\%) \quad (3.20)$$

because based on the formulas from [133], the expected support of the Rule (3.20) is 4% ($10\%/20\% \times 15\%/30\% \times 16\%$), and the expected confidence is 40% ($15\%/30\% \times 80\%$). If the computed values of support and confidence differ significantly from the expected values, then such rule should be presented to a user. If the support is larger than the expected support then the predicates which form the rule tend to exist together more frequently than expected. In our case it would mean that cemeteries tend, more often to be close to both mosques and small parks, then it would be expected based on random data distribution. If the confidence of the rule is larger than the expected confidence it means that the predicates in the head of the rule are stronger associated with the predicates in the body of the rule than it would be expected based on random distribution. In our case it would mean that cemeteries, which are close to mosques tend to be more often close to small parks, than it would be expected based on random data distribution. [133] provides a more detailed discussion of the rule interestingness and of the analysis of the changes of confidence and support when the mining process goes down through the concept hierarchies.

Precomputations of Spatial Relations in Spatial Association Rules Mining

In the process of mining spatial association rules the spatial join operation is performed to extract spatial predicates. A number of data structures was proposed for efficient processing of spatial join queries.

Rotem [123] introduced the concept of *spatial join indices* to speed up spatial query processing. This concept was extended by Lu and Han [98] who introduced *distance-associated join indices*. These indices associate distance with pairs of objects. To minimize space requirements (in native form spatial join index would require $O(n^2)$ space) a hierarchical structure was introduced. A spatial hierarchy of objects (e.g., *countries* > *provinces* > *cities*) is used and the index is created only for pairs of objects which have the same parent in the hierarchy.

Ester, *et al.* proposed neighborhood indices for spatial data mining [37]. This concept was presented in Subsection 2.3.6. For example, in mining of spatial association rules, if the *g_close_to(church, X, "1 km")* predicate is used, and a neighborhood index exists for such a relationship, then this neighborhood index can be directly used to build frequent predicates tables for mining of the rules. In this case, the coarse computations would not be necessary. However, clearly it is prohibitively expensive to construct indices for each spatial relationship because of possibly infinite number of relationships, such as *g_close_to* for different distances. Therefore, only some of the most important indices should be materialized, in order to speed up the mining process. Also, the predicates materialized during the execution of the previous queries can be reused. However, even when some of the indices exist, and an index for the relationship used in the mining process does not exist, the progressive refinement approach still could be used. Therefore, only the most important indices could be materialized. When an index for the relationship used in the mining process does not exist, the progressive refinement approach still should be used. For example, if *g_close_to(church, X, "2 km")* predicate is used for mining and there is only neighborhood index for *g_close_to(church, X, "3 km")* predicate then the more general index could be used to filter out infrequent predicates.

3.5 Summary

Based on the previous studies on spatial data mining and mining association rules in transaction-based databases, we proposed and studied a method for mining strong spatial association rules in frequent spatial databases. The discovery of spatial association rules may disclose interesting relationships among spatial and/or non-spatial data, in large spatial databases, and thus, it represents a new and promising direction in spatial data mining.

The method developed in this chapter explores efficient mining of spatial association rules at multiple approximation and abstraction levels. It proposes first to perform less costly, approximate spatial computation to obtain approximate spatial relationships at a high abstraction level. Then, fine spatial computations are performed only for data or predicates, which can contribute to the discovery of strong association rules. Such a multi-step spatial mining method facilitates mining strong spatial association rules at multiple concept levels by a top-down, progressively deepening technique.

We discussed possible extensions of the algorithm. The proposed approach can be extended using additional approximations other than MBRs and it can be used for mining of metarules. It can also be adopted for parallel and distributed processing. The discovered association rules can be validated using statistical methods.

The mining of spatial association rules has been incorporated in spatial data mining prototype *GeoMiner* [68, 93]. The experiments performed with the system show that interesting rules may be discovered from frequent amount of data very efficiently. The paper [88] describing the mining of spatial associations was used in senior-level and graduate education at Hong Kong University of Science and Technology, Anna University in Chennai, India, University of Nebraska at Omaha, and Simon Fraser University.

They are to teach my people the difference between the holy and the common and show them how to distinguish between the unclean and the clean.
Ezekiel 44:23

Chapter 4

Spatial Classification

One of the areas which has not attracted much attention in spatial data mining is the classification of spatial objects. Classification is a data mining technique where the data stored in a database is analyzed in order to find rules that describe the partition of the database into the given set of classes. Each object in a database (in relational databases, a tuple is treated as an object) is assumed to belong to a predefined class, as it is determined by one of the attributes, called the *class label attribute*. A number of classification methods was proposed by statistics and machine learning researchers [46, 121, 122, 124]. The most common classification method uses *decision trees*. Such a method employs a top-down, divide-and-conquer strategy that partitions the set of given objects into smaller subsets where the *leaf* nodes are associated mostly with a single class. Decision tree methods can deal both with numerical and symbolic data. Other common statistical techniques deal effectively with either continuous or discrete attributes, but not both [122]. Many statistical classifiers and all neural network methods represent information as equations, while decision trees can be transformed into rules, which is important for decision making, as the professionals may prefer to make decisions based on understandable information rather than based on a "black box".

Most of the methods for classification consider only relational data. Geographic data consists of spatial objects and non-spatial descriptions of these objects. Non-spatial descriptions of spatial objects can be stored in a traditional relational database where one attribute is a pointer to the spatial description of the object [9]. In the process of spatial classification one wants to find rules that divide a set of objects into a number of classes using not only non-spatial properties of the classified objects, but also their spatial relations to other objects in the database.

In this chapter we address issues regarding classification of spatial data and concentrate on building decision trees for the classification of this data. Furthermore, we analyze the problem of the classification of spatial objects in relevance to thematic maps and spatial relations to other objects in the database. Finally, we propose an algorithm that deals with the large amount of irrelevant spatial relations and we present the results of a spatial classification process for both synthetic and real datasets.

The rest of the chapter is organized as follows. In Section 4.1, we describe relevant work. Section 4.2 presents an in-depth analysis of the problem of spatial classification. An algorithm for building decision trees from spatial data is shown in Section 4.3. Possible extensions are presented in Section 4.4. Experimental analysis is presented in Section 4.5, and the chapter ends with a summary in Section 4.6.

4.1 Relevant Work

Some researchers have analyzed classification of spatial data. Fayyad *et al.* [45] used decision tree methods to classify images of stellar objects to detect stars and galaxies. About 3 TB of sky images were analyzed. Data images were preprocessed with the low-level image processing system FOCAS, which selected objects and produced basic attributes such as magnitudes, areas, intensity, image moments, ellipticity, and orientation. Objects in the training dataset were classified by astronomers. Based on this classification, approximately ten training sets, for a decision tree algorithm, were constructed. From the decision trees a minimal set of robust, and general rules was found. The method deals with image databases and is tailored to astronomical applications. Thus, it is not suitable for the analysis of vector data, often used in Geographic Information Systems.

A method for classification of spatial objects using neighborhood graphs was proposed by Ester *et al.* [37]. Their algorithm is based on the ID3 algorithm [121]. It considers not only non-spatial properties of the classified objects, but also non-spatial properties of neighboring objects. Objects are treated as neighbors if they satisfy a neighborhood relation, which may be one of the following:

- Topological relations, such as *meet, overlap and inside.*
- Metric relations, such as *close_to.*
- Direction relations, such as *south, north, west, and east.*

For example, cities $\{O_c\}$ may be classified in relevance to the economic power of the regions that are their neighbors. Since the influence of neighboring objects decreases with increasing distance, the authors limited the length of the "neighborhood paths".

The method [37] does not take into account aggregate values of non-spatial attributes for the neighboring objects. For example, if a city is close to three regions with medium populations, such a city may have similar properties as a city close to a single region with a high population. Their algorithm does not perform relevance analysis either and, therefore, it may produce overspecialized, poor quality trees. The method is based on the existence of neighborhood indices, and if they do not exist they must be constructed for every neighborhood relation, which may result in poor performance. Finally, the proposed algorithm does not take into account concept hierarchies that may exist for the non-spatial and spatial attribute values.

Ng and Yu [111] proposed a method for the extraction of strong, common, and discriminating characteristics of a given set of clusters. They proposed measures for interest/utility values of the characteristics of clusters. In the search for common characteristics of the clusters, the algorithm selects themes whose values are the most similar for a number of clusters. In the process of finding discriminating characteristics of the clusters, their algorithm selects themes that discriminate between two sets of clusters in the best way. Unfortunately, only properties of thematic maps are analyzed and this approach is not extended toward the construction of decision trees.

4.2 Problem Description

The goal of spatial classification is to find rules that divide a set of objects $\{O_c\}$ into a number of groups, where objects in each group belong mostly to one class. Spatial objects $\{O_c\}$ may be characterized by many types of information. We can classify such information into the following categories:

- Non-spatial attributes of objects (including both classified objects O_c and other objects used for description), such as *the number of sales persons in a store.*
- Spatially related attributes with non-spatial values, such as *population living within 1km from a store.*
- Spatial predicates, such as $distance\_less\_than\_10km(store, shopping\_center)$.
- Spatial functions, such as $driving\_distance(store, beach)$.

Each of these categories may be used to extract values both for *class label attribute* (e.g., attribute that divides data into classes) and *predicting attributes* (e.g., attributes on whose values the decision tree is branched). We can use aggregate values for some of these attributes. For example, calculations of the size of the population living within 1 kilometer from a store may involve finding the sum of the population of all census blocks intersecting a one-kilometer buffer around the store. Moreover, there may exist concept hierarchies for these attributes, which can result in building simpler decision trees.

A number of questions can be associated with spatial classification:

- Which attributes/predicates/functions are relevant to the classification process?
- How should one determine the size of the buffers that can be used to build high accuracy decision trees? Should it be defined by the experts or found using an algorithm?
- What should be taken into account when aggregates are used:
  - aggregate values for all objects intersecting buffers, or
  - aggregate values only for parts of objects intersecting buffers?
- Can one accelerate the process of finding relevant attributes using
  - sampling,
  - statistical or machine learning methods,
  - rough computations followed by detailed computations for promising patterns, or
  - a combination of the above methods?
- How should one handle multiple levels of concepts?

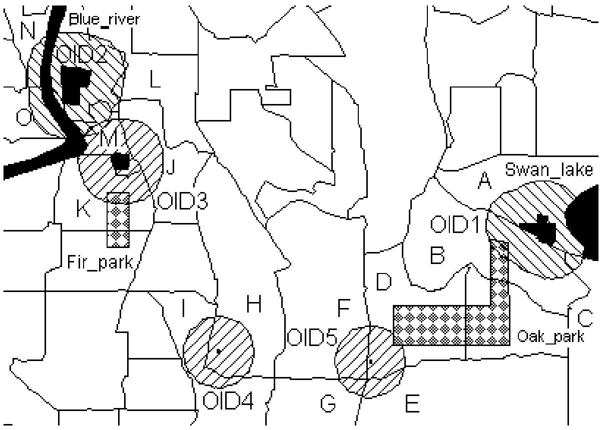

Figure 4.1: Example map

Example 4.1 We want to build a decision tree that classifies five objects O_i (for example, shopping malls or stores) that belong to two different classes Y and N, which are selected based on attribute *high_profit*, with two values, Y for "yes" and N for "no". In our example objects OID1 and OID2 belong to class Y and objects OID3, OID4 and OID5 belong to class N (Figure 4.1). We want to build a decision tree that classifies objects $\{O_i\}$ based on two types of information: (1) descriptions of the objects in the proximity of objects $\{O_i\}$ and (2) non-spatial attributes of the thematic map. The thematic map in our example consists of groups of census blocks. Table 4.1 shows non-spatial descriptions of some of the block groups. We assume that objects are characterized by areas that are close to them. Thus, one should build buffers around objects $\{O_i\}$ and construct the decision tree based on census block groups intersecting buffers and other objects located within the buffers. □

In the construction of the decision tree we use information gain utilized in the ID3 algorithm [121] to find attributes dividing datasets into classes. Information gain is computed in the following way. If there are n objects from class N and y objects from class Y, one can calculate information as:

$$I(y,n) = -\frac{y}{y+n} log_2 \frac{y}{y+n} - \frac{n}{y+n} log_2 \frac{n}{y+n} \tag{4.1}$$

If attribute A with m values is used to partition the data into m classes one can calculate the expected information as:

BlockID	population	avg_income	crimes
A	5000	$25,500	50
B	4500	$27,500	40
C	5500	$28,300	45
H	6000	$31,500	50
I	5000	$25,500	30
...	...	...	...

Table 4.1: Description of census block groups

$$E(A) = \sum_{i=1}^{m} \frac{y_i + n_i}{y + n} I(y_i, n_i) \tag{4.2}$$

where y_i and n_i are the number of objects from classes Y and N, respectively, for each of the m values of the attribute.

Finally, one can calculate information gain if attribute A is used to split dataset as:

$$info\_gain(A) = I(y, n) - E(A) \tag{4.3}$$

The attribute with the highest information gain is used to split the dataset. Other measures for building decision trees, such as the error based Gini Diversity Index [16], statistical methods [106], and class separation [42], may also be used to find the best attribute to split dataset.

4.2.1 Handling of Spatial Predicates and Functions

Spatial functions like *driving_distance* or spatial predicates like *g_close_to* can be used to describe classified objects as it is presented in Table 4.2. The description should be generalized and then the decision tree algorithm should be applied. For example, generalized data is presented in Table 4.3.

The process of finding which objects satisfy spatial predicates may be very time consuming. To accelerate this process we may use a multi-step approach in which some rough computations are performed first and then fine computations are done only for the promising

ObjectID	high_profit	others
1	Y	g_close_to(x,Oak_park), g_close_to(x,Swan_lake)
2	Y	g_close_to(x,Blue_river)
3	N	g_close_to(x,Fir_park), g_close_to(x,Blue_river)
4	N	
5	N	g_close_to(x,Oak_park)

Table 4.2: Descriptions of classified objects

ObjectID	high_profit	others
1	Y	g_close_to(x,park), g_close_to(x,water)
2	Y	g_close_to(x,water)
3	N	g_close_to(x,park), g_close_to(x,water)
4	N	
5	N	g_close_to(x,park)

Table 4.3: Generalized descriptions of classified objects

patterns. Such a process is similar to the multi-step approach for mining spatial association rules presented in Chapter 3, or to the multi-step spatial join approach [19].

In the first step we can find coarse descriptions for only a sample of objects. For example, one can use Minimum Bounding Rectangles (MBRs) to find *coarse_g_close_to* predicates, which imply that MBRs of two objects are within a specific distance threshold. Then, some machine learning methods may be used for the extraction of the relevant predicates or functions [84, 144]. In our experiments, we used the RELIEF algorithm [84] whose algorithmic description is presented in Figure 4.2. The RELIEF algorithm uses a nearest neighbor approach to find relevant predicates. For every object s in the sample, two nearest neighbors are found, where one neighbor belongs to the same class as object s (nearest hit) and the other neighbor belongs to a class different from s (nearest miss). Based on the descriptions of the nearest neighbors, weights for attributes are modified in the following way:

1. If the neighbor belongs to the same class as object s and has the same predicate value then the weight for this predicate stays the same.

2. If the neighbor belongs to the same class as object s and does not have the same

predicate value then the weight for this predicate decreases, i.e., the predicate is judged irrelevant.

3. If the neighbor belongs to a different class than object s and has the same predicate value, then the weight for this predicate stays the same.

4. If the neighbor belongs to a different class than object s and does not have the same predicate value, then the weight for this predicate increases, i.e., the predicate is judged relevant.

One can observe that weights for relevant attributes have positive values, while expected weights for irrelevant attributes are zero. Finally, only predicates with weights larger than the predefined threshold are used for classification. The value for the threshold can be set based on experimental results or based on a statistical theory [84]. In our case only relevant predicates are computed in detail for all classified objects. Besides improving the efficiency of the algorithm, the accuracy of the classification and the size of the decision tree may improve as well, because there is no branching on irrelevant predicates.

```
Procedure RELIEF(Predicate_weight, Predicate_relevant, k)
 FOR j := 1 TO max_predicate DO
     Predicate_weight[j] := 0;
 FOR sample_i := 1 TO min(200, k) DO /* size of the sample is set to min(200, k) */
     nearest_hit := FIND_NEAREST_HIT(sample_i); /* find nearest sample from the same class */
     nearest_miss := FIND_NEAREST_MISS(sample_i); /* find nearest sample from different class */
     FOR j := 1 TO max_predicate DO
         Predicate_weight[j] := Predicate_weight[j] - diff(sample_i, nearest_hit, j)
                                                    + diff(sample_i, nearest_miss, j);
 FOR j := 1 TO max_predicate DO
     IF Predicate_weight[j] > min(200, k) * threshold/sqrt(min(200, k))
         THEN Predicate_relevant[j] = TRUE
         ELSE Predicate_relevant[j] = FALSE;

 Function diff(sample_i, nearest_hit_or_miss, j)
   /* for symbolic attributes */
       IF sample_i[j] = nearest_hit_or_miss[j] THEN return(0)
           ELSE return(1);
```

Figure 4.2: RELIEF algorithm.

Recently some decision tree algorithms, that take into account a large number of features describing objects, have been proposed [28]. The algorithm [28] can be modified to analyze spatial descriptions of the data. For every classified object, a set of generalized predicates that are satisfied by this object is stored. An example of such a description is presented in Table 4.3. For every predicate P from the table we have to find p_1, i.e., the number of objects that belong to class Y, and satisfy the predicate P and n_1, i.e., the number of objects that belong to class N, and satisfy the predicate P. The same calculations have to be done for the negation of the predicate P by finding p_2, i.e., the number of objects that belong to class Y and do not satisfy the predicate P and n_2, i.e., the number of objects that belong to class N and do not satisfy the predicate P. For example, for the predicate $close\_to(x, park)$ one may find $p_1 = 1$, $n_1 = 2$, $p_2 = 1$, and $n_2 = 1$. These values are used to compute information gain for each of the predicates based on Equations 4.1, 4.2, and 4.3. In a similar manner for other attributes, instead of branching on all attribute values, the algorithm branches on one attribute value. For example, we may have separate branches if $sum(population) = MEDIUM$ and $sum(population) <> MEDIUM$. As claimed in [42], such binary decision trees are usually more compact and more accurate than the trees produced by regular ID3 algorithm, which branch on all values of the predicates. The pseudo-code of the decision tree algorithm is presented in Figure 4.3.

4.2.2 Buffer Size and Shape

It is important to determine the proper size and shape for the buffers. Buffers that are used for finding aggregate information from thematic maps and buffers used for determining spatial predicates should be considered. Buffers represent areas that have an impact on the class label attribute of classified objects. For example, in the case of stores and shopping malls, a buffer may represent the area where the customers live or work. In business geographics the spatial extent of a store's customers is called the *trade area* [116]. The analysis of trade areas helps to answer such questions as:

- Which customer segments are driving my business?
- How far are my customers willing to travel to shop?
- Is my site strategically placed to defend or dominate my market?

We can analyze different sizes of buffers and choose the best size for the discrimination between classes. It can be done by using different sizes of the buffer to compute the information gain for all predicting attributes, which are aggregates, and by choosing the size of

```
Build_tree(Decision_tree, set_of_object's_decriptions)
    Find_Predicate_Counts(set_of_object's_decriptions, Predicate_true, Predicate_false);
    Best_Predicate := Find_Predicate_with_Best_Info_Gain(Decision_tree, Predicate_true, Predicate_false);
    set_of_object's_decriptions_where_Best_Predicate_is_TRUE
                  := select_sets(set_of_object's_decriptions, Best_Predicate, TRUE);
    set_of_object's_decriptions_where_Best_Predicate_is_FALSE
                  := select_sets(set_of_object's_decriptions, Best_Predicate, FALSE);
    If not_empty(set_of_object's_decriptions_where_Best_Predicate_is_TRUE)
        Build_tree(Decision_tree.TRUE_node, set_of_object's_decriptions_where_Best_Predicate_is_TRUE);
    If not_empty(set_of_object's_decriptions_where_Best_Predicate_is_FALSE)
        Build_tree(Decision_tree.FALSE_node, set_of_object's_decriptions_where_Best_Predicate_is_FALSE);


Find_Predicate_Counts(set_of_object's_decriptions, Predicate_true, Predicate_false)
    /* Predicate_true[max_relevant_predicate, max_class] and Predicate_false are tables
         that contain the number of predicates that are true(false) for each class. For example,
         if (Predicate_true[4, 2] == 3) then the fourth predicate is satisfied by 3 objects from class 2 */
    FOR i := 1 TO max_class DO
         { num_objects[i] := 0;
           FOR j := 1 TO max_relevant_predicate DO
                  Predicate_true[j, i] := 0; }
    FOR i := 1 TO classified_object_number
         { num_objects[class(object[i])]++;
           FOR EACH predicate_j IN object_descriptions[i]
                  Predicate_true[j, class(object[i])]++; }
    FOR i := 1 TO max_relevant_predicate DO
         FOR j := 1 TO max_class DO
               Predicate_false[i, j] := num_objects[j] - Predicate_true[i, j];
```

Figure 4.3: Decision tree building.

the buffer with the largest value of information gain. Then, this size is applied to compute all aggregates for all attributes. The reason to do so is that the decision tree will be first divided on the attribute with the largest information gain value producing the best split of the dataset. Such a buffer presents the best trade area for the classification task and trade areas should not change between attributes, as they represent the areas with the most influential population of customers.

In addition to the determination of the size of the buffers, we may use different criteria to determine the shape of the buffers. The buffers may be based on rings, customer penetration polygons, drive-time polygons, Voronoi diagrams, etc. An overview of this issue is presented in [116]. The rings have some advantages such as: (1) the ease of use, (2) having no need to determine trade area based on customer data, and (3) an easy comparison between sites. One study showed a high degree of similarity between rings and customer penetration polygons [116]. In our experiments we used rings (or equidistance buffers for objects that are not points) because no other information was available to determine better neighborhoods for the classification task.

4.2.3 Aggregate Information

Aggregate values for areas close to spatial objects play an important role in the analysis of many business objects like stores, restaurants, gas stations, etc. [116]. To handle aggregate values of non-spatial attributes in thematic maps one can calculate aggregate values for the block groups intersecting buffers. For example, one can notice that only a small part of block C on the map, presented in Figure 4.1, is intersected by a buffer for OID1. Thus, one can take into account a part of the particular block group that is intersected by the buffer surrounding classified object and use population living, only in this part, with the assumption of even distribution. Such calculations, however, would be quite computationally expensive and, in reality, census blocks are much smaller in the comparison to the buffers than the block shown in Figure 4.1. Therefore, more blocks are intersected by a buffer and average influence of a single block on aggregate's value is small, making detailed computations less important. The aggregate values for our example are shown in Table 4.4. This approach transforms descriptions of the classified objects into a flat table, which can be easily mined using any classifying algorithm.

The aggregate data can be also generalized and merged with the predicate data, so finally each object can be classified using a set of predicates describing properties of both thematic

ObjectID	high_profit	sum(population)	avg_income
1	Y	15000	$27,127
2	Y	20000	$28,512
3	N	14500	$29,193
4	N	12000	$28,500
5	N	19000	$30,500

Table 4.4: Descriptions of classified objects

ObjectID	high_profit	Predicates
1	Y	sum_population(x, MEDIUM), avg_income(x, SMALL), close_to(x,park), close_to(x,water)
2	Y	sum_population(x, LARGE), avg_income(x, MEDIUM), close_to(x,water)
3	N	sum_population(x, MEDIUM), avg_income(x, LARGE), close_to(x,park), close_to(x,water)
4	N	sum_population(x, SMALL), avg_income(x, MEDIUM)
5	N	sum_population(x, LARGE), avg_income(x, LARGE), close_to(x,park)

Table 4.5: Generalized descriptions of classified objects presented as sets of predicates

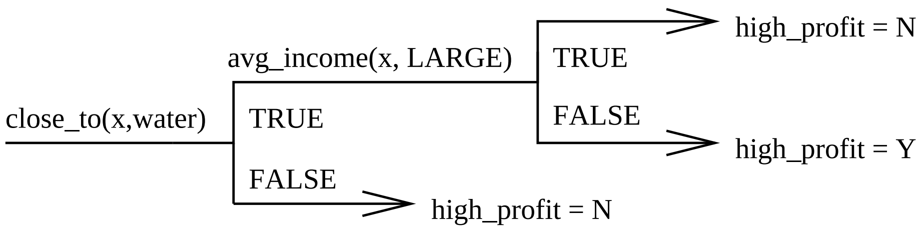

Figure 4.4: Decision tree for data from Table 4.5.

map and other object intersecting trade areas for each object. An example is presented in Table 4.5. The resulting decision tree is presented in Figure 4.4.

4.3 The Algorithm

The algorithm for building decision trees, for the classification of spatial objects using spatial predicates, spatial functions and thematic maps, based on principles discussed in the previous section, may be summarized as follows.

Algorithm 4.1 Mining the spatial classification rules in a large spatial database.

Input: The input consists of a database, a mining query, and a set of thresholds as follows:

1. A database containing:
   (a) the set of classified objects O_c,
   (b) other sets of possibly relevant spatial objects,
   (c) a set of concept hierarchies,
2. Geo-mining query specifying:
   (a) objects to be used in the classification process,
   (b) predictive attributes, functions, and predicates,
   (c) class label attribute, function, or predicate.
3. A confidence level for relevant attributes.

Output: A binary decision tree.

Method: Mining spatial classification proceeds as follows:

Step 1: *Task_relevant_DB* := extract_task_relevant_objects(*SDB, RDB*);
Step 2: If (*Aggregates_used*) Then
    Best_buffer_size := best_buffer_computations(sampling(*Task_relevant_DB*));
Step 3: *Relevant_coarse_predicate_DB* :=
    relevance_coarse_computation(sampling(*Task_relevant_DB*), *Best_buffer_size*);
Step 4: *Fine_predicate_DB* :=
    refined_spatial_computation(*Relevant_coarse_predicate_DB*);
Step 5: Generate_decision_tree(*Fine_predicate_DB*);

Explanation of the detailed steps of the algorithm.

Step 1 collects set of data specified in the query, which consists of set classified objects O_c and objects that are used for the description.

Step 2 finds the best size of the buffer for aggregates of thematic map polygons by finding, for all relevant non-spatial aggregate attributes, such size of the buffer *Best_buffer_size* where the information gain for the aggregated attribute is maximum. The computations are performed on a sample of described objects from *Task_relevant_DB*. Such a buffer would be used to compute aggregates for all relevant non-spatial attributes of thematic maps. Distance-based join index [98] may be used to accelerate the computations.

Step 3 performs relevance analysis. It takes a sample of spatial objects O_c and: (1) builds sets of predicates describing all objects, using relevant coarse predicates, functions, and attributes; (2) performs generalizations of the sets of predicates based on concept hierarchies; and (3) finds coarse predicates functions, and attributes which are relevant to the classification task using RELIEF algorithm. The relevant predicates, functions, and attributes are stored in *Relevant_coarse_predicate_DB* database.

Step 4 builds set of predicates describing all objects using relevant fine predicates, functions, and attributes.

Step 5 performs generalizations of the set of predicates, based on concept hierarchies, and generates a binary decision tree, based on the algorithm presented in Figure 4.3. □

Rationale of the Algorithm.

Step 1 is done by spatial query processing. Step 2 finds the best value for the buffer for the most relevant aggregate attribute and uses this size for all other attributes. Step 3 is an elimination of irrelevant predicates based on a statistical method. Using distribution of data and a user defined level of confidence, one can compute a threshold used to eliminate irrelevant attributes [84]. Step 4 uses spatial access methods and algorithms to create a set of relevant predicates, functions and attributes. Step 5 generates a binary decision tree using the well-known ID3 algorithm [121] modified to allow processing of descriptions in the form of sets of generalized predicates, as described in the previous section.

The goal of classification is to find a model that allows for the high quality of the estimation of the class a classified object belongs to. Under certain assumptions, the expected error rate increases when the complexity of the decision tree used for classification also increases in terms of the number of leaf nodes [120]. Unfortunately, the process of building a decision tree with the smallest number of tests is NP-complete [74]. Thus, different heuristics, like the one incorporated by ID3 algorithm are used. In our algorithm, in addition to the ID3 heuristic, we use relevance analysis, which minimizes not only the computation time for finding fine predicates, but also tends to minimize the size of the resulting decision tree. The RELIEF algorithm eliminates most of the non-relevant predicates leaving with a certain confidence level all relevant predicates. The use of a binary decision tree also leads to a smaller size of the tree, which is preferable. The size of the sample used for relevance analysis may be estimated using statistical techniques proposed in RELIEF algorithm [84].

Optimizations used in the Algorithm.

The algorithm presented above introduces a number of optimization techniques in the comparison with algorithm suggested in [37], namely:

1. It uses two-step approach (sampling and coarse predicates followed by fine computations for promising patterns) to accelerate computations.
2. It takes into account aggregate information.
3. It may use distance-based join index [98] to accelerate query processing. Neighborhood indices proposed in [37] have a large size if they present different neighborhood relations. Therefore, they cannot be built for all possible relations.
4. It uses concept hierarchies which result in simpler decision trees and faster computations [78].

Complexity analysis.

Table 4.6 lists some parameters used in the cost analysis. Values of the parameters are based on experiments done on a Pentium 166MHz computer with 64MB of main memory. Time to classify objects without filtering relevant attributes is presented by Equation 4.4. Component $k \times d \times t_f$ presents time to get fine predicates and component $k \times d_{total} \times t_{build}$ presents time to build decision tree from a table of predicate sets. Execution time of the algorithm 4.1 (without analyzing aggregates) can be estimated using Equation 4.5. Component $min(200, k) \times d \times t_c$ presents time to get coarse predicates for the sample of objects that are classified and component $(min(200,k))^2 \times d \times t_{filter}$ presents time to filter relevant predicates. Size of the sample is set to $min(200,k)$. Component $F_{ratio} \times k \times d \times t_f$ presents time to get relevant fine predicates and component $F_{ratio} \times k \times d_{total} \times t_{build}$ presents time to build a decision tree from the table of relevant predicate sets. Estimated times are presented in Figure 4.5.

$$t_1 = k \times d \times t_f + k \times d_{total} \times t_{build} \quad (4.4)$$

$$t_2 = min(200,k) \times d \times t_c + (min(200,k))^2 \times d \times t_{filter} + F_{ratio} \times k \times d \times t_f + F_{ratio} \times k \times d_{total} \times t_{build} \quad (4.5)$$

Name	Value	Meaning
k		number of classified objects
d	8	average number of predicates describing an object
d_{total}	60	number of different predicates describing objects
F_{ratio}	0.5	ratio of relevant predicates
t_c	1 ms	constant for finding coarse predicates
t_f	50 ms	constant for computing one fine predicate
t_{build}	0.05 ms	constant for building decision tree
t_{filter}	2 μs	constant for filtering out non-relevant attributes

Table 4.6: Database parameters.

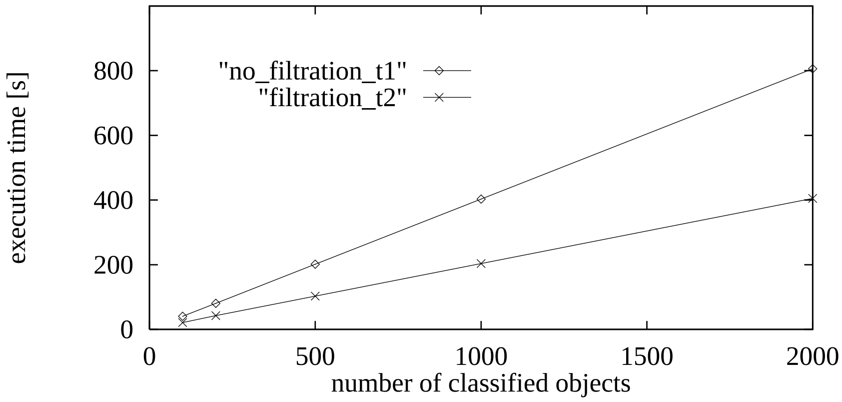

Figure 4.5: Execution times of the algorithm

4.4 Algorithm Extensions

In this section we discuss some of the possible modifications and extensions of the progressive refinement approach for spatial classification. To simplify the discussion, the problem of finding the best size of the buffer is not addressed here.

Algorithm 4.2 Mining the spatial classification rules in a large spatial database using only coarse predicates.

Input: Same as Algorithm 4.1.

Output: Same as Algorithm 4.1.

Method: Mining spatial classification proceeds as follows.

Step 1: *Task_relevant_DB* := extract_task_relevant_objects(*SDB, RDB*);
Step 2: *Relevant_coarse_predicate_DB* :=
  relevance_coarse_computation(sampling(*Task_relevant_DB*));
Step 3: *Coarse_predicate_DB* :=
  coarse_spatial_computation(*Relevant_coarse_predicate_DB*;
Step 4: Generate_decision_tree(*Coarse_predicate_DB*);

Discussion of the algorithm.

The algorithm is similar to the Algorithm 4.1, except *coarse* predicates are used for both the relevance analysis and for the building of a decision tree. As the results presented in the Section 4.5 imply, such an approach leads to a shorter computation time, but the quality of classification measured by the percentage of correctly classified objects deteriorates comparing to Algorithm 4.1. □

Algorithm 4.3 Mining the spatial classification rules in a large spatial database using only fine predicates.

Input: Same as Algorithm 4.1.

Output: Same as Algorithm 4.1.

Method: Mining spatial classification proceeds as follows.

Step 1: *Task_relevant_DB* := extract_task_relevant_objects(*SDB, RDB*);
Step 2: *Relevant_fine_predicate_DB* :=
relevance_fine_computation(sampling(*Task_relevant_DB*));
Step 3: *Fine_predicate_DB* :=
fine_spatial_computation(*Relevant_fine_predicate_DB*);
Step 4: Generate_decision_tree(*Fine_predicate_DB*);

Discussion of the algorithm.

The algorithm is similar to Algorithm 4.2, except *fine* predicates instead of *coarse* predicates are used for both the relevance analysis and for the building of a decision tree. The results presented in Section 4.5 show that such an approach leads to a similar quality, while the computation time might be even shorter in some cases than the execution time of the algorithm 4.1. The shorter execution is attributed to the smaller number of relevant attributes detected when fine predicates are used for relevance analysis. This observation leads to Algorithm 4.4 □

Algorithm 4.4 Mining the spatial classification rules in a large spatial database using both coarse and fine predicates for the multi-step relevance analysis.

Input: Same as Algorithm 4.1.

Output: Same as Algorithm 4.1.

Method: Mining spatial classification proceeds as follows.

Step 1: *Task_relevant_DB* := extract_task_relevant_objects(*SDB, RDB*);
Step 2: *Relevant_coarse_predicate_DB* :=
relevance_coarse_computation(sampling(*Task_relevant_DB*));
Step 3: *Relevant_fine_predicate_DB* :=
relevance_fine_computation(sampling(*Task_relevant_DB*),
Relevant_coarse_predicate_DB);
Step 4: *Fine_predicate_DB* :=
fine_spatial_computation(*Relevant_fine_predicate_DB*);
Step 5: Generate_decision_tree(*Fine_predicate_DB*);

Discussion of the algorithm.

In this algorithm the relevance analysis is performed first using coarse predicates based on a sample of classified objects. Then, the fine versions of coarse predicates that were judged as possibly relevant in Step 2 are computed based on the same sample. This approach may lead to a shorter execution time with good accuracy. The possible shorter execution time is due to the further reduction of the set of relevant predicates. The experimental evaluation of this algorithm should be done in the future. □

4.5 Experimental Algorithms Evaluation

In our performance study, we evaluated the quality and efficiency of our algorithm for classification of data, based on spatial predicates *g_close_to*, which describe relations between classified objects and other objects on the map located within a distance threshold. Relations of the data to thematic maps were not taken into account. In order to evaluate classification quality and execution time of the algorithm we created a synthetic dataset, which was merged with TIGER data [141] for Washington state. The synthetic data contained 1800 polygons with 5 to 55 edges. The polygons belonged to 4 sets: two class label sets (T00 and T01) and two predictive sets (S00 and S01). If object x satisfies predicate *g_close_to(x, S00, 1km)* $\oplus$ *g_close_to(x, S01, 1km)* then object x belongs to the class T00; otherwise it belongs to the class T01. TIGER data contained over 87,000 polygons, lines and points presenting roads, parks, rivers, lakes and other objects in Washington state. MapInfo Professional 4.1 Geographic Information System [102] was used as spatial database for the experiments. We tested the quality of classification and the time of the execution of the algorithm. Relevance analysis was done based on 100 objects from each class label set. The coarse predicate used for the relevance analysis was *coarse_g_close_to(x, y, 1km)*. This predicate implies that MBRs of two objects are located within 1 kilometer distance. After relevance analysis was performed fine relevant predicates were found for all objects. 400 objects from each class label set were used for a training set and the remaining 50 objects were used for testing. The results are summarized in Table 4.7.

As one can observe, the quality of the classification process increases drastically when relevance analysis is performed. If we chose to classify objects based on the algorithm proposed in [37] we would get a poor quality because this algorithm performs no relevance

threshold	number of relevant predicates	time [s]	quality
$-\infty$	58	318	82%
0.0	13	105	99%
0.1	4	85	100%
0.2	3	82	100%
0.3	2	76	100%

Table 4.7: Results of the algorithm for artificial data

analysis. After applying relevance analysis, execution time decreased 3 to 4 times, depending on the value of the threshold. The algorithm [37] is based on neighborhood indices that avoid process of spatial join which is necessary to extract spatial predicates. When no neighborhood indices exist, the algorithm [37] has to compute predicates and that computation results in the classification time similar to the time in our experiment for a threshold equal to $-\infty$ when all predicates are used for decision tree construction. The process of relevance analysis took about 17 seconds, where 16 seconds were spent on spatial operations. The process of building a decision tree from the 800 sets of predicates took about 1 second for every experiment.

We also performed a number of experiments with real data from the TIGER dataset. In most cases, classification accuracy increased drastically when relevance analysis was used. The best results were for threshold values between 0.0 and 0.2. The time to perform relevance analysis and build a decision tree was between 10% and 50% shorter than the time necessary to build a decision tree based on all predicates.

We compared the quality of classification based on fine predicates with the quality of classification based on coarse predicates. In all cases, except one, when coarse classification yielded 1% better quality, the classification with fine predicates produced a better quality of up to 8%.

Finally, we compared the algorithm where coarse predicates are used for relevance analysis with the algorithm where fine predicates are used for that purpose. Experiments showed that the quality of classification was fairly similar for both algorithms, while relevance analysis using fine predicates usually yielded a slightly smaller set of relevant predicates. In addition to fine predicates which were judged as relevant ones, some *coarse_g_close_to* predicates representing large objects were chosen as relevant ones. This happens if objects from the same class are clustered together as in the case of some of our training data. In this case all objects from the same cluster may satisfy coarse predicate for a large object even though the object related to this predicate is relatively distant, and thus, the fine predicates are not satisfied. Because of a smaller set of relevant predicates the total time of classification for both relevance analysis and tree construction time was similar for the classification of point objects (a relatively short time for computation of fine predicates) and it was longer for more complex objects.

4.6 Summary and Future Extensions

Classification of geographic objects discloses relations between spatial and non-spatial data in large spatial databases. We have proposed and evaluated an efficient method for classification of spatial objects. The proposed method enables classification of spatial objects based on aggregate values of non-spatial attributes for neighboring regions. It also takes into account spatial relations between objects on the map, which may be represented in the form of predicates. The algorithm first performs less costly, approximate spatial computations to obtain a sample of approximate spatial predicates. Relevance analysis is performed only for such a sample. Then, refined computations are done only for the set of promising patterns producing smaller and more accurate decision trees. Performing a second relevance analysis based on refined versions of possibly relevant coarse predicates can expand this method further. This multi-step approach enables a substantial reduction of candidate sets which results in faster computations.

Conducted experiments showed the importance of relevance analysis, which resulted in a better quality of decision trees in comparison to the trees produced by the algorithm proposed in [37]. We also observed a significant acceleration of time for building decision trees when only relevant attributes are used to perform classification.

An interesting research topic is to investigate the integration of the classification algorithm with a spatial query engine. For example, when sets of predicates are computed for a mall there may be no need to compute intersections of the buffer for this mall with all segments of a highway. If a predicate *g_close_to(mall, highway, 1km)* is true for one segment of the highway then there is no need to compute intersections with other sections of that highway.

Experiments using aggregate values for thematic maps and varied distances for *close_to* spatial predicates, and the experiments analyzing the impact of different levels of concepts used for the classification on classification accuracy should be performed in future.

Decision tree methods may be inappropriate when sensible tests are likely to depend on combinations of attributes that cannot be specified in advance. Such situations happen when a number of relevant factors must be weighted against each other. In such situations regression methods may be a better choice and the investigation of such methods in large spatial databases is necessary.

The LORD said, "If as one people speaking the same language they have begun to do this, then nothing they plan to do will be impossible for them ..."
Genesis 11:6

Chapter 5

Design of a Spatial Data Mining Query Language

One of the important parts of the data mining process is formulation of the user query. We believe that the success of relational databases should be credited in part to the creation of the standardized relational query language - SQL. Although many different graphical user interfaces exist for various database systems, the underling interface is based on SQL. Such a "core" language sets a foundation for information exchange, promotes commercialization and a wide acceptance of the technology. For example, ODBC standard based on SQL enables a wide exchange of data using standard ODBC drivers. We believe that the foundations for a unified spatial data mining query language should be analyzed. With such query language one could perform many different data mining tasks with a single query language interface instead of using a lot of different APIs. In the design of the language for spatial data mining we considered the following assumptions and guidelines:

- The data used in the mining process is stored in relations with spatial and non-spatial attributes.
- Spatial and non-spatial concept hierarchies exist, and relevant datasets may be specified using high level concepts.
- The relationships between data items should be specified in a way similar to Spatial SQL [34] queries.

- The language should enable a dialog using pointing devices for the selection of objects used in the discovery process.
- The language should enable the mining of different types of knowledge ranging, from generalized relations, through clustering to spatial classification.
- The thresholds for the selection of interesting patterns should be specified in a mining query.
- The visualization of the discovered knowledge is done independently.

Based on the above considerations we constructed GMQL (Geo-Mining Query Language), a language for the formulation of input request for the mining process. The language uses a unified syntax, which allows for a variety of data mining queries. GMQL is based on DMQL (Data Mining Query Language), which was developed for mining knowledge from large relational databases [66].

By using GMQL a user can specify four primitives for data mining tasks:

1. the type of knowledge to be discovered,
2. the set of relevant data for the mining process,
3. the thresholds to filter out uninteresting rules, and
4. the concept hierarchies as the background knowledge.

The mining query can be presented in an extended BNF grammar, where "[ ]" represents 0 or one occurrence, "{ }" represents 0 or more occurrences, and words in **BOLD** font represent keywords, in the following way:

MINE <rule_specification>
[**ANALYZE** <aggregate_list>]
WITH RESPECT TO <attribute_list>
FROM <relation(s)>
[**WHERE** <conditions>]
[**GROUP BY** <attribute_list>]
[**HAVING** <conditions>]
{**SET** <threshold_specification>}
{<hierarchy_specification>}

The "MINE <rule_specification>" clause specifies the type of rules to be mined and can also specify additional descriptions like the name of the rule or the name of the described class. For example, by specifying "MINE SPATIAL ASSOCIATIONS DESCRIBING "Golf Courses"" a user can choose that the output of the query should present a description of golf courses in form of association rules.

The next clauses define data relevant to the mining query. They specify which attributes and tables should be selected from the database. The optional "ANALYZE <aggregate_list>" clause specifies attributes, predicates or functions to populate cells of the data cube. For example, if a user uses "ANALYZE sum(pop)" then every cell in the cube will contain the sum of values for *pop* attribute. If no aggregate function is specified, then only count(*) aggregation is used for quantitative representation of the discovered knowledge. The "WITH RESPECT TO <attribute_list>" clause selects a list of attributes, predicates or functions relevant to the mining task. This list may also include spatial descriptions of the objects which are represented as *geo* (geometry) attribute. The selected attributes will constitute the dimensions of the data cube which is used for multidimensional analysis. The "FROM <relation(s)> [WHERE <conditions>] [GROUP BY <attribute_list>] [HAVING <conditions>]" clause is used to build an SQL query to retrieve the relevant set of data for the analysis. The relevant thresholds are determined using the optional "SET <threshold_specification>" clause. Finally, the optional {<hierarchy_specification>} clause enables specification of concept hierarchies used in the mining process.

In general, the GMQL query should facilitate the optimization of data mining queries and should satisfy *closure principle*, i.e., the results of a data mining query can be used as data for the next data mining query [75]. The SQL results satisfy the closure principle and this principle can be also satisfied by GMQL, as most of the results of the data mining queries can be presented in the form of tables.

The rest of the chapter is organized in the following way. In Section 5.1 we present other relevant languages. In Section 5.2 we describe the method for the specification of relevant data. In Section 5.3 we analyze syntax for particular spatial data mining tasks. In Section 5.4 we discuss the relevant thresholds. Section 5.5 contains discussion of the syntax for the construction of concept hierarchies. In Section 5.6 we present possible extensions for future spatial data mining tasks. The chapter ends with the summary.

5.1 Other Relevant Query Languages

5.1.1 Structured Query Language SQL

The Structured Query Language (SQL) originally called SEQUEL [21] has become a standard query language for relational databases [4]. SQL deals with the relational data model and is based on relational algebra. It has two major components:

- Data Definition Language (DDL) for defining the database structure, and
- Data Manipulation Language (DML) for retrieving and updating data.

From the a knowledge discovery point of view, only DML is of interest, because we are dealing with existing data.

The main construction of DML is the SELECT-FROM-WHERE structure. The SELECT clause describes the attributes, functions and aggregates to be displayed, the FROM clause determines the tables containing data needed to extract results and the WHERE clause specifies conditions to be satisfied by the resulting set of data.

The additional clauses GROUP BY, HAVING and ORDER BY may be used in the query. The GROUP BY clause forms groups of rows with the same column value, e.g., the rows that may be used for the computation of aggregate values. The HAVING clause filters the groups subject to some conditions, and the ORDER BY clause specifies the sorting order of the output.

A powerful feature of SQL is its ability to deal with the subqueries. A subquery is a complete SELECT statement that is used in the WHERE clause of the outer SELECT statement. For example, the following query selects names and addresses of the staff with the salaries greater than the average salary.

```
SELECT name, address
FROM staff
WHERE salary > (SELECT avg(salary)
                FROM staff)
```

Recently, a number of extensions to SQL were proposed to enable adding powerful OLAP analysis features. One of these extensions is the CUBE operator for the constructions of data cubes storing multiple aggregates [55].

5.1.2 Spatial SQL

In [32], M. Egenhofer proposed Spatial SQL, i.e., a language for spatial data handling. The language is based on SQL and incorporates the distinct characteristics of spatial data. The additional spatial data types are introduced and a number of spatial operations and relationships are analyzed.

The *unary spatial operations* enable access to the properties of spatial objects. For example, parts of spatial objects like *boundary, boundaryNodes*, or *interior* may be extracted using these operations. Unary spatial operations can be also used to find area or length of spatial objects. Other features, like a perimeter of a polygon, can be found by the combination of the operations.

The *binary spatial operations* can calculate a numerical value describing a spatial relationship between two spatial objects. Spatial SQL supports two binary spatial operations: *distance* and *direction*.

The *binary spatial relationships* are the relationships between two spatial objects. The relationships include (1) topological relationships like *disjoint, meet, overlap, inside/contains, covers/coveredBy, inside/contains*, or *equal*, and (2) spatial order relations like *left/right, north/south*, or *over/under*. These spatial operations are incorporated into GMQL.

Interactive communication, using pointing device such as mouse, is enabled by the PICK qualifier. For example, the following query:

SELECT name
FROM city
WHERE geometry = **PICK**

enables selection of the name of the city on which a user clicked a mouse.

Spatial SQL also includes the GPL (Graphical Presentation Language) which provides descriptions for graphical manipulation of query results. The Geo-Mining Query Language does not include GPL, because most of the spatial query languages concentrate on the retrieval of data, leaving the visualization details to the GUI. GMQL does not contain the DDL (Data Definition Language) part of Spatial SQL either as the data for the mining process already exists.

5.1.3 Multidimensional Expressions (MDX)

The Microsoft Corporation proposed a standard for query language for the manipulation of multidimensional data [105]. The *Multidimensional Expressions* (MDX) language enables querying of multidimensional data stored in data cubes. A typical MDX query takes the following form:

SELECT <dimension specification> {,<dimension specification>}
FROM <cube specification>
WHERE <slicer specification>

For example, an MDX query:

SELECT {Canada_West, Canada_East, Europe.CHILDREN} **ON COLUMNS**
{Qtr1.CHILDREN, Qtr2, Qtr3, Qtr4.CHILDREN} **ON ROWS**
FROM Sales
WHERE (Sales, [Office Products], 1991)

may result in the following cross table:

		Canada_West	Canada_East	Europe	
				France	Germany
Qtr1	Jan	10	12	11	11
	Feb	12	14	10	8
	Mar	11	13	12	11
Qtr2		30	40	35	25
Qtr3		25	35	40	28
Qtr4	Oct	10	13	10	11
	Nov	12	14	11	10
	Dec	14	15	13	14

The cube used to construct the cross table has the following dimensions: Geography, Time, Measures, and Products. The first two dimensions are chosen to be the columns and rows of the resulting cross table. One can notice that the SELECT clause contains a hierarchical expressions Europe.CHILDREN, Qtr1.CHILDREN and Qtr4.CHILDREN. Based on the concept hierarchies the children of the Europe, Qtr1 and Qtr4 nodes are found and displayed. The slicer clause WHERE contains conditions for Measures, Products and Time dimensions. The numbers presented in the table are sales (from the Measures dimension) of office products (from the Products dimension) for 1991 (from the Time dimension). As one can see, MDX enables easy selection of the nodes to be displayed using names from concept hierarchies. However, MDX seems to be tailored specifically for the display of cross tables and thus, it has limitations if applied to other data mining tasks. Also, the specifications of slicer conditions are not similar to the conditions used in SQL, which follow the construction "attribute_name comparator value" (e.g., Time = 1991).

GMQL makes use of multidimensional expression using names from concept hierarchies, but it is done in a way similar to SQL conditions. GMQL enables work with relations instead of data cubes and it allows the selection of the dimensions and measures used in the mining process.

5.1.4 Data Mining Query Language DMQL

DMQL, a data mining query language for relational databases, was introduced in [66] . The language was partially implemented and used in the relational data mining prototype DBMiner [61, 63, 67]. The design of the DMQL is based on five principles that are also followed by GMQL:

1. The set of data relevant to a data mining task should be specified in a data mining query.
2. The kind of knowledge to be discovered should be specified in a data mining query.
3. Background knowledge in the form of concept hierarchies could be generally available for the data mining process.
4. Data mining results should be able to be expressed in terms of generalized or multiple-level concepts.
5. Various kinds of thresholds should be flexibly specified to filter out less interesting knowledge.

A DMQL query can be presented in BNF as:

< DMQL> ::= **USE DATABASE** <database_name>
{<hierarchy_adoption>, ...}
<rule_spec>
RELATED TO <attr_or_agg_list>
FROM <relation(s)>
[**WHERE** <condition>]
[**ORDER BY** <order_list>]
{**WITH** [<kinds_of>] **THRESHOLD** = <threshold_value>
[**FOR** <attribute(s)>]}

In <DMQL>, "USE DATABASE <database_name>" clause directs the mining task to a specific database "<database_name>", and the optional clauses, "<hierarchy_adoption>", assigns <hierarchy> to a particular attribute <attribute> (otherwise, a default hierarchy is used). The clause, <rule_spec>, is the specification of the kind of rules to be discovered. The next clause, "RELATED TO <att_or_agg_list>", selects a list of relevant attributes and/or aggregations for generalization. The "FROM <relation(s)> [WHERE <condition>]" clause forms an SQL query to collect the set of relevant data. The "ORDER BY" clause specifies the order of rows to be printed. The "WITH-THRESHOLD" clause specifies various kinds of thresholds.

In the comparison with DMQL, the Geo-Mining Query Language enables the mining of additional types of knowledge like descriptions of clusters or spatial association rules, the use of spatial predicates like *g_close_to*, the use of spatial attributes, and the definition of spatial relationships like *intersects, contains,* or *within.*

GMQL also distinguishes clearly between measures and dimensions. While in the version of DMQL presented in [66] both measures and dimensions were put into the RELATED TO clause, GMQL uses separate clauses for dimensions and measures. This distinction allows (1) to create dimensions based on aggregate functions, and (2) different treatment of spatial attributes when used as measures and when used as dimensions. The SQL-like clauses GROUP BY and HAVING were also incorporated. The hierarchy definition part of DMQL is directly incorporated to GMQL with additional accommodations for the definition of spatial concept hierarchies.

5.2 Specification of Relevant Data

Spatial objects are stored in relations that contain spatial and non-spatial attributes. In the process of selecting data relevant to the mining task, one has to specify (1) the relevant tables, (2) the conditions that are satisfied by the relevant objects and (3) the properties of the objects which the mining process is based on.

In Chapter 4 we noticed that spatial data can be described using many different properties. In addition to the non-spatial attributes of objects, the relevant description may include spatial predicates (analyzed in Chapter 3), spatially related attributes with non-spatial values, and spatial functions. The spatial predicates include topological relations (e.g., *meet, overlap*, etc.), distance relations (e.g., *g_close_to*), and direction relations (e.g., *north, west, under*, etc.). When spatial predicates are to be used in the mining process the types of the predicates should be specified in the WITH RESPECT TO clause. It can be done using the following templates:

- G_CLOSE_TO(spatial_term, spatial_term, “"'distance_specification“"'),
- DIRECTION(spatial_term, spatial_term), and
- TOPOLOGY(spatial_term, spatial_term),

where spatial term is a spatial attribute or the result of a spatial operation execute on a spatial attribute. The G_CLOSE_TO template summarizes all topological predicates like *meet, overlap,* etc., as well as distance predicate *close_to* into one generalized predicate *g_close_to* which formal description is presented in Definition 3.1. The TOPOLOGY template allows a user distinguishing between particular predicates like *contains, equal, close_to, etc.*

Example 5.1 The query to find spatial associations using *g_close_to* predicates in conjunction with topological predicates like *intersects*, *meets*, etc., can be presented as:

MINE SPATIAL ASSOCIATIONS DESCRIBING DESCRIBING “Stores”
WITH RESPECT TO Store.geo, G_CLOSE_TO(Store.geo, Features.geo, “3 km”),
TOPOLOGY(Store.geo, Features.geo), Features.code
FROM Store, Census

The templates G_CLOSE_TO(Store.geo, Features.geo, “3 km”) and TOPOLOGY(Store.geo, Features.geo) are used to specify predicates used in the mining process. The distance threshold for the *g_close_to* predicate is 3 kilometers. □

Spatial functions like *driving_distance(geo_attribute, geo_attribute)* fit conceptually into the area of spatial predicates and can be incorporated into GMQL using similar templates.

In Chapter 4 aggregate values were analyzed as the way of describing and classifying spatial objects. The aggregate values are based on the non-spatial attributes of spatial objects, that are located within buffers around objects that are described or classified. The process of finding the shape and the size of the buffers was presented in Section 4.2.2. The aggregate values can be computed using (1) the maps storing the geometrical representations of buffers, or (2) distance functions and GROUP BY clauses, as shown in the following example.

Example 5.2 The query to characterize stores by presenting the relation of the sum of sales in a store in relation to the profile of the store can be presented as:

```
MINE CHARACTERISTICS DESCRIBING "Stores"
ANALYZE sum(sales), count(*)
WITH RESPECT TO Store.profile, WGAVG(Census.income, Census.pop) AS income
FROM Store, Census
WHERE DISTANCE(Store.geo, Census.geo) < "3 km"
GROUP BY Store.geo
```

Note, that the weighted average is used in the query, instead of the regular average. It is computed using census blocks located within 3 km from the stores. The *Store.profile* attribute does not appear in the GROUP BY clause, but it appears in the WITH RESPECT TO clause. Such a situation would not be permitted in the SQL SELECT-GROUP BY construct, where all attribute names used in SELECT clause as non-aggregates must appear in a GROUP BY clause. In our case we have a spatial GROUP BY and such a situation is permitted. □

5.2.1 WITH RESPECT TO and ANALYZE Clauses

If a data cube is used for data handling, then the WITH RESPECT TO clause specifies the dimensions of the data cube and the ANALYZE clause specifies measures stored in the data cube. Figure 5.1 presents an example of a three-dimensional data cube. The dimensions are *statename, time* and *capita_income*. The measures of the objects are stored in the cells of the data cube. The measures may include non-spatial measures like *count(*)*, or *sum(pop)*

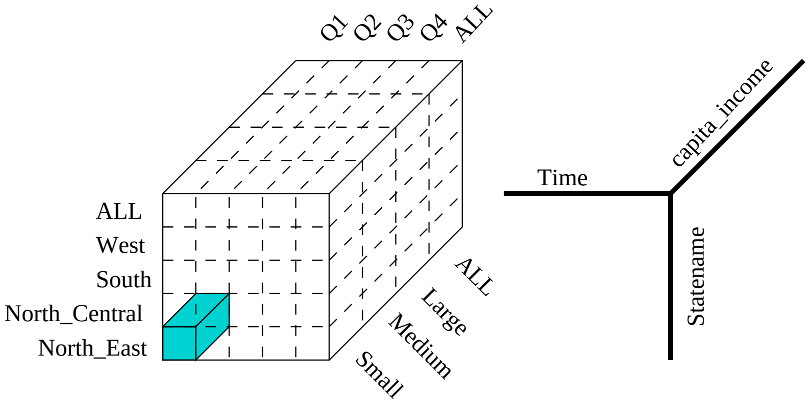

Figure 5.1: An example of a data cube

or spatial measures like the result of the spatial regions merge. For example, the shaded cell in Figure 5.1 presents data for states in North_East region of US with a small income per capita and the first quarter of the year. The measures stored in this cell correspond to these states. The additional value ALL contains aggregates for all values in the particular dimension and enables fast computation of GROUP BY queries, which are used in data analysis.

In GMQL queries, the dimensions and measures can be made up of both spatial and non-spatial attributes, functions, and aggregates. The spatial attributes presented in both clauses require special treatment, which depends on the type of the mined knowledge, as mentioned in Section 5.3.

When a user discovers spatial associations, or spatial classifications he/she may also specify predicates using the WITH RESPECT TO clause.

Example 5.3 The query to describe cities using clustering algorithm can be presented as:

MINE CLUSTERS DESCRIBING "Cities"
ANALYZE sum(pop), count(*)
WITH RESPECT TO Cities.geo, pop, 1.52 * med_fam_income AS CDN_income
FROM Cities

The data cube constructed for the query has three dimensions: *geo, pop* and *CDN_income* and two measurements: *sum(pop)* and *count(*)*. The values of *med_fam_income* are multiplied by the exchange ratio of Canadian and US dollars and the CDN_income concept hierarchy is used for the generalization. □

5.2.2 FROM Clause

The "**FROM** <relation(s)>" clause specifies the names of the relations used in the mining process. In addition to the tables stored in the database objects can be selected using a mouse. A user can select objects individually, or select an area containing required objects. The selected objects are stored in the SELECTION relation which can be a part of the "**FROM** <relation(s)>" clause like any other relation. The concept of the SELECTION relation is adapted from the SQL version used in MapInfo Professional [102].

Example 5.4 The query to characterize clusters of cities located within areas specified by a user can be presented as:

> MINE CLUSTERS DESCRIBING "Cities"
> WITH RESPECT TO Cities.geo, pop, med_fam_income
> FROM SELECTION, Cities
> WHERE Cities.geo INTERSECTS SELECTION.geo

This query presents the use of the SELECTION table. □

5.2.3 WHERE Clause

The "**WHERE** <conditions>" clause specifies conditions that are to be satisfied by the relevant data.

Like Spatial SQL the conditions may contain basic spatial relationships like *intersects, contains,* or *within*, in addition to non-spatial operations. For example, by using a condition "parks.geo WITHIN states.geo AND states.name="Washington"", we select parks that are located in Washington state. In addition to that, Spatial SQL operations like DISTANCE or DIRECTION can also be used for the construction of conditions used in the WHERE clause.

The conditions may contain high level values for the attributes stored in the concept hierarchies. However, these high level values may not exist in the database. In that case, every tuple whose values generalize to these high level concepts, will be retrieved.

Example 5.5 The query to characterize *Western* states can be presented as:

> MINE CHARACTERISTICS DESCRIBING "States"
> WITH RESPECT TO States.geo, pop, med_fam_income
> FROM States
> WHERE States.name = "West"

The high level concept *"West"* covers states like California, Oregon and Washington. □

5.3 Types of the Discovered Knowledge

In this section we present examples of the GMQL queries for different types of queries. The results of the visualization of the query execution are presented in Chapter 6.

5.3.1 Characteristic Rules

A characteristic rule is an assertion, which describes a concept satisfied by all or most of the selected objects. It provides users with a multi-level, multi-angle view of the relevant set of data in a spatial database [99]. For example, properties of US states can be summarized by a characteristic rule.

When mining characteristic and discriminant rules, if spatial_term(s) are used in the ANALYZE clause, then the non-spatial dominant algorithm is used. The pointers to the objects whose non-spatial properties correspond to the values for the attributes stored in the dimensions of the data cube are collected in the corresponding cells of the cube.

Example 5.6 The query to characterize US states using non-spatial concept hierarchies can be presented as:

> MINE CHARACTERISTICS DESCRIBING "USA_states"
> ANALYZE states_census.geo
> WITH RESPECT TO pop90, med_fam_income, with_bachelor_degp
> FROM states_census

The non-spatial dominant generalization algorithm is applied. The states with the same non-spatial attributes values can be merged for the presentation. □

If spatial_attribute is specified in the WITH RESPECT TO clause then, the spatial dominant algorithm is used for the mining. The spatial objects are generalized according to spatial concept hierarchies. See Section 2.3.3 for the details on non-spatial dominant generalization and spatial-dominant-generalization.

Example 5.7 The query to characterize US states using spatial and non-spatial concept hierarchies, can be presented as:

```
MINE CHARACTERISTICS DESCRIBING "USA_states"
ANALYZE sum(pop90)
WITH RESPECT TO states_census.geo,
        statename, pop90, med_fam_income, with_bachelor_degp
FROM states_census
```

The spatial dominant generalization algorithm is applied. The high level regions from the spatial concept hierarchy are used for the presentation. □

Otherwise, if spatial_term is not used in any of these clauses then the mining process is a non-spatial one.

5.3.2 Comparison Rules

Comparison rules are the rules that discriminate the general features of different classes of the relevant sets of data in a database. They compare one set of data, known as the target class, to other set(s) of data, known as the contrasting class(es).

The syntax for mining comparison rules is similar to that for mining characteristic rules, except that it enables definitions of the target class and one or more of the contrasting classes. The name of the target class is specified after the **FOR** keyword and objects that belong to the target class have to satisfy conditions in the **WHERE** clause which follows. The names of each of the contrasting classes are specified after the **VERSUS** keyword which is followed by the specifications of the conditions satisfied by the contrasting objects.

Example 5.8 The query to compare stores with high profits versus stores with low profits can be presented as:

MINE COMPARISON DESCRIBING "Stores"
ANALYZE sum(sales)
WITH RESPECT TO states_census.geo, statename, type
FROM states_census
WHERE Stores.geo INSIDE states_census.geo
FOR "High profit stores"
  WHERE profit_rate > 30
VERSUS "Low profit stores"
  WHERE profit_rate < 10

The stores are compared in relation to states where they are located and types of stores. □

5.3.3 Clustering Rules

During the execution of spatial clustering query, the clusters of points with relevant non-spatial description are discovered.

For clustering queries the spatial_attribute in the WITH RESPECT TO clause represents spatial properties of the objects to be clustered.

Example 5.9 The query to mine the description of the stores clusters can be presented as:

MINE CLUSTERS DESCRIBING "Stores"
ANALYZE sum(sales)
WITH RESPECT TO Stores.geo, type
FROM Stores

The discovered clusters of stores will be analyzed according to the types of store and total sales. □

5.3.4 Spatial Association Rules

As defined in Chapter 3, the spatial association rule is a rule in the form of

$$P_1 \wedge \ldots \wedge P_m \quad \rightarrow \quad Q_1 \wedge \ldots \wedge Q_n. \quad (s\%, c\%)$$

where at least one of the predicates $P_1, \ldots, P_m, Q_1, \ldots, Q_n$ is a spatial predicate, and s% and c% are respectively the *support* and the *confidence* of the rule.

For spatial associations, the first spatial_term in the WITH RESPECT TO clause represents spatial properties of the objects to be described. The WITH RESPECT TO clause contains templates for the spatial predicates which can be used in the rules.

Example 5.10 The query to characterize stores using spatial association rules can be presented as:

```
MINE SPATIAL ASSOCIATIONS DESCRIBING "Stores"
WITH RESPECT TO Stores.geo, Stores.profit,
        CLOSE_TO(Stores.geo, Shopping_centers.geo, "2 km")
FROM Stores, Shopping_centers
```

The stores will be described according to their profits and spatial relation to the shopping centers which can act as "magnets" for the customers. □

5.3.5 Classification Rules

As described in Chapter 4, supervised classification is a data mining technique where the data stored in a database is analyzed in order to find rules that describe the partition of the data into a given set of classes. Each object in a database (in relational databases, tuples are treated as objects) is assumed to belong to a predefined class, determined by one of the attributes, called the *class label attribute*. Other attributes are used in the rules, which assign data to classes.

The ANALYZE clause specifies the class label attribute, predicate, or function while the WITH RESPECT TO clause presents attributes, predicates and functions used for the prediction of the class.

Example 5.11 The query to classify stores according to their types can be presented as:

```
MINE CLASSIFICATION DESCRIBING "Stores"
ANALYZE type
WITH RESPECT TO sales, profit,
        CLOSE_TO(Stores.geo, Shopping_centers.geo, "2 km")
FROM states_census
```

The stores will be classified using their sales, profits and spatial relation to the shopping centers. □

5.4 Threshold Settings

The set of clauses "**SET** <threshold_specification>" specifies various types of thresholds that constrain the search for interesting knowledge. The thresholds may include:

1. *support* and *confidence* thresholds for mining association rules [3], as described in Chapter 3,
2. *exception* and *classification* thresholds for mining classification rules [78], and
3. *noise* threshold for the discovery of characteristic rules [61].

For example, the clause "**SET CONFIDENCE THRESHOLD** 0.5" specifies that at least 50% of objects must satisfy the association rules presented to the user.

5.5 Concept Hierarchy Manipulation

The last set of clauses "<hierarchy_specification>" specifies the type of hierarchy used in the mining process. GMQL queries contain statements for building concept hierarchies and manipulation of data cube as described for DMQL in [66].

The relational concept hierarchies can be built by concept grouping, which explicitly shows that one group of concepts is at a level lower than another group. Examples of such definitions are shown below.

```
DEFINE {B.C., Alberta, Manitoba, Saskatchewan}
    UNDER {Western_Canada} IN province
DEFINE {Western_Canada, Central_Canada, Maritime_Provinces}
    UNDER {Canada} IN province
```

Spatial concept hierarchies can be defined based on a non-spatial attribute which is a primary key for the table containing spatial attributes. For example, the spatial hierarchy for Canada can be defined based on *province* attribute as shown below.

```
DEFINE SPATIAL HIERARCHY Canada BASED ON province
```

In this case both spatial attribute and primary key attribute would be generalized simultaneously.

Another possibility is to the define concept hierarchy using a pointing device and a SELECTION table using the following construct.

DEFINE SELECTION UNDER '{'literal'}' IN hier_name

For example, a user may select Western provinces in Canada and use the statement:

DEFINE SELECTION UNDER {Western_Canada} IN Canada

to define the concept *Western_Canada*. The next level of the concept hierarchies would be defined using named concepts, like *Western_Canada* or *Central_Canada* in the same way as for non-spatial attributes.

The methods for automatic generation of numerical concept hierarchies and their dynamical adjustment were analyzed in [62]. Such methods for generation and adjustment of spatial concept hierarchies require further investigation.

5.6 Future Extensions

In this section we discuss extensions to GMQL syntax, for already discussed query types, and for the queries used for mining of other types of knowledge.

5.6.1 Extensions to Characteristic, Comparison and Clustering Rules

An open research problem is to add spatial predicates for the discovery of characteristic, comparison and clustering rules for spatial objects. Because a set of spatial predicates can be used to describe a single spatial object, the problem of generalization of such descriptions is similar to the generalization of set oriented data. The generalization of set-valued attributes was studied in [69] where it was proposed that set-valued attributes can be generalized to set-valued attributes or single-valued attributes. For example, the fact that a store is close to an elementary school, a high school and a convenience store can be generalized to the fact that a store is close to two schools and a small store, it can be generalized to the fact that it is close to three objects; or it can be generalized to the fact that most of the nearby objects are schools.

Other possible handling of the generalization of spatial predicates is to store a collection of pointers that correspond to spatial objects, which satisfy these predicates.

Also, more than one spatial term can be used in the WITH RESPECT TO clause. In this case a map overlay operation could be used to extract regions for the description. Such approach works well with categorical attributes, but poses problems in case of some numerical attributes. For example, when map overlay operation finds the intersection of Kings county with geological region of sandstone, then one can assign categorical values like *wet*, or *sedimentary*, but the assignment of numerical values like *population = 2,454,034* to the intersection of two regions would be a problem because the exact data distribution within a region is usually unknown. The solution to this may be through the assumption of an even distribution of attribute values through the area that would lead to the assignment of values proportional to the areas of the new regions.

5.6.2 Extensions to Association Rules

The ANALYZE clause can be added to compare properties of objects that satisfy both the body and the head of the rule with the objects that satisfy only the body of the rule.

Example 5.12 To compare sales of the stores from the Query 5.10 one can submit the following query:

```
MINE SPATIAL ASSOCIATIONS DESCRIBING "Stores"
ANALYZE avg(sales)
WITH RESPECT TO Stores.geo, Stores.profit,
        CLOSE_TO(Stores.geo, Shopping_centers.geo, "2 km")
FROM Stores, Shopping_centers
```

As the result of the query the following association rule might be discovered:

$$is\_a(x, store) \wedge close\_to(x, shopping\_center) \rightarrow profit(x, high)$$
$$(30\%, 60\%) \ \ (\$600{,}453,\ \$850{,}456)$$

The rule shows that, although profits for stores that are close to the shopping centers are mostly high, the sales of these high profit tend to be smaller ($600,453 on average) than the sales for low profit stores in the proximity of shopping malls ($850,456 on average). □

5.6.3 Other Types of Rules

Not all data mining algorithms, outlined in Chapter 1, were discussed in the previous section. In addition to the extensions for characteristic, comparison, association and clustering rules proposed previously, it is interesting to analyze future GMQL extensions for other tasks.

We did not implement algorithms for prediction, deviation analysis, dependency modeling, time-series analysis, statistical analysis and other data mining approaches. The syntax for these tasks may be similar to the syntax for the previously presented queries. The non-spatial prediction was analyzed in [25]. The proposed syntax for mining predictions is similar to the syntax for mining classification rules. In the prediction task instead of mapping a data item into one of the categorical classes, the system maps a data item into a numerical value. For example, if we want to predict profit of a store we may submit a query similar to the query from Example 5.11.

Example 5.13 The query to predict profit of a store can be presented as:

```
MINE CLASSIFICATION DESCRIBING "Stores"
ANALYZE profit
WITH RESPECT TO sales, type,
        CLOSE_TO(Stores.geo, Shopping_centers.geo, "2 km")
FROM states_census
```

The profits will be predicted using sales of the stores, types of the stores and spatial relation between stores and shopping centers. □

As GMQL follows Spatial SQL syntax for the specification of the data used in the mining process for all data mining tasks, the relevant data may be specified using "[ANALYZE <aggregate_list>] WITH RESPECT TO <attribute_list> FROM <relation(s)> [WHERE <conditions>] [GROUP BY <attribute_list>] [HAVING <conditions>]" clauses in the similar way as for the tasks presented in Section 5.3, and in the similar way as Spatial SQL selects data in its queries. Also, the syntax for setting the thresholds can be easily extended to accommodate new types of thresholds.

Therefore, the presented GMQL syntax can be easily extended to accommodate mining of other types of spatial knowledge.

5.7 Summary

A spatial data mining language can be used as a tool providing a programming interface to different data mining approaches and systems, much like SQL can be treated as a tool providing a unified query interface to various database systems.

The functionality of DMQL language for mining data from relational databases was extended toward mining spatial databases. We have designed GMQL - Geo Mining Query Language, which enables dealing with spatial data types and mining of different types of spatial knowledge. In GMQL queries a user can specify data to be analyzed with spatial operators and spatial relationships which use Spatial SQL constructs. Spatial functions and spatial predicates can also be used to represent discovered knowledge. The high level of user interaction in the specification of the objects to be mined is enabled by a SELECTION table which stores objects,! selected using a mouse.

The possible extensions to the language were proposed. GMQL extensions that would allow meta-rule guided mining, data prediction and data evolution mining, etc., are interesting topics for further research.

The detailed outline of GMQL grammar in BNF form is presented in the Appendix to the thesis.

A version of GMQL was implemented in the GeoMiner spatial data mining system which is presented in the next chapter.

Whatever you have learned or received or heard from me,
or seen in me–put it into practice.
And the God of peace will be with you.
Philippians 4:9

Chapter 6

GeoMiner: a Spatial Data Mining System Prototype

The rapid progress of research into data mining and data warehousing in recent years [46], resulted in the development of many data mining systems for mining knowledge in relational databases and data warehouses. Unfortunately, there have not been many spatial data mining systems reported in previous studies.

The GeoMiner research group in our *Intelligent Database Systems Research Laboratory* has been working on data mining and especially spatial data mining for several years. Based on previous research into spatial data mining [88, 90, 91, 99, 109] and experience in the research and development of a relational data mining system DBMiner [61, 63, 67], a spatial data mining system prototype, GeoMiner, has been designed and developed. The current GeoMiner system includes the *on-line analytical processing (OLAP) module* and *spatial data mining modules* which consist of data mining modules for mining *characteristic rules, comparison rules, association rules, clustering rules*, and *classification rules.* The overall design of the system and general design of the modules was done by the author of the thesis, who also developed association and clustering modules and partially developed characterizer module. The rest of the development to the characterizer module was done by N. Stefanović, who also developed a comparator module and the visualization of the association rules. The classification module was developed by Q. Chen and W. Gong. The GeoMiner OLAP engine is based on the DBMiner OLAP module which was designed and developed by Y. Tam. The YACC parser was implemented by Y. Lu.

GeoMiner integrates OLAP and Data Mining technologies. Such integration enables the extension of knowledge discovery techniques toward interactive mining on multiple levels of abstraction, from large databases and data warehouses. Such an approach is known as On-Line Analytical Mining (OLAM) [71].

The SAND (Spatial And Non-spatial Data) architecture [8] is applied in the modeling of spatial databases, whereas the spatial data mining modules include mining both spatial knowledge and the relationships between spatial and non-spatial components. A spatial data mining language, GMQL (Geo-Mining Query Language), was used for the formulation of mining queries. In addition, an interactive, user-friendly data mining interface is constructed and tools for visualization of spatial data and the discovered spatial knowledge are implemented.

The rest of the chapter is organized as follows. The outline of the GeoMiner system architecture is presented in Section 6.1. A detailed description of mining modules and the knowledge discovery process is presented in Section 6.2. A summary and a discussion of our on-going research are in Section 6.3.

6.1 Architecture

Based on our exploration of Geographic Information Systems, On-Line Analytical Processing (OLAP), data warehousing, and data mining in relational databases we have identified the following desirable characteristics for a Spatial Data Mining system:

- The system should enable interactive mining of different types of knowledge on multiple levels of concept hierarchies.
- The discovery process should be started by a query enabling specification of the type of knowledge to be discovered, set of data to be used and thresholds to be set.
- The graphical user interface should enable (1) the selection of objects for mining using a mouse and (2) for the transformation of the results of the mining for presentation using tables, charts, graphs, and maps. A user should be able to print out the results or store them on a disk.
- The system should accept different formats of data or enable easy transformation of data to the format used by GeoMiner.
- The performance of the system should enable analysis of large spatial databases.

In the process of GeoMiner construction, that would satisfy the presented characteristics we have analyzed two approaches to the design of a spatial data mining system:

1. Design and implement a new special purpose system.

2. Extend and integrate existing systems to accommodate knowledge discovery from large spatial databases.

The first approach may provide better overall performance, due to the fact that it is designed for the specific purpose. It might also be easier to optimize the mining process, due to the fact that spatial database engine would be implemented as well. However, this approach would require a much larger implementation effort and might not have many features of GIS system that are useful for data analysis. The second approach can make use of well-developed existing systems in order to reduce the implementation time and to use data stored in existing spatial databases without the necessity to transform the data to a new format. It would also extend data analysis features using existing GIS functionality.

We decided to use a commercial GIS system MapInfo Professional [102] for the implementation of the spatial database and the Graphical User Interface. This choice allows us to use the full capability of this system for the manipulation and storage of data, as well as the presentation of spatial and non-spatial information in different formats. MapInfo also provides the access to many external databases using ODBC functionality, import/export of data from/to other spatial data formats, and querying of spatial data using its native version of Spatial SQL. The multidimensional manipulation of data on different levels of concept hierarchies is enabled by the data cube of the DBMiner system, researched and developed in our laboratory [67]. The DBMiner knowledge discovery engine includes the OLAP module which uses techniques for data cube construction and manipulation [65], and for attribute-oriented induction [61]. The functions for non-spatial OLAP operations are directed to the DBMiner system; whereas those for mining spatial data and the analysis of relationships between spatial and non-spatial data are performed by the dedicated GeoMiner functions.

The subset of GMQL was created to enable expression of the mining queries. The GMQL parser was implemented using YACC utility. The access to MapInfo functionality is provided by the procedures written in MapBasic, i.e., a specialized version of Basic provided by MapInfo Corporation. MapBasic procedures communicate with other parts of GeoMiner,

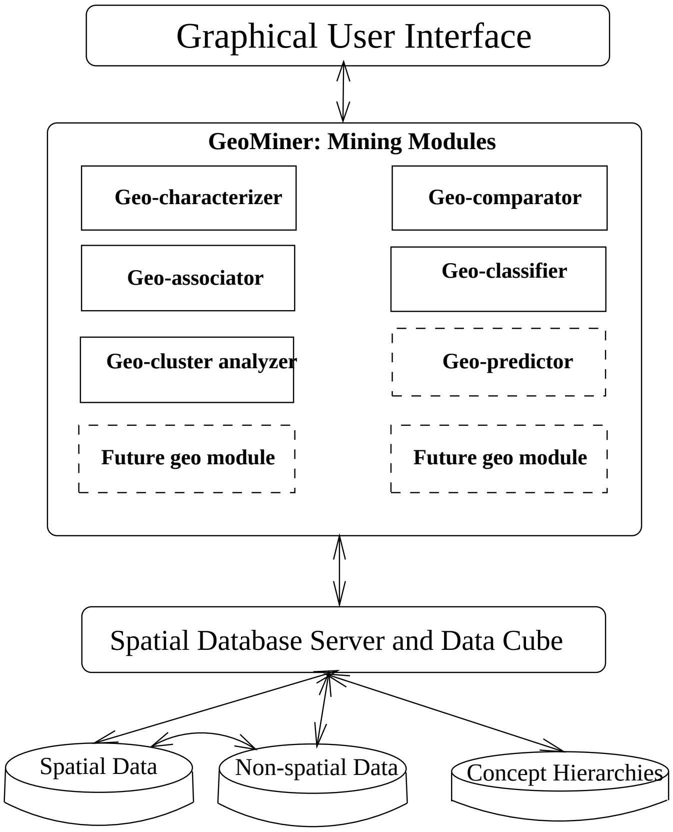

Figure 6.1: General architecture of GeoMiner

including the OLAP module, using DLL functionality. The spatial data mining modules and the OLAP module are written mainly in C++, with some parts written in MapBasic.

Figure 6.1 shows the general architecture of GeoMiner which consists of (1) a graphical user interface for interactive mining and the display of data mining results in the form of tables, charts, maps, etc., (2) a set of discovery modules, including five existing modules: *geo-characterizer, geo-comparator, geo-classifier, geo-cluster analyzer*, and *geo-associator*, and two planned ones: *geo-predictor*, and *geo-pattern analyzer*, (3) a spatial database server, which includes *MapInfo Professional 4.1.2* Geographic Information System [102], (4) a data cube mining engine based on DBMiner discovery kernel for multidimensional manipulation of data, and (5) the data- and knowledge-base, storing non-spatial data, spatial data, and concept hierarchies.

The functionalities of the five existing modules are described as follows.

- Geo-characterizer.

  This module mines a set of characteristic rules at multiple levels of abstraction from a relevant set of data in a spatial database [99]. For example, using this module, the following question can be answered: *given spatial hierarchies of the United States, describe general income patterns according to region partitions.*

- Geo-comparator.

  This module mines a set of comparison rules, which contrast the general features of different classes of the relevant sets of data in a database. For example, it may show *the differences in migration patterns in United States*, or find *the clusters or features related to the locations which differentiate the profiting stores from the non-profiting ones.*

- Geo-cluster analyzer.

  This module finds clusters of points with relevant non-spatial descriptions. For example, one can find *how the stores are clustered and then, mine descriptions for each cluster to find proper marketing strategies for each cluster.*

- Geo-associator.

  This module finds a set of strong spatial-related association rules from the relevant set(s) of data in a spatial database [88]. For example, one can show *the associations between golf courses and other nearby objects such as parks, main roads, lakes.*

- Geo-classifier

  This module adopts a generalization-based decision-tree induction method to build a decision tree, which enables supervised classification of a relevant dataset [78]. The classification tree is displayed and by clicking on any of the nodes of the resulting tree the user may highlight corresponding regions on the map. For example, one can *classify states in the US according to the medium family income in a state.*

The OLAP module enables multidimensional knowledge discovery on multiple levels as described in Chapter 2. The multidimensional data analysis utilizes concept hierarchies, which are stored in a database. The relational concept hierarchies are stored in the form of tables. The spatial concept hierarchies contain precomputed spatial aggregations to accelerate the data analysis process.

The OLAP module constructs a data cube to enable fast manipulation of multidimensional data. In order to enable usage of spatial data, the concept of *nongeneralizable attributes* was introduced. The nongeneralizable attribute is an attribute, for which there is no concept hierarchy, and which is not dropped in the attribute-oriented induction process during the data generalization. Such attributes are used as pointers to spatial objects, while non-spatial descriptions of spatial objects can be generalized to enable mining on multiple concept levels. The spatial objects can be single objects or collections of objects. For example, clusters are collections of points.

The integration of OLAP and data mining technologies provides opportunities for efficient and effective mining in large repositories of spatial data. The OLAP technology enables fast manipulation of multidimensional data while spatial data mining algorithms enable right choices of relevant data and relevant dimensions, fast data transformation, online selection of data mining functions, and user-friendly visualization of the mining results.

Communication between mining modules and the OLAP engine is based on the following four functions:

- CONSTRUCT_CUBE(TABLE_NAME). This function transfers a non-spatial table from the MapInfo database to the SQL server. Then, it constructs the data cube based on the transferred data with dimensions and measurements specified in the GMQL query. It performs the attribute-oriented induction process to generalize data as it is described in Section 6.2.

- GET_GENERALIZED_TABLE(TABLE_NAME). After the generalization, processed data from the data cube is passed to MapInfo in the form of a generalized relation using this function. The generalized relation contains attributes for every dimension and measurement of the data cube.

- SPECIALIZE(ATTRIBUTE_NAME) and GENERALIZE(ATTRIBUTE_NAME). During the interactive mining process drill-down (specialization) and roll-up (generalization) are possible by using these two functions. The attribute *Attribute_Name* is one of the dimensions of the data cube.

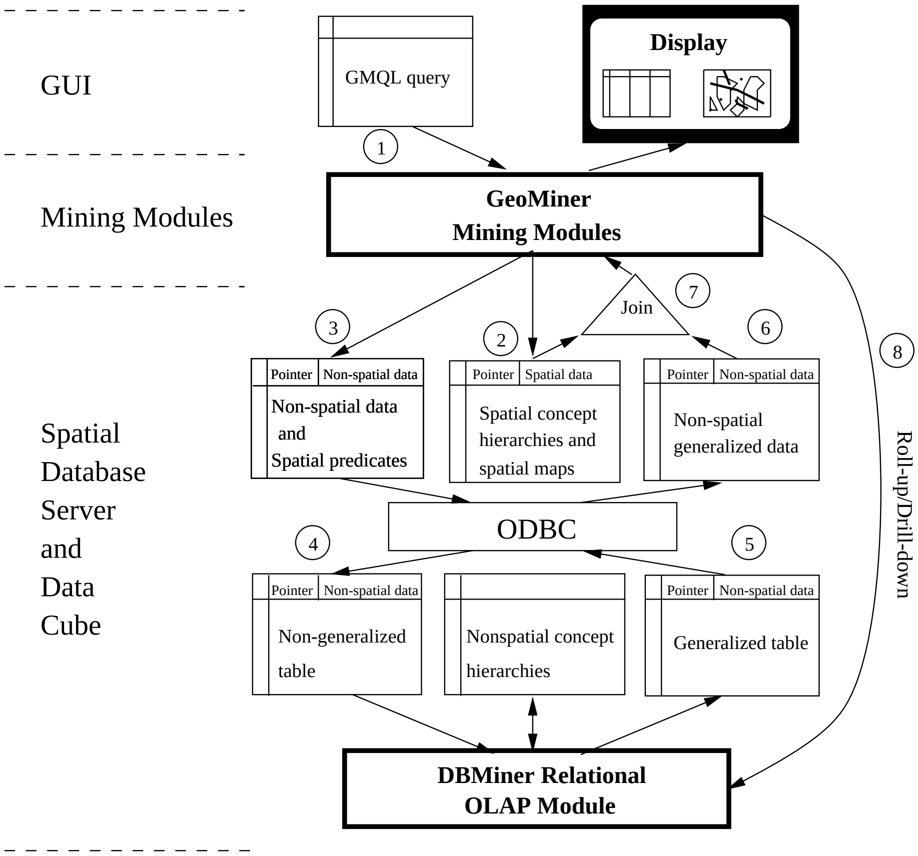

Figure 6.2: The communication process between GeoMiner mining modules and the OLAP Module

Figure 6.2 presents the interaction between GeoMiner mining modules and the OLAP Module. The data mining task is initialized by a GMQL query (1). This query is parsed, the data cube structure is initialized, and the SQL query is constructed to extract data relevant to the mining task. The data relevant to the mining task is retrieved from the MapInfo Database. This data can include spatial maps, spatial predicates and non-spatial data. The basic data generalization and data transfer processes are explained as follows. The spatial map portion of the data is processed and stored in the MapInfo table (2). The other part of the data, which may contain non-spatial data and/or spatial predicates is stored in a single relational table. The data cube is populated based on the data stored in this table. It is performed by CONSTRUCT_CUBE function which transfers the relational table to the *OLAP Module* using an ODBC connection ((3) and (4)) and then it constructs the data cube. The OLAP module uses a data cube to work with multidimensional data. The data cube enables fast manipulation (*roll-up, drill-down, slicing, dicing*) and analysis of a large amount of data. Microsoft SQL Server was used in the implementation of the data cube.

The generalized data is passed from data cube to MapInfo through ODBC module ((5) and (6)) in the form of generalized table. It is done by the GET_GENERALIZED_TABLE function. The generalized table is joined (7) with spatial portion of the data and further processed by mining modules before the presentation of the results on the screen. The join between spatial map and generalized data transferred from the OLAP module can be based on (i) distinct attribute like *statename*, which can be generalized together with spatial data, or (ii) based on *nongeneralizable* attribute, which serves as a pointer to spatial part of data. Further manipulation of the data such as slicing, roll-up and drill-down can be done by accessing data cube directly (8) using SPECIALIZE and GENERALIZE functions. Then, the generalized or specialized results from the OLAP module can be transferred to the mining modules using GET_GENERALIZED_TABLE function.

When spatial predicates, like *g_close_to* are used in the mining process the data manipulation is similar to the process described above. The predicates are stored in a relational table, which is transferred to the OLAP module for the generalization. The generalized predicates are further analyzed by the GeoMiner mining modules.

By using only relational tables for the generalization process we were able to use standard DBMiner data cube structure modified to accept nongeneralizable attributes. Other OLAP products that use the OLE DB for OLAP standard [105] or the MDAPI^{TM} standard [112] could also be used for the implementation of the OLAP module.

Finally, the user interface of GeoMiner enables presentation of the results in multiple forms of outputs, including generalized maps, generalized relations, cross-tabulation, multiple level rules, charts. The results can be further transformed using MapInfo functionality, transferred to other applications using OLE features, or they can be printed out. The detailed description of the mining process is presented for each module in Section 6.2.

6.2 Spatial Data Mining Process

In the development of the GeoMiner knowledge discovery modules, several data mining techniques are adopted based on the results of previous research, including *attribute-oriented induction* [61], *spatial-dominated and non-spatial-dominated generalization techniques* [99], *exploration of refined geo-algorithms step-by-step* [88, 92], *generalization based classification* [78, 92], etc.

It is important to note that the flexibility of GeoMiner is at its ability to perform multi-level spatial mining. Data and objects in spatial databases often contain detailed information at the primitive level of abstraction. Thus, it is desirable to summarize a large set of data and present the summary at a high abstraction level. For example, one may like to summarize the detailed temperature and precipitation data in an area and present its general weather pattern. This requires *generalization-based data mining*, which first abstracts a large set of relevant data from a low concept level to relatively high ones and then performs knowledge extraction on the generalized data. However, with a blind generalization, data may be generalized to a level that is too high. Users may like to drill-down to specific features or regions to examine the details. Therefore, the system provides not only the power of generalization, but also the power of specialization for flexible concept description. This feature is similar to that of OLAP in many data warehouses. However, a plain OLAP engine cannot find general concept descriptions, associations, and classifications, without proper extensions, whereas a data mining system will find such distinct features at multiple levels of abstraction. To present data on different levels of concept hierarchies a user may apply operators such as *drill-down* to show data in more detailed perspective, or *roll-up* when more general description is desired. Moreover, a user can also *slice* data along spatial dimension, e.g., he/she can choose regions on the map whose detailed description will be presented. For example, in Figure 6.3 only descriptions of southern regions of US are presented.

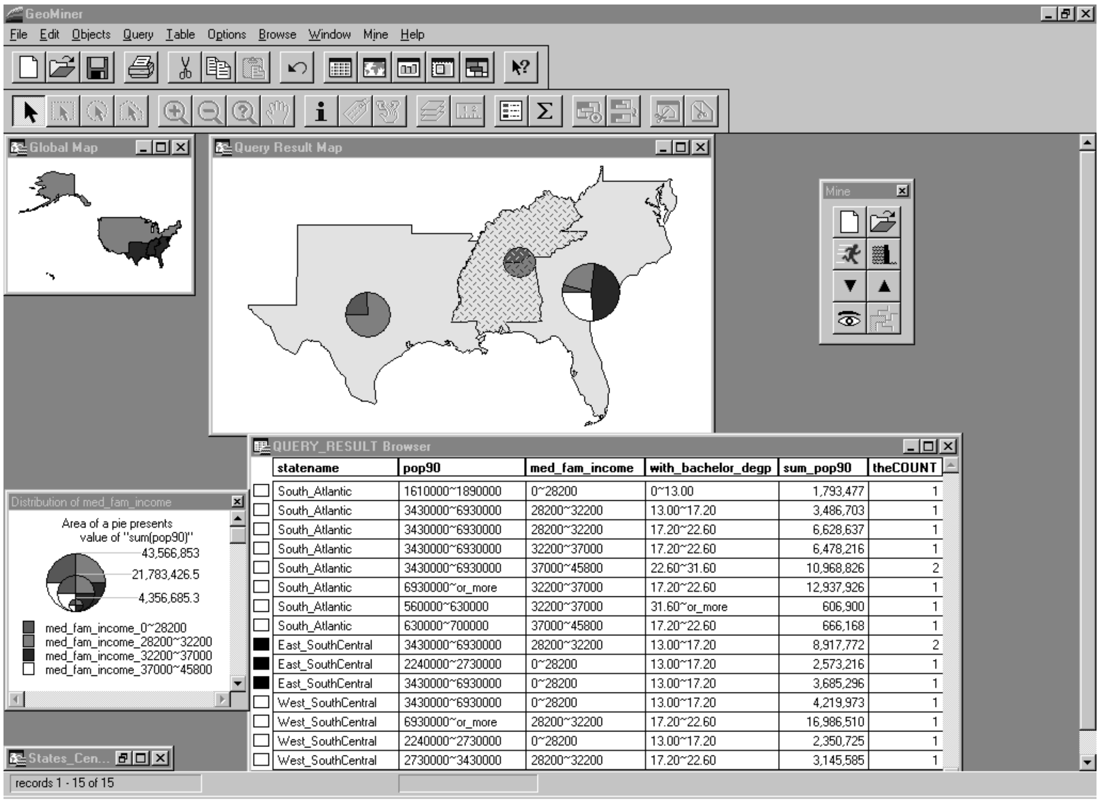

Figure 6.3: Mining spatial characteristics using GeoMiner

6.2.1 Characteristic Rules

A characteristic rule is an assertion, which describes a concept satisfied by all or most of the selected objects. It provides users with a multi-level, multi-angle view of the relevant set of data in a spatial database [99]. The algorithm to mine spatial characteristic rules may be presented as following:

STEP 1: PARSE THE GMQL QUERY.
STEP 2: RETRIEVE THE RELEVANT DATA.
STEP 3: PERFORM SPATIAL GENERALIZATION.
STEP 4: PERFORM ATTRIBUTE-ORIENTED INDUCTION ON NON-SPATIAL PART OF THE DATA.
STEP 5: JOIN SPATIAL AND NON-SPATIAL DATA.
- STEP 6: PRESENT RESULTS.

The core part of the algorithm is the attribute-oriented induction process used for generalization of spatial and non-spatial data.

The relational data is analyzed using the attribute-oriented induction technique described in Chapter 2. Together with relational data, spatial data is also generalized. Spatial generalization triggers the merge of the connected regions according to spatial concept hierarchy. The spatial merge generates a set of consolidated regions. Generalization of the spatial objects continues until the spatial *generalization threshold* is reached. The spatial generalization threshold is defined as the maximum number of regions in the resulting map. To speed up the process the merged regions are stored in a precomputed MapInfo table. The generalized spatial and non-spatial knowledge is presented to the user.

The details of the algorithm are explained in the following example.

Example 6.1 To characterize *demographic patterns* in *United States* by presenting the relationships between the regions with respect to population size, medium family income, and percentage of population with bachelor degree, the spatial data mining query can be expressed in GMQL as follows.

MINE CHARACTERISTICS DESCRIBING "USA_states"
ANALYZE sum(pop90)
WITH RESPECT TO states_census.geo,
statename, pop90, med_fam_income, with_bachelor_degp
FROM states_census

In the first step of the algorithm the GMQL query is parsed and data cube structure and SQL query are constructed.

In Step 2 the constructed SQL query is used to retrieve the map of US states, together with the data related to *name of the state* (statename), *population* (pop90), *medium family income* (med_fam_income), and *percentage of population holding bachelor degree* (with_bachelor_degp). Non-spatial part of the data constitutes the relational table GEOMINER_TABLE(STATENAME, POP90, MED_FAM_INCOME, WITH_BACHELOR_DEGP).

In Step 3 spatial descriptions of states are generalized into larger regions such as *South Atlantic*, *East South Central*, *West South Central*. During this generalization the connected regions are merged. The spatial regions are represented by the *statename* attribute on the non-spatial part of the analysis. Thus, the spatial generalization process corresponds to the generalization of the attribute *statename* by the OLAP module in the next step.

In Step 4 non-spatial part of the data is processed. The GEOMINER_TABLE table created in Step 2 is transferred to the DBMiner OLAP module by the CONSTRUCT_CUBE(GEOMINER_TABLE) function. The four-dimensional data cube is constructed. The non-spatial dimensions are statename, pop90, med_fam_income, and with_bachelor_degp. The cube also contains two measures sum(pop90) and count(*), which characterize quantitative properties of the cube cells. The values of attributes statename, pop90, med_fam_income, and with_bachelor_degp are generalized to higher level concepts based on their concept hierarchies. The tuples with identical generalized attributes are merged and the number of tuples is stored in count(*) measure. In our example, sum of population for states in merged regions is also stored to enable the presentation of measure *"sum(pop90)"*. For example, for the state Tennessee the value of the *statename* attribute is generalized to East South Central, value of *pop90* is generalized from 4877185 to the range 3430000~6930000, value of *med_fam_income* is generalized from $29546 to a range 28200~32200, and value of *with_bachelor_degp* is generalized from 16 to the range 13.00~17.20. The values of the attributes for the state of Alabama are generalized to the same values as for the state of

Tennessee. Thus, the generalized tuples of these two states are merged together, the sum of populations for both states is stored in the *sum(pop90)* measure, and value of *count(*)* measure is updated to 2.

In Step 5 the data cube is transformed into a generalized table QUERY_RESULT(STATENAME, POP90, MED_FAM_INCOME, WITH_BACHELOR_DEGP, SUM_POP90, THECOUNT). The *statename, pop90, med_fam_income,* and *with_bachelor_degp* attributes of the table contain generalized values. The *sum_pop90* attribute contains values of sum(pop90) measure and *theCOUNT* attribute contains values of count(*) measure. The generalized table is transferred from the OLAP module to a mining module via ODBC connection by GET_GENERA-LIZED_TABLE(QUERY_RESULT) function. An example of the generalized table is presented in Figure 6.3. The generalized table is joined with the precomputed generalized spatial regions based on the value of the attribute *statename*. Finally, the results of the data mining task are presented to a user. Figure 6.3 is a snapshot of the results of such mining task. It shows the results of knowledge discovery process after slicing along spatial dimension to present details describing southern regions of United States. The results are presented both in the form of *pie charts* on the map and in the form of generalized relation. Pie charts are based on the cross table that is generated from the generalized table. Notice that drill-down or roll-up can be performed interactively on such generalized data to zoom-in or zoom-out the generalized spatial region or to examine the details of their associated non-spatial properties. □

In the presented example, mining is performed by generalizing first along the spatial hierarchy. After this, the process triggers the attribute-oriented induction process on non-spatial attributes to describe non-spatial properties of the generalized spatial regions. Thus, such generalization is called *spatial data-dominated generalization* [99].

Alternatively, *non-spatial data-dominated generalization* can be performed by GeoMiner if a spatial region hierarchy is not given. This method also starts with the collection of all data relevant to the user query. For example, the DB interface can extract the economic data relevant to the state and time period specified in the query. After the data retrieval the algorithm performs attribute-oriented induction on the non-spatial attributes, generalizing them to a higher (more general) concept level. For example, the population values in the range (2,000,000; 10,000,000] can be generalized to the concept *large*. The *generalization threshold* is used to determine when to stop the generalization process. In this step the

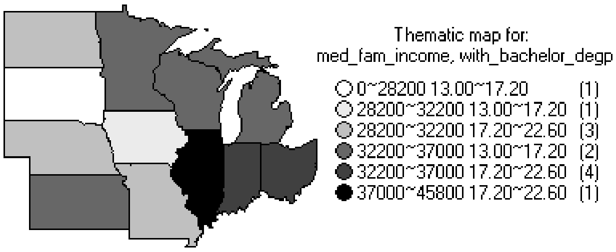

Figure 6.4: Results of non-spatial data-dominated generalization.

pointers to spatial objects are collected as a set and put with the generalized non-spatial data. In the last step of the algorithm, a user can choose to merge neighboring areas with the same generalized attributes. The results of this process can be visualized in the form of thematic map, which presents regions according to their generalized non-spatial descriptions. Figure 6.4 presents an example of such thematic map. It presents a thematic map for two attributes: med_fam_income and with_bachelor_degp. The process of creating such thematic maps can be accelerated if spatial aggregates of regions that are most frequently merged are precomputed and stored in a spatial database. An efficient algorithm to find such aggregates was described in [70].

6.2.2 Comparison Rules

The process of mining comparison rules is similar to the mining of characteristic rules. The additional non-generalizable attribute is used to distinguish among classes. The example 6.2 presents the details of this process.

Example 6.2 The task of finding the *migration patterns* in the United States may be formulated as a comparison between states with large increase in the population and states where the population decreased. The query to compare such two sets of objects with respect to *spatial location* (geo), *name of the state* (statename), *population* (pop), *ratio of crimes per 100,000* (crimes100000R), and *medium family income* (med_fam_income) can be expressed in GMQL as follows.

```
MINE COMPARISON DESCRIBING "Migration Patterns"
ANALYZE sum(pop)
WITH RESPECT TO states_census.geo, statename,
        pop, crimes100000R, med_fam_income
FROM states_census
WHERE pop > 0
FOR "Significant_population_increase"
  WHERE pop80_92P > 20
VERSUS "Population_decrease"
  WHERE pop80_92P < 0
```

To process this query, the system parses the GMQL query and constructs three SQL queries to retrieve three relevant sets of data: "*Significant_population_increase*", "*Population_decrease*", and "*Others*". The "*Significant_population_increase*" set contains states where the size of the population between 1980 and 1992 increased by more than 20% (where pop80_92P > 20). The "*Population_decrease*" set contains states where population decreased in the same period of time. Finally, the "*Others*" set consists of objects that do not belong to the first two sets. The non-spatial descriptions are stored into a single table and the nongeneralizable attribute *SID* (set ID) is used to distinguish which set an object belongs to. Then, the data analysis process is performed in the way similar to the one for the mining of the characteristic rules. Non-spatial descriptions of objects are passed to the OLAP module and generalized to the same level. Then, *roll-up*, *drill-down* and *slice* operations can be applied synchronously on all classes. Generalization and specialization are based on the hierarchies associated with the following dimensions: spatial dimension (*states_census.geo*) and four non-spatial dimensions: *statename, pop, crimes100000R,* and *med_fam_income*. The measures contain two values, *count* as the default measure and *sum(pop)* to present sum of population in particular regions. The generalized data is transferred to MapInfo using ODBC. It is joined with the precomputed generalized spatial regions based on the value of the attribute *statename* in the same manner as for spatial-data dominant generalization for the characteristic rules. The value of the attribute *SID* presents information about the set to which generalized tuples belong. An example of the result of the comparison query execution is presented in Figure 6.5 and an example of the generalized table for the comparison rules is presented in Table 6.1. □

statename	pop	crimes100000R	med_fam_income	SID	sum(pop)	count
New_England	>420000	3600.00˜5400.00	32200˜37000	3	1790686.0	2
New_England	>420000	3600.00˜5400.00	37000˜45800	3	7019889.0	2
New_England	>420000	5400.00˜6900.00	>45800	3	3287116.0	1
New_England	>420000	0.00˜3600.00	37000˜45800	1	1109252.0	1
...	...	...	...	...	...	...
WN_Central	>420000	0.00˜3600.00	28200˜32200	2	3415555.0	2
WN_Central	>420000	3600.00˜5400.00	28200˜32200	3	1578385.0	1
WN_Central	>420000	3600.00˜5400.00	32200˜37000	3	4375099.0	1
...	...	...	...	...	...	...

Table 6.1: The generalized table for the comparison rules.

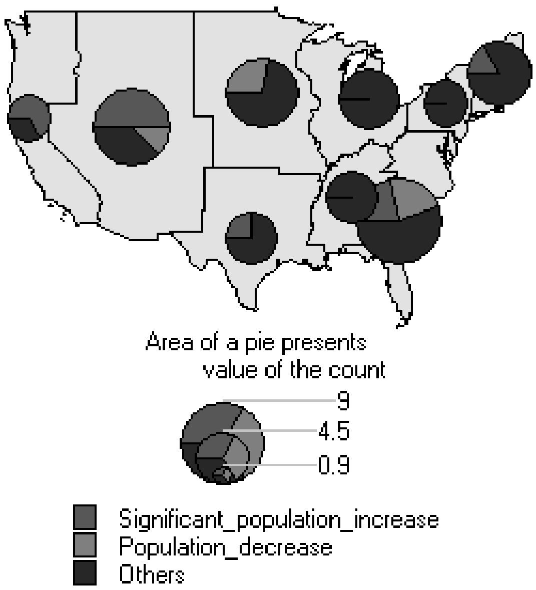

Figure 6.5: Results of a comparison query.

6.2.3 Spatial Clustering

During the execution of spatial clustering query clusters of points with relevant non-spatial description are discovered. The algorithm to mine spatial clustering rules may be presented as the following:

STEP 1: PARSE THE GMQL QUERY.
STEP 2: RETRIEVE THE RELEVANT DATA.
STEP 3: PERFORM SPATIAL CLUSTERING AND STORE THE CLUSTERS.
STEP 4: CREATE A TABLE CONTAINING NON-SPATIAL AND THE CLUSTER IDS OF THE POINTS.
STEP 5: PERFORM ATTRIBUTE-ORIENTED INDUCTION ON NON-SPATIAL DESCRIPTIONS OF THE POINTS.
STEP 6: JOIN SPATIAL AND NON-SPATIAL DATA.
STEP 7: PRESENT RESULTS.

The clustering process uses an effective algorithm [109], which finds the natural number of clusters. Other clustering algorithms can be also adapted for this purpose. Once clusters are discovered in Step 3 their spatial representations in the form of convex hulls of the sets of points belonging to the clusters are stored in a MapInfo table together with corresponding CIDs (Cluster IDs). The points are assigned to each cluster and a table with the Non-spatial attributes is created in Step 4. Each row in the table represents non-spatial properties of a point and the *CID* of the cluster to which the point belongs.

The rest of the mining process is similar to the mining of characteristic rules except that spatial generalization is not performed as the clusters already represent the generalized spatial description of the points. The table with the non-spatial descriptions of the points is transferred to the OLAP module where the data cube is constructed and the generalized descriptions of points are found for each cluster. The *CID* attribute is treated as a non-generalizable attribute and other attributes are analyzed using attribute-oriented induction method described in the Section 6.2.1. After the OLAP analysis the generalized relation is transferred to GeoMiner where it is joined with the map that contains the spatial representation of the clusters. The join is based on the value of the *CID*. The results of the mining process may be presented in the form of a generalized relation, or displayed on the map as weighted pie charts. Using the graphical user interface a user may change the number of

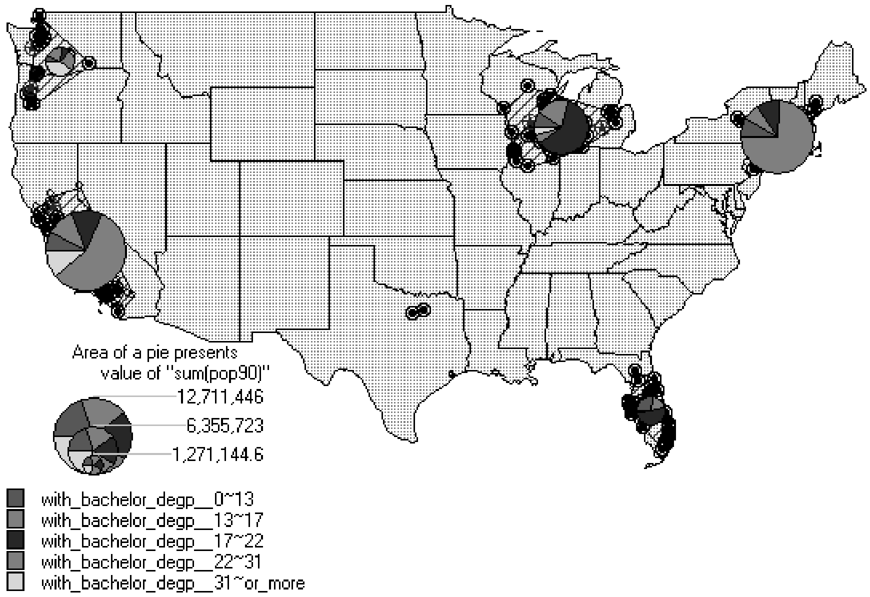

Figure 6.6: Results of clustering query

clusters to be found if a larger or a smaller number of clusters is preferred. Furthermore, OLAP operations like roll-up, drill-down can be performed on the non-spatial descriptions of clusters.

Example 6.3 To find the description of spatial clusters of points on a map a user can submit the following GMQL query.

MINE CLUSTERS DESCRIBING "DB_Cities"
ANALYZE sum(pop90)
WITH RESPECT TO DB_Cities.geo,
    pop90, med_fam_income, with_bachelor_degp
FROM DB_Cities

After the execution of the first three steps spatial clusters are found. Points from DB_Cities table are assigned to each cluster and the table with the attributes *CID, pop90, med_fam_income,* and *with_bachelor_degp* is created in Step 4. In the next step this table is passed to the OLAP module and the generalization process is performed. The results are passed to GeoMiner, and joined with the spatial description of the clusters in Step 6. Finally, the results are presented to a user in Step 7. Figure 6.6 shows five clusters of points and the distribution of population in the clusters according to education. The size of the pie charts shows values of *sum(pop90)* aggregation. The reader may notice that two cities in Texas are treated as outliers because they are considered to be too far away from other clusters and they cannot create their own cluster. One can observe that cities in the *western* clusters have larger percentage of people with bachelor degrees then *Florida* or *Midwest* clusters. □

6.2.4 Spatial Association Rules

Mining spatial association rules is based on our study presented in Chapter 3. The algorithm to process spatial association query using *g_close_to* predicates may be presented as the following:

STEP 1: PARSE THE GMQL QUERY.
STEP 2: RETRIEVE THE RELEVANT SPATIAL PREDICATES.
  STEP 2.1: CONSTRUCT BUFFERS AROUND DESCRIBED OBJECTS.
  STEP 2.2: PERFORM SPATIAL JOIN *(the optimization technique presented in Chapter 3 may be used).*
STEP 3: PERFORM GENERALIZATION OF THE PREDICATES.
STEP 4: RETRIEVE THE PREDICATES FROM THE OLAP MODULE.
STEP 5: MINE ASSOCIATION RULES.
STEP 6: PRESENT RESULTS.

During the mining process spatial predicates are extracted using spatial join query. Then, the values of predicates are generalized by the OLAP module. Finally, the algorithm Apriori [3] is used for the discovery of association rules from the table that contains generalized predicates. We use the following example to explain the details of the algorithm.

Example 6.4 A GMQL mining query to find strong spatial associations between golf courses and other objects such as lakes, roads, parks, schools, which are located in Washington State can be defined as follows.

```
MINE SPATIAL ASSOCIATIONS DESCRIBING "Golf Courses"
WITH RESPECT TO Washington.geo, Washington_Golf_Courses.CFCC,
        G_CLOSE_TO(Washington.geo, Washington_Golf_Courses.geo, "3 km")
FROM Washington_Golf_Courses, Washington
WHERE Washington.CFCC <> "Golf Course"
SET SUPPORT THRESHOLD 0.4
SET CONFIDENCE THRESHOLD 0.5
```

The *G_CLOSE_TO* predicate specifies distance threshold. In the presented query only objects within 3 kilometers distance from a golf course are taken into account in the process of finding association rules. According to the specified thresholds only rules describing at least 40% of golf courses with confidence larger than 50% will be presented.

Multi-level association rules can be mined by climbing up and stepping down along the CFCC dimension. CFCC stands for "Census Feature Class Code" and is used for description of objects in the TIGER/Line census data [141]. The Census Feature Class Codes are organized into a concept hierarchy, which is provided by the U.S. Census Bureau. For example, CFCC descriptions like schools and churches can be generalized to educational/religious institutions.

In Step 1 of the algorithm the GMQL query is parsed and data cube structure is constructed. In Step 2 spatial predicates are extracted from the spatial database using SQL query. First, the 3 kilometers buffers are constructed around described objects. Then, the spatial join query is performed to extract predicates. An optimization technique described in Chapter 3 can be used to avoid costly computations on the objects that would not be used in the process of constructing association rules because they do not satisfy the support threshold. This optimization first performs spatial join based only on the Minimum Bounding Rectangles (MBRs) of the objects. Then, it executes detailed join only for objects that satisfy the support threshold on the MBR level. For example, if only a small number of MBRs of the buffers intersect with the MBRs of campgrounds then the predicate g_close_to(golf_course, campground) will not have the high support and will not be used in the construction of association rules. Thus, it does not make sense to perform spatial join of the buffers of golf courses with the campgrounds.

The predicates are stored in a form of relational table G_CLOSE_TO(OID, CFCC). This table has two attributes. The first attribute - *OID* contains object ID, which uniquely identifies the described golf course, whereas the second attribute - *CFCC* describes an object satisfying the predicate *G_CLOSE_TO(Washington.geo, Washington_Golf_Courses.geo, "3 km")*. For example, if a golf course with $OID = 4$ is within 3 km from a lake and from an airport this information is represented by two tuples. The first tuple has OID equal to 4 and CFCC equal to H30, which corresponds to CFFC for a lake. The second tuple has OID equal to 4 and CFCC equal to D51, which corresponds to CFFC for an airport. To enable multidimensional analysis the *G_CLOSE_TO* table is transferred to the OLAP module in Step 3. The *OID* attribute is treated as a nongeneralizable attribute and values for the *CFCC* attribute are generalized. In Step 4 the G_CLOSE_TO table is retrieved back to MapInfo database and then, in Step 5 the generalized relation is processed by the geo-associator module to extract spatial association rules. Finally, in Step 6 the discovered rules are presented to the user.

The following association rules are derived from the spatial data set.

is_a(X, "Golf Course") → close_to(X, "Man-Made Channel") (61%, 61%)
is_a(X, "Golf Course") ∧ close_to(X, "Open space") → close_to(X, "Secondary road") (64%, 78%)

The first rule presents that 61% of all golf courses are within 3 kilometers from the Man-Made Channels. The second rule presents that 78% of golf courses that are within 3 kilometers from objects classified as "Open spaces" are also less than 3 kilometers from secondary roads. The support for the second rule is 64%, i.e., 64% of all golf courses are within 3 kilometers from both secondary roads and open spaces. □

Spatial association rules can be presented in the form of the table, which shows body and head of the rule. Association rules can also be shown in the form of association rules graph. An example of such graph is presented in Figure 6.7. The circles present conjunctions of spatial and non-spatial predicates. The size of a circle presents the support of the predicates (the larger the circle the more objects satisfy predicate). The arrows present rules with circles serving as the body and the head of the rule. The width of the connecting arrow is proportional to the support of the rule and the width of the arrow's head is proportional to the confidence of the rule.

6.2.5 Classification Rules

The classification method used in the system applies a generalization-based approach to classification [78]. During the execution of the algorithm values of non-spatial attributes

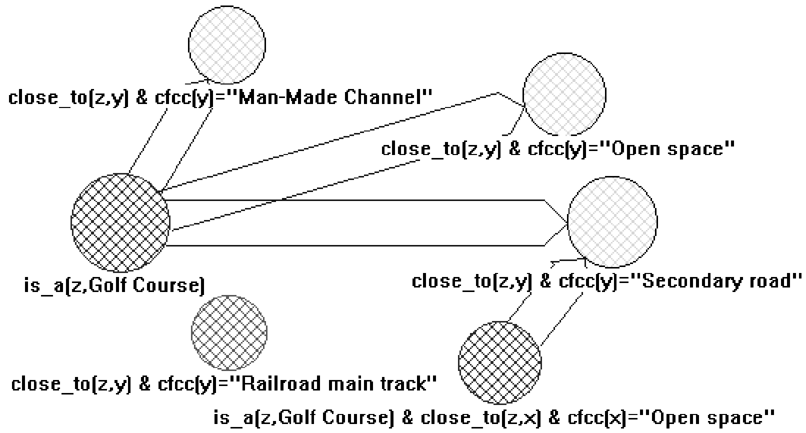

Figure 6.7: Association rules graph

of the objects are generalized and then induction of decision tree on such generalized data is performed. The classification tree is displayed and a user may view spatial location of region within particular tree node by clicking on the node (Figure 6.8).

Example 6.5 The task of classification of U.S. states according to the *ratio of crimes per 100,000 of population* may be expressed in GMQL in the following way.

MINE CLASSIFICATION DESCRIBING "US States"
ANALYZE crimes100000R
WITH RESPECT TO states_census.geo, statename,
    capita_income, with_bachelor_degp
FROM states_census

In the query attributes *statename, capita_income,* and *with_bachelor_degp* are used to determine the class an object belongs to. The values of these attributes are generalized by the OLAP module. These generalized values are used in the construction of the decision tree. Figure 6.8 presents a classification tree and states corresponding to one of the nodes. Each node of the tree shows the distribution of the states according to the number of crimes per 100,000 people (crimes100000R). The description of the classes is presented in the legend. The states are first divided into nodes according to the value of *capita_income*

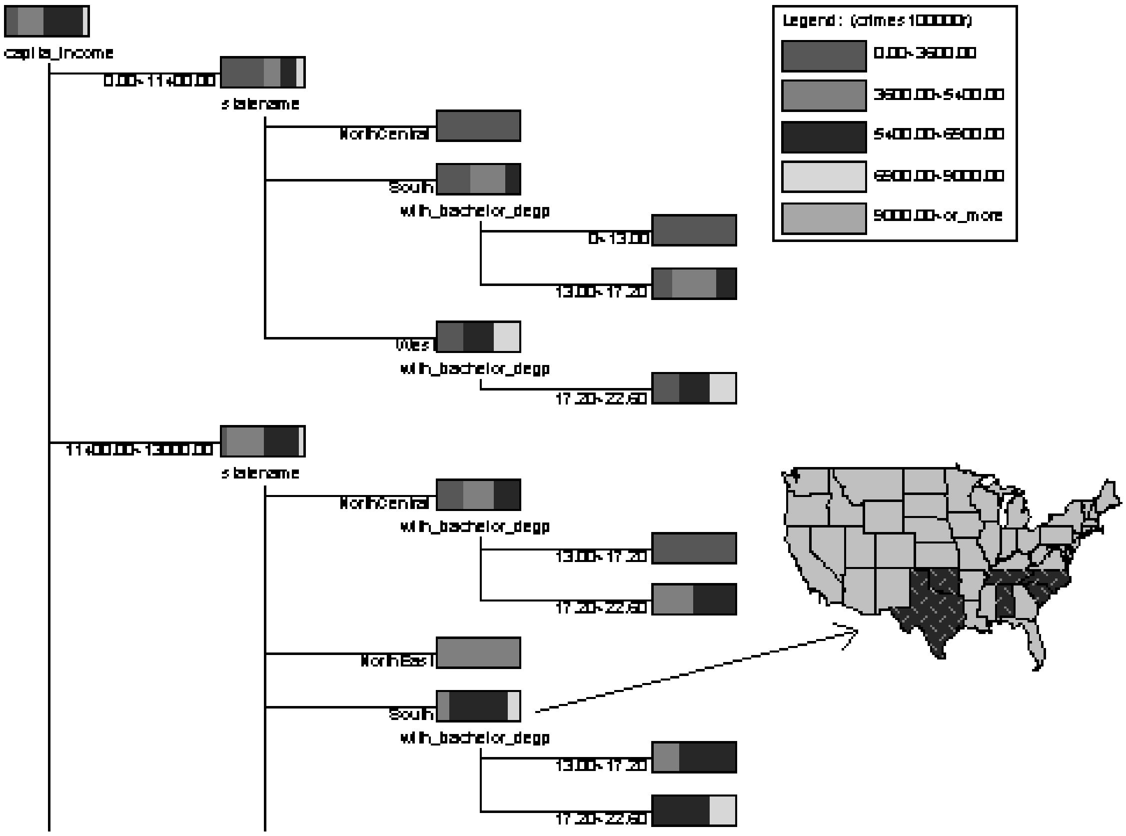

Figure 6.8: Decision Tree

attribute as this attribute provides the largest information gain, i.e., if we branch on this attribute, we will get subsets with the highest class purity. If the states in the particular node are widely distributed between crime ratio ranges, then the process of classification continues recursively by splitting nodes according to the values of one of the attributes from the WITH RESPECT TO clause. In the presented example, *statename* attribute is chosen as the second attribute in the rules construction. In Figure 6.8 we present a map and a decision tree resulting from the execution of the query from Example 6.5. The states highlighted on the map correspond to the node (capita_income $\in$ (11400.00~13000.00) $\wedge$ statename = "South") of the decision tree. □

The algorithm used in the classification process takes into account only non-spatial attributes of the data. Note that some of these attributes may be spatially related like *statename* in the presented example, but no other spatial relationships between objects or spatial predicates are used for the construction of decision tree. Recently the new algorithms, which also use spatial properties in the classification process, were proposed [37, 92] and presented in Chapter 4. The incorporation of these algorithms into a new version of GeoMiner is planned.

6.3 Summary

We have designed and developed a spatial data mining system prototype, GeoMiner, which satisfies the requirements for a spatial data mining systems presented in Section 6.1. The current system mines five kinds of knowledge from geo-spatial databases and demonstrates a promising direction in the study of spatial data mining. It presents the integration of spatial data mining algorithms with the relational OLAP technology and Geographic Information Systems. The spatial data mining methods enable fast extraction and transformation of data for the extraction of different types of knowledge. The data cube technology used by the OLAP module provides efficient way of multidimensional data generalization and manipulation, while the non-generalizable attributes enable merging spatial and non-spatial parts of the data. The MapInfo GIS allows for using different data repositories, visual presentation of the mining results and OLE automation. The GMQL (Geo-Mining Query Language) was developed for the presentation of user queries. The experiments show that an interesting spatial knowledge can be discovered from large spatial databases in an efficient manner. The system was presented in SIGMOD-97 and KDD-97 conferences. The concise description of the system [68] was used in teaching at Simon Fraser University, University of California in Santa Barbara and Queensland University of Technology.

6.3.1 GeoMiner vs. Other KDD and GIS Systems

GeoMiner provides a lot of features found in Geographic Information Systems and relational data mining systems. On top of these functionalities new features were added. Most business data contain geographic information, yet systems for knowledge discovery in relational databases do not provide any high level description using geographic maps. Moreover, these systems cannot be queried using spatial predicates, which are essential in discovering patterns in a spatial dataset. Finally, relational data mining systems do not provide support for geocoding of addresses, a process that adds a geographic dimension to relational data.

On the other hand, Geographic Information Systems do not analyze multidimensional data on multiple levels of concept hierarchies, they do not provide algorithms for efficient analysis of large amount of data, and the query language support for data analysis in GIS is limited. GeoMiner system uses the strength of a relational OLAP engine together with the analytical capabilities of Geographic Information System and adds fast spatial knowledge discovery algorithms to enable efficient and powerful data mining in large spatial databases.

6.3.2 Future Work

This preliminary experimental and development work on spatial data mining has promoted our further study in this direction. Some on-going themes that can be developed in the future are illustrated as follows.

- Further enhancement of the power and efficiency of the knowledge discovery mechanisms, including the improvement of rule quality and system performance for the existing functional modules, should be done. This includes the further exploration of multi-step spatial data mining algorithms [88, 92]. The current use of such algorithms is limited by the of-the-shelf spatial database system used in the implementation. The use of such database system has made some of the optimization techniques difficult to implement, due to the lack of access to the details of some of the spatial operations.

- The current version of geo-classifier builds a decision tree based only on non-spatial properties of objects. Then, the spatial visualization of the results is performed. The new algorithms, that can also take into account spatial properties of the classified objects, should be integrated with the system. Such an algorithm was proposed in Chapter 4.

- The OLAP module uses a relational data cube, which may perform only relational aggregation of data. To strengthen the spatial OLAP capabilities of the system one can add aggregation features of spatial data cube presented in [70].
- Further development of high-level, user-friendly interfaces for an interactive spatial data mining and a visual presentation of the discovered knowledge should be investigated.

The end of a matter is better than its beginning
Ecclesiastes 7:8

Chapter 7

Conclusions

Satellite and remote sensing methods have produced a huge amount of geographical data. In addition, modern databases contain information like addresses that can be easily converted into spatial form through the process of geocoding. Many organizations have large databases containing such information about their customers. This amount of data creates a need for the development of scalable and efficient spatial data mining algorithms and construction of spatial knowledge discovery systems.

Spatial data has been analyzed by statisticians and geographers for a long time. Most of the methods traditionally used in spatial data analysis do not take into account challenges posed by large amounts of data and do not make much use of the spatial query processing methods. This thesis addresses spatial data mining from large databases through the analysis of methods and algorithms that make use of spatial access methods and can be used to efficiently discover knowledge from large spatial data repositories.

In the first chapter of this thesis we presented three application examples. Below we show how the results of this thesis may help in the processing of the tasks presented in those examples.

Example 1.1 The analysis of crime sites
During the process of analysis, a number of GMQL queries is constructed and different spatial association rules are found. The rules are mined using Algorithm 3.1, which allows for efficient discovery of a complete set of association based on a given query, The set of rules is visualized and it is further analyzed by a criminologist. The rules such as "*is_a(X, fight)* $\wedge$ *close_to(X, pub)* $\rightarrow$ *is_a(Y, fight)* $\wedge$ *close_to(Y, casino)* $\wedge$ *after(X, Y)*" can be

presented. The confirmatory spatial data analysis may be performed on such rule to find out if the rule is present based on random distribution of data, or it presents the pattern of association. In such a case, if a fight is reported in a vicinity of a pub then a patrol can be sent out to prevent a possible fight in casino area.

Example 1.2 Real estate investment analysis
A GMQL query is submitted and a decision tree is build based on the real estate data. The data analysts can see which parameters are the most important for the price changes and how the characteristic features of the houses interact with each other. The range of price changes for new investments can be estimated based on the decision tree.

Example 1.3 Analysis of the regional sales
The summary of the sales is visualized using a map and charts. The analyst can easily change the level of details and "slice" data to concentrate on specific time intervals and specific regions. The map may show that during a Christmas season profits in the regions close to border are lower than in other parts of the market. This can be attributed to the easier cross-border shopping in these regions. Appropriate actions may be taken to increase profits.

While current GISs provide powerful tools for visualization of spatial data and for solving of some analytical problems, there are some classes of tasks that cannot be solved using existing technology in an efficient way. This thesis certainly does not provide the solution to all problems, but we believe we make a step forward in spatial data mining by providing the extension of data analysis capabilities through the introduction of the concept of spatial association rules and through the presentation of algorithms for mining spatial associations and spatial classification. The knowledge can be discovered using the developed spatial data mining prototype *GeoMiner* and conveniently presented on multiple levels.

The reminder of this chapter is organized in the following way. In Section 7.1 we present contributions of this thesis to the area of spatial data mining. In Section 7.2 we itemize strengths of our mining approach. Section 7.3 contains the some directions for future research. The thesis concludes in Section 7.4 with final thoughts on the content of this work and on future challenges.

7.1 Contributions to Spatial Data Mining

Based on previous studies on spatial data mining and mining in relational and transaction-based databases, we proposed and studied methods for mining rules in large spatial databases. Discovery of spatial rules may disclose interesting relationships among spatial and/or non-spatial data in large spatial databases and thus, it represents a new and promising research direction in data mining, spatial databases and Geographic Information Systems.

We have contributed to knowledge discovery in spatial databases through the results presented in this thesis in the following ways:

- We proposed an approach to spatial data mining which uses both (1) non-spatial properties of the spatial objects and (2) attributes, predicates and functions describing spatial relations between described objects and other features located in the spatial proximity of the described objects. Both spatial and non-spatial descriptions are generalized and transformed into a form of predicates. The discovered knowledge is presented on multiple levels of concepts.

- We designed and implemented an algorithm for mining of spatial association rules. The algorithm closely ties the mining process with spatial query processing allowing for fast discovery of interesting rules. The filtration process performed on a coarse resolution level filters out irrelevant attributes leading to a significantly smaller amount of computation on refined level.

- We proposed and analyzed an algorithm for spatial classification. The performed experiments show that due to the relevance analysis the speed and quality of the classification are much better than in the previous study [37].

- We proposed and developed a language for spatial data mining. The GMQL (Geo Mining Query Language) is an extension to the DMQL (Data Mining Query Language) used in DBMiner system. GMQL allows forming mining queries in which a user can specify type of the knowledge to be discovered, set of relevant data, thresholds and concept hierarchies. The language uses the syntax similar to Spatial SQL [34] for the specification of relevant data.

- We designed and implemented a spatial data mining system prototype, *GeoMiner*.

Five data mining modules were constructed: geo-characterizer, geo-comparator, geo-classifier, geo-cluster analyzer, and geo-associator. The implemented OLAP module enables multidimensional manipulation of data. To our knowledge, this is the first spatial data mining system that combines a number of spatial data mining methods with a data mining language and a commercial spatial data engine to allow discovery and presentation of knowledge on multiple levels.

The presented work also contributed to the educational process in Computer Science. Some of the papers developed during the research process have been used so far in senior-level and graduate education at universities in Australia, Canada, Hong Kong, India, and United States.

7.2 Major Strengths of the Approach

The approach to spatial data mining presented in this thesis has the following major strengths.

- Substantial reduction of the candidate set through approximate spatial computations.

  The proposed algorithms first apply efficient spatial computation algorithms at a coarse resolution scale (such as coarse generalized close_to, *coarse_g_close_to*), using approximate spatial computation algorithms. Only a substantially reduced set of candidate spatial predicates, which are worth detailed examination, is computed by refined spatial techniques.

- Incorporation of spatial access methods into the mining process.

  In processing of the mining queries, Spatial Access Methods like R-trees and R*-trees can be used both for the reduction of candidate predicates and for the refined computation using data filtered on the coarse level.

- Incorporation of relational OLAP module into spatial data mining.

  The use of non-generalizable attributes allows for multi-level analysis of spatial and non-spatial predicates and attributes as well as for the combination of multi-level maps with their multidimensional non-spatial descriptions. This way, most of the current OLAP systems can be incorporated into spatial data mining systems.

- User-oriented process.

  The knowledge discovery is centered toward a user, who starts the spatial data mining process by posting a GMQL query. The GeoMiner Graphical User Interface allows for flexible visualization of the discoveries using rules, maps, charts, etc. The user can zoom in/out both in spatial and non-spatial dimensions according to the user's needs. Moreover, different concept hierarchies can be specified to tailor the discovery process to the specific tasks.

- Quantitative and qualitative presentation of results.

  The discovered knowledge can be presented using both quantitative and qualitative forms. Quantitative information is used in the aggregate functions stored in the data cube cells that show information about a number of data items. Qualitative representation in the form of predicates allows for efficient discovery and human friendly representation.

- Easy extensions toward other types of set-oriented data.

  Recently new types of data have been collected in large amounts. Text, multimedia [146], object-oriented [69], WWW, and other types of data can be described using sets of predicates. Thus, our approach which uses sets of predicates for data description can be extended toward mining of these types of data repositories.

7.3 Future Directions

As we mentioned earlier, data mining is an emerging field going back no more than the late 1980s. Spatial data mining is even younger since data mining researchers first concentrated on data mining in relational databases. As the technology advances new databases, including data warehouses, object-oriented (OO), parallel, and distributed systems are being created. Spatial data mining methods should be analyzed taking into account this progress, as well as other data mining tasks, which were not explored so far. Some of the research directions are presented as follows.

- Coupling of spatial data mining methods with spatial database engine: The methods presented in this thesis use spatial database engine to process mining queries. More investigations is needed to analyze what kinds of functions should be implemented in spatial databases to allow efficient processing of many different data mining tasks. Chaudhuri [23] discussed some of the SQL extensions that could be used by a number of data mining algorithms and interact with core SQL primitives. In the case of spatial operations the main cost is associated with processing the geometric and topological functions. We believe that the possible functionalities include sampling, processing of approximate queries on rough (e.g., MBR) level, and efficient extraction of spatial predicates. Such functionalities may be used in client/server architecture for data mining.

- PARALLEL AND DISTRIBUTED DATA MINING: Due to the high volume of spatial data, mining using parallel machines or distributed farms of workstations may improve the performance, especially when dealing with high volumes of spatial data. Moreover, spatial data can be geographically distributed among different places. Such data can be mined through networks, including Internet. New approaches can be necessary for distributed spatial query processing and data mining. We expect that parallel knowledge discovery will be a growing research issue in both relational and spatial data mining.

- OLAP FOR SPATIAL DATA: While OLAP approaches to the analysis of relational data became widely used by the commercial users, the spatial OLAP is still in its infancy stage. The further investigation of algorithms and data structures for the multidimensional summarization of spatial data is needed. Some of techniques for the materialization of spatial cubes have been studied by Stefanović *et al.* [134].

- DATA MINING IN SPATIAL OBJECT-ORIENTED DATABASES: How can the OO approach be used to design a spatial database [108, 114] and how can knowledge be mined from these databases? It is an important question since many researchers have pointed out that OO database may be a better choice for handling spatial data rather than traditional relational or extended relational models. Therefore, exploiting OO technology in data mining is an area with enormous potential. Techniques for generalizations of complex data objects, methods and class hierarchies have been studied by Han *et al.* [69].

- Cooperation between statistical analysis and data mining: The enhancement of data mining techniques with mature statistical methods may produce interesting new techniques, which may work well with different kinds of problems and on different data types. For example, the statistical techniques may help in judgement on interestingness and significance of rules.

- Cooperation between cartographic modeling and data mining: The cartographic modeling has been used for the analysis of spatial data for a long time [139]. The spatial data mining methods could be enhanced by the incorporation of approaches used in the modeling of geographical data.

- Mining under uncertainty: Use of evidential reasoning [57] can be explored in the mining process for image databases and other databases where uncertainty modeling has to be done. As mentioned in Bell *et al*'s [12], evidential theory can model uncertainty better than traditional probabilistic models, like Bayesian methods. Fuzzy set approach was applied to spatial reasoning [30, 31] and it can be extended to spatial data mining.

- Mining in the presence of spatial errors: Spatial data is often digitalized with a finite precision. Therefore, we need to deal with errors in the data, boundaries that do not match boundaries in other thematic maps, etc. Robust methods which can deal with such problems and algorithms for data cleaning are necessary.

- Alternative clustering techniques: Another interesting future direction is the clusterings of possibly overlapping objects like polygons as opposed to the clustering of points. Clusters can also maintain additional information about each object they contain, which can be the degree of membership. In this way, fuzzy clustering techniques can be used to accommodate objects having the same distance from the medoid.

- Mining spatial prediction, data deviation and evolution rules: One extension of current work in spatial data mining toward other types of spatial data mining tasks is to study prediction, data deviation, and data evolution rules. For example, we can find *spatial characteristic evolution rules* which summarizes the general characteristics of the changing data. During the mining process one can discover properties of the regions with the average growth of crops over 2% per year. One of the possible approaches may involve generalization using temporal spatial data. It may involve generalization over a sequence of maps collected during different time intervals. Then, comparison or summarization can be done to discover data evolution regularities. These rules may be used, for example, in medical imaging, where one would like to find out how certain features deviate from the norm or how they evolve over time. Some other applications may include discovery and prediction of weather patterns, and land use planning.

- USING MULTIPLE THEMATIC MAPS: We discussed generalization-based methods, which used a single thematic map during generalizations. Various applications demand spatial data mining to be conducted using multiple thematic maps. This would involve not only clustering but also spatial computations such as map overlay, spatial joins, etc. For example, to extract general weather patterns, it may be better to use *temperature* and *precipitation* thematic maps and to carry out generalization in both.

- INTERLEAVED GENERALIZATION: To extend the generalization-based methods, it is interesting to consider interleaving spatial and non-spatial generalizations to get the results in more efficient manner. Efficient processing can be achieved because usually spatial operations, like joins and overlays, are more expensive than non-spatial computations. Thus, by first generalizing the non-spatial component and minimally using spatial generalizations one may save a lot of computation time.

- MINING OF REMOTELY SENSED IMAGES: Large amount of remotely sensed images asks for more data mining methods for exploration of such data including detection of anomalies, finding similar pictures and discovery of relationships between various phenomena [29]. Image analysis techniques like mathematical morphology may be adapted for the mining of large databases.

- MULTIDIMENSIONAL RULE VISUALIZATION: Just discovering knowledge is not enough, because it has to be presented in a manner that the user can understand easily. One of the most effective ways of digesting the rules discovered is through graphical visualizations. Humans are very good at interpreting visual data and scenes. This fact should be exploited in the data mining process. Multidimensional data visualization has been studied [41, 80], but multidimensional rule visualization is still an immature area. Spatial data mining can use some well-developed visualization techniques in computer graphics.

- INTELLIGENT GIS: Methods for mining spatial data should be combined with advanced spatial databases, such as object-oriented spatial databases and spatio-temporal databases, as well as statistical analysis, spatial reasoning, and expert system technology to create *Intelligent GIS Systems* [95]. Such systems would enable wider use of spatial databases by non-technical users.

7.4 Final Thoughts

We have shown that spatial data mining is a promising field of research with wide applications in GIS, medical imaging, business geographics, etc. Although the field is quite young, a number of algorithms and techniques have been proposed to discover various kinds of knowledge from spatial data. We have surveyed existing methods for spatial data mining and mentioned their strengths and weaknesses. This led us to the development of new techniques and to the suggestions for the spatial data mining field in general. The variety of yet unexplored topics and problems make knowledge discovery in spatial databases an attractive and challenging research field. We believe that some of the suggestions that we mentioned have already been thought about by researchers and work may have already started on them. We hope to give the reader a general perspective and insight of this field.

With the huge amount of spatial data stored in various kinds of information repositories and increasingly popular use of spatial data applications in environment control, city planning, earth observation, and many other applications, we expect that spatial data mining will be an important research frontier for years to come.

Appendix A

GMQL Grammar

```
GMQL_statement ::= GMQL_query |
                   hierachy_definition |
                   hierarchy-level_manipulation
```

```
GMQL_query ::= MINE rule_header
               FROM relation_list
               [WHERE conditions]
               [GROUP BY attribute {, attribute}]
               [HAVING conditions]
               [SET threshold_specification THRESHOLD number]
               {hierarchy_specification}
```

```
rule_header ::= characteristic_header | comparison_header | association_header |
                clustering_header | classification_header
```

```
characteristic_header ::= CHARACTERISTICS [DESCRIBING literal]
                          [ANALYZE [spatial_term] measure_list]
                          WITH RESPECT TO [spatial_term [AS literal],] dimension_list
```

comparison_header ::= COMPARISON [DESCRIBING literal]
[ANALYZE [spatial_term] measure_list]
WITH RESPECT TO [spatial_term [AS literal],] dimension_list
FOR literal
WHERE conditions
comparison_sets_description

comparison_sets_description ::= VERSUS literal
WHERE conditions
[comparison_sets_description]

clustering_header ::= CLUSTERS [DESCRIBING literal]
[ANALYZE measure_list]
WITH RESPECT TO spatial_term [AS literal], dimension_list

association_header ::= SPATIAL ASSOCIATIONS [DESCRIBING literal]
WITH RESPECT TO spatial_term [AS literal], pred_dimension_list

classification_header ::= CLASSIFICATION [DESCRIBING literal]
ANALYZE nonspatial_term
WITH RESPECT TO [spatial_term [AS literal],] pred_dimension_list

dimension_list ::= $*$ | nonspatial_term [AS literal] {',' nonspatial_term [AS literal]}

pred_dimension_list ::= nonspatial_term [AS literal] {',' pred_dimension_list} |
spatial_predicate {',' pred_dimension_list}

nonspatial_term ::= nonspatial_term | spatial_2_nonspatial_function(spatial_term) |
nonspatial_function(nonspatial_term) | nonspatial_term operator nonspatial_term |
literal operator nonspatial_term | measure

nonspatial_function ::= SIN | COS | TAN | LOG | ATN | ACOS | ASIN | MONTH | YEAR | DAY

operator ::= $+$ | $-$ | $*$ | /

attribute ::= spatial_term | nonspatial_term

spatial_term ::= spatial_attribute | spatial_operation(spatial_attribute) | spatial_operation(spatial_term)

spatial_attribute ::= [relation_name '.'] geo | [relation_name '.'] geometry

nonspatial_term ::= [relation_name '.'] nonspatial_term_name

spatial_2_nonspatial_function ::= AREA | LENGTH | MIN_X | MAX_X | MIN_Y | MAX_Y |
CENTROID_X | CENTROID_Y

spatial_operation ::= BOUNDARY | BOUNDING_NODES | BOUNDING_EDGES |
BOUNDING_POLYGONS | INTERIOR | INTERIOR_NODES |
INTERIOR_EDGES | INTERIOR_POLYGONS

spatial_predicate ::= G_CLOSE_TO(spatial_term, spatial_term, '"'distance_specification'"') |
DIRECTION(spatial_term, spatial_term) |
TOPOLOGY(spatial_term, spatial_term)

relation_list ::= relation_name {',' relation_name} | SELECTION {',' relation_name}

measure_list ::= measure [AS literal] {',' measure [AS literal]}

measure ::= SUM(nonspatial_term) | AVG(nonspatial_term) |
MIN(nonspatial_term) | MAX(nonspatial_term) |
COUNT(*) | WGAVG(nonspatial_term, nonspatial_term)

conditions ::= NOT conditions | conditions AND conditions | conditions OR conditions |
compare_condition | in_condition

compare_condition ::= nonspatial_term standard_comparator nonspatial_term |
nonspatial_term standard_comparator literal |
spatial_term spatial_comparator spatial_term |
DIRECTION(spatial_term, spatial_term) standard_comparator nonspatial_term |
DIRECTION(spatial_term, spatial_term) standard_comparator literal |
DISTANCE(spatial_term, spatial_term) standard_comparator nonspatial_term |
DISTANCE(spatial_term, spatial_term) standard_comparator "distance_specification"

in_condition ::= IN ‘{’ literal {, literal} ‘}’ | IN SpatialSQLquery

standard_comparator ::= ‘=’ | ‘<’ | ‘<=’ | ‘>’ | ‘>=’ | ‘<>’ | LIKE

spatial_comparator ::= CONTAINS | WITHIN | INTERSECTS | NEIGHBOR | EQUAL | UNDER |
OVER | WEST_FROM | EAST_FROM | SOUTH_FROM | NORTH_FROM

distance_specification ::= number distance_measure

distance_measure ::= ‘mm’ | ‘cm’ | ‘m’ | ‘km’ | ‘in’ | ‘ft’ | ‘yd’ | ‘mi’ | ‘nmi’

threshold_specification ::= ATTRIBUTE attribute literal | CONFIDENCE literal |
SUPPORT literal | CLASSIFICATION literal | EXCEPTION literal

hierarchy_specification ::= USE HIERARCHY hier_name FOR attribute |
DYNAMICALLY ADJUST HIERARCHY hier_name |
GENERATE HIERARCHY hier_name FOR attribute

hierachy_definition ::= DEFINE ‘{’attribute {, attribute}‘}’ UNDER ‘{’attribute {, attribute}‘}’ IN hier_name
DEFINE ‘{’literal {, literal}‘}’ UNDER ‘{’literal‘}’ IN hier_name |
DEFINE SELECTION UNDER ‘{’ literal ‘}’ IN hier_name |
DEFINE ‘[’ number ‘...’ number ‘]’ ‘<’ ‘{’ literal ‘}’ IN hier_name |
DEFINE SPATIAL HIERARCHY hier_name BASED ON attribute |
INSERT literal UNDER literal IN hier_name |
DELETE literal UNDER literal IN hier_name |
INSERT SELECTION UNDER literal IN hier_name |
DELETE SELECTION UNDER literal IN hier_name |
DROP hier_name | CREATE hier_name

hierarchy-level_manipulation ::= ADD attribute | DROP attribute |
UP ON attribute | DOWN ON attribute

Bibliography

[1] D.J. Abel, B.C. Ooi, K.-L. Tan, R. Power, and J.X. Yu. Spatial Join Strategies in Distributed Spatial DBMS. In *Advances in Spatial Databases, Proc. of 4th Symp. SSD'95*, Springer-Verlag, Berlin, 1995, pp. 348–367.

[2] R. Agrawal, T. Imielinski, and A. Swami. MIning Association Rules Between Sets of Items in Large Databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data*, Washington, DC, 1993, pp. 207–216.

[3] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proc. 20th VLDB Conference*, Santiago, Chile, 1994, pp. 487–499.

[4] American National Standards Institute. *Database Language SQL*, ANSI X3.135-1992.

[5] S.S. Anand, D.A. Bell, and J.G. Hughes. A general framework for database mining based on evidential theory. *Internal Report*, Dept. of Inf. Sys., Univ. of Ulster at Jordanstown, 1993.

[6] L. Anselin. Local Indicators of Spatial Association - LISA. In *Geographical Analysis*, Vol. 27, No. 2, pp. 93-115, 1995.

[7] L. Anselin. Interactive Techniques and Exploratory Spatial Data Analysis. In *Geographical Information Systems: Principles, Techniques, Management and Applications*, eds., P. Longley, M. Goodchild, D. Maguire, and D. Rhind. Cambridge: Geoinformation Int. (in press).

[8] W.G. Aref and H. Samet. Extending DBMS with Spatial Operations. In *Proc. 2nd Symp. SSD'91*, Zurich, Switzerland, 1991, pp. 299–318.

[9] W.G. Aref and H. Samet. Optimization Strategies for Spatial Query Processing. In *Proc. 17th VLDB Conference*, Barcelona, Spain, 1991, pp. 81–90.

[10] W. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley, 1994.

[11] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method for Point and Rectangles. In *Proc. of 1990 to ACM-SIGMOD Intl. Conf. on Management of Data*, Atlantic City, USA, 1990, pp. 322–331.

[12] D.A. Bell, S.S. Anand, and C.M. Shapcott. Database Mining in Spatial Databases. In *Int. Workshop on Spatio-Temporal Databases*, 1994.

[13] J.L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. In *Communications of the ACM*, Vol. 18, pp. 509–517, 1975.

[14] J.L. Bentley and D. Wood. An Optimal Worst Case Algorithm for Reporting Intersections of Rectangles. In *IEEE Trans. on Computers*, Vol. C-29, No.7, pp. 571–577, 1980.

[15] M. Berman. Testing for Spatial Association between a Point Process and Another Stochastic Process. In *Applied Statistics*, Vol. 35, pp. 54–62, 1986.

[16] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Belmont, CA, Wadsworth, 1984.

[17] S. Brin, R. Motwani, and C. Silverstein. Beyond Market Basket: Generalizing Association Rules to Correlations. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, Tucson, AZ, 1997, pp. 265–276.

[18] T. Brinkhoff, H.-P. Kriegel, and B. Seeger. Efficient Processing of Spatial Joins Using R-trees. In *Proc. 1993 ACM-SIGMOD Conf. Management of Data*, Washington, DC, 1993, pp. 237–246.

[19] T. Brinkhoff, H.-P. Kriegel, R. Schneider, and B. Seeger. Multistep Processing of Spatial Joins. In *Proc. 1994 ACM-SIGMOD Conf. Management of Data*, Minneapolis, Minnesota, 1994, pp. 197–208.

[20] B.P. Buttenfield and R.B. McMaster, Eds. *Map Generalization: Making Rules for Knowledge Representation.* Longman Scientific & Technical, New York, NY, Wiley, 1991.

[21] D.D. Chamberllin and R.F.Boyce. SEQUEL: A Structured English Query Language. In *Proc. of 1974 ACM-SIGMOD Workshop on Data Description, Access and Control*, Ann Arbor, MI, 1974, Vol. 1, pp. 249–264.

[22] S. Chaudhuri and S. Dayal. An Overview of Data Warehousing and OLAP Technology. In *ACM SIGMOD Record*, Vol. 26, pp. 65–74, 1997.

[23] S. Chaudhuri. Data Mining and Database Systems: Where is the intersection? In *Bulletin of the Technical Committee on Data Engineering*, March 1998, Vol. 21, No. 1, pp. 4–8, 1998.

[24] M.S. Chen, J. Han, and P.S. Yu. Data Mining: An Overview from a Database Perspective. In *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, pp. 866–883, 1996.

[25] S. Cheng. Statistical Approaches to Predictive Modeling in Large Databases, M.Sc. Thesis, Computing Science, Simon Fraser University, March 1998.

[26] D.K.Y. Chiu, A.K.C. Wong, and B. Cheung. A Statistical Technique for Extracting Classificatory Knowledge from Databases. In [117], pp. 125–141.

[27] A.D. Cliff and J.K. Ord. *Spatial Autocorrelation.* London, Pion, 1973.

[28] W.W. Cohen. Learning Trees and Rules with Set-valued Features. In *Proc. of the 13th National Conf. on Artificial Intelligence (AAAI'96)*, Portland, OR, 1996, pp. 709–716.

[29] R.F. Cromp and W.J. Cambell. Data Mining of Multidimensional Remotely Sensed Images. In *Proc. of 2nd Int. Conf. on Information and Knowledge Management*, Arlington, VA, 1993, pp. 471–480.

[30] S. Dutta. Qualitative Spatial Reasoning: A Semi-quantitative Approach Using Fuzzy Logic. In *Proc. 1st Symp. SSD'89*, Santa Barbara, CA, 1989, pp. 345–364.

[31] S. Dutta. Topological Constraints: A Representational Framework for Approximate Spatial and Temporal Reasoning. In *Proc. 2nd Symp. SSD'91*, Zurich, Switzerland, 1991, pp. 161–182.

[32] M. Egenhofer. *Spatial Query Languages*. Ph.D. Thesis. University of Maine, 1989.

[33] M. Egenhofer. Reasoning about Binary Topological Relations. In *Proc. 2nd Symp. SSD'91*, Zurich, Switzerland, 1991, pp. 143–160.

[34] M. Egenhofer. Spatial SQL: A Query and Presentation Language. In *IEEE Trans. Knowledge and Data Engineering*, Vol. 6, pp. 86–95, 1994.

[35] M. Ester, H.-P. Kriegel, and X. Xu. Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification. In *Advances in Spatial Databases, Proc. of 4th Symp. SSD'95*, Springer-Verlag, Berlin, 1995, pp. 67–82.

[36] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases. In *Proc. of the Second Int. Conf. on Data Mining KDD-96*, Portland, Oregon, 1996, pp. 226–231.

[37] M. Ester, H.-P. Kriegel, and J. Sander. Spatial Data Mining: A Database Approach. In *Proc. Int. Symp. on Large Spatial Databases (SSD'97)*, Berlin, Germany, 1997, pp. 47–66.

[38] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. Density-Connected Sets and their Application for Trend Detection in Spatial Databases. In *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining*, Newport Beach, CA, 1997, pp. 10–15.

[39] M. Ester, A. Frommelt, H.-P. Kriegel, and J. Sander. Algorithms for Characterization and Trend Detection in Spatial Databases. In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, 1998, pp. 44–50.

[40] V. Estivill-Castro and A.T. Murray. Mining Spatial Data via Clustering. In *Proc. of 8th Symp. on Spatial Data Handling SDH'98*, Vancouver, Canada, 1998, pp. 522–532.

[41] C. Faloustos and L. King-Ip. FastMap: a Fast Algorithm for Indexing, Data-mining and Visualization of traditional and Multimedia Datasets. In *ACM SIGMOD Record* Vol. 24, No. 2, pp. 163–174, 1995.

[42] U.M. Fayyad and K.B. Irani. The Attribute Selection Problem in Decision Tree Generation. In *Proc. of the Tenth National Conf. on Artificial Intelligence AAAI-92*, Cambridge, MA, MIT Press, 1992, pp. 104–110.

[43] U. Fayyad, *et al.* Automated Analysis of a Large-Scale Sky Survey: The SKICAT System. In *Proc. 1993 Knowledge Discovery in Databases Workshop*, Washington, DC, 1993, pp. 1–13.

[44] U.M. Fayyad and P. Smyth. Image Database Exploration: Progress and Challenges. In *Proc. 1993 Knowledge Discovery in Databases Workshop*, Washington, DC, 1993, pp. 14–27.

[45] U.M. Fayyad, S.G. Djorgovski, and N. Weir. Automating the Analysis and Cataloging of Sky Surveys. In [46]

[46] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, Menlo Park, CA, 1996.

[47] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD Process for Extracting Useful Knowledge from Volumes of Data. In *Communications of the ACM*, Vol. 39, No. 11, pp. 27–32, 1996.

[48] D. Fisher. Improving Inference through Conceptual Clustering. In *Proc. 1987 AAAI Conf.*, Seattle, Washington, 1987, pp. 461–465.

[49] S. Fotheringham and P. Rogerson, Eds. *Spatial Analysis and GIS*, Taylor and Francis, 1994.

[50] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus. Knowledge Discovery in Databases: An Overview. In [117], pp. 1–27.

[51] Y. Fu and J. Han. Meta-rule-guided Mining of Association Rules in Relational Databases. In *Proc. 1st Intl. Workshop om Integration of Knowledge Discovery with Deductive and Onject-Oriented Databases (KDOD'95)*, Singapore, 1995, pp. 39–46.

[52] Y. Fu. *Discovery of Multiple-Level Rules from Large Databases.* Ph.D. Thesis. Simon Fraser University, 1996.

[53] V. Gaede and O. Günther. Multidimensional Access Methods. In *ACM Computing Surveys*, Vol. 30, No. 2, pp. 170–231, 1998.

[54] A. Getis and J.K. Ord. The Analysis of Spatial Association by Use of Distance Statistics. In *Geographical Analysis*, Vol. 24, pp. 189–206, 1992.

[55] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab and Sub-totals. In *Data Mining and Knowledge Discovery*, Vol. 1, pp. 29–54, 1997.

[56] D.A. Griffith. *Spatial Statistics: Past, Present, and Future.* Institute of Mathematical Geography, University of Michigan, 1990.

[57] J. Guan and D. Bell. *Evidence Theory and its Applications, vol. I.* North-Holland, 1991.

[58] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Seattle, WA, 1998, pp. 73–84.

[59] R.H. Güting. An Introduction to Spatial Database Systems. In *VLDB Journal*, Vol. 3, pp. 357–400, 1994.

[60] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Boston, MA, 1984, pp. 47–57.

[61] J. Han, Y. Cai, and N. Cercone. Data-driven Discovery of Quantitative Rules in Relational Databases. In *IEEE Trans. Knowledge and Data Engineering*, Vol. 5, pp. 29–40, 1993.

[62] J. Han and Y. Fu. Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Databases. In *Proc. AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94)*, Seattle, WA, 1994, pp. 157–168.

[63] J. Han and Y. Fu. Discovery of Multiple-level Association Rules from Large Databases. In *Proc. 1995 Int. Conf. Very Large Data Bases*, Zurich, Switzerland, 1995, pp. 420–431.

[64] J. Han and Y. Fu. Exploration of the Power of Attribute-Oriented Induction in Data Mining. In [46].

[65] J. Han, Y. Fu, K. Koperski, G. Melli, W. Wang, and O.R. Zaïane. Knowledge Mining in Databases: An Integration of Machine Learning Methodologies with Database Technologies. In *Canadian Artificial Intelligence*, Winter 1996, pp. 4–8.

[66] J. Han, Y. Fu, W. Wang, K. Koperski, and O.R. Zaïane. DMQL: A Data Mining Query Language for Relational Databases. In *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, QB, 1996, pp. 27–34.

[67] J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanović, B. Xia, and O.R. Zaïane. DBMiner: A System for Mining Knowledge in Large Relational Databases. In *Proc. 1996 Int'l Conf. on Data Mining and Knowledge Discovery (KDD'96)*, Portland, Oregon, 1996, pp. 250–255.

[68] J. Han, K. Koperski, and N. Stefanović. GeoMiner: A System Prototype for Spatial Data Mining. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, Tucson, Arizona, 1997, pp. 553–556.

[69] J. Han, S. Nishio, and H. Kawano. Knowledge Discovery in Object-Oriented and Active Databases. In F. Fuchi and T. Yokoi, Eds, *Knowledge Building and Knowledge Sharing*, Ohmsha/IOS Press, pp. 221–230, 1994.

[70] J. Han, N. Stefanović, and K. Koperski. Selective Materialization: An Efficient Method for Spatial Data Cube Construction. In *Proc. 1998 Pacific-Asia Conf. on Knowledge Discovery in Databases*, Melbourne, Australia, 1998, pp. 144–158.

[71] J. Han. Towards On-Line Analytical Mining in Large Databases. In *ACM SIGMOD Record*, Vol. 27, pp. 97–107, 1998.

[72] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing Data Cubes Efficiently. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, Montreal, Canada, 1996, pp. 205–216.

[73] M. Holsheimer and M. Kersten. Architectural Support for Data Mining. In *CWI Technical Report CS-R9429*, Amsterdam, The Netherlands, 1994.

[74] L. Hyafil and R.L. Rivest. Constructing Optimal Binary Decision Trees in NP-complete. In *Information Processing Letters*, Vol. 5, No. 1, pp. 15–17, 1976.

[75] T. Imielinski. From File Mining to Database Mining. In *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, Quebec, 1996, pp. 35–39.

[76] W.H. Inmon. *Building the Data Warehouse.* John Wiley, 1996.

[77] M. Kamber, J. Han, J.Y. Chiang. Metarule-Guided Mining of Multi-Dimensional Association Rules Using Data Cubes. In *Proceeding of the 3rd Int. Conf. on Knowledge Discovery and Data Mining*, Newport Beach, California, 1997, pp. 207–210.

[78] M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and Decision Tree Induction: Efficient Classification in Data Mining. In *Proc. of 1997 Int. Workshop Research Issues on Data Engineering (RIDE'97)*, Birmingham, England, 1997, pp. 111–120.

[79] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

[80] D. Keim and H.-P. Kriegel. Visualization Techniques for Mining Large Databases: A Comparison. In *IEEE Trans. Knowledge and Data Engineering*, Vol. 8, pp. 923–938, 1994.

[81] Z. Kemp and H.T.K. Lee. A Marine Environmental System for Spatiotemporal Analysis. In *Proc. of 8th Symp. on Spatial Data Handling SDH'98*, Vancouver, Canada, 1998, pp. 474–483.

[82] R. Kimball. *The Data Warehouse Toolkit.* John Willey & Sons, New York, 1996

[83] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding Interesting Rules from Large Sets of Discovered Association Rules. In *Proc. 3rd Int. Conf. Information and Knowledge Management*, Gaithersburg, Maryland, 1994, pp. 401–408.

[84] K. Kira and L.A. Rendell. The Feature Selection Problem: Traditional Methods and a New Algorithm. In *Proc. of the Tenth National Conf. on Artificial Intelligence AAAI-92*, Cambridge, MA, MIT Press, 1992, pp. 129–134.

[85] E.M. Knorr and R.T. Ng. Finding Aggregate Proximity Relationships and Commonalities in Spatial Data Mining. In *IEEE Trans. Knowledge and Data Engineering*, Vol. 8, pp. 884–897, 1996.

[86] E. Knorr and R.T. Ng. Extraction of Spatial Proximity Patterns by Concept Generalization. In *Proc. of the Second Int. Conf. on Data Mining KDD-96*, Portland, Oregon, 1996, pp. 347–350.

[87] E. Knorr and R.T. Ng. Algorithms for Mining Distance-based Outliers in Large Datasets. In *Proc. of the 24th VLDB Conference*, New York, New York, 1998, pp. 392–403.

[88] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Databases. In *Advances in Spatial Databases, Proc. of 4th Symp. SSD'95*, Springer-Verlag, Berlin, 1995, pp. 47–66.

[89] K. Koperski, J. Adhikary, and J. Han. Spatial Data Mining: Progress and Challenges. In *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, QB, June 1996, pp. 55–70.

[90] K. Koperski and J. Han. Data Mining Methods for the Analysis of Large Geographic Databases. In *Proc. of 10th Annual Conf. on GIS*, Vancouver, BC, March 1996 (CD-ROM).

[91] K. Koperski, J. Han, and J. Adhikary. Mining Knowledge in Geographical Data. In *Communications of ACM* (accepted).

[92] K. Koperski, J. Han, and N. Stefanović. An Efficient Two-Step Method for Classification of Spatial Data. In *Proc. of 8th Symp. on Spatial Data Handling SDH'98* Vancouver, Canada, 1998, pp. 45–54.

[93] K. Koperski, N. Stefanović, J. Han, and Q. Chen. Design and Implementation of Spatial Data Mining Engine: GeoMiner. (submitted for publication).

[94] B. Kuipers. Modeling Spatial Knowledge. In *Cognitive Science*, Vol. 2, pp. 129–153, 1978.

[95] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems.* Academic Press, London, 1995.

[96] A. Lea. GIS and the Promotion of Geographical Analysis in Business – A Pessimistic View from the Trenches. In *Proc. of Workshop on Status and Trends in Spatial Analysis*, Santa Barbara, CA, 1998.
http://www.ncgia.ucsb.edu/conf/sa_workshop/papers/ncgia.html

[97] H.W. Lotwick and B.W. Silverman. Methods for Analyzing Spatial Processes of Several Types of Points. In *Journal of the Royal Statistics Society B*, Vol. 44, pp. 406–413, 1982.

[98] W. Lu and J. Han. Distance-Associated Join Indexes for Spatial Range Search. In *Proc. of the Int. Conf. on Data Engineering*, 1992, pp. 284–292.

[99] W. Lu, J. Han, and B.C. Ooi. Discovery of General Knowledge in Large Spatial Databases. In *Proc. of Far East Workshop on Geographic Information Systems*, World Scientific, Singapore, 1993, pp. 275–289.

[100] D.J. Maguire, M.F. Goodchild, and D.W. Rhind. *Geographical Information Systems. Principles and Applications.* Longman Scientific & Technical, Burnt Mill, UK, 1991.

[101] J. Major, and J. Mangano. Selecting among Rules Induced from a Hurricane Database. In *Proc. 1993 Knowledge Discovery in Databases Workshop*, AAAI Press, Menlo Park, CA, 1993, pp. 28–47.

[102] MapInfo Corporation. MapInfo Professional Home Page.
http://www.mapinfo.com/events/mipro/mipro.html

[103] C.J. Matheus, P.K. Chan, and G. Piatetsky-Shapiro. Systems for Knowledge Discovery in Databases. In *IEEE Trans. Knowledge and Data Engineering*, Vol. 5, pp. 903–913, 1993.

[104] R.S. Michalski, J.M. Carbonnel, and T.M. Mitchell, editors. *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann, Los Altos, CA, 1983.

[105] Microsoft Corporation. *OLE DB for OLAP Design Specification*, Beta 3, October 1997.

[106] J. Mingers. An Empirical Comparison of Selection Methods for Decision-Tree Induction. In *Machine Learning*, Vol. 4, pp. 261–284, 1989.

[107] T.M. Mitchell. Generalization as Search. In *Artificial Intelligence*, Vol. 18, pp. 203–226, 1982.

[108] L. Mohan and R.L. Kashyap. An Object-Oriented Knowledge Representation for Spatial Information. In *IEEE Transactions on Software Engineering*, Vol. 5, pp. 675–681, 1988.

[109] R.T. Ng and J. Han. Efficient and Effective Clustering Method for Spatial Data Mining. In *Proc. of 1994 Int. Conf. Very Large Data Bases*, Morgan Kaufmann, San Francisco, CA, 1994, pp. 144–155.

[110] R.T. Ng, L.V.S. Lakshmanan, J. Han and A. Pang. Exploratory Mining and Pruning Optimizations of Constrained Associations Rules, In *Proc. of 1998 ACM-SIGMOD Conf. on Management of Data*, Seattle, WA, 1998, pp. 13–23.

[111] R.T. Ng and Y. Yu. Discovering Strong, Common and Discriminating Characteristics of Clusters from Thematic Maps. In *Proc. of the 11th Annual Symp. on Geographic Information Systems*, 1997, pp. 392–394.

[112] The OLAP Council. *MDAPI^{TM} The OLAP Application Program Interface Version 2.0 Specification.*

[113] F. Olken and D. Rotem. Sampling from Spatial Databases In *Proc. of 9th Int. Conf. on Data Engineering ICDE'93*, Los Alamitos, CA, USA, IEEE Computer Society Press, 1993, pp. 199–208

[114] P. van Oosterom and J. van den Bos. An Object-oriented Approach to the Design of Geographic Information System. In *Proc. 1st Symp. SSD'89*, Santa Barbara, CA, 1989, pp. 255–269.

[115] S. Openshaw. Developing Appropriate Spatial Analysis Methods for GIS. In [100]

[116] K. Peterson. A Trade Area Primer. In *Business Geographics*, Vol. 5, No. 9, pp. 18–21, 1997.

[117] G. Piatetsky-Shapiro and W.J. Frawley. *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991.

[118] G. Piatesky-Shapiro and C.J. Matheus. The Interestingness of Deviations. In *Proc. 1994 Workshop on Knowledge Discovery in Databases*, Seattle, WA, pp. 25–36.

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

[120] J.R. Quinlan. Learning Efficient Classification Procedures and their Application to Chess Endgame. In [104].

[121] J.R. Quinlan. Induction of Decision Trees. In *Machine Learning*, Vol. 1, pp. 81–106, 1986.

[122] J.R. Quinlan. Decision Trees and Decisionmaking. In *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 20, No. 2, pp. 339–346, 1990.

[123] D. Rotem. Spatial Join Indices. In *Proc. 7th Int. conf. on Data Engineering*, Kobe, Japan, 1991, pp. 500–509.

[124] S.R. Safavian and D. Landgrebe. A Survey of Decision Tree Classifier Technology. In *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 21, No. 3, pp. 660–674, 1991.

[125] H. Samet. *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1990.

[126] E. Schikuta. Grid Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets. In *Proc. 13th Int. Conf. on Pattern Recognition*, Vienna, IEEE Computer Society Press, 1996, pp. 101–105.

[127] E. Schikuta and M. Erhart. The BANG-Clustering System: Grid-Based Data Analysis. In *Proc. of the Second Int. Symp. on Intelligent Data Analysis (IDA-97)*, London, Springer Verlag, August 1997, pp. 513–524.

[128] G. Shaw and D. Wheeler. *Statistical Techniques in Geographical Analysis.* London, David Fulton, 1994.

[129] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In Proc. of the 24th VLDB Conference, New York, NY, 1998, pp. 428–439.

[130] E.C. Shek, R.R. Muntz, E. Mesrobian, and K. Ng. Scalable Exploratory Data Mining of Distributed Geoscientific Data. In *Proc. of the Second Int. Conf. on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, 1996, pp. 32–37.

[131] W. Shen, K. Ong, B. Mitbander, and C. Zaniolo. Metaqueries for Data Mining. In [46].

[132] P. Smyth, M.C. Burl, U.M. Fayyad, and P. Perona. Knowledge Discovery in Large Image Databases: Dealing with Uncertainties in Ground Truth. In *Proc. of AAAI-94 workshop on KDD*, Seattle, WA, 1994, pp. 109–120.

[133] R. Srikant and R. Agrawal. Mining Generalized Association Rules. In *Proc. 1995 Int. Conf. Very Large Data Bases*, Zurich, Switzerland, 1995, pp. 407–419.

[134] N. Stefanović, J. Han, and K. Koperski. Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes. In *IEEE Transaction on Knowledge and Data Engineering.* (accepted)

[135] P. Stolorz, H. Nakamura, E. Mesrobian, R.R. Muntz, E.C. Shek, J.R. Santos, J. Yi, K. Ng, S.-Y. Chien, C.R. Mechoso, and J.D. Farrara. Fast Spatio-Temporal Data Mining of Large Geophysical Datasets. In *Proc. of the First Int. Conf. on Data Mining KDD-95*, Montreal, Canada, 1995, pp. 300–305.

[136] P. Stolorz and C. Dean. Quakefinder: A Scalable Data Mining System for Detecting Earthquakes from Space. In *Proc. of the Second Int. Conf. on Data Mining KDD-96*, Portland, Oregon, 1996, pp. 208–213.

[137] M. Stonebraker. *Readings in Database Systems, 2ed.* Morgan Kaufmann, 1993.

[138] D. Tjøstheim. A Measure of Association for Spatial Variables. In *Biometrika*, Vol. 65, pp. 109–114, 1978.

[139] C.D. Tomlin. *Geographic Information Systems and Cartographic Modeling.* Prentice-Hall, New York, 1990.

[140] D. Tsur, J.D. Ullman, S. Abitboul, C. Clifton, R. Motwani, and S. Nestorov. Query Flocks: A Generalization of Association-Rule Mining In *Proc. of 1998 ACM-SIGMOD Conf. on Management of Data*, Seattle, Washington, 1998, pp. 1–12.

[141] U.S. Census Bureau. Tiger/Line Documentation. http://www.census.gov/geo/www/tiger/tl95doc.html

[142] W. Wang, J. Yang, and R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. In *Proc. 23rd Int. Conf. on Very Large Data Bases*, Athens, Greece, 1997, pp. 186–195.

[143] D. Wartenberg. Exploratory Spatial Analyses: Outliers, Leverage Points, and Influence Functions. In [56], pp. 133–162, 1990.

[144] D. Wetteschereck, D.W. Aha, and T. Mohri. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. In *Artificial Intelligence Review*, Vol. 10, pp. 1–37, 1997.

[145] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases. In *Proc. 14th Int. Conf. on Data Engineering (ICDE'98)*, Orlando, FL, 1998, pp. 324–331.

[146] O.R. Zaïane. *Resource and Knowledge Discovery from the Internet and Multimedia Repositories*. Ph.D. Thesis. Simon Fraser University, 1999.

[147] K. Zeitouni. University of Versailles. Private communication.

[148] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proc. of ACM-SIGMOD Int. Conf. on Management of Data*, New York, NY, 1996, pp. 103–114.

Of making many books there is no end
Ecclesiastes 12:12