A Data Mining Query Language for Knowledge Discovery in a Geographical Information System

Donato Malerba Annalisa Appice Michelangelo Ceci

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona 4, 70125 Bari, Italy
{malerba, appice, ceci}@di.uniba.it

Abstract. Spatial data mining is a process used to discover interesting but not explicitly available, highly usable patterns embedded in both spatial and non-spatial data, which are possibly stored in a spatial database. An important application of spatial data mining methods is the extraction of knowledge from a Geographic Information System (GIS). INGENS (INductive GEographic iNformation System) is a prototype GIS which integrates data mining tools to assist users in their task of topographic map interpretation. The spatial data mining process is aimed at a user who controls the parameters of the process by means of a query written in a mining query language. In this paper, we present SDMOQL (Spatial Data Mining Object Query Language), a spatial data mining query language used in INGENS, whose design is based on the standard OQL (Object Query Language). Currently, SDMOQL supports two data mining tasks: inducing classification rules and discovering association rules. For both tasks the language permits the specification of the task-relevant data, the kind of knowledge to be mined, the background knowledge and the hierarchies, the interestingness measures and the visualization for discovered patterns. Some constraints on the query language are identified by the particular mining task. The syntax of the query language is described and the application to a real repository of maps is briefly reported.

1. Introduction

Spatial data are important in many applications, such as computer-aided design, image processing, VLSI, and GIS. This steady growth of spatial data is outpacing the human ability to interpret them. There is a pressing need for new techniques and tools to find implicit regularities hidden in the spatial data.

Advances in spatial data structures [7], spatial reasoning [3], and computational geometry [22] have paved the way for the study of knowledge discovery in spatial data, and, more specifically, in geo-referenced data. *Spatial data mining* methods have been proposed for *the extraction of implicit knowledge, spatial relations, or other patterns not explicitly stored in spatial databases* [15]. Generally speaking, a *spatial pattern* is a pattern showing the interaction between two or more spatial objects or space-dependent attributes, according to a particular spacing or set of arrangements [1].

Knowledge discovered from spatial data may include classification rules, which describe the partition of the database into a given set of classes [14], clusters of spatial objects ([11], [24]), patterns describing spatial trends, that is, regular changes of one or more non-spatial attributes when moving away from a given start object [6], and subgroup patterns, which identify subgroups of spatial objects with an unusual, an unexpected, or a deviating distribution of a target variable [13]. The problem of mining spatial association rules has been tackled by [14], who implemented the module Geo-associator of the spatial data mining system GeoMiner [9].

A database perspective on spatial data mining is given in the work by Ester *et al.* [6], who define a small set of database primitives for the manipulation of neighbourhood graphs and paths used in some spatial data mining systems. An Inductive Logic Programming (ILP) perspective on spatial data mining is reported in [21], which proposes a logical framework for spatial association rule mining.

GIS offers an important application area where spatial data mining techniques can be effectively used. In the work by Malerba *et al.* [20], it can be seen how some classification patterns, induced from georeferenced data, can be used in topographic map interpretation tasks. A prototype of GIS, named INGENS [19], has been built around this application. In INGENS the geographical data collection is organized according to an object-oriented data model and is stored in a commercial Object Oriented DBMS (ODBMS).

INGENS data mining facilities support sophisticated end users in their topographic map interpretation tasks. In INGENS, each time a user wants to query its database on some geographical objects not explicitly modelled, he/she can prospectively train the system to recognize such objects and to create a special user view. Training is based on a set of examples and counterexamples of geographical concepts of interest to the user (e.g., ravine or steep slopes). Such concepts are not explicitly modelled in the map legends, so they cannot be retrieved by simple queries. Furthermore, the user has serious difficulty formalizing their operational definitions. Therefore, it is necessary to rely on the support of a knowledge discovery system that generates some plausible "definitions". The sophisticated user is simply asked to provide a set of (counter-) examples (e.g., map cells) and a number of parameters that define the data mining task more precisely.

An INGENS user should not have problems, due to the integration of different technologies, such as data mining, OODBMS, and GIS. In general, to solve any such problems the use of *data mining languages* has been proposed, which interface users with the whole system and hide the different technologies [10]. However, the problem of designing a spatial mining language has received little attention in the literature. To our knowledge, the only spatial mining language is GMQL (Geo Mining Query Language) [16], which is based on DMQL (Data Mining Query Language) [8]. These languages have both been developed for mining knowledge from *relational* databases, so SQL remains the milestone on which their syntax and semantics are built.

This paper presents SDMOQL (Spatial Data Mining Object Query Language) a spatial mining query language for INGENS sophisticated users. Its main characteristics are the following:

- It is based on OQL, the standard defined by ODMG (Object Database Management Group) for designing object oriented models (www.odmg.org).

– It interfaces relational data mining systems [2] that work with first-order representations of input data and output patterns.
– It separates the logical representation of spatial objects from their physical or geometrical representation.

The paper is organized as follows. INGENS architecture and conceptual database schema are described in the next section, while in Section 3 the spatial data mining process in INGENS is introduced. In Section 4 the syntax of SDMOQL is presented, while in Section 5 complete example of SDMOQL's use in INGENS is described. Finally, related works are discussed in Section 6.

2. INGENS architecture and conceptual database schema

The three-layered architecture of INGENS is illustrated in Fig. 1. The interface layer implements a *Graphical User Interface* (GUI), a java applet which allows the system to be accessed by the following four categories of users:

– Administrators, who are responsible for GIS management.
– Map *maintenance* users, whose main task is updating the Map Repository.
– Sophisticated end users, who can ask the system to learn operational definitions of geographical *objects* not explicitly modelled in the database.
– Casual end users, who occasionally access the database and may need different information *each* time. Casual users cannot train INGENS.

Only sophisticated end-users are allowed to discover new patterns by using SDMOQL.

The application enablers layer makes several facilities available to the four categories of INGENS users. In particular, the *Map Descriptor* is the application enabler responsible for the automated generation of first-order logic descriptions of some geographical objects. Descriptors generated by a Map Descriptor are called *operational*. The *Data Mining Server* provides a suite of data mining systems that can be run concurrently by multiple users to discover previously unknown, useful patterns in geographical data. In particular, the Data Mining Server provides sophisticated

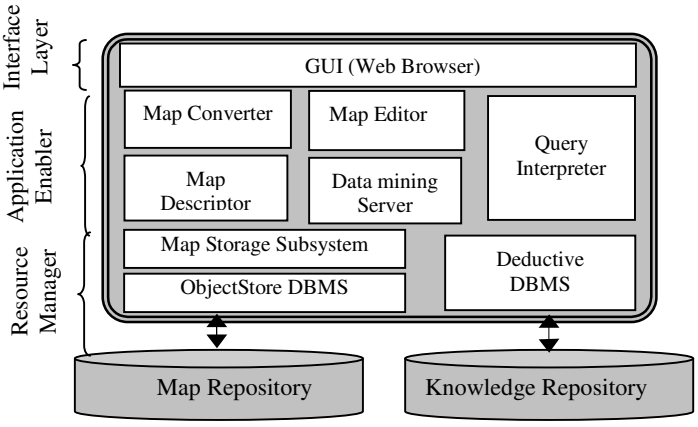


Fig. 1. INGENS three-layered software architecture

users with an inductive learning system, named ATRE [18], which can generate models of geographical objects from a set of training examples and counter-examples. The *Query Interpreter* allows any user to formulate a query in SDMOQL. The query can refer to a specific map and can contain both predefined predicates and predicates, whose operational definition has already been learned. Therefore, it is the Query Interpreter's responsibility to select the objects involved from the Map Repository, to ask the Map Descriptor to generate their logical descriptions and to invoke the inference engine of the Deductive Database, in order to check conditions expressed by both predefined and learned predicates. The *Map Converter* is a suite of tools which supports the acquisition of maps from external sources. Currently, INGENS can export maps in Drawing Interchange Format (DXF) by Autodesk Inc. (www.autodesk.com) and can automatically acquire information from vectorized maps in the MAP87 format, defined by the Italian Military Geographic Institute (IGMI) (www.nettuno.it/fiera/igmi/igmit.htm). Since IGMI's maps contain static information on orographic, hydrographic and administrative *boundaries* alone, a *Map Editor* is required to integrate and/or modify this information.

The resource layer controls the access to both the *Knowledge Repository* and the *Map Repository*. The former contains the operational definitions of geographical objects induced by the Data Mining Server. In INGENS, different users can have different definitions of the same geographical object. Knowledge is expressed according to a relational representation paradigm and managed by an XSB-based deductive relational DBMS [23]. The *Map Repository* is the database instance that contains the actual collection of maps stored in the GIS. Geographic data are organized according to an object-oriented data model. The object-oriented DBMS used to store data is a commercial one (ObjectStore 5.0 by Object Design, Inc.), so that full use is made of a well-developed, technologically mature non-spatial DBMS. Moreover, an object-oriented technology facilitates the extension of the DBMS to accommodate management of geographical objects. The *Map Storage Subsystem* is involved in storing, updating and retrieving items in and from the map collection. As a *resource manager*, it represents the only access path to the data contained in the Map Repository and which are accessed by multiple, concurrent clients.

Each map is stored according to a hybrid tessellation – topological model. The tessellation model follows the usual topographic practice of superimposing a regular grid on a map, in order to simplify the localization process. Indeed, each map in the repository is divided into square cells of the same size.

In the topological model of each cell it is possible to distinguish two different structural hierarchies: *physical* and *logical*. The physical hierarchy describes the geographical objects by means of the most appropriate physical entity, that is: point, line or region. The logical hierarchy expresses the semantics of geographical objects, independent of their physical representation. In the Map Repository, the logical hierarchy is represented by eight distinct classes of the database schema, each of which correspond to a geographic layer in a topographic map, namely hydrography, orography, land administration, vegetation, administrative (or political) boundary, ground transportation network, construction and built-up area. Objects of a layer are instances of more specific classes to which it is possible to associate a unique physical representation. For instance, the administrative boundary layer describes objects of one of the following subclasses: city, province, county or state.

Finally, each geographical object in the map has both a *physical structure* and a *logical structure*. The former is concerned with the representation of the object on some media by means of a point, a line or a region. Therefore, the physical structure of a cell associates the content of a cell with the physical hierarchy. On the other hand, the logical structure of a cell associates the content with the logical hierarchy, such as river, city, and so on. The logical structure is related to the map legend.

3. Spatial Data Mining process in INGENS

The spatial data mining process in INGENS (see Fig. 2) is aimed at a user who controls the parameters of the process. Initially, the query written in SDMOQL is syntactically and semantically analyzed. Then the Map Descriptor generates a highly conceptual qualitative representation of the raw data stored in the object-oriented database (see Fig. 3). This representation is a conjunctive formula in a first-order logic language, whose atoms have the following syntax: $f(t_1,\ldots,t_n) = value$, where f is a function symbol called *descriptor*, t_i are terms and the *value* is taken from the range of f. A set of descriptors used in INGENS is reported in Table 1. They can be roughly classified in *spatial* and *non-spatial*.

According to their *nature*, it is possible to spatial descriptors as follows:

- *Geometrical*, if they depend on the computation of some metric/distance. Their domain is typically numeric. Examples are *line_shape* and *extension*.
- *Topological*, if they are relations that are invariant under the topological transformations (translation, rotation, and scaling). The type of their domain is nominal. Examples are *region_to_region* and *point_to_region*.
- *Directional*, if they concern orientation. The type of their domain can be either numerical or nominal. An example is *geographic_direction*.
- *Locational*, if they concern the location of objects. Locations are represented by numeric values that express co-ordinates. There are no examples of locational descriptors in Table 1.

Some spatial descriptors are *hybrid*, in the sense that they merge properties of two or more categories above. For instance, the descriptor *line_to_line* that expresses conditions of parallelism and perpendicularity is both topological (it is invariant with respect to translation, rotation and scaling) and geometrical (it is based on the angle of incidence).

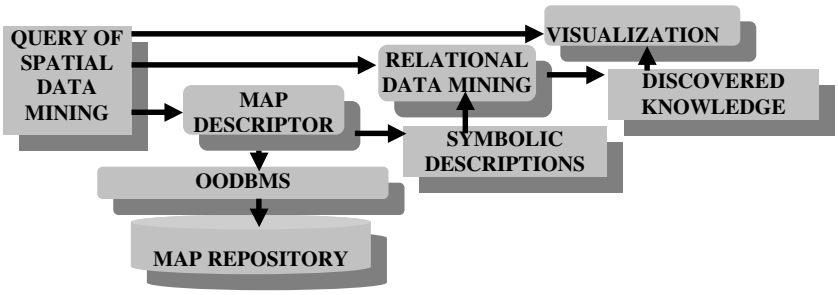
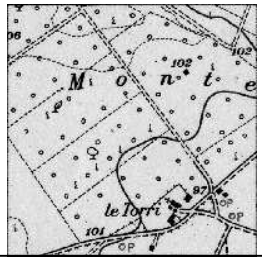
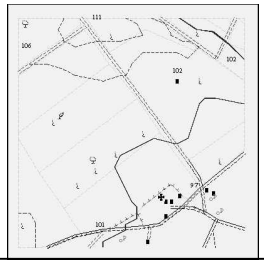


Fig. 2. Spatial data mining process in INGENS.





contain(c11,pc494_11)=true, ..., contain(c11,ss296_11)=true, contain(c11,qu61_11)=true,
type_of(pc494_11)=parcel, ..., type_of(ss296_11)=street, type_of(qu61_11)=quote,
subtype_of(pc494_11)=cultivation, ..., subtype_of(ss296_11)=cart_road,
subtype_of(ss296_11)=cart_road,
color(pc494_11)=black, ..., color(ss296_11)=black, color(qu61_11)=black,
part_of(pc494_11,x1)=true, ..., part_of(ss296_11,x68)=true, part_of(qu61_11,x69)=true,
altitude(x8)=97.00, ..., altitude(x69)=102.00,
extension(x3)=101, ..., extension(x68)=45.00,
geographic_direction(x3)=north, ..., geographic_direction(x68)=east,
line_shape(x3)=straight, ..., line_shape(x68)=straight,
area(x1)=99962,..., area(x66)=116662,
density(x1)=medium, ..., density(x66)=high,
distance(x3,x68)=80.00, ..., distance(x62,x68)=87.00,
line_to_line(x3,x68)=almost_parallel, ..., line_to_line(x62,x68)=almost_parallel,
region_to_region(x1,x9)=disjoint, ..., region_to_region(x45,x66)=disjoint,
line_to_region(x16,x1)=adjacent, ..., line_to_region(x67,x66)=intersect,
point_to_region(x2,x1)=outside, ..., point_to_region(x69,x66)=outside

Fig. 3. Raster and vector representation (above) and symbolic description of cell 11 (below). The cell is an example of a territory where there is a system of farms. The cell is extracted from a topographic chart (Canosa di Puglia 176 IV SW - Series M891) produced by the Italian Geographic Military Institute (IGMI) at scale 1:25,000 and stored in INGENS.

In INGENS geo-referenced objects can also be described by three non-spatial descriptors are *color*, *type_of* and *subtype_of*. Finally, the descriptor *part_of* associates the physical structure to a logical object. For instance, in the description:

type_of(s1)=street , part_of(s1,x1)=true, part_of(s1,x2)=true

the constant *s1* denotes a street which is physically represented by two lines, which are referred to as constants *x1* and *x2*.

The operational semantics of the descriptors is based on a set of methods defined in the object-oriented model of the map repository. More details on the computational methods for their extraction are reported in [17].

This qualitative data representation can be easily translated into Datalog with built-in predicates [18]. Thanks to this transformation, it is possible to use the output of the Map Descriptor module in many relational data mining algorithms, which return spatial patterns expressed in a first-order language. Finally, the results of the mining process are presented to the user. The graphical feedback is very important in the analysis of the results.

Table 1. Set of descriptors extracted by the Map Descriptor module in INGENS.

Feature	Meaning	Type	Domain	
			Type	Values
contain(C,L)	Cell C contains a logical object L	Topological relation	Boolean	{true, false}
part_of(L,F)	Logical object L is composed by physical object F	Topological relation	Boolean	{true, false}
type_of(L)	Type of L	Non-spatial attribute	Nominal	33 nominal values
subtype_of(L)	Specialization of the type of L	Non-spatial attribute	Nominal	101 nominal values specializing the type_of domain
color(L)	Color of L	Non-spatial attribute	Nominal	{blue, brown, black}
area(F)	Area of F	Geometrical attribute	Linear	[0..MAX_AREA]
density(F)	Density of F	Geometrical attribute	Ordinal	Symbolic names chosen by an expert
extension(F)	Extension of F	Geometrical attribute	Linear	[0..MAX_EXT]
geographic_direction(F)	Geographic direction of F	Directional attribute	Nominal	{north, east, north_west, north_east}
line_shape(F)	Shape of the linear object F	Geometrical attribute	Nominal	{straight, cuspidal, curvilinear }
altitude(F)	Altitude of F	Geometrical attribute	Linear	[0.. MAX_ALT]
line_to_line(F_1,F_2)	Spatial relation between two lines F_1 and F_2	Hybrid relation	Nominal	{almost parallel, almost perpendicular}
distance(F_1,F_2)	Distance between two lines F_1 and F_2	Geometrical relation	Linear	[0..MAX_DIST]
region_to_region(F_1,F_2)	Spatial relation between two regions F_1 and F_2	Topological relation	Nominal	{disjoint, meet, overlap, covers, contains, equal, covered_by, inside}
line_to_region(F_1,F_2)	Spatial relation between a line F_1 and a region F_2	Hybrid relation	Nominal	{along_edge, intersect}
point_to_region(F_1, F_2)	Spatial relation between a point F_1 and a region F_2	Topological relation	Nominal	{inside, outside, on_boundary, on_vertex}

4. Design of a data mining language for INGENS

SDMOQL is designed to support the interactive data mining process in INGENS. Designing a comprehensive data mining language is a challenging problem because data mining covers a wide spectrum of tasks, from data classification to mining association rules. The design of an effective data mining query language requires a deep understanding of the power, limitation and underlying mechanisms of the various kinds of data mining tasks. A data mining query language must incorporate a set of data mining primitives designed to facilitate efficient, fruitful knowledge discovery. Seven primitives have been considered as guidelines for the design of SDMOQL. They are:

1. the set of objects relevant to a data mining task,
2. the kind of knowledge to be mined,
3. the set of descriptors to be extracted from a digital map
4. the set of descriptors to be used for pattern description
5. the background knowledge to be used in the discovery process,
6. the concept hierarchies,
7. the interestingness measures and thresholds for pattern evaluation, and
8. the expected representation for visualizing the discovered patterns.

These primitives correspond directly to as many non-terminal symbols of the definition of an SDMOQL statement, according to an extended BNF grammar. Indeed, the SDMOQL top-level syntax is the following:

<SDMOQL> ::= <SDMOQL_Statement>; { <SDMOQL_Statement>;}
<SDMOQL_Statement> ::=<Spatial_Data_Mining_Statement>
| <Background_Knowledge>
| <Hierarchy>
| <Result_Displaying>
<Spatial_Data_Mining_Statement> ::= <Object_Specification_Query>
***mine** <Kind_of_Pattern>*
***analyze** <Primitive_descriptors>*
***with descriptors** <Pattern_descriptors>*
[<Background_Knowledge>]
{<Hierarchy>}
*[**with** <Interestingness_Measures>]*
[<Result_Displaying>]

where "[]" represents 0 or one occurrence and "{}" represents 0 or more occurrences, and words in bold type represent keywords. In sections 4.2 to 4.8 the detailed syntax for each data mining primitive is both formally specified and explained through various examples of possible mining problems.

4.1 Data specification: general principles

The first step in defining a data mining task is the specification of the data on which mining is to be performed. Data mining query languages presented in the literature

allows the user to specify, through an SQL query, a single data table, where each row represents one *unit of analysis*[1] while each column corresponds to a variable (i.e. an attribute) of the unit of analysis. Generally, no interaction between units of analysis is assumed.

The situation is more complex in spatial data mining. First, units of analysis are spatial objects, which means that they are characterized, among other things, by spatial properties.

Second, attributes of some spatial objects in the neighborhood of, or contained in, a unit of analysis may affect the values taken by attributes of the unit of analysis. Therefore, we need to distinguish units of analysis, which are the *reference* objects of an analysis, from other *task-relevant* spatial objects, and we need to represent interactions between them. In this context, the single table representation supported by traditional data mining query languages is totally inadequate, since different geographical objects may have different properties, which can be properly modelled by as many data tables as the number of object types.

Third, in traditional data mining relatively simple transformations are required to obtain units of analysis from the *units of observation*[2] explicitly stored in the database. The unit of observation is often the same as the unit of analysis, in which case no trasformation at all is required. On the contrary, in GIS research, the wealth of secondary data sources creates opportunities to conduct analyses with data from multiple units of observation. For instance, a major national study uses a form that collects information about each person in a dwelling and information about the housing structure, hence it collects data for two units of observation: persons and housing structures. From these data, different units of analysis may be constructed: household could be examined as a unit of analysis by combining data from people living in the same dwelling or family could be treated as the unit of analysis by combining data from all members in a dwelling sharing a familial relationship. Units of analysis can be constructed from units of observation by making explicit the spatial relations such as topological, distance and direction relations, which are implicitly defined by the location and the extension of spatial objects.

Fourth, working at the level of stored data, that is, geometrical representations (points, lines and regions) of geographical objects, is often undesirable. The GIS user is interested in working at higher conceptual levels, where human-interpretable properties and relations between geographical objects are expressed. A typical example is represented by the possible relations between two roads, which either cross each other, or run parallel, or can be confluent, independently of the fact that they are physically represented as “lines” or “regions” in a map.

To solve these problems, in SDMOQL the specification of the *geographical objects* (both reference and task-relevant) of interest for the data mining task (first primitive) is separated from the description of the units of analysis (third and fourth primitives). Each unit of analysis is described by means of both (non-)spatial properties and spatial relations between selected objects. First-order logic is adopted

---

[1] In statistics, the unit of analysis is the basic entity or object about which generalizations are to be made based on an analysis and for which data are collected in the form of variables.

[2] The unit of observation is the entity in primary research that is observed and about which information is systematically collected.

as representation language, since it overcomes the limitations of the single table representation. Some basic descriptors (see Table 1) are generated by the Map Descriptor, to support complex transformations of the data stored in the database into descriptions of units of analysis. Their specification is given in the third primitive. However, these descriptors refer to the physical representation of geographical objects of interest. To produce high-level conceptual descriptions involving objects of the logical hierarchy, the user can specify a set of new descriptors on the basis of those extracted by the Map descriptors. New descriptors are specified in the background knowledge (fifth primitive) by means of logic programs. Moreover, it is possible to specify that final patterns should be described by means of these new descriptors in the fourth primitive.

4.2 Task-relevant object specification

The selection of geographical objects is performed by means of simplified OQL queries with a SELECT-FROM-WHERE structure, namely:

<Data_Specification_Query> ::= <Query_Statement>
*{ **UNION** <Query_Statement>}*

<Query_Statement>::= ***SELECT*** *<Object> {, <Object>}*
FROM *<Class> {, <Class>}*
[WHERE <Conditions>]

The SELECT clause should return objects of a class in the database schema corresponding to a cell, a layer or a type of logical object. Therefore, the selection of object properties such as the attribute *river_name* of a river, is not permitted. Moreover, the selected objects must belong to the same symbolic level *(cell, layer or logic object)*. More formally the FROM clause can contain either a group of *Cells* or a set of *Layers*, or a set of *Logic Objects*, but never a mixture of them. Whenever the generation of the descriptions of objects belonging to different symbolic levels is necessary, the user can obtain it by means of the UNION operator. The following are examples of valid data queries:

Example 1:Cell-level query. The user selects *cell* 26 from the topographic map of Canosa (Apulia) and the Map Descriptor generates the description of all the objects in this cell.

SELECT x
FROM x in **Cell**
WHERE x->**num_cell** = 26 AND x->part_map->**map_name** = “Canosa”

Example 2: Layer-level query. The user selects the *Orography layer* from the topographic map of Canosa and the *Construction layer* from any map. The Map Descriptor generates the description of the objects in these layers.

SELECT x, y
FROM x in **Horography**, y in **Construction**
WHERE x->part_map->**map_name** = “Canosa”

Example 3: Object-level query. The user selects the *objects* of the logic class River and the objects of type motorway (instances of the class Road), from cell 26 of the topographic map of Canosa. The Map Descriptor generates the description of these objects.

SELECT x, y
FROM x in **River**, y in **Road**
WHERE x->part_map->**map_name** = "Canosa"
AND y->part_map->**map_name** = "Canosa"
AND x->log_incell->**num_cell** = 26 AND y->log_incell->**num_cell** = 26
AND y->**type_road** = "motorway"

The above queries do not present semantic problems. However, the next example is an OQL query which is syntactically correct but selects data that cannot be a valid input to the Map Descriptor.

Example 4: Semantically ambiguous query.

SELECT x, y
FROM x in **Cell**, y in **River**
WHERE x->**num_cell** = 26 AND y->log_incell->**num_cell** = 26

This query selects the object cell 26 and all rivers in it. However, it is unclear whether the Map Descriptor should describe the entire cell 26 or only the rivers in it, or both. In the first case, a cell-level query must be formulated (see example 1). In the second case, an object-level query produces the desired results (see example 3). In the (unusual) case that both kinds of descriptions have to be generated, the problem can be solved by the UNION operator, applied to the cell-level query and the object-level query. Therefore, the following constraint is imposed on SDMOQL: the selected data must belong to the same symbolic level (cell, layer or logic object). More formally the FROM clause can contain either a group of Cells or a set of Layers, or a set of Logic Objects, but never a mixture of them.

The next example is useful to present the constraints imposed on the SELECT clause.

Example 5: Attributes in the SELECT clause.

SELECT x.**name_river**
FROM x in **River**

The query selects the names of all the rivers stored in the database. The result set contains attributes and not geographic objects to be described by a set of attributes and relations. In order to select proper input data for the Map Descriptor, the SELECT clause should return objects of a class in the database schema corresponding to a cell, a layer or a type of logical object. It might be observed that the presence of an attribute in the SELECT clause can be justified when its type corresponds to a class. For instance, the following query:

SELECT x->**River**
FROM x in **Cell**
WHERE x->**num_cell** = 26

concerns all rivers in cell 26. Nevertheless, thanks to inverse relations (inverse members) characterizing an object model, it is possible to reformulate it as follows:

SELECT x
FROM x in **River**
WHERE x->log_incell->**num_cell** = 26

In this way, all the above constraints should be respected.

4.3 The kind of knowledge to be mined

The kind of knowledge to be mined determines the data mining task in hand. For instance, classification rules or decision trees are used in classification tasks, while association rules or complex correlation coefficients are extracted in association tasks. Currently, SDMOQL supports the generation of either classification rules[3] or association rules, which means that only two different mining problems can be solved in INGENS: the former has a predictive nature, while the latter is descriptive. The top-level syntax is defined below:

<*Kind_of_Pattern*> ::= <*Classification_Rules*> | <*Association_Rules*>

The non-terminal <*Classification_Rules*> specifies that patterns to be mined concern a classification task

<*Classification_Rules*> ::= ***classification as*** <*Pattern_Name*>
for <*Classification_Concept*> *{,* <*Classification_Concept*> *}*

In a classification task, the user may be interested in inducing a set of classification rules for a *subset* of the classes (or concepts) to which training examples belong. Typically, the user specifies both "positive" and "negative" examples, that is, he/she specifies examples of two different classes, but he/she is interested in classification rules for the "positive" class alone. In this case, the subset of interest for the user is specified in the <*Classification_Concept*> list.

In SDMOQL, spatial association rule mining tasks are specified as follows:

<*Association_Rules*> ::= ***association as*** <*Pattern_Name*>
key is <*Descriptor*>

As pointed out, spatial association rules define spatial patterns involving both *reference objects* and *task-relevant objects* [21]. For instance, a user may be interested in describing a given area by finding associations between large towns (reference objects) and spatial objects in the road network, hydrography, and administrative boundary layers (task-relevant objects). The atom denoting the reference objects is called *key atom*. The predicate name of the key atom is specified in the **key is** clause.

---

[3]Here, the term classification rule denotes the result of a *supervised discrimination* process. On the contrary, Han & Kamber [9,10] use the same term to denote the result of an *unsupervised clustering* process.

4.4 Specification of primitive and pattern descriptors

The **analyze** clause specifies what descriptors, among those automatically generated by the Map Descriptor, can be used to describe the geographical objects extracted by means of the first primitive. The syntax of the analyze clause is the following:

analyze <*Primitive_descriptors*>

where:

<*Primitive_descriptors*> ::= <*Descriptor*> {, <*Descriptor*>}
parameters <*Parameter_specs*>{, <*Parameter_specs*>}
<*Descriptor*> ::= <*Predicate*>/<*Arity*>
<*Parameter_specs*> ::= <*Parameter_name*> **threshold** <*Integer*>

The specification of a set of parameters is required by the Map Descriptor to automatically generate some primitive descriptors.

The language used to describe generated patters is specified by means of the following clause:

with descriptors <*Pattern_descriptors*>

where:

<*Pattern_descriptors*> ::= <*Descriptor_specification*>
{; <*Descriptor_specification*>}
<*Descriptor_specification*> ::= <*Descriptor*> [***cost*** <*Integer*>] |
<*Descriptor*> [***with*** <*Terms_Spec*>]
<*Terms_Spec*>::= <*Term_Spec*>{,<*Term_Spec*>}
<*Term_Spec*> ::= <*Constant_Type*> |<*Variable_Type*>
<*Constant_Type*> ::= ***constant*** [<*Value*>]
<*Variable_Type*> ::= ***variable mode*** <*Variable_Mode*> ***role*** < *Variable_Role*>
<*Variable_Mode*> ::= ***old*** | ***new*** | ***diff***
<*Variable_Role*> ::= ***ro*** | ***tro***

The specification of descriptors to be used in the high-level conceptual descriptions can be of two types: either the name of the descriptor and its relative cost, or the name of the descriptor and the full specification of its arguments. The former is appropriate for classification tasks, while the latter is required by association rule mining tasks.

An example of a classification task activated by an SDMOQL statement is reported in Fig. 4. In this case, the Map Descriptor generates a symbolic description of some cells by using the predicates listed in the analyze clause. These are four concepts to be learned, namely *class(_)=system_of_farms*, *class(_)=fluvial_landscape*, *class(_)=.royal_cattle_track*, and *class(_)=.system_of_cliffs*. Here the function symbol *class* is unary and "_" denotes the anonymous variables *à la* Prolog. The user can provide examples of these four classes, as well as of other classes. Examples of systems of farms are considered to be positive for the first concept in the list and negative for the others. The converse is true for examples of fluvial landscapes, royal cattle track and system of cliffs. Examples of other classes are considered to be counterexamples of all classes for which rules will be generated. The only requirement for the INGENS user is the ability to detect and mark some cells that are instances of a class. Indeed, INGENS GUI allows the user both to formulate and run an SDMOQL query and to associate the description of each cell with a class.

```
SELECT x FROM x in Cell WHERE x->num_cell = 5
UNION SELECT x  FROM x in Cell WHERE x->num_cell = 8
UNION SELECT x FROM x in Cell WHERE x->num_cell = 11
UNION SELECT x FROM x in Cell WHERE x->num_cell = 15
UNION SELECT x FROM x in Cell WHERE x->num_cell = 16
UNION SELECT x FROM x in Cell WHERE x->num_cell = 17
UNION SELECT x FROM x in Cell WHERE x->num_cell = 27
UNION SELECT x FROM x in Cell WHERE x->num_cell = 28
UNION SELECT x FROM x in Cell WHERE x->num_cell = 34
UNION SELECT x FROM x in Cell WHERE x->num_cell = 83
UNION SELECT x FROM x in Cell WHERE x->num_cell = 84
UNION SELECT x FROM x in Cell WHERE x->num_cell = 89
mine classification as MorphologicalElements
for class(_)=.system_of_farms, class(_)=.fluvial_landscape, class(_)=.royal_cattle_track,
class(_)=.system_of_cliffs
analyze contain/2, part_of/2, type_of/1, subtype_of/1, color/1, … ,
        line_to_line/2, distance/2, line_to_region/2, …, point_to_region/2
parameters
        maxValuePointRegionClose        threshold  300,
        minValueLineLong                threshold  100,
        maxValueLinesClose              threshold  300,
        minValueRegionLarge             threshold  5000,
        maxValueRegionClose             threshold  500
with descriptors contain/2  cost  1; class/1 cost  0; subtype_of/1 cost  0;
        parcel_to_parcel/2  cost  0; slope_to_slope/2 cost 0; …
        canal_to_parcel/2 cost 0; …; font_to_parcel/2 cost 0; …
define knowledge
        font_to_parcel(Font,Parcel) = Topographic_Relation :-
                type_of(Font) = font, part_of(Font,Point) = true,
                type_of(Parcel) = parcel, part_of(Parcel,Region) = true,
                point_to_region(Point,Region) = Topographic_Relation.
…
criteria
intermediate minimize negative_example_covered      with tolerance 0.6,
intermediate maximize positive_example_covered      with tolerance 0.4,
intermediate maximize selectors_of_clause           with tolerance 0.3,
intermediate minimize cost                          with tolerance 0.4
final maximize positive_example_covered     with tolerance 0.0,
final maximize selectors_of_clause          with tolerance 0.0,
final minimize cost                         with tolerance 0.0
maxstar    threshold  25,
consistent threshold  500,
max_ps     threshold  11
recursion  =  off,
verbosity  =  { off, off, on, on, on }
```

Fig. 4. A complete SDMOQL query for a classification task. In this case, the INGENS user marks 5,11,34,42 as instances of the class "system of farms" and the cells 1, 2, 3, 4, 15, 16, 17 as instances of the class "other".

Rules generated for the four concepts are expressed by means of descriptors specified in the **with descriptors** list. They are specified by Prolog programs on the basis of descriptors generated by the Map Descriptor. For instance, the descriptor

font_to_parcel/2 has two arguments which denote two logical objects, a font and a parcel. The topological relation between the two logical objects is defined by means of the clause:

font_to_parcel(Font,Parcel) = Topographic_Relation :-
type_of(Font) = font, part_of(Font,Point) = true,
type_of(Parcel) = parcel, part_of(Parcel,Region) = true,
point_to_region(Point,Region) = Topographic_Relation.

In association rule mining tasks, the specification of pattern descriptors correspond to the specification of a collection of atoms: *predicateName*(t_1,..., t_n), where the name of the predicate corresponds to a <*Descriptor*>, while <*Term_Spec*> describes each term t_i, which can be either a constant or a variable. When the term is a variable the mode and role clauses indicate respectively the type of variable to add to the atom and its role in a unification process. Three different modes are possible: *old* when the introduced variable can be unified with an existing variable in the pattern, *new* when it is a not just present in the pattern, *diff* when it is a new variable but its values are different from the values of a similar variable in the same pattern. Furthermore, the variable can fill the role of reference object (*ro*) or task-relevant object (*tro*) in a discovered pattern during the unification process. The *is key* clause specifies the atom which has the key role during the discovery process. The first term of the key object must be a variable with mode new and role ro. The following is an example of specification of pattern descriptors defined by an SDMOQL statement for :

with descriptors
contain/2 **with variable mode old role ro, variable mode new role tro;**
type_of/2 **with variable mode diff role tro, constant;**
is key with variable mode new role ro, constant cultivation;

This specification helps to select only association rules where the descriptors *contain/2* and *type_of/2* occur. The first argument of a *type_of* is always a *diff* variable denoting a spatial object, and it can play the roles of both *ro* and *tro*, whereas the second argument, i.e. the type of object, is the constant 'cultivation', if the first argument is a reference object, otherwise it is any other constant. The predicate contain links the *ro* of type cultivation with other spatial objects contained in the cultivation. The following association rule:

type_of(X,cultivation), contain(X,Y), type_of(Y,olive_tree), X≠Y → contain(X,Z),
type_of(Z,almond_tree), X≠Z, Y≠X

satisfies the constraints of the specification and express the co-presence of both almond trees and olive-trees in some extensive cultivations.

4.5 Syntax for Background Knowledge and Concept Hierarchy Specification

Many data mining algorithms use background knowledge or concept hierarchies to discover interesting patterns. Background knowledge is provided by a domain expert on the domain to be mined. It can be useful in the discovery process. The SDMOQL syntax for background knowledge specification is the following:

<Background_Knowledge> ::= [<New_Knowledge>] {<Use_Knowledge>}
<New_Knowledge> ::= ***define knowledge*** *<Clause> {, <Clause>}*
<Use_Knowledge> ::= ***use background knowledge of users*** *<User> {, <User>}*
on *<Descriptor> {, <Descriptor>}*

In INGENS, the user can define a new background knowledge expressed as a set of definite clauses; alternatively, he/she can specify a set of rules explicitly stored in a deductive database and possibly mined in a previous step. The following is an example of a background knowledge specification:

Example 7: Definition of close_to *and import of the definition of* ravine.

close_to(X,Y)=true :- region_to_region(X,Y)=meet.
close_to(X,Y)=true :- close_to(Y,X)=true.
use background knowledge of users UserName1 **on** ravine/1

Concept hierarchies allow knowledge mining at multiple abstraction levels. In order to accommodate the different viewpoints of users regarding the data, there may be more than one concept hierarchy per attribute or dimension. For instance, some users may prefer to organize census districts by wards and districts, while others may prefer to organize them according to their main purpose (industrial area, residential area, and so on). There are four major types of concept hierarchies [8]:

- *Schema hierarchies*, which define total or partial orders among attributes in the database schema.
- *Set-grouping hierarchies*, which organize values for given attributes or dimensions into groups of constants or range values.
- *Operation-derived hierarchies*, which are based on operations specified by experts or data mining systems.
- *Rule-based hierarchies*, which occur when either a whole concept or a portion of it is defined by a set of rules.

In SDMOQL a specific syntax is defined for the first two types of hierarchies:

<Hierarchy> ::= [<New_Hierarchy>] [<Use_Hierarchy>]
<New_Hierarchy> ::= ***define hierarchy*** *<Schema_Hierarchy>*
| ***define hierarchy for*** *<Set_Grouping_Hierarchy>*
<Use_Hierarchy> ::= ***use hierarchy*** *<Name_Hierarchy>* ***of user*** *<User>*

The following example shows how to define some hierarchies in SDMOQL.

Example 8:A definition of a schema hierarchy for some activity-related attributes and a set-grouping hierarchy for the descriptor distance**.**

define hierarchy Activity **as**
level1:{business_activity, other_activity} < level0: Activity;
level2:{low_business_activity,high_business_activity}<level1:business_activity;
define hierarchy Distance for distance/2 **as**
level1:{far, near} < level0: Distance;
level2:{0, 1999} < level1: near;
level2:{2000, +inf} < level1: far;

The activity hierarchy can be used to mine multi-level spatial association rules [21].

4.4 Syntax for Interestingness Measure Specification

The user can control the data mining process by specifying interestingness measures for data patterns and their corresponding thresholds. The SDMOQL syntax is the following:

<Interestingness_Measures> ::= [<Criteria>] [<Thresholds>] {<Settings>}

<Criteria> ::= ***criteria*** *(****intermediate*** *|* ***final****)* *(****minimize*** *|* ***maximize****)*
<Parameter> ***with tolerance*** *<Value> {,(****intermediate*** *|* ***final****)*
*(****minimize*** *|* ***maximize****) <Parameter>* ***with tolerance*** *<Value>}*

<Thresholds> ::=<Parameter> ***threshold*** *<Threshold_Value>*
{, <Parameter> ***threshold*** *<Threshold_Value>}*

<Settings> ::= <Parameter> = <String_Value>

Interestingness measures may include: threshold values, weights, search biases in the hypotheses space and algorithm-specific parameters. In particular the user can bias the search in the hypotheses space by a number of *preference criteria*, such as the maximization of the number of covered examples or the minimization of the number of variables in the body of a learned clause. He/she can also set thresholds such as *confidence*, *support* or *number of learned concepts*. Finally, the user can set the value of a generic input parameter of a data mining algorithm.

4.5 Syntax for Visualization

Data mining results should be displayed using rule visualization tools or some different output forms. SDMOQL provides the following primitives for displaying results in different forms:

<Result_Displaying> ::= ***display as*** *<Form>*
*[****at level*** *<Int_Value>* ***for*** *<Hierarchy_Name>],*

where *<Form>* describes the output form, for example, *if-then* rules or *tree*. Moreover, if a hierarchy is available, mined results can be represented at different concept levels. This is particularly true in the case of multiple-level association rules.

5. Mining classification rules for topographic map interpretation

In the previous section, the syntax of SDMOQL has been defined. Here we present a data problem concerning the generation of classification rules for topographic map interpretation. Let us suppose that a GIS user needs to locate a "*sistema poderale*" (system of farms) in the large territory of his/her interest. This geographical object is not present in the GIS model, thus, only the specification of its operational definition will allow the GIS to find cells containing a system of farms in a vectorized map. Who can provide it? The user is not able to do so for a number of reasons.

Firstly, providing the GIS with operational definitions of some environmental concepts is not a trivial task. Often only declarative and abstract definitions are available, which are difficult to compile into database queries.

Secondly, the operational definitions of some geographical objects are strongly dependent on the data model that is adopted by the GIS. Finding relationships between density of vegetation and climate is easier with a *raster data model*, while determining the usual orientation of some morphological elements is simpler in a *topological data model*.

Thirdly, different applications of a GIS require the recognition of different geographical elements in a map. Providing the system in advance with all the knowledge required for its various application domains is simply illusory, especially in the case of wide-ranging projects such as those set up by governmental agencies.

A solution to these problems can be found in the application of data mining techniques. For instance, an INGENS user can train the system to recognize cells with systems of farms, by performing the SDMOQL query in Fig. 4. The interpreter analyzes the query and verifies its syntactic and semantic correctness. Then the Map Descriptor generates a symbolic description for each specified cell (see Fig. 3) and the expert associates each symbolic description with a concept, in order to define the training set. Association is made by *binding* variable terms of one of the four concepts to be learned to the constants terms in the descriptions of map cells. This step is necessary to create the training set of positive and negative examples for the learning system ATRE [18], which is used in INGENS for classification tasks. The user marks 5, 11, 34 as instances of the class "system of farms", the cells 8, 16, 17 as instances of the class "fluvial landscape", the cells 15, 27, 28 as instances of the class "royal cattle track" and the cells 83, 84, 89 as instances of the class "system of Cliffs". This binding function is supported by INGENS GUI. The training set obtained is input to ATRE, which returns the classification rules. With reference to the above query, ATRE generates the following clauses:

class(X1) = system_of_farms ←
contain(X1,X2)= true,
area_parcel(X2) in [102.787..249.525],
density_parcel(X2) = high,
font_to_parcel(X3,X2) = outside,

"A cell is an example of a *system of farms* if it contains a parcel (X2) that has an area between 102,787 and 249,525 square meters and a high vegetation density, and a font (X3) that is outside the parcel."

class(X1) = .fluvial_landscape ←
contain(X1,X2) = true,
extension_road(X3) in [234.0..440.0],
canal_to_road(X2,X3) = almost_parallel,
distance_canal_to_road(X2,X3) in [42.0..300.0].

"A cell is an example of a *fluvial landscape* if it contains a canal (X2) and a street (X3). The street has an extension between 234.0 and 440.0 meters and is almost parallel to the canal. In particular, the distance between the canal and the street is between 42.0 and 300.0 meters."

class(X1) = royal_cattle_track ←
contain(X1,X2) = true,
extension_road(X2) in [1002.0..1162.0],
subtype_of(X2) = main_road.

"A cell is an example of a *royal cattle track* if it contains a street (X2) that is a main road and has an extension between 1002.0 and 1162.0 meters."

class(X1)= system_of_cliffs ←
contain(X1,X2) = true,
distance_contour_slope_to_contour_slope(X2,X3)in [2.0..74.0],
extension_contour_slope(X2) in [79.0..307.0].

"A cell is an example of a *system of cliffs* if it contains two contour slopes (X2, X3), such that the distance between them is between 79.0 and 307.0 meters. One contour slope (X2) has an extension between 2,0 and 74,0 meters."

Whether the induced theory is "correct", that is, whether it classifies correctly all other examples of map cells not in the training set is beyond the scope of this work. However, it is noteworthy that these rules are coherent with the definitions given by town planners for the four morphological concepts of interest [19].

Operational definitions like those reported above can be used either to retrieve new instances of the learned concepts from the Map Repository or to facilitate the formulation of a query involving geographical objects not present in map legends. For instance, by submitting the following query:

SELECT C
FROM M in Map, C in Cell, R in Road
WHERE M->name = "Canosa" AND C->map = M AND R->log_incell = C
AND R->type_road="main_road" AND class(C) = fluvial_landscape

the user asks INGENS to find all cells in the Canosa map that are classified as fluvial landscape and contain a main road. To check the condition defined by the predicate *class(C)=fluvial_landscape*, the *Query Interpreter* generates the symbolic description of each cell in the map and asks the Query Engine of the Deductive Database to prove the goal *←class(C)=fluvial_landscape* given the logic program above.

6. Related work

Several data mining query languages have been proposed in the literature. MSQL is a rule query language proposed by Imielinski and Virmani [12] for relational databases. It satisfies the closure property, that is, the result of a query is a relation that can be queried further. Moreover, a cross-over between data and rules is supported, which means that there are primitives in the language that can map generated rules back to the source data, and vice versa. The combined result of these two properties is that a data mining query can be nested within a regular relational query. SDMOQL do not allow users to formulate nested queries, however, as pointed out at the end of the previous section, it supports some form of cross-over between

data and mined rules. This is obtained by integrating deductive inferences for extracted rules with data selection queries expressed in OQL.

Another data mining query language for relational databases is DMQL [8]. Its design is based on five primitives, namely the set of data relevant to a data mining task, the kind of knowledge to be mined, the background knowledge to be used in the discovery process, the concept hierarchies, the interestingness measures and thresholds for pattern evaluation. As explained in Section 4.1, the design of SDMOQL is based on a different set of principles. In particular, the specification of data relevant to a data mining task involves a separate specification for the geographical objects of interest for the application, for the set of automatically generated (primitive) descriptors and for the set of descriptors used to specify the patterns. An additional design principle is that of visualization, since in spatial data mining it is important to specify whether results have to be visualized or presented in a textual form.

GMQL is based on DMQL and allows the user to specify the set of relevant data for the mining process, the type of knowledge to be discovered, the thresholds to filter out interesting rules, and the concept hierarchies as the background knowledge [16]. In the process of selecting data relevant to the mining task, the user 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. Conditions may involve spatial predicates on topological relations, distance relations and direction relations. Although data can be selected from several tables, mining is performed only on a single table which result from an SQL query (single table assumption). GMQL queries can generate different types of knowledge, namely characteristic rules, comparison rules, clustering rules and classification rules. In Koperski's thesis, an extension to association rules was also proposed but not implemented. Differently from GMQL, SDMOQL separate the physical representation of geographical objects from their logical meaning. Moreover, all observations reported above for DMQL applies to GMQL as well.

Finally, it is noteworthy that some object-oriented extension of DMQL, named ODMQL, has also been proposed [4]. The design of ODMQL is based on the same primitives used for DMQL, so the main innovation is that each primitive is in an OQL-like syntax. Path expressions are supported in ODMQL, while more advanced features of object-oriented query languages, such as the use of collections and methods, are not mentioned. An interesting aspect of ODMQL, which will be taken into account in further developments of SDMOQL, is that some concept hierarchies are automatically defined by the inheritance hierarchy of classes.

7. Conclusions

In this paper, a spatial data mining language for a prototypical GIS with knowledge discovery facilities has been partially presented. This language is based on a simplified OQL syntax and is defined in terms of the eight data mining primitives. For a given query, these primitives define the set of objects relevant to a data mining task,

the kind of knowledge to be mined, the set of descriptors to be extracted from a digital map, the set of descriptors to be used for pattern description, the background knowledge to be used in the discovery process, the concept hierarchies, the interestingness measures and thresholds for pattern evaluation, and the expected representation for visualizing the discovered patterns. An interpreter of this language has been developed in the system INGENS. It interfaces a Map Descriptor module that can generate a first-order logic description of selected geographical objects. A full example of the query formulation and its results has been reported for a classification task used in the qualitative interpretation of topographic maps. An extension of this language to other spatial data mining tasks supporting quantitative interpretation of maps is planned for the near future.

Acknowledgments

This work is part of the MURST COFIN-2001 project on "Methods of knowledge discovery, validation and representation of the statistical information in decision tasks". Thanks to Lynn Rudd for her help in reading the paper.

References

1. DeMers, M.N.: Fundamentals of Geographic Information Systems. 2nd ed., John Wiley & Sons, (2000)
2. Dzeroski, S., Lavrac, N. (eds.): Relational Data Mining. Springer, Berlin, Germany, (2001)
3. Egenhofer, M.J.: Reasoning about Binary Topological Relations. In Proceedings of the Second Symposium on Large Spatial Databases, Zurich, Switzerland, (1991) 143-160
4. Elfeky M.G., Saad A.A., Fouad S.A.. ODMQL: Object Data Mining Query Language. In Proceedings of Symposium on Objects and Databases, Sophia Antipolis, France, (2001)
5. Ester, M., Frommelt, A., Kriegel, H. P., Sander, J.: Algorithms for characterization and trend detection in spatial databases. In Proceedings of the 4th Int. Conf. on Knowledge Discovery and Data Mining, New York City, NY, (1998) 44-50
6. Ester, M., Gundlach, S., Kriegel, H.P., Sander, J.: Database primitives for spatial data mining, In Proceedings of Int. Conf. on Database in Office, Engineering and Science, (BTW '99), Freiburg, Germany, (1999)
7. Güting, R.H.: An introduction to spatial database systems. VLDB Journal, 3,4 (1994) 357-399
8. Han, J., Fu, Y., Wang, W., Koperski, K., Zaïane, O. R.: DMQL: a data mining query language for relational databases. In Proceedings of the Workshop on Research Issues on Data Mining and Knowledge Discovery, Montreal, QB, (1996) 27-34
9. Han, J., Koperski, K., Stefanovic, N.: GeoMiner: A System Prototype for Spatial Data Mining. In Peckham, J. (ed.): SIGMOD 1997, Proceedings of the ACM-SIGMOD International Conference on Management of Data. SIGMOD Record 26, 2 (1997) 553-556
10. Han, J., Kamber, M.: Data mining, Morgan Kaufmann Publishers (2000)
11. Han, J., Kamber, M., Tung, A.K.H.: Spatial clustering methods in data mining. In H. J. Miller & J. Han (eds.), Geographic Data Mining and Knowledge Discovery, Taylor and Francis, London, UK, (2001) 188-217

12. Imielinski T., Virmani A.: MSQL: A query language for database mining. Data Mining and Knowledge Discovery, 3(4) (1999) 373-408
13. Klosgen, W., May, M.: Spatial Subgroup Mining Integrated in an Object-Relational Spatial Database. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) Principles of Data Mining and Knowledge Discovery (PKDD), 6th European Conference, LNAI 2431, Springer-Verlag, Berlin, (2002) 275-286
14. Koperski, K., Han, J.: Discovery of spatial association rules in geographic information database. In Advances in Spatial Database, Proceedings of 4th Symposium, SSD '95. (Aug. 6-9. Portland, Maine), Springer-Verlag, Berlin., (1995) 47-66
15. Koperski, K., Adhikary, J., Han, J.: Knowledge discovery in spatial databases: progress and challenges, Proc. SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, (1996)
16. Koperski, K.: A progressive refinement approach to spatial data mining. Ph.D. thesis, Computing Science, Simon Fraser University, (1999)
17. Lanza, A., Malerba, D., Lisi, L.F., Appice, A., Ceci, M.: Generating Logic Descriptions for the Automated Interpretation of Topographic Maps. In D. Blostein and Y.-B. Kwon (eds.) Graphics Recognition: Algorithms and Applications, Lecture Notes in Computer Science, 2390, Springer, Berlin, Germany, (2002) 200-210
18. Malerba, D., Esposito, F., Lisi, F.A.: Learning recursive theories with ATRE. In: Prade, H. (ed.): Proc. 13th European Conference on Artificial Intelligence, John Wiley & Sons, Chichester, England, (1998) 435-439
19. Malerba, D., Esposito, F., Lanza, A., Lisi, F.A, Appice, A.: Empowering a GIS with Inductive Learning Capabilities: The Case of INGENS. Journal of Computers, Environment and Urban Systems, Elsevier Science (in press).
20. Malerba, D., Esposito, F., Lanza, A., Lisi, F.A.: Machine learning for information extraction from topographic maps. In H. J. Miller & J. Han (eds.), Geographic Data Mining and Knowledge Discovery, Taylor and Francis, London, UK, (2001) 291-314
21. Malerba, D., Lisi, F.A.: An ILP method for spatial association rule mining. Working notes of the First Workshop on Multi-Relational Data Mining, Freiburg, Germany (2001) 18-29
22. Preparata, F., Shamos, M.: Computational Geometry: An Introduction. Springer-Verlag, New York (1985)
23. Sagonas, K. F., Swift, T., & Warren, D. S.: XSB as an Efficient Deductive Database Engine. In R. T. Snodgrass, & M. Winslett (Eds.): Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, SIGMOD Record 23(2), (1994) 442-453
24. Sander J., Ester M., Kriegel H.-P., Xu X.: Density-Based Clustering in Spatial Databases: A New Algorithm and its Applications. Data Mining and Knowledge Discovery, Kluwer Academic Publishers, 2(2) (1998) 169-194