# An Integration Approach to the Management of Geographical Information: CARTECH(*)

Franco Arcieri(1), Enrico Nardelli(2)

(1) Algotech s.r.l., Via Biella 10, 00182, Roma, Italy.
(2) Istituto di Analisi dei Sistemi ed Informatica, C.N.R., Viale Manzoni 30, 00185 Roma, Italy.

## Abstract

*In this paper CARTECH, an SQL-based geographical information system, is presented. It is based on the integration approach, sketched in [14], of coupling systems for dealing with traditional, descriptive data, with tools for the management of spatial data. Its key feature is the extension of the relational data model, to deal in a uniform way with both descriptive and spatial features of geographical information. The SQL language has also been extended in order to provide a uniform interface for processing both alphanumeric and spatial-geometric data. The architecture integrates the INGRES data base management system for the treatment of non spatial data and the GEOTECH library for the management of spatial/geometric data. We briefly describe the logical data model and focus on architectural aspects of the developed integrated system, discussing the extended-SQL interpreter and query processor. An application of CARTECH to land resource management is presented and performances are discussed.*

## 1. Introduction

An increasing and increasing interest is spreading in the database field for systems able to deal with spatial data [22, 25, 26, 28]. This derives from the emerging needs of having databases able to efficiently support new applications, like CAD or territorial information management systems, which deal with information having both descriptive and spatial features.

The difference and the novelty of these applications derives from the intrinsically complex nature of such data. For example, the description of the physical shapes and relations existing among the parts constituting an assembly, or of the location and geometry of a lake is not as easy as to represent an address or a price. In the former case the information associated with each entity is not just a name or a number, but it is a geometrical shape in the space, and it has to be considered as a single atomic object. It is cumbersome to represent it using only simple data types and, in this case, efficiency problems arise during its processing.

The interest for geographical databases is due to the importance, in many application sectors, of the representation of the territory and of the structures built on it. Some examples are the planning of services structures, the localizations of geological places, the urban areas administration, the map drawing, the optimization in natural resources use, and so on. In all these cases it is important to have at disposition techniques and tools that allow an efficient representation and manipulation of territorial data [27, 24].

A geographical entity has generally associated two kinds of information. Firstly, the description of its geometrical shape and its spatial position, secondly, the presence of descriptive attributes, like the name. This double nature of geographical entities has a correspondence in the structure of queries that we can make: so, a DBMS for geographic applications have to permit queries with combined geometric and descriptive attributes, having the same efficiency and facility of use of the normal operations defined on the traditional attributes [20]. It has to support direct spatial search, based on the topological properties of objects; operations on geometric attributes, as computing the area of a given region or the river's length; indirect spatial search, allowing the identification of geometric entities on the basis of their descriptive properties.

A widely used approach to the realization of geographical databases rests on the coupling of traditional database management systems (DBMS) and specialized systems for geometrical data or image data processing. The rationale of such an approach relies both on the wide availability and reliability of the current DBMS's and the powerful capabilities of specialized systems for the spatial/geometrical data processing. In fact, since DBMS's have been dealing since early '70 with alphanumeric data and with typical requirements of commercial and business applications, they are now widely spread and able to process descriptive data in a very powerful and reliable manner. On the other side, geometric data and image data handling applications, with their strong requirements for manipulating huge quantities of spatial data and efficient algorithms for exploiting the geometrical and topological characteristics of such data, have lead during the last two decades to the development of specialized environments containing a lot of very powerful capabilities for efficiently processing geometrical and topological

(*) Partially supported by research line "Multidata" of the "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo"-CNR and working group "Basic GOODS" of the "ESPRIT - Basic Research Action"



0-8186-2697-6/92 $03.00 © 1992 IEEE

properties of spatial data.

This approach was proposed in [14], where it was firstly sketched the use of a logical data model to overcome the drawbacks due to an integration directly relying on the physical implementation. Here we present the continuation of that work, by introducing and discussing in detail the architecture of the integrated system.

Beside its practical importance, the study of geographical databases has a theoretical interest that derives from the fact they can be considered as a paradigmatic example of what we call "multimedia" applications. By "multimedia", in this case, we denote a different concept from current proposals, in which the word takes a physical meaning referring to the diversity and variety of representation and storage media.

In our interpretation, "multimedia" denotes an application whose complexity derives from the variety and from the structural richness of relations that exist among data of the application. In this framework, geographical databases may be considered as a paradigmatic example of multimedia data, because in the representation and management of spatial-geometric information, if we want a faithful representation of the reality, we have to take into account the numerous and complex topological and geometric relations which implicitly exist among data (as adjacency, intersection, inclusion, distance, ...).

This paper is structured as it follows: in section 2 relevant problems related to geographical databases and proposed solutions are addressed; in section 3 a general description of our approach to the problem is presented; section 4 briefly describes the logical data model on which CARTECH is based; section 5 presents geoSQL, our proposal of extension to the SQL language; in section 6 we describe the main features of GEOTECH, the library for the management of spatial data; in section 7 the query resolution strategies are discussed, and, finally, in section 8 the application for land resource management is discussed.

## 2. Approaches to the treatment of geographical data

In [14, 15] an analysis of advantages and drawbacks of different approaches to the realization of geographical databases was presented. In the integration approach, it is necessary to have a logical data model the defines both a uniform reference schema for the specification of the integration between subsystems and provides the user with a high level data manipulation language. The model thus guarantee to the user a uniform and integrated view of geographical entities while allow an access trough both descriptive and spatial characteristics. In this section we review approaches to the representation of complex data.

Traditional database approaches provide representation structures which are unsuitable and inefficient when they are used for the representation of complex entities like those found in more advanced applications.

The most used model in the database logic design is the relational model [13, 29], because of its simplicity, flexibility and theoretical bases founded on the set theory. But this model has severe limits when it is employed in advanced applications: the most obvious restriction is that any relation scheme has to be in first normal form. In these conditions the representation of complex objects which can not be described with simple data, is very difficult and complicated [19, 1, 20, 23].

Atomic domains can not represent adequately spatial-geometric information: we have to represent it by dividing the information regarding the object shape and position in several pieces, which are then represented as tuples of a relation and, generally, stored in different relations of the database [11]. But this information is a conceptual unity from the user point of view, and in a conceptual design it is represented with a single attribute. As a consequence, a disaggregated representation forces a more difficult formulation of the queries : the user has to know deeply the schemes of all relations, and a query, even if simple, may be composed by many nested subqueries. Every time we have to refer to the shape of an entity, for example in a predicate on a geometric attribute, it is necessary to approach a new table: therefore, a simple condition in the *where* clause is generally translated in a *select* clause.

Efficiency is also a critical point: even few entities request a great deal of memory, so to resolve geometric operations on the data becomes very expensive if we use only the primitives of relational systems. Consider, for example, a metric query that asks for the distance between two regions; here, we have to control all the possible couples of tuples of the two relation; the time complexity of this operation is quadratic, and this can easily cause unacceptable resolution time.

It has been acknowledged that the possibility to represent non atomic values is a feature of any advanced modeling tool. Therefore, to enrich the modeling capabilities of the relational model, it is necessary to give up the first normal form constraint. This choice implies that the domain of a relation attribute is not simple, but a set of values.

Models permitting set valued attributes usually also allow functions that compute single values starting from a given set of values. Some models [1] generalize this approach allowing relation valued attributes, i.e. the value of an attribute of a relation can be, in its turn, a relation.

Models looking at the representation and the manipulation of spatial data have also been considered. Some interesting approaches are based on the introduction of abstract data types to model geometric properties of the represented data and their manipulation primitives. For example, [17] introduces data types as point, line and region to allow the description of geometrical entities and a set of operators, formally specified by a many sorted algebra, allowing to carry out the usual geometrical operations on them.

A similar approach is proposed in [20]. Here, the introduction of specific domains, geometric and topological operators permits a high level manipulation of

geometrical entities. In particular, segments and regions are objects whose internal representation is hidden by the primitives of the language. To each domain are associated appropriate representation structures: for instance, the same region can be described with a sequence of segments in an application, with points in another and with points having a different accuracy in another one.

The use of object oriented paradigms [8, 7, 9] for DBMSs definition seems promising and offers many advantages: it allows high level manipulation and representation of the entities by hiding implementation details; the concept of inheritance offers many advantages to designers; moreover, data and programs are stored together because they are two parts of the same entity.

As a consequence, it is possible to face design problems at a high level of abstraction: for example, the real world may be represented in a more direct way by appropriate structures reflecting the peculiar aspects of the reality itself, without being concerned with a logical representation that matches physical structure of data.

Unfortunately, although there is a considerable experimental activity [7, 18, 10], there is today neither a model accepted as a reference nor a precise formal base for the object oriented paradigm [8]. Moreover, some of the advantages mentioned before lose most of their validity when, in the practice, we have to resolve queries that involve many entities and regard a large part of the database. In these cases, object oriented DBMSs are not efficient: a main reason is that search paths are usually designed on the basis of the messages that the different instances exchange and they are not based on the value that the attributes take up. The related search procedures can become expensive when a great deal of data is considered.

## 3. General description of the proposed approach

The idea on which we have based our proposal is that the descriptive component and the spatial-geometrical one of a given territorial datum have to be separated at the physical level, to obtain efficiency, but they have to be integrated at the logical level, so their representation and manipulation are simple and immediate. It has to be possible to refer to the shape of a region or to an entity that has a spatial-geometrical component, in the same way as to any traditional attribute of a relational DBMS. On the other hand, the treatment of spatial-geometrical data is fulfilled by special purpose functions assuring higher efficiency.

Our choice for supporting logical integration has been of relying on the well founded relational theory [12], and to suitably extend the relational data model in order to represent in integrated way the descriptive and geometric part of geographical information. The possibility of treating in a relation geometric values as well as traditional alphanumeric attributes allows to extend the SQL language with new simple syntactic structures: topological predicates may be expressed on the geographical entities and it is possible to apply some geometric operators on these. From now on, we will refer to this extension of SQL language by the name of geoSQL.

CARTECH architecture essentially consists of two functional modules: the relational DBMS and the spatial-geometric module, i.e. a set of functions for the spatial-geometric data management. The choice to represent and separately manage these two types of information, generates two different activities of data manipulation. The partial results have then to be compared and integrated for obtaining the final result. In particular, an interpreter checks the syntactic correctness of the issued queries and calls the modules appropriate to their resolution. The strategy for queries resolution firstly manages the geometric part of the query and then makes use of the outcome for the resolution of the relational part by using the relational DBMS. The different actions carried on by the relational DBMS and by the spatial module are completely hidden to the user. He can access and manipulate data only through the primitives of the geoSQL language.

The above described approach leads to an architecture, shown in figure 1, consisting of the following functional modules:

- an *interface* which allows the users to access to system;
- a *query processor* for the identification and separation of the descriptive and geometric parts of the query;
- a *processor of spatial queries*; it manages spatial subqueries and creates temporary tables containing the partial results obtained from the their resolution;
- a *relational DBMS* that resolves the descriptive part of the query using the temporary tables produced by the spatial query processor.

The *interface* connects the system core to the human user or the application programmers: it makes available the system functionalities, recognizes the commands and displays the result of the operations.

The *query processor* divides spatial subexpressions, involving exclusively geometric attributes, from non geometric ones, which refer to alphanumeric attributes. Spatial subexpressions can be found either in *select* clauses as geometric operators, or in *where* clauses as topological predicates. This functionality is carried out by a syntactic analyzer that sends the identified subexpressions to the spatial processor for their evaluation.

The *relational DBMS*, based on the INGRES system, evaluates the queries concerning alphanumeric data and executes the commands for creating and modifying the tables used by the spatial processor.

The *spatial query processor*, based on GEOTECH library, accepts as input spatial subexpressions. It carries out the computation of the temporary tables of the partial results, that is tables containing the regions satisfying the imposed topological conditions. Also, it computes the required geometric functions. This activity produces geometric values in the case of union or intersection

operators, numeric values when the area of the specified regions is computed.

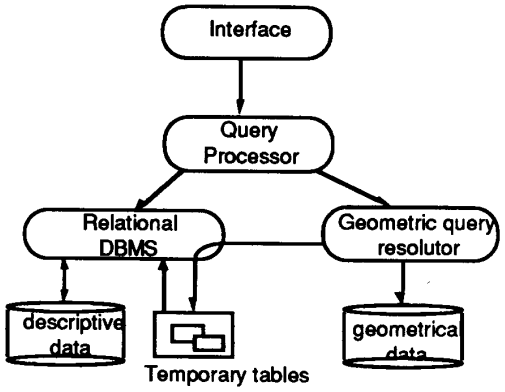



Fig. 1: CARTECH System Architecture

Algotech company has developed the CARTECH system on a 80386 based machine using the SCO-UNIX system V operating system, the INGRES data base management system, the GEOTECH library for spatial/geometric data management, the X-Window and MOTIF toolkits and the C++ programming language. GEOTECH is the Algotech internal library of specialized modules for efficiently representing and manipulating geometrical data. CARTECH has been used to develop geographical applications: as an example, a system for land resource management is presented and discussed in section 8.

## 4. The logical data model

In [15] a model for the management of complex data based on an extension of the relational model was formally defined. In [16] its formal properties were investigated and its completeness and soundness were proved. The focus of the model, which we describe briefly in this section, is to introduce abstract data types for the specification of the attribute domains. Abstract data types allow to define a set of objects through their visible properties, without considering their effective physical representation. As a consequence, a manipulation of the objects at the same level of abstraction that objects present in reality can be performed. At the same time, the approach permits the definition of possibly several physical representations of a same objects. This possibility can furnish completely hidden versions of an object, each of them appropriate to a different application. Furthermore, the approach also allows to represent and manipulate different kinds of complex data in a uniform and integrated modeling framework [15].

We have introduced an extended relational algebra that allows the safe manipulation of the relations of the extended model. In the extended algebra we define two general facilities for aggregating and disaggregating the values of the set-valued attributes of the model, respectively G-Compose and G-Decompose operators. These primitives permit a high level manipulation of the relations exploiting the semantic content of the specifications of the domains, i.e. the nature of the elements, the operations and functions defined in the specification of the relative abstract data type. Obviously, every abstract data type specification contains the description of a set of elements and functions according to the data to model and to the operations to be performed on these data. We now briefly describe the operators of the extended algebra suitably defined to manipulate such non atomic valued attributes. We refer to [15] for a formal and complete treatment, and to section 5 for examples of their use.

The operator G-Compose, applied on a set of tuples of the relation, groups them together on the base of the equality of one or more specified attributes, geometric or descriptive. A set G of attributes and a set, of the same cardinality of G, of functions, called 'fusion functions' is also specified. For each attribute of G, the G-Compose operator applies the corresponding function to the collection of values obtained from the grouping. G-Compose has a sort of inverse function, G-Decompose, which in the case of set-valued attributes ungroups the values of the set.

To support the modeling of geographical applications we have defined an abstract data type, called *Geometry*(S), to represent components and properties of geometrical object. The approach chosen for the definition of the abstract data type *Geometry*(S) is to introduce an algebra as we describe below [Kam83]. In fact an algebra is defined through its elements and the operations it makes available on them.

Let S be a not empty finite set, let P(S) denote the powerset of S, we indicate by $H_S$ the set $H_S=P(P(S))$, that is the set of all sets of sets of elements of S. We call SHAPES on S the 6-tuple $Y_S=(H_S, \cup, \cap, \cap^*, geo, compl)$, where $\cup$ and $\cap$ denote, respectively, the union and the intersection operations in the way they are used in set theory, i. e., for all $A, B \in S$,

- $A \cup B = \{ c \in H_S : c \in A \vee c \in B \}$;
- $A \cap B = \{ c \in H_S : c \in A \wedge c \in B \}$.

and $\cap^*$, *geo* and *compl* are defined as it follows

- $A \cap^* B = \{ c \in P(S) : (\exists a)\ (\exists b)\ a \in A \wedge b \in B \wedge c = a \cap b \}$;
- $geo(A) = \{ X \}$, where $X = \{ x \in S : a \in A \wedge x \in a \}$.
- $compl(A,B) = \{ Y \}$, where $Y = \{ y \in S : a \in A \cup B \wedge y \notin a \}$.

An element of the SHAPES algebra is called Shape. Given a Shape A, each element of A (notice that each element of A is a set of elements of the ground set S) is called an A-component. Informally, the $\cap^*$ operator, when it is applied to two Shapes A and B, returns a Shape C such that every C-component is the intersection between one A-component and one B-component. The *geo* operator applied to a Shape A containing more than one component

returns a Shape whose single component results from the union of all the A-components. If A contains only one component, the *geo* operator returns the same Shape. The *compl* operator applied to Shapes A and B returns a Shape whose single component, C, is the set of all points of S belonging neither to A components nor to B ones. Figures 2a,...,2f show in informal way, the operators of the algebra introduced in this section. The shadowed part of the drawings represents the result of the indicated operation.

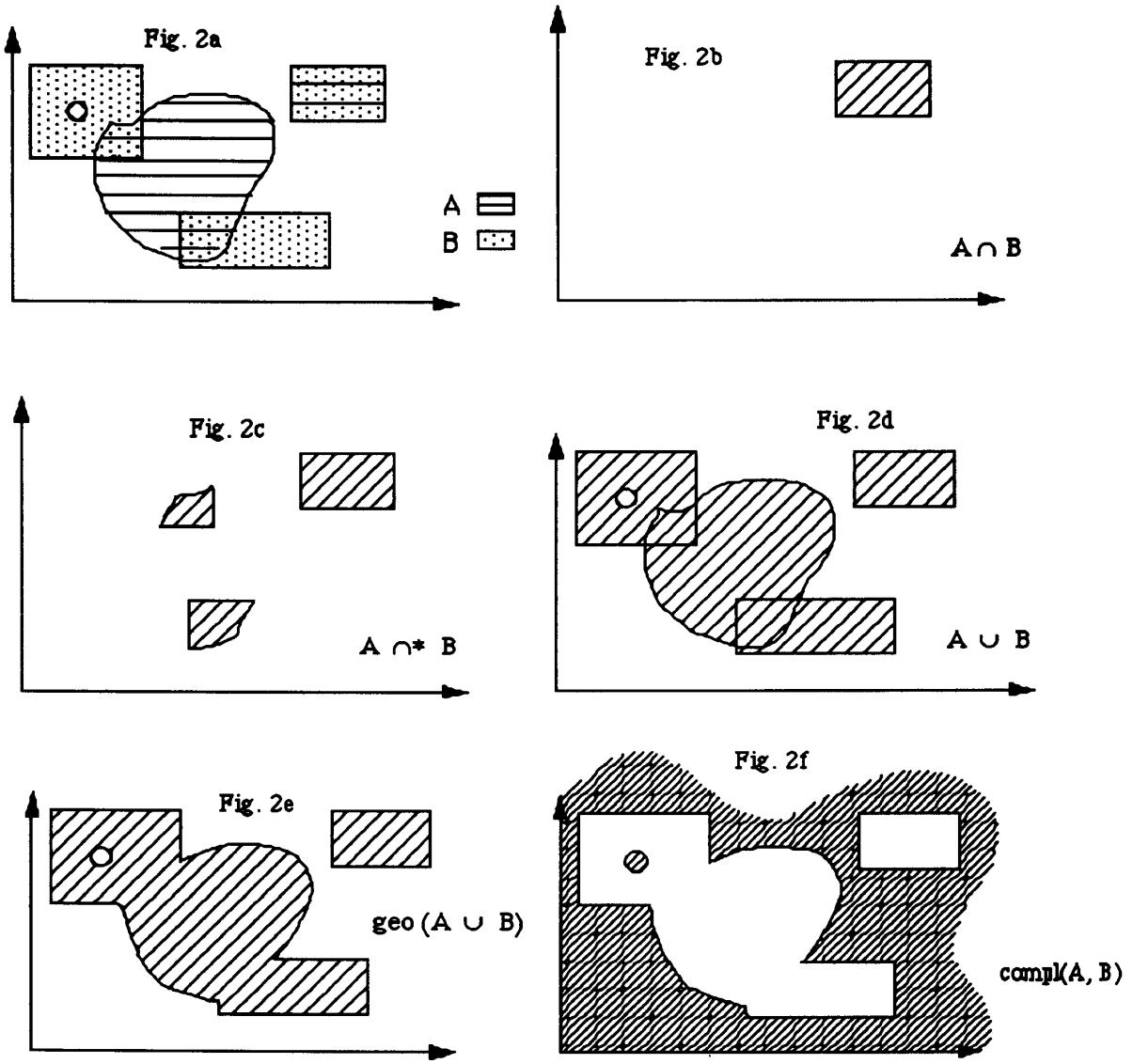



Fig. 2: Examples of the operators of SHAPES algebra.

We call 'geometrical attributes' those attributes whose values belong to the introduced abstract data type *Geometry*(S). A value of a geometrical attribute is therefore an element of the domain $Y_S=(H_S, \cup, \cap, \cap^*, geo, compl)$. We call 'descriptive' the "traditional" attributes to distinguish them from the geometrical ones. Owing to the nature of the geometrical attributes the scheme of a relation that contains a geometric attribute is not in first normal form.

## 5. The geoSQL language

Our query language geoSQL extends SQL to deal with the new relational operators (G-Compose and G-Decompose) and with the ADT *Geometry*(S). Before discussing such extensions let us first present some examples of its use, referring to [15] for a more detailed description.

Given the relation RIVERS (fig. 3) let us suppose that we want to group the river segments of those rivers that cross a same region. The following extended relational algebra expression can be issued (for syntactical explanation we refer to [15]):

G-Compose$_{\text{Crossed-region}}$($\cup$; River-shape)(RIVERS)

| RIVERS | | | |
|---|---|---|---|
| Name | Crossed-region | Length | River-shape |
| river_1 | A | 2 | { {riv_11} } |
| river_1 | B | 4 | { {riv_12} } |
| river_2 | B | 2.5 | { {riv_2} } |
| river_3 | A | 4.7 | { {riv_31} } |
| river_3 | H | 5 | { {riv_32} } |
| river_3 | B | 6 | { {riv_33} } |
| river_4 | C | 1 | { {riv_4} } |
| river_5 | B | 10 | { {riv_5} } |

Figure 3: relation RIVERS

In this case the G-Compose operator groups the values of the attribute River-shape applying to the grouped values the fusion function $\cup$ and discards the remaining attributes. The grouping is done on the basis of the equality of the values of the attribute Crossed-region. Relation RIVERS-BY-REGION, in figure 4 below, show the result:

| RIVERS-BY-REGION | |
|---|---|
| Crossed-region | River-shape |
| A | { {riv_11}, {riv_31} } |
| B | { {riv_12}, {riv_2}, {riv_5} , {riv_33}} |
| C | { {riv_4} } |
| H | { {riv_32} } |

Figure 4: an example of G-Compose operator.

The value of the geometric attribute River-shape in each tuple of the relation RIVERS-BY-REGION is thus the set of the river segments belonging to the same region.

Through the use of a set of fusion functions in the G-Compose operator it is also possible to calculate new values related to the new obtained entity from the values associated to the merged entities:

G-Compose$_{\text{Name}}$(+, *geo*; Length, River-shape)(RIVERS).

The result is shown in Relation RIVER-LENGTHS of figure 5 where, starting from relation RIVERS of figure 2, we have merged the values of the River-shape attribute on the basis of the equality of the values of Name attribute and have computed the total length of the rivers.

| RIVER-LENGTHS | | |
|---|---|---|
| Name | Total-length | River-shape |
| river_1 | 6 | { {riv_11, riv_12} } |
| river_2 | 2.5 | { {riv_2} } |
| river_3 | 15.7 | { {riv_31, riv_32, riv_33} } |
| river_4 | 1 | { {riv_4} } |
| river_5 | 10 | { {riv_5} } |

Figure 5: G-Compose operation with set of fusion functions.

Notice the difference in the expression of values for River-shape attribute, obtained through the use of the *geo* fusion function. By { {riv_11, riv_12} } we indicate the simple shape containing the union between the set of the points denoting the spatial extension of the river segment riv_11 and the one denoting the spatial extension of the river segment riv_12. The value of the geometrical attribute in RIVERS-4 is therefore a simple Shape since it results from the application of the fusion function *geo*. It follows that the resulting relation cannot be disaggregated in the original components.

Notice that a safe use of the fusion functions in the G-Compose operator is assured because, due to our ADT approach, the manipulation functions allowed on set of values of an attribute $A_i$ are only the operation defined in the ADT specification of *type*($A_i$).

Referring to the relation RIVERS-BY-REGION of Fig. 4, let us suppose that we want to derive the segments of the represented rivers, with respect to the region they cross. The following extended relational algebra expression can be issued:

G-Decompose$_{\text{Crossed-region}}$(River-shape)(RIVERS-BY-REGION)

The result is represented in relation REGIONS-AND-SHAPES shown below in figure 6. In general, this operator disaggregates the values of a specified attribute (e.g. River-shape) with respect to a specified (set of) attribute(s) (e.g. Crossed-region).

| REGIONS-AND-SHAPES | |
|---|---|
| Crossed-region | River-shape |
| A | { {riv_11} } |
| A | { {riv_31} } |
| B | { {riv_12} } |
| B | { {riv_2} } |
| B | { {riv_5} } |
| B | { {riv_33} } |
| C | { {riv_4} } |
| H | { {riv_3} } |

Figure 6: G-decompose operator.

The operator has decomposed each Shape in its constituent components, and therefore now each value of the geometric attribute River-shapes is the simple Shape (i.e. a Shape having only one component) of a river segment that crosses a given region. It can be said that the relation RIVERS-BY-REGION has been normalized by the application of the G-Decompose operator. At the same

time, the value of any geometric attribute is always an element of SHAPES though it may generally be not a simple shape.

We can therefore say that the introduced extensions of relational algebra constitute the basis of a high level geographical database query and manipulation language. The implementation of G-Compose and G-Decompose has been done through the definition in geoSQL of the operators MERGE and SEPARE. The semantics of these operators is of course based on the model defined in [15].

The operator "MERGE [WRT X] [APPLYING $F_Y$ TO] Y FROM R", where square brackets represent optionality, corresponds to our G-Compose. Informally speaking, MERGE composes the values of attribute Y of relation R, possibly with respect to the equality of values of X attribute and possibly appling the set of fusion functions $F_Y$ to the composed set. The operator "SEPARE [WRT X] Y FROM R" corresponds to the G-Decompose of our model. Informally speaking, SEPARE disaggregates the values of set-valued attribute Y of relation R, possibly coupling each of the resulting elementary component of Y with the corresponding value of X attribute. As in standard SQL, relation R may be denoted by nested SQL espressions.

We give now some more complex examples to show how general geographical queries (i.e. queries involving predicates on both geometric and descriptive properties of geographical data) can be formulated by suitable geoSQL statements.

- **Geometrical Selection**: Having at disposal the relation CULTIVATIONS (figure 7) representing cultivations and chemical treatment of pieces of lands, the user can ask the database to "return the names of the rivers whose river segments cross some tobacco cultivated lands".

| CULTIVATIONS | | | | |
|---|---|---|---|---|
| **Cultivation** | **Chemical** | **Cult-quantity** | **Chem-quantity** | **Shape** |
| corn | A | 20000 | 13 | { {A1} } |
| potatoes | B | 10050 | 17 | { {A2} } |
| tobacco | R | 14500 | 23 | { {A3} } |
| corn | C | 37850 | 30 | { {A4} } |
| potatoes | B | 15070 | 20 | { {A5} } |

Figure 7: relation CULTIVATIONS

The formulation in geoSQL is:

```
SELECT Name
FROM
   (SELECT Name, Cultivation
   FROM RIVERS, CULTIVATIONS
   WHERE RIVERS.River-shape INTERSECT
      CULTIVATIONS.Shape )
WHERE Cultivation.="tobacco".
```

Note that the use of INTERSECT predicate (one of the introduced topological predicates, see below) to manipulate SHAPES valued attributes as the join operator has the effect of performing what can be called a "spatial/geometric join" on the involved geographical entities. In fact, its effects are similar, from a logical point of view, to those of the "classical" join when applied to descriptive attributes.

- **Checking spatial differences**: The user can ask the database to calculate the difference in spatial-geometrical terms of the specified geographical entities (e.g. for eliminating slivers following a map overlay) by issuing a query like "return the parts of tobacco cultivated lands, which are not in region R1".

Let us suppose to have extracted from relation REGIONS of figure 8 the relation REGION-1 containing tuples with Name=R1. As well, let us suppose the relation CULT-1, containing the union of Shape values for all tobacco cultivated pieces of land, has been extracted from relation CULTIVATIONS of figure 7.

| REGIONS | | |
|---|---|---|
| **Name** | **Area** | **Shape** |
| R1 | 200 | { {reg_R1} } |
| R2 | 200 | { {reg_R2} } |
| R3 | 250 | { {reg_R3} } |
| R4 | 120 | { {reg_R4} } |

Figure 8: relation REGIONS

Then, the expression in geoSQL language for the above query is:

```
SELECT Shape FROM
   MERGE APPLYING ∩* TO Shape
   FROM
      ((MERGE APPLYING Compl TO Shape
        FROM REGION-1) UNION CULT-1)
```

Further examples, e.g. for processing windowing and clipping of relations, can be found in [15].

To cope with the manipulation of *Geometry*(S) a number of topological predicates and geometric operators have been introduced. Geometric operators are the implementation of the 5 operators introduced in the *Geometry*(S) ADT definition. Topological predicates are used in the WHERE clause of SQL to deal with manipulation of geometrical attributes.

The topological predicates provide a boolean result. In all cases the arguments are two geometric attributes,

which may possibly belong to the same relation.

- INTERSECT: it returns "true" when the regions represented by the two attributes have a non empty intersection; "false" if the two regions have an empty intersection.
- INCLUDE: it returns "true" if the region represented by the second attribute is included in the first one; "false" otherwise.
- ADJACENT: it returns "true" if the two regions are adjacent; "false" if they are not.
- EQUAL: it recognizes when two regions are equal. That is, it returns "true" if the regions represented by the two attributes have the same shape and location in the plane; "false" otherwise.

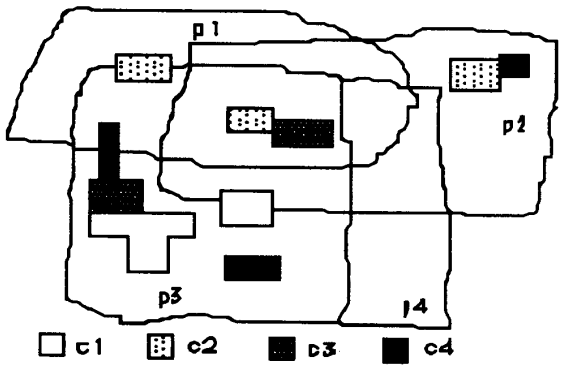
As last set of examples of geoSQL let us show the interactive use of the topological predicates, by showing the graphical results of defined manipulations. We use the following two tables:

| Urban-plans | | | |
|---|---|---|---|
| Code-Area | Use | Year | Shape |
| abc321 | building_area | 1980 | rif_p1 |
| xyz1 | building_area | 1982 | rif_p2 |
| rst55 | building_area | 1985 | rif_p3 |
| www | green_public_area | 1985 | rif_p4 |

and

| Buildings | | | |
|---|---|---|---|
| Civic-Number | Street | Category | Shape |
| 54 | corso_italia | c1 | rif_c1 |
| 21 | piazza verdi | c2 | rif_c2 |
| ... | ... | ... | ... |
| 35 | via roma | c4 | rif_c10 |

whose shapes are graphically represented as it follows:



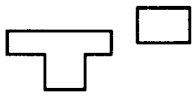

The following query

"Select geometry of all the buildings of category c1 included in the building area of 1985 urban plan"

corresponds to the geoSQL expression:

```
SELECT BUILDINGS.Shape
   FROM BUILDINGS, URBAN-PLANS
      WHERE     URBAN-PLANS.Year=1985 AND
         URBAN-PLANS.Use="Building Area" AND
         BUILDINGS.Category=c1 AND
         INCLUDE(URBAN-PLANS.Shape,
BUILDINGS.Shape);
```

whose result is the following:



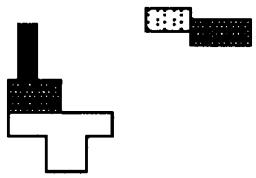
If we want to know

"the geometry and the street of all buildings that are adjacent among themselves and that lies in the building area rst55 of the 1985 urban plan",

we need to formulate the following geoSQL query:

```
SELECT BUILDINGS.Shape, BUILDINGS.Street
   FROM BUILDINGS, BUILDINGS BUILDINGS-1
      WHERE URBAN-PLANS.Code=rst55 AND
         URBAN-PLANS.Year=1985 AND
         ADJACENT(BUILDINGS.Shape,
BUILDINGS-1.Shape) AND
         INTERSECT(BUILDINGS.Shape, URBAN-
PLANS.Shape);
```

obtaining, as far as shapes of buildings is concerned:



## 6. Implementation of the ADT *Geometry*(S)

The implementation of ADT *Geometry*(S) has been done by using the GEOTECH library of Algotech for managing spatial/geometric data. This library see a geometric region as a set of integer points: in our approach, not all the couples of integers which belong to the region itself are explicitly stored. For each region only a limited amount of information is stored and every point of this region can be reached by an appropriate algorithm.

The library employs multivalued quad-trees [5, 2], a direct extension of quad-trees [21]. A multivalued quad-tree is a hierarchical data structure, introduced and analyzed in [5] and refined in [2], for the representation of sets of regions. It is built starting from a regular decomposition

of the plane in quadrants. The plane on which the queries are defined is represented as an array of $2^n x 2^n$ elements. It is recursively split in quadrants and subquadrants until we arrive either at empty quadrants or at quadrants which entirely belong to a (set of) region(s). This decomposition may be represented with a tree, in which every internal node has four children. The leaves of this tree corresponding to homogeneous decomposition blocks are black: to allow the representation of overlapping regions, to every black node is associated a pointer to the list of all regions which share the block. Leaves corresponding to empty areas are marked white. All the other nodes, which are internal nodes, are marked gray. An example of a multivalued quad-tree is show in figure 9.

The simplest implementation of a quad-tree is a representation by a pointer structure; but it needs a large amount of memory, because of the presence of a great deal of pointers. Then, we have adopted a linear structure, executing a pre-order visit of the quadtree and storing the visit order of the nodes in a manner that permit a non ambiguous reconstruction of the tree. This kind of representation is not restrictive because every operation on a tree can be carried out with a visit.

Using such a data structure, the implementation of the first three topological predicates and of the geometric operators is straightforward. The EQUAL predicate needs some discussions. It is necessary to distinguish the case when the two specified regions are values of the same attribute of a table and when they are not. In the former case, if the two regions are equal they have to be considered the same region and not two different entities

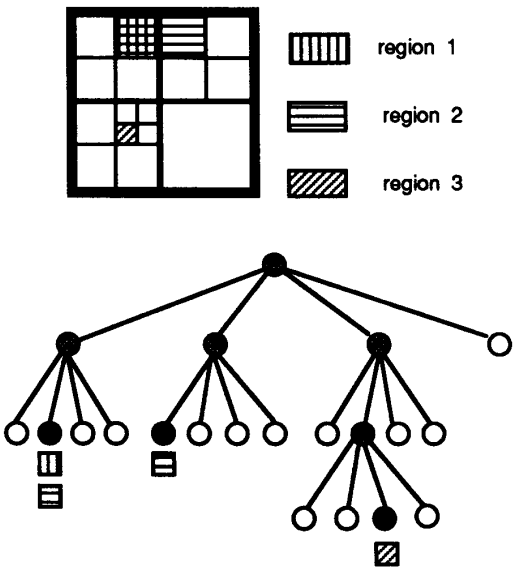



Fig. 9: a multivalued quadtree

For example let us consider the materialization of a view whose schema includes a geometrical attribute. Some regions created during the computation of the view may be identical. If we looked at them as distinct, it would be necessary to store the same information more times with a waste of resources and possible risks of inconsistency . We have to recognize this situation and identify equal regions by same identifiers so that their geometry is stored only once.

The situation is different when the two regions belong to two distinct relations, or they are values of two different geometric attributes in the same relation. The importance of stating the difference between these two situations depends upon the semantics attributed to a single quad-tree. In our approach all the regions belonging to a single attribute of a relation are represented by a single quad-tree. As a consequence, if two regions are stored on different quad-trees they can be equal and their equality can be tested by applying the predicate EQUAL. In this case they can be equal, but not coincident.

## 7. Query resolution

Before the description of the technique adopted for the queries resolution, a premise is needed about the way used to identify the regions in the database. The links between spatial data and descriptive data are maintained by means of numeric identifiers; in the geometric attributes of a relational table, the region is indicated by an integer, that identifies the region among those stored in the same quad-tree. In the quad-tree also, every reference to a region is made in the same way.

In the evaluation of a query, the system firstly verifies the syntactic correctness of the input string. Geometric predicates are then processed: whenever one of these is recognized, the spatial query processor processes it and inserts the results in a temporary relational table. After the processing of all geometric predicates, the query is slightly modified: firstly, the just solved predicates are deleted from the WHERE clause and suitable constraints are added to impose that result tuples belong to these temporary tables; then we temporarily eliminate the references to geometric operators in the SELECT clause and leave only their operands. This modified query is ready for being processed by the relational DBMS, which stores the results of this evaluation in a new relation. This is given back to spatial query processor which can now process the geometric operators. Finally, the system presents in output the requested data.

We can clarify at this point, how the system preserves the consistency of the data. All the regions belonging to a geometric attribute of a fixed relation are stored in the same quad-tree; the name of every file containing a quad-tree is equal to the name of the relation containing the attribute, extended with the name of the attribute itself and completed with the further extension ".qdt". The consistency tie is then the existence of a quad-tree opportunely named for each geometric attribute present in the relations schemes.

We have illustrated a possible strategy for query

resolution. Another possibility is to firstly calculate the relational side of the query and to next use these results for evaluating the spatial side. The choice between the two possibilities can only be made on the basis of considerations depending on the specific instance of the query and on the current extension of the database, which are usually not known a priori. It is reasonable to choose the strategy that minimize the amount of the data to process: but it is not possible to specify a general strategy which is always valid. A further possibility of optimization is to provide an information exchange between the relational DBMS and the spatial processor for decreasing the complexity of the query processing algorithms involved. To this end, an approach which is currently under investigation, is the use of the so-called "on-line" approach to quickly compute partial approximate solutions to spatial queries [14] and to use this information to drive the query optimization process.

## 8. An application to land resource management

In this section we describe an application of CARTECH to the development of a system (TMMS: Thematic Maps and Models Management System [3], [4], [6]) for land resource management and discuss its performances.

TMMS is a system realized to support interactive planning of extraction activities in the "Lazio" region (about 18.000 $km^2$ area), one of the twenty units of local government in which Italy is partitioned. The system manages ten different views of the territory of Lazio: six of them describe land constraints (e.g. urban, archeological, environmental, ...), one contains the hydrological characteristics, one represents the geological aspects, one describes the distribution and potential productivity of mineral deposits, and the last one contains the current distribution and productivity of mines.

The methodology used to define or refine the extraction plan allow to interactively build thematic maps of the land, called "cultivation charts", which describe the regions where mineral can be extracted anyhow, where its extraction is depending to the relaxation of some of the constraints, and where the extraction is anyhow forbidden. Planning is carried out by defining the geographical region of intervention, the amount and type of mineral to be extracted (the objective), the constraints that have to be taken into account and how they have to be combined and prioritized with respect to the objective. If the objective cannot be easily reached, the planner dynamically changes priority values and/or constraints combination to build and evaluate alternative scenarios.

TMMS runs on a 33-Mhz 80386 machine, equipped with the SCO Unix operating system and the INGRES relational DBMS. As an example of its performances we consider some queries which are usually part of the mine planning activities in 'Lazio' region. The queries have been selected among those considered by planners as the critical ones for the currently available systems. In this example we have two relations regarding the entire Lazio region, one describing distribution of minerals (i.e. **MINERAL(name, geometry)**) and the other describing urban constraints (i.e. **CONSTRAINT(type, geometry)**). Geometrical information is represented by two multivalued quadtrees which are 1024x1024 pixels and are 905 and 557 KBytes large.

A first query in the course of a planning section can be to search all zones containing sandstone and not classified as agricultural. To examine performance in detail we first build a table called **NOTAGR** containing all zones not classified as agricultural:

```
SELECT CONSTRAINT.geometry CONSTRAINT.type
   FROM CONSTRAINT
      WHERE CONSTRAINT.type NOT LIKE
         "%agricultural" INTO NOTAGR
```

and then select the zones containing the mineral

```
SELECT INTERSECTION(NOTAGR.geometry,
      MINERAL.geometry) geometry, MINERAL.name
   FROM NOTAGR, MINERAL
      WHERE MINERAL.name='sandstone'
```

The first subquery requires in total 20 seconds and the second one requires in total 1 minute 50 seconds for an overall total of 2 minutes 10 seconds, everything included.

A second query can be to search all zones containing clay and classified as agricultural. Again, to examine performances in detail, we first build a table called **AGR** containing all zones classified as agricultural:

```
SELECT CONSTRAINT.geometry CONSTRAINT.type
   FROM CONSTRAINT
      WHERE CONSTRAINT.type LIKE
         "%agricultural" INTO AGR
```

and then select the zones containing the mineral

```
SELECT INTERSECTION(AGR.geometry,
      MINERAL.geometry) geometry, MINERAL.name
   FROM AGR, MINERAL
      WHERE MINERAL.name='clay'
```

Here the first subquery requires in total 20 seconds and the second one requires in total 2 minute 15 seconds for an overall total of 2 minutes 35 seconds, everything included.

A third query is to build an overall view of extraction zones by intersecting zones containing minerals and zones whose constraint is "extraction".

```
SELECT INTERSECTION (MINERAL.geometry,
      CONSTRAINT.geometry) geometry,
      MINERAL.name
   FROM CONSTRAINT, MINERAL
      WHERE CONSTRAINT.type='extraction'
```

A total of 2 minutes 40 seconds is used in this case.

In the tables below the performances regarding the resolution of only geometric predicates and operators are described. Recall that while predicates just output 'yes' or 'no', operators provide regions as answer. The first table regards the computation of the intersection between regions belonging to map 1 and regions belonging to map 2. The second table is concerned with the computation of how many regions in map 1 are adjacent to at least one region in map 2. It can be noted how the processing time required is 'output sensitive', in the sense that when the expected output is small, then little time is spent during processing.

| INTERSECTION OPERATOR | | | |
|---|---|---|---|
| Involved Regions | | Number of | Average |
| Map 1 | Map 2 | output regions | resolution time |
| 10 | 10 | 40 | 15" |
| 50 | 50 | 24 | 22" |
| 100 | 100 | 13 | 17" |
| 100 | 100 | 40 | 30" |

| ADJACENCY PREDICATE | | | |
|---|---|---|---|
| Involved Regions | | N. of positive | Average |
| Map 1 | Map 2 | region in Map 1 | resolution time |
| 100 | 100 | 24 | 18" |
| 200 | 200 | 19 | 20" |

## 9. Conclusions

In this paper we have presented and discussed a system for geographical data management, CARTECH. The system is based on a logical data model developed as extension of the relational model. It also features an interface based on an extension of SQL to deal in a uniform way with descriptive and spatial-geometric information. CARTECH has been implemented and runs on 80386 machine, under the SCO Unix operating system and interfacing with INGRES database management system.

The usability analysis of the applications developed using CARTECH are encouraging both for the manageability and simplicity of use, and for the efficiency in the treatment of geographic queries. The geoSQL language, introduced for the geographical data manipulation, turns out to be a simple and powerful instrument, being able to satisfy the demands of different application problems. The behaviour of the data structures adopted for the resolution of topological queries seems really good, as it results from the efficiency comparison between our system and the commercial systems with analogous functions and comparable computing power.

The extreme modularity of the system architecture allows to easily extend the data types that it can manage, keeping always separated the manipulation of the descriptive component from the more complex ones (i.e.: three-dimensional data).

Further work is currently being done to enrich geoSQL with more powerful and user-friendly operators, to decrease time and space complexity of physical data structures, and to design query optimization algorithms. An extension of the system for the management of vector data is under analysis and practical experiments in real life applications are also continued.

## Acknowledgments

We are greatly indebted to Paolo Dell'Olmo, Giorgio Gambosi, Marinella Gargano, and Maurizio Talamo for many fruitful discussions on these issues and their contributions to the work here described. Many thanks also to Luigi Barella for his contribution to system design and implementation.

## References

[1] S. Abiteboul, N. Bidoit: Non First Normal Form Relations: An Algebra Allowing Data Restructuring, Journal of Computer and System Sciences 33, 1986

[2] F. Arcieri, L.Barella, E.Nardelli: Multivalued quadtrees for the efficient processing of spatial data, manuscript, 1991

[3] F. Arcieri, G.Gambosi: Ambienti innovativi per la realizzazione di applicazioni di supporto alla pianificazione di insediamenti sul territorio, Conference FAST, Venice, February1990

[4] F. Arcieri, G.Gambosi, M.Lancia, E.Nardelli: Un sistema per la gestione di dati spaziali e di tematismi definiti sul territorio, Conference of the Italian Association for Automated Calcoulous (AICA), Bari, October 1990

[5] F. Arcieri, P. Dell'Olmo: A Data Structure for the Efficient Treatment if Topological Join, Fourth International Symposium on Computer and Information Sciences, Turkey 1989

[6] F. Arcieri, M. Talamo: T.M.M.S.: Un sistema informativo territoriale per lagestione di dati spaziale e di tematismi definiti sul territorio: un'applicazione al Piano Regionale delle Attività Estrattive nella Regione Lazio, Proc. of 1990 Conference on State of the Art in Territorial Information Systems, Italian Chapter of AM/FM GIS, Rome, December 1990

[7] A. J. Baroody, D. J. DeWitt, An Object-Oriented Approach to Database System Implementation, ACM Transactions on Database Systems, Vol. 6, No 4, December 1981

[8] F. Bancilhon: Object-Oriented Database Systems, 7th ACM SIGART- SIGMOD SIGACT Symposium on Principles of Database Systems, March 1988

[9] G. Booch: Object-Oriented Development, IEEE Transaction on SW Engineering, vol 12, no 2, 1986

[10] J. Banerjee, W. Kim, H.-J. Kim, Semantics and Implementation of Schema Evolution in Objects Oriented Databases, Proc. ACM SIGMOD Conf. on the Management of Data, 1987

[11] N. S. Chang, K. S. Fu: A Relational Database System for Images, in Pictorial Information System, Springer, 1980

[12] E. F. Codd: A relational model for large shared data banks, Comm ACM vol.13, No. 6, June1970

[13] C. J. Date: An Introduction to Database Systems, Addison-Wesley Publishing Company

[14] P.G. Franciosa, E. Nardelli: On-line approximation of quadtree border, Int. Workshop on DBMSs for geographical

applications, Capri, May 1991.
[14] M. Gargano, E. Nardelli: A logical data model for integrated geographical databases, Proc. of 1st International Conference on Systems Integration, Morristown, NJ, April 1990.
[15] M. Gargano, E. Nardelli, M. Talamo: Abstract Data Types for the logical modeling of complex data, Information Systems, Vol. 16, N. 6, 1991.
[16] M. Gargano, E. Nardelli, M. Talamo: A model for complex data: completeness and soundness properties, Int. Workshop on DBMSs for geographical applications, Capri, May 1991.
[17] R. H. Güting: Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems, Conf. on Extending Database Technology, 1988, Venice
[18] C. Lécluse, P. Richard, F. Velez: $O_2$; an Object-Oriented Data Model, Rapport Technique Altair 10-87, Septembre 1987
[19] G. Özsoyoglu, Z. M. Özsoyoglu, V. Matos: Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions, ACM Transactions on Database Systems, vol. 12, no. 4, December 1987
[20] N. Roussopoulos, C. Faloutsos, T. Sellis: An Efficient Pictorial Database System for PSQL, IEEE Transaction on SW Engineering, vol 14, no 5, May 1988
[21] H. Samet: The quadtree and other related hierarchical data structures, Computing Surveys, vol. 16, no. 2, June 1984
[22] H.Samet: Spatial database system based on SQL, Proc. VLDB 91, June 1991.
[23] M. Scholl, A. Voisard: Modeling of thematic maps: an application to geographic databases, International Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, July 1989
[24] T.R. Smith, A.U. Frank: Very Large Spatial Database - Report from the Specialist Meeting. J. of Visual Languages and Computing 1 (3): 291-309
[25] Proceedings of First Symposium on Design and Implementation of Large Spatial Databases, Santa Barbara, Ca., July 1989.
[26] Proceedings of First International Workshop on DBMSs for geographical applications, Capri, Italy, May 1991.
[27] T.R. Smith, S.Menon, J.Star, J.E.Estes: Requirements and principles for the implementation and construction of large-scale geographical information systems, Intern. Journal of Geographical Information Systems, 1, 13-31, 1987
[28] Proceedings of Second Symposium on Large Spatial Databases, Zurich, Switzerland, August 1991.
[29] J. Ullmann: Principles of Database and Knowledge-Base Systems, Computer Science Press Inc, 1988