F. Di Loreto, F.Ferri, F.Massari, M. Rafanelli

A PICTORIAL DECLARATIVE QUERY LANGUAGE FOR GEOGRAPHIC INFORMATION SYSTEMS

R. 413 Ottobre 1995

Fabrizio Di Loreto - Collaboratore esterno dello IASI
Fernando Ferri - Istituto di Studi e Ricerche sulla Documentazione Scientifica, del CNR, Via C. De Lollis 12, 00185 Roma, Italy
Tel. (+39 6 44879265 Fax (+39 6 4463836
e-mail: ferri@www.isrds.rm.cnr.it
Fernanda Massari - Collaboratrice esterna dello IASI
Maurizio Rafanelli - Istituto di Analisi dei Sistemi ed Informatica, del CNR, Viale Manzoni, 30, 00185 Roma, Italy
Tel. (+ 39 6) 7716437 Fax (+39 6) 7716461
e-mail rafanelli@iasi.rm.cnr.it

Abstract

In this paper a Pictorial Query Language (PQL) for Geographic Information Systems (GIS) is proposed. The user queries the GIS drawing symbolic objects, combining them together and selecting the derived result among those ones proposed by the PQL. The used interface is part of the Scenario GIS developed using an object oriented environment. This PQL makes easier the formulation of a complex query and simplifies user approach to the system, maintaining a strong expressive power. An overview on the data structure type, on the operators and on the relations among geographic entities is briefly made. The Visual Algebra and the relative operators are defined. The pictorial operations associated to the above mentioned algebra are described. Finally, an example of query and its visual composition on the screen is shown.

Key Words. Pictorial query language, geographical operators, object-oriented query language.

1. Introduction

Due to the inherent complexity of Geographic Information Systems (GIS), powerful and easy to use query languages are required. Powerful, because they have to retrieve information over a very complex database schema (conceptual level), that keeps track of the multiple relations existing in this representation of reality. Easy to use, because the access to stored information, in the GIS token as base for the proposed query language (and already used for resources allocation problems [FRPS 94], [RFMS 95]), should not be limited to a restricted range of yet skilled users, considering the wide spectrum of problems that it contributes to solve.
These two basic requirements, which are of primary importance, even if contrasting, find a common solution in research and development of new user interfaces (for example, pictorial interfaces). In these last years many visual extensions of query languages have been proposed in order to obtain a better user interface for non specialised end-users. The evolution of the query languages used in GIS followed two main directions.
The former has characterised the evolution of the data models to represent information moving from the relational model towards the object oriented data model.
The complex and strongly structured data structures of a GIS are typically hierarchical structures and an object-oriented model allows a more natural approach to define, represent and manage object hierarchies (sub-classing), aggregation hierarchies, composition hierarchies, etc. [CDD 90].
The latter has characterised the evolution of the user interfaces evolved just in the direction of a more intuitive, powerful and flexible approach to the system.
The proposed pictorial query language allows the drowning on his workplace of the objects which play a role in the query. In this manner the

users can express the query semantics simply juxtaposing *symbolic objects* on the screen and expressing in a simple way the topological relationships which constitutes a constraint for the query evaluation.

These objects, to which we refer as *symbolic objects*, represent the generic instances of geographic objects or classes stored in the GIS database. Regarding to the geographic objects we are more interested about their configuration with respect to other geographic objects then their absolute shape. This fact allows us to use very simple symbolic objects to emphasize the aspects that are topologically relevant to represent properties of more complex geographic objects.

Each symbolic object manages the list of its characteristics, which are linked to a geographical object.

The user selects only those characteristics of interest for him in order to use them as selection criteria.

One of the main advantages of this approach, with respect to the traditional ones, is that topological relationships among geographic entities (bordering regions, cities contained in a region, partial or total overlaps among regions or thematic maps) are expressed in a simple manner by the juxtaposition of the objects in the workplace. They touch or overlap themselves one each other (partially or completely).

Since we express query semantic graphically we can classify the proposed language as a *visual language* [BP 94].

Other extensions are introduced with respect to conventional query language; one of these allows the users to perform queries over the conceptual schema of the database. In this manner they can search for an object class which satisfies particular criteria.

Another characteristic consists of its non-procedural modality to carry out queries. Other papers [LC 95], [MAP 90] propose the query composition specifying each time which operators apply to current configuration. In this Pictorial Query Language the user instructs the system in a simple, *pictorial* manner on what he wishes as output, without specifying how the system should proceed to obtain such an output.

In section 2 we present the model of the object-oriented GIS [FRP 94] that underlain the Pictorial Query Language. In section 3 a Pictorial Algebra over the symbolic object alphabet is proposed and topological and logical operators are discussed. In section 4 the proposed Pictorial Query Language is illustrated. In section 5 the internal key-word language is described. In section 6 a simple example of geographical query is shown. In section 7 a brief discussion is made and, finally, in section 8 conclusions are given.

2. The Visual Algebra

2.1 The Object Oriented Model

PQL is based on an object oriented model for geographical data [FPR 95]. The data structures of the model are formally defined in the following.

The base data structure is the *geographic class*. It consists of a set of elements, called *geographic objects* $go_1, go_2, \ldots, go_n$, which have the same properties $A_1, A_2, \ldots, A_m$, are defined on the value sets $D_1, D_2, \ldots, D_m$, not necessary different, and on a set of methods, which are able to describe the behaviour of the object classes.

$D_1, D_2, \ldots, D_m$ are called attribute domains and the methods represent the only way to accede to the value or to the state of the objects (*embedding* principle).

***Definition**:* a geographic class (*gc*) is a quadruple:

<*name, sc, attributes, methods*>

where:

- *name* $\in D$ is the name of *gc*
- *sc* is the class-parent of *gc*
- *attributes* defines the geographic entity properties; it consists of:

- *alphanumeric attribute* which defines the logical structure of the instances; it consists of a set of 2-tuple <*attribute, domain*> (for each object) and of the set of attributes inherited from the relative superclass *sc*. Formally, we have:

attributes ::= alphanumeric attributes (sc)$\cup$

$\{$<attribute, Domain> | $attribute \in A$, $domain \in D$ $\}$

- *geometric attribute*, which defines the abstract data type which represents the geometric structure of the geographical object. Let A be an attribute whose type is *geometric;* its domain can be one of the pre-defined four geometric domains: {*point, polyline, region, null*}. A geometric attribute is defined on a *geometric* domain

- *methods* is the non empty set of methods such that:

methods = { inherited methods from sc} $\cup$ *{methods defined on geoclass}* $\cup$ *{overridden methods}*

Definition*:* The class cardinality is the number of the objects (i.e., instances) of a class.

Definition*:* A geographic class schema (*sc*), or class type, represents the intentional definition of the geographic classes defined on the same set of methods and on the same set of attributes. It extracts the structural properties from it, i.e., the geometric and the alphanumeric attributes, and their definition domains.

Property:

1. A class *sc* exists and it is unique for each geographic class *gc* . Such a class is superclass of *gc*, that is:

$\forall\, gc \in \boldsymbol{C}\ \exists!\, sc \in \boldsymbol{C}$: $gc \quad sc$;.

Definition: A **geographic object** ($go \in \boldsymbol{C}$) is an instance of a geographic class which models a geographic entity of the real world. It consists of a set of tuples

$\{<id_i, val_i>\}$ i=1, 2, & , n; where:

- $id_i \in$ OID (Object Instance Domain) is called identifier of a geographic class instance (go_i); it is unique.

- val_i = val (A_i) is the value of the attribute A_i according to the relative three different types, pre-defined in the schema of the class of which it is part.

The following **properties** are valid for each geographic object:

1. Let $go_i \in O$ be a geographic object. Then:

- $\exists! \, gc \in C: go_i \in ist(gc)$

2. Let $go_i, go_j \in O$ be two geographic objects. Then:

$\forall$ $go_j = (id_j, val_j)$, $id_j = id_i \rightarrow go_j = go_i$

Definition: An **instance of geographic database** is the set of the geographic class instances.

Definition: A **schema of geographic database** is the set of the geographic class schemata.

2.2. The proposed Algebra

In order to give a formal definition of the proposed Visual Algebra, we define the alphabet and the set of operators that act on the alphabet.

The Alphabet **A** consists of a set of three elementary symbolic objects (formally defined in paragraph 2.3), able to represent geometric attribute of geographic objects (point, polyline and region as we have seen in paragraph 2.1).

A= {geo-region, geo-polyline, geo-point}

The set of operators **Op** which acts on the alphabet is given by:

Op={Union, Difference, Disjunction, Adjacency, Inclusion,
Crossing, Overlapping, Equality, Distance}

Notice that some operators described in the following are not applicable to any symbolic object combination because such combinations can have no sense in a generic geographic context.

2.3 The Objects involved in the Visual Algebra

The symbolic objects previously informally introduced have from the topological point of view the same characteristics of the geographic objects or classes which they represent.

> ***Definition*: A *symbolic object*** (**so**) represents the generic instance of a geographic class. It is defined by a quadruple:
>
> so:= <geometric type, objclass, objalias, {set of properties}, position>
>
> where *geometric type* is defined as in paragraph 2 and is the geometric type of the geographic objects that the **so** is able to represent; *objclass* is the class of the geographic objects represented by the **so**; *objalias* allows to refer to this **so** from inside other **so** as we will see in the following; *{set of properties}* are the properties to which user can assign selection expression to select the object instances of the class which should be inserted in the query result; *position* is the position of the **so** which is relative to the workplace that allows to check the topological relationship.

The user formulates his query by a finite set of symbolic objects which represent both geographic objects (with at maximum two dimensions), and classes (like the lake class or the city class, etc.). The user will assign to these objects the wished meaning. The use of a limited number of symbolic objects was preferred in order to guarantee the maximum of flexibility and powerful to the language and a simplicity of use to the graphical interface. The symbolic objects to which it is possible to assign a given semantics are: 1) geo-region; 2) geo-polyline; 3) geo-point. They are graphically represented in Figure 1.

Figure 1

The two dimensional geo-region is used to represent geographical objects or classes which are spatially describable by an area. Instead the geo-polyline represents the objects or the classes in which one dimension is prevalent on the other dimension. Finally, the geo-point represents objects or classes of which the position is the only important spatial characteristics.

From a geometric point of view each geographic object is defined by its boundary and its interior, and the definition, the evaluation and the applicability of the operators is analysed in term of these characteristics.

The definition given in this paper for boundary and interior [CDO 93] [EgFr 91] are slightly different from these ones given in mathematical theory.

Let **G** be a generic **go**; we denote with **∂G** its boundary and with **G°** its interior.

In the case of geo-region objects, we define their boundary as the set of their accumulation points (as defined in the classic set theory).

For the geo-polyline objects we define their boundary as the extreme points.

For the geo-point objects their boundary are the empty set.

The interior of each of the objects is defined as: **G°=G-∂G**.

Another symbolic object which has an intrinsic meaning (the distance between two symbolic objects) is shown in Figure 2.

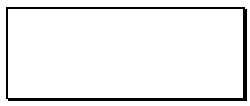


Figure 2

2.4. The operators

In this visual algebra three different categories of operators are defined. The first one is the category of operators which are characteristic of the set theory (union and difference). The second one is the category of the merely topologic operators (disjunction, adjacency, inclusion, equality and overlapping). The last category considered consists of the metric operator (distance).

Now we give a definition for each of them using the following notation

$op : G_i \times G_j := G_h$ (binary operators)

$op : G_i \times G_j \times \ldots \times G_n := G_h$ (n-ary operators)

where $\boldsymbol{G_i}$, $\boldsymbol{G_j}$, . . . , $\boldsymbol{G_n}$ represent the operands (that is the symbolic object previously depicted), while $\boldsymbol{G_h}$ represents the result of the application of the ***op*** operator.

The same operator, applied to different topological combination of the same operands, can give different results. For example, as we will see in the following section 3.2.2, with regard to the *adjacency* operator, a geo-region combined with a geo-polyline can give as result both a polyline, and a geo-point.

2.4.1 The characteristic operators of the set theory

The semantic of this operators is restricted with respect to the classic set theory definition in order to obtain, as a result, symbolic objects belonging to the base alphabet.

Union

Definition: The Union of two **so** G_i and G_j is a new **so** defined as the set of all the points which belong to G_i and/or to G_j. Formally we have:

$$U: G_i \times G_j := G_h = \{x: x \in G_i \vee x \in G_j\}$$

where x is a generic point.

Applicability: the *union* operator has sense when G_i and G_j are **so** of the same type, geo-regions or geo-polylines. In case of geo-regions we require that the **so** have at least one point of their boundary in common. The result obtained is a new geo-region that contain all the points belonging to Gi and/or to Gj. An example of union operator application is shown in Figure 3.

When G_i and G_j are both geo-polyline, to obtain another geo-polyline as a result, we require that their intersection is not empty only on their boundary (the extreme point) (see Figure 4).

Figure 3

Figure 4

The main properties satisfied by this operator are the reflexivity, the transitive and the commutative properties. For the last two properties we have:
$U(U(G_1, G_2), G_3) = U(G_1, U(G_2, G_3)) = U(G_1, G_2, G_3)$.

More in general we have:
$G_1, ..., G_n \in$ GeoObject $\quad U: G_1 \times \& \times G_n = G_h$

Difference

Definition: The difference between two symbolic objects G_i and G_j is defined as a new **so** (G_h) which contains all the points which belong to G_i but don t belong to G_j. Formally we have:

$G_i, G_j \in$ symbolic object

D: $G_i \times G_j := G_h = \{x : x \in G_i \wedge x \notin G_j)$

The result is a **so** of the same type of the first operand.

Applicability: the *Difference* operator has sense:

a) when G_i = geo-region, G_j = geo-region (see Figure 5):

Figure 5

b) when G_i = geo-polyline , G_j = geo-region (see Figure 6):

Figure 6

This operator is neither reflexive, nor commutative, nor transitive.

2.4.2 Topologic operators

The topologic operators considered are the following:

Disjunction, Adjacency, Inclusion, Crossing, Overlapping, Equality.

For this operators we use the classification introduced by [CD 94] which propose a minimal set of topological operators that forms a closed partition over the set of all topological relationships between geo-regions, geo-lines and geo-points. To distinguish these operators it is essential the distinction between boundary and interior.

Disjunction

Definition: Two symbolic objects are disjointed if the intersection of their boundary *and* the intersection of their interior side are null.
$G_i, G_j \in$ symbolic object
Disj: $G_i \times G_j := (\partial G_i \cap \partial G_j = \emptyset) \wedge (G°_i \cap G°_j = \emptyset) \wedge (\partial G_i \cap G°_j = \emptyset) \wedge (G°_i \cap \partial G_j = \emptyset)$

Applicability: Every possible combination of geo-object. This operator is neither reflexive, nor transitive, but it is commutative.

Adjacency

Definition: The adjacency between two symbolic objects G_i, G_j is not null if the points common to the two **so** are all contained in the union of their boundary. If this condition is satisfied, the adjacency between G_i, G_j is a new **so** G_h defined by the set of points common to G_i and G_j.

Formally, we have:

$G_i, G_j \in$ symbolic object
Adj: $G_i \times G_j = G_h = \{x : x \in G_i \wedge x \in G_j \wedge x \in (\partial G_i \cup \partial G_j) \}$

Applicability: The definition of the Adjacency operator requires that all the points which belong to this new object belong at least to the boundaries of one of the operands. This definition covers all possible cases of adjacency between various type of objects and allows us to discriminate between adjacency and particular cases of intersection:

a) when G_i and G_j are both geo-regions and the result is a geo-polyline

b) when G_i and G_j are both geo-regions and the result is a geo-point (see Figure 7)

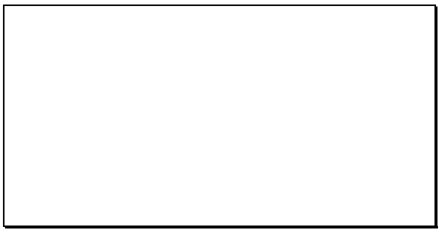


Figure 7

c) when G_i and G_j are both lines and the result is a geo-point (see Figure 8).

In this case the two points of contact are not two internal points, but at least one of these is a boundary point.

Figure 8

d) when G_i is a geo-region and G_j is a geo-point and the result is a geo-point (see Figure 9).

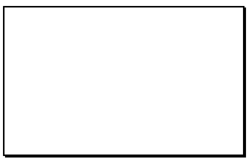


Figure 9

e) when G_i is a geo-region and G_j is a geo-polyline and the result is a geo-polyline
f) when G_i is a geo-region and G_j is a geo-polyline and the result is a geo-point (see Figure 10).

Figure 10

g) when G_i is a geo-polyline and G_j is a geo-point and the result is a geo-point (see Figure 11).

Figure 11

This operator is not reflexive, is not transitive, but is commutative.

Inclusion

Definition: a symbolic object G_i is included in G_j if and only if all the internal points of G_i are contained in G_j.
The result is a **so** G_h which coincides with the first operand. Formally:
$G_i, G_j \in$ symbolic object
Incl : $G_i \times G_j := G_h = \{x$: *for each* $x \in G_h \; x \in G°_i \wedge x \in G°_j \}$

Applicability: This definition does not make any restriction on G_i boundary, so that it must have a non null intersection with the internal points of G_j, but could have a not null intersection also with the boundary of G_j.

The *Inclusion* operator has sense:

a) when G_i and G_j are both geo-regions and the result is a geo-region

b) when G_i is a geo-region and G_j is a geo-polyline and the result is a geo-polyline.

c) when G_i is a geo-region and G_j is a geo-point and the result is a geo-point

d) when G_i is a geo-polyline and G_j is a geo-point and the result is a geo-point.

The properties of this operator are reflexive and transitive, but not commutative.

Crossing

Definition: If the operands of this operator are of the same type, the intersection of their interior sides must be a different type **so**. For example, if they are both geo-polylines, then G_i cross G_j iff they have a single point of intersection on an internal point. In this case an intersection great more then a point is classified as an overlapping between the symbolic objects.

If the operator acts on a geo-polyline and a geo-region, then G_i cross G_j iff the geo-polyline is partially included in the geo-region.

if G_i, $G_j \in$ geo-polyline then

Cross: $G_i \times G_j := G_h=\{!x \in G°_i \cap G°_j \}$

where $\dim(G°_i \cap G°_j)$ is the dimension of the intersection between G_i and G_j (see Figure 12).

Figure 12

If $G_i \in$ geo-polyline, $G_j \in$ geo-region , then

G_i Cross $G_j \Leftrightarrow (G°_i \cap \partial G_j) \wedge (G°_i \cap G°_j \neq \emptyset) \wedge (\exists x \in G°_i : x \notin G_j)$

An example is shown in Figure 13.

Figure 13

This operator is not reflexive, not transitive, and it is commutative only if the two objects are of the same type.

Overlapping

Definition: Two **so** G_i, G_j of the same type, have a not null overlap if both of them, as well their intersection, have the same dimension.

$G_i, G_j \in$ geo-object of the same type (geo-polyline or geo-region)

Over: $G_i \times G_j := G_h = \{x: x \in G_i \wedge x \in G_j\} \wedge \dim(G_h)=\dim(G_i)=\dim(G_j) \wedge (G_h \neq G_i) \wedge (G_h \neq G_j)$

Applicability: The previous definition requires that the operands are of the same type as well as the result of the operator (see Figure 14 a and b).

This operator is reflexive and commutative, but it is not transitive.

Equality

Definition: Two symbolic objects are topologically equal if they are of the same type and have the same shape.

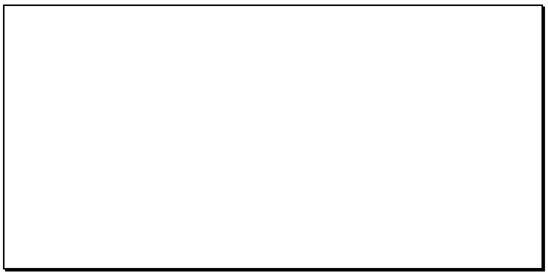
$G_i, G_j \in$ geo-region $\vee$ geo-polyline

Equa: $G_i \times G_j :=\ G_j$ --> if Shape(G_i) = Shape(G_j)

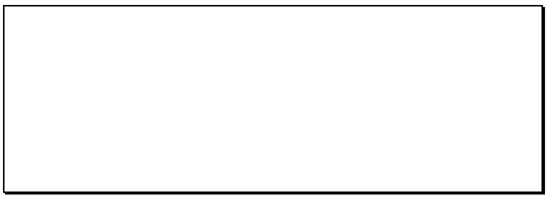
where G_i and G_j are both shape of the same type

:= 0 (no object) in the other cases

shape($G_{i,j}$) is a function which has as result the geometric shape of its argument.



(a)



(b)

Figure 14

Applicability: the *Equality* operator has sense when G_i and G_j are structures of the same type.

2.4.3 Metric Operators

In this section we consider the Distance operator from different topological points of view.

Distance:

The distance between two geo-point objects is the unique case of unambiguous situation. Let us consider, for example, the distance between two geo-point and geo-polyline objets. In this case he have at least two possible distances, maximum and minimum (see Figure 15). The situation is more complicated in the case of distance between geo-polyline and geo-region and between geo-regions.

Figure 15

For this reason we introduce two qualifiers for this operator, ϕ and Θ.
Then, the distance definition is:

Definition: Two **so** G_i, G_j of whatever type have distance ≥ 0 between them if their intersection is null. The distance (δ_ϕ) value is given by

$\delta_\phi(G_i , G_j)_\Theta = G_h$ where

- G_h indicates a bi-oriented segment representing the distance operator between G_i , G_j.
- ϕ is the qualifier which solves the ambiguity above mentioned. Its possible values are {min, max}.

- Θ is a selection expression which includes conventional operators (>, <, =, ≠, etc.) or methods which behaves like operators.

Applicability: to apply the *Distance* operator, the intersection between G_i and G_j has to be empty.

The result of the operation is a bi-oriented segment whose length satisfies the condition expresses by Θ qualifier.

2.4.4 The Logical Operators

Let T_i be a class of objects. According to the Scenario geographical data model, a set of attributes is joined to every object type.

Let a_{i1}, a_{i2}, ..., a_{ini} be the attributes of the i-th object. Each of these attributes has a domain of admissible values C_{i1}, C_{i2}, ..., C_{ini}.

Let t_{ji} be an instance of T_i.

To make a logic selection on t_i implies which we take a generic instance t_i of the class T_i and assign at least a value to one of its attributes.

It is possible to define selection criteria by the classic logic operators (AND, OR, NOT).

In an Object Oriented environment, we consider the objects as if they are instances of new abstract data types.

The methods defined in them form their interface with the external world and, moreover, model the operators which affect these data types.

Classic comparison operators (=, <, >, ≥, ≤, ≠) are available. Beside these we find other operators generated by those methods that behaves like predicates (predicative methods)

Let us examine an example that shows the use of the comparison operators. Given the geo-class State characterised by the attributes: <name, population, boundary state, capital>, the generic class instance is defined by:

{«id_i, values of name, values of population, boundary states, value of capital»};

Introducing the asterisk instead of the unknown values of the attributes, we have:

{«*, *, *, *, Paris»}

In this way we intend that our interest regards the state whose capital is Paris.

Instead, if we have:

{«*, NOT Italy, *, (Austria OR France), *»}

we mean that our interest regards the states whose name is 'not Italy' AND that border with France OR Austria.

The AND operator is represented by the comma between the values of the attributes.

To obtain a better flexibility and expressive power of the query language, we introduce the **alias** reference. By this reference, it is possible, in selection expressions, to create an explicit link to the properties of other symbolic objects. For example, let us consider the generic city class instance defined as:

{«idj, values of name, values of position, values of population»};

If *al1* represents an "alias" for a symbolic object O_1 of lake class, we are able to refer to any properties of O_1 from any other object specifying simply $al1.property_i$:

{«*, Rome, *, *, 2.500.000;»} <-- *al1*

and to use the alias *al1* in any other object instance.

For example, writing:

{«*, *, *, *, >al1.population»}

we mean that our interest regards the cities with more inhabitants than Rome.

When the user points out the query target, he specifies some aggregation functions (MAX, MIN, AVERAGE, COUNT etc.) with the usual semantics.

3. The Pictorial Query Language

The pictorial query language proposed in this paper is based on a set of elementary geometric objects (symbolic objects), which represent geographic classes and objects. In this manner, complex queries can be expressed by simple and intuitive pictorial operations. Moreover, the user has not necessity to learn a query language syntax and semantics to carry out a correct query to the system.

3.1 The queries

Using symbolic objects as a representation of geographic classes the queries are "composed" putting graphically in evidence the spatial objects (symbolic objects as their substitutes) which are involved in it and the topological relationships which link them.
Then, the semantics of the query is expressed drawing geometric figures. Two distinct levels of semantic interpretation exist. The former is the semantics assigned to the algebra operators, the latter is due to the implementation of the class methods which customise the basic behaviour of methods instanced to solve the query.
The queries which can be formulate on a geographic database express two types of geographical object properties:
- *spatial properties*, as, for example, the topological relationships or the distance; these properties are verified by the geometric attributes of the different objects which form the database.
- *descriptive properties*, as, for example, the population of a given region, or the name of a given river. These properties are verified by the alphanumeric attributes of the above mentioned objects.

We define a query as a set of seven subsets:

$$Q=\{D, O, R, S, T, H, E\}$$

where:
D = a not empty set of databases (generally more than one) which form the active domain of the current query. All the geographical object instances and classes involved in the query by some symbolic objects are searched in these databases.

O = a not empty set of symbolic objects, created directly by the user, which are involved in the query. Each symbolic object has a set of characteristic properties, some of which are devoted to represent the object state (position, colour, etc.&), others are devoted to represent the characteristics of the geographic classes and objects which the symbolic object represents.

R = a set of relationships (set-type, topological-type or metric-type), eventually empty, among symbolic objects belonging the sets **O** or **S** (defined in the following). Each relationship is expressed by:

$obj_i = r_i(arg_1, arg_2, ..., arg_n)$.

where r_i is the selected relationship, $arg_1 ... arg_n$ are the symbolic objects involved in the relation evaluation and obj_i is the resulting object.

S = a set of symbolic objects generated as a result of the relationship in **R**.

T = a not empty set of symbolic objects belonging **S∪O** which form the current query target.

H = a not empty set of type declaration which links each symbolic object (enclosed in **S ∪ O**) to a class of the class hierarchy rooted to the symbolic object data type. This set, associated with set **D**, allows the system to make a static type check (before running the query) on the selection expressions associated to the symbolic object fields. Each type declaration is expressed by:

symb_objectname.**type =** memberclass.

E = a set of selection expressions associated to symbolic object fields which form the selection criterion to choose the object instances which satisfy the query target from the databases which are enclosed in **D**. Each selection expression has the following form:

symb_objectname.**property**.propname = expression

The system, using this information, carries out the query translation from the visual representation to a key-word expression.

3.2 Pictorial Operations

Let us present now the most relevant operations defined in the Pictorial Query Language (PQL):

" databases selection
" choice of the type of a symbolic object to create
" creation of a symbolic object
" dismissing of a symbolic object
" dragging of a symbolic object
" reshaping of a symbolic object
" generation of new symbolic objects by set-type, topological-type and metric-type relationships
" selection of a symbolic object as the target of the following visual operation
" topological relations selection
" grouping and ungrouping of symbolic objects
" assignment of *member classes* to a symbolic object
" assignment of selection criteria to a symbolic object
" query target definition
" results of visualisation strategies

The final result of these operations is a string of the procedural key-words query language which is independent from the operation sequence. The creation of a symbolic object involves not only a new object class, such as instances of the query, but also generates symbolic objects to represent topological relationships among the existing objects. The results of topological operators are generated automatically and dynamically by the system. Every operation which modifies current topological relationships involves the creation, the deletion or the modification of objects to maintain a sound and update representation.

Let us show it by an example (referred to Figure 16). First, we select the geometric object type; second, we create the symbolic object, that is, two disjointed geo-region shapes (A and B). By a dragging operation we put the objects as shown in Figure 16, obtaining a non empty intersection. The system, automatically, generates results for the topological operators which are

applicable on this configuration. The result consists of four new objects (1, 2, 3, 4), highlighted by different patterns.
The object 2 is the result of the overlap operator between A and B, the objects 1 and 3 are the result of a subtraction (i.e., 1=A-B, 3=B-A) and the object 4 is the result of the Union operator between A and B.
If we create another symbolic object C and put this object inside the object 2, the object C gives rise to an inclusion relation with the object 2 (and also with A and B, because the inclusion relation is transitive). Further dragging operations on B, which restore A and B disjoined, remove the objects 1, 2 and 3, meanwhile the object C maintains its inclusion relationship with A.
Both the objects explicitly generated by user, and the objects automatically generated by the system, has got a list of attributes which characterise each class. On these attributes we can assign the selection criteria in the way described in the previous section.

Another operation which contributes to provide expressive powerful to the language is the grouping operation. This operation allows to use a single **so** to represent a set of **so** (of the same type) disposed in different topological situations (more details are given in paragraph 3.3.9).
The last operation, needed to compose a sound query allows to point out the query target. The user has to select the symbolic objects, of which the member classes represent the query target, and among these objects he has to mark the attributes of interest.
At the end of these operations the system produces a string of syntactically correct instructions, which corresponds to the graphical query.

Figure 16

This visual operation doesn t have any influence on the query set.

3.3 Description of the most relevant pictorial operations

Let us analyze more in detail the above mentioned pictorial operations.

3.3.1 Databases selection

The selection of the databases of interest specifies the domain of interest for the current query. This operation defines univocally the set of geo-objects which the query can consider simile to an instance. All the selected databases form the set **D**. User can specify if the current query involves only the current selected databases or also all the databases rooted at the current set.
The definition for this operation is the following:

DbSel: {Set of Databases} -> **D**

3.3.2 Choice of the object type to create

This operation allows to choice among the various objects which could be included in the current query, that is, if the symbolic object has to represent a geographic object with zero, one or two dimensions.

This operation does not create symbolic objects. It can be seen such as a type declaration by which the user notify to the system the object type he is going to create (see Figure 17).

Figure 17

The definition for this operation is:

ObjectTypeSelection: {geo-point, geo-polyline, geo-region} -> a single element of the argument set.

3.3.3 Symbolic object creation

After the definition of a new symbolic object type, we create a new instance of this object type and place it in the desired position on the workplace in order to generate a specific topological configuration. The type of the new object, is defined (as default) as the root of the hierarchy of this geo-object type (geo-point, geo-polyline, geo-region).

This operation inserts an element in the set **O** and in the set **H** (for each element in **O** we have a type definition in **H**).

The formal definition is the following:

ObjCreation: geometric type $\times$ ObjName -> SymbObj

where geometric type is defined in paragraph 2.

This new symbolic object is inserted in the set **O** as a new object created by the user and it is also the currently selected object, so that many of the following operations act directly on it.

3.3.4 Symbolic object selection

Many of the pictorial operations defined in the following, operate on a symbolic object or on a group of symbolic objects.
The selection operation allows to specify on which, of the created object(s), are applied the following operations.
The definition is:

ObjSelect: $\mathbf{O} \cup \mathbf{S}$ -> currObj

where currObj is the symbolic object, selected in the set **O** or in the set **S** (these sets contain the symbolic objects involved in the query), which is target of the visual operations which follow it.

3.3.5 Member class assignment

This operation overrides the default type definition derived by the object creation and allows to define exactly and in a univocal way, with which class we are dealing with.
After this operation the system knows exactly which geographic class defined in the database set **D** is represented by the symbolic object and naturally all attributes and methods characteristic of this class. Some of these methods should be considered as predicative methods (a method which result is only true or false) and then used as predicates in selection expression. This operation changes the type declaration associated to the currently selected object contained in set **H**. In this manner system can dynamically keep track of the current objects type definition (see Fig. 18).
Formally we have:

ObjMemberClass: so $\times$ geometric type $\times$ **D** -> GeoClass

The arguments GeoType and **D** allows the system to retrieve all the correct class definitions which should be involved in current query.

Figure 18

3.3.6 Symbolic object dismissing

The symbolic object can be dismissed by the user or automatically by the system. For example, if symbolic object involved in one or more relations is dispose, the system, in order to maintain a sound situation, automatically dispose all those objects that don't have any more reasons to exist.. This operation causes the elimination of the object from the set **O** and eventually from the set **R** and modify also the set **H** (where we find the type declaration of each symbolic object involved in the current query) and the set **E** (which contains the selection expression associated to the object).

The formal definition is the following:

ObjDismiss: so ->none

where so denotes the selected symbolic object.

3.3.7 Object Dragging and reshaping

These operations allow to operate on object(s) selected by the previously described operation.
By these operations, user can change relative position of the object displayed on the workplace, changing the topological relationship existing among objects.

3.3.8 Generation of new object derived from the topological operators

When we applied the topological operators to the existing objects, also if the configuration is simple, for example, to geo-region objects, the system
The system computes all the possible applicable operators to the displayed configuration, but derived object are created only if needed for the query composition.
The member class of the new object will be set to the highest level in the classes hierarchy rooted at geo-region so that it represents all the possible instances, as generic as possible, of this class.
This operation acts on the set **R** adding new relations, and on the set **S** which contains the object generated by these relations.
This operation derives directly from the application of the operator which acts on the geo-objects. Its definition coincides with the definition of the various operators previously examined.

3.3.9 Grouping operation

The grouping operation contributes to add a great expressive power to the visual language. By this operation we use a single symbolic object standing for an entire group of objects (all belonging to the same class). The relation and the selection expression sets on the object group, involve in the same manner all the objects of the group.
A typical example of query, in which this operation reveals its usefulness is the following: "find all mountain communities that extend their territories, also partially enclosed in region X and are far less then 50 Km from a generic city".

A possible representation using grouping functionality is shown in Fig. 19.

Figure 19

This operation groups topological properties of the various component objects, allowing the application of the distance operator only to the object group and not to all the possible combination of the component objects.

Each group is treated as a new single object. This operation modifies the same set which was modified by the operation of object creation.

The formal definition is:

$$\text{Group: } \{\text{set of object} \in (\mathbf{O} \cup \mathbf{S})\} \text{ -> so}$$

This new symbolic object is inserted in set **O** as a normal object created by user.

3.3.10 Selection criteria assignment

By this operation the user defines the condition that each geo-class instance has to verify (associated to symbolic object) in order to obtain by the current query an admissible result.

Each symbolic object involved in the query has its own type declaration (set **H**) which specifies exactly its class composition in term of methods (function type declaration) and attributes (type declaration).
By this information the system can at any time perform a static type checking on the expression introduced. The introduction of a new selection expression affects the **E** set.
Formally we have:

ObjSelExpr: currObj.propName -> selExpr

3.3.11 Query target definition

This operation is very important because it allows to specify the research target. The target i consists of one or more symbolic objects which user can select.
Using selection criterion user can specifies to the system to which fields of the object he is interested.
This operation, influences the set **T**.
A formal definition for this operation is given by:

TargetDef: {set of obj $\in$ (**O** $\cup$ **S**)} -> **T**

3.3.12 Results visualisation strategies

User can select among several visualisation strategies to display the results of his queries. The main aim is to combine successive query results so that user can display only the object which are important for his query strategy.
Some of these possibilities are the following:

" NEW: this criterion allows to create a new database using the current query results. In this manner is possible to create particular views of the current database.

- ADD: it updates current database adding the results of the current query. It allows to concatenate results of different queries, using the results of the first query as domain for the next one.
- REMOVE: it updates current database removing the current query results.
- INTERSECT: it highlights the common objects between current database and current query results
- HIGHLIGHT: it puts in evidence current query results

The formal definition of the strategy visualisation is:

VisStrategy: {set of possible strategy} -> currStrategy

4 From the visual query language to the keyword language

The next step is to translate the pictorial query in to language that the OO-DBMS processes. We will use a language SQL-like with four clauses:

SELECT:

In the SELECT clause we insert the targets of the query which are one or more objects and, eventually the values of some of their attributes, that is, a projection on certain attributes.

FROM:

In the FROM clause we define the domain of the query (one or more databases). It is also possible to add the qualifier **limited,** that operate a restriction of the active search domain only to the specified classes and not also to all the class hierarchies rooted to these

WITH:

In the WITH clause we describe the known value properties of the objects involved in the query.

WHERE:

In the WHERE clause we insert all the topological relationship between the objects and classes that lie on the query desktop.

Now we see how the actions of the visual language correspond to the introduction of new lines in the four clauses of the key-word language. We are assigning a semantic to our query by the pictorial primitives. Here we present only some of pictorial primitives.

Creation and dismissing of symbolic objects

The creation of a new symbolic object produces the insertion of an empty object descriptor in the FROM clause. In fact, a descriptor specifies the member class of the object instance and when the user inserts the new symbolic object, the member class is not yet assigned to the object.

Sometimes, creation and dragging of symbolic objects implies the generation of new symbolic objects by topological relationship. This generation causes to the insertion of further object descriptors in the FROM clause and the insertion of the topological relation type introduced in the WHERE clause.

The symbolic objects Deletion cancels its descriptor and all the relations in which it is involved in the WHERE clause.

Assignment of member class to symbolic objects

This operation modifies the object descriptor in the WITH clause and implies a recomputation of type checking on the selection criteria connected with the attributes, as well as an object descriptor specifies the member class of the object.

Drag and Reshape of symbolic objets

These operations have an influence mainly on the WHERE clause. In fact when the user drags or reshapes a symbolic object over the desktop, he creates or removes the symbolic objects which were the result of topologic relations (also modifying the WITH clause) between the objects in the previous topologic configuration.

Then, the WHERE clause is modified respect to the new topologic configuration.

Assignment of selection criteria to symbolic objects

This operation influences the FROM clause; in fact, it allows to specify which object instances must be considered in the query evaluation.

Query Target definition

This operation determines the SELECT clause. If a limited number of attributes of the symbolic object that represents the query target are specified, the result of the query will be projected on these attributes. If some aggregation function are specified in the selection criterion expression (*COUNT*, *MIN*, *MAX*, etc.), those functions are transferred in to the SELECT clause.

4.1 The conversion algorithm

Now we describe step by step the conversion algorithm from the pictorial query to the keyword query.

The first step is the examination of the topologic relation among objects.

For each symbolic object on the desktop we ll list all its topologic relations with other objects. At the end of the examination we ll have fill the where clause.

In the second step we consider the member classes and the selection criteria regarding the symbolic objects. This step is the real mapping from the symbolic objects to the geographic objects. This mapping creates the with clause.
The last step is the specification of the target that is the selection clause.
We give a formal description of the algorithm using a Pascal like language. Some procedure will be completely explained and for other procedure (for shortness) we ll give only the input and the output parameters.

Procedure conversion
begin
    Where-clause;
    With-clause;
    Select-clause;
    From-clause
end.

Procedure Where-clause
begin
    for i:=1 to number of symbolic objects do
    begin
        for j:=i+1 to number of symbolic objects do
        begin
        Find-relation (object(i), object(j), object(r), relationtype);
        if object(r)≠NIL
        then
            if (object(r)≠object(i)) AND (object(r)≠object(j))
            then
                Write-statement2 (object(i), relationtype,
        object(j), object(r))
            else

```
                    Write-statement1 (object(i), relationtype,
          object(j))
          end
     end
end.
```

Procedure Find-relation
Input parameters: two objects
Output: the relation between the two objects , the new object that is the result of the relation. If the relation is disjunction then the generated object is NIL.

Write-statement1
Input parameters: two objects and a topologic relation
Output: the statement of the where clause:
object(1) relation object(2).

Write-statement2
Input parameters: three objects and a topologic relation
Output: the statement2 of the where clause:
object(r) = object(1) relation object(2).

Procedure With-clause

```
begin
     for i:=1 to number of symbolic objects do
     begin
     Assign-member-class (object(i), class-name);
          for j:= 1 to number of selection criteria do
               begin
               Read-criteria (object(i), criteria)
               criteria (j) : = criteria
               end
     end
```

```
        Write-statement3 (object(i), class-name, criteria)
end.
```

Procedure Assign-member-class
Input parameters: one object
Output: the member class of the input object

Procedure Read-criteria
Input parameters: one object
Output: one selection criteria applied on the object .

Write-statement3
Input parameters: one object, one class-name, one array
Output: the statement3 of the with clause:
object /type (class-name)/property: (name = parenthetic expression), (name = parenthetic expression),...

Procedure Select-clause

```
begin
        for i:=1 to number of target objects do
        begin
                for j:=1 to number of target attributes do
                begin
                        Project (object(i), attrib);
                        attribute (j) : = attrib
                end
                Write-statement4 (objet(i), attribute)
        end
end.
```

Write-statement4

Input parameters: one object, one array
Output: the statement4 of the select clause:
object. target attribute, target attribute, ...

We don t consider the From-clause procedure because it is trivial.

4.2 An example of transformation from visual language in keyword language

Let us show the transformation by the examination of a simple example:

Find the cities of Lazio which are far more then 100 km from Rome

In the visual language the following could be the query representation:

Figure 20

The objects created are: P_1, P_2, and $Region_1$: that is three insert operations that create three empty object descriptors (P_1, P_2, $Region_1$) with member classes not yet specified and so imposed at the most generic level.
The application of the distance operator between P_1 and P_2 specifies remaining information.
From the position of the symbolic objects, the system knows, at the topologic level, that P_1 and P_2 are included in $Region_1$ and between P_1 and P_2 is applied the distance operator. The topological relations are included in the WHERE clause:

statement1: P1⊂Region1 ---- > topologic operator
P2⊂Region1 ---- > topologic operator

On P_1 and P_2 we have the statement1 because the result of the include operator is the included object. In the Where-clause Procedure: object(r)= object(j)) .
Instead, in the case of the distance operator which result is a bi-oriented segment we have:

statement2: D(P_1*,*P_2*)>100* ----> topological operator

The assignment of member classes specifies the type of the object represented by the symbolic object lying on the desktop, and the selection criteria specifies the known values of their attributes. These are the arguments of the WITH clause:

Region1/ type (Region) ----> assignment of member class
property:(name = Lazio) ----> *selection criteria*

statement3: *Region1/ type (Region)/Property:(name = Lazio)*

The query target is the object P_1 member of the class City and in particular we are interested in a specific attribute: name. Then we project the object P_1 on the attribute name. In this way we have the expression:

statement4: P1.name. ---->description of the query target

In the keyword language we have:

SELECT	P1.name
FROM	Current database
WITH	Region1/ type (Region) / property:(name = Lazio),

P1 / type (City),
P2 / type (City)/ property (name=Rome),
D/ type (bi-Or. Segment)/ Property (m.u.= "Km")

WHERE P1⊂Region1,
P2⊂Region1,
D(P1,P2)>100

END.

5. Query Examples

Let us consider the following queries:

Query 1

"South American countries whose territory is deforested for an extension>7% of the total extension of the state, and whose territory is covered by forest for an extension>20% of the total extension of the state, in which indigenous communities with a number of men >25 lives".

First of all we generate the symbolic objects standing for the objects involved in the query. The thematic maps are modelled by a rectangular shape. The indigenous communities are modelled by geo-points. If we suppose to have a database in which only data regarding South America are stored, we need three geo-regions A, B, C. The first one, A, represents the generic instance of the **country** class. B and C, instead, represent the generic instance of the thematic maps.

We define the object A (the country) as the target of the query. We don t specify any selection criteria regarding the countries, because we are looking for all the instances which satisfy the query. We mark only the attribute "name of the country to specify that we need only the country name.

Besides, to specify the selection criteria, we need to place properly the symbolic objects involved in order to specify the topological relationships existing among these objects. We place A, B and C in order to not generate a common overlapping ABC, but two distinct overlapping area AB and AC. On these geo-region classes we set as selection criteria the extension of the areas with values >7% and >20% of the country extension . To set these criteria we use the future of alias references.
Then we create a geo-point representing the *indigenous communities* class instances and place these instances inside the new symbolic object AB.
The community position inside AB implies that these communities should be placed in a country region which is covered by forest.

The final representation on the screen is shown in Figure 20. The query expressed in a language SQL-like generated by the system is the following:

```
SELECT      ALL A.name
FROM
        database := South America
        A.type = state
        P.type = community
        B.type = Thematic map
        C.type = Thematic map
        AC.type = Region
        AB.type = Region
WITH
        B.property .theme = "deforested area"
        C.property .theme = "forested area"
        A.property .extension = a1
        AC.property .extension >7% a1
        AB.property .extension >20%a1
        P.property. population >25
```

WHERE

AB = A overlapping B,

AC = A overlapping C,

P1 contained in AB

Finally we present a different type of query, in which the target is an information about the structure of the database instead of one or several data stored in the database.

Query 2

"Classes characterised by attributes with all alphanumeric values".

The result of the query is a set of classes.

First of all we should generate the symbolic object standing for the class involved in the query.

We need one geo-region A that represents the generic class.

In this case the type of the class is "undefined" and the selection criteria refers to the attributes which characterize the class and not its values.

We write this selection criteria in the following way:

Type attributes values = alphanumeric

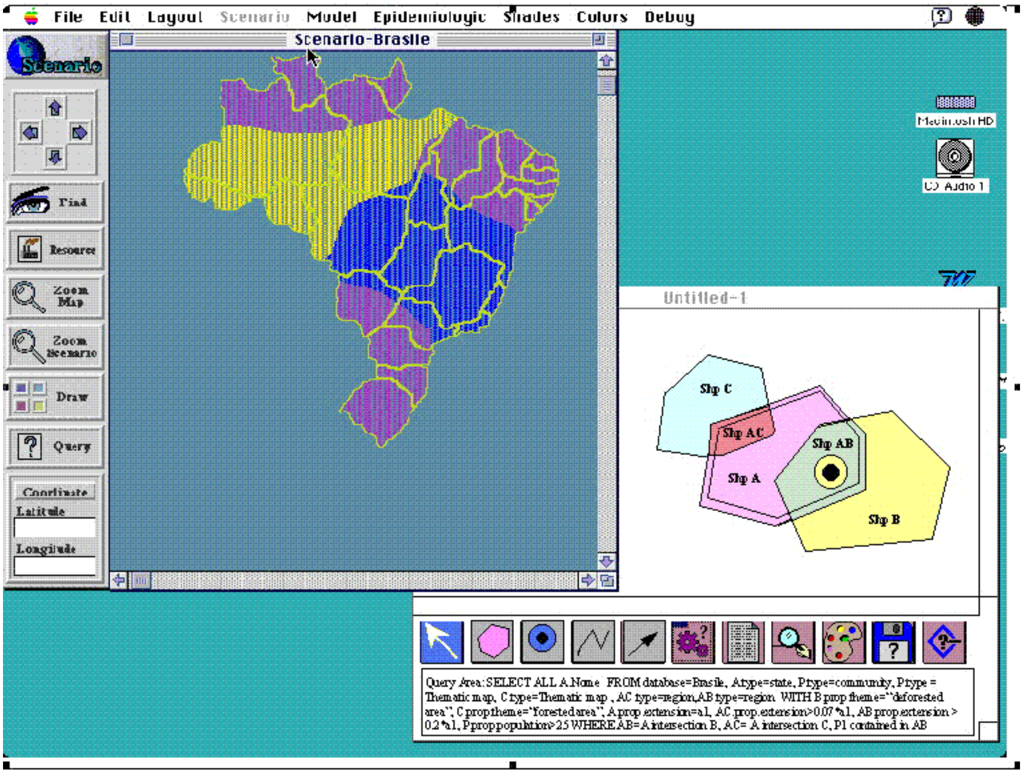


Figure 21

Now we define the object A as the target of the query. We mark the "type" of the class to specify that we need the class types.

SELECT ALL A.Type
FROM
database := whole
A.type = undefined
WITH
.ALL Domain (A.property) = alphanumeric
END

Since we wish the user uses again the result of a query as entry point of another query, the result can be considered as a subset of the database on which he can express another query.
This kind of approach allows a great modularity in a query expression, because user can specify query target with an increasing of detail level.

6. Discussion

The proposed PQL uses strongly the object-oriented paradigm and its peculiar characteristics. This fact allow to the authors to implement a tool characterised by a strong flexibility which helps the user because he has to not learn the generally complex syntax and semantics of a procedural query language.
One of the main peculiarities which characterises this PQL with regard to other proposals (for example, [LC 95], [MAP 90] and [EG 92b]) is the possibility to do query at extensional level, that is query on the database schema. The non procedurality of the query composition method is another facility of the query language which allows the user to specify only a minimal data set. Other information, such as, for example, information regarding the topological relations among the objects drawn on the workplane, are automatically defined by the system (this functionality is present only in [LC 95]. The object-oriented paradigm used allows to use only three symbolic objects to represent any geographic object: the system resolves all the ambiguities regarding the operators and their semantics, depending on the classes on which they work.

7. Conclusions

The model presented in the previous pages has been used as a starting point to project and develop a first implementation of a *Pictorial Query Language.*
All the choices made during the project development were directed toward a better and more intuitive user interface without sacrificing the overall expressive power.
The main results gained are:

" An high expressive power, because user composes pictorially his query and expresses in this manner all the topological relations among the geographic objects involved. User points his attention on symbolic objects and on their topological relationships whatever geographic class or object it represents.

" Query semantics is also assigned pictorially, and this fact qualifies this query language as a *visual language* [BP 94]. Besides, since we are in an object oriented context, the exact semantic and calculus method for each operator are established at runtime and depend on which classes is applied the operator. This feature allows the system to solve, without any further specifications, problems like the semantic difference between the belonging of a lake to country or the belonging of a country to another one (e.g. San Marino in Italy). This two situations have distinct semantic although, visually, they are represented in the same manner.

" Non-procedurality allows to devote the user attention to what he wants and not how to obtain it. Many of the previous proposed works in this field require that user specifies step by step the operators to apply to the composing operands. Our prototype proposes for each situation a list of possible operators applicable to the selected symbolic objets.

" The *alias* technique allows to compose very complex queries in a user-friendly environment.

" The introduction of a conversion algorithm towards an SQL-like query language makes the tool interesting for a more deepest analysis regarding its possible interfaces with several commercial OO-DBMS.

This prototype has been developed on computer Apple Quadra 800 in an object oriented environment, using the Object Pascal language and the MacApp Class Library.

Bibliography

[AN 92] F. Arcieri, E. Nardelli: "An Integration Approach the Managment of Geographical Information" CARTECH. ICSI 92 (International Conference on System Integration), 1992.

[ANS 89] American National Standard Institute (ANSI).X.3.135-1989 "Database Language SQL'89"

[ANS 91] American National Standard Institute (ANSI). X.3.H2-91-183 "Database Language SQL2/SQL3 '91"

[CH 76]D.D Chamberlain, Astrahan, Griffiths et al: "Se-quel 2" IBM Journal of research and Development 20 (6), 1976.

[BES 77] R. Berman, M.Stonebraker: "GEO-QUEL: a system for the manipulation and display of geographic data" ACM Computer Graphics Vol 11, pp.186-191, 1977.

[BP 94]C. Bauzer Medeiros, F.Pires: "Databases for GIS" SIGMOD Record Vol 23, n°1, March 1994.

[CD 94] E.Clementini, P. Di Felice: "Topology in Object-Oriented GIS" Second national congress Sistemi Evoluti per Basi di Dati, Rimini, Italy, 6-8 June 1994

[CDD 90] E. Clementini, P. Di Felice, A. D Atri: "An Object Oriented Conceptual Model for the Rappresentation of Geographic Information" Technical Report n. 2-90 Dipartimento di Ingegneria Elettrica, Università dell Aquila, 1990

[CF 80]N.S. Chang, K.S. Fu: "Query by Pictorial Example" IEEE Transaction on Software Engineering, SE-6, pp.519-524, 1980

[EG 92a] M. J. Egenhofer: "Why not SQL!" IJGIS, vol.6(2) pp. 71-86, 1992.

[EG 92b] M. J. Egenhofer. "Spatial SQL: a query and presentation language" IEEE Transaction on Knowledge and Data Engineering. 1992.

[EGF 88] M. J. Egenhofer, A. Frank: "Toward a Spatial Query Language: User Interface consideration" Proceedings of 14th VLDB Conference, Los Angeles, CA, 1988.

[FRPS 94] F. Ferri, M. Rafanelli, E. Pourabbas, G. Sindoni: "GIS and Health Resources Planning Problems" Int. Conf. EGIS-MARI '94, Paris, 1994

[FPR 95] F. Ferri, E. Pourabbas, M. Rafanelli: "Scenario: an object-oriented model for spatial data" Tech. Rep. IASI (in press).

[FRA 82] A. Frank: "MapQuery-Database Query Language for retrivial of geometric data and its graphical rapresentation" SIGGRAPH ACM Computer Graphics, July 1992.

[JOC 88] T.Joseph, A.Cardenas: "Piquery:a high level query language for pictorial database managment" IEEE Transaction on Software Engineering, vol. 4, pp.169-179, 1988

[LC 95]Y.C. Lee, F.L. Chin: "An Iconic Query Language for Topological Relationship in GIS" IJGIS vol. 9, N.1, pp 25-46, 1995

[MAP 90] M.Mainguenaud, M.A.Portier: "Definition og Cigales: a GIS Query Language" Int. Conf. DEXA '90, Springer Verlag Publ., 1990.

[RB 91]J.F. Raper, M.S. Bundock: "Implementation of a user environment for a spatial DBMS" UGIX Project.

[RFMS 95] M.Rafanelli, F.Ferri, R.Maceratini, G.Sindoni; "An object oriented decision support system for the planning of health resource allocation" Computer Methods and Program in Biomedicine, vol. 48, pp. 163-168, 1995

[RFS 88] N. Roussoupolos, C. Faloutsos, T. Sellis: "An efficient Pictorial Database System for PSQL" IEEE Transaction on Software Engineering, SE 14 (5), pp. 639-650, 1988.

[ST 84] M.Stonebraker et al: "Quel as a data type" Proceed. of the ACM-SIGMOD Int. Confer. of Management of Data, ACM, 1984

[WS 93] W.J. Weiland, B. Shneiderman: "A Graphical Query Interface Based on Aggregation / Generalization Hierarchies" Information Systems, Vol.18, n°4, pp 215-232, Pergamon Press.Publ., 1993

[ZLO 75] M.M. Zloof: "Query By Example" Proceed. AFIPS '75, NCC Vol. 44, pp 431-438, 1975