DEFINITION OF CIGALES†: A GEOGRAPHICAL INFORMATION SYSTEM QUERY LANGUAGE

Michel MAINGUENAUD

Institut National des Télécommunications
9, rue Charles Fourier
91 011 EVRY
email: MAINGUENAUD@FRINT51.BITNET

Marie-Aude PORTIER††

Paris VI University - Laboratoire MASI
45, avenue des Etats-Unis
78 000 VERSAILLES
email : portier@zeus.ibp.fr

Abstract

In this paper we present a graphical query language for Geographical Information System. This language is based on a graphical Query-By-Example-like (QBE) philosophy. A set of graphical primitives (icons) represent data or operations such as inclusion, intersection, etc. The user-defined query is made by composition of these icons. The application of the icons defines the query as the data are supposed to be. This graphical query is then transformed into a functional-based-language query. This expression is compiled into specific Data Management System orders or graphical operators. The main contributions of this language are its simplicity to express a query and its representative power.

Keywords : Geographic Information System, Data Manipulation Languages, User-Interface

1 - INTRODUCTION

Nowadays, Geographical Information System (GIS) need large investments in time and human resources. Various domains such as urban planning, remote sensing, military organizations, travel agencies, etc use geographical data.

A Geographical Database is a collection of two typed data : 1) spatial data (generally referenced as point, line and area) and 2) alphanumerical data (i.e. names of towns, population). To manipulate these data, GIS designers have to define a new Data Manipulation Language (DML) which can take heterogeneous data types into account. Numerous approaches were based on a SQL-like language able to deal with new operators [1]. Other approaches were based either on a logic programming language [2], on an object oriented paradigm [3] or on an algebraic approach [4]. The main drawback for these different approaches is the lack of user-friendliness. Furthermore, all these approaches, except the deductive approach, do not allow to express all kinds of geographical queries [5].

The first aim of Cigales is to increase the user-friendliness. To do so, we define a graphical language which is an easier and more natural way of manipulating geographical data. We use a simplified data model independent from the physical representation. A graphical representation is more complex but richer than a lexical one. Our second aim is to increase the expressing power allowing to manage a great number of geographical queries.

---

† CIGALES stands for Cartographical Interface Generating an Adapted Language for Extensible Systems.

†† This work has been partially financed by CGI / Fleximage

DEXA 90
29.31 Aout
Vienne Autriche

This paper presents in section 2, the user-interface of Cigales. Section 3 defines the semantic of this language. Section 4 presents the results of queries. Section 5 describes the architecture of the system and section 6 deals with the conclusion and further works.

2 - USER INTERFACE OF CIGALES

In this section, the definition of a graphical query and the alphanumerical data associated are presented.

2.1 Definition of a graphical query

Query construction is performed by a specialized editor shown in Figure 1. This editor provides query construction facilities (mainly using pop-up menus).

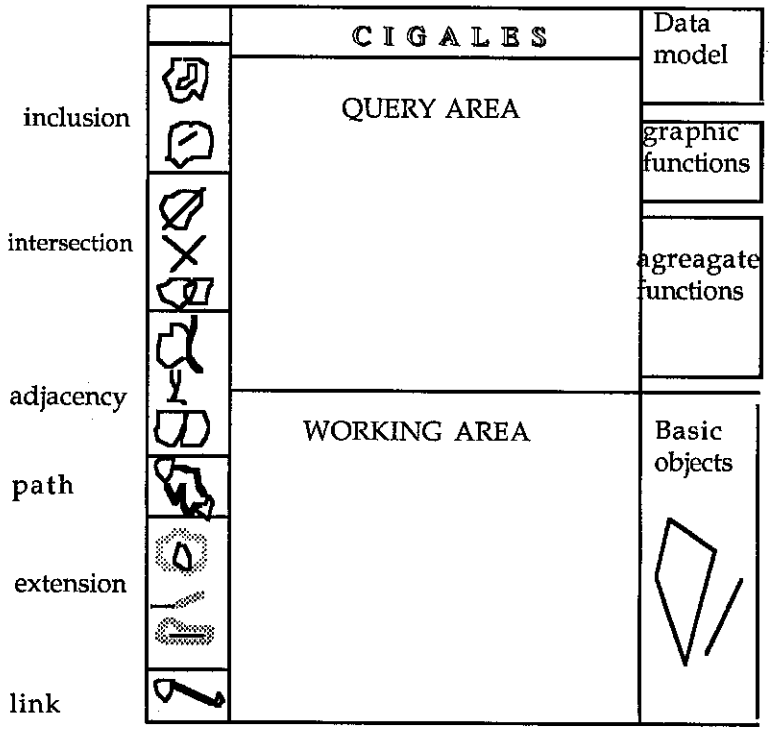


Figure 1

A set of icons representing all the available operators (inclusion, intersection, etc) is displayed. A query is a combination of different operators. We introduce the notion of current objects in order to realize these combinations. The user makes a choice of an object (either a previously-defined object or a basic object) then a choice of an operator (in case of unary operator such as extension - see section 3) or a choice of a second object and an operator to be applied to the selected objects. The rule for labelling the objects is defined in section 2.2. Two working spaces are defined : the query area and the working area. The first one contains a query at each step of its construction. Current objects will be selected in this area by the user. The second one is used to define new objects which can be combined, by application of an operator to the current objects. Aggregate functions (Min, Max, etc) can be applied to one or several objects or operators.

2.2 Definition of an alphanumerical query

An important issue of Cigales is to be independent of the physical storage of data. Therefore, alphanumerical data are modelized with an object-oriented methodology. We now only consider a classical object-oriented model relying on hierarchical concepts (simple hierarchy). A set of information is associated to each thematic object (attribute such as name and method such as length for the link operator).

An alphanumerical query is the definition of a label. A label may be defined on an object or on an operator. A label defined on an object is built from a set of information Ii of the data model for an object of type i. A label defined on an operator is built from a set of information Ij specified for this operator (or defined by the conceptor for dynamic operator such as extension).
A labelling function Li defined for each type Ti of object or operator allows to define an alphanumerical query :

Li : Ti -> D1 x ... x Dn

where Di is the domain of elements of Ii or Ij. A label can be seen as a tuple.
A label is defined as a list [s1, ... , sp] of selection criteria attached to an object or an operator. An attribute of the user-defined data model takes its value in a domain. A constant is an element of this domain. All selection criteria defined on elements are organized using a Regular Expression (R.E.). A R.E. is recursively defined as :

Manipulations of attributes :

Let ai be an attribute of the user-defined data model
Let c be a constant of the domain of ai
Let cp be an comparator (<, >, ≥, ≤, = , <>)
Let f be an aggregate function (Min, Max, ...)

Mono-valued attributes :

1) ai cp c is a R.E.

Multi-valued attributes :

2) Let c1, ... , cn be constants of the domain of ai then
   a) ai = c1, ... , cn (c1 and c2 ...) is a R.E.
   b) f(ai) cp value is a R.E.

Manipulation of expressions :

3) Let E1, E2 be R.E. then E1 ∨ E2 (E1 or E2) is a R.E.
4) Let E1, E2 be R.E. then E1 ∧ E2 (E1 and E2) is a R.E.
5) Let E1 be a R.E. then ¬ E1 (not E1) is a R.E.
6) Let E1 be a R.E. then $(E1)^+$ (repetition of E1) is a R.E.

The label of an operator is defined using a R.E. The semantic defined for the operators is a default multi-valued semantic (see section 3.2). Rules 1, 2, 3, 4 are defined as selection criteria in the relational model. For set-oriented operators the order of definition is not important for rule 2a. But this order is important for non-set-oriented operator. Rule 6 allows to define a connected or non-connected result for an operator. These rules can be applied either on objects or on operators.

Example : The label defined for the object "Town of Paris" is :

Let s1 = Type and s2 = Name

Application of rule 1 : Type = Town
Application of rule 1 : Name = Paris

Thus the label defined for the object which symbolizes Paris is [Type = Town, Name = Paris]

In this paper we limit the presentation of Cigales to this point. We shall not present an introduction of variables and the consequences. Informally a variable is a set of homogeneous values (selected data under constraints). A variable can be seen as a set of values an attribute can take within a query.

3 - SEMANTIC OF CIGALES

3.1 The functional langage

Many propositions concerning models for thematic maps can be found in the literature [4, 6, 7]. We will consider a subset of the operators defined in these approaches (only logical level operators). We do not consider operators such as map overlay because we are not interested here in physical data manipulation but in user defined query. The formalism is based on a functional language approach which provides composition of operators.

A basic object (O_i) is a graphical-typed object (G-type) : either a line (L) or an area (A). A point is considered as an area with a null surface. Each operator (op) is defined in a standard way :

op : O_i x O_j -> $\{O_k\}$ for binary operators
O_i -> $\{O_k\}$ for unary operators

O_i and O_j represent objects which are instanciated during query evaluation ; $\{O_k\}$, the query result, is a set of objects.
Two kinds of operators are defined from the user's point of view : Static operators rely on existing data while the dynamic operator (extension) is based on user-defined rules and existing data to compute a specific result (i.e., What is the extension of a fire initiated at 50 km of Toulon with a S-S-E force 6 wind ?). Two operators are available from the system point of view to be able to manage the notion of current object.

From the user's point of view, the operators are the inclusion, the intersection, the adjacency, the path, the link and the extension. They are defined by the following rules :

Inclusion : $C(O_1, O_2)$ = O_1 if O_1 C O2
= Ø otherwise

Integrity constraint : G-type (O_1) = A => G-type (O_2) = A

Intersection : $\cap(O_1, O_2)$ = {Ok} if there is a geometrical intersection between O1 and O2 (this intersection may or not be connected)
= Ø if O_1 and O_2 do not intersect

Adjacency : ¤ (O_1, O_2) = {Ok} if there is a common geometrical element for O_1 and O_2 (coordinates)
= Ø otherwise

Integrity constraint : G-type (Ok) = L

Path : -> (O_1, O_2) = {Ok}

Integrity constraint :
$\cap (O_1, O_2) = Ø$ and G-type (Oki) = L $\wedge \exists$ k1 / $O_1 \cap Ok1 \neq Ø \wedge \exists$ k2 / $O_2 \cap Ok2 \neq Ø$

A path is set of lines allowing to go from O_1 to O_2

Link : $\sqrt{}$ (O_1, O_2) = Ok

Integrity constraint : $O_1 \cap O_2 =$ Ø

G-type (Ok) = L where L is a (logical) line with the minimal distance between O_1 and O_2.

Extension : ->> (O_1) = {Ok}

Integrity constraint : G-type (O_1) = A => G-type (Ok) = A

From the system's point of view the operators are the union and the difference :

Union : $\cup (O_1, O_2)$ = {Ok} (geometrical union)

Difference : $\Delta (O_1, O_2)$ = {Ok} (geometrical difference)

Integrity constraint : G-type (O_1) = G-type (O_2) = A

The language is enhanced with a Composition operator to construct a query. Classical relational algebra operator (selection: σ), and aggregate functions (sum, min, max, avg, count) are available to define a label. The grammar which defines a query has been developped in [5].

3.2 Semantic of operators and queries

Each geometric operation is assumed by default to give back a set of objects. Yet, an unicity constraint can be explicitly specified by the user. Figure 2 is the graphical representation of the following query (query area window) : What are the roads crossing the TGV railway between Paris and Lyon ? No unicity constraint is defined on this query. So, (1) the roads may cross the TGV railway one or more times, and (2) several roads may also cross this railway. Figure 3 represents a possible result (result area window).

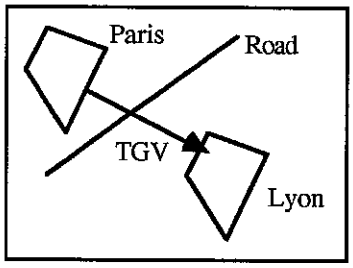


Figure 2

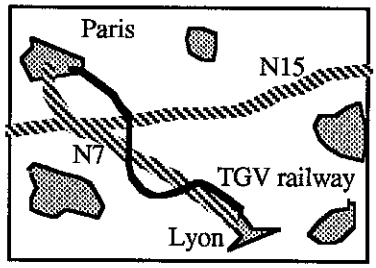


Figure 3

Two roads (N15 and N7) cross the TGV railway, and N7 does it several times.

Defining a semantic leads to set a default semantic. The default semantic will be the independence between two operators unless it is explicitly modified by the user. The default semantic is supposed to be known by the user, and the possibility to modify this semantic is offered. A graphical solution is adopted to indicate the modified semantic (use of different textures, etc).

For example the following graphic (Figure 4) represents an intersection between two areas A and B, and the intersection of the path between two areas C and D and the previously defined object A :

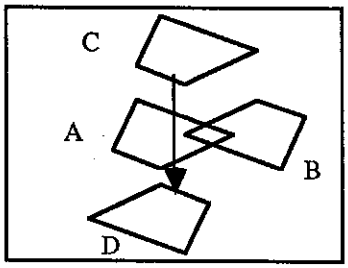


Figure 4

Two interpretations are possible :

1) intersection between the path from C to D and the object A without the intersection between A and B :

Composition ($\cap$ (A, B), $\cap$ (-> (C, D), Δ (A, B)))

2) intersection between the path from C to D and the object A (ignoring the object B) :

Composition ($\cap$ (A, B), $\cap$ (-> (C, D), A))

The default semantic is the second interpretation.

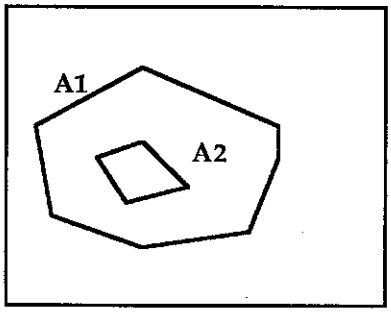
3.3 Example of queries

We now give some examples of queries translated into the functional language and their graphical representation :

Query 1 : What are the names of the towns bigger than 50 000 inhabitants of Var county ?

Composition (C (σ_{F1} (A1), σ_{F2} (A2)))

σ_{F1} : [Type = Town, Population > 50 000]
σ_{F2} : [Type = County, Name = Var]



object A1:

type = county
name = Var

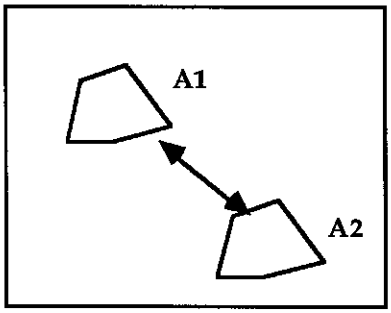
object A2:

type = town
pop> 50 000

Query 2 : What are the forests which distance from the town of Fougeres is less than 5 km ?

Composition (σ_{F3} ($\surd$ (σ_{F1} (A1), σ_{F2} (A2))))

σ_{F1} : [Type = Forest]
σ_{F2} : [Type = Town, Name = Fougères]
σ_{F3} : [Distance < 5 km]



object A1 :

type = forest

object A2 :

type = town
name = Fougères

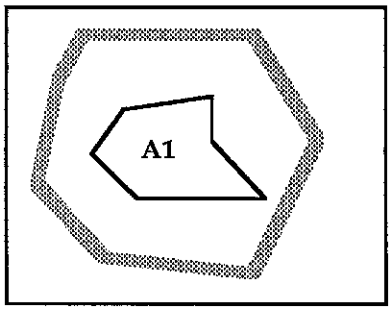
operator $\surd$:

distance < 5 km

Query 3 : What would be the radioactivity field if an incident happens in the St-Laurent-des-Eaux nuclear reactor ?

Composition (σ_{F2} ->> σ_{F1} (A1))

σ_{F1} : [Type = nuclear reactor,
Name = St-Laurent-des-Eaux]
σ_{F2} : [Intensity = 6 , Direction = South-west]



object A1 :
type= nuclear reactor
name= St-Laurent-des-Eaux

operator ->> :
parameters:
wind : intensity 6
direction : south-west

4 - RESULTS OF A QUERY

A query result is composed of a set of objects which represent an elementary result. An elementary result is an instanciation of each object and each operator of the functional expression. These results verify the properties defined by the user and may be multi-valued (i.e. the intersection between a road and a TGV railway may or not be unique).

Consider the following query : What are the roads croosing the TGV railway between Paris and Lyon ? The graphic query is represented figure 5 :

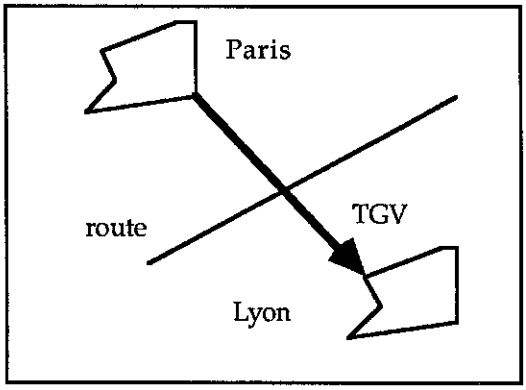


Figure 5

The result is composed of a set which contains two elements (figure 6 and 7) :

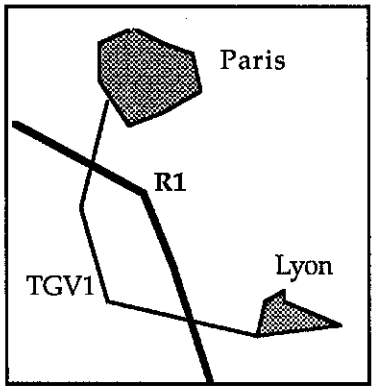


Figure 6

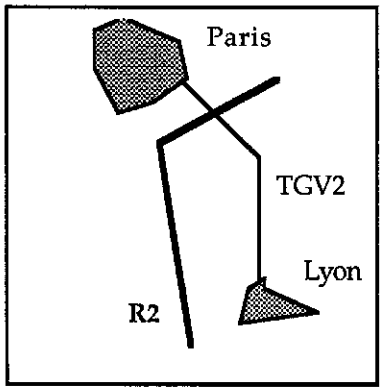


Figure 7

Several roads can cross a TGV railway, and a road may cross several times a TGV railway.

A result can be multi-valued. We explain this notion. Consider the following query represented figure 8 : What are the counties which intersect the forest of Laval ? Figure 9 and 10 give two results. The first one is a multi-valued result (composed by two areas), and the second one is mono-valued.

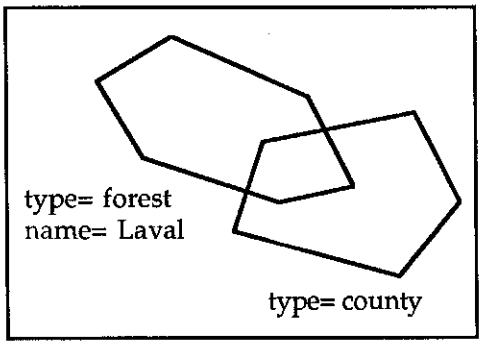


Figure 8

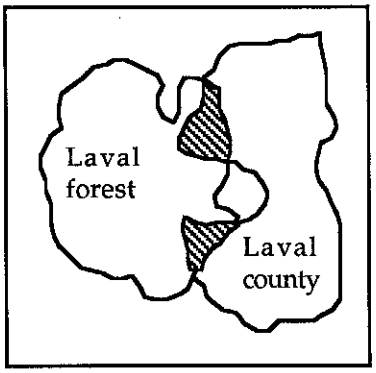


Figure 9

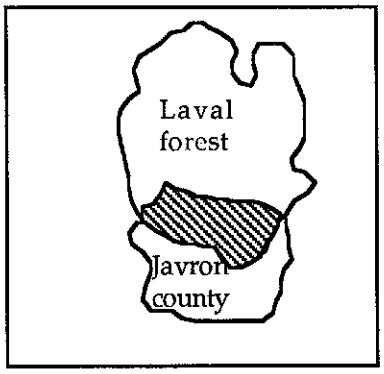


Figure 10

A query result is a set of lines and areas. This result is stored in a graphical expression.

Manipulations on a query result are made using : zoom, clipping and point location functions, but other useful functions such as : object modification, surface object measure, etc are defined.

Two possibilities are defined for the *zoom* function : (1) scale variation will increase the number of displayed information , and (2) zoom without scale variation will enlarge a particular area of the map, specified by the user (lens effect). In the second case, no acces to the database is required. The zoom function may be useful in the case of a research by *refinements*.

Clipping and point location functions allow the obtention of the characteristics of the selected objects. These functions need : (1) to take account of point location errors (by the definition of a margin of error for each object) and (2) a geometric index [8] to provide a reasonable response time.

To summarize, the user can define a Cigales query with the graphical language, specify a particular display (i.e. display in red all the main roads crossing the National Parc of Mercantour), visualize the result, zoom, etc.

5 - THE ARCHITECTURE

The introduction of new functionalities in a DBMS leads to different solutions of architecture : Extended DBMS, new DBMS or definition of a new layer on a top of a DBMS. In the first approach, new components (operators, domains, ...) are added to the system kernel [9], the second imply a complete modification of the DBMS [10] (this operation may be time consuming and costly). In the third approach, the DBMS is not modified : a layer is added at the user interface level to perform new functionalities ; the Data Manipulation Language and the external model of the DBMS are used as an entry point.

Our aim is to validate a new query language and not a complete system, so we adopt the third approach, the definition of a new layer, which is a more simple and realistic solution. In a first version, we are fully aware that our prototype will suffer from some weaknesses for the optimization (global graphical and alphanumerical optimization), but will offer a very user-friendly interface for geographical applications.
The main vocation of Cigales is to be independent of physical data management. Figure 5 represents Cigales architecture :

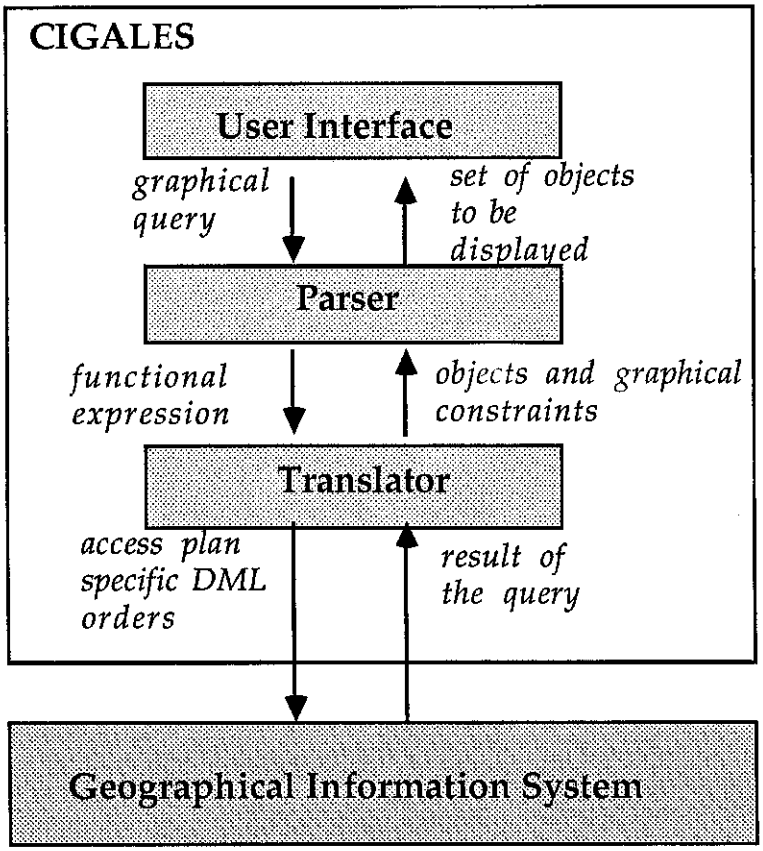


Figure 11

We now briefly describe the components involved in the architecture : the query is defined by the user with a graphical interface. The parser compiles this graphical query into a functional expression using Cigales operators ; logical optimization is performed on this expression to avoid redundancy and useless computation of expressions. The functional expression is then translated into

specific DML orders according to the physical storage of data : Extended Relational Data Base System [11], Object Oriented Database [10]. The query result is transmitted by the Physical Data Management System to the translator module, formatted into data structures associated to the logical query expression. Information about graphical display constraints defined by the user are also included in these structures. The result of the query is then transmitted to the user interface by the parser.

The main advantage of this architecture comes from the independence of the physical data management. The translator is the only module concerned with physical data storage.

A first prototype is currently under implementation on Sun workstation using an extended RDBMS [11] and a user-interface generator [12].

6 - CONCLUSION AND FURTHER WORKS

Geographical DMLs suffer from a lack of friendliness, and are too dependent of physical data representation. As relational DBMS introduced the concept of independence between physical and logical information, we need to introduce the concept of independence between logical (point, line and area) and physical (coordinates) geographical data. Abstract Data Type (ADT) is an attempt to provide this functionality. Furthermore, it seems necessary for semantic objects (town, road, etc) to be independent from their logical representation (point, line and area).

The approach we use for this new language is based on a graphical philosophy. The user defines a query with the help of a set of graphical primitives which modelize intersection, inclusion, etc. The graphical query is translated by a parser into an optimized functional expression using geometrical and classical alphanumerical operators. This functional expression is then translated into DML orders which are specific of a DBMS (extended DBMS, Object-oriented DBMS,...) in charge to manage physical data.

This architecture do not perform a global optimization of classical and geometrical operators in a first version. Our main contribution is to define and realize a graphical tool for a Geographical DBMS, independent of physical data storage.

The main advantages of Cigales are its ease-of-use and its expressive power. The ease-of-use is due to a natural approach to define a query by graphical specifications. The user defines the properties that must be verified by an object. This language is a new approach of geographical DML, and can be seen as a good complement to a language such as SQL for these applications. In further works, we will study the introduction of negation in our language and the various ways to optimize the query evaluation from the physical point of view.

REFERENCES

[1] A. Franck, MAPQUERY : Database Query Language for retrieval of geometric Data and their graphical representation, Computer Graphics, Vol 16, n° 3, July 1982

[2] E. Jungert, Inference rule in a Geographical Information System, IEEE Workshop on Language for Automation, New- Orleans, USA, November 1984

[3] JR Herring, Tigris : Topologically Integrated Geographical Information System, Proc. of the AUTOCARTO Conference, Baltimore, Maryland, USA, 29 March - 3 April 1987

[4] R.H. Güting, Gral : an Extensible Relational Database for Geometric Applications, Proc. of the VLDB Int. Conference, Amsterdam, The Netherlands, 22-26 August 1989

[5] M. Mainguenaud M.A. Portier, Cigales : a Graphical Query Language for Geographical Applications, To appear : 4th International Symposium on Spatial Data Handling, Zurich, Switzerland, 23-27 July 1990

[6] B. David, Le modèle Spatiarel, 4èmes Journées Bases de Données Avancées - INRIA, Bénodet, France, 17-20 May 1987

[7] M. Scholl A. Voisard, Thematic map modelling, Symposium on Large Spatial Data Bases, Santa-Barbara, USA, July 1989

[8] A. Guttman, R-Tree : A Dynamic Index Structure for Spatial Searching, Proc. of the ACM SIGMOD Conference, Boston, USA, 1-3 June 1984

[9] L. Rowe M. Stonebraker, The Design of Postgres, Proc. of the ACM SIGMOD Conference, Washington, USA, 26-30 May 1986

[10] F.Bancilhon, G.Barbedette, V.Benzaken, C.Delobel, S.Gamerman, C.Lecluse, P.Pfeffer, P.Richard, F.Velez, The Design and Implementation of O_2, an Object Oriented Database System, Proc. of the OODBS II Workshop, Bad- Munster, FRG, September 1988

[11] G.Gardarin, M.Jean-Noël, B.Kerherve, F.Pasquer, D.Pastre, E.Simon, P.Valduriez, L.Verlaine, Y.Viemont : Sabrina * : a Relational Database System developed in a research environment, TSI, Vol 6, n° 3, 1987

[12] Aida : User's Guide