CIGALES†: A GRAPHICAL QUERY LANGUAGE FOR GEOGRAPHICAL INFORMATION SYSTEMS

Michel MAINGUENAUD *
Marie-Aude PORTIER **

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

** Laboratoire MASI
Université Pierre et Marie CURIE
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.

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 [Fran82]. Other approaches were based either on a logic programming language [Jung84], on an object oriented paradigm [Herr87] or on an algebraic approach [Güti89]. 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.

The first aim of Cigales is to increase the user-friendliness. To do so, we define a graphical language which is easier and more natural to manipulate and 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.

This paper presents in section 2, the geographical queries and their expression. Section 3 presents the user-interface of Cigales. Section 4 defines the functional query sub-language.Section 5 presents the architecture of the system. Section 6 deals with the conclusion and further works.

---

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

4th Int. Symp on
Spatial Data Handling
22.27 juillet 90
Zurich . Suisse

2- GIS QUERIES AND THEIR FORMULATION :

In this section, a taxonomy of geographical queries and a presentation of available Data Manipulation Language (DML) are given.

2-1. Geographical queries

GIS queries can be divided in 1) alphanumerical 2) map-oriented 3) deductive selection criteria or 4) printing-oriented queries.

Alphanumerical typed queries

These queries can be resolved using a classical Data Base Management System (DBMS) for alphanumeric data, or using an Extended Relational Data Base System (ERDBMS) for complex objects. These queries do not need any geometrical operator.
Examples :
1) What are the towns of Aveyron county ?, 2) What is the population of Angers ?, 3) What are the names of the towns bigger than 50 000 inhabitants of Var county ?

Map oriented queries

These queries can be classified in topological or geometrical classes. For topological queries, links between different geographical objects (point, line and area) appear, without any notion of coordinates. Spatial relationships between objects are the most common queries. They need specialized operators such as inclusion, intersection, etc
Examples :
4) What are the forests near the town of Fougeres ?, 5) Is forest of Fontainebleau bordered by GR7 path ?, 6) What are the roads crossing the TGV railway between Paris and Lyon ?

Geometrical queries manipulate the same geographical objects. For these queries the central notion is the coordinates. They allow to express distance, surface computation queries, etc.
Example : 7) What is the name of the nuclear reactor less than 10 km far from Amboise ?

Deductive selection criteria queries

Two types of queries can also be distinguished in this case : transitive closure queries and natural deductive queries. A graph traversal operator [RHDM86] is required for the first class.
Examples : 8) Give all possible paths from Paris to Nice using a highway, 9) What is the shortest path from Dijon to Poitiers ?, 10) What waterways may I take to sail from Chateau-Gontier to Le Mans with a 1.20 meter draught ship?

Natural deductive oriented queries require complex and specific rule programs to be developed.
Examples : 11) What is the extension of a fire initiated at 50 km of Toulon with a S-S-E force 6 wind ?, 12) What would be the radioactivity field if an incident happens in the Saint-Laurent-des-Eaux nuclear reactor (taking account of parameters such as weather forecasting etc) ?

Printing-oriented queries

Results of a query may be pointed out by graphical display. Specified parameters in the query, or default associated values of the geographical objects can drive the graphical display. These default values have to be seen as the legend of a map.
Examples : 13) Shadow the towns of Corse such as born rate is superior to dead rate, 14) Display in red all the main roads crossing the National Parc of Mercantour

2-2. Data Manipulation Languages

In order to express all the queries defined in the previous section, four types of Data Manipulation Language (DML) are described : Extended-SQL approach, rule language approach, Object Oriented approach and algebraic approach.

Extended-SQL approach

These languages are based on an already available DML of a DBMS. Manipulation of geometrical information is performed using specialized operators. The SQL language [FRS88], [SMcDO87] is not the only way to define a query. A query can also be expressed using a QBE (Query By Example) philosophy [CJ88] or a SQL-like language [Fran82] (Show instead of Select etc).
As an example, PSQL [FRS88] or Spatiarel [Davi87] operators are based on the three standard geometrical information : point, line and area. Nevertheless, with PSQL approach, physical data representation is independent of the Data Model by the definition of geometrical elements as Abstract Data Types (ADT). An ADT is an encapsulation of data structures and manipulation operators to hide all the implementation details from the user. Data manipulation is made through the use of a set of defined primitives. (A basic example is network management allowing accessibility-oriented operators).
Example : What is the name of the nuclear reactor less than 10 km far from Amboise (query 7) ?

Such a PSQL query is:

Select Name, Localisation **From** Nuclear-reactor **On** France-Map
Where localisation **Within** circle ((**Select** X, Y **From** Town **On** France-Map
Where name = "Amboise"), 10)

The main advantage is the fact that the philosophy of existing DBMS is respected. The implementation seems to be simple and realistic. The introduction of the ADT concept allows modifications of data representation without any effect on the relations defined among these objects.
The main drawback concerns data access. Indeed, no global optimization of data access is performed for a query. The DBMS realizes optimizations of classical relational operators, but does not take account of specialized operators (multi-dimensional indexes are not managed by classical DBMS). In addition, data manipulation languages suffer from a lack of independence between an object and its logical representation : point, line and area (though an improvement appears by introduction of ADT allowing objects not to be anymore dependent of their physical representation). An SQL-like language is not a very user-friendly interface.

Rule language approach

These languages are based on knowledge representation systems possibly completed with a spatial component. As an example, IPL [Jung84] proposes a mechanism of icons, windows and ports (ADT modelizing a logic input/output unit). Information are stored as facts and are post-fixed by their type. A query is defined as a rule program.
Example : What is the name of the nuclear reactor less than 10 km far from Amboise (query 7) ?

Extensional Data Base:
Position (St-Laurent.nuclear-station, 10.c, 2.c)
Position (Amboise.town, 5.c, 5.c)

Intentional Data Base:
Distance (x.type, y.type, dist) :-
Position (x.type, x1.c, y1.c)
Position (y.type, x2.c, y2.c)
Compute (x1.c, y1.c, x2.c, y2.c, dist)
(Predicate compute deals with the coordinates)

Query : **? Distance** (X.nuclear-station, Amboise.town, dist) ∧ **Less-than** (dist, 10)

The main contribution of such an approach is the augmentation of the expressive power of the language [AU79] by the introduction of the recursion concept.
The opposite viewpoint is that deductive mechanisms suffer from optimization problems (recursion). Indeed, it does not seem so simple and user-friendly to express a query in a language based on a logical formalism.

Object-Oriented approach

In the object-oriented approach, information are highly structured by the introduction of classes and inheritance concepts [GR83] [Herr87]. Data encapsulation and overloading provide some manipulation facilities and independence between physical and logical data representations (as ADTs do).
Example : What are the towns of Aveyron county ? (query 1)

The data model defined in [MO86] is based on entity :

```
Type Land_division is entity
        Name (Land_division)  ->   String
        Area (Land_division)  ->   Polygon
```

and sub-type :

```
Type Area_life is Land_division
```

A query is expressed using system operators (such as OB_INCLUDE : inclusion of objects) :

```
OB_Include (Select (Area_life), Select (Land_division, Name = "Aveyron"))
```

Another way to express a query is to use a navigational language (as DML of a Network Data Base System)

```
? Area_life.Name > Land_division.Name = "Aveyron"
```

Complex objets manipulation [AB87], nest/unnest operators have already been studied and their expressive power evaluated. This approach provides a good data organization, but DMLs suffer from a lack of friendliness (like the Network Data Bases Manipulation Language), and are unadapted to muti-media information management. Manipulations rely on a logical formalism [AG87] uneasy to understand for a naive user. Furthermore, SQL approaches for complex objects "inherit" from the lack of conviviality of SQL.

Algebraïc approach

Arc-Info [More89] Geographic Information System is based on this approach. The user specifies the operations he wants to realize with a macro language AML (Arc Macro Language). Another example is the extensible system Gral [Güti89] with an algebraic model : Geo Relational Algebra. The query language also modelizes an execution language. The operators can be parametered (parameters are in curly-brackets [])
Example : What are the names of the towns bigger than 50 000 inhabitants of Var county ? (query 3)

```
cities select [population > 50 000] inside county select [county.name = "Var"]
project [city.name]
```

The main advantage of this approach is to be well-adapted to optimization problems. In Gral system, the algebra represents a query language but also an execution language. A query can be

easily translated into an optimized access plan [BG89]. A drawback is that expressing a query with an algebra is not an easy task. This kind of language suffers from an obvious lack of conviviality.

In conclusion, the main drawback for these different approaches is the lack of conviviality for the user. Another problem is that all the approaches previously described, except the deductive approach, do not allow to express all the four kinds of queries previously defined. The deductive approach provides this possibility, but with rather bad performances.

The aims of this new language are first to increase the conviviality for the user. To do so, we define a graphical language which is easier and more natural to manipulate and propose a simplified data model to hide how data are physically stored. Our second aim is to increase the expressing power allowing to express a great number of geographical queries (see, for more details, the section in which the expressive power of Cigales is evaluated).

3- USER INTERFACE OF CIGALES

The user interface of Cigales is divided in two parts : 1) geographical data manipulation, 2) display of the result on a map.

3-1 Geographical data manipulation :

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

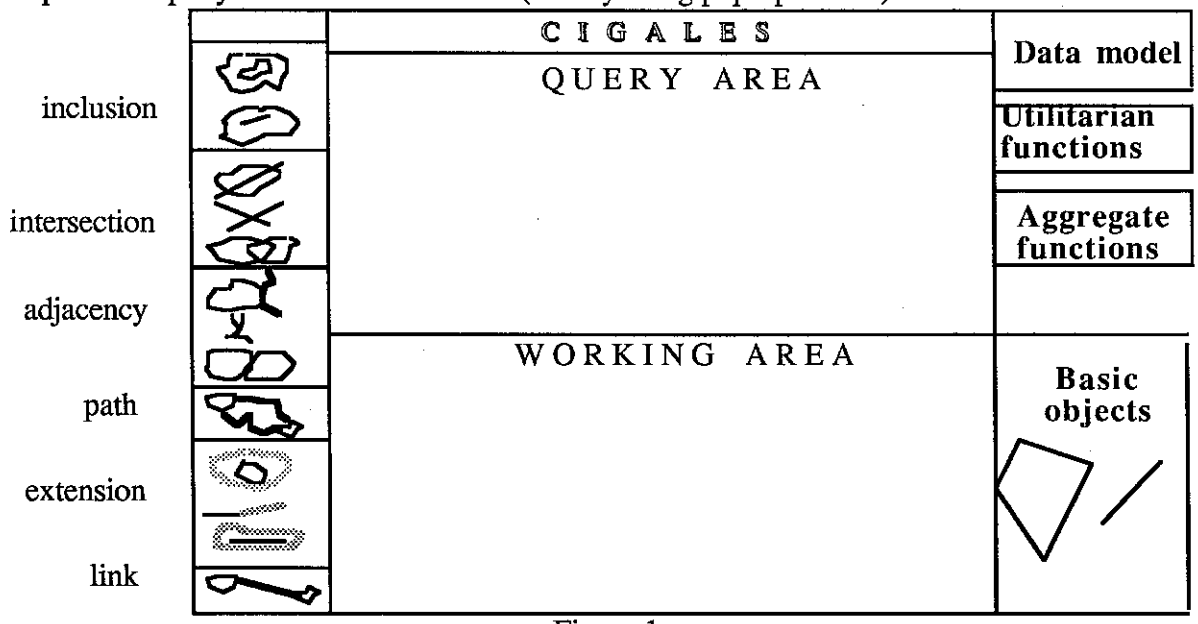

Figure 1

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 make a choice of an object (already existing or a basic object) then a choice of an operator (in case of unary operator such as extension - see section 4) or a choice of a second object and an operator to be applied to the selected objects. The rule for labelling the objects is not defined in this paper. 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.

3.2 Display of the results :

Geographical objects (roads, towns, etc) can be represented in a result area window in various ways according to different views or different scales (these objects are represented by symbols in the legend of a map). 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). 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 [Gutt84] 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.

4- THE FUNCTIONAL LANGUAGE :

Many propositions concerning models for thematic maps can be found in the literature [Davi87, Güti89, SV89]. 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. Section 3.1 presents the basic notions, section 3.2 presents the semantic of this language.

4.1 Basic notions :

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 O_2
		=	Ø	otherwise

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

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

Adjacency : $\square(O_1, O_2) = \{O_k\}$ if there is a common geometrical element for O_1 and O_2 (coordinates)
$= \emptyset$ otherwise

Integrity constraint : G-type $(O_k) = L$

Path : $\rightarrow(O_1, O_2) = \{O_k\}$

Integrity constraint : $\cap(O_1, O_2) = \emptyset$ and G-type $(O_{ki}) = L \wedge \exists k_1 / O_1 \cap O_{k1} \neq \emptyset \wedge \exists k_2 / O_2 \cap O_{k2} \neq \emptyset$

Link : $\surd(O_1, O_2) = O_k$

Integrity constraint : $O_1 \cap O_2 = \emptyset$
G-type $(O_k) = L$ where L is a (logical)line with the minimal distance between O_1 and O_2.

Extension : $\rightarrow\rightarrow(O_1) = \{O_k\}$

Integrity constraint : G-type $(O_1) = A \Rightarrow$ G-type $(O_k) = A$

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

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

Difference : $\Delta(O_1, O_2) = \{O_k\}$ (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), and aggregate functions (sum, min, max, avg, count) are available to define a label (selection criteria over an object or an operator). Labels are not formally defined in this paper.

The following grammar defines a query :

Query	::=		Composition (functional_expression	elaborated_query)		
elaborated_query	::=		,	functional_expression	elaborated_query	
		/	$\emptyset$			
functional_expression	::=		label	operator (argument_list)		
operator	::=		$\rightarrow / \rightarrow\rightarrow / \cap / \cup / \Delta / \surd / \square / C$			
argument_list	::=		argument		argument_list	
		/	elaborated_query		argument_list	
		/	$\emptyset$			
argument	::=		label	object	object_list	
object_list	::=		,	label	object	object_list
		/	$\emptyset$			
object	::=		L			
		/	A			

This grammar is regular (a query is equivalent to an elaborated_query). This characteristic allows recursive definition of a query using a previously defined query (the new one has to be seen as a refined query). Therefore, we can easily express the intersection between a new object and a subset of objects defined in a previous query.

4.2 Semantic of the language :

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).

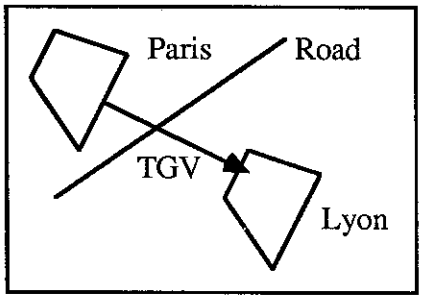

Figure 2

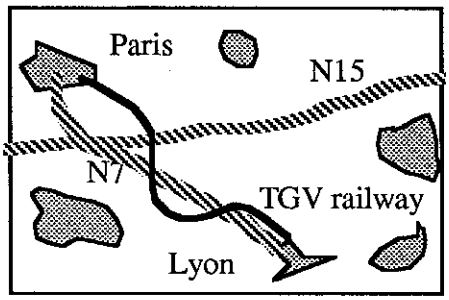

Figure 3

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 :

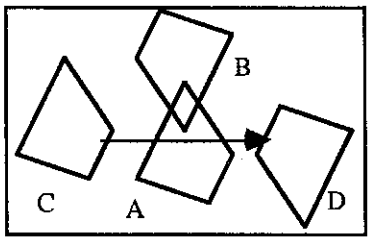

Figure 4

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.

4.3 Example of queries

We now give some examples of queries translated into the functional language :

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]

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

Composition (σ_{F3} (-> (σ_{F1} (A1), σ_{F2} (A2))))

σ_{F1} : [Type = Forest]

σ_{F2} : [Type = Town, Name = Fougères]

σ_{F3} : [Distance < 5 km]

Query 3 : What are the roads crossing the TGV railway between Paris and Lyon ?

Composition ($\cap$ (σ_{F3} -> (σ_{F1} (A1), σ_{F2} (A2)), σ_{F4} (L1)))

σ_{F1} : [Type = Town, Name = Paris]

σ_{F2} : [Type = Town, Name = Lyon]

σ_{F3} : [Type = Railway, Name = TGV]

σ_{F4} : [Type = route]

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 [RS86], the second imply a complete modification of the DBMS [B*88] (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 at the optimization and query evaluation level, 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 :

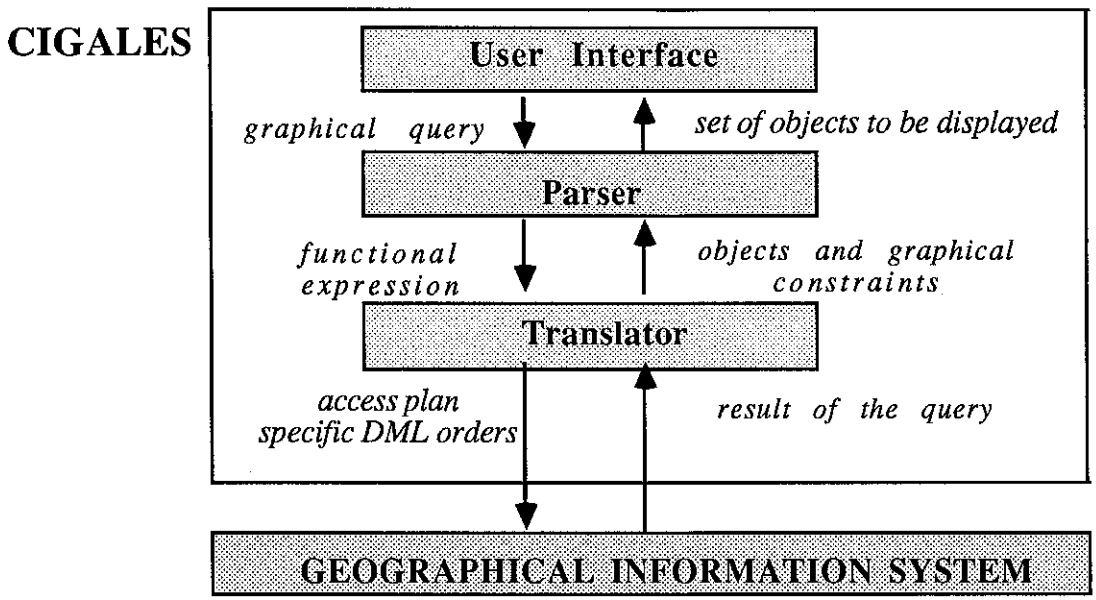

Figure 5

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 [G*87], Object Oriented Database [B*88], or GIS [BDQV90]. 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.

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. A first prototype is currently under implementation using Sun workstation and an extended RDBMS [G*87].

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.

REFERENCES :

AB87 S. Abiteboul, C. Beeri, On the power of languages for the manipulation of complex object, Int. Workshop on Theory and Application of Nested Relations and Complex Objects, Darmstadt, FRG, 1987

AU79 A. Aho, JD Ullman, Universality of data retrieval language, Proceedings of the ACM P.O.P.L. Conference, San Antonio, USA, 29-31 January 1979

B*88 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

BG89 L. Becker, R.H. Güting, Rule-based Optimization and Query Processing in an Extensible Geometric Database System, Research report 312, 1989, Dortmund University, FRG

CJ88 AF Cardenas, T. Joseph, Picquery : A high level query language for Pictorial Data Base Management, IEEE T.O.S.E., Vol 14, n°5, May 1988

BDQV90 K. Bennis, B. David, I. Quilio, Y. Viemont, GéoTropics : Database Support Alternatives for Cartographic Applications, To appear : 4th International Symposium on Spatial Data Handling, Zurich, Switzerland, 23-27 July 1990

Davi87 B. David, Le modèle Spatiarel, 4èmes Journées Bases de Données Avancées - INRIA, Bénodet, France, 17-20 May 1987

Fran82 A. Franck, MAPQUERY : Database Query Language for retrieval of geometric Data and their graphical representation, Computer Graphics, Vol 16, n° 3, July 1982

FRS88 C. Faloutsos, N. Roussopoulos, T. Sellis, An efficient Pictorial Data Base System for PSQL, IEEE T.O.S.E., Vol 14, n°5, May 1988

G*87 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

GR83 A. Goldberg, D. Robson, SMALLTALK80 : The Language and its Implementation, Addison-Wesley, 1983

Güti89 R.H. Güting, Gral : an Extensible Relational Database for Geometric Applications, Proc of the VLDB Int. Conference , Amsterdam, The Netherlands, 22-26 August 1989

Gutt84 A. Guttman, R-Tree : A Dynamic Index Structure for Spatial Searching, Proc of the ACM SIGMOD Conference, Boston, USA, 1-3 June 1984

Herr87 JR Herring, Tigris : Topologically Integrated Geographical Information System, Proc of the AUTOCARTO Conference, Baltimore, Maryland, USA, 29 March - 3 April 1987

Jung84 E. Jungert, Inference rule in a Geographical Information System, IEEE Workshop on Language for Automation, New-Orleans, USA, November 1984

MO86 F. Manola JA Orenstein, Toward a general spatial data model for an object oriented data base management system, Proc. of the 12th VLDB Conference, Kyoto, Japan, August 1986

More89 S. Morehouse, The Architecture of Arc/Info, Proc. of the Autocarto9, Baltimore, USA, April 2-7, 1989

RHDM86 A. Rosenthal, S. Heiler, U.Dayal, F. Manola, Traversal Recursion : A Practical Approach to Supporting Recursive Applications, Proc. of the ACM SIGMOD Conference, Washington, USA, May 1986

RS86 L. Rowe M. Stonebraker, The Design of Postgres, Proc of the ACM SIGMOD Conference, Washington, USA, 26-30 May 1986

SMcDO87 R. Sack-Davis, K.J. McDonell, B.C. Ooi, GEOQL : A Query Langage for Geographic Information Systems, Australian and New Zealand Association for the Advancement of Science Congress, Townsville, Australia, August 1987

SV89 M. Scholl A. Voisard, Thematic map modelling, Symposium on Large Spatial Data Bases, Santa-Barbara, USA, July 1989