WHAT IS HAPPENING AFTER THE DEFINITION OF AN END-USER QUERY?

MAINGUENAUD Michel

FRANCE TELECOM - Institut National des Télécommunications
9 rue Charles Fourrier
F91011 Evry - France
Tel: + 33 1 60 76 47 14 / fax : + 33 1 60 76 47 80
email : maing@int-evry.fr

ABSTRACT :

This paper presents the experiment we perform in the context of the CIGALES[1] project. Cigales is a graphical query language for Geographical Information Systems. We detail here the lower layers of such a prototype. To be independent-application, a graphical query language must offer an internal language to model a user's query with a high level of abstraction. To be independent-Data Base Management System, this internal language must manipulate a logical representation of a query and must translate this representation into the specific DBMS orders.
We detail in this paper the link between the graphical query language Cigales and the several architectures we used as a DBMS (file system, relational DBMS with external operators, extended relational DBMS, and object oriented DBMS).

I. Introduction:

Various domains such as hypertext, geographical databases require multi-media data. The new applications involve a huge amount of data. This characteristic implies the use of a Data Base Management System (DBMS). The classical data manipulation languages dedicated to the management of alphanumerical data (i.e., SQL [23,26]) have a very poor expressive power and they are not adapted to manage spatial data.
The definition of a user-friendly query language and the choice of a data base architecture represent two key problems in a Geographical Information System design. The independence between the query language and the data base architecture is very important to insure the portability of the applications and the software engineering. Three steps are necessary to guaranty this independence: (1) the system must provide an independent-application internal language to model the user's query, (2) the system must provide a

[1] CIGALES stands for Cartographical Interface Generating an Adapted Language for Extensible Systems

[illegible]

translation mechanism from the internal language primitive operations into the specific DBMS orders and (3) the system must provide the management of the DBMS orders when the DBMS is not powerful enough to offer an internal language primitive operation as an order (i.e. the management of a sequence of DBMS orders).
This paper describes the experiment we perform in the context of the CIGALES project. Cigales [18] is a graphical query language for a Geographical Information System. This end-user language is not based on a textual query language but the semantics of a query is expressed by a drawing. The internal representation of a graphical query is totally independent from the representation of the DBMS logical data model. To validate this approach we made an implementation following four philosophies of DBMS: (1) based on a file system, (2) based on a relational DBMS with external geometrical operators, (3) based on an extended relational DBMS and (4) based on an object oriented DBMS.
Our goal was not to provide a complete architecture of a GIS but rather to evaluate the ability of the translation mechanism to handle several storage systems. Furthermore, the geometrical algorithms we used were very simple since the basic assumption of the CIGALES project is: "The DBMS offers Abstract Data Type (ADT) like operators"[24]. This realistic assumption frees from the geometrical coding and specific geometrical algorithms. Our main interest is the definition of a query and its evaluation with an ADT philosophy.

Part II presents the internal language used to model the graphical query, part III presents the management of the internal language primitive operations, part IV presents the translation mechanism into the four possible database architectures we tested, part V presents the conclusions of the experiment.

II. THE INTERNAL LANGUAGE:

The development of the graphical (or visual) user interface implies the definition of an internal language to model a user's query. We do not detail here the various ways of defining a query [i.e., 4,6,8,11,14,15,20,21,22,25]. As an example we take the Cigales formalism (We do not detail here the graphical language [18] and its semantics [5]). A Cigales query is a drawing that expresses the semantics of a query. The figure 1 represents the query: (Q1) "I would like to know the National roads crossing the TGV railway joining Paris to Lyon."

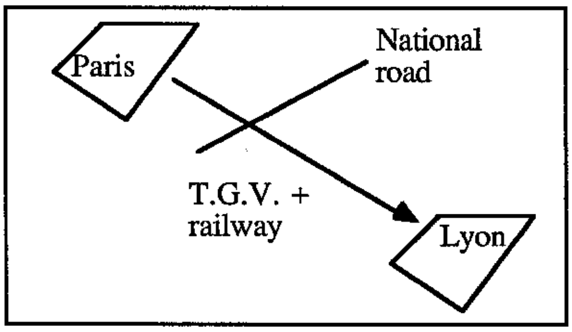

Figure 1

To model such a query, a formal expression is defined. This internal language modelling a query is based on a functional paradigm. This paradigm is well adapted to define the combination of several high level operators. For the time, SQL-like languages seem to be the level of interface between the DBMS and the applications. In fact, an end-user real size query corresponds to a set of DBMS orders. Several SQL statements are necessary to manage a high level operator [16]. The first step of a query resolution mechanism is to model the graphical query with a high level of abstraction. The operators are defined with an abstract data type methodology. At this step the query resolution process is not concerned with the logical data model constraints (i.e., the third normal form in the relational model [26]), and the physical storage optimization.

The grammar defining a query is detailed in [18]. As an example, let $\cap$ be the geometrical intersection operator and -> the path operator, the graphical query defined in figure 1 is intuitively modelled by:

Combination ($\cap$ (-> (Paris, Lyon), Road))

Several SQL statements are required to evaluate an end-user query. The following example illustrates this need:

(Q2) "What are the paths from the town Paris to the town Nice adjacent to a lake and crossing a town in a non-forest part?"

The graphical query is represented figure 2.

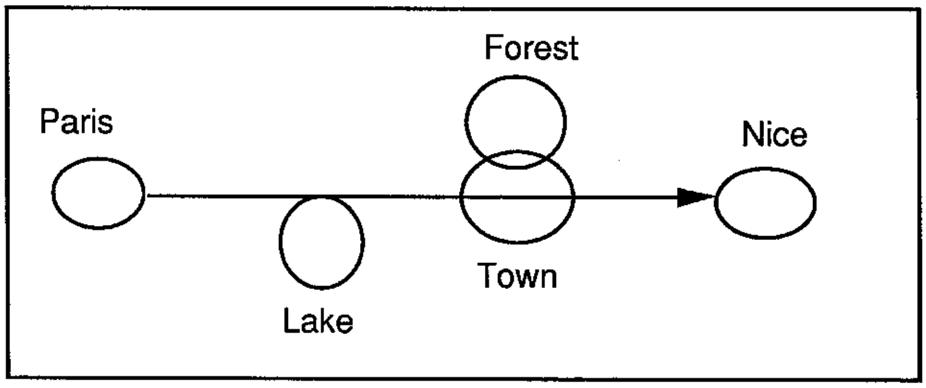

Figure 2

Let Delta (Δ) be the geometrical difference operator, Adjacence (◊) be the adjacency operator. The modelling of such a query is:

```
Combination (    ∩ ( -> (Paris, Nice), Δ (Town, Forest)),
                 ◊ ( -> (Paris, Nice), Lake ) )
```

One can remark, for example, an interaction between the Intersection and the Delta operator. An Extended SQL statement must allow the composition of operators. Let us imagine a database defined with a relational-based formalism by three relations Town, Lake and Forest. Such an order may be:

```
Select    Intersection (  Path (Paris, Nice),
                          Delta (T.geom, F.geom) ),
          Adjacence (     Path (Paris, Nice),
                          L.geom)
From      Town T, Forest F, Lake L
```

An alternative, the one we chose, is to transform such a query into several basic Extended SQL statements and to manage the partial results of the evaluation.

III. THE PRIMITIVE OPERATIONS:

Traditionally, geographical queries are divided into two classes: the thematic-oriented queries and the network-oriented queries. The basic operators we defined are represented in figure 3.

The thematic-oriented operator [18]:

the geometrical intersection (∩), the euclidian distance (√),
the localization (¿), the inclusion (C), the adjacency (¤), the difference (Δ)

The network-oriented operator [13]:

the direct link (ΠD), the end (-><), the beginning (>->), the join (-><-), the path (ΠT),
the intersection of paths (Πis_e), the inclusion of paths (Πic_e),
the inclusion of stop place(s) in a path (Πic_ne),
the intersection of stop places (Πis_n), the inclusion of stop places (Πic_n)

Figure 3

One can notice that these operators are high level operators. They do not take into account the logical data model of geometrical information such as point, line or area. Furthermore, we do not define the specifications of the

operators concerning whether a geometrical configuration respects the semantics of the operator [7]. The only condition is provided by the following assumption: "*A binary (resp. unary) operator has two objects in input and delivers an object as output*". A more complex query is obtained by a combination of these operators (i.e., the expression of the query represented figure 2).
The thematic-oriented operators require the application of a geometrical algorithm. The network-oriented operator, ΠT, requires a transitive closure algorithm; the others correspond to a set of SQL statements.
The management of a pure functional expression as a query leads to very bad performances due to the redundancy of partial sub-expressions. A logical optimization is performed by the query resolution engine to avoid the re-computation. An Update Propagation Mechanism insures the correctness of the result. The query resolution engine was written in Le_lisp language.

IV. THE TRANSLATION MECHANISM:

Once the logical optimization is performed, the execution plan is ready to be executed. A high level operator may correspond to a single data manipulation order of the DBMS (i.e., an intersection with an extended relational DBMS), but it may also correspond to a list of data manipulation statements (i.e., the intersection of paths). The translation mechanism is in charge of defining the sequence of DBMS understandable orders. We try four architectures as a data manager: (1) a file system-based manager, (2) a relational DBMS with external geometrical operators, (3) an extended relational DBMS and (4) an object oriented DBMS.
We are not interested in the response time of a query. Therefore the physical optimizations (depending on spatial indexes, the size of the data, etc.) are not relevant for this study. The spatial indexes and the physical storage would be very important in the case of the definition of a performance benchmark. This study is defined on the expressive power of the data manager query language.

IV.1 The file-system-based architecture:

This implementation is based on an algebraïc interface with a research prototype (written in C) developed at the Laboratoire de Robotique de Paris [12]. This prototype does not offer all the operators defined in figure 3 (intersection and adjacency). Nevertheless, the philosophy may be analysed.
The main drawback is that a file system-based architecture implies to rewrite the basic functions offered by a classical DBMS (i.e., the crash recovery, the concurrency control, the relational operations).
An advantage is the open-architecture since new operators can easily be added, without troubles of dynamic loader, shared memory, ...

IV.2 The relational-based architecture with external operators:

The implementation is based on a research prototype developed at INRIA [9]. This DBMS is an extended relational DBMS but we used it as a classical relational DBMS (i.e., we did not define predicates and operators to be introduced in the heart of the DBMS).
The thematic-oriented operators (intersection, adjacency, euclidian distance and inclusion written in C) imply to extract two "maps" and perform the operator with a Cartesian product of the "tuples" of each map. A predicate (Q3) such as: "road is adjacent to lake and (lake.usage = 'recreational' or road.name = 'Princess Highway') " - defined in [19] - obliges to rewrite part of the selection operator in the thematic oriented operators or to evaluate the correct sub-set of the Cartesian product in the data base which is very time and place consuming since the result is written in a file to be acceded by the external operator.
In the case of the network-oriented operator, the evaluation of a transitive closure operator [1] is a problem since three alternatives are possible: (1) the sub-network is extracted from the database and is managed as a classical file, (2) a DBMS call is performed for each evaluation of a successor of a node or (3) a DBMS call is performed for each node which has not been already visited. We retained the third solution.
This architecture is not very well adapted to a multi-users configuration since the DBMS locks a complete relation (even the tuples that are not relevant at all while evaluating a query - see the previous example). A complete optimization cannot be performed since alphanumerical operators (i.e., selection) and geometrical operators (i.e., intersection) are separated.

IV.3 The extended-relational-based architecture:

The implementation is based on a research prototype developed at INRIA [9]. The operators are written in Le_lisp language and they are integrated in the DBMS. A SQL query may use operators and predicates in the Select and Where statements.
The main drawback is that the operators do not allow a definition of recursive queries (i.e., to evaluate a path). In the case of the recursive queries, several works have been performed on the subject in the data base context [1, 2]. Unfortunately the evaluation of a query in the definition of a new operator is not permitted in the prototype we used. Therefore, the transitive closure process is the same as for the other implementations (and of course it has the same drawbacks).
This philosophy is very well adapted to the manipulation of abstract data types for geometrical or alphanumerical data and to the manipulations in the context of multi-users applications (since only the relevant information is locked by the DBMS while evaluating a query). The extension to the coupling

between the alphanumerical optimizer and the spatial optimizer should give some very good performances (but an efficient coupling is rather difficult).

IV.4 The object oriented architecture:

The implementation was based on research prototype developed by the GIP Altaïr [3]. SQL is a norm since 1986 but the object oriented query languages are not normalized at all. To be free from a particular object oriented system, the option we retained was to use the SQL-like interface to dialogue with the DBMS.
We only considered the network-oriented operators since only a transitive closure process was necessary to complement the SQL-like language. Thematic-oriented operators would require to code the geometrical algorithms with a specific programming language. Several methods have been introduced to improve the SQL-like query language since the update, insert and delete statements were not available. Furthermore the coupling between the environment of this prototype and a foreign processus is rather difficult.
This DBMS does not provide the possibility to define dynamically a class. This limitation is very important since the basic assumption is: "*An operator has two objects in input and delivers an object as output* ". A complete pre-compilation of the various possibilities following the "objects" of the database and the operators is not realistic at all. Therefore, very strong restrictions appear while managing the alphanumerical data associated to the result of an operator. Unfortunately this feature is very important while manipulating a complex query since new objects are generated by the application of an operator.
Nevertheless, the object oriented paradigm is very well adapted to model the alphanumerical data and the network-oriented data [10, 17].

V. CONCLUSION:

Many applications need to manage multi-media data. In particular, geographical applications require to model spatial data and alphanumerical data. Since the number of applications is considerable, it is of prime importance to offer an independent-application and independent-DBMS language to model an end-user query. The independence between the language and the applications allows using the same technology for several applications requiring the same class of queries (the semantics of the alphanumerical attributes is different but the required expressive power of the operators is the same). The independence between the user interface and the DBMS allows changing the technology of the data manager without changing the user interface. The intermediate level between the application and the DBMS has to be configurable to take into account the possibilities of the DBMS (i.e., the

management of the transactions, the level of the primitive operators, the management of the partial results).
In this paper we present the experiment we perform in the context of the CIGALES project, the definition of a geographical query language. We try to compare four possible architectures to manage the geographical data: (1) the file system-based architecture, (2) the relational DBMS with external geometrical operators-based architecture, (3) the extended relational DBMS-based architecture and (4) the object oriented DBMS-based architecture without changing the user interface. The user interface is the same, the internal language primitive operators are the same, the management of the query resolution engine is the same. The only change is the translation mechanisms since the DBMS interface are different. The principle follows the philosophy of the Pascal programming language compiler with the generation of a P-code (here the internal language).
Our goal was not to define a commercial product but to evaluate the feasibility of defining a graphical query language that may be free from the DBMS managing the data and from the application. The future formal work consists in increasing the expressive power of the user interface query language (i.e., the introduction of the negation that is difficult with a graphical philosophy). The future implementation work is based on our new configuration: a SUN-4 with two DBMS (Postgres and O2) and C as a programming language (Yacc for the analysis of the internal language and Motif for the user interface).

Acknowledgement:
The author wants to thank R. Jeansoulin for his help while coupling Cigales with Goodies, S. Chabonnier for the development of geometrical operators in the context of the relational DBMS with external operators and J. Torterat for the development of geometrical operator in the context of the extended relational DBMS.

REFERENCES:

[1] Aho A., Ullman JD : Universality of Data Retrieval Languages, Proc. of the ACM POPL Conf., San-Antonio, USA, Jan. 79
[2] Bancilhon F., Ramakrishnan R. : An amateur's Introduction to Recursive Query Processing Strategies, Proc. of the SIGMOD Conf., Washington, USA, May 28-30 1986
[3] Bancilhon and all, The Desing and Implementation of O2, an Object Oriented Database System, Proc. 2nd Int. Workshop on Object Oriented Data Base System, K. Dittrich (Ed), Bad-Munster, FRG, September 1988
[4] Bennis K., David B., Quilio I., Viemont Y. : Géotropics : Database Support Alternatives for Cartographic Applications, 4th Int. Symposium on Spatial Data Handling, Zurich, Switzerland, July 90
[5] Calcinelli D., Mainguenaud M. : The Management of the Ambiguities in a Graphical Query Language for GIS, 2nd Symp. on Large Spatial Databases, Zurich, Switzerland, Aug. 91, Springer-Verlag n° 525
[6] Cruz IF, Mendelzon AO, Wood PT : A Graphical Query Language Supporting Recursion, Proceedings SIGMOD Conf., San-Fransisco, USA, May 87

[7] Egenhofer M. : A Formal Definition of Binary Topological Relationships, 3rd Int. Conf. on Foundations of Data Organization and Algorithms, Paris, France, June 89
[8] Frank A., MAPQUERY : Database Query Language for Retrieval of Geometric Data and their Graphical Representation, Computer Graphics, Vol 16, n°3, July 82
[9] Gardarin and all : Managing Complex Objects in an Extensible Relational DBMS, Proc. of the VLDB Conf, Amsterdam, The Netherlands, Aug. 22-25 1989
[10] Garey MR, Johnson DS : Computer and Intractability - A guide to the theory of NP-completeness, W.H. Freeman and Co, New-York, 1979
[11] Güting RH, Geo-Relational Algebra : A Model and Query Language for Geometric Database System, Proceedings of the Int. Conf. on Extending Data Base Technology, Venice, Italy, March 88
[12] Jeansoulin R., Basic Tools for Spatial Logic, Cognitiva, Madrid, Spain, Nov. 20-23 1990
[13] Jourdas C., Mainguenaud M. : A Query Resolution Model to Manage Networks : Application to an Extended Relational DBMS, 2nd European Geographical Information System Conf. , Brussels, Belgium, Apr. 2-5 1991
[14] Jungert E. : Inference rule in a Geographical Information System, IEEE Workshop on Language for Automation, New-Orleans, USA, November 1984
[15] Kim HY and al : PICASSO : A Graphical Query Language, Software-Practice and Experience, Vol 18(3), 169-203, March 1988, Ed. J. Wiley and Sons Ltd
[16] Mainguenaud M. : Is an Extended Relational Database Management System Powerful Enough to Deal with Network Applications ?, 1st European Geographical Information System Conf., Amsterdam, The Netherlands, Apr. 9-13 1990
[17] Mainguenaud M, Raffy JL, Simatic XT : Graph Manipulations for the Network Oriented Management : Application to a Telecommunication Network, 14th Urban Data Management Symposium, Odense, Denmark, May 29-31 1991
[18] Mainguenaud M., Portier MA : CIGALES : A Graphical Query Language for Geographical Information Systems, 4th Int. Symp. on Spatial Data Handling, Zurich, Switzerland, July 22-28 1990
[19] Ooi BC, Sacks-Davis R : Query Optimization in an Extended DBMS, 3rd Int. Conf. on Foundations of Data Organization and Algorithms, Paris, France, June 89
[20] Sacks-Davis R., Mc Donnell KJ, Ooi BC : GEOQL : A Query Language for GIS, Australian and New Zealand Ass. for the Advancement of Science Congress, Townsville, Australia, Aug. 87
[21] Scholl M., Voisard A. : Thematic Map Modelling, 1st Int. Symposium on Large Spatial Databases, Santa-Barbara, USA, July 89
[22] Shu N.C. : Visual Programming, Van Nostrand Reinhold Cie, New York, 1988
[23] SQL , ISO/IEC/JTC1/SC21/WG3 ANSI X3H2-88-1, Dec 1988
[24] Stemple D, Sheard T, Bunker R : Abstract Data Types in Databases : Specification, Manipulation and Access, Proceedings of Int. Conf. on Data Engeneering, Los Angeles, USA, Fev 86
[25] Svensson P., Huang Z. : Geo-Sal : A query Language for Spatial Data Analysis, 2nd Symposium on Large Spatial Databases, Zurich, Switzerland, Aug. 91
[26] Ullman JD : Principles of Databases, Computer-Science Press, 1980