A Query Resolution Model to Manage Networks : Application to an Extended Relational DBMS

Jourdas Christophe, Mainguenaud Michel

FRANCE TELECOM
Institut National des Télécommunications
9 rue Charles Fourier
F91011 Evry - France
+ 33 1 60 76 47 82 + 33 1 60 76 47 80 (fax)
Email : MAINGUENAUD@FRINT51.bitnet

Abstract :

Network management oriented queries can be expressed using some basic operators such as graph traversal, intersection and inclusion of paths. This paper presents the resolution mechanism of a query built with such operators. The query is modeled with a functional expression. A logical optimization avoids the redundancy of common sub-expressions in order to reduce the cost of the computation time. Therefore the representation of the functional expression is transformed from a tree into a Direct Acyclic Graph. The evaluation of the DAG is performed following a breath-first philosophy. To evaluate each node of the DAG, a set of Extended SQL query are sent to the DBMS. An update propagation mechanism of the partial results guaranties the coherence of the final result.

1. INTRODUCTION

In the current researches toward the design of more powerful Data Base Management Systems (DBMS), different research groups are simultaneously concentrating their work on Geographical Information System (GIS). Now GIS needs are well known [12]. The main problems in GIS design are to define the data modelling, a Data Manipulation Language (DML), and efficient implementations of geometrical operators. Part of the GIS applications is the network managements -i.e. roads, telecommunications, railways-.
Network oriented data are often modeled with the concept of graph. The main queries are based on (1) the notion of transitive closure of a graph, (2) the notion of common parts of two graphs and (3) the notion of inclusion of a specific sub-path in a path. A query is said to be complex whenever it can not be expressed with a unique Extended SQL query [7] in an Extended Relational DBMS context.
This paper presents the formalization of the query in section 2, the logical optimization in section 3, the execution plan of a complex query in section 4, the dynamic evaluation in section 5, the graph traversal evaluation in section 6 and the conclusion in section 7.

2. THE FORMALIZATION OF A QUERY

We do not consider in this paper the data modelling of a network, the user-interface to query the database and the display of the results. The underlying data model may be a simple graph [3] or a more complex structure to take into account the logical topology of the graph [8]. The user interface is supposed to be powerful enough to handle a complex query (i.e. [9]).

A query is defined using the operators of [7].The result of each operation defined between two networks is a network. Π_D is a one-edge path between two nodes of a graph, Π_T is a graph traversal link between two nodes, Π_{Is_e} is the intersection (common sub-paths) between two paths, Π_{Is_n} is the common node operator between two set of nodes, Π_{Ic_e} is the inclusion of a

specific sub-path in a path, Π_{Ic_n} is the inclusion of a set of nodes into another one, Π_{Ic_ne} gives the set of nodes of a path. The internal representation of a query is modeled with a functional expression. The sub-queries are linked by the COMB operator.

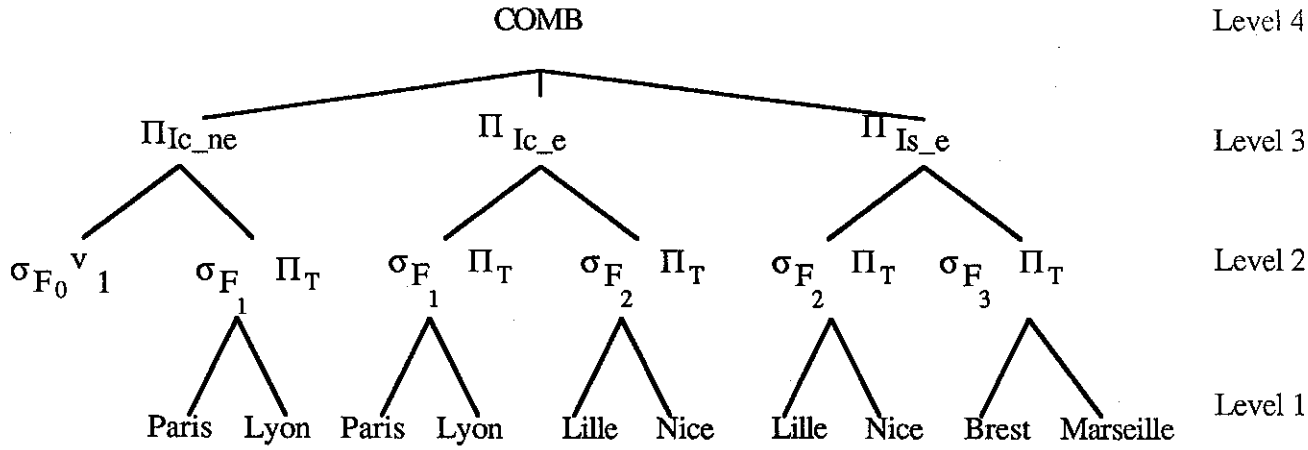
Example : I would like to know what are the common parts between a path Brest-Marseille using the TGV trains and a path Lille-Nice using the TGV or the corail trains. The path Lille-Nice must have a price less than 1500 units and must use the TGV trains at least between Paris and Lyon and I would like to know the towns of more than 100 000 inhabitants of this path Paris-Lyon.

Such a query is modeled with the following functional expression :

COMB (Π_{Is_e} (Π_T (Brest, Marseille, TGV+),
Π_T (Lille, Nice, (TGV $\vee$ corail)+, Σ cost < 1500)
),
Π_{Ic_e} (Π_T (Paris, Lyon, TGV+),
Π_T(Lille, Nice, (TGV $\vee$ corail)+, Σ cost < 1500)
),
Π_{Ic_ne} (v1 (population > 100 000),
Π_T (Paris, Lyon, TGV+)
)
)

One can remark that the towns of the path Paris-Lyon are represented by a variable, named v1. This variable is a set of towns.

This functional expression may be represented by :



σ_{F0} : population > 100 000
σ_{F1} : TGV +
σ_{F2} : (TGV $\vee$ corail) +, Σ cost < 1500
σ_{F3} : TGV +

figure 1

3. THE LOGICAL OPTIMIZATION OF A QUERY

In the previous example some sub-expressions appear several time i.e. Π_T (Paris, Lyon, TGV+). To avoid the recomputation (which may be very costly) a logical optimization is performed. The functional expression may be represented with a tree (i.e. figure 1). A unification process transforms the tree into a Direct Acyclic Graph (DAG). This unification is performed with a syntactic equality of the names and the σ_{Fi} selection criteria. This progressive algorithm is divided into two parts : the unification of the nodes and the unification of the operators.

3.1 The unification of the nodes :

In figure 1, one can remark for example that the nodes Marseille and Lyon appear several times in the functional expression. These occurrences of nodes are unified. So the figure 1 is transformed into the figure 2.

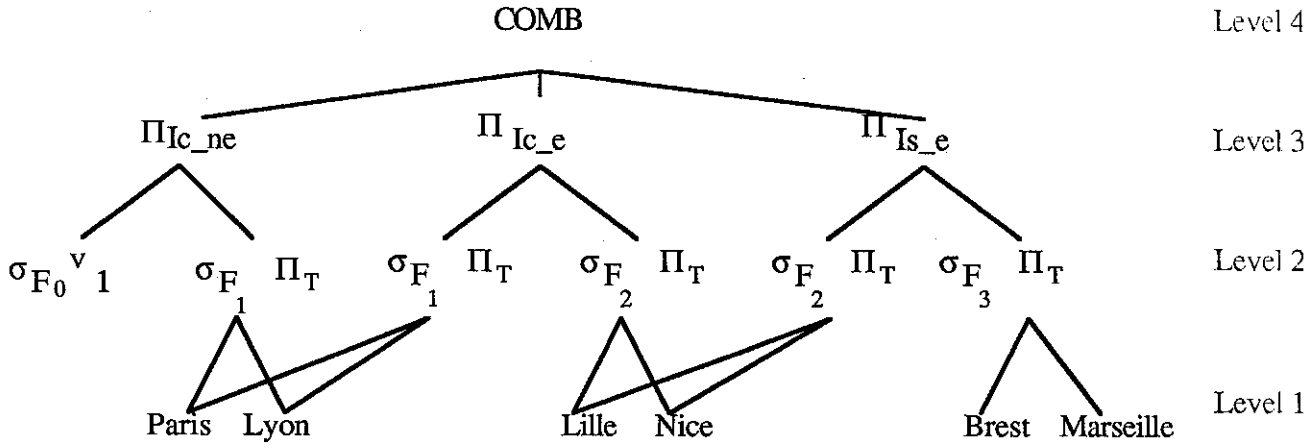


figure 2

3.2 The unification of the operators :

In figure 2, one can remark the node $\sigma_{F2}\Pi_T$ is applied on the same nodes (Lille, Nice). To avoid the recomputation of the graph traversal operator these occurrences of operators are unified. This unification is performed for all the levels i (i # 1). So the figure 2 is transformed into the figure 3.

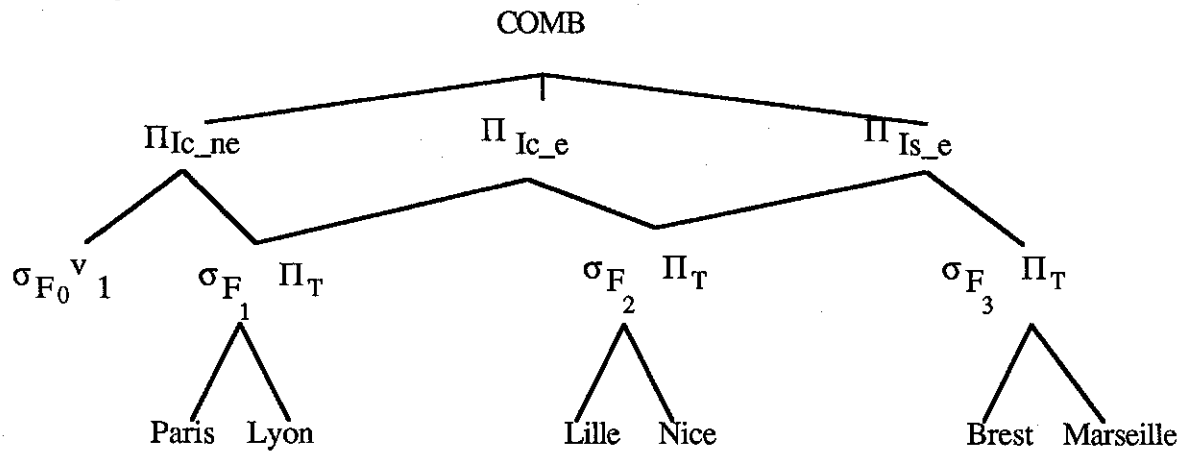


figure 3

This expression can now be evaluated. This logical optimization is entirely independent of the physical data representation. Each node of this graph may required several DBMS orders. So a global optimization may be performed [11].

4. THE EXECUTION PLAN OF A COMPLEX QUERY

The static evaluation is the compilation of the query to define "the best evaluation plan". Two steps can be defined while choosing the philosophy of the query evaluation. The first step is to defined the method of evaluating the DAG (i.e. figure 3). The second step is to defined the semantical optimization which can be performed on the execution plan.

4.1 The execution plan :

Two strategies are available to evaluate such an access plan : a depth first method or a breath first method [6].

The depth-first philosophy is a left-right-root evaluation. Following the figure 3 the order of evaluation is :

[v1, Paris, Lyon, $\sigma_{F1}\Pi_T$, Π_{Ic_ne}, Lille, Nice, $\sigma_{F2}\Pi_T$, Π_{Ic_e}, Brest, Marseille, $\sigma_{F3}\Pi_T$, Π_{Is_e}].

The breath first method consists in evaluating the DAG from the leave nodes to the root. This method introduces the notion of stratum. Following the figure 3, the different strata are :

S_0 = {v1, Paris, Lyon, Lille, Nice, Brest, Marseille}, S_1={$\sigma_{F1}\Pi_T$, $\sigma_{F2}\Pi_T$, $\sigma_{F3}\Pi_T$},
S_2 = {Π_{Ic_ne}, Π_{Ic_e}, Π_{Is_e}}, S_3 ={COMB}

This method insures the existence of all the leaves before evaluating the operators.

The breath first method is chosen as the query evaluation philosophy. The different strata introduce the notion of order between the strata. The evaluation plan is defined such as :

Let << be the precedence operator : $i < j \Rightarrow S_i << S_j$

Remark : The order of the evaluation within a stratum is not stated for the time.

4.2 The semantical optimization :

The first step of the optimization is based on the syntactic selection criteria but it does not take into account the semantics of the operators. The Π_{Is_n} and Π_{Ic_ne} operators lead to postpone the evaluation of the left hand side leave. The basic principle is the same as the frozen predicate in Prolog. One can remark that following the evaluation plan S_0, ... S_3, the evaluation of v1 is very costly since v1 does not depend on the evaluation of the path $\sigma_{F1}\Pi_T$.
To avoid such computations, the evaluation postponement is introduced and modifies the execution plan :

S0 = {Paris, Lyon, Lille, Nice, Brest, Marseille} S1 = {$\sigma_{F1}\Pi_T$, $\sigma_{F2}\Pi_T$, $\sigma_{F3}\Pi_T$},
S2 = {Π_{Ic_e}, Π_{Is_e}, Π_{Ic_ne}} S3 = {COMB}

And the evaluation of Π_{Ic_ne} leads automatically to the evaluation of v1 [7].

The semantical optimization can be improved with the management of the Π_{Ic_e} operator (i.e. to postpone the evaluation of the $\sigma_{F1}\Pi_T$ in order to extract the sub-paths from the $\sigma_{F2}\Pi_T$ operator, or to evaluate the $\sigma_{F2}\Pi_T$ operator from the results obtained by the $\sigma_{F1}\Pi_T$ operator. This optimization can take full advantage of the non-symmetry of the Π_{Ic_e} operator).

5. THE DYNAMIC EVALUATION OF A QUERY

Several ambiguities may occur while interpreting the results of the path management operators. The main troubles come from the aggregates. The following example illustrates this kind of ambiguity. Let figure 4 represents the initial network. Let figure 5 represents the result path of the query Π_T (Lille, Nice, (TGV $\vee$ corail)+, Σ cost < 1500.

figure 4 :

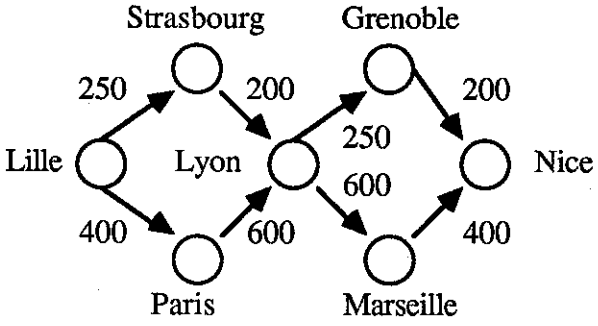
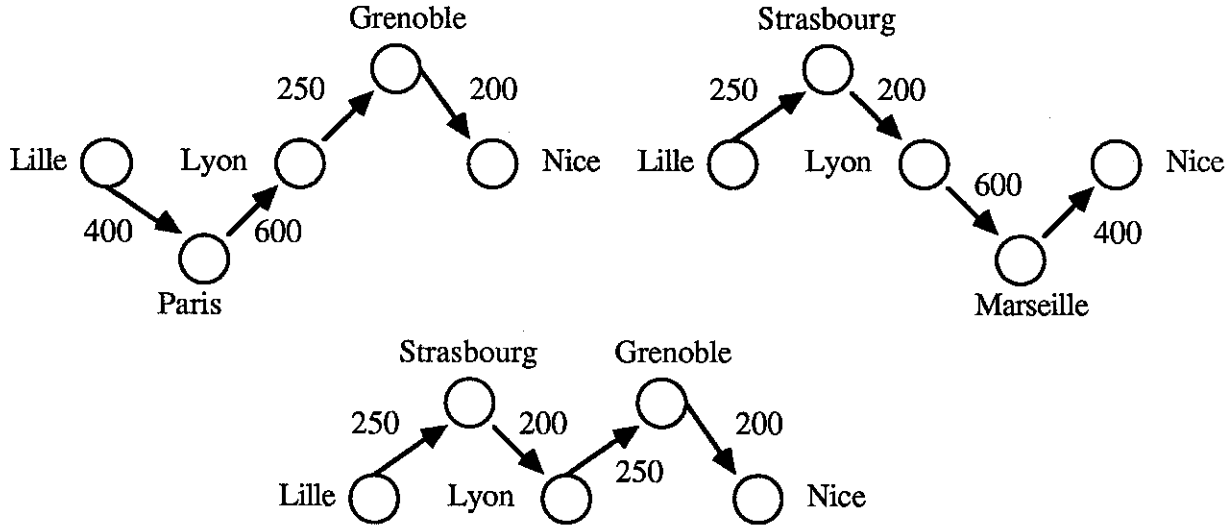


figure 5 :



Following the philosophy of the relational model, the path management operators deal with two sets of networks as input and deliver a set of networks as output.
The output is a set of networks since an ambiguity may occur if the result is defined as the union of the paths (as it is defined in [3]). Let figure 6 represents the union of the paths

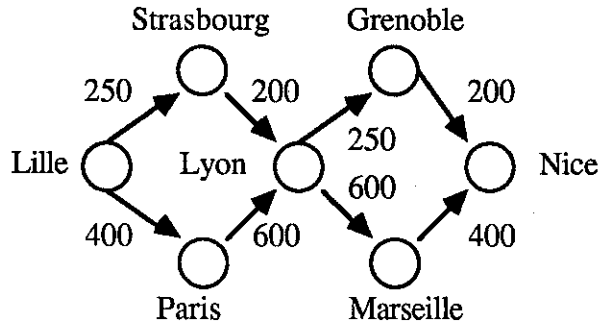


figure 6

One may conclude that the path Lille-Paris-Lyon-Marseille-Nice is a result path but it does not respect the aggregate constraint (the same ambiguity appears whenever several edges may be defined between two nodes).

Since a basic path management query may not be expressed with a unique Extended SQL statement, partial results are evaluated during the dynamic phase.
To guaranty the correctness of the final results, the evaluation process must cancel the results of the partial evaluations which do not fulfill the global query. Following the execution plan of the section 4.2 :

let P_{21}, P_{22}, P_{23} be three paths obtained by the $\sigma_{F2}\Pi_T$ operator

let P_{31}, P_{32}, P_{33} be three paths obtained by the $\sigma_{F3}\Pi_T$ operator

let $\Pi_{Is\_e}(P_{21}, P_{3i}) = \varnothing \qquad \forall i \in \{1, 2, 3\}$

The path P_{21} fulfills the partial evaluation of the $\sigma_{F2}\Pi_T$ operator but it does not fulfill the global query since there is no intersection with the paths of $\sigma_{F3}\Pi_T$ operator. This path has to be cancelled from the $\sigma_{F2}\Pi_T$ results. The (partial result) update propagation mechanism is in charge of managing these deletions. These deletions may involve several other partial results (i.e. $\sigma_{F1}\Pi_T$) since all the operators are inter-related.

6. THE GRAPH TRAVERSAL EVALUATION :

Numerous works [2] have been done on the transitive closure of a graph in a database context. This section presents the modelling of the operator, the evaluation of a path and the restriction of the implementation.

6.1 The modelling :

The Π_T operator is the key operator for the management of the networks. The data structure used in this network is a one-level graph. To simply this graph is represented by two relations :

Network (ident, origin, destination, label, geometry)
Node (ident, label, geometry)

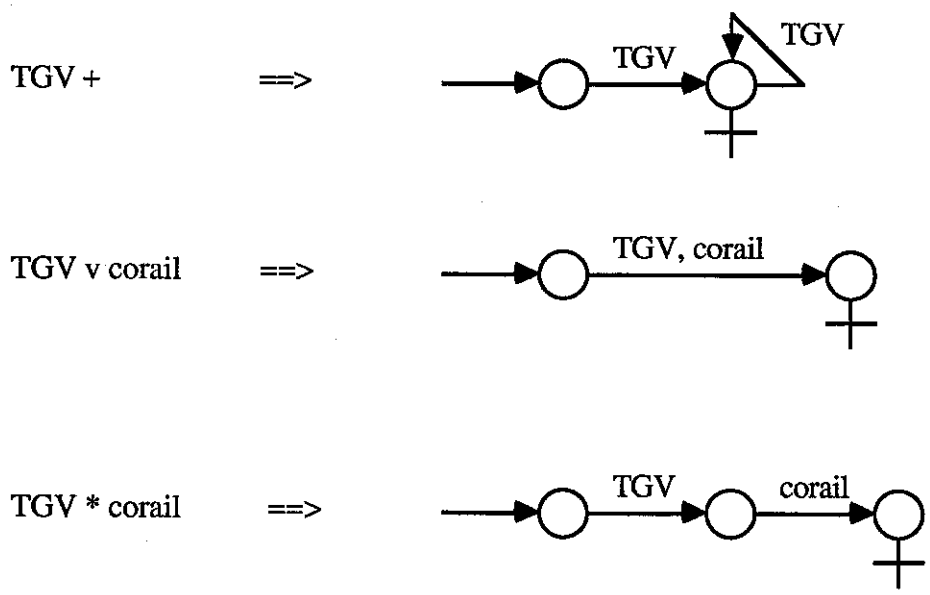
where label is a set of attributes defining the alphanumerical data (i.e. cost) of an edge (resp. of a node) and geometry is a set of attributes defining the geometrical data of an edge (resp. of a node).
The Π_T operator is defined as :

Π_T (σcriteria (Network), Initial_node, End_node, Constraints)

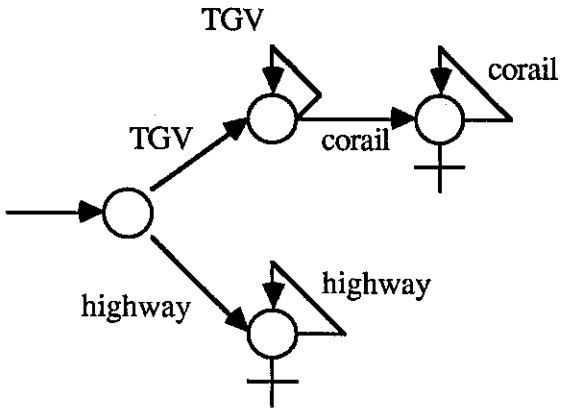
where σcriteria represents the selection criteria defined on the edges of the graph. These selection criteria allow to reduce the volume of the network data. It is important to notice that only the edges involved by the selection criteria are used in a path. A selection criterion (i.e. TGV +) is modeled with a regular expression [3]. Constraints represents the aggregates which can be defined on a path. These constraints allow to reduce the number of edges to be evaluated during the graph traversal. The classical aggregates are handled such as : Min, Max of the sum of an attribute, the average of an attribute, the number of edges, ... (i.e. Σ cost < 1500 units).

6.2 The evaluation of a path :

The graph traversal algorithm is built with a depth first philosophy. The edges are elected or rejected with a finite state automata defined from the regular expression of the selection criteria. This automata is built from the following basic steps :



The automata has one initial state and may have several final states. For example the regular expression : TGV + * corail + ∨ highway + leads to :



The concatenation of the labels of the edges represents a "word" and this word must belong to the language recognized by the automata.

6.3 The restriction of the implementation :

In the implementation we do not allow to have several initial nodes and several end nodes for a unique Π_T operator. The following functional expression is not admitted :

COMB (Π_{Ic_ne} (σ_{F0}v1, $\sigma_{F2}\Pi_T$ (Lille, Nice)),
Π_{Ic_ne} (σ_{F0}v2, $\sigma_{F3}\Pi_T$ (Brest, Marseille))
Π_T (σ_{F0}v1, σ_{F0}v2)
)

since v1 and v2 are a set of nodes (towns). Several regular expressions are also forbiden [5, 10].

To obtain the successors of a node, while evaluating a Π_T operator, a DBMS call is performed every time a node has not been already visited for the evaluation of this Π_T operator. We are fully aware that this solution is not the "best" one but it can be improved by the integration of the transitive closure into the basic operator of the DBMS instead of being an external operator.

7. CONCLUSION

A GIS query is not a simple query such as "Give me all the big towns of France". It is of prime importance to be able to deal with complex queries (queries which involve several Extended SQL statements in a database context).
To avoid the recomputations (since the operators are very costly) a more powerful language than SQL has to be defined. This language has to be user-interface independent and DBMS independent. To do so we define a functional-like query language in the context of the CIGALES project [9]. A complex query can be expressed with the basic network management operators such as Π_D, Π_T and the thematic operator such as the inclusion, the adjacency of two areas.
This paper presents the query resolution model to manage network applications. It can be extended to a general GIS (thematic data and network oriented data) under some constraints. A query is modeled with a functional expression. This expression is logically optimized with a syntactic unification to avoid the recomputations. Then an execution plan is defined with a breath first graph traversal method. The second step is a semantical optimization of the execution plan since there may exist some dependencies between the operators. The last step is to optimize this new plan with some physical storage parameters (but this step is not studied for the time).
Since a query involves several Extended SQL statements, partial results may be invalidated by the application of a new operator. To guaranty the coherence of the final result an update (of the partial results) propagation mechanism is introduced in the query resolution model.

This query resolution model is implemented in the CIGALES prototype and our next work is first to relax the constraints due to the difference of the semantics between the thematic data and the network oriented data. A thematic data is a unique element even if the physical representation corresponds to several geometrical basic elements. A network oriented datum (such as a path) is a composite element of several logical objects (i.e. an edge is a basic logical element of the thematic data). The relational model can not handle the complex object very easily even with one-level depth objects [4]. The second part of our next work is to use this query resolution model in an object oriented DBSM context [1]. The data model richer than the relational data model allows to deal with the multi-scaled networks [8].

REFERENCES

1. Bancilhon F. and al : The design and Implementation of O2, an Object Oriented Database System, Proc. of the 2nd Int. Work. on Object-Oriented Data Base Systems, K. Dittrich (Ed) Bad-Munster, FRG, September, 1988
2. Bancilhon F., Ramakrishnan R., An amateur's Introduction to Recursive Query Processing Strategies, Proc. of the ACM SIGMOD Conference, Washington - USA, May 28-30, 1986
3. Cruz IF, Mendelzon AO, Wood PT, A Graphical Query Language supporting Recursion, Proc. of the ACM SIGMOD Conference, San-Fransisco - USA May 27-29, 1987
4. Gardarin G and al, Managing Complex Objects in an Extensible Relational DBMS, Proc of the VLDB Conference, Amsterdam, The Netherlands, August 22-25, 1989
5. Garey MR, Johnson DS, Computer and Intractability - A guide to the theory of NP-completeness, W.H. Freeman and Co, New York 1979
6. Knuth D.E., The art of Programming, Addison-Wesley, 1973
7. Mainguenaud M., Is an Extended Relational DBMS Powerful Enough to Deal with Network Applications, European Geographical Information System 90, Amsterdam, The Netherlands, April 9-13, 1990
8. Mainguenaud M., Simatic X.T. : A Data Model for Multi-Scaled Network, UGI-IGU Conference - GIS Multiple Representation and Multiple Use, Brno - Czechoslovakia, April 22-25 1991
9. Mainguenaud M., Portier MA, Cigales : A Graphical Query Language for Geographical Information Systems, 4th Int. Symposium on Spatial Data Handling, July 23-27, 1990, Zürich - Switzerland

10. Mendelzon AO, Wood PT, Finding Regular Simple Paths in Graph Database, VLDB Conference, Amsterdam, The Netherlands, July 22-25, 1989
11. Park J., Segev A., Using common sub-expressions to optimize multiple queries, Data Engeneering, Los Angeles, USA, February 1-5, 1988
12. Smith TR and al, Requirements and principles for implementation and construction of large scale GIS, Int. Journal of Geographical Information Systems, vol 1, n°1, 1987, pp 13-31