A query resolution engine to handle a path operator with multiple paths

Michel Mainguenaud

Laboratoire Perception, Système et Information
Institut National des Sciences Appliquées (INSA)
Site du Madrillet
Avenue de l'Université
76800 Saint Etienne du Rouvray
France
Fax : + 33 2 32 95 97 08
Michel.Mainguenaud@insa-rouen.fr

Abstract

An end-user query of a Geographical Information System (GIS) can formally be defined as the application of a set of operators (spatial or not). Geographic Information Systems used for Transportation (GIS-T) must provide a path evaluation operator. For example, an end-user query may involve a selection based on alphanumeric criteria, an evaluation of path, and a spatial intersection. This composition of operators and the fact that the evaluation of a path may not provide a unique result impose the definition of a query resolution model or a database query language able to support this composition. In this paper we present a query resolution model. The use of multi-criteria analysis and the definition of aggregates in a query (nearly mandatory) may involve ambiguities in the final presentation of the results to an end user. The formal modeling of query results must take into account this risk. The philosophies of a query definition and the presentation of the results may be different (e.g., formular vs. visual). The management of query results must take into account the data model associated with the query results, the results themselves (metabase/database), and the interpretation. The interpretation avoids errors due to visual ambiguities of an operator with an aggregate function.

Keyword: Database, Query language, Path management, Query resolution engine, GIS

A query resolution engine to handle a path operator with multiple paths

1. Introduction

Many efforts are being undertaken to elaborate innovative solutions for the representation and exploration of complex database applications (Bauzer Medeiros and Pires, 1994). In the context of geographical databases, several spatial data models have been identified. Many proposals have been advanced to query these databases (Aufaure-Portier and Trepied, 1996; Di Loreto et al., 1996; Egenhofer, 1994; Kirby and Pazner, 1990; Meyer, 1992; Tsou and Buttenfield, 1996; Woodruff et al., 1995). Graph structures are introduced into geographic database models to handle networks. Network structures are particularly useful to represent physical or influence relationships in space. Physical networks include the distribution of electricity, gas, water or telecommunication resources. Influence networks describe economical or social patterns in space. Network structures can be also used to represent spatial navigation processes in space and time (i.e., a movement, generally a human one, between several locations in space). Such processes are represented throughout cognitive representations of space that integrate complementary levels of abstraction (i.e., large and local scales according to Kuiper, 1978). In this case, network nodes represent symbolic and discrete locations in space and edge displacements between these locations. A classic logical architecture of a Geographical Information System (GIS) is presented Figure 1. Three main levels and their interactions are presented in this figure.

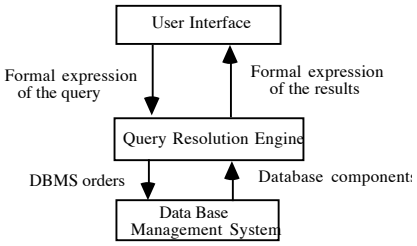

Figure 1. - Different levels of studies for GIS applications

The user interface level is responsible for the query definition process and the result display. These two tasks may follow different approaches. As an example, the query definition process may be performed through an interface based on click boxes (e.g., a formular). The result display process may be performed through an interface based on a visual language. There is no reason to constrain the query definition process to be performed on the same hardware as the query display (e.g., a query defined with real buttons and a visualization of the result on a screen). The Query Resolution Engine (QRE) is responsible for transforming an end-user query into orders that can be

understood by the Data Base Management System (DBMS). The allowed expressive power (i.e., the classes of queries an end-user can define) of the user interface must be greater than the DBMS is able to accept. Otherwise, the user interface level is not a User Interface level but a simple interface to the DBMS. Naturally, the two expressive powers (i.e., the query language of the user interface and the database query language) must be compatible. A solution based on a user interface with operators and a predicate-based query language for the DBMS does not make sense. For example, the user interface may accept in a single order the composition of operators. Because the DBMS does not accept the composition, the QRE is in charge of providing the software interface that simulates the composition. Let us consider for the time that the DBMS is only in charge of storing data and providing tools to retrieve this data. We are concerned in this research with the links between the User Interface level and the Query Resolution Engine.

Geographical Information System for Transportation (GIS-T) represents a class of GIS problems. This kind of application focuses on geographic data that can be organized with or without a spatial representation (i.e., the management of a network). Applications can manipulate a network without any knowledge (or need of knowledge) of the spatial locations (i.e., spatial coordinates) of the different entities. The manipulation is based on the concept of graphing without a spatial component. This property allows for the development of applications on very cheap hardware devices since a unique vision of alphanumeric data is required. For example, planning and management of a public bus system involves a list of buses and connections linked to a timetable and does not require a two-dimensional (2D) representation. Most of the time, a 2D representation of a network is schematic and does not respect scale. In richer applications where scale and positional accuracy are important, the logical representation of a network and its environment are merged. The quantity of data available is more important and provides the opportunity to define more complex queries than "I would like to go from place A to place B". In this case, an end-user query can formally be defined as the application of a set of operators (spatial or not). GIS-T must provide a path evaluation operator. For example, a query may be the application of a selection based on alphanumeric criteria (e.g., a route with the characteristic "highway"), an evaluation of a path (e.g., from a town named "Paris" to a town named "Nice"), and a spatial intersection operator (e.g., the path must cross a forest) to provide a unique end-user query (e.g., a route from Paris to Nice which only uses a highway and crosses a forest).

Several spatial database models and query languages have been proposed to represent the properties of networks in space (or not) (Angellacio, Catarci and Santucci, 1990; Car and Frank, 1994; Christophides, Cluet and Moerkotte, 1996; Cruz, Mendelzon and Wood, 1987; Erwig and Guting, 1994; Timpf et al., 1992). The definition of a data manipulation language that operates,

organizes, and presents a set of network queries within their geographic context remains an important research challenge to date. Database query languages are mainly based on a logic of predicates that restrict a query result to a set of tuples or objects. The projection operator restricts a query result to some of the attributes. (Note that a projection operator is also defined within object-oriented database query languages as an operator which restricts resulting attributes, hides and eventually renames or redefines object properties.) To extend the semantics of the projection operator, aggregate functions have been proposed (e.g., sum, average, maximum, minimum, count). These functions can be integrated within the projection operator in order to extend the semantics delivered by the query results. In a GIS context, a query result combines network, spatial, and alphanumeric properties. For example, routes of air flights present a network with a map as a background. Nodes have relevant geographic locations but the edges are symbolic in the sense that they do not represent the real traveling routes of the planes.

A first extension proposed to handle spatial data in query languages is the introduction of spatial predicates (e.g., in the "where" clause of an extended SQL-like query language). However, the semantics of these languages is not adapted to the complexity of geographic applications. In this domain, operations are often oriented toward the spatial and logical manipulations of entities. The introduction of spatial-oriented (or network-oriented) operators improves the benefit of spatial queries. New alphanumeric and spatial semantics may be derived from their application (Barrera and Buchmann, 1981; Mainguenaud, 1994; Claramunt and Mainguenaud, 1999). In this paper, we are concerned with the composition of operators as a query.

The consequences of the composition of operators influence two levels: the query modeling process and the result modeling process. The scope of the work reported here is limited to the conceptual modeling of such interactions, at the exclusion of the physical organization with various technologies such as Corba.

The composition of operators may lead to a situation in which the same operator with the same argument(s) is used several times in the same query. For example, a route from Paris to Nice crosses a forest and borders a polluted area (i.e., the same operator - the path operator), or a polluted area must border the non-forest part of a town as well as the forest part of this town (i.e., the same arguments - the town and the forest). Such queries can no longer be modeled with an algebraic tree (as it is in conventional DBMS) so that one must resort to a Directed Acyclic Graph (DAG). Even with the current extensions defined in SQL3, we cannot express the fact that the argument may be used by several operators during the evaluation: for instance, the interweaving of an SQL statement in a “from” clause of another SQL statement, while the inner statement does not have a “from” clause. Two solutions may be advanced to take this constraint into account. In the

first solution, the composition is introduces in the query language of the database. The composition is provided by the "from" clause (Haas and Cody, 1991) or in the "select" clause (Larue et al., 1993). This approach enables optimization at the database level. The second solution is the definition of a query resolution mechanism to handle the composition outside the DBMS (i.e., a Query Resolution Engine). The latter proposition is more portable than the former in view of the fact that the DBMS may not support the composition. This approach is followed in this research.

The QRE manages a set of operators. The expressive power of the underlying graph model used to represent a network is not considered. This graph model is very general in that it can manage planar or non-planar graphs, acyclic graphs or non-acyclic graphs, loops or non-loops, and as few or as many levels of abstraction as needed. Whereas the expressive power of the graph model is DBMS-dependent, the QRE is graph model independent. Therefore, several different formalisms can be used to define the graph model (e.g., Object-oriented, extended relational with Abstract Data Types).

The expressive power of the underlying path operator used to evaluate a path needs not be considered. The path operator can manage simple paths or complex paths, as well as variable levels of abstraction. The expressive power of the path operator is DBMS-dependent but the QRE is path operator-independent. One imposes the single restriction that the path operator must provide zero, one or several paths as an answer.

The conceptual approach of a QRE is therefore more powerful than an approach based on a single answer path operator (e.g., a shortest path). The difference stems from the fact that multiple paths exist between a given path origin and a given path destination. To provide only a unique path as an answer does not change the conventional querying of a GIS. From a conceptual point of view, the path operator is indeed similar to a spatial intersection: from a pair of arguments, a single answer is provided.

Queries commonly use aggregate measures (e.g., the total duration of a trip must be less than 10 units). These aggregates and the use of multi-criteria analysis may involve ambiguities in the final presentation of the results to an end-user. The true result is a sub-set of the Cartesian product of data involved in the global result due to the aggregate functions. The formal modeling of query results must take into account this risk (e.g., inter-dependent applications). The philosophies of a query definition and the presentation of the results may be different (e.g., formular vs. visual). The management of query results involves with three components. The first is the data model associated with the query results. The second entails the results themselves (metabase/database). The last is

the interpretation to avoid errors due to visual ambiguities of an operator with an aggregate function.

The remainder of this paper is organized as follows. The second section presents the query modeling. This is the first step to enable GIS-T database manipulations. The modeling of the query results and their basic manipulations is discussed in the following section. This is the first step to achieve data visualization to end-users. Conclusions of this work are drawn in section 4.

2. Query Modeling

An end-user query involves several (spatial or network-oriented) operators. An operator of the user's level may require the application of several database operators since the user interface and the DBMS do not manipulate at the same level of abstraction. The initial query may require the process of re-writing rules to transform the abstraction level of the initial query into the abstraction level of the DBMS (e.g., the application of an overlay operator).

A very convenient way of representing a query is the use of a functional language. For instance, in SQL, a Select From Where statement can be used in the "Where" clause of another "Select From Where" statement. The formal representation of a query is an algebraic tree: in the case of a relational algebra, leaves are the relations, intermediate nodes are the operators and the root is the result.

In our case, this tree structure must be extended to handle a composition of operators. Therefore, the structure to model a query is no longer a tree but a DAG since the same operator with the same arguments may be used several times in the same query.

Let us consider a GIS-T query applied to a database. A database schema models a transportation network with a graph data structure. To handle huge graphs, several levels of abstraction can be defined (Car and Frank, 1994; Güting, 1991; Mainguenaud, 1995). We do not consider here the expressive power of such a graph data model nor the formal model used to represent a graph (e.g., Object-oriented, extended relational model with Abstract Data Types). Application data and their manipulations (e.g., the results of a path operator) are defined on such a structure. An end-user query is transformed into a (set of) database query(ies) that involves one (or mainly several) database manipulation primitive(s). The query modeling is the formal representation merging the three previous concepts (graph, application data and database primitives).

2.1. Graph

Without any loss of generality, we consider here a graph with a single level of abstraction. We require a graph structure to illustrate the examples on transportation data. The way this graph is represented (the data model) and acquired (constructor operators) are not important at all since we are only concerned with the composition of operators in a query: this graph structure is used by a path operator. Therefore, let us use a simple definition of a graph: a graph is formally defined by a set of labeled vertices called nodes and labeled edges. Data can be associated with a graph as a whole. Figure 2 presents the formal definition of a graph G (Cruz et al., 1987). This definition also serves to model a query, in which case nodes are the operators and edges model the arguments (i.e., a conventional representation of a functional expression).

$G (N, E, \nu, \varepsilon, \Psi, \gamma)$
N is a set of nodes (i.e., labeled vertices), $N = \{n_1, \ldots, n_p\}$
E is a set of edges, $E = \{ (n_i, n_j) / n_i \in N, n_j \in N \}$
ν is a labeling function for vertices
(n_i is said to be the head and n_j the tail of an edge (n_i, n_j))
ε is a labeling function for edges
Ψ is an incident function $E \rightarrow N \times N$
γ is a labeling function for a graph

Figure 2. - Formal definition of a graph

2.2. Application data and their manipulations

The notion of a graph (an Abstract Data Type) will guide us to model the application data, whatever the data model and the semantics of the application are. The formal modeling of application data (database schema) can be defined by one of several approaches (Peckham and Maryanski, 1988; Smith and Smith, 1987; Stonebraker and Moore, 1996; Zdonik and Maier, 1990).

Let us use a traditional complex object notation to denote a sample database: [] for aggregations, {} for sets, and () for lists and conventional types integer, string, and float. Let us define an Abstract Data Type (Seshadri et al., 1997; Stemple et al., 1986) to model the cartographic representation of a node (resp. an edge) named SpatialRepresentation_type. Figure 3 presents a sample database with towns, forests, roads, and polluted areas, while Figure 4 presents the instances of this database (alphanumeric part). Without any loss of generality, we assume that all links are unidirectional. Figure 5 presents the logical view of the transportation graph, that is, the

relevant relative spatial location between nodes is missing. In this example, one mandatory link exists between Lyon and Avignon in a path from Paris to Nice, whereas several alternative paths can be followed to join Paris to Nice. Hence this is a fairly typical situation in transportation.

In the formal definition of a graph, ν is a function that provides spatialRepresentation to a node; ε is a function that provides duration and spatialRepresentation to an edge. Town_type is a subtype of Node_type, and Road_type is a subtype of Edge_type. TransportNetwork_type is a subtype of Graph_type. γ is a function that provides a visualization context for a transportation network (e.g., with some forests and some polluted areas).

```
Node_type =       [     name : string ]
Edge_type         =     [     name : string,
                              head : Node_type, tail : Node_type ]
Graph_type        =     [     name : string,
                              nodes : { Node_type}, edges : {Edge_type} ]

Town_type=        [     name : string,
                                    spatialRepresentation :  SpatialRepresentation_type]
Forest_type       =     [     name : string,
                                    spatialRepresentation :  SpatialRepresentation_type ]
PollutedArea_type       =     [     name : string,
                                    spatialRepresentation :  SpatialRepresentation_type ]
Road_type         =     [     name : string,
                                    origin : Town_type, destination : Town_type,
                                    duration : float,
                                    spatialRepresentation :  SpatialRepresentation_type ]
TransportNetwork_type =     [     name : string,
                                    nodes : {Town_type}, edges : {Road_type},
                                    context : [  forests : {Forest_type},
                                                 pollutedAreas : {PollutedArea_type} ] ]
```

Figure 3. - Sample database schema

Road

Name	Origin	Destination	Duration
1.1	Paris	Dijon	4
1.2	Paris	Dijon	4
2	Paris	Lyon	2
3	Dijon	Lyon	2
4	Lyon	Avignon	1
5	Avignon	Toulon	2.5
6	Avignon	Marseille	1
7	Toulon	Nice	2.5
8	Marseille	Nice	2

Figure 4. - Database instances (alphanumeric part)

Town : { Paris, Dijon, Lyon, Avignon, Marseille, Toulon, Nice },

Forest : {Fontainebleau, Melun}, PollutedArea : {PA1, PA2}

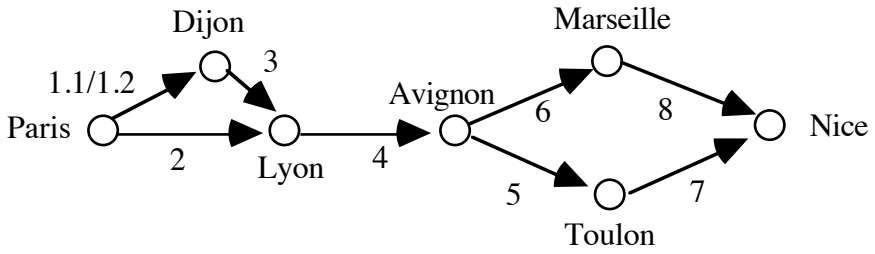

Figure 5. - Logical graph

A double spatial representation of transportation data is handled with the ε function. This allows for duration or travel time to be different. The logical representation of a given (origin, destination) pair and of the double spatial representation associated to it is depicted in Figure 6.

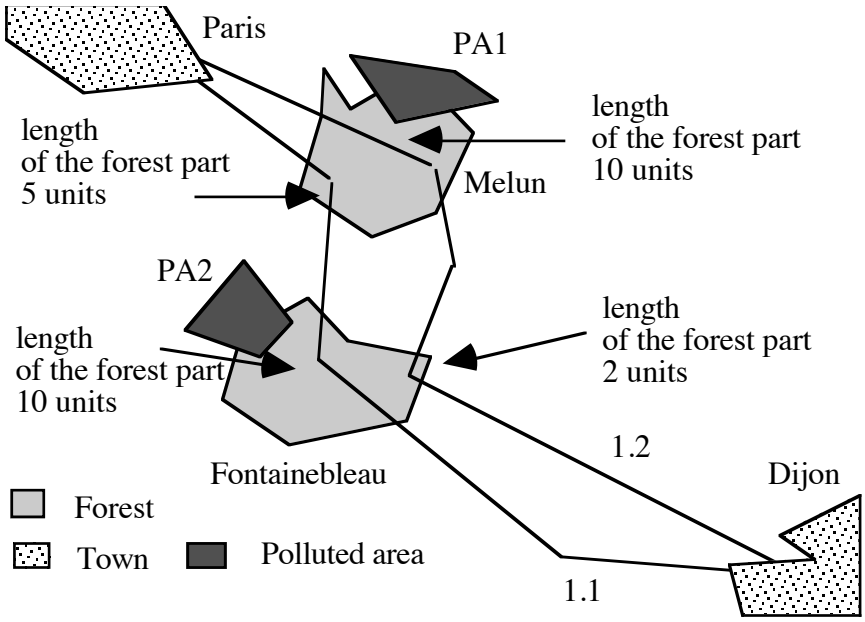

Figure 6. - Spatial representation of the edge (Paris, Dijon)

Data manipulations in a GIS-T database can be reduced to four classes. Class (a) conventional database manipulations (i.e., querying ν or ε function) are based on a selection with alphanumeric criteria: for instance , a town with a given name, "what portion of road network N is maintained by political entity P?". Class (b) conventional GIS data manipulations are based on spatial operators: for instance, an intersection operator for "a road must cross a forest", "Do the bounding boxes of two networks overlap?". This class involves the spatial representation. Class (c.1) is the evaluation of paths with a Finite State Automata-based query: the road classification must be “secondary”, then “highway”, and finally “secondary” (Cruz et al., 1987). For this query type, the constraint is evaluated at each edge of a path. Class (c.2) is the evaluation of paths with one or more aggregates: for instance, the total duration is less than 10 units. The constraint is here evaluated for the path as a whole.

Within this classification, class (b) is the only one that requires a spatial representation. Classes (c.1) and (c.2) involve a path evaluation. A path may have two representations, namely a logical one - a graph, and a spatial one. The spatial representation of a path may be used in a class (b) query.

In this paper, we are not concerned with queries of class (a) since they correspond to conventional database manipulation operators. Furthermore, classes (c.1) and (c.2) can considered as one same class (c) since we are not concerned with the query definition process (e.g., what is the kind of user interface?; how many paths are required?; what is the value of the aggregate?) or the

query evaluation process (e.g., what algorithm should be used to evaluate a path?; what is the "best" optimization?). To illustrate transportation manipulations on our sample database, let us define the following set of queries to tackles the problems of GIS-T:

Query Q_1: I would like to join Paris to Nice;

Query Q_2: I would like to join Paris to Nice with a total duration less than 10 units;

Query Q_3: Show me a map with a road(s) that crosses, with at least a 10-units length, a forest (with a polluted area) in its non-polluted part?

Query Q_4: Show me a map with a route(s) from Paris to Nice that crosses a forest and borders a polluted area?

Query Q_1 requires the evaluation of a transitive closure on a graph G defined in the database (i.e., class c). Query Q_2 has the same requirement extended with an aggregate function (i.e., path evaluation under constraints). Query Q_3 requires the composition of two spatial operators (an intersection applied on the result of a spatial difference), i.e., class (b). Query Q_3 can be generalized as a mix with query Q_1 to involve the path operator (road vs. path from Paris to Nice). Query Q_3 represents a vertical composition (i.e., the algebraic representation is a tree). Query Q_4 is also a query with a composition of operators. It represents a horizontal composition (i.e., the algebraic representation is a DAG).

In the following, we consider that some data types are available (e.g., Town_type), and some queries are defined to illustrate the different configurations (e.g., Query Q_3).

2.3. Database primitives

The output of the Query Resolution Engine to the DBMS depends on the expressive power of the database query language. The introduction of predicates or operators as primitives of a spatial database query language leads to different expressive powers (and therefore some queries may or may not be expressed).

A spatial predicate (e.g., spatial difference) applied to data provides a Boolean result (e.g., a basic manipulation in a “Where” clause in SQL). As an example, query Q_3 cannot be formulated with a predicate-based language since it requires the results of the spatial difference operator to be able to evaluate whether an intersection exists or not (i.e., using the result of an operator). The

application of a GIS to transportation data introduces several consequences for the query language definition:

- Operator-based language: The extension to transportation management requires the definition of an operator-based language. The important point in the evaluation of a path between Paris and Nice (e.g., query Q_1) is the components (i.e., edges and nodes) of the path. Information, such as a path that exists (i.e., evaluation with a predicate) is not relevant for GIS-T application (i.e., the result of the path operator instead of a Boolean result). To manage transportation data, GIS must handle other graph manipulation operators such as node/edge/graph manipulations.

- Closedness of operators: The underlying notion of a labeled graph requires that a result of a graph manipulation should be compatible with the notion of a labeled graph. This requirement is an extension of spatial data manipulations. Geographic data is composed of at least one pair of alphanumeric and spatial data. A GIS operator must provide as a result a datum with the same structure (a pair of alphanumeric and spatial data). As an extension, a GIS datum for transportation is a geographical datum with an associated graph. The result of an operator must be a geographical datum (i.e., alphanumeric and spatial data) and an associated graph. In the context of a network, the spatial part may be a symbolic representation (i.e., the real world coordinates are not mandatory).

- Expressive power: Once the query language provides the concept of operator, the next problem is to define data on which operators are applied. Traditional database query language (e.g., relational algebra) is based on the first order logic without function. To provide a realistic expressive power to develop applications, query languages (e.g., SQL) introduce some basic functions/operators (e.g., count). GIS-T must provide spatial and graph manipulation operators. In parallel with GIS databases, two levels can be defined: with and without a combination of operators. To guarantee the consistency of query results, the combination of operators must be provided with a single database order. Otherwise, an application routine must be provided to simulate a single database order (i.e., the QRE).

In the following, we consider that the expressive power of the database query language is at least an operator-based language providing the closedness of operators and not allowing the composition of operators in the equivalent of the "Select/From" clause.

2.4. Query modeling

In parallel with database languages, two levels of interaction can be defined: logical and operational. A logical order is independent of implementation. An operational order takes into account the different strategies of the database query resolution engine (e.g., indexes, statistics). We are here only concerned with a logical order since operational orders are hardware and software dependent - even if providing an optimized query is a very challenging task.

Several formalisms can be used to model a query, namely functional, procedural, and declarative. Let us use an intuitive functional approach and simplified grammar. A label (i.e., an integer - to be able to use the same operator with the same arguments several times in the same query) identifies an operator. This operator has a name and a list of arguments. For example, query Q_3 can be formalized by:

Query (1 ↔ (2 DB (Road), 3 Δ (4 DB (Forest), 5 DB (PollutedArea))))

where ↔ denotes the spatial intersection operator, Δ is the spatial difference operator, DB is the querying operator of a basic database component, and "i" is the label of an operator. Such a formalism can easily be represented with a DAG, without isolated node. With this approach, data and query are both represented with the concept of graph. Figure 7 presents the definition of a query.

$G (N, E, \nu_q, \varepsilon_q, \Psi, \gamma_q)$

N, E and Ψ have the same definitions than previous

ν_q denotes the labeled function for nodes (e.g., operator, database component constraint)

ε_q denotes the labeled function for edges (e.g., an order in the list of arguments to handle non-symmetric operator such as the path operator)

γ_q denotes the labeled function for a query (e.g., the required database model as a result).

Figure 7. - Definition of a query

Figures 8 and 9 portray the algebraic expressions associated with query Q_3 and query Q_4, respectively. The border operator is represented by "<>" and the path operator is represented by "->".

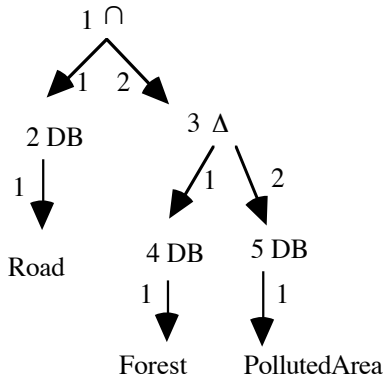

Figure 8. - Algebraic expression of query Q_3

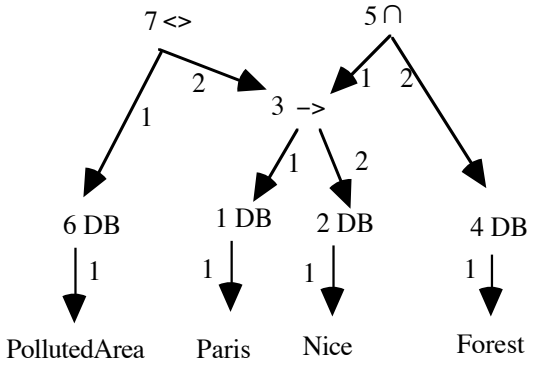

Figure 9. - Algebraic expression of query Q_4

The definitions of a transportation database and a query are similar. They correspond to a Graph_type type: Database_type = Graph_type and Query_type = Graph_type. The query modeling represents the input of the QRE. The result modeling represents the output. The added value of the QRE is introduced in the output.

3. Result Modeling

The evaluation of a path operator in a GIS-T database should provide a set of paths instead of a single path as a result. The QRE is not responsible for the definition of the relevant number of paths to be evaluated. This task belongs to the interface level or the database operator. For example, in a GIS application for tourism management and information, a shortest path may be far

from desirable since visitors are more motivated by landmarks and other sightseeing attractions that by efficiency considerations. Furthermore, the less expensive path may also be the longest. In the context of a multi-criteria analysis, it does not seem realistic to define a priori query. Human interaction may be necessary to make a choice from a small set of propositions. To reduce the volume of possible answers, a wide range of aggregate functions ought to be provided during the query definition process (e.g., total travel time under 10 units).

Aggregate functions may involve some ambiguities while the end-user interprets the results of a query. The query result modeling must provide a structure to avoid such ambiguities. To manipulate this structure, some operators must be defined.

3.1. Consequences of aggregate functions

If the result of a path operator in a GIS-T query is considered as a unique graph, modeling the set of paths (i.e., a set-oriented approach) may lead to confusion when an aggregation function is involved in the query. The result must therefore be a set of graphs.

A path is itself a graph, that is a set of nodes and a set of edges, along with some characteristic data (γ function). Let us consider the result presented to the end user in a single graph (G_r). This graph is built as the union ($\approx$) of the sets of nodes (resp. edges) of each path. Let n be the number of paths G_i (N_i, E_i, ν, ε, Ψ, γ_i), where i = 1, .., n. The result of the path operator is:

$$G_r = (\approx N_i, \approx E_i, \nu, \varepsilon, \Psi, \gamma_i) \; \forall \; i = 1, .., n$$

This formalism can be provided when no aggregate function is involved in the query (e.g., query Q_1). It cannot be used anymore once an aggregate function is involved (e.g., query Q_2). As an example, the following paths are provided while query Q_2 is evaluated (Figure 4):

(1) Paris - Dijon - Lyon - Avignon - Marseille - Nice : Duration 10
(2) Paris - Lyon - Avignon - Marseille - Nice : Duration 6
(3) Paris - Lyon - Avignon - Toulon - Nice : Duration 8

The following path:

(4) Paris - Dijon - Lyon - Avignon - Toulon - Nice

does not fulfill the requirements since its total duration is 12, which is greater than 10.

The generation of a graph built with the union of the different paths that fulfill the requirement leads to two difficulties:

- The answer is wrong. The union provides the same graph as the one presented in Figure 5 and therefore, one can infer that path (4) is relevant, This impossibility has been lost by the application of the union operator.

- The result for the aggregate function is lost. It must be re-computed as soon as a path is required since the γ function cannot individualize each path. This disappearance is due to the union operator.

A set of paths results from the path operator. Each path in this set is defined independently of the others. The result of a path operator with multiple paths as an answer cannot be managed as a shortest path would be. A path operator is applied between Paris and Nice in our example. Three paths are provided as an answer. This problem can be generalized. The only restriction is to provide a realistic computational time (i.e., several starting places for a single arriving place or the opposite).

3.2. Query results

Query results depend on the expressive power of the language used to define a query. A predicate-based query language leads to a result built as a set of database components (e.g., relations for relational-based DBMS, classes for Object Oriented DBMS). An operator-based query language leads to a result built as a set of database components and the results of operators defined in the query. The query and the result are both graphs that are isomorphic. The leaf nodes (that is, a node such as it does not exist an edge with this node as a head) of the graph modeling the results are the database components (Figure 8). The non-leaf nodes are the results of the operators. Each node represents a set of instances. Figure 10 presents a sub-graph of query Q_3 in Figure 8 with data from Figure 6.

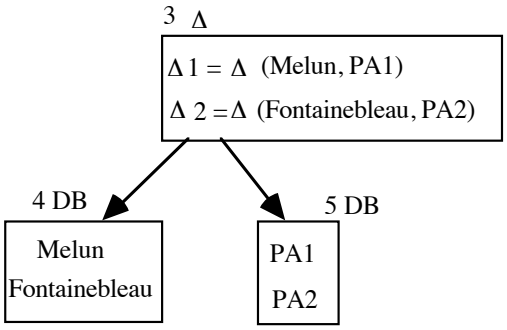

Figure 10. - Query Q3: a sub-graph for the results

A conventional interpretation of the principle is provided with the following rule for a binary operator (Figure 11): "An instance number k of the result of an operator (e.g., spatial difference, Δ) labeled K is built with an instance number i of an operator labeled I (e.g., database query, DB) and an instance number j of an operator labeled J (e.g., database query, DB)". The sub-query is:

K Δ (I DB (...), J DB (...)).

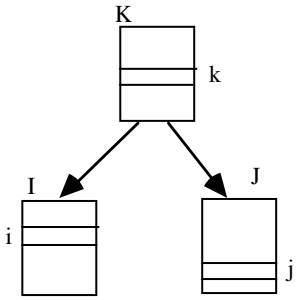

Figure 11. - Example of conventional interpretation (sub-graph)

The same sub-query may be used several times in the same query (i.e., the graph modeling a query is not a tree). For example, Figure 9 presents a horizontal composition with an operator. Figure 12 presents a horizontal composition with a leaf, that is a simplified graph associated with the following query: "a road crosses the non-polluted part of a forest and borders (<>) this polluted area" - i.e., an extension of query Q_3. The functional representation is:

Query (1 ⇔ (2 DB (Road), 3 Δ (4 DB (Forest)), 5 DB (PollutedArea)),
6 <> (2 DB (Road), 5 DB (PollutedArea))).

The evaluation of such a query is similar to a conjunction of queries. Therefore, the evaluation mechanism will end since only deletions are involved. The deletion of non-relevant data is a recursive application of the evaluation since the query is modeled with a DAG instead of being modeled with a tree. Some arguments may have a reduced set of relevant data when they share the same operator: for instance, a forest may be deleted from the relevant data set if it has a polluted part, but the polluted area is not bordered by a road that crosses this forest.

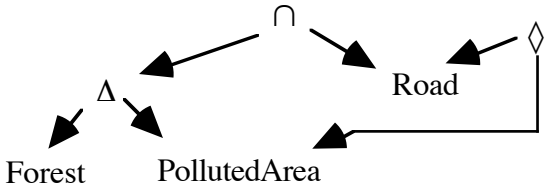

Figure 12. - Query graph

The interpretation is different for a path operator whose answer is multiple paths because the application of the path operator on an instance number i (the origin) and on an instance number j (the destination) does not provide a single result (i.e., an instance number k) but a set of instances (a set of paths). Since aggregate functions are almost unavoidable in a path evaluation, the result must be a set of graphs (and not a unique graph). This restriction leads to some troubles while defining in the query language the signatures of operators manipulating the result of a path operator since the result is a set of graphs.

In conventional GIS interpretation (Figure 10), a given pair of entities (say, Melun and PA1) provides a unique answer (resp., Δ1). When aggregate functions are involved in a query, the result of a path operator can no longer be defined as the union of the different paths. Therefore, from a given pair, several results are provided.

Two modes can be defined depending on the expressive power of the query language: the Union mode (a graph as a result) and the Elementary-at-a-time mode (a set of graphs as a result). The Union mode is appropriate to predicate-based query language or to a query in which no aggregate function is involved. It is also appropriate with a path operator providing a unique result by definition (e.g., a shortest path operator). As discussed above, when an aggregate function is involved in a query (i.e., in a path operator or on the result of a spatial operator), the Union mode can no longer be used: the Elementary-at-a-time mode becomes mandatory.

A mode is independent of a path operator. A query is defined as a graph with labeling function γ. The γ function is in charge of defining the relevant mode since its determination can be made before the evaluation or after the evaluation if the result is a unique path. Visual ambiguities may arise from the path operator or from a different operator in the query. Query Q_2 and the generalization of query Q_3 (i.e., with the query Q_1 that does not require an aggregate function) are examples of such queries. Query Q_2 involves an aggregate function in the path operator. The mode must be Elementary-at-a-time to ensure a set of graphs as a result. The generalization of query Q_3 involves a constraint on a spatial intersection operator (i.e., at least a 10-unit length). Figure 6 illustrates the visual ambiguity that the definition as a Union mode arises. The two forests are relevant and so are the two roads cross these forests. But for each road, one of the intersections is not relevant because too short: for the pair (1.1, Melun) the intersection is only 5 units, while for the pair (1.2, Fontainebleau)), it is 2 units.

Since ambiguities may arise, an interpretation structure must be provided to define the relevant instances used as arguments. An interpretation structure defines the relevant instances of the query result (i.e., instances of the database components) and the associated results for each operator. Its organization is therefore similar to a query and can be modeled by a set of graphs (isomorphic to a result graph). Each graph of this set defines an unambiguous combination of database instances that fulfills the query requirements. The manipulation of an instance in the interpretation structure is similar to the manipulation of a result graph in a Union mode (they can be therefore substituted). A node contains a pair (label of the operator, identifier of an instance). The semantics of an edge is similar to a result graph one (i.e., an edge links an instance of the result to the instances used as an argument of an operator - Figure 11). Figure 13 presents an interpretation structure for query Q_3 (Figure 8 / Figure 9) with data from Figure 6. Interestingly, in these graphs, a link does not exist between the road labeled 1.1 (resp. 1.2) and the forest named Melun (resp. Fontainebleau) since the length of the intersection is not long enough. The result is a subset of the Cartesian product of data involved in the global result (see Figure 10). The definitions of a transportation database, a query, and associated results can be extended as follows:

Result_type = [Data : Database_type,
Interpretation: (Graph_type)].

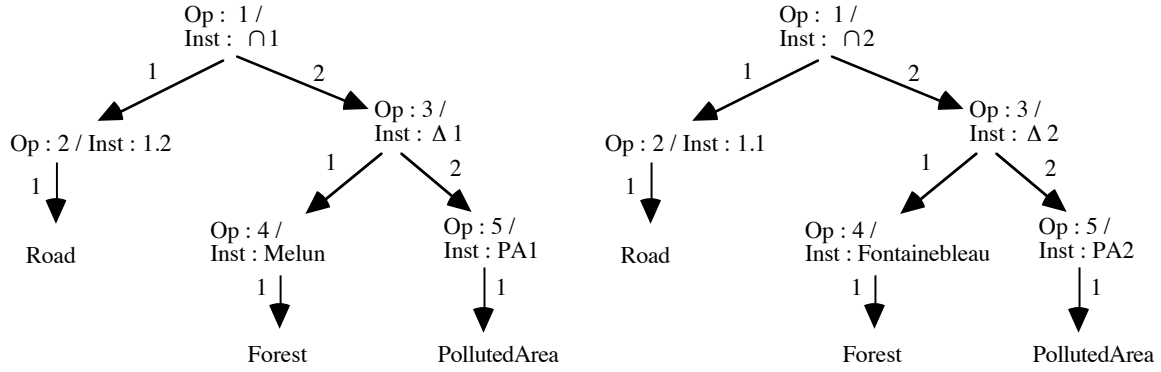

Figure 13. - Interpretation graphs for query Q_3

A query and the query results can be modeled with a graph. The links between the User Interface and the QRE and vice versa are based on the graph concept. From the User interface an exchange structure is defined as follows:

FromUIto QRE_type = [Query : Query_type].

From the Query Resolution Engine, an exchange structure is defined as follows:

FromQREtoUI_type = [Query : Query_type,
Result : Result_type].

From this structure of communication, logical manipulations can be defined.

3.3. Logical manipulations

It is useful to provide the formal representation of a query in the FromQREtoUI_type type since the philosophy of the query definition process and the query result display may be different (and the restitution devices, too). This allows a user interface in its part of display result to establish a link with the query. For example, the user may query the map obtained as a result but can also do the opposite and have get information from the query (e.g., show me the intersection of a forest and a polluted area in query Q_3).

Logical manipulations depend on the mode of a query. A Union mode is similar to a GIS that provides a path operator with a unique result (e.g., a shortest path). The result is a single map. An Elementary-at-a-time mode requires the definition of new primitives to manipulate query results. The manipulation is based on the definition of the FromQREtoUI_type type. Within this structure, the User Interface must be able to manipulate the Interpretation field.

A recursive decomposition of the User Interface (UI) level is presented in Figure 14. The human interface is in charge of communicating with an end-user. The UI Resolution Engine (UIRE) interprets and manages a query or a result manipulation. A result manipulation can be assimilated to a query on a reduced set of data. As the query belongs to the FromQREtoUI_type type, it can be presented with a result. The user can define a "new query" as "I would like to see the results of this sub-part of the query": for example, show me the intersection between the forest and the polluted area in query Q_3. The data management layer handles the FromQREtoUI_type type.

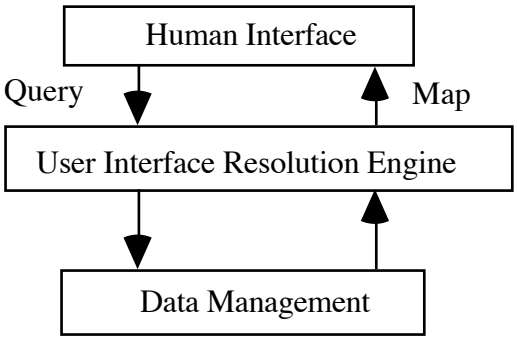

Figure 14. - Decomposition of the User Interface level

The interaction between the human interface level and the UIRE can be formalized with a conventional set of GIS manipulations, such as zoom or change of the legend). The interpretation graph (Figure 12) of a query involving manipulations with aggregate function provides several possibilities as a result. From an ordered set of graphs (i.e., a list), conventional manipulations consist of being able to initialize the process (Init), being able to present the next solution (Next), being able to return to the previous solution (Previous), being able to retain a solution (Handled), being able to select a retained solution (Select), and being able to cancel a "handled" solution (Cancel). A manipulation type, Manipulation_type, is defined to model the relevant operations:

Manipulation_type = (Init, Next, Previous, Handled, Select, Cancel).

Whenever an ambiguous query is presented, several solutions have to be displayed. The basic manipulation has the following signature:

DisplayAmbiguousResult : FromQREtoUI_type x Manipulation_type ->
Graph_type.

The graph obtained by the DisplayAmbiguousResult function can be processed as if the query was in mode Union since no visual ambiguity may occur. The context is similar to a manipulation with a GIS that provides a shortest path operator (i.e., a single path as a result).

4. Conclusion

The large diffusion of spatial database applications in scientific, planning, and business domains leads to the emergence of new requirements in terms of data representation and derivation. Particularly, current spatial data models and query language operations have to be extended in order to integrate a more complete semantic representation of complex domains. The integration and representation of graph structures within spatial data models is of particular interest for many application areas involved in the management of transportation networks.

Transportation data are modeled with the concept of a labeled graph. In this paper, the starting point is an end-user query defined as a set of operators. This query is modeled with the notion of graph. The result of a query is modeled with a graph, which is isomorphic to the query graph. This choice allows an homogeneity between the modeling of the different components of a GIS database manipulation: data, query, and results are modeled with the same concept. To avoid

visual ambiguities that may arise once an aggregate function is involved in a query, an interpretation structure is provided to define the relevant instances of query results. This structure is also modeled with a graph (isomorphic to a result graph). The approach is of great significance in transportation applications because aggregate functions are nearly mandatory to reduce the amount of data and to provide a realistic computation time.

The decomposition of a GIS logical architecture into three independent layers (a User Interface, a Query Resolution Engine and a Data Base Management System) allows for greater flexibility in the management of the expressive power. A formal communication between these components (a graph structure between the interface and the query resolution engine, an extended SQL-query language from the Query Resolution Engine to the DBMS) allows for easily changing one of them. An end-user query is a composition of operators (spatial or not). The aim of a QRE is to fill the gap between the level of abstraction of the User Interface and the level of abstraction of the DBMS (without composition of operators, with composition that does not allow the same operator with the same arguments to be used several times, and with a total composition). The introduction of a path operator that provides several results requires changing the interpretation of the final result once an aggregate function is involved in the query. Two modes are defined: the Union mode (a set oriented approach) and the Elementary-at-a-time mode (path by path). New manipulation functions are associated to the Elementary-at-a-time mode to handle the risk of visual ambiguities.

The introduction of a GIS as a decision tool requires the opportunity to propose several solutions. The added value of a human being is the decision. The user interface must be able to present different solutions without any visual ambiguity. Using the framework of the Query Resolution Engine, several concrete interfaces may be provided depending on the end-user (obviously it requires that the path operator of the database provide several paths as a result instead of a single path).

References

Angellacio, M., Catarci, T., Santucci, G., 1990. QBD*: A graphical query language with recursion, IEEE Transaction On Software Engineering 16(10), 1150-1163.

Aufaure-Portier, M.A., Trepied C., 1996. A survey of query languages for geographic information systems. 3rd International Workshop on Interfaces to Databases, Edinbugh, UK, 8-10 July.

Bauzer Medeiros, C., Pires, F., 1994. Databases for GIS, SIGMOD Record 23(1), 107-115.

Car, A., Frank, A., 1994. General Principles of Hierarchical Spatial Reasoning, - The Case of Wayfinding. 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland, 5-9 September.

Christophides, V., Cluet, S., Moerkotte, G., 1996. Evaluating queries with generalized path expressions. Proceedings of ACM SIGMOD Conference, Montreal, Canada.

Claramunt, C., Mainguenaud, M., 1999. A revisited database projection operator for network facilities in a GIS, Informatica 23, 187-201.

Cruz, I.F., Mendelzon, A.O., Wood, P.T., 1987. A graphical query language supporting recursion. Proceedings of ACM SIGMOD Conference, San Francisco, USA, 27-29 May.

Di Loreto, F., Ferri, F., Massari, F., Rafanelli, M., 1996. A pictorial query language for geographical databases. International Workshop on Advanced Visual Interfaces, Gubbio, Italy, 27-29 May.

Egenhofer, M., 1994. Spatial SQL: A query and presentation language. IEEE Transactions on Knowledge and Data Engineering 6(1), 86-95.

Erwig, M., Güting, R.H., 1994. Explicit graphs in a functional model for spatial databases. IEEE Transaction on Knowledge and Data Engineering 6(5), 787-804.

Güting, R.H., 1991. Extending a spatial database system by graphs and object class hierarchies. International Workshop on Database Management System for Geographical Applications, Capri, Italy, May.

Haas, L.M., Cody, W.F., 1991. Exploiting extensible DBMS in integrated geographical information systems, 2nd Symposium on Large Spatial Databases, Zurich, Switzerland, August 1991, Lecture Notes in Computer Science 525, Springer-Verlag, Heidelberg, pp. 423-450.

Kuiper, B., 1978. Modelling spatial knowledge. Cognitive Science 2, 129-153.

Kirby, K.C., Pazner, M., 1990. Graphic map algebra. 4th International Symposium on Spatial Data Handling, Zurich, Switzerland, 22-28 July.

Larue, T., Pastre, D., Viémont, Y., 1993. Strong integration of spatial domains and operators in a relational database system. In Advances in Spatial Databases, Abel, D.J., Ooi, B.C. (Eds). Springer-Verlag, Singapore, Lecture Notes in Computer Science 692, pp. 53-71.

Mainguenaud, M., 1994. Consistency of geographical information system query result, Computers, Environment and Urban Systems 18, 333-342.

Mainguenaud, M., 1995. The modeling of the geographic information system network component. International Journal of Geographical Information Systems 9(6), 575-593.

Meyer, B., 1992. Beyond icons: towards new metaphors for visual query languages for spatial information systems. In 1st International Workshop on Interfaces to Database Systems, Cooper R. (Ed.), Springer-Verlag, Heidelberg, pp. 113-135.

Peckham, J., Maryanski, F., 1988. Semantic data models. ACM Computing Surveys 20(3), 153-189.

Seshadri, P., Livny, M., Ramakrishnan, R., 1997. The Case for Enhanced Abstract Data Types. 23rd International Conference on Very Large Databases, Athens, Greece.

Smith, J.M., Smith, D.C., 1987. Database abstraction: aggregation and generalization. ACM Transaction On Database System 2(2), 105-133.

Stemple, D., Sheard, T., Bunker, R., 1986. Abstract Data Types in databases: specification, manipulation and access. Proceedings of the 2nd International Conference on Data Engineering, Los Angeles, CA, USA, 6-8 February.

Stonebraker, M., Moore, D., 1996. Object-Relational DBMSs: the next great wave. Morgan Kaufmann, San Mateo, CA.

Timpf, S., Volta, G.S., Pollock, D.W., Egenhofer, M., 1992. A conceptual model of wayfinding using multiple levels of abstraction. Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Frank, A.U., Campari, I., and Formentini, U. (Eds.), Springer-Verlag, Berlin, pp. 349-367.

Tsou, M.H., Buttenfield, B., 1996. A direct manipulation interface for geographical information processing. 7th International Symposium on Spatial Data Handling, Delft, The Netherlands, 12-16 August.

Woodruff, A., Su, A., Stonebraker, M., Paxson, C., Chen, J., Aiken, A., Wisnovsky, P., Taylor C., 1995. Zooming and tunneling in Tioga: Supporting navigation in Multidimensional Space. Visual Database Systems 3: Visual Information Management, Spaccapietra, S., Jain, R. (Eds.), Chapman and Hall, London, pp. 323-332.

Zdonik, S.B., Maier, D., 1990. Readings in object-oriented database systems, Morgan Kaufmann, San Mateo, CA.