16

Manipulations of Graphs with a Visual Query Language: Application to a Geographical Information System

Brossier-Wansek A.,** - Mainguenaud M.**
** France Telecom - Institut National des Télécommunications*
9 rue Charles Fourier F91011 EVRY - FRANCE
*** Cellule d'Etudes en Géographie Numérique*
16bis Av. Prieur de la Côte d'Or F94114 ARCUEIL Cedex - FRANCE
+ 33 1 60 76 47 82 (tel) + 33 1 60 77 20 06 (fax)
Email: Anne.Wansek@int-evry.fr, Michel.Mainguenaud@int-evry.fr
http://etna.int-evry.fr/BasesDeDonnees/Cigales/cigales.html

Abstract
Operators for geographical databases can be classified into two categories: thematic-oriented operators and network-oriented operators. Thematic-oriented operators are based on geometric representations (e.g., intersection, inclusion, adjacency). Network-oriented operators are based on graph manipulations whatever the geometric representation is (e.g., a transitive closure of a graph).
In this paper we present network manipulations with a visual query language for a geographical database. A query is defined with a visual expression defined with a Query-By-Example approach. The user draws a graphic representing the properties results have to verify.

Keyword
Visual query language, Graph manipulations, Geographical Information System

I INTRODUCTION

In current research toward the design of more powerful tools for urban planning and remote sensing, different research groups are simultaneously concentrating their work on Geographical Information Systems (GIS). GIS needs are very well known (Smith, 1987). Nevertheless, several problems are still open.

The need for a spatial query language has been identified. Traditionnally, operators for geographical databases are classified into two categories: thematic-oriented operators and network-oriented operators. This distinction is due to the lack of common data structures (and operators). Thematic-oriented operators are based on geometric representations (e.g., intersection, inclusion, adjacency). Network-oriented operators are based on graph manipulations whatever the geometric representation is (e.g., a transitive closure of a graph).
Visual languages are very promising. Propositions for such languages are detailed for example in (Angelaccio,1990) (Cruz, 1987) (Kim, 1988) (Kirby, 1990) (Myers, 1992) (Shu, 1988). A query

S. Spaccapietra et al. (eds.), *Visual Database Systems 3*

language is said to be visual whenever the semantics of a query is expressed by a drawing. It is said to be declarative whenever a query specifies the properties to be verified by the results, but not the way of obtaining them (i.e., imperative). This paper is based on the Cigales experiment. Cigales (Calcinelli, 1991) (Calcinelli, 1994) is a visual and declarative query language for GIS. We focus, here, on the design of a visual query language to handle networks in a GIS. This issue is one of the most important to deal with when building spatial information systems. It will be increasingly true since the geographic information is now affecting non-specialist users and general applications such as tourism.

Part II presents a brief overview of the graph data model; Part III, the graph manipulation part of a visual query language for GIS; Part IV, the conclusion.

II BRIEF OVERVIEW OF THE GRAPH DATA MODEL

The user graph data model is based on graph theory (Berge, 1983) and is similar to the definition presented in (Angelaccio,1990) (Cruz, 1987). The database graph data model is based on the graph data model defined in (Langou, 1994) (Mainguenaud, 1991). Nevertheless, this data structuring is not visible from the user interface level. In this part, we present the user graph data model, its associated operators and a toy database used along this paper.

II.1 User graph data model

The user graph data model (i.e., visible from the interface level) is based on the concepts of nodes and edges. A graph is a triplet G (N, E, Ψ) where N is a set of nodes, E is a subset of the Cartesian product N x N and Ψ is an incidence function mapping E to N x N. The formal definition is presented in Figure 1.

A graph G (N, E, Ψ) is defined as follows:

$N = \{n_1, \ldots, n_p\}$

$E = \{ (n_i, n_j) / n_i \in N, n_j \in N \}$

(n_i is said to be the initial node and n_j is said to be the end node for an edge (n_i, n_j))

Figure 1 - Formal definition of a graph

A node is used to model for example a town. An edge is used to model for example a link between two towns. In this paper, a graph is considered as orientated (i.e., the order, initial node / end node, is relevant). Nodes and edges are labelled. Several edges may be defined with the same initial and end nodes. We do not limit this number but we do not allow an edge (n_i, n_i). We extend the graph definition with a labelling function, ν, defined on nodes and with a labelling function, ε, defined on edges. Let D_1, ..., D_n be database domains. Figure 2 presents the complete definition of the user graph data model.

$\nu : N \rightarrow D_i \times \ldots \times D_j$

$\varepsilon : E \rightarrow D_k \times \ldots \times D_m$

A graph G is defined by: G (N, E, Ψ, ν, ε)

Figure 2 - Complete definition of a graph

II.2 Operators

Graph manipulations, accessible from the user interface, are divided into three classes: evaluation of paths, inclusion and intersection.
The evaluation of path is performed with a Path operator. This evaluation corresponds to the detection of acyclic paths between an origin and a destination. This path can be a direct link (i.e., a successor in a graph) or a more complex path (i.e., a transitive closure of a graph).

Three operators are defined to handle the inclusion. The first operator is an extraction of nodes in a path. The second operator is an inclusion between two sets of nodes. The third operator is an inclusion between two sets of edges.
Two operators are defined to handle the intersection. The first operator is an intersection of two sets of nodes. The second operator is an intersection of two sets of edges.

This paper deals with the Path operator. It presents the visual definition of a query involving this operator. A formal signature (taking into account constraints) of this operator is presented in (Mainguenaud, 1993a). Inclusion and intersection are handled as defined in (Calcinelli, 1991) (Calcinelli, 1994).

II.3 Toy database

To simplify the presentation, let us use the relational model extended with Abstract Data Types (Stemple, 1986) as the data model of the user interface (i.e., to model the labelling functions ν and ε). The relational model is now widely accepted (Ullman,1988), but an object-oriented data model or a semantic data model could have been used. An abstract data type,Spatial_representation_type, models spatial representations (i.e., the domain of the Spatial_representation attribute). Figure 3 presents the conceptual level of a toy database borrowed from a tourism application.

User's schema

Town	(Name, Population, Economical_activity, Hotel_cost, Spatial_representation)
Transport	(Name, Company, Transport_cost, Departure_hour, Arrival_hour, Spatial_representation)

Database schema

Town	(Name, Population, Economical_activity, Hotel_cost, Spatial_representation)
Transport	(Name, Origin, Destination, Company, Transport_cost, Departure_hour, Arrival_hour, Spatial_representation)

Figure 3 - The conceptual database

Let us define a graph G (N, E, Ψ, ν, ε) modelled with the Town and Transport relations (i.e., the nodes and the edges). The user's schema is composed of two relations (i.e., the labelling functions ν and ε). Visual queries are defined on this schema. The database schema is composed of two relations. The Town relation models the node labelling function, ν. The Transport relation represents an element of the Cartesian Product, N x N, and the edge labelling function, ε. The formal modelling of a visual query is defined on this schema. To illustrate the query language, let us define seven queries on this database (see (Boursier, 1992) for a logical benchmark to evaluate the expressive power of a query language for GIS).
Let Q_1 be the following query: "What are the direct flights with the AF company from Paris to Lausanne such as the departure time is later than 10:00 am and earlier than 3:00 pm and such as the price is less than 500 FF?". Query Q_1 requires evaluating the direct link operator. This query is a relational selection on the Transport relation.
Let Q_2 be the following query: "What are the routes from Paris to Lausanne with the AF company such as the total transport cost is less than 800 FF and the total time of inter-connection is less than 2 hours?". Query Q_2 requires the computation of a path operator in graph G with constraints on edges. This query is a transitive closure on relation Transport with two aggregate functions: the sum of the Transport_cost attribute and the sum of the differences between Departure_hour of edge i+1 and Arrival_hour of edge i.
Let Q_3 be the following query: "What are the routes from Paris to Lausanne only visiting major towns (i.e., for each town, the population is larger than 100 000 inhabitants or the economical activity is tourism)?". Query Q_3 requires a path operator with a constraint on nodes. This query is a selection on the Town relation, a join with the Transport relation and a transitive closure.

Let Q4 be the following query: "What are the routes from Paris to Lausanne such as the total cost (hotel+transport) is less than 1000 FF?". Query Q4 requires a path operator with constraints on nodes and edges. This query is a transitive closure on relation Transport with an aggregate function using a join on relation Town.
Let Q5 be the following query: "What are the routes from Paris to Lausanne such as the total time of transport is less than 4 hours". Query Q5 requires a path operator (i.e., a transitive closure on relation Transport) with a constraint involving the first and the last edges (i.e., the difference between the departure hour from Paris and the arrival hour in Lausanne is less than 4 hours).
Let Q6 be the following query: "What are the routes from Paris to Lausanne (1) leaving Paris with the AF company; (2) and then using the train up to Lausanne or using the bus up to Lausanne?". Query Q6 requires a path operator (i.e., a transitive closure on relation Transport) with a regular expression on edges (Cruz,, 1987).
Let Q7 be the following query: "What are the routes from Paris to Lausanne (1) leaving Paris with the AF company; (2) then using the train or the bus such as at each stop, the cost of a hotel is less than 200 SF and (3) finally arriving to Lausanne by bus". Query Q7 requires a path operator (i.e., a transitive closure on relation Transport) with a regular expression involving nodes and edges.

III VISUAL QUERY LANGUAGE TO MANAGE GRAPHS

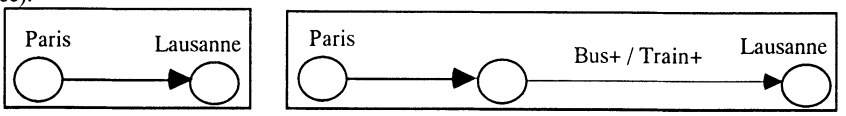
The first section presents some visual query languages to manage graphs. All of them are based at the user level on the same definition of a graph. The second and third sections present the Cigales query language from the user's point of view, to solve queries using either the direct link operator (query Q1) or the path operator (queries Q2 up to Q7).

III.1 Short overview

We present here a short overview of visual query languages to manage graphs. The two first one (G+ and QBD*) are designed to query a conventional database. The third one (Cigales) is designed to query a spatial database (i.e., with thematic-oriented operators).

III.1.1 G+

G+ (Cruz, 1987) is a language derived from a language called **G**. In **G**, a graphical query Q on a graph G is a set of labelled directed multi-graphs in which the node labels of Q may be either variables or constants and the edge labels are regular expression defined over n-tuples of variables and constants.
An edge which is labelled by a regular expression containing the positive closure operator (+) is drawn as a dashed edge in Q. A query is extended with a summary graph and summary operators. A summary graph tells the system how to restructure the answers before presenting them to the user. Summary operators allow to compute aggregate functions over paths and sets of paths.
The semantics of the graph query language is based on the existence of mappings known as homeomorphisms from query graphs to database graphs. Homeomorphisms formalize the idea of a subgraph of the database graph being "similar" to the query graph. A query graph represents a pattern; answering the query will require searching the database graph to find all subgraphs that are "similar" to the pattern.
Query Q1 is represented by a graph composed of two nodes labelled Paris and Lausanne. The edge is a tuple defined with constraints on the attributes (i.e., departure_hour, arrival_hour and price).



Nevertheless, the graphical visualisation of query Q6, for example, is not very intuitive (even if the basic concept is a regular expression). A user defines a regular expression from the keyboard. A Query-By-Example philosophy should be preferable.

*III.1.2 QBD**

The goal of QBD* (Query By Diagram*) (Angelaccio, 1990) is to achieve user friendliness for a large amount of user types by means of a uniform graphical interface and a visual language. This system is based on four basic ideas: (1) to represent the intentional part of the database by means of a conceptual model (Entity-Relationship augmented by the introduction of generalization abstractions); (2) to employ a fully graphical environment as a user friendly interface with the system; (3) to study the formal properties of the graphical language by defining an isomorphic textual language; and (4) to extend the graphical constructs of the language in order to easily express a significant class of recursive queries (e.g., transitive closure).
QBD* is mainly a navigational language on ER diagrams. The operations of the relational algebra may be expressed directly by picking up symbols (entities or relationships) in the diagram and posing conditions on their attributes by means of a simple windowing mechanism. Recursive queries are also expressed by the same mechanism, the difference being the preselection of a particular icon that signals the beginning of a recursive session. In this way, the textual interaction is completely eliminated and the complexity of some query is transparent to the user.
The general structure of the query is based on the location of a distinguished concept, call the main concept (an entity or a relationship), that can be seen as the entry point of one or more subqueries. These subqueries express possible navigations from the main concept to other concepts in the schema. The subqueries can be combined by means of the usual union and intersection operators. Once the main concept has been selected, two different types of primitives are available for navigation in the schema. The first one allows the user to follow paths of concepts. The second one is used for comparing two concepts which are not directly connected to each other. Two such primitives can be arbitrarily nested.
Nevertheless, it is not clear how (1) queries involving conditions on the first and last edge are handled (query Q_5); (2) aggregate function involving nodes and/or edges are handled (Query Q_3, Q_4).

III.1.3 Cigales approach

Cigales is a visual and declarative query language for GIS based on a Query-By-Example approach. This language is visual because a query is defined by a drawing. This language is declarative because a query only defines the properties to be verified by the results. A detailed definition of the query language is presented in (Calcinelli, 1994). Cigales uses metaphores to avoid visual overload. Thus a visual query is a symbolization of a textual query.
Cigales is designed to be an upper layer of a spatial Data Base Management System (DBMS). Two metaphores, a zone and a line, represent database objects. Their semantics are defined by a Data Model Function (i.e., the definition of selection criteria applied to the labelling functions ν and ε). Five spatial operators are available at the user interface level: inclusion, intersection, adjacency, straigth line and path. Several other operators are available such as union, difference but the user is not awared of their existence. The drawing process automatically generates these operators. A query is graphically built from a combination of metaphores and operators. This query is, then, translated into a formal expression based on a functional language (Mainguenaud, 1993a), to be compiled into DBMS understandable orders. Figure 4 presents the graphical editor used to define a query. Cigales offers a unique path operator at the user-interface level. This operator stands either for the direct link operator and for the path operator. The relevant choice is performed by the user before defining the Data Model Function (Figure 5).

III.2 Direct link queries

A direct link operator requires specifying alphanumeric properties on edges of a graph (i.e., for the Transport relation in our example). The expressive power is equivalent to the From, Where, Group By and Having clauses of SQL.

A selection criterion is defined by a triplet (attribute, operator, value). An implicit conjunction is defined between triplets. Moreover, each attribute may also have a value defined as a conjunction (e.g., the departure hour is later than 10:00 am and earlier than 3:00 pm - query Q_1). The

disjunction is graphically defined with the notion of page (i.e., Page 1, Page 2, ..., Page i). Each page has the same structure and allows one to define alphanumeric properties. A horizontal scrollbar allows to move from one page to another. This disjunction may be defined on one attribute (the departure hour is earlier than 10:00 am or later than 3:00 pm). It may also involve several attributes (i.e., the population is greater than 100 000 inhabitants or the activity is tourism - query Q_3). Details of other operators are presented in (Brossier-Wansek, 1994). Figure 6 presents the visualization of the Data Model Function associated with the direct link operator. This formalism is more intuitive for a non-specialist end user than SQL-like syntaxes.

III.3 Path operator

Queries Q_2, Q_3, Q_4, Q_5, Q_6 and Q_7 require computing a path operator. These queries illustrate two kinds of constraints. Queries Q_2, Q_3, Q_4 and Q_5 define constraints involving a complete path. Query Q_6 and Q_7 define constraints involving an edge-by-edge definition of a path. Therefore, the Data Model Function associated with the path operator is composed of two parts: a "Global conditions" part and a "Transportation" part.

III.3.1 Constraints on a complete path ("Global conditions")

Three kinds of constraints can be defined on a complete path. A constraint may be defined on edges (e.g., query Q_2), on nodes (e.g., query Q_3) or on both edges and nodes (e.g., query Q_4). Thus, the window defining the global conditions is divided into three parts.

Two kinds of constraints can be defined on edges. A constraint may involve a unary function (i.e., involving a unique parameter) or may involve a n-ary function (i.e., involving two or several parameters). The constraint: "the total transport cost of a travel is less than 800 FF of query Q_2" is an example of a unary function (i.e., an aggregate function on the attribute Transport_cost). The constraint: "the total time of interconnection is less than two hours of query Q_2" is an example of a n-ary function (i.e., an aggregate function involving a binary relationship between the attributes Arrival_hour and Departure_hour).
The constraint definition process involving a unary function is similar to the direct link definition process. The only difference is the addition of an aggregate function. This function is defined for each attribute depending on its type (i.e., alphanumeric, integer). Such a constraint is valid for each edge of a path (i.e., Each qualification - query Q_3) or for the entire path (i.e., total cost of a travel - query Q_2). Figure 7 presents the Data Model Function for a unary function.
The constraint definition process involving a n-ary function relies on a visual sub-language. To simplify the presentation, let us consider a binary relationship. Cigales defines a generic schema of a path. This path presents four specific edges labelled 1, i, i+1, n. The user selects attributes corresponding to these edges. The two icons modelling the retained attributes appear in the form below the schema. The user defines the aggregate function (if it is needed), the mathematical operator, the comparison operator and a value (i.e., the sum of differences between the departure hour of edge i+1 and the arrival hour of edge i is less than two hours - query Q_2 - or the difference between Arrival_hour of edge n and Departure_hour of edge 1 - Query Q_5). Figure 8 presents the Data Model Function for a n-ary function (Query Q_2).
Two kinds of constraints can be defined on nodes: a constraint may involve a unary function or a n-ary function.
The constraint definition process involving a unary function is similar to the constraint definition process involving a unary function for an edge.
The constraint definition process involving a n-ary function is similar to the constraint definition process involving a n-ary function for an edge. The only difference relies on labels associated with the generic schema of a path. They are defined on nodes instead of on edges. Figure 9 presents the Data Model Function associated with query Q_3 "What are the routes from Paris to Lausanne only visiting major towns (i.e., for each town, the population is larger than 100,000 inhabitants or the economical activity is tourism)?".
The constraints involving both edges and nodes are similar to a n-ary function (at least one attribute for an edge and one for a node). The generic schema of a path is made of four

components labelled edge i, node i, edge i+1, node i+1. The philosophy of the constraint definition process is similar to the previous definitions of n-ary functions. Figure 10 presents the Data Model Function associated with query Q_4 "What are the routes from Paris to Lausanne such as the cost (hotel+transport) is less than 1000 FF?".

One can remark that results of queries involving aggregate functions can not be defined as the union of paths. As soon as a node (except the origin of the path and the destination of a path) belongs to two paths, the definition of the results as the union of paths leads to ambiguities (Mainguenaud, 1993b) since results must be graphically presented.

III.3.2 Constraints edge-by-edge ("Transportation")

This second kind of constraints requires specifying a regular expression associated with a path. This edge-by-edge definition provides a very strong expressive power. As shown in (Mendelzon, 1989), some classes of queries lead to a NP-complete problem. We restrict the expressive power to a sub-set of computable queries (Garey, 1979). This part defines the retained expressive power and presents the graphical definition of these constraints.

Required expressive power

To simplify the presentation, let us consider a unique domain called D_0. Let D_0 be $\{d_0, ..., d_n\}$. Let ν be a one-to-one node labelling function that associates with each node a distinct value drawn from domain D_0. Let ε be an edge labelling function which associates with each edge a distinct value drawn from domain D_0.
An edge-by-edge constraint is defined with a regular expression. This expression is built from a triplet (N_i, E_k, N_j) label. Let N_i be the label associated with a node n_i (a disjunction of elements of D_0). Let N_j be the label associated with a node n_j (a disjunction of elements of D_0). Let E_k be the label associated with an edge e_k defined between n_i and n_j (a disjunction of elements of D_0). Let - (the hyphen) be any value of D_0. The following rules define a query label:

(1)	(N_i, E_k, N_j) is a query label	(2)	(N_i, E_k, -) is a query label
(3)	(-, E_k, N_j) is a query label	(4)	(-, E_k, -) is a query label
(5)	(N_i, - , N_j) is a query label	(6)	(N_i, -, -) is a query label
(7)	(-,-,N_j) is a query label	(8)	(-,-,-) is a query label
(9)	If A is a query label then [A]* is a query label (the repetition)		
(10)	If A and B are query labels then A∧B is a query label (the sequence)		
(11)	If A and B are query labels then (A∨B) is a query label (the alternative)		

An edge-by-edge constraint is built from a recursive application of properties (1) to (11). For the time, we limit the expressive power of a regular expression modelling a path. At least one of the two properties (a) or (b) must be verified:

(a) The first node label is not a hyphen (i.e., a selection criterion is defined on a key)
(b) The last node label is not a hyphen (i.e., a selection criterion is defined on a key).

Let us define a node label (N_i) or an edge label (E_k) as a disjunction of triplets (attribute operator value). Figure 11 (respectively Figure 12) presents the edge-by-edge constraint of query Q_6 (respectively query Q_7).

(Name = Paris, Company = AF, Hotel_cost < 200) ∧
[(Hotel_cost < 200, Company = Train ∨ Company = Bus, Hotel_cost < 200)]* ∧
(Hotel_cost < 200, Company = Bus, Name = Lausanne)

Figure 12 - Edge-by-edge constraint of query Q_7: "What are the routes from Paris to Lausanne (1) leaving Paris with the AF company; (2) then using the train or the bus such as at each stop, the cost of a hotel is less than 200 SF and finally arriving to Lausanne by bus"

(Name = Paris, Company = AF, -) ∧
(((-,Company=Train,-) ∧ [(-,Company=Train,-)]* ∧ (-,Company=Train, Name=Lausanne) ∨
(-,Company=Train, Name=Lausanne)) ∨
((-,Company = Bus,-) ∧ [(-,Company = Bus,-)]* ∧ (-,Company = Bus, Name = Lausanne) ∨
(-,Company = Bus, Name = Lausanne)))

Figure 11 - Edge-by-edge constraint of query Q6: "What are the routes from Paris to Lausanne (1) leaving Paris with the AF company; (2) and then using the train up to Lausanne or using the bus up to Lausanne?"

Figure 13 presents the Finite State Automaton (FSA) associated with query Q6. The conflict on Train (or Bus) is resolved since the choice of an edge determines an end node. Figure 14 presents the FSA associated with query Q7.

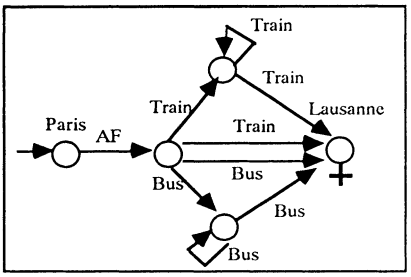


Figure 13 - FSA for query Q6

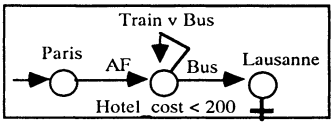


Figure 14 - FSA for query Q7

Graphical representation

A visual query language must provide a visual representation of a regular expression. Thus, the interface must provide a graph drawing mechanism. The objectives are (1) to draw an edge-by-edge path; (2) to specify alphanumeric properties on nodes and edges according to the user's selection criteria.

The graphical definition of an edge-by-edge constraint is provided by a graph editor. This editor adopts the same look as the main query editor. Two areas are defined. A working area is located at the bottom. This area is used to build step-by-step sub-queries. A query area, located on the top, is used to merge sub-queries and to build a final query. Two metaphores (a zone and an arrow) obey to the same Data Model Function as for Cigales. This editor takes full advantage of the context in which a regular expression is defined: a path defined between an origin and a destination. Therefore, no operator is provided since the path definition process is on. Figure 15 presents the graph editor.

The graphical edge-by-edge constraint is built step by step. A step defined in the working area corresponds to a sequence of query labels and must be validated (i.e., with a click on the validation button). The validation mechanism allows (1) the definition of the repetition factor (rule 9) and (2) the display of the graph drawing into the query area. The conjunction (rule 10) is implicit by a sequence of edges in the query area. The disjunction, while defining the $N_{i/j}$ (respectively E_k) part of a label is provided by the Data Model Function with the mechanism of pages (i.e., company = Train v company = Bus - query Q7). The disjunction in a regular expression (rule 11) is guided by the following assumption: "An end user thinks in term of disjunctive form". This disjunction is provided by a mechanism of pages for a complete path (i.e., each page represents a complete path - query Q6).

By this way, the graphical build up of a path provides a Query-By-Example definition process. Therefore a novice user easily handles a complex selection criteria (i.e., with multiple and/or conditions - query Q6). Moreover, the similitude between the Cigales editor and the graph editor

about the query definition process facilitates the definition of a complex regular expression for a path. Figure 16 (respectively Figure 17) presents the graph editor for query Q_6 (respectively Q_7).

IV. CONCLUSION

The number of applications using a Geographical Information System (GIS) is considerable. Therefore it is of prime importance to offer a powerful and friendly interface for end-users. By end-users, we mean people who may be experts in some domains other than programming (or querying) language and who want to use a GIS to meet their particular needs. Our focus is not so much on the spatial analysis itself, but on the computational environment in which it is embedded (i.e., query language expressive power, management of visual ambiguities). Visual programming languages are very promising since they are more natural than textual query languages (i.e., extended SQL) and they offer a higher level of abstraction. The interaction with a DBMS is more complex since this level of abstraction is higher. The implementation is still going on since the design of a user interface is an iterative process. The prototype is developed using OSF/Motif (for the user interface), Yacc (for the analysis of the internal functional expression modelling a query), C programming language (for the query management) and an object-oriented DBMS, O_2 (Bancilhon, 1991), (to manage network-oriented queries) on Sun workstations.

This paper presents a brief overview of the user interface graph data model and the query definition process to manage network-oriented queries. Future work is to handle query results since several visual ambiguities may appear (Mainguenaud, 1993b) as soon as an aggregate function is involved in the query.

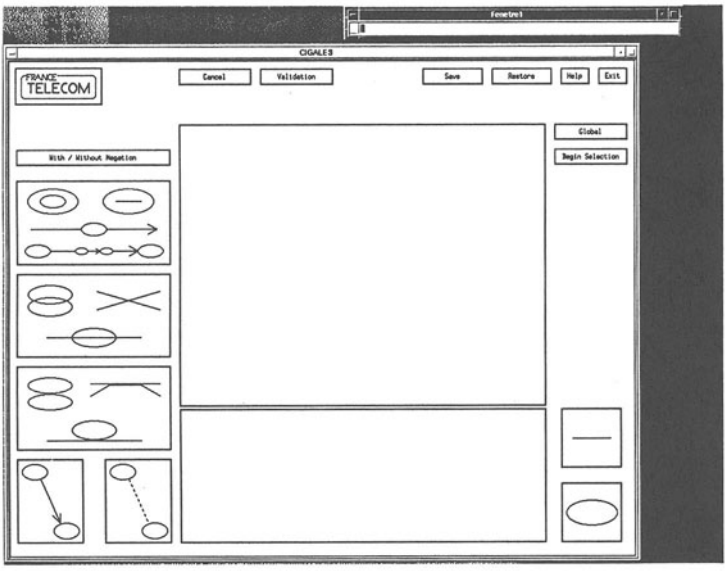


Figure 4 - The graphical editor

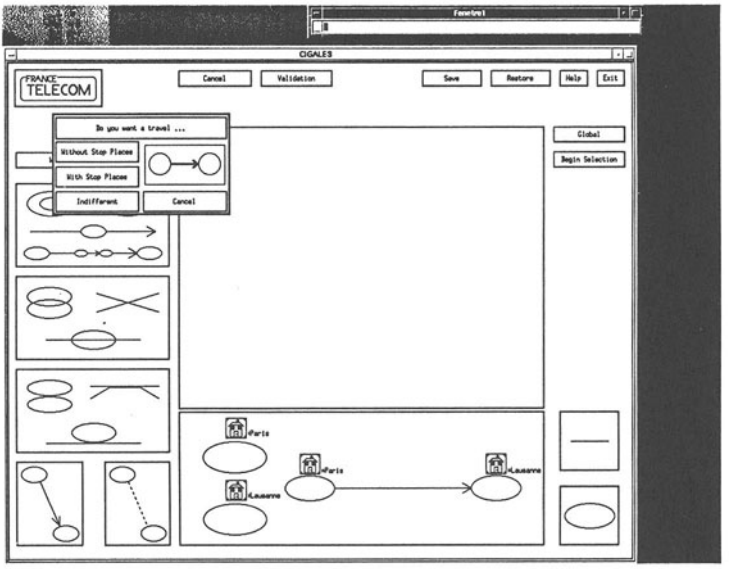


Figure 5 - Direct link or Path operator?

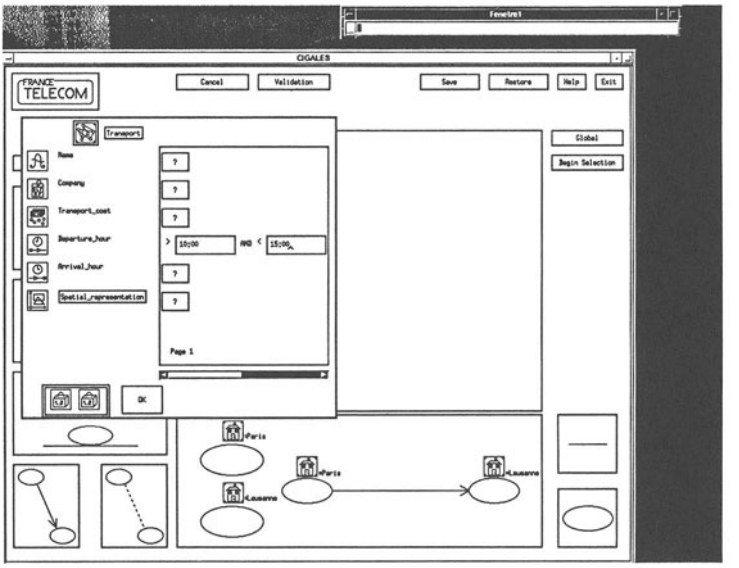


Figure 6a - The Data Model Function - conjunction
(the departure hour is later than 10:00 am and earlier than 3:00 pm - query Q_1)

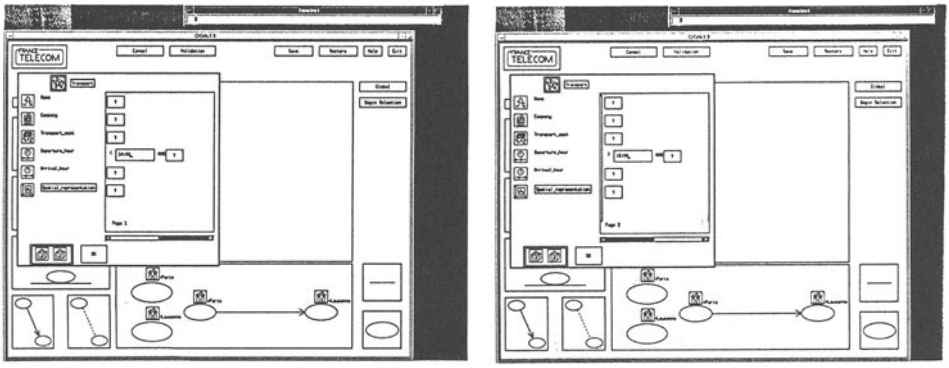


Figure 6b - The Data Model Function - disjunction
(the departure hour is earlier than 10:00 am or later than 3:00 pm)

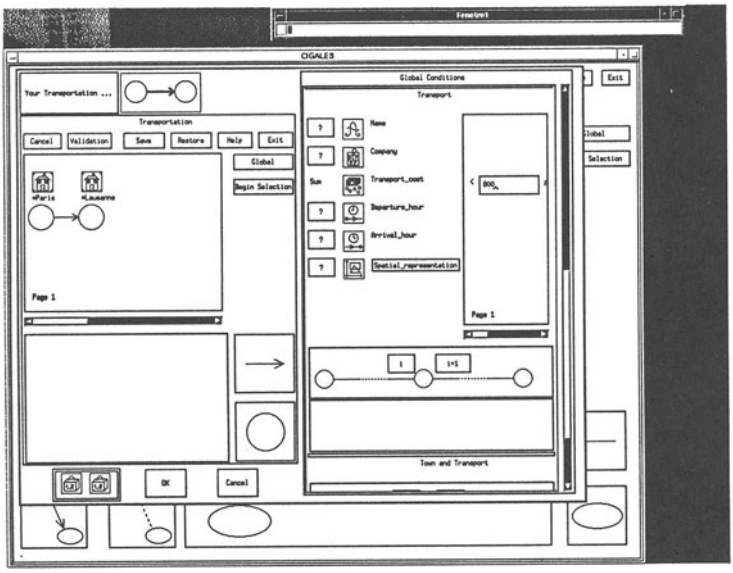


Figure 7 - The Data Model Function - a unary function
(the total cost of a travel is less than 800 units - query Q2)

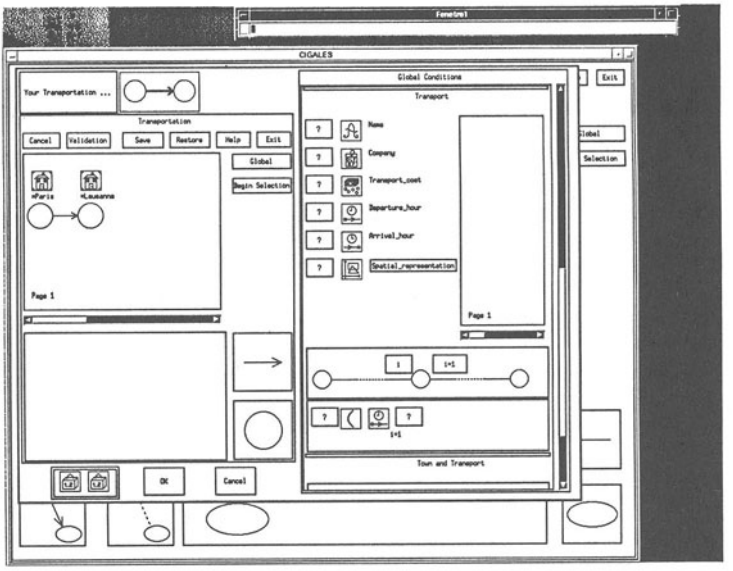


Figure 8a - Labelling of edge i+1 - departure hour of edge i+1
(the sum of differences between the departure hour of edge i+1 and the arrival hour of edge i is less than two hours - query Q2)

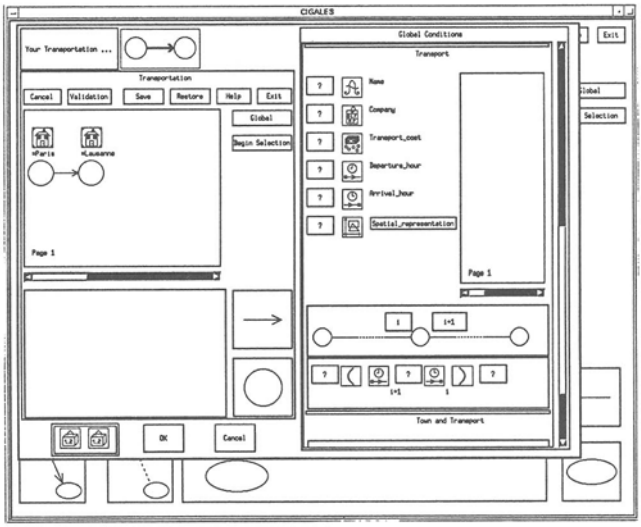


Figure 8b - Labelling of edge i - arrival hour of edge i
(the sum of differences between the departure hour of edge i+1 and the arrival hour of edge i is less than two hours - query Q2)

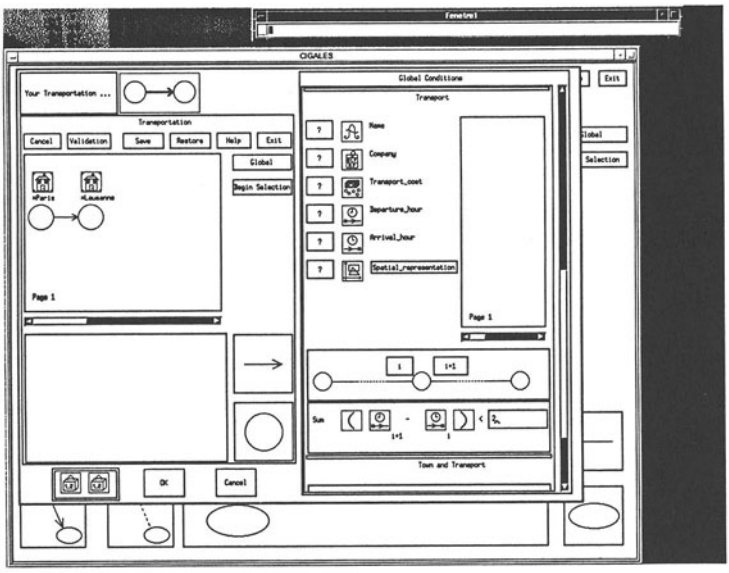


Figure 8c - Definition of an aggregate function - the sum of differences
(the sum of differences between the departure hour of edge i+1 and the arrival hour of edge i is less than two hours - query Q2)

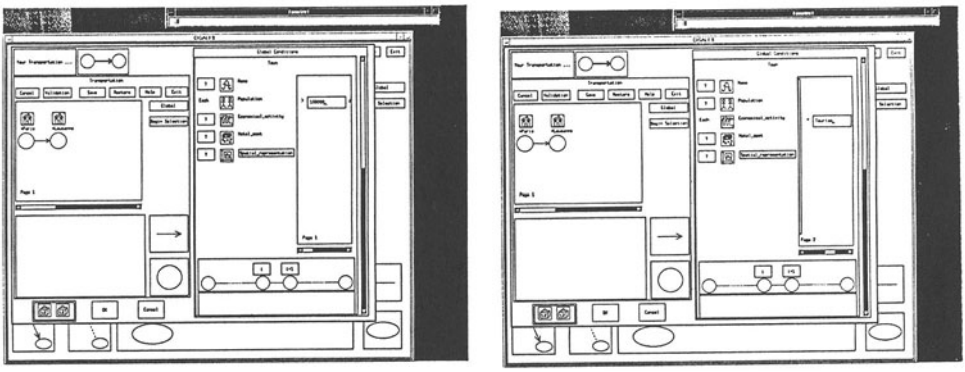


Figure 9 - Query Q3
"What are the routes from Paris to Lausanne only visiting major towns (i.e., for each town, the population is larger than 100 000 inhabitants or the economical activity is tourism)?"

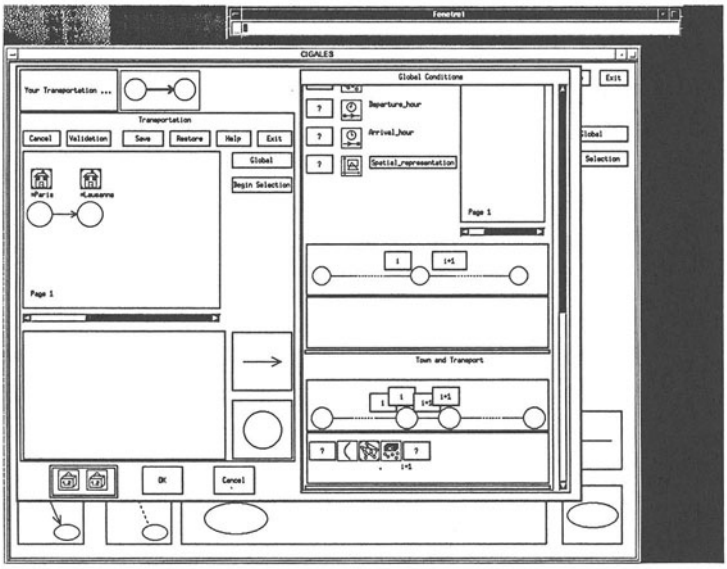


Figure 10a - Labelling of edge i+1 - Transport_cost
What are the routes from Paris to Lausanne such as the cost (hotel+transport) is less than 1000 F?

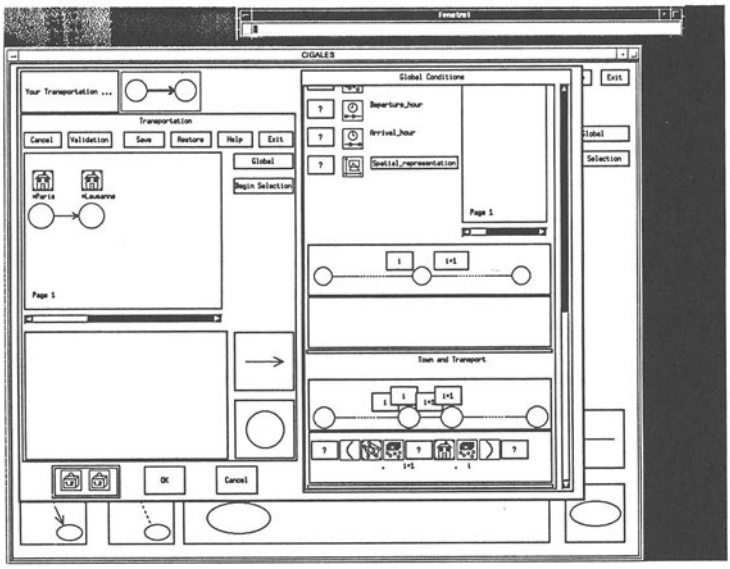


Figure 10b - Labelling of node i - Hotel_cost

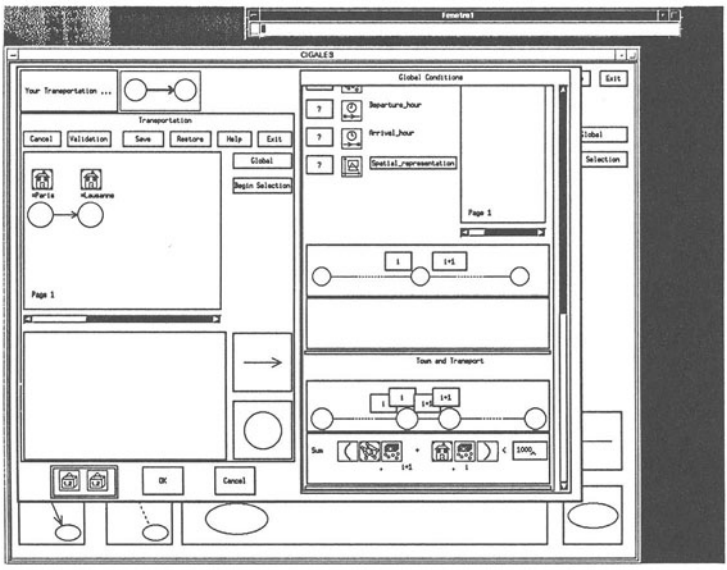


Figure 10c - Definition of an aggregate function - Sum

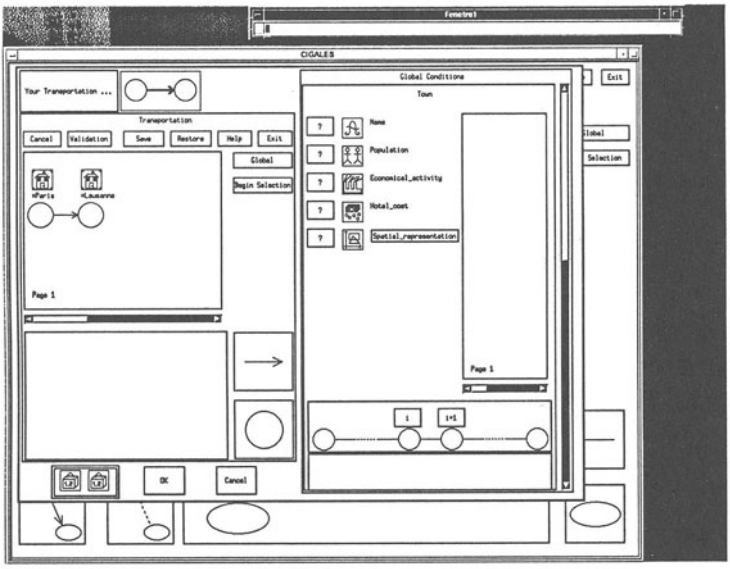


Figure 15 - The graph editor

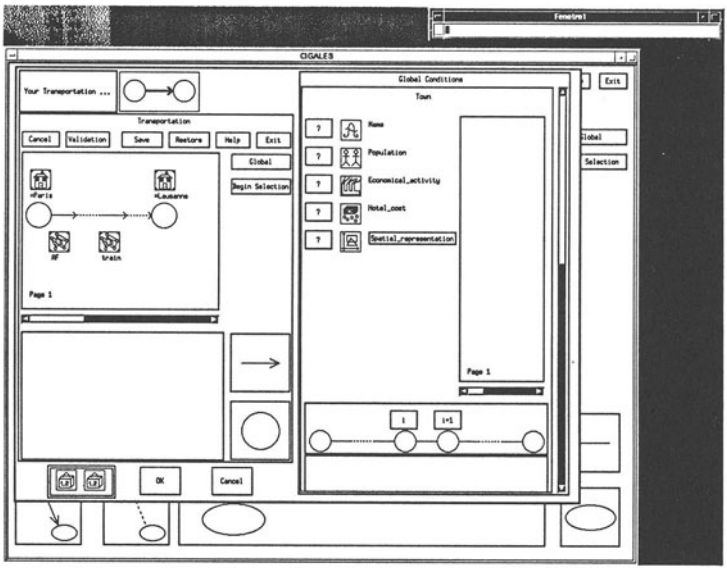


Figure 16a - Query Q6 - Page 1
"What are the routes from Paris to Lausanne (1) leaving Paris with the AF company; then using the train up to Lausanne"

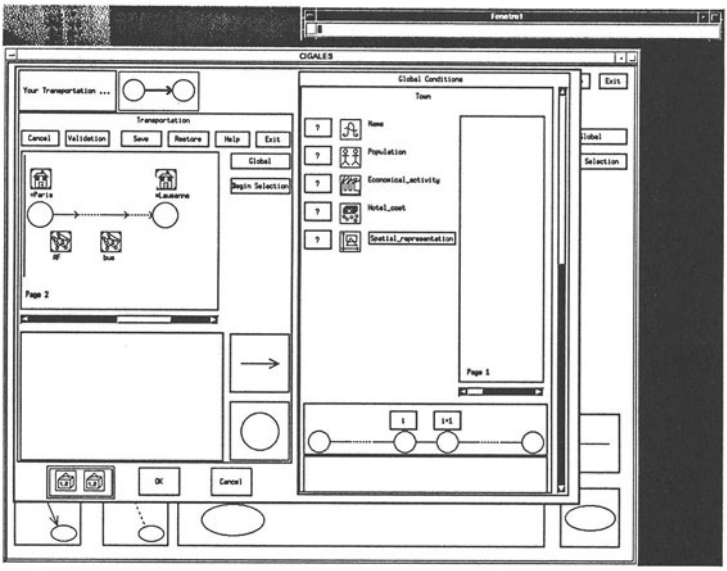


Figure 16b - Query Q6 - Page 2
"What are the routes from Paris to Lausanne (1) leaving Paris with the AF company; then using the bus up to Lausanne?"

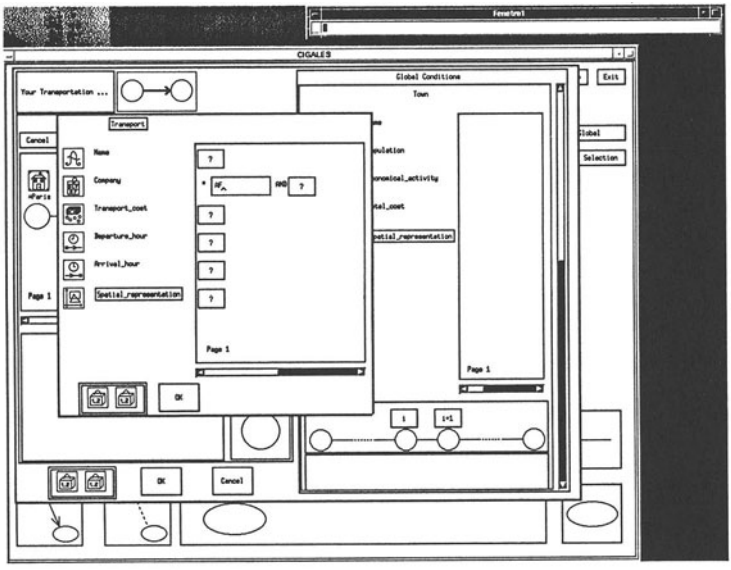


Figure 17a - Starting representation (a path from Paris to Lausanne - (Calcinelli, 1994)) What are the routes from Paris to Lausanne (1) leaving Paris with the AF company; (2) then using the train or the bus such as at each stop, the cost of a hotel is less than 200 SF and (3) finally arriving to Lausanne by bus

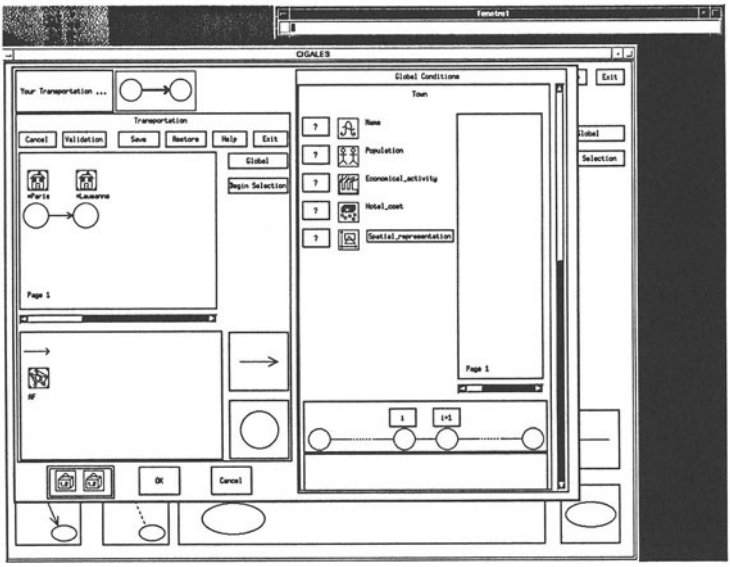


Figure 17b - The choice of an edge (labelled Company = AF)

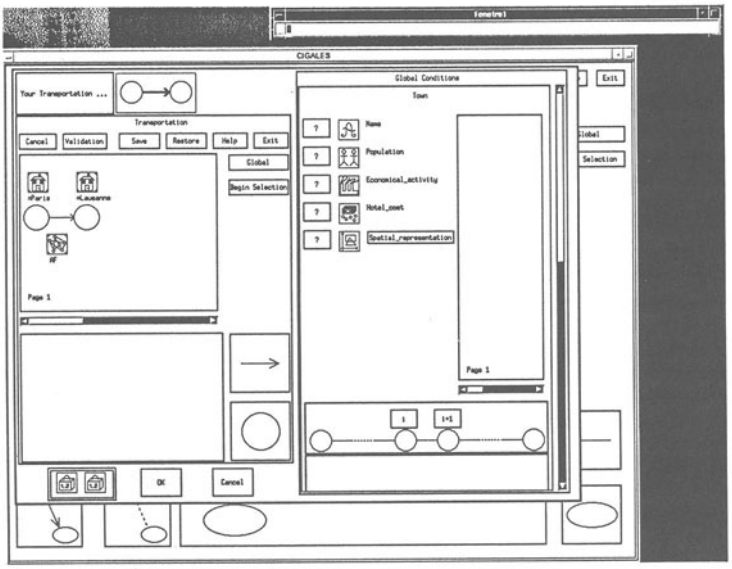


Figure 17c - After the validation (first part of the path: leaving Paris with AF)

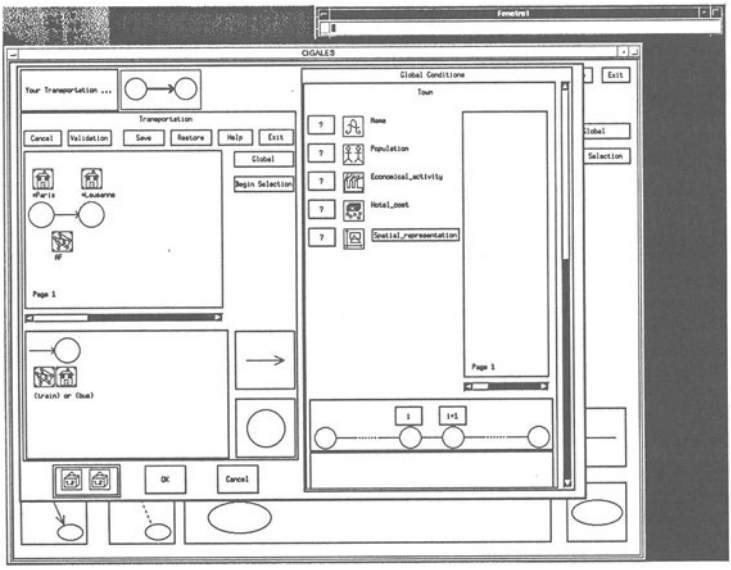


Figure 17d - The choice of an edge (labelled Company = Train/Company = Bus) and a node (labelled Hotel_cost < 200)

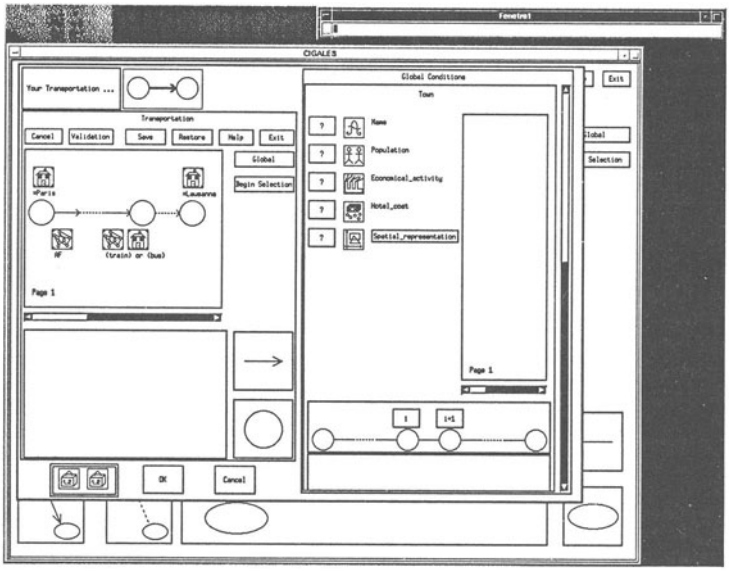


Figure 17e - After the validation with a repetition factor (dashed lines)

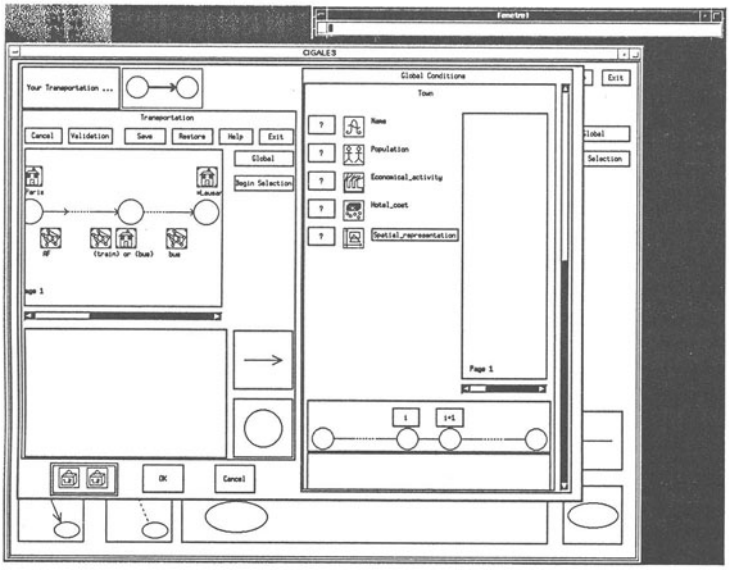


Figure 17f - After the choice of an edge (labelled Company = Bus) and the validation

REFERENCES

Angelaccio M., Catarci T. and Santucci G. (1990) QBD*: A Graphical Query Language with Recursion, *IEEE Transaction on Software Engineering*, **Vol 16 (10)**, 1150-1163.

Bancilhon F., Barbedette G., Benzaken V., Delobel C., Gamerman S., Lécluse C., Pfeffer P., Richard P. and Velez F. (1991) The Design and Implementation of O2, an Object Oriented Database System, *Proceedings of the 2nd Int. Workshop on Object-Oriented Data Base Systems*, O2 Book, Morgan Kaufmann.

Berge C. (1983) *Graphes*, Gauthier-Villars, Paris.

Boursier P. and Mainguenaud M. (1992) Spatial Query Languages: Extended SQL vs. Visual Languages vs. Hypermaps, *5th Symposium on Spatial Data Handling*, Charleston, USA.

Brossier-Wansek A. (1994) La Sémantique d'une Métaphore dans un Langage Visuel: Application à un Système d'Information Géographique, *ERGO/IA*, Biarritz, France.

Calcinelli D. and Mainguenaud M. (1991) The Management of the Ambiguities in a Graphical Query Language for Geographical Information Systems, *2nd Symposium on Large Spatial Databases*, Zurich, Switzerland, Lecture Notes in Computer Science n° 525 (LNCS).

Calcinelli D. and Mainguenaud M. (1994) CIGALES: A Visual Query Language for Geographical Information System: the User Interface, *Int. Journal of Visual Languages and Computing*, **5**, 113-132.

Cruz I.F., Mendelzon A.O. and Wood P.T. (1987) A Graphical Query Language Supporting Recursion, *Proc. SIGMOD Conference*, San-Francisco, USA.
and G+: Recursive Queries Without Recursion, *Expert Database Systems*, 645-666.

Garey M.R. and Johnson D.S. (1979) *Computers and Intractability A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York.

Kim H.Y. and al (1988) PICASSO: A Graphical Query Language, *Software-Practice and Experience*, Ed. J. Wiley and Sons Ltd, **Vol 18 (3)**, 169-203.

Kirby K.C. and Pazner M. (1990) Graphic Map Algebra, *4th Symposium on Spatial Data Handling*, Zürich, Switzerland.

Langou B. and Mainguenaud M. (1994) Graph Data Operations for Network Facilities in a Geographical Information System, *6th Symposium on Spatial Data Handling*, Edinburgh, UK.

Mainguenaud M. (1993a) From the User Interface to the Database Management System: Application to a Geographical Information System, *5th Int. Conference on Human Computer Interaction*, Orlando, USA.

Mainguenaud M. (1993b) The Results of Geographical Information System Queries, *IEEE/CS Visual Languages'93*, Bergen, Norway.

Mainguenaud M. and Simatic X.T. (1991) A Data Model to Deal with Multi-scaled Networks, *I.G.U.-U.G.I. Conference on Multi-scales Multi-uses*, Brno, Czechoslovakia and *Computer Environment and Urban System*, **Vol 16 (4)**, 1992, 281-288.

Mendelzon A.O. and Wood P.T. (1989) Finding Regular Paths in Graph Databases, *15th Int. Very Large Data Base Conference*, Amsterdam, The Netherlands.

Myers B.A. (Ed.) (1992) *Languages for Developing User Interfaces*, Jones and Bartlett Publishers, Boston.

Shu N.C. (1988) *Visual Programming*, Van Nostrand Reinhold Cie, New York.

Smith T.R., Menon S., Star J.L. and Ester J.E. (1987) Requirements and Principles for the Implementation and Construction of Large Scale GIS, *Int. Journal of Geographical Information System*, **Vol 1 (1)**, 13-31.

Stemple D., Sheard T. and Bunker R. (1986) Abstract Data Types in Databases: Specification, Manipulation and Access, *Int. Conference on Data Engeneering*, Los Angeles, USA.

Ullman J.D. (1988) *Principles of Database and Knowledge-base Systems*, Computer Science Press, Maryland.

BIOGRAPHY

Anne Brossier-Wansek received the Engineer degree in 1992 from the Institut National des Télécommunications (INT), a state engineer school.
She is presently working on her PhD in the CEGN and in the computer science department (database group) of the INT. Her research interets include data models and visual query languages for Geographical Information Systems and Hypertext and Hypermedia Systems.

Michel Mainguenaud received the Engineer degree from the Institut d'Informatique d'Entreprise (1985), a DEA in computer science (1985) and a PhD in computer science (1989).
He is presently an associate professor at the Institut National des Télécommunications in the computer science department (database group). His research interests are data models and query languages for multimedia databases (Geographical Information Systems -GIS- and Document Management Systems).
His main contributions are a data model and its operators to manipulate graph based data in a GIS; a visual language, Cigales, and a study on the query resolution mechanisms to address GIS databases; and a data model for multimedia hypertext like documents.