CONSTRAINT-BASED QUERIES IN A GEOGRAPHICAL DATABASE FOR NETWORK FACILITIES

MAINGUENAUD Michel

Institut National des Télécommunications
9 rue Charles Fourrier
F91011 Evry - France
Tel : + 33 1 60 76 47 14
Fax : + 33 1 60 77 20 06
Email : Michel.Mainguenaud@int-evry.fr
http : //www-inf.int-evry.fr/BasesDeDonnees/Cigales/cigales.html

Abstract:

The path evaluation operator is one of the most important operators in a Geographical Information System. Networks are modelled with graphs (i.e., a set of labelled vertices -named nodes- and a set of labelled edges). This operator is applied on huge graphs since the data set associated with a network is very important. A user's query does not involve numerous constraints (e.g., I would like to go from one place to another one). Many constraints are "implicit". This paper addresses the required database modelling to take into account these constraints.

These "implicit" constraints are based on the notions of time, space and applicative units. We introduce the characterization of database alphanumeric attributes in the following classes: Neutral, Time, Space and Applicative_unit. Some aggregate functions on nodes require one to evaluate this function on a sub-set of the nodes in a path (e.g., the cost of a hotel requires spending a night in a town). We introduce the notion of transparent nodes to model irrelevant nodes for an aggregate function. We define eight classes of generic functions to model a constraint: Global Time, Local Edge Time, Local Node Time, Global Space, Local Edge Space, Local Node Space, Time-Space mixed constraints and User's specific constraints.

Keyword:

Geographical Information System, Query language, Network facility

I. INTRODUCTION

In the current researches toward the design of more powerful tools for urban planning or tourism applications, different research groups are simultaneously concentrating their work on Geographical Information System (GIS). GIS needs are very well known [12] but several problems are still unanswered such as the design of user friendly query languages or the tailoring of a spatial database for specific users. In this paper we focus on the query definition process to manage network-oriented data.

The path evaluation operator is one of the most important operators in a GIS. Networks are modelled with graphs, a set of labelled vertices -named nodes- and labelled edges. This operator is applied on huge graphs since the data set associated with a network is very important (e.g., more than 36,000 towns in France). A user's query does not involve numerous constraints (e.g., "I would like to go from one place to another one"). Many constraints are "implicit". This paper addresses the required database modelling to take into account these constraints. We do not discuss the database structure adapted to model a graph or the best strategy to evaluate a transitive closure on a graph. Nevertheless this problem of performance is very important since the time function associated with the choice of a node or an edge may involve several database operations to evaluate the labelling conditions (e.g., several join operations in a relational database).

The database query (different from the user's query) must be as precise as possible to reduce the search space. An elementary path evaluation is designed as a simple path from an origin to a destination with a constraint defined as a cost function. Depending on the application, this cost function may be very difficult to express by a user since it involves many "implicit" constraints. The answer of a query is not a unique graph, a path, but a (reduced) set of graphs obtained with a multi-criteria analysis.

This paper studies the extension of a classical alphanumeric database model to increase the selectivity of a cost function. We introduce the characterization of alphanumeric attributes in the following classes: Neutral, Time, Space and Applicative_unit. Some aggregate functions on nodes require one to evaluate this function on a sub-set of the nodes in a path (e.g., the cost of a hotel requires spending a night in a town). We introduce the notion of transparent nodes to model irrelevant nodes for an aggregate function. We define eight classes of generic functions to define a constraint: Global Time, Local Edge Time, Local Node Time, Global Space, Local Edge Space, Local Node Space, Time-Space mixed constraints and User's specific constraints.

Part II presents the user's data model, user’s queries and cost functions; Part III presents basic components of a cost function; Part IV presents generic cost functions; Part V presents the conclusion.

II. USER'S DATA MODEL, USER'S QUERIES AND COST FUNCTIONS

II.1. User's data model

The user's data model is based on the concept of graph [2]. Let us consider a graph G as defined in [5] (Figure II.1).

G (N, E, ν, ε, Ψ)
N is a set of vertices (named nodes in the following)
E is a set of edges (a sub set of N x N)
ν is a labelling function for nodes
ε is a labelling function for edges
Ψ is an incident function E -> N x N

Figure II.1 - Definition of a graph

The relational database model is now widely known [15]. To simplify the presentation, let us use an extended relational model to define the labelling functions (nevertheless an object oriented data model or a richer data model could have been used). An Abstract Data Type [14] models the spatial representation [11] (i.e., Spatial_representation_type). To illustrate the concepts defined in this paper, let us define a toy database with towns (the set of nodes, N) and a transportation network (the set of edges, E). Let us use a flat graph to model the transportation network since our goal is not to illustrate the database structure and the path evaluation algorithm [7,9] but the query definition process. Figure II.2 presents the toy database schema. The relation Town models the ν labelling function. The relation Transport models the ε labelling function.

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

Figure II.2 - A toy database schema

II.2. User's queries

The definition of the completeness of GIS operators is not a realistic task timewise. An empirical approach must be used to define the required set of operators in such applications. User's queries on this toy database are modelled with a path operator and a set of constraints [3]. The path operator is the application of a transitive closure on the graph modelling a transportation network. We consider in this paper queries involving networks without any link to their environments (e.g., a path from Paris to Nice bordering a lake and crossing a town in its non-forest part). Eleven categories of queries can be defined to model constraints on a path *from the user's point of view*. A GIS network-oriented query is a combination of these categories. These categories can be grouped into two classes. The first class can be modelled with a Finite State Automata (FSA). The second class can be modelled with aggregate functions.

FSA-based queries

(1) A query may be defined with an empty set of constraints such as query Q_1: "I would like to go from Paris to Nice".
(2) A query may be defined with constraints on nodes (without any order) such as query Q_2: "I would like to go from Paris to Nice via the towns of Grenoble and Valence".
(3) A query may be defined with constraints on nodes, with an order, such as query Q_3: "I would like to go from Paris to Nice via the towns of Grenoble and then Valence".
(4) A query may be defined with a negation on nodes such as query Q_4: "I would like to go from Paris to Nice without visiting Marseille".
(5) A query may be defined with constraints on edges (without any order) such as query Q_5: "I would like to go from Paris to Nice via the TGV train".
(6) A query may be defined with constraints on edges with an order such as query Q_6: "I would like to go from Paris to Nice via the TGV train and then the Corail train".
(7) A query may be defined with a negation on edges such as query Q_7: "I would like to go from Paris to Nice without using the TGV train".

Aggregate function based queries

(8) A query may be defined with an aggregate constraint on nodes (e.g., the number of nodes) on the global path such as query Q_8: "I would like to go from Paris to Nice via a unique stop place".
(9) A query may be defined with an aggregate constraint on an alphanumeric attribute (or an ADT) of a node for the global path such as query Q_9: "I would like to go from Paris to Nice with a sum of Hotel_cost less than 500 FF".

(10) A query may be defined with an aggregate constraint on an alphanumeric attribute (or an ADT) of an edge for the global path such as query Q_{10}: "I would like to go from Paris to Nice with a sum of Transport_cost less than 1000 FF".
(11) A query may be defined with an aggregate constraint on alphanumeric attributes (or an ADT) of nodes and edges for the global path such as query Q_{11}: "I would like to go from Paris to Nice with a sum of Transport_cost and Hotel_cost less than 1500 FF".

Figure II.3 sums up the constraints. Query Q_8 and Q_9 may seem similar (i.e., an aggregate function on nodes). Let us admit for the time they are not similar. User's path-oriented queries do not have a powerful enough selectivity. We do not consider in this paper the user interface to define such queries (user interface Query Definition Process - QDP) [1,4,5] and restrict our study to computable queries [6,10]. The problem is therefore to provide a methodology to improve the selectivity of the cost function by adding selection criteria.

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
Constraint	∅	Node no order	Node order	Node Neg.	Edge no order	Edge order	Edge Neg.	Node Agg.	Node Agg.	Edge Agg.	NodeEdge Agg.

Neg. : Negation Agg. : Aggregate
Figure II.3 - The constraints

II.3. Cost functions

Three levels of vision can be defined to manage networks: Physical level (i.e., spatial coordinates), Logical level (i.e., graph modelling the transportation network) and applicative level (i.e., the structuring of a graph taking into account application dependent modellings, e.g., a night is spent in a hotel at each stop place). Figure II.4 presents the different levels of abstraction.

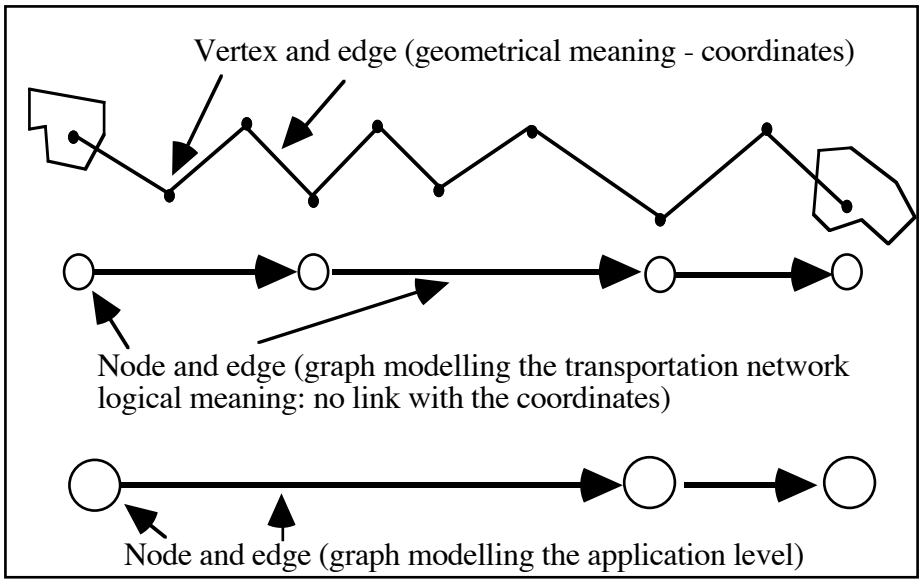

Figure II.4 - Different levels of abstraction

Three metrics can be defined: Time, Space and application dependent units (Applicative units). Two levels of resolution can be defined to solve a query: Global (i.e., constraints are defined for the whole path) and Local (i.e., constraints are defined for each components -node and edge- of a path). Three levels of spots can be defined to focus on: Node (Vertex), Edge, Node (Vertex) and Edge. The constraints cannot be obtained by a cross-product of aggregate functions and database attributes or an ad-hoc definition depending on each application. The cross-product provides 54 combinations (vision (3) x metrics (3) x resolution (2) x spot (3)). Some combinations are not relevant. Figure II.5 sums up relevant combinations.

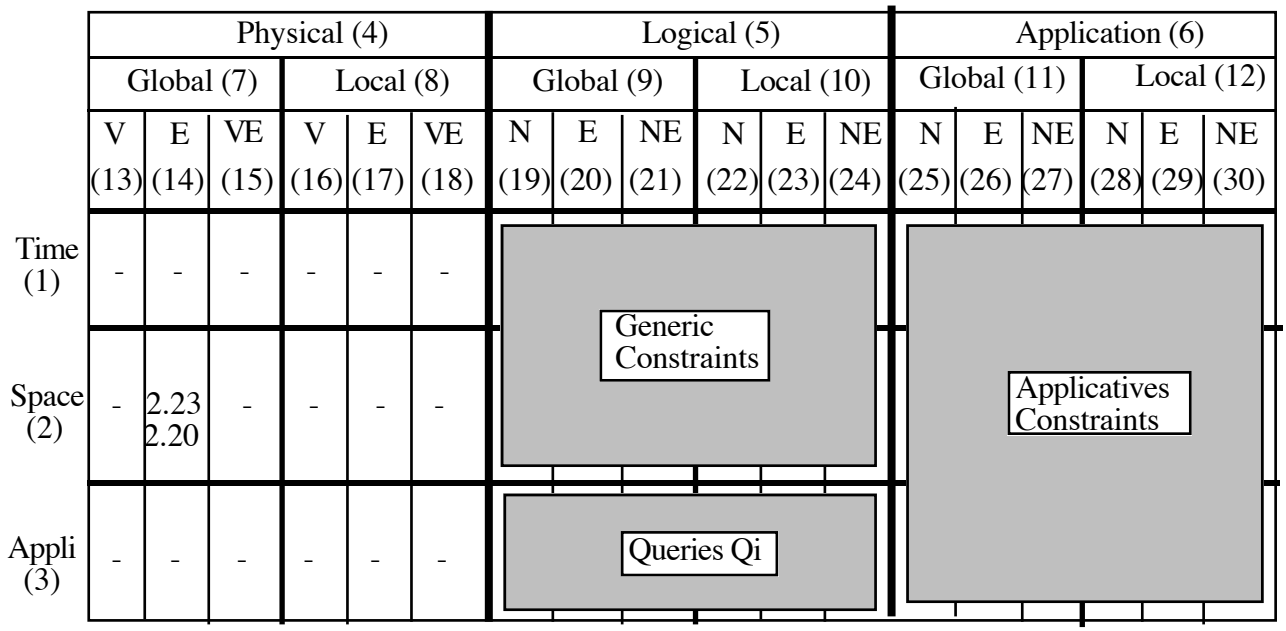

Figure II.5 - Relevant combinations

Figure II.5 presents a two-dimensional array. The lines represent the three metrics (denoted by classifications 1, 2 and 3). Each column models a level of vision (classifications 4, 5 and 6). A column is divided into two sub-columns modelling the levels of resolution (classification 7 and 8 - for the first column, 9 and 10 for the second column and 11 and 12 for the third column). Within each level of resolution, a sub-column is divided into three new sub-columns modelling in the spots (e.g., classification 13 to 15 for the sub-column with classification 7 of the column with classification 4). The notation X.Y denotes the intersection of a line with the classification X and a (sub-)column with the classification Y.

The notion of ADT allows one to hide the notion of physical representations with coordinates or other models (since these data depend on the type of acquisition, scale, quality of digitalization, etc.). The classification 1.4 is not relevant since the values of attributes classified Time are not available at the physical level (e.g., a segment of a polyline modelling a part of a railway line). No user interaction is therefore possible.
For the same reasons (i.e., no disaggregation at the physical level), the classification 3.4 is not relevant. No user interaction is therefore possible.
Space constraints can be coupled with an abstract data type by the means of functions. These functions are available for the global representation since access to a specific part of the data representation is not possible. Therefore, the classification 2.8 is not relevant. The physical modelling of a spatial data is based on the notion of vertices and edges (with the physical meaning). Vertices are not accessibles since they do not model data. The classifications 2.13 and 2.15 are not relevant. The whole set of physical edges represents a single logical edge (of graph G) or a path (in graph G). The classification 2.14 is therefore similar to constraints in the classification 2.23 or 2.20 by the introduction of ADT functions (e.g., Length or Begin and End of an edge of graph G).

III. BASIC COMPONENTS OF A COST FUNCTION

We consider the constraints defined in a user's query are divided into two parts: the main constraints and the implicit constraints. Main constraints are specified by a user. Implicit constraints are not initially defined by the user but are proposed by the QDP. They are considered as implicit from the user's point of view. Nevertheless, they must be specified to improve the selectivity of user's criteria (to reduce the search space). Implicit constraints, in a user's query on a path evaluation, are based on the definition of a function involving three components: time, space and applicative units. The characterization within these three components of database attributes must be provided. The designer of external database schematas (e.g., database view) is responsible for this task. This characterization is a complementary specification of the spatial semantics of database attributes [8]. This notion leads to three consequences: the characterization of database attributes, the creation of new attributes and the notion of transparent nodes.

III.1. Characterization of database attributes

Let us define four qualifications: Neutral, Time, Space and Applicative_unit. They are added to each database attribute depending on its semantics. A SQL-like Data Definition Language can be used to introduce these qualifications. Figure III.1 presents the extended database schema of our toy database.

Town	(	Name : Neutral, Population : Neutral, Economical_activity : Neutral, Hotel_cost : Applicative_unit, Spatial_representation : Space)
Transportation	(	Name : Neutral, Origin : Neutral, Destination : Neutral, Company : Neutral, Transport_cost : Applicative_unit, Departure_hour : Time, Arrival_hour : Time, Spatial_representation : Space)

Figure III.1 - Extended database schema

III.2. Creation of new attributes

An edge (of E) is defined as an element of N x N. Therefore, it exists a link between E and N. Some attributes defined in the labelling function of E allow one to infer some attributes in the labelling function of N. Non applicative data can be created if they are not already present in the data model.

An elementary path, p, between two nodes, n_{origin} and $n_{destination}$, is defined as a graph P (N_p, E_p, ν_p, ε_p, Ψ_p). The definition of P (N_p, E_p, ν_p, ε_p, Ψ_p) is similar to the definition of G (N, E, ν, ε, Ψ) (Figure II.1). Two components of a constraint can be defined on this graph: temporal functions and spatial functions.

Temporal functions:

Temporal functions are defined as calculated attributes [13] with the qualification: Time. Let us define $\mathbb{T}$ as a generic domain for time data. Let us define $-_{\mathbb{T}}$ as the closed difference operator between two elements defined on $\mathbb{T}$. Let us denote $\perp_{\mathbb{T}}$ the undefined value on $\mathbb{T}$ (whenever Time attributes are not available in the database schema). The application of $-_{\mathbb{T}}$ whenever an argument (at least) has the value $\perp_{\mathbb{T}}$, provides $\perp_{\mathbb{T}}$ as a result. Let us define two functions Departure_Hour and Arrival_Hour as:

Departure_Hour : $E_p \rightarrow \mathbb{T}$: Time

Arrival_Hour : $E_p \rightarrow \mathbb{T}$: Time

Four calculated attributes can be defined during the evaluation of a path: Time_in_a_Node, Time_in_an_edge, Arrival_hour_in_a_node and Departure_hour_in_a_node. One can remark that the semantics associated with these attributes is always the same but their values depend on the current state of the path evaluation process (therefore these attributes cannot be considerated as fixed attributes). Due to the closure of the $-\square_{\mathbb{T}}$ operator their qualifications are Time.

Arrival_hour_in_a_node : N -> $\mathbb{T}$: Time
Arrival_hour_in_a_node (n) =
Arrival_Hour (e_i) / e_i : $(n',n) \in E_p \wedge n' \in N_p \wedge n \in N_p$
$\perp_{\mathbb{T}}$ if $n = n_{origin}$

Departure_hour_in_a_node : N -> $\mathbb{T}$: Time
Departure_hour_in_a_node (n) =
Departure_Hour (e_i) / e_i : $(n,n') \in E_p \wedge n' \in N_p \wedge n \in N_p$
$\perp_{\mathbb{T}}$ if $n = n_{destination}$

Time_in_a_Node : N -> $\mathbb{T}$: Time
Time_in_a_Node (n) = Departure_hour_in_a_node (n) $-_{\mathbb{T}}$ Arrival_hour_in_a_node (n)

Time_in_an_edge : E -> $\mathbb{T}$: Time
Time_in_an_edge (e_i) = Arrival_Hour (e_i) $-_{\mathbb{T}}$ Departure_Hour (e_i)

Spatial functions:

Spatial functions are defined as calculated attributes or as functions on an Abstract Data Type modelling the spatial representation (i.e., Spatial_representation_type) with the qualification: Space. To simplify the presentation, let us define a unique function (Length) and $\perp_{\mathbb{R}}$ the undefined value when this function is unapplicable to this spatial representation.

Length : Spatial_représentation_type -> Real $\cup \perp_{\mathbb{R}}$: Space

The introduction of the length as a conventional fixed attribute is not desirable since it introduces a link between a logical representation of a graph and a physical representation (e.g., coordinates of a railway line). Furthermore the representation of a (physical) multi-graph by a (logical) graph is highly desirable. Several physical representations, (e.g., different railways lines), may be acceptable for the same logical edge, (e.g., link between two towns).

III.3. Transparent nodes

Two categories of nodes appear while evaluating an aggregate function involving applicative units on a path. The first category, named Visible_node, represents relevant nodes for the aggregate function. The second category, named Transparent_Node, represents irrelevant nodes for the aggregate function. These nodes have semantic information since they belong to the path but they do not contribute to the evaluation of the aggregate function.

In a path p, P (N_p, E_p, ν_p, ε_p, Ψ_p), N_p is divided into two sets, N_{pv} and N_{pt}. N_{pv} models visible nodes. N_{pt} models transparent nodes. This decomposition is provided by the definition of a function, Visible, such as :

$$\text{Visible} : N \rightarrow N$$

In a graph P (N_p, E_p, ν_p, ε_p, Ψ_p), N_{pv}, N_{pt} are defined as:

$$N_{pv} = \text{Visible}\ (N_p)$$
$$N_{pt} = N_p - N_{pv}$$

Let C (p) be a cost function of a path p. Let A be an attribute defined in the node labelling function on which the aggregate function is defined. Let Π be the relational operator of projection (for our toy database modelling). Let Agg be an aggregate function (e.g., sum, average).

$$C(p) = \underset{n \in N_{pv}}{\text{Agg}} \ (\Pi_A \nu_p(n))$$

The sum of Hotel_cost in query Q_9 is an example of an aggregate function involving visible nodes and transparent nodes (and here is the difference between query Q_8 and query Q_9). A visited town is semantically relevant for the path evaluation (it belongs to the path) but may be irrelevant for the aggregate function (if the traveller does not spend a night in this town). In our example, the function, Visible, is therefore defined with a Day function by:

Visible (N) = { $n \in N$ / Day (Departure_hour_in_a_node (n)) # Day (Arrival_hour_in_a_node (n)) }

This function is built by the QDP and depends on each constraint. It allows defining a new graph P'(N'_p, E'_p, ν'_p, ε'_p, Ψ'_p) such as:

$$\forall\ n \in N'_p,\ n \in N_{pv}$$

The graph P' is defined as a view on graph P (i.e., part of a re-structuring of graph P). This graph corresponds to the graph at the third level in Figure II.4. A recursive definition of constraints can be applied (i.e., same principles but on graph P' instead of graph G).

IV. GENERIC COST FUNCTIONS

Application-dependent cost functions are based on the user's profile. We do not consider here the way of obtaining this profile (logging, depending on the launched application, etc.). To simplify the presentation, let us consider a (unique) user's profile is available. This profile allows defining new (pre-defined) constraints that have not been explicitly expressed in the query. A (final) query is therefore obtained by a function, named Improve, applied on a triplet modelling the initial query, the user's profile and a dialogue to increase the selectivity of the initial query (Initial_Query, Profile, Dialogue).

The problem is therefore to define the relevant combinations of alphanumeric attributes (or functions on ADT), levels of vision, metrics, resolutions and spots to build a new constraint. Eigh functions are defined. Figure IV.1 presents these functions grouped into four classes.

Time:	(1) Global Time function
	(2) Local Time Edge function
	(3) Local Time Node function
Space:	(4) Global Space function
	(5) Local Space Edge function
	(6) Local Space Node function
Time/Space:	(7) Time-Space Mixed function
Applicative:	(8) Specific User's function

Figure IV.1 - Different classes of constraints

□IV.1. Time functions

a) Global

A Global Time function (classification 1.19, 1.20, 1.21) is one of the most common constraints in a path evaluation. An example of such new constraints is provided by queries built from the family of queries Q_{11}, Q_8 and Q_{10} "I would like to go from Paris to Nice". Query $Q_{11.1}$ is: "such as the total travel time is less than 15 h" (classification 1.21). Query $Q_{8.1}$ is: "such as the total time spent during connections is less than 2 h" (classification 1.19). Query $Q_{10.1}$ is: "such as the total travel time between visited towns is less than 10 h" (classification 1.20).

b) Local

Edge function: A Local Time Edge function (classification 1.23) involves the notion of visible and transparent nodes. This function can be considered as (1) the recursive application of a Global Time function on a sub-path, between two visible nodes, on graph P (i.e., applied on an edge of graph P'); or (2) a selection criterion on an edge of graph P. An example of such new constraints is provided by queries built from the family of query Q_5 "I would like to go from Paris to Nice". Query $Q_{5.1}$ is: "such as the time between two towns where a night is spent, is not less than 10h". Query $Q_{5.2}$ is: "such as the time between two stop places is not less than 2h 30 min" on graph P.

Node function: A Local Time Node function (classification 1.22) involves the notion of visible and transparent nodes (i.e., applied on graph P and P'). An example of such a new constraint is provided by queries built from the familly of query Q_2 "I would like to go from Paris to Nice". Query $Q_{2.1}$ is: "such as the time between the arrival hour and the departure hour in a stop place (resp. a town where is night is spent) is no more than 2h 30 min (resp. 10h)".

IV.2. Space functions

a) Global

A Global Space function (classification 2.19, 2.20, 2.21) is applied on the entire spatial representation of a path. Its structuring is similar for nodes, edges or nodes and edges specifications (i.e., an aggregate function, a mathematical expression involving database attributes or functions on ADT, a comparator and a value). One can remark that this function requires the computation with physical coordinates of a path. A transformation from a logical representation of a path with nodes and edges to a set of vertices and edges in the sense of geometrical data models [11] is applied.

An example of such new constraints is provided by queries built from the family of queries Q_{11}, Q_8 and Q_{10} "I would like to go from Paris to Nice". Query $Q_{11.2}$ is: "such as the total length of the path is less than 1500 Km" (classification 2.21). Query $Q_{8.2}$ is: "such as the total length in the visited towns is less than 100 Km" (classification 2.19). Query $Q_{10.2}$ is: "such as the total length outside visited towns is less than 1400 Km" (classification 2.20).

b) Local

Edge function: A Local Space Edge function (classification 2.23) is applied on each edge belonging to a path. This constraint is considered as a conventional constraint since it can be assimilated to a space function defined on the abstract data type associated with an edge and therefore as a conventional selection criterion. One can remark that this kind of constraint can be defined on a graph P or P' since the constraint is defined from the user's point of view. As for Local Time Edge function, a Local Space Edge function may be considered as the recursive application of a Global

Space function on a sub-path reduced to a unique edge (of graph P') or on a set of logical edges of graph P modelling a unique edge in graph P'.
An example of such new constraints is provided by queries built from the family of query Q_5 "I would like to go from Paris to Nice". Query $Q_{5.3}$ is: "such as the length between two stop places is less than 200 Km" on graph P. Query $Q_{5.4}$ is: "such as the length of a travel between two towns where a night is spent, is less than 800 Km" on graph P'.

Node function: A Local Space Node function (classification 2.22) can be defined within two levels: transparent nodes and visible nodes.
For transparent nodes, a Local Space Node function is the equivalent of a Local Space Edge function since it is defined between the spatial representation of two edges: the arriving "point" of the in-edge and the leaving "point" of the out-edge. An example of such a new constraint is provided by queries built from the family of query Q_2. Query $Q_{2.2}$ is: "such as the length for each connection between the arrival place of a train and the departure place of the next train is less than 100 m". The semantics of such a constraint is used for example during a connection between two railway lines requiring that the connection must be in the same railway station.
For visible nodes, a Local Space Node function may be considered as a new geographical query (i.e., a sub query of the user's query). An example of such a constraint may be a query based on the distances between (1) the arrival railway station and a hotel and (2) this hotel and the departure railway station.

IV.3. Time-Space mixed function

A constraint may be defined as a combination of the two previous classes of constraints. Time-Space mixed functions give the opportunity of defining a constraint for a single edge (on graph P or P') on both time and space. An example of such a new constraint is provided by:

"Every 300 Km by car, I stop for 1/2 hour"

This constraint cannot be handled in the database (i.e., with Time attributes). The notion of user's profile is well adapted to manage such a notion. This class of new constraint is to be taken into account while defining the Arrival_Hour function. The signature of the function remains the same but spatial functions can be applied while defining the body.

IV.4. Specific User's function

Providing a universal Data Base Management System (DBMS) for new applications such as CAD/CAM, GIS, or real time applications is not realistic. Even in a single field of application (e.g., GIS), a complete DBMS is not realistic since the notion of completeness is not (and will not be) defined. The boundary between applications and a DBMS is therefore very fuzzy.

The specific user's constraints group two classes of constraints that have not been previously classified.

The first class contains constraints involving attributes classified as Applicative_units. An example of such global constraints is provided by queries Q_9 (for classification 3.19), Q_{10} (for classification 3.20) and Q_{11} (for classification 3.21). An example of such local constraints is provided by queries built from the family of queries Q_2 (for classification 3.22), Q_5 (for classification 3.23) and a combination of queries Q_2 and Q_5 (for classification 3.24). Query $Q_{2.3}$ is: “such as each Hotel_cost is less than 250 FF”. Query $Q_{5.5}$ is: “such as each Transport_cost is less than 500 FF”.

The second class contains constraints modelling NP-complete problems (solved with heuristics) such as the traveling salesman problem (classification 3.6). These constraints must be managed by application softwares.

IV.5. Conclusion

From the user interface point of view, two alternatives are available to define constraints in a network-oriented query taking into account the notion of user's profile. The first one consists of providing the relevant cross product as a list of potential sub-queries. The second one consists of providing a two steps QDP. The user first defines its query. In a second, the QDP evaluates which classifications of constraints have not been filled and proposes this possibility to the user. The second is obviously the more user friendly. The methodology to define a network-oriented query is presented Figure IV.2.

At the database definition level:

(1) Specification of database attributes (Neutral, Time, Space, Applicative_units)

(2) Definition of the Departure_Hour function

(3) Definition of the Arrival_Hour function

During the QDP:

(1) Definition of the user's query

(2) Introduction of the constraints defined in the user's profile

(3) Classification of the query constraints (following Figure IV.1)

(4) For each constraints involving nodes (Global) or nodes and edges (Global)

Definition of the Visible function and/or modification of the Arrival_Hour function

Figure IV.2 - Methodology

V. CONCLUSION

The development of tourist applications such as way finding, merger between two types of services (e.g., plane and train or plane and hotel) shows the importance of network-oriented data. The size of geographical database implies defining precise queries to manage these data with a relevant and realistic computation time (reduced search space). Providing a unique path as a result is not sufficient since the shortest path may not be the more convenient, the more convenient may be far from being the cheapest, etc.

User's queries containt many "implicit" constraints that are not specified while defining a query. This paper addresses the required database modelling to take into account this set of constraints. “Implicit" constraints are based on the notions of time, space and applicative constraints. We introduce the characterization of alphanumeric attribute in the following classes: Neutral, Time, Space and Applicative_unit. Some aggregate functions on nodes requires evaluation this function on a sub-set of the nodes in a path (e.g., the cost of a hotel requires one to spend a night in a town). We introduce the notion of transparent nodes to model irrelevant nodes for an aggregate function. We define eight classes of generic functions to define a constraint: Global Time, Local Edge Time, Local Node Time, Global Space, Local Edge Space, Local Node Space, Time-Space mixed constraints and User's specific constraints.

REFERENCES

[1] Angelaccio M., Catarci T., Santucci G: QBD*: A Graphical Query Language with Recursion, IEEE Transaction on Software Engineering, Vol 16, n°10, 1990, p. 1150-1163

[2] Berge C: Graphes, Gauthier-Villars, Paris, 1983

[3] Boursier P, Mainguenaud M: Spatial Query Languages: Extended SQL vs. Visual Languages vs. Hypermaps, 5th Int. Symp. on Spatial Data Handling, Charleston, USA, 3-7 Aug. 1992

[4] Brossier-Wansek A, Mainguenaud M: Manipulations of Graphs with a Visual Query Language: Application to a Geographical Information System, 3rd Visual Database System, IFIP WG 2.6, Lausanne, Switzerland, 27-29 March 1995

[5] Cruz IF, Mendelzon AO, Wood PT: A Graphical Query Language Supporting Recursion, Proc. ACM SIGMOD Conference, San-Francisco, USA, May 1987

[6] Garey MR, Johnson DS: Computers and Intractability A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979

[7] Güting RH: Extending a Spatial Database System by Graphs and Object Class Hierarchies, Int. Workshop on DBMS for Geographical Application, Capri, Italy, 1991, May 1991, Springer-Verlag
Graph DB: Modeling and Querying Graphs in Databases, Proc. 20 th Very Large Data Bases Conference (VLDB), Santagio, Argentina, Aug. 1994

[8] Mainguenaud M: Consistency of Geographical Information System Query Results, Computer Environment and Urban Systems, Vol 18, pp 333-342, 1994

[9] Mainguenaud M: Modelling of the Geographical Information System Network Component, Int. Journal of Geographical Information Systems, Vol 9, n°6, pp -575-593, 1995

[10] Mendelzon AO, Wood PT: Finding Regular Paths in Graph Databases, 15th Int. Very Large Data Base Conference, Amsterdam, The Netherlands, Aug. 1989

[11] Peuquet DJ: A Concepetual Framework and Comparison of Spatial Data Models, Cartographica, Vol 21, 1984, pp 66-113

[12] Smith TR, Menon S, Star JL, Ester JE: Requirements and Principles for the Implementation and Construction of Large Scale GIS, Int. Journal of Geographical Information System, Vol□1 n°□1, 13-31, 1987

[13] Stonebraker M, Aderson E, Hanson E, Rubenstein B: QUEL as a Data Type, Proc. ACM SIGMOD Conference, Boston, USA, June 1984

[14] Stemple D, Sheard T, Bunker R: Abstract Data Types in Databases: Specification, Manipulation and Access, 2nd Int. Conference on Data Engeneering, Los Angeles, USA, Feb. 1986

[15] Ullman JD: Principles of Database and Knowledge-base Systems, Computer Science Press, Maryland, 1988