MANIPULATIONS OF GEOGRAPHICAL INFORMATION SYSTEM NETWORK COMPONENT

Langou B.
Mainguenaud M.

France Telecom - Institut National des Télécommunications
9 rue Charles Fourier
F91011 Evry - France
+ 33 1 60 76 47 14
+ 33 1 60 76 47 80 (fax)
Email: langou@etna.int-evry.fr Michel.Mainguenaud@int-evry.fr

Abstract:

In this paper, we present graph manipulations defined on the Graph Data Model. This data model allows the notion of abstraction for a node (or an edge). This model is designed to handle network component of a Geographical Information System.
Graph manipulations are divided into three classes. Basic operators manage the notion of abstraction (i.e., the DEVELOP and UNDEVELOP operators). Elementary operators manage the notion of graph and sub-graph (i.e., UNION, CONCATENATION, SELECTION and DIFFERENCE operators). High level operators manage GIS user interface operators (i.e., PATHS, INCLUSIONS and INTERSECTIONS operators).
A toy database has been developed with an object-oriented database system to validate the Graph Data Model and its operators.

Keywords:

Geographical information system, Network manipulations, Database operators

EGIS 94
Paris France
28/3 - 1/4 1994

I. INTRODUCTION

A Geographical Information System (GIS) must provide the management of network facilities (i.e., railway, electricity, telephone). A graph [3] (i.e., a set of labeled nodes and a set labeled edges) is the most common conceptual notion used to model network-oriented data. To capture more meaning, a graph data model must provide a mechanism of abstraction. A node (i.e., a town) may represent the abstraction of one (or several) network (i.e., a local area transportation network). An edge may also represent the abstraction of a network (i.e., a railway line).
In this paper, we present the manipulations defined on the Graph Data Model presented in [14]. This data model allies graph theory concepts (i.e., node, edge, graph) and object oriented database paradigm (i.e., abstraction). Since the data model is richer than a flat graph, specifications of database operators are more complex. This model is defined at a logical level (i.e., the notion of coordinates is not present). Several levels of interaction are defined to manipulate network-oriented data. Basic operators correspond to the first level (i.e., directly linked to the concept of abstraction in the Graph Data Model). Elementary operators correspond to the second level (i.e., the manipulations of graphs and sub-graphs). High level operators correspond to the third level (i.e., GIS user interface operators). We retained three classes of high level operators. They correspond to the most common queries addressed to a GIS [5]: the path evaluation, the intersection of paths and the inclusion of paths. Results of a query are influenced by the fact that the Graph Data Model presents a higher level of abstraction than a flat graph.

Part II presents some short recalls on the Graph Data Model; Part III presents the basic operators; Part IV presents the elementary operators; Part V presents the high level operators; Part VI presents a conclusion and future work.

II. SHORT RECALLS ON THE GRAPH DATA MODEL

The Graph Data Model allies notions of graph theory such as nodes, edges, graphs and notions of object-oriented paradigm such as abstraction. A node or an edge may represent the abstraction of sub-networks. We do not provide any constraint on the topology of this sub-network. In this paper, we use the notion of logical OID (Object IDentifier) to define the operators. This notion is similar to the notion of OID in object oriented database since the semantics is the same. Nevertheless, the OID defined in the Graph Data Model is completely system independent.
In this part, we present some short recalls of the notions defined in the Graph Data Model and used in this paper.

II.1. Basic concepts

The Graph Data Model is based on the concept of nodes, edges and graphs. These notions are similar to these notions in graph theory [3]. A graph is a couple (N, E) where N is a set of nodes and E is a sub-set of the Cartesian product N x N. The formal definition is presented Figure II.1.

$$N = \{n_1, \ldots, n_p\}$$
$$E = \{ (n_i, n_j) / n_i \in N, n_j \in N \}$$
(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 II.1 - Formal definition of a graph

In this paper, a graph is considered as oriented (i.e., the order, initial node / end node, is relevant). Several edges may be defined within the same initial and end nodes. We do not limit this number but we do not allow an edge (n_i, n_i). A node is used to model for example a town. An edge is used to model for example a link between two towns. Nodes and Edges are labeled. We do not consider in this paper the label associated to a node or an edge. Let a semantic information modelled by a node or an edge be named an object.

We define the notion of Master_node and Master_edge. They represent an abstraction of a sub-network. Such a sub-network is called an Associated_network. An Associated_network is defined as a graph G (N, E). Elements of N are called specialized nodes. Elements of E are called specialized edges. To connect this network to the different levels of abstraction, we define the concept In_edges and Out_edges. The In_edges (resp. Out_edges) of an Associated_network of a Master_node represent the set of edges "arriving to" (resp. "leaving") this graph. The In_edges and Out_edges are defined as presented Figure II.2.

$$In\_edges = \{ (n_i, n_j) / n_i \notin N \wedge n_j \in N \}$$
$$Out\_edges = \{ (n_i, n_j) / n_i \in N \wedge n_j \notin N \}$$

Figure II.2 - Formal definition of In_edges and Out_edges for a Master_node

The In_edges (resp. Out_edges) of an Associated_network of a Master_edge (e_i) represent the set of edges "leaving" (resp. "arriving to") the initial node (resp. end node) of a Master_edge. Let us define the function Initial_node (resp. End_node) of a Master_edge (e_i) such as Initial_node (e_i) (resp. End_node) returns the initial (resp. end) node of an edge (e_i). The In_edges and Out_edges for the Associated_network G (N, E) of a Master_edge, e_i, are defined as presented Figure II.3.

$$In\_edges = \{ (n_i, n_j) / n_i = Initial\_node\ (e_i) \wedge n_j \in N \}$$
$$Out\_edges = \{ (n_i, n_j) / n_i \in N \wedge n_j = End\_node\ (e_i) \}$$

Figure II.3 - Formal definition of In_edges and Out_edges for a Master_edge

II.2. Abstraction

The notion of abstraction [4,13,15] is introduced by the concept of OID. Each basic component (i.e., node, Master_node, edge, Master_edge, network, Associated_network) of the Graph Data Model has an OID. This OID is structured in several layers from L to 1. Each layer corresponds to a level of abstraction. To simplify the presentation, we present in this part the structuring of a node OID to take into account the notion of abstraction. The other OIDs are defined on the same principles. The structure of the OID will evolve along this paper to provide a complete structure whenever all the operators are defined.
The very beginning structure of an OID (i.e., the abstraction part) is defined by the rules presented Figure II.4 and with the database tuple constructor [].

The abstraction part of an OID is defined by a list of integer. An element of this list is called a layer. An object (o) in the layer k of abstraction is defined by:

- o is located at the top level of abstraction (i.e., k = L). The first layer has the local identifier (i.e., an integer) and the other layers are null.
- o is not located at the top level ($1 \leq k < L$). The (L - k) first layers contain each local identifier of the higher levels of abstraction. The layer k contains the local identifier. The (k - 1) last layers are null.

OID : [hierarchy_of_abstraction : list of integer]

Figure II.4 - The rules to build an OID

II.3. Conclusion

The Graph Data Model provides a structure to handle networks in a GIS context. The central notion is the abstraction of sub-networks. The notion is valid for nodes but also for edges. The Associated_networks, In_edges and Out_edges allow to model several levels of abstraction. The following sections present the various operators defined on the Graph Data Model.

III. BASIC OPERATORS

In this section, we present the operators directly linked to the concepts of the Graph Data Model (i.e., abstraction). Two basic operators are defined: the DEVELOP operator and the UNDEVELOP operator. The DEVELOP operator provides more specific details by merging a sub-network associated to a Master_node or to a Master_edge with the studied graph. The converse operation, the UNDEVELOP operator, provides a more restricted graph by the replacement of a sub-network by a Master_node or a Master_edge.

III.1. DEVELOP operator

The DEVELOP operator is applied to a set of graph. Each graph represents a path (i.e., only a unique Associated_network by Master_node and only a unique In_edge and Out_edge by Associated_network). On these graphs, some Master_nodes and/or Master_edges have to be developed. The Associated_networks are merged with the studied graph. The signature of this operator is presented Figure III.1.

	DEVELOP (G_o, {G_i}) : {G'_i}
where	G_o : is a graph with the set of nodes and /or a set of edges to be developed
	G_i : a graph to be expanded ($i = 1, ..., n$)
	G'_i : a graph after being expanded ($i = 1, ..., n$)

Figure III.1 - The signature of the DEVELOP operator

A difference is introduced between the development of a Master_edge and a Master_node. The development of a Master_edge implies the deletion of this Master_edge in the result since the Associated_network is merged with the studied graph. The development of a Master_node does not imply the deletion of this node since it may be the initial node (or the end node) of an edge (or a Master_edge).
As a convention, the DEVELOP operator provides a single development of Master_nodes and/or Master_edges. To provide a recursive application of the DEVELOP operator, a similar operator, DEVELOP*, is introduced.

III.2. UNDEVELOP operator

The UNDEVELOP operator is applied to a set of graph. Parts of these graphs are reduced. The sub-networks are reconstructed and transformed into Associated_networks. The Master_nodes and/or Master_edges are introduced into the studied graph.
The signature of the UNDEVELOP operator is presented Figure III.2.

	UNDEVELOP ({G_i}) : {G'_i}
where	G_i : a graph to be reduced ($i = 1, ..., n$)
	G'_i : a graph after being reduced ($i = 1, ..., n$)

Figure III.2 - The signature of the UNDEVELOP operator

To be able to reconstruct a sub-network as an Associated_network, the initial OID of this sub-network must be present in the OID of nodes and/or edges. The definition of the OID is therefore augmented with the abstraction of dependent networks. The semantics is the same as for the node hierarchy of abstraction. Figure III.3 presents the new structure of an OID.

OID :	[hierachy_of_abstraction : list of integer hierarchy_of_network : list of integer]

Figure III.3 - The new structure of an OID

As a convention, the UNDEVELOP operator provides a single reduction of sub-networks into Master_nodes and/or Master_edges. To provide a recursive application of the UNDEVELOP operator, a similar operator, UNDEVELOP*, is introduced.

III.3. Conclusion

The DEVELOP and UNDEVELOP operators are the core of graph manipulations. They allow to take full advantage of the abstraction defined in the Graph Data Model.
They are very similar to the NEST and UNEST operators in the algebra for complex objects [16]. Let R be a relation in Non First Normal Form. Let G be a graph structured with the Graph Data Model. Let G_0 be the graph containing the nodes (and/or the edges) to develop. Let G' be the graph G after applying DEVELOP (G_0, G). Figure III.4 presents the properties.

UNNEST (NEST (R)) = R
NEST (UNNEST (R)) # R

DEVELOP (G_0, UNDEVELOP (G')) = G'
UNDEVELOP (DEVELOP (G_0, G)) = G (from the Graph Data Model point of view)
UNDEVELOP (DEVELOP (G_0, G)) # G (with the introduction of alphanumerical data)

Figure III.4 - Properties of DEVELOP and UNDEVELOP operators

The DEVELOP operator may lead to deletion of Master_edges. The UNDEVELOP operator is able to reconstruct these edges. Unfortunately, alphanumerical data associated to these edges are no longer available. So, from the Graph Data Model point of view, the topology of the graph has been re-built. From the alphanumerical data point of view, information has been lost (as it is for the NEST/UNNEST configuration).

IV. ELEMENTARY OPERATORS

The elementary operators correspond to the second level operators. They defined operations on graphs and sub-graphs. They are similar to graph theory operators. Four operators are defined: the UNION, CONCATENATION, SELECTION and DIFFERENCE operators.

IV.1. UNION operator

The UNION operator provides the graph theory operator of union. This operator is defined between two sets of graphs. The main difference is introduced by the notion of abstraction. The union between a node and a Master_node representing the same object is defined as the Master_node. The union is recursively applied on sub-networks as soon as the two graphs have a common Master_node or Master_edge.
The signature of this operator is presented Figure IV.1.

$$\cup (\{G_i\}, \{G'_i\}) : \{G''_i\}$$

where G_i : is a graph $(i = 1, ..., n)$

G'_i : is a graph $(i = 1, ..., p)$

G''_i : is a graph after the application of the UNION on graph G_i and graph G'_i

$(i = 1, ..., n \times p)$

Figure IV.1 - The signature of the UNION operator

This operation does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

IV.2. CONCATENATION operator

The CONCATENATION operator is very similar to the UNION operator. To provide a result, the CONCATENATION operator requires a common object in the two graphs. Two graphs may have a common node without having common edges. Two graphs may have a common edge without having common nodes (i.e., the cross product of Master_node and node as initial and end nodes). Two graphs may have no common node and no common edge but a common object (i.e., an edge defined between two nodes and a Master_edge defined between two Master_nodes representing the same objects as the edges and the nodes do).

The concatenation is recursively applied on sub-networks. Whenever two graphs have no common object, the UNION operator provides a bi-graph. The CONCATENATION operator provides no result in such a configuration.
The signature of this operator is presented Figure IV.2.

$$\oplus\ (\{G_i\},\ \{G'_i\}\ ) : \{G''_i\}$$

where G_i : is a graph $(i = 1, ..., n)$
G'_i : is a graph $(i = 1, ..., p)$
G''_i : is a graph after the application of the CONCATENATION on graphs G_i and G'_i
$(i = 1, ..., k \quad / \quad k <= n \times p)$

Figure IV.2 - The signature of the CONCATENATION operator

This operation does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

IV.3. SELECTION operator

The SELECTION operator provides a sub-graph from a graph. The notion of selection criteria allows to reduce the number of nodes and edges. A selection criterion may be defined at various levels of abstraction (i.e., node, edge, network). As an example "A town having a number of inhabitant greater than 100,000" corresponds to a selection criterion associated to a node; "A cost less than 100 units" corresponds to a selection criterion associated to an edge; "Inter-city railway lines" correspond to a selection criterion associated to a network.
The main difference with a graph theory operator of selection is introduced by the notion of abstraction. The selection criteria are recursively applied to specialized nodes (or edges).
A Master_node (or a Master_edge) may be transformed to a node (or an edge) whenever the Associated_networks become empty. A Master_node may be removed and replaced by the result of the DEVELOP operator whenever it does not respect the selection criteria. A Master_edge may be changed to an edge whenever at least one edge of its Associated_network does not respect the selection criteria. Its Associated_network is merged (i.e., the DEVELOP operator and the UNION operator) with the studied graph.
The signature of this operator is presented Figure IV.3.

$$\sigma\ (\text{Criteria}, \{G_i\}\ ) : \{G'_i\}$$

where Criteria : represents the modelling of the selection criteria
G_i : is a graph $(i = 1, ..., n)$
G'_i : is a graph after the application of the SELECTION on graph G_i
$(i = 1, ..., k \quad / \quad k <= n)$

Figure IV.3 - The signature of the SELECTION operator

This operation does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

IV.4. DIFFERENCE operator

The DIFFERENCE operator provides the graph theory operator of difference. This operator is defined between a set of graphs and a graph. To introduce a change, the DIFFERENCE operator requires a common object in the two graphs (i.e., a node, Master_node, edge or Master_edge). The difference is recursively applied on sub-networks. A Master_node (or a Master_edge) may be transformed to a node (or an edge) whenever the Associated_networks become empty.
Whenever a Master_node (or a Master_edge) with a non-empty Associated_network has to be removed, the Associated_network is merged (i.e., with the DEVELOP and UNION operators) to the studied graph.
The signature of this operator is presented Figure IV.4.

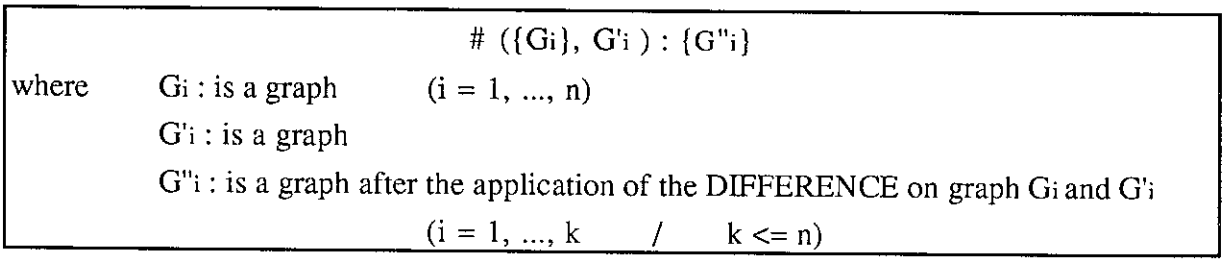

Figure IV.4 - The signature of the DIFFERENCE operator

This operation does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

IV.5. Conclusion

The elementary operators allow to construct a graph on which high levels operators are applied. They are very similar to the operators of the relational algebra [16] (i.e., selection, union, difference). A GIS query may be complex and involves several operators. The notion of execution plan of a relational query is similar to the composition of elementary operators from a high level operator.

V. HIGH LEVEL OPERATORS

In this part, three classes of operators based on classical graph manipulations are presented. Nevertheless, they are defined in the context of the Graph Data Model. They are designed to manage most part of GIS queries. The PATHS operator is the evaluation of paths between an origin and a destination [2,8,9,11,12]. The INCLUSION operators are defined from an operator of extraction by the inclusion of nodes in a path and the inclusion of a path in a path. The INTERSECTION operators present the same dichotomy: the intersection of two sets of nodes provided by two paths and the intersection of two sets of edges provided by two paths.

V.1. PATHS operator

The network management is not correctly handled by classical commercial GIS products. Generally, network processing queries are solved by an extra-component of the GIS. Path evaluation operator (often restricted to the shortest path) is not integrated in the data base operators. This weakness [10] is also present in the database query language since no extended SQL has a correct management of the path operator (even those dedicated to the transitive closure management). Defining a complete framework to express network-oriented queries is not so simple. [5] presents a set of "path" queries. These queries involve a transitive closure query with fifteen classes of interactions (such as constraints on nodes, edges, nodes and edges).
Furthermore, the definition of the result as a unique result graph is not relevant for the GIS context. The result of a path query must be a set of paths. As an example in the airplane applications, the shortest in time is generally far from being one of the less expensive (but what is the correct cost function?). The cheapest is generally far from being one of the more convenient and so on. Unfortunately, a graph defined by the union of the individual solutions is not relevant as a result as soon as aggregate constraints are defined in the query (which is generally the case).
The "classical" DBMS are based on the principle of the Close World Assumption: "If a datum does not logically follow from a set of database operations, then we conclude that the negation of this datum is valid." Therefore, the path operator provides an answer only if the complete (from the initial node to the end node) path exists. This option is coherent in the case of a flat network. The Graph Data Model allows to introduce several levels of definition. Therefore, we introduce the principle of the Open Graph Assumption: "While evaluating a path operator at a level of abstraction i, if a datum does not logically follow from a set of Graph Data Model operations, then the same evaluation is performed at level i+1." The Open Graph Assumption introduces the notion of a global direction since the level i+1 is a more general edge. Therefore, the path operator (->) presents two options: the exact operator and the approximated operator. The exact operator is the equivalent of a classical path operator. It requires the existence of the complete path. The approximated operator takes full advantage of (1) the node hierarchy and (2) the network topology to provide a multi-levels solution.
The signature of this operator is presented Figure V.1.

	-> (G, G', Criteria) : {G''i}
where	G : is a graph (N,E) such as E = ∅
	G' : is a graph (N,E) such as E = ∅
	Criteria : the path criteria [7] and constraints
	G''i : is a graph after the application of the PATHS operator on graph Gi and G'i

Figure V.1 - The signature of the PATHS operator.

This operation does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

V.2. INCLUSION operators

High level operators of inclusion provide three kinds of results. These distinctions are linked to the construction of a graph (i.e., a set of nodes and a set of edges). Π_{e_ne} represents the operator for the extraction of nodes in a path; Π_{ic_n} represents the set operator for the inclusion of nodes; and Π_{ic_e} represents the set operator for the inclusion of edges.

V.2.1. EXTRACTION of nodes from a path

The Π_{e_ne} operator is applied to a set of graphs. This operator provides for each graph the set of nodes involved in the edge definition (i.e., there is no isolated node). The signature of this operator is presented Figure V.2.

	Π_{e_ne} ({Gi}) : {G'i}
where	Gi : is a graph G(N,E)
	G'i : is a graph (such as E = ∅) after the application of the Π_{e_ne} on graph Gi

Figure V.2 - The signature of the Π_{e_ne} operator.

This operation does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

V.2.2. INCLUSION of nodes

The Π_{ic_n} operator is applied to two sets of nodes. This operator is not symmetric. It requires the first set of nodes to be included into the second set of nodes. The result of this operator is the first set of nodes whenever it fulfills the requirements. The signature of this operator is presented Figure V.3.

	Π_{ic_n} (G, G') : G"
where	G : is a graph G(N,E) such as E = ∅
	G' : is a graph G(N,E) such as E = ∅
	G" : G or a graph G"' (N,E) such as N = ∅ and E = ∅

Figure V.3 - The signature of the Π_{ic_n} operator.

This operation does not imply a modification of the OID definition since no graph is created.

V.2.3. INCLUSION of edges

The Π_{ic_e} operator is applied to two sets of edges. This operator is not symmetric. It requires the first set of edges to be included into the second set of edges. The result of this operator is the first set of edges whenever it fulfills the requirements. The signature of this operator is presented Figure V.4.

	Π_{ic_e} (G, G') : G"
where	G : is a graph G(N,E)
	G' : is a graph G(N,E)
	G" : G or a graph G"' (N,E) such as N = ∅ and E = ∅

Figure V.4 - The signature of the Π_{ic_e} operator.

This operation does not imply a modification of the OID definition since no graph is created.

V.3. INTERSECTION operators

High level operators of intersection provide two kinds of results. This dichotomy is linked to the construction of a graph (i.e., a set of nodes and a set of edges). Π_{is_n} represents the operator for the set intersection of nodes and Π_{is_e} represents the operator for the set intersection of edges.

V.3.1. INTERSECTION of nodes

The Π_{is_n} operator is applied to two sets of nodes. The intersection operator is recursively applied to the sub-networks. A Master_node may be transformed to a node whenever Associated_networks become empty. The intersection between a node and a Master_node representing the same object is defined as the node.
The signature of this operator is presented Figure V.5.

	Π_{is_n} (G, G') : G"
where	G : is a graph G(N,E) such as E = ∅
	G' : is a graph G(N,E) such as E = ∅
	G" : is a graph after the application of the Π_{is_n} on graph G and G'

Figure V.5 - The signature of the Π_{is_n} operator.

This operator does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

V.3.2. INTERSECTION of edges

The Π_{is_e} operator is applied to two sets of edges. The intersection operator is recursively applied to the sub-networks. A Master_edge may be transformed to an edge whenever the Associated_network becomes empty. The intersection between an edge and a Master_edge representing the same object is defined as the edge.
The signature of this operator is presented Figure V.6.

	Π_{is_e} (G, G') : G"
where	G : is a graph
	G' : is a graph
	G" : is a graph after the application of the Π_{is_e} on graph G and G'

Figure V.6 - The signature of the Π_{is_e} operator.

This operator does not imply a modification of the OID definition since no change is applied to the structure of the networks. New graphs are created but the intentional construction of an OID is not affected.

V.4. Conclusion

The main difference between graph theory and GIS is based on the cost evaluation function. To simplify, the time to evaluate a cost function of a node (or an edge) is considered as negligible in graph theory during the detection of a path from one node to another. Unfortunately in a GIS context,

these functions are far from being negligible since they involve several database operations. GIS queries are more interested in the properties defined on the nodes or on the edges than on the topology of an answer. Nevertheless, the notion of abstraction and the notion of global direction are very important. The Graph Data Model provides the topology of a graph. This assumption is based on the fact that topologies of geographical networks (i.e., railway lines, rivers, roads) are not modified very frequently (although alphanumerical data may be frequently updated). A pre-compiled structure becomes realistic. The data model takes full advantage of this condition. The PATHS operator offers therefore a higher expressive power (i.e., abstraction and direction).

VI. CONCLUSION

Geographical Information Systems are now widely used for several applications (i.e., transport analysis, urban planning). Most part of the time, the network analysis component is not directly integrated into the database system. This weakness is due to the lack of common data structures between thematic-oriented data (i.e., town, forest) and network-oriented data (i.e., railways, electricity).

In this paper, we present the operators defined on the Graph Data Model allowing the notion of abstraction. We define three levels of operators. Basic level operators, the DEVELOP and UNDEVELOP operators, are directly linked to the concept of abstraction present in the data model. The DEVELOP operator provides more details for a graph. The converse operation is the UNDEVELOP operator. Elementary operators manage the notion of graphs and sub-graphs. The SELECTION, UNION, CONCATENATION, DIFFERENCE operators, are similar to the basic operators of graph theory (i.e., extended to a data model allowing abstraction for nodes and edges). High level operators, the PATHS, INCLUSIONS and INTERSECTIONS operators, correspond to the most common classes of queries addressed to a GIS.

The high levels operators introduce a new conception in graph operators for GIS since the result is defined with several levels of abstraction. This notion changes the visualization of a result. A toy application is defined with the O2 object oriented database system [1] to validate the Graph Data Model and its operators. The Cigales language is used as a user interface to define a query [6]. Next work is to define a formal Data Definition Language for the Graph Data Model. The notion of view can now be integrated since the data manipulation operators are available.

References

[1] Bancilhon F. et al: The design and Implementation of O2, an Object Oriented Database System, 2nd Int. Workshop on Object-Oriented Data Base Systems, K. Dittrich (Ed) Bad-Munster, FRG, September 1988

[2] Becker RA et al: Network Visualization, 4th International Symposium on Spatial Data Handling, Zurich, Switzerland, 23-27 July 1990

[3] Berge C. : Graphs and Hypergraphs, North Holland, Amsterdam, 1973

[4] Borgida A, Myropoulos J, Wong HK, Generalization/Specialization as a Basis for Software Specification, Brodie, Myropoulos, Schmidt Eds "On Conceptual Modelling perspectives from Artificial Intelligence, Database and Programming Languages, Springer Verlag, 1984

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

[6] Calcinelli D., Mainguenaud M.: The Management of the Ambiguities in a Graphical Query Language for GIS, 2nd Symposium on Large Spatial Databases, Zurich, Switzerland, August 28-30, 1991, Lecture Notes in Computer Science n°525, Springer Verlag

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

[8] Dayal U. et al : PROBE - A research Project in Knowledge-Oriented Database Systems: Preliminary Analysis, Technical Report CCA-85-03, July 1985

[9] Eck (van) JR, De Jong T.: Adapting Datastructures and Algorithms for Faster Transport Network Computations, 4th International Spatial Data Handling Symposium, Zurich, Switzerland, July 24-27 1990

[10] Garey MR, Johnson DS: Computers and Intractability: A Guide to the Theory of NP-Completeness, WH Freeman and Co Eds, New York, 1979

[11] Güting RH: Extending a Spatial Database System by Graphs and Object Class Hierarchies, International Workshop on Database Management System for Geographical Applications, Capri, Italy, May 1991

[12] Haas LM, Cody WF: Exploiting Extensible DBMS in Integrated Geographical Information Systems, 2nd Symposium on Large Spatial Databases, Zurich, Switzerland, August 1991, Lecture Notes in Computer Science n° 525, Springer Verlag

[13] Hull R, King R: Semantic Database Modelling: Surveys, Applications and Research Issues, ACM Computing Surveys, Vol 19, September 1987

[14] Mainguenaud M, Simatic XT: A Data Model to deal with Multi-scaled networks, Computer, Environment and Urban Systems, Vol 16, p 281-288, 1992

[15] Smith JM, Smith DC, Database Abstraction: Aggregation and Generalization, ACM TODS, Vol 2 n°2, 1987

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