Modelling of the geographical information system network component

Short title:

Modelling of GIS network component

M. MAINGUENAUD

FRANCE TELECOM - Institut National des Télécommunications
9 Rue Charles FOURIER
F91011 EVRY - FRANCE
+ 33 1 60 76 47 82
+ 33 1 60 76 47 80 (fax)
Email: Michel.Mainguenaud@int-evry.fr

Keywords:

Geographical Information System, Data modelling, Data manipulation operators, Network, Object oriented modelling, Graph

Int. Journal of GIS
1995
Vol 9 n°6

Modelling of the geographical information system network component

Abstract:

This paper presents a data model and some considerations on the path operator to manage networks with a Geographical Information System.
The data model is based on the merge of the graph theory concepts and the object-oriented paradigm. The introduction of the Master_nodes (resp. Master_edges) allows to define a node (resp. an edge) as an abstraction of a sub-network. A network can be defined using several levels of definition (detail) following the importance of the nodes (resp. the edges) within an application. The object-oriented concepts are applied to the alphanumerical data part and to model the topology of the graph. The data manipulation operators are basic operators. These operators are: the union, the intersection and the difference. More elaborated operators (i.e., path under constraints) are obtained by combining these basic operators.
We present some considerations on the path operator in a GIS context and introduce the concept of "the Open Graph Assumption" and the notion of direction. These notions take full advantage of the data model. They represent an alternative way of defining the path operator.
This modelling is well adapted to take into account the multi-levels networks (i.e., the definition of a view in the context of multi-users configuration or the introduction of the generalization).

1. Introduction

In the current researches toward the design of more powerful tools for urban planning, remote sensing, ... different research groups are simultaneously concentrating their work on Geographical Information System (GIS). GIS needs are very well known (Smith et al 87) but several problems are still open. In this paper we focus on the data modelling and the data manipulation operators to take into account the network-oriented component of a GIS. Traditionally, geographic informations are divided into two parts: the polygonal-oriented data (i.e., town, forest) and the network-oriented data (i.e., railways, water). This distinction is due to the lack of common data structures and operators. Polygonal-oriented operators are based upon the spatial representation (i.e., intersection). Network-oriented operators are based upon graph manipulations whatever the spatial representation is (i.e., node, edge of a logical graph).

Transport networks (i.e., train, plane, water, telecommunication) may be modeled with the graph theory concepts. Early 1970, numerous works were based on the optimization of the graph operators. The underlying data model was only composed of nodes and edges (or fully connected sub-networks). The graph theory (Garey and Johnson, 1979) applied to operations research represents an important theoretical base for efficient data manipulation operators. Nevertheless, current implementations of this formalism have a lack of semantic power if we compare them to the semantic data models (Borgida et al 1984, Hammer and Mc Leod 1981, Hull and King 87, Peckham and Maryanski 1988) or the object-oriented data models (Bancilhon et al 1991, Banerjeree et al 1987, Manola and Orenstein 1986). The main advantage of a Data Base Management System (DBMS) is its ability to handle a huge amount of data. The underlying data model tries to avoid the redundancy of the data because this can involve incoherencies while querying or updating the data base. A view mechanism allows to define part of the data base as a user's world. The data model we present in this paper is well suited to the partial definition of a network (i.e., view). In fact, object-oriented concepts such as classification, aggregation, generalization/specialization (Smith and Smith 1987) can be applied to the alphanumeric information part of the network but also to the topology of this network. This topology can be represented with the abstraction concept of the object-oriented paradigm. The data model tries to take full advantage of this representation by allowing nodes and edges to be abstractions of sub-networks.

The aim of this paper is first to present a data model allying the power of the graph theory and the object-oriented paradigm and second the data manipulation operators associated to this data model. Section 2 presents the Graph Data Model; section 3 presents the management of the several levels of abstraction; section 4 presents some considerations on the path operator and presents an alternative way of defining the path operator; section 5 presents the implementation; section 6 deals with the conclusion and future work.

2. Data model

Three levels can be defined to model network-oriented data: the geometrical representation (i.e., vector or raster), the logical graph topology (i.e., the town is an abstraction of a local area network) and the application-dependent data (i.e., the population of a town). Figure 1 presents the three levels.

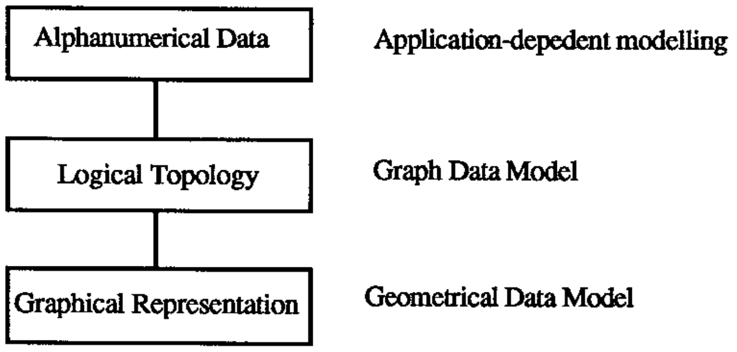

Figure 1

The three levels can be defined with the same formalism. Several propositions have been defined to model geometrical data and application dependent-data or coupling both (i.e., Bennis et al 1990, David et al 1993, Haas and Cody 1991, Scholl and Voisard 1989, Vijlbrief and Oosterom 1992). Graph data models (i.e., Becker et al 1990, Cruz et al 1987, Van Eck and De Jong 1990, Güting 1991, Kunii 1983, Mainguenaud and Simatic 1991) have received fewer propositions particularly in a Geographical Information System context. Applications of graph data models can be found for example in the definition of EDIGéO or in GDF (Heres et al 1993).

The Graph Data Model presented in this part is based on an object-oriented paradigm. Our goal is not to define a new data model but to use the classical constructors available in the literature. We assume the reader to be familiar with the object-oriented paradigm (Zdonic and Maier 1989). A class is defined as a group of objects having the same structure and the same behaviour. A type represents the structure of a class. In this paper, a class is defined using the tuple constructor (aggregation), the set constructor and with the classical basic types (i.e., integer, string). To illustrate the examples given in this paper we use the O_2 data model formalism (Bancilhon et al 1991). The classes of the Graph Data Model are organized in a tree. This tree represents the relationships between classes and super-classes and may be seen as an inheritance tree of the attributes. The constructors are used to model the topology of a graph to allow various levels of definition. Therefore we do not consider the attributes such as the population of a town or the departure and arrival time for an edge modelling a train connection nor their geometrical representations on a map. The toy example provided in this paper is based on French towns (Paris, Lyon, Marseille, Nice, Reims, Strasbourg, Quevy, Antibes, Cannes, Toulon, Roissy, Rennes, Bordeaux, Dijon, Valence, Montelimar, Orange, Toulouse). Some towns may have several railway stations (i.e., Gare De Lyon - GDL-, Gare Du Nord -GDN-, Maillot for Paris; Perrache, Part-Dieu for Lyon). The semantics of the toy example is very similar to the semantics of the examples used in Artificial Intelligence or Operations Research (Bratko 1987).

In this part we first recall the basic notions of graph theory to model a network, the basic components of the model (the node, the edge, the network), the second level of abstraction (the Associated_network, the Master_node, the Master_edge).

2.1. Basic notions of graph theory

We model the network with the concept of graph. 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.

$$N = \{n_1, \dots , n_p\}$$
$$E = \{ (n_i, n_j) / n_i \in N, n_j \in N \}$$

(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))

In this paper, the graph is considered as oriented. 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 or a crossing point between two roads. An edge is used to model for example a link between two towns. The nodes and the edges are labeled. Let us suppose we can define a labelling function that allows to name and to give some characteristics to the nodes and to the edges. We do not consider here the application-dependent part of the label.

2.2. Basic components of the Graph Data Model

The predefined classes of the model are: Node, Edge and Network. For each of them we give its declaration using the O2 formalism and its graphical representation.

2.2.1. Nodes

The class Node is defined by an aggregation of one attribute (name):

```
class Node
      private
      type tuple
            (
              name: string
            )
end;
```

Let us suppose we want to represent a town (node) labeled "Paris". To do so, an object of the class Node is defined. This node is graphically represented in figure 2.

Figure 2. A node

2.2.2. Edges

The class Edge is defined by an aggregation of three attributes (name, initial_node, end_node):

```
class Edge
      private
      type tuple
           (
             name: string,
             initial_node: Node,
             end_node: Node
           )
end;
```

Let us suppose we want to represent a link (edge) labeled "TGV" between two towns (nodes) labeled "Lyon" and "Paris". To do so an object of the class Edge is defined. This edge is graphically represented in figure 3.

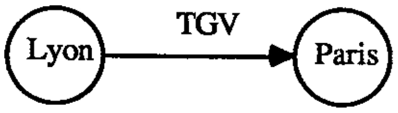

Figure 3. An edge

2.2.3. Network

Before defining the Network class, let us define the graph type since the node and edge definitions are available:

```
type graph
      tuple
           (
             nodes: unique set (Node),
             edges: unique set (Edge)
           )
```

The class Network is defined by an aggregation of two attributes (name, graph):

```
class Network
      private
      type tuple
          (
            name: string,
            graph: graph
          )
end;
```

Let us suppose we want to represent a network labeled "Railways" with two edges (Marseille, Lyon) and (Lyon, Paris) labeled "TGV". To do so an object of the class Network is defined. Let N be a finite set $\{n_1, ..., n_p\}$. A hypergraph on N is a family $H = (E_1,...,E_k)$ of parts of N such as:

$E_i \# \emptyset \qquad (i=1,..., k)$

$\cup E_i = N \qquad (i=1,..., k)$

The network is graphically represented by figure 4 with a hypergraph.

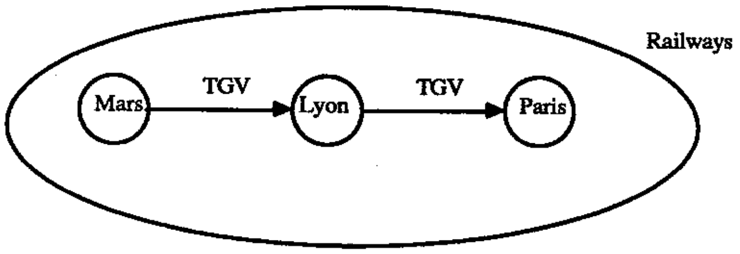

Figure 4. A network

2.3. The second level of abstraction

The decomposition following several levels of abstraction is a natural way of structuring network-oriented data. In the Graph Data Model, we consider a second level of abstraction: the Master_node/Master_edge level. The Master_node (resp. the Master_edge) allows defining a node (resp. an edge) as an abstraction of a sub-network.

Figure 5 presents a "classical" graph with only one level of abstraction. Figure 6 presents a multi-levels network. Paris is now considered as an abstraction of a sub-network. One can remark an edge between Paris and Quevy and an edge between Paris and Reims (figure 5). The first edge has an initial node Paris changed by GDN when the development of the node Paris has been performed, nevertheless Paris is still the initial node for the second edge since the very initial node may be Paris or a node that does not belong to this development of the node Paris.

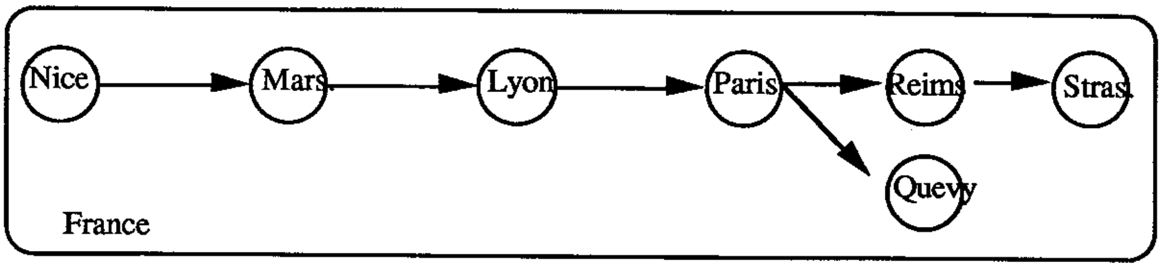

Figure 5. A flat network

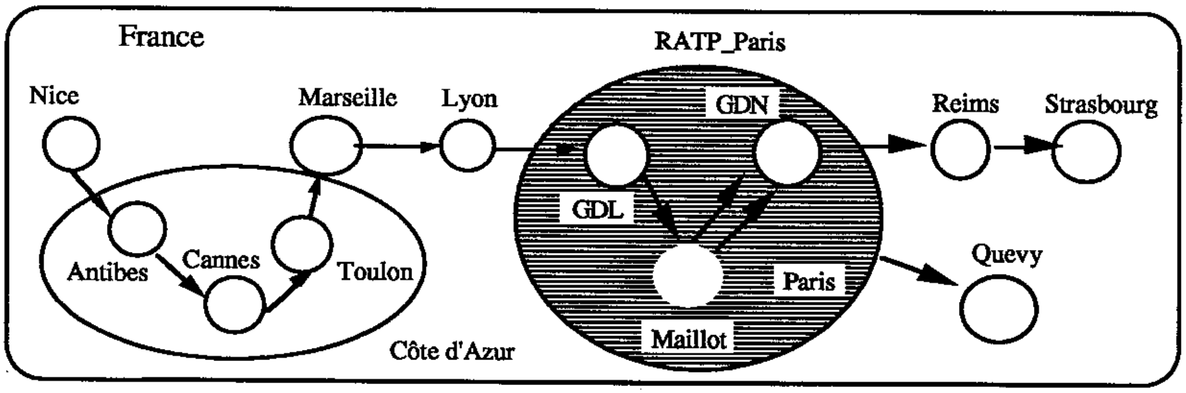

Figure 6. A multi-levels network

2.3.1. Associated_network

The Master_node and the Master_edge represent an abstraction of a sub-network. Such a sub-network is called an Associated_network. The class Associated_network is a sub-class of the class Network. An Associated_network is defined as a graph G (N, E). The elements of N are called the specialized nodes. The elements of E are called the 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:

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

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

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

In figure 6, the edge (Lyon, GDL) is an In_edge for the Associated_network of the Master_node Paris. The edge (GDN, Reims) is an Out_edge. The edge (Nice, Antibes) is an In_edge for the Associated_network to the Master_edge (Nice, Marseille). The edge (Toulon, Marseille) is an Out_edge.
The class Associated_network is defined by an aggregation of two attributes (In_edges, Out_edges):

```
class Associated_network
      inherit Network
      private
      type tuple
           (
             In_edges: unique set (Edge),
             Out_edges: unique set (Edge)
           )
end;
```

2.3.2. Master_node

The class Master_node is a sub-class of the class Node. The attribute name is inherited from the class Node. This class is defined by an aggregation of one attribute (associated_networks):

```
class Master_node
      inherit Node
      private
      type tuple
           (
             associated_networks: set (Associated_network)
           )
end;
```

A Master_node (i.e., Paris), called a generalized node, is graphically represented in figure 7.

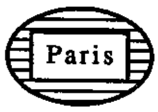

Figure 7. A Master_node

2.3.3. Master_edge

A similar notion is defined for the edges. The class Master_edge is a sub-class of the class Edge. This class is defined by an aggregation of one attribute (associated_network). The name, initial_node, end_node attributes are inherited from the class Edge:

```
class Master_edge inherit Edge
      private
      type tuple
            (
             associated_network: Associated_network
            )
end;
```

A Master_edge (i.e., the edge (Nice, Marseille)), called a generalized edge, is graphically represented in figure 8.

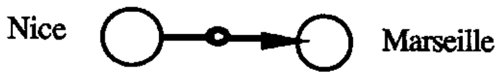

Figure 8. A Master_edge

2.4. Conclusion

This data model is well adapted to manage several levels of abstraction. The main difference between the Associated_networks of the Master_node and the Associated_network of the Master_edge relies on the set of nodes. The initial node and the end node of a generalized edge (i.e., the Master_edge) belong to the set of nodes of the Associated_network (since the generalized edge does not belong any more to the expanded graph - see figure 6 -). The main difference between a Master_node and a Master_edge relies on the number of the Associated_network. A Master_node may have several Associated_networks. A Master_edge has a unique Associated_network since the initial node and the end node of the generalized edge belong to the set of nodes of the Associated_network. Nevertheless, several Associated_networks may be defined using several Master_edges between the same initial node and end node. Figure 9 presents the network of figure 5 with the introduction of the Master_node and the Master_edge.

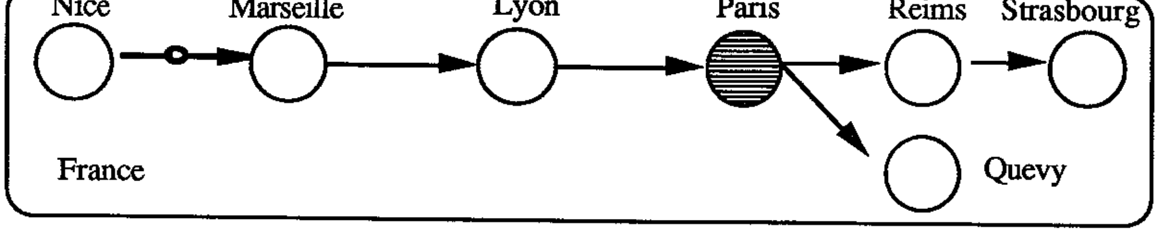

Figure 9. A network with Master_nodes and Master_edges

3. Management of several levels of abstraction

The class hierarchy of the Graph Data Model represents the framework for a node and edge classification. The nodes and the edges are also organized in a hierarchy. This hierarchy is used to define the structure of the database. The applications define their data on this hierarchy, on part of it (i.e., the view concept), or on a restructuring of it. The hierarchy is defined by a logical OID - Object IDentifier - (completely independent from the system OID of an OODBMS). Each basic element of the Graph Data Model (node, edge, network) has a logical OID.

This part presents the node hierarchy, the edge hierarchy, the network hierarchy and the passage from the hierarchies to the database.

3.1. Node hierarchy

A node belongs to a hierarchy of node. Each level of this hierarchy is called a layer. These layers represent a particular level of abstraction. Let N be the depth of this hierarchy. The OID has N layers: from layer N, the higher level of abstraction, to 1, the lower level of abstraction.
Figure 10 presents an example of a hierarchy with three levels of abstraction.

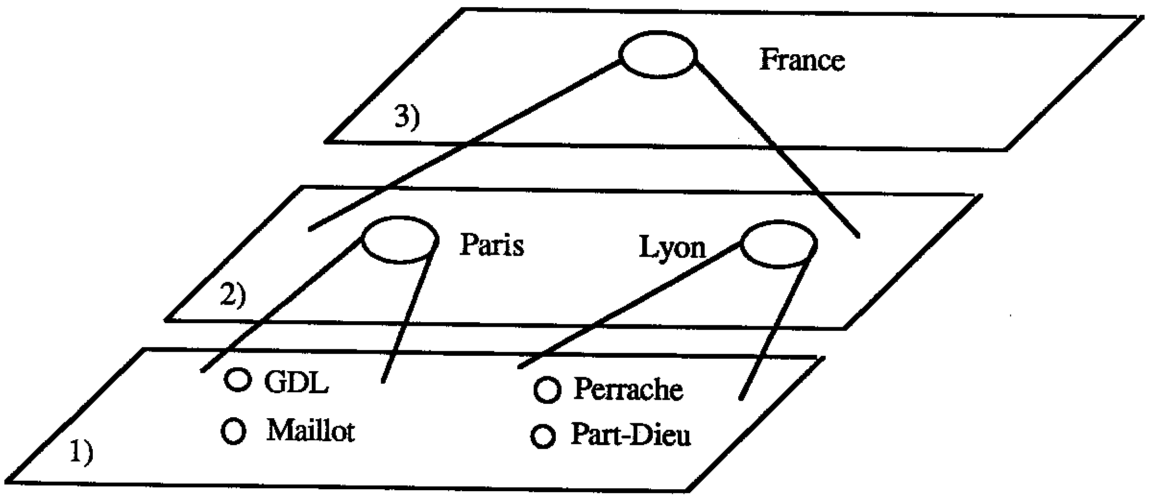

Figure 10. A node hierarchy

An OID is defined by a list of integer. An element of this list is called a layer. An OID of a node n in the layer k is defined by the following rules:

- n is located at the top level of abstraction (i.e., k = N). The first layer has the local OID and the other layers are null

- n is not located at the top level ($1 \leq k < N$). The (N - k) first layers contain each local OID of the generalized nodes. The layer k contains the local OID. The (k - 1) last layers are null.

Let N be the layer of a node (i.e., France). This node is called the generalized node of all its depending nodes (i.e., Paris, Lyon). Following the rules of the OID definition, let 300 be the local OID associated to the node France, 310 be the local OID associated to the node Paris and 320 be the local OID associated to the node GDL. Figure 11 illustrates the OID of these nodes.

France	(300	0	0)
Paris	(300	310	0)
GDL	(300	310	320)

Figure 11. OID of nodes

Depending on the various levels of abstraction already defined in the database, a node with a layer i (i.e., Roissy layer 1) may depend on the generalized node with a layer i+2 (i.e., France layer 3). To cover all the available layers, a shadow node is used at the layer i+1. The local OID associated to this node is the opposite of the local OID of the generalized node (i.e., -300). The name of the shadow node is a string built by the concatenation of '*' and the name of the generalized node (i.e., *France). This construction is recursively applied whenever the number of the missing levels is more than one, but the local OID of the various shadow nodes is still the opposite of the generalized node OID. There is only one shadow node for each group of dependent nodes (to simplify the data manipulation, a shadow node is always available for each group of dependent nodes).

Figure 10 is now transformed into figure 12 and figure 13. They present the hierarchy of the node France with the associated OID.

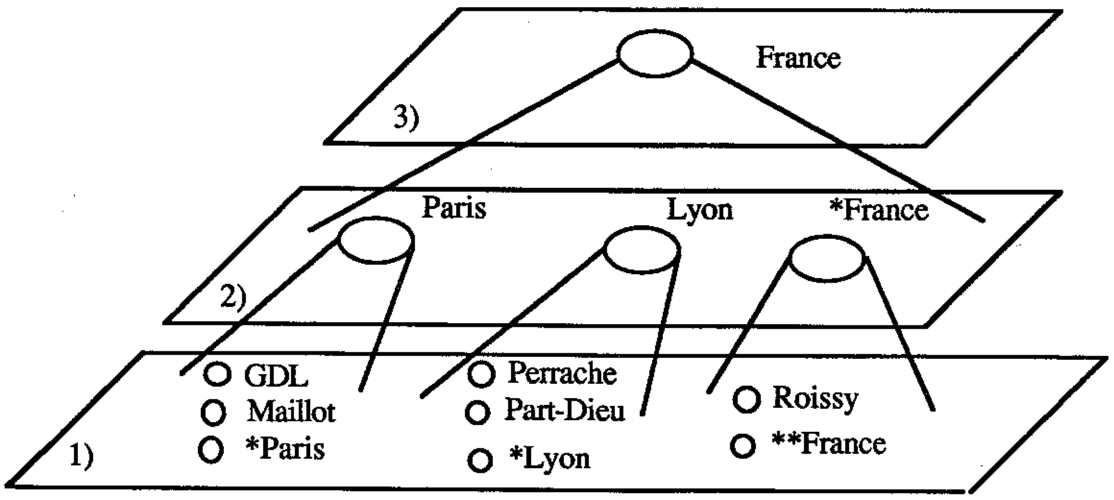

Figure 12. A node hierarchy

France	(300 0 0)				
Paris	(300 310 0)	Lyon	(300 321 0)	*France	(300 -300 0)
GDL	(300 310 320)	Maillot	(300 310 336)	*Paris	(300 310 -310)
Perrache	(300 321 338)	Part-Dieu	(300 321 337)	*Lyon	(300 321 -321)
Roissy	(300 -300 347)	**France	(300 -300 -300)		

Figure 13. OID of nodes

3.2. Edge hierarchy

This hierarchy is similar to the hierarchy of nodes. The OID of an edge is divided into three parts. Part 1 (resp. 2) contains the OID of the initial node (resp. end node). Part 3 (the edge part) contains the OID in the hierarchy of edge. The construction of the third part is similar to the construction of the node OID. An edge belongs to a hierarchy of edge. Each level of this hierarchy is called a layer. These layers represent the levels of abstraction. Figure 14 presents an example of a hierarchy with two levels. Paris and Lyon appearing in the level 1 and 2 are the same nodes.

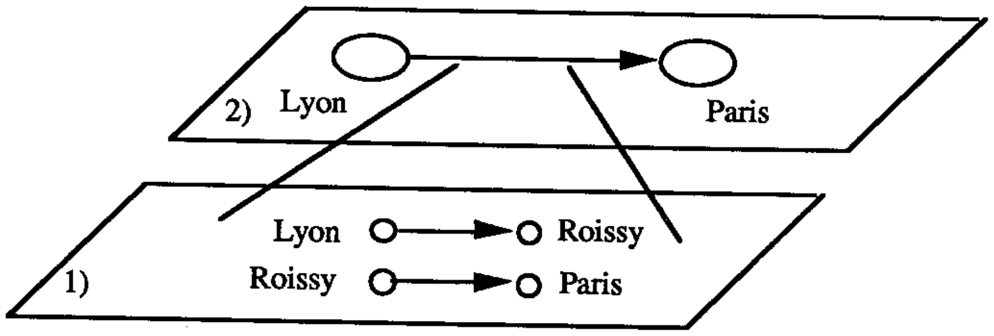

Figure 14. An edge hierarchy

Let 200 be the local OID associated to the edge (Lyon, Paris), 210 be the local OID associated to the edge (Lyon, Roissy), 220 be the local OID associated to the edge (Roissy, Paris). Figure 15 presents the OID of such edges.

(Lyon, Paris)	((300 310 0) (300 321 0) (200 0))
(Lyon, Roissy)	((300 321 0) (300 -300 347) (200 210))
(Roissy, Paris)	((300 -300 347) (300 310 0) (200 220))

Figure 15. OID of edges

Depending on the various levels of abstraction, a notion of shadow edge is created. Its aim is similar to the shadow node. The edge part of a shadow edge follows the same rules as the shadow node. A unique shadow edge is defined between the same initial and end nodes.
To simplify the examples of this paper, we introduce here a new example, based on figure 6, to illustrate the shadow edge. Figure 16 presents the hierarchy of edges.

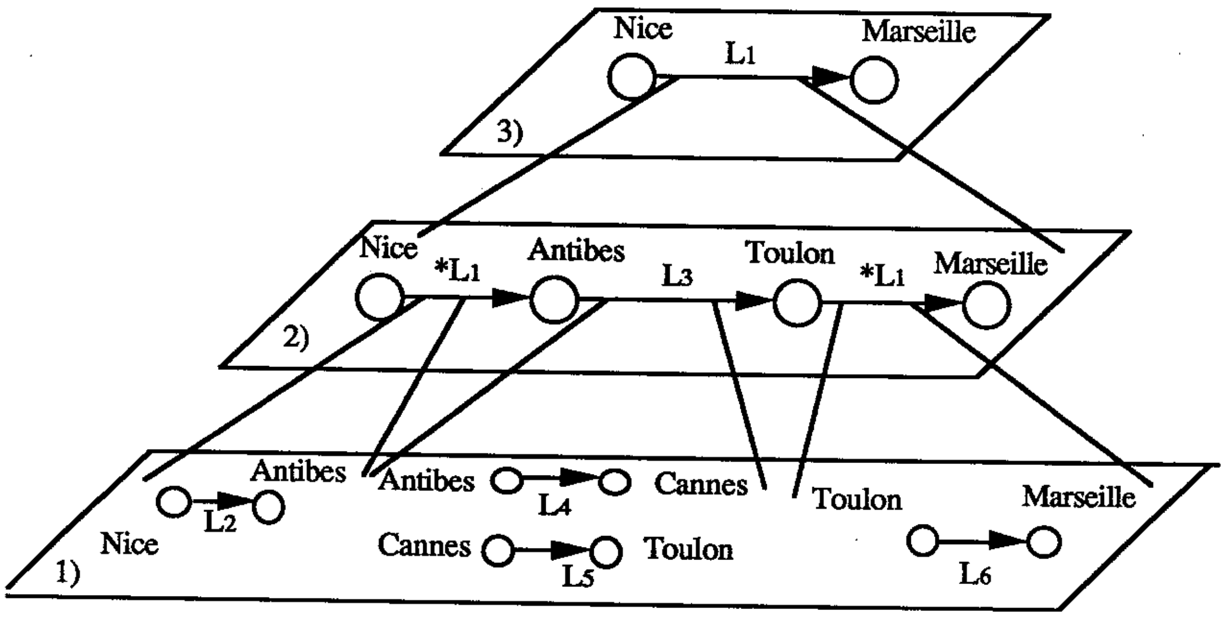

Figure 16. Hierarchy of edges

Let 300 be the local OID of the edge (Nice, Marseille), 310 for the edge (Nice, Antibes), 320 for the edge (Antibes, Toulon), 330 for the edge (Antibes, Cannes), 340 for the edge (Cannes, Toulon), 350 for the edge (Toulon, Marseille). Figure 17 presents the corresponding edge part of the OID.

L1	((...) (...) (300	0	0))
*L1	((...) (...) (300	-300	0))
L2	((...) (...) (300	-300	310))
L3	((...) (...) (300	320	0))
L4	((...) (...) (300	320	330))
L5	((...) (...) (300	320	340))
*L1[1]	((...) (...) (300	-300	0))
L6	((...) (...) (300	-300	350))

Figure 17. OID of edges

[1] Since Part1 and Part2 of the OID are different, no confusion may occur with the previous edge named *L1

3.3. Network hierarchy

This hierarchy is similar to the hierarchy of nodes. The rules of construction are the same. Nevertheless, the notion of shadow network does not exist since the hierarchy is completely defined by the nodes or the edges. The local OID of a network is divided into two parts: the node (or the edge) part and the order part. The node (resp. edge) part represents the local OID of the node (resp. edge) depending on the level, the order part allows to distinguish between several networks associated to the same node (this order is always equal to 1 for the Associated_network of a Master_edge).

3.4. From hierarchies to database

The hierarchy of nodes and the hierarchy of edges define the relationships between the data. These relationships are handled by the Graph Data Model. They are totally independent from the data model managing the alphanumerical data associated to the nodes or to the edges. Therefore no relationships are required at the alphanumerical level (see figure 1). The correspondence between the Graph Data Model and the hierarchies is defined by the following steps:

Step 1: A node of the first level of abstraction is an object of the class Node

Step 2: A node with a level of abstraction greater than 1 (i.e., level i) is an object of the class Master_node. For each group of depending nodes (level i-1), an Associated_network is created (providing the nodes of the Associated_network).

Step 3: An edge of the first level of abstraction is an object of the class Edge.

Step 4: An edge with a level of abstraction greater than 1 (i.e., level i) is an object of the class Master_edge. An Associated_network is created with the depending edges (providing the nodes and the edges of the associate network).

Step 5: For each group of depending nodes, the edges are added to the corresponding Associated_network.

Step 6: For each Associated_network, the In_edges and the Out_edges are computed.

The interface of the Graph Data Model with an application is based on three classes Object_node, Object_edge, Object_network:

```
Class Object_node
   inherit Master_node
   private
   type tuple
     (
       base_node: Node
     )
end
```

```
Class Object_edge
   inherit Master_edge
   private
   type tuple
     (
     base_edge: Edge
     )
end
```

```
Class Object_network
   inherit Associated_network
   private
   type tuple
     (
     base_network: Network
     )
end
```

Since these classes inherit from the Graph Data Model the structure of OID, several applications may share the database components and define a new logical OID. Depending on the semantics of the application, a user defined hierarchy may be built: from an abstraction hierarchy to a semantic hierarchy. The following restructurings are allowed:

- A Master_node (resp. Master_edge) may be changed into a node (resp. an edge)
- A (Master_)node (resp. (Master_)edge) may be recursively be changed to a higher level[2] but only if it is depending of a shadow node (resp. edge)
- A (Master_)node (resp. (Master_)edge) may be erased from a node (resp. edge) hierarchy

The classes allow an user defined graph. The attributes provided by the inheritance mechanism define the user's point of view. The base_node (resp. base_edge, base_network) represents the original node (resp. edge, network) stored in the database.

4. Path operator

The basic operators defined on the Graph Data Model are: the union, the intersection and the difference. They represent the core operators of the data model. They must be combined to define more elaborated operators. One of the main important elaborated operator in the network-oriented queries is the path operator. In this part we present two original ways of considering the path operator: the Open Graph Assumption and the notion of direction.

[2] This is the reason a local OID must be unique in the database otherwise conflicts may occur.

4.1. Path 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 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. (Boursier and Mainguenaud 1992) 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 aeroplane 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). Figure 18 presents individual solutions respecting an aggregate constraint such as the cost is less than 1000 units and the "result graph" defined by the union of the individual solutions. One may conclude that the thin path is relevant for the query (the non-realistic alternative is to manually re-compute the cost!).

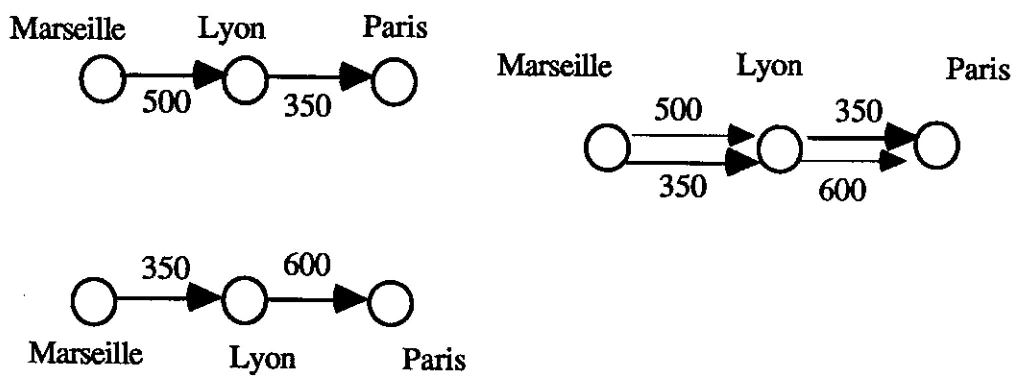

Figure 18. Individual solutions

To provide an answer, the path operator must have found a complete path from the starting node to the end node. Next section presents an alternative to this requirement.

4.2. Open Graph Assumption

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 starting node to the ending 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." 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 following example illustrates this notion. Let us suppose a network modelling a transportation network (figure 19).

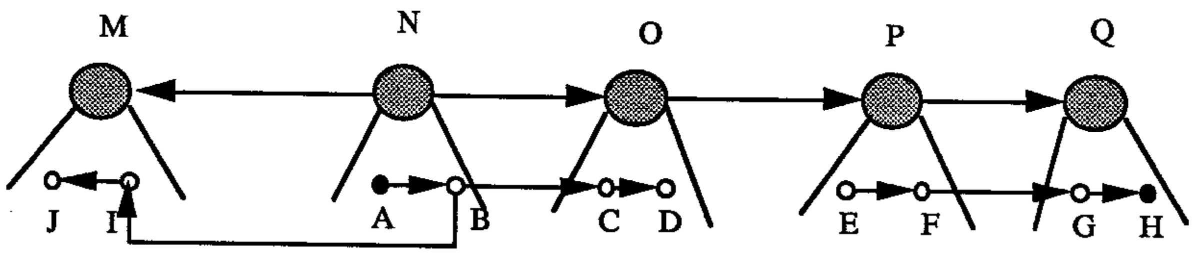

Figure 19. A toy example

To evaluate a path from A to H, the first level of abstraction provides a path (N-O-P-Q). The second level of abstraction provides the following path (figure 20):

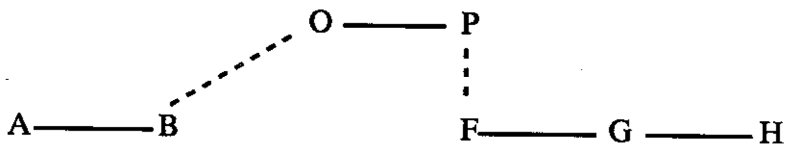

Figure 20. A path

The link between B and O is provided by the first level solution since O follows N. So the edge (B, I) is rejected. The link between P and F is provided by the first level solution and the Out_edges since the edge (F, G) is the relevant edge.

4.3. Notion of direction

While evaluating a path in a GIS context, two components are important: the next step and the global direction. Most part of the path algorithms applies the same recursive technique: (a) define the next step and (b) apply the path algorithm from this new step to the same ending node. Therefore, the next step is available, but the global direction is not defined.

The hierarchy of edges defines the global direction as an inherent part of the Graph Data Model. No restrictions are applied on the node level in an Associated_network of an edge. Two cases may appear while evaluating a path in a graph G (N, E) at a given level of abstraction: (1) the initial node and the end node belong to N and (2) one of them (at least) does not belong to N. In the first case, the evaluation of the path with the Open Graph Assumption is performed. In the second case, the hierarchy of edges allows to define the global direction so that the generalized edges of the initial and end nodes belong to E. The path can now be evaluated with the Open Graph Assumption.

The following example illustrates this notion. Let figure 21 be the basic graph, figure 22 be part of the hierarchy of edges.

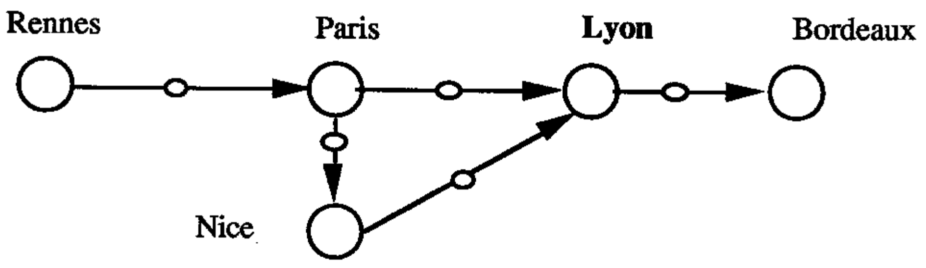

Figure 21. A toy graph

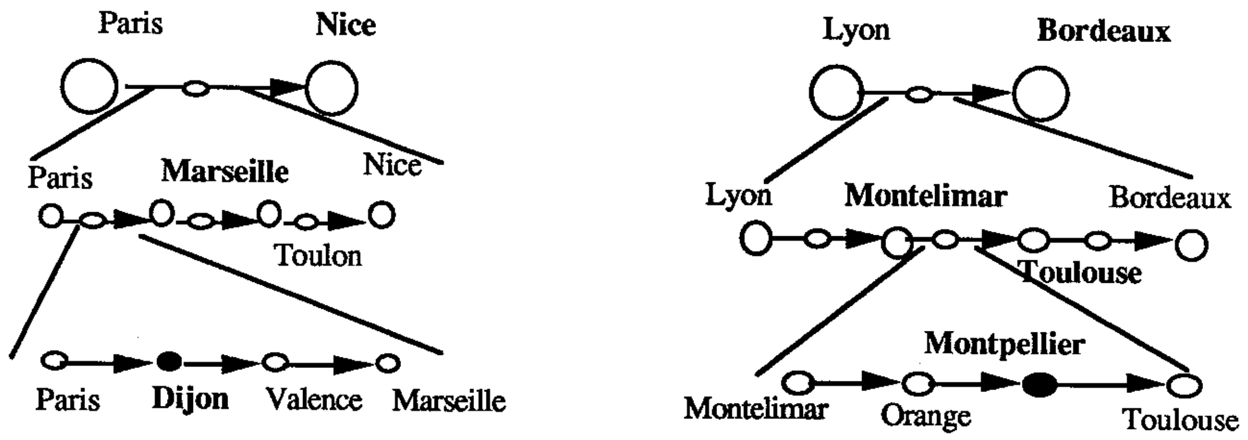

Figure 22. Edge hierarchy

To evaluate a path from Dijon to Montpellier, the hierarchy of edges defines (Paris, Nice) as the generalized edge for Dijon and (Lyon, Bordeaux) as the generalized edge for Montpellier. From now, the evaluation of the path is possible. The global directions are first to Marseille, then to Nice, Lyon Bordeaux, Montelimar and by the end Toulouse.

5. Implementation

The Graph Data Model presented in part 2 and part 3 is based on the object-oriented concepts. The choice of an Object Oriented Data Base Management System (OODBMS) to valid the data model is therefore natural. Within the two main philosophies of OODBMS: (1) extending a programming language with persistance (i.e., C++ with persistence) or (2) extending the data base model with the object-oriented constructors (i.e., Extended Relational DBMS), we choose the second one. This choice is only based on practical reasons: the O2 DBMS (Bancilhon et al 1991) was available in our programming environment. This part presents the toy application we develop with O2.

5.1. Data model component

To simulate two applications running on the same data base, we define the TOWN application and the SITE application. The towns are linked with plane connections and the sites are linked with train connections. Two alternatives may be defined to model an application: (1) the integrated approach such as the Graph Data Model is "inside" an application or (2) the separated approach such as the Graph Data Model is "outside" the application.

The integrated approach considers the Graph Data Model as the basic component of the application. Figure 23 presents the class hierarchy of such a modelling. A class of an application is a sub-class of Object_node (resp. Object_edge) instead of being a sub-class of Object.

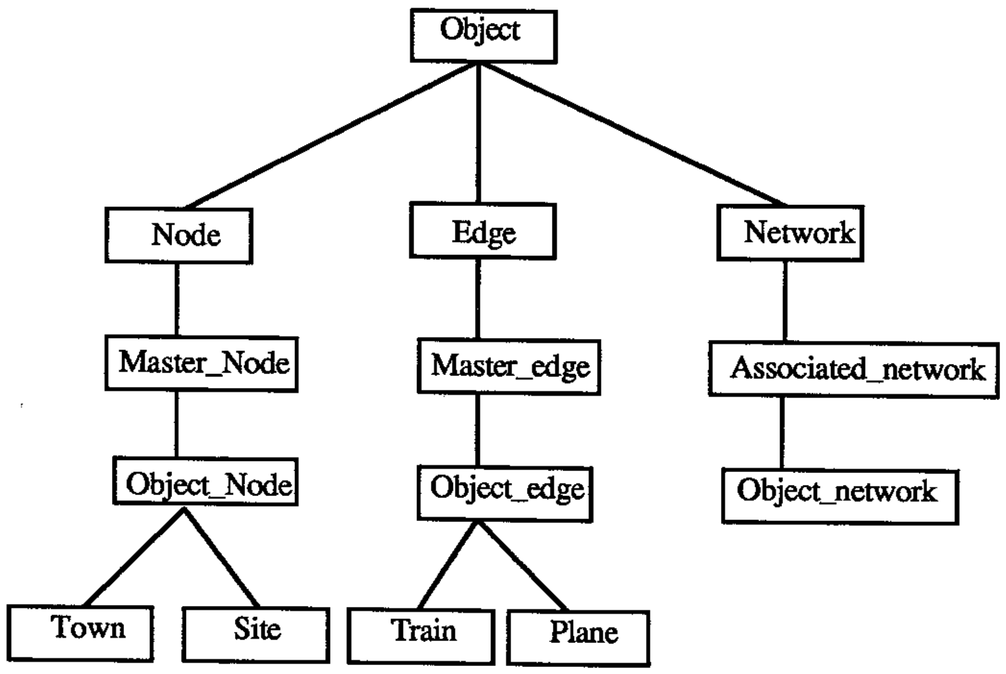

Figure 23. Class hierarchy

The separated approach considers the Graph Data Model as an element "outside" the application. This philosophy is similar to the abstract data types. An "attribute" (i.e., base_node) represents the network component as an attribute may represent the spatial representation (without knowing the physical representation). The declaration of the Town class with the O2 formalism is:

```
class Town
      private
      type tuple
         (
            population: integer,
            base_node:  Object_node
         )
end
```

The integrated approach offers several advantages such as (1) the high level data manipulation operators since the methods are applied directly to the object, (2) the data model "attributes" are hidden from the user's point of view due to the inheritance mechanism, (3) the multiple inheritance provides a common definition mechanism for the spatial representation data part and the Graph Data Model part of an object and (4) the strong and weak typing can be handled since the operators are defined on the same classes (i.e., Object_node). For these reasons we chose the integrated approach.

5.2. Data base component

Our goal was not to valid the ability for a OODBMS to handle a huge amount of data. Therefore the data volume is very low. The toy database is not designed in the context of a real application but to present different configurations to test the different operators. As an example figure 24 and figure 25 present the several levels of abstraction associated to the node France for the two applications (SITE and TOWN) in the data base. One can notice for example the change Master_node / Node for GDL or the change of level from the hierarchy to the database for Dijon (i.e., this configuration allows to test the union operator between the two nodes France). To simplify the database implementation and the database model operators, the nodes (resp. the edges) are defined as Master_nodes (resp. Master_edges).

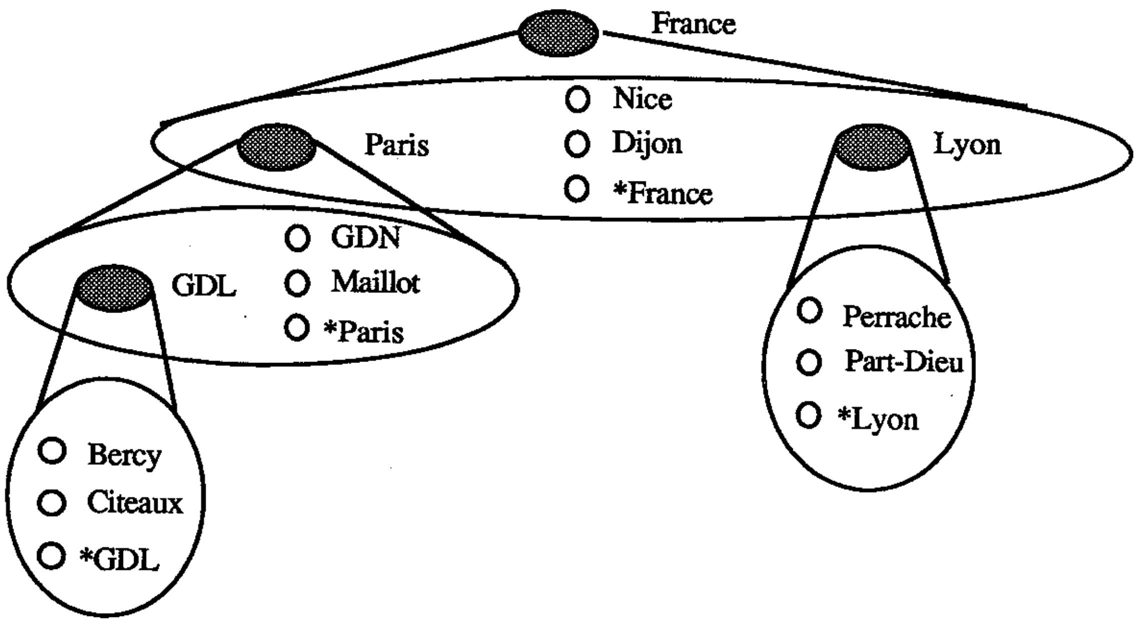

Figure 24. Node France in the application SITE

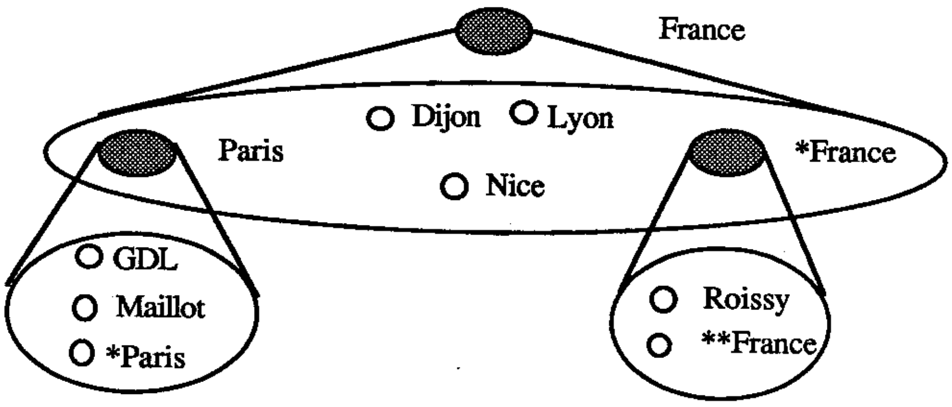

Figure 25. Node France in application TOWN

5.3. Learning of this experiment

Providing several levels of data definition is of prime importance in the context of a huge database (i.e., the generalization problem). Coupling thematic-oriented data, network-oriented data with a geometrical data model and the Graph Data Model with the same formalism allows to manage multi-sources data and to define a unified framework. We chose an object-oriented technology but an Extended Relational technology should have provided the same facilities.
The main advantage is provided by the notion of abstraction (to model the Master_node or the Master_edge) inherent to the data model. The second advantage is based on the local research area

that should improve the path evaluation. The third advantage is to change the conception of evaluating a path operator by providing "an Open Graph Assumption" since several levels of detail in the solution can be offered. The fourth advantage, independent from the data model, is provided by the flexibility of the object oriented paradigm (i.e., developing code is easier than with a classical programming language).

The main drawback is the "pre-compiled" structure of the network. Allowing dynamics or the re-structuring of the database is complex (i.e., the re-structuring of a classical database). The following example illustrates this weakness: Let us suppose the database is organized by countries such as France or Belgium. The definition of local areas near the frontier as a Master_node is more complex since the nodes depend from France or Belgium. Nevertheless, the definition of a Master_edge as an abstraction of a sub-network decreases this drawback.

6. Conclusion

Geographical Information Systems (GIS) 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 in the data base system. This weakness (in case of a multi-users configuration with updates) is due to the lack of common data structures between the thematic-oriented data (i.e., town, forest) and the network-oriented data (i.e., railways, water).

In this paper we present a Graph Data Model. The data model is based on the merge of graph theory concepts and the object-oriented paradigm. The introduction of the Master_nodes (resp. Master_edges) allows to define a node (resp. an edge) as an abstraction of a sub-network. A network can be defined using several levels of definition (detail) following the importance of the nodes (resp. the edges) within an application. The object-oriented concepts are applied to the alphanumerical parts of the data and to model the topology of the graph.

We introduce an alternative way of considering the path operator. The definition of the Open Graph Assumption, allows to define a path with the several levels of details. The notion of direction within a level of abstraction is inherent to the data model.

This model is well adapted to take into account the multi-levels networks (i.e., the definition of a view in the context of multi-users configuration or the introduction of the generalization). A toy application is defined with the O2 Object Oriented Data Base Management System. The next work is to finish the coupling of this data model and its basic operators with the path operator of the Cigales query language (Calcinelli and Mainguenaud 1991, 1994).

References

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

Banejeree J., Chou HT, Garza J., Kim W., Woelk D., Ballou N., Kim HJ, 1987, Data Model Issues for Object-oriented Applications, ACM Transactions On Information Systems, Vol 5, January, 3-26

Becker RA, Eick SG, Miller EO, Wilks AR, 1990, Network Visualization, 4th International Symposium on Spatial Data Handling (Colombus, OH:International Geographical Union), Zurich, Switzerland, 23-27 July, 285-294

Bennis K., David B., Quilio I., Viémont Y., 1990, GéoTropics: Database Support Alternatives for Cartographic Applications, 4th International Symposium on Spatial Data Handling (Colombus, OH:International Geographical Union), Zurich, Switzerland, 23-27 July, 599-610

Borgida A, Myropoulos J, Wong HK, 1984, 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, New York

Bratko I., 1987, Prolog programming for Artificial Intelligence, International Computer Science Series, Addison-Wesley

Boursier P., Mainguenaud M., 1992, Spatial Query Languages: Extended SQL vs. Visual Languages vs. Hypermaps, 5th International Symposium on Spatial Data Handling (Columbia: International Geographical Union), Charleston, SC, USA, 3-7 August, 249-259

Calcinelli D., Mainguenaud M., 1991, The Management of the Ambiguities in a Graphical Query Language for GIS, 2nd Symposium on Large Spatial Databases, Zurich, Switzerland, August, Lecture Notes in Computer Science n° 525, 99-118

Calcinelli D., Mainguenaud M., 1994, Cigales : A Visual Query Language for Geographical Information System: The User Interface, Journal of Visual Languages and Computing (to appear)

Cruz IF, Mendelzon AO, Wood PT, 1987, A Graphical Query Language Supporting Recursion, Proceedings of ACM SIGMOD Conference, San-Fransisco, USA, May, 323-330

David B., Raynal L., Schorter G, Mansart V., 1993, GeO2: Why objects in a geographical DBMS ?, Advances in Spatial Databases, D. Abel, BC Ooi Eds, Lecture Notes in Computer Science n° 692, 264-275

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

Güting RH, 1991, Extending a Spatial Database System by Graphs and Object Class Hierarchies, International Workshop on Database Management System for Geographical Applications, Capri, Italy, May, Report n° 104 FernUniversität

Hammer H., Mc Leod D., 1981, Database description with SDM: A semantical Data Model, ACM Transactions On Database Systems, Vol 6, n° 3, 351-386

Haas LM, Cody WF, 1991 Exploiting Extensible DBMS in Integrated Geographical Information Systems, 2nd Symposium on Large Spatial Databases, Zurich, Switzerland, August, Lecture Notes in Computer Science n° 525, 423-450

Heres L. et al : GDF, a Proposed Standard for Digital Road Maps to be Used in Car Navigation Systems, Technical paper, Philips CE/Cis-Lab, Eindhoven, 1993

Hull R, King R, 1987, Semantic Database Modelling: Surveys, Applications and Research Issues, ACM Computing Surveys, Vol 19, 201-260

Kunii H, 1983, Graph Data Language: A High Level Access-Path Oriented Language, Ph D Dissertation, University of Texas at Austin, May

Mainguenaud M, Simatic XT, 1991, A Data Model to Deal with Multi-scaled Networks, I.G.U.-U.G.I. Conference on Multi-scales Multi-uses, Brno - Czechoslovakia 22-25 April, Computer Environment and Urban System, Vol 16, n°4, 281-288

Manola F., Orenstein JA, 1986, Toward a General Spatial Data Model for an Object-oriented Database Management System, Proceedings of the 12th Very Large Data Base Conference, Kyoto, Japan, August, 328-335

Peckham J, Maryanski F, 1988, Semantic Data Models, ACM Computing Surveys, Vol 20, n°3, 153-189

Scholl M., Voisard A., 1989, Thematic Map Modelling, 1st International Symposium on Large Spatial Databases, Santa-Barbara, USA, July, Lecture Notes in Computer Science n° 409

Smith JM, Smith DC, 1987, Database Abstraction: Aggregation and Generalization, ACM Transaction On Database System, Vol 2, n°2, 105-133

Smith TR, Menon S., Star JL, Ester JE, 1987, Requirements and Principles for the Implementation and Construction of Large Scale GIS, International Journal of Geographical Information Systems., vol 1, n°1, 13-31

Van Eck JR, De Jong T., 1990, Adapting Datastructures and Algorithms for Faster Transport Network Computations, 4th Spatial Data Handling Symposium (Colombus, OH:International Geographical Union), Zurich, Switzerland, July 24-27, 295-305

Vijlbrief T, Van Oosterom P., 1992, The Geo++ System: an Extensible GIS, 5th International Symposium on Spatial Data Handling (Columbia: International Geographical Union), Charleston, SC, USA, 3-7 August, 40-50

Zdonic S., Maier D. (Eds), 1989, Readings in Object-Oriented Database System, San Mateo, CA:Morgan Kauffmann