Graph Manipulations for the Network Oriented Management : Application to a Telecommunication Network

M. Mainguenaud, J.L. Raffy, X.T. Simatic

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 : MAINGUENAUD@FRINT51.bitnet, RAFFY@FRINT51.bitnet, SIMATIC@FRINT51.bitnet

Abstract

This paper presents the manipulation operators defined for a data model allying the concepts of the graph theory and the object oriented concepts to manage multi-scaled networks. The main manipulations are the union, the weak and strong intersection, the difference and the graph traversal. The distinction between the weak and strong operators allows distinguishing the operations dealing with the structure of the graph and the operations dealing with the specific data since a basic data element (node, edge) may have several levels of definition (detail) within an application. We give as an example the network manipulations based on the new french telecommunication network.

1. INTRODUCTION

Transport networks (i.e. train, plane, water, telecommunication) may be modeled with the concepts of the graph theory. Early 1970, substantial work was done on the optimization of the graph operators. The underlying data model was only composed of nodes and edges. Graph theory applied to operations research represents an important theoretical base [13] for efficient data manipulation operators. Nevertheless the formalism of the data model has a lack of semantic power if we compare it to the semantic data models [8] or the object oriented data models [16].
The main advantage of a Relational Data Base Management System (RDBMS) is its ability to handle a huge amount of data but the data model [15] cannot represent the structured objects very easily. In order to improve the representative and expressive power several DBMS offer an object oriented data model. In fact object oriented concepts such as classification, aggregation, generalization/specialization can be applied to the alphanumerical information parts of the network but also to the topology of the network.

The aim of this paper is to present a processing model associated to the data model defined in [11]. This model allows to handle multi-scaled networks. As an example we use in this paper the new french telecommunication network. Section 2 presents a short recall of the data model; section 3 presents the concept of layers defined in this data model; sections 4 presents the operators; section 5 presents the application of these concepts to the management a telecommunication network and section 6 deals with the conclusion.

2. THE DATA MODEL

This section is a short recall of the data model. We assume the reader to be familiar with the concepts of the graph theory [3] and the object oriented paradigm [16]. A simple example introduces the basic concepts of the model : node, edge, network (similar to the notion of node, edge and graph in the graph theory), master-node, master-edge (which intuitively correspond to an abstraction of a sub-network). In this paper a network R is defined as R (N_R, E_R) where N_R is a set of nodes and E_R is a set of edges. Figure 4 represents the network with the maximum level of abstraction, TL_i are the nodes of this network. Figure 3 is the same network with more details (decomposition of the TL_i nodes). Figure 2 is the decomposition of the TL_3 node and figure 1 is the network with the maximum logical decomposition.

LNCS 91

An edge e_k is a couple of nodes (n_i, n_j) i.e. : (P_2, P_3), P_2 is the initial node, P_3 is the end node. A master-edge is an abstraction of a sub-graph i.e. (C_{25}, A_{22}). A master-node is an abstraction of a sub-graph i.e. figure 4 for the graph of the figure 3. The sub-graph is called an associated network. A simple node (resp. a simple edge) is a node (resp. an edge) with no associated network. The nodes of an associated network are called the specialized nodes, the master-node, their generalized node.

Each basic data element (node, edge, network, master-node, master-edge) is identified by a logical OID (Object IDentifier). To take into account of the various levels of abstraction this OID is stratified using the following rules :

(1) A node belongs to a hierarchy of node. Let N be the depth of this hierarchy. The OID has N layers (from 1, the lower level of abstraction, to N, the higher level of abstraction). Let n (a node) belongs to the layer k. Two cases may happend :

* n is located at the top level of the abstraction ($k = N$). The first layer has the local OID -an integer- and the other layers are null (equal to 0) (i.e. TL_1 figure 4).
* n is not located at the top level ($1 \leq k < N$). The ($N - k$) first layers contain each local OID -an integer- of the generalized nodes. The layer k contains the local OID. The ($k - 1$) last layers are null (i.e. τ_{11} figure 3).

Remark : Let n be a node of the layer k ($k \leq N - 2$). The generalized node of the node n may not belong to the layer k+1. Therefore a shadow node is created with an arbitrary local OID at the level k+1.

(2) The OID of an edge is divided into three parts. The first part (resp. the second) contains the OID of the initial node (resp. end node) of the edge. These OID are built following the rule defined in (1). With the same philosophy an edge belongs to a hierarchy of edges. The layer is defined by the layers of the initial and end nodes. The third part contains the local OID of an edge (since several edges with the same initial and end nodes may exist).

To manipulate the structures of the data model several functions are defined. The function S^i (e_j) returns the OID of the initial node of a given edge e_j. S^e (e_j), a similar function, returns the OID of the end node of a given edge e_j. The function Simple_node (n_i) is true whenever n_i is a simple node (resp. Simple_edge). The function Master_node (n_i) is true whenever n_i is a master-node (resp. Master_edge).

3. THE LEVELS OF DEFINITION

The databases were introduced in order to avoid the redundancy of data. The concept of view allows various users to work on the same database with different levels of details. The data model is well suited to introduce the concept of views. In the following we consider a network as a part of a unique database. This section presents the basic manipulations defined on the logical OID of an object and the comparison between the level of detail of two objects.

3.1 The basic manipulations on the logical OID

This sub-section presents several functions used in this paper to manipulate the nodes and the edges. Each of them is presented in an informal and formal ways :

S returns the local OID for a given layer :

$$S : \quad OID \times \mathbb{N} \quad \rightarrow \quad \mathbb{N}$$

#S returns the set of relevant layers of an OID (i.e. o_j) :

$$\#S : \quad OID \quad \rightarrow \quad \{i \,/\, i \in \mathbb{N}\}$$

$$\#S\ (o_j) = \{\ i \,/\, S\ (o_j, i) \neq 0\}$$

oid returns the OID of an object :

$$\text{oid} : \text{Object} \quad \text{->} \quad \text{OID}$$

id is true iff the two OID (i.e. o_i, o_j) are equal :

$$\text{id} : \text{OID} \times \text{OID} \quad \text{->} \quad \text{Boolean}$$
$$\text{id}(o_i, o_j) = \text{True} \iff \#S(o_i) = \#S(o_j) \quad \forall k \in \#S(o_i) \quad S(o_i, k) = S(o_j, k)$$

inc is true iff the non-null data part of the first OID (i.e. o_i) is a sub-part of the (non-null part of the second OID (i.e. o_j) :

$$\text{inc} : \text{OID} \times \text{OID} \quad \text{->} \quad \text{Boolean}$$
$$\text{inc}(o_i, o_j) = \text{True} \iff \#S(o_j) \supseteq \#S(o_i) \quad \forall k \in \#S(o_i) \quad S(o_i, k) = S(o_j, k)$$

trunc transforms the OID (i.e. o_i) into an OID (i.e. o_j) with the n last layers equal to null (the total number of layers is N) :

$$\text{trunc} : \text{OID} \times \mathbb{N} \quad \text{->} \quad \text{OID}$$
$$k \in \#S(o_i)$$
$$\text{trunc}(o_i, k) = o_j \,/ \quad \forall l_1 / l_1 < k \quad S(o_j, l_1) = 0$$
$$\forall l_2 / k \leq l_2 \leq N \quad S(o_j, l_2) = S(o_i, l_2)$$

$\omega+$ returns all the edges "leaving" a given node (i.e. n_i)

$$\omega+ : \text{OID} \quad \text{->} \quad \{\text{OID}\}$$
$$\omega+(\text{oid}(n_i)) = \{\text{oid}(e_j) / S^i(e_j) = \text{oid}(n_i)\}$$

$\omega-$ returns all the edges "arriving at" a given node (i.e. n_i) :

$$\omega- : \text{OID} \quad \text{->} \quad \{\text{OID}\}$$
$$\omega-(\text{oid}(n_i)) = \{\text{oid}(e_j) / S^e(e_j) = \text{oid}(n_i)\}$$

3.2 The level of an object

Three levels - IDentical, Look-Like and Hierarchical Look Like - are defined for an object - n for a node, $e_k : (n_i, n_j)$ for an edge -.
Two nodes (resp. edges) are identical ID_n (resp. ID_e) iff they have the same OID and they are simple nodes (resp. edges) :

$$ID_n : \text{Node} \times \text{Node} \quad \text{->} \quad \text{Boolean}$$
$$ID_n(n_i, n_j) = \text{True} \iff \text{id}(\text{oid}(n_i), \text{oid}(n_j)) = \text{True} \wedge \text{Simple\_node}(n_i) = \text{Simple\_node}(n_j) = \text{True}$$

$$ID_e : \text{Edge} \times \text{Edge} \quad \text{->} \quad \text{Boolean}$$
$$ID_e(e_i, e_j) = \text{True} \iff \text{id}(\text{oid}(e_i), \text{oid}(e_j)) = \text{True} \wedge \text{Simple\_edge}(e_i) = \text{Simple\_edge}(e_j) = \text{True}$$

Two nodes (resp. edges) are look-like LL_n (resp. LL_e) iff they have the same OID and they are not two simple nodes (resp. edges) :

$$LL_n : \text{Node} \times \text{Node} \quad \text{->} \quad \text{Boolean}$$
$$LL_n(n_i, n_j) = \text{True} \iff \text{id}(\text{oid}(n_i), \text{oid}(n_j)) = \text{True} \wedge \text{Simple\_node}(n_i) = \text{Simple\_node}(n_j) = \text{False}$$

LL_e : Edge x Edge -> Boolean

$LL(e_{k1} : (n_a, n_b), e_{k2} : (n_c, n_d)) = True \iff LL(n_a, n_c) \wedge LL(n_b, n_d) \wedge$
$e_{k1} : (n_a, n_b) \wedge e_{k2} : (n_c, n_d) \wedge$
$Simple\_edge(e_{k1}) = Simple\_edge(e_{k2}) = False$

Two nodes (resp. edges) are hierarchically look like HLL_n (resp. HLL_e) iff they have a common generalized node (which may be equal to one of these nodes).

HLL_n : Node x Node -> Boolean

$HLL_n(n_i, n_j) = True \iff \exists\, a, b \in \mathbb{N} /$
$trunc(oid(n_i), a) = tronc(oid(n_j), b)$[1]

HLL_e : Edge x Edge -> Boolean

$HLL_e(e_{k1} : (n_a, n_b), e_{k2}(n_c, n_d)) = True \iff HLL_n(n_a, n_c) \wedge HLL_n(n_b, n_d) \wedge$
$e_{k1}(n_a, n_b) \wedge e_{k2} : (n_c, n_d)$

Let the figures 1 and 3 be two views of the same initial network. The node P_3 and Π_3 are identical, τ_{31} and T_{31} are look like, C_{32} and C_{34} are hierarchically look like.

Ass_n returns the nth associated network of a given master-node :
Ass_n : Node x $\mathbb{N}$ -> {Node} x {Edge}

Ass_e returns the associated network of a given master-edge :
Ass_e : Edge -> {Node} x {Edge}

4. THE DATA MANIPULATIONS

Let R (N_R^0, E_R^0), S (N_S^0, E_S^0) and T (N_T, E_T) be three networks. The operators are defined by :

$T(N_T, E_T) = \langle operator\rangle (R(N_R^0, E_R^0), S(N_S^0, E_S^0))$

A naïve general case evaluation of an operator follows the step (1) to (4) :

(1) $k \leftarrow 0$, $T^k \leftarrow R$
(2) The construction of S'
    $E_S' = \{e_i\}$ e_i is an arbitrary edge chosen in E_S^k
    $N_S' = \{n_j / oid(n_j) = S^i(e_i)\} \cup \{n_j / oid(n_j) = S^e(e_i)\}$
(3) The application of the basic rules of the operator with $T^k(N_T^k, E_T^k)$ and $S'(N_S', E_S')$
    (for the nodes of N_S' which have not already been studied then for the edge)
(4) The recursive application of the step (2) to (4) until $E_S^k = \varnothing$ with
    $E_S^{k+1} = E_S^k - \{e_i\}$
    $N_S^{k+1} = N_S^k - \{n_i \in N_S^k / \omega+(oid(n_i)) = \varnothing \wedge \omega-(oid(n_i)) = \varnothing\}$

The basic manipulations are : the union, the intersection (weak and strong), the difference. The distinction between the weak and strong operators allows distinguishing the operations dealing with the structure of the graph and the operations dealing with specific data (or a specific level of detail i.e. ID_n, ID_e, LL_n, LL_e functions). This section presents the basic manipulations but do not detail the HLL level.

[1]If a (resp. b) is null then n_i (resp. n_j) is a generalized node of n_j (resp. n_i)

4.1 The union

The union of two independent networks is defined by :

$$\forall n_i \in N_R, \forall n_j \in N_S$$
$$ID_n(n_i, n_j) = False \wedge LL_n(n_i, n_j) = False \wedge HLL_n(n_i, n_j) = False \rightarrow$$
$$\cup(R, S) = T(N_T, E_T) \,/\, N_T = \{n_k\} \wedge E_T = \emptyset$$

such as n_k is a master-node of level N+1. Two associated networks are defined for this node : R and S

General formulation :

The general formulation of the union operator is :

$$T^{k+1}(N_T^{k+1}, E_T^{k+1}) = \cup(T^k(N_T^k, E_T^k), S'(N_S', E_S'))$$

Case study :

Let $n_i \in N_T^k$, $n_j \in N_S'$, $e_i \in E_T^k$, $e_j \in E_S'$:

$ID_n(n_i, n_j) = True \quad \rightarrow \quad N_T^{k+1} = N_T^k$

$LL_n(n_i, n_j) = True$

Simple_node (n_i) = Master_node (n_j) = True $\rightarrow$ $N_T^{k+1} = N_T^k - \{n_i\} \cup \{n_j\}$

Master_node (n_i) = Simple_node (n_j) = True $\rightarrow$ $N_T^{k+1} = N_T^k$

Master_node (n_i) = Master_node (n_j) = True $\rightarrow$ $N_T^{k+1} = N_T^k$

Let d_i (resp. d_j) be the number of the associated networks of n_i (resp. n_j) obtained by $Ass_n(n_i, k)$ $(k = 1, ..., d_i)$ (resp. $Ass_n(n_j, m)$ $(m = 1, ..., d_j)$). (N_{dki}, E_{dki}) (resp. (N_{dmj}, E_{dmj})) is one of the associated networks of n_i (resp. n_j) . The basic treatment of (N_{dmj}, E_{dmj}) is

$\forall m \in \{1, ..., d_j\} \,/\, \forall k \in \{1, ..., d_i\} \;\; n_a \in N_{dki}, \; n_b \in N_{dmj}$

$ID_n(n_a, n_b) = False \wedge LL_n(n_a, n_b) = False \wedge HLL_n(n_a, n_b) = False \rightarrow$

n_i has a supplementary associated networks (N_{dmj}, E_{dmj})

$\forall m \in \{1, ..., d_j\} \,/\, \exists k \in \{1, ..., d_i\} \; \exists n_a \in N_{dki}, \exists n_b \in N_{dmj}$

$ID_n(n_a, n_b) = True \vee LL_n(n_a, n_b) = True \vee HLL_n(n_a, n_b) = True \rightarrow$

A recursive application of the union operator is performed on the associated networks (N_{dki}, E_{dki}), (N_{dmj}, E_{dmj}). A recursive application of the union operation may be performed on two associated networks $A_1(N_{dti}, E_{dti})$, $A_2(N_{dsi}, E_{dsi})$ (such as $A_1 \neq A_2$) if :

$\exists n_1 \in N_{dti} \,/\, ID_n(n_a, n_1) \vee LL_n(n_a, n_1) \vee (HLL_n(n_a, n_1) \wedge$

$\exists n_2 \in N_{dsi} \,/\, ID_n(n_b, n_2) \vee LL_n(n_b, n_2) \vee HLL_n(n_b, n_2)$

This operation leads to the withdrawal of A_2 as an associated network of n_i after the recursive application of the union operator between A_1 and A_2.

$\forall$ $n_i \in N_T^k$ / $ID_n(n_i, n_j) = \text{False} \wedge LL_n(n_i, n_j) = \text{False} \wedge HLL_n(n_i, n_j) = \text{False}$ ->
$N_T^{k+1} = N_T^k \cup \{n_j\}$

$ID_e(e_i, e_j) = \text{True}$ -> $E_T^{k+1} = E_T^k$

$LL_e(e_i, e_j) = \text{True}$

Simple_edge (e_i) = Master_edge (e_j) = True -> $E_T^{k+1} = E_T^k - \{e_i\} \cup \{e_j\}$

A recursive application of the union operator is performed with the associated network of e_j - $Ass_e(e_j)$- and the network defined by :
$(\{n_j / oid(n_j) = S^i(e_i) \vee oid(n_j) = S^e(e_i)\}, \{e_i\})$

Master_edge (e_i) = Simple_edge (e_j) = True -> $E_T^{k+1} = E_T^k$

A recursive application of the union operator is performed with the associated network of e_i - $Ass_e(e_i)$-and the network defined by :
$(\{n_i / oid(n_i) = S^i(e_j) \vee oid(n_i) = S^e(e_j)\}, \{e_j\})$

Master_edge (e_i) = Master_edge (e_j) = True -> $E_T^{k+1} = E_T^k$

A recursive application of the union operator is performed on the associated networks since the initial node and the end node belong to the two associated networks : $\cup(Ass_e(e_i), Ass_e(e_j))$

$\forall$ $e_i \in E_T^k$ $ID_e(e_i, e_j) = \text{False} \wedge LL_e(e_i, e_j) = \text{False} \wedge HLL_e(e_i, e_j) = \text{False}$ ->
$E_T^{k+1} = E_T^k \cup \{e_j\}$

4.2 The intersection

The intersection of two independent networks is defined by :

$$\forall n_i \in N_R, \forall n_j \in N_S$$
$$ID_n(n_i, n_j) = \text{False} \wedge LL_n(n_i, n_j) = \text{False} \wedge HLL_n(n_i, n_j) = \text{False} \rightarrow$$
$$\cap(R, S) = T(N_T, E_T) / N_T = \varnothing \wedge E_T = \varnothing$$

General formulation :

The general formulation of the intersection operator is

$$T^{k+1}(N_T^{k+1}, E_T^{k+1}) = \cap(T^k(N_T^k, E_T^k), S'(N_S', E_S'))$$

First Step :

$\forall$ $n_i \in N_T^k$ /$\forall$ $n_j \in N_S'$:

$ID_n(n_i, n_j) = \text{False} \wedge LL_n(n_i, n_j) = \text{False} \wedge HLL_n(n_i, n_j) = \text{False}$ ->
$N_T^{k+1} = N_T^k - \{n_i\}$
$E_T^{k+1} = E_T^k - \{e_k / S^i(e_k) = oid(n_i)\} - \{e_k / S^e(e_k) = oid(n_i)\}$

Case study :

Let $n_i \in N_T^k$, $n_j \in N_S'$, $e_i \in E_T^k$, $e_j \in E_S'$:

Strong intersection :

$ID_n(n_i, n_j)$ = True -> $N_T^{k+1} = N_T^k$

$LL_n(n_i, n_j)$ = True

Simple_node (n_i) = Master_node (n_j) = True -> $N_T^{k+1} = N_T^k - \{n_i\}$
$E_T^{k+1} = E_T^k - \{e_k / S^i(e_k) = oid(n_i)\} - \{e_k / S^e(e_k) = oid(n_i)\}$

Master_node (n_i) = Simple_node (n_j) = True -> $N_T^{k+1} = N_T^k - \{n_i\}$
$E_T^{k+1} = E_T^k - \{e_k / S^i(e_k) = oid(n_i)\} - \{e_k / S^e(e_k) = oid(n_i)\}$

Master_node (n_i) = Master_node (n_j) = True -> $N_T^{k+1} = N_T^k$

A recursive application of the intersection operator is performed on the associated networks of n_i and n_j. If all the intersections are empty, n_i becomes a simple node else the associated networks are the results of the strong intersection operator.

$ID_e(e_i, e_j)$ = True -> $E_T^{k+1} = E_T^k$

$LL_e(e_i, e_j)$ = True

Simple_edge (e_i) = Master_edge (e_j) = True -> $E_T^{k+1} = E_T^k - \{e_i\}$

Master_edge (e_i) = Simple_edge (e_j) = True -> $E_T^{k+1} = E_T^k - \{e_i\}$

Master_edge (e_i) = Master_edge (e_j) = True -> $E_T^{k+1} = E_T^k$

A recursive application of the strong intersection operator is performed on the associated networks of e_i and e_j. If the set of edges is empty e_i becomes a simple edge else the associated network is the result of the strong intersection operator.

Weak intersection :

$ID_n(n_i, n_j)$ = True -> $N_T^{k+1} = N_T^k$

$LL_n(n_i, n_j)$ = True

Simple_node (n_i) = Master_node (n_j) = True -> $N_T^{k+1} = N_T^k$
Master_node (n_i) = Simple_node (n_j) = True -> $N_T^{k+1} = N_T^k - \{n_i\} \cup \{n_j\}$
Master_node (n_i) = Master_node (n_j) = True -> $N_T^{k+1} = N_T^k$

A recursive application of the weak intersection operator is performed on the associated networks. If all the intersection are empty, n_i becomes a simple node
else the associated networks are the results of the weak intersection operator.

$ID_e(e_i, e_j) = True \quad -> \quad E_T^{k+1} = E_T^k$

$LL_e(e_i, e_j) = True$

$Simple\_edge(e_i) = Master\_edge(e_j) = True \quad -> \quad E_T^{k+1} = E_T^k$
$Master\_edge(e_i) = Simple\_edge(e_j) = True \quad -> \quad E_T^{k+1} = E_T^k - \{e_i\} \cup \{e_j\}$
$Master\_edge(e_i) = Master\_edge(e_j) = True \quad -> \quad E_T^{k+1} = E_T^k$

A recursive application of the weak intersection operator is performed on the associated networks. If the set of edges is empty, e_i becomes a simple edge else the associated network is the result of the weak intersection operator.

4.3 The difference

The difference of two independent networks :

$$\forall n_i \in N_R, \forall n_j \in N_S$$
$$ID_n(n_i, n_j) = False \wedge LL_n(n_i, n_j) = False \wedge HLL_n(n_i, n_j) = False ->$$
$$-(R, S) = T(N_T, E_T) / N_T = N_R \wedge E_T = E_R$$

General formulation :

The general formulation of the difference operator is :

$$T^{k+1}(N_T^{k+1}, E_T^{k+1}) = -(T^k(N_T^k, E_T^k), S'(N_S', E_S'))$$

Case study :

Let $n_i \in N_T^k$, $n_j \in N_S'$, $e_i \in E_T^k$, $e_j \in E_S'$:

$ID_n(n_i, n_j) = True \quad -> \quad N_T^{k+1} = N_T^k - \{n_i\}$
$E_T^{k+1} = E_T^k - \{e_k / S^i(e_k) = oid(n_i)\} - \{e_k / S^e(e_k) = oid(n_i)\}$

$LL_n(n_i, n_j) = True \quad -> \quad N_T^{k+1} = N_T^k - \{n_i\}$
$E_T^{k+1} = E_T^k - \{e_k / S^i(e_k) = oid(n_i)\} - \{e_k / S^e(e_k) = oid(n_i)\}$

$ID_e(e_i, e_j) = True \quad -> \quad E_T^{k+1} = E_T^k - \{e_i\}$

$LL_e(e_i, e_j) = True \quad -> \quad E_T^{k+1} = E_T^k - \{e_i\}$

5. THE APPLICATION TO A TELECOMMUNICATION NETWORK :

This section presents in a first part the various queries addressed to a GIS for the management of a telecommunication network and the second part presents the resolution of such queries using the operators defined in the previous section.

5.1 The queries

The queries may be divided into two classes : (1) the queries about the alphanumerical data part of the networks and (2) the queries about the topology of the network.

About alphanumerical data part :

These data can be modeled using a classical data model. The queries defined on these data are fully application dependent. The ability to answer such queries depends on the representative power and the data manipulation language power of the DBMS in charge of managing the data. We do not consider this kind of queries in this paper.

About the topology of the network :

These queries are principally based on a graph traversal operator with or without constraints. Four queries may be considered as representative of the classical queries :
Query 1 : What are the paths joining C_{31} to C_{24} ?
Query 2 : What are the paths joining C_{32} to A_{21} without visiting specific nodes (i.e. with a bad quality of service, i.e. T_{32} and T_{21}) ?
Query 3 : What are the paths joining C_{25} to C_{21} without using specific edges (i.e. with a charge greater than 70 %, i.e. C_{25}-A_{22}-ED_{1}) ?
Query 4 : What are the common sub-paths between the paths C_{32}-C_{14} and C_{38}-C_{11} ?

5.2 The resolution

Evaluating a path in a graph between two nodes in the database context is equivalent to define a recursive query [1, 5]. Substential work has been done on this subject [6]. [10] presents the management of the basic queries with a flat data structure (relational-like structure) and [9] presents the management of the complex queries with a flat data structure with an Extended Relational DBMS. We do not detail [14] here the transitive closure algorithm since it involves several specific telecommunication routing rules. These specificities allow an evaluation with a mixture of the breath first and the depth first methods.

To solve query 1 to query 4, a combination of the operations presented in section 4 with the specific graph traversal operator is required.
Query 1 is solved using a graph traversal algorithm. Query 2 is solved with a recursive use of the union operator (breath first method) between the initial departure node and the arrival node. The stop is defined by the multi-level structuration of the OID. Each step of this recursive process guaranties by applying the difference operator the withdrawal of the specific nodes and the edges arriving at or leaving these nodes. Then the graph traversal operator can be applied. The resolution of the query 3 follows the same philosophy as the resolution of the query 2. Each step of the recursive process of the union operator guaranties by appling the difference operator the withdrawal of the specific edges. The resolution of the query 4 involves the strong intersection between the path C_{32}-C_{14} and the path C_{38}-C_{11}.

6. CONCLUSION

This paper presents the basic data manipulation operators defined on a logical multi-scaled network. The operators are the union, the weak and strong intersection, the difference and the graph traversal. The difference between the weak and strong operators allows the distinction between the operations dealing with the structure of the graph and the operations dealing with the specific data since a basic data element (node, edge) may have several levels of definition (detail) within an application. The answer of a query is obtained by combination of these operators. Queries about the management of a telecommunication network give an example of such a resolution.
We are currently implementing a toy application using an object oriented Data Base Management System O2 [2]. The next step is to support this data processing model in the context of the CIGALES prototype [12, 9] and compare the object oriented philosophy and the Extended Relational DBMS philosophy [7] to manage network oriented data.

REFERENCES

1. Aho A., Ullman J.D., Universality of Data Retrieval Languages, ACM Principles Of Programming Languages, San Antonio, USA, January 1979
2. Bancilhon F. and al : The design and Implementation of O2, an Object Oriented Database System, Proc. of the 2nd Int. Work. on Object-Oriented Data Base Systems, K. Dittrich (Ed) Bad-Munster, FRG, September 1988
3. Berge C. : Graphes, Gauthier-Villars, 1983
4. Codd E.F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol 13, n°6, 1970
5. Cruz I.F., Mendelzon A.O., Wood P.T., A Graphical Query Language supporting Recursion, ACM SIGMOD, San Fransisco, USA, May 27-29 1987
6. Eck (van) J.R., De Jong T. : Adapting Datastructures and Algorithms for Faster Transport Network Computations, 4th Spatial Data Handling Symp., Zurich, Switzerland, July 24-27 1990
7. Gardarin and all : Managing Complex Objects in an Extensible Relational DBMS, 15th Int. Conf. on Very Large Data Bases, Amsterdam, The Netherlands, August 22-25 1989
8. Hull R., King R. : Semantic Database Modelling : Surveys, Applications and Research Issues" ACM Computing Surveys, Vol 19, Sept 1987
9. Jourdas C., Mainguenaud M., An Extended relational DBMS to Manage Network Applications, European Geographical Information System 91, Brussels, Belgium, April 2-5 1991
10. Mainguenaud M., Is an Extended Relational DBMS Powerful Enough to Deal with Network Applications, European Geographical Information System 90, Amsterdam, The Netherlands, April 9-13 1990
11. Mainguenaud M., X.T. Simatic : A Data Model for Multi-Scaled Network, UGI-IGU Conference - GIS Multiple Representation and Multiple Use, Brno - Czechoslovakia, April 22-25 1991
12. Mainguenaud M., Portier M.A. : CIGALES : a Graphical Query Language for GIS, 4th Spatial Data Handling Symp., Zurich, Switzerland, July 24-27 1990
13. Nato Conference : The Application of Operations Research to Transport Problems, Sandefjord, Norway, 14-18 August 1972
14. Raffy J.L., Modélisation d'un Réseau de Télécommunication, Rapport de D.E.A. Université de Paris VI Jussieu, France, September 1990 (in french)
15. Ullman J. : Principles of Databases, Academic Press, 1980
16. Zdonic S., D. Maier (Eds), Readings in Object-Oriented Database System, Morgan Kauffmann, 1989

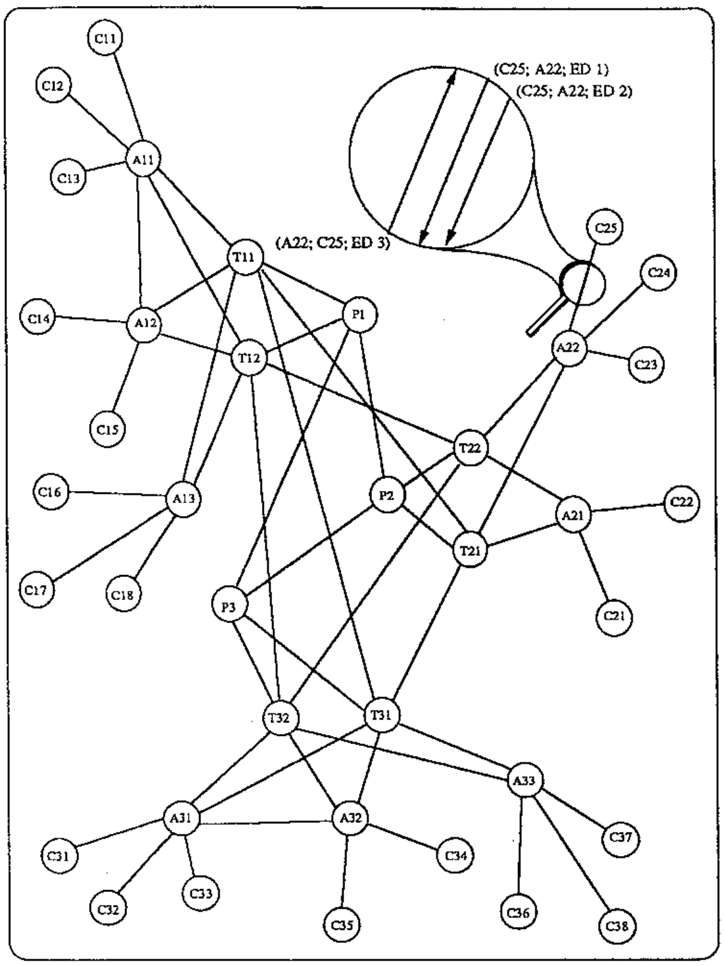

FIGURE 1

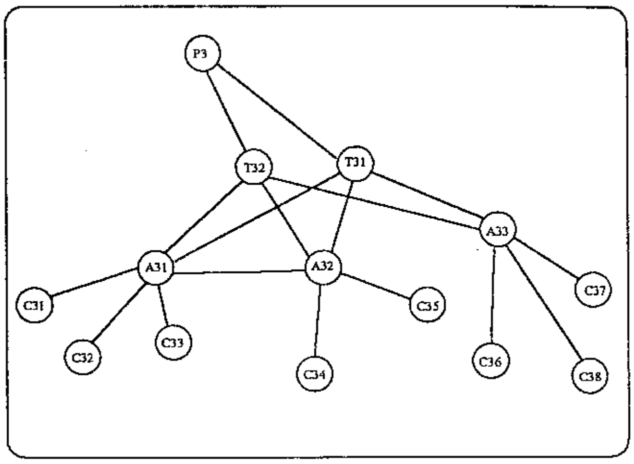

FIGURE 2

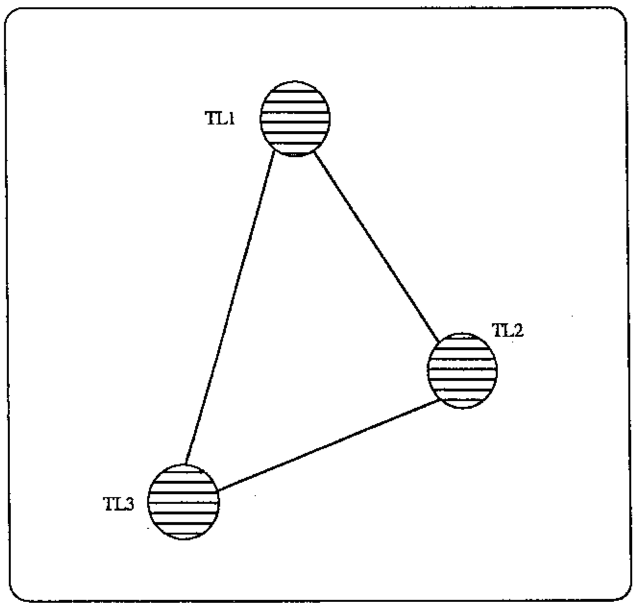

FIGURE 4

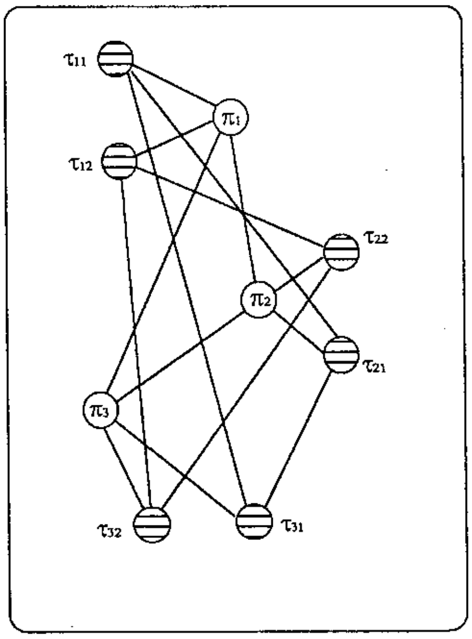

FIGURE 3