THE SEMANTICS OF THE GEOGRAPHICAL DATABASE QUERY LANGUAGES

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: maing@int-evry.fr

Abstract:

In this paper, we analyze the various philosophies to define a query language for geographical database. The differences among the three possibilities of extending the SQL language rely on the definition of the clause Select: (1) the clause Select does not accept a spatial operator and the Where clause accepts one (or several) predicate defined between attributes; (2) the clause Select accepts one (or several) spatial operator(s) with attributes as arguments; (3) the clause Select accepts one (or several) spatial operator(s) with attributes or other spatial operators as arguments. To guarantee a coherent result of a query, we present the requirements of these three philosophies.

Key words:
Geographical Information System, Spatial Database, Query Language, Extended SQL

I. INTRODUCTION

In the current research toward the design of more powerful tools for urban planning, remote sensing, ... different research groups are simultaneously concentrating their work on Geographical Information Systems (GIS). GIS needs are very well known [21]. Nevertheless, several problems are still open. In this paper, we focus on the design of a query language. Two levels can be defined while designing a query language: the user interface level and the database system level. The user interface level has already received several propositions [3,4,5,6,12,13,15,16,20,23]. Nevertheless many works have to be performed on this field. The database system level has also received several propositions [1,7,8,17,18,25]. The Extended SQL (ESQL) approach seems to be the more natural interface language to query a spatial database since the SQL query language is now widely accepted to query a relational database. In this paper, we analyze the various philosophies to define an ESQL language for GIS. The differences among the three possibilities of extending the SQL language rely on the definition of the clause Select: (1) the clause Select does not accept a spatial operator and the Where clause accepts one (or several) predicate defined between attributes; (2) the clause Select accepts one (or several) spatial operator(s) with attributes as arguments; (3) the clause Select accepts one (or several) spatial operator(s) with attributes or other spatial operators as arguments.

Part II presents the toy database and the queries used for this analysis; Part III, IV and V respectively present the three philosophies of extending the SQL language; Part VI presents the conclusion.

II. THE TOY DATABASE

Several database models have been proposed to handle geographic information. They are based on (1) an extended relational approach with abstract data types or a N1NF philosophy [18,19]; (2) the object-oriented paradigm [11]; (3) rules and facts [9] or (4) an algebraic approach [7].

Our goal is not to define a new data model. The relational database model [24] is now widely accepted. To simplify the presentation without loss of generality, the toy database is defined with a relational formalism extended with an Abstract Data Type [22] for the attribute defining the spatial representation (i.e., Spatial_representation). In the following, a tuple of such relations is called an object. Figure 1 presents the definition of the toy database (a sub-set of the database defined in [2]).

Forest	(<u>Name</u>, Spatial_representation)
Town	(<u>Name</u>, Population, Spatial_representation)
Road	(<u>Name</u>, Type, Spatial_representation)
Lake	(<u>Name</u>, Type, Spatial_representation)
Polluted_area	(<u>Name</u>, Spatial_representation)

Figure 1

To illustrate the different philosophies of Extended SQL (ESQL) languages, we define a set of queries. Each query requires a specific expressive power. Let Q1 be the following query: "Which towns have a forest part?". This query requires to be able to provide the information on the towns without taking into account in the result of the spatial components (i.e., the forest part and the non-forest part of a town). Let Q2 be the following query: "Which are the national roads crossing the non-forest part of a town with more than 100,000 inhabitants?". This query requires to be able to provide the result of a spatial operator (i.e., to evaluate the non-forest part of a town). This result is used as an argument of a second spatial operator (i.e., the intersection). Let Q3 be the following query: "What are the routes from Paris to Nice crossing the non-forest part of a town and adjacent for a while to a lake?". This query involves the management of two independent spatial operators (i.e., the intersection and the adjacency) linked to the same object (i.e., the routes from Paris to Nice). Let Q4 be the following query: "What are the national roads crossing polluted areas such as the total distance is less than 15 km?". This query involves the management of a spatial operator in the context of an overlapping space division [14].

In this paper, we do not consider the way of defining an end-user query but rather the management of a formal representation of an end-user query. Since the number of applications using a GIS is considerable, it is of prime importance to offer an application-independent and DBMS-independent query language. Figure 2 represents the three levels of a GIS.

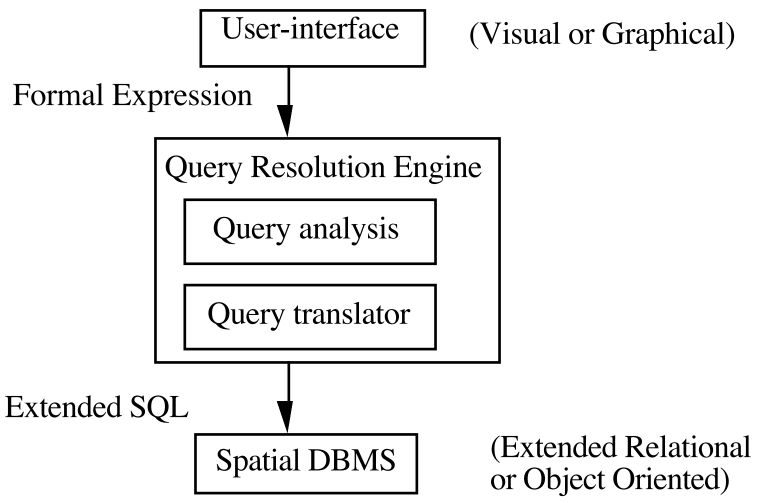


Figure 2

Since the design of the user interface is an iterative process, it is very important to separate the user interface from the query resolution engine. A formal query language is indispensable to model a graphical (or visual) end-user query. We adopt in this paper a (very intuitive) functional formalism (formally defined in [14]) used in the Cigales prototype [3] since no standard ESQL query language is available. This formalism allows being independent from the data model constraints of the spatial DBMS (i.e., the relational normal forms), from the data model organization (i.e., the modelling of the spatial representation as attributes or as an Abstract Data Type). Since the time of analysis of such an expression is very low in comparison with the time of the query resolution involving spatial operators, the query resolution process is not penalized by this organization. Furthermore the functional formalism is very well adapted to model geographical queries (SQL can be regarded as a functional language). The translator of the query resolution engine manages the link with the spatial DBMS and can make up for the possible weaknesses of the spatial DBMS query language.

III. THE SPATIAL RELATIONSHIPS AS PREDICATES

The first class of Extended SQL (ESQL) language is based on the introduction of predicates (i.e., intersection) in the Where clause. The Select clause is still defined with the attributes of the involved relations. The notion of predicate is always defined between attributes (i.e., Spatial_representation). Figure 3 presents the ESQL statement for the query Q_1.

```
SELECT    Town.*
FROM      Town, Forest
WHERE     intersection  (   Town.Spatial_representation,
                            Forest.Spatial_representation  )
```

Figure 3

This philosophy can be considered as equivalent to the generalization of the theta-join operator of the relational model [24] (with theta equals to a spatial operator). The relational DBMS are based on classical set-oriented logic programming. Spatial data cannot be handled with the classical predicates of the set-oriented logic programming out of the two dimensional structure. An example of this limitation is the query Q_2. The query Q_2 requires to define the evaluation of a spatial operator as a component of another spatial operator.

Following the logic-oriented structure, let T, F, R be three sets. Let ∩ be the set intersection. Let I (#□∅) be the intersection of the sets T and F. Figure 4 presents the logic implications for the thre sets.

$$\boxed{\begin{array}{rcl} x \in \cap\,(T, R) & \Rightarrow & x \in T \wedge x \in R \\ x \in \cap\,(T, R) \wedge x \notin \cap\,(I, R) & \Rightarrow & x \notin F \Rightarrow x \notin \cap\,(F, R) \end{array}}$$

Figure 4

As an interpretation T, F, R can be considered as the sets of identifier (i.e., the attribute Name) of the towns, the forests and the roads.

Following the spatial-oriented structure, let T (resp. F, R) be the spatial representation value of a tuple of the Town (resp. Forest, Road) relation. Let ∩g (T, F) be a Boolean predicate. ∩g (T, F) is true if and only if there exists a spatial intersection between T and F. Let I be the value representing the spatial intersection of T and F such as ∩g (T, F) = true. Figure 5 presents the logical links. Figure 6 is an example of such a spatial configuration.

$$\boxed{\cap_g\,(T, R) = \text{true} \wedge \cap_g\,(I, R) \neq \text{true} \quad \not\Rightarrow \cap_g\,(F, R) \neq \text{true}}$$

Figure 5

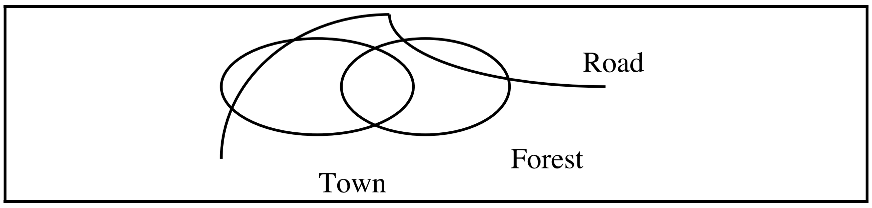


Figure 6

Conclusion :

The relational algebra is based on the set theory. The notion of excluded middle forbids all the queries involving the composition of spatial operator. Such ESQL query languages have a very weak expressive power. Therefore an ESQL query language must provide the result of the spatial operator in the clause Select.

IV. THE SPATIAL RELATIONSHIP AS ARGUMENTS

The second class of Extended SQL (ESQL) query language allows the spatial operators in the clause Select. The spatial operator must be defined between attributes (i.e., spatial_representation). The predicates introduced in the first class of ESQL languages are also supported by the second class. Therefore the figure 3 still represents the formalization of the query Q_1. This class of ESQL query language requires the closure of the spatial operators. The result of a spatial operator can be used as an argument of another spatial operator (as an attribute of a relation).

The queries Q_2 and Q_3 require the composition of operators (i.e., the intersection (∩) applied to the difference (Δ) or the adjacency (¤) and the intersection applied to the path operator (->)). The functional expression modelling a query is analysed to avoid the duplicates. Figure 7 and figure 8 present the Directed Acyclic Graph (DAG) modelling these queries.

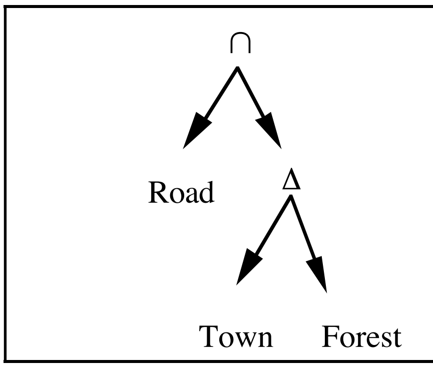


Figure 7

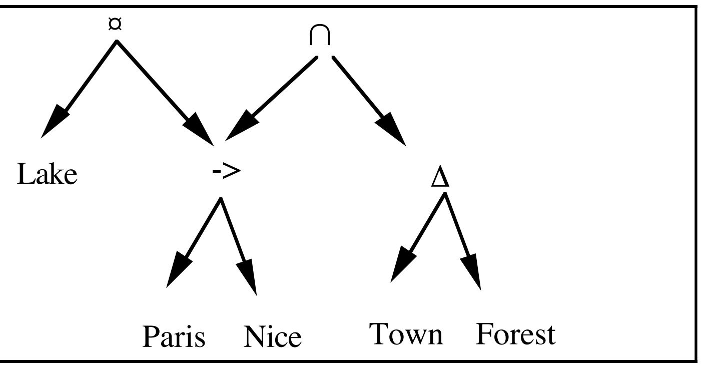


Figure 8

Since the composition of operators is not allowed in the clause Select, several statements must be defined for these queries. To manage the links between the different statements, two philosophies can be defined: (1) offering a Reference relation taking into account the history of a spatial representation or (2) introducing at each step of computation the history of a spatial representation.

IV.1. The separate history

The first solution corresponds to define a basic statement for each spatial operator. It requires to be able to manage an identifier for each new spatial representation defined by the application of the spatial operator (i.e., the function NewOid). The evaluation of the DAG may be performed with a breath first or a depth first technique [10].

To simplify the presentation, without loss of generality, let us consider only binary spatial operators. The Reference relation is equivalent to the history of a spatial object. Figure 9 presents the schema of this relation.

```
REFERENCE (Name_rel_i, Ident_i, Name_rel_j, Ident_j, Name_rel_k, Ident_k )
```

Figure 9

where Name_rel_i (resp. Name_rel_j) represents the name of the left (resp. right) argument of the spatial operator, Ident_i (resp. Ident_j) represents the identifier of the object in the relation Name_rel_i (resp. Name_rel_j), Name_rel_k is the name of the relation storing the result of the operator and Ident_k is the identifier of the object in the relation Name_rel_k.

Figure 10 presents the associated statements for the query Q_2. Let R_1 (resp. R_2, R_3) be the relation defined with the Road (resp. Town and Forest) relation since selection criteria may be applied to this road, i.e., the type is national.

```
INSERT INTO TEMP_R4 (Name_left, Name_right, Name_root,Spatial_representation)
      SELECT    T.Name, F.Name, NewOid,
                DIFFERENCE ( T.Spatial_representation, F.Spatial_representation  )
      FROM      R2 T, R3 F
      WHERE     intersection ( T.Spatial_representation, F.Spatial_representation  )


INSERT INTO R4 (Name, Spatial_representation)
      SELECT    Name_root, Spatial_representation
      FROM      TEMP_R4

INSERT INTO REFERENCE
      SELECT    'R2', Name_left, 'R3', Name_right, 'R4', Name_root
      FROM      TEMP_R4

INSERT INTO TEMP_R5 (Name_left, Name_right, Name_root,Spatial_representation)
      SELECT    R.Name, Temp.Name, NewOid,
                INTERSECTION (R.Spatial_representation,Temp.Spatial_representation )
      FROM      R1 R, R4 Temp
      WHERE     intersection ( R.Spatial_representation, Temp.Spatial_representation )

INSERT INTO R5 (Name, Spatial_representation)
      SELECT    Name_root, Spatial_representation
      FROM      TEMP_R5

INSERT INTO REFERENCE
      SELECT    'R1', Name_left, 'R4', Name_right, 'R5', Name_root
      FROM      TEMP_R5
```

Figure 10

A relation Ri may be defined as an argument of several spatial operators. Therefore to guarantee the final coherence of the partial results, an updates propagation mechanism must be developed. Figure 11 presents an example of a spatial configuration. Figure 12 presents the partial evaluation (before the intersection) of the query with a breath first evaluation and the relation Reference after the evaluation.

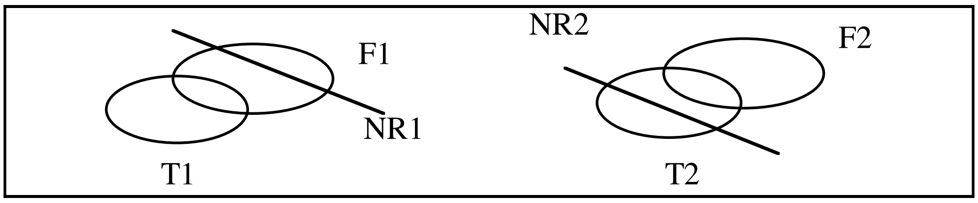
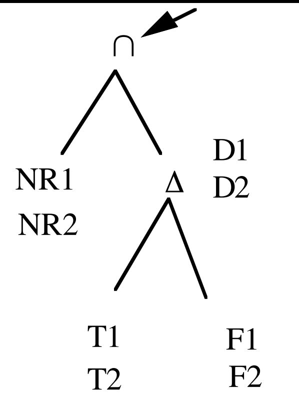


Figure 11



REFERENCE

Name_rel_i	Ident_i	Name_rel_j	Ident_j	Name_rel_k	Ident_k
R2	T1	R3	F1	R4	D1
R2	T2	R3	F2	R4	D2
R1	NR2	R4	D2	R5	I1

Figure 12

Once the intersection is evaluated, the road NR1 and the object D1 have to be removed from the relevant sets of "objects." Therefore, the town T1 and the forest F1 are also to be removed. Figure 13 presents the SQL statements associated to the management of the right argument of the intersection operator. The stop of this recursive mechanism (up and down in the DAG, i.e., figure 8) is guaranted since the only involved operation is the deletion.

```
Delete
From     R4
Where    Name not in  (  Select  Ident_j
                         From    REFERENCE
                         Where   Name_rel_i = 'R1'      and
                                 Name_rel_j = 'R4'      and
                                 Name_rel_k = 'R5'              )

Delete
From     REFERENCE
Where    Name_rel_i = 'R2'       and
         Name_rel_j = 'R3'       and
         Name_rel_k = 'R4'       and
         Ident_k not in (  Select  Name
                           From    R4    )
```

Figure 13

Conclusion :

This solution provides a standardization of the treatments. At each step of the query resolution process the schema of the relations Ri is constant. An ADT function or a system function (i.e., NewOid) must be available to deliver an Object IDentifier (OID) for each new object created by the application of a spatial operator.

IV.2. The combined history

The second solution corresponds to combine the history of the new object and its spatial representation. Let R_1, R_2, R_3 be the relations previously defined. Figure 14 presents the basic statements for the query Q_2. The generalization of this solution defines a basic statement for each level [10] in the DAG modelling a query.

```
INSERT INTO R4 (Name_left, Name_right, Spatial_representation)
        SELECT  T.Name, F.Name,
                DIFFERENCE (T.Spatial_representation, F.Spatial_representation)
        FROM    R2 T, R3 F
        WHERE   intersection (T.Spatial_representation, F.Spatial_representation)

INSERT INTO R5 (Name_left, Name_right_1, Name_right_2,  Spatial_representation)
        SELECT  R. Name, Temp.Name_left, Temp.Name_right,
                INTERSECTION (R.Spatial_representation, Temp.Spatial_representation)
        FROM    R1 R, R4 Temp
        WHERE   intersection (R.Spatial_representation, Temp.Spatial_representation)
```

Figure 14

One can remark that since these statements are independent, incoherencies may appear. In the general case, two operators sharing a same argument have no reason to belong to the same level (in the DAG). Therefore an updates propagation mechanism is also required.

Conclusion :

The advantage of this solution is to simplify the process of update propagation mechanism since the equivalent of the Reference relation is included in the relations Ri.
The main drawback of this solution is the non-standardization of the treatments since the schema of the relation Ri always changes. Furthermore since the history of an object is present, redondant information is stored in the relations Ri.

IV.3. Conclusion

This class of ESQL defines clearly the result of a GIS query. Since the update propagation mechanism guarantes the global coherence, the result of a query is provided by a set of relations (with or without the Reference relation depending on the management of the history). The management of the alphanumerical data part can be defined with join operations (depending on the semantics of the operators).
To simplify the presentation we do not introduce the management of the network component and the arity of the operator in the toy database. The network component (i.e., a graph defined with nodes and edges) only requires to change the schema of the relations Ri to take into account the order of the basic component (i.e., the edges) defining a path. The arity is managed with the null value in the complete schema of the relation Reference (in the separated history).
The main drawback is the difficulty of defining a query optimizer. The update propagation mechanism is based on the join operator (known as a costly operator). The optimizer is therefore very important. This function is handled by the query resolution engine. A global optimizer (using a set of ESQL queries) or a partial optimizer (for each ESQL query) has to be defined out of the DBMS.

V. THE SPATIAL RELATIONSHIP AS OPERATORS WITH OPERATORS AS ARGUMENTS

The third class of Extended SQL (ESQL) query language also requires the closure of the spatial operators. The predicates introduced in the first class of ESQL are also supported by this class. The spatial operators defined in the Select clause with attributes as arguments introduced in the second class are also supported by this class. Therefore, the figure 3 still represents the formalization of the query Q_1.

The main goal of this class is to avoid the external management of the update propagation mechanism. The query must therefore be expressed with a single ESQL statement. To construct a partial result in the ESQL statement, two philosophies may be defined: (1) the introduction of the composition in the Select clause or (2) the definition of a "new" relation in the From clause (like [8] does). The figure 15a and 15b present the formalization of the query Q_3 with the two philosophies.

```
SELECT   ADJACENCY (  L.Spatial_representation,
                      PATH (T1.Name, T2.Name)  ),
         INTERSECTION (  PATH (T1.Name, T2.Name),
                         DIFFERENCE  (  T3.Spatial_representation,
                                        F.Spatial_representation  ) ),
         PATH (T1.Name, T2.Name),
         DIFFERENCE (T3.Name, F.Name),
         T1.Name, T1.Spatial_representation,
         T2.Name, T2.Spatial_representation,
         T3.Name, T3.Spatial_representation,
         F.Name,  F.Spatial_representation,
         L.Name,  L.Spatial_representation
FROM     Forest F, Town T1 T2 T3, Lake L
WHERE    T1.Name = 'Paris' and
         T2.Name = 'Nice' and
         intersection (T3.Spatial_representation, F.Spatial_representation)
```

Figure 15a

```
SELECT    ADJACENCY (  L.Spatial_representation,
                       P.Spatial_representation ),
          INTERSECTION (     P.Spatial_representation,
                             D.Spatial_representation ),
          L.Name,
          D.Name_T, D.Name_F,
          P.Origin, P.Destination
FROM      D (Name_T, Name_F, Spatial_representation)
            as (      SELECT    T.Name, F.Name,
                                DIFFERENCE (  T.Spatial_representation,
                                              F.Spatial_representation)
                      FROM          Town T, Forest F
                      WHERE     intersection (      T.Spatial_representation,
                                              F.Spatial_representation) ),

          P (Origin, Destination, Spatial_representation)
               as (   SELECT    T1.Name, T2.Name
                                PATH (T1.Name, T2.Name)
                      FROM      Town T1 T2
                      WHERE     T1.Name = 'Paris' and
                                T2.Name = 'Nice'      ),
          Lake L
```

Figure 15b

Conclusion :

The first approach (i.e., figure 15a) is closer to the natural formulation of a GIS query. A redundancy of the definition appears in the Select clause whenever an operator is used twice in the definition of the query (i.e., the PATH operator in the query Q_3). Nevertheless, this drawback is reduced since this kind of language is used as the target language of a higher level user-interface. The second approach (i.e., figure 15b) implies to manage a recursive application of the From clause allowing ahead reference. The formulation of a query is much more complex to express.
The main advantage is to specify a declarative query and to rely on the DBMS for the coherent and optimized resolution of the query.
The main drawback of these formulations is the redundancy since the final relation is not normalized. The volume of the relation may be important as soon as two independent operators are defined in the query (i.e., query Q_3).

VI. CONCLUSION

Geographical Information Systems are now widely used for several applications. Since the number of applications is considerable, it is of prime importance to offer an application-independent and DBMS-independent query language. Visual query languages are very promising as a user-interface. They are more natural than textual query language and they offer a higher level of abstraction. An Extended SQL query language seems to be very promising as a spatial DBMS query language. The developement of SQL as a standard for querying relational database provides a strong basis. Nevertheless, many works have to be performed before providing an Extended SQL with the complete spatial statements.

The Select clause represents the definition of the result. The relational algebra is still coherent with the spatial extentions. Nevertheless, the management of aggregate functions of SQL implies to revise the definition of an SQL statement:

. The semantics of the star must be defined in the case of spatial operators. With SQL, the semantics of the star is coherent since the result of the relational operation is available in the result (i.e., selection or join operation). Once a spatial operator is involved in the ESQL statement, the result of the spatial operator is not directly available in the current definition of the star.

. The definition of the Group by clause must be revised since the spatial operator may appear in the clause Select. As an example, figure 16 presents the formalization of the query Q4.

```
Select      Road.Name,
            Polluted_area.Name,
            INTERSECTION (    Road.Spatial_representation,
                              Polluted_area.Spatial_representation)
From        Road, Polluted_area
Where       Road.Type = 'National' and
            intersection (  Road.Spatial_representation,
                            Polluted_area.Spatial_representation)
Group by    Road.Name
Having      Sum ( Length (INTERSECTION (  Road.Spatial_representation,
                                          Polluted_area.Spatial_representation)))
                  < 15
```

Figure 16

Following the SQL philosophy, the Select clause is not valid since for example, the "attribute" INTERSECTION is not present in the clause Group by. Providing such an attribute in the clause Group by will lead to a wrong result since the constraint on the length is applied for the complete road. Furthermore, this formulation does not provide a correct answer in case of an overlapping space division. Figure 17 presents such an example.

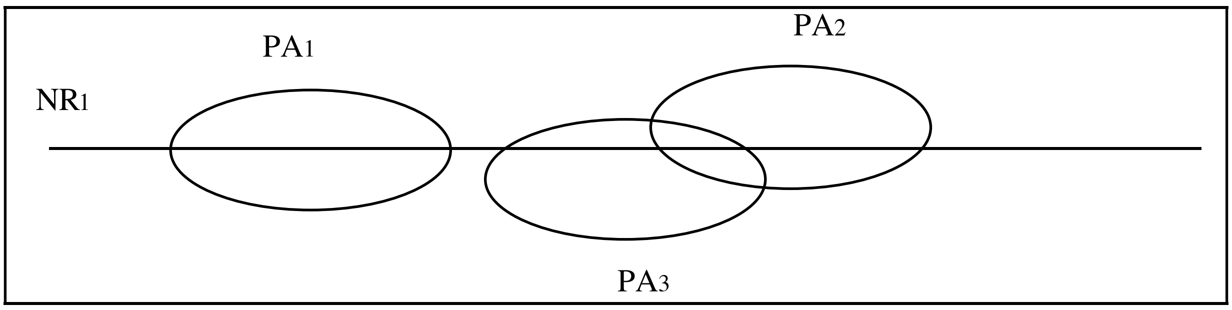


Figure 17

. The data model must take into account the spatial relevance of the alphanumerical attributes (Data Definition Language). The ADT associated to the spatial representation must at least handle the logical network representtion, the heterogeneous geometrical representation (i.e., in the same spatial_representation the notion of point, line and polygon with holes).

. The query language must provide network facilities, a core of spatial operators and a coherent management of the results (Data Manipulation Language).

REFERENCES

[1] Bennis K., David B., Quilio I., Viemont Y.: Géotropics: Database Support Alternatives for Cartographic Applications, 4th Int. Symp. on Spatial Data Handling, Zurich, Switzerland, July 1990

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

[3] Calcinelli D., Mainguenaud M.: The Management of the Ambiguities in a Graphical Query Language for GIS, 2nd Symp. on Large Spatial Databases, Zurich, Switzerland, Aug. 1991, Lecture Notes in Computer Science (LNCS) n° 525

[4] Cruz IF, Mendelzon AO, Wood PT: A Graphical Query Language Supporting Recursion, Proc. SIGMOD Conf., San-Fransisco, USA, May 1987

[5] Egenhofer M.: Why not SQL !, Int. Jou. Geographical Information Syst., vol 6 n°2, 1992

[6] Frank A., MAPQUERY: Database Query Language for Retrieval of Geometric Data and their Graphical Representation, Computer Graphics, Vol 16, n°3, July 1982

[7] Güting RH, Geo-Relational Algebra: A Model and Query Language for Geometric Database System, Proc. of the Int. Conf. on Extending Data Base Technology, Venice, Italy, March 1988

[8] Haas LM, Cody WF : Exploiting Extensible DBMS in Integrated Geographic Information Systems, 2nd Symp. on Large Spatial Databases, Zurich, Switzerland, Aug. 1991, Lecture Notes in Computer Science (LNCS) n° 525

[9] Jungert E.: Inference rule in a Geographical Information System, IEEE Workshop on Language for Automation, New-Orleans, USA, November 1984

[10] Jourdas C., Mainguenaud M. : A Query Resolution Model to Manage Networks : Application to an Extended Relational DBMS, 2nd European Geographical Information System Conf. Brussels, Belgium, Apr. 2-5, 1991

[11] Kemp Z. : An Object-Oriented Model for Spatial Data, 4th Int. Symp. on Spatial Data Handling, Zurich, Switzerland, July 23-27 1990

[12] Kim HY and al: PICASSO: A Graphical Query Language, Software-Practice and Experience, Vol 18(3), 169-203, March 1988, Ed. J. Wiley and Sons Ltd

[13] Kirby K.C., Pazner M.: Graphic Map Algebra, 4th Int. Symp. on Spatial Data Handling, Zürich, Switzerland, July 1990

[14] Mainguenaud M.: The Gap between a Visual Query Language and a Data Base Query Language: Application to a Geographical Information System (submitted to publication)

[15] Mainguenaud M., Portier MA: CIGALES: A Graphical Query Language for Geographical Information Systems, 4th Int. Symp. on Spatial Data Handling, Zurich, Switzerland, July 1990

[16] Myers BA (Ed.): Languages for Developing User Interfaces, Jones and Bartlett Publishers, Boston, 1992

[17] Sacks-Davis R., Mc Donnell KJ, Ooi BC: GEOQL: A Query Language for GIS, Australian and New Zealand Ass. for the Advancement of Science Congress, Townsville, Australia, Aug. 1987

[18] Schek HJ, Waterfeld W: A Database Kernel System for Geoscientific Applications, 2nd Spatial Data Handling, Seatlle, USA, July 1986

[19] Scholl M., Voisard A.: Themetic Map Modelling, 1st Int. Symp. on Large Spatial Databases, Santa-Barbara, USA, July, 1989, LNCS n° 274

[20] Shu N.C.: Visual Programming, Van Nostrand Reinhold Cie, New York, 1988

[21] Smith TR, Menon S, Star JL, Ester JE: Requirements and Principles for the Implementation and Construction of Large Scale GIS, Int. Jou. Geographical Information Syst., vol 1 n°1, 1987

[22] Stemple D, Sheard T, Bunker R: Abstract Data Types in Databases: Specification, Manipulation and Access, Proc of Int. Conf. on Data Engeneering, Los Angeles, USA, Feb. 1986

[23] Svensson P., Huang Z.: Geo-Sal: A Query Language for Spatial Data Analysis, 2nd Symp. on Large Spatial Databases, Zurich, Switzerland, Aug. 1991, LNCS n° 525

[24] Ullman J.D. : Principles of Databases Systems, Computer Science Press, Maryland 1982

[25] Vijlbrief T., Van Oosterom P.: The GEO++ System: an Extensible GIS, 5th Int. Symp. on Spatial Data Handling, Charleston USA, Aug. 1992