Query Models and Languages for Geographical Information Systems

Michel Mainguenaud

Laboratoire Perception, Systeme et Information
Institut National des Sciences Appliquees (INSA)
Site du Madrillet - Avenue de l'Universite
F76800 Saint Etienne du Rouvray - France
Fax : (+ 33) (0) 2 32 95 97 08
Michel.Mainguenaud@insa-rouen.fr

Abstract. This paper presents a synthesis on the query models and languages to manipulate a geographical database. We present the different classes of query languages : based on predicates, based on operators without composition and based on operators with composition. We analyze the consequences on the data model, on the expressive power and on the query modeling. The introduction of operators as query primitives requires the closedness of these operators on geographical data. The introduction of operators increases the expressive power allowing queries involving a composition of operators. As a path operator (with the same arguments) provides several answers and may appear several times in a query, the query modeling must provide such an opportunity. Depending on the required expressive power, we present the different classes of interfaces at the user's level.

1 Introduction

A lot of efforts are under progress to elaborate innovative solutions for the representation and exploration of complex database applications. Different research groups are simultaneously concentrating their works on Geographical Information Systems (GIS). GIS needs are very well known [11,12]. Nevertheless, several problems are still open. In this paper we focus on the analysis of a query modeling and the user interfaces of such systems.

Geographical data are defined with two major components : an alphanumeric part and a spatial representation. Conventional databases provide an efficient way to manage alphanumeric data. The spatial representation requires to extend the conventional data types (e.g., integer, string). With these new types, some basic manipulation primitives must be defined. The graphic representation of geographical data is very important. Geographical data are visual by essence. The user interface has a very important role in the acceptation of a new tool. Visual techniques may have a tremendous opportunity to play an important part in a query of a GIS database at the user's level. We distinguish two levels of manipulations. The first level involves the programmer of an application. The second level involves the end-user. The development of conventional database applications can be performed with two main orientations. The first orientation is the introduction of database manipulation primitives in a conventional programming language (e.g., C, C++, Java). The second orientation is the use of a new development language

LNCS n° 1929
pp 511 - 520

Visual 2000
4th Int. Conf. on Visual Inform. Systems
Lyon, France
Nov. 2-4, 2000

(often named Fourth Generation Language - 4GL). A geographical application requires the same tools. These tools are a set of database manipulations and a set of programming constructors. The expressive power of the programming level (i.e., the class of queries a user can express) depends on the expressive power of the database geographical manipulations since programming constructors nowadays are very conventional (e.g., sequence, alternative, iteration). Graphical screens and graphical co-processors increased the opportunity to define new kinds of user interfaces. We distinguish the static querying and the dynamic querying. We define the static querying as the use of a predefined scenario of database manipulations (alphanumeric and spatial). This approach is very well adapted to define very repetitive queries. We define the dynamic querying as the full use of the visual nature of geographical data. A query is therefore represented with a drawing that expresses the semantics of a query. Obviously, the expressive power depends on the ability of the database to manage geographical operators. The user friendliness of a dynamic querying language must be favored whenever some degrees of freedom are required.

In part 2, we present the various philosophies of geographical database extensions to handle a spatial database. In part 3, we study the consequences of the introduction of a composition of operators as a query. In part 4, we study the associated user interfaces to handle an end-user query. The conclusion presents a synthesis and gives some research guidelines.

2 Query Model

The query model defines the way a query uses the notion of operators. We consider here the treatment applied on geographical data with a Data Base Management System. The treatment of the spatial representation outside the DBMS (i.e., a software component able to extract data from the database world, to apply a spatial operator and to re-introduce data inside the DBMS) is not considered here. The first step is the definition of geographical (i.e., in this way graphical) data. The simplest structure of operator is a predicate. The Boolean result of a predicate applied to geographical data can be use in a 'Where' clause of an 'SQL' statement. The expressive power is increased by operators. A querying language may or may not accept a composition of operators. In the first part, we introduce a sample database to illustrate the different examples used in this paper. In the second part, we study the introduction of predicates. In the third part, we study the notion of operators and in the last part, we study the notion of composition of operators.

2.1 Sample Database

Several formalisms [14] can be used to model a geographical database (e.g., an extended relational model with Abstract Data Types, an Object Oriented model). To simplify the presentation, let us use a complex object model defined with the aggregation [], the set {} and conventional types (e.g., integer, string, real). The spatial representation (or SR for short) is managed by an Abstract Data Type [13], SpatialRepresentationType (or SRT for short). The aim of this part is to study the introduction of operators and its consequences. Therefore the retained

data model used to define the sample database is not important. Let us use the following definitions to propose some queries :

TownType = [Name : string, Population : integer, Mayor : string,
SpatialRepresentation : SpatialRepresentationType]
ForestType = [Name : string, Species : {string}, SR : SRT]
PollutedAreaType = [Name : string, Reason : string, SR : SRT]
LakeType = [Name : string, SpatialRepresentation : SRT]
RoadType = [Name : string, Origin : TownType,
Destination : TownType, SR : SRT]

Let us define the following queries : (Q1) : I would like to know if a forest has a common part with a town named 'Paris' ; (Q2) : I would like to know if a forest has a common part with a town named 'Paris' such as this part has a surface of 10 units or more ?; (Q3) : I would like to know if a road crosses a town named 'Paris' in its non-forest part; (Q4) : I would like to see the paths from the town named 'Paris' to the town named 'Nice' that border a lake and cross a forest in its non-urban part; (Q5) : I would like to see a road that crosses a forest for at least a 10 units length; (Q6) : I would like to see a road that crosses a polluted area for at least a 10 units length. If we consider query Q1, the aim is to obtain information about the town (may be about the forest) if an intersection occurs. The intersection itself is not important. If we consider query Q2, the query is similar to query Q1, but now the intersection is important since a property on this intersection is required (i.e., a surface of 10 units or more). If we consider query Q3, the query is similar to query Q2, but now a spatial operator (i.e., the spatial difference is applied to the town to extract the non forest part) is involved. A new operator (i.e., an intersection) is applied to the result of the previous operator (i.e., the spatial difference). If we consider query Q4, the query is similar to query Q3 (the non-urban part of a forest). The composition of operator is introduced by the fact the path(s) from Paris to Nice must border the lake and cross the non-urban part of the forest. Query Q5 and Q6 have the same semantics (i.e., an intersection between two database components under an aggregate constraint - 10 units long). The main difference is due to the fact that conceptually two forests do not have any reason to overlap but two polluted areas may overlap.

2.2 Notion of Predicates

From the historical point of view, the notion of predicate was the first extension to conventional database systems to manage spatial data. This solution is the simplest to introduce. The use of a predicate is reduced to the 'Where' clause of an Extended SQL statement. The result is built with the basic components of the database (i.e., attributes of base relations in an extended relational database or classes from an object oriented database). The first proposed models were based on the physical representation of geographical data (e.g., point, line). This option leads to a huge amount of predicates depending on the spatial representation of the data stored in the database. The introduction of Abstract Data Types to handle the geographical model reduced the number of predicates to a single predicate by semantics (e.g., intersection, bordering). The expressive power is very low since query Q2 cannot be expressed. This query required to manage the result of a spatial operator. The result as a Boolean answer is to weak to answer query Q2.
As an example, query Q1 in such a language would be expressed by :

```
Select      T.*
From        TownType T, ForestType F
Where       intersection ( T.SR, F.SR)
```

One can remark that in this kind of language, the definition is tuple oriented.

2.3 Notion of Operators Without Composition

From the historical point of view, the introduction of operators is the second generation of extended databases to handle spatial properties. The expressive power is increased since a query may define some properties on the result of a spatial operator. Query Q2 can now be expressed since the result of the intersection is available. A function can be defined on this result and the property of a surface more than 10 units can be evaluated. Let us consider that we have already available an intersection operator ($\leftrightarrow$), a path operator (->), a border operator (<>) and a spatial difference operator (Δ). We do not consider here the signatures to express constraints (e.g., on the path operator). The important is the fact that a path operator provides several paths as an answer of an evaluation from a given place to another one. As soon as a query requires the composition of operator (i.e., an operator applied on the result of another operator), this query cannot be expressed in a single order. As a consequence, query Q3 cannot be expressed since the intersection is applied on the result of a spatial difference. Such a query is therefore expressed with two database orders. The first one allows the evaluation of the spatial difference. The second one applies an intersection on the results of the spatial difference. Stating this fact, the final result is a sub-set of the result obtained during the evaluation of the first spatial operator. A forest that has an intersection with a town is retained in the first step but this forest may not have an intersection with a road. Therefore the result is the sub-set of the forests defined in the first step. Two main practices can be defined to keep only relevant data. The first one is the management of the history as soon as an operator is evaluated. The result of an operator has the history (that may be very long depending on the number of evaluated operators). One of the drawbacks is the lack of generality since the result of an operator is not a geographical object (an alphanumeric part and a spatial part) but a geographical object and its history. The second practice is to keep this history outside the definition of a geographical object and to define the result of a query as a set of database objects (from the database and the results of the operators) and a structure to manage the history. Whatever the retained practice, a software component must be developed (i.e., a Query Resolution Engine) to guarantee the correctness of the final result of a query. This solution can be considered as similar to the solution with the treatment of the spatial representation outside the DBMS since a software component is required. As an example, query Q2 would be expressed by :

```
Select      T.*
From        TownType T, ForestType F
Where       Surface (↔ (T.SR, F.SR)) > 10
```

One can remark that in this kind of language the definition is tuple-oriented. Furthermore the operator is spatial representation oriented. The alphanumeric part of a geographical object is not considered. As soon as a query involves an operator in the select clause the alphanumeric part may be not relevant (e.g., Population) as it may be for the beginning of the expression of query Q3:

Select T.Name, T.Population, T.Mayor, Δ (T. SR, F.SR)
From TownType T, ForestType F
Where intersection (T.SR, F.SR)

Query Q3 must be cut into at least two orders to provide the part of the road that crosses the non-forest part of the town. In fact to provide a relevant answer to query Q3, the result must be defined as a geographical data (i.e., an alphanumeric part and a spatial representation). The alphanumeric part must be a subset of the data model of a town, of the forest and of the road [4,9][1]. The spatial representation is the result of the composition of an intersection applied on a spatial difference.

2.4 Notion of Operators With Composition

From the historical point of view, this approach received very few proposals [partially in 5, 6, 8]. A query language with the composition of operators allows defining a query with several spatial operators. The argument of an operator may be an operator. The expressive power is similar to the previous set of proposition if we consider the database management system as the query language and the software component required to guarantee the correctness of a result. Query Q3 can now be expressed since the spatial difference of the forest and the town is available within the same database order. The algebraic modeling of this query is a tree (see figure 2.1)

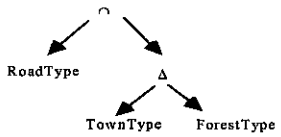

Fig. 2.1 - Algebraic representation of query Q3

We define this composition as a vertical composition. Query Q4 illustrates an horizontal composition. Figure 2.2 presents an algebraic representation of query Q4. The path operator is used as an argument of the border operator and of the intersection operator. The path must verify the two properties (bordering a lake and having an intersection with a non-urban part of a forest). The algebraic modeling of this query is an acyclic graph (DAG).

As an example, query Q3 in such a language would be expressed by :

Select R.*, F.*, ↔ (R.SR, Δ (T.SR, F.SR))
From RoadType R, ForestType F, TownType T
Where intersection (F.SR, T.SR) and
intersection (R.SR, Δ (T.SR, F.SR)) and T.Name = 'Paris'

A similar approach to the definition of query Q4 cannot be performed :

Select -> (T1.Name, T2.Name), L.*, F.*
From LakeType L, ForestType F, TownType T1, T2, T3
Where bordering [2] (-> (T1.Name, T2.Name), L.SR) and
intersection (-> (T1.Name, T2.Name), Δ (F.SR, T3.SR))
and T1.Name = 'Paris' and T2.Name = 'Nice'

---

[1] In the following, we assume this rule is respected. * denotes the relevant attributes.

[2] To simplify the expression, we do not consider here the transformation from the logical point of view of a network (i.e., a graph and the transitive closure on this graph to evaluate a path) and its spatial representation.

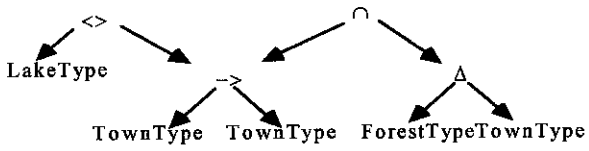

Fig. 2.2 - Algebraic representation of query Q4

As soon as a path operator provides several answers as a result instead of a single one (e.g., a shortest path), there is no guarantee that the two path operators would represent the same path. Unfortunately a path operator with a single answer is not realistic. The evaluation of the path operator must provide several paths as an answer since the shortest one (in distance) may be far from being the most convenient, the less expensive may be very long, ... The problem is due to the fact that from two given instances (i.e., Paris and Nice) several paths (i.e., several results) are provided. The signature of the path becomes a set of paths as a result instead of being a path.

3 Composition of Operators

The composition of operator provides a very high expressive power. Conventional databases relies on the opportunity to combine a reduced set of operators. Relational databases provide the selection, the projection, the union, the difference and the Cartesian product as the basic operators. To provide an opportunity to define realistic queries some functions are provided such as the minimum, the maximum, the count, the average and the sum. Object oriented database query languages provide the opportunity to define some basic manipulations attached to a specific class. These manipulations can be used in the query language. Within the geographical context, several spatial operators are provided in the literature (e.g., intersection, bordering, spatial difference). These operators can be used in the extension of the query language to manipulate spatial data. The use of a composition of operators has two main consequences. The first one is the ability to use the same operator with the same arguments in a query. The second consequence is the ability to access to the result of the composition in the end-user query language.

3.1 The Use of the Same Operator

Several formalisms can be provided to formalize a query language. To illustrate the notion of composition, we adopt in this paper a simplified grammar of a functional query language (the choice of an other formalism - e.g., logic - does not change the raised problem). Let us define a simplified grammar (e.g., we do not consider the signature of the operators, we reduce it to binary operators) based on the following rules to illustrate the use of the same operator. The start symbol is "query". The terminals are DatabaseComponent (e.g., TownType), OID (i.e., an integer), spatialrelationship (e.g., ↔, Δ, ->)

query::= composition (op , follow_op)
op::= OID spatialrelationship(op,op) / DatabaseComponent OID
follow_op::= op follow_op / ε

Query Q1 involves a predicate but can be generalized with an operator. Query Q2 is similar to query Q1 since we do not consider here the signature of the operator for the various constraints that can be expressed. Query Q3 is a vertical composition (similar to conventional functional languages).

Q1 = composition (3 ↔ (TownType 1, ForestType 2))
Q3 = composition (5 ↔ (RoadType 4, 3 Δ (TownType 1, ForestType 2)))
Q4 = composition (3 -> (Towntype 1, TownType 2),
5 <> (3 -> (TownType 1, TownType 2), LakeType 4),
8 ↔ (ForestType 6, TownType 7)
10 ↔ (3 -> (TownType 1, TownType 2),
9 Δ (ForestType 6, TownType 7)))

A path operator may be applied several times in the same query (e.g., two times). The semantics of the query may be : 'I would like two paths from a given place to another one' or 'I would like this path to verify two different properties' (this is different from the disjunction of the two properties). In the first case, the DBMS must consider these two paths as independent. This is possible since the path operator provides several paths as an answer. A path operator reduced to a single path as answer (e.g., a shortest path) cannot provide this expressive power. In the second case, the DBMS must consider these two path operators as the same one. To be able to distinguish between these two cases, the grammar must provide an OID to indicate that these paths are similar or independent. This technique is similar to the definition of an alias in a SQL statement.

3.2 Consequences on the User Interfaces

The composition of operators introduces the fact that several parts of a graphic representation are defined depending on the semantics of an operator. Let us use for example the spatial intersection. Figure 3.1 presents a symbolic representation of an intersection.

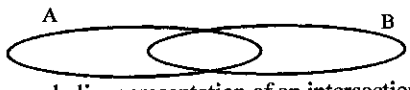

Fig. 3.1 - symbolic representation of an intersection

The application of the intersection provides (1) the intersection itself, (2) the part of A that does not intersect with B, (3) the part of B that does not intersect with A. The user interface based on a visual representation must provide the ability to precise the relevant part in a new subquery. Two queries with the same semantics must have the same visual expression. As an example a symbolic representation for query Q5 (or Q6 since they have the same semantics) is represented Figure 3.2 (we do not consider the way the drawing is performed - i.e. the type of user interface used to define such a query).

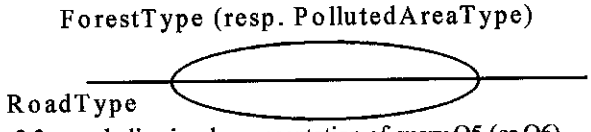

Fig. 3.2 - symbolic visual representation of query Q5 (or Q6)

Since a user do not have to know the way data are stored in the database, the management of the overlap or not of the database components must not be handled at this level. The formal modeling of a query must be similar. The query must be formally represented by an expression like

composition (3 ↔ (ForestType 1, RoadType 2))

or composition (3 ↔ (PollutedAreaType 1, RoadType 2))

The DBMS must be able to evaluate whether or not an overlap may occur or not since a wrong result may be obtained as soon as a tuple oriented philosphy is used for the query language. A SQL statement of query Q5 (or Q6) must be like :

```
Select      R.*,↔ ( R.SR, F.SR)
From        RoadType R, ForestType F
Where       intersection ( R.SR, F.SR)
Group by    R.name
Having      (length (↔ (R.SR, F.SR)) > 10 )
```

Figure 3.3 presents a symbolic representation of a data set. The strict application of such a query involves an inconstancy since the intersection of the road and the common part of the two polluted area is summed twice (in case of an overlap).

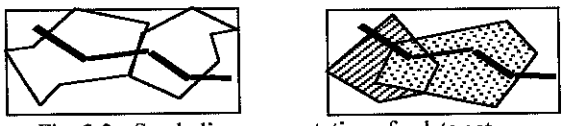

Fig. 3.3 - Symbolic representation of a data set

To provide a unique query formulation, the Data Definition Language must provide a supplementary clause managing the conceptual overlap or not between two instances of a type (relation or class depending on the data model). This precision can also be considered as an integrity constraint.

4 User Interfaces

An end user query language for GIS must emphasize the visual nature of geographical data. Numerous propositions [3, 7, 10] or studies on visual interfaces have already been performed [1, 2]. In this part, we consider the second level of manipulations (i.e., the end-user interface). Within the two kinds of querying, we consider the dynamic querying. A dynamic querying is based on the representation of the required spatial relationships as querying primitives. The drawing, modeling a query, may be provided by the end-user (with the problem of the interpretation of this query to express it with a formal language) or provided by the system using the user's directives (i.e., the required relationships). In both of them, ambiguities may appear depending on the allowed expressive power of the interface.

A database query language based on spatial predicates is not relevant for this kind of end-user interface. The operators are mandatory. Once operators are available, the problem is to determine whether they accept or not the composition. An underlying database query language, without composition and without the software able to simulate this composition reduces the expressive power since manipulations can only be performed on basic components of the database. The visual representation of a query is simplified since few ambiguities may appear. An

underlying database query language without composition but with a software able to simulate it provides a better expressive power. This expressive power is similar to the one obtained with a query language allowing the composition. The level of ambiguity is closely linked to the existence or not of the spatial difference operator (since the number of relevant sub-parts of a query may be important - Figure 3.1).
A query language with a path operator between two given places providing a unique path as an answer (e.g., shortest path) or without a path operator can be designed upon a database extended with spatial (network) operators. As soon as the path operator provides several paths as an answer, the query language must be extended to allow the horizontal composition with identification of the path operator (since it may appear several times in the same query with the same arguments). Furthermore, the definition of a query with a path operator requires the introduction of aggregates (e.g., a cost less than 100 units). These aggregates are nearly mandatory to provide a realistic computational time and ... a realistic use of the results in the case of large databases. The aggregates may introduce also some ambiguities since the result of the path operator (i.e., a set of path) cannot be considered as a unique path.

5 Conclusion

The user-friendliness of a visual interface is one of the major argument for the acceptation of a new tool. The expressive power is a second one. Geographical data are by essence visual. A visual approach to query a geographical database seems a very promising solution. The expressive power of the geographical query languages varies depending on the query primitives. From the historical point of view, the introduction of spatial predicates was the first attempt. Its weakness is the very low expressive power. The second attempt was the introduction of spatial operators. The problem is then to allow or not the composition of operators. The composition of spatial operators is the best solution to provide a powerful database query language.
The user interface may rely on a 'click in a box' approach to a full freedom visual query language. The higher is the level of abstraction, the more difficult is the interpretation of a visual query. Ambiguities may be raised whenever a component of a query is selected or whenever a spatial relationship is visualized. From a visual query language to an alphanumeric database query language, the gap may be important. The composition of operators is a major requirement to define a realistic query language for geographical applications. A difference is introduced between a visual query language and a visual interface to a geographical DBMS. A visual query language must provide a higher level of abstraction than the query language of the DBMS is able to offer. A DBMS query language based on predicates leads to a very weak expressive power. A DBMS query language with a set of spatial operators but without composition requires a visual language allowing a composition and a software (applied in the middleware) to simulate the composition and to guarantee the consistency of final results. A DBMS allowing the composition (with the possibility to use several times the same operator with the same arguments in a query) would require now a sophisticated visual interface able to take into account the various components of the composition of operators. The definition of such an interface is therefore a challenge.

References

1. Aufaure M.A., Trepied C. : A Survey of Query Languages for Geographic Information Systems. Interfaces to databases (IDS-3), Napier University Edinburgh, 8-10 July (1996)
2. Batini C., Catarci T., Costabile M.F., Levialdi S.: Visual Query Systems: A Taxonomy, Visual Database Systems II, IFIP-TC2/WG2.6, Budapest, Hungary, IFIP Transaction A7, 30/9–3/10 (1991)
3. Calcinelli D., Mainguenaud M.: Cigales, A visual Query Language for geographical Information System : the User Interface, Journal of Visual Languages and Computing, Vol 5, Academic press, (1994), 113-132
4. Claramunt C, Mainguenaud M., : A Revisited Database Projection Operator for network Facilities in a GIS, Informatica, 23, (1999), 187-201
5. Guting, R. H., GRAL: An extensible relational database system for geometric applications. In Proceedings of the 15th International Conference on Very Large Data Bases, VLDB, 22-26 August Amsterdam, The Netherlands, (1989)
6. Haas, L., Cody, W. F., Exploiting extensible DBMS in integrated GIS. In Proceedings of the 2nd International Symposium on Large Spatial Database, Gunther, O. and Schek, H.-J. Eds, Springer-Verlag, Zurich, Lecture Notes in Computer Science, n° 525, (1991)
7. Egenhofer M., Spatial-Query-by-Sketch, IEEE Symposium on Visula Languages (VL), Boulder, Colorado, USA, 3-6 September, (1996)
8. Larue, T., Pastre, D. and Viémont, Y., Strong integration of spatial domains and operators in a relational database system. In Advances in Spatial Databases, Abel, D. J. and Ooi, B. C. Eds., Springer-Verlag, Singapore, Lecture Notes in Computer Science n° 692, (1993)
9. Mainguenaud, M., Consistency of geographical information system results. Computers, Environment and Urban Systems, Vol. 18, Pergamon Press, (1994), 333-342
10. Meyer B., Beyond Icons : Towards New Metaphors for Visual Query Languages for Spatial Information Systems. Ineterfaces to database Systems (IDS92), Glasgow, UK, 1-3 July (1992)
11. Peuquet DJ: A Concepetual Framework and Comparison of Spatial Data Models, Cartographica, Vol 21 (4), (1984) 66-113
12. Smith TR, Menon S, Star JL, Ester JE: Requirements and Principles for the Implementation and Construction of Large Scale GIS, Int. Journal of Geographical Information System, Vol 1, n°1, (1987), 13-31
13. Stemple, D., Sheard, T. and Bunker, R., Abstract data types in databases: Specification, Manipulation and Access. In Proceedings of the 2nd Int. Conference on Data Engineering, Los Angeles, USA, 6-8 Feb (1986)
14. Ullman JD: Principles of Database and Knowledge-base Systems, Computer Science Press, Maryland, (1988)