FROM THE USER INTERFACE TO THE DATABASE MANAGEMENT SYSTEM: APPLICATION TO A GEOGRAPHICAL INFORMATION SYSTEM

M. MAINGUENAUD

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

Abstract

This paper presents the links between a visual programming language, Cigales, and a database management system in the context of a Geographical Information System. We study the way of filling the gap between the semantic level of a visual query language and a data base query language.

1. 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 System (GIS). GIS needs are very well known [6]. Several problems are still open. In this paper we focus on the design of a user-friendly query language. Designing the user interface is one of the main issues to deal with when building spatial information systems. It will be increasingly true since geographic information is now affecting such general applications as tourism and non-specialist users. The visual (or graphical) query language concept appeared with the development of "cheap" graphical devices. Propositions of such languages are detailed for example in [2,3,5]. A query language is said to be declarative whenever the query defines the properties to be verified by the results but not the way of obtaining them. It is said to be visual whenever the semantics of the query is expressed by a drawing. Cigales is a declarative visual query language. The query is defined with a visual Query-By-Example philosophy. The figure 1 represents the graphical editor used to define the query Q_1: "Which routes go from Paris to Nice with the qualification that they border a lake for while and then cross a forest?". A scenario of a query is defined in [1].

Figure 1

Part 2 presents the link between the user-interface and the Data Base Management System (DBMS); Part 3 presents the conclusion.

2. THE LINK BETWEEN THE USER-INTERFACE AND THE DBMS

2.1. The data model

A user should have a high level query language. Unfortunately, the DBMS offers very low level query languages. Traditionally, geographic information is divided into two parts: the thematic-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. Thematic-oriented operators are based upon the geometrical representation (i.e., intersection). Network-oriented operators are based upon graph manipulations whatever the geometrical representation is (i.e., the transitive closure of a graph). To take into account thematic-oriented data and network-oriented data, we extend the classical data models by providing a new specification: a Thematic object or a Network object. The data base designer is responsible to define an object as a Thematic or Network object. The basic rules to define these objects are the following: (1) Every object is a Thematic object; (2) An object appearing as a component of a logical graph is a Network object (i.e., a node or an edge)

2.2. The query modelling

To analyze the query and to translate it into DBMS understandable orders, a visual query must be represented with an internal formalism. A query is defined by a combination of several operators (the results of an operator may be used by any other operators). A functional-like language is well adapted to model such a combination. This part presents the operators and the formalism retained to model a query.

The operators are defined as Abstract Data Type operators. We consider, here, the logical relationships between two objects. These operators are independent from the specifications of an operator following the spatial representation (i.e., is there an intersection between two spatial representations when they have only one point in common?). The operators involved in the Cigales language are represented in figure 2. These operators are decomposed into two parts: the user level and the system level. The user level operators are manipulated by the end-users. The system level operators are used to fill the gap between the semantic level of a visual query language and a DBMS query language. The semantics of the operators are defined in [1]. Each operator has two representations: the operator and the negation of the operator (except the euclidian distance and the before operator).

The user level operators:

the geometrical intersection (∩), the euclidian distance (√),
the adjacency (¤), the path (->), the inclusion (C)

The system level operators :

the difference (Δ), the union (∪), before (<<), the join of paths (-><-), the direct link (Π_D),the intersection of paths (Π_{is_e}), the inclusion of paths (Π_{ic_e}), the inclusion of stop place(s) in a path (Π_{ic_ne}), the intersection of stop places (Π_{is_n}), the inclusion of stop places (Π_{ic_n}), the beginning of a path (>->), the end of a path (->><)

Figure 2

The basic assumption to define an operator is: **An operator has several sets of objects in input and delivers a set of objects as output.** The Thematic/Network signatures of the operators are presented in figure 3. The other combinations are not defined. An operator is said to be Network (resp. Thematic) whenever its result is Network (resp. Thematic).

∩ (X,Y) : Thematic	X ∈ {Thematic, Network}, Y ∈ {Thematic, Network
√ (X,Y) : Thematic	X ∈ {Thematic, Network}, Y ∈ {Thematic, Network}
¤ (X,Y) : Thematic	X ∈ {Thematic, Network}, Y ∈ {Thematic, Network}
-> (X,Y) : Network	X ∈ {Network}, Y ∈ {Network}
C (X,Y) : X	X ∈ {Thematic, Network}, Y ∈ {Thematic, Network}
Δ (X,Y) : Thematic	X ∈ {Thematic, Network}, Y ∈ {Thematic, Network}
∪ (X,Y) : Thematic	X ∈ {Thematic, Network}, Y ∈ {Thematic, Network}
<< (X,Y,Z) : Z	X∈{Thematic,Network},Y ∈ {Thematic,Network},Z ∈ {Network}
-><- (X,Y) : Thematic	X ∈ {Network}, Y ∈ {Network}
>-> (X) : Thematic	X ∈ {Network}
->< (X) : Thematic	X ∈ {Network}
Π_{type} (Network, Network) -> Network	type ∈ {D, is_e, ic_e, ic_ne, is_n, ic_n}

Figure 3

To simplify the presentation, without loss of generality, let us consider the query Q_1. An intuitive functional-like notation of this query is the following:

<< (¤ (-> (Paris, Nice), Lake), ∩ (-> (Paris, Nice), Forest), -> (Paris, Nice))

This query involves three user level operators: the adjacency operator (¤) applied on two labeled objects (σ_{Fi} O_i), the intersection operator (∩) and the path operator (->) and one system level operator: the before operator (<<). A more precise definition of the query is:

<< (¤(->(σ_{F1} O_1, σ_{F2} O_2), σ_{F3} O_3), ∩ (-> (σ_{F1} O_1, σ_{F2} O_2), σ_{F4} O_4), -> (σ_{F1} O_1, σ_{F2} O_2))

with σ_{F1} : [Type = Town and Name = Paris], σ_{F2} : [Type = Town and Name = Nice]

σ_{F3} : [Type = Lake], σ_{F4} : [Type = Forest]

A query is modelled using two basic components: the leaves and the operators.

The leaf represents the basic element involved in a query (σ_{Fi} O_i). The user's conditions defined are modelled by the label or by the constraint depending on their semantics. The formalism is a tuple (ident_leaf, label_leaf, constraint_leaf). It relies on three notions: (1) Ident_leaf is an identification of the leaf (i.e., an integer) since several basic data elements with the same characteristics may be involved in the same query (i.e., "Which are the forests shared by two towns ?"); (2) Label_leaf represents the selection criteria. The selection criteria on the multi-valued attributes may be defined with several classical self-explaining functions: (i.e., Min, Max, Avg, Count, Exist, Not_exist). The selection criteria on the spatial representation attribute may be defined with the following self-explaining functions: Hole, Connectivity, Equality. They are represented by classical connected triplets <attribute> <operator> <valor> (at least the type of the manipulated objects must be defined before the evaluation - i.e., Type = Lake). (3) Constraint_leaf represents the constraints defined on

the results of the evaluation following the selection criteria defined in the label (i.e., the surface is less than 10 km^2).

An operator symbolizes the results of this operator. The formalism is a tuple (ident_op, operator, label_op, constraint_op, {op_i}). It relies on five notions: (1) Ident_op is an identification of the operator (i.e., an integer) since several operators with the same characteristics may be involved in the same query (i.e., "I would like two routes without intersection to go from Paris to Nice"); (2) Operator represents the semantics of the operator (i.e., intersection); (3) Label_op represents the selection criteria; (4) Constraint_op represents the constraints defined on the results obtained by the application of the operator; (5) Op_i represents the information on which the operator is applied (i.e., the leaves or the recursive definition of the operators). The label can be defined at two levels. The first level manages the data model of the "object" obtained by the application of an operator. Since no precise rule has been defined to determine the data model, this level is for the time unused. The second level manages the components of the result. Depending on the operator, the interpretation is different. For a thematic-oriented operators, the label represents the label of the "objects" obtained in Op_i. For the network-oriented operators, the label has the same semantics as the label in the leaf structure since the basic element of the operator is equivalent to a basic element evaluated in the leaf phase. The formalism is based on [2] and relies on a regular expression (i.e., (Type = Road and Name = RN13)+). The constraints are defined in the same way as the leaf structure. Depending on the operator the interpretation is different. For the thematic-oriented operators, they represent the number of different possible solutions and no constraints are defined for the time on the data model of the result. They are defined with the At_least, No_more, Exactly clauses. For the network-oriented operators, the constraint represents the management of aggregate functions on the result (i.e., the total length is less than 1000- km long).

The internal modelling of a query has two main goals: (1) to define a query with an application-independent and DBMS-independent formal representation and (2) to hide from the user's point of view the gap between the semantic level of a visual query language and a DBMS query language.

2.3. The grammar

This part presents the grammar associated to the query modelling. Let ε be the empty component, / be the alternative, <xxx> be a post-defined or pre-defined symbol, <valor> be a constant, a variable ($$X_i$$) or a function (i.e., length), <attribute> be an attribute of the data model (The conditions defined as an expression of several attributes are not presented), <ident_op>, <ident_leaf> be an identification (i.e., an integer), <math_expression> be a mathematical expression with variable, constant or a valor. The labels of a leaf are defined by the following grammar:

<leaf_label> : [<expr_leaf_label>] [<elem_leaf_criterion>]
<expr_leaf_label> : (<expr_leaf_label_or>) <expr_leaf_label>
/ and <expr_leaf_label>
/ <simple_leaf_crit> <expr_leaf_label> / ε
<expr_leaf_label_or> : <simple_leaf_crit> <expr_leaf_label_or>
/ or <expr_leaf_label_or> / ε
<simple_leaf_crit> : <attribute> <op> <valor>
/ <geom_function> <op> <valor>
/ <multi_function> <attribute> <op> <valor>
<elem_leaf_criterion> : Group (<attribute> <foll_attribute>) Having (<expr_having>)/ε
<foll_attribute> : , <attribute> <foll_attribute> / ε
<expr_having> : (<expr_having_or>) <expr_having>
/ and <expr_having>
/ <having_criterion> <expr_having> / ε
<expr_having_or> : <having_criterion> <expr_having_or>
/ or <expr_having_or> / ε
<having_criterion> : <attribute> <op> <valor>
/ <multi_function> <attribute> <op> <valor>
<op> : < / > / = / ≠ / ≥ / ≤
<multi_function> : Sum / Min / Max / Avg / Count / Exist /Not_exist
<geom_function> : #_hole/connectivity/length/surface/perimeter/distance/angle

The constraints for a leaf are defined by the following grammar:

<leaf_constraint> : [<expr_leaf_const>]
<expr_leaf_const> : (<expr_leaf_const_and>) <expr_leaf_const>
/ or <expr_leaf_const>
/ <simple_leaf_const> <expr_leaf_const> / ε
<expr_leaf_const_and> : <simple_leaf_const> <expr_leaf_const_and>
/ and <expr_leaf_const_and> / ε
<simple_leaf_const> : <function> <op> <valor>/<geom_function> <op> <valor>
<function> : At_least / No_more / Exactly

The labels for the Thematic operators are defined by the following grammar:

<op_label_th> : [<expr_op_label_th>]
<expr_op_label_th> : (<expr_leaf_label>) ∧ (<expr_leaf_label>) <foll_expr_op_label_th>/ε
<foll_expr_op_label_th> : ∨ <expr_op_label_th> / ε

The constraints for the Thematic operators are defined by the following grammar:

<op_constraint_th> : [<expr_op_const_th>]
<expr_op_const_th> : (<expr_op_const_or_th>) <expr_op_const_th>
/ and <expr_op_const_th>
/ <simple_op_const_th> <expr_op_const_th> / ε
<expr_op_const_or_th> : <simple_op_const_th> <expr_op_const_or_th>
/ or <expr_op_const_or_th> / ε
<simple_op_const_th> : <function> <op> <valor>
/ <geom_function> <op> <valor>/<attribute> <op> <valor>

The labels for the Network operators are defined by the following grammar :

<op_label_net> : [<expr_op_label_net>]
<expr_op_label_net> : <re_net> / ε
<re_net> : <label_elem_net> <foll_label_elem_net>
/ (<re_net>) + <foll_label_elem_net>
/ [<label_elem_net> <foll_label_elem_net>]
<foll_label_elem_net> : ∧ <re_net> / ∨ <re_net> / ε
<label_elem_net> : { <expr_label_leaf>}

The constraints for the Network operators are defined by the following grammar:

<op_constraint_net> : [<expr_op_const_net>]
<expr_op_const_net> : (<expr_op_const_and_net>) <expr_op_const_net>
/ or <expr_op_const_net>
/ <simple_op_const_net> <expr_op_const_net> / ε
<expr_op_const_and_net> : <simple_op_const_net> <expr_op_const_and_net>
/ and <expr_op_const_and_net> / ε
<simple_op_const_net> : <function> <op> <valor> / length <op> <valor>
/ <aggregate> <op> <valor> / <order>
/ Avg (<attribute> <foll_attribute>) <op> <valor>
<aggregate> : <borne> (<element_net>) / <element_net>

<borne> : Min / Max
<element_net> : Sum (<math_expression>)
<foll_element_net> : , <math_expression> / ε
<order> : <attribute> [<math_expression>] <op>
<attribute> [<math_expression>]
[<valor> = <math_expression> .. <math_expression>]

The syntax of a query is defined by the following grammar:

<Query> : Combination (<expression> <elaborated_query>)
<elaborated_query > : , <expression> <elaborated_query> / ε
<expression> : ident_op <ident>
<operator> (<expression> <foll_expression>)
/ <argument>
<foll_expression> : , <expression> <foll_expression> / ε
<operator> : <user_operator> / <sys_operator>
<argument> : ident_leaf <ident>
label_leaf <leaf_label> constraint_leaf <leaf_constraint>
<user_operator> : <user_thematic_op>
label_op <op_label_th> constraint_op <op_constraint_th>
/ <user_network_op>
label_op <op_label_net> constraint_op <op_constraint_net>
<sys_operator> : <system_thematic_op>
label_op <op_label_th> constraint_op <op_constraint_th>
/ <system_network_op>
label_op <op_label_net> constraint_op <op_constraint_net>
<user_thematic_op> : ∩ /√ /C /¤
<system_thematic_op> : ∪ /Δ
<user_network_op> : ->
<system_network_op> : ΠD/-></>->/-><- /<</Πis_e/Πic_e /Πic_ne /Πis_n /Πic_n

3. 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 since they are more natural than textual query language and they offer a higher level of abstraction. Nevertheless, a

complex query may be difficult to express. Whenever many objects are involved and are interrelated, the visual representation may be confusing. Several problems are still under studies such as: the relative positions of the objects (i.e., north, south), the management of the results in case of ambiguities and multi-scaled data.

REFERENCES

[1] 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 n° 525

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

[3] 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

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

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

[6] 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