ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Jessie Kennedy and Peter Barclay (Eds)

Interfaces to Databases (IDS-3)

Proceedings of the 3rd International Workshop on Interfaces to
Databases, Napier University, Edinburgh, 8-10 July 1996

Paper:

A Survey of Query Languages for Geographic Information Systems

Marie-Aude Aufaure-Portier and Claude Trépied

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

A Survey of Query Languages for Geographic Information Systems

Marie-Aude Aufaure-Portier

Laboratoire d'Ingénierie des Systèmes d'Information
Université Lyon I, France
portier@if.insa-lyon.fr

Claude Trépied

Laboratoire d'Informatique, Ecole d'Ingénieurs en Informatique pour l'Industrie
Université de Tours, France
trepied@univ-tours.fr

Abstract

This paper presents a survey of query languages dedicated to Geographic Information Systems (GIS). Five categories of query languages can be identified : (1) natural languages, (2) extensions of SQL, (3) tabular languages : use of skeletons or forms, (4) graphical languages : use of symbols, which are only graphical conventions (like in the entity-relationship diagrams) and (5) visual languages : use of visual metaphors (e.g., icons, blackboard metaphor and map-overlay metaphor). The principles of each of them are presented. Respective strengths and weaknesses are pointed out, based on a set of predefined queries. The main objective of this paper is to present the state of the art of research work in a domain which is suitable for graphical querying. The last part of this paper describes hypermedia techniques.

1 Introduction

1.1 Problematics

Many research work have been recently done in the field of Geographic Information Systems especially for data storage, new indexing techniques, query optimization, ... [35]. One of the main characteristics of GIS is to store complex and huge amount of data. A fundamental research area is to define user-friendly interfaces for these systems. In most cases, GIS users are non-computer scientists (urbanists, architects). Various applications use geographic data such as land management, urban planning, tourism, vehicle guidance and navigation, ... Recently, cartographic applications have migrated towards citizen-oriented applications. This makes crucial the definition of simple and user-friendly interfaces.

Cartographic information can be graphically visualized (maps, pictograms, ...) using marketed GIS, but the languages which have been developed for queries and updates are very poor. In most cases, these languages have only been developed for one specific system. The consequence is that end-users applications cannot be supported by other systems. Another drawback is the complexity for a computer- illiterate user to develop applications.

However, the main characteristics of cartographic information is to be graphical. This implies that graphical languages are well suited for cartographic applications, according to the user's mental model. Many propositions of query languages for GIS have been made since the last decade. We can classify the propositions into three kinds : (1) *textual approaches* : natural language or extensions of SQL, (2) *non-textual approaches* : tabular, graphical or visual query languages and last (3) *hypermedia approach*. Natural Language approach seems to be the most suitable for the end-user. But, a serious difficulty of this approach is that many ambiguities must be solved. Another troublesome issue is that query formulation can be verbose and difficult (generally, a drawing is better than a long sentence). This approach can be seen as a good complement for graphical and visual approaches. Many extensions of the SQL language have been proposed [17, 23]. These extensions are necessary in order to allow Data Base Management Systems (DBMS) to store and retrieve spatial information. However, these languages are not suited to end-users because of the lack of conviviality of the SQL language. Tabular approaches [43, 45] are defined as extensions of QBE (Query-By-Example) [46]. The main difficulty is to express joins. Graphical languages [1, 42] make a better use of visual medium but underlying concepts are not perceived in a metaphorical way. Considering that cartographic information is visual, visual languages [4, 37] have been proposed. Some work have also be done to design new metaphors [24, 32, 44]. Finally, hypermaps [34] are integrated into hypermedia techniques [10, 11].

In this paper, we present examples of significant realisations for each kind of approach. The next section describes a toy database and queries to illustrate the presentation of each approach.

1.2 A Toy Database

The toy database concerns an urban planning application. This database contains information about land use, roads, public and private buildings, sewerage, pollution areas. Figure 1 describes this database example using the relational formalism. We assume that a data type called geometry has been defined in order to represent the geometric form of any spatial data (polygone, line or point).

PLOT	(#plot-number, owner-name, plot-address, #land-register-number, geometry)
ZONE-POS	(#plot-number, #zone-number)
ZONE	(#zone-number, type, geometry)
BUILDING	(#building-number, #plot-number, type, protected-zone, construction-zone, geometry)
HISTORICAL-MONUMENT	(#monument-number, name, address, geometry)
SEWERAGE	(#pipe-number, type, geometry)
PUBLIC_LANE	(#lane, type, covering, geometry)
POLLUTION	(#code, type, measure, geometry)

Figure 1 : Relational Schema of the Database Example

This database describes the plots of land within a city. Each plot of land is included into one or several zones which are defined into an administrative document called "Land Use Plan". The geometry of each zone is also described in this database. For each building is specified the plot on which it is built. A flag is defined in order to know whether the plot where the building stands is or not into a protected zone. Protected zones are defined by a distance of five hundred meters around historical monuments. The sewerage can be along a road or under a plot. Pollution areas (sound, industrial, ...) are described into a table with a specific attribute for the measure of pollution.

Queries addressed to a GIS may be of (1) alphanumerical type (retrieving information without using geometry), (2) geometrical or topological type and (3) deductive type. These queries may be very complex and, in most cases, involve the combination of several geometrical or topological operators (conjunction and/or disjunction of operators). To illustrate the different approaches, we propose three example queries. Q1 and Q2 involve geometrical operators. Q3 is a more complex query. We do not propose deductive queries because only a few propositions have been made to handle such queries.

We now define the example queries :

- Q1 : *What are the plots of urban zones situated in a protection zone of an historical monument ?*
 Q2 : *What are the urban zones including a natural zone ?*
 Q3 : *What are the plots included in an urban zone and in an agricultural zone, not located in a protection zone but near a sewerage, and where the sound pollution does not exceed 20 db ?*

2 Textual Languages

A difference can be made between *natural* languages (e.g., French, English and Japanese) which are already known by the general public, and *artificial* languages (e.g., SQL and ROSE Algebra [28]) which are only learned and used by database specialists. These two kinds of textual languages are usually used in written form in most of the database user interfaces but could also be used in oral form.

2.1 Natural Languages

Natural language appears as a natural tool for designing and using a database. In the process of query formulation it can obviously be used by end-user to express a query. But the system can also use it to paraphrase the database schema, the different interpretations of an ambiguous query [38] or the query result.

The idea to use natural language is not new, but until now it was not realistic : continuous interactions between user and system made the interface really unfriendly [16]. Important technological progress made this idea more popular now and an international workshop of applications of natural language to databases was created in 1995 (see [14]).

The main advantage of natural language is to eliminate the need to learn a new language. But at the present time it is often necessary to learn general strategies in order to interact with the interface [8]. For example, it is better

to use simple and direct queries than a literary style (interpretation process is easier) and to capitalise proper nouns if we type the query with lower cases.

Another good point of natural language is that some classical problems of query languages disappear or, at least, do not happen to the user but to the system. It is the case of mistakes made with Boolean operators. When using an artificial language, end-user often uses a word by word translation strategy to express his query [16] and consequently, translate an "and" of natural language by a conjunction (AND) even if this "and" expresses in fact a disjunction. The universal quantifier (ALL) and the negation seem also easier to express. But some concepts are still difficult to understand, such as the self-referential join [7].

A comparative study of three interfaces using a natural, an artificial and a graphical language respectively [8] shows that no interface is better. It depends on the query type to express. For example, the English of the DataTalker interface worked well for counting queries ("count...", "how many...") but not at all for multiple sorting because of the non-existence of the primary and secondary key concept in the common natural language.

Rather than using the natural language as an isolated means of querying database, it would be better to use it in a complementary way in an hybrid interface combining many kinds of languages. This is easily feasible in some prototypes as the KID system (Korean Interface for Databases) [14] which transforms a Korean natural language query into a graphical query based on an object-oriented data model.

Finally, it is important to note that a (written) natural language interface can use the direct manipulation paradigm. For example, the "drag and drop" of Apple's System 7 allows to transfer a part of text from a document to another. It is for why we do not use the direct manipulation as a discriminating criterion in our taxonomy of languages.

2.2 Extensions of SQL

Market GIS are based on architectures which does not provide a strong integration between alphanumerical and spatial data. Alphanumerical data are stored into a relational DBMS and graphical data are stored into files. These systems do not offer an integrated mechanism to formulate a query, and, in a few systems, consistency between alphanumerical and graphical data is not handled. Therefore, the architecture of market GIS is nowadays much more integrated but no standard query language for spatial data have been defined so far. New functionalities and running mechanisms have to be integrated into the SQL language. Many research propositions have been made to extend the SQL query language [3] in order to hold spatial data [17, 23, 25, 26, 33, 41]. These approaches are based upon the relational model. A new data type is introduced to model the geometry of spatial objects and new operators such as inclusion, intersection, ... are defined upon this data type. Operators can be defined as predicates and/or functions. Complex queries can not be expressed with query languages only based upon predicate operators. The main differences between all these approaches concern the data model and the degree of integration in the system. These works have been done before the normalization proposal [31]. Indeed, SQL3 contains extensions such as inheritance, encapsulation, modularity, definition of Abstract Data Types (ADT) [28] and hierarchy between tables. Two DBMS can be viewed as precursors for the integration of ADT : GéoSabrina [15] and Postgres [40]. We now present two approaches for the extension of SQL, GéoSabrina and Spatial SQL.

GéoSabrina [33] proposes a strong integration of spatial domains [18], spatial operators and indexing techniques into a relational DBMS [27]. A new spatial domain called "geometry" is defined. A geometry may be simple (one object) or complex (several objects as for example, a lake including an island). Predicates, functions and aggregates are defined on the geometric domain.

Let us see the expression of queries Q1 and Q2.

```

Q1 :      SELECT      *
          FROM        Plot
          WHERE        Exists (
                        SELECT      *
                        FROM        Zone
                        WHERE Zone.type = "URBAN"
                        And Incl (Zone.geometry,Plot.geometry))
          And
          Exists (
                        SELECT      *
                        FROM        Historical-Monument HM
                        WHERE Incl (Extend (HM.geometry, 500),
                                  Plot.geometry));

```

```

Q2 :      SELECT      *
          FROM        Zone Z1
          WHERE       type = "URBAN"
          And         Exists (
                    SELECT      *
                    FROM        Zone Z2
                    WHERE type = "NATURAL"
                    And         Incl (Z1.geometry, Z2.geometry));

```

The main contribution of this work was to define an extensible system. Indeed, the end-user can define new domains (or restrict existing domains) and new operators according to the needs of his own application. He can also define specific integrity constraints. But, the knowledge of the relational model is necessary to define domains or operators. Query optimization of spatial operators have been integrated into the DBMS.

Spatial SQL [23] is composed of a query language which specifies the information to retrieve and a presentation language which specifies how the information have to be displayed. The main originality of Spatial SQL is the presentation language which allows the end-user to specify graphical constraints such as the definition of a caption, a context, the modification of the display of geographic objects in terms of color, form,... [9]. Query results can be used to formulate a new query (incremental definition). The two languages are used separately so that the end-user will not define a graphical presentation if it is not necessary. If no presentation orders are specified, the result will be displayed according to the current state of the graphical environment. Four types of spatial data are defined : spatial_0, spatial_1, spatial_2 et spatial_3 repectively for points, lines, polygones and volumes. A set of spatial operators is also defined. An operator which allows users to formulate queries with reference to spatial objects on a screen is integrated to Spatial SQL (PICK: selection by mouse pointing operator).

The major concept of the Graphical Presentation Language GPL [22] is the graphical environment. The user query contains information about graphical display. This environment can be modified only by a GPL query. Graphical specifications are the following ones : (1) display mode which allows to combine results of several queries into a single picture, (2) visual variables (colors, symbols, etc.), (3) scale and display window, (4) context to allow the interpretation of the results.

Query Q1 can be expressed defining (1) the characteristics of the graphical display and (2) the query using the syntax of Spatial SQL. Graphical constraints defines the context and the caption for the query results.

SET LEGEND

COLOR black

FOR **SELECT** boundary (geometry)
FROM plot ;

SET LEGEND

COLOR blue

PATTERN dashed

FOR **SELECT** interior (geometry)
FROM zone
WHERE type = 'URBAN';

SET LEGEND

COLOR red

FOR **SELECT** interior (geometry)
FROM historical-monument ;

SET CONTEXT

FOR plot.geometry

SELECT historical-monument.geometry, zone.geometry, plot.name
FROM historical-monument, plot, zone;

We now define the query in Spatial SQL. The result will be displayed on the previously defined context.

```

SET MODE    highlight ;
SELECT      plot.geometry
FROM        plot, historical-monument, zone
WHERE       plot.geometry INSIDE zone.geometry
AND         zone.type = 'URBAN'
AND         distance (historical-monument.geometry, plot.geometry) < 500 ;

```

The originality of this approach is that the end-user can define and modify a display context. However, the definition of queries is not easy if the user have to often modify the graphical constraints. This definition have to be integrated into a graphical interface.

The main advantage of the extensions of SQL to spatial objects is that SQL is a standard and is based upon a formal model, i.e. the relational model.

The main drawback is that SQL suffers from a lack of conviviality and is difficult to learn for a end-user. Another trouble comes from the difficulty to integrate graphical specifications in order to display the results. Queries based on reasoning and qualitative results are not supported by this kind of approach [21]. The last limitation is that no standard spatial language is supported by marketed GIS.

In conclusion, extended spatial SQL approach seems to be the best formal support to retrieve spatial data but may be hidden into a graphical or visual language for the end-user. Some efforts of normalization have also to be carried out.

3 Non-Textual Languages

A *non-textual* language is defined as a language based on a privileged use of non-textual elements (e.g., form, diagram, icon and image [6]). But it is clear that (written or oral) textual elements are still necessary, at least to name entities (e.g., the cities).

We can distinguish three different approaches which correspond to the three generations of non-textual languages proposed in the literature since QBE : (1) *tabular* language : use of skeletons or forms; (2) *graphical* language : use of symbols, which are only graphical conventions (like in a entity-relationship diagram); (3) *visual* language : use of visual metaphors (e.g., icons, blackboard metaphor and map-overlay metaphor).

3.1 Tabular Languages

Two kinds of tabular languages can be distinguished depending on the representation used for the main concept (e.g., relation, object type and predicate) : a *skeleton* or a *form*.

QBE [46] is the precursor of skeleton-based languages. Each relation is represented by a two-dimensional skeleton in which the column headings are the name of the relation and the names of its attributes. To express a query, user fills the concerned skeletons with an example of a possible answer (variables and constants) and some keywords of the language (e.g., "P." indicates the desired output).

This tabular representation facilitates the user's task because this one does not have to generate his query independently of any structure, and more, he can enter the query elements in any order [16]. So QBE is more suitable to end-user (which usually is database-illiterate) than a textual language as SQL.

QBE has inspired a lot of languages the last of which is VQL (Visual Query Language) [45]. VQL has the advantage to be associated with many data models (relational, extended relational and object oriented models) and to be formally defined.

The query Q1 is expressed in VQL in Figure 2. The including skeleton (called "Output window") consists in a header and a body. The header indicates the desired output : the numbers of the wanted plots. The body contains five skeletons. The two first ones describe the object types "Plot" and "Historical-Monument". They are filled with example variables. The third skeleton indicates that z1 is a member of Z (set of the zones on which is situated the plot g1). The two last ones specify that z1 is an urban zone and that the wanted plot (g1) is inside a buffer zone of 500 meters around an historical monument (g2).

With technological evolution, and mainly the apparition of multiwindowing, form-based languages have been defined. An object type is no longer represented by a skeleton but by a form usually included in a dedicated window which can be closed or open. It is also possible to click on buttons or icons in order to see menus of commands, lists of predefined constants or menus of operators, and to choose one element of these menus by a simple click with the mouse. This kind of user interface is sometimes called WIMP (Window-Icon-Menu-Pointer) [24]. The querying philosophy is the same as before : user must fill the forms of the object types concerned by the query.

Figure 2 : VQL, a Skeleton-Based Tabular Language

OOQBE (Object-Oriented Query By Example) [43] is a good example of a form-based language. It is an extension of the relational QBE to an object-oriented data model. The query Q1 is expressed in Figure 3. The two main forms describe a plot and a monument. The example variable LINK-1 is used in order to specify that the plot must be inside a 500 meter buffer zone around a monument. When the value of an attribute is an object, it can be represented by a form in the foreground. As the attribute "zones" is multivalued, its value is represented by a stack of forms. This mechanism makes very easy the specification of a natural join. The operator " $\exists \cap$ " (non empty intersection) and the constant "urban" (predefined object) have been selected in sub-menus. They specify that the plot must be located in an urban zone (or at least, one part of the plot if it belongs to many zones).

☐ plot query form

☐ plot-number : _____

☐ owner-name : _____

☐ plot-address : _____

☐ geometry : INSIDE buffer(LINK-1,500)

☐ zones $\exists \cap$

☐ monument query form

☐ mt-number : _____

☐ name : _____

☐ address : _____

☐ geometry : LINK-1

☐ zone query form

☐ zone-number : _____

☐ type : urban

☐ geometry : _____

Figure 3 : OOQBE, a Form-Based Tabular Language

The PICQUERY+ language [13] is another example of tabular language which exists in two versions : the first one uses skeletons to visualise the predicates of the query ("tabular user interface") and the second one uses forms ("menu-icon PICQUERY+"). The last one clearly appears as the most user-friendly.

In conclusion, tabular languages are an important improvement on artificial languages as SQL. The tabular representation gives to end-user a framework to specify his query and allows him to do it in a free order.

Form-based languages are more user-friendly than skeleton-based ones. With multiwindowing, it is often possible to express a natural join by a simple click with the mouse. The use of menus first makes user always aware of the possible choices, and second, reduces the quantity of things to learn. The selection of an item from a menu avoids typing mistakes and is often perceived as more pleasant than typing from keyboard.

But there are still a lot of difficulties for end-user. For example, non-natural joins must be expressed by means of example variables (frequent source of mistakes) and the use of textual Boolean operators sometimes creates a

confusion with the natural language. Moreover, the tabular languages use a very poor visual representation of the data model concepts. This representation is not suitable for a spatial database where most of the entities are visual by nature.

3.2 Graphical Languages

A *graphical* language is based on the use of *symbols* representing the data model concepts. These symbols (e.g., rectangles, circles and arrows) are only pure graphical conventions, without any metaphorical power, and consequently need to be explained and memorised.

The language is usually associated with a semantic data model : SNAP [12] uses the IFO model, QBD* (Query By Diagram) [2] and SUPER [19] are based on the entity-relationship model and an extension of this one respectively, OHQL (Object Hypertext Query Language) [1] uses an object-oriented model.

The database schema is usually visualised by a graph where nodes represent the objects and arrows the relations between these objects. Entity-relationship diagrams are the most famous example of this kind of representation.

With the knowledge acquired by means of the schema browsing, end-user expresses his query following a mode which varies with the considered language :

- Either he builds a query graph in a separate window. This graph uses the symbols of the database schema, but he can also use some new conventions in order to visualise for example a selection predicate ("comparator arc" in SNAP) or to mark the elements which must be printed in the query result (inverse-video nodes, for example);
- Or end-user directly marks on the database schema graph which elements are concerned by the query, and then uses different menus to specify the selection criteria.

The complex query Q3 is expressed with the graphical language CANDID [42] in Figure 4. This language uses the first expression mode. As with QBE, user must describe an object which belongs to the solution of the query. The query graph cannot have more than two components (trees). Boolean operators are not available in a textual form and consequently a complex query is decomposed into subqueries, each of them corresponding to one step. Every retrieval query is in fact a reference to information derived from the database. It is the reason for that the query expressed in the Step Window (SW) is resumed by the system in the Recapitulation Window (RW) by means of a simplified graph mentioning the derived elements (object types or attributes). These elements could be used at any next step, which allows to express a complex query in an incremental way.

In Figure 4, the first step describes a plot located both in urban zone and in agricultural zone. The tag A corresponds to the set of zone types concerned by the plot. The combination of Venn diagrams specifies the inclusion of the constant set B in A. This type of wanted plot is called T1 in RW. The second step describes a plot located in a protected zone (let be the plot type T2). The next step combines the two first ones : the pointer specifies that we are looking for the T1 plots not belonging to the type T2. We have got the type T3. The fourth step specifies the proximity of a sewer (we suppose a fuzzy operator "near" exists). The last step specifies the inclusion in a specific polluted zone and indicates the desired output (the number of the plot).

1st step :

2nd step :

3rd step :

4th step :

5th step :

Figure 4 : CANDID, a Graphical Language

Graphical languages make a better use of visual medium than tabular languages. These ones come up against the scattering of semantically linked information in many separated elements (e.g., skeletons), when graphical languages use visual representations (e.g., arrows and diamonds) for semantic links. Besides, important efforts have been made to facilitate the expression of selection predicates : e.g., use of Venn diagrams and comparator arcs.

The major drawback is that graphical languages use a lot of symbols which are only graphical conventions. Consequently, underlying concepts are not perceived in a metaphorical way, i.e. the conventions must be explained and memorised. In conclusion, the semantic distance between the real word and the database universe is still too important for end-user.

3.3 Visual Languages

Visual languages use *metaphors* to show the concepts. Metaphors take into account the mental model of the end-user [20]. This approach have been developped successfully in the last years for many reasons. The first reason is the evolution of the categories of users (specialists to citizen). The second reason is that software designers take into consideration the user's requirements in terms of conviviality and ease of use. Visual languages offer an intuitive and incremental view of spatial queries but lack from a poor expressive power, execution inefficiency and multiple interpretations for a query. We can distinguish two different approaches to design a visual language: (1) the end-user draws a pattern using a set of static icons and (2) the end-user makes a drawing directly on the screen using the blackboard metaphor. The first approach is illustrated by the Cigales language [5, 36] and the second one by the *Sketch!* language [37]. We now describe these two approaches and give the advantages and limitations of visual languages.

The main objectives of Cigales [4] are : (1) to provide a natural and user-friendly query language for GIS users, (2) to support applications on several paltforms and (3) to allow the end-user to define his query by successive refinements. The basic idea is that the user draws a pattern according to the data he wants to be displayed and to his mental view of the application [39]. The main idea is to query against the database using a query by example philosophy [46].

Figure 5 : Cigales Visual Languages

A Cigales query is expressed by combining a set of static icons which represent spatial objects and operators that can be applied upon these objects like inclusion, intersection, etc. The graphical query is then compiled into a functional language [30]. This intermediate language guarantees (1) the support of any platform for the application and (2) the possibility to use again intermediate results to define a query by successive refinements. The graphical editor is made of two parts : a working window in which an elementary query (using only one operator) appears and a window in which the final query is built step by step. The end-user has first to define the operands and then applies an operator by selecting an icon. Then, the system creates the composition of the operands and the operator. The semantics of spatial objects are given by the end-user (navigation in the data model using pop-up menus). This semantics must be represented by metaphors for a more natural comprehension. Figure 5 represents the expression of query Q2 using the Cigales language.

The *Sketch!* language [37] is based on the blackboard metaphor. The user makes a drawing directly on the screen and then, the system makes the interpretation of this query. Spatial operators are deduced from the drawing. The alphanumerical part of the query is expressed graphically using the entity-relation model. The semantics of spatial objects are given using menus. The query is then translated step by step into a logical expression. The main advantage of such an approach is that the user has no constraint to express a query and does not have any new language to learn. Figure 6 represents the expression of query Q1 using the *Sketch!* language.

Figure 6 : *Sketch!* Visual Language

These languages can be seen as precursors for visual querying in the GIS domain application and provide two different approaches. The main contribution is that computer-illiterate users can express queries with an intuitive manner. These languages also permit the visual definition of spatial views. Nevertheless, many limitations still remain.

- (1) The main limitation comes from the *ambiguities* of visual languages but may be seen differently in the two approaches previously described.

In the Cigales language, the composition of operands and operators is predefined by the system and the ambiguities appears when the query contains more than one operator. Hence, the user cannot express exactly the query he wants, only by using the static icons. A few number of ambiguities (for example, in Figure 7) can be easily solved but need more manipulations from the end-user.

Figure 7 : Example of Ambiguity

Two interpretations come from this drawing : (1) either r intersects a and a intersects b or (2) a intersects b and r intersects a but does not intersect b . How can we solve these ambiguities ? A solution is to dialog with the end-user but it may be too long and verbose. Another solution is to force default semantics that the user can explicitly modify using specific manipulations. These manipulations may be very difficult in the case of complex queries.

In the *Sketch!* language, ambiguities come from multiple interpretations for a query. Another reason is that the drawing of the end-user may not correspond to the real world and then generates a wrong interpretation. The end-user may also not remember easily the spatial operators and needs help from the system to build a query.

- (2) Alphanumerical and spatial data are not uniformly supported. A proposal was made by [37] to display these data in separate windows in order to avoid ambiguities between alphanumerical and spatial data. A drawback is that the formalism used for spatial and non-spatial data is not the same. The consequence is that building a query may be more difficult for the end-user.
- (3) A few operators have no graphic equivalence like operators used for reasoning and deduction.
- (4) Temporal objects are not taken into account in these languages. For example, a spatial object can be modified (as, for example a river) and consequently, may have an impact on the representation of other objects.

- (5) Negation and disjunction have no graphical representation. We have to find metaphors to suggest the representation of such operators.
- (6) A few relations are difficult to express like "don't care" relations. For example, if a intersects b and b intersects c, the user has to specify in the drawing, the relation between a and c (either a intersects c or not). In a textual language, if the user don't care about the relation between a and c, he does not specify anything else. In this case, a can or not intersect c.

In conclusion, we can say that visual languages offer more conviviality. It is obviously easier to express a query using an intuitive way rather than a query language like SQL which is difficult to learn. Visual languages also allow an incremental definition of queries. But many problems subsist and have to be solved in order to offer a better expressive power to the end-user. These problems seems to let appear that limits of "all graphic" or "all visual" have been reached. A promising research way is to integrate visual and textual specificities in order to avoid complex manipulations. Some efforts have to be done to formalize visual languages. The expressive power of visual languages is not yet known.

4 Hypermedia Approach

It is an extension of hypertext approach with multimedia data (e.g., text, image and sound). The information base is structured as a network in which *nodes* represent the application objects (e.g., a text describing an historical monument, the map of a building and the aerial picture of a plot) and *links* represent the relationships (structural or semantic) between nodes. In an *hypermap* [34], a node can also represent a spatial object, and consequently, has a spatial position (e.g., coordinates). When displaying the contents of a node on the screen, each link is visualised by an *anchor* (e.g., icon and underlined word) which represents in fact the target node of the link. A click on an anchor produces the display of the target node. It is the reason why we call *navigation* this access mode to the information base.

The main advantage of hypermedia systems is that the user does not have to learn a new language. Navigation is a very easy task if anchors have been well designed, i.e. if they are intuitive visual representations of corresponding nodes. Then, the end-user is really aware of the possible choices and can quickly decide which links he wants to use.

The inherent drawback of this approach is that only available (i.e., predefined) links can be used. It means that none of queries Q1 to Q3 could be expressed if a link to the answer does not already exist. So this approach cannot really be considered as a query language but as a complementary tool which could be used, for instance, to define the area of interest to which an upcoming query will refer.

In conclusion, this access mode to information is very attractive but not sufficient for end-user who also needs analytic functionalities [10]. A very promising solution could be to combine visual language and hypermedia approaches [11] and to integrate them into a GIS environment.

5 Conclusion

In this paper we have presented a complete survey of approaches for querying geographical databases. Natural language appears as an attractive approach for end-user but also presents some important drawbacks (e.g., verbosity and ambiguities) and consequently, can only be used as a complement to another approach.

About the extensions of SQL, it is clear that they can only be used by people having some knowledge in database theory. Concerning the hypermedia approach, it allows a user-friendly browsing of the information base but cannot be considered as a query language and must also be combined with another approach. Visual languages seems to be the more promising approach but still have some limitations : absence of formalisation, semantic ambiguities, limited expressive power and no description of temporal aspects of the objects.

Consequently, the ideal approach could be a sort of hybrid interface combining harmoniously visual language, natural language and hypermedia approach, but for the time being, it is clear that each of them needs complementary research works.

6 References

1. ANDONOFF E., MENDIBOURNE C., MORIN C., ROUGIER V., ZURFLUH G., 1993, OHQL : un langage graphique pour l'interrogation d'une base de données, Ingénierie des systèmes d'information, vol. 1, n°2, pp 203-227.
2. ANGELACCIO M., CATARCI T., SANTUCCI G., 1990, QBD* : A Graphical Query Language with Recursion, IEEE Transactions on Software Engineering, vol. 16, n°10, pp 1150-1163.

3. ANSI, 1986, X3.135-1986 American National Standard Database Language SQL, American National Standards Institute.
4. AUFAURE-PORTIER M.A., 1992, CIGALES : un langage graphique d'interrogation de Systèmes d'Information Géographiques, Phd Thesis, University of Paris VI.
5. AUFAURE-PORTIER M.A., 1995, A High-Level Interface Language for GIS, *Journal of Visual Languages and Computing*, vol 6 (2), pp 167-182, Academic Press.
6. BATINI C., CATARCI T., COSTABILE M.F., LEVIALDI S., 1993, On Visual Representations for Database Query Systems, *Actes des journées "L'interface des mondes réels et virtuels"*, Montpellier, France, pp 273-283.
7. BELL J.E., ROWE L.A., 1992, An Exploratory Study of Ad Hoc Query Languages to Databases, *Proc. of 8th Int. Conf. on Data Engineering*, Tempe, Arizona, USA, pp 606-613.
8. BELL J.E., 1992, The Experiences of New Users of a Natural Language Interface to a Relational Database in a Controlled Setting, *Proc. of the first Int. Workshop on Interfaces to Database Systems*, Glasgow, UK, Ed. R. Cooper, Springer-Verlag 1993, pp 433-454.
9. BERTIN J., 1983, *Semiology of Graphics*, Madison, University of Wisconsin Press.
10. BIEBER M., KACMAR C., 1995, Designing Hypertext Support for Computational Applications, *Communications of the ACM*, vol. 38, n°8, pp 99-107.
11. BOURSIER P., MAINGUENAUD M., 1992, Spatial Query Languages : Extended SQL vs. Visual Languages vs. Hypermaps, *Proc. of the 5th Int. Symp. on Spatial Data Handling*, Charleston SC, USA, pp 249-259.
12. BRYCE D., HULL R., 1986, SNAP : a Graphics-Based Schema Manager, *Proc. of the 2nd IEEE Int. Conf. on Data Engineering*, IEEE, New-York, USA, pp 151-164.
13. CARDENAS A.F., IEONG I.T., TAIRA R.K., BARKER R., BREANT C.M., 1993, The Knowledge-Based Object-Oriented PICQUERY+ Language, *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, n°4, pp 644-657.
14. CHAE J., LEE S., 1995, Natural Language Query Processing in Korean Interface for Object-Oriented Databases, *Proc. of the first Int. Workshop on Applications of Natural Language to Data Bases*, Versailles, France, pp 81-94.
15. CHRETIEN D., LARUE T., PASTRE D., LEGOFF R., VIEMONT Y., 1994, The Geosabrina Design: the Way to build a GIS over a Spatial Data Server, *22th Annual ACM Computer Science Conference (ACM/SAC'94) GIS Track*, Phoenix, USA, pp 328-332.
16. CORSON Y., 1983, Ergonomie des langages de requête relationnels, *Technique et Sciences Informatiques*, vol. 2, n°5, pp 329-339.
17. COSTAGLIOLA G., TORTORA G., TUCCI M., BUSILLO M., 1995, GISQL - A Query Language Interpreter for Geographical Information Systems, *IFIP Third Working Conference on Visual Database Systems*, Lausanne, Switzerland, pp 247-258.
18. DAVID B., 1991, *Modélisation, représentation et gestion d'information géographique*, Phd Thesis, University of Paris VI.
19. DENNEBOUY Y., ANDERSSON M., AUDDINO A., DUPONT Y., FONTANA E., GENTILE M., SPACCAPIETRA S., 1995, SUPER : Visual Interfaces for Object + Relationship Data Models, *Journal of Visual Languages and Computing*, vol. 5, pp 73-99.
20. DOWNS R.M., STEA D., 1977, *Maps in Minds, Reflections on Cognitive Mapping*, Harper and Row Series in Geography.
21. EGENHOFER M., 1991, Why not SQL !, *International Journal of Geographic Information Systems*, vol 6 (2), pp 71-85.
22. EGENHOFER M., 1991, Extending SQL for Cartographic Display, *Cartography and Geographic Information Systems*, vol 18 (4), pp 230-245.

23. EGENHOFER M., 1994, Spatial SQL : A Query and Presentation Language, IEEE Transactions on Knowledge and Data Engineering, vol 6 (1), pp 86-95.
24. EGENHOFER M.J., BRUNS H.T., 1995, Visual Map Algebra : A Direct-Manipulation User Interface for GIS, Proc. of VDB-3 (Third Working Conf. on Visual Database Systems, IFIP 2.6), Lausanne, Switzerland, pp 211-226.
25. FALOUTSOS C., ROUSSOPOULOS N., SELLIS T., 1988, An Efficient Pictorial Data Base System for PSQL, IEEE T.O.S.E., vol 14 (5).
26. FRANK A., 1982, MAPQUERY : Database Query Language for Retrieval of Geometric Data and their Graphical Representation, Computer Graphics, vol 16 (3).
27. GARDARIN G., CHEINEY J.P., KIERNAN G., PASTRE D., STORA H., 1989, Managing Complex Objects in an Extensible Relational DBMS, Proceedings of the 15th International Conference on Very Large Data Bases, Amsterdam (Netherlands), pp 55-65.
28. GÜTING R. H., SCHNEIDER M., 1995, Realm-Based Spatial Data Types: The ROSE Algebra, The VLDB Journal, vol. 4, n°2, pp 243-286.
29. GUTTAG J., 1977, Abstract Data Types and the development of data structures, Communications of the ACM, vol 20, pp 396-403.
30. HUDAK P., 1989, Conception, Evolution and Application of Functional Programming Languages, ACM Computing Surveys, vol 21 (3).
31. ISO/IEC JTC1/SC21/WG3 DBL-SEL3b, 1990
32. KUHN W., 1993, Questions and Answers about Metaphors for GIS User Interfaces, Cognitive Aspects of Human-Computer Interaction for Geographic Information Systems, T. Nyerges, D. Mark, R. Laurini & M. Egenhofer ed., pp 113-122.
33. LARUE T., PASTRE D., VIEMONT Y., 1993, Strong Integration of Spatial Domains and Operators in a Relational Database System, Proceedings of the 3rd Symposium on Spatial Databases (SSD'93), Singapour, Lecture Notes in Computer Science, n°692, Springer Verlag ed., pp 53-72.
34. LAURINI R., MILLERET-RAFFORT F., 1990, Principles of Geometric Hypermaps, 4th International Symposium on Spatial Data Handling, Zurich, Switzerland, pp 642-651.
35. LAURINI R., THOMPSON D., 1992, Fundamentals of Spatial Information Systems, The APIC series, Academic Press.
36. MAINGUENAUD M., PORTIER M.A., 1990, CIGALES : a Graphical Query Language for Geographical Information Systems, 4th Int. Symposium on Spatial Data Handling, Zurich (Suisse), vol 1, pp 393-404.
37. MEYER B., 1992, Beyond Icons : Towards New Metaphors for Visual Query Languages for Spatial Information Systems, Proceedings of the first International Workshop on Interfaces to Database Systems, Ed. R. Cooper, Springer-Verlag 1993, Glasgow (UK), pp 113-135.
38. MIRANDA S., LE THANH N., LAKHAL L., 1985, Un générateur de langage pseudo-naturel pour les interfaces utilisateurs du projet BAOU, Bases d'information généralisées, Christment C.Y. and al., Dunod Informatique, pp 41-61.
39. NORMAN D.A., 1983, Some Observations on Mental Models, Mental models, Gentner and Stevens ed., Hillsdale, New-Jersey, Lawrence Erlbaum.
40. ROWE L., STONEBRAKER M., 1987, The Postgres Data Model, Proceedings of the 13th International Conference on Very Large Data Bases, Brighton, UK, edited by P. Stocker and W. Kent, pp 83-96.
41. SACK-DAVIS R., MCDONNELL K.J., OOI B.C., 1987, GEOQL : A Query Language for Geographic Information Systems, Australian and New Zealand Association for the Advancement of Science Congress, Townsville.
42. SCHNEIDER M., TREPIED C., 1989, Graphical Description and Query Interface for Databases, Proceedings of the 4th IFIP Working Conference on Engineering for Human-Computer Interaction, Napa Valley, California, USA, pp 63-84.

43. STAES F., TARANTINO L., TIEMS A., 1991, A Graphical Query Language for Object Oriented Databases, Proc. of 1991 IEEE Workshop on Visual Languages, Kobe, Japan, pp 205-210.
44. TREPIED C., 1995, Dynamic Icons for a Visual Spatio-Temporal Query Language, Proc. of the Join European Conference and Exhibition on Geographical Information, The Hague, The Netherlands, pp 84-89.
45. VADAPARTY K., ASLANDOGAN Y.A., OZSOYOGLU G., 1993, Towards a Unified Visual Database Access, Proc. of SIGMOD'93 (Special Interest Group on Management of Data), Washington, Ed. P. Buneman and S. Jajodia, SIGMOD Record, vol. 22, issue 2, pp 357-366.
46. ZLOOF M.M., 1977, Query-by-Example : A Database Language, IBM Systems Journal, vol 16 (4), pp 324-343.