A Data-Flow Approach to Visual Querying in Large Spatial Databases

Andrew J. Morris[1], Alia I. Abdelmoty[2], Baher A. El-Geresy[1] and Christopher B. Jones[2]

[1] School of Computing, University of Glamorgan, Treforest, Wales, CF37 1DL, UK
[2] Department of Computer Science, Cardiff University, Cardiff, Wales, CF24 3XF, UK

Abstract. In this paper a visual approach to querying in large spatial databases is presented. A diagrammatic technique utilising a data flow metaphor is used to express different kinds of spatial and non-spatial constraints. Basic filters are designed to represent the various types of queries in such systems. Icons for different types of spatial relations are used to denote the filters. Different granularities of the relations are presented in a hierarchical fashion when selecting the spatial constraints. The language constructs are presented in detail and examples are used to demonstrate the expressiveness of the approach in representing different kinds of queries, including spatial joins and composite spatial queries.

1 Introduction

Large spatial databases such as, Computer Aided Design and Manufacture (CAD/CAM), Geographic Information Systems (GIS) and medical and biological databases, are characterised by the need to represent and manipulate a large number of spatial objects and spatial relationships. Unlike, traditional databases, most concepts in those systems have spatial representations and are therefore naturally represented using a visual approach. GIS are a major example of spatial databases with a large number of application domains, including environmental, transportation and utility mapping. Geographic objects, usually stored in the form of maps, may be complex formed by grouping other features and may have more than one spatial representation which changes over time. For example, a road object can be represented by a set of lines forming its edges or by a set of areas between its boundaries. Users of current GIS are expected to be non-experts in the geographic domain as well as possibly casual users of database systems. Alternative design strategies for query interfaces, besides the traditional command-line interfaces, are sought to produce more effective GIS and to enhance their usability.

The current generation of GIS have mostly textual interfaces or menu-driven ones that allow some enhanced expression of the textual queries [Ege91]. Problems with textual query languages have long been recognised [Gou93] including the need to know the structure of the database schema before writing a query as well as problems of semantic and syntactic errors. Problems are compounded in a geographic database where geographic features can be represented by more than one geometric representation and the semantics and granularity of spatial relations may differ across systems and application domains.

In this paper, the focus is primarily on the process of query formulation. A visual approach is proposed to facilitate query expression in those systems. The approach addresses some of the basic manipulation issues, namely, the explicit representation of the spatial types of geographic features and the qualitative representation of spatial relationships. A diagrammatic technique is designed around the concept of a filter to represent constraints and implemented using direct manipulation. Filters, represented by icons, denote spatial and non-spatial constraints. Spatial constraints are computed through the application of spatial operators on one spatial entity, e.g. calculating the area of polygon, or on more than one spatial entity, e.g. testing whether a point object is inside a polygon object. Different granularities of binary spatial filters are used and may be defined in the language, for example, a general line-cross-area relationship may be specialised to indicate the number of points the two objects share etc. The concept of a filter is used consistently to construct complex queries from any number of sub-queries. The aim is to provide a methodology for a non-expert user to formulate and read relatively complex queries in spatial databases. Notations are used to distinguish query (and sub-query) results, to provide means of storing query history as well as to provide a mechanism for query reuse. A prototype of the approach has been implemented and evaluation experiments are currently underway. GIS are the main examples used in this paper. However, the approach proposed may be applied to other types of spatial databases.

The paper is structured as follows. Section 2 lists some general requirements and problems identified for query interfaces to spatial databases. A discussion of related work is presented in section 3. In section 4, the data flow approach is first described and the language constructs are then presented in detail. This is followed in section 5 by an overview of the implementation and evaluation of the produced interface, concluding with a summary in section 6.

2 General Requirements and Identified Problems

Several issues related to the design of query interfaces to spatial databases are identified as follows. Some of these issues can be addressed at the language design level, while others need to be addressed at the implementation level of the query interface. Issues arising due to the spatial nature of the database include,

Representation of Spatial Objects: Geographic objects have associated spatial representations to define their shape and size. Objects may be associated with more than one spatial representation in the database to handle different map scales or different application needs. Spatial representations of objects determine and limit the types of spatial relationships that they may be involved in. Explicit representation of the geometric type(s) of geographic features is needed to allow the user to express appropriate constraints over their locations.

Spatial operations and joins: It is difficult for a non-expert user to realise all the possible spatial operations that may be applied on a geographic object or the possible spatial relationships that may be computed over sets of geographic objects. The semantics of the operations and relationships are implicit in their names. Those names may not have unique meanings for all users and are dependent on their implementation in the specific system in use. For example, an overlap relationship between two regions may be generalised to encompass the inside relationship in one implementation or may be specific to only mean partial coverage in another as shown in figure 1. In this paper a visual, qualitative, representation of spatial operations and relationships is proposed to facilitate their direct recognition and correct use. Also, different granularities of spatial relationships need to be explicitly defined to express different levels of coarse and detailed spatial constraints.

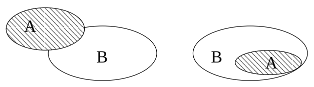


Fig. 1. Types of overlap relationship between two spatial regions.

Composite spatial constraints: Multiple spatial constraints are used in query expressions. Again, the semantics of the composite relation may be vague, especially when combined using binary logical operators of *And* and *Or*. Means of visualising composite spatial relations would therefore be useful. E.g. "Object1 is north-of Object2 and close to it but outside a buffer of 10 m. from Object3".

Self spatial joins: Problems with the expression of self joins were noted earlier in traditional databases [Wel85]. The same is true in spatial databases but complicated with the use of spatial constraints in the join. E.g. "Find all the roads that intersect type A roads?"

Query History: Visualising results of sub-queries during the process of query formulation is useful as users tend to create new queries by reworking a previous query or using parts thereof and so suggests the inclusion of query history.

Other general database issues include, parenthesis complexity when specifying the order of Boolean operators with parentheses as the query grows [Wel85, JC88, MGP98]. Also, problems when using Boolean logic operators of *And* & *Or* as well as common syntactic errors such as, omitting quotation marks around data values where required [Wel85] and applying numeric operators to non-numeric fields.

The approach proposed in this paper attempts to handle some of the above issues that can be addressed at the language design level. Other issues are left to the implementation stage of the query interface.

3 Related Work

Querying interfaces to GIS can be broadly categorised between textual interfaces and non-textual interfaces. Several text-based extensions to SQL have been proposed (e.g. [Ege91, IP87, RS99]). Spatial extensions to SQL inherit the same problems of textual query languages to traditional databases. Typing commands can be tiring and error prone [EB95], with difficult syntax that is tedious to use [Ege97]. In [Gou93] it was noted that users can spend more time thinking about command tools than thinking of the task that they have set out to complete.

The Query-by-Example model [Zlo77] has also been explored in several works. QPE [CF80] and PICQUERY [JC88] are examples of such extensions. Users formulate queries by entering examples of possible results into appropriate columns on empty tables of the relations to be considered. Form-based extensions often do not release the user from having to perform complicated operations in expressing the queries nor from having to understand the schema structure. Also, complex queries usually need to be typed into a condition box that is similar to the WHERE clause of an SQL statement.

Visual languages have been defined as languages that support the systematic use of visual expressions to convey meaning [Cha90]. A great deal of work is already being carried out to devise such languages for traditional and object-oriented databases in an attempt to bridge the gap of usability for users. Iconic, diagrammatic, graph-based and multi-modal approaches are noted. Lee and Chin [LC95] proposed an iconic language, where icons are used to represent objects and processes. A query is expressed by building an iconic diagram of a spatial configuration. Difficulties with this approach arise from the fact that objects in a query expression need to be explicitly specified along with their associated class and attributes, which renders the language cumbersome for the casual user [Ege97].

Sketch-based languages are interesting examples of the visual approach. In the CIGALES system proposed by Mainguenaud and Portier [MP90], users are allowed to sketch a query by first selecting an icon of a spatial relationship and then drawing the query in the "working area" of the interface. LVIS is an extension to CIGALES [PB99] where an attempt is made to provide the functionality of a query language. Egenhofer [Ege97] and Blaser [Bla98] have also proposed a sketch-based approach where a sketch of the query is drawn by the user and interpreted by the system. A set of query results is presented to the user including exact and near matches. Sketch-based approaches are suitable for expressing similarity-based queries to spatial databases and can become complex to use in a general context when composite queries are built. Also, they either assume that users are able to sketch a query and express spatial relationships in a drawing or rely on different modalities for offering the user guidance in developing the sketch. Exact queries can be generally ambiguous due to several possible interpretations of the visual representation

4 Language Description

Query diagrams are constructed using filters, represented by icons, between data input and output elements. Queries are visualised by a flow of information that may be filtered or refined. The approach is based on, but substantially modifies and extends an early example of a filter flow metaphor proposed by Young and Shneiderman [YS93]. In [YS93] a single relation was used over which users could select the attributes to constrain. The metaphor of water flowing through a series of pipes was used and the layout of the pipes indicated the binary logic operators of And and Or. Line thickness was used to illustrate the amount of flow, or data, passing through the pipes and attribute menus were displayed on the lines to indicate the constraints. Join operations were not expressed in [YS93] nor were there indications to means of handling query results. The idea was simply presented using one relation as input. The idea was later used by Murray et al [MPG98] to devise a visual approach to querying object-oriented databases.

In this paper, the basic idea of data flow between data source and results is utilised. The concept of a filter between both source and result is introduced to indicate the type of constraint expressed, whether non-spatial or spatial as well as the type of the spatial constraint in the later case. Spatial and non-spatial join operations are also expressed consistently. Graphical notations for intermediate query results are used to allow for tracing query histories and reuse of queries (and sub-queries). In what follows the query constructs are described in detail.

4.1 Database Schema

Consider the following object classes to be used as an example schema.

```
County (cname:string, geometry:polygon, area:float, population:integer,
        other-geometry: point)
Town (tname:string, geometry:polygon, area:float, town-twin:string,
      tpopulation:integer, county:county)
Road (rname:string, geometry:line, rtype:string, rcounty:string, rsurface:string)
Supermarket (sname:string,  geometry:point, town:string, onroad:string)
```

In figure 2, object classes are depicted using a rectangular box containing the name of the class and an icon representing its spatial data type, whether point, line, polygon or any other composite spatial data type defined in the database, e.g. a network. This offers the user initial knowledge of the spatial representation associated with the feature. A thick edge on the icon box is used if the object has more than one spatial representation in the database. Switching between representations is possible by clicking on the icon box. For example, a County object is represented by a polygon to depict its actual shape and by a point for manipulation on smaller scale maps. All other information pertaining to the class is accessible when the user selects the class and then chooses to view its attributes. At this point we are not primarily concerned about how the database schema is depicted, but we focus on the aspect of query visualisation.

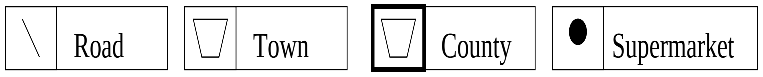


Fig. 2. Example Schema. The basic spatial representation of the objects is depicted in the icons.

As queries are constructed, the extent of the class chosen as input to the query will flow through filters to be refined according to the constraints placed on it. Results from a query or a sub query contain the new filtered extents, and these can be used to provide access to the intermediate results as well as final results of a query or as input to other sections of the query.

A basic query skeleton consists of a data input and data output elements and a filter in between. Every input object will have a related result object that can be displayed in the case of spatial objects.

4.2 Filters

Filters or constraints in a query are made on the non-spatial (aspatial) properties of the feature as well as on the spatial properties and location of the feature. Hence, two general icons are used to represent both types of filters as shown in figure 3. Figure 3(a) demonstrates a non-spatial filter depicted by an A (for (stored) Attributes) symbol and figure 3(b) demonstrates a spatial filter depicted by the "coordinates" symbol. The non-spatial filter represents constraints over the stored attributes and the location filter represents constraints that need to be computed over the spatial location of the object. After indicating the type of filter requested, the specific condition that the filter represents is built up dynamically by guiding the user to choose from menus of attributes, operators and values and the condition is then stored with the filter and may be displayed beside the icon as shown in the figure. Several filters may be used together to build more complex conditions and queries as will be shown in the following examples.

4.3 Query Results

The initial type of the data is defined by the extent that flows into the query. It is this type that will be passed along the data flow diagram, depicted by downward pointing arrows to the results. The type of the flow is not altered by the query constraints. The only way the type of flow can be altered is when it flows into a results box. The results of the query are depicted, as shown in figure 3(b), by a double-edged rectangular box with the class name along with any particular attributes selected to appear in the results. By default the result of the query is displayed if the object has a spatial representation. The results box can be examined at any time of query formulation and its content displayed as a map and/or by listing the resulting object properties. If none of the attributes has been selected for listing, then the default is to view all the attributes of the class.

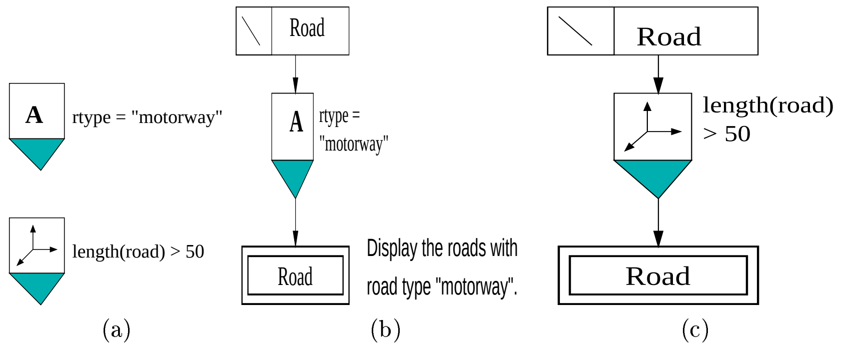


Fig. 3. a) An aspatial filter and a spatial filter. b) Depicting query results. "Select All From Road Where Road.rtype = 'motorway' ". c) A spatial filter in a simple query construct.

An English expression of the query producing the result box is also available for examination through the result box as shown in the figure.

4.4 Simple Query Constructs

The example in figure 3 demonstrates a simple filter to restrict the results based on a non-spatial condition. Other operators may be used, e.g.=, >, <, like, etc. Also, spatial (unary) operators may be used to filter the results, e.g. area, volume, perimeter/boundary, etc. An example of using a spatial filter is shown in figure 3(c).

Simple queries may be combined using Boolean expressions. Figure 4 represents the different cases. The flow will pass through only when the constraint is satisfied. In figure 4(a), multiple constraints are shown in series to represent constraints joined by *And*. The flow will pass through only when both constraints are satisfied. In figure 4(b), parallel arrangement of the filters is used to indicate that the flow will pass through when either or both constraints are satisfied. Any number of constraints may be joined together by binary logic operators as shown in the example in figure 4(c) where three constraints are used.

Negated constraints are depicted by the filters in figure 5(a). *Not* may be applied to individual constraints or to a group of constraints. Filters may be joined in any order as explained. An example of a query with negated constraints is shown in figure 5(b).

4.5 Joins

Two kinds of join operations are possible in spatial databases namely, non-spatial joins and spatial joins. Both types are represented coherently in the language. Spatial joins are expressions of spatial relationships between spatial objects in

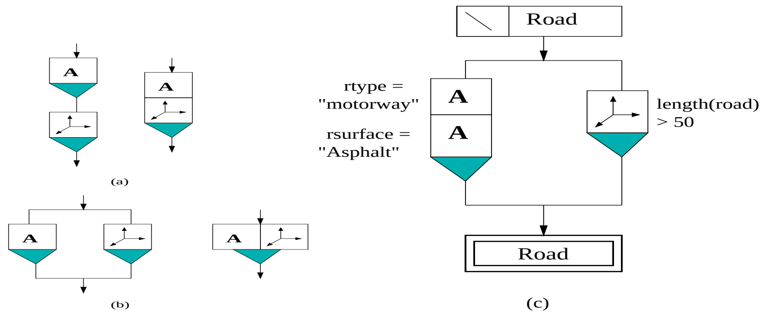


Fig. 4. (a) Filters joined by *And*. (b) Filters joined by *Or*. (c) Visualisation of multiple filters. "Display all the motorway roads with asphalt road surface or all the roads whose length is > 50."

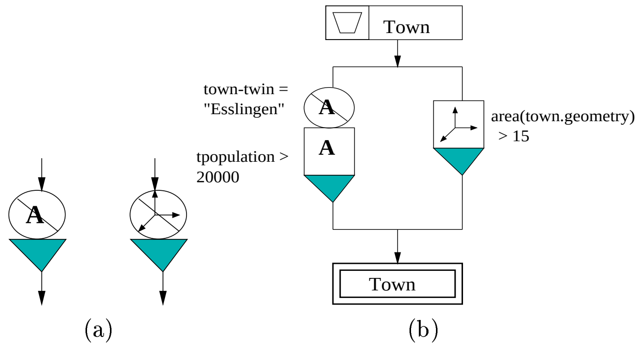


Fig. 5. (a) Negation of non-spatial and spatial filters. (b) Visualisation of the *And, Or* and *Not* operators.

the database. Examples of spatial join queries are: Display all the motorway objects crossing Mid Glamorgan, and Display all the towns north of Cardiff within South Glamorgan.

Filter notations are modified to indicate the join operation as shown in figure 6(a) and (b). A join filter is associated with more than one object type. A result box is associated with every joined object class and linked to the join filter. An example of a spatial join query is shown in figure 6(c). The query finds all the motorway roads that cross counties with population more than 50,000. Note that the result box from the join operation has been modified to reflect the contents of the join table. More than one object type has been produced, in this case, roads and counties that satisfy the join condition will be displayed on the result map.

A symbol of the spatial relationship sought is used to replace the "coordinate" symbol in the spatial join filter. A choice of possible spatial joins is available de-

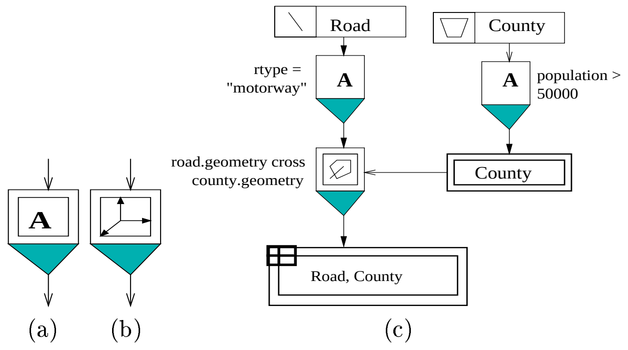


Fig. 6. (a) Non-Spatial join filter. (b) Spatial join filter (c) Example query of a spatial join. Specific relationship icon replaces general spatial join to indicate the cross relationship.

pending on the spatial data types of the objects joined. In the last example, all the possible relationships between line (for roads) and polygons (for counties) will be available. Spatial relationships may be classified between topological, directional and proximal. Relationships are grouped in hierarchical fashion to allow the use of finer granularities of relationships. Examples of hierarchies of topological and directional relationships are shown in figure 7. Qualitative proximal relationships, such as near and far are vague unless they explicitly reflect a pre-defined range of measures. Hence, using proximal relationships requires an indication of the measure of proximity required, e.g. within a distance of x m.

Multiple spatial joins may be expressed similarly either with the same object type, e.g. to find the supermarkets outside and north of towns, or with more than one object type, e.g. to find the supermarkets north of towns and within a buffer of 5 km. from motorways as shown in figure 8.

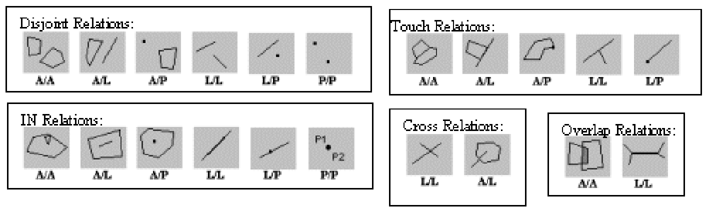


Fig. 7. Examples of symbols for some spatial relationships [CFO93]; (A) for area, (L) for line and (P) for point.

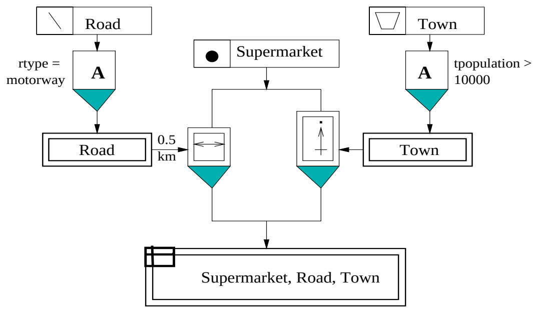


Fig. 8. Composite query. Find the supermarkets within a buffer of 0.5 km of a motorway or are outside and north-of a town whose population is greater than 10000.

5 Implementation

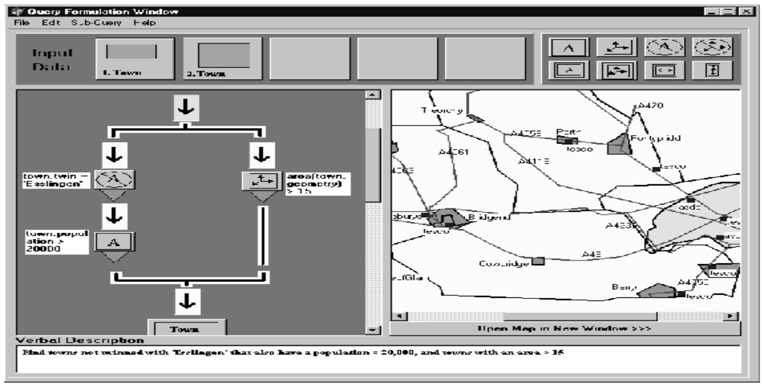
So far, the proposed language has been described independently of its implementation. In this section, an outline of the interface prototype to the language is presented. The implementation of the interface aims to address some of the issues relating to schema visualisation, structuring of query results, operator assistance in general, including guided query expression, feedback and restriction of user choice to valid options during query formulation.

A prototype of the interface is implemented in Delphi. A test spatial data set is stored in a relational database, linked to the query interface. The query interface window is shown in figure 9. Input data sets are selected in a Schema visualisation window. The query is formulated, in a guided fashion, using a collection of filters, including, spatial, aspatial, negated and various types of spatial join filters. The interfaces is context-sensitive and allows only possible filters and choices to be presented to the user at the different stages of query formulation. An spatial-SQL interpretation of the flow diagram is produced and compiled to produce the result data set presented on the result window.

Evaluation tests for both the language and interface have been designed and are being conducted using two categories of users, namely, users with some experience of using a GIS systems and users with no prior knowledge of GIS. The evaluation test for the language makes use of the "PICTIVE" approach [Mul93] where the language elements are simulated using Post-It notes and a whiteboard.

6 Conclusions

In this paper a visual approach to querying spatial databases is proposed. Examples from the GIS domain have been used throughout to demonstrate the expressiveness of the language. The design of the language tried to address several requirements and problems associated with query interfaces to spatial databases. The following is a summary of the design aspects.



Find towns not twinned with 'Esslingen' that also have a population = 20,000, and towns with an area > 15

Fig. 9. The query Formulation Window.

- Icons were used to represent the geographic features with explicit indication of their underlying spatial representation, thus offering the user a direct indication to the data type being manipulated.
- A data flow metaphor is used consistently to describe different types of query conditions namely, non-spatial and spatial constraints as well as negated constraints and spatial and non-spatial joins.
- Concise representation of the metaphor was used to join multiple constraints when dealing with one object in join operations.
- Intermediate results are preserved and could be queried at any point of the query formulation process and hence the query history is also preserved.
- Nested and complex queries are built consistently.

The consistent use of the metaphor is intended to simplify the learning process for the user and should make the query expression process easier and the query expression more readable. The approach is aimed at casual and non expert users, or at expert domain users who are not familiar with query languages to databases. The implementation of the language aims to cater for different levels of user expertise. Visual queries are parsed and translated to extended SQL queries that are linked to a GIS for evaluation.

References

[Bla98] A. Blaser. Geo-Spatial Sketches, Technical Report. Technical report, National Centre of Geographical Information Analysis: University of Maine, Orono, 1998.

[CF80] N.S. Chang and K.S. Fu. Query-by-Pictorial Example. *IEEE Transactions on Software Engineering*, 6(6):519–24, 1980.

[CFO93] E. Clementini, P.D. Felice, and P.V. Oosterom. A Small Set of Formal Topological Relationships for End-User Interaction. In *Advances in Spatial Databases - Third International Symposium, SSD'93*, pages 277–295. Springer Verlag, 1993.

[Cha90] S.K. Chang. *Principles of Visual Programming Systems.* Englewood Cliffs: Prentice Hall, 1990.

[EB95] M.J. Egenhofer and H.T. Burns. Visual Map Algebra: a direct-manipulation user interface for GIS. In *Proceedings of the Third IFIP 2.6 Working Conference on Visual Database Systems 3*, pages 235–253. Chapman and Hall, 1995.

[Ege91] M.J. Egenhofer. Extending SQL for cartographic display . *Cartography and Geographical Information Systems*, 18(4):230–245, 1991.

[Ege97] M.J. Egenhofer. Query Processing in Spatial Query by Sketch . *Journal of Visual Languages and Computing*, 8:403–424, 1997.

[Gou93] M. Gould. Two Views of the Interface. In D. Medyckyj-Scott and H.M. Hearnshaw, editors, *Human Factors in GIS*, pages 101–110. Bellhaven Press, 1993.

[IP87] K. Ingram and W. Phillips. Geographic information processing using an SQL based query language. In *Proceedings of AUTO-CARTO 8*, pages 326–335, 1987.

[JC88] T. Joseph and A.F. Cardena. PICQUERY: A High Level Query Language for Pictorial Database Management. *IEEE Transactions on Software Engineering*, 14(5):630–638, 1988.

[LC95] Y.C. Lee and F.L. Chin. An Iconic Query Language for Topological Relationships in GIS. *International Journal of Geographical Information Systems*, 9(1):24–46, 1995.

[MGP98] N. Murray, C. Goble, and N. Paton. Kaleidoscape: A 3D Environment for Querying ODMG Compliant Databases. In *Proceedings of Visual Databases 4*, pages 85–101. Chapman and Hall, 1998.

[MP90] M. Mainguenaud and M.A. Portier. CIGALES: A Graphical Query Language for Geographical Information Systems. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 393–404. Univerity of Zurich, Switzerland, 1990.

[MPG98] N. Murray, N. Paton, and C. Goble. Kaleidoquery: A Visual Query Language for Object Databases . In *Proceedings of Advanced Visual Interfaces*, pages 247–257. ACM Press, 1998.

[Mul93] M. Muller. PICTIVE: Democratizing the Dynamics of the Design Session. In *Participatory Design: Principles and Practices*, pages 211–237. Lawrence Erlbaum Associates, 1993.

[PB99] M.A.A. Portier and C. Bonhomme. A High Level Visual Language for Spatial Data Management. In *Proceedings of Visual '99*, pages 325–332. Springer Verlag, 1999.

[RS99] S. Ravada and J. Sharma. Oracle8i Spatial: Experiences with Extensible Database . In *SSD'99*, pages 355–359. Springer Verlag, 1999.

[Wel85] C. Welty. Correcting User Errors in SQL. *International Journal of Man-machine studies*, 22:463–477, 1985.

[YS93] D. Young and B. Shneiderman. A Graphical Filter/Flow Representation of Boolean Queries: A Prototype Implementation and Evaluation. *Journal of the American Society for Information Science*, 44(6):327–339, 1993.

[Zlo77] M.M. Zloof. Query-by-Example: A Database Language . *IBM Systems Journal*, 16(4):324–343, 1977.

This article was processed using the LaTeX macro package with LLNCS style