Visualizing Spatial Query Results: The Limitations of SQL

M. Egenhofer and W. Kuhn

Visual Database Systems II, Budapest, Hungary

Visualizing Spatial Query Results: The Limitations of SQL*

Max J. Egenhofer and Werner Kuhn

National Center for Geographic Information and Analysis and
Department of Surveying Engineering
Boardman Hall, University of Maine, Orono, ME 04469
max@mecan1.maine.edu, kuhn@mecan1.maine.edu

Abstract

Several extensions to the relational database query language SQL have been proposed to serve as a *spatial query language*; however, they do not sufficiently address how to visualize query results. This paper investigates the requirements for an *ad hoc* language describing the graphical representation of spatial query results from the perspective of a geographic information system with frequent map output and assesses several spatial SQL extensions with respect to their treatment of the graphical representation. It concludes that the SQL framework is inappropriate for this task at the user interface.

1 Introduction

The Structured Query Language SQL is enjoying much popularity in the database world and has become the standard for relational database management systems [ANSI 1986]. A crucial criterion for evaluating the usability of a query language for a non-standard application domain is: "How useful are the database operations provided by the query language for the particular application?" [Schek 1988]. The following list of typical queries in a geographic database will help to assess the usability of SQL as a spatial query language, with an emphasis on the graphical display of the query result:

- "Display a map of the State of Maine."

*Work of the authors was partially funded by grants from Digital Equipment Corporation under Sponsored Research Agreement No. 414, TP-765536 and BW-213800, and Intergraph Corporation (Principal Investigator: Andrew U. Franki). Additional support from NSF for the NCGIA under grant number SES 88-10917 is gratefully acknowledged.
• “Remove all second-order roads.”
• “Where is Orono?”
• “Change all ‘red disks’ to the symbol ‘blue triangle’.”
• “Differentiate between cities with a population of more or less than 50,000.”
• “What is this?”—and the user points to some place on the screen map.

Spatial queries refer to particular spatial concepts [Guenther 1990] in a spatial data model [Frank 1991c], e.g., geometric objects with a complex internal structure, spatial relationships to select objects of interest, complex graphical display, and selection by pointing. Users base their spatial queries upon such concepts, which are critical criteria in a spatial query language; however, neither SQL nor the relational data model support them and the use of a pure relational query language makes the users simulate spatial concepts in terms of the few, predefined non-spatial concepts [Westlake 1990]. Relations lack the proper set of fundamental operations to properly support the majority of geometric and pictorial operations [Joseph 1988] and relational operations per se are insufficient to solve some typical queries in a geographic database [Frank 1982, Egenhofer 1988b]. For example, important problems, such as queries over combined maps, cannot be solved unless specific support of spatial operations is provided [Laurini 1988] and spatial data are frequently selected upon their spatial location, rather than by references to some values; therefore, it has been concluded that so-called non-standard applications [Härder 1985], such as CAD/CAM, VLSI design, image databases, or geographic information systems, require a more complex data model than tables.

A number of different SQL extensions have been proposed [Roussopoulos 1985, Siketer 1985, Ingram 1987, Egenhofer 1987, Egenhofer 1988a, Herring 1988, Roussopoulos 1988, Ooi 1989, Raper 1991]. These spatial SQL dialects differ considerably in (1) the extent they cover spatial properties, (2) the degree at which they formally define the semantics of the extensions, (3) their syntactical implementations, and (4) the degree at which they comply with the standardized SQL structure. At the same time, they all struggle with incorporating spatial concepts into a framework designed for data modeled as 2-dimensional tables. Similar problems have been observed for spatial query languages as extension of other relational database query languages, such as Query-by-Pictorial-Example [Chiang 1980], GEO-QUEL [Berman 1977], and a Quel extension for map production [Ehrlich 1988].

This paper argues that these problems are due to SQL’s inappropriateness as a framework for a high-level spatial query language, because it insufficiently supports the graphical representation of query results. The functional capabilities of SQL to formulate spatial queries have been analyzed elsewhere [Egenhofer 1991a]. Our findings are based upon the results of extensive studies into the requirements for spatial query languages [Frank 1984b, Egenhofer 1988b, Egenhofer 1990a], and the design [Frank 1982, Frank 1984a, Egenhofer 1991b], and implementation [Egenhofer 1984, Egenhofer 1987, Egenhofer 1990b] of prototypes.
The remainder of this paper is structured as follows: a brief description of SQL in Section 2 is followed by a discussion of the impediments of the SQL framework for a high-level, interactive spatial query language. Section 4 analyzes how various SQL dialects try to incorporate the treatment of spatial data. The conclusions in Section 5 propose to integrate better the interaction between user and database.

2 SQL

SQL [Chamberlin 1976] is an implementation of the five relational operations selection, projection, Cartesian product, set union, and set difference [Codd 1970]—plus a few useful extensions for operations upon tuples such as aggregate functions and views. The syntax framework is the SELECT-FROM-WHERE clause which corresponds to the relational operations of projection, Cartesian product, and selection, respectively. Set union and set difference can be explicitly formulated among several SQL queries. For example, given two relations parcel and road with the respective attributes parcel.owner, parcel.own, parcel.roadname, road.name, and road.width, the following SQL query is to retrieve the owners of all parcels which are located on roads narrower than 15 feet:

```sql
SELECT parcel.owner 
FROM parcel, road 
WHERE road.width < 15 and 
    parcel.roadname = road.name;
```

The query result is a relation which is in an interactive environment by default presented in form of a table on the screen. More extensive discussions of SQL can be found in database textbooks [Date 1986, Kerth 1986, Ullman 1988] and SQL tutorials [Date 1989].

The focus of this paper will be exclusively on the role of SQL as a query language in its literate sense and any other roles, e.g., as a manipulation language, will not be discussed. To further restrict the focus, only one of the many interpretations of a query language will be considered. Currently, the use of SQL as a query language is overloaded and too many different tasks are performed with SQL. Some users consider it as a high-level interface language to tie a database management system into an application program; others use SQL as an interactive ad hoc query language; for another group of users, SQL is the data exchange language to transfer data from one database to another. Unfortunately, very different requirements exist for each of these roles of a query language and their integration into a single, universal query language adds another dimension to an already complex problem. It is difficult. For example, the interface between database and application program must fit well into a high-level programming language, while an ad hoc query language must consider the interaction between humans and computers. In order to avoid further confusions of different requirements and expectations, the discussions in this paper are limited to the use of SQL as an interactive ad hoc spatial query language.
3 Impediments of the SQL Framework for a Spatial Query Language

Any SQL extension which attempts to comply with the SQL standard has to live with the conceptual and syntactical constraints set forth by the standardized framework. This section will show why it is so difficult to extend standard SQL so that it becomes a useful language to describe the retrieval and visualization of spatial data. A number of shortcomings of the SQL framework for spatial data handling will be examined, some of which are due to the attempt of extending a given standardized query language.

3.1 SQL Framework

Standard SQL is targeted to retrieving data modeled in a tabular form. It stresses the functionality of formulating complex and powerful queries through the implementation of the relational algebra operations and, at the same time, disregards how to present the query result to the user. Only a single, default representation is provided—the display in the form of a table. This uneven balance between retrieval and display is characterized by two features of SQL:

- The SELECT clause is overloaded with the projection of the attributes onto the resulting relation and the implied tabular representation of each query result. Such a combination may be appropriate for those applications which always require the results to be presented as tables only; however, it is a major impediment for a query language with renderings other than tables, e.g., drawings, or with a choice of renderings [Egenhofer 1987].

- The retrieval language is decoupled from the representation of query results. SQL neither memorizes what has been displayed nor does it allow the users to formulate queries with respect to the query results displayed.

These shortcomings are most apparent when users request to display query results in a non-tabular format.

3.2 Graphical Display

An ad hoc spatial query language requires that query results can be displayed graphically. The graphical display of query results has two separate components in a query language: (1) specifying that the query result (or parts of it) should be graphically displayed—as opposed to the implied tabular representation of alphanumeric data in SQL—and (2) describing how to represent the query result.

The potentials for graphical variations of objects on drawings are much greater than those for manipulating the frame of a table, e.g., by adding headers over the columns or changing the sequence of columns. Graphical display involves the use of different colors, patterns, symbols, etc. [Bertin 1983]. What makes the display specifications difficult is that a set of these visual variables may be assigned to the entire result of a query as well as to specific spatial objects or classes of objects in the result. Especially for
geographic applications with high-quality map output, these display specifications are too complex to be integrated into the actual query statement.

At a first glance, it may appear as a viable solution to separate a spatial query into two parts: (1) the instruction to retrieve the data wanted and (2) the subsequent command to display the query result. The query language—or better the retrieval language—would describe what to retrieve and a display language would specify how to graphically represent the query result previously retrieved; however, such a separation does not take into account that the query and specification of graphical representation frequently depend upon each other. For instance, the following instructions are to retrieve all roads in Penobscot county which should be drawn such that the major roads will be displayed with a different line style than the remaining roads. The query result is the relation of all roads in Penobscot county.

```
SELECT road.geometry
FROM road, county
WHERE county.name = "Penobscot" and
road.geometry INSIDE district.geometry
```

In order to draw the roads according to their importance, further information is necessary. In essence, another query must be asked to separate the intermediate result—the geometry of all roads—into two sets of roads so that the elements of each set can be represented with the same visual attributes; therefore, the demand for alternative displays is not solved by just drawing the spatial attributes in the SELECT clause. It must also be considered that the graphical display of query results may require more information from the database than what has been provided by the query result.

3.3 Modifying the Content of a Graphical Rendering

Spatial query results may be depicted as drawings at which users look; however, a more dynamic interaction with query results is also necessary [Egenhofer 1988b]. Users want to refer to the current drawing by asking further questions about it or they want to modify the current drawing, e.g., by adding further information to it or by removing information displayed [Egenhofer 1990a]. Such a dynamic interaction may be richer than the interaction with tables and requires support from the query language. The SQL framework supports only the retrieval of data based upon input typed by the user and the presentation of the data retrieved for the user, and no provisions have been made for an alternative retrieval, i.e., one partially based on the currently displayed result. Extensions of the standardized SQL framework to incorporate multiple representation types are essentially impossible, unless changes are made to a degree that the extension is no longer compatible with the standard.

3.4 Modifying the Graphical Representation

Users of visual databases typically work on with their query results. Besides visually analyzing them, they frequently modify the representation of the currently displayed
objects—without updating the database. Such representation changes may be purely
graphic (“Replace color ‘red’ by ‘green’.”) or they may involve additional information
from the database (“Replace the symbols of the cities within 20 miles of an airport by
a red disk and keep the symbols of the other cities.”). These instructions are similar to
queries—actually, they contain elements of a database query—and must be processed
as such.

4 Spatial SQL Dialects

Several proposals have been made to turn SQL into a spatial query language, e.g.,
a spatial SQL dialect [Sikeler 1985], the pictorial query language PSQL [Roussopoulos
1985, Roussopoulos 1988], Spatial SQL [Egenhofer 1987, Egenhofer 1989b, Egen-
hofer 1991b], GEOQL [Ooi 1989, Ooi 1990], SQL-SX [Raper 1991], and the SQL-
based query languages for the geographic information systems KGIS [Ingram 1987]
and TGRIS [Herring 1988]. The most significant extensions will be reviewed subse-
sequently to provide the reader an overview of what has been accomplished, but also to
show how the individual extensions proposed differ.

4.1 Spatial Data Type

Almost every spatial SQL extends the domains of the relational calculus with spatial
data types. Although there have been attempts to design a spatial query language
exclusively based on the standard domains in relational calculus [Go 1975, Berman
1977], i.e., on integers, reals, and character strings, it is generally agreed upon that users
of spatial query languages need a high-level abstraction of spatial data for the multitude
of spatial data models, e.g., raster and vector, which have significantly different spatial
properties that must be accounted for in a spatial language [Frank 1991a]. An attribute
over such a spatial data type will be referred to as a spatial attribute and a relation
with a spatial attribute will be called a spatial relation.

What varies among the different spatial SQL dialects is (1) the spatial data model for
which the data types are used and (2) the degree at which this extension is integrated
into the SQL environment. The variety of spatial data types proposed for spatial SQLs
includes

- a universal spatial data type [Sikeler 1985, Herring 1988, Ooi 1989];
- data types for each spatial dimension, e.g., points, nodes, lines, polylines, surfaces,
  and volumes [Raper 1991], and their generalizations to a dimension-independent
  spatial superclass [Egenhofer 1988a]—essentially, the link to the universal spatial
data type above;
- a number of data types for a multitude of spatial properties such as area, perimeter,
  and length [Ingram 1987] and the graphical representation as well; and
- a data type for bitmap graphics [Roussopoulos 1988].
4.2 Spatial Relationships

The extension with spatial data types is useless unless the pertinent operations and relationships are also defined. Spatial relationships are boolean operations to check whether or not a particular predicate holds true between tuples of two or more spatial relations. The variety of spatial relationships necessary in a geographic query language includes qualitative concepts, such as directions and topology, which are absent in any standard query language. The semantics of the relationships differs considerably among the various spatial query languages [Egenhofer 1989b, Guenther 1990] and a pure list of operation names and their parameters [Raper 1991] treats only syntactical problems. More important is a formal definition of the semantics of the operations and their combinations. Though there have been attempts to formalize some particular subsets, e.g., directions [Peuquet 1987, Frank 1991b] or topological relations [Egenhofer 1989a, Egenhofer 1990b, Egenhofer 1991c], there exists currently no set of formal definitions for spatial relationships.

SQL tends to overload predicates, i.e., an operation may have multiple implementations and the system selects one depending on the type(s) of the argument(s) of the predicate. While such an overloading of SQL's standard predicates is sufficient for some applications, e.g., temporal relationships [Allen 1983, Sarda 1990], it is insufficient for spatial relationships. The set of standard predicates in SQL is too small to cover all spatial relationships [Egenhofer 1989a, Egenhofer 1990c]. For example, SQL lacks predicates for topological concepts, such as neighbor and intersect; therefore, syntax extensions are necessary to allow users to formulate queries with spatial predicates. Similar to the extension with a spatial data type, all spatial SQL dialects include such spatial operations and spatial relations.

4.3 Specification of Graphical Display

- All spatial attributes in the SELECT clause are displayed, while non-spatial parts of the query result are shown as alphanumeric tables [Ingram 1987].

- A qualifier for a spatial attribute in the SELECT clause specifies that a particular relation be graphically represented [Silik 1985].

- The query result is displayed according to the status set in a display environment [Egenhofer 1991b].

The description of the graphical display of the query result has drawn only little attention. Most spatial query languages disregard this part or use only a default representation. Only PSQL and Spatial SQL propose solutions for this problem.

- PSQL displays query results according to predefined definitions, called picture lists [Rousopoulos 1985]—without providing a language to create or modify a picture list.
• Spatial SQL displays spatial query results according the definitions in the _graphical representation environment_ [Egenhofer 1991b]. A comprehensive display language as a superset of SQL allows users to describe the use of colors, patterns, symbols, etc. for spatial relations.

• GKS graphic commands are appended to an SQL query [Goh 1989], a treatment similar to the Mapquery [Frank 1982], where the output specification has been bundled with the actual user query. Such a combination increases dramatically the length of any query with non-standard output.

4.4 Selection by Pointing

Each SQL query is a stand-alone instruction without any reference to the previously asked queries or their results. Likewise, query results are always represented as a single "rendering" and no interaction with the currently displayed result is possible. For a spatial language with graphical representation of query results, this SQL feature is a major restriction, because the graphical representation of query results animates users to refer to the drawings when formulating further queries.

Since SQL has no provisions for input other than typed characters, some spatial SQL dialects include an operator to identify a spatial object by pointing to its spatial location on the screen. Most commonly, pointing is implemented as a keyword, e.g., MOUSE [Ingram 1987] and PICK [Egenhofer 1988a], though the use of a spatial function named CURSOR () has been proposed as well [Raper 1991]. Such references may occur in WHERE clauses when a user refers to this object and uses a pointing device to select the object from the screen drawing.

4.5 Compliance with the Syntax Framework

Any SQL extension suffers from the dilemma of preserving the standardized form of SQL while extending SQL's functionality. A variety of syntax modifications of the SELECT-FROM-WHERE framework have been proposed in order to enable users to query spatial data. Three fundamentally different trends can be found:

• the addition of new clauses in which particular spatial properties are addressed, such as WITH LOCATION [Skeier 1985] and AT [Roussopoulos 1985] for spatial conditions, and ON [Roussopoulos 1988] to specify an output format;

• the modification of the SQL framework by treating relations in lieu of attributes both in the SELECT and WHERE clauses and cancelling the FROM clause which becomes superfluous [Herring 1988]; and

• minimal extensions within the given framework to comply with standard SQL [Ingram 1987, Egenhofer 1988a, Ooi 1989] and the definition of a display environment outside of SQL [Egenhofer 1991b].
5 Conclusion

Despite numerous attempts to extend SQL with various spatial features, no satisfying solutions for an SQL-based interactive spatial query language have been found. The major deficiency of any spatial SQL is the severe difficulty to incorporate graphical display and its specification into SQL.

Visual databases need query languages that are more powerful and better-suited than SQL or an extension of it. SQL per se is already difficult to use [Reisner 1981] and any addition to the SQL concepts increases its complexity. New, high-level languages are necessary to provide for appropriate interaction between user and system. The design of such a language must start at the user level by investigating what kinds of operations users want to perform upon spatial databases and how they do it [Egenhofer 1988b, Pizano 1989, Mainguenaud 1990]. In the complex environment of a visual database, “queries” are part of a dynamic process during which users request information with respect to the currently visible information and make modifications in the information displayed. In order to support such a working behavior, visual database interface languages are necessary which are based upon cognitive and mental models such as image schemas and metaphors [Lakoff 1980] applied to geographic data [Kuhn 1991].

Acknowledgements

The authors appreciate discussions with participants at the NATO Advanced Research Institute on “Cognitive and Linguistic Aspects of Geographic Space” in Las Navas, Spain, July 1990.

References


