M. Egenhofer
Why not SQL!
International Journal of Geographical Information Systems 6 (2): 71-85, 1992.

Why not SQL!

MAX J. EGENHOFER

National Center for Geographic Information and Analysis and
Department of Surveying Engineering, Boardman Hall,
University of Maine, Orono, ME 04469, U.S.A.

Abstract. The application of traditional database query languages, primarily the Structured Query Language SQL, for geographical information systems (GIS) and other non-standard database applications has been tried unsuccessfully; therefore, several extensions to the relational database query language SQL have been proposed to serve as a spatial query language. It is argued that the SQL framework is inappropriate for an interactive query language for a GIS and an extended SQL is at best a short term solution. Any spatial SQL dialect has a number of serious deficiencies, particularly the patches to incorporate the necessary spatial concepts into SQL.

1. Introduction

Structured Query Language SQL is enjoying much popularity in the database world and has become the standard for relational database management systems (ANSI 1989). Designed as a high level interface to relational database management systems (Chamberlin *et al.* 1976) to manipulate tables (Codd 1970), SQL has been successfully used as a database interface for applications that can be easily expressed in terms of tables. For so-called *non-standard applications* (Härder and Reuter 1985), such as geographical information systems (GIS), CAD/CAM, very large scale integration (VLSI) design, or image databases, this restricted data model, based on the underlying power of relational calculus, has proved to be insufficient. Relations lack the proper set of fundamental operations to support properly most geometric and pictorial operations (Joseph and Cardenas 1988) and relational operations *per se* are insufficient to solve some typical GIS queries (Frank 1982, Egenhofer and Frank 1988 b). For example, queries over combined maps cannot be solved unless specific support of spatial operations is provided (Laurini and Milleret 1988). To overcome some of the shortcomings, numerous specialized extensions of SQL have been proposed to deal with complex objects (Lorie and Schek 1988, Mitschang 1989), and temporal (Date 1988, Ariav 1986, Sarda 1990) and spatial data (Roussopoulos and Leifker 1985, Sikeler 1985, Ingram and Phillips 1987, Herring *et al.* 1988, Roussopoulos *et al.* 1988, Ooi *et al.* 1989, Egenhofer 1991, Raper and Bundock 1991).

This paper investigates the usefulness of the SQL framework for geographical applications. A criterion for evaluating the suitability 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). Obviously, spatial queries refer to particular *spatial concepts* (Egenhofer and Frank 1988 b, Guenther and Buchmann 1990) in a spatial data model (Frank in press), e.g. geometrical objects with a complex internal structure, spatial relationships to select objects of interest, complex graphical display and selection by pointing. Users of GIS base their spatial queries on such concepts and, therefore, they are critical in any query language for a GIS. The following list of typical queries in a GIS environment will help to assess the usefulness of SQL as a GIS query language:

- 'Display a map of the State of Maine.'
- 'What is the shortest path from Bangor International Airport to Orono?'
- 'Where is the nearest gas station?'
- 'In which direction is Mount Desert Island?'
- 'What is this?'—and the user points to some place on a screen with some graphical rendering displayed.

Currently, there is little consensus among researchers about the application of SQL for spatial data, as shown by the great number of different SQL extensions. These spatial SQL dialects differ considerably in: (1) the extent that 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 to 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 modelled as tables. Similar problems have been observed for spatial query languages as extensions of other relational database query languages such as Query-by-Pictorial-Example (Chang and Fu 1980), GEO-QUEL (Berman and Stonebraker 1977), and a Quel extension for map production (Ehrich *et al.* 1988).

This paper argues that these problems are due to the inappropriateness of SQL as a framework for high level query languages for a GIS. It is not discussed whether or not it is possible to express some spatial queries in SQL. SQL can definitely be used for querying any data modelled in a tabular format and spatial data can be mapped onto such tables (Waugh and Healey 1987, Abel 1988, Lorie 1991); however, such a representation is not the most natural form for modelling spatial data and leads to unnecessarily complex queries. Neither the relational data model nor SQL support high level spatial concepts and the use of a pure relational query language makes users simulate them in terms of the few, predefined non-spatial concepts (Westlake and Kleinschmidt 1990). In comparing database query languages with programming languages, such a low level treatment of spatial data is comparable with breaking a RECORD type in a high level programming language into physical address assignments in Assembler language.

The scope of this paper is to underline this position with a number of arguments showing severe shortcomings in the idea of extending the SQL framework to *the* GIS query language (Raper and Bundock 1991). Our findings are based on the results of extensive studies into the requirements for spatial query languages (Frank 1984 b, Pullar 1987, Egenhofer and Frank 1988 a, b, Egenhofer 1990) and the design (Frank 1982, 1984 a, Egenhofer 1991) and implementation of prototypes (Egenhofer 1984, 1989 b, Egenhofer and Frank 1990).

The remainder of this paper is structured as follows: a brief description of SQL in Section 2 is followed by a review of how various spatial extensions have been implemented in several different SQL dialects, stressing the diversity of the approaches. The impediments of the SQL framework for a high level, interactive spatial query language are analysed in Section 4. The conclusions in Section 5 propose the design of interface languages for GIS, in lieu of query languages, to integrate better the interaction between the user and the database.

2. SQL

SQL (Chamberlin *et al.* 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 on 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 multiple SQL queries). For example, given two relations **parcel** and **road** with the respective attributes **parcel.number**, **parcel.owner**, **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.

```
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 as a table on the screen. Similarly structured SQL commands also allow users to update relations. More extensive discusssions of SQL can be found in database textbooks (Date 1986, Korth and Silberschatz 1986, Ullman 1988) and SQL tutorials (Date and White 1989).

The focus of this paper will be exclusively on the role of SQL as a *query language* in its literal sense, and any other roles, e.g., as a *manipulation language* to update databases, will not be discussed. To restrict the focus further, 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 it. 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 *intergalactic dataspeak* (Stonebraker *et al.* 1990), i.e. the data exchange language to transfer data among databases. 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. For example, the interface between database and application program must fit well into a high level programming language, whereas an *ad hoc* query language must consider the interaction between humans and computers. It is doubtful whether a single language may serve these diverse purposes. 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. Spatial SQL dialects

Several proposals have been made to turn SQL into a spatial query language, e.g. the pictorial query language PSQL (Roussopoulos and Leifker 1985, Roussopoulos *et al.* 1988), Spatial SQL (Egenhofer 1987, 1989 b, 1991), GEOQL (Ooi *et al.* 1989, Ooi 1990), SQL-SX (Raper and Bundock 1991) and the SQL based GIS query languages for KGIS (Ingram and Phillips 1987) and TIGRIS (Herring *et al.* 1988). The most significant extensions will be reviewed to provide an overview of what has been accomplished, but also to show how the individual extensions proposed differ.

3.1. *Spatial data type*

Almost every spatial SQL dialect 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 *et al.* 1975, Berman and Stonebraker 1977), i.e., on the integers, reals and character strings, it is generally agreed 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 GIS language (Frank and Mark 1991). 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 these extensions are integrated into the SQL environment. The variety of spatial data types proposed for spatial SQLs includes:

- a universal spatial data type (Sikeler 1985, Herring *et al.* 1988, Ooi *et al.* 1989)
- data types for each spatial dimension, e.g. points, nodes, lines, polylines, surfaces and volumes (Raper and Bundock 1991) and their generalizations to a dimension-independent spatial superclass (Egenhofer 1988)—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 and Phillips 1987) and the graphical display
- a data type for bitmap graphics (Roussopoulos and Leifker 1985)

The actual syntax of these extensions differs considerably. Three particular approaches have been pursued: some query languages reserve specific attribute names for spatial attributes such as *loc* (Sikeler 1985, Roussopoulos and Leifker 1985) or *map*, *area* and *perimeter* (Ingram and Phillips 1987); others assign no explicit spatial attribute to a spatial relation, but implicitly assume its existence (Herring *et al.* 1988, Ooi *et al.* 1989); and others define only the names of the spatial data types and with the data definition, spatial attributes are defined by associating an attribute name with a spatial data type (Egenhofer 1988, Raper and Bundock 1991).

3.2. *Spatial relationships*

The extension with spatial data types is useless unless the pertinent operations and relationships are also defined. This follows the concept of abstract data types (ADTs) (Guttag 1977) and their integration into database management systems, as promoted by such database management systems as Postgres (Rowe and Stonebraker 1987) and Iris (Fishman *et al.* 1986). Spatial relationships are Boolean operations to check whether or not a particular predicate holds true between tuples of two or more spatial relations. The relationships of pure SQL, such as 'less' or 'greater', are too low level to address all the spatial concepts sufficiently. All spatial SQL dialects acknowledge this and include extensions for spatial relationships; however, they differ in the number of spatial relationships, their semantics and the syntax in which they are incorporated into the SQL framework.

- With two exceptions (Herring *et al.* 1988, Egenhofer 1991), no formal definitions are given for the semantics of the relationships.
- Spatial relations are introduced in prefix (Ingram and Phillips 1987) or infix form (Egenhofer 1988, Ooi *et al.* 1989, Raper and Bundock 1991).

- Spatial predicates are used on spatial attributes (Egenhofer 1988, Raper and Bundock 1991) or on relations (Sikeler 1985, Ingram and Phillips 1987, Herring *et al.* 1988, Ooi *et al.* 1989).

In respect of syntax only, it becomes obvious that 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. Such an overloading of SQL's standard predicates is sufficient for some extensions, e.g., temporal relationships (Sarda 1990), but it is insufficient for spatial relationships because the set of standard predicates in SQL is too small to cover all spatial relationships (Egenhofer 1991).

Although a list of operation names and their parameters (Raper and Bundock 1991) is useful in showing the variety of spatial relationships, it offers solutions only to syntactical problems. More important than the selection of particular terminology for handling spatial data is a *formal* definition of the semantics of the operations and their combinations. Although there have been attempts to formalize some particular subsets, e.g., cardinal directions (Peuquet and Ci-Xiang 1987, Frank 1991), topological relations (Egenhofer 1989 a, Egenhofer and Herring 1990, Egenhofer and Franzosa 1991), or their combinations (Chang *et al.* 1989, Hernández 1991), there exists currently no comprehensive set of formal definitions for spatial relationships; therefore, the semantics of the relationships differs considerably among the various spatial query languages (Egenhofer 1989 b, Guenther and Buchmann 1990).

3.3. *Graphical display*

An *ad hoc* GIS query language requires that query results can be displayed graphically. This process has two distinct 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 presentation of alphanumeric data in SQL; and (2) describing how to display the query result. For both issues, different solutions have been proposed.

- A qualifier for a spatial attribute in the SELECT clause specifies that a particular spatial relation be graphically displayed (Sikeler 1985).
- All spatial attributes in the SELECT clause are displayed, whereas non-spatial parts of the query result are shown as alphanumeric tables (Ingram and Phillips 1987).
- The query result is displayed according to a display mode in a display environment (Egenhofer 1991).

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 presentation. Only PSQL and Spatial SQL propose solutions for this problem. PSQL displays query results according to predefined definitions, called *picture lists* (Roussopoulos and Leifker 1985), without providing a language to create or modify a picture list; Spatial SQL displays spatial query results according to the definitions in the *graphical presentation environment* (Egenhofer 1991). A comprehensive display language as a superset of SQL lets users describe the use of colours, patterns, symbols, etc. for spatial relations.

3.4. *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 displayed as a single rendering and no interaction with the currently displayed result is possible. For a GIS language with a graphical display of query results, this SQL feature is a major restriction, because the graphical presentation of query results animates users to refer to the drawings when formulating further queries.

As 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 and Phillips 1987) or PICK (Egenhofer 1988), although the use of a spatial function named CURSOR has also been proposed (Raper and Bundock 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.

3.5. *Compliance with the syntax framework*

Any SQL extension suffers from the dilemma of extending SQL's functionality while preserving the standardized form of SQL. A variety of syntax modifications of the SELECT-FROM-WHERE framework have been proposed to enable users to query spatial data. Three fundamentally different trends can be found: (1) the addition of new clauses in which particular spatial properties are addressed such as WITH LOCATION (Sikeler 1985) and AT (Roussopoulos and Leifker 1985) for spatial conditions, and ON (Roussopoulos *et al.* 1988) to specify which output format to use; (2) the treatment of the SELECT and WHERE clauses so that they address relations *in lieu* of atttributes, and cancelling the FROM clause, which becomes superfluous (Herring *et al.* 1988); and (3) minimal extensions within the given framework to comply with standard SQL (Ingram and Phillips 1987, Egenhofer 1988, Ooi *et al.* 1989) and the definition of other extensions outside SQL (Egenhofer 1991).

4. Impediments of the SQL framework for a query language for GIS

SQL extensions that attempt to comply with the SQL standard have to live with the conceptual constraints set forth by the standardized framework. This section will show why it is so difficult to extend standard SQL to become a useful GIS query language based on a theoretically sound data model. A number of shortcomings of the SQL framework for handling spatial data will be examined, some of which are due to the attempt of extending a given standardized query language. Two types of deficiencies are distinguished: (1) the lack of expressive power to formulate certain types of (spatial) queries; and (2) the lack of spatial concepts in the SQL framework. All these shortcomings are crucial impediments for a GIS query language, but it is not the lack of a particular functionality that makes the SQL framework inappropriate for a GIS query language. It is rather the fact that a great number of deficiencies exists, for which easy or simple remedies cannot be provided. Some of these problems may be fixed in SQL with considerable modifications of the initial language such as patches for the integration of some syntax mechanisms to allow users to formulate recursive SQL queries (ANSI 1991). For other problems, no such 'solutions' are in sight.

4.1. *Power of the language*

The power of the SQL language to retrieve data is determined by the operations of the relational algebra plus some additional concepts such as aggregate functions. Although SQL is *relationally complete* (Ullman 1982), i.e., any manipulation of tables is possible with a combination of the operations provided, it does not support such fundamental concepts as metadata queries and higher order queries. Furthermore, the retrieval part of SQL is inherently value based, i.e., all operations are performed on attributes, and tuples can only be compared for equal values, but not for identity. These problems are general SQL problems and also apply to non-spatial applications. Although they may be less critical for some traditional SQL applications, they impose serious restrictions on the use of any spatial extension of SQL. The impact of these deficiencies on a GIS query language is discussed in the following.

4.1.1. *Object identity*

Identity is the property that distinguishes each object from all others (Khoshafian and Copeland 1986); therefore, object identity allows the comparison of whether two objects are the same, independent of their particular attribute values. Object identity has been acknowledged as a major component of next generation database systems (Atkinson *et al.* 1989, Stonebraker *et al.* 1990) because it is crucial for any application in which objects change over time. Geographical objects are prototypical examples of this (Langran 1989, Al-Taha and Barrera 1990). A land parcel, for example, may change all its characteristics over time: the boundaries are re-surveyed and adjusted, new owners buy the land, the land may be used differently, or the parcel may be given a new postal address (Hunter and Williamson 1990). Nevertheless, the piece of land is still the same and to relate its state at one time in history to a state at a different time it is necessary that each parcel has an object identifier which is independent of the descriptive values of the parcel.

4.1.2. *Metadata queries*

Although the SQL query facilities were particularly designed for retrieving data, they are limited to asking queries about the structure of tuples. Such *data queries* typically start with some knowledge about the structure of the tables, that is, users have to remember the names of the tables and the corresponding attributes. The instructions are to find corresponding tuples, the values of which fulfil certain constraints. Although these queries may cover most questions users ask against a database, there are other kinds of queries with which users request information about data, e.g. 'To which relation(s) does a particular tuple belong?', 'To which attribute(s) does a specific value belong?', 'What is the domain of a specific attribute?', or 'To which view does a particular relation belong?'. Such *metadata queries* are particularly important for spatial databases when users request information referring to graphically displayed information, as shown by the following examples.

- Topographic and thematic maps typically contain symbols and labels for various features. Users may point to them asking, 'What is this?' and expect an answer like, 'This is a castle' or 'This is a train station'. The answers to such queries are the names of relations or attributes, not particular values of attributes.
- An answer to a query about a label on a map may be, 'This is the mileage along Route 1A from Brewer to Bar Harbor'. This answer contains metadata—the mileage is the attribute name of the label— in combination with data, namely that the label identifies the mileage for a road and that it is the value from Brewer to Bar Harbor.
- The query 'What are possible soil classifications?' refers to the domain of an attribute, soil type. Note that this query is different from, 'What are all the values

stored in the attribute soil type?' The result of the first query is independent of the actual values stored in a table and its definition is the subject of the data definition, whereas the result of the latter query may change with the storage, deletion, or update of any value.

As SQL lets users query only the values of the tuples, it does not support metadata queries. Certainly, in addition to an actual database it would be possible to create metadata databases, which a user can also query with SQL. Although such a 'solution' would allow the users to ask for metadata information, the redundant storage of metadata—in the data definition and as data—would introduce new problems, because the standard update anomalies of redundant storage would also apply to the dependencies between data definition and actual data.

4.1.3. *Knowledge queries*

Queries of a nature similar to metadata queries are *knowledge queries* (Motro and Yuan 1990), explaining the reasoning process that underlies a particular query language. Spatial relationships, for example, are typically derived from a representation in a spatial data model, rather than being explicitly recorded (Davis 1986). The derivation is based on rules that formalize the criteria for the individual relationships. Spatial information systems that allow users to tailor their spatial predicates (Herring *et al.* 1988) must also include facilities to let them inquire about the rules used for a specific predicate. For example, the adjacency between two parcels may be defined such that they share at least one common boundary, but have no common interior (Egenhofer and Herring 1990) and users may want to know from the system, 'Why were the two objects identified as neighbours?' expecting an answer such as, 'Because they share a common boundary'.

In a similar way, users may ask queries about consistency constraints. For example, the query, 'What are geometrical constraints about rivers?' may be answered by, 'They must not cross each other and a river cannot cross a road, unless there is a bridge'. Such knowledge queries are of particular interest when users modifying the database want to obtain information before making a change about what data they must provide. Likewise, they may want to find out more details about their—or the system's—failure after an unsuccessful attempt to update the database.

4.1.4. *Qualitative answers*

SQL lets users ask queries with quantitative results, i.e. about the values of tuples; however, it does not support queries with qualitative results, for instance, about the kind of relationship between objects. Predicates, such as 'less' or 'equal', can only appear as part of WHERE clauses, whereas qualitative answers in SQL require that they are also part of SELECT clauses. For example, although it is possible to ask for 'all roads in Penobscot county that are wider than the road from Bangor to Bar Harbor', it is impossible to formulate an SQL query for 'the relation between the widths of Interstate I95 and Route 1A', with an expected answer such as, 'I95 is wider than Route 1A'.

The conceptual problem behind this shortcoming is that predicate names cannot be used as variables. Predicate calculus is a formal framework within which this problem of SQL can be easily analysed. By mapping relations (in the FROM clause) and conditions (in the WHERE clause) onto predicates, it is possible to translate SQL queries into Horn clauses (Egenhofer 1989 b, Draxler 1990), a common representation in logic (Kowalski 1979). For example, the query for all roads wider than Route 1A can be expressed both in SQL and as a Horn clause.

```
SELECT r1.name                  select (r1.name) IF road (r1.name, r1.width),
FROM   r1 road, r2 road                             road (r2.name, r2.width),
WHERE  r2.name = "1A" and                           equal (r2.name, "1A"),
       r1.width > r2.width                          greater (r1.width, r2.width).
```

It becomes apparent that query results are only arguments of the predicates, and the names of the predicates stand for the names of the relations and operators. Furthermore, the clausal representation shows how changing the query, so that it returns the relationship between the widths of the two roads, requires a new concept: the query result is now the *name* of a predicate, not one of its arguments, therefore, predicate names have to be variables as well.

```
select (roadRelation) IF road (r1.name, r1.width),
                         road (r2.name, r2.width),
                         road (r1.name, "195"),
                         equal (r2.name, "1A"),
                         roadRelation (r1.width, r2.width).
```

A language that has only constant values for predicate names is a *first-order language* (Gallaire *et al.* 1984), whereas a language with constants and variables over predicate names is *second-order*. Obviously, SQL belongs to the family of first-order languages and, therefore, no qualitative answers are possible. It must be admitted that the formal systems dealing with second-order languages are extremely difficult to understand and sometimes even inconsistent (Gallaire *et al.* 1984).

4.2. *Decoupled retrieval and display*

Standard SQL is targeted to retrieving data modelled 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 presentation 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 presentation 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 graphical presentation 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.

4.2.1. *Specification of graphical display*

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 colours, 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 and to specific spatial objects or classes of objects in the result. Especially for GIS applications with high quality map outputs, 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 display the query result previously retrieved; however, such a separation does not take into account that the query and the specification of graphical presentation often depend on 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
```

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 displayed with the same graphical 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 has been provided by the query results.

4.2.2. *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 and Frank 1988 b). 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 1990). 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 on 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 presentation types are essentially impossible, unless changes are made to a degree that the extension is no longer compatible with the standard.

4.2.3. *Modifying the graphical presentation*

Users of GIS typically work with their query results. In addition to analysing them visually, they often modify the presentation of the currently displayed objects—without updating the database. Such presentational changes may be purely graphical ('Replace colour "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.

5. Conclusions

The intent of this paper was to bring a new perspective to the ongoing discussion about the use of the SQL framework as the standard GIS query language. Although SQL has been fairly successfully applied as a query language for standard database applications, such as banking accounts, its success in the domain of spatial applications, such as geographical information systems or CAD/CAM systems, has been very limited. Despite continuous efforts to improve the SQL standard—there are proposals under discussion for SQL2 and SQL3 (ANSI 1991)—and numerous attempts to extend SQL with various spatial features, no satisfactory solutions for an SQL-based interactive GIS query language have been found. Some solutions may be offered by an *object-oriented* SQL, for instance to include spatial abstract data types (ADTs); however, this gives rise to incompatibilities among different implementations—different systems may have customized spatial data types and operations. Extensions of the SQL framework with a spatial ADT, including spatial relationships as predicates, are possible; however, before syntax extensions are discussed, the semantics of the operations on the spatial ADT must be defined formally. Informal sets of operations to be undertaken by a spatial database have been collected (Joseph and Cardenas 1988, Tomlin 1990); however, little effort has been put into the formalization of spatial data models (Güting 1988, Dorenbeck and Egenhofer 1991) and spatial relationships and operations such as cardinal directions (Peuquet 1988, Chang *et al.* 1989, Frank 1991, Hernández 1991) and topological relationships (Egenhofer 1989 a, Egenhofer and Herring 1990, Egenhofer and Franzosa 1991).

The two major deficiencies of any spatial SQL are: (1) the severe difficulties of incorporating such necessary spatial concepts into SQL as graphical display and its specification; and (2) the lack of power within the relational framework with missing support for qualitative answers, knowledge queries and metadata queries—all crucial when dealing with spatial data. The incapability of SQL to process qualitative and knowledge queries stems from the separation between classical database management systems (DBMS) and knowledge systems. Latest C++ based DBMS allow users some limited access to metadata; however, the incorporation of such concepts into the SQL framework attacks the basics of SQL—the relational algebra. Derivations from (and extensions of) SQL that go too far from relational algebra risk invalidating the underlying formal framework, without which SQL remains a pure—and poor—syntax shell.

Any spatial SQL dialect can be considered as only a short term 'solution' for an interactive GIS query language, and GIS 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. The latest proposals for a standardized SQL do not improve on this issue; they are instead targeted at turning SQL into a 'complete' programming language. Unfortunately, hiding SQL under a 'human interface' does not imply the creation of a better database interface—it puts the burden of simulating high level spatial concepts onto the application programmer. Instead, new high level interface languages for GIS are necessary that support the retrieval of data, the appropriate presentation of query results and the interaction between the user and the system in an integrated fashion. The design of such a language must start at the user level by investigating what kinds of operations users want to perform on spatial databases and how they do it (Egenhofer and Frank 1988 b, Pizano *et al.* 1989, Mainguenaud and Portier 1990). In a complex environment, such as a GIS, '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. To support such a working behaviour, interface languages for GISs are necessarily based on cognitive and mental models, such as image schemata and metaphors (Lakoff and Johnson 1980), applied to spatial data (Kuhn and Frank 1991).

Acknowledgments

This work was partially funded by grants from Digital Equipment Corporation under Sponsored Research Agreement No. 414, TP-765536 and BW-213860, and Intergraph Corporation. Additional support from NSF for the NCGIA under grant number SES 88-10917 is gratefully acknowledged. Renato Barrera and an anonymous referee provided helpful suggestions to improve the paper. I also appreciate Andrew Frank's and Robert Cicogna's comments on an earlier version of this paper and the discussions with participants at the NATO Advanced Study Institute on 'Cognitive and Linguistic Aspects of Geographic Space' in Las Navas, Spain, July 1990.

An earlier version of this paper appeared in the proceedings of this workshop.

References

Abel, D., 1988, Relational data management facilities for spatial information systems. *Proceedings of Third International Symposium on Spatial Data Handling, Sydney, Australia*, edited by D. Marble, pp. 9–18.

Al-Taha, K., and Barrera, R., 1990, Temporal data and GIS: an overview. *Proceedings of GIS/LIS '90, Anaheim, CA, USA*, pp. 244–254.

American National Standards Institute (ANSI), 1989, *X3.135-1989 Database Language SQL*.

American National Standards Institute (ANSI), 1991, *X3H2-91-183 Database Language SQL2/SQL3 (Working Paper)*.

Ariav, G., 1986, A temporally oriented data model. *ACM Transactions on Database Systems*, **11**, 499–527.

Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., and Zdonik, S., 1989, The object-oriented database system manifesto. *Proceedings of Deductive and Object-Oriented Databases 89*, Kyoto, Japan, pp. 40–56.

Berman, R., and Stonebraker, M., 1977, GEO-QUEL, a system for the manipulation and display of geographic data. *ACM Computer Graphics*, **11**, 186–191.

Bertin, J., 1983, *Semiology of Graphics* (Madison, WI: University of Wisconsin Press).

Chamberlin, D., Astrahan, M., Eswaran, K., Griffiths, P., Lorie, R., Mehl, J., Reisner, P., and Wade, B., 1976, Sequel 2: a unified approach to data definition, manipulation, and control. *IBM Journal of Research and Development*, **20**, 560–575.

Chang, N. S., and Fu, K. S., 1980, Query-by-pictorial-example. *IEEE Transactions on Software Engineering*, **SE-6**, 519–524.

Chang, S. K., Jungert, E., and Li, Y., 1989, The design of pictorial databases based upon the theory of symbolic projections. *Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, CA, USA*, edited by A. Buchmann, O. Günther, T. Smith, and Y. Wang, *Lecture Notes in Computer Science, Vol.* 409 (New York: Springer-Verlag), pp. 303–323.

Codd, E., 1970, A relational model for large shared data banks. *Communications of the ACM*, **13**, 377–387.

Date, C., 1986, *An Introduction to Database Systems*, 4th edn (Reading, MA: Addison Wesley).

Date, C., 1988, Defining data type in a database language. *SIGMOD Record*, **17**, 53–76.

Date, C., and White, C., 1989, *A Guide to SQL/DS* (Reading, MA: Addison Wesley).

DAVIS, E., 1986, *Representing and Acquiring Geographic Knowledge, Research Notes in Artificial Intelligence* (Los Altos, CA: Morgan Kaufmann).

DORENBECK, C., and EGENHOFER, M., 1991, Algebraic optimization of combined overlay operations. *Proceedings of Autocarto 10, Baltimore, MD, USA*, edited by D. Mark and D. White, pp. 296–312.

DRAXLER, C., 1990, Logic programming and databases. Technical Report 90.09, Institute for Informatics, University of Zurich, Switzerland.

EGENHOFER, M., 1984, Implementation of Mapquery, a query language for land information systems. Technical Report 79, Institute for Geodesy and Photogrammetry, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (in German).

EGENHOFER, M., 1987, An extended SQL syntax to treat spatial objects. *Proceedings of the Second International Seminar on Trends and Concerns of Spatial Sciences, Fredericton, New Brunswick, Canada*, edited by Y. C. Lee, pp. 83–95.

EGENHOFER, M., 1988, A spatial SQL dialect. Technical Report 100, Department of Surveying Engineering, University of Maine, Orono, ME, USA.

EGENHOFER, M., 1989 a, A formal definition of binary topological relationships. *Third International Conference on Foundations of Data Organization and Algorithms (FODO), Paris, France*, edited by W. Litwin and H.-J. Schek, *Lecture Notes in Computer Science, Vol. 367* (New York: Springer-Verlag), pp. 457–472.

EGENHOFER, M., 1989 b, Spatial query languages, *PhD Thesis*, University of Maine, Orono, ME, USA.

EGENHOFER, M., 1990, Interaction with geographic information systems via spatial queries. *Journal of Visual Languages and Computing*, **1**, 389–413.

EGENHOFER, M., 1991, Extending SQL for cartographic display. *Cartography and Geographic Information Systems*, **18**, 230–245.

EGENHOFER, M., Spatial SQL: a query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, in press.

EGENHOFER, M., and FRANK, A., 1988 a, Designing object-oriented query languages for GIS: human interface aspects. *Proceedings of Third International Symposium on Spatial Data Handling, Sydney, Australia*, edited by D. Marble, pp. 79–96.

EGENHOFER, M., and FRANK, A., 1988 b, Towards a spatial query language: user interface considerations. *Proceedings of 14th International Conference on Very Large Data Bases, Los Angeles, CA, USA*, edited by D. DeWitt and F. Bancilhon, pp. 124–133.

EGENHOFER, M., and FRANK, A., 1990, Lobster: combining AI and database techniques for GIS. *Photogrammetric Engineering & Remote Sensing*, **56**, 919–926.

EGENHOFER, M., and FRANZOSA, R., 1991, Point-set topological spatial relations. *International Journal of Geographical Information Systems*, **5**, 161–174.

EGENHOFER, M., and HERRING, J., 1990, A mathematical framework for the definition of topological relationships. *Proceedings of Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland*, edited by K. Brassel and H. Kishimoto, pp. 803–813.

EHRICH, H.-D., LOHMANN, F., NEUMANN, K., and RAMM, I., 1988, A database language for scientific map data. *Geologisches Jahrbuch*, **A104**, 139–152.

FISHMAN, D., BEECH, D., CATE, H., CHOW, E., CONNORS, T., DAVIS, J., DERRETT, N., HOCH, C., KENT, W., LYNBAEK, P., MAHBOD, B., NEIMAT, N., RYAN, T., and SHAN, M., 1986, Iris: an object-oriented database management system. *ACM Transactions on Office Information Systems*, **5**, 48–69.

FRANK, A., 1982, Mapquery—database query language for retrieval of geometric data and its graphical representation. *ACM Computer Graphics*, **16**, 199–207.

FRANK, A., 1984 a, Extending a database with Prolog. *Proceedings of First International Workshop on Expert Database Systems, Kiawah Island, SC, USA*, edited by L. Kerschberg, pp. 665–676.

FRANK, A., 1984 b, Requirements for database systems suitable to manage large spatial databases. *Proceedings of International Symposium on Spatial Data Handling, Zurich, Switzerland*, pp. 38–60.

FRANK, A., 1988, Requirements for a database management system for a GIS. *Photogrammetric Engineering & Remote Sensing*, **54**, 1557–1564.

FRANK, A., 1991, Qualitative spatial reasoning about cardinal directions. *Proceedings of Autocarto 10, Baltimore, MD, USA*, edited by D. Mark and D. White, pp. 148–167.

FRANK, A., Spatial concepts, geometric data models and data structures. *Computers and Geo-Sciences*, in press.

FRANK, A., and MARK, D., 1991, Language issues for GIS. *Geographical Information Systems: Principles and Applications*, edited by D. Maguire, M. Goodchild, and D. Rhind (London: Longman), pp. 147-163.

GALLAIRE, H., MINKER, J., and NICOLAS, J., 1984, Logic and databases: a deductive approach. *ACM Computing Surveys*, **16**, 153–185.

GO, A., STONEBRAKER, M., and WILLIAMS, C., 1975, An approach to implementing a geodata system. Technical Report, Memo ERL-M529, Electronics Research Laboratory, University of California, Berkeley, CA, USA.

GUENTHER, O., and BUCHMANN, A., 1990, Research issues in spatial databases. *SIGMOD Record*, **19**, 61–68.

GÜTING, R., 1988, Geo-relational algebra: a model and query language for geometric database systems. *Advances in Database Technology—EDBT '88, International Conference on Extending Database Technology, Venice, Italy*, edited by J. Schmidt, S. Ceri, and M. Missikoff, *Lecture Notes in Computer Science, Vol. 303*, (New York: Springer-Verlag), pp. 506–527.

GUTTAG, J., 1977, Abstract data types and the development of data structures. *Communications of the ACM*, **20**, 396–403.

HÄRDER, T., and REUTER, A., 1985, Architecture of database systems for non-standard applications. *Database Systems in Office, Engineering, and Science, Karlsruhe, Germany*, edited by A. Blaser and P. Pistor, *Fachberichte Informatik, Vol. 94* (New York: Springer-Verlag), pp. 253–286.

HERNÁNDEZ, D., 1991, Relative representation of spatial knowledge: the 2-d case. *Cognitive and Linguistic Aspects of Geographic Space*, edited by D. Mark and A. Frank (Dordrecht: Kluwer Academic), pp. 373–385.

HERRING, J., LARSEN, R., and SHIVAKUMAR, J., 1988, Extensions to the SQL language to support spatial analysis in a topological data base. *Proceedings of GIS/LIS '88, San Antonio, TX, USA*, pp. 741–750.

HUNTER, G., and WILLIAMSON, I., 1990, The development of a historical digital cadastral database. *International Journal of Geographical Information Systems*, **4**, 169–179.

INGRAM, K., and PHILLIPS, W., 1987, Geographic information processing using a SQL-based query language. *Proceedings of AUTO-CARTO 8, Eighth International Symposium on Computer-Assisted Cartography, Baltimore, MD, USA*, edited by N. Chrisman, pp. 326–335.

JOSEPH, T., and CARDENAS, A., 1988, Picquery: a high level query language for pictorial database management. *IEEE Transactions on Software Engineering*, **14**, 630–638.

KHOSHAFIAN, S., and COPELAND, G., 1986, Object identity. *Proceedings of Object-Oriented Programming Systems, Languages, and Applications—OOPSLA '86, Portland, OR, USA*, pp. 406–416.

KORTH, H., and SILBERSCHATZ, A., 1986, *Database System Concepts* (New York: McGraw Hill).

KOWALSKI, R., 1979, *Logic for Problem Solving* (New York: Elsevier Science).

KUHN, W., and FRANK, A., 1991, A formalization of metaphors and image-schemas in user interfaces. *Cognitive and Linguistic Aspects of Geographic Space*, edited by D. Mark and A. Frank (Dordrecht: Kluwer Academic), pp. 419–434.

LAKOFF, G., and JOHNSON, M., 1980, *Metaphors We Live By* (Chicago, IL: University of Chicago Press).

LANGRAN, G., 1989, A review of temporal database research and its use in GIS applications. *International Journal of Geographical Information Systems*, **3**, 215–232.

LAURINI, R., and MILLERET, F., 1988, Spatial data base queries: relational algebra versus computational geometry. *Statistical and Scientific Database Management, 4th International Conference, Rome, Italy*, edited by M. Rafanelli, J. Klensin, and P. Svensson, *Lecture Notes in Computer Science, Vol. 339* (New York: Springer-Verlag), pp. 291–313.

LORIE, R., 1991, The use of a complex object language in geographic data management. *Advances in Spatial Databases—Second Symposium, SSD '91*, edited by O. Günther and H.-J. Schek, *Lecture Notes in Computer Science, Vol. 525* (New York: Springer-Verlag), pp. 319–337.

LORIE, R., and SCHEK, H.-J., 1988, On dynamically defined complex objects and SQL. *Advances in Object-Oriented Database Systems—Proceedings of the 2nd International Workshop on Object-Oriented Database Systems, Bad Münster am Stein-Ebernburg, Germany*, edited by K. Dittrich, *Lecture Notes in Computer Science, Vol. 334* (New York, Springer-Verlag), pp. 323–328.

MAINGUENAUD, M., and PORTIER, M.-A., 1990, Cigales: a graphical query language for geographical information systems. *Proceedings of Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland*, edited by K. Brassel and H. Kishimoto, pp. 393-404.

MITSCHANG, B., 1989, Extending the relational algebra to capture complex objects. *Proceedings of Fifteenth International Conference on Very Large Data Bases, Amsterdam*, edited by P. Apers and G. Wiederhold, pp. 297-305.

MOTRO, A., and YUAN, Q., 1990, Querying database knowledge. *Proceedings of 1990 ACM SIGMOD, International Conference on Management of Data, Atlantic City, NJ, USA*, edited by H. Garcia-Molina and H. Jagadish, pp. 173-183.

OOI, B., 1990, *Efficient Query Processing in Geographic Information Systems, Lecture Notes in Computer Science, Vol. 471* (New York: Springer-Verlag).

OOI, B., SACKS-DAVIS, R., and MCDONELL, K., 1989, Extending a DBMS for geographic applications. *Proceedings of IEEE Fifth International Conference on Data Engineering, Los Angeles, CA, USA*, pp. 590-597.

PEUQUET, D., 1988, Toward the definition and use of complex spatial relationships. *Proceedings of Third International Symposium on Spatial Data Handling, Sydney, Australia*, edited by D. Marble, pp. 211–223.

PEUQUET, D., and CI-XIANG, Z., 1987, An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition*, **20**, 65–74.

PIZANO, A., KLINGER, A., and CARDENAS, A., 1989, Specification of spatial integrity constraints in pictorial databases. *Computer*, **22**, 59-71.

PULLAR, D., 1987, Query language for spatial relationships. *Proceedings of ASPRS-ACSM Annual Convention, Baltimore, MD, USA*, pp. 180-192.

RAPER, J., and BUNDOCK, M., 1991, UGIX: a layer based model for a GIS user interface. *Cognitive and Linguistic Aspects of Geographic Space*, edited by D. Mark and A. Frank (Dordrecht: Kluwer Academic), pp. 449-475.

REISNER, P., 1981, Human factors studies of database query languages: a survey and assessment. *ACM Computing Surveys*, **13**, 13–31.

ROUSSOPOULOS, N., and LEIFKER, D., 1985, Direct spatial search on pictorial databases using packed R-trees. *SIGMOD Record*, **14**, 17–31.

ROUSSOPOULOS, N., FALOUTSOS, C., and SELLIS, T., 1988, An efficient pictorial database system for PSQL. *IEEE Transactions on Software Engineering*, **14**, 630–638.

ROWE, L., and STONEBRAKER, M., 1987, The Postgres data model. *Proceedings of 13th International Conference on Very Large Data Bases, Brighton, UK*, edited by P. Stocker and W. Kent, pp. 83–96.

SARDA, N., 1990, Extensions to SQL for historical databases. *IEEE Transactions on Knowledge and Data Engineering*, **2**, 220–230.

SCHEK, H.-J., 1988, Perspectives of future database systems—extensibility and object-orientation. *Proceedings of AM/FM International, European Division, Regional Conference, Siegen, Germany*, pp. 2-14 (in German).

SIKELER, A., 1985, Examination of storage structures for 3-dimensional objects. Technical Report, University Kaiserslautern (in Germany).

STONEBRAKER, M., ROWE, L., LINDSAY, B., GRAY, J., CAREY, M., and BEECH, D., 1990, Third generation data base system manifesto. *Proceedings of IFIP DS-4 Workshop on Object-Oriented Databases, Windermere, UK.*

TOMLIN, C. D., 1990, *Geographic Information Systems and Cartographic Modeling* (Englewood Cliffs, NJ: Prentice Hall).

ULLMAN, J., 1982, *Principles of Database Systems* (Rockville, MD: Computer Science Press).

ULLMAN, J., 1988, *Principles of Database and Knowledgebase Systems* (Rockville, MD: Computer Science Press).

WAUGH, T., and HEALEY, R., 1987, The Geoview design: a relational database approach to geographical data handling. *International Journal of Geographical Information Systems*, **1**, 101–118.

WESTLAKE, A., and KLEINSCHMIDT, I., 1990, The implementation of area and membership retrievals in point geography using SQL. *Statistical and Scientific Database Management, Fifth International Conference, V SSDBM, Charlotte, NC, USA*, edited by Z. Michalewicz, *Lecture Notes in Computer Science, Vol. 420* (New York: Springer-Verlag), pp. 200-218.