# SQL/SDA: A Query Language for Supporting Spatial Data Analysis and Its Web-Based Implementation

Hui Lin and Bo Huang

**Abstract**—An important trend of current GIS development is to provide easy and effective access to spatial analysis functionalities for supporting decision making based on geo-referenced data. Within the framework of the ongoing SQL standards for spatial extensions, a spatial query language, called SQL/SDA, has been designed to meet such a requirement. Since the language needs to incorporate the important derivation functions (e.g., map-overlay and feature-fusion) as well as the spatial relationship and metric functions, the functionality of the FROM clause in SQL is developed in addition to the SELECT and WHERE clauses. By restructuring the FROM clause via a subquery, SQL/SDA is well-adapted to the general spatial analysis procedures using current GIS packages. Such an extended SQL, therefore, stretches the capabilities of the previous ones. The implementation of SQL/SDA on the Internet adopts a hybrid model, which takes advantage of the Web GIS design methods in both the client side and server side. The client side of SQL/SDA, programmed in the Java language, provides a query interface by introducing visual constructs such as icons, listboxes, and comboboxes to assist in the composition of queries, thereby enhancing the usability of the language. The server side of SQL/SDA, which is composed of a query processor and Spatial Database Engine (SDE), carries out query processing on spatial databases after receiving user requests. It was demonstrated that using the familiar SELECT-FROM-WHERE statement instead of a single ad hoc command or procedural commands like macro language in some GIS packages, SQL/SDA offers users an efficient option to perform complicated multistep spatial data analyses on the Internet.

**Index Terms**—Geographical Information System (GIS), spatial analysis, query languages, spatial database, SQL, Internet, client/server, visual interface, Java.

## 1 Introduction

Geographical Information System (GIS) technology has experienced an astonishing growth in recent decades and such development is, in many ways, closely related to the spatial analytic capabilities of GIS [17], [34], [40], [41], [42]. In fact, the capabilities to handle and analyze spatial data are usually seen as the key characteristic that distinguishes a GIS from other information or computer-aided design systems.

Currently, the rapid growth of the Internet has led to an increasing concern over the Web GIS [1], [20]. Although Web GIS has been employed for spatial data access, transmission, simple retrieval, and mapping, the access to spatial analysis functions such as map overlay, buffer, and feature fusion that are vital for GIS-based decision-making is still very limited [6], [36]. This is, in part, due to the fact that there is no effective interface to support the expression of user requests dealing with these analysis functions. In the World Wide Web (WWW) client/server environment, the client usually sends a request to the server, which processes the request and returns the result to the client. The design of a spatial query language can facilitate such a query process that is crucial for a Web GIS.

• *The authors are with the Joint Laboratory for GeoInformation Science, The Chinese University of Hong Kong, Shatin, NT, Hong Kong. E-mail: {huilin, bohuang}@cuhk.edu.hk.*

*Manuscript received 12 July 1999; accepted 16 Aug. 1999; posted to Digital Library 6 Apr. 2001.*
*For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 110128.*

The need for a spatial query language has been identified in the GIS arena [16]. In the last decade, many authors have designed their extended database languages (e.g., [2], [14], [25], [26], [31], [37], [48]). For a detailed overview; see [14], [24]. These SQL-like languages (Spatially Extended SQLs) introduce spatial data types (e.g., point, line, polygon, and image) and spatial operators (e.g., distance, direction, overlap, and contain), allowing users to inquire about spatial features, primarily in terms of spatial relationships (e.g., "A CONTAINS B") and metric constraints (e.g., "DISTANCE(A, B) < 10"). The Spatial SQL [14] also adds graphical capabilities for the presentation of a query result. On the other hand, query languages using algebraic approach have also been proposed [21], [22]. These languages, however, concentrate on the representation and management of spatial data rather than stepwise analysis of spatial data (see also [44]).

GIS, as a tool for spatial analysis, has been used in many different fields. Svensson and Huang [44] discussed the required functionalities (i.e., reclassification, measurement, overlay, neighborhood, and statistics) for a spatial analysis system. In many cases, most of these operations can be performed by current GIS packages in certain ways. But, the complexity of comprehensive GIS software packages has been an obstacle to their widespread use by application specialists. To this end, Geo-SAL was devised on top of SAL, a query language different from SQL and QUEL, to support spatial data analysis. Since the syntax and semantics of SAL are complicated, they inevitably influence Geo-SAL considerably.

Herring et al. [23] discussed user-defined extensions to SQL by using a macro expander. Such macroextensions allow users to tailor their environment flexibly by using spatial operators to suit their operational scenarios.

In addition to spatial query languages, attempts have been also made on using a procedural language for cartographic modeling and spatial data analysis. Tomlin's Map Algebra [46], which is most suited for the raster data structure, is a typical example. Different interface designs for implementing such an algebra, ranging from command-line to advanced graphical interfaces, are also assessed in [5]. The principles underlying these interfaces are considered applicable to the design of web-top interfaces. In general, Map Algebra can be seen as a macro language if several commands in it are used together. The comparison between spatial query language and macro language will be discussed in Section 6.

Recently, there are two major efforts for standardizing the storage and management of spatial data: SQL3 multimedia specification (SQL/MM) [43] and Open GIS Simple Features Specification for SQL (OpenGIS SQL) [32]. Both the ongoing standards define a set of spatial data types and operations and, in general, provide a framework for spatial query language design and development.

Within such a framework, the objective of this paper is to propose a spatial query language, called SQL/SDA (Spatial Data Analysis), to support the expression of complicated spatial queries dealing with various spatial analysis problems, such as site selection for a new building or land suitability evaluation for planting coffee. These spatial analysis problems usually involve selection or evaluation with a given set of criteria based on a sequence of spatial operations (e.g., map overlay, buffer, classification, or feature fusion). In this sense, the design of SQL/SDA is an attempt to advance the current spatial query languages and specifications mentioned above to facilitate stepwise analysis of spatial data. A critical factor to the design is that SQL/SDA is required to comply with the general spatial analysis procedure using current GIS packages, while being compatible with the SQL design concepts. Since the important spatial analysis functions, i.e., derivation functions (e.g., map overlay and feature fusion) [23], are different from those for determining spatial relationship and metric functions, they need to be incorporated in a different way. Thus, not only the main SELECT and WHERE clauses are employed, as has been done in the previous spatially extended SQLs, but the potential of the FROM clause is also explored. On the other hand, SQL/SDA is implemented on the Internet by using a hybrid model based on the client/server architecture. The SQL/SDA client provides a query interface with a style of visual language, i.e., using visual constructs such as icons and listboxes to help users formulate their queries. This is expected to minimize the code input by typing and to increase the user friendliness.

It has been argued whether the relational database language SQL can be successfully extended for spatial applications, and the problems with SQL-based spatial extensions are analyzed in [13]. However, since SQL is still a popular database language and its functionalities have been enhanced considerably, it is considered as the most preferred option for this study.

The remainder of this paper is organized as follows: Section 2 introduces the spatial data types and spatial analysis functions used in SQL/SDA. Section 3 describes the presentation of SQL/SDA with stress on its expression of the FROM clause. In Section 4, two examples are presented to illustrate how the typical spatial analysis problems are expressed by SQL/SDA. Comparisons of SQL/SDA with macro languages of GIS and the previous spatial query languages are given in Section 5. Section 6 discusses the design of SQL/SDA's prototype on the Web, including the Java-based visual interface and server functions. Finally, in Section 7, the characteristics of SQL/SDA are summarized and its future development is highlighted.

## 2 Spatial Data Types and Analysis Functions in SQL/SDA

Generally, there are two kinds of spatial data models in GIS: feature-based and layer-based. The feature-based approach models spatial features, while the layer-based approach models map or a set of thematic maps [45]. The feature-based data model is currently supported by many GIS packages, such as ArcView, MapInfo and Modular GIS Environment (MGE).

A simple feature, including both spatial attributes (e.g., coordinates and topological relationships) and nonspatial attributes (e.g., name, type, and size) as defined in OpenGIS Abstract Specification [33], is a basic geometric unit for spatial representation and processing in SQL/SDA. A class of such features, which has similar properties, are conceptually represented as a table in a Relational Database Management System (RDBMS) and each feature corresponds to a row in the table. The spatial attributes in a feature table are geometry valued and, in fact, are a stream of structured coordinates.

There are currently two implementation methods to spatial feature table, i.e., SQL and SQL with geometry types. In SQL/SDA, we mainly adopt the latter one. The term SQL with geometry types refers to a SQL environment that has been extended with a set of geometry types. In this environment, a geometry-valued column is implemented as a column whose data type is drawn from the set of geometry types. The following describes a set of SQL/SDA geometry types and the functions for spatial analysis on those types.

### 2.1 Geometry Types

The following set of geometry types:

{ Geometry, Point, Linestring, Polygon, Collection,
Multipoint, Multilinestring, Multipolygon}

is defined in SQL/SDA. Their definition can be found in [32], [43]. The special cases, e.g., self-intersecting polygons or polygons with holes are also covered in these literature.

The Geometry type has subtypes of Point, Linestring, Polygon, and Collection. A Collection is a Geometry that is a composition of possibly heterogeneous geometries.

Multipoint, Multilinestring, and Multipolygon are specific subtypes of Collection used to manage homogenous collections of points, linestrings, and polygons. Point and Multipoint are zero-dimensional geometric types. Linestring and Multilinestring are one-dimensional geometric types while Polygon and Multipolygon are two-dimensional geometric types.

Using the above geometry types, a feature table named landuse is defined with three columns named ID, type, and Location of type INTEGER, VARCHAR, and POLYGON, respectively. "Location" is a spatial attribute, while the others are nonspatial ones. Users can reference the attribute Location in the same way as other attributes.

```
CREATE TABLE landuse (
    ID          INTEGER NOT NULL, PRIMARY KEY,
    type        VARCHAR(50),
    Location    POLYGON NOT NULL,
)
```

The following tables are also defined by similar methods:

```
soil (ID, type, Location)
building (ID, name, owner, usage, Location)
sewer (ID, type, capacity, Location)
stream (ID, name, Location)
shop (ID, name, Location)
```

In the above feature tables, the soil and building features are of polygon type, the sewer and stream of linestring type, and the shop of point type.

These tables are to be used in this paper.

## 2.2 Spatial Analysis Functions

Although many attempts have been made to classify spatial analysis operations in a GIS, there is still no consensus on the taxonomy. However, some fundamental and commonly used ones have been identified [3], [18], [46].

The research [44] shows when these analysis operations are represented by a basic set of analysis-oriented spatial functions, the database query language approach can be expected to simplify the concepts needed in a spatial analysis system. Several sets of such functions have been proposed and defined [4], [19], [22], [38]. Both SQL/MM and OpenGIS SQL have also defined their sets of spatial functions. Based on them, four groups of spatial functions are defined in SQL/SDA.

### *2.2.1 Functions that Access Properties of Spatial Features (Property Functions)*

This group of spatial functions mainly includes:

| | |
|---|---|
| CENTROID(g Geometry): | Returns a Geometry that is the center point of polygon g |
| AREA(g Geometry): | Returns a numerical value that is the area of polygon g. |
| LENGTH(g Geometry): | Returns a numerical value that is the length of polygon or linestring g. |

### *2.2.2 Functions that Test Spatial Relationships (Spatial Relationship Functions)*

Topological relationships have been studied extensively [10], [12]. Those included in SQL/SDA are EQUALS, DISJOINT, TOUCH, WITHIN, OVERLAP, CROSS, INTERSECTS, and CONTAINS. The formal definition for them can be found in [12], [32]. Besides topological relationships, possible spatial order relations are left/right or over/under and others [14]. Each of these functions returns a Boolean value: TRUE or FALSE.

### *2.2.3 Functions that Calculate Distance and Direction (Metric Functions)*

Distance and direction functions can be used to calculate the distance and direction between two values of type Geometry.

| | |
|---|---|
| Distance(g1 Geometry, g2 Geometry): | Returns the minimum Euclidean distance between g1 and g2. |
| Direction(g1 Geometry, g2 Geometry): | Returns the anticlockwise angle of g1 with reference to g2. |

The distance and direction functions are defined in many different ways in the GIS domain. We adopt the above definition just to show the expressive power of SQL/SDA.

### *2.2.4 Functions that Create a New Set of Spatial Features (Derivation Functions)*

This group of functions are different from the above because they create new spatial features/topology that can be acted on by further queries. Voronoi operation is also included as it is an important interpolation function.

| | |
|---|---|
| VORONOI (g1 Geometry): | Returns Voronoi polygons of g1. |
| BUFFER (g1 Geometry, d Double): | Returns a Geometry defined by buffering a distance d around g1, where d is in the distance units for the Spatial Reference of g1. |
| CONVEXHULL (g1 Geometry): | Returns a Geometry that is the convex hull of g1. |
| INTERSECTION (g1 Geometry, g2 Geometry): | Returns a Geometry that is the intersection of geometries g1 and g2. |
| DIFFERENCE (g1 Geometry, g2 Geometry): | Returns a Geometry that is the closure of the difference set of g1 and g2. |
| UNION (g1 Geometry, g2 Geometry): | Returns a Geometry that is the union set of g1 and g2. |

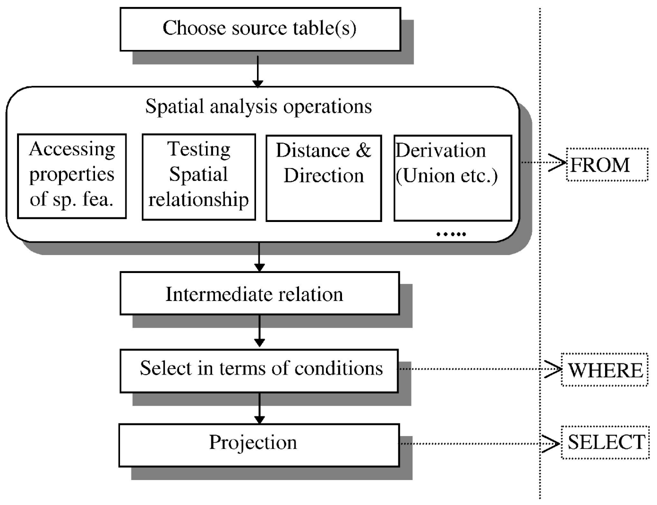

Choose source table(s)
Spatial analysis operations
Accessing properties of sp. fea.
Testing Spatial relationship
Distance & Direction
Derivation (Union etc.)
.....
FROM
Intermediate relation
Select in terms of conditions
WHERE
Projection
SELECT

Fig. 1. A general spatial analysis procedure by using current GIS packages.

| | |
|---|---|
| FUSION (g1 Geometry, g2 Geometry): | Returns a Geometry that merges g1 and g2 if g1 and g2 are adjacent and have equal user-specified attribute value. |

The above group of functions is essential to spatial analytic capabilities of GIS because spatial analysis implies creating new mapped data—either feature characteristics or new spatial partitioning [3]. It also represents the most difficult type of spatial functions to define directly in SQL [23]. Therefore, the emphasis of SQL/SDA should be put on the incorporation of such functions in a proper way.

## 3 Presentation of SQL/SDA

SQL/SDA is designed to align with the general spatial analysis procedure using current GIS packages. Such a procedure is close to the way most users conceive in analyzing spatial data. The following introduces this procedure first, then describes the overall structure and syntax of SQL/SDA.

### 3.1 A General Spatial Analysis Procedure Using Current GIS Packages

When utilizing a commercial GIS package such as Arc/Info to solve a query associated with several spatial analysis functions, we often first carry out spatial operations step by step. These operations may serve for accessing properties of spatial features, testing spatial relationship, calculating distance, and direction or deriving new spatial features. After this, an intermediate relation combining all the results of spatial operations is obtained, on which the selection operation with certain conditions can then be conducted and, finally, the desired attributes are projected. This procedure (see, also, [25]) is reflected in the design of SQL/SDA (left part of Fig. 1).

### 3.2 Structure and Syntax of SQL/SDA

Generally, an SQL query is in the form of "SELECT ... FROM ...WHERE," which corresponds to the relational algebraic operations: projection, Cartesian product, and selection, respectively. In any case, the FROM clause needs to create a single relation, as this is an intermediate or virtual relation to which the selection in the WHERE clause and the projection in the SELECT clause are applied.

Likewise, if the FROM clause is restructured to create an intermediate relation resulted by spatial operations, i.e., the Cartesian product of source relations is extended, the SQL statement can then be easily adapted to the above spatial analysis procedure (Fig. 1), while conforming well to the SQL design concepts.

Spatial operations between different relations (maps) are basically operations between spatial attributes of features in these relations, and each of these operations yields a value either geometric or numeric (including Boolean value). Based on this, an approach to creating the intermediate relation is to take the results of various spatial operations involved as new derived attributes, and to append such attributes to the Cartesian product of source relations. Since the direct insertion of spatial functions in the FROM clause, such as "FROM INTERSECTION(lu.Location, sl.Location)," will not be consistent with the SQL92 standard, it is necessary to employ a subquery, i.e., embedding an SQL statement in the FROM clause [25]. The spatial functions are, therefore, applied in the nested SELECT clause to produce an intermediate relation. For example, a query (*) is expressed by SQL/SDA as follows:

```
SELECT lu.ID, sl.ID, ILocation, areaval
FROM                                                   (*)
     (SELECT *, OVERLAP(lu.Location, sl.Location)
     AS overlapval,
                 INTERSECTION(lu.Location, sl.Location)
                   AS ILocation,
                 AREA(ILocation) AS areaval
       FROM landuse AS lu, soil AS sl)
WHERE lu.type = 'Brushland' and sl.type = 'A'
       and overlapval = True and areaval > 700
       and areaval < 900
```

This query is to display the land parcels and their corresponding area on the condition that the landuse type of each parcel is brushland and soil type is 'A' and area is between 700 hectares to 900 hectares.

The subquery in the above FROM clause creates an intermediate relation (Fig. 2), on which a selection with certain conditions is carried out. In the intermediate relation, the results of three kinds of operations, namely, OVERLAP (spatial relationship), INTERSECTION (derivation function), and AREA (property function), are Boolean, geometric, and floating values represented by the attributes "overlapval," "ILocation," and "areaval," respectively. If the value of "overlapval" is "True," the value of ILocation will be a new geometry, else NULL.

In the subquery, the three different spatial functions whether employed for accessing the property of a spatial feature, determining spatial relationship, or deriving new

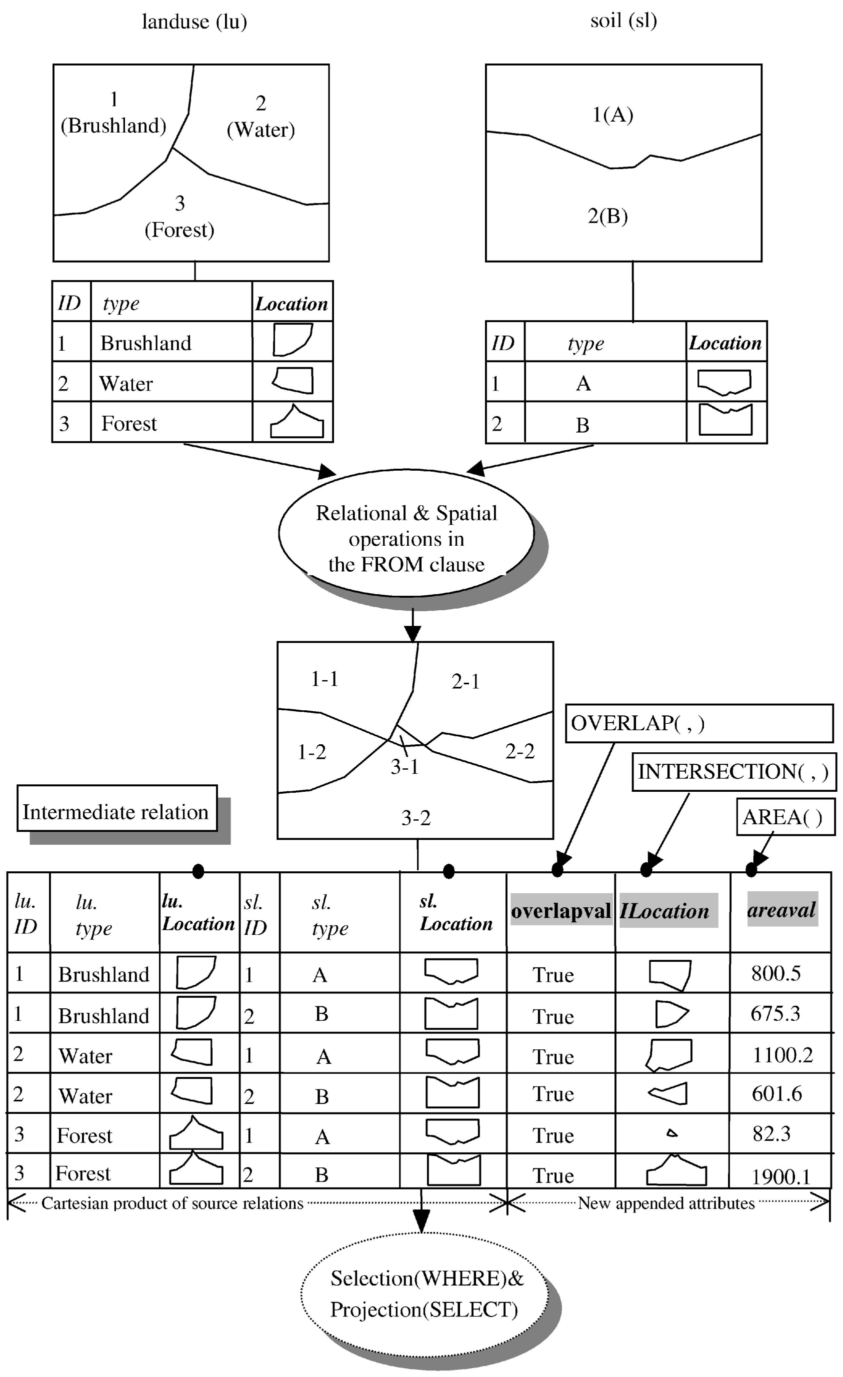


Fig. 2. The intermediate relation of the FROM clause in query (*).

spatial features are incorporated in SQL/SDA in a uniform way. The spatial constraints still occur in the WHERE clause, and the derived attributes are utilized in formulating such constraints (e.g., overlapval = True and *areaval* > 700 and *areaval* < 900). This reflects that SQL/SDA has not changed the semantics of a query as expressed by the previous extended SQLs.

In effect, just like the relational Cartesian product in the conventional FROM clause of SQL, the intermediate relation created by the subquery is a virtual result in that it can be optimized in terms of projection items and selection conditions during the implementation process.

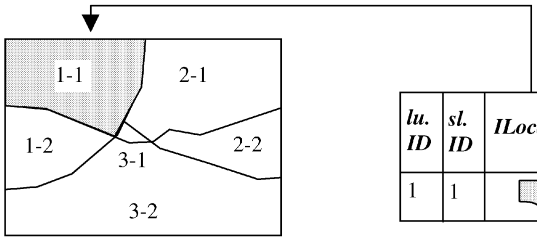
The result of the query (*) is illustrated in Fig. 3.

The syntax of SQL/SDA is based on the conventional SQL. Its basic form is

```
SELECT <select-clause>
FROM <from-clause>
WHERE <where-clause>
```

Since only the FROM clause is different from other extended SQLs, its BNF form is given below:

```
<from-clause> :: = <nested SQL>{, <tables>}
<nested SQL> :: = SELECT <sub-select clause>
                  FROM <fea-tables>
<fea-tables> :: = fea-table {, <fea-tables>}
```



| *lu. ID* | *sl. ID* | *lLocation* | *areaval* |
|---|---|---|---|
| 1 | 1 | | 800.5 |

Fig. 3. The result of the query (*).

```
<sub-select clause> :: = *, <spatially derived attributes>
<spatially derived attributes> :: =  <spatial functions>
                                     AS <attribute name>
                                     {, <spatially derived
                                     attributes> }
<spatial functions> :: = < property functions> | <spatial
                            relationship functions> |
                            <metric functions> |
                            <derivation functions>
<tables> :: = table {, <tables>}
```

The spatially derived attributes are treated in the same way as those in the source relations and, thus, can be easily applied as constraints in the main WHERE clause, and/or referenced for further analysis, aggregation, or graphical display in the main SELECT clause.

Since SQL/SDA needs to comply with both the general spatial analysis procedure and the SQL concepts, the subquery in the FROM clause is employed. In fact, the subquery in SQL/SDA is in a template-like format and can be easily composed with the help of a visual interface (see details later in Section 5.1). Such a subquery would not extend more than one level. Therefore, known problems of nested query [28] would not exist.

## 4 Query Examples

GIS has a vast number of applications. The following two examples, which are concerned with some typical spatial analysis problems, i.e., site selection and land suitability evaluation, are selected to demonstrate how they are presented by SQL/SDA.

**Example 1.** This query is to select a lab site. The selection criteria are:

- Preferred landuse is brushland.
- Soiltype should be "A."
- Site must be within 300 meters of existing sewer lines.
- Site must be beyond 20 meters of existing streams.
- Site must contain an area at least 2,000 square meters.

Using SQL/SDA, this query is formulated as:

```
SELECT labLocation
FROM
  (SELECT *, INTERSECTION(lu.Location, sl.Location)
    AS lsLocation,
       BUFFER(sw.Location, 300) AS buf1Location,
       INTERSECTION(lsLocation, buf1Location)
         AS lsbLocation,
       BUFFER(sm.Location, 20) AS buf2Location,
       DIFFERENCE(lsbLocation, buf2Location)
         AS labLocation,
       AREA(labLocation) AS areaval
   FROM  landuse AS lu, soil AS sl, sewer AS sw, stream
     AS sm)
WHERE lu.type = 'brushland' and sl.type = 'A' and
  areaval > 2000
```

In this query, different maps (i.e., landuse, soil, sewer, and stream) are represented by tables and INTERSECTION, BUFFER, DIFFERENCE, and AREA operations are employed. Through the link of intermediate values, these operations are conducted step by step and, finally, the candidate sites satisfying all the requirements are selected. The result of this query using real data will be shown in Fig. 8.

**Example 2.** The last query concerns site selection. This one is related to land suitability evaluation for building an institute, in which all the possible sites need to be classified into different suitability levels.

Assuming that there are two maps: landuse and soil, and the suitability levels are "high (III)," "medium (II)," and "low (I)."

The evaluation includes the following steps:

a. Overlay the landuse map and soil map.
b. Classify the overlay map in terms of the evaluation criteria:
   - If the landuse type is "Brushland" and soil type is "A," then the suitability level is "III."
   - If the landuse type is "Water" and soil type is "A," then the suitability level is "I."
   - Otherwise the suitability level is "II."
c. Merge the parcels (area > 100 hectares) with the same suitability level and display them.

This query is formulated in SQL/SDA as follows:

```
SELECT FUSION(ILocation)
FROM
  (SELECT *, INTERSECTION(lu.Location, sl.Location)
    AS ILocation,
           AREA(ILocation) AS  areaval
           classfyval = (CASE lu.type | | sl.type
                        WHEN 'BrushlandA' THEN 'III'
                        WHEN 'BrushlandB' THEN 'II'
                        WHEN 'WaterA' THEN 'I'
                        WHEN 'WaterB' THEN 'II'
```

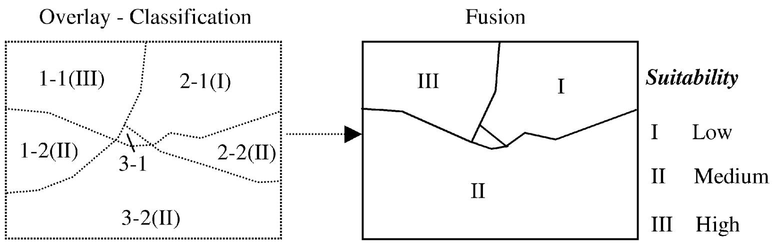


Fig. 4. The result of query Example 2.

```
                    WHEN 'ForestA' THEN 'II'
                    WHEN 'ForestB' THEN 'II'
                  END)
    FROM landuse AS lu, soil AS sl)
WHERE ILocation < > NULL and areaval > 100
GROUP BY classfyval
```

In this query, the INTERSECTION operation is employed to obtain the basic parcels that have the attributes of both landuse and soil type. After the classification of each overlaid parcel, the parcels with the same "classfyval" are merged. This is performed by the FUSION function in the SELECT clause coupled with the GROUP BY clause. Using the landuse and soil maps in Fig. 2, the result of this query is shown in Fig. 4. The central small polygon is not labeled with a suitability level because its area is less than 100 hectares.

It is found that in this case, some spatial functions (i.e., INTERSECTION, AREA, and CLASSIFICATION) occur in the subquery of the main FROM clause and some others in the main SELECT clause (i.e., FUSION). This shows a comprehensive use of spatial functions, which integrates the approaches in the previous spatially extended SQLs.

Apparently, if the FROM clause, i.e., the subquery, is not developed, it will be quite difficult to express such a query because

a. The classification value (classfyval) is intermediate. Such a transitional value need not occur in either the SELECT clause or the WHERE clause.
b. The classification operation must be conducted after the overlay operation (i.e., INTERSECTION operation) since it needs to act on the overlaid parcels that have both landuse-type and soil-type attributes.

Consequently, the above example shows an appropriate way in accommodating spatial functions step by step in the subquery of the FROM clause.

## 5 Implementation of SQL/SDA on the Web

The Internet has been changing the ways of information access, sharing, and dissemination. It will further change the means of analysis and visualization of spatial data [9]. SQL/SDA is, therefore, prototyped in a Web environment to adapt to such changes.

Currently, there have been a number of different approaches implementing Web GIS [9]. These are summarized as:

- Plug-ins by which programs are downloaded into the browser environment and are executed as a linked module to the client with the data also being local.
- Functions invoked from the browser are executed on the server (HTML/CGI model) with the results being sent back to the browser in the form of a gif file.
- A Java-based viewer and querying environment where the data is downloaded to the client for local processing. Such a client can also be developed by using ActiveX controls, which are, however, only applicable in the Microsoft Windows environment.
- Hybrid models where certain functions are executed on the server (data are also on the server side) and some others on a Java client by local processing.

Each approach has its pros and cons, which has been analyzed in [36]. In order to provide a flexible and user friendly interface while making better use of the power of the server, the fourth approach (i.e., the hybrid model), which integrates the method in both the client side and server side, is adopted in designing the prototype of SQL/SDA. The SQL/SDA client, downloaded over the Internet and executed by the Java-enabled Web browser (e.g., Netscape or Internet Explorer) on the user's machine, is a Java application program which provides an interface for users to compose their queries and view the query results. The SQL/SDA server, coupled with the Web server, is dedicated to query processing including parsing, optimization, and execution with the Spatial Database Engine (SDE) [39]. The Java-based client communicates with the SQL/SDA server through a specifically designed protocol that allows a user to control over the query execution. The architecture of SQL/SDA's prototype is illustrated in Fig. 5.

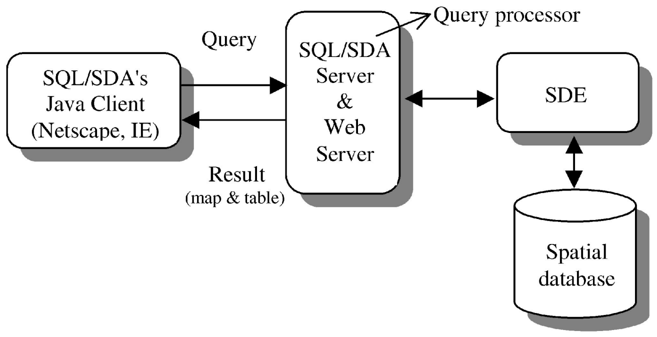

Query processor
Query
SQL/SDA's Java Client (Netscape, IE)
SQL/SDA Server & Web Server
SDE
Result (map & table)
Spatial database

Fig. 5. The architecture of SQL/SDA's prototype.

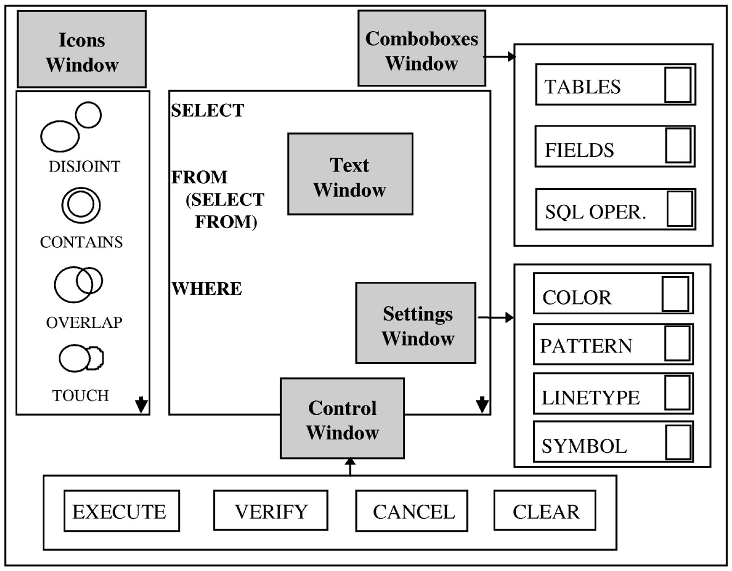


Fig. 6. Window layout of SQL/SDA's interface.

## 5.1 Query Interface

Although SQL/SDA, an extended SQL, is indeed a textual language that is often considered incompatible with the visual language, the advantages of visual query languages such as intuitiveness and ease [7], [8], [15], [27], [29], [35] can still be introduced in the interface design of SQL/SDA for reducing the tediousness of inputting SQL/SDA commands only by typing. Such an interface can be implemented by using visual constructs in the Java language, such as icons, listboxes, and comboboxes [30]. The query interface of SQL/SDA is shown in Fig. 6.

The interface is composed of five windows:

a. the text window,
b. the control window,
c. the comboboxes window,
d. the icons window, and
e. the settings window.

The *text window* is where the user enters the SQL/SDA commands in the SELECT ... FROM (SELECT - FROM) ... WHERE block.

The *control window* contains four command buttons. The "execute" button passes the SQL/SDA commands to the server. After execution, the resulting map and attribute table are displayed. The "verify" button checks the commands' grammatical correctness. If there is any syntactic error, an error message will be shown in a pop-up window. The "clear" button clears the text window and removes the selected spatial objects from both the map and attribute display windows.

The *comboboxes window* contains three comboboxes. The "tables" stores both the conventional and spatial tables. Once an item in this combobox is selected, its corresponding fields are added to the "fields" combobox. If two or more tables are selected, the field name is in the form of table_name.attribute_name. The "SQL OPER." combobox lists all the predicates and operators used in standard SQL. If any item in the above four comboboxes is selected, its corresponding text expression will be generated at where the cursor locates in the text window (Fig. 7). Hence, the manual input by typing is eliminated and the likely syntactic errors are reduced as well.

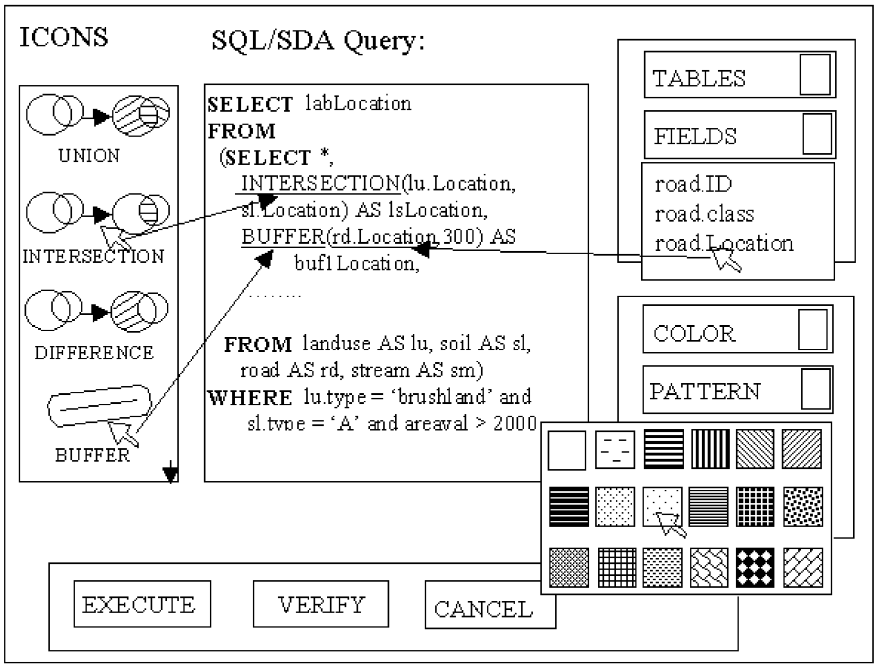
The *icons window* contains icons for all the spatial functions in SQL/SDA. The items in this listbox differ from ordinary listbox in that each of them combines an icon with its textual description. If any item in this icons window is chosen, its corresponding textual expression will also be generated at where the cursor locates in the text window (Fig. 7). It is apparent that the icons window and the comboboxes window are used together to help users formulate SQL/SDA queries.

The *settings window* sets the graphical output of the query with colors, patterns, linetypes, and symbols, which are realized by the comboboxes "Color," "Pattern (polygon)," "Linetype (line)," and "Symbols (point)," respectively. The patterns for polygon-type data output are shown in Fig. 7.

## 5.2 Query Server

The query server is primarily a query processor, which communicates with spatial database via SDE. The query processor mainly includes a parser and an optimizer. SDE contains a wealth of Application Programming Interface (API) functions for spatial features processing. Generally, the functions listed in Section 2 can find its counterparts in SDE. Therefore, we utilize the SDE's open API to develop the spatial functions in the query processor. The spatial features are stored and managed in Oracle databases based on SDE's feature data model.

After a user submits a query, the query processor receives the query commands via the Web server and starts to parse and optimize it. The format and structure of SQL/SDA shows that the optimization can be done primarily on the intermediate relation created by the subquery in the FROM clause. During the optimization, an important step is to decompose the query conditions in the WHERE clause into a nonspatial condition part and a spatial condition part. The nonspatial condition part does not include spatial attributes, both original and derived, while spatial condition part includes spatial attributes. Take query (*) in Section 3.2 as an example,



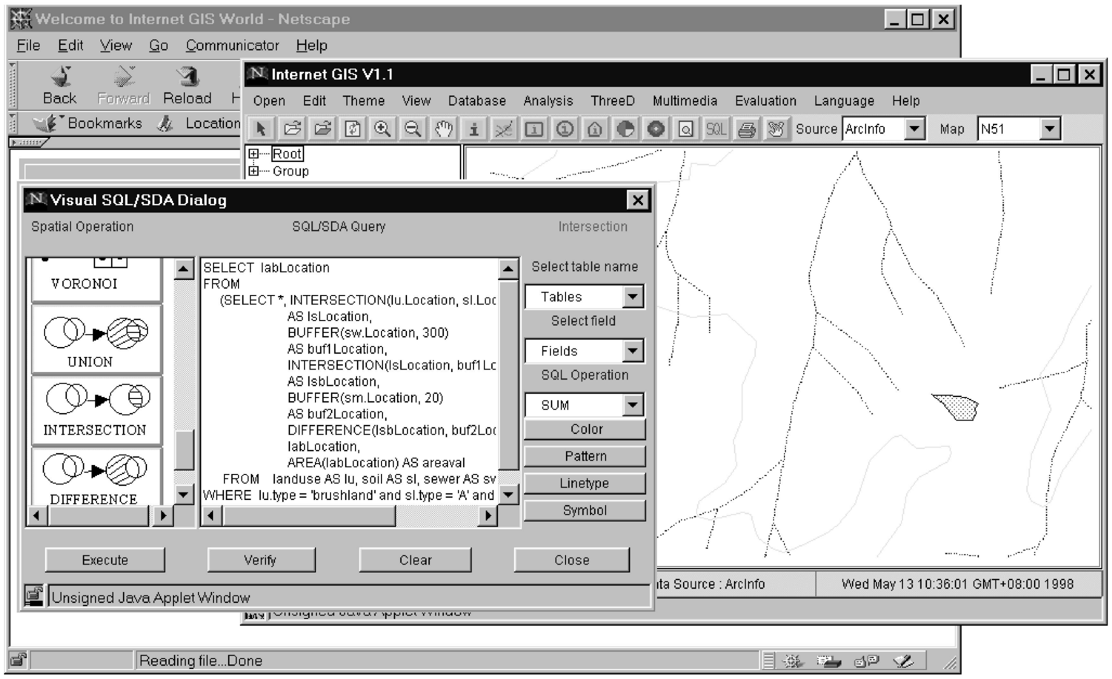
```
SELECT labLocation
FROM
  (SELECT *,
    INTERSECTION(lu.Location,
    sl.Location) AS lsLocation,
    BUFFER(rd.Location,300) AS
        buf1Location,
    ........

  FROM landuse AS lu, soil AS sl,
    road AS rd, stream AS sm)
WHERE lu.type = 'brushland' and
    sl.type = 'A' and areaval > 2000
```

Fig. 7. The assistance of visual constructs for building SQL/SDA queries.

after selection in terms of nonspatial conditions ("lu.type = 'Brushland' and sl.type = 'A'"), only record 1 (the first row) in the intermediate relation is left, on which the selection in terms of spatial conditions ("overlapval = True and areaval > 700 and areaval < 900") can then be carried out.

After optimization, the best sequence of decomposed queries [31] is selected. When spatial functions need to be executed, SDE is employed. Finally, the query result is obtained and transmitted to the SQL/SDA client for display. In SQL/SDA, the spatial data in transmission is currently in vector data format.

Through the above method, the result of query Example 1 in Section 4 is shown in Fig. 8. The shaded polygon in the right window titled "Internet GIS V1.1" is the candidate laboratory site. The browsing of the query result, such as zoom in, zoom out, or pan, is also conducted in this window. The left window titled "Visual SQL/SDA Dialog" is the interface for users to build queries.



```
SELECT labLocation
FROM
   (SELECT *, INTERSECTION(lu.Location, sl.Loc
                  AS lsLocation,
                  BUFFER(sw.Location, 300)
                  AS buf1Location,
                  INTERSECTION(lsLocation, buf1Lc
                  AS lsbLocation,
                  BUFFER(sm.Location, 20)
                  AS buf2Location,
                  DIFFERENCE(lsbLocation, buf2Loc
                  labLocation,
                  AREA(labLocation) AS areaval
    FROM   landuse AS lu, soil AS sl, sewer AS sv
WHERE lu.type = 'brushland' and sl.type = 'A' and
```

Source: ArcInfo
Map: N51
Select table name: Tables
Select field: Fields
SQL Operation: SUM
ta Source : ArcInfo
Wed May 13 10:36:01 GMT+08:00 1998
Unsigned Java Applet Window
Reading file...Done

Fig. 8. An experiment result.

It should be noted that spatial indexing, which is important to query optimization, is not considered here since spatial functions (API) are executed by the use of SDE, which has already provided spatial indexing method. For more details about spatial indexing and spatial query optimization, see [19], [31].

## 6 Comparisons with Other Related Languages

### 6.1 Comparison with Macro Languages of GIS

Like a batch file (.bat), a macro language of GIS package is composed of procedural commands and is often employed to perform a series of GIS operations. For instance, using SML in PC Arc/Info, the query in Example 1 of Section 4 can be presented as:

*IDENTITY* landuse soil landsoil
*RESELECT* landsoil POLY landtype = 'brushland' and soiltype = 'A'
*BUFFER* sewer bufsew # # 300
*IDENTITY* landsoil bufsew landsoilbuf
*BUFFER* stream bufstrm # # 20
*ERASECOV* landsoilbuf bufstrm labsite
*RESELECT* labsite POLY area > 2000

As shown in the above example, macro languages of GIS are essentially a kind of procedural language, provided for users to customize their specific applications. The advantages of SQL over procedural languages have been elaborated in database literature (e.g., [11], [47]). In general, these advantages are also applicable to SQL/SDA.

Based on the widely accepted SQL, SQL/SDA incorporates all its commands in the familiar structure of "SELECT ...FROM...WHERE" in a natural language-like style. This facilitates the formulation of queries considerably, especially when a visual interface is set up to assist with it. However, a macro language of GIS usually consists of a large number of unstandardized concepts and special-purpose functions, which makes it difficult to learn, especially for non-GIS specialists. Moreover, most macro languages of GIS just conduct the operation commands in such a mechanical way that no optimization strategy is employed during their execution.

### 6.2 Comparison with Previous Spatially Extended SQLs (SESQL)

After studying the previous SESQLs mentioned in Section 1, it is found that these languages are designed to retrieve data mainly in terms of spatial relationship and metric constraints. The important analysis functions—derivation functions (e.g., UNION, INTERSECTION or DIFFERENCE) are generally not accommodated in these languages since their intention is not particularly upon spatial data analysis, but spatial data representation and management. Derivation functions, also called set functions, are considered very difficult to be incorporated into the SQL framework because these functions, creating new sets of spatial features, are at the same level as projection, Cartesian product, and selection operations [21].

Different from the previous SESQLs, the primary concern of SQL/SDA is addressed to present the complicated multistep spatial analysis problems. As a consequence, it extends the standard SQL from another perspective, namely, extending the Cartesian product in the FROM clause for the sake of adapting itself to the general spatial analysis procedure performed by current GIS packages, while complying with the SQL design concepts. The FROM clause of SQL/SDA embeds a subquery which produces an intermediate relation. In this intermediate relation, the results of the spatial functions, which could be the properties spatial relationship and metric functions, including the derivation functions, are represented by new attributes and appended to the Cartesian product of source relations. In other words, various kinds of spatial functions are incorporated in the SQL framework in a more consistent way, which enables users to apply the necessary spatial operations with relative ease.

In the previous SESQLs, spatial functions are either applied in the main SELECT clause or the WHERE clause. This is, however, still insufficient because:

a. It is quite difficult to express some kinds of queries even for SQL specialists, for instance the query example 2 in Section 4. The main reason for this is that some transitional values are required after spatial operations, but they do not necessarily occur in both the main SELECT clause and WHERE clause (see details in Section 4). A better way is to apply such a value in the subquery of the FROM clause.
b. The value of any spatial function occurring in the WHERE clause cannot be referenced in the SELECT clause, or vice versa unless the spatial function with operands is written again. For example, we assume a query as follows:

SELECT *INTERSECTION(lu.Location, sl.Location)*,
  AREA*(INTERSECTION(lu.Location, sl.Location))*, ......
FROM landuse AS lu, soil AS sl
WHERE lu.type = 'Brushland' and sl.type = 'A' and
  (AREA*(INTERSECTION(lu.Location, sl.Location))*
   *> 700 and*
  AREA*(INTERSECTION(lu.Location, sl.Location))*
   *< 900) or*
  (AREA*(INTERSECTION(lu.Location, sl.Location))*
   *> 1000 and*
  AREA*(INTERSECTION(lu.Location, sl.Location))*
   *< 1200) ......*

As shown in the above example, the "INTERSECTION" operation occurs repeatedly, leading to the complex and redundant query presentation. Moreover, the INTERSECTION operation occurring in the WHERE clause does not conform well to the SQL concepts because such a derivation function usually creates a new set of spatial features. These problems will be worsened if a query is more complicated or a spatial function needs to be used more times. This is because only the attributes, whether original or derived, in the FROM clause can be employed in both the SELECT and WHERE clauses as well as GROUP BY and HAVING clauses.

Nevertheless, when a spatial query only deals with spatial relationship and metric functions and these functions need to be employed only once, the approaches in the previous SESQLs are more suitable. Consequently, in such cases, the strategy of SQL/SDA is to integrate the previous approaches. This has been shown by Example 2 in Section 4. It is demonstrated that SQL/SDA has explored the usage of the FROM clause in formulating queries and, therefore, stretches the capabilities of the previous SESQLs.

Apart from the above aspect of query's presentation, the implementation of SQL/SDA is also different from the previous SESQLs. SQL/SDA is prototyped in the Web environment, which allows the access to spatial analysis functions over the Internet. This approach has rarely been reported before. In addition, SQL/SDA's Java-based visual interface is characterized by its combination of visual constructs and textual language.

In summary, SQL/SDA has addressed an important problem in spatial databases, i.e., to offer spatial analytic capabilities via an extended database language and a concrete implementation approach for this language has been presented.

## 7 Conclusions

Spatial processing on the Internet is currently developed in full swing. Since spatial data access and transmission across the Web are still rudimentary, an important trend of the Web GIS development is to enhance the access to spatial analysis tools or functionalities over the Internet that is essential for decision making based on georeferenced data. Although the formulation of a spatial analysis problem via a query expression is not always obvious, the design of SQL/SDA has been in an effort to enable landuse managers, natural resource, and environmental managers, and other users to use and analyze spatial data over the Internet.

Within the framework of the ongoing spatial SQL standards, an appropriate structure for SQL/SDA is explored, which fits the general spatial analysis procedure. Unique to SQL/SDA is the ability to incorporate not only the spatial relationship and metric functions, but also the important derivation functions in a uniform way. Besides the SELECT and WHERE clauses, the functionalities of the FROM clause are explored. The FROM clause is restructured to operate when the results from spatial operations need to be acted on by further queries. This new extension makes SQL/SDA particularly suited for presenting complicated multistep spatial analysis problems. It is hoped such an approach to extending SQL can also shed light on SQL extensions for other applications.

The interface of SQL/SDA incorporates the visual language style by introducing a set of visual constructs, which effectively aid the composition of SQL/SDA queries. The visualization of spatial operations helps to understand the semantics of a specific spatial operation and, hence, reduces the possible syntactic errors. This interface is programmed in the Java language, which makes the interactive visual interface, previously not available using static CGI method, possible. Moreover, the advantages of Java applications such as independencies on any operating system or windowing environment could be shared.

The prototype of SQL/SDA is implemented on the Internet based on a client/server architecture. The effective allocation of appropriate functions to the client and server sides contributes to the overall performance boost of spatial data analysis over the Internet.

Spatial analysis tasks are sometimes very broad and complicated. It would not be realistic to conceive that any kind of analysis problems could be handled with a declarative language like SQL/SDA. It may be found that the functions provided by SQL/SDA are insufficient to support environmental models which require complex mathematical modeling and are coupled with GIS. Therefore, a better alternative for SQL/SDA design is to make itself extensible. Using an extensible SQL would be a suitable starting point to explore this problem domain.

There are still some other aspects which need to be further studied such as more efficient query optimization, caching of the spatial operation result and use of metadata. In addition, the query interface can probably be made friendlier. For example, in many cases, the subquery can be formulated automatically after a user selects the necessary operations and operands. This would increase the intelligent level of query composition.

## Acknowledgments

This research is partially supported by the Research Grants Council (RGC) of the Hong Kong Special Administrative Region of the People's Republic of China government under the RGC Earmarked Research Grant No. CUHK 150/96H and No. 4334/98E, and CUHK RAC under Grant No. 4720401.

## References

[1] D.J. Abel, "Spatial Internet Marketplaces: A Grand Challenge," *Advances in Spatial Databases,* M. Scholl and A. Voisard, eds., 1997.
[2] W. Aref and H. Samet, "Extending a DBMS with Spatial Operations," *Advances in Spatial Databases,* O. Guenther and H.J. Schek, eds., pp. 299-318, 1991.
[3] J.K. Berry, "Fundamental Operations in Computer-Assisted Map Analysis," *Int'l J. Geographical Information Systems,* vol. 1, no. 2, pp. 119-136, 1987.
[4] P. Boursier and M. Mainguenaud, "Spatial Query Languages: Extended SQL vs. Visual Language vs. Hypermaps," *Proc. Fifth Int'l Symp. Spatial Data Handling,* 1992.
[5] T. Bruns and M. Egenhofer, "User Interfaces for Map Algebra," *J. Urban and Regional Information Systems,* vol. 9, no. 1, pp. 44-54, 1997.
[6] K.P. Chang, "The Design of a Web-Based Geographic Information System for Community Participation," 1997, http://www.geog.-buffalo.edu/~kchang/ma1.html.
[7] N.S. Chang and K.S. Fu, "Query-by-Pictorial-Example," *IEEE Trans. Software Eng.,* vol. 6, no. 6, pp. 519-524, June 1980.
[8] S.K. Chang, "Visual Languages: A Tutorial and Survey," *IEEE Trans. Software Eng.,* vol. 4, no. 1, pp. 29-39, Jan. 1987.
[9] *Changing the Face of GIS,* Genasys, Inc., 1997, http://www.genasys.com/homepage/info/publications.html.
[10] E. Clementini, D. Paolino, and P. Oosterom, "A Small Set of Formal Topological Relationships Suitable for End-User Interaction," *Advances in Spatial Databases,* D.J. Abel and B.C. Ooi, eds., pp. 277-295, 1993.

[11] C.J. Date, *An Introduction to Database Systems,* second ed. Addison-Wesley, 1983.
[12] M. Egenhofer and R. Franzosa, "Point-Set Topological Spatial Relations," *Int'l J. Geographical Information Systems,* vol. 5, no. 2, pp. 161-174, 1991.
[13] M. Egenhofer, "Why not SQL!," *Int'l J. Geographical Information Systems,* vol. 6, no. 2, pp. 71-85, 1992.
[14] M. Egenhofer, "Spatial SQL: A Query and Presentation Language," *IEEE Trans. Knowledge and Data Eng.,* vol. 6, no. 1, pp. 86-95, Jan./Feb. 1994.
[15] M. Egenhofer, "Query processing in Spatial-Query-by-Sketch," *J. Visual Languages and Computing,* vol. 8, no. 4, pp. 403-424, 1997.
[16] A. Frank, "Mapquery—Database Query Languages for Retrieval of Geometric Data and Its Graphical Representation," *ACM Computer Graphics,* vol. 16, no. 3, pp. 199-207, 1982.
[17] *GIS and Environmental Modelling: Progress and Research Issues,* M.F. Goodchild, L.T. Steyaert, B.O. Parks, C. Johnston, D. Maidment, M. Crane, and S. Glendinning, eds. GIS World Books, 1996.
[18] M.F. Goodchild, "A Spatial Analytical Perspective on Geographical Information Systems," *Int'l J. Geographical Information Systems,* vol. 1, no. 4, pp. 327-334, 1987.
[19] O. Guenther, *Environmental Information Systems.* Berlin/Heidelberg/New York: Springer-Verlag, 1998.
[20] O. Guenther and R. Mueller, "From GISystems to GIServices: Spatial Computing in the Internet Marketplace," *Interoperability in Geographic Information Systems,* M. Egenhofer, R. Fegeas and M. Goodchild, eds., 1998.
[21] R.H. Gueting and M. Schneider, "Realm-Based Spatial Data Types: The ROSE Algebra," *VLDB J.,* vol. 4, pp. 243-286, 1995.
[22] R.H. Gueting, "Geo-Relational Algebra: A Model and Query Language for Geometric Database System," *Proc. Int'l Conf. Extending Database Technology,* J.W. Schmidt, S. Ceri and M. Missikoff, eds., pp. 506-527, 1988.
[23] J.R. Herring, R.C. Larsen, and J. Shivakumar, "Extensions to the SQL Query Language to Support Spatial Analysis in a Topological Database," *Proc. GIS/LIS' 88,* 1988.
[24] B. Huang, "Design of an Object-Relational Data Model for the Fully Integrated GIS," PhD dissertation, Inst. of Remote Sensing Applications, Chinese Academy of Sciences, Beijing, 1997.
[25] B. Huang and H. Lin, "Restructuring the SQL Framework for Spatial Queries," *Geographical Information Science,* vol. 3, no. 1, pp. 42-50, 1997.
[26] K.J. Ingram and W.W. Phillips, "Geographic Information Processing using a SQL Based Query Language," *Proc. AutoCarto 8,* pp. 326-335, 1987.
[27] Y.C. Lee and F.L. Chin, "An Iconic Query Language for Topological Relationships in GIS," *Int'l J. Geographical Information Systems,* vol. 9, no. 1, pp. 25-46, 1995.
[28] W.S. Luk and S. Kloster, "ELFS: English Language From SQL," *ACM Trans. Database Systems,* vol. 11, no. 4, pp. 447-472, 1986.
[29] M. Mainguenaud and M. Portier, "Cigales: A Graphical Query Language for Geographical Information Systems," *Proc. Synchronous Digital Hierarchy (SDH '90),* K. Brassel and H. Kishimoto, eds., 1990.
[30] P. Naughton and M. Morrison, *The Java Handbook.* Osborne/McGraw-Hill, 1996.
[31] B.C. Ooi, *Efficient Query Processing for Geographic Information System.* Berlin: Springer-Verlag, 1990.
[32] *OpenGIS Simple Features Specification for SQL: Revision 1.* Open GIS Consortium (OGC), Inc., 1998, http://www.opengis.org/techno/specs.htm.
[33] *OpenGIS Abstract Specification: An Object Model for Interoperable Geoprocessing, Revision 1.* Open GIS Consortium (OGC), Inc., OpenGIS Project Document Number 96-015R1, 1996.
[34] S. Openshaw and C. Openshaw, *Artificial Intelligence in Geography,* New York: Wiley, 1997.
[35] D. Papadias and T.K. Sellis, "A Pictorial Query-by-Example Language," *J. Visual Languages and Computing,* vol. 6, no. 1, pp. 53-72, 1995.
[36] Z.R. Peng, "An Assessment of the Development of Internet GIS," *Proc. ESRI User Conference,* 1997.
[37] N. Roussopoulos, C. Faloutsos, and T. Sellis, "An Efficient Pictorial Database System for PSQL," *IEEE Trans. Software Eng.,* vol. 14, no. 5, pp. 639-650, May 1988.
[38] M. Scholl and A. Voisard, "Thematic Map Modeling," *Proc. First Int'l Symp. Large Spatial Databases,* 1989.
[39] *SDE Developer's Guide, Version 2.1.* Redlands, Calif.: Environmental Systems Research Institute (ESRI), Inc., 1996.
[40] *Spatial Analysis and GIS.* S. Fotheringham and P. Rogerson, eds., London: Taylor & Francis, 1994.
[41] *Spatial Analysis: Modeling in a GIS Environment.* P. Longley and M. Batty, eds., Cambridge: Geoinformation Int'l, 1996.
[42] *Spatial Analytical Perspectives on GIS.* M. Fischer, H.J. Scholten, and D. Unwin, eds., London: Taylor & Francis, 1996.
[43] "SQL Multimedia and Application Packages (SQL/MM, Part3: Spatial)," *ISO Working Draft,* Sept. 1995.
[44] P. Svensson and Z.X. Huang, "Geo-SAL: A Query Language for Spatial Data Analysis," *Proc. Second Symp. Large Spatial Databases,* 1991.
[45] A.Y. Tang, T.M. Adams, and E.L. Usery, "A Spatial Data Model Design for Feature-Based Geographical Information Systems," *Int'l J. Geographical Information Systems,* vol. 10, no. 5, pp. 643-659, 1996.
[46] D.C. Tomlin, *Geographic Information Systems and Cartographic Modeling.* NJ: Prentice Hall, 1990.
[47] J.D. Ullman, *Principles of Database and Knowledge-base Systems.* Computer Science Press, 1988.
[48] A. Voigtmann, L. Becker, and K. Hinrichs, "A Query Language for Geo-Applications," Bericht Nr. 5/96-1, Institut fuer Informatik, Westf, Wilhelms-Univ. Muenster, Germany, 1996.

**Hui Lin** received the BSc degree in aerophotogrammetric engineering from Wuhan Technical University of Surveying and Mapping, China, in 1980, the MS degree in remote sensing and cartography from the Graduate School of Chinese Academy of Sciences in 1983, and the PhD degree in geographical information systems from the State University of New York at Buffalo in 1992. Dr. Lin is now director of the Joint Laboratory for GeoInformation Science, Chinese Academy of Sciences and The Chinese University of Hong Kong. He was elected in 1995 as academician of the International Eurasian Academy of Sciences. His current research projects include GIS interoperability, visualization of multidimensional data, and Web-based investment environment information system.

**Bo Huang** received the BSc degree in urban planning from Wuhan Technical University of Surveying and Mapping, China, in 1990, the Msc degree in urban geographical information system from the International Institute for Aerospace Survey and Earth Sciences, the Netherlands, in 1993, and the PhD degree in remote sensing and cartography from the Chinese Academy of Sciences in 1997. He is now a postdoctoral fellow at the Department of Geography, The Chinese University of Hong Kong. Dr. Huang's current research interests include Web-based GIS, geographic visualization, and spatial query languages.