CAR-TR-610
CS-TR-2865

IRI-9017393
March 1992

SPATIAL RELATIONS AND THEIR ALGEBRA

Walid G. Aref
Hanan Samet
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Studies
The University of Maryland
College Park, MD 20742-3411

Abstract

Spatial operations are classified in terms of selects and joins, thereby unifying spatial operations with relational selects and joins. Spatial and non-spatial operators are defined to act on what is termed a *spatial relation.* A spatial relation resembles a relation in a pure relational environment but is designed so that spatial selects and joins share the same algebraic properties as relational selects and joins and hence are indistinguishable from each other. Therefore, the algebraic transformation rules of relational algebra hold for mixed (spatial and relational) operators as well. This approach permits the usage of a conventional relational optimizer in a spatial database environment without a loss of efficiency in spatial query processing. Moreover, it permits the spatial aspect of the data to participate fully in the optimization process once an appropriate cost model is developed for estimating the cost of spatial as well as relational operators. From a different perspective, the concept of a spatial relation provides the flexibility of utilizing spatial data structures not only as *indexing mechanisms* to speed up the query processing but also as *containers* for organizing and manipulating the full descriptions of spatial objects. An algebra that has spatial relations as the basic entities and spatial and relational selects and joins as the basic operators is developed. In addition, some proofs are given that transformation rules hold for spatial operators as well.

This work was supported in part by the National Science Foundation under Grant IRI-9017393.

1 Introduction

In a typical spatial database environment, spatial objects (e.g., roads, rivers, lakes, etc.) are described by spatial attributes as well as non-spatial attributes. For efficient operation, we make the following assumptions. First, a set of objects that are spatially related to each other (e.g., the roads of a city) are logically clustered in some form in the database. Second, the instances of a spatial attribute are stored permanently in appropriate disk-based spatial data structures and are linked to their non-spatial counterparts [3]. Third, this augmented database environment is queried via a non-procedural language. In this paper, the ramifications of these assumptions are studied. The following questions are addressed:

- What is the basic data structure that is supported by this augmented non-procedural language (e.g., in a pure relational system, this data structure is a table)?
- What is its underlying algebra for both the spatial and database operations?

To answer these questions, the concept of a spatial relation serving as the basic data structure for an extended SQL-like non-procedural query language is introduced. A spatial relation is characterized by having spatial attributes and persistent spatial indexes that survive spatial and relational operations. In addition, a spatial relation captures in it the result of any spatial or relational operation. A spatial algebra is developed on top of spatial relations as an extension of relational algebra.

The approach taken in this paper is as follows. First, spatial operations are classified into selects and joins. Next, the spatial relation is designed in a way such that all the algebraic properties that hold for relational selects and joins also hold for spatial selects and joins (e.g., commutativity, associativity, etc.). This permits using the same transformation rules of relational algebra for handling spatial operations as well. Therefore, given an appropriate cost model, an optimizer that is used in conjunction with a relational system can still work in this augmented spatial environment. However, special implementation considerations have to be taken into account since the basic entity in this case is a spatial relation and not just a relation.

The rest of the paper is organized as follows. Section 2 classifies some spatial operations of interest into selects and joins, and gives an example spatial database that will be used throughout the paper. Section 3 presents the concept of a spatial relation which is composed of spatial and non-spatial attributes. Section 4 develops a spatial algebra. This includes defining formally the concept of a spatial relation, redefining spatial as well as relational operators that act on top of a spatial relation, developing expressions in this algebra and legal transformations (equivalences) among these expressions. The development of a spatial algebra serves as the basis for studying spatial query processing and optimization of mixed (i.e., having spatial and relational operations) queries. Section 5 discusses related work while Section 6 contains concluding remarks.

2 Spatial Operations

We classify spatial operations into selects and joins. These resemble the relational select and join operations in a relational database environment. Tables 1 and 2 list some spatial operations being classified into selects and joins, respectively. Notice the use of r_i, l_i and p_i to specify instance values of region, line and point spatial attributes, respectively. s_i is more general and is used to include any instance value of a region, line or point attribute.

We show some examples that clarify the above classification. Throughout the paper we will often refer to the schema definition given in Figures 1 and 2 which define the `land-use` and `roads` spatial database. Notice that attribute `location` is a spatial attribute of type `REGION` while attribute `road_coords` is a spatial attribute of type `LINE_SEGMENT`. We use an SQL-like syntax.

```
(create table land-use
 name  CHAR[40],
 address CHAR[100],
 location REGION,
 usage CHAR[40],
 zip_code NUMBER,
 importance NUMBER);
```

Figure 1: Land-use database schema

```
(create table roads
 road_id NUMBER,
 road_name CHAR(30),
 road_trafficability NUMBER,
 road_lanes NUMBER,
 road_width NUMBER,
 road_coords LINE_SEGMENT);
```

Figure 2: Roads database schema

Example (Spatial Select): The following query retrieves all the regions inside a given query window:

```
select all
  from land_use
 where in_window(location,x,y,width,height)
```

The `in_window` operation in the `where` clause is termed a spatial-based selection, or simply a *spatial select.* Spatial-based selection means that objects are selected by Boolean qualifications that refer to the objects' spatial attributes only.

Operating on multiple spatial attributes (and hence multiple spatial data structures) and querying on the various relationships between spatial objects introduces the concept of a *spatial join.*

Table 1: Some Spatial Selection Conditions

Spatial Selections	Description
in_window w	True if s_i lies inside the rectangular window w
in_circle c	True if s_i lies inside circle c
nearest_to p	True if s_i is nearest to point p
object_at p	True if s_i is located at point p

Table 2: Some Spatial Join Conditions

Spatial Join	Description
pass_through	True if l_i or r_i pass through s_j
adjacent_to	True if r_i is an adjacent neighbor of r_j
contained	True if s_i is contained in s_j
within n	True if s_i is within n units of distance from s_j
intersect	True if s_i and s_j overlap

Example (Spatial Join): Consider the following query, which retrieves all the regions within 5 miles from universities in the land-use database:

```
select all
  from land-use l, land-use k
 where within(l.location,k.location,5)
       and l.usage = "University"
```

There may exist more than one university in the database, and hence more than one tuple could be selected by the condition `l.usage = "University"`. Let the set of selected tuples be L. Then the spatial `within` condition generates the regions within 5 miles from each member of L. The result of the `within` condition should consist of a join relation. In particular, two tuples are merged if their corresponding spatial objects are within 5 miles of each other. The resulting relation contains all the attributes of the two participating relations, including the two spatial attributes (i.e., the `location` attribute).

We refer to the `within` condition as a *spatial join*. This is because it has the same effect as a regular join. Namely, a spatial join combines related entities from two entity sets into single entities whenever the combination satisfies the spatial join condition (e.g., if they are within n miles from each other).

3 The concept of a spatial relation

3.1 Attributes

A spatial object is described by two sets of attributes: spatial and non-spatial. A set of homogeneous spatial objects (i.e., of the same data type such as line data) that are spatially related to each other (e.g., are in proximity, or belong to a given region) is logically clustered in the database (e.g., stored together in database relations or in suitable spatial data structures).

For example, the schema definition shown in Figure 2 represents roads (homogeneous line data) in a given county (i.e., spatially related). The set of objects `roads` is described by the set of spatial attributes containing just one spatial attribute, namely `road_coords`, and the set of non-spatial attributes containing the other attributes in the schema. Different structures are used to store the data instances of the different sets of attributes.

A spatial data structure is associated with each spatial attribute in the schema and is used to store all data instances of that attribute over the set of homogeneous objects. The data structure is used as an index for spatial objects as well as a medium for performing spatially-related operations (e.g., rotation and scaling for images, point-in-region test, windowing, polygon intersection, area of a region, connected component retrieval, proximity queries, etc.). Depending on the attribute's data type (e.g., region, line, or point), a spatial data structure suitable for handling this type is selected.

The instances of a spatial attribute are merely some spatial indexes that point to the spatial description of the objects stored inside the spatial data structures. More details about how spatial and non-spatial descriptions of an object are linked to each other is presented in Section 3.2 (also, see [3]).

The data instances of the set of non-spatial attributes are stored in database relations. The spatial and non-spatial description of an object are linked to each other by what we term a *spatial relation* that is described in more detail in the following subsection.

3.2 The spatial relation

In standard SQL, the basic data structure that is supported is a *table*. In other words, each operation that is performed has a table as input and produces a table as output. Once the query has been processed in its entirety, the tuples in the table qualify for all the conditions stated in the query. Here, we discuss the form of the basic data structure which allows for spatial and non-spatial attributes as descriptors of spatial objects, given the assumptions stated at the beginning of Section 1.

Consider the following two example spatial queries that contain a mix of spatial and non-spatial conditions. The description of the spatial conditions involved in the query, along with other conditions, is given in Tables 1 and 2.

Example 1: Find roads that pass through an industrial region that is adjacent to a mountain which lies within 5 miles of an earthquake region.

```
select road_name m.name i.name
  from roads land-use m, land-use i, earthquake e
 where m.usage = "mountain"
       and i.usage = "industrial"
       and pass_through(road_coords,i.location)
       and adjacent_to(i.location,m.location)
       and within(m.location,e.location,5)
```

Example 2: Find the road (excluding Route 1 if it qualifies) that passes through the University of Maryland campus and is nearest to the Computer Science Department (for simplicity, specified here by its coordinate values (nx,ny)), and such that the road is within distance r from the point (cx,cy).

```
select road_name
  from roads land-use
 where in_circle(road_coords,cx,cy,r)
       and road_name != "Route 1"
       and pass_through(road_coords,location)
       and name = "University of Maryland"
       and nearest_to(road_coords,nx,ny)
```

The two queries contain a mix of spatial and non-spatial conditions. There is a need for some data structure that captures in it the intermediate result of the spatial conditions as well as the non-spatial conditions.

In [2, 4], we suggested the concept of a *spatial relation* as the basic data structure supported by the spatial data model. Figure 3 shows several forms of spatial relations. A spatial relation is composed of two main components: a spatial and a non-spatial repository. The spatial repository may be composed of one or more spatial data structures while the non-spatial repository is simply a relation. In Section 4, we show how to formalize the concept of a spatial relation in the context of developing a spatial algebra that is based on spatial relations and spatial as well as non-spatial operations.

Figure 3 also illustrates how we link spatial and non-spatial attribute values of an object. In particular, we maintain two types of logical links between the spatial and non-spatial data instances of an object: *forward* and *backward* links. Forward links are used to retrieve the spatial information of an object given the object's non-spatial information. On the other hand, backward links are used to retrieve the non-spatial information of an object given the object's spatial information. Notice that the description of a spatial object in the spatial data structure can point back, via backward links, into more than one tuple in the relation.

We argue informally that the concept of spatial relation indeed captures the intermediate result of select and join operations whether spatial or non-spatial. Refer to Figure 4 for illustration. Figures 4a and 4b show that a spatial relation survives a select operation (i.e., that the result of the selection is also a spatial relation). Figure 4c shows that the result of joining two spatial relations is also a spatial relation. Notice the use of multiple backward links to avoid duplicating the spatial description of an object while still binding the non-spatial information to the spatial information of the same object.

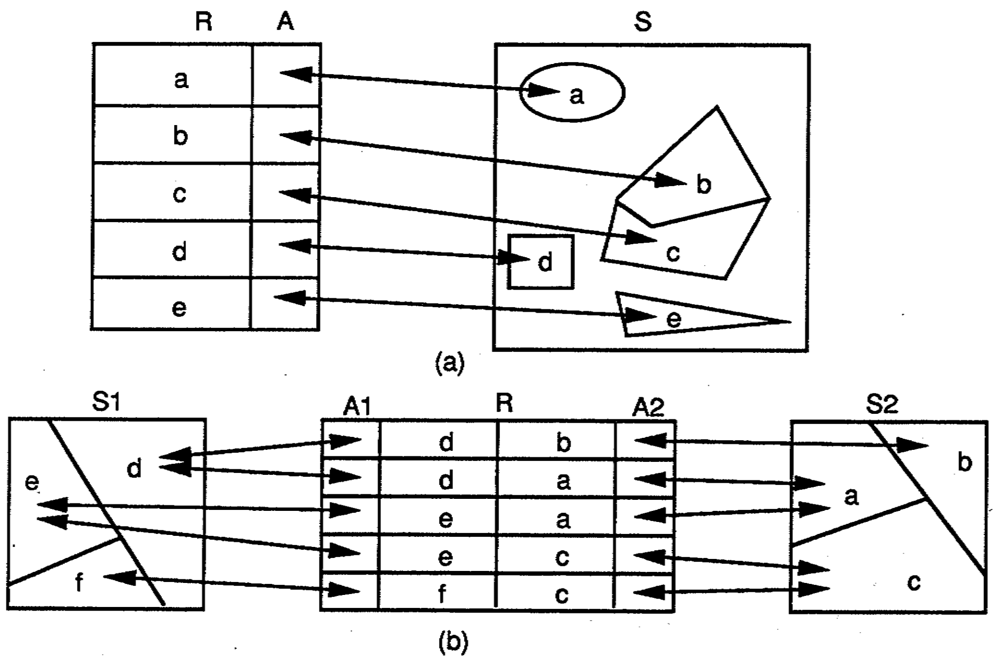

Figure 3: (a) A spatial relation with one spatial attribute A, (b) a spatial relation with two spatial attributes A1 and A2.

Observe also that spatial select and join still resulted in the generation of a spatial data structure. Although this may not always be needed (e.g., see the possible optimizations in [4]), these spatial data structures can be used to perform composite spatial operations that might be difficult to perform if the spatial data is not stored in a suitable spatial data structure. Also, the resulting spatial relation can be stored for future querying on it. This reflects the need for a persistent spatial index. A persistent spatial index exists before and after the performance of a spatial or a non-spatial operation. In fact, although used for different purposes, this technique of building indexes on the fly is used in processing some difficult queries. Also, Sellis [24] discusses constructing partial indexes on the fly in order to save on query processing time.

Backward and forward links can be realized in different ways. However, we do not address implementation issues in this paper. One example of a forward link is a candidate point inside a region to uniquely select a regional object in an object space consisting of non-overlapping regions; while a backward link can be tuple-id.

Maintaining forward and backward links between the spatial and non-spatial aspects of a set of homogeneous objects facilitates browsing in the two parts and also permits efficient query processing. Flexibility in the interaction between spatial and non-spatial attributes enables operations (whether spatial or non-spatial) to be performed in their most natural environment. In [2, 4] we showed how forward and backward links facilitate query processing and query optimization.

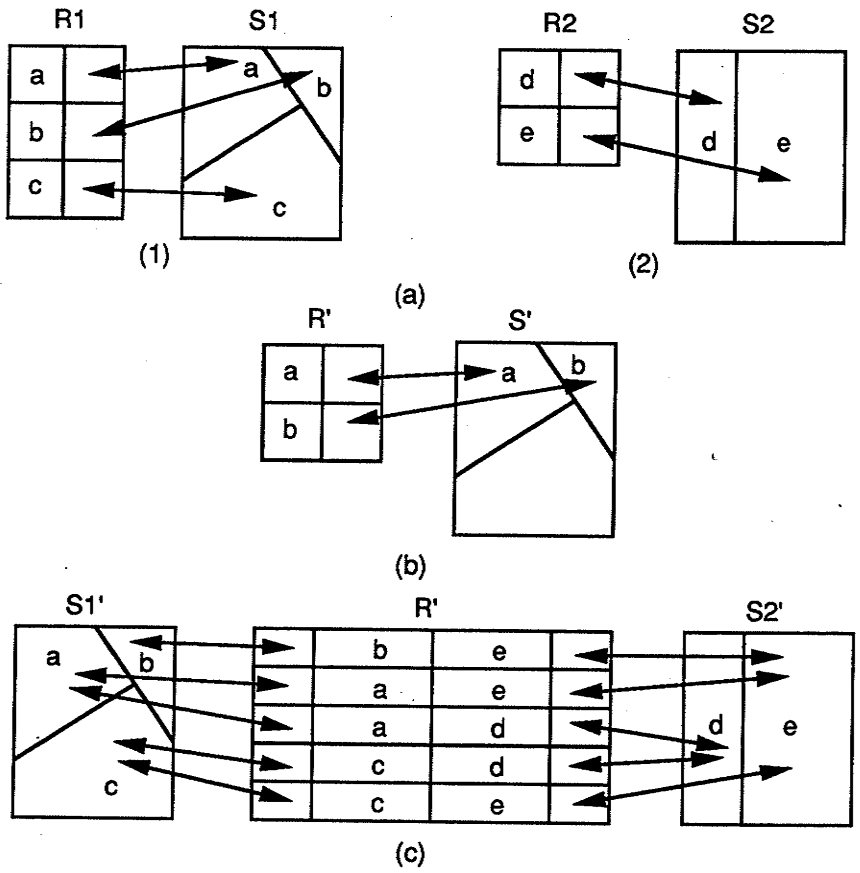

Figure 4: (a) Two spatial relations 1 and 2, (b) the result of performing a selection on 1 (that selects objects a and b), (c) the result of joining 1 and 2 using the spatial intersect join operator.

4 Spatial Algebra

We now develop a formalism for specifying the notions of a spatial attribute and a spatial relation. Using this formalism, we will redefine both spatial and relational operators in the context of spatial relations. These operators along with spatial relations form what we call the *spatial algebra*. In addition, we study equivalences among legal expressions in this algebra. This serves as a basis for spatial query processing and optimization. We investigate the usefulness of this formalism (or possibly a different one) in query processing and optimization. We start by defining a spatial relation more formally.

Definition - Spatial relation: A spatial relation is the 2-tuple $\Re = < R, L >$ where R is a relation with schema r and l spatial attributes ($l \geq 0$), and L is a set of l quadruples. Each quadruple, say q in L, is of the form $< A_{s_i}, S_i, F_i, B_i >$, where A_{s_i} is a spatial attribute in R, S_i is the spatial data structure corresponding to A_{s_i}, F_i is a function representing the forward links of the attribute A_{s_i}, and B_i is a function representing the backward links of the attribute A_{s_i}.

The domain of F_i is the set of tuples in R and the range is the set of spatial indexes of the spatial objects stored in S_i. In other words, given a tuple, say t, in R, $F_i(t)$ returns the spatial index of t's corresponding spatial object stored in S_i. The domain of B_i is the set of spatial objects stored in S_i, while the range is the power set of the set of tuples in R. In

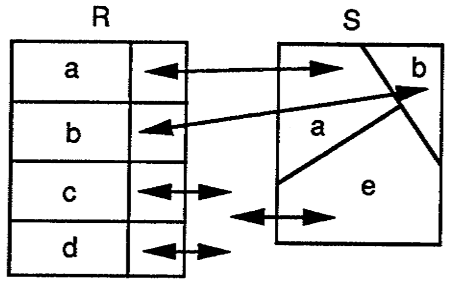

Figure 5: Some tuples do not match with a spatial description and vice versa.

other words, given a spatial object index s in S_i, $B_i(s)$ returns a set, say U, of tuples that point to s, and $U \subseteq R$. Notice that $t \in B_i(F_i(t))$.

Definition - The spatial relation invariant: Given a spatial relation $\Re =< R, \{< A_s, S, F, B >\} >$, then for every spatial object in $\Re$, there is a tuple t and a spatial description s that are linked properly to each other through the functions F and B. More formally,

$$\forall t : (t \in R \wedge (\exists s : (s \in S_A \wedge F_A(t) = s \wedge t \in B_A(s)))) \wedge$$
$$\forall s : (s \in S_A \wedge |B_A(s)| \geq 1 \wedge \forall t : (t \in B_A(s) \wedge t \in R \wedge F_A(t) = s)).$$

Definition - The relaxed spatial relation invariant: Here we allow the situation shown in Figure 5. In other words, we do not insist that each tuple matches with a spatial description in the spatial data structure and that each spatial description has a corresponding tuple.

$$\forall t : (t \in R \wedge F_A(t) = s \wedge s \in S_A \Rightarrow B_A(s) = t) \wedge$$
$$\forall s : (s \in S_A \wedge \forall t \in B_A(s) : t \in R \Rightarrow F_A(t) = s).$$

Now it remains to show that the concept of a spatial relation indeed captures the intermediate results of both spatial and non-spatial qualifications in a given query. In the following, we try to formalize spatial and non-spatial operations based on spatial relations.

4.1 The Select Operator

Select is a unary operator on spatial relations. When applied to a spatial relation $\Re$, it yields another spatial relation that is the subset of $\Re$ with a certain qualification on a specified attribute. If the qualification is based on a spatial attribute, the operation is called a spatial select, while if the qualification is based on a non-spatial attribute, the operation is called a non-spatial or relational select. We now express the definition of a spatial selection and a relational selection more formally.

Let $\Re$ be a spatial relation $< R, \{< A_s, S, F, B >\} >$ where R is a relation with schema r, A_s is a spatial attribute in r whose corresponding spatial data structure is S and mapping functions F and B, and a is a constant (which may or may not be in the domain of A_s). A spatial select on the spatial attribute A_s, $\sigma_{sp(A_s,a)}(\Re)$ ('select spatial objects where the instances of A_s satisfy the spatial condition $sp(A_s, a)$') is the spatial relation $\Re' =< R', \{< A_s, S', F', B' >\} >$ where R' is a relation with the same schema as R (i.e., with schema r). The components of $\Re'$ are defined as follows:

$$\begin{aligned}
S' &= \{s | s \in S \wedge sp(s,a)\}, \\
R' &= \{t | t \in R \wedge sp(F(t),a)\}, \\
F'(t) &= F(t) \ \forall t : (t \in R \wedge sp(F(t),a), \text{ and} \\
B'(s) &= B(s) \ \forall s : (s \in S \wedge sp(s,a)).
\end{aligned}$$

A relational select on the attribute A_r, $\sigma_{db(A_r,a)}(\Re)$ ('select tuples where the instances of A_r satisfy the condition $db(A_r,a)$') is the spatial relation $\Re' =< R', \{< A_s, S', F', B' >\} >$ where R' is a relation with the same schema as R (i.e., with schema r), and the components of $\Re'$ are defined as follows:

$$\begin{aligned}
S' &= \{s | \exists t : (t \in R \wedge db(t,a) \wedge F(t) = s \wedge t \in B(s))\}, \\
R' &= \{t | t \in R \wedge db(t,a)\}, \\
F'(t) &= F(t) \ \forall t : (t \in R \wedge db(t,a)), \text{ and} \\
B'(s) &= B(s) \ \forall s : (s \in S \wedge \exists t : (t \in R \wedge db(t,a) \wedge F(t) = s \wedge t \in B(s))).
\end{aligned}$$

Lemma 1: The result of the select operator (whether spatial or relational) satisfies the spatial relation invariant (i.e., the result of the select operation is also a spatial relation).

Proof: We prove the lemma for the case of a spatial select only. The proof for relational select is analogous. Let $\Re =< R, \{< A_s, S, F, B >\} >$ and $\Re' =< R', \{< A_s, S', F', B' >\} >$ such that $\Re' = \sigma_{sp(A_s,a)}(\Re)$. From the definition of σ_{sp},

$$\begin{aligned}
\forall t \in R' &\Rightarrow t \in R \wedge \exists s : (s \in S \wedge sp(s,a) \wedge t \in B(s) \wedge F(t) = s) \\
\text{(from the definition of S')} &\Rightarrow t \in R \wedge \exists s : (s \in S' \wedge s \in S \wedge sp(s,a) \wedge t \in B(s) \wedge F(t) = s) \\
\text{(from the definition of B')} &\Rightarrow t \in R \wedge \exists s : (s \in S' \wedge s \in S \wedge sp(s,a) \wedge t \in B'(s) \wedge F(t) = s) \\
\text{(from the definition of F')} &\Rightarrow t \in R \wedge \exists s : (s \in S' \wedge s \in S \wedge sp(s,a) \wedge t \in B'(s) \wedge F'(t) = s) \\
&\Rightarrow \exists s : (s \in S' \wedge t \in B'(s) \wedge F'(t) = s)
\end{aligned}$$

Therefore, the resulting spatial relation satisfies the first part of the invariant. Similarly, the resulting spatial relation can be proven to satisfy the second part of the invariant.

$\square$

Lemma 2: The select operators (whether spatial or relational) commute under composition. There are three different cases to consider.

$$\begin{aligned}
\sigma_{db}(\sigma_{sp}(\Re)) &= \sigma_{sp}(\sigma_{db}(\Re)), \\
\sigma_{sp1}(\sigma_{sp2}(\Re)) &= \sigma_{sp2}(\sigma_{sp1}(\Re)), \text{ and} \\
\sigma_{db1}(\sigma_{db2}(\Re)) &= \sigma_{db2}(\sigma_{db1}(\Re)).
\end{aligned}$$

Here we demonstrate only one case, namely that of $\sigma_{db}(\sigma_{sp}(\Re)) = \sigma_{sp}(\sigma_{db}(\Re))$. Consider the left-hand side. Let $\Re_1 = \sigma_{sp}(\Re) =< R_1, \{< A_s, S_1, F_1, B_1 >\} >$. Then,

$$\begin{aligned}
S_1 &= \{s | s \in S \wedge sp(s,a)\}, \\
R_1 &= \{t | t \in R \wedge sp(F(t),a)\}, \\
F_1(t) &= F(t)\ \forall t : (t \in R \wedge sp(F(t),a)), \text{ and} \\
B_1(s) &= B(s)\ \forall s : (s \in S \wedge sp(s,a)).
\end{aligned}$$

Now let $\Re_2 = \sigma_{db}(\Re_1) =< R_2, \{< A_s, S_2, F_2, B_2 >\} >$. Then

$$\begin{aligned}
S_2 &= \{s | \exists t : (t \in R_1 \wedge db(t,a) \wedge F_1(t) = s \wedge t \in B_1(s))\}, \\
R_2 &= \{t | t \in R_1 \wedge db(t,a)\}, \\
F_2(t) &= F_1(t)\ \forall t : (t \in R_1 \wedge db(t,a)), \text{ and} \\
B_2(s) &= B_1(s)\ \forall s : (s \in S_1 \wedge \exists t : (t \in R_1 \wedge db(t,a) \wedge F_1(t) = s \wedge t \in B_1(s))).
\end{aligned}$$

Substituting S for S_1 and B for B_1, we restrict the domain of s by $sp(s,a)$ as follows:

$$\begin{aligned}
S_2 &= \{s | \exists t : (t \in R_1 \wedge db(t,a) \wedge F_1(t) = s \wedge t \in B(s) \wedge sp(s,a))\}, \\
R_2 &= \{t | t \in R_1 \wedge db(t,a)\}, \\
F_2(t) &= F_1(t)\ \forall t : (t \in R_1 \wedge db(t,a)), \text{ and} \\
B_2(s) &= B(s)\ \forall s : (s \in S \wedge sp(s,a) \wedge \exists t : (t \in R_1 \wedge db(t,a) \wedge F_1(t) = s \wedge t \in B(s))).
\end{aligned}$$

Substituting R for R_1 and F for F_1, we get:

$$\begin{aligned}
S_2 &= \{s | \exists t : (t \in R_1 \wedge db(t,a) \wedge F(t) = s \wedge t \in B(s) \wedge sp(s,a))\}, \\
R_2 &= \{t | t \in R \wedge db(t,a) \wedge sp(F(t),a)\}, \\
F_2(t) &= F(t)\ \forall t : (t \in R \wedge sp(F(t),a)), \text{ and} \\
B_2(s) &= B(s)\ \forall s : (s \in S \wedge sp(s,a) \wedge \exists t : (t \in R \wedge db(t,a) \wedge F(t) = s \wedge t \in B(s))).
\end{aligned}$$

This is easily reversible to yield $\sigma_{sp}(\sigma_{db}(\Re))$.

$\square$

Lemma 3: The select operator (whether spatial or relational) is distributive over the binary Boolean operations.

This and other equivalence expressions in the context of spatial relations and spatial operations can be studied similarly. Some are given in Section 4.3.

4.2 The Join Operator

Join is a binary operator for combining two spatial relations. A theta-join is a join that handles comparisons between columns that are θ-comparable where θ is a comparator. If the comparator is based on spatial attributes, the operation is called a spatial join, while if the comparator is based on a non-spatial attribute, the operation is called a non-spatial or relational join.

Let $\Re_1$ and $\Re_2$ be the two spatial relations $< R_1, \{< A_s, S_1, F_1, B_1 >\} >$ and $< R_2, \{< B_s, S_2, F_2, B_2 >\} >$, respectively, where each of $\Re_1$ and $\Re_2$ has only one spatial attribute. The schemas for R_1 and R_2 are r_1 and r_2, respectively. We now express the definition of a spatial and relational join more formally.

A spatial join $\bowtie_{sp}$ on the spatial attributes A_s and B_s, $\Re_1 \bowtie_{sp(A_s,B_s)} \Re_2$, is the spatial relation $\Re' =< R', \{< A_s, S_1', F_1', B_1' >, < B_s, S_2', F_2', B_2' >\} >$ where R' is a relation with schema $r_1 + r_2$. The components of $\Re'$ are defined as follows: Let *tid* (denoting tuple-id) be some notion of backward link information.

$$
\begin{aligned}
S_1' &= \{s_1 | s_1 \in S_1 \wedge \exists s_2 : (s_2 \in S_2 \wedge sp(s_1, s_2))\}, \\
S_2' &= \{s_2 | s_2 \in S_2 \wedge \exists s_1 : (s_1 \in S_1 \wedge sp(s_1, s_2))\}, \\
R' &= \{t_1 t_2 | t_1 \in R_1 \wedge t_2 \in R_2 \wedge sp(F_1(t_1), F_2(t_2))\}, \\
F_1'(t_1) &= F_1(t_1),\ F_2'(t_2) = F_2(t_2)\ \forall t_1, t_2 : (t_1 \in R_1 \wedge t_2 \in R_2 \wedge sp(F_1(t_1), F_2(t_2))),\ \text{and} \\
B_1'(s_1) &= B_2'(s_2) = tid(t')\ \forall s_1, s_2 : (s_1 \in S_1 \wedge s_2 \in S_2 \wedge sp(s_1, s_2) \\
&\quad \wedge t_1 \in B_1(s_1) \wedge t_2 \in B_2(s_2) \wedge t' = t_1 t_2).
\end{aligned}
$$

Notice that since new tuples are constructed as the result of the join, the backward link information (the function B) has to be updated accordingly to point to the new tuples.

A relational join can be expressed analogously. Several properties and equivalences that involve joins hold in the context of this spatial algebra. For example, commutativity of the join operation can be seen from the symmetry in its definition. Further properties of spatial and relational joins when applied to spatial relations (e.g., associativity) need to be proved and are a subject of future research.

4.3 The Overlay Operator

The overlay operator is a special form of a spatial join where the two spatial attributes participating in the spatial join are merged in some way to yield only one output spatial attribute. In this regard, the overlay operation resembles a spatial natural join in contrast to the spatial join presented in Section 2. Another feature of the overlay operator is that the instance values of the resulting spatial attribute are the result of applying a function whose domain is the instance values of the two participating spatial attributes and whose range is the overlay of these two instances expressed in some format. The above two features are illustrated by the following example.

Example: Consider the two spatial relations $\Re_1$ and $\Re_2$, given in Figures 6a and 6b. The result of overlaying $\Re_1$ and $\Re_2$ is given in Figure 6c. Notice that the resulting spatial relation has only one spatial attribute, say a_2, and that some new spatial objects are instantiated (e.g., ae and ce whose spatial indexes did not exist in $\Re_1$ or $\Re_2$). These spatial indexes and their corresponding objects were generated on the fly while performing the overlay operation with some overlay function.

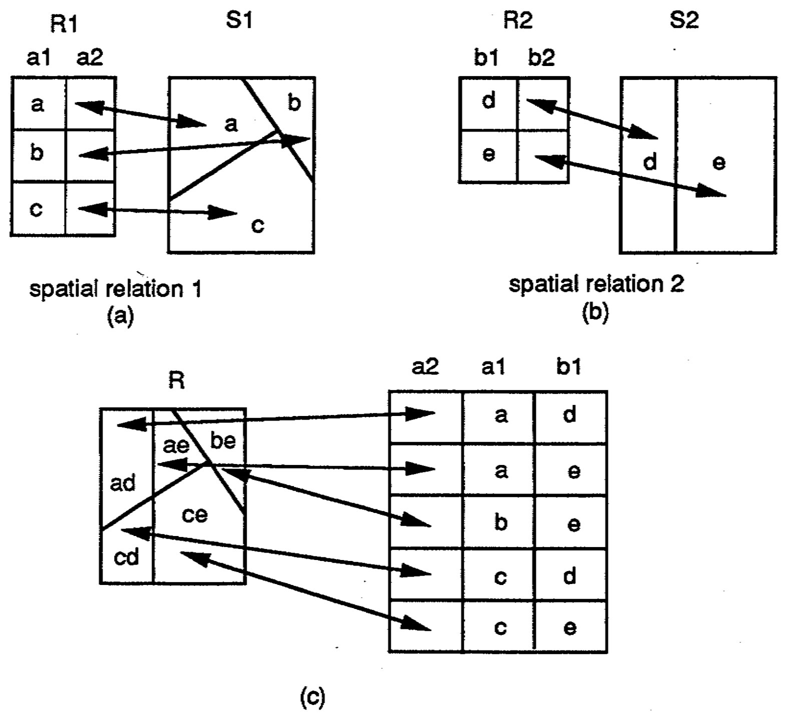

Figure 6: (a) and (b) the two input spatial relations, (c) the result of the overlay operation.

Let $\Re_1$ and $\Re_2$ be the two spatial relations $< R_1, \{< A_s, S_1, F_1, B_1 >\} >$ and $< R_2, \{< B_s, S_2, F_2, B_2 >\} >$, respectively, where each of $\Re_1$ and $\Re_2$ has only one spatial attribute. The schemas for R_1 and R_2 are r_1 and r_2, respectively. We express the definition of the overlay operator more formally. For simplicity, we make the following assumptions: each of the two spatial relations corresponds to a map having only non-overlapping spatial objects (e.g., non-overlapping regions), and there is at least one spatial object stored in each spatial relation that corresponds to the background (or white space) of each map.

An overlay $\bowtie_{sp\_o}$ on the spatial attributes A_s and B_s, $\Re_1 \bowtie_{sp\_o(A_s,B_s)} \Re_2$, is the spatial relation $\Re' =< R', \{< C_s, S', F', B' >\} >$ where R' is a relation with schema $r_1 + r_2 - \{A_s, B_s\} + \{C_s\}$. The components of $\Re'$ are defined as follows: Let *sid* (denoting spatial-id) and *tid* (denoting tuple-id) be some notion of forward and backward link information, respectively. Then

$$\begin{aligned}
S' &= \{s' | \exists s_1 \in S_1 \wedge s_2 \in S_2 : (s' = sp\_o(s_1, s_2))\}, \\
R' &= \{t_1't_2' | \exists t_1 \in R_1 \wedge t_2 \in R_2 : (t_1' = \pi_{r_1 - A_s}(t_1) \wedge t_2' = \pi_{r_2 - B_s}(t_2) \wedge sp\_o(F_1(t_1), F_2(t_2)))\}, \\
F'(t') &= sid(s')\ \forall t' : (t' = t_1't_2' \wedge t_1' = \pi_{r_1 - A_s}(t_1) \wedge t_2' = \pi_{r_2 - B_s}(t_2) \wedge t_1 \in R_1 \wedge t_2 \in R_2 \\
&\quad \wedge s' = sp\_o(F_1(t_1), F_2(t_2)) \wedge s' \neq \phi) \text{ and} \\
B'(s') &= tid(t')\ \forall s' : (\exists t' : (t' = t_1't_2' \wedge t_1' = \pi_{r_1 - A_s}(t_1) \wedge t_2' = \pi_{r_2 - B_s}(t_2) \wedge t_1 \in R_1 \wedge t_2 \in R_2 \\
&\quad \wedge s' = sp\_o(F_1(t_1), F_2(t_2)) \wedge s' \neq \phi)).
\end{aligned}$$

Other algebraic properties that involve selection, projection, and join can be proved in the context of spatial relations in a way similar to that of Section 4.1. A few examples of such equivalences are given below. We use the following notation. Let $\Re_1 =< R_1, \{< A_s, S_1, F_1, B_1 >\} >$ and $\Re_2 =< R_2, \{< B_s, S_2, F_2, B_2 >\} >$ be two spatial relations; r_1 and r_2 be the schemas of R_1 and R_2, respectively; X be a subset of schema r_1; A, A_1 be attributes in r_1; B, B_1 be attributes in r_2; and p and q be spatial or relational conditions. Then

$$\begin{aligned}
\pi_X(\sigma_{p(A)}(\Re_1)) &= \pi_X(\sigma_{p(A)}(\pi_{(X,A)}(\Re_1))) \\
\pi_X(\sigma_{p(A)}(\Re_1)) &= \sigma_{p(A)}(\pi_X(\Re_1)) \text{ if } A \in X \\
\sigma_{p(A_1)}(\Re_1 \bowtie_{q(A,B)} \Re_2) &= \sigma_{p(A_1)}(\Re_1) \bowtie_{q(A,B)} \Re_2
\end{aligned}$$

Other operations that need more investigation are map update and map display. Some operations may update the spatial data structure but may or may not change the value of the spatial index attached to the corresponding tuple in the relation. For example, in a minimum bounding box structure (e.g., an R-tree [9]), due to editing, rectangles may grow or shrink. Deleting the original rectangle and then reinserting it may change the spatial index of the rectangle in the tree.

In addition, there is a variety of map editing operations that need further consideration. Some of these operations are: splitting a region, expanding or shrinking a region, adding/deleting a new spatial object, and relabeling an object. In general, map editing is an interesting spatial update operation that deserves further study.

5 Related Work

The support of spatial data in a database environment is a subject of considerable amount of research (e.g., [1, 8, 10, 15, 18, 26, 30]). This includes ways of representing and accessing large volumes of spatial data (e.g., [7, 10, 20, 29]), data models (e.g., [31]), developing algebras and extensions to existing systems to accommodate spatial data and spatial operations (e.g., [8, 16, 19, 23, 30]), spatial query processing and optimization (e.g., [4, 14]), interface languages (e.g., [6]), etc.

Using links to support complex objects was proposed in [12] as an enhancement to System R [5]. In [13] arbitrary links (via references and identifiers) were used in the context of geographical databases where a geographical object is represented by a tuple having some unique identifier and a long field to store the geographic representation of the object in addition to other non-spatial attributes. Links were used to express explicit relationships between geographical objects (e.g., parent-child, or reference relationships). More recently, in [11] they were used to design a very promising extension of the SQL query language. Bi-directional links were also used in GEO-Kernel [30, 22] in order to allow each of the spatial and non-spatial parts of an object to access the other part. Spatial objects are partitioned into cells. Cells are clustered according to their two-dimensional neighborhood. The address of the non-spatial description of an object is stored with each cell of this object. Since GEO-Kernel uses nested relations [21], an attribute of type relation (i.e., a sub-relation) is used in order to store all the addresses of the cells comprising a spatial object. This resembles a forward link in a spatial relation. However, in contrast to sub-relations, in a spatial relation only one spatial identifier is stored as a forward link and it is the responsibility of the spatial data structure (or its encapsulating spatial process or ADT) to extract the rest of the spatial description of an object. This greatly reduces the storage overhead from the relational side as well as reduces the cost of maintaining the set of addresses stored in a sub-relation in GEO-Kernel that refer to different cells of the same object.

Abstract data types (ADTs) as new attribute domains in a relation and user-defined spatial index structures were first introduced in [26, 28]. Following this approach, spatial data structures are viewed in many extensible database systems (e.g., [8, 27, 18, 17]) as indexing structures. As a result, only backward links (i.e., from the index structure to the actual tuple) are needed in these architectures. In contrast, it is important to note that in a spatial relation, user-defined data structures serve not only as spatial indexes for speeding up operations but also as *containers* for the full description of spatial data. In the latter, bi-directional links between corresponding components of the ADT can be used for efficient access. Spatial relations can be viewed as a merge of the ADT work in [26] and the complex object support in [12]. In a spatial relation, a spatial attribute is an ADT that is implemented by user-defined data structures, and some operations that function over an instance of an ADT or over a collection of instances (e.g., all the instances of the ADT in a given spatial relation).

Another alternative (e.g., as in [8]) avoids forward links (and hence does not use containers) by employing spatial data structures as indexes that approximate spatial objects (e.g., by using bounding boxes). In addition, the full description of the spatial object (e.g., the coordinate values of the vertices of a polygon) is stored with the object's corresponding tuple in an on-line format (i.e., formatted in a memory-mapped image such that when the tuple is loaded into main memory, the spatial description of the object is ready for use without any additional format conversions). In this case, when a query is posed, the spatial data structure is used as a filtering step that produces a set of candidate objects, then the full description of these objects is used as a refinement step to produce the final answer to the query. The performance of these alternative data architectures with respect to spatial query processing and optimization is an interesting research problem. One drawback of this approach, though, is that every time a tuple participates in a join, the full description of the spatial object needs to be duplicated as well (in many cases, the size of this spatial description of an object is large; for example, this is the case with polygon objects).

Another alternative to having permanent disk-based spatial data structures for storing spatial data is the following: store each instance of a spatial attribute as a textual string, and download these instances (via conversion routines) into an appropriate on-line spatial data structure in order to query and perform spatial operations on them. Then, upon completion, upload them (if necessary) back into their string format (via another conversion routine). Since spatial data is voluminous, downloading and uploading it is expensive. In addition, the entirety of the data is not likely to fit into main memory. Therefore, disk-based spatial data structures appear to be a necessity. Therefore, we short-cut the downloading/uploading procedures, and assume that spatial data is stored permanently in disk-based structures to begin with, thereby getting rid of the textual off-line representation of spatial data.

Generally speaking, the concept of a spatial relation and the spatial algebra imposed on it, as presented in this paper, can be viewed as an extension of the Probe spatial data model [16].

[3] W. G. Aref and H. Samet. Extending a DBMS with spatial operations. In O. Günther and H. J. Schek, editors, *Advances in Spatial Databases—2nd Symposium, SSD'91. Also Lecture Notes in Computer Science 525*, pages 299–318. Springer-Verlag, Berlin, 1991.

[4] W. G. Aref and H. Samet. Optimization strategies for spatial query processing. In *Proceedings of the 17th International Conference on Very Large Databases (VLDB)*, pages 81–90, Barcelona, Spain, September 1991.

[5] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System R: relational approach to database management. *ACM Transactions on Database Systems*, 1(2):97–137, June 1976.

[6] M. J. Egenhofer and A. U. Frank. Towards a spatial query language: User interface considerations. In *Proceedings of the 14th International Conference on Very Large Databases (VLDB)*, pages 124–133, Los Angeles, CA, September 1988.

[7] O. Gunther and H. Noltemeier. Spatial database indices for large extended objects. In *Proceedings of the 5th International Conference on Data Engineering*, pages 520–526, Kobe, Japan, April 1991.

[8] R. H. Güting. Gral: An extensible relational system for geometric applications. In *Proceedings of the 15th International Conference on Very Large Databases (VLDB)*, pages 33–44, Amsterdam, The Netherlands, August 1989.

[9] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, MA, June 1984.

[10] C. P. Kolovson and M. Stonebraker. Segment Indexes: Dynamic indexing techniques for multi-dimensional interval data. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, pages 138–147, Denver, CO, May 1991.

[11] R. Lorie. The use of a complex object language in geographic data management. In O. Günther and H. J. Schek, editors, *Advances in Spatial Databases—2nd Symposium, SSD'91. Also Lecture Notes in Computer Science 525*, pages 319–337. Springer-Verlag, Berlin, 1991.

[12] R. Lorie, W. Kim, D. McNabb, W. Plouffe, and A. Meier. Supporting complex objects in a relational system for engineering databases. In W. Kim, D. Reiner, and D. Batory, editors, *Query Processing in Database Systems*, pages 145–155. Springer-Verlag, New York, 1984.

[13] R. Lorie and A. Meier. Using a relational DBMS for geographical databases. *Geo-Processing*, 2:243–257, 1984.

[14] B. C. Ooi and R. Sacks-Davis. Query optimization in an extended DBMS. In W. Litwin and H.-J. Schek, editors, *Foundations of Data Organization and algorithms*, pages 48–63,

In Probe, relations are extended by spatial attributes where spatial data is represented using Z-order encoding which is then stored in a B-tree. In this respect, a spatial relation extends Probe in that it permits Z-order or any other appropriate representation for spatial data in addition to any suitable spatial data structure. Spatial data is stored in the most appropriate spatial data structures to permit efficient spatial operations.

6 Conclusions and Future Work

The concept of a spatial relation as a means of unifying spatial and non-spatial aspects of data was introduced. The spatial relation provides the flexibility in utilizing spatial data structures not only as *indexing mechanisms* to speed up query processing but also as *containers* for organizing and processing the full descriptions of spatial objects. A spatial relation also supports efficient processing of spatial and relational operators since it permits each operation to be performed in its most natural environment. Classifying operations into selects and joins and designing the underlying basic data structure (the spatial relation) so that the main algebraic characteristics still hold paves the way for using the same logic employed by a standard query optimizer but with a different basic structure. A system called SAND (denoting Spatial And Non-spatial Data) [3] that reflects the concept of a spatial relation has been prototyped and implemented using POSTGRES [27] and QUILT [25]. QUILT serves as a spatial processor to support spatial data handling. It uses a number of variants of hierarchical data structures.

Other questions that need to be addressed include:

- How should we process queries expressed in this augmented environment where both spatial and non-spatial data are linked together?
- How should we optimize queries?
- What are the selectivity factors for spatial operations?
- What are the heuristics used for optimization?
- How can we estimate the cost of spatial queries in this environment?

Future work includes addressing the above questions in the context of spatial relations and their associated algebra, and developing a cost model for spatial operations that we can use in building a spatial query optimizer.

References

[1] D. J. Abel. Relational data management facilities for spatial information systems. In *Proceedings of the 3rd International Symposium on Spatial Data Handling*, pages 9–18, Sydney, Australia, August 1988.

[2] W. G. Aref and H. Samet. An approach to information management in geographical applications. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 589–598, Zurich, Switzerland, July 1990.

Berlin, Germany, 1989. Also *Lecture Notes in Computer Science 367*, Springer-Verlag, Berlin.

[15] J. A. Orenstein. An object-oriented approach to spatial data processing. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, volume 2, pages 669–678, Zurich, Switzerland, July 1990.

[16] J. A. Orenstein and F. A. Manola. PROBE spatial data modeling and query processing in an image database application. *IEEE Transactions on Software Engineering*, 14(5):611–629, May 1988.

[17] N. Roussopoulos. Oral communication, April 1992.

[18] N. Roussopoulos, C. Faloutsos, and T. Sellis. An efficient pictorial database system for PSQL. *IEEE Transactions on Software Engineering*, 14(5):639–650, May 1988.

[19] R. Sacks-Davis, K. J. McDonell, and B. C. Ooi. GEOQL—A query language for geographic information systems. Technical Report 87/2, Monash University, Victoria, Australia, July 1987.

[20] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[21] H. Schek and M. Scholl. The two roles of nested relations in the DASDBS project. In S. Abiteboul, P. C. Fischer, and H.-J. Schek, editors, *Nested Relations and Complex Objects in Databases. Also Lecture Notes in Computer Science 361*, pages 50–68. Springer-Verlag, Berlin, 1989.

[22] H. Schek and W. Waterfeld. A database kernel system for geoscientific applications. In *Proceedings of the 2nd International Symposium on Spatial Data Handling*, pages 273–288, Seattle, WA, July 1986.

[23] M. Scholl and A. Voisard. Thematic map modeling. In *Design and Implementation of Large Spatial Databases, Proceedings of the First Symposium SSD'89*, pages 167–190, Santa Barbara, CA, July 1989.

[24] T. Sellis. Efficiently supporting procedures in relational database systems. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 278–291, San Francisco, CA, May 1987.

[25] C. A. Shaffer, H. Samet, and R. C. Nelson. QUILT: A geographic information system based on quadtrees. *International Journal of Geographical Information Systems*, 4(2):103–131, April–June 1990.

[26] M. Stonebraker. Inclusion of new types in relational data base systems. In *Proceedings of the 2nd International Conference on Data Engineering*, pages 262–269, Los Angeles, CA, February 1986.

[27] M. Stonebraker and L. Rowe. The design of POSTGRES. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, pages 340–355, Washington, DC, May 1986.

[28] M. Stonebraker, B. Rubenstein, and A. Guttmann. Application of abstract data types and abstract indices to CAD databases. In *Proceedings of the ACM/IEEE Conference on Engineering Design Applications*, pages 107–113, San Jose, CA, 1983.

[29] P. van Oosterom. *Reactive Data Structures for Geographic Information Systems.* PhD thesis, University of Leiden, Leiden, The Netherlands, December 1990.

[30] A. Wolf. The DASDBS GEO-Kernel: Concepts, experiences, and the second step. In A. Buchmann, O. Günther, T. R. Smith, and Y.-F. Wang, editors, *Design and Implementation of Large Spatial Databases, Proceedings of the First Symposium SSD'89. Also Lecture Notes in Computer Science 409*, pages 67–88. Springer-Verlag, Berlin, 1990.

[31] M. F. Worboys, H. M. Hearnshaw, and D. J. Maguire. Object-oriented data modelling for spatial databases. *International Journal of Geographical Information Systems*, 4(4):369–383, October–December 1990.