Integrating Spatial and Temporal Relationship Operators into SQL3 for Historical Data Management

Jong-Yun Lee

A spatial object changes its states over time. However, existing spatial and temporal database systems cannot fully manage time-varying data with both spatial and non-spatial attributes. To overcome this limitation, we present a framework for spatio-temporal databases that can manage all time-varying historical information and integrate spatial and temporal relationship operators into the select statement in SQL3. For the purpose of our framework, we define three referencing macros and a history aggregate operator and classify the existing spatial and temporal relationship operators into three groups: exclusively spatial relationship operators, exclusively temporal relationship operators, and spatio-temporal common relationship operators. Finally, we believe the integration of spatial and temporal relationship operators into SQL3 will provide a useful framework for the history management of time-varying spatial objects in a uniform manner.

Manuscript received May 29, 2001; revised Feb. 28, 2002.

This work was supported by the Program 2001 of Samchok National University.

Jong-Yun Lee (phone: +82 33 570 6405, email: jongyun@ samchok.ac.kr) is with the Department of Information and Communication Engineering at Samchok National University, Korea.

I. INTRODUCTION

A geographic information system (GIS) is a computer-based information system that enables the capturing, modeling, manipulation, retrieval, analysis, and presentation of geographically referenced data [1]. Conventional GISs are able to manipulate only a snapshot image of time-varying spatial objects. However, a spatial object changes its state over time by either spatial event operations or non-spatial event operations. Major temporal studies on handling the history management of spatial objects in GISs can be divided into two categories: the temporal geographic information system (TGIS)-based approach and the spatio-temporal database-based approach. First, in the last several years, many investigations on temporal GISs, such as Beller's TGIS prototype [2], Montgomery [3] and Renolen's TGIS [4], and other GIS space-time models [5]-[7] have aimed at developing TGIS prototypes and requirements specifications. Second, in addition to reports on existing spatial databases [8]-[14] and temporal databases [15]-[19], many recent papers [18], [19]-[28] have reported spatio-temporal models and their operations, which explicitly record spatial changes over time as they relate to specific geographic entities. For example, Peuquet and Duan [18] proposed an event-based approach to represent a spatio-temporal data model, and Worboys [20] described a unified model of spatial and temporal data as well as a spatio-temporal relational algebra. Lorentzos [19] also presented an extension to SQL, IXSQL, for the management of interval data. IXSQL includes an interval-extended relational algebra, extensions of data definition language, and data manipulation language in SQL.

Peuquet and Duan [18] and Claramunt [8] suggested spatio-temporal data models based on an event time, while Worboy's spatio-temporal data model [20] provided bi-temporal elements of valid time and transaction time. Recently, practical object-oriented spatio-temporal data models were proposed in SAIF [25] and CHOROCHRONOS [26], [27].

Notice that previous research on conventional spatial and temporal databases has focused mainly on managing spatial data and time-varying historical data in databases. However, those models cannot handle time-varying spatial objects in a GIS without any modification, nor do they integrate the existing spatial and temporal relationship operators into the select statement of SQL3. Worboys [29] addressed only the needs of handling the spatial and temporal references in a uniform fashion, and Claramunt [6] suggested the need to merge space and time and to relate them to dimensional thematic data.

To overcome these limitations, we focus on solving two major problems in the development of spatio-temporal databases: unifying spatial and temporal data models into a single spatio-temporal data model and then integrating the existing spatial and temporal relationship operators into the select statement of SQL3. Our main contribution is an integration of the conventional spatial and temporal databases into a single structure. To do this, we propose a relational spatio-temporal data model, called a relational spatio-temporal data model (RSTDM), which can handle all time-varying location data and provide a framework for integrating spatial and temporal relationship operators into the select statement of SQL3. Our research on the integration of the spatial and temporal relationship operators into SQL3 for historical data management includes describing a history aggregate operator as retrieving all the spatial and non-spatial histories of objects and three referencing macros—*Spatial*, *Valid*, and *Transaction*—as extracting different dimensional information. We also classify the existing spatial and temporal relationship operators into three groups: *exclusively spatial relationship operators*, *exclusively temporal relationship operators*, and *spatio-temporal common relationship operators*. While Worboys [20] proposed only a logical unified model of spatio-temporal relational algebra, our spatio-temporal data model will provide a unification of the spatial and temporal relationship operators into the select statement of SQL3.

The remainder of this paper is organized as follows. In section II, we describe major spatial event operations that cause the spatial objects to change their states. Section III presents a framework for the proposed spatio-temporal database scheme and reviews the history management by major spatial event operations. Section IV outlines procedures to integrate the existing spatial and temporal relationship operators into SQL3, and section V designs the history aggregate operator. Finally, sections VI and VII present a summary of our research.

II. MAJOR SPATIAL EVENT OPERATIONS

Events are things that occur [6]; events are also happenings of interest to the environment and/or to database systems. For database systems, events are modeled as a set of processes that transform the states of entities. In this paper, spatial events are defined as the transactions, operations, and actions that cause the spatial or non-spatial value of spatial objects to change in a database. For information about spatial events, there are two kinds of event operations—non-spatial operations and spatial event operations—that can change the states of spatial objects.

For example, a parcel may be purchased and owned by people whose address and telephone number, called attribute data, may change over time. These events, which make up the historical information of spatial objects, are called non-spatial event operations. They are treated as important activities in spatio-temporal databases because they create many historical objects and store them permanently in the database. Existing spatial databases cannot manage the historical objects, so spatio-temporal databases must be used to store them in the database to maintain all the historical information for the spatial objects. Figures 1(a) through (c) are examples of non-spatial event operations that change only attribute data, such as a parcel's ownership and address, in which its historical objects are created and then stored in the database as historical objects.

Spatial operations are both the simple and complex events that change the size, area, or location of an object, called spatial attributes. Here, the spatial operations are categorized into simple and complex spatial event operations. Simple events refer to simple spatial event operations, such as creating and deleting a new object, and complex events refer to complex spatial event operations, such as merging, splitting, and re-allocating, the so called major spatial event operations. Thus, all spatial event operations either change the size or area of objects or move them. For example, a spatial object can be moved and its shape and size can also vary according to the geological scale. In spatio-temporal databases, operations generate one historical object or more.

Let us consider the major spatial event operations—create, delete, move, split, merge, and re-allocate—as follows:

Create Operation: For example, a new highway may be constructed in a space. As Fig. 1(d) shows, a newly created object can be stored in the database and it will be described as an insertion operation. No historical objects will be generated.

Delete Operation: After being constructed as a need, a building or a road may be destroyed. Figure 1(e) shows that a spatial object is deleted from the current databases and then

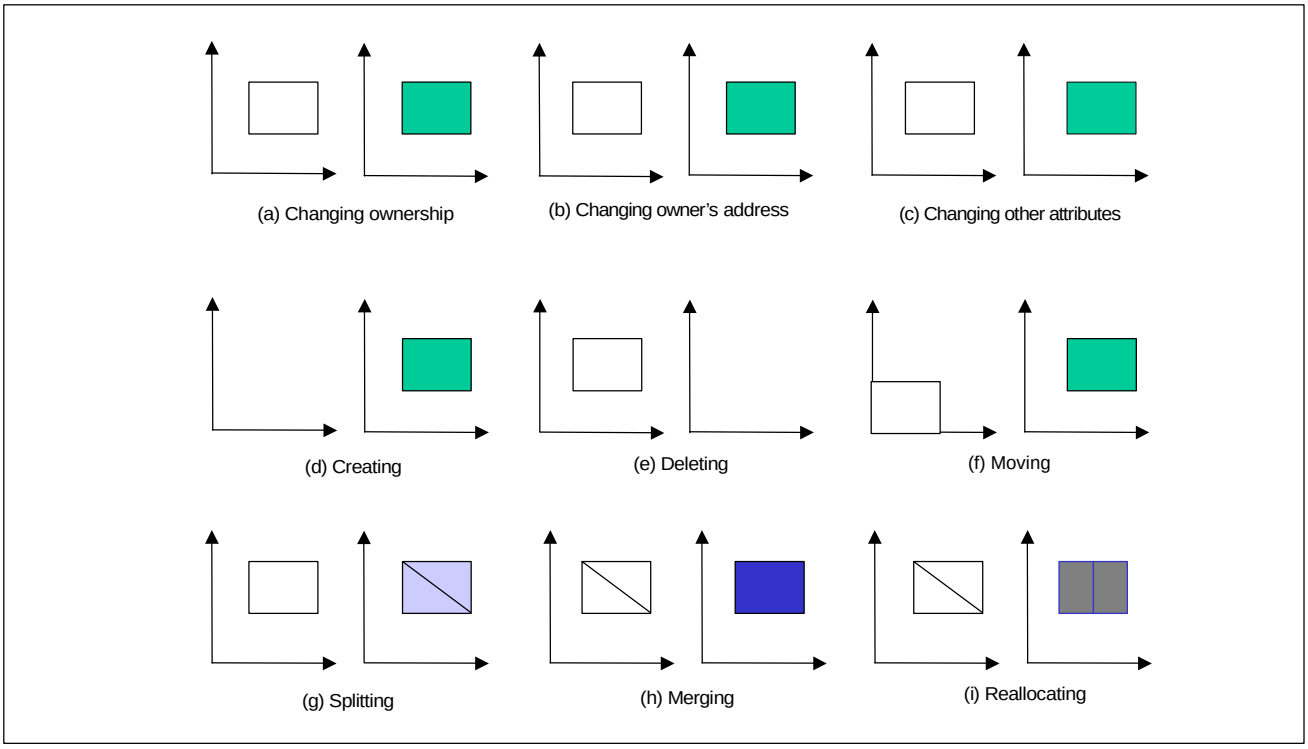

Fig. 1. Major spatial event operations.

creates a historical object. Note that conventional databases do not manage the historical information of the objects; however, such history management is necessary in a full-functioning spatio-temporal database.

Move Operation: Figure 1(f) shows that the location of a spatial object is moved without any changes in either its non-spatial or spatial information. The move operation is processed by updating a current object and creating its historical object because the operation changes its location information. For moving objects, all the paths of a movement at an event time are stored in spatio-temporal databases as historical information.

Figures 1(d) through (f) illustrate simple spatial event operations that change only the spatial information of objects, such as creating, deleting, or moving. However, three major spatial event operations—split, merge, and re-allocate—are complicated spatial operations which change spatial information, such as size, area, and feature points. The complicated spatial event operations will be explained by examples of spatio-temporal operations in section III.2.

Split Operation: A split operation indicates that an object, such as a parcel, is divided into two or more pieces (Fig. 1(g)). Thus, the object changes its size, area, and geometry. The split operation is explained by creating current objects and creating a historical object (Fig. 1(g)).

Merge Operation: A merge operation (Fig. 1(h)) is where some parcels are combined into an object, creating a new spatial object. This creates a merged spatial object and several historical objects as parent objects. In this case, there is a problem of managing the historical objects, because the spatial object has two or more historical objects.

Re-allocate Operation: It is necessary to rearrange spatial objects to construct a building or to do a land readjustment in a city. In these cases, many spatial historical objects are created whenever the size, area, or location of objects changes. Spatio-temporal databases have to be able to store and retrieve the historical information. The reallocate operation shown in Fig. 1(i) rearranges the size, location, or area of spatial objects as needed. It also creates the same number of current spatial objects and their historical objects.

III. A RELATIONAL SPATIO-TEMPORAL DATA MODEL

In this section, we describe our design of a relational spatio-temporal data model (RSTDM), which can manage all time-varying spatial and non-spatial information. The data sets that capture all time-varying location data are grouped into logical units, depending upon different working granularity. At the highest level of work granularity is a data set that logically

groups discrete geographical, tabular, and other attribute information on spatial objects. The next level of granularity is a layer. Each data set can contain multiple layers that are separate base maps, but are not partitioned geographically. The layer therefore contains a lot of similar thematic data. The lowest level of granularity has features called spatial objects.

Each layer in our spatio-temporal data model consists of five relations: Feature Relation, Attribute Relation, Feature History Relation, Attribute History Relation, and Merge Relation. All the historical information of spatial objects in one layer is stored under Feature History Relation and Attribute History Relation. The Merge Relation contains all the historical information of merged objects. The attribute histories for an object are managed under its own history relations. It is beneficial to separate current features from their historical objects because all the historical information of features can be managed independently. A framework of the spatio-temporal data model is described completely in Definition 1 and Definition 2. Table 1 also shows the notation for modeling our spatio-temporal data model.

Table 1. Notation for modeling the spatio-temporal databases.

Notation	Descriptions
U_i	Universal set
D_i	A data set *i*
L_i	A layer *i*
St_i	A spatio-temporal object *i*
fid, hid	A feature identifier
F_i	Feature *i*
FG	Geometry for a feature
FA	Spatial vector for a feature, <A, VT, TT, prev>
A	Attribute vector for a feature, $<a_1, a_2, ..., a_m>$
M	Merge relation
A	An attribute
VT	A valid time period <VTs, VTe>
TT	A transaction time period < TTs, TTe>
Prev	Previous historical pointer for a feature
T	Timestamp
'	a current relation
"	a history relation

Definition 1 (Spatio-Temporal Database Hierarchy) Suppose that U, D, L, and sto are a universal set, a data set, a layer, a spatio-temporal object, respectively. A universal set U is divided into multiple data sets $\{D_1, D_2, D_3, ..., D_m\}$ associated with the object's themes. A data set D is composed of many layers, $\{L_1, L_2, L_3, ..., L_n\}$, and each layer L_i consists of a Feature Relation and an Attribute Relation for current objects, and a Feature History Relation, an Attribute History Relation, and a Merge Relation for their historical objects. The relationship between the Feature Relation and Attribute Relation is associated with the equi-join operation of feature identifiers (fid). In addition, the relationship between the Feature and Attribute Relations and the Feature and Attribute History Relations is connected with the historical information of the Attribute Relation and Attribute History Relation. Each relation involves many spatio-temporal objects, $\{sto_1, sto_2, sto_3, ..., sto_n\}$.

Next, a layer is composed of a Feature Relation and an Attribute Relation which separately express the current spatial and non-spatial data for spatial objects. As described in Definition 2, it also includes a feature history relation and an attribute history relation for historical objects.

Definition 2 (Spatio-Temporal Database Scheme) The Feature Relation F_i' and Feature History Relation F_i" for the *i*-th layer describe a vector of spatial data, <fid, $f_1, f_2, ..., f_n$, FG_i>, where fid represents identifier, $f_1, f_2, ..., f_n$ spatial data and FG_i geometry. The Attribute History Relation FA_i" for the *i*-th layer presents an attribute vector of non-spatial data, <A_i, VT, TT, prev>, where a valid time vector VT = <VTs, VTe> and a transaction time vector TT = <TTs, TTe> denote the beginning and ending times of the valid time and transaction time, respectively. Prev is also a historical pointer of a feature in a historical relation, and A_i is an attribute vector for the spatial object of the *i*-th layer, <fid, $a_1, a_2, ..., a_m$>. On the other hand, the Attribute Relation FA_i' for the *i*-th layer is described only by an attribute vector of non-spatial data, <A_i, VTs, TT, prev>. The Merge Relation M_i describes a historical pointer vector of merged objects, <fid, hid, VT>, where hid denotes the historical pointer of a spatial object fid in the Feature History Relation. A Merge Relation stores only historical information of spatial objects on which the merge operation occurs.

Note that the structures of Attribute Relation FA_i' and Attribute History Relation FA_i" in Definition 2 are different. VT is the time period when a spatial object is true in reality, and transaction time TT is a pair of timestamps in which a spatial object was present in the database. The time domains of the valid time are $\{t_1, t_2, t_3, ..., t_k, now\}$ and those of the transaction time are $\{t_1, t_2, t_3, ..., t_k\} \cup \{UC\}$, where UC is "Until Changed" [17].

To summarize, the proposed model represents the current state of a spatial object by a spatial vector <fid, $f_1, f_2, ..., f_n$, FG_i> and an attribute vector <A_i, VTs, TT, prev> in the Feature Relation F_i' and Attribute Relation FA_i'. Any historical object is stored by a spatial vector <fid, $f_1, f_2, ..., f_n$, FG_i>, an attribute vector <A_i, VT, TT, prev>, and an optional merge vector <fid,

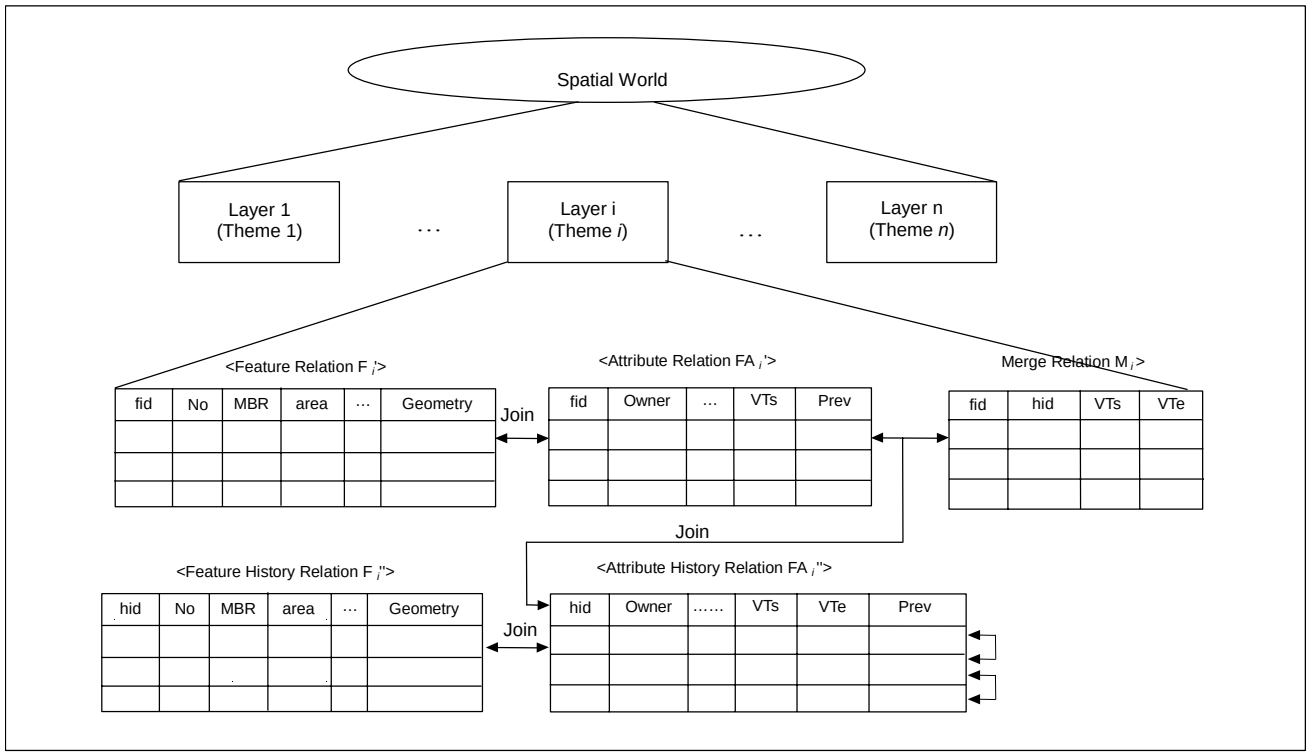

Fig. 2. Structure of relational spatio-temporal data model.

hid, VT> in the Feature History Relation F_i", Attribute History Relation FA_i", and Merge Relation M_i, respectively. Here, examples of a feature's spatial attributes vector <f_1, f_2, ... , f_n> are a feature identifier (fid), MBR (Minimum Rectangle Boundary), area, length, perimeter, etc., and sample non-spatial attributes of the spatial object <fid, a_1, a_2, ... , a_m> are owner, address, and telephone number (Fig. 2). In other words, a spatio-temporal object is an integrated object whose periods of valid and transaction timestamps are attached to the object. Thus, the Attribute History Relation in the proposed spatio-temporal databases is extended by bi-temporal elements, called valid times and transaction times, and a historical pointer (prev) to represent time-varying spatial historical information.

The structure of the proposed spatio-temporal database is shown in Fig. 2. The spatial world is modeled by lots of layers (or themes). Each layer is described by two relations—Feature Relation and Attribute Relation—for current spatial objects and by three relations—Feature History Relation, Attribute History Relation, and Merge Relation—for their historical objects. A merged object's histories are retrieved from the historical identifier with the valid time period in the Merge Relation. Therefore, all the histories for a current object can be retrieved from its historical pointer of the Attribute Relation directly or from its merged identifiers of the Merge Relation.

In the above expression, the merge relation {<fid, hid, VTs, VTe>} indicates an optional table which has historical information for the merge objects. The following section reviews the relational spatio-temporal databases through major spatial event operations that generate spatial historical information. The major spatial event operations will be performed by combinations of database operations, such as insert and update, without any deletion.

IV. INTEGRATION OF SPATIAL AND TEMPORAL RELATIONSHIP OPERATORS

A spatio-temporal database manages four- or five-dimensional information of spatial objects in addition to two- or three-dimensional spatial data and two-dimensional historical information with a valid time and a transaction time. A spatio-temporal database requires extensions of data types and query language in SQL3. Three previous studies on spatial and temporal queries only addressed the need to merge space and time and to allow the handling of the spatial and temporal references in a uniform fashion to produce a conceptually flexible language [6], [20], [29].

However, no studies have tried to resolve the problem of integrating spatial and temporal relationship operators into SQL3. We therefore focus on merging the topological comparison operators, called spatio-temporal relationship operators, for spatial and temporal queries to handle *meets*, *overlaps*, *contains*, *equals*, and so on. In order to integrate these

spatial and temporal relationship operators in SQL3, we first define three referencing macros: *Spatial*, *Valid*, and *Transaction*. We then classify the existing spatial and temporal relationship operators into three groups: exclusively spatial relationship operators, exclusively temporal relationship operators, and spatio-temporal common relationship operators. In the following sections, we describe them in detail.

1. Spatio-Temporal Referencing Operators

The aim of the proposed spatio-temporal referencing operators is to integrate the spatial and temporal relationship operators into SQL3 in a uniform fashion. In spatio-temporal databases, it is necessary to define three operators that reference the different dimensional information of a spatial object, because the operands of spatio-temporal relationship operators can be either a spatial or a temporal object, depending on the user's query statement. In this section, we therefore define three referencing operators of spatio-temporal data—*spatial*, *valid*, and *transaction*—in Definitions 1 through 3, where *i*, *m*, and *n* are natural numbers.

First, the *spatial* operator is used for indicating geometric information of a tuple from either current spatial objects or historical objects. It is defined as follows:

Definition 1 Object.*spatial* = {(x_1,y_1, …, x_n,y_n) | a feature's geometric information FG_i from spatial attributes, F_i of a current object <fid', F_1', F_2', ... , F_n', FG'_i> and a history object <fid", F_1", F_2", ... , F_n", $FG"_i$>}, where the feature's geometric information indicates the real-world coordinates of a spatial object.

For example, there is an explicit spatial reference, such as schools.*spatial*, where the school in the spatio-temporal database may be a layer relation. It is similar to the valid timestamp reference operator in temporal databases [21], [22], [29].

Second, the *valid* operator references the valid time of a spatial object, which indicates a pair of valid timestamps for a tuple from which an object is true in reality. It is defined as follows:

Definition 2 Object.*valid* = {[VTs, VTe] | valid timestamps <VTs, VTe>, which indicates the beginning to ending time period of a valid time VT from attribute data of either a current object <A', VT', TT', prev'> or a history object <A", VT", TT", prev">.

Next, the *transaction* operator is used for referencing a pair of transaction timestamps of an object.

Definition 3 Object.*transaction* = {[TTs, TTe] | transaction timestamps <TTs, TTe>, which indicates the beginning and ending time period of transaction time TT from attribute data of either a current object <A', VT', TT', prev'> or a history object <A", VT", TT", prev">.

Table 2 summarizes the spatial and temporal referencing operators. There are only three possible combinations of spatio-temporal reference operators for the spatio-temporal query expressions—*spatial*/*spatial*, *valid*/*valid*, *transaction*/*transaction*—since the referencing operators between valid time and transaction time cannot be exchanged. However, it is not necessary to combine with elements such as *valid*/*transaction* and *transaction*/*valid* because the *valid* dimension is different from the transaction dimension.

Table 2. Spatio-temporal referencing operators.

Operators \ Use	Examples of use
Spatial	- a.*spatial* disjoints b.*spatial* - b.*spatial* contains a.*spatial* - a.*spatial* overlaps b.*spatial*
Valid	- a.*valid* precedes Timestamp 'Nov-31-1998' - a.*valid* meets b.*valid*
Transaction	- a.*transaction* contains PERIOD 'Jan-01-1998, Nov-31-1998'

2. Exclusively Spatial Relationship Operators

Existing spatial databases contain many relationship operators for comparing the spatial topological relationships among spatial objects. These spatial relationship operators include the following: *common point*, *line cross*, *common line*, *area intersect*, *disjoints*, *point in polygon*, *contains* or *is_contained*, *centroid inside polygon*, *inside envelope*, *overlaps*, *meets*, and *equals*.

This paper classifies the general spatial relationship operators into two groups: spatio-temporal common relationship operators and exclusively spatial relationship operators. The exclusively spatial relationship operators include *common point*, *line cross*, *area intersect*, *point in polygon*, *centroid inside polygon*, and *inside envelope for only spatial use*.

3. Exclusively Temporal Relationship Operators

Allen [30] proposed a complete set of period relations, which included *before*, *after*, *starts*, *started_by*, *finishes*, *finished_by*, *during*, *contains*, *meets*, *met-by*, *equals*, *overlaps*, and *overlapped_by*. This paper separates temporal relationship operators, defining exclusive temporal relationship operators as only specific time-comparison operators; these include *before*,

after, *starts*, *finishes*, *finished_by*, *during*, *met_by*, and *overlapped_by*. The other category of temporal relationship operators is the spatio-temporal common relationship operators.

The exclusive temporal relationship operators are implemented by converting their semantics into relational expressions. For example, the *equals* operator should be translated into a relational expression of valid time "VTs > begin and VTe = end" in spatio-temporal databases.

4. Spatio-Temporal Common Relationship Operators

It is necessary to define some macros for referencing the dimensional information of an object in spatio-temporal databases. We therefore defined spatial and timestamp referencing macros (Definitions 1–3). The operand of a spatio-temporal relationship operator can be either a spatial or a temporal object associated with a query statement. Table 2 shows the macros, *spatial*, *valid*, and *transaction*, which are an extraction of geometry, valid time, or transaction time from a spatio-temporal object, respectively.

It is possible to employ a prefix function-style notation for reference macros or a postfix notation; we adopted the postfix notation according to the conventional query expression rules of SQL3. In addition, the spatio-temporal reference macros should be connected with the proposed spatio-temporal common relationship operators.

As Table 3 shows, the third group is the spatio-temporal common relationship operators whose operands can be either spatial or temporal objects. The topological relationship operators may be used in conjunction with the dimensional reference macros, spatial, valid, and transaction. Thus, each of the spatio-temporal common relationship operators can be operated differently as either a spatial or temporal relationship operator, depending upon its operands. There are possible combinations of spatio-temporal reference macros with spatio-temporal relationship operators for query expressions (Table 3). For example, it is only possible to combine either spatial reference operands or temporal reference operands, such as *Spatial/Spatial*, *Valid/Valid*, *Transaction/Transaction*. At this point, it is also important to define clearly whether the timestamp is a valid time or a transaction time because they are not interchangeable. The possible search expressions of a spatio-temporal query will yield a Boolean value such as true or false. As a result, spatial and temporal relationship operators in the select statement of SQL3 can be integrated in a uniform fashion by defining three reference macros and classifying existing topological relationship operators into three groups: exclusively temporal relationship operators, exclusively spatial relationship operators, and spatio-temporal common relationship operators.

V. HISTORY AGGREGATE OPERATOR

Existing relational database management systems (RDBMSs) support five aggregate functions: *count*, *sum*, *avg*, *min*, and *max*. Since the spatio-temporal databases provide the history management of spatial objects, it is necessary to design a new aggregate function that can retrieve all the historical information of an object automatically. To this end, we designed a history aggregate operator as follows.

1. What is the History Aggregate Operator?

A spatial object in a spatio-temporal database is represented by spatial data, non-spatial data, valid and transaction time periods, and its historical pointers. Among these data, geometric information, owners, and addresses of a spatial object change over time when an event occurs. The spatial object has its historical objects of either spatial or non-spatial data. To handle changes of state over time, we propose a history operator to retrieve all the historical interrogations of a spatial object (Definition 4).

Definition 4 A history aggregate operator is a selection operation in which all the histories of either the spatial or non-

Table 3. Usage of the spatio-temporal common relationship operators

Spatio-temporal relationship operators	Spatial use	Temporal use
Meets	- a.*spatial* meets b.*spatial*	- a.*valid* meets b.*valid* - a.*transaction* meets b.*transaction*
Overlaps	- a.*spatial* overlaps b.*spatial*	- a.*valid* overlaps b.*valid* - a.*transaction* overlaps b.*transaction*
Contains	- b.*spatial* contains a.*spatial*	- b.*valid* contains a.*valid* - b.*transaction* contains a.*transaction*
Equals	- a.*spatial* equals b.*spatial*	- a.*valid* equals b.*valid* - a.*transaction* equals b.*transaction*

spatial data of an object are extracted from a spatio-temporal database. The processing results may include many historical objects or be empty, depending on the object states. Figure 3 describes how an algorithm of the history operator is defined.

```
Procedure history(int fid, designated attribute type)
Input: feature identifier fid, designated attribute type
Output: feature's spatial, non-spatial historical information or both
Step 1. Identify input features from the window;
Step 2. Check current read mask from its feature layer;
Step 3. If (the current layer is not readable)
        Set the read mask of the current layer into readable;
Step 4. Read the feature id from current feature layer and its previous pointer;
Step 5. If (its previous pointer = merge)
        Retrieve all of its previous pointers;
        For each previous pointer, call history (int hid, designated attribute type);
Step 6. for (; its previous pointer != NULL; ) {
        Switch (designated attribute type) {
          Case NONSPATIAL :
             Extract the attribute historical record of the feature fid;
          Case SPATIAL :
             Extract the previous spatial history of the feature fid;
       } /* switch statement */
       Reset current previous pointer into its previous pointer;
    } /* for statement */
Step 7. Stop;
```

Fig. 3. Processing strategy of history aggregate operator.

As described in the spatio-temporal reference macros above, we also adopt a postfix notation as an interface with the history operator. There are two kinds of interfaces in the select statement of SQL3: One is the attribute history of the spatial object in the GIS and the other is the spatial history of objects designated by spatial attributes. A spatial object in spatio-temporal databases is handled as a tuple. The efficiency and convenience of use for the history operator is explained through the following queries:

Example 1 List all histories of the owners who owned the lot number of building 100.

```
SQL3> SELECT o.nonspatial.HISTORY
      > FROM buildings b, owners o
      > WHERE b.lotNumber = 100 and o.fid = b.fid ;
```

Example 2 List all the spatial histories of a building whose lot number is 100, where the spatial histories may be geometric real-world coordinates of its historical objects.

```
SQL3> SELECT *.spatial.HISTORY
      > FROM buildings
      > WHERE fid = 100;
```

Frequently, we may need to interrogate all spatial or non-spatial histories of designated objects. To do this, we suggest a history operator having an interface with the postfix notation as we described above. In our proposed spatio-temporal database, all the historical information for spatio-temporal objects can be retrieved by their historical information in the Attribute Relation and Attribute History Relation. This is possible because all the histories of either their spatial or non-spatial data are managed in the spatio-temporal database. Therefore, spatio-temporal queries can be expressed economically by using the proposed reference macros and the history aggregate operator, as we show in the following examples. Queries of examples 3, 4, and 5 below are examples in which the spatial and temporal relationship operators are integrated into the select statement of SQL3 in a uniformed manner. In addition, we can express all historical interrogations of either spatial or non-spatial data as follows:

Example 3 Retrieve all the moving paths of a car 'Seoul 4Ro-6179' on May 1, 1998.

```
SQL3> SELECT *.spatial.HISTORY
      > FROM Vehicle v
      > WHERE v.cno = 'Seoul 4Ro-6179' and
        v.valid overlaps PERIOD
              '01-May-1998, 01-May-98';
```

Example 4 List all owners and their address histories of building 'Jongro-gu A' since 1990.

```
SQL3> SELECT b.owner.HISTORY, b.address.HISTORY
      > FROM Buildings b, County c
      > WHERE b.name = 'A' and
        c.county = 'Jongro-gu' and
        c.spatial adjacent b.spatial and
        b.valid overlaps PERIOD '01-Jan-90,
              CURRENT_DATE';
```

Example 5 List all of the spatial histories of buildings adjacent to building 'Jongro-gu A' since 1990.

```
SQL3> SELECT b.spatial.HISTORY
      > FROM Buildings b, County c
      > WHERE b.name = 'A' and
        c.county = 'Jongro-gu' and
        b.spatial is_constained c.spatial and
        b.spatial meets c.spatial and
        b.valid overlaps PERIOD '01-Jan-90,
              CURRENT_DATE';
```

In the spatial case (Example 5), the query results will be visualized on maps in windows. Otherwise, the historical

attribute data can be viewed in tables having attributes with valid and transaction times.

2. The Search Statement for Spatio-Temporal Data

For any spatio-temporal database, we must define the data definition language and data manipulation language for the spatio-temporal data. In our system, we extend the following search statement in SQL3 by spatio-temporal reference macros, exclusive spatial relationship operators, exclusive temporal relationship operators, spatio-temporal common relationship operators, and the history aggregate operator in a uniform manner.

SELECT <attribute list>
FROM <relation list>
WHERE <attribute qualifications>

The following are examples of attribute qualifications for typical spatio-temporal queries. The attribute qualifications on the "where" clause may be combined with other attribute predicates, which are similar to the previous query expressions in SQL3.

<Temporal relationship operators>
a.*valid* overlaps b.*valid*
c.*valid* precedes
PERIOD '01-Jan-1997, 31-Oct-1997'
<Spatial relationship operators>
a.*spatial* overlaps b.*spatial*
b.*spatial* contains a.*spatial*

A brief BNF notation of a search statement for a spatio-temporal database extended by the spatio-temporal relationship operators and reference macros in SQL3 is described in APPENDIX. The conventions used to interpret the syntax rule are as follows: [] means it is required, {} is optional, and a bar (|) indicates OR. Lines (2)-(3) of the appendix are executed the same as in SQL2, but lines (1) and (4)-(6) in the "where" clause must be extended to retrieve spatio-temporal data. Thus, spatial, temporal, and spatio-temporal patterns can be handled in a uniform manner in the proposed spatio-temporal queries.

VI. REVIEWS OF CONTRIBUTIONS

In this section, we consider an implementation of the proposed spatio-temporal database scheme and review our research results as well.

1. Evaluation of Implementation Results

In the majority of temporal extensions to the relational model, valid and transaction times can be represented as a single chronon, sets of consecutive chronons, and arbitrary sets of chronons. In our approach, we represent the structure of valid and transaction times as a pair of points (begin, end) to provide the same query processing algorithm of the conventional relational model as described by Allen [30]. New tuples are added to the database by a tuple-level versioning method whenever any attribute of a spatial object is changed.

In this paper, we proposed a relational spatio-temporal data model, which is extended by valid time, transaction time, and a historical pointer of the spatial data model, called SDE [32]. As illustrated in Tables 2 and 3, we described the semantics and uses of spatio-temporal relationship operators and reference macros. The spatio-temporal relationship operators in SQL3 are then classified into the existing spatial and temporal relationship operators into three groups: exclusively spatial relationship operators, exclusively temporal relationship operators, and spatio-temporal common relationship operators. They were implemented by a layered approach with Oracle DBMS 7.2.3 and SDE 2.1 on Solaris 2.5.

Next, let us review the processing results of spatio-temporal queries in the databases.

Example 6 Find all the histories of a feature '201' since 1990 from the layer 'Parcel' of Table 4.

```
SQL3> SELECT *.HISTORY
      > FROM Parcel p
      > WHERE fid = 201 and
            p.Valid overlaps PERIOD
            '01-Jan-1990, CURRENT_DATE';
```

The above query processing starts with retrieving a current object '201' from the Feature Relation 'Parcel' and then continues with the previous pointer of the current record. Here, the history information is obtained from the Merge Relation 'Parcel_merge' because the previous pointer of object '201' is Merge, so two historical records, '200' and '300,' are obtained from the Merge Relation. With the historical records of '200' and '300,' each of all the histories of '200' and '300' are retrieved from the Feature History Relation repeatedly until their previous pointers are Null. Finally, we can see the query results of Example 6 computed from Table 4 as follows:

```
Result = {
    (201, Polygon, (0, 0, 200, 0, 200, 100, 0, 100),
          '01-Nov-97, Now', Merge),
    (200, Polygon, (0, 0, 100, 0, 100, 100, 0, 100),
          '01-Mar-81, 31-Oct-97', Null),
    (300, Polygon, (100, 0, 200, 0, 200, 100, 100,
          100), '01-Mar-81, 31-Oct-97', Null)};
```

Table 4. Example of a layer 'Parcel' in spatio-temporal databases.

fid	Type	Geometry	VTs	Prev
101	Polygon	(0, 0, 100, 0, 100,100, 0,100)	01-Jun-96	100
102	Polygon	(100,0,200, 0,200,100,100,200)	01-Jun-97	100
201	Polygon	(0, 0, 200, 0, 200,100, 0,100)	01-Nov-97	Merge
400	Polygon	(0, 0,100, 0, 100,100, 0,100)	01-May-98	400-1
500	Polygon	(100,0,200,0, 200,100,100,100)	01-May-98	500-1

(a) Parcel relation

hid	Type	Geometry	VTs	VTe	Prev
100	Polygon	(0, 0, 200, 0, 200, 100, 0,100)	01-Mar-81	31-May-96	Null
200	Polygon	(0, 0, 100, 0, 100, 100, 0, 100)	01-Mar-81	31-Oct-97	Null
300	Polygon	(100, 0, 200, 0,200,100,100,100)	01-Mar-81	31-Oct-97	Null
400-1	Polygon	(0, 0, 150, 0, 50, 100, 0, 100)	01-Mar-81	31-Apr-98	Null
500-1	Polygon	(150, 0,200, 0, 200,100,50,100)	01-Mar-81	3-Apr-98	Null

(b) Parcel history relation

fid	hid
201	200
201	300

(c) Parcel_merge

2. Comparisons of Previous Work

With major spatial event operations related to the history of an object, this paper presented the spatio-temporal data model, called ORSTDM and showed how to integrate the existing spatial and temporal relationship operators into the select statement of SQL3. Comparing our work with previous studies on spatio-temporal database systems, our contributions can be summarized as follows:

Integrating the Existing Spatial and Temporal Relation-ship Operators into SQL3 in a uniform fashion. Previous studies on spatial and temporal databases proposed many spatial and temporal relationship operators, including *disjoint*, *contains*, *inside*, *equal*, *meet*, *cover*, *covered_by*, and *overlap* between two connected spatial objects in two dimension space [31] and *before*, *equals*, *meets*, *overlaps*, *during*, *starts*, and *finishes* [30] in interval comparison operators. There has been no previous research result on unifying the spatio-temporal queries of SQL3 for information that has spatial and temporal components. Clalamunt [6] and Worboys [29] addressed only the needs of conceptually handling the spatial and temporal references in a uniform fashion. In this paper, however, we completely solved the problem of a unifying topology computation for different dimensions. The spatial relationship operators can integrate with the temporal relationship operators in the select statement of SQL3 by classifying the conventional spatial and temporal relationship operators into exclusively spatial relationship operators, exclusively temporal relationship operators, spatio-temporal common relationship operators, and defining three reference macros, *Spatial*, *Valid*, and *Transaction*.

In other words, we designed spatio-temporal relationship operations, including the reference macros, for extracting either spatial or temporal information, and classified the existing spatial and temporal relationship operators into three groups. In this way, we could actually unify the spatio-temporal relationship operators in SQL3, which have traditionally been handled by the different methods depending on the spatial or temporal queries, and simply express the spatio-temporal queries. Thus, all discrete time-varying historical information of spatial objects could be handled in a uniform method. However, there are complex spatio-temporal phenomena such as wildfire, that cannot be fully represented in this paper and these phenomena will require a continuous time model.

Defining the New History Aggregate Operator for Spatio-temporal Databases. We designed the new History aggregate operator to retrieve all the historical information of designated objects from stored spatio-temporal databases. It can be easily used in the select statement to represent a historical query, as the queries of examples 1 through 6 illustrate. In addition, all the histories of spatial objects in spatio-temporal databases can be retrieved through the History aggregate operator in SQL3 without constructing any application programs.

Enhancing the Expression Power of a Spatio-Temporal Query. As an example, Fig. 4(a) shows how the conventional spatial retrieval query language in SDE [32] is expressed. However, it can be described in a uniform fashion by using the exclusively spatial relationship operator, an exclusively

temporal relationship operator, and spatio-temporal reference macros (Fig. 4(b)). This Figure shows how any spatio-temporal query can be expressed very efficiently without losing any spatial analysis information, and how the expression power of spatio-temporal queries in SQL3 is improved.

In particular, the spatio-temporal queries with spatial and temporal search conditions in previous work on spatial and temporal databases cannot be described in the select statement of SQL3. For example, as Fig. 4(b) illustrates, spatio-temporal queries with spatial and time relationship operators cannot be integrated in existing spatial and temporal databases.

Notice that most of the spatio-temporal queries with spatial and temporal predicates cannot be described at the same time in the select statement of SQL2 (Fig. 5). However, with the select statement of SQL3, we solved those problems completely by defining three reference macros and classifying the existing spatial and temporal relationship operators into three different groups: exclusively spatial relationship operators, exclusively temporal relationship operators, and spatio-temporal common relationship operators.

Designing a Unified Relational Spatio-Temporal Data Model. Above all, this paper proposed a relational spatio-temporal data model, called ORSTDM, in which spatial data actually integrates temporal data by extensions of time intervals, called valid and transaction times, and a historical pointer. The ORSTDM can manage and retrieve all historical information of objects by which spatial event operations occur.

This model provides more practicable solutions than the previous spatio temporal data models [6], [18] because it can be

```
Feature zonef, bldgf;
Zonelayer = s100;
Bldcnt = 0;
For (ret = SE_get_feature_by_layer(zonelayer, &zonef,
     Zone= COMMERCIAL); ret = SUCCESS;
     Ret = SE_get_next_feature(&zonef))   {        /* for */
  SE_set_search_by_feature(&zonef);
  For (returncode=SE_search(bldlayer, SM_AI, &bldgf,
       USE=RESIDENTAL, VTs >= 01-Jan-96 and
       VTe < 31-Dec-97); ret= SUCCESS; ret =
       SE_next_search(&bldgf))
     Bldcnt++;
} /* outer loop */
```

(a) A spatio-temporal query in SDE

```
SQL3> SELECT T.spatial, count(*)
      FROM s100 S, s100 T
      WHERE S.zone = COMMERCIAL and
        T.USE = RESIDENTAL and
        T.spatial intersects S.spatial and
        T.valid overlaps PERIOD '01-Jan-96, 31-Dec-97';
```

(b) A spatio-temporal query in SQL3

Fig. 4. Comparison of query representations in SDE and SQL3.

Items	Case of overlapping spatially and temporally	Case of overlapping spatially and meeting temporally
SQL2	SELECT *.geometry, count(*) FROM relation S, relation T WHERE S.fid = 100 and S overlaps T and // cannot describe S overlaps PERIOD '01-Jan-96, 31-Dec-97';	SELECT * FROM relation S, relation T WHERE S.fid = 100 and S overlaps T and // cannot describe S meets T; // cannot describe
SQL3	SELECT *.*spatial*, count(*) FROM relation S, relation T WHERE S.fid = 100 and S.*spatial* overlaps T.*spatial* and S.*valid* overlaps PERIOD '01-Jan-96, 31-Dec-97';	SELECT * FROM relation S, relation T WHERE S.fid = 100 and S.*spatial* overlaps T.*spatial* and S.*valid* meets T.*valid*;

Fig. 5. Comparison of spatio-temporal query representations in SQL2 and SQL3.

used as a tool for history management of spatial objects directly without any changes. Finally, we can summarize our model as follows: the proposed spatio-temporal database system integrates the spatial, temporal, and spatio-temporal relationship operators into SQL3 in a uniform manner and at the same time supports all historical interrogation of spatio-temporal data.

VII. CONCLUSIONS

The features of spatio-temporal data in the real world vary over time; these features are managed in a GIS. However, conventional GIS software cannot handle time-varying data because it neither controls the historical information of spatial objects nor supports their spatio-temporal operations. Furthermore, previous spatio-temporal data models and query language cannot handle all the history management of spatial objects.

To overcome the limitations of previous models, we suggested the spatio-temporal database scheme, spatio-temporal relationship operators, and a history aggregate operator to support historical queries in SQL3 and to support all historical interrogation of spatio-temporal data from users. We also described spatio-temporal reference macros for extracting the designated dimensional information of spatio-temporal data, as well as a brief BNF statement of insert and search statements in SQL3, which are extended by data types, a time clause, and our proposed spatio-temporal operations for spatial and temporal use. We then implemented our designs and demonstrated them by examples of spatio-temporal queries. In the future, we will study spatio-temporal indices and spatio-temporal join processing algorithms to further support convenient query processing.

ACKNOWLEDGEMENT

The author would like to thank the anonymous reviewers and Juli Scherer for their helpful valuable comments and suggestions.

APPENDIX: A brief BNF description for the spatio-temporal search statement

SELECT <extended attribute list> (1)
FROM <relation list> (2)
WHERE <attribute qualifications>; (3)
<attribute qualifications>
::= [object.*valid* <exclusively temporal relationship operators>
<valid timestamp reference expressions>] | (4)
[object.*transaction* <exclusively temporal
relationship operators> <transaction timestamp
reference expressions> | (5)
[object.*spatial* <exclusively spatial relationship
operators> object.*spatial*] ; (6)
<valid timestamp reference expressions>
::= object.*valid* | timestamp expressions;
<transaction timestamp reference expressions>
::= object.*transaction* | timestamp expressions;
<timestamp expressions>
::= Timestamp '<datetime>' |
PERIOD '<datetime> - <datetime>';
<exclusively spatial relationship operators>
::= DISJOINTS | IS_CLOSEST | IS_SHORTEST | PIP |
INS_EVELOPE | CENTROID_INS_POLYGON |
<spatiotemporal common relationship operators> ;
<exclusively temporal relationship operators>
::= BEFORE | AFTER | PRECEDES | STARTS |
FINISHES |
<spatiotemporal common relationship operators> ;
<spatiotemporal common relationship operators>
::= EQUALS | MEETS |OVERLAPS |CONTAINS ;
<extended attribute list>
::=<attribute list> {*.spatial*} {*.HISTORY}* ;

REFERENCES

[1] M.F. Worboys, *GIS: A Computing Perspective*, Taylor & Francis Publishers, 1995, pp.1-4.

[2] A. Beller, T. Giblin, K.V. Le, S. Litz, T. Kittel, and D. Schimel, "Temporal GIS Prototype for Global Change Research," *GIS/LIS Proc*., vol. 2, 1991, pp. 752-765.

[3] L.D. Montgomery, *Temporal Geographical Information Systems Technology and Requirements: Where we are today*, Master's thesis, The Ohio State University, 1995.

[4] A. Renolen, *Temporal Maps and Temporal Geographical Information Systems*, Dept. of Surveying and Mapping (IKO), The Norwegian Institute of Technology, Feb. 14, 1996.

[5] M. Yuan, "Temporal GIS and Spatio-Temporal Modeling," *The 3rd Int'l Conf./Workshop Integrating GIS and Environmental Modeling*, June 21, 1996.

[6] C. Claramunt, "Managing Time in GIS: An Event-Oriented Approach," *Recent Advances in Temporal Databases, Workshops in Computing Series*, Edited by J. Clifford and A. Tuzhilin (eds), Berlin:Springer-Verlag, 1995, pp. 23-42.

[7] S. Price, "Modeling the Temporal Element in and Land Information Systems," *Int'l J. of Geographic Information systems*, vol. 3, no. 3, 1989.

[8] J.M. Carey, D.J. Dewiit, and S.L. Vandenberg, "A Data Model and Query Language for EXODUS," *Proc. ACM SIGMOD*, 1988, pp. 413-423.

[9] M.J. Egenhofer, "Spatial SQL: A Query and Presentation Language," *IEEE Trans. on Knowledge and Data Engineering*,

vol. 6, no. 1, Feb. 1994.
[10] R.H. Guting, "Gral: An Extensible Relational Database System for Geometric Applications," *Proc. of the Fifteenth Int'l Conf. on Very Large Data Bases*, Amsterdam, 1989, pp. 33-43.
[11] L.M. Haas and W.F. Cody, "Exploiting Extensible DBMS in Integrated Geographic Information Systems," *Advances in Spatial Databases, 2nd Symposium, SSD '91*, Zurich, Switzerland, Aug. 28-30, 1991, pp. 423-450.
[12] M. Stonebraker, L. Rowe, and M. Hirohama, "The Implementation of POSTGRES," *IEEE Trans. on Knowledge and Data Engineering*, vol. 2, no. 1, 1990.
[13] J. Orenstein and F. Manola, "Probe Spatial Database Application," *IEEE Trans. on Software Engineering*, vol. 14, no. 5, 1988, pp. 611-629.
[14] N. Roussopoulos, C. Faloutsos, and T. Sellis, "An Efficient Pictorial Database System for PSQL," *IEEE Trans. on Software Engineering*, vol. 14, no. 5, 1988, pp. 639-650.
[15] R.T. Snodgrass, *The TSQL2 Temporal Query Language*, Tucson, AZ, Kluwer Academic Publishers, 1995.
[16] A.U. Tansel, "Adding Time Dimension to Relational Model and Extending Relational Algebra," *Information Systems*, vol. 11, no. 4, 1986, pp. 343-355.
[17] R.T. Snodgrass, "The TSQL2 Temporal Query Language," *The TSQL2 Language Design Committee*, Kluwer Academic Publishers, 1995.
[18] D. Peuquet and N. Duan, "An Event-Based Spatiotemporal Data Model (ESTDM) for Temporal Analysis of Geographical Data," *Information Systems*, vol. 9, no. 1, 1995, pp. 7-24.
[19] N.A. Lorentzos and Y.G. Mitsopoulos, "SQL Extension for Interval Data," *IEEE Trans. on Knowledge and Data Engineering*, vol. 9, no. 3, May/June 1997, pp. 480-499.
[20] M.F. Worboys, "A Unified Model for Spatial and Temporal Information," *The Computer J.*, vol. 37, no. 1, 1994.
[21] T.S. Cheng and S.K. Gadia, "A Pattern Matching Language for Spatio-Temporal Databases," *CIKIM'94*, 1994, pp. 287-295.
[22] J.Y. Lee, B.I. Ahn, and K.H. Ryu, "A Historical Extension for SDE Data Model," *The Trans. of the Korea Information Processing Society*, vol. 5, no. 9, Sept.1998.
[23] J.Y. Lee, S.J. Lee, and K.H. Ryu, "SQL Unification of Spatiotemporal Operations for AVLS: SQL/ST," *The Int'l Association of Management 16th Annual Conf.*, Aug. 5-8, 1998.
[24] M. Bohlen, C.S. Jensen, and B. Skjellaug, "Spatio-Temporal Database Support for Legacy Applications," *A Time Center Technical Report TR-20*, July 9, 1997.
[25] Province of British Columbia, Ministry of Environment, Lands and Parks, Surveys and Resource Mapping Branch, "British Columbia Specification and Guidelines for Geomatics. Reference Series Volume 1: Spatial Archive and Interchange Format: SAIF Formal Definition," Release 3.1, Apr. 1994.
[26] N. Tryfona and T. Hadzilacos, "Logical Data Modeling of Spatio-Temporal Applications: Definitions and a Model," *CHOROCHRONOS, Technical Report CH-97-03*, Dec. 1997.
[27] M. Erwig, M. Scheider, and R.H. Guting, "Temporal and Spatio-Temporal Data Models and Their Expressive Power," *CHOROCHRONOS, Technical Report CH-97-10*, Dec. 1997.
[28] Young-Ok Shin et al., "TATS: an Efficient Technique for Computing Temporal Aggregates for Data Warehousing," *ETRI J.*, vol. 22, no. 3, Sept. 2000, pp. 41-51.
[29] M.F. Worboys, "A Data Model for Information with Spatial and Bitemporal Components," *Technical Report TR93-06*, Dept. of Computer Science, University of Keele, Feb. 1993.
[30] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. of the Association of Computing Machinery*, vol. 26, no. 11, Nov. 1983, pp. 823-843.
[31] M. Molenaar, O. Kufoniyi, and T. Bouloucos, "Modeling Topologic Relationships in Vector Maps," *Proc. Int'l Symposium On Spatial Data Handling* (*SDH*), 1994
[32] Environmental Systems Research Institute, Inc., *Introduction to SDE™*, Environmental Systems Research Institute, Inc., 1996.

Jong-Yun Lee received the BS and MS degrees in computer engineering from Chungbuk National University in 1985 and 1987, respectively and the PhD degree in computer science from Chungbuk National University, South Korea, in 1999. He worked as a Research/Project Leader in the Software Research and Development Institute of Hyundai Electronic Industrial Company Ltd. and Hyundai Information Technologies Company Ltd., South Korea, from 1990 to 1996. He also worked for Bit Computer Cooperation in 1989. He is currently an Assistant Professor in the Department of Information and Communication Engineering at Samchok National University in South Korea, from March, 1999. His current research interests include temporal databases, spatio-temporal databases, spatial databases, object-oriented databases, and GIS. He has served as a Proposal Evaluator at a national level in GIS. He is a member of the IEEE and the IEEE Computer Society, Korea Information Processing Society, and Korea Information Science Society.