Logical Data Modeling of SpatioTemporal Applications: Definitions and a Model

Nectaria Tryfona
Department of Computer Science
Aalborg University, Fredrik Bajersvej 7E,
Aalborg Øst, Denmark
tryfona@cs.auc.dk

Thanasis Hadzilacos
Computer Technology Institute
University of Patras, P.O Box 26110
Patras, Greece
thh@cti.gr

Keywords: spatiotemporal databases, logical design, spatiotemporal objects, spatial attributes, temporal layers, spatiotemporal data definition language.

Abstract

Dealing with spatiotemporal applications at the logical phase of database design reveals a set of data peculiarities which the already existing models can not serve satisfactorily. In this paper, the ontologies of a spatiotemporal environment –namely the basic concepts of objects, attributes and relationships with spatial and temporal extent, as well as operations on them– are defined. Based on these, specifications of modeling tools for the design of spatiotemporal information at the logical phase are given. A formal, yet practical, SpatioTemporal Relational data Model (STRM) is presented as part of a full automatable application design methodology; it provides a small set of representational constructs (relations, layers, virtual layers, object classes, and constraints -all with spatial and temporal extent) on top of well-established models. The power of modeling and ease-of-use of the discussed approach is demonstrated by an example taken from a real application.

1 Introduction

The development of *spatial information systems with temporal dimension* has been a long-term user requirement. Cadastral information systems with the need to record history of land parcels existence, navigational systems dealing with possible routes of vehicles in time, and forest management systems with fire and reforestry history are some typical examples of this emerging need.

Although valuable results from the research communities dealing with either spatial or temporal aspects of the aforementioned systems *do* exist (see for example [5] [12] [16] for the spatial requirements, and [9] [10] [11] for the temporal ones), it is only the last years that the analysis, design and implementation of applications with information varying over space and time became the new research challenge.

Applications dealing with *spatiotemporal information* fall into one of the three following categories:

(a) applications with moving objects, such as navigational; in these, objects change position but not shape in time, e.g., a “car” moving on a road network,

(b) applications involving objects located in space, whose characteristics as well as their position may change in time; for example, in a cadastral information system, “landparcels” change positions, but they don't "move", and

(c) applications with objects which integrate the above two behaviors; for example, in environmental applications, "pollution" is measured as a moving phenomenon and changes properties and shape over time.

In all three cases, data modeling has been ruled primarily from the spatial nature of data (see for example [4]). Moreover, because of the volume and complexity of geographic[1] information, their design is focused on the physical organization of data (i.e., the internal schema of database terminology) and tends to be tightly coupled with specific spatial information management techniques.

Our position is that spatiotemporal application development should use standard application design methodologies, while on the other hand, such methodologies should be extended to accommodate the peculiarities of space and time. Some of these peculiarities are shared by several other application areas: large and complex objects, long design transactions, intricate integrity constraints are common to the so-called new application areas [17] which include, among others, VLSI design and multimedia applications. Other peculiarities are exclusive to spatiotemporal systems: spatial relationships, versions of objects and spatiotemporal integrity constraints are among them.

[1] The terms “spatial” and “geographic” are used interchangeably.

This work focuses on the peculiarities and appropriate modeling mechanisms for the logical design of spatiotemporal databases. We view our subject as but one of the stages of a complete application design methodology, tailored to the needs of the spatial and temporal nature. The contribution is twofold:

(a) the presentation, in a common framework, of the ontologies of a spatiotemporal information system: objects, attributes and relationships with spatial and temporal extent are defined. They have been derived from a generic analysis of spatiotemporal application design requirements based on theoretical research [7] as well as applied project experience ([1] for the development of the Greek Cadastral System, and [18] for the development of a network utility management system).

(b) the translation of these concepts into a logical model, the SpatioTemporal Relational Model (STRM) –an extension of the relational one. The model is formal, yet practical, and constitutes part of a full automatable application design methodology [8]. It provides a small set of concepts on top of well-established models, so that it can be utilized with minimum training and resistance. STRM is independent of platform –as we don't extend commercial relational systems– but depended on specific architecture (relational).

The language of STRM can be used to, formally (a) define the fundamental objects, attributes and relationships of the application domain, and (b) express user views of the database environment and answer queries.

We are not aware of any other related work in this area. The substantial work on logical geographic database design which appears on [5] is related to ours: the concepts are similar, but at a different level of abstraction; it deals with the definition of a spatial algebra, while ours resides on the modeling constructs.

The rest of the paper is organized as follows: Section 2 describes the basic ontologies of the spatiotemporal environment. Section 3 discusses the peculiarities and modeling needs of such an environment at the logical design phase. Based on these, Section 4 integrates into the relational model the appropriate components for spatiotemporal data handling, and presents the formal syntax of the Spatiotemporal Relational Data Model; 4.2 discusses spatial and temporal aspects; the Appendix gives its annotated syntax. Section 5 contains a substantial practical example taken from a real project for the maintenance of a network utility management system [18], while Section 6 concludes with future research plans.

2 Definitions and Notations

In this section, we present the components (or ontologies) of a spatiotemporal application environment and their interrelationships. A more thorough investigation on the issue of space, spatial objects, and spatial attributes and their semantics can be found in [8]. Here the combination of space with temporal issues is applied. Our scope, in presenting the model, is the simplicity and integration of both spatial and temporal concepts under the same framework in order to communicate without ambiguities.

Objects, behavior and relationships. A database is a set of objects $o_1,o_2,\ldots,o_n$, which represent part of the real world; each object belongs to an object class O_1, $O_2,\ldots,O_n$. An object class O_i is characterized by a set of static properties, or attributes $A_1^i,\ldots, A_m^i$, and a set of dynamic properties or methods $M_1^i,\ldots, M_k^i$. Each attribute A_j^i , with $1\le j\le m$ is associated with a domain D_{Aj}^i, which is an unrestricted set of values. Domains are implemented by data types, and methods are implemented by procedures. Methods are the only means to access the attributes. Static and dynamic properties (attributes and methods) construct the behavior of an object. So, each object in a database instance is represented by a set of values each belonging to the domain of the corresponding attribute of the object.

Object classes can be interrelated via *relationship sets* (or association sets). A relationship set *R* of degree *n* is a condition on a tuple of *n* values of attributes possibly from different object; in other words, each relationship instance *r* in *R* is a *n-tuple* $r<o_1,o_2,\ldots,o_n>$.

Spatial objects. In real world, most objects have position, which is their link to space. In spatial information systems we are interested in the position of some objects; these are called geographic or spatial objects GO. Position p is a function on all and only on spatial objects; for each object it returns parts of space. This approach places no restriction on the model of space. For example, 2-dimensional Euclidean space is modeled using an object class homomorphic to R^2, i.e., the space contains sets of points. The object class homomorphic to R^2 corresponds to vector systems, while the one homomorphic to Z^2 corresponds to raster systems. For the rest of the paper, we consider R^2 as our space.

Positions of geographic objects are also called *geometric figures*. Geometric figures can be points, lines, regions or combinations thereof.

Spatial objects may or may not move in space, depending on the application domain. For example, in a navigation system, a moving car is a spatial object whose position changes over time, while in a cadastral

application, a position of a landparcel may change while, the object is not considered as a "moving" one.

Spatial attributes. In classical application areas objects are characterized by a set of properties or attributes. In spatiotemporal application environments, apart from attributes with the classical meaning (i.e., descriptive attributes), objects have also spatial or space-depending attributes SA. Spatial attributes are properties of space which indirectly become properties of objects situated at some position in space, i.e., geographic objects inherit them from space. A spatial attribute is a function whose domain is space and range is any set. Its value depends on position only, and not on the object itself. Spatial attributes are represented as layers.

In the case of applications dealing with moving objects, the dominant spatial attribute is the area covered by the position of an object as it moves; in other words, it is the existence or not of object in space.

Layers and operations on layers. Informally speaking, a layer L (or thematic map or field) is a representation of a spatial attribute. There are two basic types of layers: (a) those representing functions with continuous range; for example "temperature", or "erosion", and (b) those representing functions with range with discrete values; for example "landparcels" represented as regions. In other words, a layer is a set of geometric figures (which are points, lines, regions or combinations of them) with associated values. Thus, a layer can be seen either as a function from geometric figures to attributes:

$L: G \rightarrow D_1 x D_2 x \ldots x D_k$

where G is a finite set of geometric figures g, or a relationship with the geometric figure as the key attribute [2].

In case (a) the layer represents a set of areas with different values of the same attribute or positions of objects in space, while in (b) we visualize it as homogeneous area.

From the user's point of view, layers often represent derived information. In such cases they are called *virtual* and they are related to the way users need to view data –a concept similar to that of database views.

In manipulating layers it is sufficient to be able to modify the geometric figures (i.e. the domain of the function) or the attributes (i.e. the range of the function) and to combine such changes through function composition. Systematic research done by our group, shows that there are four types of operations forming the layer algebra [2]:

(a) Operations (COMPUTE ATTR) on a single layer function, which change its range (by adding, change or delete non-geometric attributes), but leave the same domain, i.e., same geometric figures. Such operations are used to derive computable attributes, such as "population density".

(b) Operations (COMPUTE SPATIAL) which operate geometrically on the domain of a layer function to produce a layer function with a new set of geometric figures as domain. For example: "given a layer with lines and regions (representing rivers and lakes) construct a layer, which consists of a buffering zone around the rivers and lakes".

(c) Reclassification (RECLASS), which operates on a single layer and concatenates adjacent figure if their range, i.e., non-geometric attributes, are identical. For example, "connect areas with the same soil type".

(d) Overlay (OVERLAY), which takes two layer functions and produces a new one with the geometric overlay as domain and the combination of the ranges. The resulting layer keeps the descriptive attributes from both layer-components.

Spatial relationships. Spatial (or spatiotemporal, if they have temporal extent) objects are related to each other in space. Relationships among geographic objects are actually conditions on objects' positions and are called spatial or geographic relationships. Spatial relationships among geographic objects are actually conditions on object's positions. The set of spatial relationships is subdivided into three subsets: topological (e.g., inside, outside, etc), directional (e.g., North, NorthEast, etc.) and metric (e.g., 5km away from) relationships. Spatial relationships are translated into spatial integrity constraints of the database. Conceptual geographic models should lead to straightforward solutions for explicitly storing them in the logical and physical levels -a common practice despite the fact that some of them are derivable from object position.

Geometric operations. Operations on spatial objects, which include their position are called geometric operations. Strictly speaking these operations are defined on objects and not on their positions. It simplifies the statement of the definitions, however, and creates no confusion if we describe them as acting on positions (see [6]) for the formal treatment, on which the following set of primary geometric operations is based). The set of operations includes: (a) primary operations whose range is real, like distance and area, (b) primary operations whose range is a geometric type, like union, difference, nodes, etc., (c) derived operations whose range is real, like perimeter and length, and (d) derived operations, whose range is a geometric data type, like intersect.

Temporally-indexed components. The temporal research community has already proposed many variations and alternatives about recording time in databases. Here, we

present two fundamental types of time used in temporally-extended databases; however our model does not depend on this: (a) valid time, showing when facts about database components (i.e., objects, relationships, attributes of objects, layers) are true (b) transaction time, showing the time a fact is present in the database as stored data [13].

Objects, attributes and relationships, with or without spatial dimension, can be temporally indexed, resulting into spatiotemporal components. Two basic techniques exist in recording time: the time-point and time-interval approach. In the first, database components get a timestamp, while in the second they are recorded in time intervals. A time interval is the time period consisting of chronons between two time points, the starting and the ending point, consisting of equal-sized, indivisible chronons.

Spatiotemporal integrity constraints. With the use of predicate calculus (atomic formulas, negation ($\neg$), conjunction ($\wedge$), disjuction ($\vee$), and universal ($\forall$) and existential ($\exists$) quantification) and the above describe components we can express complex spatiotemporal integrity constraints or queries or facts [14]. Spatiotemporal constraints (or conditions) are imposed either by the designer for the shake of database integrity, or by the user based on applications requirements.

3 Spatiotemporal requirements at the logical design phase

Logical data modeling of applications is part of a design methodology including –at least– the phases of (a) requirements analysis and specification, (b) conceptual modeling, (c) logical modeling and (c) implementation. Its scope is to provide a formal, record-oriented, system independent, yet system implementable, description of the static properties and integrity constraints of the application domain. Models used extensively in this phase are the relational, network and hierarchical, and are called *syntacti*c as they emphasize in the syntax of the data.

Logical design leads to two schemata: (a) the logical schema, which includes the definitions of the objects, attributes and relationships, and (b) the external schema, which is the integration of all user views.

Theoretical research resulting from the translation of the spatiotemporal components described in Section 2 into the logical design phase, as well as experience from real applications [1] [18] shows that, for the logical schema the designer needs techniques and tools to:

(a) Define spatial attributes, organize them into layers and give them the temporal dimension (resulting into spatiotemporal attributes). For example, "soil_erosion" is a property of space organized in a layer, representing sets of regions (with different values). Time (valid and transaction) need to be recorded for the layer.

(b) Specify non-spatial attributes, organize them in relations, assign them to time and connect them to spatial ones. For example, a relation "types_of_erosion" describing different kinds of erosion (i.e., descriptive information such as "10%-30%", "30%-60%" "60%-90%", etc) is connected to the corresponding layer ("soil_erosion"), which has codes (such as "low", "medium", "high", etc) for the represented type of erosion in each region.

(c) Specify spatiotemporal integrity constraints, imposed either by the user, or by the designer for the integrity of the database. For example: if the "soil_erosion" is more than 60%, and a "pipe" of a utility network located there, is more than 8 years old, then it needs to be replaced in the next year in order not to break.

(d) Represent versions of objects. A typical question is how many different versions of "pipe-segments" and "networks" do we have, as segments change entirely, or change positions in the same network?

For the external schema (user view) the designer needs tools to:

(a) Define complex attributes, computable from others, spatiotemporal or not. For example, "soil_type" depends on "soil_erosion" and "soil_acidity".

(b) Organize complex (virtual) layers defined on common geometric features. (We use the term virtual layers to denote such collections of computatble spatial attributes.) Typical example: "soil_type" of an area.

(c) Define complex classes of spatiotemporal objects. Spatiotemporal objects have positions of some geometric type (point, line, region or combination there of); they combine spatial and non-spatial attributes, with temporal or not, dimension from several layers and adhere to spatiotemporal or other integrity constraints. A "landparcel" is an object with "soil_type" and "soil_erosion". A "utility network" passes through it, consisting of "pipe-segments" with "erosion" which depends on the "soil_erosion".

4 The SpatioTemporal Relational Data Model

The standard data model for logical database design is the relational. Its power and elegance stems from the fact that it uses a single construct: the relation. Five fundamental closed operations are defined on relations (selection, projection, union, difference, Cartesian product) and all the others are derived. For spatiotemporal

applications, however, the resulting representation is inadequate. For example, if layers are represented with plain relations, operations such as overlaying and reclassification can not be derived from the fundamental ones. (For an extensive discussion, refer to [7]). What usually happens when the relational model is used for logical modeling of spatiotemporal applications is that these operations are hidden in the physical level and so, the system in tied to some specific implementation.

Having the relational model as a reference point (there are many good reasons to make use of the sound theory behind it) our primal goal is to model spatiotemporal attributes, i.e. spatial attributes in time or, in other words, temporal layers or fields. On the other hand, we need to model spatiotemporal objects; these are associations of sets of attributes with a geometric figure. The two approaches −layers and objects− are *orthogonal* and are integrated in the SpatioTemporal Relational data Model (STRM) syntax that follows.

STRM provides a language for the definition of: (a) relations, used for non-spatial objects and relationships, (b) layers, to represent fields, (c) object classes, to represent spatiotemporal entities that have characteristics from more than one layers, (d) constraints among objects or layers, used to describe spatiotemporal conditions.

Using relations, a well-known construct, is one of the advantages of STRM. Layers on the other hand are special relations: their keys are geometric figures (point, line, regions) while other attributes (so-called spatial) are assumed to vary on those geometric figures. Keys are implicit. Correct layer selection is analogous to correct relational design.

4.1 The syntax

The STRM allows for the definition of relations, layers, virtual layers, all with temporal dimension, as well as object classes and constraints. This section summarizes the formal syntax, while the appendix gives the annotated syntax. Upper case letters denote reserved words. Lower case words denote variables which stand for arbitrary names. Words in lower case with capitalized initial denote variables with restricted range. (e.g., Domain stands for one of INTEGER, STRING, etc., whereas $attr\_name_i$ stands for any alphanumeric string that is not a reserved word). Clauses in < > are optional arguments; in (,...) are repeatable; in { ... | ... } are alternatives (one of); in ( ) indicate grouping of arguments. Italics in the appendix explain the semantics.

Logical Schema

DEFINE RELATION rel_name
($attr\_name_1$, $Domain_1$, <KEY>),…,
($attr\_name_k$, $Domain_k$, <KEY>)
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>

DEFINE LAYER n <layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
ATTR ($attr\_name_1^n$, $Domain_1^n$, <UNIQUE>),…,
($attr\_name_{mn}^n$, $Domain_{mn}^n$, <UNIQUE>)
GEOMETRIC TYPE Geometric_type
<POSITIONING Coordinate_system>
<CONSTRAINT spatiotemporal_condition>

External Schema

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS COMPUTE ATTR
(m,$new\_attr\_name_1^n = f_1$ ($attr\_name_1^m$,…,$attr\_name_{jm}^m$)
,…,
m, $new\_attr\_name_k^n = f_k$ ($attr\_name_1^m$,…,$attr\_name_{jm}^m$))

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS COMPUTE SPATIAL
(m, $new\_attr\_name^n$, f($attr\_name_1^m$,…,$attr\_name_{jm}^m$))

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS OVERLAY (k,m)

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS RECLASS OF (m, $attr\_name_1^m$,…,$attr\_name_{jm}^m$)

Recursive definition of layer identifying expression

Every layer or virtual layer identifier is a layer identifying expression.

If m, n are layer identifying expressions, then so are:

i. COMPUTE ATTR
(m, $new\_attr\_name_1^n = f_1$ ($attr\_name_1^m$,…,$attr\_name_{jm}^m$)
,…,
m, $new\_attr\_name_k^n = f_k$($attr\_name_1^m$,…,$attr\_name_{jm}^m$))

ii. COMPUTE SPATIAL
(m, $new\_attr\_name^n$, f($attr\_name_1^m$,…,$attr\_name_{jm}^m$))

iii. OVERLAY (k,m)

iv. RECLASS OF (m, $attr\_name_1^m$,…,$attr\_name_{jm}^m$)

DEFINE OBJECT CLASS obj_class_name
<GEOMETRIC TYPE Geometric_type>
<SUBTYPE OF sup_obj_class>
<ON LAYERS $layer\_id_1$,…,$layer\_id_k$>
<WITH ATTRIBUTES $attr\_name_1$,…,$attr\_name_m$>
<CONSTRAINT constraint_name>

DEFINE CONSTRAINT constraint_name
ON {object_class_name | layer_id}
AMONG {obj_class_1,…,obj_class_k
|$layer\_id_1$,…,$layer\_id_k$}
AS spatiotemporal_condition_specification

4.2 Discussion

Temporal aspects. Options in definitions of layers, relations and objects such as:

<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>

provide a simple but efficient –as it is shown in the following example– way for handling time. Temporal databases and temporal reasoning are a subject on their own [9] [11] intricate semantics [10] and no pretension is made that STRM fully covers the need to represent time. This is the first attempt to integrate space and time in the relational model.

Valid and transaction time are recorded at the level of tuples. Another approach would have been to record them at the attribute level. The proposed model is open to this kind of modifications.

Geographic objects don't have temporal extent. The idea is that the user computes geographic objects whenever he/she needs them, and they don't change any further. Changes in an object (coming from changes on layers forming it) result in a new object.

On the other hand, associated relations and layers may have different versions in time; for example, while a layer remains the same, the descriptive characteristics that are connected to it through the relation may change.

Spatial aspects. STRM gives the ability to define a set of, otherwise similar, layers, relations and virtual layers which can be used to model the evolution of the characteristics of space in time (snapshots). Information is recorded in time-intervals.

A layer is used for representing spatiotemporal attributes. In STRM layers are functions whose domain is (or equivalently: relations whose key is) a set of geometric figures (points, lines, regions or combination, as specified by GEOMETRIC TYPE); and whose range is the Cartesian product of the domains of the attributes depicted in the layer. Virtual layers are computed from other (virtual or not) layers with the operations of the layer algebra and the geometric operations described in Section 2. If a virtual layer is based on at least one temporal layer, it is also temporal; alternatively, the designer may choose to use a specific time version of a temporal layer in the definition of a virtual layer.

Relations are used for representing relationships and non-spatial entities.

An object class is defined by layers in the following way: take the union of geometric figures of all layers in the ON LAYERS clause; restrict the set to those having the specified GEOMETRIC TYPE; these objects have as only attributes the ones mentioned in the WITH ATTRIBUTES clause; finally, only those figures satisfying the constraint are included in the object class.

Finally, in the case of applications dealing with moving objects, which don't change shape, e.g., a car in a navigational system, the concept of layer is not a dominant one. However, interesting questions can be answered, such as "what are the most busy road intersections of area X, based on the car routes"; in this case a selection of the intersections resulting from the overlay of the map of the area X and the layers indicating each car's route answers the query.

5 Example of Use

Lets assume the following excerpt from a real application [18] dealing with the maintenance of water pipes of a specific area. To find information about the condition of the *water pipes*, it is essential to keep track of the *erosion of soil type* on which the water pipes are located.

a. Starting from the year 1994 (this is the transaction time, i.e., time the information is included in the database, while valid time, i.e., time of existence, starts earlier), we record *erosion* and *water pipes*. Erosion is recorded only as a map (layer), while for water pipes we keep data about their *material type*. Wherever the transaction and valid time do not have a STOP TIME means that the system is still open in recording data. Also, we assume that we know the age of water pipes (i.e., the valid time) and we record it.

DEFINE LAYER 1 erosion
START_T_TIME, 1994
ATTR (erosion, STRING, UNIQUE)
/* two different geometric figures can not agree on this argument */
GEOMETRIC TYPE POINT
/* the dimensionality of the figures is 0, the representation is a set of points and we visualize it as surface*/

DEFINE LAYER 2 waterpipes
START_V_TIME, 1980
START_T_TIME, 1994
ATTR (waterpipe-id, INT, UNIQUE)
GEOMETRIC TYPE LINE

DEFINE RELATION waterpipes
(waterpipe-id, INT, KEY)
(material, STRING),
START_V_TIME,1980
START_T_TIME, 1994

b. We also record the *landparcels* in the same area staring from 1995.

DEFINE LAYER 3 landparcel
START_T_TIME, 1995
ATTR (lp-id, INT, UNIQUE)
GEOMETRIC TYPE REGION

c. Lets assume now (1997) that we want to find all the segments (or parts) of water pipes which: (a) are more than 10 years old, (b) are located totally inside landparcels and (c) are located in a high risk (erosion) area, so that we change them before they break.

c1. We classify points with same *erosion* into the same set (forming regions). (The way classification methods operate is well-known to the scientific areas of geography and cartography and it is outside the scope of this paper).

DEFINE V_LAYER 4 class_erosion
START_V_TIME, 1997
START_T_TIME, 1997
AS RECLASS OF (1, erosion)

c2. We overlay *landparcels* and the classes of *erosion*, the resulted layer has attributes from both component layers.

DEFINE V_LAYER 5 landparcel_erosion
START_V_TIME, 1997
START_T_TIME, 1997
AS OVERLAY (3,4)

c3. *landparcels with high erosion* are the ones in which the attribute *erosion* has value above 70%.

DEFINE V_LAYER 6 landparcel_with_high_erosion
START_V_TIME, 1997
START_T_TIME, 1997
AS COMPUTE ATTR
(5, high_erosion = class_erosion > 70%)

c4. By virtually overlaying the layer *landparcels_with_high_erosion* with the layer *waterpipes*, we can extract the water pipes existing in high erosion landparcels; the constraint ensures that the selected water pipes are INSIDE landsparcels.

DEFINE OBJECT CLASS
pipes_in_landparcels_with_high_erosion
GEOMETRIC TYPE LINE
ON LAYERS 2,6
CONSTRAINT pipes_in_landparcels_with_high_erosion

DEFINE CONSTRAINT
pipes_in_landparcels_with_high_erosion
ON waterpipes
AMONG waterpipes, landparcel_with_high_erosion
AS ($\exists$ x $\in$ waterpipes) $\wedge$ ($\exists$
landparcel_with_high_erosion)
$\wedge$ x INSIDE y $\wedge$ (1997 – START_V_TIME(x)) >10)

6 Conclusions

The logical database design phase for spatiotemporal applications is discussed. The basic components of the spatiotemporal environment and the interconnections among them are shown. The standard logical data model, namely the relational, lacks some key features necessary for handling geographic information varying over time. An extension of it –the SpatioTemporal Relational Model (STRM)– providing a small set of representation constructs (temporal relations, layers, virtual layers, objects and constraints) is presented. The proposed language can be used to express definitions, user views and queries as well as constraints of the database. An example, coming from a real time application, shows the power of modeling and ease of use of our proposal.

We are currently testing the applicability of the model into a large-scale application [1], which will hopefully give us very useful experience and deeper understanding of combined space and time.

This is the first attempt to integrate time and space into the relational model. More input and extensions from the temporal community is needed. There are several other issues, which must be studied for a complete methodology for the development of spatiotemporal systems, such as the translation of the SpatioTemporal ER [15] and the GeoER [8] to the STRM following formal rules, and algorithms and tools for the transition of the STRM to various commercial systems.

Logical data modeling and STRM is only a part of spatiotemporal database design, which in turn is only a part of spatiotemporal information systems development. We see STRM as the foundation of a future suite Compute-

Aided Software Engineering (CASE) tool to handle spatial information oven time.

Additionally, we would like to view the STRM as contributing not only to the development of spatiotemporal applications but also to the development of GIS software dealing with temporal issues as it constitutes a guide of features required from the end-user of the system.

Acknowledgements

This research was supported in part by the Danish Technical Research Council through grant 9700780, the Danish Natural Science Research Council through grant 9400911, and the CHOROCHRONOS project, funded by the European Commission DG XII Science, Research and Development, as a Network Activity of the Training and Mobility of Researchers Program, contract no. FMRX-CT96-0056.

The authors wish to thank Christian S. Jensen for the extended discussions on the issue of time.

References

[1] Cadastral, 1997. *Definition of a standard for the exchange of digital cadastral data*. Research project of the NTUA funded by CADASTRE SA, Greece.

[2] Delis, V., Hadzilacos, Th., and Tryfona, N. 1995. *An Introduction to Layer Algebra*. Proc. of the 6th International Symposium on Spatial Data Handling, Edinbrough, UK, Taylor and Francis (1995).

[3] Egenhofer. M., and Herring, J., 1990. A Mathematical Framework for the Definition of Topological Relationships. Proceedings of the 4th International Symposium on Spatial Data Handling, Zurizh, Switzerland, pp. 803-813.

[4] Frank, A., Campari, I., and Fromentini, U., (eds.) 1992. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. LNCS (639). Springer-Verlag.

[5] Guetting, R., and Schneider, M., 1995. Realm-Based Spatial Data Types: The Rose Algebra. VLDB Journal 4.

[6] Hadzilacos, Th., and Tryfona, N., 1992. *A Model for Expressing Topological Constraints in Geographic Databases.* In LNCS (639). Springer-Verlag.

[7] Hadzilacos, Th., and Tryfona, N., 1996. *Logical Data Modeling for Geographic Applications*. International Journal of Geographic Information Systems 11,1.

[8] Hadzilacos, Th., and Tryfona, N., 1997. An *Extended the Entity-Relational Model for Geographic Applications*, ACM SIGMOD Record 26(3).

[9] Jensen, C. S., Clifford, J., Elmasri, R., Gadia, S.K., Hayes, P., and Jajodia, S. (eds). 1994 *A Glossary of Temporal Database Concepts*. ACM SIGMOD Record, 23(1):52-64.

[10] Jensen, C. S., and Snodgrass, R.T., 1996. *Semantics of Time-Varying Applications*. Information Systems, 21(4): 311-352.

[11] Lorentzos, N.A., and Manolopoulos, Y., 1995. *Functional Requirements for Historical and Interval Extensions to the Relational Model*. Data and Knowledge Engineering 17, 59-86.

[12] Scholl M., and Voisard, A., 1989. *Thematic Map Modeling*. SSD'89, Springer-Verlag, LNCS 409.

[13] Snodgrass, R.T., 1992. Temporal Databases. Invited paper. In LNCS (639). Springer-Verlag.

[14] Tryfona, N., 1998. Modeling Phenomena in Spatiotemporal Databases: Desiderata and Solutions. (under submission).

[15] Tryfona, N., and Jensen, C. S., 1997. On Conceptual Data Modeling for Spatio-Temporal Applications. Working Paper.

[16] Worboys, M., 1994. *A Unified Model for Spatial and Temporal Information*. The Computer Journal, 37,1.

[17] Ullman, J.D., 1988. Principles of Database and Knpwledge-Based Systems. Vol. 1 (Rockville, Brussels, Belgium)

[18] UtilNets, [1994-1997]. Work Program of Brite-Euram Project 7120, DG XII, European Union, Brussels, Belgium.

Appendix - The annotated syntax of STRM

DEFINE RELATION rel_name
($attr\_name_1$, $Domain_1$, <KEY>),…,
($attr\_name_k$, $Domain_k$, <KEY>)
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>

"rel_name" is an identifying alphanumeric string, i.e., different from any other used in the same syntactic position. *It is used as the name of the relation being defined.*

"($attr\_name_1$, $Domain_1$, <KEY>),…,($attr\_name_k$, $Domain_k$, <KEY>)" is a list of triplets of two strings and an optional keyword. "$attr\_name_1$,…, $attr\_name_k$" *are the names of the attributes of the relation*. "$Domain_1$,…,$Domain_k$" *are the domains of the* 1^{st},…,k^{th} *attributes of the relation respectively*. "KEY" *denotes that the attribute is part of the key of the relation.*

"[start_time, end_time]" is an optional definition of time period. *Its existence shows the time interval for the valid time and transaction time.*

DEFINE LAYER n <layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
ATTR ($attr\_name_1^n$, $Domain_1^n$, <UNIQUE>),…,
($attr\_name_{mn}^n$, $Domain_{mn}^n$, <UNIQUE>)
GEOMETRIC TYPE Geometric_type
<POSITIONING Coordinate_system>
<CONSTRAINT spatial-temporal_condition>

"n" is an identifying integer. *It is used as the layer identifier.*

"layer_name" is an optional identifying string. *It is an additional alphanumeric layer identifier.*

"[start_time, end_time]" is an optional definition of time period. *Its existence shows the time interval for the valid time and transaction time.*

"attr_name$_i^n$", for 1≤i≤mn, is an identifying string. *It is the name of* the *ith attribute of layer n.*

"Domain$_i^n$", for 1≤i≤mn, is a domain of values. *It is the domain of the ith attribute of layer n. mn is the number of attributes of layer n.*

"UNIQUE" is an optional keyword. *It denotes that not two different geometric figures can agree on that attribute.*

"Geometric_type" is one of "Point", "Line", "Region", or combination thereof. *It indicates the dimensionality of geometric figures for this layer.*

"Coordinate_system" is an optional string. *It is an identifier for the coordinate system used for this layer.*

"spatial_condition" is a first order prepositional sentence whose predicate symbols are the elementary topological predicates r0,…,r15 (e.g., r15 means "partially overlaps", see [3] [6], cardinal relationships, such as NorthEast, and/or metric relationships. *The "CONSTRAINT" clause is used to define constraints among geometric figures. Layer n contains only figures that satisfy the "CONSTRAINT" predicate.*

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS COMPUTE ATTR
(m, new_attr_name$_1^n$= f_1 (attr_name$_1^m$,…,attr_name$_{jm}^m$)
,…,
m, new_attr_name$_k^n$= f_k (attr_name$_1^m$,…,attr_name$_{jm}^m$))

"n" is an identifying integer. *It is used as the virtual layer identifier.*

"v_layer_name" is an optional identifying string. *It is an additional alphanumeric virtual layer identifier.*

"[start_time, end_time]" is an optional definition of time period. *Its existence shows the time interval for the valid time and transaction time.*

"m" is a layer identifying expression (see below).

"new_attr_name$_i^n$", for 1≤i≤k, is an identifying string. *It is the name of the ith derivable attribute of layer n.*

"f_i (attr_name$_1^m$,…attr_name$_{jm}^m$)", for 1≤i≤k is an arbitrary function (mathematic, string or geometric). *It is used for the computation of the ith new attribute.*

"jm" is the number of attributes of layer m, where m is a layer identifying expression.

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS COMPUTE SPATIAL
(m, new_attr_name^n, f(attr_name$_1^m$,…,attr_name$_{jm}^m$))

"n" is an identifying integer. *It is used as the virtual layer identifier.*

"v_layer_name" is an optional identifying string. *It is an additional alphanumeric virtual layer identifier.*

"[start_time, end_time]" is an optional definition of time period. *Its existence shows the time interval for the valid time and transaction time.*

"m" is a layer identifying expression (see below).

"new_attr_name^n", is an identifying string. *It is the name of the new boolean-valued attribute of layer n, set to TRUE at the interior of the geometric figures defined by f.*

"f(attr_name$_1^m$,…attr_name$_{jm}^m$)", is any geometric operation whose range is a geometric data type. *It is used for the transformation of layer figures. It also denotes the name of the (single) attribute for layer n.*

"jm" is the number of attributes of layer m, where m is a layer identifying expression (see below).

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS OVERLAY (k,m)

"n" is an identifying integer. *It is used as the virtual layer identifier.*

"v_layer_name" is an optional identifying string. *It is an additional alphanumeric virtual layer identifier.*

"[start_time, end_time]" is an optional definition of time period. *Its existence shows the time interval for the valid time and transaction time.*

"k,m" are layer identifying expression (see below).

DEFINE V_LAYER n <v_layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
AS RECLASS OF (m, attr_name$_1^m$,…,attr_name$_{jm}^m$)

"n" is an identifying integer. *It is used as the virtual layer identifier.*

"v_layer_name" is an optional identifying string. *It is an additional alphanumeric virtual layer identifier.*

"[start_time, end_time]" is an optional definition of time period. *Its existence shows the time interval for the valid time and transaction time.*

"m" is a layer identifying expression (see below).

"attr_name$_i^n$", for 1≤i≤jm, are name of (not necessarily all) attributes of layer m. *Reclassification is being done on the basis of these attributes.*

Recursive definition of layer identifying expression

Every layer or virtual layer identifier is a layer identifying expression

If m, n are layer identifying expressions, then so are:

i. COMPUTE ATTR
(m, new_attr_name$_1^n$=f$_1$ (attr_name$_1^m$,…,attr_name$_{jm}^m$)
,…,
m, new_attr_name$_k^n$=f$_k$(attr_name$_1^m$,…,attr_name$_{jm}^m$))

ii. COMPUTE SPATIAL
(m, new_attr_name^n, f(attr_name$_1^m$,…,attr_name$_{jm}^m$))

iii. OVERLAY (k,m)

iv. RECLASS OF (m, attr_name$_1^m$,…,attr_name$_{jm}^m$)
DEFINE OBJECT CLASS obj_class_name
<GEOMETRIC TYPE Geometric_type>
<SUBTYPE OF sup_obj_class>
<ON LAYERS layer_id$_1$,…,layer_id$_k$>
<WITH ATTRIBUTES attr_name$_1$,…,attr_name$_m$>
<CONSTRAINT constraint_name>

Either one of the "GEOMETRIC TYPE" and the "SUBTYPE OF" clauses may be missing, but not both. When the "SUBTYPE OF" clause is present, then the "ON LAYERS" clause may be missing.

"obj_class_name" is an identifying alphanumeric string. *It is used as the name of the object class being defined.*

"Geometric_type" is one of "Point", "Line", "Region", or combination thereof. *It indicates the geometric type of the object class.*

"sup_obj_class" is the name of an already defined object class. *It indicates that the geometric type of the object class being defined is the same as a previously defined class. The new class is a subclass of the old.*

"layer_id$_i$", for 1≤i≤k, is a layer identifying expression. *It denotes layers from which the object class is constructed.*

"attr_name$_i$", for 1≤i≤m, are alphanumeric strings.*They are the names of attributes of layers mentioned in the previous clause.*

"composite_constraint_name" is the name of a boolean combination of previously defined constraints.
The object class defined by this declaration is that the subset of the union of the geometric figures of the layers mentioned in the "ON CLAUSE" which satisfies the constraint, with attention restricted to the attributes mentioned in the "WITH ATTRIBUTES" clause.

DEFINE CONSTRAINT constraint_name
ON {object_class_name | layer_id}
AMONG {obj_class$_1$,…,obj_class$_k$ |
layer_id$_1$,…,layer_id$_k$}
AS spatiotemporal_condition_specification

"constraint_name" is an identifying string. "object_class_name" and "obj_class$_i$", for 1≤i≤k are object class identifiers. "object_class_name" is one of the "obj_class$_i$" 's. "object_class_name" *denotes the set from which the objects satisfying the constraint will be selected*. "obj_class$_i$" 's *are the objects which must be overlaid in order to perform the test defined by the constraint.*

"layer_id" and "layer_idi", for 1≤i≤k are layer identifying expressions. "layer_id" is one of "layer_id$_i$" 's. "layer_id" *denotes the set from which the entities satisfying the constraint will be selected*. "layer_id$_i$" 's *are the layers which must be overlaid in order to perform the test defined by the constraint.*

A "spatiotemporal_condition_specification" [14] is a first order prepositional sentence whose free variable corresponds to the object class or layer mentioned in the "ON" clause and whose predicate symbols are simple selection predicates, elementary spatial predicates (metric or cardinal), elementary topological predicates [6], and elementary temporal predicates. The constraint selects those objects from the object_class_name or layer_id respectively at the "ON" clause which satisfy the "AS" predicate after an overlaying operation has been performed among all object classes or layers (respectively) in the "AMONG" clause.