A Temporal Extension of a Spatial Relational Model

Ágnes B. Novák
Institute of Mathematics and Computational Sciences
Budapest Polytechnic
Népszínház u. 8, H-1081 Budapest
Hungary
b_novak@zeus.banki.hu

Zoltán Ludányi
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics
Műegyetem rkp. 3-9. H-1111 Budapest
Hungary
quad@dpg.hu

Márton Bérces
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics
Műegyetem rkp. 3-9. H-1111 Budapest
Hungary
bm1@freemail.hu

Zsolt Tuza
Hungarian Academy of Sciences and University of Veszprém
Kende u. 13-17, Budapest / Egyetem u. 10, Veszprém
Hungary
tuza@sztaki.hu

Abstract – **In this paper a spatiotemporal relational model is given. This model is based on the PLA spatial relational model introduced by the US Bureau of the Census, especially to represent topological properties of maps. The temporal extension of this model improves the original model to reflect the changes of spatial data in time.**

I. INTRODUCTION

We first describe the background of problems and results presented here, and then introduce the notation used in the paper.

A. Motivation

The study of spatiotemporal data, their representation and manipulation is one of the major issues in database theory and practice. An important task in this area is the modification of the already existing spatial models in order to make them able to represent objects changing in time.

The importance of these types of extension lies in the fact that usually there are already well-elaborated algorithms for the original models, and therefore there is a hope that these algorithms or their slightly modified versions can be used also for the corresponding extended spatiotemporal model. In this paper we extend a relational spatial model into a spatiotemporal model.

The rest of the paper is organized as follows. The next subsection, Notation is followed by a brief review of the PLA-Model and the description of the base of its temporal extension, in Section II. Since the original model has four relations, in Section III we give algorithms to derive the analogous temporal relations. The resulting four spatiotemporal relations form the complete model, that we call PLAT. Finally, in Section IV we conclude with a discussion on further applications of the present model and some related questions for future research.

B. Notation

Throughout the paper we apply the simplest standard notation. A relational schema is denoted by Name_of_relation(Atribute_1, Attribute_2, ..., Attribute_n).

If the relation schema is given as $R(A_1, A_2, ..., A_n)$, then a corresponding relation instance is denoted by r. An attribute can be referred to simply by its name, like A_n, or, if necessary, $R.A_n$.

Due to space limitation, algorithms are described as short as possible. For that purpose, in the pseudocodes of the algorithms we use relational algebra, which provides a widely known, brief and implementation-independent tool. In this way, the symbols Π, σ, and $\triangleright\triangleleft$ stand for the relational algebraic projection, selection, and natural join, respectively.

Since a relation instance is an *unordered* set of tuples, it causes the problem that it is difficult to describe loops of an algorithm. However, in each real system there are tools through which we can fetch rows one after the other, also getting the possibility to arrange these tuples in an array, if it is needed. We enter into neither these nor other technical details, but give an informal (and still correct) description of the methods. For that purpose, we use some simple functions and procedures; their brief descriptions are given below.

Max(r, A): this function gives the maximum value of attribute A in the relation instance r.

Number_of_rows(r): this function gives the number of rows in a given relation instance r.

Insert_row(<a_1,a_2,.., a_n>, r): this procedure inserts the row given in the first argument into the relation instance r. Of course, n must be equal to the arity of R. In the loops of the algorithms, some variables will be specified in a way that is informal but still sufficiently precise for technical realization.

II.DATA MODELS

In this section we first describe the original PLA-Model (Section *A*), and then the base of its extension, PLAT (Section *B*) at the logical level of data abstraction. Because of space limitation, we restrict ourselves only to the very necessary informal introduction. For the interested reader we give references to some of the most important works in connection with these models.

A. The PLA-Model

Under the word "spatial" we usually mean that the stored object has some well-defined geometrical extent, represented in the model in some way. It is not completely true for the PLA-Model, however, as it belongs to the spatial models in a generalized sense. The aim of PLA is the representation of the *topology* of a "typical" map – or, equivalently, a planar graph together with its regions (areas) – in terms of relational databases. So, the geometrical realization of the graph is not stored explicitly, and is not at all unique either.

The PLA-Model was introduced by the US Bureau of the Census [1]. A more detailed explanation and also other spatial models can be found in [2]. Here we recall some parts of a review of the PLA-Model in [3]. We omit the set of conditions, which restricts the type of representable graphs to the 2-connected planar graphs. Detailed discussions of the PLA-Model can be found in [1–3].

The PLA database consists of four relations, R1, R2, R3, R4, whose schemas are as follows.

In R1, the edges of a graph are stored, with respect to the schema R1(LINE, FROM, TO). Attribute LINE is the identifier of the edge, FROM stands for the "starting" point, and TO for the "end" point of that line, respectively. The edges stored in the database are undirected, but in order to obtain a unique representation of the areas (regions) there is a need for some orientation. In the sequel, to avoid ambiguity, we suppose that the points are labelled with natural numbers, and the starting point of each line is always the vertex labelled with the *smaller* number.

In the second relation R2, the relative positions of an edge and its surrounding areas are given with the schema R2(LINE, LEFT, RIGHT). The attributes correspond to the identifier of the line, the name of the area on the left side and the name of the area on the right side of that line. The left and right sides are determined in accordance with the previous relation, that is, if the line e goes from point i to point j (i.e., $i<j$ is assumed) and its bordering areas are x on the left and y on the right while we are walking along the edge e from point i to point j, then the corresponding stored tuple is $<e, x, y>$. In the original model these areas x and y must be distinct, because the stored graph is required to be 2-connected.

Relation R3 stores the border of an area (region). The schema is R3(AREA, POINT, LINE, LABEL). The first attribute is the name of the area whose neighbourhood is represented. It is assumed that every area is surrounded by an alternating cycle of points and lines. So the second attribute means the identifier of a point, the third one is the identifier of a line on the border of that area, and the last attribute, LABEL is the serial number with respect to the area named in the first attribute. The labels are consistent with the orientation; that is, the enumeration of points and lines in the cycle corresponds to the clockwise orientation, except for the infinite area, for which the orientation is counterclockwise. In each row of the relation, the point and the line are incident, and the line is the successor of the point in the corresponding orientation. Labels ensure the correct enumeration of edges and points on the border of a given area.

The fourth relation, denoted R4, has a similar role as R3, but it represents the environment of a *point*. Its schema is R4(POINT, LINE, AREA, LABEL), where POINT means the identifier of the point whose neighbourhood is stored in the relation, LINE is the identifier of a line, AREA is the identifier of an area in the neighbourhood, and LABEL is a serial number. Similarly to the previous paragraph, the orientation is clockwise (but now for all points, without any exception), and of course the line is on the boundary of the area.

It is proved in [3] that the PLA-Model is redundant from the point of view that R3 alone is equivalent to the whole database; that is, from R3 one can completely generate R1, R2 and R4 without loss of information. Based on a different approach, a closely related result was given in [4].

B. The base of temporal extension: R3(t)

As we have already mentioned, the PLA-Model can be reconstructed from R3 alone, so it seems to be a good choice to supply R3 with temporal extent. R3 stores the border of each area, that is, an alternate sequence of points and lines surrounding that area. Consequently, the time interval for the lines and points will be derived from the temporal extension of R3.

First we explain the temporal extension of R3, and afterwards in Section III we deal with the extension of the other three relations of the PLA-Model.

One possible way to store the temporal extent would be to complete R3 with two further attributes, BEGIN and END, referring the time when the area starts and ends to exist, respectively. Thus, the schema for R3' would be R3'(AREA, BEGIN, END, POINT, LINE, LABEL), where AREA, POINT, LINE, LABEL are interpreted as in Section II.*A*, and BEGIN and END stand for the starting time and the latest time of the existence of

the stored object. In case we would choose this relation schema, then in an instance for R3', in each row the values for the attributes AREA, BEGIN, END would be repeated as many times as many lines it has on its border. In order to eliminate this redundancy, the following decomposition gives us a more appropriate relational schema. The spatial information is stored in R3(AREA, POINT, LINE, LABEL), as in the original model, while the corresponding temporal data are given in a new relation R5, with the schema R5(AREA, BEGIN, END). Attribute AREA in R5 must be a candidate key, otherwise the decomposition might be lossy.

III. COMPLETE EXTENSION OF THE PLA-MODEL

In this section we extend the remaining three parts of the PLA-Model; that is, we derive the temporal extensions for R1, R2, and R4. Let us recall that in the original model, the data expressiveness of R3 is eqvivalent to the whole model, as all of R1, R2, R4 can be derived from R3. This fundamental property of R3 will be maintained in the temporal version, PLAT, too.

A. Constructing R1(t) from R3 and R5

The existence of a line and of a point can be concluded from the existence of the area containing that line and point on its border.

There is no practical need to store relation R1', the temporal extension of R1 explicitly for each possible time interval. Indeed, it is better to derive a snapshot for R1' only at a given time, t. This relation is denoted by R1(t) with the schema R1(t)(LINE, FROM, TO, BEGIN, END). All lines are stored in R1(t), for which the corresponding area, that is, the area involving the line on its border, exists at that given time t.

Algorithm 1 below shows the process for getting instance r1(t), from the instances of R3 and R5.

Algorithm 1: Constructing r1(t)

```
INPUT t, r3, r5
TEMP:= (σ_{BEGIN≤t)∧(t<END)} (r5))▷◁(r3)
A*:= first value found in TEMP.AREA

FOR EACH value in TEMP.AREA
  A:=σ_{AREA=A*}(TEMP)
  B:=∏_{BEGIN}σ_{AREA=A*}(TEMP)
  E:=∏_{END}σ_{AREA=A*}(TEMP)
  /*B and E are just single values for
  R1(t).BEGIN and R1(t).END, respectively*/

  FOR i=1 TO Max(A, LABEL)
    P1:=∏ A.POINT σ LABEL=i(TEMP)
    /*just one value for one of the endpoints of a
    line*/
    L:=∏A.LINE σ LABEL=i(TEMP)
    /*just a single value for R1(t).LINE */
    i=i+1
    IF ∏ LABELσ LABEL=i(TEMP)> Max(a, LABEL)
      THEN j=1
    P2:= ∏ A.POINT σ LABEL=j(TEMP)
    IF P1<P2
      THEN Insert_row(<L,B,E,P1,P2>, r1(t))
      /*points are labelled by natural numbers*/
  END FOR
  A*:= next value found in TEMP.AREA
END FOR
```

In Algorithm 1 above, first r5 has to be selected with respect to BEGIN≤t<END. These time intervals are closed from the left, in order to avoid the possibility of inconsistency. Let us denote the resulting relation by SR5. Taking the natural join of SR5 and r3, we get the rows involving the exsisting lines at time t. This resulting relation is denoted by TEMP. The schema for TEMP is as follows: TEMP(AREA, BEGIN, END, POINT, LINE, LABEL).

These existing lines have to be put into r1(t). In R1(t) the lines are represented by their endpoints, which can be got from the given instance of TEMP.

In the inner loop, each area A* is browsed for the endpoints of a line L, and the endpoints are inserted into r1(t). Relation A is a temporary relation for grouping the data with respect to one area. In the inner loop, the starting point and the end point of a line are decided.

B. Constructing R2(t) from R3 and R5

Relation R2 informs us about the neighbour areas of a line, having that line in common on their border. Generally, since some parts of the object may appear or disappear, the requirement of having always two different areas on the two sides of a line can no longer be kept. This would require to drop the original restrictions on the PLA-Model. Due to space limitation, these complication (for allowing also edge insertion/deletion) will be discussed in a forthcoming paper. Here we assume that only entire areas are inserted into, or deleted from the database, implicitly also involving cases where the region is constructed by an adge having its endpoints already in the graph. When deleting an area, however, the graph – originally 2-connected in PLA – could be split into several planar parts, causing difficulties especially in the recognition process of the border of the infinite area. On the other hand, at each time, area update (insertion/deletion) can be organized in a way that all those areas are updated whose bordering line(s) have been affected. Deletion can be modelled as a simple time update, if we view the stored object as the union of all its parts that existed at a time. This requirement can ensure that each line has two different areas on its sides, at every time.

Similarly to R1(t), we can define R2(t) that stores the information about the relative position of lines and areas existing at the same time. The schema for R2(t) is: R2(t)(LINE, BEGIN, END, LEFT, RIGHT). This R2(t) can be constructed by Algorithm 2 below.

The first two lines are the same as in Algorithm 1. In the FOR loop, an identifier for a line can be taken from R3, and then TEMP is selected with respect to that line resulting in A. Since each line is on the border of exactly two areas, we already get the areas, and only their relative position has to be figured out. It is based on the facts that the two points must be distinct for each line, and that the points are labelled by (natural) numbers.

Algorithm 2: Constructing r2(t)

INPUT t, r3, r5
TEMP:= ($\sigma_{(BEGIN \le t) \wedge (t < END)}$(r5)) $\triangleright\triangleleft$ (r3)
L:= first line found in TEMP.LINE
FOR each value of TEMP.LINE
  A:=$\sigma_{LINE=L}$ (TEMP)
  /*A consists of 2 rows, say x-row and y-row*/
  x:= Π_{POINT}(A) in x-row
  y:= Π_{POINT}(A) in y-row
  IF x<y THEN Left:= Π_{AREA} (A) in y-row
    Right:= Π_{AREA} (A) in x-row
  ELSE Right:= Π_{AREA} (A) in y-row
    Left := Π_{AREA} (A) in x-row
  TIME:= $\Pi_{BEGIN, END}$ $\sigma_{LINE=L}$ (A)
  /*TIME consists of 2 rows, say x-row and y-row*/
  Tbx:= Π_{BEGIN}(A) in x-row
  Tby= Π_{BEGIN}(A) in y-row
  IF Tbx<Tby THEN B:=Tbx ELSE B:=Tby
  Tex:= Π_{END} (A) in x-row
  Tey:= Π_{END}(A) in y-row
  IF Tex<Tey THEN E:=Tby ELSE E:=Tbx
  Insert_row(<L, B, E, Left, Right>, r2(t))
  L:= next line found in TEMP.LINE
END FOR

In most of the practical cases, if we already have the relation R1(t), it is not necessary to store again the time intervals for the lines in R2(t). In this situation we do not put the attributes BEGIN and END into relation R2(t), and, accordingly, do not insert the values B and E into r2(t).

C. Constructing R4(t) from R3 and R5

Recall that R4 stores the environment of a point with the schema R4(POINT, LINE, AREA, LABEL). The corresponding r4(t) is interpreted in an analogous way to r1(t) and r2(t). In order to avoid that a line is surrounded by the same area on its two sides, similarly to the previous paragraph, we assume that only entire areas are allowed for insertion into the database, and that after each insertion or deletion of an area, all the affected areas are updated. Moreover, the stored object is required to be 2-connected at any time.

As proved in [3], r3 determines r4 apart from the cyclic rotation of labels around each point. Therefore, the local choice of a starting label is arbitrary; but after that, each pair of consecutive lines has to be on the border of the same area, that can be identified using r3.

Algorithm 3: Constructing R4(t)

INPUT t, r3, r5
TEMP:= ($\sigma_{(BEGIN \le t) \wedge (t < END)}$r5)) $\triangleright\triangleleft$ (r3)
P*:= first point found in TEMP
FOR each value in TEMP.POINT
  AR1:=$\sigma_{POINT=P^*}$ (TEMP)
  A^*:= first area found in AR1
  FOR i=1 to Number_of_rows(AR1)
    AR2=$\sigma_{AREA=A^*}$ (AR1)
    /*just one row: <A*,P, L*, j> */
    A*:=Π_{AREA}(AR2)
    P:=Π_{POINT}(AR2)
    L*:=Π_{LINE}(AR2)
    j:= Π_{LABEL}(AR2)
    Insert_row(<P, L*, A*, i>, r4(t))
    AR3:= $\sigma_{AREA:=A^*}$ (TEMP)
    IF j-1=0 THEN j:= Max (ar3, LABEL)+1
    AR4:= $\sigma_{LABEL=j-1}$ (AR3) //
    /*just one row: <A*,P', L', j-1>*/
    L':= Π_{LINE}(AR4)
    A*:=Π($\sigma_{LINE=L'}$(AR1))
    /*result of selection is just one row: <A**, P, L', l**> */
  END FOR
P*:= next point found in TEMP
END FOR

In this algorithm we haven't inserted the values of BEGIN and END for representing the time. The time interval can be inserted for example before or after the line AR4:= $\sigma_{LABEL=j-1}$ (AR3), marked by // above. This can be done in a similar manner as it was for r2(t). The existence of the environment of a point, however, cannot be decided based on the existence of an area. So there are several choices for interpreting the time interval for R4(t). From the point of view that this relation stands primarily for points, the union of the time intervals for a given point is probably the most reasonable solution, but there can be some applications where the intersection of the time intervals would fit better to the nature of the problem.

IV. CONCLUSIONS

In this paper we introduced a new extension of the PLA-Model, that also involves time. The new model, PLAT has been developed on the basis of only one relation in the original PLA-Model, namely R3 that stores the boundaries of the areas.

The time extensions of the other three relations are constructed by Algorithms 1, 2, and 3.

As we have mentioned, the general case is restricted here to the one in which only entire areas are inserted or deleted. It would be useful, however, to answer questions arising when one allows some further modifications in the PLA or PLAT model; e.g., what happens if also single lines can be inserted into the object, as it frequently occurs in real-life applications.

Also, the following problem would be interesting to solve: consider a given planar embedding, how can it be handled if drawing a region inside another region is allowed; that is, when an "island" is "cut out" from a region?

Though any expert in the field can easily decide whether a given model is a spatiotemporal one, a standardized model for spatiotemporal data is not available so far. Instead, there are several different models, some of them are well-fit at a certain type of problem, while some others are more appropriate for other types of problems. Then, of course, it can be the case that there is a need to translate one model into another, and vice versa.

This need for interoperability between the models becomes clearer if we think of the many already written specialized programs, that usually can manipulate on some predefined type of models only.

Motivated by these problems, we provide some algorithms transferring PLAT into other spatio-temporal data models and vice versa, in the forth-coming paper [7].

V. REFERENCES

[1] J. P. Corbett, "Topological Principles in Cartography, Technical paper 48," *US Bureau of the Census*, Washington DC, 1979.

[2] R. Laurini and D. Thomson, *Fundamentals of Spatial Information Systems*, APIC Series Vol. 37, Academic Press, 1992.

[3] Á. B. Novák and Zs. Tuza, "Reconstruction Graphs and Testing Their Properties in a Relational Spatial Database," *Computers and Mathematics with Aplications*, vol. 43, no. 10–11, 2002, pp. 1391–1406.

[4] B. Kuijpers, J. Paredaens and J. Van den Bussche, "Lossless Representation of Topological Spatial Data," in *Advances in Spatial Databases, 4th International Symposium*, Portland, Lecture Notes in Computer Science Vol. 951, Springer-Verlag, 1995, pp. 1–13.

[5] P. Revesz, *Introduction to Constraint Databases*, Springer-Verlag, New York, 2002.

[6] J. Chomicki and P. Revesz, "Constraint-based Interoperability of Spatiotemporal Databases," *Geoinformatica*, vol. 3, no. 3, 1999, pp. 211–243.

[7] Á. B. Novák, M. Bérces, Z. Ludányi and Zs. Tuza, "On the Interoperability Problems of the Spatio-temporal Relational Model, PLAT," submitted.