Including Spatial/Temporal Objects in ORM

Gerhard Skagestein[1] and Håvard Tveite[2]

[1] University of Oslo, Department of Informatics
[2] Norwegian University of Life Sciences

Abstract. We suggest to include spatial entities and corresponding spatial values as first class citizens in fact oriented models using ORM. A spatial entity is just a piece of some space. A spatial value is seen as a possibly infinite set of positions in some coordinate system. This makes the conceptual model very independent of the implementation platform. On this basis, we suggest a notation for spatial values. Further, we investigate some rather useful spatial constraints; the no overlap constraint and the no overlap exclusion constraint, which turn out to be spesializations of the uniqueness and the exclusion constraints. Finally, we discuss some possible implementation platforms.

1 Introduction

The idea to be discussed here is to introduce spatial entities and values as first order citizens in ORM, which hopefully will make it easier to model spatial concepts, such as countries, rivers, buildings and mechanical constructions.

Fig. 1a depicts a traditional model for a time period. The reader of this model has to implicitly assume that not only the startday and the endday belong to the time period, but also all the days in between. The same holds for the application program that handles this time period. In Fig. 1b, we rather see the time period as a spatial entity represented by a spatial value consisting of an infinite set of time positions on the time line. The notation for the value will be explained later. What we have done here, is to exploit to its full extent one

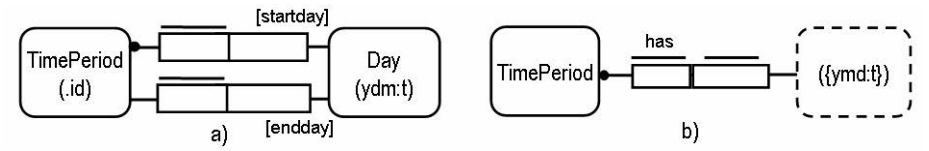

Fig. 1. Introducing a spatial value as a first order citizen

of the fundamental principles of ORM, namely the principle of separating the model from its representation. The model in Fig. 1b may be implemented in many different ways, but the model will stay the same, independent of how we choose to represent the spatial value.

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 477–486, 2013.

This distinction between the conceptual and the implementation level will get even more important for multidimensional spatial values, where a large variety of representations may be found in the literature, see for example [19, Chapter 10] (raster and vector), [16] (time) and [3] (indeterminate boundaries).

The paper is organized as follows: Section 2 gives an overview of related work. Extensions to ORM to cover spatial facts are described in Sect. 3 to 5. In Sect. 6, we look at some of the rather challenging implementation issues. Sect. 7 concludes the paper with a discussion and some suggestions for further work.

2 Related Work

There are many proposals for including spatial values as first class citizens in modeling languages. For example, Tryfona and Hadzilacos [25] have included spatial values in the relational model, whereas Shekhar et al. [23] have proposed an enhanced entity relationship model with spatial values. A rather thorough work along the same lines is the MADS data model, described in the book by Parent, Spaccapietra and Zimnyi [18]. The introduction of Abstract Data Types (ADTs) in relational databases (see for example [9]) and object-oriented models (see [4]) made it possible to define also spatial values. A comprehensive overview of the theoretical foundation of spatial modeling can be found in the book by Molenaar [17]. Halpin has discussed the inclusion of time in fact-oriented models [13,14], but mentions only briefly period seen as an atomic, not decomposable spatial value. To our knowledge, a thorough treatment of extending ORM to also handle spatial entities and values does not yet exist. A taxonomy of spatial data integrity constraints has been published by Cockcroft [5].

Most discussions are based on the point set theory and the point set topology approach [11], where a spatial value is seen as an infinite set of points. Based on the point set theory, it is possible to define topological predicates in a consistent way. Well known is the 9-intersection model [6,7], where the topological relations between two spatial objects A and B are defined in terms of the intersections of A's interior, boundary an exterior with B's interior, boundary and exterior. Schneider and Behr [22] have extended the domain of spatial values from simple points, curves and regions (surfaces) to complex points, curves and regions (surfaces). Complex values are finite sets of the simple ones, and may have holes. Schneider and Behr have further used the 9-intersection model to investigate all possible combinations of complex points, curves and regions. However, their discussions handle only up to two-dimensional spatial values. For the three-dimensional case there are several publications [2,26]. In for example [10], the authors describe a spatial/temporal datatype, the simple abstract model and the more complicated discrete model that is needed for an implementation.

3 Concepts and Notation

In ORM, we make a clear distinction between the entities found in the real world and the representations – i.e. the values. A spatial entity is a real world entity

that describes a location in space. Its representation is a spatial value. Now we may establish a fact saying that Something in the real world is occupying that location, see Fig. 2.

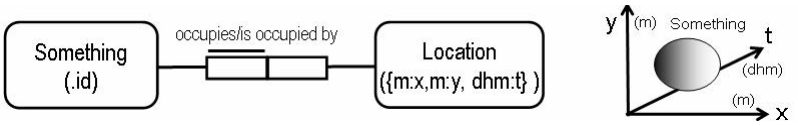

Fig. 2. Notation for spatial entities and values

In the point set theory approach [11], a spatial value is seen as a usually infinite set of points. We adopt this view here, with the modification that we follow the International Standards Organization (ISO) [15] and the Object Management Group (OMG) [12] and instead of points are using so called *direct positions*, each referenced by a coordinate tuple – i.e. a value – within a coordinate system. A direct position, then, is a position in space where something happens to be — like the middle point of a line. Thus we can avoid paradoxes like the question of where the middle point goes when a line is cut in two halves [24].

A coordinate in n-dimensional space is denoted as an n-tuple, for example x_i, y_i, t_i where x, y and t are the axes for a 3-dimensional space. The units along the axes may be included as usual in ORM, for example m:x, m:y, dhm:t where m stands for meter and dhm for day-hour-minute. Then we propose to use set brackets { } to denote the set of coordinates and hence the spatial value type, i.e. {m:x, m:y, dhm:t}.

Note that the number of elements in the coordinate tuple is determined by the dimensionality of the space at hand, i.e. the number of axes in the coordinate system. It does not say anything about the dimensionality of the spatial entity, which may be equal to or smaller than the dimensionality of the space.

4 Topological Relationships

Because we see a spatial value as a mathematical set of direct positions, all the mathematical operators of set theory stand at our disposal, and combined with first order predicate logic, this gives us a rather strong language for handling spatial values. Here, we will concentrate on the interesting topic of topology.

Informally, a topological relationship (like for example being inside or adjacent) is a relationship that does not change even if the space itself is subject to topological transformations like scaling, rotation and translation — see for example [21], page 81. Topology and topological relationships are of great interest, especially to the cartography and the GIS community, and there exists an abundant amount of written material on the matter [2,6,7,17,22]. By using set operators combined with Boolean operators, it is possible to define topological relationships like disjoint, overlap, meet and so on.

The most precise definitions are based on the so called three object part system (see for example [21], page 87), where we distinguish between the *interior* A°, the *boundary* ∂A and the *exterior* A⁻of a spatial value. In [22], Schneider and Behr give precise definitions of the interior, boundary and exterior of a possibly complex (i.e. non-contiguous) spatial value. Informally, we can say that a zero-dimensional spatial value (a Point swarm) has no boundary, the boundary of a one-dimensional value (a set of lines) is the end points of the lines, the boundary of a two-dimensional value (a set of regions) is the bounding lines — also around any holes, and the boundary of a three-dimensional value (a set of solids) is the bounding surfaces. Schneider an Behr further investigate all possible combinations of the interior, boundary and exterior between two zero-, one- or two-dimensional spatial values A and B. Then, by sensible grouping they arrive at the eight predicates *disjoint*, *meet*, *inside*, *contains*, *coveredBy*, *covers*, *equal* and *overlap*.

Three of those are of special interest to us: A and B are *disjoint* if the parts of one value intersects at most with the exterior of the other, A and B *meet* if their interiors do not intersect, but the interior or the boundary of one value intersects the boundary of the other, and A is *equal* to B if the interiors are the same — another way of saying this is that the interior of one value has no direct positions in common with the exterior of the other, and the boundary of one value has no direct postions in common with the interior or the exterior of the other.

$$disjoint(A,B) \stackrel{\text{def}}{=} A^\circ \cap B^\circ = \emptyset \wedge A^\circ \cap \partial B = \emptyset \wedge \partial A \cap B^\circ = \emptyset \wedge \partial A \cap \partial B = \emptyset$$

$$meet(A,B) \stackrel{\text{def}}{=} A^\circ \cap B^\circ = \emptyset \wedge (A^\circ \cap \partial B \neq \emptyset \vee \partial A \cap B^\circ \neq \emptyset \vee \partial A \cap \partial B \neq \emptyset)$$

$$equal(A,B) \stackrel{\text{def}}{=} A^\circ \cap B^- = \emptyset \wedge A^- \cap B^\circ = \emptyset \wedge A^\circ \cap \partial B = \emptyset \wedge \partial A \cap B^\circ = \emptyset \wedge \partial A \cap B^- = \emptyset \wedge A^- \cap \partial B = \emptyset$$

Later, we will have good use of a *noOverlap*-predicate saying that two spatial values either are *disjoint* or *meet*. Using the framework in [22], A *noOverlap* B if the only direct positions they have in common are on a boundary:

$$noOverlap(A,B) \stackrel{\text{def}}{=} A^\circ \cap B^\circ = \emptyset \wedge A^- \cap B^\circ \neq \emptyset \wedge A^\circ \cap B^- \neq \emptyset$$

In plain language, this states that

1. the interior of one of the spatial values has no direct positions in common with the interior of the other, and
2. the interior of one of the spatial values has at least one direct position in common with the exterior of the other and vice versa. This last condition is included to rule out situations where one of the spatial values is *equal* to or *inside* the boundary of the other.

5 Spatial Constraints

5.1 Topological Constraints

All kinds of spatial topological constraints can be defined by checking the outcome of topological relationships against the empty set $\emptyset$. These constraints may be expressed by extensions to the ORM diagram language or by a textual language like FORML or CQL. Here, we will only discuss some constraints that are of particular interest, namely the ORM uniqueness and set comparison constraints when extended to spatial entities and values.

The uniqueness constraint requires a test on whether two entities or values are different. But what does "different" mean when we come to spatial values? One interpretation is *"not equal"*, i.e. that the two spatial values differ in at least one direct position. The normal uniqueness constraint on spatial values would then be: For the occurences x_i of the spatial entity type X in some fact or reference type, the following should hold:

$$Uniqueness\ constraint: \ \forall i, j(i \neq j \Rightarrow \neg(x_i\ equal\ x_j))$$

This interpretation is needed when referencing a spatial entity by a spatial value, as already used in Fig. 1.

For fact types, however, it turns out that a stronger requirement is far more useful, namely that the spatial values do not overlap at all. Using the predicate noOverlap from Sect. 4, we can now define what we choose to call the *no overlap constraint*: For the occurences x_i of the spatial entity type X in some fact type, the following should hold:

$$No\ overlap\ constraint: \ \forall i, j(i \neq j \Rightarrow x_i\ noOverlap\ x_j)$$

Since the no overlap constraint is stronger than the uniqueness constraint, we suggest drawing it by means of a fatter uniqueness constraint line, as shown in Fig. 3a and b. In Fig. 3a, the no overlap constraint does not only say that a certain region can be the region of only one country, but also that these regions do not overlap. In Fig. 3b, the long uniqueness constraint, of which a part is a no overlap constraint, says that a person can not be busy with more than one task during a certain time period. Observe that for non-spatial values, the no overlap constraint coinsides with a uniqueness constraint. This means that we in Fig. 3b could have used a long, fat line without any change in the meaning. Further, since the no overlap constraint is a specialization of a uniqueness constraint, a long uniqueness/no overlap constraint may be objectified, as shown in Fig. 3b.

Of the set comparison constraints, both the equality constraint and the subset constraint require tests on "equality" between occurences, and when it comes to spatial values, the *equal* predicate as defined in Sect. 4.1 is the obvious choice. Likewise, the usual exclusion constraint can be defined by means of the expression *not equal*.

However, similar to the uniqueness constraint, also the exclusion constraint needs a spatial companion; a no overlap exclusion constraint where the occurences should not only be *not equal*, but also satisfy the *noOverlap*-predicate.

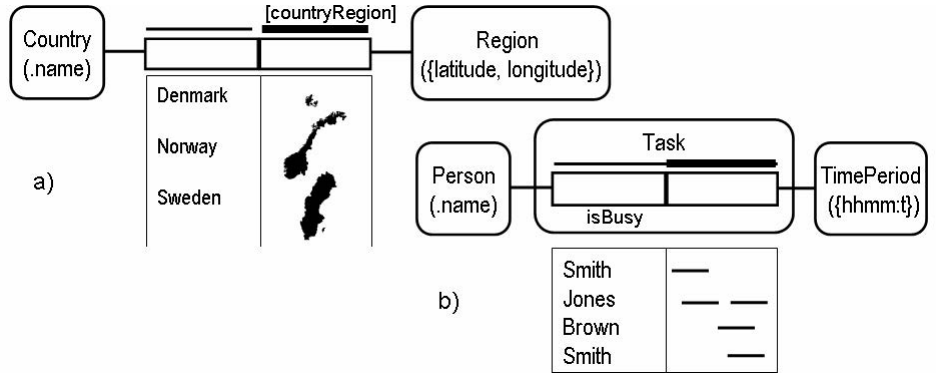

Fig. 3. The no overlap constraint

Since the no overlap exclusion constraint is stronger than the normal exclusion constraint, we suggest drawing it in the ORM-diagrams as a fatter exclusion constraint symbol. Further, since the *noOverlap* predicate coincides with a plain not equal for non-spatial values, there will be no change in meaning by letting the no overlap exclusion constraint symbol involve both roles in the fact types in Fig. 4.

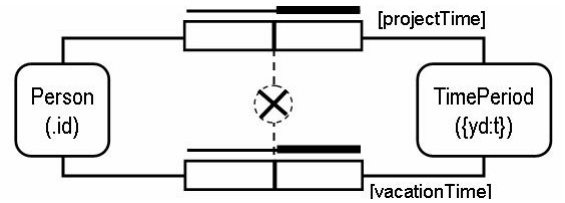

Fig. 4. The no overlap exclusion constraint

5.2 The Pattern "The whole and its Parts"

In the example in Fig. 5, we see Countries that are partitioned into smaller pieces, here called Provinces. Then each countryRegion of a Country should equal the union of the provinceRegions of the Provinces in the same country. This pattern shows up everywhere where something that occupies a space is partitioned into smaller pieces. It is for example very useful for cutting a spatial/temporal entity into slices along the t-axis.

5.3 Value Constraints

If that adds to the clarity of the model (and makes the implementation easier), we may use well known subtypes of the spatial entity, like for example

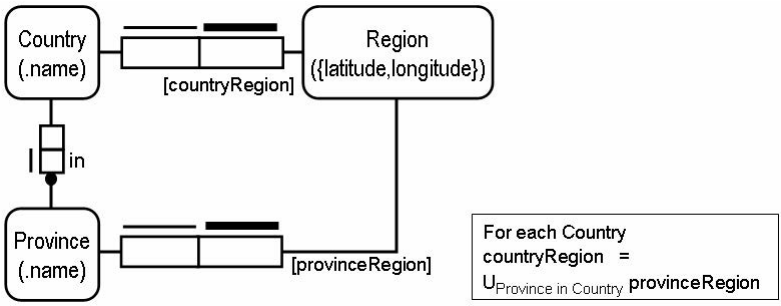

Fig. 5. The whole and its parts

MultiCurve, MultiRegion (MultiSurface), MultiPolygon and MultiPoint (Point swarm) (see [15]). For some subtypes, the set of direct positions may be written as a mathematical expression, like for example $\{x, y \mid x_1 < x < x_2 \wedge y_1 < y < y_2\}$ for a filled Polygon that has the shape of a rectangle. Such subtypes are really the result of value constraints, in that only certain direct positions within the space may be included in the value. Please note that in our approach, a Point swarm with only one element is still a Point swarm.

6 Implementation Issues

Spatial values can usually not be physically represented directly and exactly. In the literature, we find an abundant number of proposals for representations, whose primary objective is to speed up operations and save storage space. We find vector representations of network and spaghetti type, and a variety of raster representations (see for example [21], chapter 2.2). We may even see that different kinds of physical representations are used in the different roles played by the very same spatial entity.

Our modeling approach, characterized by spatial entities and spatial values defined in a very general way on a high level of abstraction, may ease the modeling effort. However, to map those values to efficient physical representations, e.g. in a database, is as challenging as ever. With a traditional ORM-model, automated tools for mapping to a relational database structure are available (like the NORMA mapping tool). An ORM-model including spatial entities and values is considerably more difficult to handle.

The major problem is that a traditional relational database (pre SQL3) simply cannot cope with non-zero dimensional values in multidimensional space[8]. If such a database is our only option, we have to replace the spatial values by a number of non-spatial values and give the application programs the responsibility to "see" the space behind those values. This is exactly what we do when we model a time period by means of the start- and end-day, like in Fig. 1a.

So, if we abandon the traditional relational database, what then? That depends on the nature of the problem at hand and the capabilities of the available

implementation platforms. We have one model, but many possibilities for implementation. This makes it challenging to create an automated tool for mapping the model to some kind of database structure. On the other hand, this is a sign of health: We are now able to model the Universe of Discourse without being influenced by concerns for the implementation.

Some possible implementation platforms are:

1. A tessellation based implementation, where the infinite sets of direct positions are replaced by finite sets of tiles. The tiles can be of equal size, as in a raster model (pixles in 2D and voxels in 3D). A Triangulated Irregular Network (TIN) is an example of a tessellation with irregular tiles. The main advantage of a tessellation based representation is that it can represent any spatial entity, although with a precision determined by the size of the tiles. The primary disadvantage is that it is relatively costly in terms of storage space and processing time. This problem may with time be alleviated by cheaper and larger computer storage and quicker processors.
   In a tessellation based implementation, boundaries are not represented explicitly – they will always run along the edges of the tessellation elements. Then it is rather convenient that our definition of the no overlap constraint does not involve the boundary.
2. The relational database language SQL/MM Spatial [1], which has been extended with spatial abstract datatypes (ADTs) for geometry (abstract), 0D (point), 1D (curve with specialisations), 2D (surface with specialisations), 2D tesselations (polyhedral surface with TIN as a specialisation), collections (of geometries, points, curves and surfaces) and topology.
3. A special purpose DBMS, like a Geographic Information System (GIS), which may be an attractive option for the many systems that handle geographic information. Most such systems have at least a built in datatype called polygon, which can be used for representing the boundary, and then inherently also the interior, of a two-dimensional spatial entity in two-dimensional space. However, those systems are not really set up to handle spatial values of the kind encountered outside the GIS-world.
   If the model follows the pattern "The whole and its parts" from Sect. 5.2, as is often the case for maps, we may use the so called network model (see for example [21] paragraph 2.3.2).
4. A Constraint database system, see [20] and [21, chapter 4]. The general idea is to represent an infinite set of direct positions by a set of constraints that cuts the spatial value out of the space, and then to store only the constraints. Constraint databases seem to be a promising technology for databases containing spatial values.
5. An object-oriented system, where all the difficulties of representing the spatial values are hidden inside classes and objects; from the outside we can only see the interface as a collection of methods to set and get the spatial values, to do spatial operations and to check constraints. This means that the implementor has full freedom in choosing the internal representation of a spatial value, as long as the methods are working as expected. Further, all

objects have their own internal object identifier (OID) that may be used to refer to the object from elsewhere.
However, in the object-oriented world there may be no police force that is enforcing the rules defined by the constraints. The philosophy is rather that every object is responsible for behaving according to the rules. So, for example, to enforce the no overlap constraint, a new (or changed) spatial entity has to call some method like checkNoOverlap(anotherObject) for all the other objects. It may be a better option to maintain a spatial object for the unused space (which is the exterior of the union of the already existing spatial entities) and let a new (or changed) object check for being inside that unused space.

7 Discussion and Further Work

In this paper, we have proposed to extend ORM with spatial entities and values on a rather high abstraction level, based on values defined as possibly infinite sets of direct positions. Spatial calculations can then be done in point set algebra. Many other modeling language approaches work on a more implementation oriented level where you see lines and polygons, nodes and edges.

This high level, however, makes the mapping from the model to a database structure rather challenging. Many spatial constraints should not be mapped into constraints on the implementation level, but rather be used to find "smart" representations where the constraints are inherently satisfied. Other constraints may be mapped into a topological structure on top of the geometric one. Much work is needed to establish the mapping rules, and to investigate whether parts of this process can be automated. It may very well be that some intermediate schema-transformations requiring manual intervention will be necessary. Advances in storage- and database technology may in the future make the transition to implementations more straightforward.

More work has to be done on the notation for specifying value-constraints on the spatial entities so that more complicated cases can be covered. And it would probably be of value to work out the constraints that correspond to the primitive geometries and the complexes found in the ISO 19100 series of standards.

Integrating spatial values within different spatial domains has not been addressed in this paper. This must be solved using transformations, such as projections from a higher dimensional space to a lower, or coordinate transformations between different coordinate systems.

Even though the spatial constraints mentioned in this paper cover some of the cases found in the real world, more types of constraints are needed. It would also be nice to have some general applicable modeling patterns, beyond the pattern "the whole and its parts".

References

1. Ashworth, M.: Information Technology, Database Languages, SQL Multimedia, and Application Packages, Part 3: Spatial. ISO/IEC 13249-3 (1999)

2. Borrmann, A., Rank, E.: Topological Operators in a 3D Spatial Query Language for Building Information Models. In: Proc. of the 12th Int. Conf. on Computing in Civil and Building Engineering, ICCCBE-XII (2008)
3. Burrough, P.A., Frank, A.: Geographic Objects with Indeterminate Boundaries. Taylor and Francis (1996)
4. Catell, R.G.G.: Object Databases and Standards. In: Goble, C.A., Keane, J.A. (eds.) BNCOD 1995. LNCS, vol. 940, pp. 1–11. Springer, Heidelberg (1995)
5. Cockcroft, S.: A Taxonomy of Spatial Data Integrity Constraints. Geoinformatica 4(4), 419–433 (2000)
6. Egenhofer, M.J., Franzosa, R.D.: Point-set Topological Spatial Relations. Int. J. Geographical Information Systems 5(2), 161–174 (1991)
7. Egenhofer, M.J., Clementini, E., di Felice, P.: Topological Relations between Regions with Holes. Int. J. Geographical Information Systems 8(2), 128–142 (1994)
8. Egenhofer, M.J., et al.: Progress in Computational Methods for Representing Geographical Concepts. Int. J. of Geographical Information Science 13(8), 775–796 (1999)
9. Elmasri, R., Navathe, S.: Fundamentals of Database Systems. Benjamin Cummings, Redwood City (1994)
10. Erwig, M., et al.: Abstract and Discrete Modeling of Spatio-Temporal Data Types. In: 6th ACM Symp. on Geographic Information Systems, pp. 131–136 (1998)
11. Gaal, S.: Point Set Topology. Academic Press, New York (1964)
12. OGC. OGC Abstract Specification. OpenGIS Consortium (OGC) (1999), `http://www.opengis.org/techno/specs.htm`
13. Halpin, T., Morgan, T.: Information Modeling and Relational Databases. Morgan Kaufmann, Amsterdam (2008)
14. Halpin, T.: Temporal Modeling and ORM. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 688–698. Springer, Heidelberg (2008)
15. International Standard ISO 19107. Geographic information – Spatial schema (2003)
16. Langran, G.: Time in Geographic Information Systems. Taylor & Francis (1992)
17. Molenaar, M.: An Introduction to the Theory of Spatial Object Modelling for GIS. Taylor and Francis, London (1998)
18. Parent, C., et al.: Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach. Springer, Heidelberg (2006)
19. Peuquet, D.J.: A Conceptual Framework and Comparison of Spatial Data Models. Cartographica 21(4), 66–113 (1984)
20. Revesz, P.: Introduction to Constraint Databases. Springer, Heidelberg (2002)
21. Rigaux, P., Scholl, M., Voisard, A.: Spatial Databases with Application to GIS. Morgan Kaufmann, Academic Press, London (2002)
22. Schneider, M., Behr, T.: Topological Relationships between Complex Spatial Objects. ACM Transactions on Database Systems 31(1), 39–81 (2006)
23. Shekhar, S., Coyle, M., Goyal, B., Liu, D., Sarkar, S.: Data Models in Geographic Information Systems. Communication of the ACM 40(4) (1997)
24. Sowa, J.: Knowledge Representation. Logical, Philosophical, and Computational Foundations. Brooks/Cole Pacific Grove, CA (2000)
25. Tryfona, N., Hadzilacos, T.: Logical Data Modeling of SpatioTemporal Applications: Definitions and a Model. In: Proc. of the 1998 International Symposium on Database Engineering & Applications, IDEAS, p. 14 (1998)
26. van Oosterom, P., et al.: Integrated 3D modelling within a GIS. In: Proc. of the Workshop on Advanced Geographic Data Modelling (1994)