

# SPATIAL AND SPATIO-TEMPORAL DATA MODELS AND LANGUAGES

Markus Schneider  
Department of Computer & Information Science & Engineering  
University of Florida  
Gainesville, FL 32611, USA  
mschneid@cise.ufl.edu

## SYNONYMS

N/A

## DEFINITION

A *data model* provides a formalism consisting of a notation for describing data of interest and of a set of operations for manipulating these data. It abstracts from reality and provides a generalized view of data representing a specific and bounded scope of the real world. In the context of databases, a data model describes the organization, that is, the structure, of a database. In the context of complex objects like video, genomic, and multimedia objects, a data model describes a type system consisting of data types, operations, and predicates. Spatial and spatio-temporal data models are of this second kind. A *spatial data model* is a data model defining the properties of and operations on static objects in space. These objects are described by *spatial data types* like *point* (for example, representing the locations of cities in the U.S.), *line* (for example, describing the ramifications of the Nile Delta), and *region* (for example, depicting school districts). Operations on spatial data types include, for instance, the geometric *intersection*, *union*, and *difference* of spatial objects, the computation of the *length* of a line or the *area* of a region, the test whether two spatial objects *overlap* or *meet*, and whether one object is *north* or *southeast* of another object. A *spatio-temporal data model* is a data model representing the temporal evolution of spatial objects over time. These evolutions can be discrete, that is, they happen from time to time (for example, the change of the boundary of a land parcel) or continuous, that is, they happen permanently and smoothly (for example, the devastating trajectory of a hurricane). In the continuous case, one speaks about *moving objects* and represents them by *spatio-temporal data types* like *moving point* (for example, recording the route of a cell phone user), *moving line* (for example, representing the boundary of a tsunami), and *moving region* (for example, describing the motion of an air polluted cloud). Operations on spatio-temporal data types comprise, for instance, the spatio-temporal *intersection*, *union*, and *difference* of moving objects, the computation of the *trajectory* of a moving point as a line object, the determination of the *location* of a moving object at a particular time, the calculation of a moving object during a given set of intervals, and the test whether a moving point *enters* or *crosses* a moving region. *Spatial* and *spatio-temporal query languages* enable the user to query databases enhanced by these concepts.

## HISTORICAL BACKGROUND

The interest to store geometric data in databases began in the late seventies. Due to the increasing success of relational databases, the first approach has been to decompose a spatial object recursively into its constituent parts until they can be stored in tables. For example, this approach decomposes a polygon into its set of segments. Each segment is decomposed into a pair of points. A point is decomposed into a pair of two float numbers. Float numbers are a DBMS data type and can be stored in a table. This approach has revealed a number of fundamental drawbacks. Since all lines and polygons are decomposed into their constituent parts scattered as tuples over a relation, a spatial object is not treated as an entity or unit but only *corresponds* to a collection of tuples. Since this approach is based on standard domains and has no concept of spatial data types, it cannot provide and support any meaningful geometric operations. A more detailed discussion can be found in [7].

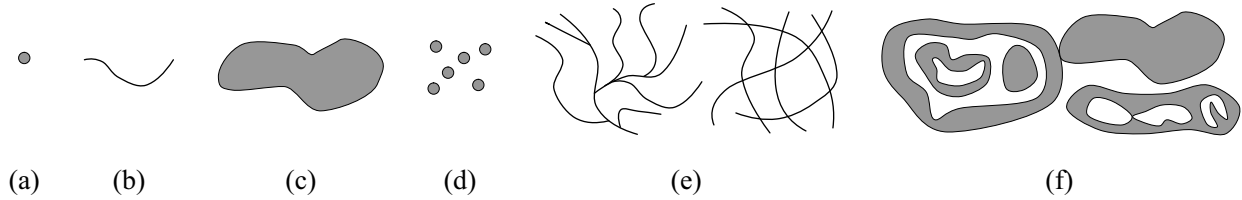


Figure 1: Examples of a simple point object (a), a simple line object (b), a simple region object (c), a complex point object (d), a complex line object (e), and a complex region object (f).

Classical research on time-varying geometric data has focused on *discrete* changes of spatial objects over time. For example, cadastral applications deal with the management of land parcels whose boundaries can change from time to time due to specific legal actions such as splitting, merging, or land consolidation. Political boundaries can suddenly disappear, as the reunification of West and East Germany shows. Different approaches have been proposed to model these discrete changes. One of them is to enhance *temporal databases* [12] with spatial data types. Each discrete change leads to a new stored snapshot with a modified spatial object in the temporal database. Another approach [15] keeps a single version of each spatial object only but annotates each of its components (for instance, a point or a segment) with a temporal element indicating the period of validity or existence of this component. Hence, discrete changes of a spatial object are registered within the object.

## SCIENTIFIC FUNDAMENTALS

Spatial data types form the basis of a large number of data models and query languages for spatial data. They are extensively leveraged by *spatial databases* [9] and embedded as attribute data types into their data models, that is, in the same way as standard data types such as *integer*, *real*, and *string*. The geometric types are designed as *abstract data types*, that is, the internal structure of a spatial object is hidden from the user, and its features can only be retrieved by (abstract) operations on this object. In this manner, they provide a high-level view of geometric data. One can distinguish the older generation of *simple* spatial data types and the newer generation of *complex* spatial data types, depending on the spatial complexity the types are able to model. In the two-dimensional space, simple spatial data types only provide simple object structures like single points, continuous lines, and simple regions (Figure 1(a)-(c)). However, from an application perspective, simple spatial data types have turned out to be inadequate abstractions for spatial applications since they are insufficient to cope with the variety and complexity of geographic reality. From a formal perspective, they are not closed under the geometric set operations *intersection*, *union*, and *difference*. This means that these operations applied to two simple spatial objects can produce a spatial object that is *not* simple. Complex spatial data types solve these problems. They provide universal and versatile spatial objects and are closed under geometric set operations. They allow objects with multiple components, region components that may have holes, and line components that may model ramified, connected geometric networks (Figure 1(d)-(f)).

As an example, in a relational setting, states, cities, and rivers are represented in the following relations:

```

states(sname: string, area: region)
cities(name: string, population: integer, location: point)
rivers(name: string, route: line)

```

Queries can then be formulated by employing operations and predicates on spatial attribute values within an extended standard database query language such as SQL, leading to *Spatial SQL* [1]. Assume that the following operations and predicates are available:

<i>area:</i>	<i>region</i>	$\rightarrow$ <i>real</i>
<i>inside:</i>	<i>point</i> $\times$ <i>region</i>	$\rightarrow$ <i>bool</i>
<i>intersection:</i>	<i>line</i> $\times$ <i>region</i>	$\rightarrow$ <i>line</i>
<i>length:</i>	<i>line</i>	$\rightarrow$ <i>real</i>
<i>meet:</i>	<i>region</i> $\times$ <i>region</i>	$\rightarrow$ <i>bool</i>

The predicates *inside* and *meet* represent *topological relationships* [8] that characterize the relative position between spatial objects. The operation *length* is a numerical function computing the length of a *line* object. The operation *intersection* computes the part of a *line* object intersecting a *region* object.

One can now pose queries: What is the total population of the cities in France?

```
select sum(c.pop) as total
from   cities as c, states as s
where  c.location inside s.area and s.name = 'France'
```

Compute the part of the river Rhine that is located within Germany and determine its length.

```
select intersection(r.route, s.area) as rhine,
       length(intersection(r.route, s.area)) as len
from   rivers as r, states as s
where  r.name = 'Rhine' and s.name = 'Germany'
```

Make a list that shows for each state the number of its neighbor states, and their total area.

```
select  s.name, count(*), sum(area(t.area))
from    states as s, states as t
where   s.area meet t.area
group  by s.name
```

Spatio-temporal data types enable the user to describe the dynamic behavior of spatial objects over time. The dynamic behavior refers to the continuous change of the locations of spatial objects over time. That is, the spatial objects move, and they are therefore called *moving objects*. They are stored in special spatio-temporal databases called *moving objects databases* [6]. In the same way as spatial data types, spatio-temporal data types are also designed as abstract data types and embedded as attribute types into a DBMS data model. Spatio-temporal data types are available for moving points (type *mpoint* for short), moving lines (*mline*), and moving regions (*mregion*). In case of moving regions, one can also represent the change of their extent and shape over time. Conceptually, a moving point is a function  $f : \textit{time} \rightarrow \textit{point}$ , a moving line is a function  $f : \textit{time} \rightarrow \textit{line}$ , and a moving region is a function  $f : \textit{time} \rightarrow \textit{region}$ . For example, for a moving region this means that at each time instant an object of type *region* has to be returned. Geometrically, moving objects correspond to the three-dimensional shapes. In case of a moving point it is a three-dimensional line (Figure 2a) and in case of a moving region it is a volume (Figure 2b). One can distinguish moving objects databases that model and query the history of movement for spatio-temporal analysis [2, 5] and moving objects databases that model, predict, and query current and future movement [10, 11]. In the latter case, location updates require a balancing of update costs and imprecision [14] and introduce the feature of uncertainty [13].

As an example, consider relations describing the movements of airplanes or storms:

```
flight(id: string, from: string, to: string, route: mpoint)
weather(id: string, kind: string, area: mregion)
```

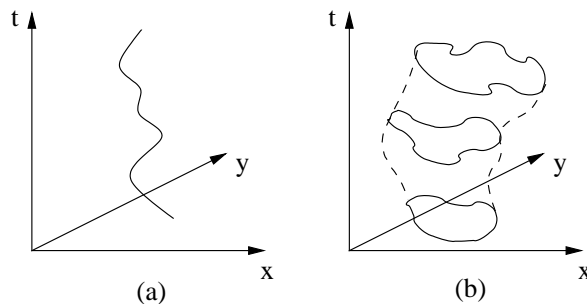


Figure 2: Examples of a moving point object (a) and a moving region object (b)

One can pose queries by employing operations and predicates on spatio-temporal attribute values within an extended standard database query language such as SQL, leading to *Spatio-Temporal Query Language (STQL)* [3]. Assume that the following operations and predicates are available:

<i>deftime</i> :	<i>mpoint</i>	→ <i>periods</i>
<i>Disjoint</i> :	<i>mpoint</i> × <i>mregion</i>	→ <i>bool</i>
<i>distance</i> :	<i>mpoint</i> × <i>mpoint</i>	→ <i>mreal</i>
<i>Inside</i> :	<i>mpoint</i> × <i>mregion</i>	→ <i>bool</i>
<i>intersection</i> :	<i>mpoint</i> × <i>mregion</i>	→ <i>mpoint</i>
<i>meet</i> :	<i>point</i> × <i>region</i>	→ <i>bool</i>
<i>min</i> :	<i>mreal</i>	→ <i>real</i>
<i>trajectory</i> :	<i>mpoint</i>	→ <i>line</i>

The function *deftime* returns the set of time intervals when a moving point is defined. The *spatio-temporal predicate* [3, 4, 6] *Disjoint* checks whether a moving point and a moving region are disjoint for some period. The function *distance* computes the distance between two moving points and is a real-valued function of time, captured here in a data type *mreal* for *moving reals*. The spatio-temporal predicate *Inside* tests whether a moving point is located inside a moving region for some period. The operation *intersection* returns the part of a moving point whenever it lies inside a moving region, which is a moving point again. The topological predicate *meet* checks whether a point object is located on the boundary of a region object. The function *min* yields the minimal value assumed over time by a moving real.

One can now pose queries: Find all flights from Frankfurt that are longer than 5000 kms.

```
select id
from flight
where from = 'FRA' and length(trajectory(route)) > 5000
```

Retrieve any pairs of airplanes, which, during their flight, came closer to each other than 500 meters.

```
select f.id, g.id
from flight as f, flight as g
where f.id <> g.id and min(distance(f.route, g.route)) < 0.5
```

At what time was flight TB691 within a snowstorm with id RS316?

```
select deftime(intersection(f.route, w.area))
from flight as f, weather as w
where f.id = 'TB691' and w.id = 'RS316'
```

Which are the planes that ran into a hurricane and had to traverse it?

```
select f.id, w.id
from flight as f, weather as w
where w.kind = 'hurricane' and
      f.route Disjoint >> meet >> Inside >> meet >> Disjoint w.area
```

The term *Disjoint >> meet >> Inside >> meet >> Disjoint* is a spatio-temporal predicate that is composed of a *temporal sequence* of the basic spatio-temporal predicates *Disjoint* and *Inside* as well as the topological predicate *meet*. The *temporal composition operator* is indicated by the symbol *>>*. The query above searches for a spatio-temporal pattern in which a plane is disjoint from a hurricane for some period, then meets the boundary of the hurricane at a time instant, is inside the hurricane for some period, meets the boundary of the hurricane again at a time instant, and is disjoint again from the hurricane for some period. The alternating sequence of topological predicates that hold for some period or for some time instant is characteristic for composite spatio-temporal predicates.

## KEY APPLICATIONS

Spatial data models containing spatial data types, operations, and predicates are a universal and general concept for representing geometric information in all kinds of spatial applications. They have found broad acceptance in spatial extension packages of commercially and publicly available database systems as well as in geographic information systems. Further, all applications in the geosciences (for example, geography, hydrology, soil sciences) as well as many applications in government and administration (for example, cadastral applications, urban planning) already benefit from them. Independent studies have shown that about 80% of all data have spatial features (like geometric attributes) or a spatial reference (like an address). Thus, it is not surprising that independent international studies have predicted that geoinformation technology will belong to the most important and promising technologies in the future, besides biotechnology and nanotechnology.

The usage of moving objects in databases and especially in geographic information systems is still in its infancy since it is a relatively new technology. But increasingly, applications like location management, GPS-equipped PDAs, phones, and vehicles, navigation systems, RFID-tag tracking, sensor networks, hurricane research, and national security show interest in moving objects databases.

## CROSS REFERENCES

SPATIAL DATA TYPES

SPATIO-TEMPORAL TRAJECTORIES

TEMPORAL DATABASES

## RECOMMENDED READING

- [1] M. J. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Trans. on Knowledge and Data Engineering*, 6(1):86–94, 1994.
- [2] M. Erwig, R.H. Güting, M. Schneider, and M. Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *GeoInformatica*, 3(3):265–291, 1999.
- [3] M. Erwig and M. Schneider. Developments in Spatio-Temporal Query Languages. In *IEEE Int. Workshop on Spatio-Temporal Data Models and Languages*, pages 441–449, 1999.
- [4] M. Erwig and M. Schneider. Spatio-Temporal Predicates. *IEEE Trans. on Knowledge and Data Engineering*, 14(4):1–42, 2002.
- [5] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A Foundation for Representing and Querying Moving Objects. *ACM Trans. on Database Systems*, 25(1):1–42, 2000.
- [6] R. H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005.
- [7] M. Schneider. *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems*, volume LNCS 1288. Springer-Verlag, 1997.
- [8] M. Schneider and T. Behr. Topological Relationships between Complex Spatial Objects. *ACM Trans. on Database Systems*, 31(1):39–81, 2006.
- [9] S. Shekar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2003.
- [10] A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Int. Conf. on Data Engineering*, pages 422–432, 1997.
- [11] A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Querying the Uncertain Position of Moving Objects. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal Databases: Research and Practice*, LNCS 1399, pages 310–337. Springer-Verlag, 1998.
- [12] A. U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.
- [13] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing Uncertainty in Moving Objects Databases. In *ACM Trans. on Database Systems*, pages 463–507, 2004.
- [14] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and Imprecision in Modeling the Position of Moving Objects. In *Int. Conf. on Data Engineering*, pages 588–596, 1998.
- [15] M.F. Worboys. A Unified Model for Spatial and Temporal Information. *The Computer Journal*, 37(1):25–34, 1994.