Literature Review

Michael Stewart

27 March 2015

Title:	A Modern Perspective of Spatio-temporal Databases
Author:	Michael Stewart
Supervisors:	Professor Wei Liu and Professor Rachel Cardell-Oliver
Degree:	Honours in Computer Science (24 point project)

Abstract

The demand for analysable spatio-temporal databases is currently rising due to the increase in large datasets, sensor data, and trackable moving objects. This literature review focuses specifically on the realm of spatio-temporal databases and explores relevant literature by breaking the databases down into the sum of their parts: spatial databases, and temporal databases. These two database types are discussed in terms of the theories behind them, query types, and existing implementations. The literature on spatio-temporal databases is then reviewed and existing NoSQL systems that support spatio-temporal data are compared. The goal of this literature review is to provide an overview of the concepts relating to spatio-temporal databases and to discuss the available software implementations in order to present a modern perspective of the technology.

1 Introduction

Techniques for visualising and analysing spatio-temporal data are currently in demand. This demand is rising with the increase of large geo-spatial datasets, sensor data, and trackable moving objects [7]. Spatio-temporal data can be stored inside spatio-temporal databases, which keep track of the location of objects over time [11]. As a result of this, spatio-temporal databases can be used to track moving objects. Spatial and temporal databases have many important applications including geographic information systems and traffic monitoring systems [17].

This review explores the theory behind, queries, and existing implementations of both spatial and temporal databases, as well as spatio-temporal databases, the combination of the two. It also briefly touches on Geographic Information Systems (GIS), one of the most popular applications of spatial databases. Finally, this paper concludes by discussing the current spatio-temporal database software implementations available, and compares each of these implementations in terms of their features and architecture.

2 Spatial Databases

A spatial database holds information relating to the location of objects in space. Güting [12] defines the term "spatial database system" as a database that offers "spatial data types (SDTs) in its data model and query language" as well one that "supports spacial data types in its implementation, providing at least spatial indexing and efficient algorithms for spacial join". Egenhofer also discussed the underlying mathematics behind relations in these databases in his paper entitled "Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases" [10]. These two papers are some of the pioneering research papers on spatial databases. Egenhofer contributed to the field by investigating spatial relations. Güting, on the other hand, contributed by expanding on Egenhofer's research and defining the term "spatial database system".

2.1 Theory

When modeling single objects in a spatial database, data is represented using three different spatial data types (SDTs). According to Güting [12], SDTs, also known as spatial algebras, provide a means to represent points, lines and regions in space. A point represents an object's location in space, irrespective of other objects. A line, on the other hand, is able to represent connections in space, such as a path from one location to another. A region can be used to represent a two-dimensional area, such as a field. Spatial databases are able to support the definition of spatial relationships between these three data types, which allow queries that can, for example, return all objects within a region. Egenhofer [10] also discusses the idea of points, lines and regions, and provides geometric interpretations of relations between each of these data types. Figure 1 shows a visual representation of a point, a line, and a region.

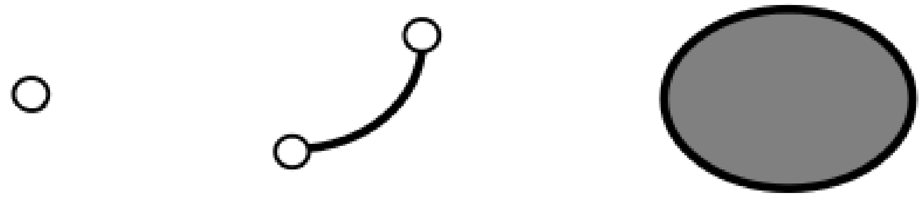


Figure 1: A visual representation of a (a) point, (b) simple line, and (c) region [10].

2.2 Querying

The design of a spatial database system allows for certain queries to be performed on the data. Güting [12] outlines a few example queries, and demonstrates the relevance of the point, line, and region system. His examples include:

"Find all cities inside Bavaria" (where Bavaria is a region)

```
cities select [center inside Bavaria]
```

" Find all big cities no more than 100kms from Hagen" (where Hagen is a point)

```
cities select[dist(center, Hagen) < 100 and pop > 500000]
```

The queries above are examples of spatial selection queries, which are used to find points, lines and regions based on spatial predicates.

Zhang et al. [18] expand on the early ideas behind spatial queries by introducing techniques to perform more complex query types, known as "window" queries and "nearest neighbour" queries. A window query retrieves all objects that lie within a given window. This is achieved by constructing a tree, known as an R*-tree (a variant of the R-tree), which clusters objects together into rectangular spaces based on proximity, and assembles them in a hierarchal structure [6]. On the other hand, a nearest neighbour query returns the data point closest to a query point, by recursively searching through the data points after they have been arranged to form an R*-tree. Figure 2 shows a visual representation of an example R-Tree, where each node has been inserted into a tree based on its location within the query windows E_1 - E_9.

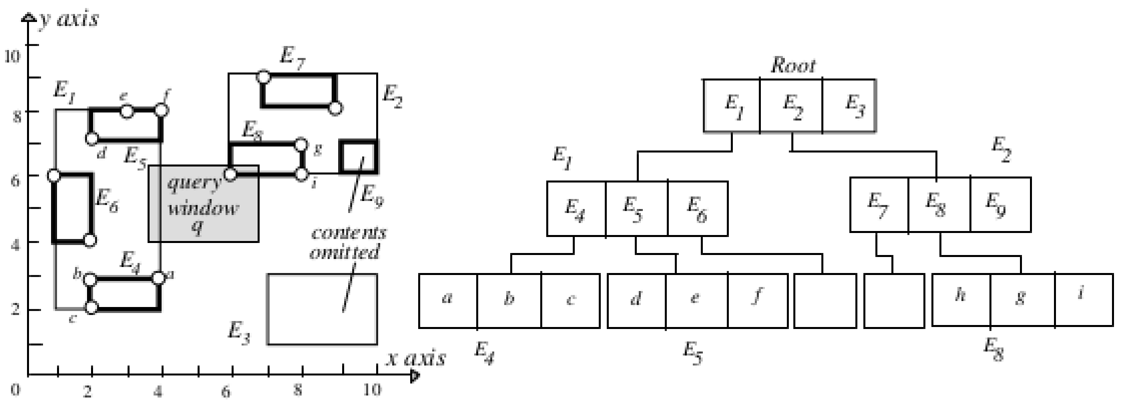


Figure 2: A visual representation of an example R-Tree [18].

2.3 Geographic Information Systems

One of the most widely used applications of spatial database systems are Geographic Information Systems (GIS), which are programs that allow users to manage spatial data [8]. GIS typically restrict data to two dimensions, and are designed to aggregate spatial data from across multiple sources. Bonham-Carter's book in 2014 [8], "Geographic information systems for geoscientists: modelling with GIS", provides a detailed overview of GIS, and is one of the more recent books that cover GIS. The concepts behind GIS are important to the topic of spatio-temporal databases as many features present in GIS will be present in the final implementation of this project. These features include the ability to unify data from a range of different sources, and visualise the data in a uniformed format such as a map.

2.4 Existing Implementations

Several modern implementations of spatial databases exist. Two of the most popular are PostGIS[1] and Neo4J[2], which both have support for geographic objects. Baas [4] explored the difference between these two systems and noted that the key difference between PostGIS and Neo4J is that PostGIS is a relational database management system (RDBMS), whereas Neo4J is a NoSQL database. NoSQL databases have flexible schemas, which allow for dynamic and ad-hoc data to be stored in the database. On the other hand, RDBMS such as PostGIS cannot handle unstructured data. Baas contends that RDBMS also fall short when handling very large data sets, as they are not designed to function with distributed systems. Baas's research confirms that it is suitable to use a NoSQL system when handling spatial data.

3 Temporal Databases

In contrast to spatial databases, which store geographic data, temporal databases store data that contains time stamps. Snodgrass [15] identifies the differences between temporal databases and conventional databases, outlining temporal databases' ability to easily specify queries over previous database states. Dyreson et al. [9] aimed to reach a consensus on temporal database terminology by defining many key concepts relating to temporal databases, as detailed in Section 3.1. Allen [2] discussed the logic behind querying temporal intervals in 1983, and this logic continues to be used in modern temporal database systems today [13].

3.1 Theory

There are two primary dimensions of time in a temporal database [9]. The *valid time* determines when a fact was true in reality. For example, when an event takes place in the real world, the time this event occurred is stored in the database.

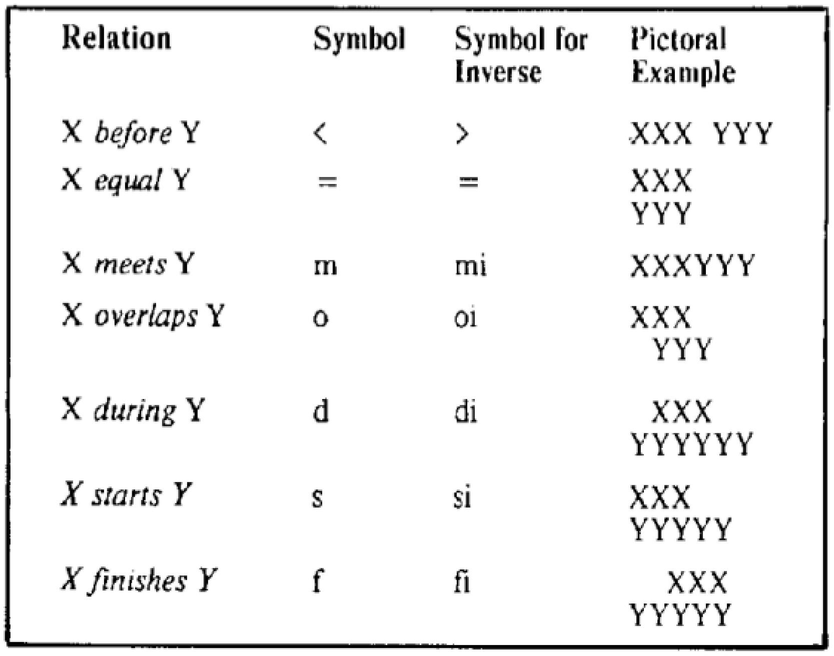
[1] *PostGIS*. http://postgis.net.

[2] *Neo4J*. http://neo4j.com.

This time value is independent of when the data was recorded in the database. On the other hand, the *transaction time* is the time that a record was inserted into the database. Temporal databases also support User-defined time, which can be recognised by users but not the database management system. Dyreson also discusses the *bitemporal relation*, which contains both a valid and transaction time.

3.2 Querying

Queries performed on temporal databases aim to retrieve objects based on temporal information. Allen [2] outlined the possible temporal relationships between intervals of time as shown in Figure 3. According to Monger at al. [13], modern temporal data management systems (TDMS) are able to handle these relationships by utilising database triggers and procedures that serve to handle temporal data without affecting the other abilities of the database. Toman [16] discusses a few example temporal queries, one of which is "find names of [employees] whose salary has decreased". This query demonstrates a temporal database's ability to recall the past state of an object. A more complex query, such as "find all periods when no country was independent", requires the system to retrieve time period intervals based on the value of an object's attribute.



Relation	Symbol	Symbol for Inverse	Pictoral Example
X *before* Y	<	>	XXX YYY
X *equal* Y	=	=	XXX YYY
X *meets* Y	m	mi	XXXYYY
X *overlaps* Y	o	oi	XXX YYY
X *during* Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Figure 3: The thirteen possible relationships between temporal intervals [2].

3.3 Existing Implementations

It is rare to find existing implementations of purely temporal databases today, however many database management systems incorporate temporal features. Monger et al. [13] provide an overview of temporal data management and NoSQL databases, and use RavenDB[3] as an example DBMS. RavenDB is a NoSQL document store, meaning it stores its data in a semi-structured format that the DBMS is able to understand. Modifying the database allows for temporal queries based on the operators outlined by Allen [2] in Figure 3 to be possible.

4 Spatio-temporal Databases

The desire to handle both spatial and temporal data storage and retrieval results in spatio-temporal databases. These databases are widely used due to their ability to store both space and time information. This section will review work conducted by Abraham and Roddick [1], as well as Erwig et al. [11] and Pelekis et al. [14]. Each researcher discusses the background of spatio-temporal databases and the queries supported by these databases, although Erwig et al. place emphasis on the use of spatio-temporal databases to model and query moving objects.

4.1 Theory

Pelekis et al. [14] begin their paper by focusing on the semantics, or terminology, relating to spatio-temporal databases. These semantics are divided into three sections — temporal, spatial, and spatio-temporal. One of these terms is the *type of change*, which determines the types of change spatio-temporal data models are able to handle. Abraham and Roddick [1] also discuss this idea in their paper, and, similarly to Pelekis et al., outline the eight possible spatio-temporal changes of a geographic object. Both researchers utilise the same diagram to demonstrate this notion, as shown in Figure 4. The theory behind the spatio-temporal database is a combination of the theories behind the sum of its two key components, spatial and temporal databases. The interaction between spatial and temporal query abilities is what makes the spatio-temporal database a powerful tool for tracking the location of objects over time.

[3] *RavenDB*. http://ravendb.net.

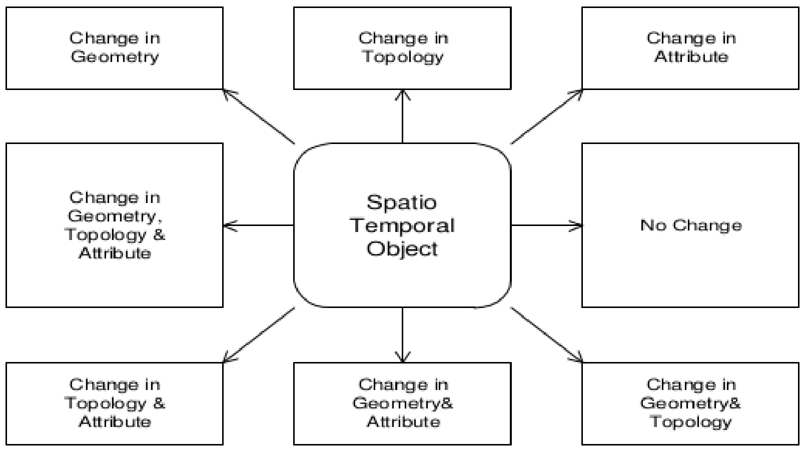

Change in Geometry
Change in Topology
Change in Attribute
Change in Geometry, Topology & Attribute
Spatio Temporal Object
No Change
Change in Topology & Attribute
Change in Geometry& Attribute
Change in Geometry& Topology

Figure 4: The eight possible types of change of a spatio-temporal object [14, 1].

4.2 Querying

The queries supported by spatio-temporal databases are described by each of the aforementioned researchers. Pelekis et al. [14] focus on the different queries available, categorising the queries into spatial, temporal, and spatio-temporal groups. These queries encompass all of the queries supported by spatial databases, such as the location of an object, the object's proximity to other objects, and the nearest neighbour of the object. Spatio-temporal queries also comprises the queries available to temporal databases, such as determining the spatial state of an object at a particular time. Finally, spatio-temporal queries include simple queries, range queries, and behaviour queries. Erwig et al. [11] provide numerous examples of these queries, such as:

"Which destinations can be reached from San Francisco within 2 hours?", which can be written in SQL as

```
SELECT to
FROM flights
WHERE from = ‘‘SFO’’ AND duration(route) <= 2.0
```

"Which flights went through a snow storm?", written as

```
SELECT id
FROM flights, weather
WHERE kind = ‘‘snow storm’’ AND duration(visits(route, area)) > 0
```

These queries are both spatio-temporal and involve the combination of spatial and temporal queries.

4.3 Existing Implementations

Several spatio-temporal database management systems exist today. The most popular is MongoDB[4], a NoSQL document-oriented database that is designed to be flexible, fast, and scalable [5]. OrientDB[5] shares many features with MongoDB, but the two differ in that OrientDB extends the document model to a document-graph model, in order to better represent relationships between models. Finally, CouchDB[6] is another NoSQL database that can be extended to cater to spatio-temporal data. CouchDB is covered in detail by Anderson et al. in their book entitled "CouchDB - the Definitive Guide" [3]. Each of the aforementioned NoSQL databases is well-documented and relatively mature when compared to other databases of the same type.

5 Conclusion

The concepts behind spatial, temporal, and spatio-temporal databases have been explored in this literature review. This has been achieved by conducting research across a range of papers and books written by prominent researchers in the area. There are a few popular NoSQL DBMS that are able to handle spatio-temporal data, namely MongoDB, OrientDB, and CouchDB, each of which is well-documented and mature. Ultimately, this literature review has provided a modern perspective on spatio-temporal databases by discussing their components, theory, query types, and existing implementations.

[4] *MongoDB*. https://www.mongodb.org/ .
[5] *OrientDB*. http://http://www.orientechnologies.com/
[6] *CouchDB*. https://http://couchdb.apache.org/.

References

[1] Tamas Abraham and John F Roddick. Survey of spatio-temporal databases. *GeoInformatica*, 3(1):61–99, 1999.

[2] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[3] J Chris Anderson, Jan Lehnardt, and Noah Slater. *CouchDB: the definitive guide.* " O'Reilly Media, Inc.", 2010.

[4] B Baas. *NoSQL spatial: Neo4j versus PostGIS.* PhD thesis, TU Delft, Delft University of Technology, 2012.

[5] Kyle Banker. *MongoDB in action.* Manning Publications Co., 2011.

[6] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. *The R*-tree: an efficient and robust access method for points and rectangles*, volume 19. ACM, 1990.

[7] Vania Bogorny and Shashi Shekhar. Spatial and spatio-temporal data mining. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1217–1217. IEEE, 2010.

[8] Graeme F Bonham-Carter. *Geographic information systems for geoscientists: modelling with GIS.* Elsevier, 2014.

[9] Curtis Dyreson, Fabio Grandi, Wolfgang Käfer, Nick Kline, Nikos Lorentzos, Yannis Mitsopoulos, Angelo Montanari, Daniel Nonen, Elisa Peressi, Barbara Pernici, et al. A consensus glossary of temporal database concepts. *ACM Sigmod Record*, 23(1):52–64, 1994.

[10] Max J Egenhofer and John Herring. Categorizing binary topological relations between regions, lines, and points in geographic databases. *The*, 9:94–1, 1990.

[11] Martin Erwig, Ralf Hartmut Gu, Markus Schneider, Michalis Vazirgiannis, et al. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296, 1999.

[12] Ralf Hartmut Güting. An introduction to spatial database systems. *The VLDB JournalThe International Journal on Very Large Data Bases*, 3(4):357–399, 1994.

[13] Morgan D Monger, Ramon A Mata-Toledo, and Pranshu Gupta. Temporal data management in nosql databases. *Journal of Information Systems & Operations Management*, 6(2):237–243, 2012.

[14] Nikos Pelekis, Babis Theodoulidis, Ioannis Kopanakis, and Yannis Theodoridis. Literature review of spatio-temporal database models. *The Knowledge Engineering Review*, 19(03):235–274, 2004.

[15] Richard Thomas Snodgrass. Temporal databases. In *IEEE computer*. Citeseer, 1986.

[16] David Toman. Point vs. interval-based query languages for temporal databases. In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 58–67. ACM, 1996.

[17] Xiaopeng Xiong, Mohamed F. Mokbel, and Walid G. Aref. Spatio-temporal database. In *Encyclopedia of GIS*, pages 1114–1115. Springer US, 2008.

[18] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 443–454. ACM, 2003.