Chapter 9 – Spatial and Temporal DBMS Extensions

Swetha Namburi

Introduction

In our day-to-day life, maintaining an accurate database is very important. A database is a tool which is used to store and keep record of information. The database can be anything right from list of groceries to a telephone directory. For example, when you are travelling to a new place, you might continuously want to know the closest gas station on your way or you need to be reminded to buy drinks when you are close to a supermarket. So, in these two cases the data is continuously changing as per location and time. This information requires a separate database unlike relational database management system (DBMS) for it to be processed as the queries are based on space and time which is called a *spatio –temporal database.*

Spatio-temporal database concepts

Before getting to know about the implementation of the spatio-temporal DB, let me first explain about spatial and temporal databases because they are needed to create spatio-temporal database systems.

Spatial Databases

- Definition: Many applications in various fields require management of *geometric, geographic or spatial* data (data related to space) such as model of the human brain, a geographic space: surface of the earth, man-made space: layout of VLSI design, 3-D space representation of the chains of protein molecules etc. A spatial database is a database system (DBMS) that is optimized to store and query basic spatial objects. It stores the spatial attributes, which have properties related to space. A relational database manages different types of numeric and character data but not objects such as points, lines and polygons. To manage this kind of data and also complex structures such as linear networks, 3D objects, Triangulated irregular networks and linear networks, spatial databases are used. For a typical database, additional features have to be added for the efficient processing of spatial data types.
- Modeling: Let us assume a 2 dimensional Geographic Information System application; two basic things need to be represented. They are:
  - *Objects in space* – rivers, cities or roads etc. different entities that are arranged in space and each of them has its geometric description. This comes under modeling single objects.
  - *Space* – to describe the total space that is saying something about every point in space. This is an example of modeling spatially related collection of objects.
- Fundamental data types – These can be used for modeling single objects.
  - Point: a moving vehicle, a University
  - Line: a road segment, road network
  - Region: a count, voting area

Fig 1: Fundamental data types in spatial DBMS, point, line and region

- Spatial Relationships

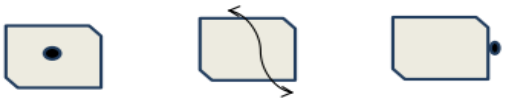


Fig 2: Few relationships between spatial objects, covered, intersect & adjacent

  - Topological relationships: Disjoint, touch, overlap, in, cover, equal
  - Direct relationships: Above, below, south_of, northeast_of etc.
  - Metric relationships: Distance
- Spatial Operations: There are four classes of operations based on the sets defined from the fundamental data types. Let E = {lines, regions}, G = {points, lines, regions}
    1. Spatial Predicates for topological relationships:
       ∀ g in G, ∀ e1, e2 in E, ∀ area in regions
       g x regions -> bool **inside**
       e1 x e2 -> bool **intersects, meets**
       area x area -> bool **adjacent, encloses**
    2. Operations returning atomic spatial data type values:
       ∀ g in G,
       lines x lines -> points **intersection**
       regions x regions -> regions **intersection**
       g x g -> g **plus, minus**
       regions -> lines **contour**
    3. Spatial operations returning number:
       ∀ g1 x g2 in G,
       g1 x g2 -> real **dist**
       regions -> real **perimeter, area**
    4. Spatial operations on set of objects:
       ∀ obj in OBJ, ∀ g, g1, g2 in G,
       Set(obj) x (obj->g) -> geo **sum**
       Set(obj) x (obj->g1) x g2 -> set(obj) **closest**
- Spatial Querying: Below listed are the fundamental algebraic operations on spatial data.
  - **Spatial selection:** This query returns the objects which satisfies a spatial predicate with the query object. Example: All small cities no more than 200kms and population no less than 500 from Fayetteville
    SELECT name from cities c WHERE dist(c.center, Fayetteville.center)<200 and c.pop>500
  - **Spatial Join:** This compares any two joined objects based on a predicate on their spatial attribute values. Example: Find all cities within less than 100kms for each river pass through texas.
    SELECT c.name FROM rivers r, cities c WHERE r.route intersects Texas.area and dist(r.route, c.area) < 100km

Below listed are some general spatial queries:

- **Nearness queries:** requests objects that lie near a specified location
- **Nearest neighbor queries:** Find the nearest object that satisfies given conditions based on a given point or an object
- **Region queries:** These deal with objects that lie partially or fully inside a specified region

Spatial data is generally queried using a graphical query language and the results are also displayed in a graphical manner. To support the data types such as lines, polygons and bit maps, many extensions to SQL have been proposed to interface with back end and the graphical interface constitutes the front-end. This allows relational databases to store and retrieve spatial information.

Temporal Databases

- Definition: A traditional DBMS is not good at handling queries which are related to moving objects because it cannot store a time series of data. So, the temporal DB came into existence which can store attributes of objects that changes with respect to time. While most databases tend to model reality at a point in time that is the "current" time, these databases model the states of real world across time. An RDBMS can also record changes in time by using a timestamp but it is not very efficient as the timestamp is not a continuously stored value for every trigger.

  Temporal DBMS manages time- referenced data, and times are associated with database entities. Most applications of database technology are temporal in nature:

  - Record-keeping apps : personnel, medical record and inventory management
  - Scheduling apps: airline, car, hotel reservations and project management
  - Scientific apps: weather monitoring
  - Financial apps: accounting and banking, portfolio management

  To handle temporal data objects, temporal DBMS systems should have the concept of valid time and transaction time integrated into it.

  Valid Time (vt): It is the collected times when the fact or value of the object is true with respect to the real world. It is like covering the past, present and future times.

  Transaction Time (tt): It is the time when the fact is current in the database. It may be associated with any database entity, not only with facts. Transaction time of an entity has duration from insertion to deletion.

Employee ID	Employee Name	Title	Valid Start Time	Valid End Time	Transaction Start Time
100200	John	Manager	12-Feb-2000	1-Jan-2004	1-Jan-2000
100200	John	Sr.Manager	2-Jan-2000	10-Mar-2008	31-Dec-2003
100300	Mary	Engineer	15-Feb-2008	18-Nov-2011	1-Jan-2008

Table 1: Example for Valid and Transaction time

This table represents the valid time and transaction time as valid start time, valid end time and transaction start time. We can observe in this table that the past history is not deleted like

the non-temporal DBMS tables. Time domain can be discrete or continuous but typically assumes that time domain is finite and discrete in database.

- Modeling: Two basic things have to be considered. One is predicting the future positions in which each object has a velocity vector and the database can predict the location at any time assuming linear movement. The second one is storing the history in which queries refer to the past states of the spatial database. For temporal database modeling, many extensions for relational models have been proposed. One of them is Bitemporal Conceptual Data Model (BCDM).

Customer ID	Tape Num	T
C1	T1	{(2,2),(2,3),(2,4),(3,2),(3,3),(3,4),…(UC,2), (UC,3),(UC,4)}
C2	T2	{(5,5),(6,5),(6,6),(7,5),(7,6),(7,7),(8,5),(8,6),(8,7)…(UC,5),(UC,6),(UC,7)}
C2	T1	{(9,9),(9,10),(9,11),(10,9),(10,10),(10,11),(10,12),(10,13),…(13,9),(13,10),(13,11),(13,12),(13,13),(14,9)…(14,14),(15,9),..(15,15),(16,9),…(16,15),..(UC,9),…(UC,15)}

Table 2: Example of Bitemporal Conceptual Data Model

In this example, the tuples are represented as a pair of transaction and valid time values. The values explanation is as follows:

1. Customer C1 borrowed T1 on 2nd for 3 days, and returned it on 5th.
2. Customer C2 borrowed T2 on 5th open-ended and returned it on 8th.
3. Customer C2 borrowed T1 on 9th and it should be returned on 12th. On 10th the date is extended to include 13th, but the tape is returned on 16th.

Advantages of BCDM:

- The representation is simple and also captures the temporal aspects of the facts stored in a database
- Since no two tuples with mutually identical existing values are allowed in BCDM relation instance, the full history of a fact is contained in exactly one tuple.

Disadvantages of BCDM:

- Internal representation of temporal info and its display to users is not good.
- It is very difficult to manage many timestamps of tuples as they keep on increasing as the time length increases.
- Timestamp values are hard to understand in BCDM format.

- Querying: Temporal queries can be expressed in any general query language such as SQL, but with great difficulty. A temporal language design should consider predicates on temporal values, time- varying nature of data, temporal constructs, supporting states and events, cursors, views, integrity constraints, periodic data, schemas, modification of temporal relations. Many temporal query languages have been defined to simplify modeling of time as well as time related queries. Some of the operations on temporal databases:
  - Snapshot: A snapshot of a temporal relation at time t consists of the tuples that are valid at time t, with the time-interval attributes projected out.
  - Temporal Selection: selects data based on time attributes.
  - Temporal projection: the tuples in the projection get their timestamps from the tuples in the original relation.

- Temporal Join: the time-interval of a tuple in the result is the intersection of the time-intervals of the tuples from which it is derived.

Example: Find where and when will it snow given Clouds(X, Y, Time, humidity) and Region(X, Y, Time, temperature)
(SELECT x, y, time FROM Cloud WHERE humidity>=80) INTERSECT (SELECT x, y, time FROM Region WHERE temperature <= 32)

Spatio-Temporal Databases

- Definition: Spatio-temporal databases can be defined as a database that embodies spatial, temporal and spatio-temporal database concepts and captures both spatial and temporal aspects of data as per Wikipedia.
- Applications: There are three types of Spatio-temporal applications.
  Involving objects with continuous motion: navigational systems manage moving objects, objects change position, but not shape
  Dealing with discrete changes of and among objects: objects shape and their positions may change discretely in time
  Managing objects integrating continuous motion as well as changes of shape: A "storm" is modeled as a "moving" object with changing properties and shape over time.
- Spatio-Temporal Semantics: To explain about database model, we need to know some semantics.
  Spatio-temporal attribute: An attribute that contains the evolution of a spatial object in time that is spatial attribute and time attribute.
  Spatio-temporal object: An object that contains a ST attribute
  Spatio-temporal evolution: the evolution of an object in time
  Examples: land parcels are evaluated when a weekday is finished and this kind of evolution is called a discrete point based that is the shape of a land parcel is changing in time, but only in discrete steps.
- Spatio-Temporal Database Models: A data model gives a detailed understanding of the system for which the design is created. They can ease communication among the main programmer, designer and the ultimate customer. The main aspect of spatio-temporal Information systems is the spatio-temporal Data models. These models describe the data types, relationships, operations and rules to maintain database integrity for the entities of spatio-temporal databases. They also must provide adequate support for spatio-temporal queries and analytical methods to be implemented in the spatio-temporal Information Systems.

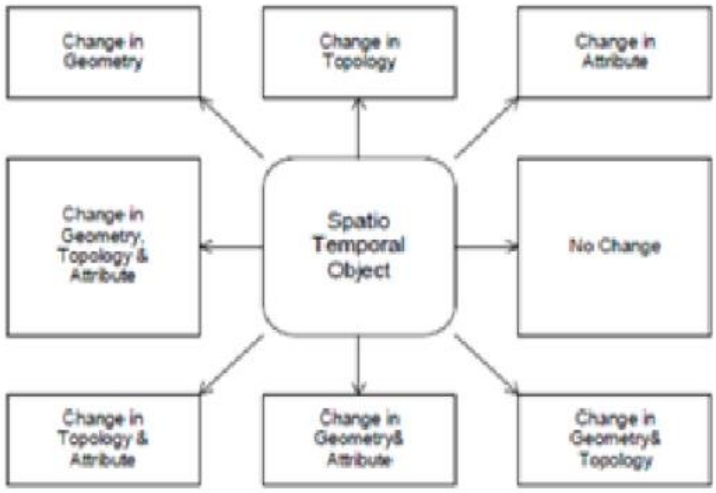


Fig 3: Possible types of changes for spatio- temporal object

To design these models the following things should be considered:

Temporal data models – granularity, temporal operations, time density and representation.

Spatial data models – structure of space, orientation, direction, and topology and measurement information.

A Spatio-temporal model is formed by combining the data types, objects, topology of space-time, changes with respect to time and space, object identities and dimensionality.

The different data models that have been suggested for designing spatio-temporal database systems are:

The Snapshot Model – This is the simplest model. In this model, time is considered as a characteristic of the location. It stores redundant information and so, occupies more memory. This model represents temporal aspects of data time-stamped layers on top of spatial data model. Below figure is an example of the snapshot model. Each layer is a collection of temporally homogenous units of one theme. It shows the states of a geographic distribution at different times without explicit temporal relations among layers. There is no direct relation between two successive layers. If at least one spatial object position or shape is changed, one spatial object is created or one spatial object disappears, a new layer is stored with a new timestamp.

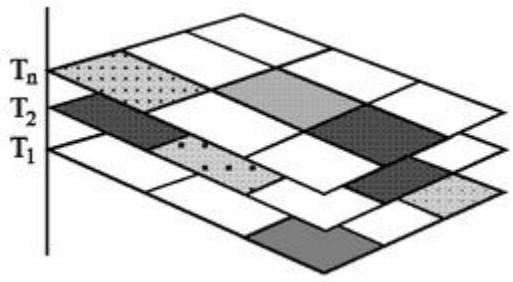


Fig 4: An example of the snapshot model

Pros: This model can be easily implemented as the present state of all objects is available at any moment.

Cons: If one object changes more rapidly than the other objects, all the newly formed layers contain the same information about those objects. The list of layers does not contain explicit information about the changes and in order to see the changing suffered by an object, we have to compare the successive layers.

To avoid the disadvantage that is to reduce the amount of redundant data, delta-files are proposed. In the usage of delta-files only the current and initial layers are stored. The changes that took place are stored in delta-files. To find out the evolution of one spatial or its state in a particular moment, then we can read the delta-files beginning with the first layer to know the object's state.

Simple Time Stamping – In this approach, formation and deletion time of the object is available in the form of a pair of time stamps for each object. Through this model, we can easily obtain particular states of an object at any time.

Event Oriented Model – Instead of pair of time stamps, changes and events made to the objects are maintained in a transaction log. By using this model, we can easily obtain the current state by using data from the transaction logs.

Three-Domain Model - This model considers that the data belong to one of the three domains: spatial, semantic and temporal. So the objects of each domain are allowed to be treated in an independent manner.

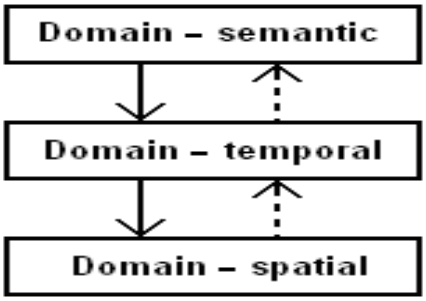


Fig 5: Three-Domain model

Space-Time composite Data Model (STC) – In this model, a polygon mesh is created by projecting each line in time and space onto a spatial plane and they are intersected with each other.

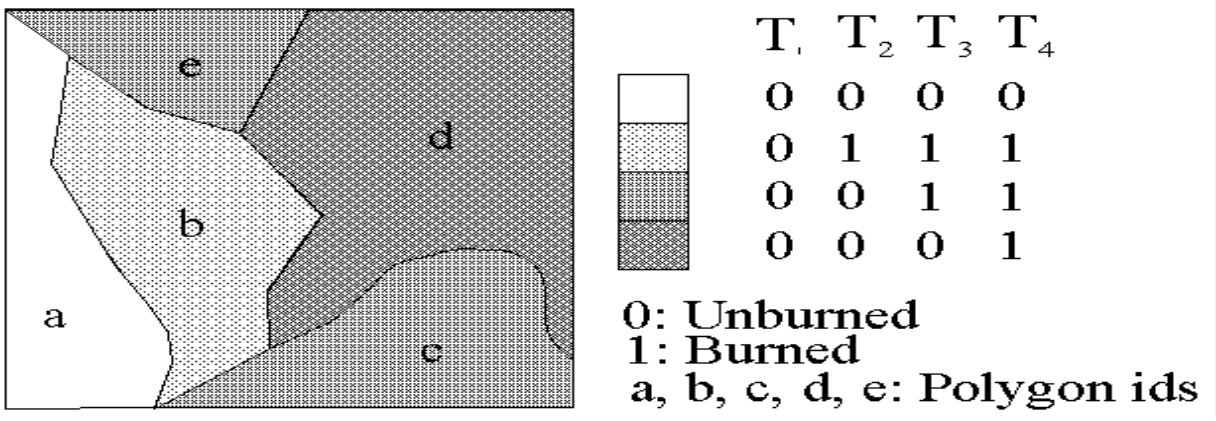


Fig 6: An example of an STC layer for burns

In the above figure, each of the regions a, b, c, d, e can be in one of two states: Unburned (1) and burned (0). Each region has its own spatial characteristic and the evolution of its state in time.

The spatial objects are represented in a vectorial manner, and the temporal domain is linear, discrete and both time types that is transaction and valid time are supported. This model is capable of capturing temporality with respect to space and time in a attribute but fails to record temporality with respect to space among the attributes. The advantage of this model over snapshot model is this does not store redundant data.

- Spatio-temporal Operators: Below listed are some of the operations available for spatial-temporal database.
  1. Location-temporal Operator – returns the spatial representations of object A valid at a time T. ST_SP(A,T)
  2. Orientation-temporal operators - returns a Boolean value indicating whether there exists specific relationship between two objects (A and B) Example: ST_SOUTH (A, B) and ST_WEST (A, B) etc.
  3. Metric-temporal operators – To find the metric of object A at a time value T, ST_AREA (A, T). To find the distance between two spatial component A and B at time T: ST_DISTANCE(A,B,T)
  4. Topological-temporal operators – To find the topological relationship between A and B during a certain time T. This returns a Boolean value. ST_DISJOINT(A,B,T)
- Spatio-temporal Querying: To retrieve the data from the database, we need queries so that it is easy to find things instead of searching the whole database. So, we need queries to handle spatial, temporal and spatio-temporal properties.

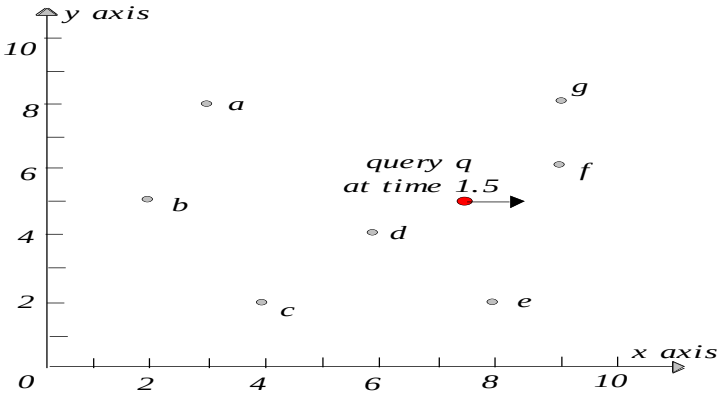
  Range Queries: To find all the objects that will intersect a given range Q and the time they intersect Q.

  Nearest Neighbor queries (NN queries): find the nearest object to a given query point q at all timestamps.

  Result:

  R= {d}

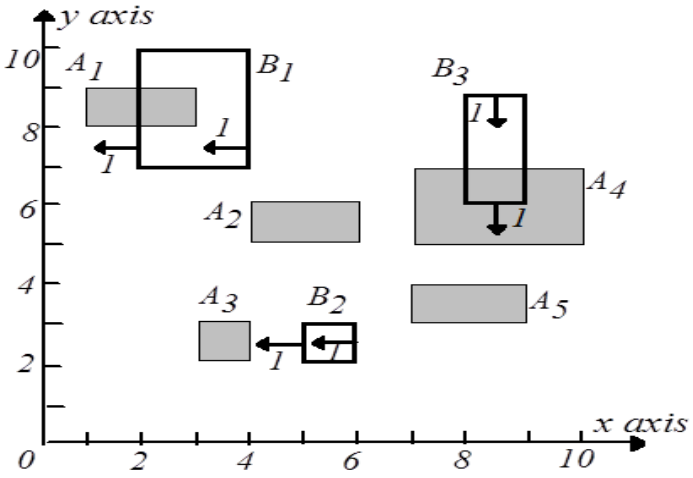
  {d, [0, 1.5], **f (1.5,** ∞**]}**



  Aggregate Queries: There are two types in this query type-

  - Aggregate range query: find how many objects passed through a range Q during a given time interval T
  - Density query: find all regions whose density at t is larger than μ.

  Join Queries: Find all the pairs of objects whose extents intersect for every timestamp



Result:
R={
‹A_1, B_1› [0 , 3] ,
‹A_4, B_3› [0 , 4] ,
‹A_3, B_2› [1 , 3] ,
‹A_5, B_3› [2 , 6]
}

Similarity Queries: Find objects that moved similarly to the movement of a given object O over an interval T.

Spatial Queries: Find the super market nearby, where is this park?

Spatial Query to check whether a particular river flows through a particular state or not –

SELECT rivers, states FROM river, state WHERE river INTERSECT state.

Temporal Queries: position of an employee at a particular time

SELECT position_title, employee, name FROM employee time = now ()

Spatio-temporal queries**:** These queries ask for data which includes both space and time such as moving objects.

SELECT routes (10.00...11.00) FROM routes WHERE flight id = "AR123". – Query to examine routes between a certain times based on the id of a particular flight.

- **Query Languages –** To handle a spatio-temporal query, additional features must be added to query languages of spatial and temporal databases to handle the complexity added from both the temporal and spatial dimensions.
  Query Languages that are convenient for the processing of spatio-temporal query are:

  Hibernate Query Language (HQL) – It is an extension of the relational query language. Operations of this language are similar to spatial relationship operators. It has nested queries, conditional statements, loops and function definitions.

  Temporal query language extensions – Ariav's TOSQL, ATSQL2, Snodgrass' TQuel

  Spatial Query language extensions – Berman's Geo-Quel, Joseph's PicQuery, Ooi's GeoQL

  SQL based – STSQL

- **Spatio- temporal DBMS architecture:** Now that I have discussed about the different models that can be used for spatio-temporal databases, Query languages that can be extended from spatial and temporal databases, operators that are needed to be considered into account for the efficient processing of spatio-temporal databases, let me now explain about the architectures proposed for spatio-temporal database management systems.

Designing a good architecture is very important because that is the one which describes how data is viewed by the users in the database.

A lot of architectures have been suggested for these database management systems, but only the important ones are described below:

Standard Relational with Additional Layer: In this traditional DBMS acts as the bottom layer on which another layer of spatio-temporal database is added. Two different approaches are available in this architecture:

Thin layer approach - The main idea here is use the facilities of existing DBMS as much as possible and spatio-temporal aspects are represented by the abstract data types.

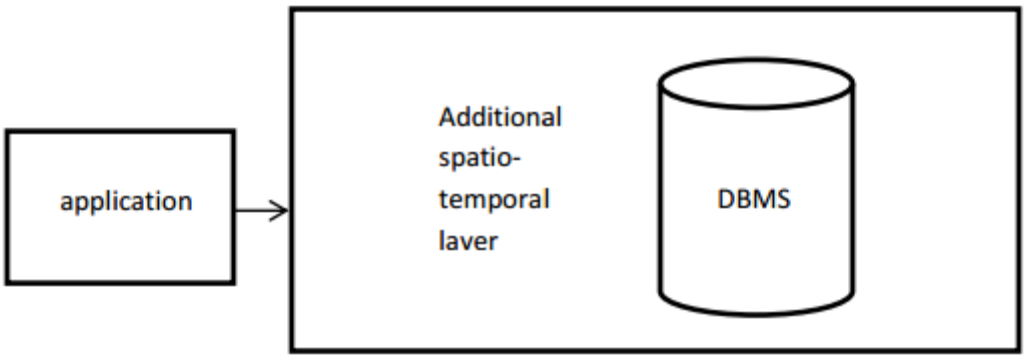


Fig 7: Thin-layer spatio-temporal DBMS architecture

Thick layer approach – DBMS is used as constant object storage and spatio-temporal aspects are represented by the middle-ware.

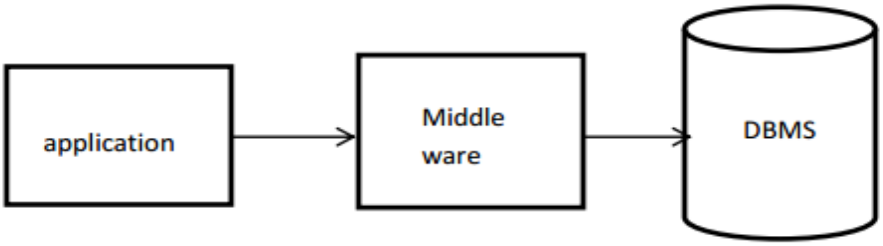


Fig 8: Thick-layer spatio-temporal DBMS architecture

File system based spatio-temporal DBMS: Same as above, traditional DBMS is used as the bottom layer. Instead of a middle-ware, spatial and temporal data are stored by using the file system. The main concern of this architecture is maintaining good communication between file system and DBMS which is very important without leaking the data between file system and DBMS.

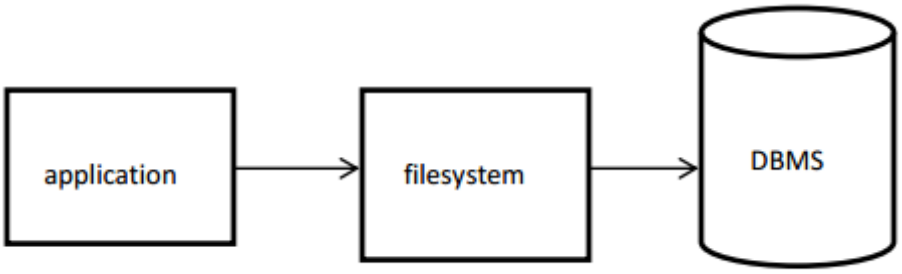


Fig 9: A file system based spatio-temporal DBMS

Extensible DBMS: Without adding any additional layers to the DBMS, the database kernel itself is extended to support spatio-temporal aspects such as storage structures, data types, access methods and query processing.

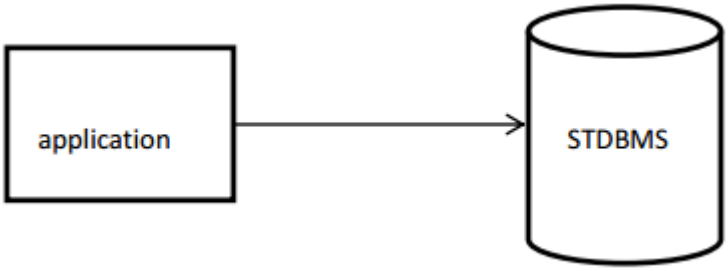


Fig 10: Extensible DBMS

- **Spatio-temporal Storage and Indexing:** Spatio-temporal databases need a lot a storage when compared to traditional DBMS as the data varies continuously based on space and time which leads to the generation of large volumes of data. Apart from traditional DBMS, spatio-temporal databases are always used for real – world applications and the data should be processed in a timely manner. Because of these reasons, the cost of I/O and computation is high. Therefore, to process spatio-temporal aspects of data, using good indexing and storage techniques are necessary.

  Indexing Methods: Spatio-temporal data indexing is generally divided into two types –

  - Indexing historical data: storing the history of a spatio-temporal evolution. Available method is HR-tree
  - Indexing current data: Finding the current and future positions of moving objects. Methods available – Dual transformation and TPR-tree

  Requirements – Minimal I/O cost, low space, best data clustering

  To meet the above requirements, the following indexing methods are proposed:

  Multi-dimensional spatial indexing – On the top of a spatial object, time is handled as an additional component.

R- tree based indexing approach – Information about the spatial objects is stored by referencing the maximum extent of the objects which is called Minimum Bounding Rectangle (MBR).

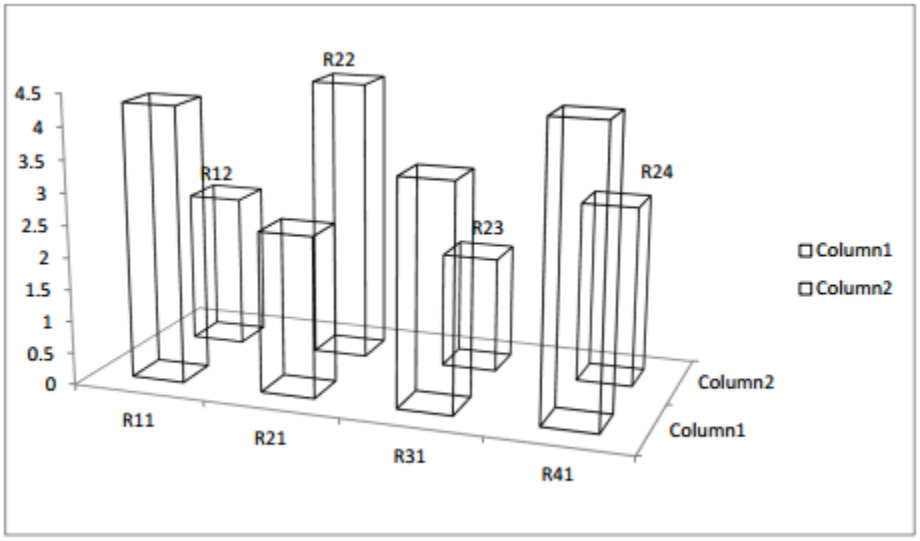


Fig 11: 3D Visualization of R-tree

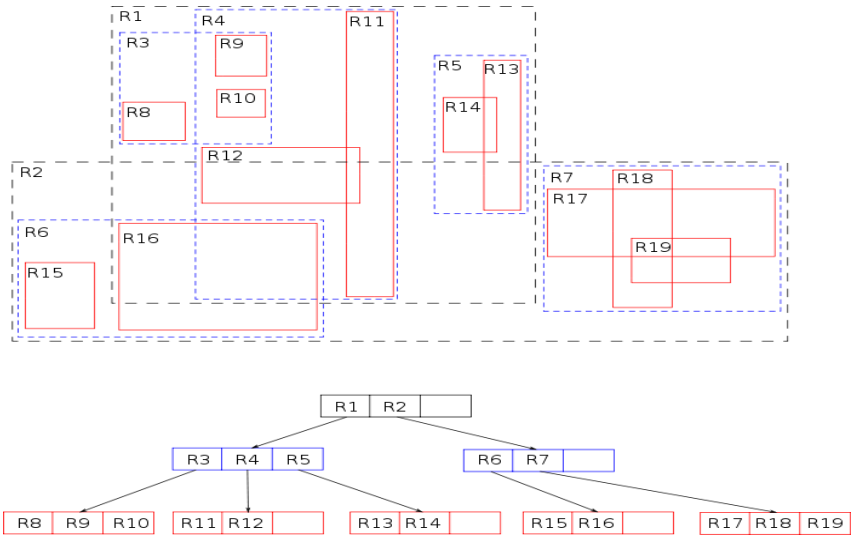
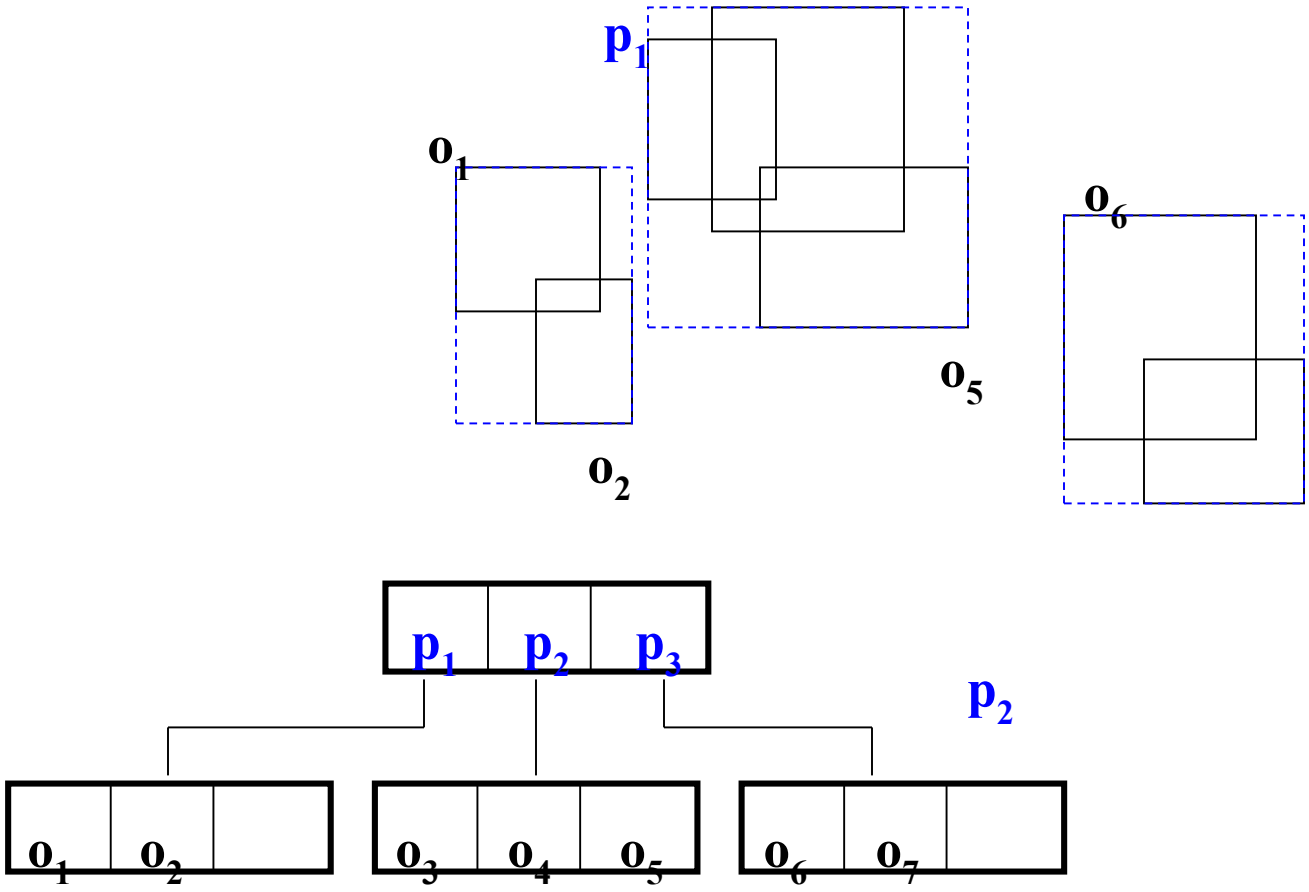


Fig 12: 2D visualization of MBR and its corresponding R-tree

Partitioning based indexing – The data is partitioned based on the dimensions, space and time. There are two different approaches available depending on the dominance of dimensions – space dominant and temporal dominant partitioning. It depends on the application considered.

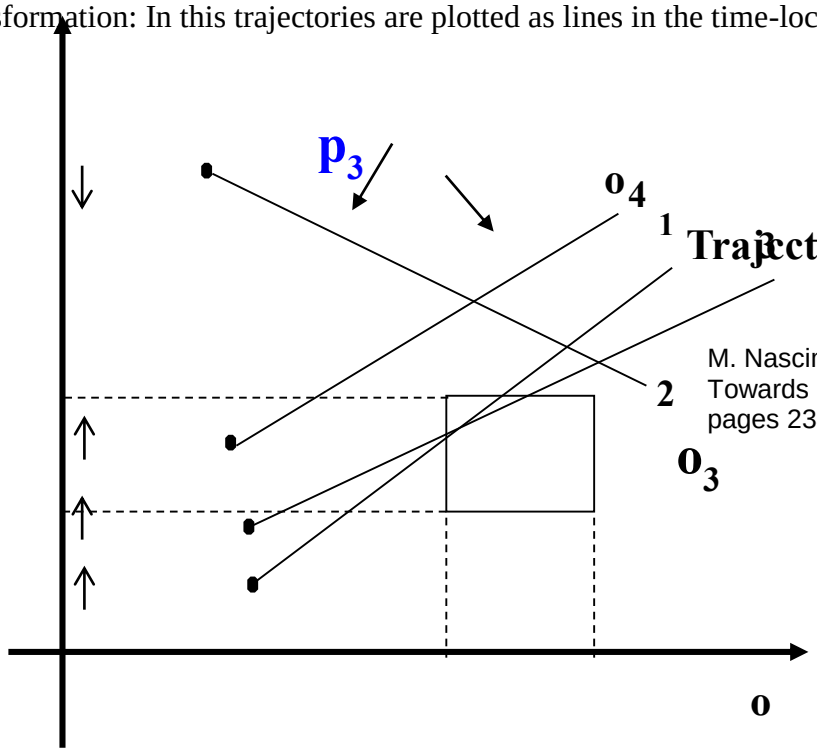
Multidimensional segment indexing approach – In this approach, historical data is indexed based on the time intervals considering time as an independent attribute. This is mainly used to handle historical data.

Historical R-tree (HR-tree): In this, an R-tree is maintained for each timestamp in history. All trees at continuous timestamps can share branches to save space.



Time-parameterized R-tree (TPR-tree): The minimum bounded rectangles are stored as functions of time *MBR (t) = MBR (t_0) + V (t).* We can calculate the MBR at any time instant in the future as the MBRs grow with time.

Dual transformation: In this trajectories are plotted as lines in the time-location space.



M. Nascimento and M. Silva, Towards historical R-trees, in SAC, pages 234-240, 1998

- **Summary of Indexing Techniques:**

Index	**Disk-based/In-memory**	**Balanced**	**Efficient query type**	**Dimensionality**	**Comments**
HR-tree	Disk-based	Yes	Timestamp queries	Low	High Space consumption; inefficient time range query
Dual-transformation	Disk-based	Yes	Range queries	Low	Doubling dimensionality only for point objects
TPR-tree	Disk-based	Yes	Time parameterized queries – range queries, KNN queries etc	Low	Complex tree structure; inefficient update

Table 3: Summary of Indexing Techniques

Extension of widely known spatial DBMS (Oracle Spatial) within a Query language (ATSQL2)

We have discussed about the different query languages and DBMS above. Spatio-temporal databases are widely used in Geographical Information Systems (GIS). GIS is a computer system used by businesses, schools, governments etc. GIS grants the user to work with huge volumes of data to map, model, and query based on their location. It allows combining

information, creating maps, proposing effective solutions, present powerful ideas and visualizing scenarios. By using GIS, all the information can be stored as a collection of layers which can be linked together based on the time, location.

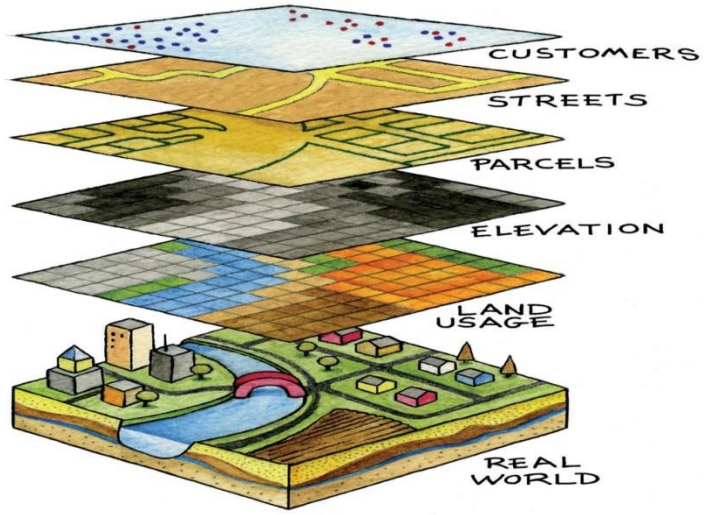


Fig 13: A GIS as a Layered Cake

Importance of spatial and temporal data in real world applications is the main reason for the evolution of Geographic Information Systems. Even though the research and development is GIS has been increasing there are still issues like the regular GIS not giving support to the temporal dimension of data by giving priority to the spatial dimensions. There are no GIS providing full temporal support over the valid time domain, agreeing the combined management of spatial-temporal data at the DBMS level.

To overcome the above limitation, an idea was proposed and to extend the spatial DBMS within a query language ATSQL2. In simple words, a spatial extension is added to ATSQL2 in order to provide spatial-temporal data management, through the ability to query the underlying DBMS with questions having sequenced and non-sequenced valid-time semantics, combined with the usage of spatial data types, operators and spatial functions[1].

This idea was proposed in a paper named “Spatial Time DB – Valid Time Support in Spatial DBMS”. It was proved in the paper by this extension; temporal dimensions will be given a higher priority when compared to spatial dimensions by using TimeDB as an underlying DBMS. For this to be proved TimeDB architecture was first analyzed and changes needed to be done to ATSQL2 were identified.

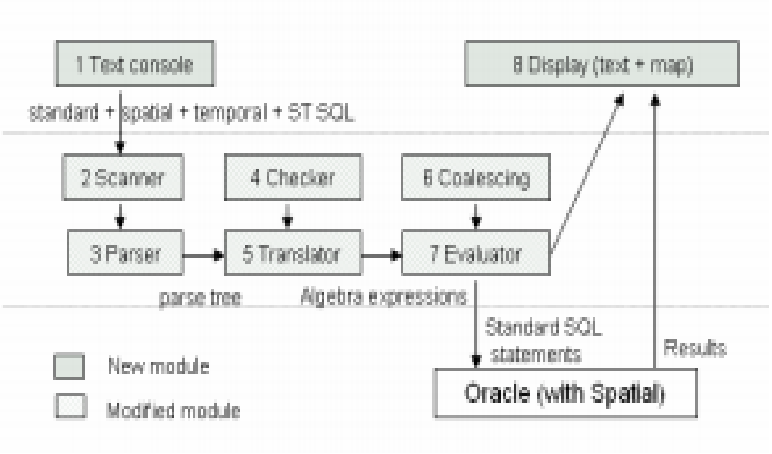


Fig 14: Spatio-temporal layer architecture

The system proposed required changes to most of the initial TimeDB modules. Some of the changes include:

- Scanner – being able to identify new spatial constructs
- Parser – being able to support spatial tables, arguments, method calls, indexing
- Translator – being able to analyze relation attributes used as spatial arguments

For testing, they used a database called TimeDB, which was subjected to changes in all of its components and results were produced.

Conclusion

To summarize, spatial –temporal DBMS is very important for moving objects and it has many applications in our day-to-day life. In this paper, I have covered individually about spatial, temporal and integration of spatial l and temporal databases. There are still many researches going on about the open issues in this database such as Database size- These databases contain large amount of information and the temporal information further increases the database size and difficulty of rapid data retrieval, Legacy Systems and Data Quality. It would be beneficial from doing research in both spatial and temporal database.

References

(i) T. Abraham and J.F. Roddick. ``Survey of Spatio-temporal databases," Geoinformatica, Vol. 3:61±69, 1999.

(ii) Spatio-Temporal Database presentation by Jiyong Zhang, School of Computer and Communication Sciences, Jan 25, 2005

(iii) Markus Innerebner, Michael Bohlen, Igor Timko "A Web Enabled Extension of a Spatio-Temporal DBMS", Proceedings of the 15th International Symposium on Advances in Geographic Information Systems, 2007

(iv) Alexandre Carvalho, Cristina Ribeiro, A.Augusto Sousa, "Spatial TimeDB – Valid Time Support in Spatial DBMS"

(v) http://en.wikipedia.org/wiki/R-tree and http://en.wikipedia.org/wiki/Spatiotemporal_database