Specifications for Efficient Indexing in Spatiotemporal Databases

Yannis Theodoridis, Timos Sellis, Apostolos N. Papadopoulos, Yannis Manolopoulos

October 1, 2007

Presented by: Kristian Torp
Outline

1. Motivation
2. Specification
3. Query Support
4. Classification of STAMs
5. Evaluation
Outline

1. Motivation
2. Specification
3. Query Support
4. Classification of STAMs
5. Evaluation
A spatio-temporal object $o$ (declared by its identification number $o_{id}$) is a time-evolving spatial object, i.e., its evolution (or history) is represented by a set of triplets $(o_{id}, s_i, t_i)$, where $s_i$ is the location of object $o_{id}$ at instant $t_i$ ($s_i$ and $t_i$ are called space-stamp and time-stamp, respectively).
Definition of Spatio-Temporal Object

Definition

A spatio-temporal object $o$ (declared by its identification number $o_{id}$) is a time-evolving spatial object, i.e., its evolution (or history) is represented by a set of triplets $(o_{id}, s_i, t_i)$, where $s_i$ is the location of object $o_{id}$ at instant $t_i$ ($s_i$ and $t_i$ are called space-stamp and time-stamp, respectively).

A merge of two existing research fields

- Spatial databases
- Temporal databases
New Challenges

The Main Challenge

Efficient manipulation of spatial objects and the relationship among them
The Main Challenge

Efficient manipulation of spatial objects and the relationship among them

Details to consider

- Object representation
- Query processing
- Index methods
Types of Indices

- Point Access Method (PAM)
- Spatial Access Method (SAM)
- Spatio-Temporal Access Method (STAM)
Outline

1 Motivation
2 Specification
3 Query Support
4 Classification of STAMs
5 Evaluation
The main discriminator

- Point
- Region

A spatio-temporal DBMS must support both types of data
<table>
<thead>
<tr>
<th>name</th>
<th>dept</th>
<th>VTS</th>
<th>VTE</th>
<th>TTS</th>
<th>TTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>NY</td>
<td>3</td>
<td>now</td>
<td>3</td>
<td>uc</td>
</tr>
<tr>
<td>Joe</td>
<td>LA</td>
<td>4</td>
<td>now</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>Joe</td>
<td>LA</td>
<td>4</td>
<td>11</td>
<td>11</td>
<td>uc</td>
</tr>
<tr>
<td>Joe</td>
<td>UK</td>
<td>11</td>
<td>now</td>
<td>11</td>
<td>uc</td>
</tr>
<tr>
<td>Sam</td>
<td>NY</td>
<td>12</td>
<td>now</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>Sam</td>
<td>NY</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>uc</td>
</tr>
</tbody>
</table>
sequenced select * 
from emp 
as of '2007-09-13';

select name, dept, vts, vte 
from emp 
where tts <= '2007-09-13'
and tte > '2007-09-13'

ATSQL

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>VTS</th>
<th>VTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>NY</td>
<td>3</td>
<td>now</td>
</tr>
<tr>
<td>Joe</td>
<td>LA</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>Joe</td>
<td>UK</td>
<td>11</td>
<td>now</td>
</tr>
<tr>
<td>Sam</td>
<td>NY</td>
<td>12</td>
<td>now</td>
</tr>
</tbody>
</table>
Temporal Support III

\textbf{ATSQL Result}

\begin{tabular}{|c|c|c|}
\hline
\textbf{Count} & \textbf{VTS} & \textbf{VTE} \\
\hline
1 & 3 & 4 \\
2 & 4 & 11 \\
2 & 11 & 12 \\
3 & 12 & \textit{now} \\
\hline
\end{tabular}
Temporal Support IV

```sql
select count(name), const_period.vts as vts, const_period.vte as vte
from empat13,
    (select t1.vts as vts, t1.vte as vte
from empat13 t1
where not exists (select *
from empat13 t2
where (t1.vts < t2.vts and t2.vts < t1.vte) or
      (t1.vts < t2.vte and t2.vte < t1.vte))
union
    select t1.vts as vts, t2.vts as vte
from empat13 t1, empat13 t2
where t1.vts < t2.vts and t2.vts < t1.vte
and not exists (select *
from empat13 t3
where (t1.vts < t3.vts and t3.vts < t2.vts) or
      (t1.vts < t3.vte and t3.vte < t2.vts))
union
    select t1.vts as vts, t2.vte as vte
from empat13 t1, empat13 t2
where t1.vts < t2.vte and t2.vte < t1.vte
and not exists (select *
from empat13 t3
where (t1.vts <= t3.vts and t3.vts <= t2.vts) or
      (t1.vts <= t3.vte and t3.vte <= t2.vts))
union
    select t1.vte as vts, t2.vts as vte
from empat13 t1, empat13 t2
where t1.vte < t2.vts
and not exists (select *
from empat13 t3
where (t1.vte < t3.vts and t3.vts < t2.vts) or
      (t1.vte < t3.vte and t3.vte < t2.vts))
```
(t1.vte < t3.vte and t3.vte < t2.vts))

union

select t1.vte as vts, t2.vte as vte
from empat13 t1, empat13 t2
where t2.vts < t1.vte and t1.vte < t2.vte
and not exists (select *
from empat13 t3
where (t1.vte < t3.vts and t3.vts < t2.vte) or
(t1.vte < t3.vte and t3.vte < t2.vte))

const_period

where empat13.vts <= const_period.vts and const_period.vte <= empat13.vte

group by const_period.vts, const_period.vte

order by const_period.vts;
Temporal Support VI

- Valid time
- Transaction time

<table>
<thead>
<tr>
<th></th>
<th>Transaction-Time: No</th>
<th>Transaction-Time: Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid Time: No</td>
<td>snapshot</td>
<td>transaction-time</td>
</tr>
<tr>
<td>Valid Time: Yes</td>
<td>valid-time</td>
<td>bitemporal</td>
</tr>
</tbody>
</table>
## Database Mobility

<table>
<thead>
<tr>
<th></th>
<th>Fixed Size</th>
<th>Dynamic Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Still objects</td>
<td></td>
<td>growing</td>
</tr>
<tr>
<td>Moving objects</td>
<td>evolving</td>
<td>full-dynamic</td>
</tr>
</tbody>
</table>
Other Criteria

- Loading of data
- Time-dimension peculiarity
- Support of spatio-temporal queries
  - More on this shortly
Outline

1 Motivation
2 Specification
3 Query Support
4 Classification of STAMs
5 Evaluation
Selection Queries

Example
Find all objects that have lied within a specific area (or at a specific point), during a specific time interval (or at a specific time instant)

Example
Find all spots where it rained yesterday at 12.00 pm

- Filtering on space, within a spatial range
- Filtering on time, at a specific instance or time interval
- Most common type of query
Join Queries

Example
Find all pairs of objects that have lied spatially close (i.e., within distance X), during a specific time interval (or at a specific time instant)

Example
Find all person biking from the railway station to the university staring at the railway station between 12:00 to 14:00 that got wet on the bike ride due to rain.

- Space and time is used in the join condition
- Properly a very expensive operation
Nearest-Neighbor Queries

Example
Find the 5 closest ambulances with respect to the accident place in a time interval of 2 minutes before and after the accident, knowing the directions and velocities of ambulances and the street map.

Example
Find the 10 persons that was closest to the rain shower that passed down town Aalborg between 11:12 and 11:17 yesterday.

Additional information needed
- Road network
- Speed and direction of the ambulances
Outline

1 Motivation
2 Specification
3 Query Support
4 Classification of STAMs
5 Evaluation
### Evaluation of Four Index Structures

<table>
<thead>
<tr>
<th>Specification / STAM</th>
<th>MR-tree</th>
<th>RT-tree</th>
<th>3D R-tree</th>
<th>HR-tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type support</td>
<td>region</td>
<td>region</td>
<td>region</td>
<td>region</td>
</tr>
<tr>
<td>Time support</td>
<td>transaction</td>
<td>transaction</td>
<td>valid</td>
<td>transaction</td>
</tr>
<tr>
<td>Data set mobility</td>
<td>full-dynamic</td>
<td>full-dynamic</td>
<td>growing</td>
<td>full-dynamic</td>
</tr>
<tr>
<td>Time-stamp update</td>
<td>chronological</td>
<td>chronological</td>
<td>static</td>
<td>chronological</td>
</tr>
<tr>
<td>Object approximation</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Obsolete entries</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Query processing</td>
<td>timeslice</td>
<td>no</td>
<td>timeslice</td>
<td>timeslice</td>
</tr>
</tbody>
</table>
Outline

1. Motivation
2. Specification
3. Query Support
4. Classification of STAMs
5. Evaluation
Good Points

- Provides an overview of a new field (at the time of publication)
- Overall good criteria
- Compares four specific index structures
- Paper is very readable
Could be Improved

- Missing more details on timeslicing and obsolete entries
- Missing 2D, 3D, nD discussion
- Not clear (to me) why loading of data is interesting
- Only R-trees in evaluation no quad-trees