Representation of Periodic Moving Objects in Databases

T. Behr  V. Teixeira de Almeida  R. H. Güting

Faculty of Mathematics and Computer Science
Database Systems for New Applications
FernUniversität in Hagen

Bolzano, February 23th 2007
Motivation

- some existing database systems can store moving objects
- there are two approaches
  - storing the current movement
  - storing the complete history of the movement
- movement is stored as a sequence of simple movements
- if a movement is repeated $n$ times, this movement is also stored $n$ times
Outline

Goals

Basic Ideas

The used Model

Example

Detection of Repetitions in Movements

Implementation

Graphical User Interface

Comparison
Goals

- representation of the history of the movement
  the model should not only store the current value of an object but the complete history of all changes
- avoid multiple storing of repeated movement
  changes which are repeated several times should be stored only once
- representation of non-periodic movements
  the model is not restricted to periodic movements
- nested repetitions
  the model should cover also more complicated types of repetitions
- independency of the underlying type
  the model should work with different datatypes in the same way
Basic Ideas

- sliced representation

we divide the definition time of the object into slices, each slice describes a simple movement of the object

- relative intervals

instead of using intervals fixed in time, we use relative intervals consisting of its duration together with closure properties

- tree representation

we represent a moving object within a tree with different kinds of nodes described on the next slides
Nodes describing the Basic Movement

- corresponds to a single slice
- dependent on the data type
- leaves of the tree
- movement described by a simple function
  
  e.g. store the location at the start and the end for a moving point and
  use a linear approximation for the inner positions

- may be undefined
Nodes describing Compositions

- describe a linear sequence of submoves
- minimum 2 submoves have to exist
- holes in the deftime modeled by undefined basic moves
Periodic Movement

- represents a repeated submove
- has exactly one submove which must not be a periodic one
- stores the number of repetitions
The Root of the Tree

- all non-root nodes use relative intervals
- the root stores an anchor time as starting instant of the represented move
Example
Example

starts at 8:00 AM from A
Example

A starts at 8:00 AM from A

B

10 minutes

C
Example

starts at 8:00 AM from A

10 minutes

1 minute
Example

Example: A starts at 8:00 AM from A.

A → B: 10 minutes
B → C: 13 minutes
C:

1 minute
Example

A starts at 8:00 AM from A

B

10 minutes

1 minute

C

13 minutes

1 minute

starts at 8:00 AM from A
Example

A starts at 8:00 AM from A

10 minutes

1 minute

B

13 minutes

1 minute

C
Example

A starts at 8:00 AM from A

B

10 minutes

1 minute

C

13 minutes

1 minute

starts at 8:00 AM from A
Example

A starts at 8:00 AM from A

- 10 minutes to B
- 1 minute from B to C
- 13 minutes to C

C starts at 8:00 AM from A

1 minute
Example

A starts at 8:00 AM from A

A → B: 10 minutes
B → C: 1 minute
C → B: 13 minutes

starts at 8:00 AM from A
Example

Example

starts at 8:00 AM from A
repeated until 4:20 PM
Example

starts at 8:00 AM from A
repeated until 4:20 PM
remaining day and at the weekend the train stays at station A
T 2000-1-3-8:00

\[ R_2 \ 520 \]

\[ C_2 \]

\[ R_1 \ 5 \]

\[ L_8 \]

\[ C_1 \]

\[ R_0 \ 10 \]

\[ L_7 \]

\[ C_0 \]

\[ L_0 \]

\[ L_1 \]

\[ L_2 \]

\[ L_3 \]

\[ L_4 \]

\[ L_5 \]

\[ L_6 \]

\[ C_0 \]: a single run
\[ L_7 \]: staying at the night
\[ C_1 \]: a single working day
\[ L_8 \]: staying on weekend
\[ C_2 \]: a complete week
\[ R_2 \]: lifetime of the train
Detection of Repetitions in Movements

- problem: detection of (nested) repetitions in movements
- starts with a single composite move
- algorithm works bottom up
- searches for repetitions of growing length
Detection of Repetitions in Movements - Example

- analyse sequence 0 1 2 3 2 3 2 3 1 2 3 2 3 2 3
- each number represents an ID of a basic move

- construct a single composition
- no repetition of length 1
- repetition of length 2 (232323 and 232323)
Detection of Repetitions in Movements-Example

resulting tree is:

- repetition of length 2 (1R₁)
Detection of Repetitions in Movements - Example

resulting tree

```
C
  /  \
R2 2
  \
  /
C2
  /
  /
  /
1
  /
R1 3
  \
  /
  /
  /
  /
C1
  /
  /
  /
  /
  /
2 3
```

no further repetitions
SECONDO

- extensible database system
- open source project
- running on Windows, Linux, and Mac platforms
- project site:
  www.informatik.fernuni-hagen.de/seconde
- beside the SECONDO system also available:
  - some datasets
  - generators
  - converters
An algebra provides:

- new types
  - data structures for the new types (C++ classes)
  - some additional functions to make the new type available as attributes of a relation
  - functions for the connection with SECONDO

- new operations
  - type mapping
  - value mapping
  - selection function
  - operator instance
Periodic Algebra

- implemented types (partially derived from template classes)
  - `pmpoint`
  - `pmreal`
  - `pmbool`
  - `pmpoints`
- some operations
  - `atinstant: pm (x) × instant → x`
  - trajectory: `pmpoint → line`
  - distance: `pmpoint × point → pmreal`
Structure of the classes

root record

anchor
duration
topmove

Basic Moves $B$
Periodic Moves $P$
Composite Moves $C$
Composite Submoves $S$

separately stored arrays
Graphical User Interface

- connected via TCP with a SECONDO server
- objects are transmitted as nested lists
- extensible by viewers
- the HoeseViewer is able to display spatial and spatio temporal objects
- extensible by display classes
Periodic Moving Objects within the Javagui

- the HoeseViewer is used for displaying periodic moving objects
- set of classes for easy creating display classes for the HoeseViewer
- two classes are to implement
  - unit representation
    - reading from nested list representation
    - bounding box, interval
    - computation of the current state for a given instant
  - display class
    - only the interface to the HoeseViewer
- implemented Display classes
  - pmpoint (Label)
  - pmpoints
  - pmreal (externDisplay)
  - pmregion
Comparison with the flat model

• “flat model”
  • uses slices with anchored intervals
  • holes are not represented
  • repetitions are not exploited
  • implemented as another algebra module within SECONDO

• two datasets
  • generated data:
    underground trains of Berlin (the best case)
  • collected data:
    traces from a GPS receiver without any repetition (worst case)

• comparison criteria
  • storage size
  • used time for the trajectory and atinstant operation
## Results of the Comparison

### Trains

<table>
<thead>
<tr>
<th></th>
<th>Flat Model</th>
<th>Periodic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage Size (MB)</td>
<td>429.56</td>
<td>0.37</td>
</tr>
<tr>
<td>Trajectory (s)</td>
<td>149</td>
<td>0.25</td>
</tr>
<tr>
<td>Instant (s)</td>
<td>6.7</td>
<td>0.0033</td>
</tr>
</tbody>
</table>

### GPS-Traces

<table>
<thead>
<tr>
<th></th>
<th>Flat Model</th>
<th>Periodic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage Size (MB)</td>
<td>35.58</td>
<td>39.43</td>
</tr>
<tr>
<td>Trajectory (s)</td>
<td>11.4</td>
<td>11.2</td>
</tr>
<tr>
<td>Instant (s)</td>
<td>0.58</td>
<td>0.33</td>
</tr>
</tbody>
</table>
Conclusions

• model representing periodic moving objects
• algorithm for detecting repetitions in movements
• very good results if repetitions are present
• small overhead if no repetitions exist
thanks for your attention