Temporal and Spatio-Temporal Databases
(continuation of Spatial ...)
Robert M. Keller
Harvey Mudd College
April 2004

Possible Applications
- Personnel or medical records
- Reservation systems
- Archaeological records
- Calendars, planning, scheduling, configuration management
- Real-time process monitoring, data streaming

Breadth of Interest
- According to Zaniolo, et al., 1995:
  - It is hard to find applications that don't involve some form of temporal information.
  - Over 40 distinct temporal data models have been defined.

Temporal Concepts
- Represent time of events
- Represent time intervals
- Represent histories (= mapping from time interval to events or value)
- Represent periods of repeating events
- Sometimes only need sequences of events rather than times themselves:
  - Loosely think of this as "time".

“Valid-Temporal” Concepts
- Conventional database:
  - Tuple represents a fact
- Temporal database:
  - Tuple represents a fact that:
    - holds at a specified time (called the valid time), or
    - holds over an interval of time (called an invariant) over the interval (which can include the future as well as the past)
    - holds at some time during an interval

Sample Valid-Temporal Queries
- Was Mary the manager of the IT Department during the year 2002?
- Find all managers of the IT Department from 1/1/2000 to 12/31/2003.
- Find out who was promoted during the year 2003.
- What was the configuration of the system on March 12, 2001?
“Transaction-Temporal”
- Tuple can also show that time at which it was stored in the database.
- Note possible connection with multi-version DB.
- "Bi-temporal": both temporal and transaction temporal.
- Ancillary information: Who authorized the change?
- Might also want to show when info deleted, when a request to delete was made, etc.

Transaction-Temporal Queries = Introspective
- Queries that ask questions about changes to the database.
- When was the information that John was promoted to manager inserted?
- When was the fact that John is manager of the sales department deleted?
- When was the configuration of the system changed to include the TeraTuple 2000 disk farm?

Transaction Time
- Interval representation - DBMS maintained

<table>
<thead>
<tr>
<th>Movie</th>
<th>Actor</th>
<th>Transaction Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy Story</td>
<td>Tim</td>
<td>Jan/4/1999 now</td>
</tr>
<tr>
<td>Lion King</td>
<td>Mar</td>
<td>Jan/4/1999 now</td>
</tr>
<tr>
<td>True Lies</td>
<td>Arnold</td>
<td>Jul/2/2000 now</td>
</tr>
</tbody>
</table>

- On Jul/2/2000
  DELETE FROM Movie WHERE Actor='Arnuld';
  INSERT INTO Movie ('True Lies', 'Arnold');
- Schema evolution (e.g., Roddick and Snodgrass, TSQL2, 1995) Schema itself can change

Example: Stock Market DB
- Stock price changes throughout session.
- Price has 3 components: ask (by seller), bid (by buyers), sold
- Bid transaction includes amount buying, price, expiration, buyer, broker
- Ask transaction includes amount offered, price, expiration, seller, broker
- Sell transaction includes amount sold, price, buyer, seller, broker
- Other transactions:
  - Cancel transaction
  - Limit order transaction
  - Stop-loss order transaction
- Which are valid-time and which are transaction-time aspects?

Formal Time Models
- Continuous, Rational, Discrete, or Refinable
- Finite, singly-infinite, doubly-infinite
- Linear or Branching (Non-Determinism)
- Centralized or Distributed
- Qualitative or Quantitative

Example: TEMPOS
- Temporal Extension Models for Persistent Object Servers
- Dumas, et al., University of Grenoble
- Extends the Object Database Management Group's (ODMG) object model by enabling classes to be declared as temporal.
- Values are classified as either:
  - Temporal, or
  - Fleeting (only the most recent value kept)
- Recall that this is a precursor to JDO.
- Extends languages:
  - OCL — Temporal OCL
  - OQL — Temporal OQL
- Was implemented atop O2 (an OODBMS).
**Formal Modeling**

**Definition 1** (data model). A data model $M$ is a quadruplet $(D, Q, U, \mathcal{ML})$ composed of

- a set of database instances $D$,
- a set of legal query statements $Q$,
- a set of legal update statements $U$, and
- an evaluation function $\mathcal{ML}$.

Given an update statement $u \in U$, a query statement $q \in Q$, and a database instance $d \in D$, $\mathcal{ML}(d, q, u)$ yields a database instance, and $\mathcal{ML}(d, q, u)$ yields a valid query instance over data structure.

---

**TEMPOS Time Granularity Modeling**

For each pair of time units $(u_1, u_2)$, such that $u_1 < u_2$, two conversion functions are defined. One for expanding a granularity of the coarser unit $(u_2)$ into an interval of granules of the finer one $(u_1)$. The other for approximating granularity of $u_1$ by a granularity of $u_2$ when $u_2 < u_1$, as shown in Figure 1. Units $u_1$ and $u_2$ $(u_1 < u_2)$ are said to be scalable if the intervals of granules generated by $u_1$ are the same at all the same intervals.

---

**Histories**

Histories may be represented in several ways, readily by means of collections of timestamped values, termed *chronicles*. Among these representations some are useful for query expression, so that specific operators are defined on the History ADT allowing to represent a history in any of these forms. Concretely, a history may be represented by at least three kinds of chronicles:

- Instant-based representation: chronologically ordered sequence of instant-timestamped values, e.g., $\{[1.2, v_1], [2.1, v_1], [4.2, v_1], [5.2, v_2], [8.2, v_2], [10.2, v_3], [14.2, v_4]\}$.

- Interval-based representation: chronologically ordered, coalesced sequence of interval-timestamped values, e.g., $\{([1.2, 3.2], v_1), ([4.2, 5.2], v_1), ([8.2, 12.2], v_2), ([10.2, 14.2], v_3)\}$.

- Temporal sequence-based representation: set of distinct values timestamped by disjoint temporal sequences, e.g., $\{\{(1.2, 2.4, 9.3, v_1), (5.6, 7.2), (10.8, v_3)\}\}$, which are termed $\mathcal{D}$-chronicles.

---

**Operators on Histories**

The structure of operators on histories is captured on the microschema of the arguments of each operator.

---

**Temporal Types Class Diagram**

- Sequence
- Instant
- Interval
- Periodic
- Sequence of Instants
- Sequence of Coalesced Intervals

---

**Operators on Histories**

- Operators on histories
- Instant
- Interval
- Periodic
- Sequence of Instants
- Sequence of Coalesced Intervals
- Algebras of Duration
- Constraints of Instant
- Constraints of Interval
- Constraints of Periodic

---

**TEMPOS Time Granularity Modeling**

We adopt a discrete, linear and bounded time model in which time is structured in a multi-granular way [CRGS87, W80] by means of time units. A time unit is a partition of the time line into a set of coarse sets, each of which is then seen as an atomic granularity and every point of the time-line is approximated by that granularity which contains it. Thus, a time unit defines the precision at which time is observed. The granules of a time unit are numbered by natural integers; the order among these integers defines the notion of succession in time and the distance between them defines the notion of duration.
Example: Temporal Join

The inner temporal join of two histories $H_1$ and $H_2$ in a database where structural values are pairs obtained by combining "synchronous" values of $H_1$ and $H_2$ (i.e., values attached to the same instant). The outer temporal join $H_1 \bowtie H_2$ is similar to the corresponding inner temporal join, except that it attaches structural values of the form $(v, H_1, H_2)$ or $(v, H_2, H_1)$ to those instants where one of the segment histories defined while the other is not. Here, $H_1$ denotes the structural element of the structural domain type (e.g., $H$ for integers, $H$ for objects, etc.).

More precisely:

$$H_1 \bowtie H_2 = \{ (v, (x_1, x_2)) | (x_1, x_2) \in H_1 \cap (1, 2) \cap H_2 \}$$

preconditions: $\forall h \in H_1 \Rightarrow \forall h \in H_2$

Temporal Join Example
(from Zaniolo, et al. 1995)

<table>
<thead>
<tr>
<th>Employee1</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Salary</td>
<td>Position</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td>60k</td>
<td>assistant</td>
<td>1993-06-01</td>
<td></td>
</tr>
<tr>
<td>Mary</td>
<td>70k</td>
<td>provost</td>
<td>1993-10-01</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee2</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Date</td>
<td>Position</td>
<td>Salary</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td>1994-02-01</td>
<td>provost</td>
<td>65k</td>
<td></td>
</tr>
<tr>
<td>Mary</td>
<td>1995-01-01</td>
<td>full professor</td>
<td>80k</td>
<td></td>
</tr>
</tbody>
</table>

Temporal Join in Plain SQL
(from Zaniolo, et al. 1995)

```sql
SELECT Employee1.Name, Employee1.Salary, Employee1.Position
FROM Employee1
JOIN Employee2
ON Employee1.Name = Employee2.Name
WHERE Employee1.Date BETWEEN '1993-01-01' AND '1993-10-01'
```

Temporal Join in TSQL2
(from Zaniolo, et al. 1995)

```sql
SELECT Employee1.Name, Employee1.Salary, Employee1.Position
FROM Employee1
JOIN Employee2
ON Employee1.Name = Employee2.Name
WHERE Employee1.Date BETWEEN '1993-01-01' AND '1993-10-01'
```

Example Spatio-Temporal Applications

- Record the historical evolution of a landform (such as a volcanic region, or global earthquakes)
- Understand weather data, or flow of groundwater and contaminants
- Chart the propagation of spoken or written languages
- Trace the migration of a population of animals
- Anticipate the spread of an epidemic
- MRI or PET imaging
- Track terrorists
- Choreography, any motion picture or animation, video games

A well-sampling application that uses wireless sensor networks
(Ganeson, et al., UCLA)
General Desiderata for Spatio-Temporal Models

- Should be upward compatible from classical methods (i.e. should subsume them)
- Deal with interactions of time and space, not just the two independently (e.g. space-time constraints)
- Expressive, yet easy to use
- Compatible with existing frameworks

Possible Test-beds

- Any of a variety of 3D modeling tools, such as Maya, AutoCAD, GeoToolKit

Spatio-temporal Evolution

- Assume that time is another dimension, use a 3D R-tree

Indexing using R-trees

Additional Directions

- Constraint Logic Programming
- Machine learning
- Data mining

Assertion: The WWW is a Database

- What aspects are:
  - relational?
  - object-oriented?
  - temporal?
Query Language

- URL is, in effect, a primitive query
  - identifies single, whole document as the answer
- Some information exists, but cannot be queried
  - Requires natural language text understanding

Other Ways to Access Data over WWW

- CGI/HTTP
- Applets
- Servlets (= server-side applets)
- .NET

HTTP (CGI) Gateway

- HTML forms, limited GUI
- result formatting
- HTTP/database protocol mismatch
  - "stateless" protocol
  - single query transactions

Java Applets

- "stateful"
- fancy GUI
- robust, secure, platform-independent
- database API - JDBC
WWW Data Structure

- WWW is data repository
- unstructured (semi-structured)
  - HTML tags
  - XML tags
  - WWW graph (graph of hyperlinks)
- primitive query
  - URL identifies single, whole document

Consistency

- redundant information
  - plenty of it: hardly normalized
- incorrect information
  - 5% of links are broken
  - little certification, if any
- triggers/assertions/constraints
  - guarantee consistency

An Issue for Investigation

- Does the WWW support transaction-time temporality? If not, can it be extended to do so?