Temporal Databases

Outline

- Spatial Databases
  - Indexing, Query processing
- Temporal Databases
- Spatio-temporal
- ....
Temporal DBs – Motivation

- Conventional databases represent the state of an enterprise at a single moment of time
- Many applications need information about the past
  - Financial (payroll)
  - Medical (patient history)
  - Government
- Temporal DBs: a system that manages time varying data

Comparison

- Conventional DBs:
  - Evolve through transactions from one state to the next
  - Changes are viewed as modifications to the state
  - No information about the past
  - Snapshot of the enterprise
- Temporal DBs:
  - Maintain historical information
  - Changes are viewed as additions to the information stored in the database
  - Incorporate notion of time in the system
  - Efficient access to past states
Temporal Databases

- Temporal Data Models: extension of relational model by adding temporal attributes to each relation
- Temporal Query Languages: TQUEL, SQL3
- Temporal Indexing Methods and Query Processing

Taxonomy of time

- Transaction time databases
  - Transaction time is the time when a fact is stored in the database
- Valid time databases:
  - Valid time is the time that a fact becomes effective in reality
- Bi-temporal databases:
  - Support both notions of time
Example

- Sales example: data about sales are stored at the end of the day
- Transaction time is different than valid time
- Valid time can refer to the future also!
  - Credit card: 03/01-04/06

Transaction Time DBs

- Time evolves discretely, usually is associated with the transaction number:
  \[ T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \ldots \]
- A record R is extended with an interval \([t\text{.start}, t\text{.end})\). When we insert an object at \(t1\) the temporal attributes are updated \(\rightarrow [t1, \text{now})\)
- Updates can be made only to the current state!
  - Past cannot be changed
  - “Rollback” characteristics
Transaction Time DBs

- Deletion is logical (never physical deletions!)
  - When an object is deleted at t2, its temporal attribute changes from [t1, now) → [t1, t.t2) (lifetime)
  - Object is “alive” from insertion to deletion time, ex. t1 to t2. If “now” then the object is still alive

<table>
<thead>
<tr>
<th>eid</th>
<th>salary</th>
<th>start</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20K</td>
<td>9/93</td>
<td>10/94</td>
</tr>
<tr>
<td>20</td>
<td>50K</td>
<td>4/94</td>
<td>*</td>
</tr>
<tr>
<td>33</td>
<td>30K</td>
<td>5/94</td>
<td>6/95</td>
</tr>
<tr>
<td>10</td>
<td>50K</td>
<td>1/95</td>
<td>*</td>
</tr>
</tbody>
</table>

Database evolves through insertions and deletions
Transaction Time DBs

- Requirements for index methods:
  - Store past logical states
  - Support addition/deletion/modification changes on the objects of the current state
  - Efficiently access and query any database state

Transaction Time DBs

- Queries:
  - Timestamp (timeslice) queries: ex. “Give me all employees at 05/94”
  - Range-timeslice: “Find all employees with id between 100 and 200 that worked in the company on 05/94”
  - Interval (period) queries: “Find all employees with id in [100,200] from 05/94 to 06/96”
Valid Time DBs

- Time evolves continuously
- Each object is a line segment representing its time span (e.g., Credit card valid time)
- Support full operations on interval data:
  - Deletion at any time
  - Insertion at any time
  - Value change (modification) at any time (no ordering)

Deletion is physical:

- No way to know about the previous states of intervals
- The notion of “future”, “present” and “past” is relative to a certain timestamp $t$
Valid Time DBs

The reality “best know now !”

Valid Time DBs

- Requirements for an Index method:
  - Store the latest collection of interval-objects
  - Support add/del/mod changes to this collection
  - Efficiently query the intervals in the collection
    - Timestamp query
    - Interval (period) query
Bitemporal DBs

- A transaction-time Database, but each record is an interval (plus the other attributes of the record)
- Keeping the evolution of a dynamic collection of interval-objects
- At each timestamp, it is a valid time database
Bitemporal DBs

- Requirements for access methods:
  - Store past/logical states of collections of objects
  - Support add/del/mod of interval objects of the current logical state
  - Efficient query answering

Temporal Indexing

- Straight-forward approaches:
  - B+-tree and R-tree
  - Problems?
- Transaction time:
  - Snapshot Index, TSB-tree, MVB-tree, MVAS
- Valid time:
  - Interval structures: Segment tree, even R-tree
- Bitemporal:
  - Bitemporal R-tree
Temporal Indexing

- Lower bound on answering timeslice and range-timeslice queries:
  - Space $O(n/B)$, search $O(\log_B n + s/B)$
  - $n$: number of changes, $s$: answer size, $B$: page capacity