Versioned Data Management: Overview

Summary of Lecture Slides by Alexandre de Spindler (2014)

Aspects of Versioned Data Management

Granularity

- Application Data, Database
- Object Set (e.g. Instances of a Class, Tuples in a Relation, Collections)
- Object
- Attributes

Version Organisations

<table>
<thead>
<tr>
<th>Set</th>
<th>List</th>
<th>Tree</th>
<th>DAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>$v_1$</td>
<td>$v_1$</td>
<td>$v_1$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$v_2$</td>
<td>$v_2$</td>
<td>$v_2$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$v_3$</td>
<td>$v_3$</td>
<td>$v_3$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>$v_4$</td>
<td>$v_4$</td>
<td>$v_4$</td>
</tr>
<tr>
<td>$v_5$</td>
<td>$v_5$</td>
<td>$v_5$</td>
<td>$v_5$</td>
</tr>
</tbody>
</table>

Examples of applications:

- **Set**
  - Different languages, geographic regions, user groups, licence models
- **List**
  - Revisions of Software (e.g. “Commits” of Code)
  - Temporal Succession of Values (e.g. number of employees in 2011, 2012 and 2013)
- **Tree** (List + Branches)
  - Revisions of Software (combination of Set and List)
  - Different sources for number of employees (e.g. $v_1$: value found for 2011, $v_2$: value found for 2012 using XING, $v_3$: value found for 2012 using LinkedIn, $v_4$: value found for 2013 using LinkedIn)
- **DAG** (Tree + Merges)
References

- Specific reference
  References specific version directly
- Generic reference
  Refers to entire object (or equivalent depending on granularity). Generic references need to be
dereferences to a specific version when traversed, e.g. latest version, default language.
  - Parallel versions (Tree, DAG) need “main” branch
  - Sequential versions (List) and Sets need “active” version

Note: unless versions are an explicit and integral part of the application domain, generic references are
required. Ideally, the domain model does not include any notion of versions. Realistically, the notion of
version should be hidden as far as possible, e.g. using generic references.

Version Representation

- Complete versions of objects (depending on granularity, this typically implies some redundancy)
- One full version + deltas
  - State-based deltas, e.g. attributes that differ
  - Operation-based deltas, e.g. update operation
  - In both cases, backward vs. forward deltas

Note: Consider space vs. time performance requirements

Versioning Operations

- Create new version
- Branch a parallel version
- Merge to parallel versions
- Delete a version (often not supported)

Interaction Models

- Automatic versioning, transparent to user (as long as they are pleased with default versions)
- Explicit user operations, typically high-level (e.g. check-out, commit, update)
- Implicit vs. explicit query expansion, possibly involving generic references
Temporal Databases

Typical Applications

- What was the value of attribute R.a of tuple t in relation R three days ago?
- How has the value of attribute R.a of tuple t in relation R evolved over the last three days?
- The value of attribute R.a of tuple t in relation R as it was three days ago should be changed to a new value.
- The future value of attribute R.a of tuple t in relation R should be set to a particular value.

Aspects of Time

- Physical time (a.k.a. transaction time)
  Captures when values were stored or manipulated in the database
- Logical time (a.k.a. valid time)
  Expresses when values existed in the real world
- Bitemporal
  Combination of physical and logical time
- User-defined time
  All aspects not covered by physical and logical time

Classification of temporal databases

- Static or snapshot database
  Conventional database, does not manage temporal data
- Static roll-back database
  Keeps track of physical time (e.g. Subversion). Not retro- or proactive updates, no correction of errors
- Historical database
  Keeps track of logical time. Changes can be made to previous values.
- Temporal database
  Keeps track of physical and logical time.

Representation of Temporal Data (RM-Approaches)

- Object Versioning
  e.g. employees(id, office, salary, t-from, t-till)
  What about primary key?
- Attribute Versioning
  e.g. employees(id, office, office-from, office-to, salary, salary-from, salary-to)
  Could be simplified using non-first-normal-form systems
- Bitemporal conceptual data model (BCDM)
  Tuple versioning implemented using four additional columns (physical and logical, from and to)
- Homogeneity vs. Heterogeneity
  - Homogeneous if all attributes of an object change at the same points in time
  - Heterogenous if attributes may change at different points in time, e.g. salary varies differently to office.
- Temporal anomalies
  - Vertical anomaly if homogenous (multiple tuples represent single entity)
  - Horizontal anomaly if heterogeneous (entity spread over multiple relations)
Storage Models

- Three-dimensional tables (sequence of relations, data cube)
- Two-level structures
  - Primary store with current/default versions used for non-temporal queries
  - History store with remaining versions
References & Misc

- Lecture slides for Object Databases-Course (M. Norrie, M. Grossniklaus, A. de Spindler),
  http://www.globis.ethz.ch/education/oodb
- C. Jensen and M. Soo and R. Snodgrass, Unifying Temporal Data Models via a Conceptual Model,
  https://www.cs.arizona.edu/~rts/pubs/ISDec94.pdf
- Oracle Flashback,
  http://docs.oracle.com/cd/B28359_01/appdev.111/b28424/adfns_flashback.htm#g1026131