Comparision of Access Methods for Time-Evolving Data

Betty Salzberg and Vassilis J. Tsotras

Abhishek Srivastava
Student ID: 861307778

February 28, 2017
CS 236, Winter 2017

The problem:

The paper discusses the need of temporal data since conventional database system is unable to store evolved states of databases from one logical state to other for managing time varying data.

The contribution:

The authors presents a survey comparing different indexing mechanism proposed of *Transaction Time*, *Valid Time* and *Bitemporal* databases for efficient access.

The method:

Transaction Time stores historical sequences of a database. *Valid Time* stores the present snapshot of a database at a specific time. *Bitemporal* is combines features from both databases. Each tuple in general have a time variant attribute can be represented by *start_time* and *end_time*.

Performance of queries is measured for worst case using three kind of transaction time queries: *pure-key*, *pure-timeslice* and *range-timeslice*.

Transaction-Time methods can be further classified into:

Key-Only Methods: In these methods data is organized/clustered by its key value which make pure-key queries efficiently. Some of the methods are *Reverse Chaining* in which previous version of given key are chained together in reverse chronological order, *Accession Lists* which clusters a given key together based on its timestamp value, *Time Sequence Arrays* which uses 2-D array to store value of key x at time y which makes query time minimal and *C-Lists* similar to accession lists but provide different access and maintenance mechanisms.

Time-Only Methods: These methods use some form of maintaining "history logs" which is used as a indexing parameter for efficient pure-timeslice queries. *Append-Only Tree* is hybrid of ISAM and B+ tree indexing to implement multiway search, *Time Index* based on B-tree and does indexing on time when any write/update or delete operation is performed, *Differential File Approach* does not create index but stores relational log with sorted timestamp changes in *(time,key,op)* format, *Checkpoint Index* which indexes on the checkpoints periodically on evolving relation, *Archivable Time Index* is improved version of checkpoint index by making checkpoints un-periodic, *Snapshot Index* uses three data structure to provide I/O optimal solution: multilevel index, multilinked structure and hashing function and *Windows method* partitions data into windows and perform indexing.

Time-Key Methods: Data is clustered both by key and time. *R-Tree-Based Methods* old and current data is separated by using *vaccum cleaner* process, indexing in done over both key and time for current data,*B-Tree Based Methods* is also a persistent database which follow same rule of separation, *Write-Once B-Tree*, *Time Split B-Tree*, *Multiversion B-Tree*, *Multiversion Access Structure* , *Exodus and Overlapping B+ Trees* and *Multiattribute Indexes* are some of its example.

Valid Time indexing methods consist of *Metablock Tree*, *External Segment Tree*, *External Interval Tree* and *MAP21* on dynamic collection of interval data.

Bitemporal data consists of snapshots over time and methods used to index these are *M-IVTT*, *Bitemporal Interval Tree* and *Bitemporal R-Tree*.

Paper also discusses issues such as index pagination, data clustering, efficient migration of data, lower bounds on I/O complexity and Optimal design/requirements of queries.

Comments:

Paper covers multiple methods for temporal data to match different type of needs but some of the shortcomings were: No comparison among all the methods for same kind of data and "Pathological" cost evaluation for some mechanisms.