
Exploiting Versions for Online Warehouse Maintenance in MOLAP Servers

Heum-Geun Kang and Chin-Wan Chung
KAIST
VLDB 2002

Organizations

- Introduction
- Multi-Dimensional Arrays(MDAs) for MOLAP
- Multi-Version Concurrency Control for Data Warehouses(MVCC-DW)
- Experiments
- Conclusion



Introduction

- Data warehouses
 - Enable users to make better and fast decisions
 - Collect information from several data sources
 - Support online analytical processing
- OLAP
 - Multi-dimensional OLAP(MOLAP)
 - Relational OLAP(ROLAP)

Introduction

- Query Transaction
 - Sequence of interactive queries
 - Queries tend to be complex and involve large volumes of data
- Maintenance transaction
 - Gather changes to the source data and propagate the changes to the warehouse data
 - Executed periodically
- The differences between OLAP and OLTP
 - Transaction execution time
 - The number of update transactions(maintenance transactions)
 - Volume of data to be accessed

Introduction

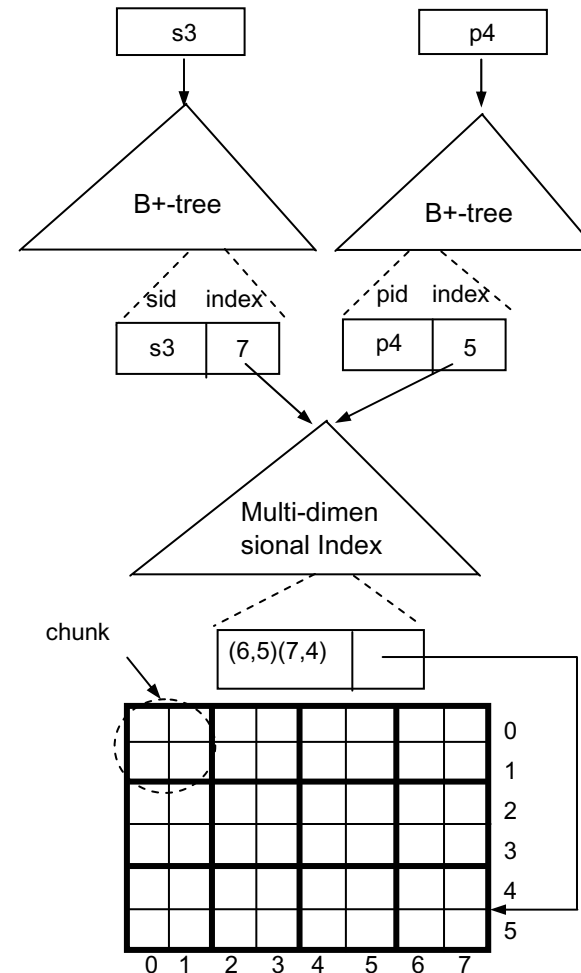
- It has been known that the CC mechanisms for OLTP systems are not adequate for OLAP systems
- A naive method
 - Not to run queries during the maintenance time
 - As corporations become globalized, the OLAP systems should be able to respond to the queries submitted by users in multiple time zones

Contribution

- Propose MVCC-DW suited for data warehouses managed by MOLAP servers
- Features of the MVCC-DW
 - Non Blocking
 - No Lock
- Prove the correctness of MVCC-DW
- Implement the MVCC-DW mechanism
- Show the MVCC-DW mechanism works efficiently

MDAs for MOLAP

- A set of B-trees
 - Map dimension values to array index values
 - One for each dimension
- A multi-dimensional index
 - Maps a sequence of array index values to a chunk
 - A chunk is a small multi-dimensional array
- A chunked file
 - Stores a set of chunks rather than a large array



MVCC-DW

- Motivation and Idea
 - Locking mechanisms result in a high blocking rate
 - Optimistic concurrency control mechanisms can have a high abort rate of long transactions
 - Our basic idea is to use a version mechanism
 - A chunk instead of a cell is used as the unit of version control
 - We devise a new access method which supports the versioning concept

MVCC-DW(Revision)

- A revision is a snapshot of the data warehouse
- State
 - Active : being changed
 - There is at most one at a time
 - Frozen : is not changed anymore
- Current : the most recently frozen revision
- Oldest : the least recently frozen revision
- Every revision is assigned a revision number



MVCC-DW(Revisions and Transactions)

- A transaction uses only one revision throughout the lifetime

Who	When	Operation
maintenance transaction	begin	create an active revision
maintenance transaction	end	freeze the active revision
query transaction	begin	open the current revision
query transaction	end	close the opened revision

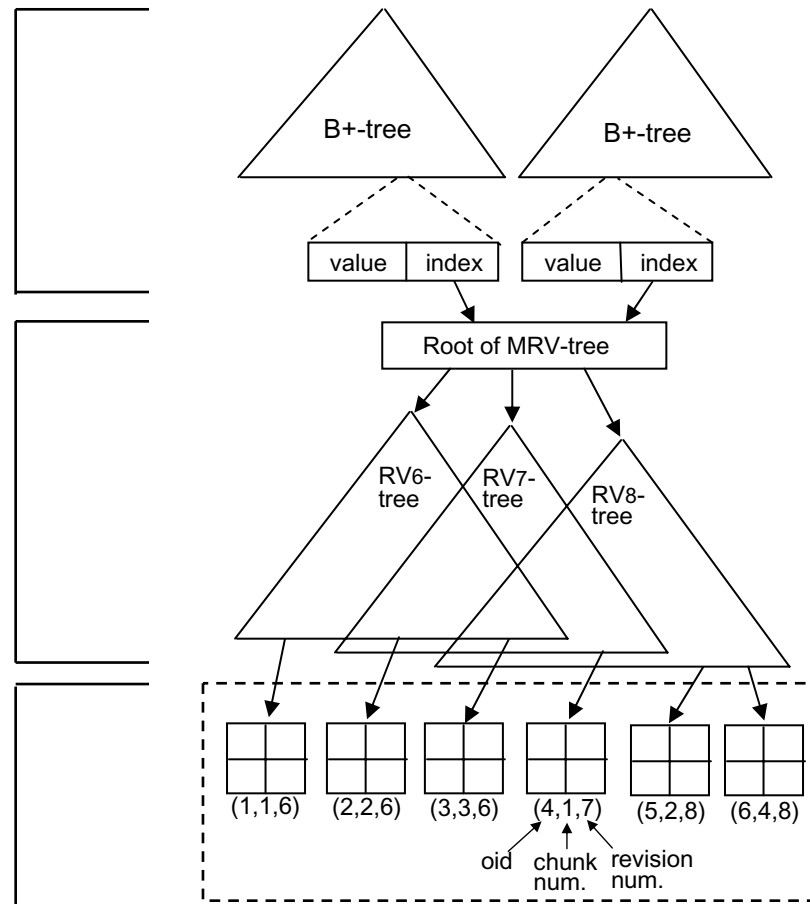


MVCC-DW(Arch. and Data Structures)

A set of B+trees

Multi-ReVision(MRV) –tree :
manages information about
revisions

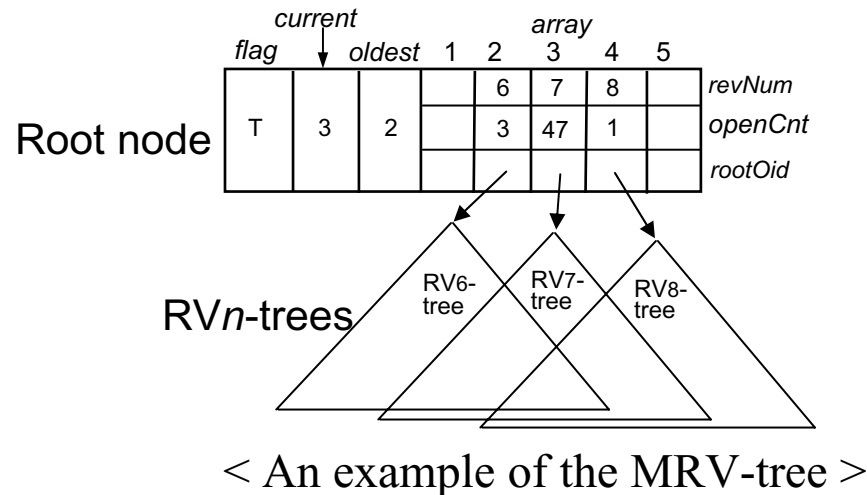
Chunked File



< The architecture of multi-versioned 2-dimensional array >

MVCC-DW(Arch. and Data Structures)

- MRV-tree
 - Consists of a root node and a sequence of ReVision_n(RV_n)-trees
 - Adjacent RV_n-trees share nodes if the nodes are not changed



MVCC-DW(Arch. and Data Structures)

- RV_n -tree
 - Manages the chunks contained in a revision
 - Maps a sequence of index values to a chunk
 - Each node has a revision number of a revision that was active at node creation time

Directory Node

rNum	C_1, MBR_1	C_2, MBR_2	C_3, MBR_3	...
------	--------------	--------------	--------------	-----

(rNum : revision number, C_i : address of child node i)

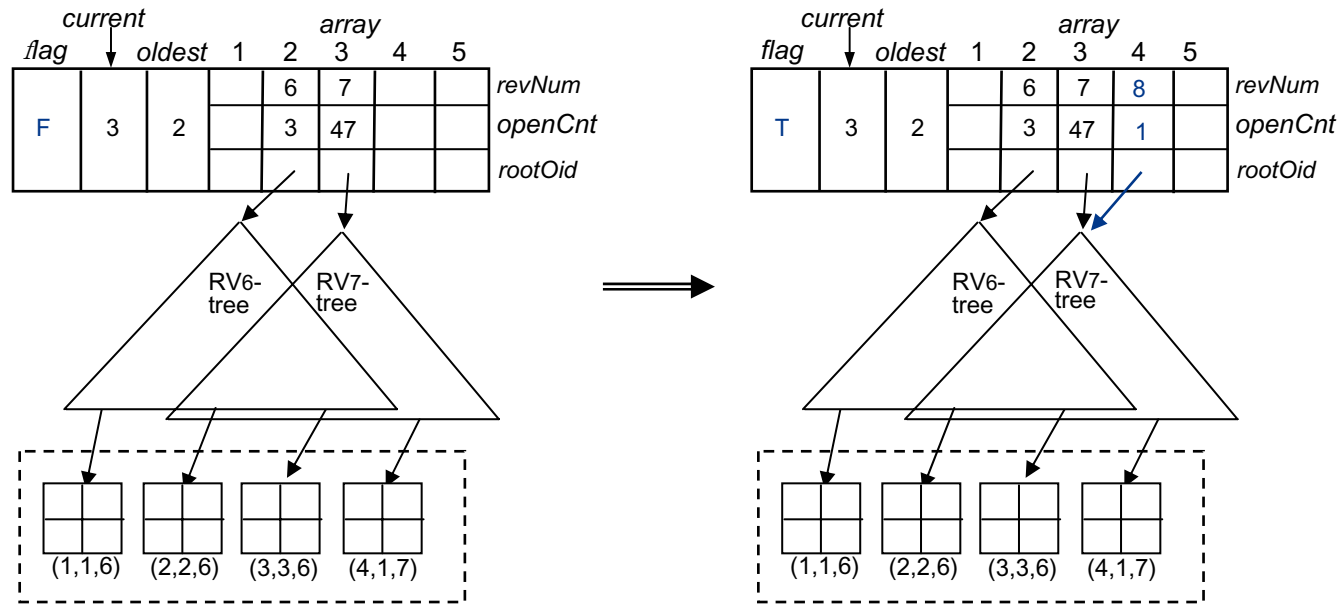
Leaf Node

rNum	O_1, MBR_1	O_2, MBR_2	O_3, MBR_3	...
------	--------------	--------------	--------------	-----

(rNum : revision number, O_i : id of object storing chunk i)

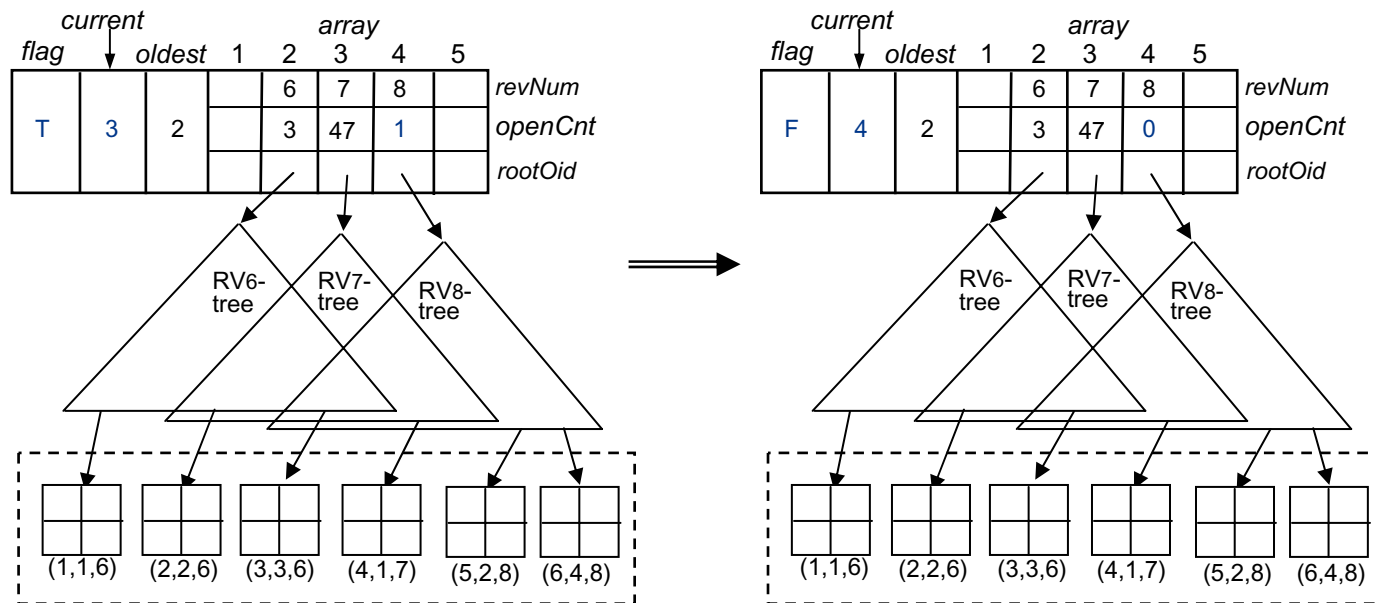
MVCC-DW(CreateRevision)

- Creates a revision and sets the revision to active



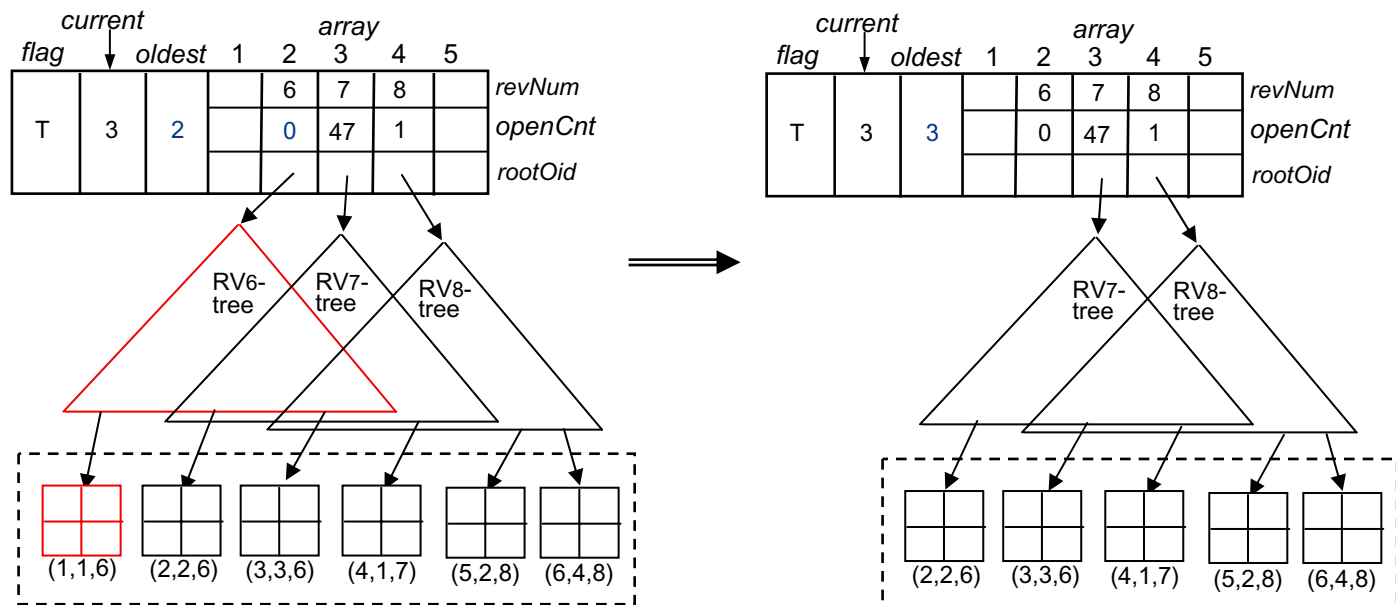
MVCC-DW(FreezeRevision)

- Closes the active revision and makes the revision frozen so that query transactions can retrieve the updated data in the revision



MVCC-DW(GarbageCollection)

- The oldest revision that is not the current revision can be released
- Since adjacent revisions share some nodes and chunks, a care is required



MVCC-DW(Correctness)

- Lemma 1 : Let T_i use the revision n and T_j use the revision m where $n < m$. There is no edge from T_j to T_i in a serialization graph
- Lemma 2 : There is no cycle between transactions using the same revision
- Theorem : An SG(H) for a history H produced by MVCC-DW is acyclic.

MVCC-DW(Clustering of cells)

- The number of chunks to be versioned has an impact on the performance of MVCC-DW
- The whole chunk is versioned even if a cell in the chunk is updated
- It is desirable that all cells in a chunk have the same time dimension value

Experiments

- Data set : APB Benchmark
 - Dimensions : customer, product, channel, time
 - Composed of historical data and incremental data
 - The number of valid cells in the data cube constructed from the historical data : 21,000,000
- Prototype
 - Built by modifying the Shore storage manager. Particularly, R*-tree

Experiments

	Customer	Product	Channel	Time
Size A	10	10	10	1
Size B	20	20	20	1
Size C	30	30	30	1
Size D	40	40	40	1
Size E	50	50	50	1

< Chunk Sizes >



Experimental Results

Table 3: Status of the RV₀-tree and the chunked file after the historical data is loaded

Chunk Size	RV ₀ -tree Nodes			Num. of chunks	Valid cells in chunks		
	3	2	1		Min.	Max.	Avg.
A	1	103	13,522	1,603,000	2	765	13
B	1	40	5,097	600,525	2	2,669	36
C	1	17	2,381	281,100	2	5,304	77
D	1	9	1,318	156,250	2	8,669	139
E	1	7	862	100,000	10	13,205	218

Table 4: Status of the RV₁-tree and the chunked file after the incremental data is loaded

Chunk Size	Num. of RV ₁ -tree nodes				Num. of Chunks	
	revision 0	revision 1			revision 0	revision 1
	1	3	2	1		
A	7,419	1	108	6,652	1,474,674	192,360
B	2,944	1	42	2,368	552,418	72,025
C	1,477	1	18	998	258,612	33,732
D	812	1	9	553	143,750	18,750
E	550	1	8	350	92,000	12,000

Conclusion

- Conventional concurrency control mechanisms are not adequate for a data warehouse environment
- Proposed a multi-version concurrency control mechanism, MVCC-DW, that exploits versions for online data warehouse maintenance in MOLAP servers
- Demonstrated the efficiency of MVCC-DW
 - The number of versioned chunks is small