An Experimental Evaluation of In-Memory Multi-Version Concurrency Control

Yingjun Wu, Joy Arulraj, Jiexi Lin, Ran Xian, Andrew Pavlo

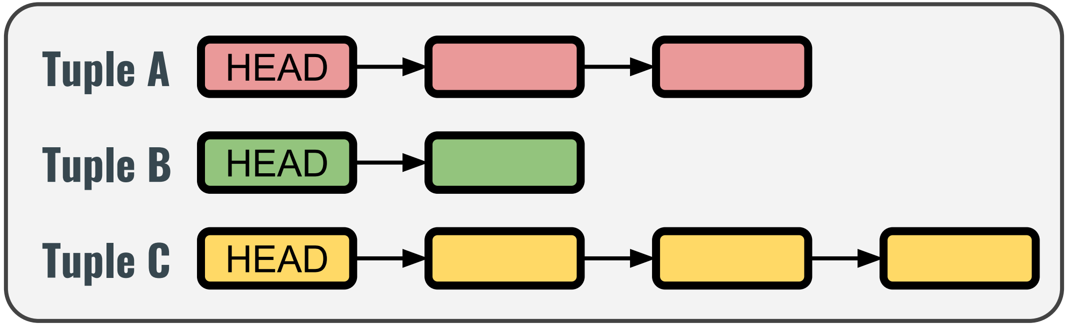
NUS National University of Singapore

Carnegie Mellon University

CARNEGIE MELLON DATABASE GROUP

MULTI-VERSION CONCURRENCY CONTROL

- ❑ Avoid read-write conflicts
- ❑ Support time-travel queries
- ❑ Enable snapshot isolation



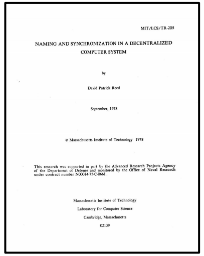
VERSION CHAINS

A BRIEF HISTORY OF MVCC

1979: FIRST MENTION

DAVID

REED



MIT/LCS/TR-205

NAMING AND SYNCHRONIZATION IN A DECENTRALIZED
COMPUTER SYSTEM

by

David Patrick Reed

September, 1978

© Massachusetts Institute of Technology 1978

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract number N00014-75-C-0661.

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts
02139

Naming and synchronization in a decentralized computer system
Ph.D. Thesis, 1979

A BRIEF HISTORY OF MVCC

- **1979:** FIRST MENTION
- **1981:** FIRST IMPLEMENTATION

InterBase/Firebird

A BRIEF HISTORY OF MVCC

- **1979: FIRST MENTION**
- **1981: FIRST IMPLEMENTATION**
- **1984:** Oracle
- **1985:** Postgres

A BRIEF HISTORY OF MVCC

- **1979: FIRST MENTION**
- **1981: FIRST IMPLEMENTATION**
- **1984:** Oracle
- **1985:** Postgres
- **2001:** MySQL-InnoDB

A BRIEF HISTORY OF MVCC

- **1979: FIRST MENTION**
- **1981: FIRST IMPLEMENTATION**
- **1984:** Oracle
- **1985:** Postgres
- **2001:** MySQL-InnoDB
- **2010-2017:** Hyrise, Hekaton, MemSQL, SAP HANA, NuoDB, HyPer...

A BRIEF HISTORY OF MVCC

ORACLE
HEKATON
Microsoft SQL Server
HYRISE
MySQL
NUODB
HyPer
PostgreSQL
SAP HANA
MEMSQL

Search for the best MVCC scheme for multi-core main-memory DBMSs

DESIGN DECISIONS OF MVCC

Concurrency Control Protocol

Version Storage

Garbage Collection

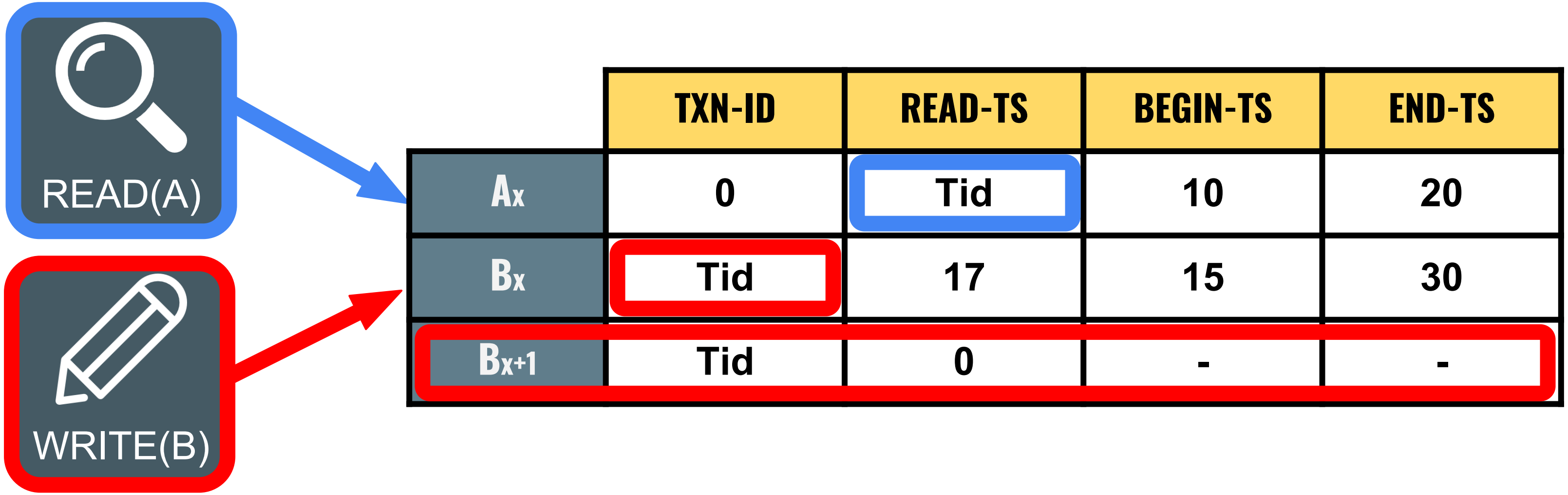
Index Management

CONCURRENCY CONTROL PROTOCOL

SCHEME	DBMS
Timestamp Ordering (MVTO)	N/A
Optimistic Concurrency Control (MVOCC)	HYRISE HEKATON Microsoft SQL Server HyPer MEMSQL
Two-phase Locking (MV2PL)	ORACLE MySQL SAP HANA NUODB
Serialization Certifier	PostgreSQL

CONCURRENCY CONTROL PROTOCOL

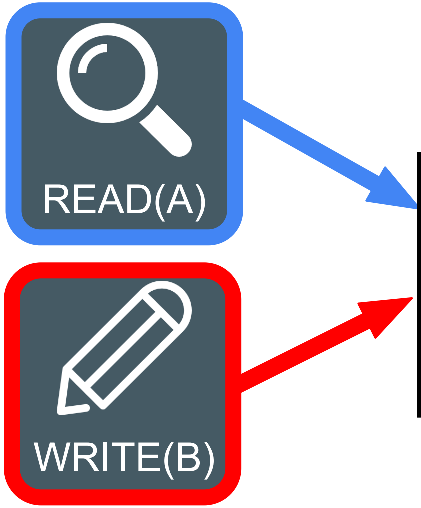
- ❑ **Approach #1: Timestamp Ordering (MVTO)**
  - ❑ The DBMS assigns transactions timestamps that determine serial order.



	TXN-ID	READ-TS	BEGIN-TS	END-TS
A_x	0	Tid	10	20
B_x	Tid	17	15	30
B_{x+1}	Tid	0	-	-

CONCURRENCY CONTROL PROTOCOL

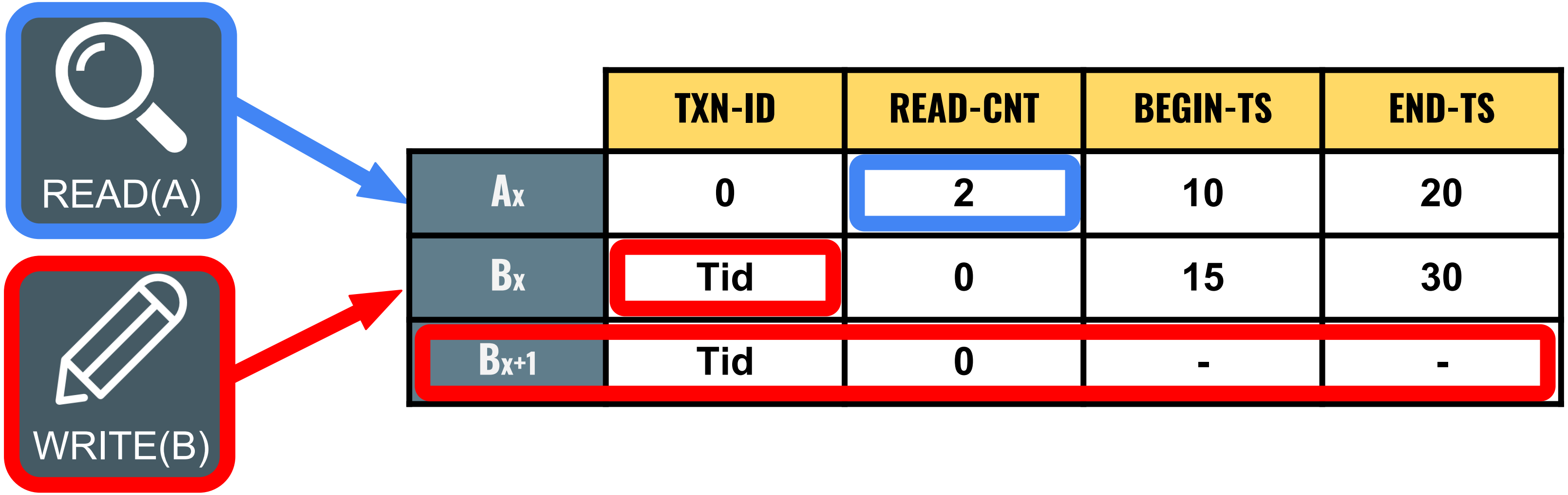
- ❑ **Approach #3: Optimistic Concurrency Control (MVOCC)**
  - ❑ Transactions optimistically access physical versions before validating the read-set consistency.



	TXN-ID	BEGIN-TS	END-TS
A_x	0	10	20
B_x	Tid	15	30
B_{x+1}	Tid	-	-

CONCURRENCY CONTROL PROTOCOL

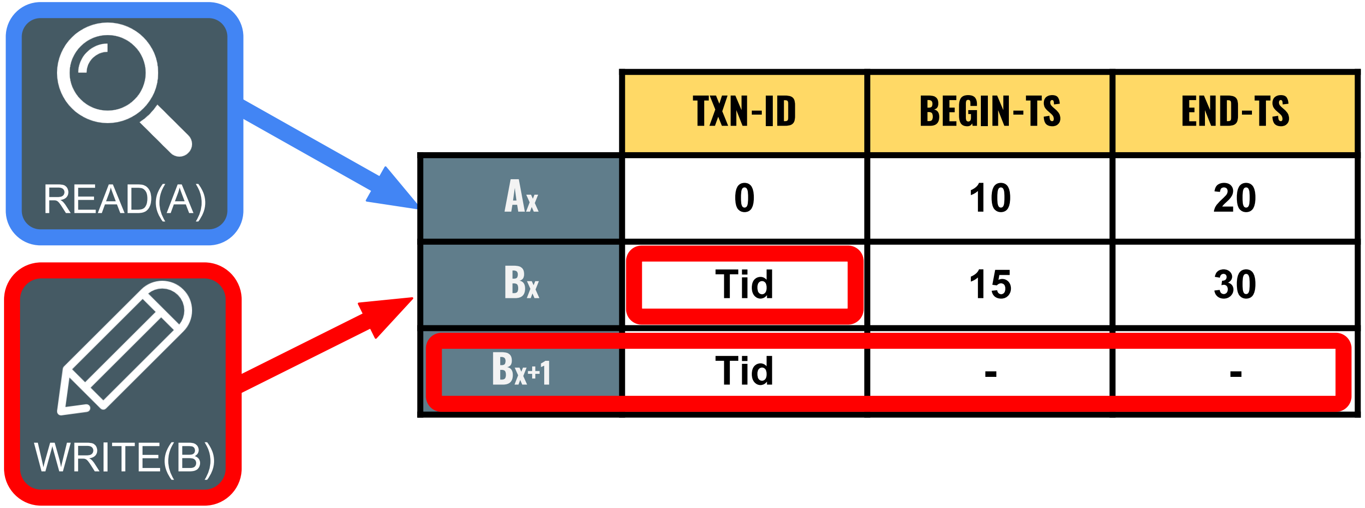
- ❑ **Approach #3: Two-Phase Locking (MV2PL)**
  - ❑ Transactions acquire appropriate lock on physical version before they can read/write a logical tuple.



	TXN-ID	READ-CNT	BEGIN-TS	END-TS
A_x	0	2	10	20
B_x	Tid	0	15	30
B_{x+1}	Tid	0	-	-

CONCURRENCY CONTROL PROTOCOL

- **Approach #4: Serialization Certifier**
  - The DBMS maintains a serialization graph for detecting and removing "dangerous structures" formed by concurrent transactions.



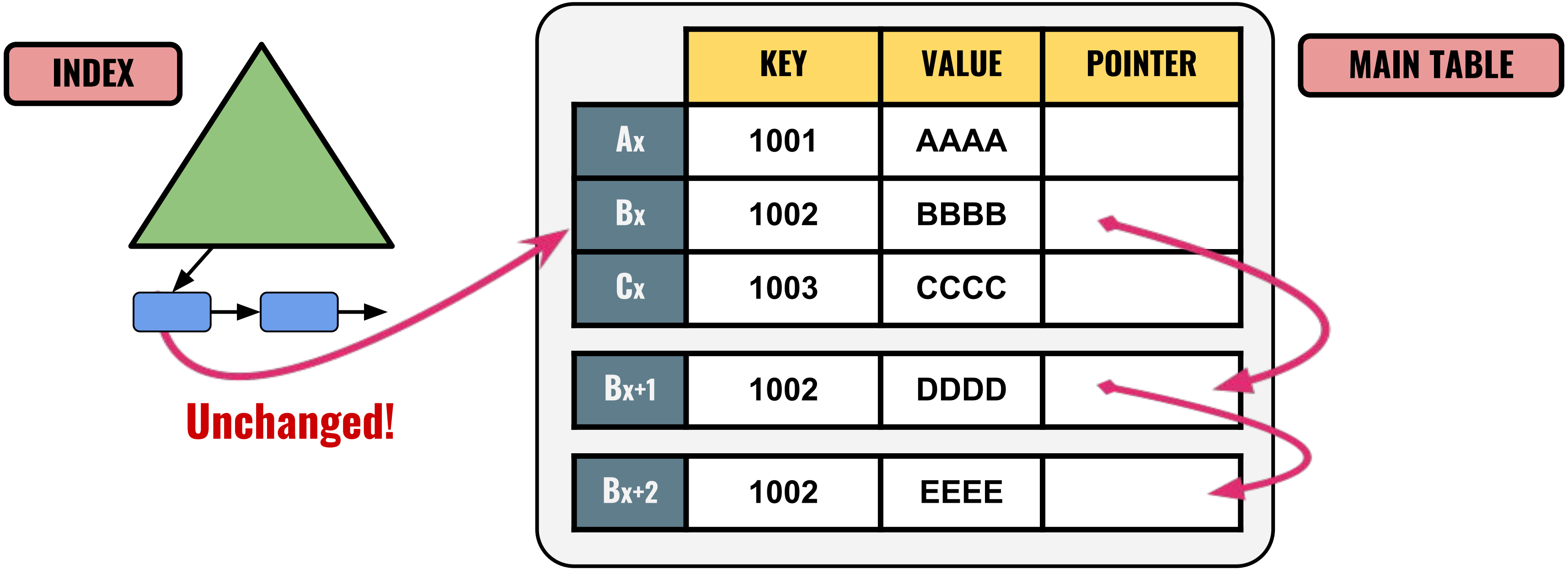
	TXN-ID	BEGIN-TS	END-TS
A_x	0	10	20
B_x	Tid	15	30
B_{x+1}	Tid	-	-

VERSION STORAGE

SCHEME	DBMS
Append-Only	PostgreSQL HYRISE HEKATON Microsoft SQL Server NUODB MEMSQL
Time-Travel	SAP HANA
Delta	ORACLE MySQL HyPer

VERSION STORAGE

- Approach #1: Append-Only Storage (Oldest-to-Newest)
  - New versions are appended to the same table space.



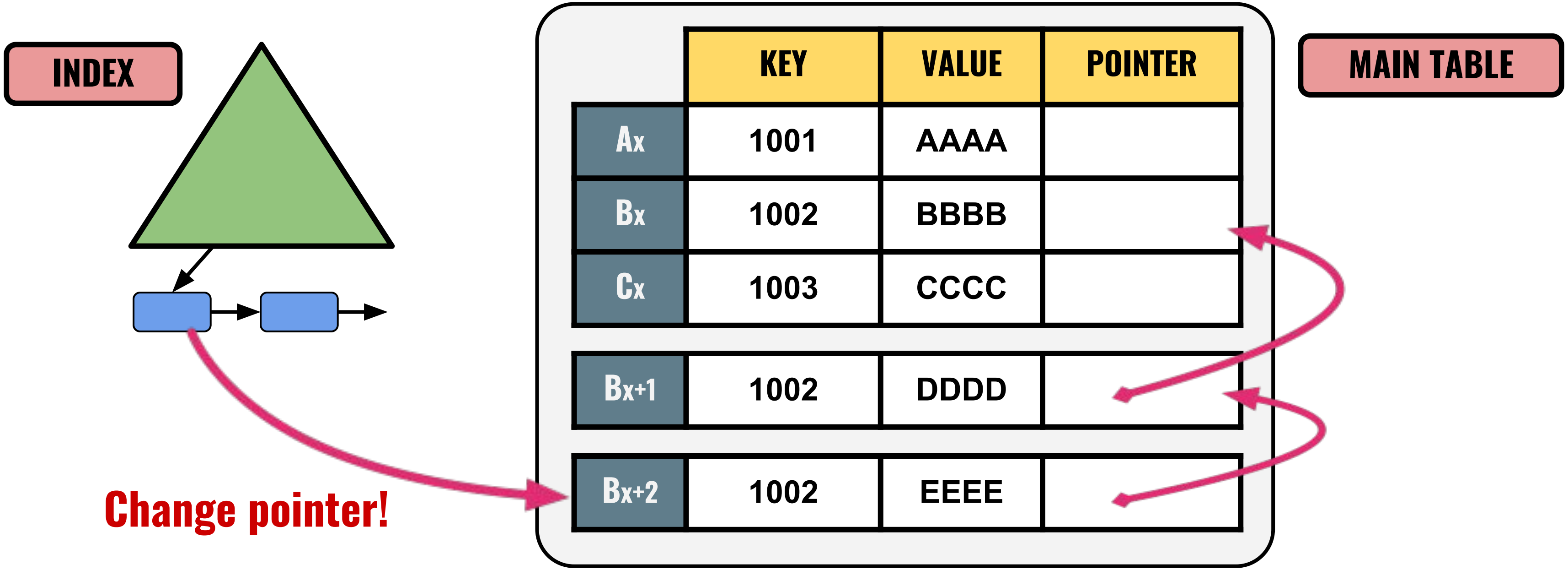
	KEY	VALUE	POINTER
Ax	1001	AAAA	
Bx	1002	BBBB	
Cx	1003	CCCC	
Bx+1	1002	DDDD	
Bx+2	1002	EEEE	

Unchanged!

VERSION STORAGE

Approach #1: Append-Only Storage (Newest-to-Oldest)

- New versions are appended to the same table space.

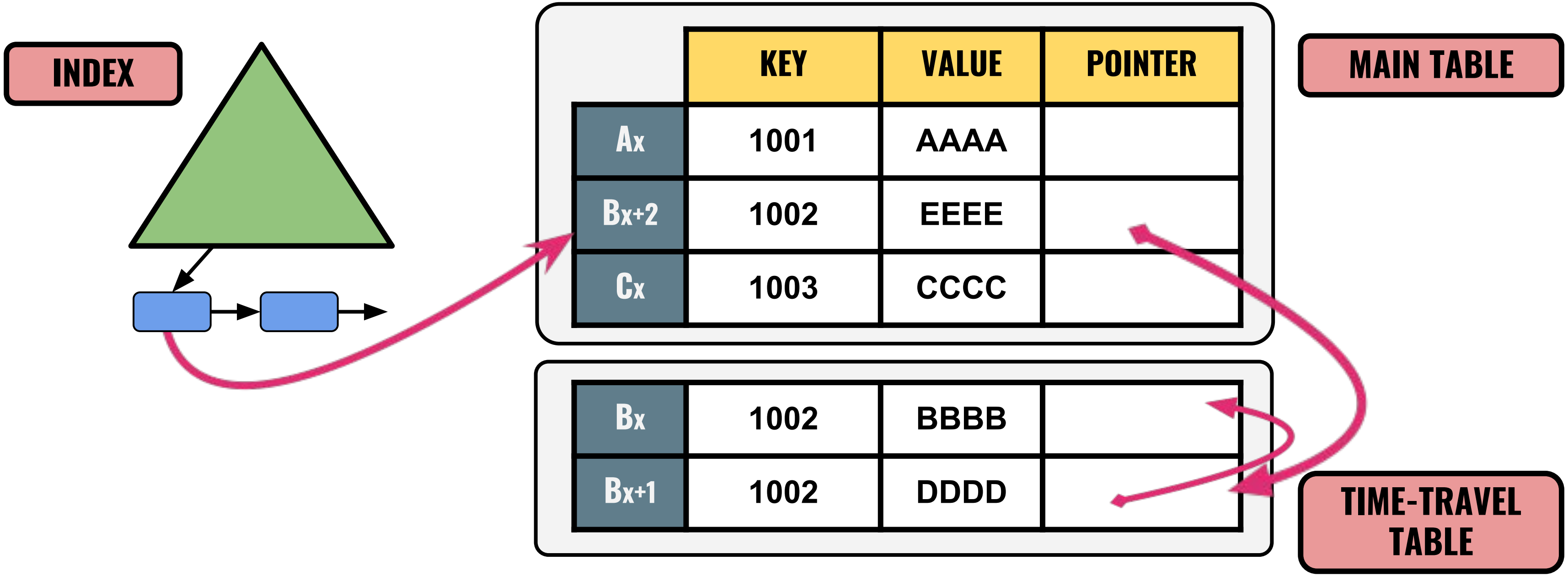


	KEY	VALUE	POINTER
Ax	1001	AAAA	
Bx	1002	BBBB	
Cx	1003	CCCC	
Bx+1	1002	DDDD	
Bx+2	1002	EEEE	

Change pointer!

VERSION STORAGE

- ❑ **Approach #2: Time-Travel Storage**
  - ❑ Old versions are copied to separate table space.



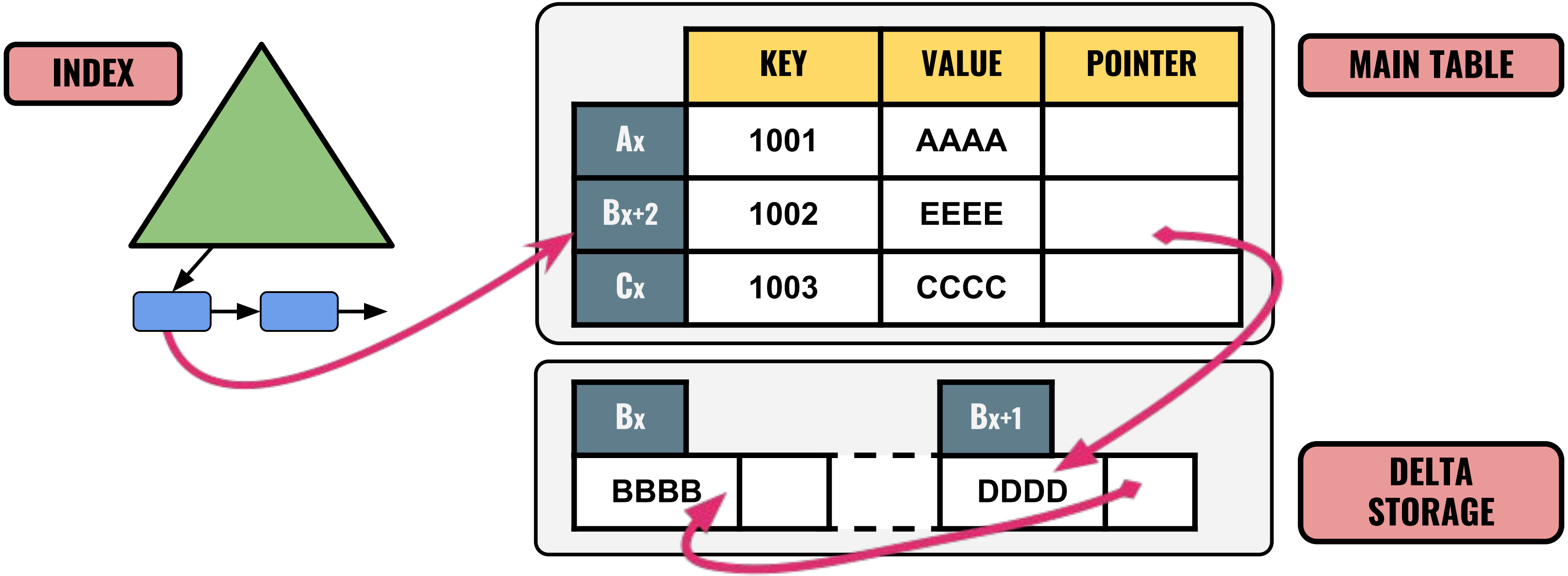
	KEY	VALUE	POINTER
Ax	1001	AAAA	
Bx+2	1002	EEEE	
Cx	1003	CCCC	

Bx	1002	BBBB	
Bx+1	1002	DDDD	

VERSION STORAGE

❑ Approach #3: Delta Storage

- ❑ The original values of the modified attributes are copied into a separate delta space.

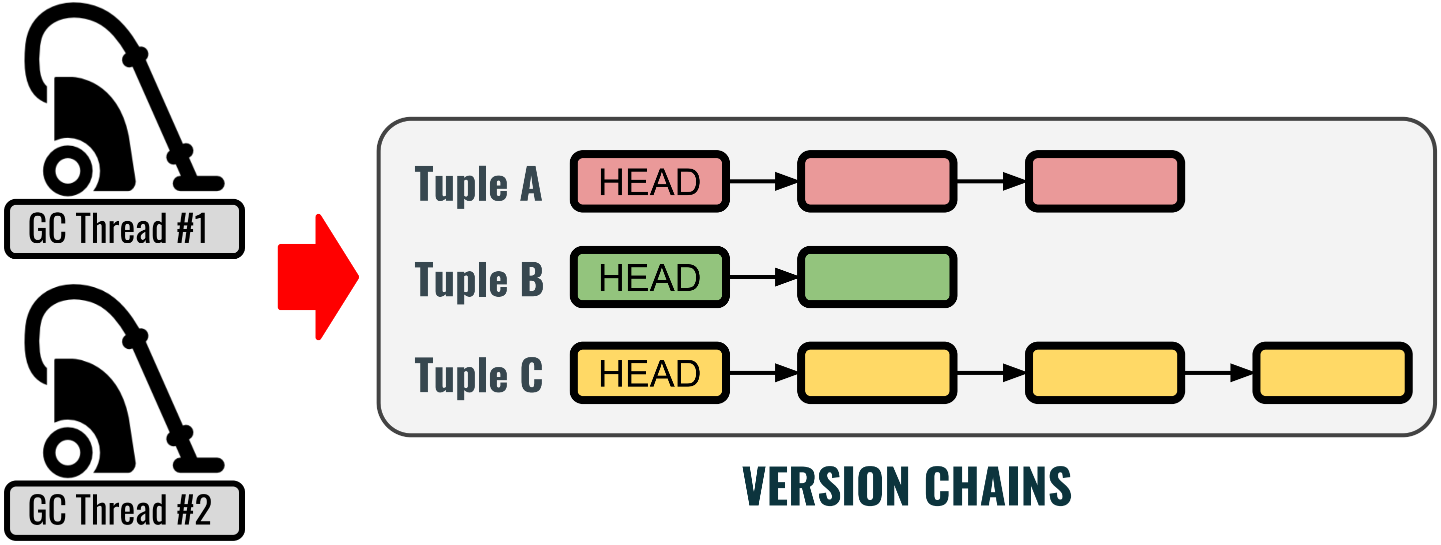


GARBAGE COLLECTION

SCHEME	DBMS
Tuple-Level	ORACLE MySQL PostgreSQL HEKATON Microsoft SQL Server SAP HANA MEMSQL HYRISE NUODB
Transaction-Level	HyPer

GARBAGE COLLECTION

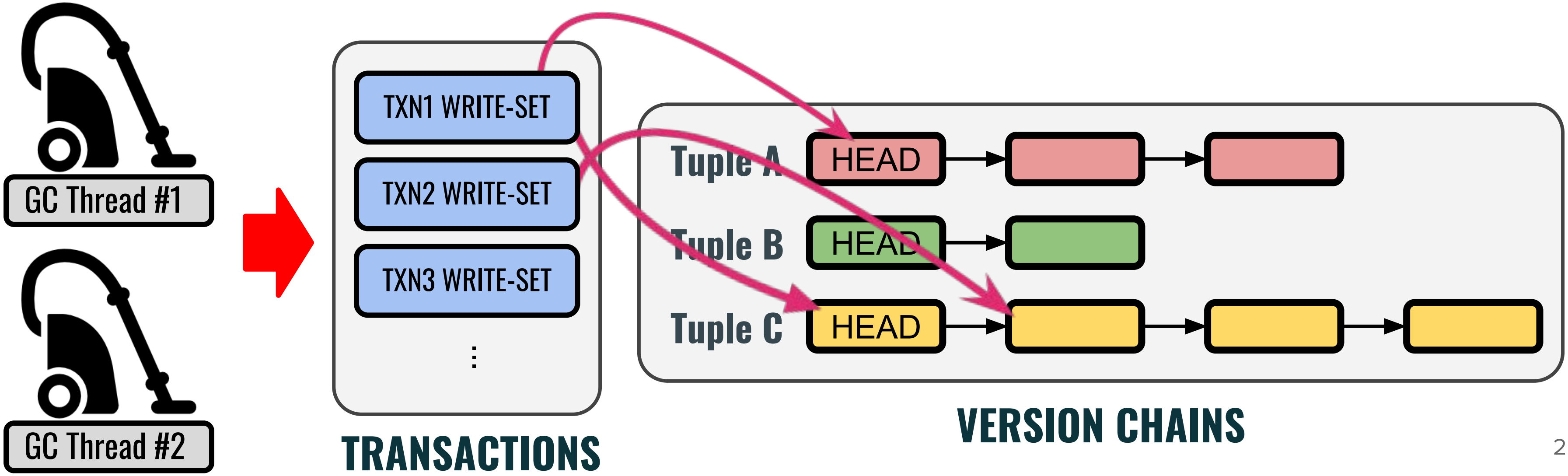
- Approach #1: Tuple-level
  - Find old versions by examining tuples directly.



VERSION CHAINS

GARBAGE COLLECTION

- Approach #2: Transaction-level
  - Transactions keep track of their old versions so the DBMS does not have to scan tuples to determine visibility.

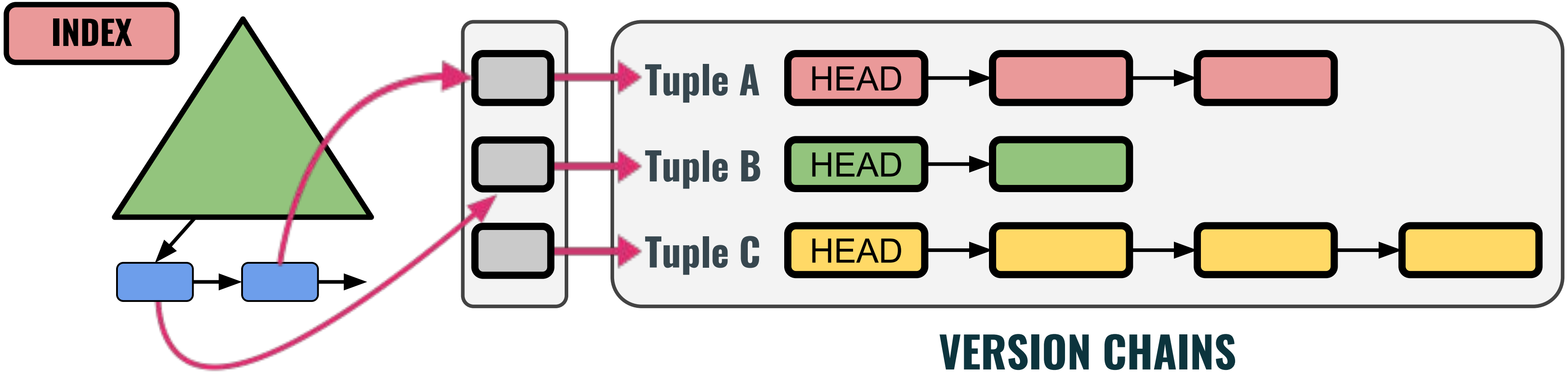


INDEX MANAGEMENT

SCHEME	DBMS
Logical Pointers	ORACLE MySQL SAP HANA HyPer NUODB
Physical Pointers	PostgreSQL HEKATON Microsoft SQL Server HYRISE MEMSQL

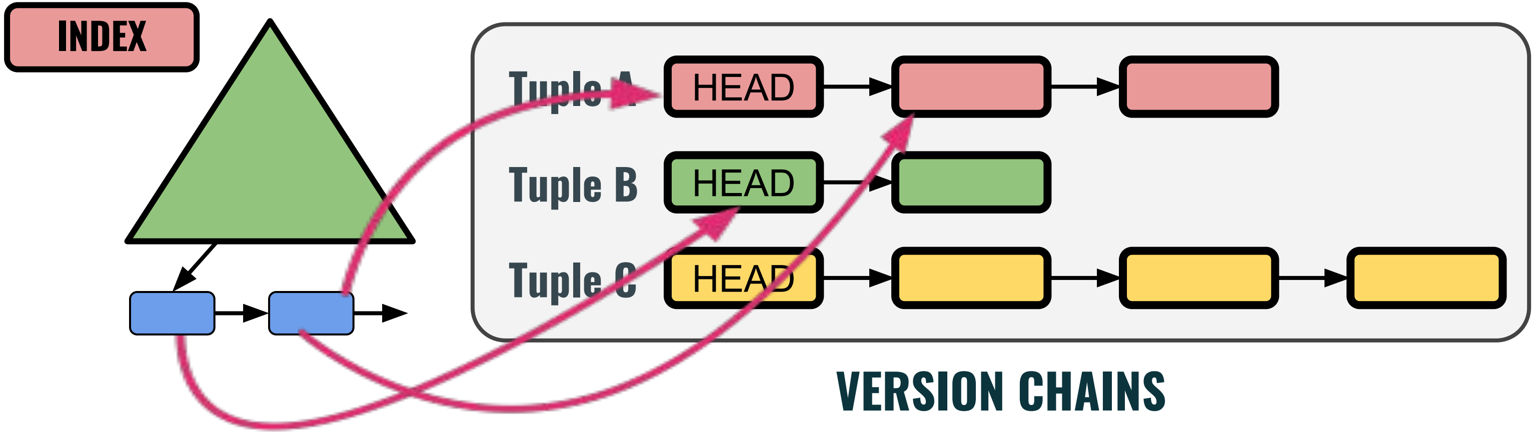
INDEX MANAGEMENT

- **Approach #1: Logical Pointers**
  - Use a fixed identifier per tuple that does not change.



INDEX MANAGEMENT

- Approach #2: Physical Pointers
  - Use the physical address to the version chain head.



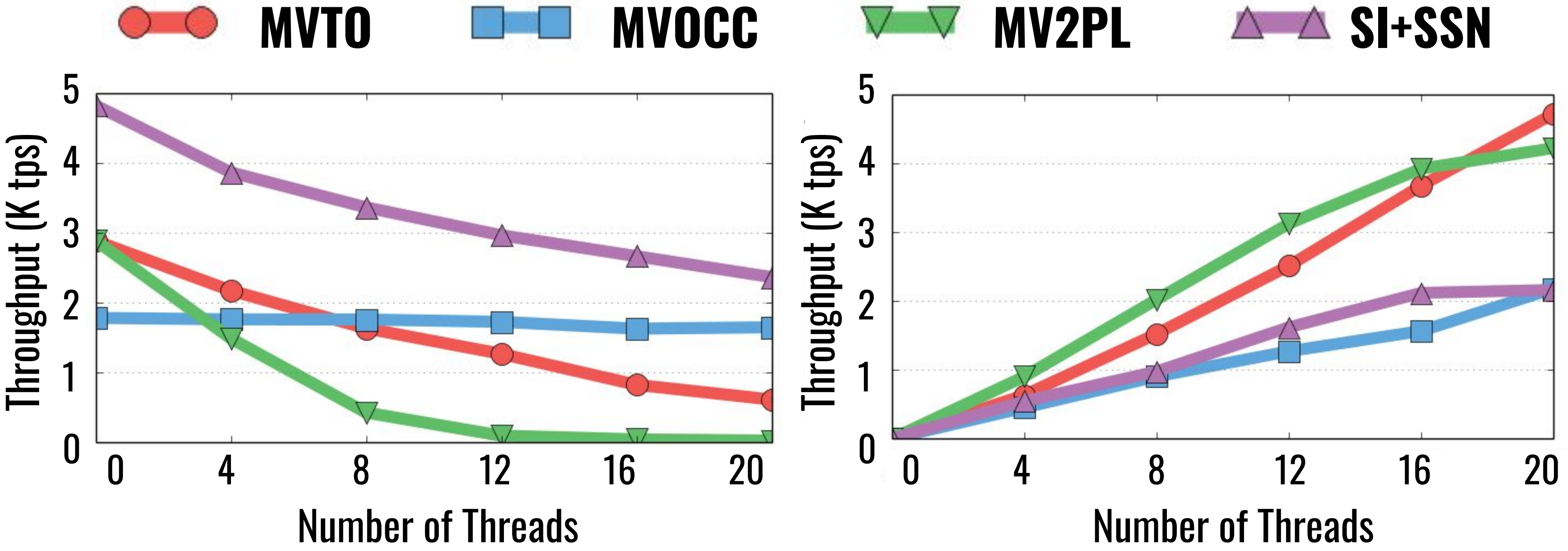
EVALUATION

- Benchmarks
  - YCSB
  - TPC-C
- Configuration
  - 4X Intel Xeon E7-4820 (40 cores)
  - 128 GB DRAM

PELOTON

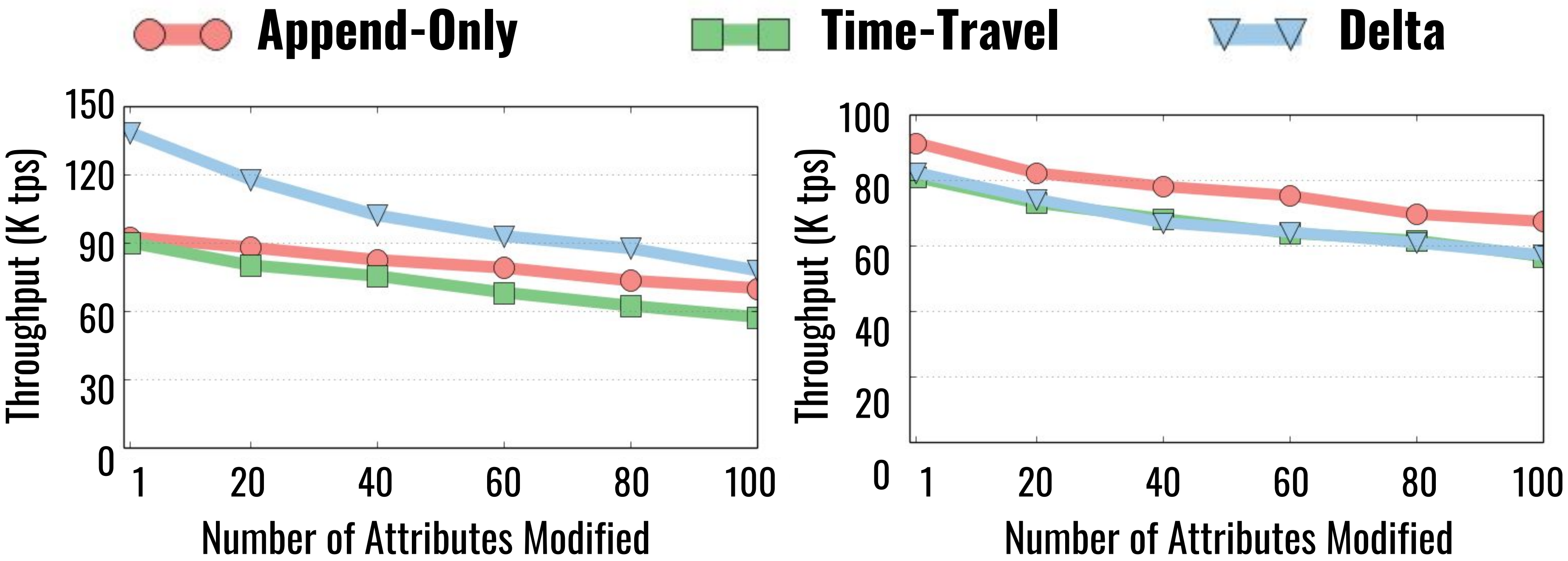
EVALUATION

- Concurrency Control Protocol



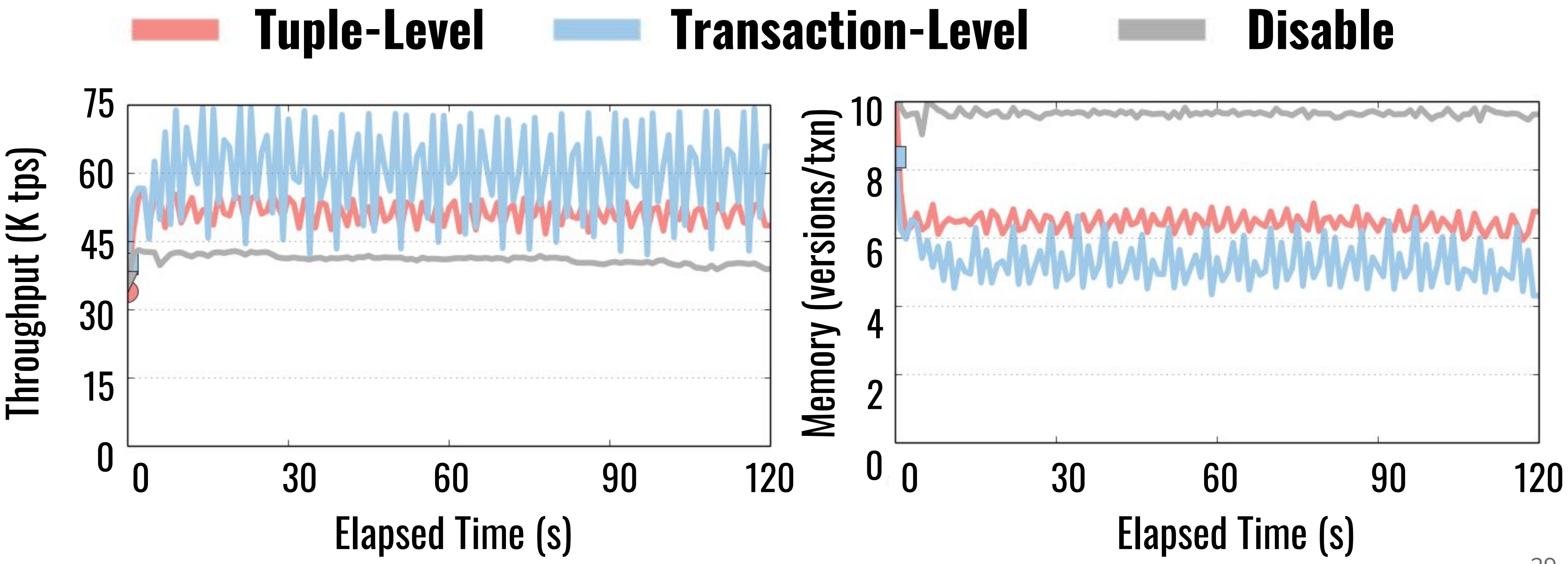
EVALUATION

- Version Storage



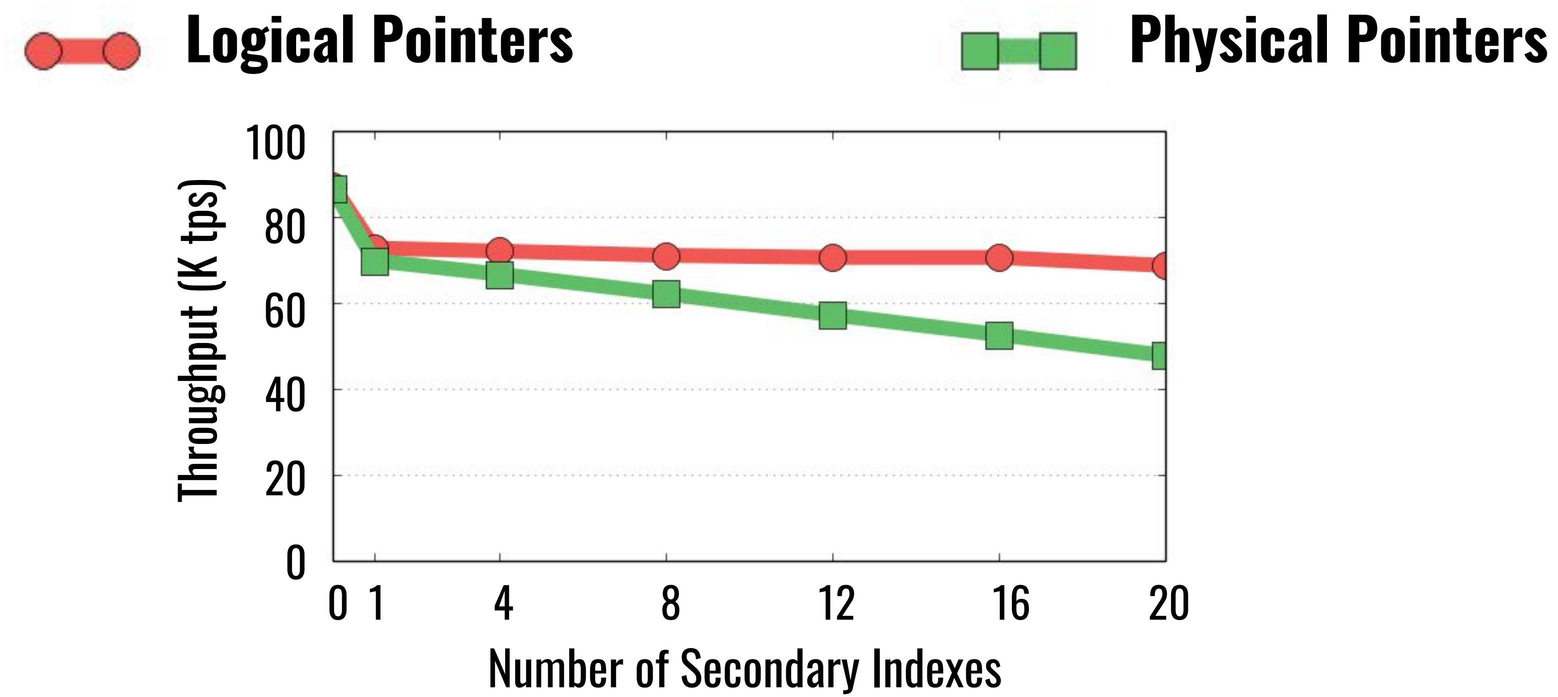
EVALUATION

- Garbage Collection



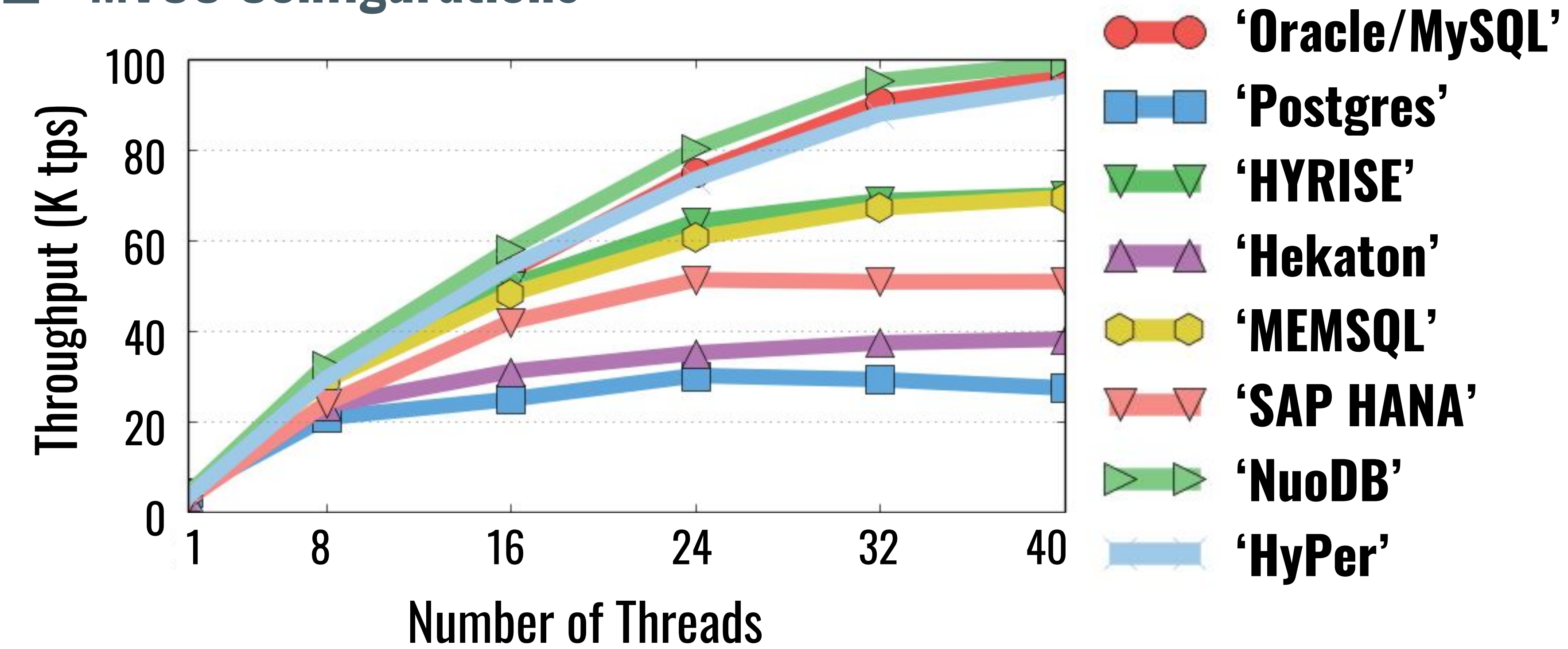
EVALUATION

❑ Index Management



EVALUATION

- MVCC Configurations



CONCLUSION

- ❑ Choosing the best MVCC scheme is challenging
  - ❑ Four design aspects
  - ❑ Multiple design decision combinations
  - ❑ Optimize for different objectives

END

@yingjunwu