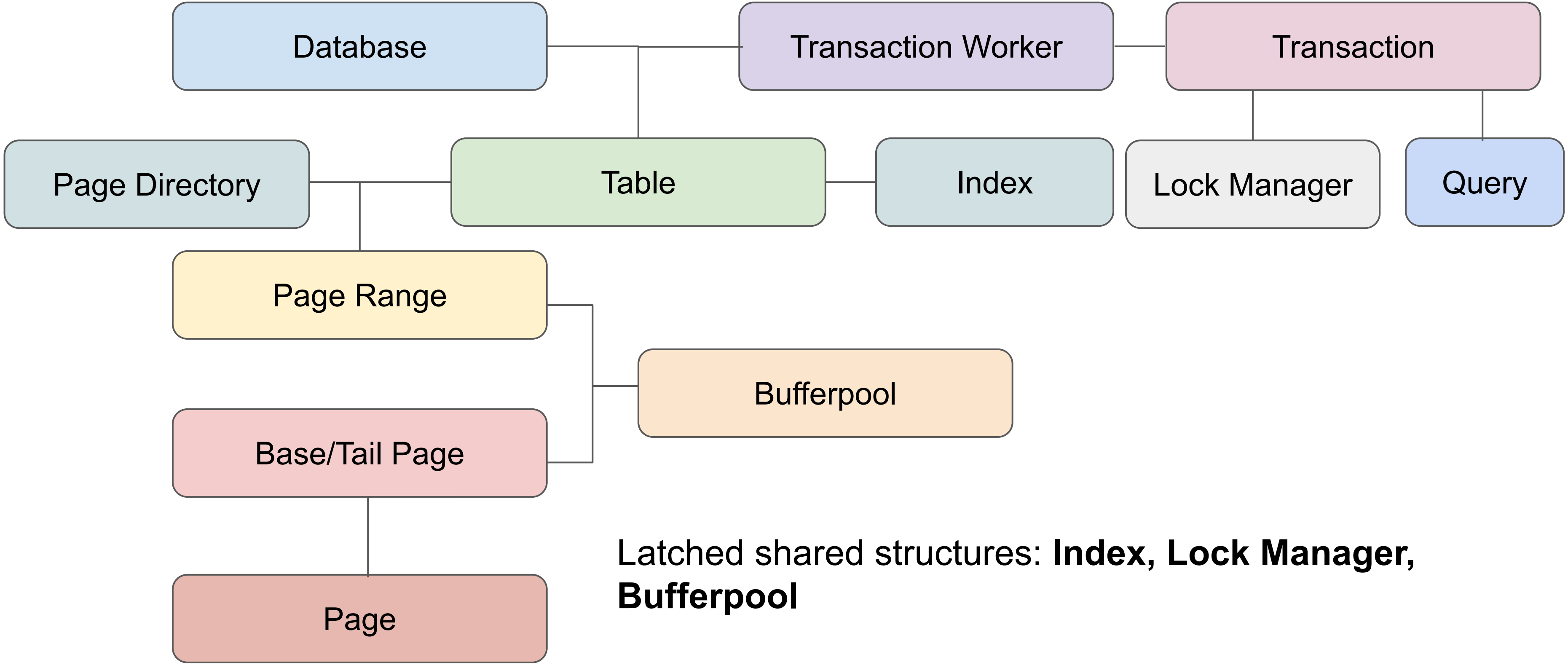
ECS 165A Milestone 3: Multi-threaded, in-memory, and durable L-Store

“Nameless DB”

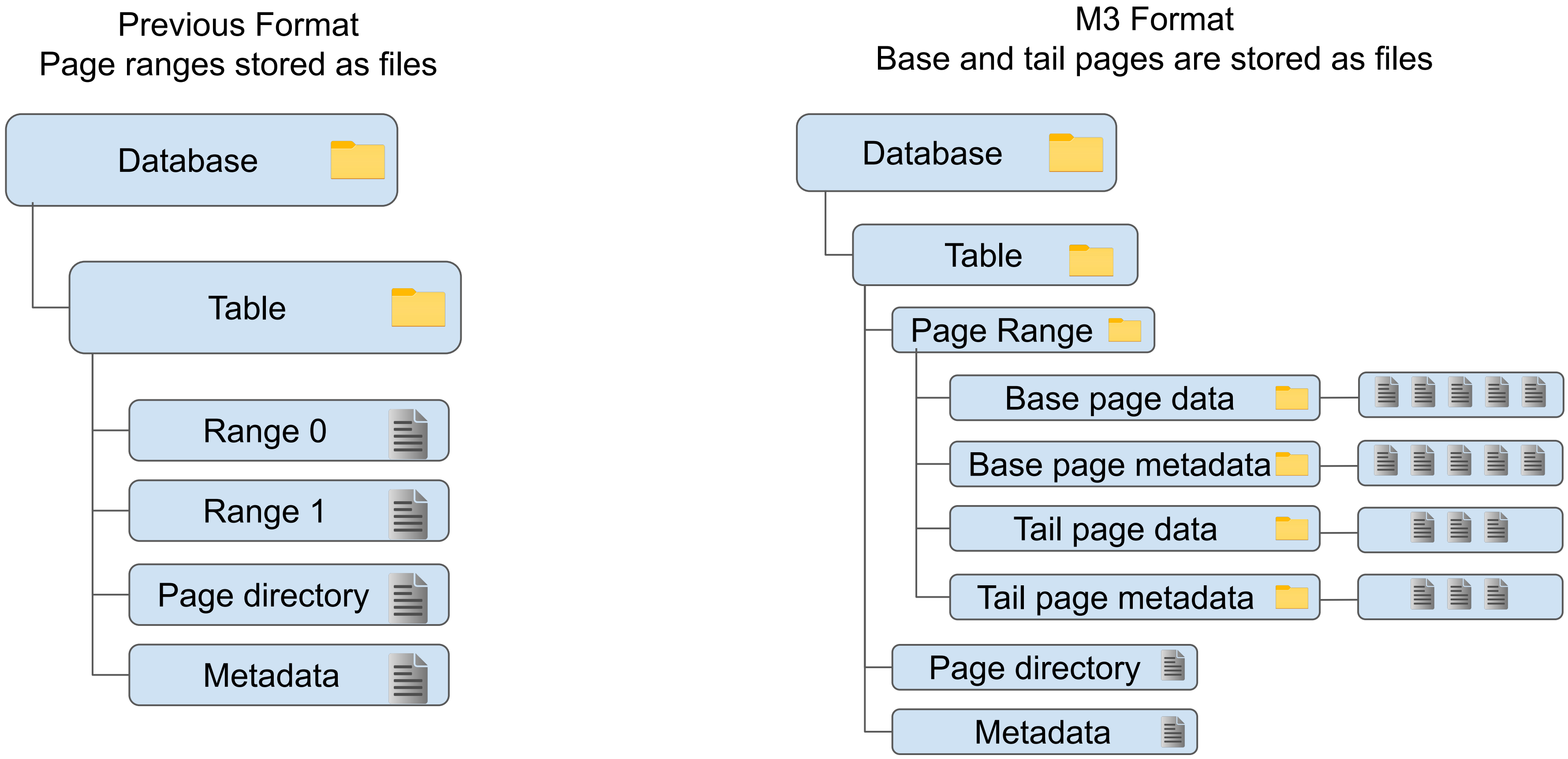
- Nicholas Chan
- Kevin Pack
- Anirudh Shenai (Ani)
- Jay Titterud
- Qing Zhou (Dave)

Database Architecture



Latched shared structures: **Index, Lock Manager, Bufferpool**

Revamped How NamelessDB Write to Disk

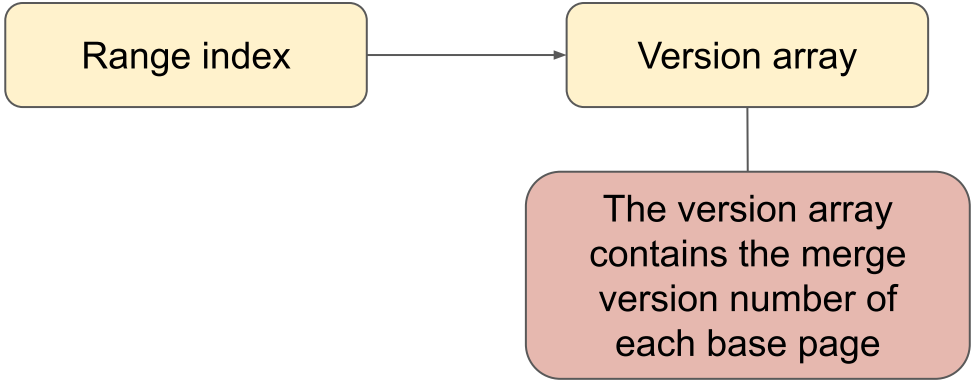


Simplify Page Directory

Older implementation:

Merge: Originally had page range granularity. Calling a merge would update the range path to that of the merged range.

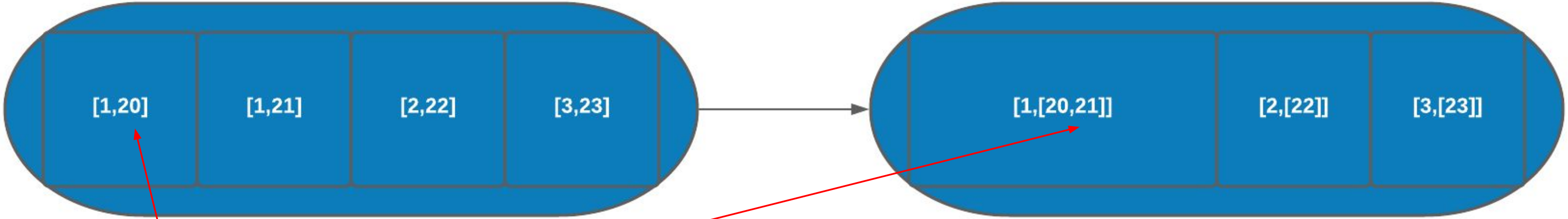
New implementation:



Merge: Now has base page granularity. Calling a merge increments the version number of the base page which is being merged.

Index - Optimization

Adjusted key structure to help query performance on columns featuring large amounts of duplicate keys



Queries Optimized:

[1,20], [1,21] → [1,[20,21]]

Index.compress_key:

Searches for pre existing key value, appending the subsequent rid.

Index.mini_delete:

In the case where rid list contains more than one rid, the delete function just pops the appropriate rid from list, maintaining tree balance.

Query Improvement

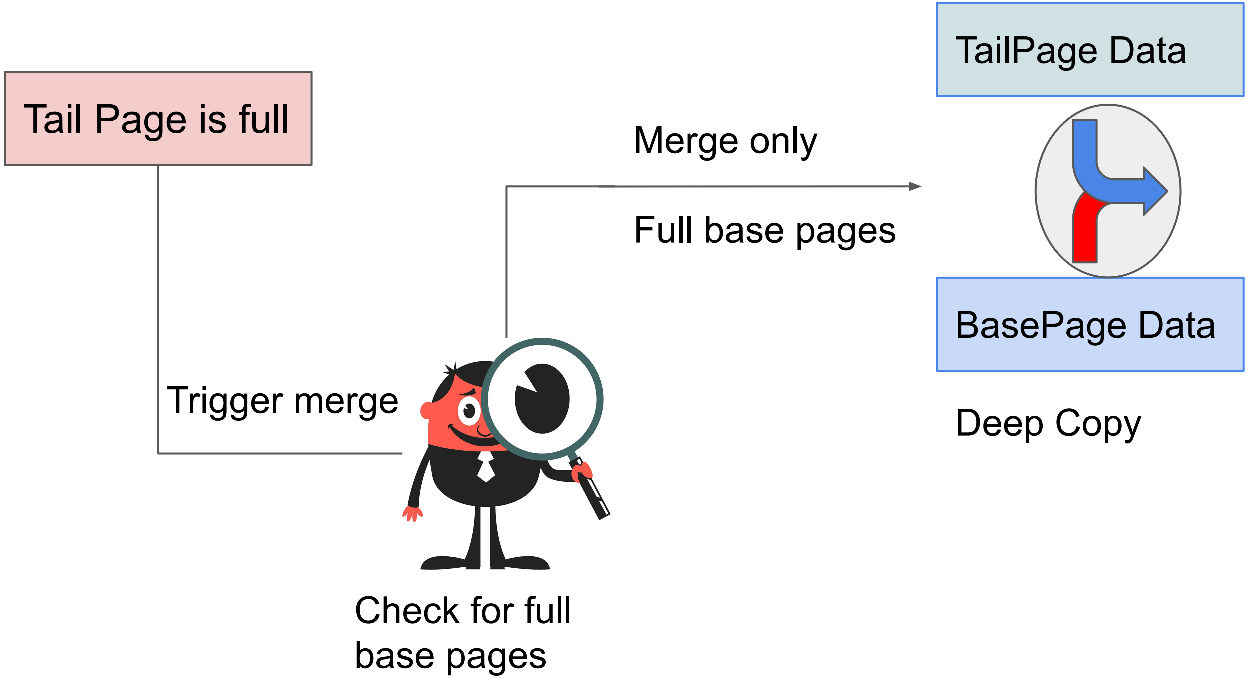
self.table.index.indices:

[Index, None, None, Index, None]

Ensures accurate searching by updating existing BTrees after table data is altered. Takes place with write queries: Update, Insert, Delete.

M2 Average	M3 Average
Update	Update
10.44387	6.703125

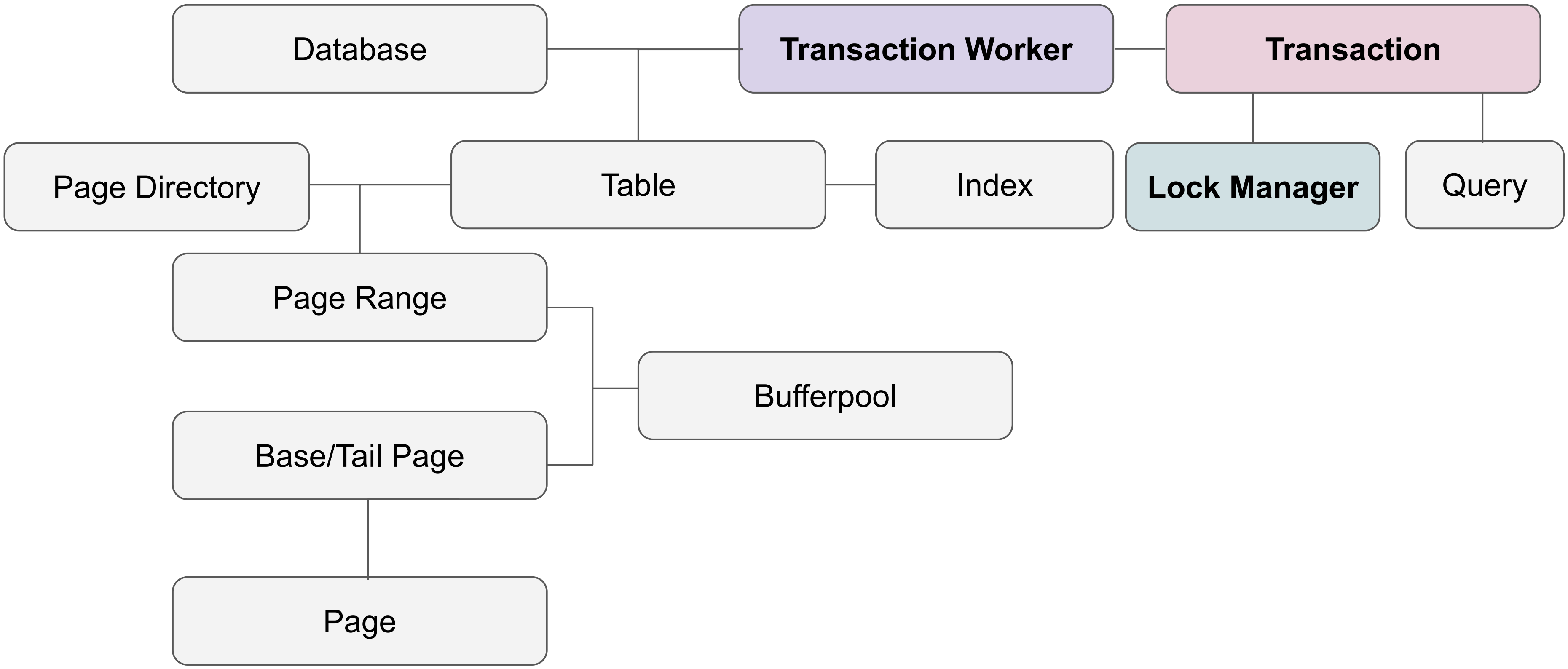
Merge



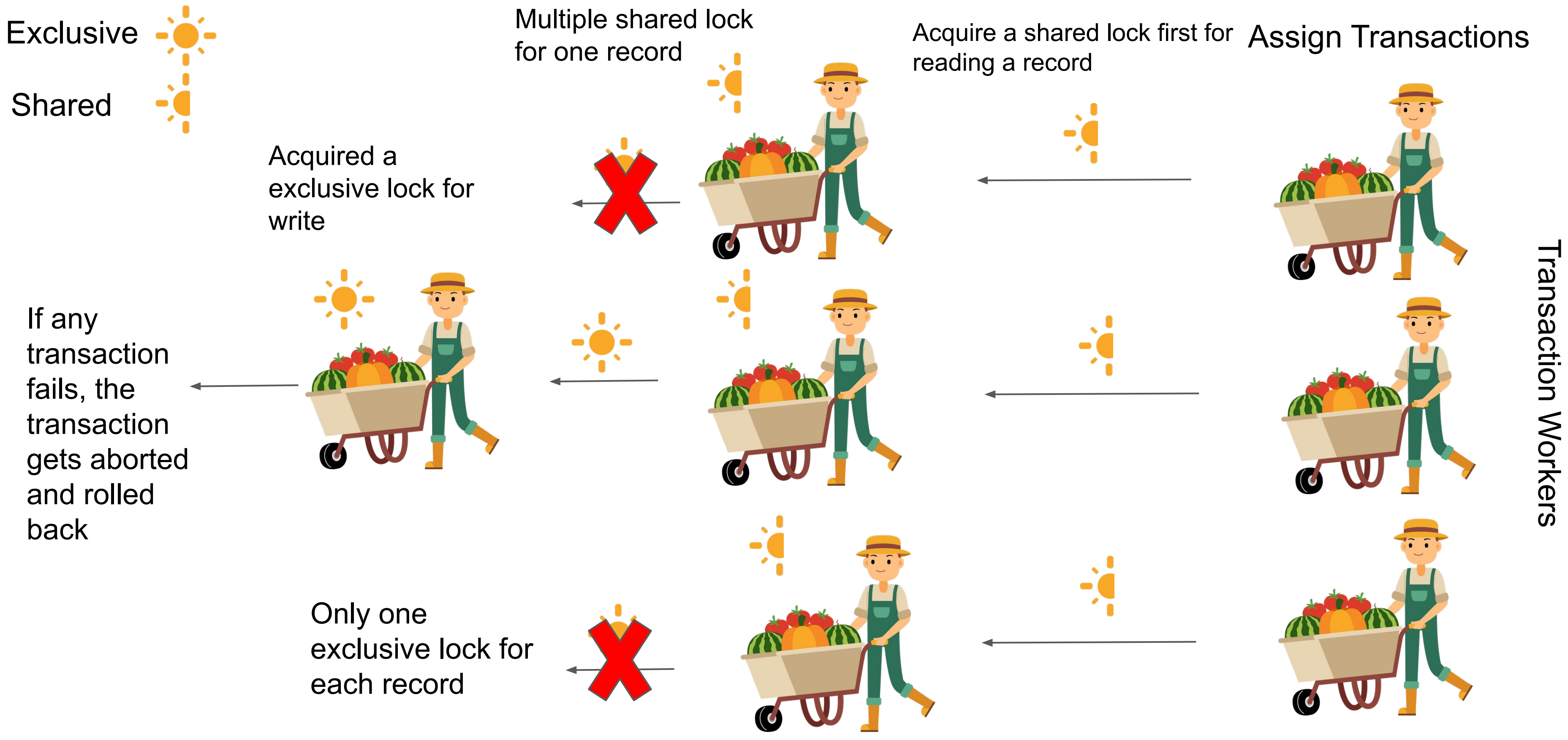
- Update the TPS column
- Update version control on Page Directory
- Preserves isolation and consistency

Merge will trigger when tail is page full but we only execute the merge on full base pages

M3 Implementation for Concurrency



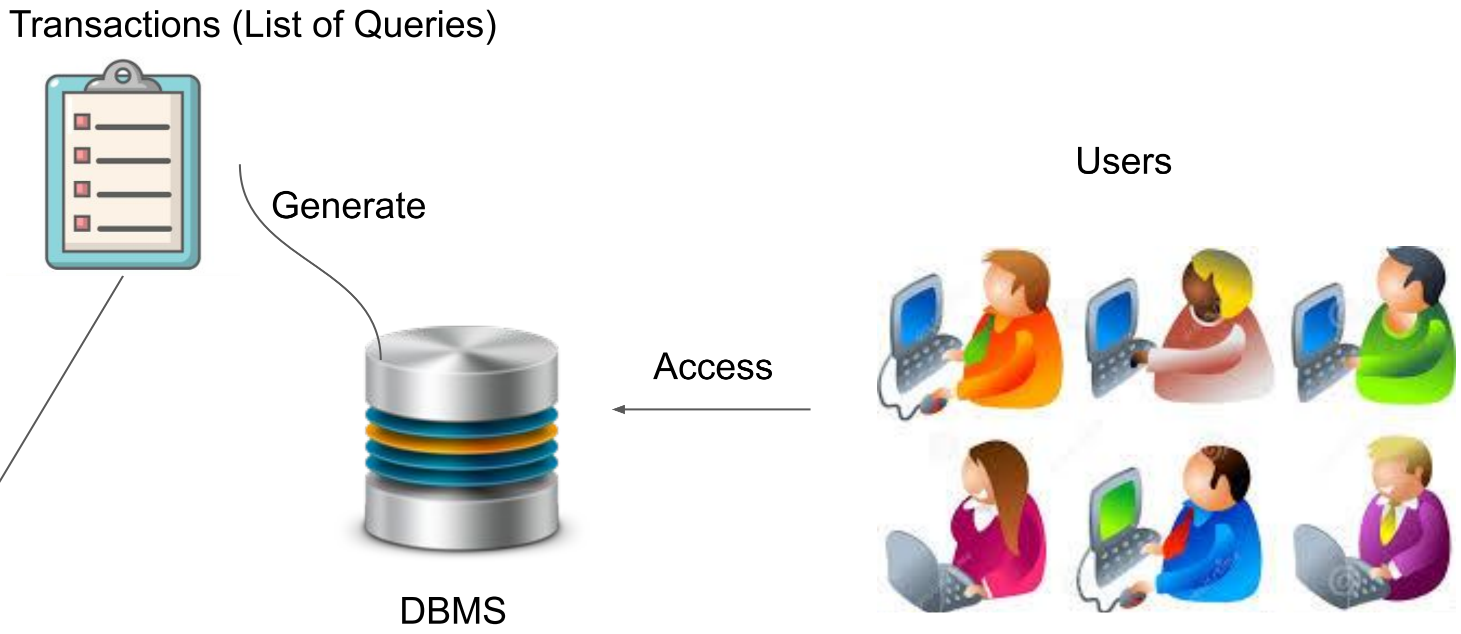
Transaction Worker



Transaction

1. Attempts to execute

1. If any query fails, the transaction get aborted and rollback



Lock Manager

Transaction Workers

Lock Manager

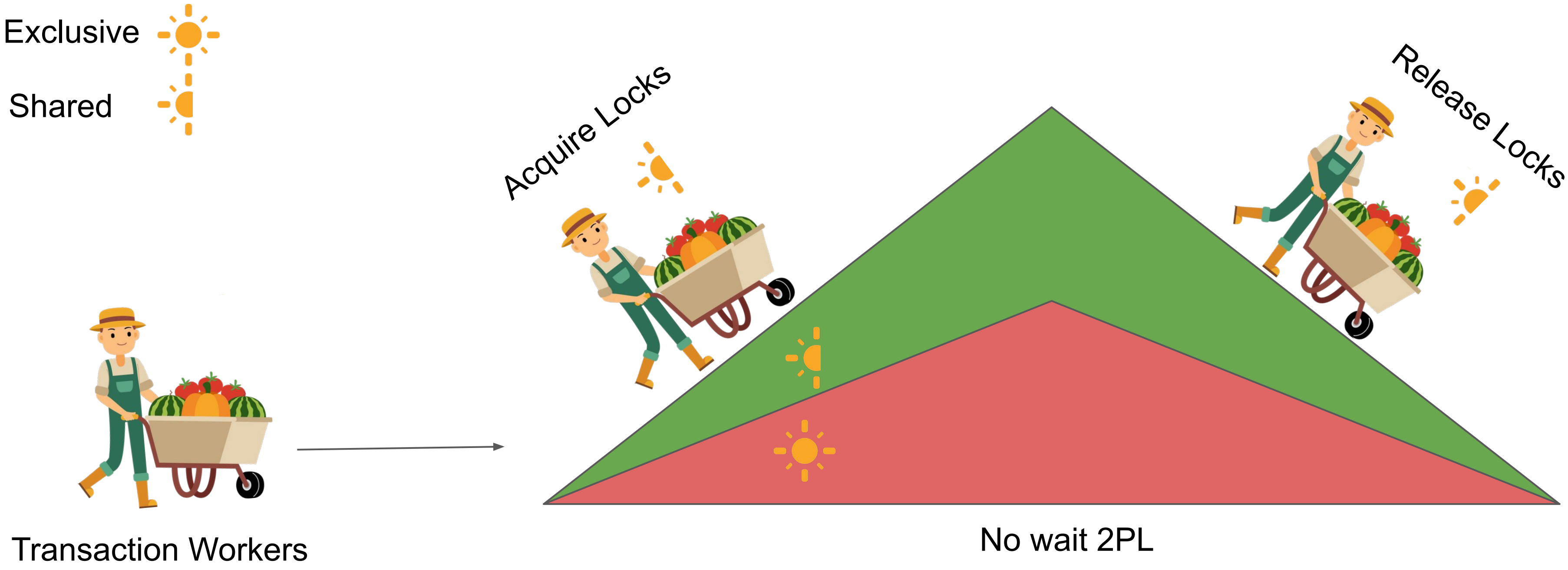
When the transaction worker acquires lock, the RID is manage by our Lock Manager

Ensure **no shared lock** when there is a exclusive lock

&

Maintain a **set of readers** and up to **one writer** for each record (RID)

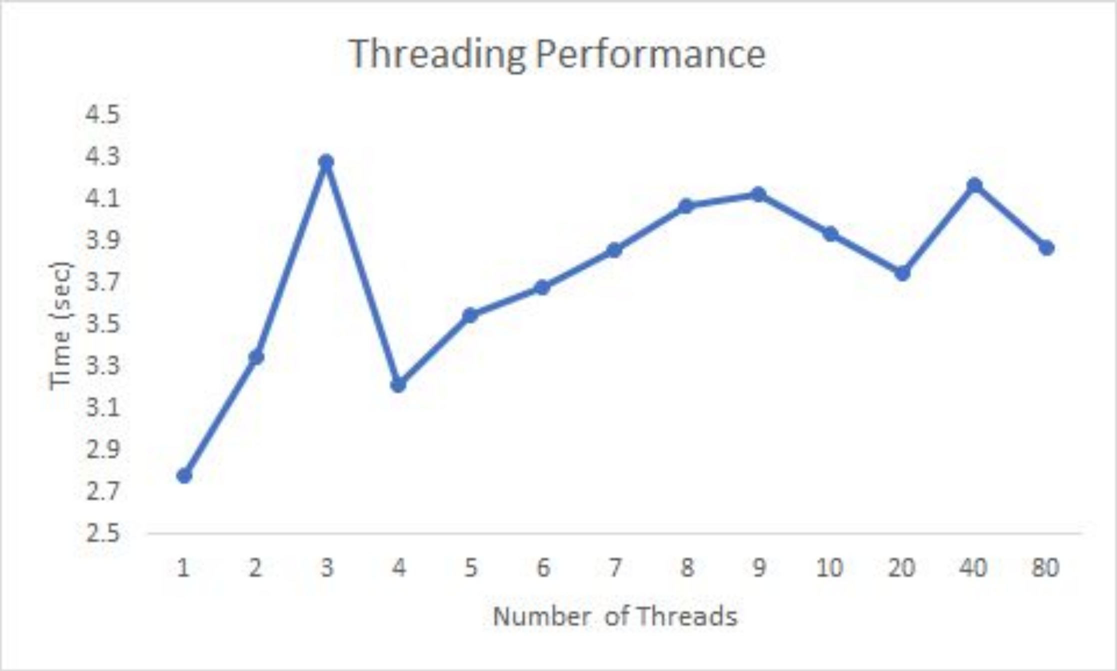
Strict 2PL



Exclusive
Shared
Acquire Locks
Release Locks
Transaction Workers
No wait 2PL

M3 Performance by Threading

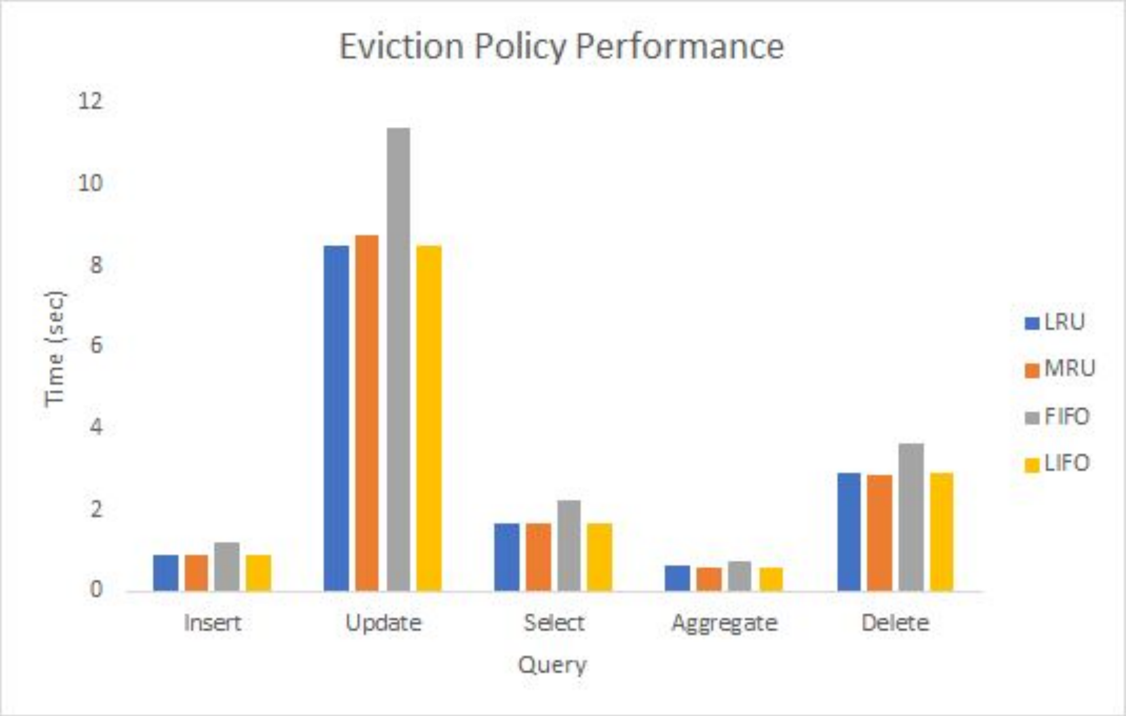
- Window 10 OS
- Intel i7 @ 2.60GHz
- 16GB RAM



*1k queries function
*Used LRU eviction policy

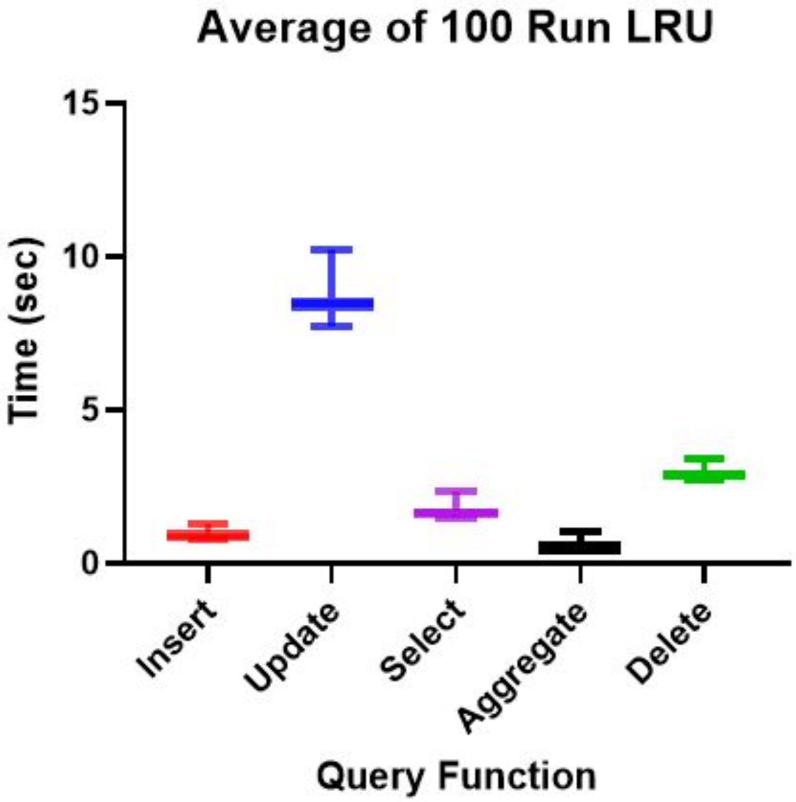
M3 Performance by Eviction Policy

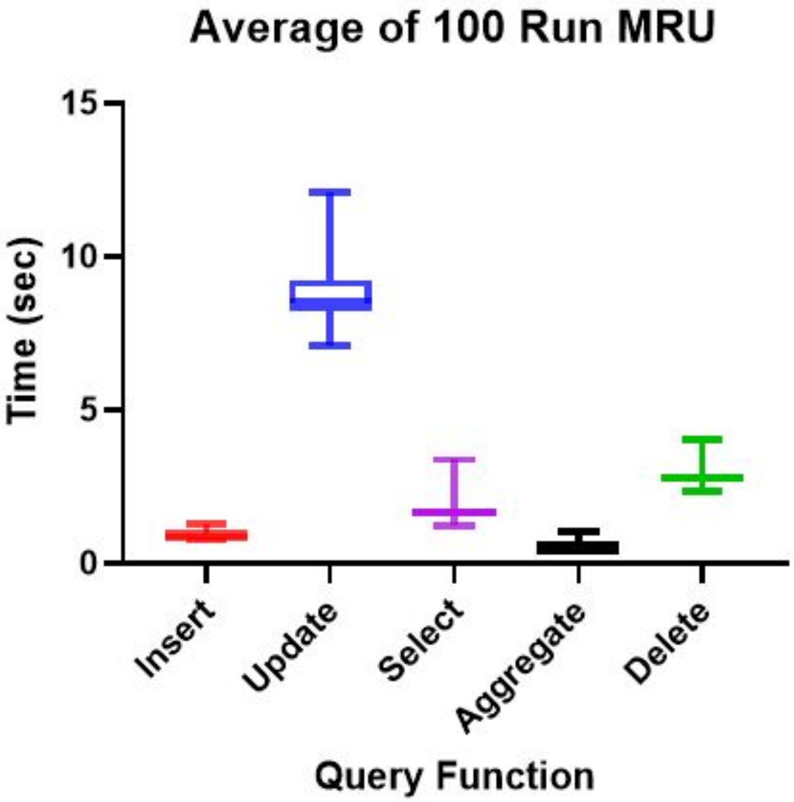
- Window 10 OS
- Intel i7 @ 2.60GHz
- 16GB RAM



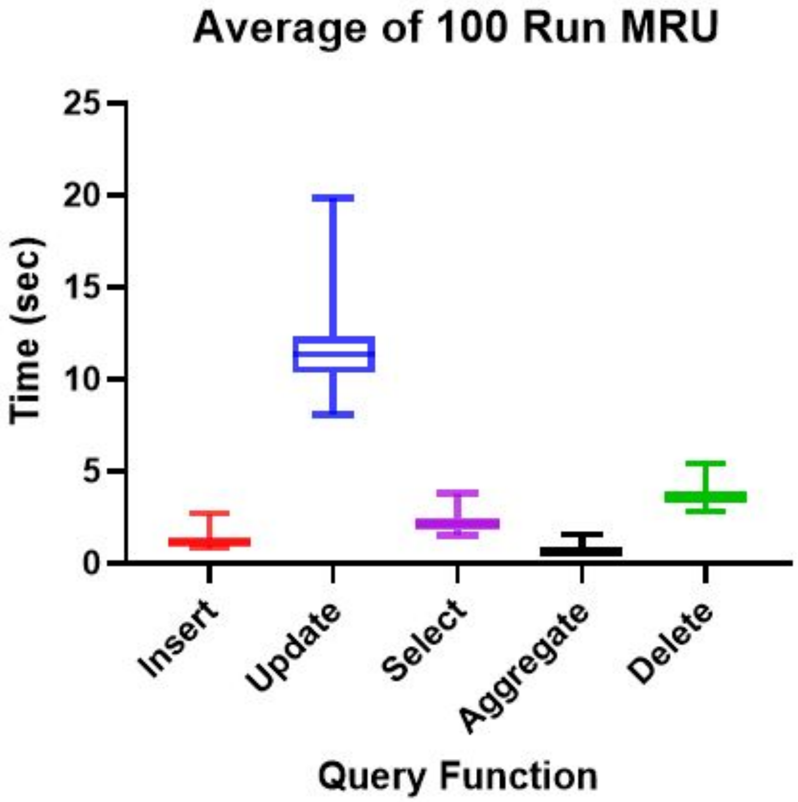
*10K was used for all function except Aggregate
*Aggregate 100 of 100 record batch took
*Default 8 threads

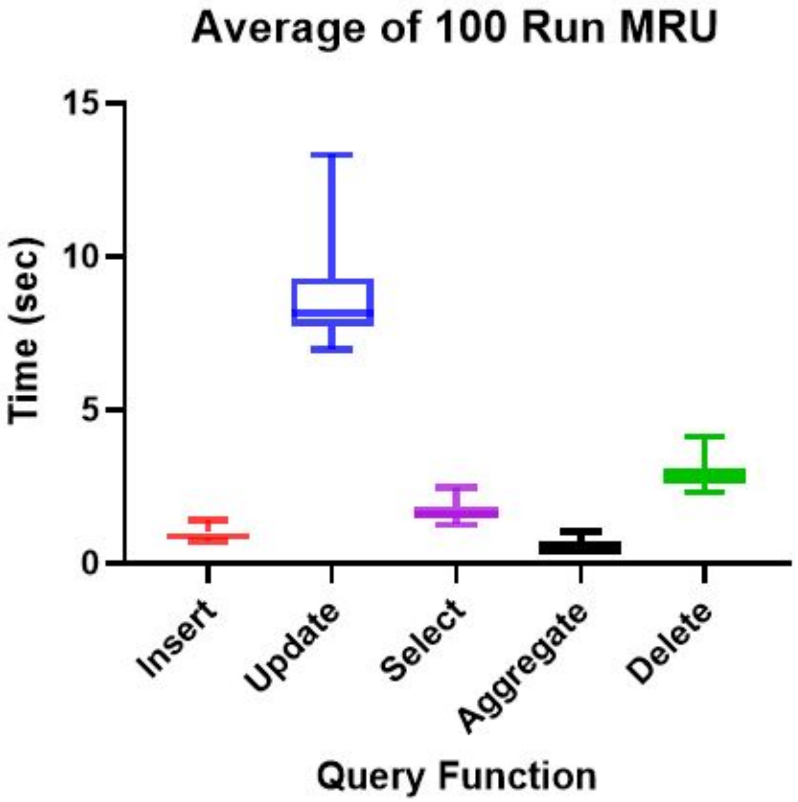
M3 Performance by Eviction Policy





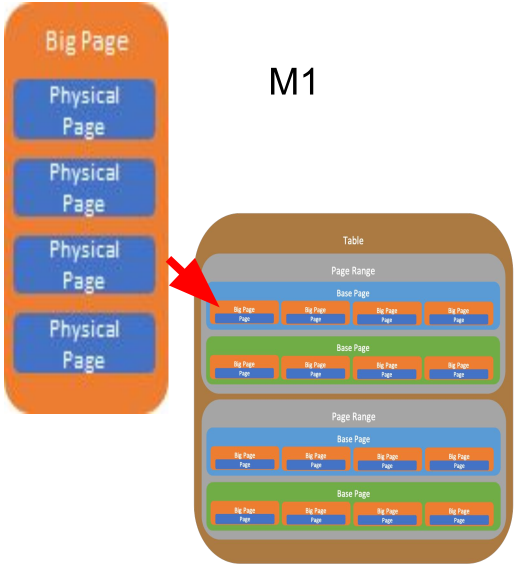
M3 Performance by Eviction Policy



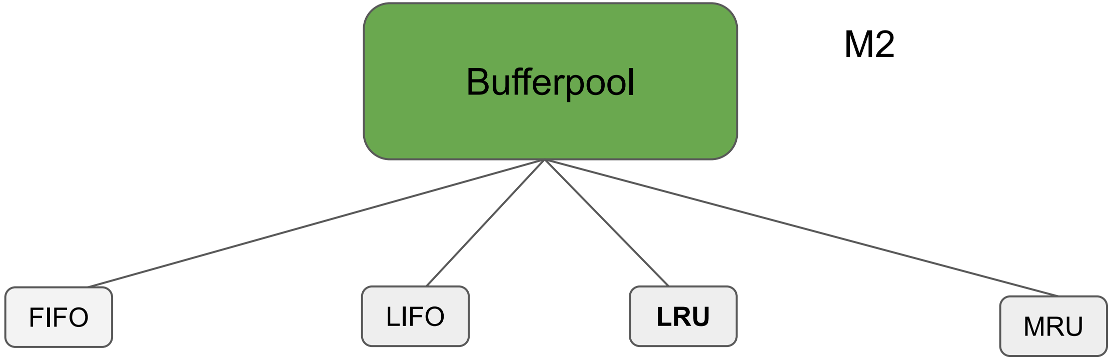


Experiments in Each Milestone

M1



M2



- Big Page (Negligible impact on performance)
- Designed a rudimentary Index for key to RID pair
- Designed a rudimentary page directory that maps RIDS to records

- BufferPool Eviction Policies (LRU is most consistent in performance)

Conclusion

- Learned how to build a database from scratch, including implementation of queries, indexing, disk storage, merge, and multi-threading
- Worked on testing including overall test, unit tests, and performance tests, as well as experimental design, debugging as we programmed.
- Developed a team work ethic which involved division of work, accountability and communication, and understanding different parts of the software design process.
- **We would like to thank Dr. Sadoghi and the TAs** for giving us this challenging assignment which pushed our creative limits and improved our skills. We are also grateful for all the help we received through the process.

Sadoghi M, Bhattacherjee S, Bhattacharjee B, Canim M (2016) L-store: a real-time OLTP and OLAP system. In: CoRR, pp 540–551