LTeam Milestone 3

Alejandro Torres, Jenny Wang,
Ho-Chih Ma, Jamie Wu,
Karthik Palanisamy

Team Member Roles

Leadership Roles:

- Team Coordinators: Jenny, Alejandro
- System Architects: Everyone
- Developers: Everyone
- Testers: Everyone

Implementation and Design Areas:

- Transaction Semantics: Jamie, Howard
- Multithreading Concurrency Control: Jamie, Howard
- Future Implementation: Karthik, Jenny, Alejandro
- Performance: Karthik, Jenny, Alejandro

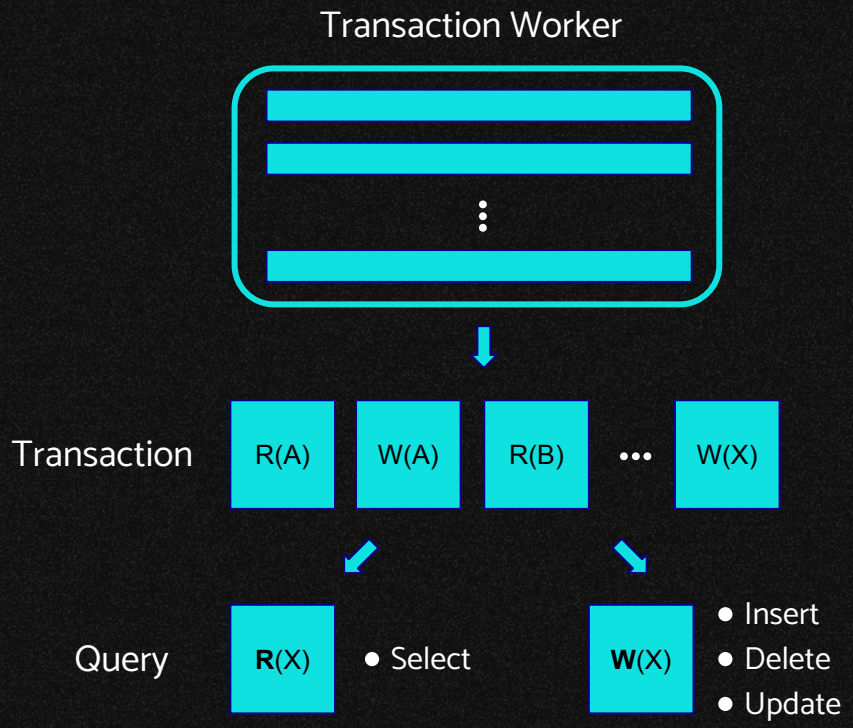
(1) Transaction Semantics
(2) Multithreading Concurrency Control
(3) Future Implementation
(4) Performance
(5) Live Demo and Q&A

[1] Transaction Semantics

Transaction & Transaction Worker

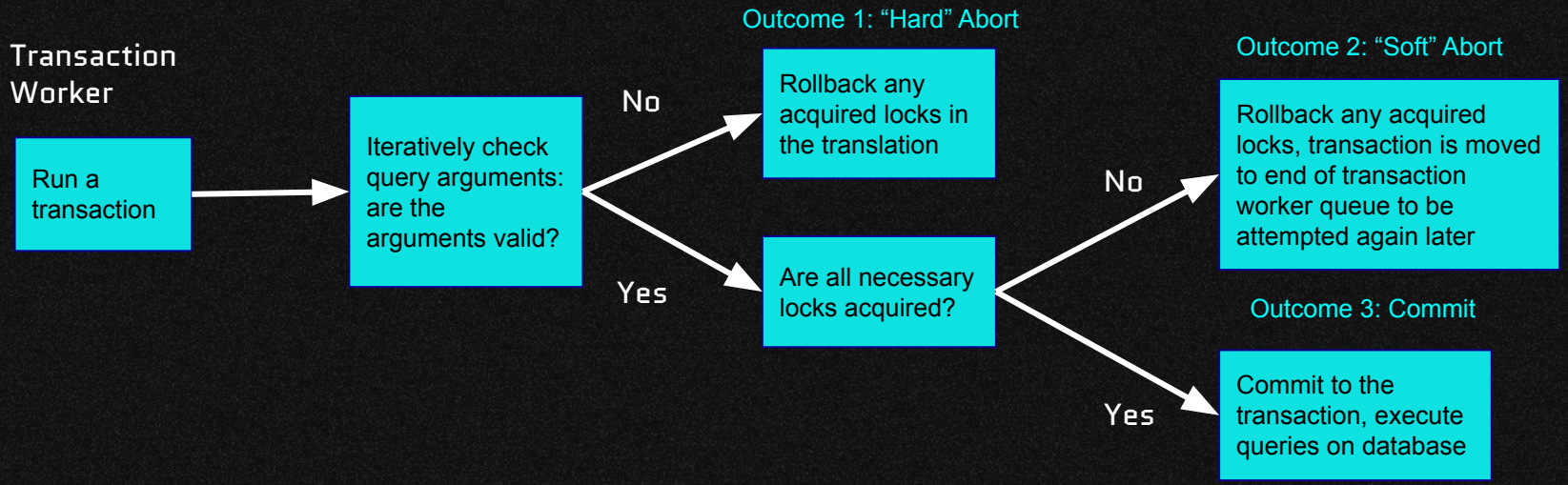
Transaction:
- A sequence queries
- Either read operations or write operations

Transaction worker:
- Contains a list of transactions
- Keeps track of the status of transactions



Commit & Abort

Goal: Achieve atomicity by determining whether transactions are valid and committing groups of queries or none with aborts to release any acquired locks to prevent deadlock

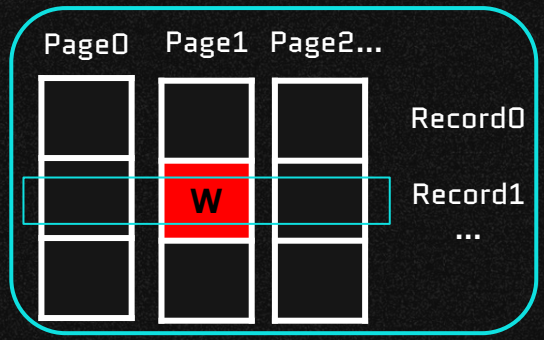


(3) Multithreading / Concurrency Control

Concurrency Using Strict 2PL Policy

Motivation

- To avoid race conditions – no threads should have access to resources at the same time
- Preventing anomalies with interleaved execution
  - WR Conflicts
  - RW Conflicts
  - WW Conflicts



Implementation

- Locks are implemented in record level (Physical Page)
- If a column in a record is access, all columns will be locked
- Using Python Threading, Lock(), Acquire(), and Release()

Shared and Exclusive Locks

Goal: To achieve isolation in the database while preserving read efficiency

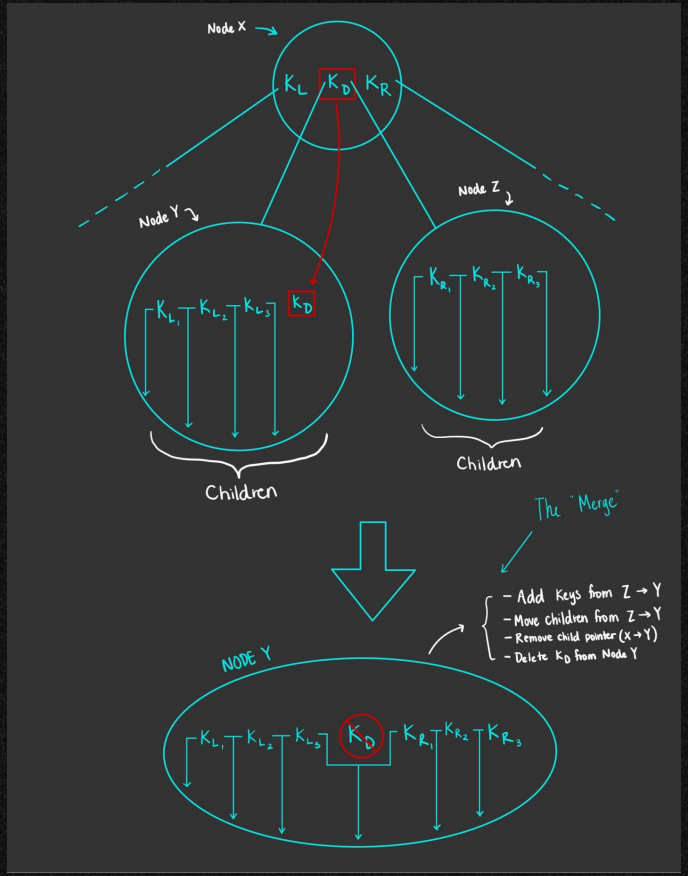
Operation Type	Reading	Writing
Reading	Yes	No
Writing	No	No

[4] Future Implementation

Post Milestone 3 Ideas

1) Use a different programming language, such as C
   a) To achieve true parallelism
   b) Better memory management
   c) Overall increase in performance time
2) Establish a special priority algorithm that executes threads in the most efficient order

Delete


Node X
K_L K_D K_R
Node Y
Node Z
K_{L_1} K_{L_2} K_{L_3} k_D
K_{R_1} K_{R_2} K_{R_3}
Children
Children
The "Merge"
- Add Keys from Z → Y
- Move children from Z → Y
- Remove child pointer (X → Y)
- Delete K_D from Node Y
NODE Y
K_{L_1} K_{L_2} K_{L_3} K_D K_{R_1} K_{R_2} K_{R_3}

(5) Performance

Overall Milestone Performance

```
Inserting 10k records took:                        3.875
Updating 10k records took:                         19.46875
Selecting 10k records took:                        2.3125
Aggregate 10k of 100 record batch took:  0.234375
DELETING 10 RECORDS
Deleting 10k records took:                         0.765625
total db time: 26.65625
```

Live Demo and Q&A
Thank you!