Milestone 3

Ashwin Chembu, Ruohan Huang, Trevor Lash, Mira Stenger, Yuecheng Zhao

TEAM

Yuecheng Zhao

Transactions, concurrency control, and debugging

Mira Stenger

Transactions and concurrency control

Ruohan Huang

Debugging

Presentation

Trevor Lash

C++ compatibility

Foreign key

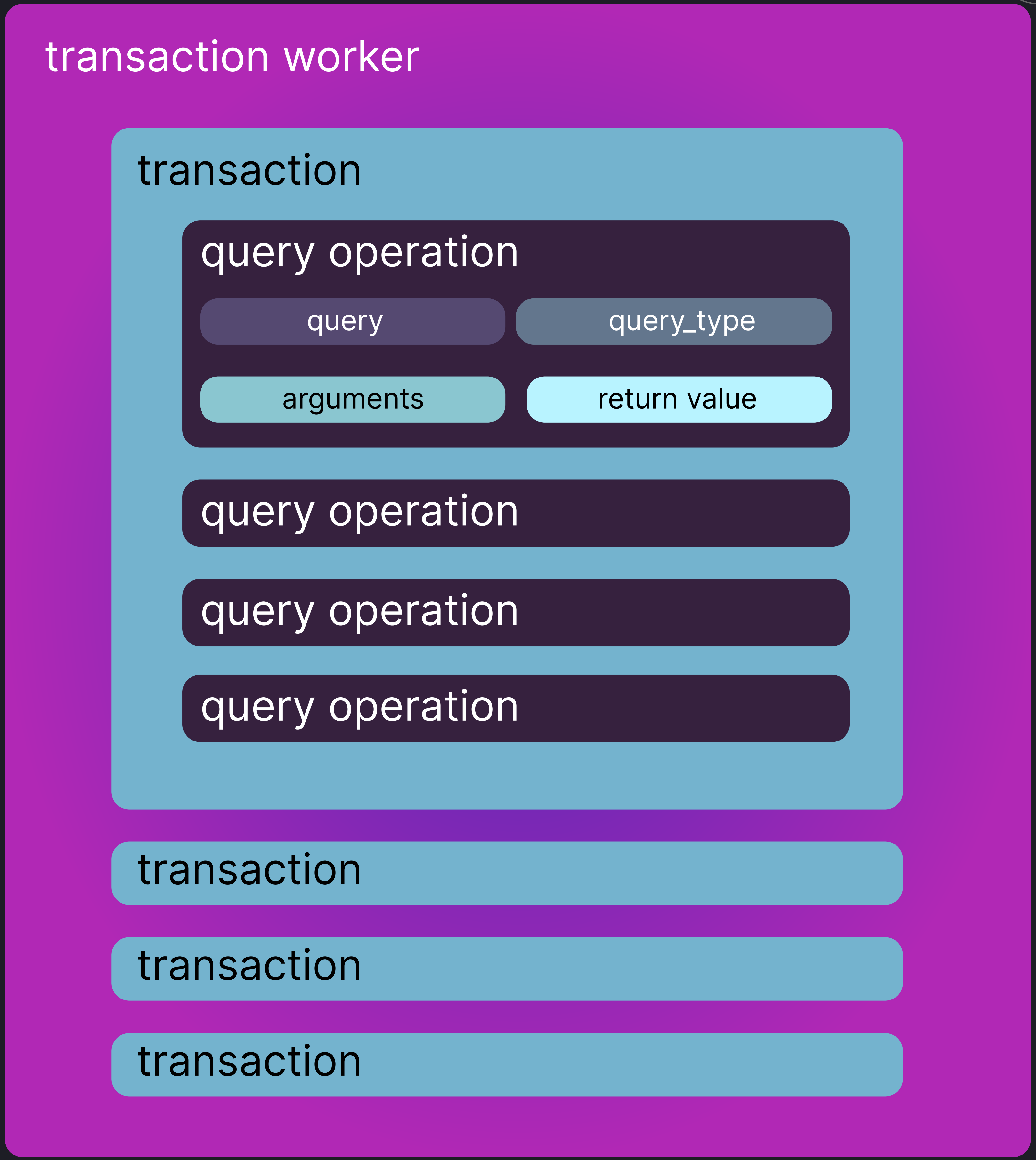
Ashwin Chembu

Debugging

Presentation

TRANSACTIONS

- Each Transaction object has a vector of Query Operation objects for queries
- Query Operation objects contain a Query object and execution data
- Transaction Worker executes transactions on a thread using a vector of Transaction objects
- Failed transactions are retried



transaction worker

transaction

query operation

query

query_type

arguments

return value

query operation

query operation

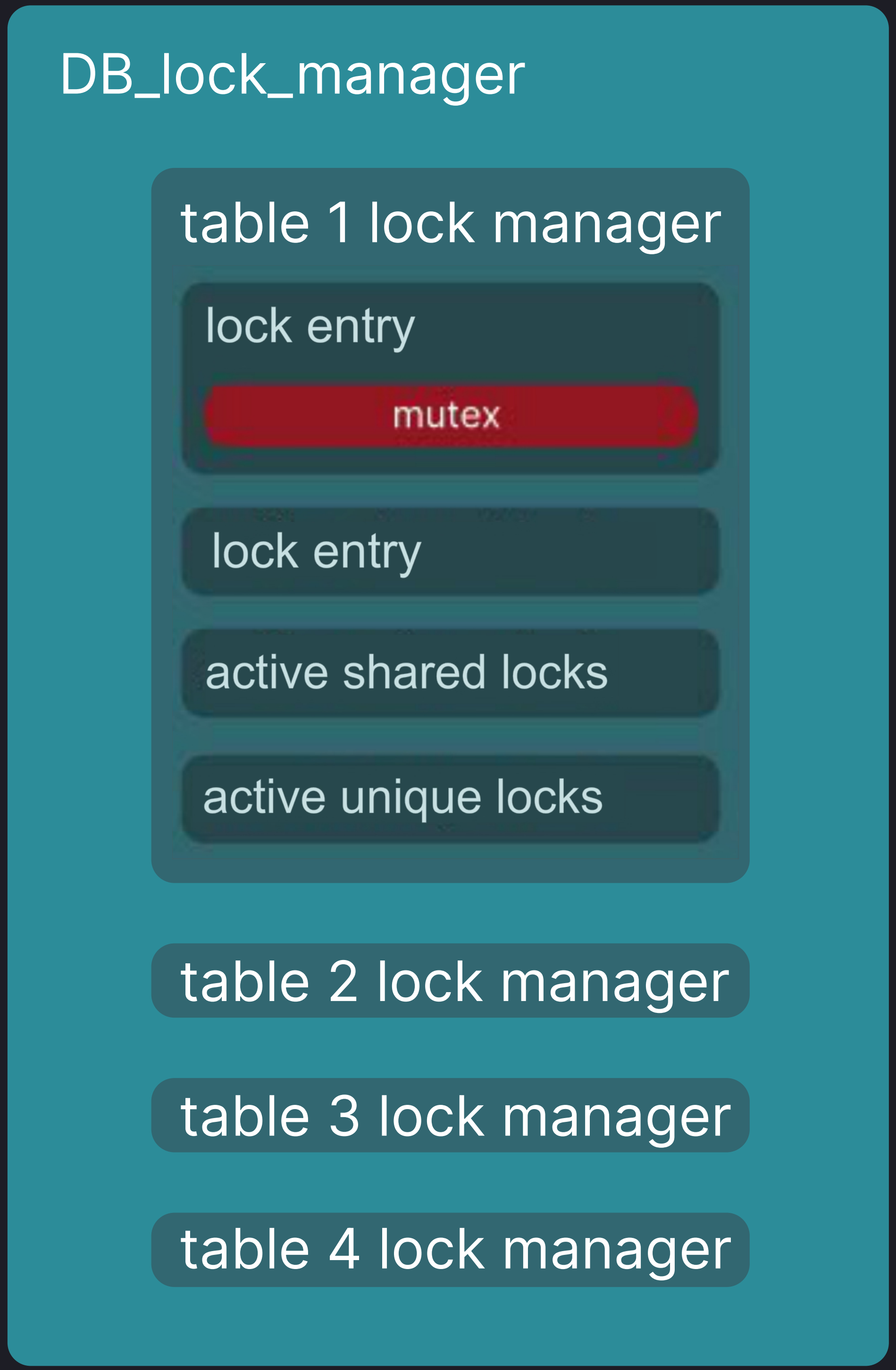
query operation

transaction

transaction

transaction

LOCK MANAGER



DB_lock_manager

table 1 lock manager

lock entry

mutex

lock entry

active shared locks

active unique locks

table 2 lock manager

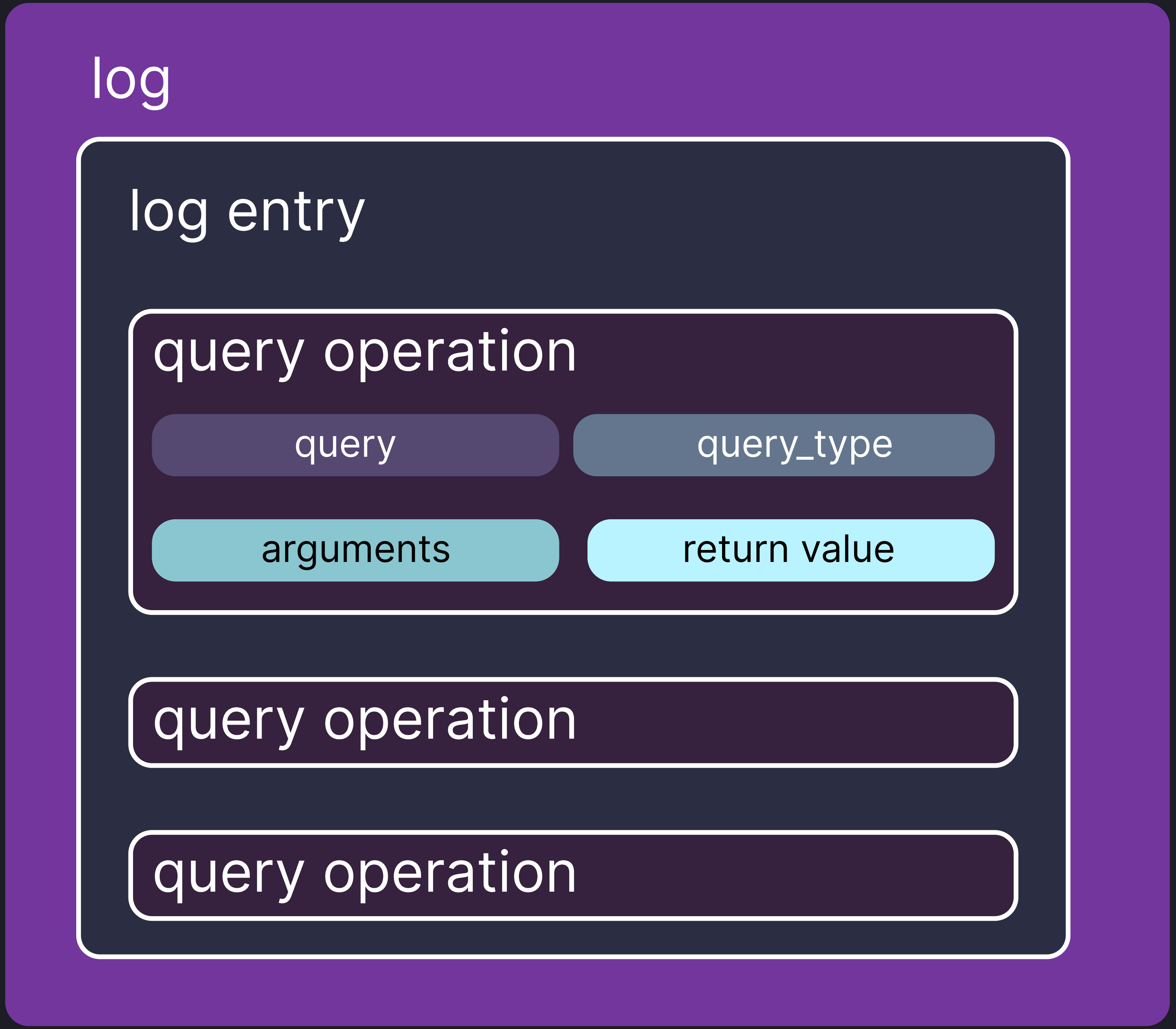
table 3 lock manager

table 4 lock manager

- Lock Manager controls locks for all database tables
- Each table managed by a Lock Manager object
- Entries in Lock Manager identified by record primary keys
- Lock Manager holds vectors for actively locked shared and exclusive locks

QUERIES & LOG

- Log records state of transaction
- Log entry contains queries identified by transaction id
- Contains vector of query operations
- Query return value indicates success or failure
- Successful queries removed
- Failed queries undone, leading to abortion

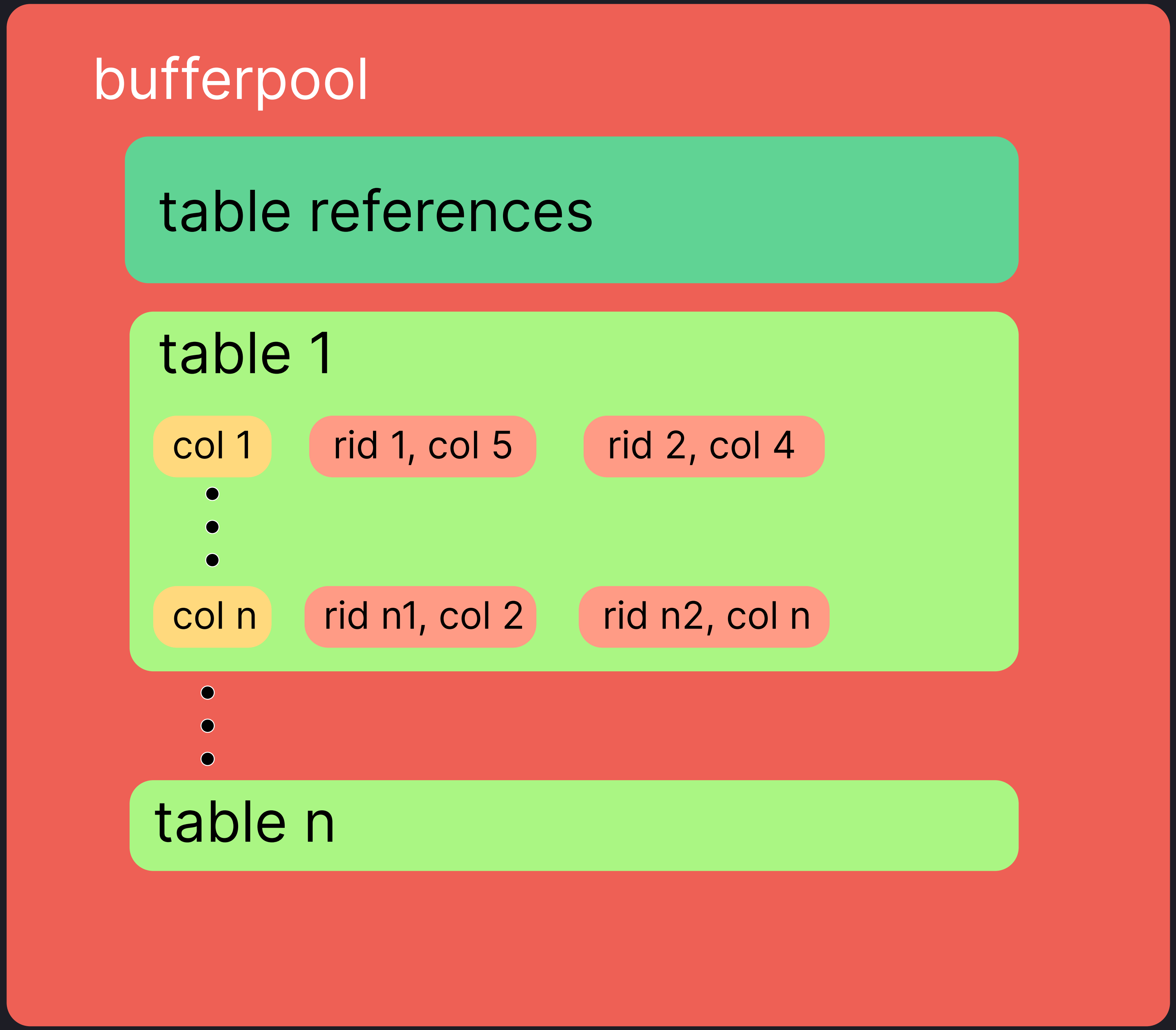


2PL & ABORT

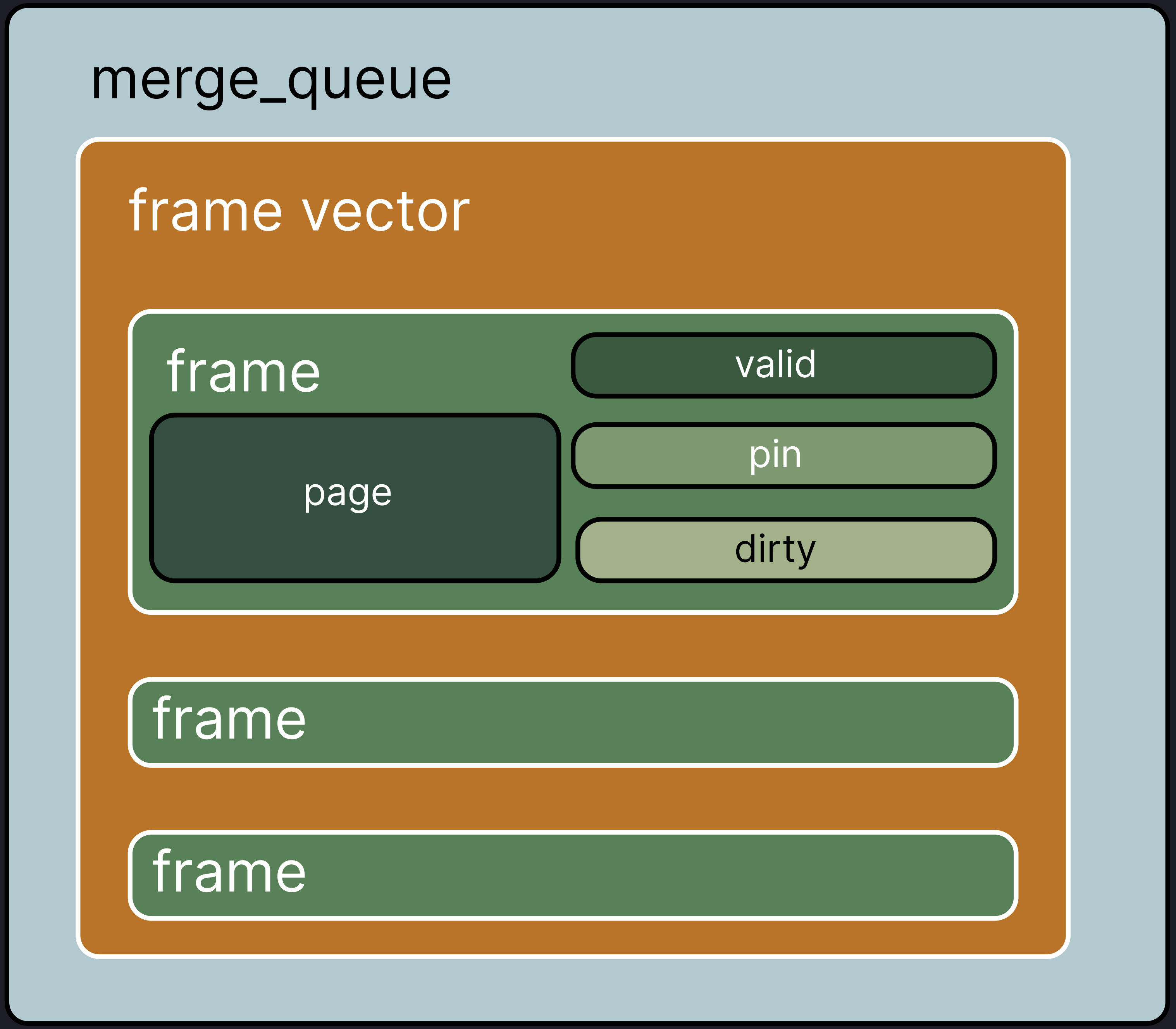
- Abort process undoes insert and update queries
- Insert queries mark records as deleted
- Update queries delete tail record and adjust indirection columns
- Phase two of 2PL unlocks all initially locked records
- Lock manager retrieves and unlocks thread's locks

C++ COMPATIBILITY, FOREIGN KEY

- Reflection library used to determine function calls to wrapper in add_query
- Arguments parsed into constants and vectors, then passed to C++ library
- Foreign key is implemented using a map in bufferpool
- When a table is dropped, the table is found in the map, and columns traversed to modify table's referencing rids
- To avoid accessing a deleted table, the table's rids are deleted from the entries for other tables



MERGE



merge_queue

Modifications:

- No longer merge meta-data columns
- No in place updates
- Merge running on a different transaction thread
- Contention free merge

Thank you