Lazy add/drop columns

Project Overview

Summary: Implement a lazy, non-blocking backend for schema changes that only migrates a tuple to the new version once an update occurs on a new column.

Milestones

1. Support Drop Column - **Done**
2. Support Add (75%) - **Done**
3. Support Lazy Updates (100%) - **Done**
4. Add a compactor to remove old versions (110%) - **Framework in-place**
5. Unsafe ALTER TABLE (Stretch goal, 125%) - **Framework in-place**

Development Status

- SqlTable:
  - Maintains multiple version
  - API refactored to require schema versions
  - Implements more efficient version transformations
- Transactions:
  - Implemented Action framework for processing deferred actions
    - Framework for aborting transactions and cleaning up
  - Implemented Constraint framework for checking unsafe transactions
- Tests & Benchmarks:
  - Sequential correctness tests complete
  - Concurrent tests complete
  - Concurrent benchmarks complete (thanks to Yangjuns!)

Test Coverage & Correctness

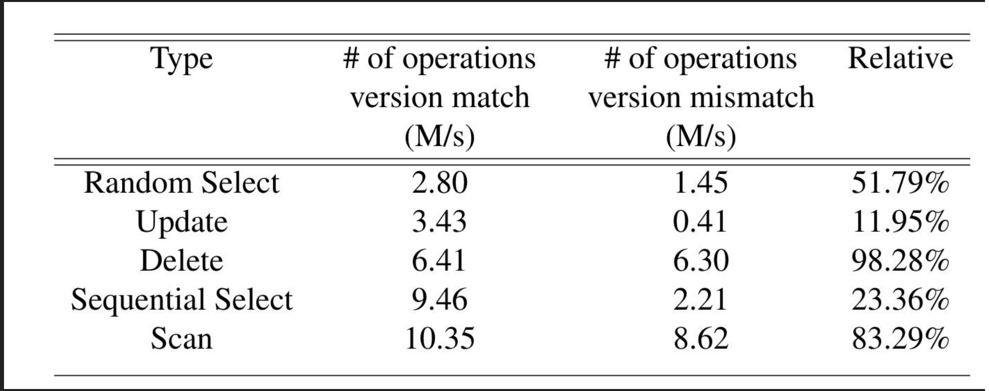
- sql_table.h - 96%
- sql_table.cpp - 96%
- Single-Threaded Tests
  - Insert Test - Update schema, verify inserts go to latest schema
  - Select Test - Update schema, retrieve tuple in old schema version
  - Update Test - Update schema, update tuple in old schema version
  - Scan Test - Scan a table that has tuples in multiple versions
- Concurrent Tests - Performs inserts, selects, and updates concurrently with schema updates and uses scan to verify correctness.

https://codecov.io/gh/yash620/terrier/branch/schema_change

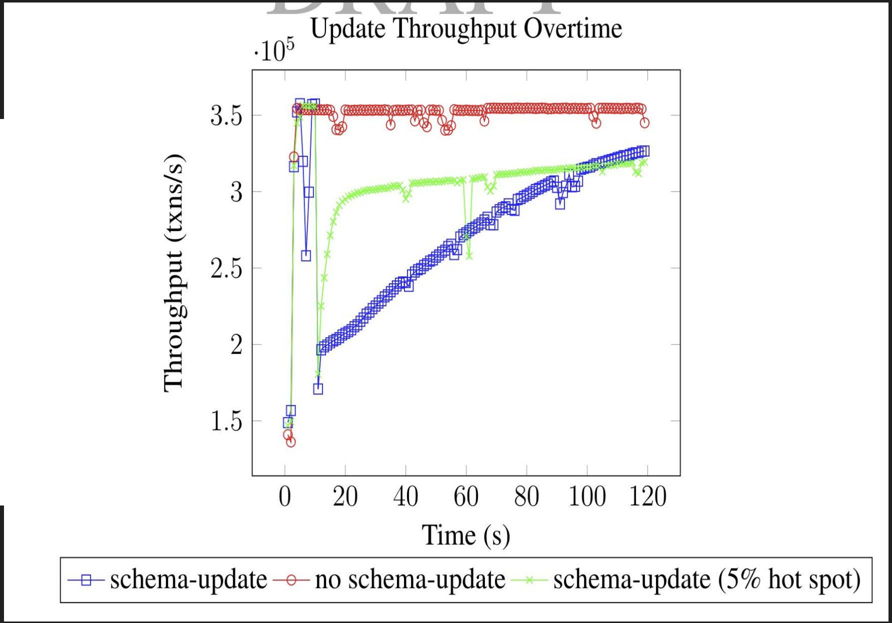
Code Quality Assessment

- Multiversion handling of data - **Strong**
- Tracking schema versions - **Needs review** (choice of backend/container)
- Default values - **Needs review** (better abstraction?)
- Single-threaded tests - **Needs to be refactored** (inherited a hacky codebase)

Benchmarks



Type	# of operations version match (M/s)	# of operations version mismatch (M/s)	Relative
Random Select	2.80	1.45	51.79%
Update	3.43	0.41	11.95%
Delete	6.41	6.30	98.28%
Sequential Select	9.46	2.21	23.36%
Scan	10.35	8.62	83.29%



Future Work

- Finish implementation of compaction:
  - Decide on API
  - Use event framework to implement (deferred scans that delete & insert)
- Finish support unsafe ALTER TABLE:
  - Needs execution engine and catalog
  - Finish implementing rollback (tied to backend/container)