Add/Drop Indexes

Alex, Cal, and Kunal

Progress

- We now support CREATE INDEX!
- Unfortunately, supporting a single-pass CREATE INDEX proved much more difficult than initially expected.
- Some unexpected roadblocks:
    - Concurrent DELETEs and UPDATEs require modifying the index at different times (e.g. at commit time) to ensure that the index contains all the right keys
    - Aborts / Rollbacks complicate the design space as one must deal with inserts that roll back and other such anomalies
    - Allowing the newly created index to be visible to later parts of the same transaction prevented us from using mini-transactions to take advantage of MVCC
- How it works now
    - Shared lock on table is taken by modifiers (so, anyone doing an insert/update/delete)
    - These locks are held until the transaction for the modification ends
    - CREATE INDEX takes the lock in exclusive mode, meaning we know that all modifications have committed / aborted in the version chains for the table
    - We build the index, and then set the index to live

Testing and Benchmarking

- We have some lower level basic tests
- Higher level JDBC tests
  - Correct index creation, correct update, correct delete
- Writing some more tests to verify correctness further

Assessment of Codebase

- We feel that our code is for the most part production level
  - Locking is implemented well without much inefficiency
  - The scan of the table's version chain uses a new method which could be used for other things (such as CREATE INDEX CONCURRENTLY)
- Improvements that need to be made before merging
  - Documentation improvements
  - Pruning of testing facilities (e.g. debug-specific locks)
  - Some small bug fixes
  - More test cases
  - Where should the table locks live. Right now, they are in SqlTable, but they may need to be moved

Future Work

- CREATE INDEX CONCURRENTLY
  - Currently implemented infrastructure:
    - A mechanism for marking an index as created but not yet live
    - A mechanism for iterating through the version chain of a column
    - Infrastructure to pass the CONCURRENTLY flag from the postres parser all the way to the traffic cop
  - Needs to be done:
    - Only allowed in a single statement transaction, meaning we can use multiple transactions to do separate passes before finally setting the index to live
    - Whitebox tests
- Using multiple worker threads to build the index
  - This relies on the worker thread pool implementation
- Allow creating an index while uncommitted changes exist in the same transaction
  - We disallow this to avoid deadlocking with our current locks, but this could be improved
  - We decided to table it because it seems like a silly workload that could be fixed by moving the CREATE INDEX earlier