ACM-DB Lab5

This is the answer document for lab5.

Design Choices

1. Exercise 1:
    - `LockManager.java` to work as the lock granter.
    - `Lock.java` to work as a lock on single page, shared locks and exclusive lock are all maintained inside this class.
2. Exercise 2 & 3 & 4:
    - `BUfferPool` calls `LockManager` to get corresponding locks in `getPage()` and release locks in `TransactionComplete`
3. Exercise 5:
    - Maintained a `DependencyGraph` inside `LockManager.java` to detect dead lock, once a dead lock is detected, throw a `TransactionAbortedException`

Changes to API

I believe None

Missing Components

I believe None

Time Spent

1 week

- Stuck at `DeadLockTest.java`, it took me much time on debugging deadlock management and had to re-implement `LockManager.java` for several times.
- Many bugs in the first implementation of `DependencyGraph` on updating graph and dead lock cycle, much of them appears when a thread completes a `Transaction` and yet another thread fails to change the dependency graph, leading to a thread waiting for an already completed `Transaction` and thus leading to an deadlock even when a thread knows a deadlock will appear. Add some `synchronized()` to ensure correct update of dependency graph.
- Many bugs in the first implementation of `LockManager.java` on lock granting policy, much of them appears when multiple thread tried to complete a `Transaction`. Changed all `HashMap` in `LockManager` to `ConcurrentHashMap` to fix them.
- Found a bug in `BufferPool` when reimplementing `EvictPage` where `LRUList` is not maintained well in previous labs.