LAB 3

Emmanuel J Lopez 1005407

Nicholas Gandhi Peradidjaya 1005295

1. LockManager

LockManager is implemented using two concurrent hashmaps, one for exclusive locks (READ_WRITE) which has a key of the PageId and stores the TransactionId of the lock holder, and the other one is for shared locks (READ_ONLY) which has a key of the PageId and stores a HashSet containing TransactionIds who are lock holders.

2. Special Cases

When inserting tuples if the transaction finds no free slot on the page it is currently searching it will use the unsafeReleasePage method to release it's lock. When it finds a page with a free slot it will upgrade the permission to READ_WRITE.

3. Eviction

The eviction policy is updated to not evict any page that is marked dirty or any page that has a lock on it.

4. Complete Transaction

If a transaction is aborted the pages dirtied by the transaction are discarded and overwritten with the data from the disk. If the transaction is completed the data on the dirtied pages is flushed to the disk.

5. Deadlock

When a transaction acquires a lock for a page, all the transactions with locks on the page are added to depGraph. Before acquirring the lock, the depGraph is checked for any cycles indicating that giving the lock would result in a deadlock. If such a cycle is detected then the transaction is aborted.

All unit test run except for the ReadWriteDeadlockTest under DeadlockTest unit test.