# Lab3 Write Up:

Yuanfeng Li

Feb.26.2021 (2 days late)

**Describe what this lab was about?**

In general, this lab is all about building up a lockManager that can grab, release, hold, and even detect deadlocks. By modifying the BufferPool.java I have achieved No Steal/Force scheme that having locks working on a page granularity level.

I have developed a lock manager that controls the shared and exclusive locks used by the simpleDB. When we have a new transaction, it will grab the lock when it is using getPage() from buffer pool. And the transaction's lock will be released when it commits or aborts. The lock manager have a function to check the deadlock and aborts transaction when the deadlock situation appears as well.

**Unit test for improvement:**

I think they are pretty good.

**Design decisions**

- I have decided to set my locking granularity as page-level shared or exclusive lock.
- I have used the graph to detect the deadlock. Furthermore, when a deadlock appears, I chose to abort the current transaction instead of aborting all other transactions. And if the lock manager finds that there is no deadlock, and transaction cannot successfully grab a lock, it will wait until a lock is released, and then the transaction will try to grab the lock again.

**Discuss and justify any changes you made to the API.**

No changes.

**Describe any missing or incomplete elements of your code.**

Nope