Lab4

Overview:

The primary objective of Lab 4 is to implement log-based rollback and recovery. This entails changing the buffer management policy from FORCE to NOFORCE. In other words, after commit, dirty pages will not be forcibly flushed to disk. This modification grants SimpleDB greater flexibility and efficiency by reducing the need for unnecessary disk output. Additionally, due to the page-level lock management approach, logging also occurs on the entire page. Therefore, when an update operation modifies a page, the after image of the whole page will be recorded in the log.

Components:

- Rollback
    - record type (INT)
    - transaction id (LONG)
    - undo target log start offset (LONG)
    - start offset (LONG)
  - undo: iterate through the log and setting the state of specific pages it updated to their pre-updated state
- Recover: recover the commit state of the while database, redo the committed page and undo the uncommitted page
  - redo: iterate through the log and setting the state of specific pages it updated to their updated state
- Other:
  - BufferPool change to accommodate NO-FORCE policy:
    - evictPage(): remove the dirty-page check before flushing a page to make more space
    - transactionComplete(): add logging of the writeLog. Set before image

All the changes are straight forward, just to add the support for the lock managing system. These change includes add the lock acquire before read a page; Modify the eviction policy to adjust to the NO-FORCE. Since I'm using Least-Recent-Used eviction policy, I use to only evict dirty page. Now I adjust it to able to evict any pages from LRU cache queue) ; flush and release all lock when transaction is completed whether it is committed or abort.

The most difficult part for me is to get familiar with random access file and debug with it. During lab1, 2 I used to use Random access file but only with very simple functions. But I think in order to bug free in this lab, you have to be very familiar with how it works. I once get a very strange error that I'm not able to read a target log position and struggle for couple of days. Thanks to ed post #352. It help me out. When writing to a random access file, the writing process will start from the file pointer's current position. Therefore, if you have finished reading previous log records and wish to write a new one after that always first seek to the current offset and then write the log record.

I passed 9/10 of the tests in LogTest. Currently failed on TestAbortCrash test, it strangely find a tuple that should present. I tried to print out the tuple but I didn't find anything wrong with it(It should present but still give an error):

```
    [junit] Testcase: TestAbortCrash took 0.03 sec
    [junit]     Caused an ERROR
    [junit] LogTest: tuple present but shouldn't be
    [junit] java.lang.RuntimeException: LogTest: tuple present but shouldn't be
    [junit]     at simpledb.systemtest.LogTest.look(LogTest.java:58)
    [junit]     at simpledb.systemtest.LogTest.TestAbortCrash(LogTest.java:271)
    [junit]     at java.base/
jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:104)
```

I'm still debugging on it. Not sure if I can make it before the ddl.