Report of Lab 5

Yuxuan Duan 516030910573

1 Design Decisions

1.1 Lock

My buffer pool uses page-level locks. I did not use anything special to implement the locks. Instead, they are just instances of an inner class *PageLock* of the buffer pool containing its transaction ID, page ID and lock type (shared or exclusive).

And the information of which transaction holds which locks and which page is locked by which transactions is kept in maps by the buffer pool.

1.2 Deadlock Detection

I used a simplest way of resolving deadlocks: the timeout policy. After *wait()* is invoked, a thread will wait at most $WAIT\_TIME$ milliseconds or otherwise notified by *notifyAll()*. However, from the first time it started to wait until now, if it has been waiting for more than

$$ABORT\_BASE\_TIME + \text{random}(0, ABORT\_VAR\_TIME)$$

milliseconds, we will judge that a deadlock occurred, and the transaction run by this thread will be aborted.

In my implementation, $WAIT\_TIME = 50$, $ABORT\_BASE\_TIME = ABORT\_VAR\_TIME = 200$. A random increment to the abort time is necessary because two threads running the same code may ask for the same locks at the same time. If the abort time is fixed, they may be aborted together, restart together and aborted together again.

1.3 Other Minor Designs

Some designs are used to deal with special situations.

- Assume that a page was first required its exclusive lock by a transaction. Then it was modified and was about to be marked dirty. However, possible situation that this page was evicted before marked dirty may happens, and *markDirty()* or *flushPages()* may fail. Therefore, I double checked the page when I was about to mark it dirty. If this page is in the buffer pool, I will do nothing. And if this page has been evicted, put the dirty page into the buffer pool. (Cannot read from the disk because the page on the disk has not been modified.)

- Some pages may be modified, and then the transaction was aborted before marking these pages dirty. So if a transaction is aborted, we cannot judge which pages should be recovered by *isDirty()*. Instead, we should discard all the pages whose exclusive locks are held by this transaction, for they are all possibly dirty.

2 API Changes

No API was changed in this Lab.

3 Time Spent and Difficulties

I roughly spent 18 hours on this Lab. This Lab confronts me with multi-thread programming which is relatively unfamiliar to me. So it took me quite much time to think about the way to design the buffer pool and to debug.

However, discussing with my classmates was interesting. In discussion, Yaoyao Ding helped me a lot with some insightful ideas, so here I want to express my gratitude to him.

Also I want to thank you for designing these labs. It is a good way to both get a macro view and know the details of a database system. Trying on our own is always better than just listening in class.