SUTD 2021 50.043 SimpleDB Project Part 3 Writeup Report Document

> James Raphael Tiovalen / 1004555

Implementation Description

For part 3 of the project, I also simply implemented most of the skeleton methods/functions and classes specified by the handout. I also followed the implementation guide as laid out by the exercises closely, and hence, there are very few deviations from the intended path.

Several design decisions that I have made include:

- Implementing a `LockManager` class, as specified by the handout, which is used by the `BufferPool` class.
- Implementing strict two-phase locking at page granularity.
- Implementing a NO STEAL/FORCE buffer management policy.
- Implementing a deadlock detection method by building and using a wait-for graph and then detecting cycles in said dependency graph.

Several challenges that I have faced include:

- Implementing certain methods in the `LockManager` class with the specific Java oddities/eccentricities. Some methods exhibit weird behavior such as deadlocking if the method is `synchronized` for some reason. Race conditions and various `ConcurrentModificationException`-related errors had to be taken care of as well during development.
- Even when my code passes all the tests in the `DeadlockTest` unit test, it might still very rarely time out when running the `TransactionTest` system test (not the unit test). I was not able to figure out the cause (since deadlocks have been handled properly) and most of the time, it passes said system test anyway. To solve this, we can just simply re-run the `TransactionTest` system test (it might have something to do with how threads and processes are handled and garbage collected in Java, as the error might occur when re-executing the system test repeatedly in a quite rapid manner, although I am not sure).

I did not change the provided API at all. All the custom, additional classes are designed around the current API to support it, instead of subverting, modifying, or redirecting it.

There are no missing or incomplete elements of my code, assuming that all that is being considered is for part 3. Any other extra additional methods will be implemented in future parts, since this provides ease of debugging and separation between different functionalities as the project timeline progresses.