CS 6530 Database Systems

Project 5: SimpleDB Transactions

Team 15 - Tanvi Gangadhar(u1205740), Greeshma Mahadeva Prasad (u1141804)

DECISION DESIGN

Granting Locks

- We are implementing a NO STEAL/FORCE Buffer mangement policy
- Strict two phase locking has been implemented
- Locking has been done at page granularity and not tuple/table granularity
- Transactions that cannot be granted a lock will be blocked
- We are acquirng the lock when the page is first requested from BufferPool before we read or modify them. Hence we are not acquiring lock in any of the other operators.

Lock Lifetime

- We acquire a shared lock on any page before we read it, and acquire an exclusive lock on any page before we write it. we are using the Permissions objects in the BufferPool that were already being passed around
- We had already implemented a call to markDirty() on any of the pages accessed in insertTuple() and deleteTuple() for project 2

Implementing NO STEAL

- We have modified the evictPage() method to not evict pages which are marked dirty. We got rid of our LRU stack to make this implementation simpler. We are implementing MRU instead.
- We throw a DbException if all pages in the BufferPool are dirty

Transactions

- The Transaction Handler class has five maps to store the following - all shared locks on a page, all exclusive locks on a page, all pages associated with a transaction, all locks on an object and the depedencies of every transaction.
- We are using Breadth First Search to populate the dependency graph map

Deadlocks and Aborts

- We have implemented cycle-detection in a dependency graph data structure.
- We check for cycles in the dependency graph whenever there is an attempt to grant a new lock, and abort if a cycle exists.

API CHANGES

We have not made any changes to the provided API. We have implemented a new class called TransactionHandler.java for this project.

MISSING/INCOMPLETE ELEMENTS
All the unit and system test cases passed, we believe we have implemented everything needed as per project 5 requirements.

TIME SPENT
Similar to previous projects, we pair programmed most of part of the project and together we spent about 20 hours over a few days. The breakup is as shown below,

- Understanding Transactions, Locking, and Concurrency Control and the general structure - around 3 hours
- Granting Locks - 2 hours
- Lock Lifetime - 2 hours
- Implementing NO STEAL- 5 hours
- Transactions- 3 hours
- Deadlocks and Aborts - 5 hours

DIFFICULTIES ENCOUNTERED
We had difficulty detecting and managing race conditions among threads while accessing files in Btree and HeapFile classes