Khai Tran

CSE 444

Lab 4 WriteUp

This lab 4, we implement log-based rollback for aborts and log-based crash recovery. All of transactions activities of database will be recorded to this LogFile.java in disk, and it uses to rollback transaction or in case crash the short-term memory, we can recover all the transaction committed which have not save to disk. Therefore, log is very importance to database to prevent lose data while transaction committed, but it is not saved to disk.

We changed the code in BufferPool.flushPage() before your call to writePage(p), where p is a reference to the page being written. When we test the LogFile data then flushAllPages is called then committed and set a page as dirty.

Implement rollback to LogFile.java from what we know is transaction ID. We use a transaction we know, use it to find the position of the Page in the disk through a random-access file (raf). We use no-steal and force a log file, and we needs to use to rollback the transaction. The transactions should be rollbacked and iterated over the log file for all update the page with transaction the before image of the page. For convenience, we call a rollback method with long parameter and we can reuse it for recover method. We implement while loop to get the type of transaction and transaction ID to update new Pages with the transaction to BufferPool and discard the Pages was changed or committed, but we will skip the Pages that in CHECKPOINT_RECORD, and keep going to entire data in the data log in the disk to rollback.

Implement recover with we have nothing to know about the data such as the loss of BufferPool, Dirty Page Table, but what we know is the data log on disk after the whole system data crashes. The transactions with committed before the crash should have their updates saved on disk, and transactions with aborted or did not commit should update undone. The log file data iterated all update are redone and a set of loser transactions that did not commit kept maintained. Now, we read from data log on disk and load the transactions to the transaction table, BufferPool, Dirty Page Table. Now, we implement while loop to read the type of transaction and transaction ID to recover all the transactions on disk. First, if we meet the type is UPDATE_RECORD, we load the update Page from disk to BufferPool through Catalog. Second, if type is COMMIT_RECORD, we just remove the transaction ID from transaction table. Then, if there is no check point, we will put transaction ID with file pointer to log data and add transaction ID to transaction table, otherwise we will skip that record with a LONG_SIZE. If type is the ABORT_RECORD, we just remove the transaction ID from transaction table and call the rollback function with a Long to rollback those transactions without submitted. At the end, we also rollbacked all transactions without check point and submitted.

There are so many errors that I had made because I did not know how the structure of Log File stored data on disk. For the rollback, it took very long for me to put the right type to update the record to pass the system test. The recover method with me was extremely hard because I did not have transaction ID like rollback method. I did not know how to get the transaction ID and

FilePointer to the set, so I could make the recover operation. Moreover, when I finished the recover the test did not pass with abort test. I had to change from switch() to if() else to pass all the test, it took me two days to figure out where was the errors. At the end know that the switch() sometime had errors with the TestOpenCrash() in system test, but it alright with if() else in the iterator part.