Name: Vishnu Gopal Rajan

Team No: 12

ID: 1001755911

DBMS Models and Implementation
Project 2 Report

<u>Overall Status</u>

Zgt_tm.c:

- In this file we were asked to write code for the TxRead, TxWrite, CommitTx and AbortTx. We had to update the parameters of the node for each of these operations such as tid, obno, Txtype, count and status. For read and write the default obno is 0 whereas for abort and commit the obno is set to -1. The SEQNUM is a counter which keeps track of the number of operations and for each operation it gets decremented.

Zgt_tx.c:

- In this file we had to implement the basic functions of readtx, writetx, committx, aborttx and do_commit_abort.
- In readtx and writetx, we had to start the operation using the tid and count, upon which the transaction manager had to be locked for its use. From the transaction manager we had to get the status of the transaction to check if it was an active transaction or inactive transaction. If the transaction was an active transaction then we had to set lock on it either readlock or writelock. Once this was done we had to call finish operation and exit.
- In committx and aborttx we had to lock the transaction manager and check if the current transaction with the given tid either existed in the hash table or not. If it did exist in the hash table then we had to either commit or abort it based on the requirement by calling do_commit_abort. If it didn't not exist in the hash table then it we don't have to bother about the transaction.
- In do_commit_abort function we append the status to the logfile. Once this was done we had to get the respective tid and free the locks used by that tid.

Once this was done we had to check for the semaphores used by the transaction and check if any other transaction is waiting for them, if they are then we need to free the semaphores as well.

- Set_lock function is basically used to set lock on the transactions being used actively. We find the transaction on the hash table using the tid, sgno and obno. We first check if the same transaction is requesting for the object, if yes then we let the transaction use the object. Else we find the node of the transaction and check if it has a lock on it. If there exists a lock then the transaction has to wait before it can proceed with the operation. If the lock is cleared then the transaction can proceed, where it is set to active and the lock is set on it. Another case is if the node we checked is empty, then we need to get the status of the transaction. If it is greater than -1 then we can add to the hash table, else it cannot be added to the table.
- Perform_readwrite function is used to write or append the values to the log file. It basically adds the write and read operations along with its functions to the log file.

Difficulty Experienced:

- Basically understanding the functions already created and objects already in use was a challenge. Also after understanding them, putting them to use was another big challenge. I used jetbrains Clion IDE which had a prompt so this made it a bit easy to call the correct function and objects.
- The logic in order to check for transactions waiting for semaphores was also a big challenge during the do_commit_abort function. After releasing the locks we also had to check if the semaphores used have any other transaction waiting for them.
- Another difficulty I experienced was keeping a track of the semaphores pool during setting the locks and making the transaction wait.
- I wasn't able to run the RR testfile. It also cannot be deleted or modified on the omega server and persists no matter what.
- I wasn't able to run the tmtest command either to run the transactions 5000-10000 times.

File Descriptions:

- In the function set_lock we had to keep a check on the current transaction in use and the new transaction as well. In this function I had to create a curTxn and curNode to keep track of these transactions.
- I'm also making use of semNum to keep track of the semaphores in the semaphore pool in order to provide the required semaphore to the waiting transaction.
- In other functions such as readTx and writeTx I have created curTid to retrieve the value of tid
  and getTid in commit and abort.

Division of Labor: This project was completed by myself.

Logical Errors:

- One of the errors I committed was no taking into account the possibility of the node being empty, which made the transactions run incorrectly. This was corrected by if statement to check for this condition and update the table accordingly
- Another logical error was not releasing the resources using zgt_v(0). This made it give incorrect sequences. I corrected this by releasing the resources after the transactions were done using them.
- I also did not take into account the need for releasing the semaphores after aborting or ending a transaction. I corrected this error by using zgt_v(txPtr->semno).