2238-CSE-5331- 002

DBMSMODELS AND IMPLEMENTATION

Fall 2023

Group number: 10

Sri Harsha Kolli – 1002063354

Abhinay Reddy Gurrala - 1002058438

Project:

Implementation of a Transaction manager

Overall Status:

We started the project by understanding the given skeleton code and reading the files we had to work on (zgt_tm and zgt_tx). In zgt_tm.c we have completed writing the code for BeginTx, TxRead, TxWrite. CommitTx, AbortTx . After that we wrote the code for zgt_tx.c :- readtx(), writetx(), aborttx(), committx(), do_commit_abort(), set_lock().

File: Zgt_tm.C:

TxRead() – This method is used to perform read operation. This creates the thread and calls the method readtx() in the zgt_tx.c file, this allows to perform read operation.

TxWrite() – This method is used to perform write operation. This creates the thread and calls the method writetx() in the zgt_tx.c file, this allows to perform write operation.

CommitTx() – This method is used to perform commit operation. This creates the thread and calls the method committx() in the zgt_tx.c file, this allows to perform commit operation.

AbortTx() – This method is used to perform abort operation. This creates the thread and calls the method aborttx() in the zgt_tx.c file, this allows to perform abort operation.

File: Zgt_tx.C:

readtx() – This method is used to read data from a transaction. The set lock() method is used by the read operation to check the object's lock status. If the object is not currently being used by another transaction, the read operation is carried out after locking the object.

writetx() - This method is used to perform a transaction's write operation. The write operation uses the set lock() method to check the lock status of the object, and if it is not being utilized by another transaction, it conducts the read operation after locking the object.

aborttx() – This method is used to perform abort operation on the transaction. The abort operation releases all locks held by the transaction through the free_locks() function.

committx() – This method is used to perform commit operation on the transaction. This commits the transaction and releases the locks using free_locks() function.

do_commint_abort() – This is method is used to perform the commit and abort operation based upon the given conditions in the code.

set_lock() – This method is used for locking transaction before doing an operation. This checks for multiple conditions and sets the lock accordingly to the transactions.

Difficulty:

The implementation of the thread idea for semaphore was both interesting and demanding. Furthermore, deadlock conditions in set_lock() method was extremely difficult, requiring a significant amount of time and effort to analyse. The initial setup took some time, and the parameters specified in the make file were required to successfully compile the given code.

File Descriptions:

Added a new test file – mytest.txt for debugging and fixing errors.

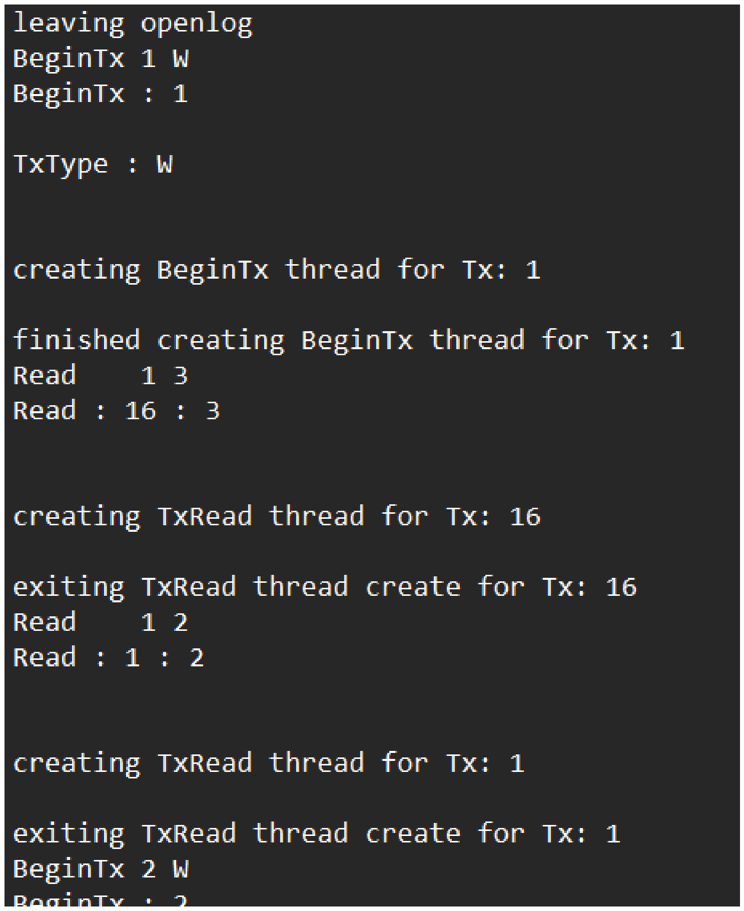
Division of Labour:

We have separated the code according to the methods that we are required to write. The methods readtx(), writetx(), aborttx() are done by Sri Harsha Kolli (1002063354) and the methods committx(), do_commit_abort(), set_lock() are done by Abhinayreddy Gurrala (1002058438).

We have spent around two weeks for the project and worked around 2-3 hours each day. We either met each other or connected through online and implemented the individual parts of our codes so that we can help each other.

Logical errors and handling them:

1. Error reading from file when executing the same test file multiple times. In the image below, the TID is read as ‘16’ which is invalid.



```
leaving openlog
BeginTx 1 W
BeginTx : 1

TxType : W


creating BeginTx thread for Tx: 1

finished creating BeginTx thread for Tx: 1
Read    1 3
Read : 16 : 3


creating TxRead thread for Tx: 16

exiting TxRead thread create for Tx: 16
Read    1 2
Read : 1 : 2


creating TxRead thread for Tx: 1

exiting TxRead thread create for Tx: 1
BeginTx 2 W
```

This is the primary reason why many of the test cases are hanging.

2. Segmentation fault (code dumped): This problem arose because of incorrect locking and unlocking of semaphores. The incorrect transaction numbers from the file also produced this error.
3. Inappropriate thread control led to the disruption of atomicity attribute of the transactions. Added blocking functions wherever necessary and this fixed the issue.