**Project Report**

Implementation of Transaction Manager

**Contents:**

|  |  |  |
| --- | --- | --- |
| S.No |  | PageNo |
| 1 | Team Details | 3 |
| 2 | Overall Status | 3 |
| 3 | Files Description | 5 |
| 4 | Division of work | 5 |
| 5 | Logical errors | 6 |
| 6 | Output | 6 |

**Team Details**:

|  |  |  |
| --- | --- | --- |
| S.No | Name | UtaId |
| 1 | Muthyam Pruthvi Raj | 1001400715 |
| 2 | Yalamanchili Vamsi Krishna | 1001554490 |

**Overall Status**:

We have implemented readtx(), writetx(), committx(), aborttx(), set\_lock(), perform\_readWrite(), do\_commit\_abort() methods in zgt\_tx.c and WriteTx(), AbortTx() and CommitTx() methods in zgt\_tm.c files. We implemented the methods as follows:

**Begin the transaction (begintx())**:

A thread is created using the fork system call in the BeginTx() function of the transaction manager (zgt\_tm.c). This thread is passed to the begintx() function of the zgt\_tx.c transaction program and this function sets the status of transaction to Active and also creates a new transaction node to make use of the transaction id, count. To keep track of the transactions, it writes to the log file. Then the transaction manager gets lock by the function to perform the operations. The lock is released once the operations are completed and then the thread exits.

**Read operation (readtx())**:

We check the status of the transaction. We perform read operation, if the transaction state is Active by setting the lock in Shared mode. After the lock is obtained we perform the read operation by decrementing the object value. Then we release the transaction manager resources and exit the thread that is handling the current transaction.

**Write Operation(writetx()):**

Write transaction is similar to read transaction, we check the status of the transaction. We perform write operation, if the transaction state is Active by setting the lock in Exclusive mode. After the lock is obtained we perform the write operation by incrementing the object value. Then we release the transaction manager resources and exit the thread that is handling the current transaction.

**Commit Operation(committx()):**

This method is used to commit transactions. Using a pointer we retrieve the current transaction and then we check the status of the transaction. If the value of the pointer is not NULL, then we invoke do\_commit\_abort() method to perform required operation.

In do\_commit\_abort() method, we append the current operations by opening the log file. And then the status of the transaction is retrieved and updated. The transaction locks are then released and then remove the transaction.

**Abort Operation(aborttx()):**

It performs the abort operation. It is similar to the commit operation , call the do\_commit\_abort() method and abort the operation.

**Setlock Method(set\_lock())**:

The locks for the transaction to perform the read and write operation are acquired using this set\_lock() method. This method is used in readtx() and writetx() methods.

Initially, we retrieve the current owner of lock for the requested object. We change the status of the transaction to TR\_ACTIVE mode.

- If the owner that requested lock is same and then we grant the lock.

- If the current transaction is read only transaction and the other transaction that has lock on the object with ‘S’ and is read only transaction, then the transaction is allowed to get a lock on that object.

-If the Transaction has lock on object in the ‘X’ then we make the current transaction to wait.

**Files Description**:

No new Files were created during the implementation of the program.

**Division of work**:

Initially we learned the how the Transaction Manager handles the transactions. Both of us worked together and then we got to learn about the overall organization of the transaction manager data structures and hash table objects. Then we divided the work and implemented the project.

Pruthvi Raj M

* Came up with the algorithm.
* Coding part for zgt\_tm.c, and zgt\_tx.c.
* Documenting the code.

Vamsi Krishna Y

* Came up with the algorithm.
* Coding part for zgt\_tm.c, and zgt\_tx.c.
* Report.

**Logical errors**:

* Program was hanging many times while executing multiple times in succession. Problem was with the transaction not being removed after doing zgt\_v(tid) of the transaction. After we corrected that program worked fine.
* We had many difficulties in implementing the set\_lock method to follow every case of the algorithm. We read the algorithm description and followed from there to implement in code.
* Program was hanging while performing commit and abort operations as we were setting a value for objno which we don’t need in case of both commit and abort.

**Output**:

[pxm0715@omega src]$ make clean  
rm -f \*.o \*~ zgt\_test  
[pxm0715@omega src]$ make  
/usr/bin/g++  -I/home/p/px/pxm0715/project2//include -I. -L/usr/lib -DTX\_DEBUG -DTM\_DEBUG -DHT\_DEBUG -c zgt\_test.C  
/usr/bin/g++  -I/home/p/px/pxm0715/project2//include -I. -L/usr/lib -DTX\_DEBUG -DTM\_DEBUG -DHT\_DEBUG -c zgt\_tm.C  
/usr/bin/g++  -I/home/p/px/pxm0715/project2//include -I. -L/usr/lib -DTX\_DEBUG -DTM\_DEBUG -DHT\_DEBUG -c zgt\_tx.C  
/usr/bin/g++  -I/home/p/px/pxm0715/project2//include -I. -L/usr/lib -DTX\_DEBUG -DTM\_DEBUG -DHT\_DEBUG -c zgt\_ht.C  
/usr/bin/g++  -I/home/p/px/pxm0715/project2//include -I. -L/usr/lib -DTX\_DEBUG -DTM\_DEBUG -DHT\_DEBUG -c zgt\_semaphore.C  
/usr/bin/g++ -lpthread  -DTX\_DEBUG -DTM\_DEBUG -DHT\_DEBUG -I/home/p/px/pxm0715/project2//include -I. zgt\_test.o zgt\_tm.o zgt\_tx.o zgt\_ht.o zgt\_semaphore.o -o zgt\_test  
[pxm0715@omega src]$ ./zgt\_test ../test-files/S2T.txt  
// serial history  
// serial history  
// 2 transactions  
// 2 transactions  
// same object accessed  
// same object accessed  
// multiple times  
// multiple times  
Log S2T.log  
Log file name:S2T.log  
  
leaving openlog  
BeginTx 1 W  
BeginTx : 1  
  
TxType : W  
  
  
entering BeginTx  
  
leaving BeginTx  
Read    1 1  
Read : 1 : 1  
  
  
entering TxRead  
  
leaving TxRead  
Read    1 2  
Read : 1 : 2  
  
  
entering TxRead  
  
leaving TxRead  
Write   1 3  
Write : 1 : 3  
  
  
entering TxWrite  
  
leaving TxWrite  
Write   1 4  
Write : 1 : 4  
  
  
entering TxWrite  
  
leaving TxWrite  
read    1 1  
Read : 1 : 1  
  
  
entering TxRead  
  
leaving TxRead  
write   1 2  
Write : 1 : 2  
  
  
entering TxWrite  
  
leaving TxWrite  
write   1 4  
Write : 1 : 4  
  
  
entering TxWrite  
  
leaving TxWrite  
write   1 4  
Write : 1 : 4  
  
  
entering TxWrite  
  
leaving TxWrite  
commit 1  
Commit : 1  
  
  
entering TxCommit  
  
leaving TxCommit  
begintx 2 W  
BeginTx : 2  
  
TxType : W  
  
  
entering BeginTx  
  
leaving BeginTx  
read    2 5  
Read : 2 : 5  
  
  
entering TxRead  
  
leaving TxRead  
write   2 5  
Write : 2 : 5  
  
  
entering TxWrite  
  
leaving TxWrite  
write   2 6  
Write : 2 : 6  
  
  
entering TxWrite  
  
leaving TxWrite  
read    2 6  
Read : 2 : 6  
  
  
entering TxRead  
  
leaving TxRead  
commit  2  
Commit : 2  
  
  
entering TxCommit  
  
leaving TxCommit  
  
  
:::Hash node with Tid:1, obno:4 lockmode:X removed

[pxm0715@omega src]$ ./zgt\_test ../test-files/deadlock.txt  
// 2 transactions  
// 2 transactions  
// generates a deadlock  
// generates a deadlock  
// will hang w/o deadlock resolution  
deadlock  
Log deadlock.log  
Log file name:deadlock.log  
  
leaving openlog  
BeginTx 1 W  
BeginTx : 1  
  
TxType : W  
  
  
entering BeginTx  
  
leaving BeginTx  
BeginTx 2 W  
BeginTx : 2  
  
TxType : W  
  
  
entering BeginTx  
  
leaving BeginTx  
Read    1 1  
Read : 1 : 1  
  
  
entering TxRead  
  
leaving TxRead  
Read    2 2  
Read : 2 : 2  
  
  
entering TxRead  
  
leaving TxRead  
Write   1 2  
Write : 1 : 2  
  
  
entering TxWrite  
  
leaving TxWrite  
Write   2 1  
Write : 2 : 1  
  
  
entering TxWrite  
  
leaving TxWrite  
Commit  1  
Commit : 2  
  
  
entering TxCommit  
  
leaving TxCommit  
commit 2  
Commit : 1  
  
  
entering TxCommit  
  
leaving TxCommit

[pxm0715@omega src]$ ./zgt\_test ../test-files/C2Tsz.txt  
// serializable history  
// serializable history  
// conflicts but total order  
order  
log C2Tsz.log  
Log file name:C2Tsz.log  
  
leaving openlog  
BeginTx 1 W  
BeginTx : 1  
  
TxType : W  
  
  
entering BeginTx  
  
leaving BeginTx  
Read    1 3  
Read : 1 : 3  
  
  
entering TxRead  
  
leaving TxRead  
Read    1 2  
Read : 1 : 2  
  
  
entering TxRead  
  
leaving TxRead  
BeginTx 2 W  
BeginTx : 2  
  
TxType : W  
  
  
entering BeginTx  
  
leaving BeginTx  
Read    2 1  
Read : 2 : 1  
  
  
entering TxRead  
  
leaving TxRead  
Write   2 3  
Write : 2 : 3  
  
  
entering TxWrite  
  
leaving TxWrite  
Write   1 3  
Write : 1 : 3  
  
  
entering TxWrite  
  
leaving TxWrite  
Write   1 2  
Write : 1 : 2  
  
  
entering TxWrite  
  
leaving TxWrite  
Write   2 2  
Write : 2 : 2  
  
  
entering TxWrite  
  
leaving TxWrite  
Commit  1  
Commit : 2  
  
  
entering TxCommit  
:::ERROR:node with tid:1 and onjno:2 was not found for deleting:::ERROR:node with tid:1 and onjno:3 was not found for deleting  
leaving TxCommit  
commit 2  
Commit : 1  
  
  
entering TxCommit  
  
leaving TxCommit  
  
  
:::Hash node with Tid:2, obno:1 lockmode:S removed

[pxm0715@omega src]$ ./zgt\_test ../test-files/NoC2T.txt  
// serializable history  
// serializable history  
// 2 transactions (no conflicts)  
conflicts)  
// same object accessed  
// same object accessed  
// multiple times  
// multiple times  
Log NoC2T.log  
Log file name:NoC2T.log  
  
leaving openlog  
BeginTx 1 R  
BeginTx : 1  
  
TxType : R  
  
  
entering BeginTx  
  
leaving BeginTx  
Read    1 1  
Read : 1 : 1  
  
  
entering TxRead  
  
leaving TxRead  
Read    1 2  
Read : 1 : 2  
  
  
entering TxRead  
  
leaving TxRead  
BeginTx 2 W  
BeginTx : 2  
  
TxType : W  
  
  
entering BeginTx  
  
leaving BeginTx  
Read    2 8  
Read : 2 : 8  
  
  
entering TxRead  
  
leaving TxRead  
Read    2 7  
Read : 2 : 7  
  
  
entering TxRead  
  
leaving TxRead  
Write   2 6  
Write : 2 : 6  
  
  
entering TxWrite  
  
leaving TxWrite  
Write   2 5  
Write : 2 : 5  
  
  
entering TxWrite  
  
leaving TxWrite  
Commit  2  
Commit : 2  
  
  
entering TxCommit  
  
leaving TxCommit  
read    1 3  
Read : 1 : 3  
  
  
entering TxRead  
  
leaving TxRead  
read    1 4  
Read : 1 : 4  
  
  
entering TxRead  
  
leaving TxRead  
Commit  1  
Commit : 1  
  
  
entering TxCommit  
  
leaving TxCommit  
  
  
:::Hash node with Tid:2, obno:5 lockmode:X removed  
  
:::Hash node with Tid:2, obno:6 lockmode:X removed  
  
:::Hash node with Tid:2, obno:7 lockmode:S removed  
  
:::Hash node with Tid:2, obno:8 lockmode:S removed  
  
:::Hash node with Tid:1, obno:4 lockmode:S removed  
  
:::Hash node with Tid:1, obno:3 lockmode:S removed  
  
:::Hash node with Tid:1, obno:2 lockmode:S removed  
  
:::Hash node with Tid:1, obno:1 lockmode:S removed

`