**Project 1: Implementing 2PL protocol**

**Intermediate Report**

**Team Details:**

Ridham Kothari (1001446086)

Yash Shah (1001563334)

**Pseudo-code for the program**:

#main function

**function main:**

*InputFile* = Read Given file line by line

*timestemp* = 0 #for giving timestemp to transactions

For every line in *inputFile* as *operation*:

Parse(*operation*)

end for

**end main**

# begin transaction method

**function begin\_Transaction (*timestemp, tranNum*):**

insert transaction into transaction table with *tranNum*, *timestamp* and *status* as ‘active’

**end begin\_transaction**

# read item method

**function read\_item (*tranNum, dataItem*):**

If *dataItem* is already lock:

If *dataitem.lock* = ‘read’:

Allow transaction to read data item and

Append *tranNum* into *tranNumHolding* field of lock table

If *dataitem.lock* = ‘write’:

Wait\_die(*tranNum*,*dataItem*)

Else:

Allow transaction to read data item and

insert entry into lock table with *dataItem* as *dataItem*, *tranNum* as *tranNum* and lock as ‘read’

**end read\_item**

# write item method

**function write\_item(*tranNum,dataItem*):**

if *dataitem* is already lock:

If *dataItem* is already lock by *tranNum*:

If *dataitem.lock* = ‘’read”:

Allow transaction to write data item and

Upgrade lock in lock table as ’write’ for data item

Else:

Wait\_die(*tranNum,dataItem*)

Else:

Allow transaction to write data item and

insert entry into lock table with *dataItem* as *dataItem*, *tranNum* as *tranNum* and lock as ‘write’

**end write\_item**

# wait and die method for deadlock

**Function wait\_die (*tranNum,dataitem*):**

get *dataitem.tranNum* of holder Transaction from lock table as *hold\_tranNum*

get *hold\_tranNum.timestamp* of holder Transaction from transaction table as *hold\_timestemp*

get *tranNum.timestamp* of request Transaction from transaction table as *request\_timestemp*

if *request\_timestemp* < *hold\_timestemp*:

request Transaction will wait

update *tranNum.status* = ’blocked’ in transaction table

append *tranNum* in *dataitem.waitlist* in lock table

else:

request Transaction will abort

abort(*tranNum*)

**end wait\_die**

# adding item to waitlist method

**function add\_to\_waitlist** (*opName, TranNum, dataItem*)

join string with *OpName,TranNum,dataItem* as *operation*

append *operation* to *tranNum.opwaitlist*

**end add\_to\_waitlist**

# abort method

**function abort(*tranNum*):**

update *tranNum.status* = ‘aborted’ in transaction table

Release all locks by transaction by updating the entry in lock table and transaction table

Release\_lock(*tranNum*)

**end abort**

#function for end transaction

**function end\_transaction(*tranNum*):**

update *tranNum.status* =”commited” in transaction table

Release all locks by transaction by updating the entry in lock table and transaction table

Release\_lock(*tranNum*)

**end end\_transaction**

#function for the release lock

**function release\_lock(*tranNum*)**

for each *lock* held by *tranNum*

if *tranNumHolding* has other transaction in the list:

remove *tranNum* from that *tranNumHolding* list

Else:

Get first record from *tranNumWaiting* as *active\_transaction\_id*

modify *active\_transaction\_id* into *tranNumHolding* list

update *active\_transaction\_id*.*status* = ’active’ in transaction table

get *active\_transaction\_id.operation*s*InWaiting* as *operationlist*

for each *record* in *operationlist* as record

parse(*record*)

**end release\_lock**

#operation parsing function

**function parse(*operation*)**

Split operationname, number,item in *opName* and *tranNum* and *dataItem* from *operation*

If *opName* =’b’(begin)

*Timestemp* =+ 1

Begin\_transection(*timestemp,tranNum*)

Else:

If *tranNum* is already in transaction table

If *tranNum.status* =’active’

If *opName* = ‘r’: (read )

read\_item(*tranNum,dataItem*)

If *opName* = ‘w’: (write)

write\_item(*tranNum,dataItem*)

if *opName* is ‘e’: (end operation)

end\_transaction(*tranNum*)

Else If *tranNum.status* = ’blocked’:

Add\_to\_waitlist(*opName, TranNum,dataItem*)

Else if *tranNum.status* = ’aborted’:

Ignore the operation

**end parse**

**Data Structure:**

Data structure of this program has two tables, one is to maintain transactions and other is to maintain locks. Schema of both the table are as follow

**TABLE: Transaction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| tranNum | Timestamp | State | ListofItemsLocked | OperationsInWaiting |

**TABLE: Lock**

|  |  |  |  |
| --- | --- | --- | --- |
| dataItem | State | tranNumHolding | tranNumWaiting |