DB2 PROJECT 1 DESIGN DOCUMENT

CSE 5331

18 July 2017

**2 Phase Locking Protocol**

**Goal**: To implement a program that simulates the behavior of the two-phase locking (2PL) protocol for concurrency control. The implementation is achieved using **rigorous 2PL**, with the **wait-die**method for dealing with deadlock.

**Programming Language:** Java

**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| *S.No* | *Methods* | *Page No* |
| 1. | Table of Contents | 2 |
| 2. | Description | 3 |
| 3. | Function 1 – main () | 3 |
| 4. | Function 2 – beginTransaction () | 3 |
| 5. | Function 3 – readDataItem () | 4 |
| 6. | Function 4 – writeDataItem () | 5 |
| 7. | Function 5 – endTransaction () | 6 |
| 8. | Function 6 - abortTransaction() | 6 |
| 9. | Function 7 – waitDie() | 6 |

**Transaction Table**

***Data Structures Used for each Transaction:***

*HashMap<String,String>*

transaction Id

timestamp

transactionState

*HashMap<String, ArrayList<String>>*

ItemsLocked

**Lock Table**

***Data Structures Used for each Transaction:***

*HashMap<String,String>*

itemName

lockState

*HashMap<String, ArrayList<String>>*

tansactionIds(requesting)

tansactionIds(Locking)

**Pseudo Code:**

**Function 1 – main ():**

* Read input file line by line and store in ***StringBuffer***
* Read the operations one by one separated by delimiter (;) in the StringBuffer and store it in as string value and pass the string as parameter to the following functions.

If operation begins with b

Execute: beginTransaction(operation)

If operation begins with r

Execute: readDataItem (operation)

If operation begins with w

Execute: writeDataItem(operation)

If operation begins with e

Execute: endTransaction(operation)

**Function 2 – beginTransaction(**operation**) :**

* Get Transaction number from the operation
* Set Transaction State to ACTIVE and Counter to ‘0’.
* Store the transaction number (Eg: T1), State and Timestamp (incrementing counter) in *Transaction Table*

**Function 3 – readDataItem(**operation**) :**

* Get Data Item, Transaction Number from the operation
* Check status from *Transaction Table*

if(transactionState == “ACTIVE”){

#Check dataItem in *TransactionTable* for LockedItems

if(dataItem in LockedItems){

readData

}

else{

if(! dataItem in *LockTable*){

Set dataItem to ReadLock() in *LockTable*

Add dataItem to LockedItems in *Transaction Table*

readData

}

else{

if(lock\_state == “READ”){

Add TransactionId to TansactionIds in *LockTable*

Add dataItem to LockedItems in *Transaction Table*

readData

}

else{

set transactionState = “BLOCKED”

execute waitdie(reqTransaction, lockingTransaction)

}

}

}

}

**Function 4 – writeDataItem(**operation**)**

* Get Data Item, Transaction Number from the operation
* Check status from *Transaction Table*

if(transactionState == “ACTIVE”){

#Check dataItem in *TransactionTable* for LockedItems

if(dataItem in LockedItems){

if(lock\_state == READ){

if(tansactionIds.length == 1){

#update lock in locktable

lockState = WRITE

}

}

else

writeData;

}

else{

if(dataItem in *LockTable*)

set transactionState = “BLOCKED”

execute waitdie(reqTransaction, lockingTransaction);

else{

Set dataItem to writeLock() in *LockTable*

Add dataItem to LockedItems in *TransactionTable*

writeData

}

}

}

**Function 5 – endTransaction(**operation**):**

* Get Transaction number from the operation
* Check if the transaction State is ACTIVE
* Set Transaction State to COMMIT.
* Release all locked items and Update Transaction Table and LockTable

**Fuction 6 – abortTransaction(**transaction**):**

* Set Transaction State as “ABORTED”
* Release dataItems and update *TransactionTable* and *LockTable*.

**Fuction 7 – waitDie(t1,t2)**

* Check Transaction Table for TimeStamps of the transactions

if(timestamp(t1)>timestamp(t2)){

abortTransaction(t1);

}

else{

#add transaction id to waiting list in *LockTable*;

}