NEW YORK UNIVERSITY

COURANT INSTITUTE OF MATHEMATICAL SCIENCES

Subject: Advanced Database Systems

Project Title: Replicated Concurrency Control and Recovery

Authors: Aditya Poduval & Adheip Nagarajan

1 CONTENTS

2 SCHEME

2.1 THIS PROJECT DEALS WITH THE IMPLEMENTATION OF A DISTRIBUTED DATABASE SYSTEM INCLUSIVE OF DATABASE MANAGEMENT AND TUNING TECHNIQUES LIKE

a. Multiversion Concurency Control
b. Deadlock Detection
c. Replication
d. Failure Recovery

2.2 DATA TO BE USED FOR CREATING THE REPLICATED DATABASE SYSTEM

a. Number of Distinct Variables – 20 (X1, X2, X3….X20)
b. Number of Sites – 10 (1, 2, 3….10)

2.3 CONDITIONS OF DATA DISTRIBUTION

a. Odd indexed variables (X3, X5, X7….X19) are at site number 1 + index number mod 10
b. Even indexed variables are at all sites
c. Value (Data) of each variable Xi is initialized to 10i

2.4 ALGORITHMS TO USE

a. Available Copies for Replication using Two Phase Locking
b. Deadlock detection using Cycle Detection and Abort Youngest Transaction

3 DESCRIPTION

The Replicated Concurrency Control and Recovery approach for a distributed Database system was implemented in C as part of this project. Most of the functionalities as described in the problem statement and as a part of the explanation we attempted well to be sufficed by the execution of this project. Coding standards have been maintained across the implementation of this project and functionalities have been broken down into different files and functions to improves code execution and readability. The header files created as part of this project were

1. operations.h
   operations header holds all structures and definitions related to the Operations that would be performed on the distributed database system and the corresponding information affiliated with these Operations

2. site
   site header holds structure definitions associated to the sites including the version table and lock table information

3. transaction_manager.h
   This header fie stores structures associated with transactions that would be requesting operations on variables on each of the sites.

The code was routed via the main C file to other C files like the site_data.c and transaction_manager.c to obtain correct flow and distribution in the execution of each of the requested fucntionalites.

4 FUNCTIONS

The below table highlights the major and minor functions in detail existing in each of the files

4.1 MAIN.C

Function Name	Label	Information
Main	Input	Filename
	Description	This function accepts the input file, parses the file to store the operations and calls the transaction manager to perform the operation
	Output	0
cehckFileExists	Input	Filename
	Description	This function checks if the given filename exists or not and returns errors if the input file is empty
	Output	-1 for error 0 for no errors

4.2 SITE_DATA.C

Function Name	Label	Information
iniSiteData	Input	NA
	Description	Initialize all variables and the corresponding values and ids for all the sites
	Output	NA
perfOpn	Input	Transaction operation, site number
	Description	This function performs the operation in the transaction queue for each transaction and checks if the transaction requires locks
	Output	NA
release_lock	Input	Site number, transaction id
	Description	Function is called whenever a transaction wants to commit or abort or it fails. It performs operations from the active list. It scans the blocked list and set the operations to NULL
	Output	NA
updateVersionTable	Input	Site number, variable number, operation
	Description	This function maintains the version table at each site for each variable
	Output	NA
readVarValues	Input	Site number, Variable number and transaction timestamp

	Description	It retrieves the read value from the requestd site based upon the transaction timestamp
	Output	Read value
failSite	Input	Site number
	Description	Fails the site by releasing the locks on each variable from the failed site
	Output	NA
readOnlyVarValues	Input	Site number, variable number, transaction timestamp, operation timestamp, transaction id
	Description	This function is only called for a Read only transaction. Reads the value of the variable from the requested site. It also returns the most recent committed value of the variable
	Output	Read Value
isReadAvailable	Input	Site number, variable number, transaction id
	Description	This function checks if a variable at the input site can be read.
	Output	1 if variable can be read else 0
addOpnToActvList	Input	Site Number, Variable number, Transaction operation, status
	Description	Adds an operation for the transaction to the active list of operations to be executed at the site at the end of the queue
	Output	NA
addOpnToBlckdList	Input	Site number, variable number, transaction operations
	Description	Adds an operation for the transaction to the blocked list of operations to be executed at the site at the end of the queue
	Output	NA
isLockRequired	Input	Site number,
	Description	Checks if a transaction requires a lock on a particular variable is requested at the input site
	Output	1 if the lock is granted else 0

4.3 TRANSACTION_MANAGER.C

Function Name	**Label**	**Information**
parseInputFile	Input	Inputfile
	Description	This function parses the file received in input. It stores the individual operations like begin, end, read, write etc into the transaction queue. It also assigns timestamp to each operation
	Output	0 for successful parsed and 1 for errors

addOpn	Input	Operation structure, and operation timestamp
	Description	A new transaction is created for a begin transaction. It also prepares operation information for all other transactions and adds the operation to the transaction queue.
	Output	Returns -1 for errors
initTransMngr	Input	NA
	Description	Initializes all values of the transaction structure and also initializes site information for all sites.
	Output	NA
generateTrx	Input	Transaction ID, Transaction Type, Timestamp
	Description	This function creates a new transaction for each begin or beginRO operation.
	Output	-1 if limit has been exceeded or is a duplicate transaction else 1 if the new transaction was created successfully
prepOpnInfo	Input	Transaction ID, Operation type, variable number, site number, timestamp, transaction operation
	Description	This function sets Operation parameters for each operation and sets status for each operation as pending. Assigns transaction type to corresponding operation’s timestamp
	Output	0
addToQueue	Input	Transaction ID, transaction operation
	Description	This function adds current operation to the respective transaction’s queue. It assigns current operation to transaction’s first, current and last operation too.
	Output	NA
startTrxMngr	Input	NA
	Description	This function performs multiple operations. It fetches the operations from the transaction queue, calls perform operation function to perform each of the operations in the queue. In case of conflicts, blocks the operation and adds it to the waitlist. At every tick if a blocked operation is present it attempts to perform it if possible. It also commits a transaction in case an end operation is encountered
	Output	NA
abortTrx	Input	Transaction operation
	Description	This function is used to abort the current transaction if requested and clear all operations for that transaction. It also clears all the site information for that site.
	Output	NA

5 TEST CASES

5.1 CASE 1

```
begin(T1)

beginRO(T2)

W(T1,x1,101)

R(T2,x2)

W(T1,x2,102)

R(T2,x1)

end(T1)

end(T2)

dump()
```

Results:

Multi-version read protocol is applicable only to the read only transactions in this case as the Read operations would wait till the commit occurs the RO would still return the value of X1 that it had before the Write i.e 10. No aborts would happen as eventually the locks would be released.

5.2 CASE 2

```
begin(T1)

begin(T2)

R(T1,x3)

fail(2)

W(T2,x8,88)

R(T2,x3)

W(T1, x5,91)

end(T2)

recover(2)

end(T1)
```

Results:

T1 reads the value of X3 successfully. The write operation by T2 of X8 is successful and X8 is written as 88 at all sites apart from 2. T2 can still read the original value of X3. T1 then gets the lock and writes to X5 with the value 91.

5.3 CASE 3

```
begin(T1)
begin(T2)
R(T1,x1)
fail(2)
W(T2,x8,88)
R(T2,x3)
R(T1, x5)
end(T2)
recover(2)
end(T1)
```

Results:

Read operation of T1 reads original value of X1 from site 2. However, as site 2 fails T1 aborts hence only T2 commits.

5.4 CASE 4

```
begin(T1)
begin(T2)
fail(3)
fail(4)
R(T1,x1)
W(T2,x8,88)
end(T1)
recover(4)
recover(3)
R(T2,x3)
end(T2)
```

Results:

T1 reads original value of x1, T2 acquires write and writes 88 on x8 lock. Once the sites are recovered T2 reads x3 and then commits.