Design document

Author: Jiayun Yu, Kailin Luo

Date: Dec 8th, 2017

Final Project for Advanced Database System class

We wrote this skeleton of our project before writing code. And after we finish coding, we modified this design document again. I hope this document can describe our design ideas well. For more detail things, we also have more comments in our coding part.

compile command line: python project.py textfile

files:
1. module.py: all the obejcts
2. project.py: read input, and run command
3. status.py: all the global variables

Global variables:

DataManagerList	// dict of sites 10(10 data manager)
VariableSiteList	// 20 dict store site for each variables
TransactionList	// dict to store 10 transactions

Object1： Transaction Machine
--- get command, ask lock, update states
--- property: index //(initialize with 0, increase when new transaction come)
Graph // for deadlock detect
waitlist of command[]
wait commit transaction list[]
cycle transaction list[]
--- function:
1. begin(string): create transaction, add index, append to transactionList, add vertex to graph

2. write (string): Create command object, check whether variable legal (if not legal, change state to fail directly), get sits list from variable object, look for lock (through for loop, call checkLock(), (if wait, add edge), not success until all sites return success or fail) update status (success -- change value, wait --- do nothing, fail(caused by all site fail) -- do nothing)

3. read (string): Create command object, check if transaction is read-only:
1) true, read from the transaction currentVariableValue[]
2) false, check whether variable legal (if not legal, change state to fail directly), get sits list from variable object, look for lock (through for loop: break once we get a success, if wait add edge), getValue(), update status (success -- change value, wait --- do nothing, fail(caused by all site fail) -- do nothing)

3.5 readRO: read only transaction to call read command

4. end (string): Check T's command list,
If has fail: change T's state to abort, print information, remove vertex, edges, break
Else If has wait, change T's states to waitCommit, break
Else If all success: change states to go through the command list:
For readCommand: print variable and value, remove lock

For writeCommand: print result, update value to all []site, (
If (variable is replicate && some update success) success
If (variable is replicate && none update success) or If
(variable is not replicate && update failed) change
command's states to fail, change T's states to abort.
remove lock; go through every transaction, if transaction's states == commitWait,
call end(t) again. Else set transaction's states == success
remove vertex, edges

4.5 endRO: readonly transaction to call end command

5. Fail (string): change site's states

6. Recover(string): change site's states by call site's recover function.

7. beginRO(string): create transaction(readonly = true), add index, append to transactionList, get a list of current variable value.
8. deadLock() bool: check current graph cycle

Object 2 Data Manager:
--- property: states: true for on; false for off // describe the states of site, true for initialize
index
[]VariableInSite // store the list of variable
[][] current LockTable (each variable has a lock list // store the lock list for each variable
[][] wait lock table
[] transactionforDM // store the index of transaction that reach to this site

--- function:
update(Variable): update the variable value of this site.
return false if site failed; else change value and change variable's accessible = true
return true.

checkLock(Transaction t): return fail if site failed; return wait if lock conflict exist. add transaction to the end of list, add transaction's index to transactionforDM; return success if no conflict, add transaction to the list, add transaction's index to transactionforDM.
removeLock(variable, transactionid): remove lock, change next wait lock's state.

removeLock(): remove lock, change next wait lock's state

fail(): Fail the site

recover(): clear lock table, go through transactionForDM, abort if transaction is not success; clear transactionForDM, change to variable's accessible to false

getValue(variableName): return -1 when accessible == false, return value else.

Object 3 Transaction
--- property: index //timestamp
name //number
readOnly // true for readonly
CommandList[] // store all command
States // wait for not receive end(t), commitWait when received end(t),
//abort when fail, success if everything.

```
                    [20]currentVariableValue        // a copy of current variable value (none for not
                                                    // readOnly
                    command number
                    last commit version

Object 4 Command
--- property:       type            // read or write
                    index           //timestamp
                    TransactionId   // indicate the transaction
                    command num
                    Variable num    // read directly from command string
                    Value           // -1 for read initialize
                    State         // success, wait, fail
                    lock granted number

Object 5 VariableInSite
---property:        type            // replicated or not
                    index           // name of variable 0, 1,....
                    Value           // initialize with 10 i
                    Accessible      // initialize with true
                    version

Object 6 VariableInCommand
---property:
                    index           // name of variable 0, 1, …
                    Value           // initialize with -1 for read, value for write

Object 7 Lock
---property:        type            // read/ write
                    transaction number
                    commad number
```