Advanced Database Systems: Project Design Document (Distributed replicated concurrency control and recovery)

Due on 12/08/2021

Dennis Shasha

Xinyi Zhao

Section 1: Project Structure

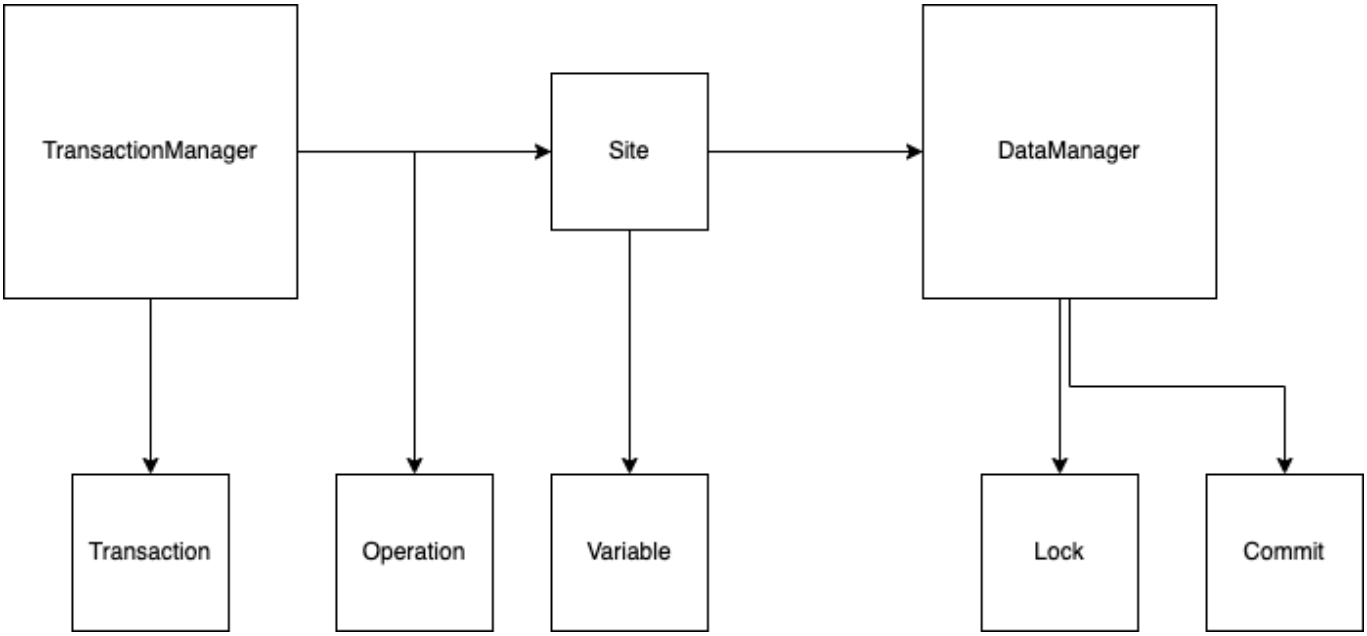
1.1 Language: Java

1.2 Package: FinalProject

1.2.1 Test input folder: tests

1.2.2 Test output folder: output

1.2.3 Entrance: Main.java



Section 2: Classes

2.1 TransactionManager

Attributes

- currentTick: int; track the current timestamp
- sites: Site; which contains a list of DataManager instances
- operationList: List; a list of Operation instances
- transactions: Map; key:transName, val:transaction
- holdLockTrans: Map; key:varName, val:[transName]
- waitForTrans: Map; key:varName, val:[transName]
- siteToFailTime: Map; key:siteID, val:[failedTimestamp]
- siteToRecoverTime: Map; key:siteID, val:[recoveryTimestamp]

Methods

- public
  - void checkOperationList(): check available operations before operate new operations
  - boolean operate(Operation operation): handle the current operation
- private
  - operate helper functions:
    - boolean write
    - boolean read
    - boolean recover
    - boolean fail
    - boolean end
    - boolean beginro
    - private void addOperation(Operation operation)
    - private List getAvailableSites(String varName)
  - check deadlock:
    - Map createWaitForGraph()

    - void abortDeadlock(String transName)
    - List getDeadlocks(Map graph, String transName)
    - void dfs(Map graph, String transName, String cur, List path, List deadlocks)
  - transaction helper functions:
    - void putHoldLock(String varName, String transName)
    - void putWaitFor(String varName, String transName)
    - void deleteHoldLock(String transName)
    - void deleteWaitFor(String transName)
    - void abort(String transName)

2.2 DataManager

Attributes

- siteID: int; site id
- isUp: boolean; whether this site is up
- lockTable: Map; key:varName, val:Lock instance
- variables: Map; key:varName, val:Variable instance
- commits: Map; key:varName, val:[Commit]

Methods

- read helper functions:
  - boolean canRead(String varName)
  - int read(String varName)
  - Integer readOnly(String varName, int latestRecovery, int transStartTime)
  - Integer readOnlyUnique(String varName, int transStartTime)
- write helper functions:
  - void write(String varName, int val)
- operation and transaction helper functions:
  - void fail()
  - void recover()
  - void dump()
  - void commit(String transName, int timestamp)
  - void reset(String transName)
- lock handler helper functions:
  - void initLockTable()
  - boolean getWriteLock(Operation operation, boolean hasWaitingCandidate)
  - boolean getReadLock(Operation operation)
  - void releaseWriteLock(String transName, String varName)
  - void releaseLocks(String transName)

2.3 Site

Attributes

- dm: List; list of DataManager instances

Methods

- private void initialize(int varCnt): initialize sites and variables

2.4 Transaction

Attributes

- timestamp: int; record the begin time for each transaction
- readOnly: boolean; record whether this transaction is RO
- aborted: boolean; record whether this transaction is aborted
- accessedSites: List; a list of siteID this transaction can access
- readOnlyList; List; a list of [siteID, latestRecoveryTime]

Methods

- public void abort(): abort this transaction

2.5 Operation

Attributes

- siteID: int;
- value: int;
- readUniqueVar: boolean; record whether this variable is unique through all sites
- operation: String; read, write, begin, end
- transName: String;
- varName: String;
- operateStr: String; record the original input operation string
- readOnlySites: Set; a set of siteID which are RO sites
- readOnlySiteRecovery: List; a list of [siteID, latestRecoveryTime]

Constructor

- public Operation(String input): handle each input line, transfer it to an Operation instance

2.6 Variable

Attributes

- isUnique: boolean; whether the variable is unique
- canRead: boolean; whether the variable is readable
- value: int; value of the variable
- valueToCommit: int; value of a write to commit

Methods

- void commit(): overwrite the current value with the new value.
- void reset(): reset valueToCommit to current value.

2.7 Lock

Attributes

- readLock: Set; a set of read locks of a variable
- writeLock: String; a write lock of a variable

Methods

- void releaseWriteLock()

2.8 Commit

Attributes

- timestamp: int; the timestamp for this commit
- value: int; the written value for this commit

Section 3: Test

Test the program using all testcases from:
https://cs.nyu.edu/courses/fall21/CSCI-GA.2434-001/projectsampletests.deadlockdetection
By comparing the output with the given results, this program passed all tests.