Software Design Document

Distributed replicated concurrency control & recovery

Anand Kumar (ak8288), Devesh Devendra (dd3053)
Supervised by: Prof. Dennis Shasha

December 4, 2022

NOTE: PLEASE IGNORE THE DATE. IT COMES WITH LATEX PACKAGE AND UNABLE TO TAKE EST TIME ZONE.

1 Introduction

1.1 Purpose

To create a sample software design prototype for Distributed Replicated Concurrency control & recovery.

1.2 Scope

The main scope of this project is to show how we can manage transactions in a distributed systems where multiple processes can initiate a transaction. We show how we try to control the concurrency so that the overall systems remains consistent at the same time ensuring the performance of the overall system. We also show how to overcome conflicts if any, among multiple transactions occurring simultaneously.

2 System Overview

2.1 Language

JAVA

2.2 Modules

1. Command Module [command.java]

Description : The proper Command is created and is executed accordingly.

- command [abstract class]
  - An abstract class prototype for all valid command types

  - public abstract void executeCommand(TransactionManager transactionManager, int currentTime);

- BeginCommand [extends command]
  - Executes the begin command
  - Stores the name and other Transaction Specific information in the Transaction Manager

- BeginROCommand [extends command]
  - Executes the BeginRO Command.
  - Stores the name and other Transaction Specific information in the Transaction Manager

- ReadCommand [extends command]
  - Executes the Read Command :
  - The Transaction can be Read Only or Read-Write Transaction.

- WriteCommand [extends Command]
  - Executes the Write Command for a given Transaction.

- DumpCommand [extends Command]
  - Executes the dump() command and prints all the data from all the sites.

- EndCommand [extends Command]
  - Ends a Transaction. Shows if a transaction is committed or Aborted

- FailCommand [extends Command]
  - Simulates the failure of a Site.

- RecoverCommand [extends Command]
  - Simulates the recovery of a Site

2. Command Manager [CommandManager.java]

Description : Creates a Command and its parameters in accordance with the input provided by the user.

- public Command getCommand(String inputString);
  - Returns a Command instance depending upon type, e.g; BeginCommand, ReadCommand, FailCommand etc.

3. Data Manager [DataManager.java]

Description : This module manages the data at all the sites.

- private boolean alreadyHasWriteLock(listOfLocks, transaction);

- public boolean acquireReadLock(Transaction transaction, String variableName);
- public boolean checkWriteLock(Transaction transaction, String variableName);
- public boolean acquireWriteLock(Transaction transaction, String variableName, int variableValue);
- public boolean waitListForWriteLock(Transaction transaction, String variableName, int variableValue)
- ublic boolean removeLock(Transaction transaction, String variableName);
- public boolean commitValue(String variableName, int variableValue, int commitTime);
- public void acquireWaitingLock(Lock lock);

4. Lock Module [Lock.java]

Description : Lock represents a lock by some transaction on some variable.

- Attributes
  - public int lockID;
  - public LockType lockType;
  - public LockStatus lockStatus;
  - private static int lockCount;
  - public Transaction transaction;
  - public String variableName;
  - public int variableValue;
  - public DataManager dataManager;

5. Transaction Module [Transaction.java]

Description : This module manages each transaction.

- Attributes
  - public int creationTime;
  - public String transactionName;
  - private static int transactionCount;
  - public TransactionStatus transactionStatus;
  - public TransactionType transactionType;
  - public HashMap<String, Integer> variableValues;
  - public HashMap<Integer, Integer> timeHoldUp;
  - public ArrayList<Lock> lockList;

6. Transaction Manager [TransactionManager.java]

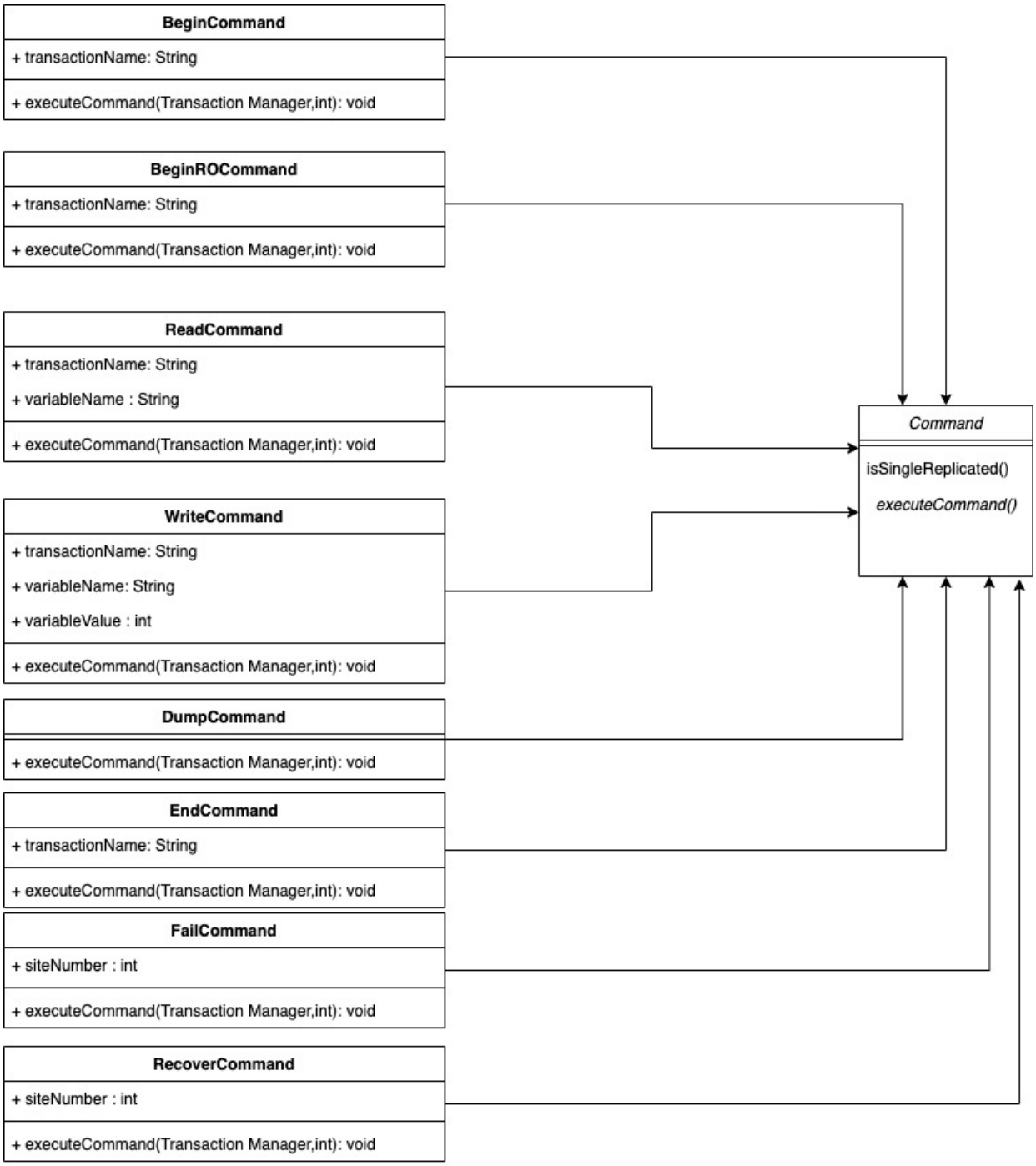
Description : Transaction Manager manages each transaction. It manages which transactions to put in WAITING state, which to EXECUTE etc.

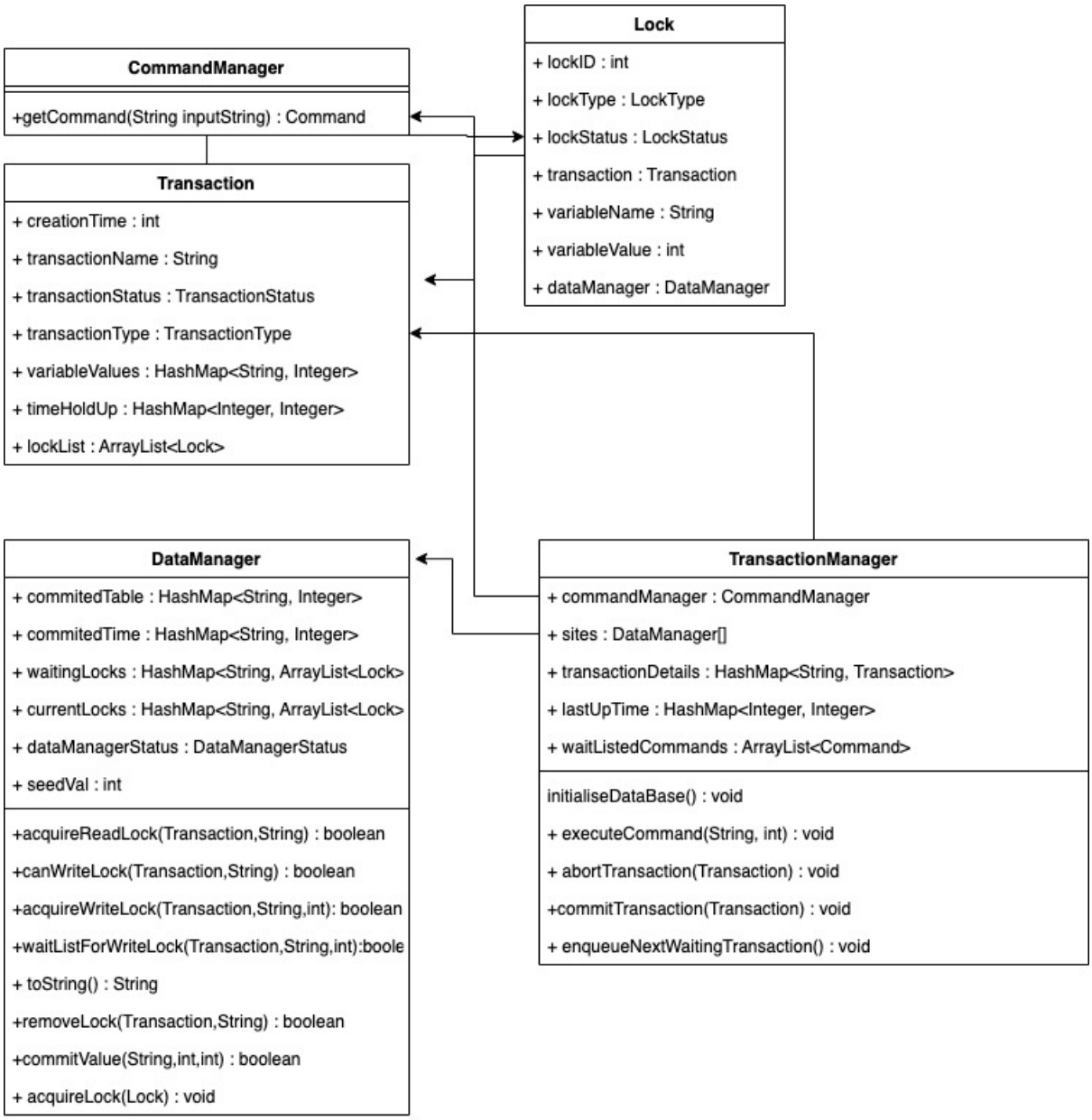
- Attributes
  - private static int currentTime;
  - private CommandManager commandManager;
  - public DataManager[] sites;
  - public HashMap<String, Transaction> transactionDetails;
  - public HashMap<Integer, Integer> lastUpTime;
  - public ArrayList<Command> waitListedCommands;
- Functions
  - public void initialiseDataBase() : Initialises the data sites
  - private boolean hasActiveTransaction() : checks if there are any waiting transactions
  - private boolean hasDeadLock();
  - private Transaction getYoungestTransaction();
  - public void executeCommand(String inputCommand, int currentTime);
  - public void abortTransaction(Transaction transaction);
  - public void commitTransaction(Transaction transaction);
  - private ArrayList<Transaction> getReadyTransactions();
  - public void enqueueNextWaitingTransaction();

7. Logger Module [Logger.java]

Description : Provides the Flexibility to either Log the details on the Standard Output[Console] or a File[as per the provided Command Line Argument].

- Attributes
  - public boolean isFileMode;
  - private PrintStream out;
- Functions
  - public void log(String str) : Log into either terminal or output file as specified by the user


BeginCommand
+ transactionName: String
+ executeCommand(Transaction Manager,int): void
BeginROCommand
+ transactionName: String
+ executeCommand(Transaction Manager,int): void
ReadCommand
+ transactionName: String
+ variableName : String
+ executeCommand(Transaction Manager,int): void
WriteCommand
+ transactionName: String
+ variableName: String
+ variableValue : int
+ executeCommand(Transaction Manager,int): void
DumpCommand
+ executeCommand(Transaction Manager,int): void
EndCommand
+ transactionName: String
+ executeCommand(Transaction Manager,int): void
FailCommand
+ siteNumber : int
+ executeCommand(Transaction Manager,int): void
RecoverCommand
+ siteNumber : int
+ executeCommand(Transaction Manager,int): void
Command
isSingleReplicated()
executeCommand()


CommandManager
+getCommand(String inputString) : Command
Lock
+ lockID : int
+ lockType : LockType
+ lockStatus : LockStatus
+ transaction : Transaction
+ variableName : String
+ variableValue : int
+ dataManager : DataManager
Transaction
+ creationTime : int
+ transactionName : String
+ transactionStatus : TransactionStatus
+ transactionType : TransactionType
+ variableValues : HashMap<String, Integer>
+ timeHoldUp : HashMap<Integer, Integer>
+ lockList : ArrayList<Lock>
DataManager
+ commitedTable : HashMap<String, Integer>
+ commitedTime : HashMap<String, Integer>
+ waitingLocks : HashMap<String, ArrayList<Lock>
+ currentLocks : HashMap<String, ArrayList<Lock>
+ dataManagerStatus : DataManagerStatus
+ seedVal : int
+acquireReadLock(Transaction,String) : boolean
+canWriteLock(Transaction,String) : boolean
+acquireWriteLock(Transaction,String,int): boolean
+waitListForWriteLock(Transaction,String,int):boole
+ toString() : String
+removeLock(Transaction,String) : boolean
+commitValue(String,int,int) : boolean
+ acquireLock(Lock) : void
TransactionManager
+ commandManager : CommandManager
+ sites : DataManager[]
+ transactionDetails : HashMap<String, Transaction>
+ lastUpTime : HashMap<Integer, Integer>
+ waitListedCommands : ArrayList<Command>
initialiseDataBase() : void
+ executeCommand(String, int) : void
+ abortTransaction(Transaction) : void
+commitTransaction(Transaction) : void
+ enqueueNextWaitingTransaction() : void