Project Design

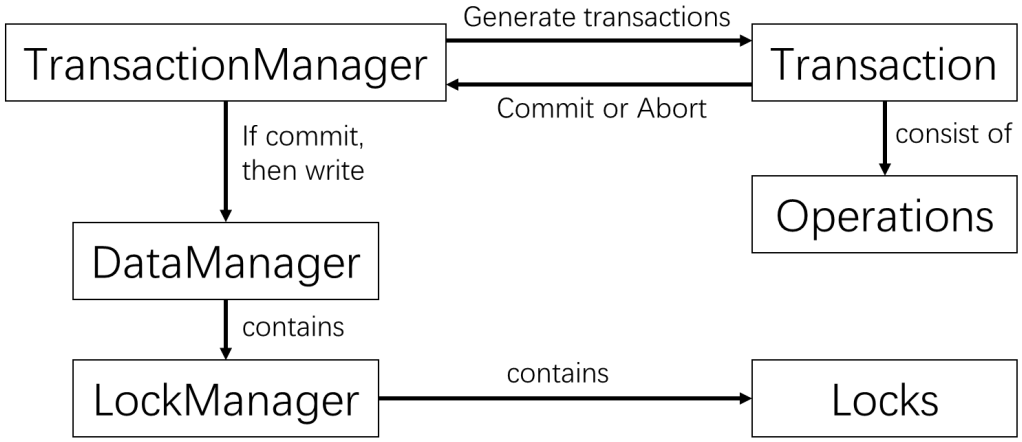
Name: Xiao Ma, Junru He
NetID: xm2074, jh7948

1. Introduction

This project is to implement a distributed database, complete with multi-version concurrency control, deadlock detection, replication, and failure recovery. The programming language of this project is Java. The program will read some input instructions from some files and output the results.

2. Design

2.1 Diagram



2. 2 Data Objects

Transaction: Each transaction can read and write on the data and consists of several Operations. **(assignee: Junru He)**

```
public class Transaction {
        private int id;
        private int timestamp;   // record the transaction begin time
        private TransactionType type; // Enum, can be READ_ONLY or READ_WRITE
        private boolean Blocked;
        private boolean Aborted;
}
```

Operation: Represent read and write operation. **(assignee: Junru He)**

```
public class Operation {
        private int timestamp;
        private int transactionId;
        private String variable;       // variable needs to be read or written
        private OperationType type; // Enum, can be READ or WRITE
        private int value;
}
```

Lock: Read or write lock in a specific transaction. **(assignee: Xiao Ma)**

```
public class Lock {
        private int transactionId;          // corresponding transaction
        private String variable;            // R(x1)
        private LockType type;               // Read_lock or Write lock
}
```

2. 3 Managers

Transaction Manager: Translate read and write requests on variables and write the new value on sites using a specific algorithm. **(assignee: Xiao Ma)**

```
public class TransactionManager {
        // key: transaction ID, value: corresponding transaction object
        private Map<Integer, Transaction> transactionTable;
        // key: site ID, value: corresponding transaction object
        private Map<Integer, DataManager> sites;
        // key: transactionId, value: Set<transactionId>
        private Map<Integer, Set<Integer>> waitsForGraph;

        public TransactionManager();
        public begin (int transactionId, int timestamp);            // start a transaction
        public beginRO (int transactionId, int timestamp);          // start a read-only transaction
        public end (int transactionId, int timestamp);              // end a transaction
        public readOnly (int transactionId, int timestamp);         // read-only operation
        public int read(int transactionId, int timestamp);          // read operation
        public write (int transactionId, int timestamp);            // write operation
        public boolean detectDeadLock ();                           // detect if there is a deadlock
        public boolean dump ();
        public boolean fail (int siteID);
        public boolean recover (int siteID);        // recover a site
}
```

Data Manager: Sites. Each site contains a copy of the variables. The data on each site can be modified by committed transactions. It has a lock to manage operations. When a site fails, it stores the last committed change of the data. **(assignee: Junru He)**

```
public class DataManager {
        private int siteId;
        // Enum, active, failed, recovered(can write, but cannot read)
        private StatusType type;
        // all available variables in a site, key: variable name, eg. x1, value: current value
        private Map<String, Integer> variables;
        private Map<String, LockManager> lockTable; // <variableName, lockManager>
        // key: variable name, value: a list of lists contain: [the value committed, timestamp]
        private Map<String, List<List<Integer>>>  commitHistory;

        public int getId();
        public boolean hasVariable(string variable); // if the site has this variable
        // try to get read lock on a variable
        public boolean canRead(TransactionType transactionType, Operation operation);
        public void read(TransactionType transactionType, int timestamp, Operation
operation);
        // try to get write lock on a variable
        public boolean canWrite(TransactionType transactionType, Operation operation);
        public void write(TransactionType transactionType, Operation operation);
        // when a transaction is aborted, do some operations on current site
        public void abort(int transactionId);
        public void commit(int timestamp, int transactionId);
        // print site status, committed value of all variables
        public void dump();
        public void fail();
        public void recover();
}
```

Lock Manager: Each instance is a lock manager of a site. It contains some information and a list of locks of that site. **(assignee: Xiao Ma)**

```
public class LockManager {
        private int siteID;
        private String variableName;
        private List<Lock> locks;          // the locks in this site

        public String getVariableName ();
        public Lock lock (OperationType, opType, int transactionId, String variableName);
        public void unlock (int transaction Id);
        public void lockAll ();
        public boolean canLock (OperationType, opType, int transactionId);
}
```