Advanced Database Systems

Fall 2019

Design Document

Sanjan Prakash Kumar (spk363)
Sudharshann D. (sd3770)

Objective :

To simulate a distributed database with MVCC, deadlock detection, replication and recovery from failure. Each request for an access to a variable will be processed using the available copies algorithm. The input will be a series of commands from a text file.

Modules implemented :

- ## Main (done by Sanjan)

1. Takes the input file with all the commands and parses it.
2. It calls the respective functions in the TransactionManager class specified in each line of the input such as :
   a. begin
   b. beginRO
   c. write
   d. read
   e. end
   f. dump
   g. fail
   h. recover

- ## Transaction Manager (done by Sanjan)

1. Uses the available copies algorithm to run the read and write requests. It also tries all the sites with the variable if the previous site has failed.
2. Initializing sites and variables.
3. It adds transactions to the waitlist if they cannot immediately get a lock on the variables required to complete the operation.
4. After any transaction has either failed or aborted, all the other transactions in the waitlist will be tried again.
5. If the transaction successfully ends, it will commit all the values it has modified.
6. Detects deadlocks
   a. Aborts the youngest transaction that is causing the deadlock.

- ## Site (done by Sudharshann)

1. Ten sites, each with its own lock table and set of variables.
2. Initializes the lock table

3. Initialize all the variables
4. The lock tables for the site is erased if the transaction fails
5. dump() function is implemented here.
6. Boolean to set the status of the site
7. Since it has access to the lock manager, it releases all locks upon commit and abort.

- **Variable (done by Sudharshann)**

1. Total 20 distinct variables.
2. Even-indexed variables are at all sites.
3. Odd-indexed variables are only at one site.
4. A site may have more than one different odd-indexed variable according to the formula : (1 + variable-index-number) mod 10.
5. Each variable xi is initialized to the value 10i. For eg : x1 is initialized to 10.

- **LockManager (done by Sudharshann)**

1. Initialise the read and the write lock tables for the particular site.
2. Facilitates the entire acquiring and releasing of locks for appropriate commands.
3. Raises custom exceptions when a conflicting transaction makes a request.

- **Transaction (done by Sanjan)**

1. Defines attributes of each transaction such as timestamp of its start, ID, type and status.
2. Implement functions that assign the right values to the above variables.

Output :

The output for each statement will be visible on the terminal.

How To Run :

1. Start all the ten sites with : `./Start.sh` (these will get created with ports 9090-9099)
2. Start the transaction manager (it will be created with port 7777) :
   `python2 TransactionManager.py`
3. Run the simulator and provide the path to the input file :

```
python2 Simulator.py ./Test1.txt
```

4. Stop all the sites and the transaction manager before starting a new test case :

   ```
   ./Stop.sh
   ```

All these instructions are included in the `run-simulation.sh` file.