Assignment 4: Database Server

Jon A. Solworth

Due: April 18th

1 Introduction

You are to implement a database server (`dbserver`). The database server will implement ACID properties. This is a substantially larger program then the ones you've done so far, so you'll need to break it down into much smaller pieces and understand them before you go on.

2 Client/Server interface

The client/server RPC interface includes the RPCs shown in Table 1. The client may have only one transaction at a time. The server must enforce all transition rules, including that other than `TransactionBegin`, all calls must be within a transaction.

Call	**Description**	**in a transaction**
`TransactionBegin`	start a new transaction.	no
`TransactionEnd`	end a transaction.	yes
`Read`	read a variable	yes
`Write`	write a variable	yes

Table 1: Client/Server transaction RPC Interface

Transactions might `abort` at any time, but can commit only at `TransactionEnd`.

3 Locking

Each database variable must be locked in the `dbserver` before it is accessed, the access is then said to be *covered* by a lock.

All `Read` request must be covered by an `SLock` (i.e., a shared lock); all `Write` request must be covered by an `Xlock` (i.e., an exclusive lock). As the name indicates, a variable may have multiple `SLock`s on it (from different clients), but if it has an `Xlock` it must be the only lock on it.

Each transaction should have at most one lock on a variable.

If a lock cannot be obtained, the request is queued and the reply to the `Read` or `Write` is delayed until the lock is obtained (or the transaction is aborted).

Hint: You'll need a datastructure which tracks locking (and other information on a per (database) variable basis.

4 Commit/Abort

Commit happens at `TransactionEnd`; abort can happen at any time. Commit occurs when all the log entries are written for the transaction after which the client is informed.

4.1 Commit

At a TransactionEnd the `dbserver` will make the transaction durable, by writing a log record for each `Write` as well as some information indicating that all writes for the transaction have been written.

The log records for a transaction must be written to the filesystem before any of the writes to the store.

Hint You want a directory for the log and one for the store. The name of store files should be the variable name. The log files should not be of type `string`, but of some more structured type.

4.2 Abort

An abort can occur due to a transaction time out or other error. The time out is tracked by the server, and addresses issues of deadlock as well as clients which are malfunctioning.

Transactions which abort for a non-permanent reason should be retried by the client up to `Retry` times.

If a server aborts a transaction because it times out (i.e., is not complete by `InitialTimeOut`), `dbserver` will give the next transaction from that client will be given double the amount of time. This doubling will continue for a total of `Retry`-1 times or until a transaction commits.

5 Recovery

When the `dbserver` starts up, it should recover the committed transactions in the log and write them to the store before accepting new transactions.

6 Conditions to test

Your server should test the following conditions:

- Multiple clients
- Transaction which reads a variable and then writes a variable
- Transaction which does multiple locks on the same variable
- Deadlock transaction
- Server failure and recovery
- Client failure
- Malformed client transactions