Project Report 2

weilil

Step 3:

To implement lock conversion, we use Update Lock mode. That is,

The following are examples of how a sequence of operations converts in the implementation

(ul1 means lock update lock on transaction 1, uul2 means unlock update lock on transaction 2, operations in () are consider one atomic operation)

**rl1[x], wl1[x] 🡪(the actual implementation) rl1[x],(url1[x],ul1[x]), (uul1[x],wl1[x])**

**rl1[x], rl2[x], wl1[x], url2[x]🡪 rl1[x], rl2[x], (url1[x],ul1[x]) *(transaction 1 blocked here)*, url2[x], (uul1[x],wl1[x])**

**rl1[x], rl2[x], wl3[x], wl1[x], url2[x], uwl1[x]🡪 rl1[x], rl2[x], *(transaction 3 blocked here)*. (url1[x],ul1[x]) *(transaction 1 blocked here)*, url2[x], (uul1[x],wl1[x]), uwl1[x], wl3[x]**

Step 4:

The lock is on Resource and Customer granularity. Therefore, there can be at most one transaction writing to a resource or customer.

Extra points:

* We use Update lock to implement read lock to write lock conversion. The Update mode tells the LM that this transaction is requesting to convert the read lock it holds to write lock
  + Allow to wait for other transactions to release their read locks on the resource before the write lock can be granted.
  + Avoid other transactions to put write lock on after transaction releases the read lock and before the write lock is in.
* Implement multiple transaction support. Commit may overwrite the resources other transaction written. Need to implement dirty bits to make sure commit only updates the changed resources.