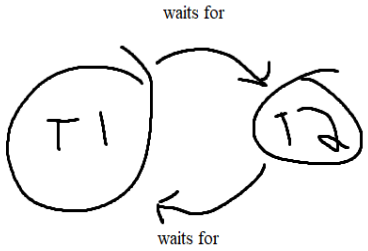
Ryan Griffin Lab 12: Locking

1.) Rigorous 2PL with timestamps used for deadlock prevention (using Wait-Die policy).
    - S1: T1 gets a shared-lock on X; When T2 requests an exclusive lock on X, since T2 has a lower priority, it will abort; T3 now gets exclusive-lock on Y; When T1 also asks for an exclusive-lock on Y which is still held by T3, since T1 has higher priority, T1 will be blocked waiting; T3 finishes write, commits and releases the locks; T1 wakes up, gets the lock, proceeds and finishes. T2 now can be restarted.
    - S2: The sequence and consequence are the same with Sequence S1, however T2 is able to advance more before it aborts.
2.) Rigorous 2PL with deadlock detection. (Show the waits-for graph in case of deadlock.)
    - Transactions allowed to wait; not aborted until a deadlock is detected
    - S1: T1 gets a shared-lock on X; T2 blocks waiting for an exclusive-lock on X; T3 gets an exclusive-lock on Y; T1 blocks waiting for an exclusive-lock on Y; T3 finishes, commits, and releases locks; T1 wakes up, gets an exclusive-lock on Y, finishes up and releases lock on X and Y; T2 now gets both an exclusive-lock on X and Y, and proceeds to finish. No deadlock.
    - S2: There is a deadlock. T1 waits for T2, while T2 waits for T1



3.) Timestamp concurrency control with buffering of reads and writes (to ensure recoverability) and the Thomas Write Rule.
    - S1: This sequence will be allowed.
    - S2: This sequence will be allowed.
4.) Validation (Optimistic) version control
    - S1: Since T1 gets the earliest timestamp, it will commit without problem; but when validating T2 against T1, none of the three conditions hold, so T2 will be aborted and restarted later; so is T3 (same as T2).
    - S2: Same as S1

5.) Multiversion timestamp concurrency control

- S1: T1 reads X, so RTS(X) = 1; T2 is able to write X, since TS(T2) ¿ RTS(X); and RTS(X) and WTS(X) are set to 2; T2 writes Y, RTS(Y) and WTS(Y) are set to 2; T3 is able to write Y as well, so RTS(Y) and WTS(Y) are set to 3; Now when T1 tries to write Y, since TS(T1) ¡ RTS(Y), T1 needs to be aborted and restarted later.
- S2: Similar to S1