Weekly exercises IN3020&4020 (2024)
Week 10: Concurrency Protocols

1 Review

1. Remind yourself what is a transaction, schedule, (conflict-)serialisability, locks (binary, read/write, with conversion), and two-phased locking protocol.
2. What do we mean by deadlocks and starvation of a transaction?
3. Briefly discuss different strategies for avoiding deadlocks in two-phase locking protocols: naive, deadlock prevention (wait-die, wound-wait, no-waiting, cautious-waiting), deadlock detection via wait-for graph, timeout-based
4. What is a timestamp? Briefly discuss the basic timestamp ordering protocol for concurrency control (without and with the Thomas write rule). How does strict timestamp ordering differ from basic timestamp ordering?
5. What are the main multiversion protocols for concurrency control and their main ideas?
6. How can you characterise the serial schedules (conflict-)equivalent to the schedules produced by multiversion timestamp-based and validation-based protocols?
7. How does the granularity of data items affect the performance of concurrency control? What factors affect selection of granularity size for data items? What is multiple granularity locking? Under what circumstances is it used? What are the main ideas of Multiple-granularity two-phase locking?

2 Locks and deadlocks, two-phase locking

1. Consider the following two schedules of transactions T_1, T_2 and T_3:
   - $S_1 : r_1(A), r_2(B), r_3(C), r_1(B), r_2(C), r_3(D), w_1(C), w_2(D), w_3(E)$

- $S_2 : r_1(A), r_2(B), r_3(C), r_1(B), r_2(C), r_3(A), w_1(A), w_2(B), w_3(C)$

(a) Insert read and write locks, and unlocks into these schedules as follows: place a read lock immediately in front of each read operation that is not followed by a write operation of the same element by the same transaction (i.e., without lock upgrades); place a write lock in front of every other read or write operation; place the necessary unlocks at the end of every transaction.

(b) Write the schedules of the resulting transactions that try to evaluate the operations in the same order as S_1 and S_2, and violate this order only if necessary (assuming that the transaction manager supports read/write locks).

2. (*) Explain how to modify the data structures for multiple-mode locks and the algorithms for `read`(X), `write`(X), and `unlock`(X) so that upgrading and downgrading of locks are possible. (Hint: The lock needs to check the transaction ids that hold the lock, if any.)

3. For each of the (intended) schedules below, assume that read locks are requested immediately before each read action, and write locks are requested immediately before every write action. Also, unlocks occur immediately after the final action that a trans- action executes. Explain what operations has to wait if a schedule is evaluated under two-phase locking protocol, and whether deadlock occurs. Also explain how the waits-for graph evolves during the execution of the actions. If there are deadlocks, pick a transaction to abort, and show how the schedule continues:

   (a) $r_1(A), r_3(B), r_2(C), w_1(B), w_3(C), w_2(D)$

   (b) $r_1(A)$,$r_3(B)$,$w_2(C)$,$r_2(D)$,$r_4(E)$,$w_2(B)$,$w_3(C)$,$w_4(A)$,$w_1(D)$,$w_4(A)$

4. For the schedule constructed in Question 3(b), explain what happens under the wait-die deadlock prevention protocol, assuming the order of timestamps is the same as the order of subscripts for the transactions—that is, T_1, T_2, T_3, T_4. Assume also that transactions that need to restart do so in the order that they were rolled back.

5. For the schedule constructed in Question 3(a), explain what happens under the wound-wait deadlock prevention protocol, assuming the order of timestamps is the same as the order of subscripts for the transactions—that is, T_1, T_2, T_3. Assume also that transactions that need to restart do so in the order that they were rolled back.

3 Timestamps

1. Consider the following schedules (which include the beginning operations):

   (a) $b_1, r_1(A), b_2, w_2(B), r_2(A), w_1(B)$

   (b) $b_1, b_2, r_1(A), r_2(B), w_2(A), w_1(B)$

   Show what happens as each executes under the timestamp-based concurrency control protocol.

2. Give an example of an application of Thomas write rule (and explain its purpose).

4 Multiversioning and multigranularity concurrency

1. Consider the following schedules:

   (a) $S_1 : b_1, r_1(A); b_2, r_2(A); r_1(B); w_2(A); w_1(A); r_2(B); w_2(B);$

   (b) $S_2 : b_1, r_1(A); b_2, r_2(B); w_2(B); r_1(B); w_1(A); w_1(B);$

   (c) $S_3 : b_1, r_1(A); b_2, r_2(A); b_3, r_3(A); w_1(B); r_3(B); w_2(A); w_3(A); w_2(B);$

   Describe what happens when these schedules evaluate under the multiversioning timestamp-based protocol.

2. Do the same as in Question 1, but using multiversion two-phase locking.

3. Give an example of a schedule that has to roll-back when evaluated under validation-based protocol.