Weekly exercises IN3020 and IN4020
Week 11

April 7, 2022

1 Review

1. What do we mean by starvation of a transaction?
2. Discuss different strategies for avoiding deadlocks in two-phase locking protocols
3. What is a wait-for graph?
4. What is a timestamp? Discuss the timestamp ordering protocol for concurrency control. How does strict timestamp ordering differ from basic timestamp ordering?
5. What are the main multiversion protocols for concurrency control? What are their advantages and disadvantages?
6. How can you characterise the serial schedules (conflict-)equivalent to the schedules produced by multiversion timestamp-based and validation 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?
8. Open-ended discussion questions based on Section 21.6 of the book: What are the specific properties of B+-trees that do not allow to efficiently use two-phase locking and other usual concurrency control protocols for B+-tree indexes? Can you think of approaches to concurrency control for B+-tree indexes?

2 Locks and deadlocks, two-phase locking

1. For each of the 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 shared and exclusive locks, and insert unlock operations. Place a shared lock immediately in front of each read action that is not followed by a write action of the same element by the same transaction (i.e., without lock upgrades). Place an exclusive lock in front of every other read or write action. Place the necessary unlocks at the end of every transaction.

(b) Tell what happens when each schedule is run by a scheduler that supports shared and exclusive locks.

(c) Insert shared and exclusive locks in a way that allows upgrading. Place a shared lock in front of every read, an exclusive lock in front of every write, and place the necessary unlocks at the ends of the transactions.

(d) Tell what happens when each schedule from previous item is run by a scheduler that supports shared locks, exclusive locks, and upgrading.

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

3. For each of the schedules below, assume that shared locks are requested immediately before each read action, and exclusive locks are requested immediately before every write action. Also, unlocks occur immediately after the final action that a trans- action executes. Tell what actions are denied, and whether deadlock occurs. Also tell 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 sequence of actions continues.

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

   (b) $r_1(A), r_3(B), w_2(C), r_2(D), r_4(E), w_2(B), w_3(C), w_4(A), w_1(D), w_4(A)$[1]

4. For each of the schedules below (the same as in the previous question), tell what happens under the wait-die deadlock avoidance system. Assume the order of deadlock-timestamps is the same as the order of subscripts for the transactions, that is, T_1 , T_2 , T_3 , and T_4. Also assume that transactions that need to restart do so in the order that they were rolled back.

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

[1] updated 7/4

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

5. For each of the schedules in the previous question, explain what happens under the wound-wait deadlock avoidance system.

3 Timestamps

1. Consider the following schedules, which include the beginning operations start events. Tell what happens as each executes under the timestamp-based concurrency protocol.

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

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

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 multi-versioning timestamp-based protocol.

2. Same as above, but using multiversion two-phase locking

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