Weekly exercises IN3020 and IN4020
Week 10

March 25, 2022

1 Review

1. What is (conflict-)serialisability and how can serialisability be used as a way to ensure concurrency control?
2. What does it mean for two schedules to be conflict-equivalent?
3. Discuss the four levels of isolation in SQL. What is snapshot isolation?
4. What is the two-phase locking protocol?
5. If you are using the two-phase locking protocol, can you be sure that a schedule is serialisable?
6. In what cases can we have a deadlock? What are the approaches to this problem?

2 Serialisability

1. Let T_1 be the transaction $r_1(X), w_1(X), r_1(Y), w_1(Y)$ and T_2 the transaction $r_2(X), w_2(X)$. List all possible schedules of T_1 and T_2 and decide which are serialisable.
2. Below are two transactions, described in terms of their effect on the database elements A and B, which we may assume are integers.

T_1
$r_1(A)$
$A := A + 1$
$w_1(A)$
$r_1(B)$
$B := B * 3$
$w_1(B)$

T_2
$r_2(B)$
$B := B * 2$
$w_2(B)$
$r_2(A)$
$A := A + 3$
$w_2(A)$

(a) Give examples of a serialisable schedule and a non-serialisable schedule of the these transactions

(b) It turns out that both serial schedules have the same effect on the database, that is, (T_1, T_2) and (T_2, T_1) are equivalent. Demonstrate this fact by showing the effect of the two transactions on an arbitrary initial database state.

3. Consider the following two schedules:

   S1 $w_3(A); r_1(A); w_1(B); r_2(B); w_2(C); r_3(C)$

   S2 $r_1(A); r_2(A); r_3(B); w_1(A); r_2(C); r_2(B); w_2(B); w_1(C)$

   Answer the following questions:

   (a) What is the precedence graph for the schedule?

   (b) Is the schedule serialisable? If so, what are the equivalent serial schedules?

   (c) Are there any serial schedules that must be equivalent (regardless of what the transactions do to the data), but are not conflict-equivalent?

3 Locks

1. Again consider the two schedules:

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

   (b) $r_1(A); r_2(A); r_3(B); w_1(A); r_2(C); r_2(B); w_2(B); w_1(C)$

   Assume that at transaction takes a lock on each database element immediately before it reads or writes the element, and that each transaction releases its locks immediately after the last time it accesses an element. Tell what the locking scheduler would do with each of these schedules; i.e., what request would get delayed, and when would they be allowed to resume? Then give an example of how the scheduler can look like.

2. Prove that two-phase locking always gives serialisable schedules (Hint: if a serialisability graph for a schedule has a cycle, at least one of the transactions in the schedule does not obey the two-phase locking protocol)