Explanation of answers to Reading Quiz 16

Q1: Answer is 1 because that schedule is conflict equivalent to doing T1 before T2, but not conflict equivalent to doing T2 before T1. The way we check for conflict equivalence is that we try to reorder the schedule using legal swap operations until we get to a serial schedule. Let’s take T2 (which contains 4 instructions) and try to push it ahead of the READ(A) instruction from T1. If we can do that, then the schedule would be conflict equivalent to T2 before T1. Unfortunately we can’t do that, because the WRITE(A) of T2 conflicts with READ(A) from T1. So instead, let’s try to push all of T2 to after all of T1. This, we can do, because none of T2’s 4 instructions conflict with WRITE(B), and the other of T1’s 2 instructions that we’re trying to push T2 past are not READs or WRITEs, and we only look at READs and WRITEs when checking for conflict serializability.

Q2: Answer is 1 because that schedule is equivalent to doing T1 before T2, but not equivalent to doing T2 before T1. Best way to answer these types of questions is to just plug in some values for the DB records, and then run these schedules and see what happens. In this case, there are two relevant DB records: A and B. From the fact that we are doing simple arithmetic on those records, we can assume that the records contain a single attribute that is a number (on an exam I would try be careful to tell you the schema rather than allowing you to make your own assumptions about the data). Assign A and B to some random number, and then run the schedules serially --- try T1 before T2, and also try T2 before T1. Then run the schedule as shown. If the final result of running this schedule is the same as one or both of the serial options you had run previously, then the answer is likely 1 (or 2). But to be sure, you should try other values for A and B and make sure that no matter what A and B start off as, the schedule shown is equivalent to one (or both) of the serial options. Otherwise, the answer is 0.

Q3: Answer is 1 again, because that schedule is conflict equivalent to doing T1 before T2. We can push all of T2 after the 4 instructions from T1 that are currently after T2 since those instructions from T1 are reading and writing B, and T2 reads and writes a totally different database record. We can’t push T2 to before T1 because READ(A) conflicts with WRITE(A) (and also WRITE(A) conflicts with WRITE(A)).

Q4: Answer is 2 because the final result of running T1 before T2 is the same as running T2 before T1 and also the same as the interleaved schedule shown. As mentioned above (for Q2), the best way to answer these types of questions is to just try out values for A and B and see what happens.

Q5: Answer is true as stated explicitly in textbook.

Q6: Answer is false as textbook says most common is **read committed** (not repeatable read).

Q7: Answer is false as the definition of read uncommitted is that it allows dirty reads.

Q8: Answer is true because the only difference between repeatable read and read committed is what happens when the same value is read multiple times. But a scan only reads each value once.

Q9: Answer is false because the example given by textbook to explain the difference between repeatable read and serializable would not go away as a result of doing a nested-loops join.

Q10: Running at the isolation level of read committed can result in **more** bugs being introduced for an application, not less!!!

Q11: See discussion of shared locks in 14.9.1

Q12: See section 14.9.3, especially the last paragraph.