Introduction to Database Systems

Problem Set 3 Solution

1. (a) This answer must show a workload that needs ACID properties, is write-intensive, and does not rely heavily on distributed transactions, i.e, has limited joins. Businesses, medicine, and government fall into this category.

   (b) This workload needs to be massive, and have limited consistency demands. Web apps and games make good candidates for this case.

2. (a) Precedence Graph: T1 –(3, 5)–> T2
   <–(4, 7)–
   The precedence graph has cycles, so it cannot be conflict serializable. It is also not view serializable. We know this because if it were ordered T1, T2, Step 4 would read the a different value. Likewise, a T2, T1 execution not be possible because Step 5 read the version of R1 post-T1's write.

   (b) Precedence graph: T1 –(1, 2) –> T2.
   This schedule is conflict serializable because we can swap (5, 6) and (4, 6), producing an order of T1, T2. It is also view serializable because all of T1's reads come first and it has no interleaving writes.

   (c) PG: T1 – (2, 3) –> T2
   <– (3, 4) –
   This is not conflict serializable because of the cycle shown in the precedence graph. It is view serializable with the schedule (T1 T2) because T1 has all of its reads before T2 and the last write of of W1 is also done by T2.

3. Two-phase locking will have the following states for T1:
   S3
   S2
   U2 - lock point, release 2
   ....
   - release 3 at end

   T2 will have:
   S2
   U2
   S1
   X4
   U1 - release all

4. Phantom problem:

   (a) Yes, it is vulnerable. The customers may be updated if they have a family size of two and live in Illinois.

   (b) No, it is not vulnerable because there is no overlap in the tuples each transaction accesses.

   (c) Yes, inserting the tuple will increase the count by one.

5. Optimistic concurrency control:

   (a) T_2 aborts because its read of A is potentially dirty.

(b) Serial validation prevents more than one transaction from running its write phase at the same time. Therefore, we permit overlapping write sets.

(c) T_2 commits because it does not overlap with T_1's write set.

6. Consistent hashing:

(a) It would be valid to write to any set of three in (H, A, B, C, D).

(b) This configuration would be able to write to the set (H, A, B, C).

(c) It will succeed upon writing to any pair in the set above, such as (H, C)

(d) In the absence of A, all pairs now span (H, B, C, D). Here, we might write to (B, D).