CS532 Homework #5 Solutions
Due at the beginning of class on December 6

1. (20%) Investigate (a) whether the default lock granularity in Oracle is at the table level or at the tuple level; (b) whether deferred database modification or immediate database modification is used in Oracle; (c) whether you can still access (select) a tuple from a different session when the tuple is being modified (it is modified but the change has not been committed). Report the steps and examples you used in your investigation using Oracle.

   Answer (examples and steps are not provided). (a) Tuple level because different tuples of the same table can be modified by different transactions at the same time. (b) Deferred database modification because changes will not be visible to others (including different sessions) before a commit is issued. (c) You can still select the tuple but you cannot see the change being made before the commitment.

2. (16%) Suppose when a crash occurred, the log in the stable storage has the following records in the given order (where T1, T2 and T3 represent different transactions):

   <T1 start>, <T1,A,35,40>, <T2 start>, <T2,B,40,65>, <T1,D,25,45>, <T3 start>, <T2 commit>, <T1,B,65,60>, <T3,D,45,55>,

   (a) If the deferred database modification recovery technique is used, what should be done to T1, T2 and T3 (the choices are "no action", "redo" and "undo")? Before the crash, what are the values of A, B, and D as can be seen by other database users with access privilege? After the recovery is completed, what are the values of A, B and D in the database?

   Answer:
   no action to T1 and T3; redo T2.
   before crash: A = 35, B = 65, D = 25
   after recovery: A = 35, B = 65, D = 25

   (b) If the immediate database modification recovery technique is used and the real database is updated as soon as possible (assume that for all log records in the stable storage, the corresponding changes have been made to the real database), what should be done to T1, T2 and T3? Before the crash, what are the values of A, B, and D as can be seen by other database users with access privilege? After the recovery is completed, what are the values of A, B and D in the database?

   Answer:
   undo T1 and T3; redo T2.
   before crash: A = 40, B = 60, D = 55
   after recovery: A = 35, B = 65, D = 25

3. (14%) Consider the following three transactions (time goes from left to right):

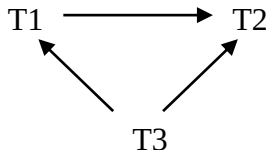
   T1: R1(X) R1(Y)W1(Y)
   T2: W2(X) R2(Z)W2(Z)
   T3: R3(Y)W3(Y) R3(X)W3(Z)

   Give a schedule that satisfies the strict two-phase locking (S2PL) protocol. In addition, enforce the rule that each transaction releases its locks as soon as possible. Is your schedule in a serial schedule? If the answer is no, give a serial schedule that is equivalent to your non-serial schedule.

   Answer: The schedule is:

   rl1(X) R1(X) wl3(Y) R3(Y) W3(Y) rl3(X) R3(X) wl3(Z) ul3(X) W3(Z) ul3(Y) ul3(Z) wl1(Y) ul1(X) wl2(X) W2(X) wl2(Z) R2(Z) W2(Z) ul2(X) ul2(Z) R1(Y) W1(Y) ul1(Y)

This is not a serial schedule but it is equivalent to the serial schedule (T3 T1 T2) based on the topological sort of the following precedence graph of the schedule:



4. (10%) Briefly describe two objectives of Write-Ahead-Logging supported in many database management systems.

Answer:

1) Store any changes to log records to perform undo and/or redo when the DBMS recovers from a crash to support atomicity and durability.
2) Do sequential writes of log records to the log file before actually writing changes to the database on persistent storage via potentially random writes.

5. (40%) Assume that there are 3 data items x, y and z in the database. Consider the following three transactions T1, T2 and T3 with their operations coming in the given order (from left to right):

T1: R1(x) W1(x) R1(y) W1(y)
T2: R2(x) W2(x) R2(z) W2(y)
T3: W3(z) R3(y)

Provide the schedules that can be generated by (a) 2PL; (b) resource ordering (with order: x y z); (c) Wait-Die Rule; and (d) Wound-Wait Rule. Note that (b), (c) and (d) should be used in combination with 2PL. For (c) and (d), you are not required to show the restart of aborted transactions (however, we assume that we do not kill a transaction that has not actually started; just let it wait until the next earliest possible time to start but it should keep its original starting/arriving time). Whenever possible, requests should be accommodated based on first-come-first-service and locks should be released as soon as possible. You may stop when a deadlock occurs.

Answer:

(a) wl1(x) R1(x) wl3(z) W3(z) W1(x) wl1(y) ul1(x) wl2(x) R2(x) W2(x) R1(y) W1(y) ul1(y) rl3(y) ul3(z) rl2(z) R2(z) R3(y) ul3(y) wl2(y) ul2(x) ul2(z) W2(y) ul2(y)

(b) wl1(x) R1(x) rl3(y) wl3(z) W3(z) ul3(z) W1(x) R3(y) ul3(y) wl1(y) ul1(x) wl2(x) R2(x) W2(x) R1(y) W1(y) ul1(y) wl2(y) rl2(z) ul2(x) R2(z) ul2(z) W2(y) ul2(y)

(c) wl1(x) R1(x) wl3(z) W3(z) (T2 should die, however, since it has not started, we will let it wait) W1(x) wl1(y) ul1(x) wl2(x) R2(x) W2(x) (T2 dies while waiting for T3 on z) ul2(x) R1(y) (T3-dies while waiting for T1 on y) ul3(z) W1(y) ul1(y)

(d) wl1(x) R1(x) wl3(z) W3(z) (T2-waits-for-T1-on-x) W1(x) wl1(y) ul1(x) wl2(x) R2(x) W2(x) (T2-waits-for-T3-on-z) R1(y) (T3-waits-for-T1-on-y) W1(y) ul1(y) rl3(y) ul3(z) rl2(z) R2(z) R3(y) ul3(y) wl2(y) ul2(x) ul2(z) W2(y) ul2(y)