Practice Problems

Quiz 4 Transactions

1. Define a transaction.
   Transaction: An action, or series of actions, carried out by a single user or application program, which reads or updates the contents of the database. A logical unit of work that transforms the database from one consistent state to another. Also the unit of concurrency and recovery control.
2. Describe the types of problems that occur in a multi-user environment when concurrent access to the database is allowed. List example of these problems.
   1) *Lost update problem: two active transactions update the same data item. The first write is lost because of the 2nd write.*
   2) *Uncommitted dependency problem: one active transaction updates a data item, another active transaction reads the data item BEFORE the first active transaction commits.*
   3) *The inconsistent analysis problem: an active transaction reads a subset of records from the database, it then performs the same action and generates a different result than the prior read because another transaction has updated some of the read values or added new records.*
3. Explain the concepts of serial, nonserial and serializable schedules.

Schedule *A sequence of the operations by a set of concurrent transactions that preserves the order of the operations in each of the individual transactions.*

Serial schedule *A schedule where the operations of each transaction are executed consecutively without any interleaved operations from other transactions*

Nonserial schedule *A schedule where the operations from a set of concurrent transactions are interleaved.*

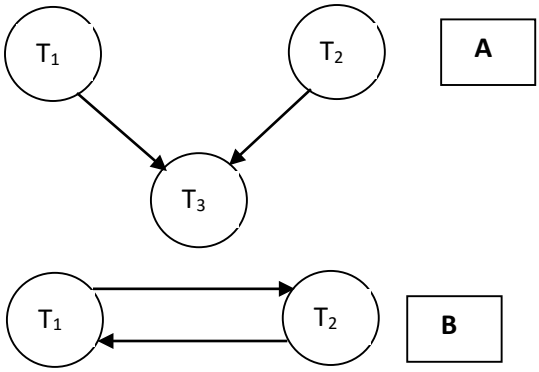
Serializable schedule *A nonserial schedule where the result to the database is equivalent to a serial schedule.*

4. Define deadlock, give an example of a schedule with a deadlock.

Deadlock: when two (or more) transactions are each waiting for locks held by the other to be released.

Time	T_{17}	T_{18}
t_1	begin_transaction	
t_2	write_lock($\mathbf{bal_x}$)	begin_transaction
t_3	read($\mathbf{bal_x}$)	write_lock($\mathbf{bal_y}$)
t_4	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	read($\mathbf{bal_y}$)
t_5	write($\mathbf{bal_x}$)	$\mathbf{bal_y} = \mathbf{bal_y} + 100$
t_6	write_lock($\mathbf{bal_y}$)	write($\mathbf{bal_y}$)
t_7	WAIT	write_lock($\mathbf{bal_x}$)
t_8	WAIT	WAIT
t_9	WAIT	WAIT
t_{10}	⋮	WAIT
t_{11}	⋮	⋮

5. Review the following precedence graphs



A. **True** or False: Schedule A is a serializable schedule.

B. True or **False**: Schedule B is a serializable schedule.

6. Describe the circumstances when the recovery manager has to 'redo' a transaction during the recovery process. Describe the circumstances when the recovery manager has to 'undo' a transaction during the recovery period.
Recovery manager has to redo all operations associated with transactions that were active since the last checkpoint in order to guarantee their effects are stored in the database. This is especially true when a no-force policy is used for buffer management. No-force policy means that committed updates are not guaranteed to be written to the database at commit time.

Recovery manager has to undo all operations associated with active transactions at the time of the crash. It undoes the operations in reverse order of the log, starting with the most recent update for the uncommitted transactions. If a steal policy is used for the buffer manager the database management system may have updated the database or given concurrent transactions access to uncommitted results. These updates must be undone, this means restoring the original value for the updated value found in the log record as the value for the object in the database.

7. Describe an algorithm associated with deadlock detection, describe an algorithm associated with deadlock prevention.
   Deadlock prevention: wait-die. The algorithm assigns each transaction a unique timestamp, that is ordered by time. If an older transaction requests a resource locked by a younger transaction then it is allowed to wait for the resource. If a younger transaction requests a resource locked by an older transaction it must abort and restart with its original timestamp. Deadlock prevention: wound-wait. Wound-wait algorithm actually allows older transactions to abort younger transactions that have a resource they want. If a younger transaction requests a resource that is allocated to an older transaction it is allowed to wait.
   Deadlock detection and recovery: waits for graph. Create a waits for graph G(N,E) where N are the nodes of the graph and represent each active transaction and E are the edges of the graph. An edge is drawn from Ni -→ Nj if Ni is waiting to lock a resource locked by Nj.
8. **True**/False An exclusive lock gives a transaction exclusive access to that data object.
9. True/**False** All relational database systems must implement a locking mechanism.
10. State why the *Wait-Die* deadlock prevention algorithm does not reassign the timestamp associated with the restarted transaction.
    In the wait-die algorithm older transactions can wait for completion of conflicting transactions. By keeping its original timestamp, the restarted transaction will have the highest precedence once all older transactions have finished. This allows the original timestamp to determine the order of the completed transactions.
11. State whether the following schedule is a serial schedule. State whether the following schedule is a conflict serializable schedule. State whether the following schedule is a recoverable schedule. Draw the precedence graph.
    Interleaves opertions so not a serial schedule, Cycle : so not a conflict serializable. Recoverable.

TIME1	READ(T1, X)
TIME2	READ(T2,X)
TIME3	WRITE(T1,X)
TIME4	WRITE(T2,X)
TIME5	COMMIT(T1)
TIME6	COMMIT(T2)