REG: 2020CA089 DBMS
NAME: SHISHU

Q1 Consider the partial schedule S involving two transations T_1 and T_2. Only the read and the write operations have been shown.
Table given in picture from.
Suppose the transaction Ti fails immediatily after time instance 9.. Justify.

If transaction fails, atomicity requires effect of transaction to be undone.
Durability states that once transaction commits, its change cannot be undone.

Recoverable Schedule :- A schedule exactly where, for every set of transaction Ti and Tj. If Tj reads a data items previously written by Ti, then commit operation of Ti precedes the commit operation of Tj.

Option B:- Schedule S is non-recoverable and cannot ensure transaction atomicity Correct, it is by definition an irrecoverable schedule so now even if we start to undo the actions one by one in order to ensure transaction atomicity. Still we cannot undo a committed transaction. hence this schedule is an incosistent state. Simply dirty read so nonrecoverable.

Q2 Consider the following two phase locking protocol. Suppose a transaction T accesses a certain set of {O1 ... Ok}. This is done in the following manner.

Step1 To acquire exclusive locks to O1 .... Ok in increasing order of their addresses.

step2 To required operations are performed

Step3 All locks are released. ~~This~~

Solution The above scenario is Conservative 2PL (or Static 2PL). In Conservative 2PL protocal, a transaction has to lock all the items in access before the transaction begins execute -ion. It is used to aboid deadlocks. Also, 2PL is conflict serializable, ~~transaction~~ therefore it gurantees serializability.

Therefore Option A

Advantages of Conservative 2PL:
→ No possibility of deadlock.
→ Ensure Serializability.

Drawbacks:
- → Less troughput and resource utilization
- → Starvation possible
- → It is deadlock free but hard to use.

Q3. Consider the following transactions with data items P and Q initialized to zero.

T_1 : read (P) :
read (Q) ;
if P = 0 then Q := Q+1 ;
write (Q) ;

T_2 : read (Q) :
read (P) ;
if Q = 0 , then P := P+1 ;
write (P)

Any non-serial interleaving of T_1 and T_2 for concurrent execution leads to ?

Solution. A schedule is said to be conflict-serializable when the schedule is conflict-equivalent to one or more serial schedules.

In the given scenario, there are two possible serial schedules:

i> T1 followed by T2
ii> T2 followed by T1.

In both of the serial schedules, one of the transactions reads the value written by other transaction as a first step. Therefore, any non-serial interleaving T1 and T2 will not be conflict serializable.

Q4. Consider the following four schedules due to three transactions using read and write on a data item X, denoted by r(X) and w(X) respectively. Which one of them is conflict serializable?

(A) $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_3(X)$; $w_2(X)$

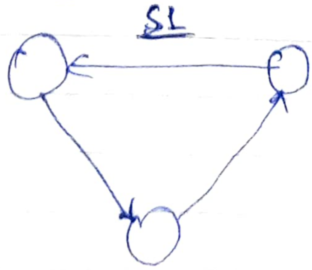
(B) $r_2(X)$; $r_1(X)$; $w_2(X)$; $r_3(X)$; $w_1(X)$

(C) $r_3(X)$; $r_2(X)$; $r_1(X)$; $w_2(X)$; $w_1(X)$

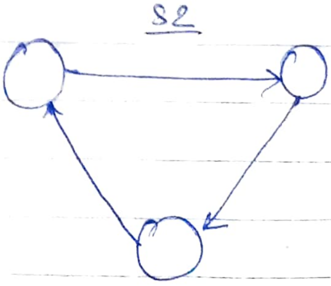
(D) $r_2(X)$; $w_2(X)$; $r_3(X)$; $r_1(X)$; $w_1(X)$

We can draw precedence graph for each schedule and for conflict serializability graph must not contain cycle.

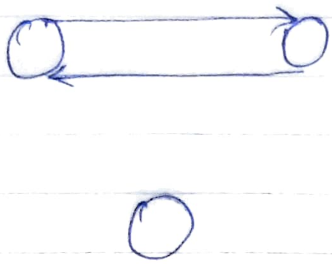
S1



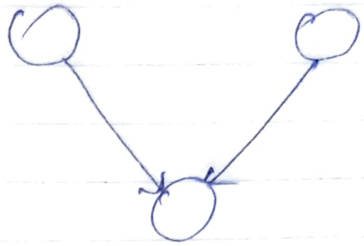
S2



S3



S4



So, option (D) is correct.