BSCCS2001: Practice with Solutions
Week 10

1. Which of the following properties of transactions will be satisfied for a serial schedule always?

[MCQ 2points]

- ◯ Atomicity of all the transactions in the schedule
- ✓ Isolation of all the transactions in the schedule
- ◯ Consistency of all the transactions in the schedule
- ◯ Durability of all the transactions in the schedule

Solution: If a schedule is serial, then it implicitly follows the Isolation property of all its constituting transactions.
Isolation property makes it essential that each of the transaction should be able to operate in such a way that it does not get affected by the execution of another transaction in parallel. In a serial schedule, transactions are executed serially with no interleaving. Hence, Isolation property is trivially satisfied.

In a serial schedule, it is not a guarantee that all transactions in it will be completely executed. Hence, Atomicity is not guaranteed.

A transaction may keep the database inconsistent in a serial schedule. Hence, it is not a guarantee for Consistency

Serial schedules have nothing to do with Durability property.

2. The number of correct statements among the two given statements are

[NAT 2points]

   1. All conflict serializable schedules are cascadeless recoverable
   2. All strict schedules are conflict serializable

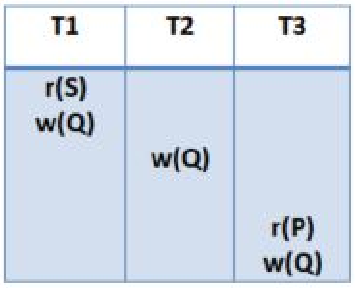
Ans: 0

Solution: Consider the schedule
S1: *r1(A), w1(A), r2(A), T2.commit, w1(B), T1.commit*
This schedule is conflict serializable but not recoverable. Hence, statement 1 is false

Consider the schedule
S2: *r1(X), w2(X), w2(Y), T2.commit, r1(Y)*
This schedule is strict but not conflict serializable.
So both statements are incorrect.

3. What is the number of schedules which are view equivalent to the given schedule **S**?

[NAT 2points]



T1	T2	T3
r(S)		
w(Q)		
	w(Q)	
		r(P)
		w(Q)

Figure 1: S

Ans: 11

Solution: Two schedules S1 and S2 are said to be view equivalent, if they satisfy all the following conditions:

1. Initial read of all data items must be done by same transactions in both schedules
2. Final write of all data items must be done by same transactions in both schedules
3. If in schedule S1, the transaction T1 is reading a data item updated by T2 and then, in schedule S2. T1 should read the value after the write operation of T2 on same data item

In the given schedule there are total 5 operations- 2 writes and 3 reads. This schedule does not have any instance where the update done by one transaction is read by another, and hence all possible arrangements of these 5 operations will follow the 3rd and 1st conditions implicitly.
But to make the schedules follow 2nd condition, we must ensure that w3(Q) is kept at its place.
Thus, except w3(Q) we have 4 operations in a grouping of $(2, 1, 1)$ for $(T1, T2, T3)$.
Hence, the number of possible arrangements following the rules are

$= \frac{4!}{2!1!1!} = 12$

Since the given schedule is itself one arrangement, the number of other view equivalent schedules are 11

For the given schedule **S**, answer questions 4 and 5
S: *r5(Z), w1(Y), r2(Y), w3(Y), r4(Y), w2(P), r5(P), w4(X), r1(Q), r5(X), w5(Y)*

4. The schedule **S** is serializable to which of the following serial schedules?[MCQ 2points]

   ✓ $T3 \rightarrow T4 \rightarrow T1 \rightarrow T2 \rightarrow T5$

   ◯ $T1 \rightarrow T5 \rightarrow T3 \rightarrow T2 \rightarrow T4$

   ◯ $T5 \rightarrow T4 \rightarrow T3 \rightarrow T2 \rightarrow T1$

   ◯ $T3 \rightarrow T2 \rightarrow T5 \rightarrow T1 \rightarrow T4$

Solution: As the read-write relations should be maintained, hence
Transaction T2 must be after T1.
Transaction T4 must be after T3
Last write of attribute Y must be preserved. Therefore, T5 must be the last transaction in a serial schedule.
Thus, the answer is : $T3 \rightarrow T4 \rightarrow T1 \rightarrow T2 \rightarrow T5$

5. Given,
   m = Number of serial schedules to which **S** is equivalent
   n = Number of serial schedules to which **S** is conflict equivalent.
   What is the value of m−n?

   [NAT 2points]

Ans: 1

Solution:
$T3 \rightarrow T4 \rightarrow T1 \rightarrow T2 \rightarrow T5$: is a serial schedule equivalent to **S**
$T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \rightarrow T5$: is a serial schedule conflict equivalent to **S**.

6. Tom is working as a System Designer in a reputed firm. He has 3 transactions to be analyzed as follows. [NAT 2points]
T1: *w1(A), w1(C).*
T2: *r2(D), w2(E).*
T3: *w3(B).*
His analysis involves checking each possible concurrent schedule that can be made using the transactions.
How many schedules does Tom have to check?

Ans: 30

Solution: If a schedule has 3 transactions with a, b, c number of operations in each respectively, then the number of concurrent schedules that can be formed using these transactions = $\frac{(a+b+c)!}{a!.b!.c!}$

Keep in mind that there should not be any reader-writer relation among the operations. Otherwise, all possible arrangements will not hold.
Putting a= 2, b=2, c=1 we get the answer = 30

7. Schedule **S** is as given: *w2(P), w1(P), w3(P), w2(Q), w1(Q)*
Consider the statements:

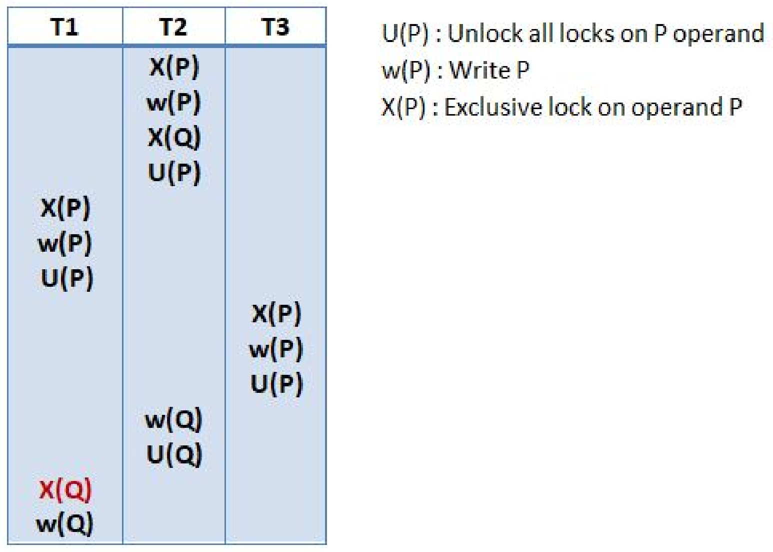
- All Conflict serializable schedules are 2-P lockable
- The given schedule **S** is Conflict serializable
- The given schedule is 2-P lockable

The number of correct statements is

[MCQ 2points]

- ○ 0
- [✓ 1]
- ○ 2
- ○ 3

Solution: Let's try to apply 2 phase locking on the schedule:

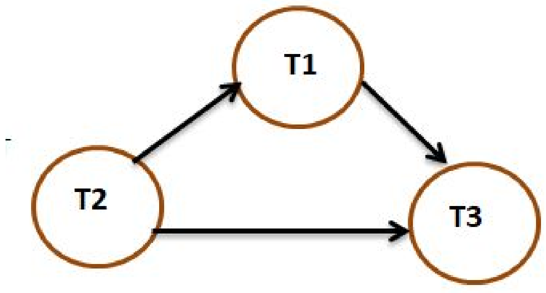


T1	T2	T3
	X(P)	
	w(P)	
	X(Q)	
	U(P)	
X(P)		
w(P)		
U(P)		
		X(P)
		w(P)
		U(P)
	w(Q)	
	U(Q)	
X(Q)		
w(Q)		

U(P) : Unlock all locks on P operand
w(P) : Write P
X(P) : Exclusive lock on operand P

We see that the red colored X(Q) is not allowed according to the rule of 2 phase locking protocol, since unlocking phase of T1 has started already because U(P) was done previously by T1. So this schedule is not 2 phase lockable.

The precedence graph of the given schedule is :



We observe that there exists no cycle in this graph and hence the schedule is Conflict serializable.

Hence, only the second statement is true.
The number of correct statements is 1.

8. Schedule **S** is as given: *w2(A), r1(A), r3(A), w1(A), w3(A), r2(B), w1(C)*
If the timestamps for transactions T1, T2, T3 are 20, 5, 10 respectively, then choose all the correct options.

[MSQ 2points]

- ✓ Timestamp ordering protocol does not allow the transaction.
- ◯ Timestamp ordering with Thomas Write rule allows the transaction.
- ✓ Neither Timestamp ordering nor Thomas Write rule allows the transaction
- ◯ None of the above

Solution: When T3 tries to perform w3(A), it checks what is the read and write timestamp of A.
T1 has both read and written A previously in the schedule. Therefore, both read and write timestamp of operand A will be set to 20, as a result of which w3(A) will not be performed since $T3.timestamp(10) < A.writetimestamp(20)$.
So T3 is rolled back by Timestamp ordering protocol.

Even Thomas write rule will not allow T3 to get executed because Thomas write rule ignores a late write only if the write timestamp of the operand is greater. In the given schedule, both read and write timestamps of operand A are greater than the timestamp value of T3.
So neither of the two protocols allow this schedule.
So correct choices are option 1 and 3.

9. Schedule **S** is as given: *w2(A), r3(A), w1(A), w3(A), r2(B), w1(C)*
If the timestamps for transactions T1, T2, T3 are 20, 5, 10 respectively, then choose the correct options from the following.

[MSQ 2points]

- ✓ Timestamp ordering protocol does not allow the transaction
- ✓ Timestamp ordering with Thomas Write rule allows the transaction
- ○ Neither Timestamp ordering nor Thomas Write rule allows the transaction
- ○ None of the above

Solution: When T3 tries to perform w3(A), it checks what is the read and write timestamp of A.
As T1 has the greatest timestamp value and it has till now only wrote operand A and never did read it, so write timestamp of A will be 20 and read timestamp of A will be 10 at this point of time.
Timestamp ordering protocol will not allow the transaction since
$writetimestampofA(20) > timestampofT3$
but Thomas write rule will allow the transaction since
$readtimestampofA(10) = timestampofT3$.
Thomas write rule ignores the write timestamp.
So, option 1 and 2 are correct choices.

10. In the *wait-die* scheme for deadlock prevention, starvation can be avoided to a large extent by: [MCQ: 2 points]

   ○ non pre-emptive rollback of younger transactions

   ✓ restarting rolled back transactions with original timestamp

   ○ pre-emptive rollback of older transactions

   ○ detecting deadlocks using precedence graphs

Solution: When both younger and older transactions are restarted, they both start with same timestamp and hence the older transactions lose the advantage of their seniority. There is a possibility that even in the next rollback step, the same transaction gets rolled back again. This can lead to starvation. When the timestamp of transactions are retained during rollback, the older transactions get precedence over younger ones, and hence the likelihood of starvation is reduced, though not completely avoided.

11. Choose the correct statement(s) about the lock-based protocols.
Statement I: A transaction will never be granted a lock on an item, even if the requested lock is compatible with locks already held on the item by other transactions.
Statement II: Any number of transactions can hold shared locks on an item, unless any transaction holds an exclusive on the item.
Statement III: If a lock cannot be granted, the requesting transaction is made to wait until all incompatible locks held by other transactions have been released.

[MCQ: 2 points]

- ◯ I & II
- [✓ II & III]
- ◯ I & III
- ◯ I, II, & III

Solution: Statements II & III are correct by the definitions of shared and exclusive locks. In the case of Statements I, the correct statement would be - a transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by other transactions.

12. Choose the correct output obtained on running the given SQL statements on Table **Employee**. [MCQ: 2 points]

EID	EName
E01	Arthur
E02	Raina
E03	Meena
E04	Arthur
E06	Joey

Table **Employee**

```
SQL> SAVEPOINT SP1;
SQL> UPDATE Employee SET EName='Jainie'
     WHERE EID='E06';
SQL> SAVEPOINT SP2;
SQL> DELETE FROM Employee WHERE EID='E02';
SQL> SAVEPOINT SP3;
SQL> UPDATE Employee SET EName='Raina'
     WHERE EID='E04';
SQL> ROLLBACK TO SP1;
```

○

EID	EName
E01	Arthur
E02	Raina
E03	Meena
E04	Arthur
E06	Jainie

○

EID	EName
E01	Arthur
E03	Meena
E04	Arthur
E06	Jainie

○

EID	EName
E01	Arthur
E03	Meena
E04	Raina
E06	Jainie

✓

EID	EName
E01	Arthur
E02	Raina
E03	Meena
E04	Arthur
E06	Joey

Solution: Since savepoint SP1 is the first savepoint added before any of the DML statements in the list are executed, a rollback to SP1 will undo all modifications (since no commit statements have been executed).

13. Given below are four statements. Match each of them with the corresponding property in the set of ACID properties.
Statement 1: Any data written to the database must be valid according to all the defined rules like the check and key constraints and triggers.
Statement 2: Every completed transaction is saved into the secondary storage.
Statement 3: During money transfer, either the amount debited from the source account must be credited to the destination account or the money should not be debited from the source account at all.
Statement 4: If multiple transactions are being executed concurrently, then the final result should be the same immaterial of the sequence in which the transactions were executed.
Let A denote Atomicity, C denote Consistency, I denote Isolation and D denote Durability. From among the given options, find the correct match.

[MCQ: 2 points]

- ○ 1 - A, 2 - C, 3 - I, 4 - D
- ✓ 1 - C, 2 - D, 3 - A, 4 - I
- ○ 1 - D, 2 - I, 3 - C, 4 - A
- ○ 1 - I, 2 - A, 3 - D, 4 - C

Solution: The statements follow from definition of each property in the set of ACID properties.