# NOC24-CS75
# Data Base Management System

Tutorial by: Adwait P. Parsodkar
Week - 7

# Question - 1

Consider the following `schedule S` of transactions $T_1$ and $T_2$.
The read operation on data item `A` is denoted by `read(A)` and the write operation on data item `A` is denoted by `write(A)`.

| $T_1$ | $T_2$ |
|---|---|
| read(A) | |
| A:=A-500 | |
| write(A) | |
| | read(C) |
| read(B) | |
| B:=B+500 | |
| | temp:=C*0.5 |
| | C:=C-temp |
| write(B) | |
| | write(C) |
| | read(D) |
| | D:=D+temp |
| | write(D) |

Which of the following is `TRUE` about the `schedule S`?

a) S is serializable both as $T_1$, $T_2$ and $T_2$, $T_1$.

b) S is not serializable neither as $T_1$, $T_2$ nor $T_2$, $T_1$.

c) S is serializable only as $T_1$, $T_2$.

d) S is serializable only as $T_2$, $T_1$.

# Solution - 1

Solution: (a)

Explanation: First, swap all non-conflicting instructions of the above schedule S. T1 and T2 both are working on the different data items.

So, S is serializable both as T1, T2 and T2, T1.

Hence, option (a) is correct.

# Question - 2

Consider the following **schedule S** involving six transactions $T_1$, $T_2$, $T_3$, $T_4$, $T_5$ and $T_6$:

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|
| | R(X) | | | | |
| | | W(X) | | | |
| | | | R(Y) | | |
| | | | W(Y) | | |
| | | | | W(X) | |
| | | | | W(Y) | |
| W(X) | | | | | |
| R(Z) | | | | | |
| | | | | | W(Z) |

R(X) denotes read operation on data item X by **transaction** $T_i$.
W(X) denotes write operation on data item X by **transaction** $T_i$.
Choose the correct option for the above **transaction schedule**.

a) The **schedule** is only **view serializable schedule**.

b) The **schedule** is only **conflict serializable schedule**.

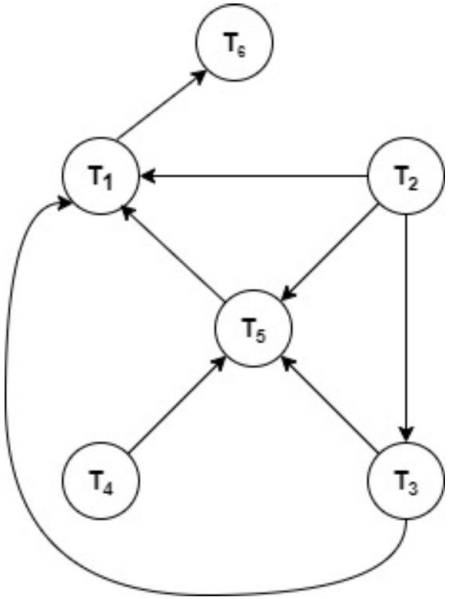
c) The **schedule** is both **view and conflict serializable schedule**.

d) The **schedule** is neither **conflict serializable** nor **view serializable schedule**.

# Solution - 2

Solution: (c)

Explanation: If we draw the precedence graph of the transactions as shown in the following, we can observe that the graph has no cycle.




So, the above schedule is a conflict serializable schedule.

All conflict serializable schedules are view serializable too.

# Question - 3

Consider the following **schedule S** involving four transactions $T_1$, $T_2$, $T_3$ and $T_4$.

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| R(X) | | | |
| R(Z) | | | |
| | W(X) | | |
| | | R(Y) | |
| | | W(Y) | |
| | | | W(X) |
| | | | W(Y) |
| | | | W(Z) |

R(X) denotes read operation on data item X by **transaction** $T_i$.
W(X) denotes write operation on data item X by **transaction** $T_i$.
Identify the possible number of **conflict serializable** schedules of the above **schedule S**.

a) 1

b) 2

c) 3

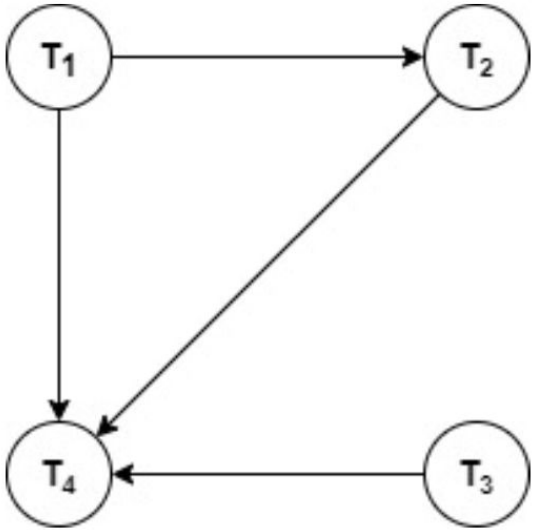
d) 4

# Solution - 3

Solution: (c)

Explanation: If we draw the precedence graph of the schedule, we can observe that the graph has no cycle. Hence, the above schedule is conflict serializable schedules.

All possible topological orderings of the above precedence graph will be the possible conflict serializable schedule.




1. T1→T3→T2→T4

2. T1→T2→T3→T4

3. T3→T1→T2→T4

Hence, option c) is correct.

# Question - 4

Consider the following `schedule S`.

| T1 | T2 | T3 |
|---|---|---|
| | | R(Y) |
| | W(Y) | |
| R(Y) | | |
| | | W(Y) |
| W(Y) | | |
| W(Z) | | |

`R(Y)` denotes read operation on data item Y by Transaction $T_i$.
`W(Y)` denotes write operation on data item Y by Transaction $T_i$.
Identify the possible number of `view serializable` schedule of the above schedule S.

a) 1

b) 2

c) 3

d) 4

# Solution - 4

Solution: (a)

Explanation:

**Step 1:** Final Update on data item: Y-T1

Final Update on data item: Z-T1

Since the final update on Y and Z is made by T1, the transaction T1 must execute after all the transactions: (T2,T3) → T1.

**Step 2:** Initial Read + Which transaction updates after read?

Y : T3 (initial read) then T2 firstly updates it.

Hence, the dependency is: T3 →T2

**Step 3:** Write Read Sequence: T2 firstly writes Y and then T1 reads Y.

Hence, the dependency is: T2 →T1

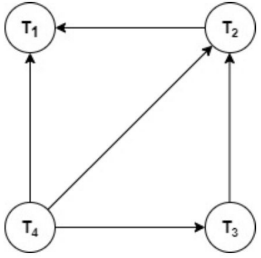
The ways we can arrange (T3→T2 → T1).

Hence, total possible view serializable schedule of S= 1.

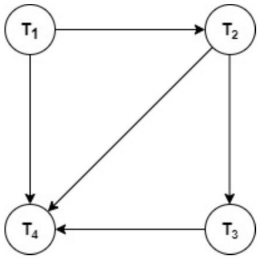
# Question - 5

Suppose in a database, there are four transactions T1, T2, T3 and T4. Transaction T1 is waiting for transactions T2 and T4, transaction T2 is waiting for transaction T3 and T4, and transaction T3 is waiting for transactions T4 to release a data item. Identify the correct wait-for graph for the above scenario.

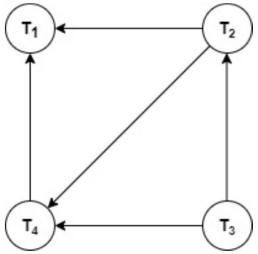
a)



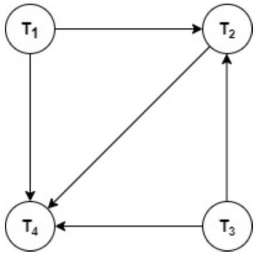

b)




c)




d)



# Solution - 5

Solution: (b)

Explanation: When $T_i$ requests a data item currently being held by $T_j$, then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph. $T_i \rightarrow T_j$, implying that $T_i$ waiting for $T_j$ to release a data item.

Hence, option b) is correct.

# Question - 6

Consider two transactions given below where `lock-X(A)` denotes $T_i$ has obtained an `Exclusive-mode` lock on item A and `lock-S(A)` denotes $T_i$ has obtained a `Shared-mode` lock on item A.

| $T_1$ |
|---|
| lock-X(A) |
| read(A) |
| A:= A-100 |
| write(A) |
| lock-X(B) |
| read(B) |
| B:= B+100 |
| write(B) |
| lock-S(C) |
| read(C) |
| commit |
| unlock(A) |
| unlock(B) |
| unlock(C) |

| $T_2$ |
|---|
| lock-X(A) |
| read(A) |
| write(A) |
| lock-S(B) |
| read(B) |
| unlock(B) |
| commit |
| unlock(A) |

Which of the following statement is (are) `true`?

a) Both $T_1$ and $T_2$ follow the `rigorous two-phase locking protocol`.

b) Both $T_1$ and $T_2$ do not follow the `rigorous two-phase locking protocol`.

c) Only $T_1$ follows the `rigorous two-phase locking protocol`.

d) Only $T_2$ follows the `rigorous two-phase locking protocol`.

# Solution - 6

Solution: (c)

Explanation:

Transaction T1 first commits and then unlocks all locks(Exclusive-mode lock and Shared-mode lock). That is why, T1 follows the rigorous two-phase locking protocol. But transaction T2 first unlocks all Shared-mode lock and then commits.

So, option (c) is correct.

# Question - 7

Suppose in a database, there are three transactions T1, T2 and T3 with timestamp 12, 14 and 16 respectively. T1 is holding a data item which T2 and T3 are requesting to acquire. Which of the following statement(s) is (are) correct in respect of Wound-Wait Deadlock Prevention scheme?

a) Transaction T2 will wait for T1 to release the data item.

b) Transaction T3 will wait for T1 to release the data item.

c) Transaction T2 will be aborted.

d) Transaction T3 will be aborted.

# Solution - 7

Solution: (a), (b)

Explanation:

Wound-Wait Deadlock Prevention scheme: When TA1 requests data item held by TA2 (older means smaller timestamp), two cases may arise.

• If TA1 is older than TA2 then TA1 wounds (forces rollback) TA2 (TA2 will be aborted).

• If TA1 younger to TA2 then TA1 will wait for release the data item held by TA2.

In this case Transaction T2 and T3 are younger in respect of Transaction T1.

# Question - 8

Consider the following `schedule S`.

| T1 | T2 | T3 |
|---|---|---|
| R(Y) | | |
| R(Z) | | |
| W(Y) | | |
| | R(Y) | |
| | W(Y) | |
| | | R(Y) |
| | | W(Y) |
| abort | | |

`R(Y)` denotes read operation on data item `Y` by Transaction $T_i$.
`W(Y)` denotes write operation on data item `Y` by Transaction $T_i$.
Identify the correct statement(s) based on the above schedule S.

a) If T1 fails(aborted), only T2 must be rolled back.

b) If T1 fails(aborted), only T3 must be rolled back.

c) If T1 fails(aborted), both T2 and T3 must be rolled back.

d) If T1 fails(aborted), neither T2 nor T3 must be rolled back.

# Solution - 8

Solution: (c)

Explanation:

As per the cascading rollback a single transaction failure leads to a series of transaction rollbacks.

Hence, if T1 fails(aborted), both T2 and T3 must also be rolled back.

So, option (c) is correct.

Consider the following schedule S.

# Question - 9

| T1 | T2 |
|---|---|
| R(X) | |
| W(X) | |
| R(Y) | |
| | R(X) |
| COMMIT | |
| | W(X) |
| | COMMIT |

R(X) denotes read operation on data item X by Transaction $T_i$.
W(X) denotes write operation on data item X by Transaction $T_i$.
Choose the correct options for the above schedule.

a) The schedule is only **`recoverable schedule`**.

b) The schedule is only **`cascadeless schedule`**.

c) The schedule is **`recoverable schedule`** and **`cascadeless schedule`** both.

d) The schedule is neither **`recoverable nor cascadeless schedule`**.

# Solution - 9

Solution: (a)

Explanation:

Recoverable schedule: If a transaction $T_j$ reads a data item previously written by a transaction $T_i$ , the commit operation of $T_i$ must appear before the commit operation of $T_j$.

Cascadeless schedules: For each pair of transactions $T_i$ and $T_j$ such that $T_j$ reads a data item previously written by $T_i$, the commit operation of $T_i$ appears before the read operation of $T_j$.

# Question - 10

Identify the correct option(s) which is (are) used in Transaction Control Language (TCL).

a) Alter

b) Insert

c) Update

d) Savepoint

# Solution - 10

Solution: (d)

Explanation:

The following commands are used to control transactions: Commit, Rollback, Savepoint and Set Transaction.

So, option (d) is correct.

# Question - 11

Consider the following schedule S involving five transactions $T_1$, $T_2$, $T_3$, $T_4$ and $T_5$:

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ |
|---|---|---|---|---|
| | W(X) | | | |
| R(X) | | | | |
| | | R(X) | | |
| | | | R(Z) | |
| | | W(X) | | |
| | | | | W(Z) |
| R(Z) | | | | |

R(X) denotes read operation on data item X by transaction $T_i$.
W(X) denotes write operation on data item X by transaction $T_i$.
Choose the correct option for the above transaction schedule.

a) The schedule is both view and conflict serializable schedule.

b) The schedule is neither conflict serializable nor view serializable schedule.

c) The schedule is only view serializable schedule.

d) The schedule is only conflict serializable schedule.

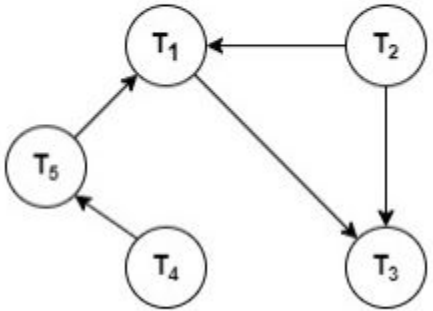
# Solution - 11

**Answer**: a)

**Explanation:** If we draw the `precedence graph` of the `transactions` as shown in the following, we can observe that the graph has no cycle.

So, the above schedule is a `conflict serializable schedule`.

All `conflict serializable schedules` are `view serializable` too.




So, option (a) is correct

# Question - 12

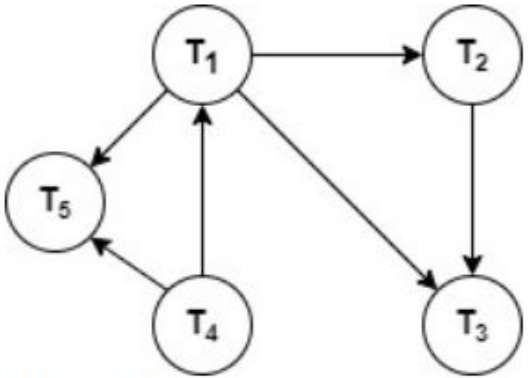
Consider the following `schedule S` involving five transactions $T_1$, $T_2$, $T_3$, $T_4$ and $T_5$:

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ |
|---|---|---|---|---|
| R(X) | | | | |
| | R(X) | | | |
| | W(X) | | | |
| | | R(X) | | |
| | | | R(Z) | |
| | | W(X) | | |
| | | | W(Z) | |
| R(Z) | | | | |
| | | | | W(Z) |

`R(X)` denotes read operation on data item `X` by `transaction` $T_i$.
`W(X)` denotes write operation on data item `X` by `transaction` $T_i$.
Identify the possible number of `conflict serializable` schedules of the above `schedule S`.

# Solution - 12

**Answer**: c)
**Explanation:** If we draw the `precedence graph` of the `schedule`, we can observe that the graph has no cycle. Hence, the above schedule is `conflict serializable schedules`.




All possible `topological orderings` of the above `precedence graph` will be the possible `conflict serializable schedule`.
1. T4 → T1 → T2 → T5 → T3
2. T4 → T1 → T2 → T3 → T5
3. T4 → T1 → T5 → T2 → T3

Hence, option c) is correct.

# Question - 13

Consider the following `schedule S` involving four transactions $T_1$, $T_2$, $T_3$ and $T_4$.

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| R(X) | | | |
| | W(X) | | |
| | | W(Z) | |
| | | | R(Z) |
| R(Z) | | | |
| W(Z) | | | |

`R(X)` denotes read operation on data item `X` by `transaction` $T_i$.
`W(Y)` denotes write operation on data item `Y` by `transaction` $T_i$.
Identify the correct option(s) that represent the order of execution of all transactions of the above `schedule S`.

a) T1→T2→T3→T4

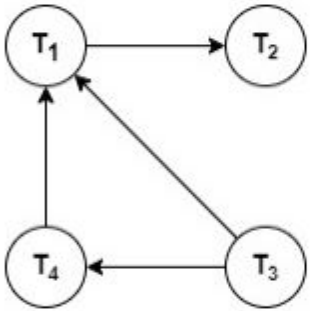
b) T3→T4→T1→T2

c) T4→T1→T3→T2

d) T3→T2→T1→T4

# Solution - 13

**Answer**: b)
**Explanation:** If we draw the `precedence graph` of the `transactions` as shown in the following, we can observe that the graph has no cycle.
So, the above schedule is a `conflict serializable schedule`.
All `conflict serializable schedules` are `view serializable` too.

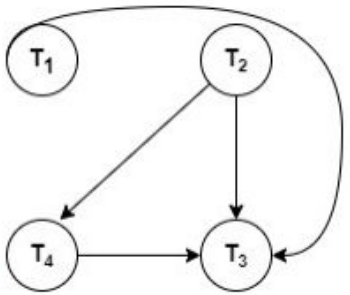



All possible `topological orderings` of the above `precedence graph` will be the possible `conflict serializable schedule`.
Hence, the correct order of execution of all transactions is: (T3→T4→T1→T2)
So, option (b) is correct

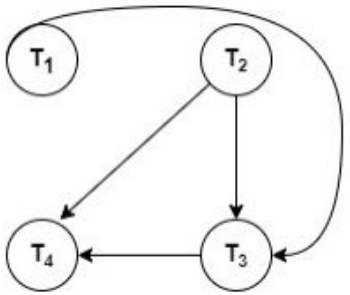
# Question - 14

Suppose in a database, there are four `transactions` $T_1$, $T_2$, $T_3$ and $T_4$. `Transaction` $T_2$ is waiting for `transactions` $T_3$ and $T_4$, `transaction` $T_1$ is waiting for `transaction` $T_3$, and `transaction` $T_3$ is waiting for `transaction` $T_4$ to release a data item.
Identify the correct `wait-for graph` for the above scenario.

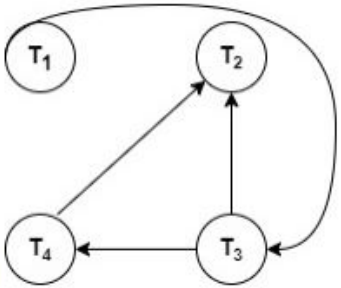
a)



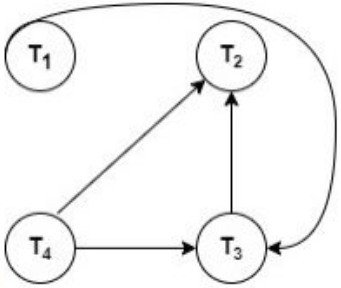

b)




c)




d)



# Solution - 14

**Answer**: b)
**Explanation:** When $T_i$ requests a `data item` currently being held by $T_j$, then the edge $T_i \rightarrow T_j$ is inserted in the `wait-for graph`. $T_i \rightarrow T_j$, implying that $T_i$ waiting for $T_j$ to release a data item.
Hence, option b) is correct.

# Question - 15

Consider the following `schedule S` of transactions $T_1$ and $T_2$. The read operation on data item `A` is denoted by `read(A)` and the write operation on data item `A` is denoted by `write(A)`.

| $T_1$ | $T_2$ |
|---|---|
| read(A) | |
| A:=A-400 | |
| write(A) | |
| | read(A) |
| | A:=A-100 |
| read(B) | |
| B:=B+400 | |
| write(B) | |
| | write(A) |
| | read(D) |
| | D:=D+100 |
| | write(D) |

Which of the following is `TRUE` about the `schedule S`?

a) S is not serializable either as $T_1$, $T_2$ or $T_2$, $T_1$.

b) S is serializable both as $T_1$, $T_2$ and $T_2$, $T_1$.

c) S is serializable only as $T_1$, $T_2$.

d) S is serializable only as $T_2$, $T_1$.

# Solution - 15

**Answer**: c)
**Explanation:** First, swap all non-conflicting instructions of the above schedule S.
Here, $T_1$ and $T_2$ both are working on the same data item A only. So, S is serializable as $T_1$, $T_2$ only.
Hence, option (c) is correct.

# Question - 16

Identify the correct statement(s) about the `lock compatibility matrix` given below, where S denotes a `shared mode lock` and X denotes an `exclusive mode lock`.

| | S | X |
|---|---|---|
| S | True | False |
| X | False | False |

a) If a transaction holds a `S lock` on a data item, other transaction will not be allowed to obtain a `S lock` on the same data item.

b) If a transaction holds a `S lock` on a data item, other transaction will not be allowed to obtain a `X lock` on the same data item.

c) If a transaction holds an `X lock` on a data item, other transactions can not be allowed to obtain a `S lock` on the same data item.

d) If a transaction holds an `X lock` on an item, other transactions can be allowed to obtain a `S lock` on the same data item.

# Solution - 16

**Answer**: b), c)
**Explanation:** As per lock based protocols. Refer Module 34 slide 10.

# Question - 17

Suppose in a database, there are three transactions $T_1$, $T_2$ and $T_3$ with timestamps 15, 18, and 19 respectively. $T_2$ is holding some data items which $T_1$, $T_3$ are requesting to acquire. Which of the following statement(s) is (are) correct in respect of `Wait-Die` Deadlock Prevention scheme?

a) Transaction $T_1$ will wait for $T_2$ to release the data item.

b) Transaction $T_3$ will wait for $T_2$ to release the data item.

c) Transaction $T_1$ will be rollback.

d) Transaction $T_3$ will be rollback.

# Solution - 17

**Answer**: a), d)
**Explanation:** In `Wait-Die` Deadlock Prevention scheme:
`Older` transaction may `wait` for younger one to release the data item. (older means smaller timestamp)
`Younger` transactions never wait for older ones; they are `rolled back` instead.
Transaction $T_1$ is older and $T_3$ is younger in respect of Transaction $T_2$.
Hence, options a) and d) are correct.

# Question - 18

Consider two transactions given below where `lock-X(A)` denotes $T_i$ has obtained an `Exclusive-mode` lock on data item A and `lock-S(A)` denotes $T_i$ has obtained a `Shared-mode` lock on data item A. `read(A)` denotes read operation on data item A by the transaction $T_i$. `write(A)` denotes write operation on data item A by the transaction $T_i$.

| $T_1$ |
|---|
| lock-S(A) |
| read(A) |
| lock-X(B) |
| read(B) |
| write(B) |
| commit |
| unlock(A) |
| unlock(B) |

| $T_2$ |
|---|
| lock-X(A) |
| read(A) |
| write(A) |
| lock-S(B) |
| read(B) |
| unlock(A) |
| unlock(B) |
| commit |

Which of the following statement(s) is/are `true`?

a) $T_1$ follows the `rigorous two-phase locking protocol` only, but $T_2$ follows the `strict two-phase locking protocol`.

b) $T_1$ follows the `rigorous two-phase locking protocol` only, but $T_2$ follows the `two-phase locking protocol` only.

c) Both $T_1$ and $T_2$ follow the `rigorous two-phase locking protocol`.

d) Both $T_1$ and $T_2$ do not follow the `rigorous two-phase locking protocol`.

# Solution - 18

**Answer**: b)
**Explanation:** Transaction $T_1$ after commit unlocks all `Shared-mode, and Exclusive-mode lock`. That is why, it follows the `rigorous two-phase locking protocol`.

Transaction $T_2$ before commit unlocks all locks (`Shared-mode lock, Exclusive-mode lock`). That is why, it does not follow the `strict two phase locking protocol` as well as the `rigorous two-phase locking protocol`. It is following `two-phase locking protocol` only. The first is the growing phase in which it is acquiring locks, the second is one in which it is releasing locks.
Hence, option (b) is correct.

# Question - 19

Consider two schedules $S_1$ and $S_2$ as follows, where S denotes a `shared mode lock` and X denotes an `exclusive mode lock`.

| $S_1$ | |
|---|---|
| $T_1$ | $T_2$ |
| lock-X(A) | |
| read(A) | |
| write(A) | |
| unlock(A) | |
| | lock-S(B) |
| | read(B) |
| | lock-S(A) |
| | read(A) |
| lock-S(B) | |
| read(B) | |
| unlock(B) | |
| | unlock(A) |
| | unlock(B) |

| $S_2$ | |
|---|---|
| $T_1$ | $T_2$ |
| lock-S(A) | |
| read(A) | |
| lock-S(B) | |
| read(B) | |
| | lock-S(A) |
| | read(A) |
| unlock(A) | |
| unlock(B) | |
| | lock-X(B) |
| | read(B) |
| | write(B) |
| | unlock(A) |
| | unlock(B) |

Identify the correct statement from the following which relates to whether the `schedules are deadlock free`. Please note that if any schedule suffers from `deadlock`, some operations of the transactions in that schedule may not be executed.

a) Both $S_1$ and $S_2$ will suffer from deadlock.

b) $S_1$ will suffer from deadlock, $S_2$ will not suffer from deadlock.

c) $S_1$ will not suffer from `deadlock`, $S_2$ will suffer from deadlock.

d) Neither $S_1$ nor $S_2$ will suffer from deadlock.

# Solution - 19

**Answer: d)**

**Explanation:** In $S_1$, $T_1$ after unlocking `exclusive mode lock on (A)` and $T_2$ has requested `shared mode lock on (A)`. Hence, it is granted. Similarly, $T_2$ is holding `shared lock on (B)` and $T_1$ has requested `shared mode lock on (B)`.
If a transaction holds a S lock on a data item, other transaction will be allowed to obtain a S lock on the same data item.
Hence, it will be also granted to $T_1$ and no deadlock will occur in $S_1$.

In $S_2$, $T_1$ has acquired `shared mode lock on (A) and (B)`. Then, $T_2$ wants to acquire `shared mode lock on (A)`. It will be granted to $T_2$.
After unlocking on data item B by $T_1$, $T_2$ requested to acquire `exclusive mode lock on (B)` hence, it is also granted to $T2$ and no deadlock occurs in $S2$.
Hence, option (d) is correct.

# Question - 20

Consider the following two schedules S1 and S2.

| S1 | |
|---|---|
| $T_1$ | $T_2$ |
| R(X) | |
| W(X) | |
| COMMIT | |
| | R(X) |
| | W(X) |
| | COMMIT |

| S2 | |
|---|---|
| $T_1$ | $T_2$ |
| R(X) | |
| W(X) | |
| | R(X) |
| COMMIT | |
| | W(X) |
| | COMMIT |

R(X) denotes read operation on data item X by Transaction $T_i$.
W(X) denotes write operation on data item X by Transaction $T_i$.

Which of the following statement(s) is/are true for the above two schedules S1 and S2?

a) Both schedules S1 and S2 are Recoverable Schedules.

b) Both schedules S1 and S2 are Cascadeless Schedules.

c) The schedule S1 is Cascadeless Schedule, the schedule S2 is Recoverable Schedule.

d) The schedule S1 is not a Recoverable Schedule, the schedule S2 is Cascadeless Schedule.

# Solution - 20

**Answer**: a), c)

**Explanation:** `Recoverable Schedule`: If a transaction $T_j$ reads a data item previously written by a transaction $T_i$ , the commit operation of $T_i$ must appear before the commit operation of $T_j$.

`Cascadeless schedules`: For each pair of transactions $T_i$ and $T_j$ such that $T_j$ reads a data item previously written by $T_i$, the commit operation of $T_i$ appears before the read operation of $T_j$.

In S1, $T_2$ read the data item X was previously written by $T_1$, $T_1$ commited before the read(X) and write(X) operation by $T_2$ . Hence, the schedule is `recoverable schedule as well as cascadeless schedule`.

In S2, $T_2$ read the data item X was previously written by $T_1$, $T_1$ commited after read(X) by $T_2$ but before the commit of $T_2$. Hence, the schedule S2 is `recoverable schedule` only.

Hence, options (a) and (c) are correct.