COMPUTER SCIENCE

Database Management System

Transaction & Concurrency Control

Transaction Concept Part-02

Lecture_3

Vijay Agarwal sir

01 Serializable Schedule

02 Conflict & View Serializable

Transaction

- Transaction Concept
- Read | write | commit
- ACID Properties
- Transaction State

Schedule

- Serial Schedule ($m!$ m: # of transaction)
- Non serial Schedule.

Serial Schedule

n Transaction then n! Serial Schedule

2 Transaction then

2! = 2 Serial Schedule

$\langle T_1\ T_2 \rangle$ T_1 Followed by T_2

$\langle T_2\ T_1 \rangle$ T_2 followed by T_1

3 Transaction then 3!

= 6 Serial Schedule

Non Serial Schedule .:. Interleaved execution of Two (or) More Transaction.

Let T_1 transfer 100 Rs from A to B, and T_2 transfer 10% of the balance from A to B.

Schedule 1

T_1	T_2
read (A) A=2000	
A: = A – 100	
write (A) A=1900	
read (B) B=3000	
B: = B + 100	
write (B) B=3100	
commit	
	read (A) A=1900
	temp := A * 0.1 temp=190
	A := A – temp 1900-190=1710
	write (A) A=1710
	read (B) B=3100
	B := B + temp
	write (B) B=3290
	Commit

A: 1710 + B: 3290 = 5000 Consistent

$S_1 < T_1\ T_2 >$

Serial schedule in which T_1 is followed by T_2:

Schedule 2

T_1	T_2
	read (A) A=2000
	temp := A * 0.1 temp=200
	A := A – temp
	write (A) A=1800
	read (B) B=3000
	B := B + temp
	write (B) B=3200
	Commit
read (A) A=1800	
A: = A – 100	
write (A) A=1700	
read (B) B=3200	
B: = B + 100	
write (B) B=3300	
commit	

A: 1700 + B: 3300 = 5000 Consistent

$S_2 < T_2\ T_1 >$

serial schedule where T_2 is followed by T_1

Serial Schedule

n!

<T1 T2>

<T2 T1>

All serial schedule always consistent.

Non Serial Schedule means Concurrent Execution

Schedule 3

T_1	T_2
read (A). A=2000	
A: = A – 100	
write (A) A=1900	
	read (A) A=1900
	temp := A * 0.1 temp=190
	A := A – temp
	write (A) A=1710
read (B) B=3000	
B: = B + 100	
write (B) B=3100	
commit	
	read (B) B=3100
	B := B + temp
	write (B) B=3290
	Commit

C1: A: 2000 + B: 3000 = 5000

A: 1710 + B: 3290 = 5000 Consistent

$\equiv S_1 \subset T_1, T_2 \rangle$

S_1

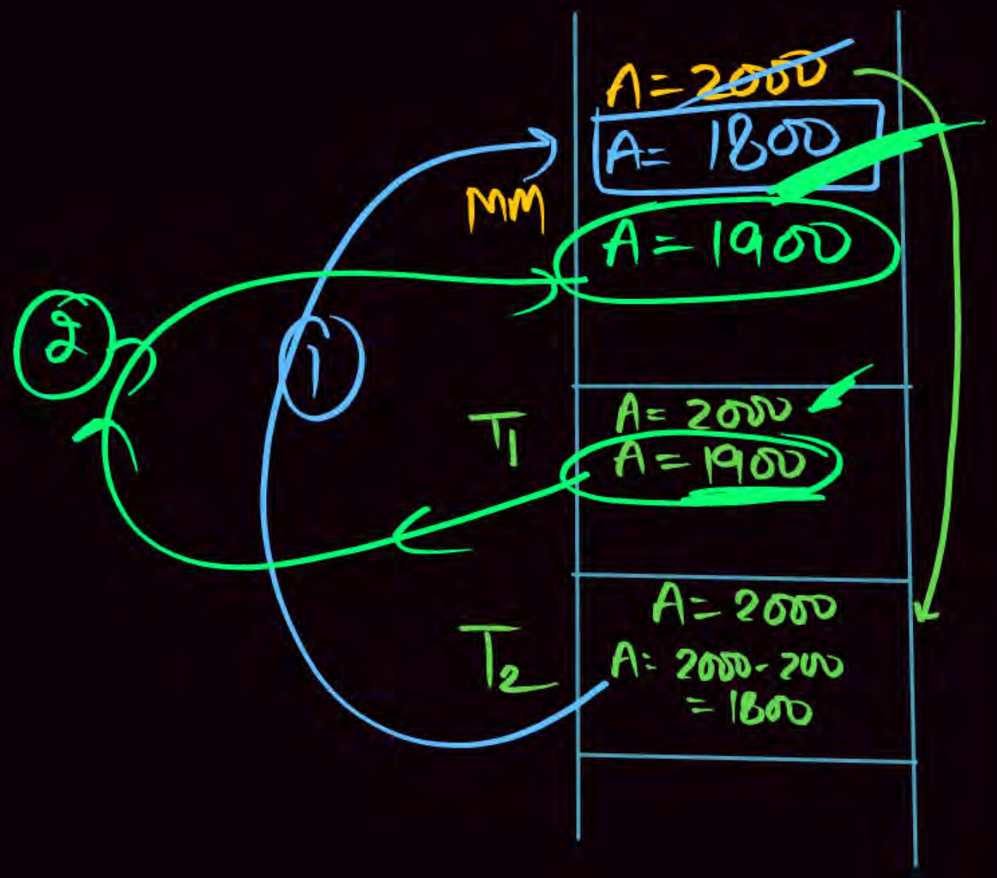
Schedule 4

T_1	T_2
A=2000 read (A)	
A: = A – 100	
1900	
	read (A) A=2000
	temp := A * 0.1 temp=200
	A := A – temp
	write (A) A=1800
A=1900 write (A)	
B=3000 read (B)	
B: = B + 100	
B=3100 write (B)	
commit	
	read (B) B=3100
	B := B + temp
	write (B) B=3300
	Commit

C2: A: 2000 + B: 3000 = 5000

A: 1900 + B: 3300 = 5200 Inconsistent

C_2



DB

A=2000

B=3000

Serial Schedule

- After Commit of one transaction, begins (Start) another transaction.
- Number of possible serial Schedules with 'n' transactions is "n!"
- The execution sequence of Serial Schedule always generates consistent result.

Example

S : $R_1(A)$ $W_1(A)$ Commit (T_1) $R_2(A)W_2(A)$ commit (T_2).

$\langle T_1, T_2 \rangle$; T_1 followed by T_2.

Advantage

- Serial Schedule always produce correct result (integrity guaranteed) as no resource sharing.

Disadvantage

- Less degree of concurrency.
- Through put of system is low.
- It allows transactions to execute one after another.

n: # of transaction.

Note: Serial Schedule (n!) are always Consistent.

But Non Serial Schedule May or May Not be Consistent.

But we execute a schedule in Concurrent Manner.

Why Concurrent execution ?

- To Improve CPU Utilization
- Enhanced throughput
- Decrease the waiting time
- Fast Response, Effective Utilization of Resource etc.

Enjoyjup
cc
c
Doubt

WHY ?

- Serializablity

Serializable Schedule.

Concurrent Execution $\equiv$ Any Serial Schedule $\Rightarrow$ Consistent

& the Process is called serializability.

1. Conflict Serializable
2. View Serializable

Serializable Schedule

A Schedule is serializable Schedule if it is equivalent to a Serial Schedule.

(i) Conflict Serializability

(ii) View Serializability

Serializability

- Basic Assumption: Each transaction preserves database consistency.
- Thus, serial execution of a set of transactions preserves database consistency.
- A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of:
  1. Conflict serializability
  2. view serializability

conflict serializability.

Conflict Instruction

Same Data Item

$T_i \rightarrow T_j$

T_i	T_j	
R(A)	R(A)	: Non Conflict Instruction
R(A)	W(A)	Conflict Instruction / operation [swapping Not possible]
W(A)	R(A)	
W(A)	W(A)	

T_i	T_j	
W(A)	R(B)	Non Conflict Instn/operation
W(A)	W(B)	
R(A)	W(B)	

Different Data Item

Conflict Serializable.

1. Basic Concept
2. Testing Method (Precedence Graph Method) — Imp.
3. Conflict Equivalent
   (Conflict Pair)

   Conflict Equivalent Schedule.

Conflict Serializability

- If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are conflict equivalent.
- We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule

Question

S: Given Non Serial Schedule [Concurrent Execution]

S': Any Serial Schedule of S (Given Question)

Given schedule S

Q.1

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

C_1: 1710, 3290, 5000 $\equiv S_1 \langle T_1 T_2 \rangle$

[S] C_1

$\hookrightarrow \langle T_2, T_1 \rangle$? ✗

$C_1 \equiv S' \langle T_1 T_2 \rangle$

$S_1' \langle T_1, T_2 \rangle$

T_1 followed by T_2.

T_1	T_2
	Read(A)
	Write(A)
	Read(B)
	Write(B)
Read(A)	
Write(A)	
Read(B)	
Write(B)	

$S_2' \langle T_2, T_1 \rangle$

T_2 followed by T_1.

(i) $C_1 \not\equiv S_2' < T_2, T_1 >$

C_1 is Not Possible to Convert Serial Schedule S_2'

$S_2' < T_2\ T_1 >$ T_2 followed by T_1.

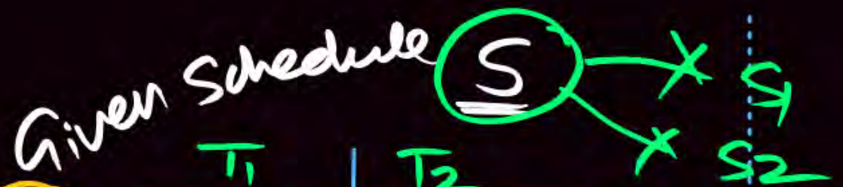
But

C_1 is possible to Convert into Serial Schedule $S_1' < T_1\ T_2 >$ (T_1 followed by T_2) So $C_1[S]$ is Conflict Serializable as a Equivalent Serial Schedule

$C_1 \equiv S_1 < T_1, T_2 >$ $< T_1\ T_2 >$ T_1 followed by T_2.

A: 1710
B: 329
5000

Given schedule S



Q.2

T_1	T_2
Read(A)	
	Read(A)
	Write(A)
Write(A)	
Read(B)	
Write(B)	
	Read(B)
	Write(B)

[S] ≠ S_2

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)
	Write(A)
	Read(B)
	write(B)

$S_1' \equiv \langle T_1, T_2 \rangle$

T_1 followed by T_2.

T_1	T_2
	Read(A)
	Write(A)
	Read(B)
	Write(B)
Read(A)	
Write(A)	
Read(B)	
Write(B)	

$S_2' \equiv \langle T_2, T_1 \rangle$

T_2 followed by T_1.

$C_2[S]$ is Not possible to Convert into

Serial Schedule $<T_1\ T_2>$ & $<T_2\ T_1>$.

$C_2[S] \neq S_1'<T_1\ T_2>$

$C_2[S] \neq S_2'<T_2\ T_1>$

$C_2[S]$ is Not Conflict Serializable

Conflict Serializability (Cont.)

- Schedule 3 can be transformed into Schedule 6, a serial schedule where T_1 follows T_2, by series of swaps of non-conflicting instructions. Therefore Schedule 3 is conflict serializable.

Schedule 3

T_1	T_2
read (A)	
Write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

Schedule 6

T_1	T_2
read (A)	
write (A)	
read (B)	
write (B)	
	read (A)
	write (A)
	read (B)
	write (B)

Concept

@ enjoying & cc

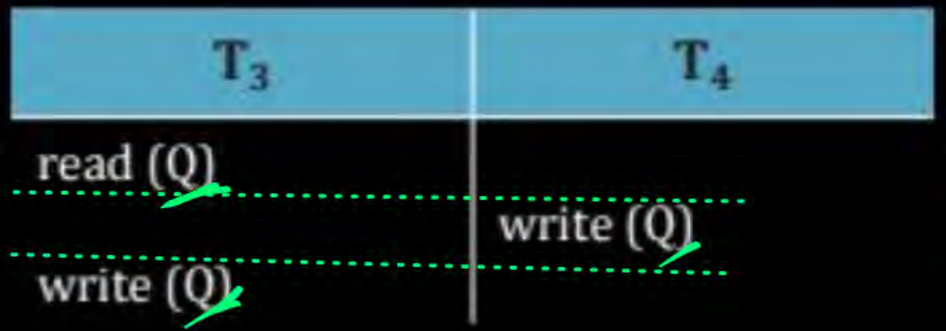
ⓑ cc

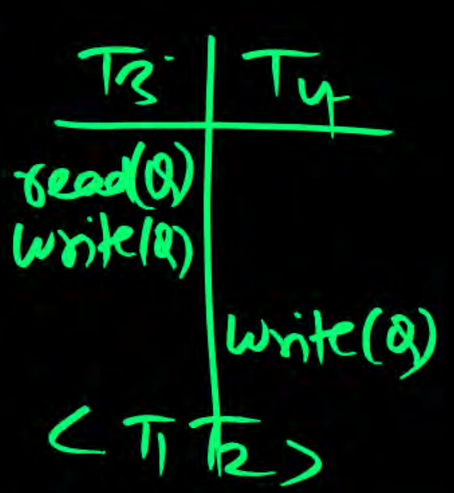
ⓒ c

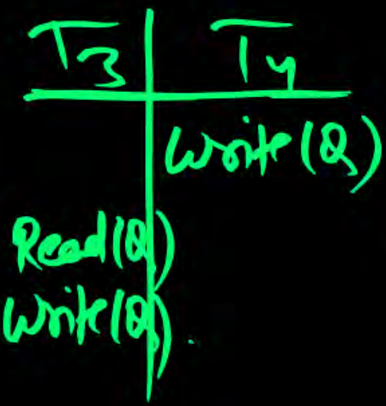
ⓓ Doubt

Conflict Serializability (Cont.)

- Example of a schedule that is not conflict serializable:







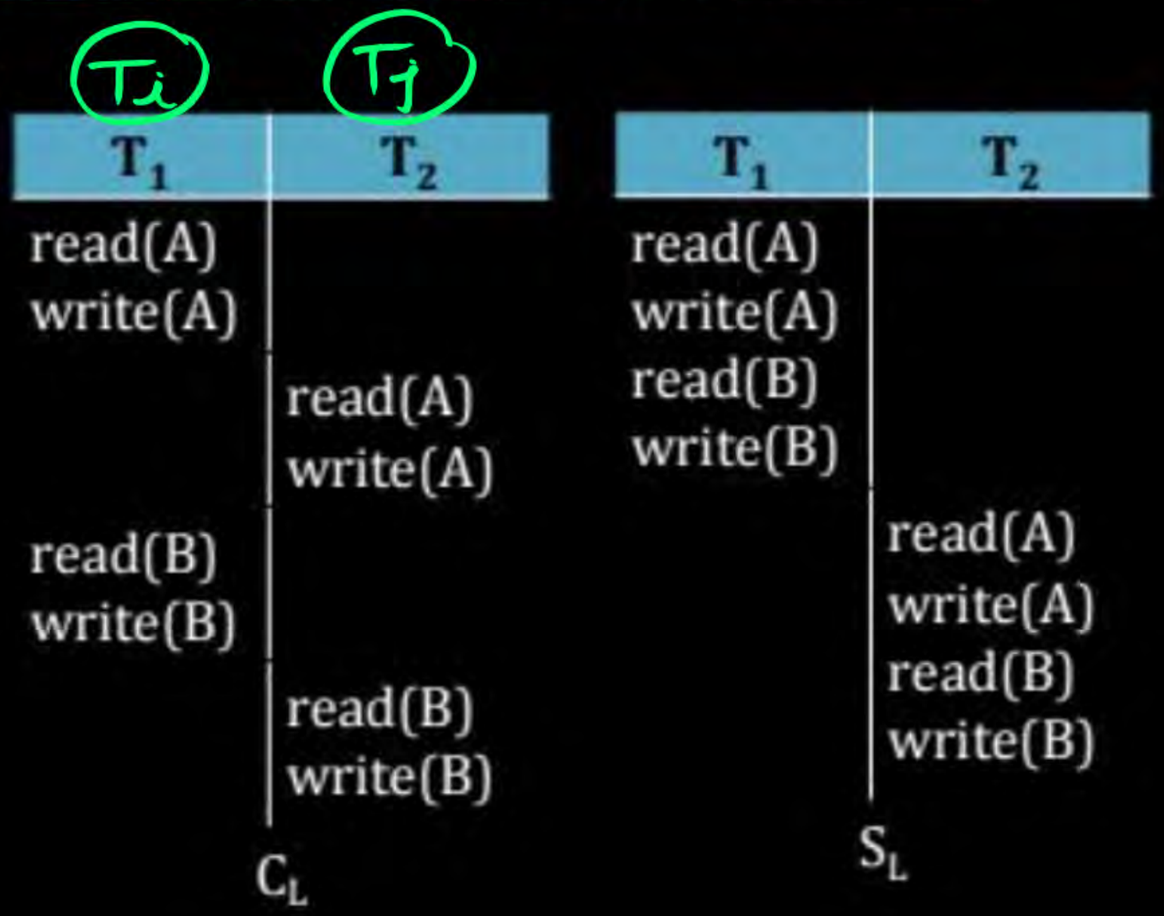
- We are unable to swap instructions in the above schedule to obtain either the serial schedule < T3, T4 >, or the serial schedule < T4, T3 >

Conflict Serializable

A schedule is said to be conflict serializable if it is conflict equivalent to a serial schedule.

Same conflicting operation order in C_1 & S_1

$\therefore$ Its $\{C_1\}$ conflict is conflict serializable.



Conflicting Instructions

- Instructions I_i, and I_j of transactions T_i and T_j respectively, conflict if and only if there exists some item Q accessed by both I_i, and I_j, and at least one of these instructions write Q. $(i \neq j)$

T_i	T_j	
R(Q)	W(Q)	Conflict
W(Q)	R(Q)	
W(Q)	W(Q)	

1. I_i, = read(Q), I_j = read(Q). I_i and I_j don't conflict.
2. I_i, = read(Q) I_j = write(Q). They conflict.
3. I_i, = write(Q) I_j = read(Q). They conflict
4. I_i= write(Q) I_j = write(Q). They conflict

- Intuitively, a conflict between I_i and I_j forces a (logical) temporal order between them.
  - If I_i, and I_j are consecutive in a schedule and they do not conflict, their results would remain the same even if they had been interchanged in the schedule.

Testing for Conflict serializability

Precedence Graph method

$G(V, E)$

[Vertex] V: Set of Transaction

[Edge] E: Edge 

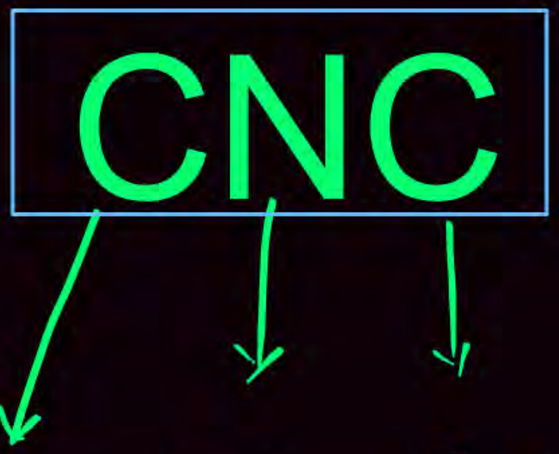
edge occur if

	T_i		T_j
(i) execute	W(A)	Before	R(A)
(ii) "	R(A)	"	W(A)
(iii)	W(A)		W(A)

Edge : Conflict operation.

Testing for Serializability

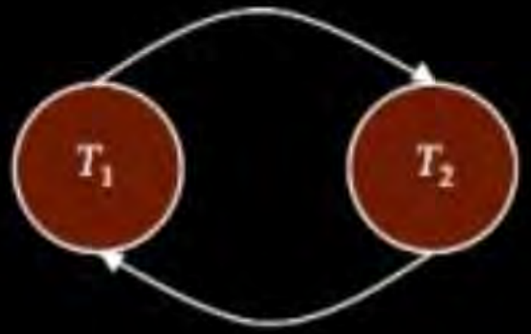
- Testing for conflict serializability.
  - Consider some schedule of a set of transactions T_1, T_2, ...T_n
  - Precedence graph — a direct graph where the vertices are the transactions (names).
  - We draw an arc from T_i to T_j if the two transaction conflict, and T_i accessed the data item on which the conflict arose earlier.
  - We may label the arc by the item that was accessed.



Conflict operation.

T_i	T_j
R(Q)	W(Q)
W(Q)	R(Q)
W(Q)	W(Q)

Example:

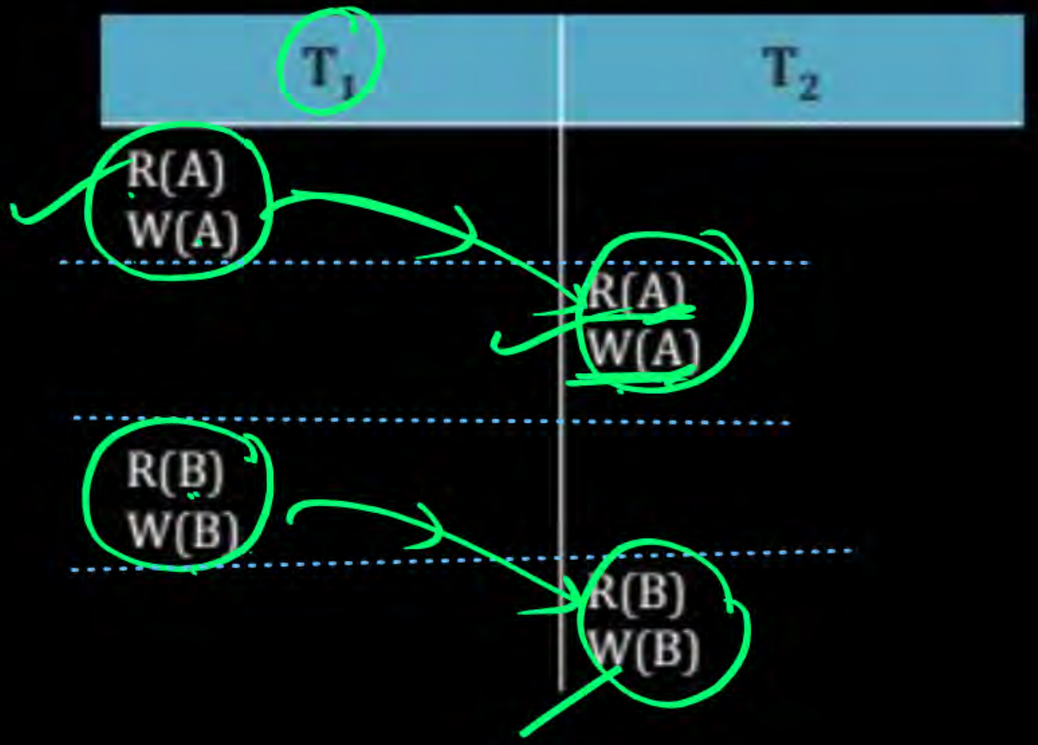


A schedule is conflict serializable if and only if its precedence graph is acyclic.

NOTE: CNC [Cycle not conflict serializable]

Q.1

S: $R_1(A)\ W_1(A)\ R_2(A)\ W_2(A)\ R_1(B)\ W_1(B)\ R_2(B)\ W_2(B)$



Conflict serializable

$\langle T_1\ T_2 \rangle$

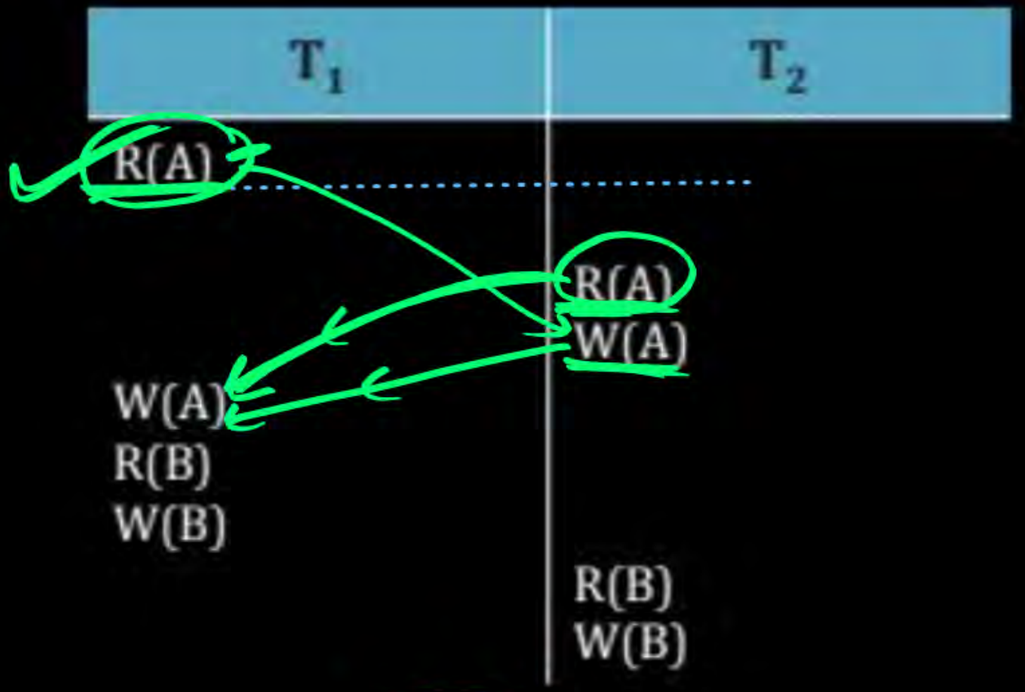
$S_1 \equiv \langle T_1\ T_2 \rangle$ Consistent

T_1 followed by T_2.

Any Conflict Operation

Q.2

$R_1(A)\ R_2(A)\ W_2(A)\ W_1(A)\ R_1(B)\ W_1(B)\ R_2(B)\ W_2(B)$



C2

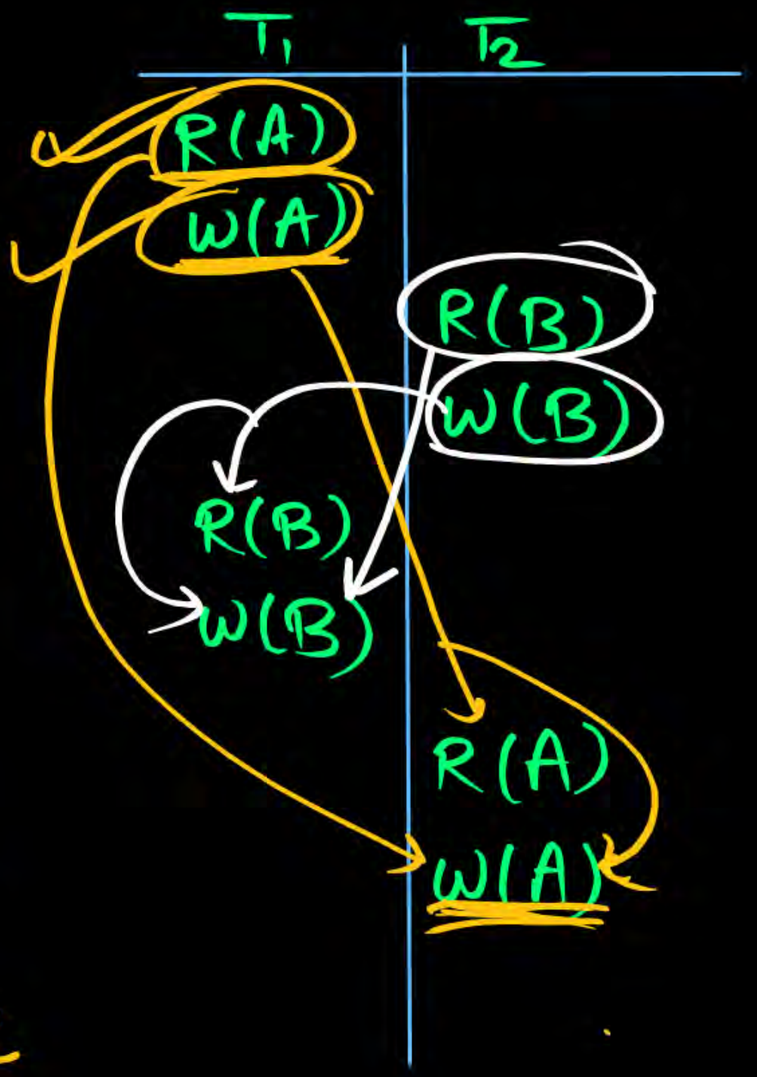
Inconsistent

CNC

Cycle Not Conflict

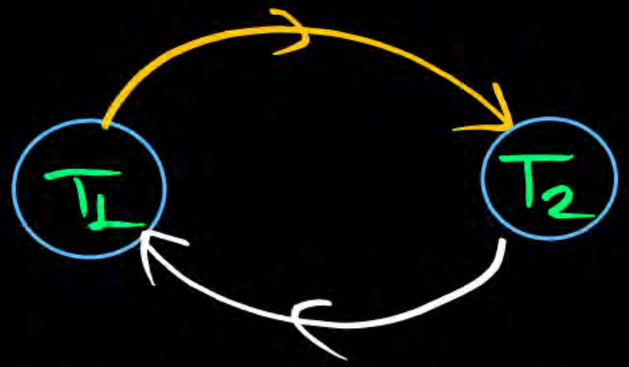
Q.3

$R_1(A)\ W_1(A)\ R_2(B)\ W_2(B)\ R_1(B)\ W_1(B)\ R_2(A)\ W_2(A)$



R(Q) → W(Q)
W(Q) → R(Q)
W(Q) → W(Q)
Create edge

CNC



Cycle Not conflict.

Serializability Order

Important Point 1:

1. If S_1, S_2 Schedule are conflict equal then precedence graph of S_1 and S_2 must be same.
2. If S_1 and S_2 have same precedence graph then S_1 and S_2 may or may not conflict equal.

Q.4

Consider the following schedules involving two transactions. Which one of the following statements is TRUE?

S_1: $r_1(X); r_1(Y); r_2(X); r_2(Y); w_2(Y); w_1(X)$

S_2: $r_1(X); r_2(X); r_2(Y); W_2(Y); r_1(Y); w_1(X)$

[2007: 2 Marks]

A. Both S_1 and S_2 are conflict serializable

B. S_1 is conflict serializable and S_2 is not conflict serializable

C. S_1 is not conflict serializable and S_2 is conflict serializable

D. Both S_1 and S_2 are not conflict serializable

Q.5

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

[2014(Set-1): 2 Marks]

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 $r_2(x); w_2(x); r_3(x); r_1(x); w_1(x)$

Q. 6

Let $r_i(z)$ and $w_i(z)$ denote read and write operations respectively on a data item by a transaction T_i. Consider the following two schedules.

S_1: $r_1(x)$ $r_1(y)$ $r_2(x)$ $r_2(y)$ $w_2(y)$ $w_1(x)$

S_2: $r_1(x)$ $r_2(x)$ $r_2(y)$ $w_2(y)$ $r_1(y)$ $w_1(x)$

Which one of the following options is correct? [MCQ: 2021: 2M]

A. S_1 is conflict serializable, and S_2 is not conflict serializable.

B. S_1 is not conflict serializable, and S_2 is conflict serializable.

C. Both S_1 and S_2 are conflict serializable.

D. Neither S_1 nor S_2 is conflict serializable.

Any Doubt ?

THANK YOU!