COMPUTER SCIENCE

Database Management System

Transaction & Concurrency Control

Lecture_7

Vijay Agarwal sir

01 Problem Due to Concurrent Execution

Serializable Schedule

1. Conflict Serializable
2. View Serializable
   - (i) Initial Read
   - (ii) Final Write
   - (iii) Updated - Read Sequence

Problem Due to Concurrent execution

1. WR / Dirty Read / Uncommitted Read Problem.
2. RW / Non/unrepeatable Read Problem.
3. WW / Lost Update Problem.
4. Phantom Table Problem.

MCQ Q.13

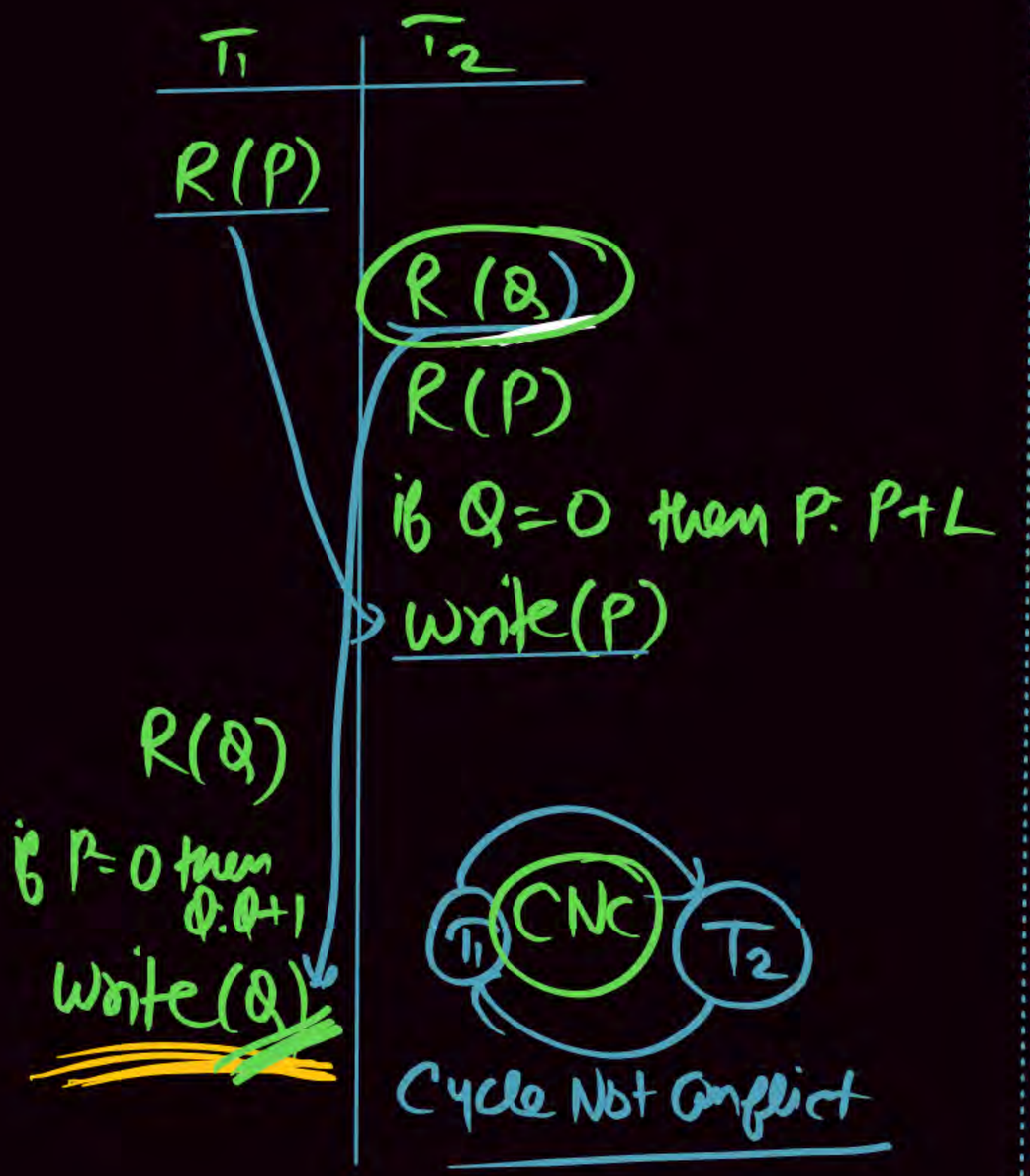
Consider the following transactions with data items P and Q initialized to zero:

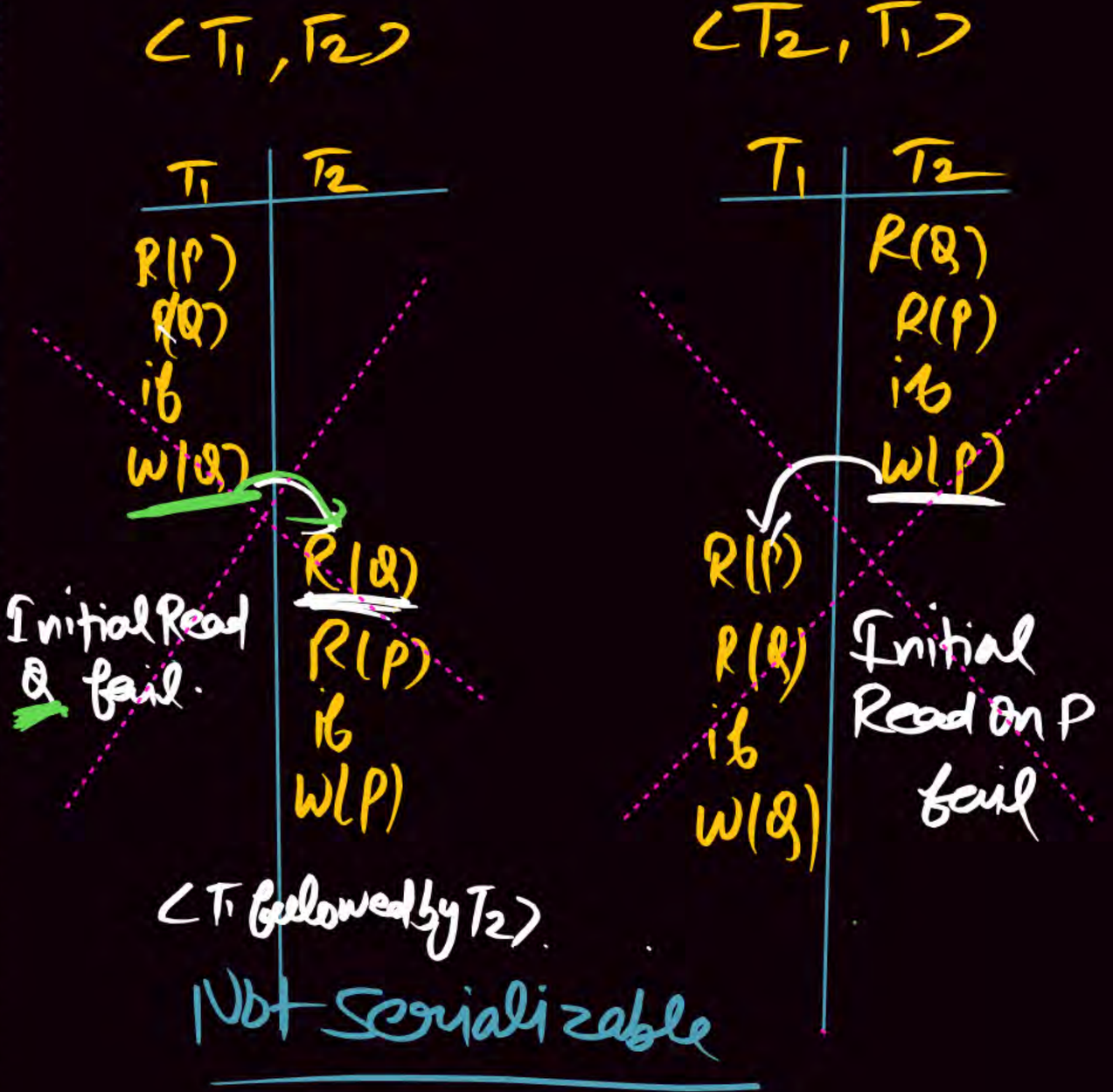
T_1: read (P);
read (Q);
if P = 0 then Q := Q + 1 ;
write (Q).

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

[GATE-2012-CS: 1M]


T1
T2
R(P)
R(Q)
R(P)
if Q=0 then P: P+L
write(P)
R(Q)
if P=0 then Q: Q+1
write(Q)
T1
CNC
T2
Cycle Not Conflict


<T1, T2>
T1
T2
R(P)
R(Q)
if
W(Q)
Initial Read Q fail.
R(Q)
R(P)
if
W(P)
<T1 followed by T2>
Not Serializable
<T2, T1>
T1
T2
R(Q)
R(P)
if
W(P)
R(P)
R(Q)
if
W(Q)
Initial Read on P fail

A. a serializable schedule...

B. a schedule that is not conflict serializable

C. a conflict serializable schedule

D. a schedule for which a precedence graph cannot be drawn

Not Serializable Schedule.

Q.14

Consider the following schedule S of transactions T_1 and T_2:

Which of the following is TRUE about the schedule S? [2004: 2 Marks]

T_1	T_2
Read(A)	
A = A–10	
	Read(A)
	Temp = 0.2 * A
	Write(A)
	Read(B)
Write(A)	
Read(B)	
B = B +10	
Write(B)	
	B = B + Temp
	Write(B)

(A) S is serializable only as T_1, T_2

(B) S is serializable only as T_2, T_1

(C) S is serializable both as T_1, T_2 and T_2 T_1

(D) S is not serializable either as T_1 or as T_2

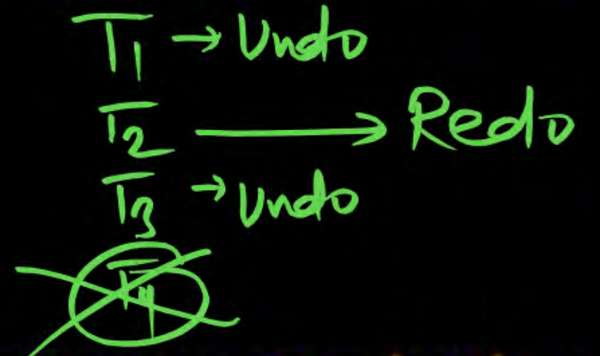
Not Conflict

Not View

MCQ Q.15

Consider a simple checkpointing protocol and the following set of operations in the log.
(start, T4); (write, T4, y, 2, 3); (start, T1);
(commit, T4); (write, T1, z, 5, 7);
(checkpoint);
(start, T2); (write, T2, x, 1, 9); (commit, T2);
(start, T3); (write, T3, z, 7, 2);
If a crash happens now and the system tries to recover using both undo and redo operations, what are the contents of the undo list and the redo list?

[GATE-2015-CS: 2M]



Redo : T_2

UNDO : T_3, T_1

A. Undo: T3, T1; Redo: T2

B. Undo: T3, T1; Redo: T2, T4

C. Undo: none; Redo: T2, T4, T3, T1

D. Undo: T3, T1, T4; Redo: T2

CHECK POINT : Those transaction Commit before the Check point Not Require Any Redo (or) Undo operation.

Those transaction Commit After the check point Require 'Redo operation'.

Those transactions Not Commit yet, Require 'Undo operation'.

Log Based Recovery.

UNDO & REDO Concept.

UNDO : OLD Value

REDO : NEW Value.

(Transaction Data Item OLD Value New Value)

Time t=1 (T_1 X 5 7)

Time t=2 (T_1 X 7 11)

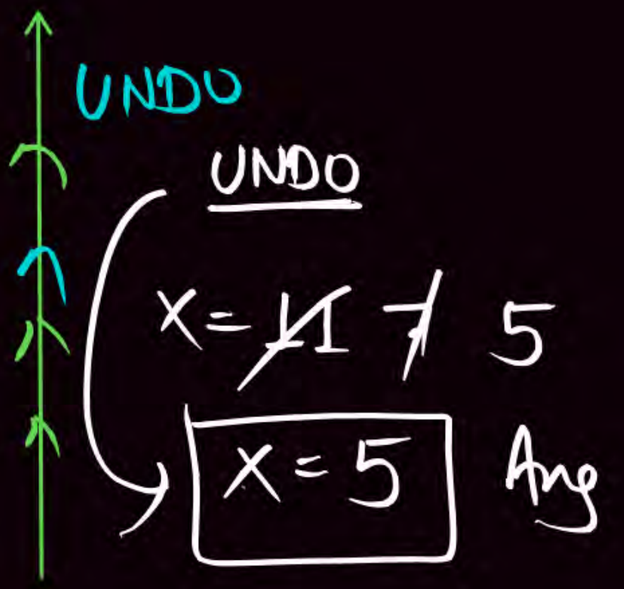
REDO : Top to Bottom

UNDO : Bottom to TOP.

REDO(New Value)

X = 5 7 11

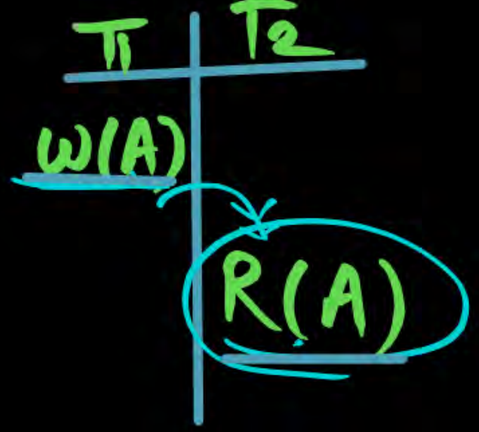
X = 11



Problem Due to Concurrent execution

1. WR / Dirty Read / Uncommitted Read Problem.
2. RW / Non/un repeatable Read Problem.
3. WW / Lost Update Problem.
4. Phantom Tuple Problem.

① WR [Write - Read] Problem / Dirty Read / Uncommitted Read



Uncommitted / Dirty Read : Here transaction T_2 Read the value of Data Item A, that is updated (Write) By Uncommitted Transaction T_1.

OR

Transaction Read a value that is updated (written) by Uncommitted Transaction.

Note

Modification (Update) by One transaction [Uncommitted Transaction) & Read by another transaction.

② RW [Read-Write] / Un/Non Repeatable Read Problem.

T_1	T_2
R(A)	
	W(A)

eg

Library DB

```
Read(A)
if(A>0)
{
A = A-1
write(A) //Book Issue.
}
else
No Book Issue.
```

Kunth Book
#Copies(A) = 10

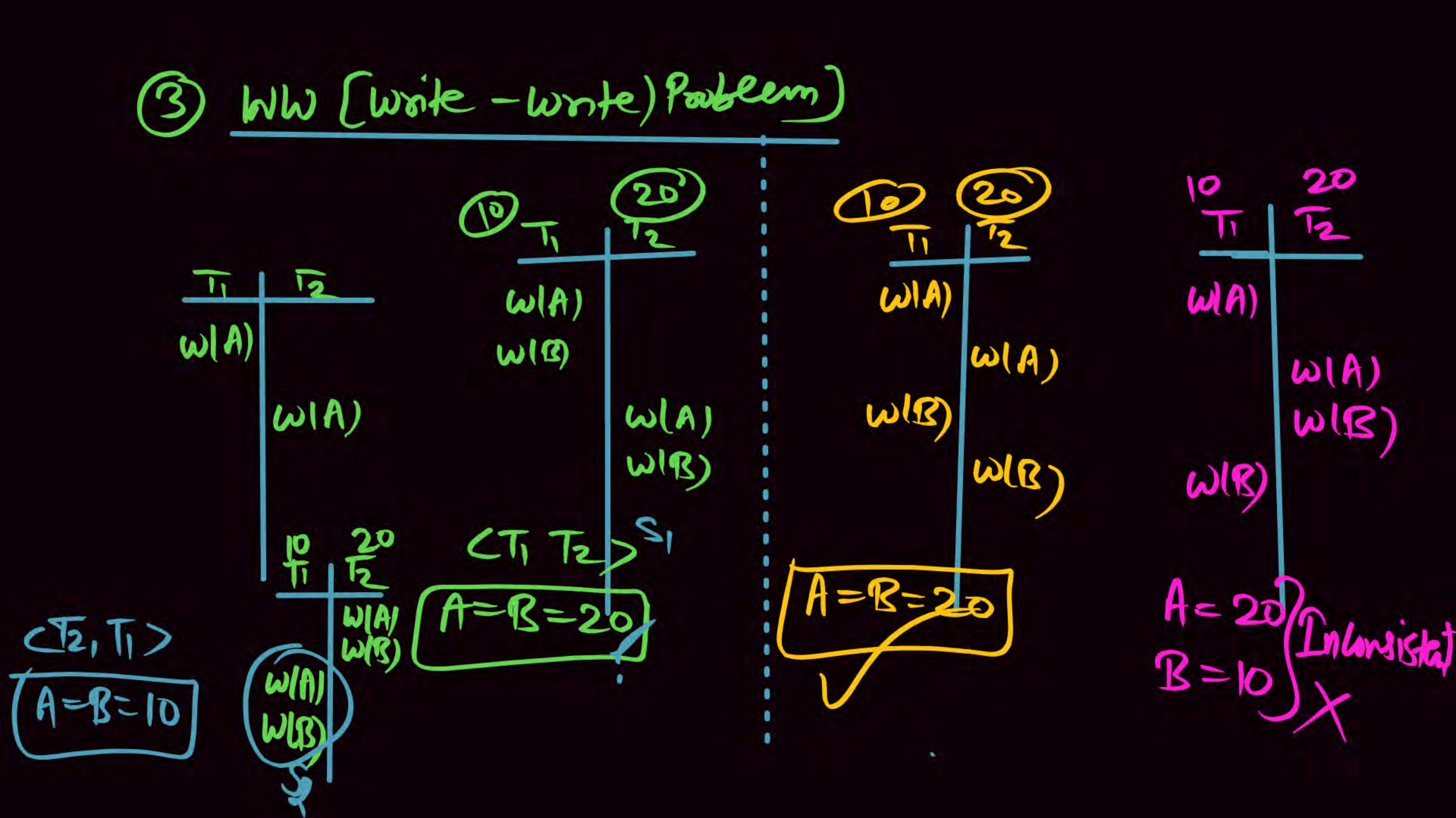
T_1	T_2
A=10 Read(A)	
if(A>0)	
{	
A = A-1	
	Read(A) A=10
	if(A>0)
	{
	A = A-1
	write(A) A=9
	}
A=9 write(A)	
}	

T_1 T_2

A=9
2 Book Issue
(!) Inconsistent


A=50000
T1
R(A)
R(A)
A=50000
A=A-47000
W(A)
A=3000
A=3000
R(A)

③ WW (Write - Write) Problem



③ WW [write - write] | Lost update Problem.

T_1	T_2
	Read(A)
A = A - 100	
	Read(A)
	temp = A * .10
	A = A + temp
write(A)	
	write(A)

A = 1000

A = 1000

A = 900

A = 900

temp = 100

A = 1100

A = 1100

1000 X .10 = 100

temp.

T_1	T_2
W(A)	
	W(A)

Suppose that Transaction T_1 & T_2 interleaved in such a manner, T_2 Read the Value of Account before T_1 Update.

& Now when transaction T_1 update 'A' then after that Transaction T_2 Update the Value of Account 'A' So Here Lost of Update Problem. (Overwritten the Value of 'A')

(Note) LOST UPDATE: If there is Two write operation of Different transaction & between these 2 write there is No Read operation then 2nd write operation Overwritten the first write Value So Lost of Updation of T_1.

(4) Phantom Tuple

Emp.

Eid	Name	Salary
1	A	5000
2	B	6000
4	D	7000

T_1	T_2
Select * From Emp WHERE Sal > 4700	
	Insert in Emp Value <5, E, 5100>
Select * From Emp WHERE Sal > 4700	

Eid	Name	Salary
1	A	5000
2	B	6000
4	D	7000
5	E	5100

Emp.

Eid	Name	Salary
1	A	5000
2	B	6000
3	C	4000
4	D	7000
5	E	5100

Here T_1 may Read a Set of Tuples from the Table Based on Some Condition.

Now Suppose other transaction T_2 insert a New Tuple Such that also satisfying the Condition of T_1 Transaction.

Now if T_1 is Repeated then T_1 will see a New Tuple that Previously did not exist, Called Phantom Tuple.

Concurrent Schedule (S + NS)

Serial Schedule

↳ m!

m is the No. of Transaction.

Non Serial Schedule

Non Serial = Total Concurrent Schedule − Serial Schedule.

$T_1 \rightarrow n_1$ operation

$T_2 \rightarrow n_2$ operation

$$\text{Total Concurrent Schedule} = \frac{(n_1 + n_2)!}{[(n_1)!]\,[n_2!]}$$

T_1 : 2 Operation

T_2 : 2 operation

Total Concurrent Schedule $= \frac{(2+2)!}{2!\,2!} = 6$

Serial Schedule $= 2! = 2$

Non Serial Schedule $= 6 - 2 = 4$ Ans

Finding Total Number of concurrent Schedule

T_1	T_2
$R_1(A)$ $W_1(A)$	$R_2(B)$ $W_2(B)$

T_1	T_2
L_1 L_2	L_3 L_4

T_1	T_2
0 0	1 1

$L_1L_2L_3L_4$

$L_3L_4L_1L_2$

$L_1L_3L_2L_4$ (or) $L_1L_3L_4L_2$

$L_3L_1L_4L_2$(or)$L_3L_1L_2L_4$

T_1	T_2
R(A) W(A)	R(B) W(B)

$S_1<T_1T_2>$

(1)

T_1	T_2
R(A) W(A)	R(B) W(B)

$S_2<T_2T_1>$

(2)

T_1	T_2
R(A) W(A)	R(B) W(B)

(3)

T_1	T_2
R(A) W(A)	R(B) W(B)

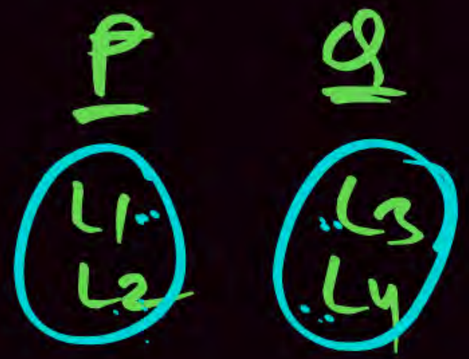
(4)

T_1	T_2
R(A) W(A)	R(B) W(B)

(5)

T_1	T_2
R(A) W(A)	R(B) W(B)

(6)



L1 L2 L3 L4
L3 L4 L1 L2
] Serially.

L1 L3 L2 L4 (OR) L1 L3 L4 L2

L3 L4 L4 L2 (OR) L3 L1 L2 L4.

Total # Concurrent Schedule $= \frac{(n_1+n_2)!}{(n_2)!(n_2)!}$

$= \frac{(2+2)!}{(2)!(2)!} = \frac{4\times3\times2}{2\times2} = 6$

$T_1 \rightarrow n_1$ operation
2 operation

$T_2 \rightarrow n_2$ operation
2 operation

Total Concurrent = 6

Total non serial Schedule = Total Concurrent - Serial schedule(m!)

m: # of transaction

= 6 - 2

Serial = 2

Total non Serial = 4

NOTE:

The Number of Concurrent schedule that can be formed Over m transaction having n_1 n_2 n_3 ....n_m operation respectively

operation

$T_1 \rightarrow n_1$

$T_2 \rightarrow n_2$

$T_3 \rightarrow n_3$

$T_m \rightarrow n_m$

$$\text{Total \# of Concurrent Schedule} = \frac{(n_1+n_2+n_3+\cdots n_m)!}{(n_1!)(n_2!)(n_2!)\ldots(n_m!)}$$

$$\text{Total \# of Non Serial Schedule} = \frac{(n_1+n_2+n_3+\cdots n_m)!}{(n_1!)(n_2!)(n_2!)\ldots(n_m!)} - m!$$

T_1 : 2 operation
T_2 : 3 operation
T_3 : 4 operation

THANK YOU!