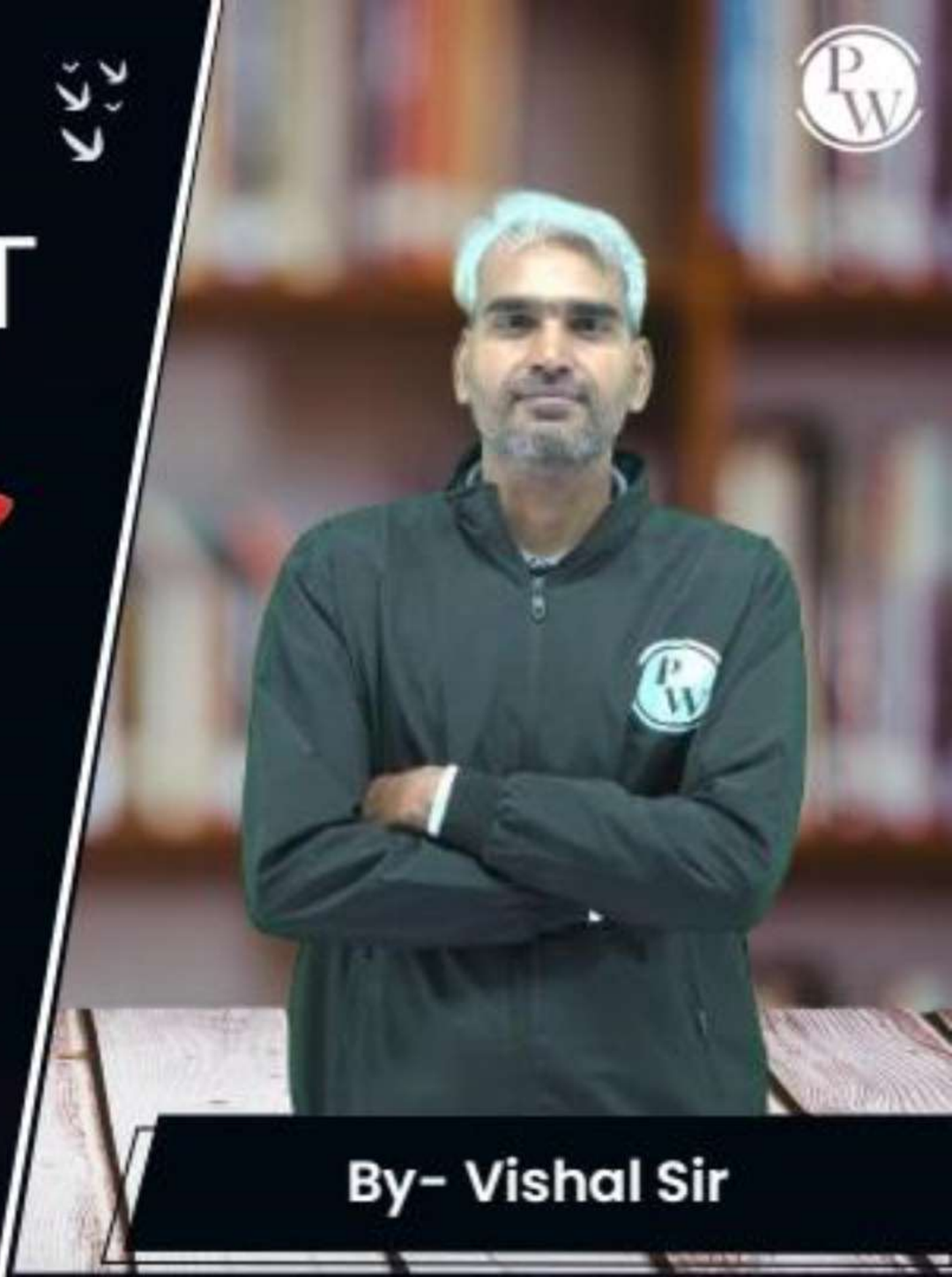
Computer Science & IT

Database Management System

Transaction & Concurrency control

Lecture No. 04



By- Vishal Sir

Recap of Previous Lecture

- Topic: Isolation { Serializable Schedule }
- Topic: Consistency
- Topic: Equivalent schedule
- Topic: Serializable and non-serializable schedule

Slide

Topics to be Covered

- Topic: Serializable and non-serializable schedule
- Topic: Problems because of concurrent execution
- Topic: RW problem
- Topic: WR problem
- Topic: WW problem
- Topic: Lost update problem

Slide

H.W. Check whether the following schedule is a serializable schedule or not.

(i) S_1

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

Non-serializable

(ii) S_2

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

* Serializable schedule
* Equivalent serial schedule is $T_1 \rightarrow T_2$

(iii) S_3

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

* Serializable schedule
* Equivalent serial schedule is $T_2 \rightarrow T_1$

(iv) S_4

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

Non-serializable

H.W: Check whether the following schedule is a serializable schedule or not.

(i) S_1

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

$R_2(A)$ ← from initial DB $\not\equiv T_1 \to T_2$

$R_2(B)$ ← value updated by $T_1 \not\equiv T_2 \to T_1$

$S_1 \not\equiv T_1 \to T_2$
$S_1 \not\equiv T_2 \to T_1$
$\therefore$ Non-Serializable

$T_1 \to T_2$

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

$R_2(A)$ ← value updated by T_1

$T_2 \to T_1$

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

$R_2(A)$, $R_2(B)$ ← from initial DB

H.W. Check whether the following schedule is a serializable schedule or not.

(ii) S_2

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

All Read from initial DB {Read are same as $T_1 \rightarrow T_2$}

'C' is finally updated by T_2 {same as $T_1 \rightarrow T_2$}

$S_2 \equiv T_1 \rightarrow T_2$

∴ is Serializable Schedule

$T_1 \rightarrow T_2$

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

all from initial DB

'C' is finally updated by T_2

$T_2 \rightarrow T_1$

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

Value updated by T_2 {not same as S_2}

∴ $S_2 \neq T_2 \rightarrow T_1$

H.W: Check whether the following schedule is a serializable schedule or not.

(iii) S_3

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

from initial DB

'B' is finally updated by T_1

$T_1 \rightarrow T_2$

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

'B' is finally updated by T_2

$\therefore T_1 \rightarrow T_2 \neq S_3$

$T_2 \rightarrow T_1$

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

from initial DB (same as S_3)

B is finally updated by T_1 (same as S_3)

$\therefore S_3 \equiv T_2 \rightarrow T_1$ $\therefore$ S_3 is serializable

H.W. Check whether the following schedule is a serializable schedule or not.

(iv) S_4

T_1	T_2
$R_1(A)$	
	$R_2(B)$ ← from initial DB
$W_1(B)$	
	$W_2(B)$ ← 'B' is finally updated by T_2

$T_1 \rightarrow T_2$

T_1	T_2
$R_1(A)$	
$W_1(B)$	
	$R_2(B)$ ← value updated by T_1 {not same as S_4}
	$W_2(B)$

$\therefore T_1 \rightarrow T_2 \not\equiv S_4$

$T_2 \rightarrow T_1$

T_1	T_2
	$R_2(B)$
	$W_2(B)$
$R_1(A)$	
$W_1(B)$ ← 'B' is finally updated by T_1 {not same as S_4}	

$\therefore T_2 \rightarrow T_1 \not\equiv S_4$

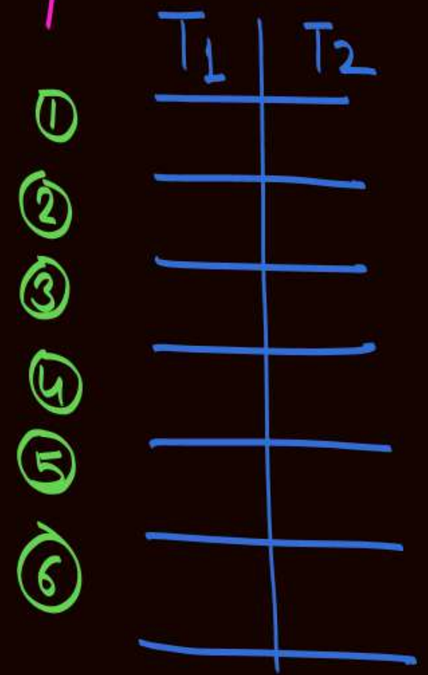
$\therefore S_4$ is non-serializable

Q:- Consider two transactions

T_1 : $R_1(A)$, $W_1(A)$, $R_1(B)$, $W_1(B)$

T_2 : $R_2(A)$, $R_2(B)$

How many different schedules are possible with operations of transaction T_1 & T_2. {i.e. Serial + non-serial}



Total no. of schedules possible $= \left({}^{6}C_4 * 1\right) * \left({}^{(6-4)}C_2 * 1\right)$

$= {}^{6}C_4 * 1 * {}^{2}C_1 * 1$

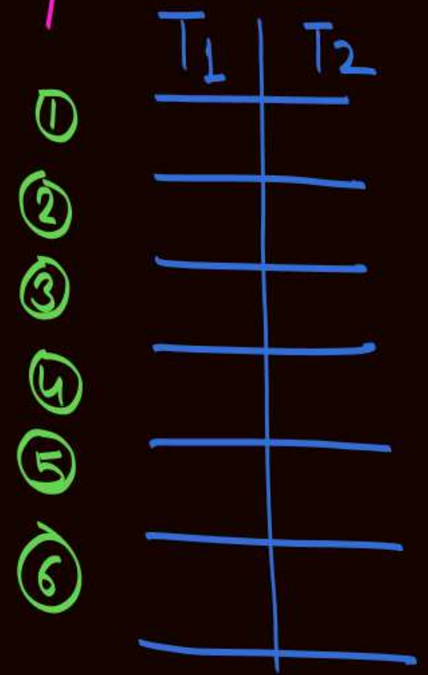
$= {}^{6}C_4 = (15)$

Q:- Consider two transactions

T_1 : $R_1(A)$, $W_1(A)$, $R_1(B)$, $W_1(B)$

T_2 : $R_2(A)$, $R_2(B)$

How many different schedules are possible with operations of transaction T_1 & T_2. {i.e. Serial + non-serial}



Total no. of schedules possible $= \left({}^{6}C_2 * 1 \right) * \left({}^{(6-2)}C_4 * 1 \right)$

$= {}^{6}C_2 * 1 * {}^{4}C_4 * 1$

$= {}^{6}C_2 = 15$

Q:- Consider two transactions

T_1 : $R_1(A)$, $W_1(A)$, $R_1(B)$, $W_1(B)$

T_2 : $R_2(A)$, $R_2(B)$

Total no. of serial schedules possible = $2! = 2$

How many non-serial schedules are possible with transactions T_1 & T_2

- No. of non-serial schedules = Total no. of schedules − No. of serial schedules

$= 15 - 2$

$= (13)$ Ans

General formula for Number of schedules :-

Let T_1, T_2 & T_3 are three transactions with 'm', 'n' and 'p' operations respectively, then how many schedules are possible with transactions T_1, T_2 & T_3.

$$\text{Total no. of schedules possible} = \left({}^{(m+n+p)}C_m * 1\right) * \left({}^{(n+p)}C_n * 1\right) * \left({}^{p}C_p * 1\right)$$

$$= \boxed{{}^{(m+n+p)}C_m * {}^{(n+p)}C_n * {}^{p}C_p}$$

Topic : Problems because of concurrent execution

1. R W problem / Read-Write Problem / Write after Read Problem
2. W R Problem / Write-Read Problem / Read after write Problem
3. W W Problem / Write-Write Problem / Write after write Problem

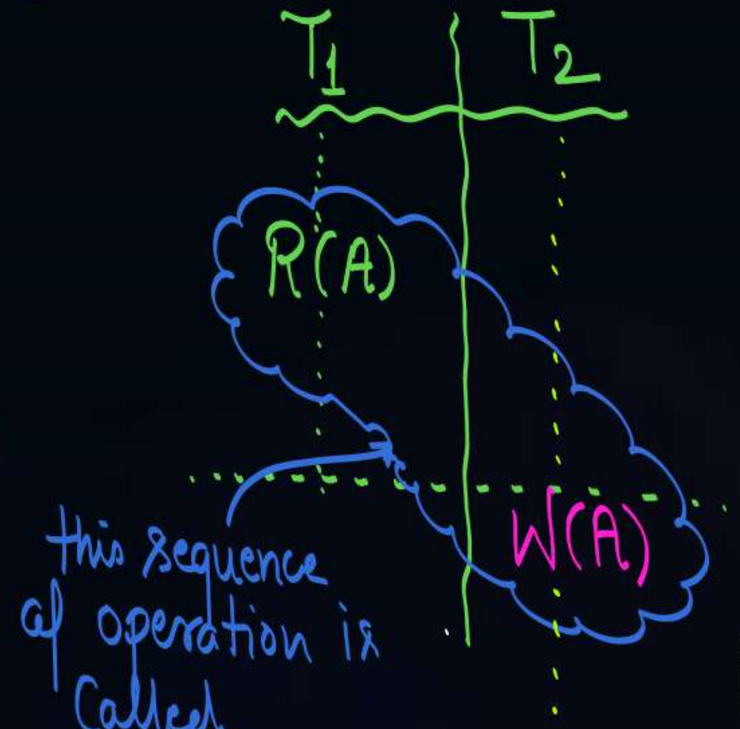
Exist only if schedule is a non-serializable schedule

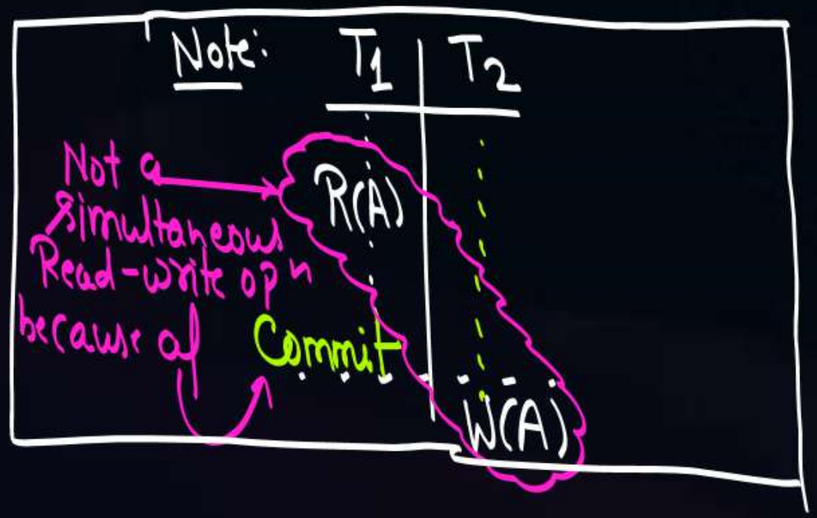
4. Lost update problem { It can exist with both serializable as well as non-serializable schedule }

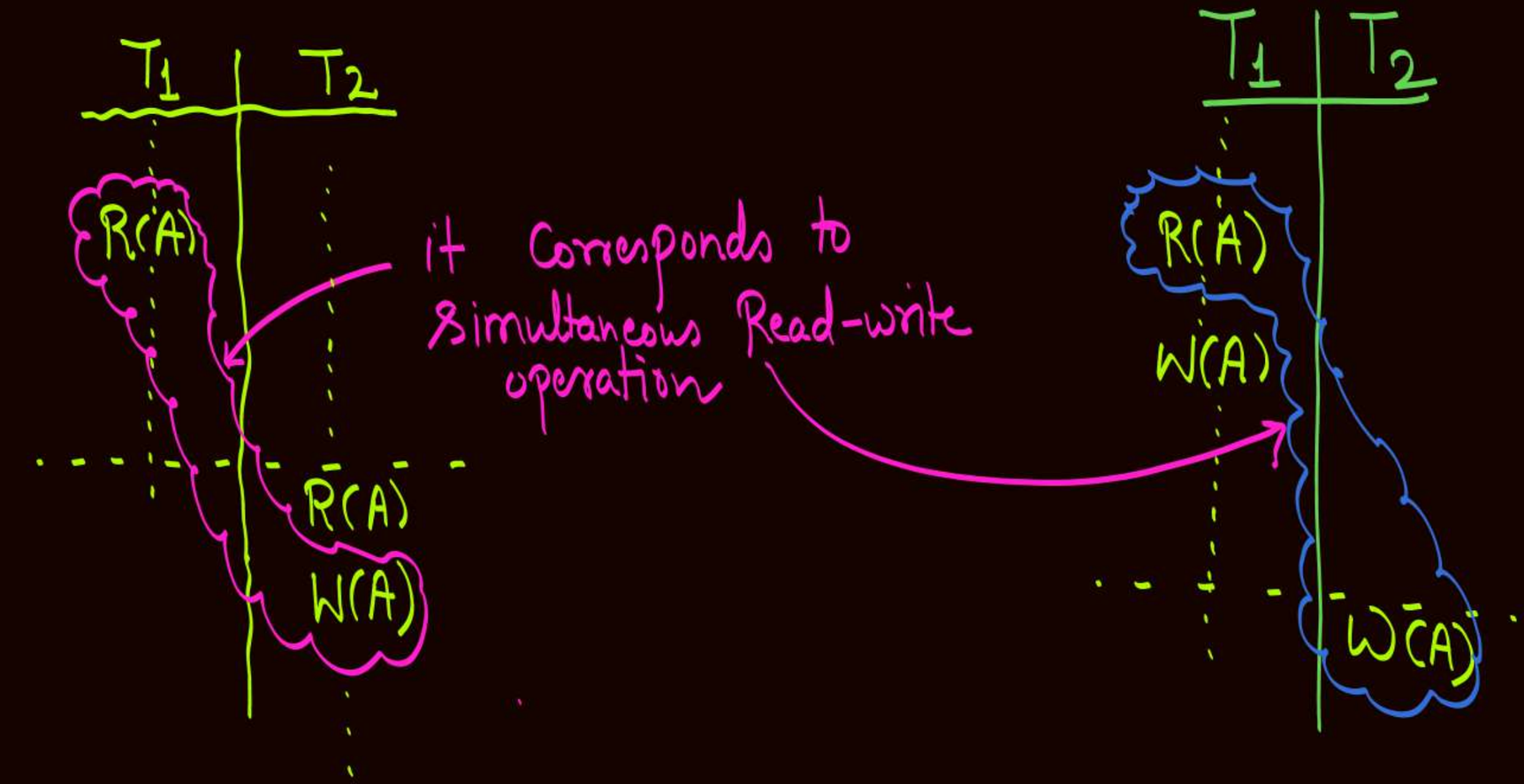
Topic : RW problem

Write-after-Read Problem :→

→ If transaction T_2 updates the dataitem 'A' which is already read by an uncommitted transaction T_1 then that sequence of operation is called Simultaneous Read-write opn






T1
T2
R(A)
R(A)
W(A)
it Corresponds to
Simultaneous Read-write
operation
T1
T2
R(A)
W(A)
W(A)

* RW Problem / Write after Read Problem :-

A schedule is said to have RW Problem if and only if,

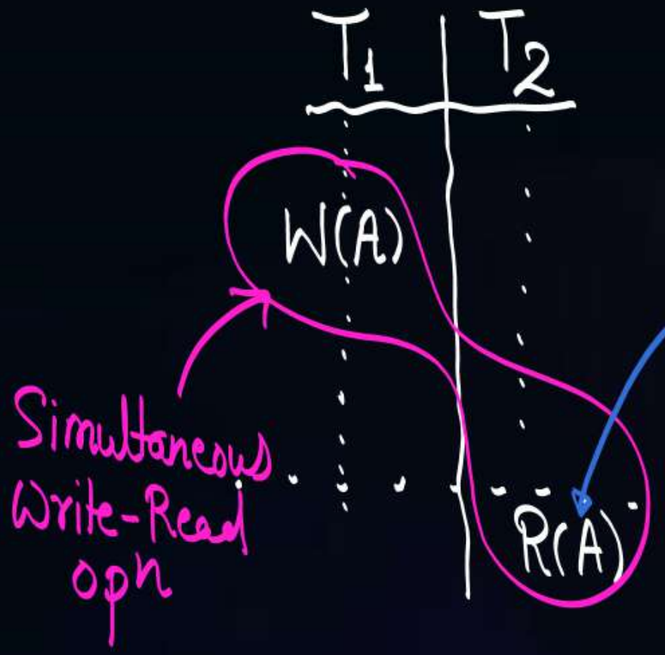
(1) Simultaneous Read-Write opⁿ exist in the schedule

and (2) Schedule is a non-serializable schedule

Topic : WR problem

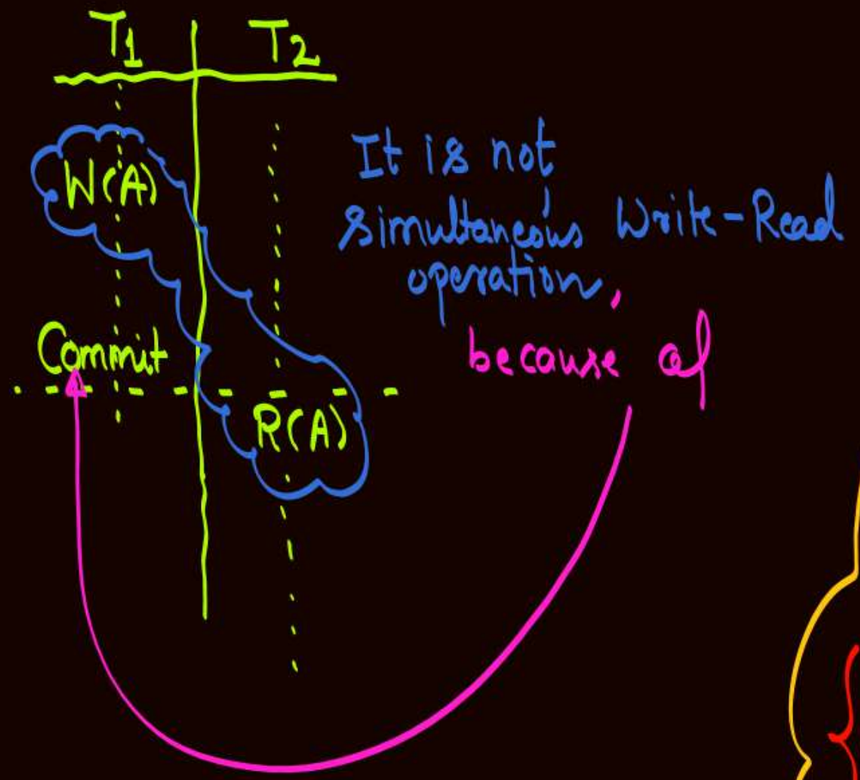
Read-after-write Problem

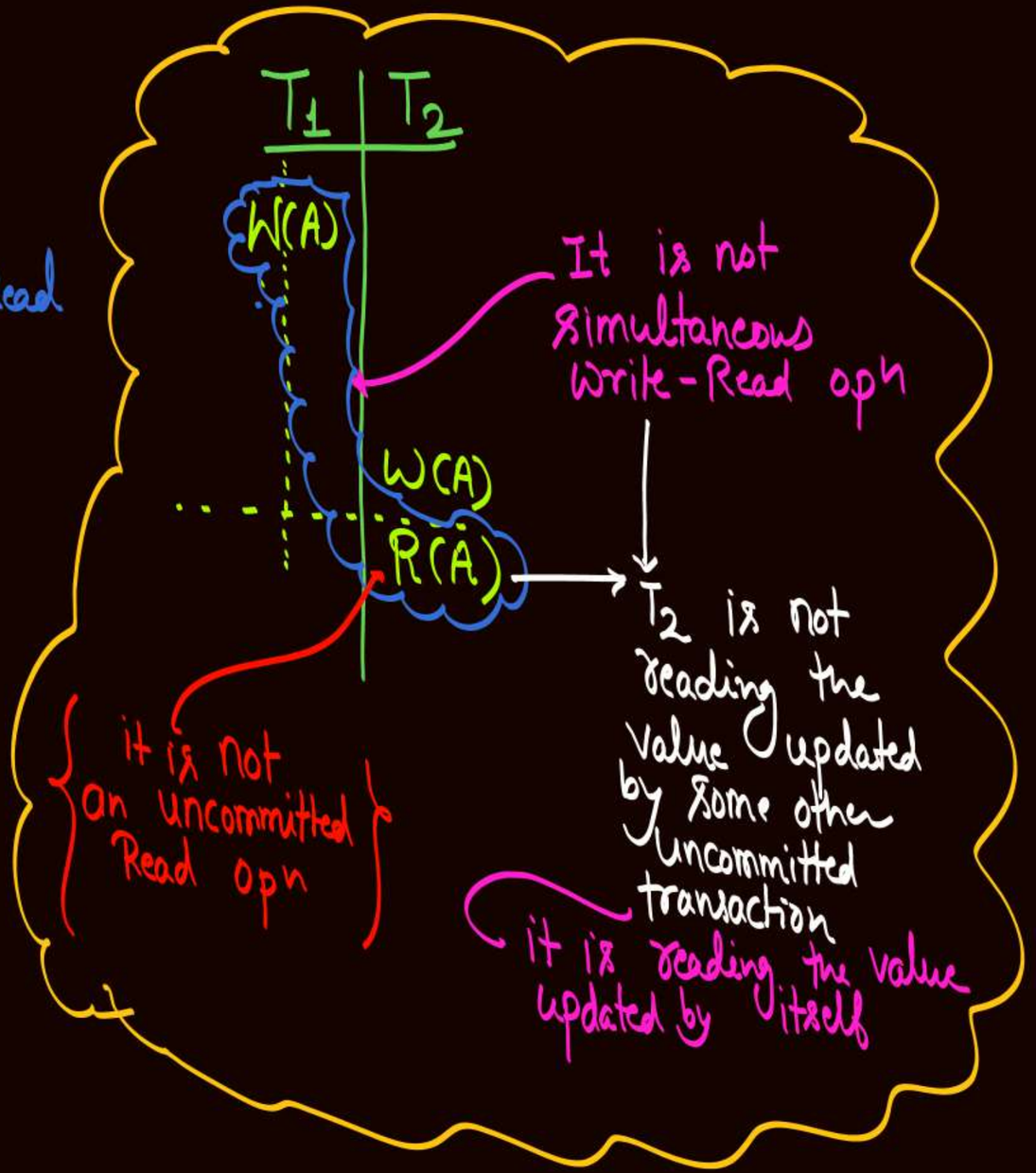
Dirty Read Problem / Uncommitted Read Problem :—

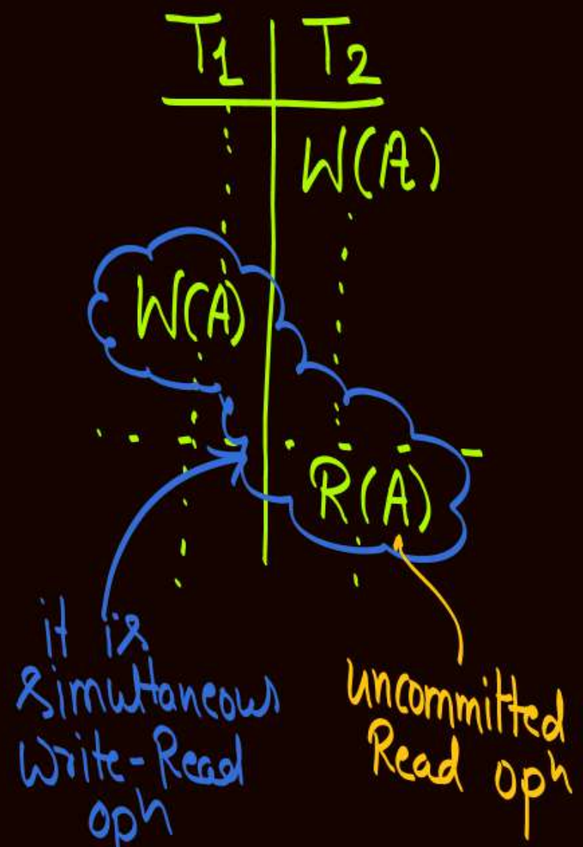


If transaction T_2 reads the valued updated (written) by an uncommitted transaction T_1, then that sequence of operation is called Simultaneous Write-Read opn. {Corresponding Read opn is called Uncommitted Read or dirty read}

Note:- If transaction T_i is working with a value updated by some other transaction T_j, then transaction T_i is dependent on T_j —∴ If transaction T_j rollback, then we also need to rollback T_i.


T1
T2
W(A)
Commit
R(A)
It is not simultaneous Write-Read operation, because of


T1
T2
W(A)
It is not simultaneous Write-Read opn
W(A)
R(A)
T2 is not reading the value updated by some other uncommitted transaction
it is not an uncommitted Read opn
it is reading the value updated by itself

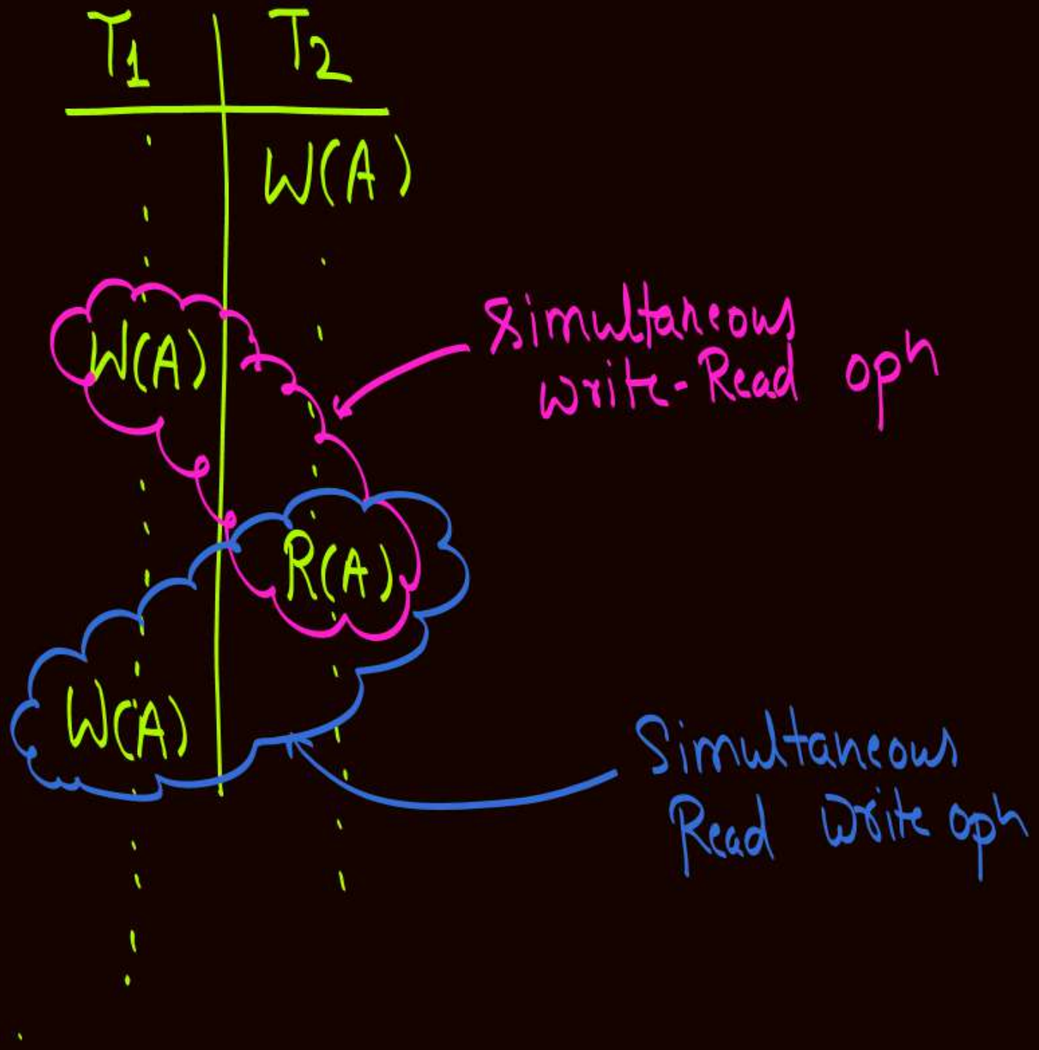

T1
T2
W(A)
W(A)
R(A)
it is simultaneous Write-Read opn
uncommitted Read opn

* WR Problem / Read after Write Problem / Dirty Read Problem :-

A schedule is said to have WR Problem if and only if,

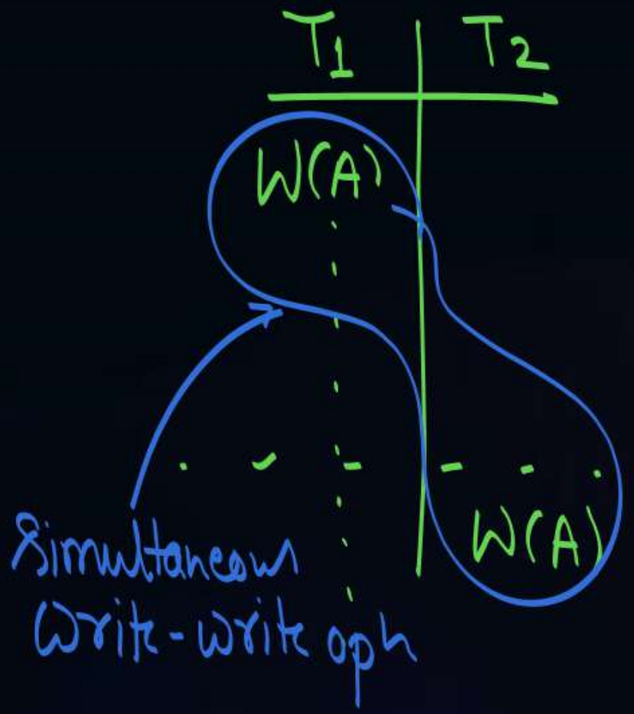
(1) Simultaneous Write-Read opn exist in the schedule
{i.e. uncommitted Read opn must exist}

and (2) Schedule is a non-serializable schedule

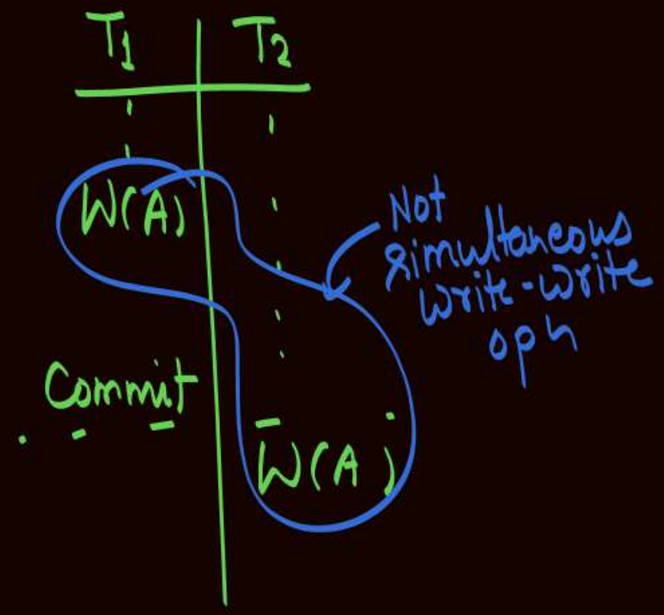

T1
T2
W(A)
W(A)
R(A)
W(A)
Simultaneous Write-Read oph
Simultaneous Read Write oph

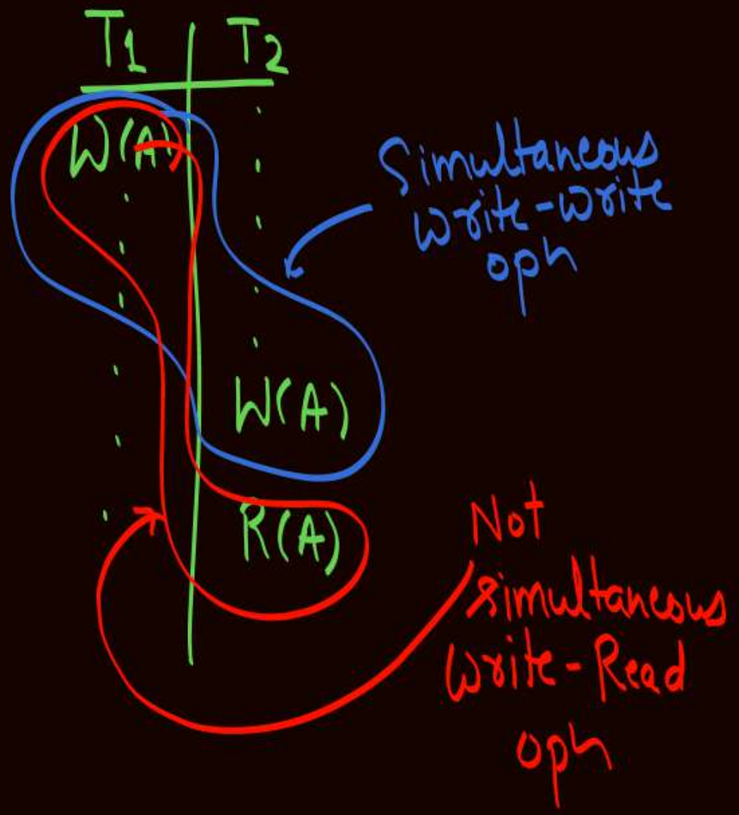
Topic : WW problem

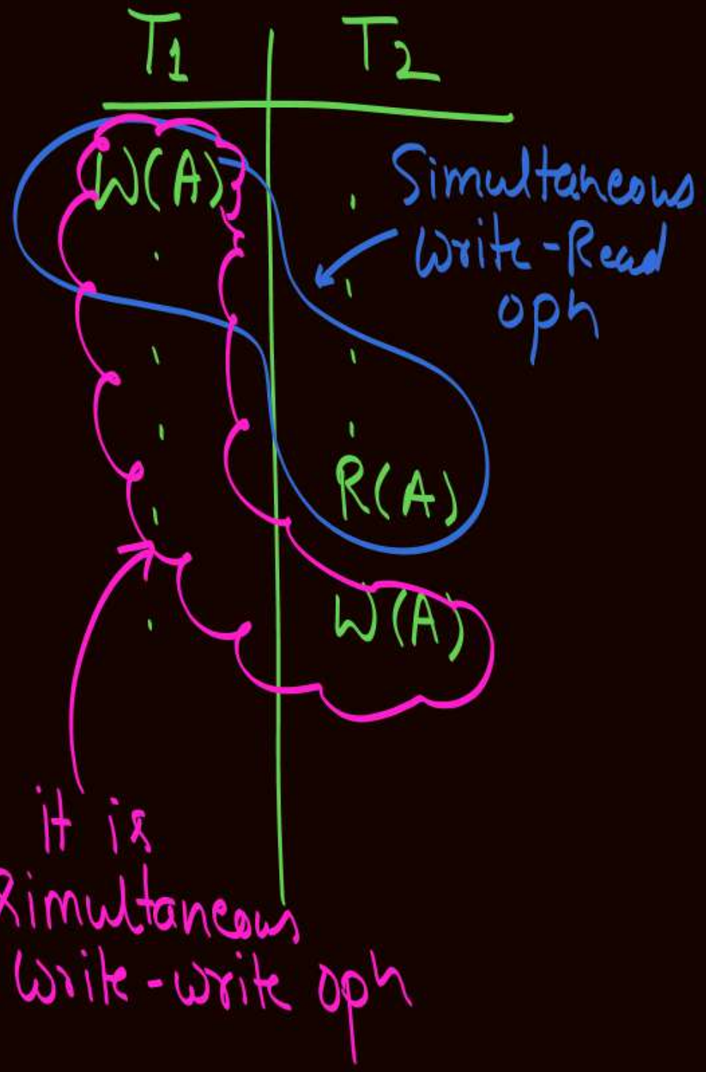
Write after Write Problem :-



If transaction T_2 updates any dataitem 'A' which is already updated by some other uncommitted transaction T_1, then that sequence of operation is called "Simultaneous Write-Write oph".


T1
T2
W(A)
W(A)
Not simultaneous write-write oph
Commit


T1
T2
W(A)
Simultaneous write-write oph
W(A)
R(A)
Not simultaneous write-Read oph

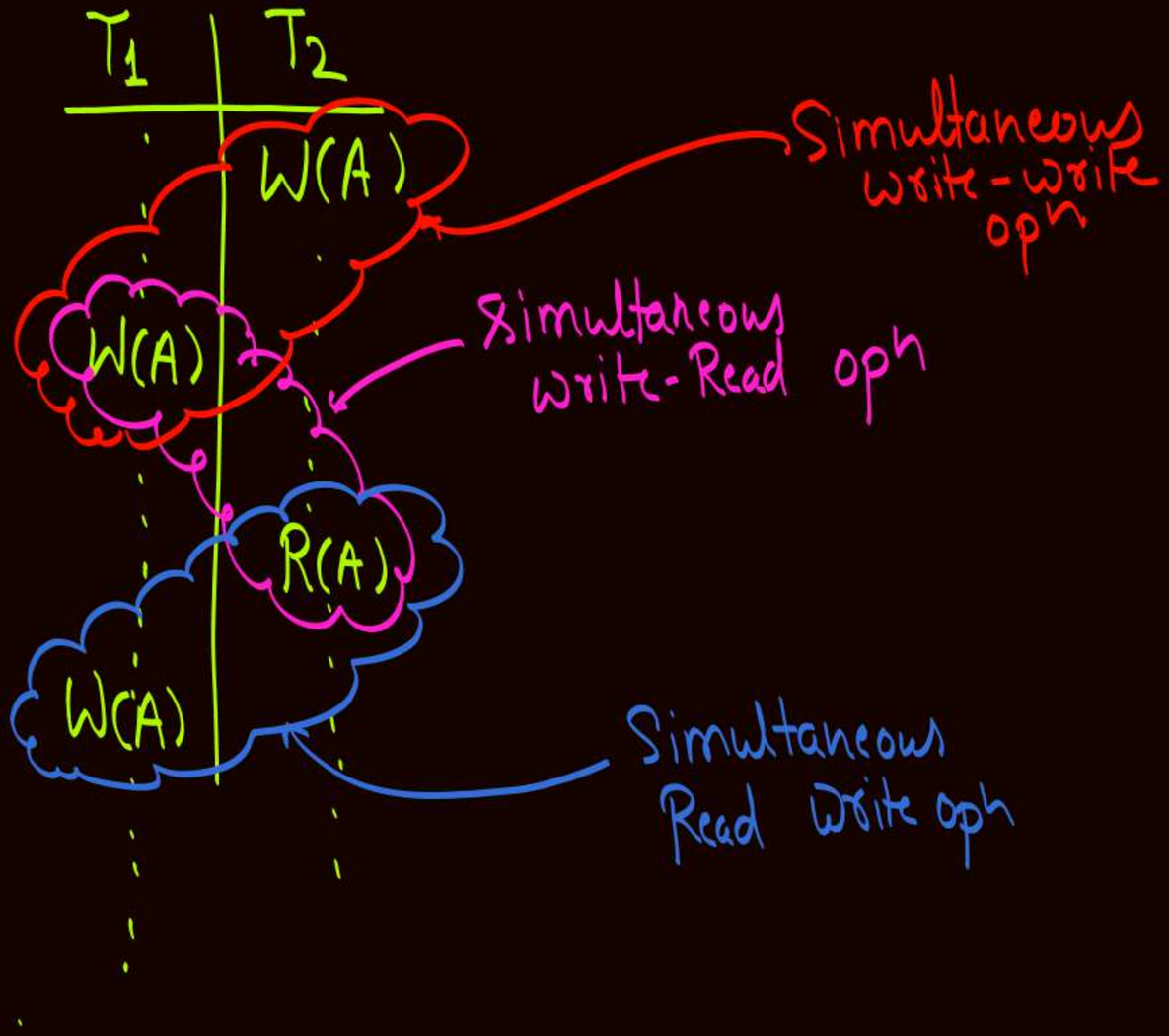

T1
T2
W(A)
Simultaneous write-Read oph
R(A)
W(A)
it is simultaneous write-write oph

WW Problem / Write after Write Problem

A schedule is said to have WW Problem if and only if,

(1) Simultaneous Write-write opn should exist in schedule

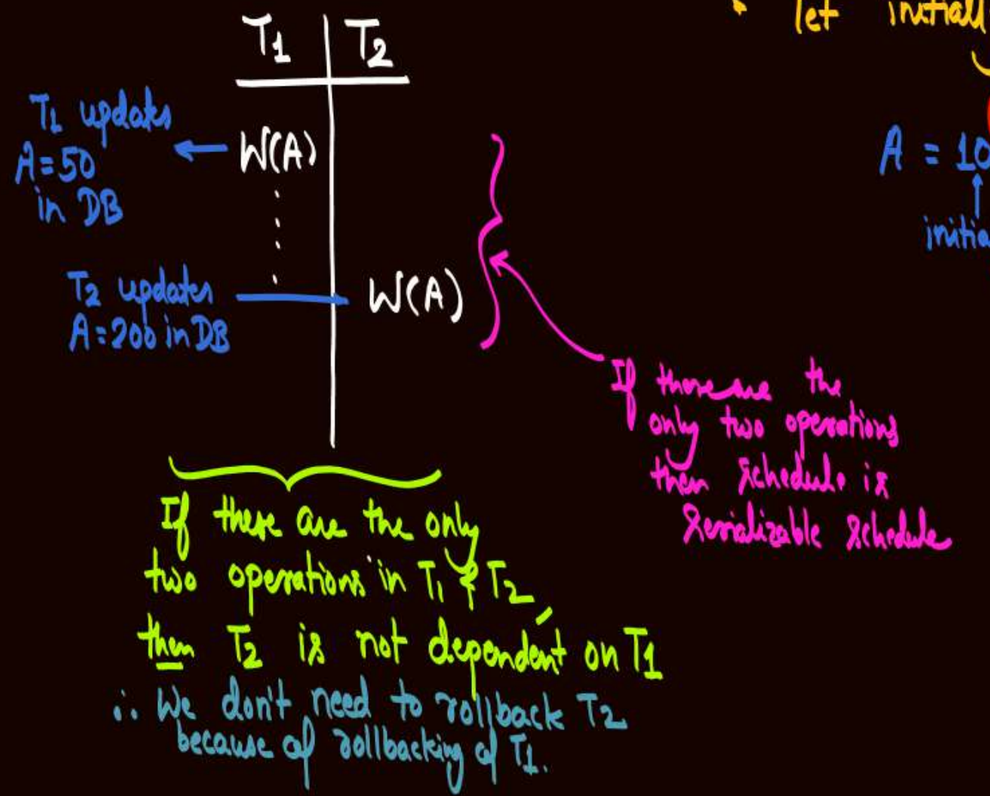
and (2) Schedule is a non-serializable schedule


T1
T2
W(A)
W(A)
R(A)
W(A)
Simultaneous write-write oph
Simultaneous write-Read oph
Simultaneous Read Write oph

Note :- If schedule is a serializable schedule then none RW problem, WR Problem or WW problem is possible but "lost-update" problem is still possible.

Topic : Lost update problem

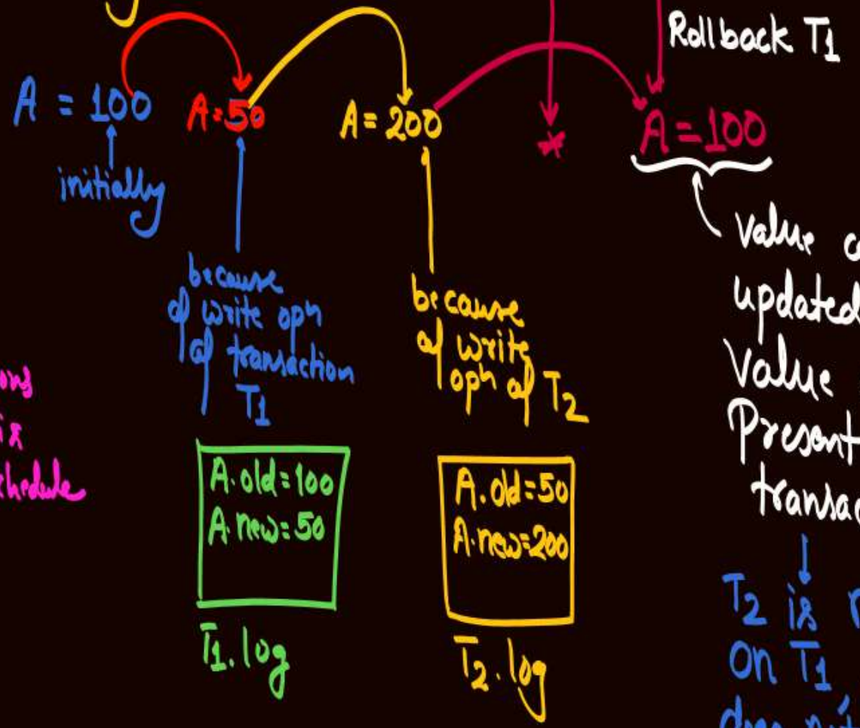
A schedule is said to have "lost-update" problem if and only if simultaneous Write-Write oph exist in the schedule { Schedule may be serializable schedule or non-serializable schedule }



If there are the only two operations in T_1 & T_2, then T_2 is not dependent on T_1
∴ We don't need to rollback T_2 because of rollbacking of T_1.

If these are the only two operations then Schedule is Serializable Schedule

- let initially A = 100 in database



Value of A is updated by old value of A present in transaction log of T_1

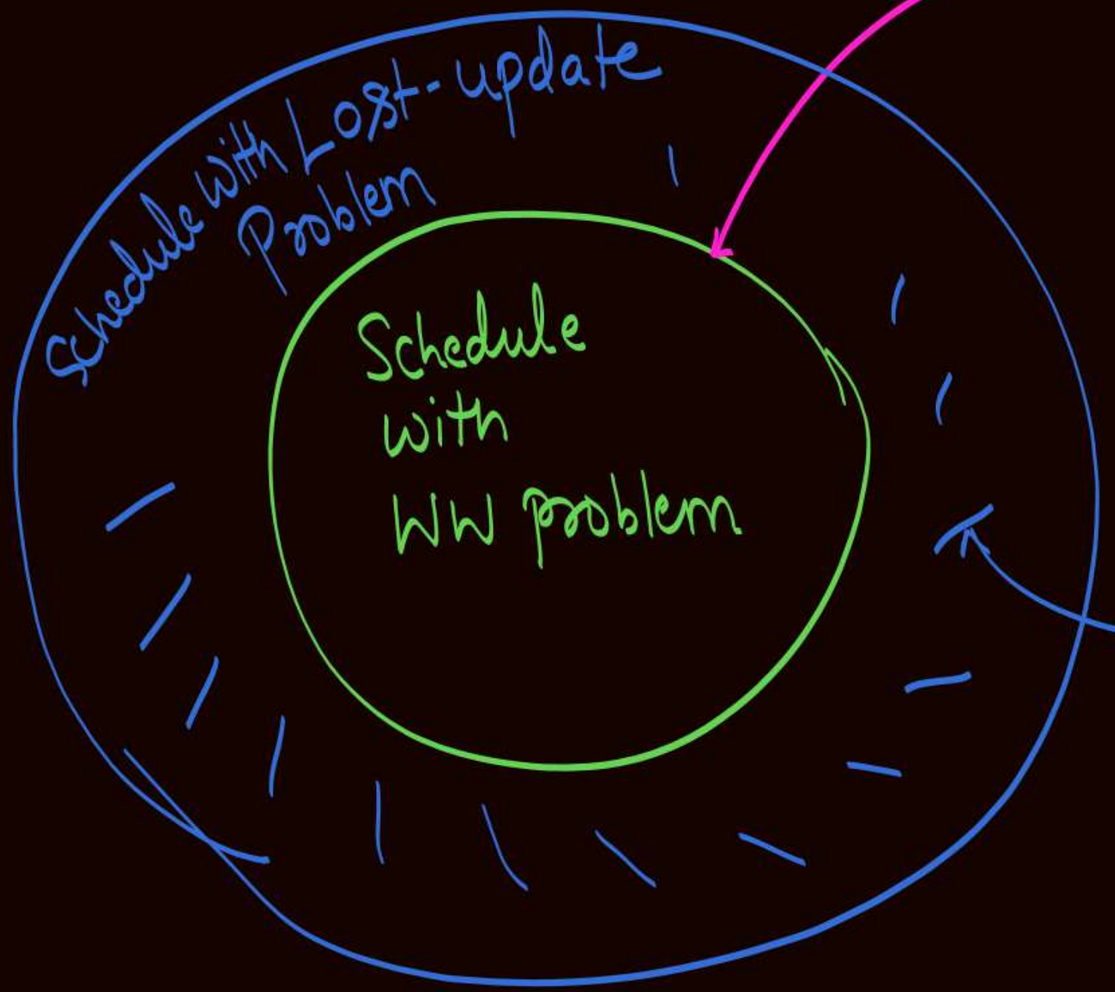
↓

T_2 is not dependent on T_1 ∴ T_2 does not know anything about this rollback

T_2 still think that value of A in database is 200, but value updated by T_2 is "lost"

∴ Lost update Problem

Note:



Simultaneous Write-write oph exist + Non Serializable

Simultaneous write-write oph exist, but Schedules are Serializable

∴ No WW problem

2 mins Summary

- **Topic** Serializable and non-serializable schedule
- **Topic** Problems because of concurrent execution
- **Topic** RW problem
- **Topic** WR problem
- **Topic** WW problem
- **Topic** Lost update problem

THANK - YOU