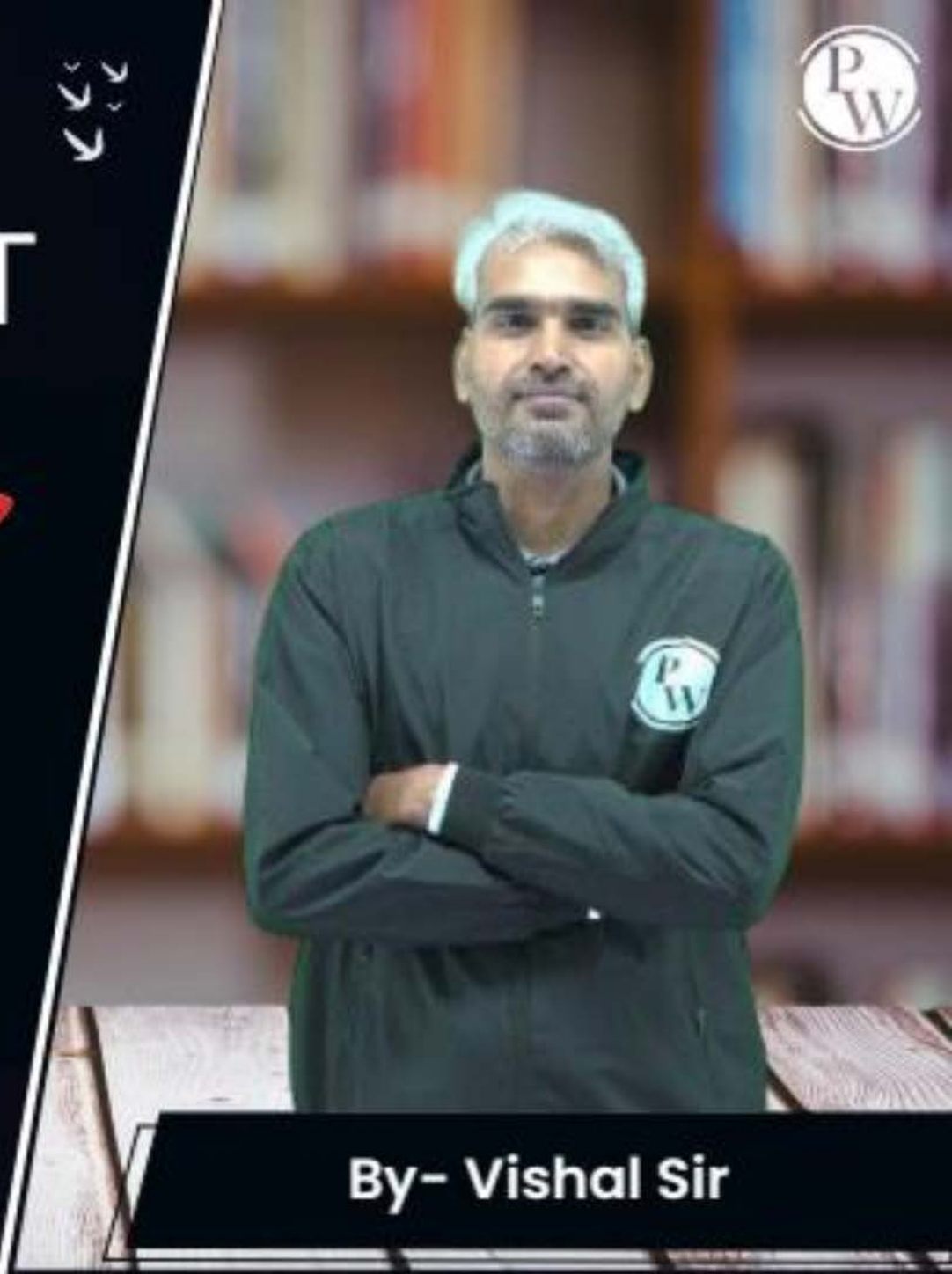
Computer Science & IT

Database Management System

Transaction & concurrency control

Lecture No. 05



By- Vishal Sir

Recap of Previous Lecture

1. Topic — RW problem
2. Topic — WR problem
3. Topic — WW problem

Schedule must be Non-Serializable

4. Topic — Lost update problem

{ Possible in serializable as well as non-serializable schedule }

5. Topic — Classification of schedules

(Irrecoverable & Recoverable Schedule)

Slide

Topics to be Covered

- Topic — Cascading rollback problem
- Topic — Cascadeless rollback recoverable schedule
- Topic — Strict recoverable schedule
- Topic — Conflicting and non-conflicting pairs of operations
- Topic — Conflict serializable schedule

Slide

Topic : Classification of schedule

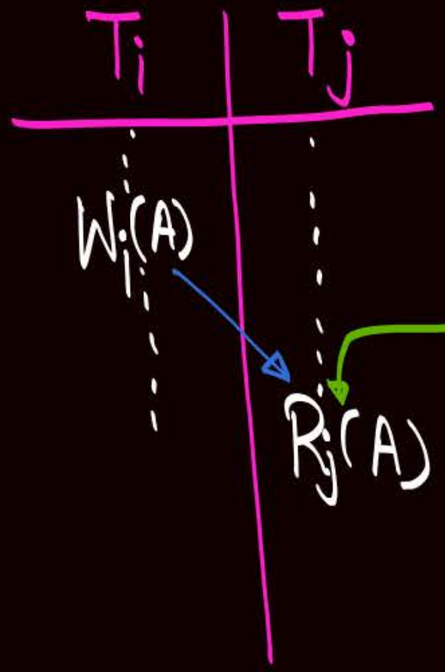
Classification based on Recoverability

1. Irrecoverable schedule
2. Recoverable schedule
3. Cascadeless rollback recoverable schedule
4. Strict recoverable schedule

Classification based on Serializability

1. Conflict serializable schedule
2. View serializable schedule

Note :- If transaction T_j reads the value written by an uncommitted transaction T_i, then transaction T_j is dependent on transaction T_i.



T_j reads the value of dataitem 'A' written by T_i
∴ T_j is dependent on T_i

If transaction T_j is dependent on T_i, and for some reason if we rollback transaction T_i, then we must rollback transaction T_j as well

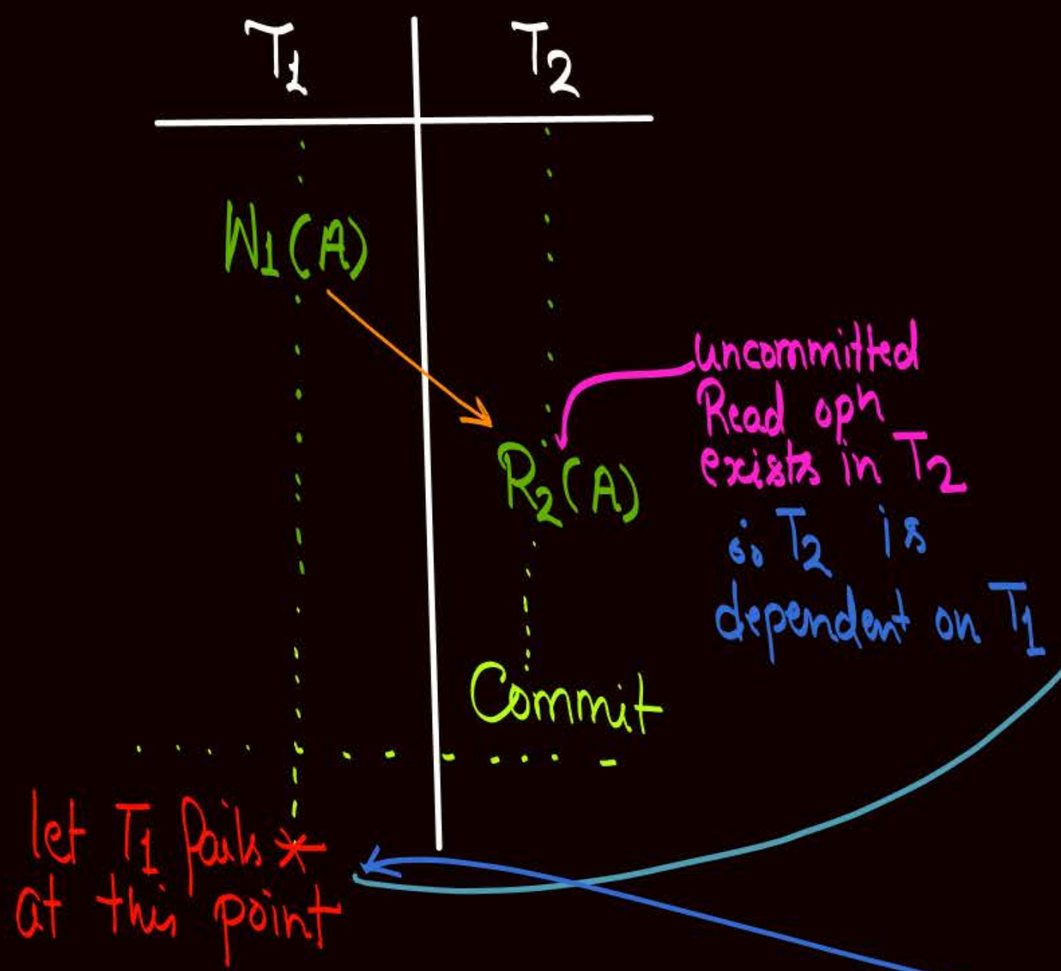
We can rollback the transaction only if it has not executed its Commit op^n successfully

We can not rollback a Committed transaction

Topic : Irrecoverable schedule

- It is not possible to rollback a committed transaction.
- If there arise a situation when we are required to rollback a committed transaction, then it will not be possible to rollback that transaction. And hence we will not be able to recover from that failure, such schedules are called irrecoverable schedule

Irrecoverable Schedule



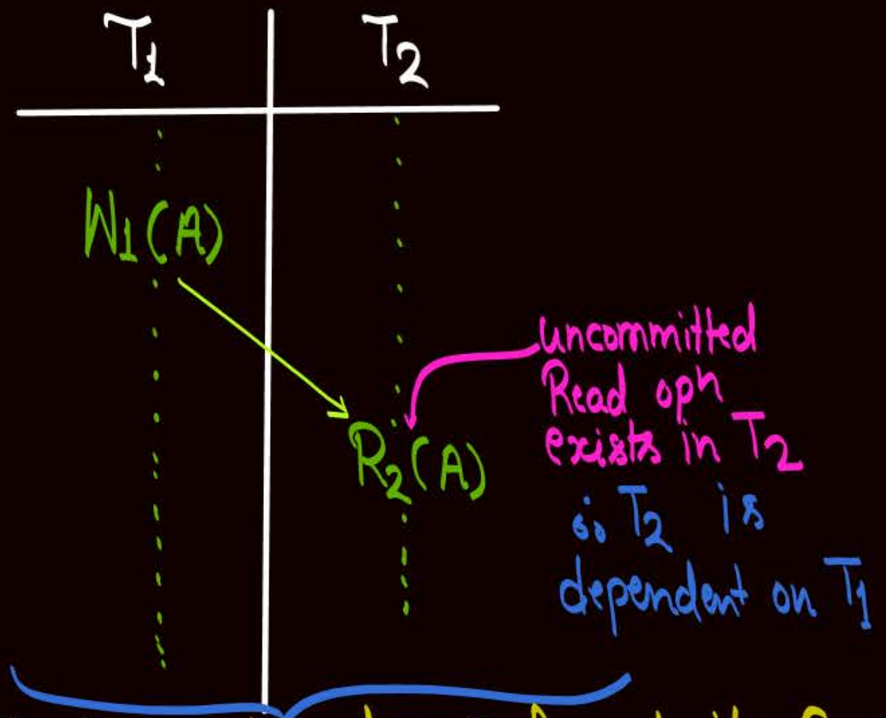
T1 failed, ∴ We will rollback T1.
Because of rollback of T1, we need to rollback all the transactions that are dependent on T1
{ in our eg: T2 is dependent on T1.
∴ We must rollback T2 as well }

But T2 has already Executed its commit oph, ∴ We can not rollback T2

Hence we can not recover from this failure

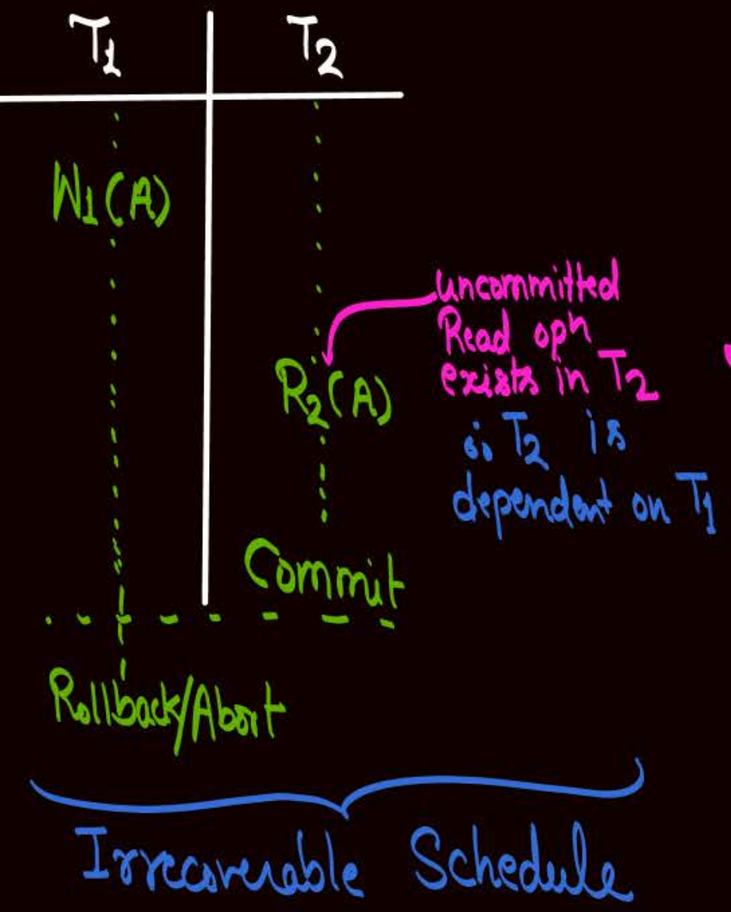
Hence schedule is irrecoverable schedule

Irrecoverable Schedule

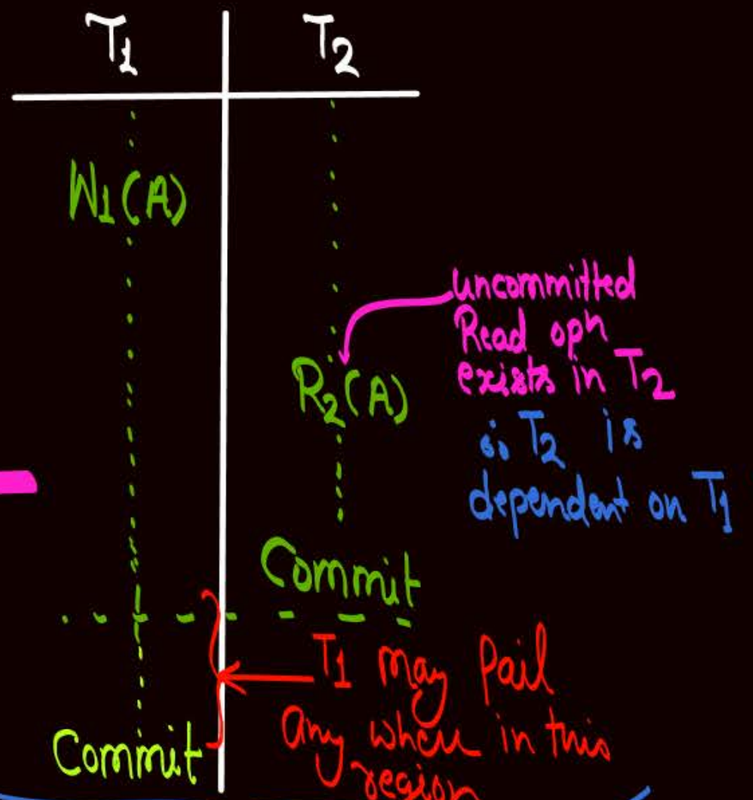


- No transaction has performed its Commit oph
  so If any failure occur, then we can rollback both transaction.
  i.e, we will be able to recover from failure
  Hence, Schedule is recoverable schedule

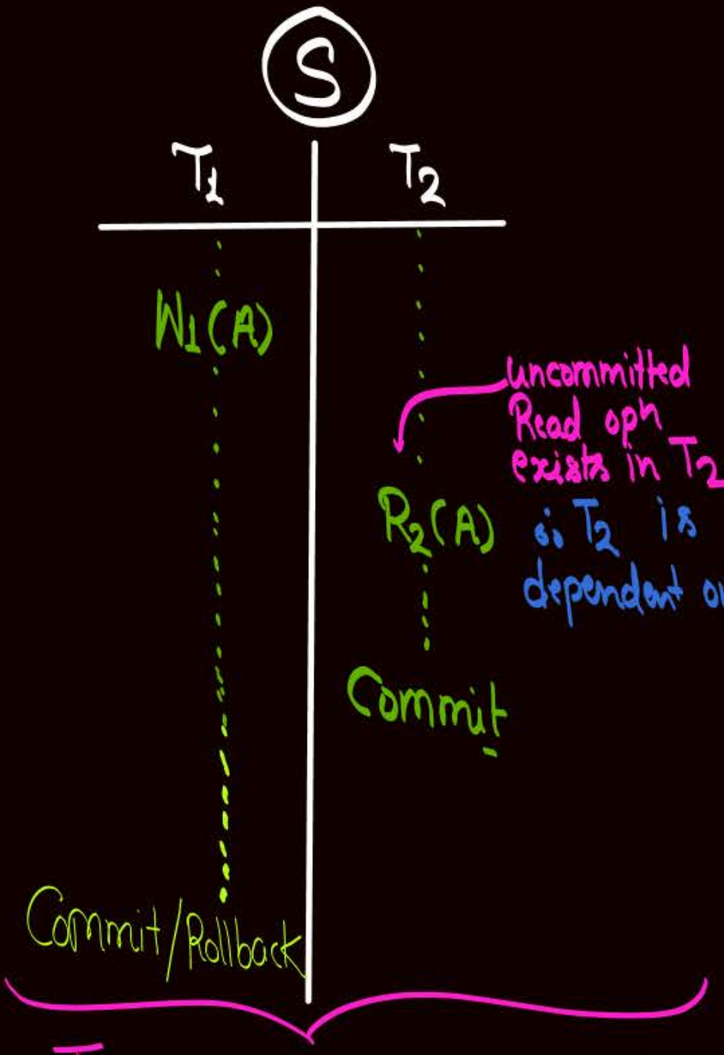
Irrecoverable Schedule



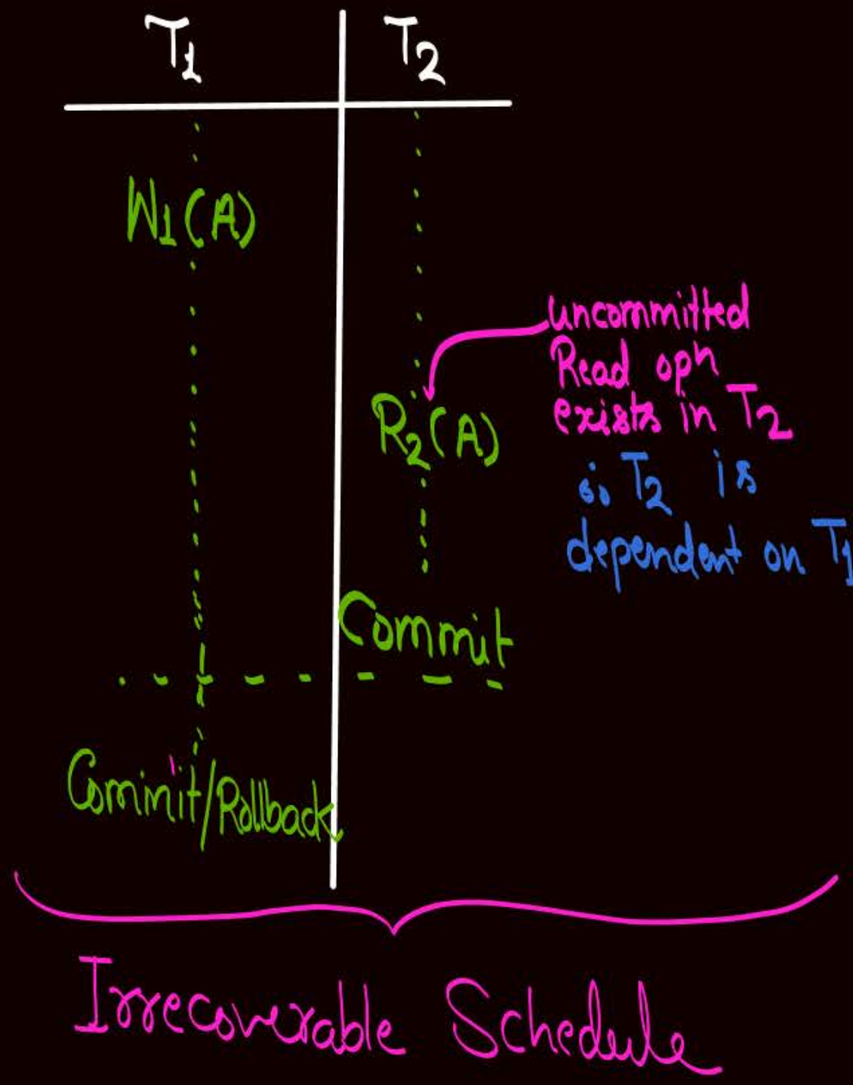
+



⇒



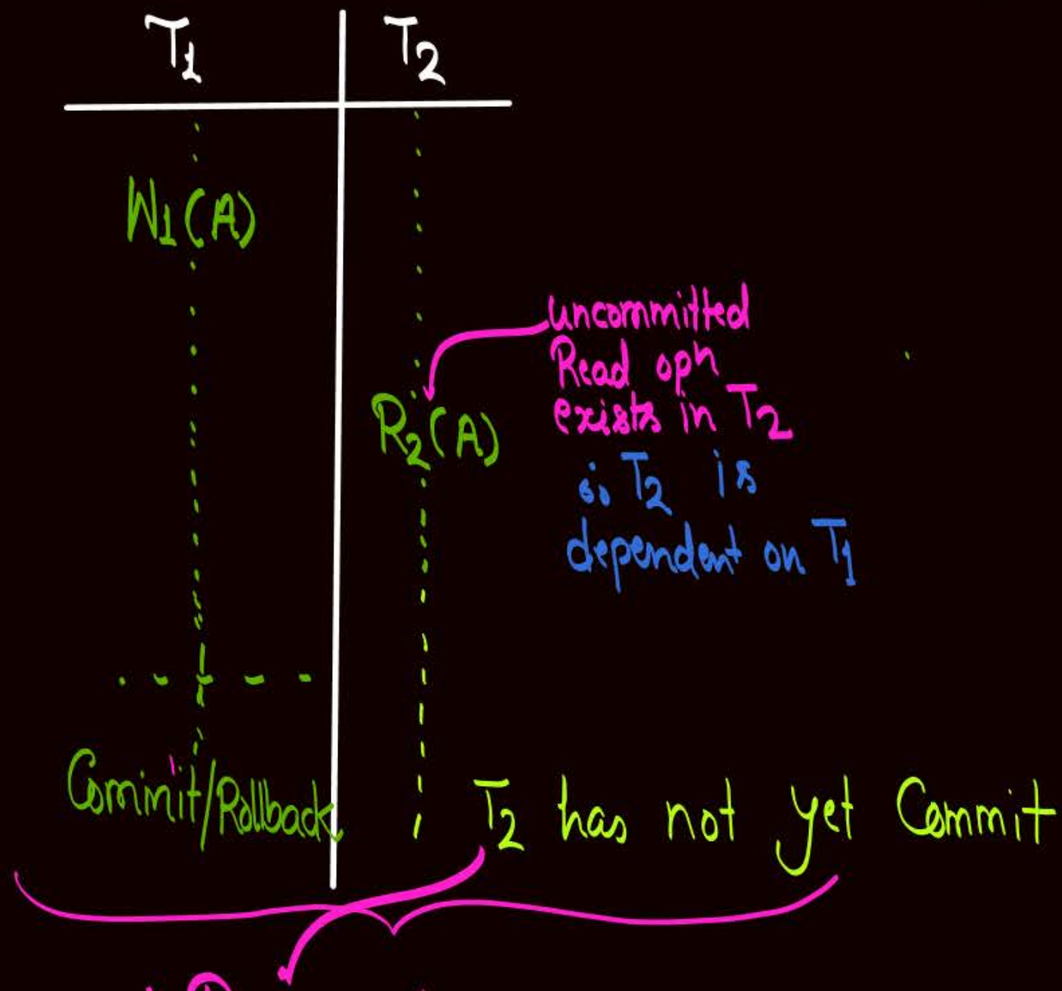
Irrecoverable Schedule



→ Let two transactions T_1 & T_2 exists in a schedule such that T_2 is dependent on T_1,

If transaction T_2 {i.e. dependent transaction} Commit before the commit/Rollback of transaction T_1 {ie, transaction on which T_2 depends}, then such schedule is called "Irrecoverable Schedule"

Irrecoverable Schedule



→ Let two transactions T_1 & T_2 exists in a schedule such that T_2 is dependent on T_1,

If transaction T_2 {i.e. dependent transaction} Commit before the commit/Rollback of transaction T_1 {ie, transaction on which T_2 depends}, then such schedule is called "Irrecoverable Schedule"

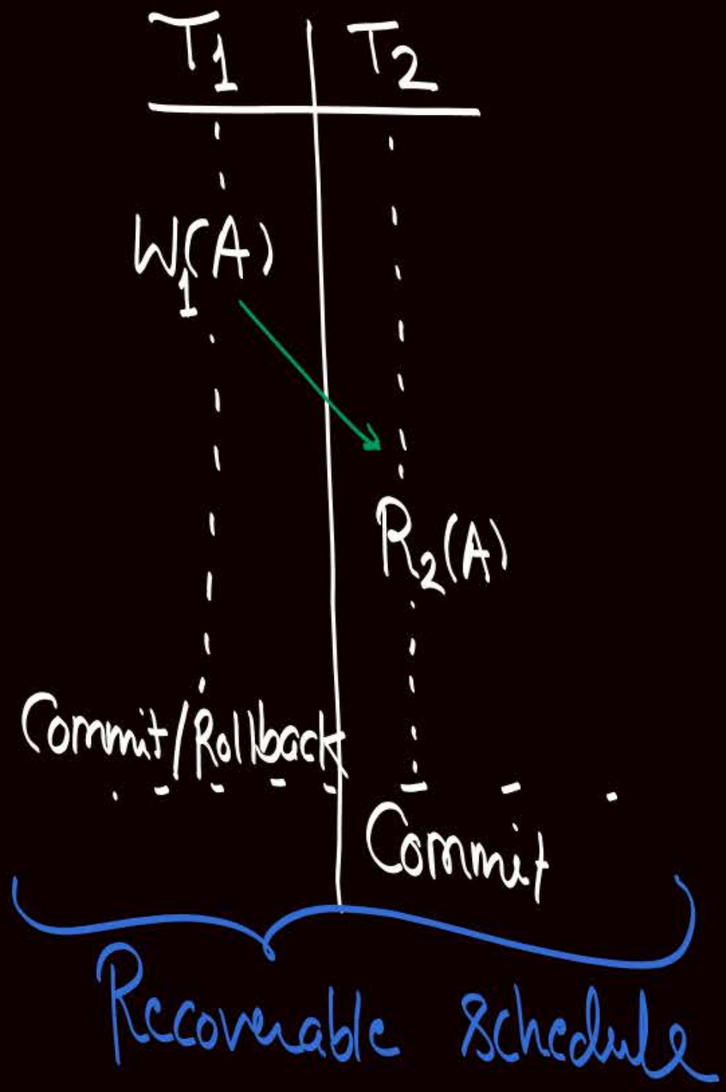
Irrecoverable Schedule

Uncommitted read oph are necessary (but not sufficient) for a schedule to be irrecoverable Schedule.

i.e, ① If uncommitted read oph does not exist in a schedule then that schedule is always recoverable

and ② If uncommitted read oph exists in a schedule then that schedule may or may not be an irrecoverable schedule.

Recoverable Schedule



For a schedule to be called recoverable schedule,

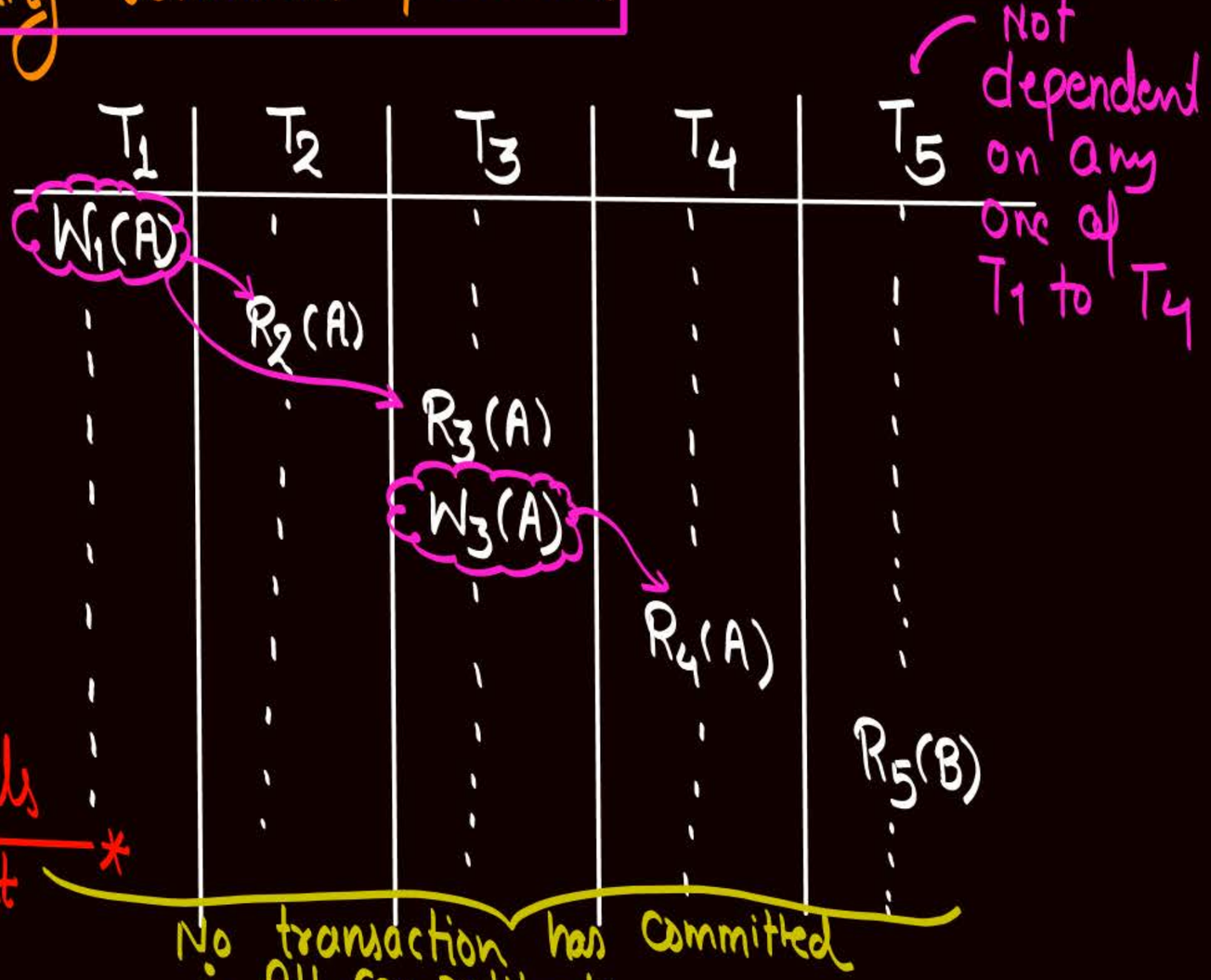
Either uncommitted read oph should not exist in the schedule

Or if any uncommitted read oph exist, then Commit oph of dependent transaction must appear after the Commit/Rollback of the transaction on which it depends

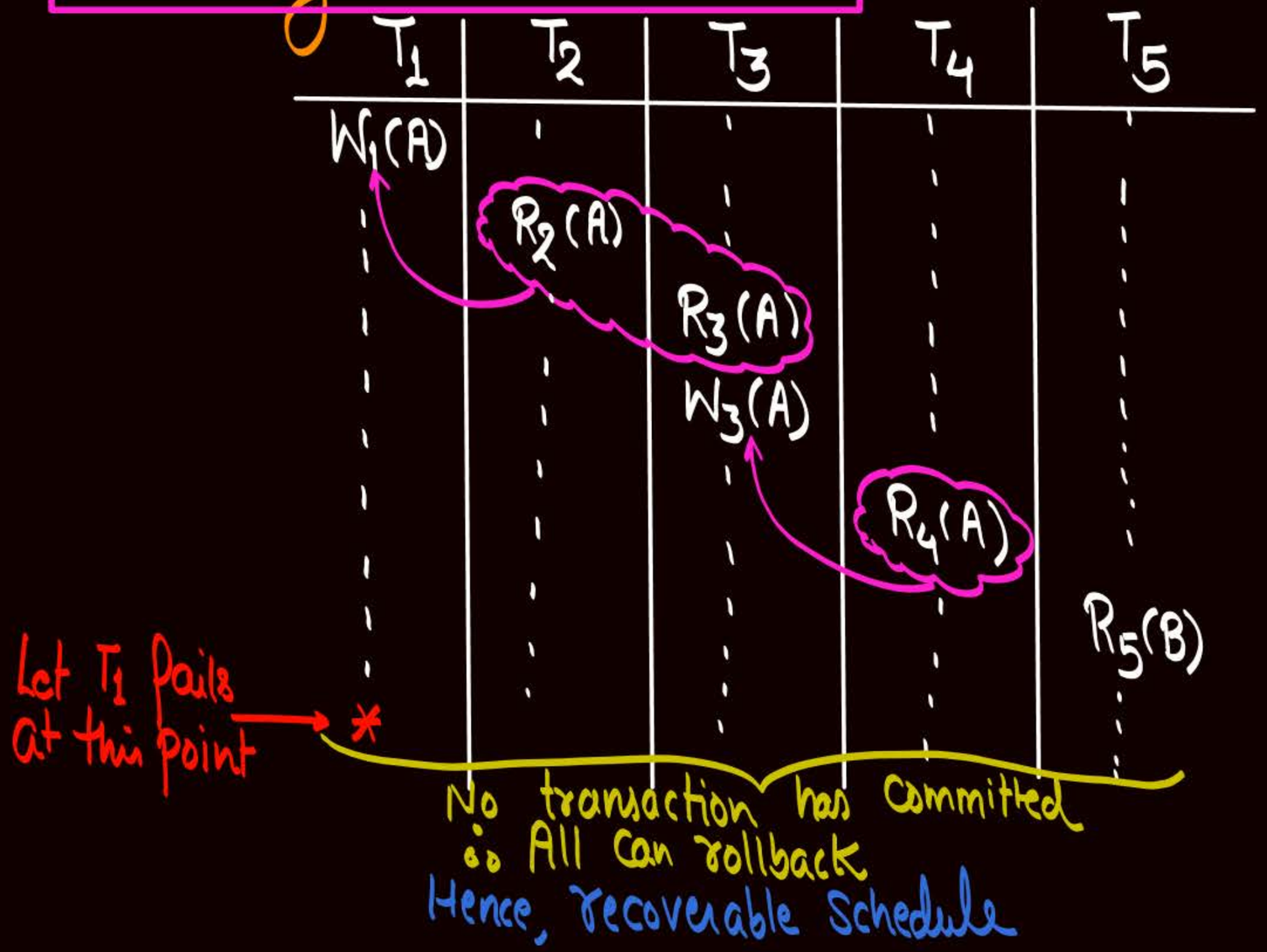
Topic : Cascading rollback problem

- If failure of one transaction results in rollback of a set of other transactions along with failed transaction, then it is called Cascading rollback. And we say that the schedule suffer from Cascading rollback problem.

Cascading rollback problem



Cascading rollback problem



* If T_1 fails, we rollback T_1.
- Because of rollback of T_1 we need to rollback transactions dependent on T_1, i.e, T_2 & T_3
* No other transaction depends on T_2 but T_4 depends on T_3.
∴ Because of rollback of T_3 we will rollback T_4 as well.
- No other transaction depends on T_4

Because of failure of T_1, we need to rollback T_2, T_3 & T_4 as well along with T_1 ∴ Cascading rollback Problem exist.

* Because of Cascading rollback, Wastage of CPU time increases.

Topic : Cascadeless recoverable schedule

- If uncommitted read opn exist in the schedule then dependency will exist among the transactions, and if dependency exists among the transaction then because of failure of one transaction we may have to rollback some other transactions as well, i.e., Cascading rollback Problem will exist.

- For Cascadeless recoverable schedule, uncommitted read operation must not be allowed

Topic : Cascadeless recoverable schedule

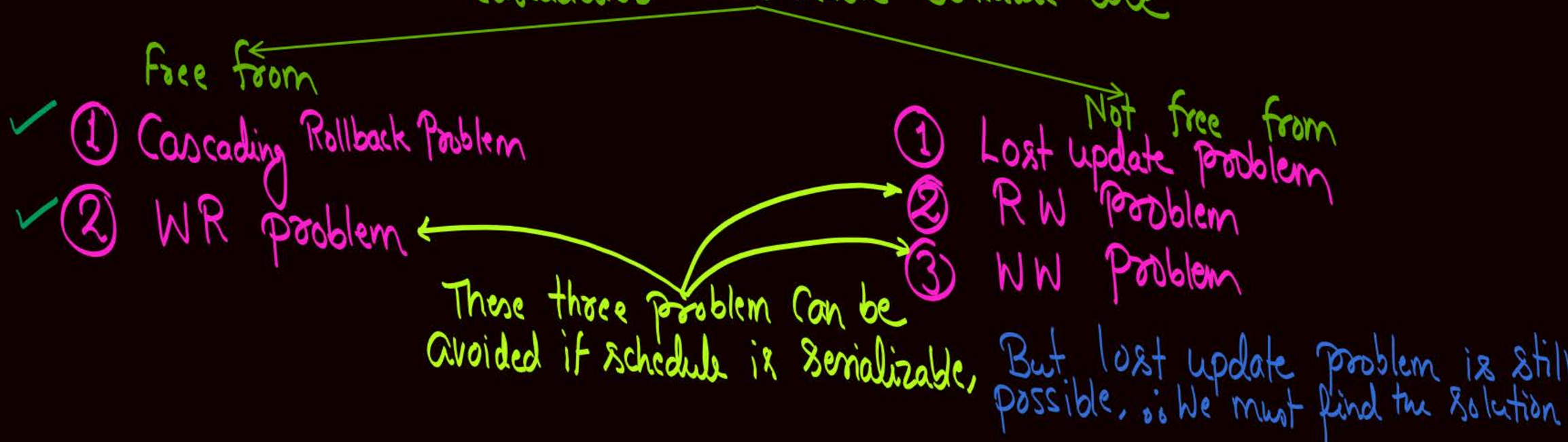
T_1	T_2
$W_1(A)$	
Commit/Rollback	
	$R_2(A)$

- For cascadeless recoverable schedule uncommitted read op^n must not be allowed.
- For cascadeless recoverable schedule, If transaction T_1 writes (update) the data item 'A', then no other transaction should be allowed to read the value of dataitem 'A' written by transaction T_1, until the Commit / Rollback of transaction T_1.

Note:- Uncommitted read opⁿ are possible in Serializable as well as non-serializable schedule.

i.e; Cascading rollback problem is possible in serializable schedule as well.

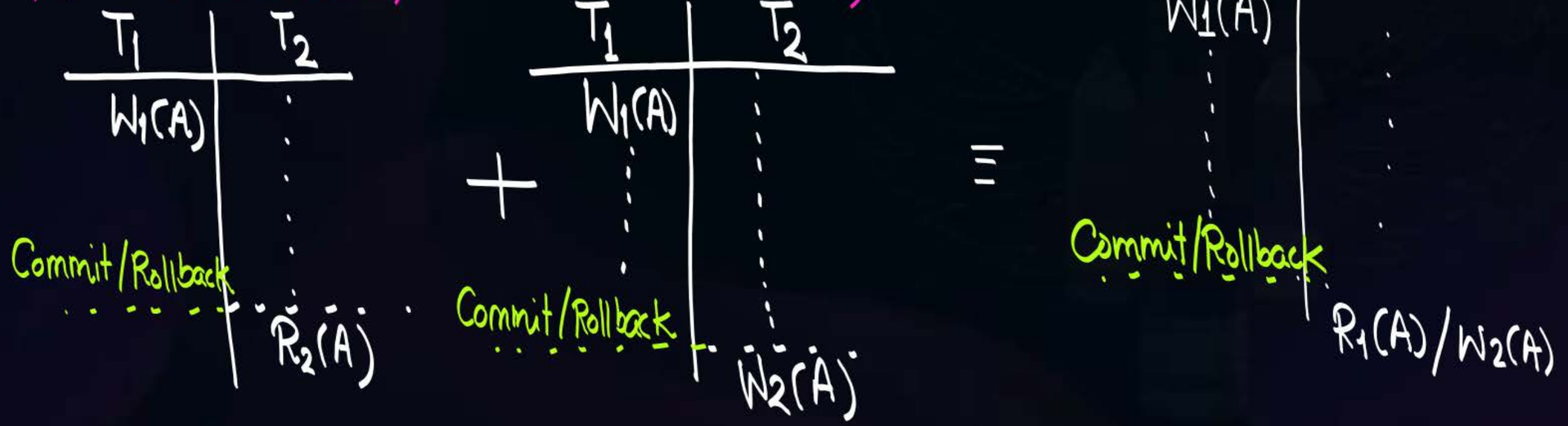
Hence we had to find the solution for Cascading rollback problem. (i.e. Cascadeless Recoverable schedule)



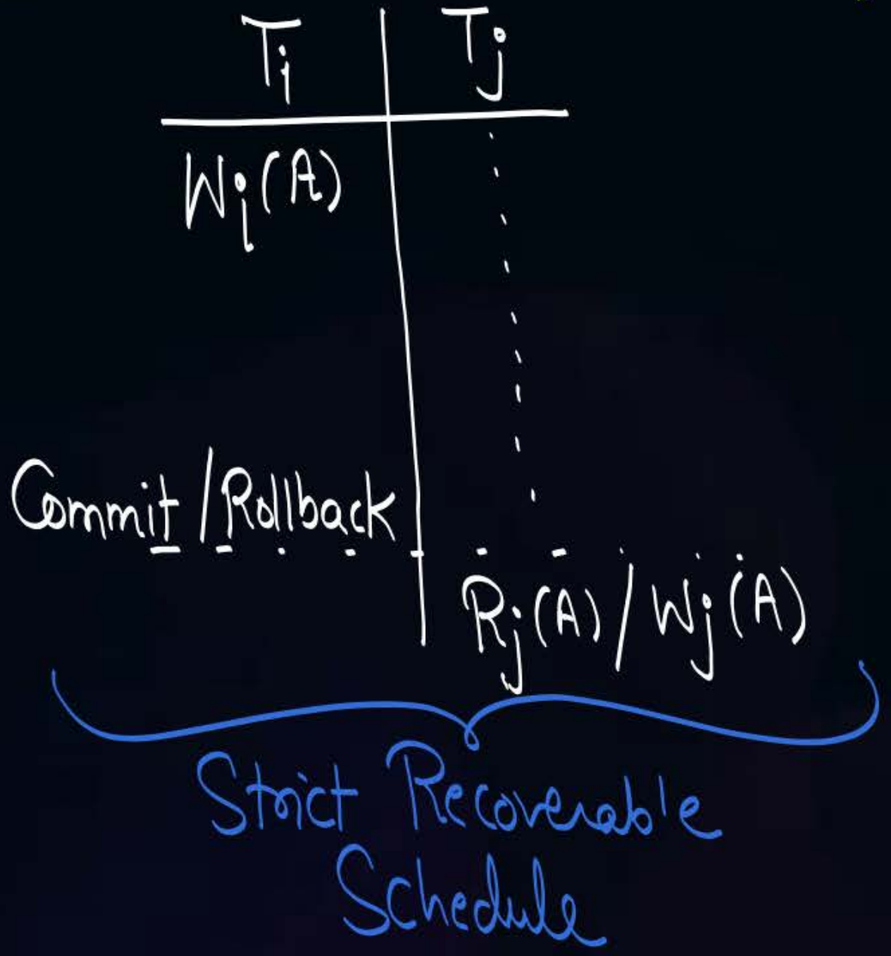
Topic : Strict recoverable schedule

* Strict recoverable schedule is Cascadeless recoverable schedule as well as it is free from lost-update problem

(Cascadeless Recoverable Schedule (i.e, No uncommitted Read) + No lost-update Problem (i.e, No simultaneous write-write opn)) = Strict Recoverable Schedule



Topic : Strict recoverable schedule



For a schedule to be called strict recoverable, if transaction Ti updates any dataitem 'A', then no other transaction Tj is allowed to read or write the dataitem 'A' until the commit or Rollback of transaction Ti

Criteria for Consistency :-

1. Schedule must be Strict recoverable
2. Schedule must be Serializable

These two can solve all problems

Strict Recoverable Schedule

Ans.

Free from

1. Cascading Rollback Problem
2. Lost-update problem
3. WR problem
4. WW Problem

Not free from

1. RW problem

Don't worry about this, it will not exist if schedule is Serializable

Topic : Classification of schedule

Classification based on Recoverability

1. Irrecoverable schedule
2. Recoverable schedule
3. Cascadeless rollback recoverable schedule
4. Strict recoverable schedule

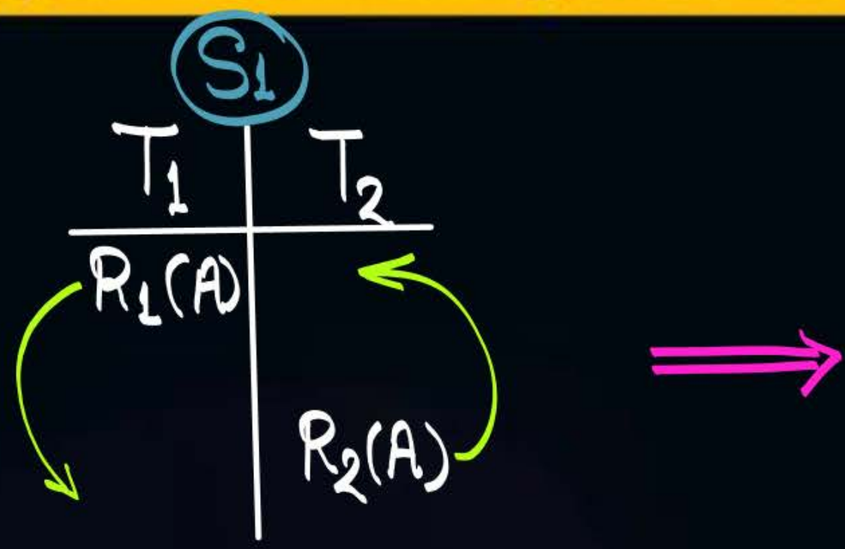
Classification based on Serializability

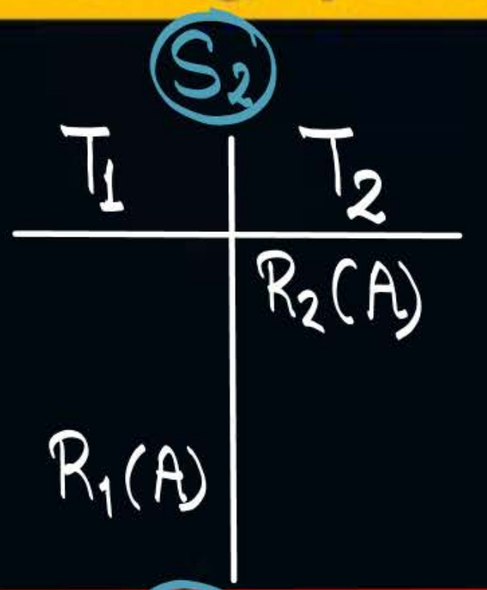
1. Conflict serializable schedule
2. View serializable schedule

Conflict Serializable Schedule

Topic : Conflicting and non-conflicting operations

1

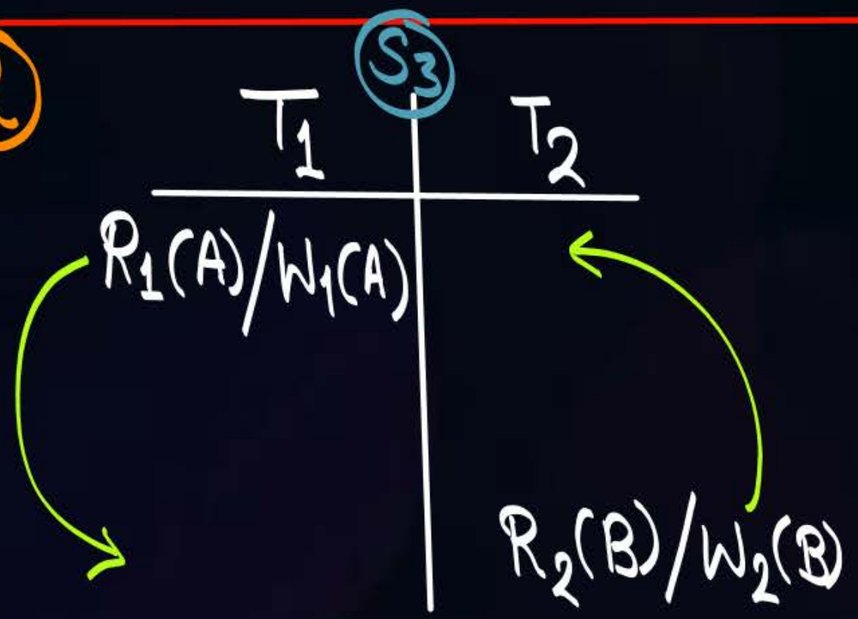


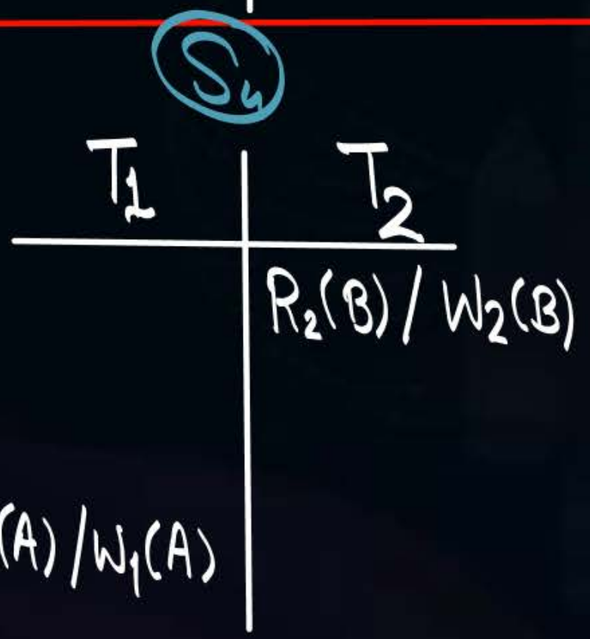


$S_1 \equiv S_2$

∴ $R_i(A)$ & $R_j(A)$ are non-conflicting operation

2





$S_3 \equiv S_4$

∴ $R_i(A)/W_i(A)$ is non-conflicting with $R_j(B)/W_j(B)$

Topic : Conflicting and non-conflicting operations

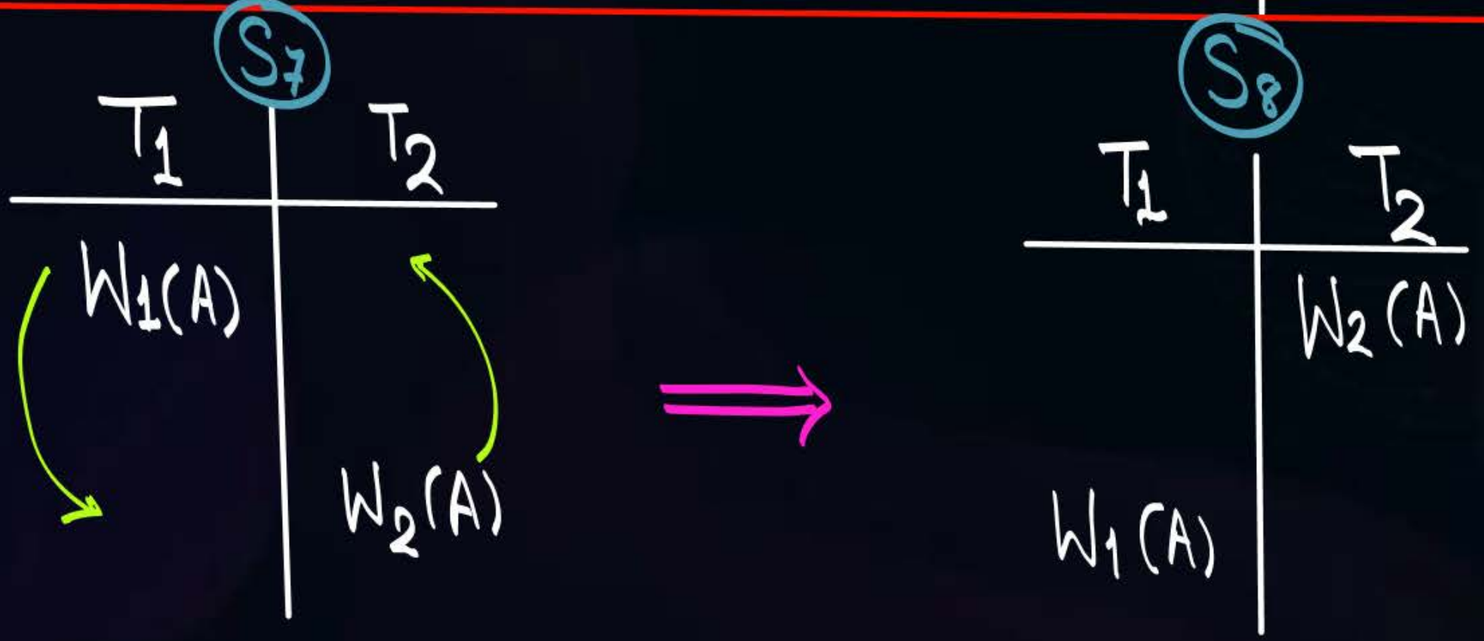
③

$S_5 \not\equiv S_6$

$R_i(A)$ & $W_j(A)$ are Conflicting oph

Similarly $W_i(A)$ & $R_j(A)$ are also Conflicting oph

④



$S_7 \not\equiv S_8$

$\therefore$ $W_i(A)$ & $W_j(A)$ are Conflicting oph

Topic : Conflicting and non-conflicting operations

Conflicting Operations : Two operations are called as conflicting operations if all the following conditions hold true for them -

- Both the operations belong to different transactions
- Both the operations are on the same data item
- At least one of the two operations is a write operation

We can never Swap the Position of two operations with in Same transaction

Topic : Conflicting and non-conflicting operations

- Non-Conflicting Operations : Two operations are non-conflicting if and operations only if
  - Both the operations are on different data items

    or

  - Both the operations are read operations

Topic : Conflict equivalent schedule

Consider two schedules Si and Sj,

If one can be transformed into another by swapping the position of some of the non-conflicting operations in it, then schedule Si and Sj are called conflict equivalent schedule.

Note:- If two schedules are Conflict Equivalent then they are Equivalent schedules, but if two schedules are Equivalent then they need not be Conflict Equivalent

eg.

S_1

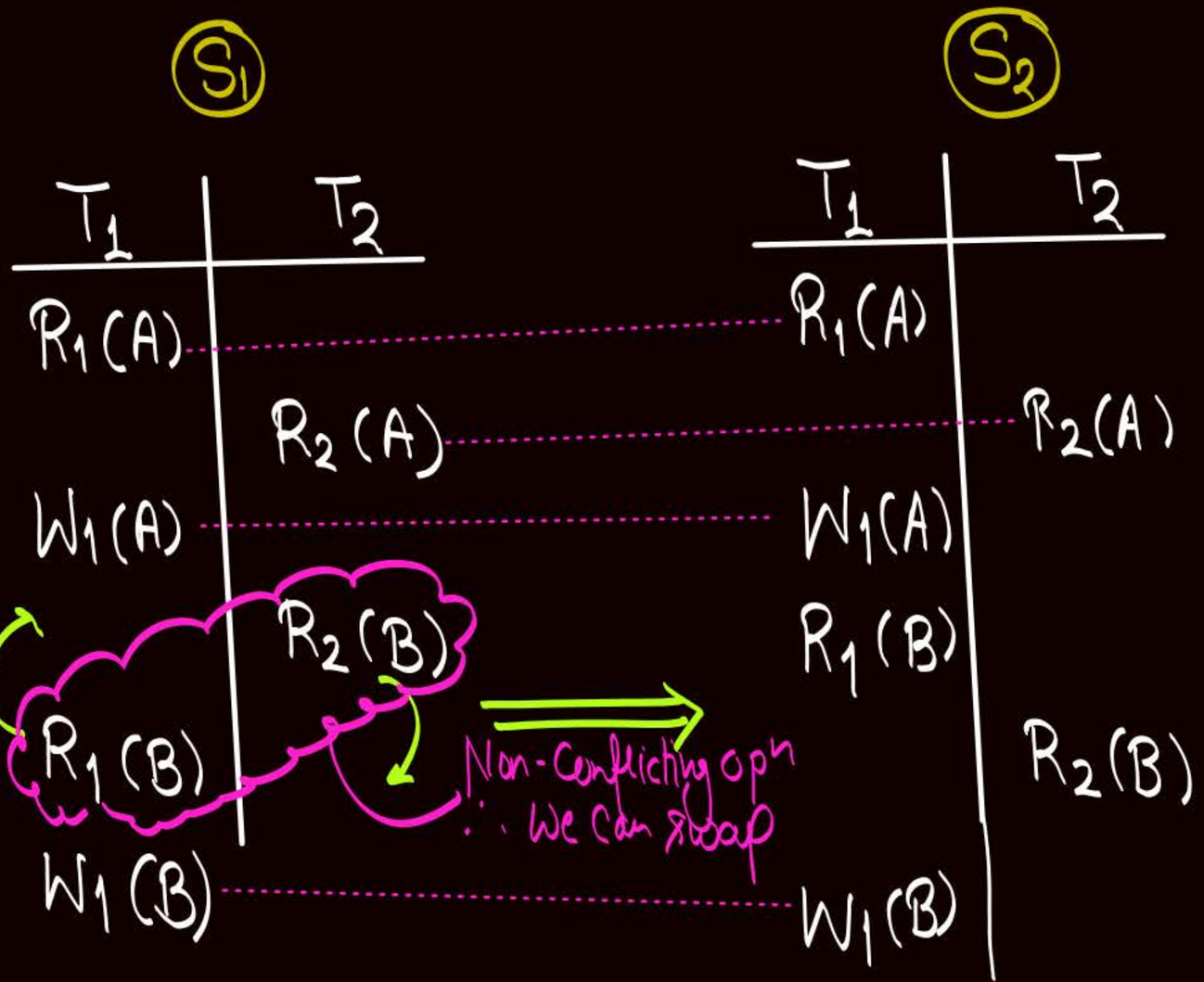
T_1	T_2
	$W_2(A)$
$W_1(A)$	
	$W_2(A)$

S_2

T_1	T_2
$W_1(A)$	
	$W_2(A)$
	$W_2(A)$

In the example $S_1 \equiv S_2$ but none can be transformed into another by swapping the position of non-conflicting opn only.
$\therefore S_1$ is not Conflict Equivalent to S_2

eg: Consider two Schedules S_1 & S_2



- S_1 can be converted into S_2, by swapping the position of some of the non-conflicting operations.

∴ S_1 is Conflict equivalent to S_2

eg: Consider two Schedules S_1 & S_2

S_1

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

S_2

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

Conflicting op^n
∴ We can not swap

Neither S_1 can be converted into S_2, nor S_2 can be converted into S_1 by swapping the position of non-conflicting operations only.

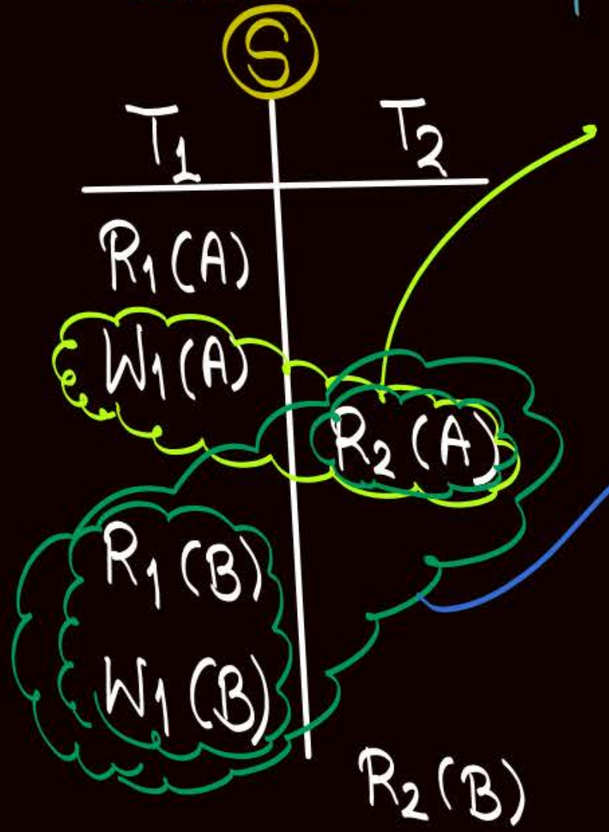
∴ S_1 & S_2 are not Conflict Equivalent

Topic : Conflict serializable schedule

- If a given schedule can be converted into a serial schedule by swapping the positions of some of its non-conflicting operations, then it is called a conflict serializable schedule.

* If given schedule is conflict equivalent to at least one of the serial schedule, then given schedule is conflict serializable schedule.

eg: Consider the following schedule 'S'



Check, whether the schedule 'S' is Conflict Serializable schedule or not?

Conflicting ∴ $R_2(A)$ can not be moved above $W_1(A)$,

Hence S is not Conflict Equivalent to serial schedule $T_2 \rightarrow T_1$

$R_2(A)$ is non-conflicting with $R_1(B)$ as well as $W_1(B)$

∴ it can be moved down

and schedule will become serial schedule $T_1 \rightarrow T_2$

i.e. Schedule S is Conflict Equivalent to at least one serial schedule

Hence 'S' is a Conflict Serializable Schedule and Conflict Equivalent serial schedule is $T_1 \rightarrow T_2$

Two serial schedules are possible over the transactions of schedule S

i.e:

$T_1 \rightarrow T_2$ =

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

&

$T_2 \rightarrow T_1$ =

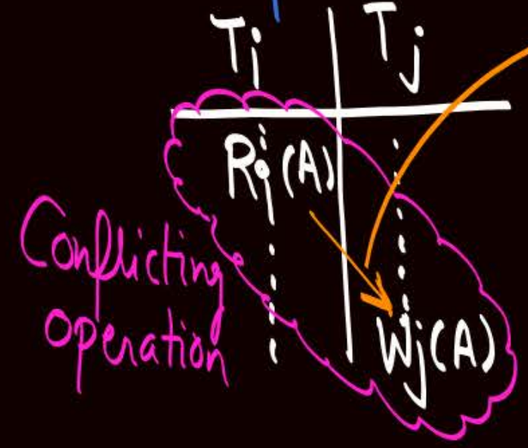
T_1	T_2
	$R_2(A)$
	$R_2(B)$
$R_1(A)$	
$W_1(A)$	
$R_1(B)$	
$W_1(B)$	

Note

① If schedule S is a Conflict Serializable schedule, then it is guaranteed that schedule S is a Serializable schedule

② If schedule S is not a Conflict serializable schedule, then schedule S may or may not be a serializable schedule

③ Conflicting pairs of operations will define the precedence order in which transactions must execute in an equivalent serial schedule



∴ If there is any serial schedule which is Conflict Equivalent to given schedule, then in that serial schedule Ti must execute before Tj

Topic : Precedence graph

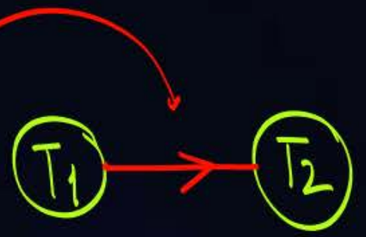
Precedence graph $G = (V, E)$ is a directed graph.

Where Set of vertices = V = Set of all transactions of the schedule

Set of Edges = E = Set of directed edges based on Precedence order defined by pairs of Conflicting operations

S_1

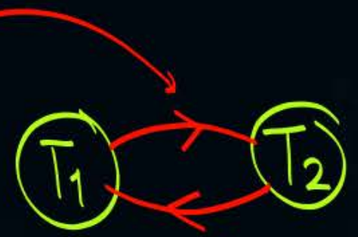
T_1	T_2
$R_1(A)$	
	$W_2(A)$



Precedence graph w.r.t: S_1

S_2

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



Precedence graph

S_3

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

Precedence Graph

No Pair of Conflicting opn

∴ No edge in the graph

Topic : Testing condition for conflict serializable schedule

- ① If precedence graph is acyclic, then given schedule is Conflict Serializable schedule (hence serializable), and Conflict equivalent serial schedules can be given by "Topological order" of precedence graph.

② If precedence graph is Cyclic, then given schedule is not a Conflict serializable schedule, (it may or may not be serializable)

Topic : Topological order

Let $G_T = (V, E)$ be an acyclic precedence graph

{ Indegree : No. of Edges in-coming to vertex v }

Step-1:- Visit a vertex $V \in G_T$, such that in-degree of vertex $V = 0$. And after visiting the vertex 'v' delete vertex 'v' from graph G_T.

{ when we delete a vertex, associated edges will also be deleted }

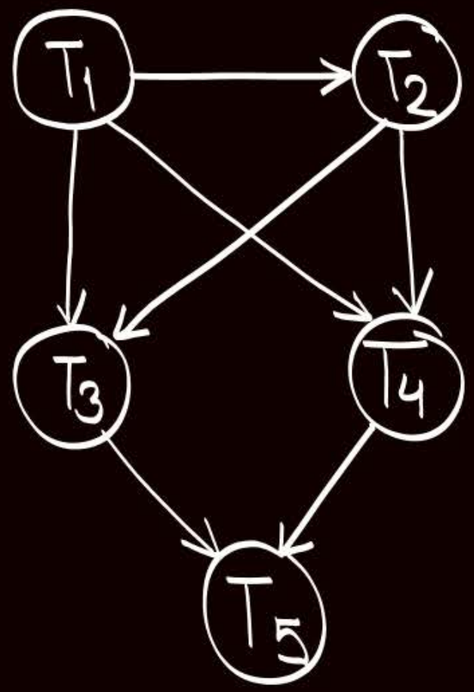
Step-2:- Repeat "Step-1" until graph becomes Empty { i.e. When there is no vertex in the graph }

The order in which vertices of precedence graph G_T can be visited is called topological ordering W.r.t. given Precedence graph

There may be more than one topological ordering for the given precedence graph

eg: Consider the following precedence graph w.r.t. some schedule "S"

H.W.



Identify all topological ordering w.r.t. given Precedence graph

2 mins Summary

- Topic: Cascading rollback problem
- Topic: Cascadeless rollback recoverable schedule
- Topic: Strict recoverable schedule
- Topic: Conflicting and non-conflicting pairs of operations
- Topic: Conflict serializable schedule

THANK - YOU

Slide