COMPUTER SCIENCE

Database Management System

Transaction & Concurrency Control

Lecture_8

Vijay Agarwal sir

01 Recoverable Schedule

Total # of Concurrent Schedule.

- Serial Schedule
- Non Serial Schedule.

$T_1 : 2$

$T_2 : 3$

$T_3 : 4$

$$\text{Total \# Concurrent Schedule} = \frac{(2+3+4)!}{(2!)(3!)(4!)} = 1260 \quad \text{Ans}$$

$$\text{Serial Schedule} = 3! = 6 \text{ Serial Schedule} \quad \text{Ans}$$

$$\text{Non Serial schedule} = 1260 - 6 = 1254 \quad \text{Ans}$$

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)

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

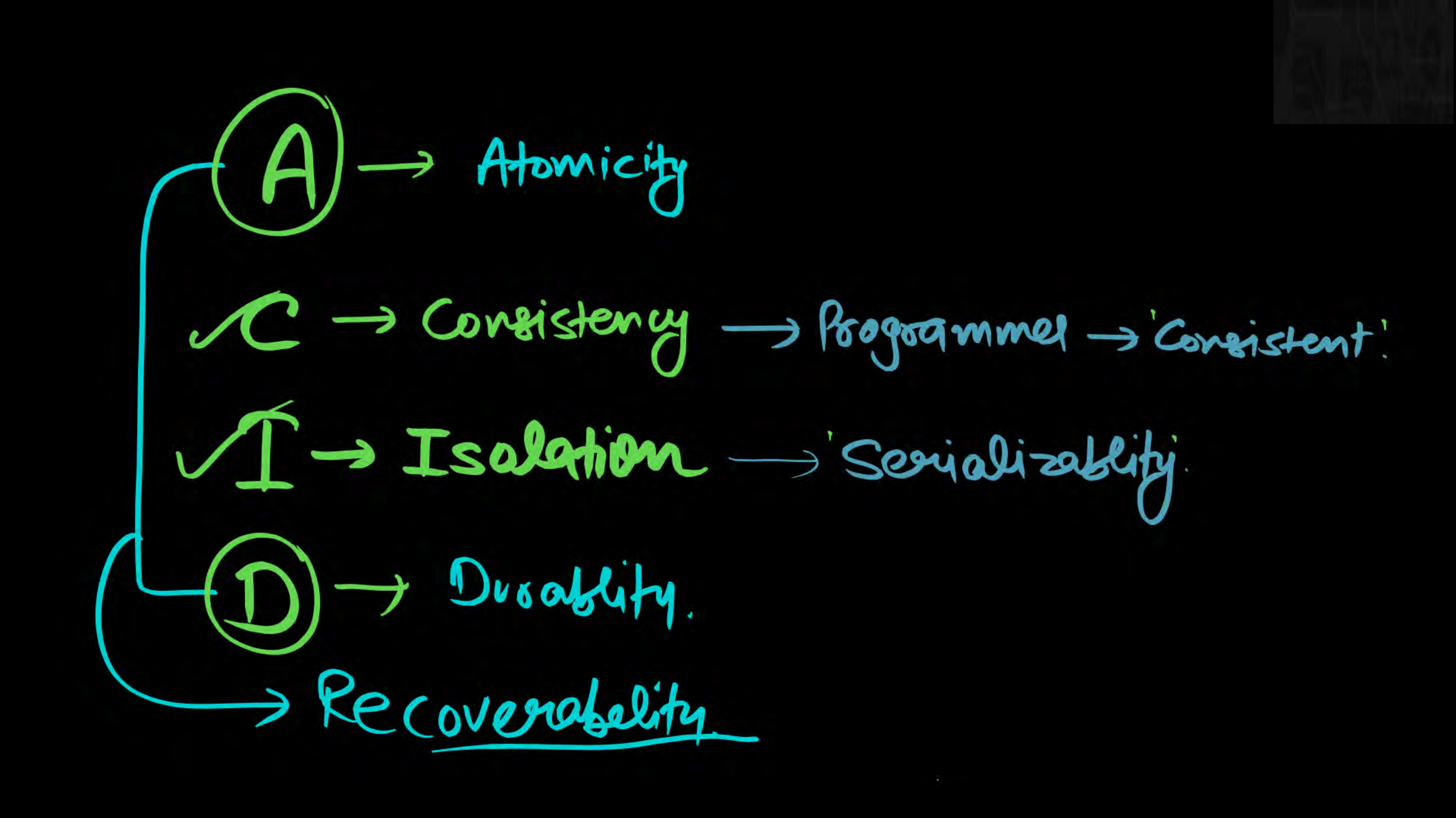
Total non Serial = 4

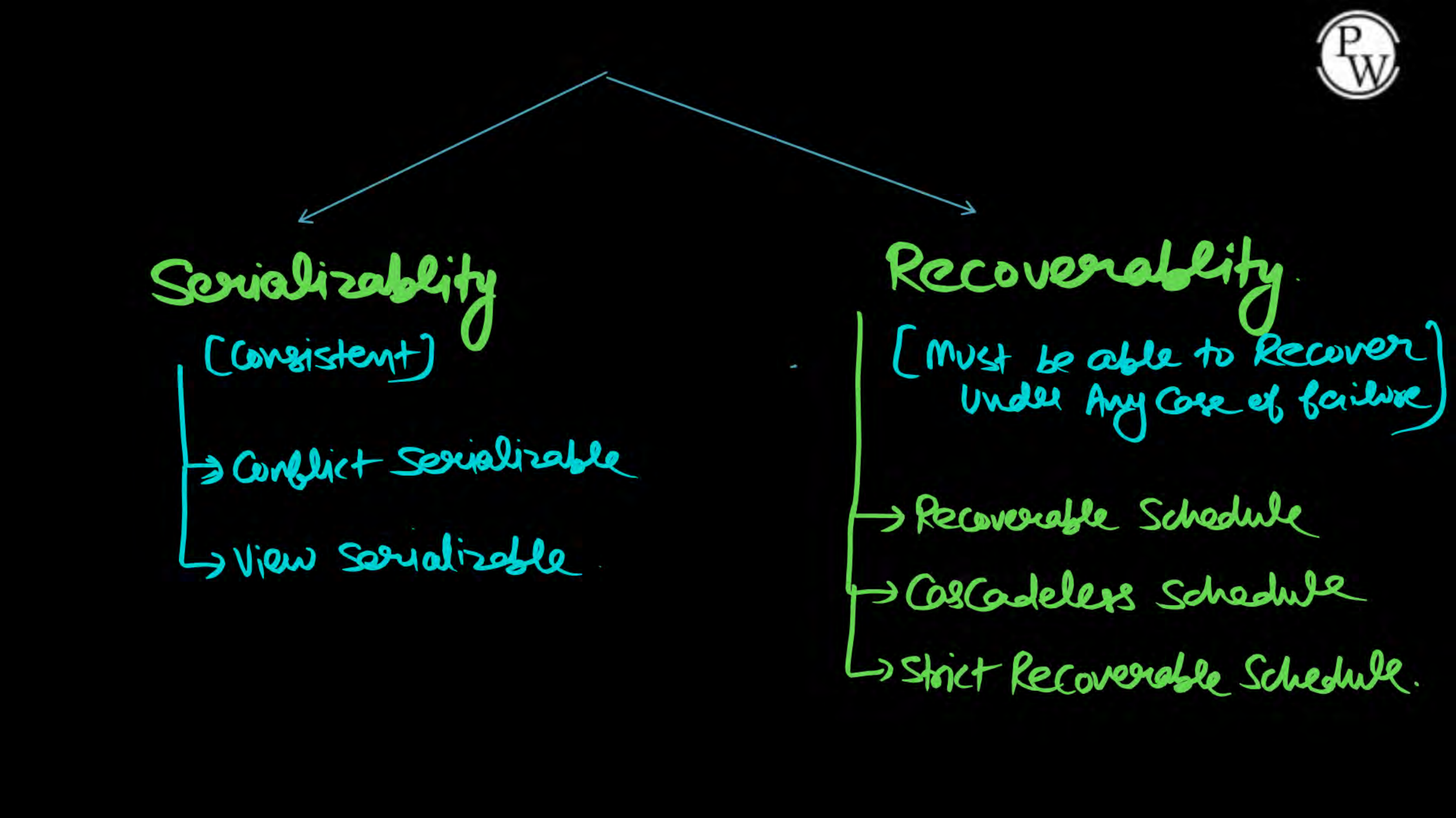
NOTE:

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

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

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


A → Atomicity
C → Consistency → Programmer → 'Consistent'
I → Isolation → 'Serializability'
D → Durability.
→ Recoverability


Serializability
[Consistent]
→ Conflict Serializable
→ View Serializable
Recoverability
[Must be able to Recover Under Any Case of failure]
→ Recoverable Schedule
→ Cascadeless Schedule
→ Strict Recoverable Schedule.

Recoverability.

Complete Schedule : Commit

. Abort

& Read by 'another'

Dependency → 04 06

(1)

T_1	T_2
W(A)	
⋮	
⋮	R(A) → R(B)
Commit	

YES.

(2)

T_1	T_2
W(A)	
commit	
	R(A)
	⋮

No

(3)

T_1	T_2
W(A)	
rollback	
	R(A)

No

(4)

T_1	T_2
W(A)	
	w(A)

No

(5)

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

No

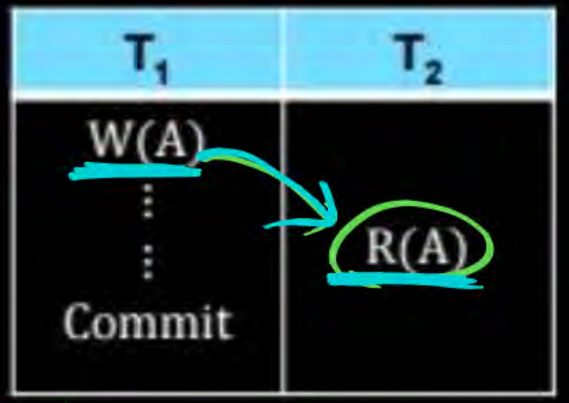
(6)

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

YES.

Dependency

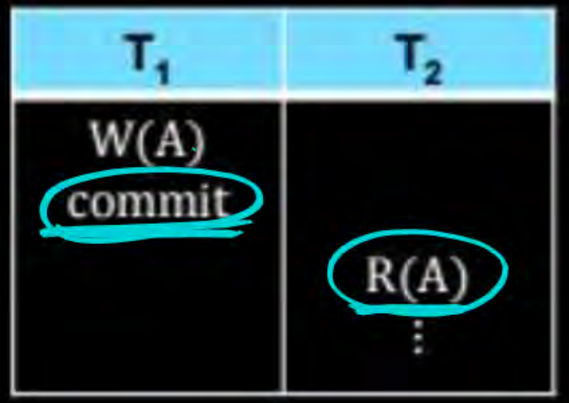
(1)



YES (T_2 Depends on T_1)

Here T_2 Read a value of DATA Item 'A' that is written by Uncommitted Transaction T_1.

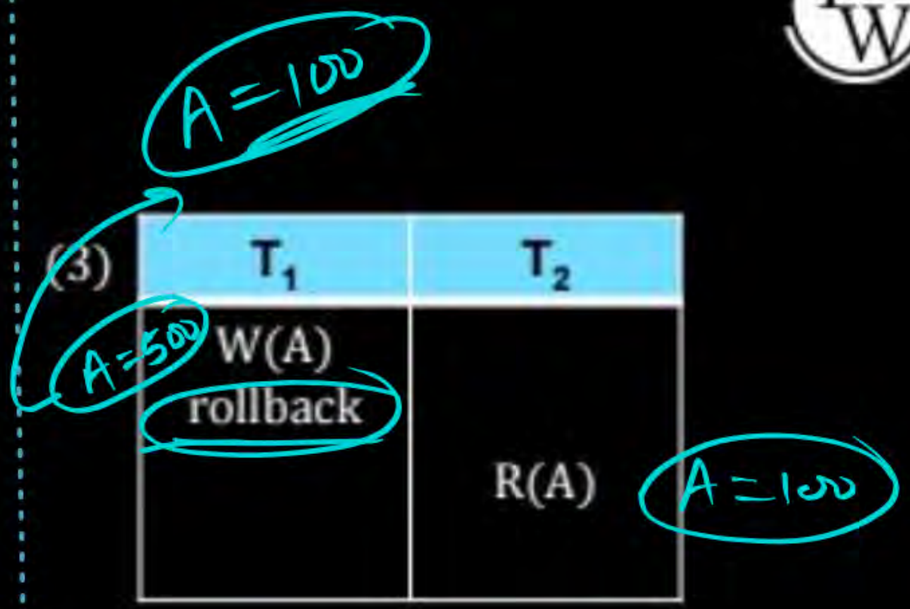
(2)



No Dependency

T_2, Not Depend on T_1 Bcz T_1 Committed

(3)



No Dependency

Bcz T_1 fail then Rollback.

Dependency

Uncommitted Read

Update by one Uncommitted Transaction & Read by another transaction.

(Not Uncommitted / Dirty) Read.

No Dependency.

(4)

T_1	T_2
W(A)	
	w(A)

No Dependency

(5)

T_1	T_2
A=500 W(A)	
	W(A) A=2000
	R(A)

T_2 Depends on T_1

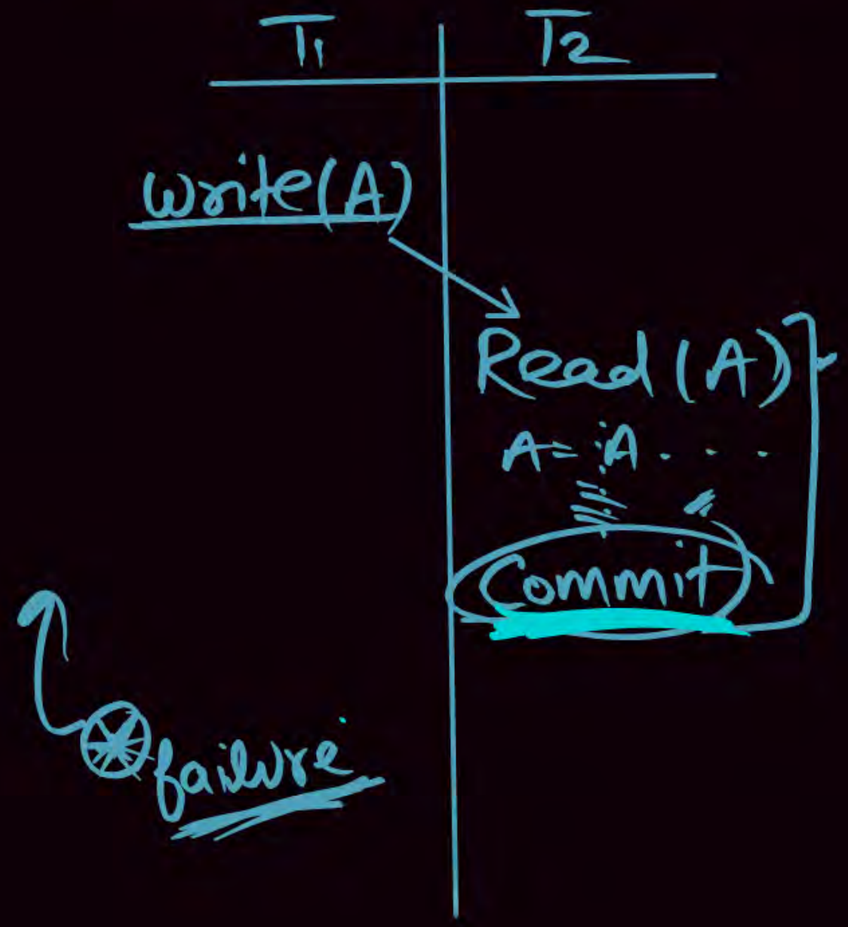
T_1 Depend on T_2.

YES, Dep.

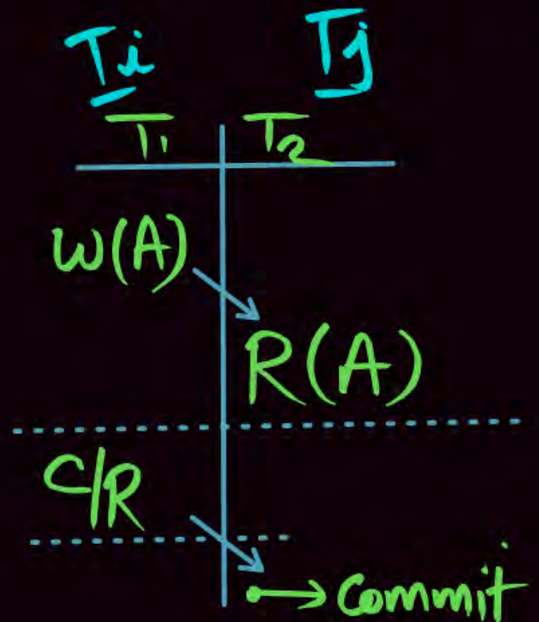
(6)

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

IR - Recoverable | Non-recoverable Schedule



Recoverable Schedule:



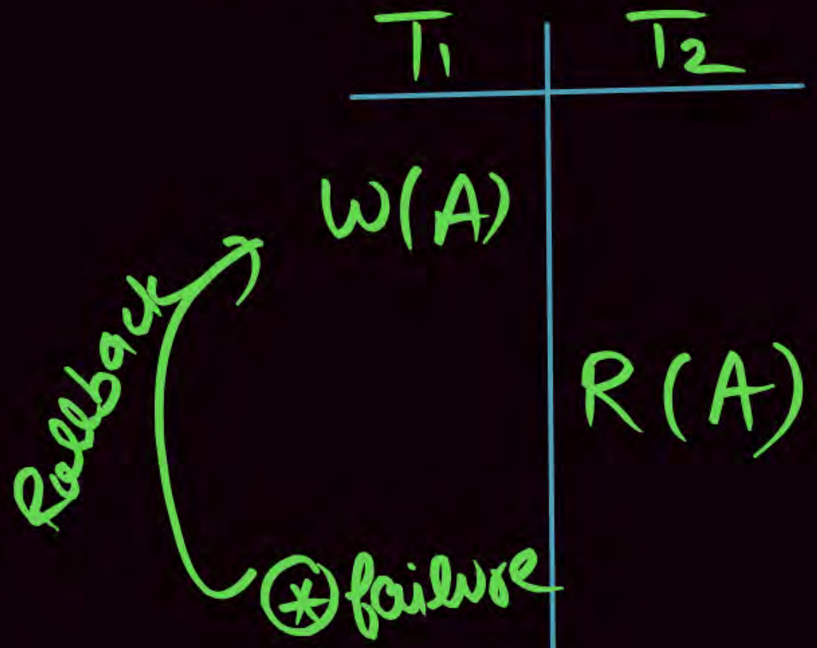
Recoverable Schedule

Recoverable Schedule is one, where for each pair of transaction T_i & T_j such that, if T_j Read a Data Item, that was Previously written by T_i then Commit of T_i Appear before Commit of T_j.

OR

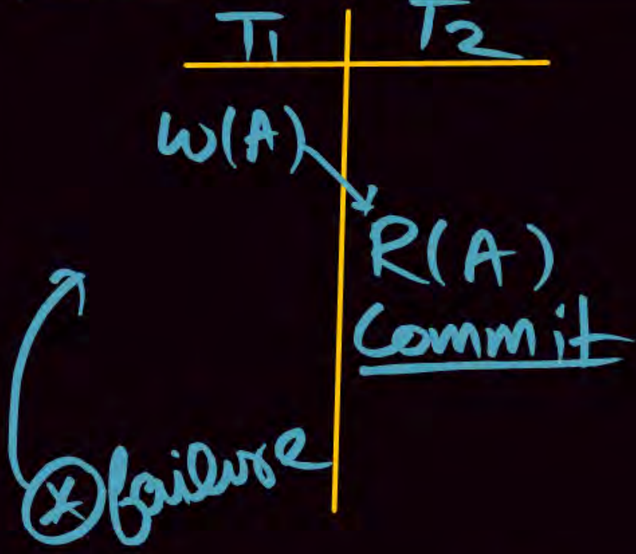
If T_2 Depends on T_1 (T_2 Read a value written by Uncommited transaction) then Commit of T_2 must be Delayed Untill Commit/Rollback of T_1.

Recoverable Schedule

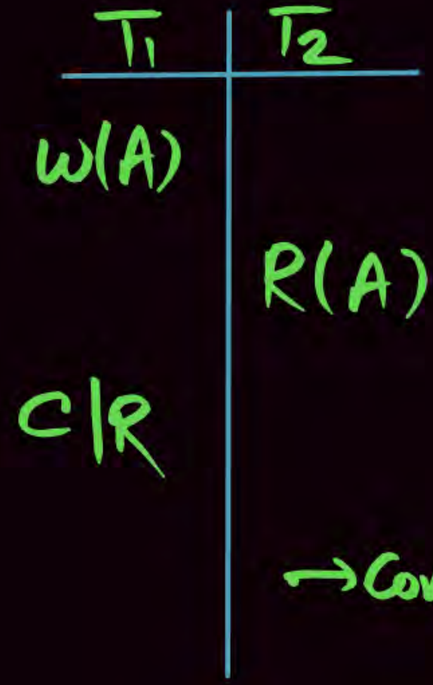


If T_1 fail then T_1 Rollback
& Due to Dependency T_2 also Rollback.

Irrecoverable Schedule.



If T_1 failur then T_1 Tries to Rollback But this value Read by T_2 & T_2 Commit so Not able to Recover.



Recoverable Schedule.

C: Commit

R: Rollback

C_1: Commit of transaction T_1.

C_2: Commit of transaction T_2.

Recoverable Schedules

Need to address the effect of transaction failures on concurrently running transactions.

- Recoverable schedule — if a transaction T_j reads a data item previously written by a transaction T_i, then the commit operation of T_i appears before the commit operation of T_j.
- The following schedule is not recoverable

T_8	T_9
read(A)	
write(A)	Read(A)
	commit
read(B)	

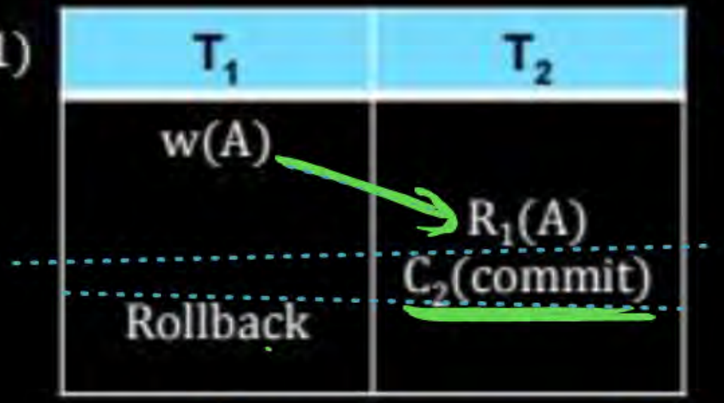
Irrecoverable
(or)
Non Recoverable

- If T_8 should abort, T_9 would have read (and possibly shown to the user) an inconsistent database state. Hence, database must ensure that schedules are recoverable.

Examples

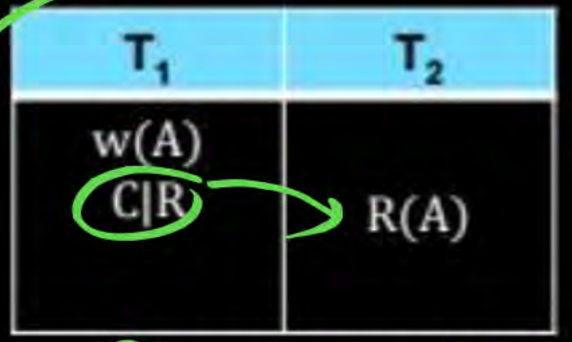
3, 4, 5 & 6 Recoverable.

(1)



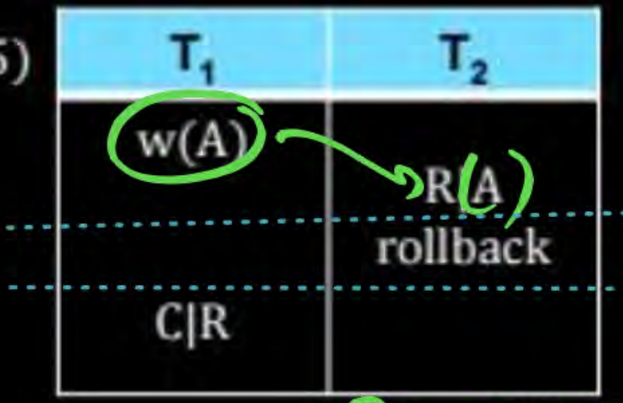
Irrecoverable.

(3)



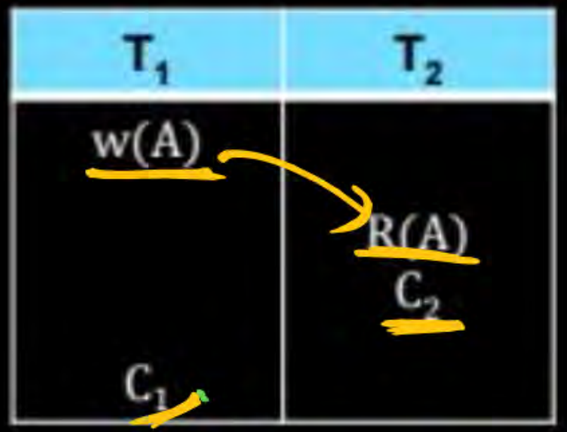
Recoverable

(5)



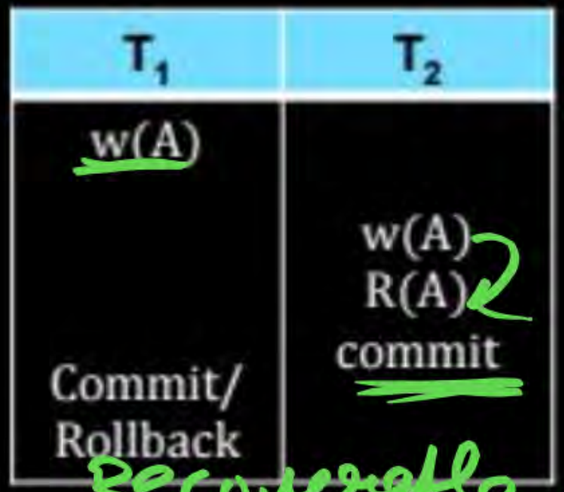
Recoverable

(2)



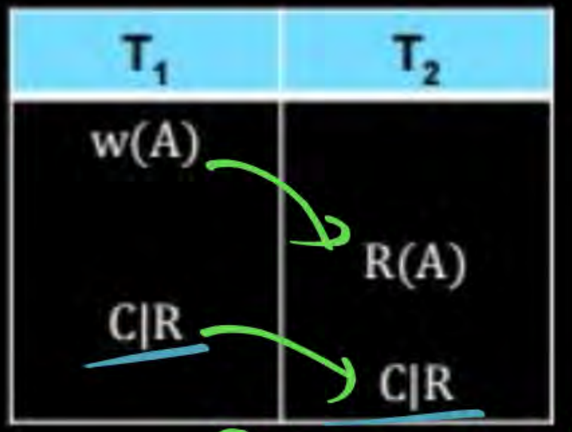
Irrecoverable

(4)



Recoverable.

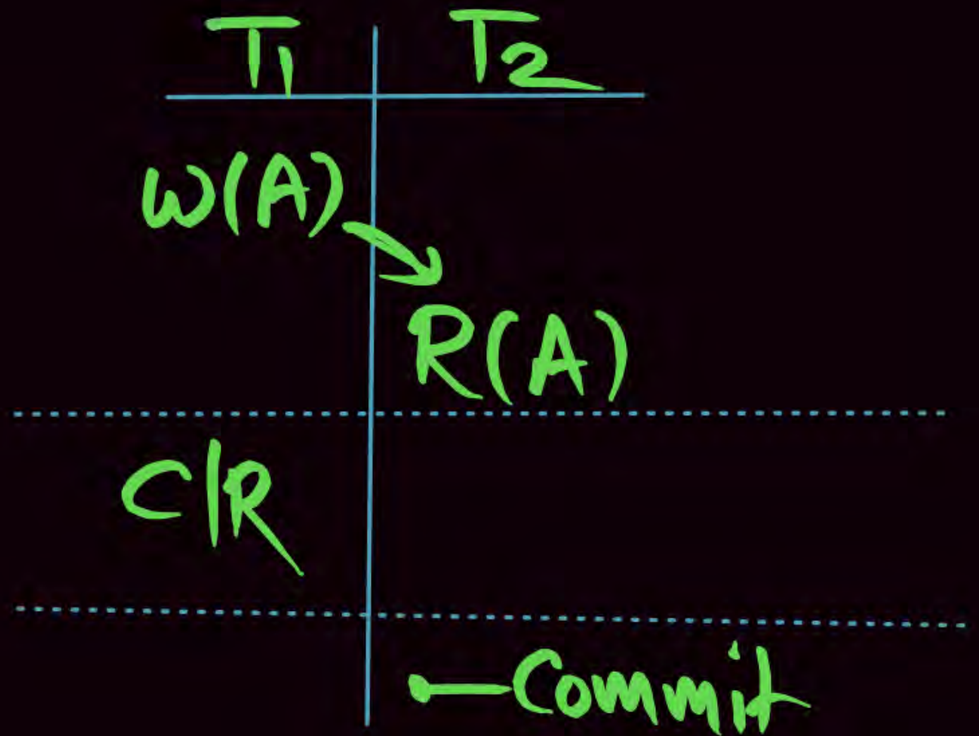
(6)



Recoverable.

Recoverable Schedule.

1. WR / Dirty Read / Uncommitted Problem.
2. WW / Lost Update Problem.
3. RW Problem
4. Cascading Rollback are there.


T1
T2
W(A)
R(A)
C/R
Commit

NOTE: Recoverable schedule may or may not be free from

1. WR problem / uncommitted Read
2. RW Problem
3. WW Problem

T_1	T_2
R(A)	
	W(A)
	Commit
Commit	

Recoverable
But RW Problem

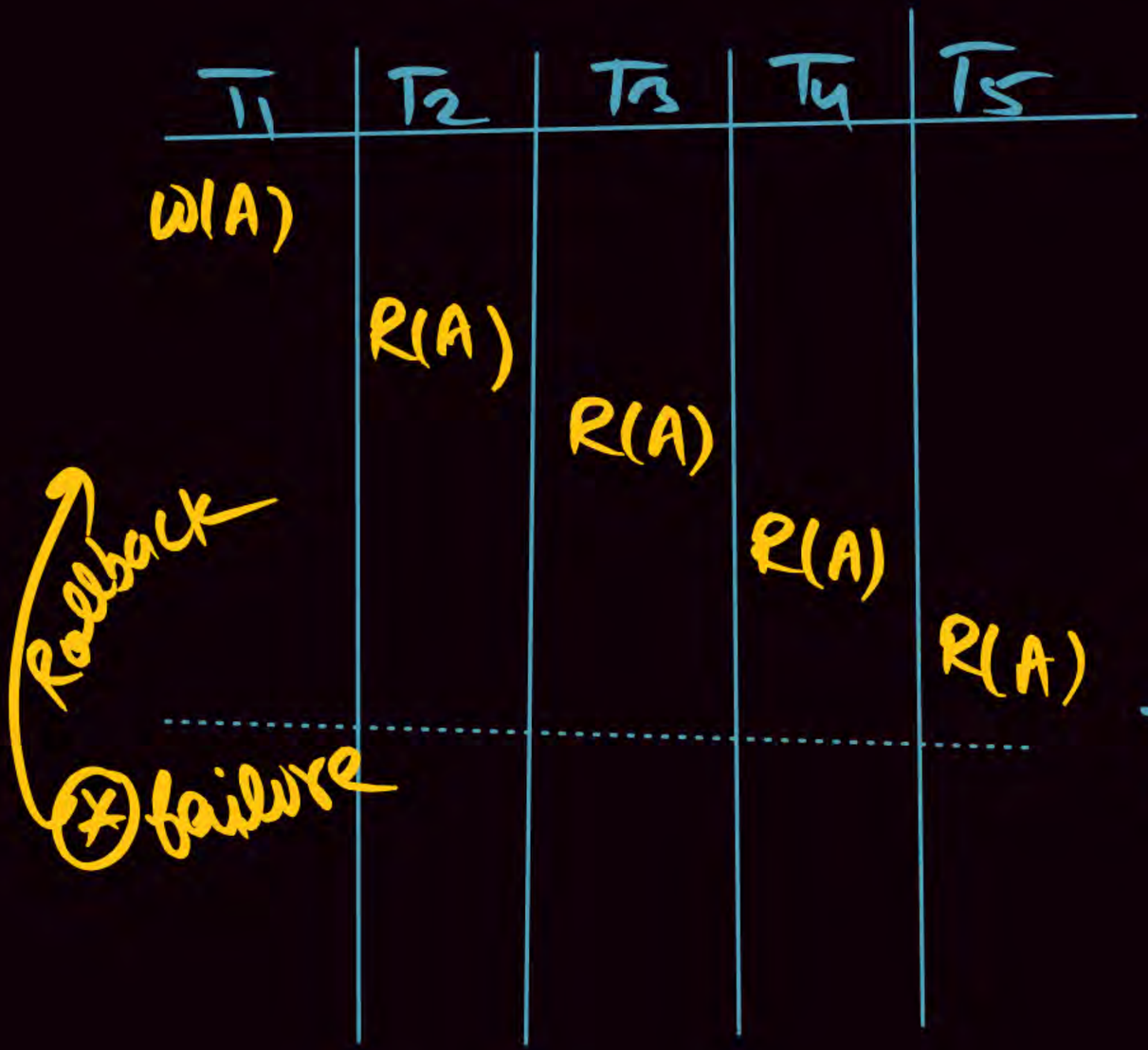
T_1	T_2
W(A)	
	W(A)
	Commit
Commit	

Recoverable
But WW Problem

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

Recoverable
But WR Problem

Cascading Rollback are possible.



Here T_2, T_3, T_4 & T_5 Depends on Transaction T_1.

If T_1 fails then T_1 Rollback then Due to Dependency T_2, T_3, T_4 & T_5 also Rollback.

Cascading Rollbacks

- Cascading rollback – a single transaction failure leads to a series of transaction rollbacks. Consider the following schedule where none of the transactions has yet committed (so the schedule is recoverable)

T_{10}	T_{11}	T_{12}
read(A)		
read(B)		
write(A)	read(A)	
	write(A)	
		read(A)
abort		

If T_{10} fails, T_{11} and T_{12} must also be rolled back.

- Can lead to the undoing of a significant amount of work

② Cascadeless Schedule:

T_1	T_2
W(A)	
C/R	
	R(A)

A Cascadeless Schedule is one where for each pair of Transaction T_i & T_j such that T_j Read a Data Item that was Poreviously written by T_i, then Commit of T_i appear before Read of T_j.

OR

Not allowed Uncommitted Read.

Cascadeless Schedules

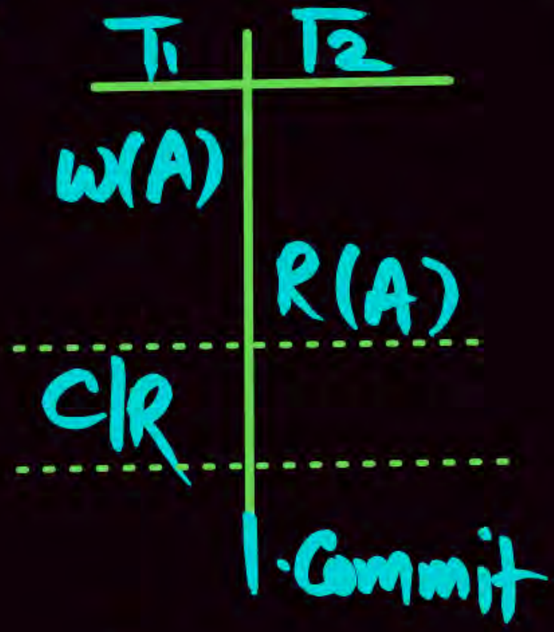
Cascadeless Rollback Recoverable

- Cascadeless schedules — cascading rollbacks cannot occur;
  - 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.
- Every cascadeless schedule is also recoverable

T_1	T_2
W(A) C|R	R(A)

Cascadeless Schedule

(1)

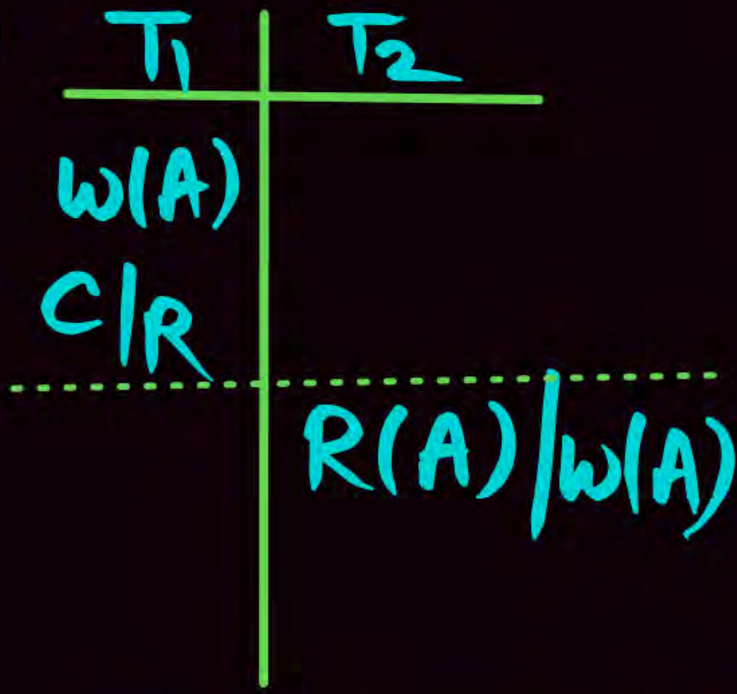


Recoverable.

(2)

Cascadeless.

(3)



Strict Recoverable (OR) STRICT Schedule.

NOTE: Cascadeless schedule may or may not be free from

- RW Problem
- WW Problem

T_1	T_2
R(A)	
	W(A)
	Commit
Commit	

Cascadeless
But RW Problem

T_1	T_2
W(A)	
	W(A)
	Commit
Commit	

Cascadeless
But WW Problem

Strict Recoverable Schedule

T_1	T_2
W(A) C|R	R(A)|w(A)

No WR Problem.
No WW Problem.

Only RW Problem

NOTE: Strict Recoverable schedule may or may not be free from

- RW Problem

T_1	T_2
R(A)	
	W(A)
	Commit
Commit	

Strict Recoverable
But RW Problem

(1)

T_1	T_2
W(A)	
	R(A)
C|R	
	Commit

Recoverable

(2)

T_1	T_2
W(A)	
C|R	
	R(A)

Cascadeless

(3)

T_1	T_2
W(A)	
C|R	
	R(A)/W(A)

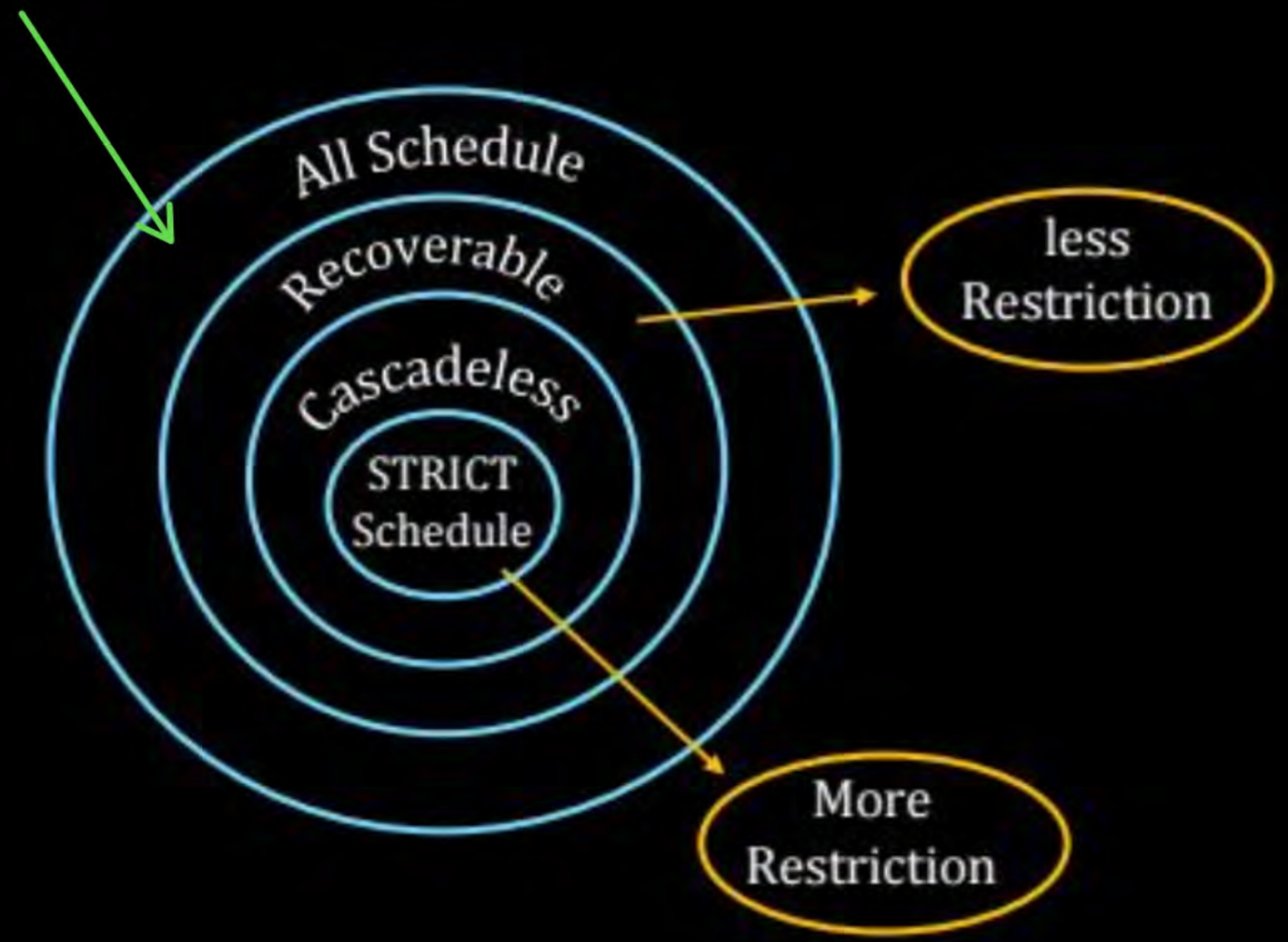
Strict schedule

WR
WW
RW
Problems

Cascading Rollback

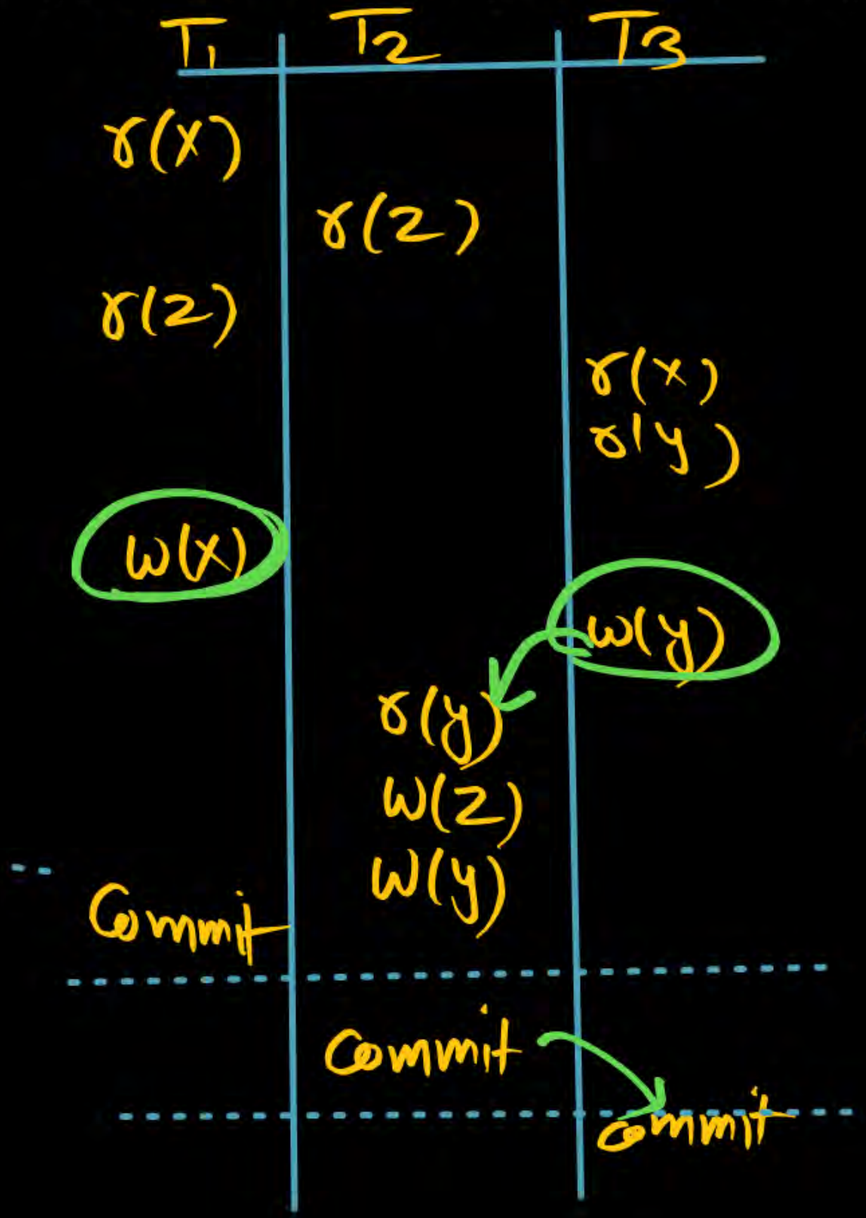
WW
RW
Problems

RW Problem


All Schedule
Recoverable
Cascadeless
STRICT
Schedule
less
Restriction
More
Restriction

Q. $r_1(x)r_2(z)r_1(z)r_3(x)r_3(y)w_1(x)w_3(y)r_2(y)w_2(z)w_2(y)\ C_1\ C_2\ C_3$

↓ Commit 3

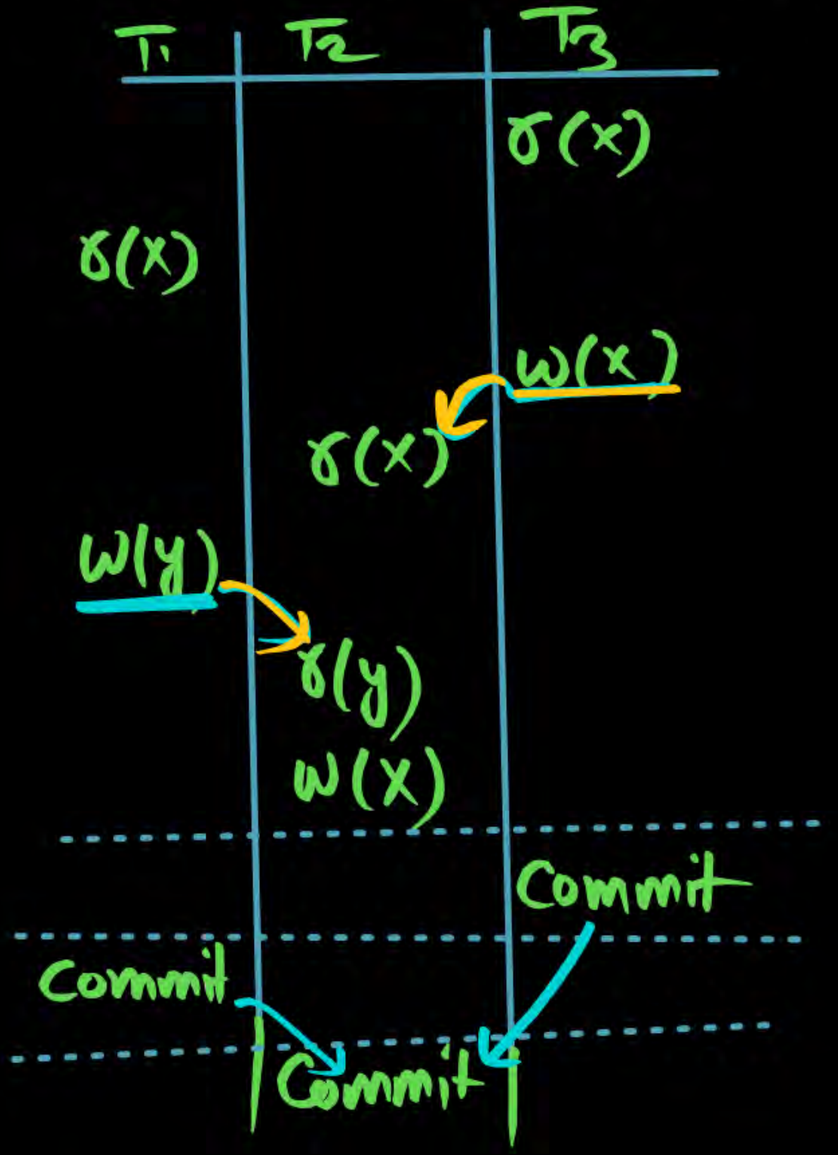


W → R

✗ Recoverable.

Non Recoverable.

Q. $r_3(x)\ r_1(x) w_3(x) r_2(x) w_1(y)\ r_2(y) w_2(x) C_3 C_1\ C_2$

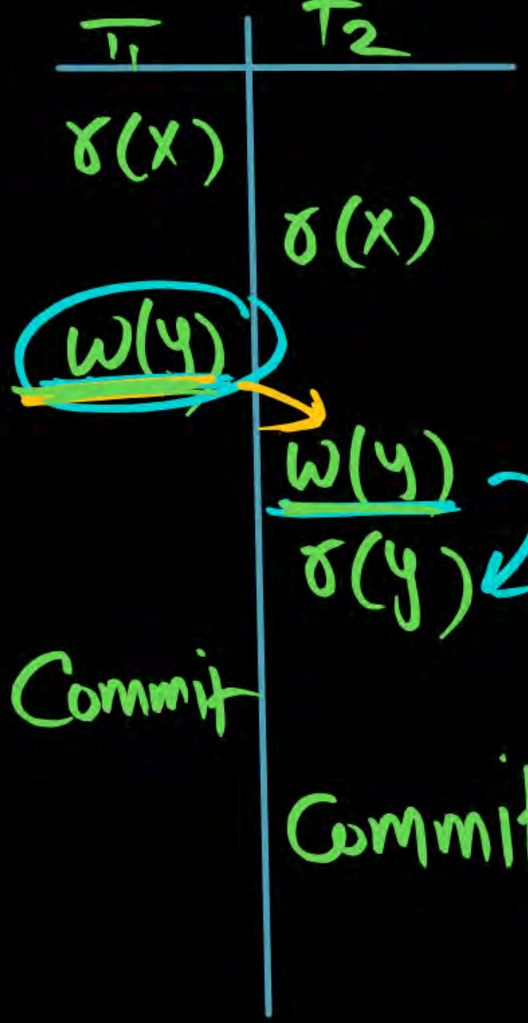


✓ Recoverable.

✗ Cascadeless.

Bcz on Data Item X & Y Uncommitted Read.

Q. $r_1(x)\ r_2(x) w_1(y) w_2(y) r_2(y)\ C_1\ C_2$



✓ Recoverable

✓ Cascadeless
[No Uncommitted Read
(No WR Case)]

✗ Strict Recoverable

Q.68

Consider the following database schedule with two transactions, T_1 and T_2.

$S = r_2(X); r_1(X); r_2(Y); w_1(X); r_1(Y); w_2(X); a_1; a_2$

where $r_i(Z)$ denotes a read operation by transaction T_i on a variable Z, $w_i(Z)$ denotes a write operation by T_i on a variable Z and a_i denotes an abort by transaction T_i

Which one of the following statements about the above schedule is TRUE?

[MCQ:2016–2M]

A. S is non-recoverable

B. S is recoverable, but has a cascading abort

C. S does not have a cascading abort

D. S is strict

Q.69

Let S be the following schedule of operations of three transactions T_1, T_2 and T_3 in a relational database system:

$$R_2(Y), R_1(X), R_3(Z), R_1(Y), W_1(X), R_2(Z), W_2(Y), R_3(X), W_3(Z)$$

Consider the statements P and Q below:

P: S is conflict-serializable.

Q: If T_3 commits before T_1 finishes, then S is recoverable.

Which one of the following choices is correct?

[MCQ: 2021-2M]

(A) Both P and Q are true.

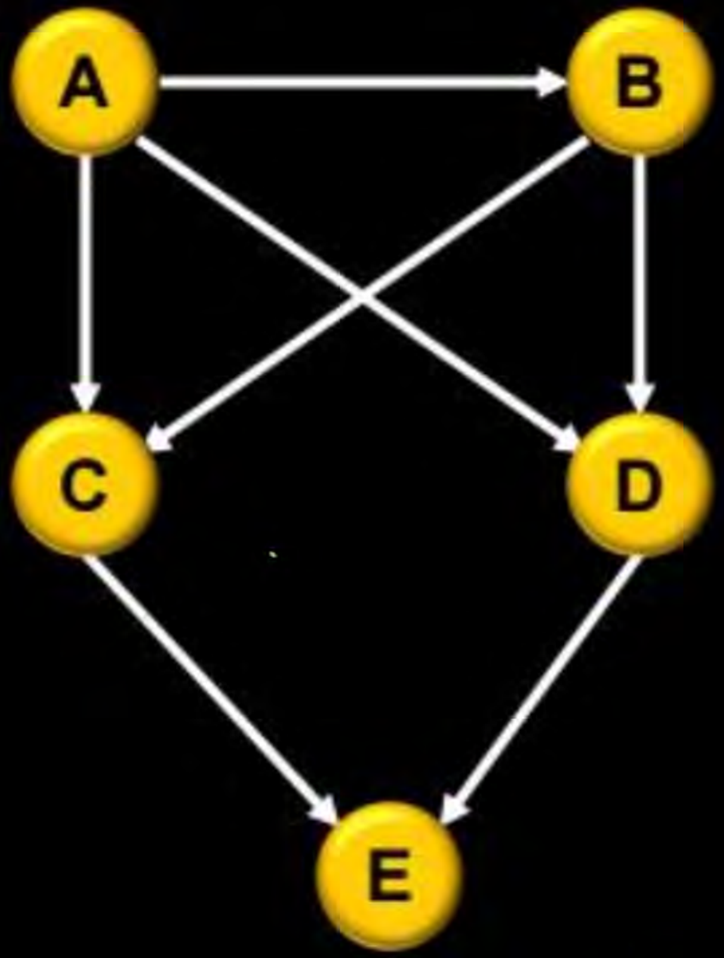
(B) P is true and Q is false.

(C) P is false and Q is true.

(D) Both P and Q are false.

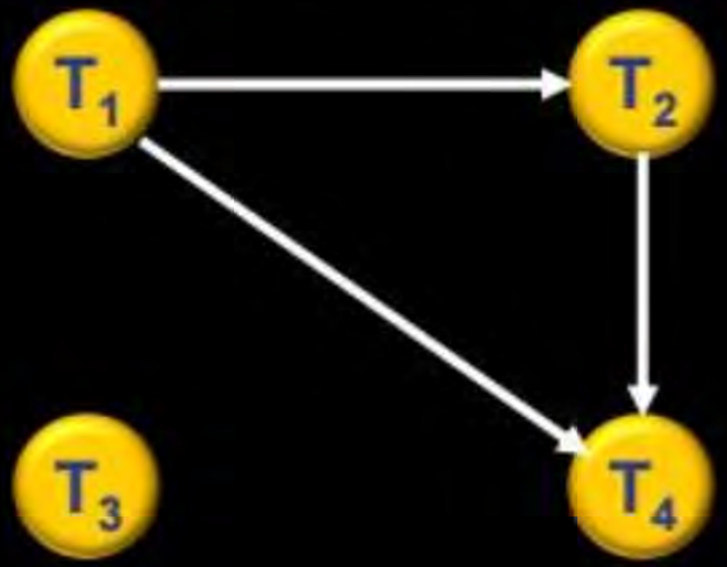
Topological Sorting

Q.

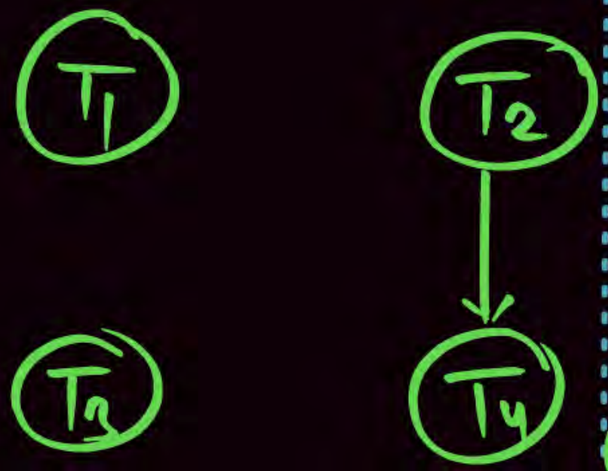


Topological Sorting

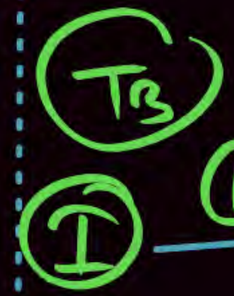
Q.



Q



① $\underline{\quad}$ T_2 ② $\underline{\quad}$ T_4 ③ $\underline{\quad}$



I ① $\underline{\quad}$ T1 ② $\underline{\quad}$ T_2 ③ $\underline{\quad}$ T_4 ④ $\underline{\quad}$ = 4

II $\_\_$ T_2 $\_$ T1 $\_\_$ T_4 $\_\_$ = 4

III $\_$ T_2 $\_$ T_4 $\_$ T1 $\_\_$ = 4

12 Ans

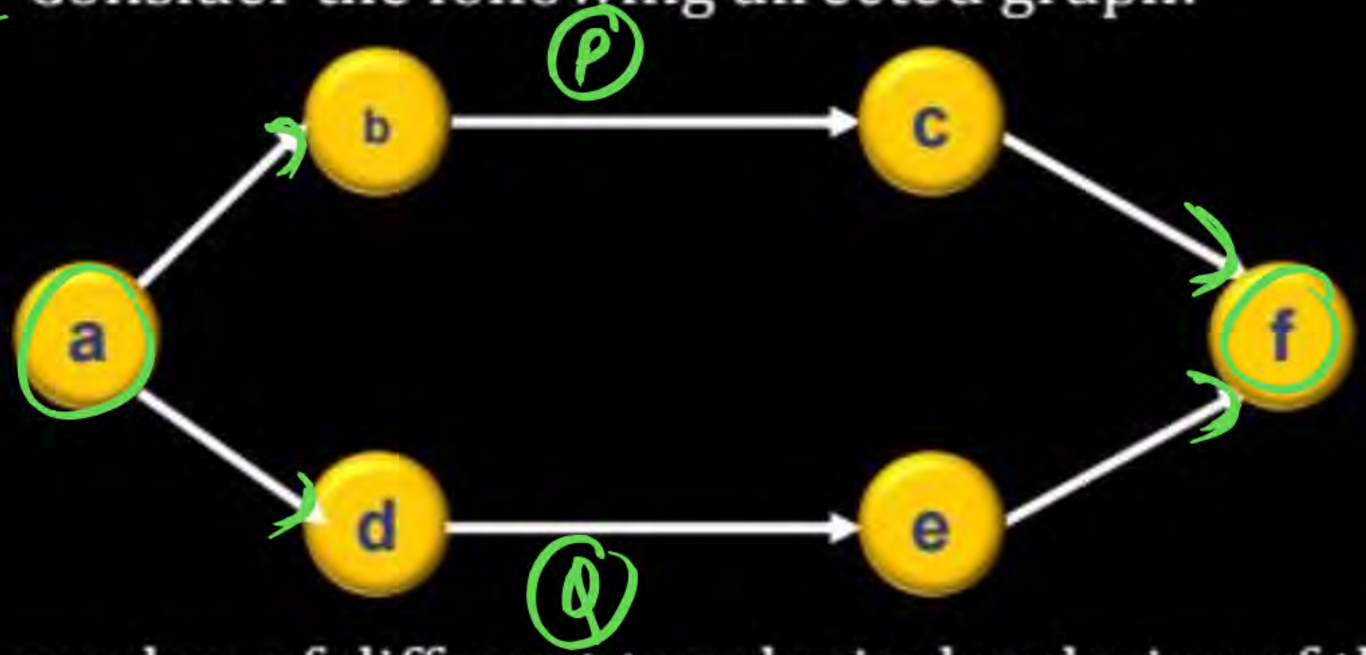
Topological Sorting

Q.

$R_4(A)$ $R_2(A)$ $R_3(A)$ $W_1(B)$ $W_2(A)$ $R_3(B)$ $W_2(B)$

Topological Sorting

Q.70 Consider the following directed graph:



a b c d e f
a d e b c f
a b d c e f
a b d e c f
a d b e c f
a d b c e f

The number of different topological ordering of the vertices of the graph is _____.

[MCQ: 2016]

THANK YOU!