COMPUTER SCIENCE

Database Management System

Transaction & Concurrency Control

& Lock Based Protocol -01

Lecture_8

Vijay Agarwal sir


TOPICS
TO BE
COVERED
01
Recoverable Schedule
02
Lock Based Protocol

Transaction Concept

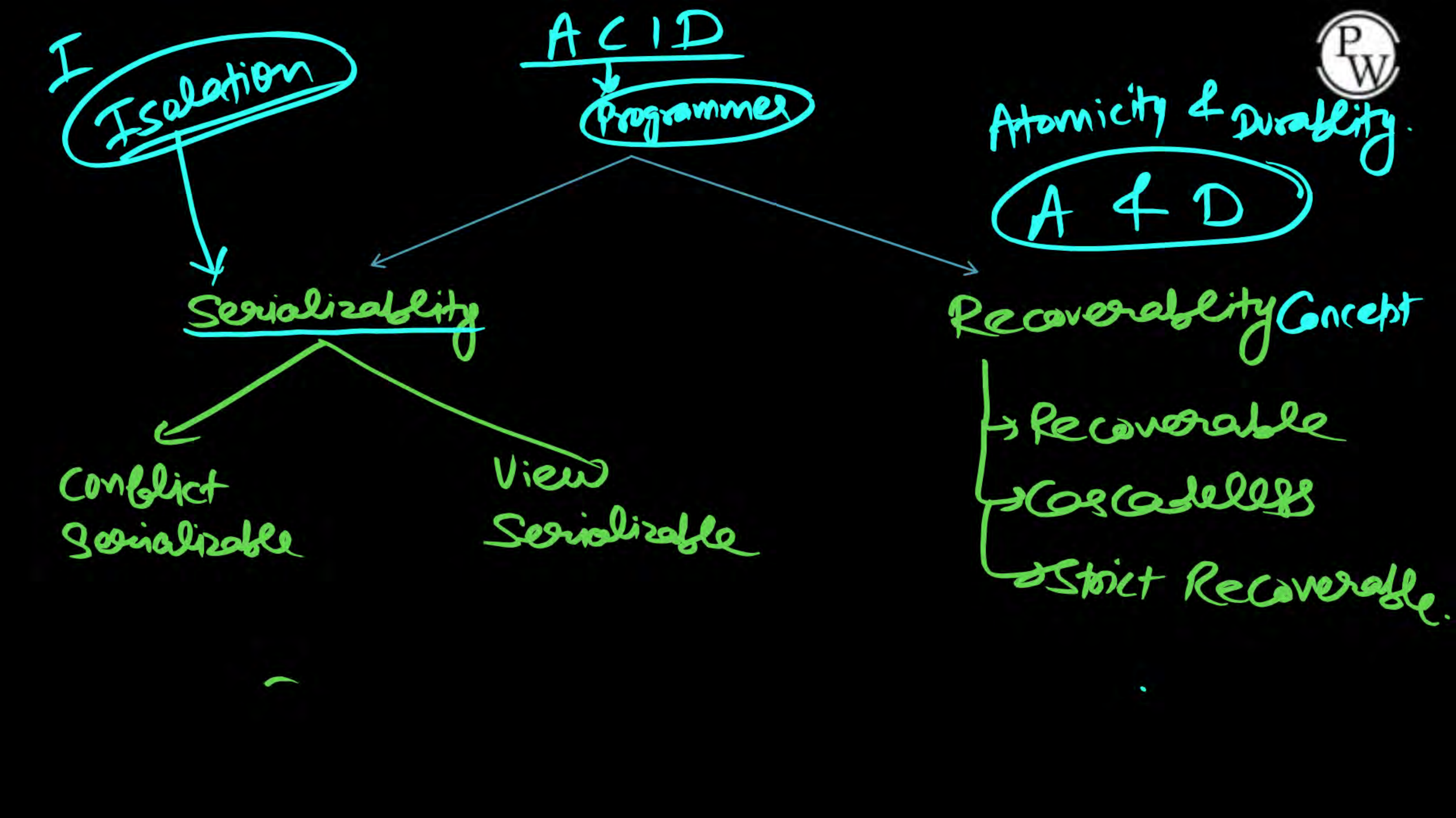
ACID Properties.

Schedule

- Serial
- Non Serial

Serializable

- Conflict
- View.


I
Isolation
ACID
Programmer
Atomicity & Durability.
A & D
Serializability
Recoverability Concept
Conflict Serializable
View Serializable
Recoverable
Cascadeless
Strict Recoverable.

$T_1 \rightarrow (T_2)$

Case I : (3)

Case II : (2)

Case III : (1)

(6)

$T_2 \rightarrow (1)$

3

2

1

(6)

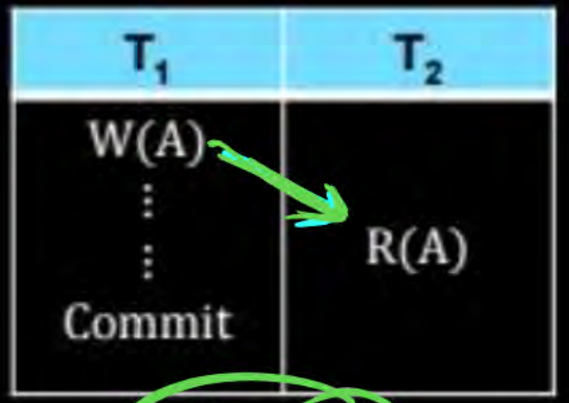
$R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)$

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

Dependency

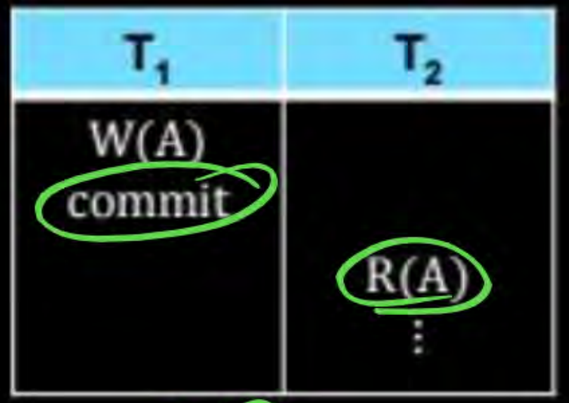
Between the transactions.

(1)



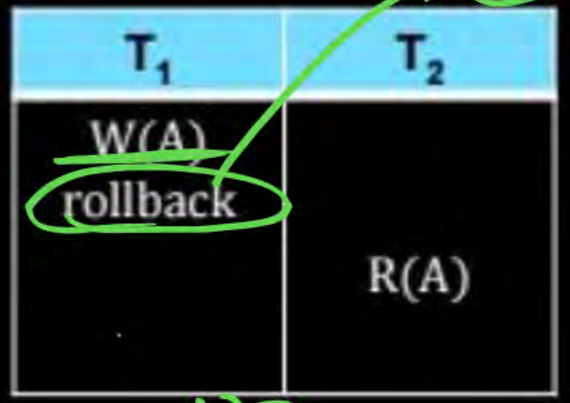
YES

(2)



NO

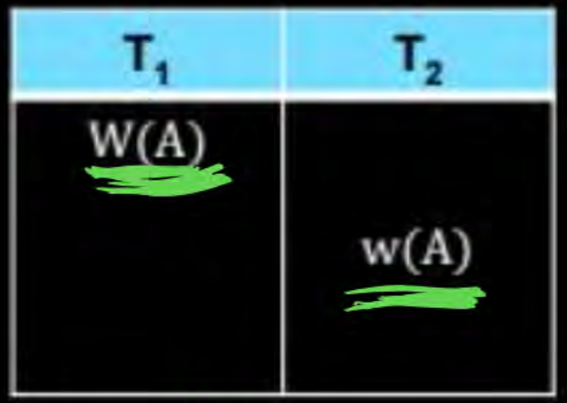
(3)



UNDO

NO

(4)

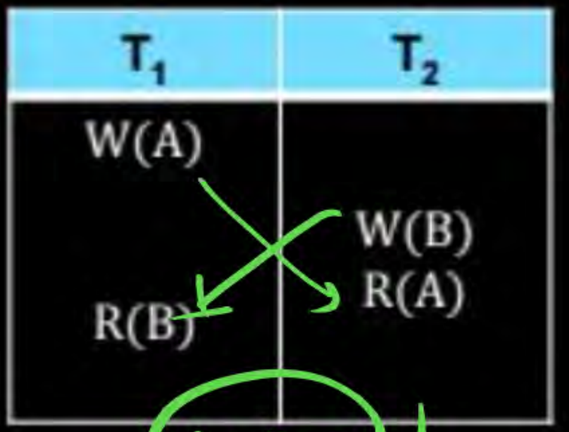


No

(5)

No

(6)



YES

Dependency

(1)

T_1	T_2
W(A) ⋮ Commit	R(A)

YES

Here T_2 Depends on T_1 Bcz T_2 Read the Value of Data Item A Which is written (Updated) by T_1, its Uncommitted Read (Dirty Read)

(2)

T_1	T_2
W(A) commit	R(A) ⋮

No

No Dependency

Bcz Transaction T_1 Committed. So Now T_2 Read Value of A from DB, Not From T_1.

(3)

T_1	T_2
W(A) Rollback	R(A)

A=50,000

A:200

50,000

A:200

No

No Dependency

Bcz After Updation on A, T_1 Rollback so UNDO ALL Modification.

So T_2 Not Depend on T_1

Dependency

No Dependency
Bcz T_2 Not Read the Value of A which is Updated by T_1.

(4)

T_1	T_2
W(A)	
	w(A)

No Dependency.
Bcz T_2 Read the Value of A which is Updated by Same Transaction
(its Not Uncommitted/ Dirty Read)

(5) No

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

YES Dependency
T_2 Depend On T_1 & T_1 Dependy On T_2

(6) YES

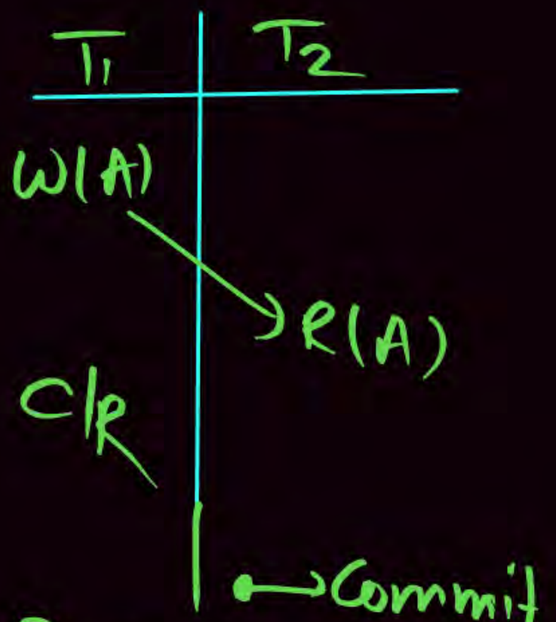
T_1	T_2
W(A)	
	W(B)
R(B)	
	R(A)

Uncommitted / Dirty Read

Modified (Update / written) by One transaction

& Read by another / Different Transaction

Recoverable Schedule



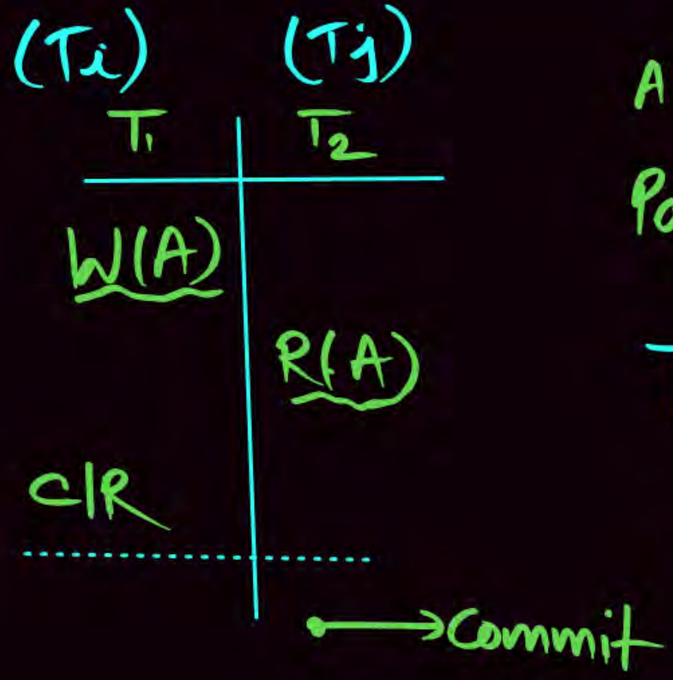
Recoverable

Recoverable Schedule

If T_2 Depends on T_1 (T_2 Read the value which is written/UPDATED By T_1 then T_2 Commit Must be Delayed Untill Commit[C] (OR) Rollback[R] of T_1.

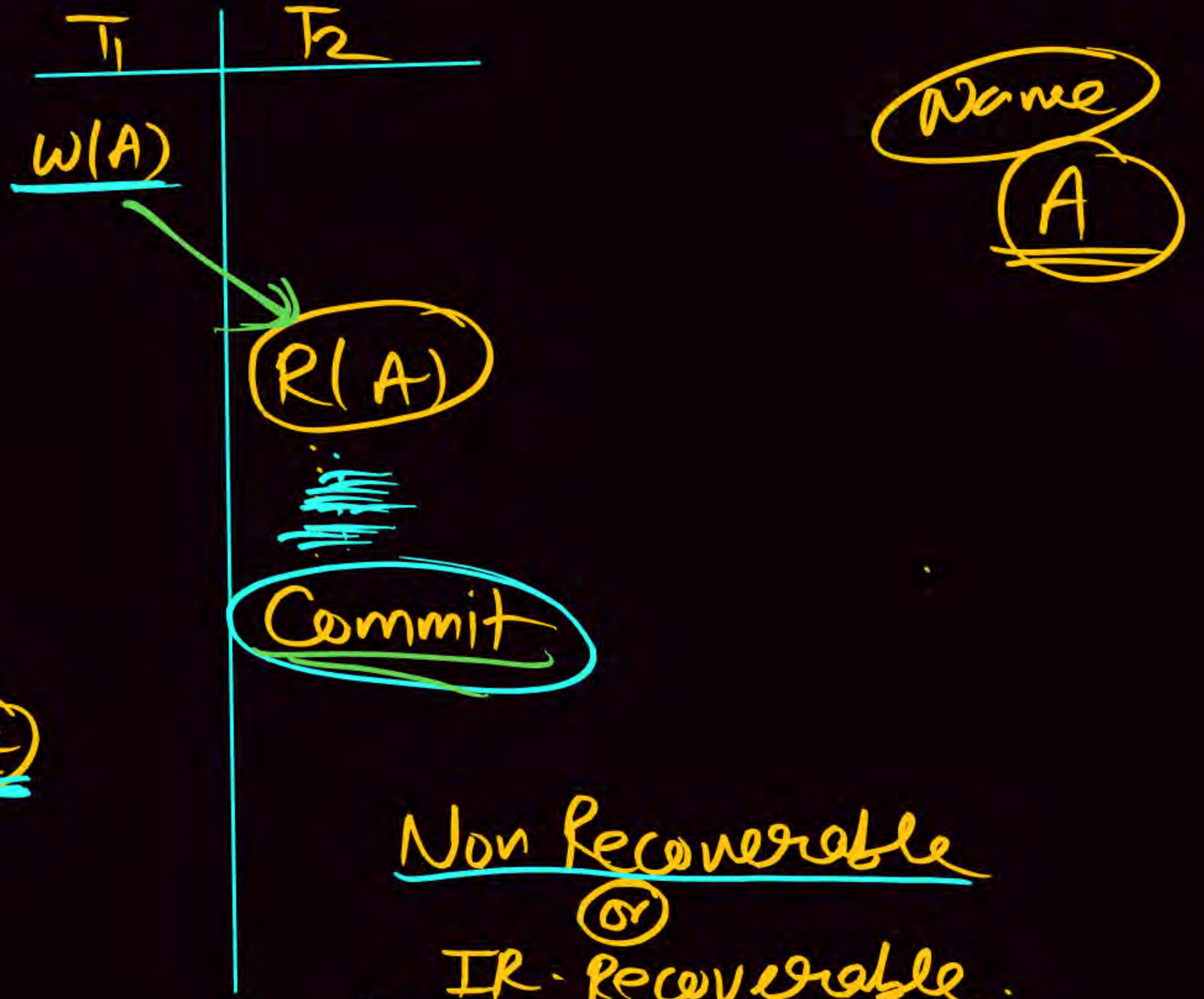
is Called Recoverable Schedule.

Recoverable Schedule



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

IR Recoverable/
Non Recoverable.



Non Recoverable
(or)
IR - Recoverable.

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

If T_8 Commit Appear before of Commit of T_9 then Recoverable

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

→ Irrecoverable

Examples

Recoverable Schedule

(1)

T_1	T_2
w(A)	
	R_2(A)
	C_2(commit)
Rollback	

IR Recoverable

(3)

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

Recoverable

(5)

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

Recoverable

(2)

T_1	T_2
w(A)	
	R(A)
	C_2
C_1	

Irrecoverable

(4)

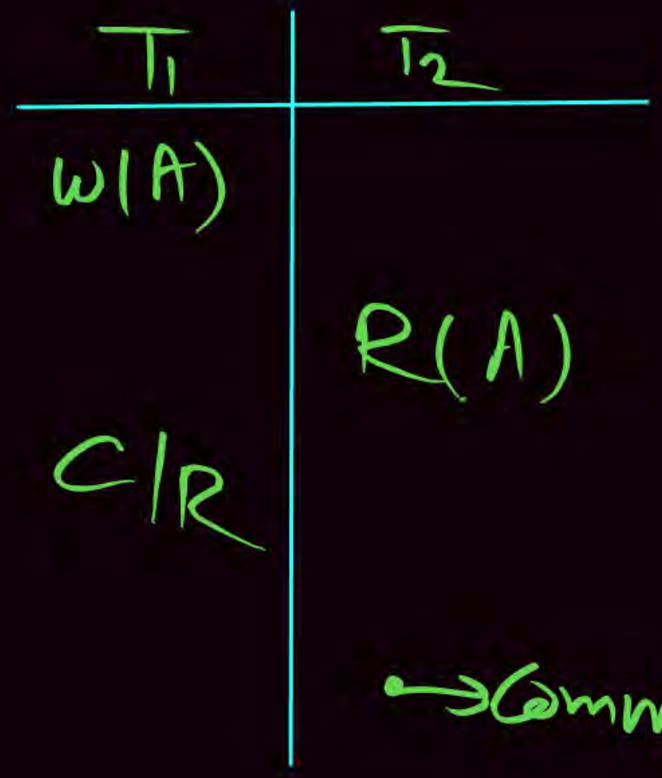
T_1	T_2
w(A)	
	w(A)
	R(A)
	commit
Commit/ Rollback	

Recoverable

(6)

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

Recoverable



Uncommitted / Dirty Read.

Recoverable schedule

Subset from

① WR, RW, WW

② Cascading Rollback Possible.

NOTE: Recoverable schedule may or may not be free from

- WR problem / uncommitted Read
- RW Problem
- 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
R|A	
W(A)	
	R(A)
W(B)	
	R(B)
Commit	
	Commit

Recoverable
But WR Problem

T_1	T_2
R(A)	
	W(A)
C_1	
	C_2

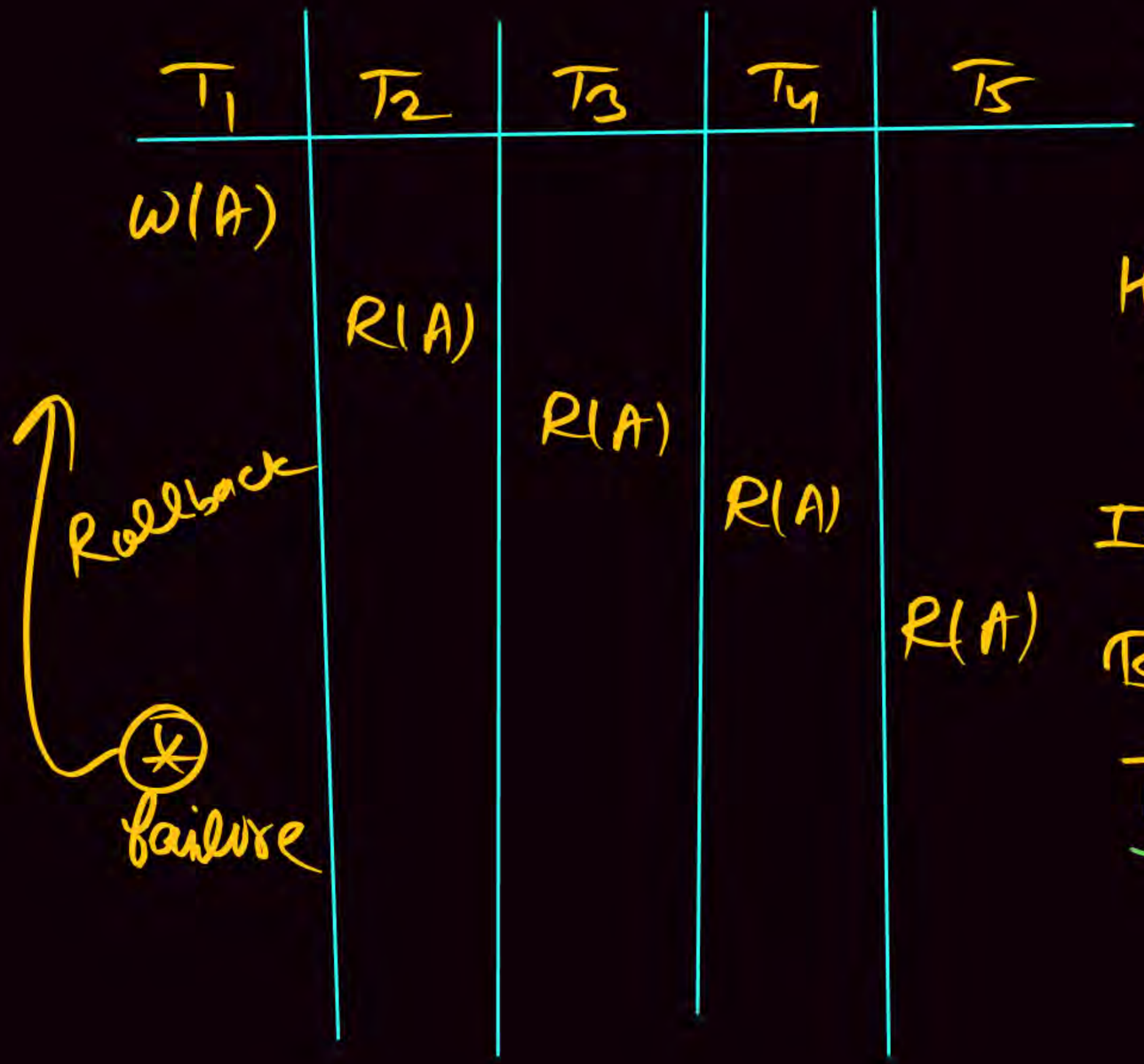
T_1	T_2
W(A)	
	Commit
✗ Fail	

Recoverable

T_1	T_2
	R(A)
	R(B)
W(A)	
	W(A) or W(B) or R(B)
	Commit
✗ failure	

Recoverable

Cascading Rollback



Here T_2, T_3, T_4 & T_5 Depends on T_1

If T_1 fail the Rollback T_1

But 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

Cascadeless Schedule (or) Cascadeless Rollback Recoverable 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 which was Previously written by T_i then Commit of T_i appear before Read of T_j

Cascadeless

⇓

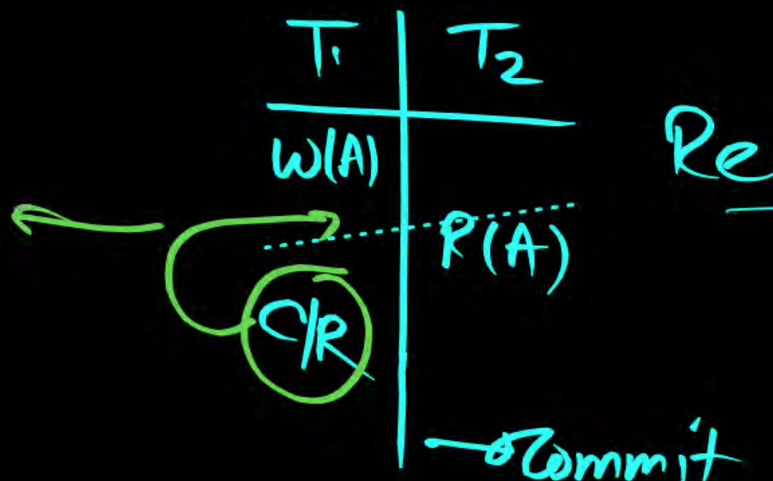
Not allowed Uncommitted / Dirty Read.

Cascadeless Schedules

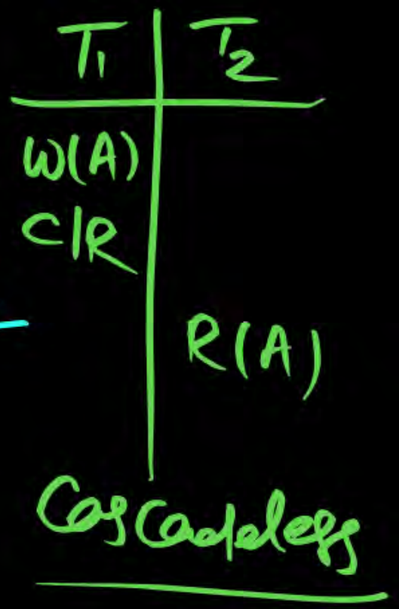
- Cascadeless schedules — cascading rollbacks cannot occur;
  - For each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before the read operation of T_j.
- Every cascadeless schedule is also recoverable

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

Cascadeless Schedule



Recoverable



Cascadeless

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

Cascadeless Schedule

- Free from WR Problem
- No Uncommitted Read
- No Cascading Rollback

(1)

T_1	T_2
W(A)	
	R(A)
C/R	
	→ Commit

Recoverable Schedule

(2)

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

Cascadeless Schedule

(3)

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

Strict Recoverable.

Strict Recoverable Schedule

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

If one Transaction Perform Write operation on Data Item A, then Other transaction T_2 Not allowed to Performed Read(A) & Write(A) untill Commit/Rollback of T_1 Called Strict Recoverable Schedule.

Free from
WR Problem
WW Problem
& No Cascading RollBack

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

V.V.V.V.V.V. Imp.

(1)

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

Recoverable

(2)

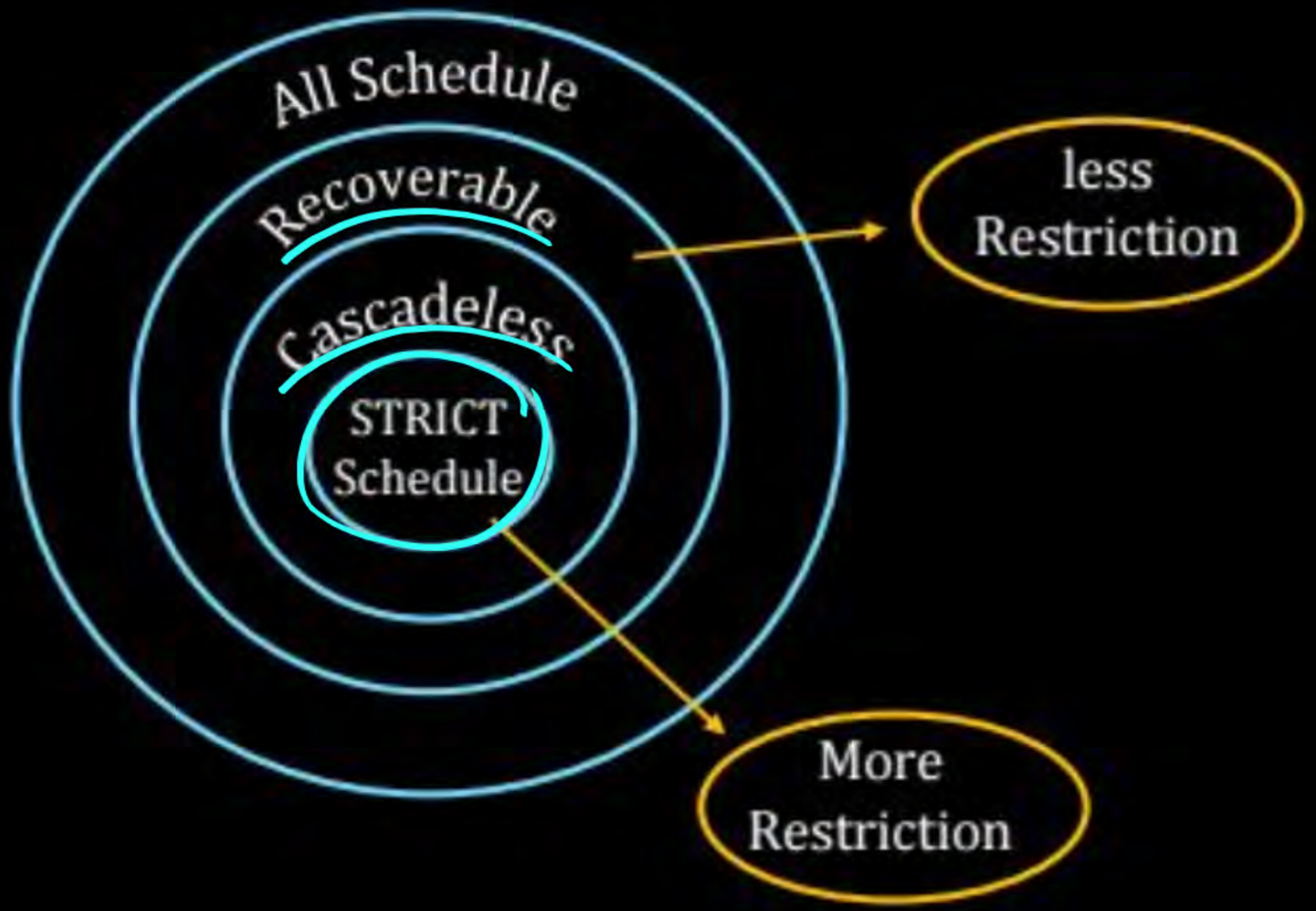
T_1	T_2
W(A)	
C|R	
	R(A)

Cascadeless

(3)

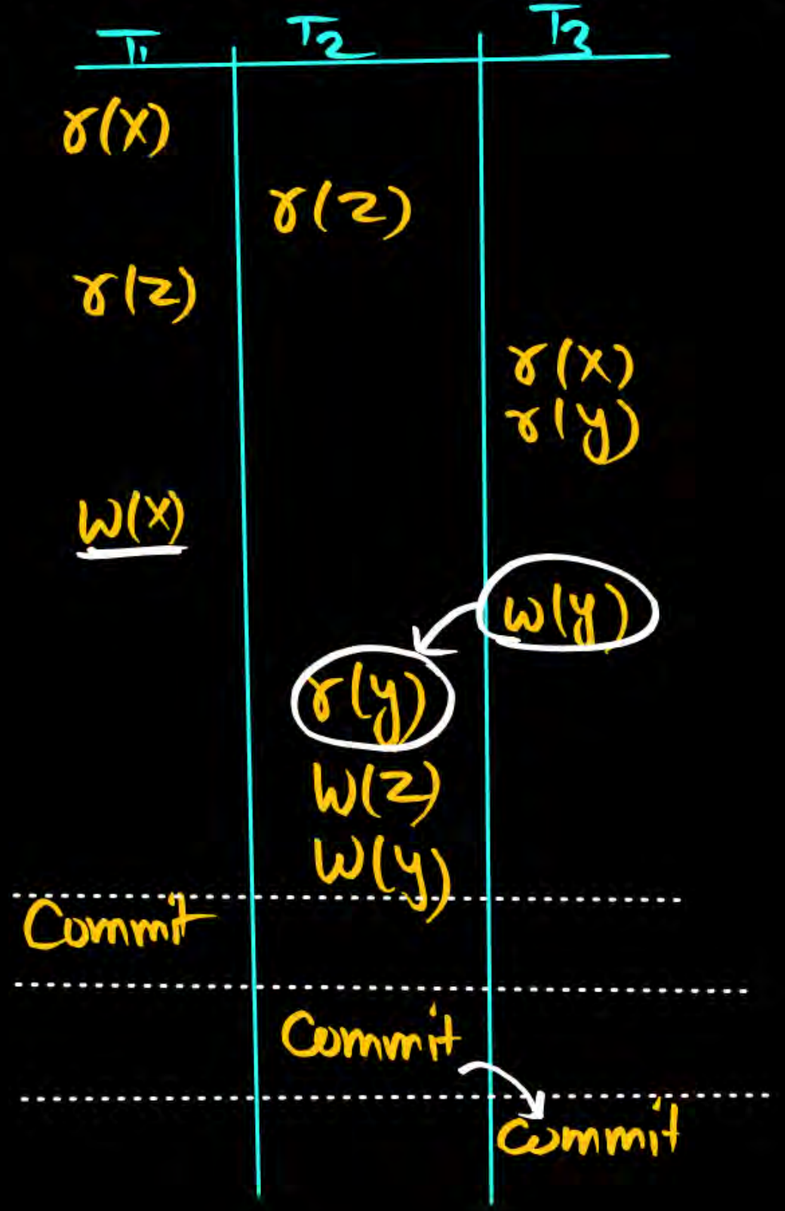
T_1	T_2
W(A)	
C|R	
	R(Q)
	W(Q)

Strict schedule


All Schedule
Recoverable
Cascadeless
STRICT
Schedule
less
Restriction
More
Restriction

Q.1 $r_1(x)r_2(z)r_1(z)r_3(x)r_3(y)w_1(x)w_3(y)r_2(y)w_2(z)w_2(y)\ C_1\ C_2\ C_3$

C_3 → Commit 3



✗ Recoverable

Not / IR / Non Recoverable

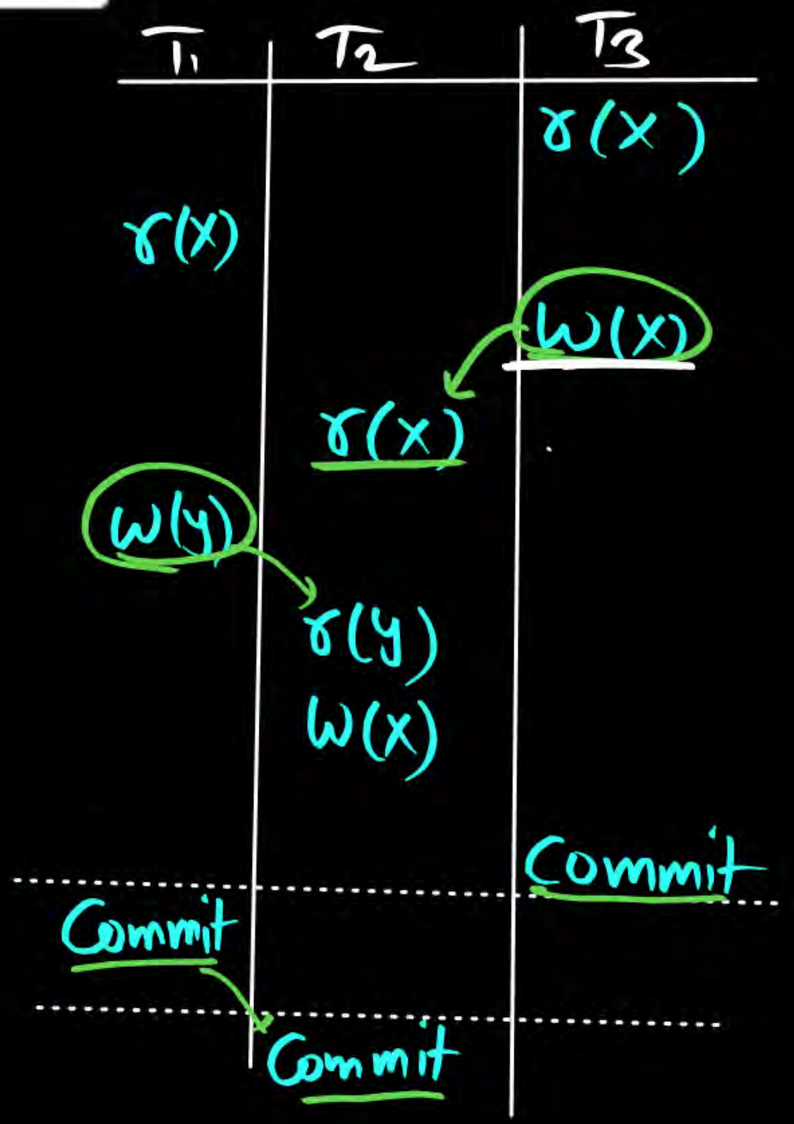
For each Data Item

Write operation

then Read

WR case

Q. 2 $r_3(x)\, r_1(x) w_3(x) r_2(x) w_1(y)\, r_2(y) w_2(x) C_3 C_1\, C_2$



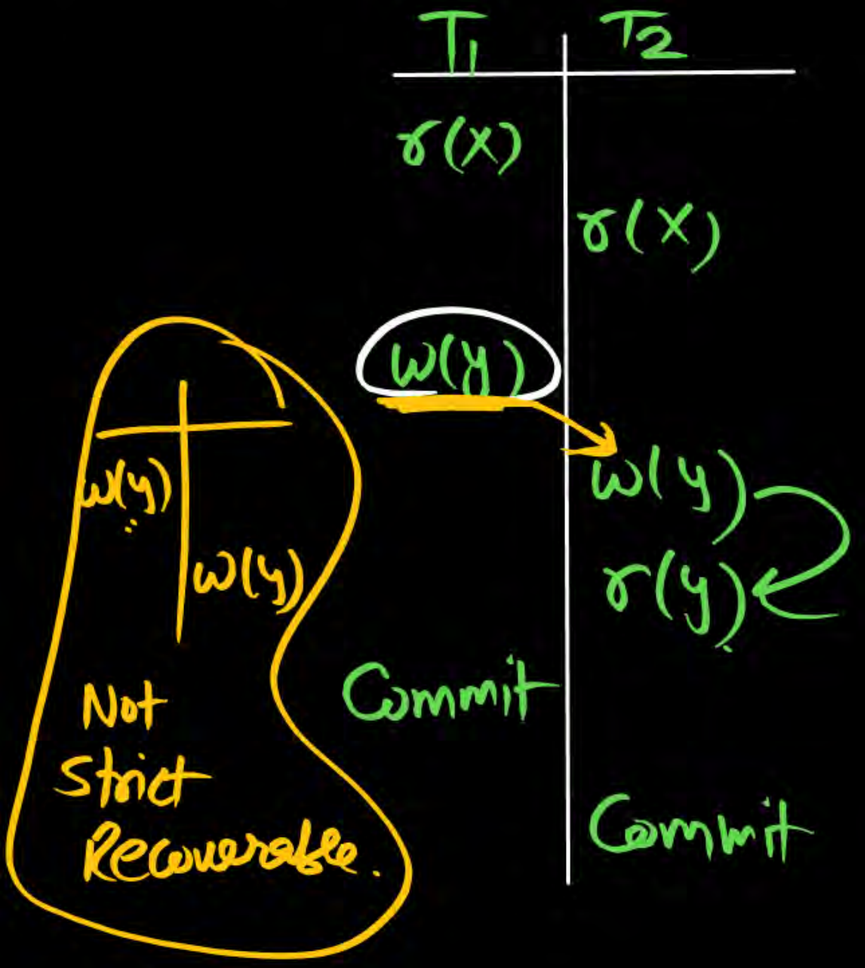
✓ Recoverable.

✗ Cascadeless

Bcz Uncommitted Read

So Only Recoverable

Q.3 $r_1(x)\ r_2(x) w_1(y) w_2(y) r_2(y)\ C_1\ C_2$



✓ Recoverable (NO WR case)

✓ Cascadeless
(Bcz No Uncommitted Read)

✗ STRICT Recoverable
NO

Only Recoverable & Cascadeless.

STRICT

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

then Strict Recoverable.

Q. 1

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. 2

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.

Q.3

Consider a simple checkpointing protocol and the following set of operations in the log.

(start, T4); (write, T4, y, 2, 3); (start, Tl);

(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

[MCQ: 2015–2M]

A Undo: T3, T1; Redo: T2

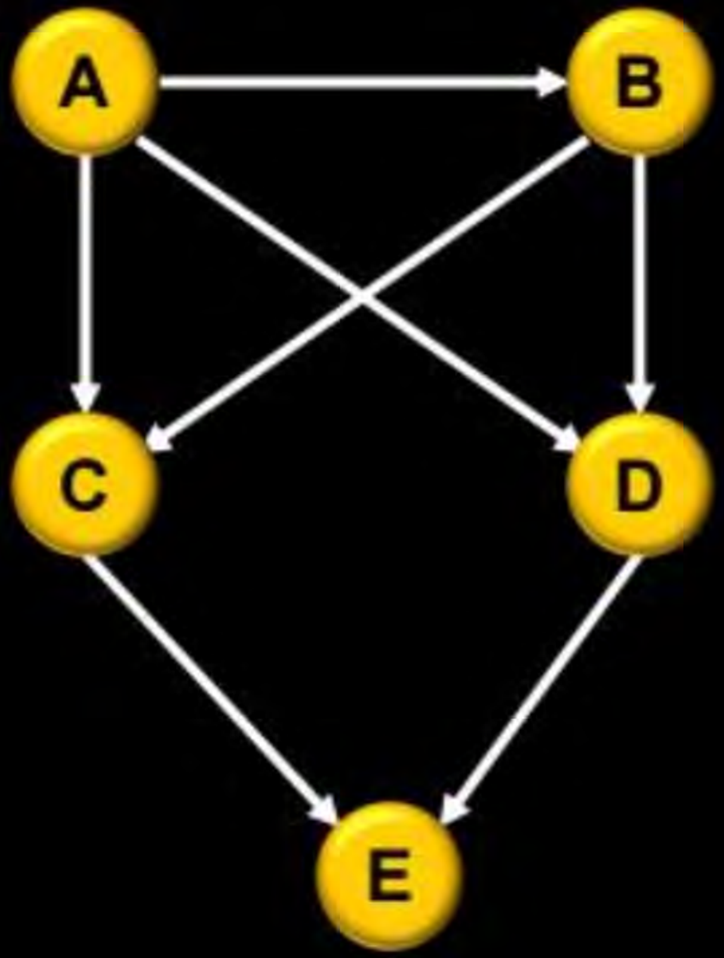
B Undo: T3, T1; Redo: T2, T4

C Undo: none; Redo: T2, T4, T3, Tl

D Undo: T3, Tl, T4; Redo: T2

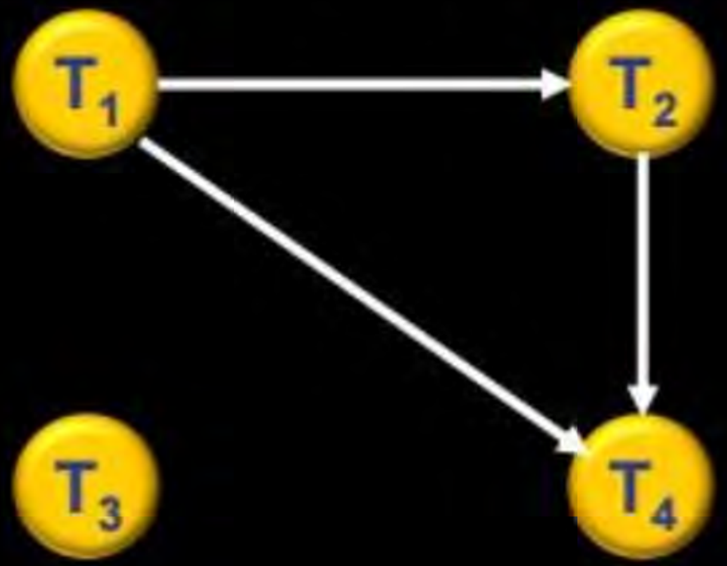
Topological Sorting

Q.4



Topological Sorting

Q.5



Topological Sorting

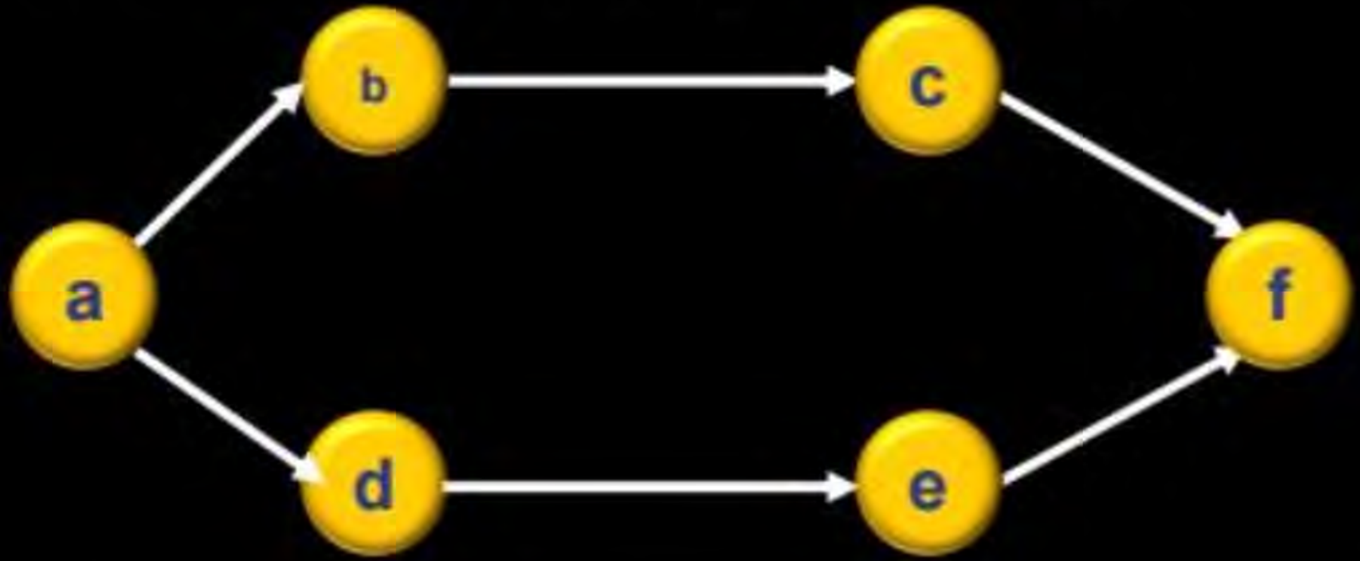
Q.6

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

Topological Sorting

Q.7

Consider the following directed graph:



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

[MCQ: 2016]

THANK YOU!