COMPUTER SCIENCE

Database Management System

Transaction & Concurrency Control

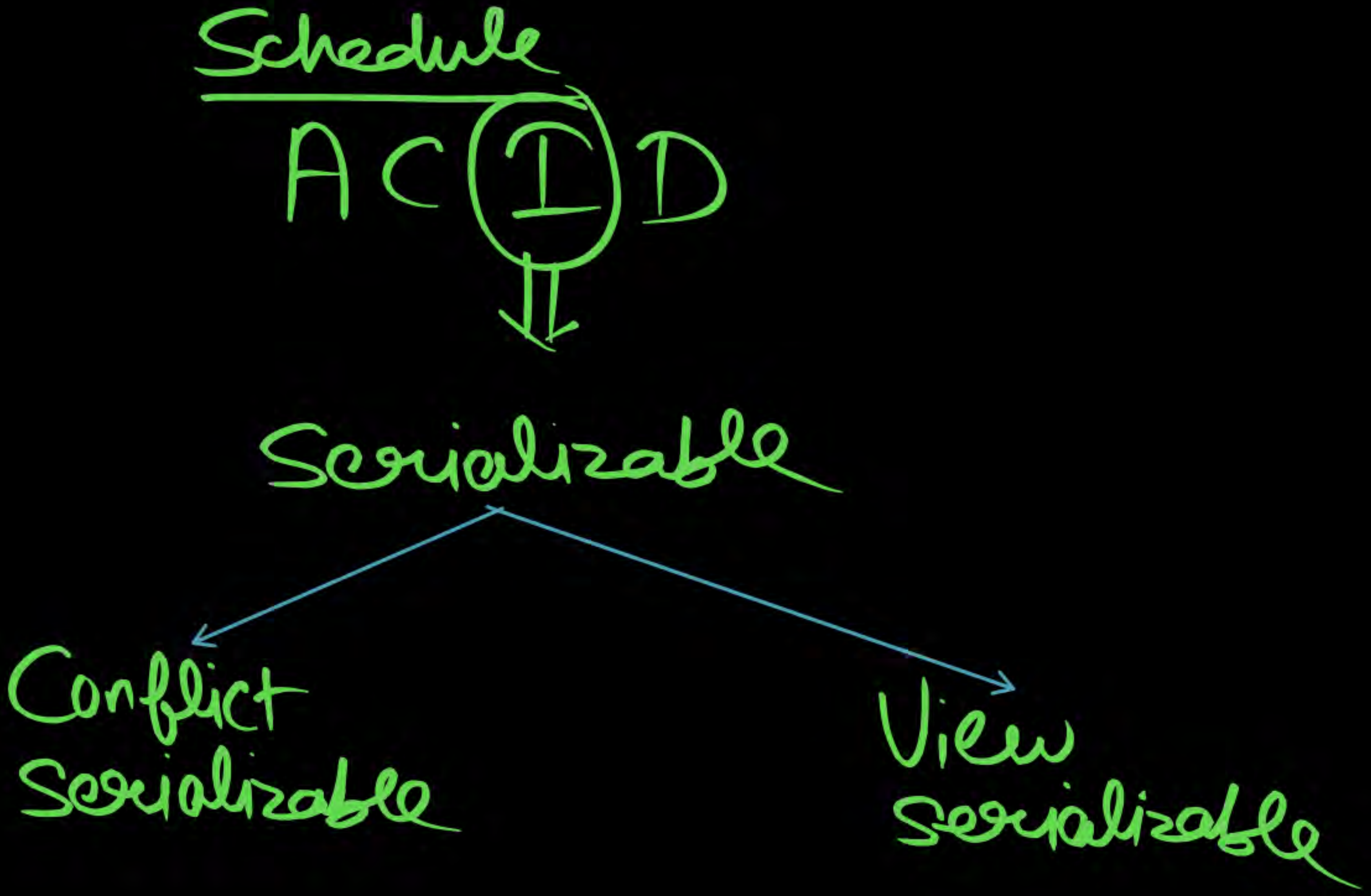
Lecture_5

Vijay Agarwal sir

TOPICS TO BE COVERED

01 Problem due to concurrent execution

02 Recoverable Schedule



Schedule

A C I D

Serializable

Conflict Serializable

View serializable

Problem due to Concurrent Execution.

① WR | Uncommitted Read | Dirty Read Problem.

② RW Problem / Un / Non Repeatable Read Problem.

③ WW Problem | Lost update Problem.

④ Phantom Tuple Problem.

R: Read.

W: write.

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

T_1	T_2
W(A)	
	R(A)

Uncommitted / Dirty Read : Here T_2 Read the value of Data Item (A) that is updated (written) by uncommitted Transaction T_1.

~~A=10~~

② RW Problem / Non/un Repeatable Read

T1	T2
R(A)	
	W(A)

eg: Library DB

```
R(A)
if (A > 0)
{
  A = A - 1
  W(A)
}
else
No Book Issue.
```

A=10

T1	T2
Read(A)	
if(A>0)	
{	
	Read(A)
	if (A > 0)
	{
	A = A - 1
	write(A)
	}
A=A-1	
write(A)	
}	

A=10

A=9

10-1=9

A=9

$$\begin{array}{l} 9 \text{ Book in Stock} \\ +2 \text{ Book Issue} \\ \hline 11 \text{ InConsistency} \end{array}$$

③ WW Problem | Lost Update Problem

T1	T2
W(A)	
	W(A)

$\boxed{A = 1000}$

	T1	T2	
A = 1000	Read(A)		
←	A = A − 100		
		Read(A) →	A = 1000
		temp = A * 0.1	temp = 100
		A = A + temp →	~~900~~
$\boxed{A = 900}$	write(A)		
		write(A)	$\boxed{A = 1100}$

Suppose the operation of transaction T_1, & T_2 in Such a Manner T_2 Read the Value of Account (A) Before T_1 Updates.

& T_2 update the Value of Account just after, T_1 update.

So whatever Update done by T_1 is overwritten by the Later transaction (So Loss of Updation of T_1 Transaction).

Lost Update Checking: If there are 2 (Two) write operation of Different Transaction & between these 2 write there is No Read operation, then Second (Later) transaction overwriteen the value of first Transaction Called Lost Update.

③ Phantom Tuple Problem

T1	T2
Select * From Employee WHERE Salary>4700	
	Insert into Employee Value <e5, E, 6700>

Select *
 FROM Employee
 WHERE Salary >4700	

eno	ename	Salary
e1	A	5000
e2	B	6000
e4	D	7000

eno	ename	Salary
e1	A	5000
e2	B	6000
e4	D	7000
e5	E	6700

e5 E 6700 ← Phantom Tuple.

Employee

eno	ename	Salary	
e1	A	5000	✓
e2	B	6000	✓
e3	C	4500	
e4	D	7000	✓
e5	E	6700	

Isolation

Durablity

& Atomicity → Trans. manager

Serializablity

↓

Consistent

- → Conflict Serializable
- → View Serializable

Recoverablity

↓

Recover if Any Case of failure

- → Recoverable schedule
- → Cascadeless Schedule
- → Strict Recoverable Schedule

Dependency

✓ (1)

T_1	T_2
W(A)	
⋮	R(A)
⋮	
Commit	

YES Dep.

(2)

T_1	T_2
W(A)	
commit	
	R(A)
	⋮

(3)

T_1	T_2
W(A)	
rollback	
	R(A)

(4)

T_1	T_2
W(A)	
	w(A)

(5)

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

✓ (6)

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

YES Dep.

Dependency

(1)

T_1	T_2
W(A)	
⋮	R(A)
⋮	
Commit	

YES its Dep.

(T_2 Depends on T_1)

Uncommitted | Dirty Read.

(2)

T_1	T_2
W(A)	
commit	
	R(A)
	⋮

No Dependency

A=500

(3)

T_1	T_2
W(A)	
rollback	
	R(A) → A=500

A=3700

UNDO ALL MODIFICATION

A: ~~3700~~

500

No Dependency

Dependency

Dirty Read
Uncommitted Read : Modified by one transaction & Read by another Transaction

No Dependency

(4)

T_1	T_2
W(A)	
	w(A)

No Dependency

(5)

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

same

No Uncommitted Read

YES Dep.
T_2 Depend on T_1
T_1 Depend on T_2

(6)

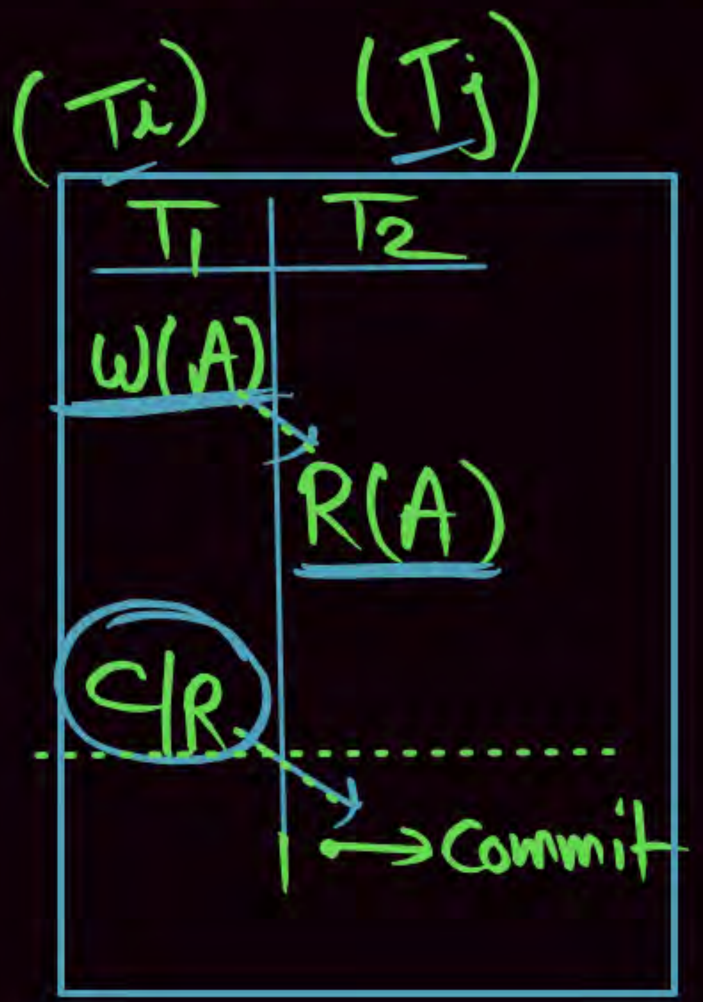
T_1	T_2
W(A)	
	W(B)
	R(A)
R(B)	

T1	T2
R(A)	
⋮	
W(A)	
	R(A)
	⋮
	Commit
failure	

500Cr.

Irrecoverable

Recoverable schedule



A Recoverable Schedule is One, 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 of T_i appear before Commit of T_j.

If T_2 Depands on T_1 then Commit of T_2 Must be Delay untill Commit/Rollback of T_1.

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)	

T8	T9
R(A)	
W(A)	
	R(A)
Commit	
	Commit

- □ 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

C: Commit

R: Rollback

✗ (1)

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

Irrecoverable

✗ (2)

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

Irrecoverable

✓ (3)

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

Recoverable

✓ (4)

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

Recoverable

✓ (5)

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

Recoverable

✓ (6)

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

Recoverable

Note

Recoverable Schedule May / May Not Free from

① WR / Uncommitted Read

② RW Poobleem

③ WW Poobleem

Cascading Rollback are bossible in Recoverable Schedule.

T1	T2
W(A)	
	R(A)
C/R	
	→ Commit

Recoverable Schedule

C : Commit

R : Rollback

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

Cascading Rollback.

T_1	T_2	T_3	T_4
W(A)			
	R(A)		
		R(A)	
			R(A)

Rollback

⊛ failure

T_2, T_3 & T_4 Depands on T_1

If T_1 fail the Rollback, Due to Dependency T_2, T_3 & T_4 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

Cascades schedule

(T_i) T_1	(T_j) 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 Previosuly written by T_i, then Commit of T_i appear before Read of T_j.

No UnCommitted 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.
- Note □ Every cascadeless schedule is also recoverable

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

Cascadeless Schedule

① No WR Problem
No Uncommitted (Dirty) Read

② No Cascading Rollback.

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

But Not Strict Recoverable

①

T1	T2
W(A)	
	R(A)
C/R	
	→Commit

Recoverable Schedule

②

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

Cascadeless Schedule.

③

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

Strict Recoverable.

- WW Problem
- RW Problem.

Strict Recoverable Schedule

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

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

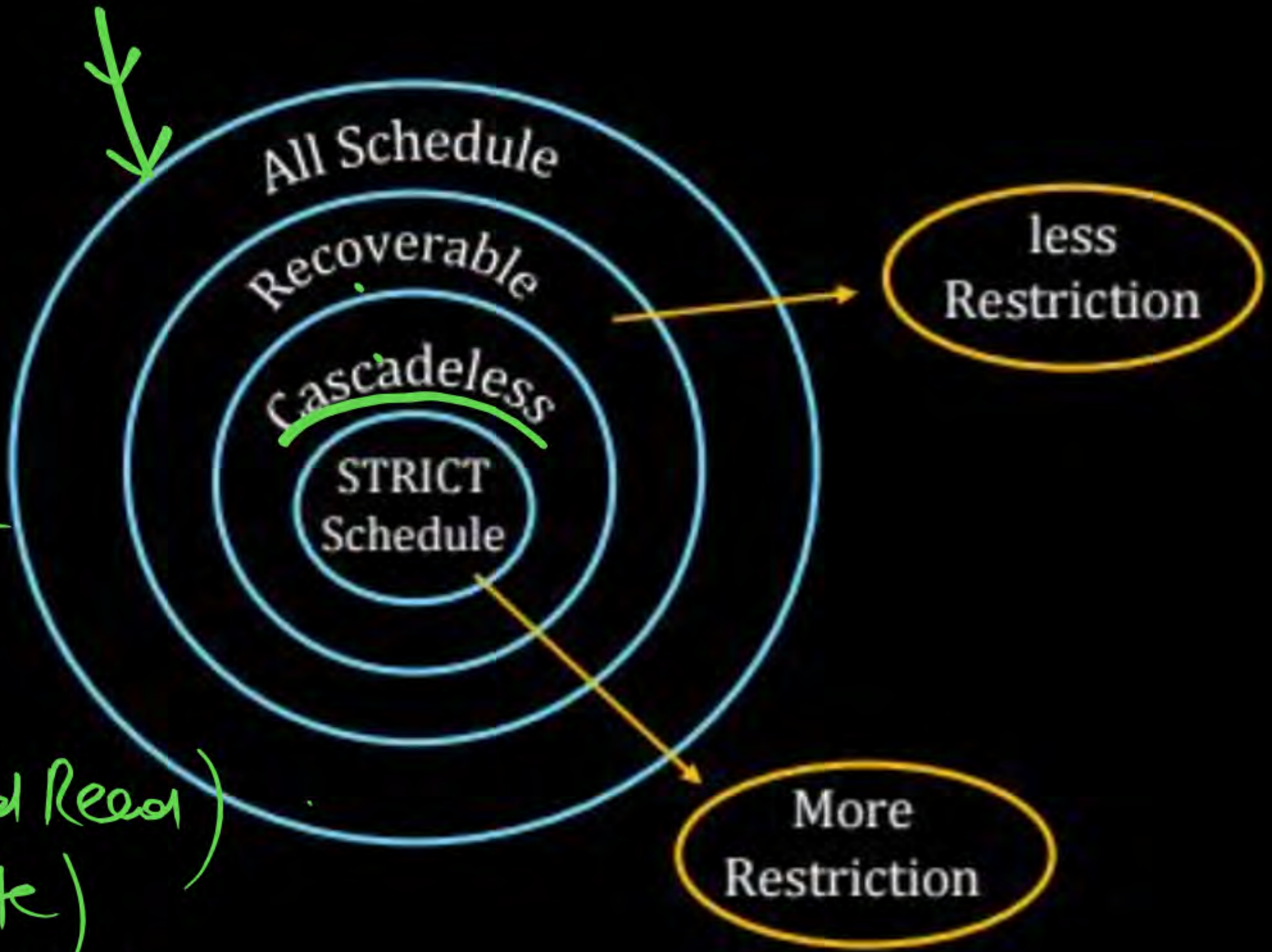
(3)

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

Strict
schedule

STRICT Recoverable

- → Cascadeless
- → Recoverable
- → No WR (Uncommitted Read)
- → No WW (Lost Update)
- → No Cascading Rollback.



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

T1	T2	T3
r(x)		
	r(z)	
r(z)		
		r(x)
		r(y)
w(x)		
		w(y)
	r(y)	
	w(z)	
Commit	w(y)	
	Commit	
		Commit

X Recoverable

Not/Ir/Non Recoverable

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

T1	T2	T3
		r(x)
r(x)		
		w(x).
	r(x).	
w(y)		
	r(y)	
	w(x)	
		Commit
Commit		
	Commit	

✓ Recoverable.

X Cascadeless (Uncommitted Read)

Only Recoverable.

T1	T2
W(A)	
	R(A)
C1	
	C2

← Recoverable

Cascadeless

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

W → R

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

T1	T2
r(x)	
	r(X)
w(y)	
	w(y)
	r(y)
Commit	
	Commit

No WR Case

✓ Recoverable

✓ Cascadeless

✗ Strict Recoverable

Cascadeless & Recoverable

No Cascading Rollback

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

Recoverable

Cascadeless

Strict Recoverable

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

- ✓ Recoverable
- ✓ Cascadeless
- ✓ Strict Recoverable

Q.

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. 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.

MCQ **Q.15**

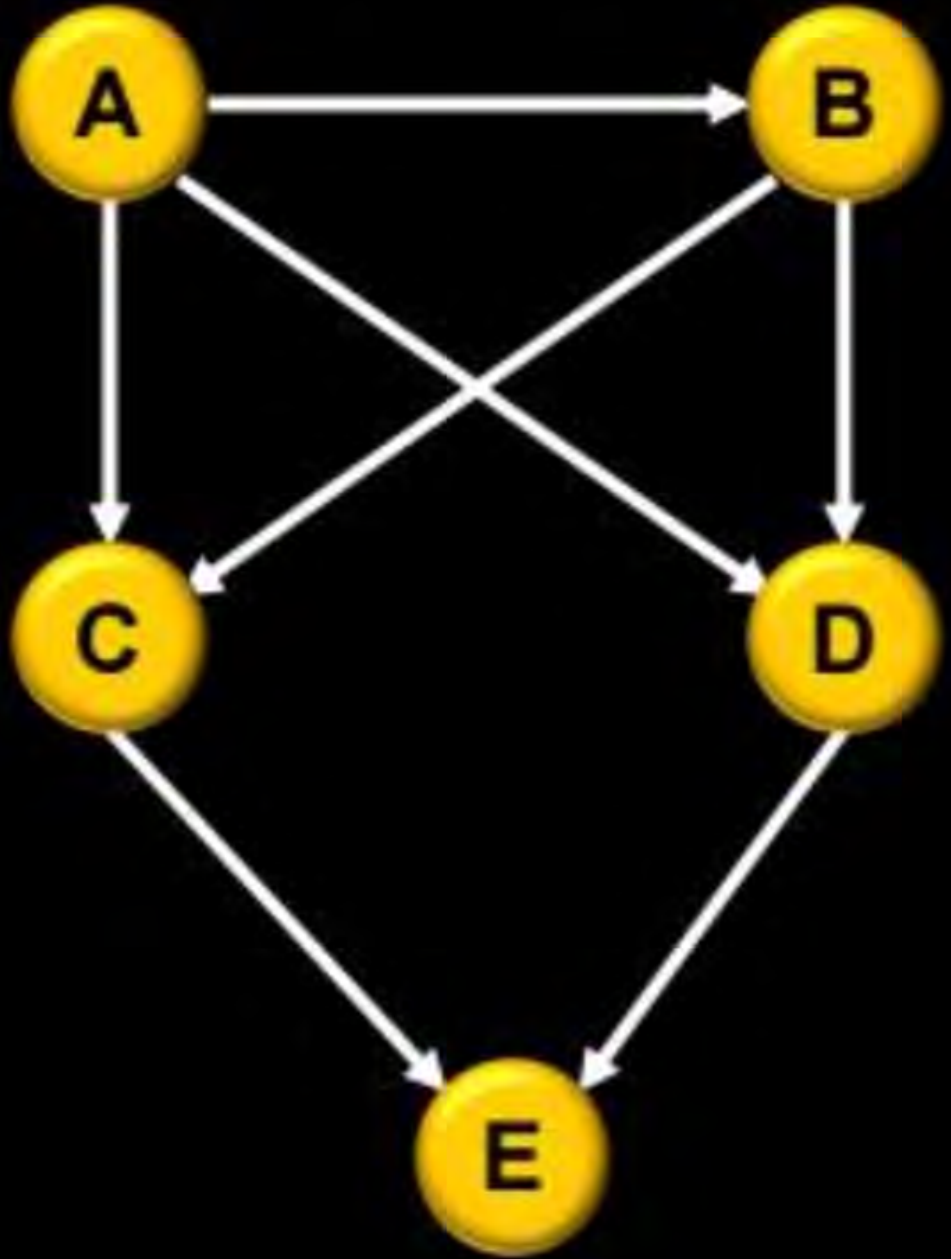
Consider a simple checkpointing protocol and the following set of operations in the log.
(start, T4); (write, T4, y, 2, 3); (start, T1);
(commit, T4); (write, T1, z, 5, 7);
(checkpoint);
(start, T2); (write, T2, x, 1, 9); (commit, T2);
(start, T3); (write, T3, z, 7, 2);
If a crash happens now and the 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]

- (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

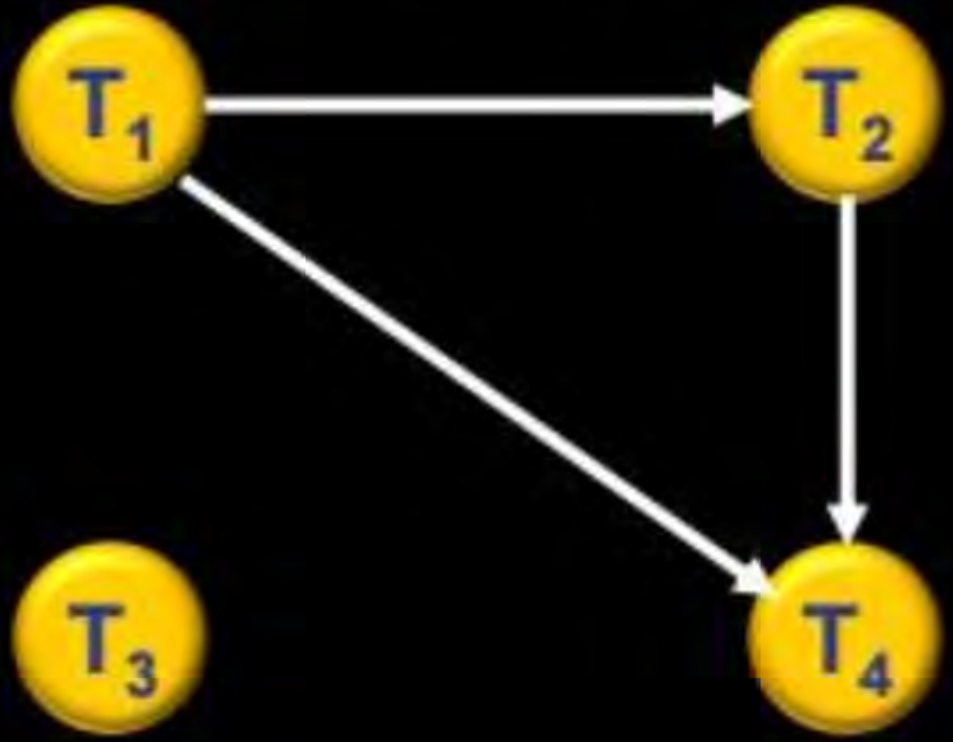
Topological Sorting

Q.



Topological Sorting

Q.

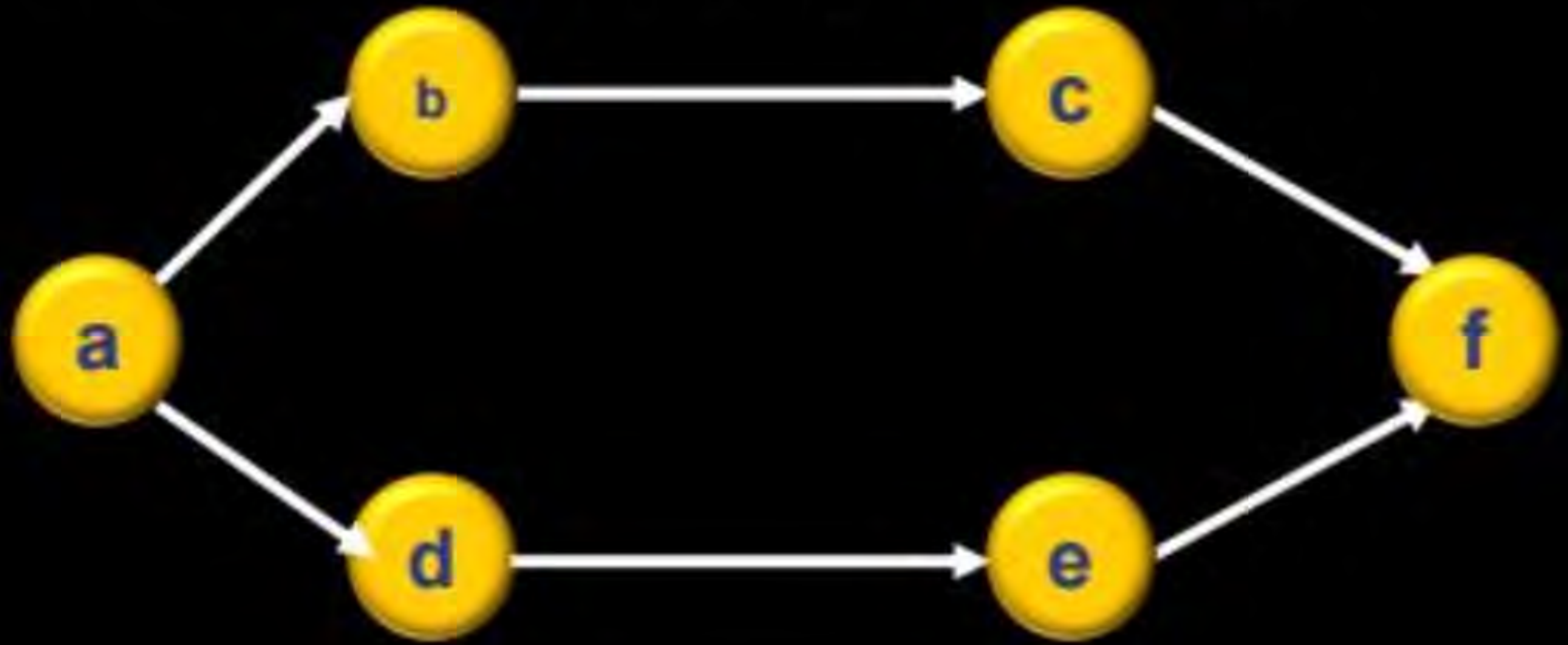


Topological Sorting

Q. $R_4(x)\ R_2(x)\ R_3(x)\ W_1(y)\ W_2(x)\ R_3(y)\ W_2(y)$

Topological Sorting

Q. Consider the following directed graph:



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

[MCQ: 2016]

THANK YOU!