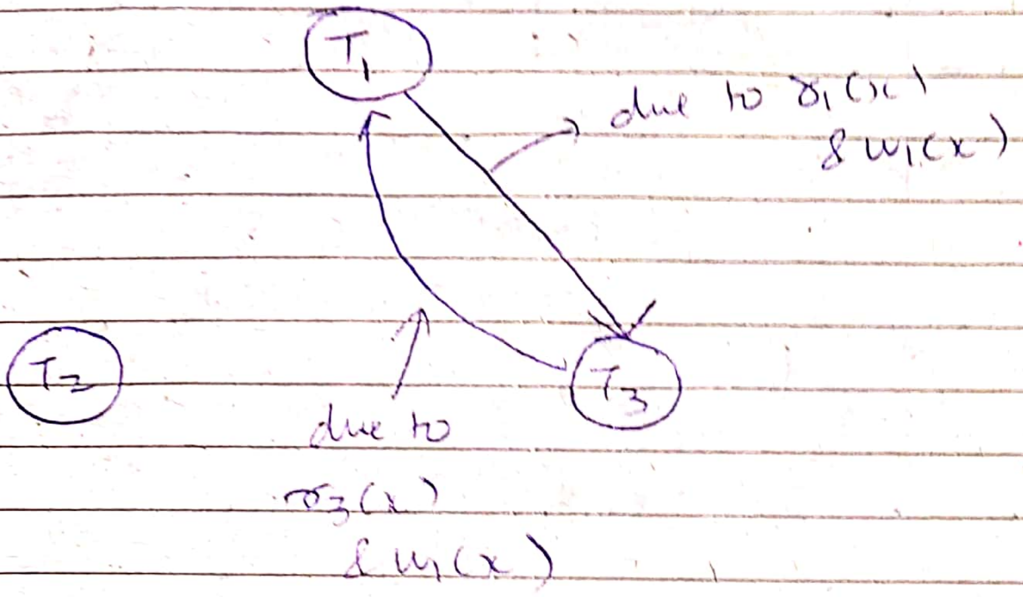
Tut-13

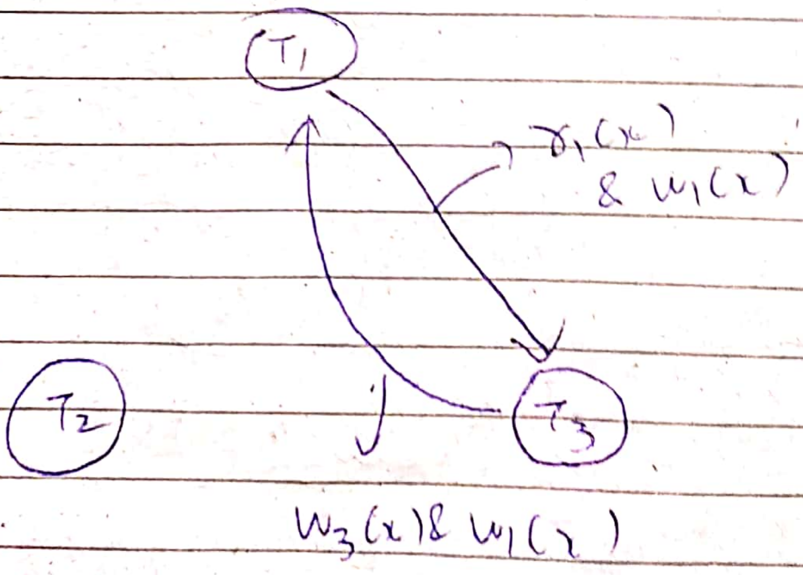
1)

a)



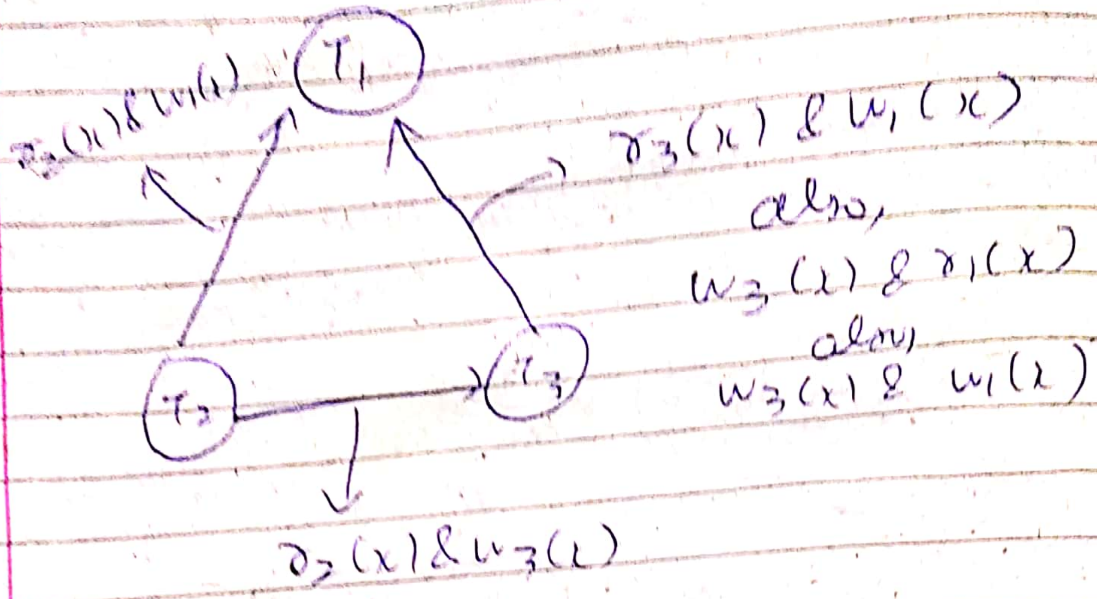
So, Non-conflicty As cycle is there

b)



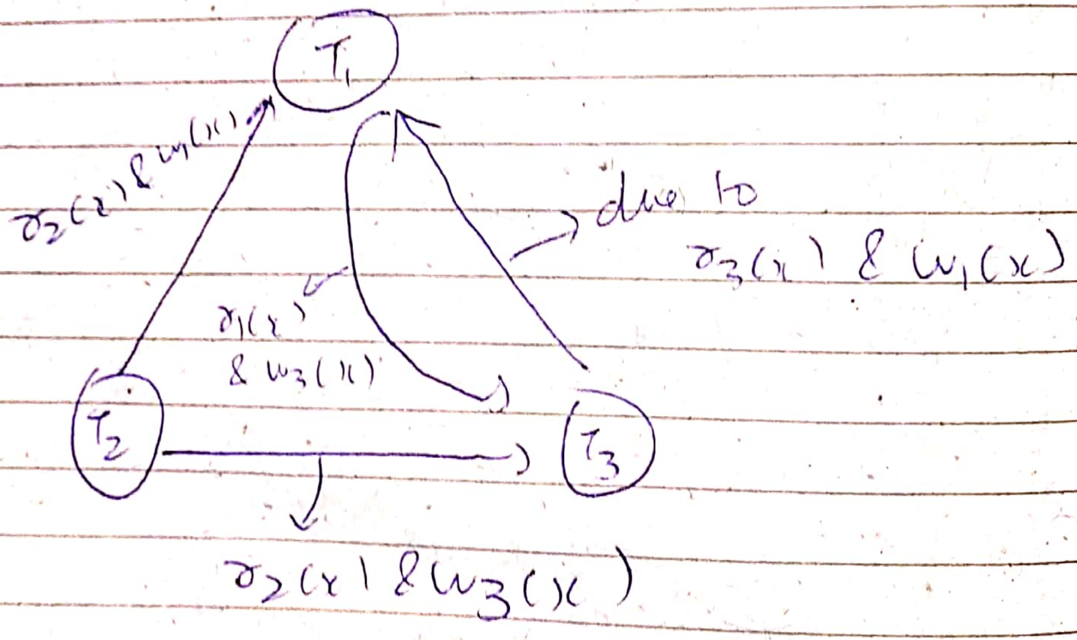
So, Non conflicting

c)



No, cycle
⇓
So, Conflict Serializable ✓

d)



As ~~graph~~ cycle exist
So, Non-Serializable

2)

S₃

a)

T_1	T_2	T_3
$R_1(X)$		
	$R_2(Z)$	
$R_1(Z)$		
		$R_3(X)$
		B$R_3(Y)$
$W_1(X)$		
C_1		
		$W_3(Y)$
		C_3
	$R_2(Y)$	
	$W_2(Z)$	
	$W_2(Y)$	
	C_2	

No, dirty Read

↳ So, Recoverable & Cascadeless

Strict Schedule

↳ As No Read/write operation is here which occurs ~~at~~ after write without Commit b)

T_1	T_2	T_3
$r_1(x)$		
	$r_2(z)$	
$r_1(z)$		
		$r_3(x)$
		$r_3(Y)$
$W_1(X)$		
		$W_3(Y)$
	$r_2(Y)$	
	$W_2(Z)$	
	$W_2(Y)$	
C_1		
	C_2	
		C_3

Non Cascadeless

↳ As dirty Read between $W_3(Y)$ & $R_2(Y)$

Non Recoverable

↳ As C_3 Commit should be done before C_2 as there is dirty read between $W_3(Y)$ & $R_2(Y)$

but here C_2 is Committed ~~off~~ before C_3

c)

T_1	T_2	T_3
$R_1(z)$		
	$R_2(z)$	
		$R_3(x)$
$R_1(z)$		
	$R_2(z)$	
		$R_3(Y)$
$W_1(x)$		
C_1		
	$W_2(z)$	
		$W_3(Y)$
$W_2(Y)$		
		C_3
	C_2	

Cascadeless & Recoverable

⇓

As No dirty Read

Not Strict Schedule

⇓

Because $W_3(Y)$ & $W_2(Y)$ occurs

even when $W_3(Y)$ was not Commited

So, write operation is there before commit

3) For Schedule

S1

a) By Wait-Die Policy

1) T_1 acquires Shared lock on X

2) T_2 ~~ask~~ ask exclusive lock
but as $T_2 > T_1 \Rightarrow T_2$ rollback

3) T_3 gets exclusive lock for Y

4) T_1 ask for exclusive lock on Y

As $T_3 > T_1 \Rightarrow$ So, T_1 waits

5) T_3 Commit & releas lock on Y

6) T_1 gets Exclusive lock on Y

7) T_1 Commits

8) T_2 get exclusive lock on X & Y
& then Commit

Date / /

Page No.

~~b) 1) T₁ gets shared lock on X~~
~~2) T₂ gets exclusive lock on Y~~

b) 1) T_1 gets shared lock on X.

2) T_1 waits for T_1 to get exclusive lock on X

3) T_3 gets exclusive lock on Y

4) T_1 waits for T_3

5) T_3 Commit. & release lock

6) T_1 gets exclusive lock on Y

7) T_1 Comits

8) T_2 gets exclusive lock on X & Y

9) T_2 Comits

No Deadlock

c) T_1 gets lock on X & Y & commits
T_2 gets lock on X & Y & commits

T_3 get lock on Y & commits

<u>For Schedule S_2</u>

a) By Wait - Die policy

1) T_1 acquires shared lock on X

2) T_2 acquires exclusive lock on Y

3) T_2 ask exclusive lock for Y

but as $T_2 > T_1$ then T_2 roll back

4) T_3 acquires exclusive lock on Y.

5) T_1 ask lock for Y but as $T_3 > T_1$ so, T_1 waits

6) T_3 Commits

7) T_1 get exclusive lock on Y & Commit

8) T_2 get exclusive lock on X & Y => Commit.

Sol

b) 1) T_1 gets a shared lock on Y

2) T_2 waits to get lock on X

3) T_2 get exclusive lock on Y

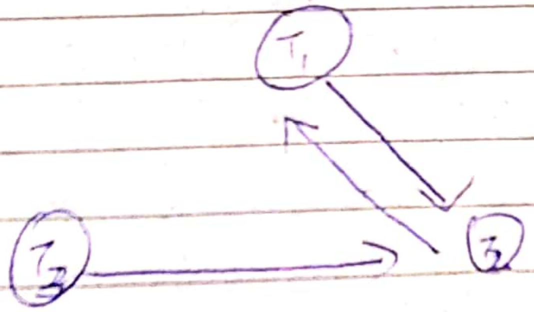
4) T_3 waits for T_2 to get exclusive lock on Y

5) T_1 waits for T_2 to get exclusive lock on Y

As T_1 waits for T_2 & T_2 waits ~~of~~ for T_1

⇓

deadlock



wait - for graph

c) T_1 gets lock on X & Y, then Commit
T_2 gets lock on X & Y, then Commit
T_3 gets lock on Y & then Commit