USN: 01fe18bcs211
Roll : 235

2)

i) r1(x)
r3(x)
w1(x)
r2(x)
w3(x)

Conflict operations:-

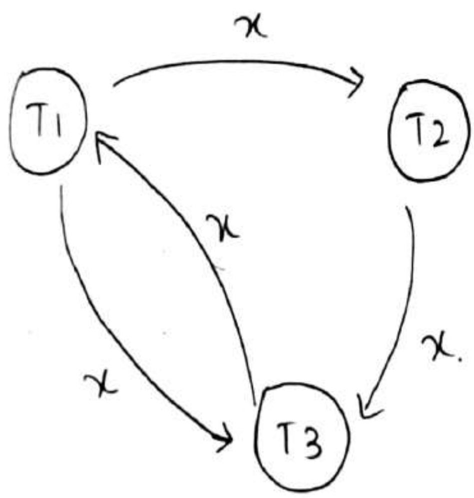
r1(x) ; w3(x)

r3(x) ; w1(x)

w1(x) ; r2(x)

w1(x) ; w3(x)

r2(x) ; w3(x)

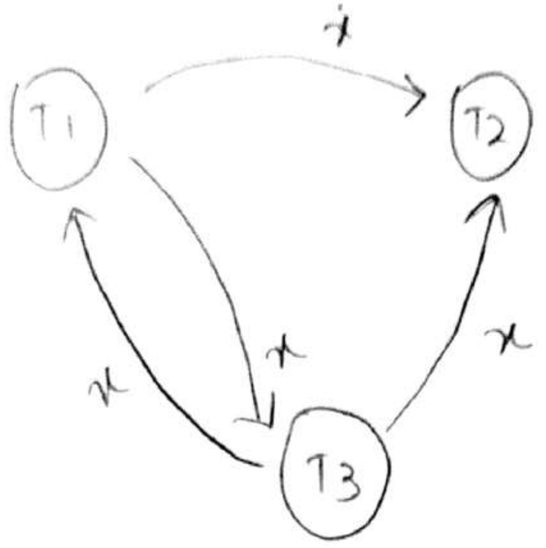


It is not conflict serializable. As there is a cycle.

ii) r1(x) ; r3(x) ; w3(x) ; w1(x) ; r2(x)

Conflict operations : r1(x) w3(x)
r3(x) w1(x)
w3(x) w1(x)
w3(x) r2(x)
w1(x) r2(x)

Scanned with CamScanner

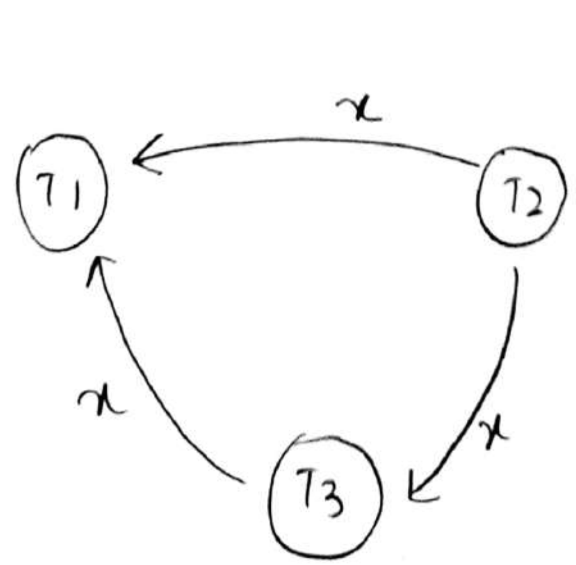


It is not conflict serializable As the precedence graph has cycles

iii) $r3(x)$; $r2(x)$; $w3(x)$; $r1(x)$; $w1(x)$

Conflict operations:

$r3(x)$	$w1(x)$
$r2(x)$	$w3(x)$
$r2(x)$	$w1(x)$
$w3(x)$	$r1(x)$



Equivalent Serial Schedule:

T2, T1, T3 or

231

This is conflict serializable

Scanned with CamScanner

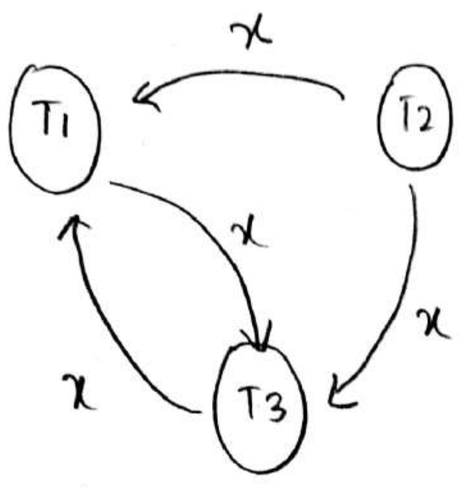
iv) $r3(x)$; $r2(x)$; $r1(x)$; $w3(x)$; $w1(x)$

Conflict operations: $r3(x)$; $w1(x)$

$r2(x)$; $w3(x)$

$r2(x)$; $w1(x)$

$r1(x)$; $w3(x)$



This is not conflict serializable

1) (i) Does F2 cover F1

$A^+ = \{ ACD \}$

$AC^+ = \{ ACD \}$

$E^+ = \{ EADFC \}$

with respect to F1

$A^+ = \{ ACD \}$

$AC^+ = \{ ACD \}$

$E^+ = \{ EAFCD \}$

with respect F2.

So all fd's of f1 is covered in f2

Scanned with CamScanner

(ii) Equivalent

In (i) F2 covers F1

So

F1 covers F2.

$A^+ = \{ACD\}$

$E^+ = \{EAFCD\}$

with respect to F2

$A^+ = \{ACD\}$

$E^+ = \{EADFC\}$

with respect to F1

So F1 covers F2.

F2 covers F1 proved in (i)

So they are equivalent

Scanned with CamScanner

3)

(i) Write - read conflict (dirty read problem)

$S1 = r2(x), r2(y), \underline{w2(x), r1(x)}, r1(y)$

$w1(x), w2(y)$

(ii) Read - Write conflict (Unrepeatable read)

$S2 = r2(x), r2(y), w2(x), r1(x), \underline{r1(y), w2(y)}, w1(x)$

(iii) Write - Write Conflict

$S3 = r2(x), r2(y), r1(x), r1(y), \underline{w2(x), w1(x)}, w2(y)$

(iv) If we use 2PL the locking and unlocking will be done in two phase i.e growing and shrinking phases respectively and all the exclusive locks will be held till the transaction commits or aborts and shared locks can be released any time during the second phase. If we apply 2PL, only serializable schedules will be allowed and all the three schedules above will be disallowed.

Scanned with CamScanner

x → denotes exclusive lock

s → denotes shared lock

c → denotes commit.

u → unlock. So the execution of following schedules where strict 2PL ensure serializibility by not granting locks which may create conflicts.

S1 = s2(x), r2(x), s2(y), r2(y), x2(x), w2(x), s1(x)-request not granted, x2(y), w2(y), u2(x), u2(y), c2, s1(x), r1(x), s1(y), r1(y), x1(x), w1(x), u1(x), u1(y), c1

S2 = s2(x), r2(x), s2(y), r2(y), x2(x), w2(x), s1(x)-request not granted, x2(y), w2(y), u2(x), u2(y), c2, s1(x), r1(x), s1(y), r1(y), x1(x), w1(x), u1(x), u1(y), c1

S3 = s2(x), r2(x), s2(y), r2(y), s1(x)-request not granted, x2(x), w2(x), x2(y), w2(y), u2(x), u2(y) c2, s1(x), r1(x), s1(y), r1(y), x1(x), w1(x), u1(x), u1(y), c1

Scanned with CamScanner

Name: Sneha . K. Majjigudda
USN: 01fe18bcs211
Roll : 235

2)

i) There are two possible executions : T1T2 and T2T1

Case1:

	A	B
initially	0	0
after T1	0	1
after T2	0	1

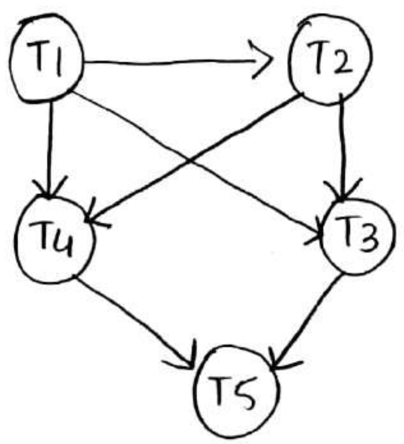
Consistency met
$A = 0 \vee B = 0 \equiv T \vee F = T$

Case 2 :

	A	B
initially	0	0
after T2	1	0
after T1	1	0

Consistency met : $A = 0 \vee B = 0 \equiv f \vee T = T$

ii) Any interleaving of T1 and T2 results in a non-serializable schedule



Scanned with CamScanner

T1	T2
read (A)	
	read(B)
	read (A)
read (B)	
if A=0 then B=B+1	
	if B=0 then A:=A+1
	write(A)
write (B)	

iii) There is no parallel execution resulting in a serializable schedule. From (i) we know that a serializable schedule results in $A=0 \vee B=0$

Suppose we start with T1 read(A). Then when the schedule ends, no matter when we run the steps of T2, B=1

Now suppose we start executing T2 prior to completion of T1. Then T2 read (B) will give B a value of 0. So when T2 completes A=1

Scanned with CamScanner

Thus $B = 1 \wedge A = 1 \rightarrow \neg(A = 0 \vee B = 0)$

Similarly for starting with T2 read (B)

3)

i) T1

```
lock_s(A)
read(A)
lock_x(B)
read(B)
if A = 0
then B := B + 1
write(B)
unlock(A)
unlock(B)
```

→ lock_s(x) stands for shared lock on x

→ lock_x(x) stands for exclusive lock on x

T2:

```
lock_s(B)
read(B)
lock_x(A)
read(A)
if B = 0
then A := A + 1
write(A)
unlock(B)
unlock(A)
```

Scanned with CamScanner

Here there are two types of locks shared and exclusive represented by lock_s(dataitem) and lock_x (dataitem) respectively.

Locking happens only in growing phase and unlocking happens in shrinking phase.

So serializability is guaranteed

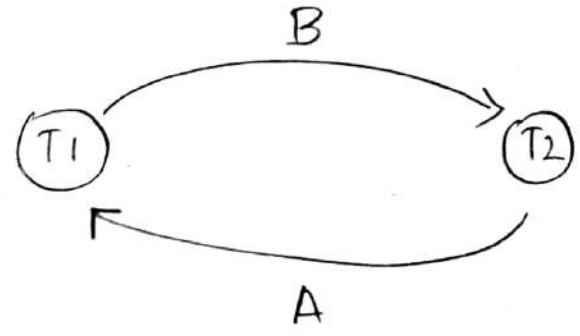
ii) yes the execution of these transaction may result in deadlock. Consider the schedule shown below

T1	T2
lock_s(A)	
	lock_s (B)
	read(B)
read(A)	
lock_x(B)	
(waits for T2 to release lock B)	
	lock_x(A)
	(waits for T1 to release lock A)

So there is a chance that deadlock occurs

If we use conservative 2PL then deadlock can be avoided.

Scanned with CamScanner



Hence deadlock.

1)

i) Consider the set of FD : $AB \rightarrow CD$ and $C \rightarrow D$. AB is obviously a key for this relation since $AB \rightarrow CD$ implies $AB \rightarrow ABCD$. It is a primary key since there are no smaller subsets of keys that hold over R(A, B, C, D). The FD : $C \rightarrow D$ violates 3NF but not 2NF Since

- $D \in C$ is false ; that is, it is not a trivial FD
- C is not a super key
- D is not part of some key R

ii) Candidate key : {B, C, E}

Decomposition:

From FD1 : R3a = {B, C, A, D}

From FD2 : R3b = {E, F}

Scanned with CamScanner

From FD3 : R3c = {F, G, H}

Since no relation contains the candidate key, additionally create a relation R3d = {B, C, E}.

Now R3a, R3b, R3c, R3d are in 3NF and BCNF

Scanned with CamScanner