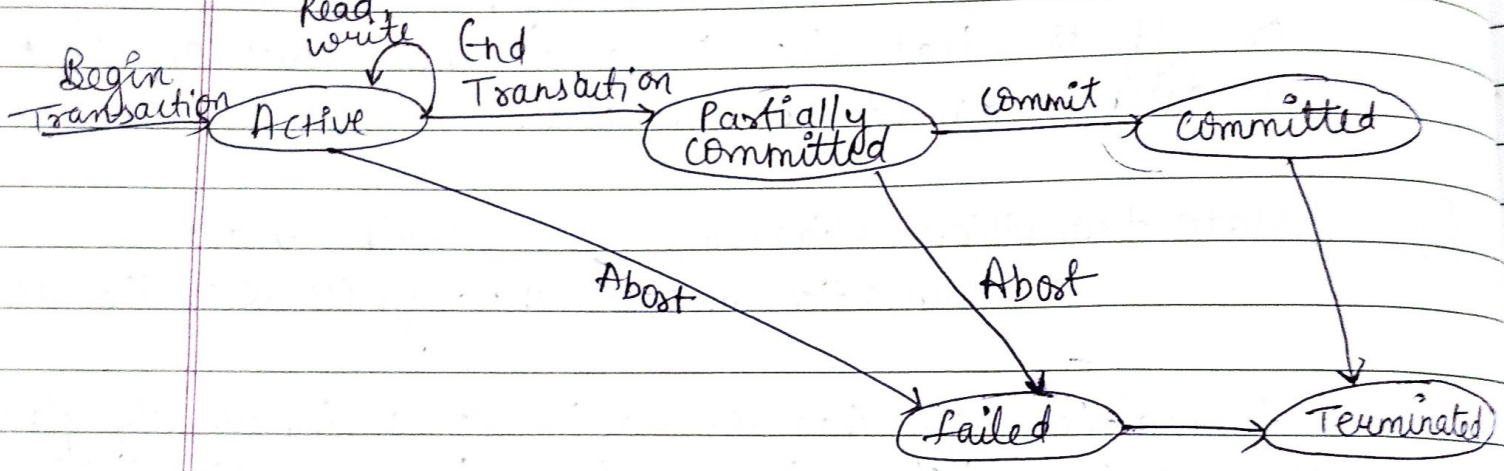
State-transition diagram :- It describes how a transaction moves through its execution states.

A transaction goes into an active state immediately after its execution, where it can issue "read" and "write" operations. When the transaction ends, it move to the partially committed state. At this point some recovery protocols need to ensure that a system failure will not result in an inability to record the changes of the transaction permanently. Once this check is successful, the transaction is said to have reached its commit point and enters the committed state. Once the T is committed, it has concluded its execution successfully and all its changes must be recorded permanently in the database. However, a transaction can go to the failed

state if one of the checks fails or if the T is aborted during its active stage. The transaction may then have to be rolled back to undo the effect of its "write" on the database. The terminated state corresponds to the T leaving the system. The transaction information that is maintained in system tables while the T has been running is removed when the T terminates. failed or aborted T may be restarted later automatically or by user.

((T = transaction))



Desirable Properties of Transactions. | ACID Properties

Atomicity - A transaction is an atomic unit of processing, it is either performed in its entirely or not performed at all.

Consistency preservation → A transaction is consistency preserving if its complete execution takes the database from one consistent state to another. It is the responsibility of the programmer

Isolation - A transaction should appear as though it is being executed in isolation from other transactions. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.

Durability or permanency - The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

Level of Isolation

Level (0) isolation → A transaction is said to have level 0 if it does not overwrite the dirty read of higher level transaction

Level (1) isolation → It has no lost updates.

Level (2) isolation → It has no lost updates and no dirty reads

Level (3) isolation / true isolation → It has in addition to degree 2 properties, repeatable reads.

The System Log - To be able to recover from failures that affect transactions, the system maintains a log, to keep track of all transaction operations that affect the values of database items.

3 Schedules (or History) - When transactions are executing concurrently in an interleaved fashion, then the order of execution of operations from various transactions is known as schedule (or history).

Types of schedule based on recoverability properties.

(1) Complete schedule - A schedule S of n transactions $T_1, T_2, \ldots T_n$ is said to be complete schedule if the conditions hold
   (a) The last operation is either abort or commit
   (b) For any pair of operations from the same transaction T_i, their order of appearance in S is the same as their order of appearance in T_i
   (c) For any two conflicting operations, one of the two must occur before the other in the schedule.

Ex- R1(A) R2(A) W1(A) commit W2(A) abort

(2) Recoverable and non-recoverable Schedule.
A schedule S is recoverable if no transaction T in S commits until all transactions T' that have written some item X that T reads have committed.

Ex- S1: R1(A) W1(A) R2(A) W2(A) R1(B) W1(B) commit 1 commit 2 → Recoverable
S2: R1(A) W1(A) R2(A) W2(A) commit 2 R1(B) RB W1(B) commit 1 → Non Recoverable

(3) Cascading Abort / Cascading Rollback - If one transaction failure causes multiple transaction to rollback, it is called cascading rollback.

Date / /
Page No.
Shivalal

4 Cascadeless schedule or to avoid cascading rollback - If every transaction in the schedule reads only items that were written by committed transactions. In this case, all items read will not be discarded, so no cascading rollback will occur.

Ex- S : $r_1(X)$ $w_1(X)$, $r_1(Y)$, $w_2(X)$, $w_1(Y)$, C_1 $R_2(X)$, C_2

5 Strict schedule - In which transactions can neither read nor write by other transactions until the transaction is either commits or aborts
All strict schedule are cascadeless schedules.
Ex- S : $r_1(X)$ $w_1(X)$ $r_1(Y)$ $w_1(Y)$ C_1 $r_2(X)$ $w_2(X)$ C_2

6 Serial schedule -
⇒ If we don't have any interleaving then that type of schedule is called serial schedule.
⇒ When transactions are executing serially that always ensure the consistency
⇒ If there are n - transactions in a schedule the possible number of serial schedule is n!,
Ex -
S : R1(A) W1(A) R2(A) W2(A)

Conflict : Two operations in a schedule are said to be conflict if they satisfy all three condition
(1) they belong to different transactions
(2) they access the same item X
(3) at least one of the operations is a write_item(X).

Q. Conflict equivalent and conflict serializable -

Two schedules are said to be conflict equivalent if the order of any two conflicting operations is the same in both schedules.

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

(A)

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

(B)

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

(C)

Conflicting operations

$S_{(A)}$ = R(A) W(A)
W(A) R(A)
W(A) W(A)

$S_{(B)}$ = R(A) W(A)
W(A) R(A)
W(A) W(A)

$S_{(C)}$ = R(A) W(A)
W(A) R(A)
W(A) W(A)

Since, the order of conflicting operations is same
So, it is conflict serializable

Testing conflict serializability of a schedule S

⇒ For each transaction T_i participating in schedule S, create a node labelled T_i in the precedence graph.

⇒ for each case in S where T_j executes a R(X) after T_i executes a W(X), ~~create an edge~~ or W(X) after T_i executes a R(X) or W(X) after T_i executes a W(X) create an edge ($T_i \rightarrow T_j$) in the precedence graph

⇒ The schedule s is serializable if and only if the precedence graph has no cycles.

Equivalent serial Schedule

⇒ Check which node has not incoming ~~node~~ edge
⇒ Delete the outgoing edge from that node and delete itself. Repeat this and noted the order

View Equivalence and view serializability.

Two schedules S and S' are said to be view equivalent if the following 3 conditions hold:-

⇒ If T_i reads initial value of 'A' in S, then T_i should also read initial value of data item A in S'
⇒ If T_j produces final write operation of A in S then T_j should also perform the final write operation of A in S'

5 Given $r1(X)$; $r3(X)$; $W1(X)$; $r2(X)$; $W3(X)$;
$r1(X)$; $r3(X)$; $W3(X)$; $W1(X)$; $r2(X)$;
$r3(X)$; $r2(X)$; $W3(X)$; $r1(X)$; $W1(X)$
$r3(X)$; $r2(X)$; $r1(X)$; $W3(X)$; $W1(X)$

(1)

T_1	T_2	T_3
$r(X)$		
		$r(X)$
$W(X)$		
	$r(X)$	
		$W(X)$

(2)

T_1	T_2	T_3
$r(X)$		
		$r(X)$
		$W(X)$
$W(X)$		
	$r(X)$	

(3)

T_1	T_2	T_3
		$r(X)$
	$r(X)$	
~~$r(X)$~~		$W(X)$
$r(X)$		
$W(X)$		

(4)

T_1	T_2	T_3
		$r(X)$
	$r(X)$	
$r(X)$		
		$W(X)$
$W(X)$		

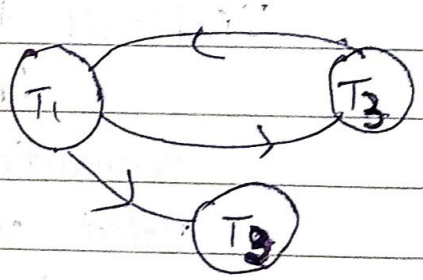
(1) Conflicting order

$r_3(X)$ $W_1(X)$

$W_1(X)$ $r_2(X)$

$W_1(X)$ $W_3(X)$

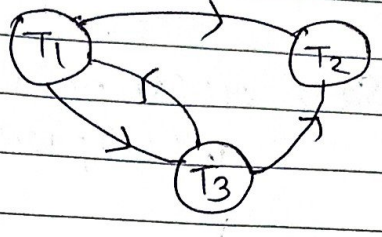
$r_2(X)$ $W_3(X)$



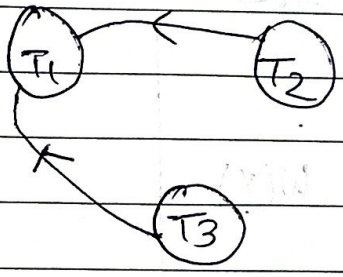
Date / /
Page No.
Shivalal

2. Conflicting order

$r_1(X)$ $W_3(X)$
$r_3(X)$ $W_1(X)$
$W_3(X)$ $W_1(X)$
$W_3(X)$ $r_2(X)$
$W_1(X)$ $r_2(X)$

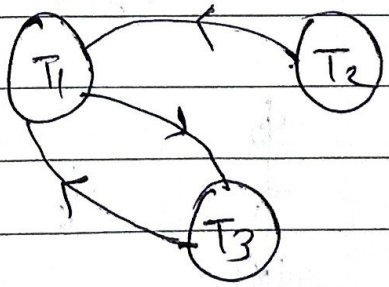


(3) Conflicting order
$r_3(X)$ $W_1(X)$
$r_2(X)$ $W_1(X)$
$W_3(X)$ $r_1(X)$
$W_3(X)$ $W_1(X)$



Conflict serializable
Equivalent serial schedule

(4) Conflicting order
$r_3(X)$ $W_1(X)$
$r_2(X)$ $W_1(X)$
$r_1(X)$ $W_3(X)$
$W_3(X)$ $W_1(X)$

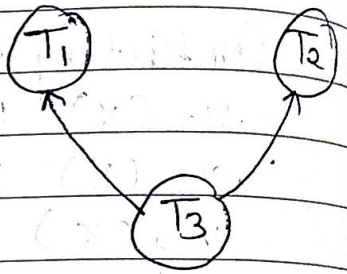


6 S_1 : $r1(X)$; $r2(z)$; $r1(z)$; $r3(X)$; $r3(Y)$; $w1(X)$; $w3(Y)$; $r2(Y)$; $w2(z)$; $w2(Y)$

S_2 : $r1(X)$; $r2(z)$; $r3(X)$; $r1(z)$; $r2(Y)$; $r3($ $w1(X)$; $w2(z)$; $w3(Y)$; $w2(Y)$

S_1

T_1	T_2	T_3
$r(X)$		
	$r(z)$	
$r(z)$		
		$r(X)$
.		$r(Y)$
$W(X)$		
		$W(Y)$
	$r(Y)$	
	$W(z)$	
	$W(Y)$	



equivalent serial sched

T_3 T_1 T_2

T_3 T_2 T_1

$r_3(X)$ $W_1(X)$

$r_3(Y)$ $W_2(Y)$

$W_3(Y)$ $r_3(Y)$

$W_3(Y)$ $W_2(Y)$

S_2

T_1 T_2 T_3

$r(X)$

$r(z)$

$r(X)$

$r(z)$

$r(Y)$

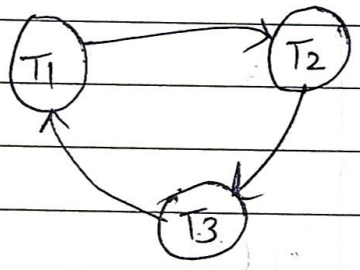
$r(Y)$

Date / /
Page No.
Shivalal

S_2

T_1	T_2	T_3
$r(X)$		
	$r(z)$	
		$r(X)$
$r(z)$		
	$r(y)$	
		$r(y)$
$w(X)$		
	$w(z)$	
		$w(y)$
	$w(y)$	

$r_3(X)$ $w_1(X)$
$r_1(z)$ $w_2(z)$
$r_2(y)$ $w_3(y)$
$r_3(y)$ $w_2(y)$



Since cycle occurs so it is not a conflict serializable

7 S_3

T_1	T_2	T_3
$r(X)$		
	$r(z)$	
$r(z)$		
		$r(X)$
		$r(y)$
$w(X)$		
C1		
		$w(y)$
		C3
	$r(y)$	
	$w(z)$	
	$w_2(y)$	
	C2	

recoverable schedule

Z = 5
X = 10
Y = 20

S4

T_1	T_2	T_3	
r(X)			
	r(Z)		
r(Z)			
		r(X)	
		r(Y)	
W(X)			
		W(Y) 20	Non-recoverable
	r(Y) 20		
	W(Z) 5		
	W(Y) 20		
C1			
	C2	C3	

S5

T_1	T_2	T_3	
r(X)			
	r(Z)		
		r(X)	
r(Z)			
	r(Y)		
		r(Y)	Recoverable.
W(X)			
C1			
	W(Z)		
		W(Y)	
	W(Y)		
		C3	
	C2		

Read, write of another transaction should be done after commit of the transaction.

Date / /
Page No.
Shivalal

Q S_1

T_1	T_2
$r(X)$	
$r(Y)$	
	$r(X)$
	$r(Y)$
	$W(Y)$
$W(X)$	

$r_1(y)\ W_2(y)$
$r_2(x)\ W_1(x)$

S_2

T_1	T_2
$r(X)$	
	$r(X)$
	$r(Y)$
	$W(Y)$
$r(Y)$	
$W(X)$	

$r_2(x)\ W_1(x)$
$W_2(Y)\ r_1(Y)$

T_1	T_2
$r(X)$	
$r(Y)$	
$W(X)$	
	$r(X)$
	$r(Y)$
	$W(Y)$

$r_1(Y)\ W_2(Y)$
$W_1(X)\ R_2(X)$

T_1	T_2
$r(X)$	
$W(Y)$	
$W(X)$	
	$r(X)$
	$r(Y)$
	$W(Y)$

$r_1(Y)\ W_2(Y)$
$W_1(X)\ R_2(X)$

T_1	T_2
	$r(X)$
	$r(Y)$
	$W(Y)$
$r(X)$	
$r(Y)$	
$W(X)$	

Since the order of conflicting operations is not same
So, it is not conflict

Since the order of conflicting operations is not same
So, it is not conflict
Serialisation

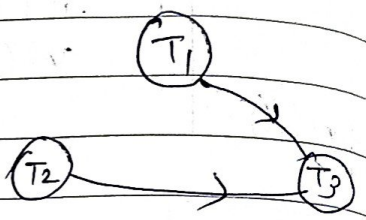
⇒ Hence both S_1 & S_2 are not conflict serializable

Date / /
Page No.

9 ≡ S1

T_1	T_2	T_3
R(A)		
R(C)		
	R(B)	
	W(B)	
		R(B)
R(A)		
		R(C)
		W(C)
W(A)		

$R_1(C)$ $W_3(C)$
$W_2(B)$ $R_3(B)$

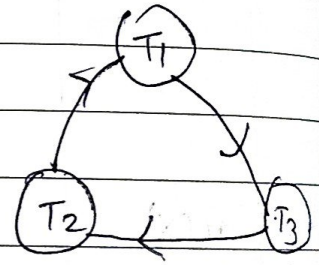


No cycle exist

S2

T_1	T_2	T_3
	R(A)	
R(C)		
	R(B)	
		R(B)
	W(B)	
R(A)		
		R(C)
		W(C)
W(A)		

$R_2(A)$ $W_1(A)$
$R_1(C)$ $W_3(C)$
$R_3(B)$ $W_2(B)$



Cycle exist

So option (a) is correct i.e. S_1 is conflict serizable.

10 S1

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
$r(z)$		
		$r(x)$
		$r(y)$
$w(x)$		
$C1$		
		~~w~~
		$W(y)$
		$C3$
	$r(y)$	
	$W(y)$	
	$W_2(z)$	
	$C2$	

Recoverable

S2

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
$r(z)$		
		$r(x)$
		$r(y)$
$W(x)$		
		$W(y)$
	$r(y)$	
	$W(z)$	
	$W(y)$	
$C1$		
	$C2$	
		$C3$

Non-recoverable

S3

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
		$r(x)$
$r(z)$		
	$r(y)$	
		$r(y)$
$W(x)$		
$C1$		
	$W(z)$	
		$W(y)$
	$W_2(y)$	
	$C2$	$C3$

Recoverable

Page No.

11 S_1

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

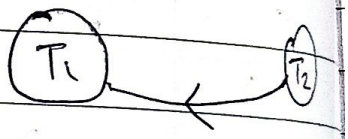
$R_1(A)$ $W_2(A)$
$W_1(A)$ $W_2(A)$
$R_2(B)$ $W_2(B)$

Cycle exist

S_2

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

$R_2(B)$ $W_2(B)$
$W_2(A)$ $R_1(A)$
$W_2(A)$ $W_1(A)$



No cycle exist

~~(b)~~ (b) option is correct → S_2 is conflict serializable

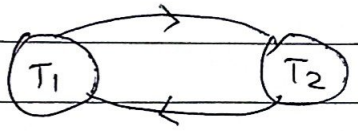
Date / /
Page No.
Shivalal

12

S_1

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

$R_1(A)$ $W_2(A)$
$W_1(A)$ ~~$R_2(A)$~~
~~W_1~~ $R_2(B)$ $W_1(B)$

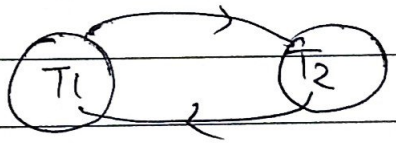


Cycle exist

S_2

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

$R_1(A)$ $W_2(A)$
$W_2(A)$ $W_1(A)$



Cycle exist

None of these is conflict serializable.