Tutorial – 13
Transaction Processing & Concurrency Control

1) List all possible schedules for transactions T1 and T2 from below figure and determine which are conflict serializable (correct) and which are not.

(a)

T_1
read_item (X);
X:=X-N;
write_item (X);
read_item (Y);
Y:=Y+N;
write_item (Y);

(b)

T_2
read_item (X);
X:=X+M;
write_item (X);

Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

190031249 Tutorial- 13 P.Mohith

1. using shortcut method we represent the Transaction like this

T1 : r1(x) ; w1(x) ; r1(y); w1(y);

T2 : r2(x); w2(x)

Here m=2, n1 = 4, n2 = 2

The no of possible schedules =

(4+2)! / (4!*2!) = 720/(24×2) = 15

Following are the 15 possible schedules and the type of each schedule

S1: r1(x); w1(x); r1(y); w1(y), r2(x); w2(x)
Serial (and hence serializable)

S2: r1(x); w1(x); r1(y); r2(x); w1(y); w2(x);
(conflict) serializable

S3: r1(x); w1(x); r1(y); r2(x); w2(x); w1(y);
(conflict) serializable

S4: r1(x); w1(x); r2(x); r1(y); w1(y); w2(x);
(conflict) serializable

S5: r1(x); w1(x); r2(x); r1(y); w2(x); w1(y);
(conflict) serializable

S6: r1(x); w1(x); r2(x); w2(x); r1(y); w1(y);
(conflict) serializable

S7: r1(x); r2(x); w1(x); r1(y); w1(y); w2(x);
not (conflict) serializable

S8 : r1(x) ; r2(x) ; w1(x) ; r1(y) ; w2(x) ; w1(y);
not (conflict) serializable

S9 : r1(x) ; r2(x) ; w1(x) ; w2(x) ; r1(y) ; w1(y);
not (conflict) serializable

S10 : r1(x) ; r2(x) ; w2(x) ; w1(x) ; r1(y) ; w1(y);
not (conflict) serializable

S11 : r2(x) ; r1(x) ; w1(x) ; r1(y) ; w1(y) ; w2(x);
not (conflict) serializable

S12 : r2(x) ; r1(x) ; w1(x) ; r1(y) ; w2(x) ; w1(y);
not (conflict) serializable

S13 : r2(x) ; r1(x) ; w1(x) ; w2(x) ; r1(y) ; w1(y);
not (conflict) serializable

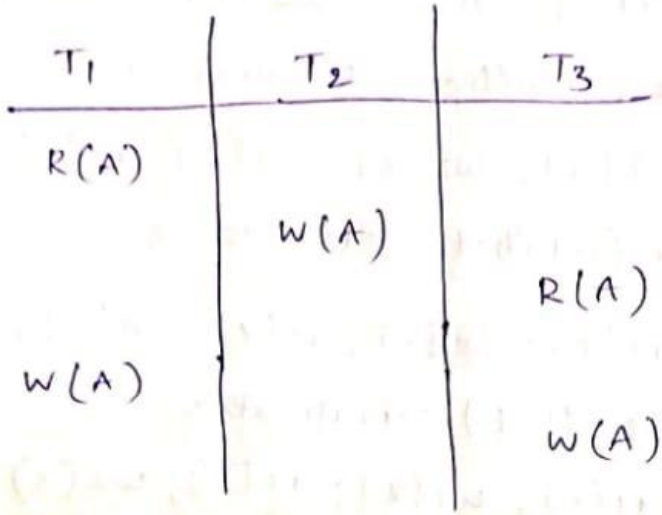
S14 : r2(x) ; r1(x) ; w2(x) ; w1(x) ; r1(y) ; w1(y);
not (conflict) serializable

S15 : r2(x) ; w2(x) ; r1(x) ; w1(x) ; r1(y) ; w1(y);
serial (and hence also serializable)

2) Check whether the given schedule S is view serializable or not. If yes, then give the serial schedule.

S : R1(A) , W2(A) , R3(A) , W1(A) , W3(A)

2. For simplicity and better understanding, we can represent the given schedule pictorially as:



We know if a schedule is conflict serializable, then it is surely view serializable

So, let us check whether the given schedule is conflict serializable or not.

Checking Whether S is Conflict Serializable or Not

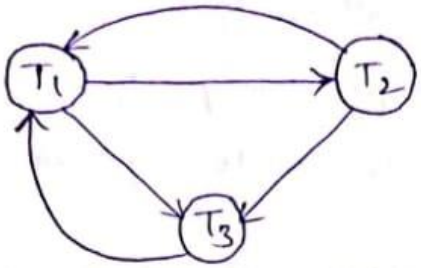
Step - 1

List all the conflicting operations and determine the dependency between the transactions.

$R_1(A), W_2(A)$	$(T_1 \rightarrow T_2)$
$W_2(A), R_3(A)$	$(T_2 \rightarrow T_3)$
$W_2(A), W_1(A)$	$(T_2 \rightarrow T_1)$
$W_2(A), W_3(A)$	$(T_2 \rightarrow T_3)$
$R_3(A), W_1(A)$	$(T_3 \rightarrow T_1)$
$W_1(A), W_3(A)$	$(T_1 \rightarrow T_2)$
$R_1(A), W_3(A)$	$(T_1 \rightarrow T_3)$

Step - 2 :-

Draw the precedence graph



Clearly, there exists a cycle in the precedence graph.

Therefore, the graph schedule S is not conflict Serializable. Now,

Since, the given schedule S is not conflict Serializable, so, it may or may not be view Serializable

To check whether S is view serializable or not, let us use another method.

Let us check for blind writes.

Checking for Blind writes :-

There exists a blind write $W_2(A)$ in the given schedule S.

Therefore, the given schedule S may or may not be view serializable.

Now, To check whether S is view serializable or not, let us use another method

Let us derive the dependencies and then draw a dependency graph.

Drawing a Dependency Graph :-

→ T_1 firstly reads and T_2 firstly updates A.

→ So, T_1 must execute before T_2

→ Thus, we got the dependency $T_1 \rightarrow T_2$

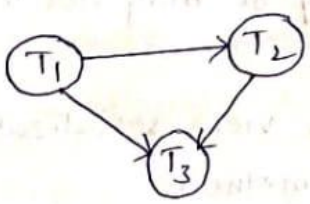
→ Final updation on A is made by the Transaction T_3.

→ So, T_3 must execute after all other transactions

→ Thus, we get the dependency $(T_1, T_2) \rightarrow T_3$

→ From, write-read sequence, we got the dependency $T_2 \rightarrow T_3$

Now, let us draw a dependency graph using these dependencies.



→ clearly, there exists no cycle in the dependency graph.

→ Therefore, the given schedule S is view serializable

→ The serialization order

$$T_1 \rightarrow T_2 \rightarrow T_3$$