Course code : **CSE2007**
Course title : **Database Management System**
Module : **5**
Topic : **4**

Characterizing Schedules based on Serializability

Objectives

This session will give the knowledge about

- Characterizing Schedules based on Serializability
- Result equivalent schedules
- Conflict serializability
- View serializability

Serializability

Serial schedule:

- A schedule S is serial if, for every transaction T participating in the schedule, all the operations of T are executed consecutively in the schedule.
- Otherwise, the schedule is called non-serial schedule.

Serializable schedule:

- A schedule S is serializable if it is equivalent to some serial schedule of the same n transactions.

Serializability

(a) Serial schedule A: T1 followed by T2

(b) Serial schedule B: T2 followed by T1

(c) Two nonserial schedules C and D with interleaving of operations

(a)

Time

T_1	T_2

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

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

Schedule A

(b)

Time

T_1	T_2

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

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

Schedule B

(c)

Time

T_1	T_2
read_item(X); $X := X - N$;	
	read_item(X); $X := X + M$;
write_item(X); read_item(Y);	
	write_item(X);
$Y := Y + N$; write_item(Y);	

Schedule C

(d)

Time

T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);	

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

read_item(Y);
$Y := Y + N$;
write_item(Y);	

Schedule D

Problem with serial schedules

- Limit concurrency by prohibiting interleaving of operations
- Unacceptable in practice
- Solution: determine which schedules are equivalent to a serial schedule and allow those to occur

Serializable schedule of n transactions:

- Equivalent to some serial schedule of same n transactions
  - Result Equivalent Schedules
  - View Serializability
  - Conflict Serializability

Serializable schedules

A non-serial schedule of n number of transactions is said to be serializable schedule, if it is equivalent to the serial schedule of those n transactions.

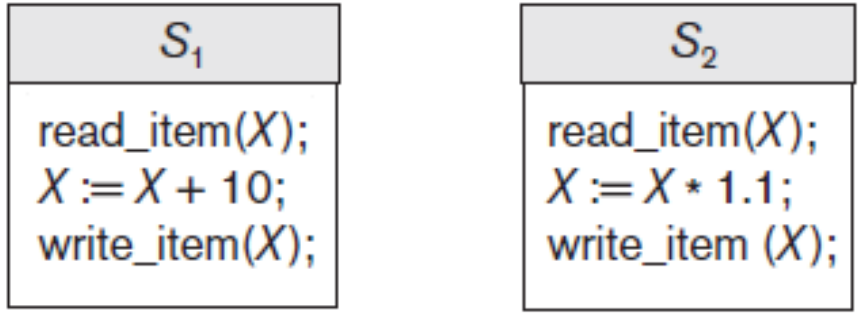
t1	t2	t1	t2	
----	----	----	----	a = 10
r1(a)		r1(a)		b = 20
	r2(b)	w1(a+5)		
w1(a+5)		c1		
	w2(b+5)		r2(b)	
c1			w2(b+5)	
	r2(a)		r2(a)	
	c2		c2	

Result equivalent schedules

Produce the same final state of the database

May be accidental

Cannot be used alone to define equivalence of schedules



Two schedules that are result equivalent for the initial value of X = 100 but are not result equivalent in general

Conflict Serializability

A schedule is called conflict serializable if it is conflict equivalent to serial schedule.

Conflicting operations:

Two operations are said to be conflicting if all conditions satisfy:

- They belong to different transactions
- They operate on the same data item
- At Least one of them is a write operation

Examples for Conflict Operations

- Conflicting operations pair (R1(A), W2(A)) because they belong to two different transactions on same data item A and one of them is write operation.
- Similarly, (W1(A), W2(A)) and (W1(A), R2(A)) pairs are also conflicting.
- On the other hand, (R1(A), W2(B)) pair is non-conflicting because they operate on different data item.
- Similarly, ((W1(A), W2(B)) pair is non-conflicting.

Examples for Conflict Serialization

S1

t1	t2
----	----
r1(a)	
w1(a)	
	r2(a)
	w2(a)
r1(b)	
w1(b)	
	r2(b)
	w2(b)

S2

t1	t2
----	----
r1(a)	
w1(a)	
r1(b)	
w1(b)	
	r2(a)
	w2(a)
	r2(b)
	w2(b)

Procedure to check Conflict Serialization

Precedence graph is a simple algorithm for determining whether a particular schedule is conflict serializable or not.

The algorithm looks at only the read_item and write_item operations in a schedule to construct a precedence graph.

Precedence graph is a directed graph G = (N, E) that consists of a set of nodes N = {T_1, T_2, ..., T_n } and a set of directed edges E = {e_1, e_2, ..., e_m }.

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

Procedure to check Conflict Serialization

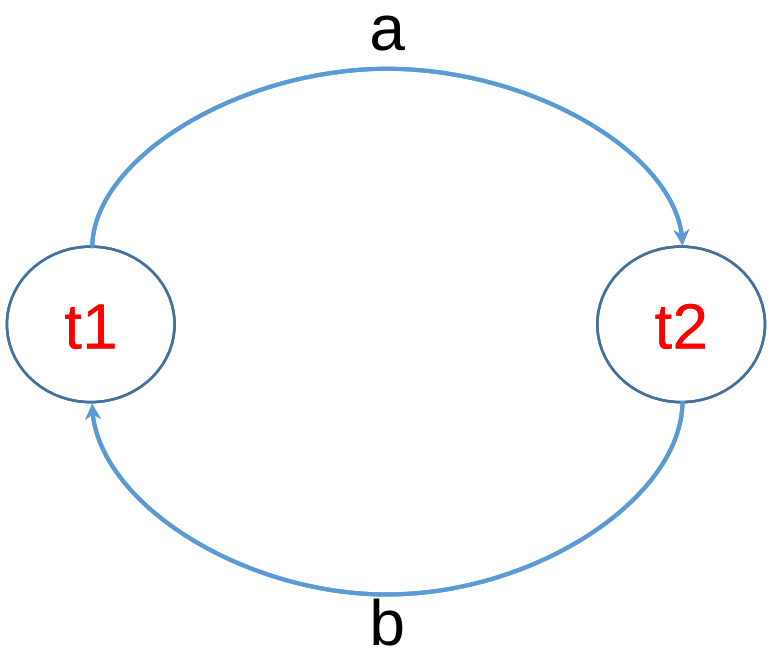
Rules to construct precedence graph:

- For each transaction Ti participating in schedule S, create a node labeled Ti in the precedence graph.
- For each case in S where Tj executes a read_item(X) after Ti executes a write_item(X), create an edge (Ti → Tj) in the precedence graph.
- For each case in S where Tj executes a write_item(X) after Ti executes a read_item(X), create an edge (Ti → Tj) in the precedence graph.
- For each case in S where Tj executes a write_item(X) after Ti executes a write_item(X), create an edge (Ti → Tj) in the precedence graph.

Procedure Graph construction - Example

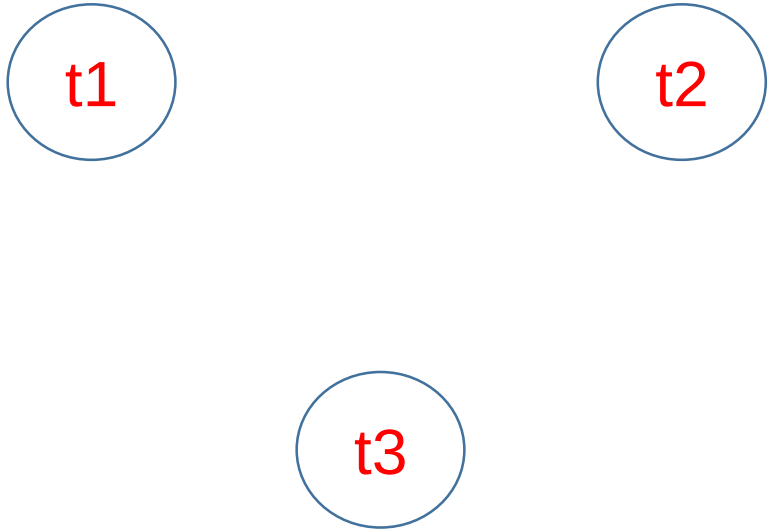
S1

t1	t2
r1(a)	
w1(a)	
	r2(b)
	w2(a)
w1(b)	



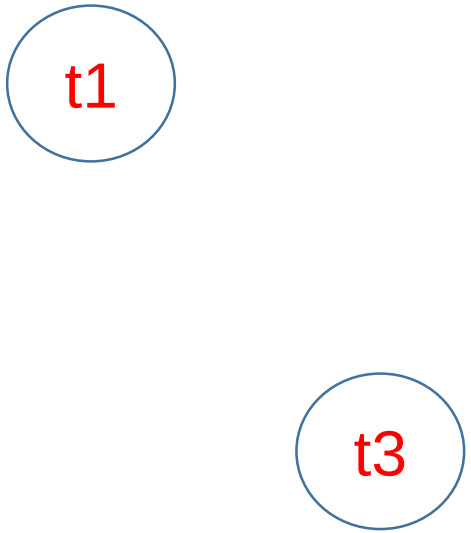
Procedure Graph construction - Example

t1	t2	t3
----	----	----
r1(a)		
	r2(c)	
r1(c)		
		r3(a)
		r3(b)
		w3(a)
	r2(b)	
	w2(c)	
	w2(b)	



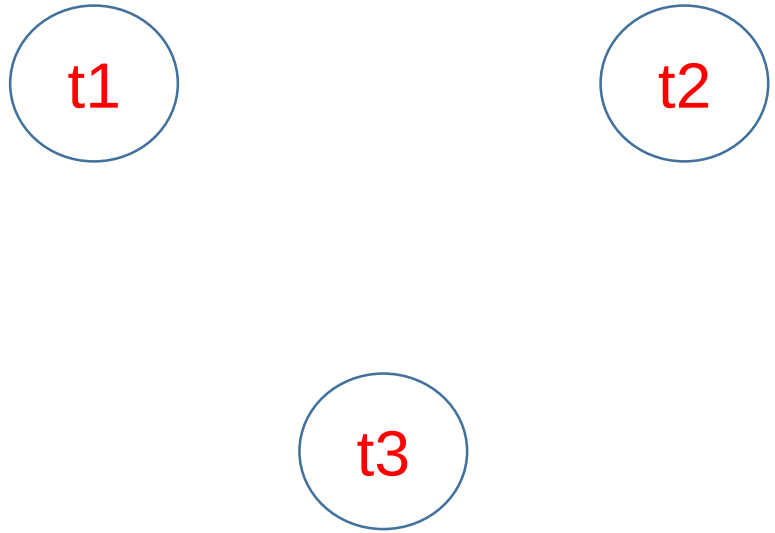
Procedure Graph construction - Example

t1	t2	t3
----	----	----
r1(a)		
		r3(a)
		w3(a)
w1(a)		
	r2(a)	



Procedure Graph construction - Example

t1	t2	t3
r1(a)		
	r2(a)	
		r3(b)
w1(a)		
	r2(c)	
	r2(b)	
	w2(b)	
		w3(c)

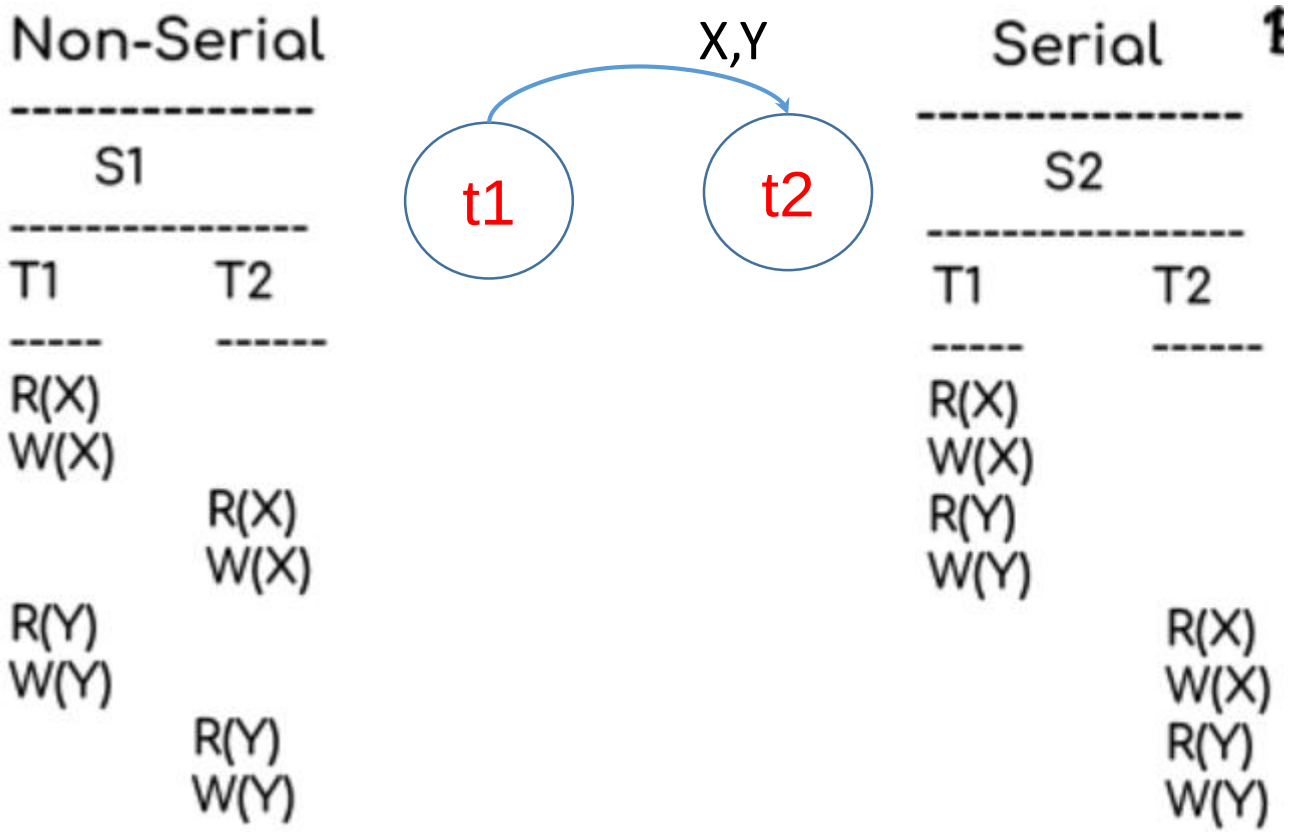


View Serializability

Two schedules are said to be view equivalent if the following three conditions hold:

- The same set of transactions participates in S and S', and S and S' include the same operations of those transactions. (Initial Read)
- For any operation Ri(X) of Ti in S, if the value of X read by the operation has been written by an operation Wj(X) of Tj (or if it is the original value of X before the schedule started), the same condition must hold for the value of X read by operation Ri(X) of Ti in S'. (Read and Write sequence)
- If the operation Wk(Y) of Tk is the last operation to write item Y in S, then Wk(Y) of Tk must also be the last operation to write item Y in S'. (Last write)

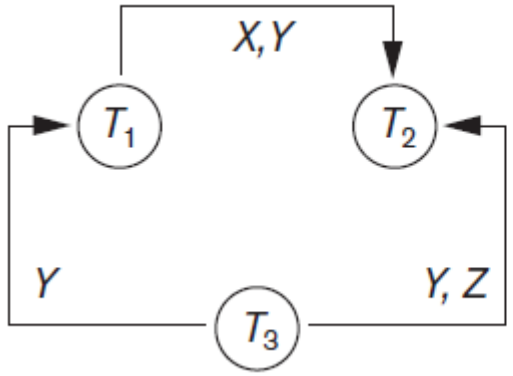
Example to check View Serializability



	X		Y	
Initial Read	T1	T1	T1	T1
Last Write	T2	T2	T2	T2
Read /Write order	T1 -> T2		T1 -> T2	

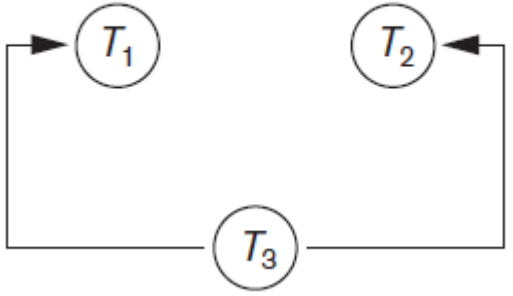
Finding equivalent serial schedules

To find an equivalent serial schedule, start with a node that does not have any incoming edges, and then make sure that the node order for every edge is not violated.



Equivalent serial schedules

$T_3 \rightarrow T_1 \rightarrow T_2$

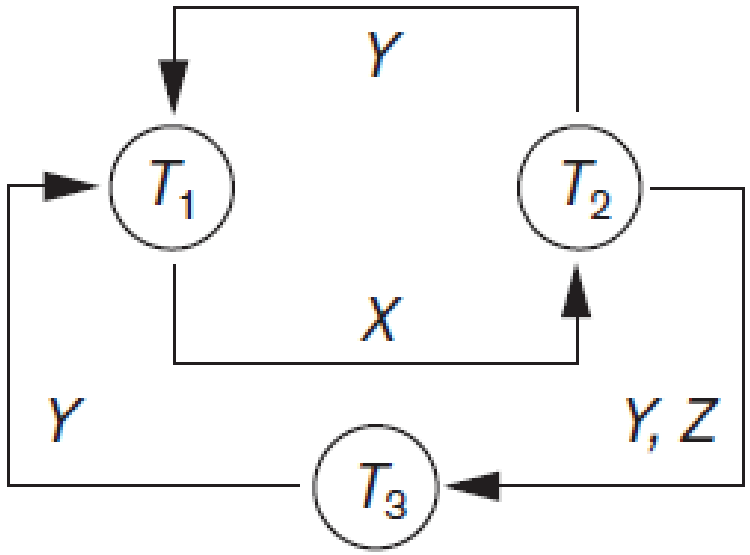


Equivalent serial schedules

$T_3 \rightarrow T_1 \rightarrow T_2$

$T_3 \rightarrow T_2 \rightarrow T_1$

Finding equivalent serial schedules



Equivalent serial schedules

None

Reason

Cycle $X(T_1 \rightarrow T_2), Y(T_2 \rightarrow T_1)$

Cycle $X(T_1 \rightarrow T_2), YZ\ (T_2 \rightarrow T_3), Y(T_3 \rightarrow T_1)$

Key Note:

Being serializable is not the same as being serial.

Being serializable implies that the schedule is a correct schedule.

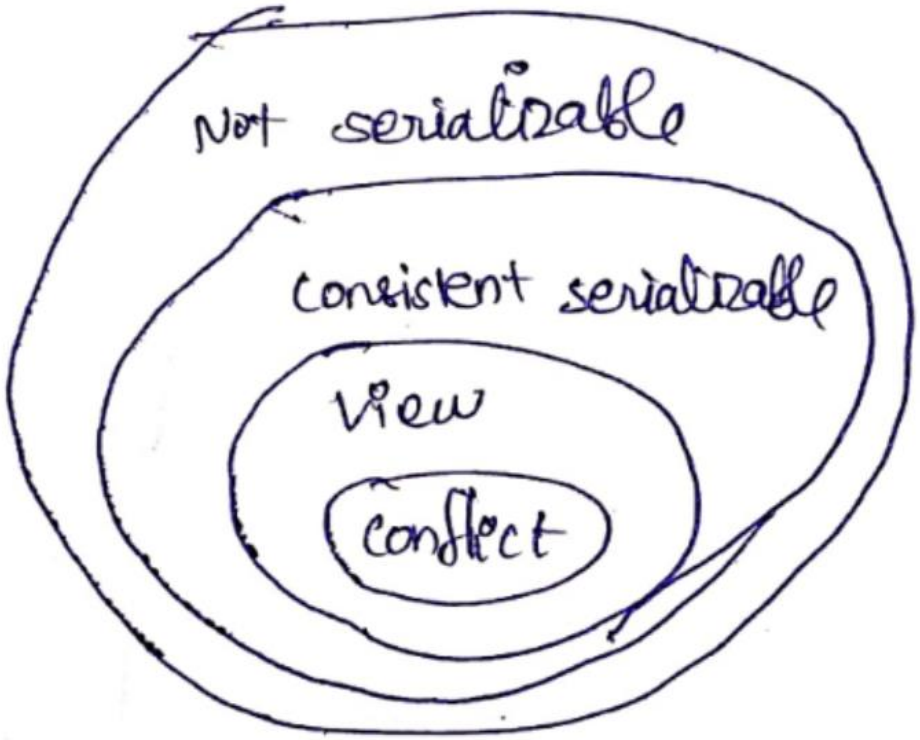
It will leave the database in a consistent state.

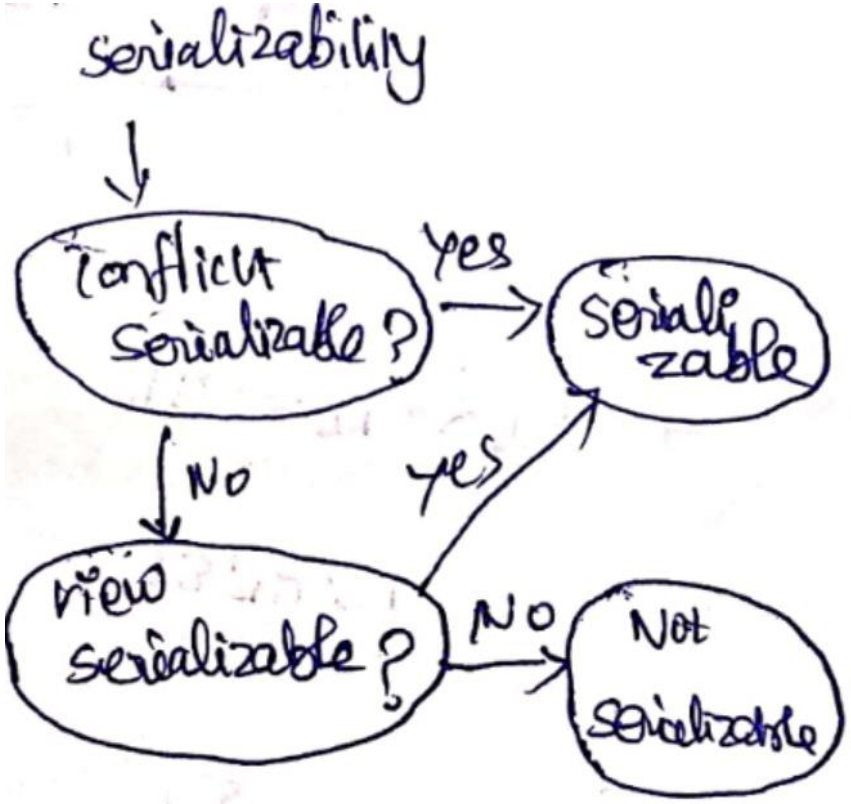
The interleaving is appropriate and will result in a state as if the transactions were serially executed, yet will achieve efficiency due to concurrent execution.

Serializability is hard to check.

- Interleaving of operations occurs in an operating system through some scheduler
- Difficult to determine beforehand how the operations in a schedule will be interleaved.

Short note:


Not serializable
Consistent serializable
View
conflict


serializability
conflict serializable?
yes
serializable
No
yes
view serializable?
No
Not serializable

Problems

Find the serializable category of the given schedule

s1 : r3(y); r3(z); r1(x); w1(x); w3(y); w3(z); r2(z); r1(y); w1(y); r2(y); w2(y); r2(x); w2(x);

s2: r1(x); r1(y); r2(x); r2(y); w2(y); w1(x);

s3: r1(x); r2(x); r2(y); w2(y); r1(y); w1(x);

Problems

Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.

a. $r_1(X); r_3(X); w_1(X); r_2(X); w_3(X);$

b. $r_1(X); r_3(X); w_3(X); w_1(X); r_2(X);$

c. $r_3(X); r_2(X); w_3(X); r_1(X); w_1(X);$

d. $r_3(X); r_2(X); r_1(X); w_3(X); w_1(X);$

Problems

Consider the three transactions T_1, T_2, and T_3, and the schedules S_1 and S_2 given below. Draw the serializability (precedence) graphs for S_1 and S_2, and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).

T_1: $r_1(X)$; $r_1(Z)$; $w_1(X)$;
T_2: $r_2(Z)$; $r_2(Y)$; $w_2(Z)$; $w_2(Y)$;
T_3: $r_3(X)$; $r_3(Y)$; $w_3(Y)$;
S_1: $r_1(X)$; $r_2(Z)$; $r_1(Z)$; $r_3(X)$; $r_3(Y)$; $w_1(X)$; $w_3(Y)$; $r_2(Y)$; $w_2(Z)$; $w_2(Y)$;
S_2: $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $r_2(Y)$; $r_3(Y)$; $w_1(X)$; $w_2(Z)$; $w_3(Y)$; $w_2(Y)$;

Summary

This session will give the knowledge about

- Characterizing Schedules based on Serializability
- Result equivalent schedules
- Conflict serializability
- View serializability