Serializability

- A schedule is serializable if it is **equivalent to a serial schedule**.
- In **serial schedules**, only **one transaction is allowed to execute at a time** i.e. **no concurrency is allowed**.
- Whereas in **serializable schedules**, **multiple transactions can execute simultaneously** i.e. **concurrency is allowed**.
- Types (forms) of serializability
  - Conflict serializability
  - View serializability

Conflicting instructions

- Let I_i and I_j be two instructions of transactions T_i and T_j respectively.

1. I_i = read(Q), I_j = read(Q)
   I_i and I_j don't conflict

2. I_i = read(Q), I_j = write(Q)
   I_i and I_j conflict

3. I_i = write(Q), I_j = read(Q)
   I_i and I_j conflict

4. I_i = write(Q), I_j = write(Q)
   I_i and I_j conflict

T_i	T_j
read (Q)	
	read (Q)

T_i	T_j
	read (Q)
read (Q)	

T_i	T_j
read (Q)	
	write(Q)

T_i	T_j
	read (Q)
write(Q)	

T_i	T_j
write(Q)	
	read (Q)

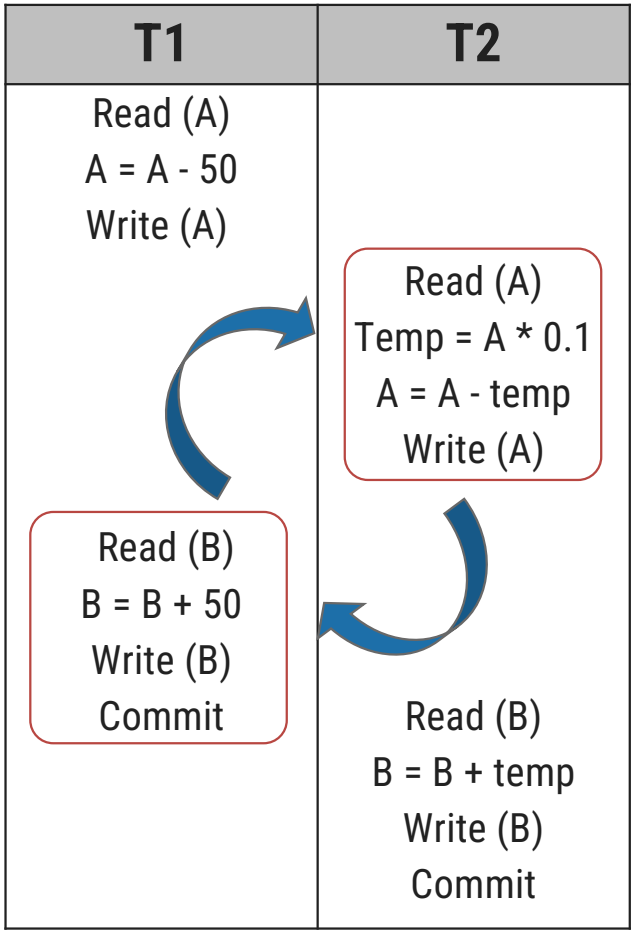
T_i	T_j
	write(Q)
read (Q)	

T_i	T_j
write(Q)	
	write(Q)

T_i	T_j
	write(Q)
write(Q)	

Conflict serializability

- If a given schedule can be **converted into a serial schedule by swapping its non-conflicting operations**, then it is called as a conflict serializable schedule.



T1	T2
Read (A) A = A - 50 Write (A) Read (B) B = B + 50 Write (B) Commit	Read (A) Temp = A * 0.1 A = A - temp Write (A) Read (B) B = B + temp Write (B) Commit

Conflict serializability (Example)

- Example of a **schedule that is not conflict serializable**:

T1	T2
Read (A)	
	Write (A)
Read (A)	

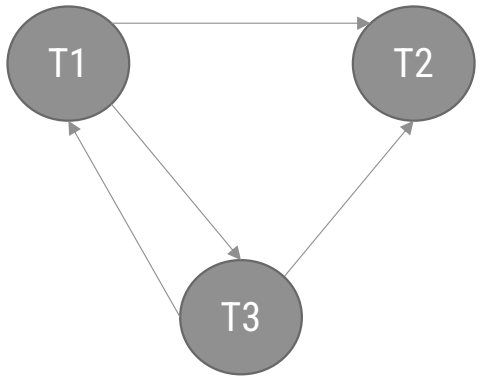
- We are **unable to swap instructions** in the above schedule to obtain either the serial schedule <T1, T2>, or the serial schedule <T2, T1>.

Cycle is formed, i.e., it is not conflict serializable schedule

Conflict serializability (Example)

- Is given **schedule conflict serializable**?

T1	T2	T3
Read (A)		
		Read (Z)
		Write (Z)
	Read (Y)	
Read (Y)		
	Write (Y)	
		Write (X)
	Write (Z)	
Write (X)		

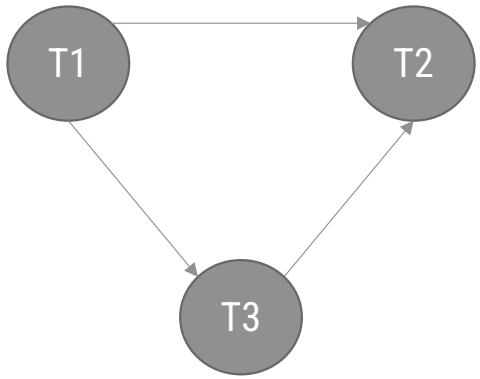


Cycle is formed, i.e., it is not conflict serializable schedule

Conflict serializability (Example)

- Is given **schedule conflict serializable**?

T1	T2	T3
Read (X)		
	Read (Y)	
		Read (Y)
	Write (Y)	
Write (X)		
		Write (X)
	Read (X)	
	Write (X)	

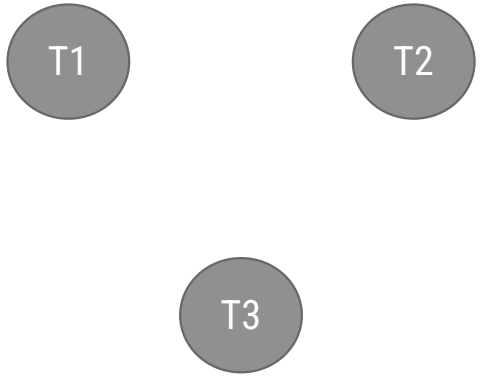


Cycle is not formed, i.e., it is a conflict serializable schedule

Conflict serializability (Example)

- Is given **schedule conflict serializable**?

T1	T2	T3
Read (a)		
	Read (b)	
		Read (c)
		Write (c)
	Write (b)	
Write (a)		

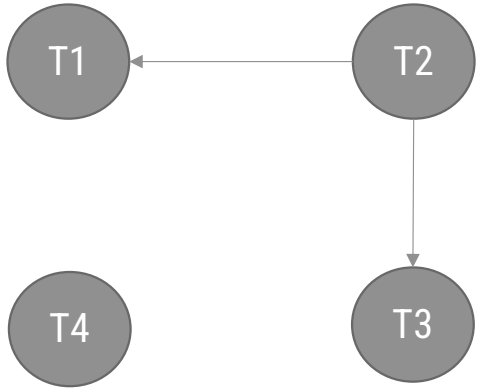


Cycle is not formed, i.e., it is a conflict serializable schedule

Conflict serializability (Example)

- Is given **schedule conflict serializable**?

T1	T2	T3	T4
	Read (X)		
		Write (X)	
		Commit	
Write (X)			
Commit			
	Write (Y)		
	Read (Z)		
	Commit		
			Read (X)
			Read (Y)
			Commit

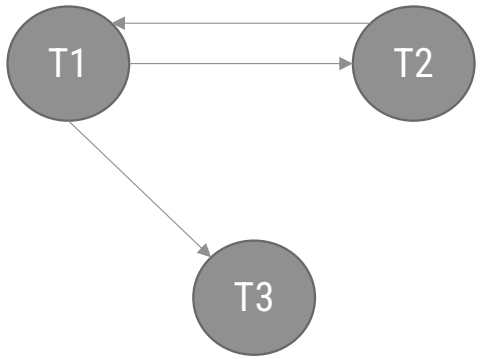


Cycle is not formed, i.e., it is a conflict serializable schedule

Conflict serializability (Example)

- Is given **schedule conflict serializable**?

T1	T2	T3
Read (a)		
	Write (a)	
Write (a)		
		Write (a)



Cycle is formed, i.e., it is not conflict serializable schedule

View serializability

- Let S1 and S2 be two schedules with the same set of transactions. S1 and S2 are view equivalent if the following three conditions are satisfied, for each data item Q
  - Initial Read
  - Updated Read
  - Final Write
- If a schedule is view equivalent to its serial schedule then the given schedule is said to be view serializable.

Initial Read

- If in **schedule S1, transaction Ti reads the initial value of Q**, then in **schedule S2 also transaction Ti must read the initial value of Q**.

S1	
T1	**T2**
Read (A)	
	Write (A)

S2	
T1	**T2**
	Write (A)
Read (A)	

S3	
T1	**T2**
	Read (A)
Write (A)	

- Above two schedules **S1 and S3 are not view equivalent** because **initial read operation in S1 is done by T1 and in S3 it is done by T2**.
- Above two schedules **S1 and S2 are view equivalent** because **initial read operation in S1 is done by T1 and in S2 it is also done by T1**.

Updated Read

- If in **schedule S1, transaction Ti executes read(Q), and that value was produced by transaction Tj** (if any), then in **schedule S2 also transaction Ti must read the value of Q that was produced by transaction Tj**.

S1		
T1	**T2**	**T3**
Write (A)		
	Write (A)	
		Read (A)

S2		
T1	**T2**	**T3**
	Write (A)	
Write (A)		Read (A)

S3		
T1	**T2**	**T3**
	Write (A)	
Write (A)		
		Read (A)

- Above two schedules **S1 and S3 are not view equal** because, **in S1, T3 is reading A that is updated by T2 and in S3, T3 is reading A which is updated by T1**.
- Above two schedules **S1 and S2 are view equal** because, **in S1, T3 is reading A that is updated by T2 and in S2 also, T3 is reading A which is updated by T2**.

Final Write

- If **Ti performs the final write on the data value in S1, then it also performs the final write on the data value in S2**.

S1		
T1	**T2**	**T3**
Write (A)		
	Read (A)	
		Write (A)

S2		
T1	**T2**	**T3**
	Read (A)	
Write (A)		
		Write (A)

S3		
T1	**T2**	**T3**
	Write (A)	
Write (A)		
		Read (A)

- Above two schedules **S1 and S3 are not view equal** because **final write operation in S1 is done by T3 and in S3 final write operation is also done by T1**.
- Above two schedules **S1 and S2 are view equal** because **final write operation in S1 is done by T3 and in S2 also the final write operation is also done by T3**.