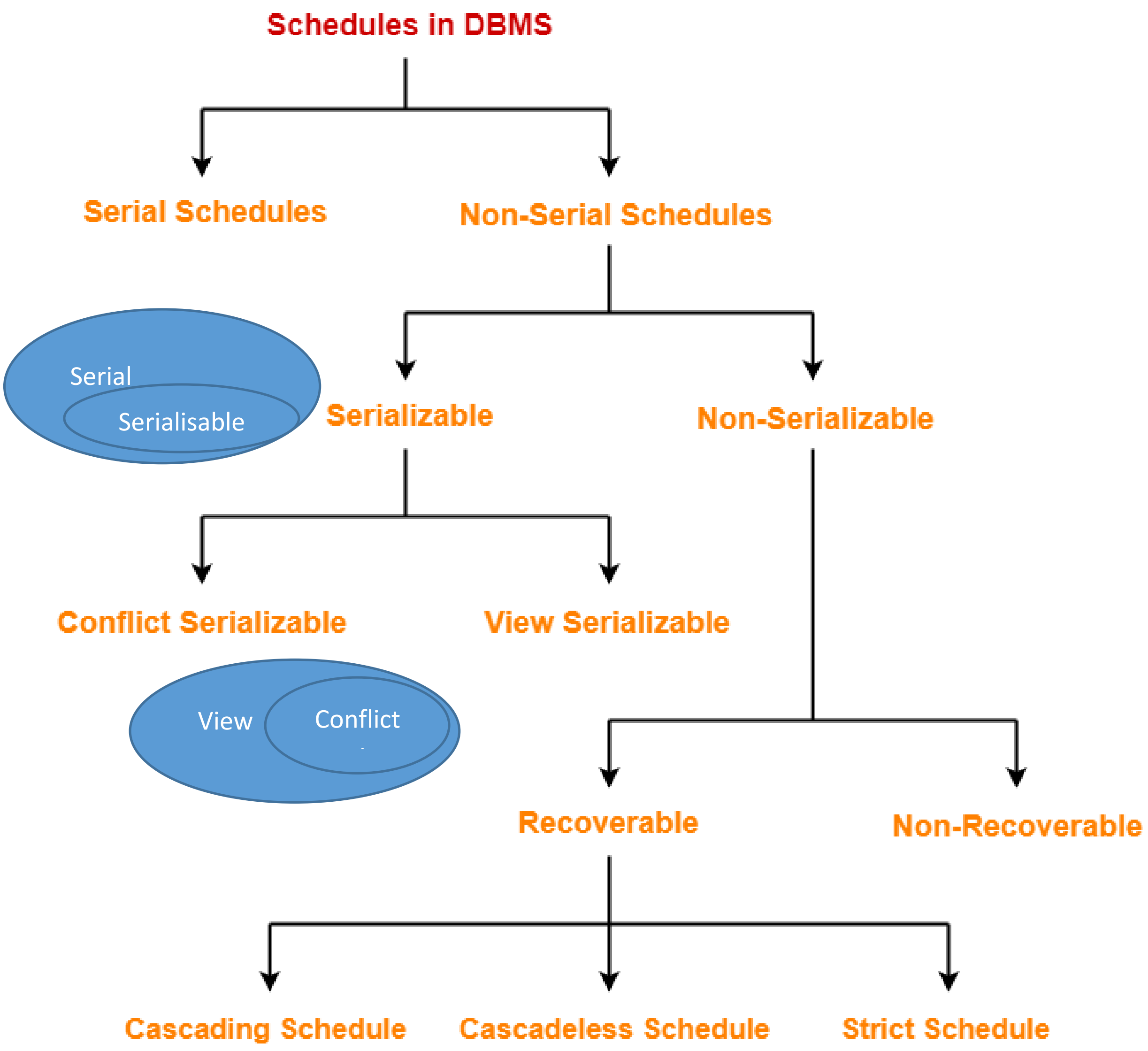
Transactions: Scheduling

Transactions are set of instructions and these instructions perform operations on database. When multiple transactions are running concurrently then there needs to be a sequence in which the operations are performed because at a time only one operation can be performed on the database. This sequence of operations is known as Schedule.



<u>**Serializability in DBMS-**</u>

- Serializability is a concept that helps to identify which **non-serial schedules are correct and will maintain the consistency of the database**.
- Some non-serial schedules may lead to inconsistency of the database.

Serializable Schedules-

If a given non-serial schedule of ‘n’ transactions is equivalent to some serial schedule of ‘n’ transactions, then it is called as a **serializable schedule.**

Characteristics-

Serializable schedules behave exactly same as serial schedules. Thus, serializable schedules are always-

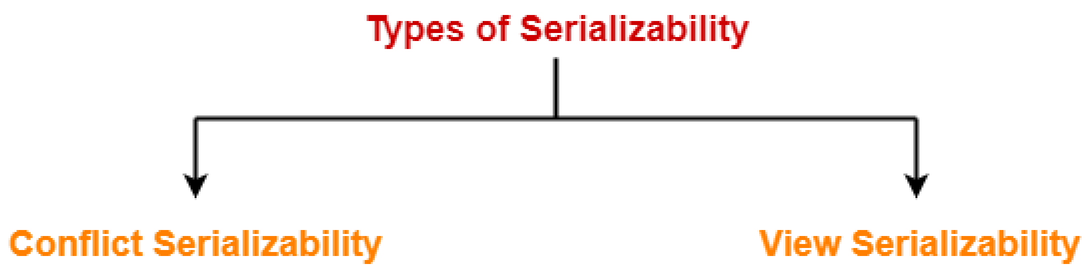
- Consistent
- Recoverable
- Casacadeless
- Strict

Serial Schedules Vs Serializable Schedules-

Serial Schedules	Serializable Schedules
No concurrency is allowed. Thus, all the transactions necessarily execute serially one after the other.	Concurrency is allowed. Thus, multiple transactions can execute concurrently.
Serial schedules lead to less resource utilization and CPU throughput.	Serializable schedules improve both resource utilization and CPU throughput.
Serial Schedules are less efficient as compared to serializable schedules. (due to above reason)	Serializable Schedules are always better than serial schedules. (due to above reason)

Types of Serializability-

Serializability is mainly of two types-



- Conflict Serializability
- View Serializability

Conflict Serializability-

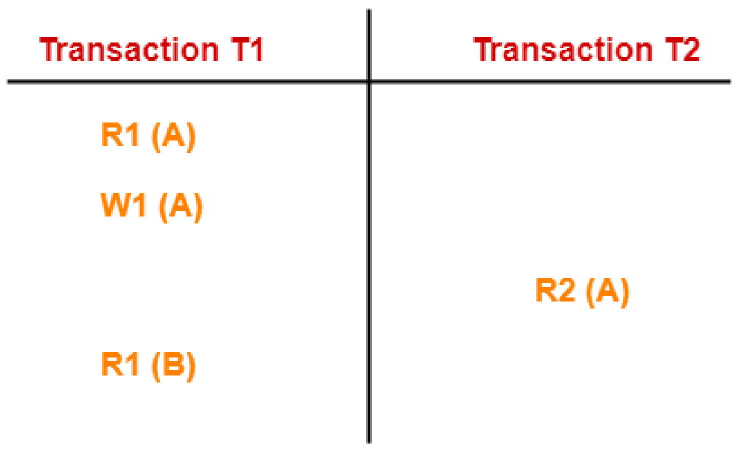
If a given non-serial schedule can be converted into a serial schedule by swapping its non-conflicting operations, then it is called as a **conflict serializable schedule**.

Conflicting Operations-

Two operations are called as **conflicting operations** if all the following conditions hold true for them-

- Both the operations belong to different transactions
- Both the operations are on the same data item
- At least one of the two operations is a write operation

Example-



Transaction T1	Transaction T2
R1 (A)	
W1 (A)	
	R2 (A)
R1 (B)	

- W1 (A) and R2 (A) are called as conflicting operations.
- This is because all the above conditions hold true for them.

Checking Whether a Schedule is Conflict Serializable Or Not-

Problem-01:

Check whether the given schedule S is conflict serializable or not-

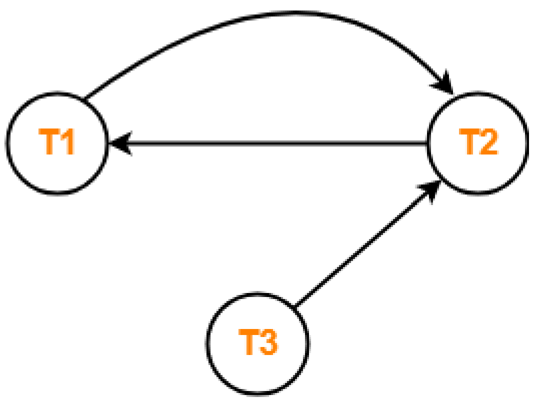
S : $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$

Solution-

Step-01: **List all the conflicting operations and determine the dependency between the transactions-**

➢ $R_2(A)$, $W_1(A)$	$(T_2 \rightarrow T_1)$
➢ $R_1(B)$, $W_2(B)$	$(T_1 \rightarrow T_2)$
➢ $R_3(B)$, $W_2(B)$	$(T_3 \rightarrow T_2)$

Step-02: Draw the precedence graph-



- there exists a cycle in the precedence graph.
- Therefore, the given schedule S is **not conflict serializable**.

Problem-02:

Check whether the given schedule S is conflict serializable and recoverable or not-

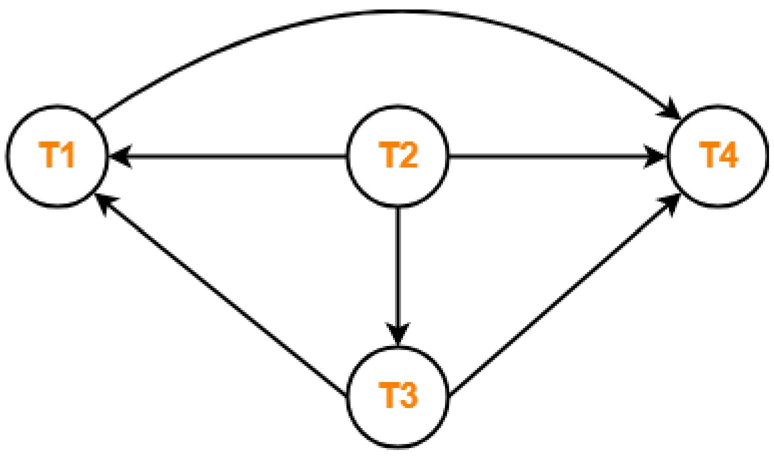
T1	T2	T3	T4
	R(X)		
		W(X)	
		Commit	
W(X)			
Commit			
	W(Y)		
	R(Z)		
	Commit		
			R(X)
			R(Y)
			Commit

Solution- Checking Whether S is Conflict Serializable Or Not-

Step-01: List all the conflicting operations and determine the dependency between the transactions-

- $R_2(X)$, $W_3(X)$ $(T_2 \rightarrow T_3)$
- $R_2(X)$, $W_1(X)$ $(T_2 \rightarrow T_1)$
- $W_3(X)$, $W_1(X)$ $(T_3 \rightarrow T_1)$
- $W_3(X)$, $R_4(X)$ $(T_3 \rightarrow T_4)$
- $W_1(X)$, $R_4(X)$ $(T_1 \rightarrow T_4)$
- $W_2(Y)$, $R_4(Y)$ $(T_2 \rightarrow T_4)$

Step-02: Draw the precedence graph-



- Clearly, there exists no cycle in the precedence graph.
- Therefore, the given schedule S is **conflict serializable.**

Checking Whether S is Recoverable Or Not-

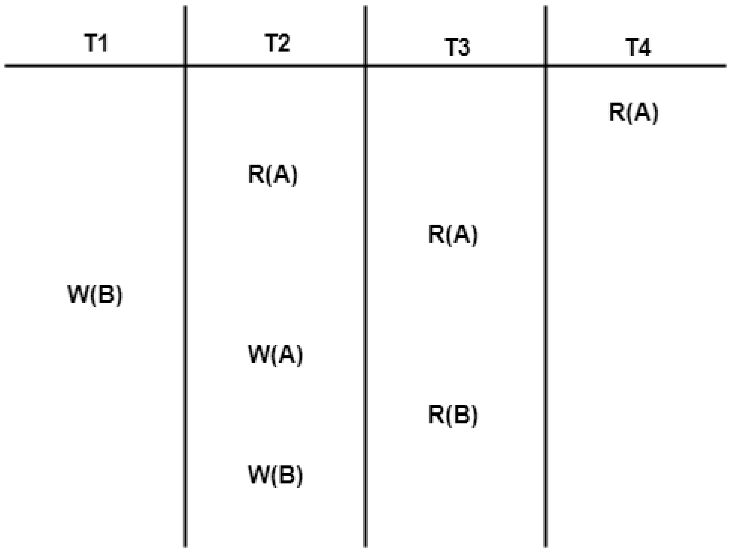
- Conflict serializable schedules are always recoverable.
- Therefore, the given schedule S is recoverable.

Alternatively,

- There exists no dirty read operation.
- This is because all the transactions which update the values commits immediately.
- Therefore, the given schedule S is recoverable.
- Also, S is a **Cascadeless Schedule**.

Problem-03:

Check whether the given schedule S is conflict serializable or not. If yes, then determine all the possible serialized schedules-



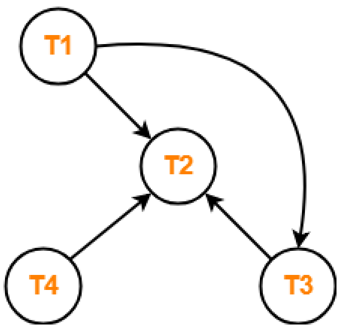
T1	T2	T3	T4
			R(A)
	R(A)		
		R(A)	
W(B)			
	W(A)		
		R(B)	
	W(B)		

Solution: **Checking Whether S is Conflict Serializable Or Not-**

Step-01: List all the conflicting operations and determine the dependency between the transactions-

- $R_4(A)$, $W_2(A)$ ($T_4 \rightarrow T_2$)
- $R_3(A)$, $W_2(A)$ ($T_3 \rightarrow T_2$)
- $W_1(B)$, $R_3(B)$ ($T_1 \rightarrow T_3$)
- $W_1(B)$, $W_2(B)$ ($T_1 \rightarrow T_2$)
- $R_3(B)$, $W_2(B)$ ($T_3 \rightarrow T_2$)

Step-02: Draw the precedence graph-



- Clearly, there exists no cycle in the precedence graph.
- Therefore, the given schedule S is **conflict serializable**.

Problem-04: **Determine all the possible serialized schedules for the given schedule-**

T1	T2
R(A)	
A = A-10	
	R(A)
	Temp = 0.2 x A
	W(A)
	R(B)
W(A)	
R(B)	
B = B+10	
W(B)	
	B = B+Temp
	W(B)

Solution- The given schedule S can be rewritten as-

T1	T2
R(A)	
	R(A)
	W(A)
	R(B)
W(A)	
R(B)	
W(B)	
	W(B)

Checking Whether S is Conflict Serializable Or Not-

Step-01: List all the conflicting operations and determine the dependency between the transactions-

- $R_1(A)$, $W_2(A)$ $(T_1 \rightarrow T_2)$
- $R_2(A)$, $W_1(A)$ $(T_2 \rightarrow T_1)$
- $W_2(A)$, $W_1(A)$ $(T_2 \rightarrow T_1)$
- $R_2(B)$, $W_1(B)$ $(T_2 \rightarrow T_1)$
- $R_1(B)$, $W_2(B)$ $(T_1 \rightarrow T_2)$
- $W_1(B)$, $W_2(B)$ $(T_1 \rightarrow T_2)$

Step-02: Draw the precedence graph-

$$\begin{tikzcd}
T1 \arrow[r, bend left] & T2 \arrow[l, bend left]
\end{tikzcd}$$

- Clearly, there exists a cycle in the precedence graph.
- Therefore, the given schedule S is **not conflict serializable**.

View Serializability:

- A schedule is view serializable if it is view equivalent to a serial schedule.
- If a schedule is conflict serializable, then it will be view serializable.
- The view serializable which does not conflict serializable contains blind writes.

View Equivalent : Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:

1. Initial Read : An initial read of both schedules must be the same.

Suppose two schedule S1 and S2. In schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

T1	T2
Read(A)	
	Write(A)

Schedule S1

T1	T2
	Write(A)
Read(A)	

Schedule S2

Above two schedules are view equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

Updated Read : In schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated by Tj.

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

Schedule S1

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

Schedule S2

Above two schedules are not view equal because, in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

3. Final Write

A final write must be the same between both the schedules. In schedule S1, if a transaction T1 updates A at last then in S2, final writes operations should also be done by T1.

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

Schedule S1

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

Schedule S2

Above two schedules is view equal because Final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

Example:

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

Schedule S

With 3 transactions, the total number of possible schedule = 3! = 6

- S1 = <T1 T2 T3>
- S2 = <T1 T3 T2>
- S3 = <T2 T3 T1>
- S4 = <T2 T1 T3>
- S5 = <T3 T1 T2>
- S6 = <T3 T2 T1>

Taking first schedule S1:

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

Schedule S1

Step 1: Update read on data items : In both schedules S and S1, there is no read except the initial read that's why we don't need to check that condition.

Step 2: Initial Read : The initial read operation in S is done by T1 and in S1, it is also done by T1.

Step 3: Final Write : The final write operation in S is done by T3 and in S1, it is also done by T3.

So, S and S1 are view Equivalent.

Note: The first schedule S1 satisfies all three conditions, so we don't need to check another schedule.

Checking whether a Schedule is View Serializable Or Not-

Method-01: **Check whether the given schedule is conflict serializable or not.**

- If the given schedule is conflict serializable, then it is surely view serializable..
- If the given schedule is not conflict serializable, then it may or may not be view serializable. check using other methods.

Thumb Rules

- *All conflict serializable schedules are view serializable.*
- *All view serializable schedules may or may not be conflict serializable.*

Method-02: Check if there exists any blind write operation. Writing without reading is called as a blind write.

- If there is no blind write, then the schedule is surely **not view serializable**.
- If there exists any blind write, then the schedule may or may not be view serializable. check using other methods.

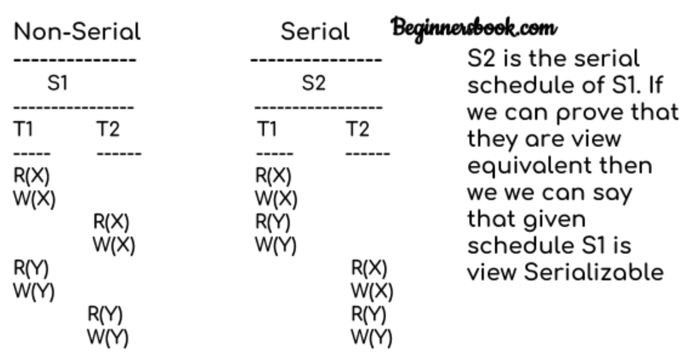
Thumb Rule

No blind write means not a view serializable schedule.

Method-03: In this method, try finding a view equivalent serial schedule.

View Serializable **: If a schedule is view equivalent to its serial schedule then the given schedule is said to be View Serializable.**

View Serializable Example



```
Non-Serial              Serial
--------------          ---------------
    S1                      S2
----------------        -----------------
T1      T2              T1      T2
-----   ------          -----   ------
R(X)                    R(X)
W(X)                    W(X)
        R(X)            R(Y)
        W(X)            W(Y)
R(Y)                            R(X)
W(Y)                            W(X)
        R(Y)                    R(Y)
        W(Y)                    W(Y)
```

Beginnersbook.com

S2 is the serial schedule of S1. If we can prove that they are view equivalent then we we can say that given schedule S1 is view Serializable

Lets check the three conditions of view serializability:

Initial Read :

- In schedule S1, transaction T1 first reads the data item X. In S2 also transaction T1 first reads the data item X.

- Check for Y. In schedule S1, transaction T1 first reads the data item Y. In S2 also the first read operation on Y is performed by T1. We checked for both data items X & Y and the **initial read** condition is satisfied in S1 & S2.

Final Write

In schedule S1, the final write operation on X is done by transaction T2. In S2 also transaction T2 performs the final write on X.

Lets check for Y. In schedule S1, the final write operation on Y is done by transaction T2. In schedule S2, final write on Y is done by T2. We checked for both the data items X & Y and the final write **condition is satisfied in S1 & S2.**

Update Read

In S1, transaction T2 reads the value of X, written by T1. In S2, the same transaction T2 reads the X after it is written by T1. In S1, transaction T2 reads the value of Y, written by T1. In S2, the same transaction T2 reads the value of Y after it is updated by T1. **The update read condition is also satisfied for both the schedules.**

Result: Since all the three conditions that checks whether the two schedules are view equivalent are satisfied in this example, which means S1 and S2 are view equivalent. Also, as we know that the schedule S2 is the serial schedule of S1, thus we can say that the schedule S1 is view serializable schedule.

- **By using the above three conditions, write all the dependencies.**
- **Then, draw a graph using those dependencies.**
- **If there exists no cycle in the graph, then the schedule is view serializable otherwise not.**

PRACTICE PROBLEMS BASED ON VIEW SERIALIZABILITY-

Problem-01: Check whether the given schedule S is view serializable or not-

T1	T2	T3	T4
R (A)			
	R (A)		
		R (A)	
			R (A)
W (B)			
	W (B)		
		W (B)	
			W (B)

Solution-

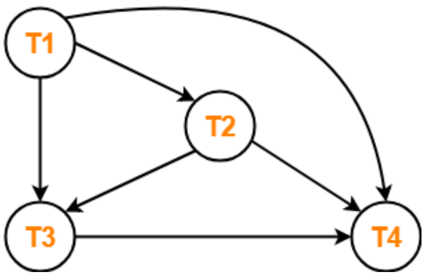
- If a schedule is conflict serializable, then it is surely view serializable.
- So, let us check whether the given schedule is conflict serializable or not.

Checking Whether S is Conflict Serializable Or Not-

Step-01: List all the conflicting operations and determine the dependency between the transactions-

- $W_1(B)$, $W_2(B)$ $(T_1 \rightarrow T_2)$
- $W_1(B)$, $W_3(B)$ $(T_1 \rightarrow T_3)$
- $W_1(B)$, $W_4(B)$ $(T_1 \rightarrow T_4)$
- $W_2(B)$, $W_3(B)$ $(T_2 \rightarrow T_3)$
- $W_2(B)$, $W_4(B)$ $(T_2 \rightarrow T_4)$
- $W_3(B)$, $W_4(B)$ $(T_3 \rightarrow T_4)$

Step-02: Draw the precedence graph-



- Clearly, there **exists no cycle** in the precedence graph. mTherefore, the given schedule S is **conflict serializable**.
- Thus, we conclude that the given schedule is also **view serializable**.

Problem-02: Check whether the given schedule S is view serializable or not-

T1	T2	T3
R (A)		
	R (A)	
		W (A)
W (A)		

Solution-

- if a schedule is conflict serializable, then it is surely view serializable.

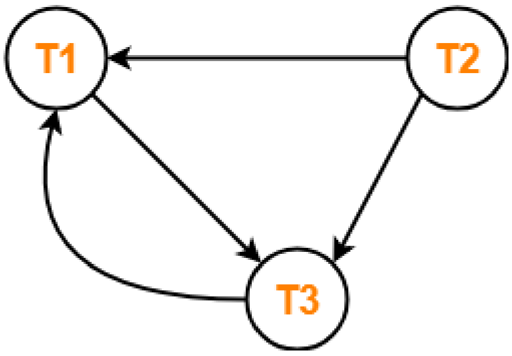
So, let us check whether the given schedule is conflict serializable or not.

Checking Whether S is Conflict Serializable Or Not-

Step-01: List all the conflicting operations and determine the dependency between the transactions-

$R_1(A)$, $W_3(A)$	($T_1 \rightarrow T_3$)
$R_2(A)$, $W_3(A)$	($T_2 \rightarrow T_3$)
$R_2(A)$, $W_1(A)$	($T_2 \rightarrow T_1$)
$W_3(A)$, $W_1(A)$	($T_3 \rightarrow T_1$)

Step-02: Draw the precedence graph-



- **There exists a cycle** in the precedence graph. Therefore, the given schedule S is **not conflict serializable**.
- 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 check for blind writes.

Checking for Blind Writes :

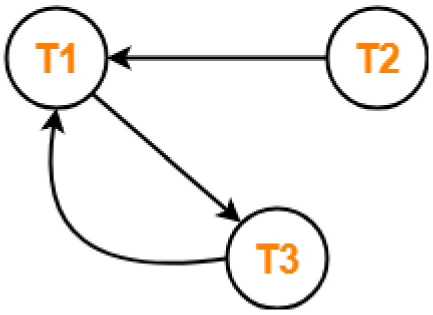
- There exists a blind write W_3(A) in the given schedule S. Therefore, the given schedule S may or may not be view serializable.

To check whether S is view serializable or not, let us derive the dependencies and then draw a dependency graph.

Drawing a Dependency Graph-

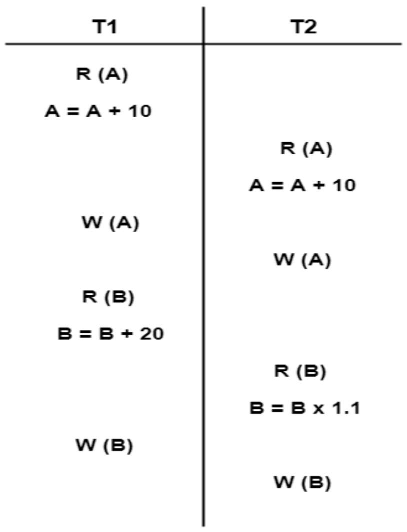
- T1 firstly reads A and T3 firstly updates A. So, T1 must execute before T3.
- Thus, we get the dependency **T1** → **T3**.
- Final updation on A is made by the transaction T1. So, T1 must execute after all other transactions.
- Thus, we get the dependency **(T2, T3)** → **T1**.
- There exists no write-read sequence.

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



- Clearly, there exists a cycle in the dependency graph.
- Thus, we conclude that the given schedule S is not view serializable.

Problem-03: : Check whether the given schedule S is view serializable or not-



T1	T2
R (A)	
A = A + 10	
	R (A)
	A = A + 10
W (A)	
	W (A)
R (B)	
B = B + 20	
	R (B)
	B = B x 1.1
W (B)	
	W (B)

Solution-

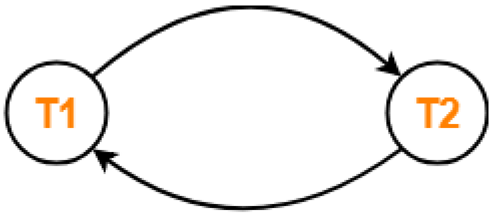
- if a schedule is conflict serializable, then it is surely view serializable.
- So, let us check whether the given schedule is conflict serializable or not.

Checking Whether S is Conflict Serializable Or Not-

Step-01: List all the conflicting operations and determine the dependency between the transactions-

➢ $R_1(A)$, $W_2(A)$	$(T_1 \rightarrow T_2)$
➢ $R_2(A)$, $W_1(A)$	$(T_2 \rightarrow T_1)$
➢ $W_1(A)$, $W_2(A)$	$(T_1 \rightarrow T_2)$
➢ $R_1(B)$, $W_2(B)$	$(T_1 \rightarrow T_2)$
➢ $R_2(B)$, $W_1(B)$	$(T_2 \rightarrow T_1)$

Step-02: Draw the precedence graph-



- There exists a cycle in the precedence graph. Therefore, the given schedule S is not conflict serializable.
- 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 check for blind writes.

Checking for Blind Writes-

- There exists no blind write in the given schedule S.
- Therefore, it is surely not view serializable.

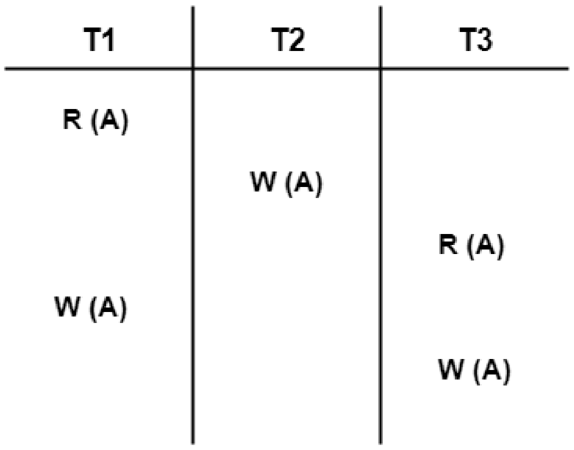
Alternatively,

- Given schedule S is not view serializable. This is because there exists no blind write in the schedule.
- No need to check for conflict serializability.

Problem-04: Check whether the given schedule S is view serializable or not. If yes, then give the serial schedule.

S : $R_1(A)$, $W_2(A)$, $R_3(A)$, $W_1(A)$, $W_3(A)$

Solution- Schedule can be represented pictorially as-



T1	T2	T3
R (A)		
	W (A)	
		R (A)
W (A)		
		W (A)

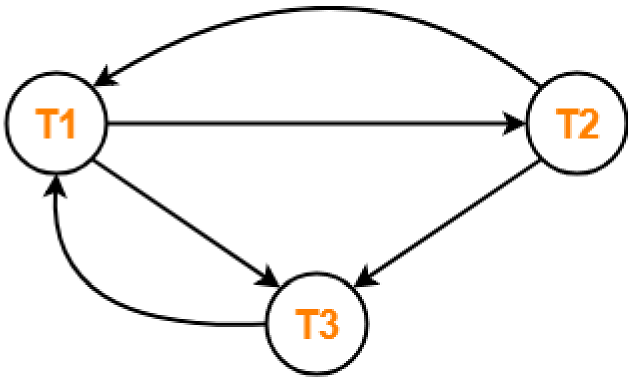
- if a schedule is conflict serializable, then it is surely view serializable.
- So, let us check whether the given schedule is conflict serializable or not.

Checking Whether S is Conflict Serializable Or Not-

Step-01: List all the conflicting operations and determine the dependency between the transactions-

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

Step-02: Draw the precedence graph-



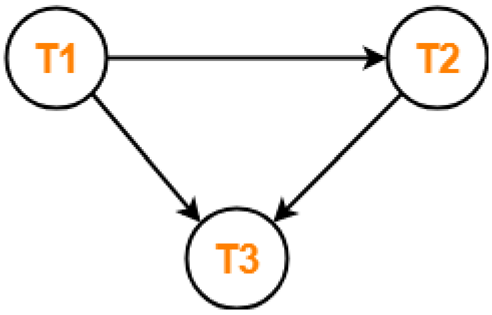
- there exists a cycle in the precedence graph.
- Therefore, the given schedule S is not conflict serializable.
- Since, the given schedule S is not conflict serializable, so, it may or may not be view serializable.
- To check whether S is view serializable or not, let us check for blind writes.

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.
- To check whether S is view serializable or not, let us derive the dependencies and then draw a dependency graph.

Drawing a Dependency Graph-

- T1 firstly reads A and T2 firstly updates A.
- So, T1 must execute before T2.
- Thus, we get the dependency **T1 → T2**.
- Final updation on A is made by the transaction T3.
- So, T3 must execute after all other transactions.
- Thus, we get the dependency **(T1, T2) → T3.**
- From write-read sequence, we get the dependency **T2 → T3**

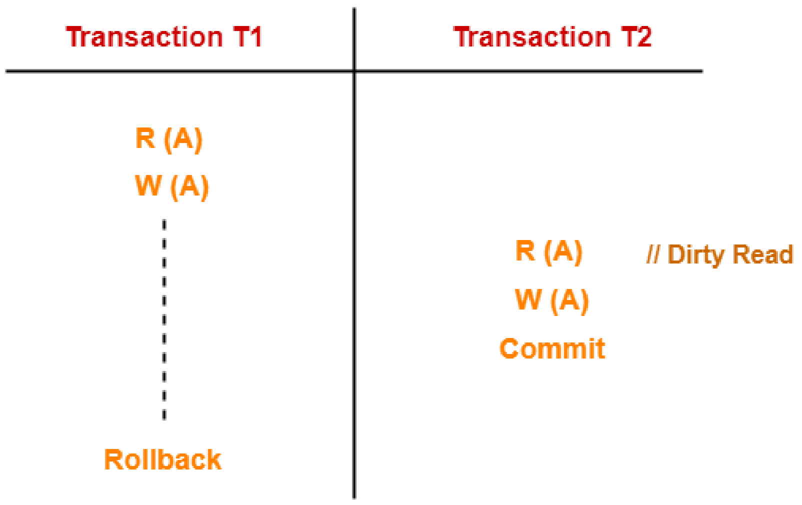


- Clearly, there exists no cycle in the dependency graph.
- Therefore, the given schedule S is view serializable.
- The serialization order T1 → T2 → T3.

Irrecoverable Schedules-

If in a schedule, A transaction performs a dirty read operation from an uncommitted transaction and commits before the transaction from which it has read the value then such a schedule is known as an **Irrecoverable Schedule**.

Example- Consider the following schedule-



Irrecoverable Schedule

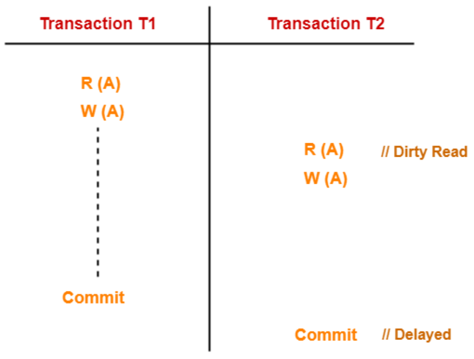
Here,

- T2 performs a dirty read operation.
- T2 commits before T1.
- T1 fails later and roll backs.
- The value that T2 read now stands to be incorrect.
- T2 can not recover since it has already committed.

Recoverable Schedules- **If in a schedule,**

- A transaction performs a dirty read operation from an uncommitted transaction
- And its commit operation is delayed till the uncommitted transaction either commits or roll backs then such a schedule is known as a **Recoverable Schedule**.
- The commit operation of the transaction that performs the dirty read is delayed.
- This ensures that it still has a chance to recover if the uncommitted transaction fails later.

Example- Consider the following schedule-



Recoverable Schedule

- T2 performs a dirty read operation.

- The commit operation of T2 is delayed till T1 commits or roll backs.
- T1 commits later.
- T2 is now allowed to commit.
- In case, T1 would have failed, T2 has a chance to recover by rolling back.

Checking Whether a Schedule is Recoverable or Irrecoverable-
Method-01: Check whether the given schedule is conflict serializable or not.

- If the given schedule is conflict serializable, then it is surely recoverable. If the given schedule is not conflict serializable, then it may or may not be recoverable. check using other methods.

Thumb Rules

- All conflict serializable schedules are recoverable.
- All recoverable schedules may or may not be conflict serializable.

Method-02: Check if there exists any dirty read operation. (Reading from an uncommitted transaction is called as a dirty read)

- If there does not exist any dirty read operation, then the schedule is surely recoverable.
- If there exists any dirty read operation, then the schedule may or may not be recoverable.
    - If there exists a dirty read operation, then follow the following cases-

    - Case-01:
        - If the commit operation of the transaction performing the dirty read occurs before the commit or abort operation of the transaction which updated the value, then the schedule is irrecoverable.

    - Case-02:
        - If the commit operation of the transaction performing the dirty read is delayed till the commit or abort operation of the transaction which updated the value, then the schedule is recoverable.

Thumb Rule

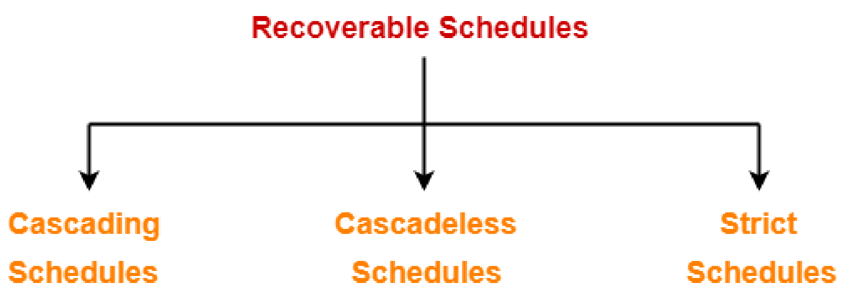
No dirty read means a recoverable schedule.

If in a schedule,

- A transaction performs a dirty read operation from an uncommitted transaction and its commit operation is delayed till the uncommitted transaction either commits or roll backs then such a schedule is called as a **Recoverable Schedule**.

Types of Recoverable Schedules- A recoverable schedule may be any one of these kinds:

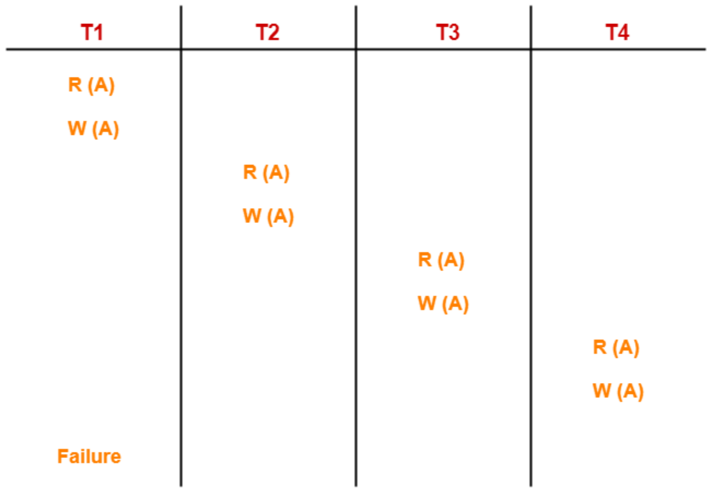
- **Cascading Schedule**
- **Cascadeless Schedule**
- **Strict Schedule**



Cascading Schedule-

- If in a schedule, failure of one transaction causes several other dependent transactions to rollback or abort, then such a schedule is called as a **Cascading Schedule** or **Cascading Rollback** or **Cascading Abort**.
- It simply leads to the wastage of CPU time.

Example-



Cascading Recoverable Schedule

➢ Transaction T2 depends on transaction T1.
➢ Transaction T3 depends on transaction T2.
➢ Transaction T4 depends on transaction T3.

In this schedule,

➢ The failure of transaction T1 causes the transaction T2 to rollback.
➢ The rollback of transaction T2 causes the transaction T3 to rollback.
➢ The rollback of transaction T3 causes the transaction T4 to rollback. Such a rollback is called as a **Cascading Rollback**.

NOTE- If the transactions T2, T3 and T4 would have committed before the failure of transaction T1, then the schedule would have been irrecoverable.

Cascadeless Schedule-

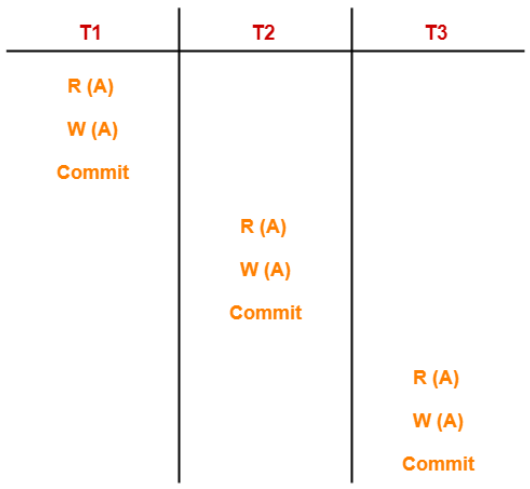
If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a **Cascadeless Schedule**.

In other words,

➢ Cascadeless schedule allows only committed read operations.

- Therefore, it avoids cascading roll back and thus saves CPU time.

Example-



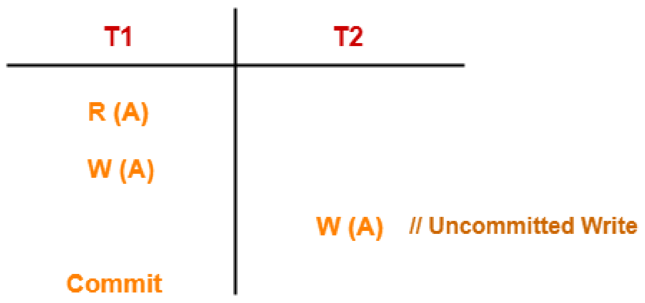
T1	T2	T3
R (A)		
W (A)		
Commit		
	R (A)	
	W (A)	
	Commit	
		R (A)
		W (A)
		Commit

Cascadeless Schedule

NOTE-

- Cascadeless schedule allows only committed read operations.
- However, it allows uncommitted write operations.

Example-



T1	T2
R (A)	
W (A)	
	W (A) // Uncommitted Write
Commit	

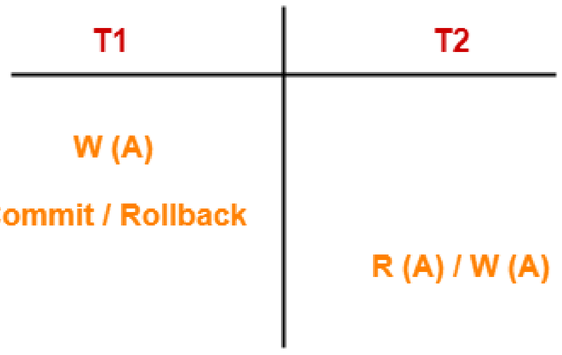
Cascadeless Schedule

Strict Schedule- If in a schedule, a transaction is neither allowed to read nor write a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a **Strict Schedule**.

In other words,

- Strict schedule allows only committed read and write operations.
- Clearly, strict schedule implements more restrictions than cascadeless schedule.

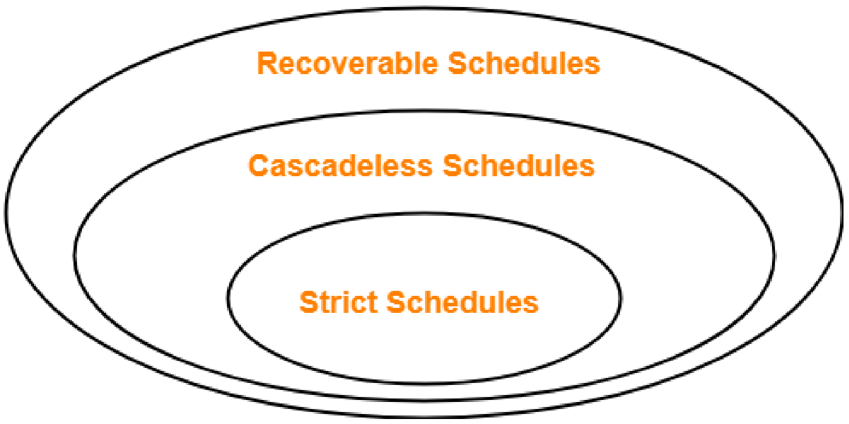
Example-



Strict Schedule

Note-

- Strict schedules are more strict than cascadeless schedules.
- All strict schedules are cascadeless schedules.
- All cascadeless schedules are not strict schedules.



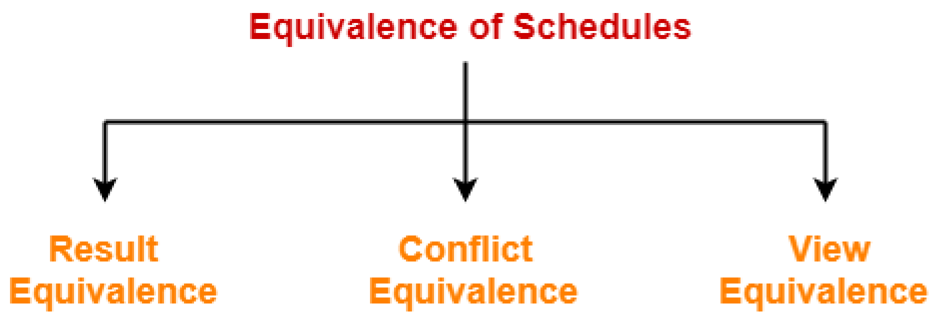
- Result Equivalence
- Conflict Equivalence
- View Equivalence

1. Result Equivalent Schedules-

- If any two schedules generate the same result after their execution, then they are called as result equivalent schedules.
- This equivalence relation is considered of least significance.
- This is because some schedules might produce same results for some set of values and different results for some other set of values.

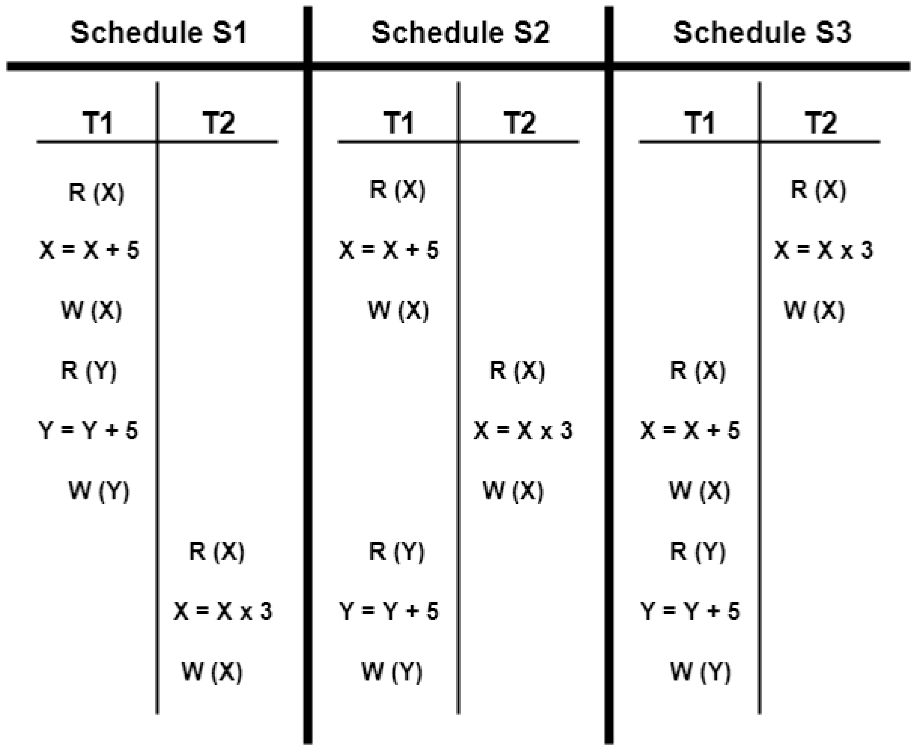
2. Conflict Equivalent Schedules- If any two schedules satisfy the following two conditions, then they are called as conflict equivalent schedules-

- The set of transactions present in both the schedules is same.
- The order of pairs of conflicting operations of both the schedules is same.



PRACTICE PROBLEMS BASED ON EQUIVALENCE OF SCHEDULES-

Problem-01: **Are the following three schedules result equivalent?**



Schedule S1		Schedule S2		Schedule S3	
T1	T2	T1	T2	T1	T2
R (X)		R (X)			R (X)
X = X + 5		X = X + 5			X = X x 3
W (X)		W (X)			W (X)
R (Y)			R (X)	R (X)	
Y = Y + 5			X = X x 3	X = X + 5	
W (Y)			W (X)	W (X)	
	R (X)	R (Y)		R (Y)	
	X = X x 3	Y = Y + 5		Y = Y + 5	
	W (X)	W (Y)		W (Y)	

Solution- **To check whether the given schedules are result equivalent or not,**

- consider some arbitrary values of X and Y.
- Then, compare the results produced by each schedule.
- Those schedules which produce the same results will be result equivalent.

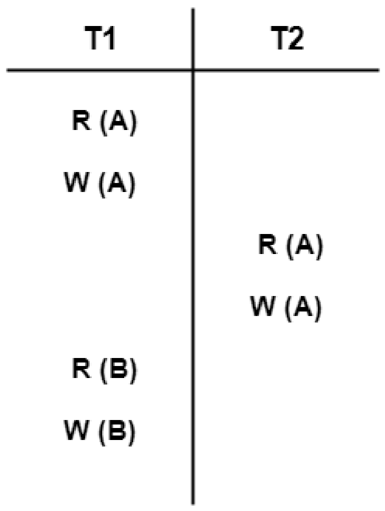
Let X = 2 and Y = 5. On substituting these values, the results produced by each schedule are-

- **Results by Schedule S1-** X = 21 and Y = 10
- **Results by Schedule S2-** X = 21 and Y = 10
- **Results by Schedule S3-** X = 11 and Y = 10

Clearly, the results produced by schedules S1 and S2 are same.

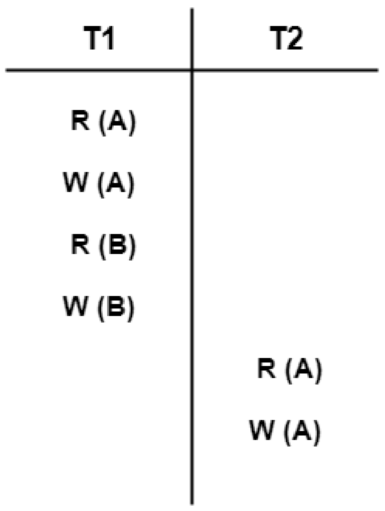
Thus, we conclude that S1 and S2 are result equivalent schedules.

Problem-02: Are the following two schedules conflict equivalent?



T1	T2
R (A)	
W (A)	
	R (A)
	W (A)
R (B)	
W (B)	

Schedule S1



T1	T2
R (A)	
W (A)	
R (B)	
W (B)	
	R (A)
	W (A)

Schedule S2

Solution- To check whether the given schedules are conflict equivalent or not,

- write their order of pairs of conflicting operations.
- compare the order of both the schedules.
- If both the schedules are found to have the same order, then they will be conflict equivalent.

For schedule S1- The required order is-

- $R_1(A)$, $W_2(A)$
- $W_1(A)$, $R_2(A)$
- $W_1(A)$, $W_2(A)$

For schedule S2- The required order is-

- $R_1(A)$, $W_2(A)$
- $W_1(A)$, $R_2(A)$
- $W_1(A)$, $W_2(A)$

Clearly, both the given schedules have the same order.

Thus, we conclude that S1 and S2 are conflict equivalent schedules.