Example:

A=1000

B=2000

```
R(A)-----1000----RAM
A=A-500-----500----CPU

W(A)---500(change in RAM)

R(B)----2000(read from DB)

B=B+500------CPU

W(B)-----2500(change in ram)

Commit;-----A=500, B=2500
```

ACID properties in transaction:

1. **Atomicity(Either all or none)**

```
Transaction T1:
R(A)
A=A-50
W(A)
R(B)
R(C):
Commit;
```

2. **Consistency(Before the transaction start and after the transaction completed)**

Sum of money should be same.

A=2000,B=3000

A+B=5000

```
Transaction T1:
R(A)-----2000 RAM
A=A-1000-------1000 CPU
W(A)------1000 RAM
R(B)----3000 RAM
B=B+1000------4000 CPU
W(B)-------4000 RAM
Commit; ----------Database
```

A=1000, B=4000

A+B=5000

3. **Isolation:**

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

T1	**T2**
R(A)	
W(A)	
	R(A)
	W(A)

T1-------------------T2

T1<--------------------T2

4. **Durability(permanent):**

Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Schedule:

It is chronological execution sequence of multiple transaction.

T1 T2 T3 T4 Tn

Schedule S(T1 T2 T3 T4 Tn)

Types of schedule:

I. Serial
II. Parallel

- **Serial Schedule (one by one)**

T1 T2 T3

T1 - - -		
	T2 - -	

	-	
		T3 - - -

Advantage: consistent

Disadvantage: performance degrade(waiting time). Throughput not good.

- **Parallel schedule**

T1 T2 T3

T1 - -		
	T2 - -	
		T3 - -
T1 -		
	T2 -	
		T3 -

Advantage: performance higher. Throughput high.

Disadvantage: consistency.

<u>Read Write Problem OR (Read Write conflict) OR (Unrepeatable read):</u>

R(A)	R(A)--------→	user 1 read A	user 2 read A
R(A)	W(A) --------→	user 1 read A	user 2 write A
W(A)	R(A) --------→	user 1 write A	user 2 read A
W(A)	W(A) --------→	user 1 write A	user 2 write A

a. **Read Read: No problem**

b. **Read Write:**

Example:

A(seats)=2----0

User1 T1	User2 T2

R(A)---2	
	R(A)----2
	A=A-2------0
	W(A)-------0
	Commit;----0
R(A)-----0	
W(A)----	
Commit;----	

Example:

Book copies: 10

T1	**T2**
R(A)----10	
A=A-1-----9	
	R(A)------10
	A=A-1----9
	W(A)-----9
	Commit;---9
W(A)----9	
Commit;	

c. Write Read:

Irrecoverable vs. Recoverable Schedule"

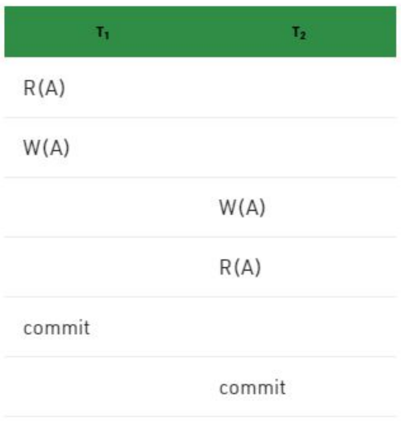
Recoverable schedule:

- One where no transaction needs to be rolled back.
- A schedule S is recoverable if no transaction T in S commits until all transactions T' that have written an item that T reads have committed.

Irrecoverable schedule:

- Recovery not possible.

Example 2: Consider the following schedule involving two transactions T_1 and T_2.



T_1	T_2
R(A)	
W(A)	
	W(A)
	R(A)
commit	
	commit

This is a recoverable schedule since T_1 commits before T_2, that makes the value read by T_2 correct.

Classification of schedule:

I. Serializability
II. Recoverability

A=10------10, B=20

Schedule S	
T1	T2
R(A)----10 A=A-5------5 W(A)------5 RAM - - - - R(B)----20 * Fail	 R(A)-----5 A=A-2----3 W(A)-----3 RAM Commit;---3 DB

Cascading Schedule vs. Cascade-less Schedule
Cascade:

- Due to occurrence of one event multiple events automatically occurring.

Schedules requiring cascaded rollback:

- A schedule in which uncommitted transactions that read an item from a failed transaction must be rolled back.

Example:
A=100

T1	T2	T3	T4

R(A)---100 A=A-50--50 W(A)----50 * Fail	R(A)---50	R(A)---50	R(A)---50

Cascade-less Schedule:

- One where every transaction reads only the items that are written by committed transactions .
- trying to stop read A by T2, T3 and T4. T2, T3 and T4 can read B, C and D.

Cascade:	Cascade-less:
T1: R(A), W(A); T2: R(A)	T1: R(A)----100, W(A)----50, Commit;; T2: R(A)

Cascade:

T1	T2

R(A)
W(A)
	R(A)

Cascade-less:

T1	T2
R(A)----100 W(A)----50 Commit;	R(A)

d. Write Write:

A=100---90-----80----100

T1	T2
R(A)---100	
	R(A)---100
A=A-10---90	
	A=A-20----80
W(A)---90	
	W(A)-----80
* Fail	

Strict Schedules:

- A schedule in which a transaction can neither read or write an item X until the last transaction that wrote X has committed.

Characterizing Schedules Based on Serializability:

Schedule:

- Collection of transaction.

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 has ability to become a serializable)

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

Result equivalent:

- Two schedules are called result equivalent if they produce the same final state of the database.

Conflict equivalent:

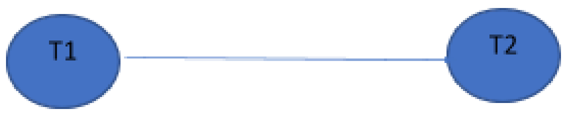
- Two schedules are said to be conflict equivalent if the order of any two conflicting operations is the same in both schedules.

Conflict serializable:

- A schedule S is said to be conflict serializable if it is conflict equivalent to some serial schedule S'.

Can we make the schedule into serializable?

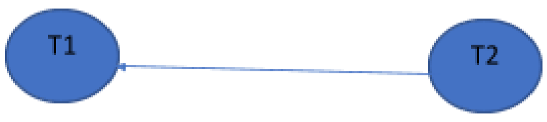
T1	T2
R(A)	
W(A)	
	R(A)
	W(A)



This is already serial.

T1----------→T2

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

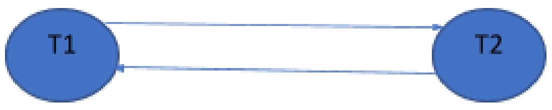


This is already serial.

T1←---------T2

Example:

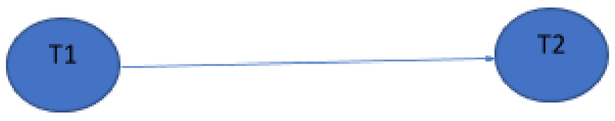
T1	T2



R(A)	
	R(A)
	W(A)
W(A)	

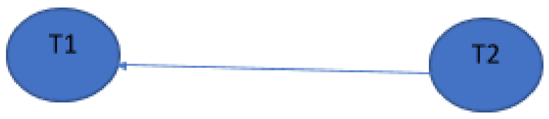
Solution:

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



T1----------□T2

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



T1□---------T2

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.
- It's not possible to determine when a schedule begins and when it ends (Hence, we reduce the problem of checking the whole schedule to checking only a **committed project** of the schedule (i.e. operations from only the committed transactions.)).

Methods of Serializable:

I. Conflict:
II. View:

Conflict:

- Through this we can check, is it possible to become a serializable schedule.

T1	T2	T3

	R(A)	
		R(A)
		W(A)
	W(A)	
R(B)		
W(B)		
	W(B)	

We have 3 transaction so 3! = 6 serializable.

T1□ T2□ T3

T1□ T3□ T2

T2□ T1□ T3

T2□ T3□ T1

T3□ T1□ T2

T3□ T2□ T1

Convert the above schedule into serializable if possible?

Conflict Equivalent:

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

Finding the conflict equivalent of above schedule?

Conflict: more than transaction work on same data. **No swap. No change.**

Non conflict: There is no problem both are work on different data. **Just swap the positions not operations.**

S ≡ S’

Schedule(S)		Schedule(S’)	
T1	T2	T1	T2
R(A)		R(A)	
W(A)		W(A)	
	R(A)	R(B)	
	W(A)		R(A)
R(B)			W(A)

We are going to check above 2 schedules are equivalent or not?

Solution:

In first glance both are different. Let’s check

R(A)	R(A)	non conflict pairs
R(A)	W(A)	Conflict pairs
W(A)	R(A)	
W(A)	W(A)	

R(B)	R(A)	non conflict pairs
W(B)	R(A)	
R(B)	W(A)	
W(A)	W(B)	

Firstly we should find out adjacent non-conflict pairs in schedule.

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

1st:

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

2nd:

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

Both are equal. **S ≡ S'**

S------conflict equivalent----□ S'----□ serializable.

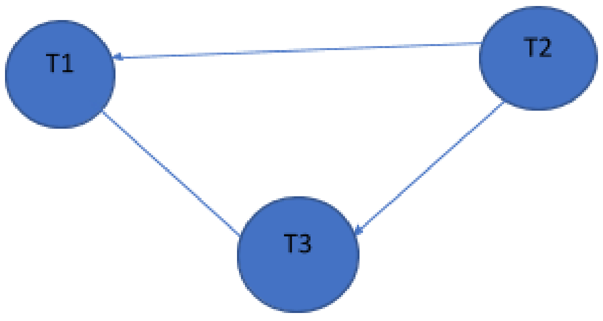
Conflict Serializable:

Schedule(S)		
T1	T2	T3

R(X)------W(X)
 	
R(Z)----
W(X)----
W(Z)-----
	
	R(Y)---W(Y)
	R(Z)---W(Z)
	
	W(Z)----R(Z), W(Z)
		
		R(Y)----W(Y)
		R(X)---W(X)
		
		W(Y)----R(Y), W(Y)

Check conflict pairs in other transactions and draw edges.

Precedences Graph:



Loop/ Cycle?

In above graph there is no loop no cycle.

It means this is serializable.

Conflict serializable schedule---� serializable--� consistent

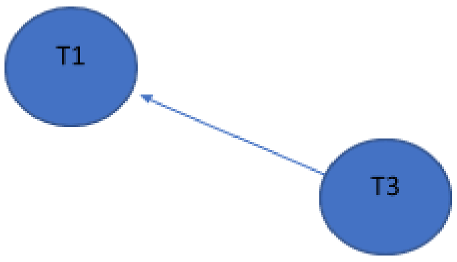
If this is serializable what are the execution(schedule) steps.

Check indegree == 0;

Check vertex(transaction) which has 0 indegree.

T2 is the vertex which has 0 indegree.

Remove T2 we have only 2 vertices. T2 execute 1st.



Again check indegree 0 from remaining 2 vertices. T1 has 0 indegree remove it from graph and 2nd remove T3. And last T1

T1	T2	T3

	1st	
		2nd
3rd		

Conflict serializable schedule-------□ serializable(T2□T3□T1)--------□ consistent