Schedule

What is schedule?

- A schedule is a **process of grouping the transactions** into one and **executing them in a predefined order**.
- A schedule is the **chronological (sequential) order in which instructions are executed** in a system.
- A schedule is required in a database because when some transactions execute in parallel, they may affect the result of the transaction.
- Means if one transaction is updating the values which the other transaction is accessing, then the order of these two transactions will change the result of another transaction.
- Hence a schedule is created to execute the transactions.

Example of schedule

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

Schedule Execution
A=B=1000
Read (1000)
A = 1000 - 50
Write (950)
Read (1000)
B = 1000 + 50
Write (1050)
Commit
Read (950)
temp = 950 * 0.1
A = 950 - 95
Write (855)
Read (1050)
B = 1050 + 95
Write (1145)
Commit

Example of schedule

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

Schedule Execution
A=B=1000
Read (1000)
Temp = 1000 * 0.1
A = 1000 - 100
Write (900)
Read (1000)
B = 1000 + 100
Write (1100)
Commit
Read (900)
A = 900 - 50
Write (850)
Read (1100)
B = 1100 + 50
Write (1150)
Commit

Serial schedule

- A serial schedule is a schedule in which **no transaction starts until a running transaction has ended**.
- A serial schedule is a schedule in which **one transaction is executed completely before starting another transaction**.
- Transactions are executed one after the other.
- This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

Example of Serial Schedule

Serial 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

Serial 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	

Non-serial Schedule (Interleaved Schedule)

- Schedule that **interleave the execution of different transactions**.
- Means **second transaction is started before the first one could end** and execution can switch between the transactions back and forth.
- It contains many possible orders in which the system can execute the individual operations of the transactions.

Example of Non-serial Schedule (Interleaved Schedule)

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

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

Equivalent Schedule

- If two schedules **produce the same result after execution**, they are said to be equivalent schedule.
- They may yield the same result for some value and different results for another set of values.
- That's why this equivalence is not generally considered significant.

Equivalent Schedule

Schedule-1 (A=B=1000)	
T1	**T2**
Read (A) A = A - 50 Write (A)	
	Read (A) temp = A * 0.1 A = A - temp Write (A)
Read (B) B = B + 50 Write (B) Commit	
	Read (B) B = B + temp Write (B) Commit

Both schedules are equivalent

In both schedules the sum "A + B" is preserved.

Schedule-2 (A=B=1000)	
T1	**T2**
Read (A) A = A - 50 Write (A) Read (B) B = B + temp Write (B) Commit	
	Read (A) temp = A * 0.1 A = A - temp Write (A) Read (B) B = B + 50 Write (B) Commit

Problems due to concurrent execution

➢ Dirty Read Problem

T_i	T_j
Read (A)	
Write (A)	
	Read (A)
	Commit
Commit	

Reading the value from any uncommitted transaction

➢ Lost Update Problem

T_i	T_j
Read (A)	
Write (A)	
	Write (A)
	Commit
Commit	

Blind Write

The value of variable A written by T_i is lost due to blind write upon variable A by T_j

➢ Unrepeatable Read Problem

T_i	T_j
Read (A)	
	Read (A)
Write (A)	
	Read (A)

T_j is reading the same variable (A) more than one time and getting different values

➢ Phantom Read Problem

T_i	T_j
Read (A)	
	Read (A)
Delete (A)	
	Read (A)

T_j is reading a variable which doesn't exist anymore.