LET'S START WITH DBMS :)

Schedule and its Types

Schedule : It refers to the sequence in which a set of concurrent/multiple transactions are executed. You can also say it as a sequence in which the operations (such as read, write, commit, and abort) of multiple transactions are executed. It is really helpful to ensure data consistency and integrity.
If there are T1, T2, T3....TN (n) transactions then the possible schedules= n! (n factorial)

Ex : Schedule sc1

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

LET'S START WITH DBMS :)

Schedule and its Types

Incomplete schedule : An incomplete schedule is one where not all transactions have reached their final state of either commit or abort.

T1:Read(A)
T1:Write(A)
T2:Read(B)
T2:Write(B)
T2:COMMIT

Here, T1 is still in progress as there is no COMMIT for transaction T1.

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

LET'S START WITH DBMS :)

Schedule and its Types

Complete schedule : A complete schedule is one where all the transactions in the schedule have either committed or aborted.

T1:Read(A)
T1:Write(A)
T1:COMMIT
T2:Read(B)
T2:Write(B)
T2:COMMIT

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

LET'S START WITH DBMS :)

Schedule and its Types

Types of Schedule

1. Serial Schedule
2. Concurrent or Non-Serial Schedule
3. Conflict-Serializable Schedule
4. View-Serializable Schedule
5. Recoverable Schedule
6. Cascadeless Schedule
7. Strict Schedule

LET'S START WITH DBMS :)

Schedule and its Types

1.Serial Schedule : A serial schedule is one where transactions are executed one after another. We can say it like if there are two transactions T1 and T2, T1 should commit to completeion before T2 starts.

Example : T1->T2
T1:Read(A)
T1:Write(A)
T1:COMMIT(T1)
T2:Read(B)
T2:Write(B)
T2:COMMIT(T2)

T2 starts only after T1 is completed/ commited.

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

LET'S START WITH DBMS :)

Schedule and its Types

Serial Schedule

Advantages :

1. It follows the ACID properties. Transactions are isolated from each other.
2. It maintains consistency.

Challenges:

1. Since there is poor throughput(no of transactions completed per unit time) and memory utilisation, this is not suggested as it can be can be inefficient.
2. Since wait time is high, less no of transactions are completed.

LET'S START WITH DBMS :)

Schedule and its Types

2. Non-Serial/Concurrent Schedule : A non-serial schedule is one where multiple transactions can execute simultaneously(operations of multiple transactions are alternate/interleaved executions). We can say it like if there are two transactions T1 and T2, T2 doesn't need to wait for T1 to commit, it can start at any point.

Example : T1 ,T2, T3

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

T2 didn't waited for T1 to commit

LET'S START WITH DBMS :)

Schedule and its Types

Non-Serial Schedule

Advantages :

1. Better resource utilization
2. Wait time is not involved. One transaction doesn't needs to wait for the running one to finish.
3. Throughput(no of transactions completed per unit time) is high

Challenges:

1. Consistentcy issue may arise because of non-serial execution. It requires robust concurrency control mechanisms to ensure data consistency and integrity.
2. We can use Serializability and Concurrency Control Mechanisms to ensure consistency.

LET'S START WITH DBMS :)

Schedule and its Types

3. Conflict-Serializable Schedule : A schedule is conflict-serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.

R(A) T1 & R(A) T2: No conflict (both reads)
R(A) T1 & W(A) T2: RW conflict (T2 reads A after T1 writes A)
W(A) T1 & W(A) T2: WW conflict (both write to A)

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

Conflict pairs : Read-Write, Write-Read , Write-Write(perfomed on the same data item)
Non-conflict pairs :
Read-Read (perfomed on the same data item),
Read-Read (perfomed on the different data item),
Write-Write(perfomed on the different data item)

LET'S START WITH DBMS :)

Schedule and its Types

4. View-Serializable Schedule : A schedule is view-serializable if it is equivalent to some serial schedule with respect to the view of the database.
This means that the result of the read operations in the schedule must be consistent with the result of reading from a serial execution.

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

LET'S START WITH DBMS :)

Schedule and its Types

5. Recoverable Schedule : It ensures that transactions do not commit based on uncommitted data from other transactions.

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

Recoverable Schedule

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
	COMMIT
FAIL	

Irrecoverable Schedule

LET'S START WITH DBMS :)

Schedule and its Types

6. Cascadeless Schedule : This schedule happens when the failure or abort of one transaction causes a series of other transactions to also abort.

- T1 Writes to A: T1 writes to data item A
- T2 Reads A: T2 reads the uncommitted value of A written by T1

Now, if T1 fails and aborts, T2 must also abort because it has read an uncommitted value from T1.
Consequently, T3 must abort as well

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

Cascadeless Schedule

Issues:

1. Performance degradation because multiple transactions need to be rolled back
2. Improper CPU resource utilisation

LET'S START WITH DBMS :)

Schedule and its Types

Cascading Schedule : It ensures transactions only read committed data, such that the abort of one transaction does not lead to the abort of other transactions that have already read its uncommitted changes.

- T1 Writes to A: T1 writes to data item A
- T2 Reads A: T2 reads the committed value of A

Ensuring there is no write read problem(dirty read)
Also, T2 does not read any uncommitted data,
there are no cascading aborts in this schedule.

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

Cascading Schedule

Issues :

1. Write-Write problem is encountered.
2. Performance overhead is there

LET'S START WITH DBMS :)

Schedule and its Types

7. Strict Schedule : It ensures transaction is not allowed to read or write a data item that another transaction has written until the first transaction has either committed or aborted. It prevents cascading aborts.

- T2 cannot read or write the value of A until T1 has committed.
- This ensures that T2 only sees the committed value of A

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

Strict Schedule