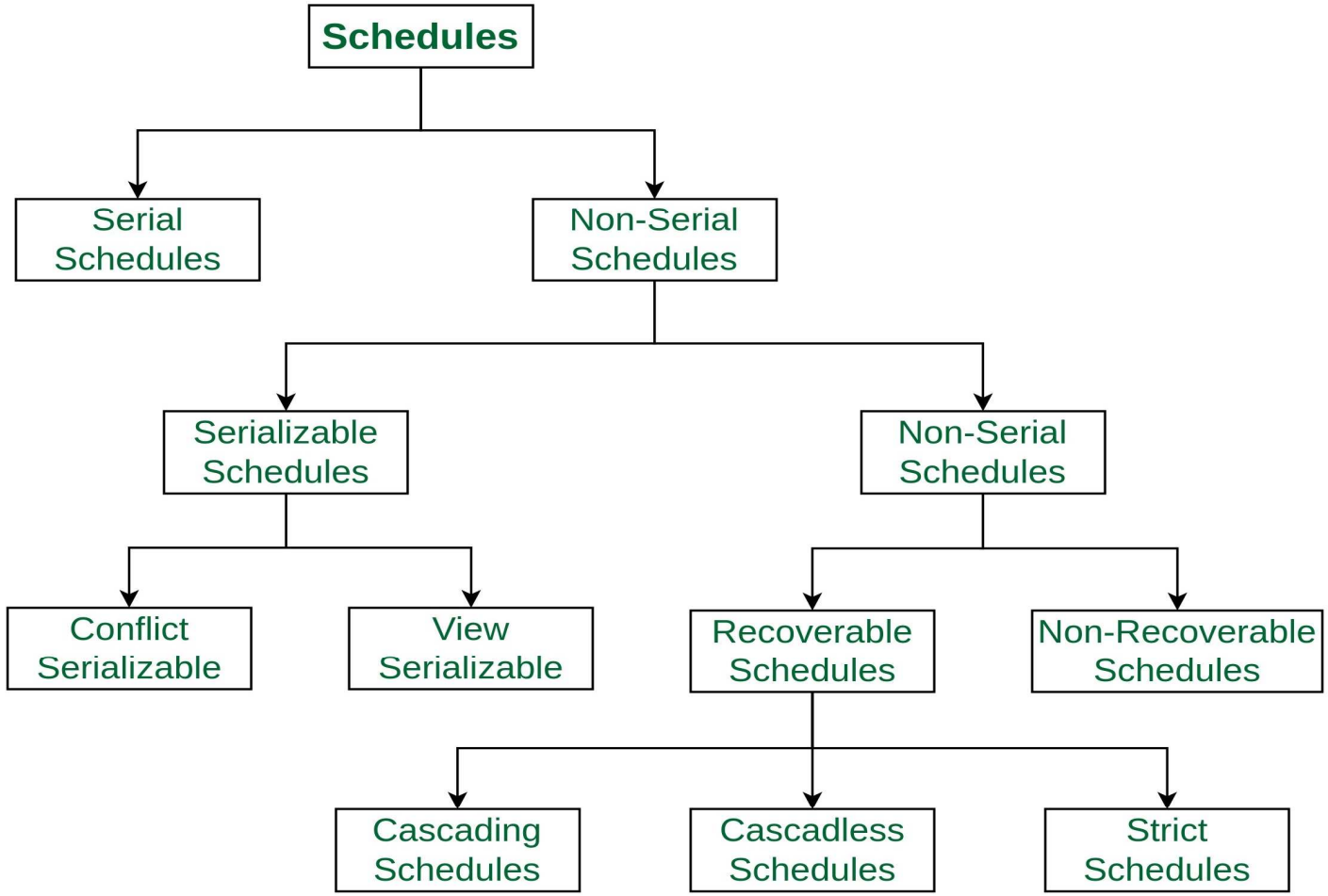
UNIT V - FAILURE RECOVERY AND CONCURRENCY CONTROL

PRASHANT TOMAR

SERIAL SCHEDULE

- Schedule, as the name suggests, is a process of lining the transactions and executing them one by one. When there are multiple transactions that are running in a concurrent manner and the order of operation is needed to be set so that the operations do not overlap each other, Scheduling is brought into play and the transactions are timed accordingly.
- The basics of Transactions and Schedules is discussed in Concurrency Control (Introduction), and Transaction Isolation Levels in DBMS articles.
- Schedules in which the transactions are executed non-interleaved, i.e., a serial schedule is one in which no transaction starts until a running transaction has ended are called serial schedules.

Types of schedules in DBMS



- Consider the following schedule involving two transactions T_1 and T_2.

T_1	T_2
R(A)	
W(A)	
R(B)	
	W(B)
	R(A)
	R(B)

- Where R(A) denotes that a read operation is performed on some data item ‘A’
- This is a serial schedule since the transactions perform serially in the order

  T_1 —> T_2

NON-SERIAL SCHEDULE

- This is a type of Scheduling where the operations of multiple transactions are interleaved. This might lead to a rise in the concurrency problem. The transactions are executed in a non-serial manner, keeping the end result correct and same as the serial schedule. Unlike the serial schedule where one transaction must wait for another to complete all its operation, in the non-serial schedule, the other transaction proceeds without waiting for the previous transaction to complete. This sort of schedule does not provide any benefit of the concurrent transaction. It can be of two types namely, Serializable and Non-Serializable Schedule.
- The Non-Serial Schedule can be divided further into Serializable and Non-Serializable.

SERIALIZABLE SCHEDULE

- This is used to maintain the consistency of the database. It is mainly used in the Non-Serial scheduling to verify whether the scheduling will lead to any inconsistency or not. On the other hand, a serial schedule does not need the serializability because it follows a transaction only when the previous transaction is complete.
- The non-serial schedule is said to be in a serializable schedule only when it is equivalent to the serial schedules, for an n number of transactions. Since concurrency is allowed in this case thus, multiple transactions can execute concurrently.
- A serializable schedule helps in improving both resource utilization and CPU throughput.

- A serializable schedule always leaves the database in consistent state. A **serial schedule** is always a serializable schedule because in serial schedule, a transaction only starts when the other transaction finished execution. However a non-serial schedule needs to be checked for Serializability. These are of two types:

1. **Conflict Serializable:** A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations. Two operations are said to be conflicting if all conditions satisfy : They belong to different transactions -

- They operate on the same data item
- At Least one of them is a write operation
- At Least one of them is a write operation

Example

- **Conflicting** operations pair ($R_1(A)$, $W_2(A)$) because they belong to two different transactions on same data item A and one of them is write operation.
- Similarly, ($W_1(A)$, $W_2(A)$) and ($W_1(A)$, $R_2(A)$) pairs are also **conflicting**.
- On the other hand, ($R_1(A)$, $W_2(B)$) pair is **non-conflicting** because they operate on different data item.
- Similarly, (($W_1(A)$, $W_2(B)$) pair is **non-conflicting.**

2. **View Serializable:** A Schedule is called view serializable if it is view equal to a serial schedule (no overlapping transactions). A conflict schedule is a view serializable but if the serializability contains blind writes, then the view serializable does not conflict serializable.

- Two schedules S1 and S2 are said to be view equal if below conditions are satisfied :

1) **Initial Read**

- If a transaction T1 reading data item A from initial database in S1 then in S2 also T1 should read A from initial database.

- Transaction T2 is reading A from initial database.

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

2)**Updated Read**

If Ti is reading A which is updated by Tj in S1 then in S2 also Ti should read A which is updated by Tj.

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

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

- Above two schedule are not view equal as in S1 :T3 is reading A updated by T2, in S2 T3 is reading A updated by T1.

3)Final Write operation

If a transaction T1 updated A at last in S1, then in S2 also T1 should perform final write operations.

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

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

- Above two schedule are not view as Final write operation in S1 is done by T1 while in S2 done by T2.