Schedules

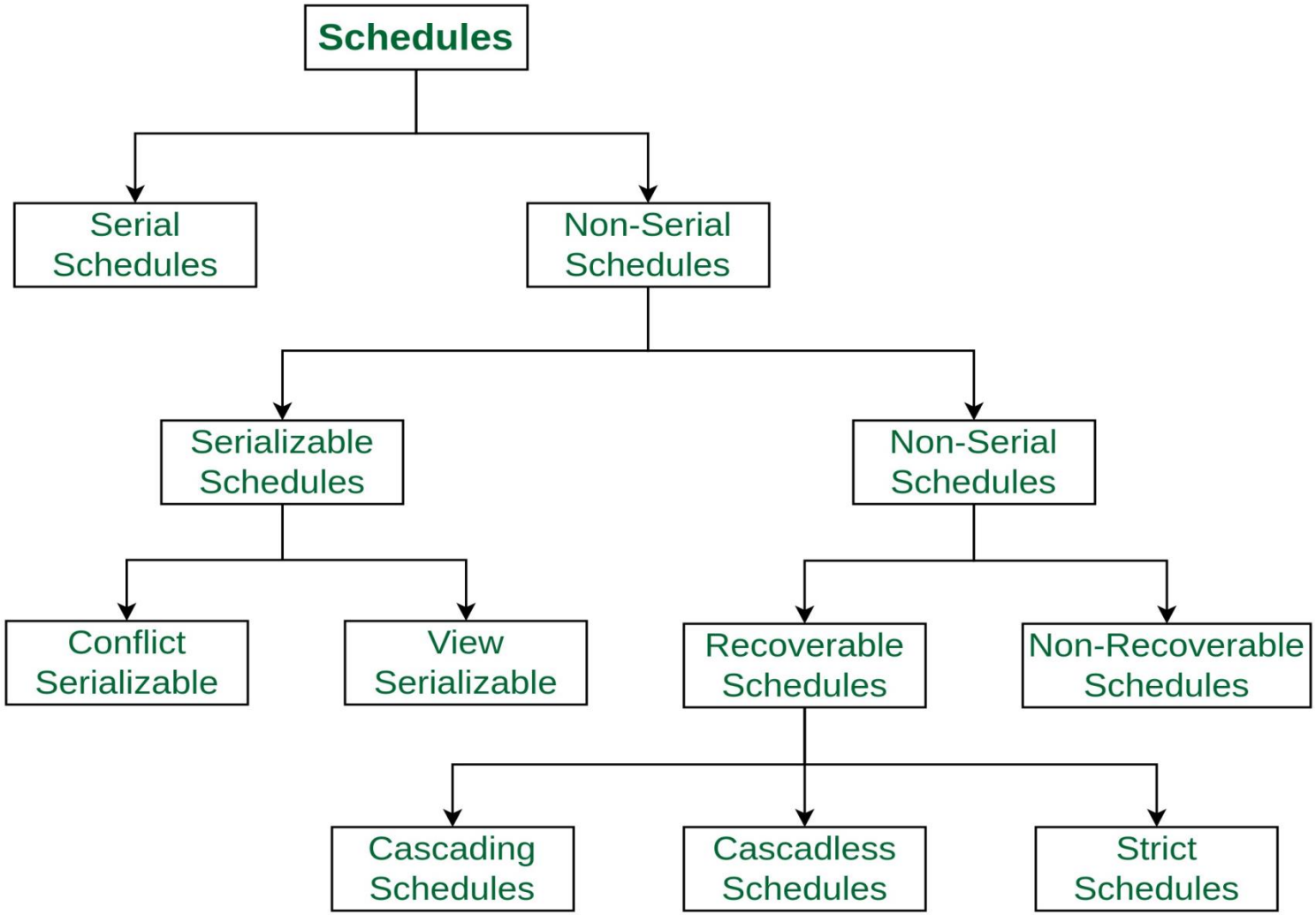
Schedule

- It is the order of operations from different transactions executing one by one.

- S1:R1(X),R2(X),W2(X)W1(X)
- 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,

- S2: R1(X)W1(X)R2(X)W2(X)

Types of schedules in DBMS



Serial Schedules:

- 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.
  - EXAMPLE
  - S3: R1(X)W1(X) C1 R2(X)W2(X)C2

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.
- Two types-
- Serializable and Non-Serializable Schedule.

Serializable

- 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.

Serializable Schedules

- -Conflict serializable
- -View serializable

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

View serializable

- A Schedule is called view serializable if it is view equallent 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.
  - S5:R1(X),W1(X),W2(X) C1 C2

Based on Recoverability

- Recoverable and Non-recoverable Schedule.

- **Recoverable Schedule:**
- Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules.
  - if some transaction T_j is reading value updated or written by some other transaction T_i, then the commit of T_j must occur after the commit of T_i.

Types of Recoverable schedule

- There can be three types of recoverable schedule:
- **Cascading Schedule:** Also called Avoids cascading aborts/rollbacks (ACA).
- When there is a failure in one transaction and this leads to the rolling back or aborting other dependent transactions, then such scheduling is referred to as Cascading rollback or cascading abort.

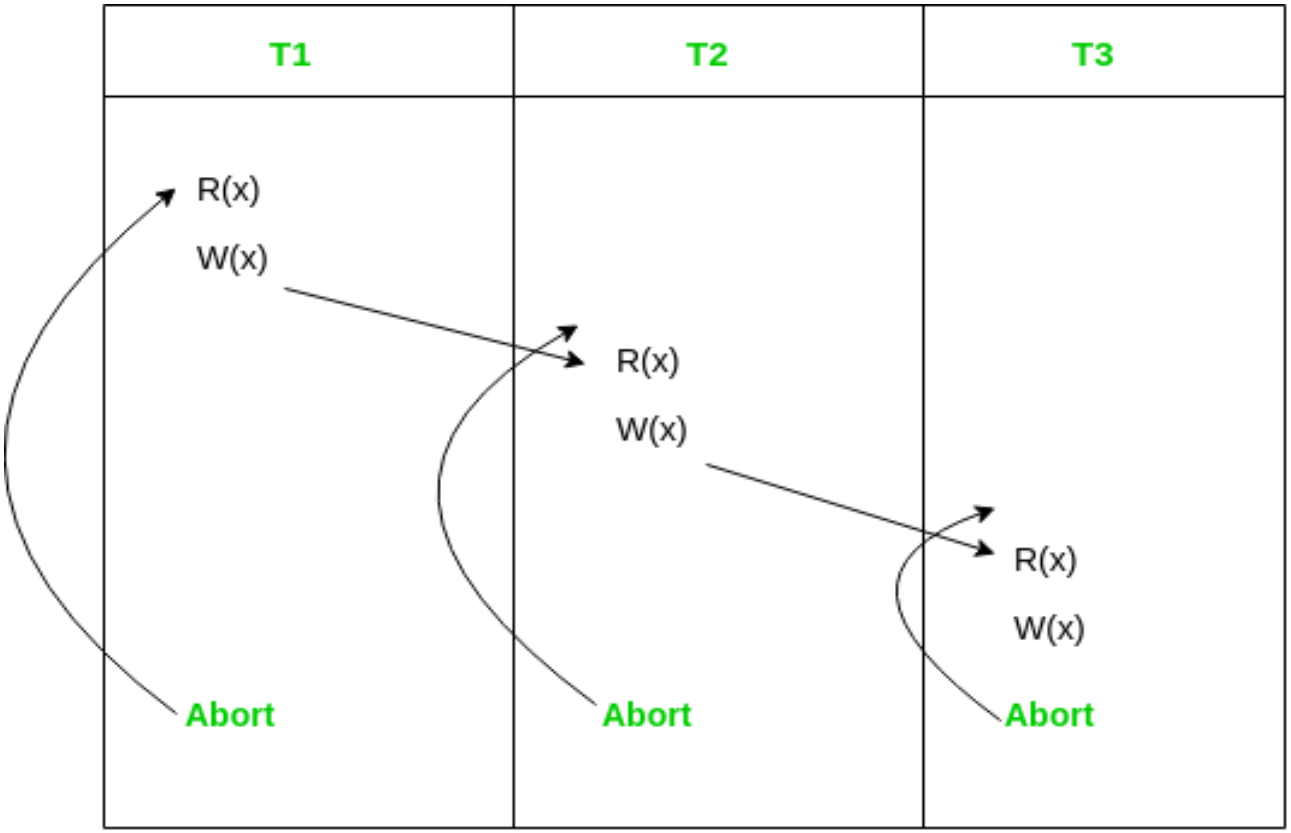


Figure - Cascading Abort

Cascadeless Schedule:

- In other words, if some transaction T_j wants to read value updated or written by some other transaction T_i, then the commit of T_j must read it after the commit of T_i.

T1	***T2***
R1(x)	
W1(x)	
	R2(X)
C1	
	W1(X)
	C2

Cascadeless Schedule

T1	T2
R1(A)	
W1(A)	
	W2(A)
C1	
	R2(A)
	C2

Cascadeless Schedule

T1	T2
R1(A)	
W1(A)	
	R2(A)
	W2(A)
A1	
	A2

It is a recoverable schedule but it does not avoid cascading aborts. It can be seen that if T_1 aborts, T_2 will have to be aborted too in order to maintain the correctness of the schedule as T_2 has already read the uncommitted value written by T1

Strict Schedule:

- A schedule is strict if for any two transactions T_i, T_j, if a write operation of T_i precedes a conflicting operation of T_j (either read or write), then the commit or abort event of T_i also precedes that conflicting operation of T_j.
-

  In other words, T_j can read or write updated or written value of T_i only after T_i commits/aborts.

T1	T2
R1(A)	
	R2(A)
W1(A)	
C1	
	W2(A)
	R2(A)
	C2

- S21:R1(X),R2(X),W1(X),R1(Y),W1(Y),C1,W2(X),C2;

- S21:R1(X),R2(X),W1(X),R1(Y),W1(Y),C1,W2(X),C2;-Strict

- Read/Write operation of a transaction Ti(T2) succeeds after a write operation of Tj(T1), so Read/write operation of Ti(t1) should come only after the commit operation of Ti

Example

- S22:R1(X),R2(X),W1(X),R1(Y),W1(Y),W2(X),c1,C2;-

Example

- S22:R1(X),R2(X),W1(X),R1(Y),W1(Y),W2(X),c1,C 2;-Cascadeless