Schedules

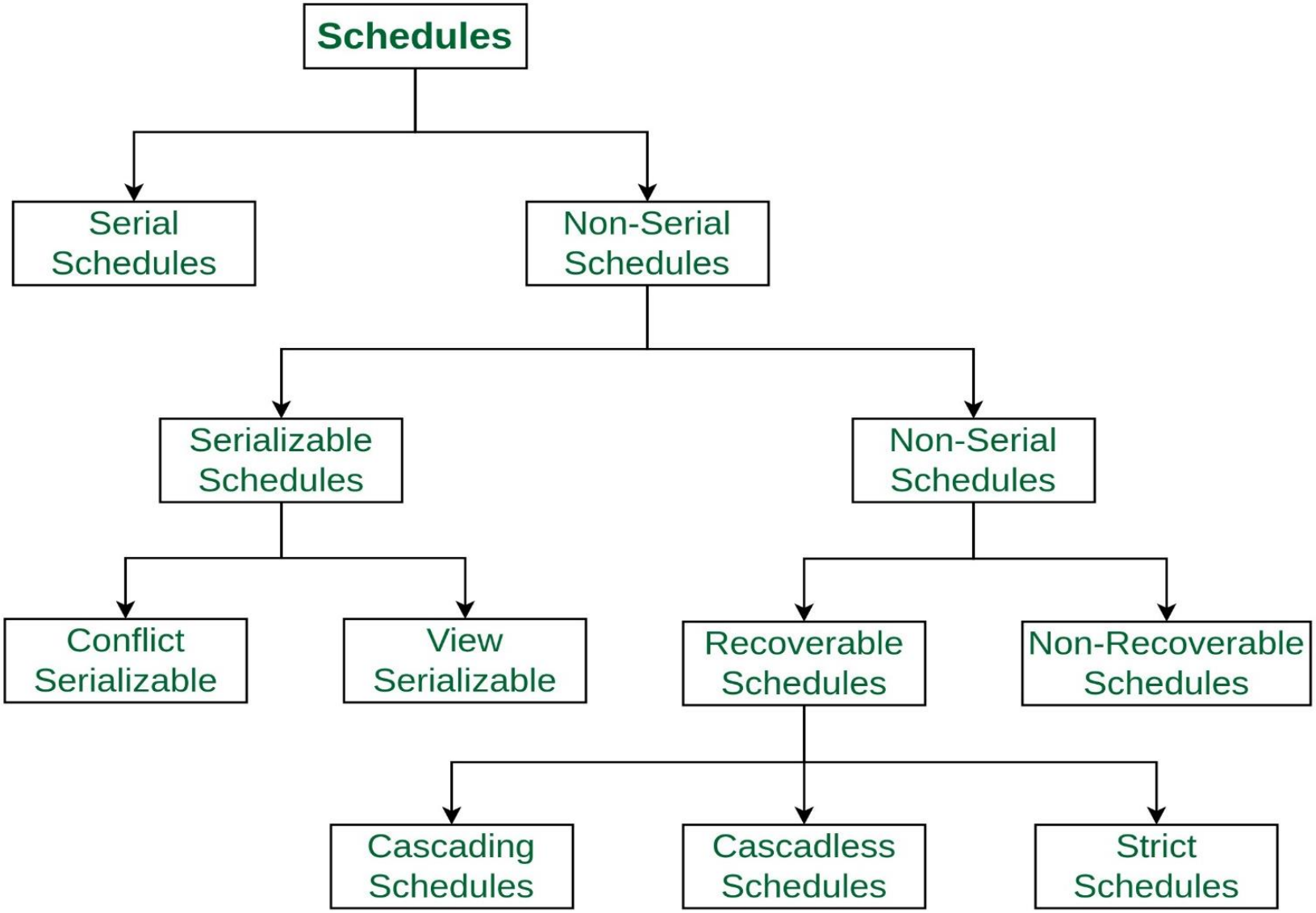
Schedules

- A schedule is a process of lining the transactions and executing them one by one
- It is a sequence of read, write, abort and commit operations from a set of transactions.
  - Each schedule must preserve the order of the operations in each of the individual transactions
  - A transaction comprises a sequence of operations consisting of read and/or write actions to the database followed by a commit or abort action

Example:

S_1:R1(A) W1(A) R2(A) W2(A) R1(B) W1(B) R2(B) W2(B) C2 C1

Types of Schedules



Types of Schedules

Serial Schedules

- A serial schedule is a schedule were the operations of each transaction are executed consecutively without any interleaved operations from other transactions (suspension).
  - The transactions are performed in serial order.
  - The transactions are executed non-interleaved, i.e., no transaction starts until a running transaction has ended

Types of Schedules

Serial Schedules

Example

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

T1	T2

read(A);
A := A – N;
write(A);
read(B);
B := B + N;
write(B);
	read(A);
	A := A + M;
	write(A);

Types of Schedules

Non-Serial Schedules

- Non-Serial Schedule 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.
- The Non-Serial Schedule can be divided further into two:
  - Serializable schedules
  - Non-Serializable schedules

Types of Schedules

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.
- If a set of transactions execute concurrently, we say that the (nonserial) schedule is correct if it produces the same results as some serial execution, hence such schedule is called serializable.

Types of Schedules

Serializable Schedule

- 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.
- These are of two types:
  - Conflict Serializable
  - View Serializable

Types of Schedules

Conflict Serializable

- A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.
- **Recall:** Two operations are said to be conflicting if they satisfy the following conditions :
  1. They belong to different transactions
  2. They operate on the same data item
  3. At Least one of them is a write operation

Types of Schedules

Example ~ Conflict Serializable

Consider the following schedule:

- S1: $R_1(A), W_1(A), R_2(A), W_2(A), R_1(B), W_1(B), R_2(B), W_2(B)$
- If O_i and O_j are two operations in a transaction and $O_i < O_j$ (O_i is executed before O_j), same order will follow in schedule as well.

- Using this property, we can get two transactions of schedule S1 as:
  - T1: $R_1(A), W_1(A), R_1(B), W_1(B)$
  - T2: $R_2(A), W_2(A), R_2(B), W_2(B)$

Types of Schedules

Example ~ Conflict Serializable

Possible Serial Schedules are: T1->T2 or T2->T1

- **Swapping non-conflicting operations** $R_2(A)$ and $R_1(B)$ in S1, the schedule becomes:
  - S1: $R_1(A)$, $W_1(A)$, $R_2(A)$, $W_2(A)$, $R_1(B)$, $W_1(B)$, $R_2(B)$, $W_2(B)$
  - S11: $R_1(A)$, $W_1(A)$, $R_1(B)$, $W_2(A)$, $R_2(A)$, $W_1(B)$, $R_2(B)$, $W_2(B)$
- Similarly, **swapping non-conflicting operations** $W_2(A)$ and $W_1(B)$ in S11, the schedule becomes:
  - S11: $R_1(A)$, $W_1(A)$, $R_1(B)$, $W_2(A)$, $R_2(A)$, $W_1(B)$, $R_2(B)$, $W_2(B)$
  - **S12:** $R_1(A)$, $W_1(A)$, $R_1(B)$, $W_1(B)$, $R_2(A)$, $W_2(A)$, $R_2(B)$, $W_2(B)$
- S12 is a serial schedule in which all operations of T1 are performed before starting any operation of T2. Since S has been transformed into a serial schedule S12 by swapping non-conflicting operations of S1, S1 is conflict serializable.

Types of Schedules

View Serializable

- A schedule will view serializable if it is view equivalent to a serial schedule.
  - If a schedule is conflict serializable, then it will be view serializable.
  - The view serializable which does not conflict serializable contains blind writes.

Types of Schedules

View Serializable

What is a View Equivalent Schedule?

- Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:

1.Initial Read

  - An initial read of both schedules must be the same.
  - Suppose two schedule S1 and S2. In schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

Types of Schedules

View Serializable

What is a View Equivalent Schedule?

- The two schedules below are view equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1

T1	T2
Read(A)	
	Write(A)

Schedule S1

T1	T2
	Write(A)
Read(A)	

Schedule S2

Types of Schedules

View Serializable

What is a View Equivalent Schedule?

2.Updated Read

- In schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated by Tj.

Types of Schedules

View Serializable

What is a View Equivalent Schedule?

- The two schedules below are not view equal because, in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

T1	T2	T3
Write(A)		
	Write(A)	
		Read(A)

Schedule S1

T1	T2	T3
	Write(A)	
Write(A)		
		Read(A)

Schedule S2

Types of Schedules

View Serializable

What is a View Equivalent Schedule?

3.Final Write

- A final write must be the same between both the schedules.
- In schedule S1, if a transaction T1 updates A at last then in S2, final writes operations should also be done by T1.

Types of Schedules

View Serializable

What is a View Equivalent Schedule?

- The two schedules below are view equal because Final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

T1	T2	T3
Write(A)	Read(A)	
		Write(A)

Schedule S1

T1	T2	T3
	Read(A)	
Write(A)		
		Write(A)

Schedule S2

Types of Schedules

Non-Serializable

- The non-serializable schedule is divided into two types:
  - Recoverable Schedule and
  - Non-recoverable Schedule.

Types of Schedules

Recoverable Schedule

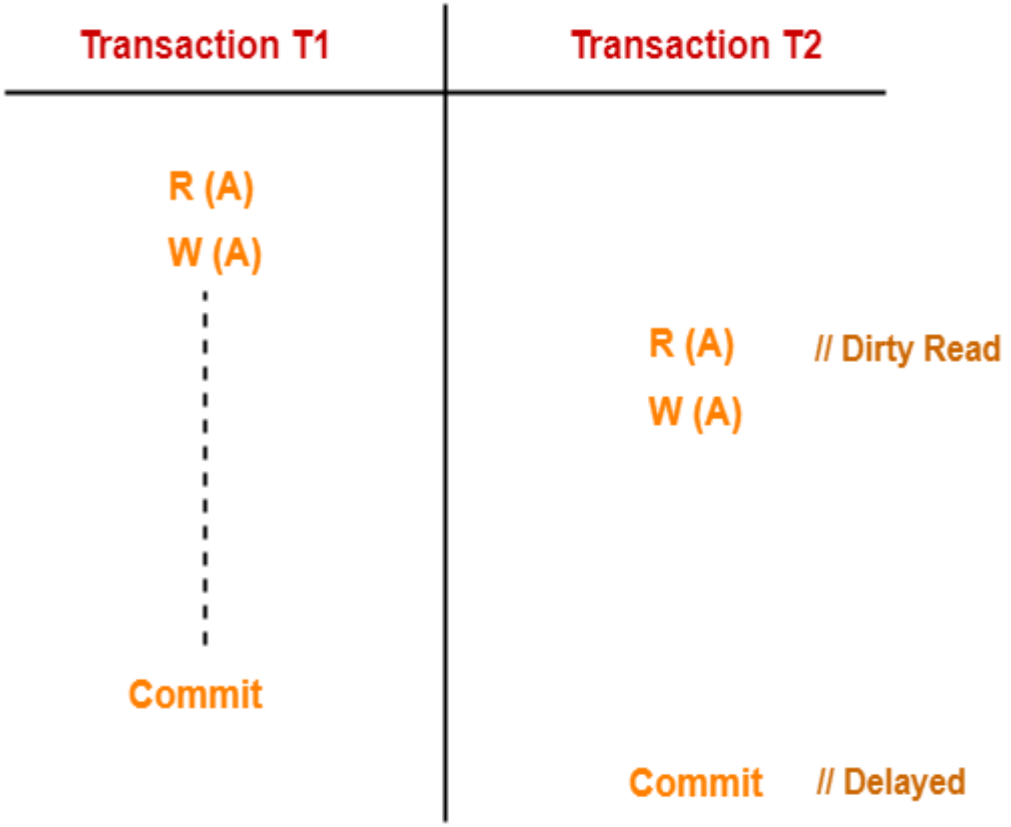
- Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules.
- In other words, if some transaction T_j is reading value updated or written by some other transaction T_i, then the commit of T_j must occur after the commit of T_i.

Types of Schedules

Recoverable Schedule

Example

- Consider the following schedule involving two transactions T_1 and T_2
- This is a recoverable schedule since T_1 commits before T_2, that makes the value read by T_2 correct.



Types of Schedules

Recoverable Schedule

- There can be three types of recoverable schedule:
  - Cascading Schedule
  - Cascadeless Schedule
  - Strict Schedule

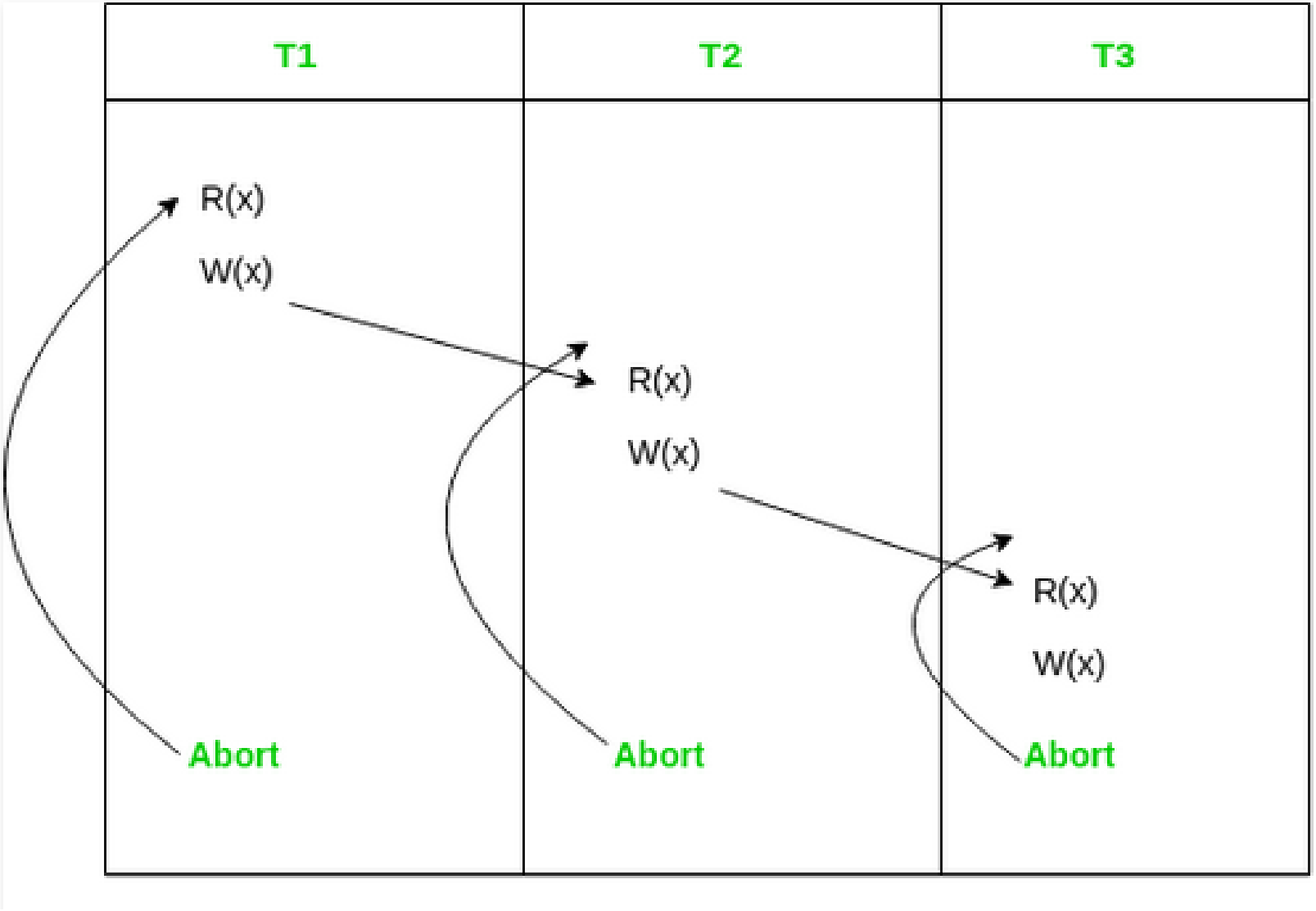
Types of Schedules

Cascading Schedule

- 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

Types of Schedules

Example of Cascading Schedule



Types of Schedules

Cascadeless Schedule

- Also called Avoids cascading aborts/rollbacks (ACA).
- Schedules in which transactions read values only after all transactions whose changes they are going to read commit are called cascadeless schedules.
- Avoids that a single transaction abort leads to a series of transaction rollbacks.
- A strategy to prevent cascading aborts is to disallow a transaction from reading uncommitted changes from another transaction in the same schedule.

Types of Schedules

Cascadeless Schedule

Example:

- Consider the following schedule involving three transactions T_1 and T_2.
- This schedule is cascadeless Since the updated value of **A** is read by T_2 only after the updating transaction i.e. T_1 commits.

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

Types of Schedules

Cascadeless Schedule

Example:

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

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
abort	
	abort

Types of Schedules

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.

Types of Schedules

Strict Schedule

Example:

- Consider the following schedule involving two trasactions T_1 and T_2.
- This is a strict schedule since T_2 reads and writes A which is written by T_1 only after the commit of T_1.

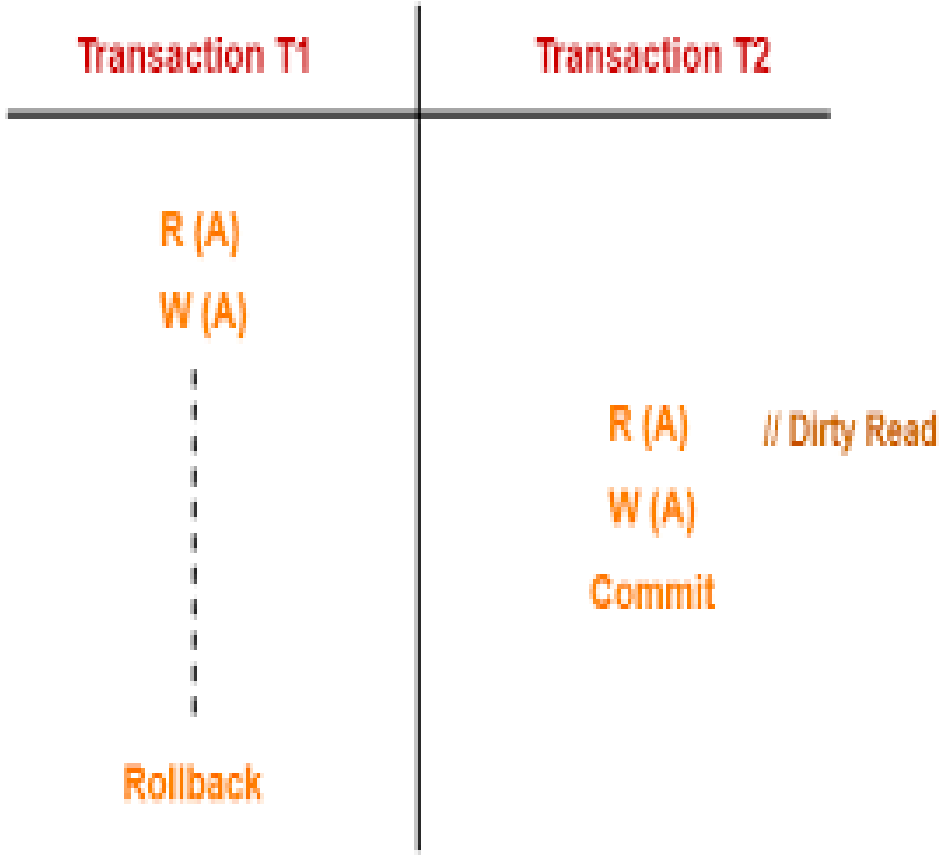
T_1	T_2
R(A)	
	R(A)
W(A)	
commit	
	W(A)
	R(A)
	commit

Types of Schedules

Non-Recoverable Schedule:

Using an Example:

- Consider the following schedule involving two transactions T_1 and T_2.
- T_2 read the value of A written by T_1, and committed. T_1 later aborted, therefore the value read by T_2 is wrong, but since T_2 committed, this schedule is **non-recoverable**



References

- https://www.geeksforgeeks.org/types-of-schedules-in-dbms/
- http://www.ccs.neu.edu/home/kathleen/classes/cs3200/9-Transactions.pdf