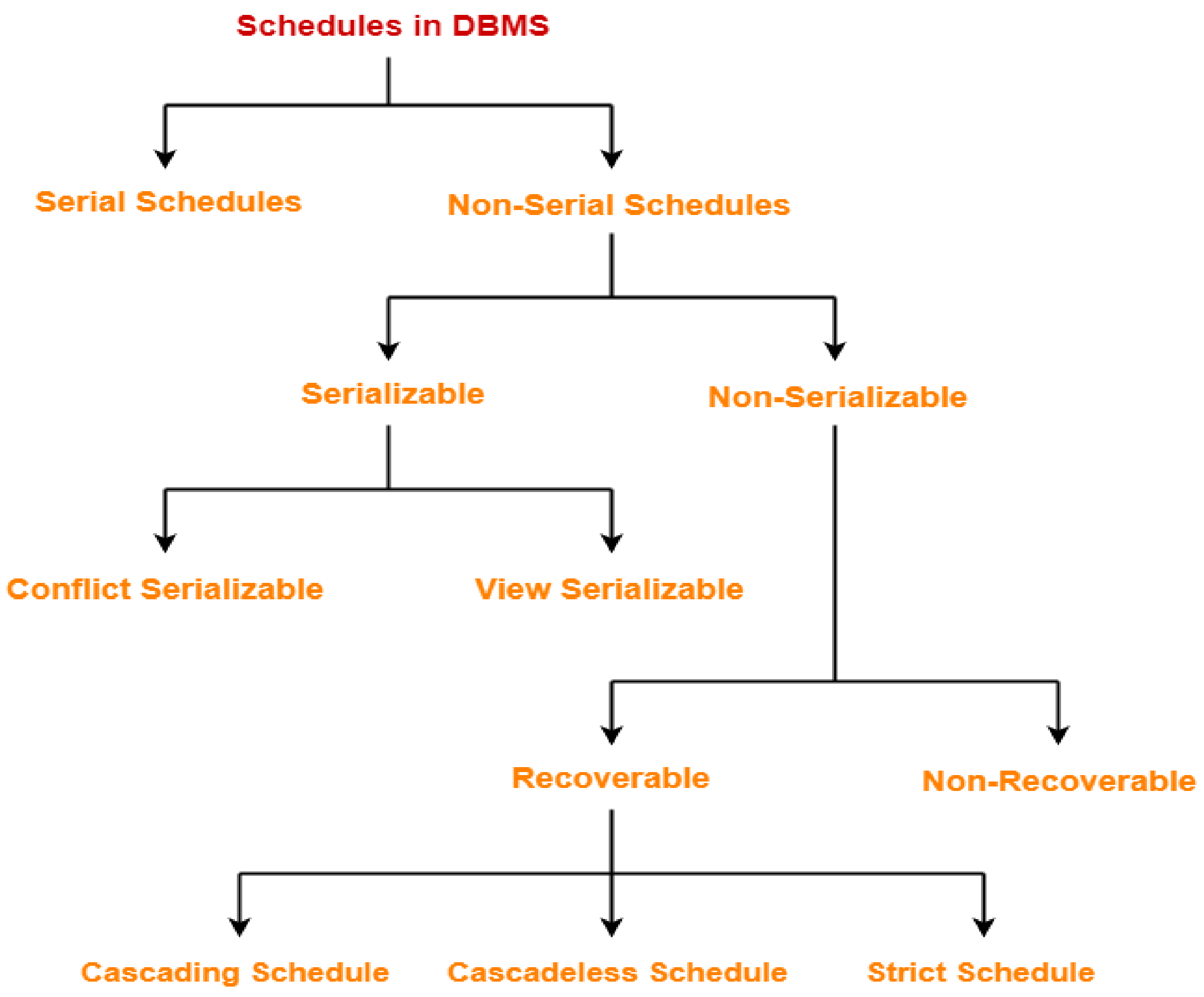
# Types of Schedule

- A schedule is the order in which the operations of multiple transactions appear for execution.
- Non-serial schedules may be serializable or non-serializable.



## Non-Serializable Schedules-

- A non-serial schedule which is not serializable is called as a non-serializable schedule.

- A non-serializable schedule is not guaranteed to produce the same effect as produced by some serial schedule on any consistent database.

**Characteristics-**

Non-serializable schedules-

- may or may not be consistent
- may or may not be recoverable

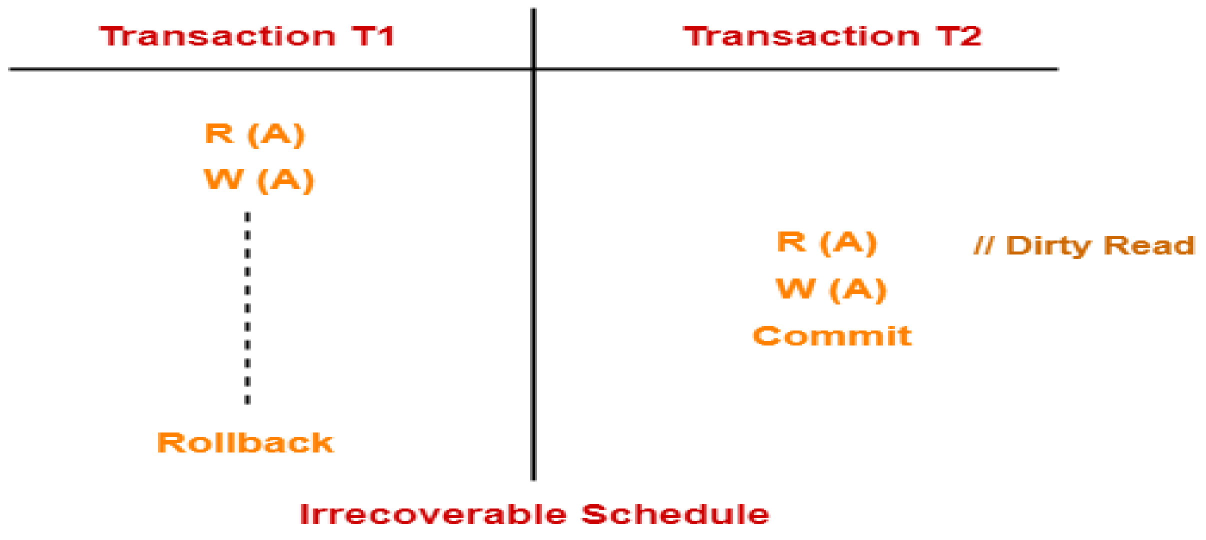
## Irrecoverable Schedules

If in a schedule,

- A transaction performs a dirty read operation from an uncommitted transaction
- And commits before the transaction from which it has read the value then such a schedule is known as an **Irrecoverable Schedule**.

**Example-**

Consider the following schedule-



Here,

- T2 performs a dirty read operation.
- T2 commits before T1.
- T1 fails later and roll backs.

- The value that T2 read now stands to be incorrect.
- T2 cannot recover since it has already committed.

## Recoverable Schedules

If in a schedule,

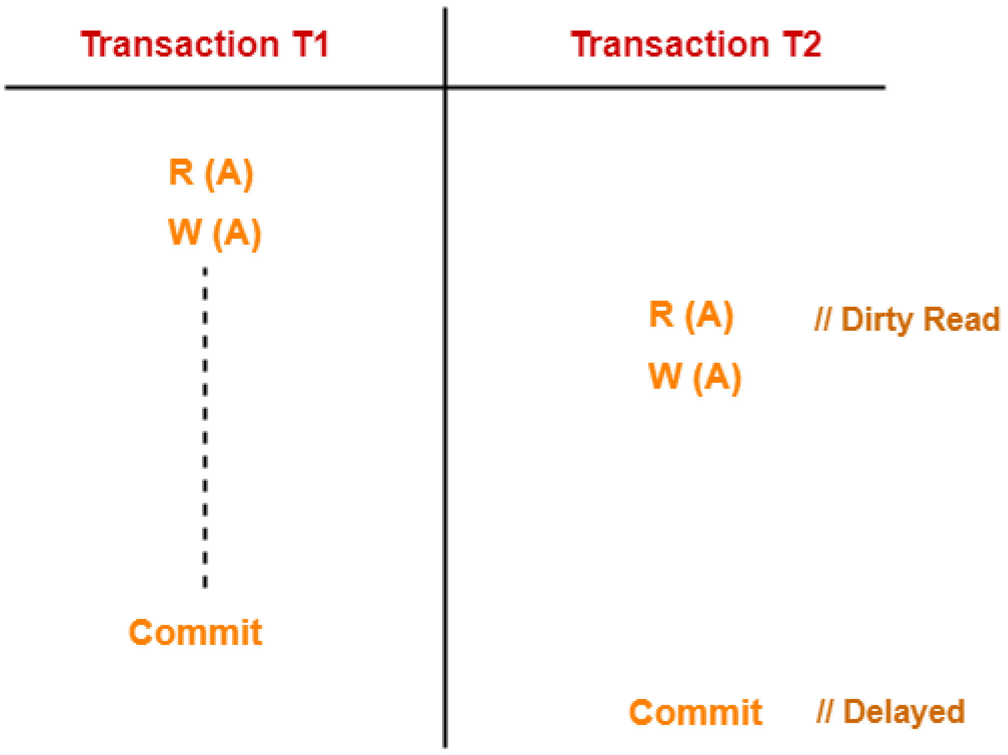
- A transaction performs a dirty read operation from an uncommitted transaction
- And its commit operation is delayed till the uncommitted transaction either commits or roll backs then such a schedule is known as a **Recoverable Schedule**.

Here,

- The commit operation of the transaction that performs the dirty read is delayed.
- This ensures that it still has a chance to recover if the uncommitted transaction fails later.

**Example-**

Consider the following schedule-



Recoverable Schedule

Here,

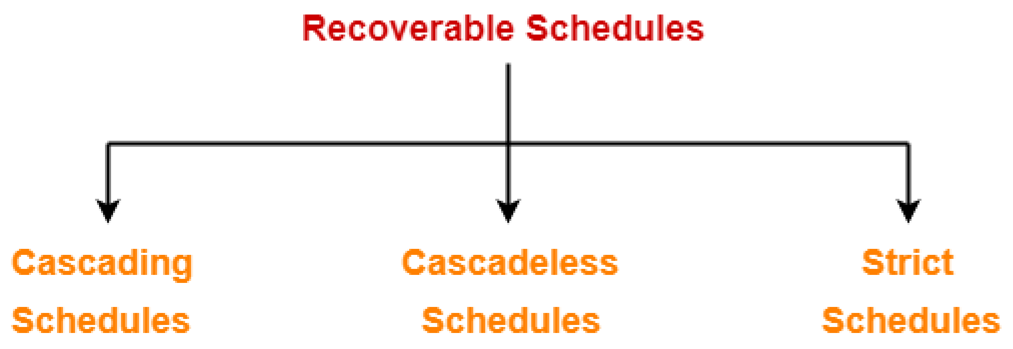
- T2 performs a dirty read operation.
- The commit operation of T2 is delayed till T1 commits or roll backs.
- T1 commits later.
- T2 is now allowed to commit.
- In case, T1 would have failed, T2 has a chance to recover by rolling back.

## Recoverable Schedules-

If in a schedule,

- A transaction performs a dirty read operation from an uncommitted transaction
- And its commit operation is delayed till the uncommitted transaction either commits or roll backs
- then such a schedule is called as a Recoverable Schedule.

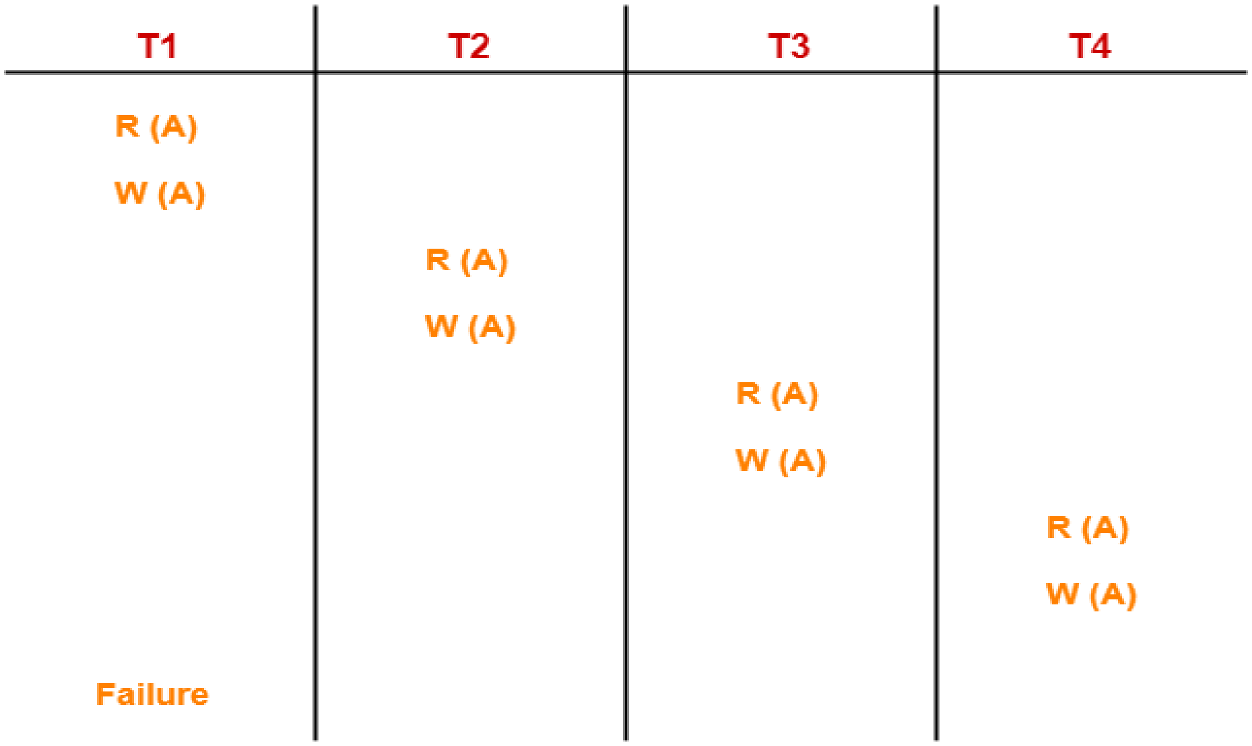
### Types of Recoverable Schedules-



## Cascading Schedule-

- If in a schedule, failure of one transaction causes several other dependent transactions to rollback or abort, then such a schedule is called as a Cascading Schedule or Cascading Rollback or Cascading Abort.
- It simply leads to the wastage of CPU time.

**Consider the Example given below -**



| T1 | T2 | T3 | T4 |
|---|---|---|---|
| R (A) | | | |
| W (A) | | | |
| | R (A) | | |
| | W (A) | | |
| | | R (A) | |
| | | W (A) | |
| | | | R (A) |
| | | | W (A) |
| Failure | | | |

Cascading Recoverable Schedule

**Here,**

- Transaction T2 depends on transaction T1.
- Transaction T3 depends on transaction T2.
- Transaction T4 depends on transaction T3.

**In this schedule,**

- The failure of transaction T1 causes the transaction T2 to rollback.
- The rollback of transaction T2 causes the transaction T3 to rollback.
- The rollback of transaction T3 causes the transaction T4 to rollback.
- Such a rollback is called as a Cascading Rollback.

**NOTE**

if the transactions T2, T3 and T4 would have committed before the failure of transaction T1, then the schedule would have been irrecoverable.

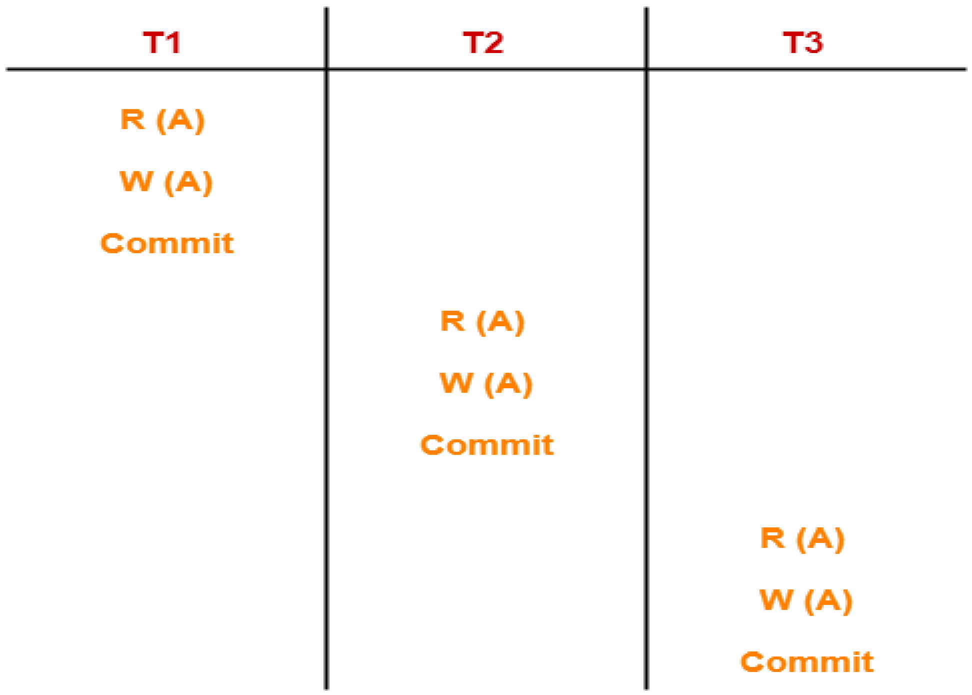
## Cascadeless Schedule

If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a Cascadeless Schedule.

In other words,

- Cascadeless schedule allows only committed read operations.
- Therefore, it avoids cascading roll back and thus saves CPU time.

**Example:**



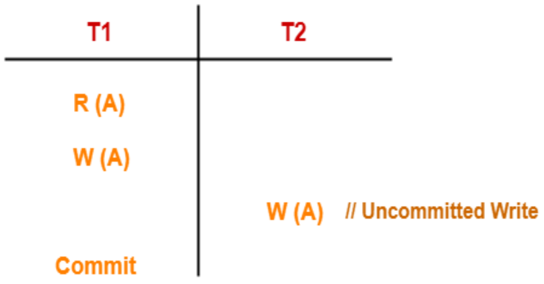
| T1 | T2 | T3 |
|---|---|---|
| R (A) | | |
| W (A) | | |
| Commit | | |
| | R (A) | |
| | W (A) | |
| | Commit | |
| | | R (A) |
| | | W (A) |
| | | Commit |

Cascadeless Schedule

**NOTE-**

- Cascadeless schedule allows only committed read operations.
- However, it allows uncommitted write operations.

**Example:**



| T1 | T2 |
|---|---|
| R (A) | |
| W (A) | |
| | W (A) // Uncommitted Write |
| Commit | |

Cascadeless Schedule

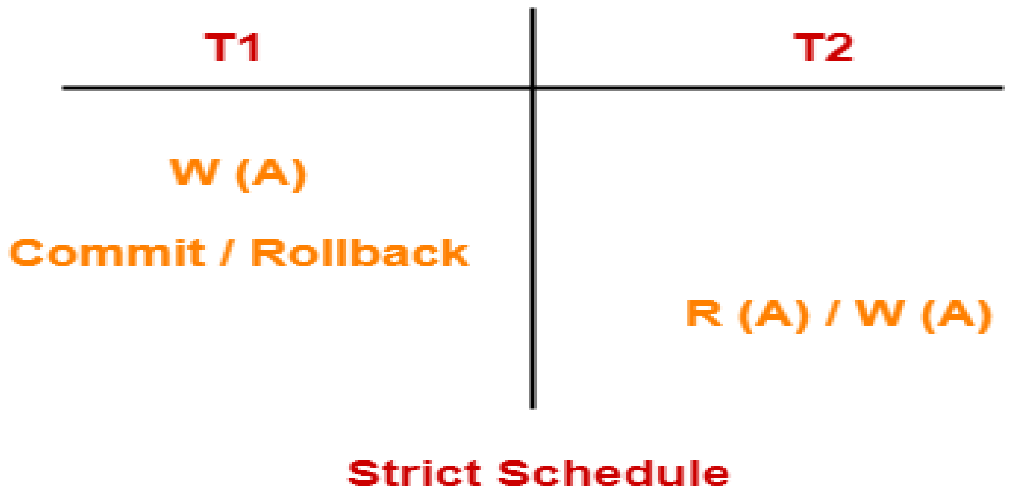
## Strict Schedule

If in a schedule, a transaction is neither allowed to read nor write a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a Strict Schedule.

In other words,

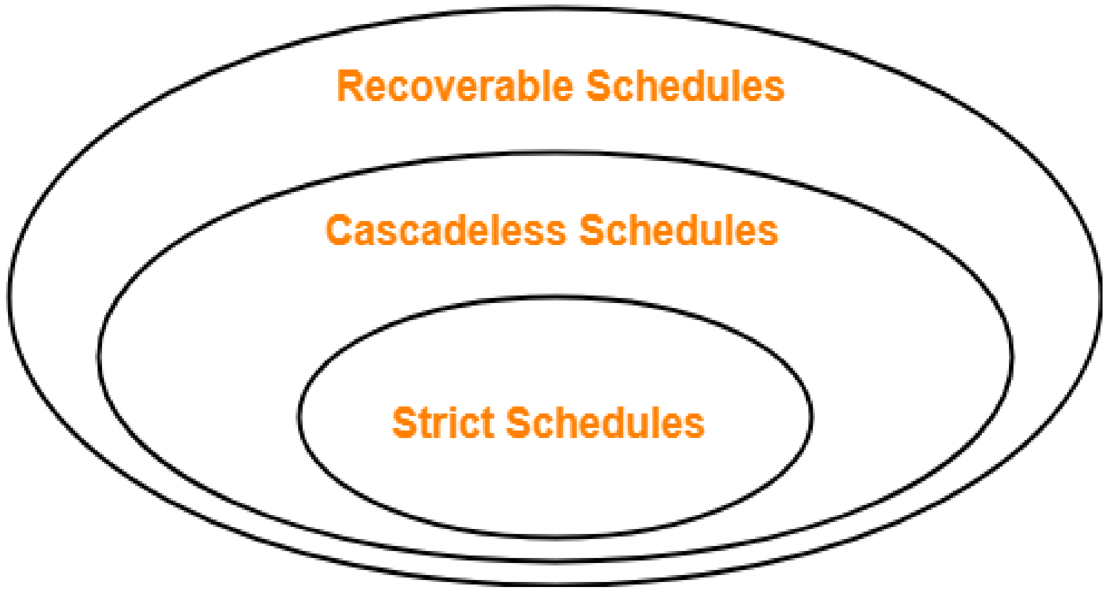
- Strict schedule allows only committed read and write operations.
- Clearly, strict schedule implements more restrictions than cascadeless schedule.

**Example**



**Remember,**

- Strict schedules are more strict than cascadeless schedules.
- All strict schedules are cascadeless schedules.
- All cascadeless schedules are not strict schedules.


Recoverable Schedules
Cascadeless Schedules
Strict Schedules