CSE 301 – DATABASE

Lecture 15

Chapter 6: Transaction Management

Transaction

- "Transaction is a set of operations which are all logically related."

OR

- "Transaction is a set of instruction which are all logically related."

OR

- "Transaction is a single logical unit of work formed by a set of operations."
- The main operations in a transaction are-
  - ✓ Read Operation
  - ✓ Write Operation

Transaction

- Read Operation:
  - Read operation reads the data from the database and then stores it in the buffer in main memory.
  - For example- Read(A) instruction will read the value of A from the database and will store it in the buffer in main memory.

.

Transaction

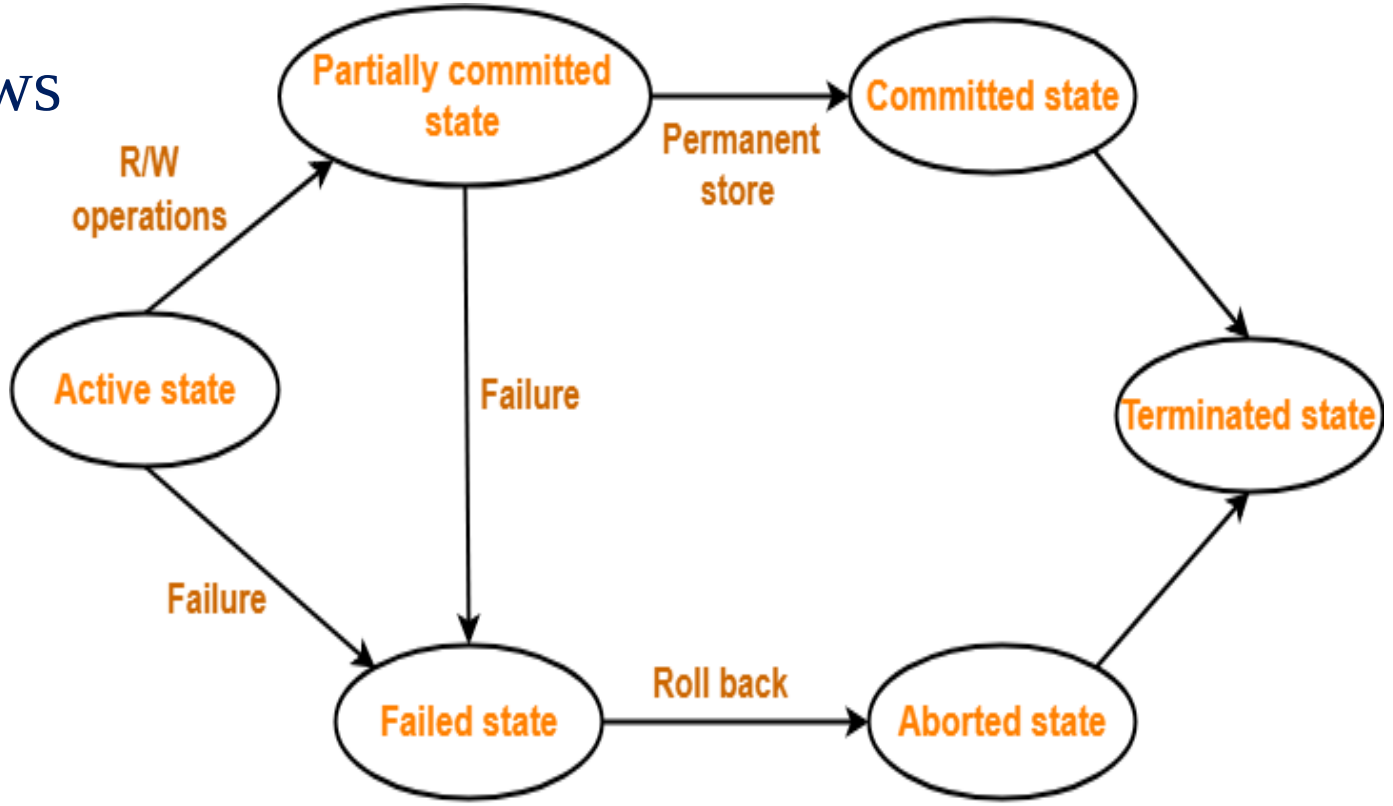
- Write Operation
  - Write operation writes the updated data value back to the database from the buffer.
  - For example- Write(A) will write the updated value of A from the buffer to the database.

Transaction States

- A transaction goes through many different states throughout its life cycle.

  These states are called as transaction states.

- Transaction states are as follows
  - ✓ Active state
  - ✓ Partially committed state
  - ✓ Committed state
  - ✓ Failed state
  - ✓ Aborted state
  - ✓ Terminated state



Transaction States in DBMS

Transaction States

- Active State:
  - ✓ This is the first state in the life cycle of a transaction.
  - ✓ A transaction is called in an active state as long as its instructions are getting executed.
  - ✓ All the changes made by the transaction now are stored in the buffer in main memory.

Transaction States

- Partially Committed State:
  - ✓ After the last instruction of transaction has executed, it enters into a partially committed state.
  - ✓ After entering this state, the transaction is considered to be partially committed.
  - ✓ It is not considered fully committed because all the changes made by the transaction are still stored in the buffer in main memory.

Transaction States

- Committed State:
  - ✓ After all the changes made by the transaction have been successfully stored into the database, it enters into a committed state.
  - ✓ Now, the transaction is considered to be fully committed.

NOTE:

- ✓ After a transaction has entered the committed state, it is not possible to roll back the transaction.
- ✓ In other words, it is not possible to undo the changes that has been made by the transaction.
- ✓ This is because the system is updated into a new consistent state.
- ✓ The only way to undo the changes is by carrying out another transaction called as compensating transaction that performs the reverse operations.

Transaction States

- Failed State:
  - ✓ When a transaction is getting executed in the active state or partially committed state and some failure occurs due to which it becomes impossible to continue the execution, it enters into a failed state.
- Aborted State:
  - ✓ After the transaction has failed and entered into a failed state, all the changes made by it have to be undone.
  - ✓ To undo the changes made by the transaction, it becomes necessary to roll back the transaction.
  - ✓ After the transaction has rolled back completely, it enters into an aborted state.

Transaction States

- Terminated State:
  - ✓ This is the last state in the life cycle of a transaction.
  - ✓ After entering the committed state or aborted state, the transaction finally enters into a terminated state where its life cycle finally comes to an end.

ACID Properties in Transaction

- It is important to ensure that the database remains consistent before and after the transaction.
- To ensure the consistency of database, certain properties are followed by all the transactions occurring in the system.
- These properties are called as ACID Properties of a transaction.

A = Atomicity

C = Consistency

I = Isolation

D = Durability

ACID Properties

- Atomicity:
  - ✓ This property ensures that either the transaction occurs completely, or it does not occur at all.
  - ✓ In other words, it ensures that no transaction occurs partially.
  - ✓ That is why, it is also referred to as “All or nothing rule“.
  - ✓ It is the responsibility of Transaction Control Manager to ensure atomicity of the transactions.

ACID Properties

- Consistency:
  - ✓ This property ensures that integrity constraints are maintained.
  - ✓ In other words, it ensures that the database remains consistent before and after the transaction.
  - ✓ It is the responsibility of DBMS and application programmer to ensure consistency of the database.

ACID Properties

- Isolation:
  - ✓ This property ensures that multiple transactions can occur simultaneously without causing any inconsistency.
  - ✓ During execution, each transaction feels as if it is getting executed alone in the system.
  - ✓ A transaction does not realize that there are other transactions as well getting executed parallelly.
  - ✓ Changes made by a transaction becomes visible to other transactions only after they are written in the memory.

ACID Properties

- Isolation:
  - ✓ The resultant state of the system after executing all the transactions is same as the state that would be achieved if the transactions were executed serially one after the other.
  - ✓ It is the responsibility of concurrency control manager to ensure isolation for all the transactions.

ACID Properties

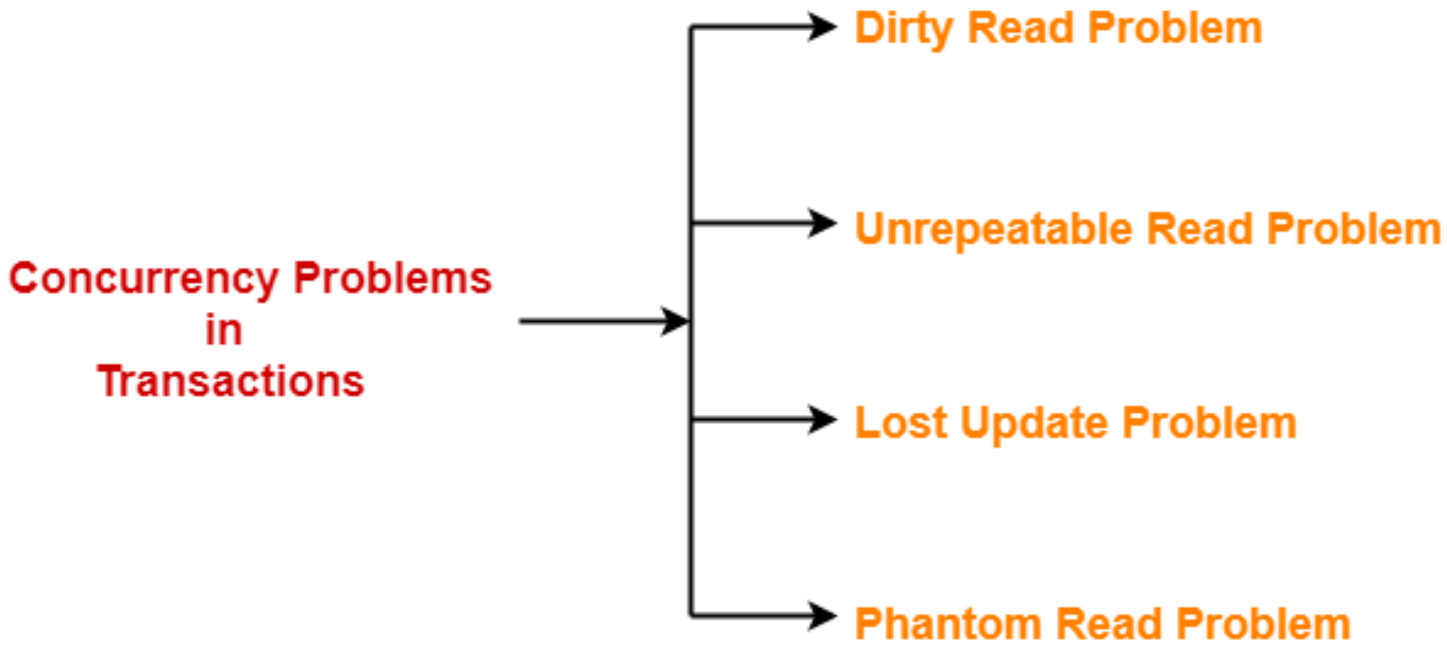
- Durability:
  - ✓ This property ensures that all the changes made by a transaction after its successful execution are written successfully to the disk.
  - ✓ It also ensures that these changes exist permanently and are never lost even if there occurs a failure of any kind.
  - ✓ It is the responsibility of recovery manager to ensure durability in the database.

Advantages of Concurrency in Transaction

- Decrease Waiting Time:
- Decrease Response Time
- Increase Resource Utilization
- Increase Efficiency:

Concurrency Problems in Transaction

- When multiple transactions execute concurrently in an uncontrolled or unrestricted manner, then it might lead to several problems.
- Such problems are called as concurrency problems.



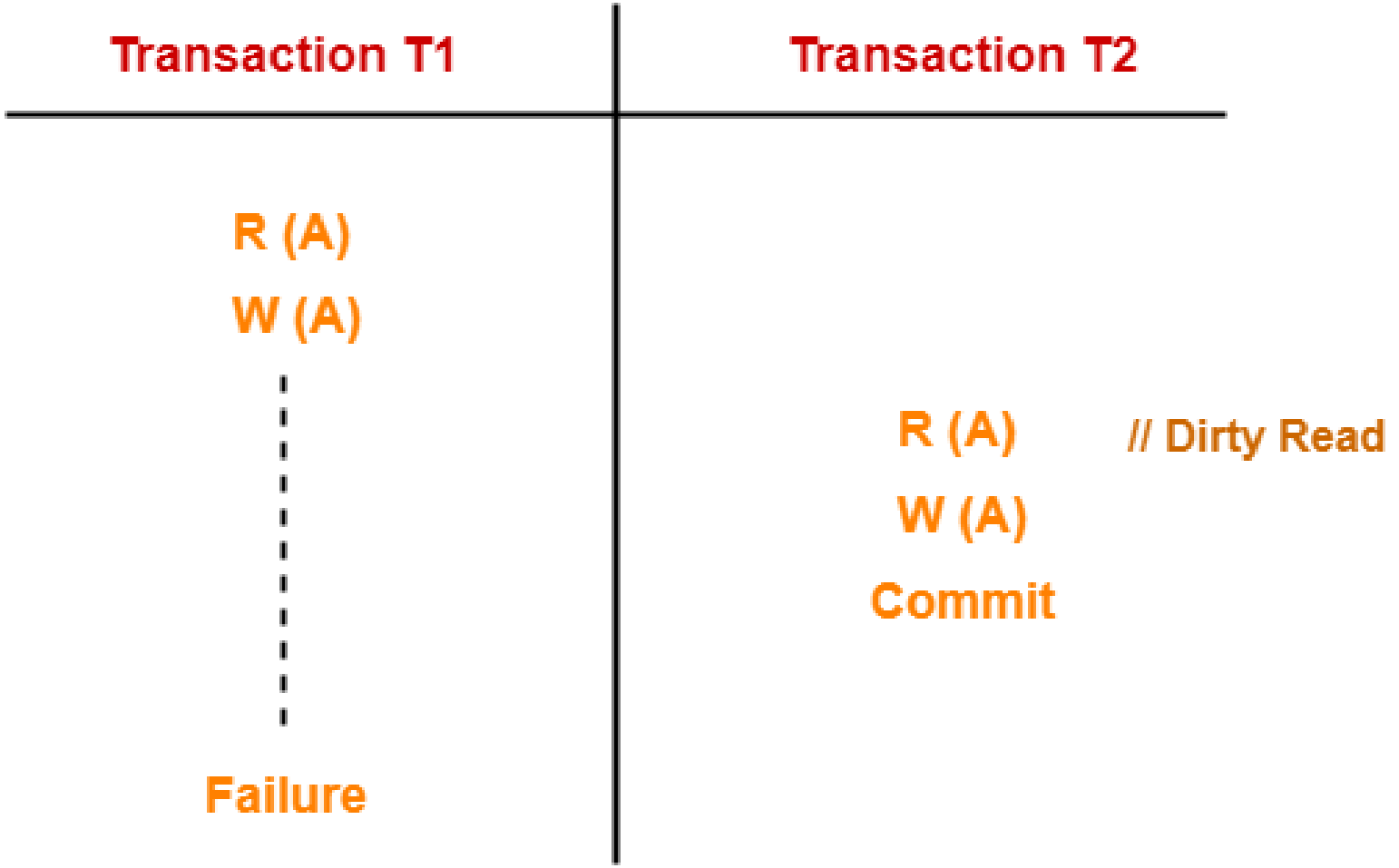
Dirty Read Problem

- Reading the data written by an uncommitted transaction is called as dirty read.
- This read is called as dirty read because-
  - ✓ There is always a chance that the uncommitted transaction might roll back later.
  - ✓ Thus, uncommitted transaction might make other transactions read a value that does not even exist.
  - ✓ This leads to inconsistency of the database.

NOTE:

- ✓ Dirty read does not lead to inconsistency always.
- ✓ It becomes problematic only when the uncommitted transaction fails and roll backs later due to some reason.

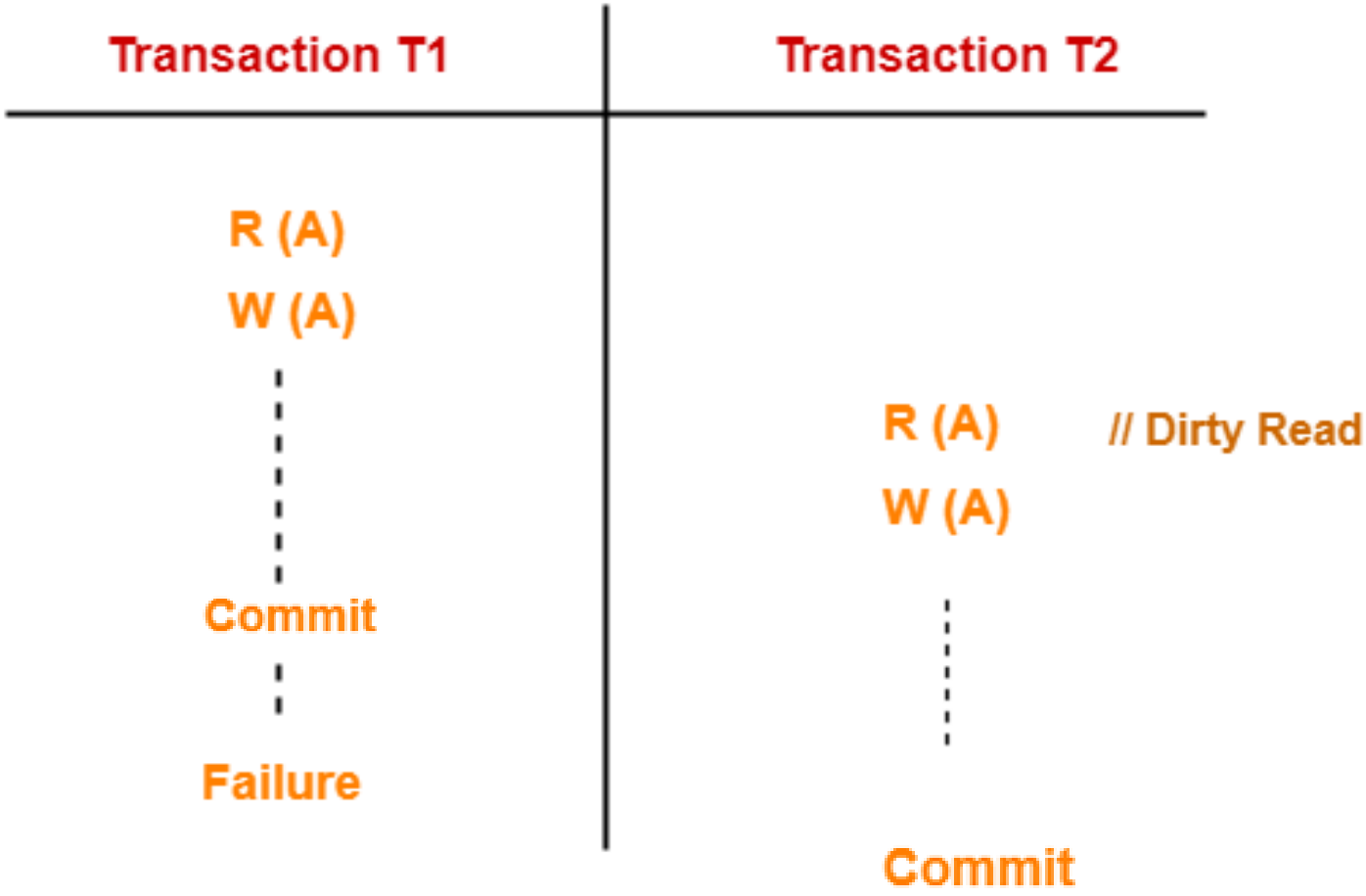
Example of Dirty Read Problem



Example of Dirty Read Problem

- Here, T1 reads the value of A. T1 updates the value of A in the buffer. T2 reads the value of A from the buffer. T2 writes the updated the value of A. T2 commits. T1 fails in later stages and rolls back.
- In this example, T2 reads the dirty value of A written by the uncommitted transaction T1. T1 fails in later stages and roll backs. Thus, the value that T2 read now stands to be incorrect. Therefore, database becomes inconsistent.

How to Avoid Dirty Read Problem



Unrepeatable Read Problem

- This problem occurs when a transaction gets to read unrepeated i.e., different values of the same variable in its different read operations even when it has not updated its value.

Transaction T1	Transaction T2
R (X)	
	R (X)
W (X)	
	R (X) // Unrepeated Read

Example of Unrepeatable Read Problem

- Here, T1 reads the value of X (= 10 say). T2 reads the value of X (= 10). T1 updates the value of X (from 10 to 15 say) in the buffer. T2 again reads the value of X (but = 15).
- In this example, T2 gets to read a different value of X in its second reading. T2 wonders how the value of X got changed because according to it, it is running in isolation.

Phantom Read Problem

- This problem occurs when a transaction reads some variable from the buffer and when it reads the same variable later, it finds that the variable does not exist.

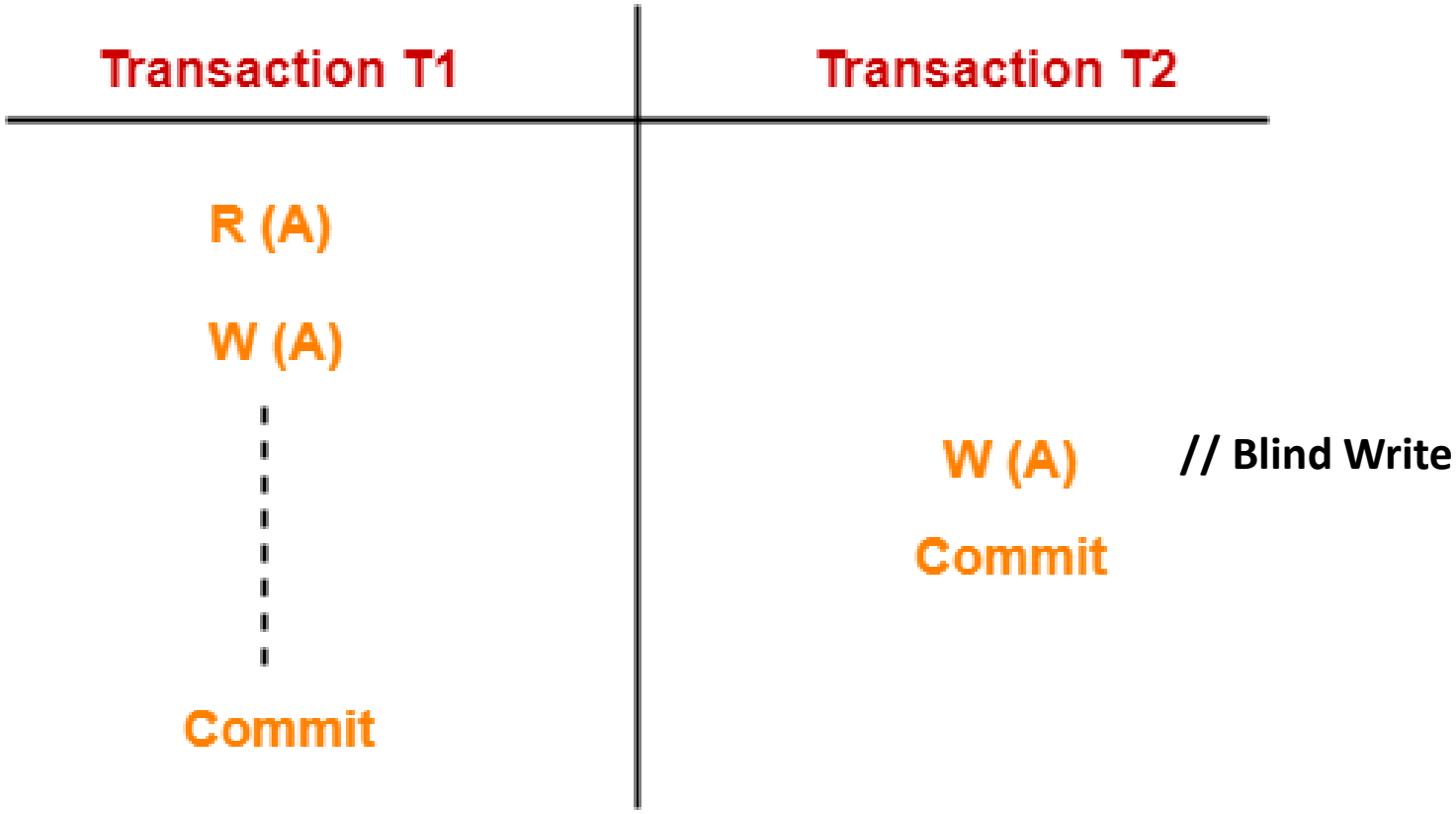
Transaction T1	Transaction T2
R (X)	
	R (X)
Delete (X)	
	Read (X)

Example of Phantom Read Problem

- Here, T1 reads X. T2 reads X. T1 deletes X. T2 tries reading X but does not find it.
- In this example, T2 finds that there does not exist any variable X when it tries reading X again. T2 wonders who deleted the variable X because according to it, it is running in isolation.

Lost Update Problem

- This problem occurs when multiple transactions execute concurrently and updates from one or more transactions get lost.



Example of Lost Update Problem

- Here, T1 reads the value of A (= 10 say). T2 updates the value to A (= 15 say) in the buffer. T2 does blind write A = 25 (write without read) in the buffer. T2 commits. When T1 commits, it writes A = 25 in the database.
- In this example, T1 writes the over written value of X in the database. Thus, update from T1 gets lost.

NOTE:

- ✓ This problem occurs whenever there is a write-write conflict.
- ✓ In write-write conflict, there are two writes one by each transaction on the same data item without any read in the middle.

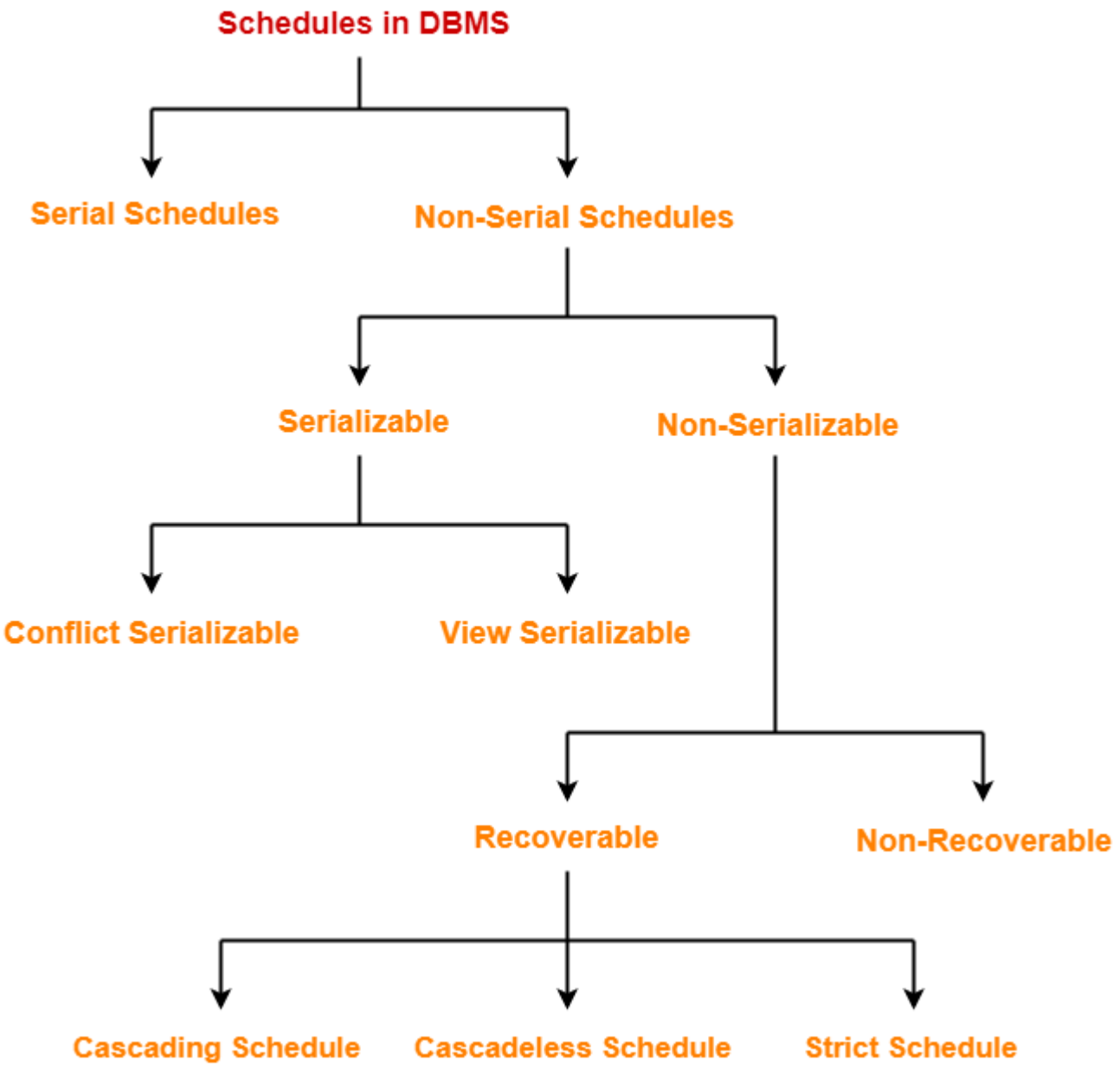
Avoiding Concurrency Problems

- To ensure consistency of the database, it is very important to prevent the occurrence of above problems.
- Concurrency Control Protocols help to prevent the occurrence of above problems and maintain the consistency of the database.

Schedules in DBMS

- The order in which the operations of multiple transactions appear for execution is called as a schedule.

Types of Schedules



Serial Schedules

- All the transactions execute serially one after the other.
- When one transaction executes, no other transaction is allowed to execute.
- Characteristics: Serial schedules are always-
  - ✓ Consistent
  - ✓ Recoverable
  - ✓ Cascadeless
  - ✓ Strict

Example of Serial Schedules

- In this schedule,
  - ✓ There are two transactions T1 and T2 executing serially one after the other.
  - ✓ Transaction T1 executes first.
  - ✓ After T1 completes its execution, transaction T2 executes.
  - ✓ So, this schedule is an example of a Serial Schedule.

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

Example of Serial Schedules

- In this schedule,
  - ✓ There are two transactions T1 and T2 executing serially one after the other.
  - ✓ Transaction T2 executes first.
  - ✓ After T2 completes its execution, transaction T1 executes.
  - ✓ So, this schedule is an example of a Serial Schedule.

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

Example of Non-Serial Schedules

- Multiple transactions execute concurrently.
- Operations of all the transactions are inter leaved or mixed with each other.
- Characteristics: Serial schedules are not always
  - ✓ Consistent
  - ✓ Recoverable
  - ✓ Cascadeless
  - ✓ Strict

Example of Non-Serial Schedules

- In this schedule,
  - ✓ There are two transactions T1 and T2 executing concurrently.
  - ✓ The operations of T1 and T2 are interleaved.
  - ✓ So, this schedule is an example of a Non-Serial Schedule.

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

Example of Non-Serial Schedules

- In this schedule,
  - ✓ There are two transactions T1 and T2 executing concurrently.
  - ✓ The operations of T1 and T2 are interleaved.
  - ✓ So, this schedule is an example of a Non-Serial Schedule.

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

Finding Number Of Schedules

- Consider there are n number of transactions T1, T2, T3 …. , Tn with N1, N2, N3 …. , Nn number of operations respectively.

- Total number of possible schedules (serial + non-serial) is given by

$$\frac{(N1 + N2 + N3 + ... + Nn)!}{N1! \times N2! \times N3! \times .... \times Nn!}$$

- Total number of serial schedules = Number of different ways of arranging n transactions = n!

- Total number of non-serial schedules = Total number of schedules – Total number of serial schedules

Practice Problem Based On Finding Number Of Schedules

- Problem: Consider there are three transactions with 2, 3, 4 operations respectively, find:
  1. How many total number of schedules are possible?
  2. How many total number of serial schedules are possible?
  3. How many total number of non-serial schedules are possible?

Practice Problem Based On Finding Number Of Schedules

- Solution:

1. $$\text{Total number of schedules} = \frac{(2+3+4)!}{2! \times 3! \times 4!}$$

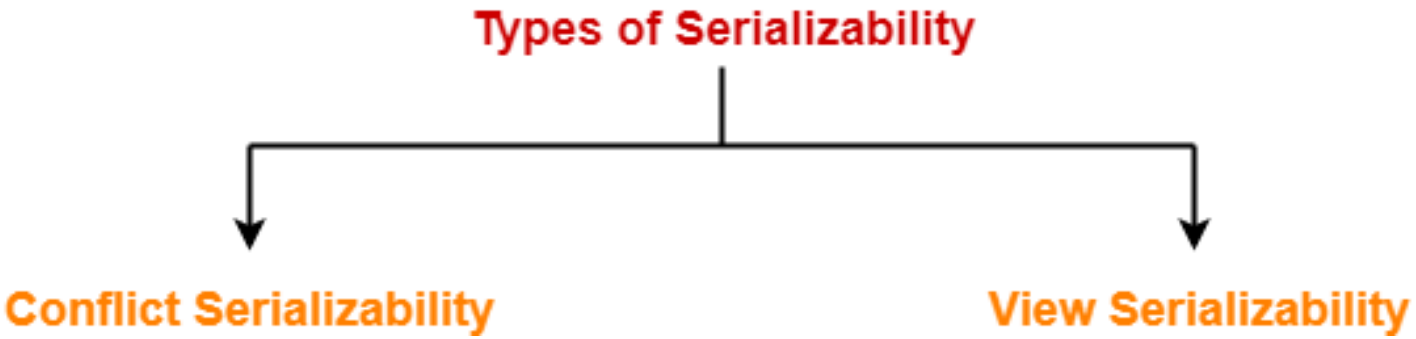
$$= 1260$$

2. Total number of serial schedules = Number of different ways of arranging 3 transactions = 3! = 6

3. Total number of non-serial schedules = Total number of schedules – Total number of serial schedules = 1260 – 6 = 1254

Serializability

- Some non-serial schedules may lead to inconsistency of the database.
- Serializability is a concept that helps to identify which non-serial schedules are correct and will maintain the consistency of the database.



Serializable Schedules

- If a given non-serial schedule of ‘n’ transactions is equivalent to some serial schedule of ‘n’ transactions, then it is called as a serializable schedule.
- Characteristics: Serializable schedules behave exactly same as serial schedules. Thus, serializable schedules are always
  - ✓ Consistent
  - ✓ Recoverable
  - ✓ Casacadeless
  - ✓ Strict

Serial Schedules Vs Serializable Schedules

Serial Schedules	Serializable Schedules
No concurrency is allowed. Thus, all the transactions necessarily execute serially one after the other.	Concurrency is allowed. Thus, multiple transactions can execute concurrently.
Serial schedules lead to less resource utilization and CPU throughput.	Serializable schedules improve both resource utilization and CPU throughput.
Serial Schedules are less efficient as compared to serializable schedules. (due to above reason)	Serializable Schedules are always better than serial schedules. (due to above reason)

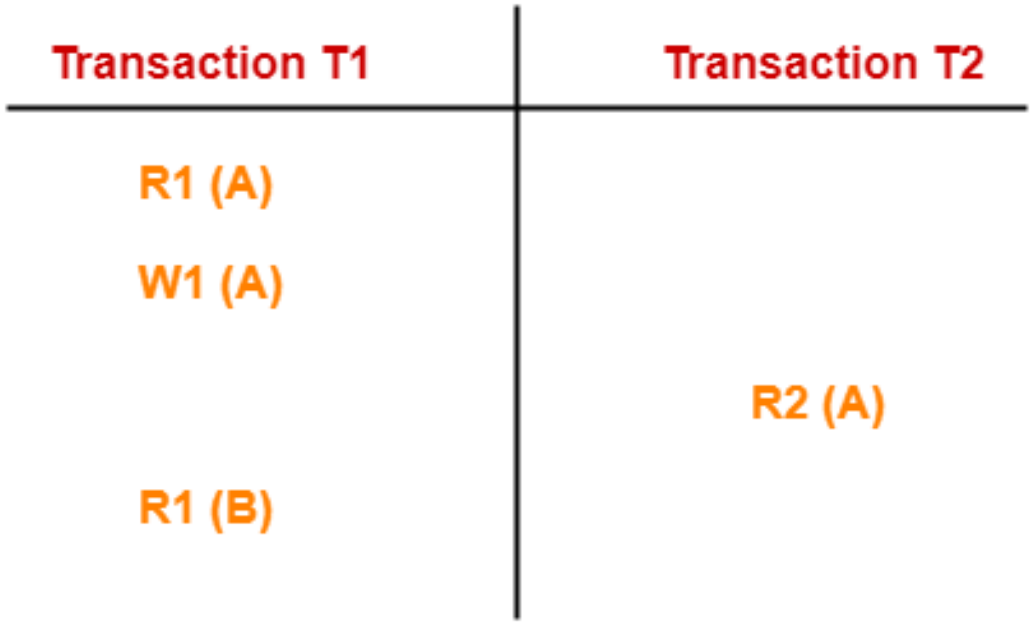
Conflict Serializability

- If a given non-serial schedule can be converted into a serial schedule by swapping its non-conflicting operations, then it is called as a conflict serializable schedule.

- Conflicting Operations:
  - ✓ Two operations are called as conflicting operations if all the following conditions hold true for them-
  - ✓ Both the operations belong to different transactions
  - ✓ Both the operations are on the same data item
  - ✓ At least one of the two operations is a write operation

Example of Conflict Serializability

- In this schedule,
  - ✓ W1 (A) and R2 (A) are called as conflicting operations.
  - ✓ This is because all the above conditions hold true for them.



Checking Whether a Schedule is Conflict Serializable Or Not

- Follow the following steps to check whether a given non-serial schedule is conflict serializable or not.
- Step-01: Find and list all the conflicting operations.
- Step-02: Start creating a precedence graph by drawing one node for each transaction.
- Step-03:
  - Draw an edge for each conflict pair such that if Xi (V) and Yj (V) forms a conflict pair then draw an edge from Ti to Tj.
  - This ensures that Ti gets executed before Tj.

Checking Whether a Schedule is Conflict Serializable Or Not

- Step-04:
  - Check if there is any cycle formed in the graph.
  - If there is no cycle found, then the schedule is conflict serializable otherwise not.

NOTE:

- ✓ By performing the Topological Sort of the Directed Acyclic Graph so obtained, the corresponding serial schedule(s) can be found.
- ✓ Such schedules can be more than 1.

Practice Problems Based On Conflict Serializability

- Problem 1: Check whether the given schedule S is conflict serializable or not?

  S : R1(A) , R2(A) , R1(B) , R2(B) , R3(B) , W1(A) , W2(B)

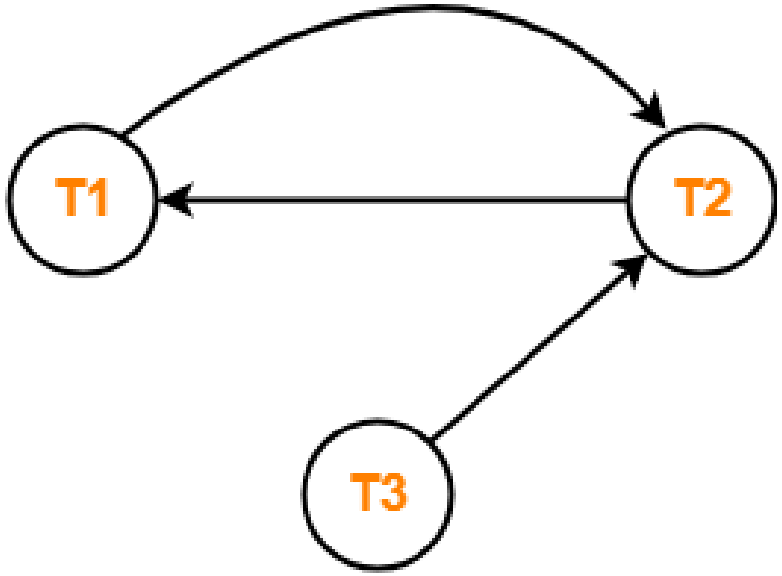
- Solution:

- Step-01: List all the conflicting operations and determine the dependency between the transactions

  - R2(A) , W1(A) (T2 → T1)
  - R1(B) , W2(B) (T1 → T2)
  - R3(B) , W2(B) (T3 → T2)

Practice Problems Based On Conflict Serializability

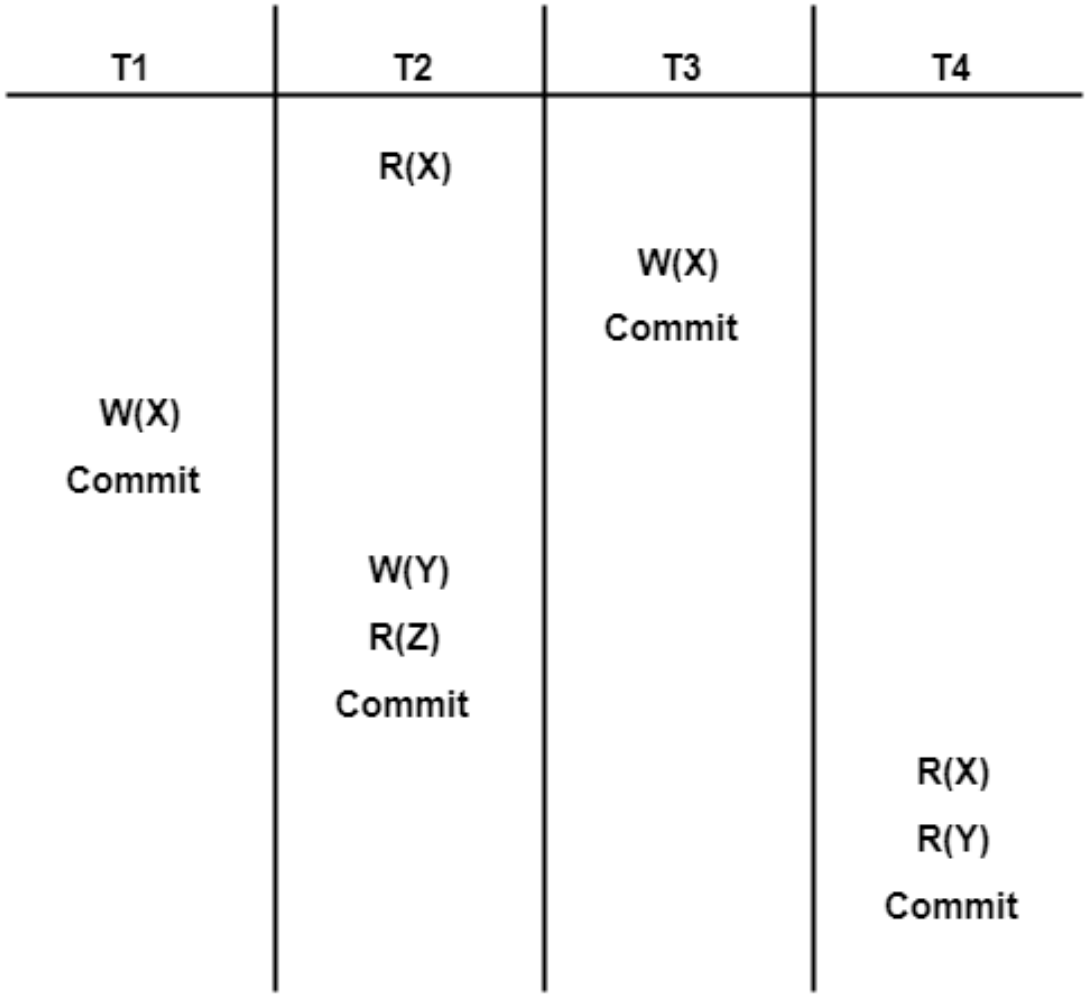
- ✓ R2(A) , W1(A) (T2 → T1)
- ✓ R1(B) , W2(B) (T1 → T2)
- ✓ R3(B) , W2(B) (T3 → T2)



- ❑ Step-02: Draw the precedence graph
- ❑ Clearly, there exists a cycle in the precedence graph.
- ❑ Therefore, the given schedule S is not conflict serializable.

Practice Problems Based On Conflict Serializability

- Problem 2: Check whether the given schedule S is conflict serializable and recoverable or not?

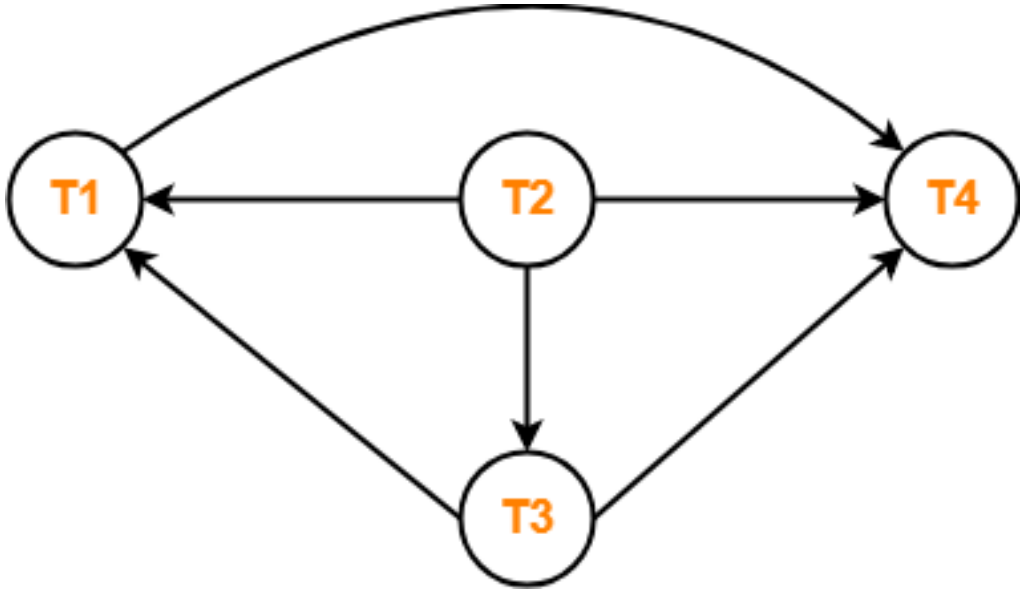


Practice Problems Based On Conflict Serializability

- Solution:
- Step-01: List all the conflicting operations and determine the dependency between the transactions
  - R2(X) , W3(X) (T2 → T3)
  - R2(X) , W1(X) (T2 → T1)
  - W3(X) , W1(X) (T3 → T1)
  - W3(X) , R4(X) (T3 → T4)
  - W1(X) , R4(X) (T1 → T4)
  - W2(Y) , R4(Y) (T2 → T4)

Practice Problems Based On Conflict Serializability

- Step-02: Draw the precedence graph
  - Clearly, there exists no cycle in the precedence graph.
  - Therefore, the given schedule S is conflict serializable.

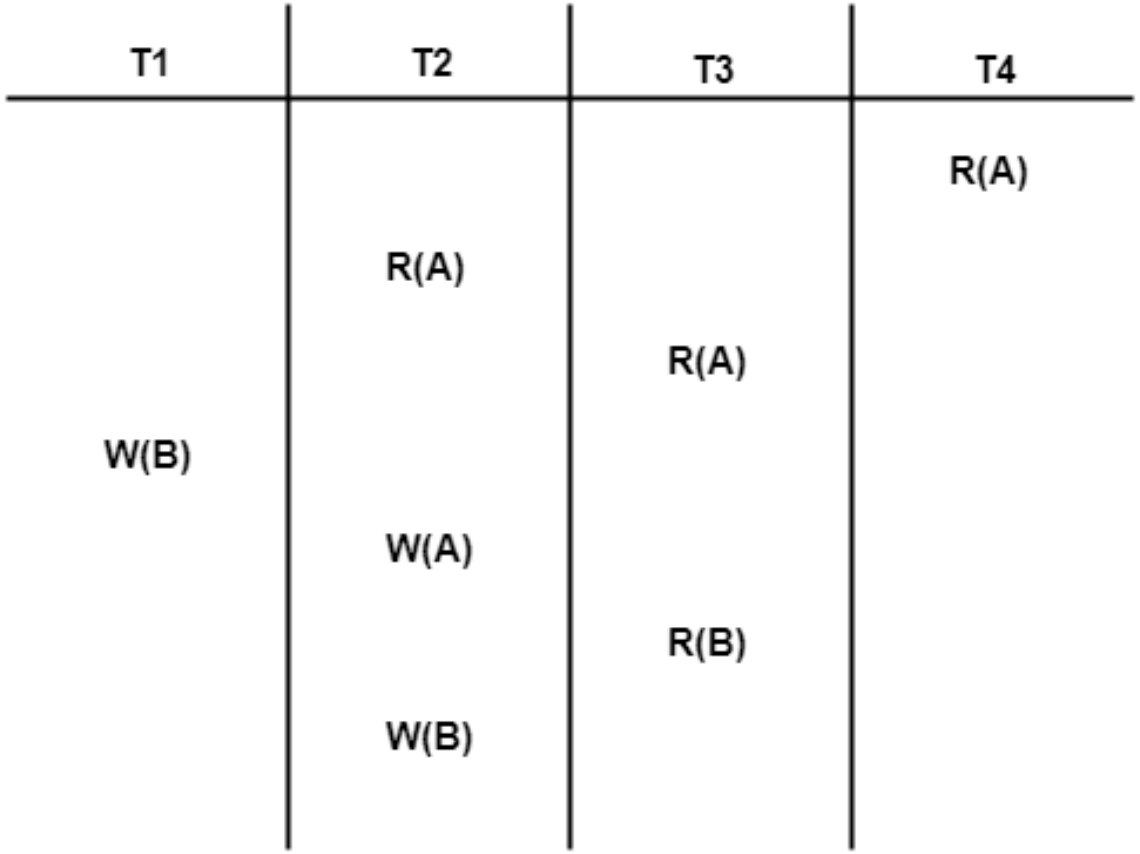


Practice Problems Based On Conflict Serializability

- Checking Whether S is Recoverable Or Not?
  - Conflict serializable schedules are always recoverable.
  - Therefore, the given schedule S is recoverable.
- Alternatively,
  - There exists no dirty read operation.
  - This is because all the transactions which update the values commits immediately.
  - Therefore, the given schedule S is recoverable.
  - Also, S is a Cascadeless Schedule.

Practice Problems Based On Conflict Serializability

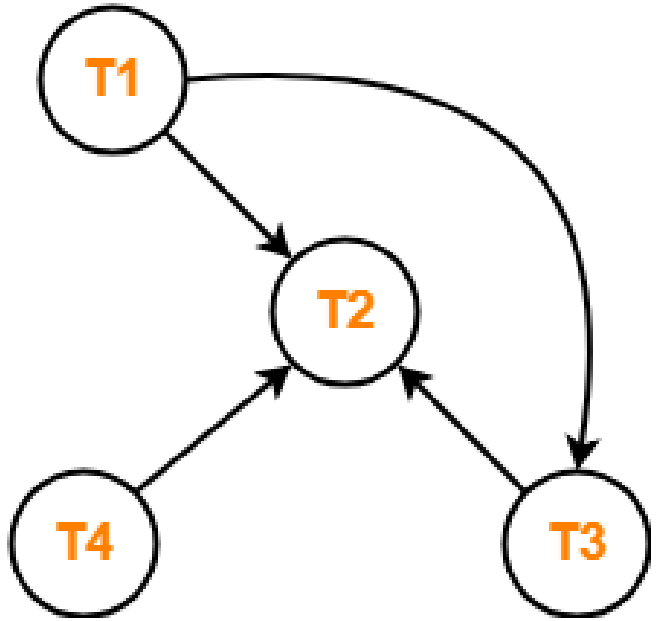
- Problem-03: Check whether the given schedule S is conflict serializable or not. If yes, then determine all the possible serialized schedules



Practice Problems Based On Conflict Serializability

- Solution:

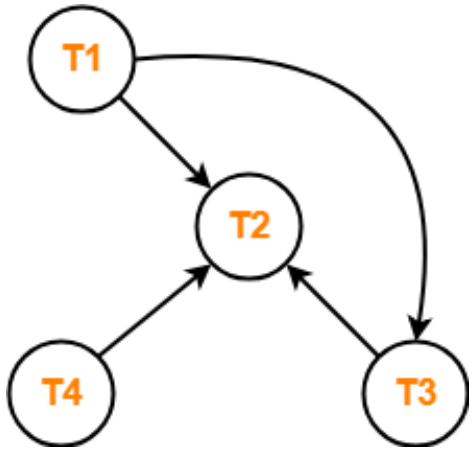
- Step-01: List all the conflicting operations and determine the dependency between the transactions.
  - R4(A) , W2(A) (T4 → T2)
  - R3(A) , W2(A) (T3 → T2)
  - W1(B) , R3(B) (T1 → T3)
  - W1(B) , W2(B) (T1 → T2)
  - R3(B) , W2(B) (T3 → T2)



- Step-02: Draw the precedence graph

Practice Problems Based On Conflict Serializability

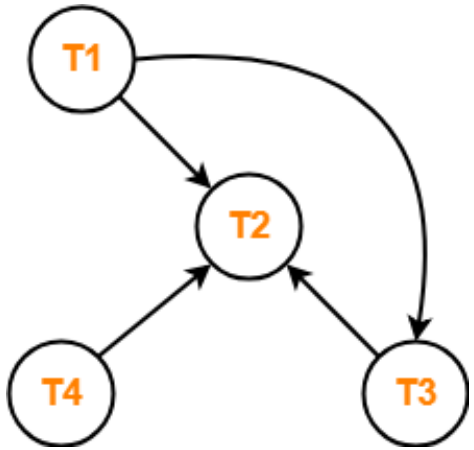
- Solution: Clearly, there exists no cycle in the precedence graph.
- Therefore, the given schedule S is conflict serializable.
- Finding the Serialized Schedules
- All the possible topological orderings of the above precedence graph will be the possible serialized schedules.
- The topological orderings can be found by performing the Topological Sort of the above precedence graph.



- So, serialized schedules are:
  - T1 → T3 → T4 → T2
  - T1 → T4 → T3 → T2
  - T4 → T1 → T3 → T2

Practice Problems Based On Conflict Serializability

- Solution: Clearly, there exists no cycle in the precedence graph.
- Therefore, the given schedule S is conflict serializable.
- Finding the Serialized Schedules
- All the possible topological orderings of the above precedence graph will be the possible serialized schedules.
- The topological orderings can be found by performing the Topological Sort of the above precedence graph.



- So, serialized schedules are:
  - T1 → T3 → T4 → T2
  - T1 → T4 → T3 → T2
  - T4 → T1 → T3 → T2

Practice Problems Based On Conflict Serializability

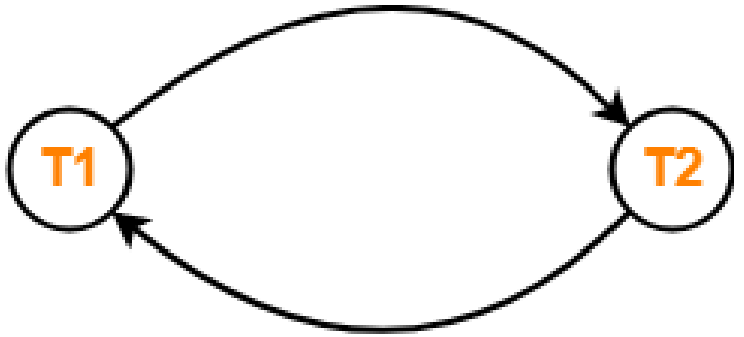
- Problem-04: Determine all the possible serialized schedules for the given schedul

T1	T2
R(A)	
A = A-10	
	R(A)
	Temp = 0.2 x A
	W(A)
	R(B)
W(A)	
R(B)	
B = B+10	
W(B)	
	B = B+Temp
	W(B)

Practice Problems Based On Conflict Serializability

- Solution: Step-01: List all the conflicting operations and determine the dependency between the transactions
  - R1(A) , W2(A) (T1 → T2)
  - R2(A) , W1(A) (T2 → T1)
  - W2(A) , W1(A) (T2 → T1)
  - R2(B) , W1(B) (T2 → T1)
  - R1(B) , W2(B) (T1 → T2)
  - W1(B) , W2(B) (T1 → T2)

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



- Step-02: Draw the precedence graph

Practice Problems Based On Conflict Serializability

- Clearly, there exists a cycle in the precedence graph.
- Therefore, the given schedule S is not conflict serializable.
- Thus, Number of possible serialized schedules = 0.

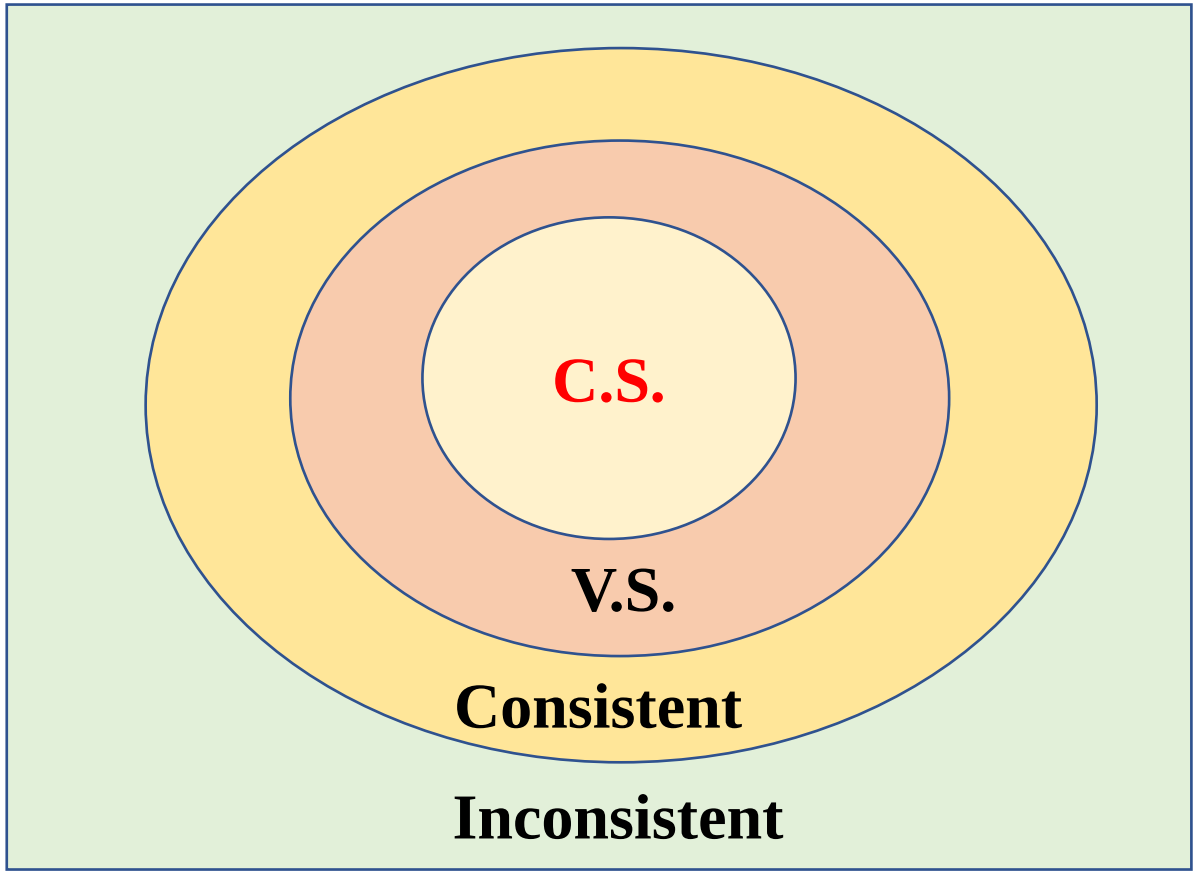
View Serializability

- If a given schedule is found to be view equivalent to some serial schedule, then it is called as a view serializable schedule.
- Consider two schedules S1 and S2 each consisting of two transactions T1 and T2. Schedules S1 and S2 are called view equivalent if the following three conditions hold true for them:
- Condition-01:
  - For each data item X, if transaction Ti reads X from the database initially in schedule S1, then in schedule S2 also, Ti must perform the initial read of X from the database.
  - Thumb Rule: “Initial readers must be same for all the data items”.

View Equivalent Schedules

- Condition-02:
  - If transaction Ti reads a data item that has been updated by the transaction Tj in schedule S1, then in schedule S2 also, transaction Ti must read the same data item that has been updated by the transaction Tj.
  - Thumb Rule: “Write-Read sequence must be same.”.
- Condition-03:
  - For each data item X, if X has been updated at last by transaction Ti in schedule S1, then in schedule S2 also, X must be updated at last by transaction Ti.
  - Thumb Rule: “Final write must be same for all the data items”.

View Serializability



Checking Whether a Schedule is View Serializable Or Not

- Method-01:
  - Check whether the given schedule is conflict serializable or not.
    - ✓ If the given schedule is conflict serializable, then it is surely view serializable. Stop and report your answer.
    - ✓ If the given schedule is not conflict serializable, then it may or may not be view serializable. Go and check using other methods.
  - Thumb Rules:
    - ✓ All conflict serializable schedules are view serializable.
    - ✓ All view serializable schedules may or may not be conflict serializable.

Checking Whether a Schedule is View Serializable Or Not

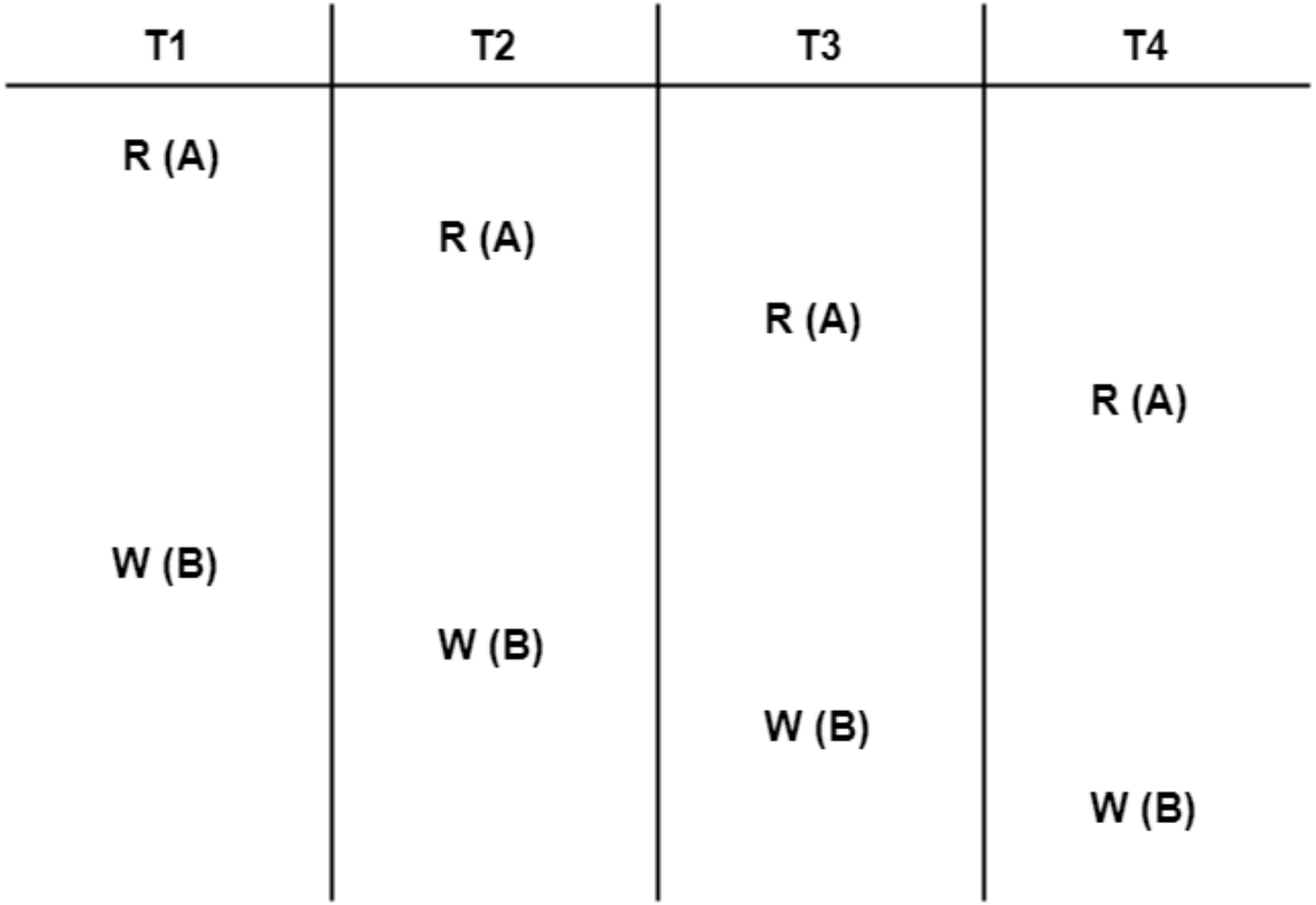
- Method-02:
  - Check if there exists any blind write operation. (Writing without reading is called as a blind write).
    - ✓ If there does not exist any blind write, then the schedule is surely not view serializable. Stop and report your answer.
    - ✓ If there exists any blind write, then the schedule may or may not be view serializable. Go and check using other methods.
  - Thumb Rule:
    - ✓ No blind write means not a view serializable.

Checking Whether a Schedule is View Serializable Or Not

- Method-03:
  - In this method, try finding a view equivalent serial schedule.
    - ✓ By using the above three conditions, write all the dependencies.
    - ✓ Then, draw a graph using those dependencies.
    - ✓ If there exists no cycle in the graph, then the schedule is view serializable otherwise not.

Practice Problems Based On View Serializability

- Problem-01: Check whether the given schedule S is view serializable or not?

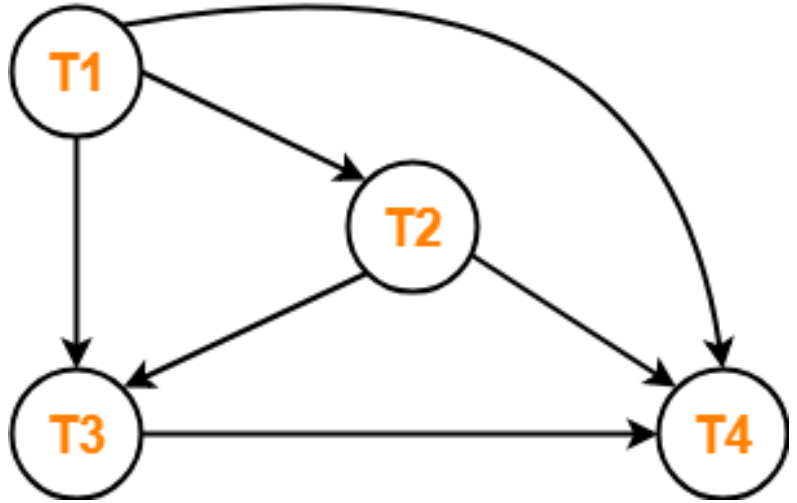


Practice Problems Based On View Serializability

- Solution:
- We know, if a schedule is conflict serializable, then it is surely view serializable.
- So, let us check whether the given schedule is conflict serializable or not.
- Checking Whether S is Conflict Serializable Or Not:
- Step-01: List all the conflicting operations and determine the dependency between the transactions.
  - W1(B) , W2(B) (T1 → T2)
  - W1(B) , W3(B) (T1 → T3)
  - W1(B) , W4(B) (T1 → T4)
  - W2(B) , W3(B) (T2 → T3)
  - W2(B) , W4(B) (T2 → T4)
  - W3(B) , W4(B) (T3 → T4)

Practice Problems Based On View Serializability

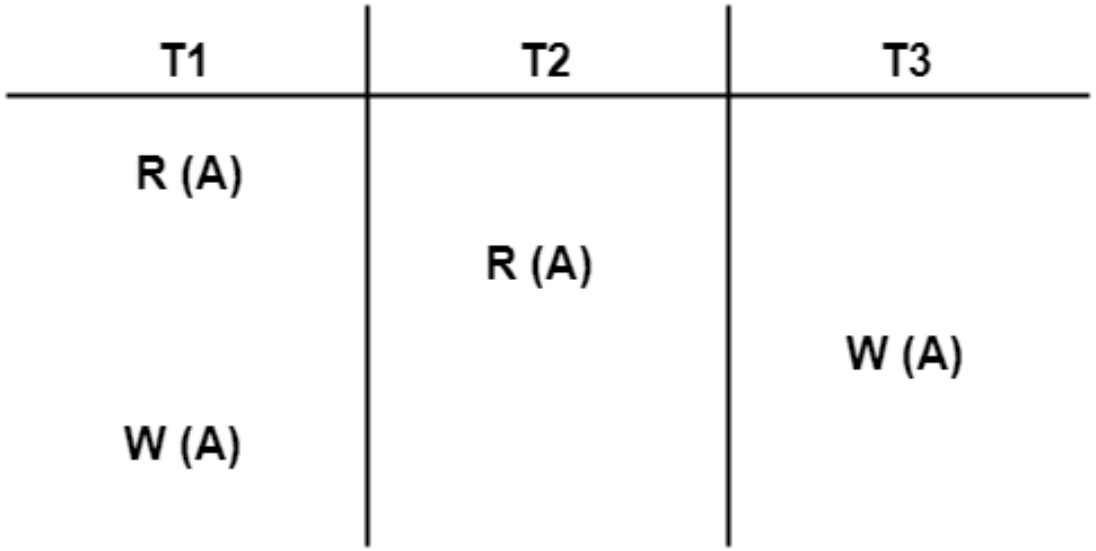
- W1(B) , W2(B) (T1 → T2)
- W1(B) , W3(B) (T1 → T3)
- W1(B) , W4(B) (T1 → T4)
- W2(B) , W3(B) (T2 → T3)
- W2(B) , W4(B) (T2 → T4)
- W3(B) , W4(B) (T3 → T4)



- Step-02: Draw the precedence graph
  - ✓ Clearly, there exists no cycle in the precedence graph.
  - ✓ Therefore, the given schedule S is conflict serializable.
  - ✓ Thus, we conclude that the given schedule is also view serializable.

Practice Problems Based On View Serializability

- Problem-02: Check whether the given schedule S is view serializable or not?



- Solution:
  - ✓ We know, if a schedule is conflict serializable, then it is surely view serializable.
  - ✓ So, let us check whether the given schedule is conflict serializable or not.

Practice Problems Based On View Serializability

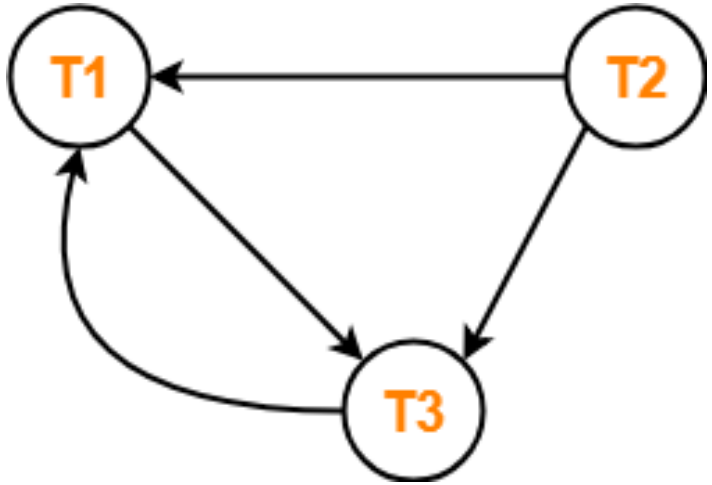
- Solution: Checking Whether S is Conflict Serializable Or Not-

  - ✓ Step-01: List all the conflicting operations and determine the dependency between the transactions.

    - R1(A) , W3(A) (T1 → T3)
    - R2(A) , W3(A) (T2 → T3)
    - R2(A) , W1(A) (T2 → T1)
    - W3(A) , W1(A) (T3 → T1)

  - ✓ Step-02: Draw the precedence graph

    - Clearly, there exists a cycle in the precedence graph.
    - Therefore, the given schedule S is not conflict serializable.



Practice Problems Based On View Serializability

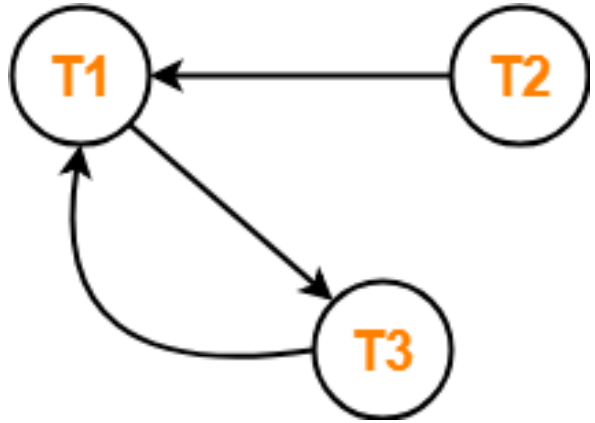
- Now, Since, the given schedule S is not conflict serializable, so, it may or may not be view serializable.
  - ✓ To check whether S is view serializable or not, let us use another method.
  - ✓ Let us check for blind writes.

- Checking for Blind Writes
  - ✓ There exists a blind write W3(A) in the given schedule S.
  - ✓ Therefore, the given schedule S may or may not be view serializable.

- Now,
  - ✓ To check whether S is view serializable or not, let us use another method.
  - ✓ Let us derive the dependencies and then draw a dependency graph.

Practice Problems Based On View Serializability

- Drawing a Dependency Graph
  - ✓ T1 firstly reads A and T3 firstly updates A.
  - ✓ So, T1 must execute before T3.
  - ✓ Thus, we get the dependency T1 → T3.
  - ✓ Final updation on A is made by the transaction T1.
  - ✓ So, T1 must execute after all other transactions.
  - ✓ Thus, we get the dependency (T2, T3) → T1.
  - ✓ There exists no write-read sequence.



- Clearly, there exists a cycle in the dependency graph.
- Thus, we conclude that the given schedule S is not view 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

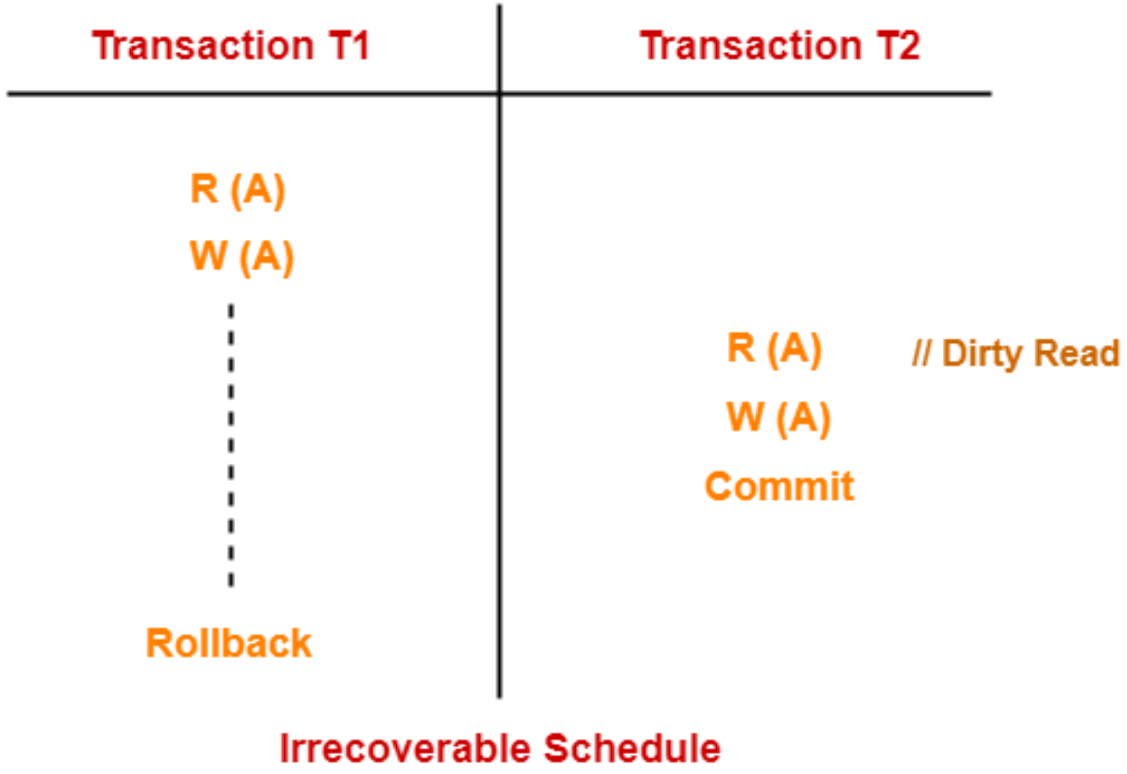
Irrecoverable Schedules

- 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 of Irrecoverable Schedules

- 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 can not 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

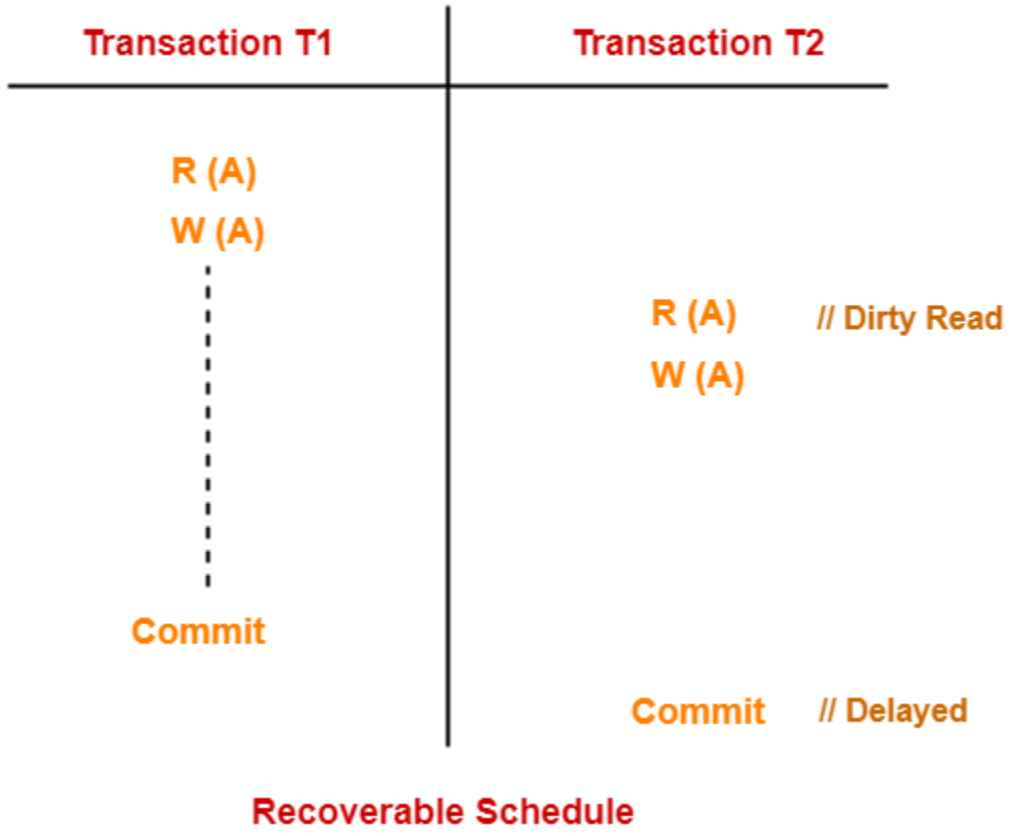
then such a schedule is known as an Recoverable Schedule.

Note:

- 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 of Recoverable Schedules

- 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.



Checking Whether a Schedule is Recoverable or Irrecoverable

- Method-01: Check whether the given schedule is conflict serializable or not?
  - ✓ If the given schedule is conflict serializable, then it is surely recoverable. Stop and report your answer.
  - ✓ If the given schedule is not conflict serializable, then it may or may not be recoverable. Go and check using other methods.
- Thumb Rules
  - ➢ All conflict serializable schedules are recoverable.
  - ➢ All recoverable schedules may or may not be conflict serializable.

Checking Whether a Schedule is Recoverable or Irrecoverable

- Method-01: Check if there exists any dirty read operation. (Reading from an uncommitted transaction is called as a dirty read)
  - ✓ If there does not exist any dirty read operation, then the schedule is surely recoverable. Stop and report your answer.
  - ✓ If there exists any dirty read operation, then the schedule may or may not be recoverable.
  - ✓ If there exists a dirty read operation, then follow the following cases:

Checking Whether a Schedule is Recoverable or Irrecoverable

- ✓ Case-01: If the commit operation of the transaction performing the dirty read occurs before the commit or abort operation of the transaction which updated the value, then the schedule is irrecoverable.
- ✓ Case-02: If the commit operation of the transaction performing the dirty read is delayed till the commit or abort operation of the transaction which updated the value, then the schedule is recoverable.

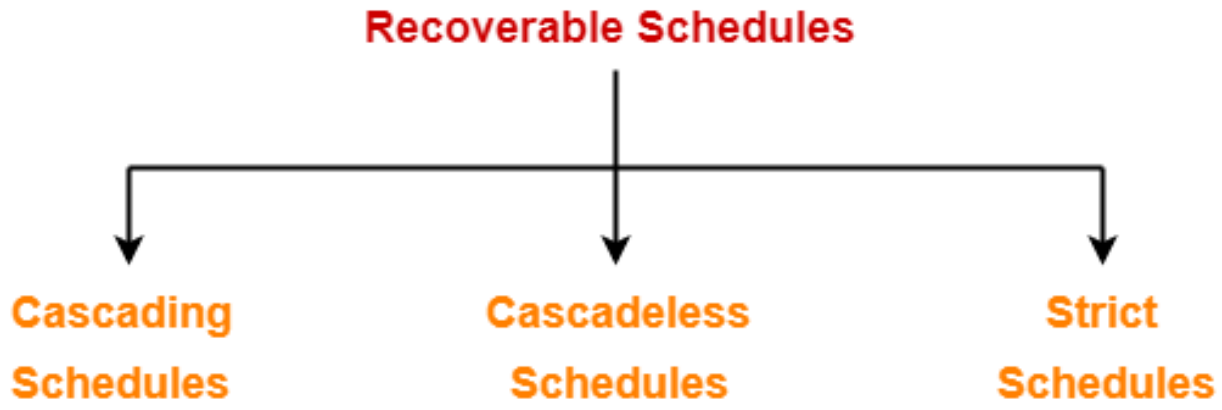
❑ Thumb Rules

- ➢ No dirty read means a recoverable schedule.

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.

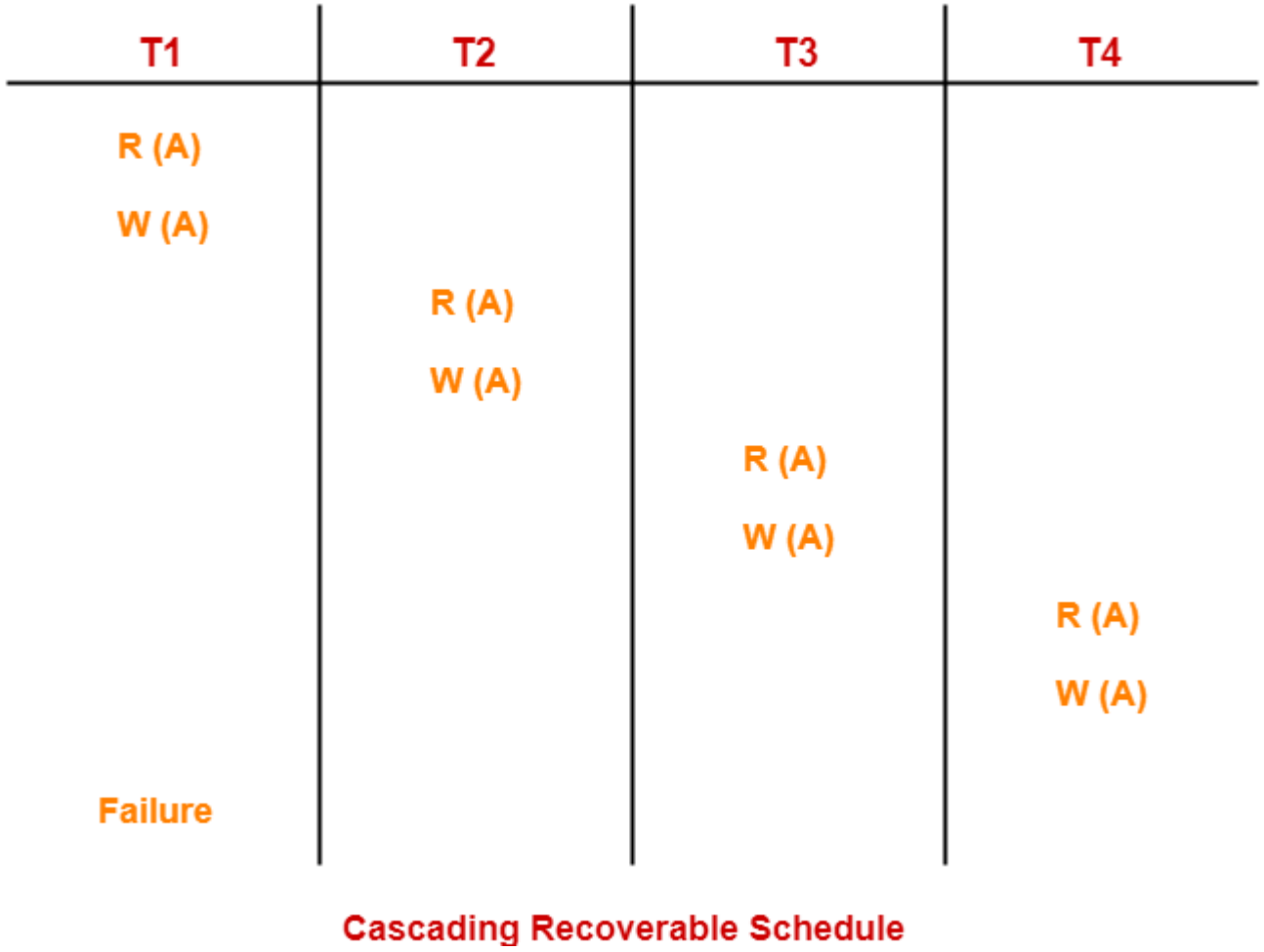


Cascading Schedule

- 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.

Example of Cascading Schedule

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

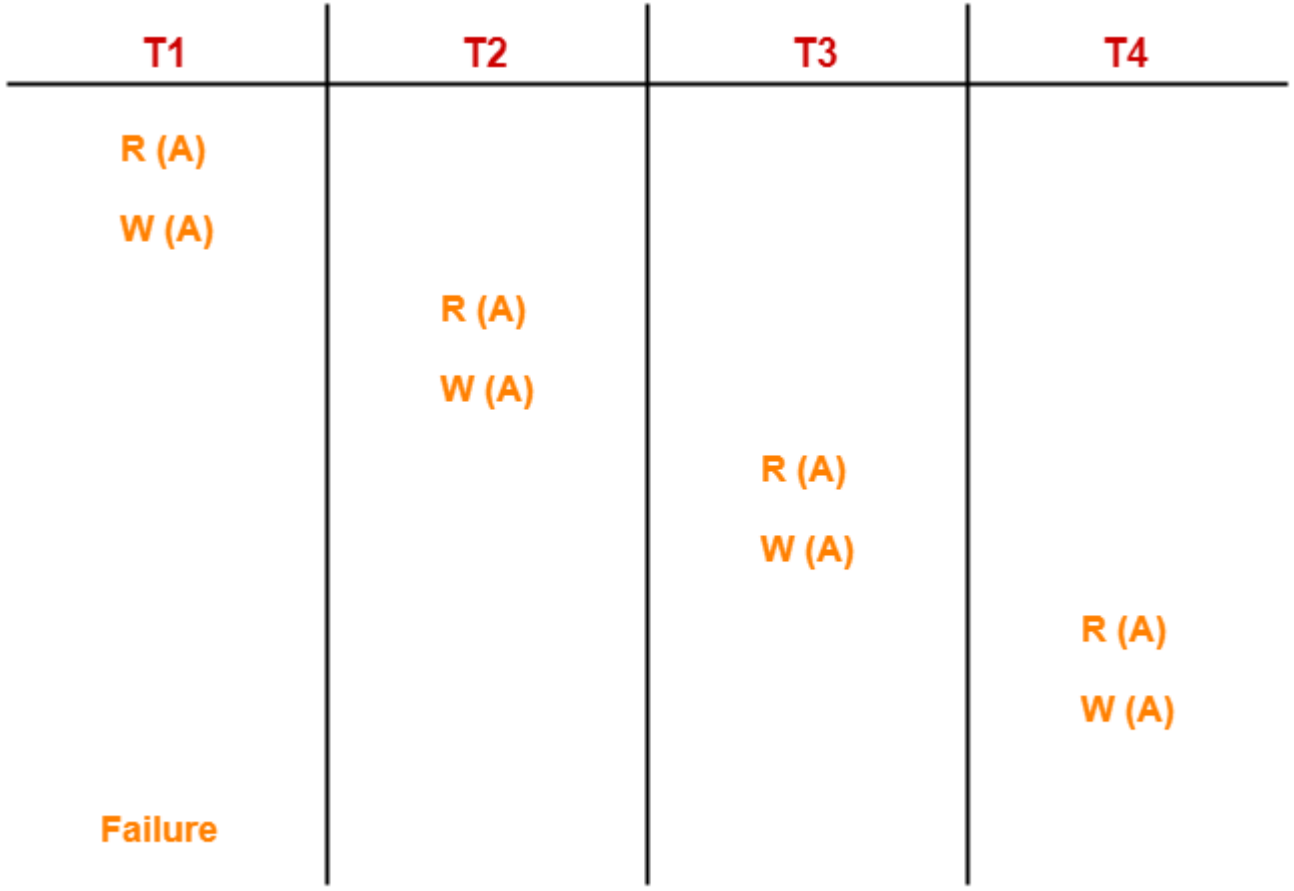


Cascading Recoverable Schedule

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

Example of Cascading Schedule

- 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.



Cascading Recoverable Schedule

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 of Cascadeless Schedule

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

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

Cascadeless Schedule

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

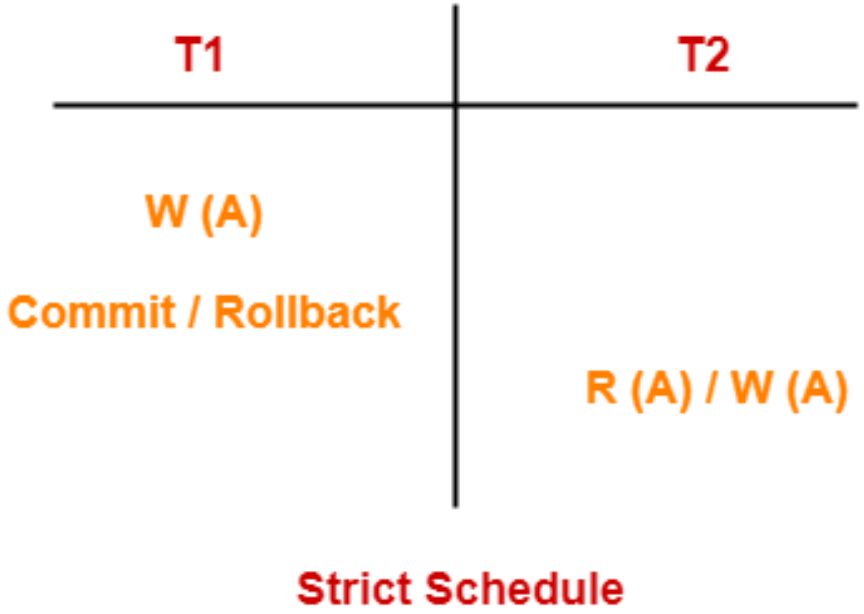
Cascadeless Schedule

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.

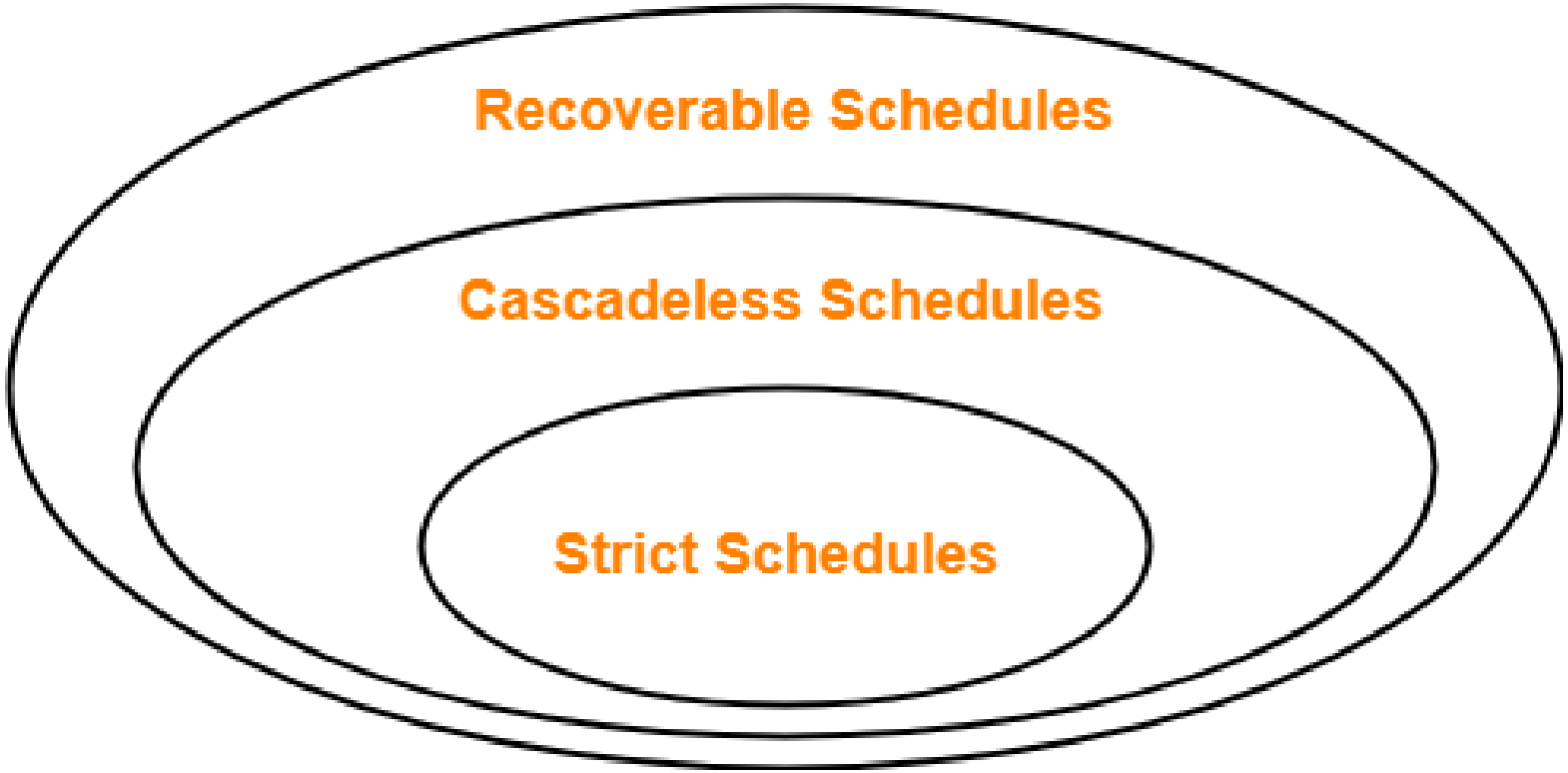
Example of Strict Schedule

- Remember,
  - ✓ Strict schedules are more strict than cascadeless schedules.
  - ✓ All strict schedules are cascadeless schedules.
  - ✓ All cascadeless schedules are not strict schedules.



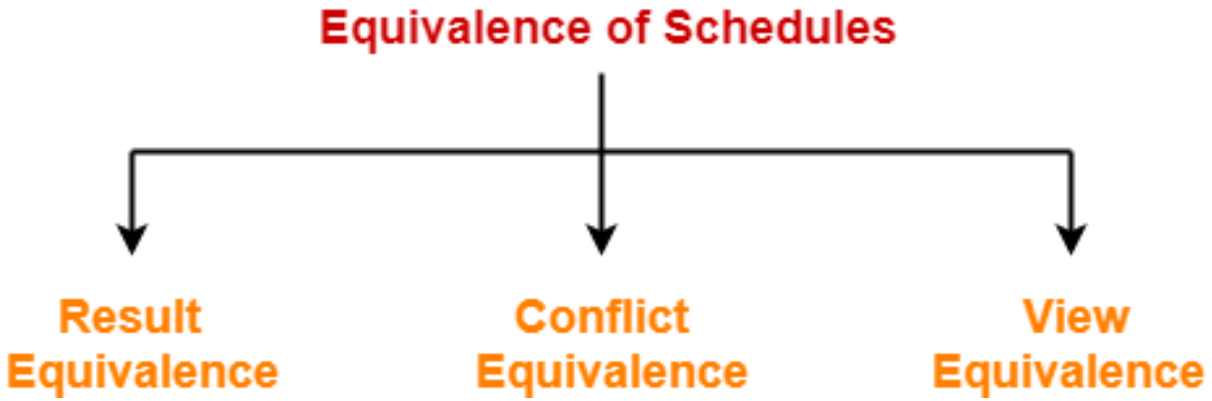
Strict Schedule

Schedules


Recoverable Schedules
Cascadeless Schedules
Strict Schedules

Equivalence of Schedules

- In DBMS, schedules may have the following three different kinds of equivalence relations among ther



- Result Equivalent Schedules:
  - ✓ If any two schedules generate the same result after their execution, then they are called as result equivalent schedules.
  - ✓ This equivalence relation is considered of least significance.
  - ✓ This is because some schedules might produce same results for some set of values and different results for some other set of values.

Equivalence of Schedules

- Conflict Equivalent Schedules: If any two schedules satisfy the following two conditions, then they are called as conflict equivalent schedules
  1. The set of transactions present in both the schedules is same.
  2. The order of pairs of conflicting operations of both the schedules is same.
- View Equivalent Schedules:
  - ✓ We have already discussed about View Equivalent Schedules.

Practice Problems Based On Equivalence Of Schedules

- Problem-01: Are the following three schedules result equivalent?

Schedule S1

T1	T2
R (X)	
X = X + 5	
W (X)	
R (Y)	
Y = Y + 5	
W (Y)	
	R (X)
	X = X x 3
	W (X)

Schedule S2

T1	T2
R (X)	
X = X + 5	
W (X)	
	R (X)
	X = X x 3
	W (X)
R (Y)	
Y = Y + 5	
W (Y)	

Schedule S3

T1	T2
	R (X)
	X = X x 3
	W (X)
R (X)	
X = X + 5	
W (X)	
R (Y)	
Y = Y + 5	
W (Y)	

Practice Problems Based On Equivalence Of Schedules

- Solution: To check whether the given schedules are result equivalent or not.

We will consider some arbitrary values of X and Y. Then, we will compare the results produced by each schedule. Those schedules which produce the same results will be result equivalent. Let X = 2 and Y = 5.

On substituting these values, the results produced by each schedule are

- Results by Schedule S1- X = 21 and Y = 10
- Results by Schedule S2- X = 21 and Y = 10
- Results by Schedule S3- X = 11 and Y = 10

Clearly, the results produced by schedules S1 and S2 are same. Thus, we conclude that S1 and S2 are result equivalent schedules.

Practice Problems Based On Equivalence Of Schedules

- Problem-02: Are the following two schedules conflict equivalent?

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

Schedule S1

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

Schedule S2

Practice Problems Based On Equivalence Of Schedules

- Solution: To check whether the given schedules are conflict equivalent or not,
  - ✓ We will write their order of pairs of conflicting operations.
  - ✓ Then, we will compare the order of both the schedules.
  - ✓ If both the schedules are found to have the same order, then they will be conflict equivalent.

- For schedule S1:

  The required order is
  - R1(A) , W2(A)
  - W1(A) , R2(A)
  - W1(A) , W2(A)

- For schedule S2:

  The required order is
  - R1(A) , W2(A)
  - W1(A) , R2(A)
  - W1(A) , W2(A)

- Clearly, both the given schedules have the same order.
- Thus, we conclude that S1 and S2 are conflict equivalent schedules.