PRACTICE PROBLEMS BASED ON CONFLICT SERIALIZABILITY-

Problem-1:

Check whether the given schedule S is conflict serializable or not-

S : $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$

Solution-

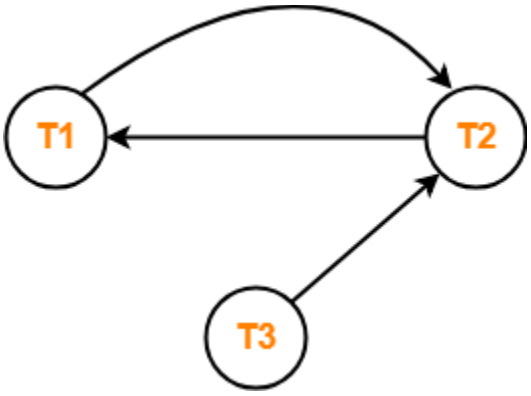
Step-01:

List all the conflicting operations and determine the dependency between the transactions-

- $R_2(A)$, $W_1(A)$ ($T_2 \rightarrow T_1$)
- $R_1(B)$, $W_2(B)$ ($T_1 \rightarrow T_2$)
- $R_3(B)$, $W_2(B)$ ($T_3 \rightarrow T_2$)

Step-02:

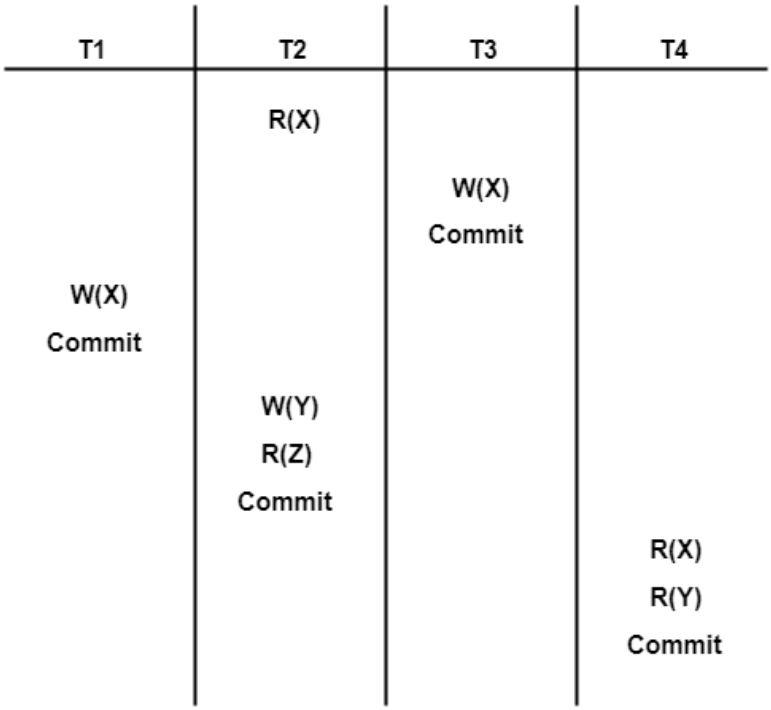
Draw the precedence graph-



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

Problem-2:

Check whether the given schedule S is conflict serializable and recoverable or not-



Solution-

Checking Whether S is Conflict Serializable Or Not-

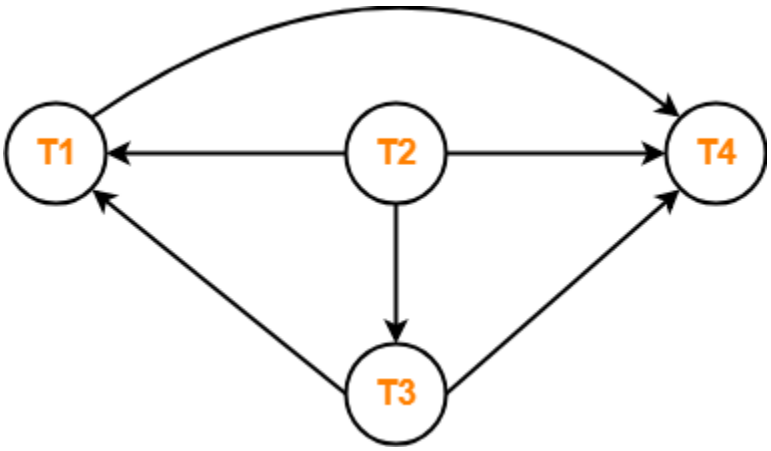
Step-01:

List all the conflicting operations and determine the dependency between the transactions-

- $R_2(X)$, $W_3(X)$ $(T_2 \rightarrow T_3)$
- $R_2(X)$, $W_1(X)$ $(T_2 \rightarrow T_1)$
- $W_3(X)$, $W_1(X)$ $(T_3 \rightarrow T_1)$
- $W_3(X)$, $R_4(X)$ $(T_3 \rightarrow T_4)$
- $W_1(X)$, $R_4(X)$ $(T_1 \rightarrow T_4)$
- $W_2(Y)$, $R_4(Y)$ $(T_2 \rightarrow T_4)$

Step-02:

Draw the precedence graph-



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

Checking Whether S is Recoverable Or Not-

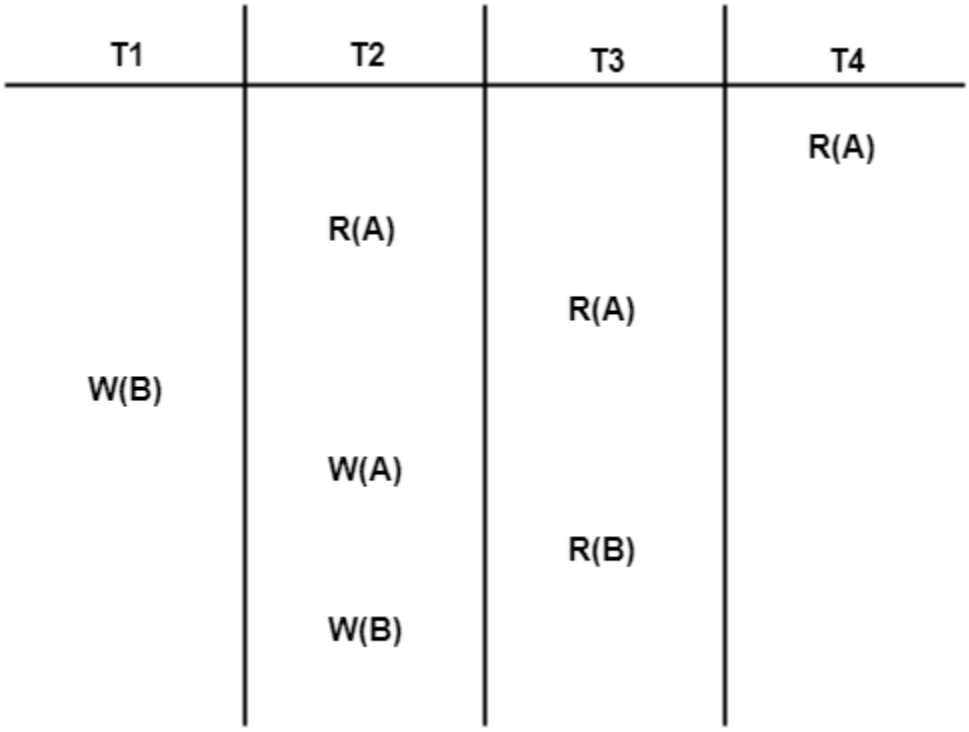
- Conflict serializable schedules are always recoverable.
- Therefore, the given schedule S is recoverable.

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**.

Problem-03:

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



Solution-

Checking Whether S is Conflict Serializable Or Not-

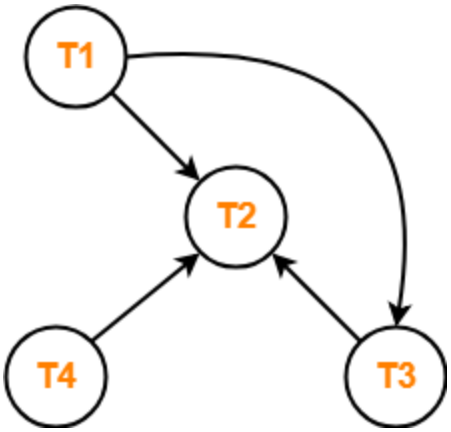
Step-01:

List all the conflicting operations and determine the dependency between the transactions-

- $R_4(A)$, $W_2(A)$ $(T_4 \rightarrow T_2)$
- $R_3(A)$, $W_2(A)$ $(T_3 \rightarrow T_2)$
- $W_1(B)$, $R_3(B)$ $(T_1 \rightarrow T_3)$
- $W_1(B)$, $W_2(B)$ $(T_1 \rightarrow T_2)$
- $R_3(B)$, $W_2(B)$ $(T_3 \rightarrow T_2)$

Step-02:

Draw the precedence graph-



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

Finding the Serialized Schedules-

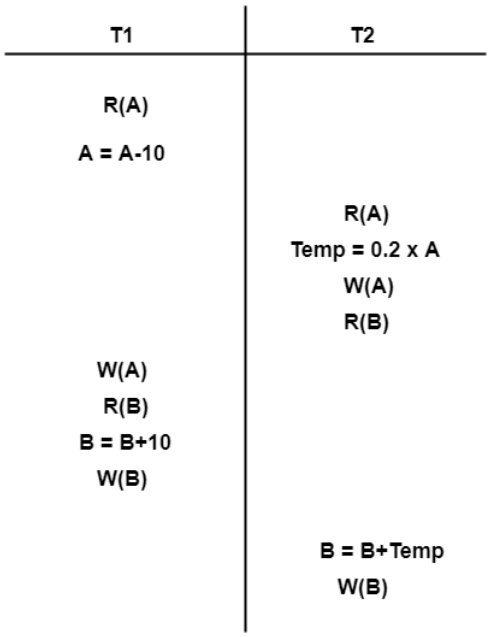
- All the possible topological orderings of the above precedence graph will be the possible serialized schedules.
- The topological orderings can be found by performing the **Topological Sort** of the above precedence graph.

After performing the topological sort, the possible serialized schedules are-

1. $T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_2$
2. $T_1 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2$
3. $T_4 \rightarrow T_1 \rightarrow T_3 \rightarrow T_2$

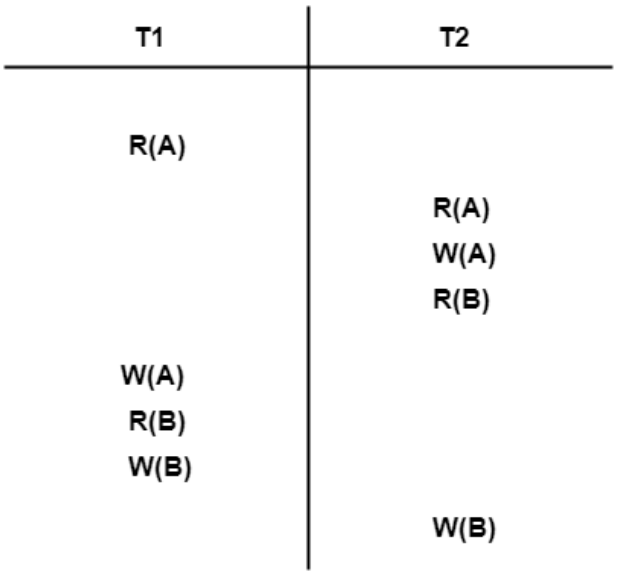
Problem-04:

Determine all the possible serialized schedules for the given schedule-



Solution-

The given schedule S can be rewritten as-



This is because we are only concerned about the read and write operations taking place on the database.

Checking Whether S is Conflict Serializable Or Not-

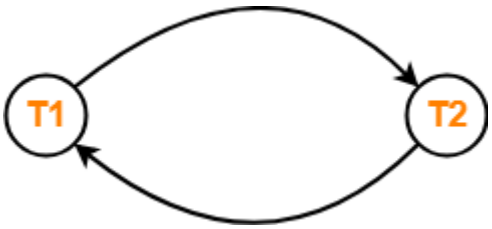
Step-01:

List all the conflicting operations and determine the dependency between the transactions-

- $R_1(A)$, $W_2(A)$ $(T_1 \rightarrow T_2)$
- $R_2(A)$, $W_1(A)$ $(T_2 \rightarrow T_1)$
- $W_2(A)$, $W_1(A)$ $(T_2 \rightarrow T_1)$
- $R_2(B)$, $W_1(B)$ $(T_2 \rightarrow T_1)$
- $R_1(B)$, $W_2(B)$ $(T_1 \rightarrow T_2)$
- $W_1(B)$, $W_2(B)$ $(T_1 \rightarrow T_2)$

Step-02:

Draw the precedence graph-



- 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.