**Precedence Graph For Testing Conflict Serializability in DBMS**

**Conflict serializable**

Definition: A given non-serial schedule is conflict equivalent to a serial schedule then it is conflict serializable.

**Precedence Graph** or **Serialization Graph** is used commonly to test Conflict Serializability of a schedule.

Serial schedule : correct non serial : wrong

Non serial schedule

Algorithm To draw Precedence Graph

1. Create a node T in the graph for each participating transaction in the schedule.
2. For the conflicting operation read\_item(X) and write\_item(X) – If a Transaction Tj executes a read\_item (X) after Ti executes a write\_item (X), draw an edge from Ti to Tj in the graph.
3. For the conflicting operation write\_item(X) and read\_item(X) – If a Transaction Tj executes a write\_item (X) after Ti executes a read\_item (X), draw an edge from Ti to Tj in the graph.
4. For the conflicting operation write\_item(X) and write\_item(X) – If a Transaction Tj executes a write\_item (X) after Ti executes a write\_item (X), draw an edge from Ti to Tj in the graph.
5. The Schedule S is not conflict serializable to any serial schedule if there is cycle in the precedence graph
6. If there is no cycle in the precedence graph, it means we can construct a serial schedule S’ which is conflict equivalent to the schedule S.
7. The serial schedule S’ can be found by Topological Sorting of the acyclic precedence graph. Such schedules can be more than 1.

**Examples on Precedence Graph**

Consider following concurrent schedule ,draw precedence graph and check whether given schedule is conflict serializable or not

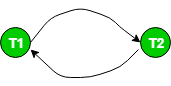
|  |  |
| --- | --- |
| T1 | T2 |
| READ(X) |  |
| READ(Y) |  |
|  | WRITE(X) |
| WRITE(X) |  |
|  | READ(Y) |

**Creating Precedence graph:**

1. Make two nodes corresponding to Transaction T1 and T2.  
   2
2. For the conflicting pair r1(x) w2(x), where r1(x) happens before w2(x), draw an edge from T1 to T2.

3

1. For the conflicting pair w2(x) w1(x), where w2(x) happens before w1(x), draw an edge from T2 to T1.



1. Since the graph is cyclic(as there is cycle present in graph), we can conclude that it is not conflict serializable to any serial schedule.

|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| READ(X) |  |  |
|  |  | READ(Y) |
| WRITE(X) |  |  |
|  | WRITE(Y) |  |
|  |  | READ(X) |
|  | WRITE(X) |  |

1. Consider following concurrent schedule ,draw precedence graph and check whether given schedule is conflict serializable or not

**Creating Precedence graph:**

1. Make two nodes corresponding to Transaction T1 ,T2 and T3

1. For the conflicting pair READ(X) in T1 and WRITE(X) in T2 , where r1(X) happens before w2(X), draw an edge from T1 to T2.

1. For the conflicting pair READ(Y) in T3 and WRITE(Y) in T2 , where r3(Y) happens before w2(Y), draw an edge from T3 to T2.

1. For the conflicting pair WRITE(X) in T1 and READ(X) in T3 , where w1(x) happens before r3(x), draw an edge from T1 to T3.

1. Since the graph is acyclic(as there is no cycle present in graph), we can conclude that it is a conflict serializable to any serial schedule.

1. In Topological Sort, we first select the node with indegree 0, which is T1. This would be followed by T3 and T2.

Calculate indgree of all vertices : T1 Is 0 , T2 is 2, and T3 is1

Arrange vertices in ascending order of their indegree : T1,T3,T1

|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| READ(P) |  |  |
|  | READ(P) |  |
|  |  | READ(Q) |
| WRITE(P) |  |  |
|  |  | WRITE(Q) |
|  | WRITE(P) |  |
| READ(Q) |  |  |
|  | WRITE(Q) |  |

So, S1 is conflict serializable since it is conflict equivalent to the serial schedule T1 T3 T2.

1. Consider following concurrent schedule ,draw precedence graph and check whether given schedule is conflict serializable or not

1. Consider following concurrent schedule ,draw precedence graph and check whether given schedule is conflict serializable or not

|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| READ(A) |  |  |
|  | READ(A) |  |
|  |  | READ(B) |
| WRITE(A) |  |  |
|  |  | WRITE(B) |
|  | WRITE(A) |  |
| READ(B) |  |  |
|  | READ(B) |  |
|  | WRITE(B) |  |
| WRITE(B) |  |  |