LET'S START WITH DBMS :)

Conflict-Serializability

Serializibility : Serializability is a property of a schedule where the outcome is equivalent to some serial execution of the transactions.

Conflict-Serializability : A schedule is said to be conflict serializable when one of its conflict equivalent is serializabe. So basically, if a schedule can be transformed into a serial schedule by swapping non-conflicting operations, then the schedule is conflict serializable.

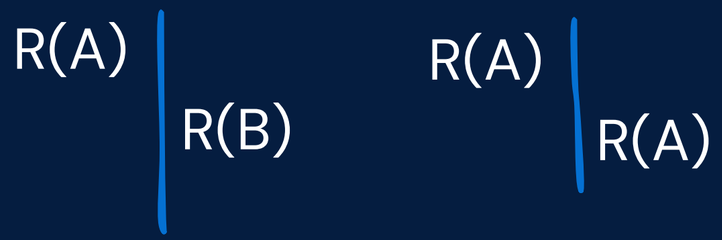
Conflict equivalent : If a schedule S1 is formed after swapping adjacent non-conflicting operations/pairs in a given schedule S, then S1 and S are conflict equivalent.

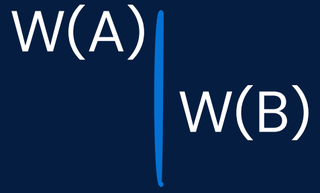
Conflict pairs : Read-Write, Write-Read , Write-Write(perfomed on the same data item)
Non-conflict pairs : Read-Read (perfomed on the same data item), Read-Read (perfomed on the different data item), Write-Write(perfomed on the different data item)

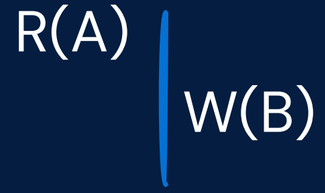
LET'S START WITH DBMS :)

Conflict-Serializability

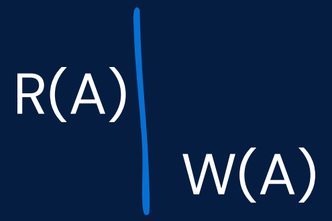
Non- Conflict pairs

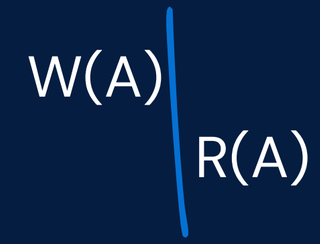






Conflict pairs





LET'S START WITH DBMS :)

Conflict-Serializability

How to achieve conflict equivalent schedule :

When a schedule can be transformed into a serial schedule by swapping adjacent non-conflicting operations. Conflicts arise when two transactions access the same data item, and at least one of them is a write operations.

Now, why are we only swapping the non-conflict pairs and not the conflict ones?

So if we swap the conflict pairs, the order of exceution if it was

T1 : R(A)

T2: W(A)

the results values may change as first we were reading A and then writing/modifying it, but now it will be writing A and then reading the modified value so the result might change if we change the order of execution.

LET'S START WITH DBMS :)

Conflict-Serializability

Q. Find a conflict equivalent for a schedule S1

S1

T1	T2
R(X)	
W(X)	
	R(X)
R(Y)	
W(Y)	
	R(Y)

LET'S START WITH DBMS :)

Conflict-Serializability

Q. Find a conflict equivalent for a schedule S1

1. Find the adjacent non-conflicting pairs and do a swap

T2 : R(X) T1: R(Y)

T1	T2
R(X)	
W(X)	
R(Y)	
	R(X)
W(Y)	
	R(Y)

T1	T2
R(X)	
W(X)	
	R(X)
R(Y)	
W(Y)	
	R(Y)

S1

LET'S START WITH DBMS :)

Conflict-Serializability

Q. Find a conflict equivalent for a schedule S1

2. After the first swap again search for adjacent non-conflicting pairs and swap if any
T2 : R(X) T1: W(Y)

T1	T2
R(X)	
W(X)	
R(Y)	
W(Y)	
	R(X)
	R(Y)

So, S1 is a serializable schedule
as S1' has a serial execution(i.e its conflict equivalent)

T1	T2
R(X)	
W(X)	
R(Y)	
	R(X)
W(Y)	
	R(Y)

S1 after swap

LET'S START WITH DBMS :)

When done on the same data item

- Read-Write (RW) Conflict
- Write-Read (WR) Conflict
- Write-Write (WW) Conflict

Conflict-Serializability

How to check whether a schedule is conflict-serializable or not?

Conflicts occur when two operations from different transactions access the same data item and at least one of them is a write operation.

Conflict graph/ precedence graph : A conflict graph, or precedence graph, is a directed graph used to determine conflict serializability. The nodes represent transactions, and the edges represent conflicts between transactions

LET'S START WITH DBMS :)

Conflict-Serializability

Conflict graph/ precedence graph:

- **Nodes:** Each transaction in the schedule is represented as a node in the graph.

- **Edges:** An edge from transaction T(X) to transaction T(Y)(denoted T(X)→T(Y)) is added if
  a. an operation of T(X) conflicts with an operation of T(Y) and
  b.T(X) operation precedes T(Y) operation in the schedule.

- **Cycle Detection:** The schedule is conflict-serializable if and only if the conflict graph is acyclic. If there are no cycles in the graph, it means that the schedule can be serialized without violating the order of conflicting operations.

LET'S START WITH DBMS :)

Conflict-Serializability

Q. Check if the given schedule is conflict-serializable or not?

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

- T1 reads A
- T2 reads A
- T1 writes A
- T3 writes A
- T2 writes B
- T3 reads B

LET'S START WITH DBMS :)

Conflict-Serializability

Q. Check if the given schedule is conflict-serializable or not?

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

Step 1: Find the conflict in the schedule

- RW Conflict (T1, T3) on A
- RW Conflict (T2, T3) on A
- RW Conflict (T2.T1) on A
- WW Conflict (T1, T3) on A
- WR Conflict(T2.T3) on B

Write-Write conflict
W(B) - W(B)
W(A)-W(A)

Write-Read conflict
W(B) - R(B)
W(A)-R(A)

Read-write conflict
R(B) - W(B)
R(A)-W(A)

Step 2: Find the nodes

Each transaction T1 T2, T3 is a node in the graph

Step 3: Find the edges/conflicts

- T1→T3: Because T1 reads A before T3 writes A.
- T2→T3: Because T2 reads A before T3 writes A
- T1→T3: Because T1 writes A before T3 writes A.
- T2→T3: Because T2 writes B before T3 reads B.
- T2->T1 : Because T2 reads A before T1 writes A.

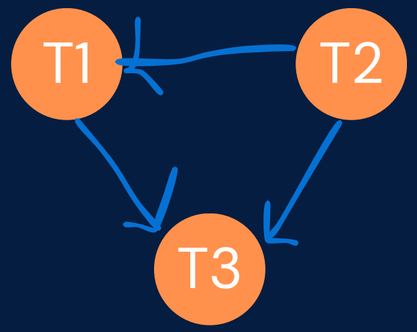
LET'S START WITH DBMS :)

Conflict-Serializability

Q. Check if the given schedule is conflict-serializable or not?

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

Step 4 : Draw the graph



edges

- T2→T1
- T1→T3
- T2→T3

Step 5 : If the graph has cycle it is not confliict-serializable or serializable, if not lets find the serial execution of transactions

Since the conflict graph is acyclic, the schedule is conflict-serializable

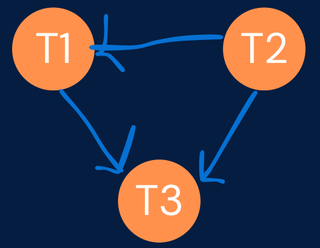
LET'S START WITH DBMS :)

Conflict-Serializability

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

Q. Check if the given schedule is conflict-serializable or not?

Step 6 : Lets find the serial execution of transactions



edges
T2→T1
T1→T3
T2→T3

Possible combinations are :
T1->T2->T3
T1->T3->T2
T2->T1->T3
T2->T3->T1
T3->T2->T1
T3->T1->T2

Find the indegree(the number of edges directed into that node) and if its 0 it can be the first in serial execution
T1 - 1 ,T2- 0, T3- 2 , T2 would be the first as indegree is 0

- T2 must precede T1
- T1 must precede T3

Therefore, one possible equivalent serial schedule is **T2→T1→T3.**