LET'S START WITH DBMS :)

Serializability and its types

Serializability: It ensures that concurrent transactions yield results that are consistent with some serial execution i.e the final state of the database after executing a set of transactions concurrently should be the same as if the transactions had been executed one after another in some order.

In case of concurrent schedule consistentcy issue may arise because of non-serial execution and we do serializability there, serial schedules are aready serial

Consider nodes as transactions, and edges represent conflicts and detect if the resulting graph has cycles

LET'S START WITH DBMS :)

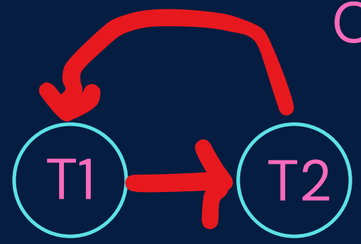
T1	T2
R(A)	
W(A)	
COMMIT	
	R(A)
	W(A)

SERIAL SCHEDULE

Nodes: T1, T2
Edges : T1->T2
Cycle : No

T1	T2
R(A)	
	R(A)
	W(A)
W(A)	
COMMIT	

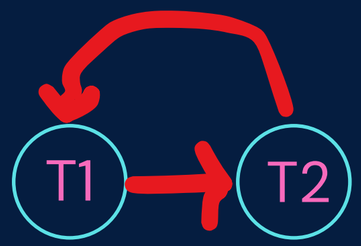
CONCURRENT SCHEDULE



Nodes: T1, T2
Edges : T1->T2, T2->T1
Cycle : Yes

LET'S START WITH DBMS :)

T1	T2
R(A)	
	R(A)
	W(A)
W(A)	
COMMIT	



CONCURRENT SCHEDULE

Nodes: T1, T2

Edges : T1->T2, T2->T1

Cycle : Yes

Now since the schedule has cycle we need to serialize this
So, there are two methods we can follow for the same:

1. Conflict Serializability
2. View Serializability

Why serializing them?

- To avoid consistentcy issue which may arise because of non-serial execution

LET'S START WITH DBMS :)

Serializability and its types

Types of Serializability

- Conflict-Serializability
- View-Serializability