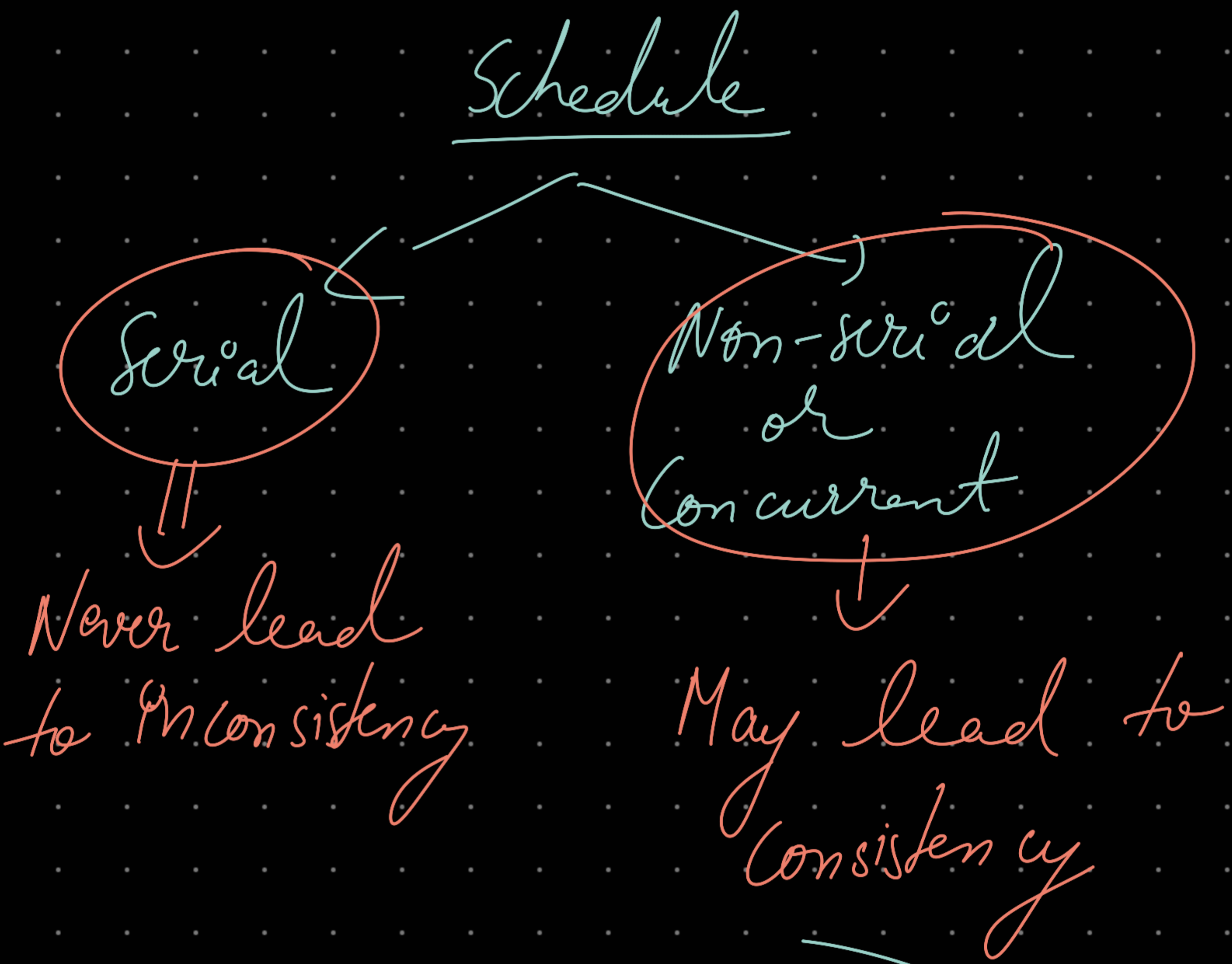
[Schedule]

↳ Represents the order in which the operations of transactions are executed.



Schedule

Serial

Never lead to inconsistency

Non-serial or Concurrent

May lead to consistency

E.g → T_1 → a_1 a_2

T_2 → b_1 b_2

Consider this as SQL queries/operations.

Schedule possible

Time

T_1	T_2
a_1	
a_2	
	b_1
	b_2

T_1	T_2
	b_1
	b_2
a_1	
a_2	

Serial schedule

T_1	T_2
a_1	
	b_1
a_2	
	b_2

T_1	T_2
	b_1
a_1	
	b_2
a_2	

↓

Non-serial / Concurrent

* Non-serial / concurrent schedule leads may lead to inconsistency

↳ not always.

Serializibility

↳ A transaction schedule is serializable if its outcome is equal to outcome of its transaction executed serially.

E.g →

T_1	T_2
a_1	
a_2	
	b_1
	l_2

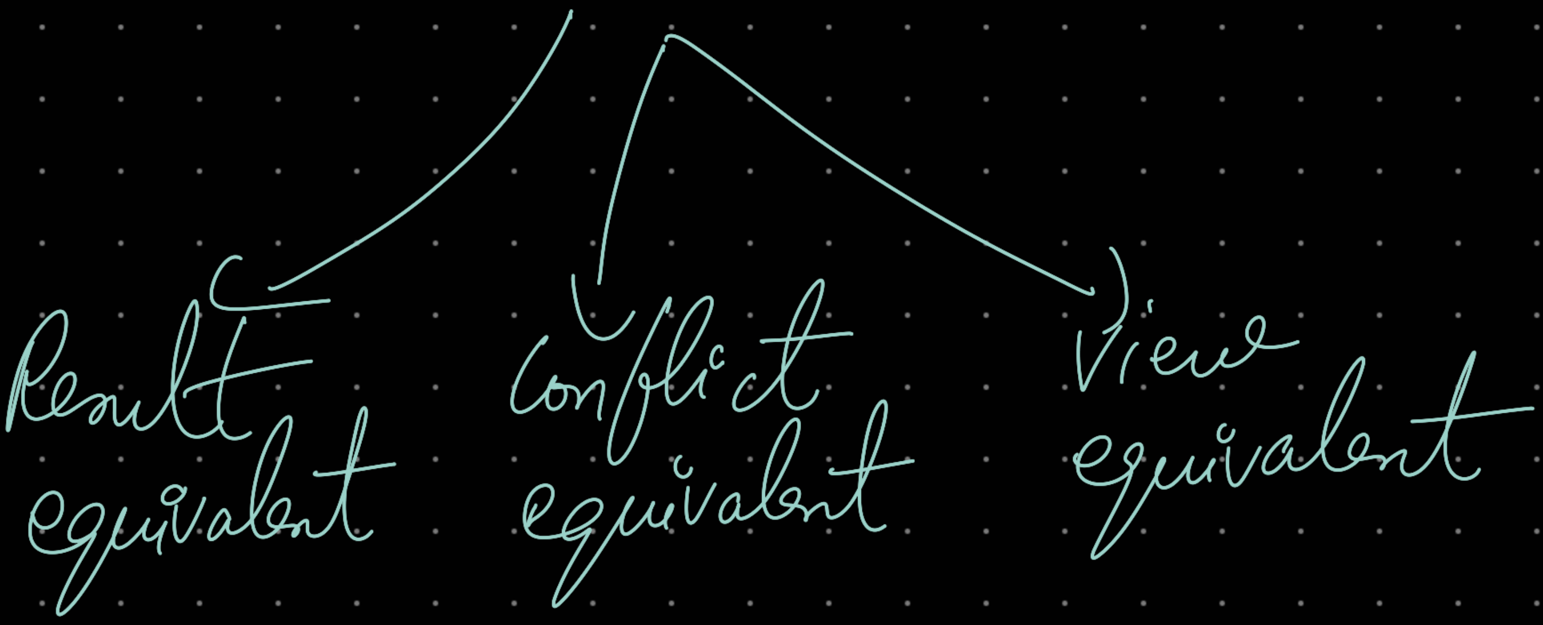
T_1	T_2
a_1	
	b_1
a_2	
	b_2

Serial

Non-serial

If both have same outcome

The schedule is serializable



Result equivalent

Conflict equivalent

View equivalent

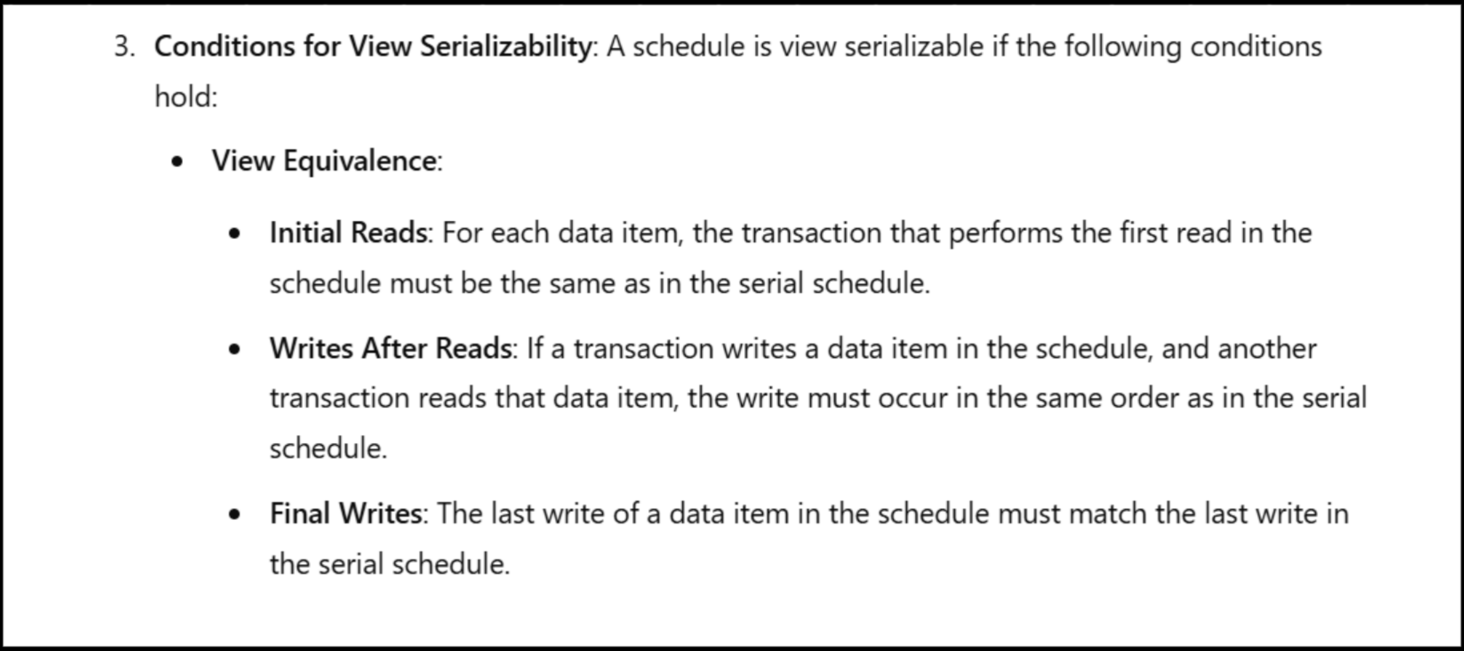
* Result equivalent

↳ If two schedule gives same final database state

* Conflict equivalent

↳ If order of conflicting operation is same in bothe schedule

* View equivalent



3. **Conditions for View Serializability**: A schedule is view serializable if the following conditions hold:
   - **View Equivalence**:
     - **Initial Reads**: For each data item, the transaction that performs the first read in the schedule must be the same as in the serial schedule.
     - **Writes After Reads**: If a transaction writes a data item in the schedule, and another transaction reads that data item, the write must occur in the same order as in the serial schedule.
     - **Final Writes**: The last write of a data item in the schedule must match the last write in the serial schedule.