COMPUTER SCIENCE

Database Management System

Transaction & Concurrency Control

Lecture_3

Vijay Agarwal sir

01 Serializable Schedule

02 Conflict & View Serializable

Transaction

Read (A)
A = A - 1000
Write (A)

Read (B)
B = B + 1000
Write (B)

} 1 Transaction

A C I D

A: Atomicity

C: Consistency

I: Isolation

D: Durability

Transaction Concept

- A transaction is a unit of program execution that accesses and possibly updates various data items.
- E.g. Transaction to transfer Rs 500 from account A to account B:

1. read (A)
2. A : = A – 500
3. write(A)
4. read(B)
5. B: = B + 500
6. write(B)

ACID Properties

- A transaction is a unit of program execution that accesses and possibly updates various data items. To preserve the integrity of data the database system must ensure:
  - A. Atomicity
  - C. Consistency
  - I. Isolation
  - D. Durability

Atomicity :

Either execute All operation of the transaction successfully (or) None of them.

[FULL (or) None].

Reason of Transaction failure.

- Power failure.
- Hardware crash.
- S/w crash.
- system & N/w Error etc.

Due to Any of these Reason if Atomicity is getting failed then Recovery Management Component are there.

- If Transaction is failed before Commit then Recovery Management Component ROLLBACK [UNDO ALL MODIFICATION] the transactions.

Log's : Log Contain All the Activies of the transaction.

ACID Properties

1. **Atomicity:** Either all operations of the transaction are properly reflected in the database or none are.
2. **Consistency:** Execution of a transaction in isolation preserves the consistency of the database.
3. **Isolation:** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.

② Consistency : Before & After the transaction DATABASE Must be Consistent.

$A = A - x$

$B = B + x$

Sum of A +B

Constant

eg Before

A: 4000

+B: 2000

6000

500

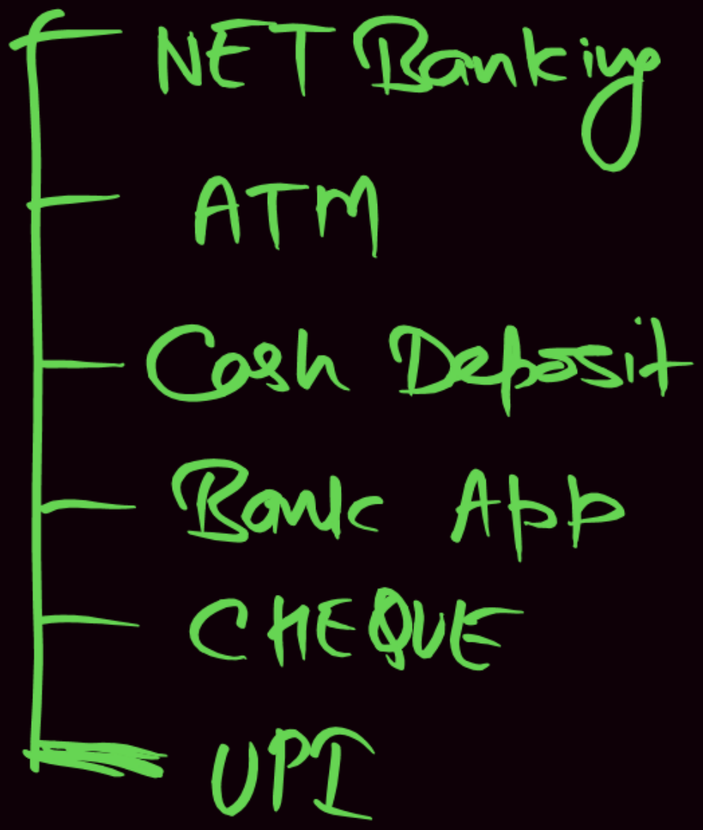
AFTER

A: 3500

+B: 2500

6000

③ **Isolation** : Concurrent execution of 2 or More transaction Should be equal to any serial Schedule.



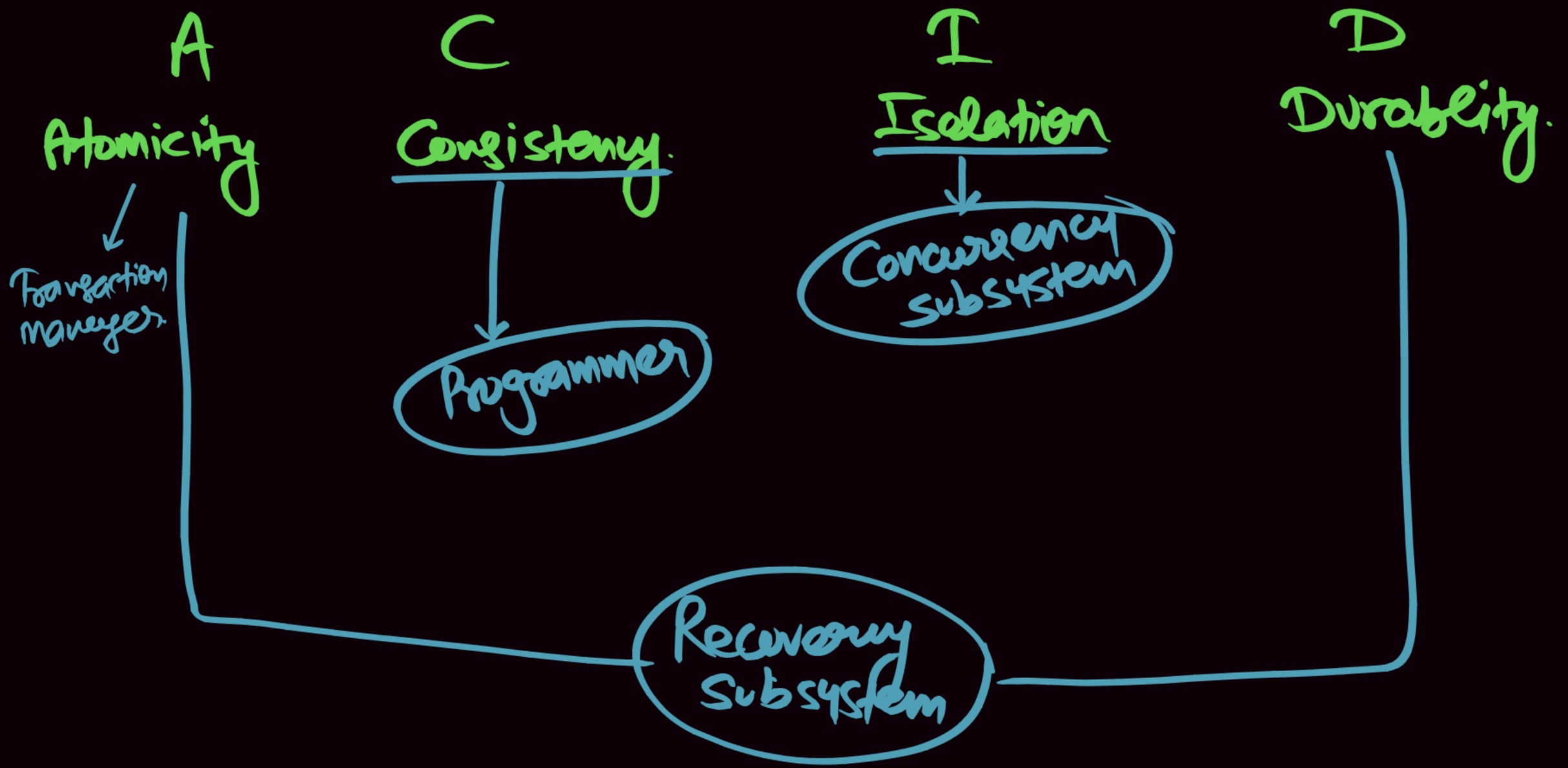
④ Durability : Any Change in the Database must Persist for a Long period of time.

(OR)

Database must be able to Recover Under Any Case of failure.

- That is, for every pair of transactions T_i and T_j, it appears to T_i that either T_j, finished exection before T_i started, or T_j started execution after T_i finished.

- Durability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.


A
Atomicity
Transaction manager.
C
Consistency.
Programmer
I
Isolation
Concurrency subsystem
D
Durability.
Recovery subsystem

Read (A): Accessing the Data Item A from DB to MM.

Write(A): Update the Data Item A.

Commit: Indicate successfully Completion of transaction

(or)

Indicate Transaction execute successfully.

Abort/kill ⇒ Fail.

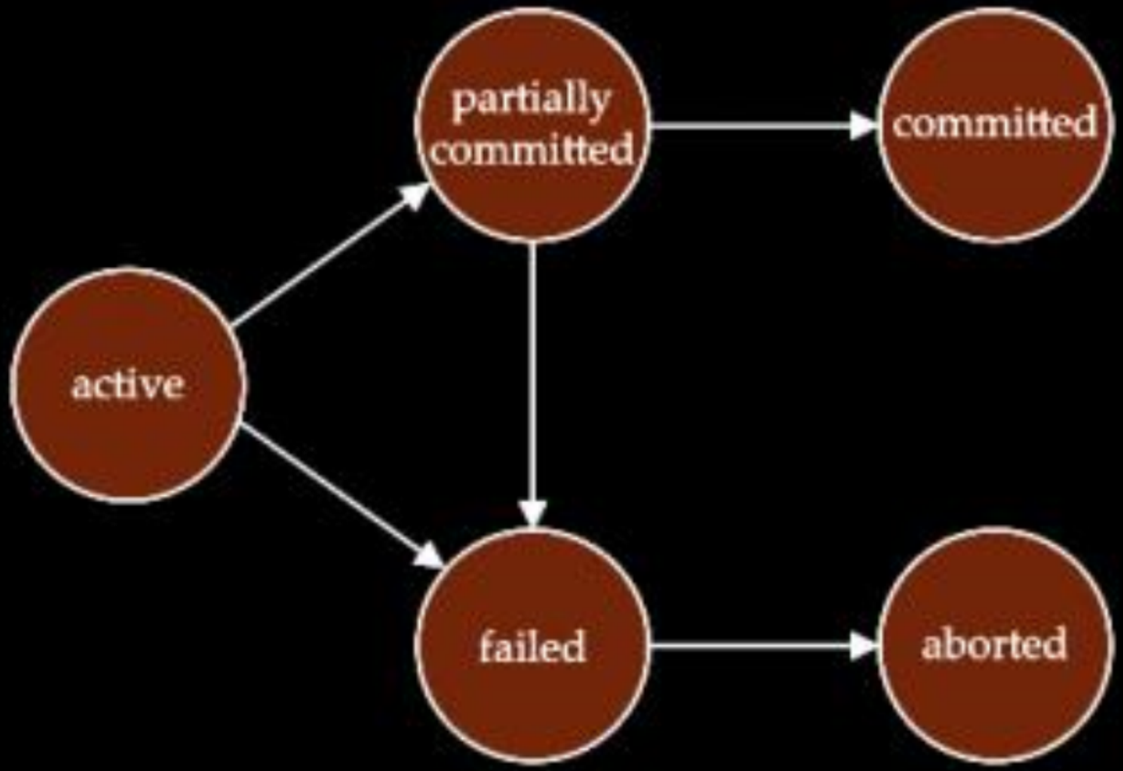
Transaction State

- Active : the initial state; the transaction stays in this state while it is executing.
- Partially committed : after the final statement has been executed.
- Failed: after the discovery that normal execution can no longer proceed.
- Aborted: after the transaction has been rolled back and the database restored to its state prior to the start of the transaction. Two options after it has been aborted:

- Restart the transaction
  - Can be done only if no internal logical error
- Kill the transaction

- Committed: After successful completion.

Transaction State (Cont.)

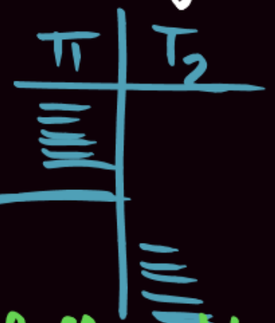


Schedule is a.

Schedule: Time order sequence of Two or More transaction.

① Serial Schedule

Execution of One transaction Completely then another will start.

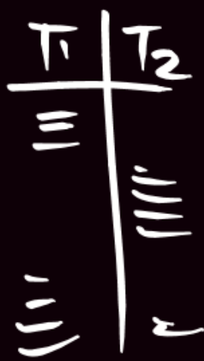


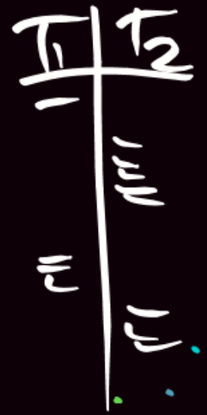
T1 followed by T2
<T1 T2>

T2 followed by T1
<T2 T1>

② Non-Serial Schedule.

[Interleaved Execution of Two or More transaction]





& many more.

2 Transaction then 2! ⟹ 2 Serial Schedule

$\langle T_1, T_2 \rangle$ T_1 followed by T_2

$\langle T_2, T_1 \rangle$ T_2 followed by T_1.

If 3 transaction then 3! = 6 Serial Schedule.

$\langle T_1\ T_2\ T_3 \rangle$

$\langle T_1\ T_3\ T_2 \rangle$

$\langle T_2\ T_1\ T_3 \rangle$

$\langle T_2\ T_3\ T_1 \rangle$

$\langle T_3\ T_1\ T_2 \rangle$

$\langle T_3\ T_2\ T_1 \rangle$

} 6 Serial Schedule.

Schedules

- Schedule: a sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed.
  - A schedule for a set of transactions must consist of all instructions of those transactions
  - Must preserve the order in which the instructions appear in each individual transaction.
- A transaction that successfully completes its execution will have a commit instructions as the last statement
  - By default transaction assumed to execute commit instruction as its last step

- A transaction that fails to successfully complete its execution will have an abort instruction as the last statement.

SERIAL SCHEDULE.

Let T_1 transfer 100 Rs from A to B, and T_2 transfer 10% of the balance from A to B.

Schedule 1

T_1	T_2
read (A)	
A: = A – 100	
write (A)	
read (B)	
B: = B + 100	
write (B)	
commit	
	read (A)
	temp := A * 0.1
	A := A – temp
	write (A)
	read (B)
	B := B + temp
	write (B)
	Commit

$S_1 < T_1\ T_2 >$

Serial schedule in which T_1 is followed by T_2:

Schedule 2

T_1	T_2
read (A)	
temp := A * 0.1	
A := A – temp	
write (A)	
read (B)	
B := B + temp	
write (B)	
Commit	
read (A)	
A: = A – 100	
write (A)	
read (B)	
B: = B + 100	
write (B)	
commit	

$S_2 < T_2\ T_1 >$

serial schedule where T_2 is followed by T_1

NON SERIAL SCHEDULE [Concurrent execution]

Schedule 3

T_1	T_2
read (A) A=2000	
A: = A – 100	
write (A) A=1900	
	read (A) A=1900
	temp := A * 0.1 temp=190
	A := A – temp
	write (A) A=1710
read (B) B=3000	
B: = B + 100	
write (B) B=3100	
commit	
	read (B) B=3100
	B := B + temp
	write (B) B=3290
	Commit

C_1

Expected value
A : 1710
+B : 3290
5000 Consistent

A: 2000
+B: 3000
5000

Schedule 4

T_1	T_2
A=2000 read (A)	
A: = A – 100	
	read (A) A=2000
	temp := A * 0.1 temp=200
	A := A – temp A=1800
	write (A) A=1800
A=1900 write (A)	
B=3000 read (B)	
B: = B + 100	
B=3100 write (B)	
commit	
	read (B) B=3100
	B := B + temp
	write (B) B=3300
	Commit

C_2

A : 1900
+B: 3300
5200 Inconsistent

5200

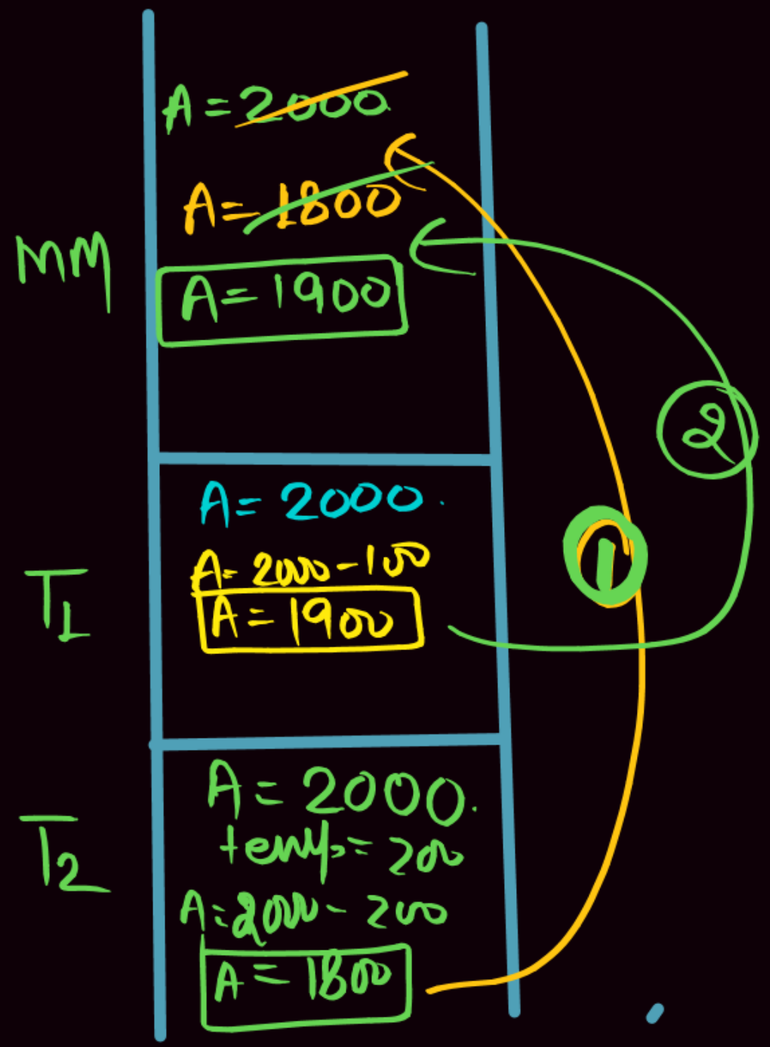
Note

C_1 (Non Serial (concurrent execution)

C_1 is Serializable, is equivalent to

Serial Schedule $S_1 <T_1, T_2>$ T_1 followed by T_2.

C_2 (Non serial Schedule C_2) is Not Serializable



DB

A = 2000

B = 3000

Note

Serial Schedule (ALL Serial Schedule (n!)) are always Consistent.

Note

Non Serial Schedule (Concurrent execution) may or may Not be Consistent.

But we execute a Schedule in Concurrent Execution.

WHY Concurrent Execution ?

- To Improve Utilization.
- Decrease Waiting time.
- Enhanced Throughput
- Effective Utilization of Resources.

Serial Schedule

- After Commit of one transaction, begins (Start) another transaction.
- Number of possible serial Schedules with 'n' transactions is "n!"
- The execution sequence of Serial Schedule always generates consistent result.

Example

S : $R_1(A)$ $W_1(A)$ Commit (T_1) $R_2(A)W_2(A)$ commit (T_2).

Advantage

- Serial Schedule always produce correct result (integrity guaranteed)

as no resource sharing.

Disadvantage

- Less degree of concurrency.
- Through put of system is low.
- It allows transactions to execute one after another.

Serializable Schedule:

If a Non Serial Schedule [Concurrent Execution] has been executed & that could have Same effect on a Database, As a schedule executed without Any Concurrent Execution [Serial Schedule] is called Serializable Schedule.

(Note) Serializable schedule are always Consistent.

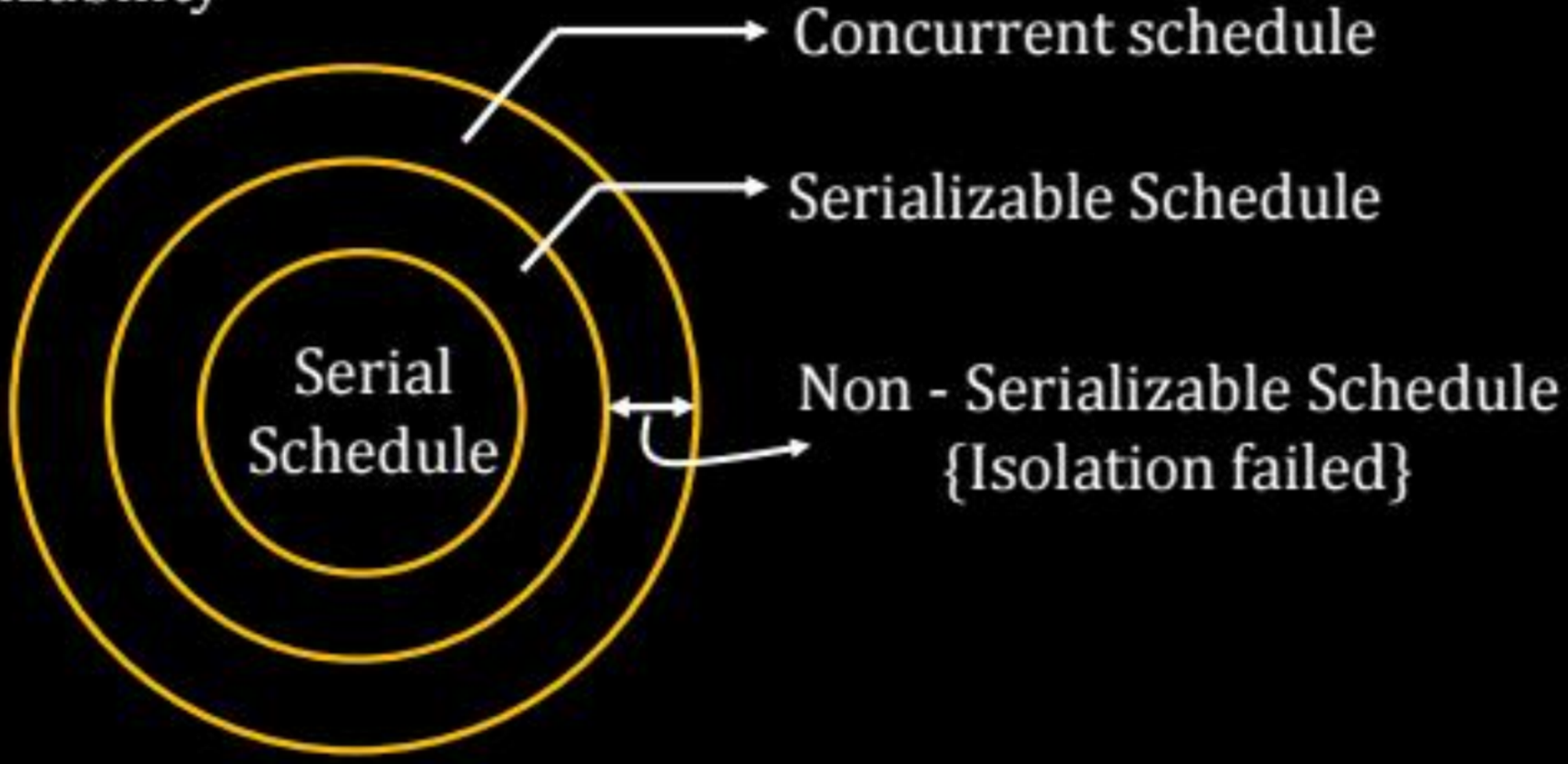
& this Process is Called serializability.

Serializable Schedule

A Schedule is serializable Schedule if it is equivalent to a Serial Schedule.

(i)Conflict Serializability

(ii)View Serializability



Serializability

- Basic Assumption: Each transaction preserves database consistency.
- Thus, serial execution of a set of transactions preserves database consistency.
- A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of:
  1. Conflict serializability
  2. view serializability

Serializable

① Conflict Serializable

② View Serializable.

Note A Schedule is Serializable if either it is Conflict Serializable (or) it is View Serializable (or) Both.

Conflict Serializable.

1. BASIC Concept.
2. Testing Method [Precedence Graph Method] (VImp)
3. Conflict Pair
4. Conflict Equivalent to Any serial Schedule.

Any Doubt ?

THANK YOU!