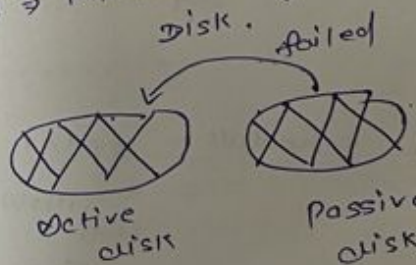
D: Durability:- [Recovery Management Component].
Transaction should be able to recover under any case of failure.

failure:- (1) Power failure
(2) S/W crash
(3) OS/DBMS kill Transaction } "log file" [stored in the disk]
(4) H/W crash
(5) Disk crash } → RAID Architecture:
→ [Redundant Array of Independent Disk].

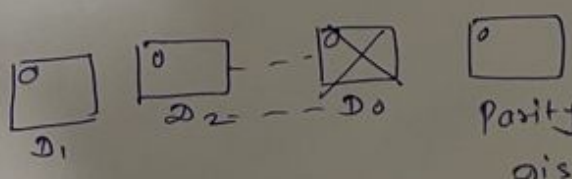
RAID 0 → No Redundant Disk.
For exa:- in PC's harddisk
→ High Risk
→ Not Durable:-

RAID 1 → Mirror Image of Independent Disk.



→ More costly → More time for write the data.
→ Low risk
→ Durability.

RAID 2:- It uses recovery Methods like Parity checker etc.



I: Isolation:- [concurrency control component]

→ Concurrent execution of two or more transaction result should not be inconsistent.

→ schedule:- Time order execution sequence of two or more transaction.

Ex:-

Clock cycle	T_1	T_2
1	$R_1(A)$	
2		$R_2(B)$
3	$W_1(A)$	
4		$R_2(A)$
5		$W_2(B)$
6	$R_1(B)$	

~~Sequence:- After commit / Rollback of current transaction allowed.~~

sequence:- $R_1(A)$ $R_2(B)$ $W_1(A)$ $R_2(A)$ $W_2(B)$ $R_1(B)$.

MCMP

Serial Schedule:- After commit / Rollback of current transaction allowed begin other transaction.

[Transaction executing one after other].

T_1: $R_1(A)$ $W_1(A)$ $R_1(B)$ $W_1(B)$

[500 from A to B]

T_2: $R_1(A)$ $R_2(B)$ display $(A+B)$.

[display total balance of A and B].

[two possibility for execution of transaction shown in the next page]

	T_1	T_2
1000	$R_1(A)$	
500	$W_1(A)$	
2000	$R_1(B)$	
2500	$W_1(B)$	
		$R_2(A)$ 1000
		$R_2(B)$ 2000
		display (A+B) 3000
1000	$R_1(A)$	
5000	$W_1(A)$	
2000	$R_1(B)$	
2500	$W_1(B)$	

$T_1 \rightarrow T_2$ (serial)

$T_2 \rightarrow T_1$ (serial)

* Advantages:- Every serial schedule result always consistent [Inconsistency never occurs if transaction executing serially]

* Disadvantages:- → Less degree of concurrency.
→ Less throughput
[# of transaction that are executed in unit time]
→ More Response time
→ less Resource utilization.

* Concurrent Schedule:- Transaction allowed to execute simultaneously (or) concurrently.

* [Every Serial schedule also called as Concurrent schedule]

A: ~~1000~~ 500
B: 2000

T_1	T_2
1000 $R_1(A)$	
500 $W_1(A)$	
	$R_2(A)$ 500
	$R_2(B)$ 2000
	display (A+B)
	⇓ 2500
2000 $R_1(B)$	wrong (error)
2500 $W_1(B)$	

(I)

[Non-Serial schedule]

Inconsistent data

A: ~~1000~~ 500
B: ~~2000~~ 2500

T_1	T_2
1000 $R_1(A)$	
500 $W_1(A)$	
	$R_2(A)$ 500
2000 $R_1(B)$	
2500 $W_1(B)$	
	$R_2(B)$ 2500
	dis(A+B) 3000

(II)

[Non serial schedule]

right data

[consistent data] schedule

allowed.

× restricted

done by Concurrency controller

⇒ compare (I) with serial $T_1 \rightarrow T_2$ ⇒

T_1 schedule of (I)

T_2 schedule of (I) is not equal × -

with serial $T_2 \rightarrow T_1$ = × also

So [(I) may Inconsistent schedule]

⇒ same for (II) compare with $T_1 \rightarrow T_2$ and $T_2 \rightarrow T_1$ Serial.

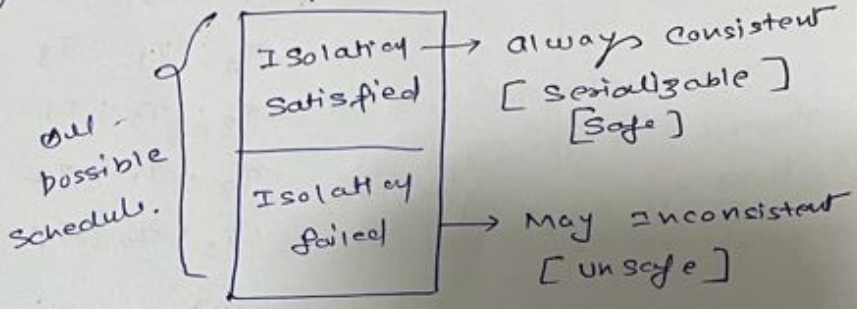
all are equal so [schedule is consistent schedule].

[This schedule is known as serializable schedule].

Isolation Definition:- Concurrent execution of two (or) more transaction result must be equal to result of any serial schedule.

* If any concurrent schedule follow Isolation (or) equal to any serial schedule, known as serializable schedule (or)

Concurrent execution of Transaction must be equal to any serial.



** Goal of concurrency control component:- "Concurrency control component should not allow to execute any schedule if schedule is fail Isolation rule".

C: Consistency:- [It is user responsibility]

⇒ DB operations requested by User (Transaction) must be logically correct.

⇒ Transaction written by the user must be logically correct if it is wrong they o/p is wrong.

⇒ Atomicity : [Recovery mgmt Component]
⇒ Durability : [" "] } DBMS responsible.
⇒ Isolation : [Concurrency Controller]
⇒ Consistency : [User] [DBA] } DBA responsible

Q:- $T_1\ T_2 - - - T_n$: n Transactions :-

How many possible serial schedules possibles :-

Ans :- L_n or n! possible serial Schedule :-

Ex: $(T_1\ T_2\ T_3) \Rightarrow$ 3! serial schedule

$T_1 : T_2 : T_3$

$T_1 : T_3 : T_2$

$T_2 : T_1 : T_3$

$T_2 : T_3 : T_1$

$T_3 : T_1 : T_2$

$T_3 : T_2 : T_1$

$\Rightarrow$ $T_1 : R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)$

$T_2 : R_2(A)\ R_2(B)$ How many concurrent schedule possible ?

Ans $\rightarrow$ $\frac{6!}{2!\,4!} = {}^6C_4 \Rightarrow {}^6C_2 = 15.$

$\Rightarrow$ All concurrent schedules : 15 [Serial & Non-serial]

$\Rightarrow$ # of Non serial schedules = { All Concurrent schedules } - { serial schedules }

$= 15 - 2! = 13.$