Schedule In (DBMS)

It is a chronological execution sequence of multiple transaction.

types

- Serial
- Parallel

Serial → one by one

like T_1, T_2, T_3

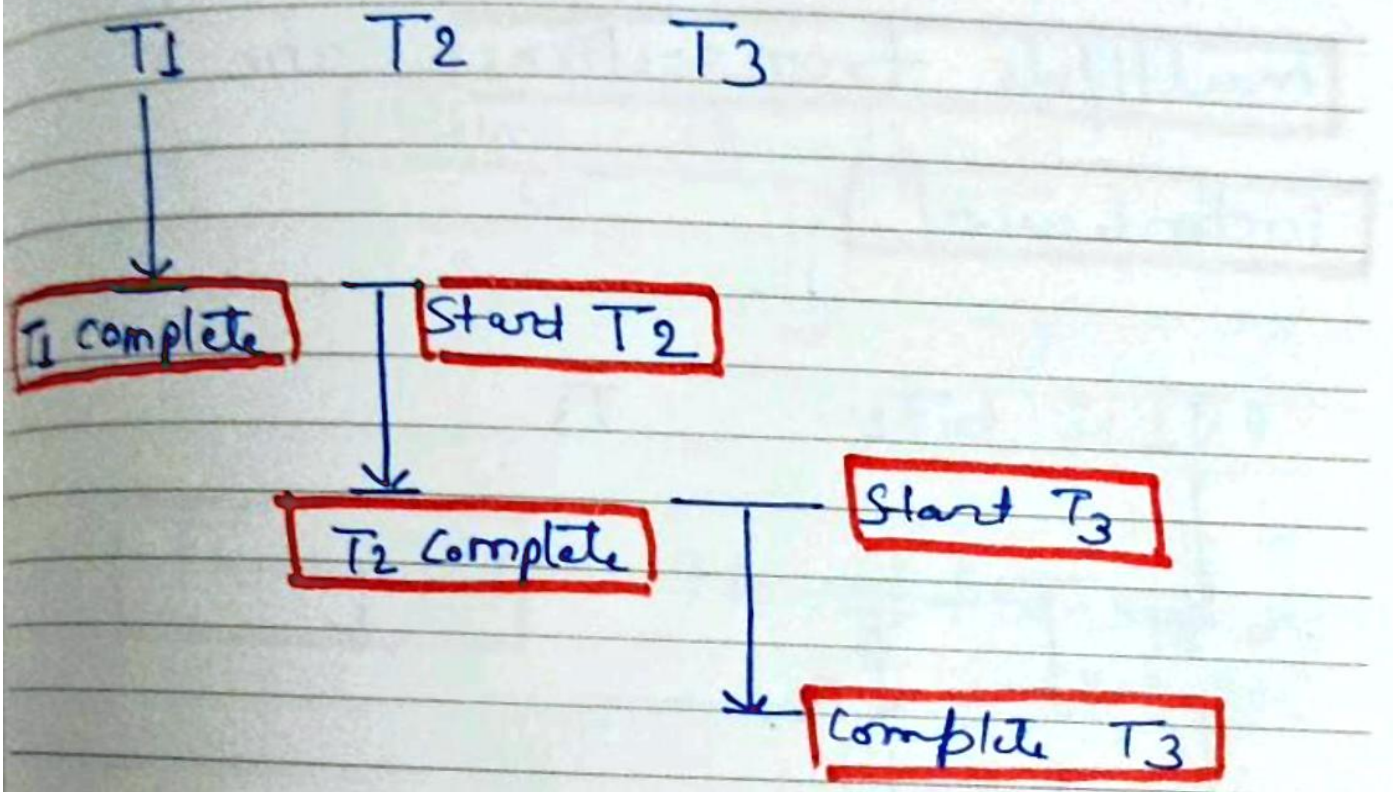
start → first T_1 is start then

Remarks

until it is not completed T_2 can't start. After

T1 execute completely

then and then T2 will start.



Advantage → Consistency, Secure

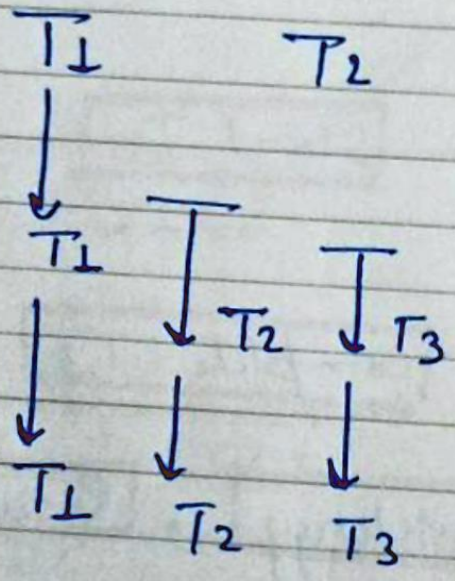
Problem → waiting time

Remarks: Real life example → ATM life there are 3 people in a queue after first then second will start.

Parallel Schedule

⇓

where the **operations** of **multiple transactions** are **interleaved**.



multiple transaction **executed** at the **same** time.

Remarks

Real life ex: **SBI online**

Advantage — Performance

Increases ← How ??

↓

Using Throughput

⇓

what is ??

Throughput = No. of transactions executed / time.

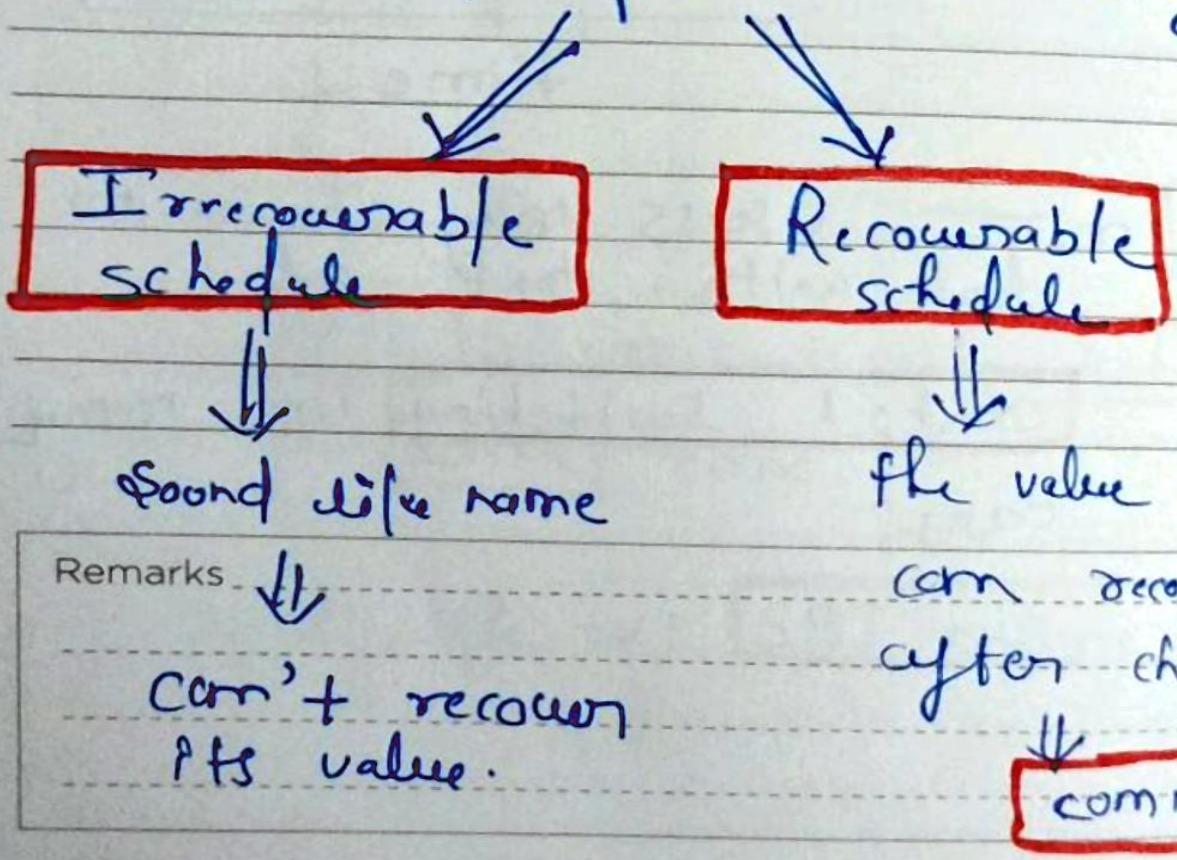
Problem — users may have to deal with tons of context switching in many cases.

Remarks

Context Switching:

Storing the **state of process** so that it can be **reloaded** when **required** and **execution** can be resumed from the **same point** as earlier.

Inside parallel scheduling



Remarks

Can't recover its value.

Irrecoverable issue

↳ explain from example -

T1	T2
10 R(A)	
5 A=(A-5)	
5 W(A)	→ R(A) 5
	A = A-2 = 3
	W(A) 3
	commit
	↳ save changes.
R(B)	↓
*	A=3
fail	

given
A = 10
B = 20

↳ transaction fail at point B, then due to atomicity property it will do rollback

↓

undo

Remarks

Again value of A
||
10

Date__/__/_____ Note

The changes done by T2 is lost and can't be recoverable.

Remarks .....

.....

.....

.....