

---

---

---

---

---




---

---

---

---

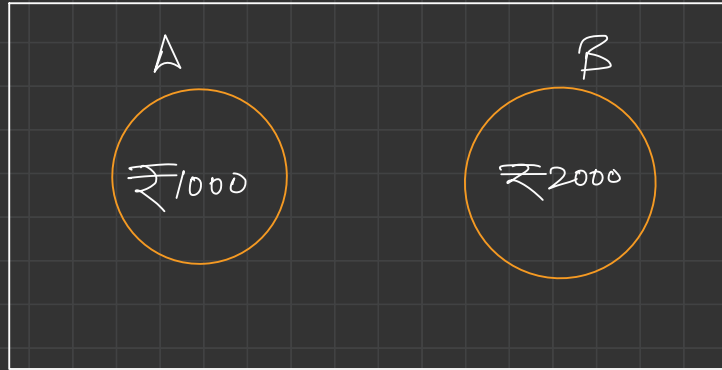
---



# Lec-12

Transaction 1

ABC Bank



System ₹3000

Task 1 → ₹50

from Bank A/c A to A/c B

T<sub>1</sub>:

```
read(A)
A := A - 50;
write(A);
read(B)
B := B + 50;
write(B)
```

6 steps

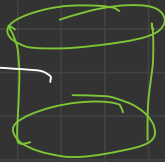
Atomic →  
they are considered  
to be a single Task.

# ACID Properties

read(A)  
↓

1000

Buffer



write(A);

950

Buffer



DB

Commit

950

↓  
main mem

$T_i$ : read(A)  
 $A := A - 50$ ;  
write(A);  
read(B);  
 $B := B + 50$ ;  
write(B);

① Atomicity

→  
A  
read 1000  
 $950 = 1000 - 50$   
[950] write(A)

950.

B  
read 2000

read 2000  
 $2000 + 50$   
write(2050)  
2050

→ transaction → successful.  
→ fail → old state recovery.

DB → maintain → old state.

↓ intermediate state.

↓  
Roll back success X,  
old state

② Consistency

③ Isolation 1-

Banking system

$T_1$   $T_2$   $T_3$   $T_4$

$t=0 \Rightarrow T_1$  (Google Pay)  $t=10 = T_2$  (Netbanking)

$\text{read}(A) \rightarrow 1000 \longrightarrow \text{read}_1(A) \rightarrow 1000$

$A := A - 50 = 950$

$A = 950$   
 $\text{write}(950)$

$1000 - 50 = 950$

$B \longrightarrow$

$B \rightarrow 50 + 50 = 100$

④ Durability →

$T_i \rightarrow \text{Success}$



all 6 step → Success



DB updates permanent (persistent)



even if there is a system failure after  
T completely  
execute.

→ recovery - mgmt Component.

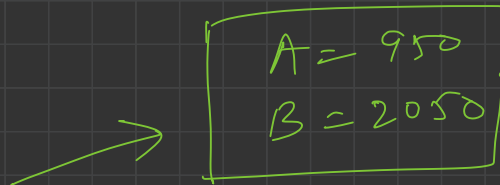
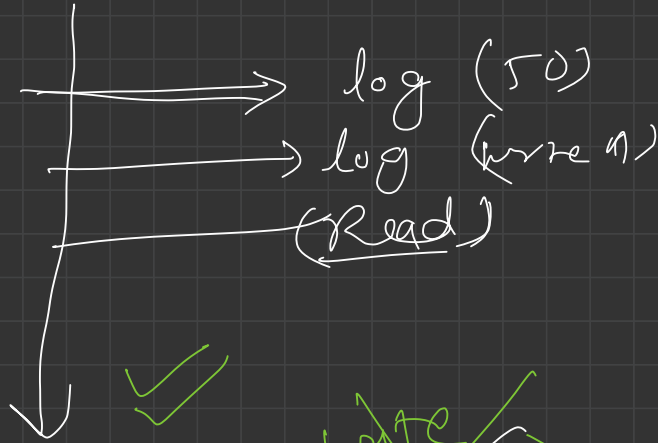
read(A)  
 $A := A - 50$

write(A)

Read(B)

$B := B + 50$

write B,



main mem



system crash



\* States of Transition (life cycle)

How to Implement Atomicity?

Lec - 13