Transactions - 2

TABLE OF CONTENTS

Notes

*   31st Hard day :

     - A quick revision of session

     - Assignment   (MCQ)

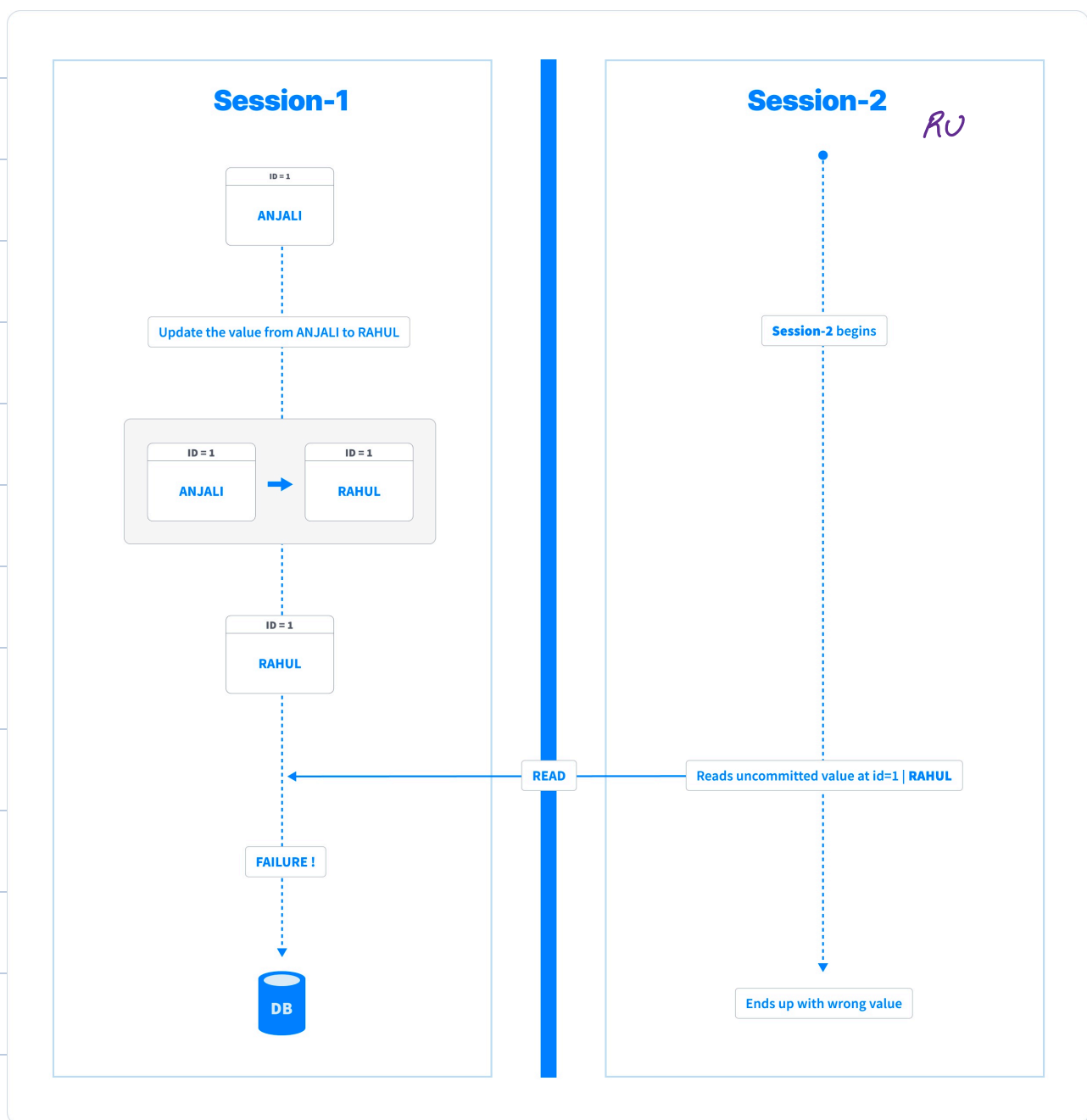   - Revision / backlog

Isolation Levels

Isolation have different levels (4)

1. **Read Uncommitted**

2. **Read Committed**

3. **Repeatable Read**

4. **Serialisable**
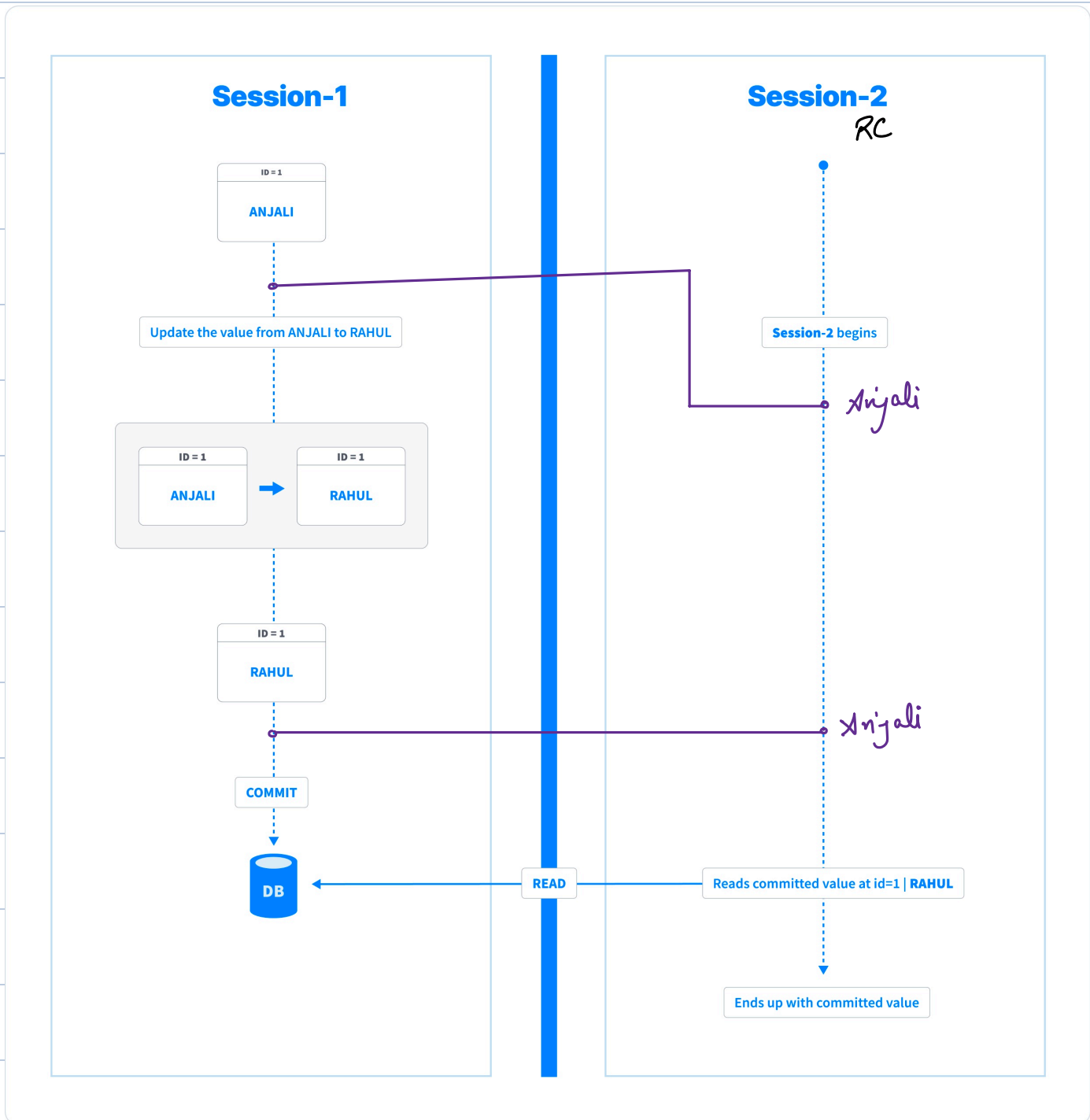
1. Read Uncommitted

- It allows a transaction to read uncommitted data from other transaction running

  at the same time.

- Reads the latest data. (Committed / Uncommitted)

2. Read Committed

- It ensures that we read only committed data. (Latest committed data)

Session-1

ID = 1

ANJALI

Update the value from ANJALI to RAHUL

ID = 1		ID = 1
ANJALI	➡	**RAHUL**

ID = 1

RAHUL

COMMIT

DB

Session-2

RC

Session-2 begins

Anjali

Anjali

READ — Reads committed value at id=1 | **RAHUL**

Ends up with committed value

* **Pros :**

      – We don't have dirty reads.

* **Cons :**

      – Might not be efficient

      – Non-Repeatable Read

Users

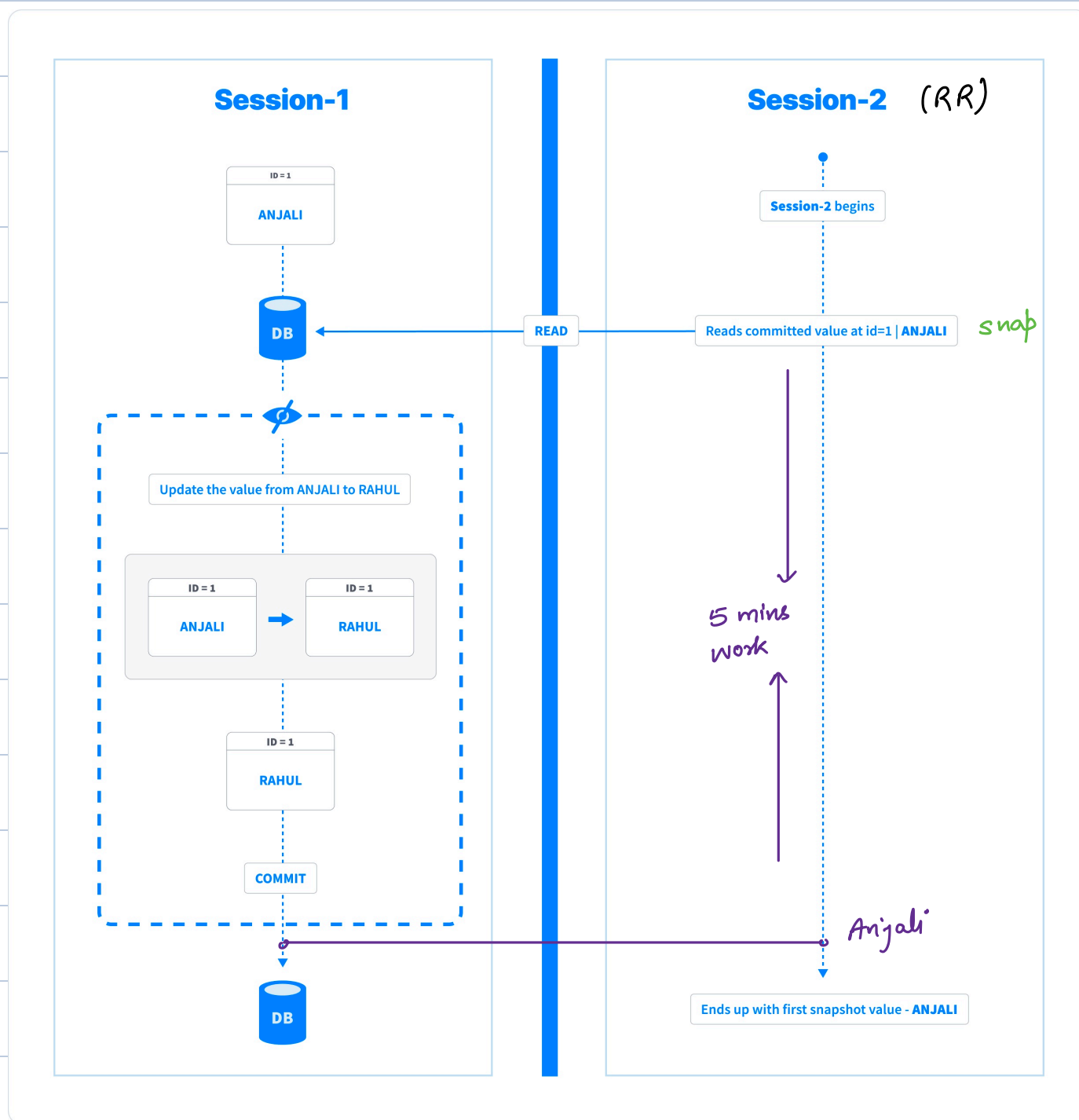| id	name	psp	mail_sent
1	A	75	~~F~~ T
2	B	85	F
3	C	~~84~~ 79	~~F~~ T
4	D	98	F

- Send mail to learners psp < 80

Steps :

1. Get data of users having psp < 80   (A)

2. Send e-mails.   (5 mins)   (A)

3. Update email_sent col^m. (if psp < 80 → T) (A, C)

\# When a transaction reads the same data of same row with different value → **Non-Repeatable Read**.

3. Repeatable Reads (Default in MySQL)

- For first read in a transaction it reads latest committed value of current row.

- After that in the same transaction it will read the same value.

Session-1

ID = 1

ANJALI

DB

Update the value from ANJALI to RAHUL

ID = 1 — ANJALI → ID = 1 — RAHUL

ID = 1

RAHUL

COMMIT

DB

Session-2 (RR)

Session-2 begins

READ ← Reads committed value at id=1 | ANJALI   *snap*

5 mins work

Anjali

Ends up with first snapshot value - ANJALI

Advantage :
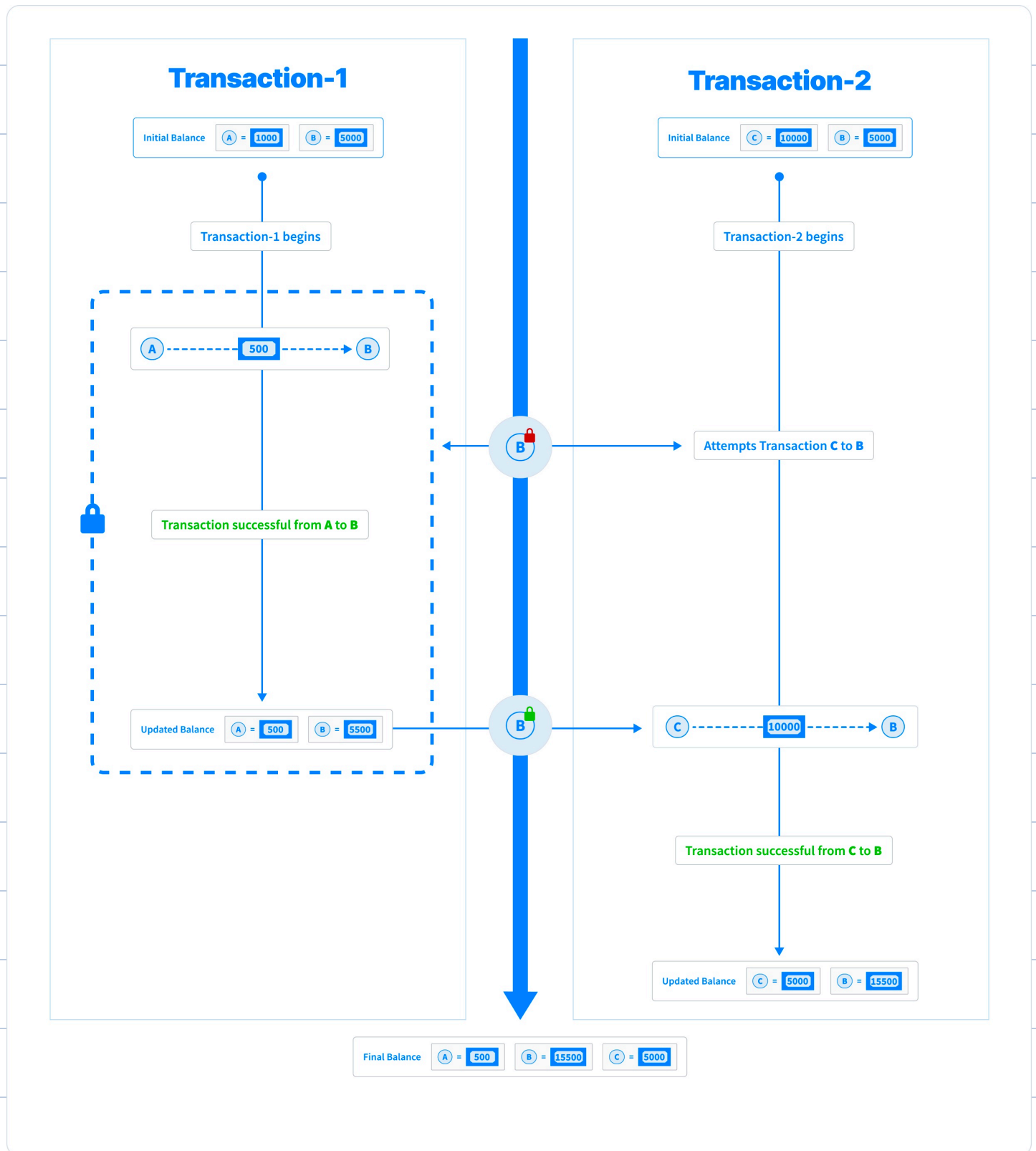
- We don't have non-repeatable read issue here.

Disadvantage :

→ 'Phantom Read'

* If a new entry is added in the table during the transaction & we perform some update operation on this new entry then this row magically gets read resulting in Phantom Read.

4. Serializable

Transaction-1

Initial Balance | A = 1000 | B = 5000

Transaction-1 begins

A ----- 500 -----> B

Transaction successful from **A** to **B**

Updated Balance | A = 500 | B = 5500

Transaction-2

Initial Balance | C = 10000 | B = 5000

Transaction-2 begins

Attempts Transaction **C** to **B**

C -------- 10000 --------> B

Transaction successful from **C** to **B**

Updated Balance | C = 5000 | B = 15500

Final Balance | A = 500 | B = 15500 | C = 5000

A ⟶ B	C ⟶ B
bank_transfer (A, B, 500) {	bank_transfer (C, B, 10000) {
A ⟶ X (1000)	C ⟶ X (15,000)
If (X >= 500)	If (X >= 10000)
A ⟵ X - 500	C ⟵ X - 10000
B ⟶ X (5000) → lock 🔒	B ⟶ X (5500)
B ⟵ 5000 + 500	B ⟵ 5500 + 10000
B ⟵ 5500	B ⟵ 15500
Commit;	Commit;

	Initial	final
A	1000	500
B	5000	15500
C	15000	5000
	21,000 ◡	21,000

Deadlock

DEADLOCK

Wait

Boy → Girl

Girl

Wait

(A,B) T1   Wait for B   T2 (B,A)

A   🔒 lock   B   🔒 lock

B   A