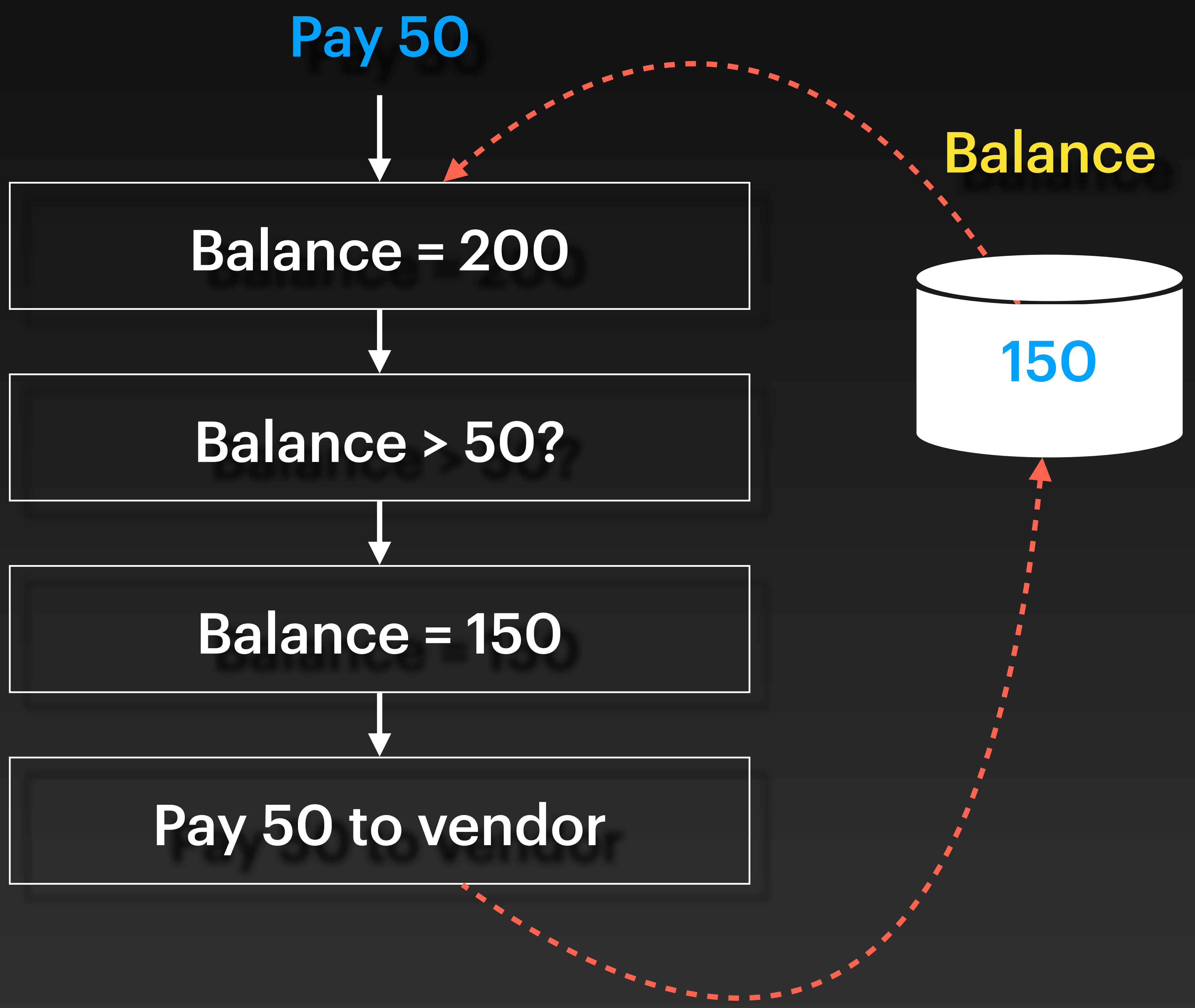
TECH VAULT

TRANSACTIONS ACID Properties

AMR ELHELW

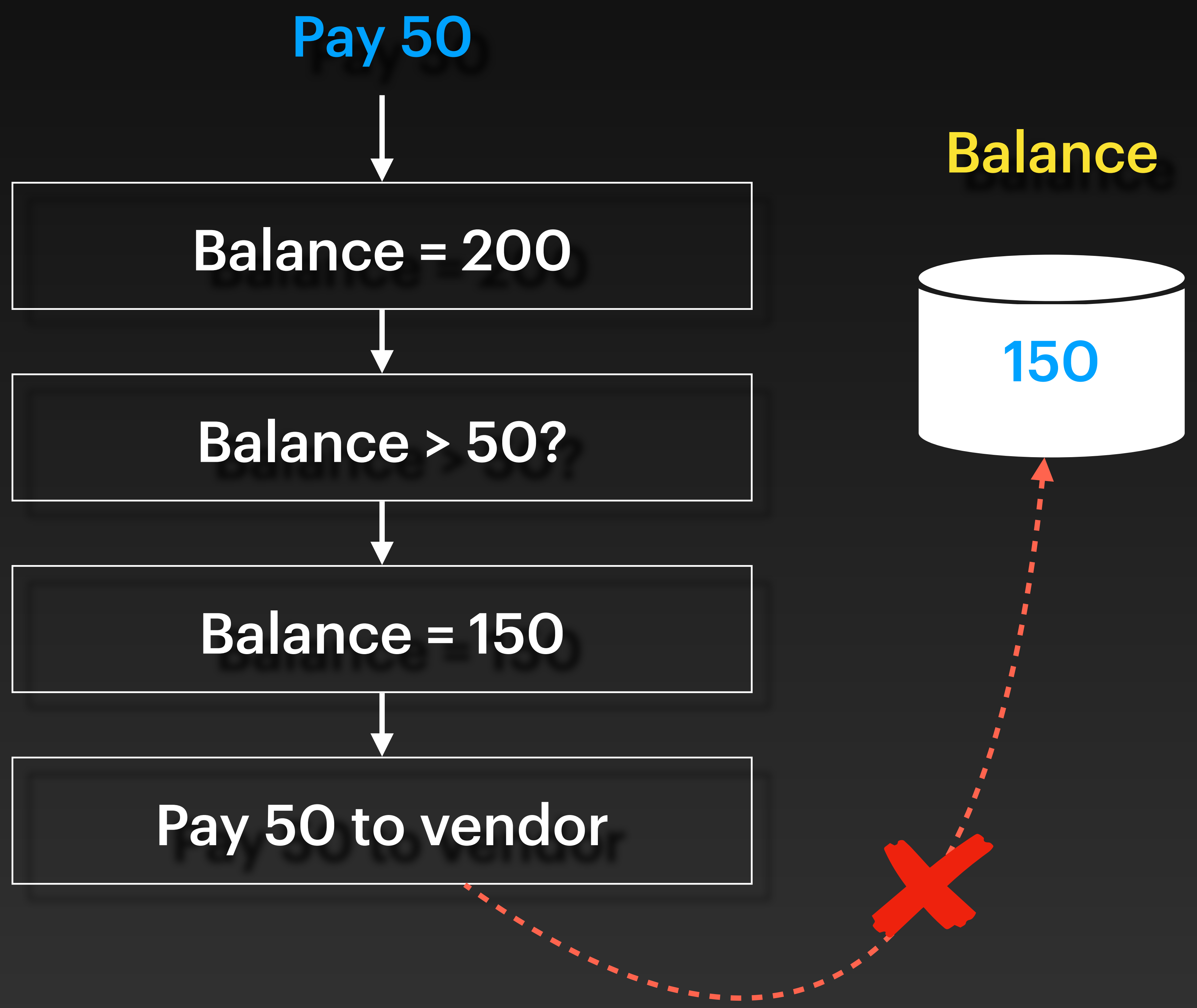
Pay (amount)

```
Read (balance)
Check (balance > amount)
balance = balance - amount
Pay amount to vendor
Write (balance)
```



Pay (amount)

```
Read (balance)
Check (balance > amount)
balance = balance - amount
Pay amount to vendor
Write (balance)
```



Transaction

- A sequence of Read, Write operations on DB objects

- In SQL (DBMS-specific):
  - Usually starts with BEGIN
  - Ends with COMMIT or ROLLBACK

```sql
START TRANSACTION;

SET @source_balance = (SELECT balance FROM accounts WHERE account_id = 1);
SET @transfer_amount = 100.00;

IF @source_balance < @transfer_amount THEN
    ROLLBACK;
ELSE
    UPDATE accounts SET balance = balance - @transfer_amount WHERE account_id = 1;
    UPDATE accounts SET balance = balance + @transfer_amount WHERE account_id = 2;
    COMMIT;
END IF;
```

READ

WRITE

WRITE

MySQL Example

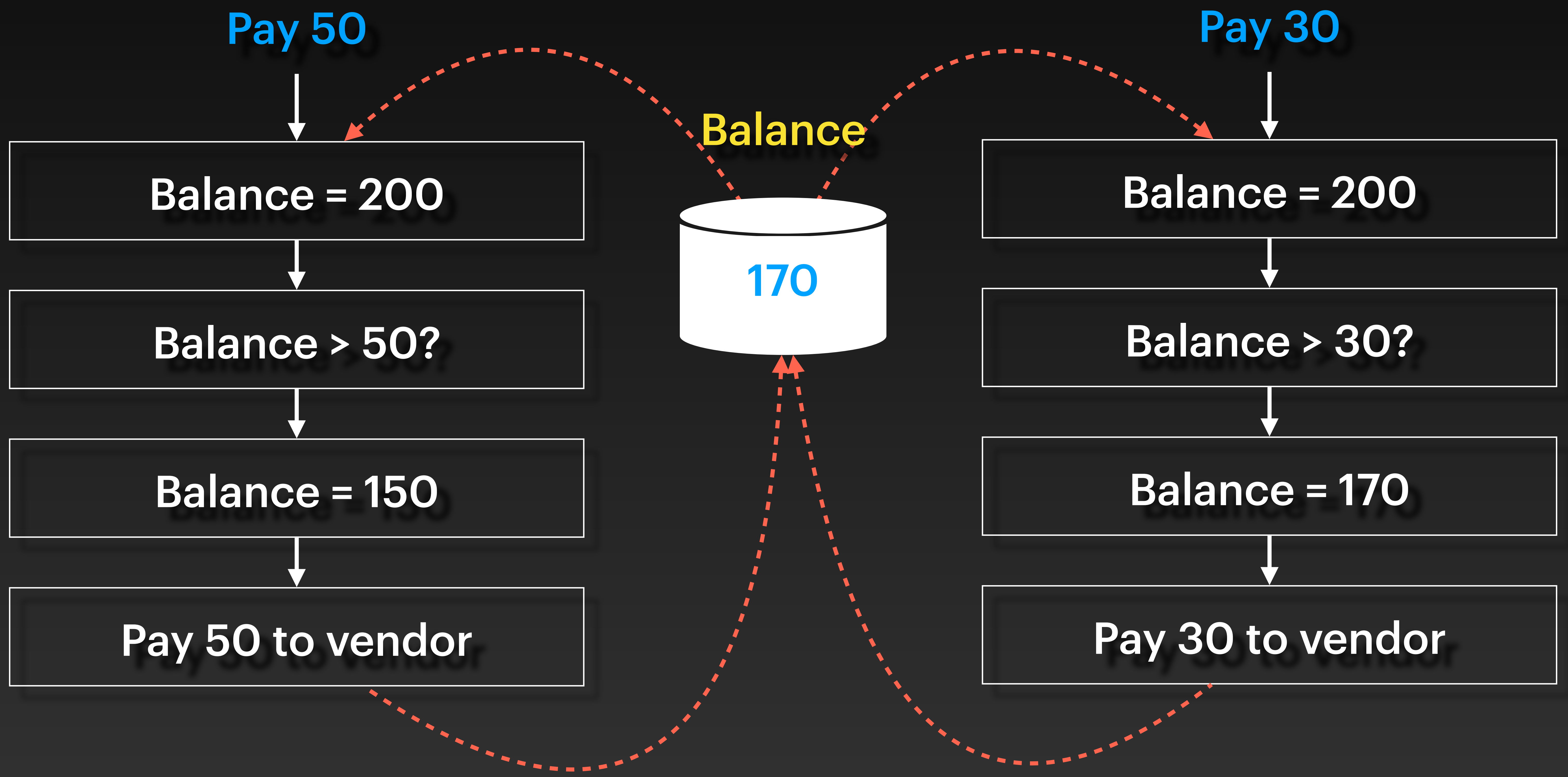
```sql
DO $$
DECLARE
    source_balance DECIMAL(10, 2);
    transfer_amount DECIMAL(10, 2) := 100.00;
BEGIN
    SELECT balance INTO source_balance
    FROM accounts
    WHERE account_id = 1;

    IF source_balance < transfer_amount THEN
        RAISE EXCEPTION 'Insufficient funds';
    ELSE
        UPDATE accounts SET balance = balance - transfer_amount WHERE account_id = 1;
        UPDATE accounts SET balance = balance + transfer_amount WHERE account_id = 2;
        COMMIT;
    END IF;
END $$;
```

PostgreSQL Example

Pay (amount)

```
Read (balance)
Check (balance > amount)
balance = balance - amount
Pay amount to vendor
Write (balance)
```



ACID Properties

A Atomicity

Each transaction is "all or nothing"

C Consistency

Data should be valid according to all defined rules

I Isolation

Transactions do not affect each other

D Durability

Committed data would not be lost, even after power failure.

BEGIN	
Read (A)	500
A = A - 100	
Write (A)	400
Read (B)	200
B = B + 100	
Write (B)	300
COMMIT	

Before:

A = 500

B = 200

A = 400

B = 300

```
BEGIN
Read (A)        500
A = A - 100
Write (A)       400
Read (B)        200
B = B + 100
Write (B)
COMMIT
```

Before:

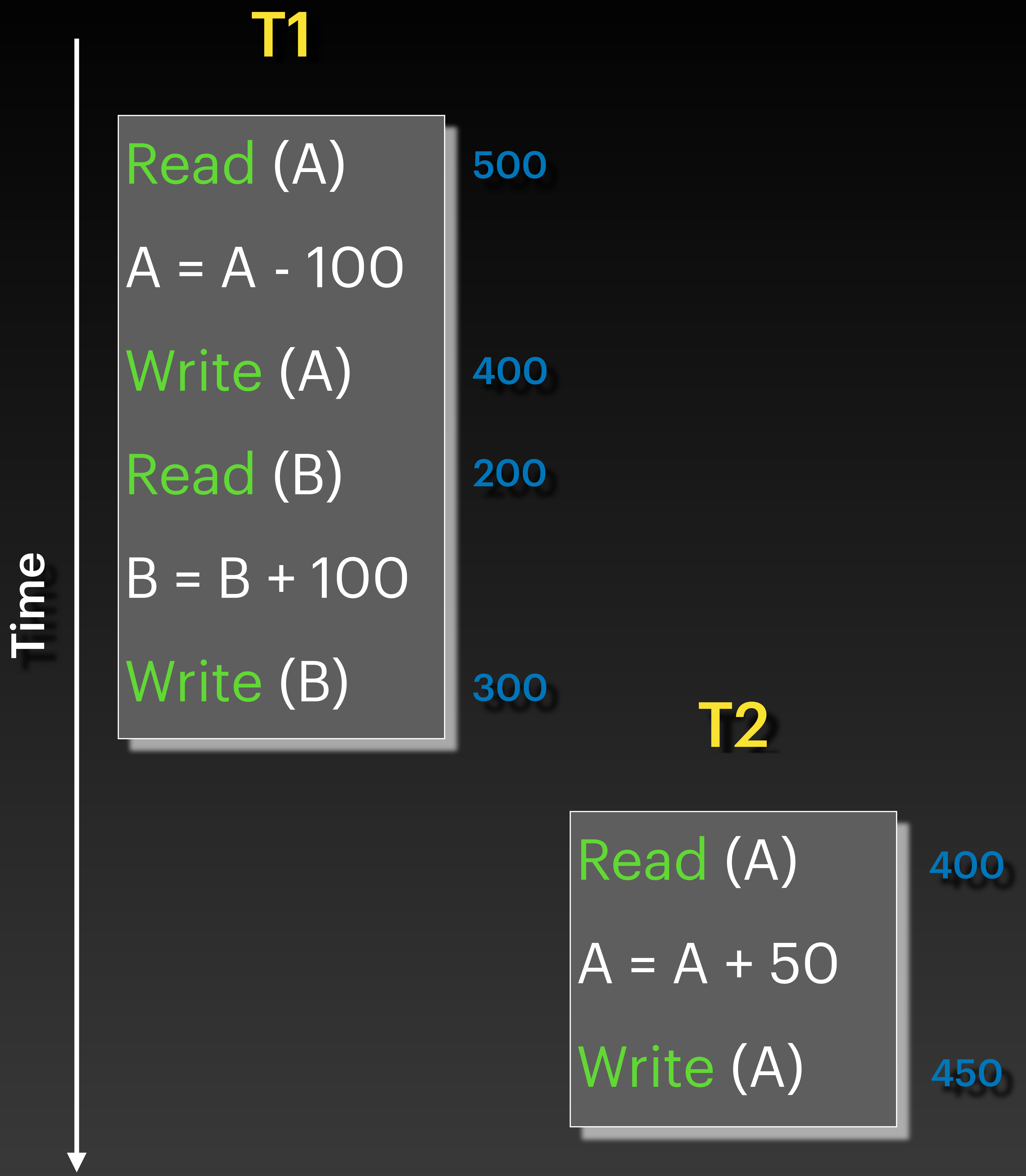
A = 500

B = 200

A = 400

B = 200

✗



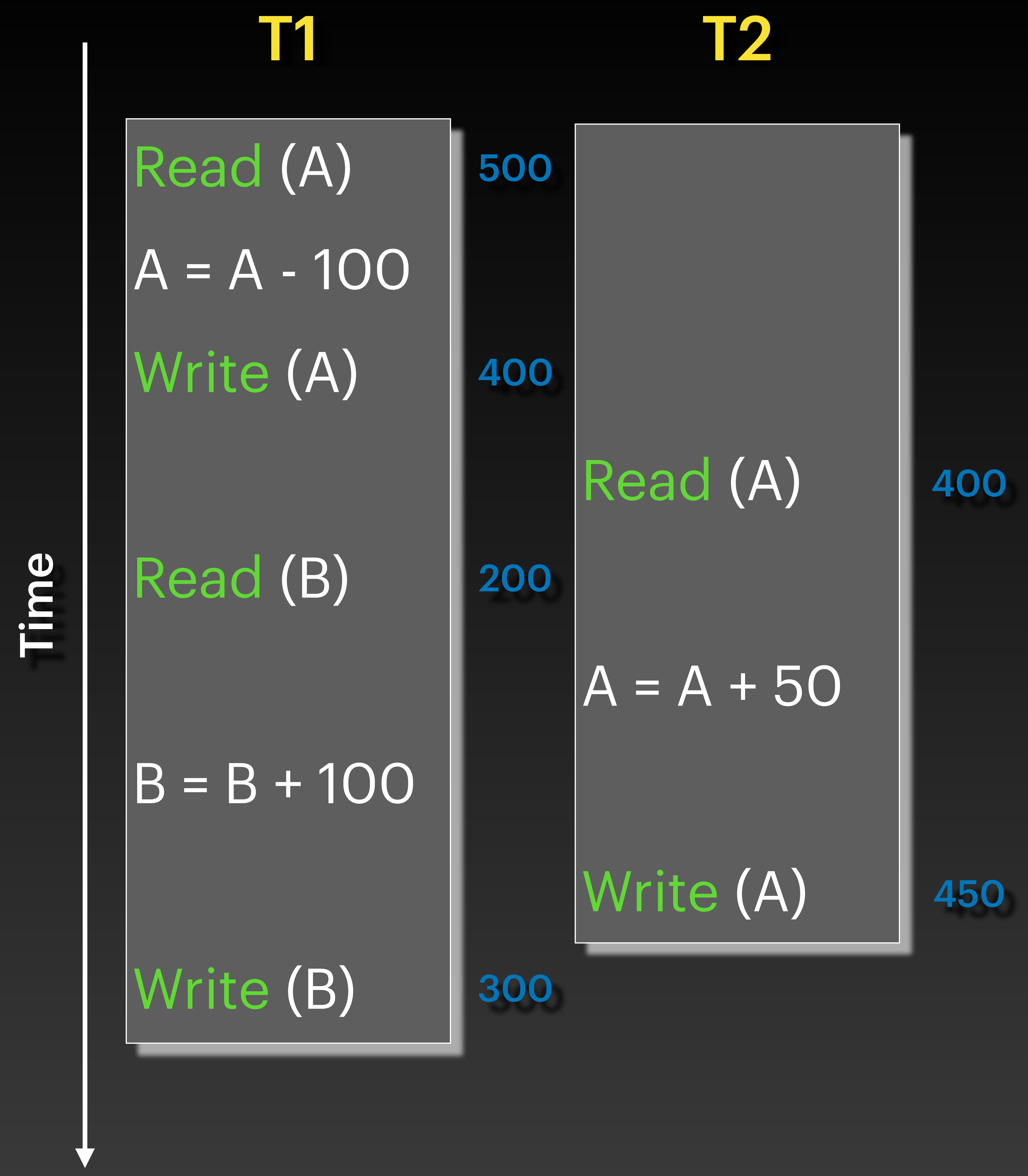
T1

```
Read (A)
A = A - 100
Write (A)
Read (B)
B = B + 100
Write (B)
```

T2

```
Read (A)
A = A + 50
Write (A)
```

Before:	After:
A = 500	A = 450
B = 200	B = 300



Before:	After:
A = 500	A = 450
B = 200	B = 300

Time

T1		T2	
Read (A)	500		
A = A - 100			
		Read (A)	500
		A = A + 50	
Write (A)	400		
Read (B)	200		
		Write (A)	550
B = B + 100			
Write (B)	300		

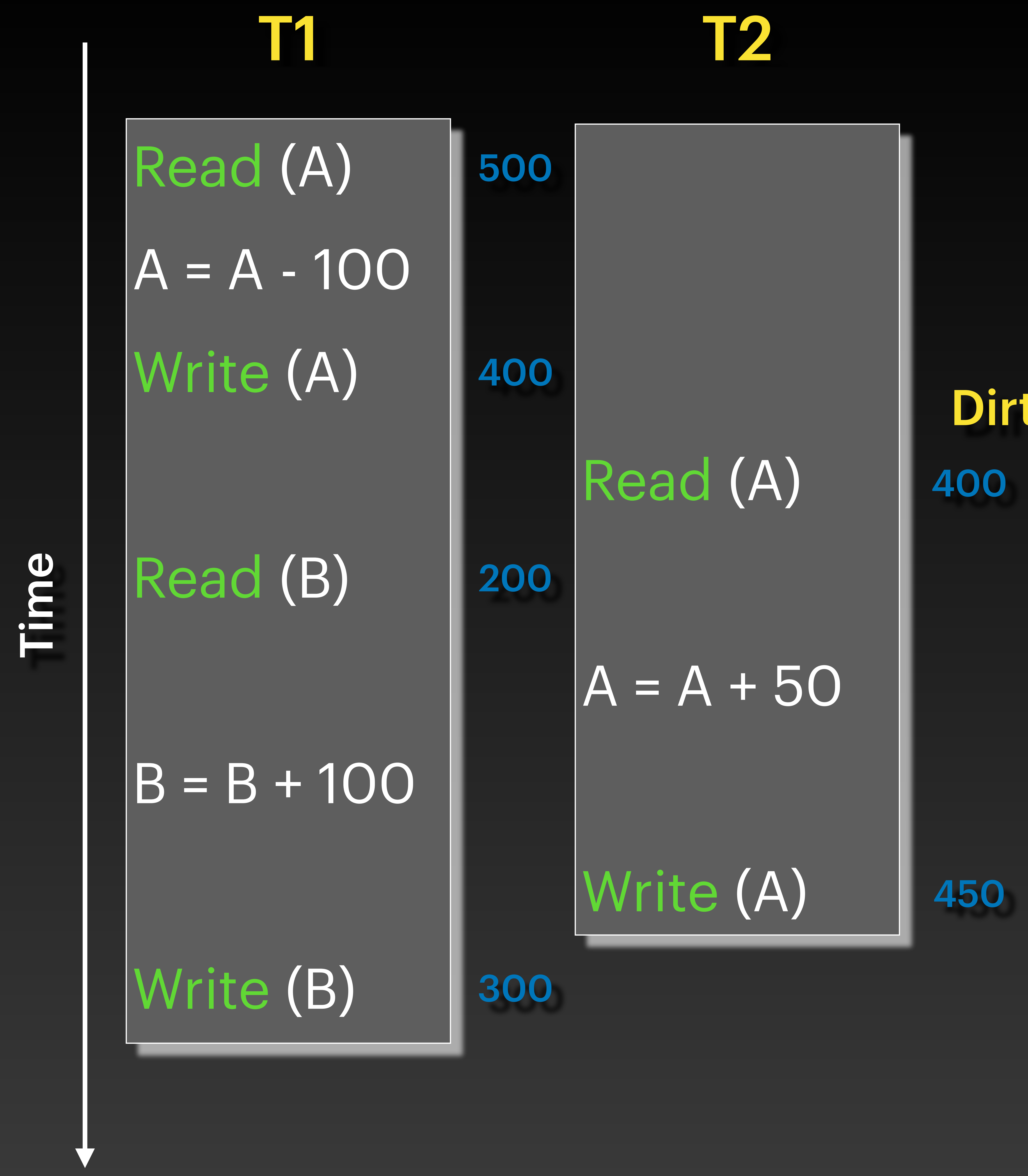
Before:	After:
A = 500	A = 550
B = 200	B = 300

Lost Update

Common Problems with Concurrent Transactions

- Lost Update
- Dirty Read
- Non-repeatable Read

Dirty Read



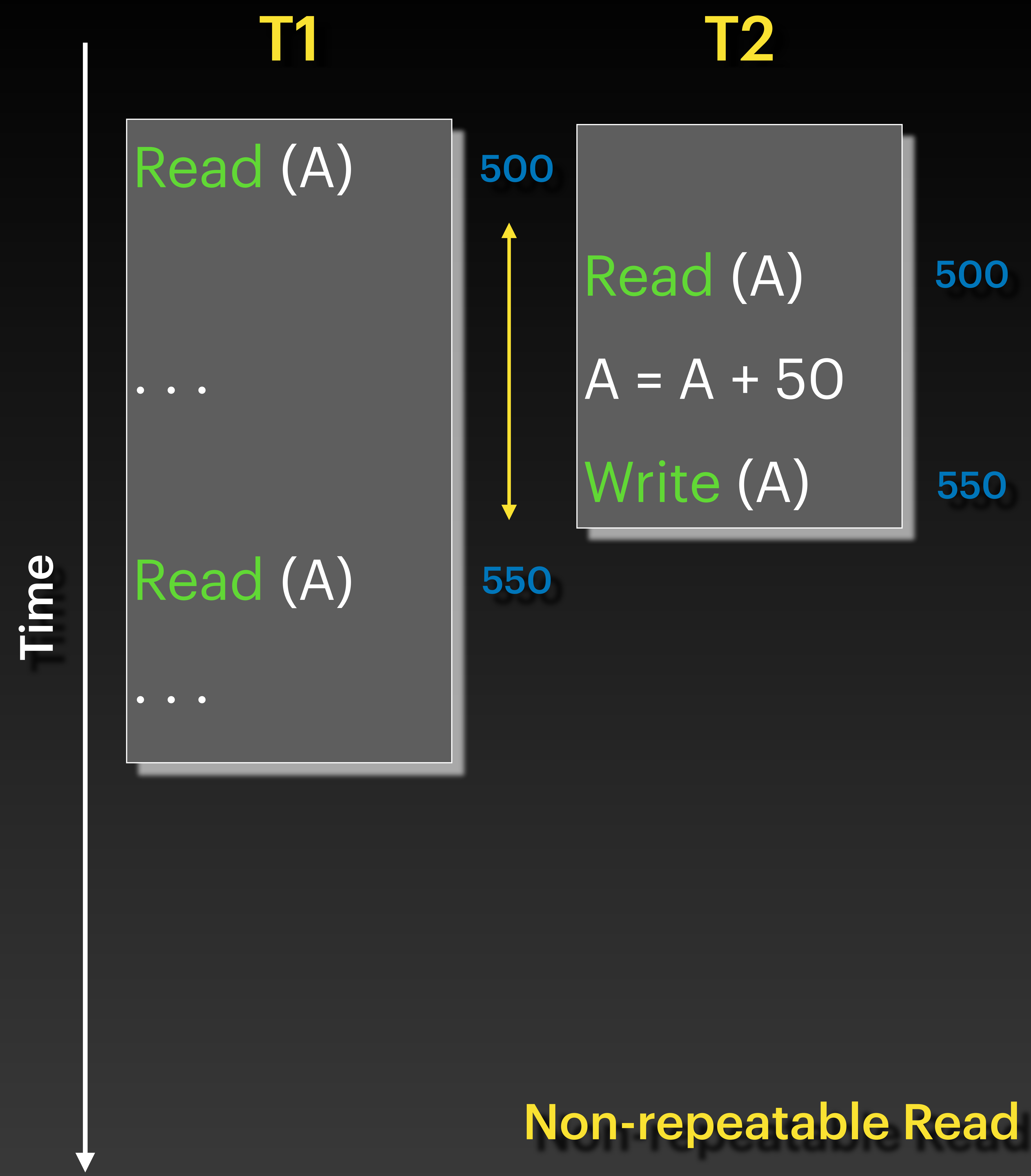
Before:	After:
A = 500	A = 450
B = 200	B = 300

Dirty Read

Time

T1		T2	
Read (A)	500		
A = A - 100			
Write (A)	400		
		Read (A)	400
~~Read (B)~~			
		A = A + 50	
~~B = B + 100~~			
		Write (A)	450
~~Write (B)~~			

Before:	After:
A = 500	A = 450
B = 200	B = 200



Non-repeatable Read

Isolation Levels

- Read Uncommitted
  - No isolation - best throughput
  - Any transaction can read (uncommitted) data written by other transactions.
- Read Committed
  - Each operation in a transaction can only read committed data (at the time of that operation).
  - Multiple read operations may still read different values
- Repeatable Read
  - All reads within the transaction only read committed data at the beginning of the transaction
- Serializable
  - Highest isolation level - worst throughput
  - Concurrently executing transactions appears to be executing serially.