TRANSACTIONS

1.) Transactions - what / why
2.) ACID Properties
3.) Commits / Rollbacks
4.) Isolation levels
    - Read uncommitted

start by 9:07 PM

*) TRANSACTIONS:

Hdfc

$$\begin{tikzcd} \boxed{A} \arrow[r, "500"] & \boxed{B} \\ 1000 & 5000 \end{tikzcd}$$

accounts

id	Name	balance
1	yash	5000
2	ANan	2000

operations:

- S1.) check balance of (A)
- S2.) debit from A
- S3.) credit to B

S1 → Read (select)
S2 → write (update)
S3 → write (update)

```
Send Money (A, B, amount) {
    S1
    S2
    S3
}
```

S2 → update balance

B = 2000
b → 200 Rs to A

B = B - 200

(1) xxx

- Read current value of B
- updates current value
- saves

EXAMPLE:

total = 1000 + 5000 + 15,000
= 21k

A	B	C
₹ 1000	₹ 5000	₹ 15000

$A \xrightarrow[(t_1)]{₹ 500} B \xleftarrow[(t_2)]{₹ 10,000} C$

```
transferMoney(x, y, amount){
S1    Read → x
      if (x >= amount)
✓ S2    = write(x) = x - amount
        = Read → (y)
S3      write(y) = y + amount   // update query
}
```

↓	↓
→ 1000	→ 15,000
↓	↓
• 1000 - 500	• 15k - 10k
↓	↓
★ 5000	5000
	↓
→ (t₁) = 500 + 5000	5000 + 10,000
✓ (B) = 5500	(B) = 15,000

B = $\underline{15,000}$

A = 500 = 20,500

B = 5000

xxxx

usecase of transactions.

1.) balance lost
2.) inconsistency $\}$

TRANSACTIONS:

set of d/b queries logically grouped together

```
start transaction;

    S1

    S2

    S3

COMMIT;  OR  Rollback;
```

syntax.

*) ACID PROPERTIES:
(features of transaction)

A	Atomic
C	consistency
I	Isolation
D	durability

1.) Atomic:

↳ ATOM → single unit

transaction is either completed fully OR
Not completed.

→ should Never End in intermediate state xxx

2.) consistency:

exactness / correctness.

3.) Isolation:

one transaction should not affect
other transaction ON same data.

4.) Durability:

Persistence →

once the transaction is completed → it should stay.

*) Rollback and commit:

> start transaction; ✓

s1 ✓

s2 ✓

s3 ✓
....

commit;

Rollback: ↓↓↓

start transaction;

s1

s2

s3

⋮

× × ×

where id:-
199;
Rollback;

*) TRANSACTION ISOLATION LEVELS:

Justify – what you as a user will view data from SQL.

4 levels of isolation:

1.) Read uncommitted	– Most lenient
2.) Read committed	PSQL – default
3.) Repeatable Read	MySQL – default
4.) Serializable	Most strict

1.) READ UNCOMMITTED:

allows to Read the latest data even if that is uncommitted. / committed.

S1 (session)	S2 (sessions)	S3 (Read UNC)
start trans;	start trans;	Read (a)
set a = 10	set a = 11	11
set a=10;	set a = 11 ; commit;	Read (a) 11

prev (a) = 10;

session(1) Repeatable Read

S1 set a = 11;

S3 Rollback;

session(2) (Read uncomm.)

S2 Read (11) 11

S4 Read (a) 10

Note:

Isolation levels will Read data $\longrightarrow$ your isolation level

Others isolation level doesn't matter

Advantages:

latest data ⟶ very fast }
very lenient }

*) PROBLEMS:

A = 2000
B = 2000

bank transaction ⟶ ??

def

[Repeatable Read]		[Read UNCOMM.]	
● (T_1)	10 ₹	(T_2)	100 ₹
$(A \to B)$		$(B \to A)$	
① Read (A)	2000.	④ Read (B) → x	2000
② x = x − A	1990	⑤ x = x − 100	1900
③ write(x) → A	1990	⑥ write (x) → B	1900
	↓↓ 1990	↓↓↓	
⑦ → Read (B) → x	2000	⑧ Read(A)	1990
x = x + 10	2010	⑨ x = x + 100	→ 2090
write(B) → x	2010	⑩ write (A) → x	→ 2090

we Read Latest value, which was NOT commited / updated.

sql ⟶ DIRTY READ

DIRTY Reads: Reading uncomm. data from table ⟶

***