Transactions

[Talk - 5 min]

Agenda

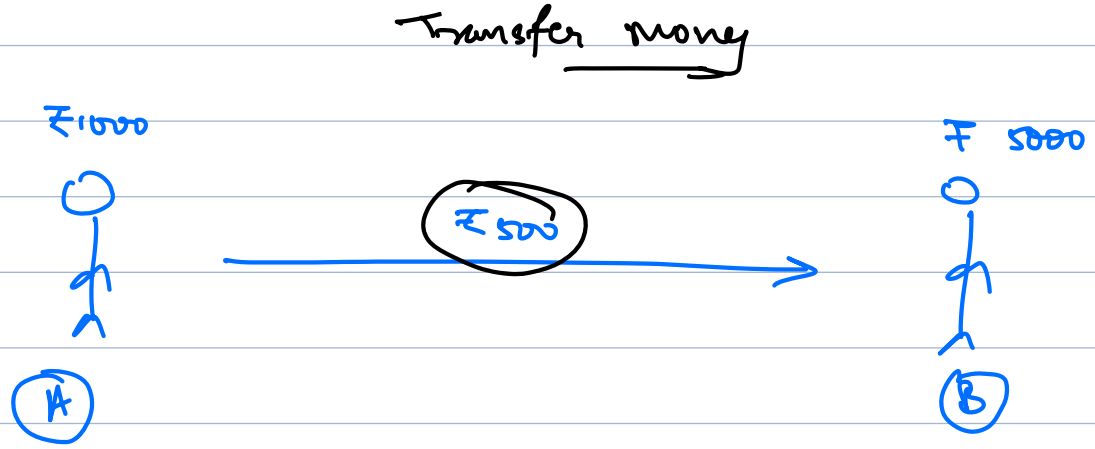
1. Transactions
2. Properties of transactions
   - A
   - C
   - I
   - D
3. Read & Write ops
4. Commit & Rollbacks
5. Transaction Isolation level
   - lost updates
   - dirty reads
   - phantom reads
   - Non-repeatable reads
6. Deadlocks

(5 Talk)

- 5
- 4
- 3
- 2
- 1

What are transactions.

Bank Example.



Accounts

id	name	balance	created_At	branch_id
1	A	₹ 1000	-	-
1	B	₹ 5000	-	-

1. Check the balance A ✓ DB call
2. Check > 500 ✗ App work
3. Reduce balance of A by 500 ✓ DB call
4. Increase balance of B by 500 ✓ DB call

transferMoney(from, to, amount)

≡ if (bal >= amount)

≡

```
else
{
}
```

⇒ Transfer Money may be getting executed by multiple people at the same time.

A → B (₹500)
C → B (₹500)

1. Check the balance A → Read ✓
2. Check > 500 → Nothing
3. Reduce balance of A by 500 → write
4. Increase balance of B by 500 → read & write

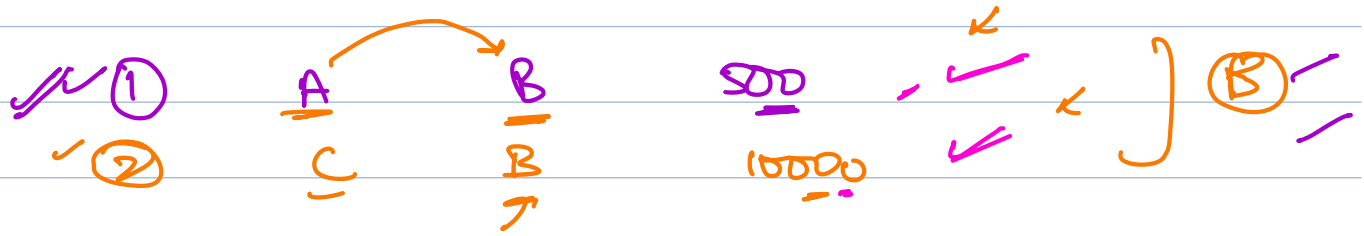
B += 500

↓

B = B + 500

This is NOT a single operation

→ Read B → temp
temp → temp + 500
write ← temp



A → 10000 500

B → 5000 5500 15000

C → 15000 5000

21000 20500

transfer Money (a, b, amount) {

Read A → x ① → 1000 ② 15000

if (x ≥ amount) {

write A ← x − amount ① → 500 ② 5000

Read B → x ① → 5000 ② → 5000

x := x + amount

write B ← x ① → 5500 ② 15000

}

5000 5500 15000

B → 5000 5500 15000

① Inconsistent / illogical state

② Entire operation may be incomplete

Transaction → A set of DB operations logically grouped together to perform a task

Transfer Money

```
A ← X
A ← X − _
B ← X
B ← X + _
```

Expectations / Properties of a Transaction

(ACID Properties)

A → Atomicity
C → Consistency
I → Isolation
D → Durability

Atomicity

from the word Atom → smallest indivisible unit.

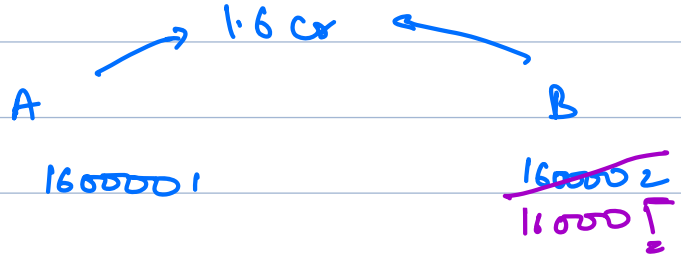
→ Transaction should appear indivisible to an end user
→ to an outside it should feel that either nothing has happened or everything is done.
→ A transaction should never end in an intermediary state.

Consistency

→ correctness
→ exactness
→ accuracy
→ logical correctness

Example where consistency is not needed.

Hotstar



stream-id	count
CSK vs DC	—

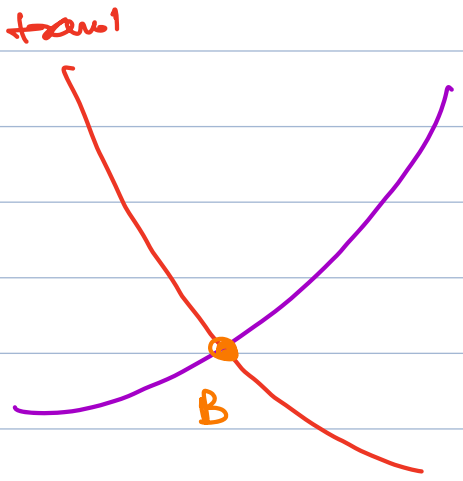
Consistency hampered

But performance improved.

The state of DB transn started and also after transn completed should be accurate.

ISOLATION.

→ One transaction shouldnt affect other transactions running at the same time on the same data.



Durability.

once a transaction is completed we would want its work to stay persistent.

↳ Store it on disk.

Break → 5 min

8:18

↓

8:23 PM →.

Commit and Roll back

students

id	name	psp	b_id
1	Ujjwal	80	2

update students
set psp = 100
where id = 1

↓

select * from students;

↓

psp = 100;

Auto commit.

→ every SQL query starts a transaction
→ execution itself.
→ saves the changes // commit

1. start transaction
2. Execute operation
3. Commits

↳ statement that is used to post it the result of an SQL query.

★ Commit takes care of durability

Auto commit = true By default.

session 1

auto commit = 0;

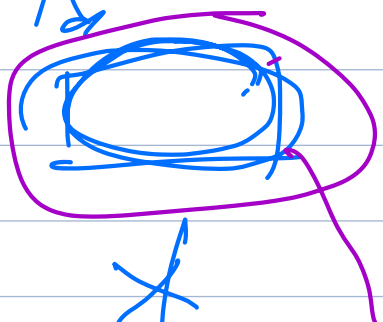
Update ⇒

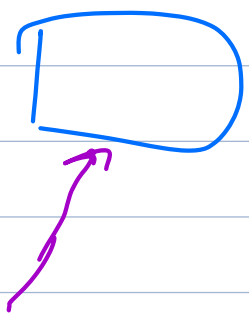
select

Commit;

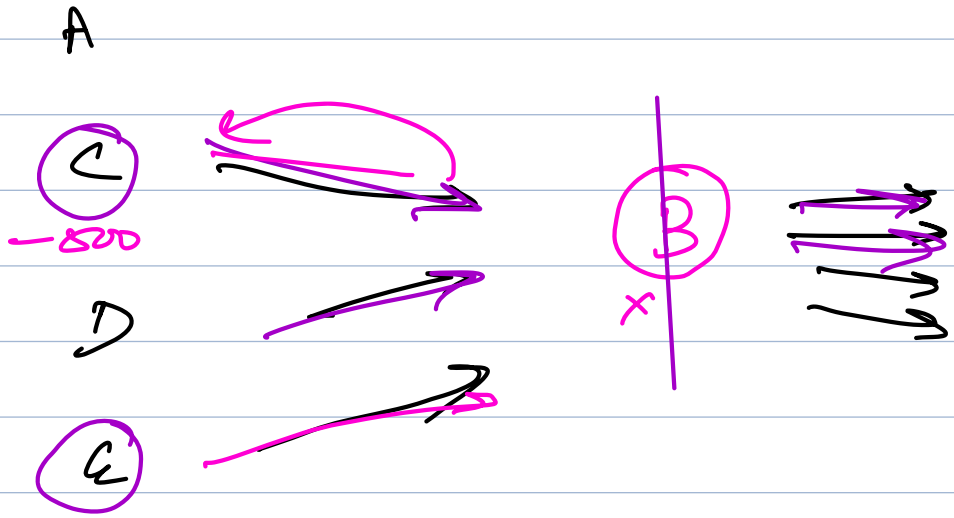
session 2

select



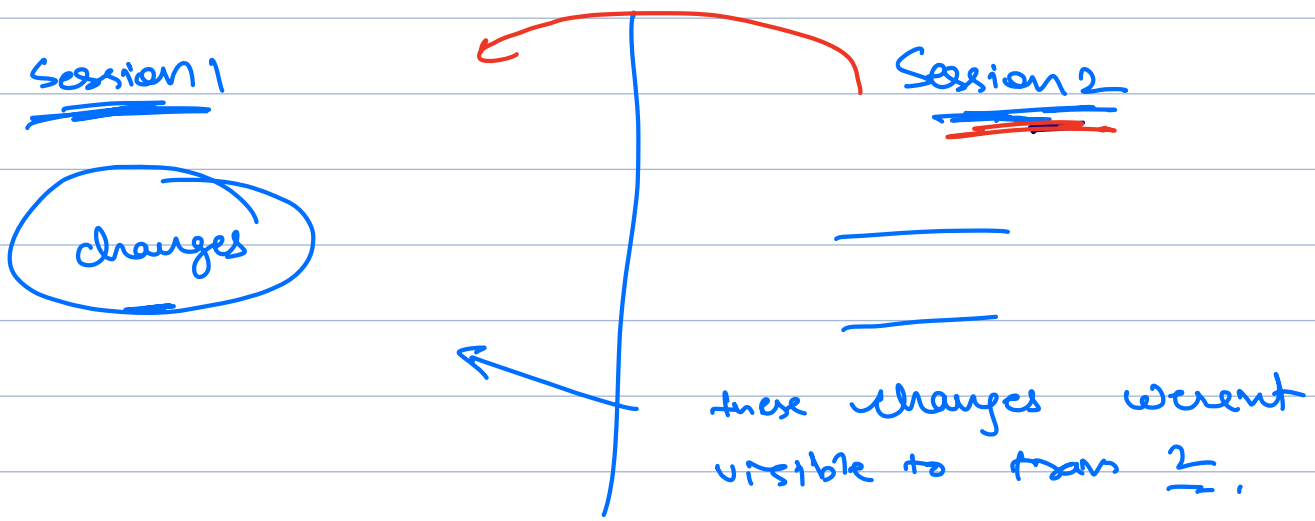


Rollbacks.



undo all changes done since last commit;

→ You cannot rollback after a commit is done.



ISOLATED.

4 levels of Isolation.

→ relaxed

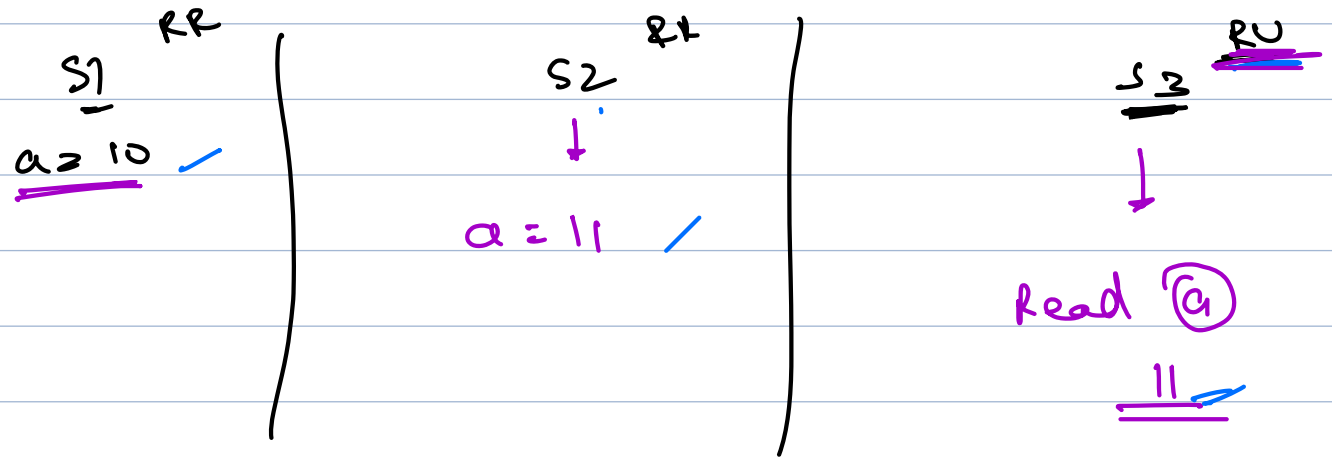
1) Read uncommitted
2) Read committed
3) Repeatable read → default for MySQL.
4) serializable → strict

↓ more order of security.

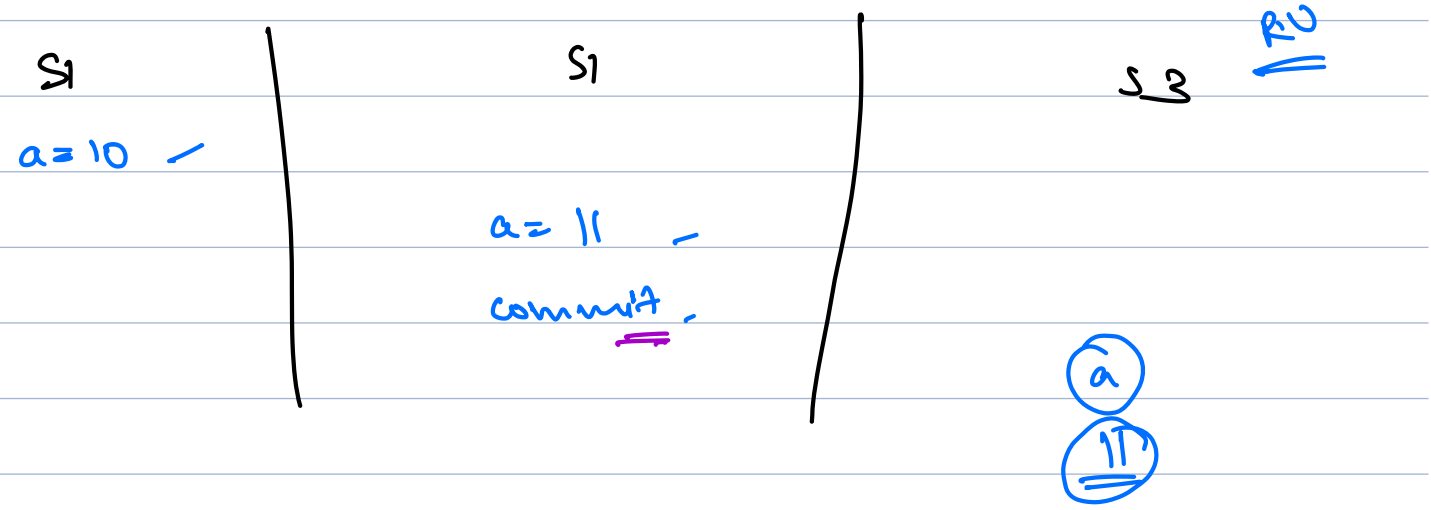
* Isolation levels are talking about what data is read, not what is updated.

Read uncommitted.

→ Allows a transaction to read even uncommitted data from another transaction
→ latest uncommitted data also



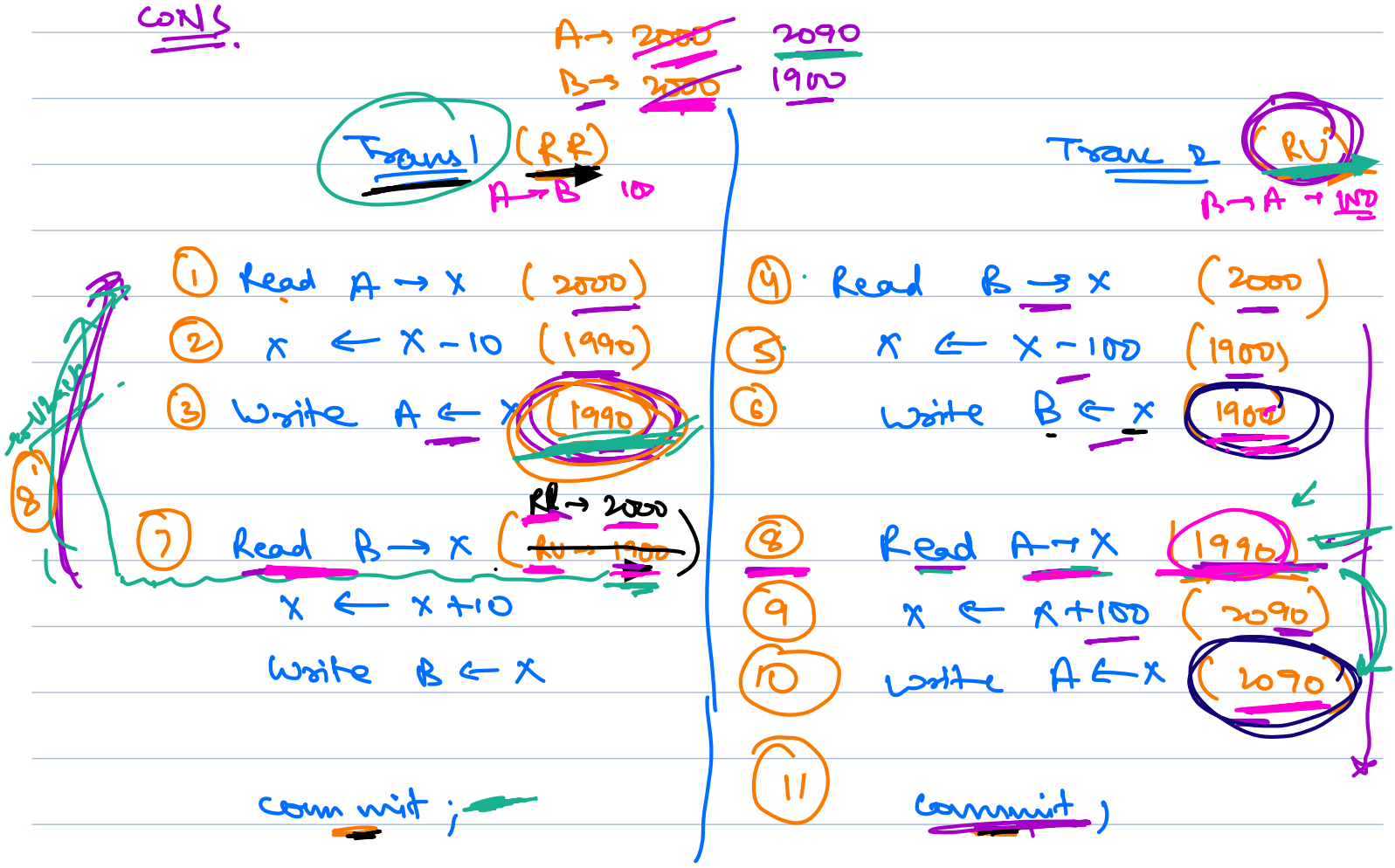
* Isolation level only talks about how your session will act (read data)



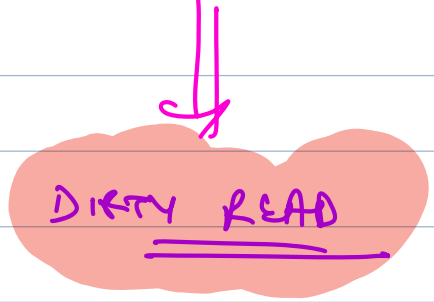
PROS.

Fast → performance

CONS.



At line 8 → trans 2 read data that was not committed and rolled back.



when a trans reads data that is not committed. That dat might change / rolledback / not yet committed.