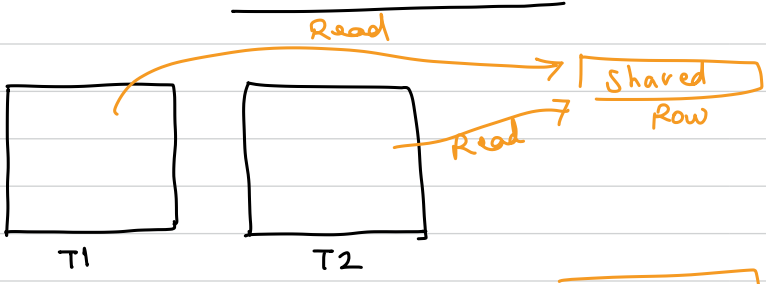
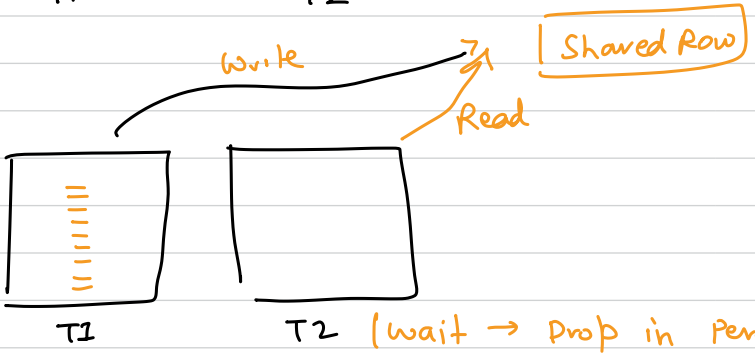
Transactions - II



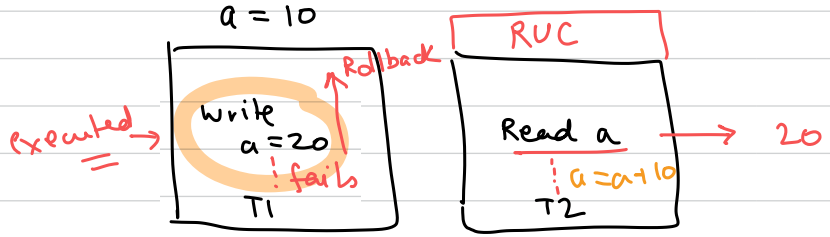
No Problem.



(wait → Drop in Performance)

TRANSACTION ISOLATION Levels.

1) Read uncommitted



→ fast
→ but can cause dirty reads.

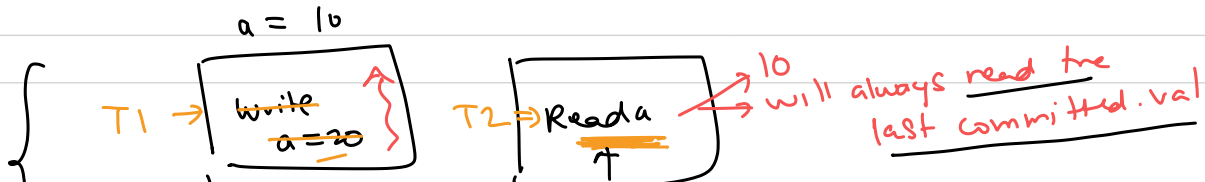
1.6 M views

used wrong value of 'A'

DIRTY Read.

1.60364
1.60070

2) Read Committed




Commit

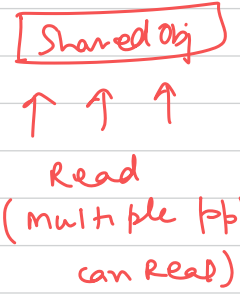

monkey
⇒ last Comm
Horse

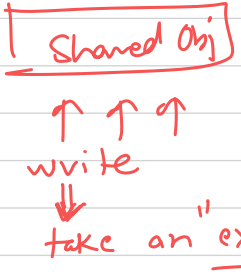
Demo

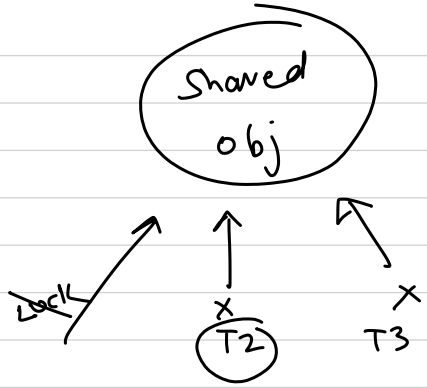

write horse
⇒ write Dog
Read 3

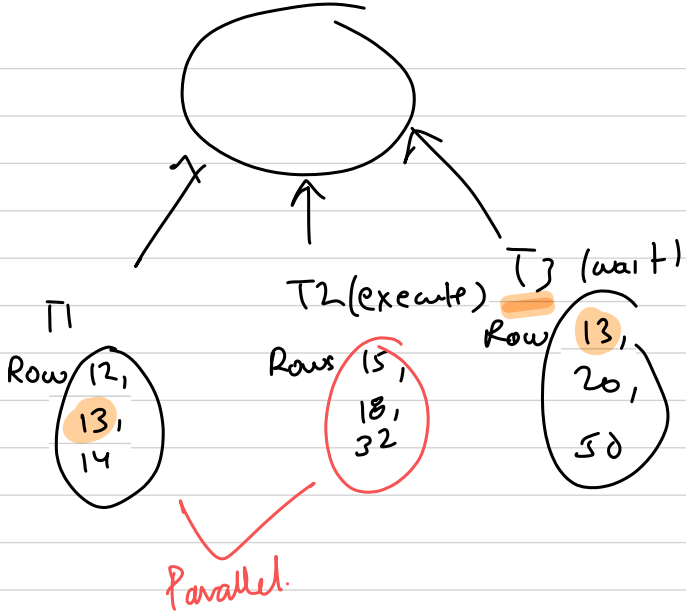
RC


begin
Commit
begin
Commit


Shared Obj
Read
(multiple ppl
can Read)


Shared Obj
write
take an "exclusive lock"





Adv:

→ No dirty read

→ last committed val.

Disadv: → less efficient than RUC.

→ one more issue ⇒ non-repeatable read.

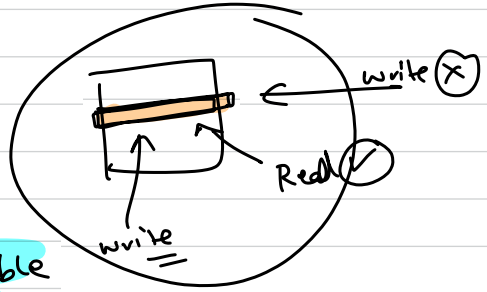
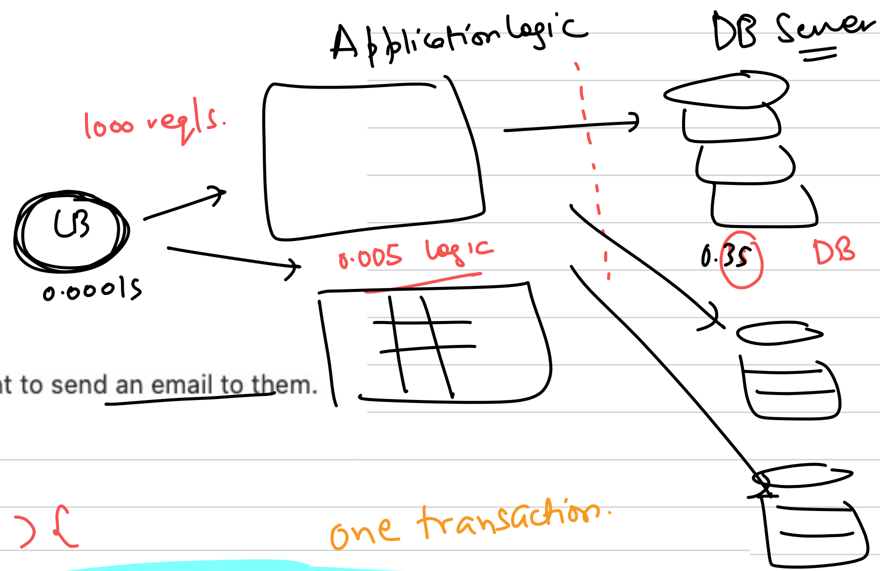


Table users

id	name	psp	email-sent
1	N	60	false T
2	A	80 78	false T
3	M	70	false T
4	AB	45	false T

Suppose for all students having psp less than 80, I want to send an email to them.



send_emails ( ) {

one transaction.

```
get    { list<users> l = SELECT * FROM users where psp < 80;
                            [N, M, AB]

send   {   for u in l:
               trigger_email (u);          ] 1 minute

update {   update ussers
           set email_sent = true
           where psp < 80;
```

N, M, AB

Repeatable Read.

N, M, AB

Non Rep. Read Problem

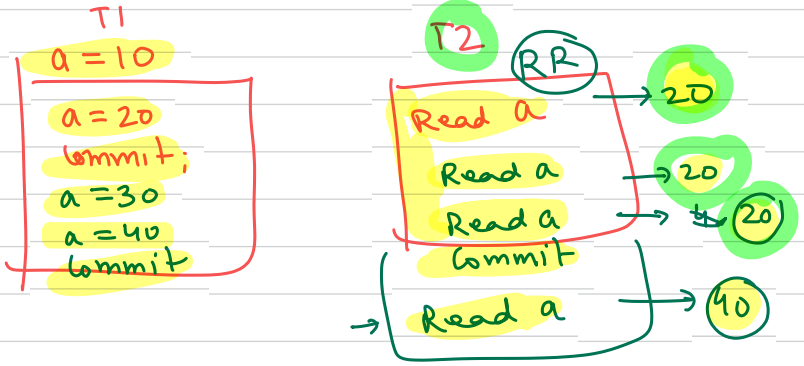
N, A, M, AB

ideal:
(N, M, AB)

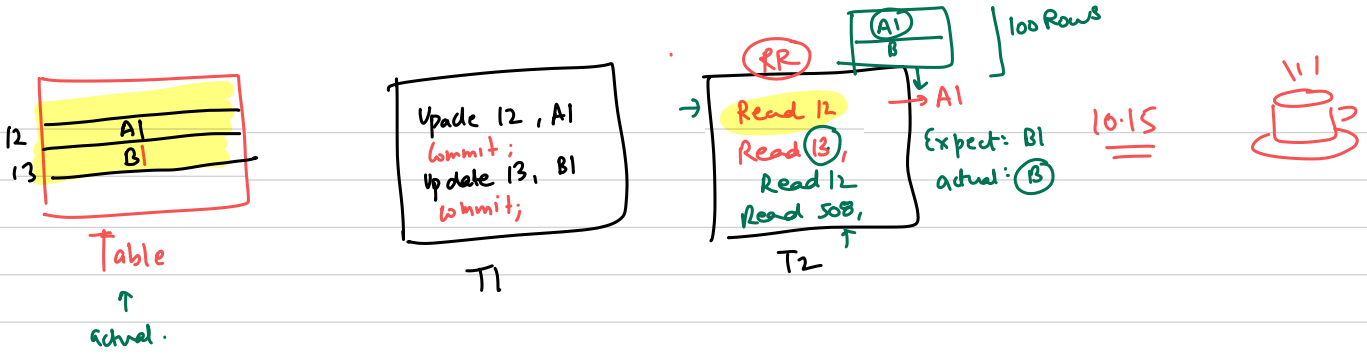
3 Repeatable Read. (Default isolation in MySQL)

For the very first time, it reads the latest committed value of the current row.

After that, until the transaction completes, it keeps on reading the same value it read the first time.



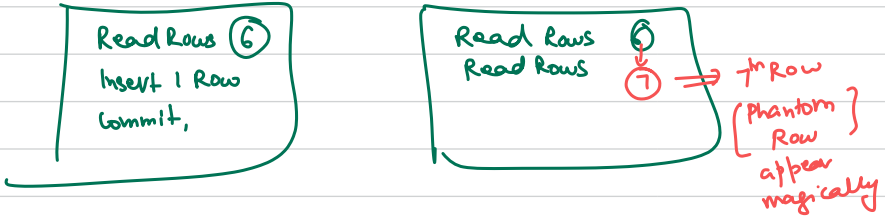
Snapshot



RR can read from the snapshot (for nearby rows also)
Or can read from the table the latest committed value for the first time, and it will get saved to snapshot along with nearby rows.

Phantom Reads:

RR → doesn't apply to new Rows
↳ talks about the rows which were already there.



Problems

Dirty Read → Non-Repeatable Read → Phantom Read → Performance

(RUC) (RC) (RR) Serializable