Agenda

- Isolation levels.
  - read uncommitted
  - read committed
  - repeatable reads
  - serializable
- dead locks.

start @ 9:10

Isolation levels

The form in which transactions will be isolated from each other in a database.

1. Read uncommitted
2. Read comitted
3. Repeatable reads
4. Serializable

→ most relaxed

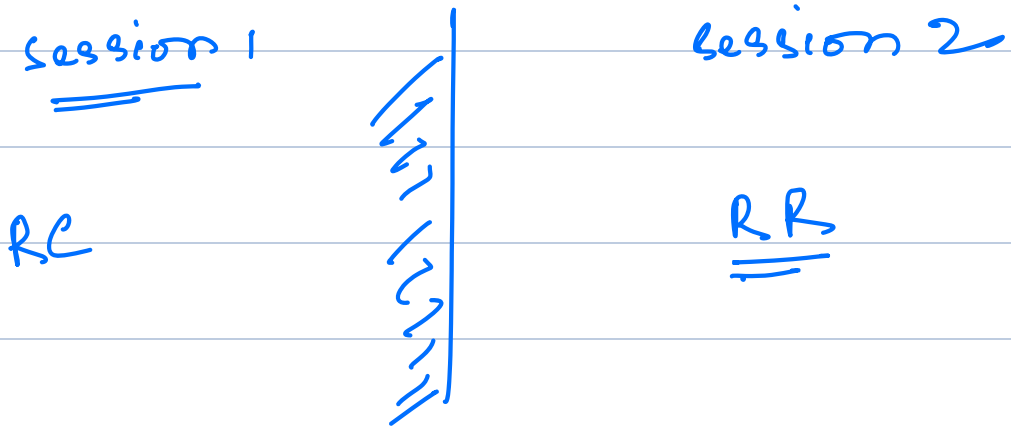
severity

→ most strict

default

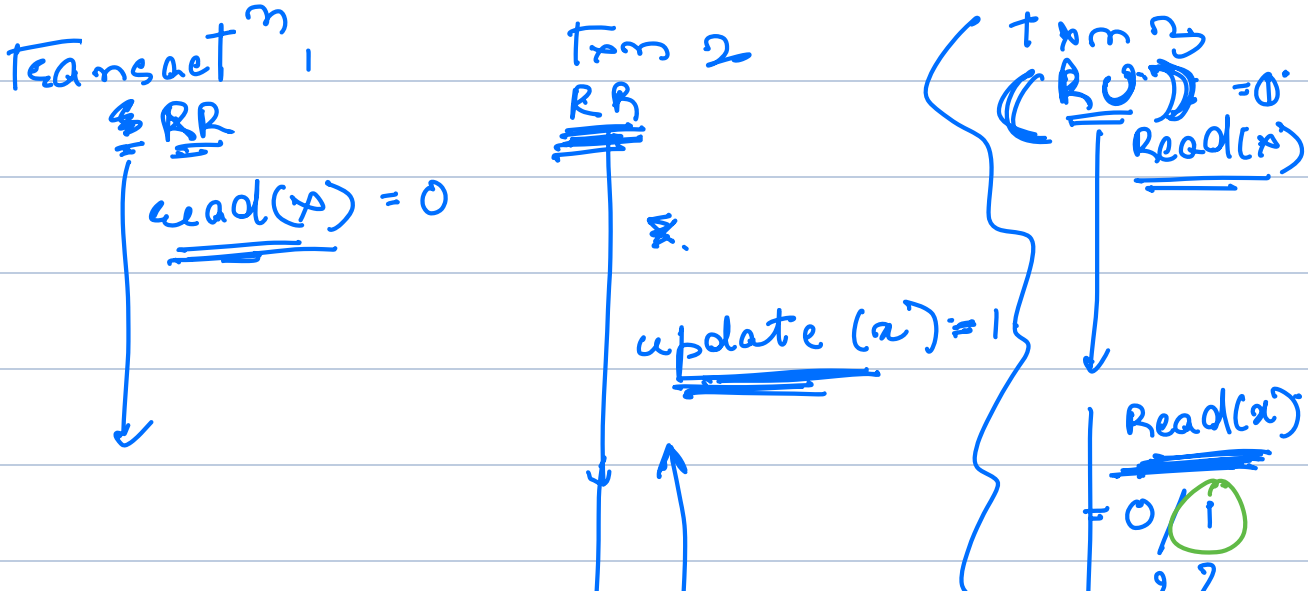
generally default.

each connections/sessions has its own isolation and are also not affected by others.



Read uncomitted

Allows a transaction to read uncommitted data from another transaction.



update (x) = 0.

read(x) ?? = (1)

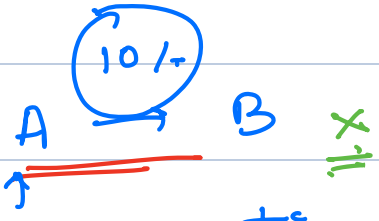
(0, 1).

→ basically means that you will read the latest data.

less strict ↓ fast ↑.

A = 2000
B = 2000

A = 2100
B = 100



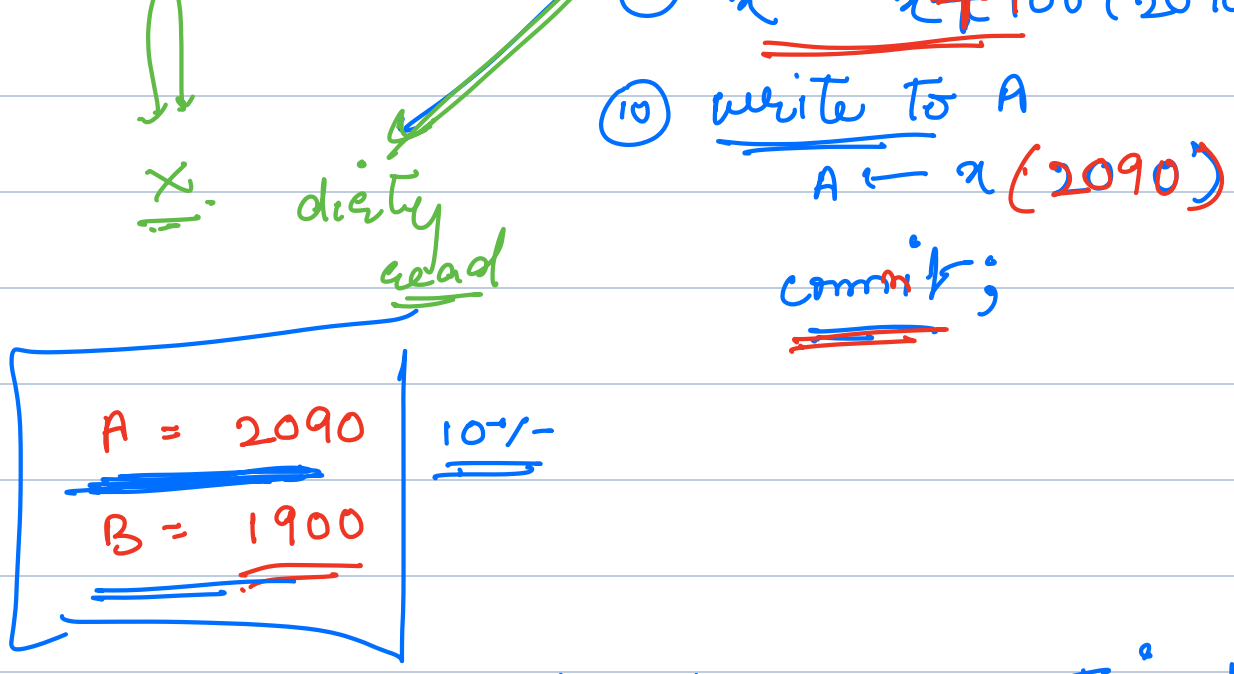
Transaction 1 (RR)

Transact^n 2 (RU)

① Read A → x (2000)
② x → x - 10 VA = 1990 CA = 2000
③ write A ← x (1990) CB = 2000
⑦ Read B → x (2000)

① Read B → x (2000)
② x → x - 100.
③ write B ← x (1900)
⑧ Read A → x (1990).
⑨ x → x + 100 (2090)

fails & Roll Backs



→ Tom 1 rolled back; Tom (RU) continued

→ dirty read : the state of reading uncommitted data.

→ we are not sure about the consitency of data that is read because we don't know the result of a an open transaction.

→ the open Trans MIGHT commit or MIGHT Rollback.

Read committed :

↑ ↑ data.

reads → allowed.

writes

RU — allowed

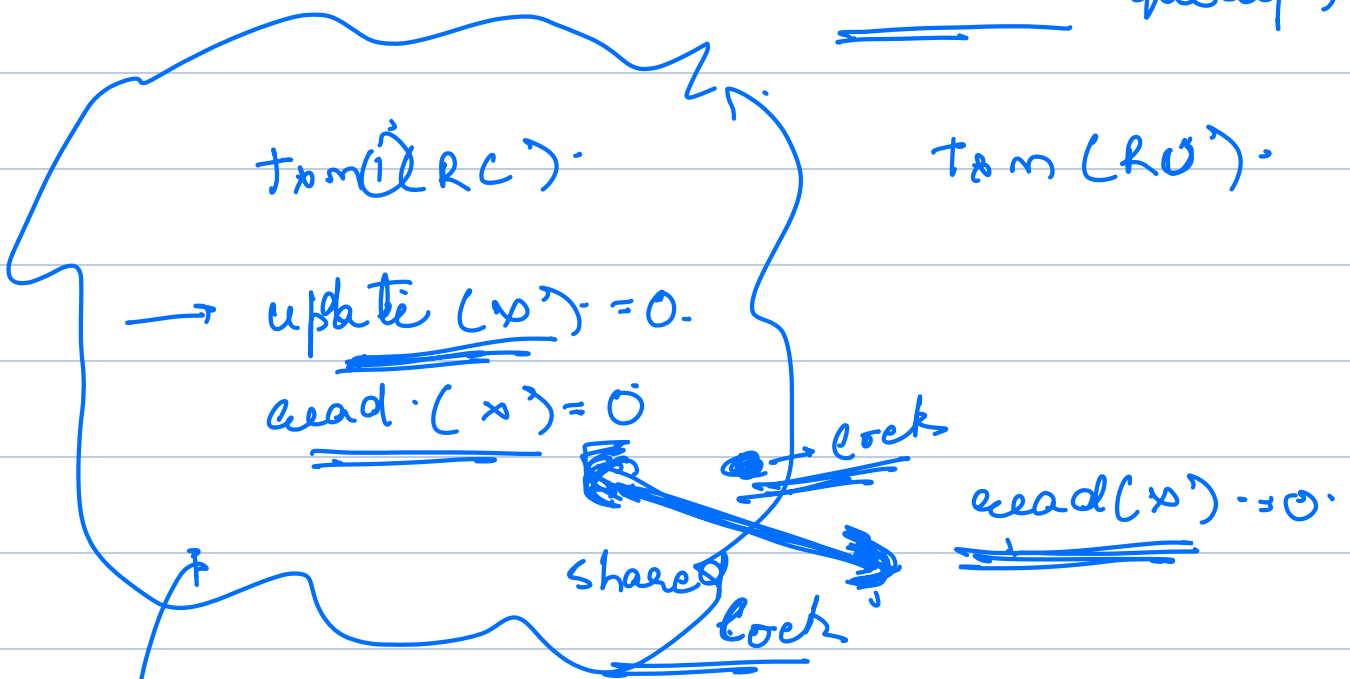
RC — halt??

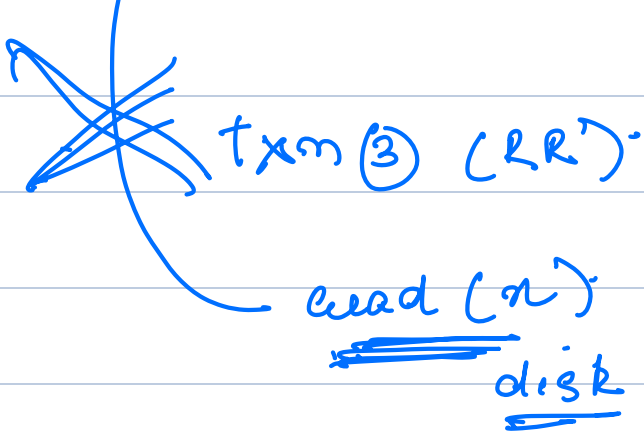
from diff txn

→ update on same rows. you get a hault.

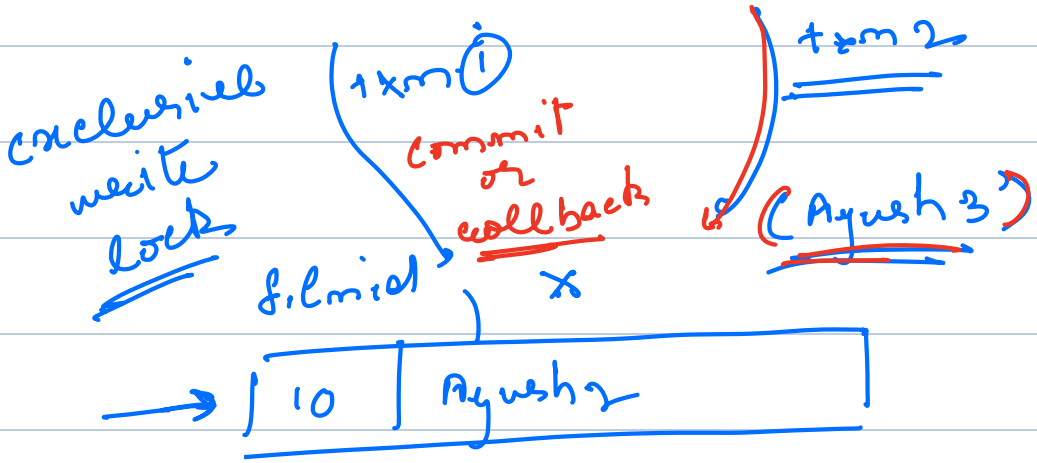
→ Beaz the 1st Transact^n aquired a Lock.

→ shared lock: (caused by a read query).





→ exclusive lock (updates / write query).



→ only and only one transactⁿ can write on a row when concurrent txns are going on.

Read committed :

① read values which are committed.

② No dirty reads.

RC is slower than RU.

→ validation

→ lots of concurrent control mechanisms.

break till 10:42

Problems with read committed

id	name	psp	is_email_sent?
1	n	60	false
2	A	(80)	false
3	m	70	false
4	ab	45	false.
5	ac	55	

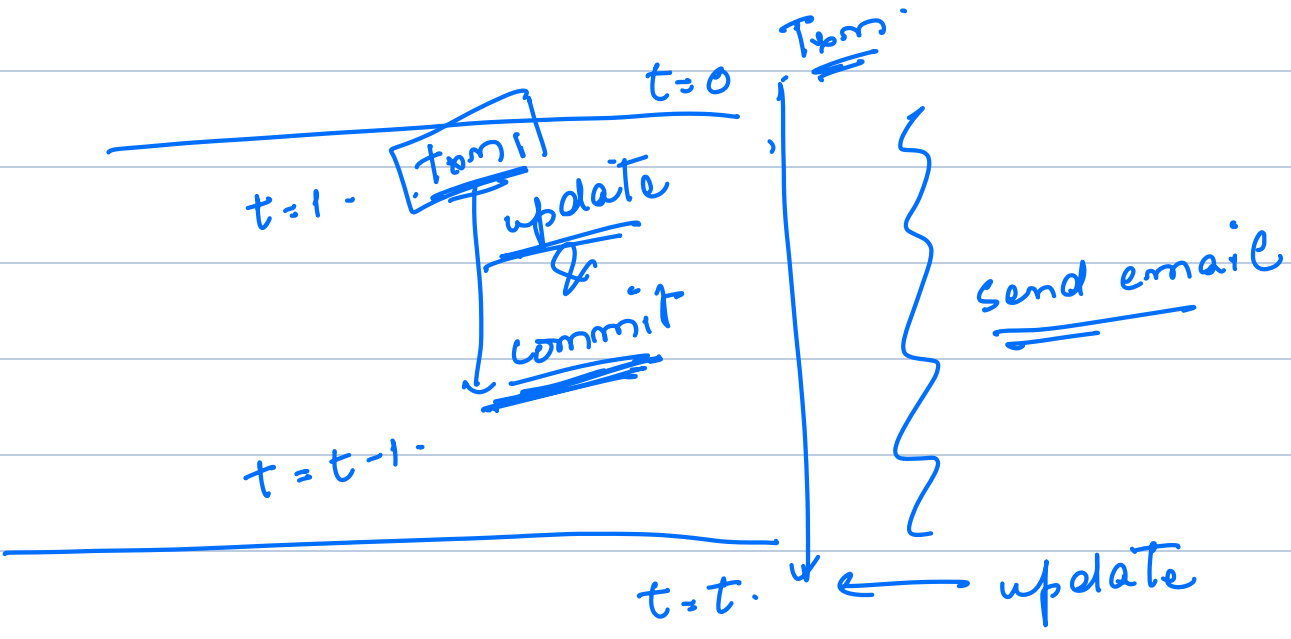
→ I want to send an email to students PSP < 80

step 1: getting list of users PSP < 80 (id 2 will definitely not be there.

step 2 : sending emails to users in the list.

(2 will not get the mail)

step 3 : due to some reason the PSP of user 2 was changed to 79.



step ④ : while updating "is_email_sent" we go Through all the values with PSP < 80

```
{ update students
    set is_email_sent = true
    where PSP < 80
}.
```

80 79 ?

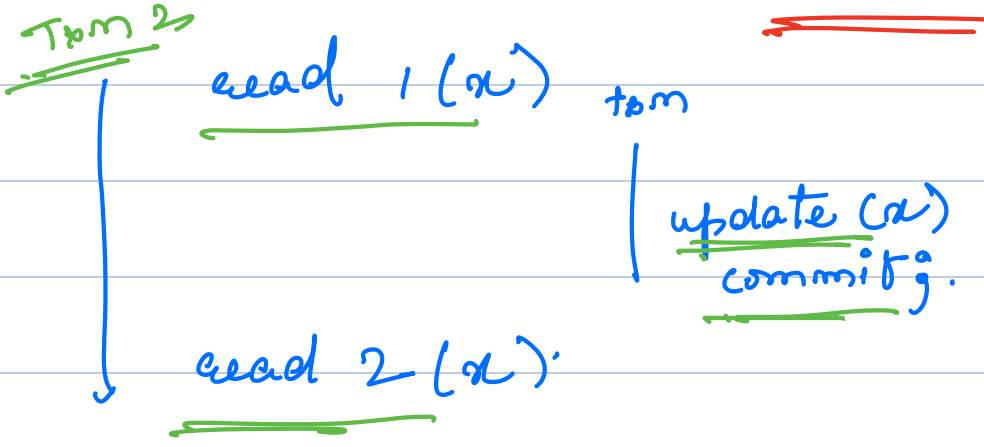
Did we send an email to user 2 ?

NO.

was it marked as sent ??.

Yes!!

with RR
80, 80, 80
80, 80
NO!!

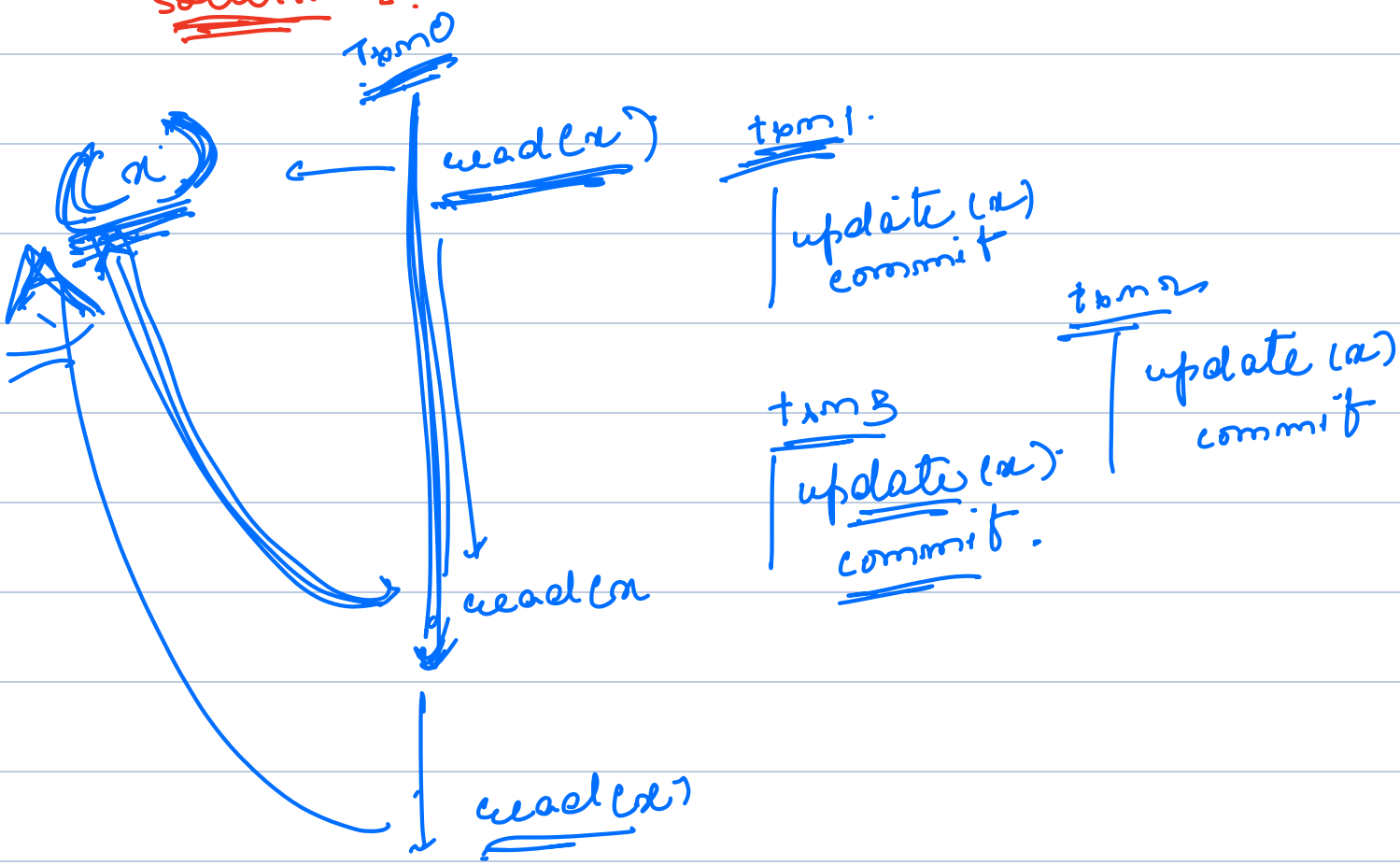


expectation
same.

reality
not same.

non - Repeatable read:
a transaction makes 2 reads at diff times and get diff values.

Solution:



Repeatable reads

→ for the first time, it reads the latest committed.

→ after that, until the txn completes, it keeps on reading the same value as the first time.

Dirty reads ? ? yes.

Non repeatable → yes Repeatable read isolatn

→ if we read a row again, will we get the same value?

→ 100 → 120

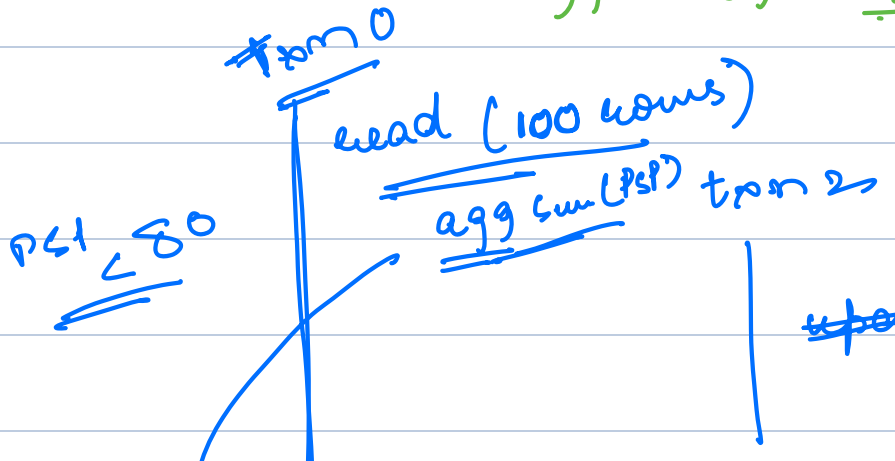
100
<80 PSP & C

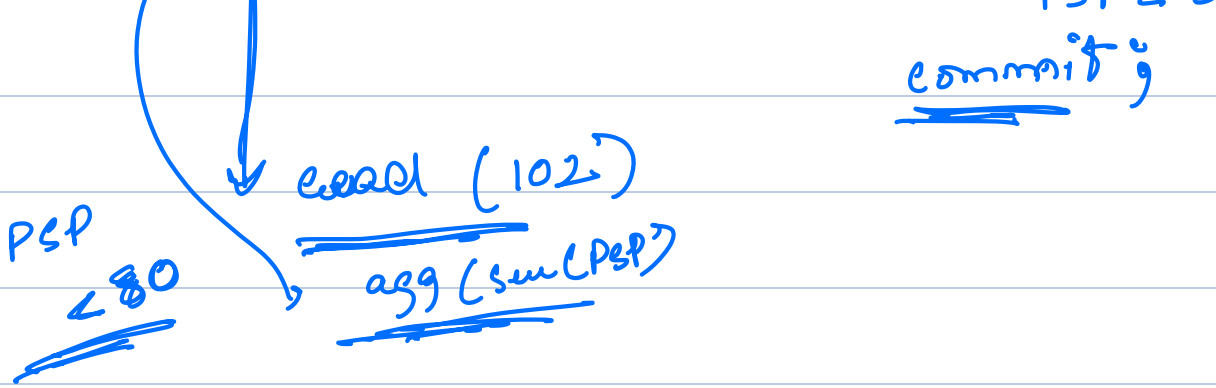
Phantoms reads:

the value of repeatable reads are same, but more or less number of rows satisfy your read conditions

100 rows → 120 rows.
(agg). (agg')

agg = agg' ✗




commit ;
exec (102)
agg (sum(PSP)
PSP
< 80