Agenda

- → Serializable
- → Dead Locks

} previous class left overs.

- → what is schema design
- → How to approach a schema design
- → Representing cardinalities
- → Deciding primary keys.
- → sparse relations
- → Relation with attributes

start @ 9:10.

Isolation level - serializable.

Phantom reads : more or less amount of rows in the same Txn.

one after another ?

or

one at a Time } Serializations

peak: multiple Queries running in a concurrent fashion

changing

A → [locked] {Lock}

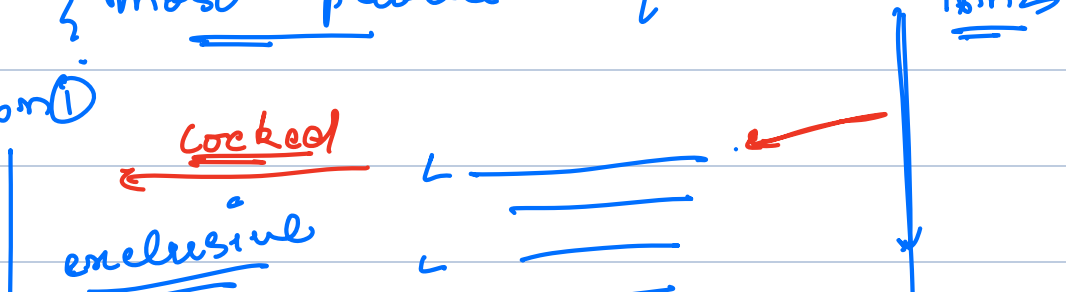
←

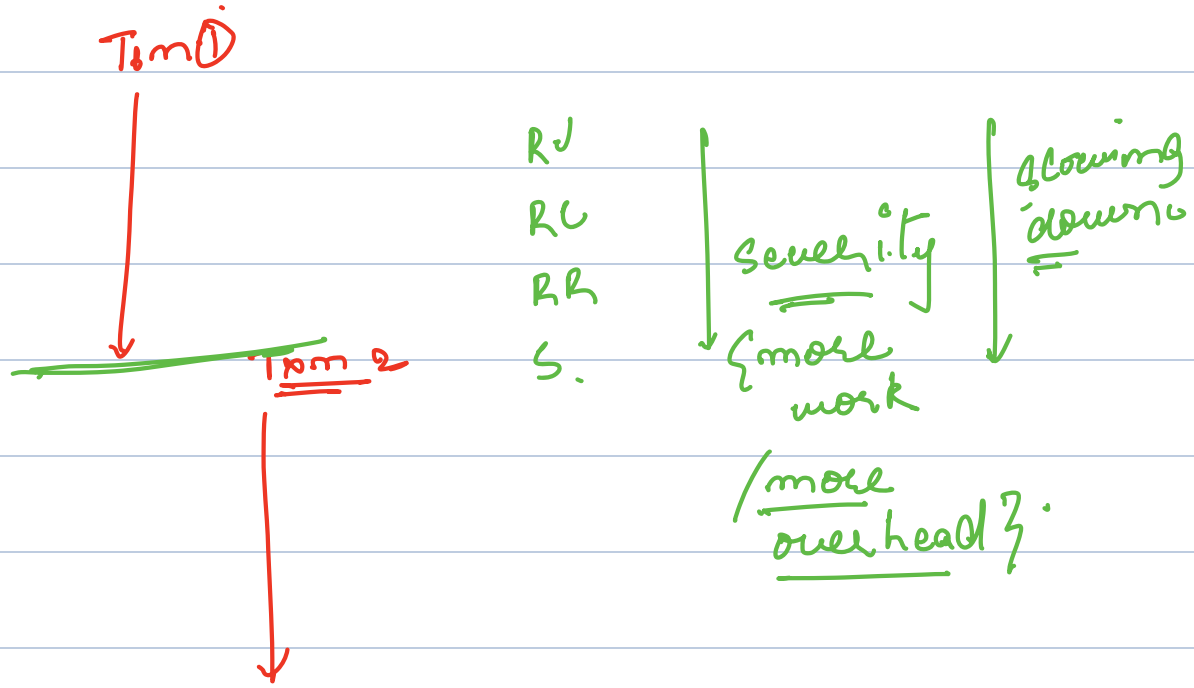
essentially eliminated. concurrency

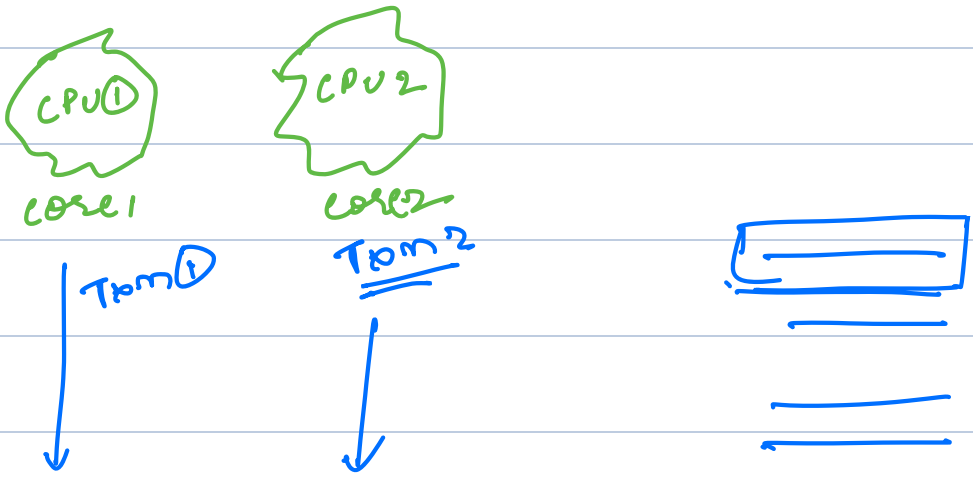
{ B → C → } wait in line

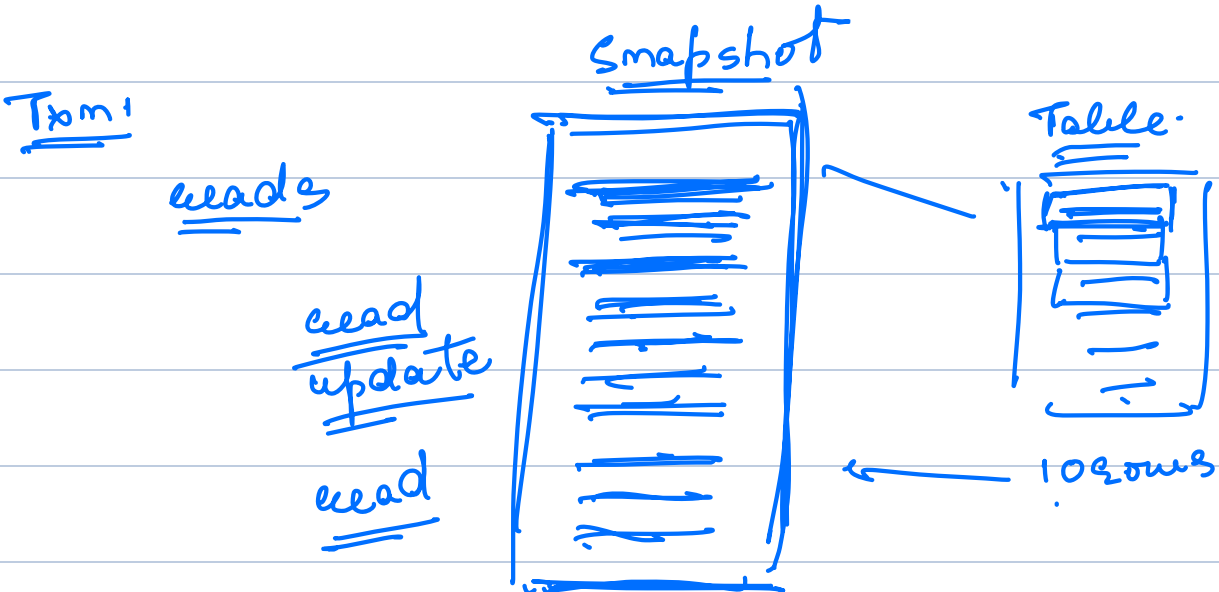
→ In serializable Isolation, you are only allowed to {read.} rows that are NOT LOCKED. {exclusive}.

read {most problems}.



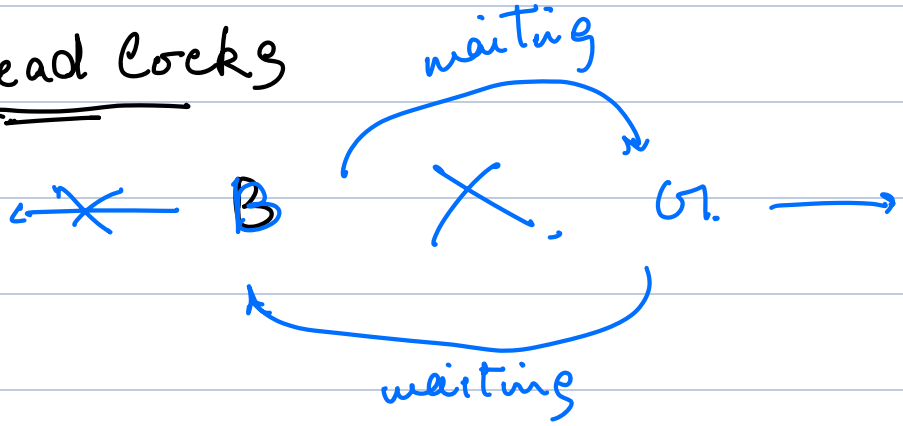

Txn①
Txn 2
RU
RC
RR
S.
security
more work
more overhead
slowing down

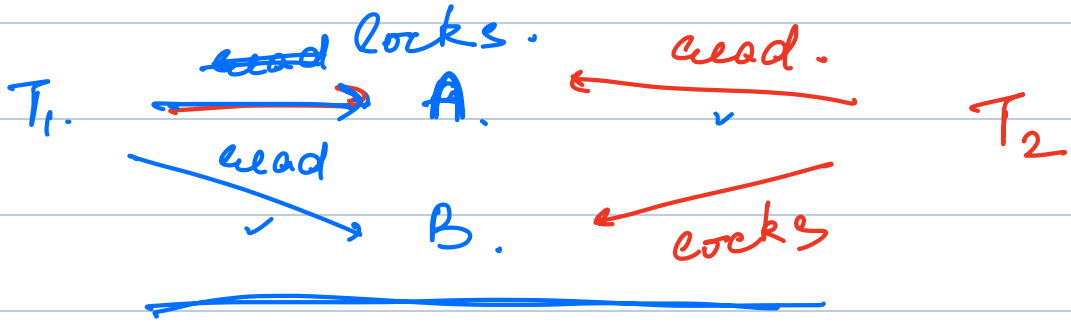

CPU①
CPU 2
core 1
core 2
Txn①
Txn 2


Snapshot
Txn 1
reads
read update
read
Table
10 rows

Thus, Serialization makes the concept of one person at a time, avoiding concurrency and maitaining consistency and complete isolation.

Dead Locks



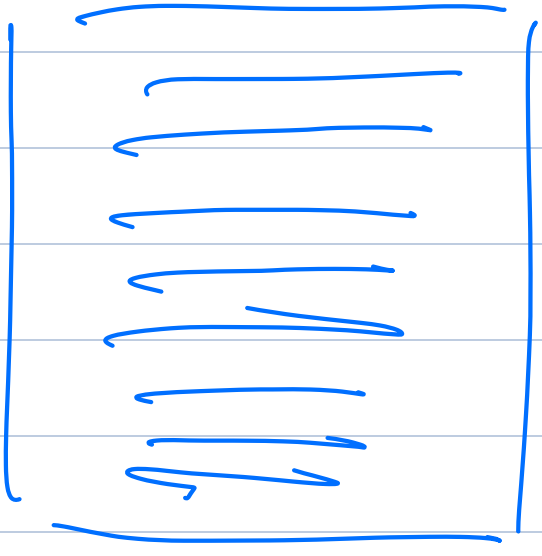


if there is a dead lock, either one or both transaction will be called back

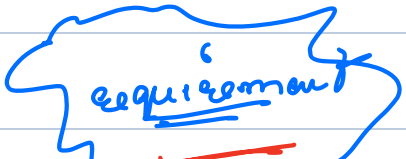
Schema design

What is a Schema??

- structure of your database.
- Tables
- columns (attributes).
- Primary keys
- FK and indexes
- one big table



Pictorial representation of how the database is structured..



→ holistic idea of all the components.

→ how all components interact with each other or in simple (a relationship).

→ layout a plan.

→ follow that plan.

In Tech

→ PRD { Product requirement document }.

→ ERD / RFC

Engineering requirement Document.

Request for comments.

→ data diagram or schema design

→ LLD or class diagrams (manipulate data)

→ HLD or architecture diagrams

→ Testing ().

break till 10:35

How To approach a schema design

→ case study of scalar.

① Scalar has multiple batches, name start date, current instructor, curriculum etc.

② Each batch has multiple students

③ Each batch has " classes

④ for each class we have, name of class date, time, instructors, attendance.

⑤ student, we store name, grad year, university name, email, Phone number.

⑥ we have a buddy, who is also

a student

(7) A student might move to another batch.

(8) for each batch a student goes to, we store the start date.

(9) every student has a mentor.

(10) mentor, name and company.

(11) store info about mentor-sessions, Time, duration, rating, student, mentor

(12) for every batch, we store if it was Academy / DSML

(1) creating Tables.

→ find the nouns

→ for each noun; if you have to store info about that

→ Yes, create a Table

→ No, move ahead.

(2) Naming of the Table.

→ Plural (batches., students).

→ snake case.

→ lower case.

→ "mentor_sessions"

③ Primary key.

① PK should rarely change.

→ Index.

→ if this changes, we have to reindex,

② data Type: Ideally a simple datatype.

③ meet.com/some-id.

→ auto-increment is predictable

→ security concerns.

④ name.

~~Table~~ noun_id.

batch_id

student_id.

⑤ attributes

→ ignore FK.

→ ignore the relations

→ just mention the attributes

the entity might have to store.

Tables

① batches.

batch_id , name , start_month

② instructors

instructor_id , name , email , avg_rating

③ students

student_id , name , email , Ph_number ,
grad_year , university_name .

④ classes

class_id , name . . .

⑤ mentors

mentor_id . . . .

Relations

→ cardinality

→ 1-1

→ 1-many.

→ m : m.

eg:

Student ⟶ batch.

→ How many batches can a student have.

1 ⟶ 1

→ How many students can in ① batch

m ⟵ 1

1 ⟶ 1

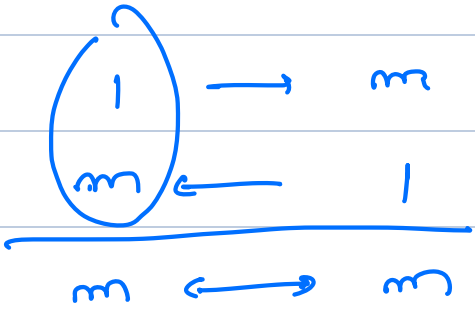
m ⟶ 1

m ⟶ 1

m : 1

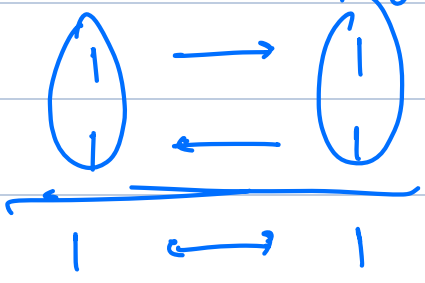
② cardinality of 2 users on facebook.

→ How many friends can a user have.



m : m .

③ Husband → wife



1 : 1

④ student → courses.

1 → m

m ← 1

m : m

⑤ order → Product.

1 → m

m ← 1

m : m

How To represent different cardinalities

① 1 : 1

husband → wife

id name. wife_id wife_id name.

② 1 : m

Student mentor

id mentor_id mentor_id students

[1, 2, 6, 7].

for 1:m, m:1, the id col of 1 side relation is included as an attribute in m side relation.

③ m : m

order and Products

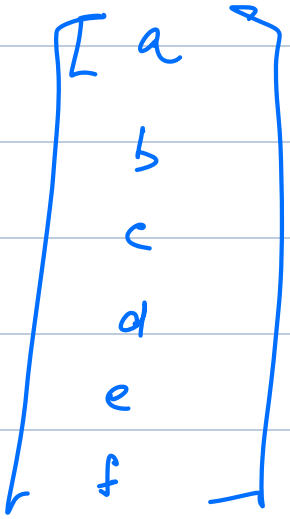
order_Products

order_id	Product_id
1	1
1	2
1	3
2	1
2	2
2	3

for m:m cardinality create a seperate Table {which stores ids of the entities}.

1 [a - g]
2 [g .. z].

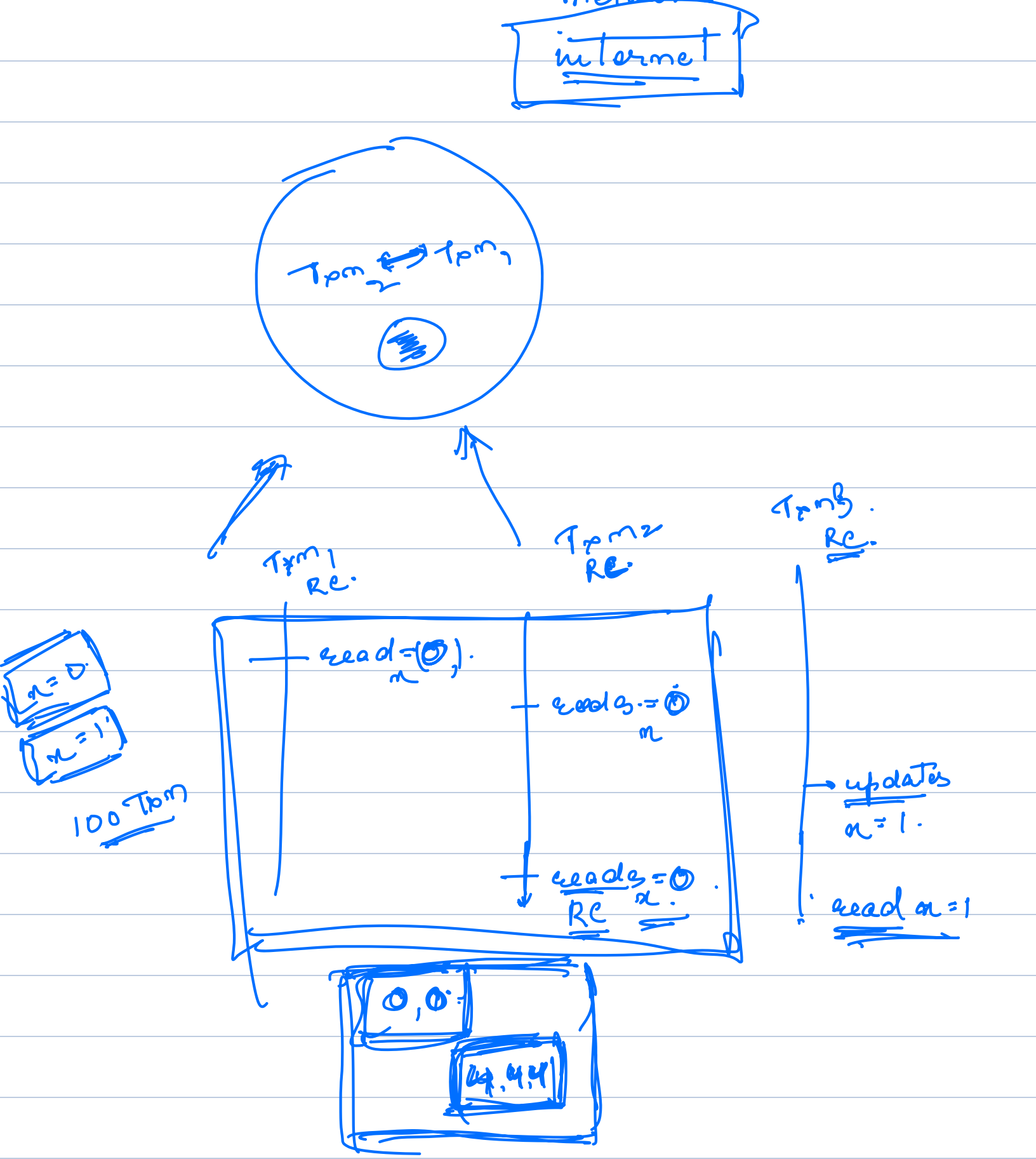
26



device - curretime stamp

100 → for (100) send email → update → 100

network


internet
Tpm2
Tpm1
Tpm1
RC.
Tpm2
RC.
Tpm3.
RC.
read x = (0,1).
read y = 0
x
x = 0
x = 1
100 Tpm
updates
x = 1.
read y = 0
RC x.
read x = 1
0,0
4,4,4