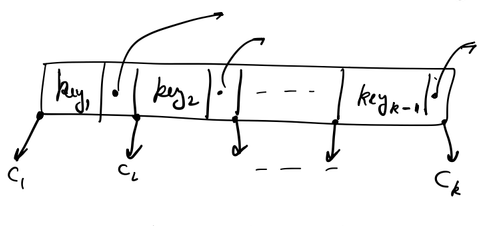
DBMS 2

Monday, 10 May 2021 9:10 PM

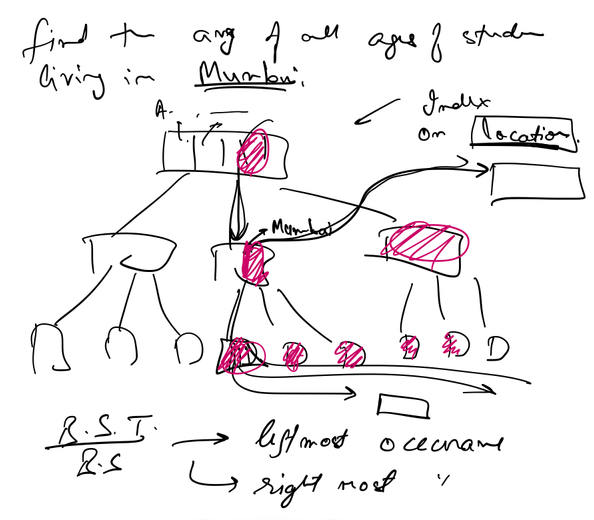
Indexing → B trees
- ① optimized for disk access
  - ↳ minimize the no of disk seeks (# of blocks that we have to read)
- ② All leaves are at the same level
- k → max k children
  - max (k-1) keys in a node.



key₁
key₂
- - -
key_{k-1}
C₁
C₂
- - -
C_k

st_id	name	age	Location

find the avg of all ages of students living in Mumbai.



A.
Index on Location.
Mumbai

B.S.T. / B.S → leftmost occurrence
↳ right most "

---

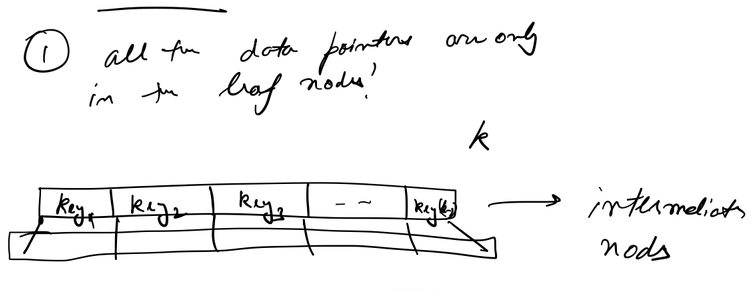
① B Trees are not optimized enough
↳ still increase the width

② B Trees are not the best for sequential data access

↳ B^+ trees

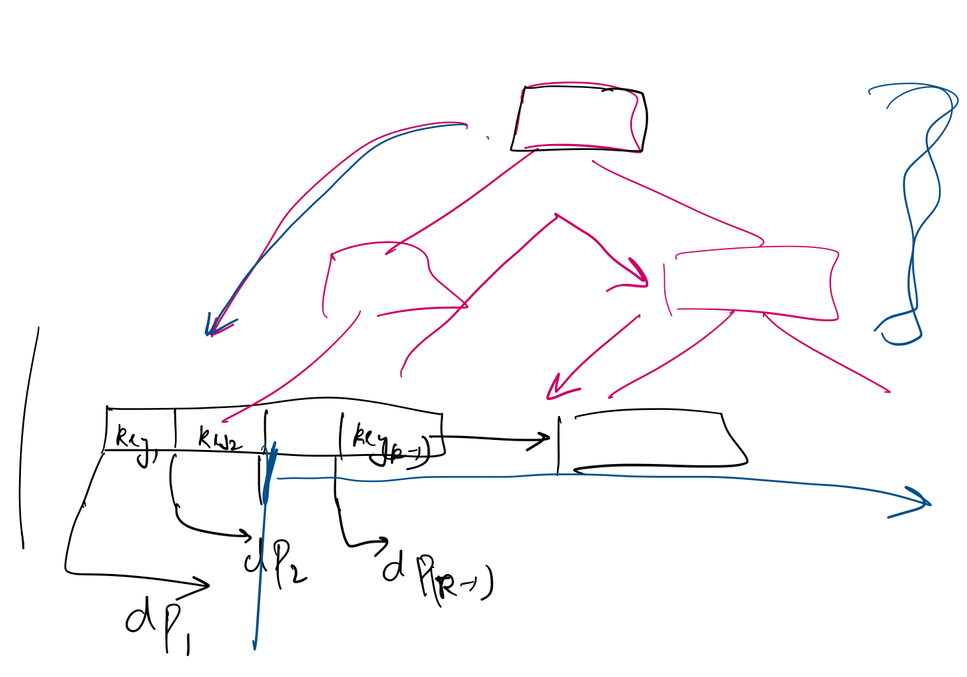
① all the data pointers are only in the leaf nodes!

k



Key₁
Key₂
Key₃
- -
Key(k)

→ intermediate nodes



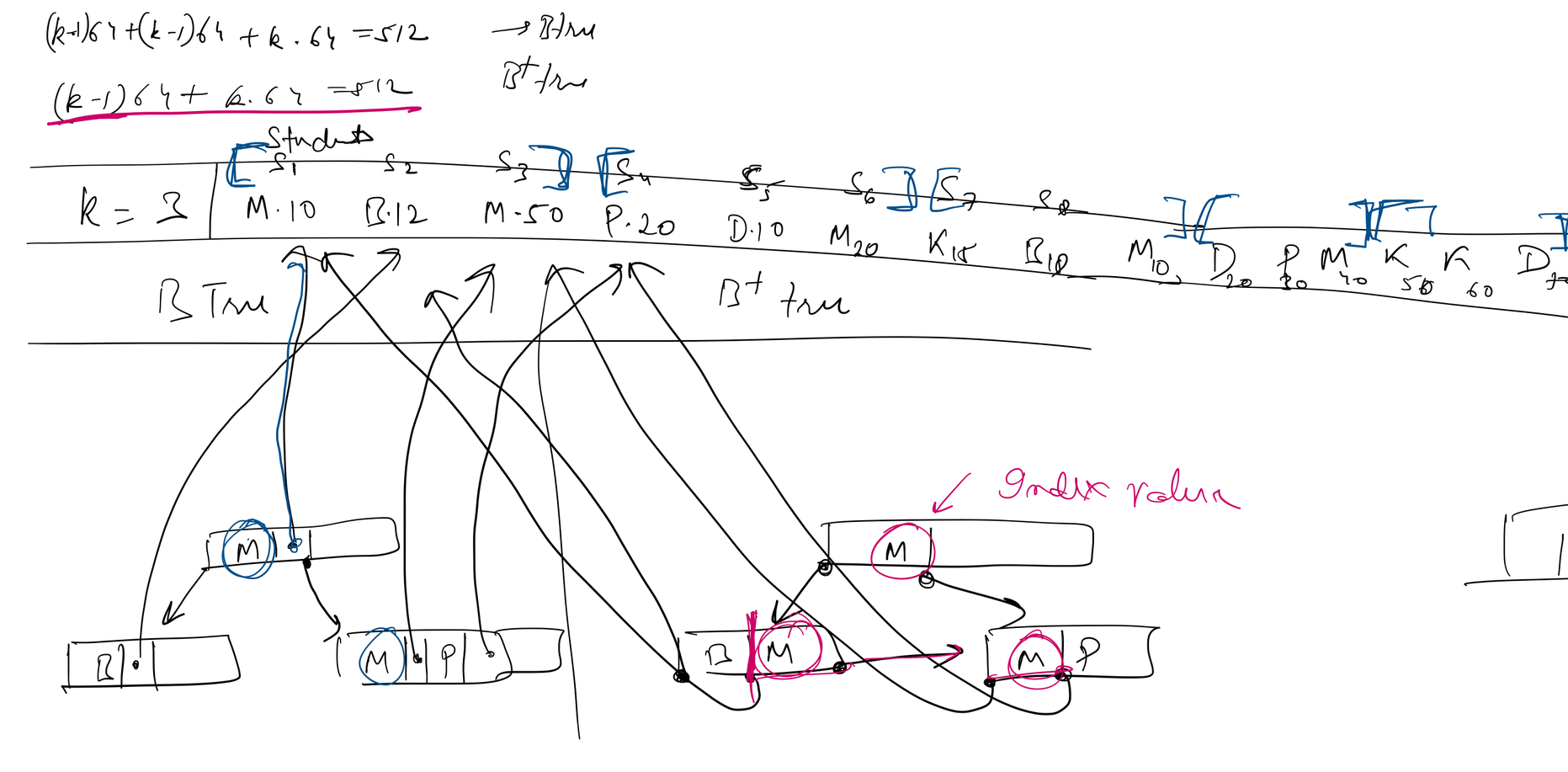
Key₁
Key₂
Key(k-1)
dp₁
dp₂
dp(k-1)
Mumbai

Btree	B^+ tree
each node	each node
(k-1) keys → 64 bits	(k-1) keys
(k-1) data → 64 bits	k child
k child → 64 bits	

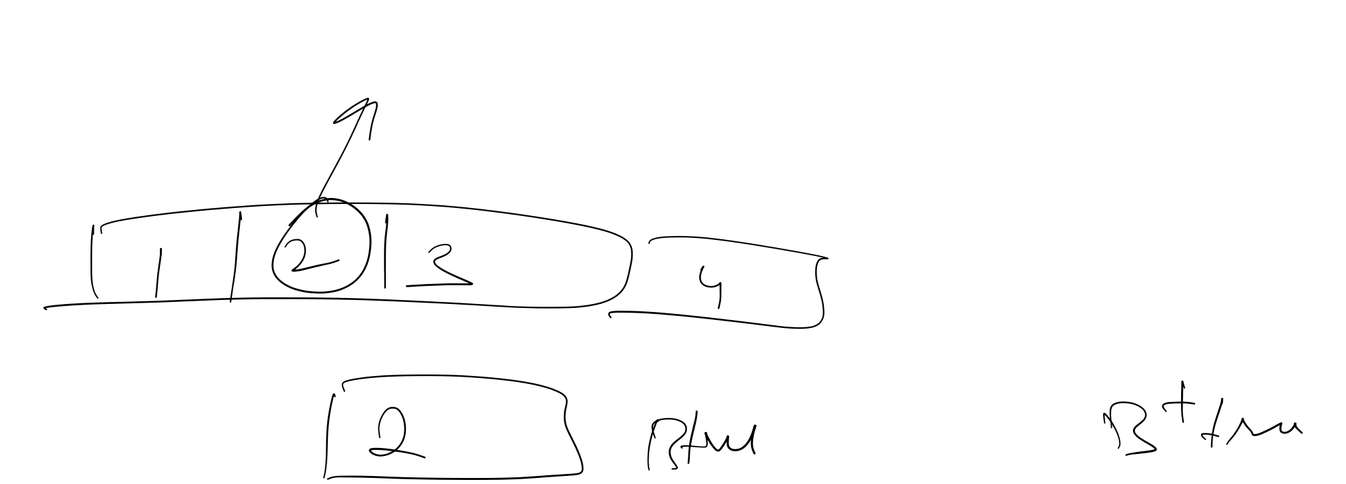
block size = 512 Kb

(k-1)64 + (k-1)64 + k.64 = 512 → Btree

(k-1)64 + k.64 = 512 → B^+ tree



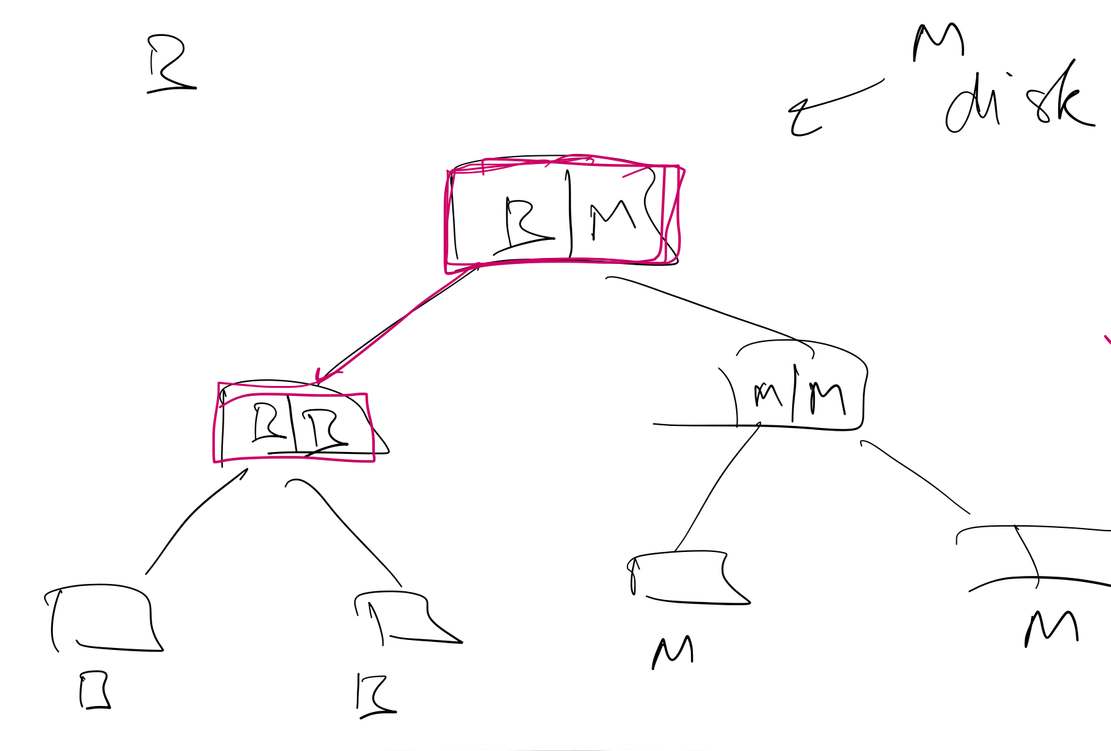
Students
K = 3
S₁ M.10
S₂ B.12
S₃ M.50
S₄ P.20
S₅ D.10
S₆ M₂₀
S₇ K₁₅
S₈ B₁₀
M₁₀
D₂₀
P₃₀
M₄₀
K₅₀
K₆₀
D₇₀
B Tree
B^+ tree
Index value
M
B
M P
B M
M P



1
2
3
4
2
Btree
1
3 4
B^+ tree
2
1 2
3 4

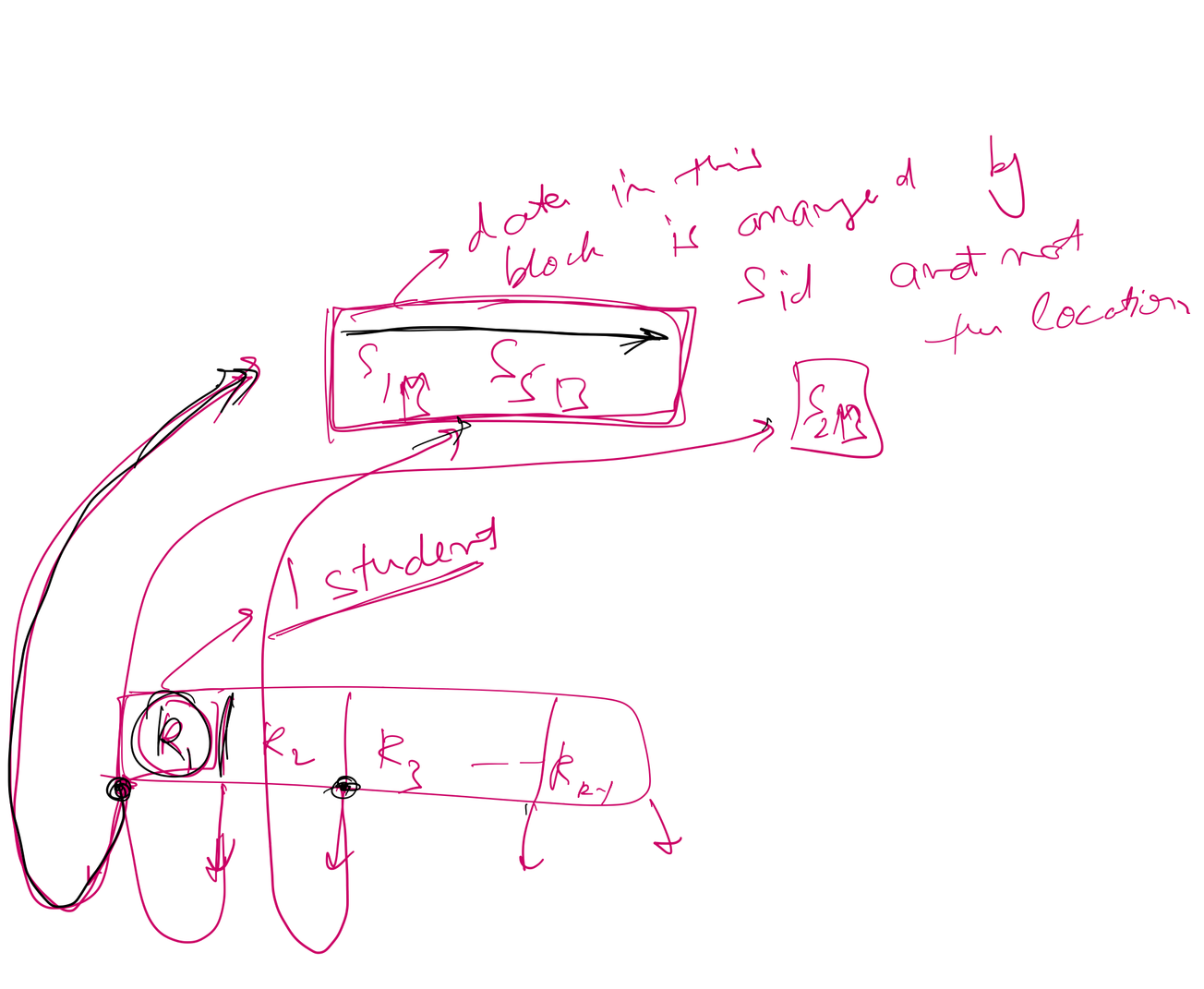
S₁ – S₅₀
B

S₅₀ – S₁₀₀
M
disk



B M
B B
M M
B
B
M
M

↓ height as low as poss



data in this block is arranged by sid and not the location
S₁ ... S₅₀
S₅₁
1 student
K₁
K₂
K₃
-- K_key

Indexes on the non-primary key

Index on the primary key

Order the data in the order of the pkey

if we insert that data is inserted in the order of pkey in my disk

pkey index B^+ tree → small

↳ clustered index → the data is arranged in the order of the key

an index on the pkey will perform better than an index on the non-prim key.

---

creating index on 1 key / col.
↳ simple indexes

create composite indexes as well

(location, age) — age

key | B 50 | B 60 | M 10 |

sid | location | age

| B 10 | B 20 | B 30 | B 50 | M 1 | M 2 | M 5 | M 10 |

find all students with age b/w 20 & 30

as the no of cols in index increases → key size increases
↳ depth increases

index (location, age, name)

find all people with location = '' ✓

find all people in Mumbai with age b/w — — ✓

find all Mumbaic people with age b/w — — & name = '' ✓

find all people with name = '' ✗

Table → students — — —

sid (pkey) location age name

clustered index

(location, age) (name, age)

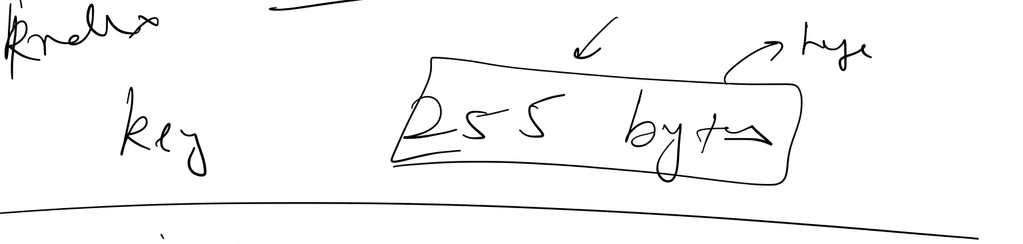
(age)

Indexes → help us speed up queries!

down sides
↳ if requires maintenance
insert / delete / update

if no index insert

if index insert



append
append

improves query speed
decreases insertion / del / update speeds

---

index on the (name)
↳ varchar (30) ? ✗
↳ varchar (255) ✓

dense index → key | 255 bytes | → huge

---

sparse index

index on the first (3 chars) of the name

key 3 bytes

---

never have an index on any text col.

if an index on a varchar col
↳ make sure that it is a sparse index.

[assumption (3 chars) → uniform distribution]

db → 1 million people → first name / → last name differs

~~name = [~~]

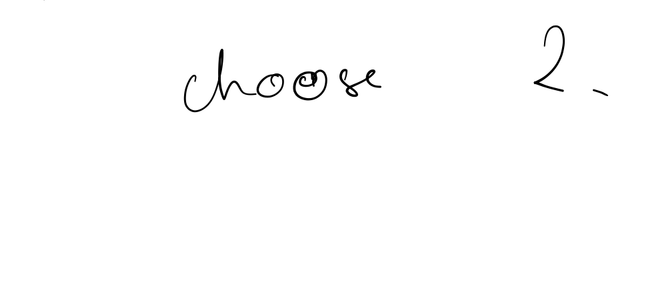
(name)

query on the reverse (name)

s_id | name | age

pkey

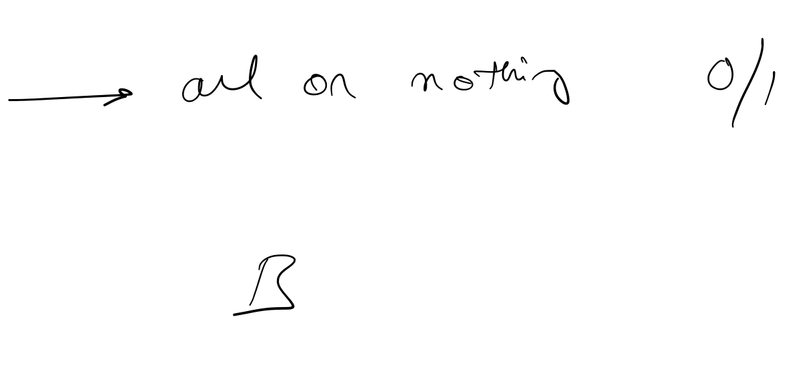
create index —— on rev(name)
↓
calculated cols.



DNA → 100s of Tb
Persistent
large
fast
16 Tb HDD ~ 15000Rs
super expensive SSD ~ 80,000Rs ~ 1Tb
RAM → 1Tb ~ 80,000
choose 2.

Transactions

query
↳ db engine



Transactions — Relational database

↳ certain guarantees

ACID → transactions

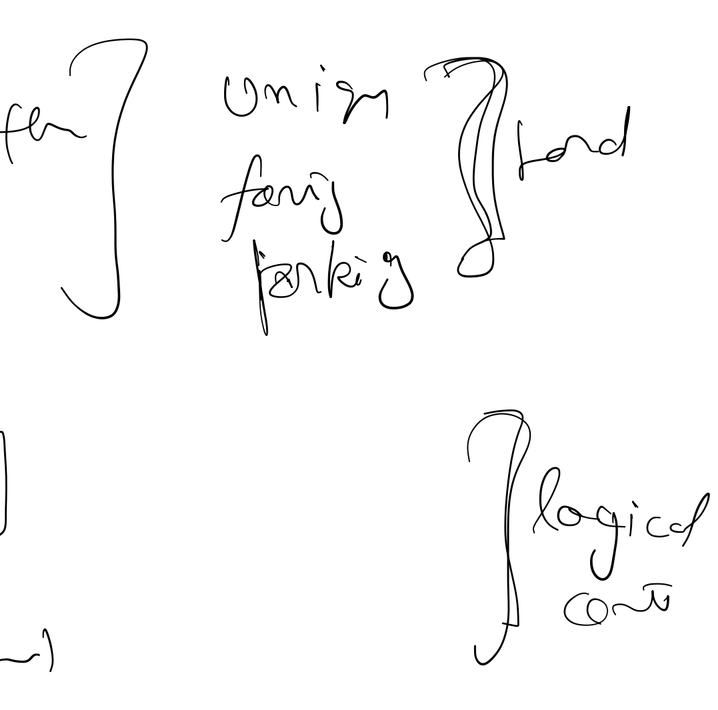
Atomicity → all or nothing 0/1

A B

transaction → A gives 100 Rs to B

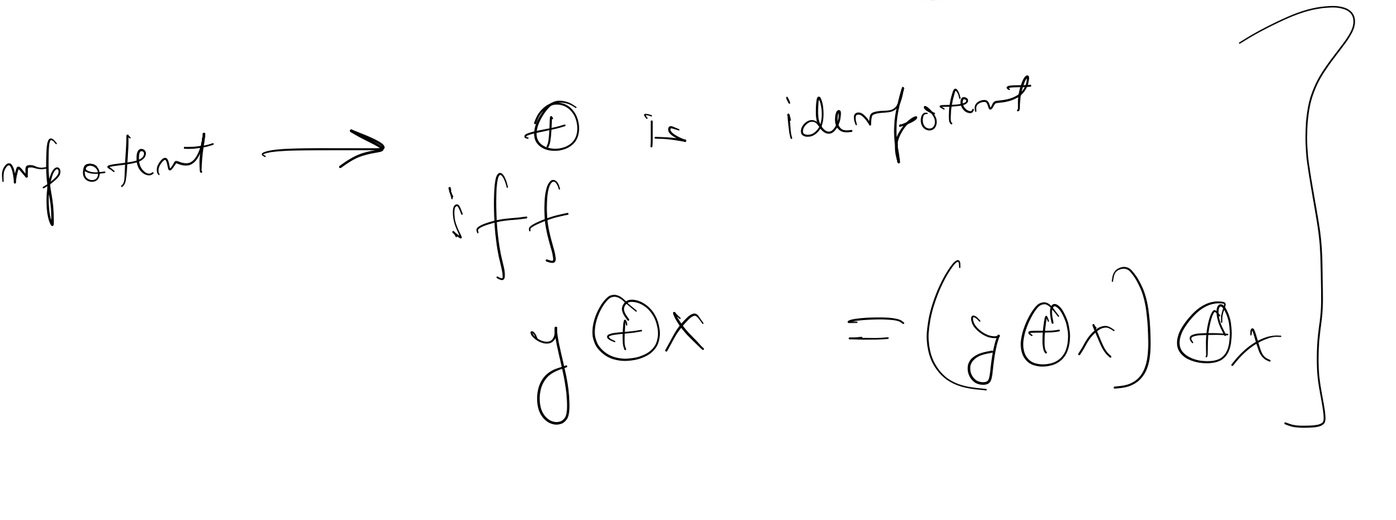
Begin Transaction
A balance ≥ 100
A balance — = 100
B balance + = 100
Commit/end

rolled back



total money remains const
the programmer
A
B
200
100
200
100

Consistency → satisfying all the constraints } unique, foreign, ranking } hard



S₁
DB
consistent state
transaction
S₂
DB
consistent state

} logical cons

Isolation → DBMS multiple transaction
↳ unaware of the other

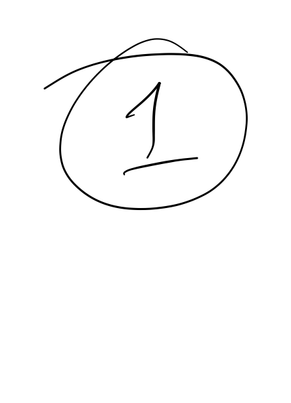
Api endpoints REST } idempotent → ⊕ is idempotent iff y ⊕ x = (y ⊕ x) ⊕ x

youtube like a video

(count # of likes on a video] ++

2

①



Isolation | Serializability

different

Durability → persistance

ACID } Relational database

↳ BASE, RAID } HLD

DBMS → theoritical
↳ sparse
↳ clustered
↳
↳ data
↳ indexing

↓

question on SQL queries

Relational databases

Choice of db engine
→ Postgres } → has no licensing issues
→ MySQL } →
→ SQLite → stripped down / minimal version
→ Oracle db → propriotary technology

---

clustered index → disk points → block head

non-clus → disk pt → block head / (int) → row #