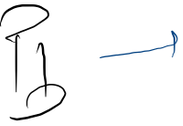
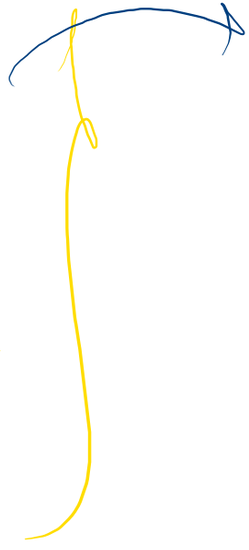


→ FDBSS

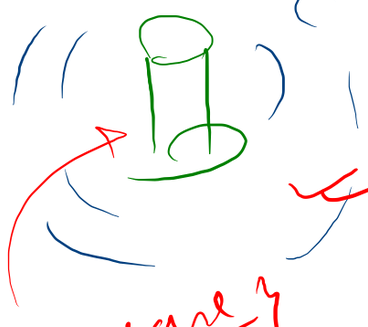
① Redundancy

② security

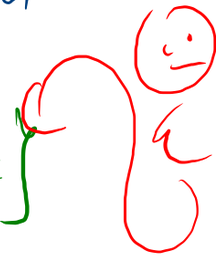
③ ease of usage



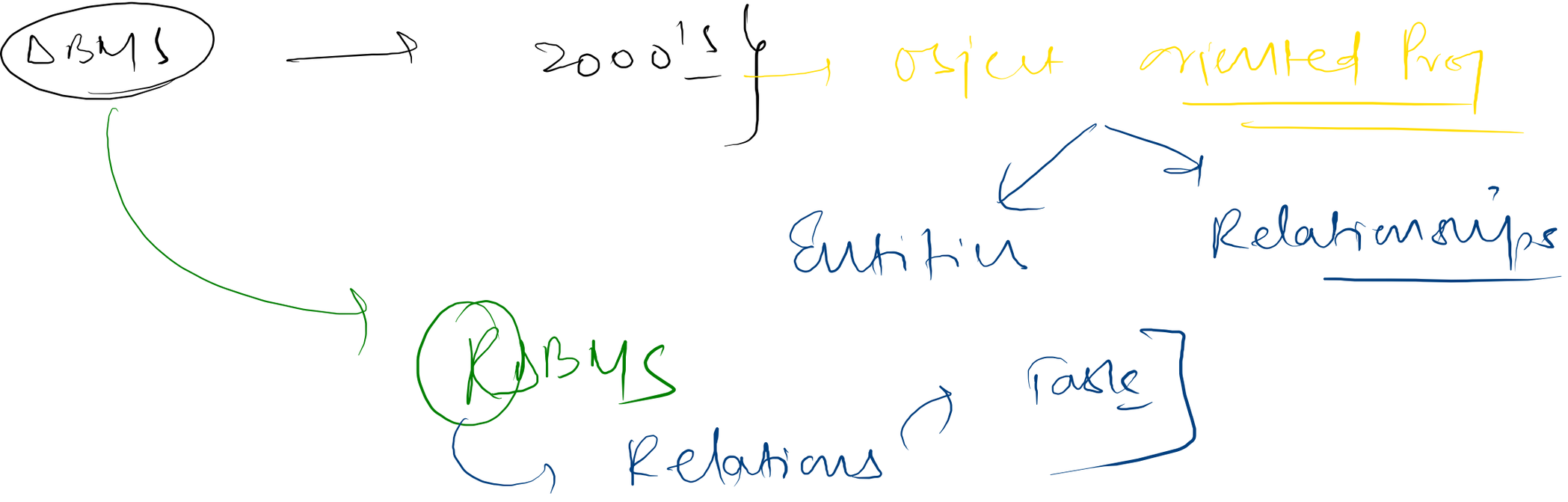
DBMS Auth / Authorisation Management System



SQL



Database


DBMS
2000's
Object oriented Prog
Entities
Relationships
RDBMS
Relations
Table

RDBMS

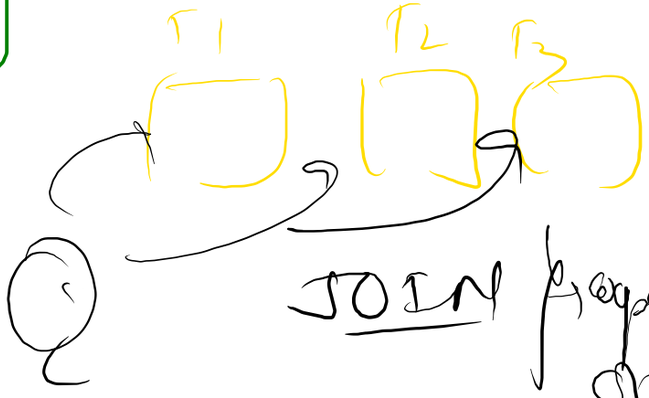
① Entity (Data) → Tables

② Relationships → FK

→ ACID, Transaction

Normalized

→ multiple small tables



JOIN (expensive operation)

RDBMS

① Transaction ✓

structured ✓ schema design

Less data ✓ Change frequently

RDBMS

① Scaling ↑ → slow very slow

② Unstructure X

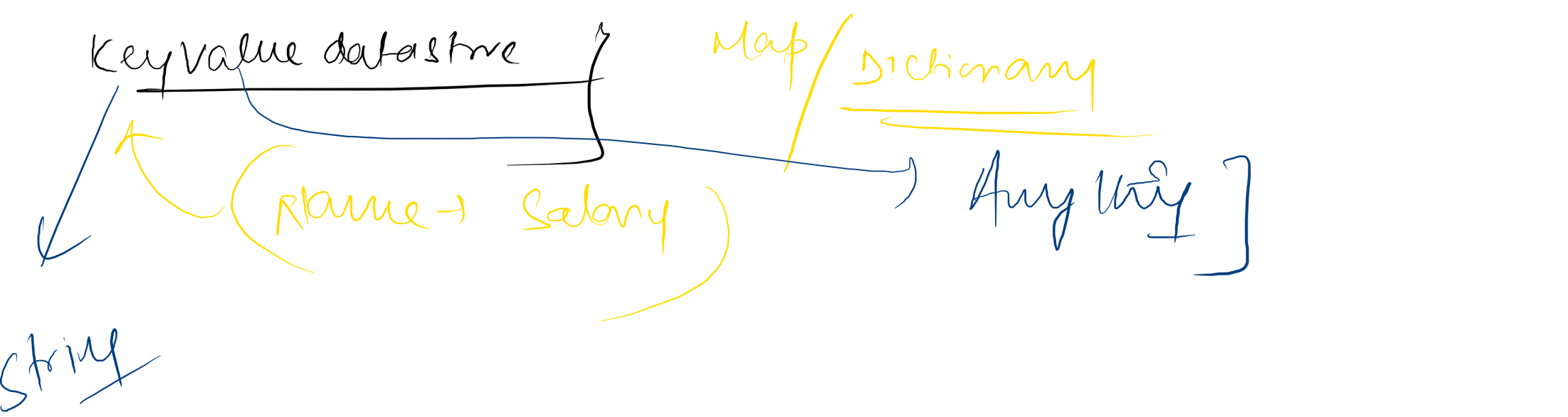
③ Structured → schema → changes very frequently

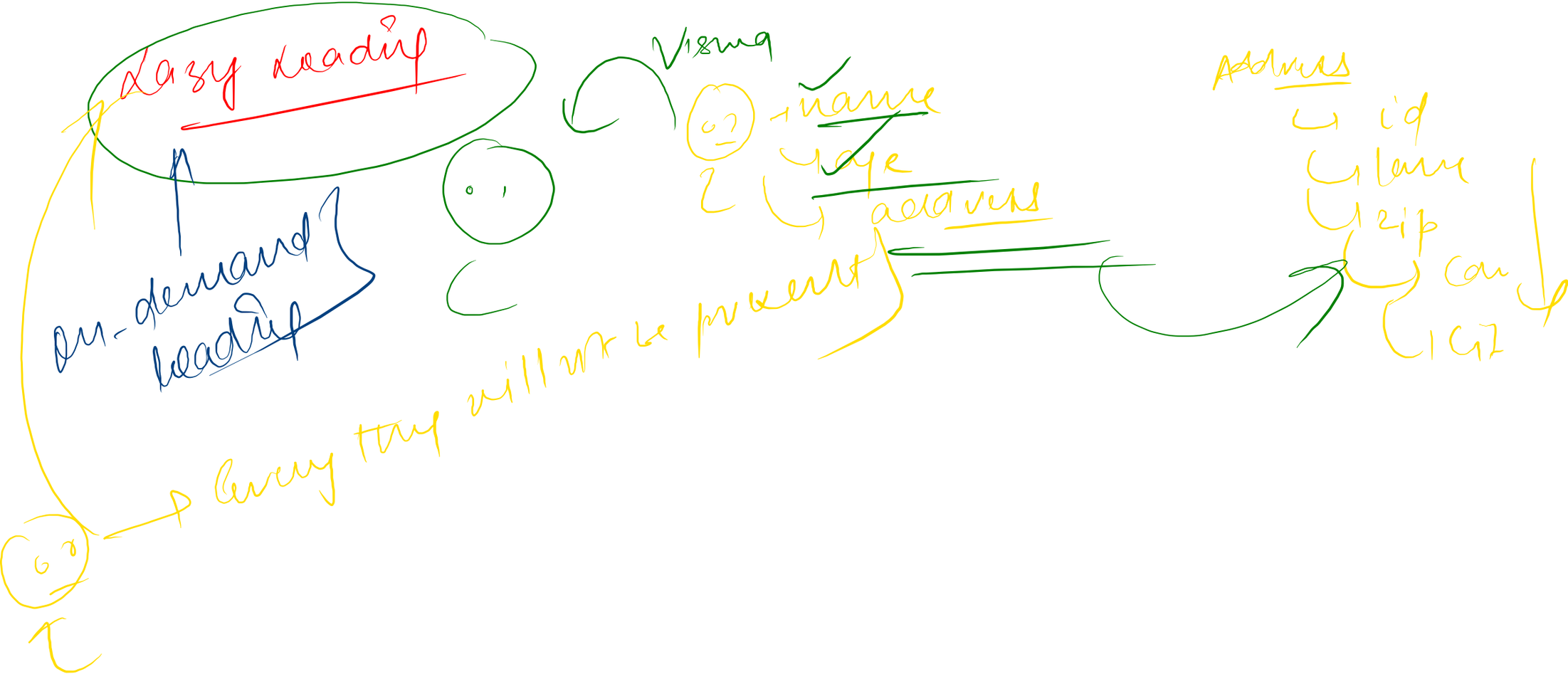
NoSQL } Not only sql

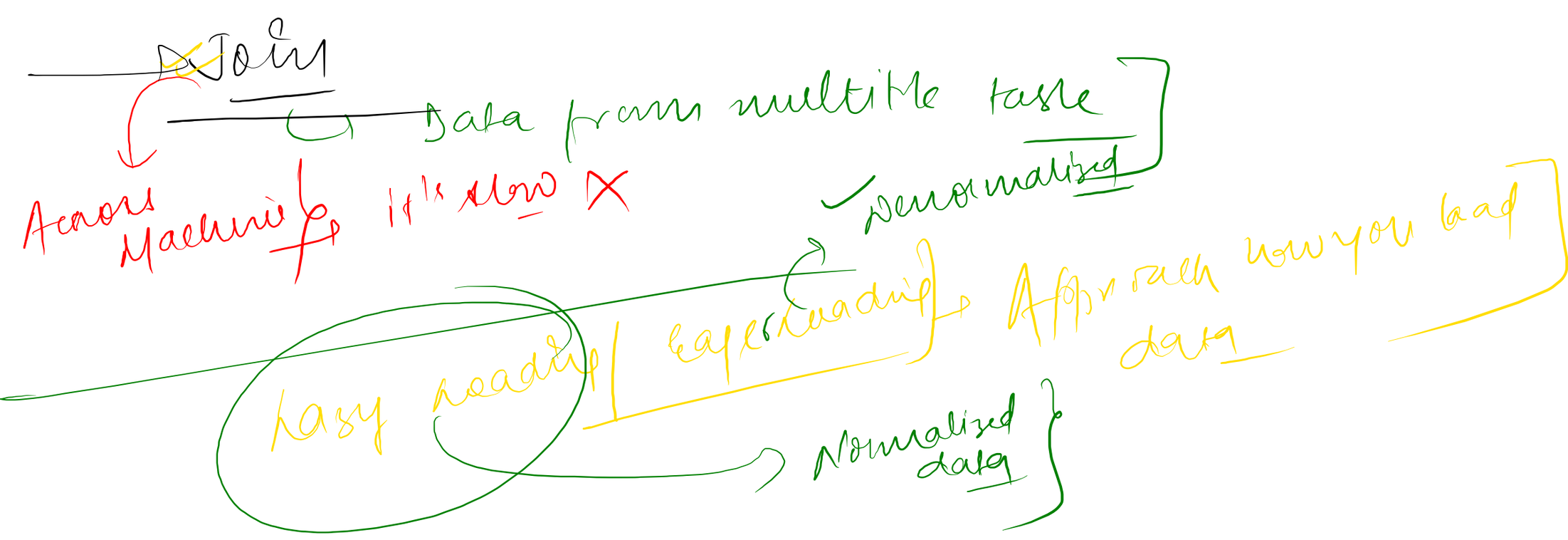
Natively scalable

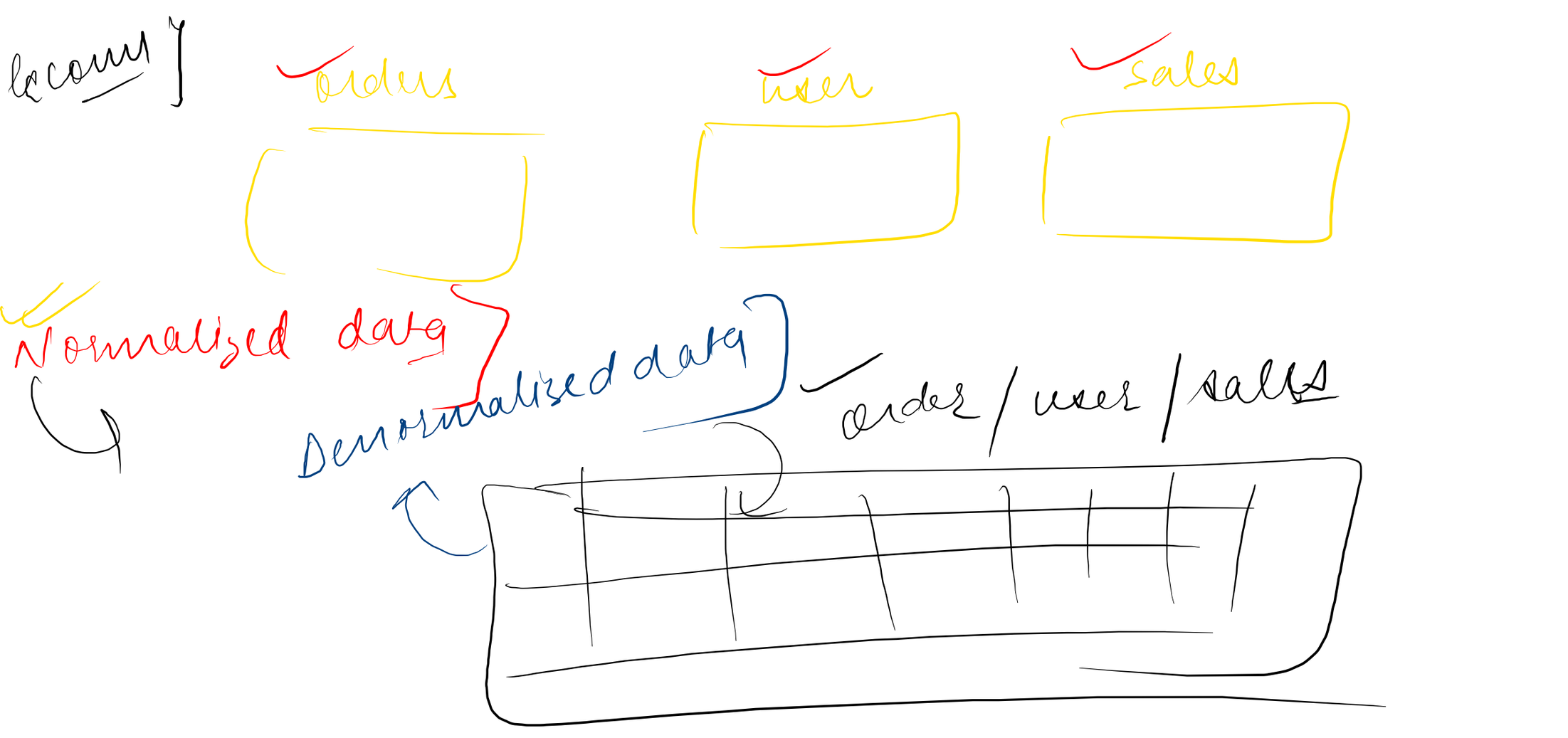
Denormalized data

BASE
① Basically Available
② soft state
③ eventually consistent


Key value datastore
Map / Dictionary
Name → Salary
Any key
String


Lazy loading
On-demand loading
Virat
name
age
address
Address
id
lane
zip
country
Every time will not be present


Join
Data from multiple tasks
Actors Machine
it's slow
Denormalized
Lazy loading
Eager loading
Approach how you load data
Normalized data


Orders
User
Sales
Normalized data
Denormalized data
order / user / sales

Document database

↓

data → documents

MongoDB

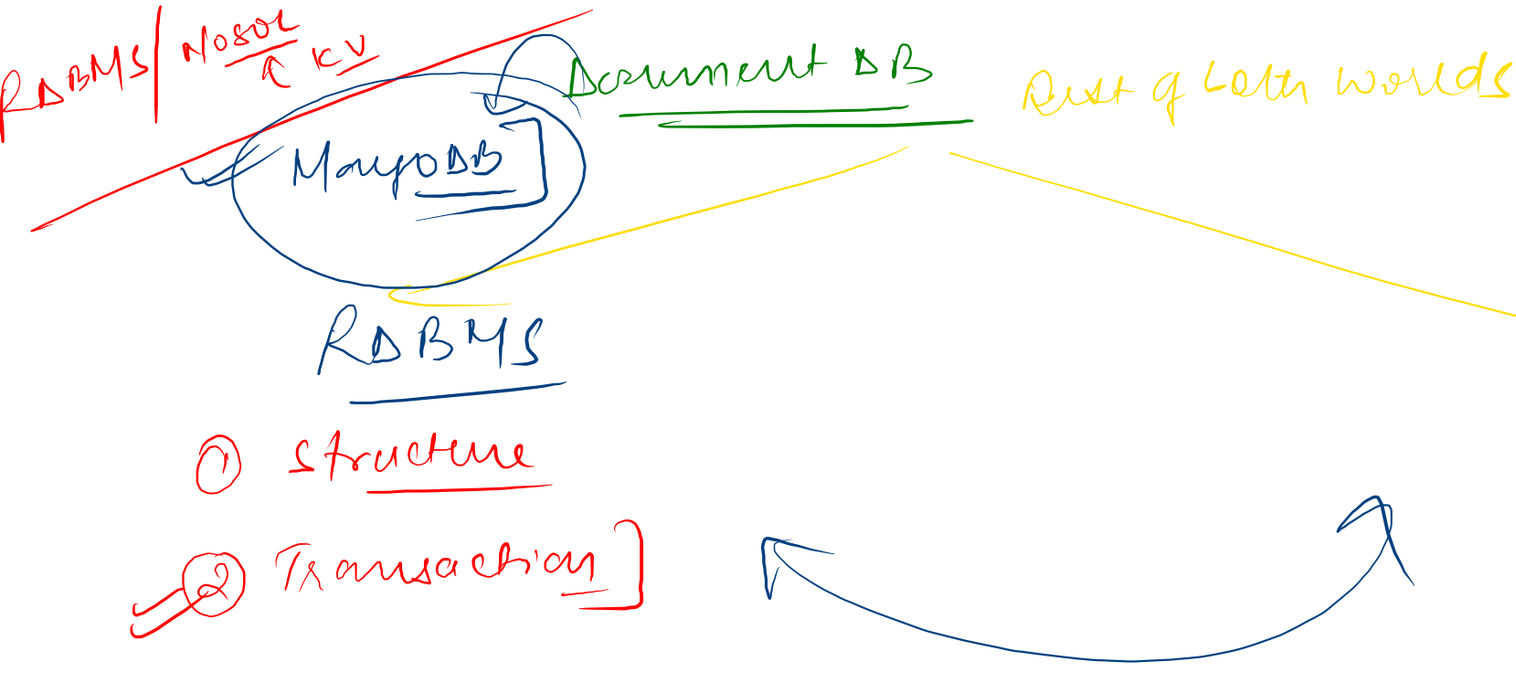
JSON

name
age
gender

↓

JSON document

```
{
  'name' : 'Vishwa'
  'age' : 99
  'gender' : 'M'
}
```


RDBMS / NoSQL
KV
Document DB
MongoDB
Rest of both worlds
RDBMS
1 Structure
2 Transaction
NOSQL
1 Dynamic schema
2 Scalable


Любой сад
Мир

How the data is stored?

RDBMS

Database

Table

Rows

columns

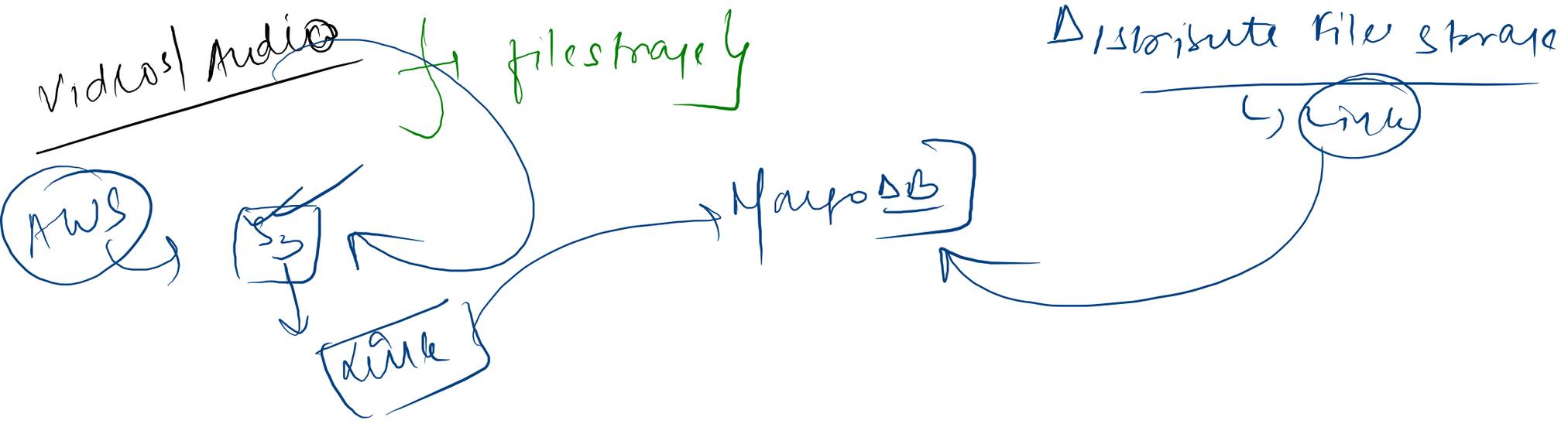
Document DB

Database

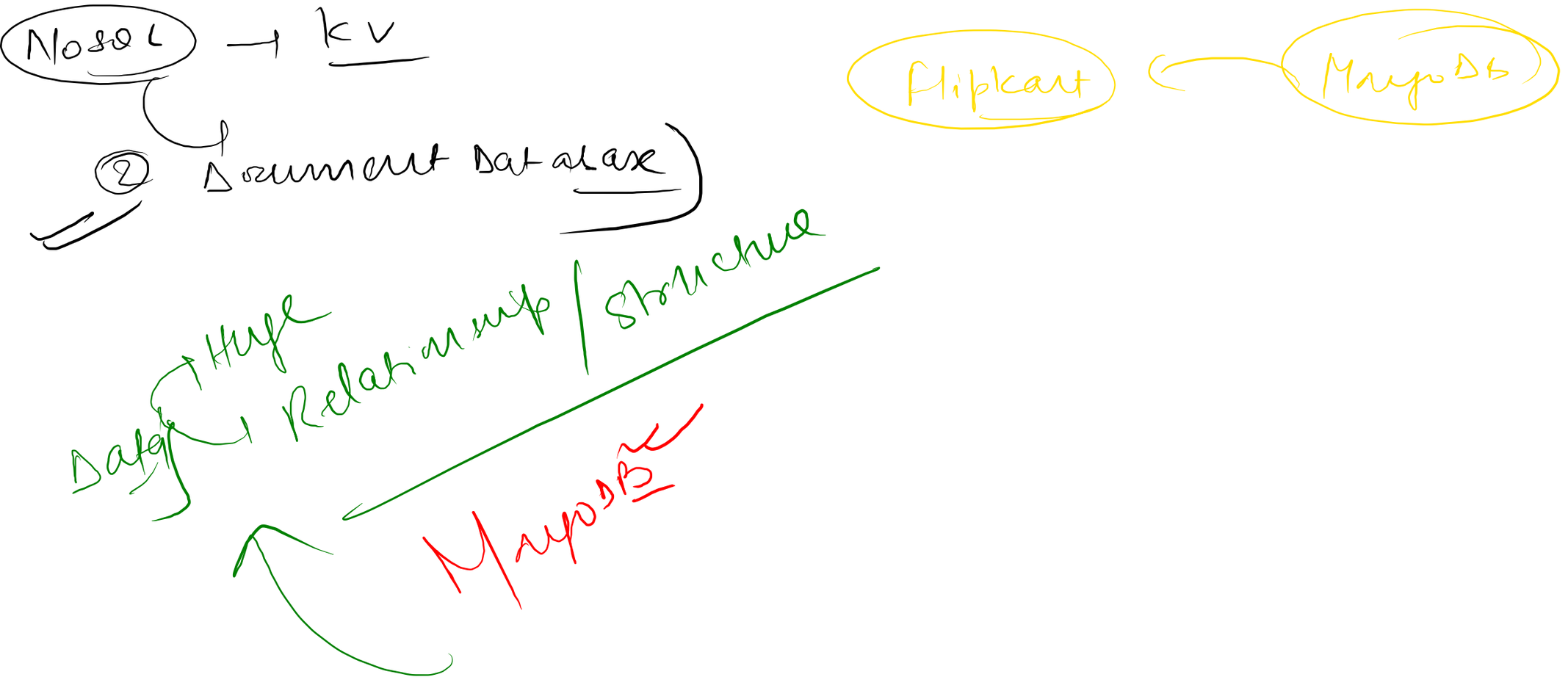
Collection

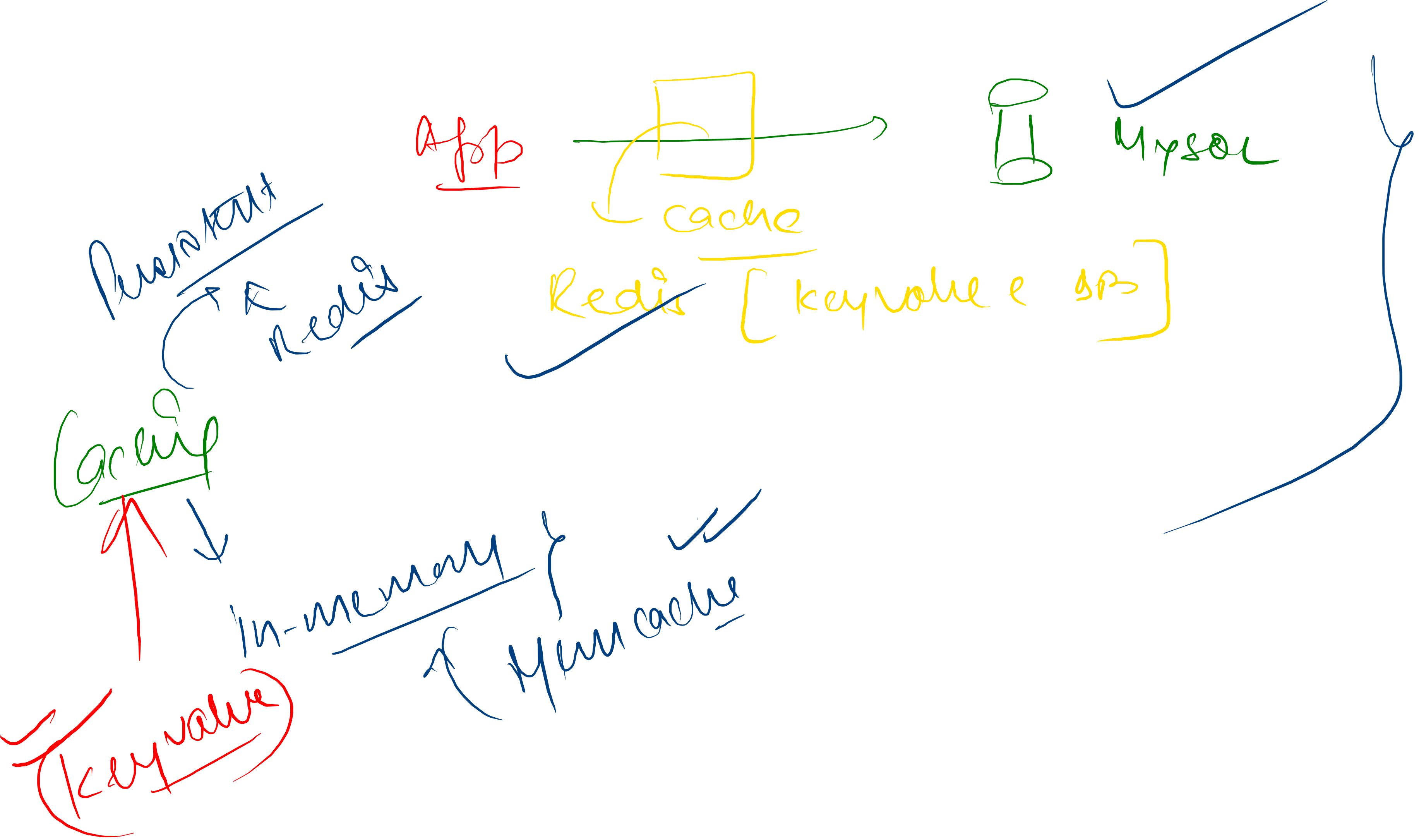
Documents

Fields


Videos/Audios
filestrage
Distributed file strage
AWS
S3
MongoDB
Linux
Linux

Company → Hybrid approach


NoSQL
kv
Document Database
Flipkart
MongoDB
Huge
Data
Relationship / Structure
MongoDB

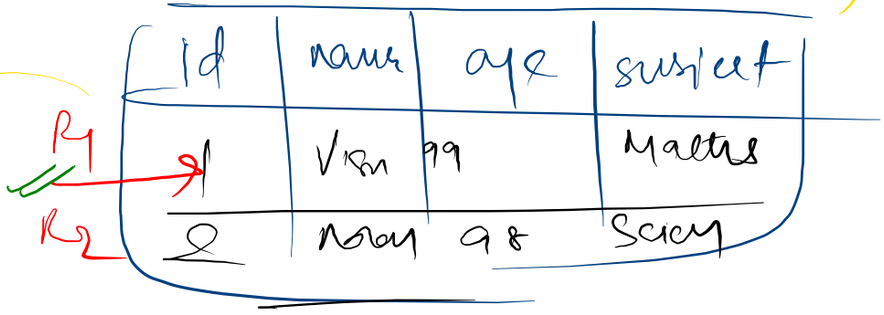

App
cache
MySQL
Redis [keyvalue e DB]
Persistent
Redis
Cache
in-memory
Memcache
keyvalue

③ Columnar Database :-

Database } RDBMS

Students

single array
↳ entire information about entity
↳ online App



Row based

↓

Rows are stored together

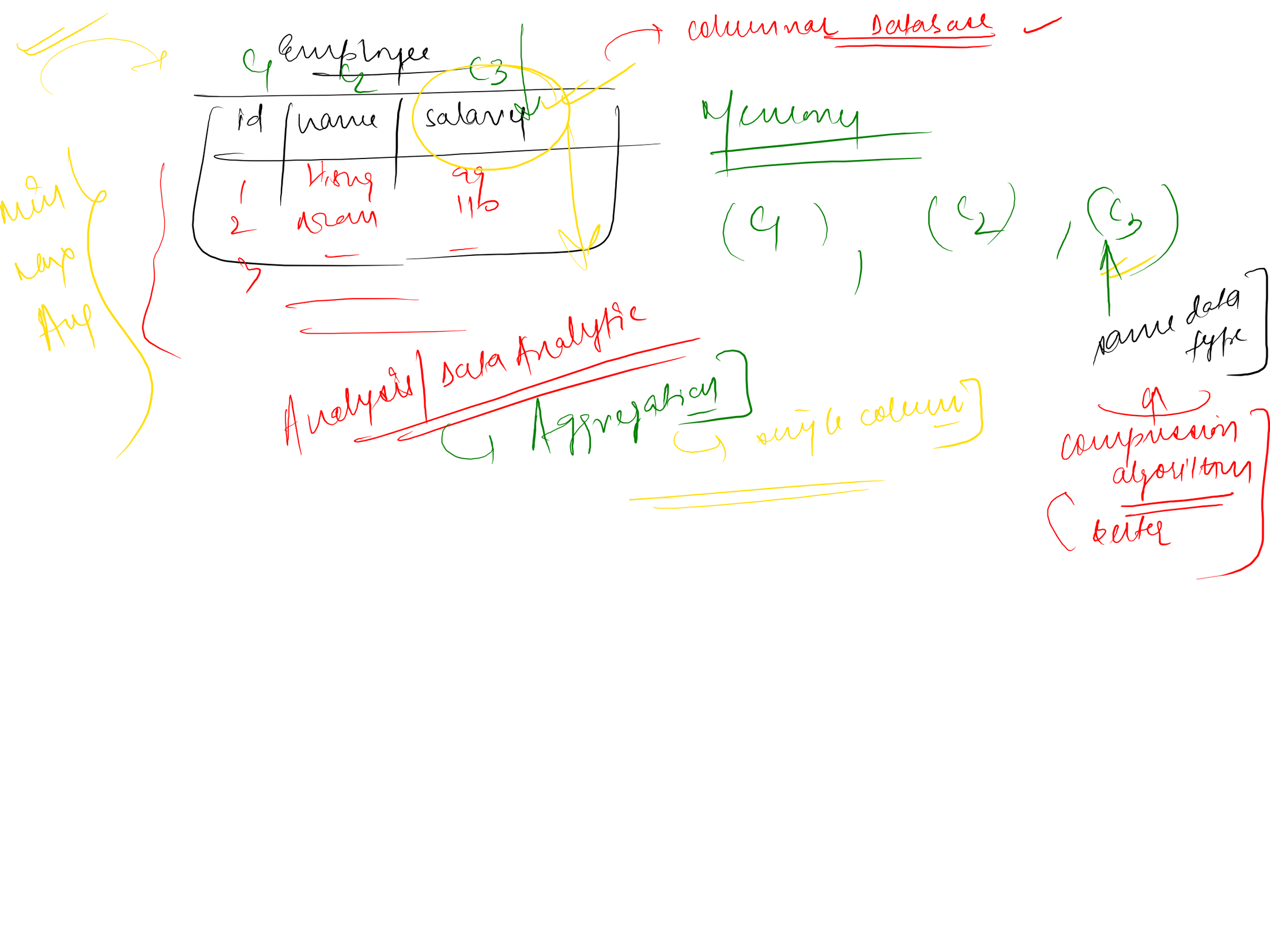
Memory

int string int

R_1 (1, Vishu, 99, Maths)

Different data types

R_2 (2, Ram, 98, Sciey)


Employee
C1
C2
C3
id
name
salary
1
Viking
99
2
Ocean
115
Memory
(C1)
(C2)
(C3)
name data type
columnar Database
Analysis | Data Analytics
Aggregation
single column
compression algorithms
better

Columnar DB

① Aggregation fast ↑

② Better compaction → store more data in less space

Cassandra

read/write ↑

scale

Walmart

MCSE

Multi channel Sourcing Engine

Walmart

Order

Date item will arrive

Arrival

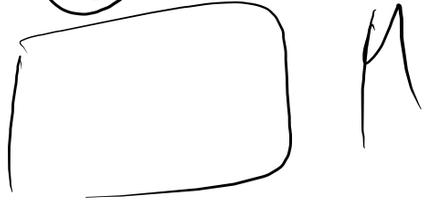
Distributers

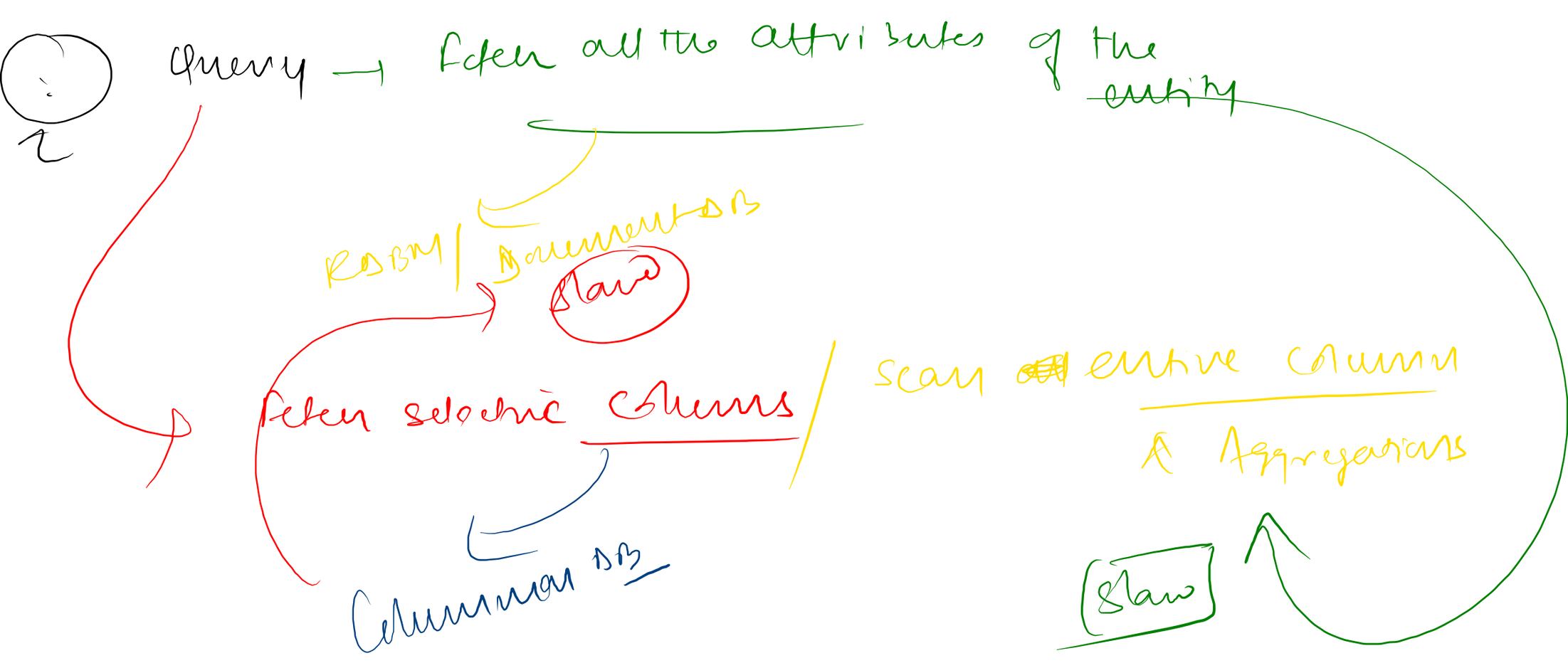
530 possible solutions

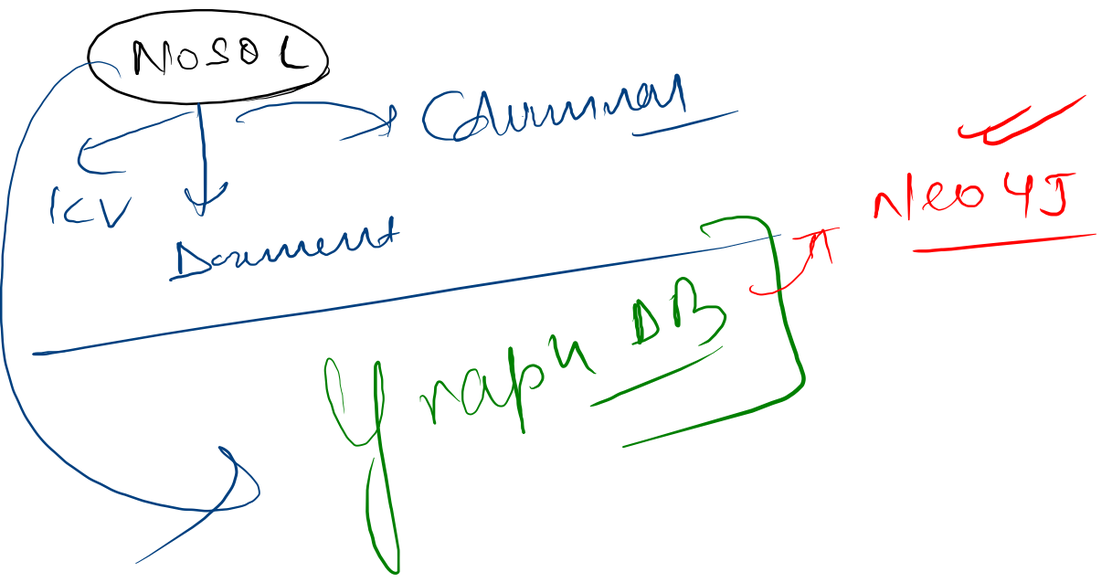
pick 1 best solution

Cassandra

DB




Query → Fetch all the attributes of the entity
Row/Document
Slow
Fetch selective columns
Column
Scan entire column
& Aggregations
Slow

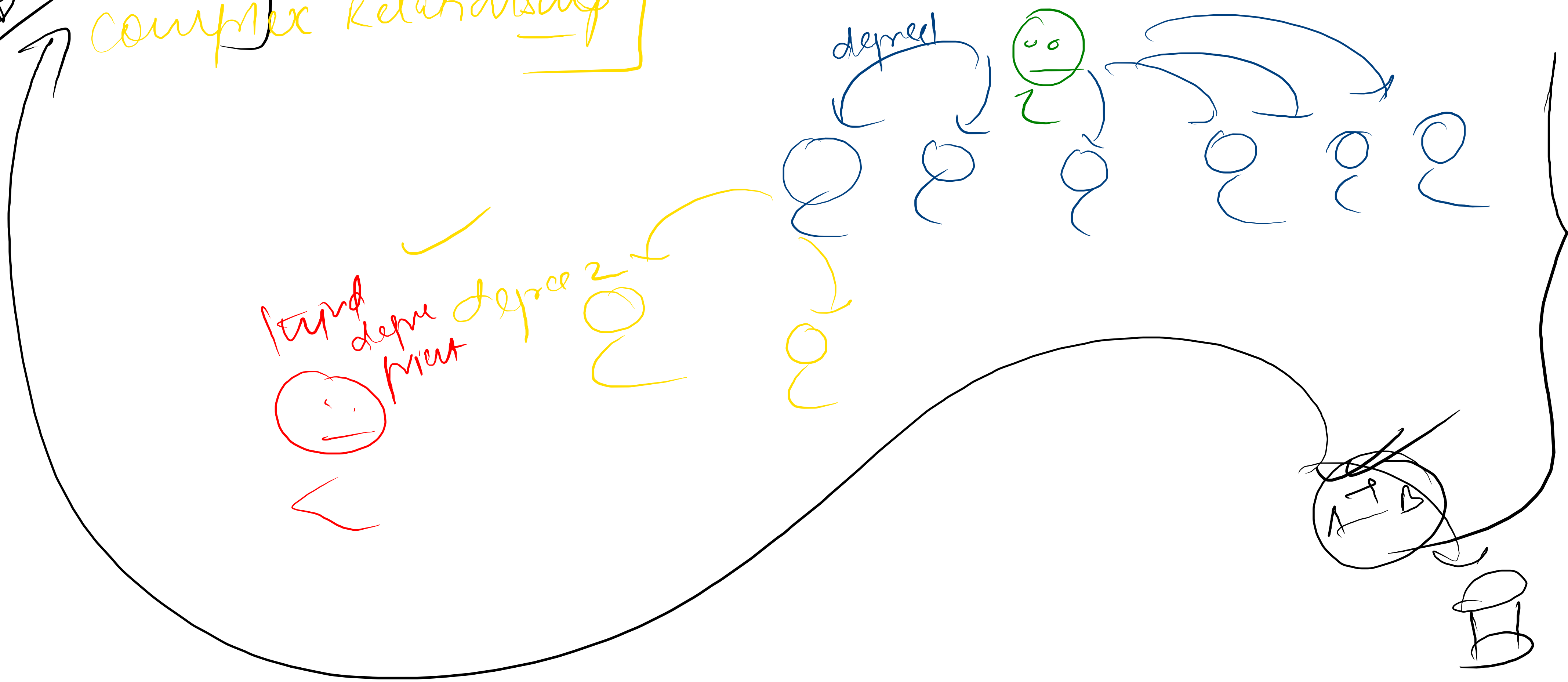

NoSQL
Column
KV
Document
Graph DB
Neo4J

Modern application

Graph DB

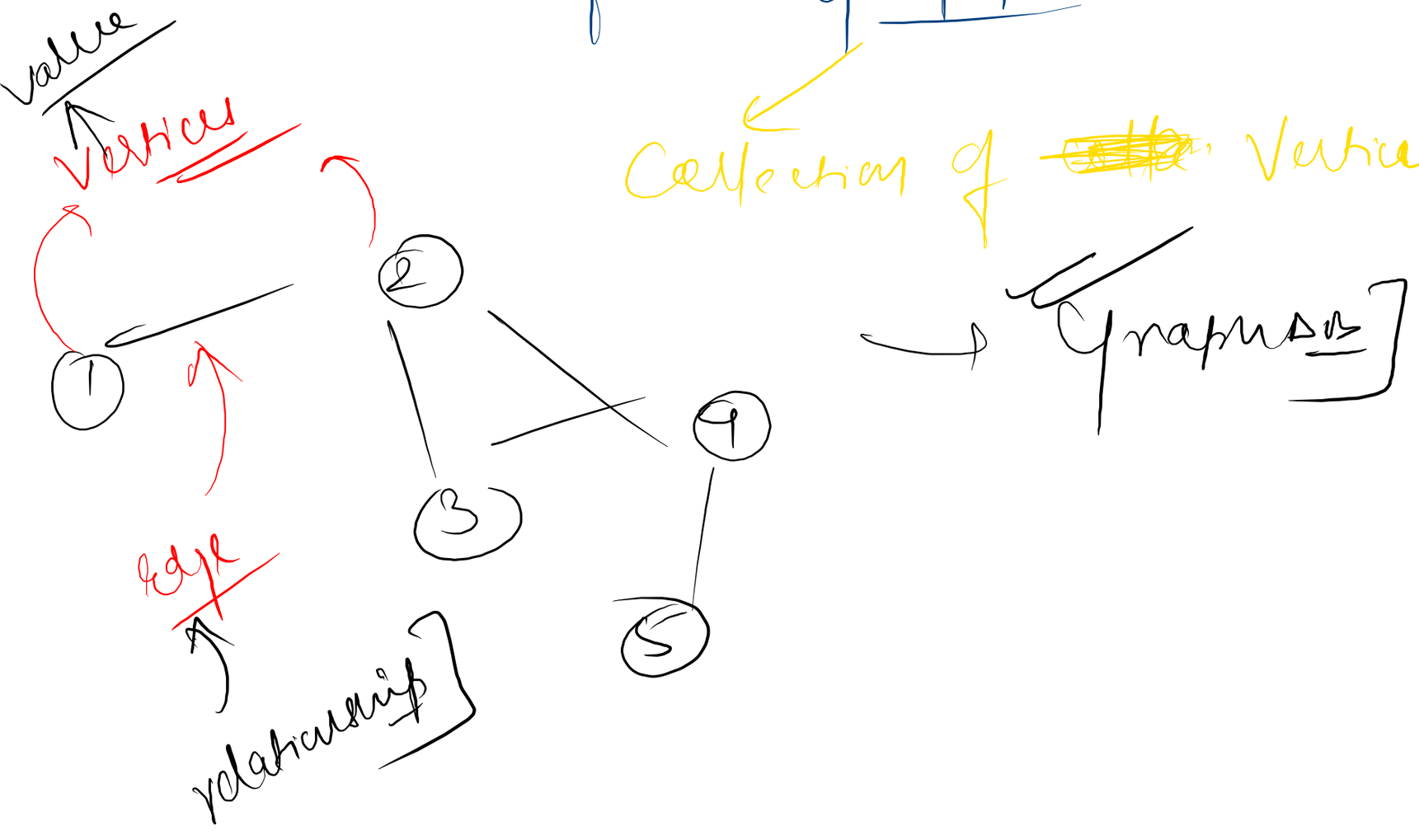
complex Relationship

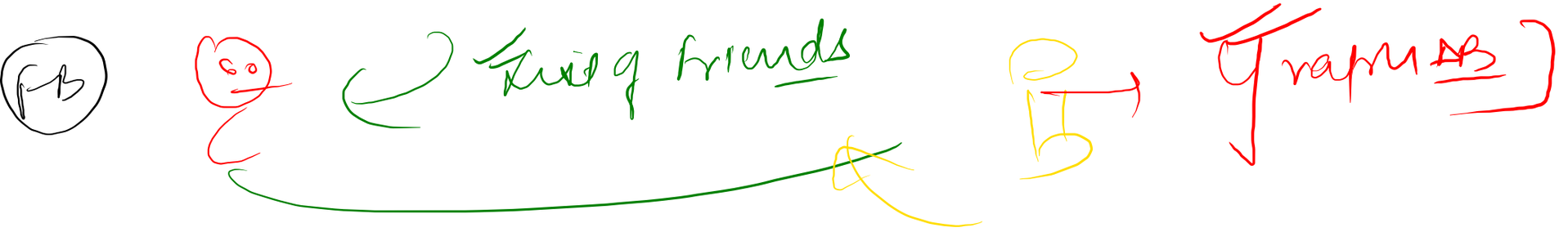
LinkedIn

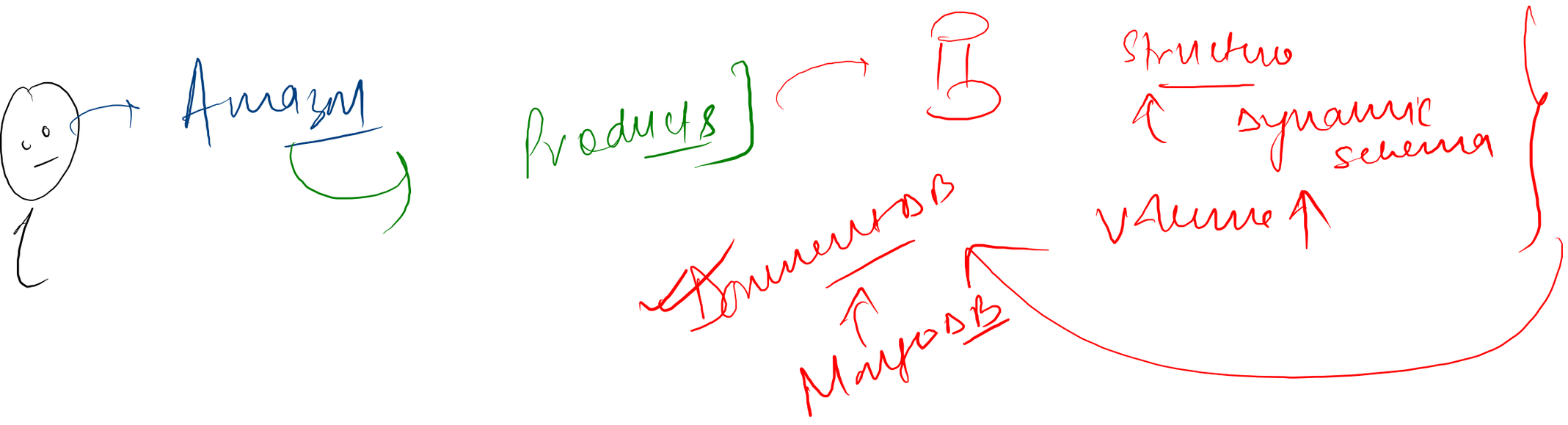


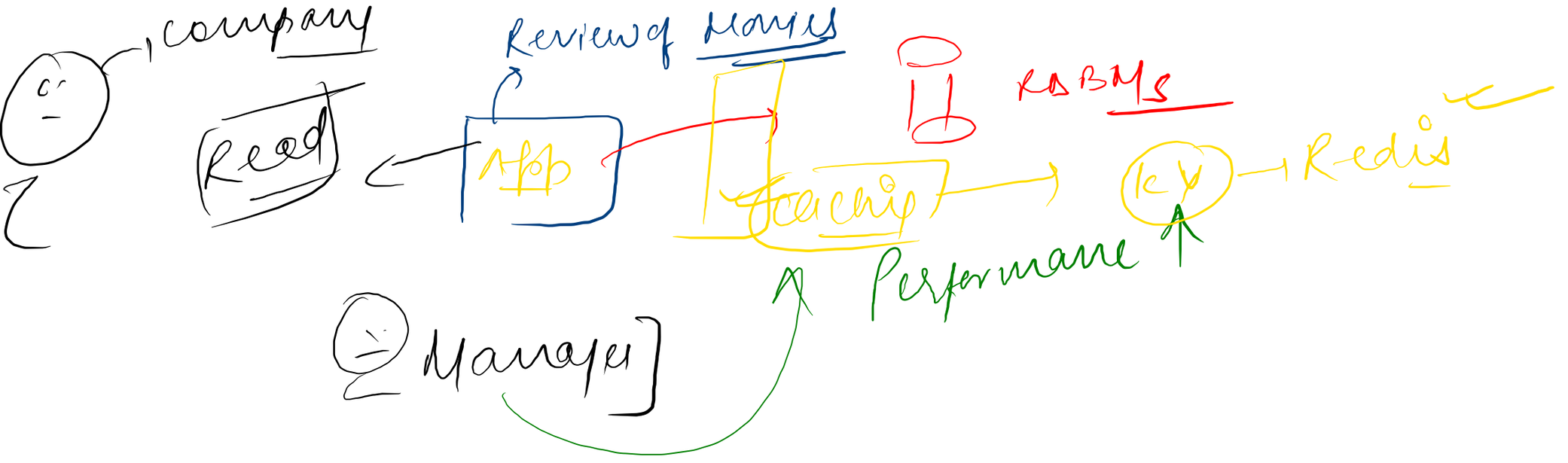
Store data in the forms Graph

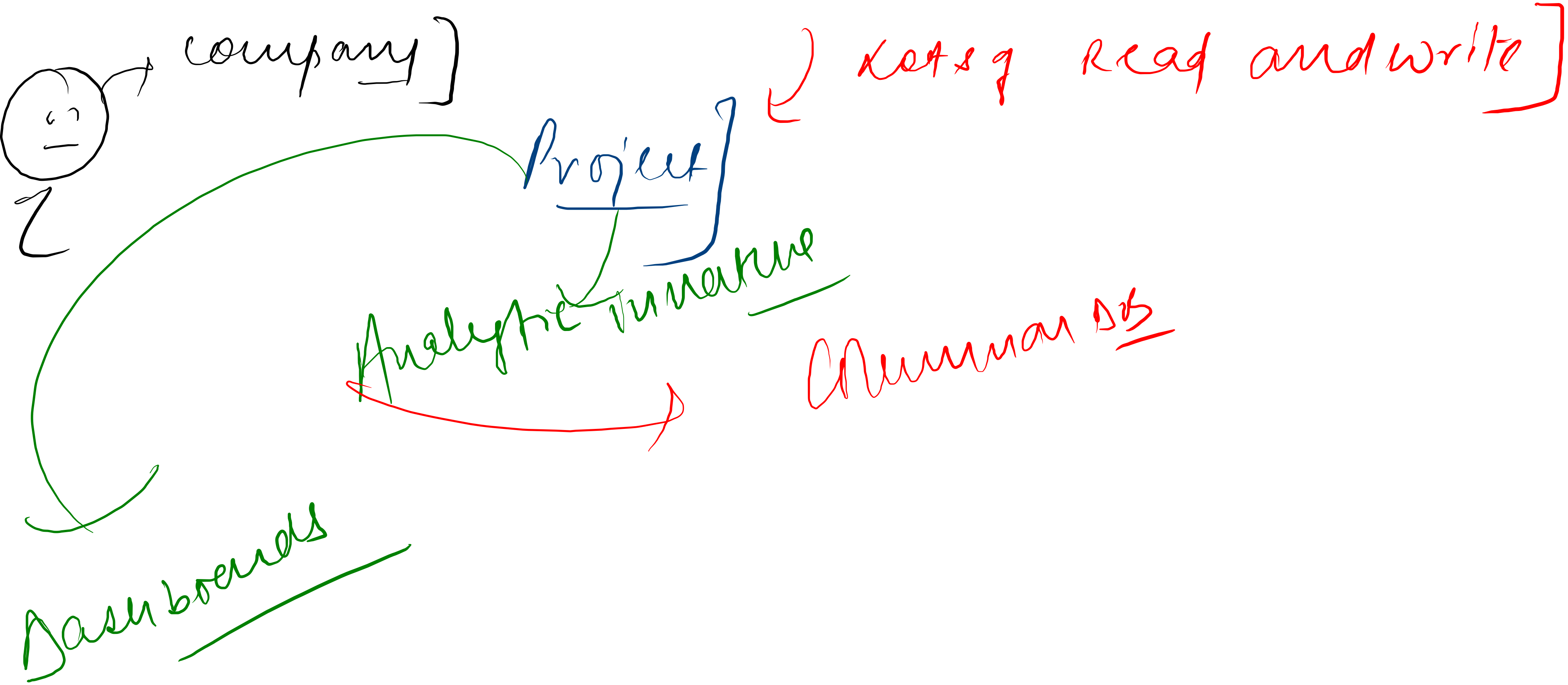
Collection of Vertices and edges.

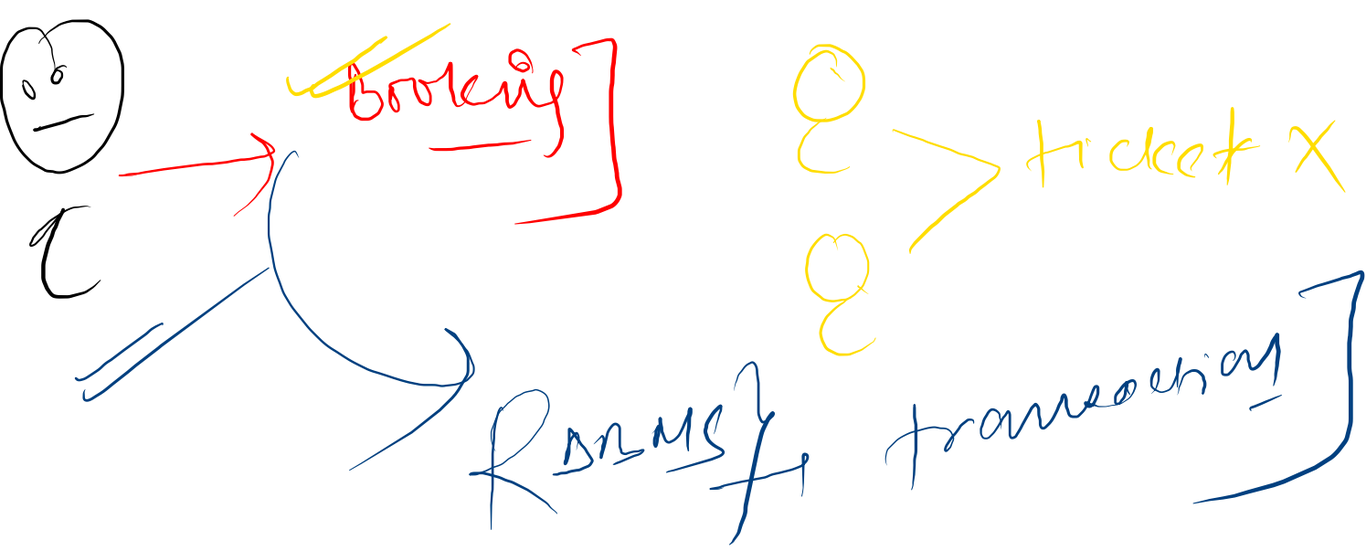


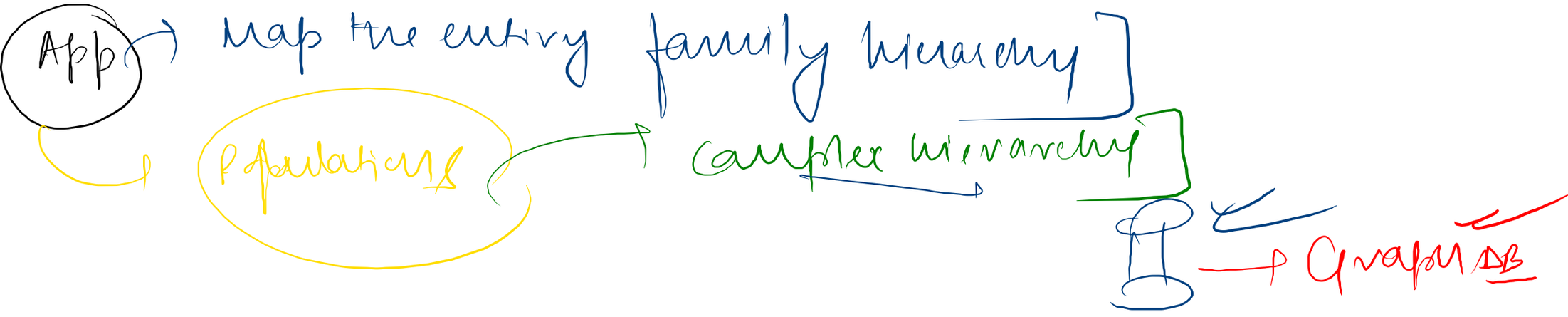

FB
Fwing friends
Trapped


Amazon
Products
Structure
Dynamic
Values
Functions
Methods


company
Read
Review of Movies
Movies
RDBMS
Redis
Performance
Manager

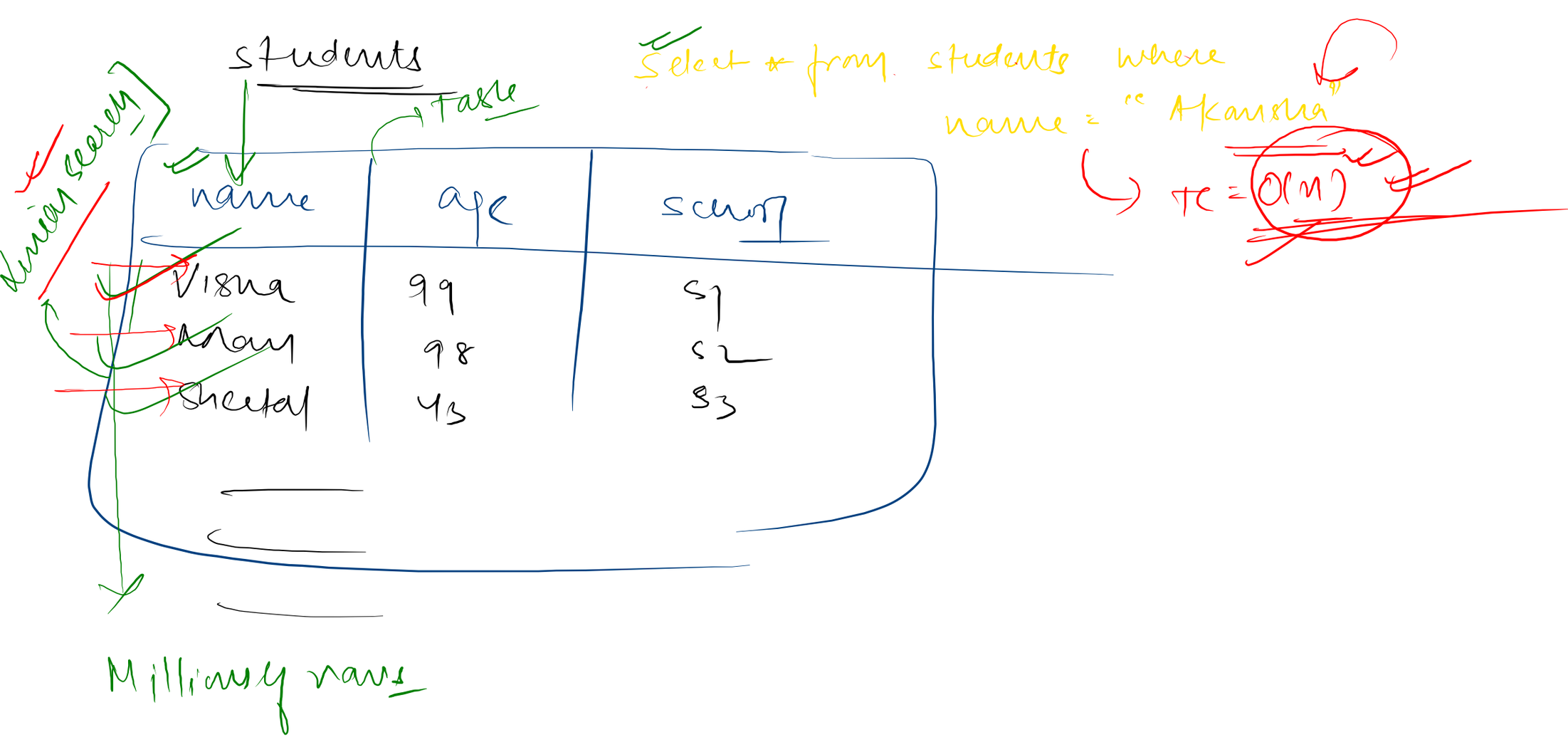

company
Project
read and write
Dashboards


Booking
ticket X
RDBMS, transaction

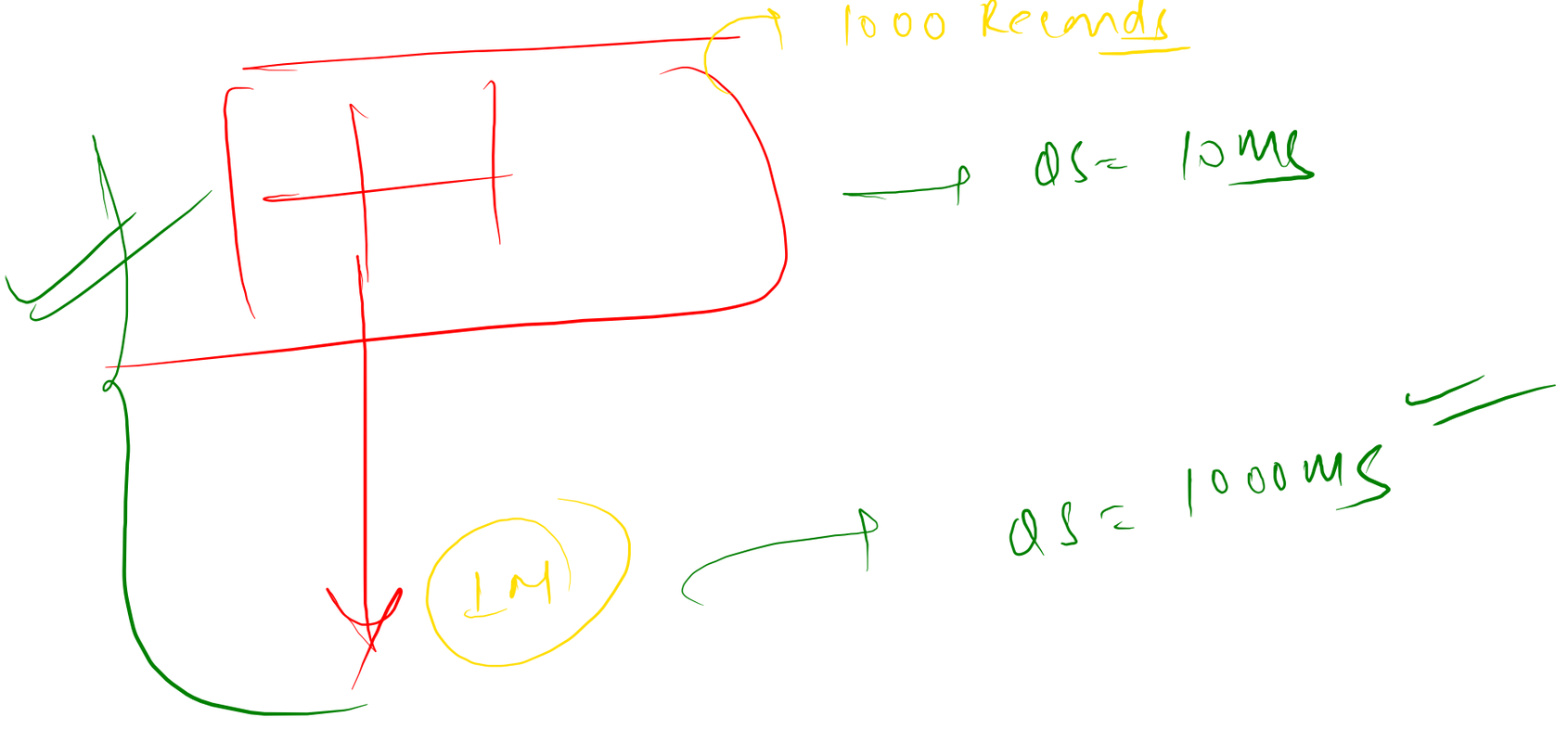

App
Map the entry family hierarchy
Population
complex hierarchy
Graph DB

Database Optimization :- [Improve the query speed]

1. Indexing → Read queries performance


students
Table
Select * from students where
name = "Akanksha"
TC = O(n)
Linear search
name
age
score
Vishu
99
91
Akash
98
92
Sheetal
93
93
Millions rows

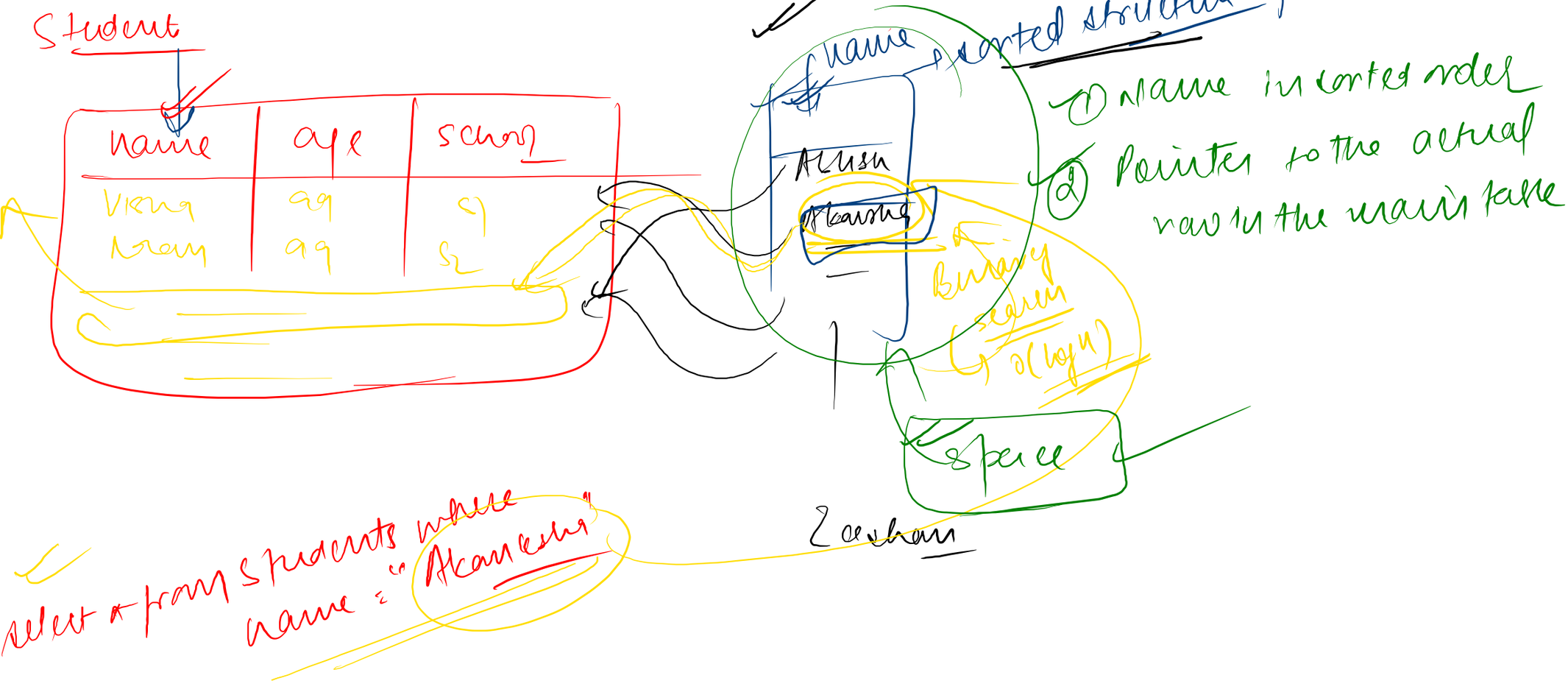
Start



Select * from students where name = < >

SS + B trees

Balanced trees

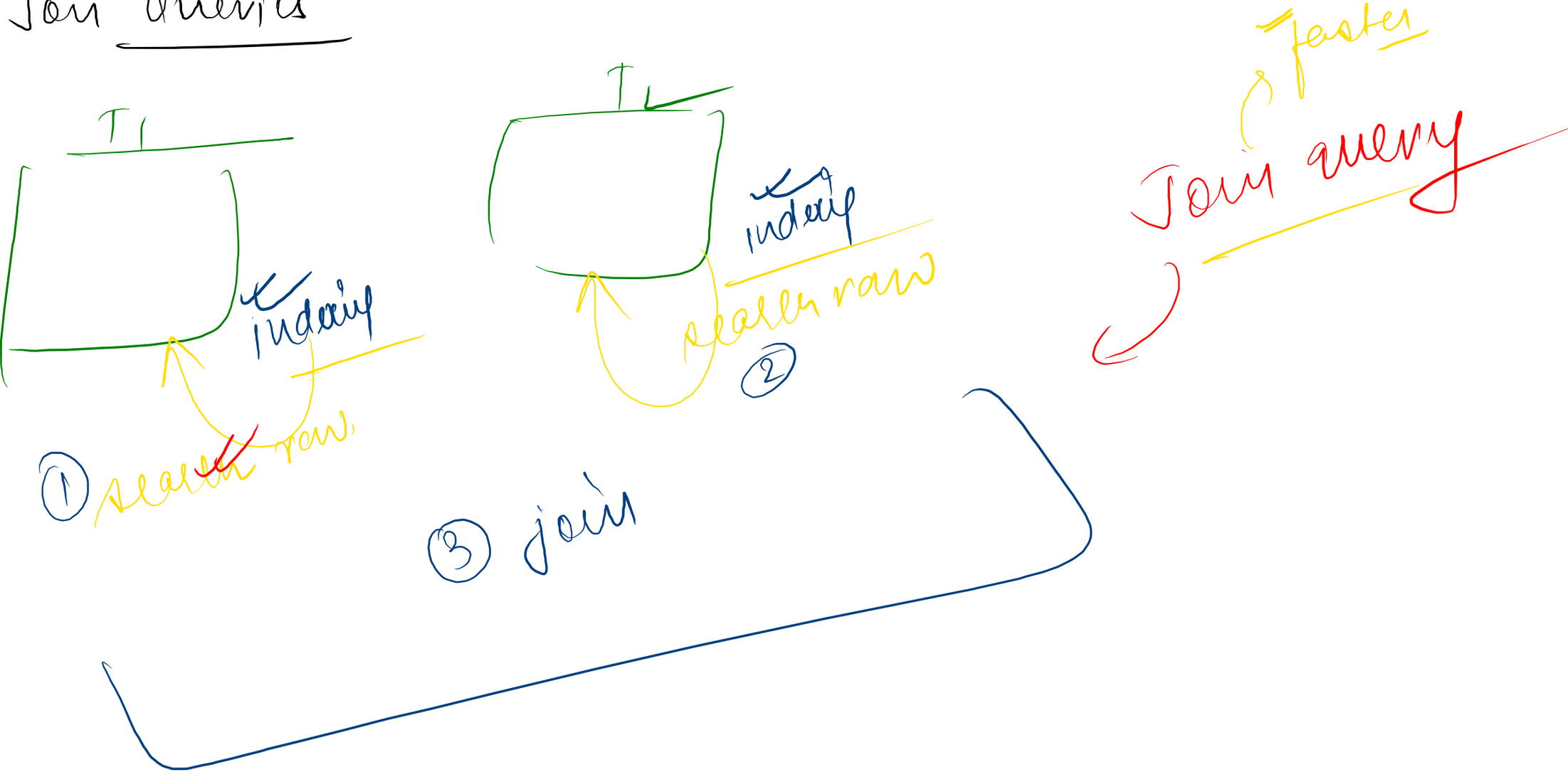


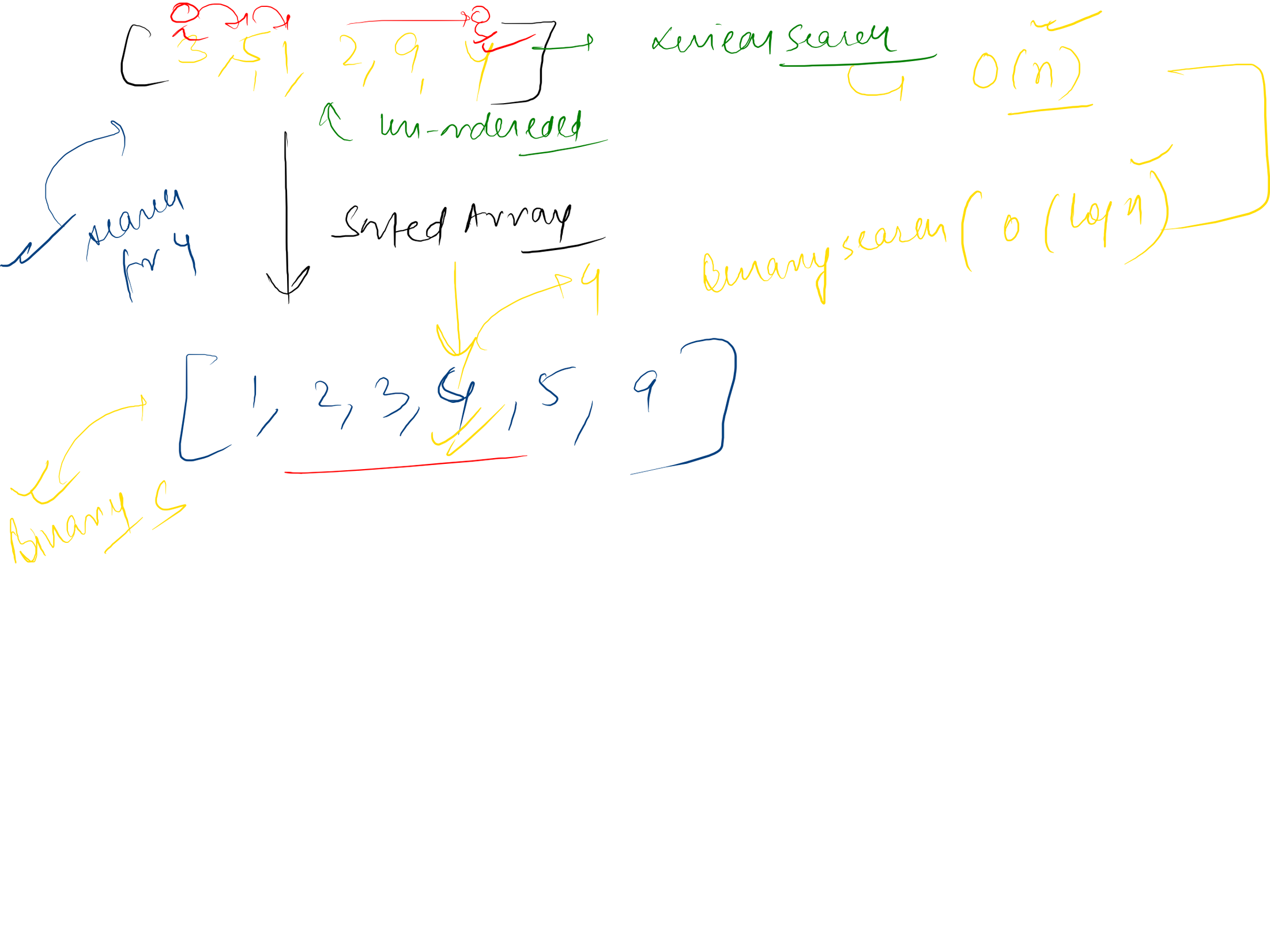
① name in sorted order

② Pointer to the actual row in the main table

Select * from students where name = "Akash"

Join Queries




[3, 5, 1, 2, 9, 4]
Linear search
O(n)
Un-ordered
search for 4
Sorted Array
Binary search (O(log n))
4
[1, 2, 3, 4, 5, 9]
Binary S

Indexing → Read performance query time

Disadvantage

① A lot of space ↑

② → write heavy

Unnecessary

Indexing → Improve the query performance

Indexing → Reduce the write speed

write → Restructuring and updating of B trees

[extra time]

Indexing → ① [DB] → Read heavy

② identify the columns mostly used in where clause

indexing only on these columns

Alloed

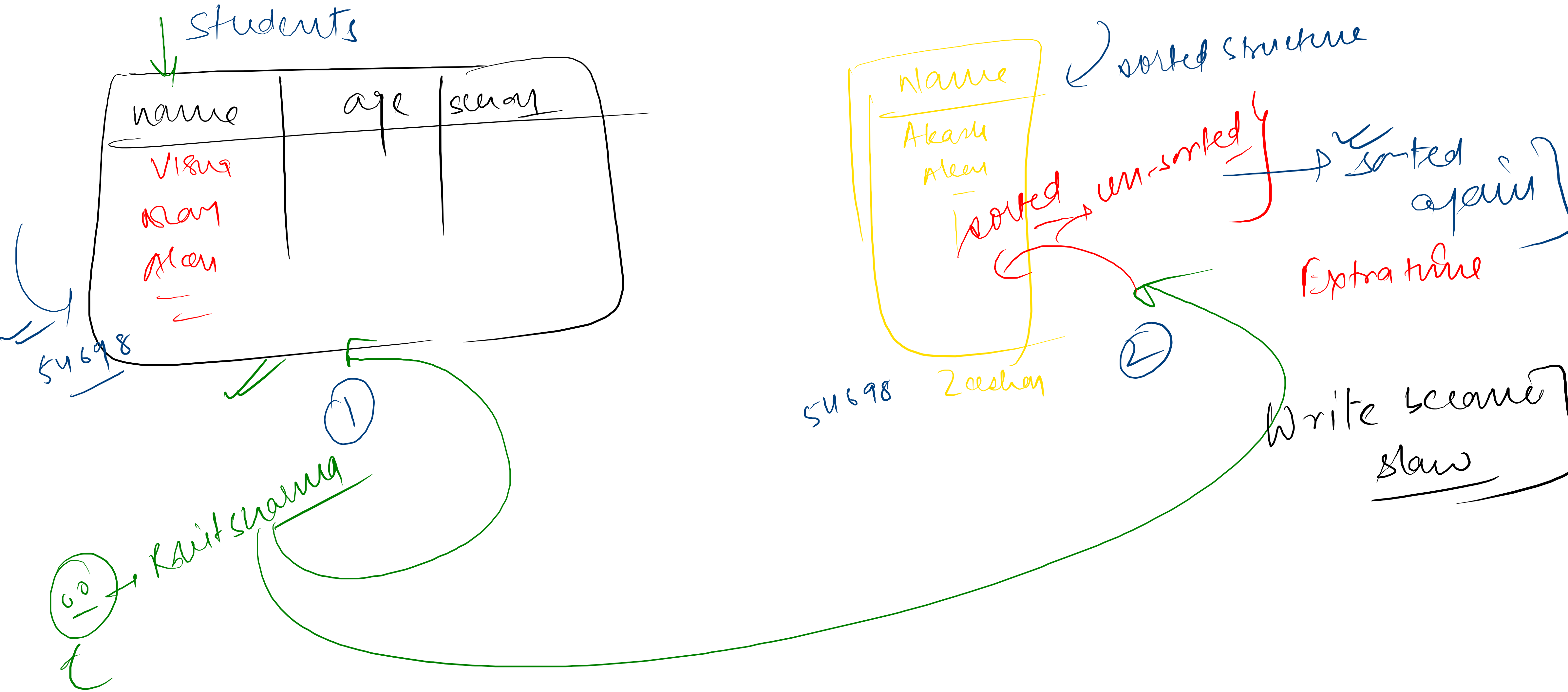
NW → MongoDB → table | collection

indexing on all fields → extra space → $$$ → slow queries

8:17 → 8:25 PM

indexing can't be done everywhere

write heavy → indexing → costly



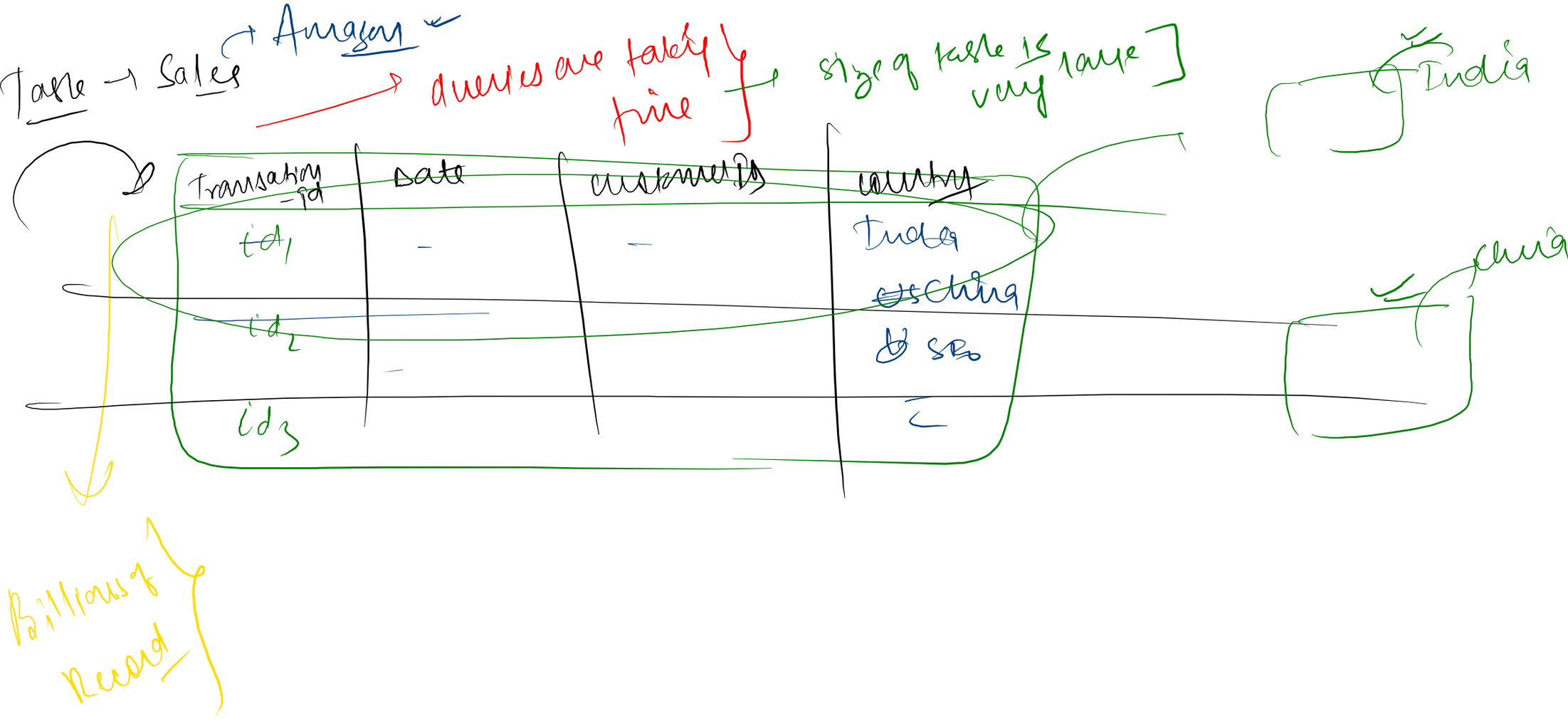
students
name
age
salary
Vishal
Aslam
Aleem
54698
name
Akash
Aleem
Zeeshan
54698
sorted structure
sorted
un-sorted
sorted again
Extra time
Write become slow

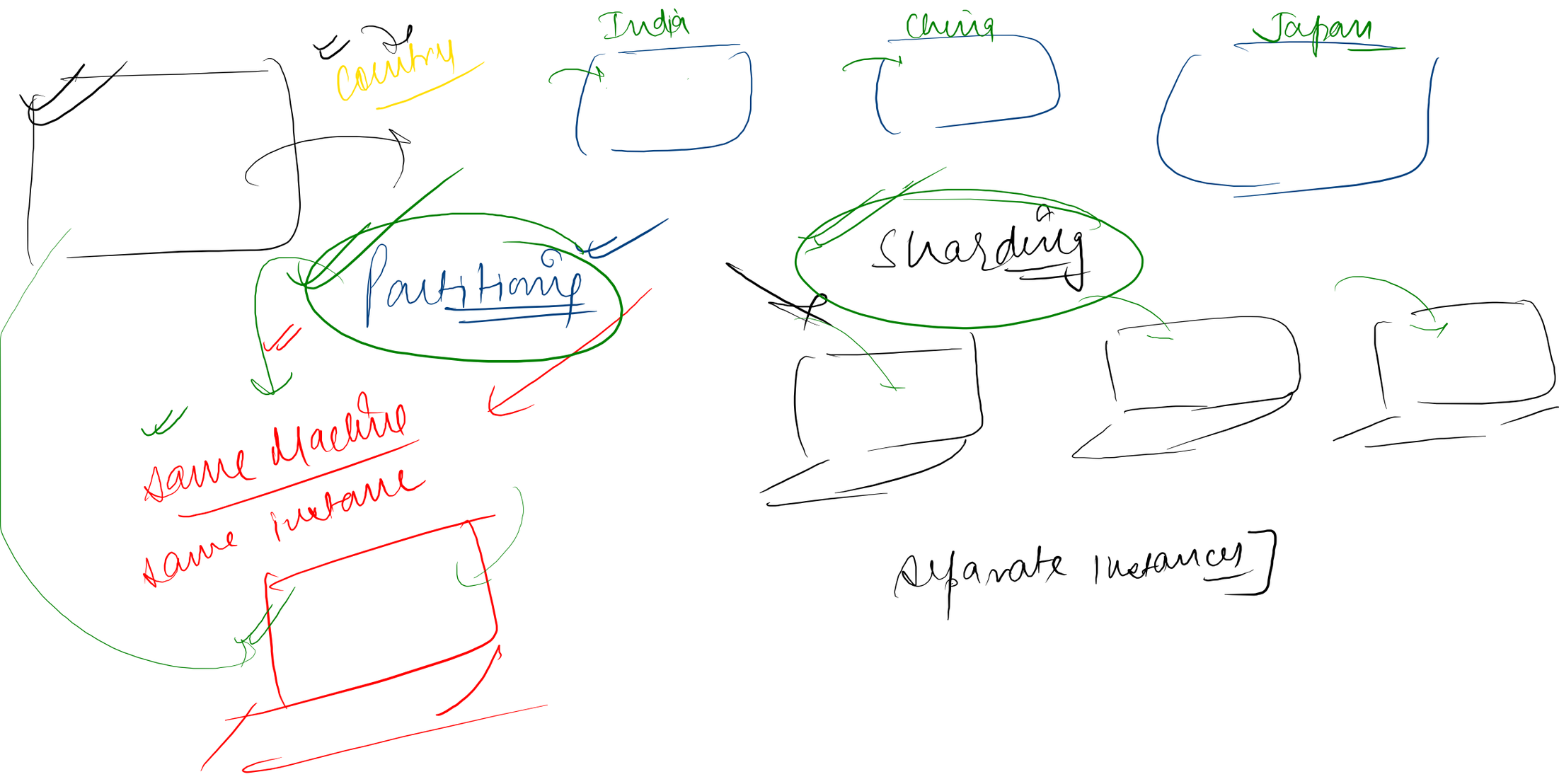
Optimizing performance

Partitioning | sharding

Table → Sales

Amazon




Country
India
China
Japan
Partitioning
Sharding
Same Machine
Same instance
Separate instances

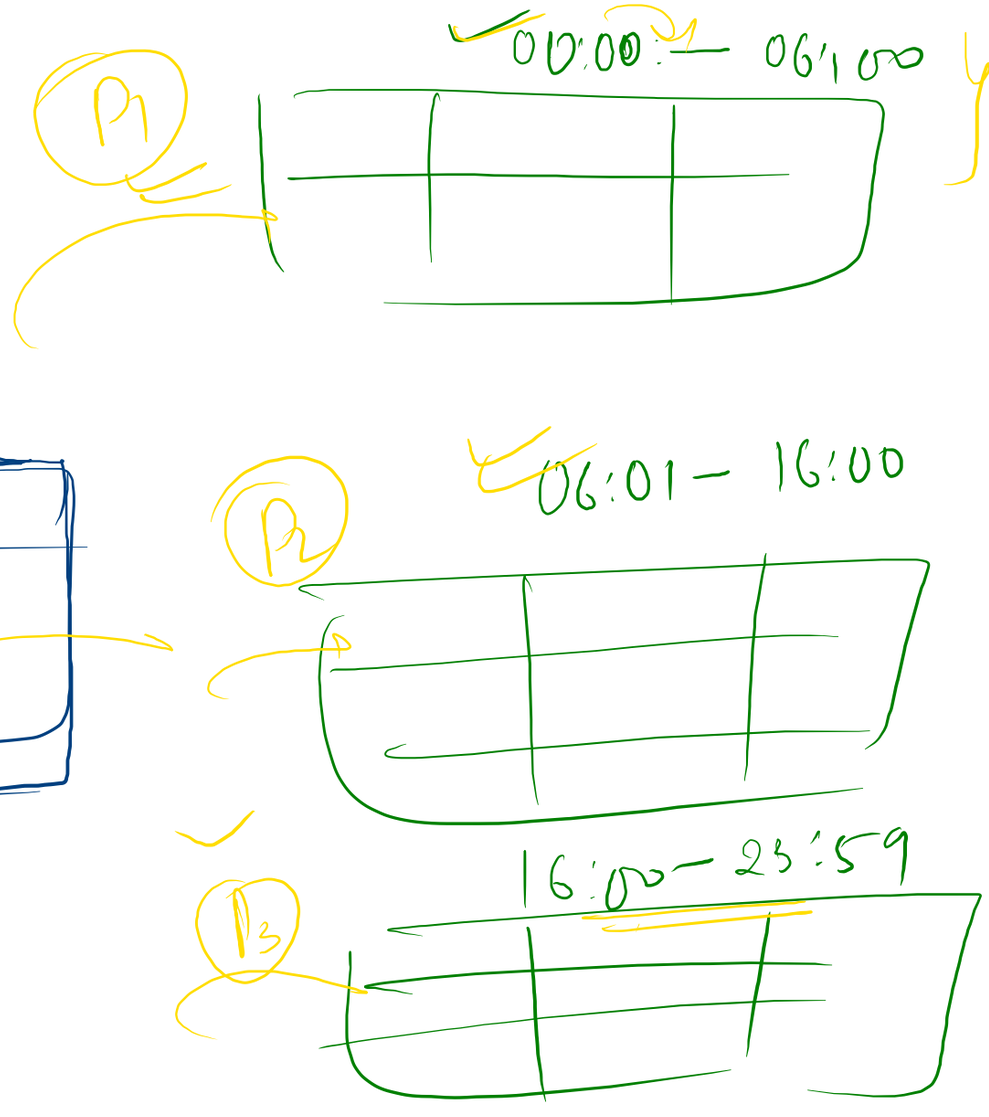
Partitioning / sharding :-

① Range Based partitioning

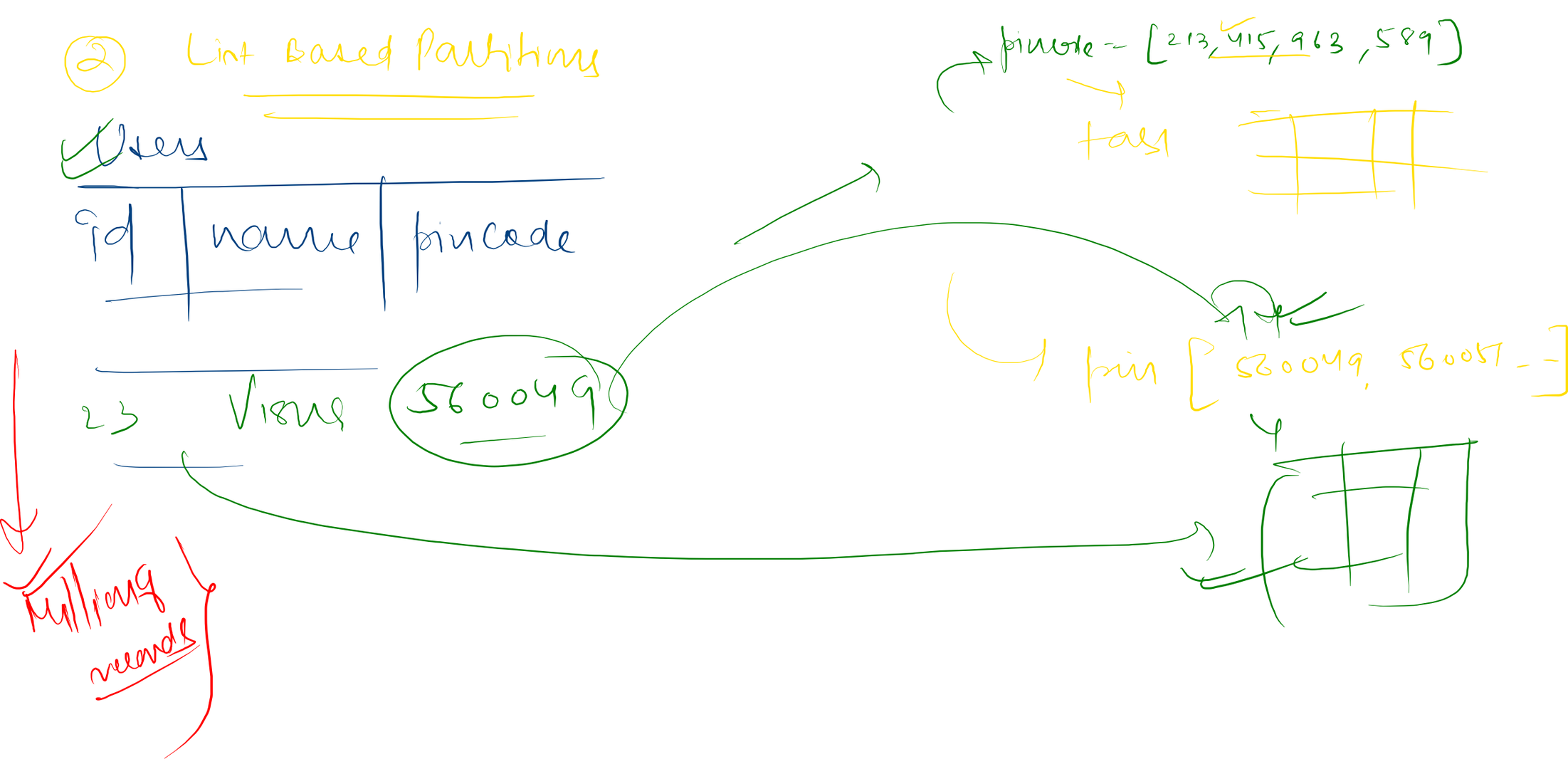
Analog

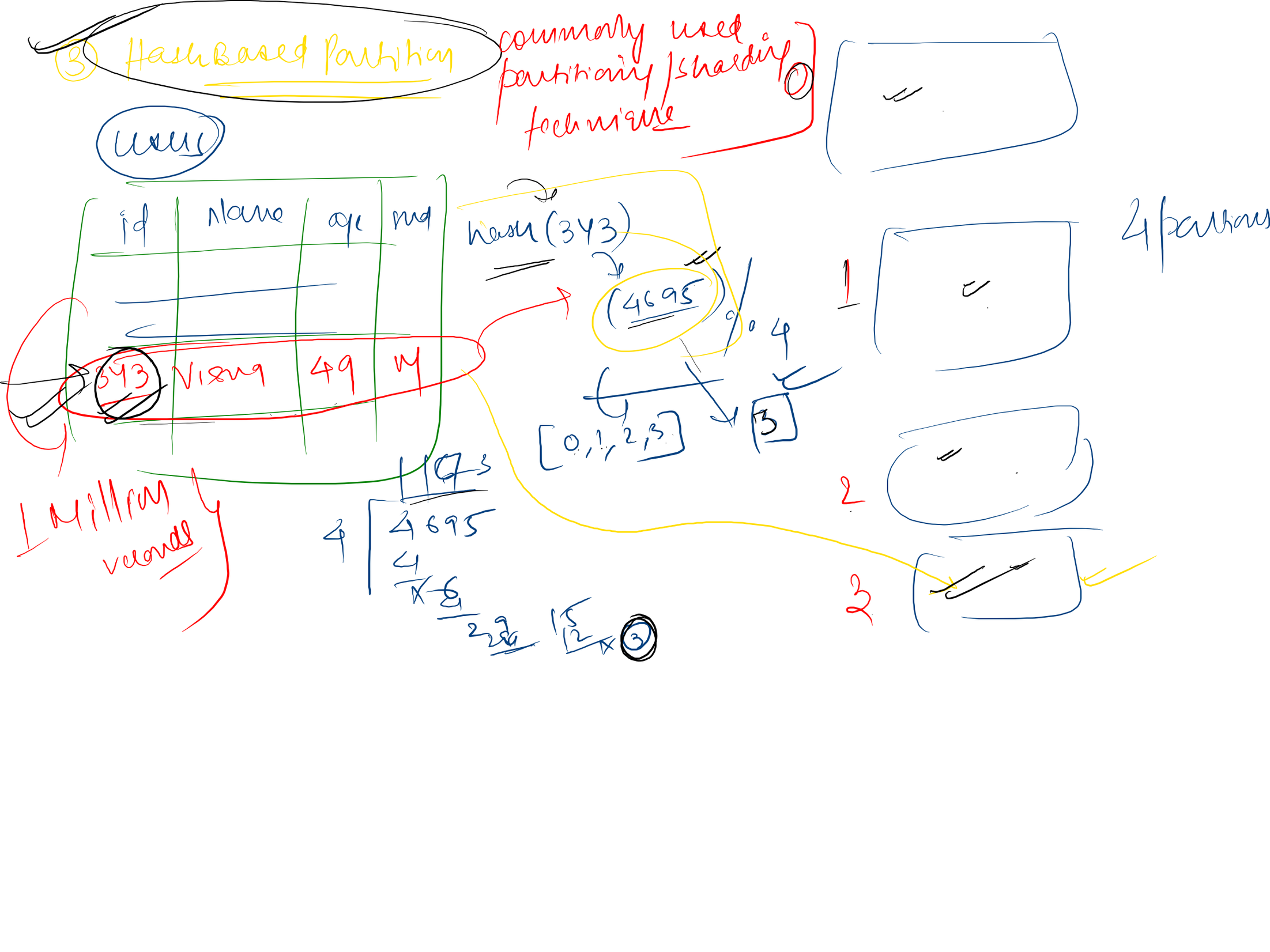
Sales — 24 hours

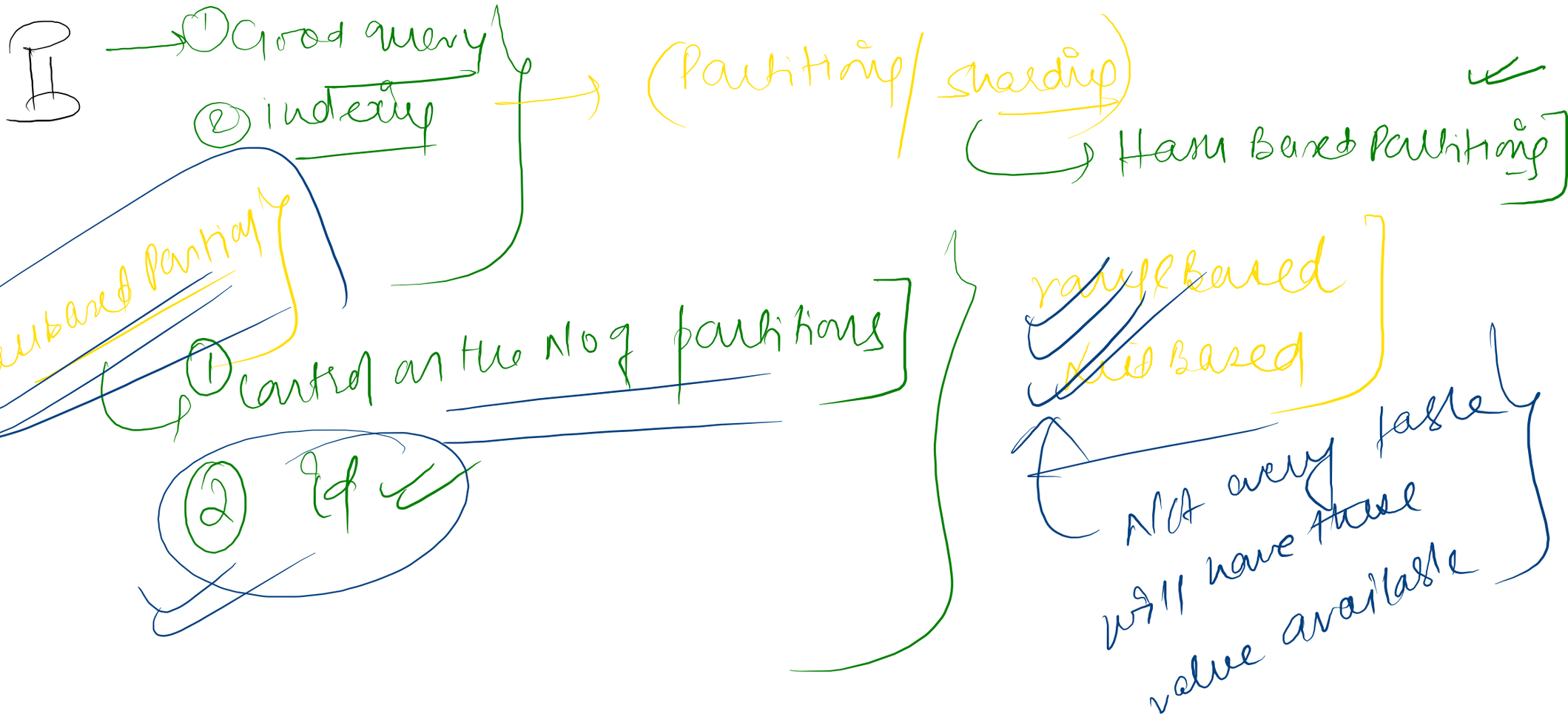
id	hour	count



② List Based Partitions




3 Hash Based Partitioning
commonly used partitioning/sharding technique
users
id
Name
age
md
343
Vinay
49
M
1 Million records
hash(343)
4695
[0,1,2,3]
3
4 partitions
1
2
3
4695
4
4
6
23
15
12
3


① Good query
② indexing
(Partitioning/ sharding)
Hash Based Partitioning
Hash based Partitioning
① control on the No of partitions
② Eq
range based
list based
N/A every table will have these value available

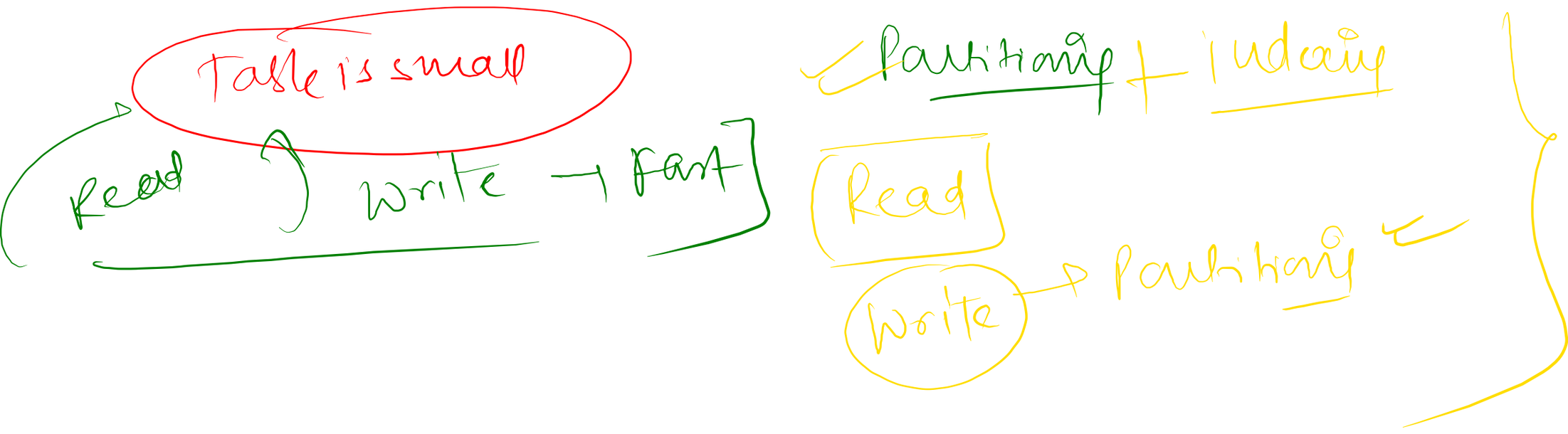


Table is small
Read
Write → fast
Partitioning + Indexing
Read
Write → Partitioning

Best way to optimise DB queries is to not use DB queries at all

— Viswa Ram

↓

Caching

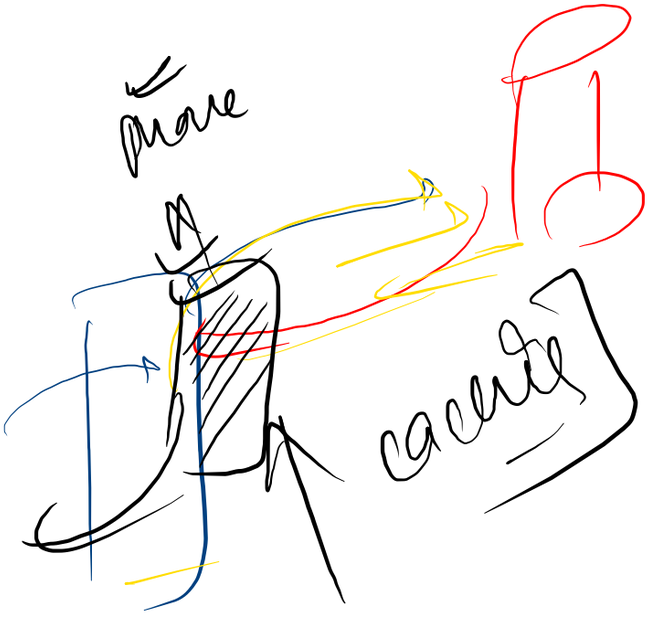
Cache

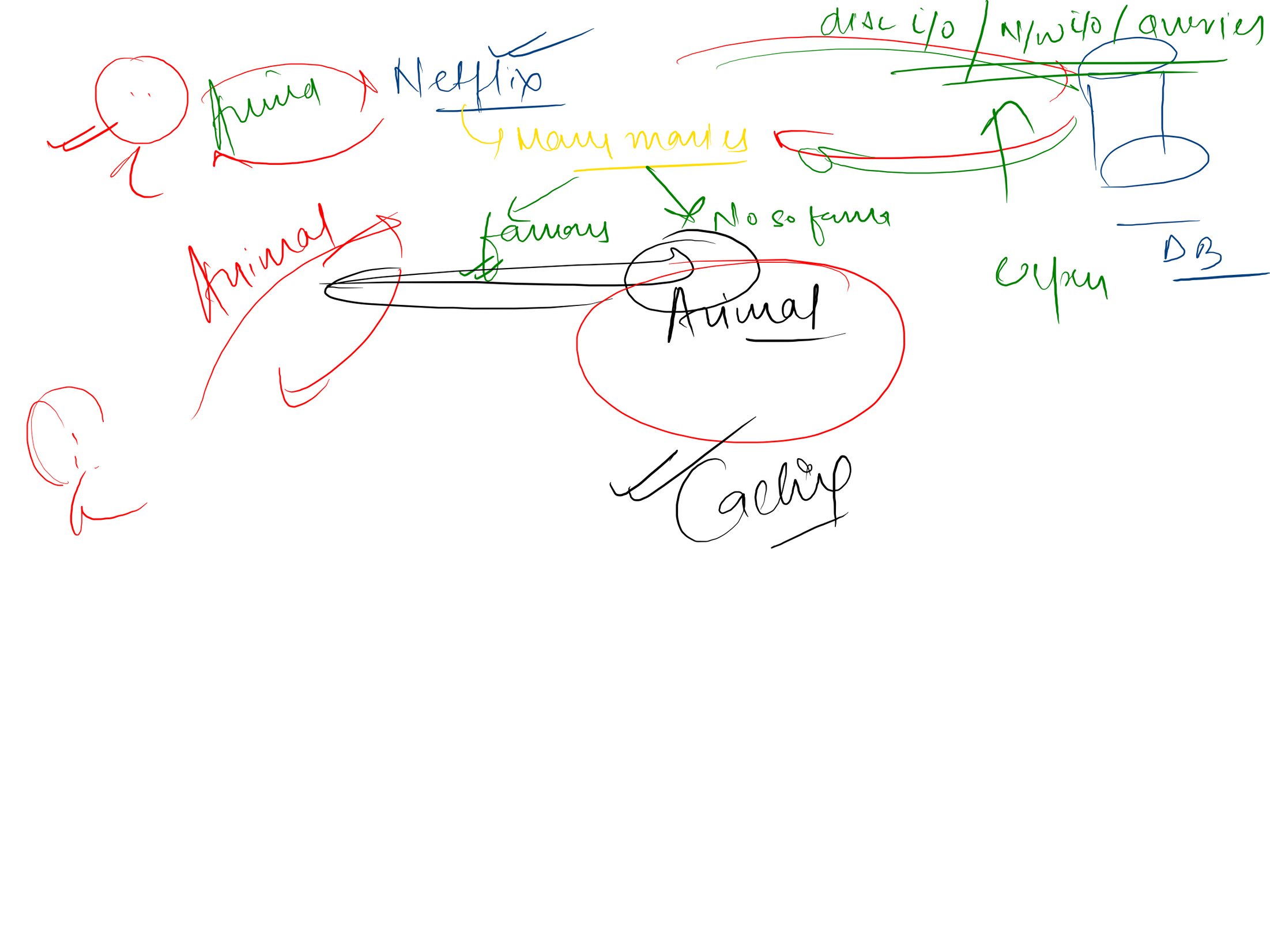
Avoid redundant reads

Diagram

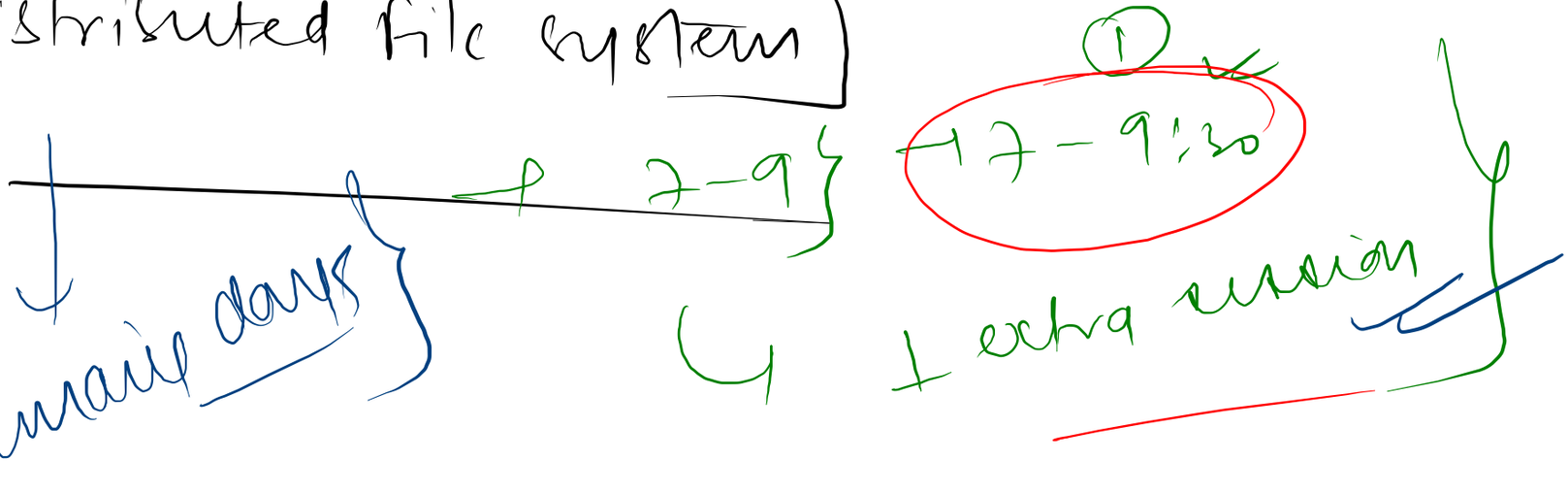
reels

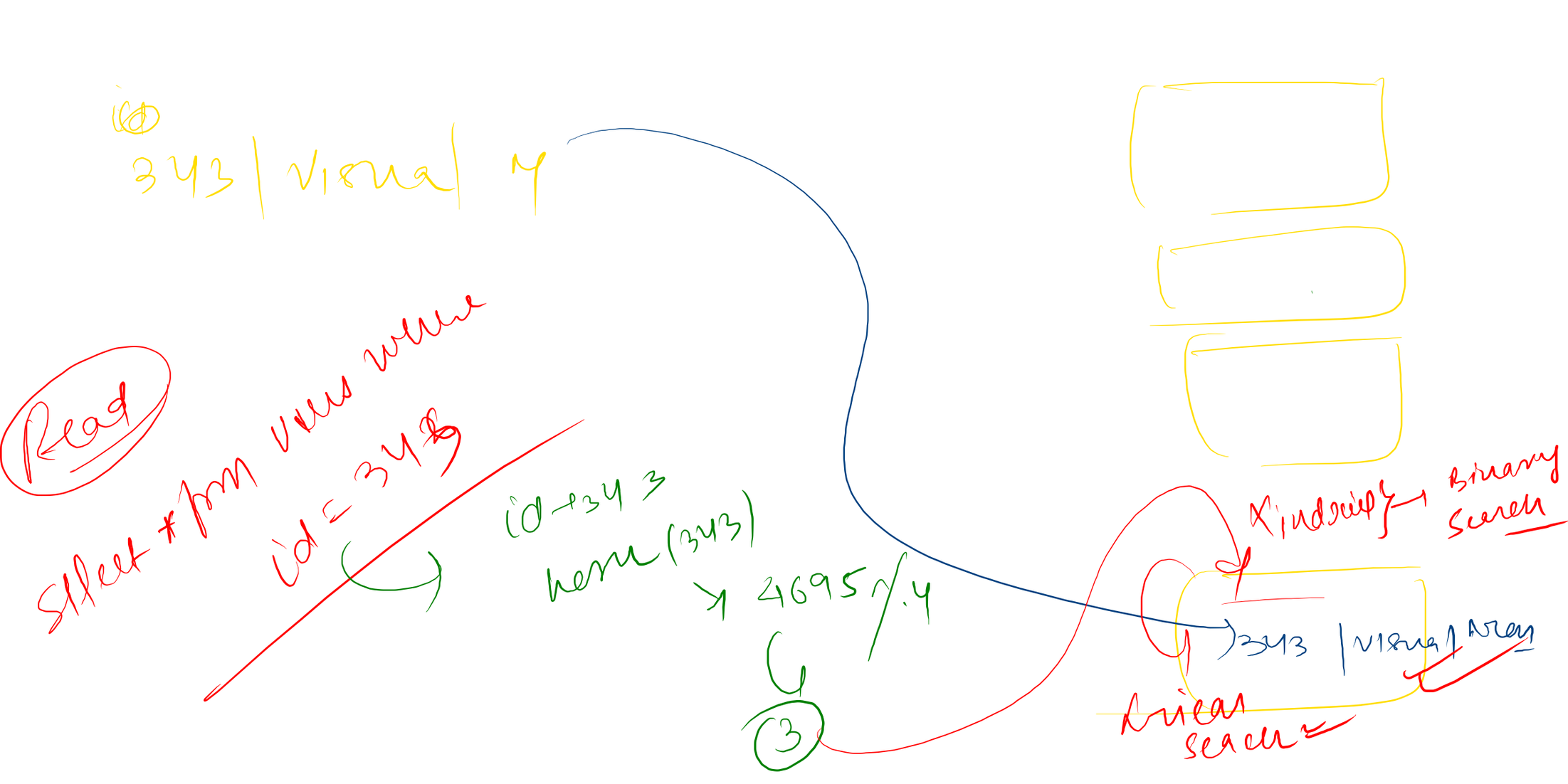
Animals

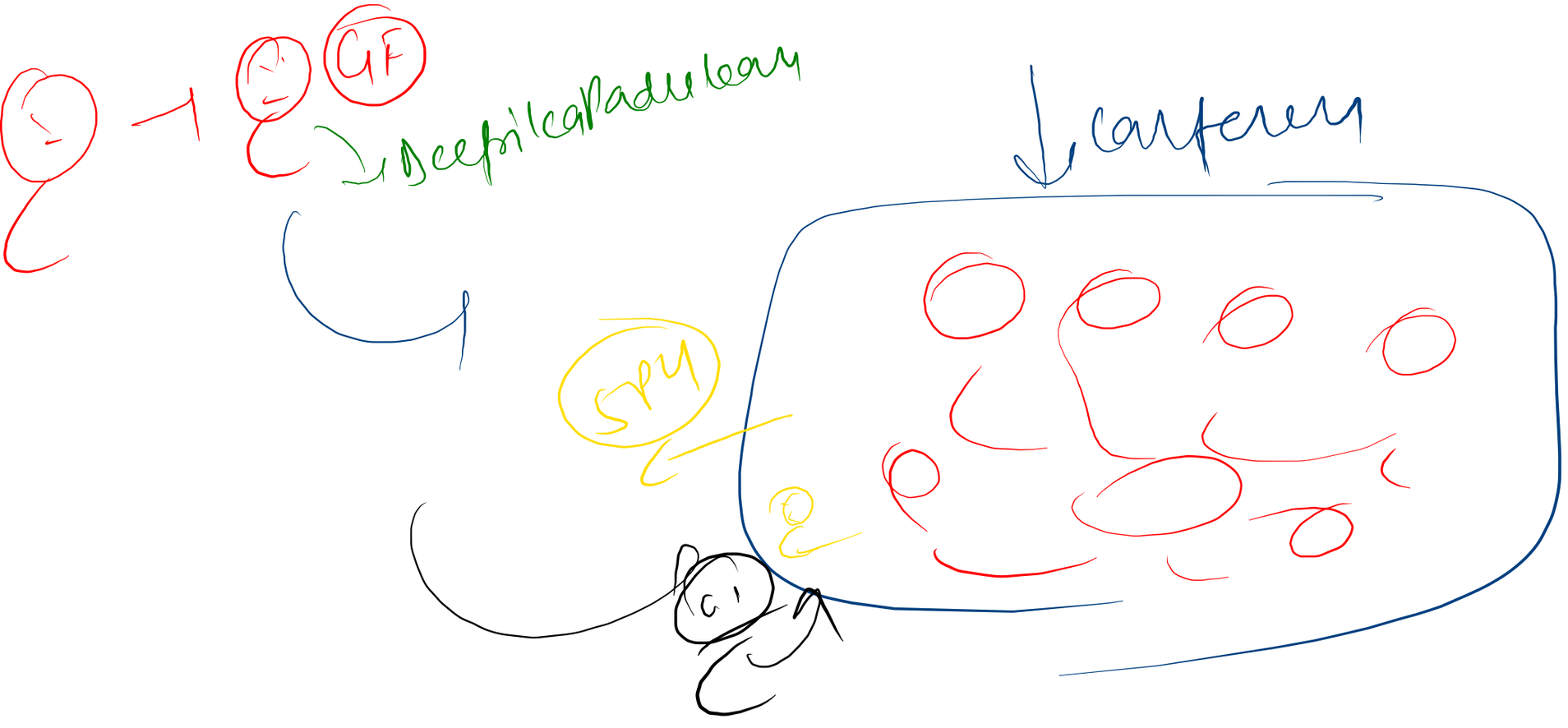



disc i/o / N/w i/o / queries
Netflix
Many many
famous
No so famous
Animal
Animal
Cache
DB

Distributed file system




343 | Virat | 4
Read
Select * from virat where id = 343
id = 343
here (343)
4695 % 4
3
Findmid
Binary search
343 | Virat | Array
Linear search


GF
Deepikapadukone
conference
spy