SQL vs NOSQL

SQL
↳ Relational D

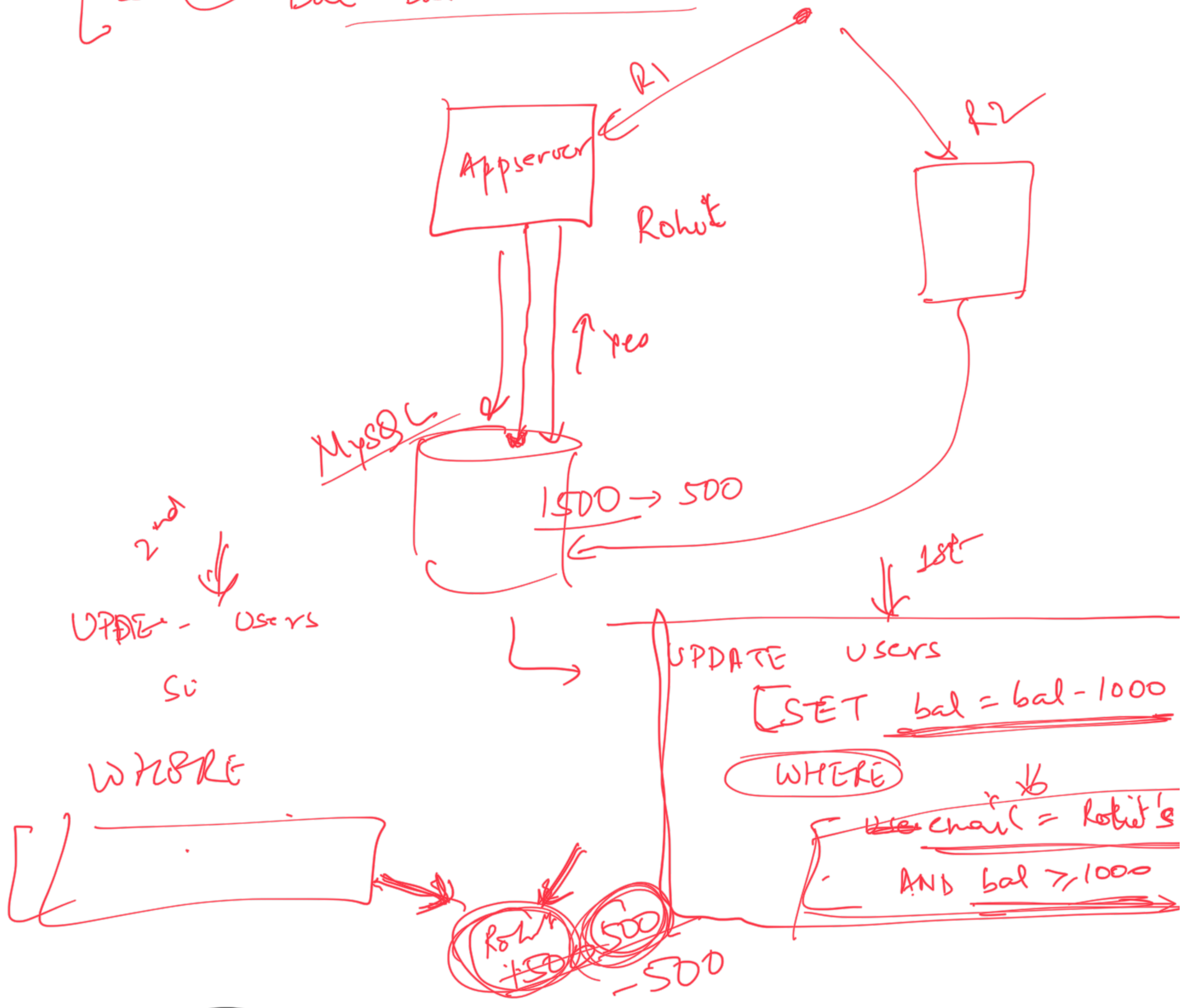
Normalised ←

ACID
- A → Atomicity
- C → Consistency
- I → Isolation
- D → Durabili

1000 INR

⇓

1. ✓ Check if bank bal ≥ 1000
   ↳ No, return . . .
2. ✓ Bal = bal − 1000

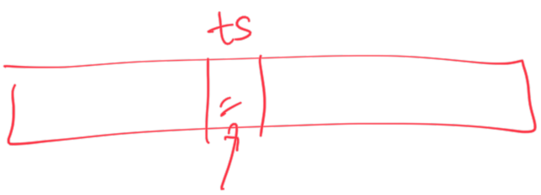


```
UPDATE - Users
SE
WHERE
```

```
UPDATE users
SET bal = bal - 1000
WHERE
~~user~~ email = Rohit's
AND bal >= 1000
```

ACID	Normalised	Defined schema

users



userid	username	timesta
int	varchar(20)	DATETIME
4 bytes	20 bytes	8 bytes

$\boxed{32 \text{ bytes}}$

Schema for e-commerce website

t-shirt
- color
- size
- collar type
- ⋮

Macbook-air
- RAM size
- HDD size
- HDD type
- Processor typ
- Freq /
- Screen siz
- Color

$\boxed{100{,}000 \text{ tables}}$

costly query.

```
SELECT
     *
   FROM table
        WHERE attribute LIKE "% size:
```

1. Fixed schema/columns might not fit every usecase
2. SQL has almost zero power post sharding.

$\boxed{\text{NoSQL}}$

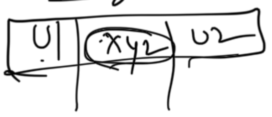
0. Sharding-key
1. De-normalisation

Users

Conversations

U1 | U2

messages

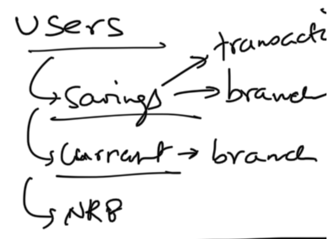


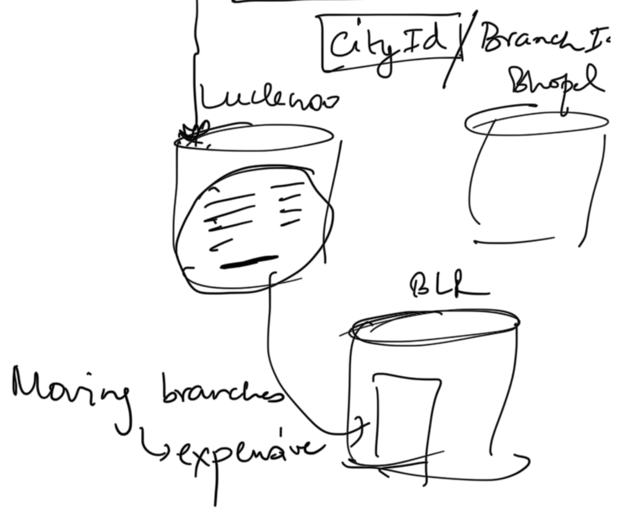
U1

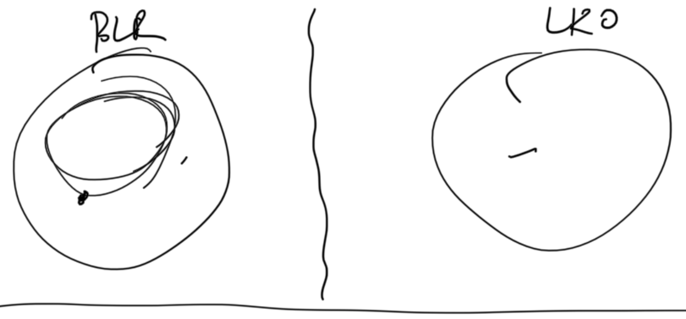
U2

Banking

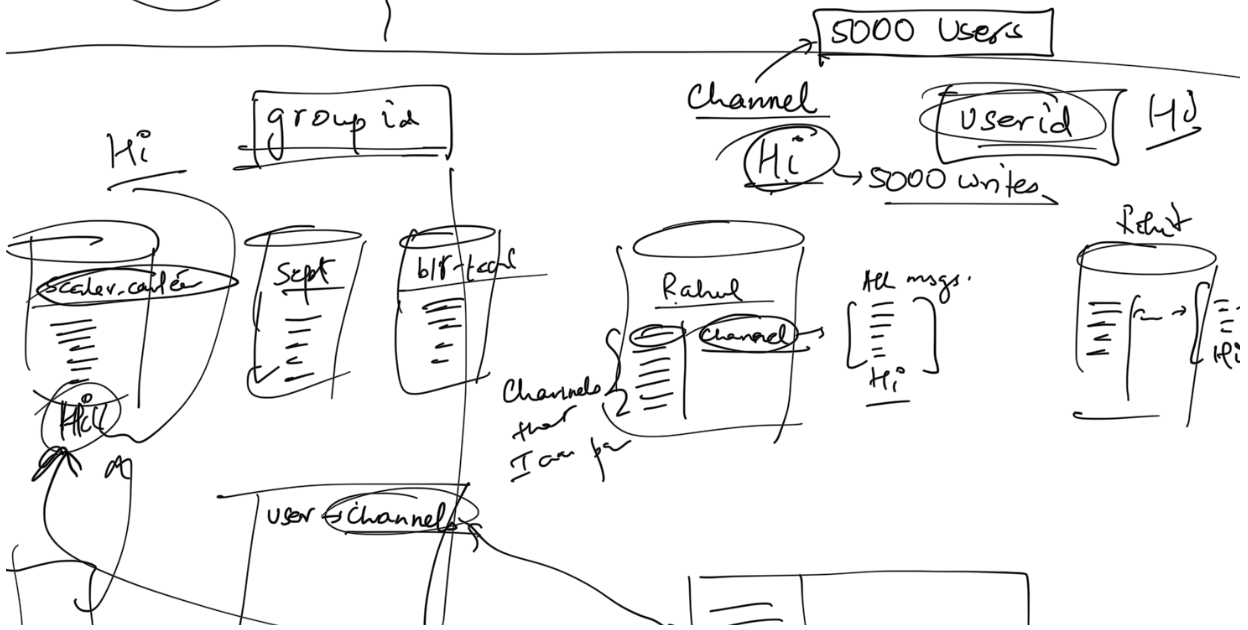
1. Account → Tell me balance
2. Account → Transaction History
3. User → List of Accounts
4. Create new transaction

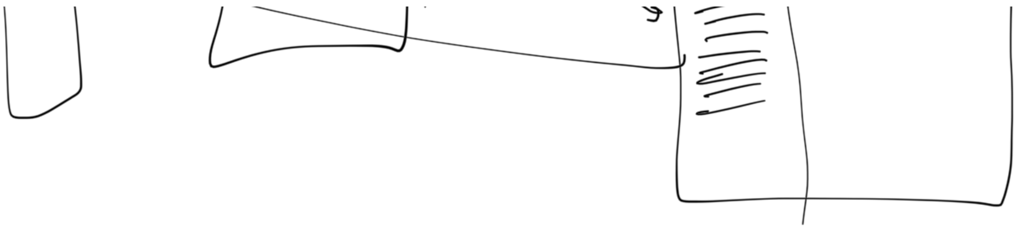


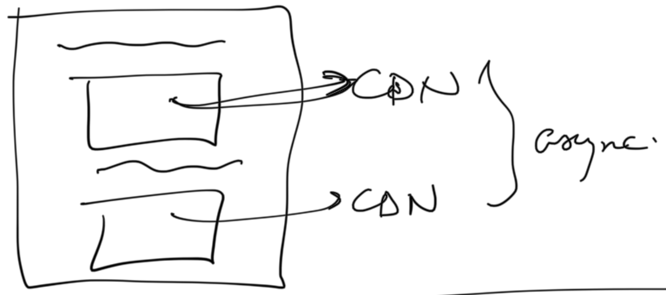




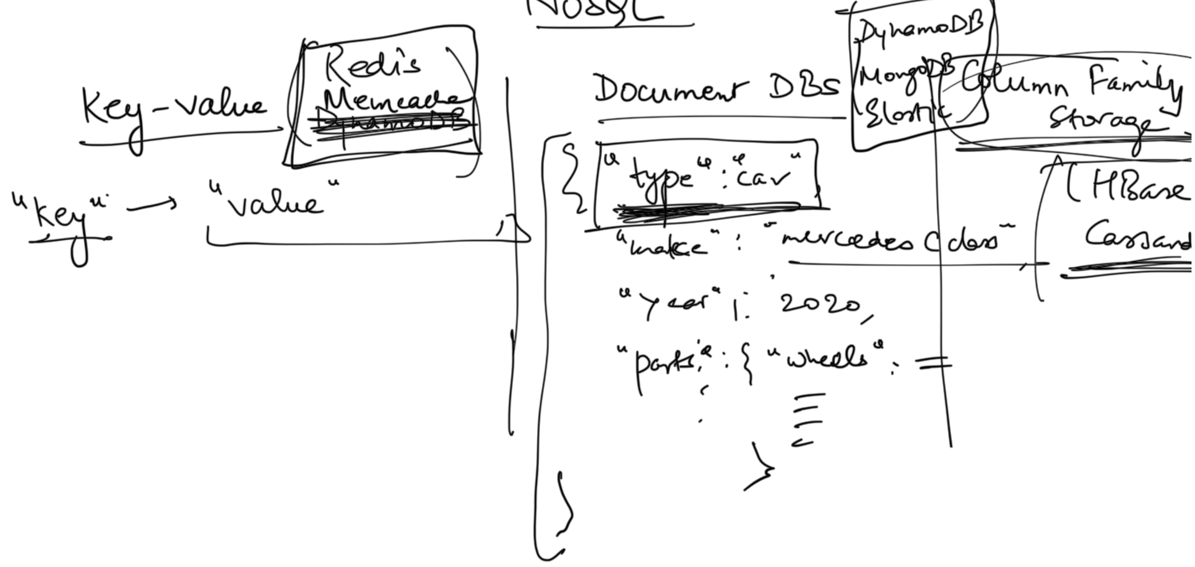
UBER



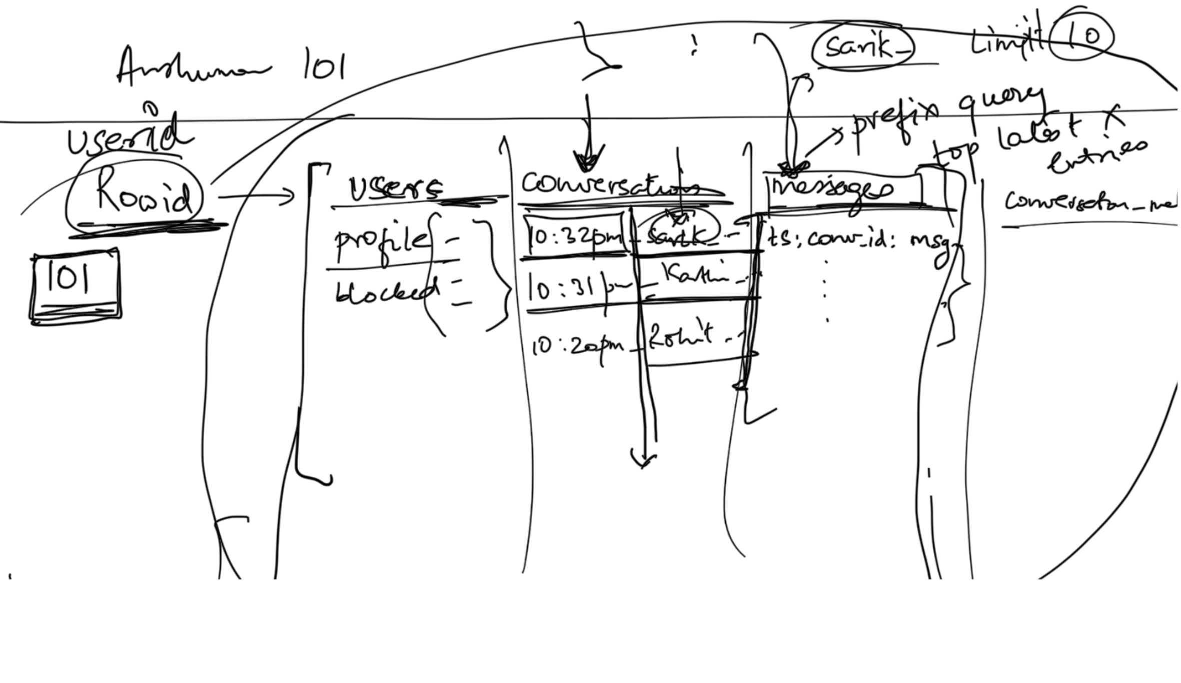




NoSQL



```
{ "item": 'laptop
  "cpu" : "2GHz"
  "cpu-type" : "M3"
  :
}
```



101 →

Sharding

document.find("utype" ~~[illegible]~~)

10

10,000

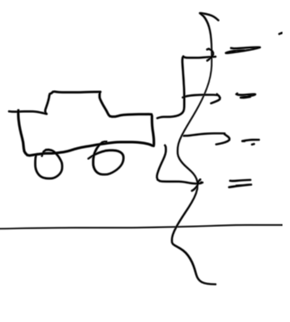
"#sachin"

① Data does not fit on single machine

2

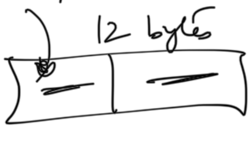
Key-value store	CF Store	Document DB
✓ Key → String Value → String	✓ #sachin → Tweets ts DESC ~~Popular Tweet~~ Top.10, Offset.10	

KV	CF	Document DB
✓ No History needed ↳ Key value	History needed ↳ CF	



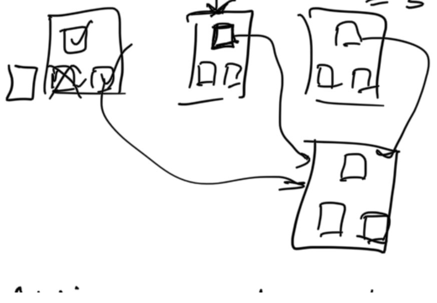
e-commerce application
↳ Document DB

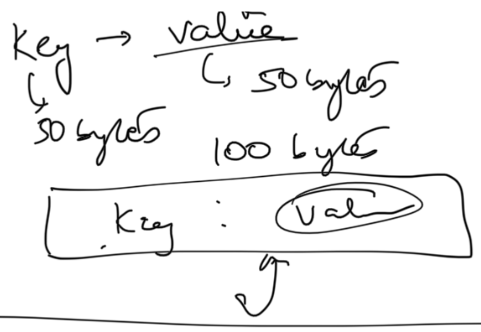
① ts userid
8 by 4 bytes
12 bytes

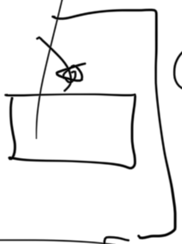


100 byte

② replication_f = 3



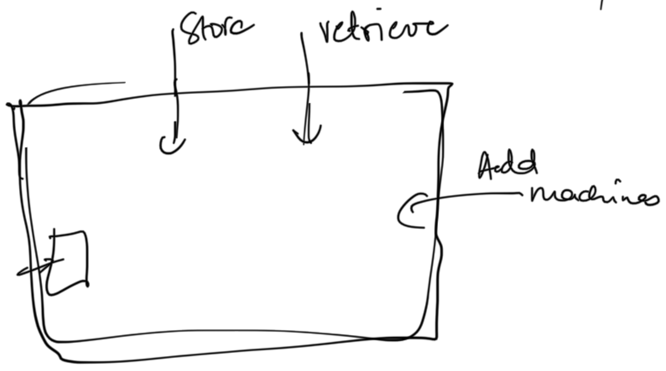


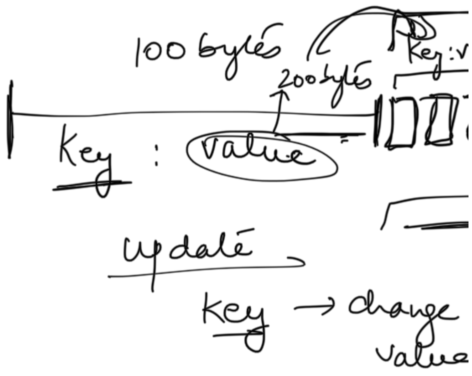


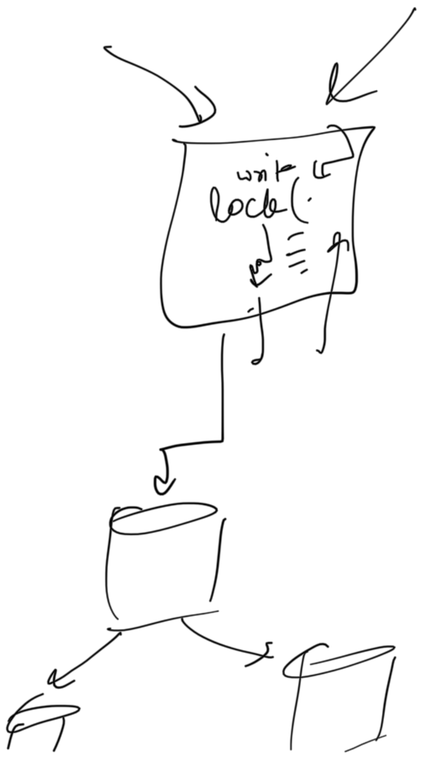
(1) Adding new shard + data migratio

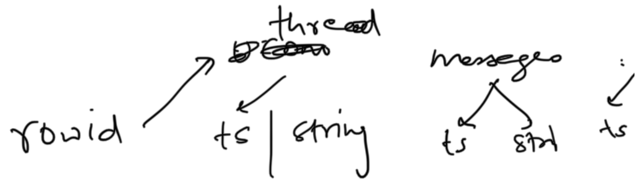
(2) When machine dies, actions to get repli level back to 3.

- Find entries quickly
- Support updates



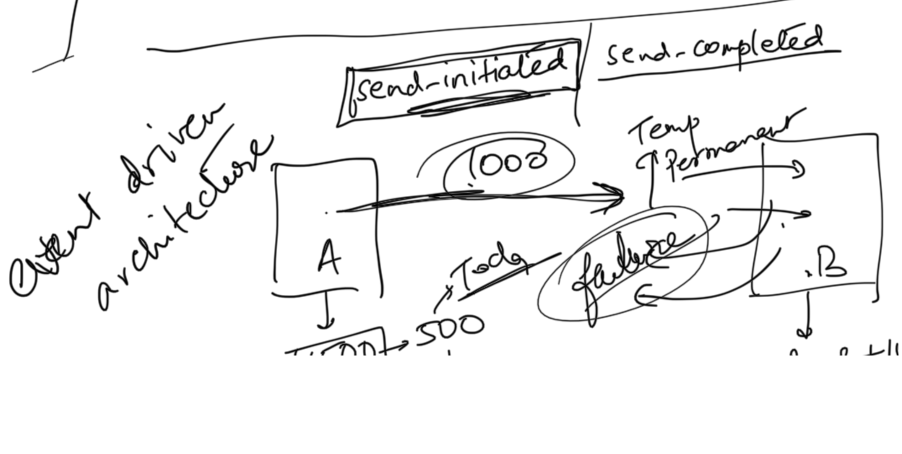






Appservers

send-initiated | send-completed



1500

B = B + 1

1500 Rollback

Tomorrow

lacoste t-shirt

sharding

man

"type: "t-shirt"

~~item~~ item

Male
size: L
Color: Bl

type: "t-shi

brand: "laco

Key → value

Master

Key | value
UNIQUE
Varchar(256) | varchar(256)

user

user_friends
~~userid~~
u1 = userid

posts
creator_id = userid

SELECT * FROM Master
WHERE key = <key

Friends

Posts

Profile

friends: [
posts: {

Userid

or