Agenda.

→ Types of NoSQL DBs.
- ↳ K:V DB
- ↳ Document DB
- ↳ Column family DB
- ↳ File Storages.

Key Value DB.

↳ Giant Hashmap distributed across servers.

Ex : REDIS. (Inmemory KV DB, optional Disk persistence)

Global cache. RAM

Memcached (Inmemory)

DynamoDB (AWS)

↳ Disk persistence

Key	Value
"abc"	1234
	{ }

Key : String

Value : String, Array, JSON, Set, Sorted Sets, Bloomfilter, Custom Data Structures.

⇒ Very High R|W throughput.

$\downarrow$

No. of operations|sec.

⇒ Disadvantages.

- No complex queries.
- No aggregates.
- No searches.

⇒ Sharding Key

$\downarrow$

$\bigcirc$ Key

Soft limit
$\begin{cases} \text{Key} \sim 100B. \\ \text{Value} \sim 10KB. \end{cases}$

⇒ Rate Limiting Counter : REDIS.

Document DBs.

↳ Storing unstructured | semi-structured data.

⇒ JSON | JSONB.

⇒ Ex: MongoDB | CouchBase | ElasticSearch | ......

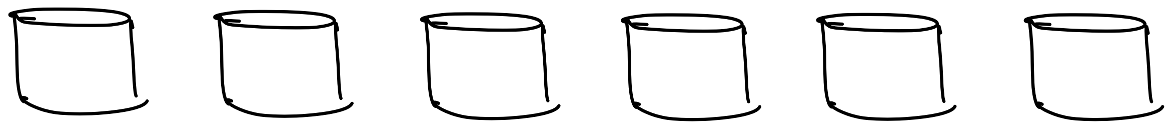
```
{
    __doc_id : ____          → Unique.
    product_id : ____
    title : ____
    brand : ____
    type : ____
    Ram : {
            Capacity : 8GB.
            type : - - - -.

          }

    ====
}
```

Sharding Key

- ↳ Doc_id : default Sharding key.
- ↳ We can choose our custom Sharding key as well.



MongoDB.

↳ Indexing → Only on top level attrs.

↳ Local Indexing.

Index : type | Sharding key : Doc-id.

↓ Laptops.

⇒ Query : fetch all the laptops.

↳ Fan Out.

→ ACID

↳ MongoDB provides ACID properties but they become extremely slow across SHARDS.

MongoDB : limit of 16MB on document size.

Strict.

Why such limit ?

⇒ Because R/W operations happens at Document level.

⇒ Even if we modify 1 char, Complete doc will be rewritten.

Column family DBs.

$\downarrow$

Columnar | Wide family.

$\Rightarrow$ Data is stored in wide-column format.

$\swarrow$

Provides very fast aggregate queries.

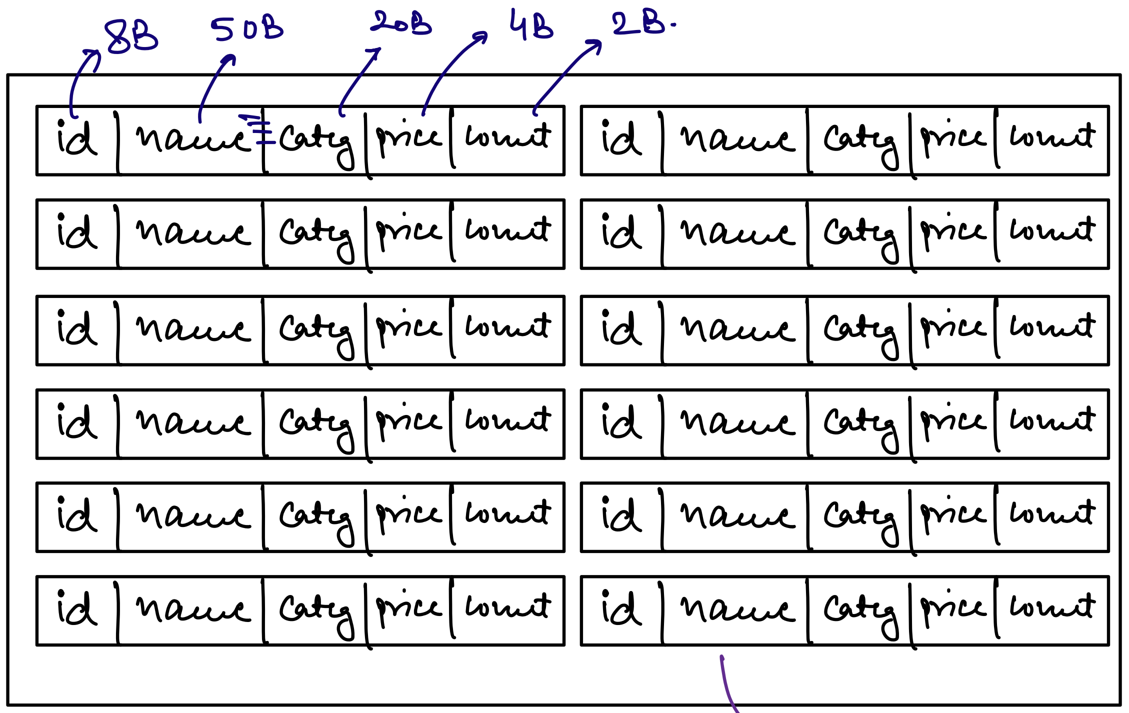
$\Rightarrow$ Data is tabular, but No relation b/w tables.

Ex: Cassandra, BigTable, - - -

$\hookrightarrow$ TimeSeries DB.

- $\rightarrow$ Aggregate Queries.
- $\rightarrow$ Analytics
- $\rightarrow$ Extremely fast writes.
- $\rightarrow$ Paginated Queries.

SQL



Row wise.

Q. Product with id = 100.

→ fast in SQL DB.

→ Utilization = 100%.

Q. Get count of all the products.

→ Read the entire table

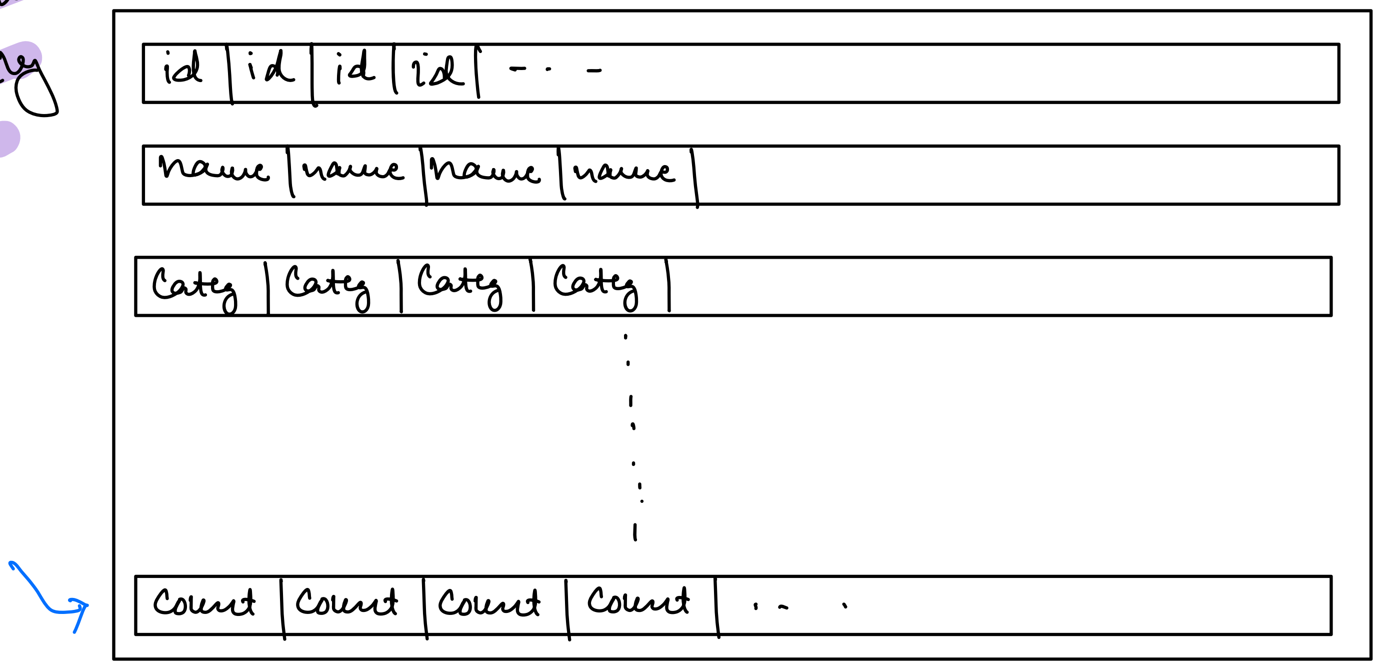
from every row, we are only interested in count.

Size of 1 row = 100B

Count → 2B.

$$\text{Utilization} = \frac{2B}{100B} = 2\%.$$

Wide Column family DB.


id | id | id | id | - - -
name | name | name | name
Categ | Categ | Categ | Categ
Count | Count | Count | Count | . . .

Q. Get count of all the products.

↳ Just get the count column & aggregate

Utilization = 100%

File Storge.

↳ files

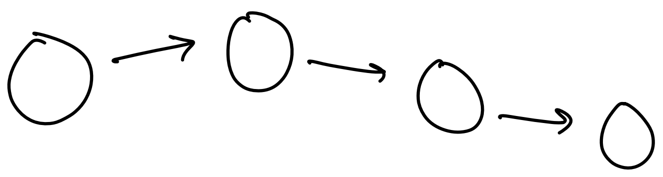
↳ Can be used for very big file.

↳ Distributed.

- → AWS S3
- → Azure Blob
- → Google Cloud Storage
- → HDFS.

Graph DB.

↳ FB friendships | LI followers.



Choosing the right DB.

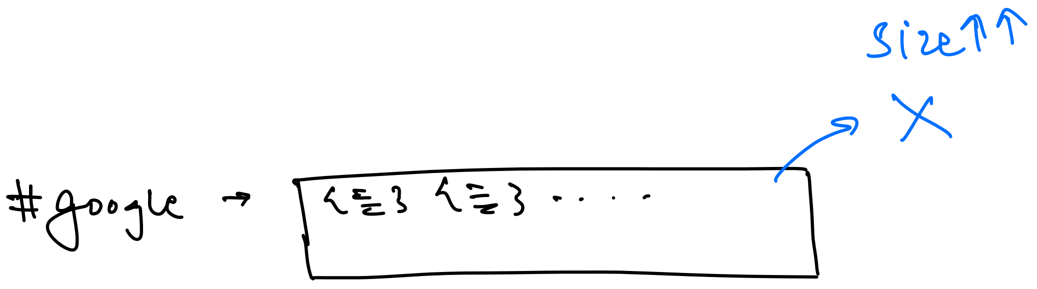
1. Twitter Hashtag

↳ Store most popular & recent tweets for each hashtag.

↳ Paginated response.

Graph DB. ✗

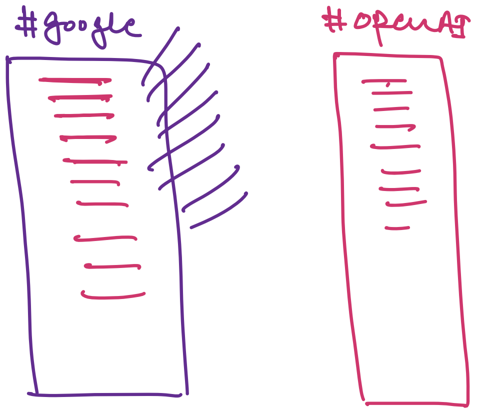
K:V ✗



SQLDB ✗

DocumentId ✗

Column DB ✓



2. Live score of a Cricket Match.

⇒ 6 hrs. : 240 balls.

↳ 0.01 writes | sec.

⇒ $\boxed{k:v}$

↳ matchId : {
—
—
—
—
—
—
—
—
—
—
}

3. Uber.

↳ I) Show the current location of Driver.

↳ k:v   driver-id : locⁿ.

II) Show the historical location of a Cab driver.

↳ Column family.

—— * ——