NoSQL

① NoSQL Databases

It stands for Not only SQL or Not SQL.
It is an non-relational data management system that does not require a fixed schema.
It avoids joins and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs.
It is used for Big data and real time web apps. System response time becomes slow when you use RDBMS for massive volumes of data.
To resolve this issue distribute database and load on multiple hosts whenever the load increases. This method is known as scaling out.

Features of NoSQL

- **Non relational**

It never follow the relational model.
Never provide tables with fixed column records. Work with self contained aggregates.
Doesn't require object relational mapping and data normalization.
No complex features like query language.

- **ACID**

schema free. NoSQL database are either schema free or have relaxed schemas.
Do not require any sort of definition of the schema of the data.
Offers heterogeneous structures of data in same domain.

- NoSQL databases are mainly categorized into four types
  - → key value pair based
  - → column oriented
  - → Graph based
  - → Document oriented

② Main characteristics of key - value DB (example from Redis).

A key value database used a simple key value method to store data. These database contain a simple string that is always unique and an arbitary large data field.
They are easy to design and implement.
This type of NoSQL database implements hash table to store unique key along with the pointers to the corresponding data values.
The values can be scalar data types as Integer or complex structures such as JSON, BLOB.
Key value stores have no query language but they do provide a way to add and remove key - value pairs

REDIS EXAMPLE

For vast majority of data storage with Redis, data will be stored in a simple key value pair. This is best shown through the redis - cli (command line interface) using GET and SET commands.

eg: we may want to store information about books. Such as title and author of few of our favourites.

> SET title " The Habbit "
OK
> SET author " JRR Tolkien "
OK

> GET title
" The Habbit
> GET author
" JRR Tolkien "

③ Document DB (Example from Mongo DB)

Built around JSON - like documents, document databases are both natural and flexible for developer to work with.

They promise higher developer productivity, faster evolution with application needs.

As a class of non relational sometimes called NoSQL database the document data model has become the most popular alternative to tabular relational databases

MONGO DB EXAMPLE

Relational

ID	First_name	last_name	Cell	City	Year-of_birth	location-x	location-y
1	'Mary'	'Jones'	516-555-2048	'Long Island'	1986'	'-73.9876'	'40.7574'

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'My App'	1.0.4
21	1	'DocFinder'	2.5.7

ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

MongoDB

first_name: = "Mary",
last_name: = "Jones",
cell: "516-555-2048",
city: "Long Island",

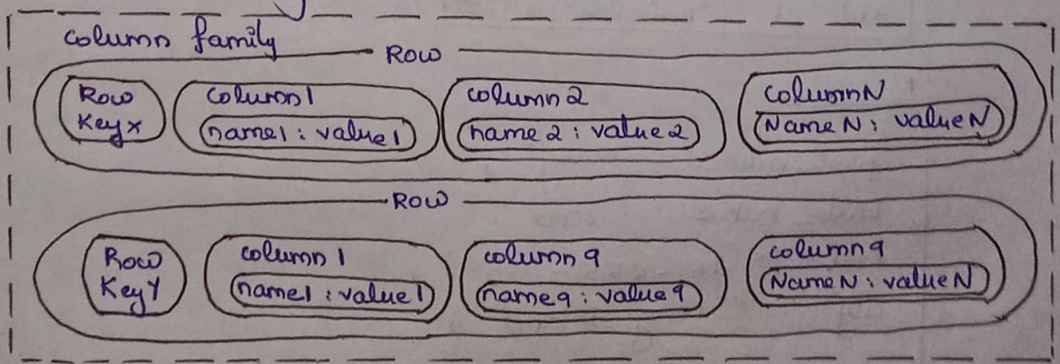
```
year-of-birth : 1986 ,
location : {
    type : "Point",
    coordinates : [-73.9876, 40.7574]
},
profession : ["Developer", "Engineer"],
apps : [
 { name : "My App",
   version : 1.0.4 },
 { name : "DocFinder",
   version : 2.5.7 }
],
cars : [
 { make : "Bentley",
   year : 1973 },
 { make : "Rolls Royce",
   year : 1965 }
]
```

④ Main characteristics from Column-Family db (Example from Cassandra).

Column Database use the concept of key space, which is sort of like schema in relational models.

This keyspace contains all the column families which then contain row, which then contain column

eg: RDBMS Table having the column ID, Name, Age, Gender, city.



2023

Cassandra Example

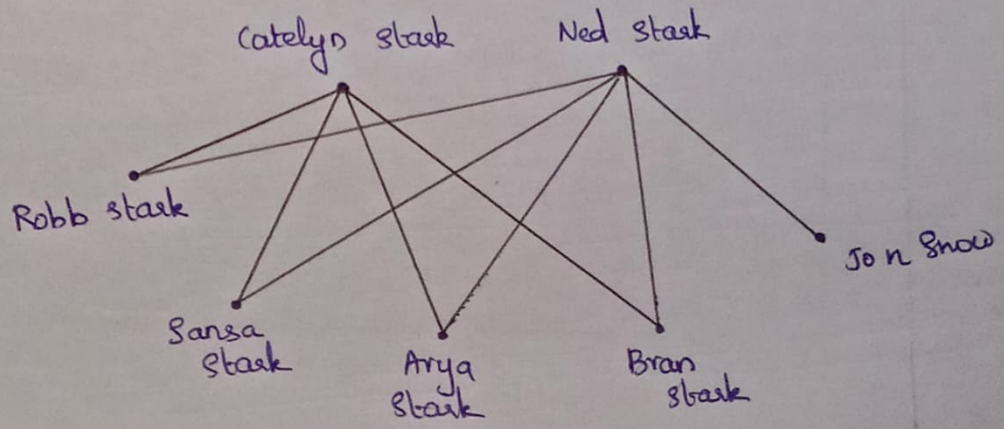
```
// column family
{
// row
  "pramod - sadalage" : {
        firstName : "Pramod",
        lastName : "Sadalage"
        lastVisit : "2012/12/12"
    }
// row
    "martin - fowler" : {
        firstName : "Martin",
        lastName : "Fowler",
        location : "Boston"
    }
}
```

⑤ Graph db (Example from ArangoDB)

It is nodes free objects where arbitary data can be stored and relations btw the objects (edges). Edges typically have a direction going from one object to another or multiple objects.

Vertices and edges from a network of data points which is called a "graph".

Graph



ArangoDB Example

```
LET data = [
  {
  "parent": {"name": "Ned", "surname": "stark"},
  "child": {"name": "Robb", "surname": "stark"}
 }, {
  "parent": {"name": "Ned", "surname": "stark"},
  "child": {"name": "sansa", "surname": "stark"}
}, {
  "parent": {"name": "Ned", "surname": "stark"},
  "child": {"name": "Arya", "surname": "stark"}
}, {
 "parent": {"name": "Ned", "surname": "stark"},
 "child": {"name": "Bran", "surname": "stark"}
}.....
]
```