1. What is NoSQL data base?

Answer: NoSQL (Not only SQL) is a type of database management and persistence system (DBMS).

- Does not use a relational model.
- Schema less i.e. no fixed schema.
- Open source.
- Does not use SQL as a querying language.
- Distributed fault tolerance architecture – runs very well on clusters.
- No joins needed.
- It does not replace RDBMS but compliments it.

Below are some types of NoSQL solutions:

- Key value Databases (DynamoDB).
- Column Family Databases (Big Table, HBase, Cassandra).
- Document Databases (CouchDB, MongoDB).
- Graph Databases (Neo4J).

2. How does data get stored in NoSQL database?

Answer: There are various NoSQL Databases. Each one uses a different method to store data. Some might use column store, some document, some graph, etc. Each database has its own unique characteristics.

Below are various categories of NoSQL databases:

A. **Key value storage databases**: Data is stored in the form of key value pair. Key – Value is based on a hash table where there is a unique key and a pointer to a particular item of data (value).
Mappings are usually accompanied by cache mechanisms to maximize performance.
API is typically simple - implementation is often complex. The values are not queryable.
DynamoDB is an example of Key Value database.

Key	Value
name	Bharat
location	Hyderabad

B. **Column family storage databases**: The database key points to column families comprising of multiple columns.
Google's Big Table, HBase, Cassandra are some examples of Column family databases.

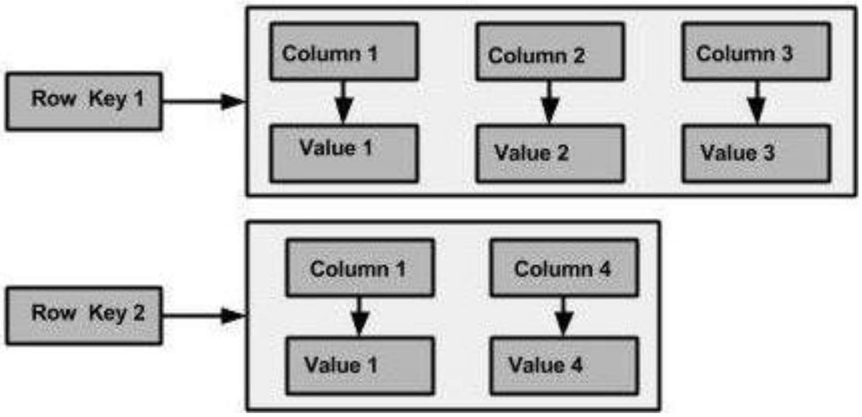


Fig 2.b.1: Column Family logical storage

C. **Document based storage databases**: Documents are addressed in the database via a unique key that represents that document. The structure of documents can be XML, JSON or BSON formatted, for instance. In addition to the key, documents can be retrieved with queries within the values. Example of Document based storage databases are CouchDB, MongoDB etc.

```
{
  name: "Ankit",
  phone: 1234567890,
  address:
          {
           street: "1234 Some_XYZ Pkwy" ,
           Apt: 1001,
           City: "Delhi",
           State: "Delhi"
          }
}
```

D. **Graph Storage Databases**: Graph Databases are built with nodes, relationships between nodes (edges) and the properties of nodes. Nodes represent entities (e.g. "Bob" or "Alice").
Similar in nature to the objects as in object-oriented programming.
Properties are pertinent information related to nodes (e.g. age: 18).
Edges connect nodes to nodes or nodes to properties.
Edges represent the relationship between the two nodes.
Example of Graph storage databases are Neo4J, FlockDB etc.

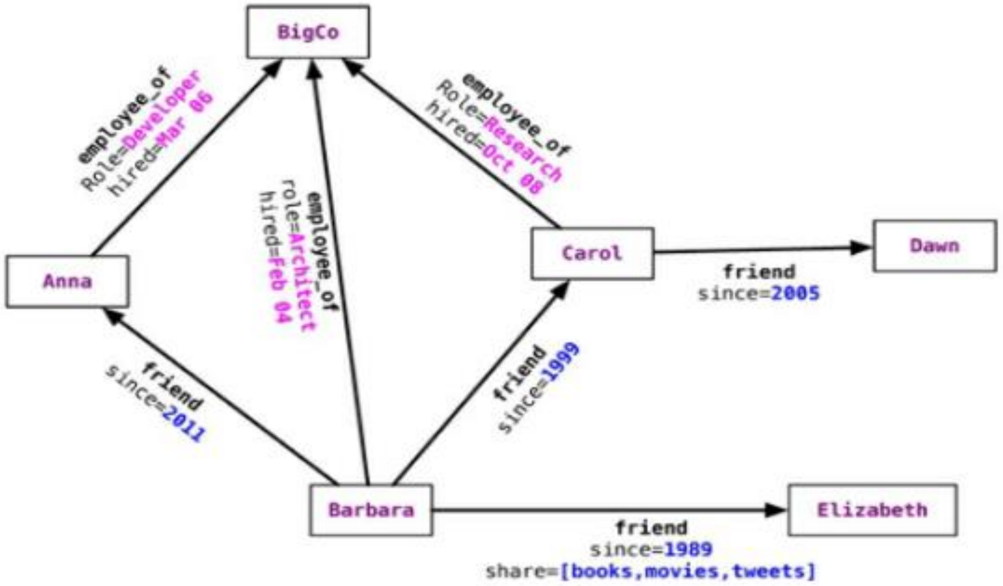


Fig 2.d.1: Graph based storage

3. What is a column family in HBase?

Answer: Columns in Apache HBase are grouped into *column families*. All column members of a column family have the same prefix. For example, the columns *courses: history* and *courses: math* are both members of the *courses* column family.

put 'student','r1','**courses**:math','v1'

Above command will insert data in student table having the value 'v1' in the column math of column family **courses.**

4. How many maximum numbers of columns can be added to HBase table?

Answer: There is no hard limit to number of columns in HBase table.

5. Why columns are not defined at the time of table creation in HBase?

Answer: As HBase is a NoSQL database and it is schema less so it is not mandatory to define columns at the time of table creation. Initially you can create table witch column family and later on you can define the columns on the fly, put attribute names in column qualifiers, and group data by column families.

From below example it would be clear how we can add columns on the fly:

1. **Create table :** creating employee table with column family personal

create 'employee','personal'

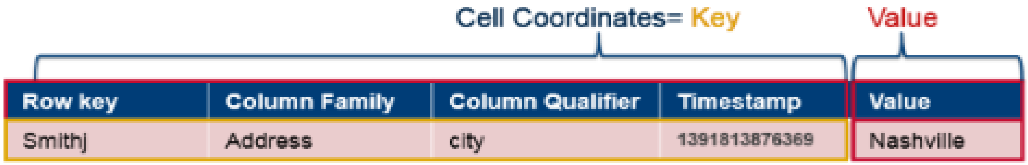
2. **Ingesting data into table:**
   put 'employee', '101', 'personal:empname', 'madhav'

In the above command we have inserted data into column **empname** of column family **personal.**

So here we have defined the column on the fly.

6. How does data get managed in HBase?

Answer: Data in HBase is organized into tables. Any characters that are legal in file paths are used to name tables. Tables are further organized into rows that store data. Each row is identified by a unique row key which does not belong to any data type but is stored as a bytearray. Column families are further used to group data in rows. Column families define the physical structure of data so they are defined upfront and their modification is difficult. Each row in a table has same column families. Data in a column family is addressed using a column qualifier. It is not necessary to specify column qualifiers in advance and there is no consistency requirement between rows. No data types are specified for column qualifiers; as such they are just stored as bytearrays. A unique combination of row key, column family and column qualifier forms a cell. Data contained in a cell is referred to as cell value. There is no concept of data type when referring to cell values and they are stored as bytearrays. Versioning happens to cell values using a timestamp of when the cell was written.



Row key	Column Family	Column Qualifier	Timestamp	Value
Smithj	Address	city	1391813876369	Nashville

Fig. 6.1 HBase Data Model – Physical Representation

7. What happens internally when new data gets inserted into HBase table?

Answer: Below are the steps that are performed internally when new data gets inserted into HBase table:

a. Whenever **put** command is issued by client the data is written to WAL (Write Ahead Log) for durability, updates are appended sequentially.
b. Updates are available to queries after **put** returns i.e. an acknowledgement is given by WAL.
c. Next updates are written to the Memstore.
d. Memstore is basically a write cache, once the Memstore is full data is written in the form of Hfiles (Java Objects) which are basically sorted key values on the disk.

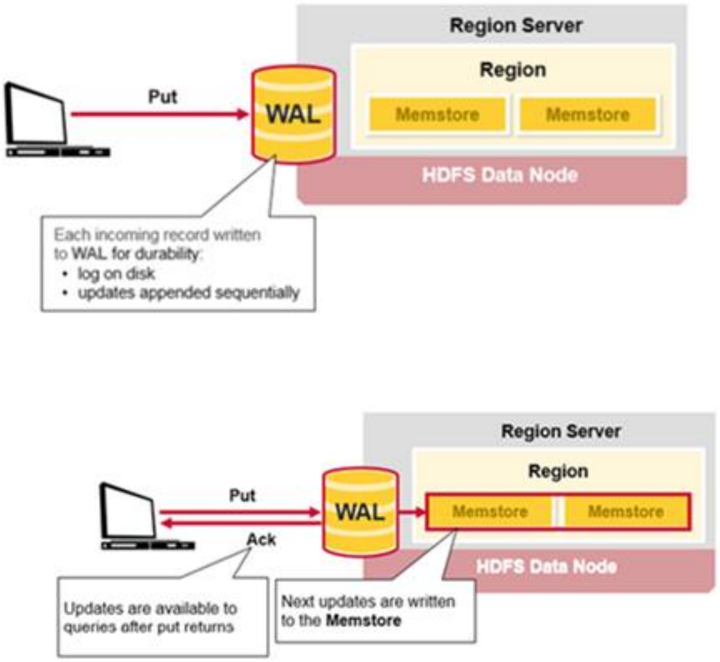
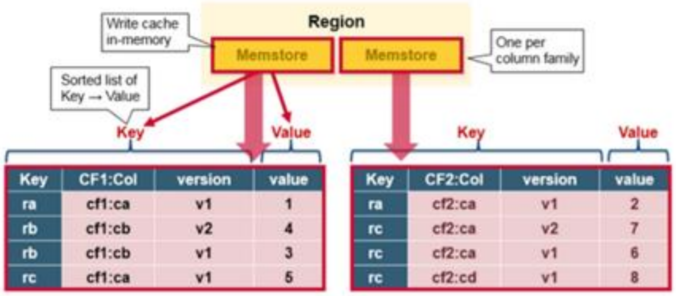

Region Server
Region
Memstore
Memstore
HDFS Data Node
Put
WAL
Each incoming record written to WAL for durability:
• log on disk
• updates appended sequentially
Region Server
Region
Put
Ack
WAL
Memstore
Memstore
HDFS Data Node
Updates are available to queries after put returns
Next updates are written to the **Memstore**

Fig. 7.1 HBase Writes



Key	CF1:Col	version	value
ra	cf1:ca	v1	1
rb	cf1:cb	v2	4
rb	cf1:cb	v1	3
rc	cf1:ca	v1	5

Key	CF2:Col	version	value
ra	cf2:ca	v1	2
rc	cf2:ca	v2	7
rc	cf2:ca	v1	6
rc	cf2:cd	v1	8

Fig. 7.2 Memstore