Big Data Hadoop and Spark Development

Session 10: HBASE BASICS Assignment 1

Big Data Hadoop and Spark Development

Assignment 1 – You must perform the given tasks.

Table of Contents

Big Data Hadoop and Spark Development

1. Introduction

In this assignment, you need to perform the given tasks.

2. Objective

This assignment will help you to consolidate the concepts learnt in the session 4.

3. Prerequisites: None

4. Associated Data Files None

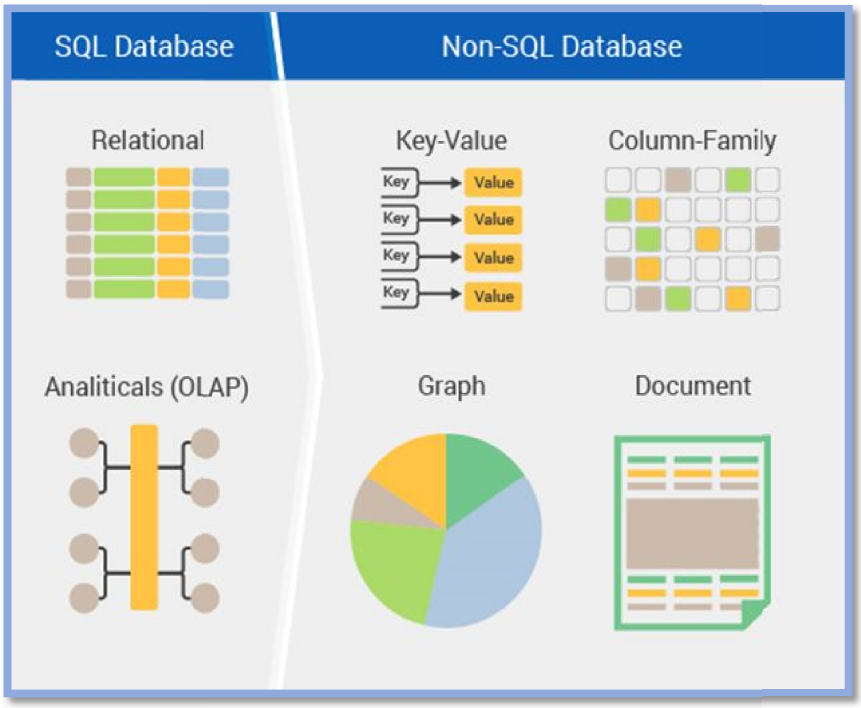
5. Problem Statement

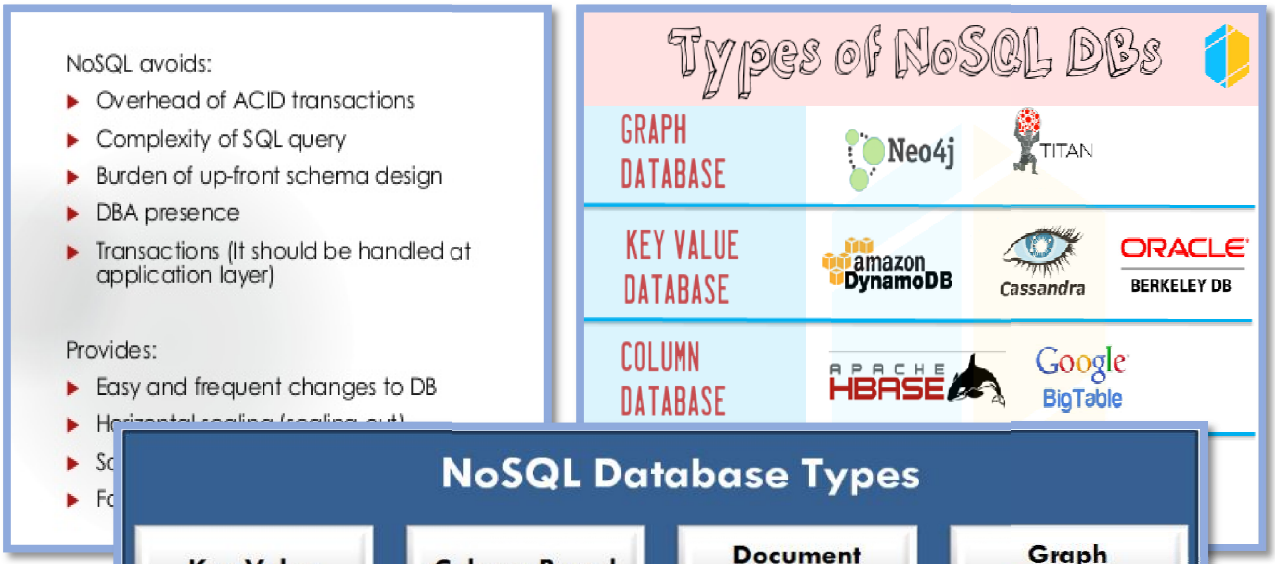
Task 1

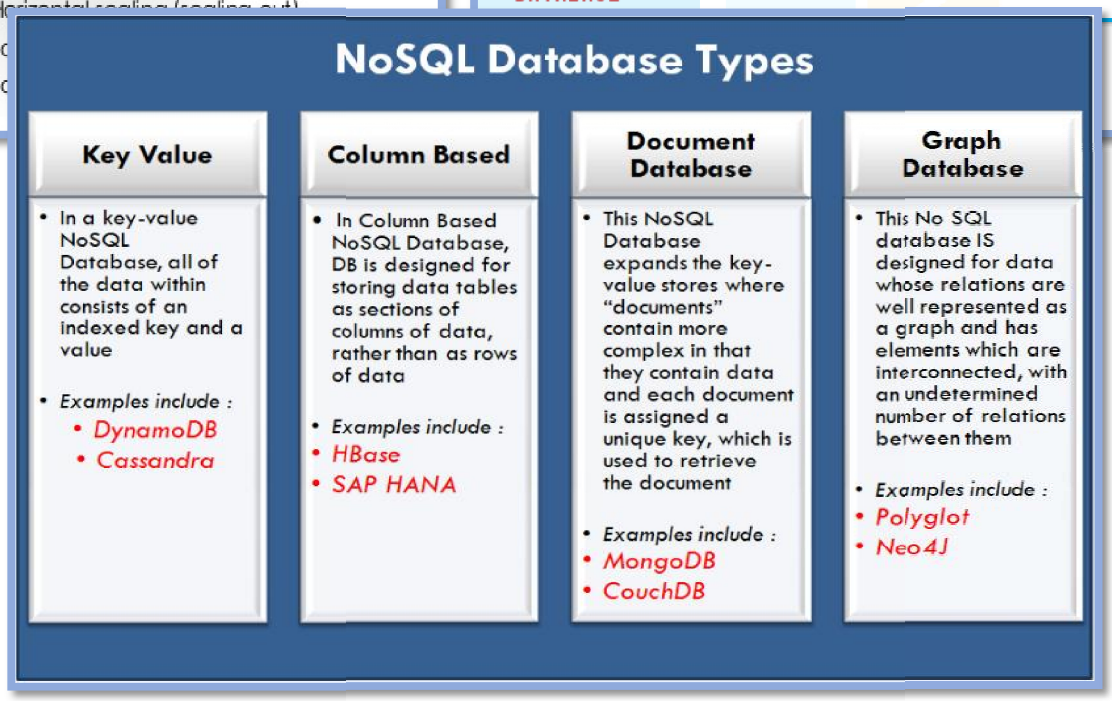
Answer in your own words with example.

1. What is NoSQL data base?

- NoSQL database, also called Not Only SQL, is an approach to data management and database design that's useful for very large set of distributed data. This database system is non-relational, distributed, open-source and horizontally scalable.
- NoSQL, which encompasses a wide range of technologies and architectures, seeks to solve the scalability and big data performance issues that relational databases weren't designed to address.
- NoSQL does not prohibit SQL. Some NoSQL systems are entirely non-relational, others simply avoid selected relational functionality such as fixed table schemas and join operations.


SQL Database
Non-SQL Database
Relational
Key-Value
Key
Value
Key
Value
Key
Value
Key
Value
Column-Family
Analiticals (OLAP)
Graph
Document


NoSQL avoids:
Overhead of ACID transactions
Complexity of SQL query
Burden of up-front schema design
DBA presence
Transactions (It should be handled at application layer)
Provides:
Easy and frequent changes to DB
Types of NoSQL DBs
GRAPH DATABASE
Neo4j
TITAN
KEY VALUE DATABASE
amazon DynamoDB
Cassandra
ORACLE
BERKELEY DB
COLUMN DATABASE
APACHE HBASE
Google BigTable


NoSQL Database Types
Key Value
In a key-value NoSQL Database, all of the data within consists of an indexed key and a value
Examples include :
DynamoDB
Cassandra
Column Based
In Column Based NoSQL Database, DB is designed for storing data tables as sections of columns of data, rather than as rows of data
Examples include :
HBase
SAP HANA
Document Database
This NoSQL Database expands the key-value stores where "documents" contain more complex in that they contain data and each document is assigned a unique key, which is used to retrieve the document
Examples include :
MongoDB
CouchDB
Graph Database
This No SQL database IS designed for data whose relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them
Examples include :
Polyglot
Neo4J

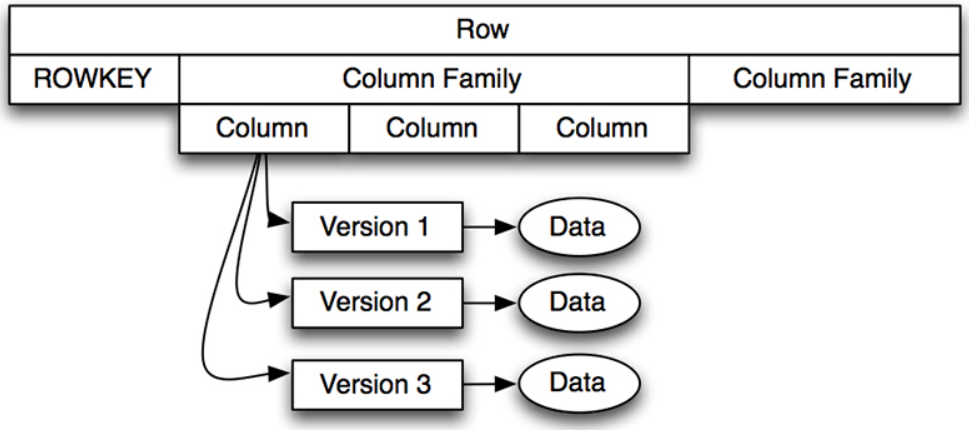
2. How does da

ta get stored in NoSQL database?

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.

How data is stored in Hbase?

HBase is not a relational database and requires a different approach to model the data. HBase actually defines a four-dimensional data model and the following four coordinates define each cell:



- **Row Key:** Each row has a unique row key; the row key does not have a data type and is treated internally as a byte array.
- **Column Family:** Data inside a row is organized into column families; each row has the same set of column families, but across rows, the same column families do not need the same column qualifiers. Under-the-hood, HBase stores column families in their own data files, so they need to be defined upfront, and changes to column families are difficult to make.
- **Column Qualifier:** Column families define actual columns, which are called column qualifiers. We can think of column qualifiers as the columns themselves.
- **Version:** Each column can have a configurable number of versions, and we can access the data for a specific version of a column qualifier.

Example

COLUMN FAMILIES

Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

How data is stored in MongoDB?

MongoDB is a document store, where data is stored as **Key: Value** pairs in **JSON** format.

Example

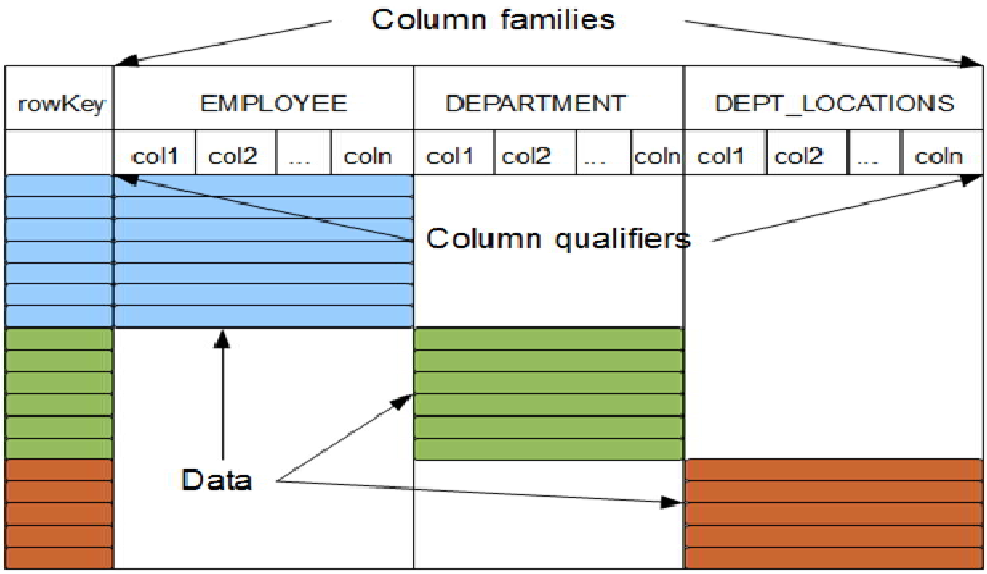
```
{
    name: "sid",
    phone: 1234567890,
    address:
                {
                  street: "1234 Some_XYZ Pkwy" ,
                  Apt: 1001,
                  City: "Richardson",
                  State: "Texas"
                   }
 }
```

Note: B-tree is used for indexing if we are using **MMAP storage Engine** in Mongodb and **B+ tree** if **WiredTiger storage Engine** is used.

3. What is a column family in HBase?

- 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. The colon character (:) delimits the column family from the member.
- The column family prefix must be composed of *printable* characters. The qualifying tail, the column family *qualifier*, can be made of any arbitrary bytes. Column families must be declared up front at schema definition time whereas columns do not need to be defined at schema time but can be conjured on the fly while the table is up on running.
- **Physically, all column family members are stored together on the filesystem. Because tunings and storage specifications are done at the column family level, it is advised that**

all column family members have the same general access pattern and size characteristics.



HadoopStore.com Product Table

RowID	ProductDetails Column Family			ProductAnalytics Column Family		
	#inStock	Price	Weight	Sales1Mon	Sales3Mo	Bundle
Toy Elephant	25	5.99	0.5	183	600	USB Key
USB Key	50	7.99	0.01	421	1491	YARN Book
YARN Book	30	30.78	2.4	301	999	USB Key

1. Data in HBase Tables identified by a unique key.
2. Related Columns grouped into Column Families which are saved into different files.
3. For performance reasons, you should usually not use more than 3 column families.

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

- There is **no hard limit to number of columns** in HBase, we can have more than 1 million columns but **usually three column families are recommended** (not more than three) as HBase currently does not do well with anything above two or three column families.
- Because column families are stored in separate HFiles, we should keep the number of column families as small as possible. So reducing the number of column families, reduces the frequency of MemStore flushes, and the frequency of compactions.
- And, by using the smallest number of column families possible, we can improve the LOAD time and reduce disk consumption.

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

The column qualifiers (columns) do not have to be defined at schema definition time because each row in HBase can have a different set of columns so they can be added on the fly while the database is up and running.

6. How does data get managed in HBase?

The main characteristics that make Hbase an excellent data management platform are fault tolerance, speed and usability.

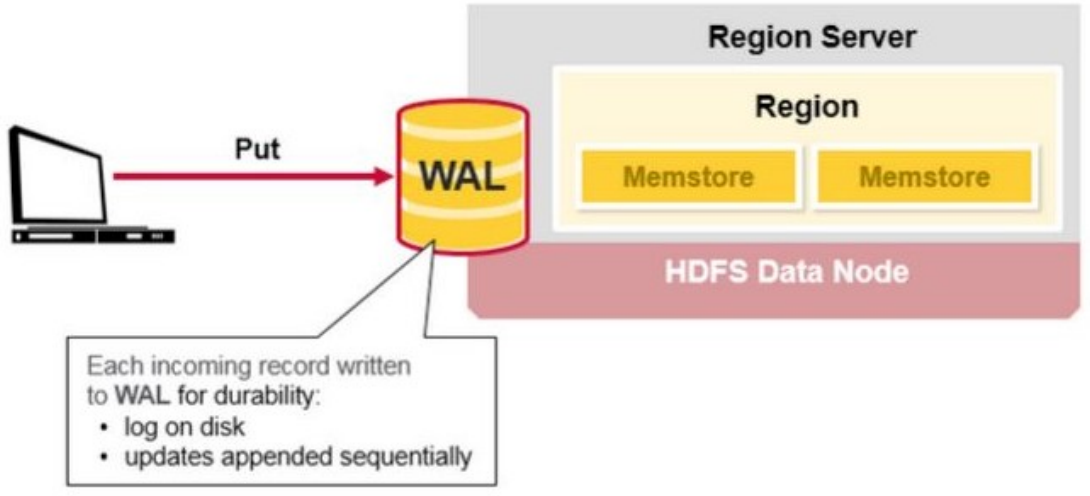
- **Fault tolerance** is provided by automatic fail-over, automatically sharded and load balanced tables, strong consistency in row level operations and replication.
- **Speed** is provided by almost real time lookups, in memory caching and server side processing.
- **Usability** is provided by a flexible data model that allows many uses, a simple Java API and ability to export metrics.

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.

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

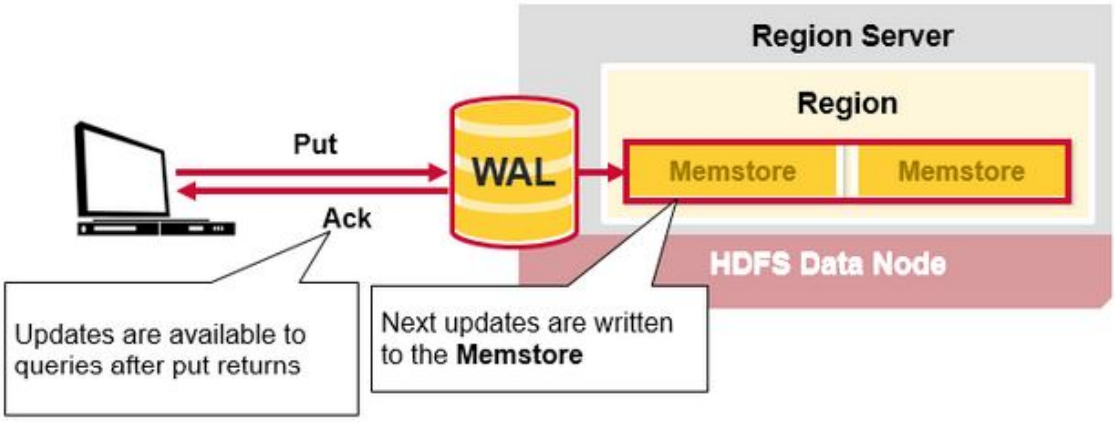
When the client issues a Put request, the first step is to write the data to the write-ahead log, the WAL:

- Edits are appended to the end of the WAL file that is stored on disk.
- The WAL is used to recover not-yet-persisted data in case a server crashes.



Once the data is written to the

WAL, it is placed in the MemStore. Then, the put request acknowledgement returns to the client.



When data is inserted using put command.....

When we issue a Put command, the coordinates of the data are the row, the column, and the timestamp and when data is put into HBase, a timestamp is required. The timestamp can be generated automatically by the RegionServer or can be supplied by us and must be a long integer. The timestamp must be unique per version of a given cell, because the timestamp identifies the version. To modify a previous version of a cell, for instance, we would issue a Put command with a different value for the data itself, but the same timestamp.

Task2

Create an HBase table named **'clicks'** with a column family **'hits'** such that it should be able to store last 5 values of qualifiers inside **'hits'** column family.

Table name: ***clicks***

Column Family: ***hits***

Versions: 5

Command format:

Create <'table name'>,<'column family=>value'>,VERSIONS=>5

HBASE Shell command

hbase(main):007:0* create 'clicks',NAME=>'hits',VERSIONS=>5

```
hbase(main):006:0>
hbase(main):007:0* create 'clicks',NAME=>'hits',VERSIONS=>5
0 row(s) in 1.0230 seconds
```

```
hbase(main):010:0* list
TABLE
clicks
customer
studentAcad
3 row(s) in 0.0860 seconds
```

Task2

Add few records in the table and update some of them. Use IP Address as row-key. Scan the table to view if all the previous versions are getting displayed.

Here we are using ip address ***'27.59.2.31'*** as a row key,

HBASE shell command

put 'clicks','27.59.2.31','hits:No_Of_Times','12'

scan 'clicks'

```
hbase(main):014:0* put 'clicks','27.59.2.31','hits:No_Of_Times','12'
0 row(s) in 0.2190 seconds

hbase(main):015:0> scan 'clicks'
ROW                                   COLUMN+CELL
 27.59.2.31                           column=hits:No_Of_Times, timestamp=1511876988874, value=12
1 row(s) in 0.1310 seconds
```

Update the row by adding some values,

put 'clicks','27.59.2.31','hits:No_Of_Times','32'

put 'clicks','27.59.2.31','hits:No_Of_Times','8'

put 'clicks','27.59.2.31','hits:No_Of_Times','44'

put 'clicks','27.59.2.31','hits:No_Of_Times','99'

We are going to see the last 5 qualifiers inside hits column family,

scan 'clicks',{COLUMN=>'hits:No_Of_Times',VERSIONS=>5}

```
hbase(main):023:0> scan 'clicks',{COLUMN=>'hits:No_Of_Times',VERSIONS=>5}
ROW                                   COLUMN+CELL
 27.59.2.31                           column=hits:No_Of_Times, timestamp=1511877052223, value=99
 27.59.2.31                           column=hits:No_Of_Times, timestamp=1511877050801, value=44
 27.59.2.31                           column=hits:No_Of_Times, timestamp=1511877050655, value=8
 27.59.2.31                           column=hits:No_Of_Times, timestamp=1511877050455, value=32
 27.59.2.31                           column=hits:No_Of_Times, timestamp=1511876988874, value=12
1 row(s) in 0.1910 seconds
```

We can see the Ip address that how many number of times it was hits in a regular interval manner.