MODULE 3

Nosql BigData Management, MongoDB and Cassandra

Syllabus:

Introduction, NoSQL Data Store, NoSQL Data Architecture Patterns, NoSQL to Manage Big Data, Shared-Nothing Architecture for Big Data Tasks, MongoDB, Databases, Cassandra Databases.

Introduction:

Big Data uses distributed systems. A distributed system consists of multiple data nodes at clusters. The tasks execute in parallel with data at nodes in clusters. The computing nodes communicate with the applications through a network.

Features of distributed-computing architecture:

1. **Increased reliability and fault tolerance**. If a segment of machines in a cluster fails then the rest of the machines continue work. When the datasets replicate at a number of data nodes, the fault tolerance increases further.

2. **Flexibility** makes it very easy to install, implement and debug new services in a distributed

3. **Sharding** is storing the different parts of data onto different sets of data nodes, clusters or servers. For example, university students huge database, on sharding divides in databases, called shards. Each shard may correspond to a database for an individual course and year.

4**. Speed**: Computing power increases in a distributed computing system as shards run parallelly on individual data nodes in clusters independently.

5. **Scalability**: Consider sharding of a large database into a number of shards, distributed for computing in different systems. When the database expands further, then adding more machines and increasing the number of shards provides horizontal scalability. Increased computing power and running number of algorithms on the same machines provides vertical scalability.

6.**Resources sharing**: Shared resources of memory, machines and network architecture reduce the cost.

7. **Open system** makes the service accessible to all nodes.

8. **Performance:** The collection of processors in the system provides higher performance than a centralized computer, due to lesser cost of communication among machines.

The demerits of distributed computing:

(i) issues in troubleshooting in a larger networking infrastructure,

(ii) additional software requirements and

(iii) security risks for data and resources.

Nosql Data Store

Transactions on SQL databases exhibit ACID properties. ACID stands for atomicity, consistency, isolation and durability.

Atomicity of transaction means all operations in the transaction must complete, and if interrupted, then must be undone (rolled back). For example, if a customer withdraws an amount then the bank in first operation enters the withdrawn amount in the table and in the next operation modifies the balance with a new amount available. Atomicity means both should be completed, else undone if interrupted in between. Consistency in transactions means that a transaction must maintain the integrity constraint, and follow the **consistency** principle.

For example, the difference of sum of deposited amounts and withdrawn amounts in a bank account must equal the last balance. All three data need to be consistent.

Isolation of transactions means two transactions of the database must be isolated from each other and done separately.

Durability means a transaction must persist once completed.

Triggers, Views,Schedules Joins in SQL Databases:

Trigger is a special stored procedure. Trigger executes when a specific action(s) occurs within a database,such as change in table data or actions such as UPDATE, INSERT and DELETE.

View refers to a logical construct, used i 1 query statements. A View saves a division of complex query statement instructions and that reduces the query complexity. Viewing of a division is similar to a view of a table.

Schedule refers to a chronological sequence of instructions which execute concurrently. When a transaction is in the schedule then all instructions of the transaction are included in the schedule. Scheduled order of instructions is maintained during the transaction. Scheduling enables execution of multiple transactions in allotted time intervals.

Join refers to a clause which combines. Combining the products (AND operations) follows next the selection process. A Join operation does pairing of two tuples obtained from different relational expressions. Joins, if and only if a given Join condition satisfies.

SQL compliant format means that database tables constructed using SQL and they enable processing of the queries written using SQL. 'NoSQL term conveys two different meanings: (i) does not follow SQL compliant formats, (ii)"Not only SQL" use SQL compliant formats with a variety of other querying and access methods.

NoSQL

A new category of data stores is NoSQL (means Not Only SQL) data stores. NoSQL is an altogether new approach to thinking about databases, such as schema flexibility, simple relationships, dynamic schemas. auto sharding, replication, integrated caching, horizontal scalability of shards, distributable tuples, semi structured data and flexibility in approach. Issues with NoSQL data stores are lack of standardization in approaches, processing difficulties for complex queries.

Big Data NoSQL or Not-Only SQL

NoSQL data stores are considered semi-structured data. Big Data Store uses NoSQL

NoSQL or Not Only SQL is a Class of non-relational data storage systems, flexible data models and multiple schema with SQL.

NoSQL data store characteristics are as follows

1. NoSQL is a class of non-relational data storage system with flexible data model. Examples of NoSQL data-architecture patterns of datasets are key-value pairs, name/value pairs, Column family Big-data store, Tabular data store, Cassandra (used in Facebook/Apache), HBase, hash table [Dynamo (Amazon S3)], unordered keys using JSON (CouchDB), JSON (PNUTS), JSON (MongoDB), Graph Store, Object Store, ordered keys and semi-structured data storage systems.

2. NoSQL not necessarily has a fixed schema, such as table, do not use the concept of Joins, Data written at one node can be replicated to multiple nodes. Data store is thus fault-tolerant. The store can be partitioned into unshared shards.

Features in NoSQL Transactions

(i) Relax one or more of the ACID properties.

(ii) Characterize by two out of three properties (consistency, availability and partitions) of CAP theorem.

(iii) Can be characterized by BASE properties

CAP Theorem :

Among C,A and P two are at least present for the application service process.

Consistency means all copies have the same value like in traditional DB,***Availabilit***y means at least one copy is available in case a partition becomes active or fails, ***Partition*** means parts which are active but may not cooperate (share) as in distributed DBs.

1.Consistency in distributed database means that all nodes observe the same data at the same time. Therefore, the operations in one partition of the database should reflect in other related partitions in case of distributed database Operations, which change the sales data from a specific showroom in a table should also reflect in changes in related tables which are using that sales data.

2. Availability means that during the transactions, the field values must be available in other partitions of the database so that each request receives a response on success as well as failure. Replication ensures availability.

3 Partition means division of a large database into different databases without affecting the operations on them by adopting specified procedures.

Partition tolerance: Refers to continuation of operations as a whole even in case of message loss, node failure or node not reachable

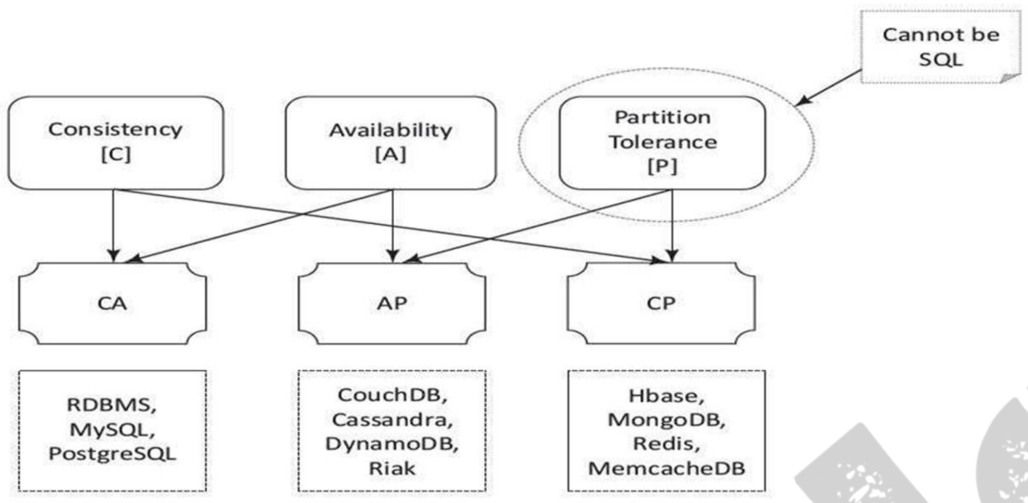
Brewer's CAP (Consistency. Availability and Partition Tolerance) theorem demonstrates that any distributed system cannot guarantee C. A and P together.

1. Consistency-All nodes observe the same data at the same time.

2 Availability-Each request receives a response on success failure.

3 Partition Tolerance-The system continues to operate as a whole even in case of message loss, node failure or node not reachable.In case of any network failure, a choice can be:

- Database must answer, and that answer would be old or wrong data (AP)
- Database should not answer, unless it receives the latest copy of the data (CP).



The CAP theorem implies that for a network partition system, the choice of consistency and availability are mutually exclusive. CA means consistency and availability, AP means availability and partition tolerance and CP means consistency and partition tolerance.

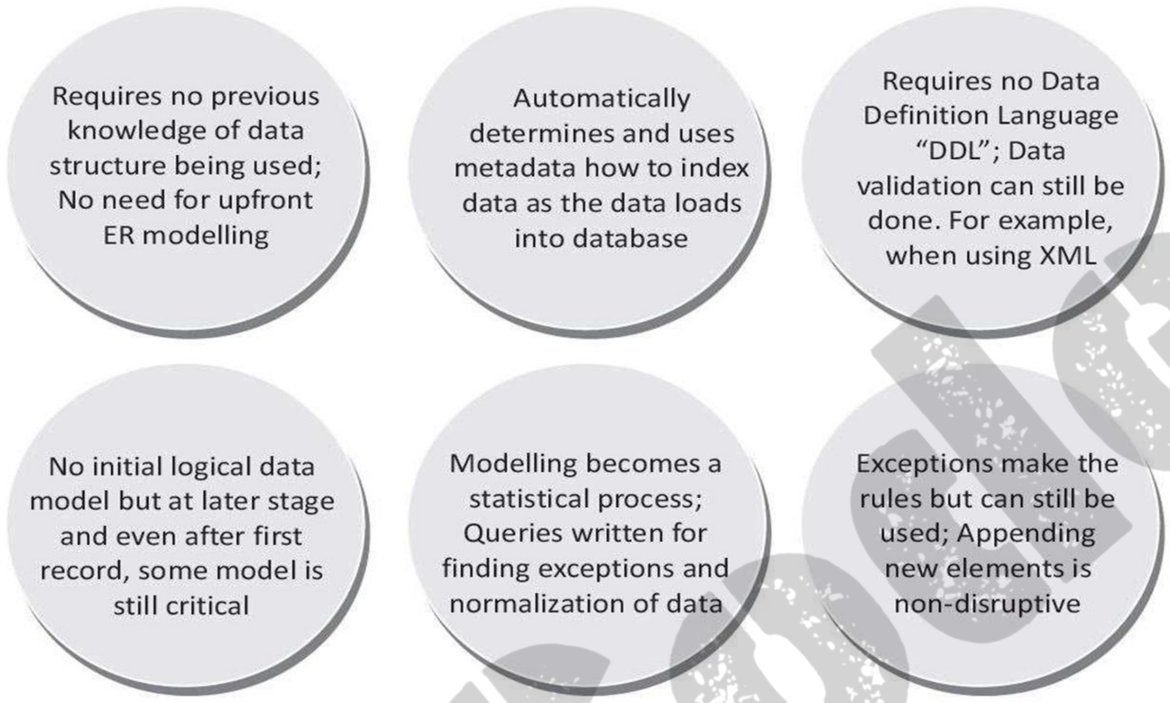
Schema-less Models

Schema of a database system refers to designing of a structure for datasets and data structures for storing into the database. NoSQL data necessarily have a fixed table schema. The systems not the concept Join (between distributed datasets). cluster-based highly distributed node manages single large data store with a NoSQL DB.

Relations in a database build the connections between various tables of data. For example, a table of subjects offered in an academic programme can be connected to a table of programmes offered in the academic institution. NoSQL data stores use non-mathematical relations but store this information as an aggregate called **metadata**

Metadata refers to data describing specifying an object or is the information a particular dataset the inter-linkages.

Characteristics of schema-less model


Requires no previous knowledge of data structure being used; No need for upfront ER modelling
Automatically determines and uses metadata how to index data as the data loads into database
Requires no Data Definition Language "DDL"; Data validation can still be done. For example, when using XML
No initial logical data model but at later stage and even after first record, some model is still critical
Modelling becomes a statistical process; Queries written for finding exceptions and normalization of data
Exceptions make the rules but can still be used; Appending new elements is non-disruptive

Increasing Flexibility for Data Manipulation:

Now consider students' admission database. That follows a fixed schema. Later, additional data is added as the course progresses. NoSQL data store characteristics are schema-less. The additional data may not be structured and follow fixed schema. NoSQL data store possess characteristic of increasing flexibility for data manipulation. The new attributes to database can be increasingly added.

BASE is a flexible model for NoSQL data stores. Provisions of BASE increase flexibility.

BASE Properties:

1. (BA)Basic availability ensures by distribution of shards (many partitions of huge data store) across many data nodes with a high degree of replication. Then, a segment failure does not necessarily mean a complete data store unavailability.

2. Soft state ensures processing even in the presence of inconsistencies but achieving consistency eventually.

3. Eventual consistency means consistency requirement in NoSQL databases meeting at some point of time in future.

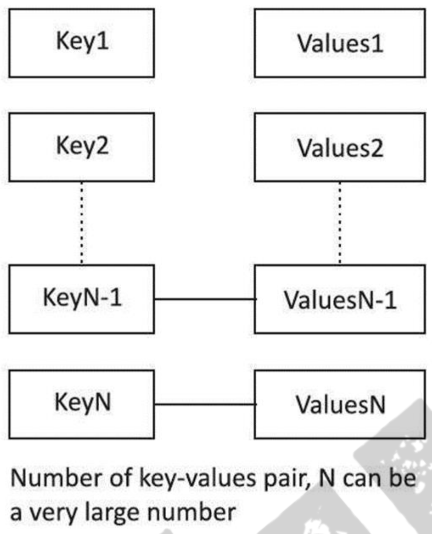
ACID rules require consistency all along the processing on completion of each transaction. **BASE** does not have that requirement and has the flexibility.

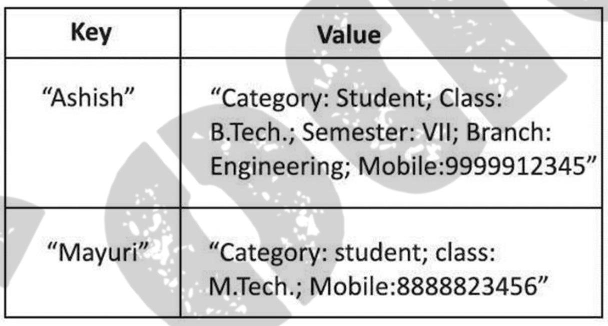
Nosql Data Architecture Patterns:

Key-value pair:

The simplest way to implement a schema-less data store is to use key-value pairs. The data store characteristics are high performance, scalability and flexibility. Data retrieval is fast in Key value pairs data store, A simple string called, key maps to a large data string or BLOB (Basic Large Object). Key value store accesses use a primary key for accessing the values. Therefore, the store can be easily scaled up for very large data.

Example of Key-value pairs in data architectural pattern





Key	Value
"Ashish"	"Category: Student; Class: B.Tech.; Semester: VII; Branch: Engineering; Mobile:9999912345"
"Mayuri"	"Category: student; class: M.Tech.; Mobile:8888823456"

Advantages of a key-value store are as follows:

1.Data Store can store any data type in a value field. The key-value system stores the information as a BLOB of data (such as text, hypertext, images, video and audio).

2.A query just requests the values and returns the values as a single item.

3.Key-value store is eventually consistent.

4.Key—value data store may be hierarchical or may be ordered key—value store.

5.Returned values on queries can be used to convert into lists, table- columns, data-frame fields and columns

6.Have (i) scalability, (ii) reliability, (iii) portability and (iv) low operational cost.

7.The key can be synthetic or auto-generated.

The key-value store provides client to read and write values using a key as follows:

Get(key)-->returns the value associated with the key.

Put(key,value)-->associates the value with the key and updates a value if this key is already present.

Multi-get(keyl,key2,...,keyN)-->returns the list of values associated with the list of keys.

Delete(key)-->removes a key and its value from the data store.

Limitations of key-value store architectural pattern are:

i.No indexes are maintained on values, thus a subset of values is not searchable.

ii.Key-value store does not provide traditional database capabilities.

iii.Maintaining unique values as keys may become more difficult when the volume of data increases.

iv.Queries cannot be performed on individual values. No clause like 'where' in a relational database usable that filters a

Typical uses of key-value store are: (i) image store, (ii) Document or file store, (iii)Lookup table and query-cache.

Document Store:

Following are the features in Document Store:

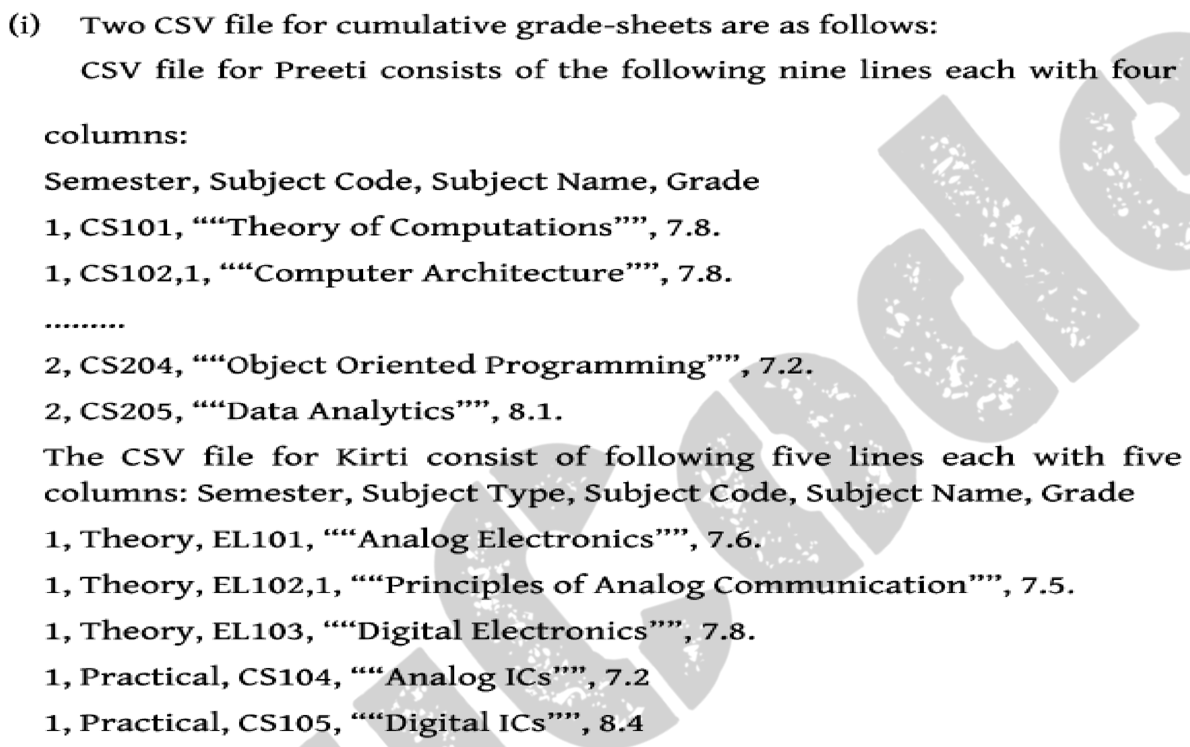
1. Document stores unstructured data.
2. Storage has similarity with object store.
3. Data stores in nested hierarchies. Hierarchical information stores in a single unit called document tree.
4. Querying is easy.
5. No object relational mapping enables easy search by following paths from the root of document tree.
6. Transactions on the document store exhibit ACID properties.

Typical uses of a document store are: (i) office documents, (ii) inventory store, (iii) forms data,(iv) document exchange and (v) document search.

The demerits in Document Store are incompatibility with SQL and complexity for implementation. Examples of Document Data Stores are CouchDB and MongoDB.

CSV and JSON File Formats

CSV data store is a format for records CSV does not represent object-oriented databases or hierarchical data records. JSON and XML represent semistructured data, object-oriented records and hierarchical data records. JSON (Java Script Object Notation) refers to a language format for semistructured data, JSON represents object-oriented and hierarchical data records.



(i) Two CSV file for cumulative grade-sheets are as follows:
CSV file for Preeti consists of the following nine lines each with four columns:

Semester, Subject Code, Subject Name, Grade
1, CS101, ““Theory of Computations””, 7.8.
1, CS102,1, ““Computer Architecture””, 7.8.
....
2, CS204, ““Object Oriented Programming””, 7.2.
2, CS205, ““Data Analytics””, 8.1.

The CSV file for Kirti consist of following five lines each with five columns: Semester, Subject Type, Subject Code, Subject Name, Grade
1, Theory, EL101, ““Analog Electronics””, 7.6.
1, Theory, EL102,1, ““Principles of Analog Communication””, 7.5.
1, Theory, EL103, ““Digital Electronics””, 7.8.
1, Practical, CS104, ““Analog ICs””, 7.2
1, Practical, CS105, ““Digital ICs””, 8.4

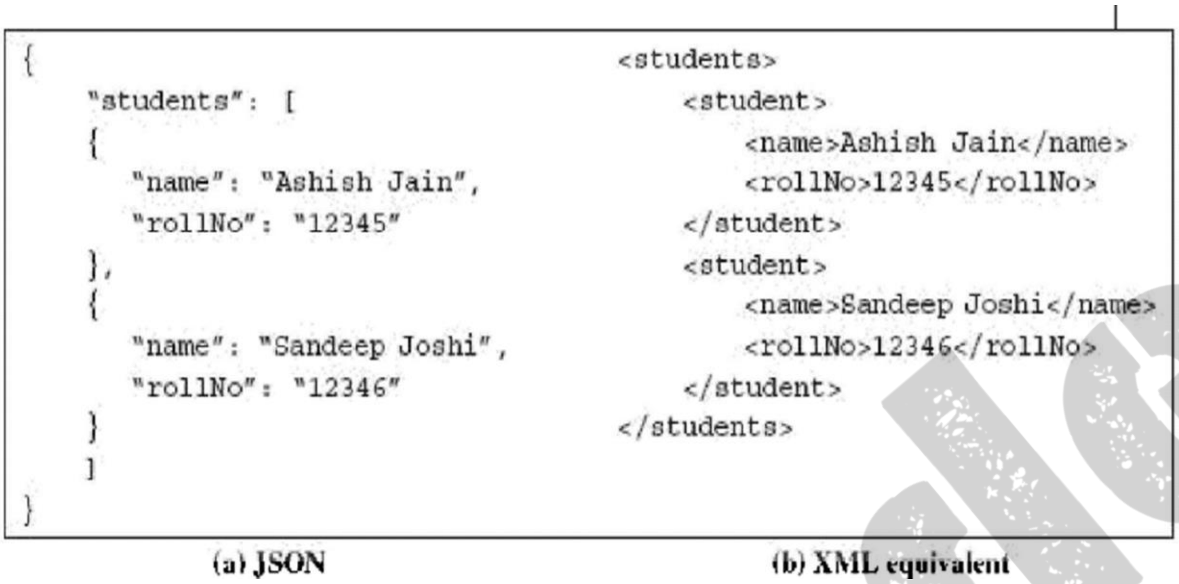
JSON example:

```
{
    "employee": {
        "name":      "sonoo",
        "salary":     56000,
        "married":    true
    }
}
```

XML (eXtensible Markup Language) : Is an extensible, simple and scalable language.

- Its self-describing format describes structure and contents in an easy to understand format. XML is widely used.
- The document model consists of root element and their sub-elements.
- XML document model has a hierarchical structure.

● XML document model has features of object-oriented records.



```
{                                            <students>
    "students": [                               <student>
    {                                              <name>Ashish Jain</name>
      "name": "Ashish Jain",                       <rollNo>12345</rollNo>
      "rollNo": "12345"                         </student>
    },                                          <student>
    {                                              <name>Sandeep Joshi</name>
      "name": "Sandeep Joshi",                     <rollNo>12346</rollNo>
      "rollNo": "12346"                         </student>
    }                                        </students>
    ]
}
```

(a) JSON (b) XML equivalent

Document JSON Format—MongoDB Database

Example of document in Document store:

{

“id”: “1001”

“Student Name”:

{

“First”: “Ashish”,

“Middle”: “Kumar”

“Last”: “Rai”

}

“Category”: “Student”,

“Class”: “B.Tech”,

“Semester”: “VII”,

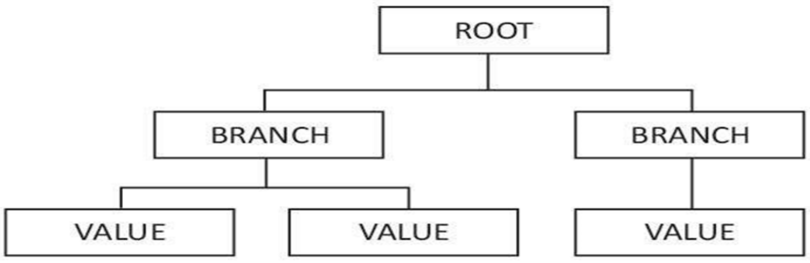
“Branch”: “Computer Engineering”,

“Mobile”: “12345”

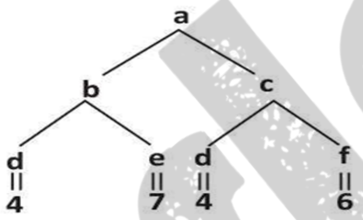
}

The document store allows querying the data based on the contents as well. For example, it is possible to search the document where student's first name is "Ashish". Document store can also provide the search value's exact location. The search is by using the document path. A type of key accesses the leaf values in the tree structure. Since the document stores are schema-less, adding fields to documents (XML or JSON) becomes a simple task.

Document Architectural pattern and discovering Hierarchical Structure:



```
<a>
   <b>
         <d> 4 </d>
         <e> 7 </e>
   </b>
   <c>
         <d> 4 </d>
         <f> 6 </f>
   </c>
</a>
```



(a) XML document fragment (b) Tree representation of fragment

The document store follows a tree-like structure (similar to directory structure in file system). Beneath the root element there are multiple branches. Each branch has a related path expression that provides a way to navigate from the root to any given branch, sub-branch or value.

XQuery and XPath are query languages for finding and extracting elements and attributes from XML documents. The query commands use sub-trees and attributes of documents. The querying is similar as in SQL for databases. XPath treats XML document as a tree of nodes. XPath queries are expressed in the form of XPath expressions. Following is an example of XPath expressions:

When compared with XML, JSON has the following advantages:

- ➢ XML is easier to understand but XML is more verbose than JSON.
- ➢ XML is used to describe structured data and does not include arrays, whereas JSON includes arrays.
- ➢ JSON has basically key-value pairs and is easier to parse from JavaScript.
- ➢ The concise syntax of JSON for defining lists of elements makes it preferable for serialization of text format objects.

Document Collection

A collection can be used in many ways for managing a large document store. Three uses of a document collection are:

1. Group the documents together, similar to a directory structure in a file-system.
2. Enables navigating through document hierarchies, logically grouping similar documents and storing business rules such as permissions, indexes and triggers.
3. A collection can contain other collections as well.

Column Family store:

Columnar Data Store

A way to implement a schema is the divisions into columns. Storage of each column, successive values is at the successive memory addresses. Analytics processing (AP) In-memory uses columnar storage in memory. A pair of row-head and column-head is a key-pair. The pair accesses a field in the table.

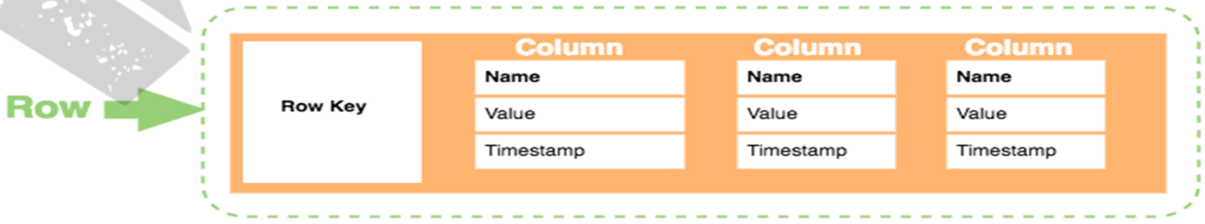
Column-Family Data Store

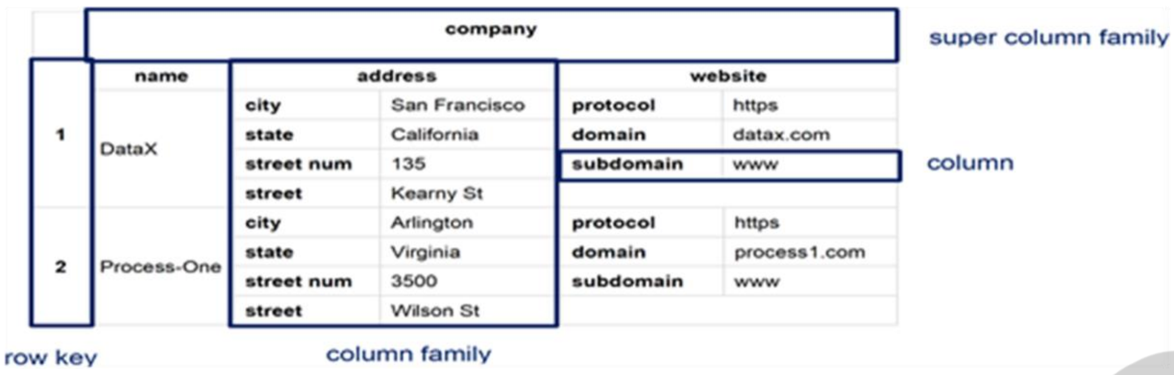
Column-family data-store has a group of columns as a column family. A combination of row-head, column-family head and table-column head can also be a key to access a field in a column of the table during querying. Combination of row head, column families head, column-family head and column head for values in column fields can also be a key to access fields of a column. A column-family head is also called a super-column head.

Examples of columnar family data stores are HBase, BigTable, HyperTable and Cassandra.

Columns Families Two or more columns in data-store group into one column family.

Grouping of Column Families Two or more column-families in data store form a super group, called super column.




super column family
column
row key
column family

	company				
	name	address		website	
1	DataX	city	San Francisco	protocol	https
		state	California	domain	datax.com
		street num	135	subdomain	www
		street	Kearny St		
2	Process-One	city	Arlington	protocol	https
		state	Virginia	domain	process1.com
		street num	3500	subdomain	www
		street	Wilson St		

.

Characteristics of Columnar Family Data Store

- ➔ high performance and scalability,
- ➔ moderate level of flexibility and
- ➔ lower complexity when compared to the object and graph databases.

Advantages of column stores are:

1. **Scalability:** The back-end system can distribute queries over a large number of processing nodes without performing any Join operations.
2. **Portitionability:** large data can be partitioned into datasets of smaller MB size.
3. Availability: The cost of replication is lower since the system scales on distributed nodes efficiently.
4. **Tree-like columnar structur**e: consisting of column-family groups, column families and columns. The columns group into families. The column families group into column groups (super columns). A key for the column fields consists of three secondary keys: column-families group ID, column- family ID and column-head name.
5. **Adding new data at ease**: Permits new column insert operations.
6. **Querying all the held values**: in a column in a family, all columns in the family or a group of column-families, is fast in in-memory column-family data store.
7. **Replication of columns**: HDFS-compatible column-family data stores replicate each data store with default replication factor = 3.
8. **No optimization for Join:** Column-family data stores are similar to sparse matrix data. The data do not optimize for Join operations

Typical uses of column store are: (i). Web crawling, (ii)large sparsely populated tables (iii) system that has high varience

BigTable Data Store:

Following are features of a BigTable:

1. Massively scalable NoSQL. BigTable scales up to 100s of petabytes.

2. Integrates easily with Hadoop and Hadoop compatible systems.

3. Compatibility with MapReduce, HBase APIs which are open-source Big Data platforms.

4. Key for a field uses not only row_ID and Column_ID but also timestamp and attributes. Values are ordered bytes. Therefore, multiple versions of values may be present in the BigTable.

5. Handles million of operations per second.

6. Handle large workloads with low latency and high throughput

7. Consistent low latency and high throughput

8. APIs include security and permissions

9. BigTable, being Google's cloud service, has global availability and its service is seamless.

Object Data Store:

As object refers to a repository which stores the

1 Objects(such as files, images, documents, folders, and business reports)

2.System metadata which provides information such as filename, creation date, last modified language used (such as Java, C, C#, C++, Smalltalk, Python), access permissions, supported query

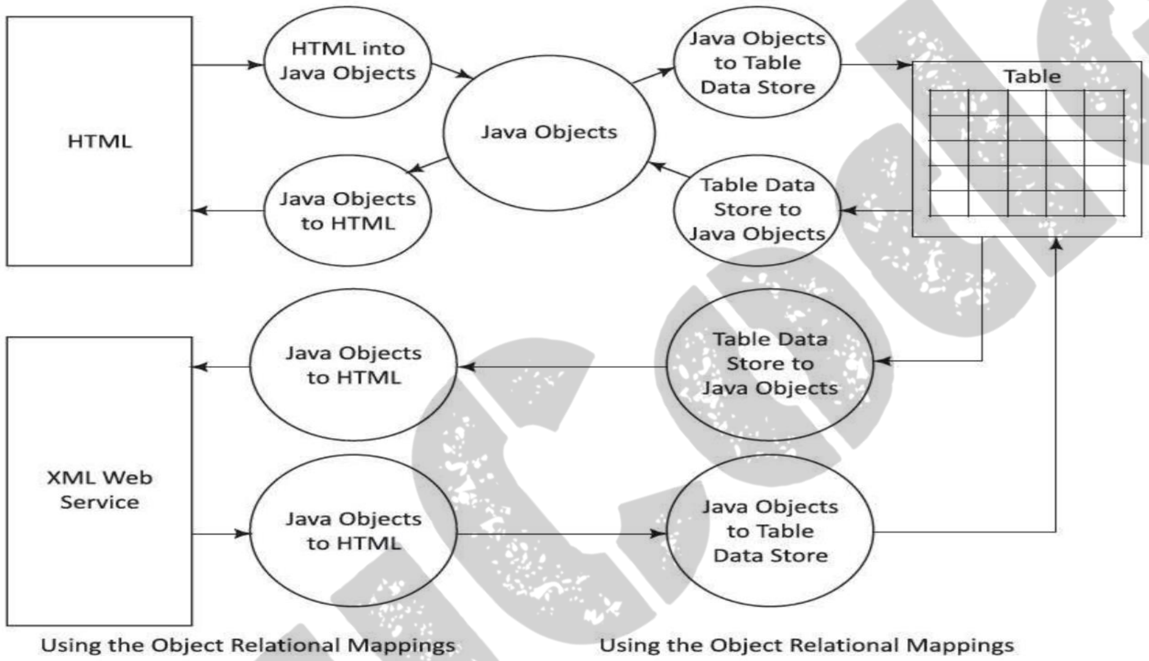
3 Custom metadata which provides information, such as subject, category, sharing permissions.

Eleven Functions Supporting APIs An Object data store consists of functions supporting APIs for (i)scalability, (ii) indexing. (iii) large collections, (iv) querying language, processing and optimization (s). (v) Transactions, (vi) data replication for high availability, data distribution model, data integration (vii) schema evolution, (viii) persistency. (ix) persistent object life cycle, (x) adding modules and (xi) locking and caching strategy.

Amazon S3 (Simple Storage Service) S3 refers to Amazon web service on the cloud named S3. The S3 provides the Object Store. The Object Store offers from the block and file-based cloud storage. Objects along with their metadata store for each object store as the files. S3 assigns an ID number for each stored object. The service has two storage classes: Standard and infrequent access. Interfaces for S3 service are REST, SOAP and Bit Torrent. S3 uses include web hosting, image hosting and storage for backup systems. S3 is scalable storage infrastructure, same as used in Amazon e-commerce service. S3 may store trillions of Objects.

Object Relational Mapping:

Object relational mapping of HTML document and XML web service store with the tabular data store:


HTML
HTML into Java Objects
Java Objects to HTML
Java Objects
Java Objects to Table Data Store
Table Data Store to Java Objects
Table
XML Web Service
Java Objects to HTML
Table Data Store to Java Objects
Java Objects to HTML
Java Objects to Table Data Store

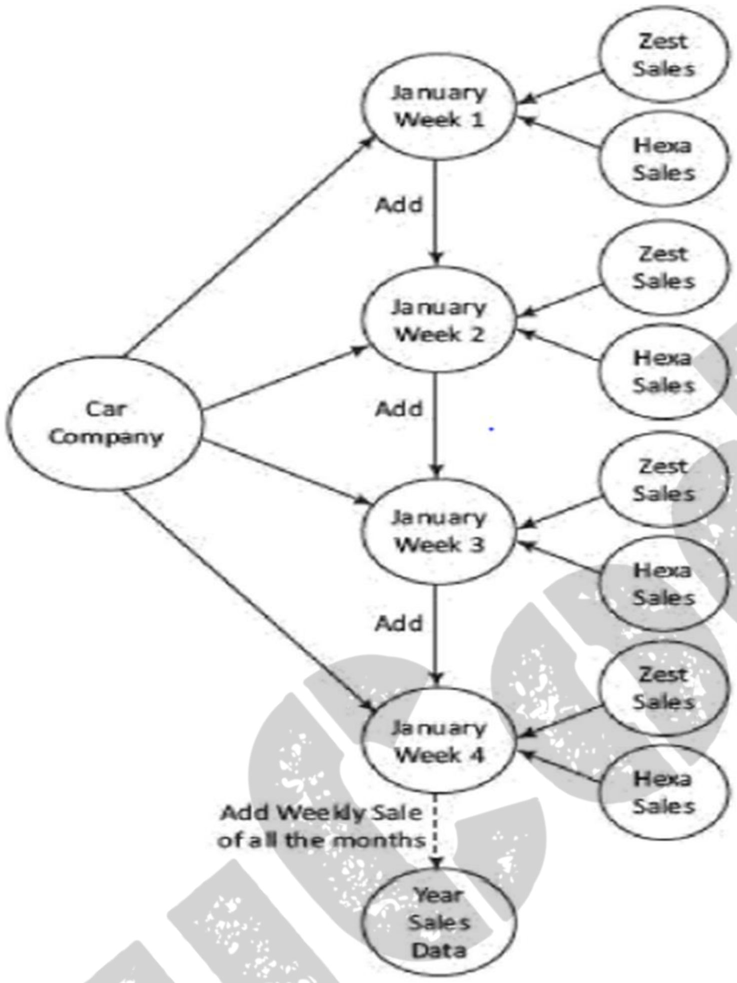
Using the Object Relational Mappings Using the Object Relational Mappings

Graph Database:

A characteristic of graph is high flexibility. Any number of nodes and any number of edges can be added to expand a graph. The complexity is high and the performance is variable with scalability. Data store as series of interconnected nodes. Graph with data nodes interconnected provides one of the best database system when relationships and relationship types have critical values.

Data Store focuses on modeling interconnected structure of data.Nodes represent entities or objects. Edges encode relationships between nodes. Some operations become simpler to perform using graph models. Examples of graph model usages are social networks of connected people. The connections to related persons become easier to model when using the graph model.

The following example explains the graph database application in describing entities relationships and relationship types.



Characteristics of graph databases are:

1. Use specialized query languages, such as RDF uses SPARQL

2. Create a database system which models the data in a completely different way than the key-values, document, columnar and object data store models.

3. Can have hyper-edges. A hyper-edge is a set of vertices of a hypergraph. A hypergraph is a generalization of a graph in which an edge can join any number of vertices

4. Consists of a collection of small data size records, which have complex interactions between graph-nodes and hypergraph nodes.

Typical uses of graph databases are: (i) link analysis, (ii) friend of friend queries, (iii) Rules and inference, (iv) rule induction and (v) Pattern matching. Link analysis is needed to perform searches and look for patterns and relationships in situations, such as social networking, telephone, or email records .Rules and inference are used to run queries on complex structures such as class libraries, taxonomies and rule-based systems.

Examples of graph DBs are Neo4J, AllegroGraph, HyperGraph, Infinite Graph, Titan and FlockDB. Neo4]

<u>NoSQL TO MANAGE BIGDATA:</u>

Using NoSQL to manage Big data:

NoSQL (i) limits the support for Join queries, supports sparse matrix like columnar-family, (ii) characteristics of easy creation and high processing speed, scalability and storability of much higher magnitude of data (terabytes and petabytes).

NoSQL sacrifices the support of ACID properties, and instead supports CAP and BASE properties NoSQL data processing scales horizontally as well vertically.

NoSQL Solutions for Big Data

Big Data solution needs scalable storage of terabytes and petabytes, dropping of support for database Joins, and storing data differently on several distributed servers (data nodes) together as a cluster.

Characteristics of Big Data NoSQL solution are:

1. High and easy scalability: NoSQL data stores are designed to expand horizontally. Horizontal scaling means that scaling out by adding more machines as data nodes (servers) into the pool of resources (processing, memory, network connections). The design scales out using multi-utility cloud services.

2. Support to replication: Multiple copies of data store across multiple nodes of a cluster. This ensures high availability, partition, reliability and fault tolerance.

3. Distributable:Big Data solutions permit sharding and distributing of shards on multiple clusters which enhances performance and throughput.

4. **Usages of NoSQL servers** which are less expensive. NoSQL data stores require less management efforts. It supports many features like automatic repair, easier data distribution and simpler data models that makes database administrator (DBA) and tuning requirements less stringent.

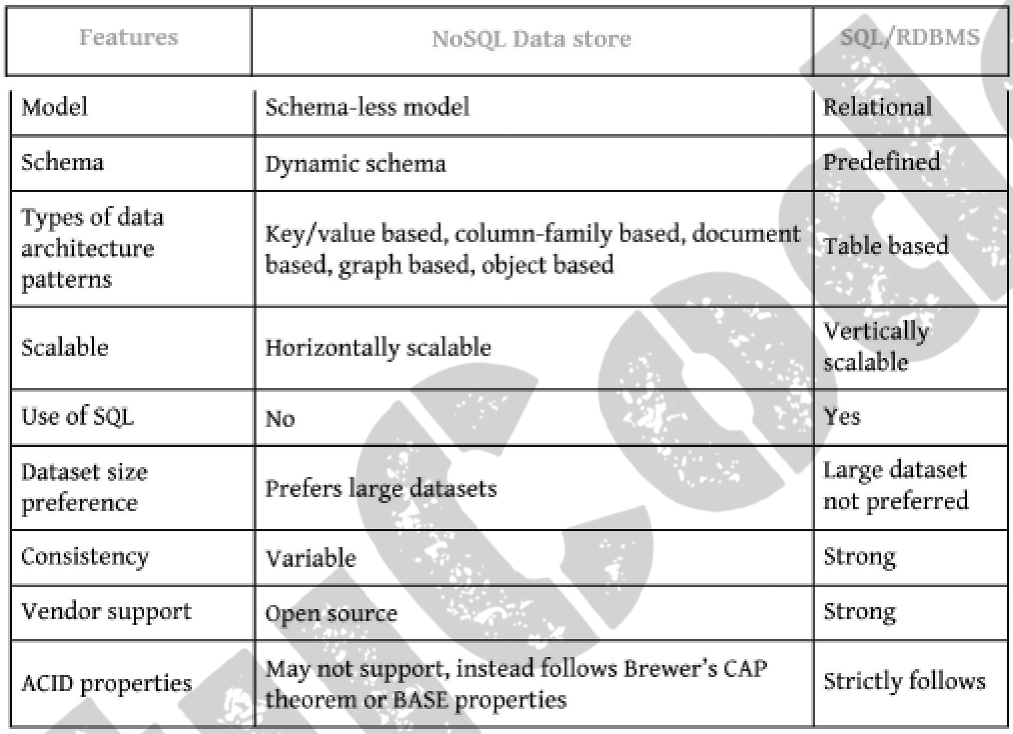
5. Usages of open-source tools: NoSQL data stores are cheap and open source. Database implementation is easy and typically uses cheap servers to manage the exploding data and transaction while RDBMS database are expensive and use big servers and storage system

6.Support to schema-less data model: NoSQL data store is schema less, so data can be inserted in a NoSQL data store without any predefined schema.

7. Support to integrated caching: NoSQL data store support the caching in system memory. That increases output performance. SQL database needs a separate infrastructure for that.

8. No inflexibility unlike the SQL/RDBMS,NoSQL DBs are flexible (not rigid) and have no structured way of storing and manipulating data..

<u>Comparision of NoSQL with SQL</u>



Features	NoSQL Data store	SQL/RDBMS
Model	Schema-less model	Relational
Schema	Dynamic schema	Predefined
Types of data architecture patterns	Key/value based, column-family based, document based, graph based, object based	Table based
Scalable	Horizontally scalable	Vertically scalable
Use of SQL	No	Yes
Dataset size preference	Prefers large datasets	Large dataset not preferred
Consistency	Variable	Strong
Vendor support	Open source	Strong
ACID properties	May not support, instead follows Brewer's CAP theorem or BASE properties	Strictly follows

Shared-Nothing Architecture for Big Data Tasks:

Shared nothing (SN) is cluster architecture. A node does not share data with any other node. Shared nothing architecture is an architecture which is used in distributed computing in which each node is independent and different nodes are interconnected by a network. A partition processes the different queries on data of the different users at each node independently.

Big Data store consists of SN architecture. Big Data store, therefore, easily partitions into shards. A partition processes the different queries on data of the different users at each node independently. Thus, data processes run in parallel at the nodes.

Data of different data stores partition among the number of nodes. Processing may require every node to maintain its own copy of the application's data, using a coordination protocol.

EX: Hadoop, Flink and Spark.

The features of SN architecture are as follows:

- *Independence:* Each node with no memory sharing; thus possesses computational self-sufficiency
- Self-Healing: A link failure causes **creation** of another link
- Each node Jitnctioninq os a shnrd: Each node stores a shard (a partition of large DBs)
- No network contention.

Choosing the Distribution Models

Big Data requires distribution on multiple data nodes at clusters. Distributed software components give advantage of parallel processing; thus providing horizontal scalability.

Distribution gives (i) ability to handle large-sized data

(ii) processing of many read and write operations simultaneously in an application.

Four models for distribution of the data stores:

1) *Single Server* Model

Simplest distribution option for NoSQL data store and access is Single Server Distribution (SSD) of an application.

A graph database processes the relationships between nodes at a server. The SSD model suits well for graph DBs. An application executes the data sequentially on a single server.

Following figure shows the SSD model.

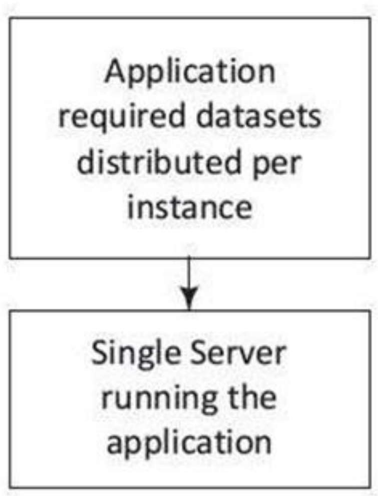

Application required datasets distributed per instance
Single Server running the application

Figure: single server model

2) Sharding Very Large Datnboses:

- The application programming model in SN architecture is such that an application process runs on multiple shards in parallel.
- Sharding provides horizontal scalability. A data store may add an auto-sharding feature.
- The performance improves in the SN. However, in case of a link failure with the application, the application can migrate the shard DB to another node.

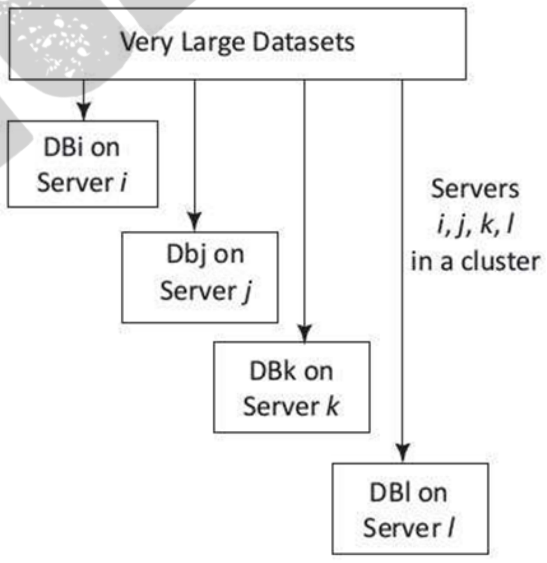


Figure: Shards distributed on four servers in a cluster

Above figure shows sharding of very large datasets into four divisions, each running the application on four i,j, k and I different servers at the cluster. DB_i, DBMS DB_k and DB_l are four shards.

3) Master-Slave Distribution Model

A node serves as a master or primary node and the other nodes are slave nodes. Master directs the slaves. Slave nodes data replicate on multiple slave servers in Master Slave Ditribution (MSD) model.

When a process updates the master, it updates the slaves also. A process uses the slaves for read operations. Processing performance improves when process runs large datasets distributed onto the slave nodes.

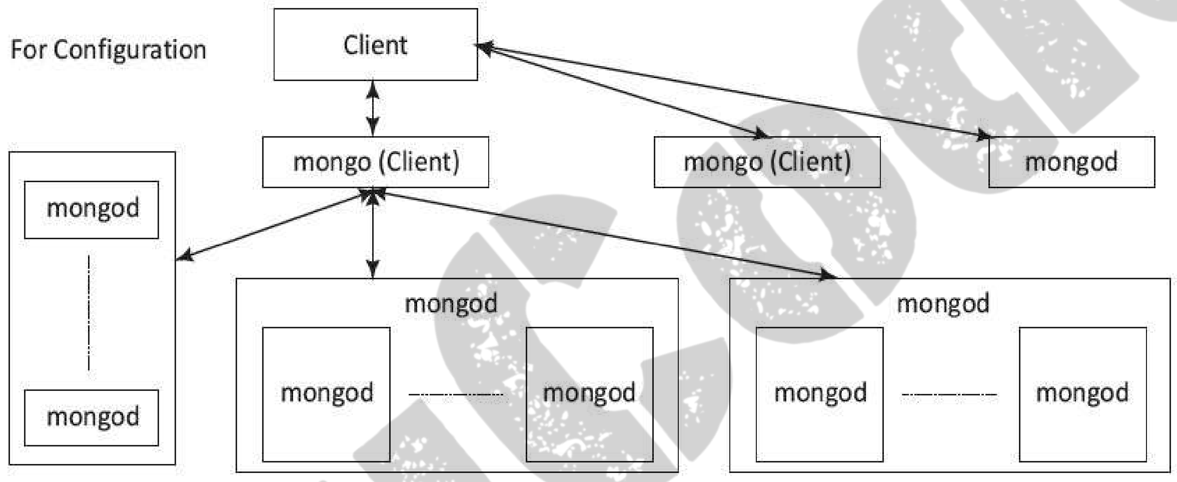


Figure: Master-slave distribution model. Mongo is a client and mongod is the server

Master-Slave Replication Processing performance decreases due to replication in MSD distribution model.

Complexity Cluster-based processing has greater complexity than the other architectures.

4) Peer-to-Peer Distribution Model

Peer-to-Peer distribution (PPD) model and replication show the following characteristics:

(1) All replication nodes accept read request and send the responses.

(2) All replicas function equally.

(3) Node failures do not cause loss of write capability

Cassandra adopts the PPD model. The data distributes among all the nodes in a cluster. Performance can further be enhanced by adding the nodes. Since nodes read and write both, a replicated node also has updated data. Therefore, the biggest advantage in the model is consistency.

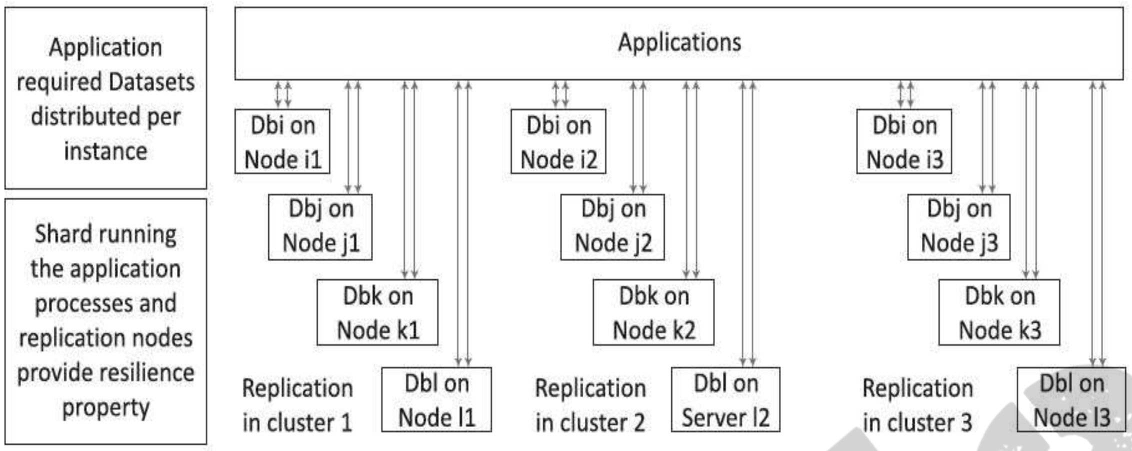


Figure: Shards replicating on the nodes, which does read and write operations both

Ways of Handling Big Data Problems:

1. **Evenly distribute the data on n *cluster* using the hash rings:** Consistent hashing refers to a process where the datasets in a collection distribute using a hashing algorithm which generates the pointer for a collection. Using only the hash of Collection_ID, a Big Data solution client node determines the data location in the cluster. Hash Ring refers to a map of hashes with locations. The client, resource manager or scripts use the hash ring for data searches and Big Data solutions. The ring enables the consistent assignment and usages of the dataset to a specific processor.

2. ***Use replication* to horizontally *distribute* the client read-requests**: Replication means creating backup copies of data in real time. Many Big Data clusters use replication to make the failure-proof retrieval of data in a distributed environment. Using replication enables horizontal scaling out of the client requests.

3. ***Moving queries to* the dnto, not the dntn *to* the *queries:*** Most NoSQL data stores use cloud utility services (Large graph databases may use enterprise servers). Moving client node queries to the data is efficient as well as a requirement in Big Data solutions.

4. **Queries distribution to multiple nodes**: Client queries for the DBs analyze at the anazers, which evenly distribute the queries to data nodes/ replica nodes. High performance query processing requires usages of multiple nodes. The query execution takes place separately from the query evaluation.

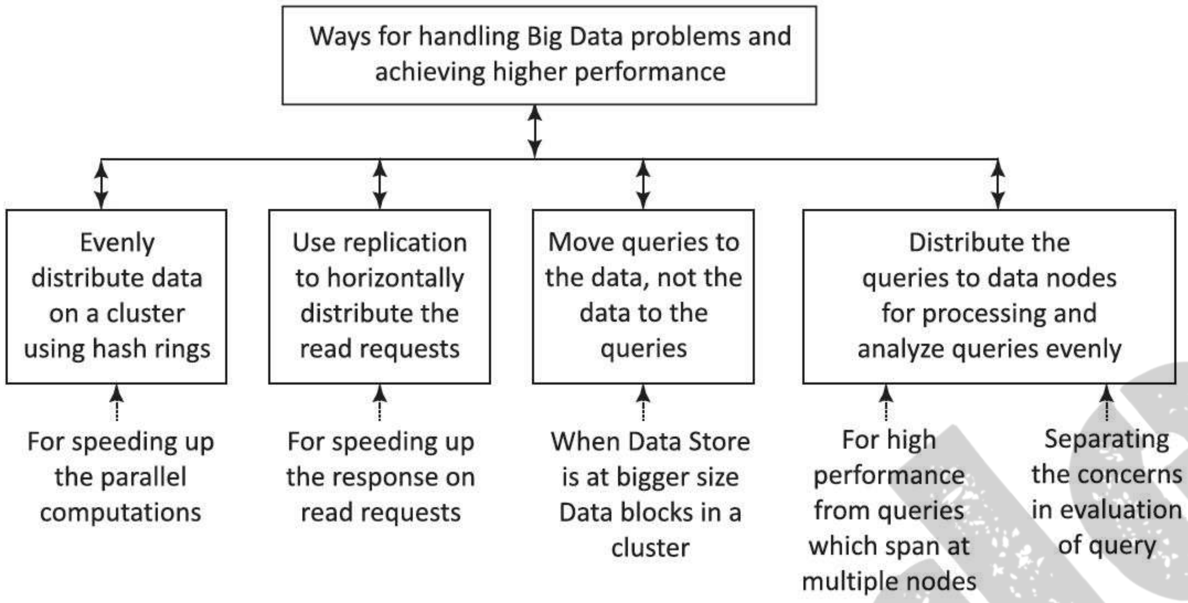

Ways for handling Big Data problems and achieving higher performance
Evenly distribute data on a cluster using hash rings
Use replication to horizontally distribute the read requests
Move queries to the data, not the data to the queries
Distribute the queries to data nodes for processing and analyze queries evenly
For speeding up the parallel computations
For speeding up the response on read requests
When Data Store is at bigger size Data blocks in a cluster
For high performance from queries which span at multiple nodes
Separating the concerns in evaluation of query

Figure: Four ways for handling big data problems

Mongodb Database:

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc.

It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.

- MongoDB is an open source DBMS. MongoDB programs create and manage databases.
- MongoDB manages the collection and document data store.
- MongoDB functions do querying and accessing the required information.
- The functions include viewing, querying, changing, visualizing and running the transactions. Changing includes updating, inserting, appending or deleting.

Characteristics of MongoDB are:

(i) non-relational, (ii) NoSQL, (iii) distributed, (iv) open source, (v) document based, cross-platform, (vii) Scalable, (viii) flexible data model, (ix) Indexed, (x) multi-master (xi) fault tolerant.

Features of MongoDB:

1. MongoDB data store is a physical container for collections. Each DB gets its own set of files on the file system. A number of DBs can run on a single MongoDB server. DB is default DB in MongoDB that stores within a data folder. The database server of MongoDB is mongod and the client is monqo.
2. Collection stores a number of MongoDB documents. It is analogous to a table of RDBMS. A collection exists within a single DB to achieve a single purpose. Collections may store documents that do not have the same fields. Thus, documents of the collection are schema-less.
3. Document model is well defined. Structure of document is clear; Document is the unit of storing data in a MongoDB database. Documents are analogous to the records of RDBMS table. Insert, update and delete operations can be performed on a collection. Document use JASON (JavaScript Object Notation) approach for storing data.
4. MongoDB is a document data store in which one collection holds different documents. Data store in the form of JSON-style documents.
5. Storing of data is flexible, and data store consists of JSON-like documents. This implies that the fields can vary from document to document and data structure can be changed over time.
6. Storing of documents on disk is in BSON serialization format. BSON is a binary representation of JSON documents.
7. Querying, indexing, and real time aggregation allows accessing and analyzing the data efficiently.
8. Deep query-ability—Supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
9. No complex Joins.
10. Distributed DB makes availability high, and provides horizontal scalability.

Comparison of RDBMS and MongoDB databases:

RDBMS	MongoDB
Database	**Data store**
Table	**Collection**
Column	**Key**
Value	**Value**
Records/Rows/ Tuple	**Document / Object**
Joins	**Embedded Documents**
Index	**Index**
Primary key	**Primary key (id) is default key provided by MongoDB itself**

Replication in MongoDB:

Replication ensures high availability in Big Data. Presence of multiple copies increases on different database servers. This makes DBs fault- tolerant against any database server failure.

MongoDB replicates with the help of a replica set. A replica set in MongoDB is a group of mongod (MongoDb server) processes that store the same dataset. Replica sets provide redundancy but high availability. A replica set usually has minimum three nodes. Any one out of them is called primary. The primary node receives all the write operations. All the other nodes are termed as secondary. The data replicates from primary to secondary nodes. A new primary node can be chosen among the secondary nodes at the time of automatic failover or maintenance. The failed node when recovered can join the replica set as secondary node again.

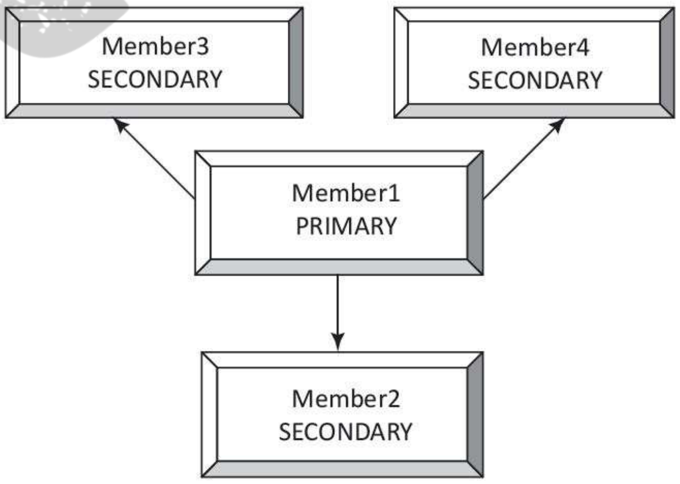


Figure: Replicated set on creating secondary members

Commands	Description
rs.initiate()	To initiate new replica set
rs.config()	To check the replica set configuration
rs.status()	To check the status of replica set
rs.add()	To add members to replica set

Auto-sharding: Sharding is a method for distributing data across multiple machines in a distributed application environment. MongoDB uses sharding to provide services to Big Data applications.

- A single machine may not be adequate to store the data. When the data size increases, do not provide data retrieval operation. Vertical scaling by increasing the resources of a single machine is quite expensive.
- Thus, horizontal scaling of the data can be achieved using sharding mechanism where more database servers can be added to support data growth and the demands of more read and write operations.
- A shard stores lesser data than the actual data and handles lesser number of operations in a single instance. For example, to insert data into a collection, the application needs to access only the shard that contains the specified collection. A cluster can thus easily increase its capacity horizontally.

MongoDB data types:

Double	**Represents a float value.**
String	**UTF-8 format string.**
Object	**Represents an embedded document.**
Array	**Sets or lists of values.**
Binary data	**String of arbitrary bytes to store images, binaries.**

Object id	ObjectIds (MongoDB document identifier, equivalent to a primary key) are: small, likely unique, fast to generate, and ordered. The value consists of 12- bytes, where the first four bytes are for timestamp that reflects the instance when ObjectId creates.
Boolean	Represents logical true or false value.
Date	BSON Date is a 64-bit integer that represents the number of milliseconds since the Unix epoch (Jan 1, 1970).
Null	Represents a null value. A value which is missing or unknown is Null.
Regular Expression	RegExp maps directly to a JavaScript RegExp
32—bit integer	Numbers without decimal points save and return as 32-bit integers.
Timestamp	A special timestamp type for internal MongoDB use and is not associated with the regular date type. Timestamp values are a 64-bit value, where first 32 bits are time, t (seconds since the Unix epoch), and next 32 bits are an incrementing ordinal for operations within a given second.

MongoDB querying commands

Command	Functionality
Mongo	Starts MongoDB; (*mongo is MongoDB client). The default database in MongoDB is test.
db.help ()	Runs help. This displays the list of all the commands.
db.stats ()	Gets statistics about MongoDB server.

Use <database name>	**Creates database**
Db	**Outputs the names of existing database, if created earlier**
Dbs	**Gets list of all the databases**
db.dropDatabase ()	**Drops a database**
db.database name.insert ()	**Creates a collection using insert ()**
db.<database name>.find ()	**Views all documents in a collection**
db.<database name>.update ()	**Updates a document**
db.<database name>.remove ()	**Deletes a document**

Examples:

To Create database Command use - use command creates a database;
use student creates a database named *student*

To see the existence of database Command db - db command shows that *student* database is created

To get list of all the databases Command show dbs - This command shows the names of all databases

To drop database Command db . dropDataba se ( ) - This command drops a database.

To create a **collection** command is db.createCoection('name of the collection')

db.createCollection('CS')

To insert document :

```
db.CS.insert
(
        {
```

```
                    “name”:”Mahesh”,
                    “Roll num”:”R12”
          }
)
```

To add array in collection:

- Insert command can also be used to insert multiple documents into a collection at one time.

```
          db.abc.insert
          (
                [
                      {
                            "ProductCategory": "Airplane’,
                            ”Productld’: 10725,
                            ’ProductName": "Loot Temple"
                      },
                      {
                            ’ProductCategory": ’Airplane",
                            "Pzoduc t I d" : 31047,
                            "Pxoduc tlfame" : "Pxope1 ter Plane"
                      },
                      {
                            ’ProductCategory": ’Airplane’, "Productld’:
                            31049,
                            "DroductName’: ’Twin Spin Helicopter’
                      }
                ]
          )
```

To view all documents in a collection

Command db.<daLabase name> . f ind ( ) is used which is similar to select * in RDBMS or SQL

db.CSE.find()

Cassandra Databases:

Cassandra was developed by Facebook and released by Apache. Cassandra was named after Trojan mythological prophet Cassandra, who had classical allusions to a curse on oracle. Later on, IBM also released the enhancement of Cassandra, as open source version.

- Cassandra is basically a column family database that stores and handles massive data of any format including structured, semi-structured and unstructured data.
- Apache Cassandra DBMS contains a set of programs. They create and *manage* databases.
- Cassandra provides functions (commands) for querying the data and accessing the required information.
- Functions do the viewing, querying and changing (update, insert or append or delete), visualizing and perform transactions on the DB.
- Cassandra is written in Java. Big organizations, such as Facebook, IBM, Twitter, Cisco, Rackspace, eBay, Twitter and Netflix have adopted Cassandra.

Characteristics of Cassandra are :

(i) open source, (ii) scalable (iii) non- relational (v) NoSQL (iv) Distributed (vi) column based, (vii) decentralized, (viii) fault tolerant and (ix) tuneable consistency.

Features of Cassandra are as follows:

- Maximizes the number of writes
- Maximizes data duplication
- Does not support joins, group by, OR clause and aggregations
- Uses Classes consisting of ordered keys and semi-structured data storage systems
- Is fast and easily scalable with write operations spread across the cluster.
- Is a distributed DBMS designed for handling a high volume of structured data across multiple cloud servers
- Has peer-to-peer distribution in the system across its nodes, and the data is distributed among all the nodes in a cluster.

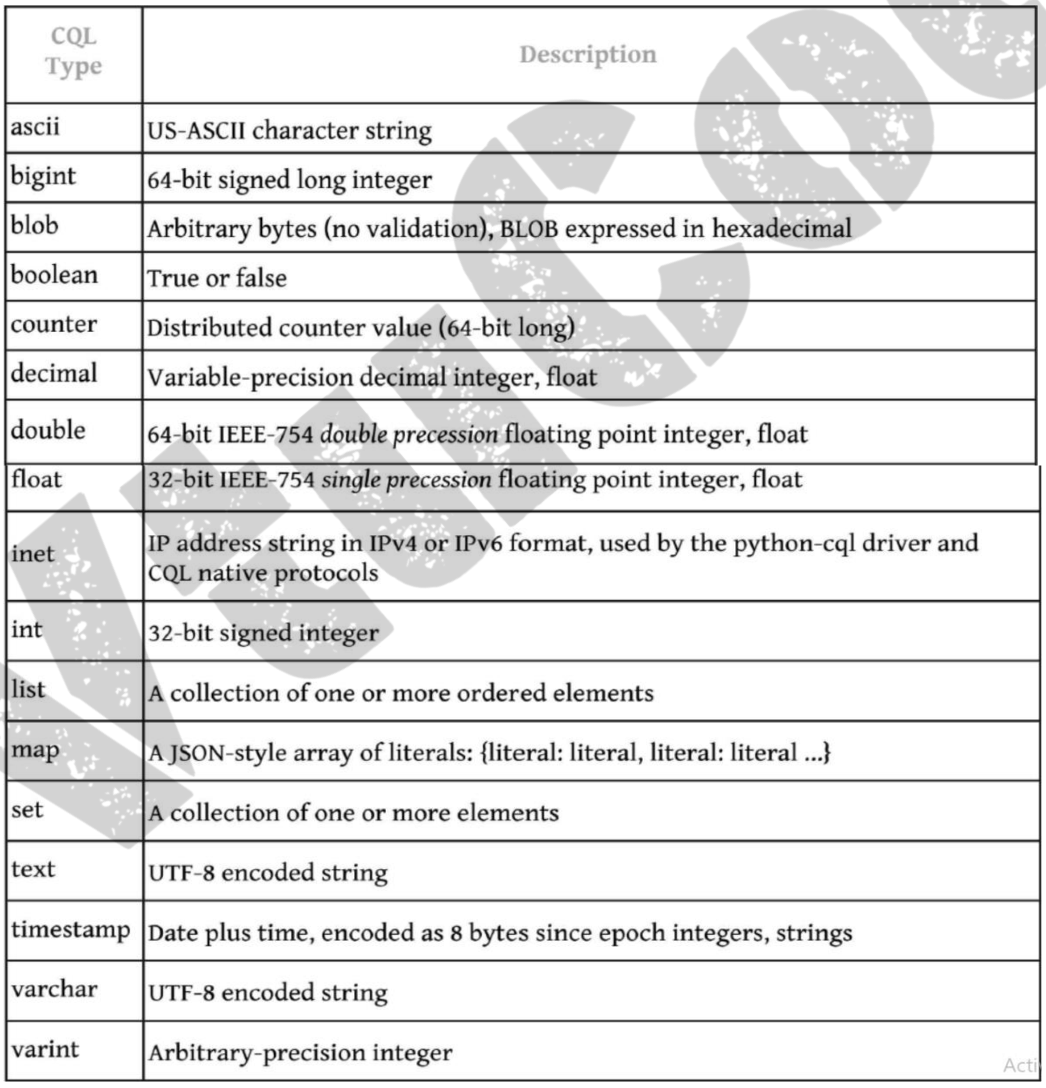
Data Replication : Cassandra stores data on multiple nodes (data replication) and thus has no single point of failure, and ensures availability, a requirement in CAP theorem. Data replication uses a replication strategy. Replication factor determines the total number of replicas placed on different nodes. Cassandra returns the most recent value of the data to the client. If it has detected that some of the nodes responded with a stale value, Cassandra performs a read repair in the background to update the stale values.

Scalability Cassandra provides linear scalability which increases the throughput and decreases the response time on increase in the number of nodes at cluster. Transaction Support Supports ACID properties (Atomicity, Consistency, Isolation, and Durability).

Replication Option Specifies any of the two replica placement strategy names. The strategy names are Simple Strategy or Network Topology Strategy. The replica placement strategies are:

1. Simple Strategy: Specifies simply a replication factor for the cluster.
2. Network Topology Strategy: Allows setting the replication factor for each data center independently.

Data Types in Cassandra:



CQL Type	Description
ascii	US-ASCII character string
bigint	64-bit signed long integer
blob	Arbitrary bytes (no validation), BLOB expressed in hexadecimal
boolean	True or false
counter	Distributed counter value (64-bit long)
decimal	Variable-precision decimal integer, float
double	64-bit IEEE-754 *double precession* floating point integer, float
float	32-bit IEEE-754 *single precession* floating point integer, float
inet	IP address string in IPv4 or IPv6 format, used by the python-cql driver and CQL native protocols
int	32-bit signed integer
list	A collection of one or more ordered elements
map	A JSON-style array of literals: {literal: literal, literal: literal ...}
set	A collection of one or more elements
text	UTF-8 encoded string
timestamp	Date plus time, encoded as 8 bytes since epoch integers, strings
varchar	UTF-8 encoded string
varint	Arbitrary-precision integer

Cassandra Data Model consists of four main components:

(i) Cluster: Made up of multiple nodes and keyspaces,

(ii) Keyspace: a namespace to group multiple column families, especially one per partition,

(iii) Column: consists of a column name, value and timestamp and

(iv) Column- family: multiple columns with row key reference. Cassandra does keyspace management using partitioning of keys into ranges and assigning different key- ranges to specific nodes.

Following Commands prints a description (typically a series of DDL statements) of a schema element or the cluster:

DESCRIBE CLUSTER

DESCRIBE SCHEMA

DESCRIBE KEYSPACES

DESCRIBE KEYSPACE «keyspace name»

DESCRIBE TABLES

DESCRIBE TABLE «table name»

DESCRIBE INDEX **«index name»**

DESCRIBE MATERIALIZED VIEW «view name»

DESCRIBE TYPES

DESCRIBE TYPE «type name»

DESCRIBE FUNCTIONS

DESCRIBE FUNCTION «function name»

DESCRIBE AGGREGATES

DESCRIBE AGGREGATE «aggregate function name»

Consistency Command CONSISTENCY shows **the current** consistency level. CONSISTENCY «LEVEL» sets a new consistency level.

Valid consistency levels are ANY, ONE, TWO, THREE,QUORUM, LOCAL_ONE, LOCAL_QUORUM, EACH_QUORUM, SERIAL AND LOCAL_SERIAL.

Keyspaces:

- A keyspace (or key space) in a NoSQL data store is an object that **contains** all column families of a design as a bundle.
- Keyspace is the **outermost** grouping of the data in the data **store. It is similar to**

relational database. Generally, there is one keyspace per application.

- Keyspace in Cassandra is a namespace that defines data replication on nodes. A cluster contains one keyspace per node.

Create Keyspace Command:

CREATE KEYSPACE <Keyspace Name> WITH replication = {’class’:’<Strategy_name>,replication_factor:Number of replicas} AND durable writes=<‘TRUE/FALSE’>;

Ex: CREATE KEYSPACE test_key WITH replication = {’class’:’simple strategy’,replication_factor:’1’} AND durable writes=‘TRUE’;

ALTER KEYSPACE command changes (alter) **properties,** such as the number of replicas and the durable_writes of a keyspace.

DESCRIBE KEYSPACE command displays the existing keyspaces.

DROP KEYSPACE command drops a keyspace

Use KEYSPACE command connects the client session with a keyspace.

Commandd	**Functionaity**
CQLSH	**A command line shell for interacting with Cassandra through CQL**
HELP	**Runs help. This displays the list of all the commands**
CONSISTENCY	**Shows the current consistency level**
EXIT	**Terminate the CQL shell**
SHOW HOST	**Displays the host**
SHOW VERSION	**Displays the details of current cqlsh session such as host, Cassandra version, or data type assumptions**
CREATE KEYSPACE <Keyspace Name>	**Creates keyspace with a name**
DESCRIBE KEYSPACE <Keyspace Name>	**Displays the keyspace with a name**
ALTER KEYSPACE <Keyspace Name>	**Modifies keyspace with a name**
DROP KEYSPACE <Keyspace Name>	**Deletes keyspace with a name**
CREATE (TABLE	**Creates a table or column family**

' **COLUMNFAMILY)**	
COLLECTIONS	**Lists the Collections**

Create table:

CREATE (TABLE | COLUMNFAMILY) <tablename> ('<column-definition>' , '<column-definition>') (WITH <option> AND <option>)

Note: Primary key is a column used to uniquely identify a row. Therefore, defining a primary key is compulsory.

Ex: Create table ProductInfo(ProductId int primary key, ProductType text);

Describe Tables Command:

DESCRIBE TABLE Command displays all the tables in the current keyspace:

DESCRIBE TABLE <TABLE NAME>;

Ex: DESCRIBE TABLE Productlnfo;

Alter Tables Command:

ALTER TABLE Command ALTER (TABLE|COLUMNFAMILY)<tablename> (ADD | DROP) <column name>

Add a column dateofManufacturing in the table Productfn/o:

ALTER TABLE ProductInfo add dateOfManufacturing timestamp;

Cassandra CURD Operations: (CURD—Create, Update, Read and Delete data into tables) :

Insert Command:

INSERT command creates data in a table:

INSERT INTO <tablename>(<columnl name>, <column2 name>....)

VALUES(<valuel>,<value2>....) USING <option>

Update Command:

UPDATE command updates data in a table. The following keywords are used while updating data in a table:

Where — This clause is used to select the row to be updated. Set —

Set the value using this keyword.

Must - Includes all the columns composing the primary key. If a

given row is unavailable, then UPDATE creates a new row.

UPDATE <tablename> SET <column name> = <new value> <column name> = <value>.... WHERE<condition>

Select Command

SELECT command reads the data from a table. The command can read a whole table, a single column, or a particular cell:

SELECT <column name(s)> FROM <Table Name>

To select all records:

SELECT * FROM <Table Name>

To select records that fulfils required condition:

SELECT <columnl,column2,..> FROM <Table Name> where <Condition>

Ex:

SELECT Product Type, Product Id, Product Name,and Product Cost from ProductInfo where ProductId = 31047;

Delete Command

DELETE command deletes data from a table:

DELETE FROM <identifier> WHERE <condition>;

Ex: Delete row from a table where Product id is 31047: DELETEFROMProductInfo WHERE ProductId =31047;

Cassandra Client A relational database client connects to DB server using drivers. Java JDBC driver API enables storing and retrieving data. Cassandra has peer-to-peer distribution architecture. Several instances require the clients. The driver enables the use of different languages for connecting to DBs. Cassandra does not include the drivers.

Cassandra Hadoop Support Cassandra 2.1 has Hadoop 2 support. The **setup** and configuration overlays a Hadoop cluster on the Cassandra nodes. A server is configured for the NameNode and JobTracker. Each Cassandra node then installs the TaskTracker and Data Node.