

Course Contents

Unit-08:NoSQL

- Structured and Unstructured Data;
- Introduction to NoSQL Databases (NoSQL Database, Types of NoSQL Database, Advantage of NoSQL);
- Discussion of basic architecture of Hbase, Cassandra and MongoDb;

Course Contents

Unit-08:NoSQL

Questions:

- What is NoSQL? Explain the characteristics of NoSQL.
- What are the types of NoSQL?
- Explain HBase architecture.
- Explain Cassandra architecture.
- Explain MongoDb architecture.

Course Contents

Structured and Unstructured Data:

Structured and unstructured data are two fundamental types of data that refer to the way information is organized and stored. These terms are commonly used in the context of data management and analysis.

Structured Data - Structured data refers to data that is organized into a specific format with well-defined rows and columns. This data type is highly organized and follows a predefined schema. Examples of structured data include data stored in relational databases, spreadsheets, and tables. Structured data is easy to search, query, and analyze because of its organized nature.

Characteristics of structured data:

- > Organized in tables or grids with rows and columns.
- > Follows a consistent schema or data model.
- > Well-suited for databases and traditional data storage systems.
- > Easily accessible, searchable, and analyzable using SQL or similar query languages.

Examples of structured data - Employee records with columns for name, age, job title, salary, etc. Sales transactions with columns for date, product, quantity, and price. Inventory data with columns for item number, description, quantity on hand, etc.

Course Contents

Structured and Unstructured Data:

Unstructured Data - Unstructured data refers to data that doesn't have a predefined structure or format. It is often more complex and can't be easily organized into rows and columns like structured data. Unstructured data includes various types of content, such as text, images, audio files, videos, social media posts, and more. Analyzing unstructured data can be challenging due to its lack of predefined organization.

Characteristics of unstructured data:

- > Lacks a predefined structure or schema.
- > Includes various formats like text, images, audio, and video.
- > More difficult to process and analyze compared to structured data.
- > Requires specialized tools and techniques for extraction and analysis.

Course Contents

Introduction to NoSQL Databases:

- NoSQL Database is a 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 huge data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.
- NoSQL database stands for “Not Only SQL” or “Not SQL.” Though a better term would be “NoREL”, NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.
- Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

Course Contents

Introduction to NoSQL Databases:

- Polymorphic data means that in one collection you have many versions of document schema. For example in below documents email field is string or array of string:

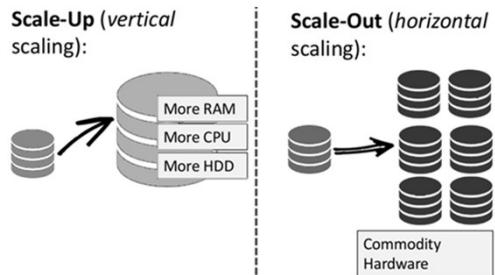
```
{  
    "user": "Anna",  
    "email" : "anna@gmail.com"  
}  
  
{  
    "user": "Jon",  
    "email" : [  
        "jon@gmail.com",  
        "jon@yahoo.com"  
    ]  
}
```

Course Contents

Why NoSQL?

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data. To resolve this problem, we could “scale up” our systems by upgrading our existing hardware. This process is expensive.

The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as “scaling out.”



Course Contents

Characteristics of NOSQL Systems: The characteristics of many NOSQL systems, and how these systems differ from traditional SQL systems. The characteristics of NOSQL Systems are divided into two categories.

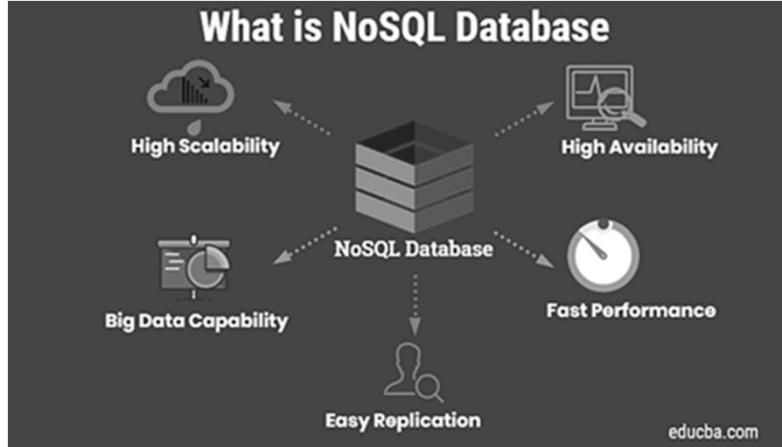
- related to distributed databases and distributed systems, and
 - related to data models and query languages.
1. NOSQL characteristics related to distributed databases and distributed systems.

NOSQL systems emphasize high availability, so replicating the data is inherent in many of these systems. Scalability is another important characteristic, because many of the applications that use NOSQL systems tend to have data that keeps growing in volume. High performance is another required characteristic, whereas serializable consistency may not be as important for some of the NOSQL applications.

Course Contents

Characteristics of NOSQL Systems:

1. NOSQL characteristics related to distributed databases and distributed systems.



Course Contents

Characteristics of NOSQL Systems:

- a) **Scalability** - In distributed systems, scalability comes in two forms: horizontal and vertical. NoSQL systems typically favor horizontal scalability, achieved by adding more nodes as data volume increases. In contrast, vertical scalability involves enhancing individual nodes' storage and computing capacity. NoSQL systems prioritize operational horizontal scalability, requiring methods to smoothly distribute data across new nodes without disrupting the system.
- b) **Availability, Replication and Eventual Consistency** - In NOSQL systems, continuous availability is essential. Data replication across nodes ensures data is accessible even if one fails. Replication boosts availability and read speed, but updating all copies can slow writes. For most NOSQL apps, eventual consistency suffices, not strict serializability.

Course Contents

Characteristics of NOSQL Systems:

- c) **Replication Models:** Two major replication models are used in NOSQL systems 1) Master-slave and 2) master-master replication.
 - Master-slave replication - requires one copy to be the master copy; all write operations must be applied to the master copy and then propagated to the slave copies, usually using **eventual consistency** (the slave copies will eventually be the same as the master copy). For read, the master-slave paradigm can be configured in various ways. One configuration requires all reads to also be at the master copy, so this would be similar to the primary site or primary copy methods of distributed concurrency control, with similar advantages and disadvantages. Another configuration would allow reads at the slave copies but would not guarantee that the values are the latest writes, since writes to the slave nodes can be done after they are applied to the master copy.

Course Contents

Characteristics of NOSQL Systems:

- Master-Master replication - allows reads and writes at any of the replicas but may not guarantee that reads at nodes that store different copies see the same values. Different users may write the same data item concurrently at different nodes of the system, so the values of the item will be temporarily inconsistent. A reconciliation method to resolve conflicting write operations of the same data item at different nodes must be implemented as part of the master-master replication scheme.
- d) **Sharding of Files** - In NOSQL applications, files can hold millions of records accessed by many users simultaneously. Storing entire files on one node isn't feasible. Sharding, or horizontal partitioning, splits records across nodes to balance load. Combined with shard replication, this enhances load balancing and data availability.

Course Contents

Characteristics of NOSQL Systems:

- e) **High-Performance Data Access** - In NOSQL applications, finding specific data items among large volumes of records is crucial. Two common methods are used: hashing and range partitioning based on object keys. Most accesses involve providing the key value, similar to an object ID. Hashing applies a function to the key, determining the object's location. Range partitioning uses key ranges, like $K_{\min} \leq K \leq K_{\max}$, suitable for range queries. Additional indexes can locate objects based on different attributes than the key.

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)
PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN (21)
);
```

Course Contents

Characteristics of NOSQL Systems:

2. NOSQL characteristics related to data models and query languages.

NOSQL systems emphasize performance and flexibility over modeling power and complex querying.

- a) **Not Requiring a Schema** - The flexibility of not requiring a schema is achieved in many NOSQL systems by allowing semi-structured, self-describing data. The users can specify a partial schema in some systems to improve storage efficiency, but it is not required to have a schema in most of the NOSQL systems. As there may not be a schema to specify constraints, any constraints on the data would have to be programmed in the application programs that access the data items. There are various languages for describing semi structured data, such as JSON (JavaScript Object Notation) and XML (Extensible Markup Language). JSON is used in several NOSQL systems, but other methods for describing semi-structured data can also be used.

Course Contents

Characteristics of NOSQL Systems:

- b) **Less Powerful Query Languages:** - NOSQL systems often lack complex querying like SQL. They're mainly used for simple single-object reads based on keys. Programming APIs handle data access through CRUD (and sometimes SCRUD) operations. Limited query languages might be available, but not as powerful as SQL, lacking features like joins. Join operations are usually implemented in application code.
- c) **Versioning** - Some NOSQL systems provide storage of multiple versions of the data items, with the timestamps of when the data version was created.

Course Contents

What is the CAP Theorem?

CAP theorem is also called brewer's theorem. It states that it is impossible for a distributed data store to offer more than two out of three guarantees

1. Consistency
2. Availability
3. Partition Tolerance

Consistency - The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

Availability - The database should always be available and responsive. It should not have any downtime.

Partition Tolerance - Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

Eventual Consistency - The term “eventual consistency” means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item

Course Contents

What is the CAP Theorem?

CAP theorem is also called brewer's theorem. It states that it is impossible for a distributed data store to offer more than two out of three guarantees

1. Consistency
2. Availability
3. Partition Tolerance

Consistency - The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

Availability - The database should always be available and responsive. It should not have any downtime.

Partition Tolerance - Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

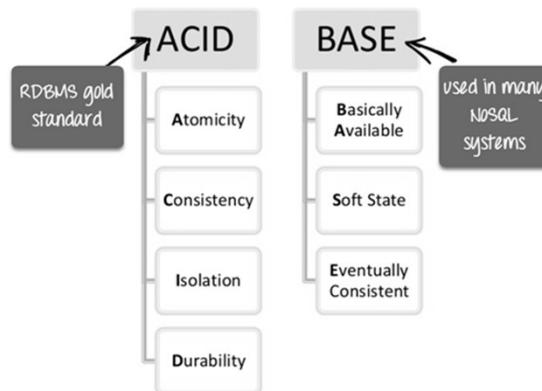
Eventual Consistency - The term “eventual consistency” means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item on one machine has to be propagated to other replicas.

Course Contents

What is the CAP Theorem?

BASE: Basically Available, Soft state, Eventual consistency

- Basically available means DB is available all the time as per CAP theorem
- Soft state means even without an input; the system state may change
- Eventual consistency means that the system will become consistent over time



Course Contents

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don't need a dedicated high-performance server
- Support Key Developer Languages and Platforms
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Course Contents

Disadvantages of NoSQL

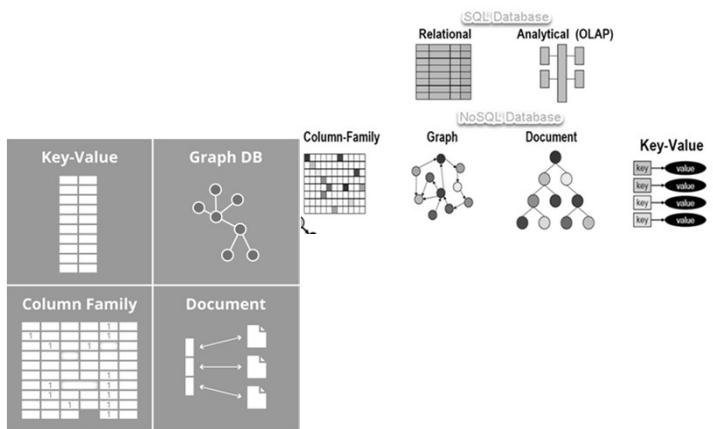
- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn't work as well with relational data
- The learning curve is stiff for new developers
- Open-source options so not so popular for enterprises.

Course Contents

Types of NoSQL Databases:

NoSQL Databases are mainly categorized into four types: Key-value pair, Column-oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs.

1. Key-value pair,
2. Column-oriented,
3. Graph-based
4. Document-oriented.



Course Contents

Types of NoSQL Databases:

1. **Document-based NOSQL systems:** These systems store data in the form of documents using well-known formats, such as JSON (JavaScript Object Notation). Documents are accessible via their document id, but can also be accessed rapidly using other indexes.
2. **NOSQL key-value stores:** These systems have a simple data model based on fast access by the key to the value associated with the key; the value can be a record or an object or a document or even have a more complex data structure.
3. **Column-based or wide column NOSQL systems:** These systems partition a table by column into column families (a form of vertical partitioning), where each column family is stored in its own files. They also allow versioning of data values.
4. **Graph-based NOSQL systems:** Data is represented as graphs, and related nodes can be found by traversing the edges using path expressions.

Course Contents

Column-based or wide column NOSQL systems:

Column Oriented Storage

Traditional Database

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213



row	column	value
1	type	Friend request status
1	for user	Ryan
1	from user	Jessica
1	timestamp	146710201
2	type	Comment
2	for user	Chaz
2	from user	Daniel
2	timestamp	146711200
3	type	Comment
3	for user	Rick
3	from user	Brendan
3	timestamp	1467112205

Course Contents

Column-based or wide column NOSQL systems:

Sales			
Product	Customer	Date	Sale
Beer	Thomas	2011-11-25	2 GBP
Beer	Thomas	2011-11-25	2 GBP
Vodka	Thomas	2011-11-25	10 GBP
Whiskey	Christian	2011-11-25	5 GBP
Whiskey	Christian	2011-11-25	5 GBP
Vodka	Alexei	2011-11-25	10 GBP
Vodka	Alexei	2011-11-25	10 GBP

Product	
ID	Value
1	Beer
2	Beer
3	Vodka
4	Whiskey
5	Whiskey
6	Vodka
7	Vodka

Customer	
ID	Customer
1	Thomas
2	Thomas
3	Thomas
4	Christian
5	Christian
6	Alexei
7	Alexei

- Table with single-row partitions

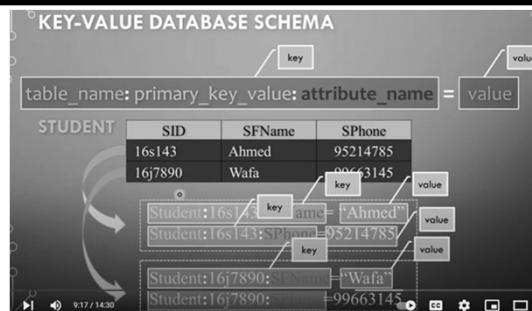


- Column family view



Course Contents

NOSQL key-value stores:



* Convert the below EMPLOYEE relation into Key-Value database schema.

EmpID	EmpName	EmpAddress	EmpBdate
100	Laika Al-Mamari	Al Batinah, Sohar	1/21/1980
101	Khalid Al-Ameri	Al Aqr, Shinas	2/3/1990
102	Ranya Al-Balushi	Al Mutaqa, Sohar	5/25/1985

SOLUTION:

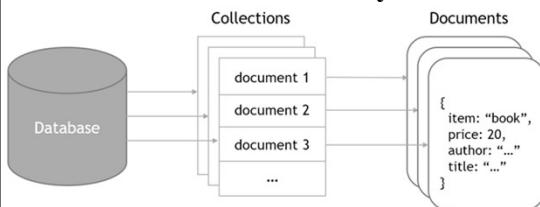
Employee:100:EmpName = "Laika Al-Mamari"
 Employee:100:EmpAddress = "Al Batinah, Sohar"
 Employee:100:EmpBdate = 1/21/1980

Employee:101:EmpName = "Khalid Al-Ameri"
 Employee:101:EmpAddress = "Al Aqr, Shinas"
 Employee:101:EmpBdate = 2/3/1990

Employee:102:EmpName = "Ranya Al-Balushi"
 Employee:102:EmpAddress = "Al Mutaqa, Sohar"
 Employee:102:EmpBdate = 5/25/1985

Course Contents

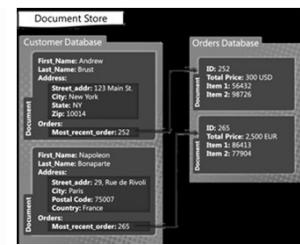
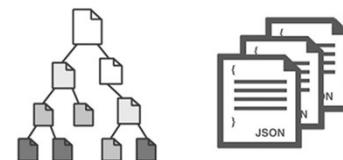
Document-based NOSQL systems:



Look familiar?

Document-based NoSQL stores are a superset of Key-Value NoSQL stores, often adding greatly enhanced query and indexing capabilities.

NoSQL Document Databases



Microsoft

Course Contents

Basic architecture of HBase, Cassandra and MongoDb:

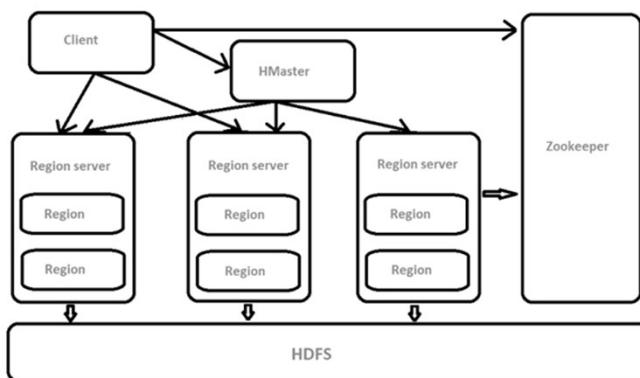
- Basic architecture of Hbase
- Basic architecture of MongoDB
- Basic architecture of Cassandra

Course Contents

HBase Architecture: HDFS(Hadoop Distributed File System):

HBase architecture has 3 main components:

1. HMaster,
2. Region Server
3. Zookeeper



Course Contents

HBase Architecture: HDFS(Hadoop Distributed File System):

All the 3 components are described below:

HMaster: The implementation of Master Server in HBase is HMaster. It is a process in which regions are assigned to region server as well as DDL (create, delete table) operations. It monitors all Region Server instances present in the cluster. In a distributed environment, Master runs several background threads. HMaster has many features like controlling load balancing, failover etc.

Region Server: HBase Tables are divided horizontally by row key range into Regions. Regions are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families. Region Server runs on HDFS DataNode which is present in Hadoop cluster. Regions of Region Server are responsible for several things, like handling, managing, executing as well as reads and writes HBase operations on that set of regions. The default size of a region is 256 MB.

Zookeeper: It is like a coordinator in HBase. It provides services like maintaining configuration information, naming, providing distributed synchronization, server failure notification etc. Clients communicate with region servers via zookeeper.

Note: HBase is extensively used for online analytical operations, like in banking applications such as real-time data updates in ATM machines, HBase can be used.

Course Contents

HBase Architecture: HDFS(Hadoop Distributed File System):

Advantages of HBase:

1. Can store large data sets
2. Database can be shared
3. Cost-effective from gigabytes to petabytes
4. High availability through failover and replication

Disadvantages of HBase:

1. No support SQL structure
2. No transaction supports
3. Sorted only on key
4. Memory issues on the cluster

Course Contents

MongoDB Architecture:

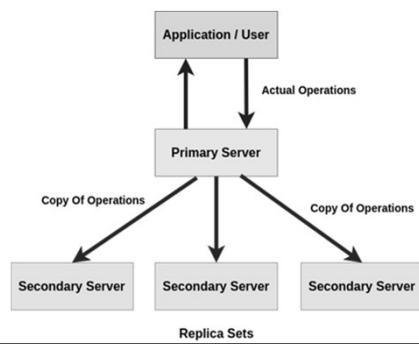
- Both MongoDB and HBase are Distributed NoSQL databases, used extensively to handle large data problems.
- MongoDB is a document-based NoSQL database. It requires no fixed schema definition. Mongo DB stores data as Binary JSON or BSON. It supports horizontal scaling.
- Several servers instance group together as a cluster to support Mongo DB as a distributed system.
- MongoDB uses MongoDB query language and supports Adhoc queries, Replication, and Sharding.
- Sharding is a feature of MongoDB that helps it to operate as a distributed data system.

Course Contents

MongoDB Architecture: Replication in Mongo

Replication serves as a very important feature in order to protect data loss from one server, increase the availability of data, and give protection from failures. MongoDB achieves replication using the concept replica sets. A replica set is a group of mongoDB instances that host the same data set. One of the nodes is selected as the primary or main node. This is called the primary node.

All others are called secondary nodes. The primary node receives all the operations from the user and the secondaries are updated from the primary one by using the same operation to maintain consistency. If the primary node goes down, one of the secondary nodes is selected as the primary node and the operations are carried forward. When the fallen node recovers, it joins the cluster as the secondary nodes. We can control our cluster of mongo instances using Mongo Atlas. Mongo clusters are created based on the idea of horizontal scaling and adding of instances.



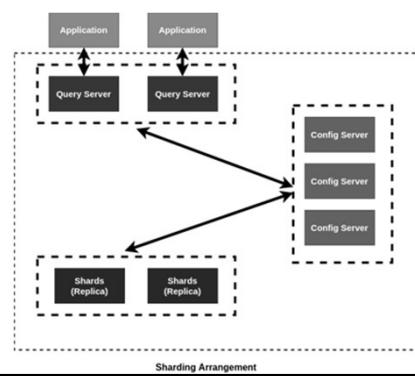
Course Contents

MongoDB Architecture: Sharding in Mongo DB

Sharding is used by MongoDB to store data across multiple machines. It uses horizontal scaling to add more machines to distribute data and operation with respect to the growth of load and demand. Sharding arrangement in MongoDB has mainly three components:

Shards or replica sets: Each shard serves as a separate replica set. They store all the data. They target to increase the consistency and availability of the data.

Configuration Servers: They are like the managers of the clusters. These servers contain the cluster's metadata. They actually have the mapping of the cluster's data to the shards. When a query comes, query routers use these mappings from the config servers to target the required shard.



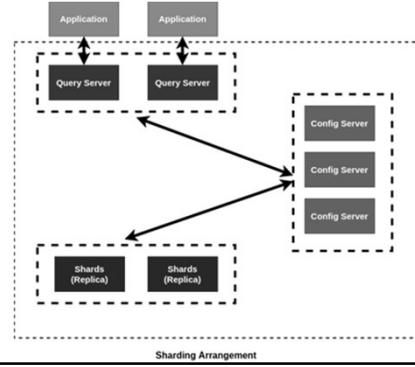
Course Contents

MongoDB Architecture: Sharding in Mongo DB

Query Router: The query router is mongo instances which serve as interfaces for user applications. They take in the user queries from the applications and serve the applications with the required results. Usually, there are multiple query router per cluster for load distribution.

The image shows the sharding arrangement for MongoDB. Though the image has only 2 query servers and shards, generally there are more in an actual cluster, though by default case there are 3 config servers in a cluster.

The ideas we talked about above are the two most important ideas behind MongoDB operations and its distributed data system architecture.

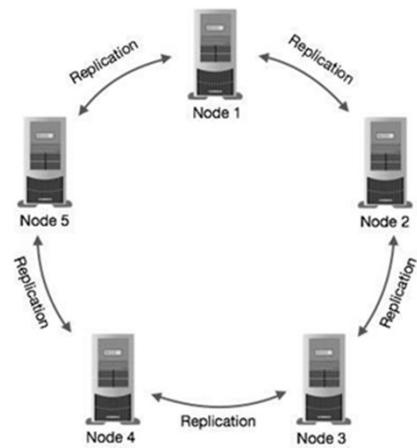


Course Contents

Cassandra Architecture:

Cassandra was designed to handle big data workloads across multiple nodes without a single point of failure. It has a peer-to-peer distributed system across its nodes, and data is distributed among all the nodes in a cluster.

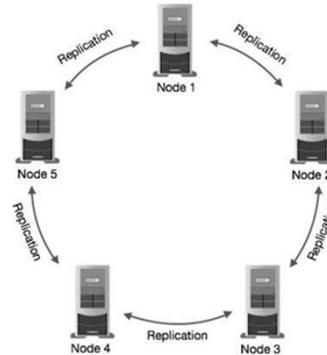
- In Cassandra, each node is independent and at the same time interconnected to other nodes. All the nodes in a cluster play the same role.
- Every node in a cluster can accept read and write requests, regardless of where the data is actually located in the cluster.
- In the case of failure of one node, Read/Write requests can be served from other nodes in the network.



Course Contents

Cassandra Architecture: Data Replication in Cassandra

In Cassandra, nodes in a cluster act as replicas for a given piece of data. If some of the nodes are responded with an out-of-date value, Cassandra will return the most recent value to the client. After returning the most recent value, Cassandra performs a read repair in the background to update the stale values. See the following image to understand the schematic view of how Cassandra uses data replication among the nodes in a cluster to ensure no single point of failure.



Course Contents

Cassandra Architecture: Components of Cassandra

The main components of Cassandra are:

- **Node:** A Cassandra node is a place where data is stored.
- **Data center:** Data center is a collection of related nodes.
- **Cluster:** A cluster is a component which contains one or more data centers.
- **Commit log:** In Cassandra, the commit log is a crash-recovery mechanism. Every write operation is written to the commit log.
- **Mem-table:** A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.
- **SSTable:** It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.
- **Bloom filter:** These are nothing but quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.