SNS COLLEGE OF TECHNOLOGY
(An Autonomous institution)
COIMBATORE-641 035

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Unit 3

INTRODUCTION TO MONGODB AND CASSANDRA

Why Mongo DB

MongoDB makes it easy for developers to store structured or unstructured data. MongoDB is a document database built on a horizontal scale-out architecture that uses a flexible schema for storing data. Instead of storing data in tables of rows or columns like SQL databases, each record in a MongoDB database is a document described in BSON, a binary representation of the data. Applications can then retrieve this information in a JSON format.

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document. Database Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases. Collection Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose. Document A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or

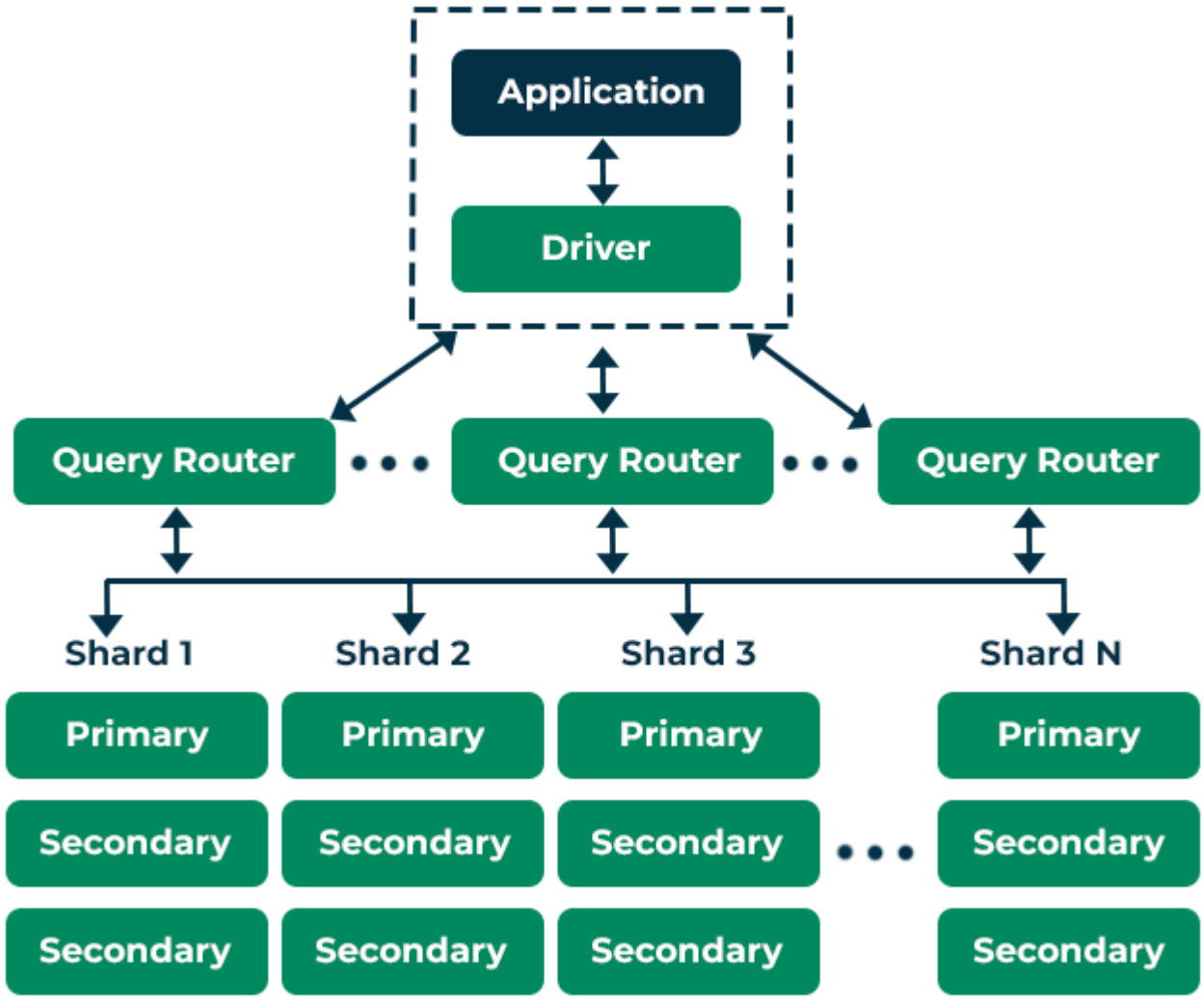
structure, and common fields in a collection's documents may hold different types of data.

Key Features of MongoDB

- Document-oriented Database
- Stores data in BSON-like documents.
- Schema Less database.
- It provides **horizontal scalability** with the help of sharding.
- It provides high availability and **redundancy** with the help of **replication**.
- It allows one to perform operations on the grouped data and get a single result or computed result.
- It has very high performance.

MongoDB Architecture and its Components

MongoDB's architecture design involves several important parts that work together to create a strong and flexible database system. these are the following MongoDB's architecture



1. Drivers & Storage Engine

MongoDB store the data on the server but that data we will try to retrieve from our application. So that time how the communication is happening between our application and MongoDB server.

Any application which is written in python, .net and java or any kind of frontend application, these application are trying to access the data from these physical storage in server. First they will interact with driver which will communicate with **MongoDB** server. What happen is once the request is going from the **frontend application** through the driver then driver will change appropriate query by using query

engine and then the query will get executed in MongoDB data model. Left side is security which provides security to the database that who will access the data and right side is management this management will manage all these things.

Drivers

Drivers are client libraries that offer **interfaces** and **methods** for applications to communicate with MongoDB databases. Drivers will handle the translation of documents between **BSON objects** and mapping application structures.

.NET, Java, JavaScript, Node.js, Python, etc are some of the widely used drives supported by MongoDB.

Storage Engine

The storage engine significantly influences the performance of applications, serving as an intermediary between the MongoDB database and persistent storage, typically disks. MongoDB supports different storage engines:

- **MMAPv1** – It is a traditional **storage engine** based on **memory mapped files**. This storage engine is optimized for workloads with high volumes of read operations, insertions, and in-place updates. It uses **B-tress** to store indexes. Storage Engine works on multiple reader single writer lock. A user cannot have two write calls to be processes in parallel on the same collection. It is fast for reads and slow for writes.
- **Wired Tiger** – Default Storage Engine starts from MongoDB 3version. No locking Algorithms like **hash pointer**. It yields 7x-10x better write operations and 80% of the file system compression than MMAP.
- **InMemory** – Instead of storing documents on disk, the engine uses in-memory for more **predictable data latencies**. It uses 50% of physical **RAM** minimum 1 GB as default. It requires all its data. When dealing with large datasets, the in-memory engine may not be the most suitable choice.

2. Security

- Authentication
- Authorization
- Encryption on data
- Hardening (Ensure only trusted hosts have access)

3. MongoDB Server

It serves as the **central element** and is in charge of **maintaining**, **storing**, and **retrieving data** from the database through a number of interfaces. The system's heart is the MongoDB server. Each mongod server instance is in charge of handling client requests, maintaining data storage, and performing database operations. Several mongod instances work together to form a cluster in a typical MongoDB setup.

4. MongoDB Shell

For dealing with MongoDB databases, MongoDB provides the MongoDB Shell **command-line interface** (CLI) tool. The ability to handle and query MongoDB data straight from the terminal is **robust and flexible**. After installing MongoDB, you may access the MongoDB Shell, often known as mongo. It interacts with the database using JavaScript-based syntax. Additionally, it has built-in help that shows details about possible commands and how to use them.

5. Data Storage in MongoDB

5.1 Collections

A database can contain as many collections as it wishes, and MongoDB stores **data inside collections**.

As an example, a database might contain three collections **a user's collection**, **a blog post collection**, and **a comments collection**. The user collection would hold user data and documents, the blog post collection would hold blog posts and documents, and the comments collection would hold documents related to comments. This would allow for the easy retrieval of all the documents from a single collection.

5.2 Documents

Documents themselves represent the individual records in a specific collection.

For example inside the blog posts collection we'd store a lot of blog post documents and each one represents a single blog post now the way that data is structured inside a document looks very much like a **JSON** object with key value pairs but actually it's being stored as something called BSON which is just binary JSON.

6. Indexes

Indexes are data structures that make it simple to navigate across the collection's data set. They help to execute queries and find documents that match the query criteria without a collection scan.

These are the following different types of indexes in MongoDB:

6.1 Single field

MongoDB can traverse the indexes either in the **ascending** or **descending order** for single-field index

```
db.students.createIndex({"item":1})
```

In this example, we are creating a single index on the item field and 1 here represents the filed is in ascending order.

A **compound index** in MongoDB contains multiple single filed indexes separated by a comma. MongoDB restricts the number of fields in a compound index to a maximum of 31.

```
db.students.createIndex({"item": 1, "stock":1})
```

Here, we create a compound index on item: 1, stock:1

6.2 Multi-Key

When indexing a filed containing an **array value**, MongoDB creates **separate index** entries for each array component. MongoDB allows you to create multi-key indexes for arrays containing scalar values, including strings, numbers, and nested documents.

```
db.students.createIndex({<filed>: <1 or -1>})
```

6.3 Geo Spatial

Two geospatial indexes offered by MongoDB are called **2d indexes** and **2d sphere indexes**. These indexes allow us to query

geospatial data. On this case, queries intended to locate data stored on a two-dimensional plane are supported by the 2d indexes. On the other hand, queries that are used to locate data stored in spherical geometry are supported by 2D sphere indexes.

6.4 Hashed

To maintain the entries with **hashes** of the values of the indexed field we use Hash Index. MongoDB supports hash based sharding and provides hashed indexes.

```
db.<collection>.createIndex( { item: “hashed” } )
```

7. Replication

Within a MongoDB cluster, data replication entails keeping several copies of the same data on various servers or nodes. Enhancing **data availability** and **dependability** is the main objective of data replication. A replica may seamlessly replace a failing server in the cluster to maintain service continuity and data integrity.

- **Primary Node (Primary Replica):** In a replica set, the primary node serves as the main source for all write operations. It’s the only node that accepts write requests. The main node is where all data modifications begin and are implemented initially.
- **Secondary Nodes:** Secondary nodes duplicate data from the primary node (also known as secondary replicas). They are useful for dispersing read workloads and load balancing since they are read-only and mostly utilized for read activities.

8. Sharding

Sharding is basically **horizontal scaling** of databases as compared to the traditional vertical scaling of **adding more CPUS** and ram to the current system.

For example, you have huge set of files you might segregate it into smaller sets for ease. Similarly what mongo database does is it **segregates** its data into smaller chunks to improve the efficiency.

you have a machine with these configuration and mongo db instance running on it **storing 100 million documents**.

Now with time your data will grow in your mongo db instance and suppose 100 million extra documents get added. Now to manage the processing of these extra records you might need to add extra ram, extra storage and extra CPU to the server. Such type of scaling is called vertical scaling.

Now consider another situation if you have 4 small machines with small configurations. You can divide 200 million of document into each of the server such that each of the server might hold around 50 million documents. By dividing the data into multiple servers you have reduced the computation requirements and such kind of scaling is known as horizontal scaling and this horizontal scaling is known as sharding in mongo and each of the servers S1, S2, S3, S4 are the shards.

The partioning of data in a sharded environment is done on a range based basis by deciding a field as a shard key.

Conclusion

The architecture of MongoDB has been thoroughly examined in this extensive article, including its essential parts, data storage, replication, sharding, high availability, security, scalability, and performance optimization. MongoDB is a top choice for a wide range of applications, from small-scale initiatives to massive, data-intensive systems, due to its adaptable and potent design. To fully utilize MongoDB and create reliable, scalable, and secure solutions, developers and administrators must have a thorough understanding of the database's architecture.

•

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
It is a relational database.	It is a non-relational and document-oriented database.
Not suitable for hierarchical data storage.	Suitable for hierarchical data storage.
It is vertically scalable i.e increasing RAM.	It is horizontally scalable i.e we can add more servers.
It has a predefined schema.	It has a dynamic schema.
It is quite vulnerable to SQL injection.	It is not affected by SQL injection.
It centers around ACID properties (Atomicity, Consistency, Isolation, and Durability).	It centers around the CAP theorem (Consistency, Availability, and Partition tolerance).
It is row-based.	It is document-based.
It is slower in comparison with MongoDB.	It is almost 100 times faster than RDBMS.
Supports complex joins.	No support for complex joins.
It is column-based.	It is field-based.
It does not provide JavaScript client for querying.	It provides a JavaScript client for querying.
It supports SQL query language only.	It supports JSON query language along with SQL.