day 2-mongodb

introduction:

MongoDB is the most popular NoSQL open source document-oriented database. The term 'NoSQL' means 'non-relational'. This means that MongoDB is not based on a table like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON ( similar to JSON format).

SQL databases store data in tabular format. These data are stored in a predefined data model which is not flexible for highly growing applications in today's real world.

Modern applications are more social, interactive and networked than ever. Applications stores more and more data and accessing them at higher rates.

Relational Database Management System is not the correct choice when it comes to handle big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit.

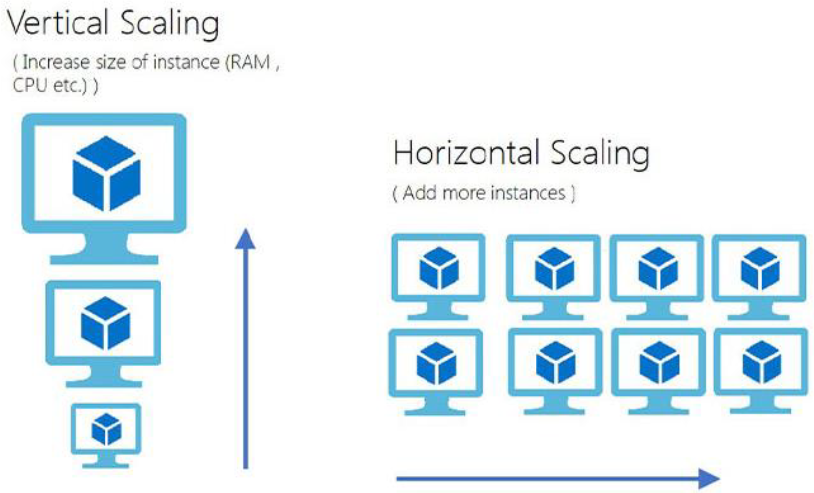
NoSQL databases are more scalable and provide superior performance. MongoDB scales by adding more and more servers and increases productivity with its flexible document model.

**HORIZONTAL SCALING:**

Horizontal scaling, also referred to as "scaling out," is the process of improving system performance and capacity by adding more machines or nodes to a system rather than upgrading existing hardware. This approach is commonly used in modern distributed systems and cloud environments where demand fluctuates and high availability is critical. In horizontal scaling, the workload is distributed across multiple servers, which helps prevent any single point of failure and allows for zero-downtime maintenance and deployment. It offers virtually unlimited scalability since additional resources can be added as needed without hitting hardware limits. For example, instead of upgrading a single database server with more RAM and CPU, an organization might deploy multiple database instances and use sharding or replication to distribute the data. Although it introduces some complexity in terms of system architecture and data synchronization, horizontal scaling is a highly effective strategy for applications requiring fast, resilient, and large-scale data processing—such as social media platforms, e-commerce websites, or real-time analytics engines.

**VERTICAL SCALING:**

Vertical scaling, also known as "scaling up," involves increasing the capacity of a single server by adding more powerful hardware resources such as CPU, RAM, or storage. This method enhances the performance of the existing system without changing the application architecture, making it simpler to implement compared to horizontal scaling. It is particularly useful for legacy applications or databases that are not designed to run on multiple nodes. However, vertical scaling has physical limitations—there’s only so much hardware you can add to a single machine. It can also become costly, as high-performance components tend to be expensive, and any upgrade may require system downtime. Additionally, if the scaled-up machine fails, the entire application could become unavailable due to the lack of redundancy. Despite these drawbacks, vertical scaling is often the initial choice for small to medium workloads or when an application demands strong consistency and cannot easily be distributed across servers.



**Why Choose MongoDB Over SQL?**

MongoDB is a modern, document-oriented NoSQL database designed for flexibility, scalability, and performance. Unlike traditional SQL databases, which store data in rigid tables with predefined schemas, MongoDB uses JSON-like documents that can store nested data and vary in structure. This makes it ideal for applications with rapidly changing requirements, such as agile development, real-time analytics, IoT platforms, and content management systems. MongoDB also supports horizontal scaling through sharding, allowing massive datasets to be distributed across multiple servers for improved performance and availability. Its flexible data model reduces the need for complex joins and schema migrations, which are often time-consuming in SQL databases. While SQL databases like MySQL or PostgreSQL are better suited for complex transactional systems requiring strong consistency, MongoDB shines when handling large volumes of semi-structured or unstructured data with high throughput and low latency. As a result, developers often choose MongoDB for its ease of development, scalability, and adaptability to evolving application needs.

**Advantages of MongoDB**

* MongoDB offers numerous advantages that make it a popular choice for modern application development.
* One of its key benefits is **flexibility**—it uses a schema-less document model, allowing developers to store varying data structures without strict schema definitions, making it ideal for agile development.
* MongoDB supports **horizontal scaling** through sharding, enabling it to handle large volumes of data across multiple machines efficiently. It also provides **high availability** with built-in replication and automatic failover.
* The **performance** of MongoDB is optimized for read and write operations, particularly for large datasets, real-time analytics, and applications that require quick responses.
* It supports rich **querying and indexing**, including geospatial and text search.
* Furthermore, MongoDB integrates well with modern development stacks, supports a wide range of programming languages, and offers powerful tools for **backup, monitoring, and cloud deployment**.
* Overall, MongoDB is a strong fit for big data, IoT, mobile, content management, and other applications requiring speed, scalability, and schema flexibility.

**CRUD Operations in MongoDB**

CRUD stands for Create, Read, Update, and Delete—the four basic functions used to interact with any database system, including MongoDB. These operations allow developers to manage documents stored in MongoDB collections, providing the core functionality required in every application.

**1. Create**

The Create operation adds new documents to a collection. In MongoDB, the insertOne() or insertMany() method is used for this purpose. For example:

**db.students.insertOne({ name: "Alice", age: 25, course: "Computer Science" })**

This command creates a new document inside the students collection. In MongoDB Compass, this can be done by clicking the “Insert Document” button and entering the JSON data.

**2. Read**

The Read operation is used to retrieve documents from a collection. The most common method is find(), which can return one or more documents that match a specific query. For instance:

**db.students.find({ name: "Alice" })**

This command retrieves all documents from the students collection where the name is "Alice". In MongoDB Compass, you can perform this operation using the query bar at the top of the collection view.

**3. Update**

The Update operation modifies existing documents. You can use methods like updateOne() or updateMany() with the $set operator to change the values of specific fields. Example:

**db.students.updateOne({ name: "Alice" }, { $set: { age: 26 } })**

This updates the age of the student named "Alice" to 26. In Compass, this is done by clicking the pencil/edit icon next to the document and modifying the fields.

**4. Delete**

The Delete operation removes documents from a collection. You can use deleteOne() to delete a single document or deleteMany() to remove multiple documents at once. Example:

**db.students.deleteOne({ name: "Alice" })**

This deletes the first document that matches the name "Alice". In Compass, documents can be deleted by clicking the trash bin icon next to the desired record.