# Mongo DB

## INTRODUCTION:

MongoDB: A Flexible Document-Oriented Database

MongoDB is a popular open-source NoSQL (non-relational) database system designed for storing and retrieving data in a flexible and scalable manner. It's a powerful choice for modern applications that deal with:

Large and complex data structures: MongoDB stores data in flexible documents, similar to JSON objects. Documents can contain various data types, including nested structures and arrays, making them ideal for representing real-world entities and their relationships.

Evolving data models: Unlike relational databases with rigid schemas, MongoDB allows documents within a collection (analogous to a table) to have different structures. This flexibility is crucial for applications where data models are likely to change over time.

High performance and scalability: MongoDB is built for horizontal scaling, meaning you can easily add more servers to handle growing data volumes or increased query load. It offers features like sharding to distribute data efficiently across a cluster.

Data Model and Schema Analysis: This report delves into the design of your MongoDB data model. It explains how collections and documents are structured, how effectively they represent your data, and potential areas for optimization.

Migration Planning: If you're considering migrating data to MongoDB, this report would assess the feasibility and potential challenges. It might compare data models, analyze query performance implications, and outline the migration process.

Security Assessment: This report focuses on the security posture of your MongoDB deployment. It would evaluate access controls, encryption strategies, and any potential security vulnerabilities.

Evaluating Performance: This type of report focuses on measuring the query performance, throughput, and scalability of MongoDB for a specific application. It might analyze factors like average query time, latency, and how well MongoDB handles increasing data volumes or workloads.

Comparing MongoDB to Relational Databases: This report pits MongoDB against a relational database for a particular use case. It would highlight the strengths and weaknesses of each approach, considering factors like data model complexity, query patterns, and scalability requirements.

# SQL vs. NoSQL in MongoDB

| Feature | SQL (Relational Databases) | MongoDB (NoSQL) |
|---|---|---|
| Data Model | Structured tables with rows and columns (fixed schema) | Flexible documents (JSON-like) with dynamic schema |
| Schema | Predefined schema enforced for all data | Flexible schema, documents within a collection can have different structures |
| Relationships | Defined through foreign keys and joins between tables | Embedded documents or references within documents |
| Scaling | Primarily vertical scaling (upgrades) | Horizontal scaling (adding more servers) |
| Queries | Uses SQL for querying data | Uses its own query language specific to document structure |
| ACID Properties | Typically enforces ACID properties (Atomicity, Consistency, Isolation, Durability) | Often prioritizes Availability over Consistency (following the CAP theorem) |

## SQL and NoSQL Databases

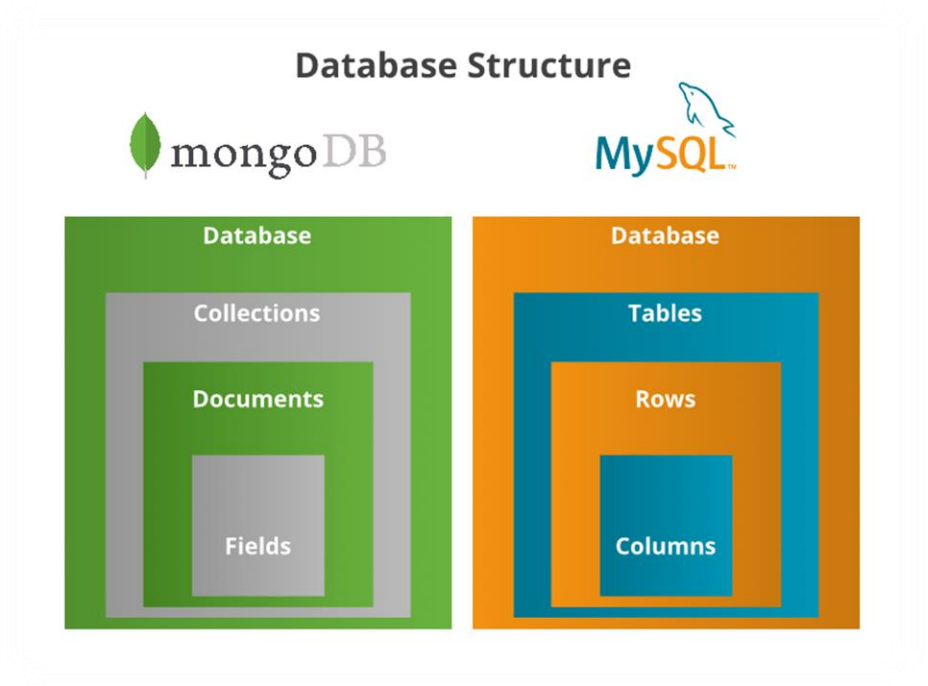
- **SQL (Structured Query Language):** This is a language for relational databases. Data is organized in fixed tables with rows and columns, enforcing a predefined schema. SQL excels in handling well-structured data and complex queries that involve joining data from multiple tables. It also ensures data integrity through transactions.
- **NoSQL (Not Only SQL):** This refers to non-relational databases. They offer more flexible schema for storing data in various formats like documents (MongoDB), key-value pairs, or graphs. NoSQL databases are a good choice for large, unstructured, or frequently changing datasets, as well as for high scalability needs. They typically use different query languages depending on the data model.

## MongoDB as a NoSQL Database

MongoDB is a document-based NoSQL database. Documents are like JSON objects with flexible schema, allowing you to store complex data structures naturally. It leverages the strengths of NoSQL by providing:

- **Schema Flexibility:** Add new fields to documents without altering the entire schema, unlike the rigid structure of SQL tables.
- **Horizontal Scalability:** Easily add more servers to handle growing data or user load, whereas SQL databases often require expensive vertical scaling (upgrading hardware).

- **Performance:** Optimized for queries on documents, which can be faster than joining tables in SQL.




## Database Structures in MongoDB

1. **Fields:** These are the fundamental building blocks of documents in MongoDB. They represent specific attributes or properties within a document, similar to columns in a relational database table. Each field has a unique name (key) and a corresponding value. Values can hold various data types like strings, numbers, booleans, dates, arrays, or even nested documents.

   For example, in a document representing a customer, a field named "name" might have a value of "Tiyansha", while a "balance" field could hold a number like “100.50”.

2. **Document:** This is the core unit of data storage in MongoDB. Imagine a document as a self-contained record resembling a JSON object. It consists of a collection of key-value pairs (fields). A document can hold various data types within its fields, allowing you to store complex information. Documents are like envelopes containing details about a particular entity (e.g., customer, product, order).

3. **Collection:** Think of a collection as a group of similar documents, analogous to tables in a relational database. However, unlike tables with a fixed schema, collections in MongoDB offer more flexibility. Documents within a collection can have different field structures as long as they share some relevance to a specific domain or category. For instance, a "customers" collection might hold documents for each customer, with each document containing details like name, address, and purchase history.

4. **Database:** This is the highest level of organization in MongoDB. It acts as a container that holds multiple collections. Think of a database as a library with different sections (collections) for various categories of books (documents). Databases are typically created for a specific purpose, like storing user data or product information. A database can have many collections, allowing you to segregate data logically.


```
_id: ObjectId('66570191c2173a0885516126')
name : "Student 948"
age : 19
courses : "['English', 'Computer Science', 'Physics', 'Mathematics']"
gpa : 3.44
home_city : "City 2"
blood_group : "O+"
is_hotel_resident : true
```


In above example **,**

**Fields** are "_id" **,**"name" **,**"age" , "courses" , "gpa" , "home_city" , "blood_group" "is_hostel_resident" .

**Document** this is JSON Object type.

**Collection** name can be saved as required here I have saved as "Students"

**Database** you might have a database named "db" that holds collections like "Students" and "Information" to store user information.