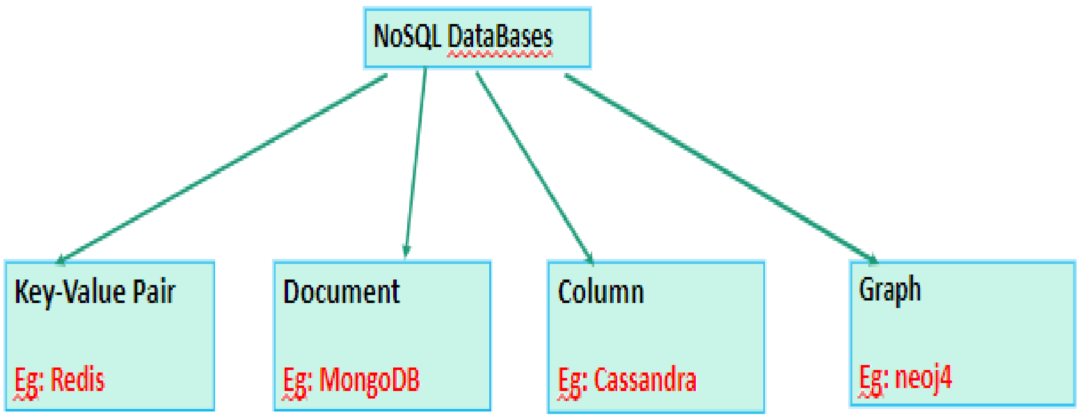
Session 12: Introduction to MongoDB (NoSQL)

- MongoDB is an open source, nonrelational database management system that uses flexible documents instead of tables and rows to process and store various forms of data. MongoDB name is derived from the word "Humongous" which means huge, enormous. it provides an elastic data storage model that enables users to store and query multivariate data types with ease.
- Install MongoDB Community Edition
  https://www.mongodb.com/try/download/community
- MongoDB documents or collections of documents are the basic units of data. Formatted as Binary JSON, these documents can store various types of data and be distributed across multiple systems.
- Dwight Merriman, Eliot Horowitz, and Kevin Ryan created MongoDB in 2007.
- MongoDB was first developed by **10gen Software in 2007** as part of a proposed platform as a service solution.
- The firm switched to an open-source development approach in 2009, with commercial support and additional services available.
- MongoDB Inc. replaced 10gen as the company's name in 2013.
- It became a publicly traded business on October 20, 2017.
- It announced a partnership with Alibaba Cloud on October 30, 2019, to provide a MongoDB-as-a-Service solution to its clients. Customers can access BABA's managed services from any of the company's worldwide data centres.



MongoDB Features

a) **Queries:** It supports ad-hoc queries and document-based queries.

b) **Index Support:** Any field in the document can be indexed.

c) **Replication**: It supports Master-Slave replication. MongoDB uses the native applications to maintain multiple copies of data.

d) **Multiple Servers:** The database can run over multiple servers. Data is duplicated to foolproof the system in the case of hardware failure**.**

e) **Auto-sharding:** This process distributes data across multiple physical partitions called shards. Due to sharding, MongoDB has an automatic load balancing feature.

f) **MapReduce:** It supports MapReduce and flexible aggregation tools.

g) **Failure Handling:** In MongoDB, it's easy to cope with cases of failures. Huge numbers of replicas give out increased protection and data availability against database downtimes like rack failures, multiple machine failures, and data centre failures, or even network partitions.

h) **GridFS:** GridFS feature divides files into smaller parts and stores them as separate documents.

i) **Schema-less Database:** It is a schema-less database written in C++.

j) **Document-oriented Storage:** It uses the BSON format which is a JSON-like format.

k) **Procedures:** MongoDB JavaScript works well as the database uses the language instead of procedures.

MongoDB Data Types: MongoDB supports a wide range of datatypes:

- **String –** Must be UTF-8 valid
- **Integer –** Stores a numerical value of 32 bit or 64 bits depending upon the server
- **Boolean –** Stores true/ false value
- **Double –** Stores floating point values
- **Min/Max keys –** Compares a value against the lowest and highest BSON elements
- **Arrays –** Stores arrays, lists, or multiple values into one key
- **Date –** Stores the current date or time in UNIX format
- **Timestamp –** Useful for keeping a record of the modifications or additions to a document
- **Object –** Used for embedded documents
- **Object ID –** Stores the ID of a document
- **Binary data –** For storing binary data

- **Null –** Stores a null value
- **Symbol –** Used identically to a string but mainly for languages that have specific symbol types
- **Code –** For storing JavaScript code into the document
- **Regular expression –** Stores regular expression

<u>Using MongoDB?</u>

MongoDB is used by a significant number of organizations in the IT sector today as a database service for applications or data storage systems.

The following are some of the organizations that are using MongoDB.

- IBM, Uber.
- Lyft.
- Intercom
- Citrix
- Twitter
- InVision
- HTC
- T-Mobile
- Sony
- Stack.
- Castlight Health
- Accenture

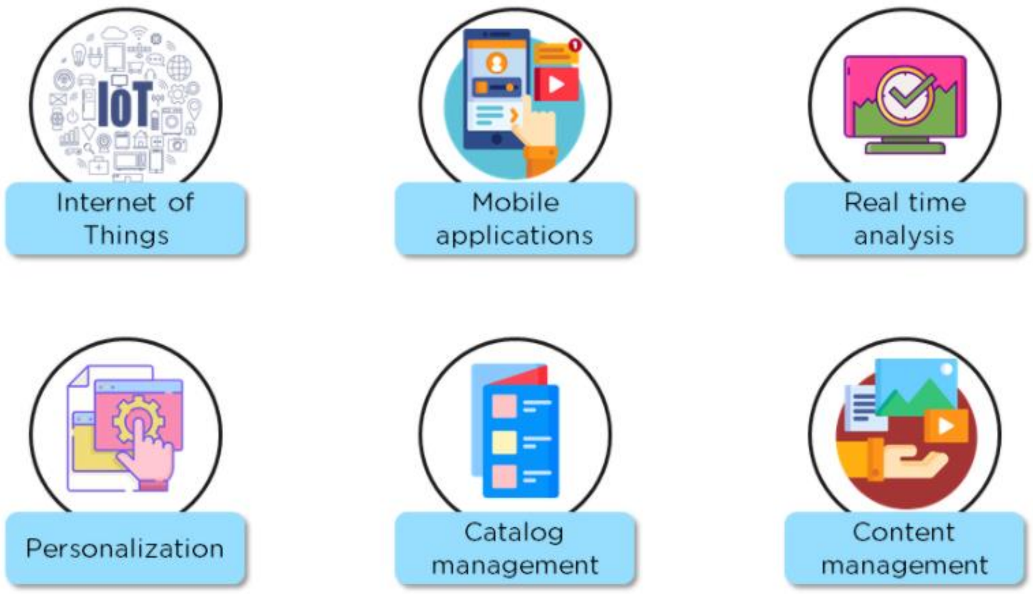
<u>MongoDB Applications in Real time:</u>

1. **Mobile applications**

   MongoDB's JSON document model lets you store back-end application data wherever you need it, including in Apple iOS and Android devices as well as cloud-based storage solutions.

2. **Real-time analytics**

   MongoDB handles the conversion of JSON and JSON-like documents, such as BSON, into Java objects effortlessly, making the reading and writing of data in MongoDB fast and incredibly efficient when analysing real-time information across multiple development environments. This has proved beneficial for several business sectors, including government, financial services and retail.



3. **Content management systems**

   Content management systems (CMS) are powerful tools that play an important role in ensuring positive user experiences when accessing e-commerce sites, online publications, document management platforms and other applications and services.

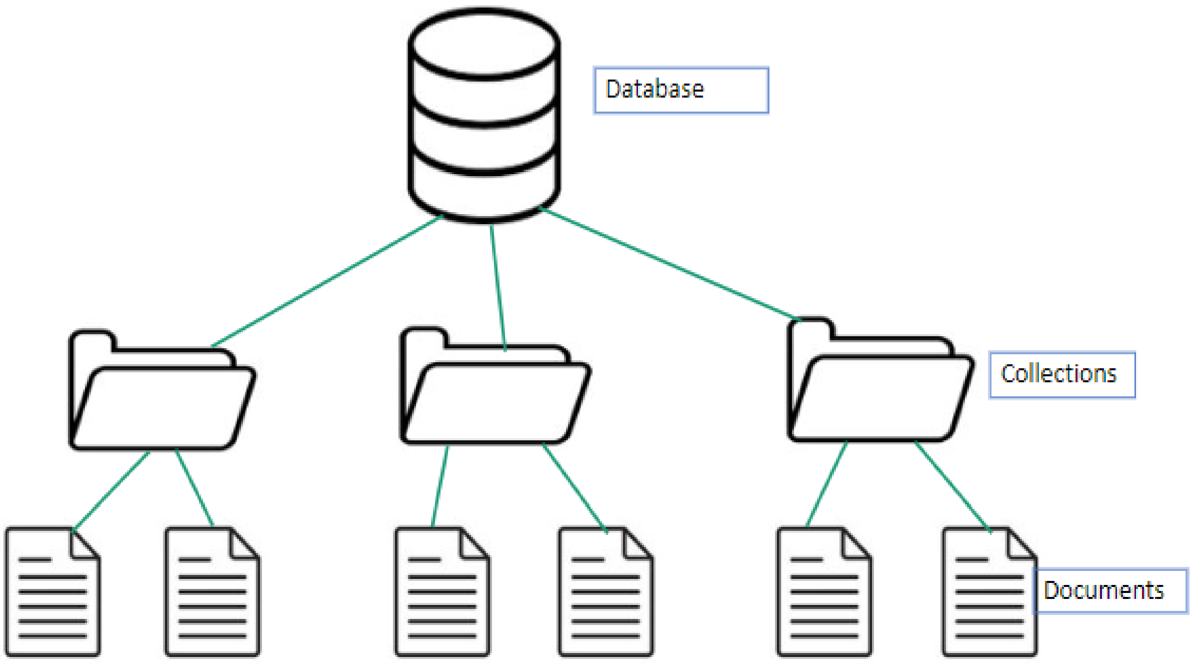
4. **Enterprise Data Warehouse**

   The Apache Hadoop framework is a collection of open-source modules, including Hadoop Distributed File System and Hadoop MapReduce, that work with MongoDB to store, process and analyse large amounts of data. Organizations can use MongoDB and Hadoop to perform risk modelling, predictive analytics and real-time data processing.

Topic 1: Performing CRUD Operations

The term "CRUD operations" refers to the standard set of database operations used to work with collections of data. CREATE, READ, UPDATE, and DELETE actions are referred to as CRUD operations in MongoDB. MongoDB falls under Document-based databases.

A. **Create:** To add new documents to a collection, use the Create operation.
B. **Read:** Data from a collection is retrieved using the Read operation.
C. **Update:** The Update operation is used to edit existing documents in a collection.
D. **Delete:** A collection of documents can be deleted using the Delete procedure.



Create a database in MongoDB

To create a new database, you can simply run any command against a non-existing database, and MongoDB will automatically create it for you.

We can create by using the **insertOne()** or **insertMany()** methods. If database is not available then first it will create the database.

insertOne(): The insertOne() method is used to insert a new document into a collection.

*[Note: **cls command will be used for clear the screen.]***

To show whether data base is created or not use

Show dbs command.

To switch between the database, use the command:

Use database_name;

Creating a New Collection:

Syntax: db.createCollection(name)

Example: db.createCollection("users")

Topic2.Creating Records: Create new documents to a collection in MongoDB:

1. insertOne(): Adding a single document to a collection is done using this method. A document to be added is the only argument it accepts, and it returns a result object with details about the insertion.

Syntax: db.collectionName.insertOne()

Example: db.users.insertOne({ name: "Angela",age: 27,});

2. insertMany()

This method is used to insert multiple documents into a collection at once. It takes an array of documents as its argument and returns a result object that contains information about the insertion.

Syntax: db.collectionName.insertMany([ ]);

Example:

```
db.users.insertMany([
 {
  name: "Angela",
  age: 27,
  },
 {
name: "Dwight",
 age: 30,
  },
   {
     name: "Jim",
  age: 29,
 }
]);
```

<u>Topic3: Accessing Data [Read Operations]</u>

1. **Read**: The process of retrieving or viewing data from your database. In MongoDB, this is done using the **find()** and **findOne()** methods.

   Syntax: db.collectionName.find(query, projection)

   Query - It specifies the selection criteria for the documents to be retrieved.

   Projection - It specifies which fields to include or exclude in the result set.

   Note: Both query and projection are optional.

   - **to include (1) or exclude (0) the field in the result set.**

A. **Find() :** **db.students.find({"regNo":"3014"})**

   Example: db.users.find()

Example: db.users.find({ age: { $gt: 29 } }, { name: 1, age: 1 })

B. **findOne():**The findOne() method is used to retrieve a single document from a collection.

Syntax: db.collectionName.findOne()

let student = db.collection('students').findOne({ name: 'Logged In Student' });

console.log(student);

Example: db.users.findOne({ name: "Jim" })

Topic4: Updating [DataUpdate Operations]

Update" operation is used to modify existing documents in a collection. We have three **updateOne(), updateMany(), or replaceOne()** methods.

A. **updateOne()**

The updateOne() method is used to update a single document that matches a specified filter.

Syntax: db.collectionName.updateOne(filter, update, options)

a.**Update**: an optional Boolean that specifies whether to insert a new document if no document matches the filter.

- If Update is set to true and no document matches the filter, a new document will be inserted. The default value of Update is false.

b. WriteConcern: an optional document that specifies the level of acknowledgement requested from MongoDB for write operations. If not specified, the default write concern will be used.

Example: db.users.updateOne({ name: "Angela" },
{ $set: { email: "angela@gmail.com" } })

Available operations:

- **$set:** Sets the value of a field in a document. If the field does not exist, the set will create it.
- **$unset:** Removes a field from a document.
- **$inc:** Increments the value of a field in a document by a specified amount.

```
db.products.insertOne(
  {
```

```
                    _id: 1,
              Product: "Car",
            quantity: 10,
            metrics: { orders: 3, ratings: 4.5 }
              }
            )
```

- **increase the "metrics.orders" field by 1**
- **increase the quantity field by -2 (which decreases quantity)**

```
            db.products.updateOne(
              { Product: "Car" },
             { $inc: { quantity: -3, "metrics.orders": 1 } }
                        )
```

db.products.find() // output

```
{
  _id: 1,
  Product: 'Car',
  quantity: 7,
  metrics: {
    orders: 4,
    ratings: 4.5
  }
}
```

- **$push:** Adds an element to the end of an array field in a document. If the field does not exist, push will create it as an array with the specified element.

  ```
  db.sunil2.updateOne({ name: "Angela" },
  { $push : { "Gender":"Male" } })
  ```
- **$pull:** Removes all occurrences of a specified value from an array field in a document.

  ```
      db.fruit.insertMany( [
      {
      _id: 1,
       fruits: [ "apples", "pears", "oranges", "grapes", "bananas" ],
  ```

```
 vegetables: [ "carrots", "celery", "squash", "carrots" ]
},
{
_id: 2,
 fruits: [ "plums", "kiwis", "oranges", "bananas", "apples" ],
vegetables: [ "broccoli", "zucchini", "carrots", "onions" ]
 }
] )
```

db.fruit.find()

B. **updateMany:** The updateMany () method is used to update multiple documents that match a specified filter.

Syntax: db.collectionName.updateMany(filter, update, options)

Example: db.users.updateMany({ age: { $lt: 30 } }, { $set: { status: "active" } })

Topic4: Deleting Data

Delete: Used to remove documents from a collection. We can delete documents using the **deleteOne( )** or **deleteMany( )** methods.

A. **deleteOne() :** The deleteOne() method is used to remove a single document that matches a specified filter.

Syntax: db.collectionName.deleteOne(filter, options)

- **filter:** Specifies deletion criteria using query operators. Specify an empty document { } to delete the first document returned in the collection
- **Options**:
  - **WriteConcern (Optional):** A document expressing the write concern. Omit to use the default write concern.
  - **Collation (Optional):** Specifies the collation to use for the operation. Collation allows users to specify language-specific rules for string comparison, such as rules for letter case and accent marks.
  - **Hint (Optional):** A document or string that specifies the index to use to support the query predicate.

Example: db.users.deleteOne({ name: "Angela" })

B. deleteMany():used to remove multiple documents that match a specified filter.

Syntax: db.collectionName.deleteMany(filter, options)

Example: db.users.deleteMany({ age: { $lt: 30 } })

db.collection('students').**deleteOne**({ name: 'Aniket' });

db.collection('students').**deleteMany**({ major: 'Undeclared' });

3. drop() : The drop() method is used to remove an entire collection.

Syntax: db.collectionName.drop()

Example: db.users.drop()   // Output: true

Note: This operation is irreversible, and all data in the collection will be permanently deleted.

Topic5: Language Bindings with MongoDb

MongoDB is a NoSQL database that is widely used for storing and managing both complex and larger data. For good integration between applications and the MongoDB database, it provides drivers that allow the developer to interact with the database using the programming language of their choice**.**

Programming Languages supported by MongoDB are **C, C++, C#, Go, Java, Kotlin, Node.js, PHP, Python, Ruby, Rust, Scala, and Swift.**

Topic6: Querying NoSQL Stores

Documents are stored and the group of documents is called "Collection". The document will be in JSON format. The data is called a "Document" and the collection of documents is called a "Collection".

The common structures adapted by NoSQL databases to store data are key-value pairs, wide column, graph, or document.

There are few key queries in NOSQL-

1. **greater than [$gt]**

test

```
{
"Brand":"Benz"
"Max_Speed":250
"Color":"Green"
}
db.test.find({Max_speed: {$gt:100}}).pretty()  // test is the database name
```

- **pretty()** method is used to configure the cursor to display results in an easy-to-read format.

2. **$eq – This operator is used to check 2 values and returns the data which is equal to the specified value.**

   db.test.find({Max_speed: {$eq:250}}).pretty()

3. **$gte (greater than or equal to)**
4. **$lte (lesser than or equal to)**
5. **$lt(less than),**
6. **$ne (Not equal)**
7. **($and) operator.**

   db.test.find({$and:[{Max_speed:{$lt:500}},{Brand: {$eq:"Benz"}}}.pretty()}]})

Topic7: Similarities Between SQL and MongoDB Query Features

MongoDB	MySQL
▪ MongoDB is an open-source document-based and NoSQL database. ▪ Document represents the hierarchical relationship using a single record. ▪ no schema that defines how the data is stored in databases.	MySQL is an open-source relational database management system Accessing any data stored in any rows or columns. MySQL is developed by Oracle.
MongoDB does not require any functionality or proper definition of a schema.	Requires a schema to define the tables in the database. The database administrator is requested to introduce the schema, which defines how different tables are stored in the databases.
In MongoDB you just need to drop the data in the BSON, JSON documents, the data, and the MongoDB database are stored in a binary format called BSON.	Required to define your tables and columns before storing anything and every row in a table must have the same columns.
it supports JSON Query Language to work with various data.	it uses Structured Query Language(SQL) to perform actions on the database, and also to create new data, retrieve data, add and update data, etc.
MongoDB database can be scaled both vertically and horizontally. Horizontal scaling means scaling by adding more machines to your resources and vertical scaling means scaling by adding more powers like the RAM, CPU, etc.	MySQL database can be scaled vertically.

The use cases of MongoDB are content management, mobile applications, IoT, etc.	Used for legacy applications or applications that require multi-row transactions, like accounting applications, etc.

Comparison Basis	**MySQL**	**MongoDB**
Definition	open-source, cross-platform, relational database management system built by Swedish Company MYSQL AB and currently supported by the Oracle.	open-source NoSQL database management system developed and owned by MongoDB Inc. that stores data in JSON-like format.
Release	It was released on 23 May 1995.	It was released on 11 February 2009.
Written in	It is written in C and C++.	It is written in C, C++, and Java.
Database Structure	Stores records in tables and can access it by using the SQL queries.	MongoDB stores record in JSON-like documents that may vary in structures.
SQL or NoSQL	MySQL uses Structured Query Language to process and access the database. We cannot change its schema. The inputs can only enter with a defined schema. SQL does not allow to work with unstructured and semi-structured data.	MongoDB is a NoSQL database system. NoSQL allows working with unstructured and semi-structured data, which is not possible in RDBMS. Its schema can be changed.
Queries Differences	Seletc , Insert, Update, delete etc.	o find, create, delete etc.
Indexes Needed	If the index is not found, the database engine searches an entire table for finding the rows.	If the index is not found, the database engine searches each document, including collection, for selecting the exact match documents.
Features	MySQL supports the following features: o It is secure ,scalable. o It follows the client-server architecture.	MongoDB supports the following features: o It supports ad hoc queries.

	○ It provides High Performance ○ It allows transactions to be rolled back, commit, and crash recovery. ○ It is flexible. ○ It support schema structure. ○ Triggers	○ It provides duplication of data running over multiple servers. ○ It supports Master-Slave Replication. ○ It has automatic load balancing. ○ It does not have any schema. ○ It uses JavaScript instead of stored procedures. ○ It supports the JSON-like data model. ○ It supports rich query language.
The Flexibility of Schema Design	Once the schema design is defined, it cannot be changed.	Its schema design can be changed, which means it supports dynamic schema.
Architecture	MySQL does not build on Distributed System Architecture.	MongoDB is Completely built on Distributed System Architecture.
Terminologies Differences	It uses: ○ Table ○ Row ○ Columns ○ Joins	It uses: ○ Collection ○ Document ○ Field ○ Embedded Document, linking

Who uses?	MySQL used organization: o Pinterest o Twitter o YouTube o Netflix o Spotify o US Navy o NASA o Walmart o Paypal	MongoDB used organization: o Klout o Citrix o Twitter o T-Mobile o Zendesk o Sony o Hootsuite o SurveyMonkey o MuleSoft o Foursquare
Scaling	It scales in vertically	It scales in Horizontally.
Latest Release Version	MySQL 8.0.21 (February 2020)	MongoDB 4.2 (February 2020)

<u>some similarities in their query features:</u>

1. **Query Language:** SQL and MongoDB both provide a query language to interact with their respective databases. SQL uses a structured language with a syntax that resembles natural language (e.g., SELECT, INSERT, UPDATE, DELETE), while MongoDB uses a JSON-like syntax with methods like find(), insertOne(), updateOne(), and deleteOne().
2. **CRUD Operations:** Both SQL and MongoDB support CRUD (Create, Read, Update, Delete) operations. You can create new records, read existing data, update records, and delete records in both systems.
3. **Filtering:** Both databases allow you to filter data based on specific criteria. In SQL, you use the WHERE clause to specify conditions for filtering records. In MongoDB, you use the find() method with query operators (e.g., $eq, $gt, $lt) to filter documents.
4. **Sorting:** SQL and MongoDB support sorting query results. SQL uses the ORDER BY clause to sort records based on one or more columns, while MongoDB uses the sort()

method to specify sorting criteria (ascending or descending) based on document fields**.**

5. **Aggregation:** Both databases offer aggregation capabilities to perform operations like grouping, counting, summing, averaging, and more on data sets. SQL uses the GROUP BY clause with aggregate functions (e.g., SUM, COUNT, AVG), while MongoDB uses the aggregation pipeline framework with stages like $group, $match, and $sort.
6. **Indexing:** SQL and MongoDB support indexing to improve query performance. Indexes help speed up data retrieval operations by creating efficient access paths to data. Both systems allow you to create indexes on specific columns/fields.

   [Note:]

   Joins (SQL-specific): One major difference is that SQL supports joins to combine data from multiple tables based on related columns. This feature is not directly available in MongoDB, as it follows a document-oriented data model where related data is typically stored within the same document or referenced using embedded documents or arrays.

Topic8: Accessing Data from Column-Oriented Databases Like HBase

Hbase column data