**MongoDB**

* MongoDB is **an open-source NoSQL database management program**.
* NoSQL is used as an alternative to traditional relational databases.
* **MongoDB is a non-relational document database**
* The MongoDB database has a flexible data model that enables you to store unstructured data, and it provides full indexing support, and replication with rich and intuitive APIs.

Note: MangoDB is case sensitive.

### **Advantages and Disadvantages of MongoDB**

**Pros:**

1. MongoDB stores data as JSON based documents that do not enforce the schema.
2. It allows us to store hierarchical data in a document. This makes it easy to store and retrieve data in an efficient manner.
3. MongoDB also allows us to split data across multiple servers.
4. MongoDB performs fast with huge data.
5. MongoDB provides drivers to store and fetch data from different applications developed in different technologies such as C#, Java, Python, Node.js, etc.

**Cons:**

1. Joins not Supported
2. High Memory Usage
3. Limited Data Size
4. Limited Nesting

### **Features of MongoDB**

**1.Replication:** is a group of mongod instances that maintain the same data set.

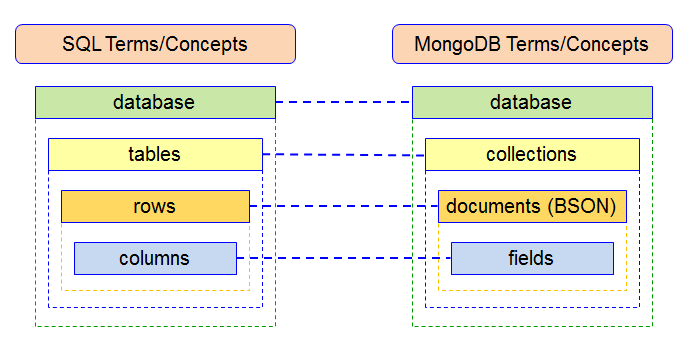
**2.Indexing:** The index stores the value of a specific field or set of fields, ordered by the value of the field. The ordering of the index entries supports efficient equality matches and range-based query operations. In addition, MongoDB can return sorted results by using the ordering in the index.

**3.File Storage:** MongoDB stores objects in a binary format called **BSON**. MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is **c:\data\db**.

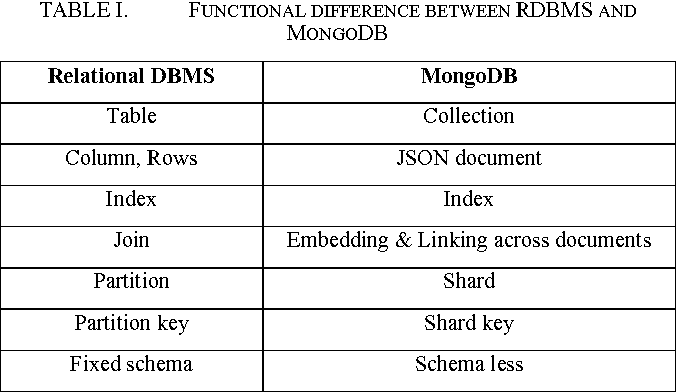
**4.Aggregation:** is a way of processing a large number of documents in a collection by means of passing them through different stages.

**5.Sharding:** is a method for distributing data across multiple machines

**Difference between MongoDB and SQL**



**Functional difference between RDBMS & MongoDB**



**Example of JSON based document**

Diagram

Description automatically generated with medium confidence

**MongoDB Commands**

1. **To show database:**

“show dbs” command to see all the databases on the connected MongoDB server.

1. **To show current database:**

“db” command to check the current database.

1. **Switch or create database:**

“use databasename” command lets us use the database if exists otherwise creates database with given name. Example: >use employeeDB

1. **Drop database:**

“db.dropDatabase()” command lets us drop(delete) the database.

1. **Inserting document:**

MongoDB provides the following methods to insert documents into a collection

1.   [insertOne()](https://www.tutorialsteacher.com/mongodb/insert-single-document#insertone) - Inserts a single document into a collection.

2.   [insert()](https://www.tutorialsteacher.com/mongodb/insert-documents) - Inserts one or more documents into a collection.

3.   [insertMany()](https://www.tutorialsteacher.com/mongodb/insert-documents) - Insert multiple documents into a collection.

* insertOne(): db.employee.insertOne({empId:1,empName:”Deepti”})

**\_id**: ObjectId('6387a013eb1f17bb8cc40bfa')

**empId**: 1

**empName**: "Deepti"

* insert():

Ex:

db.employee.insert([

{firstName:"Deepti",lastName:"Mhatre",dept:"IT",salary:60000},

{firstName:"Krisha",lastName:"Gharat",dept:"HR",salary:50000},

{firstName:"Sejal",lastName:"Mhatre",dept:"CS",salary:54000}])

Output:

**\_id**: ObjectId('6387a013eb1f17bb8cc40bfb')

**firstName**: "Deepti"

**lastName**: "Mhatre"

**dept**: "IT"

**salary**: 60000

**\_id**: ObjectId('6387a013eb1f17bb8cc40bfc')

**firstName**: "Krisha"

**lastName**: "Gharat"

**dept**: "HR"

**salary**: 50000

**\_id**: ObjectId('6387a013eb1f17bb8cc40bfd')

**firstName**: "Sejal"

**lastName**: "Mhatre"

**dept**: "CS"

**salary**: 54000

* insertMany():

db.employee.insertMany([

{firstName:"Hardik",lastName:"Mhatre",dept:"IT",salary:60000},

{firstName:"Suyash",lastName:"Patil",dept:"HR",salary:70000},

{firstName:"Aryan",lastName:"Mhatre",dept:"CS",salary:54000}])

Output:

**\_id**: ObjectId('6387a013eb1f17bb8cc40bfe')

**firstName**: "Hardik"

**lastName**: "Mhatre"

**dept**: "IT"

**salary**: 60000

**\_id**: ObjectId('6387a013eb1f17bb8cc40bff')

**firstName**: "Suyash"

**lastName**: "Patil"

**dept**: "HR"

**salary**: 70000

**\_id**: ObjectId('6387a013eb1f17bb8cc40c00')

**firstName**: "Aryan"

**lastName**: "Mhatre"

**dept**: "CS"

**salary**: 54000

1. **Finding Document:**

| **S. No.** | **Commands** | **Description** |
| --- | --- | --- |
| 1. | db.docx.findOne() | Finds one random document. |
| 2. | db.docx.find().prettyPrint() | Finds all documents. |
| 3. | db.docx.find({},{name:true,id:false}) | Displays only the names of the document Docx. |
| 4. | db.docx.find({},{name:true,id:false}) | Can find one document by attribute among many documents. |

1. **Finding Document using Operators:**

| **Operator** | **Description** | **Commands** |
| --- | --- | --- |
| $gt | greater than | db.docx.find({class:{$gt:'T'} |
| $gte | greater than equals | db.docx.find({class:{$gt:'T'} |
| $lt | lesser than | db.docx.find({class:{$lt:'T'} |
| $lte | lesser than equals | db.docx.find({class:{$lte:'T'} |
| $exists | does an attribute exist or not | db.docx.find({class:{$gt:'T'} |
| $regex | Matching pattern in pearl-style | db.docx.find({name:{$regex:'^USS\\sE'}}) |
| $type | search by type of an element | db.docx.find({name : {$type:4}}) |

1. **Find one row:**

employee> db.employee.findOne({firstName:"Deepti"})

{\_id: ObjectId("6387a130eb1f17bb8cc40bfb"),

firstName: 'Deepti',

lastName: 'Mhatre',

dept: 'IT',

salary: 60000

}

1. **Delete a document:**

MongoDB provides the following methods to delete one or more documents in a collection.

1.db.collection.deleteOne() - deletes the first matching document in a collection

2.db.collection.deleteMany() - deletes all the matching documents in a collection

* deleteOne()

db.employee.deleteOne({empId:1})

Output: { acknowledged: true, deletedCount: 1 }

* deleteMany()

db.employee.deleteMany({lastName:”Mhatre”})

Output: { acknowledged: true, deletedCount: 3 }

1. **Delete row:**

db.employee.remove({firstName:"Aryan"})

Output: { acknowledged: true, deletedCount: 1 }

1. **Sort rows:**

**#Ascending Sort:** db.employee.find()sort({firstName:1}).pretty()

**#Descending Sort:** db.employee.find()sort({firstName:-1}).pretty()

1. **Count Rows:** Counts the number of documents in a collection or a view

db.employee.find().count()

//Output: 8

1. **Limit Rows:** To limit the records in MongoDB

db.employee.find().limit(5).pretty()

1. **Update one Document:**

A document stored in the database can be changed using one of several update methods: updateOne, updateMany, and replaceOne.

1. db.collection.updateOne(): Modifies a single document in a collection.
2. db.colection.updateMany(): Modifies one or more documents in a collection

Syntax: db.collection.updateOne(filter,document,options)

Parameters:

1.filter: The selection criteria for the update , same as the find() method.

2.document: A document or pipeline that contains modifications to apply.

3.options: Optional. May contains options for update behaviour. It includes upsert, write Concern, collation, etc.

* updateOne():

db.employee.updateOne({firstName:"Krisha"},{$set:{firstName:"Pari"}})

Output:{acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0}

db.employee.updateOne({firstName:"Sejal"},{$set:{location:"Navi Mumbai"}})

{acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0}

 NOTE: Here if the specified field is not found , it add the field given.

* updateMany()

db.employee.updateMany({},{$set:{location:"Mumbai"}})

{acknowledged: true,

insertedId: null,

matchedCount: 8,

modifiedCount: 8,

upsertedCount: 0}

1. **Update an Array:**

* We can use updateOne() and updateMany() methods to add, update and remove array elements based on the specified criteria.
* It is recommended that to use the updatMany() method to update multiple arrays in collection.

db.employee.updateMany({},{$set:{skills:["java","C","MySQL","HTML","Kotlin"]}})

Output:

{

acknowledged: true,

insertedId: null,

matchedCount: 8,

modifiedCount: 8,

upsertedCount: 0

}