MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

Document Database

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.

Key Features

High Performance

- Support for embedded data models reduces I/O activity on database system.
- Indexes support faster queries and can include keys from embedded documents and arrays

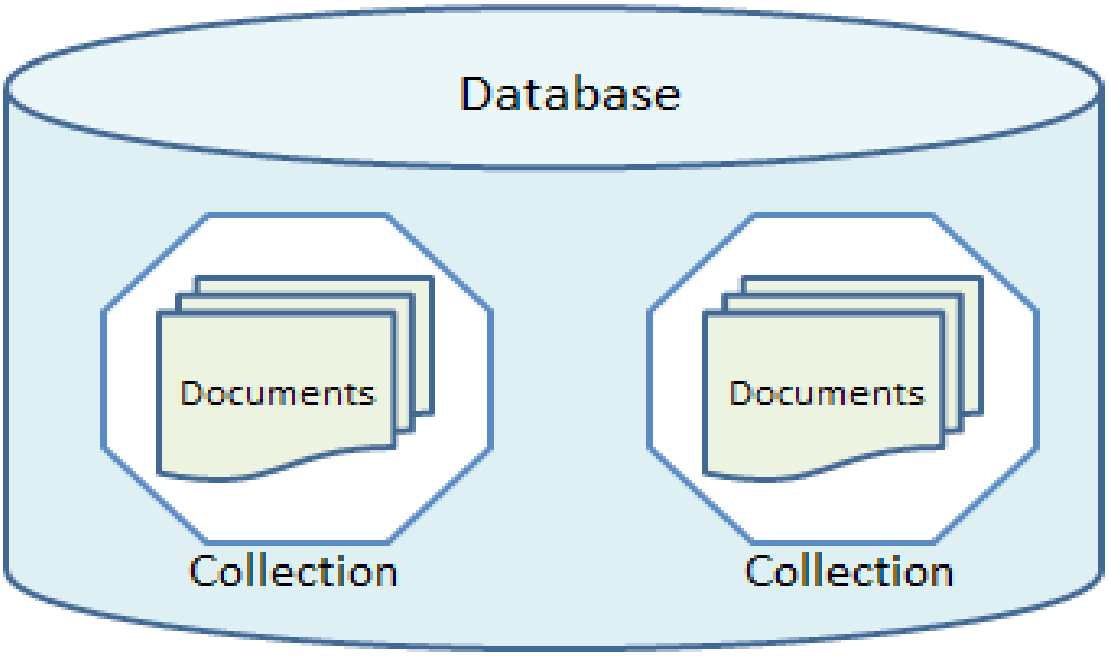
Rich Query Language

MongoDB supports a rich query language to support read and write operations (CRUD)

High Availability

MongoDB's replication facility, called replica set, provides:

- automatic failover and
- data redundancy.



• Database: This is, just like for the database, the top-level element. However,a relational database contains (mostly) tables and views. A Mongo Database on the other hand, is a physical container of a structure called a collection.

Each database has its own set of files on the filesystem. A single MongoDB server typically has multiple databases.

• Collection: This is a set of MongoDB documents. A collection is the equivalent of an RDBMS table. There can be only one collection with that name on the database but obviously multiple collections can coexist in a database. Typically, the collections contained in a database are related, although they do not enforce a schema as RDBMS tables do.

• Documents: This is the most basic unit of data in MongoDB. Basically, it is composed by a set of key-value pairs. Unlike database records, documents have a dynamic schema, which means documents that are part of the same collection do not need to have the same set of fields.

Installing and starting MongoDB

For installing MongoDB on Windows, perform the following steps:

1. Download the latest stable release of MongoDB from http://www.mongodb.org/downloads. (At the time of writing, the latest stable release is 3.0.3, which is available as Microsoft Installer or as a ZIP file). Ensure you download the correct version of MongoDB for your Windows system.

2. If you have downloaded MongoDB as a ZIP file, simply extract the downloaded file to C:\drive or any other location.

```
db.students.find()

db.students.find().pretty()

db.students.find(<<where conditions>>)

db.students.find({"studentName":"student1"})

db.students.find().{columnName:0/1}

db.students.find({}, {_id:0, city:1, state:1})

db.students.find().skip(20).limit(10)
```

Query Conditions:-

```
{ status: { $in: [ "Open", "Received" ] } }
```

$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$in	Matches any of the values specified in an array.
$nin	Matches none of the values specified in an array.

$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do not match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

$exists	Matches documents that have the specified field.

```
{ field: { $exists: <boolean> } }

db.inventory.find( { qty: { $exists: true, $nin: [ 5, 15 ] } } )
```

```
$text     Performs text search.

{
  $text:
    {
      $search: <string>,
      $language: <string>,
      $caseSensitive: <boolean>,
      $diacriticSensitive: <boolean>
    }
}

db.articles.createIndex( { subject: "text" } )

db.articles.find( { $text: { $search: "searchterm" } } )

search multiple by providing the space

db.articles.find( { $text: { $search: "bake coffee cake" } } )

Search for a Phrase

db.articles.find( { $text: { $search: "\"coffee shop\"" } } )

Exclude Documents That Contain a Term

db.articles.find( { $text: { $search: "coffee -shop" } } )

Search a Different Language

db.articles.find(
   { $text: { $search: "leche", $language: "es" } }
)

Case Sensitive Search for a Term

db.articles.find( { $text: { $search: "Coffee", $caseSensitive: true } } )


Update Operators

$inc      Increments the value of the field by the specified amount.
$set      Sets the value of a field in a document.
$unset    Removes the specified field from a document.
$min      Only updates the field if the specified value is less than the existing field value.
$max      Only updates the field if the specified value is greater than the existing field value.
```

$currentDate Sets the value of a field to current date, either as a Date or a Timestamp.

$inc

The $inc operator increments a field by a specified value and has the following form:

{ $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }

```
db.products.update(
  { sku: "abc123" },
  { $inc: { quantity: -2, "metrics.orders": 1 } }
)
```

$set

The $set operator replaces the value of a field with the specified value.

The $set operator expression has the following form:

{ $set: { <field1>: <value1>, ... } }

```
db.products.update(
  { _id: 100 },
  { $set:
    {
      quantity: 500,
      details: { model: "14Q3", make: "xyz" },
      tags: [ "coats", "outerwear", "clothing" ]
    }
  }
)

db.products.update(
  { _id: 100 },
  { $set:
    {
      "tags.1": "rain gear",
      "ratings.0.rating": 2
    }
  }
```

$unset

The $unset operator deletes a particular field. Consider the following syntax:

{ $unset: { <field1>: "", ... } }

The specified value in the $unset expression (i.e. "") does not impact the operation.

```
db.products.update(
  { sku: "unknown" },
  { $unset: { quantity: "", instock: "" } }
)
```

$currentDate

The $currentDate operator sets the value of a field to the current date, either as a Date or a timestamp.

The default type is Date.

```
db.users.update(
  { _id: 1 },
  {
    $currentDate: {
      lastModified: true,
      "cancellation.date": { $type: "timestamp" }
    },
    $set: {
      status: "D",
      "cancellation.reason": "user request"
    }
  }
)
db.students.insert(<<json data>>);

db.students.insert({
    "studentName" : "student11",
    "pincode" : 744302,
    "districtsName" : "Krishna",
    "city" : "Vijayawada",
    "state" : "Andhra Pradesh"
});

db.students.insertOne({
    "studentName" : "student11",
    "pincode" : 744302,
    "districtsName" : "Krishna",
    "city" : "Vijayawada",
    "state" : "Andhra Pradesh"
});

db.students.insertMany([{
```

```
        "studentName" : "student11",
        "pincode" : 744302,
        "districtsName" : "Krishna",
        "city" : "Vijayawada",
        "state" : "Andhra Pradesh"
},
{
        "studentName" : "student11",
        "pincode" : 744302,
        "districtsName" : "Krishna",
        "city" : "Vijayawada",
        "state" : "Andhra Pradesh"
}]);



db.students.update(<<whare conditions>>,{$set:<<values>>})

db.students.update({"studentName":"student1"},{$set:{"city":"Hyderabad","state":"Telangana"}})

db.students.remove(<<whare conditions>>})

db.students.remove({"studentName":"student11"})

 db.collection.drop():-

 Removes a collection or view from the database. The method also removes any indexes associated with
the dropped collection.

 The method provides a wrapper around the drop command.

 db.students.drop()


 DB Operations:-

  connect(url, user, password)

    Creates a connection to a MongoDB instance and returns the reference to the database.

            var db = connect("localhost:27017/myDatabase")

  Authentication

            db.auth()

```

Allows a user to authenticate to the database from within the shell.

The db.auth() method can accept either:

the username and password.

db.auth(<username>, <password>)

```
db.auth( {
                    user: <username>,
                    pwd: <password>,
                    mechanism: <authentication mechanism>,
                    digestPassword: <boolean>
                   }
  )
```

Create User

db.createUser(user, writeConcern)

Creates a new user for the database where the method runs.

db.createUser() returns a duplicate user error if the user already exists on the database.