Overview

Wednesday, July 31, 2024 9:57 PM

1. **Introduction to MongoDB**
2. **Installation & Setup**
    - *MongoDB Compass*
    - *MongoDB Shell*
    - *VS Code Integration*
3. **Important Terminology**
4. **Basic Commands**
5. **Insertion Operations**
6. **Querying Data**
7. **Updating Data**
8. **Deleting Data**
9. **Comparison Operators**
10. **Logical Operators**
11. **Indexing**
    - *Create Indexes*

- *Delete Indexes*
- *Analyse Query Performance*

12. Aggregation Pipeline

- *Match*
- *Group*
- *Aggregate*

Introduction

Wednesday, July 31, 2024 9:57 PM

What is MongoDB?

- MongoDB is a NoSQL database program that uses a document-oriented data model. It's known for its flexibility, scalability, and ease of use, making it a popular choice for modern applications that handle large amounts of diverse data types.
- NoSQL simply means Not Only SQL. This implies MongoDB can store data in various formats and not just as typical SQL tables containing ordered rows and columns.
- It gets its name from being able to store and process hu**mongo**us data.
- MongoDB's GUI is called Compass. It also has terminal version called Shell.

Key Features:

- **Document-Oriented**: Stores data in flexible, JSON-like documents.
- **Schema-Less**: Allows for dynamic schemas, which means documents in the same collection do not need to have the same structure.
- **Scalability**: Supports horizontal scaling through sharding (dividing data into chunks and distributing among nodes/servers)
- **High Performance**: Optimized for read and write performance.
- **Aggregation Framework**: Provides powerful tools for data aggregation and transformation.
- **Indexing**: Supports various types of indexes to improve query performance.
- **Geospatial Queries**: Offers built-in support for geospatial data and queries.
- **Replication**: Ensures high availability and data redundancy through replica sets.

Popular Use Cases:

- Applications that need to manage large volumes of data. MongoDB's sharding capability allows it to handle large datasets by distributing data across multiple servers.
- Applications where the data model is not fixed or is expected to evolve. MongoDB's document-oriented nature allows for a flexible schema, making it easier to adapt to changing requirements without downtime.
- Applications that require geospatial indexing and queries, such as location-based services and mapping applications. MongoDB has built-in support for geospatial data types and queries, making it easy to store and query location-based data.
- Applications that require real-time data processing and analytics.

- Applications that are deployed in the cloud and need to scale horizontally.
- Applications that manage a variety of content types, such as text, images, videos, and metadata.
- Social media platforms that require dynamic data models for user profiles, posts, comments, and relationships. MongoDB's flexibility and ability to handle complex data relationships make it suitable for social networking applications.

History of MongoDB: (self-reference)

2007: Initial Development

- MongoDB was originally developed by 10gen, a company founded by Dwight Merriman, Eliot Horowitz, and Kevin Ryan. The goal was to create a platform-as-a-service (PaaS) similar to Google App Engine.
- During development, the team realized that existing databases did not meet their needs for high performance, flexibility, and scalability, leading them to create a new database.

2009: First Open-Source Release

- MongoDB was released as an open-source project in February 2009.
- The name "MongoDB" comes from the word "humongous," reflecting the database's ability to handle large amounts of data.

2011-2012: Rapid Adoption and Features Expansion

- MongoDB gained significant traction in the developer community and among startups due to its ease of use, flexibility, and scalability.
- MongoDB 2.0 introduced improved indexing and sharding capabilities.
- MongoDB 2.2, released in August 2012, introduced the aggregation framework, enabling more complex queries and data processing within the database.

2014: Major Performance Improvements

- MongoDB 2.6 was released, featuring a new query engine, enhanced security features, and the introduction of text search.
- MongoDB Inc. raised significant funding, further fueling its growth and development efforts.

2018-2019: Enhancing Usability and Flexibility

- MongoDB 4.0 introduced support for multi-document ACID transactions, making it easier to ensure data consistency across multiple documents.
- MongoDB 4.2, released in August 2019, added distributed transactions, field-level encryption, and wildcard indexes, further enhancing its capabilities for modern applications.

2020-2021: Time Series and API Versioning

- MongoDB 4.4 brought features like the new aggregation pipeline builder, refinable sharding, and improvements in performance and stability.
- MongoDB 5.0, released in July 2021, introduced time series collections, versioned APIs, and live resharding, catering to evolving application needs.

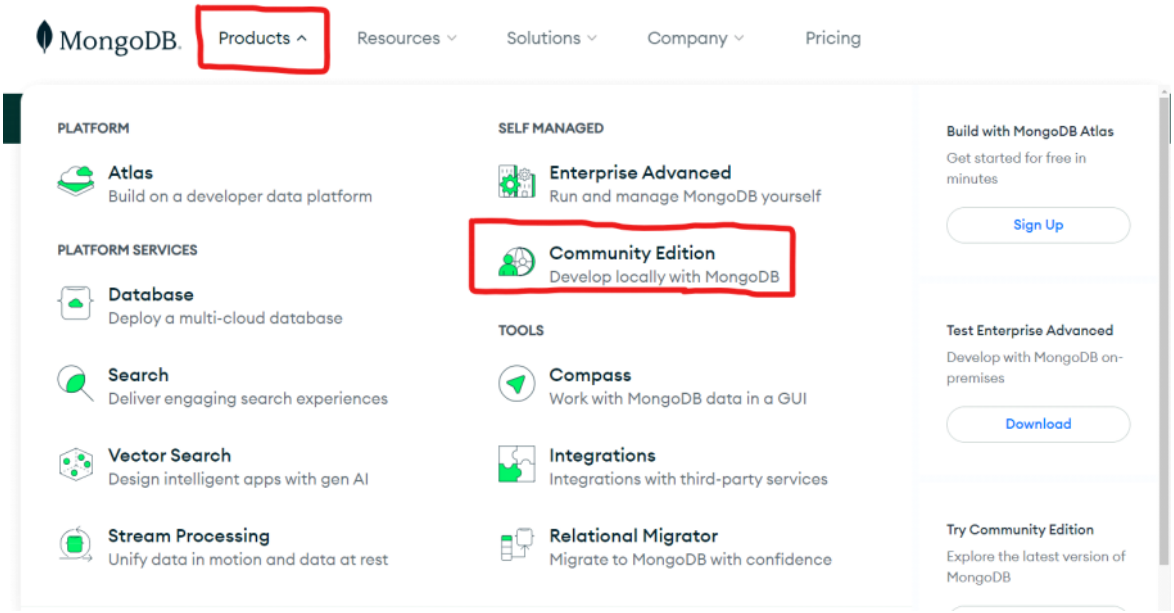
2022: Continuous Improvement

- MongoDB 6.0 continued to build on previous versions, offering enhancements in performance, security, and developer experience.
- Features included better support for cloud-native applications and improvements in data processing and storage efficiency.

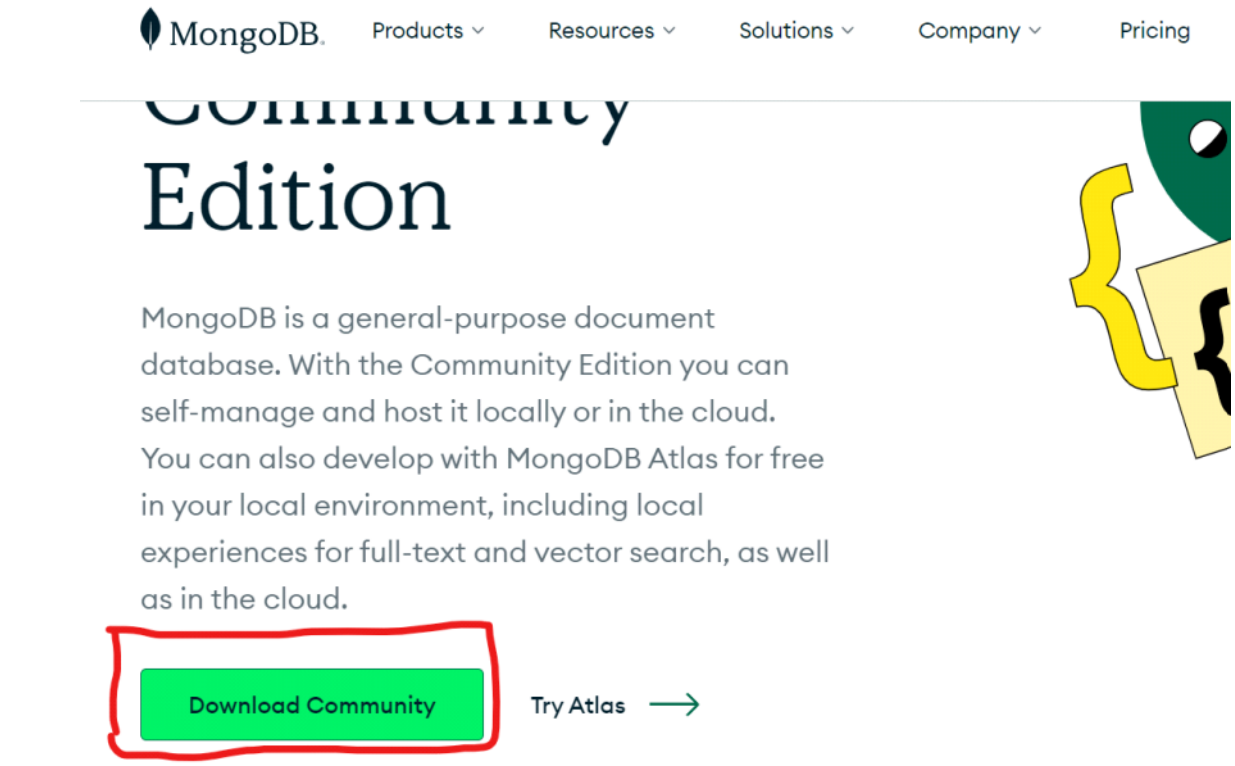
Installation & Setup

Wednesday, July 31, 2024 10:26 PM

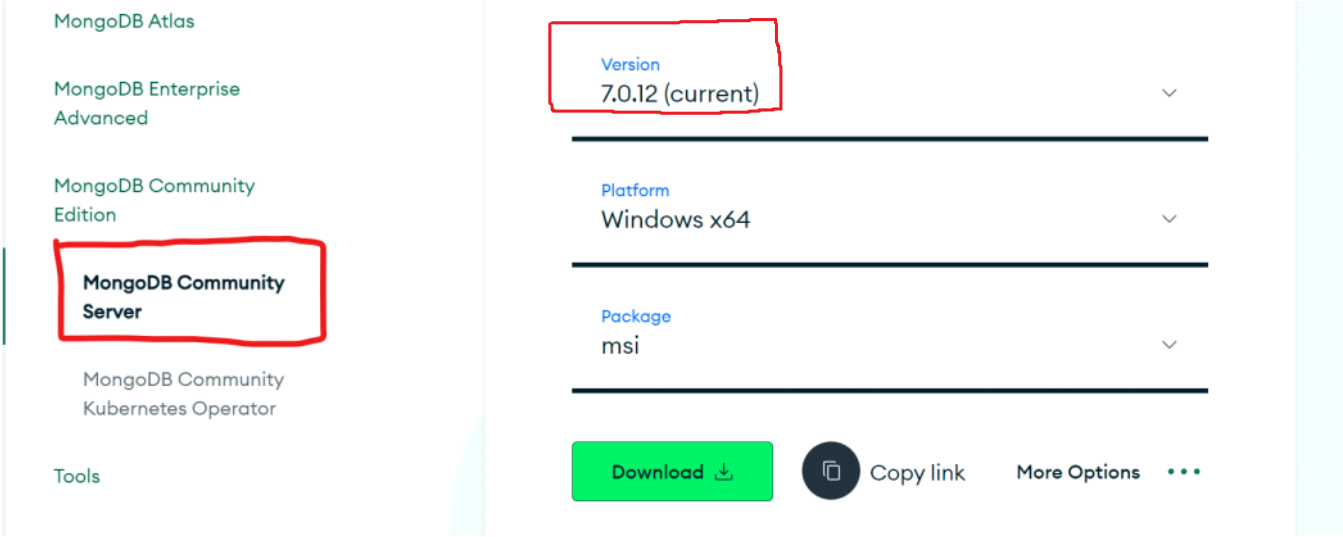
1. Click on **Products**
    - https://www.mongodb.com/
2. Click on **Community Edition**



3. Click on **Download Community**

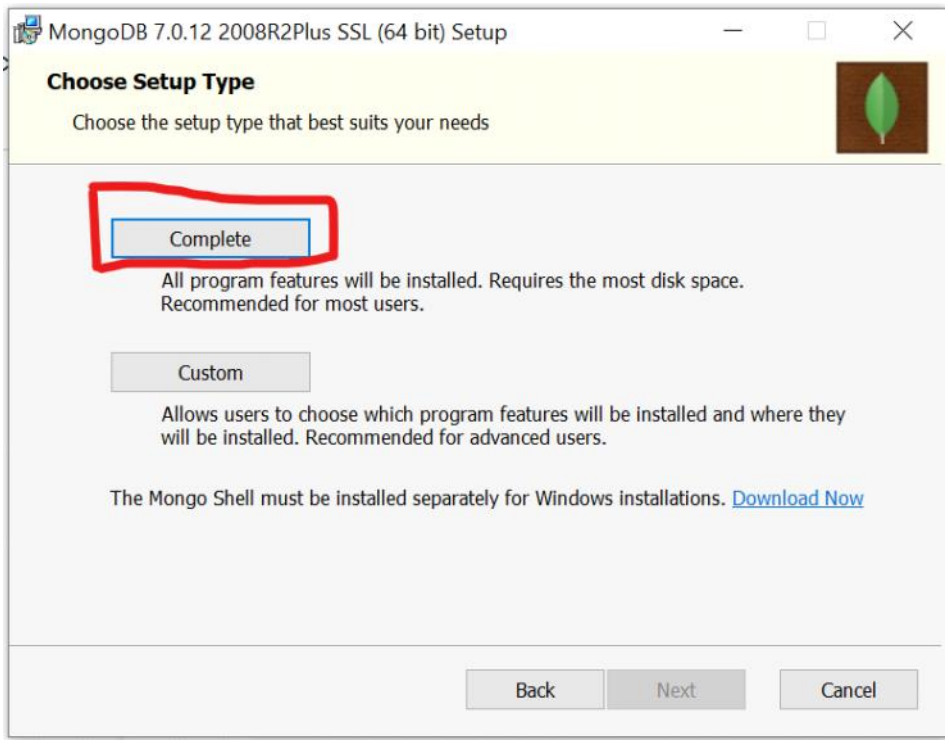


4. Download the **MongoDB Community Server**



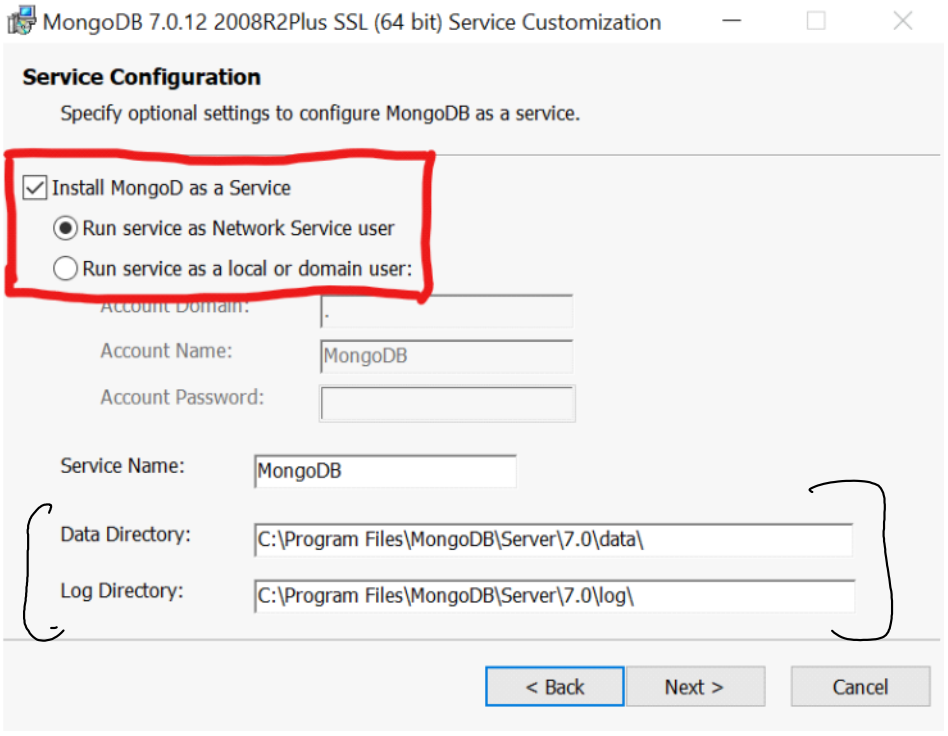
5. Follow the installation wizard

6. Click on **Complete**

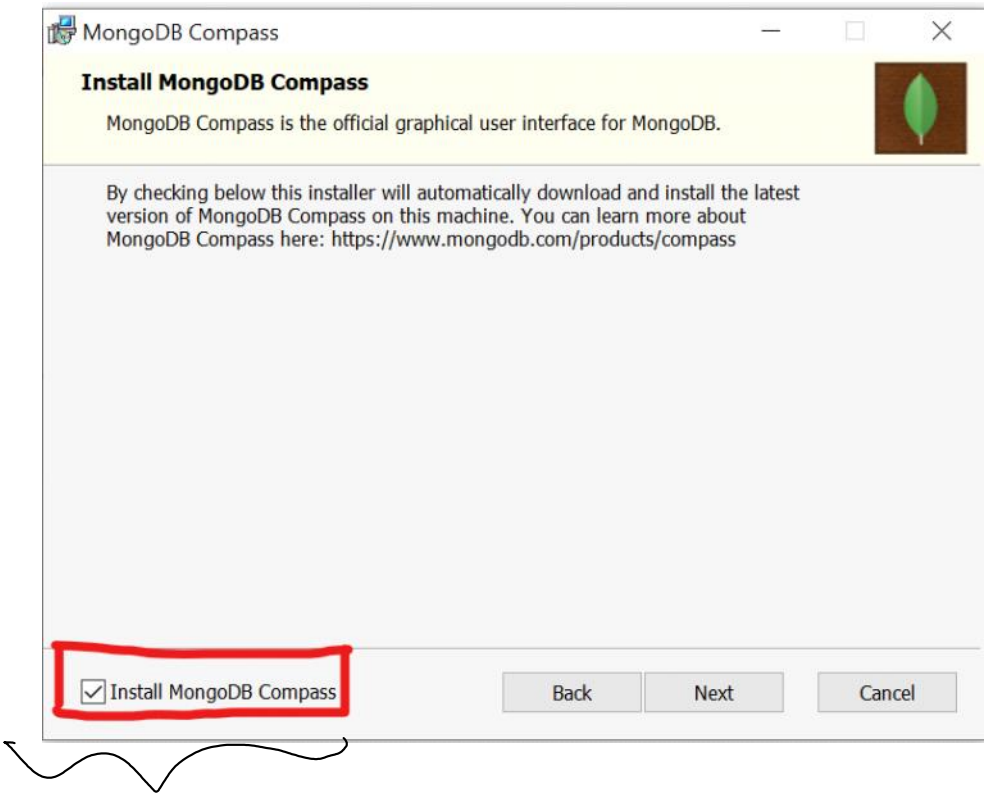


7. Ensure these settings are as is

8. Choose installation path

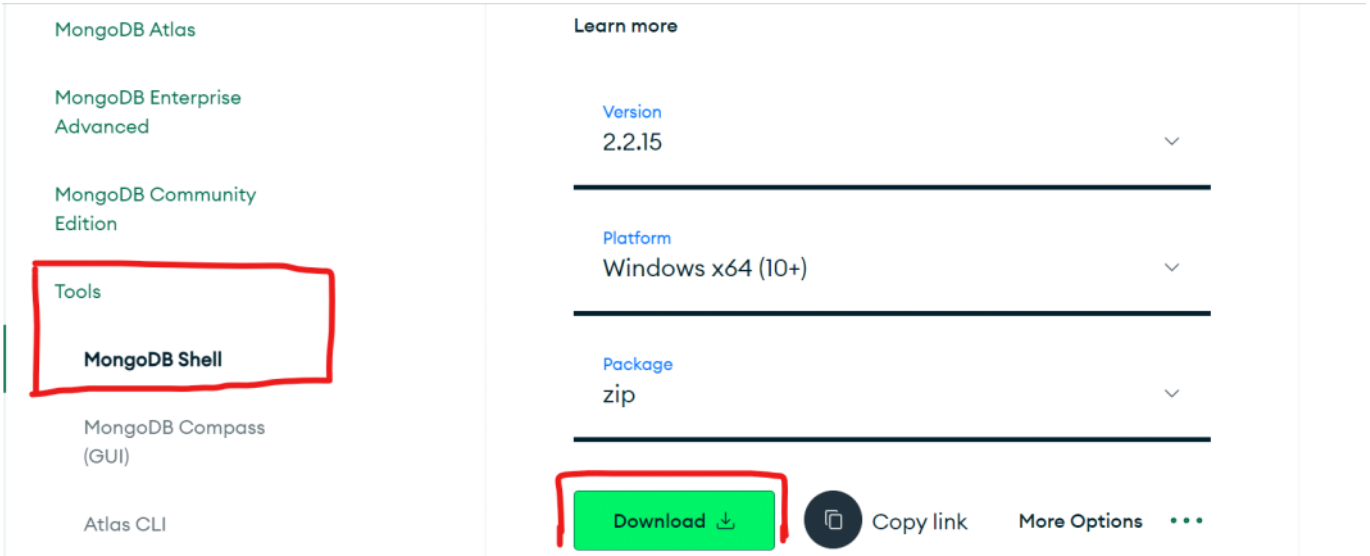


9. Install MongoDB Compass



10. Scroll down and select **Tools**

11. To install **MongoDB Shell**, click on **Download**

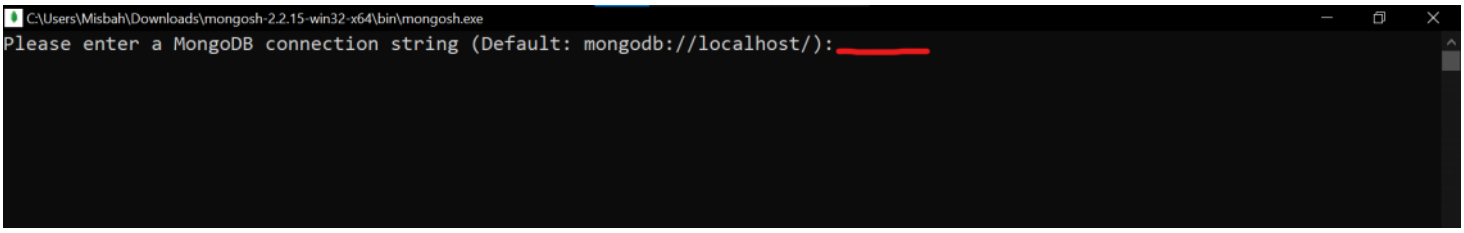


12. Extract the downloaded file

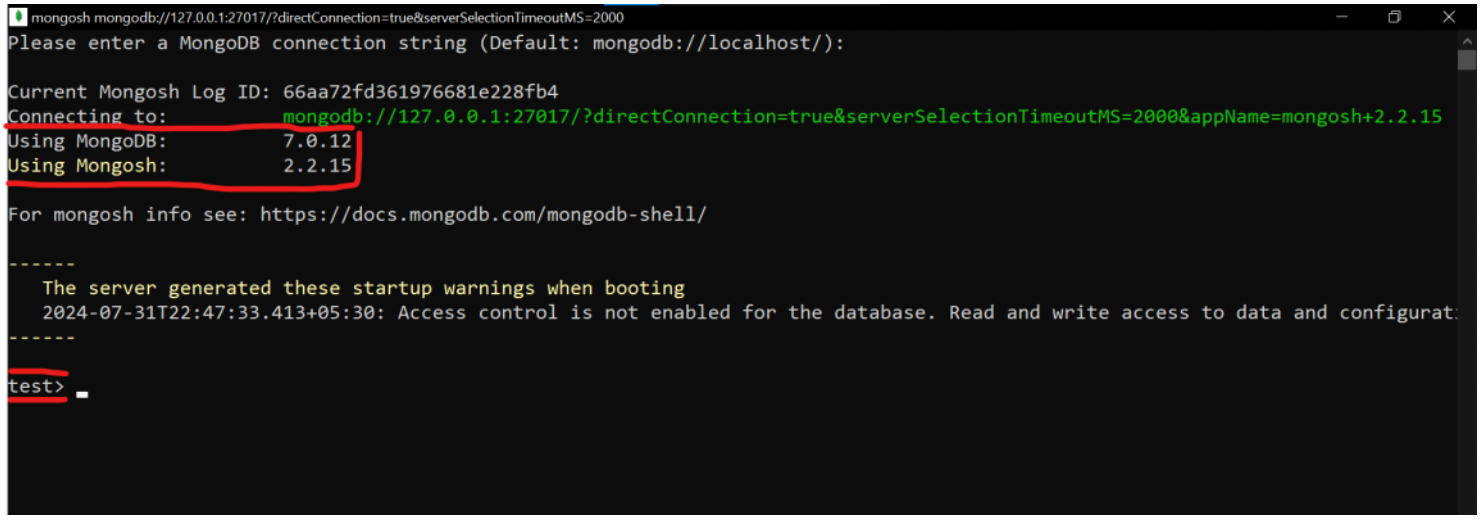
13. Open the extracted folder

14. Navigate to the **bin** folder and open the **mongosh.exe** file

15. Hit enter key when prompted to provide a **MongoDB connection string**



16. The shell should look as below:



Important Terminology

Thursday, August 1, 2024 10:56 PM

1. **Database:**
    - Similar to databases in SQL
    - Collection of data/information

2. **Collection:**
    - Similar to tables in SQL
    - Database contains several collections
    - Collections don't have any schema or pre-defined order

3. **Document:**
    - An entry in a collection
    - Similar to table rows in SQL
    - Documents in a collection can be of any length, nesting depth
    - Documents are stored in BSON format (Binary JSON)

4. **Field:**
    - A key-value pair in a document
    - Written within curly brackets {}

5. **_id:**
    - A unique identifier for each document in a collection
    - MongoDB automatically creates an **_id** field in a document if it doesn’t already have one

6. **Sharding:**
    - Refers to distributing large datasets across multiple servers
    - Involves breaking down datasets into smaller, manageable chunks

- Helps improve time efficiency to handle and process large datasets

SQL	Mongo DB
Table	Collection
Rows	Documents
features / Properties	fields

Basic Commands

Thursday, August 1, 2024 11:12 PM

1. **Show all databases:**
   - show dbs
2. **Create new / Use existing database:**
   - use <db name>
3. **Show all collections within selected database:**
   - show collections
4. **Create a new collection within a database:**
   - db.createCollection("<collection_name>")
5. **Drop a collection:**
   - db.<collection_name>.drop()
6. **Drop a database:**
   - db.dropDatabase()
5. **Show current working database:**
   - db
6. **Clear terminal screen:**
   - cls

7. Close shell:

- exit

Inserting Data

Saturday, August 3, 2024 6:56 PM

We can create a new collection within a database in 2 ways:

- By using the db.createCollection function
- By directly inserting documents using the insertOne and/or insertMany functions

1. insertOne:

- Used for adding a single document into a collection
- Provide the field values (key-value pairs) within curly brackets
- **Syntax:** db.<collection_name>.insertOne({key-value pairs})

2. insertMany:

- Used for adding multiple documents into a collection
- Each document must be provided as key-value pairs within curly brackets
- The multiple documents must be comma-separated within an array []
- **Syntax:** db.<collection_name>.insertMany([{doc1}, {doc2}, {doc3}])

Querying Data

Sunday, August 4, 2024 2:27 AM

- To query data in MongoDB, we mainly use the find method
- The find method contains 2 parameters:

```
db.<collection_name>.find({query}, {projection})
```

1. **Display all documents in a collection:**
   - Use the find method
   - **Syntax:** db.<collection_name>.find()

2. **Display documents having specific values:**
   - To display specific values, we'll use the **query** parameter
   - Simply mention the field name and specific value to search for, in document style (key-value pairs)
   - Leave the query parameter empty to display all the documents
   - **Syntax:** db.<collection_name>.find({department: "Sales"})

3. **Display specific fields and not all:**
   - For this, we'll use the **projection** parameter of the find method
   - The projection parameter always comes after the query parameter
   - Mention the field name and a Boolean flag, in document style (key-value pairs)
   - If the flag is true, the field will be display; If false, it'll be hidden
   - **Syntax:** db.<collection_name>.find({}, {_id: false, name: true})

4. **Sorting documents:**
   - Chain the sort method on the find method
   - Must specify which field(s) to sort on and in which order, within curly brackets
   - To sort in ascending order, use 1; For descending order, use -1
   - **Syntax:** db.<collection_name>.find().sort({field1: 1, field2: -1})

5. **Limit the no. of results returned:**
   - Chain the limit method
   - Must always provide an integer argument
   - If a non-integer argument or no argument is given, MongoDB throws an error

MongoInvalidArgumentError

- **Syntax:** db.<collection_name>.find().limit(2)

Updating Data

Sunday, August 4, 2024 2:27 AM

We can update data using 2 methods:

- updateOne - used for updating a single document
- updateMany - used for updating multiple documents

The update methods provides 2 parameters:

- The filter parameter is used to identify which particular document to update
- The update parameter uses operators to update the various fields accordingly
- It's a good practice to filter documents by the **_id** field

```
db.<collection_name>.updateOne({filter}, {update})
```

1. To change the field value of a single document:

- Use the $set operator to provide new value
- If the field is already existing, it'll update the value
- If the field isn't present, it'll create a new field in the document
- **Syntax:** db.<collection_name>.updateOne({_id: <id>}, {$set: {name: "Pam"}})

2. To create a new field:

- Use the same syntax as above
- Provide appropriate name and value for new field within the $set operator

3. To remove a field:

- Use the $unset operator
- Provide the field name to remove and the value ""
- **Syntax:** db.<collection_name>.updateOne({_id: <id>}, {$unset: {fullTime: ""}})

4. To update all documents with a field:

- Use the updateMany method
- Leave the filter parameter empty
- Just specify the desired field and value to add in the update parameter
- **Syntax:** db.<collection_name>.updateMany({}, {$set: {fullTime: true}})

5. To check a particular field for all documents and update accordingly:

- This is used to check a particular field in all documents
- If the filter criteria is met, those documents will be updated accordingly
- This is achieved using the updateMany method
- **Syntax:** db.<collection_name>.updateMany({fullTime: false}, {$set: {fullTime: true}})

Deleting Data

Sunday, August 4, 2024 2:28 AM

Documents can be deleted from a collection by following ways:

- Using the deleteOne or deleteMany methods
- Directly deleting from Compass GUI; this is the easiest way

1. deleteOne:

- Used for deleting a single document from a collection based on a filter criteria
- Syntax: db.<collection_name>.deleteOne({filter})

2. deleteMany:

- Used for deleting a single document from a collection based on a filter criteria
- Syntax: db.<collection_name>.deleteMany({filter})

Comparison Operators

Sunday, August 4, 2024 6:56 PM

- MongoDB offers various comparison operators to use
- These are very helpful for querying data and used with find method
- Operators in MongoDB are written along with the $ symbol
- We can combine multiple operators over a particular field

Operation	Operator
Not Equal To	$ne
Greater Than	$gt
Greater Than Equal To	$gte
Lesser Than	$lt
Lesser Than Equal To	$lte
Any one of given values	$in
None of given values	$nin

- To check for values within a numeric range, combine $gt/$gte and $lt/$lte operators
- For the $in and $nin operators, provide the sequence of values to check within an array []

Logical Operators

Sunday, August 4, 2024 7:16 PM

- MongoDB providers the AND, OR, NOR and NOT logical operators
- Each of the operators must be preceded with the $ symbol
- For the AND, OR and NOR operators, the conditions must be provided within an array []
- The NOT operator is simple wrapped around a comparison operator

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

Indexing

Sunday, August 4, 2024 7:40 PM

Indexes

Indexes support efficient execution of queries in MongoDB. Without indexes, MongoDB must scan every document in a collection to return query results. If an appropriate index exists for a query, MongoDB uses the index to limit the number of documents it must scan.

- Without indexing, MongoDB will scan each and every document in a collection to match the query statement
- Indexing helps reduce the query run time
- In MongoDB, indexes are stored in B-Trees, which are balanced tree data structures
- By default, MongoDB creates an Index for the **_id** field

1. **Create an Index:**
   - Specify the name of the field and the index order
   - To index in ascending order, use 1
   - **Syntax:** db.<collection_name>.createIndex({<field_name>: 1})

2. **Analyze Query Performance:**
   - Chain the explain method on top of the find method
   - Pass an argument executionStats
   - **Syntax:** db.<collection_name>.find().explain("executionStats")

3. **View all Indexes:**
   - **Syntax:** db.<collection_name>.getIndexes()

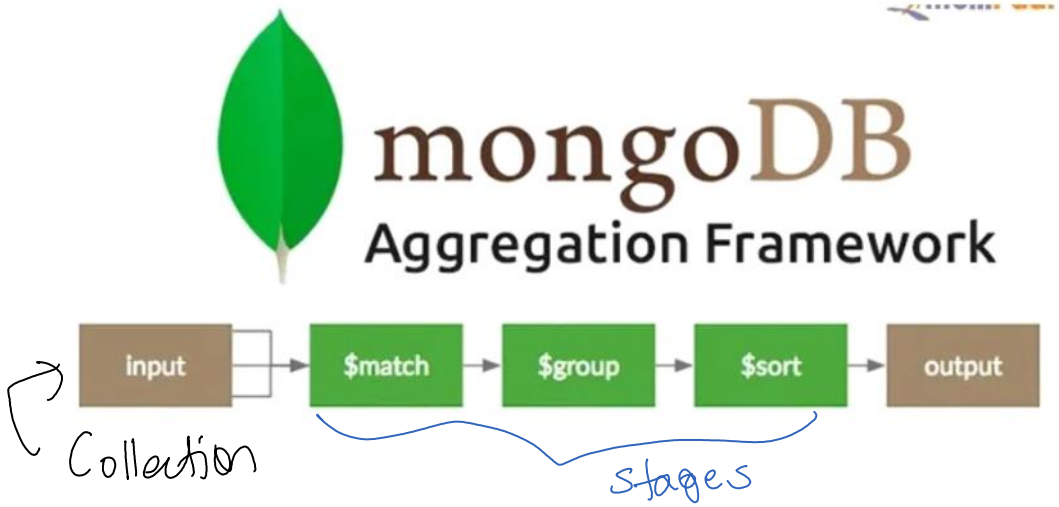
4. Delete an Index:

- **Syntax:** db.<collection_name>.dropIndex(<index_name>)

Aggregation Pipeline

Sunday, August 4, 2024 8:40 PM

- Aggregation in MongoDB is a powerful operation that processes data records and returns computed results.
- It is used to perform complex data processing tasks such as filtering, grouping, sorting, and transforming data.
- The MongoDB aggregation framework allows for the creation of sophisticated pipelines to process and analyze data.
- The aggregation pipeline includes various stages, each of which act on the collection documents in a sequential manner.

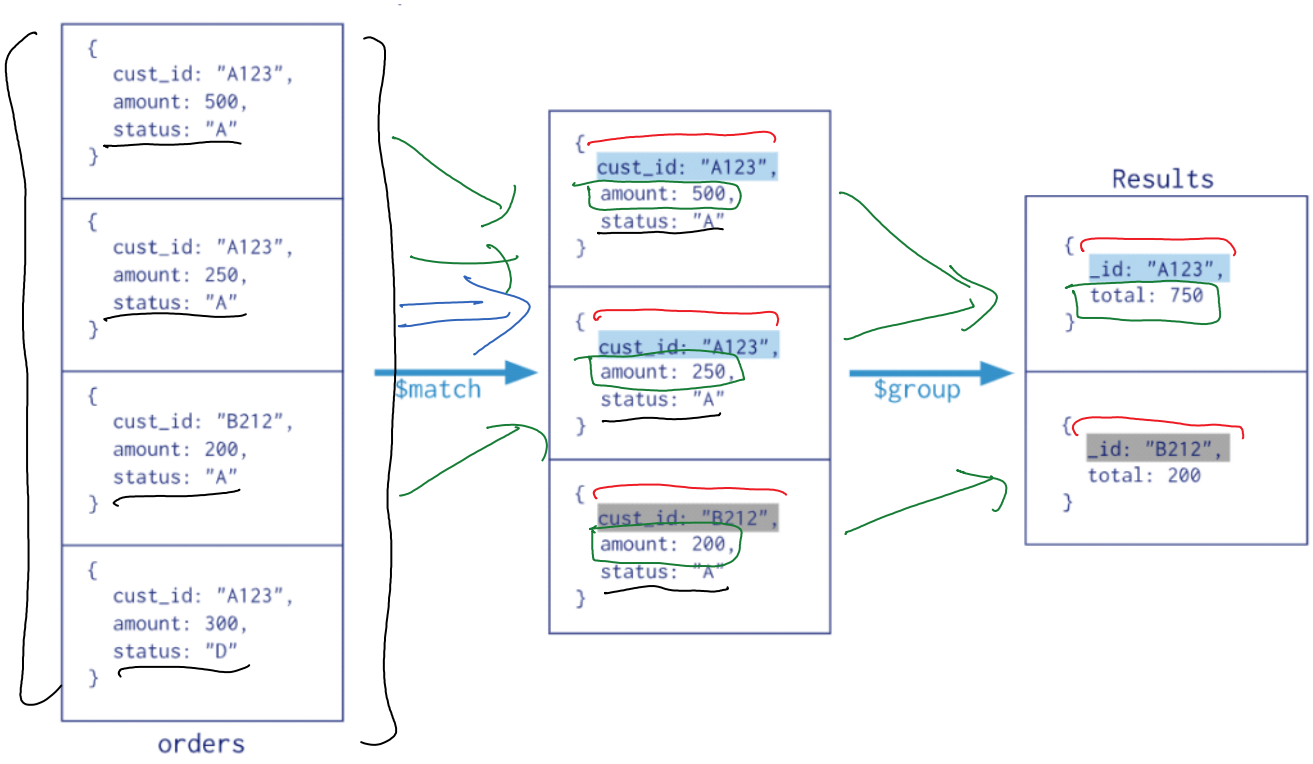


- Stages commonly used within aggregation pipeline:

STAGE	DESCRIPTION
$match	Filters documents
$group	Groups documents by a specified key and performs aggregations
$sort	Sorts documents
$project	Reshapes documents by including, excluding, or adding new fields
$limit	Limits the number of documents
$skip	Skips a specified number of documents
$unwind	Deconstructs an array field from the input documents to output a document for each element
$lookup	Performs a left outer join to a collection in the same database
$addFields	Adds new fields to documents

- MongoDB provides the following mathematical aggregation functions:

OPERATOR	DESCRIPTION
$sum	Calculates the sum of all values
$avg	Calculates average value
$min	Calculates the minimum of all values
$max	Calculates the maximum of all values
$count	Calculates the count of all values



Q. Identify the name of the department in Pennsylvania, which has the highest average salary.

- **Also determine the no. of employees in each department.**

1. Filter by location
2. Group by Department
3. No. of employees
4. Avg. Salary

4. Avg. salary
5. Dept. with Max. Salary