Page No. :
Date. : / /

Name : Devanshu Surana
Roll no: ~~23~~ 50
1032210755, Batch C1, BDT-1 Batch 2
BDT Lab Assignment 3
Problem statement:
To understand the Aggregation pipeline and indexing in MongoDB

Objectives:
1. To learn concepts of aggregation pipeline
2. To learn how to create indexes in MongoDB

Theory:

1. Aggregation Pipeline stages in MongoDB:

Between input and output, we have:
- $match stage - filters those documents we need to work with, those that fit our needs.
- $ group stage - does the aggregation job
- $sort stage - sorts the resulting documents the way we require.

2. Use of Indexing in MongoDB
   1. Improved Query Performance: Indexes accelerate query execution by enabling MongoDB to quickly locate and retrieve relevant data.
   2. Faster Sorting and Aggregation: Indexes enhance the speed of sorting and aggregation operations, making data analysis tasks more efficient.

Page No. :
Date. : / /

3. Enforced Uniqueness : Indexes support unique constraints preventing the insertion of duplicate values and ensuring data Integrity.

3. State Syntax of importing JSON into MongoDB as a collection.

mongoimport --db your_database_name --collection your_collection_name --file path/to/your/json/file.json

Platform : 64-bit Open Source Linux / Windows

Conclusion : Hence, I learned to import JSON into MongoDB, create Indexes, and execute aggregation pipeline operations.

FAQs:

Q1) Explain with syntax the concept of various indexes in MongoDB.

→ 1. Single Field Index : Simplest type of index and is created on a single field in collection.

Syntax :

db.collection.createIndex({ field_name : 1 })

2. Compound Index : A compound index is created on multiple fields. Can Improve performance of queries that involve filtering, sorting.

Syntax :

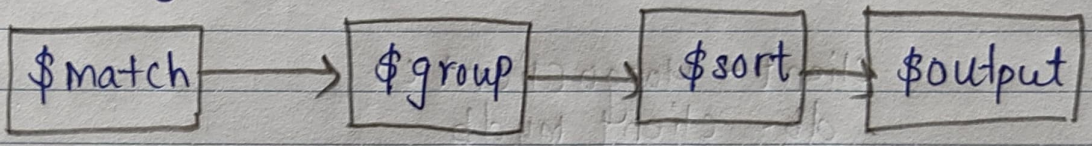
db.collection.CreateIndex({ field1 : 1, field2 : -1 })

3. Text Index : db.collection.createIndex({ textfield : 'text' })

Page No. :
Date. : / /

Q2) Explain various stages of the aggregation pipeline with a neat diagram.

Ans:



Stage 1: $match
Filters and selects documents based on a given condition.
ex: $match : {age : {$age : 18}}

Stage 2: $group
Group documents by a specified and perform aggregation.
ex: {$group : {_id : "$category", totalSales : {$sum : "Sales"}}}

Stage 3: $sort
Sort documents based on specific criteria.
ex: {$sort : {totalSales : -1}}

Q3 Explain all possible ways, along with their syntax, to import JSON/csv file in MongoDB.

Ans. 1. mongoImport command:
mongoImport --db <databasename> --collectn <collectn name> --file (path-to-file)

2. MongoShell:
load ("<path-to-json-file>");
using .load() function.

3. Programming languages and MongoDB drivers using PyMongo:

```
from pymongo import MongoClient
import json

client = MongoClient()
db = client.mydb
collection = db.mycollection

with open('data.json') as file
    data = json.load(file)
    collection.insert_many(data)
```

Q4) Explain aggregation expression with syntax.

Ans. 1. $match: filters document based on criteria

```
ex: db.collection.aggregate([
        {$match: {age: {$gte: 18}}}
])
```

2. $group: Group documents by specific fields.

```
ex: db.collection.aggregate([
        {
            $group: {
                _id: "$dept",
                avgsal: {$avg: "$salary"}
            }
        }
    ]);
```

3. Sort: sorts the documents based on specific fields.

```
ex: db.collection.aggregate([
        {$sort: {score: -1}}
    ])
```