

Task 9: Performance Tuning in MongoDB

Objective:

Understand and practice performance tuning techniques in MongoDB, including analyzing query performance and using the explain plan.

Prerequisites:

- Basic understanding of JavaScript and MongoDB.
- Node.js installation.
- MongoDB installed and running.
- A MongoDB collection with sample movie data.

Concepts:

1. Analyzing Query Performance:

- Analyze the performance of your queries to identify bottlenecks and optimize them for better performance.

Example:

JavaScript:

```
const { MongoClient } = require('mongodb');

async function analyzeQueryPerformance() {
  const uri = 'mongodb://localhost:27017'; // Replace with your MongoDB URI
  const client = new MongoClient(uri, { useNewUrlParser: true,
    useUnifiedTopology: true });

  try {
    await client.connect();
    const database = client.db('moviesDB');
    const collection = database.collection('movies');

    // Analyze query performance
    const start = Date.now();
    const result = await collection.find({ genre: 'Action' }).toArray();
    const end = Date.now();
    console.log(`Query time: ${end - start} ms`);
    console.log('Query Result:', result);
  } finally {
    await client.close();
  }
}

analyzeQueryPerformance().catch(console.dir);
```

MongoDB Compass:

- Open MongoDB Compass.

- Connect to your MongoDB instance.
- Select your database and collection.
- Click on the Explain Plan tab.
- Enter the query `{ genre: 'Action' }` and click Analyze Query to see the performance.

2. Using Explain Plan:

- Use the explain plan to get detailed information about how MongoDB processes a query.
- The explain plan provides insights into the query execution, such as the indexes used, the number of documents scanned, and the query execution time.

Example:

JavaScript:

```
const { MongoClient } = require('mongodb');

async function useExplainPlan() {
  const uri = 'mongodb://localhost:27017'; // Replace with your MongoDB URI
  const client = new MongoClient(uri, { useNewUrlParser: true,
    useUnifiedTopology: true });

  try {
    await client.connect();
    const database = client.db('moviesDB');
    const collection = database.collection('movies');

    // Use explain plan
    const explainPlan = await collection.find({ genre:
'Action' }).explain('executionStats');
    console.log('Explain Plan:', JSON.stringify(explainPlan, null, 2));
  } finally {
    await client.close();
  }
}

useExplainPlan().catch(console.dir);
```

MongoDB Compass:

- In MongoDB Compass, go to the Explain Plan tab.
- Enter the query `{ genre: 'Action' }` and click Analyze Query to see the detailed execution plan.

Instructions:


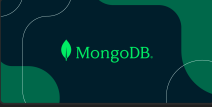
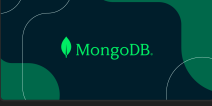
Perform the following tasks:

1. Analyze the performance of a query that retrieves all movies in a specific genre (e.g., "Action") and log the query execution time.


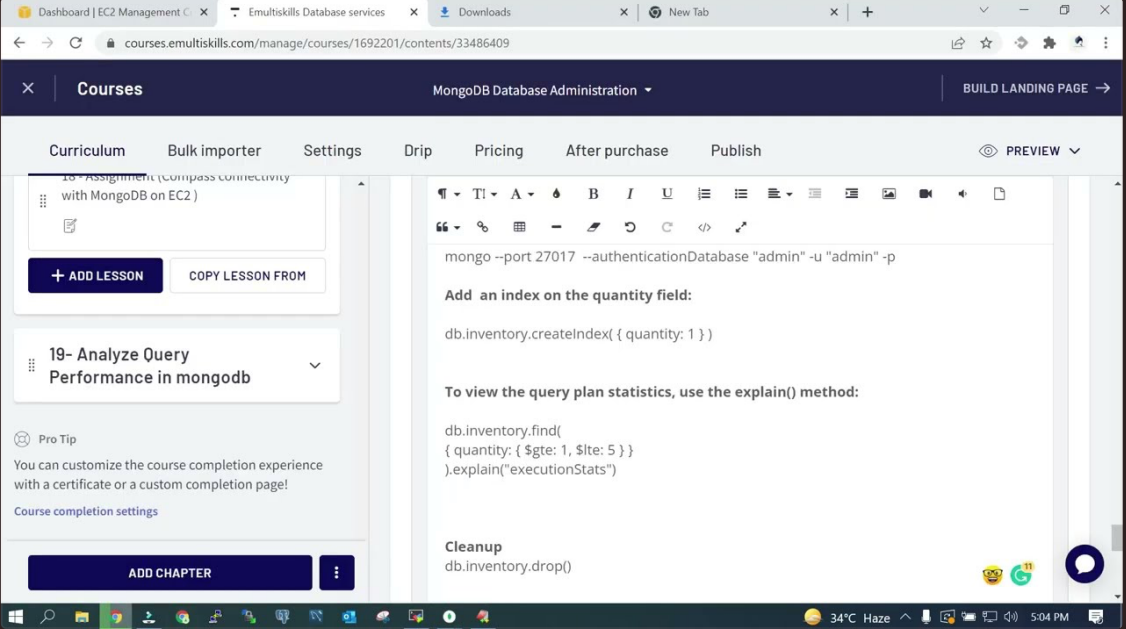
2. Use the explain plan to analyze the query execution details for a query that retrieves all movies released after a specific date.
3. Compare the performance of queries with and without indexes by running the same query before and after creating an index on a field (e.g., `release_date`).
4. Optimize a query by creating the appropriate indexes and using the explain plan to verify the improvements.

Resources:

- **Documentation:**

-  **Production Notes**
This page details system configurations that affect MongoDB, — especially when ru...
<https://docs.mongodb.com/manual/administration/production-notes/>
-  **cursor.explain()**
This page documents a .leafygreen-ui-rp2r6i{font-family:'Euclid Circular A','Helvetic...
<https://docs.mongodb.com/manual/reference/method/cursor.explain/>
-  **Indexes**
Indexes support efficient execution of queries in MongoDB. Without — indexes, Mon...
<https://docs.mongodb.com/manual/indexes/>

- **Videos:**

-  **MongoDB Courses and Trainings | MongoDB University**
Discover our MongoDB Database Management courses and begin improving your CV with Mong...
<https://university.mongodb.com/courses/M201/about>
- 

GitHub Instructions

Open in Visual Studio Code:

After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, Visual Studio Code (VSCode) will open the repository directly.

If prompted, select "Open" or "Allow" to open the repository in VSCode.

Complete the Task:

In VSCode, open the `index.js` file in the root directory of your repository and write your solution.

Ensure the `package.json` file is present and contains all necessary dependencies. If you need to install additional packages, use:

```
npm i
```

Run and Test Your Code:

Run your code to ensure it works correctly. Use the following command:

```
node index.js
```

Commit Your Changes:

Commit your changes with a meaningful message:

```
git commit -m "Completed task 9"
```

Push Your Changes to Your Forked Repository:

Push your changes to your forked repository:

```
git push origin main
```

Create a Pull Request:

Go to your forked repository on GitHub.

Click on the "Pull Requests" tab.

Click the "New Pull Request" button.

Ensure the base repository is the original template repository and the base branch is `main`.

Ensure the head repository is your forked repository and the compare branch is `main`.

Click "Create Pull Request".

Add a title and description for your pull request and submit it.

Summary of Commands

```
# Fork the repository on GitHub

# Clone the forked repository
git clone https://github.com/your-github-username/repository-name.git
cd repository-name

# Complete the task by editing index.js

# Run your code
node index.js

# Add, commit, and push your changes
git commit -m "Completed task 9"
git push origin main

# Create a pull request on GitHub
```