**Final Assignment: MongoDB Advanced Concepts**

**Course:** Advanced Database Services  
**Instructor:** Leonardo Moura  
**Due Date:** 8/13/2024

**Student name:** Priti Kharal

**Student ID:** 140701228

**Final MongoDB Assignment: Student Records Database**

**Overview**

In this assignment, we will be working with a MongoDB database containing student records. The goal is to apply what we’ve learned about MongoDB by performing various tasks, including indexing, validation, using the MongoDB Data API, and visualizing data with MongoDB Charts.

**Scenario**

I have been hired by a fictional university, Seneca Polytechnic, to manage their student records database. My tasks include optimizing the database, ensuring data integrity, and creating tools to visualize and interact with the data.

**Objectives**

1. **MongoDB Indexing and Search Optimization**
   * *Objective:* Improve the performance of queries on the student records database.
   * **Task:**
     + Create indexes on the student\_records collection to optimize searches by student names, email, and enrollment date.
     + Compare the query performance with and without indexes. Document your findings.

Ans:

To improve the performance of searches in the student\_records database, I created indexes on the fields for student names, email, and enrollment dates. Indexes help MongoDB quickly locate the data without scanning the entire collection. I then compared the query performance before and after adding these indexes. Initially, queries took longer because MongoDB had to scan all documents. After creating the indexes, the search times significantly decreased, demonstrating that the indexes efficiently sped up data retrieval. This improvement shows how indexing can optimize query performance in MongoDB.

// Index on student names

db.students.createIndex({ "name": 1 })

// Index on email

db.students.createIndex({ "email": 1 })

// Index on enrollment date

db.students.createIndex({ "enrollment\_date": 1 })

**A computer screen shot of a code

Description automatically generated**

With index:

A screen shot of a computer program

Description automatically generated

When I used the index, the query quickly found the matching command by accessing the relevant index quickly rather than looking in the whole collection.

Without index:

A screen shot of a computer

Description automatically generated

When I didn’t use the index, the query started scanning the whole collection with consumed a lot of time, especially when the database is long.

1. **MongoDB Validation**
   * *Objective:* Ensure that all student records adhere to the expected format.
   * **Task:**
     + Implement schema validation for the student\_records collection to enforce data integrity.
     + Ensure that the email field contains a valid email format, age is within a reasonable range (e.g., 18-30), and status is one of the predefined values (e.g., "active", "graduated", "dropped").
     + Test your validation by attempting to insert an invalid document and observe the results.

Ans:

The goal was to maintain data integrity in the students collection by setting up schema validation. This ensures that every student record follows certain rules for format and values. To achieve this, I implemented a schema validator for the students collection, making sure that all documents adhere to a specific structure. The schema requires that fields like student\_id, name, email, age, and status are present and validates their content to ensure consistency and correctness.

db.createCollection("students", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["student\_id", "name", "email", "age", "status"],

properties: {

email: {

bsonType: "string",

pattern: "^.+@.+$",

description: "must be a valid email and is required"

},

age: {

bsonType: "int",

minimum: 18,

maximum: 30,

description: "must be an integer in [18, 30] and is required"

},

status: {

enum: ["active", "graduated", "dropped"],

description: "can only be one of the enum values and is required"

}

}

}

}

})

A screenshot of a computer program

Description automatically generated

To test the validation, I ran the following query.

A screenshot of a computer error

Description automatically generated

As expected, MongoDB rejected the insertion of the invalid document and provided an error message pointing out the validation issues. This confirmed that the schema validation was functioning properly and enforcing the required standards.

**Conclusion:** By setting up schema validation, I made sure that every record in the students collection meets the defined criteria, preventing any incorrect or poorly formatted data from being added. This is essential for keeping the database accurate and reliable.

1. **MongoDB Data API**
   * *Objective:* Create a simple API to interact with the student records database.
   * **Task:**
     + Use MongoDB's Data API (or build your own using Node.js) to create endpoints that allow you to:
       - Retrieve all active students.
       - Add a new student record.
       - Update the status of a student (e.g., from "active" to "graduated").
       - Delete a student record.
     + Document how to use these endpoints.

Ans:

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

I've encountered a persistent MongoParseError in my Node.js application that I've tried to resolve multiple times, but the issue continues to occur. The error is related to the MongoDB connection string in my server.js file, which appears to be improperly formatted. Specifically, the connection string includes a misplaced @ symbol within the username field (pkharal:<#Priti123>), causing the parser to fail. This error suggests that the format is not correctly following the expected structure of a MongoDB URI, which requires a clear separation of the username, password, and host information. Despite my repeated attempts to correct this, the error persists, indicating there might be a recurring mistake in how I'm structuring or passing the connection string.

1. **MongoDB Charts**
   * *Objective:* Visualize the student data to gain insights.
   * **Task:**
     + Use MongoDB Charts to create visual representations of the data.
     + Create at least three charts, such as:
       - A pie chart showing the distribution of students by status ("active", "graduated", "dropped").

A screenshot of a computer

Description automatically generated

* + - * A bar chart showing the number of students enrolled each year.

A screenshot of a computer

Description automatically generated

* + - * A line chart tracking the average grade for each course over time.

A screenshot of a computer

Description automatically generated

* + - Embed these charts into a report or dashboard for easy access.

**Bonus Task (Optional)**

* **Fun with Data**:
  + Create a small web application that displays a leaderboard of the top 10 students with the highest average grades. The application should fetch data from the MongoDB database and update the leaderboard dynamically.

App.js:

A screen shot of a computer program

Description automatically generated

A black screen with many colorful lights

Description automatically generated with medium confidence

Leaderboard.ejs:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

I've been putting a lot of effort into creating leaderboard.ejs and app.js in Visual Studio Code, but no matter how many times I try, I keep running into errors. I've double-checked my code, retraced my steps, and tried running app.js multiple times, but nothing seems to work. It's been really frustrating because I can't seem to figure out what's going wrong, and I feel stuck.

Here is the vs code terminal with error:

A screenshot of a computer code

Description automatically generated

**Submission**

* Submit your MongoDB scripts, API code, and any additional files (charts, reports) in a GitHub repository.
* Provide a link to your GitHub repository in your submission, along with a brief report (one page) summarizing your findings and reflecting on what you learned.

**Evaluation Criteria**

* Correctness: Does your code work as expected?
* Optimization: Have you successfully improved query performance?
* Data Integrity: Is your schema validation thorough?
* Documentation: Are your API endpoints and charts well-documented?
* Creativity: Have you gone above and beyond to make your submission unique and engaging?

**MongoDB Database Setup**

A sample database named seneca\_students has been created for this assignment. The collection student\_records contains 100 documents representing student data with the following structure:

{

"student\_id": "S12345",

"name": "John Doe",

"email": "john.doe@example.com",

"age": 21,

"courses": [

{

"course\_name": "Database Systems",

"grade": "A"

},

{

"course\_name": "Web Development",

"grade": "B+"

}

],

"enrollment\_date": "2023-09-01T00:00:00Z",

"status": "active"

}

You can download the database https://github.com/mouraleonardo/Student\_Records\_Database\_Sample.

**Grading Criteria**

* **Indexing/Search (25%)**: Effectiveness of indexes and demonstrated performance improvements.
* **Validation (20%)**: Correctness, completeness, and robustness of schema validation.
* **Data API (25%)**: Functionality, security, and documentation of the Data API interactions.
* **Charts (20%)**: Quality, variety, and insights provided by the MongoDB Charts visualizations.
* **Bonus Task (Optional - 10%)**: Creativity and functionality of the optional web application (leaderboard).

**Submission Guidelines**

* Submit all code files, documentation, and any relevant MongoDB scripts through the course’s learning management system by the due date.
* Include a README file in your submission that explains how to run your scripts and applications.