03 MongoDB Quick Notes

MongoDB Crash Course Notes

What is MongoDB?

MongoDB is a **NoSQL**, document-oriented database.

- **NoSQL:** It doesn't rely on the traditional table-based relational model used in SQL databases.
- **Document-oriented:** It stores data in flexible, JSON-like documents (called BSON internally), allowing for a more dynamic and adaptable schema.

Advantages of MongoDB

- **Flexibility:** The schema-less nature of MongoDB allows you to easily adapt to changing data requirements without major database migrations.
- **Scalability:** MongoDB is designed for horizontal scaling, meaning you can add more servers to handle increased data volume and traffic.
- **High Performance:** MongoDB's architecture and indexing capabilities enable fast read and write operations, making it suitable for high-performance applications.
- **Rich Query Language:** MongoDB provides a powerful query language that supports complex queries, aggregations, and geospatial operations.
- **Developer Friendly:** MongoDB integrates well with popular programming languages and frameworks, and its document model often aligns closely with application data structures.

Key Concepts

1. **Documents:** The fundamental unit of data in MongoDB. Documents are similar to JSON objects and can contain various data types (strings, numbers, arrays, embedded documents, etc.).
2. **Collections:** Groups of related documents. Collections are analogous to tables in relational databases, but they don't enforce a strict schema.
3. **Databases:** Containers for collections. Databases provide a logical separation of data and can be used to organize your application's data.

Use Cases of MongoDB

- **Content Management**: MongoDB is suitable for managing content-heavy applications like blogs, news sites, and e-commerce platforms.
- **Real-time Analytics**: MongoDB can handle real-time data processing and analytics for applications that require quick insights.
- **Internet of Things (IoT)**: MongoDB is used in IoT applications to store and process sensor data from connected devices.

Important Note: Connecting MongoDB Shell to VS Code Terminal in Windows

mongosh Command Not Found If you run into the problem "mongosh is not recognizable as the name of a cmdlet" or "INFO: Could not find files for the given pattern(s). or $ mongosh $MDB_CONNECTION_STRING, bash: mongosh: command not found" or something similar. At 5:06 in the system variables, you should click on "Path" and then add the location of the file called: mongosh.exe . Otherwise, VS Code won't recognize it.

MongoDB Shell Terminal Basic Commands

Basic Navigation

1. `mongosh` - starts new MongoDB shell session and connect to a local MongoDB instance running on the default port.
2. `cls` - clears terminal screen.
3. `exit` - exits the current MongoDB shell session.

Database Operations

4. `show dbs` - shows current list of all databases.
5. `use <name of db>` - switches to specified db.
6. `use <name of db>` - this same command can create a new db.
7. `db.dropDatabase()` - drops a database.

Collection Operations

8. `db.createCollection("<add name of collection>")` - creates a collection.
9. `db.collection.drop("")` & `db.dropCollection("")` & `db.<name of collection>.drop()` - All methods can drop collections.

Document Insertion

10. `db.<name of collection>.insertOne({name:"My", age: 23})` - inserts documents into collection.
11. `db.<name of collection>.insertMany([{..},{..},{..}])` - inserts multiple documents into a collection.

Document Retrieval

12. `db.<name of collection>.find()` - shows all documents within a collection.
13. `db.<name of collection>.find({name:"Sue"})` - shows only specified documents within a collection.
14. `db.<name of collection>.find().sort({name:1})` - The "1" sorts through data in ascending order. Whether is alphabetically or numerically.
15. `db.<name of collection>.find().sort({name:-1})` - The "-1" sorts through data in descending order. Whether is alphabetically or numerically.
16. `db.<name of collection>.find().limit(1)` - Retrieves only the first document from the collection.
17. `db.<name of collection>.find().sort({gpa:-1}).limit(1)` - Retrieves the document with the highest 'gpa' value from the collection.
18. `db.<name of collection>.find({},{name:true})` - This query retrieves documents from the specified collection, but only includes specific information in this example it will only give you the name in the result.
19. `db.<name of collection>.find({},{_id:false,name:true})` - This does the same thing as above but "_id:false" prevents the terminal from showing the IDs of the documents.

Document Updates

20. `db.<name of collection>.updateOne({name:"Spongebob"},{$set:{fullTime:true}})` - $set operator: It's the most common update operator, used to set or update the value of a field. In this example it added fullTime status but if I wanted to change the name I could've add name:"another name" instead of fullTime:true.
21. `db.<name of collection>.updateMany({},{$set:{fullTime:false}})` - performs a bulk update operation on a collection. Be cautious when using updateMany with an empty filter, as it will modify

all documents in the collection. In this example, the specified collection will add a fullTime field to false, regardless of its previous value.

22. `db.<name of collection>.updateOne({name:"Spongebob"},{$unset:{fullTime:""}})` - $unset operator: This operator is used to remove a field from a document. The value provided to the $unset field doesn't matter; the field will be removed regardless you use "" or true/false.
23. `db.<name of collection>.updateMany({fullTime:{$exists: false}},{$set:{fullTime:true}})` - this query finds all collection documents that don't have a fullTime field (or have it set to null) and sets their fullTime status to true.

Document Deletion

24. `db.<name of collection>.deleteOne({name:"Larry"})` - Will search your specified collection for the first document where the name is "Larry" and remove it from the database.
25. `db.<name of collection>.deleteMany({fullTime:false})` - Deletes multiple documents from a collection in a MongoDB database. It specifically targets documents that meet a certain condition.

Comparison Operators

26. `db.<name of collection>.find({name:{$ne:"Spongebob"}})` - The "$ne" stands for not equal. This query finds all documents but excludes documents with the name "Spongebob".
27. `db.<name of collection>.find({age:{$lt:20}})` - The "$lt" stands for less than. This query finds all documents in the collection where the "age" field is less than 20.
28. `db.<name of collection>.find({age:{$lte:27}})` - The "$lte" stands for less than or equal. This query finds all documents in the collection where the "age" field is less or equal to 27.
29. `db.<name of collection>.find({age:{$gt:27}})` - The "$gt" stands for greater than. This query finds all documents in the collection where the "age" field is greater than 27.
30. `db.<name of collection>.find({age:{$gte:27}})` - The "$gte" stands for greater than or equal. This query finds all documents in the collection where the "age" field is greater or equal to 27.
31. `db.<name of collection>.find({gpa:{$gte:3,$lte:4}})` - Finds gpa greater or equal to 3 and less than or equal to 4.
32. `db.<name of collection>.find({name:{$in:["Spongebob", "Patrick", "Sandy"]}})` - the $in operator is used within a query to match any of the values specified in an array.
33. `db.<name of collection>.find({name:{$nin:["Spongebob", "Patrick", "Sandy"]}})` - The $nin operator is indeed used to exclude any of the values specified in an array from the results.

Logical Operators

34. `db.<name of collection>.find({$and:[{fullTime:true},{age:{$lte:22}}]})` - With $and you can have more than one condition it will only return a result if boths conditions are true.
35. `db.<name of collection>.find({$or:[{fullTime:true},{age:{$lte:22}}]})` - uses the $or operator to retrieve documents where either the "fullTime" field is true OR the "age" field is less than or equal to 22. It returns a result only if at least one of these conditions is met.
36. `db.<name of collection>.find({$nor:[{fullTime:true},{age:{$lte:22}}]})` - With $nor you can have more than one condition it will only return a result if boths conditions are false.
37. `db.<name of collection>.find({age:{$not:{$gte:30}}})` - retrieves all documents from the "students" collection where the "age" field is less than 30. It is similar to $lt but the main difference is that $not can retrieve null values which $lt cannot.

Logical Query Operators

Logical operators return data based on expressions that evaluate to true or false. There are four:

1. **$and:** Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
2. **$not:** Inverts the effect of a query predicate and returns documents that do not match the query predicate.
3. **$nor:** Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
4. **$or:** Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

Indexes

Index Performance Trade-off Indexes allows for the quick lookup of a field, however, it takes up more memory and slows insert, update and remove operations.

Indexes support efficient execution of queries. 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.

Index Commands

- ***Create an index:*** `db.<name of collection>.createIndex({name: 1})` - this creates an index called: name_1. FYI, the "1" is just for decreasing/increasing order. I could've used "-1" as well.
- ***Get all your Indexes:*** `db.<name of collection>.getIndexes()`.
- **Retrieve your Index:** `db.<name of collection>.find({name:"Larry"}).explain("executionStats")`.
- **Drop Indexes:** `db.<name of collection>.dropIndex("name_1")`.

When to Use Indexes Indexes are good if you are only searching but not if you are updating.

Collections

Collection Commands

- **Show collections:** `show collections`.
- ***Create collection:*** `db.createCollection("<add name of collection>")` - creates a collection.
- **Drop collection:** `db.<name of collection>.drop()` - drops a collection.

Capped Collections

```
db.createCollection("teachers",{capped:true, size:10000000,max:100},{autoIndexId:false})
```

1. **"teachers":** will be the name of the collection.
2. **"capped:true":** instructs that this collection will have a limit.
3. **"size:10000000":** sets the maximum size of the collection in bytes. In this case, it's 10 megabytes (10000000 bytes).
4. **"max: 100":** This limits the number of documents the collection can hold to 100.
5. **"autoIndexId: false":** it can be true or false. When set to false, it prevents MongoDB from automatically creating an index on the _id field. Usually, you'd want to keep this as true unless you have a specific reason not to.

Important Notes

Key Concepts Summary

1. **Document:** A single record containing data in a structured format, typically represented as a JSON object.
2. **Collections:** is a group of documents.
3. **Database:** is a group of collections.

Additional Tips 4. You can also download the extension for MongoDB on VS Code which allows connection to MongoDB & Atlas. 5. If a collection doesn't exist, MongoDB will automatically create it for you when you execute the insertOne() command. 6. When using the "use" command, MongoDB will not implicitly create a database if it doesn't exist. You will need to explicitly create the database or perform an operation like an "insert". 7. MongoDB adds an ID automatically to data submitted.

Query Syntax 8. `find({query},{projection})`: these are two optional parameters and based on arguments you can retrieve documents from a collection. "Query" is similar to WHERE in SQL and "Projection" to SELECT in SQL. If you omit query/projection, it defaults to an empty object {}.

Handling Duplicates 9. What IF you are working with a large collection and you happen to have duplicate names? Simple, each document within a collection has its own unique ID. You can "update" using the ID of the document: `db.students.updateOne({_id: ObjectId('66e2faa5ff86f6acc77c9ea3')},{$set:{fullTime:false}})`.

MongoDB Compass 10. MongoDB Compass makes adding, deleting, importing and exporting data easier than using the Shell.