query.

MongoDB Comprehensive Notes:

* MongoDB:

Definition: MongoDB is a NoSQL, document-oriented database designed for ease of development and scaling. Unlike traditional relational databases that stores data in rows and columns, MongoDB stores data in flexible JSON like documents. This allows for a more natural representation of hierarchial data and dynamic schemas.

Key features include:-

- Flexibility
- Scalability.
- Powerful Query language.
- High availability.

* Data Modelling:

As name suggest Data modelling it is a way to model NoSQL data.

Documents and Collections:

- Documents: Basic unit of data in MongoDB, similar to a record in a relational database, but with a flexible structure.

- Collections: Group of documents, like a table in a RDBMS.

Example document:

```
{ "id": "1",
  "name": "John Doe",
  "age": 29,
```

* Flexible Schema:

MongoDB's Schema-less nature allows different documents in the same Collection do have different field, structures and data types.

* This flexibility makes it suitable for handling evolving and varied data requirements.

* CRUD Operations :-

CRUD stand for Create, Read, Update, Delete we can perform these operations in mongoDB in following ways :-

1) Create :

- (Inserting, Creating Single Document)

```
db.collection.InsertOne({
        name: "Hulo 123",
        age: 30,
        address: {
            Street: "123 main St",
            City: "NY",
            State: "NY"
        },
});
```

- (Insert multiple documents):

```
db.collection.InsertMany([
        {name: "Alice", age: 40},
        {name: "Vijay", age: 60}
]);
```

operations process data records and return compiled results

2) Read : Read Document Operations.

- Find All Documents :

  Ex: db.collection.find();

- find with Criteria:

  db.collection.find({age : {$gt: 25}});

  find documents with age > 25.

- Projection : Select Specific fields

  db.collection.find({age: {$gt: 25}}, {name: 1, age: 1});

  (Selecting specific files their name and age)

3) Update : Updation of Document.

- Update Single document :

  db.collection.updateOne(

  {name: "Vijay"},

  {$set: {age: 30}}

  );

- update multiple documents :

  db.collection.updateMany(

  {age: {$lt: 35}},

  {$set: {status: "young"}}

  );

4) Delete : Deletion of Documents.

- Delete Single document:

  db.collection.deleteOne({name: "Vivek"});

- Delete multiple document

  db.collection.deleteMany({age: {$gt: 25}});

* Aggrega

Aggrega

It is

Pipelin

Aggre

Ex:

Com

1

2

3

4

* Indexing :

- Indexing improves the performance of search queries by creating a sorted data structure.

- Creating an Index

1. Single field Index :

```
db.collection.createIndex({name: 1});
```

'1' for ascending order and '-1' for descending.

2. Compound Index :

```
db.collection.createIndex({name: 1, age: -1});
```

Index types :

1. Single field Index.
2. Compound Index.
3. Multikey Index.
4. Text Index.

- Index Management :

Managing Indexes

- List Indexes

```
db.collection.getIndexes();
```

- Drop Index

```
db.collection.dropIndex("index-name");
```

* Aggregations:

Aggregation operations process data records and return computed results. It is similar to SQL groupBy operations.

Pipeline

Aggregation framework uses pipeline of stages to process documents

Ex:
```
db.collections.aggregate([
    { $match: { status: "A" } },
    { $group: { _id: "$category", total: { $sum: "$quantity" } } },
    { $sort: { total: -1 } }
]);
```

Common Aggregation Stages.

1) $match: filter document by condition.
2) $group: Group documents by a specified field and perform aggregations.
3) $sort: Sort documents by specified fields.
4) $project: Reshape documents by including/excluding fields.
5) $limit: Limits the number of documents in the output.
6) $skip: Skip specified number of documents.

* Query Optimization :

- Explain plan:

  Use explain() to analyze and understand the performance characteristics of a query.

  db.collection.find({status: "A"}).explain("executionStats");

- Index Use

  Ensure that the query indexes to improve performance. Analyze query patterns and index fields accordingly.

- Optimize aggregations :

  - Place '$match' stages early in the pipeline to reduce the no. of documents processed in subsequent stages.
  - Use indexes to support the initial stages of the pipeline.

* Security :

Authentication and Authorization :

- Authentication : Verify the identity of users connecting to the Database.
- Authorization : Grant permission to authenticated users based on roles.

Enabling Authentication

```
security:
  authorization: enabled.
```