Capped collections

What are the accumulators in Mongodb?

What is the use of $facet in mongodb?

- To process multiple aggregation pipelines within a single stage on the same set of input documents.
- Each sub-pipeline has its own field in the output document where its results are stored as an array of documents.
- So you don't need to query N times for retrieving aggregations on N groups.
- $facet enables various aggregations on the same set of input documents, without needing to retrieve the input documents multiple times.

Find the max salary of an Employee with the same name ?

```
// { "_id" : 1, "firstName" : "ank", "salary" : 1000 }
// { "_id" : 2, "firstName" : "ank", "salary" : 2000 }
// { "_id" : 3, "firstName" : "sag", "salary" : 3000 }
// { "_id" : 4, "firstName" : "sag", "salary" : 4000 }
// { "_id" : 5, "firstName" : "neh", "salary" : 5000 }
db.employee.aggregate([{$group : {_id : "$firstName", salary_maximum :
{$max : "$salary"}}}])
// Output >
// { "_id" : "ank", "salary_maximum" : 2000 }
// { "_id" : "sag", "salary_maximum" : 4000 }
// { "_id" : "neh", "salary_maximum" : 5000 }
```

Find the second highest salary in the employees collection ?

```
db.Employees.find({}).sort({"salary":-1}).limit(1) //for first highest
salary

db.Employees.find({}).sort({"salary":-1}).skip(1).limit(1) // for second
highest salary
========================================================================
// With aggregate:

db.collection.aggregate([
```

```
 { $sort: { "salary": -1} },
 { $skip : 2 },  // Skip the first 2 and get the 3rd
 { $limit: 1}    // Only get the 3rd
])
```

How to exactly match an array of 2 strings in mongodb?

- To match the array field exactly Mongo provides $eq operator which can be operated over an array also like a value.
- Also **$eq** checks the order in which you specify the elements.

```
db.collection.find({ "hobbies": {$eq: [ "singing", "Music" ] }});
```

- If u dont want order but length is 2

db.coll.find({ "hobbies": { "$size" : 2, "$all": ["2", "1"] } });

MongoDB findone with exact match but case insensitive?

```
//You can Use $options => i for case insensitive search. Giving some
possible examples required for string match.

//Exact case insensitive string

db.collection.find({name:{'$regex' : '^string$', '$options' : 'i'}})
//Contains string

db.collection.find({name:{'$regex' : 'string', '$options' : 'i'}})
//Start with string

db.collection.find({name:{'$regex' : '^string', '$options' : 'i'}})
//End with string
```

```
db.collection.find({name:{'$regex' : 'string$', '$options' : 'i'}})
//does not contain string
db.collection.find({name:{'$regex' : '^((?!string).)*$','$options' :
'i'}})
```

- The **^** matches the beginning of the input, the **$** matches the end. This way, any substrings won't match.

What is MongoDB?

MongoDB is a **document database,** which is designed for ease of application development and scaling.

with which we can store and query data and transform data with aggregations.

What is aggregation in mongodb?

- Aggregations are used to process multiple documents and return computed results.
- `db.collection.aggregate()` command to run the aggregation
- ❖ In aggregation pipeline stages appear in an array. Documents pass through the stages in sequence.
- ❖ An aggregation pipeline consists of one or more stages that process documents:
- ❖ The aggregation pipeline refers to a specific flow of operations that processes, transforms, and returns results. In a pipeline, successive operations are informed by the previous result.
- ❖ Implementing a pipeline helps us to break down queries into easier stages.
- ❖ Each stage of the aggregation pipeline transforms the document as the documents pass through it

$project, $match,$group,$sort,$skip,$limit

How does MongoDB store data?

Being document-based, MongoDB stores documents in BSON or Binary JavaScript Object notation which is the binary encoded format of JSON.

What is ObjectId? How is it structured?

- An **ObjectId** is a 12-byte BSON type.which are likely unique, fast to generate, and ordered.
- In MongoDB, each document stored in a collection requires a unique _id field that acts as a primary key. If an inserted document omits the _id field, the MongoDB driver automatically generates an ObjectId for the _id field.

The structure is –

- A **4-byte** value that represents seconds (since Unix epoch)
- a **5-byte** *random value* generated once per process. This random value is unique to the machine and process.
- a **3-byte** *incrementing counter*, initialized to a random value

What is a document and collection in Mongodb?

- A record in MongoDB is a **document**, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects.
- MongoDB stores documents in collections. **Collections** are analogous to tables in relational databases.

How to use the primary key in MongoDB?

Ans. **_id** field is reserved for the primary key in MongoDB, and that is a unique value for each document.

If you don't set anything to _id it will automatically fill it with "MongoDB Id Object". But you can put any unique info in that field.

How to perform join operations in MongoDB?

From MongoDB 3.2 onwards we can perform join operations. The new **$lookup** stage added with the aggregation pipeline is essentially similar to a left outer join.

To each input document, the `$lookup` stage adds a new array field whose elements are the matching documents from the "joined" collection. The `$lookup` stage passes these reshaped documents to the next stage.

What is a Collection in MongoDB?

- A collection in MongoDB is a group of documents. If a document is the MongoDB analog of a row in a relational database, then a collection can be thought of as the analog to a table.
- Documents within a single collection can have any number of different “shapes.”, i.e. collections have dynamic schemas.
- For example, both of the following documents could be stored in a single collection:

Does MongoDB have Schema?

Yes, MongoDB has the Dynamic Schema so there is no need to define structure to create collections in MongoDB.

What is the max size of a document in mongodb ?
Ans : - **16MB**

What are the operators in Mongodb?
Query operators :
https://docs.mongodb.com/manual/reference/operator/query/
$eq
$gte
$in :
$lt
$nin
$in
$exists
Projection operators : $,$elemMatch

The positional $ operator limits the contents of an array to return the first element that matches the query condition on the array.

```
db.students.find( { semester: 1, grades: { $gte: 85 } },{ "grades.$": 1 } )
```

The $elemMatch operator limits the contents of an <array> field from the query results to contain only the first element matching the $elemMatch condition.

```
db.schools.find( { zipcode: "63109" },{ students: { $elemMatch: { school: 102 } } } )
```

Update operators : $inc , $min, $max ,$set, $unset,$push, $pull

What are the key features in mongodb?

- Support for embedded data models reduces I/O activity on database systems.
- Indexes support faster queries and can include keys from embedded documents and arrays.

MongoDB supports a **rich query language** to support **read and write operations** (CRUD) as well as **Text Search** and **Geospatial Queries**.

How to Optimize Performance in MongoDB Aggregation

One of the best features of the MongoDB aggregation pipeline is that it automatically reshapes the query to improve its performance. Having said that, here are a few things to consider for optimized query performance.

1. Pipeline stages have a limit of 100 megabytes of RAM. If a stage exceeds this limit, MongoDB will produce an error. To allow for the handling of large datasets, use the `allowDiskUse` option to enable aggregation pipeline stages to write data to temporary files. Keep in mind that `allowDiskUse` will store the data into the disk rather than the memory, which might result in slower performance.

2.If you have multiple stages in your pipeline, it's always better to understand the overhead associated with each stage. For instance, if you have both `$sort` and `$match` stage in your pipeline, it's highly recommended that you use a `$match` before `$sort` in order to minimize the documents that need to be sorted.

What does $lookup in mongodb?

In MongoDB, we can combine data from multiple collections into one **through the $lookup aggregation stage**. In this, you have to specify which collection you want to join with the current collection and select the field that matches in both the collection

```
{
  $lookup:
    {
      from: <collection to join>,
      localField: <field from the input documents>,
      foreignField: <field from the documents of the "from"
collection>,
      as: <output array field>
    }
}
```

What is $$ROOT?

The $$ROOT variable contains the source documents for the group. If you'd like to just pass them through unmodified, you can do this by $pushing $$ROOT into the output from the group.

i.e. the top-level document, currently being processed in the aggregation pipeline stage.

```
db.entrycodes.aggregate([{

   $group: {

     _id: "$email",

     name: "$firstName"

     count: {$sum: 1},

     entries: { $push: "$$ROOT"}}

   }

 }

])
```

What are capped collections?

Capped collections are fixed-size collections, which means when we create the collection, we must fix the maximum size of the collection(in bytes) and the maximum number of documents that it can store. After creation, if we try to add more than documents to their capacity, it overwrites the existing documents. It supports high-throughput operations that are useful when we insert and retrieve documents based on insertion order.

What is the Mongo Shell?

It is a JavaScript shell that allows interaction with a MongoDB instance from the command line. With that one can perform administrative functions, inspecting an instance, or exploring MongoDB.

What is MongoDB $unwind?

The MongoDB $unwind operator is **used to deconstruct an array field in a document and create separate output documents for each item in the array**