Aggregation stages

Count,group,limit,lookup,match,merge,sort,project,unwind,unset

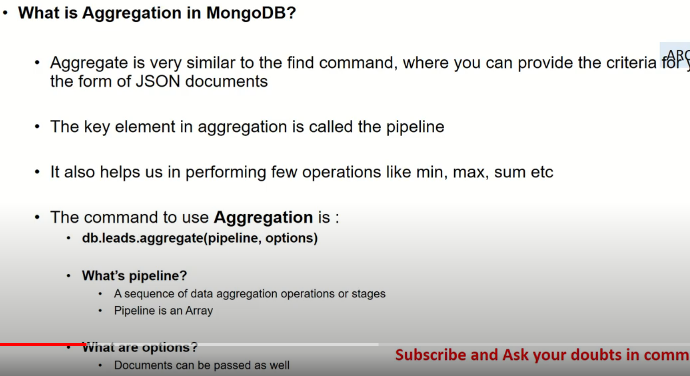
When to use this aggregation? -

If u have sequence of operations, then use aggregations

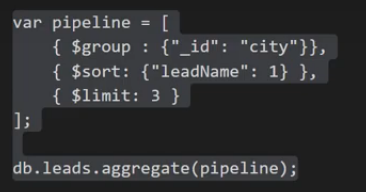
Like :1st if u want to filter ,then sort and limit only for 10 records

Means here u are using sequence of operations then aggregations is the best choice

**Pipeline is a sequence of operations**



Examples



See here ur are 3 operations, so instead of using find by operations better use aggregations

Indexing

**Index Introduction**

An **index** is required for faster retrieval of data. Indexes in MongoDB are sorted and stored as B-tree structure. There should be a balance between Indexes and queries.

Following are the index types supported in MongoDB.

* Default\_id: Each collection **contains an index named default\_id**
* **Single Field: Indexes can be either in ascending order or descending order**
* **Compound Index: used for multiple fields**
* **Text indexes: To support text search queries on string content**
* **Multikey Index: These are used to index array data**
* **Geospatial Index: Indexes used are of 2d and 2d spheres**

What happens Without index

When query in MongoDB is not indexed, a **full collection** scan will be performed. The absence of index can cause significant database performance degradation..

Documents inspected in memory should be reduced. The need to perform in-memory sorts must be removed.

Following are some of the factors to be considered for an index selection include:

What data is written to the database

What kind of data is read-only

What piece of data are used together

Rich Documents

How to create index

We can use **ensureIndex()** or **createIndex()** method to create an index in MongoDB.

**Syntax:**

db.collection.createIndex({KEY:1})

db.Player.insert(

{

"\_id": "1",

"score": 10340,

"location": { state: "NSW", city: "Sydney" }

})

**Create Index on score**

db.Player.createIndex( { score: 1 } )

* **1** for index indicates scores are sorted in ascending order
* **-1** for index indicates scores are sorted in descending orde

Since version 3.0 ensureIndex() method is deprecated.

**Example :**

db.file.createIndex({tags: 1});

**Drop Index:**

we can use **dropIndex()** command to remove or drop index from collection. Default index on the \_id field cannot be removed with dropIndexes() command.

**Example**

db.file.dropIndex({tags: 1});

**Default index**

During the creation of a collection,

* MongoDB will create a **unique index** on the ***\_id***.
* The \_id index will restrict clients from inserting two documents with the same values (duplicates) for the \_id field.

Compound Indexes

**Compound indexes:**

* On compound indexes, there will be **single index structure** that holds references to multiple fields.
* MongoDB has to limit restriction of **31 fields** for any compound index.

db.collectionname.createIndex( { <field1>: <type>, <field2>: <type2>, ... } )

db.school.createIndex({ subject: 1 ,score: 1})

db.school.createIndex({ score: 1,subject: 1 })

**Performance will vary based on the *order of fields* mentioned on Compound index**

Multi key indexes

In this index will be applied on array field

Multikey indexes are used to make efficient queries against array fields. This can be created over arrays which hold both scalar values and nested documents.

Suppose we have **Employees** collection that contains details of employees with multiple skills

try {

db.Employees.insertMany( [

{ "\_id" :"1", "Name" :"Mridhula", "EmployeeCode" : "EC01", "Country" : "IND" ,"Skills": ["java", "oracle", "Informatica"]},

{ "\_id" :"2", "Name" : "Akhila", "EmployeeCode" : "EC02", "Country" : "US","Skills": ["java", "oracle", "Informatica"]},

We can create a MultiKey Index of Skills as below.

db.Employees.createIndex({"Skills":1});