MongoDB

Vraj Suratwala

Department of ICT, VNSGU, Surat

Topics Covered

- Indexes
- Aggregations

What is Index in MongoDB ?

- indexes are allowing us to Optimize the data Retrieval.

- need of indexes ?
  - Let’s understand by one example : sometimes we need to retrieve one object or data from Database so that time the Query will search from the whole database.
  - it increases the time complexity.
  - that time we can create the index for fast data retrieval.
- Indexes store a function of data in a more searchable format.
- and one Collection can have Multiple Index.

Advantages of Indexs

- faster Quering
- Improved Aggregation
- Efficient Sorting
- Indexing on Multiple files.
- Reduces execution time.

What Official Website says about it ?

- Indexes support efficient execution of queries in MongoDB. 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.

- Although indexes improve query performance, adding an index has negative performance impact for write operations. For collections with a high write-to-read ratio, indexes are expensive because each insert must also update any indexes.

Types of Index.

- We can create an index in two ways.
  - 1. Single Field Index
  - 2. Compound Field Index.

- 1. Single Field Index : it is based on only one feild, so the index which is created on only one field that is Single Feild Index.
  - Example: A human resources department often needs to look up employees by employee ID. You can create an index on the employee ID field to improve query performance.

Types of Indexes

- 2. Compound Feild Index : Index on Multiple Field on one collection or multiple collection.
  - Example : A grocery store manager often needs to look up inventory items by name and quantity to determine which items are low stock. You can create a single index on both the item and quantity fields to improve query performance.

- Other Types : Multikey, Wildcard, Geospatial, Hashed...

Operations on Index : Create Operation

db.<collection>.createIndex({ <field>: <sort-order> })

Where Sorting Order can be -1 or 1 : ascending order (+1) and descending order (-1).

Example : Products.ele_products.createIndex({_id : 1})

Index on Embedded Fields.

- db.students.createIndex({ "location.state": 1 })

```
db.students.find( { "location.state": "California" } )
```

```
db.students.find( { "location.city": "Albany", "location.state": "New York" } )
```

Droping an Index

- Note : in any collection the _id is a default index.
- Syntax :
  - db.collection.dropIndex({field : 1});
  - you can also give the index name rather then field name.
- Example :
  - products.ele_product.dropIndex({index1})

Reading from an Index

- db.collection.getIndexes({Fieldname : value}) - simple retrieval.

- here’s the raw truth: abour Update Operation :
- You can’t directly update an existing index in MongoDB.

How to Make Unique Index in MongoDB ?

- Syntax : db.users.createIndex({email : 1},{unique : true})
- Example :

```
products.ele_product.createIndex(
        {price : 1},
        {unique : true}
)
```

What is Text Index?

- Searching using index is faster then $regex Searching.
- Syntax :
  - db.collection.find({$text : “keyword”})

- Example :
  - db.products.find({$field : {$regex : “Air”}})

When to Use Indexes ?

- When You Frequently Query a Field.
- When You Use sort() Often.
- When You Use range queries.
- When You Use text search.
- When a Field is Used in Joins ($lookup).

When to Not Use Indexes?

- Indexes on Rarely Used Fields.
- Balancing Act.
- Indexing Small Collections.
- Don't create too many indexes → slows down insert, update, delete.

How The Indexes works Internaly ?

- Reference : https://www.pankajtanwar.in/blog/how-database-indexing-actually-works-internally.
- this artical represents full concept of indexing in an easy way!

Aggregation in MongoDB

- What is Aggregations ?
  - an aggregation is the process of performing transformation on documents and combining them to produce computed results.
  - PIPELINE STAGES : Aggregations consists of Multiple pipeline stages, each performing a Specific Operation on the input area.
- Benifits :
  - Aggregation Data
  - Advanced Transformation
  - Efficient Processing

- $match - same as find()
- $group - Output in an Single value.

- Example :

```
db.products.aggregate([
    {$match : {$Query}}
])
```

- Group Example :
  - Example :

```
db.products.aggregate([
    $group : {
        _id : “$company”,
        totalProducts : {$sum : 1}
    }
])
```

also refer : $project, $push, $addToSet, $limit, $sort, $filter

Thank You

Vraj Suratwala