Indexing MongoDB

~ Hardeep Singh

Index -

Indexes are data structures that can store collection’s data set in a form that is easy to traverse. Queries are efficiently executed with the help of indexes in MongoDB.

Indexes help MongoDB find documents that match the query criteria without performing a collection scan. If a query has an appropriate index, MongoDB uses the index and limits the number of documents it examines.

Types of Indexes -

- Default _id
- Single Field
- Compound Index
- Multikey Index
- Geospatial Index
- Hashed Indexes

Default **_id** index:

Each MongoDB collection contains an index on the default _id (Read as underscore id) field. If no value is specified for _id, the language driver or the mongod (read as mongo D) creates a _id field and provides an **ObjectId** (read as Object ID) value.

```
{
    "_id" : ObjectId("5cbf4fcb37be3aaea4f00bf7"),
    "username" : "mkyong"
}
```

Single Field Index

For a single-field index and sort operation, the sort order of the index keys do not matter. MongoDB can traverse the indexes either in the ascending or descending order.

```
db.items.createIndex( { “item" : 1 } )
```

Compound Index

MongoDB supports compound indexes to **query multiple fields**. A compound index contains multiple single field indexes separated by a comma.

```
db.products.createIndex( { "item": 1, "stock": 1 } )
```

Multi key Index

To index a field that holds an **array value**, MongoDB creates an index key for each element in the array. These *multikey* indexes support efficient queries against array fields.

Example -

```
{ _id: 1, item: "ABC", ratings: [ 2, 5, 9 ] }
```

```
db.survey.createIndex( { ratings: 1 } )
```

Limitations For Compound Multi-Key Index

- You cannot create a compound multikey index if more than one to-be-indexed field of a document is an array.

```
{ _id: 1, a: [ 1, 2 ], b: [ 1, 2 ], category: "AB - both arrays" }
```

- Or, if a compound multikey index already exists, you cannot insert a document that would violate this restriction.

```
{ _id: 1, a: [1, 2], b: 1, category: "A array" }

{ _id: 2, a: 1, b: [1, 2], category: "B array" }
```

Geospatial Index

With the increased usage of handheld devices, geospatial queries are becoming increasingly frequent for finding the nearest data points for a given location.

```
db.collection.createIndex( { location : "2d" } )
```

2d (longitude, latitude) / 2dsphare (other geometric queries)

Hashed Index

When you use a hashed index on a field, MongoDB computes a hash of the field value and stores the hash in the index. Hashed indexes support only equality comparison and do not support range queries, and are typically used in **sharding scenarios**. This not work in range. Hashed indexes **cannot** be multikey.

```
db.collection.createIndex( { _id: "hashed" } )
```

Thank you

~ Hardeep Singh