Advanced index operation in MongoDB

Create an index on the array ie create separate indexes for each of its fields. So in this situation, when we create an index on the array of tags, individual indexes will be created for their values: music, cricket and blogs.

You follow the following Document of users collection:

```
{ "address" : { "city" : "Los Angeles" , "state" : "California" , "pincode" : "
```

The above document contains a Subdocument as **address** and an array of **tags** .

Index array fields in MongoDB

Suppose that we want to search the user document based on tags. To do this, we will create an index on the array of tags in that Collection.

Create an index on the array ie create separate indexes for each of its fields. So in this situation, when we create an index on the array of tags, individual indexes will be created for their values: music, cricket and blogs.

To create an index on the array of tags, you use:

```
> db . users . ensureIndex ({ "tags" : 1 })
```

After creating the index, we can perform a search on that Collection's tags field like this:

```
> db . users . find ({ tags : "cricket" })
```

To check if a valid index has been used, you use the **explain** command.

```
> db . users . find ({ tags : "cricket" }). explain ()
```

The explain command returns the result in "cursor": "BtreeCursor tags_1" which confirms that the logical index is used.

Field indexing is the Subdocument

Assume that you want to search in Documents based on city, state, and pincode fields. When all these fields are part of the address field, which is the Subdocument, you will create an index on all the fields of that Subdocument.

To create an index on all the fields of that Subdocument, you use:

```
> db . users . ensureIndex ({ "address.city" : 1 , "address.state" : 1 , "addres
```

Once the index has been created, you can search for any field in the fields of that Subdocument. Take advantage of this index as follows:

```
> db . users . find ({ "address.city" : "Los Angeles" })
```

Remember, the query expression must follow the order of the specified index. Therefore, the index created above will support the following queries:

```
> db . users . find ({ "address.city" : "Los Angeles" , "address.state" : "Cali
```

It will also support the following query:

```
> db . users . find ({ "address.city" : "LosAngeles" , "address.state" : "Califo
```

According to Tutorialspoint

Previous article: Atomic Operation in MongoDB

Lesson: Limit of index in MongoDB