

MongoDB Indexing Tutorial - createIndex(), dropindex() Example

Details

Last Updated: 26 December 2019

Indexes are very important in any database, and with MongoDB it's no different. With the use of Indexes, performing queries in MongoDB becomes more efficient.

If you had a collection with thousands of documents with no indexes, and then you query to find certain documents, then in such case MongoDB would need to scan the entire collection to find the documents. But if you had indexes, MongoDB would use these indexes to limit the number of documents that had to be searched in the collection.

Indexes are special data sets which store a partial part of the collection's data. Since the data is partial, it becomes easier to read this data. This partial set stores the value of a specific field or a set of fields ordered by the value of the field.

In this tutorial, you will learn –

- Understanding Impact of Indexes
- How to Create Indexes: createIndex()
- How to Find Indexes: getindexes()
- How to Drop Indexes: dropindex()

Understanding Impact of Indexes

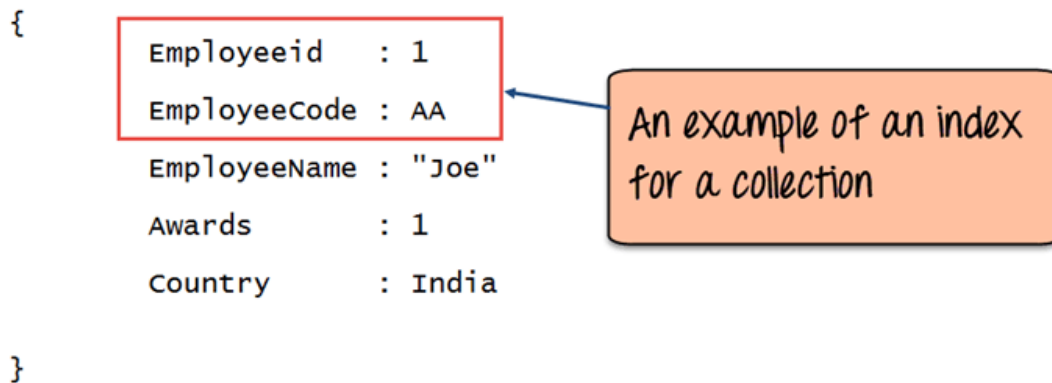
Now even though from the introduction we have seen that indexes are good for queries, but having too many indexes can slow down other operations such as the Insert, Delete and Update operation.

If there are frequent insert, delete and update operations carried out on documents, then the indexes would need to change that often, which would just be an overhead for the collection.

The below example shows an example of what field values could constitute an index in a collection. An index can either be based on just one field in the collection, or it can be based on multiple fields in the collection.

In the example below, the Employeeid "1" and EmployeeCode "AA" are used to index the documents in the collection. So when a query search is made, these indexes will be used to quickly and efficiently find the required documents in the collection.

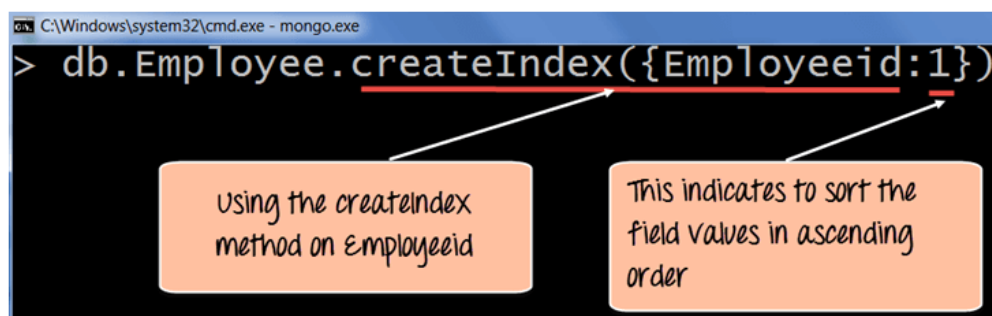
So even if the search query is based on the EmployeeCode "AA", that document would be returned.



How to Create Indexes: createIndex()

Creating an Index in MongoDB is done by using the "**createIndex**" method.

The following example shows how add index to collection. Let's assume that we have our same Employee collection which has the Field names of "Employeeid" and "EmployeeName".



```
db.Employee.createIndex({Employeeid:1})
```

Code Explanation:

1. The **createIndex** method is used to create an index based on the "Employeeid" of the document.
2. The '1' parameter indicates that when the index is created with the "Employeeid" Field values, they should be sorted in ascending order. Please note that this is different from the `_id` field (The id field is used to uniquely identify each document in the collection) which is created automatically in the collection by MongoDB. The documents will now be sorted as per the Employeeid and not the `_id` field.

If the command is executed successfully, the following Output will be shown:

Output:

Output:

```
C:\Windows\system32\cmd.exe - mongo.exe
> db.Employee.createIndex({Employeeid:1})
{"createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1}
```

1. The numIndexesBefore: 1 indicates the number of Field values (The actual fields in the collection) which were there in the indexes before the command was run. Remember that each collection has the _id field which also counts as a Field value to the index. Since the _id index field is part of the collection when it is initially created, the value of numIndexesBefore is 1.
2. The numIndexesAfter: 2 indicates the number of Field values which were there in the indexes after the command was run.
3. Here the "ok: 1" output specifies that the operation was successful, and the new index is added to the collection.

The above code shows how to create an index based on one field value, but one can also create an index based on multiple field values.

The following example shows how this can be done;

```
C:\Windows\system32\cmd.exe - mongo.exe
> db.Employee.createIndex({Employeeid:1, EmployeeName:1})
```

```
db.Employee.createIndex({Employeeid:1, EmployeeName:1})
```

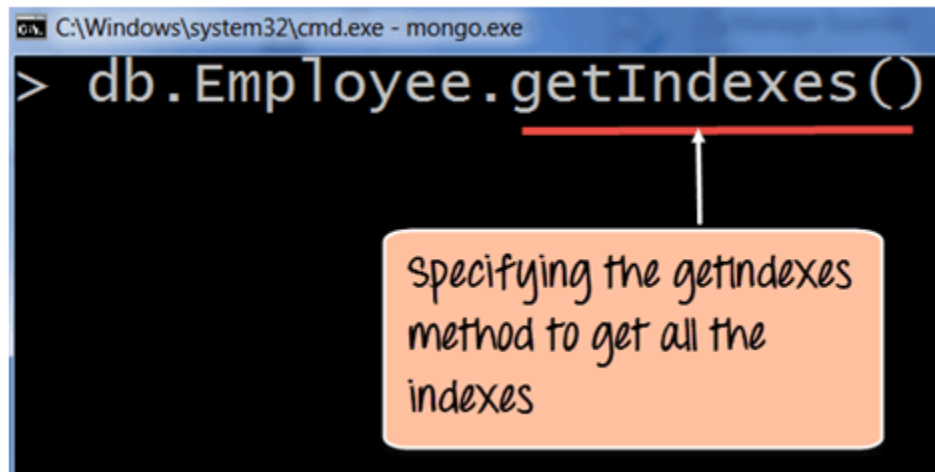
Code Explanation:

1. The createIndex method now takes into account multiple Field values which will now cause the index to be created based on the "Employeeid" and "EmployeeName". The Employeeid:1 and EmployeeName:1 indicates that the index should be created on these 2 field values with the :1 indicating that it should be in ascending order.

How to Find Indexes: getindexes()

Finding an Index in MongoDB is done by using the "getIndexes" method.

The following example shows how this can be done;



```
C:\Windows\system32\cmd.exe - mongo.exe
> db.Employee.getIndexes()
```

Specifying the getIndexes method to get all the indexes

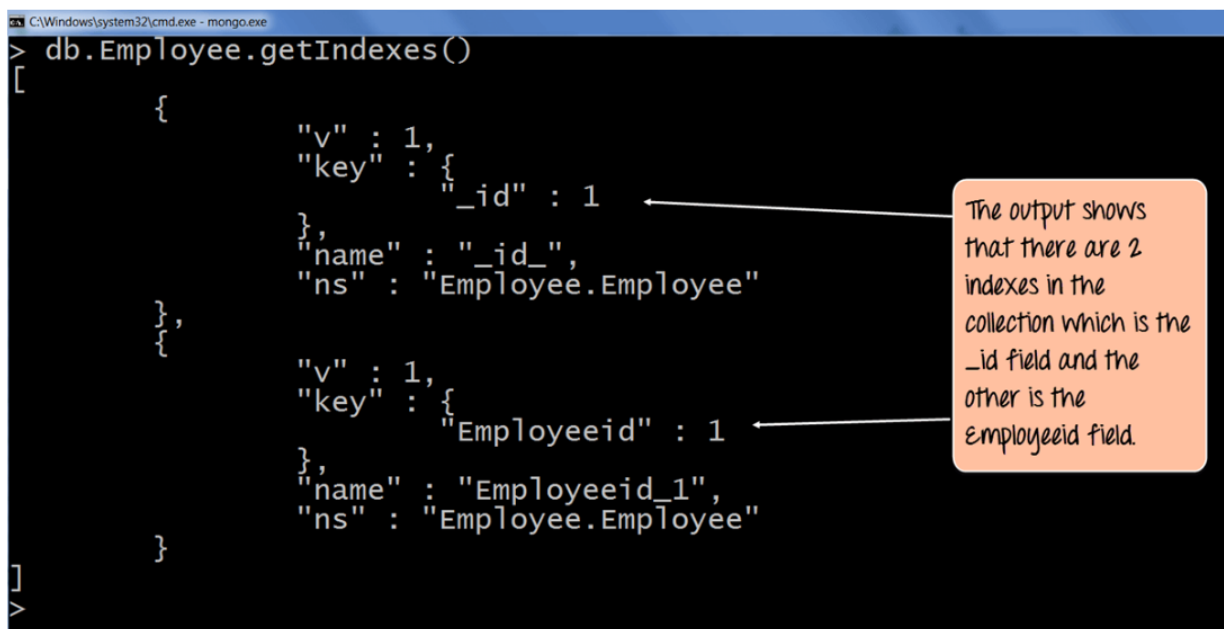
```
db.Employee.getIndexes()
```

Code Explanation:

1. The getIndexes method is used to find all of the indexes in a collection.

If the command is executed successfully, the following Output will be shown:

Output:



```
C:\Windows\system32\cmd.exe - mongo.exe
> db.Employee.getIndexes()
[
  {
    "v" : 1,
    "key" : { "_id" : 1 },
    "name" : "_id_",
    "ns" : "Employee.Employee"
  },
  {
    "v" : 1,
    "key" : { "Employeeid" : 1 },
    "name" : "Employeeid_1",
    "ns" : "Employee.Employee"
  }
]
```

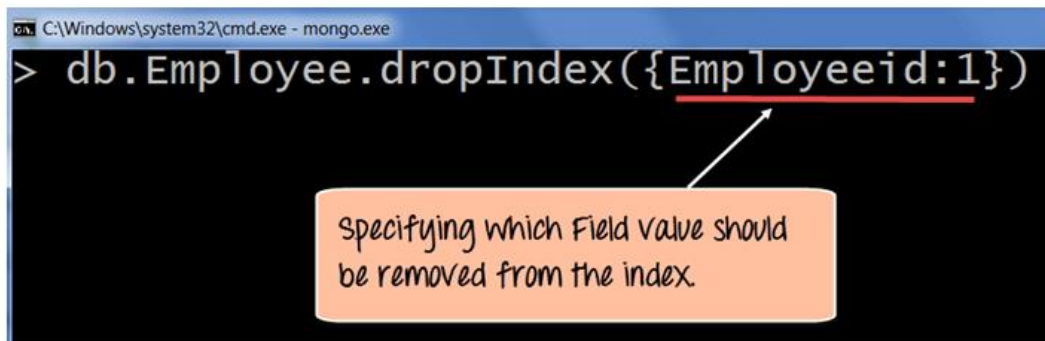
The output shows that there are 2 indexes in the collection which is the _id field and the other is the Employeeid field.

1. The output returns a document which just shows that there are 2 indexes in the collection which is the `_id` field, and the other is the Employee id field. The `:1` indicates that the field values in the index are created in ascending order.

How to Drop Indexes: `dropindex()`

Removing an Index in MongoDB is done by using the `dropIndex` method.

The following example shows how this can be done;



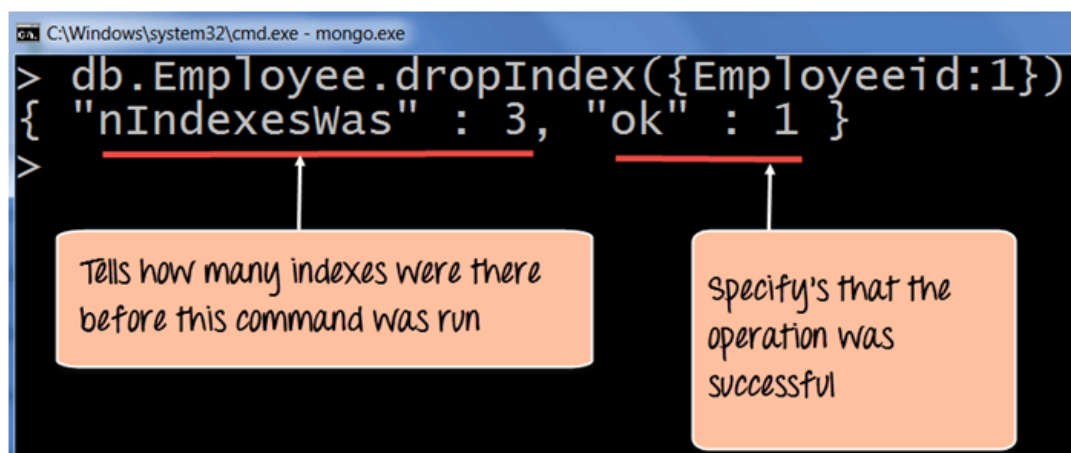
```
db.Employee.dropIndex(Employeeid:1)
```

Code Explanation:

1. The `dropIndex` method takes the required Field values which needs to be removed from the Index.

If the command is executed successfully, the following Output will be shown:

Output:

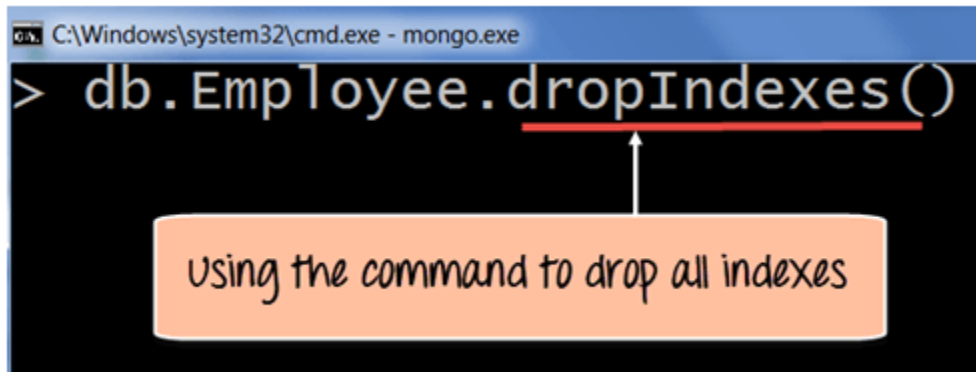


1. The `nIndexesWas: 3` indicates the number of Field values which were there in the indexes before the command was run. Remember that each collection has the `_id` field which also counts as a Field value to the index.

2. The ok: 1 output specifies that the operation was successful, and the "Employeeid" field is removed from the index.

To remove all of the indexes at once in the collection, one can use the dropIndexes command.

The following example shows how this can be done.



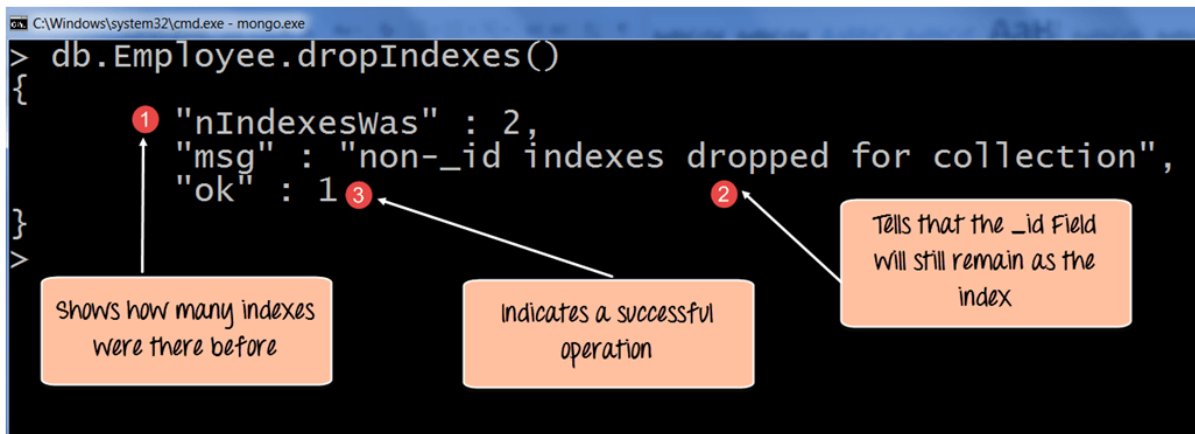
```
db.Employee.dropIndex()
```

Code Explanation:

1. The dropIndexes method will drop all of the indexes except for the _id index.

If the command is executed successfully, the following Output will be shown:

Output:



1. The nIndexesWas: 2 indicates the number of Field values which were there in the indexes before the command was run.
2. Remember again that each collection has the _id field which also counts as a Field value to the index, and that will not be removed by MongoDB and that is what this message indicates.
3. The ok: 1 output specifies that the operation was successful.

Summary

- Defining indexes are important for faster and efficient searching of documents in a collection.
- Indexes can be created by using the `createIndex` method. Indexes can be created on just one field or multiple field values.
- Indexes can be found by using the `getIndexes` method.
- Indexes can be removed by using the `dropIndex` for single indexes or `dropIndexes` for dropping all indexes.