Capped Collection in MongoDB

Capped collections are fixed-sized Circular Collection that follow the insert order to enhance the performance of create, read, and delete operations.

Capped collections are fixed-sized Circular Collection that follow the insert order to enhance the performance of create, read, and delete operations. With Circular, it means that when fixed sizes are allocated for Collection, it will start deleting the oldest Document in that Collection without providing any explicit commands.

Capped Collection limits update activities to the Document if those updates increase the size of the Document. When Capped Collection stores documents in the order of Disk Storage, it ensures that the document size does not increase more than the size allocated on Disk. Capped Collection is best for storing log information, cache data, .

Create Capped Collection in MongoDB

To create a **Capped Collection** , we use the commonly used createCollection command but with the **capped** option **true** and specify the maximum size (in bytes) for that Collection.

```
> db . createCollection ( "cappedLogCollection" ,{ capped : true , size : 10000
```

With the size of the Collection, we can also limit the Document number in that Collection by using the **max** parameter:

```
> db . createCollection ( "cappedLogCollection" ,{ capped : true , size : 10000
```

If you want to check if a Collection is capped or not, use the **isCapped** command:

```
> db . cappedLogCollection . isCapped ()
```

If there is an existing Collection that you intend to convert into capped, use the following code:

```
> db . runCommand ({ "convertToCapped" : "posts" , size : 10000 })
```

This code will convert existing **posts** collection into a Capped Collection.

Query on Capped Collection

By default, a find query on a Capped Collection will display the results in the insert order. But if you want to retrieve documents in reverse order, use the sort command as follows:

```
> db . cappedLogCollection . find (). sort ({ $natural :- 1 })
```

Here are some things to keep in mind while working with Capped Collection:

We cannot delete documents from a Capped Collection.

There is no default index present in a Capped Collection, even on the _id field.

While inserting a new Document, MongoDB doesn't really have to look at the location of the new Document on Disk. It can insert the new Document at the end of Collection. This makes the inserting of Capped Collection very fast.

Similarly, when reading documents, MongoDB only has to return documents in the same order as on Disk. This makes reading operations very fast.

According to Tutorialspoint

Previous post: GridFS in MongoDB