MONGODB - CAPPED COLLECTIONS

Capped collections are fixed-size circular collections that follow the insertion order to support high performance for create, read, and delete operations. By circular, it means that when the fixed size allocated to the collection is exhausted, it will start deleting the oldest document in the collection without providing any explicit commands.

Capped collections restrict updates to the documents if the update results in increased document size. Since capped collections store documents in the order of the disk storage, it ensures that the document size does not increase the size allocated on the disk. Capped collections are best for storing log information, cache data, or any other high volume data.

Creating Capped Collection

To create a capped collection, we use the normal createCollection command but with **capped** option as **true** and specifying the maximum size of collection in bytes.

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

In addition to collection size, we can also limit the number of documents in the collection using the **max** parameter –

```
>db.createCollection("cappedLogCollection",{capped:true,size:10000,max:1000})
```

If you want to check whether a collection is capped or not, use the following **isCapped** command –

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

If there is an existing collection which you are planning to convert to capped, you can do it with the following code –

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

This code would convert our existing collection **posts** to a capped collection.

Querying Capped Collection

By default, a find query on a capped collection will display results in insertion order. But if you want the documents to be retrieved in reverse order, use the **sort** command as shown in the following code –

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

There are few other important points regarding capped collections worth knowing –

- We cannot delete documents from a capped collection.
- There are no default indexes present in a capped collection, not even on _id field.

- While inserting a new document, MongoDB does not have to actually look for a place to accommodate new document on the disk. It can blindly insert the new document at the tail of the collection. This makes insert operations in capped collections very fast.
- Similarly, while reading documents MongoDB returns the documents in the same order as present on disk. This makes the read operation very fast.