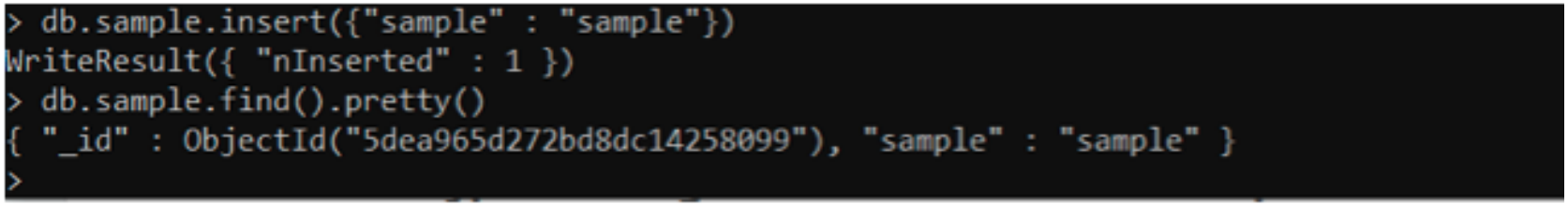
Understanding the mongoose __v field

Introduction

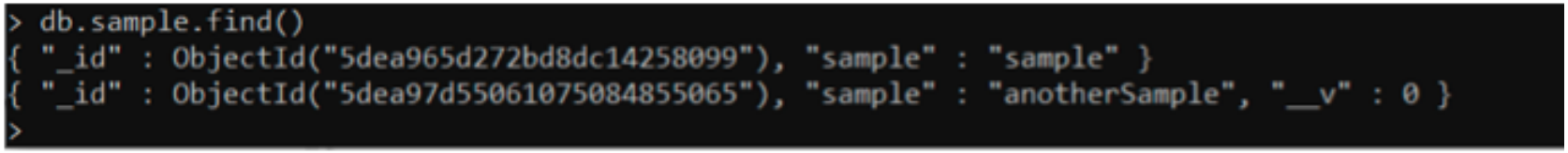
Data in MongoDB is stored in JSON format. More precisely said, data is stored in BSON format. BSON format provides a variety of data types. We can insert data from mongo shell or through mongoose. In both cases, the _id field is generated automatically in each document. The _id field acts as a primary key. It is unique throughout the collection. But there is one difference when inserting document(s) through mongoose. It auto-generates one more field. This field is the __v field. In this article, we will discuss what this __v field is and when it gets created.

__v field

This is a document inserted through the mongo shell in a collection.



Only one field is inserted but it automatically generates an _id field. Let's insert another document, but this time through mongoose.



The second document also contains an auto-generated _id field. But it also contains one extra field. This is __v field that is only generated when a document(s) is inserted through mongoose.

The __v field is called the version key. It describes the internal revision of a document. This __v field is used to track the revisions of a document. By default, its value is zero. In real practice, the __v field increments by one only when an array is updated. In other situations, the value of the __v field remains unaffected. So to keep the track of __v field in such situations, we can do it manually using the increment operator provided by the mongoose.

Conclusion

We explained the __v field and showed how the field is added when you use mongoose as your ORM. If you're unfamiliar with mongoose, it is an ORM or Object Relational Mapper used with MongoDb. It is extremely popular and can make your database queries much more manageable. Please check out their documentation if you're interested in using mongoose.

We also showed a few screenshots of how the documents are different when inserting through MongoDB compared to mongoose. When using MongoDb we used the `insert` function and passed in the document that we wanted to insert. Since we only provided a screenshot before we'll paste those command snippets here in case you need it. The first command does the insert. The second command uses `find` to retrieve the document that was created (pretty formatted).

```
1 > db.sample.insert({"sample" : "sample"})
2 > db.sample.find().pretty()
```