Exercise 1 - Insert documents

- Let's insert 200k documents into the newly created collection.
- This should take a few seconds to complete.
- The code given below will insert these documents into the `bigdata` collection.
- Each document has a field named `account_no` which is assigned to incrementing variable `i`.
- Another field `balance` contains a randomly generated number, to simulate the bank balance for the account.

Copy the below code and paste it on the mongo client.

list

start

upper boundary

index similarly to python

key

key

```
docsToInsert = []
for (i = 1; i <= 200000; i++) { docsToInsert[i-1] = { "account_no": i, "balance": Math.round(Math.random() * 20000) } }
db.bigdata.insertMany(docsToInsert);
```

from 0,1,2,...,199999 in list

insert all

Copied!

Verify that `200000` documents got inserted by running the below command.

```
db.bigdata.countDocuments()
```

Copied!

Exercise 2 - Measure the time taken by a query

Let's run a query and find out how much time it takes to complete. You will query for the details of account number 58982.

We will make use of the `explain` function to find the time taken to run the query in milliseconds.

> The db.collection.explain("executionStats") method provides statistics about the performance of a query. These statistics can be useful in measuring if and how a query uses an index. See db.collection.explain() for details.

Run the below command.

collection

filter

measure

return detailed statistics about the execution

measure total execution time in ms

```
db.bigdata.find({"account_no":58982}).explain("executionStats").executionStats.executionTimeMillis
```

specify to return the detailed statistics

specify to get time in milli in detailed statistic

Copied!

Make a note of the milliseconds it took to run the query. We will need this at a later stage.

→ 136 ms

Exercise 3 - Working with indexes

Before you create an index, choose the field you wish to create an index on. It is usually the field that you query most.

Run the below command to create an index on the field `account_no`.

```
db.bigdata.createIndex({"account_no":1})
```

Copied!

Where `1` means `ascending` order.

Run the below command to get a list of indexes on the `bigdata` collection.

```
db.bigdata.getIndexes()
```

Copied!

You should see an index named `account_no_1`

Exercise 4 - Find out how effective an index is

You will now run the same query for account `58982` and compare the execution time from previous run. This way, you can compare the improvement.

Run the below command.

```
db.bigdata.find({"account_no":58982}).explain("executionStats").executionStats.executionTimeMillis
```

Copied!

Yep

This time, the execution time should be a lot less than previously. If you see `0`, it means the query completed under 1 millisecond.

Exercise 6 - Delete an index

Use the below command to delete the index we created earlier. Here you can provide index definition or name.

```
db.bigdata.dropIndex({"account_no":1})
```

Copied!

Bonus information

MongoDB creates a unique index on the `_id` field during the creation of a collection. The `_id` index prevents clients from inserting two documents with the same value for the `_id` field. You cannot drop this index on the `_id` field.

Practice exercises

1. Problem:

   Create an index on the `balance` field.

▶ Click here for Hint
▶ Click here for Solution

2. Problem:

   Query for documents with a balance of `10000` and record the time taken.

▶ Click here for Hint
▶ Click here for Solution

3. Problem:

   Drop the index you have created.

▶ Click here for Hint
▶ Click here for Solution

4. Problem:

   Query for documents with a balance of 10000 and record the time taken, and compare it with the previously recorded time.

▶ Click here for Hint
▶ Click here for Solution

Summary

In this lab, you have gained an understanding of indexing in MongoDB.

Author(s)

Muhammad Yahya