Work Integrated Learning Programmes

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Computer Science & Information Systems

Big Data Systems – MongoDB Lab Sheet:3

Data Querying

1. Objective

Students should be able to

A. Query the documents from a collection
B. Use queries involving relational operators
C. Use logical operators in the query

Note: This lab assumes that you already have 5 books documents added into the "books" collection. If that is not the case then you can insert those documents using the following command.

```
> db.books.insert([
{"id":1, "title":"Cloud computing", "tags":["cloud", "prog"],
"reviews":5},
{"id":2, "title":"IDS", "tags":["DataScience"], "reviews":3},
{"id":3, "title":"Data Analytics", "reviews":5},
{"id":4, "title":"Big Data Systems", "reviews":5},
{"id":5, "title":"Big Data", "reviews":3}
])
```

2. Steps to be performed

A. Query the documents from a collection

A1. The "find" command

To query data from MongoDB collection, you need to use MongoDB's **find()**method.

Syntax

The basic syntax of **find()** method is as follows –

```
> db.collection_name.find()
```

find() method will display all the documents in a non-structured way.

```
> db.books.find()

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"), "id" : 1, "title" :
"Cloud computing", "tags" : [ "cloud", "prog" ], "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "id" : 2, "title" :
"IDS", "tags" : [ "DataScience" ], "reviews" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"), "id" : 3, "title" :
"Data Analytics", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"), "id" : 4, "title" :
"Big Data Systems", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"), "id" : 5, "title" :
"Big Data", "reviews" : 3 }
```

A2. The “pretty” command

The pretty() Method

To display the results in a formatted way, you can use **pretty()** method.

Syntax

```
> db.books.find().pretty()

{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"),
        "id" : 1,
        "title" : "Cloud computing",
        "tags" : [
                "cloud",
                "prog"
        ],
        "reviews" : 5
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"),
        "id" : 2,
        "title" : "IDS",
        "tags" : [
                "DataScience"
        ],
        "reviews" : 3
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"),
        "id" : 3,
```

```
        "title" : "Data Analytics",
        "reviews" : 5
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"),
        "id" : 4,
        "title" : "Big Data Systems",
        "reviews" : 5
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"),
        "id" : 5,
        "title" : "Big Data",
        "reviews" : 3
}
```

A3. **Projecting an attribute from the document**

In MongoDB, projection means selecting only the necessary data rather than selecting whole of the data of a document. If a document has 5 fields and you need to show only 3, then select only 3 fields from them.

The find() Method

MongoDB's **find()** method, explained earlier accepts second optional parameter that is list of fields that you want to retrieve. In MongoDB, when you execute **find()** method, then it displays all fields of a document. To limit this, you need to set a list of fields with value 1 or 0. 1 is used to show the field while 0 is used to hide the fields.

Syntax

The basic syntax of **find()** method with projection is as follows –

```
> db.COLLECTION_NAME.find({},{KEY:1})
```

Display the key “id” from “books” collection

```
> db.books.find({}, {"id":1})

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"), "id" : 1 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "id" : 2 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"), "id" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"), "id" : 4 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"), "id" : 5 }
```

Display the keys “title” and “reviews” from the “books” collection

```
> db.books.find({}, {"title":1, "reviews":1})

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"), "title" : "Cloud
computing", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "title" : "IDS",
"reviews" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"), "title" : "Data
Analytics", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"), "title" : "Big Data
Systems", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"), "title" : "Big
Data", "reviews" : 3 }
```

A4. The “limit” option in “find” command

In MongoDB to limit the records in MongoDB, you need to use **limit()** method. The method accepts one number type argument, which is the number of documents that you want to be displayed.

Syntax

The basic syntax of **limit()** method is as follows –

```
> db.COLLECTION_NAME.find().limit(NUMBER)
```

Use the following command to list first three documents of “books” collection

```
> db.books.find().limit(3)

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"), "id" : 1, "title" :
"Cloud computing", "tags" : [ "cloud", "prog" ], "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "id" : 2, "title" :
"IDS", "tags" : [ "DataScience" ], "reviews" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"), "id" : 3, "title" :
"Data Analytics", "reviews" : 5 }
```

A5. The "sort" option in "find" command

To sort documents in MongoDB, you need to use **sort()** method. The method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

Syntax

The basic syntax of **sort()** method is as follows –

```
> db.COLLECTION_NAME.find().sort({KEY:1})
```

To sort the documents based on the "reviews" in ascending order use the following command

```
> db.books.find().sort({"reviews":1})

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "id" : 2, "title" :
"IDS", "tags" : [ "DataScience" ], "reviews" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"), "id" : 5, "title" :
"Big Data", "reviews" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"), "id" : 1, "title" :
"Cloud computing", "tags" : [ "cloud", "prog" ], "reviews" : 5 }
{ " id" : ObjectId("5dc239aaa9c1f4f88abc9d53"), "id" : 3, "title" :
"Data Analytics", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"), "id" : 4, "title" :
"Big Data Systems", "reviews" : 5 }
```

To sort the documents based on the "reviews" in descending order use command

```
> db.books.find().sort({"reviews":-1})

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"), "id" : 1, "title" :
"Cloud computing", "tags" : [ "cloud", "prog" ], "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"), "id" : 3, "title" :
"Data Analytics", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"), "id" : 4, "title" :
"Big Data Systems", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "id" : 2, "title" :
"IDS", "tags" : [ "DataScience" ], "reviews" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"), "id" : 5, "title" :
"Big Data", "reviews" : 3 }
```

B. Use queries involving relational operators

B1. Relational operators

To query the document on the basis of some condition, you can use following operations.

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db. books.find({"title":"IDS"}).pretty()	where author = 'myself'
Less Than	{<key>:{$lt:<value>}}	db. books.find({"reviews":{$lt:4}}).pretty()	where reviews < 4
Less Than Equals	{<key>:{$lte:<value>}}	db. books.find({"reviews":{$lte:3}}).pretty()	where reviews <= 3
Greater Than	{<key>:{$gt:<value>}}	db. books.find({"reviews":{$gt:4}}).pretty()	where reviews > 4
Greater Than Equals	{<key>:{$gte:<value>}}	db. books.find({"reviews":{$gte:5}}).pretty()	where reviews >= 5
Not Equals	{<key>:{$ne:<value>}}	db. books.find({"reviews":{$ne:5}}).pretty()	where reviews!= 5

B2. Examples

To find the books having title as "IDS"

```
> db.books.find({"title":"IDS"}).pretty()
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"),
        "id" : 2,
        "title" : "IDS",
        "tags" : [
                "DataScience"
        ],
        "reviews" : 3
}
```

To find all the books having reviews count less than 4

```
db. books.find({"reviews":{$lt:4}}).pretty()

{
        " id" : ObjectId("5dc239aaa9c1f4f88abc9d52"),
        "id" : 2,
        "title" : "IDS",
        "tags" : [
                "DataScience"
        ],
        "reviews" : 3
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"),
        "id" : 5,
        "title" : "Big Data",
        "reviews" : 3
}
```

To find all the books having reviews count less than or equal to 3

```
db. books.find({"reviews":{$lte:3}}).pretty()
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"),
        "id" : 2,
        "title" : "IDS",
        "tags" : [
                "DataScience"
        ],
        "reviews" : 3
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"),
        "id" : 5,
        "title" : "Big Data",
        "reviews" : 3
}
```

To find all the books having reviews count greater than 4

```
db. books.find({"reviews":{$gt:4}}).pretty()
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"),
        "id" : 1,
        "title" : "Cloud computing",
        "tags" : [
                "cloud",
                "prog"
        ],
        "reviews" : 5
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"),
        "id" : 3,
        "title" : "Data Analytics",
        "reviews" : 5
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"),
        "id" : 4,
        "title" : "Big Data Systems",
        "reviews" : 5
}
```

To find all the books having reviews count greater than or equal to 5

```
db. books.find({"reviews":{$gte:5}}).pretty()
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"),
        "id" : 1,
        "title" : "Cloud computing",
        "tags" : [
                "cloud",
                "prog"
        ],
        "reviews" : 5
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"),
        "id" : 3,
        "title" : "Data Analytics",
        "reviews" : 5
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"),
        "id" : 4,
        "title" : "Big Data Systems",
        "reviews" : 5
}
```

To find all the books having reviews count not equal to 5

```
db.books.find({"reviews":{$ne:5}}).pretty()
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"),
        "id" : 2,
        "title" : "IDS",
        "tags" : [
                "DataScience"
        ],
        "reviews" : 3
}
{
        "_id" : ObjectId("5dc239aaa9c1f4f88abc9d55"),
        "id" : 5,
        "title" : "Big Data",
        "reviews" : 3
}
```

C. Use logical operators in the query

C1. Logical "AND" operators

Syntax

In the **find()** method, if you pass multiple keys by separating them by ',' then MongoDB treats it as **AND** condition. Following is the basic syntax of **AND** –

```
> db.COLLECTION_NAME.find(
{
    $and: [
              {key1: value1}, {key2:value2}
          ]
}
).pretty()
```

Following example will show all the books named "IDS" and "reviews" count as 3.

```
> db.books.find({$and: [{"title":"IDS"}, {"reviews":3}]})

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "id" : 2, "title" :
"IDS", "tags" : [ "DataScience" ], "reviews" : 3 }
```

C2. Logical "OR" operator

Syntax

To query documents based on the OR condition, you need to use **$or** keyword. Following is the basic syntax of **OR** –

```
> db.COLLECTION_NAME.find(
{
    $or: [
              {key1: value1}, {key2:value2}
          ]
}
).pretty()
```

Following example will show all the books named “IDS” and “reviews” count as 3.

```
> db.books.find({$or: [{"title":"IDS"}, {"reviews":5}]})

{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d51"), "id" : 1, "title" :
"Cloud computing", "tags" : [ "cloud", "prog" ], "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d52"), "id" : 2, "title" :
"IDS", "tags" : [ "DataScience" ], "reviews" : 3 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d53"), "id" : 3, "title" :
"Data Analytics", "reviews" : 5 }
{ "_id" : ObjectId("5dc239aaa9c1f4f88abc9d54"), "id" : 4, "title" :
"Big Data Systems", "reviews" : 5 }
```

3. Outputs/Results

Students should be able to appreciate the usage of commands to

- Query the documents using various options available
- Query the documents using relational and logical operators

4. Observations

Students should carefully observe the syntax of retrieval queries and their output.

5. References

- MongoDB documentation