

Map-Reduce

MapReduce is the data processing mechanism for condensing large amount of data into useful aggregated results. This task is executed by **MapReduce** command which subsequently perform **map** and **reduce** operations. **MapReduce** is used for processing large data sets. In straightforward terms, the **MapReduce** command takes two primary inputs, the mapper function, and the reducer function.

The **Mapper** function reads the collection of data and build the **Map** with the required fields that which are required to process and group them into one array. Subsequently, this **key-value** pair is fed into the **Reducer**, which further transform the values. The two phases of Map/Reduce is:

1. **map** phase: filter / transform / convert data
2. **reduce** phase: perform aggregations over the data

Syntax

```
db.collectionName.mapReduce(  
  function() {emit(key,value);}, //here is the map operation  
  function(key,values) {return reduceFunction}, { //here is the reduce operation  
    out: collection,  
    query: document,  
    sort: document,  
    limit: number  
  }  
)
```

Where,

- **map** function maps a value with a key and emits a key-value pair. It's a JavaScript function
- **reduce** function reduces or groups all the documents having the same key. It's a JavaScript function
- **out** specifies the location of the map-reduce query result
- **query** specifies the optional selection criteria for selecting documents
- **sort** specifies the optional sort criteria i.e. ascending or descending
- **limit** specifies the optional maximum number of documents to be returned

The map-reduce function first queries the collection, then maps the result documents to emit key-value pairs, which is then reduced based on the keys that have multiple values.

Example:

Lets have the marks collection having the student name, roll, no, class and subject marks in each document.

```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
> db.marks.find()
{ "_id" : ObjectId("5ec103443b6e4f8f5b4f1148"), "name" : "rohit", "class" : "9th", "rollno" : 3, "sub" : "computer", "marks" : 48 }
{ "_id" : ObjectId("5ec103583b6e4f8f5b4f1149"), "name" : "rohit", "class" : "9th", "rollno" : 3, "sub" : "english", "marks" : 44 }
{ "_id" : ObjectId("5ec1036c3b6e4f8f5b4f114a"), "name" : "rohit", "class" : "9th", "rollno" : 3, "sub" : "hindi", "marks" : 41 }
{ "_id" : ObjectId("5ec103913b6e4f8f5b4f114b"), "name" : "suman", "class" : "9th", "rollno" : 2, "sub" : "computer", "marks" : 41 }
{ "_id" : ObjectId("5ec103a33b6e4f8f5b4f114c"), "name" : "suman", "class" : "9th", "rollno" : 2, "sub" : "english", "marks" : 43 }
{ "_id" : ObjectId("5ec103b53b6e4f8f5b4f114d"), "name" : "suman", "class" : "9th", "rollno" : 2, "sub" : "hindi", "marks" : 43 }
{ "_id" : ObjectId("5ec103cf3b6e4f8f5b4f114e"), "name" : "ajay", "class" : "10th", "rollno" : 8, "sub" : "hindi", "marks" : 45 }
{ "_id" : ObjectId("5ec103e73b6e4f8f5b4f114f"), "name" : "ajay", "class" : "10th", "rollno" : 8, "sub" : "english", "marks" : 39 }
{ "_id" : ObjectId("5ec103f83b6e4f8f5b4f1150"), "name" : "ajay", "class" : "10th", "rollno" : 8, "sub" : "computer", "marks" : 44 }
{ "_id" : ObjectId("5ec1041b3b6e4f8f5b4f1151"), "name" : "manoj", "class" : "10th", "rollno" : 9, "sub" : "hindi", "marks" : 44 }
{ "_id" : ObjectId("5ec104283b6e4f8f5b4f1152"), "name" : "manoj", "class" : "10th", "rollno" : 9, "sub" : "computer", "marks" : 49 }
{ "_id" : ObjectId("5ec104373b6e4f8f5b4f1153"), "name" : "manoj", "class" : "10th", "rollno" : 9, "sub" : "english", "marks" : 40 }
```

1. Now, we will use a mapReduce function on **marks** collection to select all sum of marks of all the students in class **9th**. The candidates of the class will be grouped based on the **"name"** and then the count the **marks** of each student. The output of the operation will be saved as collection **"result_9th"**.

The code will be

```
>db.marks.mapReduce(
  function() { emit(this.name,this.marks); },
  function(key, values) {return Array.sum(values)}, {
    query:{class:"9th"},
    out:"result_9th"
  }
)
```

As a result a new collection "result_9th" will be created and the same can be seen using show collections command. The documents in the newly created collection may be seen using find() command.

```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
> db.marks.mapReduce( function() { emit(this.name, this.marks); }, function(key, values) {return Array.sum(values)},
{ query:{class:"9th"}, out:"result_9th" } )
{
  "result" : "result_9th",
  "timeMillis" : 343,
  "counts" : {
    "input" : 6,
    "emit" : 6,
    "reduce" : 2,
    "output" : 2
  },
  "ok" : 1
}
```

The result of the operation displays that

- Total 6 documents matched the query (class:"9th"),
- The map function emitted 6 documents with key-value pairs and
- Finally the reduce function grouped mapped documents having the same keys into 2 documents.

We may list the collections, and also see the documents in the collection :

```
>
> show collections
abc
college
marks
old
result
result_9th
result_totals
school
stock
store
>
> db.result_9th.find()
{ "_id" : "rohit", "value" : 133 }
{ "_id" : "suman", "value" : 127 }
```

2. Lets use mapReduce function on **marks** collection again to find sum of marks of all the students in class **10th**. The candidates of the class will be grouped based on the **rollno** and then the count their marks of each student. The output of the operation will be saved as collection "**result_10th**".

```
>db.marks.mapReduce(
  function() { emit(this.rollno, this.marks); },
  function(key, values) {return Array.sum(values)}, {
    query:{class:"10th"},
    out:"result_10th"
  }
)
```

```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
> db.marks.mapReduce(
...   function() { emit(this.rollno, this.marks); },
...   function(key, values) {return Array.sum(values)}, {
...     query:{class:"10th"},
...     out:"result_10th"
...   }
... )
{
  "result" : "result_10th",
  "timeMillis" : 551,
  "counts" : {
    "input" : 6,
    "emit" : 6,
    "reduce" : 2,
    "output" : 2
  },
  "ok" : 1
}
> db.result_10th.find()
{ "_id" : 8, "value" : 128 }
{ "_id" : 9, "value" : 133 }
```

Note:

- In above examples, the resulted documents are saved in the collection and we have to run find() command to show them.
- We can use find () command with the mapReducer command to display the resulted documents after the operation. The command for the this to find the result of 10th class is

```
>db.marks.mapReduce(
    function() { emit(this.rollno, this.marks); },
    function(key, values) {return Array.sum(values)}, {
        query:{class:"10th"},
        out:"result_10th"
    }
).find()
```

```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
> db.marks.mapReduce(
...   function() { emit(this.rollno, this.marks); },
...   function(key, values) {return Array.sum(values)}, {
...     query:{class:"10th"},
...     out:"result_10th"
...   }
... ).find()
{ "_id" : 8, "value" : 128 }
{ "_id" : 9, "value" : 133 }
```

Assignments:

1. What is mapReduce? Explain the syntax.
2. Explain Map and Reduce operations in mapReduce with examples.