Aggregation and MapReduce in MongoDB

Prashanth B S[1]

[1]Department of Information Science & Engineering, Nitte Meenakshi Institute of Technology, Yelahanka - 560064, Bengaluru

April 20, 2022

1 Aggregation

Aggregation refers to the process of grouping values from multiple documents and perform wide range of operations on it. The concept is similar to the aggregate functions in SQL. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In order to apply aggregation, MongoDB provides *aggregage*() method. The general syntax for the *aggregage*() method is as shown below,

```
> db.col_name.aggregate(AGGREGATE_OPERATION)
```

Aggregate function usually operated with $\$group$ to group the values first and then apply operations on it such as $\$sum$, $\$avg$,$\$max$, $\$min$, $\$first$, and $\$last$. An example is as shown below,

```
> db.mycol.aggregate([{$group : {_id : "$by_which_column", aggregated_column_name :
    {$operation : "$column"}}}])
// SELECT by_which_column, count(*) FROM mycol GROUPBY column ;
// _id : Mandatory key value
// $operation = $max, $min, $first, $last, $avg, $sum
```

In order to work with various Aggregate operations, we will use the dataset shown below,

```
// Database: BookDB
// Collection: Book
{ "_id" : 1, "Author" : { "initials" : "Dan", "name" : "Brown" }, "BID" : "AE001", "tags"
    : [ "Fiction", "Action", "Adventure", "Scifi" ], "Price" : 180 }
{ "_id" : 2, "Author" : { "initials" : "Dan", "name" : "Brown" }, "BID" : "AE002", "tags"
    : [ "Fiction", "Action", "Adventure", "Scifi" ], "Price" : 180 }
{ "_id" : 3, "Author" : { "initials" : "John", "name" : "Grisham" }, "BID" : "AE003",
    "tags" : [ "Fiction", "Action", "Adventure", "Scifi" ], "Price" : 180 }
{ "_id" : 4, "Author" : { "initials" : "John", "name" : "Grisham" }, "BID" : "AE004",
    "tags" : [ "Fiction", "Action", "Adventure", "Scifi" ], "Price" : 180 }
```

Suppose if we want to group the Book by the Author name and count How many number of books authored by each Author, The following is the aggregate function to achieve it,

```
> db.Book.aggregate([{$group : {_id : "$Author.name", num_tutorial : {$sum : 1}}}])
{ "_id" : "Brown", "num_tutorial" : 2 }
{ "_id" : "Grisham", "num_tutorial" : 2 }
```

Suppose if we want to find the overall amount and count for all the books grouped based on Author initials, The following is the aggregate function to achieve it,

```
> db.Book.aggregate([{$group : {_id : "$Author.initials", total_cost : {$sum :
    "$Price"},count:{$sum : 1}}}])
{ "_id" : "John", "total_cost" : 360, "count" : 2 }
{ "_id" : "Dan", "total_cost" : 360, "count" : 2 }
```

The following segment discusses the usecases using the flags such as $\$avg$,$\$max$, and $\$last$

```
// Aggregate function to compute avg price grouped by Author name
> db.Book.aggregate([{$group : {_id : "$Author.name", avg_price : {$avg : "$Price"}}}])
{ "_id" : "Brown", "avg_price" : 180 }
{ "_id" : "Grisham", "avg_price" : 180 }

// To perform aggregate function say finding the max Price of the book published grouped
    by Author.name
> db.Book.aggregate([{$group : {_id : "$Author.name", Max_Price : {$max : "$Price"}}}])
{ "_id" : "Brown", "Max_Price" : 180 }
{ "_id" : "Grisham", "Max_Price" : 180 }

// To perform aggregate function finding the last document grouped by Author.name
> db.Book.aggregate([{$group : {_id : "$Author.name", Max_Price : {$last : "$Price"}}}])
{ "_id" : "Grisham", "Max_Price" : 180 }
{ "_id" : "Brown", "Max_Price" : 180 }
```

On the Similar lines $\$min$, $\$sum$ and $\$first$ can be explored.

2 Map-reduce in MongoDB

Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. 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. The following are the functions which are involved in the process[1],

- *map* is a javascript function that maps a value with a key and emits a key-value pair
- *reduce* is a javascript function that reduces or groups all the documents based on the same key
- *out* specifies the location of the map-reduce query result
- *query* specifies the optional selection criteria for selecting documents

To demonstrate the working of MapReduce, we will have a dataset as shown below,

```
// Database: MapReduceDB, Collection: MR
{ "_id" : ObjectId("625fe312ed3d662f49bf6b45"), "name" : "ABC", "age" : 10 }
{ "_id" : ObjectId("625fe321ed3d662f49bf6b46"), "name" : "ABC", "age" : 20 }
{ "_id" : ObjectId("625fe32ded3d662f49bf6b47"), "name" : "LMN", "age" : 20 }
{ "_id" : ObjectId("625fe336ed3d662f49bf6b48"), "name" : "LMN", "age" : 30 }
```

The steps are as follows,

1. Create a Mapper function which takes in a value and emits a key,value pair

   ```
   > var mapc = function(){emit(this.name, this.age)}
   ```

[1] https://www.tutorialspoint.com/mongodb/mongodb_map_reduce.htm

2. Create a Reducer function which operates on emitted value and performs aggregate operations on it.

```
> var redc = function(key, value){return Array.sum(value)}
```

3. Pass in the mapper and reducer function to MongoDB *mapReduce*() method whose general syntax is as follows,

```
// General Syntax: mapReduce(mapper_fun, reducer_fun, {out:"Result Object"})
// For the above example,
> db.MR.mapReduce(mapc, redc,{out:"myresult"}) ;
{ "result" : "myresult", "ok" : 1 }

// Displaying the documents based on the object myresult
> db.myresult.find().pretty()
{ "_id" : "ABC", "value" : 30 }
{ "_id" : "LMN", "value" : 50 }
```