MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part B-03**

GROUP: B) ASSIGNMENT NO: 03

AIM: MongoDB – Map-reduces operations:

Implement Map reduces operation with suitable example using MongoDB.

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To develop basic Database administration skill.
- To understand concepts of map reduce in aggregation of NoSQL database MongoDB.

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative.
- Front End: Java/PHP/Python.
- Back End: MongoDB.

THEORY:

1. **MongoDB Aggregation**
   - Aggregations are operations that process data records and return computed results.
   - MongoDB provides a rich set of aggregation operations that examine and perform calculations on the data sets.
   - Running data aggregation on the mongod instance simplifies application code and limits resource requirements.
   - Like queries, aggregation operations in MongoDB use collections of documents as an input and return results in the form of one or more documents.
2. **Map-Reduce**
   - Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results.
   - For map-reduce operations, MongoDB provides the mapReduce database command.

Syntax: db.COLLECTION_NAME.mapReduce()

- In general, map-reduce operations have two phases: a map stage that processes each document and emits one or more objects for each input document, and reduce phase that combines the output of the map operation.
- Optionally, map-reduce can have a finalize stage to make final modifications to the result. Like other aggregation operations, map-reduce can specify a query condition to select the input documents as well as sort and limit the results.
- All map-reduce functions in MongoDB are JavaScript and run within the mongod process. Map-reduce operations take the documents of a single collection as the input and can perform any arbitrary sorting and limiting before beginning the map stage.
- MapReduce can return the results of a map-reduce operation as a document, or may write the results to collections. The input and the output collections may be sharded.

3. MongoDB and MapReduce

In MapReduce we use mapreduce, map, and reduce keys. These three keys are required, but there are many optional keys that can be passed to the MapReduce command:

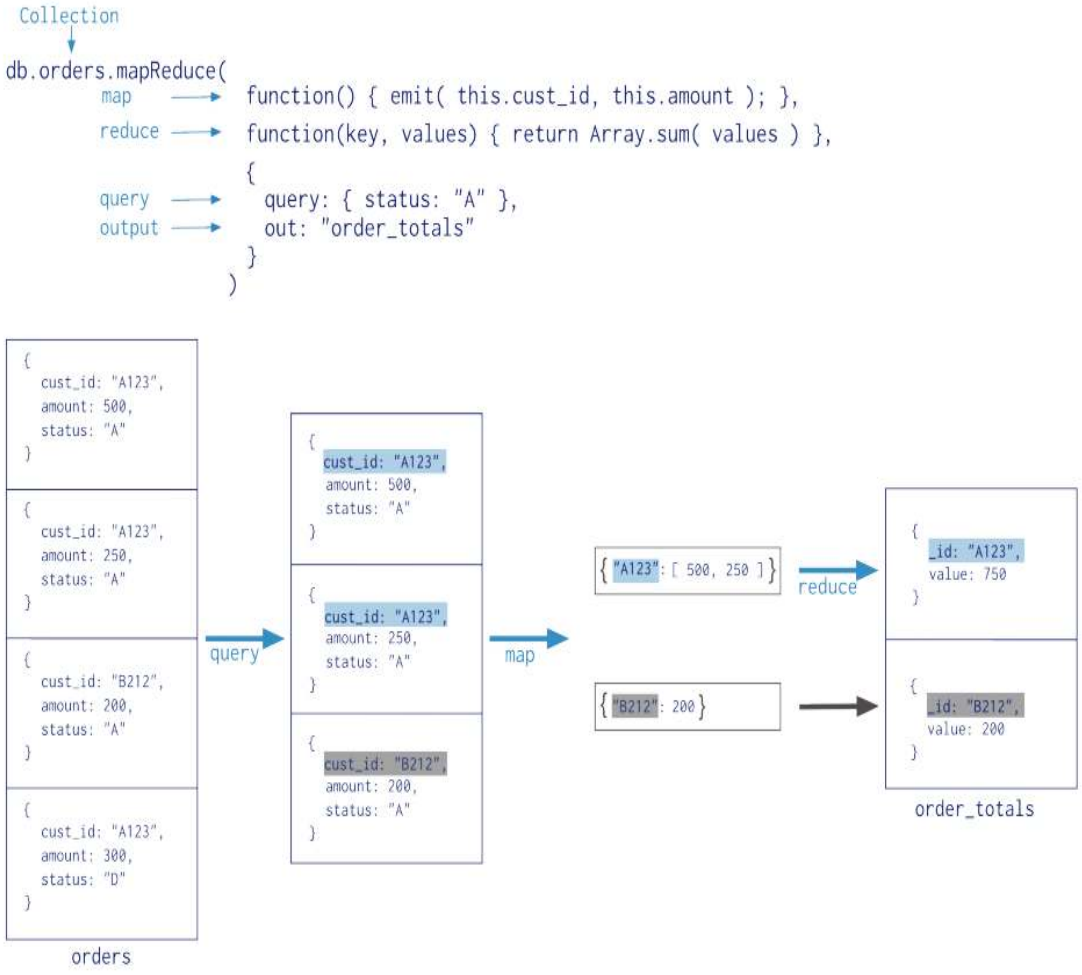
- "*finalize" : function*
  - ✓ A final step to send reduce's output to.
- "*keeptemp" : boolean*
  - ✓ If the temporary result collection should be saved when the connection is closed.
- *"out" : string*
  - ✓ Name for the output collection. Setting this option implies keep temp : true.
- *"query" : document*
  - ✓ Query to filter documents by before sending to the map function.
- *"sort" : document*
  - ✓ Sort to use on documents before sending to the map (useful in conjunction with the limit option).
- *"limit" : integer*
  - ✓ Maximum number of documents to send to the map function.
- *"scope" : document*
  - ✓ Variables that can be used in any of the JavaScript code.
- *"verbose" : boolean*
  - ✓ Whether or not to use more verbose output in the server logs.

4. Aggregation Commands

- **Aggregate:** Performs aggregation tasks such as group using the aggregation framework.

- **Count:** Counts the number of documents in a collection.
- **Distinct:** Displays the distinct values found for a specified key in a collection.
- **Group:** Groups documents in a collection by the specified key and performs simple aggregation.
- **mapReduce:** Performs map-reduce aggregation for large data sets.

5. **Aggregation Methods**
   - **db.collection.aggregate():** Provides access to the aggregation pipeline.
   - **db.collection.group():** Groups documents in a collection by the specified key and performs simple aggregation.
   - **db.collection.mapReduce():** Performs map-reduce aggregation for large data sets.



6. **Example for MapReduce Operation**

   In the mongo shell, the **db.collection.mapReduce()** method is a wrapper around the mapReduce command.

   The following examples use the **db.collection.mapReduce() method**:

   Consider the following map-reduce operations on a collection orders that contains documents of the following prototype:

```
{
        _id: ObjectId("50a8240b927d5d8b5891743c"),
        cust_id: "abc123",
        ord_date: new Date("Oct 04, 2012"),
        status: 'A',
        price: 25,
        items: [ { sku: "mmm", qty: 5, price: 2.5 },
        { sku: "nnn", qty: 5, price: 2.5 } ]
}
```

Return the Total Price Per Customer

Perform the map-reduce operation on the orders collection to group by the cust_id, and calculate the sum of the price for each cust_id:

a) **Define the map function to process each input document:**
   - In the function, this refers to the document that the map-reduce operation is processing.
   - The function maps the price to the cust_id for each document and emits the cust_id and price pair.

```
var mapFunction1 = function() {
                        emit(this.cust_id, this.price);
                        };
```

b) **Define the corresponding reduce function with two arguments keyCustId and valuesPrices:**
   - The valuesPrices is an array whose elements are the price values emitted by the map function and grouped by keyCustId.
   - The function reduces the valuesPrice array to the sum of its elements.

```
var reduceFunction1 = function(keyCustId, valuesPrices) {
                            return Array.sum(valuesPrices);
                        };
```

c) **Perform the map-reduce on all documents in the orders collection using the mapFunction1 map function and the reduceFunction1 reduce function.**

```
db.orders.mapReduce(
                            mapFunction1,
                            reduceFunction1,
                            { out: "map_reduce_example" }
```

)

This operation outputs the results to a collection named map_reduce_example. If the map_reduce_example collection already exists, the operation will replace the contents with the results of this map-reduce operation.

IMPLEMENTATION:

Use Employee database created in Assignment B-01 and perform Map reduces operation for following statements:

1. Return the Total Salary of per Company
2. Return the Total Salary of Company Name:"TCS"
3. Return the Avg Salary of Company whose address is “Pune".
4. Return the Total Salary for each Designation of Infosys.
5. Return total count for “State=AP”
6. Return Count for State AP and Age greater than 40.

CONCLUSION:

QUESTIONS:

1. What are different Aggregation commands?
2. What is map and reduce phase?
3. Explain Map Reduce Concurrency.
4. Write Step for MapReduce Operation with example.
5. What is Map-Reduce JavaScript Function?