ASSIGNMENT NO: 8

AIM : Implement MapReduce example in MongoDB with suitable dataset.

INDEX TERMS: Map Reduce, MongoDB,

THEORY

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 map Reduce database command. 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. Map Reduce 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 shared.

In MongoDB, map-reduce operations use custom JavaScript functions to map, or associate, values to a key. If a key has multiple values mapped to it, the operation reduces the values for the key to a single object. The use of custom JavaScript functions provides flexibility to map-reduce operations. For instance, when processing a document, the map function can create more than one key and value mapping or no mapping. Map-reduce operations can also use a custom JavaScript function to make final modifications to the results at the end of the map and reduce operation, such as per form additional calculations.

Map Reduce is a powerful and flexible tool for aggregating data. It can solve some problems that are too complex to express using the aggregation framework's query language. Map Reduce uses JavaScript as its "query language" so it can express arbitrarily complex logic. However, this power comes at a price: Map Reduce tends to be fairly slow and should not be used for real-time data analysis. Map Reduce can be easily parallelized across multiple servers. It splits up a problem, sends chunks of it to different machines, and lets each machine solve its part of the problem. When all the machines are finished, they merge all the pieces of the solution back into a full solution. Map Reduce has a couple of steps. It starts with the map step, which maps an operation onto every document in a collection. That operation could be either "do nothing" or "emit these keys with X values." There is then an intermediary stage called the shuffle step: keys are grouped and lists of emitted values are created for each key. The reduce takes this list of values and reduces it to a single element. This element is returned to the shuffle step until each key has a list containing a single value: the result.

Map-Reduce Behavior :

In MongoDB, the map-reduce operation can write results to a collection or return the results inline. If you write map-reduce output to a collection, you can perform subsequent map-reduce operations on the same input collection that merge replace, merge, or reduce new results with previous results. When returning the results of a map reduce operation inline, the result documents must be within the BSON Document Size limit, which is currently 16 megabytes. For additional information on limits and restrictions on map-reduce operations, see the map Reduce reference page. MongoDB supports map-reduce operations on shared collections. Map-reduce operations can also output the results to a shared collection.

MapReduce Example:

In the mongo shell, the db.collection.mapReduce() method is a wrapper around the map Reduce 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: newDate("Oct 04, 2012"),
status: 'A',
price: 25,
items:[{sku: "mmm",qty: 5, price: 2.5},
          {sku: "nnn",qty: 5, price: 2.5}]
}
```

To 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:

1) 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);
                                        };
```

2) Define the corresponding reduce function with two arguments key CustId and values Prices: The values Prices is an array whose elements are the price values emitted by the map function and grouped by key CustId. The function reduces the values Price array to the sum of its elements.

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

3) Perform the map-reduce on all documents in the orders collection using the map Function1 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. We can also use 'finalize'to perform the aggregation on map reduced result.

FAQS:
Q.1.Do Map Reduce works upon aggregation principles of MongoDB?
Q2. What is MapReduce?
Q3. .Which method is used to perform the map reduce operation?

Conclusion:
Outcome of the experiment is students are able to implement MapReduce Operation with MongoDB.