

MONGODB - MAP REDUCE

http://www.tutorialspoint.com/mongodb/mongodb_map_reduce.htm

Copyright © tutorialspoint.com

As per the MongoDB documentation, **Map-reduce** is a data processing paradigm for condensing large volumes of data into useful aggregated results. MongoDB uses **mapReduce** command for map-reduce operations. MapReduce is generally used for processing large data sets.

MapReduce Command:

Following is the syntax of the basic mapReduce command:

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

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.

In the above syntax:

- **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 having the same key
- **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
- **limit** specifies the optional maximum number of documents to be returned

Using MapReduce:

Consider the following document structure storing user posts. The document stores user_name of the user and the status of post.

```
{  
  "post_text": "tutorialspoint is an awesome website for tutorials",  
  "user_name": "mark",  
  "status": "active"  
}
```

Now, we will use a mapReduce function on our **posts** collection to select all the active posts, group them on the basis of user_name and then count the number of posts by each user using the following code:

```
>db.posts.mapReduce(  
  function() { emit(this.user_id,1); },  
  function(key, values) {return Array.sum(values)},  
  {  
    query:{status:"active"},  
    out:"post_total"  
  }  
)
```

The above mapReduce query outputs the following result:

```
{
  "result" : "post_total",
  "timeMillis" : 9,
  "counts" : {
    "input" : 4,
    "emit" : 4,
    "reduce" : 2,
    "output" : 2
  },
  "ok" : 1,
}
```

The result shows that a total of 4 documents matched the query *status:"active"* , the map function emitted 4 documents with key-value pairs and finally the reduce function grouped mapped documents having the same keys into 2.

To see the result of this mapReduce query use the find operator:

```
>db.posts.mapReduce(
  function() { emit(this.user_id,1); },
  function(key, values) {return Array.sum(values)},
  {
    query:{status:"active"},
    out:"post_total"
  }
).find()
```

The above query gives the following result which indicates that both users **tom** and **mark** have two posts in active states:

```
{ "_id" : "tom", "value" : 2 }
{ "_id" : "mark", "value" : 2 }
```

In similar manner, MapReduce queries can be used to construct large complex aggregation queries. The use of custom Javascript functions makes usage of MapReduce very flexible and powerful

Loading [MathJax]/jax/output/HTML-CSS/jax.js