MongoDB ObjectId MongoDB 全文检索

MongoDB Map Reduce

Map-Reduce是一种计算模型，简单的说就是将大批量的工作（数据）分解（MAP）执行，然后再将结果合并成最终结果（REDUCE）。

MongoDB提供的Map-Reduce非常灵活，对于大规模数据分析也相当实用。

MapReduce 命令

以下是MapReduce的基本语法：

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

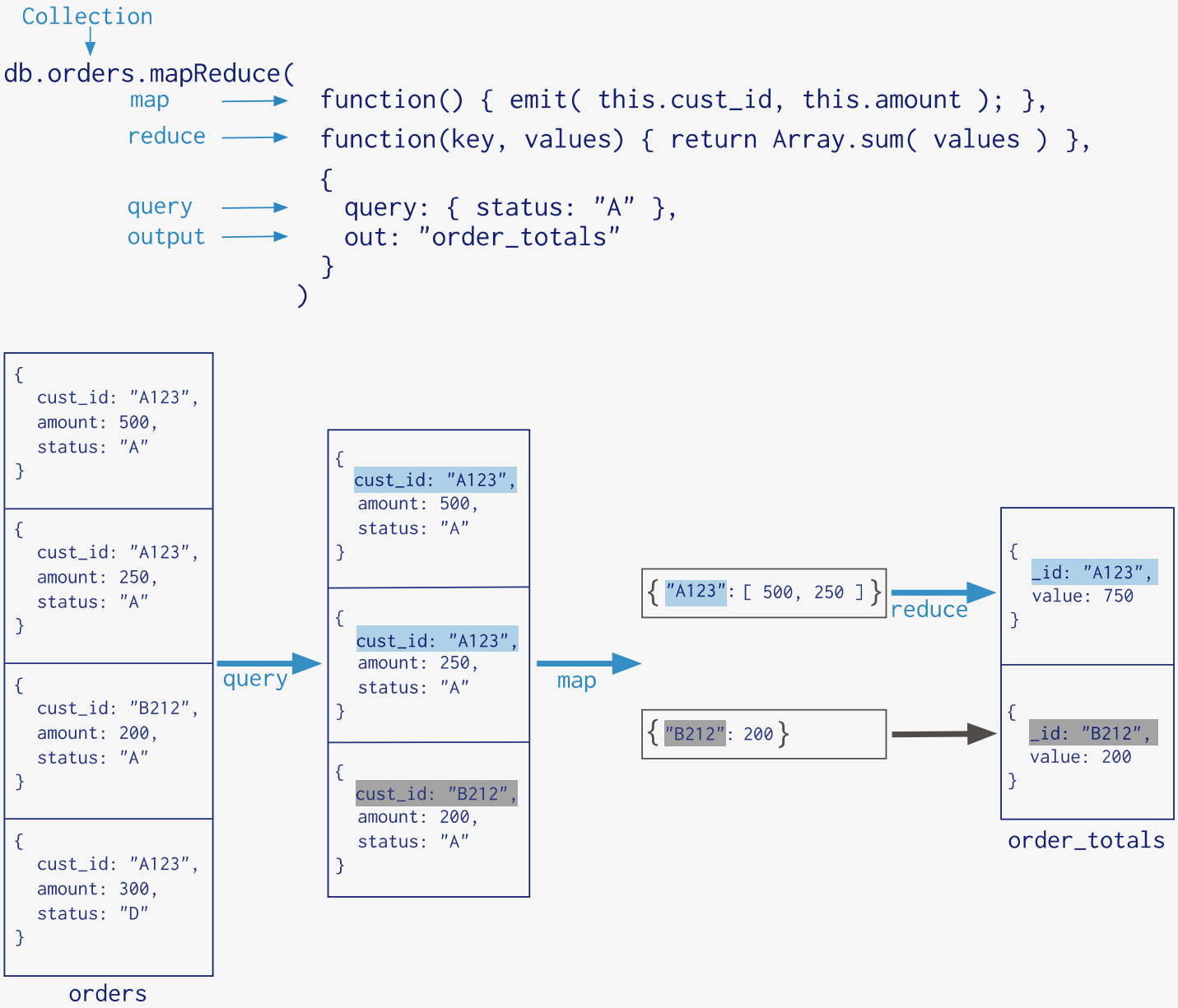
使用 MapReduce 要实现两个函数 Map 函数和 Reduce 函数,Map 函数调用 emit(key, value), 遍历 collection 中所有的记录, 将 key 与 value 传递给 Reduce 函数进行处理。

Map 函数必须调用 emit(key, value) 返回键值对。

参数说明:

- **map** ：映射函数 (生成键值对序列,作为 reduce 函数参数)。
- **reduce** 统计函数，reduce函数的任务就是将key-values变成key-value，也就是把values数组变成一个单一的值value。。
- **out** 统计结果存放集合 (不指定则使用临时集合,在客户端断开后自动删除)。
- **query** 一个筛选条件，只有满足条件的文档才会调用map函数。（query。limit，sort可以随意组合）
- **sort** 和limit结合的sort排序参数（也是在发往map函数前给文档排序），可以优化分组机制
- **limit** 发往map函数的文档数量的上限（要是没有limit，单独使用sort的用处不大）

以下实例在集合 orders 中查找 status:"A" 的数据，并根据 cust_id 来分组，并计算 amount 的总和。



使用 MapReduce

考虑以下文档结构存储用户的文章，文档存储了用户的 user_name 和文章的 status 字段：

```
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "mark",
   "status":"active"
})
WriteResult({ "nInserted" : 1 })
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "mark",
   "status":"active"
})
WriteResult({ "nInserted" : 1 })
```

```
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "mark",
   "status":"active"
})
WriteResult({ "nInserted" : 1 })
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "mark",
   "status":"active"
})
WriteResult({ "nInserted" : 1 })
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "mark",
   "status":"disabled"
})
WriteResult({ "nInserted" : 1 })
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "runoob",
   "status":"disabled"
})
WriteResult({ "nInserted" : 1 })
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "runoob",
   "status":"disabled"
})
WriteResult({ "nInserted" : 1 })
>db.posts.insert({
   "post_text": "菜鸟教程，最全的技术文档。",
   "user_name": "runoob",
   "status":"active"
})
WriteResult({ "nInserted" : 1 })
```

现在，我们将在 posts 集合中使用 mapReduce 函数来选取已发布的文章(status:"active")，并通过user_name分组，计算每个用户的文章数：

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

```
        }
)
```

以上 mapReduce 输出结果为：

```
{
        "result" : "post_total",
        "timeMillis" : 23,
        "counts" : {
                "input" : 5,
                "emit" : 5,
                "reduce" : 1,
                "output" : 2
        },
        "ok" : 1
}
```

结果表明，共有 5 个符合查询条件（status:"active"）的文档， 在map函数中生成了 5 个键值对文档，最后使用reduce函数将相同的键值分为 2 组。

具体参数说明：

- result：储存结果的collection的名字,这是个临时集合，MapReduce的连接关闭后自动就被删除了。
- timeMillis：执行花费的时间，毫秒为单位
- input：满足条件被发送到map函数的文档个数
- emit：在map函数中emit被调用的次数，也就是所有集合中的数据总量
- ouput：结果集合中的文档个数**（count对调试非常有帮助）**
- ok：是否成功，成功为1
- err：如果失败，这里可以有失败原因，不过从经验上来看，原因比较模糊，作用不大

使用 find 操作符来查看 mapReduce 的查询结果：

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

以上查询显示如下结果:

```
{ "_id" : "mark", "value" : 4 }
{ "_id" : "runoob", "value" : 1 }
```

用类似的方式，MapReduce可以被用来构建大型复杂的聚合查询。

Map函数和Reduce函数可以使用 JavaScript 来实现，使得MapReduce的使用非常灵活和强大。

← MongoDB ObjectId MongoDB 全文检索 →

1 篇笔记 写笔记

临时集合参数是这样写的

```
out: { inline: 1 }
```

设置了 `{inline:1}` 将不会创建集合，整个 **Map/Reduce** 的操作将会在内存中进行。

注意，这个选项只有在结果集单个文档大小在16MB限制范围内时才有效。

```
db.users.mapReduce(map,reduce,{out:{inline:1}});
```

forthxu 8个月前 (07-20)