[MongoDB 聚合](http://www.cnblogs.com/refactor/archive/2012/08/06/2592734.html)

MongoDB除了基本的查询功能,还提供了很多强大的聚合工具,其中简单的可计算集合中的文档个数,

复杂的可利用MapReduce做复杂数据分析.

1.count

count返回集合中的文档数量

db.refactor.count()

不管集合有多大,都能很快的返回文档数量.

可以传递查询,MongoDB会计算查询结果的数量

db.refactor.count({"username":"refactor"})

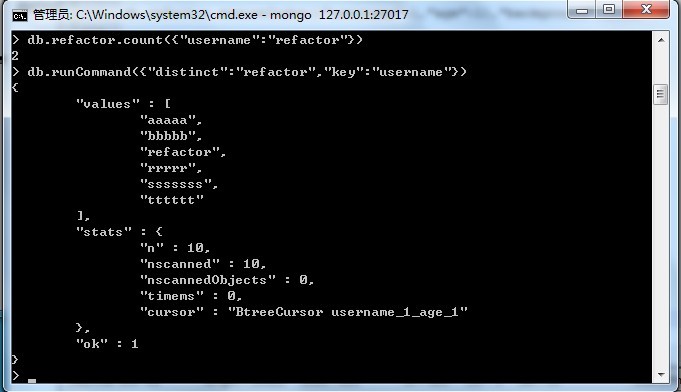
但是增加查询条件会使count变慢.

2.distinct

distinct用来找出给定键的所有不同值.使用时必须指定集合和键.

如:

db.runCommand({"distinct":"refactor","key":"username"})



 3.group

group先选定分组所依据的键,MongoDB将会将集合依据选定键值的不同分成若干组.然后可以通过聚合每一组内的文档,

产生一个结果文档.

如：

db.runCommand(  
{  
　　"group":  
　　{  
　　　　"ns":"refactor",  
　　　　"key":{"username":true},  
　　　　"initial":{"count":0},  
　　　　"$reduce":function(doc,prev)  
　　　　{  
　　　　　　prev.count++;  
　　　　},  
　　　　"condition":{"age":{"$gt":40}}  
　　}  
}  
)

   "ns":"refactor",

指定要进行分组的集合  
    "key":{"username":true},

指定文档分组的依据,这里是username键,所有username键的值相等的被划分到一组,true为返回键username的值  
    "initial":{"count":0},

每一组reduce函数调用的初始个数.每一组的所有成员都会使用这个累加器.  
    "$reduce":function(doc,prev){...}

每个文档都对应的调用一次.系统会传递两个参数:当前文档和累加器文档.

"condition":{"age":{"$gt":40}}

这个age的值大于40的条件

4.使用完成器

完成器用于精简从数据库传到用户的数据.group命令的输出一定要能放在单个数据库相应中.

"finalize"附带一个函数,在数组结果传递到客户端之前被调用一次.

db.runCommand(  
　　{  
　　　　"group":  
　　　　{  
　　　　　　"ns":"refactor",  
　　　　　　"key":{"username":true},  
　　　　　　"initial":{"count":0},  
　　　　　　"$reduce":function(doc,prev)  
　　　　　　{  
　　　　　　　　prev.count++;  
　　　　　　},  
　　　　　　"finalize":function(doc)  
　　　　　　{  
　　　　　　　　doc.num=doc.count;  
　　　　　　　　delete doc.count;  
　　　　　　}  
　　　　}  
　　}  
)

finalize能修改传递的参数也能返回新值.

5.将数组作为键使用

有些时候分组所依据的条件很复杂,不仅是一个键.比如要使用group计算每个类别有多篇博客文章.由于有很多作者,

给文章分类时可能不规律的使用了大小写.所以,如果要是按类别名来分组,最后"MongoDB"和"mongodb"就是不同的组.

为了消除这种大小写的影响,就要定义一个函数来确定文档所依据的键.

定义分组要用到$keyf

db.runCommand(  
 {  
  "group":  
   {  
    "ns":"refactor",  
    "$keyf":function(doc){return {"username":doc.username.toLowerCase()}},  
    "initial":{"count":0},  
    "$reduce":function(doc,prev)  
       {  
        prev.count++;  
       }  
   }  
 }  
)

6.MapReduce

count,distinct,group能做的事情MapReduce都能做.它是一个可以轻松并行化到多个服务器的聚合方法.它会

拆分问题,再将各个部分发送到不同机器上,让每台机器完成一部分.当所有机器都完成时候,再把结果汇集起来形成

最终完整的结果.

MapReduce需要几个步骤:

1.映射,将操作映射到集合中的每个文档.这个操作要么什么都不做,要么 产生一个键和n个值.

2.洗牌,按照键分组,并将产生的键值组成列表放到对应键中.

3.化简,把列表中的值 化简 成一个单值,这个值被返回.

4.重新洗牌,直到每个键的列表只有一个值为止,这个值就是最终结果.

MapReduce的速度比group慢,group也很慢.在应用程序中,最好不要用MapReduce,可以在后台运行MapReduce

创建一个保存结果的集合,可以对这个集合进行实时查询.

找出集合中的所有键

MongoDB没有模式,所以并不知晓每个文档有多少个键.通常找到集合的所有键的做好方式是用MapReduce.

在映射阶段,想得到文档中的每个键.map函数使用emit 返回要处理的值.emit会给MapReduce一个键和一个值.

这里用emit将文档某个键的记数(count)返回({count:1}).我们为每个键单独记数,所以为文档中的每一个键调用一次emit,

this是当前文档的引用:

map=function(){  
　　for(var key in this)  
　　{  
　　　　emit(key,{count:1})  
　　}  
};

这样返回了许许多多的{count:1}文档,每一个都与集合中的一个键相关.这种有一个或多个{count:1}文档组成的数组,

会传递给reduce函数.reduce函数有两个参数,一个是key,也就是emit返回的第一个值,另一个参数是数组,由一个或者多个

对应键的{count:1}文档组成.

reduce=function(key,emits){  
　　total=0;  
　　for(var i in emits){  
　　　　total+=emits[i].count;   
　　}  
　　return {count:total};  
}

reduce要能被反复被调用,不论是映射环节还是前一个化简环节.reduce返回的文档必须能作为reduce的

第二个参数的一个元素.如x键映射到了3个文档{"count":1,id:1},{"count":1,id:2},{"count":1,id:3}

其中id键用于区别.MongoDB可能这样调用reduce:

>r1=reduce("x",[{"count":1,id:1},{"count":1,id:2}])

{count:2}

>r2=reduce("x",[{"count":1,id:3}])

{count:1}

>reduce("x",[r1,r2])

{count:3}

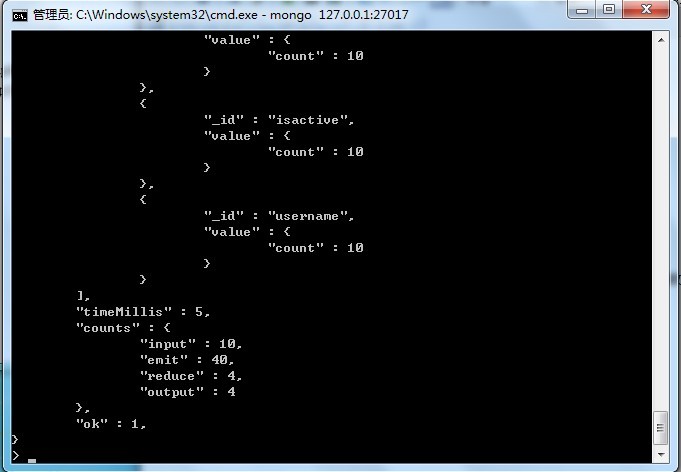
reduce应该能处理emit文档和其他reduce结果的各种集合.

如:

mr=db.runCommand(  
　　{  
　　"mapreduce":"refactor",  
　　"map":map,  
　　"reduce":reduce,  
　　"out":{inline:1}  
　　}  
)

或:

db.refactor.mapReduce(map,reduce,{out:{inline:1}})



"timeMillis" : 5,//操作花费的时间  
"counts" : {  
"input" : 10,//发往到map函数的文档个数  
"emit" : 40,//在map函数中emit被调用的次数  
"reduce" : 4,//在map函数中reduce被调用的次数  
"output" : 4//结果集合中创建的文档数量.  
},

1.mapreduce是根据map函数里调用的emit函数的第一个参数来进行分组的  
2.仅当根据分组键分组后一个键匹配多个文档，才会将key和文档集合交由reduce函数处理

注意MongoDB 1.8版本以上,必须指明 out 参数

否则会报如下错误:

"assertion" : "'out' has to be a string or an object",  
"assertionCode" : 13606,

MapReduce中的其他键

mapreduce,map,reduce这三个键是必须的,MapReduce命令还有其他的可选键

finalize:函数

将reduce的结果发送给这个键,这是处理过程的最后一步

keeptemp:布尔值

连接关闭时,临时结果是否保存

output:字符串

结果集合的名字,设定该项则隐含着keeptemp:true

query:文档

会在发往map函数前,先用指定条件过滤文档

sort:文档

会在发往map函数前先给文档排序

limit:整数

发往map函数文档的最大数量

scope:文档

javascript代码中要用到的变量

verbose:布尔值

是否产生更加信息的服务器日志