mongodb MongoDB 聚合 group

MongoDB 聚合

MongoDB中聚合(aggregate)主要用于处理数据(诸如统计平均值,求和等)，并返回计算后的数据结果。有点类似sql语句中的 count(*)。

基本语法为：db.collection.aggregate([<stage1>, <stage2>, ...])

现在在mycol集合中有以下数据：

{ "_id" : 1, "name" : "tom", "sex" : "男", "score" : 100, "age" : 34 }
{ "_id" : 2, "name" : "jeke", "sex" : "男", "score" : 90, "age" : 24 }
{ "_id" : 3, "name" : "kite", "sex" : "女", "score" : 40, "age" : 36 }
{ "_id" : 4, "name" : "herry", "sex" : "男", "score" : 90, "age" : 56 }
{ "_id" : 5, "name" : "marry", "sex" : "女", "score" : 70, "age" : 18 }
{ "_id" : 6, "name" : "john", "sex" : "男", "score" : 100, "age" : 31 }

1、$sum 计算总和。

Sql: select sex,count(*) from mycol group by sex

MongoDb: db.mycol.aggregate([{group: {_id: 'sex', personCount: {$sum: 1}}}])

```
> db.mycol.aggregate([{$group: {_id: '$sex', personCount: {$sum: 1}}}])
{ "_id" : "男", "personCount" : 4 }
{ "_id" : "女", "personCount" : 2 }
```

Sql: select sex,sum(score) totalScore from mycol group by sex

MongoDb: db.mycol.aggregate([{group: {_id: 'sex', totalScore: {$sum :'score'}}}])

```
> db.mycol.aggregate([{$group: {_id: '$sex', totalScore: {$sum: '$score'}}}])
{ "_id" : "男", "totalScore" : 380 }
{ "_id" : "女", "totalScore" : 110 }
```

2、$avg 计算平均值

Sql: select sex,avg(score) avgScore from mycol group by sex

Mongodb: db.mycol.aggregate([{group: {_id: 'sex', avgScore: {$avg :'score'}}}])

```
> db.mycol.aggregate([{$group: {_id: '$sex', avgScore: {$avg: '$score'}}}])
{ "_id" : "男", "avgScore" : 95 }
{ "_id" : "女", "avgScore" : 55 }
```

3、$max 获取集合中所有文档对应值得最大值。

Sql： select sex,max(score) maxScore from mycol group by sex

Mongodb: db.mycol.aggregate([{group: {_id: 'sex', maxScore : {$max :'score'}}}])

```
> db.mycol.aggregate([{$group: {_id: '$sex', maxScore : {$max: '$score'}}}])
{ "_id" : "男", "maxScore" : 100 }
{ "_id" : "女", "maxScore" : 70 }
```

4、$min 获取集合中所有文档对应值得最小值。

Sql： select sex,min(score) minScore from mycol group by sex

Mongodb: db.mycol.aggregate([{group: {_id: 'sex', minScore : {$min :'score'}}}])

```
> db.mycol.aggregate([{$group: {_id: '$sex', minScore : {$min: '$score'}}}])
{ "_id" : "男", "minScore" : 90 }
{ "_id" : "女", "minScore" : 40 }
```

5、$push 把文档中某一列对应的所有数据插入值到一个数组中。

Mongodb: db.mycol.aggregate([{group: {_id: 'sex', scores : {$push :'score'}}}])

```
> db.mycol.aggregate([{$group: {_id: '$sex', scores : {$push: '$score'}}}])
{ "_id" : "男", "scores" : [ 100, 90, 90, 100 ] }
{ "_id" : "女", "scores" : [ 40, 70 ] }
```

6、$addToSet 把文档中某一列对应的所有数据插入值到一个数组中,去掉重复的

db.mycol.aggregate([{group: {_id: 'sex', scores : {$addToSet :'score'}}}])

```
> Mongodb: db.mycol.aggregate([{$group: {_id: '$sex', scores : {$addToSet: '$score'}}}])
{ "_id" : "男", "scores" : [ 90, 100 ] }
{ "_id" : "女", "scores" : [ 70, 40 ] }
```

7、 $first 根据资源文档的排序获取第一个文档数据。

db.mycol.aggregate([{group: {_id: 'sex', firstPerson : {$first :'name'}}}])

```
db.mycol.aggregate([{$group: {_id: '$sex', firstPerson : {$first: '$name'}}}])
"_id" : "男", "firstPerson" : "tom" }
"_id" : "女", "firstPerson" : "kite" }
```

8、 $last 根据资源文档的排序获取最后一个文档数据。

db.mycol.aggregate([{group: {_id: 'sex', lastPerson : {$last :'name'}}}])

```
> db.mycol.aggregate([{$group: {_id: '$sex', lastPerson : {$last: '$name'}}}])
{ "_id" : "男", "lastPerson" : "john" }
{ "_id" : "女", "lastPerson" : "marry" }
```

9、全部统计 null

db.mycol.aggregate([{group:{_id:null,totalScore:{push:'$score'}}}])

```
> db.mycol.aggregate([{$group:{_id:null,totalScore:{$push:'$score'}}}])
{ "_id" : null, "totalScore" : [ 100, 90, 40, 90, 70, 100 ] }
```

例子

现在在t2集合中有以下数据：

{ "country" : "china", "province" : "sh", "userid" : "a" }
{ "country" : "china", "province" : "sh", "userid" : "b" }
{ "country" : "china", "province" : "sh", "userid" : "a" }
{ "country" : "china", "province" : "sh", "userid" : "c" }
{ "country" : "china", "province" : "bj", "userid" : "da" }
{ "country" : "china", "province" : "bj", "userid" : "fa" }

需求是统计出每个country/province下的userid的数量（同一个userid只统计一次）

过程如下。

首先试着这样来统计：

db.t2.aggregate([{ group: {"_id": { "country" : "country", "prov": "province"} , "number":{sum:1}} }])

结果是错误的：

```
{ "_id" : { "country" : "china", "prov" : "sh" }, "number" : 4 }
{ "_id" : { "country" : "china", "prov" : "bj" }, "number" : 3 }
```

原因是，这样来统计不能区分userid相同的情况（上面的数据中sh有两个 userid = a）

为了解决这个问题，首先执行一个group，其id 是 country, province, userid三个field：

db.t2.aggregate([{ group: {"_id": { "country" : "country", "province": "*province*","*uid*":"userid" } } }])

```
{ "_id" : { "country" : "china", "province" : "bj", "uid" : "da" } }
{ "_id" : { "country" : "china", "province" : "sh", "uid" : "a" } }
{ "_id" : { "country" : "china", "province" : "bj", "uid" : "fa" } }
{ "_id" : { "country" : "china", "province" : "sh", "uid" : "b" } }
{ "_id" : { "country" : "china", "province" : "sh", "uid" : "c" } }
```

可以看出，这步的目的是把相同的userid只剩下一个。

然后第二步，再第一步的结果之上再执行统计：

db.t2.aggregate([
{ group: {"_id": { "country" : "country", "province": "*province*","*uid*":"userid" } } } ,
{ group: {"_id": { "country" : "_id.country", "province": "_id.province" }, count : {sum : 1 } } }
])

这回就对了

```
{ "_id" : { "country" : "china", "province" : "bj" }, "count" : 2 }
{ "_id" : { "country" : "china", "province" : "sh" }, "count" : 3 }
```

加入一个$project操作符，把_id去掉

db.t2.aggregate([{ group: {"_id": { "country" : "country", "province": "*province*","*uid*":"userid" } } } ,
{ group: {"_id": { "country" : "_id.country", "province": "_id.province" }, count: { sum : 1 } } },
{ project : {"_id": 0, "country" : "_id.country", "province" : "$_id.province", "count" : 1}}
])

最终结果如下：

```
{ "count" : 2, "country" : "china", "province" : "bj" }
{ "count" : 3, "country" : "china", "province" : "sh" }
```

管道的概念

管道在Unix和Linux中一般用于将当前命令的输出结果作为下一个命令的参数。

MongoDB的聚合管道将MongoDB文档在一个管道处理完毕后将结果传递给下一个管道处理。管道操作是可以重复的。

表达式：处理输入文档并输出。表达式是无状态的，只能用于计算当前聚合管道的文档，不能处理其它的文档。

这里我们介绍一下聚合框架中常用的几个操作：

- $project：修改输入文档的结构。可以用来重命名、增加或删除域，也可以用于创建计算结果以及嵌套文档。
- *match*：用于过滤数据，只输出符合条件的文档。match使用MongoDB的标准查询操作。
- $limit：用来限制MongoDB聚合管道返回的文档数。
- $skip：在聚合管道中跳过指定数量的文档，并返回余下的文档。
- $unwind：将文档中的某一个数组类型字段拆分成多条，每条包含数组中的一个值。
- $group：将集合中的文档分组，可用于统计结果。
- $sort：将输入文档排序后输出。
- $geoNear：输出接近某一地理位置的有序文档。

1、$project实例

db.mycol.aggregate({$project:{name : 1, score : 1}})

```
{ "_id" : 1, "name" : "tom", "score" : 100 }
{ "_id" : 2, "name" : "jeke", "score" : 90 }
{ "_id" : 3, "name" : "kite", "score" : 40 }
{ "_id" : 4, "name" : "herry", "score" : 90 }
{ "_id" : 5, "name" : "marry", "score" : 70 }
{ "_id" : 6, "name" : "john", "score" : 100 }
```

这样的话结果中就只还有_id,name和score三个字段了，默认情况下_id字段是被包含的，如果要想不包含_id话可以这样：

db.mycol.aggregate({$project:{_id : 0, name : 1, score : 1}})

```
{ "name" : "tom", "score" : 100 }
{ "name" : "jeke", "score" : 90 }
{ "name" : "kite", "score" : 40 }
{ "name" : "herry", "score" : 90 }
{ "name" : "marry", "score" : 70 }
{ "name" : "john", "score" : 100 }
```

2、$match实例

*match*用于获取分数大于30小于并且小于100的记录，然后将符合条件的记录送到下一阶段group管道操作符进行处理

db.mycol.aggregate([{match :{score: {gt: 30, lt: 100}}},{group:{_id:'sex',count:{sum:1}}}])

```
> db.mycol.aggregate([{$match :{score: {$gt: 30, $lt: 100}}},{$group:{_id:'$sex',count:{$sum:1}}}])
{ "_id" : "男", "count" : 2 }
{ "_id" : "女", "count" : 2 }
>
```

分类: MongoDB

标签: mongodb MongoDB 聚合 group

shaomine
关注 - 9
粉丝 - 46
+加关注

2 0

« 上一篇 : mongodb Install the MongoDB service
» 下一篇 : MongoDB 覆盖索引查询

posted on 2016-08-11 14:25 shaomine 阅读(4715) 评论(0) 编辑 收藏