MongoDB常见的聚合操作

MongoDB中聚合使用aggregate(),语法为

```
db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

聚合表达式

表达式	描述
$sum	计算总和
$avg	计算平均值
$min	获取集合中所有文档对应值得最小值
$max	获取集合中所有文档对应值得最大值
$push	将值加入一个数组中，不会判断是否有重复的值
$addToSet	将值加入一个数组中，会判断是否有重复的值，若相同的值在数组中已经存在了，则不加入
$first	根据资源文档的排序获取第一个文档数据
$last	根据资源文档的排序获取最后一个文档数据

聚合管道

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。

管道操作符	描述
$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

管道操作符	描述
$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

聚合操作练习

1.创建数据库，创建集合，并插入数据

```
use mongotest #创建数据库
db.createCollection("student") #创建集合
db.student.deleteMany({}) #清空集合
db.student.insertMany([
{ name: "Joe", gender: "m", age: 23, birthdate: { "day": 15, "month": 3, "year": 1997
}, hobby: ["football", "basketball", "reading"], city: "Beijing", time: [9, 18] },
{ name: "Kate", gender: "f", age: 22, birthdate: { "day": 25, "month": 7, "year": 1998
}, hobby: ["reading", "piano"], city: "Hangzhou", time: [8, 17] },
{ name: "Rose", gender: "f", age: 24, birthdate: { "day": 3, "month": 3, "year": 1996
}, hobby: ["basketball", "running", "traveling"], city: "Shanghai", time: [9, 19] },
{ name: "Jason", gender: "m", age: 21, birthdate: { "day": 17, "month": 12, "year":
1999 }, hobby: ["cooking", "photography"], city: "Chengdu", time: [8, 20] },
{ name: "Grace", gender: "f", age: 22, birthdate: { "day": 10, "month": 6, "year":
1998 }, hobby: ["photography", "cooking", "drama"], city: "Nanjing", time: [9, 18] },
{ name: "Jessica", gender: "f", age: 22, birthdate: { "day": 21, "month": 3, "year":
1998 }, hobby: ["cooking", "piano"], city: "Shanghai", time: [10, 19] },
{ name: "Donna", gender: "f", age: 22, birthdate: { "day": 24, "month": 9, "year":
1998 }, hobby: ["violin", "drama"], city: "Shanghai", time: [9, 20] },
{ name: "Apple", gender: "m", age: 23, birthdate: { "day": 20, "month": 9, "year":
1997 }, hobby: ["violin", "running"], city: "Chengdu", time: [9, 19] },
{ name: "Baba", gender: "f", age: 25, birthdate: { "day": 20, "month": 9, "year":
1995 }, hobby: ["violin", "basketball"], city: "Chengdu", time: [10, 19] }
])
```

2.查询城市名称以及在这座城市的学生姓名

```
db.student.aggregate({$group:{_id:"$city",name:{$push:"$name"}}})
```

output

```
{ "_id" : "Chengdu", "name" : [ "Jason", "Apple", "Baba" ] }
{ "_id" : "Beijing", "name" : [ "Joe" ] }
{ "_id" : "Nanjing", "name" : [ "Grace" ] }
{ "_id" : "Hangzhou", "name" : [ "Kate" ] }
{ "_id" : "Shanghai", "name" : [ "Rose", "Jessica", "Donna" ] }
```

3.查询城市名称以及在这座城市的学生人数

```
db.student.aggregate({$group:{_id:"$city",count:{$sum:1}}})
```

output

```
{ "_id" : "Chengdu", "count" : 3 }
{ "_id" : "Beijing", "count" : 1 }
{ "_id" : "Nanjing", "count" : 1 }
{ "_id" : "Hangzhou", "count" : 1 }
{ "_id" : "Shanghai", "count" : 3 }
```

4.查询男同学和女同学的人数

```
db.student.aggregate({$group:{_id:"$gender",count:{$sum:1}}})
```

output

```
{ "_id" : "m", "count" : 3 }
{ "_id" : "f", "count" : 6 }
```

5.查询男同学和女同学的平均年龄，最大年龄，最小年龄

```
db.student.aggregate({$group:{_id:"$gender",avg_age:{$avg:"$age"},max_age:
{$max:"$age"},min_age:{$min:"$age"}}})
```

output

```
{ "_id" : "m", "avg_age" : 22.333333333333332, "max_age" : 23, "min_age" : 21 }
{ "_id" : "f", "avg_age" : 22.833333333333332, "max_age" : 25, "min_age" : 22 }
```

6.查询各城市内学生数小于2的城市的学生姓名，学生个数

```
db.student.aggregate(
{$group:{_id:"$city",name:{$push:"$name"},count:{$sum:1}}},
```

```
{$match:{count:{$lt:2}}},
{$project:{_id:0,name:1,count:1}}
)
```

output

```
{ "name" : [ "Joe" ], "count" : 1 }
{ "name" : [ "Grace" ], "count" : 1 }
{ "name" : [ "Kate" ], "count" : 1 }
```

7.将学生按照年龄由小到大排序（年龄相同看birthdate）,并显示学生姓名和年龄

```
db.student.aggregate({$sort:{age:1,"birthdate.month":-1, "birthdate.day":-1}},{$project:
{_id:0,name:1,age:1}})
```

output

```
{ "name" : "Jason", "age" : 21 }
{ "name" : "Donna", "age" : 22 }
{ "name" : "Kate", "age" : 22 }
{ "name" : "Grace", "age" : 22 }
{ "name" : "Jessica", "age" : 22 }
{ "name" : "Apple", "age" : 23 }
{ "name" : "Joe", "age" : 23 }
{ "name" : "Rose", "age" : 24 }
{ "name" : "Baba", "age" : 25 }
```

8.查询学生人数大于等于2的城市，并将这些城市学生的平均年龄升序排列,显示城市和平均年龄

```
db.student.aggregate(
{$group:{_id:"$city",avg_age:{$avg:"$age"},count_num:{$sum:1}}},
{$match:{count_num:{$gte:2}}},
{$sort:{avg_age:1}},
{$project:{avg_age:1}}
)
```

output

```
{ "_id" : "Shanghai", "avg_age" : 22.666666666666668 }
{ "_id" : "Chengdu", "avg_age" : 23 }
```

9.查询学生人数大于等于2的城市，并将这些城市学生的平均年龄降序排列，取第一个，显示城市和平均年龄

```
db.student.aggregate(
{$group:{_id:"$city",avg_age:{$avg:"$age"},count_num:{$sum:1}}},
{$match:{count_num:{$gte:2}}},
{$sort:{avg_age:-1}},
{$limit:1},
{$project:{avg_age:1}}
)
```

output

```
{ "_id" : "Chengdu", "avg_age" : 23 }
```

10.查询在各个时间点开始工作的学生人数

```
db.student.aggregate({$group:{_id:{"$arrayElemAt":["$time",0]},count:{$sum:1}}})
```

output

```
{ "_id" : 9, "count" : 5 }
{ "_id" : 8, "count" : 2 }
{ "_id" : 10, "count" : 2 }
```

11.查询拥有各个爱好的学生人数

```
db.student.aggregate({$unwind:"$hobby"},{$group:{_id:"$hobby",count:{$sum:1}}})
```

output

```
{ "_id" : "football", "count" : 1 }
{ "_id" : "cooking", "count" : 3 }
{ "_id" : "violin", "count" : 3 }
{ "_id" : "drama", "count" : 2 }
{ "_id" : "basketball", "count" : 3 }
{ "_id" : "reading", "count" : 2 }
{ "_id" : "piano", "count" : 2 }
{ "_id" : "running", "count" : 2 }
{ "_id" : "traveling", "count" : 1 }
{ "_id" : "photography", "count" : 2 }
```

索引的使用

创建索引的语法为

```
db.collection.createIndex(keys, options)
```

key为创建的索引字段，options为1或-1，指定按升序或降序创建索引

查看索引

```
db.col.dropIndex(keys, options)
```

删除索引

```
db.col.getIndexes()
```

1.在姓名上按升序建立索引

```
db.student.createIndex({name:1})
```

2.在姓名和年龄上建立复合索引，姓名按升序，年龄按降序

```
db.student.createIndex({name:1,age:-1})
```

性能示例

1.创建三十万条数据

```
for (var i = 1; i <= 300000; i++) {
    db.getCollection('testindex').insert({
       "name": "zhangsan" + i,
       "sex": Math.round(Math.random() * 10) % 2,
       "age": Math.round(Math.random() * 6) + 3
   });
}
```

2.对姓名建立索引

```
db.testindex.createIndex({name:1})
```

3.查询

```
db.testindex.find(
{
        $or: [{"name":"zhangsan10000"},{"name":"zhangsan14000"},{"name":"zhangsan9000"},
{"name":"zhangsan23000"},{"name":"zhangsan24050"},
              {"name":"zhangsan12000"},{"name":"zhangsan14300"},{"name":"zhangsan9300"},
{"name":"zhangsan23300"},{"name":"zhangsan24350"},
              {"name":"zhangsan11100"},{"name":"zhangsan15200"},{"name":"zhangsan8100"},
{"name":"zhangsan22100"},{"name":"zhangsan26150"},
              {"name":"zhangsan10200"},{"name":"zhangsan14020"},{"name":"zhangsan9020"},
{"name":"zhangsan23020"},{"name":"zhangsan24070"},
              {"name":"zhangsan10300"},{"name":"zhangsan14030"},{"name":"zhangsan9030"},
{"name":"zhangsan23030"},{"name":"zhangsan24080"}]
    }
)
```

4.测试以下三种方式需要花费的时间

- 创建数据——查询——建立索引
- 创建数据——建立索引——查询
- 建立索引——创建数据——查询