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。

MongoDB中采用aggregate()方法进行聚合，语法如下:

```
> db.COLLECTION_NAME.aggregate(pipeline, options)
```

其中: (1)pipeline类型为包含一系列aggreate pipeline操作符的array，例如$addFields, $bucket, $bucketAuto, $collStats, $count, $facet, $geoNear, graphlookup, $group, $indexStats, $limit, $listSessions, $lookup, $match, $out, $project, $redact, $replaceRoot, $sample, $skip,$sort, $sortByCount, $unwind等；(2)
options为可选参数，包含expain, allowDiskUse, cursor, maxTimeMS, bypassDocumentValidation, readConcern, collation, hint, comment等参数。

以下为一个简单的示例：

```
> db.orders.insertMany([{ _id: 1, cust_id: "abc1", ord_date: ISODate("2012-11-02T17:04:11.1
{ _id: 2, cust_id: "xyz1", ord_date: ISODate("2013-10-01T17:04:11.102Z"), status: "A", amou
{ _id: 3, cust_id: "xyz1", ord_date: ISODate("2013-10-12T17:04:11.102Z"), status: "D", amou
{ _id: 4, cust_id: "xyz1", ord_date: ISODate("2013-10-11T17:04:11.102Z"), status: "D", amou
{ _id: 5, cust_id: "abc1", ord_date: ISODate("2013-11-12T17:04:11.102Z"), status: "A", amou
BulkWriteResult({
    "writeErrors" : [ ],
    "writeConcernErrors" : [ ],
    "nInserted" : 5,
    "nUpserted" : 0,
    "nMatched" : 0,
    "nModified" : 0,
    "nRemoved" : 0,
    "upserted" : [ ]
})
> db.orders.aggregate([
                     { $match: { status: "A" } },
                     { $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
                     { $sort: { total: -1 } }
                   ])
{ "_id" : "xyz1", "total" : 100 }
{ "_id" : "abc1", "total" : 75 }
```

pipeline

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。

```
> db.books.insertOne({"_id" : 1,
  title: "abc123",
  isbn: "0001122223334",
  author: { last: "zzz", first: "aaa" },
  copies: 5})
{ "acknowledged" : true, "insertedId" : 1 }
> db.books.aggregate([{$project: {title: 1 , author:1}}])
{ "_id" : 1, "title" : "abc123", "author" : { "last" : "zzz", "first" : "aaa" } }
> db.books.aggregate([{$project: {_id: 0, title: 1, author: 1}}])
{ "title" : "abc123", "author" : { "last" : "zzz", "first" : "aaa" } }
> db.books.aggregate([{$project : {"author.first": 0}}])
{ "_id" : 1, "title" : "abc123", "isbn" : "0001122223334", "author" : { "last" : "zzz" }, "
> db.books.aggregate([{$project: {"author": {"first": 0}}}])
{ "_id" : 1, "title" : "abc123", "isbn" : "0001122223334", "author" : { "last" : "zzz" }, "
> db.books.aggregate(
   [
      {
         $project: {
            title: 1,
            isbn: {
               prefix: { $substr: [ "$isbn", 0, 3 ] },
               group: { $substr: [ "$isbn", 3, 2 ] },
               publisher: { $substr: [ "$isbn", 5, 4 ] },
               title: { $substr: [ "$isbn", 9, 3 ] },
               checkDigit: { $substr: [ "$isbn", 12, 1] }
            },
            lastName: "$author.last",
            copiesSold: "$copies"
         }
      }
   ]
).pretty()
{
    "_id" : 1,
    "title" : "abc123",
    "isbn" : {
        "prefix" : "000",
        "group" : "11",
        "publisher" : "2222",
        "title" : "333",
        "checkDigit" : "4"
    },
    "lastName" : "zzz",
    "copiesSold" : 5
}

> db.books.aggregate([{$project: {name:["$author.first", "$author.last"]}}])
{ "_id" : 1, "name" : [ "aaa", "zzz" ] }
```

- $match：用于过滤数据，只输出符合条件的文档。$match使用MongoDB的标准查询操作。

```
> db.articles.insertMany([
  {"author" : "dave", "score" : 80, "views" : 100},
  {"author" : "dave", "score" : 85, "views" : 521},
  {"author" : "ahn", "score" : 60, "views" : 1000},
  {"author" : "li", "score" : 55, "views" : 5000},
  {"author" : "annT", "score" : 60, "views" : 50},
  {"author" : "li", "score" : 94, "views" : 999},
  {"author" : "ty", "score" : 95, "views" : 1000}
  ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5bd71b15ddbdddebd6e3e9db"),
                ObjectId("5bd71b15ddbdddebd6e3e9dc"),
                ObjectId("5bd71b15ddbdddebd6e3e9dd"),
                ObjectId("5bd71b15ddbdddebd6e3e9de"),
                ObjectId("5bd71b15ddbdddebd6e3e9df"),
                ObjectId("5bd71b15ddbdddebd6e3e9e0"),
                ObjectId("5bd71b15ddbdddebd6e3e9e1")
        ]
}
> db.articles.aggregate([{$match : {author : "dave" }}])
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e2"), "author" : "dave", "score" : 80, "views" :
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e3"), "author" : "dave", "score" : 85, "views" :
> db.articles.aggregate([
  { $match: { $or: [ { score: { $gt: 70, $lt: 90 } }, { views: { $gte: 1000 } } ] } },
  { $group: { _id: null, count: { $sum: 1 } } }
  ])
{ "_id" : null, "count" : 5 }
```

- $limit：用来限制MongoDB聚合管道返回的文档数。

```
> db.articles.aggregate([{$limit:2}])
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e2"), "author" : "dave", "score" : 80, "views" :
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e3"), "author" : "dave", "score" : 85, "views" :
```

- $skip：在聚合管道中跳过指定数量的文档，并返回余下的文档。

```
> db.articles.aggregate([{$skip:2}])
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e4"), "author" : "ahn", "score" : 60, "views" : 1
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e5"), "author" : "li", "score" : 55, "views" : 50
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e6"), "author" : "annT", "score" : 60, "views" :
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e7"), "author" : "li", "score" : 94, "views" : 99
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e8"), "author" : "ty", "score" : 95, "views" : 10
```

- $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。

```
> db.inventory.insertOne({ "_id" : 1, "item" : "ABC1", sizes: [ "S", "M", "L"]})
{ "acknowledged" : true, "insertedId" : 1 }
> db.inventory.aggregate([{$unwind:"$sizes"}])
{ "_id" : 1, "item" : "ABC1", "sizes" : "S" }
{ "_id" : 1, "item" : "ABC1", "sizes" : "M" }
{ "_id" : 1, "item" : "ABC1", "sizes" : "L" }
```

- $group：将集合中的文档分组，可用于统计结果，语法：`{ $group: { _id: <expression>, <field1>: { <accumulator1> : <expression1> }, ... } }`。`_id`必须给定，但可通过设置其为`null`统计所有输入文档的累积值(总和，平均)。以下为聚合中$group中的一些表达式：

表达式	描述	实例
$sum	计算和	`db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])`
$avg	计算平均值	`db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])`
$min	获取集合中所有文档对应值得最小值。	`db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])`
$max	获取集合中所有文档对应值得最大值。	`db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])`

$push	在结果文档中插入值到一个数组中。	`db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])`
$addToSet	在结果文档中插入值到一个数组中，但不创建副本。	`db.mycol.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])`
$first	根据资源文档的排序获取第一个文档数据。	`db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])`
$last	根据资源文档的排序获取最后一个文档数据	`db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])`

```
> db.sales.insertMany([{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" :
{ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-03-01T09:
{ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-03-15T09:
{ "_id" : 4, "item" : "xyz", "price" : 5, "quantity" : 20, "date" : ISODate("2014-04-04T11:
{ "_id" : 5, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-04-04T21
{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4, 5 ] }
> db.sales.aggregate(
   [
      {
        $group : {
           _id : { month: { $month: "$date" }, day: { $dayOfMonth: "$date" }, year: { $year
           totalPrice: { $sum: { $multiply: [ "$price", "$quantity" ] } },
           averageQuantity: { $avg: "$quantity" },
           count: { $sum: 1 }
        }
      }
   ]
).pretty()
{
    "_id" : {
        "month" : 4,
        "day" : 4,
        "year" : 2014
    },
    "totalPrice" : 200,
    "averageQuantity" : 15,
    "count" : 2
}
{
    "_id" : {
        "month" : 3,
        "day" : 15,
        "year" : 2014
    },
    "totalPrice" : 50,
    "averageQuantity" : 10,
    "count" : 1
}
{
    "_id" : {
        "month" : 3,
        "day" : 1,
        "year" : 2014
    },
    "totalPrice" : 40,
    "averageQuantity" : 1.5,
    "count" : 2
}

> db.sales.aggregate(
   [
      {
        $group : {
           _id : null,
           totalPrice: { $sum: { $multiply: [ "$price", "$quantity" ] } },
           averageQuantity: { $avg: "$quantity" },
           count: { $sum: 1 }
        }
      }
   ]
).pretty()
{ "_id" : null, "totalPrice" : 290, "averageQuantity" : 8.6, "count" : 5 }
> db.sales.aggregate( [ { $group : { _id : "$item" } } ] )
{ "_id" : "xyz" }
{ "_id" : "jkl" }
{ "_id" : "abc" }
> db.books.aggregate([{$group:{_id:"$author", t_views: {$push:"$views"}}}])
{ "_id" : "ty", "t_views" : [ 1000 ] }
{ "_id" : "annT", "t_views" : [ 50 ] }
{ "_id" : "ahn", "t_views" : [ 1000 ] }
{ "_id" : "dave", "t_views" : [ 100, 521 ] }
{ "_id" : "li", "t_views" : [ 5000, 999 ] }
{ "_id" : { "last" : "zzz", "first" : "aaa" }, "t_views" : [ ] }
```

- $sort：将输入文档排序后输出。语法: `{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }`。$sort takes a document that specifies the field(s) to sort by and the respective sort order. can have one of the following values: (1) 1 to specify ascending order. (2) -1 to specify descending order. (3) { $meta: "textScore" } to sort by the computed textScore metadata in descending order. See Metadata Sort for an example.

```
> db.articles.aggregate([
...   { $match: { $or: [ { score: { $gt: 70, $lt: 90 } }, { views: { $gte: 1000 } } ] } },
... ])
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e2"), "author" : "dave", "score" : 80, "views" :
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e3"), "author" : "dave", "score" : 85, "views" :
```

```
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e8"), "author" : "ty", "score" : 95, "views" : 10
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e4"), "author" : "ahn", "score" : 60, "views" : 1
{ "_id" : ObjectId("5bd71b3dddbdddebd6e3e9e5"), "author" : "li", "score" : 55, "views" : 50
```