# MongoDB 概念解析

不管我们学习什么数据库都应该学习其中的基础概念，在mongodb中基本的概念是文档、集合、数据库，下面我们挨个介绍。

下表将帮助您更容易理解Mongo中的一些概念：

|  |  |  |
| --- | --- | --- |
| **SQL术语/概念** | **MongoDB术语/概念** | **解释/说明** |
| database | database | 数据库 |
| table | collection | 数据库表/集合 |
| row | document | 数据记录行/文档 |
| column | field | 数据字段/域 |
| index | index | 索引 |
| table joins |  | 表连接,MongoDB不支持 |
| primary key | primary key | 主键,MongoDB自动将\_id字段设置为主键 |

通过下图实例，我们也可以更直观的了解Mongo中的一些概念：



## 数据库

一个mongodb中可以建立多个数据库。

MongoDB的默认数据库为"db"，该数据库存储在data目录中。

MongoDB的单个实例可以容纳多个独立的数据库，每一个都有自己的集合和权限，不同的数据库也放置在不同的文件中。

**"show dbs"** 命令可以显示所有数据的列表。

$ ./mongo

MongoDB shell version: 3.0.6

connecting to: test

> show dbs

local 0.078GB

test 0.078GB

>

执行 **"db"** 命令可以显示当前数据库对象或集合。

$ ./mongo

MongoDB shell version: 3.0.6

connecting to: test

> db

test

>



运行"use"命令，可以连接到一个指定的数据库。

> use local

switched to db local

> db

local

>

以上实例命令中，"local" 是你要链接的数据库。

在下一个章节我们将详细讲解MongoDB中命令的使用。

数据库也通过名字来标识。数据库名可以是满足以下条件的任意UTF-8字符串。

* **不能是空字符串（"")。**
* **不得含有' '（空格)、.、$、/、\和\0 (空字符)。**
* **应全部小写。**
* **最多64字节。**

有一些数据库名是保留的，可以直接访问这些有特殊作用的数据库。

* **admin**： 从权限的角度来看，这是"root"数据库。要是将一个用户添加到这个数据库，这个用户自动继承所有数据库的权限。一些特定的服务器端命令也只能从这个数据库运行，比如列出所有的数据库或者关闭服务器。
* **local:** 这个数据永远不会被复制，可以用来存储限于本地单台服务器的任意集合
* **config**: 当Mongo用于分片设置时，config数据库在内部使用，用于保存分片的相关信息。

## 文档

文档是一组键值(key-value)对(即BSON)。MongoDB 的文档不需要设置相同的字段，并且相同的字段不需要相同的数据类型，这与关系型数据库有很大的区别，也是 MongoDB 非常突出的特点。

一个简单的文档例子如下：

{"site":"www.runoob.com", "name":"菜鸟教程"}

下表列出了 RDBMS 与 MongoDB 对应的术语：

|  |  |
| --- | --- |
| **RDBMS** | **MongoDB** |
| 数据库 | 数据库(database) |
| 表格 | 集合(collection) |
| 行 | 文档(document) |
| 列 | 字段 |
| 表联合 | 嵌入文档 |
| 主键 | 主键 (MongoDB 提供了 key 为 \_id ) |
| **数据库服务和客户端** | |
| Mysqld/Oracle | mongod |
| mysql/sqlplus | mongo |

需要注意的是：

1. 文档中的键/值对是有序的。
2. 文档中的值不仅可以是在双引号里面的字符串，还可以是其他几种数据类型（甚至可以是整个嵌入的文档)。
3. MongoDB区分类型和大小写。
4. MongoDB的文档不能有重复的键。
5. 文档的键是字符串。除了少数例外情况，键可以使用任意UTF-8字符。

文档键命名规范：

* **键不能含有\0 (空字符)。这个字符用来表示键的结尾。**
* **.和$有特别的意义，只有在特定环境下才能使用。**
* **以下划线"\_"开头的键是保留的(不是严格要求的)。**

## 集合

集合就是 MongoDB 文档组，类似于 RDBMS （关系数据库管理系统：Relational Database Management System)中的表格。

集合存在于数据库中，集合没有固定的结构，这意味着你在对集合可以插入不同格式和类型的数据，但通常情况下我们插入集合的数据都会有一定的关联性。

比如，我们可以将以下不同数据结构的文档插入到集合中：

{"site":"www.baidu.com"}

{"site":"www.google.com","name":"Google"}

{"site":"www.runoob.com","name":"菜鸟教程","num":5}

当第一个文档插入时，集合就会被创建。

### 合法的集合名

* 集合名不能是空字符串""。
* 集合名不能含有\0字符（空字符)，这个字符表示集合名的结尾。
* 集合名不能以"system."开头，这是为系统集合保留的前缀。
* 用户创建的集合名字不能含有保留字符。有些驱动程序的确支持在集合名里面包含，这是因为某些系统生成的集合中包含该字符。除非你要访问这种系统创建的集合，否则千万不要在名字里出现$。

如下实例：

db.col.findOne()

### capped collections

Capped collections 就是固定大小的collection。

它有很高的性能以及队列过期的特性(过期按照插入的顺序). 有点和 "RRD" 概念类似。

Capped collections 是高性能自动的维护对象的插入顺序。它非常适合类似记录日志的功能和标准的 collection 不同，你必须要显式的创建一个capped collection，指定一个 collection 的大小，单位是字节。collection 的数据存储空间值提前分配的。

Capped collections 可以按照文档的插入顺序保存到集合中，而且这些文档在磁盘上存放位置也是按照插入顺序来保存的，所以当我们更新Capped collections 中文档的时候，更新后的文档不可以超过之前文档的大小，这样话就可以确保所有文档在磁盘上的位置一直保持不变。

由于 Capped collection 是按照文档的插入顺序而不是使用索引确定插入位置，这样的话可以提高增添数据的效率。MongoDB 的操作日志文件 oplog.rs 就是利用 Capped Collection 来实现的。

要注意的是指定的存储大小包含了数据库的头信息。

db.createCollection("mycoll", {capped:true, size:100000})

* 在 capped collection 中，你能添加新的对象。
* 能进行更新，然而，对象不会增加存储空间。如果增加，更新就会失败 。
* 使用 Capped Collection 不能删除一个文档，可以使用 drop() 方法删除 collection 所有的行。
* 删除之后，你必须显式的重新创建这个 collection。
* 在32bit机器中，capped collection 最大存储为 1e9( 1X109)个字节。

## 元数据

数据库的信息是存储在集合中。它们使用了系统的命名空间：

dbname.system.\*

在MongoDB数据库中名字空间 <dbname>.system.\* 是包含多种系统信息的特殊集合(Collection)，如下:

|  |  |
| --- | --- |
| **集合命名空间** | **描述** |
| dbname.system.namespaces | 列出所有名字空间。 |
| dbname.system.indexes | 列出所有索引。 |
| dbname.system.profile | 包含数据库概要(profile)信息。 |
| dbname.system.users | 列出所有可访问数据库的用户。 |
| dbname.local.sources | 包含复制对端（slave）的服务器信息和状态。 |

对于修改系统集合中的对象有如下限制。

在{{system.indexes}}插入数据，可以创建索引。但除此之外该表信息是不可变的(特殊的drop index命令将自动更新相关信息)。

{{system.users}}是可修改的。 {{system.profile}}是可删除的。

## MongoDB 数据类型

下表为MongoDB中常用的几种数据类型。

|  |  |
| --- | --- |
| **数据类型** | **描述** |
| String | 字符串。存储数据常用的数据类型。在 MongoDB 中，UTF-8 编码的字符串才是合法的。 |
| Integer | 整型数值。用于存储数值。根据你所采用的服务器，可分为 32 位或 64 位。 |
| Boolean | 布尔值。用于存储布尔值（真/假）。 |
| Double | 双精度浮点值。用于存储浮点值。 |
| Min/Max keys | 将一个值与 BSON（二进制的 JSON）元素的最低值和最高值相对比。 |
| Array | 用于将数组或列表或多个值存储为一个键。 |
| Timestamp | 时间戳。记录文档修改或添加的具体时间。 |
| Object | 用于内嵌文档。 |
| Null | 用于创建空值。 |
| Symbol | 符号。该数据类型基本上等同于字符串类型，但不同的是，它一般用于采用特殊符号类型的语言。 |
| Date | 日期时间。用 UNIX 时间格式来存储当前日期或时间。你可以指定自己的日期时间：创建 Date 对象，传入年月日信息。 |
| Object ID | 对象 ID。用于创建文档的 ID。 |
| Binary Data | 二进制数据。用于存储二进制数据。 |
| Code | 代码类型。用于在文档中存储 JavaScript 代码。 |
| Regular expression | 正则表达式类型。用于存储正则表达式。 |

下面说明下几种重要的数据类型。

### ObjectId

ObjectId 类似唯一主键，可以很快的去生成和排序，包含 12 bytes，含义是：

* 前 4 个字节表示创建 **unix** 时间戳,格林尼治时间 **UTC** 时间，比北京时间晚了 8 个小时
* 接下来的 3 个字节是机器标识码
* 紧接的两个字节由进程 id 组成 PID
* 最后三个字节是随机数



MongoDB 中存储的文档必须有一个 \_id 键。这个键的值可以是任何类型的，默认是个 ObjectId 对象

由于 ObjectId 中保存了创建的时间戳，所以你不需要为你的文档保存时间戳字段，你可以通过 getTimestamp 函数来获取文档的创建时间:

> var newObject = ObjectId()

> newObject.getTimestamp()

ISODate("2017-11-25T07:21:10Z")

ObjectId 转为字符串

> newObject.str

5a1919e63df83ce79df8b38f

### 字符串

**BSON 字符串都是 UTF-8 编码。**

### 时间戳

BSON 有一个特殊的时间戳类型用于 MongoDB 内部使用，与普通的 日期 类型不相关。 时间戳值是一个 64 位的值。其中：

* 前32位是一个 time\_t 值（与Unix新纪元相差的秒数）
* 后32位是在某秒中操作的一个递增的序数

在单个 mongod 实例中，时间戳值通常是唯一的。

在复制集中， oplog 有一个 ts 字段。这个字段中的值使用BSON时间戳表示了操作时间。

*BSON 时间戳类型主要用于 MongoDB 内部使用。在大多数情况下的应用开发中，你可以使用 BSON 日期类型。*

### 日期

表示当前距离 Unix新纪元（1970年1月1日）的毫秒数。日期类型是有符号的, 负数表示 1970 年之前的日期。

> var mydate1 = new Date() //格林尼治时间

> mydate1

ISODate("2018-03-04T14:58:51.233Z")

> typeof mydate1

object

> var mydate2 = ISODate() //格林尼治时间

> mydate2

ISODate("2018-03-04T15:00:45.479Z")

> typeof mydate2

object

这样创建的时间是日期类型，可以使用 JS 中的 Date 类型的方法。

返回一个时间类型的字符串：

> var mydate1str = mydate1.toString()

> mydate1str

Sun Mar 04 2018 14:58:51 GMT+0000 (UTC)

> typeof mydate1str

string

或者

> Date()

Sun Mar 04 2018 15:02:59 GMT+0000 (UTC)

# MongoDB 创建数据库

## 语法

MongoDB 创建数据库的语法格式如下：

use DATABASE\_NAME

如果数据库不存在，则创建数据库，否则切换到指定数据库。

## 实例

以下实例我们创建了数据库 runoob:

> use runoob

switched to db runoob

> db

runoob

>

如果你想查看所有数据库，可以使用 **show dbs** 命令：

> show dbs

admin 0.000GB

local 0.000GB

>

可以看到，我们刚创建的数据库 runoob 并不在数据库的列表中， 要显示它，我们需要向 runoob 数据库插入一些数据。

> db.runoob.insert({"name":"菜鸟教程"})

WriteResult({ "nInserted" : 1 })

> show dbs

local 0.078GB

runoob 0.078GB

test 0.078GB

>

MongoDB 中默认的数据库为 test，如果你没有创建新的数据库，集合将存放在 test 数据库中。

***注意:****在 MongoDB 中，集合只有在内容插入后才会创建! 就是说，创建集合(数据表)后要再插入一个文档(记录)，集合才会真正创建。*

# MongoDB 删除数据库

## 语法

MongoDB 删除数据库的语法格式如下：

db.dropDatabase()

删除当前数据库，默认为 test，你可以使用 db 命令查看当前数据库名。

## 实例

以下实例我们删除了数据库 runoob。

首先，查看所有数据库：

> show dbs

local 0.078GB

runoob 0.078GB

test 0.078GB

接下来我们切换到数据库 runoob：

> use runoob

switched to db runoob

>

执行删除命令：

> db.dropDatabase()

{ "dropped" : "runoob", "ok" : 1 }

最后，我们再通过 show dbs 命令数据库是否删除成功：

> show dbs

local 0.078GB

test 0.078GB

>

## 删除集合

集合删除语法格式如下：

db.collection.drop()

以下实例删除了 runoob 数据库中的集合 site：

> use runoob

switched to db runoob

> show tables

site

> db.site.drop()

true

> show tables

>

# MongoDB 创建集合

本章节我们为大家介绍如何使用 MongoDB 来创建集合。

MongoDB 中使用 **createCollection()** 方法来创建集合。

语法格式：

db.createCollection(name, options)

参数说明：

* name: 要创建的集合名称
* options: 可选参数, 指定有关内存大小及索引的选项

options 可以是如下参数：

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| capped | 布尔 | （可选）如果为 true，则创建固定集合。固定集合是指有着固定大小的集合，当达到最大值时，它会自动覆盖最早的文档。 **当该值为 true 时，必须指定 size 参数。** |
| autoIndexId | 布尔 | （可选）如为 true，自动在 \_id 字段创建索引。默认为 false。 |
| size | 数值 | （可选）为固定集合指定一个最大值（以字节计）。 **如果 capped 为 true，也需要指定该字段。** |
| max | 数值 | （可选）指定固定集合中包含文档的最大数量。 |

在插入文档时，MongoDB 首先检查固定集合的 size 字段，然后检查 max 字段。

## 实例

在 test 数据库中创建 runoob 集合：

> use test

switched to db test

> db.createCollection("runoob")

{ "ok" : 1 }

>

如果要查看已有集合，可以使用 show collections 命令：

> show collections

runoob

system.indexes

下面是带有几个关键参数的 createCollection() 的用法：

创建固定集合 mycol，整个集合空间大小 6142800 KB, 文档最大个数为 10000 个。

> db.createCollection("mycol", { capped : true, autoIndexId : true, size :

6142800, max : 10000 } )

{ "ok" : 1 }

>

在 MongoDB 中，你不需要创建集合。当你插入一些文档时，MongoDB 会自动创建集合。

> db.mycol2.insert({"name" : "菜鸟教程"})

> show collections

mycol2

...

# MongoDB 删除集合

本章节我们为大家介绍如何使用 MongoDB 来删除集合。

MongoDB 中使用 drop() 方法来删除集合。

**语法格式：**

db.collection.drop()

参数说明：

* 无

**返回值**

如果成功删除选定集合，则 drop() 方法返回 true，否则返回 false。

## 实例

在数据库 mydb 中，我们可以先通过 **show collections** 命令查看已存在的集合：

>use mydb

switched to db mydb

>show collections

mycol

mycol2

system.indexes

runoob

>

接着删除集合 mycol2 :

>db.mycol2.drop()

true

>

通过 show collections 再次查看数据库 mydb 中的集合：

>show collections

mycol

system.indexes

runoob

>

从结果中可以看出 mycol2 集合已被删除。

# MongoDB 插入文档

本章节中我们将向大家介绍如何将数据插入到MongoDB的集合中。

文档的数据结构和JSON基本一样。

所有存储在集合中的数据都是BSON格式。

BSON是一种类json的一种二进制形式的存储格式,简称Binary JSON。

## 插入文档

MongoDB 使用 insert() 或 save() 方法向集合中插入文档，语法如下：

db.COLLECTION\_NAME.insert(document)

### 实例

以下文档可以存储在 MongoDB 的 runoob 数据库 的 col 集合中：

>db.col.insert({title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

})

以上实例中 col 是我们的集合名，如果该集合不在该数据库中， MongoDB 会自动创建该集合并插入文档。

查看已插入文档：

> db.col.find()

{ "\_id" : ObjectId("56064886ade2f21f36b03134"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb", "database", "NoSQL" ], "likes" : 100 }

>

我们也可以将数据定义为一个变量，如下所示：

> document=({title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

});

执行后显示结果如下：

{

"title" : "MongoDB 教程",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100

}

执行插入操作：

> db.col.insert(document)

WriteResult({ "nInserted" : 1 })

>

插入文档你也可以使用 db.col.save(document) 命令。如果不指定 \_id 字段 save() 方法类似于 insert() 方法。如果指定 \_id 字段，则会更新该 \_id 的数据。

* db.collection.insertOne():向指定集合中插入一条文档数据
* db.collection.insertMany():向指定集合中插入多条文档数据

# 插入单条数据

> var document = db.collection.insertOne({"a": 3})

> document

{

"acknowledged" : true,

"insertedId" : ObjectId("571a218011a82a1d94c02333")

}

# 插入多条数据

> var res = db.collection.insertMany([{"b": 3}, {'c': 4}])

> res

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("571a22a911a82a1d94c02337"),

ObjectId("571a22a911a82a1d94c02338")

]

}

# MongoDB 更新文档

MongoDB 使用 **update()** 和 **save()** 方法来更新集合中的文档。接下来让我们详细来看下两个函数的应用及其区别。

## update() 方法

update() 方法用于更新已存在的文档。语法格式如下：

db.collection.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>

}

)

**参数说明：**

* **query**: update的查询条件，类似sql update查询内where后面的。
* **update**: update的对象和一些更新的操作符（如$,$inc...）等，也可以理解为sql update查询内set后面的
* **upsert**: 可选，这个参数的意思是，如果不存在update的记录，是否插入objNew,true为插入，默认是false，不插入。
* **multi**: 可选，mongodb 默认是false,只更新找到的第一条记录，如果这个参数为true,就把按条件查出来多条记录全部更新。
* **writeConcern**:可选，抛出异常的级别。

### 实例

我们在集合 col 中插入如下数据：

>db.col.insert({

title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

})

接着我们通过 update() 方法来更新标题(title):

>db.col.update({'title':'MongoDB 教程'},{$set:{'title':'MongoDB'}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) # 输出信息

> db.col.find().pretty()

{

"\_id" : ObjectId("56064f89ade2f21f36b03136"),

"title" : "MongoDB",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100

}

>

可以看到标题(title)由原来的 "MongoDB 教程" 更新为了 "MongoDB"。

以上语句只会修改第一条发现的文档，如果你要修改多条相同的文档，则需要设置 multi 参数为 true。

>db.col.update({'title':'MongoDB 教程'},{$set:{'title':'MongoDB'}},{multi:true})

## save() 方法

save() 方法通过传入的文档来替换已有文档。语法格式如下：

db.collection.save(

<document>,

{

writeConcern: <document>

}

)

**参数说明：**

* **document**: 文档数据。
* **writeConcern**:可选，抛出异常的级别。

### 实例

以下实例中我们替换了 \_id 为 56064f89ade2f21f36b03136 的文档数据：

>db.col.save({

    "\_id" : ObjectId("56064f89ade2f21f36b03136"),

"title" : "MongoDB",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "Runoob",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"NoSQL"

],

"likes" : 110

})

替换成功后，我们可以通过 find() 命令来查看替换后的数据

>db.col.find().pretty()

{

"\_id" : ObjectId("56064f89ade2f21f36b03136"),

"title" : "MongoDB",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "Runoob",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"NoSQL"

],

"likes" : 110

}

>

## 更多实例

只更新第一条记录：

db.col.update( { "count" : { $gt : 1 } } , { $set : { "test2" : "OK"} } );

全部更新：

db.col.update( { "count" : { $gt : 3 } } , { $set : { "test2" : "OK"} },false,true );

只添加第一条：

db.col.update( { "count" : { $gt : 4 } } , { $set : { "test5" : "OK"} },true,false );

全部添加进去:

db.col.update( { "count" : { $gt : 5 } } , { $set : { "test5" : "OK"} },true,true );

全部更新：

db.col.update( { "count" : { $gt : 15 } } , { $inc : { "count" : 1} },false,true );

只更新第一条记录：

db.col.update( { "count" : { $gt : 10 } } , { $inc : { "count" : 1} },false,false );

# MongoDB 删除文档

在前面的几个章节中我们已经学习了MongoDB中如何为集合添加数据和更新数据。在本章节中我们将继续学习MongoDB集合的删除。

MongoDB remove()函数是用来移除集合中的数据。

MongoDB数据更新可以使用update()函数。在执行remove()函数前先执行find()命令来判断执行的条件是否正确，这是一个比较好的习惯。

## 语法

remove() 方法的基本语法格式如下所示：

db.collection.remove(

<query>,

<justOne>

)

如果你的 MongoDB 是 2.6 版本以后的，语法格式如下：

db.collection.remove(

<query>,

{

justOne: <boolean>,

writeConcern: <document>

}

)

**参数说明：**

* **query**:（可选）删除的文档的条件。
* **justOne**: （可选）如果设为 true 或 1，则只删除一个文档，如果不设置该参数，或使用默认值 false，则删除所有匹配条件的文档。
* **writeConcern**:（可选）抛出异常的级别。

## 实例

以下文档我们执行两次插入操作：

>db.col.insert({title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

})

使用 find() 函数查询数据：

> db.col.find()

{ "\_id" : ObjectId("56066169ade2f21f36b03137"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb", "database", "NoSQL" ], "likes" : 100 }

{ "\_id" : ObjectId("5606616dade2f21f36b03138"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb", "database", "NoSQL" ], "likes" : 100 }

接下来我们移除 title 为 'MongoDB 教程' 的文档：

>db.col.remove({'title':'MongoDB 教程'})

WriteResult({ "nRemoved" : 2 }) # 删除了两条数据

>db.col.find()

…… # 没有数据

如果你只想删除第一条找到的记录可以设置 justOne 为 1，如下所示：

>db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1)

如果你想删除所有数据，可以使用以下方式（类似常规 SQL 的 truncate 命令）：

>db.col.remove({})

>db.col.find()

>

# MongoDB 查询文档

MongoDB 查询文档使用 find() 方法。

find() 方法以非结构化的方式来显示所有文档。

## 语法

MongoDB 查询数据的语法格式如下：

db.collection.find(query, projection)

* **query** ：可选，使用查询操作符指定查询条件
* **projection** ：可选，使用投影操作符指定返回的键。查询时返回文档中所有键值， 只需省略该参数即可（默认省略）。

如果你需要以易读的方式来读取数据，可以使用 pretty() 方法，语法格式如下：

>db.col.find().pretty()

pretty() 方法以格式化的方式来显示所有文档。

## 实例

以下实例我们查询了集合 col 中的数据：

> db.col.find().pretty()

{

"\_id" : ObjectId("56063f17ade2f21f36b03133"),

"title" : "MongoDB 教程",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100

}

除了 find() 方法之外，还有一个 findOne() 方法，它只返回一个文档。

## MongoDB 与 RDBMS Where 语句比较

如果你熟悉常规的 SQL 数据，通过下表可以更好的理解 MongoDB 的条件语句查询：

| **操作** | **格式** | **范例** | **RDBMS中的类似语句** |
| --- | --- | --- | --- |
| 等于 | {<key>:<value>} | db.col.find({"by":"菜鸟教程"}).pretty() | where by = '菜鸟教程' |
| 小于 | {<key>:{$lt:<value>}} | db.col.find({"likes":{$lt:50}}).pretty() | where likes < 50 |
| 小于或等于 | {<key>:{$lte:<value>}} | db.col.find({"likes":{$lte:50}}).pretty() | where likes <= 50 |
| 大于 | {<key>:{$gt:<value>}} | db.col.find({"likes":{$gt:50}}).pretty() | where likes > 50 |
| 大于或等于 | {<key>:{$gte:<value>}} | db.col.find({"likes":{$gte:50}}).pretty() | where likes >= 50 |
| 不等于 | {<key>:{$ne:<value>}} | db.col.find({"likes":{$ne:50}}).pretty() | where likes != 50 |

## MongoDB AND 条件

MongoDB 的 find() 方法可以传入多个键(key)，每个键(key)以逗号隔开，即常规 SQL 的 AND 条件。

语法格式如下：

>db.col.find({key1:value1, key2:value2}).pretty()

### 实例

以下实例通过 **by** 和 **title** 键来查询 **菜鸟教程** 中 **MongoDB 教程** 的数据

> db.col.find({"by":"菜鸟教程", "title":"MongoDB 教程"}).pretty()

{

"\_id" : ObjectId("56063f17ade2f21f36b03133"),

"title" : "MongoDB 教程",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100

}

以上实例中类似于 WHERE 语句：**WHERE by='菜鸟教程' AND title='MongoDB 教程'**

## MongoDB OR 条件

MongoDB OR 条件语句使用了关键字 **$or**,语法格式如下：

>db.col.find(

{

$or: [

     {key1: value1}, {key2:value2}

]

}

).pretty()

### 实例

以下实例中，我们演示了查询键 **by** 值为 菜鸟教程 或键 **title** 值为 **MongoDB 教程** 的文档。

>db.col.find({$or:[{"by":"菜鸟教程"},{"title": "MongoDB 教程"}]}).pretty()

{

"\_id" : ObjectId("56063f17ade2f21f36b03133"),

"title" : "MongoDB 教程",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100

}

>

## AND 和 OR 联合使用

以下实例演示了 AND 和 OR 联合使用，类似常规 SQL 语句为： **'where likes>50 AND (by = '菜鸟教程' OR title = 'MongoDB 教程')'**

>db.col.find({"likes": {$gt:50}, $or: [{"by": "菜鸟教程"},{"title": "MongoDB 教程"}]}).pretty()

{

"\_id" : ObjectId("56063f17ade2f21f36b03133"),

"title" : "MongoDB 教程",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100

}

# MongoDB 条件操作符

## 描述

条件操作符用于比较两个表达式并从mongoDB集合中获取数据。

在本章节中，我们将讨论如何在MongoDB中使用条件操作符。

MongoDB中条件操作符有：

* (>) 大于 - $gt
* (<) 小于 - $lt
* (>=) 大于等于 - $gte
* (<= ) 小于等于 - $lte

**我们使用的数据库名称为"runoob" 我们的集合名称为"col"，以下为我们插入的数据。**

为了方便测试，我们可以先使用以下命令清空集合 "col" 的数据：

db.col.remove({})

插入以下数据

>db.col.insert({

title: 'PHP 教程',

description: 'PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['php'],

likes: 200

})

>db.col.insert({title: 'Java 教程',

description: 'Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['java'],

likes: 150

})

>db.col.insert({title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['mongodb'],

likes: 100

})

使用find()命令查看数据：

> db.col.find()

{ "\_id" : ObjectId("56066542ade2f21f36b0313a"), "title" : "PHP 教程", "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "php" ], "likes" : 200 }

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

## MongoDB (>) 大于操作符 - $gt

如果你想获取 "col" 集合中 "likes" 大于 100 的数据，你可以使用以下命令：

db.col.find({likes : {$gt : 100}})

类似于SQL语句：

Select \* from col where likes > 100;

输出结果：

> db.col.find({likes : {$gt : 100}})

{ "\_id" : ObjectId("56066542ade2f21f36b0313a"), "title" : "PHP 教程", "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "php" ], "likes" : 200 }

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

>

## MongoDB（>=）大于等于操作符 - $gte

如果你想获取"col"集合中 "likes" 大于等于 100 的数据，你可以使用以下命令：

db.col.find({likes : {$gte : 100}})

类似于SQL语句：

Select \* from col where likes >=100;

输出结果：

> db.col.find({likes : {$gte : 100}})

{ "\_id" : ObjectId("56066542ade2f21f36b0313a"), "title" : "PHP 教程", "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "php" ], "likes" : 200 }

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

>

## MongoDB (<) 小于操作符 - $lt

如果你想获取"col"集合中 "likes" 小于 150 的数据，你可以使用以下命令：

db.col.find({likes : {$lt : 150}})

类似于SQL语句：

Select \* from col where likes < 150;

输出结果：

> db.col.find({likes : {$lt : 150}})

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

## MongoDB (<=) 小于操作符 - $lte

如果你想获取"col"集合中 "likes" 小于等于 150 的数据，你可以使用以下命令：

db.col.find({likes : {$lte : 150}})

类似于SQL语句：

Select \* from col where likes <= 150;

输出结果：

> db.col.find({likes : {$lte : 150}})

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

## MongoDB 使用 (<) 和 (>) 查询 - $lt 和 $gt

如果你想获取"col"集合中 "likes" 大于100，小于 200 的数据，你可以使用以下命令：

db.col.find({likes : {$lt :200, $gt : 100}})

类似于SQL语句：

Select \* from col where likes>100 AND likes<200;

输出结果：

> db.col.find({likes : {$lt :200, $gt : 100}})

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

>

## **模糊查询**

查询 title 包含"教"字的文档：

db.col.find({title:/教/})

查询 title 字段以"教"字开头的文档：

db.col.find({title:/^教/})

查询 titl e字段以"教"字结尾的文档：

db.col.find({title:/教$/})

# MongoDB $type 操作符

## 描述

在本章节中，我们将继续讨论MongoDB中条件操作符 $type。

$type操作符是基于BSON类型来检索集合中匹配的数据类型，并返回结果。

MongoDB 中可以使用的类型如下表所示：

|  |  |  |
| --- | --- | --- |
| **类型** | **数字** | **备注** |
| Double | 1 |  |
| String | 2 |  |
| Object | 3 |  |
| Array | 4 |  |
| Binary data | 5 |  |
| Undefined | 6 | 已废弃。 |
| Object id | 7 |  |
| Boolean | 8 |  |
| Date | 9 |  |
| Null | 10 |  |
| Regular Expression | 11 |  |
| JavaScript | 13 |  |
| Symbol | 14 |  |
| JavaScript (with scope) | 15 |  |
| 32-bit integer | 16 |  |
| Timestamp | 17 |  |
| 64-bit integer | 18 |  |
| Min key | 255 | Query with -1. |
| Max key | 127 |  |

**我们使用的数据库名称为"runoob" 我们的集合名称为"col"，以下为我们插入的数据。**

简单的集合"col"：

>db.col.insert({

title: 'PHP 教程',

description: 'PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['php'],

likes: 200

})

>db.col.insert({title: 'Java 教程',

description: 'Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['java'],

likes: 150

})

>db.col.insert({title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['mongodb'],

likes: 100

})

使用find()命令查看数据：

> db.col.find()

{ "\_id" : ObjectId("56066542ade2f21f36b0313a"), "title" : "PHP 教程", "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "php" ], "likes" : 200 }

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

## MongoDB 操作符 - $type 实例

如果想获取 "col" 集合中 title 为 String 的数据，你可以使用以下命令：

db.col.find({"title" : {$type : 2}})

或

db.col.find({"title" : {$type : 'string'}})

输出结果为：

{ "\_id" : ObjectId("56066542ade2f21f36b0313a"), "title" : "PHP 教程", "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "php" ], "likes" : 200 }

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

# MongoDB Limit与Skip方法

## MongoDB Limit() 方法

如果你需要在MongoDB中读取指定数量的数据记录，可以使用MongoDB的Limit方法，limit()方法接受一个数字参数，该参数指定从MongoDB中读取的记录条数。

### 语法

limit()方法基本语法如下所示：

>db.COLLECTION\_NAME.find().limit(NUMBER)

### 实例

集合 col 中的数据如下：

{ "\_id" : ObjectId("56066542ade2f21f36b0313a"), "title" : "PHP 教程", "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "php" ], "likes" : 200 }

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

以下实例为显示查询文档中的两条记录：

> db.col.find({},{"title":1,\_id:0}).limit(2)

{ "title" : "PHP 教程" }

{ "title" : "Java 教程" }

>

注：如果你们没有指定limit()方法中的参数则显示集合中的所有数据。

## MongoDB Skip() 方法

我们除了可以使用limit()方法来读取指定数量的数据外，还可以使用skip()方法来跳过指定数量的数据，skip方法同样接受一个数字参数作为跳过的记录条数。

### 语法

skip() 方法脚本语法格式如下：

>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

### 实例

以下实例只会显示第二条文档数据

>db.col.find({},{"title":1,\_id:0}).limit(1).skip(1)

{ "title" : "Java 教程" }

>

**注:**skip()方法默认参数为 0 。

# MongoDB 排序

## MongoDB sort() 方法

在 MongoDB 中使用 sort() 方法对数据进行排序，sort() 方法可以通过参数指定排序的字段，并使用 1 和 -1 来指定排序的方式，其中 1 为升序排列，而 -1 是用于降序排列。

### 语法

sort()方法基本语法如下所示：

>db.COLLECTION\_NAME.find().sort({KEY:1})

### 实例

col 集合中的数据如下：

{ "\_id" : ObjectId("56066542ade2f21f36b0313a"), "title" : "PHP 教程", "description" : "PHP 是一种创建动态交互性站点的强有力的服务器端脚本语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "php" ], "likes" : 200 }

{ "\_id" : ObjectId("56066549ade2f21f36b0313b"), "title" : "Java 教程", "description" : "Java 是由Sun Microsystems公司于1995年5月推出的高级程序设计语言。", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "java" ], "likes" : 150 }

{ "\_id" : ObjectId("5606654fade2f21f36b0313c"), "title" : "MongoDB 教程", "description" : "MongoDB 是一个 Nosql 数据库", "by" : "菜鸟教程", "url" : "http://www.runoob.com", "tags" : [ "mongodb" ], "likes" : 100 }

以下实例演示了 col 集合中的数据按字段 likes 的降序排列：

>db.col.find({},{"title":1,\_id:0}).sort({"likes":-1})

{ "title" : "PHP 教程" }

{ "title" : "Java 教程" }

{ "title" : "MongoDB 教程" }

>

skip(), limilt(), sort()三个放在一起执行的时候，执行的顺序是先 sort(), 然后是 skip()，最后是显示的 limit()。

、

# MongoDB 索引

索引通常能够极大的提高查询的效率，如果没有索引，MongoDB在读取数据时必须扫描集合中的每个文件并选取那些符合查询条件的记录。

这种扫描全集合的查询效率是非常低的，特别在处理大量的数据时，查询可以要花费几十秒甚至几分钟，这对网站的性能是非常致命的。

索引是特殊的数据结构，索引存储在一个易于遍历读取的数据集合中，索引是对数据库表中一列或多列的值进行排序的一种结构

## createIndex() 方法

MongoDB使用 createIndex() 方法来创建索引。

*注意在 3.0.0 版本前创建索引方法为 db.collection.ensureIndex()，之后的版本使用了 db.collection.createIndex() 方法，ensureIndex() 还能用，但只是 createIndex() 的别名。*

### 语法

createIndex()方法基本语法格式如下所示：

>db.collection.createIndex(keys, options)

语法中 Key 值为你要创建的索引字段，1 为指定按升序创建索引，如果你想按降序来创建索引指定为 -1 即可。

### 实例

>db.col.createIndex({"title":1})

createIndex() 方法中你也可以设置使用多个字段创建索引（关系型数据库中称作复合索引）。

>db.col.createIndex({"title":1,"description":-1})

createIndex() 接收可选参数，可选参数列表如下：

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| background | Boolean | 建索引过程会阻塞其它数据库操作，background可指定以后台方式创建索引，即增加 "background" 可选参数。 "background" 默认值为**false**。 |
| unique | Boolean | 建立的索引是否唯一。指定为true创建唯一索引。默认值为**false**. |
| name | string | 索引的名称。如果未指定，MongoDB的通过连接索引的字段名和排序顺序生成一个索引名称。 |
| dropDups | Boolean | **3.0+版本已废弃。**在建立唯一索引时是否删除重复记录,指定 true 创建唯一索引。默认值为 **false**. |
| sparse | Boolean | 对文档中不存在的字段数据不启用索引；这个参数需要特别注意，如果设置为true的话，在索引字段中不会查询出不包含对应字段的文档.。默认值为 **false**. |
| expireAfterSeconds | integer | 指定一个以秒为单位的数值，完成 TTL设定，设定集合的生存时间。 |
| v | index version | 索引的版本号。默认的索引版本取决于mongod创建索引时运行的版本。 |
| weights | document | 索引权重值，数值在 1 到 99,999 之间，表示该索引相对于其他索引字段的得分权重。 |
| default\_language | string | 对于文本索引，该参数决定了停用词及词干和词器的规则的列表。 默认为英语 |
| language\_override | string | 对于文本索引，该参数指定了包含在文档中的字段名，语言覆盖默认的language，默认值为 language. |

### 实例

在后台创建索引：

db.values.createIndex({open: 1, close: 1}, {background: true})

通过在创建索引时加 background:true 的选项，让创建工作在后台执行

# 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(\*)。

## aggregate() 方法

MongoDB中聚合的方法使用aggregate()。

### 语法

aggregate() 方法的基本语法格式如下所示：

>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

### 实例

集合中的数据如下：

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by\_user: 'runoob.com',

url: 'http://www.runoob.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

},

{

\_id: ObjectId(7df78ad8902d)

title: 'NoSQL Overview',

description: 'No sql database is very fast',

by\_user: 'runoob.com',

url: 'http://www.runoob.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 10

},

{

\_id: ObjectId(7df78ad8902e)

title: 'Neo4j Overview',

description: 'Neo4j is no sql database',

by\_user: 'Neo4j',

url: 'http://www.neo4j.com',

tags: ['neo4j', 'database', 'NoSQL'],

likes: 750

},

现在我们通过以上集合计算每个作者所写的文章数，使用aggregate()计算结果如下：

> db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}])

{

"result" : [

{

"\_id" : "runoob.com",

"num\_tutorial" : 2

},

{

"\_id" : "Neo4j",

"num\_tutorial" : 1

}

],

"ok" : 1

}

>

以上实例类似sql语句：

select by\_user, count(\*) from mycol group by by\_user

在上面的例子中，我们通过字段 by\_user 字段对数据进行分组，并计算 by\_user 字段相同值的总和。

下表展示了一些聚合的表达式:

|  |  |  |
| --- | --- | --- |
| **表达式** | **描述** | **实例** |
| $sum | 计算总和。 | db.mycol.aggregate([{$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 : "$by\_user", num\_tutorial : {$max : "$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"}}}]) |

## 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。

### 管道操作符实例

1、$project实例

db.article.aggregate(

{ $project : {

title : 1 ,

author : 1 ,

}}

);

这样的话结果中就只还有\_id,tilte和author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.article.aggregate(

{ $project : {

\_id : 0 ,

title : 1 ,

author : 1

}});

2.$match实例

db.articles.aggregate( [

{ $match : { score : { $gt : 70, $lte : 90 } } },

{ $group: { \_id: null, count: { $sum: 1 } } }

] );

$match用于获取分数大于70小于或等于90记录，然后将符合条件的记录送到下一阶段$group管道操作符进行处理。

3.$skip实例

db.article.aggregate(

{ $skip : 5 });

经过$skip管道操作符处理后，前五个文档被"过滤"掉。