MongoDB 命令

帮助命令

help()

查看当前MongoDB 版本

version()

创建数据库 & 切换数据库

use db_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。

查看当前数据库

db

删除数据库

db.dropDatabase() 删除当前数据库

删除集合

db.collection.drop()

插入文档

3.2 版本后还有以下几种语法可用于插入文档:

db.collection.insert({....});向指定集合中插入一条文档数据

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

删除文档

remove()

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。

remove() 方法的基本语法格式:

```
db.collection.remove(
  <query>,
  <justOne>
)
```

MongoDB 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，默认查全部
- **justOne :（可选）如果设为 true 或 1，则只删除一个文档，默认false**
- writeConcern :（可选）抛出异常的级别。

更新文档

MongoDB 使用 **update()** 和 **save()** 方法来更新集合中的文档。

update() 方法更新文档。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
- **update** : update的对象和一些更新的操作符（如$,$inc...）等
- **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

save() 方法通过传入的文档来替换已有文档。语法格式如下：

根绝 _id 来更新文档

```
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。

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。

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。

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

大于，小于，等于及其组合

操作	格式	范例	RDBMS中的类似语句
等于	{<key>:<value>}	db.col.find({"name":"jiang"}).pretty()	where name = 'jiang'
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

and 条件

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

or 条件

```
db.col.find(
  {
    $or: [
    {key1: value1}, {key2:value2}
    ]
  }
).pretty()
```

and & or

eg:

```
db.user.find(
    {
        age:25,
        $or:[
            {gender:1},
            {gender:2}
        ]
    }
)
```

$in 条件

db.user.find({age:{$in:[24,25,26]}}).pretty();

数据类型匹配查询 $type

db.user.find({gender:{$type:1}}).pretty();

分页查询，排序

skip() ， limit()

db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:1});

db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:-1});

db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:-1}).skip(2);

db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:-1}).skip(2).limit(3);

升降序用 1 （升）， -1（降） 表示

```
> db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:1});
{ "uid" : 1001, "salary" : 4588 }
{ "uid" : 1002, "salary" : 4588 }
{ "uid" : 1003, "salary" : 4588 }
{ "uid" : 1008, "salary" : 5896 }
{ "uid" : 1004, "salary" : 8888 }
{ "uid" : 1005, "salary" : 8888 }
{ "uid" : 1006, "salary" : 8888 }
{ "uid" : 1007, "salary" : 8888 }
> db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:-1});
{ "uid" : 1004, "salary" : 8888 }
{ "uid" : 1005, "salary" : 8888 }
{ "uid" : 1006, "salary" : 8888 }
{ "uid" : 1007, "salary" : 8888 }
{ "uid" : 1008, "salary" : 5896 }
{ "uid" : 1001, "salary" : 4588 }
{ "uid" : 1002, "salary" : 4588 }
{ "uid" : 1003, "salary" : 4588 }
> db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:-1}).skip(2);
{ "uid" : 1006, "salary" : 8888 }
{ "uid" : 1007, "salary" : 8888 }
{ "uid" : 1008, "salary" : 5896 }
{ "uid" : 1001, "salary" : 4588 }
{ "uid" : 1002, "salary" : 4588 }
{ "uid" : 1003, "salary" : 4588 }
> db.user.find({},{_id:0, uid:1, usernmae:1,salary:1}).sort({salary:-1}).skip(2).limit(3);
{ "uid" : 1006, "salary" : 8888 }
{ "uid" : 1007, "salary" : 8888 }
{ "uid" : 1008, "salary" : 5896 }
```

当查询时同时使用sort,skip,limit，无论位置先后，最先执行顺序 sort再skip再limit。

Mongodb 修改器

1.$inc

修改器$inc可以对文档的某个值为数字型（只能为满足要求的数字）的键进行增减的操作

```
db.user.update(
 {
  uid:1001
 },
 {
  $inc:{age:1}
 },
 {
  multi:false
 }
)
```

2.$set

用来指定一个键并更新键值，若键不存在并创建

```
db.user.update(
 {
  uid:1001
 },
 {
  $inc:{age:1},
  $set:{password:"223314"}
 },
 {
  multi:true
 }
)
```

3.$unset

主要是用来删除键

使用修改器$unset时，不论对目标键使用1、0、-1或者具体的字符串等都是可以删除该目标键

```
 db.user.update(
 {
  uid:1001
 },
 {
  $set:{addkey:"1"}
 },
 {
  multi:false
 }
)

 db.user.update(
 {
  uid:1001
 },
 {
  $unset:{addkey:"del"}
 },
 {
  multi:true
 }
)
```

4.数组修改器--$push

```
db.user.update(
 {
  uid:{$lt:1004}
 },
 {
  $push:{"hobbies":"music"}
 },
 {
  upsert:false,
  multi:true
 }
)

db.user.update(
 {
```

```
 uid:{$lt:1004}
},
{
 $push:{"hobbies":
  {
   $each:["dog", "cat", "run", "basketball"]
  }
 }
},
{
 upsert:false,
 multi:true
}
)
```

$push--

向文档的某个数组类型的键添加一个数组元素，

不过滤重复的数据

添加时键存在，要求键值类型必须是数组；

键不存在，则创建数组类型的键

5.数组修改器--$addToSet

给数组类型键值添加一个元素时，避免在数组中产生重复数据

```
db.user.update(
{
 uid:1004
},
{
 $addToSet:{"hobbies":"music"}
},
{
 upsert:false,
 multi:true
}
)
```

6. 数组修改器 $ne

$ne也是用来操作数组的修改器，在查询文档中，如果一个值不在数组里面就把他加进去，如果在不添加。

7. $each

$each数组修改器可以一次添加多个不同的值

```
db.user.update(
{
 uid:{$lt:1004}
},
{
 $push:{"hobbies":
  {
   $each:["dog", "cat", "run", "basketball"]
  }
 }
},
{
 upsert:false,
 multi:true
}
)
```

8. 数组修改器 $pop

$pop修改器主要于从数组中删除元素，他可以从数组中的任何一端删除元素

{$pop:{key:1}} 从数组末尾删除一个元素

{$pop:{key:-1}} 从数组头部删除一个元素

9. 数组修改器 $pull

$pull可以基于特定条件来删除元素

10. 数组下标操作 和 数组定位修改器

db.names.update({_id:24},{$set:{"value.users.0.username":"huanhuanlan"}})

必须使用引号 “”

```
db.names.update(
    {
        _id:25,
        "value.users.uid":1002
    },
    {
        $set:
            {
                "value.users.$.username":"zhangluhui"
            }
    }
)
```

MongoDB 监控

MongoDB中提供了mongostat 和 mongotop 两个命令来监控MongoDB的运行情况。

```
[root@localhost bin]#
[root@localhost bin]# ./mongostat
insert query update delete getmore command dirty  used flushes vsize  res qrw arw net_in net_out conn                time
    *0    *0     *0     *0       1     2|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   210b   46.4k    9 Aug 11 14:06:07.620
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.9k    9 Aug 11 14:06:08.620
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.8k    9 Aug 11 14:06:09.620
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.9k    9 Aug 11 14:06:10.621
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.9k    9 Aug 11 14:06:11.621
    *0    *0     *0     *0       1     2|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   208b   46.0k    9 Aug 11 14:06:12.618
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.8k    9 Aug 11 14:06:13.618
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.8k    9 Aug 11 14:06:14.619
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.9k    9 Aug 11 14:06:15.619
    *0    *0     *0     *0       0     1|0  0.1% 79.9%       0 1.15G 297M 0|0 0|0   207b   45.9k    9 Aug 11 14:06:16.621
^C2017-08-11T14:06:17.007+0800  signal 'interrupt' received; forcefully terminating
[root@localhost bin]#
[root@localhost bin]# ./mongotop
2017-08-11T14:06:22.516+0800    connected to: 127.0.0.1

                    ns    total    read    write    2017-08-11T14:06:23+08:00
  admin.system.roles      0ms      0ms      0ms
  admin.system.users      0ms      0ms      0ms
admin.system.version      0ms      0ms      0ms
           local.me       0ms      0ms      0ms
   local.oplog.$main      0ms      0ms      0ms
      local.oplog.rs      0ms      0ms      0ms
   local.startup_log      0ms      0ms      0ms
local.system.replset      0ms      0ms      0ms
        river.person      0ms      0ms      0ms
          river.user      0ms      0ms      0ms

                    ns    total    read    write    2017-08-11T14:06:24+08:00
  admin.system.roles      0ms      0ms      0ms
  admin.system.users      0ms      0ms      0ms
admin.system.version      0ms      0ms      0ms
           local.me       0ms      0ms      0ms
   local.oplog.$main      0ms      0ms      0ms
      local.oplog.rs      0ms      0ms      0ms
   local.startup_log      0ms      0ms      0ms
local.system.replset      0ms      0ms      0ms
        river.person      0ms      0ms      0ms
          river.user      0ms      0ms      0ms

^C2017-08-11T14:06:24.831+0800  signal 'interrupt' received; forcefully terminating
[root@localhost bin]#
```