MongoDB 数据库

1 简介

1.1 数据库

数据库(Database)是按照数据结构来组织、存储和管理数据的仓库。

分类

- 关系型数据库（SQL 结构化查询语言)
- 非关系型数据库(noSQL)

常见数据库

- MySQL （开源免费）
- SQLServer （微软公司）
- Oracle （甲骨文公司）
- DB2 （IBM公司）
- PostgreSQL （开源 免费）
- SQLite（开源 免费）（小巧、快速、潜入）
- MongoDB （NoSQL）
- Redis （NoSQL）
- Memcache （NoSQL）

数据库相关该概念

- DB 数据库
- DBA 数据库管理员
- DBMS 数据库管理系统
- DBS 数据库系统

1.2 NoSQL

NoSQL，指的是非关系型的数据库。NoSQL有时也称作Not Only SQL的缩写，是对不同于传统的关系型数据库的数据库管理系统的统称。

NoSQL用于超大规模数据的存储。（例如谷歌或Facebook每天为他们的用户收集万亿比特的数据）。这些类型的数据存储不需要固定的模式，无需多余操作就可以横向扩展。

优缺点：

优点:

- 高可扩展性
- 分布式计算
- 低成本
- 架构的灵活性，半结构化数据
- 没有复杂的关系

缺点

- 没有标准化
- 有限的查询功能（到目前为止）
- 最终一致是不直观的程序

NoSQL分类

类型	代表	特点
列存储	Hbase、Cassandra、Hypertable	顾名思义，是按列存储数据的。最大的特点是方便存储结构化和半结构化数据，方便做数据压缩，对针对某一列或者某几列的查询有非常大的IO优势。
文档存储	MongoDB、CouchDB	文档存储一般用类似json的格式存储，存储的内容是文档型的。这样也就有有机会对某些字段建立索引，实现关系数据库的某些功能。
key-value存储	Tokyo Cabinet / Tyrant、Berkeley DB、MemcacheDB、Redis	可以通过key快速查询到其value。一般来说，存储不管value的格式，照单全收。（Redis包含了其他功能）
图存储	Neo4J、FlockDB	图形关系的最佳存储。使用传统关系数据库来解决的话性能低下，而且设计使用不方便。
对象存储	db4o、Versant	通过类似面向对象语言的语法操作数据库，通过对象的方式存取数据。
xml数据库	Berkeley DB XML、BaseX	高效的存储XML数据，并支持XML的内部查询语法，比如XQuery,Xpath。

1.3 MongoDB 简介

- MongoDB是开源，高性能的NoSQL数据库；支持索引、集群、复制和故障转移、各种语言的驱动程序丰富；高伸缩性；
- MongoDB 是一个基于分布式文件存储的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。
- MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。
- shell命令操作语法和JavaScript很类似，其实控制台底层的查询语句都是用JavaScript脚本完成操作的。
- 相关网址

> 官网地址：http://www.mongodb.org/ github: https://github.com/mongodb/ API Docs：http://docs.mongodb.org/manual/ nodejs驱动：https://github.com/mongodb/node-mongodb-native

2 安装

2.1 windows 下安装

① 下载

- 官方下载列表:https://www.mongodb.org/dl/win32/
- 注意下载合适的版本，大部分版本是x64，x86请下载 win32-i386版本
- 下载 *-signed.msi 版本，可以双击安装

② 安装

双击安装，下一步即可

我安装的版本 mongodb-win32-i386-v3.0-latest-signed.msi

③ 添加环境变量

把 mongodb的安装的路径 添加到path中

④启动

- **创建数据目录**

MongoDB将数据目录存储在 db 目录下。但是这个数据目录不会主动创建，我们在安装完成后需要创建它。

我的目录是C:\data\db，其中db文件夹也需要创建，岂不会自动生成

- **运行命令**

```
mongod.exe --dbpath C:\data\db
```

⑤ 配置

像上面那样启动太麻烦，所以我配置成Windows服务

为日志文件和配置文件创建目录

配置文件目录：C:\data\config\mongod.cfg

日志文件目录：C:\data\dblog\

配置 mongod.cfg

第一种写法

```
##数据库目录##
dbpath=C:\data\db
##日志输出文件##
logpath=C:\data\log\db.log
```

第二种写法 (推荐)

```
storage:
    journal:
        enabled: true
    dbPath: c:\data\db
systemLog:
    destination: file
    path: c:\data\dblog\mongod.log
    logAppend: true
net:
    port: 27017
```

以下是对第二种配置方法的解释：

```
storage数据存储配置：
    journal:
        enabled: true
        #描述：是否开启journal日志持久存储，journal日志用来数据恢复，是 mongod最基础的特性，通常用于故障恢复。64位系统默认为true，32位默认为false，建议开启，仅对mongod进程有效。
    dbPath: D:\data\db
    #描述：mongodb数据存储位置。


ystemLog系统日志配置：
```

```
    destination: file:
    #描述：日志输出目的地，可以指定为“ file”或者“syslog”，表述输出到日志文件，如果不指定，则会输出到标准输出中（standard output）。
    path: D:\data\dblog\mongod.log
    #描述：日志的路径，其中D:\data\dblog这个路径是我自己创建的。mongod.log启动后，会自动生成。
    logAppend: true:
    #描述：如果为true，当mongod/mongos重启后，将在现有日志的尾部继续添加日志。否则，将会备份当前日志文件，然后创建一个新的日志文件；默认为false。

net:
    port: 27017
    #描述：mongod/mongos侦听端口，默认为27017；不过因为mongodb有2种典型的架构模式：replica set和sharding，如果开发者在一个节点上部署多个mongod实例，需要注意修改此端口以避免冲突。
```

注意，第二种配置方式，格式要求严格`:`后面必须有空格， 缩进必须是4个空格，不能用Tab代替

配置文件选项文档 https://docs.mongodb.com/manual/reference/configuration-options/

⑥ 安装服务

使用管理员方式，打开CMD，运行命令

```
mongod.exe --config "c:\data\config\mongod.cfg" --install
```

⑦ 启动服务

第一种 图形操作

依次打开 “控制面板\所有控制面板项\管理工具\服务”，找到"MongoDB"

第二种 命令行操作

打开CMD

```
net start MongoDB
```

⑧ 关闭服务

```
net stop MongoDB
```

⑨ 删除Windows服务

```
mongod.exe --config "c:\data\config\mongod.cfg" --remove
```

2.2 MacOS 下安装

① 安装brew

打开终端，执行

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

brew官网： https://brew.sh/

② 安装MongoDB

```
brew install mongodb #安装
berw uninstall mongodb #卸载
```

③ 启动/关闭 MongoDB

```
brew services start mongodb  #启动 MongoDB服务
brew services stop mongodb #关闭MongoDB服务
brew services restart mongodb #重启MongoDB服务
```

④ 配置文件

默认配置文件位置：`/usr/local/etc/mongod.conf`

配置文件选项文档 https://docs.mongodb.com/manual/reference/configuration-options/

3 Mongo 客户端工具和基本概念

3.1 GUI客户端工具

- robo3T https://robomongo.org/
- Nosqlclient https://github.com/nosqlclient/nosqlclient
- NoSQL Manager for MongoDB https://www.mongodbmanager.com/

3.2 MongoDB 相关概念

- database 数据库
- collection 集合
- document 文档
- field 字段/属性
- index 索引
- primary key 主键

MongoDB 概念	关系型数据库概念
database 数据库	database 数据库
collection 集合	table 表格
document 文档	row 行/记录
field 字段	column 列/字段
index 索引	index 索引
primary key 主键	primary key 主键

3.3 数据库

- 一个mongodb中可以建立多个数据库。
- 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。

- 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(默认存在的数据库)

  > **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

3.4 集合

- 集合就是 MongoDB 文档组，类似于 关系型数据库 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。
- 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$。

- Capped collections 就是固定大小的集合 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)

  > 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 。
  >
  > 数据库不允许进行删除。使用drop()方法删除collection所有的行。
  >
  > 注意: 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。

3.5 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。
- 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)。

3.6 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。
Arrays	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。

4 MongoDB 操作

4.1 连接MongoDB服务

连接

```
mongo host:port/database -u user -p password
```

- host MongoDB服务器的主机名(地址)，默认 locahost
- port MongoDB服务器端口，默认27017
- database 连接后打开的数据库，默认test

退出

```
exit
```

4.2 数据库操作

帮助

```
help  #查看帮助
db.help()  #db对象的方法 属性
```

切换/创建 数据库

```
use dbname
```

当创建一个集合(collection)的时候会自动创建当前数据库

查看所有数据库

```
show dbs;
db.getCollectionName(); # 返回数组
```

查看当前使员的数据库

```
db.getName();
db;
```

查看数据库状态

```
db.stats()  #当前数据库的状态
```

删除当前的数据库

```
db.dropDatabase()
```

获取数据库相关信息

```
db.versions();  #数据库版本
db.getMongo();  #当前数据库服务器地址
```

4.3 集合操作

创建集合

```
db.createCollection(“collName”, options); #options是对象，指定集合特性 可以说省略
```

字段	类型	描述
capped	Boolean	（可选）如果为true，则启用封顶集合。封顶集合是固定大小的集合，会自动覆盖最早的条目，当它达到其最大大小。如果指定true，则需要也指定尺寸参数。
autoIndexID	Boolean	（可选）如果为true，自动创建索引_id字段的默认值是false。
size	number	（可选）指定最大大小字节封顶集合。如果封顶如果是true，那么你还需要指定这个字段。
max	number	（可选）指定封顶集合允许在文件的最大数量。

查看帮助

```
db.collName.help();  # db.collName是对象，查看该对象下的方法
```

集合状态

```
db.printCollectionStats(); # 数据库下所有集合的状态
db.collName.stats();  #指定集合的状态
```

集合重命名

```
db.collName.renameCollection('newName');
```

删除集合

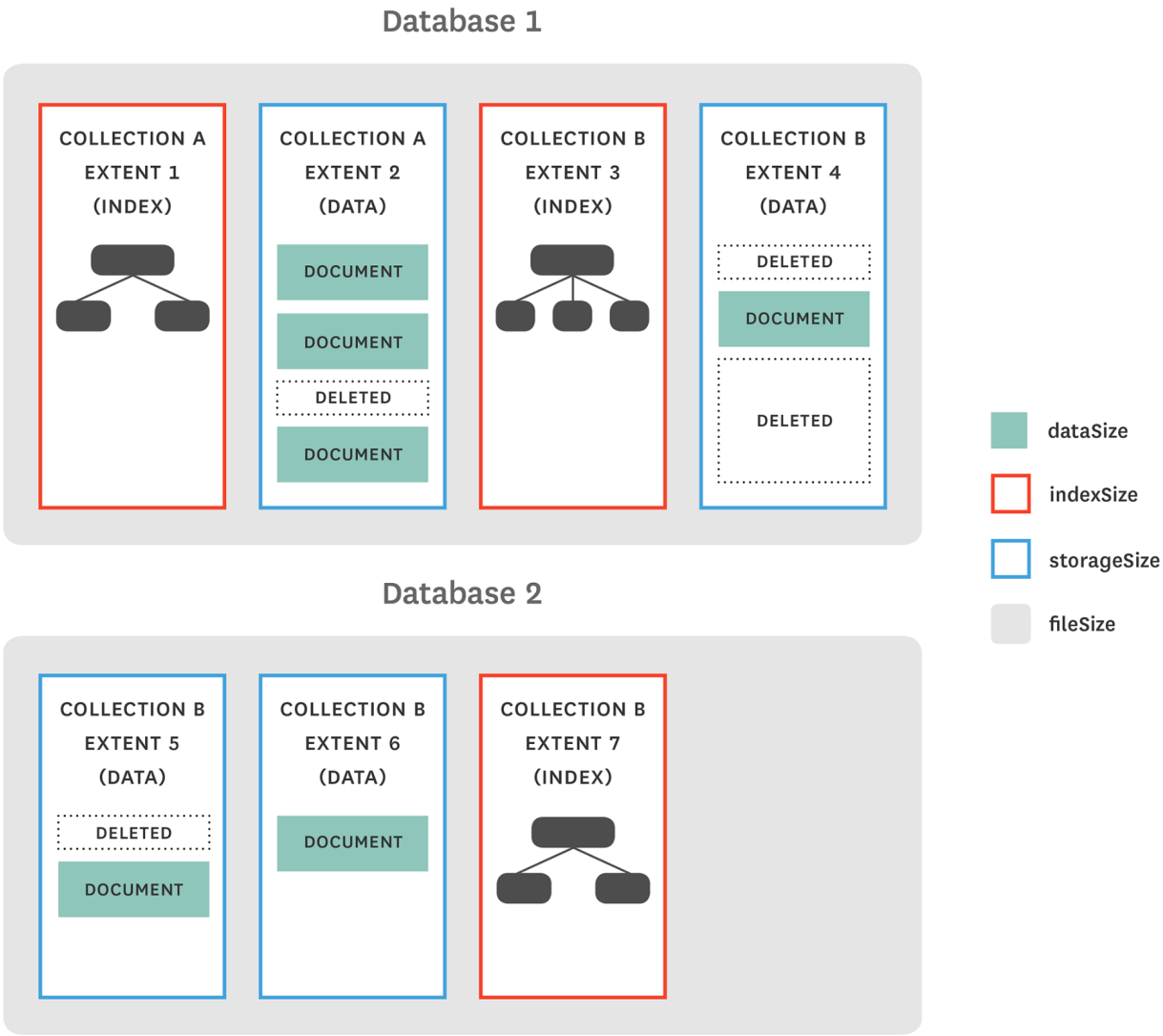
```
db.collName.drop();
```

得到当前集合所在的db

```
db.collName.getDB()
```

空间大小

```
db.collName.dataSize();  #数据空间大小
db.collName.storageSize();  #集合储存空间大小
db.collName.totalSize();  #集合总大小
```



4.4 集合数据(文档)增删改

4.4.1 插入文档

① 方法

- db.collName.insertOne()
- db.collNaem.inserMany()
- db.collName.insert()
- db.collName.save()

② 语法

```
db.collection.insert(
   <document or array of documents>,
   {
     writeConcern: <document>,
     ordered: <boolean>
   }
)
```

- writeConcern 用于控制写入安全的级别 { w: , j: , wtimeout: }

> w : 该选项要求确认操作已经传播到指定数量的mongod实例或指定标签的mongod实例
>
> w可选的的值
>
> w:1(应答式写入) 要求确认操作已经传播到指定的单个mongod实例或副本集主实例(缺省为1)
>
> w:0(非应答式写入) 不返回任何响应，所以无法知道写入是否成功 但是对于尝试向已关闭的套接字写入或者网络故障会返回异常信息
>
> w:>1(用于副本集环境) 该值用于设定写入节点的数目，包括主节点
>
> j : 该选项要求确认写操作已经写入journal日志之后应答客户端(需要开启journal功能)
>
> 则在意外重启，宕机等情形下可以通过journal来进行数据恢复
>
> 写入journal操作必须等待直到下次提交日志时完成写入
>
> 为降低延迟，MongoDB可以通过增加commit journal的频率来加快journal写入
>
> 默认false
>
> wtimeout: 该选项指定一个时间限制,以防止写操作无限制被阻塞导致无法应答给客户端
>
> wtimeout的单位为ms，当w值大于1时生效，该参数即仅适用于集群环境
>
> 当某个节点写入时超出指定wtimeout之后，mongod将返回一个错误
>
> 在捕获到超时之前，mongod并不会撤销其他节点已成功完成的写入
>
> wtimeout值为0时等同于没有配置wtimeout选项，容易导致由于某个节点挂起而无法应答

- ordered 可选的。如果为true，则执行文档的有序插入，如果文档中出现错误，则将返回而不处理数组中的剩余文档。如果false，执行一个无序的插入，如果一个文件发生错误，继续处理数组中的剩余文档。 默认为true。 用于插入多个文档

③ 示例

```
// 插入一条文档数据
db.collName.save({name: ‘zhangsan’, age: 25, sex: true});

//插入多条文档数据
db.collName.insert([{name:'JIM', age:19, 'address':'北京'},{name:'Jack',
age:29, 'address':'洛杉矶'}]);

// 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.collName.save({name: ‘zhangsan’, age: 25, sex: true});

//插入一条文档数据
db.collName.insertOne({});

//插入多条文档数据
db.collName.insertMany([{},{},{});
```

4.4.2 更新文档

① 方法

- db.collName.updateOne()
- db.collName.updateMany()
- db.collName.update()
- db.collName.relpaceOne()

② 语法

```
db.collName.update(
   <query>,
   <update>,
   {
     upsert: <boolean>,
     multi: <boolean>,
     writeConcern: <document>
   }
)
```

- **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，用于控制写入安全的级别

③ 更新操作符

- $set 设置字段的值 用法 : { $set : { field : value } }
- $unset 删除字段　用法：{ $unset : { field : 1} }
- $inc 字段增加 字段要求是数字 用法：{ $inc : { field : value } }
- $push 给字段添加值，字段要求是数组 用法：{ $push : { field : value } }
- $pushAll　同pusth 一次可以追加多个值 { $pushAll : { field : value_array } }
- $pop 删除数组内的一个值　用法：删除最后一个值：{ $pop : { field : 1 } }
  删除第一个值：{ $pop : { field : -1 } }
- $min 更改比键对应值小的记录：{ $min: { : } }
- $max 更改比键对应值大的记录：{ $max: { : } }
- $currentDate 更改时间类的值到当前时间：{$currentDate:{ "CreateAt" : true}}

③ 示例

```
//只更新(满足条件的)第一条文档
db.col.update( { "count" : { $gt : 1 } } , { $set : { "test2" : "OK"} } );

//(满足条件)全部更新


db.colName.update( { "count" : { $gt : 3 } } , { $set : { "test2" : "OK"}
```

```
},false,true );

// (如果没有满足条件的) 添加一条数据
db.col.update( { "count" : { $gt : 4 } } , { $set : { "test5" : "OK"}
},true,false );

//updateOne() 方法更新单个文档
db.users.updateOne({"name":"abc"},{$set:{"age":"28"}});

//updateMany() 更新多个文档
db.users.updateMany({"age":{$gt:"10"}},{$set:{"status":"xyz"}})
```

4.4.3 删除文档

① 方法

- db.collName.deleteOne()
- db.collName.deleteMany()
- db.collName.remove()

② 语法

```
db.collName.remove(
   <query>,
   {
     justOne: <boolean>,
     writeConcern: <document>
   }
)
```

- **query**：(可选) 删除的文档的条件。
- **justOne**：(可选) 默认false。如果设为 true 或 1，则只删除一个文档。
- **writeConcern**：(可选) 抛出异常的级别。

③ 示例

```
// 删除所有满足条件的数据
db.collName.remove({'title':'MongoDB'});

//删除满足条件的第一条数据
db.collName.remove({title:'MongoDB'}, {justOne: true})
```

```
//删除集合下全部文档
db.collName.deleteMany({})

//删除满足条件的全部文档
db.collName.deleteMany({ status : "A" })

//删除满足条件的第一个语法
db.collName.deleteOne( { status: "D" } )
```

4.5 集合数据(文档)查询

4.5.1 查询

① 方法

- db.collName.find() 查询所有文档
- db.collName.findOne() 查询满足条件的第一个文档
- db.collName.find().pretty() 返回易读的格式
- db.collName.find().count() 结果中文档的数量
- db.collName.count() 集合中文档的数量
- db.collName.distinct() 询去掉后的当前集合中的某列的重复数据

② 语法

```
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）。

> ```
> db.collName.find(query, {title: 1, by: 1}) // inclusion模式 指定返回的键，不返回其他键
> db.collName.find(query, {title: 0, by: 0}) // exclusion模式 指定不返回的键,返回其他键
> ```

4.5.2 查询条件

① 常规条件

操作	格式	范例
等于	`{<key>:<value>}` value可以是正则	`db.collName.find({"by":"钢铁侠"});`
小于	`{<key>:{$lt:<value>}}`	`db.collName.find({"likes":{$lt:50}});`
小于或等于	`{<key>:{$lte:<value>}}`	`db.collName.find({"likes":{$lte:50}});`
大于	`{<key>:{$gt:<value>}}`	`db.collName.find({"likes":{$gt:50}});`
大于或等于	`{<key>:{$gte:<value>}}`	`db.collName.find({"likes":{$gte:50}});`
不等于	`{<key>:{$ne:<value>}}`	`db.collName.find({"likes":{$ne:50}});`
存在于	`{key:{$in:[1,2,3,5]}}`	`db.collName.find({'likes':{$in:[10,23]}})`

② AND 条件

传入多个键(key)，每个键(key)以逗号隔开

```
db.collName.find({key1:value1, key2:value2});
```

③ OR 条件

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

④ AND 和 OR 联合使用

```
//查询 likes值大于50 并且 (name值是‘史塔克’或title值是‘钢铁侠牛逼’)
db.collName.find({"likes": {$gt:50}, $or: [{"name": "史塔克"},{"title": "钢铁侠牛逼"}]})
```

4.5.3 读取指定数量 limit

① 方法

- db.collName.find().limit()
- db.collName.find().skip()
- db.collName.find().skit().limit()

② 示例

```
// 前两条数据
db.col.find({}).limit(2);

// 跳过前两条 剩下的数据
db.collName.find().skip(0)

//从第三条数据开始 取5条
db.collName.find().skip(2).limit(5)
```

4.5.4 排序

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。

```
db.collName.find().sort({KEY:1})
```

4.6 分组操作(统计)

```
db.collName.aggregate(AGGREGATE_OPERATION)
```

表达式	描述	实例
$sum	计算总和。	db.collName.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	计算平均值	db.collName.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	获取集合中所有文档对应值得最小值。	db.collName.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max	获取集合中所有文档对应值得最大值。	db.collName.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$first	根据资源文档的排序获取第一个文档数据。	db.collName.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	根据资源文档的排序获取最后一个文档数据	db.collName.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

4.7 索引

查看索引

```
db.collName.getIndexs()
```

查看执行时间

```
db.collName.find().explain( "executionStats" )
```

创建索引

```
db.collName.ensureIndex({KEY:1},[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。

ensureIndex() 接收可选参数，可选参数列表如下：

选项	类型	描述
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	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.

删除索引

```
//删除指定索引
db.collName.dropIndex("INDEX-NAME")

//删除所有索引
db.collName.dropIndexes();
```

5. MongoDB 验证登陆

5.1 用户(管理员)管理

添加管理员

```
use admin; //管理员 只能 添加到 admin 数据库下


db.createUser(
  {
    user: "adminUser",
    pwd: "adminPass",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
  }
)
```

添加普通用户

```
use dbName; //所有数据库下 都可以添加普通用户

db.createUser(
  {
    user: "simpleUser",
    pwd: "simplePass",
    roles: [ { role: "readWrite", db: "foo" },
             { role: "read", db: "bar" } ]
  }
)
```

过程类似创建管理员账户，只是 role 有所不同

一点需要注意，如果 admin 库没有任何用户的话，即使在其他数据库中创建了用户，启用身份验证，默认的连接方式依然会有超级权限

角色

- read：允许用户读取指定数据库
- readWrite：允许用户读写指定数据库
- dbAdmin：允许用户在指定数据库中执行管理函数，如索引创建、删除，查看统计或访问system.profile
- userAdmin：允许用户向system.users集合写入，可以找指定数据库里创建、删除和管理用户
- clusterAdmin：只在admin数据库中可用，赋予用户所有分片和复制集相关函数的管理权限。
- readAnyDatabase：只在admin数据库中可用，赋予用户所有数据库的读权限
- readWriteAnyDatabase：只在admin数据库中可用，赋予用户所有数据库的读写权限
- userAdminAnyDatabase：只在admin数据库中可用，赋予用户所有数据库的userAdmin权限
- dbAdminAnyDatabase：只在admin数据库中可用，赋予用户所有数据库的dbAdmin权限。
- root：只在admin数据库中可用。超级账号，超级权限

用户操作

```
//查找指定用户
db.getUser(username)

//查找全部用户
db.getUsers();
show users;


//修改用户
db.updateUser(
   "<username>",
   {
     roles : [
               { role: "<role>", db: "<database>" } | "<role>",
               ...
             ],

     pwd: "<cleartext password>"
```

```
    }
)

//修改用户密码
db.changeUserPassword(username, password)

//删除用户
db.dropUser(username);
```

5.2 验证登陆

第一种方式，连接mongoDB服务器后再验证

```
mongo
db.auth(user, pwd)
```

第二种方式， 连接的时候指定用户名密码

```
mongo -u username -p pwd host:port/dbName
```

启动带访问控制的MongoDB服务器

```
mongod --auth --config ...
```

5.3 MongoDB 安全问题

- 暴露在公网上的MongoDB服务器 https://www.shodan.io/report/h0bgF6zM