# **1**、MongoDB的简介

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。

它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

MongoDB服务端可运行在Linux、Windows平台，支持32位和64位应用，默认端口为27017。

推荐运行在64位平台，因为MongoDB在32位模式运行时支持的最大文件尺寸为2GB。

# **2**、MongoDB的特点

## 2.1文档

定义：

MongoDB中的记录是一个文档，它是由键值对组成的数据结构。多个键及其关联的值有序地放在一起就构成了文档。MongoDB文档类似于JSON对象。字段的值可以包括其他文档，数组和文档数组。



{“greeting”:“hello,world”}这个文档只有一个键“greeting”，对应的值为“hello,world”。多数情况下，文档比这个更复杂，它包含多个键/值对。

例如：{“greeting”:“hello,world”,“foo”: 3} 文档中的键/值对是**有序的**，下面的文档与上面的文档是完全不同的两个文档。{“foo”: 3 ,“greeting”:“hello,world”}；

文档中的值不仅可以是双引号中的字符串，也可以是其他的数据类型，例如，整型、布尔型等，也可以是另外一个文档，即文档可以嵌套。文档中的键类型**只能**是字符串。

**使用文档的优点：**

文档（即对象）对应于许多编程语言中的本机数据类型

嵌入式文档和数组减少了对昂贵连接的需求

动态模式支持流畅的多态性

## 2.2集合

定义：

集合就是一组文档，类似于关系数据库中的**表**。

集合是无模式的，集合中的文档可以是各式各样的。例如，{“hello,word”:“Mike”}和{“foo”: 3}，它们的键不同，值的类型也不同，但是它们可以存放在同一个集合中，也就是不同模式的文档都可以放在同一个集合中。

**既然集合中可以存放任何类型的文档，那么为什么还需要使用多个集合？**

这是因为所有文档都放在同一个集合中，无论对于开发者还是管理员，都很难对集合进行管理，而且这种情形下，对集合的查询等操作效率都不高。所以在实际使用中，往往将文档分类存放在不同的集合中。

例如，对于网站的日志记录，可以根据日志的级别进行存储，Info级别日志存放在Info 集合中，Debug 级别日志存放在Debug 集合中，这样既方便了管理，也提供了查询性能。

但是需要注意的是，这种对文档进行划分来分别存储并不是MongoDB 的强制要求，用户可以灵活选择。

可以使用“.”按照命名空间将集合划分为子集合。

例如，对于一个博客系统，可能包括blog.user 和blog.article 两个子集合，这样划分只是让组织结构更好一些，blog 集合和blog.user、blog.article 没有任何关系。虽然子集合没有任何特殊的地方，但是使用子集合组织数据结构清晰，这也是MongoDB 推荐的方法。

## 2.3数据库

定义：

MongoDB 中多个文档组成集合，多个集合组成数据库。

一个MongoDB 实例可以承载多个数据库。它们之间可以看作相互独立，每个数据库都有独立的权限控制。在磁盘上，不同的数据库存放在不同的文件中。

**MongoDB 中存在以下系统数据库：**

Admin 数据库：一个权限数据库，如果创建用户的时候将该用户添加到admin 数据库中，那么该用户就自动继承了所有数据库的权限。

Local 数据库：这个数据库永远不会被复制，可以用来存储本地单台服务器的任意集合。

Config 数据库：当MongoDB 使用分片模式时，config 数据库在内部使用，用于保存分片的信息。

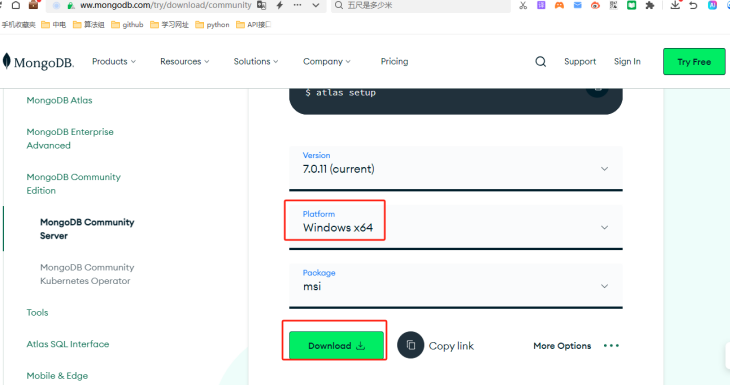
## 2.4数据模型

一个MongoDB 实例可以包含一组数据库，一个DataBase可以包含一组Collection(集合)，一个集合可以包含一组Document(文档)。

# **3**、MongoDB的安装

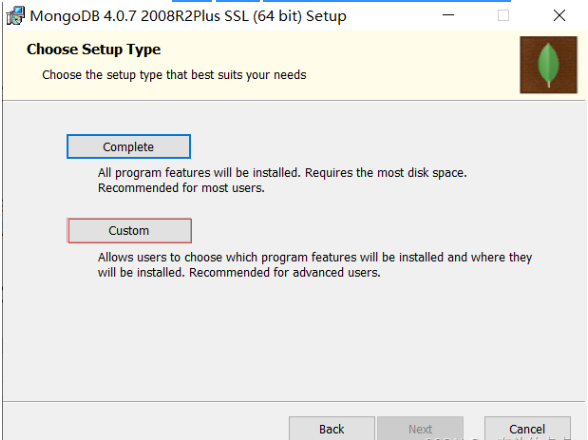
## 3.1 MongoDB的下载

进入官网：https://www.mongodb.com/download-center/community

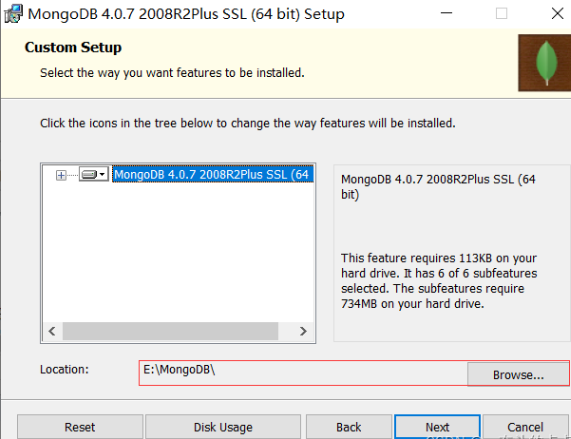


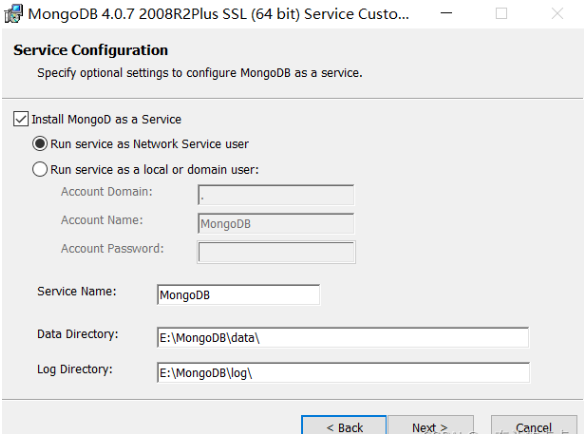
## **3.2** MongoDB的**安装**

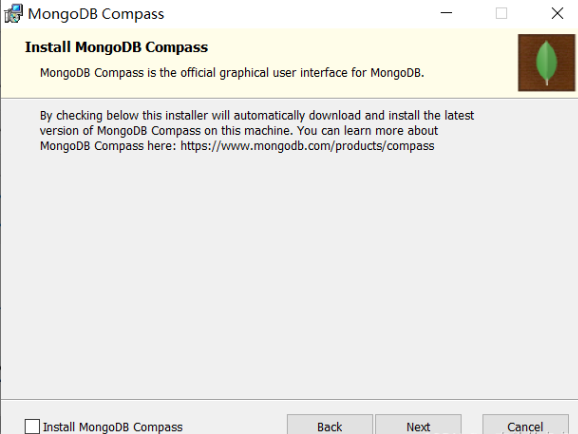
第一步：点击 “Custom(自定义)” 按钮来设置你的安装目录。



第二步：安装在E:\MongoDB这个目录下(安装目录会影响我们后面的配置)



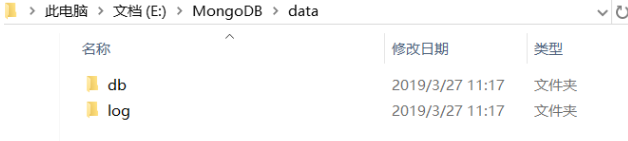




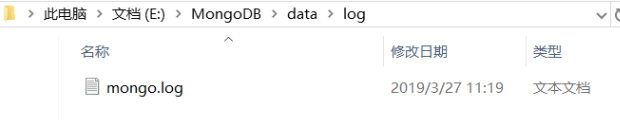
这里安装 "Install MongoDB Compass" 不勾选，否则可能要很长时间都一直在执行安装，MongoDB Compass是一个图形界面管理工具，这里不安装也是没有问题的，可以自己去下载一个图形界面管理工具，比如Robo3T。

# **4**、MongoDB的配置

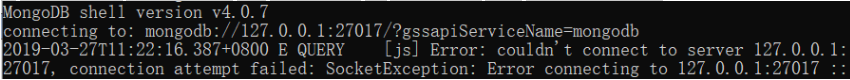
第一步：在MongoDB的data文件夹里新建一个db文件夹和一个log文件夹；



第二步：在log文件夹下新建一个mongo.log；



第三步：将E:\MongoDB\bin添加到环境变量path中，此时打开cmd窗口运行一下mongo命令，出现如下情况：



这是为什么呢？这是因为我们还没有启动MongoDB服务，自然也就连接不上服务了。那要怎么启动呢？在cmd窗口中运行如下命令：



需要注意的是：如果你没有提前创建db文件夹，是无法启动成功的。运行成功之后，我们打开浏览器，输入127.0.0.1:27017，看到如下图，就说明MongoDB服务已经成功启动了。

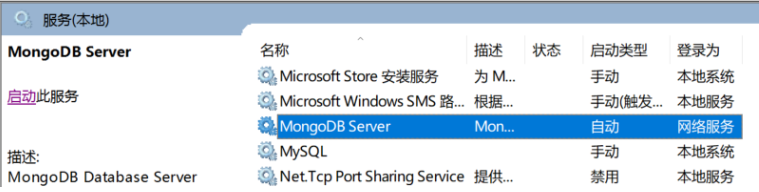


但是如果每次都要这么启动服务的话也太麻烦了吧，这里你可以选择设置成开机自启动，也可以选择用命令net start mongodb来手动启动，这里我选择使用后者，具体方法如下。

还是打开cmd窗口，不过这次是以管理员身份运行，然后输入如下命令：



如果没有报错的话就说明成功添加到服务里了，可以使用win+R然后输入services.msc命令进行查看：



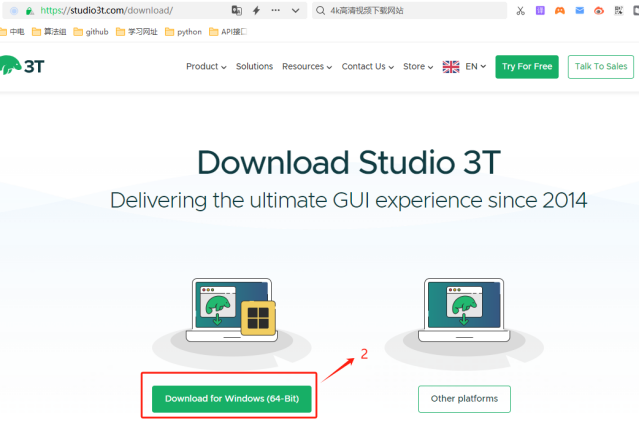
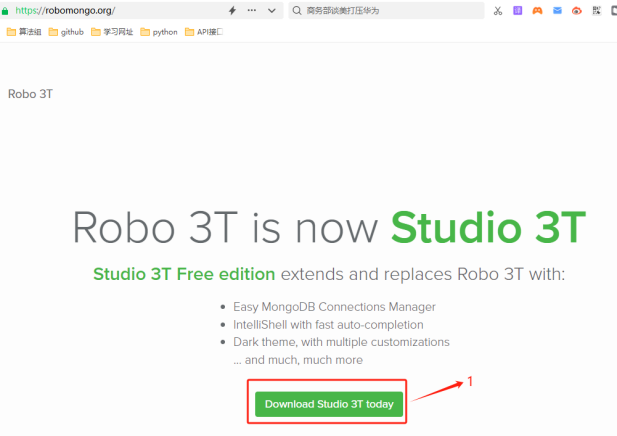
默认是自动运行的，这里我选择把它改成手动的。然后在cmd窗口中运行net start mongodb：



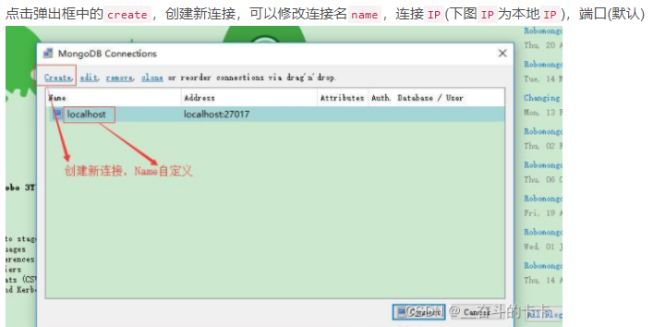
# **5、MongoDB的可视化工具**RoBo 3T

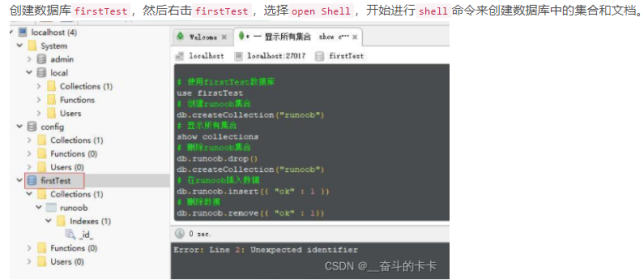
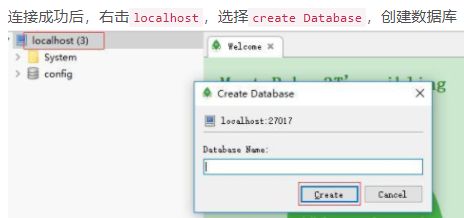
## 5.1 RoBo 3T的下载

进入官网：https://robomongo.org/download



## **5.2 RoBo 3T的安装**





# **6、MongoDB的基本操作**

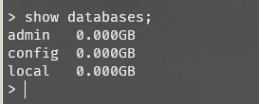
## 6.1数据库

### **登录数据库**

mongo

### **查看数据库**

show databases



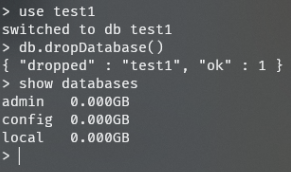
### **选择数据库**

use 数据库名



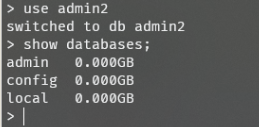
### **删除数据库**

db.dropDataBase()



**注意：**

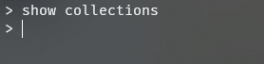
如果切换到一个没有的数据库，例如use admin2，那么会隐式创建这个数据库。(后期当该数据库有数据时，系统自动创建)



## 6.2集合

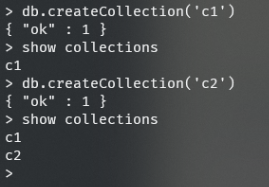
### **查看集合**

show collections



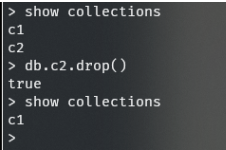
### **创建集合**

db.createCollection('集合名')



### **删除集合**

db.集合名.drop()



## 6.3增查改删

### 增加

格式：

db.集合名.insert(JSON数据)

db.集合名.save({"name":"zhangsan"})

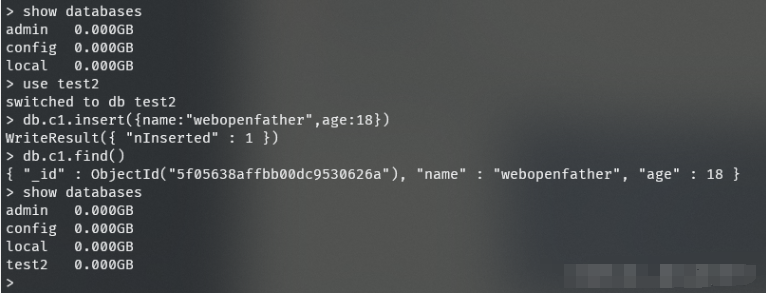
**注意：**

如果集合存在，那么直接插入数据。如果集合不存在，那么会隐式创建。

·数据库和集合不存在都隐式创建；

·对象的键统一不加引号（方便看），但是查看集合数据时系统会自动加；

·mongodb会给每条数据增加一个全球唯一的\_id键；



### 查询

格式：

db.task.find()

db.getCollection("task").find()

#条件查询

db.task.find({'name':'张三'})

### 修改

格式：

# update:遇到满足条件的第一条数据修改

db.task.update({"name":"zhangsan"},{$set:{"name":"lisi"}})

# updateMany :修改所有满足条件的

db.task.updateMany({"name":"zhangsan"},{$set:{"name":"lisi"}})

# updateOne:修改满足条件的第一条数据

db.task.updateOne({"name":"zhangsan"},{$set:{"name":"lisi"}})

### 删除

格式：

# 删除

db.task.remove({name:"zhangsan"})

# 删除所有

db.task.remove({})

## 6.4文档

### **查询文档**

db.集合名.find(条件[,查询的列])



其他语法：

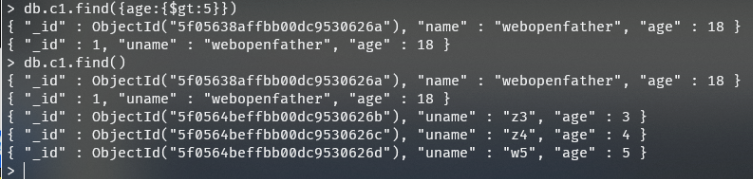
db.集合名.find({键:{运算符：值}})



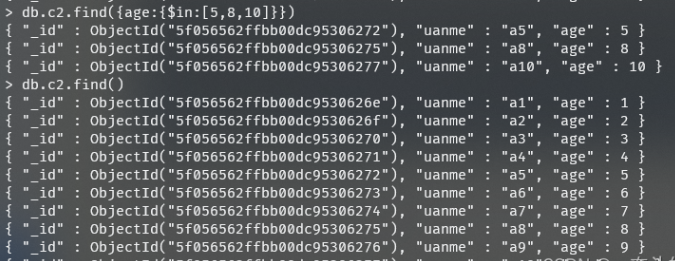
实例：



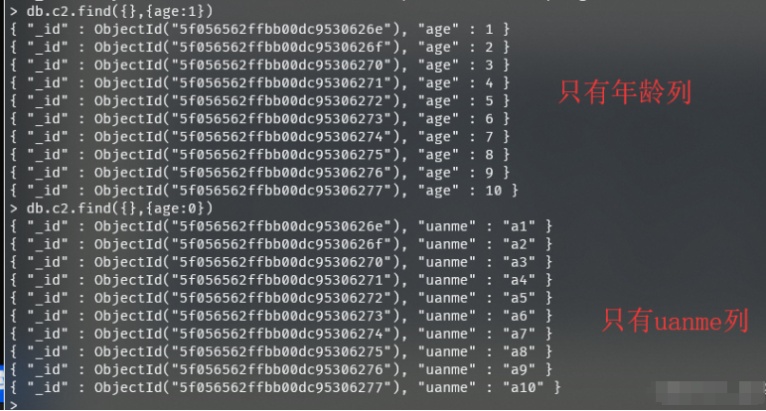
1、查询age大于5的数据



2、查询年龄是5岁、8岁、10岁的数据



3、只看年龄列，或者年龄以外的列



### 修改文档

db.集合名.update(条件,新数据[是否新增,是否修改多条,])

**注意：**

更新数据需要使用修改器，如果不使用，那么会将新数据替换原来的数据。格式：

db.集合名.update(条件,{修改器:{键:值}}[是否新增,是否修改多条,])

修改器作用：

Inc：递增；

Rename：重命名列；

set：修改列值；

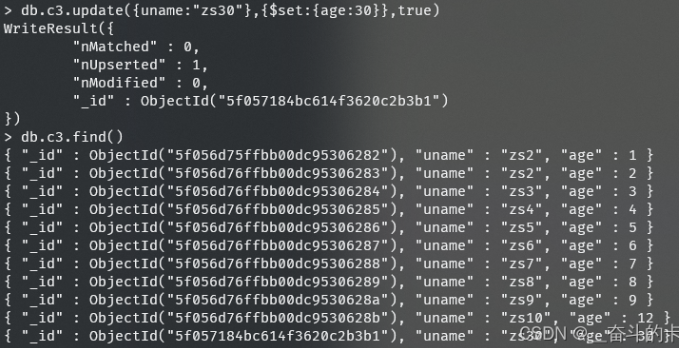
unset：删除列；

是否新增：

指条件匹配不到数据则插入(true是插入，false否不插入默认)

db.c3.update({uname:"zs30"},{$set:{age:30}},true)

是否修改多条：  
 指将匹配成功的数据都修改(true是，false否默认)  
 db.c3.update({uname:"zs2"},{$set:{age:30}},false,true)



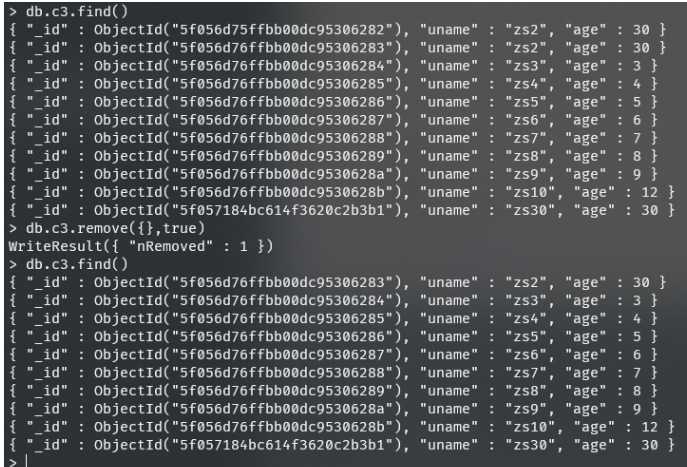
### 删除文档

格式：

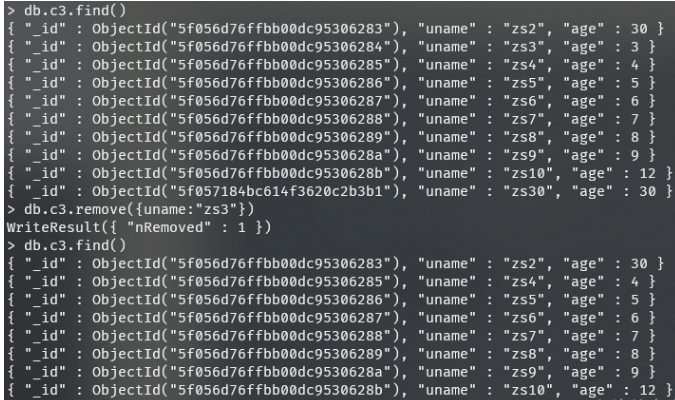
db.集合名.remove(条件[,是否删除一条])

实例：

1.是否删除一条：  
true：是(删除的数据为第一条)



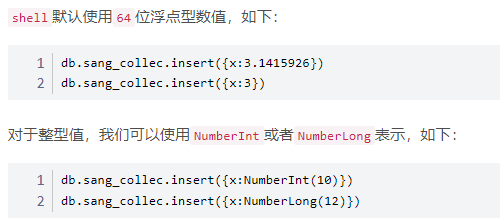
false：否



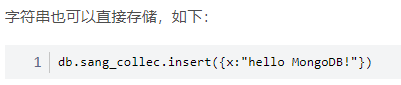
# **7、MongoDB存储数据类型**

MongoDB中每条记录称作一个文档，这个文档和我们平时用的JSON有点像，但也不完全一样。简洁和清晰的层次结构使得JSON成为理想的数据交换语言，JSON易于阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率，但是JSON也有它的局限性，比如它只有null、布尔、数字、字符串、数组和对象这几种数据类型，没有日期类型，只有一种数字类型，无法区分浮点数和整数，也没法表示正则表达式或者函数。由于这些局限性，BSON闪亮登场啦，BSON是一种类JSON的二进制形式的存储格式，简称Binary JSON，它和JSON一样，支持内嵌的文档对象和数组对象，但是BSON有JSON没有的一些数据类型，如Date和BinData类型，MongoDB使用BSON做为文档数据存储和网络传输格式。

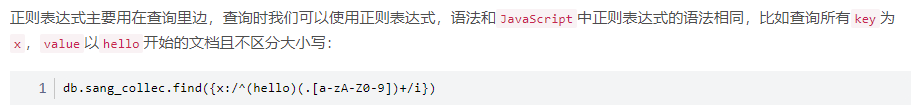
## 7.1数字



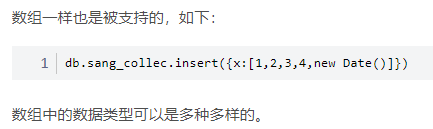
## 7.2字符串



## 7.3正则表达式



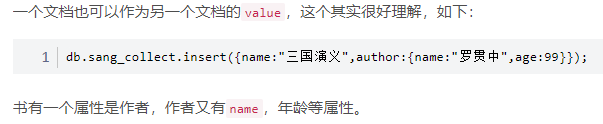
## 7.4数组



## 7.5日期



## 7.6内嵌文档

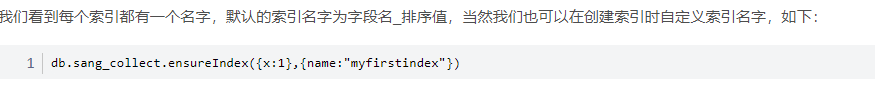


# **8、MongoDB中的索引**

## 8.1索引创建

格式：

db.sang\_collect.ensureIndex({x:1})



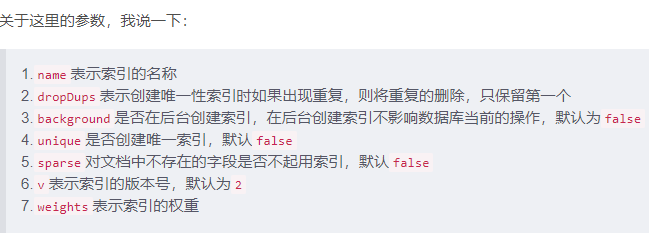
查看索引的大小：

db.sang\_collect.totalIndexSize()

**注意：**

当然索引在创建的过程中还有许多其他可选参数，如下：

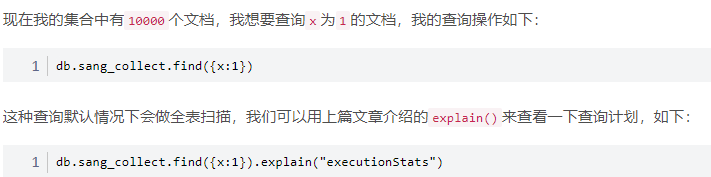
db.sang\_collect.ensureIndex({x:1},{name:"myfirstindex",dropDups:true,background:true,unique:true,sparse:true,v:1,weights:99999})



## 8.2查看索引

格式：

db.sang\_collect.getIndexes()





## 8.3删除索引

按名称删除索引：

db.sang\_collect.dropIndex("xIndex") #表示删除一个名为xIndex的索引

删除所有索引：

db.sang\_collect.dropIndexes()

**注意：**

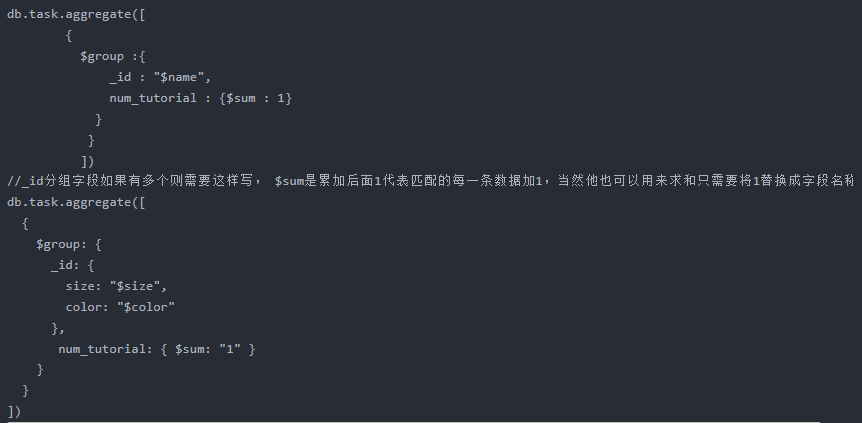
索引是个好东西，可以有效的提高查询速度，但是索引会降低插入、更新和删除的速度，因为这些操作不仅要更新文档，还要更新索引，MongoDB 限制每个集合上最多有64个索引，我们在创建索引时要仔细斟酌索引的字段。

# **9、聚合aggregate**

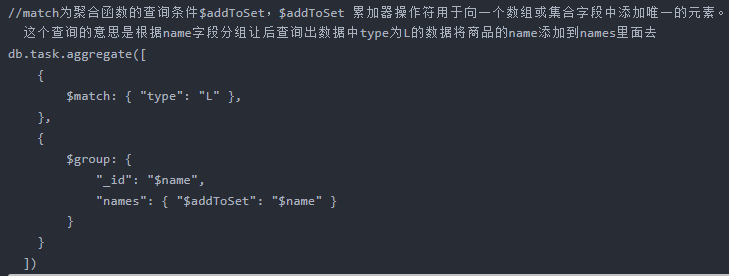
以下的操作符必须在aggregate聚合函数中使用：



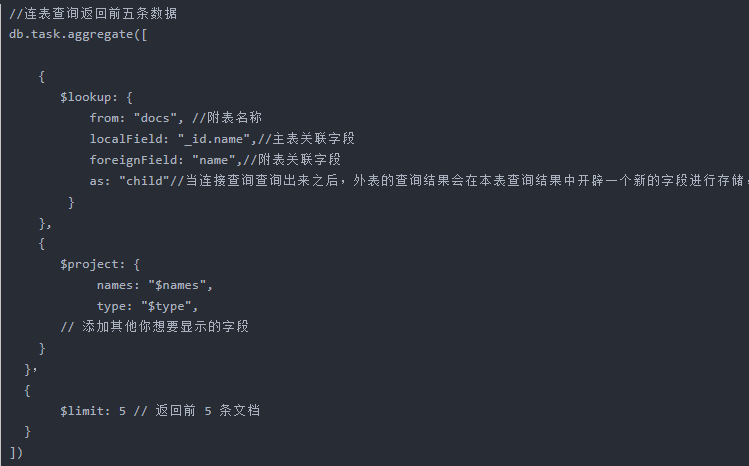
## 9.1分组

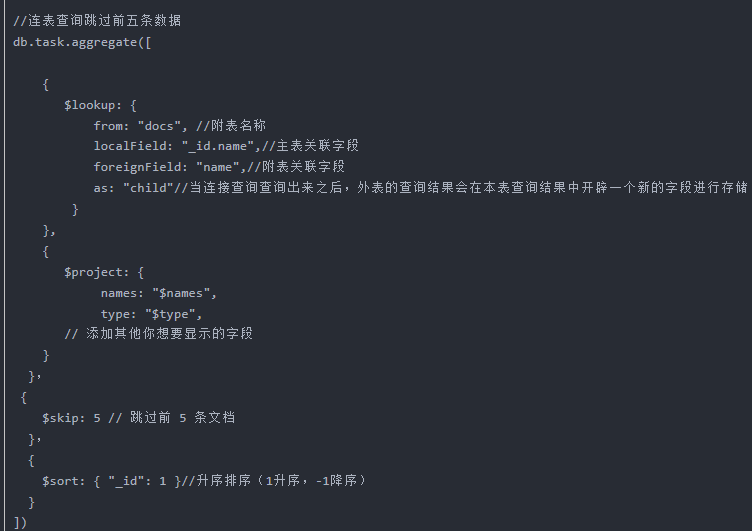


## 9.2删除文档筛选

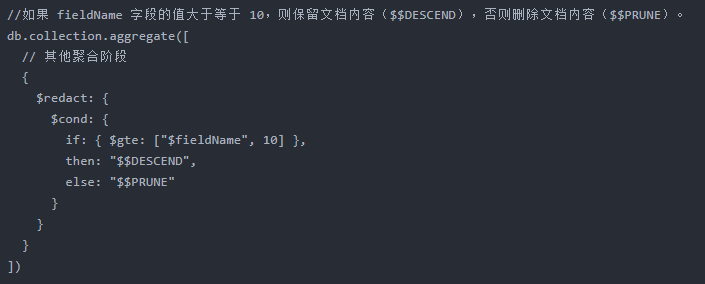


## 9.3连表查询



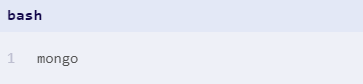


## 9.4删除索引限制文档

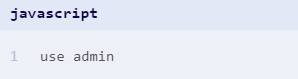


# 10、添加用户

第一步：启动mongo shell；



第二步： 切换到admin数据库；



第三步：添加用户和密码；



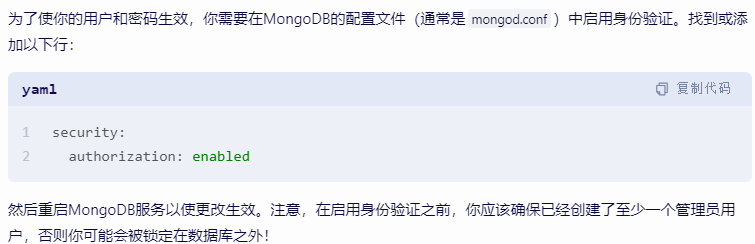
第四步：查看用户；



第五步：退出mongo shell；

exit或quit()

第六步：可选）启用身份验证；



在MongoDB中，roles字段用于指定用户的权限。角色定义了用户可以执行哪些操作以及他们可以访问哪些数据。当你在db.createUser()方法中使用roles字段时，你是在定义这个用户账户在MongoDB系统中的行为权限。如果你不指定roles字段，那么用户将没有任何内置的权限。这通常不是一个好的做法，因为即使用户能够认证到MongoDB系统，他们也将无法执行任何操作。