**MongoDB相关笔记**

# 目 录

# MongoDB数据库概述

## 数据库服务器概述

### 如何保存数据

1、数据库和文件的主要区别

1、 数据库有数据库表、行和列的概念，让我们存储操作数据更方便

2、 数据库提供了非常方便的接口，可以让 nodejs、php java .net 很方便的实现增加修改删

除功能。

2、NoSql 介绍

2.1 NoSQL 介绍

由于互联网的迅速发展，云计算与 Web2.0。这样大量的交互给数据库提出了更高的性能要

求，传统的数据库（本文泛指 SQL 数据库），即关系数据库虽然具备良好的事物管理，但在处理

大量数据的应用时很难在性能上满足设计要求。NoSQL 就是主要为了解决当下大量高并发高要

求的数据库应用需求，关系数据库具有严格的参照性，一致性，可用性，原子性，隔离性等特点，

因此会产生一些例如表连接等操作，这样会大大降低系统的性能。而在当前很多应用场景下对性

能的要求远远强于传统数据库关注的点，NoSQL 就是为了解决大规模数据与多样数据种类等问

题，尤其是其中大数据的相关问题。 。

NoSQL(NoSQL = Not Only SQL )，意即“不仅仅是 SQL”，它指的是非关系型的数据库，是以 key-value形式存储，和传统的关系型数据库不一样，不一定遵循传统数据库的一些基本要求，比如说遵循 SQL 标准、

ACID 属性、表结构等等。NoSQL 最早被提出是在 20 世纪 80 年代，在当时更多是强调的是与关系

数据库区别对待，最近这些年被提及的更多是强调协助解决大数据等相关问题。NoSQL 在大数

据时代有自己的意义。

2.2 NoSQL 应用情况介绍

国内的互联网蓬勃发展，不仅涌现出 BAT（百度，阿里巴巴，腾讯）之类的巨头，也带动了整个互联

网行业的发展，大量的创业型公司如春笋般的涌出，在国家层面也提出了“互联网+”和“万众创业”的口

号。更多传统的行业也开始拥抱互联网。但是无论是做所谓的生态平台还是传统业务的转型，涉及到的业

务是多种多样的。这个时候企业架构师对于应用系统的核心——数据库管理 不仅有传统的 SQL 选项也有了

NoSQL 这种适合特定场景需求的选项

。

NoSQL 数据库在以下的这几种情况下比较适用：

1、数据模型比较简单；

2、需要灵活性更强的 IT 系统；

3、对数据库性能要求较高；

4、不需要高度的数据一致性；

5、对于给定 key，比较容易映射复杂值的环境。

2.3 NoSQL 发展现状

**国外： Google 的 BigTable 和 Amazon 的 Dynamo 使用的就是 NoSQL 型数据库。**

**国内：百度、阿里、腾讯、新浪微博、视觉中国、优酷运营数据分析、飞信空间、豆瓣社区等..**

3、什么时候建议使用 NoSql

1、对数据库高并发读写的需求

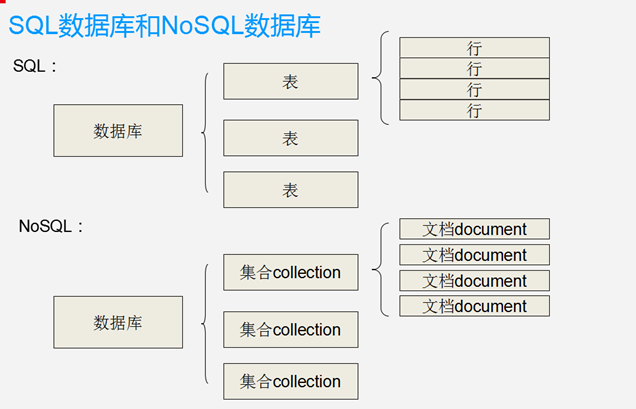
2、对海量数据的高效率存储和访问的需求

3、对数据库的高可扩展性和高可用性的需求

4、NoSql 和传统数据库简单对比。

**非结构型数据库。没有行、列的概念。用 JSON 来存储数据。**

**集合就相当于“表”，文档就相当于“行”。**

****

5、NoSql 种**类**



6、MongoDb 介绍

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。他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 最大的特点是他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。它的特点是高性能、易部署、

**易使用，存储数据非常方便。**

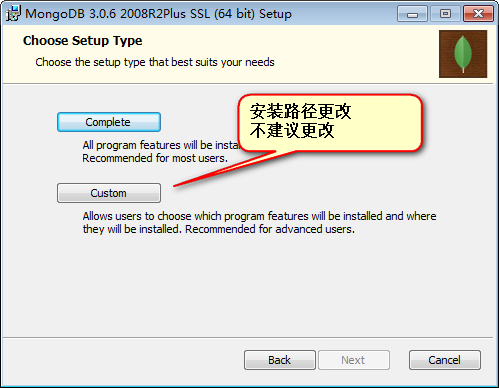
7、MongoDb 安装

官网：https://www.mongodb.com/

手册：https://docs.mongodb.org/manual/

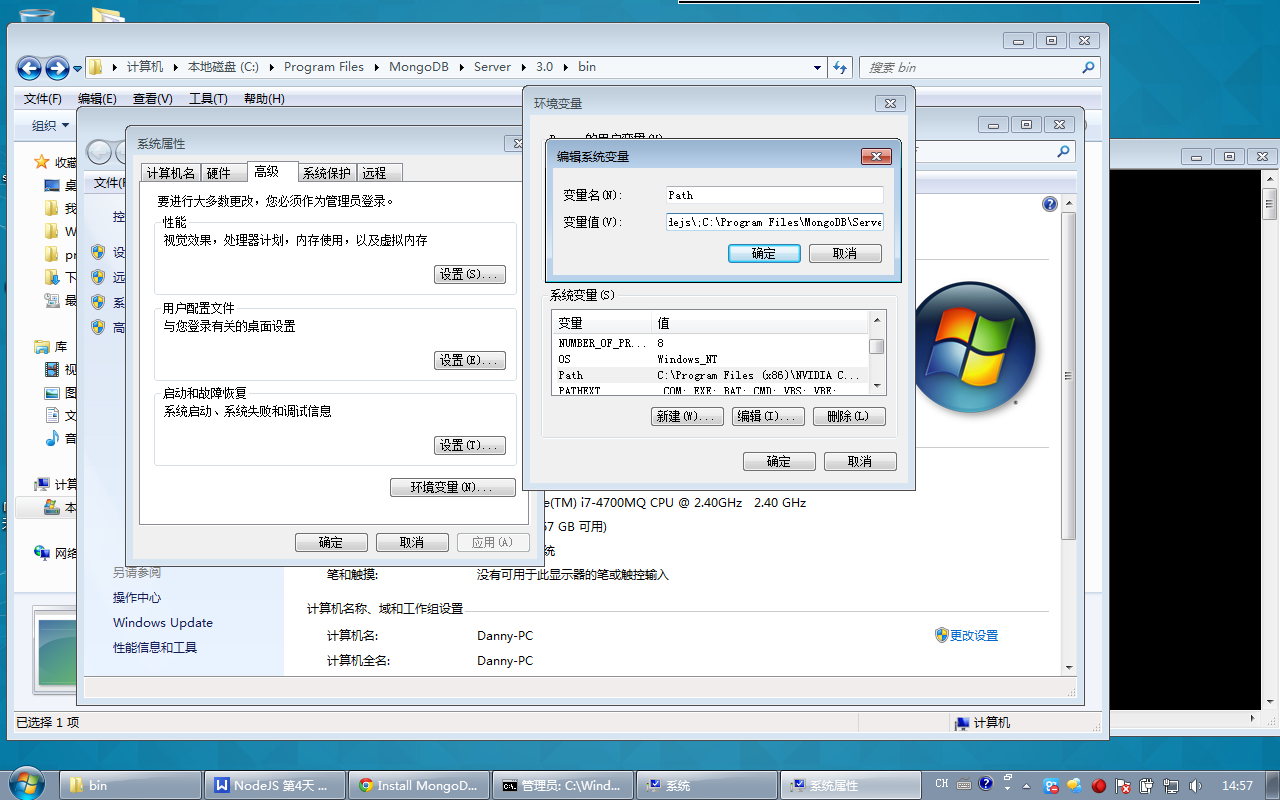
**注意：部分 win7 系统需要安装补丁，KB2731284。**

**1. 双击 MongoDB 软件下一步下一步安装**

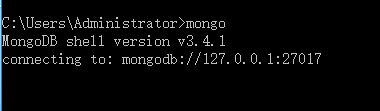


2. 安装完成配置环境变量 C:\Program Files\MongoDB\Server\3.0\bin 加入到系统的

**path 环境变量中**



3. 打开 cmd 输入 ：mongo命令看看是否成功。如果出来下图说明 mongodb配置成功。



8、使用 MongoDb

1. 新建一个存放数据库的文件夹，注意：不能有中文和空格，建议不要放在 C 盘

2.启动 MongoDb 服务

服务端：mongod 开启数据库服务 mongod --dbpath C:\mongodb

**开启 MongoDb 服务命令：**



--dbpath 就是选择数据库文档所在的文件夹。

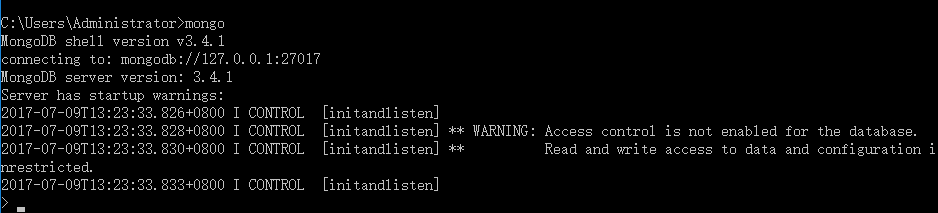
也就是说，mongoDB 中，真的有物理文件，对应一个个数据库。U 盘可以拷走。

**注意：一定要保持，开机这个 CMD 不能动了，不能关，不能 ctrl+c。 一旦这个 cmd 有问题了，数据**

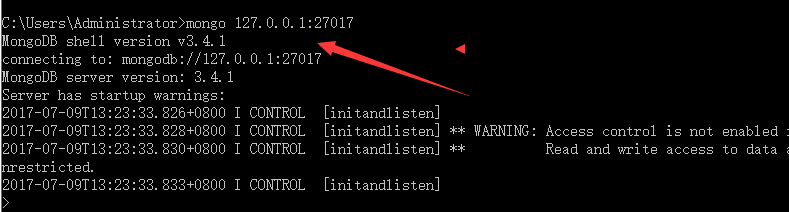
库就自动关闭了。

3.客户端输入 mongo 命令连接服务端

客户端：mongo 使用数据库



客户端：mongo 使用数据库 ip 地址:端口号



## MongoDB 数据库创建删除、表（集合）创建删除、数据增删改查

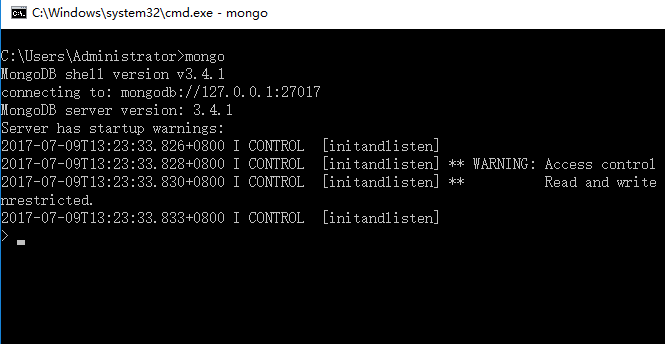
**一、 数据库使用**

**开启 mongodb 服务：要管理数据库，必须先开启服务，开启服务使用 mongod --dbpath**

c:\mongodb



管理**mongodb**数据库：mongo (一定要在新的cmd中输入)



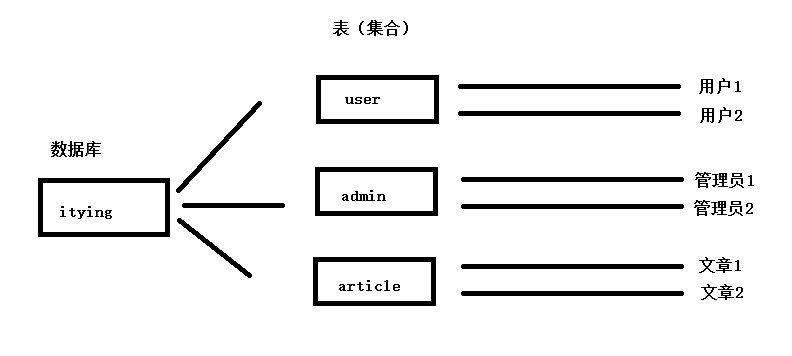
清屏：

cls

**查看所有数据库列表**

show dbs

**二、 创建数据库**



**使用数据库、创建数据库**

use student

如果真的想把这个数据库创建成功，那么必须插入一个数据。

数据库中不能直接插入数据，只能往集合(collections)中插入数据。不需要专门创建集合，只

需要写点语法插入数据就会创建集合：

db.student.insert({“name”:”xiaoming”});

db.student 系统发现 student 是一个陌生的集合名字，所以就自动创建了集合。

显示当前的数据集合（mysql 中叫表）

show collections

删除数据库，删除当前所在的数据库

db.dropDatabase();

删除集合，删除指定的集合 删除表

删除集合 db.COLLECTION

\_NAME.drop()

db.user.drop()

**三、 插入（增加）数据**

插入数据，随着数据的插入，数据库创建成功了，集合也创建成功了。

db.表名.insert({"name":"zhangsan"}); student 集合名称（表）

**四、 查找数据**

1、查询所有记录

db.userInfo.find();

相当于：select\* from userInfo;

**2、查询去掉后的当前聚集集合中的某列的重复数据**

db.userInfo.distinct("name");

会过滤掉 name 中的相同数据

相当于：select distict name from userInfo;

3、查询 age = 22 的记录

db.userInfo.find({"age": 22});

相当于： select \* from userInfo where age = 22;

4、查询 age > 22 的记录

db.userInfo.find({age: {$gt: 22}});

相当于：select \* from userInfo where age >22;

5、查询 age < 22 的记录

db.userInfo.find({age: {$lt: 22}});

相当于：select \* from userInfo where age <22;

6、查询 age >= 25 的记录

db.userInfo.find({age: {$gte: 25}});

相当于：select \* from userInfo where age >= 25;

7、查询 age <= 25 的记录

db.userInfo.find({age: {$lte: 25}});

8、查询 age >= 23 并且 age <= 26 注意书写格式

db.userInfo.find({age: {$gte: 23, $lte: 26}});

9、查询 name 中包含 mongo 的数据 模糊查询用于搜索

db.userInfo.find({name: /mongo/});

//相当于%%

select \* from userInfo where name like ‘%mongo%’;

10、查询 name 中以 mongo 开头的

db.userInfo.find({name: /^mongo/});

select \* from userInfo where name like ‘mongo%’;

11、查询指定列 name、age 数据

db.userInfo.find({}, {name: 1, age: 1});

相当于：select name, age from userInfo;

当然 name 也可以用 true 或 false,当用 ture 的情况下河 name:1 效果一样，如果用 false 就

是排除 name，显示 name 以外的列信息。

12、查询指定列 name、age 数据, age > 25

db.userInfo.find({age: {$gt: 25}}, {name: 1, age: 1});

相当于：select name, age from userInfo where age >25;

13、按照年龄排序 1 升序 -1 降序

升序：db.userInfo.find().sort({age: 1});

降序：db.userInfo.find().sort({age: -1});

14、查询 name = zhangsan, age = 22 的数据

db.userInfo.find({name: 'zhangsan', age: 22});

相当于：select \* from userInfo where name = ‘zhangsan’ and age = ‘22’;

15、查询前 5 条数据

db.userInfo.find().limit(5);

相当于：selecttop 5 \* from userInfo;

16、查询 10 条以后的数据

db.userInfo.find().skip(10);

相当于：select \* from userInfo where id not in (

selecttop 10 \* from userInfo

);

17、查询在 5-10 之间的数据

db.userInfo.find().limit(10).skip(5);

可用于分页，limit 是 pageSize，skip 是第几页\*pageSize

18、or 与 查询

db.userInfo.find({$or: [{age: 22}, {age: 25}]});

相当于：select \* from userInfo where age = 22 or age = 25;

19、findOne 查询第一条数据

db.userInfo.findOne();

相当于：selecttop 1 \* from userInfo;

db.userInfo.find().limit(1);

**20、查询某个结果集的记录条数 统计数量**

db.userInfo.find({age: {$gte: 25}}).count();

相当于：select count(\*) from userInfo where age >= 20;

如果要返回限制之后的记录数量，要使用 count(true)或者 count(非 0)

db.users.find().skip(10).limit(5).count(true);

**四、修改数据**

修改里面还有查询条件。你要该谁，要告诉 mongo。

查找名字叫做小明的，把年龄更改为 16 岁：

1 db.student.update({"name":"小明"},{$set:{"age":16}});

查找数学成绩是 70，把年龄更改为 33 岁：

1 db.student.update({"score.shuxue":70},{$set:{"age":33}});

更改所有匹配项目："

By default, the update() method updates a single document. To update multiple documents, use

the multi option in the update() method.

1 db.student.update({"sex":"男"},{$set:{"age":33}},{multi: true});

完整替换，不出现$set 关键字了： 注意

1 db.student.update({"name":"小明"},{"name":"大明","age":16});

db.users.update({name: 'Lisi'}, {$inc: {age: 50}}, false, true);

相当于：update users set age = age + 50 where name = ‘Lisi’;

db.users.update({name: 'Lisi'}, {$inc: {age: 50}, $set: {name: 'hoho'}}, false, true);

相当于：update users set age = age + 50, name = ‘hoho’ where name = ‘Lisi’;

**五、 删除数据**

db.collectionsNames.remove( { "borough": "Manhattan" } )

db.users.remove({age: 132});

By default, the remove() method removes all documents that match the remove condition. Use

the justOne option to limit the remove operation to only one of the matching documents.

db.restaurants.remove( { "borough": "Queens" }, { justOne: true } )

## MongoDB 索引 和 explain 的使用

**一、索引基础**

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，可以让我们查询数据库变得

更快。MongoDB 的索引几乎与传统的关系型数据库一模一样，这其中也包括一些基本的查

询优化技巧。

**下面是创建索引的命令：**

db.user.ensureIndex({"username":1})

**获取当前集合的索引：**

db.user.getIndexes()

**删除索引的命令是：**

db.user.dropIndex({"username":1})

在 MongoDB 中，我们同样可以创建复合索引，如：

数字 1 表示 username 键的索引按升序存储，-1 表示 age 键的索引按照降序方式存储。

db.user.ensureIndex({"username":1, "age":-1})

该索引被创建后，基于 username 和 age 的查询将会用到该索引，或者是基于 username

的查询也会用到该索引，但是只是基于 age 的查询将不会用到该复合索引。因此可以说，

**如果想用到复合索引，必须在查询条件中包含复合索引中的前 N 个索引列。然而如果查询**

条件中的键值顺序和复合索引中的创建顺序不一致的话，MongoDB 可以智能的帮助我们调

整该顺序，以便使复合索引可以为查询所用。如：

db.user.find({"age": 30, "username": "stephen"})

对于上面示例中的查询条件，MongoDB 在检索之前将会动态的调整查询条件文档的顺

序，以使该查询可以用到刚刚创建的复合索引。

对于上面创建的索引，MongoDB 都会根据索引的 keyname 和索引方向为新创建的索引

自动分配一个索引名，下面的命令可以在创建索引时为其指定索引名，如：

db.user.ensureIndex({"username":1},{"name":"userindex"})

随着集合的增长，需要针对查询中大量的排序做索引。如果没有对索引的键调用 sort，

MongoDB 需要将所有数据提取到内存并排序。因此在做无索引排序时，如果数据量过大以

**致无法在内存中进行排序，此时 MongoDB 将会报错。**

**二、唯一索引**

在缺省情况下创建的索引均不是唯一索引。下面的示例将创建唯一索引，如：

db.user.ensureIndex({"userid":1},{"unique":true})

如果再次插入 userid 重复的文档时，MongoDB 将报错，以提示插入重复键，如：

db.user.insert({"userid":5})

db.user.insert({"userid":5})

E11000 duplicate key error index: user.user.$userid\_1 dup key: { : 5.0 }

如果插入的文档中不包含 userid 键，那么该文档中该键的值为 null，如果多次插入类似

的文档，MongoDB 将会报出同样的错误，如：

db.user.insert({"userid1":5}) db.user.insert({"userid1":5})

E11000 duplicate key error index: user.user.$userid\_1 dup key: { : null }

如果在创建唯一索引时已经存在了重复项，我们可以通过下面的命令帮助我们在创建唯

一索引时消除重复文档，仅保留发现的第一个文档，如：

先删除刚刚创建的唯一索引。

db.user.dropIndex({"userid":1})

插入测试数据，以保证集合中有重复键存在。

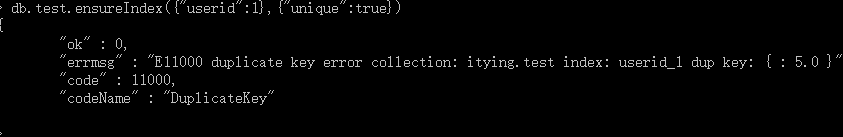
db.user.remove()

db.user.insert({"userid":5})

db.user.insert({"userid":5})

重新创建唯一索引

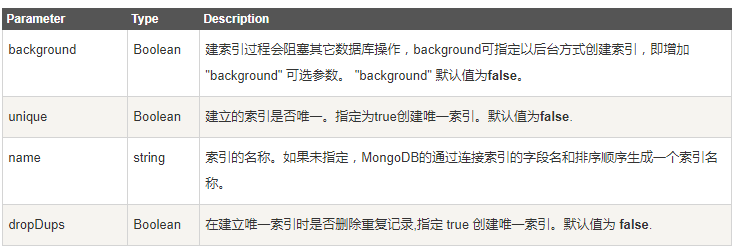
db.user.ensureIndex({"userid":1},{"unique":true })



**我们同样可以创建复合唯一索引，即保证复合键值唯一即可。如：**

db.user.ensureIndex({"userid":1,"age":1},{"unique":true})

**三、索引的一些参数**



如果在为已有数据的文档创建索引时，可以执行下面的命令，以使 MongoDB 在后台创

建索引，这样的创建时就不会阻塞其他操作。但是相比而言，以阻塞方式创建索引，会使整

个创建过程效率更高，但是在创建时 MongoDB 将无法接收其他的操作。

db.user.ensureIndex({"username":1},{"background":true})

**四、使用 explain**

explain 是非常有用的工具，会帮助你获得查询方面诸多有用的信息。只要对游标调用

该方法，就可以得到查询细节。explain 会返回一个文档，而不是游标本身。如：



explain 会返回查询使用的索引情况，耗时和扫描文档数的统计信息。

**五、explain executionStats 查询具体的执行时间**

db.tablename.find().explain( "executionStats" )

关注输出的如下数值：explain.executionStats.executionTimeMillis

## Nodejs 操作 MongoDb 数据库

**一、 在 Nodejs 中使用 Mongodb**

前面的课程我们讲了用命令操作 MongoDB，这里我们看下如何用 nodejs 来操作数据库

**需要引包：**

npm install mongodb --save-dev / cnpm install mongodb --save-dev

**网址：**

https://docs.mongodb.com/getting-started/node/introduction/

https://docs.mongodb.com/getting-started/node/query/

**二、Nodejs 连接 MongoDb 数据库**

**var express = require("express");**

//数据库引用

**var MongoClient = require('mongodb').MongoClient;**

**var app = express();**

//数据库连接的地址，最后的斜杠表示数据库名字

**var shujukuURL = 'mongodb://localhost:27017/news';**

**app.get("/",function(req,res){**

//连接数据库，这是一个异步的操作 MongoClient.connect(shujukuURL, function(err, db) { res.writeHead(200,{"Content-Type":"text/html;charset=UTF8"}); if(err){ res.send("数据库连接失败"); return; } res.write("恭喜，数据库已经成功连接 \n"); db.collection("user").insertOne({"name":"哈哈"},function(err,result)

{

if(err){ res.send("数据库写入失败"); return; } res.write("恭喜，数据已经成功插入"); res.end(); //关闭数据库 db.close(); }); });

});

app.listen(8020);

**二、Nodejs 查找 MongoDb 数据库集合**

**MongoClient.connect(dbUrl,function(err,db){**

if(err){ /\*数据库连接失败\*/ console.log('数据库连接失败'); return; } var result=[]; var userRel=db.collection('user').find(); //res.send(userRel); userRel.each(function(err, doc) { if(err){ res.write("游标遍历错误"); return; } if (doc != null) { result.push(doc); } else { console.log(result); //遍历完毕 db.close(); res.render("index",{ "result" : result }); } });

})

**三、Nodejs 给 MongoDb 增加数据**

**MongoClient.connect(dbUrl,function(err,db){**

if(err){ return } db.collection('user').insertOne({ "name" : name, "age" : age, "score" : { "shuxue" : shuxuechengji, "yuwen" : yuwenchengji } },function(err,result){ if(err){ console.log('写入数据失败'); } //关闭数据库 db.close(); //res.redirect('/add'); res.redirect('/'); /\*路由跳转\*/ res.end(); ////res.location('/add') })

})

**四、Nodejs 修改 MongoDb 数据**

**MongoClient.connect(dbUrl,function(err,db){**

**if(err){ console.log('数据库连接错误'); return; } db.collection('user').updateOne({"\_id":ObjectID(id)}, { "name": name, "age": age, "sco"shuxue": shuxue,**

"yuwen": yuwen } },function (err, results) { console.log(results); db.close(); res.redirect('/'); /\*路由跳转\*/ res.end('end'); })

})

**五、Nodejs 删除 MongoDb 数据**

**MongoClient.connect(dbUrl,function(err,db){**

if(err){ throw new Error("数据库连接失败"); return; } db.collection('user').deleteOne({"

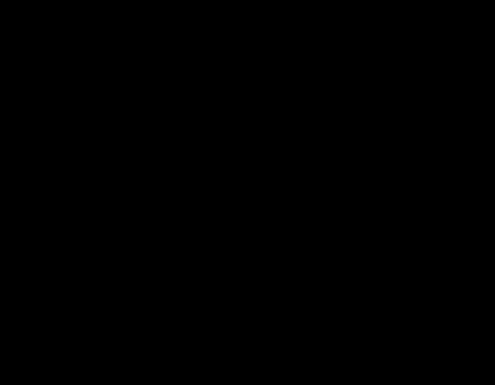
**\_id":ObjectID(id)},function(error,result){**

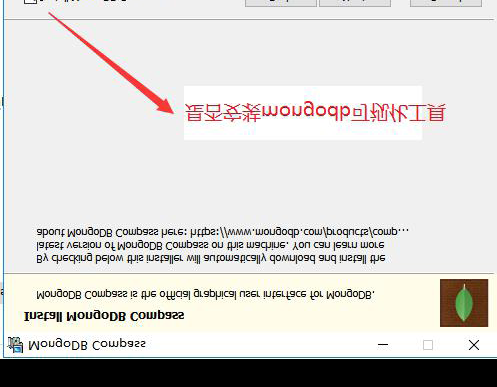
if(error) { throw new Error('删除数据失败'); return; } db.close(); res.redirect('/'); /\*路由跳转\*/ })

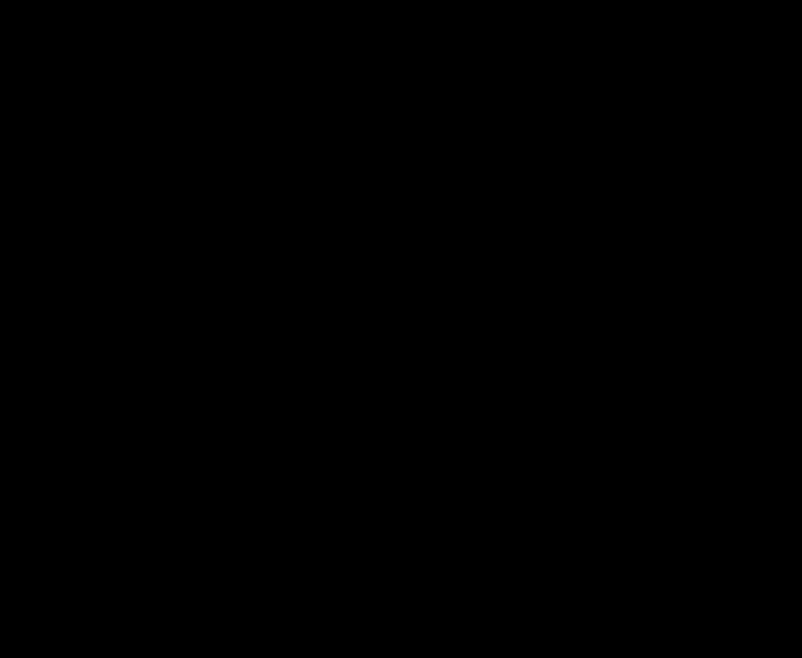
})

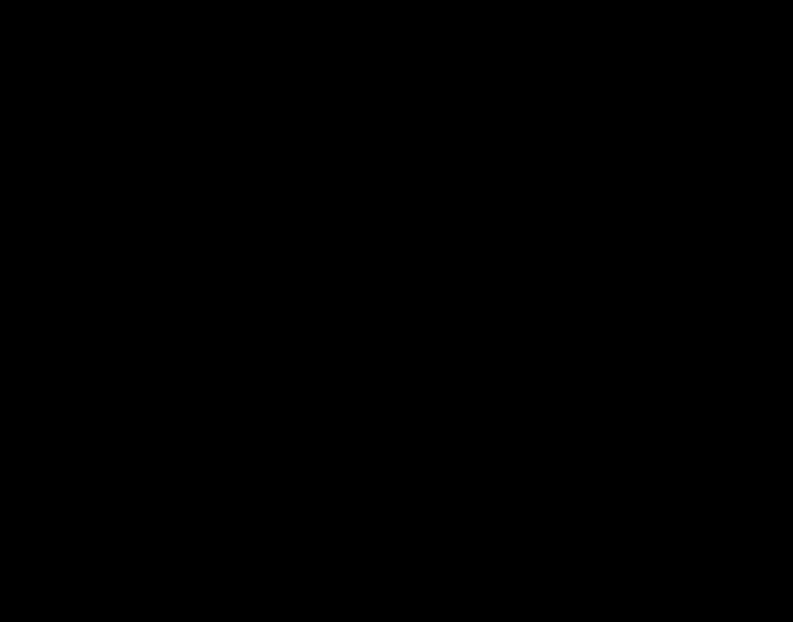
## Mongodb4.x 的使用以及 Mongodb 账户权限配置

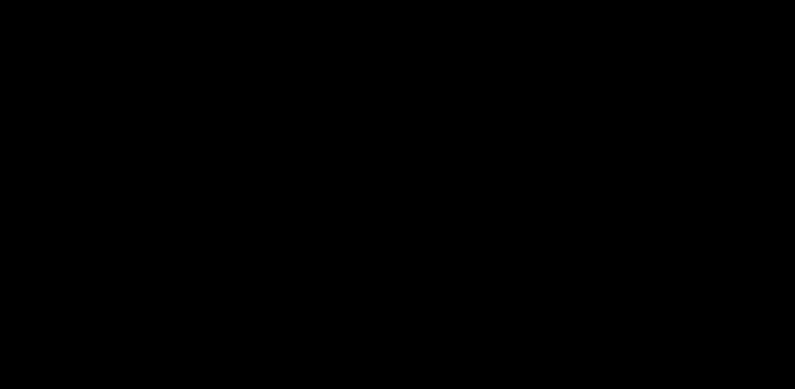
1. **Mongodb4.x 的使用**











二、**Mongodb** 账户权限配置

**1**、第一步创建超级管理用户

use admin

db.createUser({

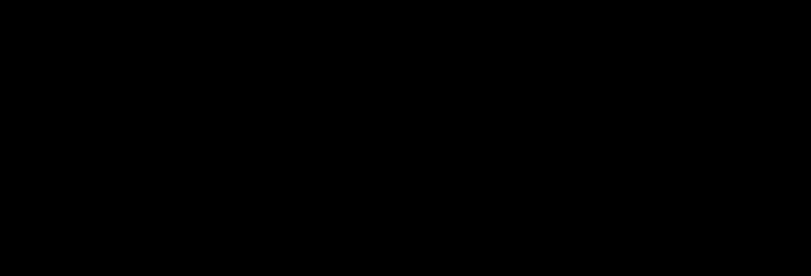
user:'admin',

pwd:'123456',

roles:[{role:'root',db:'admin'}]

})

**23**、第三步重启**mongodb** 服务



**4、第四步用超级管理员账户连接数据库**

mongo admin -u 用户名 -p 密码

mongo 192.168.1.200:27017/test -u user -p password

5、第五步给 eggcms 数据库创建一个用户 只能访问 eggcms 不能访问其他数据库

use eggcms

db.createUser(

{

user: "eggadmin", pwd: "123456", roles: [ { role: "dbOwner", db: "eggcms" } ]

}

)

**三、Mongodb 账户权限配置中常用的命令**

1、show users; #查看当前库下的用户

2、db.dropUser("eggadmin") #删除用户

3、db.updateUser( "admin",{pwd:"password"}); #修改用户密码

4、db.auth("admin","password"); #密码认证

**四、Mongodb 数据库角色**

1.数据库用户角色：read、readWrite;

2.数据库管理角色：dbAdmin、dbOwner、userAdmin；

3.集群管理角色：clusterAdmin、clusterManager、clusterMonitor、hostManager；

4.备份恢复角色：backup、restore；

5.所有数据库角色：readAnyDatabase、readWriteAnyDatabase、userAdminAnyDatabase、

dbAdminAnyDatabase

6.超级用户角色：root

参考：https://www.cnblogs.com/zzw1787044/p/5773178.html

**五、连接数据库的时候需要配置账户密码**

const url = 'mongodb://admin:123456@localhost:27017/';