1.MongoDB介绍

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。

说明:BSON是一种计算机数据交换格式，主要被用作MongoDB数据库中的数据存储和网络传输格式。它是一种二进制表示形式，能用来表示简单数据结构、关联数组（MongoDB中称为"对象"或"文档"）以及MongoDB中的各种数据类型。BSON之名缘于JSON，含义为Binary JSON（二进制JSON）。

1.1 特点

(1) 面向集合存储，易存储对象类型的数据

(2) 支持动态查询

(3) 支持完全索引，包含内部对象

(4) 支持复制和故障恢复

(5) 支持多种开发语言

(6) 使用高效的二进制数据存储，包括大型对象(如视频等)

1.2 适用场景

1）网站实时数据处理。它非常适合实时的插入、更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性。

2）缓存。由于性能很高，它适合作为信息基础设施的缓存层。在系统重启之后，由它搭建的持久化缓存层可以避免下层的数据源过载。

3）高伸缩性的场景。非常适合由数十或数百台服务器组成的数据库，它的路线图中已经包含对MapReduce引擎的内置支持。

1.3 不适用的场景如下

1）要求高度事务性的系统。

2）传统的商业智能应用。

3）复杂的跨文档（表）级联查询。

1.4 相关概念

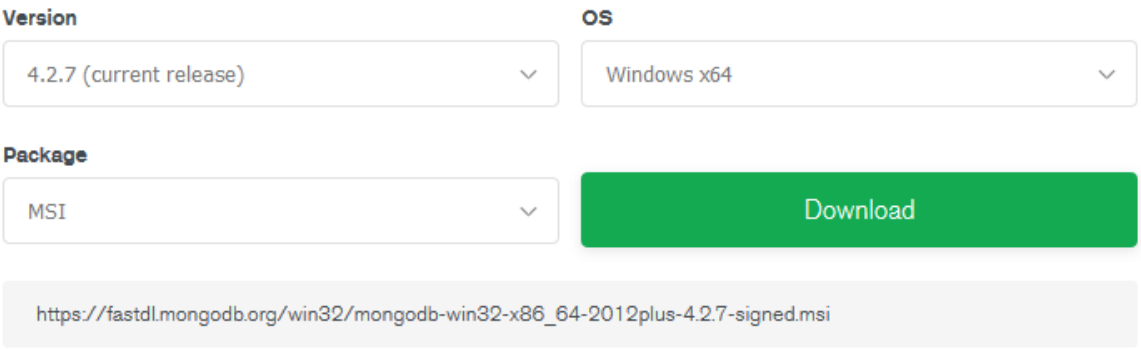
SQL术语/概念	MongoDB术语/概念	解释/说明
database	database	数据库
table	collection	数据库表/集合
row	document	数据记录行/文档
column	field	数据字段/域
index	index	索引
table joins		表连接,MongoDB不支持
primary key	primary key	主键,MongoDB自动将_id字段设置为主键

2.安装MongoDB

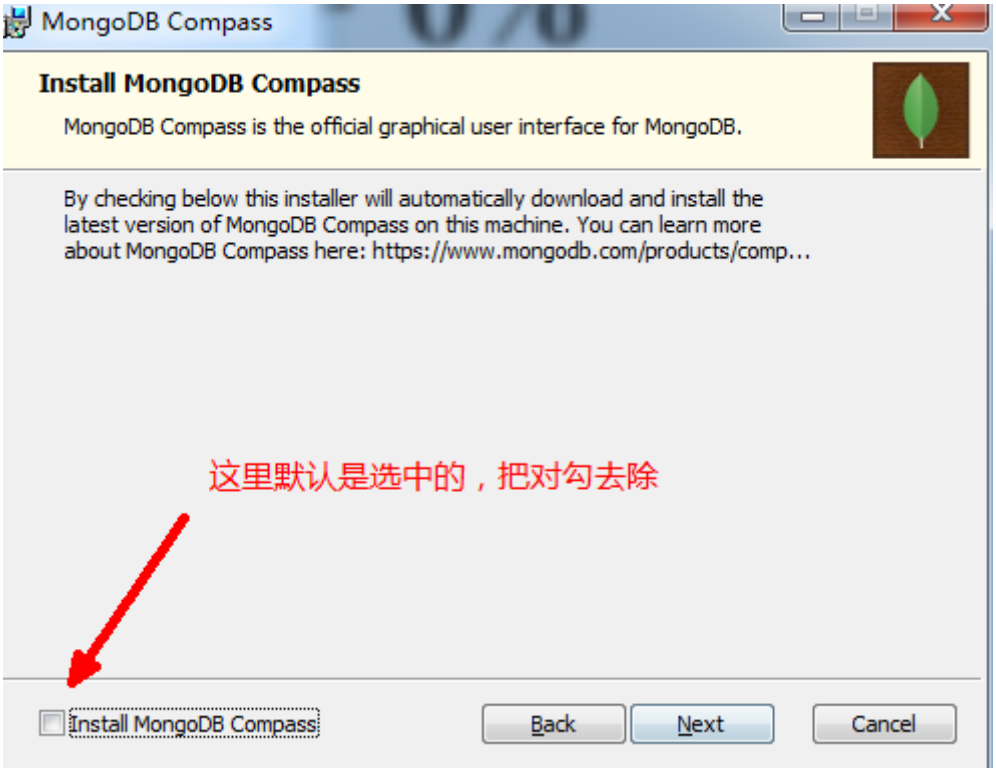
2.1 window下安装mongoDB

第一步:下载压缩包

地址:https://www.mongodb.com/download-center/community



第二步:双击.msi文件，按照提示安装即可

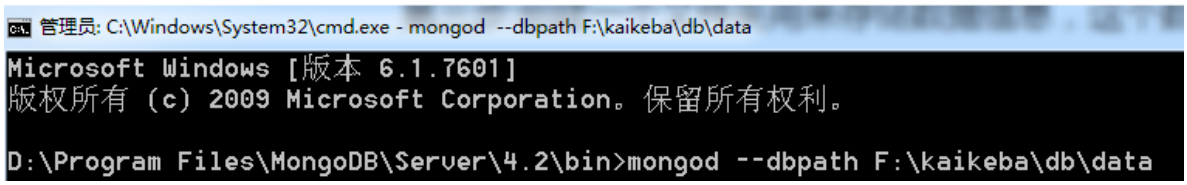


第三步:创建一个文件夹用来存储数据信息，这个数据目录不会主动创建，我们在安装完成后需要创建它

(如:F:\kaikeba\db\data等)

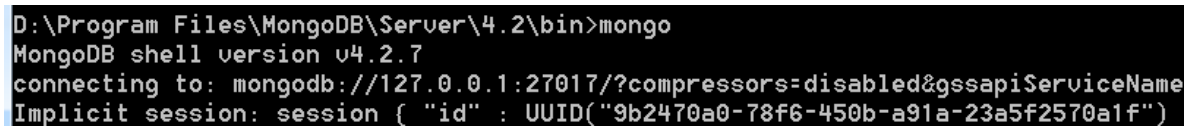
第四步:命令行下运行MongoDB服务器

为了从命令提示符下运行 MongoDB 服务器，你必须从 MongoDB 目录的 bin 目录中执行 mongod.exe 文件。

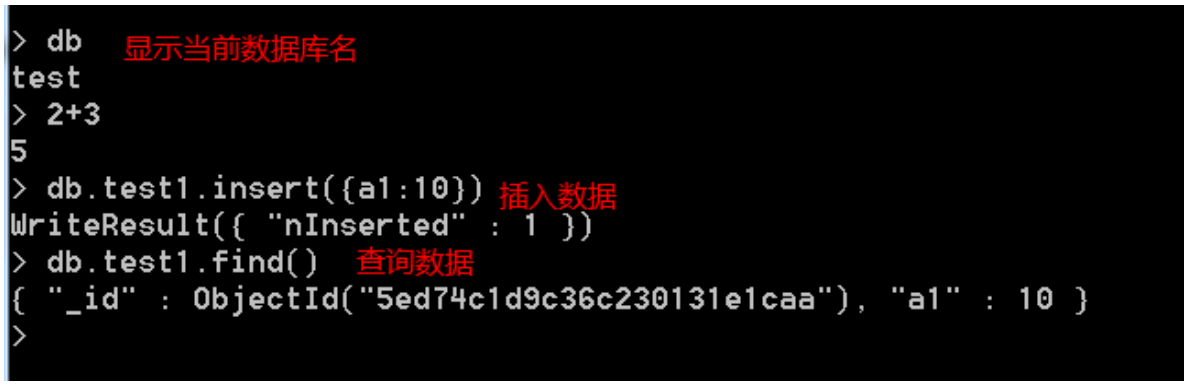


第五步:连接服务器

新打开一个cmd界面，在mongoDB目录的bin目录中执行mongo.exe文件



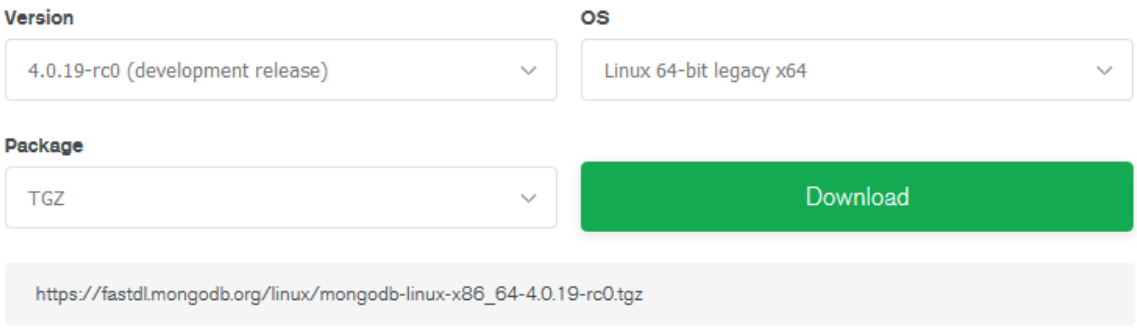
测试:



2.2 Liunx下安装mongoDB

第一步:下载安装包

下载地址：https://www.mongodb.com/download-center#community



第二步:上传到linux,并解压

```
[root@localhost local]# tar -zxvf mongodb-linux-x86_64-4.0.19-rc0.tgz
```

第三步:移动文件到自定义目录

```
[root@localhost local]# mv mongodb-linux-x86_64-4.0.19-rc0
/home/admin/myapps/mongodb
[root@localhost local]# cd /home/admin/myapps/mongodb/
[root@localhost mongodb]# ll
total 200
drwxr-xr-x. 2 root root  4096 Jun  3 00:10 bin
-rw-r--r--. 1 root root 30608 Jun  2 15:01 LICENSE-Community.txt
-rw-r--r--. 1 root root 16726 Jun  2 15:01 MPL-2
-rw-r--r--. 1 root root  2601 Jun  2 15:01 README
-rw-r--r--. 1 root root 60005 Jun  2 15:01 THIRD-PARTY-NOTICES
-rw-r--r--. 1 root root 81355 Jun  2 15:02 THIRD-PARTY-NOTICES.gotools
```

第四步:系统profile配置

```
[root@localhost mongodb]# vi /etc/profile
```

编辑代码:

```
export mongodb_home=/home/admin/myapps/mongodb    //mongodb的安装目录
export PATH=$PATH:$mongodb_home/bin
```

保存后，重启系统配置

```
[root@localhost mongodb]# source /etc/profile
```

第五步:创建数据库目录

```
[root@localhost myapps]# pwd
/home/admin/myapps
[root@localhost myapps]# mkdir mongodbdata    //存放数据库的目录
```

第六步:创建日志文件和配置文件

```
[root@localhost myapps]# mkdir logs
[root@localhost myapps]# cd logs
[root@localhost logs]# touch mongodb.log
[root@localhost logs]# cd ..
[root@localhost myapps]# cd mongodb
[root@localhost mongodb]# cd bin
[root@localhost bin]# vi mongodb.conf  //配置文件存放在bin目录下
```

mongodb.conf配置内容:

```
dbpath = /home/admin/myapps/mongodbdata #数据文件存放目录
logpath = /home/admin/myapps/logs/mongodb.log #日志文件存放目录
port = 27017  #端口
fork = true  #以守护程序的方式启用，即在后台运行
```

第七步:启动mongodb服务端

```
[root@localhost bin]# ./mongod -f mongodb.conf
about to fork child process, waiting until server is ready for connections.
forked process: 2962
child process started successfully, parent exiting
```

第八步:运行客户端

```
[root@localhost bin]# ./mongo
```

3.mongodb支持的数据类型

3.1 null

null用于表示空值或不存在的字段。示例如下：

```
{"x" : null}
```

3.2 布尔类型

布尔型数据有true和false两个值。示例如下：

```
{"x" : true}
```

3.3 数值类型

在Mongo shell中，默认使用64位浮点型数据。因此，会有以下两种数值形式：

```
{"x" : 2.32}//或{"x" : 2}
```

对于整数类型，可以使用NumberInt()(位有符号整型)或NumberLong()(8位有符号整型)方法进行转换。示例如下：

```
{"x" : NumberInt(2)}
{"x" : NumberLong(2)}
```

3.4 字符串

MongoDB中字符串类型使用UTF-8编码的字符表示。示例如下：

```
{"x" : "123@qq.com"}
```

3.5 日期类型

MongoDB中日期使用时间戳表示，单位为毫秒，不存储时区。示例如下：

```
{"x" : new Date()}
```

创建日期对象时应该使用new Date()，而非构造函数Date()。将构造函数作为函数时返回的日期格式是字符串，而非日期对象（与JavaScript工作机制有关）。

3.6 正则表达式

MongoDB中可使用与JavaScript相同的正则表达式进行查询筛选等。示例如下：

```
{"x" : /kaikeba/i}
```

3.7 数组

数据集可以用数组格式存储，与JavaSript中的数组表示相同。示例如下：

```
{"x" : ["kaikeba", "kaikeba.com"]}
```

数组中可以包含不同类型的数据元素，包括内嵌文档和数组等。所有MongoDB中键-值对支持的数据类型都可以用做数组的值。

3.8 内嵌文档

文档中可以嵌套一个子文档。在MongoDB文档总大小限制为16MB，建议使用子文档的形式组织数据，子文档查询效率要高于多键查询。示例如下：

```
{"x" : {"kaikeba" : "kaikeba.com"}}
```

文档可以做为键的值，即：内嵌文档。MongoDB与关系型数据库相比，最大的优势就是内嵌文档。与关系型数据库的扁平化数据结构相比，使用内嵌文档可以数据的组织方式更加自然。

3.9 _id和ObjectId

MongoDB中每个文档都有一个"*id"键，*"id"可以是任何类型，不指"_id"时MongoDB会生成一个ObjectId对象。。示例如下：

```
{"_id" : ObjectId()}
```

ObjectId是一个12字节（24个十六进制数字）的存储空间，ObjectId的12字节数据组织方式如下：

```
 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11
     时间戳      |  机器码     |  PID  |  计数器
```

对于如下一个ObjectId，其各位字符含义为：

```
{"_id" : ObjectId("5444cce6aef53b0f343e2b9b")}
/* 上面ObjectId各位字符含义如下 */
//5444cce6，第0~3字节（第1~8位）为时间戳
//aef53b，第4~6字节（第9~14位）为机器码
//0f34，第7~8字节（第15~18位）为进程ID
//3e2b9b，第9~11字节（第19~24位）为自增计数器
```

3.10 代码

MongoDB的文档和代码中可以包括JavaScript代码。示例如下：

```
{"x" : function(){ /*这里是一段JavaScript代码*/}}
```

3.11 二进制数据

二进制数据是一个二进制字节的字作串，要保存非UTF-8字符到数据库中，只能使用十进制数据。

4.mongoDB常用指令

登录:

```
>mongo ip地址    //默认端口号27017
```

注意:如果没有指定bind_ip，会导致mongodb默认绑定为127.0.0.1，导致外部无法访问

修改mongodb.conf文件:

```
bind_ip=0.0.0.0
```

退出:

```
>exit
```

查看数据库(数据库中至少有一条数据，此时的数据库才会显示出来)

```
>show dbs
```

切换数据库

```
>use 数据库名 //这个指令也可以直接创建数据库，但只有添加数据后，
                show dbs才能看到该数据库
```

查看当前数据库，默认数据库:test

```
>db
```

查看所有的数据集

```
>show collections
```

删除当前数据库

```
>db.dropDatabase()
```

创建集合（相当于创建表）

```
>db.createCollection("user1")
```

删除集合

```
>db.collectionName.drop()
```

集合重命名

```
>db.oldCollectionName.renameCollection("newName")
```

新增数据

```
>db.collectionName.insert({"key":value,"key":value})
或
>db.collectionName.save({"key":value,"key":value})
```

查询所有数据

```
>db.collectionName.find()
```

条件查询-find()以非结构化的方式展示文档

```
>db.collectionName.find({"age":26}) //查询等值关系
>db.collectionName.find({age : {$gt : 100}})  // 大于100
>db.collectionName.find({age : {$gte : 100}})  //大于等于100
>db.collectionName.find({age : {$lt : 150}}) //小于150
>db.collectionName.find({age : {$lte : 150}}) //小于等于150
>db.collectionName.find({age : {$lt :200, $gt : 100}})  //大于100，小于200
```

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.collectionName.find().pretty()
```

操作	格式	实例	sql中的类似语句
等于	{:}	db.collectionName.find({"by":"java"}).pretty()	where by = 'java'
小于	{:{$lt:}}	db.collectionName.find({"likes":{$lt:50}}).pretty()	where likes < 50
小于或等于	{:{$lte:}}	db.collectionName.find({"likes":{$lte:50}}).pretty()	where likes <= 50
大于	{:{$gt:}}	db.collectionName.find({"likes":{$gt:50}}).pretty()	where likes > 50
大于或等于	{:{$gte:}}	db.collectionName.find({"likes":{$gte:50}}).pretty()	where likes >= 50
不等于	{:{$ne:}}	db.collectionName.find({"likes":{$ne:50}}).pretty()	where likes != 50

and 关系

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:

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

示例:

```
> db.collectionName.find({"by":"java", "title":"MongoDB学习"}).pretty()
```

or关系

MongoDB OR 条件语句使用了关键字 **$or**,语法格式如下：

语法:

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

示例:

```
>db.collectionName.find({$or:[{"by":"java"},{"title": "MongoDB学习"}]}).pretty()
```

清空集合数据

```
>db.collectionName.remove({}) //条件删除:remove({key:value})
                              //删除满足条件的一条数据:remove({key:value},1)
```

查询一条数据

```
>db.collectionName.findOne();
```

查询指定列

```
> db.collectionName.find({},{name:1,age:1,sex_orientation:true})
```

查询指定字段的数据，并去重

```
> db.collectionName.distinct('sex')
```

对结果集排序

```
> db.collectionName.find().sort({salary:1})   //升序
> db.collectionName.find().sort({salary:-1}) //降序
```

统计记录数

```
> db.collectionName.find().count()
```

查询限定条数

```
>db.collectionName.find().limit(number)
```

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。

```
>db.collectionName.find().limit(NUMBER).skip(NUMBER)
```

更新数据

```
db.collectionName.update(<query>,<update>,{upsert: <boolean>, multi: <boolean>)
```

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，也可以理解为sql update查询内set后面的

upsert : 可选，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。

示例:

```
> db.collectionName.update({name:'tom'},{$set:{age:23}},false,true)
```

5.Java调取mongoDB

第一步：添加mongodb-java-driver驱动包

```
<dependencies>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>3.2.2</version>
    </dependency>
</dependencies>
```

第二步:连接数据库并操作数据

```
  public static void main(String[] args) {
        // 1.连接到 mongodb 服务
        MongoClient mongoClient = new MongoClient("192.168.197.133", 27017);
        // 2.连接到数据库
        MongoDatabase mongoDatabase = mongoClient.getDatabase("mydb1");
        //3.创建集合
        // mongoDatabase.createCollection("student");
        //获得集合
        MongoCollection<Document> collection =
mongoDatabase.getCollection("users");
        //4.新增
        Document document = new Document("stu1name", "张三");
        document.append("stu1age", 29);
        document.append("sex", "男");
        collection.insertOne(document);
        //插入多行时，将document对象添加到List集合中，调取insertMany()
        //5.获得所有文档
        FindIterable<Document> documents = collection.find();
        MongoCursor<Document> iterator = documents.iterator();
```

```
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
        //6.更新文档updateOne(),updateMany(条件表达式，新值表达式)
        // collection.updateMany(Filters.eq("sex","男"),new Document("$set",new
Document("name","testzhangsan")));
        //7.删除文档
        //删除20岁用户的信息
        collection.deleteOne(Filters.eq("age",20));
    }
```

6.mongDB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。

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。

6.1 创建索引的语法

```
>db.collection.createIndex(keys, options)
```

options取值的含义:

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。

Parameter	Type	Description
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.

6.2 索引分类

(1) 默认索引

MongoDB有个默认的“*id”的键，相当于"主键"的角色。集合创建后系统会自动创建一个索引在*“id”键上，它是默认索引，索引名叫“_id_”，是无法被删除的。我们可以通过以下方式查看：

```
>db.collectionName.getIndexes()
```

(2) 单列索引

在单个键上创建的索引就是单列索引，例如我们要在Users集合上给title键创建一个单列索引，语法如下：

（1表示正序，-1逆序）

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

(3) 组合索引

另外，我们还可以同时对多个键创建组合索引。如下代码创建了按照“UserId”正序，“UserName”逆序的组合索引:

```
>db.collectionName.createIndex({"userid":1,"username":-1})
```

(4) 唯一索引

唯一索引限制了对当前键添加值时，不能添加重复的信息。值得注意的是，当文档不存在指定键时，会被认为键值是“null”，所以“null”也会被认为是重复的，所以一般被作为唯一索引的键，最好都要有键值对。

对“UserId”创建唯一索引(这时候最后一个参数为“true”)：

```
>db.collectionName.CreateIndex({"UserId":1}, { unique: true });
```

(5)TTL索引

TTL指生命周期的意思。即存储的document存储带有过期时间属性，超过生命周期自己主动删除。像日志数据、系统自己主动产生的暂时数据、会话数据等均符合这一场景。
构建方式如：

```
    db.log_events.createIndex( { "createdAt": 1 }, { expireAfterSeconds: 3600 }
)
```

(6) 删除索引

新手常陷入的误区是，认为集合被删除，索引就不存在了。关系型数据库中，表被删除了，索引也不会存在。在MongoDB中不存在删除集合的说法，就算集合数据清空，索引都是还在的，要移除索引还需要手工删除。

```
>db.collectionName.dropIndexes()
```

删除集合指定索引

```
>db.collectionName.dropIndex("索引名称")
```

说明:drop()集合时，索引也会删除，remove()集合时，索引仍然存在

示例代码:

```
> db.users.remove({})
WriteResult({ "nRemoved" : 3 })
> db.users.getIndexes()
[
    {
        "v" : 2,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "mydb1.users"
    }
]
> db.users.drop()
true
> db.users.getIndexes()
[ ]
```

7.mongoDB备份与恢复

7.1 mongodump命令来备份数据

该命令可以导出所有数据到指定目录中。

mongodump命令可以通过参数指定导出的数据量级转存的服务器。

语法

```
>mongodump -h dbhost -d dbname -o dbdirectory
```

- -h：

  MongDB所在服务器地址，例如：127.0.0.1，当然也可以指定端口号：127.0.0.1:27017

- -d：

  需要备份的数据库实例，例如：test

- -o：

备份的数据存放位置，例如：c:\data\dump，当然该目录需要提前建立，在备份完成后，系统自动在dump目录下建立一个test目录，这个目录里面存放该数据库实例的备份数据。

示例

在本地使用 27017 启动你的mongod服务。打开命令提示符窗口，进入MongoDB安装目录的bin目录输入命令mongodump:

```
>mongodump
```

执行以上命令后，客户端会连接到ip为 127.0.0.1 端口号为 27017 的MongoDB服务上，并备份所有数据到 bin/dump/ 目录中。命令输出结果如下：

```
[root@localhost bin]# mongodump
    writing admin.system.version to
    done dumping admin.system.version (1 document)
    writing mydb1.users to
    writing mydb1.student to
    done dumping mydb1.users (4 documents)
    done dumping mydb1.student (0 documents)
```

mongodump 命令可选参数列表如下所示：

语法	描述	实例
mongodump --host HOST_NAME --port PORT_NUMBER	该命令将备份所有MongoDB数据	mongodump --host runoob.com --port 27017
mongodump --dbpath DB_PATH --out BACKUP_DIRECTORY		mongodump --dbpath /data/db/ --out /data/backup/
mongodump --collection COLLECTION --db DB_NAME	该命令将备份指定数据库的集合。	mongodump --collection mycol --db test

7.2 MongoDB数据恢复

mongodb使用 mongorestore 命令来恢复备份的数据。

语法

```
>mongorestore -h <hostname><:port> -d dbname <path>
```

- --host <:port>, -h <:port>：

  MongoDB所在服务器地址，默认为： localhost:27017

- --db , -d ：

  需要恢复的数据库实例，例如：test，当然这个名称也可以和备份时候的不一样，比如test2

- --drop：

  恢复的时候，先删除当前数据，然后恢复备份的数据。就是说，恢复后，备份后添加修改的数据都会被删除，慎用哦！

- ：

  mongorestore 最后的一个参数，设置备份数据所在位置，例如：c:\data\dump\test。

  你不能同时指定 和 --dir 选项，--dir也可以设置备份目录。

- --dir：

  指定备份的目录

  你不能同时指定 和 --dir 选项。

接下来我们执行以下命令:

```
>mongorestore
```

执行以上命令输出结果如下：

```
[root@localhost bin]# mongorestore
.....
    no indexes to restore
    finished restoring mydb1.users (4 documents)
    done
```

8.集群搭建

集群搭建方式之一就是mongoDB复制集，即一组mongod的进程。他们维护同一个数据集合。复制集保证了数据的可靠性和高读取能力。

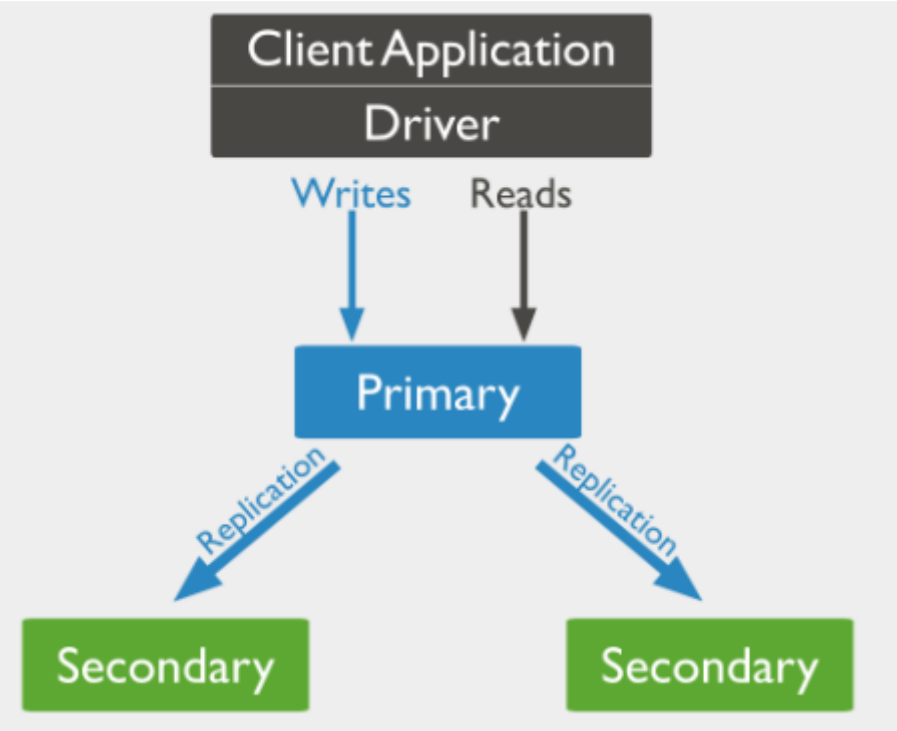
8.1 机制

一组复制集就是一组mongod实例管理同一个数据集，实例key在不同的机器上，实例包含主实例(primary),接受所有的写操作，其他的属于副本实例(Secondary),从服务器保持与主服务器数据同步，类似于redis中的主从复制。

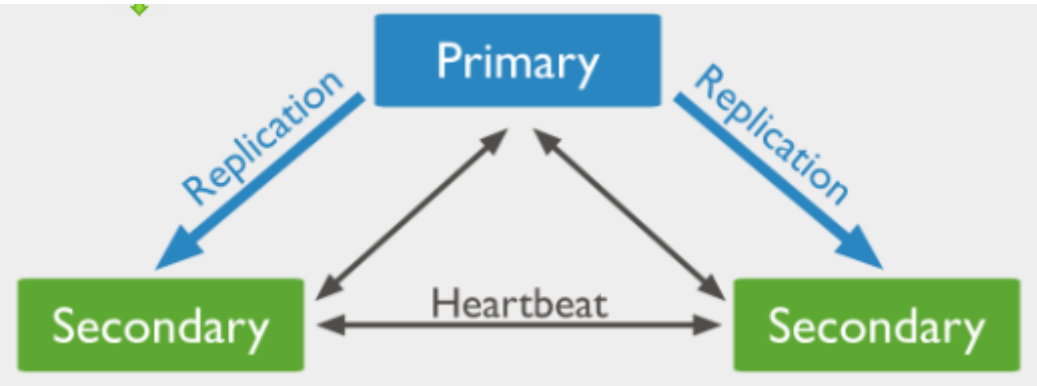
每个复制集还有一个仲裁者(Arbiter),仲裁者的任务就是通过心跳机制来确认集群中集合的数量，并在选举主服务器的过程中进行裁决。仲裁者并不存储数据，性质等价于redis中的哨兵机制。

8.2 架构

在数据承载节点中，一个且只有一个成员被视为主节点，而其他节点则被视为辅助节点。节点接收所有写入操作，一个副本集只能有一个主实例能够写入，主节点记录所有变更到它的记录



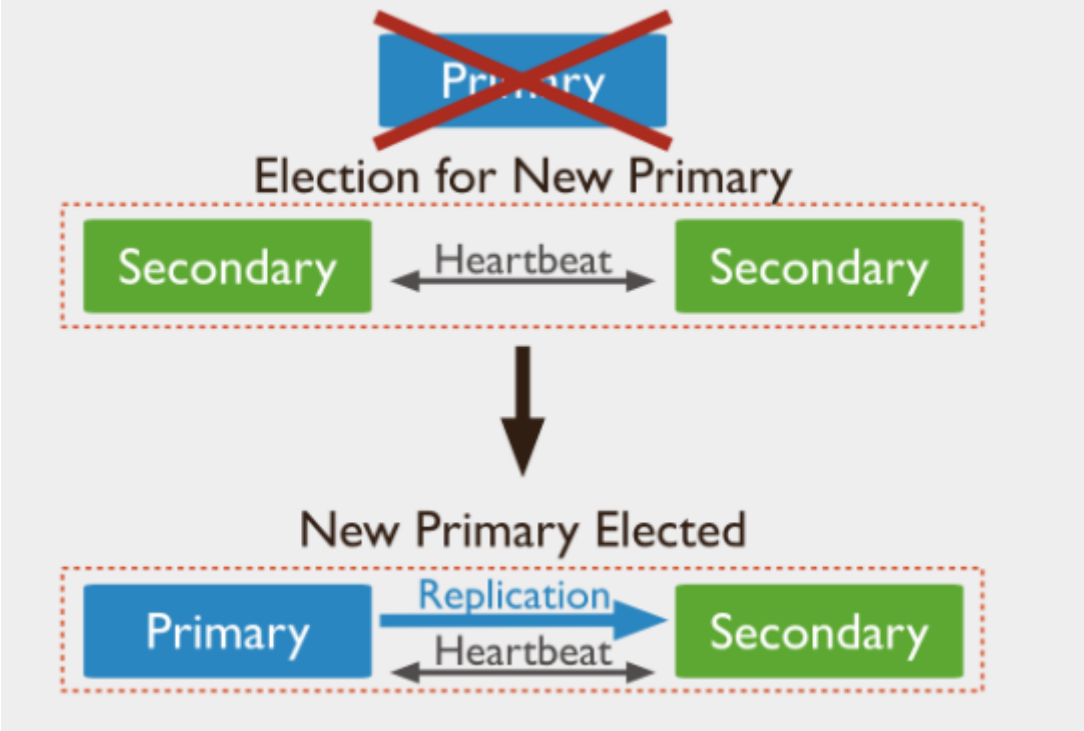
辅助节点复制主节点的 oplog 并将操作应用于数据集。



也可以将一个额外的实例作为仲裁者添加到副本集。仲裁员不维护数据集，仲裁器的目的是通过响应其他副本集成员的心跳和选择请求来维护副本集中的仲裁。因为它们不存储数据集，所以仲裁器是提供副本集仲裁功能的一种好方法。

与具有数据集的完全功能副本集成员相比，仲裁器的资源成本更低，如果副本集的成员数为偶数，则添加一个仲裁器以在初选中获得多数票。

当一个主服务器在超过配置的周期（默认为 10 秒）内未与该组的其他成员通信时，符合条件的辅助服务器将要求选择将其自身指定为新的主服务器。集群试图完成新的初选并恢复正常操作。



8.3 搭建步骤

(1)准备三台虚拟机服务器，并各自安装好mongoDB

注：为了保证复制集中三个服务器之间正常连接，请保证三个服务器的防火墙都已关闭！

```
192.168.132:27017
192.168.133:27017
192.168.134:27017
```

(2)修改mongodb.conf文件，添加replSet配置(三台都需要修改成同一个名称)，然后启动服务器

```
#复制集名称
replSet=rep1
```

(3)初始化复制集

登录任意一台执行初始化操作

```
rs.initiate({_id:'rep1',members:[{_id:1,host:'192.168.197.132:27017'},
{_id:2,host:'192.168.197.133:27017'},{_id:3,host:'192.168.197.134:27017'}]})
```

说明: _id 指复制集名称，members指复制集服务器列表，数组中的_id是服务器唯一的id,host服务器主机ip

(4)查看集群状态

```
rs.status()
```

(5)测试

```
#添加数据
db.users.insert({"name":"lisi","age":11})
#查询数据
db.users.find()
#切换到从数据库查询数据
如果不允许查询，是因为默认情况下从数据库是不允许读写操作的，需要设置。
>rs.slaveOK()
执行该命令后可以查询数据
```

(6)测试复制集主从节点故障转移功能

```
#关闭主数据库,注意从数据库的变化
>db.shutdownServer()
```

(7) 主复制集添加仲裁者(arbiter)

现在我们的环境是一主两从，仲裁者对偶数集群有效。需要停止一个从机，在主服务器中运行下面命令

```
rs.remove("ip:端口号")  //删除从节点
```

在一主一从关系中，任意节点宕机都无法选举出主节点，无法提供写操作，此时需要加入仲裁者节点即可。

```
rs.addArb("ip:端口号")
```