咕泡学院 JavaVIP 高级课程教案

MongoDB 数据库(第 2 版)

第一章

MongoDB 设计原理及

常用命令

关于本文档

主题	咕泡学院 Java VIP 高级课程教案--MongoDB 数据库（第二版）
主讲	Tom 老师
适用对象	咕泡学院 Java 高级 VIP 学员及 VIP 授课老师
数据库版本	MongoDB Community Server 4.0.1
客户端版本	RoboMongo 0.9.0

一、MongoDB 中的应用场景及设计原理

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。在这里我们有必要先简单介绍一下非关系型数据库（NoSQL）

1.1、什么是 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。

1.2、关系型数据库 PK 非关系型数据库

关系型数据库	NoSQL 数据库
高度组织化结构化数据	代表着不仅仅是 SQL
结构化查询语言（SQL）	没有声明性查询语言
数据和关系都存储在单独的表中	没有预定义的模式
数据操作语言，数据定义语言	键-值对存储，列存储，文档存储，图形数据库
严格的一致性	最终一致性，而非 ACID 属性
基础事务	非结构化和不可预知的数据
	CAP 定理
	高性能，高可用性和可伸缩性

1.3、NoSQL 数据库分类

类型	典型代表	特点
列存储	Hbase Cassandra Hypertable	顾名思义，是按照列存储数据的。最大的特点是方便存储结构化和半结构化的数据，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 CounchDB	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 Berkelery DB Memcache 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，通过对象的方式存储数据。
XML 数据库	Berkeley DB XML BaseX	高效的存储 XML 数据，并存储 XML 的内部查询语法，比如 XQuery,Xpath。

1.4、MongoDB 的数据结构与关系型数据库数据结构对比

关系型数据库术语/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	数据列/数据字段
Index	Index	索引
Table joins		表关联/MongoDB 不支持
Primary Key	Object ID	主键/MongoDB 自动将_id 设置为主键

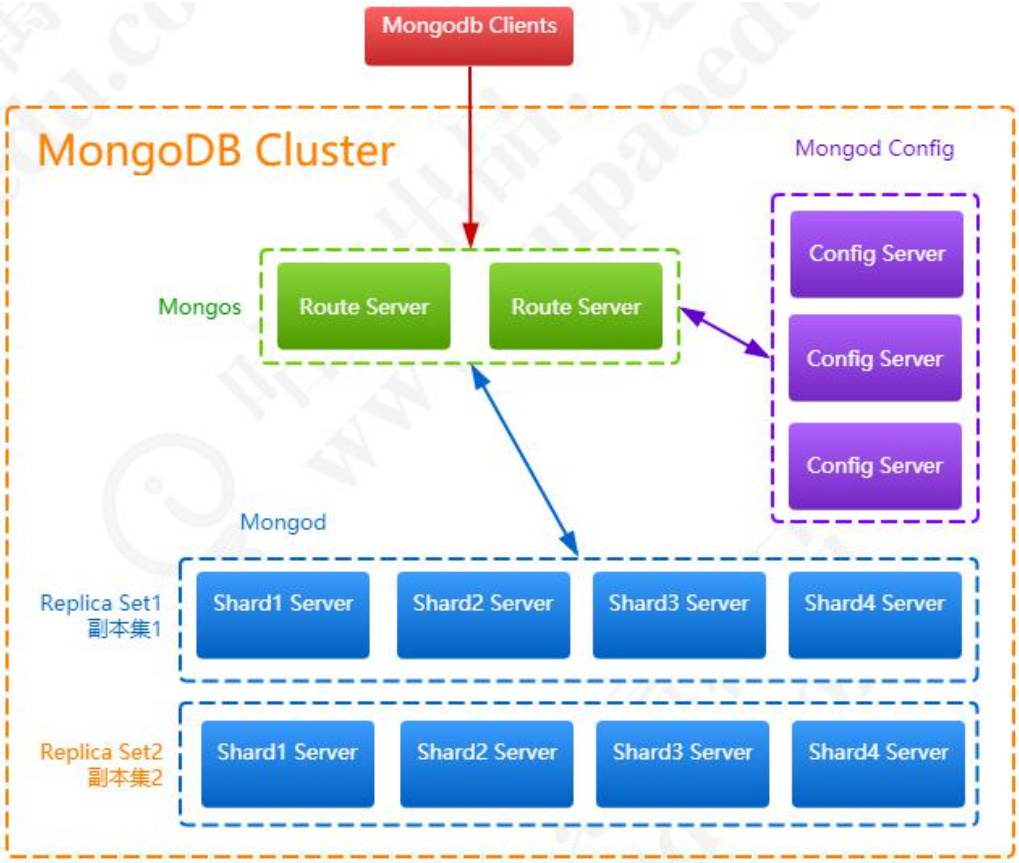
1.5、MongoDB 中的数据类型

数据类型	说明	解释	举例
Null	空值	表示空值或者未定义的对象	{“x”:null}
Boolean	布尔值	真或者假：true 或者 false	{“x”:true}
Integer	整数	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。	
Double	浮点数	双精度浮点值。	{“x”：3.14，”y”：3}
String	字符串	UTF-8 字符串	
Symbol	符号	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。	
ObjectID	对象 ID	对象 ID。用于创建文档的 ID。	{“id”：ObjectId()}
Date	日期	日期时间。用 UNIX 时间格式来存储当前日期或时间。	{“date”:new Date()}
Timestamp	时间戳	从标准纪元开始的毫秒数	
Regular	正则表达式	文档中可以包含正则表达式，遵循 JavaScript 的语法	{“foo”:/testdb/i}
Code	代码	可以包含 JavaScript 代码	{“x”:function(){}}
Undefined	未定义	已废弃	
Array	数组	值的集合或者列表	{“arr”:[“a”,”b”]}
Binary Data	二进制	用于存储二进制数据。	
Object	内嵌文档	文档可以作为文档中某	{“x”:{“foo”:”bar”}}

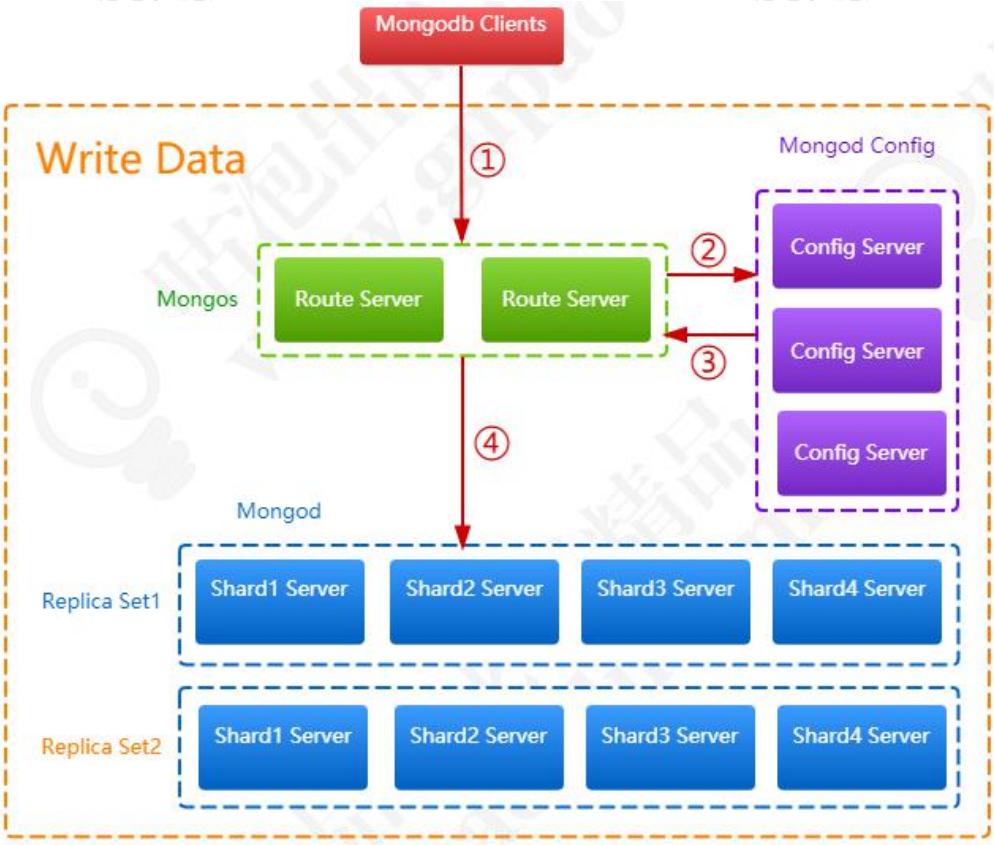
		个 key 的 value	
Min/Max keys	最小/大值	将一个值与 BSON（二进制的 JSON）元素的最低值和最高值相对比。	

1.6、图解 MongoDB 底层原理

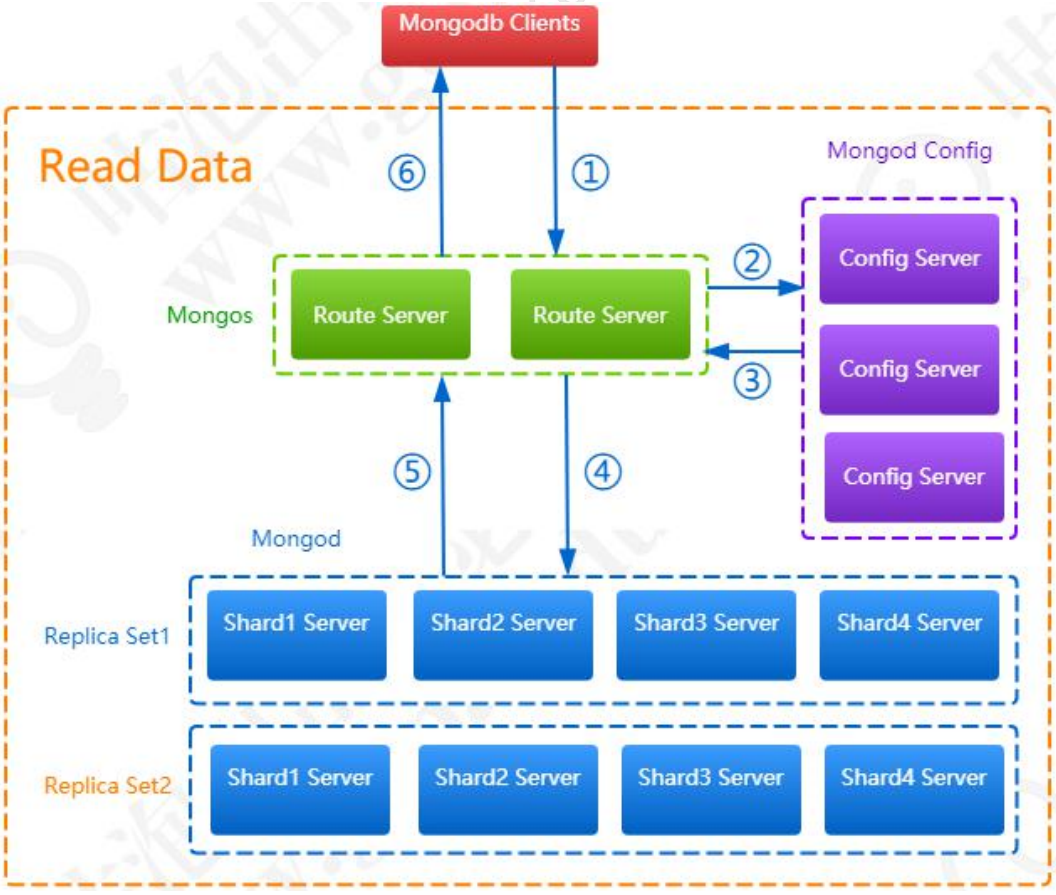
MongoDB 的集群部署方案中有三类角色：实际数据存储结点、配置文件存储结点和路由接入结点。连接的客户端直接与路由结点相连，从配置结点上查询数据，根据查询结果到实际的存储结点上查询和存储数据。MongoDB 的部署方案有单机部署、复本集（主备）部署、分片部署、复本集与分片混合部署。混合的部署方式如图：



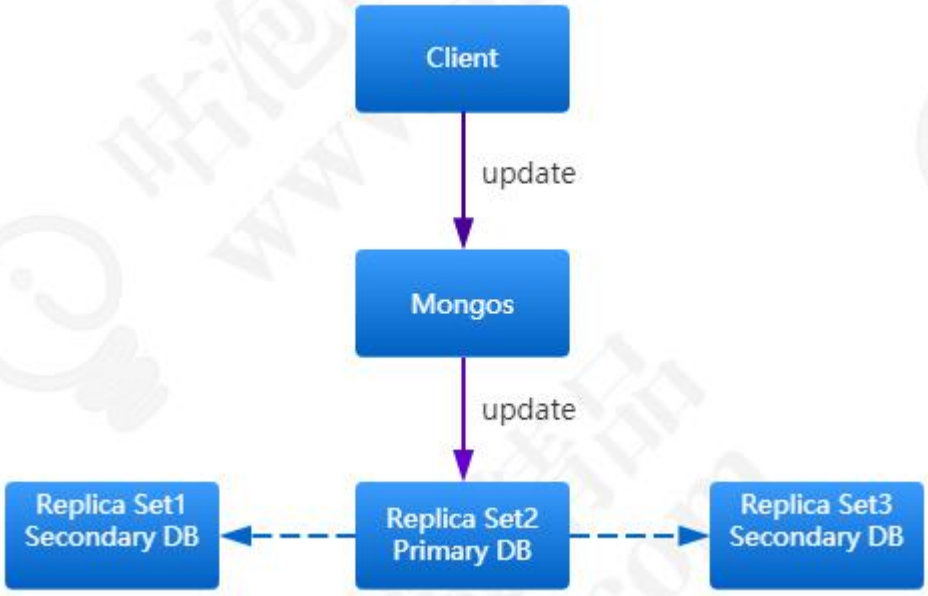
混合部署方式下向 MongoDB 写数据的流程如图：



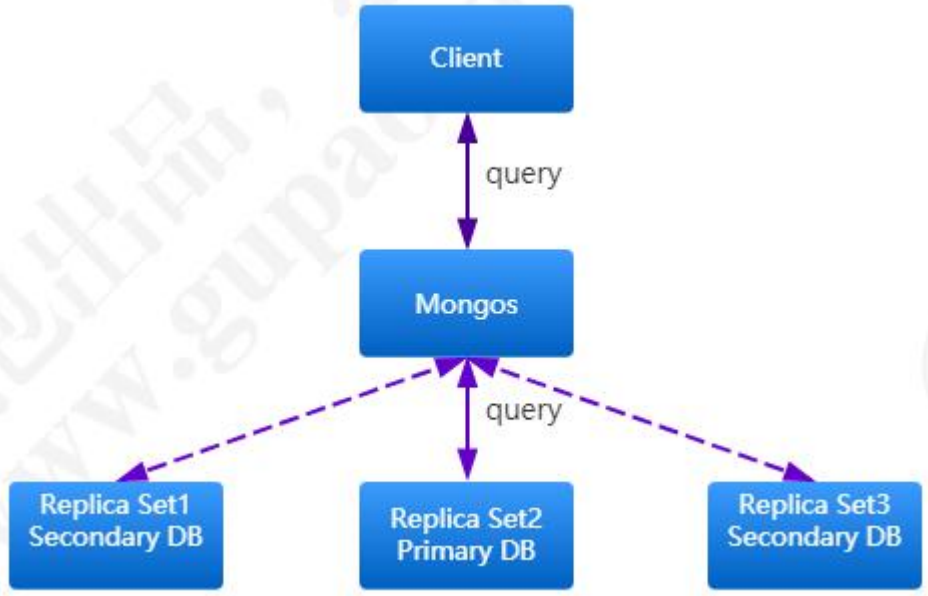
混合部署方式下读 MongoDB 里的数据流程如图：



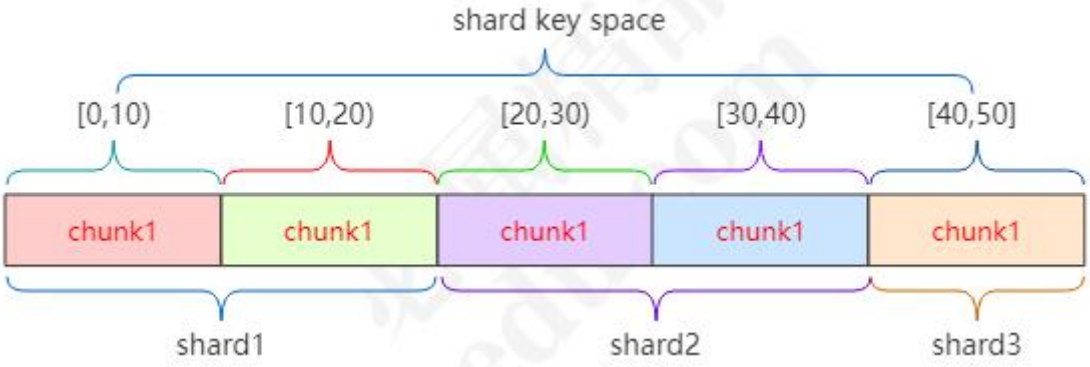
对于复本集，又有主和从两种角色，写数据和读数据也是不同，写数据的过程是只写到主结点中，由主结点以异步的方式同步到从结点中：



而读数据则只要从任一结点中读取，具体到哪个结点读取是可以指定的：



对于 MongoDB 的分片，假设我们以某一索引键（ID）为片键，ID 的区间[0,50]，划分成 5 个 chunk，分别存储到 3 个片服务器中，如图所示：



假如数据量很大，需要增加片服务器时可以只要移动 chunk 来均分数据即可。

配置结点：

存储配置文件的服务器其实存储的是片键与 chunk 以及 chunk 与 server 的映射关系，用上面的数据表示的配置结点存储的数据模型如下表：

Map1

Key range	chunk
[0,10)	chunk1
[10,20)	chunk2
[20,30)	chunk3
[30,40)	chunk4
[40,50)	chunk5

Map2

chunk	shard
chunk1	shard1
chunk2	shard1
chunk3	shard2
chunk4	shard2
chunk5	shard3

路由结点：

路由角色的结点在分片的情况下起到负载均衡的作用。

1.7、MongoDB 的应用场景和不适用场景

1、适用场景

对于 MongoDB 实际应用来讲，是否使用 MongoDB 需要根据项目的特定特点进行一一甄别，这就要求我们对 MongoDB 适用和不适用的场景有一定的了解。

根据 MongoDB 官网的说明，MongoDB 的适用场景如下：

1）网站实时数据:MongoDB 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）数据缓存:由于性能很高，MongoDB 也适合作为信息基础设施的缓存层。在系统重启之后，由 MongoDB 搭建的持久化缓存层可以避免下层的数据源过载。

3）大尺寸、低价值数据存储:使用传统的关系型数据库存储一些数据时可能会比较昂贵，在此之前，很多时候程序员往往会选择传统的文件进行存储。

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

5）对象或 JSON 数据存储:MongoDB 的 BSON 数据格式非常适合文档化格式的存储及查询。

2、不适用场景

了解了 MongoDB 适用场景之后，还需要了解哪些场景下不适合使用 MongoDB，具体如下：

1）高度事务性系统:例如银行或会计系统。传统的关系型数据库目前还是更适用于需要大量原子性复杂事务的应用程序。

2）传统的商业智能应用:针对特定问题的 BI 数据库会对产生高度优化的查询方式。对于此类应用，数据仓库可能是更合适的选择。

3）需要复杂 SQL 查询的问题。

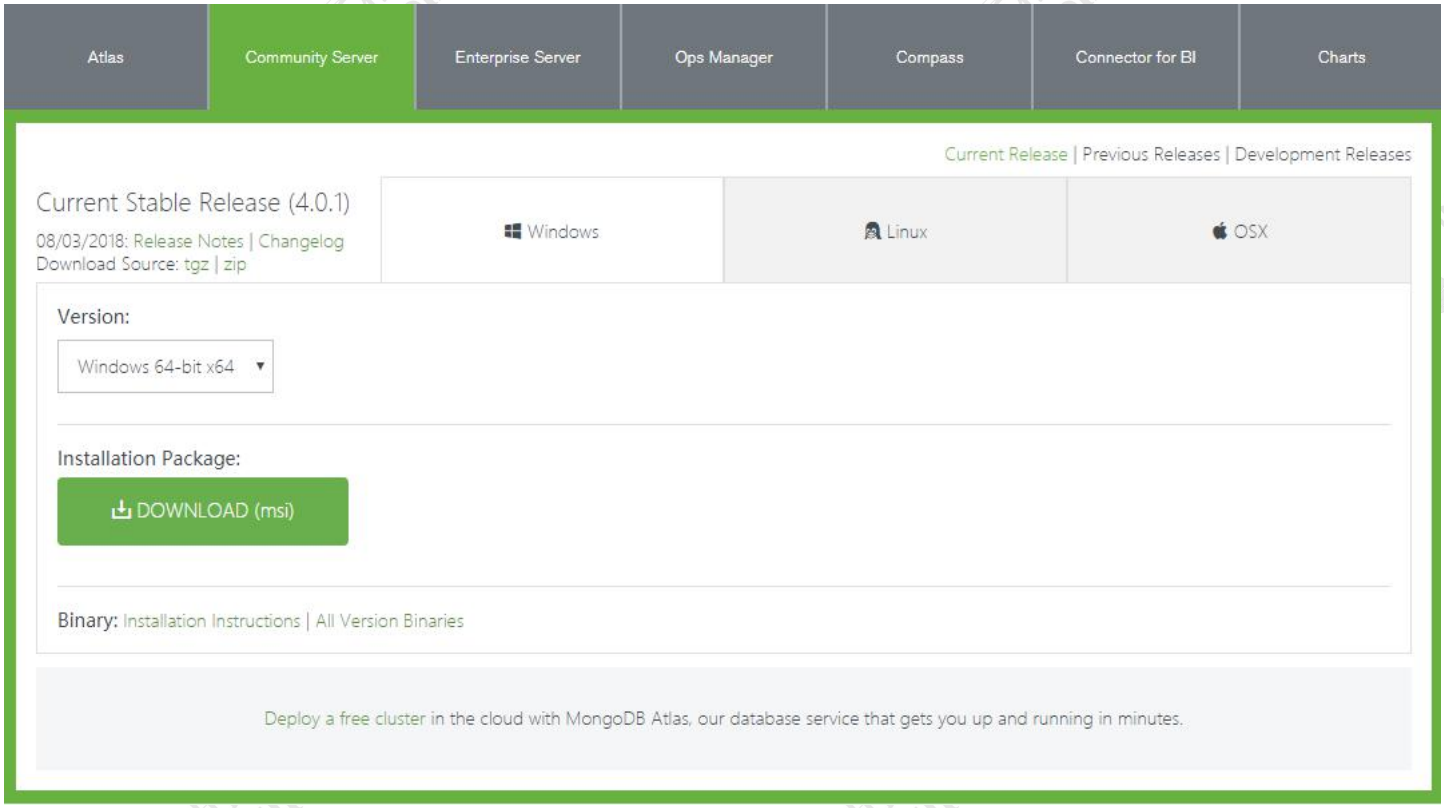
相信通过上面的说明，你已经大致了解了 MongoDB 的使用规则，需要说明一点的是，MongoDB 不仅仅是数据库，更多的使用是将 MongoDB 作为一个数据库中间件在实际应用中合理划分使用细节，这一点对于 MongoDB 应用来讲至关重要！

二、MongoDB 基本配置及常用命令

2.1、安装 MongoDB 数据库（Windows 和 Linux 环境）

打开官网：https://www.mongodb.com/download-center?jmp=nav#community

选择 Community Server 4.0.1 的版本。

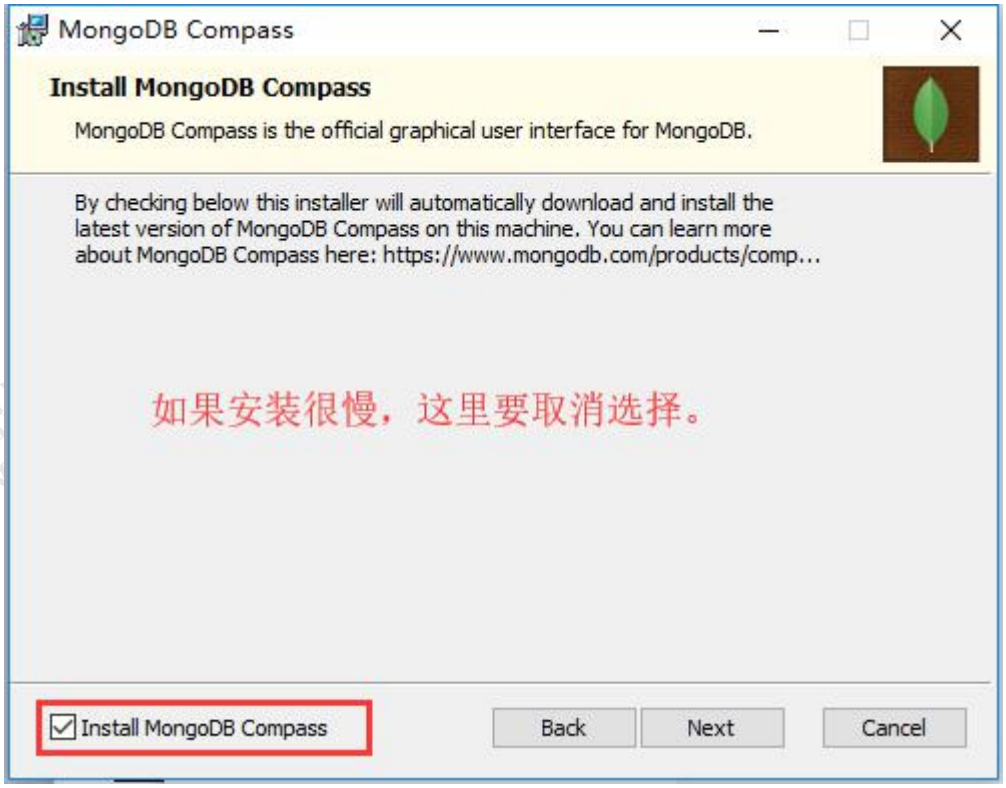


安装与启动

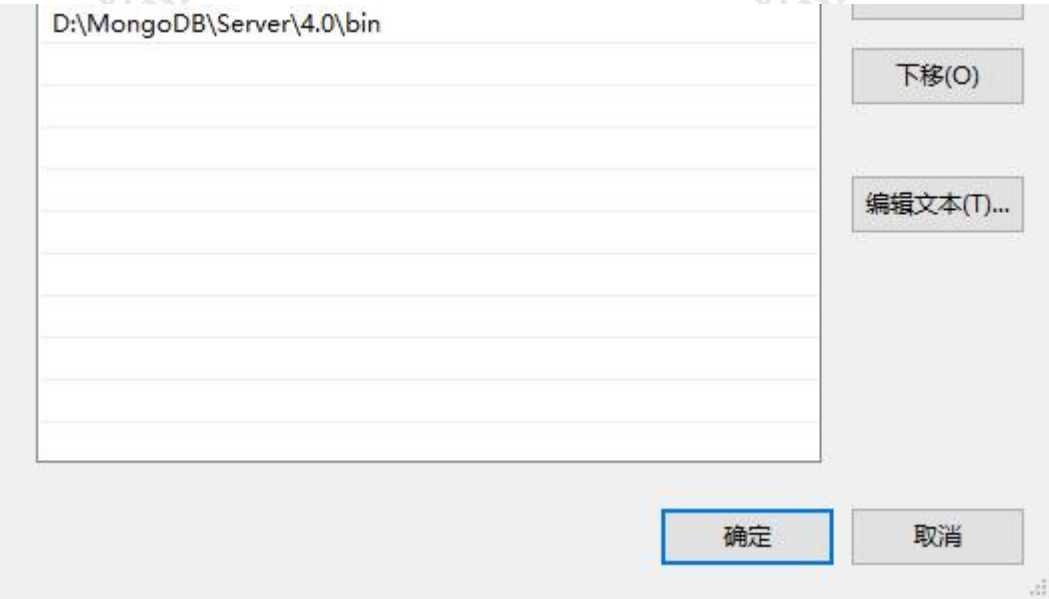
在 D 盘创建安装目录，D:\MongoDB，将解压后的文件拷入新建的文件。

在 D 盘创建一个目录，D:\MongoDB\Server\4.0\Data，用于存放 MongoDB 的数据。

执行安装，使用命令行，进入 MongDB 的安装目录，执行安装命令，并指明存放 MongoDB 的路径。



安装完成后配置环境变量




```
C:\Users\Tom>mongod -help
Options:

General options:
  -v [ --verbose ] [=arg(=v)]           be more verbose (include multiple times
                                        for more verbosity e.g. -vvvvv)
  --quiet                               quieter output
  --port arg                            specify port number - 27017 by default
  --logpath arg                         log file to send write to instead of
                                        stdout - has to be a file, not
                                        directory
  --logappend                           append to logpath instead of
                                        over-writing
  --logRotate arg                       set the log rotation behavior
                                        (rename|reopen)
  --timeStampFormat arg                 Desired format for timestamps in log
                                        messages. One of ctime, iso8601-utc or
                                        iso8601-local
  --setParameter arg                    Set a configurable parameter
  -h [ --help ]                         show this usage information
  --version                             show version information
  -f [ --config ] arg                   configuration file specifying
                                        additional options
  --bind_ip arg                         comma separated list of ip addresses to
                                        listen on - localhost by default
  --bind_ip_all                         bind to all ip addresses
  --ipv6                                enable IPv6 support (disabled by
                                        default)
```

启动数据库

```
C:\Users\Tom>mongod.exe -dbpath="D:\MongoDB\Server\4.0\data"
2018-08-21T15:34:43.475+0800 I CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2018-08-21T15:34:43.813+0800 I CONTROL  [initandlisten] MongoDB starting : pid=7344 port=27017 dbpath=D:\MongoDB\Server\4.0\data 64-bit host=GP-TOM-WORK
2018-08-21T15:34:43.813+0800 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-08-21T15:34:43.814+0800 I CONTROL  [initandlisten] db version v4.0.0
2018-08-21T15:34:43.815+0800 I CONTROL  [initandlisten] git version: 3b07af3d4f471ae89e8186d33bbb1d5259597d51
2018-08-21T15:34:43.815+0800 I CONTROL  [initandlisten] allocator: tcmalloc
2018-08-21T15:34:43.816+0800 I CONTROL  [initandlisten] modules: none
2018-08-21T15:34:43.817+0800 I CONTROL  [initandlisten] build environment:
2018-08-21T15:34:43.817+0800 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-08-21T15:34:43.818+0800 I CONTROL  [initandlisten]     distarch: x86_64
2018-08-21T15:34:43.819+0800 I CONTROL  [initandlisten]     target_arch: x86_64
2018-08-21T15:34:43.819+0800 I CONTROL  [initandlisten] options: { storage: { dbPath: "D:\MongoDB\Server\4.0\data" } }
2018-08-21T15:34:43.820+0800 W STORAGE  [initandlisten] Detected unclean shutdown - D:\MongoDB\Server\4.0\data\mongod.lock is not empty.
2018-08-21T15:34:43.823+0800 I STORAGE  [initandlisten] Detected data files in D:\MongoDB\Server\4.0\data created by the 'wiredTiger' storage engine, so setting the acti
2018-08-21T15:34:43.823+0800 W STORAGE  [initandlisten] Recovering data from the last clean checkpoint.
2018-08-21T15:34:43.824+0800 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7623M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_b
rue,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2018-08-21T15:34:44.126+0800 I STORAGE  [initandlisten] WiredTiger message [1534836884:126397][7344:140704668070656], txn-recover: Main recovery loop: starting at 1/2150
2018-08-21T15:34:44.127+0800 I STORAGE  [initandlisten] WiredTiger message [1534836884:127398][7344:140704668070656], txn-recover: Recovering log 1 through 2
2018-08-21T15:34:44.219+0800 I STORAGE  [initandlisten] WiredTiger message [1534836884:218906][7344:140704668070656], txn-recover: Recovering log 2 through 2
2018-08-21T15:34:44.272+0800 I STORAGE  [initandlisten] WiredTiger message [1534836884:272396][7344:140704668070656], txn-recover: Set global recovery timestamp: 0
2018-08-21T15:34:44.561+0800 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2018-08-21T15:34:44.802+0800 I CONTROL  [initandlisten]
2018-08-21T15:34:44.802+0800 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-08-21T15:34:44.803+0800 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-08-21T15:34:44.804+0800 I CONTROL  [initandlisten]
2018-08-21T15:34:44.804+0800 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2018-08-21T15:34:44.805+0800 I CONTROL  [initandlisten] **          Remote systems will be unable to connect to this server.
2018-08-21T15:34:44.806+0800 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify which IP
2018-08-21T15:34:44.806+0800 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --bind_ip_all to
2018-08-21T15:34:44.807+0800 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired, start the
2018-08-21T15:34:44.808+0800 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warning.
2018-08-21T15:34:44.808+0800 I CONTROL  [initandlisten]
2018-08-21T15:34:45.251+0800 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'D:/MongoDB/Server/4.0/data/diagnostic.data'
2018-08-21T15:34:45.254+0800 I NETWORK  [initandlisten] waiting for connections on port 27017
2018-08-21T15:34:46.016+0800 I FTDC     [ftdc] Unclean full-time diagnostic data capture shutdown detected, found interim file, some metrics may have been lost. OK
2018-08-21T15:35:38.535+0800 I NETWORK  [listener] connection accepted from 127.0.0.1:53559 #1 (1 connection now open)
```

注意，如果这是你的目录中有空格，会报 Invalid command 错误，将 dbpath 后面的值加上双引号即可 mongod.exe -dbpath=”D:\MongoDB\Server\4.0\data”。

最后一行显示我们的 MongoDB 已经连接到 27017,它是默认的数据库的端口；它建立完数据库之后，会在我们的 MongoDbData 文件夹下，生成一些文件夹和文件：在 journal 文件夹中会存储相应的数据文件，NoSQL 的 MongoDB，它以文件的形式，也就是说被二进制码转换过的 json 形式来存储所有的数据模型。

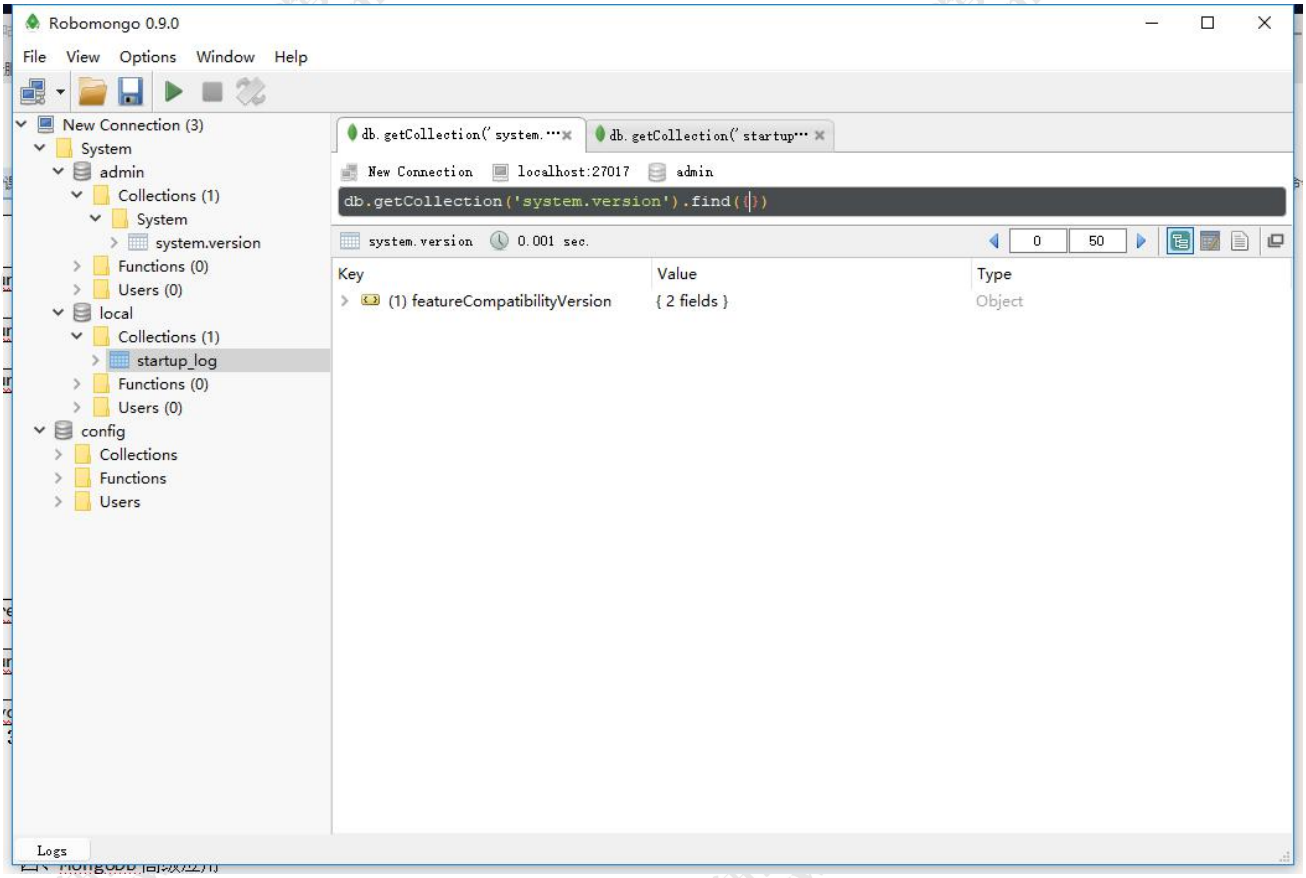
启动 MongoDB 数据库，也可以根据自己配置 mongodb.bat 文件，在 D:\MongoDB\Server\4.0\bin 中创建一个 mongodb.bat 文件，然后我们来编写这个可执行文件如下：

```
mongod --dbpath=D:\MongoDB\Server\4.0\data
```

运行 mongodb.bat 文件，MongoDB 便启动成功！

```
C:\Users\Tom>mongodb
```

2.2、安装 RoboMongo 客户端



2.3、MongoDB 常用命令

1、创建数据库

```
use testdb
```

2、创建集合

```
db.t_member.insert({name:"zhaomin",age:23})
```

3、查询

```
db.t_member.find()

db.t_member.findOne()
```

4、修改

```
#不会影响其他属性列 ， 主键冲突会报错

db.t_member.update({name:"zhaomin"},{$set:{age:18}})

#第三个参数为 true 则执行 insertOrUpdate 操作，查询出则更新，没查出则插入，

或者
```

```
db.t_member.update({name:"zhaomin"},{$set:{age:18}},true)
```

5、删除

```
#删除满足条件的第一条 只删除数据 不删除索引

db.t_member.remove({age:1})

#删除集合

db.t_member.drop();

#删除数据库

db.dropDatabase();
```

6、查看集合

```
show collections
```

7、查看数据库

```
show dbs
```

8、插入数据

```
db.t_member.insert() #不允许键值重复

db.t_member.save() #若键值重复，可改为插入操作
```

9、批量更新

```
db.t_member.update({name:"zhaomin"},{$set:{name:"zhanmin11"}},false,t

rue);
```

批量操作需要和选择器同时使用，第一个 false 表示不执行 insertOrUpdate 操作，第二个 true 表示执行批量

10、更新器使用$set ： 指定一个键值对，若存在就进行修改，不存在则添加

$inc ： 只使用于数字类型，可以为指定键值对的数字类型进行加减操作：

```
db.t_member.update({name:"zhangsan"},{$inc:{age:2}})
```

执行结果是名字叫“zhangsan”的年龄加了 2

$unset ： 删除指定的键

```
db.t_member.update({name:"zhangsan"},{$unset:{age:1}})
```

$push ： 数组键操作：1、如果存在指定的数组，则为其添加值；2、如果不存在指定的数组，则创建数组键，并添加值；3、如果指定的键不为数组类型，则报错；

$addToSet ： 当指定的数组中有这个值时，不插入，反之插入

```
#则不会添加到数组里
db.t_member.update({name:"zhangsan"},{$addToSet:{classes:"English"}})
;
```

$pop：删除指定数组的值，当 value=1 删除最后一个值，当 value=-1 删除第一个值

```
#删除了最后一个值
db.t_member.update({name:"zhangsan"},{$pop:{classes:1}})
```

$pull ： 删除指定数组指定的值

```
#$pullAll 批量删除指定数组
db.persons.update({name:"zhangsan"},{$pull:{classes:"Chinese"}})
```

```
#若数组中有多个 Chinese，则全删除
db.t_member.update({name:"zhangsan"},{$pull:{classes:["Chinese"]}})
```

$ ： 修改指定数组时，若数组有多个对象，但只想修改其中一些，则需要定位器：

```
db.t_member.update({"classes.type":"AA"},{$set:{"classes.$.sex":"male
"}})
```

$addToSet 与 $each 结合完成批量数组更新操作

```
db.t_member.update({name:"zhangsan"},{$set:{classes:{$each:["chinese"
,"art"]}}})
```

runCommand 函数和 findAndModify 函数

```
runCommand({
    findAndModify:"persons",
```

```
    query:{查询器},
    sort:{排序},
    update:{修改器},
    new:true 是否返回修改后的数据
});
```

runCommand 函数可执行 mongdb 中的特殊函数

findAndModify 就是特殊函数之一，用于返回执行返回 update 或 remove 后的文档

例如：

```
db.runCommand({
    findAndModify:"persons",
    query:{name:"zhangsan"},
    update:{$set:{name:"lisi"}},
    new:true
})
```

12、高级查询详解

```
db.t_member.find({},{_id:0,name:1})
```

第一个空括号表示查询全部数据，第二个括号中值为 0 表示不返回，值为 1 表示返回，默认情况下若不指定主键，主键总是会被返回；

```
db.persons.find({条件},{指定键});
```

比较操作符：$lt: < $lte: <= $gt: > $gte: >= $ne: !=

12.1、查询条件

```
#查询年龄大于等于 25 小于等于 27 的人
db.t_member.find({age:{$gte:25,$lte:27}},{_id:0,name:1,age:1})
#查询出所有国籍不是韩国的人的数学成绩
```

```
db.t_member.find({country:{$ne:"韩国"}},{_id:0,name:1,country:1})
```

12.2、包含与不包含（仅针对于数组）

$in 或 $nin

```
#查询国籍是中国或美国的学生信息
db.t_member.find({country:{$in:["China","USA"]}},{_id:0,name:1:countr
y:1})
```

12.3、$or 查询

```
#查询语文成绩大于 85 或者英语大于 90 的学生信息
db.t_member.find({$or:[{c:{$gt:85}},{e:{$gt:90}}]},{_id:0,name:1,c:1,e:1})
#把中国国籍的学生上增加新的键 sex
db.t_member.update({country:"China"},{$set:{sex:"m"}},false,true)
#查询出 sex 为 null 的人
db.t_member.find({sex:{$in:[null]}},{_id:0,name:1,sex:1})
```

12.4、正则表达式

```
#查询出名字中存在”li”的学生的信息
db.t_member.find({name:/li/i},{_id:0,name:1})
```

12.5、$not 的使用

$not 和$nin 的区别是$not 可以用在任何地方儿$nin 是用到集合上的

```
#查询出名字中不存在”li”的学生的信息
db.t_member.find({name:{$not:/li/i}},{_id:0,name:1})
```

12.6、$all 与 index 的使用

```
#查询喜欢看 MONGOD 和 JS 的学生
db.t_member.find({books:{$all:["JS","MONGODB"]}},{_id:0,name:1})
#查询第二本书是 JAVA 的学习信息
db.t_member.find({"books.1":"JAVA"},{_id:0,name:1,books:1})
```

12.7、$size 的使用，不能与比较查询符同时使用

```
#查询出喜欢的书籍数量是 4 本的学生
```

```
db.t_member.find({books:{$size:4}},{_id:0,name:1})
```

12.8、查询出喜欢的书籍数量大于 4 本的学生本的学生

1）增加 size 键

```
db.t_member.update({},{$set:{size:4}},false,true)
```

2）添加书籍,同时更新 size

```
db.t_member.update({name:"jim"},{$push:{books:"ORACL"},$inc:{size:1}
})
```

3）查询大于 3 本的

```
db.t_member.find({size:{$gt:4}},{_id:0,name:1,size:1})
```

12.9、$slice 操作符返回文档中指定数组的内部值

```
#查询出 Jim 书架中第 2~4 本书
db.t_member.find({name:"jim"},{_id:0,name:1,books:{$slice:[1,3]}})
#查询出最后一本书
db.t_member.find({name:"jim"},{_id:0,name:1,books:{$slice:-1}})
```

12.10、文档查询

查询出在 K 上过学且成绩为 A 的学生

1）绝对查询，顺序和键个数要完全符合

```
db.t_member.find({school:{school:"K","score":"A"}},{_id:0,name:1})
```

2）对象方式,但是会出错，多个条件可能会去多个对象查询

```
db.t_member.find({"school.school":"K","school.score":"A"},{_id:0,nam
e:1})
```

正确做法单条条件组查询$elemMatch

```
db.t_member.find({school:{$elemMatch:{school:"K",score:"A"}},{_id:0,n
ame:1})
db.t_member.find({age:{$gt:22},books:"C++",school:"K"},{_id:0,name:1,
```

```
age:1,books:1,school:1})
```

12.11、分页与排序

1）limit 返回指定条数 查询出 persons 文档中前 5 条数据：

```
db.t_member.find({},{_id:0,name:1}).limit(5)
```

2）指定数据跨度 查询出 persons 文档中第 3 条数据后的 5 条数据

```
db.t_member.find({},{_id:0,name:1}).limit(5).skip(3)
```

3）sort 排序 1 为正序，-1 为倒序

```
db.t_member.find({},{_id:0,name:1,age:1}).limit(5).skip(3).sort({age:
1})
```

注意:mongodb 的 key 可以存不同类型的数据排序就也有优先级

最小值->null->数字->字符串->对象/文档->数组->二进制->对象 ID->布尔->日期->时间戳->正则->最大值

12.12、游标

利用游标遍历查询数据

```
var persons = db.persons.find();
    while(persons.hasNext()){
    obj = persons.next();
    print(obj.name)
}
```

游标几个销毁条件

1).客户端发来信息叫他销毁

2).游标迭代完毕

3).默认游标超过 10 分钟没用也会别清除

12.13、查询快照

快照后就会针对不变的集合进行游标运动了,看看使用方法.

```
#用快照则需要用高级查询
db.persons.find({$query:{name:”Jim”},$snapshot:true})
```

高级查询选项

1)$query

2)$orderby

3)$maxsan：integer 最多扫描的文档数

4)$min：doc 查询开始

5)$max：doc 查询结束

6)$hint：doc 使用哪个索引

7)$explain:boolean 统计

8)$snapshot:boolean 一致快照

14.1、查询点(70,180)最近的 3 个点

```
db.map.find({gis:{$near:[70,180]}},{_id:0,gis:1}).limit(3)
```

14.2、查询以点(50,50)和点(190,190)为对角线的正方形中的所有的点

```
db.map.find({gis:{$within:{$box:[[50,50],[190,190]]}}},{_id:0,gis:1})
```

14.3、查询出以圆心为(56,80)半径为 50 规则下的圆心面积中的点

```
db.map.find({gis:{$with:{$center:[[56,80],50]}}},{_id:0,gis:1})
```

15、Count+Distinct+Group

15.1、count 查询结果条数

```
db.persons.find({country:"USA"}).count()
```

15.2、Distinct 去重

请查询出 persons 中一共有多少个国家分别是什么

```
#key 表示去重的键
db.runCommand({distinct:"persons",key:"country"}).values
```

15.3、group 分组

```
db.runCommand({ group:{
    ns:"集合的名字",
    key:"分组键对象",
    initial:"初始化累加器",
    $reduce:"分解器",
    condition:"条件",
    finalize:"组完成器"
}})
```

分组首先会按照 key 进行分组,每组的 每一个文档全要执行$reduce 的方法,他接收 2 个参数一个是组内本条记录,一个是累加器数据.

请查出 persons 中每个国家学生数学成绩最好的学生信息(必须在 90 以上)

```
db.runCommand({
    group:{
    ns:"persons",
    key:{"country":true},
    initial:{m:0},
    $reduce:function(doc,prev){
        if(doc.m>prev.m){
            prev.m = doc.m;
            prev.name = doc.m;
            prev.country = doc.country;
        }
    },
```

```
    condition:{m:{$gt:90}},
    finalize:function(prev){
        prev.m = prev.name+" comes from "+prev.country+" ,Math score is
"+prev.m;
    }
  }
})
```

15.4.函数格式化分组键

如果集合中出现键 Counrty 和 counTry 同时存在

```
$keyf:function(doc){
    if(doc.country){
        return {country:doc.country}
    }
    return {country:doc.counTry}
}
```

16、常用命令举例

16.1、查询服务器版本号和主机操作系统

```
db.runCommand({buildInfo:1})
```

16.2、查询执行集合的详细信息,大小,空间,索引等

```
db.runCommand({collStats:"persons"})
```

16.3、查看操作本集合最后一次错误信息

```
db.runCommand({getLastError:"persons"})
```

17、固定集合

17.1、特性

固定集合默认是没有索引的就算是_id 也是没有索引的

由于不需分配新的空间他的插入速度是非常快的

固定集合的顺是确定的导致查询速度是非常快的

最适合就是日志管理

17.2、创建固定集合

创建一个新的固定集合要求大小是 100 个字节,可以存储文档 10 个

```
db.createCollection("mycoll",{size:100,capped:true,max:10})
```

把一个普通集合转换成固定集合

```
db.runCommand({convertToCapped:"persons",size:1000})
```

17.3、对固定集合反向排序，默认情况是插入的顺序排序

```
db.mycoll.find().sort({$natural:-1})
```