## 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。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。

## 使用场景

MongoDB 的主要目标是在键/值存储方式（提供了高性能和高度伸缩性）和传统的RDBMS 系统（具有丰富的功能）之间架起一座桥梁，它集两者的优势于一身。根据官方网站的描述，Mongo 适用于以下场景。

● 网站数据：Mongo 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。

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

● 大尺寸、低价值的数据：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。

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

● 用于对象及JSON 数据的存储：Mongo 的BSON 数据格式非常适合文档化格式的存储及查询。

你期望一个更高的写负载

● 默认情况下，对比事务安全，MongoDB更关注高的插入速度。如果你需要加载大量低价值的业务数据，那么MongoDB将很适合你的用例。但是必须避免在要求高事务安全的情景下使用MongoDB，比如一个1000万美元的交易。

不可靠环境保证高可用性

* 设置副本集（主-从服务器设置）不仅方便而且很快，此外，使用MongoDB还可以快速、安全及自动化的实现节点（或数据中心）故障转移。

未来会有一个很大的规模

* 数据库扩展是非常有挑战性的，当单表格大小达到5-10GB时，MySQL表格性能会毫无疑问的降低。如果你需要分片并且分割你的数据库，MongoDB将很容易实现这一点。

使用基于位置的数据查询

● MongoDB支持二维空间索引，因此可以快速及精确的从指定位置获取数据。

非结构化数据的爆发增长

* 给RDBMS增加列在有些情况下可能锁定整个数据库，或者增加负载从而导致性能下降，这个问题通常发生在表格大于1GB（更是下文提到BillRun系统中的痛点——单表格动辄几GB）的情况下。鉴于MongoDB的弱数据结构模式，添加1个新字段不会对旧表格有任何影响，整个过程会非常快速；因此，在应用程序发生改变时，你不需要专门的1个DBA去修改数据库模式。

缺少专业的数据库管理员

* 如果你没有专业的DBA，同时你也不需要结构化你的数据及做join查询，MongoDB将会是你的首选。MongoDB非常适合类的持久化，类可以被序列化成JSON并储存在MongoDB。需要注意的是，如果期望获得一个更大的规模，你必须要了解一些最佳实践来避免走入误区。

**不适场景**

● 高度事务性的系统：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。

● 传统的商业智能应用：针对特定问题的BI 数据库会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。

● 需要SQL 的问题。

## 与mysql、redis对比

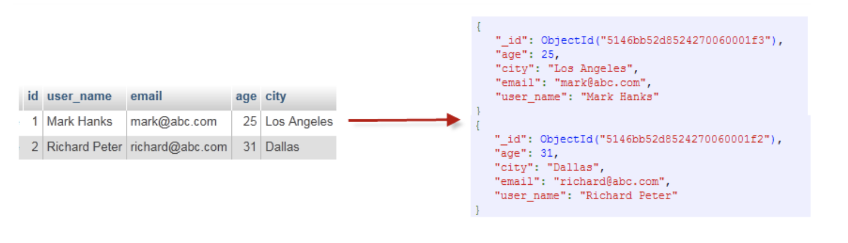
MongoDB 与 MySQL 命令对比 传统的关系数据库一般由数据库（database）、表（table）、记录（record）三个层次概念组成，

MongoDB 是由数据库（database）、集合（collection）、文档对象（document）三个层次组成。



文档就是键值对的一个有序集{'msg':'hello','foo':3}

集合就是一组文档。如果将MongoDB中的一个文档比喻为关系型数据的一行，那么一个集合就是相当于一张表



MongoDB 对于关系型数据库里的表，但是集合中没有列、行和关系概念，这体现了模式自由的特点。索引，索引放在内存中，能够提升随机读写的性能。如果索引不能完全放在内存，一旦出现随机读写比较高的时候，就会频繁地进行磁盘交换，MongoDB 的性能就会急剧下降.

与redis对比：

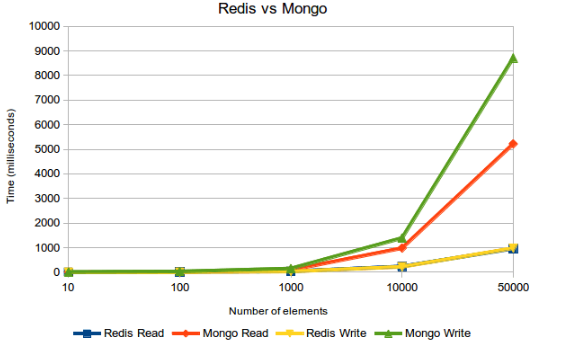
Redis 数据全部存在内存，定期写入磁盘，当内存不够时，可以选择指定的 LRU 算法删除数据。

MongoDB 数据存在内存，由 linux系统 mmap 实现，当内存不够时，只将热点数据放入内存，其他数据存在磁盘。

Redis 支持的数据结构丰富，包括hash、set、list等。

MongoDB 数据结构比较单一，但是支持丰富的数据表达，索引，最类似关系型数据库，支持的查询语言非常丰富。

二者性能都比较高，应该说都不会是瓶颈。



## 基础概念

### 文档

文档是MongoDB的核心概念。文档就是键值对的一个有序集{'msg':'hello','foo':3}。

#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、文档中的值可以是多种不同的数据类型，也可以是一个完整的内嵌文档。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：

#1、键不能含有\0 (空字符)。这个字符用来表示键的结尾。

#2、.和$有特别的意义，只有在特定环境下才能使用。

#3、以下划线"\_"开头的键是保留的(不是严格要求的)。

### 集合

集合就是一组文档。如果将MongoDB中的一个文档比喻为关系型数据的一行，那么一个集合就是相当于一张表。

#1、集合存在于数据库中，通常情况下为了方便管理，不同格式和类型的数据应该插入到不同的集合，但其实集合没有固定的结构，

这意味着我们完全可以把不同格式和类型的数据统统插入一个集合中。

#2、组织子集合的方式就是使用“.”，分隔不同命名空间的子集合。

比如一个具有博客功能的应用可能包含两个集合，分别是blog.posts和blog.authors，这是为了使组织结构更清晰，

这里的blog集合（这个集合甚至不需要存在）跟它的两个子集合没有任何关系。

在MongoDB中，使用子集合来组织数据非常高效，值得推荐

#3、当第一个文档插入时，集合就会被创建。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$。

### 数据库

在MongoDB中，多个文档组成集合，多个集合可以组成数据库。

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：

#1、不能是空字符串（"")。

#2、不得含有' '（空格)、.、$、/、\和\0 (空字符)。

#3、应全部小写。

#4、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。

#1、admin： 从身份认证的角度讲，这是“root”数据库，如果将一个用户添加到admin数据库，这个用户将自动获得所有数据库的权限。再者，一些特定的服务器端命令也只能从admin数据库运行，如列出所有数据库或关闭服务器

#2、local: 这个数据库永远都不可以复制，且一台服务器上的所有本地集合都可以存储在这个数据库中

#3、config: MongoDB用于分片设置时，分片信息会存储在config数据库

**把数据库名添加到集合名前，得到集合的完全限定名，即命名空间。**

例如：

如果要使用cms数据库中的blog.posts集合，这个集合的命名空间就是

cms.blog.posts。命名空间的长度不得超过121个字节，且在实际使用中应该小于100个字节.

## 安装

### Window安装

1. 下载地址 <https://www.mongodb.com/download-center/community> 如果不想填写信息,可以在<https://www.mongodb.org/dl/win32/x86_64> 下载需要的zip版本,当前最新版本4.2.注意直接下载很慢,将下载链接地址域名替换为[https://fastdl.mongodb.org](https://fastdl.mongodb.org/win32/mongodb-win32-x86_64-2012plus-v4.2-latest.zip" \t "_blank),后面的不变.
2. 图形化管理工具下载地址 : <https://www.mongodb.com/download-center/compass>
3. 将下载好的zip文件解压到D:\MongoDB\mongodb下,然后创建数据目录,数据目录默认是在安装盘的根目录下,不然会报错.创建D:\data\db和D:\data\log;
4. Bin目录下直接运行mongod就可以启动mongodb服务了.

**配置MongoDB服务**

1. **管理员权限**打开cmd进入到bin目录下,否则安装会报错.
2. 创建配置文件位于D:\MongoDB\mongodb下,命名mongod.cfg,设置 systemLog.path 参数.

**systemLog:**

**destination: file**

**path: D:\data\log\mongod.log**

**storage:**

**dbPath: D:\data\db**

1. 通过--install命令安装服务

**mongod.exe --config "** **D:\MongoDB\mongodb\mongod.cfg" --install**

**这样就能在services.msc里看到mongodb服务了,可以直接在这启动,也可以命令行启动,为了方便也可以设为开机启动.**

**net start MongoDB 开启服务**

**net stop MongoDB 停止服务**

**mongod.exe –remove 移除服务**

## 基本命令

db 显示当前正在使用的数据库

show dbs 列出所有可用数据库

show tables 列出所有集合

user <database> 进入数据库,没有则新建

db.getSiblingDB(‘users’) 切换到不同数据库

切换到一个不存在的数据库。当你第一次向数据库存储数据，如通过创建一个集合，MongoDB将自动创建数据库。例如，当执行 insert() 操作期间，数据库``myNewDatabase`` 和 collection ``myCollection``都将被创建。

use myNewDatabase

db.myCollection.insert( { x: 1 } );

[``](http://www.mongoing.com/docs/mongo.html#id1)db``指当前数据库。myCollection``是集合的名称

db.myCollection.find().pretty()

格式化打印结果

### 创建

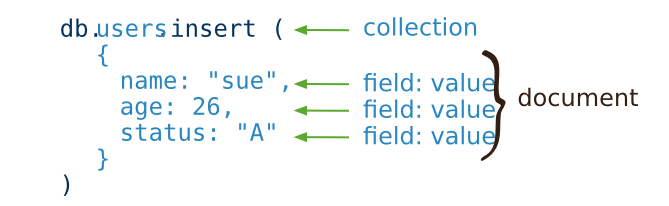
use database 创建数据库,如果database不存在会自动创建

[db.collection.insert()](http://www.mongoing.com/docs/reference/method/db.collection.insert.html#db.collection.insert)

[db.collection.insertOne()](http://www.mongoing.com/docs/reference/method/db.collection.insertOne.html#db.collection.insertOne)

[db.collection.insertMany()](http://www.mongoing.com/docs/reference/method/db.collection.insertMany.html#db.collection.insertMany)

在MongoDB中，插入操作作用于单个 集合collection 。MongoDB中所有的写操作在单个 [集合 document](http://www.mongoing.com/docs/core/document.html) 的层级上是 [原子性](http://www.mongoing.com/docs/core/write-operations-atomicity.html)。



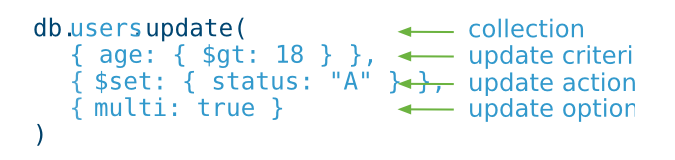
### 更新

更新操作修改 [集合 collection](http://www.mongoing.com/docs/core/databases-and-collections.html#collections) 中已经存在的 [文档 documents](http://www.mongoing.com/docs/core/document.html#bson-document-format)。MongoDB提供了以下方法去更新集合中的文档:

* [db.collection.update()](http://www.mongoing.com/docs/reference/method/db.collection.update.html#db.collection.update)
* [db.collection.updateOne()](http://www.mongoing.com/docs/reference/method/db.collection.updateOne.html#db.collection.updateOne) *New in version 3.2*
* method:db.collection.updateMany() *New in version 3.2*
* [db.collection.replaceOne()](http://www.mongoing.com/docs/reference/method/db.collection.replaceOne.html#db.collection.replaceOne) *New in version 3.2*

在MongoDB中,更新操作作用于单个集合。MongoDB中所有的写操作在单个 [文档 document](http://www.mongoing.com/docs/core/document.html) 层级上是 [原子性的](http://www.mongoing.com/docs/core/write-operations-atomicity.html).

你可以指定条件或过滤器来找到要更新的文档。这些 [过滤器](http://www.mongoing.com/docs/core/document.html#document-query-filter) 的使用与读操作一样的语法



db.collection.update(

<query>,

<update>,

{

upsert: <boolean>, //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: <boolean>, // 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: <document> //抛出异常的级别

}

)

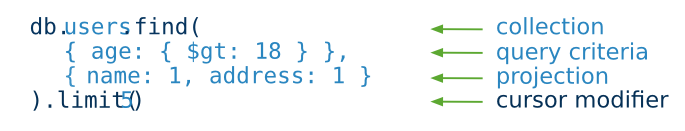
示例请查看 [更新文档](http://www.mongoing.com/docs/tutorial/update-documents.html).

### 查询

读操作获取 [集合 collection](http://www.mongoing.com/docs/core/databases-and-collections.html#collections) 中的 [文档 documents](http://www.mongoing.com/docs/core/document.html#bson-document-format) ;例如查询一个集合中的文档。MongoDB提供了如下方法从集合中读取文档:

[db.collection.find()](http://www.mongoing.com/docs/reference/method/db.collection.find.html#db.collection.find)

你可以指定 [条件或者过滤器](http://www.mongoing.com/docs/tutorial/query-documents.html#read-operations-query-argument) 找到指定的文档.



示例请查看 [查询文档](http://www.mongoing.com/docs/tutorial/query-documents.html).

db.collection.find(

query, //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）

)

| **操作** | **格式** | **范例** | **RDBMS中的类似语句** |
| --- | --- | --- | --- |
| 等于 | {<key>:<value>} | db.col.find({"by":"菜鸟教程"}).pretty() | where by = '菜鸟教程' |
| 小于 | {<key>:{$lt:<value>}} | db.col.find({"likes":{$lt:50}}).pretty() | where likes < 50 |
| 小于或等于 | {<key>:{$lte:<value>}} | db.col.find({"likes":{$lte:50}}).pretty() | where likes <= 50 |
| 大于 | {<key>:{$gt:<value>}} | db.col.find({"likes":{$gt:50}}).pretty() | where likes > 50 |
| 大于或等于 | {<key>:{$gte:<value>}} | db.col.find({"likes":{$gte:50}}).pretty() | where likes >= 50 |
| 不等于 | {<key>:{$ne:<value>}} | db.col.find({"likes":{$ne:50}}).pretty() | where likes != 50 |

**AND条件**

传入多个键(key)，每个键(key)以逗号隔开，即常规 SQL 的 AND 条件

db.col.find({key1:value1, key2:value2}).pretty()

**OR条件**

**使用关键字$or**

db.col.find(

{

$or: [

     {key1: value1}, {key2:value2}

]

}

).pretty()

**Limit**

**只取number条**

db.COLLECTION\_NAME.find().limit(NUMBER)

**skip()方法**

**跳过number条**

db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

**sort方法**

**排序,1:升序,-1:降序**

db.COLLECTION\_NAME.find().sort({KEY:1})

**聚合查询**

db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}])

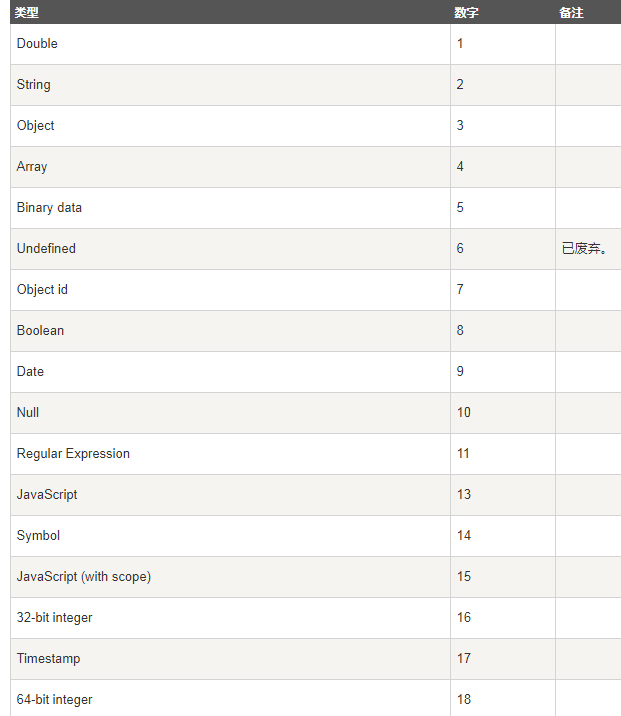
|  |  |  |
| --- | --- | --- |
| **表达式** | **描述** | **实例** |
| $sum | 计算总和。 | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : "$likes"}}}]) |
| $avg | 计算平均值 | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$avg : "$likes"}}}]) |
| $min | 获取集合中所有文档对应值得最小值。 | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$min : "$likes"}}}]) |
| $max | 获取集合中所有文档对应值得最大值。 | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$max : "$likes"}}}]) |
| $push | 在结果文档中插入值到一个数组中。 | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$push: "$url"}}}]) |
| $addToSet | 在结果文档中插入值到一个数组中，但不创建副本。 | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$addToSet : "$url"}}}]) |
| $first | 根据资源文档的排序获取第一个文档数据。 | db.mycol.aggregate([{$group : {\_id : "$by\_user", first\_url : {$first : "$url"}}}]) |
| $last | 根据资源文档的排序获取最后一个文档数据 | db.mycol.aggregate([{$group : {\_id : "$by\_user", last\_url : {$last : "$url"}}}]) |

**$Type操作符**

db.col.find({"title" : {$type : 'string'}})

或

db.col.find({"title" : {$type : 2}})



**管道查询**

**将当前命令的参数作为下一命令的参数,类似于java中的流概念**

* $project：修改输入文档的结构。可以用来重命名、增加或删除域，也可以用于创建计算结果以及嵌套文档。
* $match：用于过滤数据，只输出符合条件的文档。$match使用MongoDB的标准查询操作。
* $limit：用来限制MongoDB聚合管道返回的文档数。
* $skip：在聚合管道中跳过指定数量的文档，并返回余下的文档。
* $unwind：将文档中的某一个数组类型字段拆分成多条，每条包含数组中的一个值。
* $group：将集合中的文档分组，可用于统计结果。
* $sort：将输入文档排序后输出。
* $geoNear：输出接近某一地理位置的有序文档。
* db.article.aggregate(
* { $project : {
* title : 1 ,
* author : 1 ,
* }}
* );
* //只包含 \_id,tilte和author三个字段,去掉id用\_id:0
* $match 匹配查询
* db.articles.aggregate( [
* { $match : { score : { $gt : 70, $lte : 90 } } },
* { $group: { \_id: null, count: { $sum: 1 } } }
* ] );

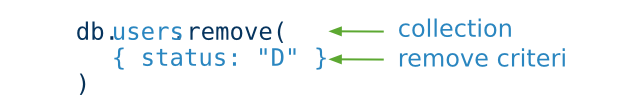
### 删除

删除是从一个集合中删除文档的操作。MongoDB提供下列方法从集合删除文档。

* [db.collection.remove()](http://www.mongoing.com/docs/reference/method/db.collection.insert.html#db.collection.insert) 删除文档(过时)
* [db.collection.deleteOne()](http://www.mongoing.com/docs/reference/method/db.collection.insertOne.html#db.collection.insertOne) *New in version 3.2*
* [db.collection.deleteMany()](http://www.mongoing.com/docs/reference/method/db.collection.deleteMany.html#db.collection.deleteMany) *New in version 3.2*
* db.dropDatabase() 删除数据库
* db.collection.drop() 删除集合

在MongoDB中。删除作作用于单个集合。MongoDB中所有的写操作在单个 [文档 document](http://www.mongoing.com/docs/core/document.html) 层级上是 [原子性的](http://www.mongoing.com/docs/core/write-operations-atomicity.html)。

你可以指定条件或过滤器来找到要删除的文档。这些 [过滤器](http://www.mongoing.com/docs/core/document.html#document-query-filter) 的使用与读操作一样的语法。



db.collection.remove(

<query>,

{

justOne: <boolean>, // true 或 1，则只删除一个文档，如果不设置该参数，或使用默认值 false，则删除所有匹配条件的文档。

writeConcern: <document> //抛出异常的级别

}

)

示例请查看 [删除文档](http://www.mongoing.com/docs/tutorial/remove-documents.html)。

### 批量插入

ongoDB提供批量写操作的能力。细节请参考：[Bulk Write Operations](http://www.mongoing.com/docs/core/bulk-write-operations.html)。

## 索引

### 基础索引

db.collection.createIndex(keys, options) //1升序 -1 降序

db.col.createIndex({"title":1,"description":-1}) //复合索引

db.values.createIndex({open: 1, close: 1}, {background: true})

|  |  |  |
| --- | --- | --- |
| **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。 |
| 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. |

db.col.getIndexes() //查看集合索引

db.col.totalIndexSize() //集合索引大小

db.col.dropIndexes() //删除集合所有索引

db.col.dropIndex("索引名称") //删除指定索引

### 文本索引

**支持在字符串内容上的文本检索查询,关键字$text**

db.stores.createIndex( { name: "text", description: "text" } )

db.stores.find( { $text: { $search: "java coffee shop" } } )// 所有包含 “coffee”, “shop”, 以及 “java” 列表中任何词语的,相当于or

db.stores.find( { $text: { $search: "java \"coffee shop\"" } } )// 包含”java” 或者 “coffee shop” 的文档 精确查找

db.stores.find( { $text: { $search: "java shop -coffee" } } )//排除包含coffee的文档

db.stores.find(

{ $text: { $search: "java coffee shop" } },

{ score: { $meta: "textScore" } }

).sort( { score: { $meta: "textScore" } } )

//排序 支持在管道中使用

## 副本集

MongoDB复制是将数据同步在多个服务器的过程。

复制提供了数据的冗余备份，并在多个服务器上存储数据副本，提高了数据的可用性， 并可以保证数据的安全性。

复制还允许您从硬件故障和服务中断中恢复数据。

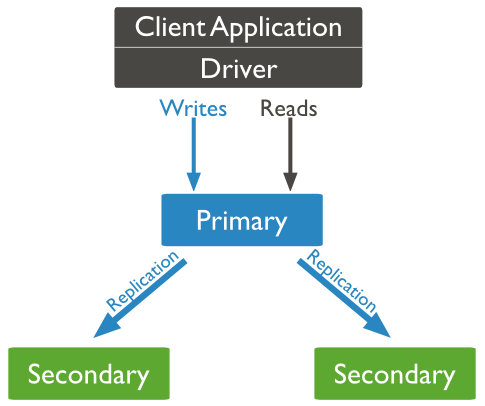
1. **原理:**

mongodb的复制至少需要两个节点。其中一个是主节点，负责处理客户端请求，其余的都是从节点，负责复制主节点上的数据。

mongodb各个节点常见的搭配方式为：一主一从、一主多从。

主节点记录在其上的所有操作oplog，从节点定期轮询主节点获取这些操作，然后对自己的数据副本执行这些操作，从而保证从节点的数据与主节点一致。

MongoDB复制结构图如下所示：



1. **特征**

* N 个节点的集群
* 任何节点可作为主节点
* 所有写入操作都在主节点上
* 自动故障转移
* 自动恢复

1. **设置**

指定 --replSet 选项来启动mongoDB。--replSet 基本语法格式如下：

mongod --port "PORT" --dbpath "YOUR\_DB\_DATA\_PATH" --replSet "REPLICA\_SET\_INSTANCE\_NAME"

mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0

以上实例会启动一个名为rs0的MongoDB实例，其端口号为27017。

启动后打开命令提示框并连接上mongoDB服务。

在Mongo客户端使用命令rs.initiate()来启动一个新的副本集。

我们可以使用rs.conf()来查看副本集的配置

查看副本集状态使用 rs.status() 命令

**添加副本集的成员，我们需要使用多台服务器来启动mongo服务。进入Mongo客户端，并使用rs.add()方法来添加副本集的成员。**

rs.add(HOST\_NAME:PORT)

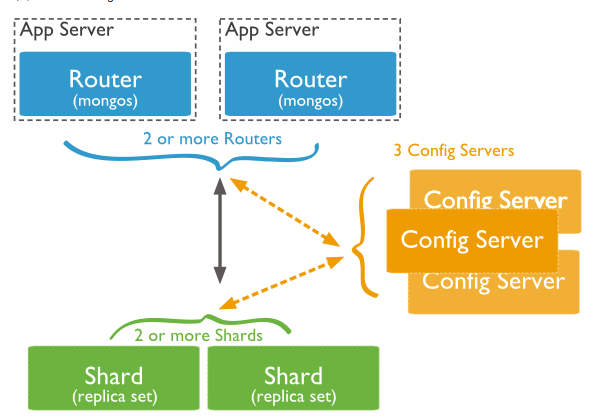
rs.add("mongod1.net:27017")

只能通过主节点将Mongo服务添加到副本集中， 判断当前运行的Mongo服务是否为主节点可以使用命令db.isMaster() 。

副本集在主机宕机后，副本会接管主节点成为主节点，不会出现宕机的情况。

## 分片

当MongoDB存储海量的数据时，一台机器可能不足以存储数据，也可能不足以提供可接受的读写吞吐量。这时，我们就可以通过在多台机器上分割数据，使得数据库系统能存储和处理更多的数据。



* **Shard:**

用于存储实际的数据块，实际生产环境中一个shard server角色可由几台机器组个一个replica set承担，防止主机单点故障

* **Config Server:**

mongod实例，存储了整个 ClusterMetadata，其中包括 chunk信息。

* **Query Routers:**

前端路由，客户端由此接入，且让整个集群看上去像单一数据库，前端应用可以透明使用。