# 显示所有数据库列表

show dbs

# 显示当前数据库对象或集合

db

# 连接

mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[database][?options]]

mongodb:// 这是固定的格式，必须要指定。

username:password@ 可选项，如果设置，在连接数据库服务器之后，驱动都会尝试登陆这个数据库

host1 必须的指定至少一个host, host1 是这个URI唯一要填写的。它指定了要连接服务器的地址。如果要连接复制集，请指定多个主机地址。

portX 可选的指定端口，如果不填，默认为27017

/database 如果指定username:password@，连接并验证登陆指定数据库。若不指定，默认打开 test 数据库。

?options 是连接选项。如果不使用/database，则前面需要加上/。所有连接选项都是键值对name=value，键值对之间通过&或;（分号）隔开

标准的连接格式包含了多个选项(options)，如下所示：

|  |  |
| --- | --- |
| **选项** | **描述** |
| replicaSet=name | 验证replica set的名称。 Impliesconnect=replicaSet. |
| slaveOk=true|false | * true:在connect=direct模式下，驱动会连接第一台机器，即使这台服务器不是主。在connect=replicaSet模式下，驱动会发送所有的写请求到主并且把读取操作分布在其他从服务器。 * false: 在 connect=direct模式下，驱动会自动找寻主服务器. 在connect=replicaSet 模式下，驱动仅仅连接主服务器，并且所有的读写命令都连接到主服务器。 |
| safe=true|false | * + true: 在执行更新操作之后，驱动都会发送getLastError命令来确保更新成功。(还要参考 wtimeoutMS).   false: 在每次更新之后，驱动不会发送getLastError来确保更新成功。 |
| w=n | 驱动添加 { w : n } 到getLastError命令. 应用于safe=true。 |
| wtimeoutMS=ms | 驱动添加 { wtimeout : ms } 到 getlasterror 命令. 应用于 safe=true. |
| fsync=true|false | * true: 驱动添加 { fsync : true } 到 getlasterror 命令.应用于 safe=true. * false: 驱动不会添加到getLastError命令中。 |
| journal=true|false | 如果设置为 true, 同步到 journal (在提交到数据库前写入到实体中). 应用于 safe=true |
| connectTimeoutMS=ms | 可以打开连接的时间。 |
| socketTimeoutMS=ms | 发送和接受sockets的时间。 |

# 创建数据库

use local

# 删除数据库

db.dropDatabase()

# 查看已存在集合

show collections

# 删除集合

db.collection.drop()

collention:集合名称

# 创建集合

db.createCollection(name, options)

参数说明：

name: 要创建的集合名称

options: 可选参数, 指定有关内存大小及索引的选项

options 可以是如下参数：

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| capped | 布尔 | （可选）如果为 true，则创建固定集合。固定集合是指有着固定大小的集合，当达到最大值时，它会自动覆盖最早的文档。 **当该值为 true 时，必须指定 size 参数。** |
| autoIndexId | 布尔 | （可选）如为 true，自动在 \_id 字段创建索引。默认为 false。 |
| size | 数值 | （可选）为固定集合指定一个最大值（以字节计）。 **如果 capped 为 true，也需要指定该字段。** |
| max | 数值 | （可选）指定固定集合中包含文档的最大数量。 |

在插入文档时，MongoDB 首先检查固定集合的 size 字段，然后检查 max 字段。

# 插入文档

db.COLLECTION\_NAME.insert(document)

db.col.save(document):如果不指定 \_id 字段 save() 方法类似于 insert() 方法。如果指定 \_id 字段，则会更新该 \_id 的数据。

db.collection.insertOne():向指定集合中插入一条文档数据

db.collection.insertMany():向指定集合中插入多条文档数据

# 更新文档

db.collection.updateOne() 向指定集合更新单个文档

db.collection.updateMany() 向指定集合更新多个文档

db.collection.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>

})

参数说明：

query : update的查询条件，类似sql update查询内where后面的。

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. { $unset: { <field1>: "", ... } } 移除集合中的键值对, 如果指定的字段不存在则操作不做任何处理。

upsert : 可选，这个参数的意思是，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。

writeConcern :可选，抛出异常的级别。

db.collection.save(

<document>,

{

writeConcern: <document>

}

)

参数说明：

document : 文档数据。

writeConcern :可选，抛出异常的级别。

WriteConcern.NONE:没有异常抛出

WriteConcern.NORMAL:仅抛出网络错误异常，没有服务器错误异常

WriteConcern.SAFE:抛出网络错误异常、服务器错误异常；并等待服务器完成写操作。

WriteConcern.MAJORITY: 抛出网络错误异常、服务器错误异常；并等待一个主服务器完成写操作。

WriteConcern.FSYNC\_SAFE: 抛出网络错误异常、服务器错误异常；写操作等待服务器将数据刷新到磁盘。

WriteConcern.JOURNAL\_SAFE:抛出网络错误异常、服务器错误异常；写操作等待服务器提交到磁盘的日志文件。

WriteConcern.REPLICAS\_SAFE:抛出网络错误异常、服务器错误异常；等待至少2台服务器完成写操作。

# 删除文档

db.collection.remove(

<query>,

{

justOne: <boolean>,

writeConcern: <document>

}

)

参数说明：

query :（可选）删除的文档的条件。

justOne : （可选）如果设为 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 :（可选）抛出异常的级别。

remove() 方法已经过时了，现在官方推荐使用 deleteOne() 和 deleteMany() 方法

删除 status 等于 A 的全部文档：

db.inventory.deleteMany({ status : "A" })

删除 status 等于 D 的一个文档：

db.inventory.deleteOne( { status: "D" } )

需要继续执行 db.repairDatabase() 来回收磁盘空间或db.runCommand({ repairDatabase: 1 })

# 查询文档

db.collection.find(query, projection)

query ：可选，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）。

除了 find() 方法之外，还有一个 findOne() 方法，它只返回一个文档。

db.col.find().pretty() 以格式化的方式来显示所有文档。

MongoDB 与 RDBMS Where 语句比较

| **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

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

OR

db.col.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

).pretty()

db.collection.find(query, {title: 1, by: 1}) // inclusion模式 指定返回的键，不返回其他键

db.collection.find(query, {title: 0, by: 0}) // exclusion模式 指定不返回的键,返回其他键

两种模式不可混用（因为这样的话无法推断其他键是否应返回）

\_id 键默认返回，需要主动指定 \_id:0 才会隐藏

只能全1或全0，除了在inclusion模式时可以指定\_id为0

若不想指定查询条件参数 query 可以 用 {} 代替，但是需要指定 projection 参数：querydb.collection.find({}, {title: 1})

如果是 qty 大于 50 小于 80: db.posts.find( { qty: { $gt: 50 ,$lt: 80}} )

# 条件操作符