# mongodb的基本使用

## 操作数据库

* 1. 新建数据库

1. use dbname(数据库名称)
2. 如果数据库存在则进入该数据库，如果不存在则创建一个新的数据库。
   1. 查看数据库
3. show dbs 查看所有的数据库
4. db.getName() 查看当前数据库名
   1. 删除数据库
5. db.dropDatabase() 删除当前数据库

## 操作集合

* 1. 新建集合

1. 语法格式：db.createCollection(name, options)

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,具体如下：



1. 查看当前数据库所有集合

show collections

db.getCollectionNames()

1. MongoDB 中，你不需要创建集合。当你插入一些文档时，MongoDB 会自动创建集合。



* 1. 删除集合

1. db.collection.drop() 删除集合

## 操作文档

* 1. 插入文档

1. 使用insert()，save() 效果一样

db.COLLECTION\_NAME.insert(document)

1. 直接插入

db.yu.insert({title:’cianiao’,name:’jack’});

1. 定义变量后插入

doc={name:’roos’,corlor:’green’}

db.yu.save(doc)

* 1. 更新文档
     1. update语法格式

db.collection.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>

}

)

* + 1. 参数说明

1. query : update的查询条件，类似sql update查询内where后面的。
2. 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
3. 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。
4. 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。
5. writeConcern :可选，抛出异常的级别。
   * 1. update实例
6. 只更新查到的第一条数据：

db.yu.update({title:'cainiao'},{$set:{title:"yeyeye"}})

1. 更新所有数据：

db.yu.update({title:'cainiao'},{$set:{title:"yeyeye"}}，{multi:true})

* + 1. save更新语法格式

db.collection.save(

<document>,

{

writeConcern: <document>

}

)

1. document为更新的文档数据
2. writeConcern:可选，抛出异常级别
   * 1. 实例

db.yu.save({"\_id": ObjectId("5bed30e3099238d1ce2373cf"),"title":"nonono"})

1. 原理是根据\_id来更新文档数据
   * 1. 更多实例
2. 只更新第一条记录：

db.col.update( { "count" : { $gt : 1 } } , { $set : { "test2" : "OK"} } );

1. 全部更新：

db.col.update( { "count" : { $gt : 3 } } , { $set : { "test2" : "OK"} },false,true );

1. 只添加第一条：

db.col.update( { "count" : { $gt : 4 } } , { $set : { "test5" : "OK"} },true,false );

1. 全部添加进去:

db.col.update( { "count" : { $gt : 5 } } , { $set : { "test5" : "OK"} },true,true );

1. 全部更新：

db.col.update( { "count" : { $gt : 15 } } , { $inc : { "count" : 1} },false,true );

1. 只更新第一条记录：

db.col.update( { "count" : { $gt : 10 } } , { $inc : { "count" : 1} },false,false );

* 1. 删除文档
     1. remove()删除

db.collection.remove(

<query>,

{

justOne: <boolean>,

writeConcern: <document>

}

)

1. query :（可选）删除的文档的条件。
2. justOne : （可选）如果设为 true 或 1，则只删除一个文档。
3. writeConcern :（可选）抛出异常的级别。
4. 实例：db.yu.remove({"title" : "nonono"})
   * 1. delete删除（**推荐**）
5. 如删除集合下全部文档：

db.inventory.deleteMany({})

1. 删除 status 等于 A 的全部文档：

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

1. 删除 status 等于 D 的一个文档：

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

* 1. 查询文档
     1. 语法格式

db.collection.find(query, projection)

1. query ：可选，使用查询操作符指定查询条件
2. 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）。
3. 格式化可以用 db.collectionname.find().pretty()
4. db.collection.findOne()只返回一个文档
   * 1. MongoDB与关系型数据库where语句比较



* + 1. AND条件

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。

语法格式：>db.col.find({key1:value1, key2:value2}).pretty()

1. 实例代码



* + 1. OR条件

1. 语法格式：

>db.col.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

).pretty()

1. 实例代码：



* 1. AND 和 OR 联合使用



* 1. projection参数的用法（指定返回的键）

db.collection.find(query, projection)

1. 若不指定projection，则默认返回所有键。
2. 指定projection有两种模式

// inclusion模式 指定返回的键，不返回其他键

db.collection.find(query, {title: 1, by: 1})

// exclusion模式 指定不返回的键,返回其他键

db.collection.find(query, {title: 0, by: 0})

//\_id 键默认返回，需要主动指定 \_id:0 才会隐藏

1. 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）

db.collection.find(query, {title: 1, by: 0}) // 错误

1. 只能全1或全0，除了在inclusion模式时可以指定\_id为0

db.collection.find(query, {\_id:0, title: 1, by: 1}) // 正确

1. 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})

## 条件操作符

* 1. 概览



1. likes大于100的数据

db.col.find({likes : {$gt : 100}})

1. "likes" 大于等于 100 的数据

db.col.find({likes : {$gte : 100}})

1. "likes" 小于 150 的数据

db.col.find({likes : {$lt : 150}})

1. "likes" 小于等于 150 的数据

db.col.find({likes : {$lte : 150}})

1. "likes" 大于100，小于 200 的数据

db.col.find({likes : {$lt :200, $gt : 100}})

* 1. 模糊查询

1. 查询 title 包含"教"字的文档：

db.col.find({title:/教/})

1. 查询 title 字段以"教"字开头的文档：

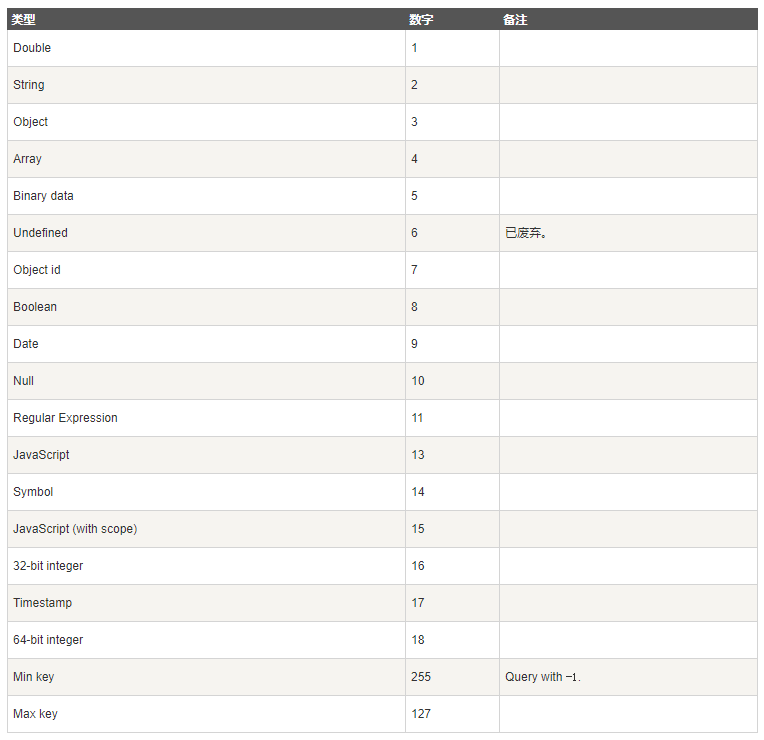
db.col.find({title:/^教/})

1. 查询 titl e字段以"教"字结尾的文档：

db.col.find({title:/教$/})

## $type操作符

* 1. 根据数据类型查询



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

## limit（）和skip（）

* 1. 概念

1. limit是限制查询出来的记录总数。
2. skip是跳过符合条件的记录数，比如skip（1）跳过一条记录数。
   1. 查询实例

db.col.find({title:{$type: 2}},{title:1,\_id:0}).skip(1).limit(1)

相当于

mysql中的limit(1,1)

## 排序

* 1. sort()方法

1. sort() 方法可以通过参数指定排序的字段，并使用 1 和 -1 来指定排序的方式，其中 1 为升序排列，而 -1 是用于降序排列。
2. 语法格式：

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

1. 实例

db.col.find({title:{$type: 2}},{title:1,\_id:0}).sort({ title:-1 })

## 索引

* 1. 基本介绍

1. 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。
   1. 创建索引

语法格式：db.collection.createIndex(keys, options)

1. Key 值为你要创建的索引字段，1 为指定按升序创建索引，如果你想按降序来创建索引指定为 -1 即可。
2. 也可以设置使用多个字段创建索引（关系型数据库中称作复合索引）。
3. 实例代码

db.col.createIndex({"title":1}) //单个字段索引

db.col.createIndex({"title":1,"url":1}) //复合索引，遵循最左前缀原则

* 1. option参数使用



1. 实例代码：

db.col.createIndex({"title":1,"url":1},{background:true})

## 聚合

* 1. 简介

MongoDB中聚合(aggregate)主要用于处理数据(诸如统计平均值,求和等)，并返回计算后的数据结果。有点类似sql语句中的 count(\*)。

* 1. aggregate()方法实现聚合
     1. 语法格式

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

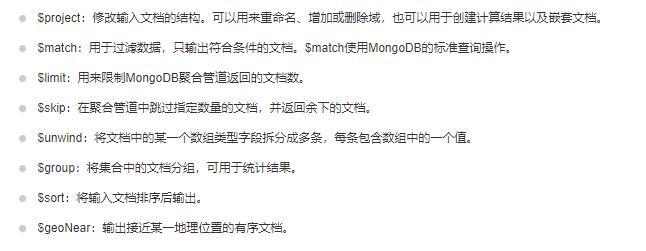


1. 实例代码：

db.col.aggregate([{$group: { \_id: "$tags",total:{$sum: 1}}}])

* 1. 管道
     1. 简介

1. 管道在Unix和Linux中一般用于将当前命令的输出结果作为下一个命令的参数。
2. MongoDB的聚合管道将MongoDB文档在一个管道处理完毕后将结果传递给下一个管道处理。管道操作是可以重复的。
3. 表达式：处理输入文档并输出。表达式是无状态的，只能用于计算当前聚合管道的文档，不能处理其它的文档。
   * 1. 常用操作



* + 1. 实例

1. $project – 用来指定输出的字段，默认有\_id

查询语句：

db.col.aggregate([{

$project: {"title":1,"by":1,"url":1}

}])

输出结果：



1. $match – 用于过滤数据

查询语句：

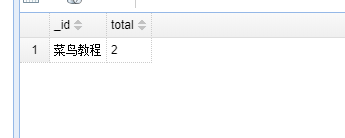
db.col.aggregate([

{$match:{likes:{$gt:99,$lt:199}}},

{$group: { \_id: "$by",total:{$sum:1}}}

])

输出结果：



1. $skip –用于跳过记录

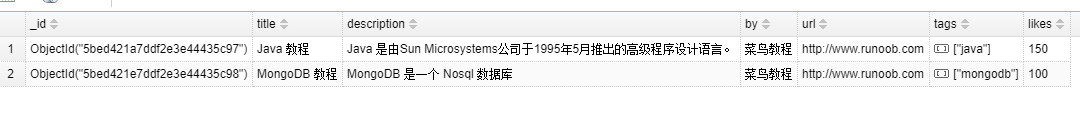
查询语句：

db.col.aggregate([{

$skip: 1

}])

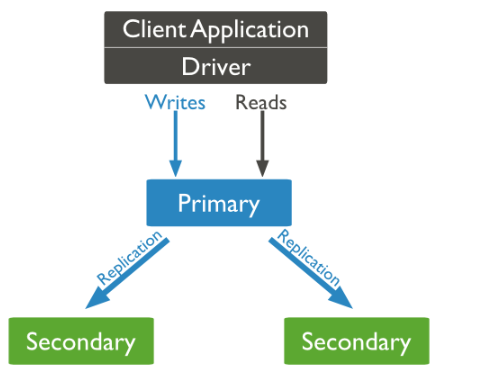
输出结果：



## 复制副本集

* 1. 简介

1. MongoDB复制是将数据同步在多个服务器的过程。
2. 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。
3. 复制还允许您从硬件故障和服务中断中恢复数据。
   1. 基本原理
4. 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。
5. mongodb各个节点常见的搭配方式为：一主一从、一主多从。
6. 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。



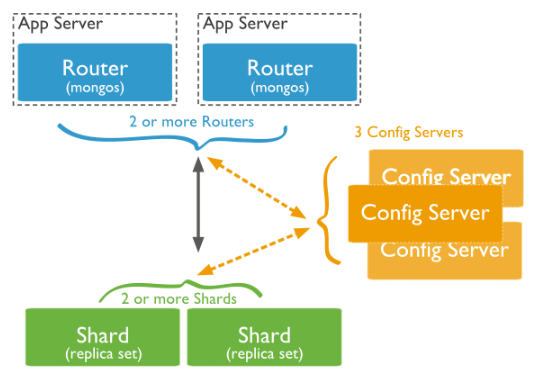
* 1. 副本集特征

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

## MongoDB 分片

* 1. 简介

1. 在Mongodb里面存在另一种集群，就是分片技术,可以满足MongoDB数据量大量增长的需求。
2. 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。
   1. 基本原理



1. 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

1. Config Server:

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

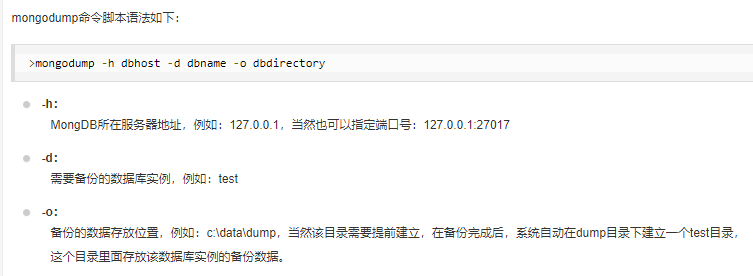
1. Query Routers:

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

## 备份(mongodump)与恢复(mongorestore)

* 1. 备份

1. 在Mongodb中我们使用mongodump命令来备份MongoDB数据。该命令可以导出所有数据到指定目录中。
2. mongodump命令可以通过参数指定导出的数据量级转存的服务器。
3. 语法格式：



* 1. 恢复

1. 使用命令来恢复备份的数据。
2. 语法格式：



## MongoDB 监控

MongoDB中提供了mongostat 和 mongotop 两个命令来监控MongoDB的运行情况。

* 1. mongostat 命令

1. mongostat是mongodb自带的状态检测工具，在命令行下使用。它会间隔固定时间获取mongodb的当前运行状态，并输出。如果你发现数据库突然变慢或者有其他问题的话，你第一手的操作就考虑采用mongostat来查看mongo的状态。
2. 执行命令

启动你的Mongod服务，进入到你安装的MongoDB目录下的bin目录， 然后输入mongostat命令

C:\Users\MONSTE~1\AppData\Local\Temp\1542680874(1).png

* 1. mongotop命令

1. mongotop也是mongodb下的一个内置工具，mongotop提供了一个方法，用来跟踪一个MongoDB的实例，查看哪些大量的时间花费在读取和写入数据。 mongotop提供每个集合的水平的统计数据。默认情况下，mongotop返回值的每一秒。
2. 执行命令

启动你的Mongod服务，进入到你安装的MongoDB目录下的bin目录， 然后输入mongotop命令

