MongoDB学习笔记

# MongoDB简介

MongoDB是有c++编写，是一个基于分布式文件存储的开源数据库（旨在为WEB提供可扩展的高性能数据存储解决方案，将数据以键值对的形式存储为一个文档）支持的数据结构为BSON格式，类似JSON的一种格式，因此可以存储非常复杂的数据。

MongDB 是一个介于[关系数据库](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93)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](https://baike.baidu.com/item/json)的[bson](https://baike.baidu.com/item/bson)格式，因此可以存储比较复杂的数据类型。MongoDB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](https://baike.baidu.com/item/%E7%B4%A2%E5%BC%95)

# MongoDB的主要特点

2.1支持各种编程语言,java, C++ ,PHP ,c# ,Python

2.2面向文档存储，文档格式格式是类似JSON的BSON格式

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。它基于[JavaScript Programming Language](http://www.crockford.com/javascript)的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）。 这些特性使JSON成为理想的数据交换语言。

Json结构示例

|  |
| --- |
| {  "test1": "wx9fdb8ble7ce3c68f",  "test2": "123456789",  "testData1": {  "testdatason1": "97895455"  "testdatason2":3,  "testData2": [  {  "testshuzu1":"12"  "testshuzu1":"11"  },  {  "testshuzu2":"13"  "testshuzu2":"14"  },  ,  {  "testshuzu3":"15"  "testshuzu3":"16"  }  ]  }  } |

2.3提供丰富的查询功能

支持按照主键查询，也可以根据条件查询，支持类似于and or in等操作。

2.4模式自由

即格式自由：MongoDB存储数据并不是按照关系型数据库那种“表格”方式进行存储，取而使用键值对方式进行存储。“每一行”数据由多个键值对的数据组合而成变成记录，并使多条记录组成集合中的数据。键值对的值可以有很多的类型，例如：数组，集合，文档，语言所提供的基本类型等。

2.5可通过网络访问（通过ip地址远程访问）

2.6MongoDB记录中设置任何属性的索引

如：(FirstName="Sameer",Address="8 Gandhi Road")来实现更快的排序

# MongoDB数据架构

Document

Collection

Database

Document

Document

Collection

|  |  |  |
| --- | --- | --- |
| **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 |

# 为什么使用MongoDB

MongoDB的设计是要结合键值存储和关系型数据库最好的特性。键值存储，因为非常简单，所以速度极快而且相对容易伸缩。关系型数据库更难伸缩，至少水平伸缩很难，但拥有丰富的数据模型和强大的查询语言。如果MongoDB能介于两者之间，就能成为一款易于伸缩、能存储丰富数据结构、提供复杂查询机制的数据库。

在使用场景方面，MongoDB非常适合用做以下应用程序的主要数据存储——Web应用程序、分析与记录应用程序，以及任何要求有中等级别缓存的应用程序。此外，由于能方便地存储无Schema数据，MongoDB还很适合保存事先无法知晓其数据结构的数据。

# MongoDB的历史及版本演变

2007年10月，MongoDB由10gen团队所发展。2009年2月首度推出。

2012年05月23日，MongoDB2.1 开发分支发布了! 该版本采用全新架构，包含诸多增强。

2012年06月06日，MongoDB 2.0.6 发布，分布式文档数据库。

2013年04月23日，MongoDB 2.4.3 发布，此版本包括了一些性能优化，功能增强以及bug修复。

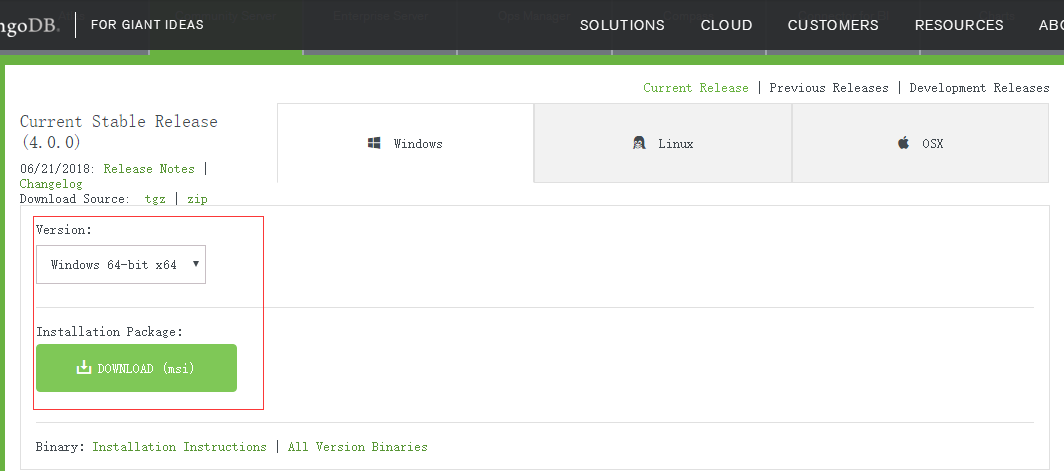
2013年08月20日，MongoDB 2.4.6 发布，是目前最新的稳定版。

# MongoDB的下载和安装

6.1支持的操作系统版本

|  |
| --- |
| OS X 32-bit  OS X 64-bit  Linux 32-bit  Linux 64-bit  Windows 32-bit  Windows 64-bit  Solaris i86pc  Solaris 64 |

6.2下载地址[http://www.mongodb.org/downloads](mk:@MSITStore:C:\Users\汪进银\AppData\Local\Temp\360zip$Temp\360$0\MongoDB参考手册.CHM::/%20http:/www.mongodb.org/downloads)



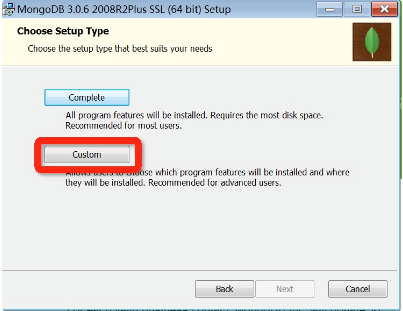
6.3windows平台下安装MongoDB

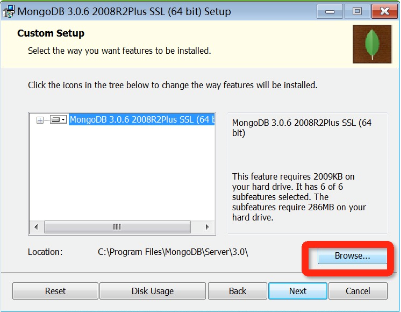
注意：在 MongoDB2.2 版本后已经不再支持 Windows XP 系统。

6.3.1在官网下载相应的版本（[**http://www.mongodb.org/downloads**](mk:@MSITStore:C:\Users\汪进银\AppData\Local\Temp\360zip$Temp\360$0\MongoDB参考手册.CHM::/%20http:/www.mongodb.org/downloads)）

6.3.2根据你的系统下载 32 位或 64 位的 .msi 文件，下载后双击该文件，按操作提示安装即可。

安装过程中，你可以通过点击 "Custom(自定义)" 按钮来设置你的安装目录。





6.3.3必须需要指定一个数据存放的目录 默认（/data/db）如没有则会 启动失败

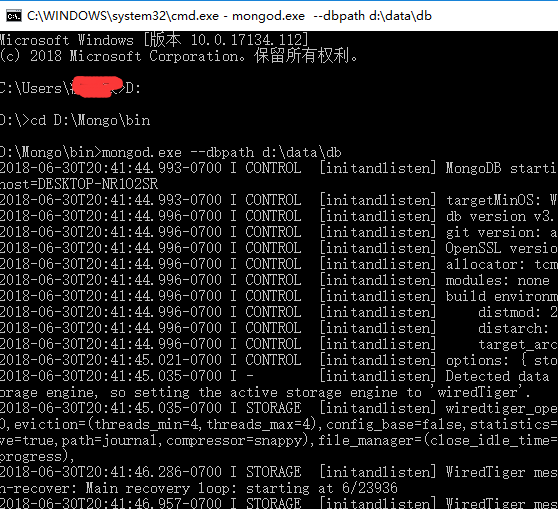
A．建D:\data\db路径

在命令行下运行MongoDB服务器

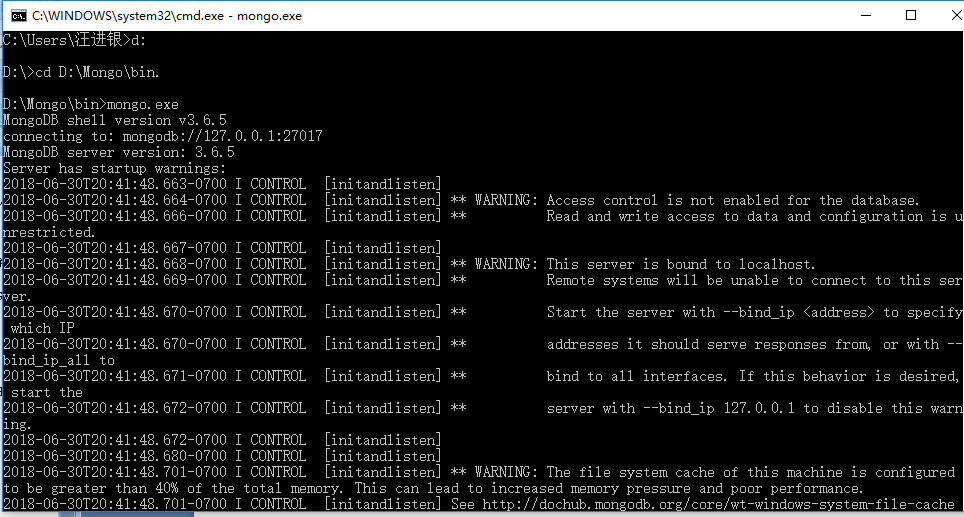
B．为了从命令提示符下运行MongoDB服务器，你必须从MongoDB目录的bin目录中执行mongod.exe文件。

|  |
| --- |
| mongod.exe --dbpath D:\data\db |

如果执行成功，会输出如下信息：



Mongo.exe 连接服务器（另外打开一个cmd窗口）进入bin目录下执行Mongo.exe



6.3.4配置服务器

管理员模式打开命令行窗口 创建目录，执行下面的语句来创建数据库和日志文件的目录：

|  |
| --- |
| Mkdir d:\data\db //数据保留文件  mkdir d:\data\log //日志存放文件 |

创建配置文件：

创建一个配置文件。该文件必须设置 systemLog.path 参数，包括一些附加的配置选更好。例如，创建一个配置文件位于 D:\Mongo\mongod.cfg，其中指定 systemLog.path 和 storage.dbPath。具体配置内容如下：

|  |
| --- |
| systemLog:  destination: file  path: d:\data\log\mongod.log  storage:  dbPath: d:\data\db |

6.3.5安装MongoDB服务器

通过执行mongod.exe，使用--install选项来安装服务，使用--config选项来指定之前创建的配置文件。

|  |
| --- |
| d:\Mongo\bin\mongod.exe --config "d:\Mongo\mongod.cfg" --install |

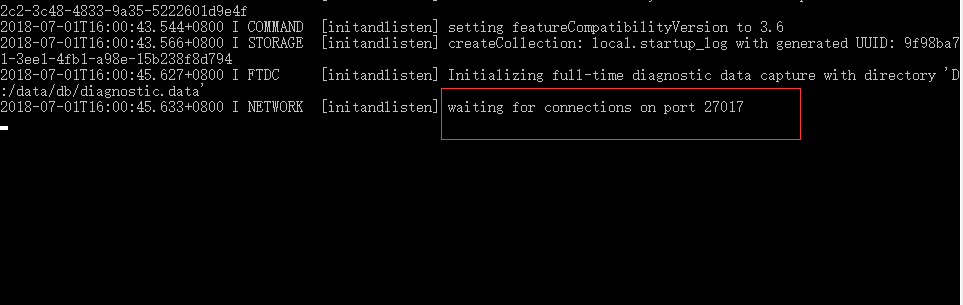
6.3.6启动,关闭，移出服务等

|  |
| --- |
| Net start MongoDB //启动服务  Net stop MongoDB //停止服务  D:\Mongo\bin\mongod.exe –remove //移出服务 |

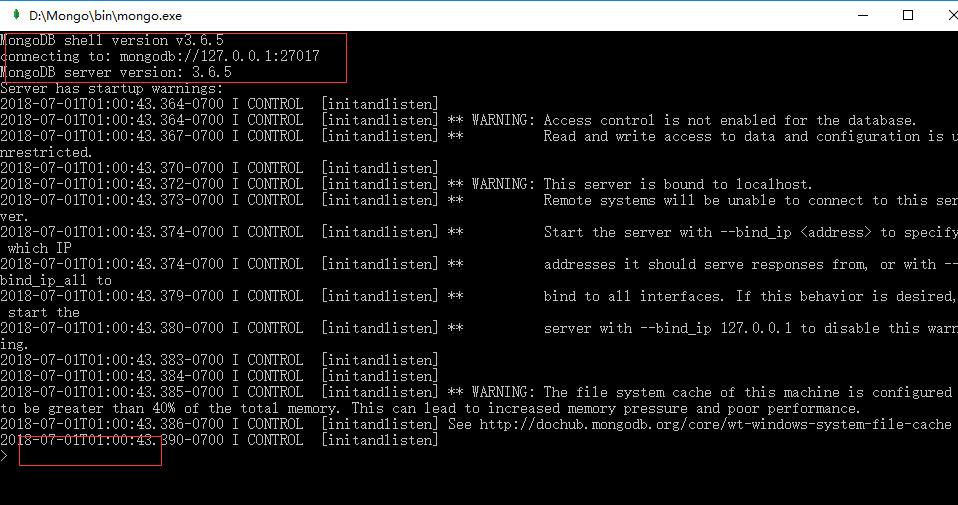
# MongoDB后台管理shell（建立一个简单的数据库）

7.1用Shell来操作MongoDB，对数据库进行增删查改

进入bin，运行mongod启动MongoDB数据库服务器（mongod.exe）



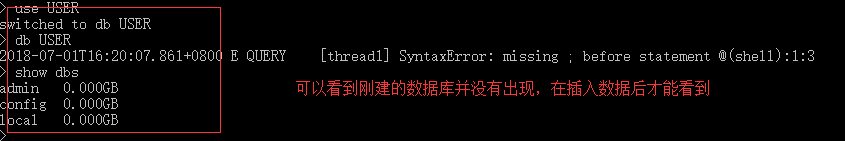
进入bin 运行 运行mongo.exe进入shell



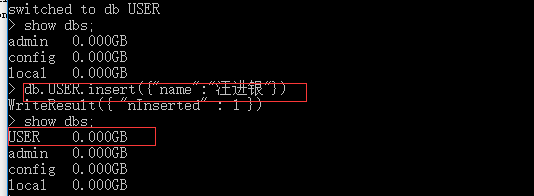
7.2Shell操作数据库

7.2.1创建数据库及查看所有数据库

|  |
| --- |
| use USER //查询数据库，如果没有则创建该数据库  show dbs //查看所有数据库 |



可以看到，我们刚创建的数据库 runoob 并不在数据库的列表中， 要显示它，我们需要向 USER 数据库插入一些数据。



7.3删除数据库

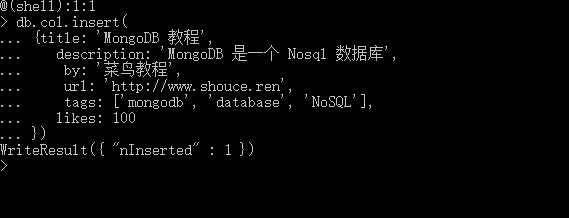
|  |
| --- |
| db.dropDatabase() //删除数据库 |

7.4插入文档

|  |
| --- |
| db.COLLECTION\_NAME.insert(document) //插入文档 |

示例如下

|  |
| --- |
| db.col.insert(  {title: 'MongoDB 教程',  description: 'MongoDB 是一个 Nosql 数据库',  by: '菜鸟教程',  url: 'http://www.shouce.ren',  tags: ['mongodb', 'database', 'NoSQL'],  likes: 100  }) |



|  |
| --- |
| db.col.find() //查询文档 |



第二种插入方式：

|  |
| --- |
| document=({title: 'MongoDB 教程',  description: 'MongoDB 是一个 Nosql 数据库',  by: '菜鸟教程',  url: 'http://www.shouce.ren',  tags: ['mongodb', 'database', 'NoSQL'],  likes: 100  });  db.col.insert(document)  WriteResult({ "nInserted" : 1 }) |

7.5更新文档

MongoDB 使用 **update()** 和 **save()** 方法来更新集合中的文档。接下来让我们详细来看下两个函数的应用及其区别。

7.5.1update方法

|  |
| --- |
| db.collection.update(  <query>,  <update>,  {  upsert: <boolean>,  multi: <boolean>,  writeConcern: <document>  }  ) |

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

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

set后面的

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,tru

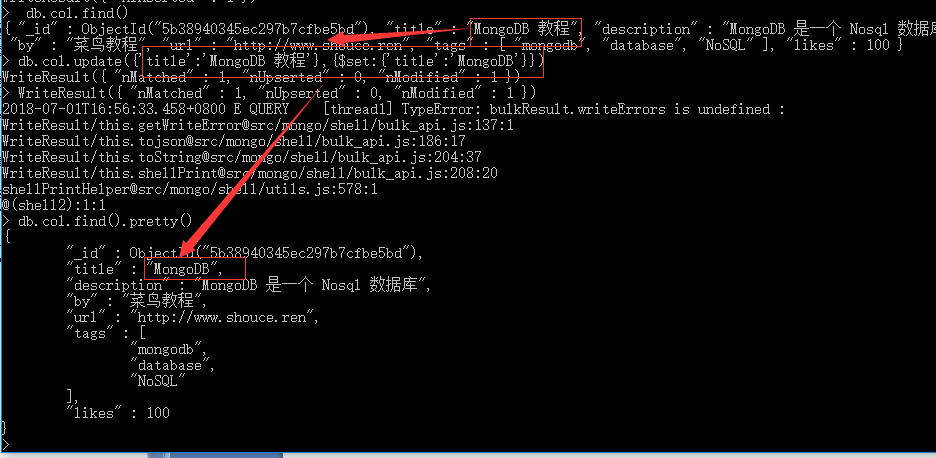
为插入，默认是false，不插入。

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。

示例：

|  |
| --- |
| db.col.update({'title':'MongoDB 教程'},{$set:{'title':'MongoDB'}})  WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) # 输出信息  > db.col.find().pretty()  {  "\_id" : ObjectId("56064f89ade2f21f36b03136"),  "title" : "MongoDB",  "description" : "MongoDB 是一个 Nosql 数据库",  "by" : "菜鸟教程",  "url" : "http://www.runoob.com",  "tags" : [  "mongodb",  "database",  "NoSQL"  ],  "likes" : 100  }  > |



7.5.2save方法

save() 方法通过传入的文档来替换已有文档。语法格式如下：

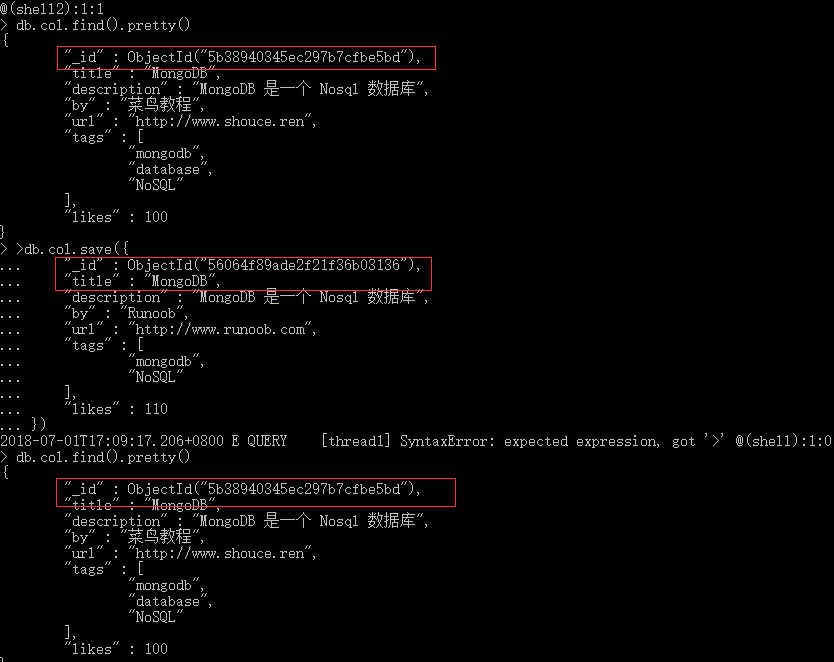
|  |
| --- |
| db.collection.save(  <document>,  {  writeConcern: <document>  }  ) |

document : 文档数据。

**writeConcern** :可选，抛出异常的级别

示例：

|  |
| --- |
| >db.col.save({      "\_id" : ObjectId("56064f89ade2f21f36b03136"),  "title" : "MongoDB",  "description" : "MongoDB 是一个 Nosql 数据库",  "by" : "Runoob",  "url" : "http://www.runoob.com",  "tags" : [  "mongodb",  "NoSQL"  ],  "likes" : 110  }) |



# 删除文档

remove() 方法的基本语法格式如下所示：

|  |
| --- |
| db.collection.remove(  <query>,  {  justOne: <boolean>,  writeConcern: <document>  }  .) |

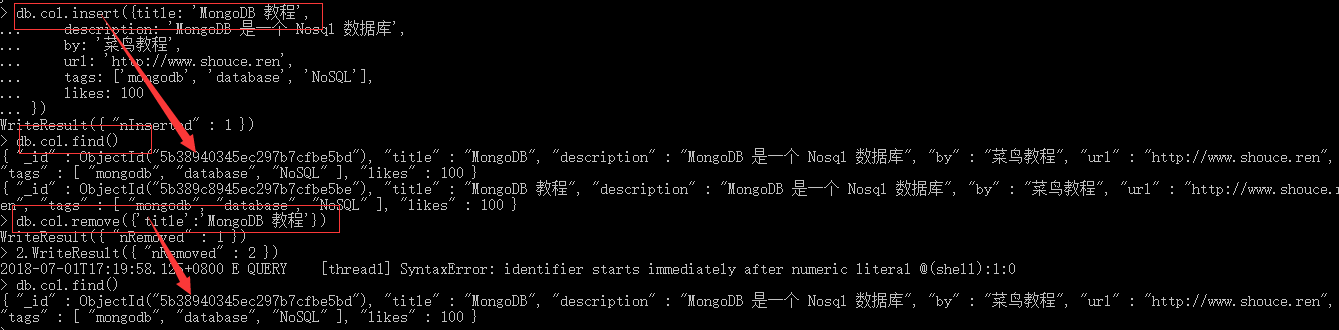
query :（可选）删除的文档的条件。

justOne : （可选）如果设为 true 或 1，则只删除一个文档。

writeConcern :（可选）抛出异常的级别。

示例：

|  |
| --- |
| db.col.insert({title: 'MongoDB 教程', //在插入一条相同数据  description: 'MongoDB 是一个 Nosql 数据库',  by: '菜鸟教程',  url: 'http://www.shouce.ren',  tags: ['mongodb', 'database', 'NoSQL'],  likes: 100  })  db.col.find() //查询  db.col.remove({'title':'MongoDB 教程'})//删除  db.col.find() //查询 |



# 基本查询

9.1基本查询

MongoDB 查询数据的语法格式如下：

|  |
| --- |
| db.COLLECTION\_NAME.find() |

find() 方法以非结构化的方式来显示所有文档

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.col.find().pretty() |

前面示例多次使用（这里不进行操作）

9.2 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 |

## 9.3 MongoDB 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，及常规 SQL 的 AND 条件。

语法格式如下：

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

示例：

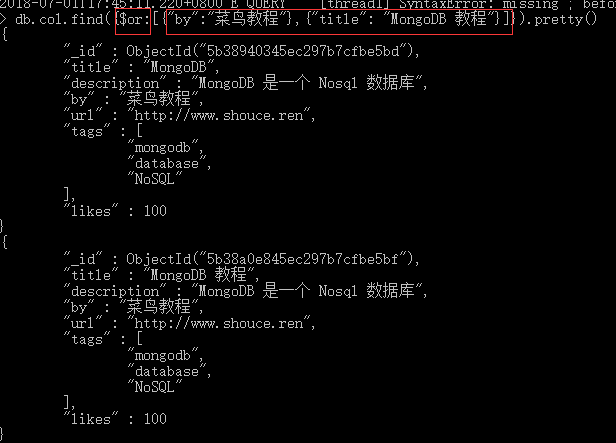
|  |
| --- |
| db.col.find({"by":"菜鸟教程", "title":"MongoDB 教程"}).pretty()  //根据"by":"菜鸟教程", "title":"MongoDB 教程"查询 |



## 9.4MongoDB OR 条件

MongoDB OR 条件语句使用了关键字 **$or**,语法格式如下：

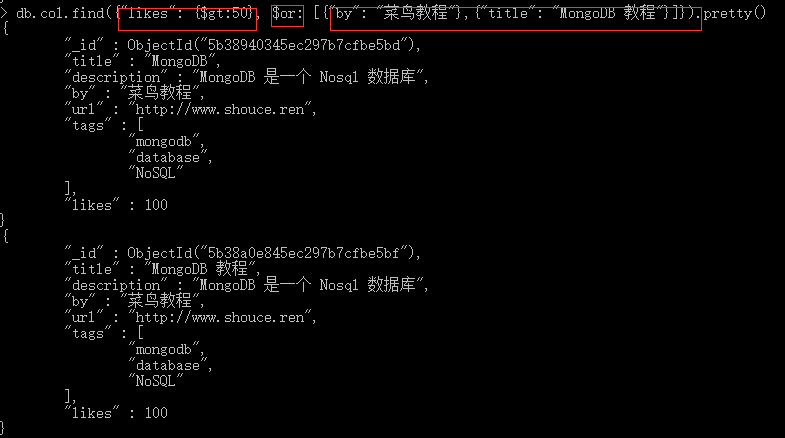
|  |
| --- |
| db.col.find(  {  $or: [  {key1: value1}, {key2:value2}  ]  }  ).pretty() |



## 9.5AND 和 OR 联合使用

示例：

|  |
| --- |
| db.col.find({"likes": {$gt:50}, $or: [{"by": "菜鸟教程"},{"title": "MongoDB 教程"}]}).pretty() |



# MongoDB常见面试题

10.1Nosql数据库的兴起

随着大数据时代的到来，关系型数据库已经无法满足web2.0的需求。

无法满足海量实时数据的存储需求

无法满足高并发的需求

无法满足高扩展和高可用的需求

10.1什么是NoSQL数据库？NoSQL和RDBMS有什么区别？在哪些情况下使用和不使用NoSQL数据库？

Nosql数据库：英文解释 不仅仅是数据库 泛指：关系型数据库：

结构化数据：指具有固定格式或有限长度的数据，如数据库，元数据等。

非结构化数据：指不定长或无固定格式的数据，如邮件，word文档等。

关系型数据库采用的结构化的数据：

NoSQL采用的是键值对的方式存储数据。

在处理结构化/半结构化大数据时，水平方向需要扩展时，随时应对动态增加的数据项时可以优先考虑使用NoSQL数据库

10.2为什么要使用mongoDB

**特点:**

* 高性能、易部署、易使用，存储数据非常方便。主要功能特性有：
* 面向集合存储，易存储对象类型的数据。
* 模式自由。
* 支持动态查询。
* 支持完全索引，包含内部对象。
* 支持查询。
* 支持复制和故障恢复。
* 使用高效的二进制数据存储，包括大型对象（如视频等）。
* 自动处理碎片，以支持云计算层次的扩展性
* 支持Python，PHP，Ruby，Java，C，C#，Javascript，Perl及C++语言的驱动程序，社区中也提供了对Erlang及.NET等平台的驱动程序。
* 文件存储格式为BSON（一种JSON的扩展）。
* 可通过网络访问。

**功能:**

* **面向集合的存储：**适合存储对象及JSON形式的数据。
* **动态查询：**Mongo支持丰富的查询表达式。查询指令使用JSON形式的标记，可轻易查询文档中内嵌的对象及数组。
* **完整的索引支持：**包括文档内嵌对象及数组。Mongo的查询优化器会分析查询表达式，并生成一个高效的查询计划。
* **查询监视：**Mongo包含一个监视工具用于分析数据库操作的性能。
* **复制及自动故障转移：**Mongo数据库支持服务器之间的数据复制，支持主-从模式及服务器之间的相互复制。复制的主要目标是提供冗余及自动故障转移。
* **高效的传统存储方式：**支持二进制数据及大型对象（如照片或图片）
* **自动分片以支持云级别的伸缩性：**自动分片功能支持水平的数据库集群，可动态添加额外的机器。

**使用场景：**

* 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。

**3. MySQL与MongoDB之间最基本的差别是什么?**

MySQL和MongoDB两者都是免费开源的数据库。MySQL和MongoDB有许多基本差别包括数据的表示(data representation)，查询，关系，事务，schema的设计和定义，标准化(normalization)，速度和性能。

通过比较MySQL和MongoDB，实际上我们是在比较关系型和非关系型数据库，即数据存储结构不同。[详细阅读](http://theprofessionalspoint.blogspot.in/2013/12/mysql-vs-mongodb-basic-differences.html)

**4. 你怎么比较MongoDB、CouchDB及CouchBase?**

MongoDB和CouchDB都是面向文档的数据库。MongoDB和CouchDB都是开源NoSQL数据库的最典型代表。 除了都以文档形式存储外它们没有其他的共同点。MongoDB和CouchDB在数据模型实现、接口、对象存储以及复制方法等方面有很多不同。

细节可以参见下面的链接：

[MongDB vs CouchDB](http://theprofessionalspoint.blogspot.in/2014/01/couchdb-vs-couchbase-differences-and.html)

[CouchDB vs CouchBase](http://theprofessionalspoint.blogspot.in/2014/01/couchdb-vs-couchbase-differences-and.html)

**5. MongoDB成为最好NoSQL数据库的原因是什么?**

以下特点使得MongoDB成为最好的NoSQL数据库：

* 面向文件的
* 高性能
* 高可用性
* 易扩展性
* 丰富的查询语言

**6.32位系统上有什么细微差别?**

journaling会激活额外的内存映射文件。这将进一步抑制32位版本上的数据库大小。因此，现在journaling在32位系统上默认是禁用的。

**7. journal回放在条目(entry)不完整时(比如恰巧有一个中途故障了)会遇到问题吗?**

每个journal (group)的写操作都是一致的，除非它是完整的否则在恢复过程中它不会回放。

**8. 分析器在MongoDB中的作用是什么?**

MongoDB中包括了一个可以显示数据库中每个操作性能特点的数据库分析器。通过这个分析器你可以找到比预期慢的查询(或写操作);利用这一信息，比如，可以确定是否需要添加索引。

**9. 名字空间(namespace)是什么?**

MongoDB存储BSON对象在丛集(collection)中。数据库名字和丛集名字以句点连结起来叫做名字空间(namespace)。

**10. 如果用户移除对象的属性，该属性是否从存储层中删除?**

是的，用户移除属性然后对象会重新保存(re-save())。

**11. 能否使用日志特征进行安全备份?**

是的。

**12. 允许空值null吗?**

对于对象成员而言，是的。然而用户不能够添加空值(null)到数据库丛集(collection)因为空值不是对象。然而用户能够添加空对象{}。

**13. 更新操作立刻fsync到磁盘?**

不会，磁盘写操作默认是延迟执行的。写操作可能在两三秒(默认在60秒内)后到达磁盘。例如，如果一秒内数据库收到一千个对一个对象递增的操作，仅刷新磁盘一次。(注意，尽管fsync选项在命令行和经过getLastError\_old是有效的)(译者：也许是坑人的面试题??)。

**14. 如何执行事务/加锁?**

MongoDB没有使用传统的锁或者复杂的带回滚的事务，因为它设计的宗旨是轻量，快速以及可预计的高性能。可以把它类比成MySQL MylSAM的自动提交模式。通过精简对事务的支持，性能得到了提升，特别是在一个可能会穿过多个服务器的系统里。

**15. 为什么我的数据文件如此庞大?**

MongoDB会积极的预分配预留空间来防止文件系统碎片。

**16. 启用备份故障恢复需要多久?**

从备份数据库声明主数据库宕机到选出一个备份数据库作为新的主数据库将花费10到30秒时间。这期间在主数据库上的操作将会失败--包括写入和强一致性读取(strong consistent read)操作。然而，你还能在第二数据库上执行最终一致性查询(eventually consistent query)(在slaveOk模式下)，即使在这段时间里。

**17. 什么是master或primary?**

它是当前备份集群(replica set)中负责处理所有写入操作的主要节点/成员。在一个备份集群中，当失效备援(failover)事件发生时，一个另外的成员会变成primary。

**18. 什么是secondary或slave?**

Seconday从当前的primary上复制相应的操作。它是通过跟踪复制oplog(local.oplog.rs)做到的。

**19. 我必须调用getLastError来确保写操作生效了么?**

不用。不管你有没有调用getLastError(又叫"Safe Mode")服务器做的操作都一样。调用getLastError只是为了确认写操作成功提交了。当然，你经常想得到确认，但是写操作的安全性和是否生效不是由这个决定的。

**20. 我应该启动一个集群分片(sharded)还是一个非集群分片的 MongoDB 环境?**

为开发便捷起见，我们建议以非集群分片(unsharded)方式开始一个 MongoDB 环境，除非一台服务器不足以存放你的初始数据集。从非集群分片升级到集群分片(sharding)是无缝的，所以在你的数据集还不是很大的时候没必要考虑集群分片(sharding)。

**21. 分片(sharding)和复制(replication)是怎样工作的?**

每一个分片(shard)是一个分区数据的逻辑集合。分片可能由单一服务器或者集群组成，我们推荐为每一个分片(shard)使用集群。

**22. 数据在什么时候才会扩展到多个分片(shard)里?**

MongoDB 分片是基于区域(range)的。所以一个集合(collection)中的所有的对象都被存放到一个块(chunk)中。只有当存在多余一个块的时候，才会有多个分片获取数据的选项。现在，每个默认块的大小是 64Mb，所以你需要至少 64 Mb 空间才可以实施一个迁移。

**23. 当我试图更新一个正在被迁移的块(chunk)上的文档时会发生什么?**

更新操作会立即发生在旧的分片(shard)上，然后更改才会在所有权转移(ownership transfers)前复制到新的分片上。

24. 如果在一个分片(shard)停止或者很慢的时候，我发起一个查询会怎样?

如果一个分片(shard)停止了，除非查询设置了“Partial”选项，否则查询会返回一个错误。如果一个分片(shard)响应很慢，MongoDB则会等待它的响应。

**25. 我可以把moveChunk目录里的旧文件删除吗?**

没问题，这些文件是在分片(shard)进行均衡操作(balancing)的时候产生的临时文件。一旦这些操作已经完成，相关的临时文件也应该被删除掉。但目前清理工作是需要手动的，所以请小心地考虑再释放这些文件的空间。

**26. 我怎么查看 Mongo 正在使用的链接?**

db.\_adminCommand("connPoolStats");

**27. 如果块移动操作(moveChunk)失败了，我需要手动清除部分转移的文档吗?**

不需要，移动操作是一致(consistent)并且是确定性的(deterministic);一次失败后，移动操作会不断重试;当完成后，数据只会出现在新的分片里(shard)。

**28. 如果我在使用复制技术(replication)，可以一部分使用日志(journaling)而其他部分则不使用吗?**

可以。

**29.当更新一个正在被迁移的块（Chunk）上的文档时会发生什么？**

更新操作会立即发生在旧的块（Chunk）上，然后更改才会在所有权转移前复制到新的分片上。

**30.MongoDB在A:{B,C}上建立索引，查询A:{B,C}和A:{C,B}都会使用索引吗？**

不会，只会在A:{B,C}上使用索引。

**31.如果一个分片（Shard）停止或很慢的时候，发起一个查询会怎样？**

如果一个分片停止了，除非查询设置了“Partial”选项，否则查询会返回一个错误。如果一个分片响应很慢，MongoDB会等待它的响应。

**32. MongoDB支持存储过程吗？如果支持的话，怎么用？**

MongoDB支持存储过程，它是javascript写的，保存在db.system.js表中。

**33.如何理解MongoDB中的GridFS机制，MongoDB为何使用GridFS来存储文件？**

GridFS是一种将大型文件存储在MongoDB中的文件规范。使用GridFS可以将大文件分隔成多个小文档存放，这样我们能够有效的保存大文档，而且解决了BSON对象有限制的问题。

## MongoDB要注意的问题

1 因为MongoDB是全索引的，所以它直接把索引放在内存中，因此最多支持2.5G的数据。如果是64位的会更多。

2 因为没有恢复机制，因此要做好数据备份

3 因为默认监听地址是127.0.0.1，因此要进行身份验证，否则不够安全；如果是自己使用，建议配置成localhost主机名

4 通过GetLastError确保变更。（**这个不懂，实际中没用过**）