MongoDB介绍

## 应用篇

## 一：插入文档

1插入举例

db.inventory.insert(

 {

 item: "ABC1",

 details: {

 model: "14Q3",

 manufacturer: "XYZ Company"

 },

 stock: [ { size: "S", qty: 25 }, { size: "M", qty: 50 } ],

 category: "clothing"

 }

)

db.inventory.find()



2.插入多个文档

var mydocuments =

 [

 {

 item: "ABC2",

 details: { model: "14Q3", manufacturer: "M1 Corporation" },

 stock: [ { size: "M", qty: 50 } ],

 category: "clothing"

 },

 {

 item: "MNO2",

 details: { model: "14Q3", manufacturer: "ABC Company" },

 stock: [ { size: "S", qty: 5 }, { size: "M", qty: 5 }, { size: "L", qty: 1 } ],

 category: "clothing"

 },

 {

 item: "IJK2",

 details: { model: "14Q2", manufacturer: "M5 Corporation" },

 stock: [ { size: "S", qty: 5 }, { size: "L", qty: 1 } ],

 category: "houseware"

 }

 ];

db.inventory.insert( mydocuments );

3.利用bulk插入多个文档(2.6以后版本)

3.1初始化指定集合的批处理操作构造器

var bulk = db.inventory.initializeUnorderedBulkOp();

3.2插入添加进builk

bulk.insert(

 {

 item: "BE10",

 details: { model: "14Q2", manufacturer: "XYZ Company" },

 stock: [ { size: "L", qty: 5 } ],

 category: "clothing"

 }

);

bulk.insert(

 {

 item: "ZYT1",

 details: { model: "14Q1", manufacturer: "ABC Company" },

 stock: [ { size: "S", qty: 5 }, { size: "M", qty: 5 } ],

 category: "houseware"

 }

);

3.3执行

bulk.execute();

## 二：删除文档

1.删除所有文档

db.inventory.remove({})

2.删除满足条件的文档

db.inventory.remove( { type : "food" } )

3.删除一条满足条件的文档

db.inventory.remove( { type : "food" }, true ) or db.inventory.remove( { type : "food" }, 1 )

## 三: 修改文档

1.修改字段值

db.inventory.update(

 { item: "MNO2" },

 {

 $set: {

 category: "apparel",

 details: { model: "14Q3", manufacturer: "XYZ Company" }

 },

 $currentDate: { lastModified: true }

 }

)

修改匹配的第一条文档

2.修改内嵌的字段值

db.inventory.update(

 { item: "ABC1" },

 { $set: { "details.model": "14Q2" } }

)

3.修改匹配的多行文档

db.inventory.update(

 { category: "clothing" },

 {

 $set: { category: "apparel" },

 $currentDate: { lastModified: true }

 },

 { multi: true }

)

4.替换一个文档

db.inventory.update(

 { item: "BE10" },

 {

 item: "BE05",

 stock: [ { size: "S", qty: 20 }, { size: "M", qty: 5 } ],

 category: "apparel"

 }

)

找到item为BE10的那个文档，用新文档替换，替换后\_id不变，其他字段为新文档的字段。

5.在替换一个文档操作时使用upsert(找到匹配文档就替换，无匹配就插入新文档)

db.inventory.update(

 { item: "TBD1" },

 {

 item: "TBD1",

 details: { "model" : "14Q4", "manufacturer" : "ABC Company" },

 stock: [ { "size" : "S", "qty" : 25 } ],

 category: "houseware"

 },

 { upsert: true }

)

6. 在更新操作时候使用upsert(找到匹配文档就更新，无匹配就插入新文档)

db.inventory.update(

 { item: "TBD2" },

 {

 $set: {

 details: { "model" : "14Q3", "manufacturer" : "IJK Co." },

 category: "houseware"

 }

 },

 { upsert: true }

)

## 四：查询文档

1.查询集合下所有文档

db.inventory.find( {} )

db.inventory.find()

2.条件查询

db.inventory.find( { type: "snacks" } )

db.inventory.find( { type: { $in: [ 'food', 'snacks' ] } } )

3.And 查询

db.inventory.find( { type: 'food', price: { $lt: 9.95 } } )

4.Or 查询

db.inventory.find(

 {

 $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]

 }

)

5.And Or 查询

db.inventory.find(

 {

 type: 'food',

 $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]

 }

)

6.内嵌的文档查询

db.inventory.find(

 {

 details:

 {

 "model" : "14Q3",

 "manufacturer" : "XYZ Company"

 }

 }

)

也可写成

db.inventory.find(

 {

 "details.model":"14Q3",

 "details.manufacturer": "XYZ Company"

 }

)

7.数组查询

集合中有如下文档

{ \_id: 5, type: "food", item: "aaa", ratings: [ 5, 8, 9 ] }

{ \_id: 6, type: "food", item: "bbb", ratings: [ 5, 9 ] }

{ \_id: 7, type: "food", item: "ccc", ratings: [ 9, 5, 8 ] }

7.1查找

db.inventory.find( { ratings: [ 5, 8, 9 ] } )

结果

{ \_id: 5, type: "food", item: "aaa", ratings: [ 5, 8, 9 ] }

7.2查找包含5的

db.inventory.find( { ratings: 5 } )

7.3查找第一个元素是5的

db.inventory.find( { 'ratings.0': 5 } )

7.4 查找包含一个>5且<9的

db.inventory.find( { ratings: { $elemMatch: { $gt: 5, $lt: 9 } } } )

7.5查找包含一个>5而另一个<9的

db.inventory.find( { ratings: { $gt: 5, $lt: 9 } } )

还有其他数据查询参考https://docs.mongodb.org/manual/tutorial/query-documents/

8.限制返回的列

db.inventory.find( { category: 'houseware' }, { category: 1, item: 1, \_id:0 } )

0表示不输出\_id(false)

1表示输出(true)

9.控制数组的输出

已有数据

{

 \_id: 1,

 scores: [

 { attempt: 1, score: 10 },

 { attempt: 2 , score:8 }

 ]

}

执行

db.students.update(

 { \_id: 1 },

 {

 $push: {

 scores: {

 $each: [ { attempt: 3, score: 7 }, { attempt: 4, score: 4 } ],

 $sort: { score: 1 },

 $slice: -3

 }

 }

 }

)

查询集合结果

{

 "\_id" : 1,

 "scores" : [

 { "attempt" : 3, "score" : 7 },

 { "attempt" : 2, "score" : 8 },

 { "attempt" : 1, "score" : 10 }

 ]

}

$each表示追加两个数组内对象

$sort 表示数组内的元素升序

$slice 是一个<=0的数，0表示空数组，如果是负数表示从后向前截取负数绝对值长度的数组

修改器参考：http://blog.csdn.net/mengxiangyue/article/details/18560357

## 五: 游标

var myCursor = db.inventory.find( {} );

while (myCursor.hasNext()) {

 print(tojson(myCursor.next()));

}

或

var myCursor = db.inventory.find( {} );

while (myCursor.hasNext()) {

 printjson(myCursor.next());

}

或

var myCursor = db.inventory.find( { type: 'food' } );

myCursor.forEach(printjson);

## 六: 存储过程

MongoDB的存储过程用JavaScript写，在db.system.js集合（表）中

每一步：定义函数（可以理解为mysql的存储过程）
db.system.js.save({\_id:"addNumbers",value:function(x,y) {return x+y;}});

db.system.js.find();#可以看到自定义的方法

第二步：调用此函数
db.eval('addNumbers(3,4.2)');
db.eval(function(){addnumbers(3,5)});
db.eval(function(a,b){return a+b;},1,5); //参数a=1,b=5 ,函数是匿名函数 输出是 6

## 七：索引

索引通常能够极大的提高查询的效率。在系统中使用查询时，应该考虑建立相关的索引。

1.创建一个索引

db.records.createIndex( { “userid”: 1 } )

2.查询索引

db.system.indexes.find();

3.创建一个唯一索引

db.records.createIndex( { “userid”: 1 } ,{“unique”:true});

4.删除索引

db.dropIndex({“userid”:1})

## 八：GridFS

1.作用：

以GridFS 规范提供了一种透明的机制，可

以将一个大文件分割成为多个较小的文档，这样的机制允许我们有效的保存大文件对象，特

别对于那些巨大的文件，比如视频、高清图片等。

2.介绍

GridFS 使用两个表来存储数据：

* files 包含元数据对象
* chunks 包含其他一些相关信息的二进制块

为了使多个GridFS 命名为一个单一的数据库，文件和块都有一个前缀，默认情况下，前缀

是fs，所以任何默认的GridFS 存储将包括命名空间fs.files 和fs.chunks。各种第三方语言的

驱动有权限改变这个前缀，所以你可以尝试设置另一个GridFS 命名空间用于存储照片，它

的具体位置为:photos.files 和photos.chunks

操作命令

1.上传

mongofiles /h 127.0.0.1 /port 27017 put E:\重要文件\电子版-证

件\1.jpg

2.查看有哪些GridFs文件

mongofiles  list

3.下载到本地

mongofiles /h 127.0.0.1 /port 27017 get  E:\重要文件\电子版-证

件\1.jpg

mongo连接mongoDB后

db.fs.files.find()

字段说明**:**

* Filename: 存储的文件名
* chunkSize: chunks 分块的大小
* uploadDate: 入库时间
* md5: 此文件的md5 码
* length: 文件大小, 单位”字节”

看来fs.files 中存储的是一些基础的元数据信息

db.fs.chunks.find()

这里存储的是一些实际内容数据信息

## 九：MapReduce

MongoDB 的MapReduce 相当于Mysql 中的"group by"，所以在MongoDB 上使用 Map/Reduce进行并行"统计"很容易。

使用MapReduce 要实现两个函数 Map 函数和Reduce 函数，Map 函数调用emit(key, value)，

遍历collection中所有的记录，将key与value传递给Reduce 函数进行处理。Map函数和Reduce

函数可以使用JavaScript 来实现，可以通过db.runCommand 或mapReduce 命令来执行一个

MapReduce 的操作

db.students.insert({classid:1, age:14, name:'Tom'})

db.students.insert({classid:1, age:12, name:'Jacky'})

db.students.insert({classid:2, age:16, name:'Lily'})

db.students.insert({classid:2, age:9, name:'Tony'})

db.students.insert({classid:2, age:19, name:'Harry'})

db.students.insert({classid:2, age:13, name:'Vincent'})

db.students.insert({classid:1, age:14, name:'Bill'})

db.students.insert({classid:2, age:17, name:'Bruce'})

m = function() { emit(this.classid, 1) }

r = function(key, values) {

 var x = 0;

 values.forEach(function(v) { x += v });

 return x;

 }

f = function(key, value) { return {classid:key, count:value}; }

res = db.runCommand({

 mapreduce:"students",

 map:m,

 reduce:r,

 out:"students\_res",

 finalize:f

 });

db.students\_res.find()

