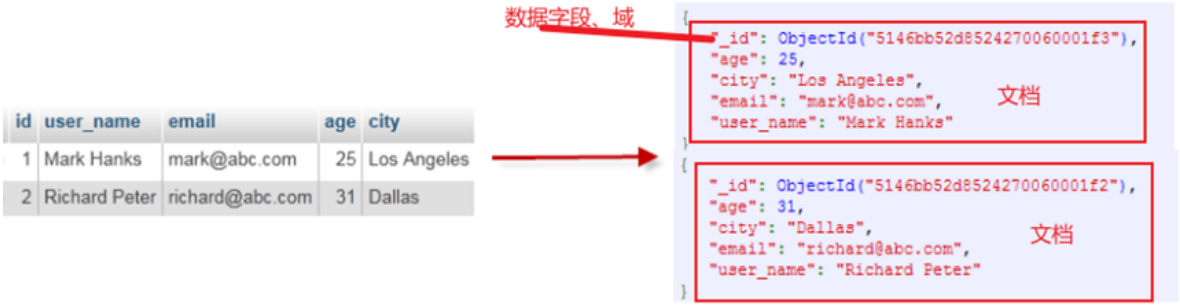
Mysql和Mongo对应关系

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

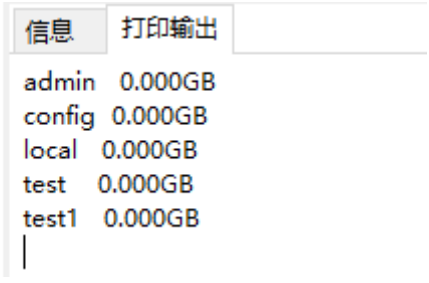


常用命令

数据库db操作

显示所有数据库：show dbs

可以显示所有数据的列表，新创建的数据库如果没数据从这里是不显示数据库的



显示当前数据库对象：db

连接/切换/创建数据库：use dbname

如果数据库不存在，就创建数据库

```
use test
```

删除当前数据库db.dropDatabase()

所以删除只有要先使用use 使用当前数据库

```
db.dropDatabase()
```

集合（表）and文档(行数据)操作

查看当前数据库所有的集合

```
show tables
show collections
```

插入/创建数据到collection：表

```
db.myCollection.insert({"name":"sz","age":123})
```

db.colletctionname.insert({})会自动创建collection集合

*在 MongoDB 中，集合只有在内容插入后才会创建**! 就是说，创建集合(数据表)后要再插入一个文档(记录)，集合才会真正创建。*

创建集合

```
db.createCollection("site")
```

删除集合

```
db.myCollection.drop()
```

更新文档

```
db.myCollection.update({'name':'sz'},{$set:{'name':'sunzhong'}})
```

插入不同文档（允许不同列名）

```
db.myCollection.insert({"name":"sz","age":123})
db.myCollection.insert({"name":"skk","age":123})
db.myCollection.insert({"name":"stt","age":123})
db.myCollection.insert({"tittle":"斗破苍穹","author":"土豆"})
db.myCollection.find()
```

_id	name	age	tittle	author
62d2ab0d802200007c006ea7	sz	123	(N/A)	(N/A)
62d2ab0d802200007c006ea8	skk	123	(N/A)	(N/A)
62d2ab0d802200007c006ea9	stt	123	(N/A)	(N/A)
62d2ab64802200007c006eaa	(N/A)	(N/A)	斗破苍穹	土豆

相同记录的插入（id随机不同）

```
father={"name":"sqz"}
mother={"name":"yxx"}
db.myCollection.save(father)
db.myCollection.save(father)
db.myCollection.save(mother)
db.myCollection.find()
```

_id	name	age	tittle	author
62d2ab0d802200007c006ea7	sz	123	(N/A)	(N/A)
62d2ab0d802200007c006ea8	skk	123	(N/A)	(N/A)
62d2ab0d802200007c006ea9	stt	123	(N/A)	(N/A)
62d2ab64802200007c006eaa	(N/A)	(N/A)	斗破苍穹	土豆
62d2ac47802200007c006eab	stu	124	(N/A)	(N/A)
62d2ac47802200007c006eac	ssz	121	(N/A)	(N/A)
62d2ad6b802200007c006ead	sqz	(N/A)	(N/A)	(N/A)
62d2ad88802200007c006eae	sqz	(N/A)	(N/A)	(N/A)
62d2ad9e802200007c006eaf	yxx	(N/A)	(N/A)	(N/A)

mongodb数据类型

数据类型	描述
String	字符串。存储数据常用的数据类型。在 MongoDB 中，UTF-8 编码的字符串才是合法的。
Integer	整型数值。用于存储数值。根据你所采用的服务器，可分为 32 位或 64 位。
Boolean	布尔值。用于存储布尔值（真/假）。
Double	双精度浮点值。用于存储浮点值。
Min/Max keys	将一个值与 BSON（二进制的 JSON）元素的最低值和最高值相对比。
Array	用于将数组或列表或多个值存储为一个键。
Timestamp	时间戳。记录文档修改或添加的具体时间。
Object	用于内嵌文档。
Null	用于创建空值。
Symbol	符号。该数据类型基本上等同于字符串类型，但不同的是，它一般用于采用特殊符号类型的语言。
Date	日期时间。用 UNIX 时间格式来存储当前日期或时间。你可以指定自己的日期时间：创建 Date 对象，传入年月日信息。
Object ID	对象 ID。用于创建文档的 ID。
Binary Data	二进制数据。用于存储二进制数据。
Code	代码类型。用于在文档中存储 JavaScript 代码。

shell连接数据库

```
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，驱动都会尝试登录这个数据库

· **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@，连接并验证登录指定数据库。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

查询

查询所有记录

```
db.myCollection.find()
```

查询一条记录

```
db.myCollection.findOne()
```

_id	name	age
62d2ab0d802200007c006ea7	sz	123

删除记录

相当于where 会删除满足条件的多个记录

```
db.myCollection.remove({"name":"skk"});
db.myCollection.remove({"age":123});
```

and or 一起用

类似常规 SQL 语句为： 'where likes>50 AND (by = '菜鸟教程' OR title = 'MongoDB 教程')'

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

{
    "_id" : ObjectId("56063f17ade2f21f36b03133"),
    "title" : "MongoDB  教程",
    "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
}
```

限制条目数limit 跳过条目数 skip(=offset)

Find()..Limit() Find()..skip()

```
db.myCollection.find().limit(2)
```

_id	name	age
62d2ab0d802200007c006ea7	sz	123
62d2ab0d802200007c006ea8	skk	123

排序

.sort({key:1}) 1升序排 -1降序排

空的 也被差出来，在前面

```
db.myCollection.find().sort({"age":1})
```

_id	tittle	author	name	age
62d2ab64802200007c006eaa	斗破苍穹	土豆	(N/A)	(N/A)
62d2ac47802200007c006eac	(N/A)	(N/A)	ssz	121
62d2ab0d802200007c006ea7	(N/A)	(N/A)	sz	123
62d2ab0d802200007c006ea8	(N/A)	(N/A)	skk	123
62d2ab0d802200007c006ea9	(N/A)	(N/A)	stt	123
62d2ac47802200007c006eab	(N/A)	(N/A)	stu	124

索引

```
db.col.createIndex({"title":1,"description":-1})#联合索引，1升序-1降序
```

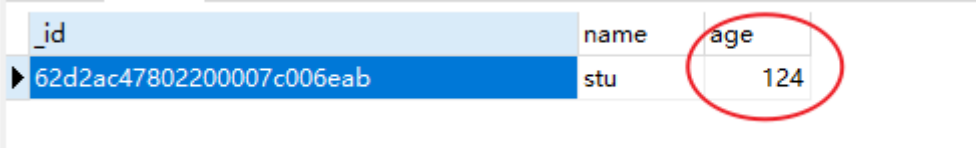
高级查询

比较运算符

等于	{:}	db.col.find({"by":"菜鸟教程"}).pretty()	where by = '菜鸟教程'
小于	{:{$lt:}}	db.col.find({"likes":{$lt:50}}).pretty()	where likes < 50

条件查询

```
db.myCollection.find({"age":{$gt:123}})
```



_id	name	age
62d2ac47802200007c006eab	stu	124

$all 匹配所有

```
db.myCollection.find({"age":{$gt:123}})
db.favorite.insert({"name":"sz","favthing":
["banana","apple","book","computer"]})
db.favorite.insert({"name":"sa","favthing":["banana","apple","book"]})
db.favorite.insert({"name":"sb","favthing":["banana","book","computer"]})
db.favorite.insert({"name":"sc","favthing":["banana","apple","computer"]})
db.favorite.find({"favthing":{$all:["book","apple"]}}).forEach(printjson)
```

```
27 db.favorite.insert({"name":"sz","favthing":["banana","apple","book","computer"]})
28 db.favorite.insert({"name":"sa","favthing":["banana","apple","book"]})
29 db.favorite.insert({"name":"sb","favthing":["banana","book","computer"]})
30 db.favorite.insert({"name":"sc","favthing":["banana","apple","computer"]})
31 db.favorite.find({"favthing":{$all:["book","apple"]}}).forEach(printjson)
```

信息 打印输出

```
{
        "_id" : ObjectId("62d2b015802200007c006eb0"),
        "name" : "sz",
        "favthing" : [
                "banana",
                "apple",
                "book",
                "computer"
        ]
}
{
        "_id" : ObjectId("62d2b015802200007c006eb1"),
        "name" : "sa",
        "favthing" : [
                "banana",
                "apple",
                "book"
        ]
}
```

$exists 判断字段是否存在

_id	name	favthing	age
62d2b015802200007c006eb0	sz	(Array) 4 Elements	(N/A)
62d2b015802200007c006eb1	sa	(Array) 3 Elements	(N/A)
62d2b015802200007c006eb2	sb	(Array) 3 Elements	(N/A)
62d2b015802200007c006eb3	sc	(Array) 3 Elements	(N/A)
62d2b11a802200007c006eb4	sc	(Array) 3 Elements	1000

```
db.favorite.find({"age":{$exists:true}}).forEach(printjson)#查询有这个字段的
```

```
{
        "_id" : ObjectId("62d2b11a802200007c006eb4"),
        "name" : "sc",
        "favthing" : [
                "banana",
                "apple",
                "computer"
        ],
        "age" : 1000
}
```

$ne 不等于

$in 包含

$nin 不包含

$size 数组元素个数

```
db.favorite.find({"favthing":{$size:4}}).forEach(printjson)
```

```
db.favorite.insert({"name":"sz","favthing":["banana","apple","book","computer"]})
db.favorite.insert({"name":"sa","favthing":["banana","apple","book"]})
db.favorite.insert({"name":"sb","favthing":["banana","book","computer"]})
db.favorite.insert({"name":"sc","favthing":["banana","apple","computer"]})
db.favorite.find({"favthing":{$all:["book","apple"]}}).forEach(printjson)
db.favorite.insert({"name":"sc","favthing":["banana","apple","computer"],"age":1000})
```

输出

```
{
        "_id" : ObjectId("62d2b015802200007c006eb0"),
        "name" : "sz",
        "favthing" : [
                "banana",
                "apple",
                "book",
                "computer"
        ]
}
```

count 查询记录条数

```
db.favorite.find({"favthing":{$size:4}}).count()
```

聚合操作

没找到

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"}}}])
$push	将值加入一个数组中，不会判断是否有重复的值。	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet	将值加入一个数组中，会判断是否有重复的值，若相同的值在数组中已经存在了，则不加入。	db.mycol.aggregate([{$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"}}}])

小结：测试代码

```
show dbs
db
use test
db.myCollection.insert({"name":"sz","age":123})
db.myCollection.insert({"name":"skk","age":123})
db.myCollection.insert({"name":"stt","age":123})
db.myCollection.insert({"name":"stu","age":124})
db.myCollection.insert({"name":"ssz","age":121})
db.myCollection.insert({"tittle":"斗破苍穹","author":"土豆"})
db.createCollection("newCollection")
show collections
db.myCollection.update({'name':'sz'},{$set:{'name':'sunzhong'}})
db.myCollection.find()
db.myCollection.find().limit(2)
db.myCollection.remove({"name":"skk"});
db.myCollection.remove({"age":123});
db.myCollection.find({"name":"sunzhong"})
db.myCollection.find().sort({"age":1})
db.myCollection.findOne()
father={"name":"sqz"}
mother={"name":"yxx"}
db.myCollection.save(father)
db.myCollection.save(father)
db.myCollection.save(mother)
db.myCollection.find()
db.myCollection.find({"age":{$gt:123}})
db.favorite.insert({"name":"sz","favthing":
["banana","apple","book","computer"]})
db.favorite.insert({"name":"sa","favthing":["banana","apple","book"]})
db.favorite.insert({"name":"sb","favthing":["banana","book","computer"]})
```

```
db.favorite.insert({"name":"sc","favthing":["banana","apple","computer"]})
db.favorite.find({"favthing":{$all:["book","apple"]}}).forEach(printjson)
db.favorite.insert({"name":"sc","favthing":
["banana","apple","computer"],"age":1000})
db.favorite.find()
db.favorite.find({"age":{$exists:true}}).forEach(printjson)
// $size 数组元素个数
db.favorite.find({"favthing":{$size:4}}).forEach(printjson)
db.favorite.find({"favthing":{$size:4}}).count()
```

java操作mongodb

插入文档

```
import java.util.ArrayList;
import java.util.List;
import org.bson.Document;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
         // 连接到 mongodb 服务
         MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

         // 连接到数据库
         MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
         System.out.println("Connect to database successfully");

         MongoCollection<Document> collection =
mongoDatabase.getCollection("test");
         //mongoDatabase.createCollection("test");  创建集合
         System.out.println("集合 test 选择成功");
         //插入文档
         /**
         * 1. 创建文档 org.bson.Document 参数为key-value的格式
         * 2. 创建文档集合List<Document>
         * 3. 将文档集合插入数据库集合中 mongoCollection.insertMany(List<Document>)
插入单个文档可以用 mongoCollection.insertOne(Document)
         * */
         Document document = new Document("title", "MongoDB").
         append("description", "database").
         append("likes", 100).
         append("by", "Fly");
         List<Document> documents = new ArrayList<Document>();
         documents.add(document);
         collection.insertMany(documents);
         System.out.println("文档插入成功");
      }catch(Exception e){
```

```
            System.err.println( e.getClass().getName() + ": " + e.getMessage() );
        }
    }
}
```

检索文档

```
//检索所有文档
/**
         * 1. 获取迭代器FindIterable<Document>
         * 2. 获取游标MongoCursor<Document>
         * 3. 通过游标遍历检索出的文档集合
         * */
FindIterable<Document> findIterable = collection.find();
MongoCursor<Document> mongoCursor = findIterable.iterator();
while(mongoCursor.hasNext()){
    System.out.println(mongoCursor.next());
}
```

更新文档

```
//更新文档   将文档中likes=100的文档修改为likes=200
collection.updateMany(Filters.eq("likes", 100), new Document("$set",new
Document("likes",200)));
//检索查看结果
FindIterable<Document> findIterable = collection.find();
MongoCursor<Document> mongoCursor = findIterable.iterator();
while(mongoCursor.hasNext()){
    System.out.println(mongoCursor.next());
}
```

删除文档

```
//删除符合条件的第一个文档
collection.deleteOne(Filters.eq("likes", 200));
//删除所有符合条件的文档
collection.deleteMany (Filters.eq("likes", 200));
```