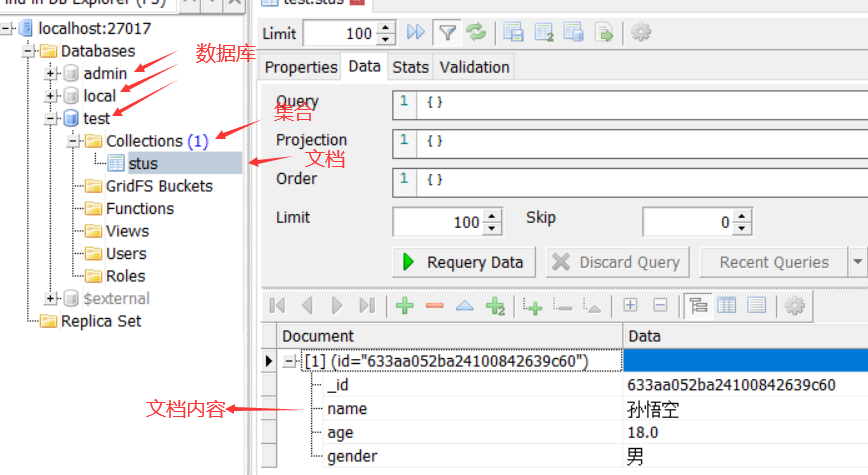
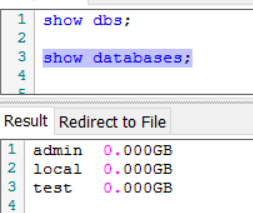
MongoDb学习笔记

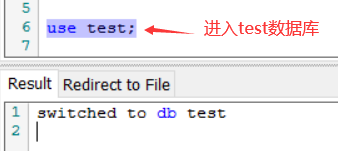
1. 基本介绍
2. MongoDb是非关系型的 文档数据库
3. 在经过服务器和客户端安装后（参照同目录的安装教程），我们的基本环境已经配置完成，下面讲常用的操作指令
4. MongoDB的基本结构
5. 和MySQL相似，MongoDB也大致分为三层，依次是：数据库，集合和文档（文档指的并不是文件，而是key-val形式的json结构），对应于MySQL中的数据库，table和数据



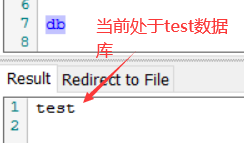
1. 操作指令
2. 基本操作指令
3. mongoDB命令行操作，可参照官网：<https://www.mongodb.com/docs/manual/crud/> 下面介绍以下基本操作
4. 显示所有数据库：show dbs/show databases



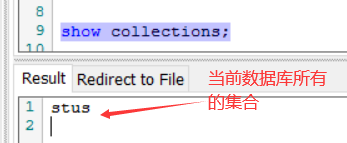
1. 进入数据库：use 数据库名



1. 查询当前所在数据库：db



1. 查询当前数据库所有的集合：show collections;



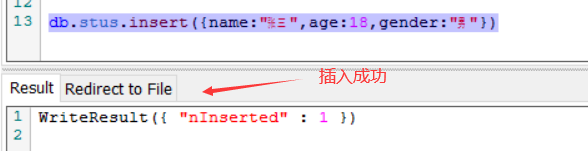
1. 增加操作
2. 向当前数据库的集合中插入一个或多个文档（文档指的不是文件，而是一个json数据）：

db.集合名称.insert(文档)

db.集合名称.insert([文档1],[文档2],[文档3])

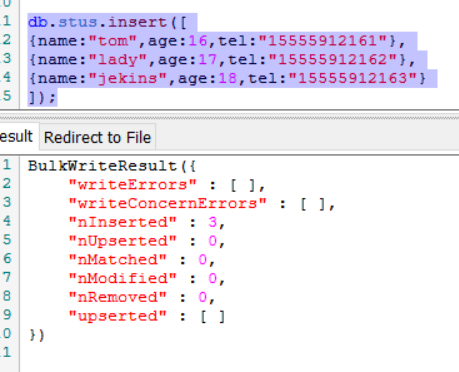
* 如要向stus集合中插入一个json结构:{name:"张三",age:18,gender:"男"}

db.stus.insert({name:"张三",age:18,gender:"男"})

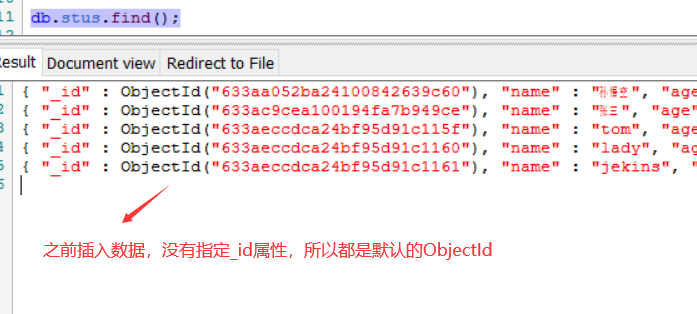


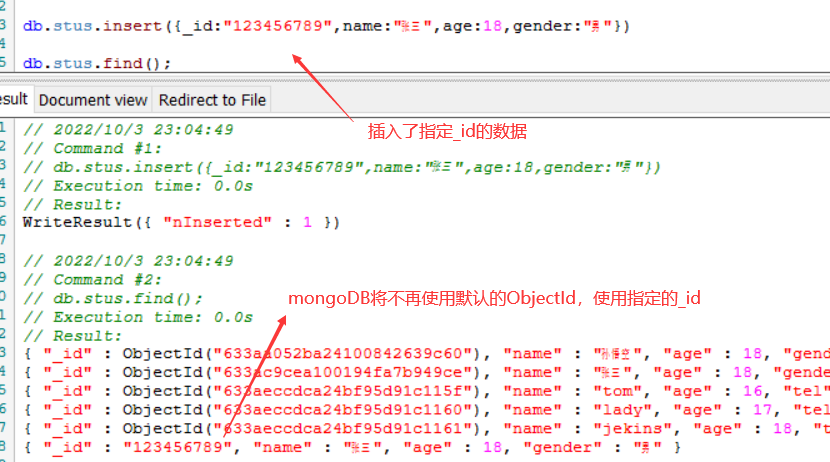
* 如要向stus集合中插入多个json结构：

db.stus.insert([{name:"tom",age:16,tel:"15555912161"},{name:"lady",age:17,tel:"15555912162"},{name:"jekins",age:18,tel:"15555912163"}]);



* 注意，当我们插入数据时，如果不指定\_id，mongoDB会为我们提供默认的ObjectId作为属性\_id的值,当我们提供了\_id，mongoDB将不会再自动生成。\_id作为mongoDb文档的唯一标识，不会也不能重复；





1. 查询操作
2. 查询操作的语法是：

db.集合名称.find() 查询集合中所有的文档,返回数组

db.集合名称.find({属性1:val1,属性2:val2,……,属性n:valn}) 查询集合中符合条件的文档，返回数组

db.集合名称.find()[n] 获取返回数组中的第N个元素

db.集合名称.find({属性1:val1,属性2:val2,……,属性n:valn})[N] 获取返回数据中的第N个元素

db.集合名称.findOne() 查询默认排序的第一个元素

db.集合名称.findOne({属性1:val1,属性2:val2,……,属性n:valn }) 查询符合条件默认排序的第一个元素

db.集合名称.findOne().name 查询默认排序的第一个元素的name属性值

db.集合名称.findOne({属性1:val1,属性2:val2,……,属性n:valn }).name 查询符合条件默认排序的第一个元素name属性值

db.集合名称.find().count() 查询集合中文档的数量

db.集合名称.find({属性1:val1,属性2:val2,……,属性n:valn}).count() 查询符合条件的文档的数量

1. 例子①

db.stus.find();

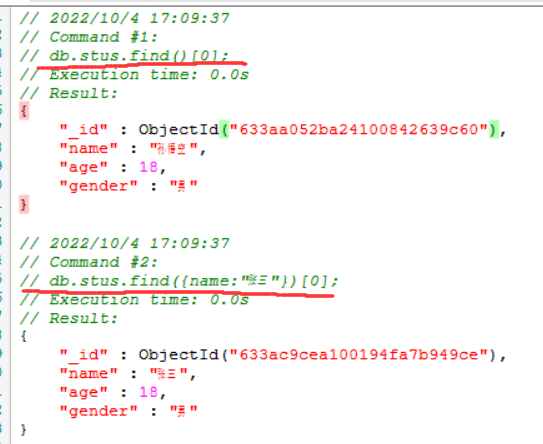
db.stus.find({name:"张三"});



1. 例子②：

db.stus.find()[0];

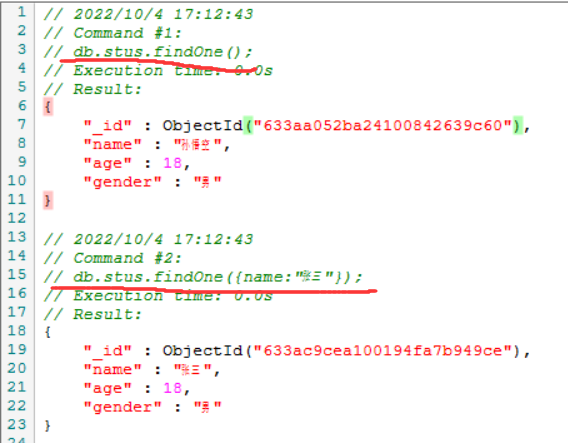
db.stus.find({name:"张三"})[0];



1. 例子③：

db.stus.findOne();

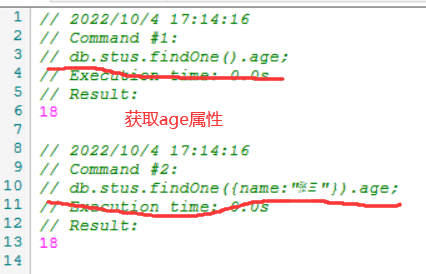
db.stus.findOne({name:"张三"});



1. 例子④：

db.stus.findOne().age;

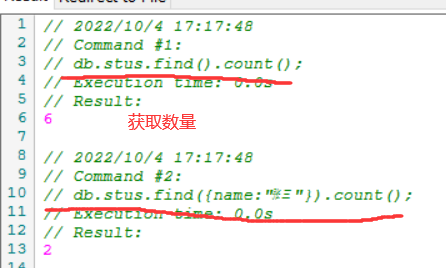
db.stus.findOne({name:"张三"}).age;



1. 例子⑤：

db.stus.find().count();

db.stus.find({name:"张三"}).count();



1. 修改操作
2. Update的基本语法是

db.collection.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>,

collation: <document>,

arrayFilters: [ <filterdocument1>, ... ],

hint: <document|string>, // Added in MongoDB 4.2

let: <document> // Added in MongoDB 5.0

}

)

db.collection.updateOne(

<filter>,

<update>,

{

upsert: <boolean>,

writeConcern: <document>,

collation: <document>,

arrayFilters: [ <filterdocument1>, ... ],

hint: <document|string> // Available starting in MongoDB 4.2.1

}

)

db.collection.updateMany(

<filter>,

<update>,

{

upsert: <boolean>,

writeConcern: <document>,

collation: <document>,

arrayFilters: [ <filterdocument1>, ... ],

hint: <document|string> // Available starting in MongoDB 4.2.1

}

)

1. 相关参数说明

|  |  |  |
| --- | --- | --- |
| 范围 | 类型 | 描述 |
| query | 文档 | 使用查询运算符指定条件。和find()方法，一样的查询条件 |
| update | 文档或聚合函数 | 1. 如果是文档类型的话，那么就是将查询到的集合进行替换 2. 如果是聚合函数的话，那么就是对查询到的集合进行聚合操作，常见的聚合函数有：  * $addFields或别名$set * $project或别名$unset * $replaceRoot * $replaceWith |
| upsert | 布尔值 | 可选的。如果设置为true，则在没有文档符合查询条件时会创建一个新文档。默认值为 false，在找不到匹配项时，不插入文档 |
| multi | 布尔值 | 可选。如果设置为true，则更新满足query条件的多个文档。如果设置为false，则更新一个文档。默认值为false |
| writeConcern | 文档 | 可选, MongoDB写入安全机制，如果在事务中运行，请不要为操作明确设置writeConcern |
| collation | 文档 | 可选, 允许用户为字符串比较指定特定于语言的规则，例如字母大写和重音符号的规则 |
| arrayFilters | 文档数组 | 可选。过滤文档数组，在更新数组字段时，用于确定要进行更新的数组元素 |

1. 一般情况下，可选内容是用不到的，常用的语句如下：

*// ① 替换 将name 是lady的数据 替换为 文档{age:100}*

db.stus.update({name:"lady"},{age:100});

*//② 修改 将符合声明id的文档的name，age属性 替换为ladygaga和61*

db.stus.update({\_id:ObjectId("6342ddd1a100194fa7b949cf")},{**$set**:{name:"ladygaga",age:61}})

*//③ 删除文档的属性 将符合声明id的文档的adress属性删除*

db.stus.update({\_id:"123456789"},{**$unset**:{adress:"ss"}});

*//④ 同时修改多个符合条件的文档*

db.stus.updateMany()

*//⑤ 只修改符合条件的一个文档*

db.stus.updateOne()

*//⑥ 替换 一个*

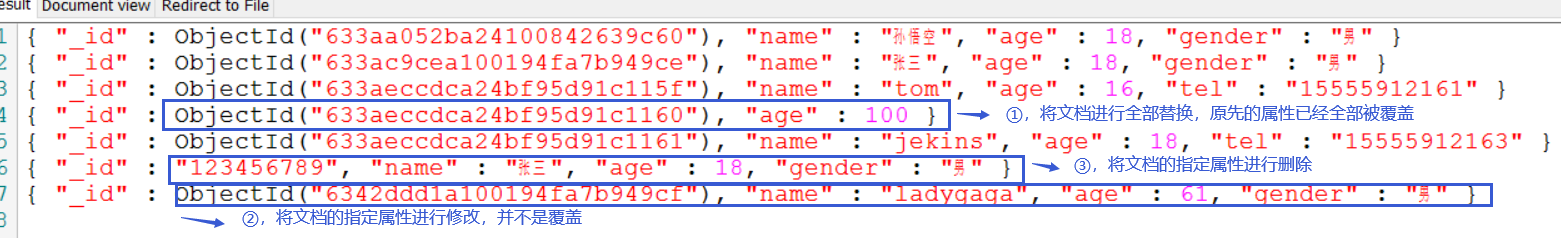
db.stus.replaceOne()

*//update 默认修改查询到的第一个 如果要修改多个 就需要更改参数*

*//⑦ 同时修改多个 并且 不存在就插入*

db.stus.update({name:"张三"},{**$set**:{age:65}},{multi:true,upsert:true});

然后执行语句的结果如下：



1. 一些注意事项

* update 默认只修改查找到的第一个文档，如果想修改查询到的所有文档，可以添加参数multi，或者使用updateMany
* $set 用于替换查找到文档的指定属性 $unset用于删除查找到文档的指定属性

1. 删除操作
2. 删除操作的基本语法及用法如下：

① 删除集合中 符合条件的所有文档

db.集合名.remove({属性1:val1,属性2:val2,……,属性n:valn});

db.集合名.removeMany({属性1:val1,属性2:val2,……,属性n:valn});

②删除集合中 符合条件的 第一个文档

db.集合名.remove({属性1:val1,属性2:val2,……,属性n:valn},true);

db.集合名.removeOne({属性1:val1,属性2:val2,……,属性n:valn});

③ 删除集合

db.集合名.drop();

④删除数据库

db. dropDatabase();

*// 删除*

*// 默认情况下 删除符合条件的所有文档 如果只删除一个 那么需要在后面加参数true*

db.stus.remove({\_id:ObjectId("6342ddd1a100194fa7b949cf")});

db.stus.remove({"age":18},true);

*//便捷的方法//只删一个*

db.stus.removeOne({"name":"张三","age":18});

*//删除多个*

db.stus.removeMany({"name":"张三","age":18});

*//删除所有*

db.stus.remove({});

*//删除集合*

db.stus.drop();

*//删除数据库*

db.dropDatabase();

1. 数据关联关系（存储和查询）
2. 数据关联关系 一半分为三种：一对一，一对多和多对多，但是记录方式 大致有以下两种记录方式：

* 当关联的信息比较简单时，直接将关联的信息记录在关联信息中即可，如：

小明和小红，他们分别有两个一对多的关系，都对应不同的老师和课程

db.students.insert(

[{

"name": "小明",

"age": 12,

"teachers": ["孔子", "孟子", "老子"],

"classes": {

"数学": "五年级上册",

"语文": "四年级下册",

"英语": "三年级上册"

}

}, {

"name": "小红",

"age": 18,

"teachers": ["孙武", "孙膑", "司马迁"],

"classes": {

"三十六计": "第五卷",

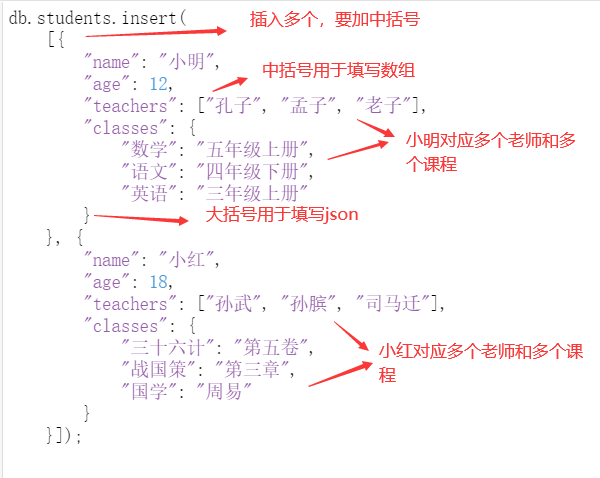
"战国策": "第三章",

"国学": "周易"

}

}]);

注意在mongoDB中，大括号中填写的是键值对，中括号中填写的是数组



这样，对于简单的对应关系，就可以使用直接填写的形式来说明

* 当关联的信息比较复杂 ，或者扩展性比较强时，可以使用id的形式进行关联，如将上面的信息改为：

将关联项进行入库：

db. students.insert([{

"name": "孔子",

"age": 58,

"address": "山东曲阜",

"tel": "123456780"

}, {

"name": "孟子",

"age": 30,

"address": "西藏",

"tel": "123456781"

}, {

"name": "老子",

"age": 48,

"address": "武当山",

"tel": "123456782"

}, {

"name": "孙武",

"age": 56,

"address": "新疆",

"tel": "123456783"

}, {

"name": "孙膑",

"age": 89,

"address": "云南",

"tel": "123456784"

}, {

"name": "司马迁",

"age": 12,

"address": "海南",

"tel": "123456785"

}]);

db. students.insert([{

"科目名称": "数学",

"章节": "五年级上册"

}, {

"科目名称": "语文",

"章节": "四年级下册"

}, {

"科目名称": "英语",

"章节": "三年级上册"

}, {

"科目名称": "三十六计",

"章节": "第五卷"

}, {

"科目名称": "战国策",

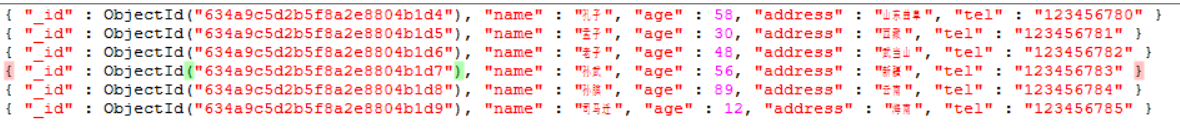
"章节": "第三章"

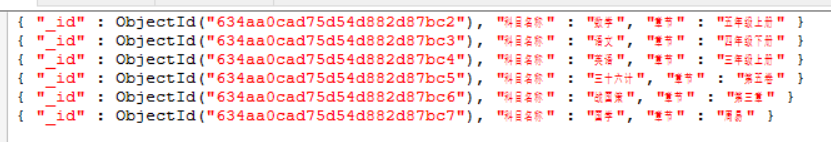
}, {

"科目名称": "国学",

"章节": "周易"

}]);





然后将关联信息 改为 关联 id

db.students.insert(

[{

"name": "小明",

"age": 12,

"teachers": [

ObjectId("634a9c5d2b5f8a2e8804b1d4"),

ObjectId("634a9c5d2b5f8a2e8804b1d5"),

ObjectId("634a9c5d2b5f8a2e8804b1d6")

],

"classes": [

ObjectId("634aa0cad75d54d882d87bc2"),

ObjectId("634aa0cad75d54d882d87bc3"),

ObjectId("634aa0cad75d54d882d87bc4")

]

}, {

"name": "小红",

"age": 18,

"teachers": [

ObjectId("634a9c5d2b5f8a2e8804b1d7"),

ObjectId("634a9c5d2b5f8a2e8804b1d8"),

ObjectId("634a9c5d2b5f8a2e8804b1d9")

],

"classes": [

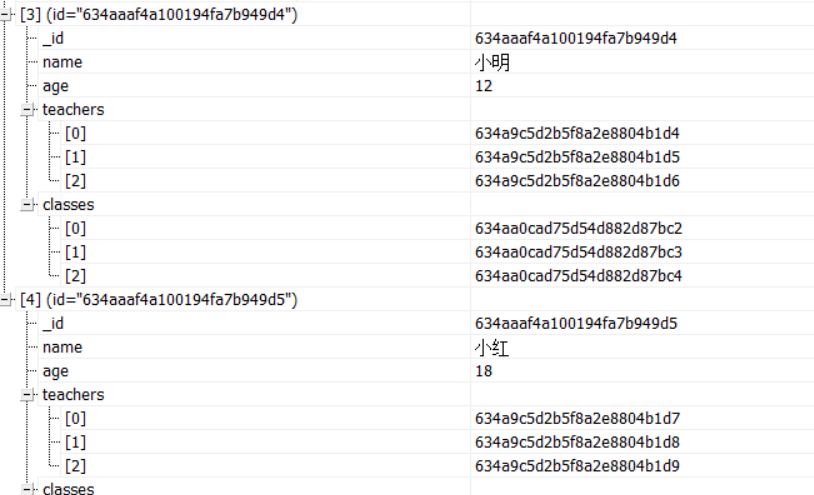
ObjectId("634aa0cad75d54d882d87bc5"),

ObjectId("634aa0cad75d54d882d87bc6"),

ObjectId("634aa0cad75d54d882d87bc7")

]

}]);



* 关联查询

假如查询小红第一个老师的信息，应该这样查（mongodb支持简单的变量定义）

//创建变量 teacs

var teacs = db.students.findOne({

"name": "小红"

}).teachers;

//输出数组的第一个元素

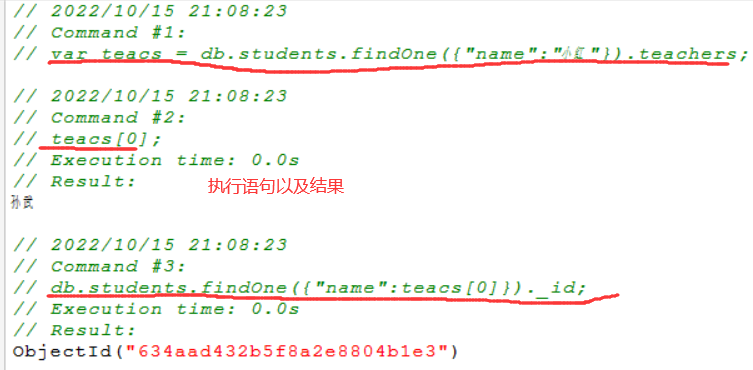
teacs[0];

//然后条件查询

db.students.findOne({

"name": teacs[0]

}).\_id;



1. 排序和投影
2. Mongodb的排序默认是按照id进行升序排序的，而id是由时间戳生成的，所以mongodb默认是按照时间进行升序排序的
3. Mongodb提供sort函数来来支持自定义排序，其中1表示升序，-1表示降序，示例如下：

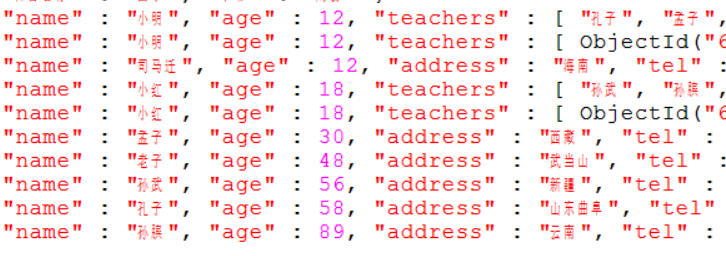
如在students表中，按照age进行升序，name进行降序排列

db.students.find().sort({

"age": 1,

"name": -1

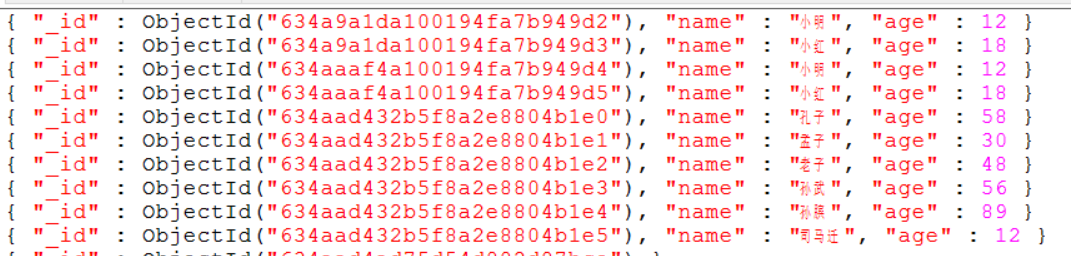
});



1. 当我们查询数据时，尤其是当文档属性特别多的时候，就需要有选择地进行显示，否则影响阅读，find后有个参数就为我们提供了相应的功能，其中1表示显示，0表示隐藏,\_id列默认是显示的

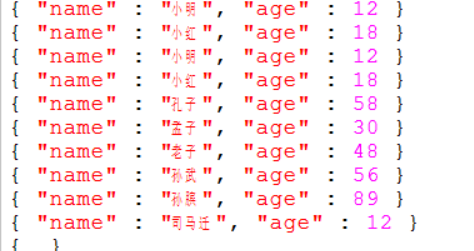
如：我们在students表中只显示name和age属性，其他的都隐藏

db.students.find({},{"name":1,"age":1});

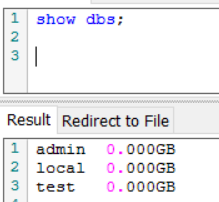


可见，\_id列如果不隐藏，默认是显示的，所以需要主动隐藏\_id

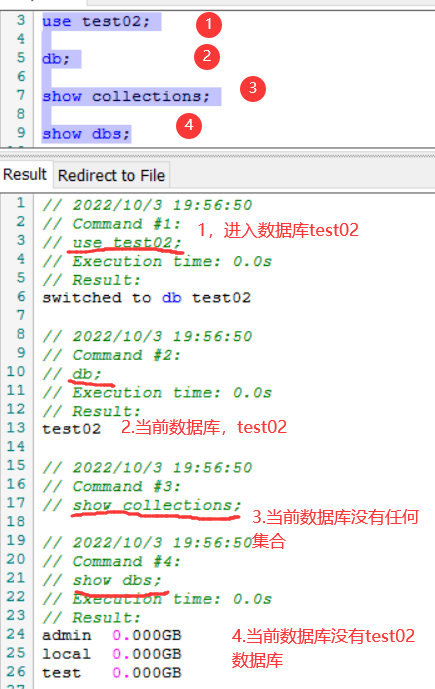
db.students.find({},{"name":1,"age":1,\_id:0});



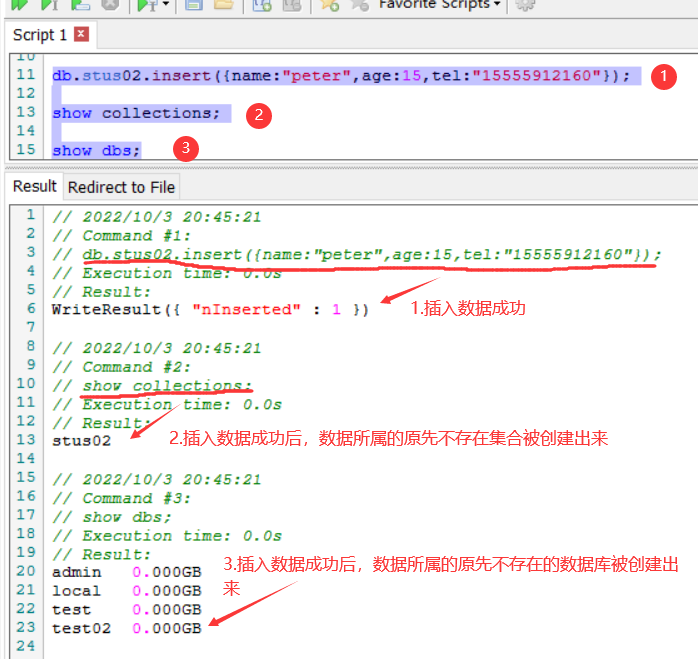
1. MongoDB的特别之处
2. 同其他数据库不同的是，mongoDB可以在原先不存在的数据库中，不存在的集合中插入文档，当文档创建时，它所属的数据库以及集合都会自动创建
3. 如查询当前所有数据库



1. 进入一个不存在的数据库，查询当前数据库，查询当前数据库的集合，并查询数据库列表



1. 在这个虚构的数据库下，虚构的集合中，创建一个文档（json结构），并查询数据库列表，以及本数据库的集合列表



1. 可以看到，原先不存在的数据库和集合，在被塞入文档后，就被自动创建了，当然也可以手动去创建好数据库和集合，然后往里面塞文档