two

复习：

1. 非关系型数据库和关系型数据库的区别

* 不是以关系模型构建的，机构比较自由
* nosql不保障数据的一致性
* nosql弥补了关系型数据库的一些不足，比如高并发，海量数据处理
* nosql技术好在不断地快速发展，功能没有没有关系型数据库成熟强大

MongoDB

创建数据库：use database

删除数据库：db.dropDatabase()

创建集合：db.createCollection()

db.collection.insert()

删除集合： db.collection.drop()

集合重命名：db.collection.renameCollection(‘’)

查看数据库：show dbs

查看集合：show tables

插入数据文档：insert() save（）

查找操作：

MySQL：select \* from表名where条件

MongoDB：db.collection.find(query，field)

查找所有内容

db.collection.find()---》select \* from表名

find（query，field）

功能：查找文档

参数：query查找条件，相当于where子句

field查找的域

返回值：查找的所有文档

query：以键值对方式传递参数，如果是空｛｝表示查找所有内容

e.g. 查找所有性别为w的文档

db.class0.find({sex:'w'})

field ：以键值对的方式给出要查找（不查找）的域

以域名为键。以0，1为值分别表示不查找和查找

\*如果某一个或者多个域设置为0表示这些域不查找，其他域均查找

\*如果某一个或多个域设置为1，表示这些域查找，其他域均不查找

\*\_id除非设置为0 ，否则均会查找

\*除\_id域其他的域不能有的设置1有的设置0

e.g.查找结构只有name的域：

db.class0.find({sex:’w’}，{\_id:0，name:1})

findOne（query，field）

功能：查找第一条符合条件的文档

参数：同find

返回值：返回查找到的文档

e.g.

db.class0.findOne({sex:'m'},{\_id:0})

query更多的筛选用法：

操作 符：使用$符号著名一个特殊字符串，表示一定的含义，比如$lt表示小于

比较操作符

$eq d等于

> db.class0.find({age:{$eq:18}},{\_id:0})

$lt 小于

> db.class0.find({age:{$lt:19}},{\_id:0})

\*在MongoDB中字符串可以比较大小

> db.class0.find({name:{$lt:'jame'}},{\_id:0})

> db.class0.find({name:{$lt:'Alex'}},{\_id:0})

$lte小于等于

e.g.年龄小于等于19的

> db.class0.find({age:{$lte:19}},{\_id:0})

$gt大于

大于17小于19的人

> db.class0.find({age:{$gt:17,$lt:19}},{\_id:0})

\*在mongodb中所有的｛｝[]中都可以都可以写多个条件。但是根据参数的不同表达的意思不一样

$gte大于等于

大于等于17小于等于19的

> db.class0.find({age:{$gte:17,$lte:19}},{\_id:0})

$ne不等于

性别不等于；w；

> db.class0.find({sex:{$ne:'w'}},{\_id:0})

\*使用ne查找也会找到该域不存在的文档

$in包含

e.g.找到年龄为【16，19，20】

db.class0.find({age:{$in:[16,19,20]}},{\_id:0})

$nin不包含

不包含【17，19，20】

> db.class0.find({age:{$nin:[17,19,20]}},{\_id:0})

逻辑操作符：

$and 逻辑与

1、在query如果写多个条件默认即为and关系

> db.class0.find({age:{$lt:19},sex:'w'},{\_id:0})

2、 > db.class0.find({$and:[{age:{$lt:23}},{sex:'w'}]},{\_id:0})

> db.class0.find({$and:[{age:{$gte:19}},{sex:'m'}]},{\_id:0})

$or逻辑或

> db.class0.find({$or:[{age:{$lt:17}},{age:{$gt:22}}]},{\_id:0})

{ "name" : "xiao", "age" : 23, "sex" : "m" }

{ "name" : "john", "age" : 15, "sex" : "m" }

$not逻辑非

> db.class0.find({sex:{$not:{$eq:'m'}}},{\_id:0})

> db.class0.find({age:{$not:{$lt:20}}},{\_id:0})

$nor (not (a or b)) ===> (not a)and(not b)

性别不是m且年龄不小于18

> db.class0.find({$nor:[{sex:'w'},{age:{$lt:18}}]},{\_id:0})

逻辑条件混合

（年龄大于17并且为男生）或者姓名为Abby

> db.class0.find({$or:[{$and:[{age:{$gt:17}},{sex:'m'}]},{name:'Abby'}]},{\_id:0})

(年级不大于18或者为女性)并且姓名大于Lucy

> db.class0.find({$and:[{$or:[{age:{$lte:18}},{sex:'w'}]},{name:{$gt:'LucY'}}]},{\_id:0})

数组：array

[1,2,3,4]

\*有序的数据集合

\*mongo中数组也可以有多重数据元素混合

查找数组中包含某一条件的元素

e.g.只要score数组中包含小于60的元素即可查询过滤

> db.class1.find({score:{$lt:60}},{\_id:0})

$all

查找数组中包含多项的文档：

> db.class1.find({score:{$all:[49,67]}},{\_id:0})

$size通过数组元素个数查找

e.g.查找score中包含俩个元素的文档

db.class1.find({score:{$size:2}},{\_id:0})

$slice

显示数组中的指定项

e.g． 显示数组的前两项

> db.class1.find({},{\_id:0,score:{$slice:2}})

e.g． 跳过第一个显示后面两个

> db.class1.find({},{\_id:0,score:{$slice:[1,2]}})

其他常用查找操作符

$exists

通过某个域是否存在筛选（true存在。false不存在）

e.g.查找不存在sex域的文档

> db.class1.find({sex:{$exists:false}},{\_id:0})

$mod余数查找

e.g.找出年龄为单数的文档

> db.class1.find({age:{$mod:[2,1]}},{\_id:0})

$type找出指定数据类型的文档

e.g． 查找name域值类型为2的文档

> db.class1.find({name:{$type:2}},{\_id:0})

查找结果的操作函数

db.collection.distinct(filed)

功能：查看某个域的值范围

e.g.获取某个域的值，去重

db.class0.distinct(‘age’)

pretty（）

功能：格式化显示查询结果

e.g.

> db.class0.find().pretty()

limit（n）

功能：显示前n条结果

e.g.显示查询结果 的前三条

> db.class0.find({},{\_id:0}).limit(3)

skip（n）

功能：跳过前n条显示后面的结果

e.g.跳过前五条结果，显示后面的查询结果

> db.class0.find({},{\_id:0}).skip(5)

count（）

功能：统计查询结果数量

\*在统计数量时要给出一定query条件

e.g. 统计性别为w的文档个数

db.class0. find({sex:'w'},{\_id:0}).count()

sort（｛field：1/-1｝）

功能：对查找结果进行排序

参数：以键值对表示是按照哪一个field排序

1表示升序，-1降序

e.g.查找结果按照降序排序

> db.class0.find({},{\_id:0}).sort({age:-1})

复合排序：



> db.class0.find({},{\_id:0}).sort({age:1,name:-1})

函数的连续调用

当返回值是文档的时候，可以连续调用函数

e.g.

> db.class0.find({},{\_id:0}).sort({age:1,name:-1}).limit(3)

删除文档

MySQL ：delete from表名where条件

MongoDB： db.collection.remove(query，justOne)

remove（query，justOne）

功能：删除文档

参数：query用法同find

justOne布尔值 默认为false表示删除所有符合条件的文档

设置为True则表示只删除一条

e.g.删除所有不存在sex域的文档

db.class1.remove({sex:{$exists:false}})

e.g.删除第一条性别为w的文档

db.class1.remove({sex:'w'})

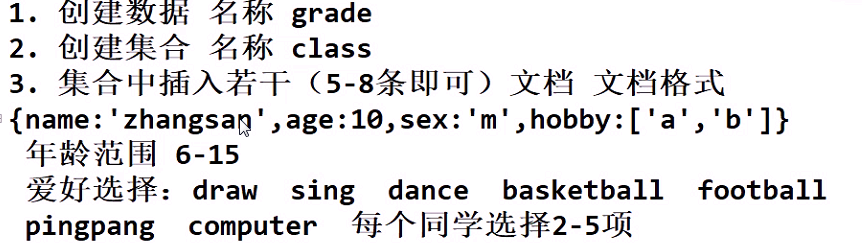
删除集合中所有文档

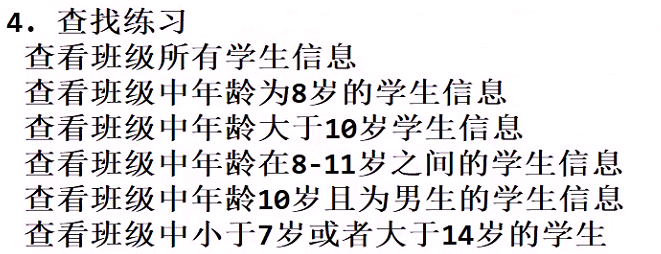
e.g.

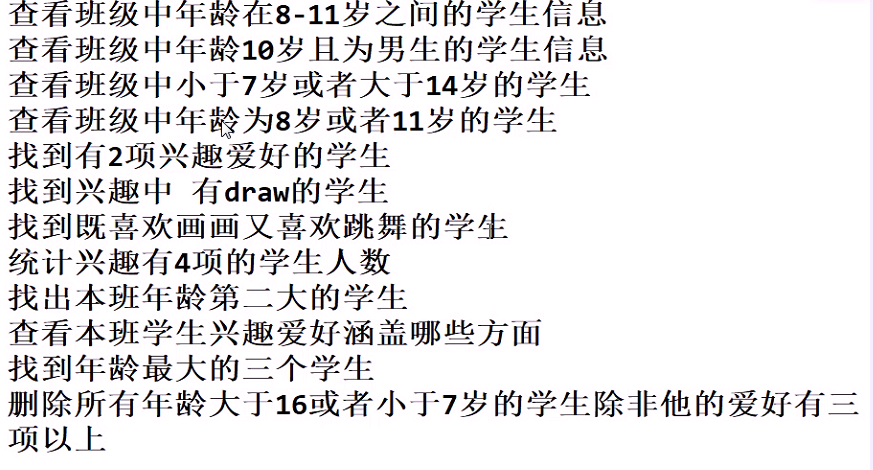
db.class1.remove()

删除集合中ID类型不是objectID的文档

> db.class0.remove({\_id:{$not:{$type:7}}})





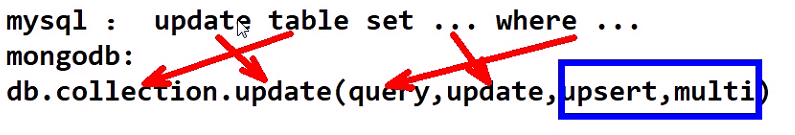


修改文档

MySQL：update table set 。。。where..

mongodb

db.collection.update(query,update,upsert,multi)



db.collection.update(query,update,upsert,multi)

功能：修改文档

参数： query筛选条件 用法同find

update要修改成什么内容，通常配合修改操作符（修改器）使用

upsert布尔值，默认是false如果query没有筛选到文档这不做任何操作

如果设置为TURE，则如果query没有筛选到匹配文档则根据query和update内容插入新的文档

multi布尔值 默认false表示如果有多条符合条件则只修改第一条

如果设置为true则表示修改所有符合条件的文档

e.g.将tom的年龄改为18

db.class0.update({name:'Tom'},{$set:{age:18}})

e.g.如果有name=jarry的文档则修改，如果没有则根据query和update增加新的一条

db.class0.update({name:'Jarry'},{$set:{age:15}},true)

e.g/修改所有年龄小于17的为18

> db.class0.update({age:{$lt:17}},{$set:{age:18}},false,true)

