from pymongo import MongoClient

创建连接到mongodb数据库

>>>client = MongoClient('localhost', 27017) #连接外部数据库

>>>client = MongoClient() #默认连接本地数据库

创建“test-database”数据库

>>>db = client['test\_database']

#查看下面的数据库

client.database\_names()

#删除数据库

client.drop\_database()

从数据库中选择集合

>>>collection = db['test\_collection']

#查看下面的集合

db.collection\_names()

文件（JSON风格）

post = {**'标题'**: self.title,  
 **'主题页面'**: self.url,  
}

**$set：替换某个field的值** { $set: { <field1>: <value1>, ... } }

**$unset：取消某个field** { $unset: { <field1>: "", ... } }

**$addToSet：插入值至array**

**$inc：+1**

**$mul：乘**

**$rename：重命名** {$rename: { <field1>: <newName1>, ... } }

**$min：指定值比存在值小则更新**

**$max**

**$currentDate：将某field值设定为日期（”$currentDate”:{lastModified:true}）**

**$lt小于{'age': {'$lt': 20}}**

**$gt大于{'age': {'$gt': 20}}**

**$lte小于等于{'age': {'$lte': 20}}**

**$gte大于等于{'age': {'$gte': 20}}**

**$ne不等于{'age': {'$ne': 20}}**

**$in在范围内{'age': {'$in': [20, 23]}}**

**$nin不在范围内{'age': {'$nin': [20, 23]}}**

**$regex匹配正则{'name': {'$regex': '^M.\*'}}name以M开头**

**$exists属性是否存在{'name': {'$exists': True}}name属性存在**

**$type类型判断{'age': {'$type': 'int'}}age的类型为int**

**$mod数字模操作{'age': {'$mod': [5, 0]}}年龄模5余0**

**$text文本查询{'$text': {'$search': 'Mike'}}**

**$where高级条件查询{'$where': 'obj.fans\_count == obj.follows\_count'}**

#用于替代”$lt”和”$gt”：collection.find({“where”:”this.age>7 || this.age<3”})

**插入**

**插入单条数据：insert\_one**(document)

collection.insert\_one({“x”:1})

**插入多条数据**：**insert\_many**(documents)

#documents为iterable

collection.insert\_many([{“x”:1} for I in range(2)])

**查询**

**查询单条数据：find\_one**()

collection.find\_one({'主题页面': href})

collection.find\_one({“**$or**”: [{key1: value1}, {key2:value2}]})

**查询多条数据：find**()

collection.find({'age': {'$gt': 20}})

#可利用此查询整个collection

for item in collection.find(): #查看所有数据  
 print(item)

**其它：总是返回对应内容**

**find\_one\_and\_delete**(filter, projection=None, sort=None)

collection.find\_one\_and\_delete({“x”:1})

若有多个匹配，可排序

collection. find\_one\_and\_delete({'x': 1}, sort=[('\_id', pymongo.DESCENDING)])

**find\_one\_and\_replace**(filter, replacement, projection=None, sort=None)

collection. find\_one\_and\_replace({'x': 1}, {'y': 1})

**find\_one\_and\_update**(filter*,*update*,*projection=None*,*sort=None)

collection. find\_one\_and\_update({'\_id': 665},

{'$inc': {'count': 1}, '$set': {'done': True}})

collection. find\_one\_and\_update({'\_id': 'userid'},{'$inc': {'seq': 1}},

projection={'seq': True, '\_id': False}

**替换**

**替换单条数据：replace\_one**(filter, replacement, upsert=False)

#upsert=True ：若匹配文档不存在，则插入新的

collection.replace\_one({“x”:1},{“y”:1})

**删除文档**

**删除单条数据**

**delete\_one**()

**删除多条数据**

**delete\_many**()

**删除集合**

**collection.drop()**

**删除集合里所有记录  
collection.remove()**

**更新**

**更新单条数据：update\_one(filter, update, upsert=False)**

collection.update\_one({'x': 1}, {'$inc': {'x': 3}})

**更新多条数据：update\_many(filter, update, upsert=False)**

collection.update\_many({'x': 1}, {'$inc': {'x': 3}})

**计数：count**()

collection.find({"author": "Mike"}).count()

**排序：sort()**

1为升序排列, -1为降序排列；或者使用pymongo.ASCENDING

collection.find().sort({"likes":-1})

collection.find().sort("likes, pymongo.ASCENDING)

**偏倚**

**>>>skip()：指定忽略的个数**

collection.find().sort('name', pymongo.ASCENDING).skip(2)

#skip(2)为忽略前2个元素，得到第三个及以后的元素

**>>>limit()：指定选取的个数**

collection.find().sort('name', pymongo.ASCENDING).skip(2).limit(2)

**对数组进行操作**

**$all：判断数组属性是否包含全部条件**

collection.find({“data”:{“$all”:[2,3,4]}})

**$size：匹配数组属性元素数量**

collection.find({“data”:{“$size”:3}})

**$type：判断属性**

**collection.find({“t”:{“$type”:1}})**

#1：数字，2：字符串，4：array，8：boolean

**$each：添加多个元素**

collection.update({“name”:”user2”,

{“addToSet”:{“data”:{“$each”:[1,2,3,4]}}}})

**$push, $pushAll：向数组添加元素**

**而$addToSet仅在该元素不存在时才添加**

**$pop：移除数组属性的元素(数组下标)**

collection.update({“name”:”user2”,{“$pop”:{“data”:1}})

#1：移除最后一个元素; -1：移除第一个元素

**$pull：按值移除**

collection.update({“name”:”user2”,{“$pull”:{“data”:2}})

**$pullAll：移除所有符合提交的元素**

collection.update({“name”:”user2”,{“$pullAll”:{“data”:[3,5,6]}})

**索引：createIndex()**

#能够极大的提高查询的效率,升序降序同sort（）

## collection.createIndex({“title”:1}) #单索引

## collection.createIndex({“title”:1,“description”:-1}) #多索引

## 寻找key的特异值：distinct(*key*, *filter=None*, *\*\*kwargs*)