MongoDB

- 讲师：pansir

nosql，非关系型数据库，官方使用文档：https://docs.mongodb.com/manual/

基本操作

- `show dbs`：显示数据库列表
- `show collections`： 显示数据库集合，类似mysql的表
- `show users`：显示用户
- `use {db name}`：切换数据库，若没有该库，则创建
- `db.dropDatabase()`：删除当前使用的库
- `db.help()`：显示数据库命令
- `db.foo.help()`：显示集合操作命令帮助
- `db.foo.find().help()`：显示查询操作的命令帮助

数据库常用命令

1. Help查看命令提示：
    - db.help();
    - db.yourColl.help();
    - db.youColl.find().help();
2. 切换/创建数据库： use yourDB; 当创建一个集合(table)的时候会自动创建当前数据库
3. 查询所有数据库：show dbs;
4. 删除当前使用数据库： db.dropDatabase();
5. 从指定主机上克隆数据库： db.cloneDatabase(“127.0.0.1”); 将指定机器上的数据库的数据克隆到当前数据库
6. 从指定的机器上复制指定数据库数据到某个数据库： db.copyDatabase("mydb", "temp", "127.0.0.1");将本机的mydb的数据复制到temp数据库中
7. 修复当前数据库： db.repairDatabase();
8. 查看当前使用的数据库：
    - db.getName();
    - db; db和getName方法是一样的效果，都可以查询当前使用的数据库
9. 显示当前db状态：db.stats();
10. 当前db版本：db.version();
11. 查看当前db的链接机器地址：db.getMongo();

集合常用命令

1. 创建一个聚集集合（table）： db.createCollection(“collName”, {size: 20, capped: 5, max: 100});
2. 得到指定名称的聚集集合（table）：db.getCollection("account");
3. 得到当前db的所有聚集集合： db.getCollectionNames();
4. 显示当前db所有聚集索引的状态：db.printCollectionStats();

用户相关命令

1. 添加一个用户：
    - db.addUser("name");
    - db.addUser("userName", "pwd123", true); 添加用户、设置密码、是否只读
2. 数据库认证、安全模式：db.auth("userName", "123123");
3. 显示当前所有用户：show users;
4. 删除用户：db.removeUser("userName");

增删改

新增数据

- insert新增：db.box.insert({"name":"bird"})
- save新增：db.box.save({"name":"bird"})

更新数据

```
# 语法格式：
db.collection.update(
    <query>,  # 查询条件
    <update>,  # 要更新的字段
    {
        upsert: <bool>,  # 如果update字段不存在，是否插入，默认false，不插入
        multi: <bool>,  # 默认false，只找第一条数据，如果为true，就更新全部
        writeConcern: <document>,  # 抛出异常级别，不常用
        collation: <document>  # 指定排序规则，不常用
    }
)
```

参考案例：

- `db.box.update({"name": "bird"}, {$set: {"name": "birdgugu"}}, {upsert: true, multi: true})`
- `db.box.update({"name": "birdgugu"}, {$set: {"name": "gugu"}}, true, true)`

删除文档

```
# 语法格式：
db.collection.remove(
    <query>,  # 查询条件
    {
        justOne: <bool>,  # 默认true，只删除第一个
        writeConcern: <document>,  # 抛出异常级别，不常用
    }
)
```

参考案例：

- 删除所有文档：`db.box.remove({ })`
- 删除指定文档：`db.box.remove({"name": "hello"})`

查询命令

1、查询所有记录 db.userInfo.find(); 相当于：select* from userInfo; 默认每页显示20条记录，当显示不下的情况下，可以用it迭代命令查询下一页数据。注意：键入it命令不能带“；” 但是你可以设置每页显示数据的大小，用DBQuery.shellBatchSize= 50;这样每页就显示50条记录了。

2、查询去掉后的当前聚集集合中的某列的重复数据 db.userInfo.distinct("name"); 会过滤掉name中的相同数据 相当于：select distict name from userInfo;

3、查询age = 22的记录 db.userInfo.find({"age": 22}); 相当于： select * from userInfo where age = 22;

4、查询age > 22的记录 db.userInfo.find({age: {$gt: 22}}); 相当于：select * from userInfo where age >22;

5、查询age < 22的记录 db.userInfo.find({age: {$lt: 22}}); 相当于：select * from userInfo where age <22;

6、查询age >= 25的记录 db.userInfo.find({age: {$gte: 25}}); 相当于：select * from userInfo where age >= 25;

7、查询age <= 25的记录 db.userInfo.find({age: {$lte: 25}});

8、查询age >= 23 并且 age <= 26 db.userInfo.find({age: {$gte: 23,lte: 26}});

9、查询name中包含 mongo的数据 db.userInfo.find({name: /mongo/}); //相当于%% select * from userInfo where name like ‘%mongo%’;

10、查询name中以mongo开头的 db.userInfo.find({name: /^mongo/}); select * from userInfo where name like ‘mongo%’;

11、查询指定列name、age数据 db.userInfo.find({}, {name: 1, age: 1}); 相当于：select name, age from userInfo; 当然name也可以用true或false,当用ture的情况下河name:1效果一样，如果用false就是排除name，显示name以外的列信息。

12、查询指定列name、age数据, age > 25 db.userInfo.find({age: {$gt: 25}}, {name: 1, age: 1}); 相当于：select name, age from userInfo where age >25;

13、按照年龄排序 升序：db.userInfo.find().sort({age: 1}); 降序：db.userInfo.find().sort({age: -1});

14、查询name = zhangsan, age = 22的数据 db.userInfo.find({name: 'zhangsan', age: 22}); 相当于：select * from userInfo where name = ‘zhangsan’ and age = ‘22’;

15、查询前5条数据 db.userInfo.find().limit(5); 相当于：selecttop 5 * from userInfo;

16、查询10条以后的数据 db.userInfo.find().skip(10); 相当于：select * from userInfo where id not in (selecttop 10 * from userInfo);

17、查询在5-10之间的数据 db.userInfo.find().limit(10).skip(5); 可用于分页，limit是pageSize，skip是第几页*pageSize

18、or与 查询 db.userInfo.find({$or: [{age: 22}, {age: 25}]}); 相当于：select * from userInfo where age = 22 or age = 25;

19、查询第一条数据 db.userInfo.findOne(); 相当于：selecttop 1 * from userInfo; db.userInfo.find().limit(1);

20、查询某个结果集的记录条数 db.userInfo.find({age: {$gte: 25}}).count(); 相当于：select count(*) from userInfo where age >= 20;

21、按照某列进行排序 db.userInfo.find({sex: {$exists: true}}).count(); 相当于：select count(sex) from userInfo;

Python操作mongo

安装库：`pip install pymongo`

连接方式

```
from pymongo import MongoClient
# 1. 参数形式
client1 = MongoClient(host="148.70.194.135", port=27017)
# 2. 字符串形式
client2 = MongoClient("mongodb://148.70.194.135:27017/")
```

指定数据库/集合

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
# 指定数据库
db1 = client.box2
db2 = client["box"]
# 指定集合
col1 = db1.box
col2 = db2.box2
print(col2)
```

```
Collection(Database(MongoClient(host=['148.70.194.135:27017'],
document_class=dict, tz_aware=False, connect=True), 'box'), 'box2')
```

插入数据

新增数据，都必须先获取collection对象，获取对象顺序为：客户端连接-> db -> collection

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

tester = {
    "name": "dcs",
    "age": 18
}
# 旧版本使用这个方法，不推荐使用
# col.insert(tester)
# 新版本使用这个方法
col.insert_one(tester)
```

```
<pymongo.results.InsertOneResult at 0x10b252608>
```

- 插入多条数据

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

tester1 = {
    "name": "gugu",
    "age": 18
}

tester2 = {
    "name": "dcs",
    "age": 18,
    "friends": 18,
}
# 使用列表即可
res = col.insert_many([tester1, tester2])
# 获取id
print(res.inserted_ids)
```

```
[ObjectId('5e47d1b1e9171d040f0acfcf'), ObjectId('5e47d1b1e9171d040f0acfd0')]
```

查询数据

- 基础查询

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

# 1. 按条件查询，默认返回第一个
res = col.find_one({"name": "gugu"})
print(res)

# 2. 查找全部
res = col.find()
print(res, type(res))
for i in res:
    print(i)

# 3. 按条件查询，查询多条
res = col.find({"age": 18})
for i in res:
    print(i)

# 4. 查询大于18的
res = col.find({"age": {"$gt": 18}})
for i in res:
    print(i)
```

```
None
<pymongo.cursor.Cursor object at 0x10affaf60> <class 'pymongo.cursor.Cursor'>
{'_id': ObjectId('5e47d0e7e9171d040f0acfc8'), 'name': 'dcs', 'age': 18}
{'_id': ObjectId('5e47d13de9171d040f0acfca'), 'name': 'dcs', 'age': 18}
{'_id': ObjectId('5e47d13de9171d040f0acfcb'), 'name': 'dcs', 'age': 18,
'friends': 18}
{'_id': ObjectId('5e47d0e7e9171d040f0acfc8'), 'name': 'dcs', 'age': 18}
{'_id': ObjectId('5e47d13de9171d040f0acfca'), 'name': 'dcs', 'age': 18}
{'_id': ObjectId('5e47d13de9171d040f0acfcb'), 'name': 'dcs', 'age': 18,
'friends': 18}
```

- 比较符号的使用

符号	含义	实例
`$lt`	小于	`{'age': {'$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]}}`

- 正则匹配的使用

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

res = col.find({'name': {'$regex': '^gu.*'}})
for i in res:
    print(i)
```

```
{'_id': ObjectId('5e47d1b1e9171d040f0acfcf'), 'name': 'gugu', 'age': 18}
```

- 属性是否存在：查询包含指定字段的数据

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

res = col.find({'age': {'$exists': True}})
for i in res:
    print(i)
```

```
{'_id': ObjectId('5e47d0e7e9171d040f0acfc8'), 'name': 'dcs', 'age': 18}
{'_id': ObjectId('5e47d13de9171d040f0acfca'), 'name': 'dcs', 'age': 18}
{'_id': ObjectId('5e47d13de9171d040f0acfcb'), 'name': 'dcs', 'age': 18,
'friends': 18}
{'_id': ObjectId('5e47d1b1e9171d040f0acfcf'), 'name': 'gugu', 'age': 18}
{'_id': ObjectId('5e47d1b1e9171d040f0acfd0'), 'name': 'dcs', 'age': 18,
'friends': 18}
```

- 查询指定数据类型

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

res = col.find({'age': {'$type': 1}})
for i in res:
    print(i)
```

```
{'_id': ObjectId('5e47d22f663a9ddd50cccf7e'), 'name': 'bird', 'age': 18.8}
```

Type	Number	Alias	Notes
Double	1	“double”	
String	2	“string”	
Object	3	“object”	
Array	4	“array”	
Binary data	5	“binData”	
Undefined	6	“undefined”	Deprecated.
ObjectId	7	“objectId”	
Boolean	8	“bool”	
Date	9	“date”	
Null	10	“null”	
Regular Expression	11	“regex”	
DBPointer	12	“dbPointer”	Deprecated.
JavaScript	13	“javascript”	
Symbol	14	“symbol”	Deprecated.
JavaScript (with scope)	15	“javascriptWithScope”	
32-bit integer	16	“int”	
Timestamp	17	“timestamp”	
64-bit integer	18	“long”	
Decimal128	19	“decimal”	New in version 3.4.
Min key	-1	“minKey”	
Max key	127	“maxKey”	

- 文本查询

在使用文本查询之前，必须为字段添加索引：`db.box.ensureIndex({ name: "text", description : "text", category : "text" });`

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box
# {$text:{$search:"index"}}
res = col.find({'$text': {'$search': 'gugu'}})
for i in res:
    print(i)
```

```
{'_id': ObjectId('5e47d1b1e9171d040f0acfcf'), 'name': 'gugu', 'age': 18}
```

- 高级条件查询：使用自身的字段进行判断

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box
res = col.find({'$where': 'obj.age == obj.friends'})
for i in res:
    print(i)
```

```
{'_id': ObjectId('5e47d13de9171d040f0acfcb'), 'name': 'dcs', 'age': 18,
'friends': 18}
{'_id': ObjectId('5e47d1b1e9171d040f0acfd0'), 'name': 'dcs', 'age': 18,
'friends': 18}
```

更新操作

- update()直接修改，如果查询多个，默认修改第一个

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

condition = {"name": "gugu"}
res = col.find_one(condition)
# 更改字段
res["age"] = 21
# 更新mongo，默认修改第一个查询，官方不推荐
res2 = col.update(condition, res)
# nModified表示修改的行数
print(res2)
```

```
{'ok': 1, 'nModified': 1, 'n': 1, 'updatedExisting': True}
```

```
/Users/panwj/.local/share/virtualenvs/mypython3-Xz6Q90qj/lib/python3.7/site-packages/ipykernel_launcher.py:11: DeprecationWarning: update is deprecated. Use replace_one, update_one or update_many instead.
  # This is added back by InteractiveShellApp.init_path()
```

- 单独修改1个：`update_one()`；注意更新格式：`{'$set': res}`

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

condition = {"name": "dcs"}
result = col.update_one(condition, {'$set': {"age": 19}})
# 打印匹配的数据条数和影响的数据条数
print(result.matched_count, result.modified_count)
```

```
1 1
```

- 更新多条

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

condition = {"name": "dcs"}
result = col.update_many(condition, {'$set': {"age":21}})
# 打印匹配的数据条数和影响的数据条数
print(result.matched_count, result.modified_count)
```

```
4 4
```

删除操作

- 删除所有数据：`remove()`

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

result = col.remove({'name': 'gugu'})
print(result)
```

```
{'ok': 1, 'n': 1}
```

```
/Users/panwj/.local/share/virtualenvs/mypython3-Xz6Q90qj/lib/python3.7/site-packages/ipykernel_launcher.py:6: DeprecationWarning: remove is deprecated. Use delete_one or delete_many instead.
```

- 删除单条

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

result = col.delete_one({'name': 'dcs'})
print(result.deleted_count)
```

```
1
```

- 删除多条

```
from pymongo import MongoClient
client = MongoClient("mongodb://148.70.194.135:27017/")
db = client.box
col = db.box

result = col.delete_many({'name': 'dcs'})
print(result.deleted_count)
```

```
14
```