目录

主流文档型数据库：MongoDB

- 最新版本：`v2.0`
- 更新时间：`20200918`

简介

介绍主流文档型关系数据库MongoDB的起源；如何安装；基本的使用方法，包括如何启动服务端和如何用shell、GUI图形界面工具、代码调用API接口等方式去Client端的使用；以及部分高级用法，包括IP限制、用户和权限等；总结各种心得，包括pymongo的各种方面，备份和回复，如何新建空数据库和集合，如何更新已有数据，如何进行正则、列表、嵌套等高级搜索查询，如何使用GridFS保存二进制文件，以及一些常用教程和文档；总结出常见的各种坑及解决办法。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/popular_document_db_mongodb: 主流文档型数据库：MongoDB

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 主流文档型数据库：MongoDB book.crifan.org
- 主流文档型数据库：MongoDB crifan.github.io

离线下载阅读

- 主流文档型数据库：MongoDB PDF
- 主流文档型数据库：MongoDB ePub
- 主流文档型数据库：MongoDB Mobi

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.org` ，我会尽快删除。谢谢合作。

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

MongoDB简介

下面针对MongoDB做一下简要介绍：

- MongoDB
  - Logo
    - 
  - 一句话描述：**一种主流的文档数据库**
    - 一种：还有其他的
      - 其他文档型数据库：`CouchDB`、`Amazon DynamoDB`、`Couchbase`、`MarkLogic`
    - 文档数据库：
      - = `面向文档的数据库` = `Document-Oriented Datdabase`
      - 范畴：属于非关系型数据库
        - 常称为：`NoSQL`
      - 与之对应：
        - (传统的)关系型数据库：`MySQL`
        - 其他的NoSQL
          - 键值对(K/V)数据库：`redis`、`Cassandra`、`LevelDB`
          - 图数椐库：`Neo4j`
          - 时序数据库：`InfluxDB`
          - （全文）搜索（数据库）引擎：`ElasticSearch`、`Solr`
          - 列式数据库：`HBase`
      - 数据格式
        - 主要：`JSON`
          - 引申：`BSON`
          - 其他：`XML`
      - 优势
        - 修改数据和结构很方便
          - 直接修改JSON数据本身
          - 对比：传统MySQL需要改表结构
          - 引申出：容易兼容历史数据
            - 字段不存在只是空值不会报错
        - 复杂JSON可以描述复杂（嵌套）的数据结构

        - 适用场景
            - 数据量很大或者未来会变得很大
            - 表结构不明确，且字段在不断增加
        - 不适用场景
            - 在不同的文档上需要添加事务支持
                - 文档数据库不支持文档间的事务
            - 多个文档直接需要复杂查询
                - 例如join
    - 起源
        - 移动互联网兴起
            - 不仅：数据量大（高并发），架构复杂
            - 还要：快速响应
        - 结论：传统MySQL类关系型数据库无法满足
        - 出现：关系型数据库
            - 其中最流行：`MongoDB`
                - 胜出关键：易用、架构良好、功能丰富
    - 概述
        - 底层实现：`C++`
        - 支持平台：`Windows`、`Mac`、`Linux`、`Solaris`等
        - 编程接口API：常见语言都支持
            - `C`、`C++`、`C#`、`Go`、`Java`、`Node.js`、`Perl`、`PHP`、`Python`、`Ruby`、`Rust`、`Scala`、`Swift`
    - 优势
        - 高性能
        - 富查询语言（支持 CRUD、数据聚合、文本搜索和地理空间查询）
        - 高可靠性
        - 自动伸缩架构
        - 支持多存储引擎

安装

MongoDB有2个版本：

- `Community Edition` = 社区版 = **免费版**
- `Enterprise Edition` = 企业版 = **收费版**

下面主要介绍**免费的社区版本**：

MongoDB支持多种操作系统和平台，对应安装方式可参考：

- Linux
  - 各种发行版
    - `Red Hat / CentOS`
    - `Ubuntu`
    - `Debian`
    - `SUSE`
    - `Amazon Linux`
- macOS
- Windows

Mac中安装MongoDB

接下来详细介绍Mac中如何安装MongoDB

目前免费的Community社区版MongoDB最新版是：`v4.2`

Mac中安装最新版MongoDB 4.2 community社区版方式：

先去：

```
brew tap mongodb/brew
```

再去安装 `MongoDB` ：

```
brew install mongodb-community
```

再去[可选]安装 `mongo shell` ：

```
brew install mongodb-community-shell
```

基本用法

安装好了MongoDB后，接下来去搞清楚如何操作MongoDB。

操作和使用MongoDB的基本逻辑都是：

- 启动MongoDB的服务端
- 用某种Client去操作MongoDB
  - 根据类型分，常见Client有3种方式
    - 命令行
      - MongoDB自带：`mongo shell`
        - 提供了一个最基础的操作数据库的方式
    - GUI图形界面工具
      - 通过图形界面工具去查看和操作数据会更加直观和方便
        - 如`MongoDB Compass`、`Robot 3T`等
    - API
      - 用各种语言的代码，通过API操作数据库
        - 如`Python`、`Java`等

接下来分别去介绍MongoDB的服务器和客户端。

基本概念

MongoDB中的一些概念和术语的含义，可以通过和SQL对比去理解：

SQL术语和概念	MongoDB术语概念	解释和说明
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`作为主键

MongoDB的Server端

启动本地MongoDB服务

- 对于普通安装的MongoDB

直接在终端中运行：

```
mongod
```

- 对于安装的是mongodb-community

如果Mac中安装的MongoDB是 `mongodb-community` ，那么用：

```
brew services start mongodb-community
```

以及，关于MongoDB服务端的其他常见管理方式有：

- 启动： `brew services start mongodb-community`
- 停止： `brew services stop mongodb-community`
- 重启： `brew services restart mongodb-community`
- 查看状态：
  - `brew services list`
    - 如何确定已运行：看到 `mongodb-community` 是 `started`
  - 用 `ps`
    - `ps -ef | grep mongod`
    - `ps aux | grep mongod`
      - -》有输出 `mongod`

配置

典型的MongoDB的服务端文件是：

`/etc/rc.d/init.d/mongod`

而其中核心配置是：

```
CONFIGFILE="/etc/mongod.conf"
OPTIONS=" -f $CONFIGFILE"
mongod=${MONGOD-/usr/bin/mongod}
MONGO_USER=mongod
MONGO_GROUP=mongod
```

对应配置文件是：

`/etc/mongod.conf`

内部核心参数是：

```
# Where and how to store data.
storage:
  dbPath: /var/lib/mongo
```

表示数据库存放位置是：`/var/lib/mongo`

启动远程服务器中的MongoDB服务

- 方案1：切换到mongo用户再去启动

如果是通过SSH连接的远程服务器中：

mongod的服务，是作为mongod的组和用户，去在开机时启动的，可以正常启动的话

那么自己用ssh作为root用户登录进来后，是root用户，是无法启动属于`mongod:mongod`的服务mongod

在当前登录用户root的情况下，换用mongd的用户去，启动mongod：

```
sudo -u mongod mongod -f /etc/mongod.conf
```

才可以。

- 方案2：用systemctl或service去管理mongod

如果本身已有服务可以管理mongod，则可以通过服务管理去启动：

```
systemctl restart mongod
```

等价于：

```
service mongod restart
```

MongoDB的Client端

下面介绍，在启动了MongoDB的服务端之后，如何通过客户端，主要指的是各种语言和工具，去使用MongoDB。

命令行shell

MongoDB在安装后，自带 `mongo shell` ，是个交互式终端。可以在里面以命令行方式操作数据库。

最新版需要单独安装shell

最新的社区免费版mongodb-community的shell需要单独安装

`brew install mongodb-community-shell`

启动mongo shell

在命令行中输入

```
mongo
```

回车后，即可进入shell界面：

基本语法

切换到想要进入的数据库

```
use dbToSwitch
```

举例：

```
> use log
switched to db log
```

确认当前所在是哪个数据库

语法：

```
db
```

举例：

```
> db
admin
```

和

```
> db
log
```

新建用户

在创建新用户之前，先要用有权限的用户，比如 `admin` 登录进去，才能有权限创建新用户

注：此处之前已创建过 `admin` 超级用户，所以先去登录进入

```
mongo --host localhost --port 32018 -u root -p P@w --auther
```

然后再去给还没创建的新数据库去创建新用户

需要先去切换到对应的（虽然此时还不存在的）新的数据库：

```
use newDbName
```

然后才能（在当前数据库下）创建新用户：

```
db.createUser({
  user: "newUserName",
  pwd: "yourPassword",
  roles: [ { role: "dbOwner", db: "newDbName" } ]
})
```

然后可以确认一下是否创建成功：

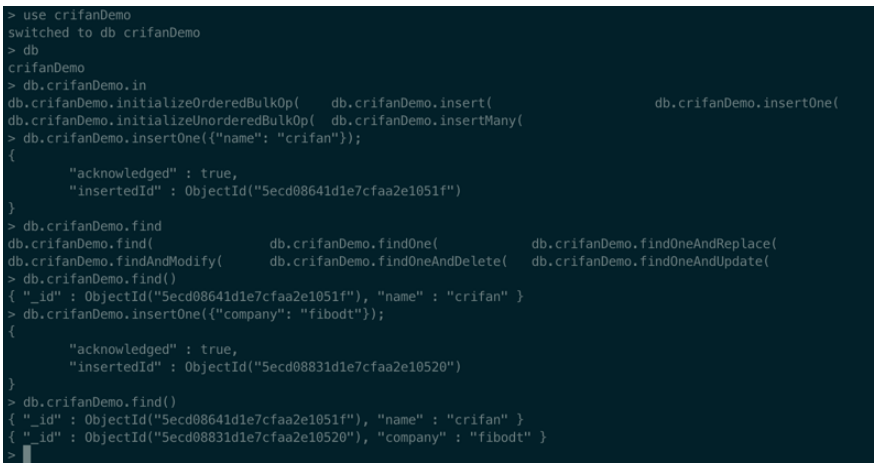
```
show users
```

即可看到新创建的用户。

举例

创建数据库并新增数据

举例：



过程解释：

直接 `use crifanDemo`

```
> use crifanDemo
switched to db crifanDemo
```

即可创建新的数据库 `crifanDemo`

```
> db
crifanDemo
```

用 `db` 查看当前所在数据库

后续用：

```
> db.crifanDemo.insertOne({"name": "crifan"});
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5ecd08641d1e7cfaa2e1051f")
}
```

去真正写入数据，才会真正（自动）创建一个 `database` 。

然后再去确认数据的确已经写入，可以用 `find` 去查找当前所有数据

在输入了 `db.crifanDemo.find` ，再按 `Tab` 键，则可以自动匹配出相关命令：

```
> db.crifanDemo.find
db.crifanDemo.find(                db.crifanDemo.findOne(
db.crifanDemo.findAndModify(       db.crifanDemo.findOneAndD
```

然后用 `find` 可以找出当前的所有的数据：

```
> db.crifanDemo.find()
{ "_id" : ObjectId("5ecd08641d1e7cfaa2e1051f"), "name" : "c
```

再去新增一条数据：

```
> db.crifanDemo.insertOne({"company": "company_name"});
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5ecd08831d1e7cfaa2e10520")
}
```

再去 `find` 即可看到数据的确增加到2条了：

```
> db.crifanDemo.find()
{ "_id" : ObjectId("5ecd08641d1e7cfaa2e1051f"), "name" : "c
{ "_id" : ObjectId("5ecd08831d1e7cfaa2e10520"), "company" :
```

`find` 也支持参数查找，比如：

```
> db.crifanDemo.find({"name": "crifan"})
{ "_id" : ObjectId("5ecd08641d1e7cfaa2e1051f"), "name" : "c
```

以及支持更多其他高级用法，比如搜索条件支持正则：

```
> db.crifanDemo.find({"name": {$regex: "cri"}})
{ "_id" : ObjectId("5ecd08641d1e7cfaa2e1051f"), "name" : "c
```

更多数据查询和用法，详见后面章节：

高级搜索

想要给新的暂时还不存在的数据库log中创建用户log

具体过程是：

```
> use log
switched to db log
> db.createUser({
...   user: "log",
...   pwd: "NL2@18Log",
...   roles: [ { role: "dbOwner", db: "log" } ]
... })
Successfully added user: {
        "user" : "log",
        "roles" : [
                {
                        "role" : "dbOwner",
                        "db" : "log"
                }
        ]
}
> show users
{
        "_id" : "log.log",
        "user" : "log",
        "db" : "log",
        "roles" : [
                {
                        "role" : "dbOwner",
                        "db" : "log"
                }
        ]
}
```

清空旧用户创建新用户

查看（当前数据库的）用户：

```
show users;
```

举例

```
> use admin
switched to db admin
> show users
{
    "_id" : "admin.root",
    "user" : "root",
    "db" : "admin",
    "roles" : [
        {
            "role" : "root",
            "db" : "admin"
        }
    ]
}
> use gridfs
```

创建用户：

- 切换到amind数据库
- 清除掉admin的之前其他用户
- 创建一个叫root的，角色是root的用户（拥有超级管理员，操作任意其他数据库的权限）

```
> use admin
switched to db admin
> db.runCommand({dropAllUsersFromDatabase: 1})
{ "n" : 1, "ok" : 1 }
> db.createUser({
...     user: "root",
...     pwd: "pwd",
...     roles: [ { role: "root", db: "admin" } ]
... })
Successfully added user: {
        "user" : "root",
        "roles" : [
                {
                        "role" : "root",
                        "db" : "admin"
                }
        ]
}
> show users
{
        "_id" : "admin.root",
        "user" : "root",
        "db" : "admin",
        "roles" : [
                {
                        "role" : "root",
                        "db" : "admin"
                }
        ]
}
```

查看当前用户

```
show users
```

举例：

```
> show users
{
        "_id" : "admin.root",
        "user" : "root",
        "db" : "admin",
        "roles" : [
                {
                        "role" : "root",
                        "db" : "admin"
                }
        ]
}
```

删除用户

语法：

```
db.dropUser("userToDelete")
```

举例：

```
> db.dropUser("log")
true
```

删除数据库

```
> db.dropDatabase()
{ "dropped" : "storybook", "ok" : 1 }
```

从GridFS中找歌曲类型的文件

命令行shell

```
> db.fs.files.find({"metadata.resourceType": "song"}).limit
{
    "_id" : ObjectId("5b21d3787f4d384d04543f6e"),
    "contentType" : "audio/x-ms-wma",
    "chunkSize" : 261120,
    "metadata" : {
        "song" : {
            "singers" : [ ]
        },
        "fitAgeStart" : 2,
        "topics" : [
            "Fingerplay",
            "Animal",
            "Weather"
        ],
        "storybook" : {
            "publisher" : "",
            "isFiction" : "",
            "lexileIndex" : "",
            "awards" : "",
            "authors" : [ ],
            "foreignCountry" : ""
        },
        "keywords" : {
            "fromName" : [
                "animal animal",
                "animal"
            ],
            "other" : [
                "Sun",
                "Rain",
                "water spout"
            ],
            "fromContent" : [ ]
        },
        "name" : "Animals, Animals",
        "resourceType" : "song",
        "mainActors" : [
            "Spider"
        ],
        "contentAbstract" : "",
        "isSeries" : true,
        "series" : {
            "number" : 1,
            "name" : "Wee Sing-Animals, Animals, Animals"
        },
        "fitAgeEnd" : 6,
        "fileInfo" : {
            "isAudio" : true,
            "contentType" : "audio/x-ms-wma",
```

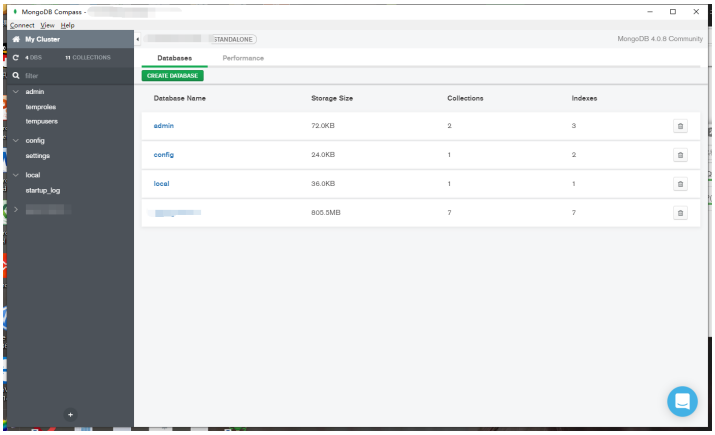
```
            "name" : "Animals, Animals.wma",
            "suffix" : "wma"
        }
    },
    "filename" : "Animals, Animals.wma",
    "length" : 2277430,
    "uploadDate" : ISODate("2018-06-14T02:31:20.767Z"),
    "md5" : "b334806c280cc37c4b873a8e2a2086cd"
}
{
    "_id" : ObjectId("5b21d3787f4d384d04543f78"),
    "contentType" : "audio/x-ms-wma",
    "chunkSize" : 261120,
    "metadata" : {
        "song" : {
            "singers" : [ ]
        },
        "fitAgeStart" : 2,
        "topics" : [
            "Fingerplay",
            "Family",
            "Others"
        ],
        "storybook" : {
            "publisher" : "",
            "isFiction" : "",
            "lexileIndex" : "",
            "awards" : "",
            "authors" : [ ],
            "foreignCountry" : ""
        },
        "keywords" : {
            "fromName" : [
                "old macdonald",
                "old Macdonald farm"
            ],
            "other" : [
                "knives",
                "forks",
                "mirror",
                "table",
                "looking glass",
                "cradle"
            ],
            "fromContent" : [ ]
        },
        "name" : "Old Macdonald Had a Farm",
        "resourceType" : "song",
        "mainActors" : [
            "mother",
            "baby"
```

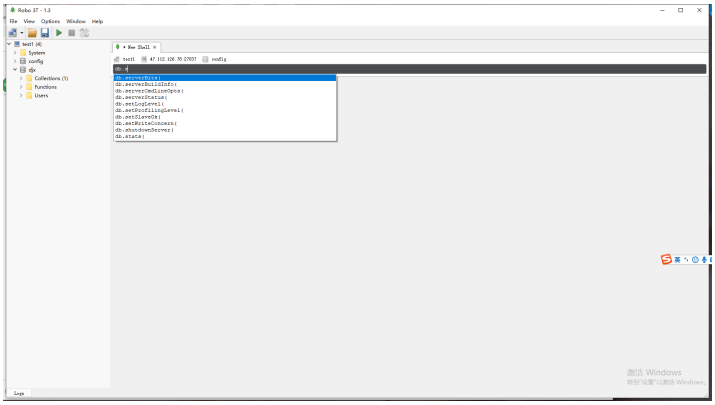
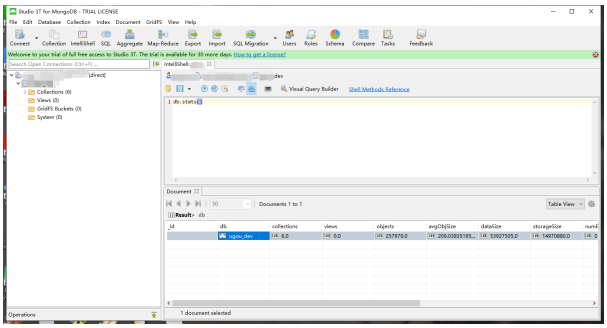
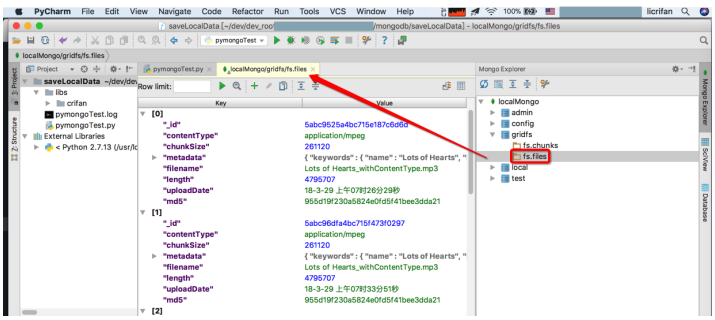
```
        ],
        "contentAbstract" : "",
        "isSeries" : true,
        "series" : {
            "number" : 2,
            "name" : "Wee Sing-Animals, Animals, Animals"
        },
        "fitAgeEnd" : 6,
        "fileInfo" : {
            "isAudio" : true,
            "contentType" : "audio/x-ms-wma",
            "name" : "Old Macdonald Had a Farm.wma",
            "suffix" : "wma"
        }
    },
    "filename" : "Old Macdonald Had a Farm.wma",
    "length" : 1924864,
    "uploadDate" : ISODate("2018-06-14T02:31:21.192Z"),
    "md5" : "a2b65c25d117d428beaa346b0b7e232f"
}
```

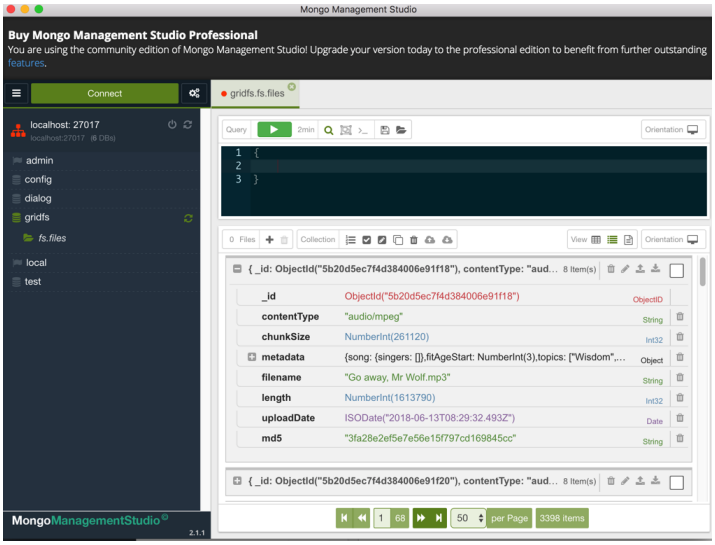
统计歌曲类型文件总数

```
> db.fs.files.find({"metadata.resourceType": "song", "metad
378

> db.fs.files.find({"metadata.resourceType": "song", "metad
523
> db.fs.files.find({"metadata.resourceType": "song"}).count
901
```

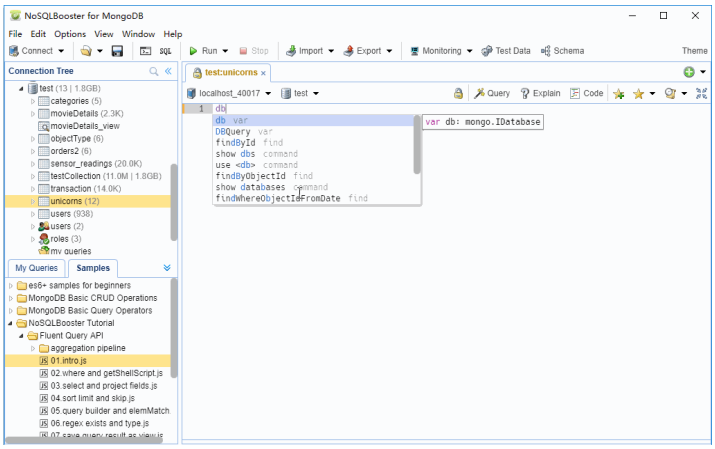
GUI工具

- `MongoDB Compass`
  - Logo
    - 
  - 特点
    - 官方出品
    - 免费
    - 颜值高
    - 功能够用
  - 截图
    - 
  - 官网
    - MongoDB Compass — MongoDB Compass stable
- `Robot 3T`
  - 旧称：`Robomongo`
  - Logo
    - 
  - 特点
    - 免费
    - 功能够用
  - 截图

    - 截图
      - 
    - 官网
      - Robo 3T | Free, open-source MongoDB GUI (formerly Robomongo)
    - 同一家公司的另外一款收费版
      - `Studio 3T`
        - 截图
          - 
        - 官网
          - The Professional GUI, IDE & Client for MongoDB | Studio 3T
- `mongo4idea`
  - PyCharm的MongoDB的插件
  - 截图
    - 
  - 官网
    - Github
      - dboissier/mongo4idea: MongoDB integration in Intellij
- `MMS = Mongo Management Studio`
  - 截图



    - 特点
        - 号称支持 `GridFS`
            - 但是支持的不完美
                - 不支持写入
        - 官网
            - Mongo Management Studio - the professional MongoDB GUI
- `NoSQLBooster`
    - 有免费版
        - 也有收费版
    - 截图



- 其他
    - VSCode支持MongoDB

部分对比

Robot 3T vs Studio 3T

- Studio 3T

    - Migrate databases & relations between SQL & MongoDB
    - Auto-complete queries with IntelliShell
    - Drag & drop fields to visually build queries
    - Use SQL to query MongoDB
    - Build aggregation queries stage by stage
    - Generate driver code in 6 languages
    - Automate repetitive MongoDB tasks like imports
    - And so much more...
- Robo 3T
    - For simple tasks
    - Embedded shell
    - Lightweight & fun

MongoDB Compass

TODO:

- 【已解决】MongoDB Compass中如何快速高效地刷新数据

下载和安装MongoDB Compass

根据官网介绍

Download and Install Compass — MongoDB Compass stable

去下载页面

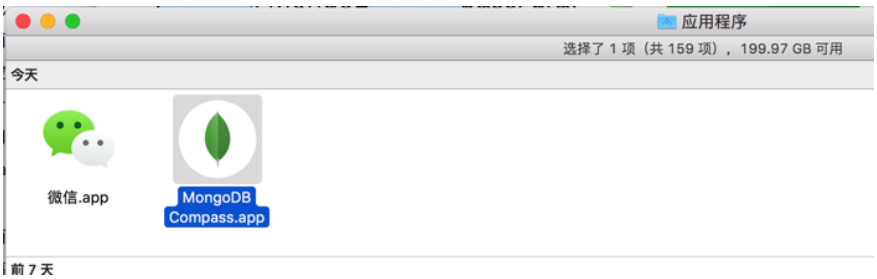
Compass | MongoDB

下载安装包

比如Mac的是

https://downloads.mongodb.com/compass/mongodb-compass-1.14.5-darwin-x64.dmg

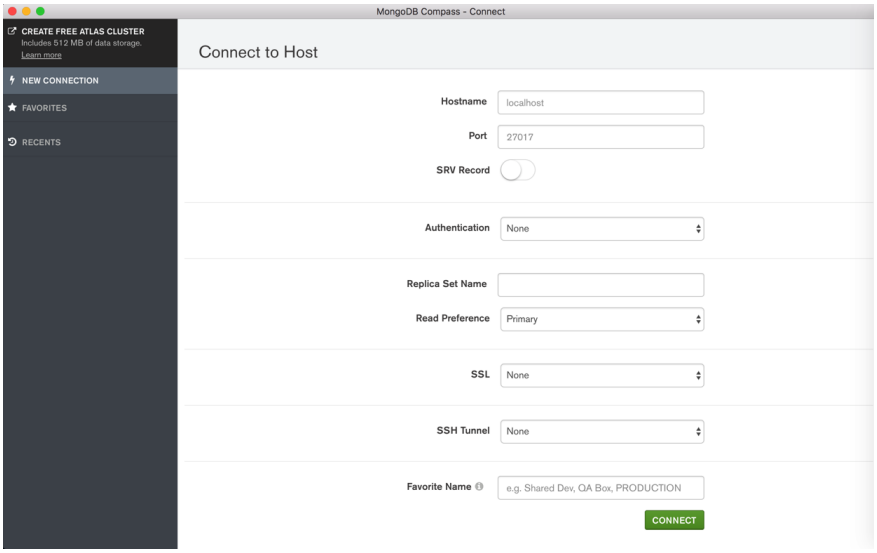
下载后，安装即可。安装后是：



当前版本是：`1.14.5`

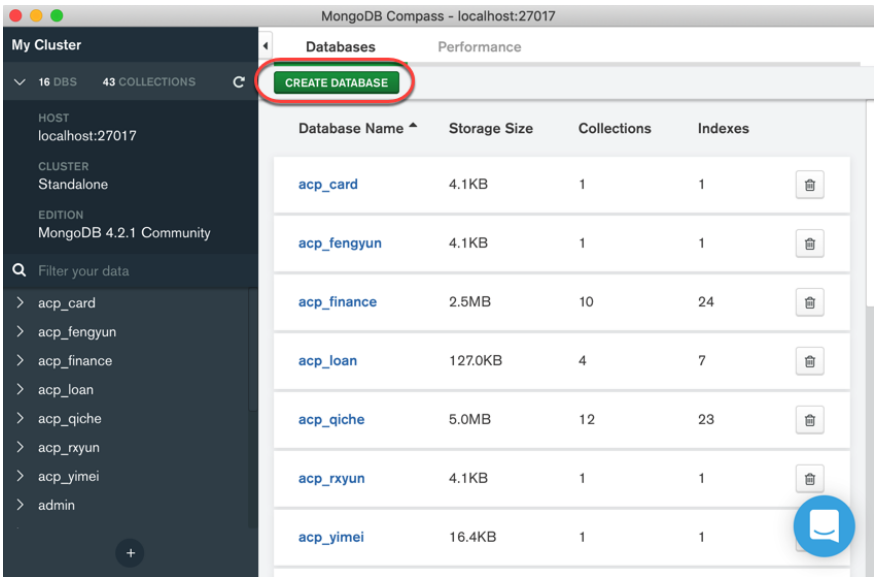
基本使用

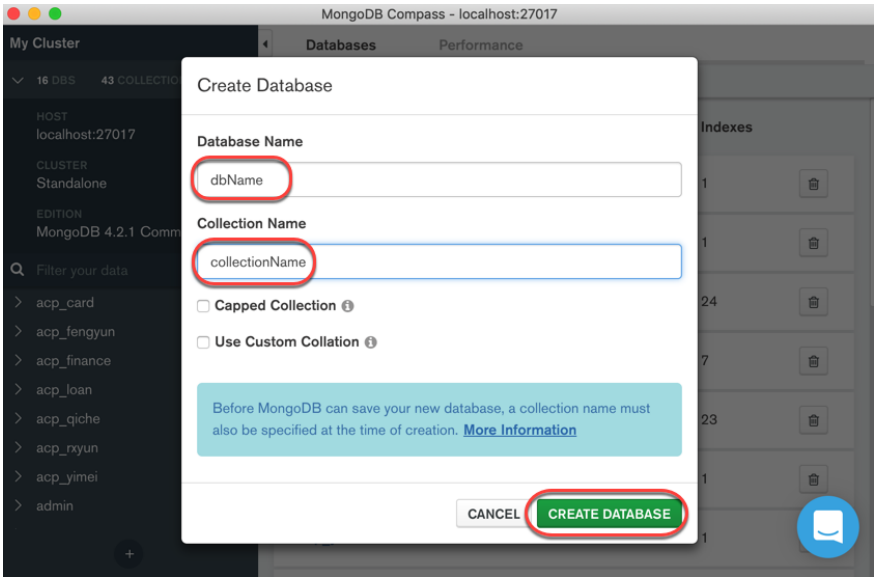
打开后，进入连接数据库页：

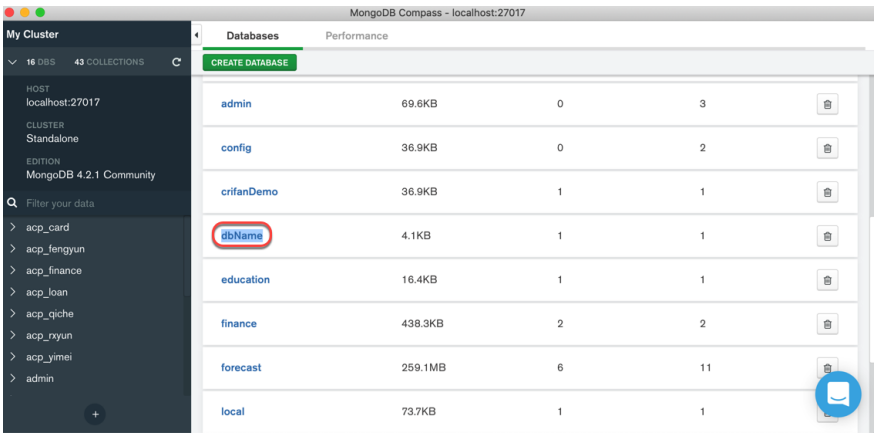


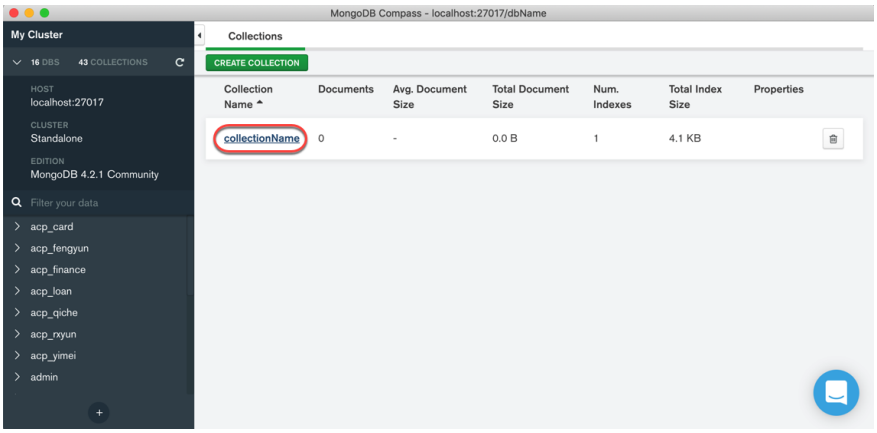
点击连接后，进入数据库列表页：

创建数据库和集合



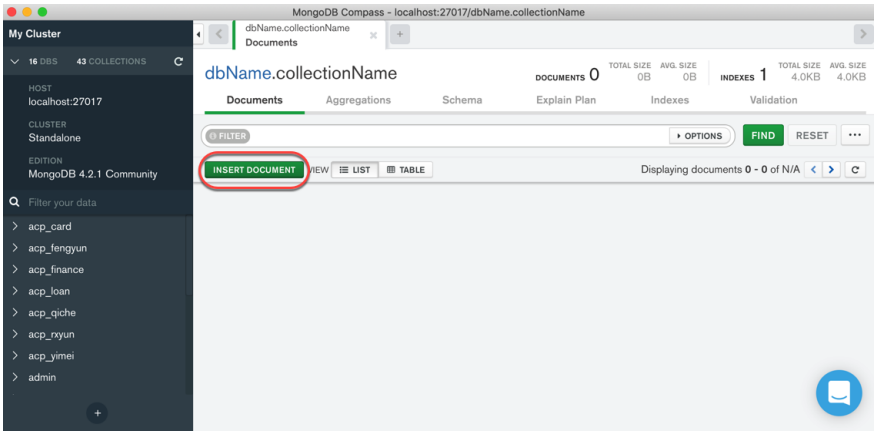




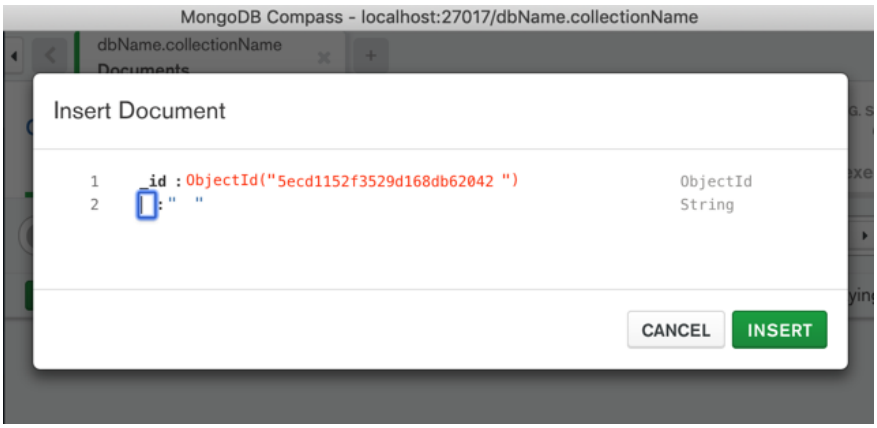


写入数据

点击 `INSERT DOCUMENT`：

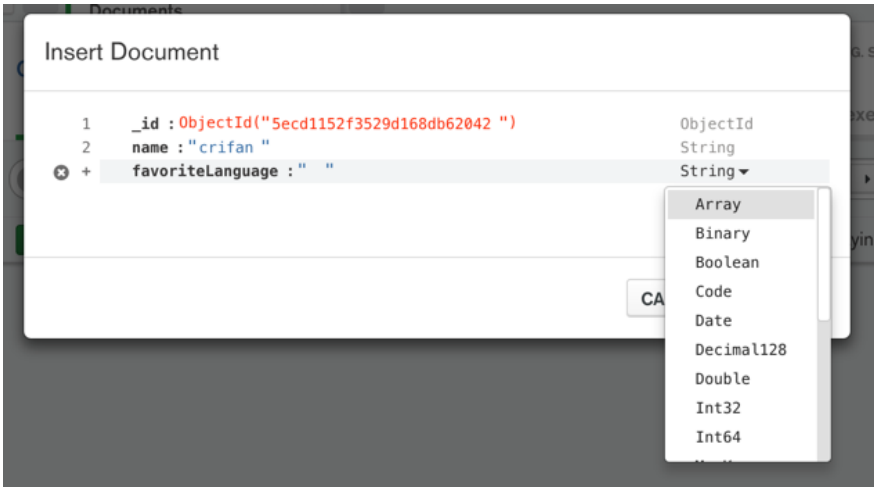


会出现编辑数据的弹框：

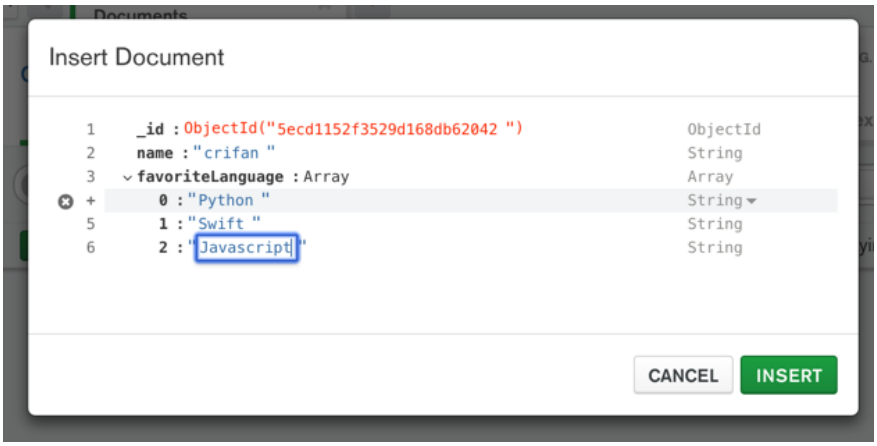


输入对应的数据的 `key` 和 `value`，如果想要新增字段，则点击左边的 `加号` ➕，会弹出 `Add Field after xxx`：

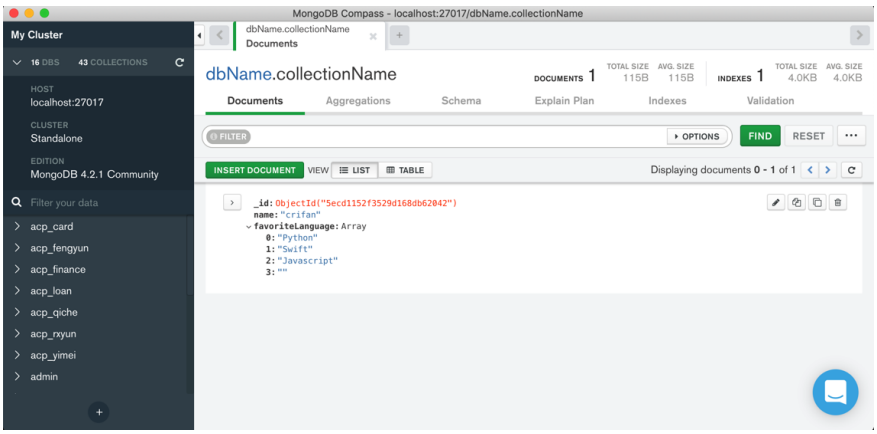
数据的类型，除了默认的 `String`，还支持其他类型，比如 `Array` 数组：



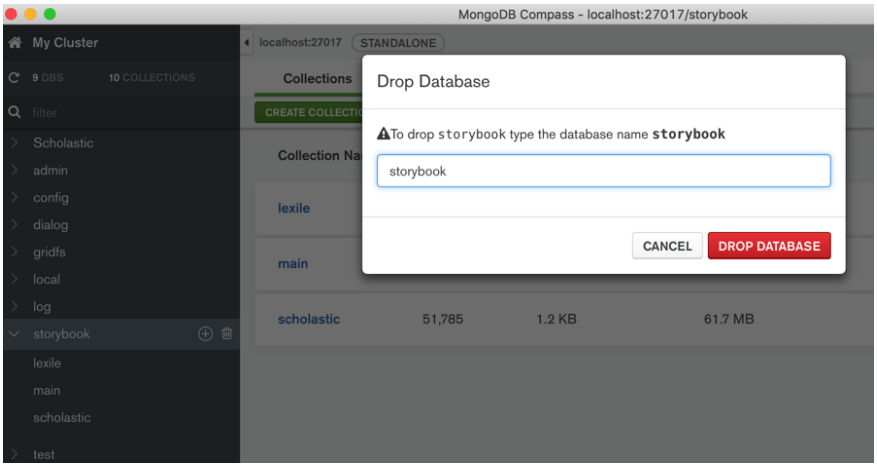
分别输入数组的每项的值后，点击 `INSERT` ：



即可返回列表页，看到刚插入的数据：



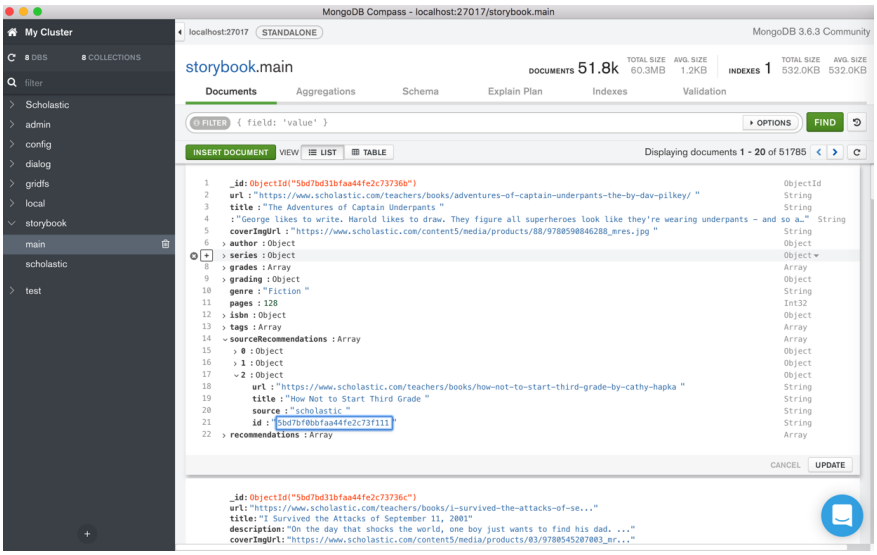
删除数据库



好用之处

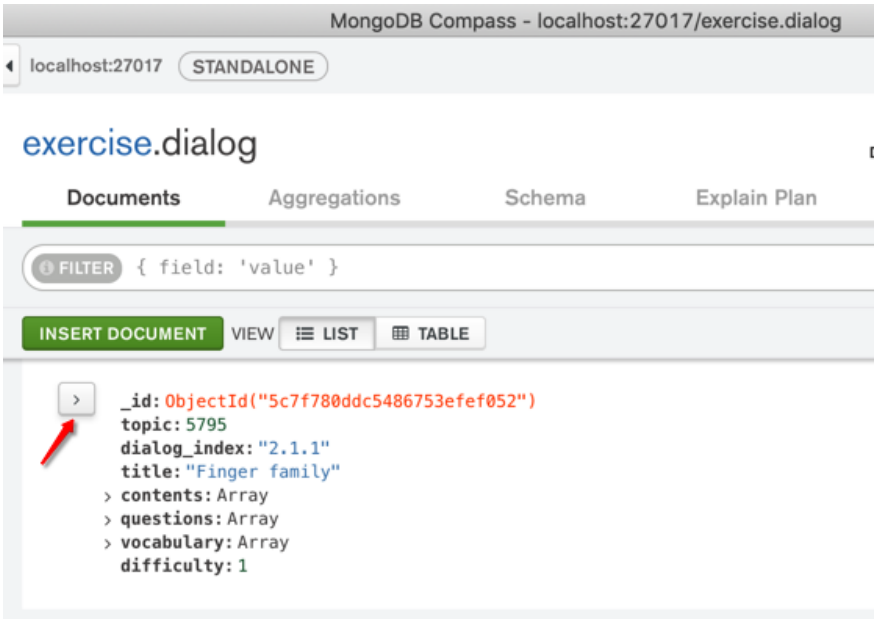
直接编辑内容

截图举例：

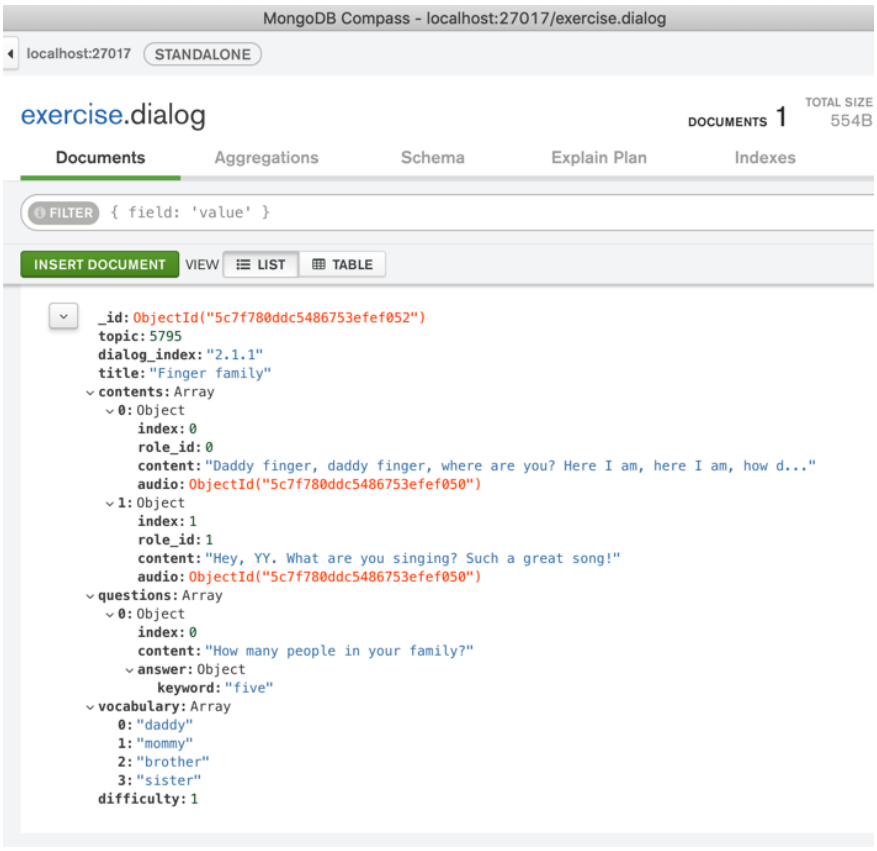


字段可以很方便的折叠和展开

点击每条记录前面的箭头：

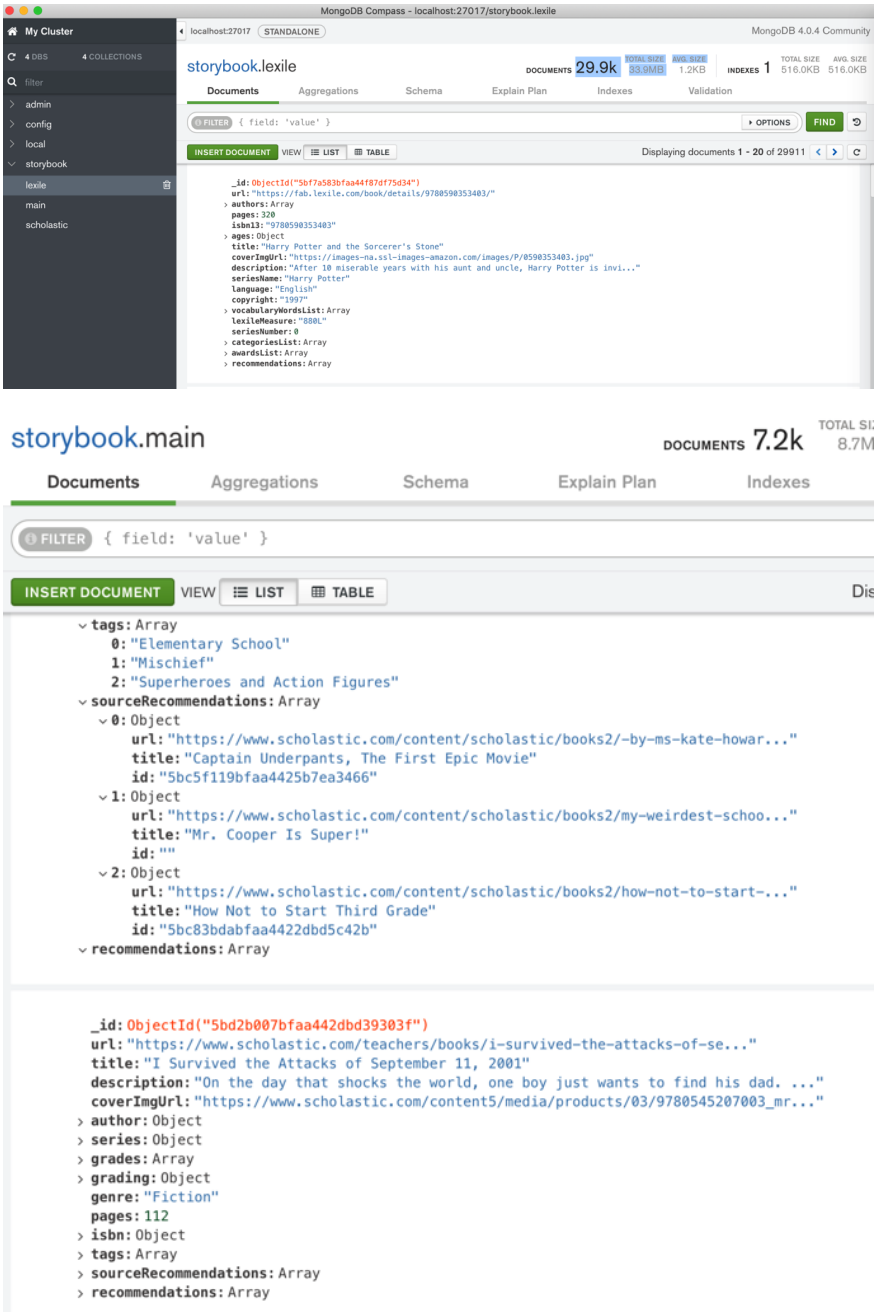


即可展开所有字段：



再次点击，即可缩回。

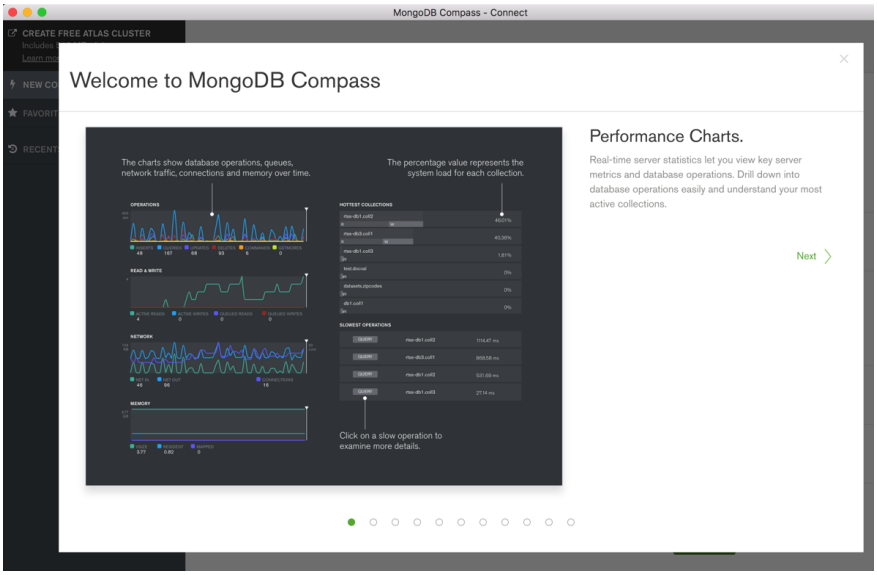
其他实际使用效果举例



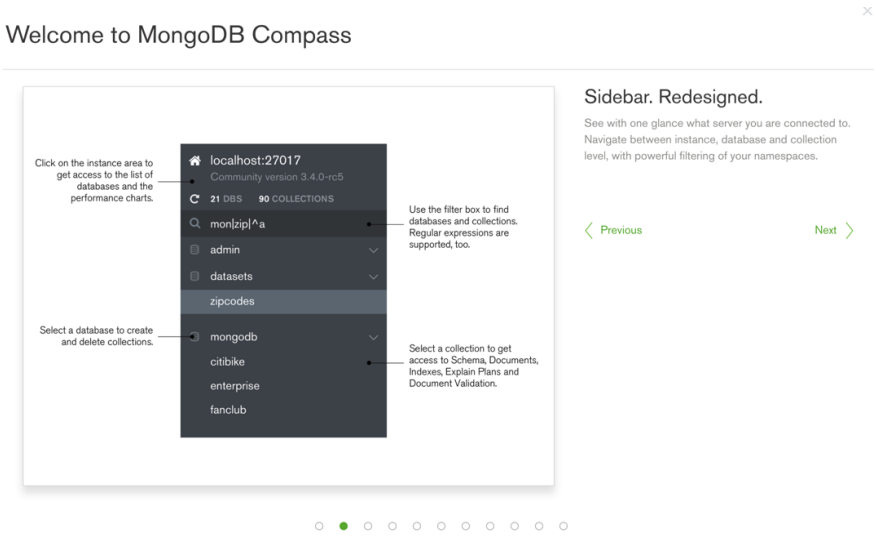
功能特性介绍=引导页

引导页有截图介绍其好用的功能和特点，供参考：

Perfromance Charts

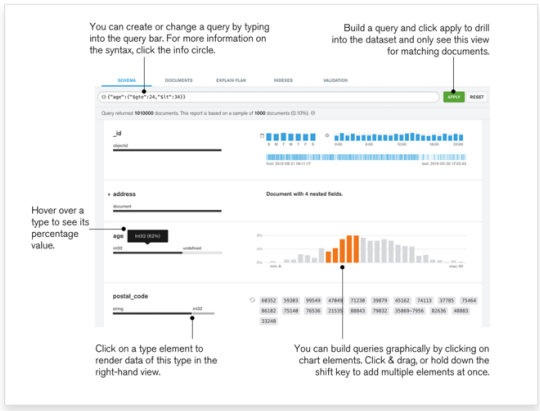


Sidebar Redesigned



Visualize your Schema

Welcome to MongoDB Compass



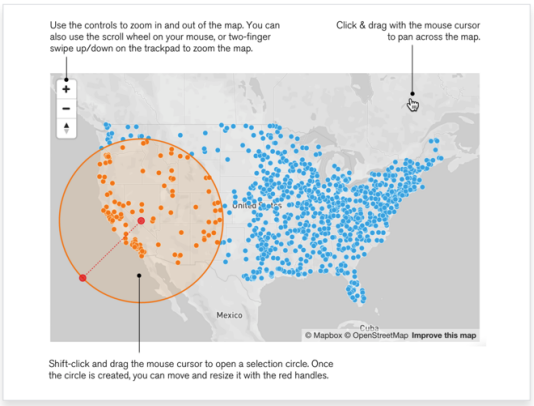
Visualize your Schema.

MongoDB Compass analyzes your documents and displays rich structures within your collections through an intuitive GUI. It allows you to quickly visualize and explore your schema to understand the frequency, types and ranges of fields in your data set.

Previous Next

Build Geo Queries

Welcome to MongoDB Compass



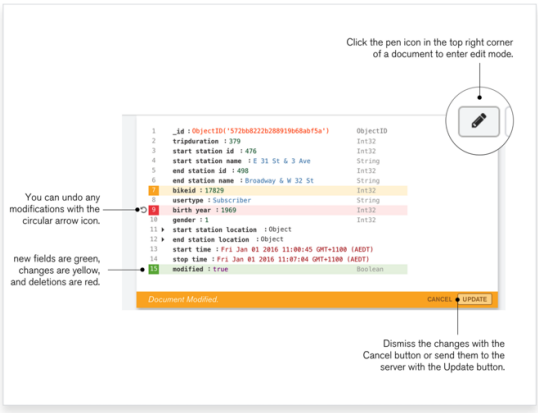
Build Geo Queries.

Visualize, understand, and work with your geospatial data. Point and click to construct sophisticated queries, execute them with the push of a button and Compass will display your results both graphically and as sets of JSON documents.

Previous Next

Interactive Document Editor

Welcome to MongoDB Compass



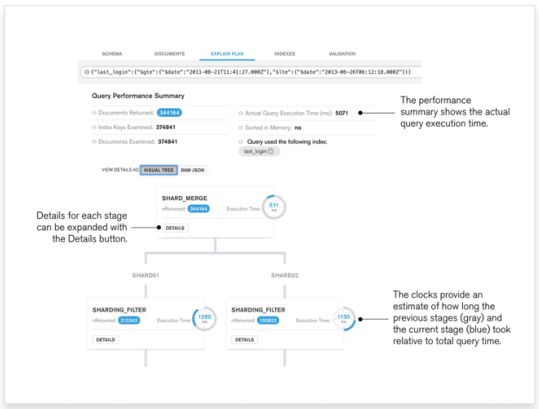
Interactive Document Editor.

Modify existing documents with greater confidence using the intuitive visual editor, or insert new documents and clone or delete existing ones in just a few clicks.

Previous Next

Visual Explain Plains

Welcome to MongoDB Compass



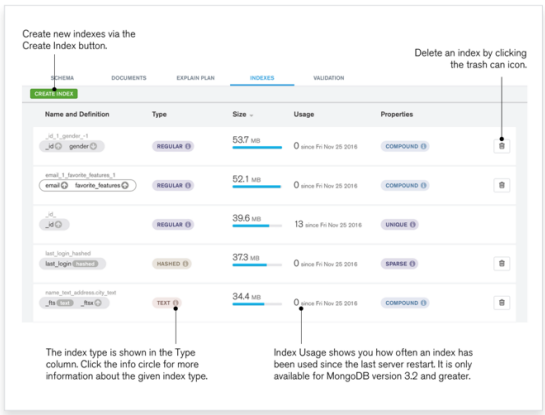
Visual Explain Plans.

Know how queries are running through an easy-to-understand GUI that helps you identify and resolve performance issues.

Previous Next

Index Management

Welcome to MongoDB Compass



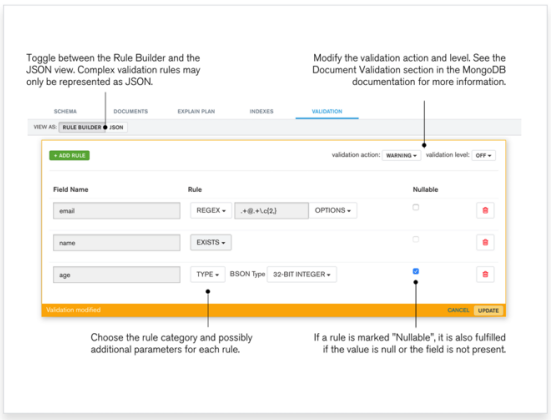
Index Management.

Understand the type and size of your indexes, their utilization and special properties. Add and remove indexes at the click of a button.

Previous Next

Document Validation

Welcome to MongoDB Compass

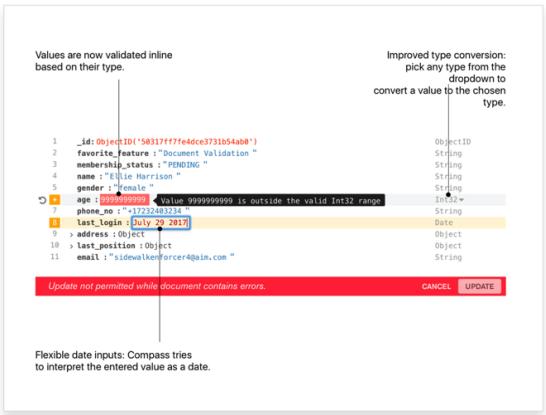


Document Validation.

Create and modify rules that validate your data using a simple point and click interface. CRUD support lets you fix data quality issues easily in individual documents.

Previous Next

Improved CURD



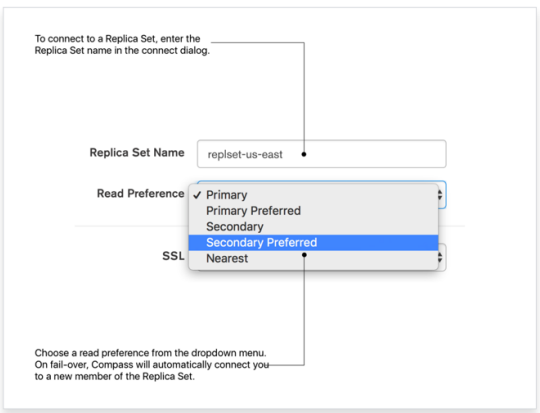
Improved CRUD

Better editing with validation of individual BSON types.

Previous Next

Deployment Awareness

Welcome to MongoDB Compass



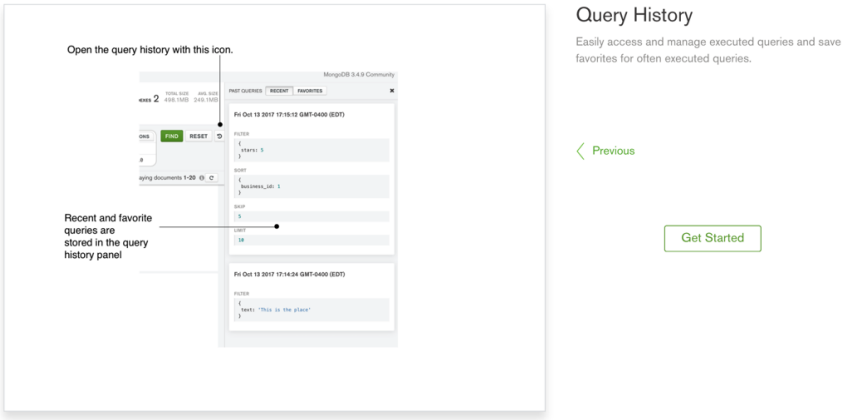
Deployment Awareness

Replica set aware connections allow for continued use during replica set configuration changes and provides additional information of the connected cluster.

Previous Next

Query History

Welcome to MongoDB Compass



通过API操作MongoDB

MongoDB支持通过API操作，支持多种语言：

- C
- C++
- C#/.NET
- Go
- Java
- Node.js
- Perl
- PHP
- Python
  - 同步模式：PyMongo
  - 异步模式：Motor
- Ruby
- Scala
- Swift
- Rust

PyMongo

用Python通过API操作的MongoDB，最常见的库是：`PyMongo`

基本使用

运行服务端：

```
mongod
```

代码中导入

```
import pymongo
```

连接Client

```
from pymongo import MongoClient
client = MongoClient()
```

如果需要，可以指定host和port：

```
client = MongoClient('localhost', 27017)
```

或者通过 `URI` 指定更多参数：

```
client = MongoClient('mongodb://localhost:27017/')
```

创建数据库：

```
db = client.test_database
```

也可以用字典属性方式访问数据库：

```
db = client['test-database']
```

访问集合：

```
collection = db.test_collection
```

也可以用字典属性方式访问集合：

```
collection = db['test-collection']
```

然后就可以正常操作了。

比如：插入（文档）数据

```
demoDict = {"name": "Crifan"}
new_doc_id = collection.insert_one(demoDict).inserted_id
print("new_doc_id=%s" % new_doc_id)
```

更多基本操作用法可官网教程：

Tutorial — PyMongo 3.9.0 documentation

更多心得和总结可参考后续章节：

PyMongo心得

资料

- 官网
  - 旧
    - 版本：`<= 3.9`
      - PyMongo教程
        - Tutorial — PyMongo 3.9.0 documentation
          - Mongo的Client
            - mongo_client – Tools for connecting to MongoDB — PyMongo 3.9.0 documentation
      - API概览
        - API Documentation — PyMongo 3.9.0 documentation
  - 最新：readthedocs.io
    - 版本：`>= 3.10`
      - PyMongo 3.10.1 Documentation — PyMongo 3.10.1 documentation
- 第三方
  - nummy/pymongo-tutorial-cn: pymongo中文教程

高级用法

此处介绍一些，相对来说属于高级的，MongoDB的用法。

IP限制

有时候为了安全，需要限定只有某个或某些IP才可以访问自己的（远端）MongoDB数据库。

这时候，就可以用上IP限制的功能了。

举例：

编辑配置文件 `vi /etc/mongod.conf` ，改为：

```
bindIp: 127.0.0.1,112.4.64.141 # Listen to specific IP
```

即可允许除了本机外的别的固定IP的访问。

添加了IP限制的mongod重启出错：Job for mongod.service failed because the control process exited with error code

不过，可能会遇到，添加了IP限制后，mongod启动出错：

`Job for mongod.service failed because the control process exited with error code`

的原因是：

其实有多种多样，具体的情况，需要根据实际情况去找原因，再去解决。

其中，辅助的办法是：

参考提提示的：

`Job for mongod.service failed because the control process exited with error code. See "systemctl status mongod.service" and "journalctl -xe" for details.`

去运行：

```
journalctl -xe
```

去看看具体的错误。

比如此处的：

```
Unregistered Authentication Agent for unix-process:4637:887
```

但是后来证明，无法根据此现象找到根本原因。

只是看起来像是：

端口权限方面的问题，有些人是SELINUX限制了端口导致的，而我此处没有开启SELINU，所以不是这方面的问题。

而真正解决问题的办法是：

此处主要是去看log文件

-> 从中（先后多次）找到真正的具体的错误信息

-> 然后才能知道错误原因

-> 然后才得以解决：

(1) 错误：listen(): bind() failed errno:98 Address already in use for socket: 127.0.0.1:32018

办法：停止掉mongo后再重新启动：

- 如果和我一样，没有root用户密码，则：reboot重启服务器
- 有root密码：以mongod用户去停止mongo
  - sudo -u mongod systemctl stop mongod

(2) 错误：getaddrinfo("112.4.64.141") failed: Name or service not known

办法：确保bindIp中多个IP中间逗号分隔时，没有空格：

```
bindIp: 127.0.0.1,112.4.64.141
```

(3) 错误：WiredTiger (13) [1523341777:968015][1095:0x7fa3cf097dc0], file:WiredTiger.wt, connection: /var/lib/mongo/WiredTiger.turtle: handle-open: open: Permission denied

办法：确认

/var/lib/mongo

其下的所有的文件，包括此处的WiredTiger.turtle，对于此处的mongod用户，要有权限

-> 可以考虑把所有文件的owner所有者，改为此处mongo 服务对应的所属用户：mongod:mongod

-> 以及后续还有另外一个相关的：

```
/var/lib/mongo/journal/
```

(4) 错误：Failed to unlink socket file /tmp/mongodb-32018.sock errno:1 Operation not permitted

办法：

删除这个sock文件：

```
sudo rm /tmp/mongodb-32018.sock
```

后，重启mongod服务，或直接重启服务器-》启动服务器会去启动mongod服务的

用户和权限

为了数据安全，可以针对同一个数据库，以及不同的内部的集合，创建和允许不同的用户和权限，去访问。

MongoDB支持 `基于用户权限的访问控制=` RBAC `=` Role-Based Access Control`

一个用户可以有一个或多个角色Role，决定了能访问哪个数据库，能进行什么操作

如何验证权限：

- 方式：启动mongod时加 `--auth` 参数
- 方式2：设置：security.authorization
  - 配置文件中去设置
    - `security` 的 `authorization` 为 `enabled`

在开启认证auth==开启访问控制之前，需要你在admin数据库中，创建一个是带有 `userAdmin` 或 `userAdminAnyDatabase` 权限的用户

-》否则意味着你没有一个拥有管理权限的用户，就没法创建其他普通用户了

不过对于创建用户本身来说：在开启访问控制之前或者之后，都是可以的。

关于权限的详细解释如下：

- Authentication Database
  - 创建一个用户的时候，需要制定对应数据库的
    - 这个数据库就是该用户的Authentication Database
    - 不过一个用户的权限，可以不限定Authentication Database，只要针对别的数据库给了相应权限即可
  - 创建用户
    - 主要含义：增加一个用户，对于某个资源resource，允许某些操作action（就有了某些权限privilege）
      - resource：
        - 典型的只有一个db
        - 也可以针对其他的，比如collection：
          - { db: "products", collection: "inventory" }
    - 典型写法：
      - createUser
        - 其中参数：
          - role的参数：
            - 可以用系统内置的，比如：

                - read
                - readWrite
                - dbAdmin
                - dbOwner
                - userAdmin
            - 也可以用用户自定义的role
        - 在创建用户时没有指定对应role的话，可以后续用：grantRolesToUser去额外加上role
        - 用户的唯一标识是： 用户名+授权数据库
            - 换句话说：如果在创建用户时，两个用户名一样，但是授权数据库不同，也是不同的用户
            - 如果想要用一个用户对多个数据库都有权限，那么应该是：创建单个用户，带上对应角色role，而这个role是允许访问多个数据库的
    - MongoDB中是把所有用户相关信息都保存到admin数据库中的system.users中了
        - 包括：用户名，密码，用户的授权数据库
        - 不建议直接操作该数据库，而建议（在mongo shell中）用用户管理命令，比如：
            - createUser
            - grantRolesToUser
            - usersInfo
            - 等等

认证Authentication和授权Authorization

官网总结：

> **Authentication** verifies the identity of a user
>
> **Authorization** determines the verified user's access to resources and operations

自己想到的比喻：

- Authentication=认证=识别用户（的身份）：是否是之前添加过（认识的）的用户，否则不让进
    - 就像要进去一间屋子，需要一把钥匙
    - 对于Mongo shell：
        - 需要提供：
            - 用户名username
            - 密码password
            - 对应的哪个数据库database
        - 对应着 mongo shell（或其他mongo API）的参数：
            - `--username <username>`，`-u <username>`

        - `--password <password>`，`-p <password>`
        - `--authenticationDatabase <dbname>`
      - 或者，先进去，再认证：
        - 先运行mongo，在shell里再去：
          - `use xxxDb`
          - `db.auth("username", "password")`
    - 而对于其他语言的Driver
      - Python
        - pymongo
          - MongoClient初始化时传递对应参数即可

```
mongoClient = MongoClient(
    host="11.22.33.44",
    port=27017,
    username="usr",
    password="pwd"
)
```

          - 或者是用Mongo URI，比如：

```
mongodb://usr:pwd@11.22.33.44:27017/db
```

- Authorization=授权=判断用户能干嘛：针对哪个数据库，有哪些操作权限
  - 就像进了一间屋子，到底能进哪间房，且每间房里只能允许能干什么事
    - 不能进别的房间，进了允许进的房间后，也不能干不允许干的事

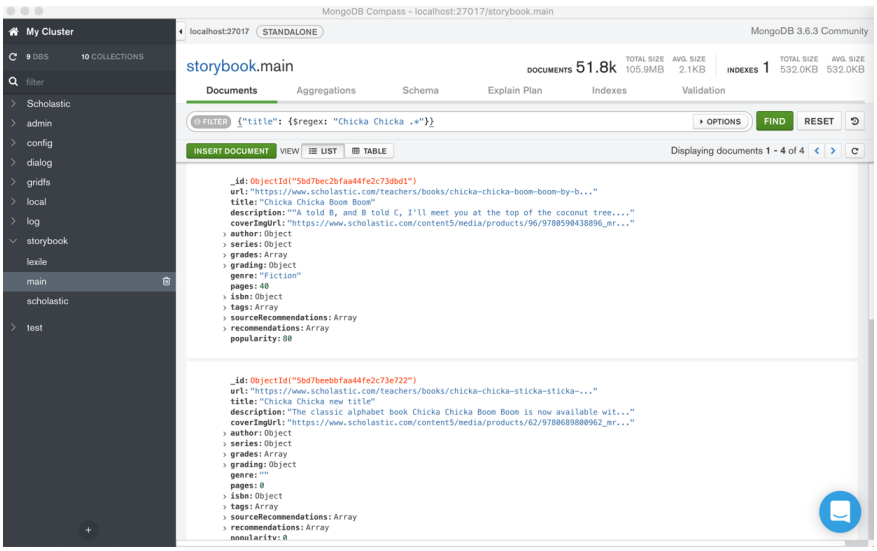
心得和总结

下面总结一些在MongoDB使用期间的心得和注意事项等内容。

先列出一些小的心得

导入数据之前，确保ID不能重复，否则会由于ID重复而无法覆盖

之前某次去恢复数据，故意没有删除本地之前已有（同样但是旧的）数据：



看看导入能否直接覆盖，结果由于ID重复而报错，覆盖失败：

```
  - E11000 duplicate key error collection: storybook.main :

^C2018-12-06T11:02:12.531+0800    signal 'interrupt' receiv
2018-12-06T11:02:12.560+0800    error: multiple errors in b
  - E11000 duplicate key error collection: storybook.schola
  - E11000 duplicate key error collection: storybook.schola
  - E11000 duplicate key error collection: storybook.schola
```

MongoDB Compass心得

编辑功能很好用

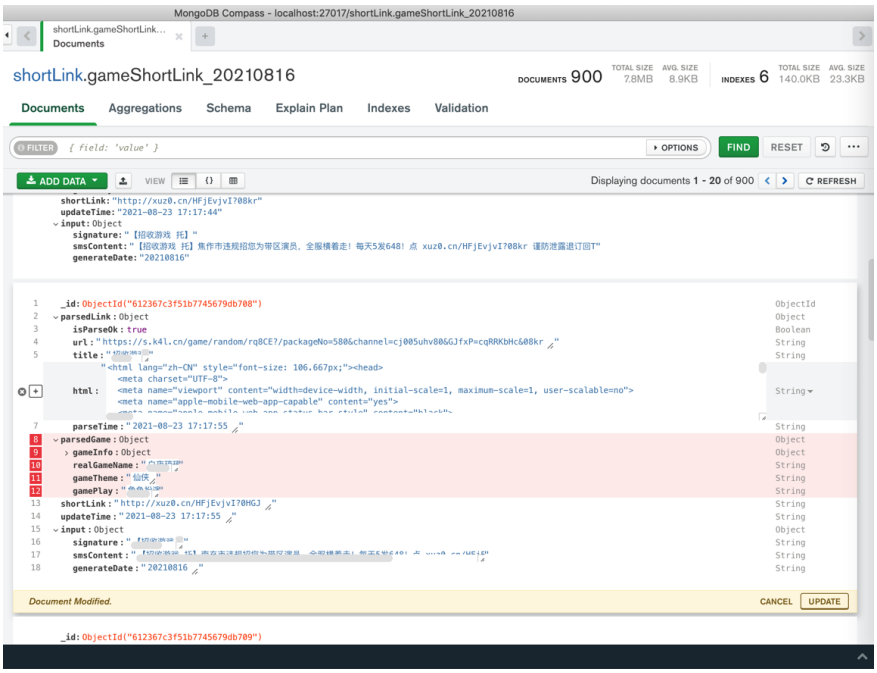
举例：

点击编辑：

或 双击字段的值，即可进入编辑模式

去编辑第三条数据，删除：parsedGame部分

鼠标移动到 改字段前面，点击 x 叉号：

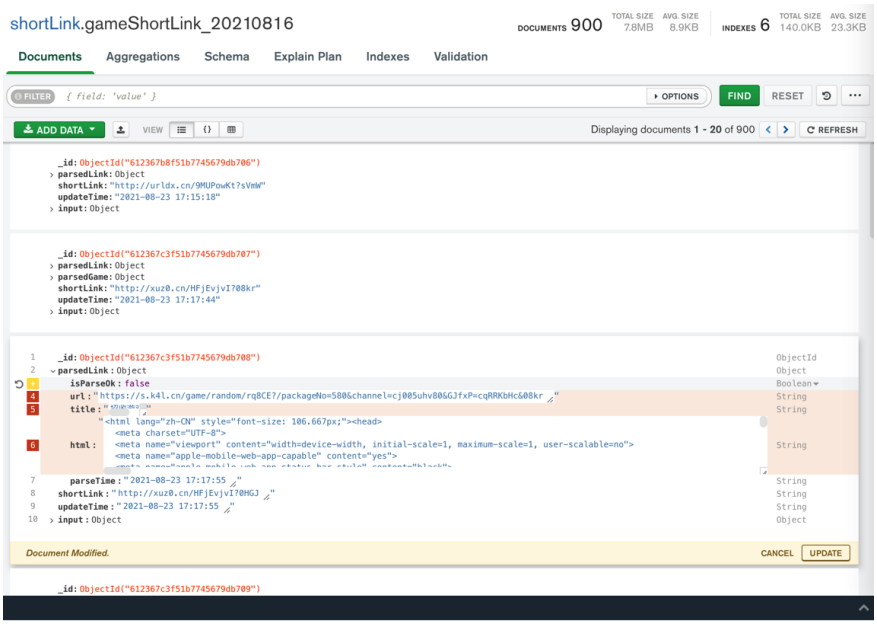


再点击 右下角的 UPDATE，即可删除对应字段。

再去改 parsedLink中的值：

把parsedLink中的isParseOk从true，改为 false

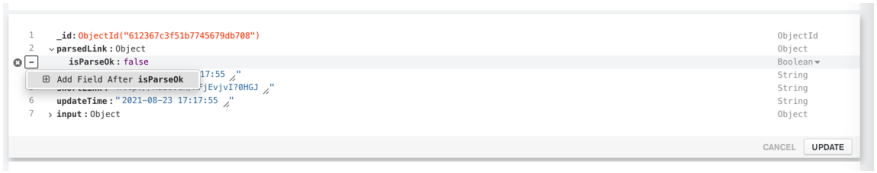
以及删除其他几个字段：



新增字段：

鼠标移动到要加的位置，点击 `加号` = ➕ ：

显示 `Add Field after isParseOk` ，点击



新增了一项：

PyMongo心得

此处整理Python语言操作MongoDB的常用的库 `PyMongo` 的一些使用心得。

生成URI

在开发期间，可能会涉及到，从配置中生成MongoDB的URI

下面代码供参考使用

```
def generateMongoUri(host=None,
                     port=None,
                     isUseAuth=False,
                     username=None,
                     password=None,
                     authSource=None,
                     authMechanism=None):
    """generate mongodb uri"""
    mongodbUri = ""

    if not host:
        # host = "127.0.0.0"
        host = "localhost"

    if not port:
        port = 27017

    mongodbUri = "mongodb://%s:%s" % (
        host, \
        port
    )
    # 'mongodb://localhost:27017'
    # 'mongodb://ip:27017'

    if isUseAuth:
        mongodbUri = "mongodb://%s:%s@%s:%s" % (
            quote_plus(username), \
            quote_plus(password), \
            host, \
            port \
        )
        print(mongodbUri)

        if authSource:
            mongodbUri = mongodbUri + ("/%s" % authSource)
            print("mongodbUri=%s" % mongodbUri)

        if authMechanism:
            mongodbUri = mongodbUri + ("?authMechanism=%s"
            print("mongodbUri=%s" % mongodbUri)

    print("return mongodbUri=%s" % mongodbUri)
    #mongodb://username:quoted_password@host:port/authSour
    #mongodb://localhost:27017

    return mongodbUri
```

调用举例：

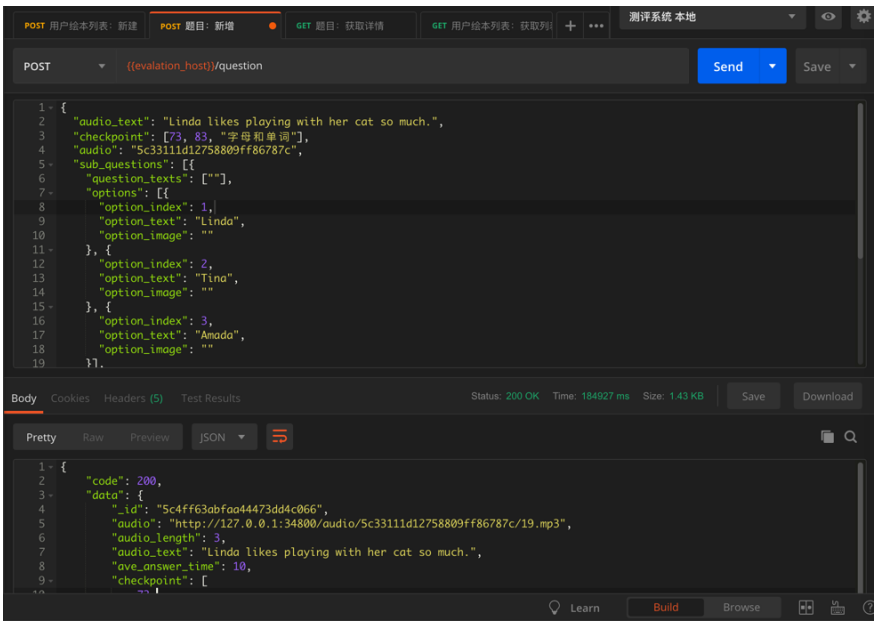
```
from pymongo import MongoClient

mongoUri = generateMongoUri(
    host=MONGODB_HOST,
    port=int(MONGODB_PORT),
    username=MONGODB_USERNAME,
    password=MONGODB_PASSWORD,
    authSource=MONGODB_AUTH_SOURCE,
    authMechanism=MONGODB_AUTH_MECHANISM,
)
mongoClient = MongoClient(mongoUri)
```

插入新增记录

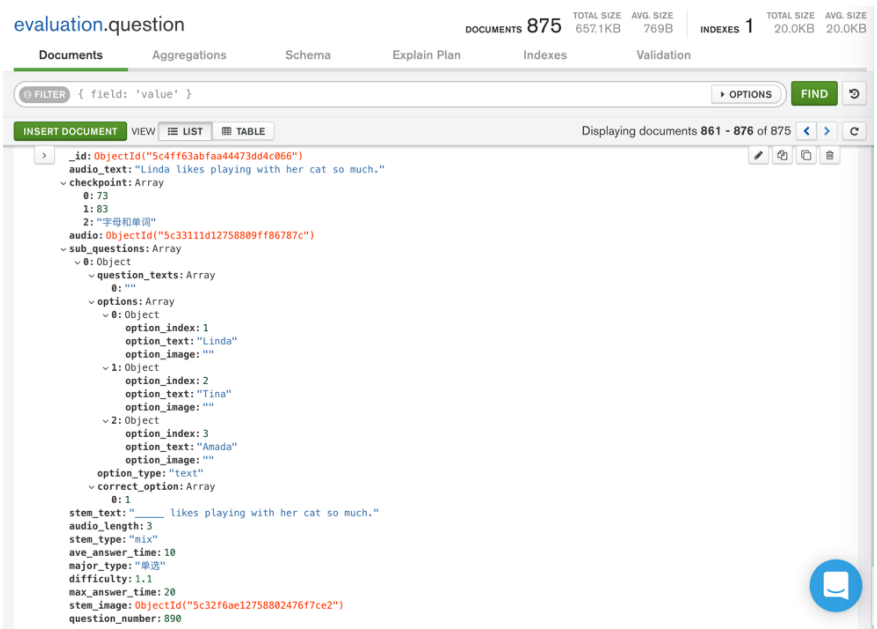
举例：

```
insertResult = questionCollection.insert_one(newQuestion)
newQuestionIdObj = insertResult.inserted_id
log.debug("newQuestionIdObj=%s", newQuestionIdObj)
```

类似的用Postman通过API调用效果：



插入后的数据在 `MongoDB Compass` 的显示效果：



设置时间范围等查询条件

代码：

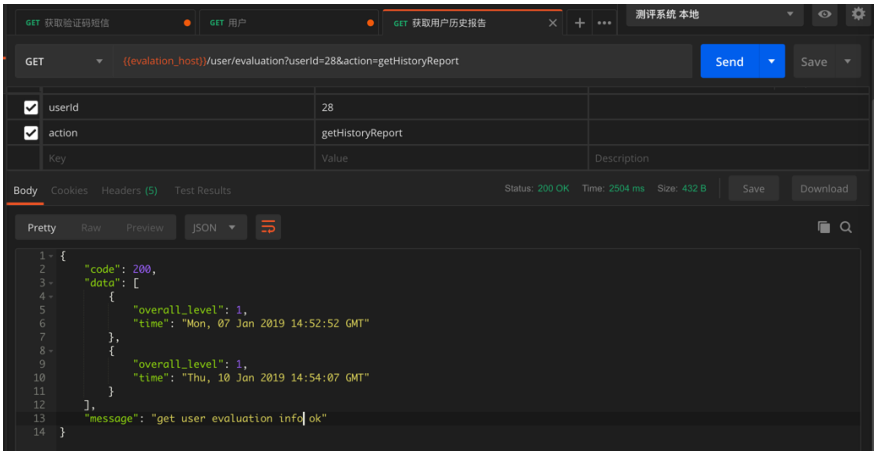
```
import pymongo
import datetime

curTime = datetime.datetime.now()
# timeKeyName = "start_time"
timeKeyName = "finish_time"
earliestTime = curTime - datetime.timedelta(days = 180)
userEvaluations = evalCollection.find({
    "user_id": userId,
    timeKeyName: {
        "$gte": earliestTime,
        "$lte": curTime
    }
}).sort(timeKeyName, pymongo.DESCENDING).limit(20)
userEvaluationList = list(userEvaluations)
userEvaluationList.reverse()
historyList = []
for eachEvaluation in userEvaluationList:
    if "overall_level" in eachEvaluation:
        curTimeLevelDict = {
            "time": eachEvaluation[timeKeyName],
            "overall_level": eachEvaluation["overall_level"
        }
        historyList.append(curTimeLevelDict)
log.debug("historyList=%s", historyList)
```

实现了期望的逻辑：

- 去mongodb查询
  - 最早半年前，最晚当前时间
  - 结果中再去根据时间倒序
  - 然后再去取最多20个
    - 从而实现：最近半年，最多20个，只不过顺序是反的而已
- 对于时间倒序后的结果，再调换顺序

即可得到需要的：最近半年的，最多20个，按照时间升序排列

然后输出希望的结果：



备份和恢复

MongoDB数据库，支持导出数据到文件，也支持从数据库文件导入，对应着备份和恢复。

与之对应，MongoDB提供了两个成对的配套的工具：

- `mongodump` 和 `mongorestore`
- `mongoexport` 和 `mongoimport`

下面详细介绍一下区别和具体用法。

`mongodump/mongorestore` 和 `mongoimport/mongoexport` 的区别

概述=结论：

- `mongodump` 和 `mongorestore` ：适合**整个database**，甚至**所有database**的数据的备份和恢复
  - 导出数据格式是： `bson`
- `mongoexport` 和 `mongoimport` ：适合**单个database**的**单个collection**的数据备份和恢复
  - 支持导入格式是： `json`

详解：

- 背景知识
  - JSON vs BSON
    - MongoDB内部数据保存用BSON
      - 原因：JSON是BSON的子集
        - -> 部分数据类型无法用JSON保存
          - -> 所以如果导出为JSON格式，可能会丢失部分复杂的数据类型的数据
- 区别和对比

	mongodump/mongorestore	mongoimport/mongoexport
导出格式	BSON	JSON
主要用途	简单且高效的 小型的MongoDB的数据库的备份/恢复	小规模的或部分的Mongo的数据的，测试期间的数据备份/恢复
额外说明	不适合备份/恢复 大型MongoDB数据库 mongodump不导出local（这个特殊的）数据库 mongodump不导出index索引	导出数据时使用严格模式（strict mode representation=MongoDB Extended JSON)仅支持UTF-8编码的数据
经验和结论	支持普通导出单个db的所有的collection 建议：普通的，简单的，数据的备份和恢复，都还是用mongodump/mongorestore这套工具	不支持普通导出单个db的所有的collection

- 其他相关
  - 如果是在线云平台部署的Mongo，备份/恢复可以使用
    - MongoDB Atlas
      - Fully Managed MongoDB, hosted on AWS, Azure, and GCP | MongoDB
  - 如果是 `replica sets` ，备份/恢复可以使用：
    - MongoDB Cloud Manager
    - 或
    - Ops Manager

mongodump 和 mongorestore

概述：

- **mongodump**
  - 导出整个数据库(db)

    ```
    mongodump -d databaseName -o outputFolder
    ```

    - 输出目录如果是：当前目录 .

      ```
      mongodump -d databaseName -o .
      ```

  - 导出数据库(db)中某个集合(Collection)= 子表

    ```
    mongodump -d databaseName -c CollectionName -o outp
    ```

    - 输出目录如果是：当前目录 .

      ```
      mongodump -d databaseName -c CollectionName -o
      ```

  - 其他说明
    - 导出的单个collection文件名一般是 collectionName.bson 和 collectionName.metadata.json
- **mongorestore**
  - 恢复（导入）某个目录下的某个数据库(db)中的所有的集合(collection)

    ```
    mongorestore -d databaseName ./localSubFoler
    ```

    - 说明：当前目录 localSubFoler 中有一个或多个 *.bson (以及对应的 *.metadata.json)
    - 举例

      ```
      mongorestore -d storybook ./storybook
      ```

  - 恢复（导入）某个数据库(db)中的单个集合(collection)

    ```
    mongorestore -d databaseName -c CollectionName subF
    ```

    - 注：mongorestore从文件导入数据的话，**不支持JSON**文件，**只支持BSON**文件
      - 且是用mongodump导出的BSON文件
    - 举例

```
mongorestore -d storybook -c lexile ./storybook
```

- 通用参数
  - 额外指定 `host` 和 `port`
    - 举例

      ```
      mongorestore -h localhost --port 32018 -d story
      ```

  - 额外指定（对应数据库和表的） `用户名` 和 `密码`
    - 举例

      ```
      mongorestore -h localhost --port 32018 -u story
      ```

详解：

mongodump备份

某次导出的命令：

```
 □ master □ mongodump -d shortLink -c gameShortLink -o .
2021-09-10T08:58:12.258+0800    writing shortLink.gameShort
2021-09-10T08:58:12.306+0800    done dumping shortLink.game
 □ master □ mongodump -d shortLink -c parsedPureShortLink
2021-09-10T08:58:21.449+0800    writing shortLink.parsedPu
2021-09-10T08:58:21.451+0800    done dumping shortLink.pars
```

导出后的文件：

```
 □ master □ ll
total 0
drwxr-xr-x  6 limao  1748468295   192B  9 10 08:58 shortLi
 □ master □ ll shortLink
total 16224
-rw-r--r--  1 limao  1748468295   7.8M  9 10 08:58 gameSho
-rw-r--r--  1 limao  1748468295   871B  9 10 08:58 gameSho
-rw-r--r--  1 limao  1748468295   131K  9 10 08:58 parsedP
-rw-r--r--  1 limao  1748468295   223B  9 10 08:58 parsedP
```

- 导出 本地MongoDB 的某表到当前文件夹
  - `mongodump -d storybook -o .`
- 导出 本地MongoDB 某表中某集合到当前文件夹，且指定host和port
  - `mongodump -h 127.0.0.1 --port 27017 -d Scholastic -c Storybook -o .`
- 导出 远程阿里云的MongoDB 某表中某集合到当前文件夹，且指定host和port，以及指定用户名和密码

    - `mongodump --host dds-xxx.mongodb.rds.aliyuncs.com --port xxx --authenticationDatabase admin -u root -p xxx -d exercise -o .`

参数解释：

- `-d` = `--database` ： 数据库 Scholastic
- `-c` = `--collection` ： 集合=表，Storybook
- `--type` ： 默认为 `json`
    - 所以此处可以不传此参数，用默认值
- `-o` == `--out` ： `.` 表示 **当前文件夹**

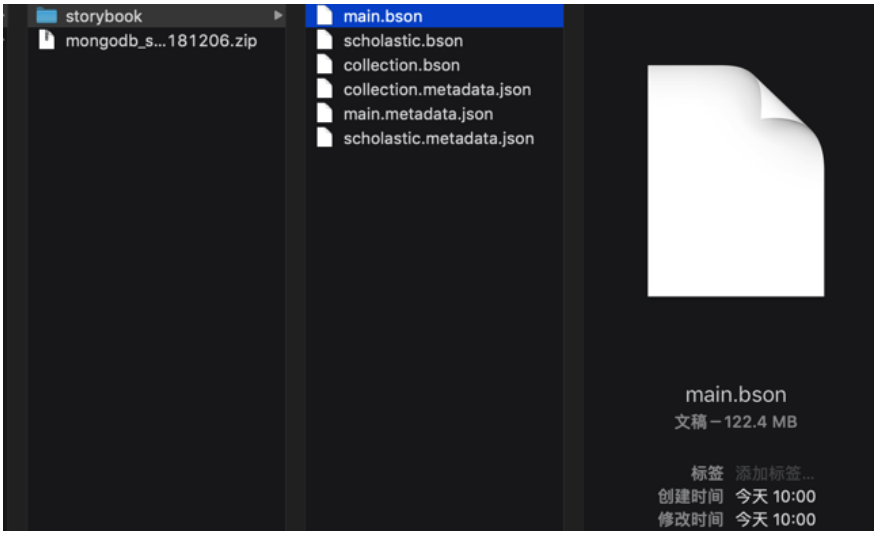
shell输出：

```
Scholastic
    Storybook.bson # 数据文件, 60M
    Storybook.metadata.json # 元数据, 130B
```

举例：

```
[root@xxx-general-01 exercise]# mongodump --host dds-xxx.mo
2019-03-07T11:30:24.036+0800    writing exercise.storybook
2019-03-07T11:30:24.037+0800    writing exercise.unit to
2019-03-07T11:30:24.037+0800    writing exercise.dialog to
2019-03-07T11:30:24.037+0800    writing exercise.audio.file
2019-03-07T11:30:24.038+0800    done dumping exercise.story
2019-03-07T11:30:24.038+0800    writing exercise.audio.chun
2019-03-07T11:30:24.038+0800    done dumping exercise.unit

2019-03-07T11:30:24.039+0800    done dumping exercise.audio
2019-03-07T11:30:24.047+0800    done dumping exercise.audio
2019-03-07T11:30:24.048+0800    done dumping exercise.dialo
[root@xxx-general-01 exercise]# ll
total 4
drwxr-xr-x 2 root root 4096 Mar  7 11:30 exercise
[root@xxx-general-01 exercise]# ll exercise/
total 60
-rw-r--r-- 1 root root 20737 Mar  7 11:30 audio.chunks.bson
-rw-r--r-- 1 root root   195 Mar  7 11:30 audio.chunks.meta
-rw-r--r-- 1 root root   249 Mar  7 11:30 audio.files.bson
-rw-r--r-- 1 root root   197 Mar  7 11:30 audio.files.metad
-rw-r--r-- 1 root root   554 Mar  7 11:30 dialog.bson
-rw-r--r-- 1 root root    87 Mar  7 11:30 dialog.metadata.j
-rw-r--r-- 1 root root   834 Mar  7 11:30 storybook.bson
-rw-r--r-- 1 root root    90 Mar  7 11:30 storybook.metadat
-rw-r--r-- 1 root root   309 Mar  7 11:30 unit.bson
-rw-r--r-- 1 root root    85 Mar  7 11:30 unit.metadata.jso
```

另附上，之前某次导出的数据的文件如下：



mongodump语法help帮助

```
➜  output git:(master) ✗ mongodump --help
Usage:
  mongodump <options>

Export the content of a running server into .bson files.

Specify a database with -d and a collection with -c to only

See http://docs.mongodb.org/manual/reference/program/mongod

general options:
      --help
      --version

verbosity options:
  -v, --verbose=<level>

      --quiet

connection options:
  -h, --host=<hostname>
      --port=<port>

ssl options:
      --ssl
      --sslCAFile=<filename>
      --sslPEMKeyFile=<filename>
      --sslPEMKeyPassword=<password>
      --sslCRLFile=<filename>
      --sslAllowInvalidCertificates
      --sslAllowInvalidHostnames
      --sslFIPSMode

authentication options:
  -u, --username=<username>
  -p, --password=<password>
      --authenticationDatabase=<database-name>
      --authenticationMechanism=<mechanism>

namespace options:
  -d, --db=<database-name>
  -c, --collection=<collection-name>

uri options:
      --uri=mongodb-uri

query options:
  -q, --query=
      --queryFile=
      --readPreference=<string>|<json>
```

```
      --forceTableScan

output options:
  -o, --out=<directory-path>
      --gzip
      --repair
      --oplog
      --archive=<file-path>

      --dumpDbUsersAndRoles
      --excludeCollection=<collection-name>

      --excludeCollectionsWithPrefix=<collection-prefix>

  -j, --numParallelCollections=
      --viewsAsCollections
```

mongorestore恢复

举例：

- 从某个目录，导入整个database：

```
➜  from_server ll
total 48
drwxr-xr-x  12 crifan  staff   384B  3  7 11:30 exercise
-rw-r--r--@  1 crifan  staff    20K  3  7 11:32 exercise_2
➜  from_server ll exercise
total 120
-rw-r--r--  1 crifan  staff    20K  3  7 11:30 audio.chunk
-rw-r--r--  1 crifan  staff   195B  3  7 11:30 audio.chunk
-rw-r--r--  1 crifan  staff   249B  3  7 11:30 audio.files
-rw-r--r--  1 crifan  staff   197B  3  7 11:30 audio.files
-rw-r--r--  1 crifan  staff   554B  3  7 11:30 dialog.bson
-rw-r--r--  1 crifan  staff    87B  3  7 11:30 dialog.meta
-rw-r--r--  1 crifan  staff   834B  3  7 11:30 storybook.b
-rw-r--r--  1 crifan  staff    90B  3  7 11:30 storybook.m
-rw-r--r--  1 crifan  staff   309B  3  7 11:30 unit.bson
-rw-r--r--  1 crifan  staff    85B  3  7 11:30 unit.metada
➜  from_server mongorestore -d exercise ./exercise
2019-03-07T11:51:15.303+0800    the --db and --collection a
2019-03-07T11:51:15.305+0800    building a list of collect
2019-03-07T11:51:15.308+0800    reading metadata for exerc
2019-03-07T11:51:15.309+0800    reading metadata for exerc
2019-03-07T11:51:15.309+0800    reading metadata for exerc
2019-03-07T11:51:15.310+0800    reading metadata for exerc
2019-03-07T11:51:15.585+0800    restoring exercise.unit fr
2019-03-07T11:51:15.678+0800    restoring exercise.storybo
2019-03-07T11:51:15.760+0800    restoring exercise.dialog
2019-03-07T11:51:15.846+0800    restoring exercise.audio.c
2019-03-07T11:51:15.851+0800    no indexes to restore
2019-03-07T11:51:15.851+0800    finished restoring exercis
2019-03-07T11:51:15.851+0800    no indexes to restore
2019-03-07T11:51:15.851+0800    finished restoring exercis
2019-03-07T11:51:15.858+0800    restoring indexes for coll
2019-03-07T11:51:15.860+0800    no indexes to restore
2019-03-07T11:51:15.860+0800    finished restoring exercis
2019-03-07T11:51:15.864+0800    reading metadata for exerc
2019-03-07T11:51:15.930+0800    finished restoring exercis
2019-03-07T11:51:16.029+0800    restoring exercise.audio.f
2019-03-07T11:51:16.031+0800    restoring indexes for coll
2019-03-07T11:51:16.077+0800    finished restoring exercis
2019-03-07T11:51:16.077+0800    done
➜  from_server
```

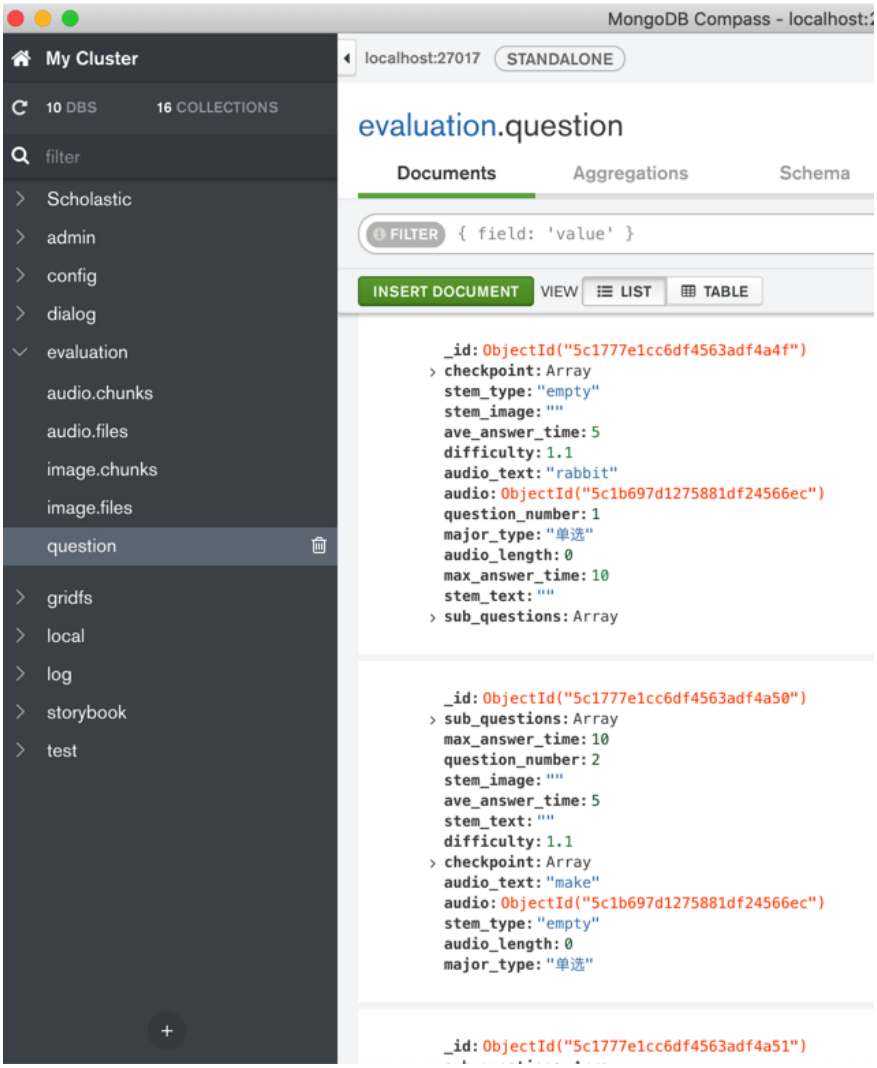
- 导入单个collection：

```
➜  mongodb_migration git:(master) pwd
/Users/crifan/xxx/pyspider_migration/mongodb_migration
➜  mongodb_migration git:(master) ll storybook
total 416536
-rw-r--r--  1 crifan  staff    34M 11 26 11:58 lexile.bson
-rw-r--r--  1 crifan  staff   130B 11 26 11:58 lexile.metac
-rw-r--r--  1 crifan  staff   106M 11 26 11:58 main.bson
-rw-r--r--  1 crifan  staff   128B 11 26 11:58 main.metadat
-rw-r--r--  1 crifan  staff    62M 11 26 11:58 scholastic.b
-rw-r--r--  1 crifan  staff   134B 11 26 11:58 scholastic.m
➜  mongodb_migration git:(master) mongorestore -d storybook
2019-01-03T14:58:44.324+0800    checking for collection dat
2019-01-03T14:58:44.328+0800    reading metadata for storyb
2019-01-03T14:58:44.480+0800    restoring storybook.lexile
2019-01-03T14:58:45.166+0800    no indexes to restore
2019-01-03T14:58:45.166+0800    finished restoring storyboo
2019-01-03T14:58:45.166+0800    done
```

- 已有一个之前用mongodump备份出来的文件夹：evaluation，其中保存了整个evaluation的database的数据，将其恢复到此处本地的mongodb数据库

```
➜  mongodb mongorestore -d evaluation ./evaluation
2018-12-21T13:36:00.173+0800    the --db and --collection a
2018-12-21T13:36:00.175+0800    building a list of collecti
2018-12-21T13:36:00.176+0800    reading metadata for evalua

2018-12-21T13:36:00.176+0800    reading metadata for evalua
2018-12-21T13:36:00.176+0800    reading metadata for evalua
2018-12-21T13:36:00.274+0800    restoring evaluation.image.
2018-12-21T13:36:00.277+0800    reading metadata for evalua
2018-12-21T13:36:00.393+0800    restoring evaluation.audio.
2018-12-21T13:36:00.475+0800    restoring evaluation.questi
2018-12-21T13:36:00.571+0800    restoring evaluation.image.
2018-12-21T13:36:00.579+0800    restoring indexes for colle
2018-12-21T13:36:00.653+0800    finished restoring evaluati
2018-12-21T13:36:00.654+0800    reading metadata for evalua
2018-12-21T13:36:00.739+0800    restoring evaluation.audio.
2018-12-21T13:36:00.747+0800    no indexes to restore
2018-12-21T13:36:00.747+0800    finished restoring evaluati
2018-12-21T13:36:00.747+0800    restoring indexes for colle
2018-12-21T13:36:00.823+0800    finished restoring evaluati
2018-12-21T13:36:00.825+0800    restoring indexes for colle
2018-12-21T13:36:00.932+0800    finished restoring evaluati
2018-12-21T13:36:03.167+0800    [###############.........]
2018-12-21T13:36:04.849+0800    [########################]
2018-12-21T13:36:04.849+0800    restoring indexes for colle
2018-12-21T13:36:04.974+0800    finished restoring evaluati
2018-12-21T13:36:04.974+0800    done
```

导入后的MongoDB Compass中数据效果：



- 带指定用户名和密码的

```
[root@xxx-general-01 for_backup_mongodb]# mongorestore -h
2018-10-30T13:59:21.040+0800    building a list of collect
2018-10-30T13:59:21.041+0800    reading metadata for storyb
2018-10-30T13:59:21.041+0800    reading metadata for storyb
2018-10-30T13:59:21.061+0800    restoring storybook.main fr
2018-10-30T13:59:21.075+0800    restoring storybook.scholas
2018-10-30T13:59:22.567+0800    restoring indexes for colle
2018-10-30T13:59:22.567+0800    finished restoring storyboo
2018-10-30T13:59:22.629+0800    restoring indexes for colle
2018-10-30T13:59:22.629+0800    finished restoring storyboo
2018-10-30T13:59:22.629+0800    done
```

`mongoexport` 和 `mongoimport`

基本语法：

```
mongoimport -d database_name -c collection_name --file exported_mongodb_collection_file.json
```

举例

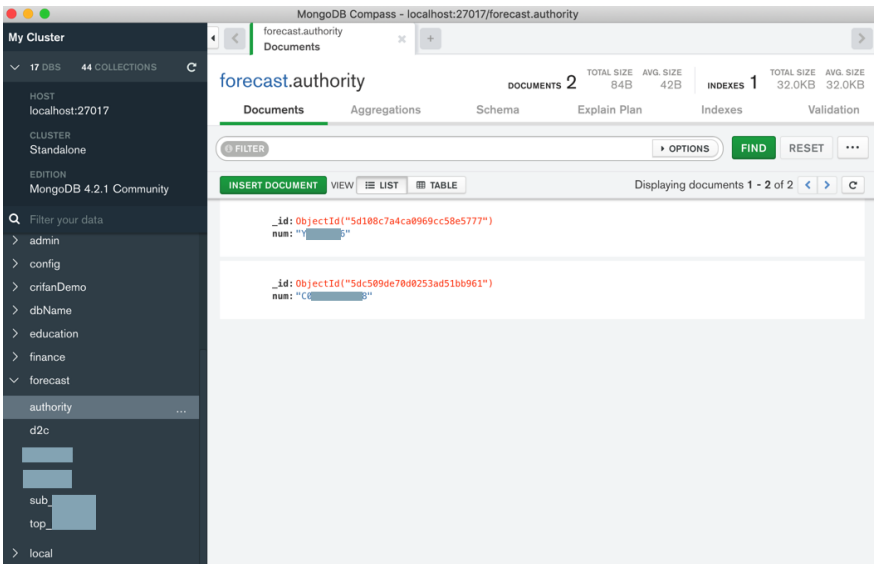
导出数据库 `evaluation` 的集合 `question`

```
mongoimport -d evaluation -c question --file questions_181219_2.json
```

导出和恢复authority

背景：

Mac中MongoDB中有如下数据：



然后去导出：

单个database数据库 `forecast` 中某个特定的collection集合 `authority` 的数据

```
mongoexport -d forecast -c authority -o forcast.authority_
```

具体log：

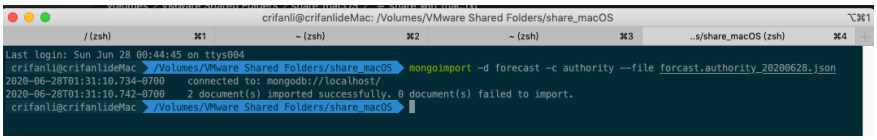
```
limao@xxx □ ~/Downloads/mongo_data □ mongoexport -d foreca
2020-06-28T16:26:15.600+0800    connected to: mongodb://loc
2020-06-28T16:26:15.614+0800    exported 2 records
limao@xxx □ ~/Downloads/mongo_data □ ll
total 8
-rw-r--r--  1 limao  CORP\Domain Users   126B  6 28 16:26 
```

然后把此 `json` 文件弄到 `Win` 中 `VMWare` 中的 `macOS` 之后，再去恢复：

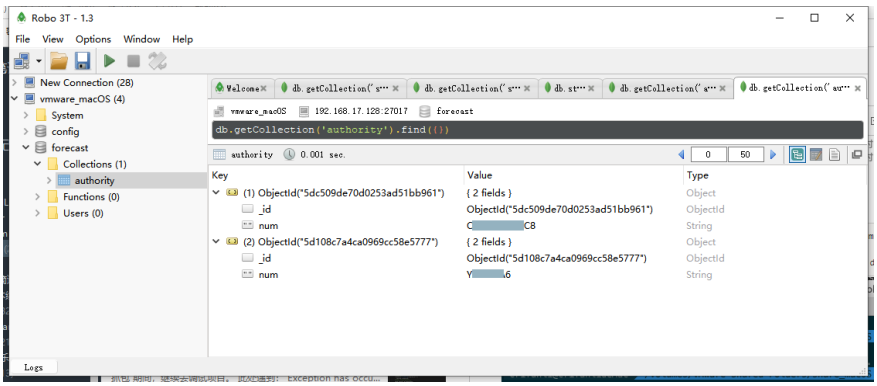
```
mongoimport -d forecast -c authority --file forcast.author
```

具体log：

```
crifanli@crifanlideMac □ /Volumes/VMware Shared Folders/sh
2020-06-28T01:31:10.734-0700    connected to: mongodb://loc
2020-06-28T01:31:10.742-0700    2 document(s) imported succ
```



Win中（能连接到macOS中MongoDB的）mongo的GUI客户端 `Robot 3T` 中，刷新后即可看到导入的数据：



新建空数据库和空集合

想要新建一个空的MongoDB的数据库，以及在空数据库中创建一个空的集合。

思路是：新建一个db，给db插入一个值，再删掉，就得到空的db的collection（和空的db数据库）了。

具体做法：

```
use newDb
db.newCollection.insert({"fakeKey": "fakeValue"})
db.newCollection.find() # 会输出刚插入的记录的_id
db.newCollection.deleteOne({"_id": ObjectId("newInsertedId'
```

然后用：

```
db.newCollection.find()
```

会看到输出是空的 -》 说明的确已经新建一个空的db和空的collection了。

提示：

在此期间可以随时用：

```
db
```

查看当前所处的数据库

和

```
show dbs
```

看看当前有哪些数据库

和：

```
show collections
```

看看当前数据库中有哪些集合

举例：新建一个db=storybook和collection=main

步骤：

```
> use storybook
switched to db storybook
> show dbs
admin    0.000GB
dialog   0.272GB
gridfs  13.869GB
local    0.000GB
> db.main.insert({"fakeKey": "fakeValue"})
WriteResult({ "nInserted" : 1 })
> show collections
main
> db
storybook
> db.main.find()
{ "_id" : ObjectId("5bd2bddb6ec5fdeabfbd6ecb"), "fakeKey"
> db.main.deleteOne({"_id": ObjectId("5bd2bddb6ec5fdeabfbd
{ "acknowledged" : true, "deletedCount" : 1 }
> db.main.find()
>
```

如何更新数据

对于已有一个MongoDB的记录，想要去更新其中的部分数据，有如下几种方式：

MongoDB shell

官网举例

```
db.col.update( { "count" : { $gt : 5 } } , { $set : { "test
```

举例：设置active为Y

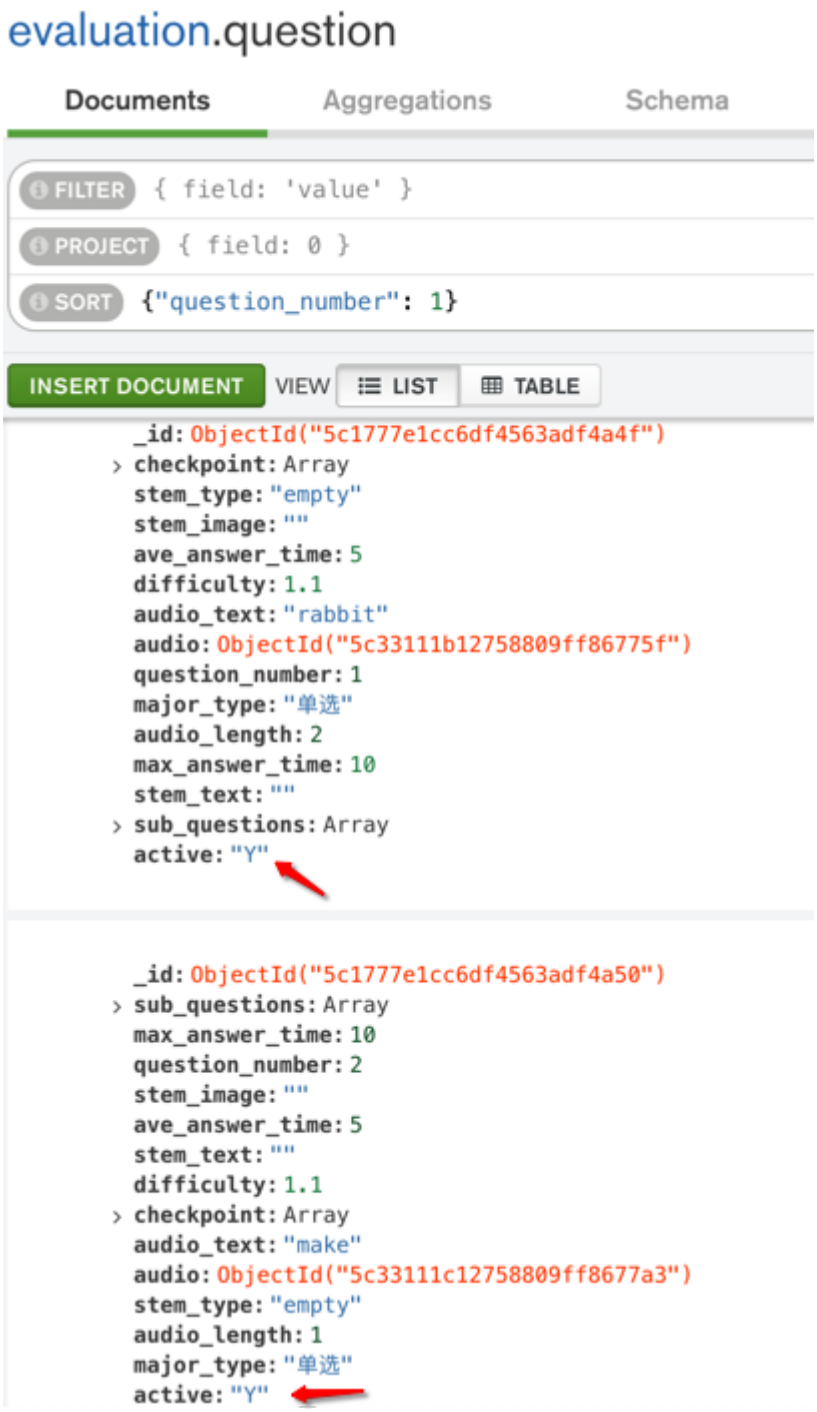
mongo shell中去插入新字段

```
db.question.update({}, {$set: {"active": "Y"}}, {upsert: fa
```

输出：

```
> db.question.update({}, {$set: {"active": "Y"}}, {upsert:
WriteResult({ "nMatched" : 883, "nUpserted" : 0, "nModifie
```

效果：



pymongo

通过PyMongo去更新某个记录的部分数据

举例：

```
        insertResult = mongoDstCollection.insert_one(dstBo
        logging.debug("insertResult=%s", insertResult)
        newDstRecordId = insertResult.inserted_id
        logging.debug("newDstRecordId=%s", newDstRecordId)
        # update inserted id into existing old source reco
        mergedId = ""
        if newDstRecordId:
            mergedId = str(newDstRecordId)
        logging.debug("mergedId=%s", mergedId) # '5bd28426
        updateResult = mongoSrcCollection.update_one(filte
        logging.debug("updateResult=%s", updateResult)
        matched_count = updateResult.matched_count
        modified_count = updateResult.modified_count
        logging.debug("matched_count=%s, modified_count=%s'
```

Python

PyMongo

更新单条数据的整个值

MongoDB中，想要Python的pymongo中，实现（除了_id不变外的）单个项目，单条数据的整个值的替换：

代码：

```
            mongoCollection.update_one(
                {"_id": oldMongoId},
                {'$set': newValueDict},
                upsert=False,
            )
```

举例：

```
eachFailedItem = mongoCollectionShortlink.find(queryDict)

oldMongoId = eachFailedItem.pop("_id", None)

mongoCollectionShortlink.update_one(
    {"_id": oldMongoId},
    {'$set': newRespResult},
    upsert=False,
)
```

操作collection

给collection改名

`mongo shell` 或代码（比如 `Python` 的 `Pymongo` ）语法：

```
db.yourCollection.renameCollection("newCollectionName")
```

举例：

```
db.gameShortLink_20210816.renameCollection("gameShortLink_
```

注： `MongoDB Compass` 中不支持collection改名

高级搜索

下面介绍在搜索和查询想要的记录时，涉及到的一些高级搜索技巧，尤其是正则搜索和嵌套搜索。

搜索期间的条件组合

背景：搜索数据期间，可能会用到多种条件，不同条件，可能需要组合在一起。

对于组合的种类和语法，详见官网：

Operators — MongoDB Manual

->

Query and Projection Operators — MongoDB Manual

先列出有哪些类型：

- Comparison
  - $eq
    - Matches values that are equal to a specified value.
  - $gt
    - Matches values that are greater than a specified value.
  - $gte
    - Matches values that are greater than or equal to a specified value.
  - $in
    - Matches any of the values specified in an array.
  - $lt
    - Matches values that are less than a specified value.
  - $lte
    - Matches values that are less than or equal to a specified value.
  - $ne
    - Matches all values that are not equal to a specified value.
  - $nin
    - Matches none of the values specified in an array.
- Logical
  - $and
    - Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
  - $not

        - Inverts the effect of a query expression and returns documents that do not match the query expression.
    - $nor
        - Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
    - $or
        - Joins query clauses with a logical OR returns all documents that match the conditions of either clause.
- Element
    - $exists
        - Matches documents that have the specified field.
    - $type
        - Selects documents if a field is of the specified type.
- Evaluation
    - $expr
        - Allows use of aggregation expressions within the query language.
    - $jsonSchema
        - Validate documents against the given JSON Schema.
    - $mod
        - Performs a modulo operation on the value of a field and selects documents with a specified result.
    - $regex
        - Selects documents where values match a specified regular expression.
    - $text
        - Performs text search.
    - $where
        - Matches documents that satisfy a JavaScript expression.
- Geospatial
    - $geoIntersects
        - Selects geometries that intersect with a GeoJSON geometry. The 2dsphere index supports$geoIntersects.
    - $geoWithin
        - Selects geometries within a bounding GeoJSON geometry. The 2dsphere and 2d indexes support $geoWithin.
    - $near
        - Returns geospatial objects in proximity to a point. Requires a geospatial index. The 2dsphereand 2d indexes support $near.
    - $nearSphere
        - Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The 2dsphere and 2d indexes support $nearSphere.

- Array
  - $all
    - Matches arrays that contain all elements specified in the query.
  - $elemMatch
    - Selects documents if element in the array field matches all the specified $elemMatch conditions.
  - $size
    - Selects documents if the array field is a specified size.
- Bitwise
  - $bitsAllClear
    - Matches numeric or binary values in which a set of bit positions all have a value of 0.
  - $bitsAllSet
    - Matches numeric or binary values in which a set of bit positions all have a value of 1.
  - $bitsAnyClear
    - Matches numeric or binary values in which any bit from a set of bit positions has a value of 0.
  - $bitsAnySet
    - Matches numeric or binary values in which any bit from a set of bit positions has a value of 1.
- Comments
  - $comment
    - Adds a comment to a query predicate.
- Projection Operators
  - $
    - Projects the first element in an array that matches the query condition.
  - $elemMatch
    - Projects the first element in an array that matches the specified $elemMatch condition.
  - $meta
    - Projects the document’s score assigned during $text operation.
  - $slice
    - Limits the number of elements projected from an array. Supports skip and limit slices.

可以根据需要选择合适的组合方式。

举例：

```
db.inventory.find( { $or: [ { quantity: { $lt: 20 } }, { pr
```

和：

```
var cursor = db.collection('inventory').find({
  $or: [ {status: "A" }, { qty: { $lt: 30 } } ]
});
```

列表子元素搜索

如果是搜索list中所有的元素，即不关心list中具体是第几个元素，那么就直接写成：

```
mainField.subFieldList
```

如果要关心具体是哪个位置的元素的值，则使用：`.N`

比如 第一个是

```
mainField.subFieldList.0
```

而更加复杂的组合，可以用到：

- `$size`
  - 表示list的个数
- `$all`
  - 表示所有的
- `$elemMatch`

即可。

更多语法细节可参考：

Array Query Operators — MongoDB Manual

字段嵌套搜索

用 **点** `.` 实现字段的嵌套的搜索：

举例：搜索子字段sub_questions的options的option_text

对于内容：

```
{
  "_id": "5c1777e1cc6df4563adf4a5c",
  "max_answer_time": 20,
  "audio": "5c33111d12758809ff867931",
  "stem_type": "mix",
  "sub_questions": [{
    "option_type": "text",
    "correct_option": [3],
    "question_texts": [""],
    "options": [{
      "option_index": 1,
      "option_text": "lean",
      "option_image": ""
    }, {
      "option_index": 2,
      "option_text": "less",
      "option_image": ""
    }, {
      "option_index": 3,
      "option_text": "lesson",
      "option_image": ""
    }]
  }],
。。。
```

想要通过子字段 `sub_questions` 的 `options` 的 `option_text` 之类的嵌套字段去搜索

写成：

```
sub_questions.options.option_text
```

即可。

支持正则查询

举例：

MongoDB Compass

举例：搜索AD开头的lexile

MongoDB Compass中，想要用正则搜索字段：

```
grading
    lexile: "AD450L"
```

写法是：

```
{"grading.lexile": {$regex: "AD.*"}}
```

或：regex加上行首和行尾判断：

```
{"grading.lexile": {$regex: "^AD.*$"}}
```

或 regex用引号引起来

```
{"grading.lexile": {"$regex": "AD.*"}}
```

注：

另外试了试：

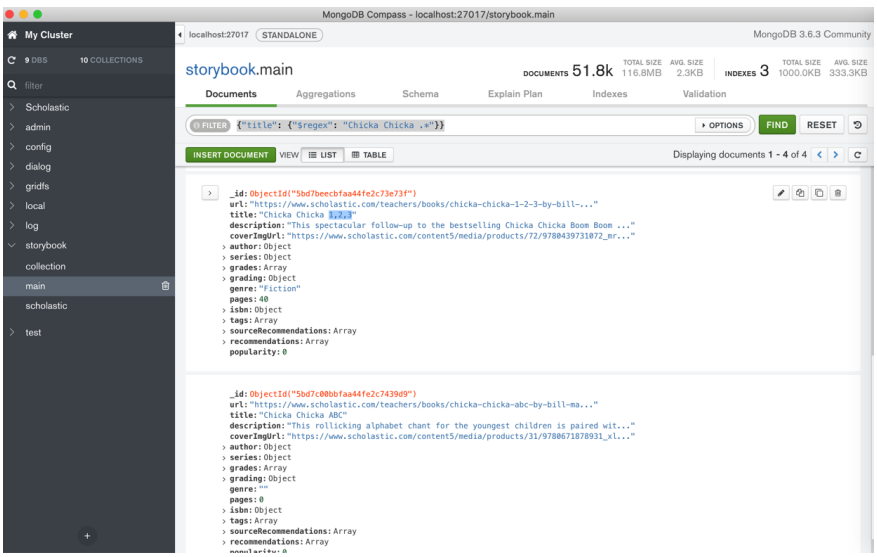
```
{"grading.lexile": {$regex: "^AD\d+.*$"}}
```

发现搜索不到，所以结论是：此处正则搜索不支持 `\d` 数字

举例：用正则语法去搜索title

```
{"title": {"$regex": "Chicka Chicka .*"}}
```

可以搜到匹配的多个数据：



举例：多个字段同时正则搜索+嵌套搜索+列表字段搜索

此处去实现一个更加复杂的：

- 正则搜索
- 多个条件组合搜索

- 字段嵌套搜索
- 列表字段搜索

要搜索的内容：

```
{
    "_id": "5c1777e1cc6df4563adf4b3c",
    "audio": "http://10.108.133.251:34800/audio/5c33111e12
    "audio_length": 29,
    "audio_text": "f: Hi. I'm Tania. What's your name? m: H
    "ave_answer_time": 70,
    "checkpoint": [
        73,
        83,
        "对话和应答"
    ],
    "difficulty": 3.2,
    "major_type": "单选多题",
    "max_answer_time": 140,
    "question_number": 238,
    "stem_image": "http://10.108.133.251:34800/image/5c32f0
    "stem_text": "",
    "stem_type": "mix",
    "sub_questions": [
        {
            "correct_option": [
                1
            ],
            "option_type": "text",
            "options": [
                {
                    "option_image": "",
                    "option_index": 1,
                    "option_text": "Yes"
                },
                {
                    "option_image": "",
                    "option_index": 2,
                    "option_text": "No"
                }
            ],
            "question_texts": [
                "Jing and Tania are in the same class."
            ]
        },
        {
            "correct_option": [
                1
            ],
            "option_type": "text",
            "options": [
                {
                    "option_image": "",
                    "option_index": 1,
                    "option_text": "Yes"
```

```
                },
                {
                    "option_image": "",
                    "option_index": 2,
                    "option_text": "No"
                }
            ],
            "question_texts": [
                "Their teacher is a man."
            ]
        },
        {
            "correct_option": [
                2
            ],
            "option_type": "text",
            "options": [
                {
                    "option_image": "",
                    "option_index": 1,
                    "option_text": "Yes"
                },
                {
                    "option_image": "",
                    "option_index": 2,
                    "option_text": "No"
                }
            ],
            "question_texts": [
                "Jing knows where the classroom is."
            ]
        }
    ]
},
```

想要搜索字段：

- stem_text
- audio_text
- sub_questions的question_texts（这个列表）中的任何一个
- sub_questions的options（这个列表中的）任何一个的option_text

包含了teacher的话

则搜索条件可以写成：

```
{
  '$or': [{
      'stem_text': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    },
    {
      'audio_text': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    },
    {
      'sub_questions.question_texts': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    },
    {
      'sub_questions.options.option_text': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    }
  ]
}
```

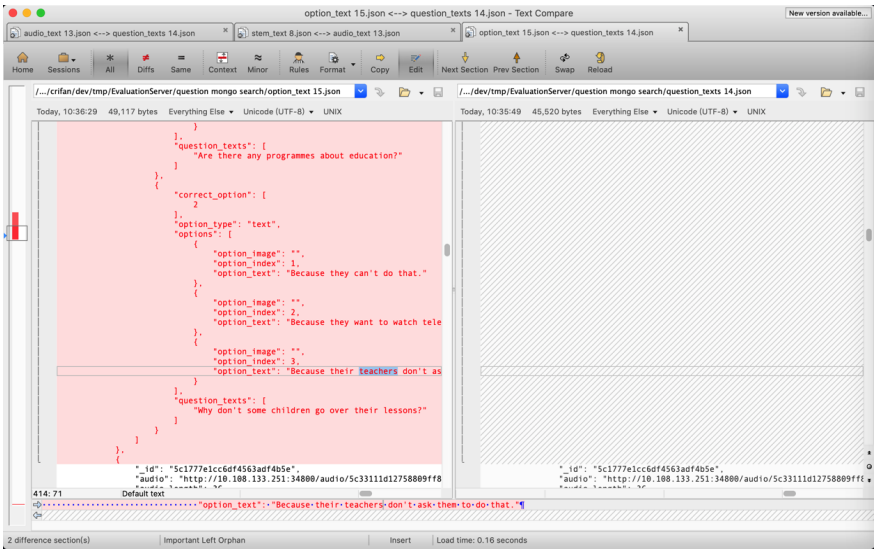
对应代码：

```
            findParam["$or"] = [
                {"stem_text": {"$regex": searchText, "$optio
                {"audio_text": {"$regex": searchText, "$opti
                {"sub_questions.question_texts": {"$regex": 
                {"sub_questions.options.option_text": {"$rege
            ]
```

相关部分完整代码：

```
        findParam = {}

        searchText = parsedArgs["searchText"]
        if searchText:
            findParam["$or"] = [
                {"stem_text": {"$regex": searchText, "$optio
                {"audio_text": {"$regex": searchText, "$opti
                {"sub_questions.question_texts": {"$regex": 
                {"sub_questions.options.option_text": {"$rege
            ]

        majorType = parsedArgs["majorType"]
        if majorType:
            findParam["major_type"] = majorType

        sortBy = "question_number"
        log.debug("findParam=%s", findParam)
        sortedQuestionsCursor = questionCollection.find(findP
        totalCount = sortedQuestionsCursor.count()
```

即可搜索到要的内容：



举例：搜索题干类型题目中包含teach的题目

举例另一个复杂的例子：

代码：

```
findParam = {}

majorType = parsedArgs["majorType"]
if majorType:
    findParam["major_type"] = majorType

stemType = parsedArgs["stemType"]
if stemType:
    findParam["stem_type"] = stemType

checkpointList = parsedArgs["checkpointList"]
if checkpointList:
    checkpointList = checkpointList.split(",")
    checkpointOrParamList = []
    for eachCheckpoint in checkpointList:
        curCheckpointRegex = {"checkpoint": {"$regex": eacl
        checkpointOrParamList.append(curCheckpointRegex)
    checkpointAndParamList = [ { "$or": checkpointOrParamLi
    if "$and" in findParam:
        findParam["$and"].extend(checkpointAndParamList)
    else:
        findParam["$and"] = checkpointAndParamList

searchText = parsedArgs["searchText"]
if searchText:
    searchTextOrParamList = [
        {"stem_text": {"$regex": searchText, "$options": "
        {"audio_text": {"$regex": searchText, "$options": '
        # {"sub_questions.option_type": "text"},
        {"sub_questions.question_texts": {"$regex": searchT
        {"sub_questions.options.option_text": {"$regex": se
        # {"sub_questions.options.option_image": {"$regex"
    ]

    searchTextAndParamList = [{ "$or": searchTextOrParamLis
    if "$and" in findParam:
        findParam["$and"].extend(searchTextAndParamList)
    else:
        findParam["$and"] = searchTextAndParamList


sortBy = "question_number"
log.debug("findParam=%s", findParam)
sortedQuestionsCursor = questionCollection.find(findParam)
```

生成的相关搜索条写法是：

```
{
  'major_type': '单选多题',
  'stem_type': 'mix',
  '$and': [{
    '$or': [{
      'checkpoint': {
        '$regex': '对话',
        '$options': 'im'
      }
    }, {
      'checkpoint': {
        '$regex': '理解',
        '$options': 'im'
      }
    }]
  }, {
    '$or': [{
      'stem_text': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    }, {
      'audio_text': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    }, {
      'sub_questions.question_texts': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    }, {
      'sub_questions.options.option_text': {
        '$regex': 'teacher',
        '$options': 'im'
      }
    }]
  }]
}
```

即可查到需要的内容。

对搜索结果排序

对于搜索出的结果，想要针对某个或某些键去排序，则：

- 总体逻辑
  - 给 `sort` 传入 `key` 和**排序方向**
    - 排序方向值

- 顺序：`1`
- 倒序：`-1`

Python的API代码

语法：

```
collection.find(xxx).sort("key", pymongo.ASCENDING)
```

举例：不区分大小写找包含sleep的文件名

```
findFileCursor = fsCollection2.find({"filename": {"$regex"
```

举例：搜索结果根据question_number排序

举例：

```
        sortBy = "question_number"
        sortedQuestionsCursor = questionCollection.find(findI
```

工具

`MongoDB shell` 或其他GUI工具（比如 `Robot 3T` ）中是：

```
collection.find(xxx).sort({"key": 1})
```

举例：

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

GridFS存储文件

MongoDB针对于文件，尤其是大文件，专门设计了一套接口，叫做：GridFS

此处总结一下GridFS的使用心得。

TODO：

- 如何基于Flask搭建一个文件下载服务，甚至可以支持断点续传功能。把之前相关代码贴过来。
- 把更多帖子内容整理至此
  - GridFS
    - 【已解决】GridFS保存文件时如何得到文件的id或_id
    - 【已解决】用Python去连接本地mongoDB去用GridFS保存文件
    - 【已解决】MongoDB的GridFS中基于文件名或id去下载文件
    - 【已解决】MongoDB的GridFS中只返回file的chunks的个数而不返回chunks.data
    - 【基本解决】Python中把wma、wav等格式音频转换为mp3
    - 【未解决】Mongo中更新gridfs中的mp3文件的metadata信息且尽量保持id不变
    - 【已解决】远程的mongoDB中GridFS报错：AttributeError GridFS object has no attribute totalSize
    - 【规避解决】Flask-PyMongo中如何查询gridfs中的文件
    - 【已解决】如何用PyMongo中的GridFS的put去保存添加文件
    - 【已解决】PyMongo中GridFS的exists始终检测不到文件已存在
    - 【已解决】Flask中Mongo的GridFS数据如何保存为绝对路径的下载文件地址
    - 【已解决】MongoDB通过GridFS的API的put保存文件时添加额外信息
    - 【已解决】PyMongo的GridFS中使用fs的collection去put出错：TypeError Collection object is not callable
    - 【已解决】把本地mp3文件存入在线Mongo中且填写meta信息
    - 【已解决】Flask中连接远程MongoDB数据库的gridfs并返回查询到的文件数据
  - 和其中mongofiles相关的：
    - 【已解决】用mongofiles去删除GridFS中的文件
    - 【无法也无须解决】用mongofiles给GridFS中添加文件时添加额外参数属性字段

        - 【已解决】mongofiles中put保存和get下载获取时指定文件名

一些供参考的资料：

- GirdFS官网文档
  - GridFS — MongoDB Manual 3.6
- Pymongo
  - GridFS的例子
    - GridFS Example — PyMongo 3.6.1 documentation
  - GridFS的API
    - gridfs – Tools for working with GridFS — PyMongo 3.6.1 documentation
  - 相关的grid_file
    - grid_file – Tools for representing files stored in GridFS — PyMongo 3.6.1 documentation

一些GirdFS的心得

通过id查找gridfs中file文件

注意不是：

```
> db.fs.files.find({"id": "5b556ee47f4d38ba6a189222"})
> db.fs.files.find({"_id": "5b556ee47f4d38ba6a189222"})
```

而是 `ObjectId("5b556ee47f4d38ba6a189222")` ：

```
> db.fs.files.find({"_id": ObjectId("5b556ee47f4d38ba6a189
{ "_id" : ObjectId("5b556ee47f4d38ba6a189222"), "contentTy
>
```

删除文件

举例：

```
> db.fs.files.deleteOne({"_id": ObjectId("5b556ee47f4d38ba
{ "acknowledged" : true, "deletedCount" : 1 }
```

格式化带缩进美观的输出结果

后面加上： `.pretty()`

命令行shell

```
> db.fs.files.find({"_id": ObjectId("5b556ee47f4d38ba6a1892
{
    "_id" : ObjectId("5b556ee47f4d38ba6a189222"),
    "contentType" : "audio/mpeg",
    "chunkSize" : 261120,
    "metadata" : {
        "song" : {
            "singers" : [
                "ChuChu TV"
            ]
        },
        "series" : {
            "number" : 0,
            "name" : ""
        },
        "topics" : [ ],
        "storybook" : {
            "publisher" : "",
            "isFiction" : "未知",
            "lexileIndex" : "",
            "awards" : "",
            "authors" : [ ],
            "foreignCountry" : ""
        },
        "keywords" : {
            "fromName" : [
                "Finger",
                "Family"
            ],
            "other" : [ ],
            "fromContent" : [
                "Daddy Finger"
            ]
        },
        "name" : "Finger Family",
        "resourceType" : "song",
        "mainActors" : [ ],
        "contentAbstract" : "",
        "isSeries" : false,
        "fitAgeStart" : 3,
        "fitAgeEnd" : 6,
        "fileInfo" : {
            "isAudio" : true,
            "contentType" : "audio/mpeg",
            "name" : "Finger Family.mp3",
            "suffix" : "mp3"
        }
    },
    "filename" : "Finger Family.mp3",
    "length" : 2604280,
```

```
    "uploadDate" : ISODate("2018-07-23T06:00:04.925Z"),
    "md5" : "946564effc4c0a835c55564377e7d819"
}
>
```

查看当前MongoDB信息

想要查看当前MongoDB信息，即如何得知mongod的 启动文件、 配置文件、log日志文件、数据文件路径等等是什么

经过研究，可以用如下方法：

可以通过：

```
systemctl status mongod
```

中看到启动文件，比如：

```
[root@xxx-general-01 ~]# systemctl status mongod
[0m mongod.service - SYSV: Mongo is a scalable, document-or
   Loaded: loaded (/etc/rc.d/init.d/mongod; bad; vendor pre
   Active: active (running) since Tue 2018-04-10 15:37:29 C
     Docs: man:systemd-sysv-generator(8)
   CGroup: /system.slice/mongod.service
           1096 /usr/bin/mongod -f /etc/mongod.conf

Apr 10 15:37:28 xxx-general-01 systemd[1]: Starting SYSV: M
Apr 10 15:37:28 xxx-general-01 runuser[1077]: pam_unix(runu
Apr 10 15:37:29 xxx-general-01 runuser[1077]: pam_unix(runu
Apr 10 15:37:29 xxx-general-01 mongod[1058]: Starting mongo
Apr 10 15:37:29 xxx-general-01 systemd[1]: Started SYSV: Mo
```

中的：

`/etc/rc.d/init.d/mongod`

是启动脚本

然后再去查看此启动脚本的内容，可以找到配置文件：

```
[root@xxx-general-01 ~]# cat /etc/rc.d/init.d/mongod
#!/bin/bash

# mongod - Startup script for mongod

# chkconfig: 35 85 15
# description: Mongo is a scalable, document-oriented datab
# processname: mongod
# config: /etc/mongod.conf

. /etc/rc.d/init.d/functions

# NOTE: if you change any OPTIONS here, you get what you pa
# this script assumes all options are in the config file.
CONFIGFILE="/etc/mongod.conf"
OPTIONS=" -f $CONFIGFILE"

mongod=${MONGOD-/usr/bin/mongod}

MONGO_USER=mongod
MONGO_GROUP=mongod
...
```

中的：

`/etc/mongod.conf`

然后再从配置文件中看到，对应的log日志，数据文件等信息：

命令行shell

```
[root@xxx-general-01 ~]# cat /etc/mongod.conf
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# Where and how to store data.
storage:
  dbPath: /var/lib/mongo
  journal:
    enabled: true
#  engine:
#  mmapv1:
#  wiredTiger:

# how the process runs
processManagement:
  fork: true  # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid  # location of p

# network interfaces
net:
  port: 32018
  bindIp: 127.0.0.1,172.16.141.197 # Listen to specific IP
#  bindIp: 127.0.0.1  # Listen to local interface only, cor
#  bindIp: 0.0.0.0  # Listen to all interfaces
#  port: 27017

security:
  authorization: 'enabled'

#operationProfiling:

#replication:

#sharding:

## Enterprise-Only Options

#auditLog:

#snmp:
```

中，找到：

- 日志文件：`/var/log/mongodb/mongod.log`
- 数据文件（路径）：`/var/lib/mongo`

从当前运行的MongoDB找到conf配置路径

- 前提：当前mongodb（服务端 `mongod` ）正在运行
- 目的：想要找到系统中MongoDB的配置文件所在位置
  - 配置文件一般是 `mongod.conf`
- 解决方案
  - 通过查看进程详情中可以看到conf配置文件路径
    - 举例

```
crifanli@crifanlideMac □ / □ ps aux | grep mong
crifanli          8712   0.0  0.0  4258648    2
crifanli          8676   0.0  1.3  5524376  403
```

      - 对应的是：
        - `/usr/local/etc/mongod.conf`

连接远程MongoDB的方式

背景：

在远程阿里云ECS服务器中有个MongoDB数据库：

- IP：`47.96.131.109`
- Port：`32018`

其中有个：

- gridfs数据库
  - 用户名：gridfs
  - 角色：dbOwner

且服务端mongod正在运行了

下面是如何去连接该远程的mongo数据库几种方式：

服务器中本地连接

服务器上本地客户端：`mongo shell`

```
[root@xxx-general-01 ~]# mongo --port 32018
MongoDB shell version: 3.2.19
connecting to: 127.0.0.1:32018/test
```

进去后再：

```
use gridfs
db.auth("gridfs", "password")
```

或者直接：

```
mongo gridfs --port 32018 -u gridfs -p password --authentic
```

Mac本地连接远端MongoDB

Mac中本地`mongo shell`去连接远程MongoDB

以gridfs用户去登录

以用户gridfs去登录，且（限定了）只（能）访问数据库gridfs：

```
mongo 47.96.131.109:32018/gridfs  -u gridfs -p password --a
```

或：

```
mongo gridfs --host 47.96.131.109 --port 32018  -u gridfs -
```

以admin用户去登录

以用户admin去登录，没有限定访问哪个数据库（后续则可以访问其他数据，前提是admin本身有这个权限）：

```
mongo --host 47.96.131.109 --port 32018 -u root -p pwd --au
```

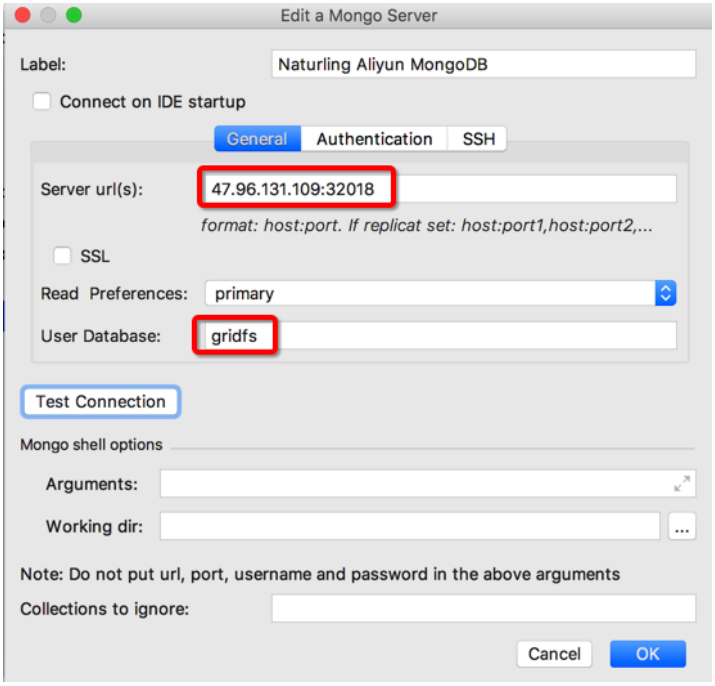
Python的pymongo代码连接远程MongoDB

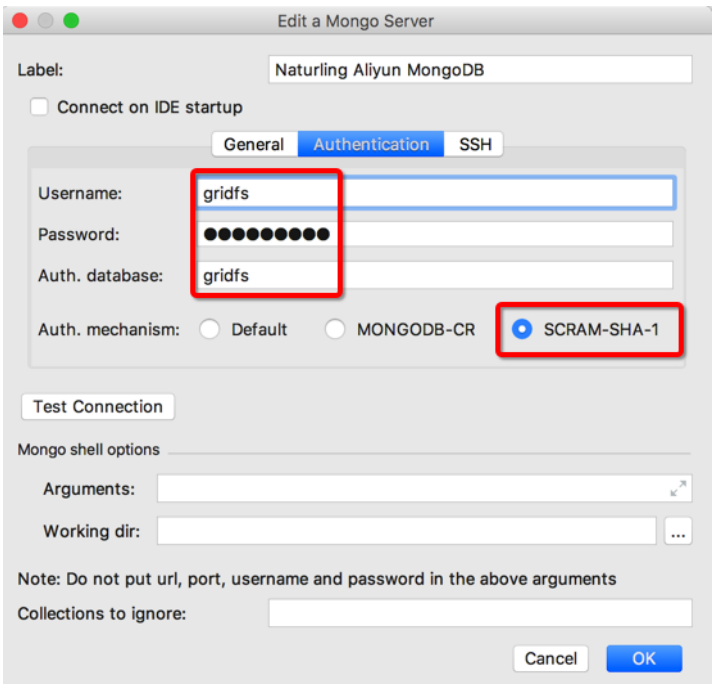
```
import pymongo
from pymongo import MongoClient
import gridfs

# from pymongo.objectid import ObjectId
# from pymongo import objectid
from bson.objectid import ObjectId

from gridfs import GridFS

MongoHost = "47.96.131.109"
MongoPort = 32018

MongoUseAuth = True
# MongoUseAuth = False

# with auth
MongoUsername = "gridfs"
MongoPassword = “password"
MongoAuthenticationDatabase = "gridfs"

mongodbUri = ""
if MongoUseAuth :
    mongodbUri = "mongodb://%s:%s@%s:%s/%s" % (
        quote_plus(MongoUsername), \
        quote_plus(MongoPassword), \
        MongoHost, \
        MongoPort, \
        MongoAuthenticationDatabase \
    )
    #'mongodb://gridfs:password@47.96.131.109:32018/gridfs
else:
    mongodbUri = "mongodb://%s:%s" % (
        MongoHost, \
        MongoPort
    )
    #'mongodb://localhost:32018'
    #'mongodb://47.96.131.109:32018’

mongoClient = MongoClient(mongodbUri)
gridfsDb = mongoClient.gridfs
fsCollection = GridFS(gridfsDb)
```

PyCharm中mongo4idea中的连接远端MongoDB

配置参数：

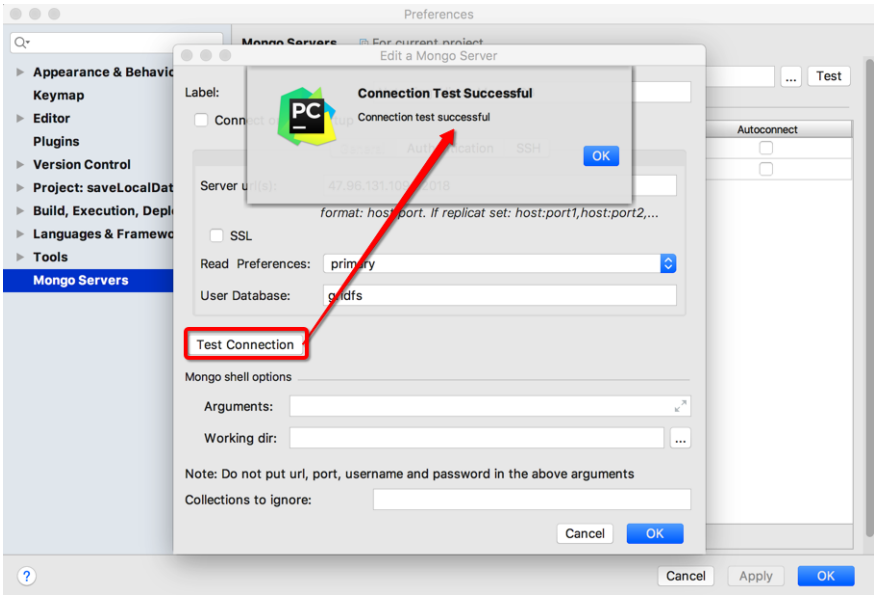
- General

    - 截图：
      - 

    - 文字：
      - Server url(s)：`47.96.131.109:32018`
      - User Database：`gridfs`
- Authentication
    - 截图：
      - 

    - 文字：
      - Username：`gridfs`
      - Password：`your_password`
      - Auth. database：`gridfs`
      - Auth. mechanism：`SCRAM-SHA-1`

即可成功连接：



坑

此处整理一些常见的MongoDB的坑及解决办法。

停止MongoDB

想要去停止MongoDB的服务端。

对于之前是正常安装方式

```
brew install mongodb-community
```

则去停止运行方式是

```
brew services stop mongodb-community
```

不过之前遇到过特殊情况，试了各种方式都无法直接关闭掉，经研究最后是用：

```
sudo mongod -f /etc/mongod.conf --shutdown
```

输出：

```
killing process with pid: 30213
```

才算真正的关闭了MongoDB。

详见：

【已解决】CentOS中如何彻底真正关闭mongod的服务

Cursor not found

用pymongo期间可能会报错：

```
pymongo.errors.CursorNotFound: Cursor not found
```

原因：

此处的：

```
for eachDialog in dialogCollection.find():
```

find会返回多个数据，而这些数据的处理时间，超过了默认最长时间**10分钟**，所以超时报错了。

解决办法：

去掉超时时间的设置，加上参数 `no_cursor_timeout=True`

代码改为：

```
# for eachDialog in dialogCollection.find():
cursor = dialogCollection.find(no_cursor_timeout=True)
for eachDialog in cursor:
    yield eachDialog
cursor.close()
```

即可。

参数_id是ObjectId对象而不是字符串

直接说具体的坑是：Pymongo中的很多函数，比如exists等，传入的参数id是ObjectId对象而不是id的字符串

发现并解决坑的具体过程：

此处，gridfs的官网文档：

gridfs – Tools for working with GridFS — PyMongo 3.6.1 documentation

> exists(document_or_id=None, session=None, **kwargs)

函数来说，没有说明 `document_or_id` 是个特殊的 `ObjectId` 的这个类的实例，而其例子：

```
>>> fs.exists(file_id)
>>> fs.exists({"_id":file_id})
>>> fs.exists(_id=file_id)
>>> fs.exists({"filename": "mike.txt"})
>>> fs.exists(filename="mike.txt")
```

很容易让人误解就是普通的_id的值的字符串，比如：

`"5abc96dfa4bc715f473f0297"`

或

`"ObjectId('5abc96dfa4bc715f473f0297')"`

搞得尝试半天也无法正常执行，找不到本来已存在的文件。

幸亏：

django - Querying by "_id" doesn't return file in MongoDB in Python, using PyMongo - Stack Overflow

中解释的，对于pymongo

2.2版本之前：

```
from pymongo.objectid import ObjectId
```

2.2版本之后：

```
from bson.objectid import ObjectId
```

然后才能用于exists：

```
fileIdToDelete = '5abc8d77a4bc71563222d455'
fileObjectIdToDelete = ObjectId(fileIdToDelete)
if fsCollection.exists(fileObjectIdToDelete):
    fsCollection.delete(fileObjectIdToDelete)
```

并且注意到作者是2012年，6年前就回复了该答案，结果此处pymongo在6年后，都没有及时更新此内容，真是醉了。

希望mongodb的文档以后能及时更新啊。

不要在admin中创建普通用户

创建用户时注意不要在admin数据库中创建：

如果不小心是在最开始的 `admin` 中去用 `db.createUser` 新建的普通用户，则实际上新建的用户只是属于 `admin` 数据库的。

可以用admin账号登录后，通过如下命令看到：

```
> use admin
switched to db admin
> show users
{
        "_id" : "admin.root",
        "user" : "root",
        "db" : "admin",
        "roles" : [
                {
                        "role" : "root",
                        "db" : "admin"
                }
        ]
}
{
        "_id" : "admin.log",
        "user" : "log",
        "db" : "admin",
        "roles" : [
                {
                        "role" : "dbOwner",
                        "db" : "log"
                }
        ]
}
```

所以要切换到对应新数据库中，再去创建才可以。

否则就会导致后续没有权限操作其下数据。

附录

下面列出相关参考资料。

教程和文档

此处整理一些关于MongoDB的官方和非官网的各种有用资料，供需要时查询和参考。

MongoDB官网资料

官方文档入口：

The MongoDB 4.0 Manual — MongoDB Manual

官网还有个系列教程：

- MongoDB University

部分细节内容：

- 更新数据
  - db.collection.update() — MongoDB Manual
- 查询
  - Query Documents — MongoDB Manual
  - Query on Embedded/Nested Documents — MongoDB Manual
  - Query an Array — MongoDB Manual
  - Query an Array of Embedded Documents — MongoDB Manual
- 查询期间条件组合
  - $or — MongoDB Manual
  - Read Data using Operators and Compound Queries
- 正则搜索
  - $regex
- 列表查询
  - Array Query Operators — MongoDB Manual
- 聚合
  - Aggregation — MongoDB Manual 3.4
- Driver的
  - MongoDB Drivers and Client Libraries — MongoDB Manual 3.6
- 用户和权限认证
  - Authentication
    - Authentication — MongoDB Manual 3.6
    - authenticate — MongoDB Manual 3.6
  - Users
    - Users — MongoDB Manual 3.6
  - 权限
    - RBAC
      - Role-Based Access Control — MongoDB Manual 3.6

        - 角色
            - Built-In Roles — MongoDB Manual 3.6
            - User-Defined Roles — MongoDB Manual 3.6
            - grantRolesToUser — MongoDB Manual 3.6
    - Resource
        - Resource Document — MongoDB Manual 3.6
    - 新建用户
        - createUser — MongoDB Manual 3.6

第三方资料

官网的（部分完成的）中文翻译：

- mongoing.com
    - MongoDB 4.2 手册
        - Documentation | MongoDB中文社区
    - MongoDB 3.4 手册
        - 安装MongoDB — MongoDB Manual 3.4
- 极客学院
    - mongodb 数据库 (1) - 《从零开始学 Python》(第二版) - 极客学院Wiki

Mongo Shell资料

Mongo Shell的资料：

- 英文
    - mongo Shell Quick Reference — MongoDB Manual 3.6
- 中文
    - mongo shell — MongoDB Manual 3.4
        - 注：翻译未必是完整的，且不一定是最新版本

参考资料

- 【已解决】MongoDB的Pymongo中如何更新单条纪录的整个值
- 【已解决】MongoDB版本区别：Community vs Enterprise
- 【记录】mac搭建开发环境：安装最新版MongoDB
- 【已解决】从MongoDB数据库中导出数据
- 【已解决】把json数据恢复导入到本地MongoDB数据库的某个集合中
- 【已解决】重新将本地MongoDB数据storybook导入到在线环境
- 【记录】把在线的dev的MongoDB备份后恢复到本地
- 【已解决】PySpider项目迁移到别的电脑重新继续运行
- 【已解决】确认服务器是否已经正在运行MongoDB的服务mongod
- 【已解决】去更新绘本查询中的MongoDB中的绘本的title字段数据
- 【已解决】MongoDB中更新现有记录新插入字段
- 【已解决】Mongo的shell中新建一个空的db数据库和collection集合
- 【已解决】Python的MongoDB的Pymongo中实现嵌套字段即子字段的搜索
- 【已解决】Python的MongoDB的Pymongo中实现list列表中的内容的搜索
- 【已解决】Python的MongoDB的Pymongo中实现多个正则条件的组合搜索
- 【已解决】Python的MongoDB的pymongo中搜索查询支持regex正则和多个条件组合和字段嵌套和列表字段
- 【已解决】Python的MongoDB的Pymongo中搜索列表字段中是否包含某个列表值中任何一个以及正则匹配部分内容
- 【已解决】Win中Robot 3T访问VMWare中macOS中MongoDB报错：Network is unreachable Reason couldn't connect to server connection attempt failed SocketException
- 【已解决】Win中尝试用Robot 3T连接和操作VMWare中macOS中MongoDB
- 【规避解决】MongoDB Compass中如何给集合Collection改名
-
- 【已解决】Python的Flask中pymysql中mysql返回分页查询结果
- 【已解决】Pymongo中新增插入记录
- 【已解决】Python的pymongo中根据指定字段的时间范围去查询数据且排序
- 【或许解决】Python的Pymongo中gridfs文件去更新保存metadata信息
- 【已解决】mongoimport/mongoexport和mongodump/mongorestore的区别
- 【已解决】Mongo中让搜索支持不区分大小写

- 【已解决】Pymongo出错：pymongo.errors.OperationFailure: Authentication failed
- 【已解决】MongoDB的MongoDB Compass中用正则进行模糊匹配字段
- 【规避解决】Flask-PyMongo中如何查询gridfs中的文件 – 在路上
- 【已解决】python中mongo出错：pymongo.errors.CursorNotFound: Cursor not found
- 【记录】MongoDB中本地写代码实现数据合并和迁移
- 【已解决】在线CentOS中Flask运行mongo出错：pymongo.errors.ServerSelectionTimeoutError localhost Errno 111 Connection refused
- 【已解决】添加了IP限制的mongod重启出错：Job for mongod.service failed because the control process exited with error code
- 【已解决】mongo中给bindIp添加多个IP后出错：getaddrinfo failed Name or service not known
- 【已解决】mongo启动失败：connection /var/lib/mongo/WiredTiger.turtle handle-open open Permission denied
- 【已解决】mongo启动失败：Failed to unlink socket file /tmp/mongodb-xxx.sock errno 1 Operation not permitted
- 【已解决】添加了IP限制的mongod重启出错：Job for mongod.service failed because the control process exited with error code – 在路上
- 【已解决】mongo启动失败：connection /var/lib/mongo/WiredTiger.turtle handle-open open Permission denied
- 【已解决】配置mongod以允许内网其他服务器访问mongo服务
- 【无法解决】尝试用mongo的bindIp去实现限制特定IP才能连接mongo服务
- 【未解决】systemctl或service无法重启或启动mongod的服务mongod
- 【已解决】mongo中给bindIp添加多个IP后出错：getaddrinfo failed Name or service not known
- 【已解决】mongo启动失败：Failed to unlink socket file /tmp/mongodb-xxx.sock errno 1 Operation not permitted
- 【已解决】给MongoDB数据库新建用户和权限
- 【已解决】pymongo中用MongoClient去连接远程加了权限控制的mongoDB
- 【已解决】本地mongo shell中连接远程加了权限控制的mongoDB
- 【已解决】如何允许外网IP远程访问MongoDB数据库
- 【已解决】连接远程mongoDB失败：Failed to connect to after 5000ms milliseconds giving up

- 【已解决】阿里云ECS服务器中已有的MongoDB的用户名密码和端口
- 【已解决】PyMongo中GridFS的exists始终检测不到文件已存在
- 【已解决】安装MongoDB Compass去图形化查看Mongo数据 – 在路上
-
- How to Import .bson file format on mongodb - Stack Overflow
- 最佳的MongoDB客户端管理工具 - chszs的专栏 - CSDN博客
- Working with MongoDB in Visual Studio Code
- 文档数据库_文档型数据库_AWS 数据库服务 - AWS 云服务
- Top 12 NoSQL Document Databases in 2020 - Reviews, Features, Pricing, Comparison
- 常用数据库及nosql - 简书
- Sql Or NoSql，看完这一篇你就懂了 - 五月的仓颉 - 博客园
- NoSQL 还是 SQL？这一篇讲清楚 - 掘金
- 常用数据库有哪些（附带数据库排名）？
- MongoDB跑步进中国
- MongoDB API
- MongoDB Drivers and Ecosystem — MongoDB Drivers
- VSCode支持MongoDB
- MongoDB 常用的几大GUI工具 - 自由早晚乱余生 - 博客园
- Mongodb 工具 Studio 3T 和 Robo 3T · 码农装备
- Robo 3T | Free, open-source MongoDB GUI (formerly Robomongo)
- MongoDB的可视化工具：Studio 3T和Robo 3T有什么区别啊？ - SegmentFault 思否
- cursor.sort() — MongoDB Manual
- $orderby — MongoDB Manual
- MongoDB 排序 | 菜鸟教程
- mongodb - How do you tell Mongo to sort a collection before limiting the results? - Stack Overflow
- MongoDB find() Method: Introduction & Query Examples | Studio 3T
- MongoDB Python Drivers — MongoDB Drivers
- 初尝node.js + Express + MongoDB + Vue.js 项目构建(2) - 个人文章 - SegmentFault
-

TODO 待整理

- 【已解决】mongo命令行中如何删除文件
- 【已解决】把本地的音频字幕等数据存储到远程服务器的MongoDB数据库中
- 【已解决】mongo中给普通数据库gridfs创建root的角色失败：Error couldn't add user No role named root@gridfs

- 【已解决】本地mongo shell中连接远程加了权限控制的mongoDB
- 【已解决】阿里云ECS服务器中已有的MongoDB的用户名密码和端口
- 【未解决】尝试用Mongo Management Studio去实现导入文件到Mongo的gridfs且带metadata信息
- 【已解决】确认服务器中MongoDB数据库是否有或已开启记录登录的日志
- 【已解决】确认服务器中MongoDB数据库是否有或已开启记录登录的日志
- 【已解决】MongoDB开启访问控制后currentOp出错：not authorized on admin to execute command
- 【已解决】给MongoDB数据库新建用户和权限
- 【已解决】修改MongoDB的默认端口号27017为别的端口
- 【已解决】mongo的shell中find返回多个有限个数的结果
- 【已解决】公司Wi-Fi更换运营商导致IP变化导致远程Mongo连不上
- 【已解决】连接远程mongoDB失败：Failed to connect to after 5000ms milliseconds giving up
- 【已解决】pymongo中用MongoClient去连接远程加了权限控制的mongoDB
- 【已解决】用PyCharm的MongoDB插件连接远程MongoDB数据库
- 【已解决】MongoDB的用户的密码中包含@如何写URI
- 【已解决】给MongoDB限制IP访问
- 【已解决】远程MongoDB新增dialog数据库并新增对应用户和权限
- 【已解决】PyCharm连接远程添加security的authorization的MongoDB出错：com.mongodb.MongoCommandExceptions: Command failed with error 13
- 【已解决】Flask-PyMongo出错：RuntimeError Working outside of application context
- 【已解决】pymongo的count()出错：pymongo.errors.ServerSelectionTimeoutError timed out
- 【记录】通过阿里云ECS服务器安全组限制访问mongo的IP和端口
- 【已解决】用PyCharm写Python的MongoDB代码并调试
- 【已解决】配置mongod以允许内网其他服务器访问mongo服务
- 【已解决】PyCharm中安装MongoDB的插件：mongo4idea
- mongodump — MongoDB Database Tools
-