1. 定义:开源，高性能，无模式的文档数据库

2. 应用场景：解决 高并发，海量数据的访问和存储，对数据库的高扩展性和高可用性，具体场景： 社交，游戏，物流，物联网，直播等等

3. 缺点：对事务性的支持不大友好，没有复杂的join

4. show dbs ：列出全部的数据库

5. use articledb： use 切换数据库，没有的话就创建articledb数据库，但是只有这个库中有内容时，才会真正被创建。数据库名称必须全小写，最长字64字节

db: 查看正在使用的数据库; db.dropDatabase() 删除数据库;

6. 基本数据库:

admin: root数据库

local: 这个库的数据永远不会被复制，用来存储限于本地单台服务器的任意集合

config: 分片设置时，数据库内部使用，保存分片相关信息

7. 创建集合: db.createCollection**(**"member"**); 查询集合:** show collections; 删除集合: db.member.drop**();**

8. 插入 db.user.insert**({** "Name" **:** "rrr"**,** "Age" : NumberInt(18),"LoginTime" : ISODate("2020-06-13T16**:**13**:**41.084+0000"**)})**

**批量插入 db.user.**insertMany([{"Name" **:** "rrr"},{"Name" **:** "ttt"，"LoginTime" :new Date()}])

**int类型使用**NumberInt（）

如果发生异常，并不会回滚，所以为防止出现错误，使用try catch

try{

**db.user.**insertMany([{"Name" **:** "rrr","LoginTime" :new Date("2020-06-13T16**:**13**:**41.084+0000")},{"Name" **:** "ttt"，"LoginTime" :new Date()}])

}catch(e){

Print(e)

}

**9. 修改**

**10.1 覆盖修改(只改符合条件的第一个数据): 第一个{}是查询条件,后面的{}是需要修改的字段，这个**update**会使全部的json都清空，然后把后面的{}修改到文档中**

db.user.update**({**Name **:** 'aaa'**},{**Age**:**NumberInt**(**20**)})**

**原数据: {** "\_id" **:** ObjectId**(**"5ee622264d8d36af1ed8e6f5"**),** "Name" **:** "www"**,** "Age" **:** NumberInt**(**18**),** Account" **:** "qwe"**}**

**修改后的数据: {** "\_id" **:** ObjectId**(**"5ee622264d8d36af1ed8e6f5"**),** "Age" **:** NumberInt**(**20**) }**

**10.2 局部修改(只改符合条件的第一个数据): 使用{$set:{要修改的字段:值}} 只修改某个字段，而不修改整个文档**

db.user.update**({**Name **:** 'www'**},{**$set**:{**Age**:**NumberInt**(**25**)}})**

10.3 批量更新 最后一个参数添加multi**:**true 选项

db.user.update**({**Name **:** 'www'**},{**$set**:{**Age**:**NumberInt**(**10**)}},{**multi**:**true**})**

**10.4 列值增长** $inc 给指定数字列增加值

db.user.update**({**Name **:** 'www'**},{**$inc**:{**Age**:**NumberInt**(**10**)}})**

**10. 删除**

db.user.remove**({**\_id**:** ObjectId**(**"5ee4fb35c2f10a458c1cd7c7"**)})**

11. 查询全部 **db.user.**find()

查询匹配 **db.user.**find({"Name" **:** "rrr"})

查询单个 db.user.findOne**({**Name **:** 'rrr'**})**

**只查询部分字段 第一个{}是查询条件，后面的是显示字段,1表示查询该字段,0代表隐藏不显示该\_id字段，默认查询是显示\_id**

db.user.find**({**Name **:** 'rrr'**},{**Name**:**1**,**Age**:**1**,**\_id**:**0**})**

**统计总数:** db.user.count**() 条件统计总数:** db.user.count**({**Name**:**"www"**})**

**排序: 1升序 -1降序**

db.user.find**()**.skip**(**1**)**.limit**(**3**)**.sort**({**Sex**:**1**,**Age**:**-1**})**

**正则查询：**find**({**Name**:**/正则表达式/**})**

db.user.find**({**Name**:**/e/**})**

比较查询 $gt(大于) $lt(小于) $gte(大于等于) $lte(小于等于) $ne(不等于) $eq(等于)

db.user.find**({**Age**:{**$gt**:**NumberInt**(**1**)}})**

**包含查询** $in包含 $nin(不包含)

db.user.find**({**Age**:{**$in**:[**NumberInt**(**18**),**NumberInt**(**19**)]}})**

**条件连接查询 $and $or**

db.user.find**({**$and**:[{**Age**:{**$gt**:**NumberInt**(**10**)}},{**Age**:{**$lt**:**NumberInt**(**19**)}}]})**

**12. 索引:**

**12.1 单字段索引 复合索引(多字段) 其他索引(地理空间索引,文本索引,哈希索引)**

**12.2 查看索引** db. user.getIndexes**()**

**[{**

"v" **:** 2.0**,** //索引版本号

"key" **:** **{ //主键**

"\_id" **:** 1.0 //哪个字段添加的索引 1升序 -1降序

**},**

"name" **:** "\_id\_"**,** //索引名称，就是索引字段又添加了个\_

"ns" **:** "zhaoxi.collection" //索引所在的表

**}]**

**12.3 创建索引** createIndex({keys},{options})

db. user.createIndex**({**Name**:**1**,**Age**:**-1**},{unique:true})**

**12.4 删除索引，使用主键或者索引名称**

db.user.dropIndex({Name:1,Age:-1})

db.user.dropIndex(“\_id\_”)

db.user.dropIndexes**() //删除全部索引**

**12.5 索引使用**

**12.5.1 执行计划，find必须是索引字段,查询语句后追加**explain**()**

db.user.find**({**Name**:**"qwe"**})**.explain**()**

**12.5.2 覆盖查询,只查询索引字段，不会进入文档数据库，直接返回索引，速度非常快**

db.user.find**({**Name**:**/e/**},{**Name**:**1**})**