## MongoDB

### 数据库基本概念

* 数据：能够输入到计算机中并被识别处理的信息集合
* 数据结构：研究一个数据集合中，数据元素关系
* 数据库：按照数据结构，存储管理数据的仓库
* 数据库管理系统：数据库管理软件，用于建立维护操作数据库
* 数据库系统：由数据库+数据库管理系统等构成的数据库开发工具集合

### 关系型数据库

* 采用关系模式（二维表）来组织数据结构的数据库
* 优点：
  + 逻辑清晰，容易理解，结构类似常见表格
  + 使用SQL语句，技术成熟，使用方便
  + 关系型数据库比较成熟，可以使用一些复杂的操作
* 缺点：
  + 每次操作都需要专门的sql解析
  + 结构严格，内部加锁
  + 在应对海量数据并发处理时读写速度差

### 非关系型数据库NoSQL

* 不是采用关系模型构建的数据库
  + 键值型：Redis
  + 列存储：HBase
  + 文档型：MongoDB
  + 图形：Graph
* 优点：
  + 读写速度快，更好的针对并发处理
  + 使用灵活，容易扩展
* 缺点：
  + 没有SQL那样统一成熟的语句
  + 技术成熟度较差，缺少一些复杂操作

### 应用场景

* 对数据格式要求不严格，比较灵活
* 对数据处理速度，特别是海量数据的并发要求比较高
* 某些特定场景
  + 需要灵活扩展
  + 需要作为缓存

### MongoDB基础

* 非关系文档的开源数据库
* 由C++编写的数据库管理系统
* 支持丰富的存储类型和数据操作
* 提供了丰富的编程语言接口
* 方便扩展和部署
* 默认端口：27017

#### Linux安装目录

* 安装位置：/var/lib/mongodb..
* 配置文件：/etc/mongodb.conf
* 命令集：/use/bin

#### 进入mongodb交互界面

* 名称：mongo shell
* 命令：quit()
* 设置mongodb基本信息
  + mogod –h查看帮助
  + mongod –dbpath [dir]设置数据库存储位置
  + mongod –port [port] 设置数据库端口

#### mongodDB数据库数据结构

* 数据组织结构：键值对》文档》集合》数据库
* 表/集合 collection
* 字段/域 field
* 记录文档 document

#### 数据库操作

* 创建数据库：use [database]
  + Use创建数据库并不会立即建立，而是当真正插入数据才建立
* 查询数据库：show dbs
* 删除数据库：db.dropDatabase()
  + 全局变量db：当前正在使用的db（默认是test数据库）
* 数据库的备份和恢复：
  + mongodump – h[host] –d[db] –o[path]
    - monggodump –h 127.0.0.1 –d stu –o D:/
  + mongorestore –h [host:port] –d [db] [bak]
    - mongorestore –h 127.0.0.1 –d student stu
* 数据库监控：mongostat
  + insert query update delete
* 查看数据库集合读写时长（每个集合在一秒内的读写时间）：mongotop

### 集合创建

* db.createCollection( collectionName )
* db.collction.insert(…)
  + 插入数据时如果不存在，则自动创建
* 查看集合：show collections
* 删除集合：db.class.drop()
* 重命名：db.collection.renameCollection(newName)

### 文档

* 文档是mongodb数据库中基本的数据组织单元
* 文档由键值对构成，每个键值对表达一个数据
* Mongodb文档数据bson类型数据
* 无序、不能重复，键取值
* 键只能是utf-8字符串，不能有\0字符
* 值为beson支持的数据类型，即存储的值
* 一个集合中的文档可以有不同的域，域的个数也可以不一致
  + 但文档层次不宜嵌套过多，如果层次过多应考虑分为多个集合

### 文档操作

* 插入文档：
  + db.collection.insertOne({‘name’:’Lucy’,’age’:18})
    - 操作数据时，键可以不带引号
    - 可以自己设置\_id值，但是不能重复
  + db.collection.insertMany([{},{}])
    - []表示的一组数据有序集合，使用序列号取值
  + db.collection.save()
    - 插入一行或者多行数据
    - 如果id冲突会替换原有文档
* 查找文档：
  + db.collection.find(query,field)
    - query查找条件
      * 是一个键值对文档，默认{}为空表示所有
      * 键值对表示筛选条件，往往配合查找操作符完成
    - field要查找的域
      * 域：0表示不查找这个域，1表示查找如\_id:0
      * Field如果不想查找则\_id:0，其他域要么全0要么全1
    - collection要查找的集合

### query操作符

* 比较操作符
  + $lt <
  + $eq =
  + $gt >
  + $lte <=
  + $gte >=
  + $ne !=
  + $in 包含
  + $nin 不包含
    - {'age':{"$gt":8,"$lt":14}} ->年龄8-14之间的
* 逻辑操作符
  + $and
    - { '$and' : [ {'age': {'$lt':18} } , { 'sex' : 'w' } ] } , {'\_id':0}
  + $or
    - {'$or': [{'age':{'$lt':18}}, {'sex':'w'}]},{'\_id':0}
  + $not
    - class0.find({'age':{'$not':{'$gt':18}}},{'\_id':0})
  + $nor 都假才真
* 数组
  + 查找数组是否包含某个数,直接在数组中查找
  + 查找数组是否包含多个$all
    - Class0.find({score:{‘$all’:[87,89]}},{‘\_id’:0})
  + 根据数组的元组数量查找：$size
  + 表示查找数组哪些项：$slice
    - {‘\_id’:0,score:{‘$slice’:2}} ----------只显示score的前2项
  + 表示查找数组的第几项匹配数据
    - {‘score.1’:{‘$qt’：80}}--------------------查找score列表第二个元素的值
* 其他操作符
  + $exists判断某域是否存在
    - class0.find({'sex':{'$exists':'true'}}
  + $mod某域的值%值
    - class0.find({'age':{'$mod':[4,0]}})
  + $type 匹配某域的值的类型

### 数据操作函数

* 获取集合中某个域的值范围：db.collection.distinct(field) ->数组
* 将find结果格式化显示：find().pretty()
* 显示查询结果的前n条：find().limit(n)
* 跳过前n条显示后面的文档：find().skip()
* 查看查询出多少条数据：find().count()
* 排序1升序：find().sort(field:1/-1{})
* 查找指定位置的文档：find()[0] ->获取第一个文档

### 修改

* Db.collection.updateOne(query,update,upsert)
  + Update要修改的数据，需要同修改操作符一起使用
  + Upsert如果没有筛选到文档是否插入新的文档
    - db.class0.updateOne({"name":"HHH"},{"$set":{"name":"John"}},{upsert:true})
* Db.collection.updateMany(query,update,upsert)
  + db.class0.updateMany({"name":"A"},{"$set":{"name":"JohnA"}},{upsert:true})
* Db.collection.update(query,update,true,false)
  + 第一个true表示查找不到文档则插入新的文档,false相反
  + 第二个false表示只修改一个
    - db.class0.update({"name":"C"},{"$set":{"name":"HaliLUYa"}},true,true)
* findOneAndUpdate(query,update)
  + - 查找到一个文档并修改
    - 返回修改前的文档
* findOneAndReplace(query,doc)
  + 查找一个文档并替换
  + 新的文档
  + 返回：源文档

### 修改操作符（修改器）

* $set：修改一个域的值或者增加一个域
  + db.class0.updateMany({"sex":{"$exists":false}},{"$set":{"sex":"man"}})—为没有sex域的文档添加sex值
  + db.class0.updateMany({"sex":{"$exists":false}},{"$set":{"sex":"man",”age”:18}})—添加多个域
* $unset:删除一个域
* $rename:给域重命名
  + db.class0.updateMany({},{$rename:{'name':'nickname'}})
* $setOnInsert:如果使用update\*执行了插入文档操作，则作为插入的内容
* $inc：加法修改器
* $mul：乘法修改器
* $max：修改某个域的值，如果小于指定值则改为指定值，大于则不变
* $min：如果大于指定值则修改为指定值，小于则不变
* 数组修改器
  + $push：向数组中添加一项
    - {$push:{score:10}}
  + $pushAll：向数组中添加多项
  + $pull：向数组中删除一项
  + $pullAll:向数组中删除多项
  + $pop：向数组中删除最后一项1，第一项-1
  + $addToSet:向数组中添加一项，但是不能和已有的数值重复
  + $each:对多个值逐一操作
  + $postion：指定数组插入位置，配合$each
  + $sort：给数组排序，配合$each使用-1降序，1升序
    - {$push:{score:{$each:[],$set:-1}}}

### 删除

* db.collection.deleteOne(query)
  + db.class0.deleteOne({hobby:{$exists:false}})
* db.collection.deleteMany(query)
* 删除一个或多个：db.collection.remove(query,justOne)
* 查找一个文档并删除：db.collection.findOneAndDelete(query)
  + 返回查找到的文档

### 函数

* new Date()：自动生成当前时间
* Date()：获取计算机当前时间生成字符串
* ISODate()：将指定时间转换为标准时间输入
  + 参数：默认和new Date()一样
  + (”2019-1-12”)
* valueOf()：将标准时间转化为时间戳
* null类型
  + 表示某个域的值为空
  + 查找时表示某个域没有或值为null

### 内部文档object类型

* 当使用内部文档某个域的值得时候，需要采用”外部域.内部域”的方法
  + “intro.author”

### 固定集合

* 指的是mongoDB中创建固定大小的集合，称之为固定集合
  + 如果超过了大小，把最早的一个数值剔除
  + 数据插入，查找速度快
  + 可以控制集合大小
* 日志处理，临时缓存
* 创建：db.createCollection(collection,{capped:true,size:10000,max:20})
  + Capped：创建固定集合
  + Size：固定集合大小（字节）
  + Max：固定集合可存放文档数量

### 文件存储

* 存储路径
  + 操作简单，节省数据库空间
  + 当数据库或者文件发生变化需要修改数据库
* 存储文件本身：将文件转化为二进制存储到数据库中
  + 文件绑定数据库不易丢失
  + 文件存取效率低，占用数据空间
* GridFS文件存储方案
  + 更方便的存储mongodb中大文件（>16M）
    - 创建2个集合共同存储文件
    - Fs.files集合中为每个文件建立一个信息文档，存储文件的基本信息
    - Fs.chunks集合中为每个文档建立与fs.files的关联，并将文件分块存储
      * 将postman.tar.gz存储到grid中
        + mongoFiles:-d grid put postman.tar.gz
        + db.fs.files.find()查询
      * 将grid的postman.tar.gz读取出来
        + （在命令行下）mongofiles – d grid get postman.tar.gz
  + 对单文件的存款提取方便，但是读写效率依旧比较低，不建议存小数据文件

### Mongo shell 对JavaScript支持

* Mongo shell界面中支持基本的JavaScript代码

### 权限

* 创建管理用户
  + db.createUser({user:”userAdmin”,pwd:”123456”,roles:[{role:”userAdminAnyDatabase”,db:”admin”}]})
* 授权用户：
  + db.auth(“userAdmin”,”123456”)
* 查看权限
  + db.auth(“userAdmin”,”123456”)
  + show dbs
* 创建一个读写用户
  + use dm
  + db
  + db.createUser({user:“dba”,pwd:”dba”,roles:[{role:”readWrite”,db:”dm”}]})
* 重新授权
  + db.auth(“dba”,”dba”)
* db.auth(“dba”,”dba”)

### 1. 创建／重建索引

* 索引约束
  + 数据量很小时不需要创建索引
  + 创建索引会增加磁盘的使用空间
  + 对数据库操作大多是写操作而不是读操作时不宜创建索引

#### 1.1 创建索引

* db.collection.createIndex(key[,options])
  + keys，要建立索引的参数列表。如：{KEY:1}，其中key表示字段名，1表示升序排序，也可使用使用数字-1降序
  + \_id索引系统自动生成，无法删除
  + options，可选参数，表示建立索引的设置。可选值如下：
    - background，Boolean，在后台建立索引，以便建立索引时不阻止其他数据库活动。默认值 false。
    - unique，Boolean，创建唯一索引。默认值 false。
    - name，String，指定索引的名称。如果未指定，MongoDB会生成一个索引字段的名称和排序顺序串联。
    - dropDups，Boolean，创建唯一索引时，如果出现重复删除后续出现的相同索引，只保留第一个。
    - sparse，Boolean，对文档中不存在的字段数据不启用索引。默认值是 false。
    - v，index version，索引的版本号。
    - weights，document，索引权重值，数值在 1 到 99,999 之间，表示该索引相对于其他索引字段的得分权重。
      * {‘age’:-1},{‘name’:’ageIndex’}
* db.collection.createIndexs(key[,options])
* db.COLLECTION\_NAME.ensureIndex(keys[,options])
* db.createIndex({‘name’:1,’age’:1})创建name和age组合的复合索引
* 唯一索引：,{upique:true}
* 稀疏索引：对没有索引对应的域自动忽略
  + ,{sparse:true}

### 2. 查看索引

* db.collction.getIndexes()：查看集合的所有索引，
* db.COLLECTION\_NAME.totalIndexSize()：查看集合索引的总大小
* db.system.indexes.find()：查看数据库中所有索引信息。

### 3. 删除索引

* db.COLLECTION\_NAME.dropIndex("INDEX-NAME")：删除集合中的某一索引
  + db.sites.dropIndex("name\_1\_domain\_-1")
* db.COLLECTION\_NAME.dropIndexes():删除全部索引
  + db.sites.dropIndexes()

### 聚合索引

* db.collection.aggregate()
  + 执行聚合操作
  + 聚合条件，配合聚合操作符使用
    - $group
      * db.class0.aggregate({$group:{\_id:"$age",num:{$sum:1}}})---按照age进行分组,每种属性+1
      * aggregate({$group:{\_id:"$sex",num:{$avg:'$age'}}})—按照sex分组，得到年龄平均值
      * $max/$min最大最小值
      * $first/$last求第一个值、求最后一个值
      * $match写法基本和query相同
        + aggregate({$match:{age:18}})
      * $limit 获取集合前几条文档
      * $skip 跳过前几个文档
      * $sort 对文档排序
      * 聚合管道
        + db.class0.aggregate([{},{},{},…]) 将前一个聚合产生的结果交给后一个聚合继续操作

db.class0.aggregate([{$sort:{'age':-1}},{$limit:3}])

### Python连接

class MONGOHelper:

def \_\_init\_\_(self, address=('localhost', 27017), link\_db="moko", user=()):

self.con = MongoClient(\*address)#获取数据库连接

self.db = eval("self.con." + link\_db)#连接到对应的数据库

if user:

self.db.authenticate(\*user)

def \_\_del\_\_(self):

self.con.close() #关闭连接

* self.db.class0.find()---------------得到游标对象cursor
* cursor.count()等属性来进行筛选
* sort([(‘age’,1),(‘name’,1)])
  + 调用skip/sort/limit时游标必须没有遍历过
  + Sort排序写法与mongo不同
* update({‘name’:”a”},{'$set’:{‘king\_name’:’b’}},multi=True)
  + multi按照域名区修改
  + upset如果没有筛选到文档，是否插入新文档
* create\_index(index,\*\*kwargs)
  + 创建索引
  + {name:1} - - > （[‘name’,1]）

（"name”）：默认1

Kwargs索引选项

* + 索引名称
* list\_indexs()查看索引

#### 文件存储

* import bson
* 将文件内容转换为bson二进制格式存储
  + Content = bson.binary.Binary(bytes)
    - 将python字节串转换为bson格式
    - Bytes：要转换的字节串
    - 返回：bson数据
  + 将bson数据插入到mongo数据库文档中