MongoDB权威指南

mongo是JavaScript shell：这就需要好好再去学习一下js了

怎么学习

> 1，两本书《mongodb权威指南》入门和《深入学习mongodb》更进一步
> 2，阿里云云栖社区：MongoDB资料大全： https://yq.aliyun.com/articles/53867?utm_campaign=wenzhang&utm_medium=article&utm_source=QQ-qun&utm_content=m_10349
> 3，英文文档：https://docs.mongodb.com/manual/
> 4，中文文档：http://docs.mongoing.com/manual-zh/
> 5，官网，以及相关问题问答部分
> 6，源码
> 7，技术群：qq群，微信群

基本命令

MongoDB是面向文档的数据库，具有更加方便的拓展性

MongoDB VS Mysql

概念	mysql	mongodb
数据库	database	database
表	table	collection
记录	record	document

document文档：核心概念

> 1，文档的键是字符串，不能含有\0（空格，是以空格为结尾符的）区分类型，大小写，不可有重复的键

collection集合：一组文档

> 1，集合是无模式的，即集合里面的文档的格式各样，但是开发上，还是一个集合一种文档格式，这样对提取数据也方便
> 2，集合名不能是空串""，不能含有空格，不能以system.开头，用户命名的集合不能含有保留字符$

子集合:例如 blog.authors

数据库：最终变成文件系统里的文件

> 1，数据库名不能有空串，不能含有空格 . \ / \0,应全部小写，最多64字节
> 2，保留字段不能用，比如admin，config，root等

类型

> 1，JavaScript数值类型仅支持64位浮点数，所以32位整型会被自动转换成64位浮点型
> 2，不支持64位整型，但是shell使一个特殊内嵌文档来显示64位整数

> 3，JavaScript只有一种数字类型（64位浮点数），但是mongodb有3种（32位整型，64位整型，64位浮点型），javascript shell通过办法绕过JavaScript的限制，但是存取时，会造成不准确的可能，要是插入的64位整数不能精确的双精度浮点显示，shell会自动添加两个键，top（高32位）和bottom（低32位）
> 4，数组可以包含不同数据类型的元素{"things":["pie",3.14]}

_id:ObjectId()数据插入时额外生成的唯一健，适应分布式

> 1，12字节存储空间，24位十六进制字符串，例如："_id" : ObjectId("5d36d01d121c0f10bd2df3e4")
> 5d36d01d（16进制）=1563873309（十进制）=》 1563873309（时间戳：秒）=2019/7/23 17:15:9（北京时间）
> 2，格式：时间戳（4字节，秒） 机器（3字节，通常是机器名散列值） PID（2字节，产生ObjectId的进程id） 计数器（3字节，范围2的24次方，即每个机器每个进程每秒中最多允许16777216个不同的ObjectId） 时间戳在前，数据大致以插入顺序排列

insert插入

> 1，批量插入只支持对统一集合操作，不支持不同集合的批量插入
> 2，只检查是否包含_id，文档大小不超过4MB，不做别的数据校验，当然可以再启动数据库的时候，开启 --objcheck选项，插入之间先检查文档结构的有效性，有性能开销

remove删除：不能撤销，不能恢复，所以删除操作要谨慎

update更新：两个参数：查询文档+修改器(原子性)文档 mongodb为文档预留了空间，当超过，则分配一块新空间

> 1，$inc：不存在则创建，增加修改器
> {"_id":ObjectId("5d36d01d121c0f10bd2df3e4"),"url":"xxx","pages":51} db.coll.update({"url":"xxx"},{"$inc":{"pages":1}})=>{"_id":ObjectId("5d36d01d121c0f10bd2df3e4"),"url":"xxx","pages":52}
> 2，$set：不存在则创建 $unset
> 3，数组修改器 $push
> 4，如果一个值不在数组里就加进去 $ne $addToSet $each
> 5，$pop：把数组看出队列或栈 $pull：根据特定条件删除
> 6，定位符$ 只更新第一个匹配的元素

不可靠请求：即client发送数据，不管server是否存在，server就算在也不会给个回应，说，嘿，收到了你的数据，有点UDP协议的感觉

> 1，安全的版本再执行完操作之后立即运行getLastError命令。来检查是否执行成功

为每一个链接创建一个请求队列：顺序执行

> 1，在一个shell里（一个链接中），操作1插入，操作2查询可以看到新插入的数据，当在不同的shell里，shell1插入数据，在shell2中查询，不一定可以看到先插入的数据---mysql的隔离性？？

查询：没办法join查询

> 1，$in $or
> 2，$not $mod取余
> 3，null 会匹配自己也匹配不存在这个键的数据 db.coll.find({"xx":null}) 会返回"xx":null的数据和没有"xx"的数据 用"$exist"：true 来避免不存在键的数据查出

> 4，正则
> 5，$all 类似mysql and 与条件
> 6，$slice 返回数组中指定个数元素或之间索引范围返回的元素
> 7，$where实现其他查询满足不了查询
> 8，游标（客户端游标，服务器端游标：消耗内存等资源） 类似索引指针，超时（十分钟）自动销毁或客户端发来消息主动终止
> 9，调用find，shell不是立即执行查询，而是等待真正开始要求获得结果的时候才发送查询请求到server
> 10，健值可能是多种类型，顺序查询，返回结果有个固定预定义的排列顺序
> 11，高级查询：limit sort skip （分页查询 .limit(100) skip(100).limit(100)） $hint:指定查询索引
> $explain：获取查询细节 $snapshot：快照

既然查询时针对不变的集合视图运行，那如何保证数据的一致性，即一个在读一个在写同一个集合的时候？？？如果没有锁，是如何做到的

索引：若是需要对多个键加索引，需要考虑索引排序方向问题

创建索引或者其他的一些操作，会导致什么样的数据存放和调整，这涉及在数据量大的时候性能问题

> 1，对无索引字段排序，在数据量大时，数据不能在内存中排序时，mongodb会报错
> 2，建立索引耗时费力，需要消耗很多内存和计算资源，可以用db.coll.ensureIndex({"username":1}, {"background":true}) 后台完成，同时正常处理请求，否则在建立索引期间会阻塞所有请求

地理空间索引："2d"

固定集合：集合大小固定，类似环形队列，如果内存不够，最早文档会被删除，为新文档腾出空间

文档基本是按时间顺序存储

GridFS文件系统

> 1，基本思想是将大文件分成很多小块，每块作为一个单独的文档存储
> 2，支持在文档中存储二进制数据，可以最小限度减小块的存储开销，但是需要一个单独的文档存储分块的信息和元数据

db.eval执行自定义JavaScript脚本

> 1，db.eval("function(uname){ return 'hello, '+uname;}",['liujun'])

数据库引用 DBRef

停止mongodb：kill 或kill -2 或shutdown命令

> 1，不用用kill -9 pid 会导致数据库直接关闭，一些关闭必要的步骤没有执行，会使数据文件损毁

安全认证 --auth：类似mysql的读写权限控制等

> 1，传输协议没加密，需要加密，可以用ssh隧道或其他加密技术
> 2，建议不知在防火墙后或者设置只有应用服务器可以访问，但是要外面也能访问的话，建议使用--

> bindip localhost 选项 3，可以用--noscripting完全禁止服务端执行脚本

数据备份： --dbpath 存放这mongodb所有数据，备份就是对这个配置的文件夹中所有文件创建副本

> 1，在运行中穿件数据目录的副本不安全，备份数据容易损坏
> 2，mongodump能在运行时备份，不一定是服务器数据的实时快照

数据库修复

> 1，因为宕机，停电，软件故障等，因为mongodb的存储方式不能保证磁盘上的数据还能用，有可能被损坏 重启时，加上--repair：mongod
> 2，修复过程：将所有文档导出，再马上导入，忽略无效文档，并重新建立索引，损毁的文档会被丢弃

尽可能稳妥的停掉服务器，利用复制功能恢复故障，经常做备份，才是最有效的数据管理的方式

主从复制：一个主节点和一个或多个从节点

> 1，mongodb --dbpath ~/dbs/master --port 1000 --master 开启主节点 mongodb --dbpath ~/dbs/slave --port 1001 --slave --source localhost:1000 开启从节点 ：可以在一个服务器上实现主从集群

副本集群vs主从集群：副本集群是选举出一个主节点，当不能工作的时候则变更为选举其他节点 主从是设定主从关系

> 1，不能用localhost地址作为成员（vim /etc/hostname morton） 2，通过集群名字比如blort，去节点在局域网内自己组成一个集群
> 3，1)mongodb --dbpath ~/dbs/node1 --port 10001 --replSet blort/morton:10002 2)mongodb --dbpath ~/dbs/node1 --port 10002 --replSet blort/morton:10001 3)mongodb --dbpath ~/dbs/node1 --port 10003 --replSet blort/morton:10001,morton:10002 (只写一个也行，应该它们之间会打电话，就彼此都知道在群里)

副本集群节点类型

> 1，standard：常规节点，存储一份完整的数据，参与选举投票，有可能成为活跃节点
> 2，passive:存储了完整的数据，参与投票，不能成为活跃节点
> 3，arbiter：仲裁者只参与投票，不接受复制的数据，也不能成为活跃节点
> 4，选举：优先权priority由大到小选出活跃节点。优先值=0，被动节点，不为0，两个优先值一样大，谁的数据新谁是活跃节点（就算分成两派，也可以通过数据新旧来得出最后的选举结果）
> 5，活跃节点使用更心跳来跟踪集群中油多少节点对其可见，如果不够一半，则自动降为备份节点，防止活跃节点不放权（怎么知道一共有多少个？）

读扩展

> 1，为了减少主节点的负载，可以对从节点查询数据 ，使用--slaveOkay，默认是不可以操作从节点

写扩展

原理

> 1，原理：至少需要两个服务器或节点，一个主节点负责处理请求，其他从节点负责映射主节点的数据，主节点记录执行的所有操作，从节点定期获取主节点这些操作，然后对自己的数据副本执行操作
> 2，主节点记录操作的文件oplog，只保存改变数据库状态的操作，查询不存储在oplog中，固定文件，新的会替换旧的文件

自动分片：将集合切分成小块

> 1，片键：作为数据分片拆分的依据，比如name作为片键，则可能分片的时候，A-F开头的名字存放在第一片中,G-P存第二片中。
> 2，分片之后，如果对不是片键进行操作，mongos（路由）向所有片顺序发送查询，并归并结果

块：快信息保存在chunks集合中，可以看到数据到底怎么切分到集群的