# Redis学习笔记

## 简介

基于内存的key-value类型的非关系型数据库，支持持久化。

分为客户端和服务端，可以分别部署在不同的机器上，我们通常指的是Redis的服务端。

Redis有两种完备的持久化机制，AOF，和RDB

并发场景下，所有的请求都直接访问数据，数据库可能会出现连接异常，此时使用Redis做缓冲操作，让请求先访问Redis，而不是直接访问数据库。

## 五种数据类型

|  |  |
| --- | --- |
| String | Redis最基本的数据类型，一个key对应一个value，字符串的value值最多可以存储512M |
| List | 单键多值，底层实现是双向链表，可以做为消息队列 |
| Set | 相对于List是不重复，比如微博中求共同好友 |
| Hash | 键值对集合，key为studentID，value中存储一个对象（姓名，年龄，性别） |
| Zset | 与set相似，但是每个成员都有一个评分。 |

详见：

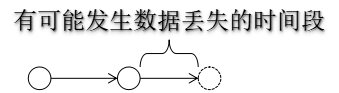
**https://blog.csdn.net/qq\_31807385/article/details/83269329**

## Redis的持久化：RDB和AOF

#### RDB：

在指定的时间间隔内将内存中的数据集快照写入磁盘（snapshot），它在恢复的时候，直接将快照文件读到内存中。

RDB是默认开启的**。Redis会单独创建（fork）一个子进程来进行持久化，会先将数据写到一个临时文件中（\*\*.rdb），等持久化过程都结束了，在用这个临时文件替换上次持久化的文件，**整个过程中主进程是不会进行任何IO操作的，这样做的缺点是：最后一次持久化后的数据可能会丢失。如果数据恢复的完整性不是恒敏感的数据，RDB要比AOF方式更加高效。



#### AOF

Append to File ，AOF是以日志的形式来记录每个写操作，并包存到特定的文件中，默认不开启

当RDB和AOF的数据不一致的情况下，按照AOF来恢复数据。AOF是对于RDB的补充。

AOF会占用更多的存储空间，恢复备份的速度也会比较慢。

## Redis主从复制

详见：

**https://blog.csdn.net/qq\_31807385/article/details/83269329**

**只有主机才可以写，从机只能读**。此时读写是分离的。

从机宕机之后，重启，宕机期间主机新增的记录，从机是否能够顺利复制？可以

其中一台从机down机之后重启，能否认识旧主？不一定，如果配置文件中配置了slaveof 就可以认识旧主，如果没有配置，就会变成master

如果所有的从机都从主机同步数据，此时主机的IO压力会比较大，如果解决？使用链式的方式来配置主从机。薪火相传，这样的方式也会有一定的风险。

主机down掉之后，从机是原地待命，还是上位 原地待命。

从机是从头开始复制主机的信息，还是复制切入以后的信息？从头开始复制，也即完全复制，读取主机的RDB文件。

### 主从复制的原理：

每次从机联通主机之后，都会发动sync指令给主机，主机立即进行存盘操作，发送RDB文件给从机，从机收到RDB文件后，进行全盘加载，之后每次主机的写操作，都会立即发送给从机，从机执行相同的命令。

### 哨兵模式sentinel

哨兵模式能够检测主机是否故障，如果故障了，根据投票数，自动将从机转换为主机。

哨兵可以有多个，哨兵不会变成主机。该配置在sentinel.conf 中配置。原本宕机的主机在恢复之后，会变成从机。具体选择哪一个作为主机，这里有一个优先级的概念，数值越小，会被哨兵推荐为主机（选择设备好的和网络好的优先，为0值的时候，不可能升级为master）。

1. 优先级靠前的
2. 选择偏移量最大的
3. 选择runid最小的从服务

线程的优先级是5，越小，优先级越高。Windows和Linux是支持优先级的，但macOS是不支持的，线程的调度机制是由操作系统决定的。

### Redis集群

一台机器的内存有限，所以能够存储在内存中的数据也非常有限，所以要搭建Redis集群。另外一个问题是：如果只是有主从复制，一台主机在面临并发的写做操作的时候，一台Redis是没有办法承担的。

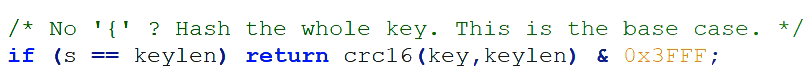
Redis集群实现了对Redis的水平扩容，即启动N个节点将整个数据库分布到这N个节点上（这N个节点都是Master）每个节点存储的总数据的N分之一

至少要有三个Master，每一个Master还需要有一个slave，所以一个Redis集群至少要有六台机器，

16384 = 16 \* 1024 = 16K

一个Redis集群包含16384个slot，数据库中的每一个键都是属于这个16384

个slot中的一个，集群使用公式CRC16（key） % 16384 来计算key是属于哪个slot，集群中的每一个节点负责一部分的slot。使用某一个机器在添加值的时候，如果计算key值不在当前的机器上，（会计算key在16384中的哪个小格子里面）集群会重新定位到目标机器。

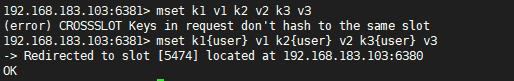


& （与运算，同1才是1）为运算之后，范围落在0~16383

集群的方式录入数据：



不在一个solt下的健值，不能使用mget，mset等多键操作，除非使用别名。



集群中如果有节点宕机了，整儿集群是否还能够工作？设定下面的参数，yes的时候，如果集群有主节点宕机，则整个集群不能工作。



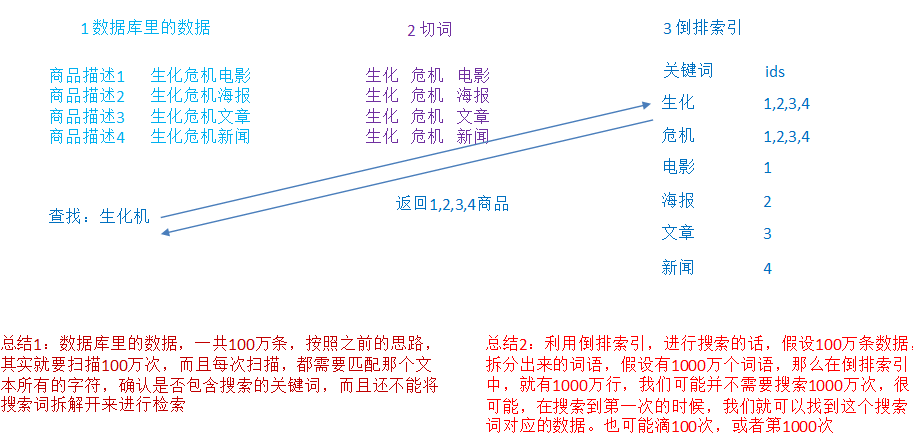
默认情况下RDB，什么都不放置的情况下，RDB是76个字节。

# Elasticsearch

搜索引擎要做的两件事情是：爬取网页，和 索引的构建；

什么是全文检索 和Lucene？

指的是计算机索引程序通过扫描文章中的每一个词，对每一个词建立一个索引，指明该词在文章中出现的次数和位置，当用户查询的时候，检索程序就根据事先建立的索引进行查找，并将查找的结果返回给用户的检索方式。



lucene，就是一个jar包，里面包含了封装好的各种建立倒排索引，以及进行搜索的代码，包括各种算法。我们就用java开发的时候，引入lucene jar，然后基于lucene的api进行去进行开发就可以了。

Elasticsearch 是一个实时分布式搜索和分析引擎，它用于全文搜索，结构化搜索，分析。

Index：（索引对应数据库）

索引包含一堆有相似结构的文档数据，比如可以有一个客户索引，商品分类索引，订单索引。商品索引里面存放的是所有的商品数据，所有的商品document。

Type（类型对应表）

每个索引里都可以有一个或者是多个type，type是index中的一个逻辑数据分类，一个type下的document，都有相同的field（只要是字段相同，就想相当于是在一张表中，一个type中，如果不在里面，就说明是另外的一张表）。

比如商品索引里很多的type，日化商品type，电器商品type，生鲜商品type

|  |
| --- |
| 日化商品type：product\_id，product\_name，product\_desc，category\_id，category\_name  电器商品type：product\_id，product\_name，product\_desc，category\_id，category\_name，service\_period  生鲜商品type：product\_id，product\_name，product\_desc，category\_id，category\_name，eat\_period |

每一个type里面都有一堆的document：

|  |  |  |
| --- | --- | --- |
| 第一个document ：   |  | | --- | | {  "product\_id": "1",  "product\_name": "长虹电视机",  "product\_desc": "4k高清",  "category\_id": "3",  "category\_name": "电器",  "service\_period": "1年"  } |  |  | | --- | | {  "product\_id": "2",  "product\_name": "基围虾",  "product\_desc": "纯天然，冰岛产",  "category\_id": "4",  "category\_name": "生鲜",  "eat\_period": "7天"  } | |

Document（文档相当于行）

文档是es中最小的数据单元，一个document可以是一条客户数据，一条商品分类数据，一条订单数据，通常使用JSON数据结构来表示。每一个index下的type都可以存储多个document。

Field（字段-列）

Field是Elasticsearch的最小单位，一个document里面有多个field，每个field就是一个数据字段。

Mapping（映射相当于约束）

数据如何存放到索引对象上，需要一个映射配置，包括：数据类型，是否存储，是否分词（数据是否作为一个整体，还是拆分开来）。

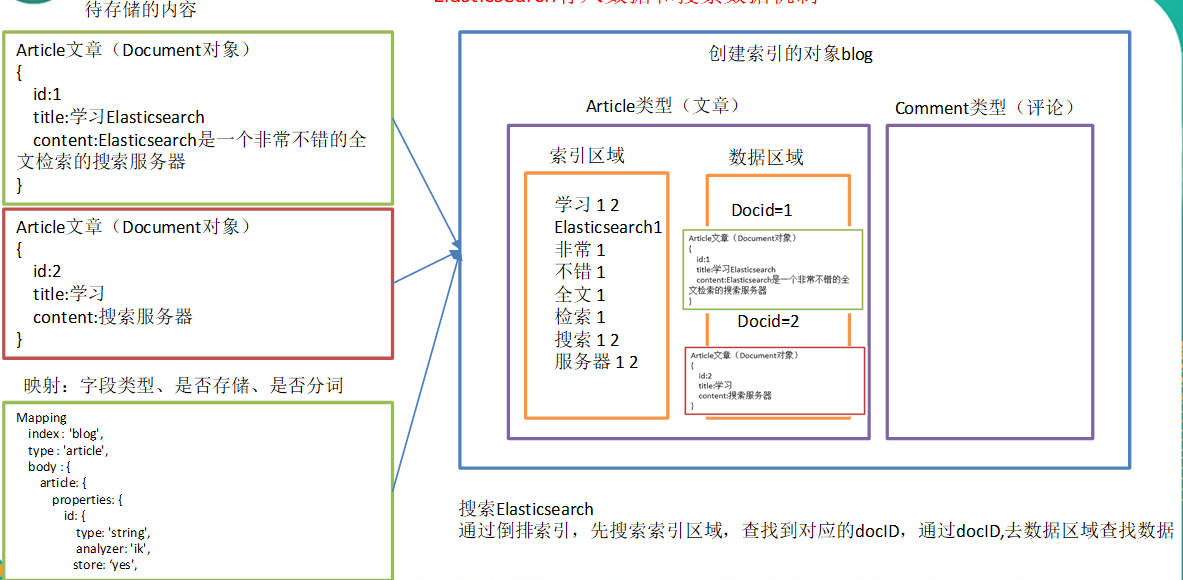
下面是建立mapping：

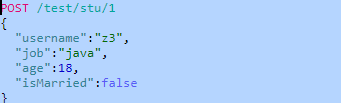
|  |
| --- |
| client.indices.putMapping({      index : 'blog',      type : 'article',      body : {          article: {              properties: {                  id: {                      type: 'string',                      analyzer: 'ik',                      store: 'yes',                  },                  title: {                      type: 'string',                      analyzer: 'ik',                      store: 'no',                  },                  content: {                      type: 'string',                      analyzer: 'ik',                      store: 'yes',                  }              }          }      }  }); |

ElasticSearch和数据库的比较。

|  |  |
| --- | --- |
| 关系型数据库（比如Mysql） | 非关系型数据库（Elasticsearch） |
| 数据库Database | 索引Index |
| 表Table | 类型Type |
| 数据行Row | 文档Document |
| 数据列Column | 字段Field |
| 约束 Schema | 映射Mapping |

Elasticsearch存入数据和搜索数据机制





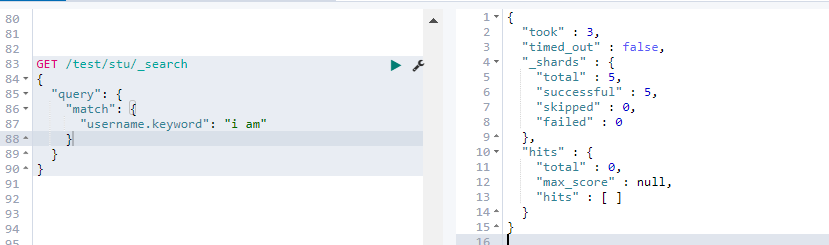
Text类型的值，会存储两种类型：text是支持分词的（默认的分词器是按照空格来分词的），但是keyword是不支持分词的



1）



使用username\_keyword的时候，是不支持分词的。



Put => create，alter

Post=> insert update

在部署ES的集群的时候，多台机器构成集群的时候，确定data是没有任何的数据的。如果有

Must是匹配：含有就OK。

Filter是过滤：必须要完全一样