# ElasticSearch

## 为什么使用es

业务系统最常用的检索方式是模糊查询，但是模糊查询 会导致数据库放弃索引，数据量大时会效率低下。但是es会对我们的数据进行分词后创建倒排索引，保证查询效率。

一个es服务中可以多个索引(indices)（数据库），每一个索引可以包含多个类型(types)（表），每一个类型包含多个文档(documents)（行），然后每个文档包含多个字段(Fields)（列）。

## Docker方式安装

1. 设置系统进程可分配的虚拟内存数量：

sysctl -w vm.max\_map\_count=262144

1. 启动容器：

单节点：

docker run --name es --restart=always -d -e ES\_JAVA\_OPTS="-Xms512m -Xmx512m" -e "discovery.type=single-node" -p 9200:9200 -p 9300:9300 elasticsearch:7.7.0

集群模式：

docker run -e ES\_JAVA\_OPTS="-Xms1024m -Xmx1024m" -d --restart=always --privileged=true -p 9200:9200 -p 9300:9300 -v /home/docker/elasticsearch/config/config.yml:/usr/share/elasticsearch/config/elasticsearch.yml

-v /home/docker/elasticsearch/plugins:/usr/share/elasticsearch/plugins --name es elasticsearch:7.7.0

踩坑：

1.Es不能用root用户创建，

sudo groupadd docker 创建docker组

gpasswd -d mk docker /usermod -a -G groupname username 将用户加入docker组中,然后重启docker服务

2.启动后发现max virtual memory areas vm.max\_map\_count [65530] is too low, increase to at least [262144]

修改文件：/etc/sysctl.conf

增加vm.max\_map\_count=262144

## 索引

概念：（indices）名词索引：一种字典作用数据节后，存储文档内容分词后的索引。

Indexing(动词索引)：es中数据存储的行为

## 文档（doc）

概念：doc：一条以文档方式存储的json字符串

## 类型（type）

概念：相当于关系数据库中的表，7.x版本已经去除此概念。

去除的原因时：类型的概念是es设计初为了映射传统关系库的表概念而创建的，但是实际数据存储时相同索引下不同类型的数据对应的是一个lucene的数据域。强加上类型的概念再使用是反而会影响效率。

## 映射（Mapping）

概念：设置es数据存储时的处理方式，比如那些字段可以被索引、数据类型、分析器、默认值。

## 分词

Es在存储数据时会对数据字段的值使用创建者指定的分词器进行分词然后创建倒排索引。在检索数据时也会对关键字使用同样的分词器进行分词再去从倒排索引中去匹配数据。

## 分析器

作用：

将文档数据转化为适合创建倒排索引的独立词条，再将词条转化为易于搜索的标准格式

使用：

创建索引时可以在setting中设置索引使用的分词器（比如："analyzer": "ik\_max\_word",//中文分词器），可以指定整个索引的分词器，也可以设置mapping映射额外指定字段独有的分词器

分析器有三部分组成：

****Character filters （字符过滤器）****：匹配特定的内容将内容进行转换(比如将&号转换为and)，一个分析器可以有0个或者多个过滤器

****Tokenizer （分词器）****：按照特定规则将文本内容拆分为独立的词条（比如按照标点符号，常用语法结束词拆分）

****Token filters （token过滤器）****：英文词条的小写化，删除a、then、an等无用词条，增加如jump、leap词条、修改词条，一个分析器可以有0个或者多个token过滤器

### es中内置的分析器：

* Standard Analyer 默认分词器，按词切分，小写处理
* Simple Analyer 按照非字母切分（符号被过滤），小写处理
* Stop Analyer 小写处理,停用过滤词（the, is , a）
* Whitespace Analyer 按照空格切分，不转小写
* Keyword Analyer 不分词，直接将输入当作输出
* Pattern Analyer 正则表达式，默认 \W+(非字符分隔)
* Language 提供30种分词器
* Customer Analyzer 自定义分词器

### 第三方分析器：

Ik分词器

Ik分词器是一个中文的分词器，可以根据中文的语法进行分词，也可以创建自定义的dict

后缀的词汇文件，然后在IKAnalyzer.cfg.xml中引用。也可以引用互联网上的词汇库。

## 分片

由于物理硬件限制，单个es节点数据存储能力有限，并且数据存储太多则查询会变慢。所以es允许对数据进行分片存储,分片可以存在于不同的es节点上（单一节点也可以设置多个分片），同一个索引可以存储在不同的分片和es节点上，同一个查询请求会在每个分片 中执行。索引创建时确定分片数量后不能再更改主分片数量，但是可以修改从分片数量。每一个分片都是独立的lucene索引都可以对外提供服务。

分片类别：

number\_of\_shards：主分片(默认值1) number\_of\_replicas：从分片（主分片数据的备份，用于灾难后数据恢复，以及处理数据检索的读请求，默认值1）

数据存储在那个分片：

数据存储时会根据数据的\_id 属性计算hash值分散到具体的分片上，计算公式：hash(routing) % number\_of\_shards

## 倒排索引

正排索引：以数据的id作为索引的key。

优点：索引好维护（新增数据时增加一个key即可）；

缺点：关键词查询时效率低（以数据内容检索时需要扫描整个索引数据）

倒排索引：具体数据项分词后作为索引的key。

优点：关键词查询时效率高

缺点：索引的维护成本高。

es中的索引：

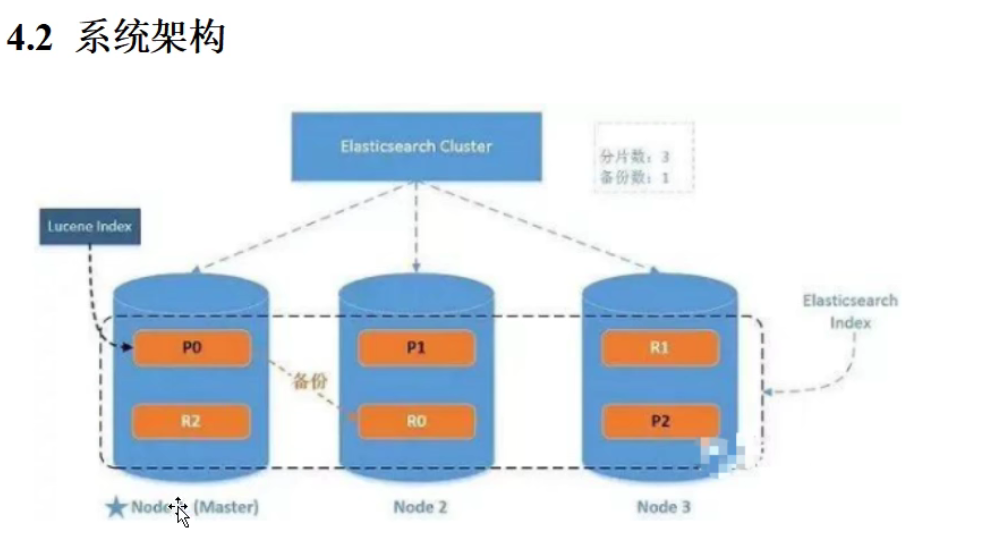
es中的索引为（哈希表+链表）存储，一个节点内分为**term dictionary**（具体数据的分词）和**Posting list TF**（分词出现的文档id）。

es中未使用B+树索引原因：

1. es需要进行全文检索，所以关键字会比较大，会消耗内存，导致b+树每个节点存储的索引数少，数据量大时会导致树高度增大，查询效率低下。
2. B+树对匹配索引时遵从最左匹配原则，es再检索时很容易索引失效。

数据写操作：

## ES集群



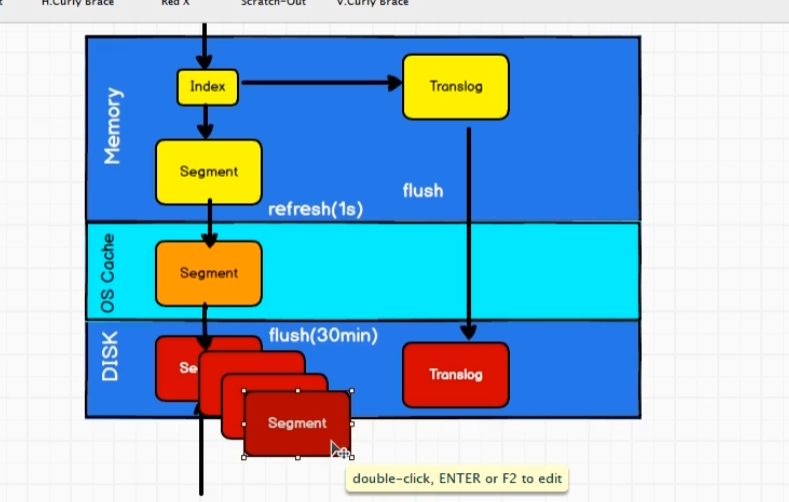
数据写过程:

任意一节点可以作为我们使用时连接的服务器，写数据时会根据路由算法计算分片把数据发送到对应分片的服务器上，然后再进行副本数据的同步。

数据查过程：

客户端连接的协调节点会计算分片位置，然后轮训所有此分片以及副本，找到可以相应的节点，然后查数据。

## 数据存储过程：



存入es的数据会先在内存中构建好lucene的索引对象，但是内存中的数据是不能直接用于检索的。es的搜索为近实时搜索（实时搜索：数据一提交还在内存中就可以进行检索）。es为了保证数据检索的效率会先将内存中的数据refresh（默认1秒钟一次）到系统的文件缓冲区，文件缓冲区的数据可以用于检索（lucene使用此内存中的数据进行检索）。文件缓冲区数据在固定间隔时间后会flush（默认30分钟）到磁盘中。文件缓冲区的数据可能会在系统故障时丢失，于是又引入了Translog记录操作日志到磁盘中。

## 数据查询：

数据查询分为两个阶段Query then fetch

当数据查询请求到来时，协调节点会广播给此索引的所有分片以及副本，然后这些分片和副本返回自己分片中所有命中文档的id和排序值并组建到一个内存中的列表中，然后协调节点会从中挑选数据，再次发送get请求到各个分片中然后取回所需要查询的数据返回给客户端

## 并发访问数据冲突：

es数据访问问多线程并发，会出现数据安全问题，为此es支持对数据上锁。

乐观锁：

方法1：在更新数据时增加版本号(更新请求带上参数：if\_seq\_no=2&if\_primary\_term=1)，如果版本号不一致则更新失败。

方法2：也可以指定数据中可以作为版本号的字段为版本号，如使用version字段作为版本号（更新请求带上参数：version=1&version\_type=external）

## es常用工具：

Kibana:可视化数据管理工具

docker 安装：docker run -d --name=kibana --restart=always --privileged=true -p 5601:5601 -v /home/docker/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml kibana:7.7.0



elasticsearch -head：集群管理工具

Spring-data

## Es性能提升

硬件选择

1. 使用固态硬盘 2.使用多块硬盘通过path.data配置 3.不要使用远程挂载的硬盘

合理设置分片

分片过多会消耗资源，太少又会导致服务器压力过大。合理的增加分片能增加系统吞吐量。

路由选择

查询时如果知道路由值则查询时附带路由参数（routing），此时查询数据不会去所有的分片去查数据

批量数据提交

通过Bulk api去对数据进行批量操作

在满足业务需求前提下，适量增加refresh和flush时间

减少副本数量：

Es保存时副本也会执行分析和索引行为，而不是从主分片复制数据，所以数据量大时可以先关闭副本，数据增加完毕后再打开副本

## Es如何保证读写一致性（高并发下）

1. 乐观锁使用版本号保证旧数据不会覆盖新数据