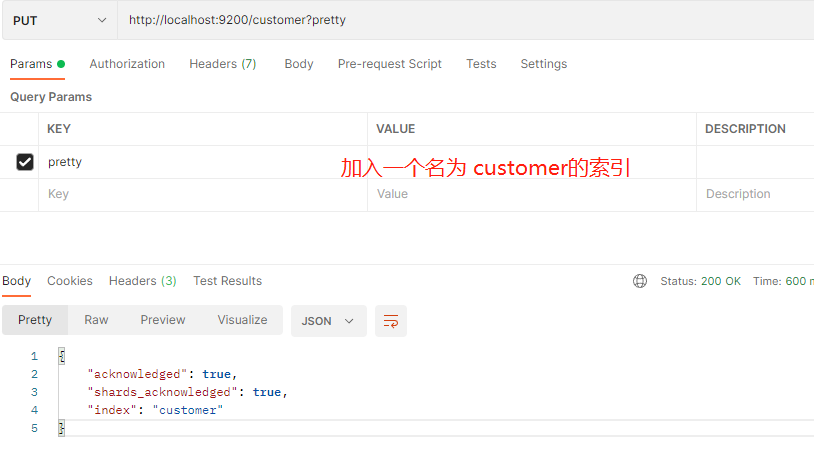
GET /\_cat/health?v

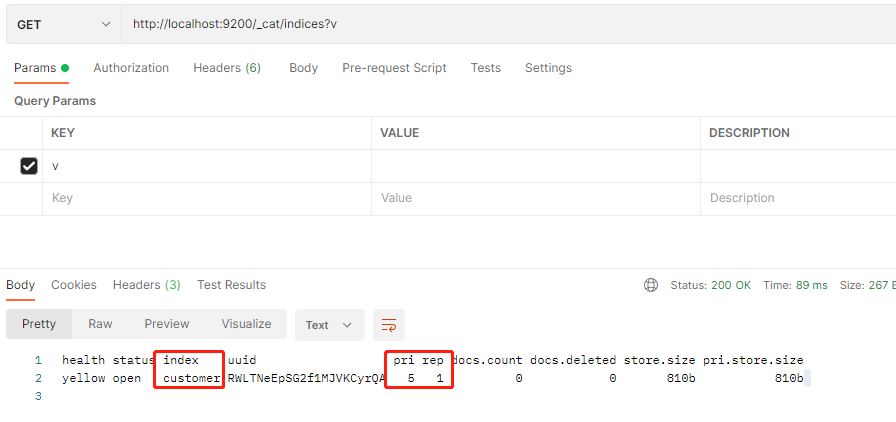
GET /\_cat/nodes?v

PUT /customer



检查索引:

GET /\_cat/indices?v



it has 5 primary shards and 1 replica (the defaults) and it contains 0 documents in it.

放点东西进去:

PUT /customer/external/1

{

"name": "John Doe"

}

取点东西:

/customer/external/1

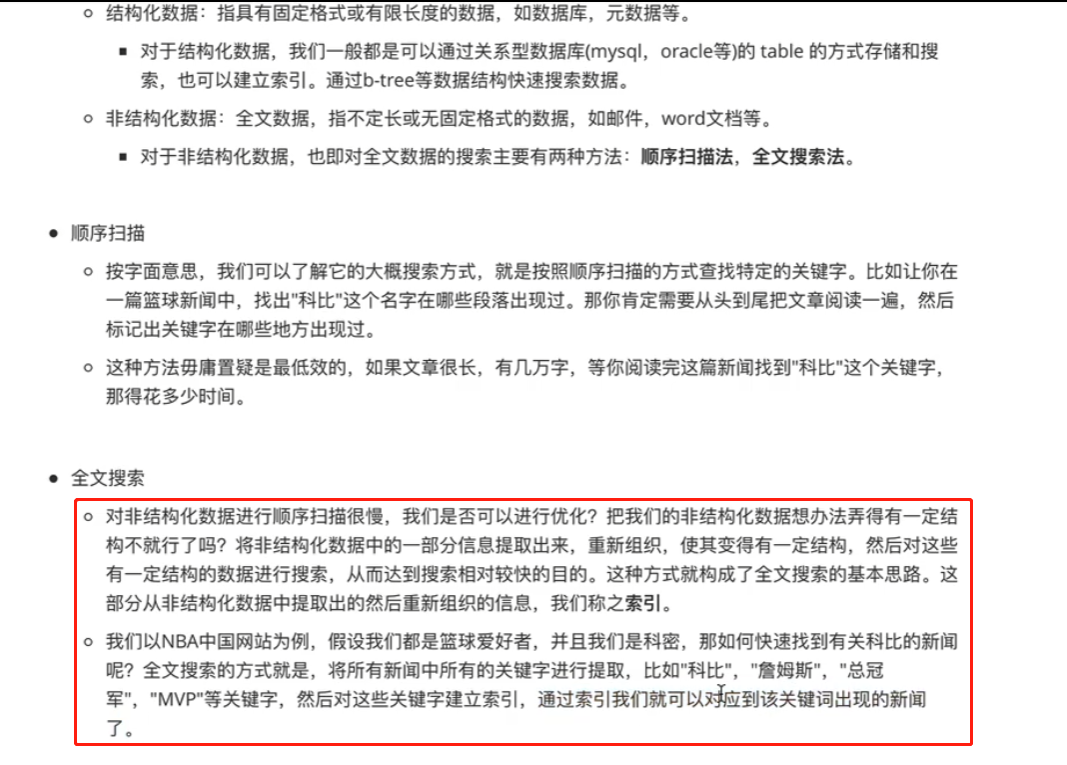
1. 结构化数据和非结构化数据的搜索:

结构化数据，就是有元数据进行约束的那种，查找起来会很方便。

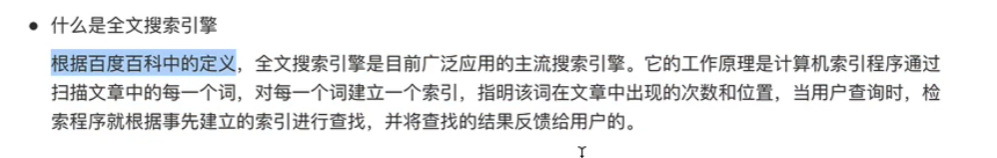
非结构化数据，有两种搜索方式，

1.无脑顺序搜索（效率极低）

2.全文搜索（建立索引，提高查找效率）

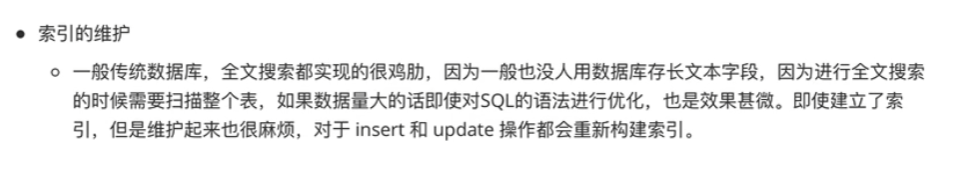


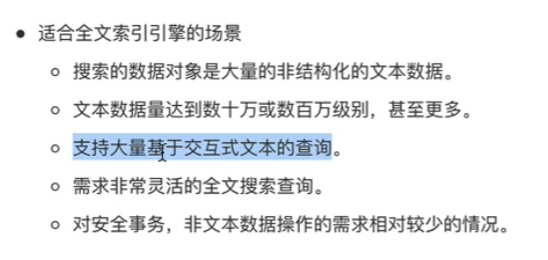
主要还是建立索引加速搜索。



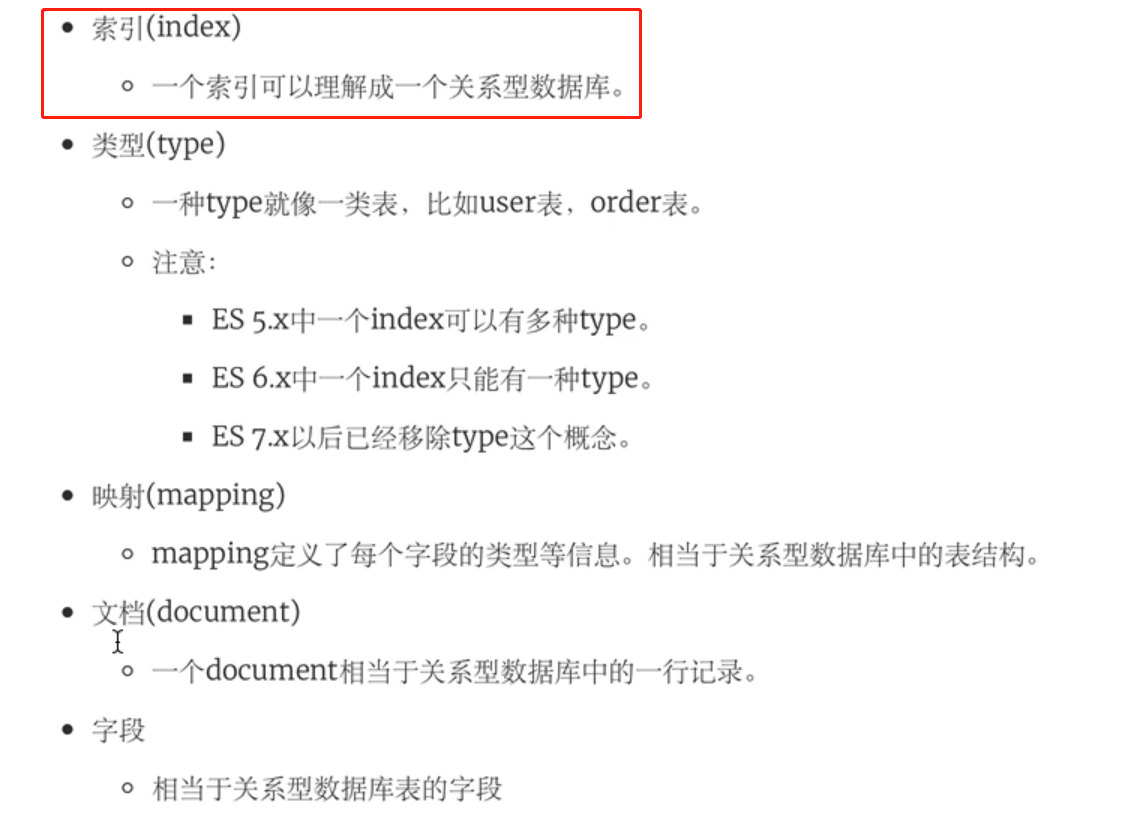
扫描每一个词，并对每一个词都进行建立索引，记录出现的次数和位置

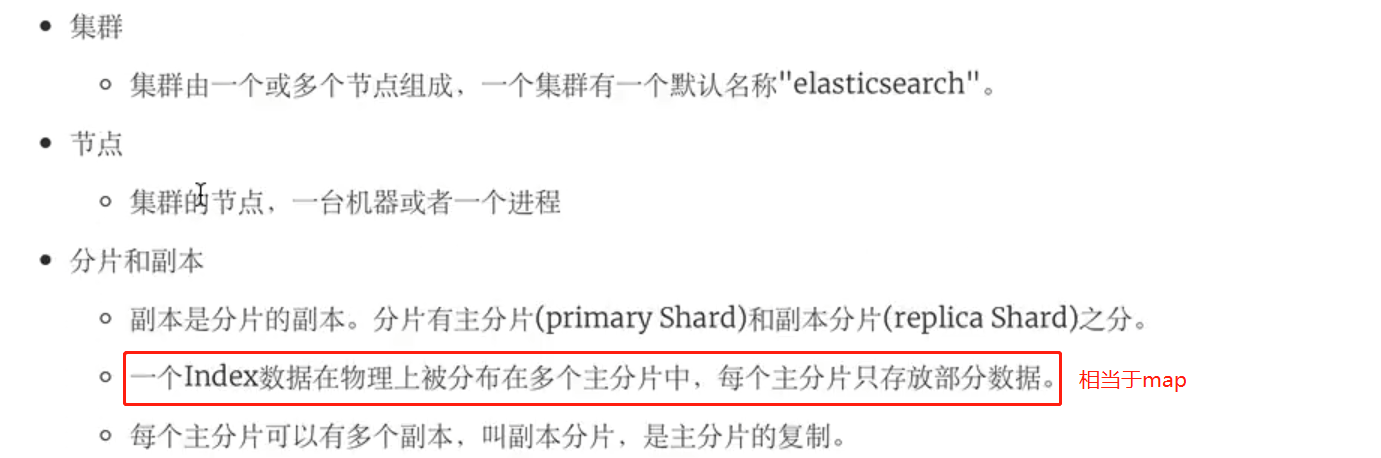
1. 为什么不用数据库做搜索？
2. 数据库的数据一般都是结构化数据，查找不灵活，而且效率绝对不如专业的搜索引擎。数据量一大，而且还不走索引的话，效率就特别的低。
3. 关键字补充的功能





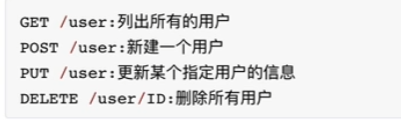
对应数据库的概念:



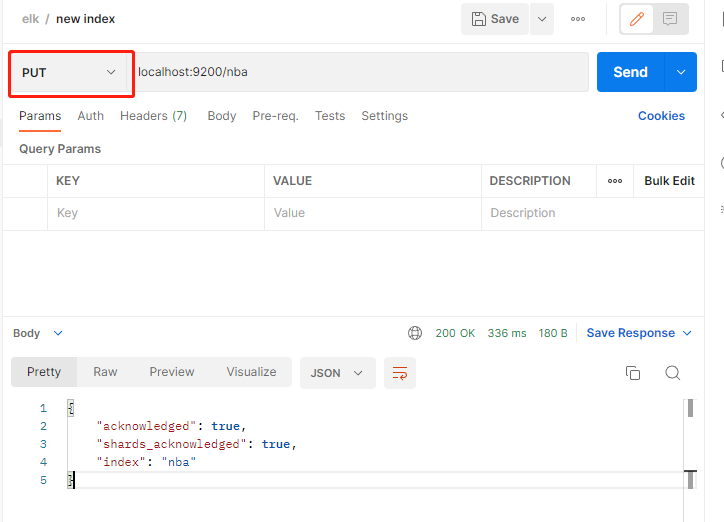


1. Restful 风格的操作:

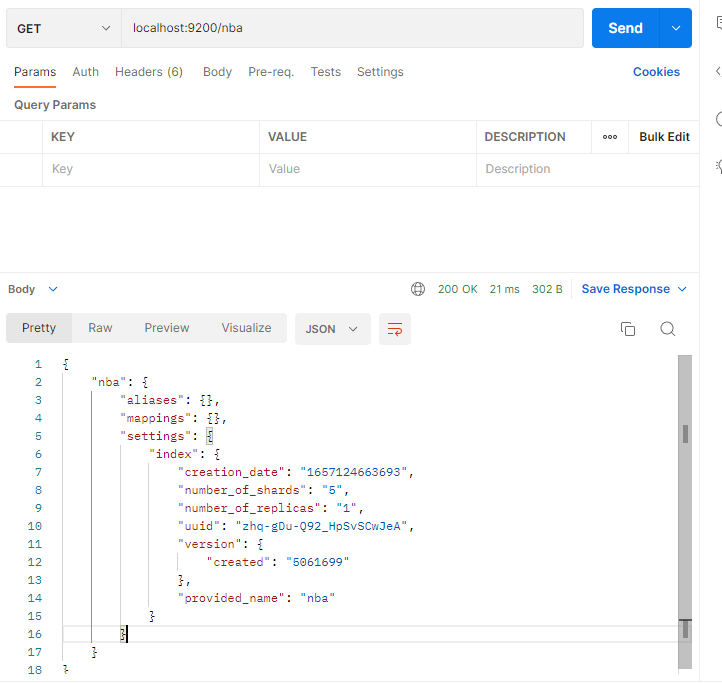




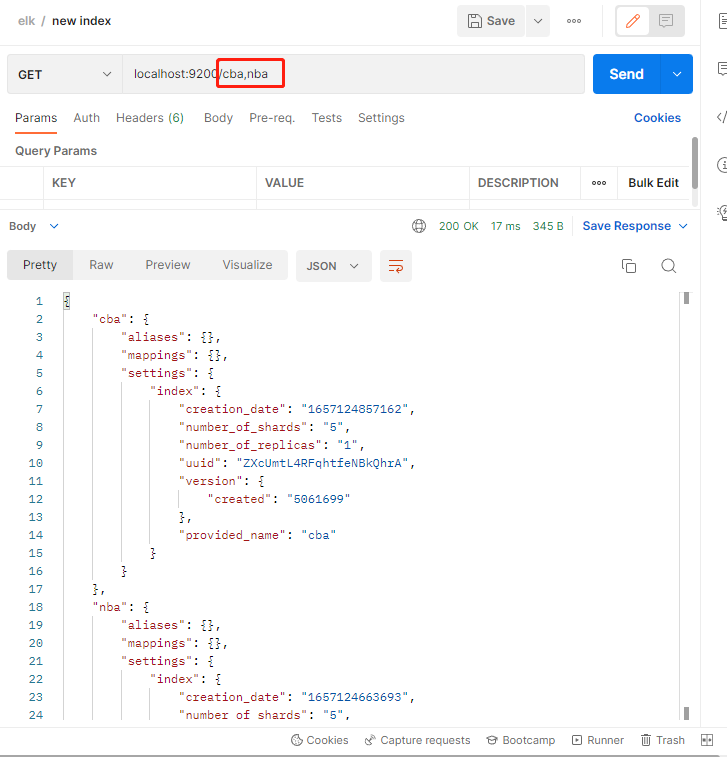
## 4.新建索引（数据库）:



获取索引：

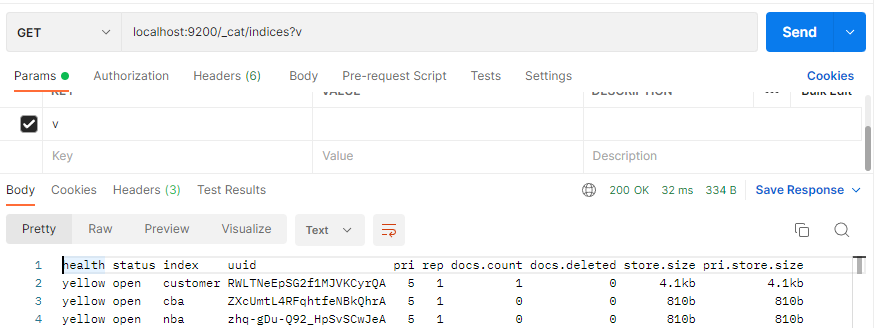


同时获取多个索引的做法:



或者\_all

或者/\_cat/indices



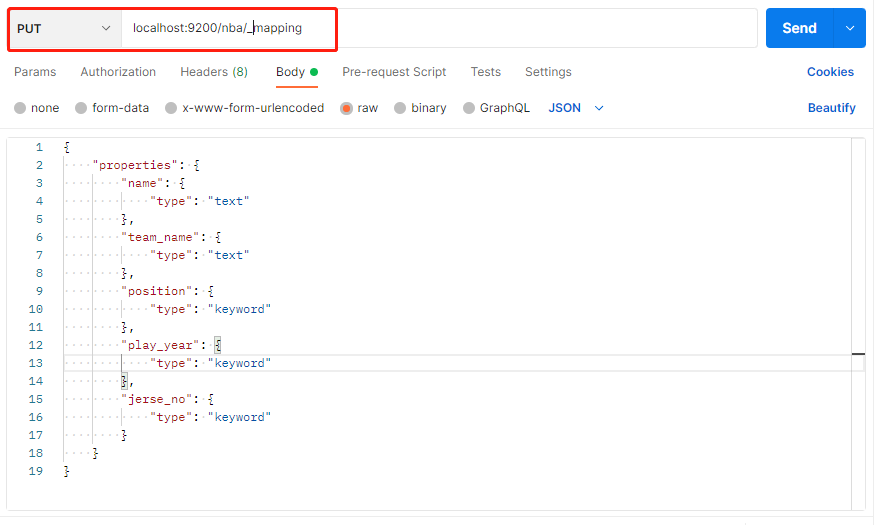
关闭，开启一个索引:

localhost:9200/nba/\_close

localhost:9200/nba/\_open

注意都是post请求

为表设置mapping(即table filed):

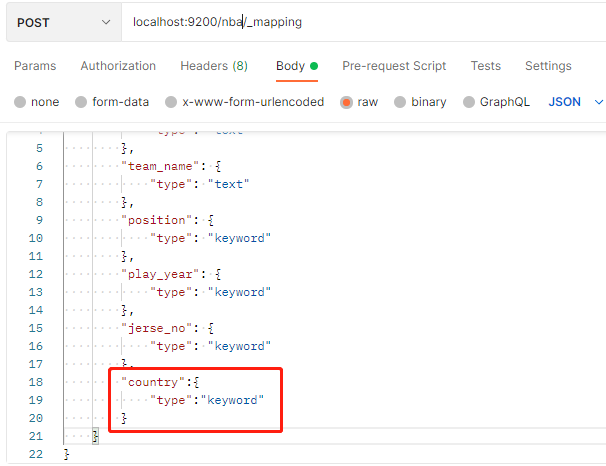


返回所有\_mapping all:

localhost:9200/\_all/\_mapping

修改，新增属性的话，用post方式即可。

而且只能新增字段，字段的type值是不能被修改的。只能重新建一个索引！



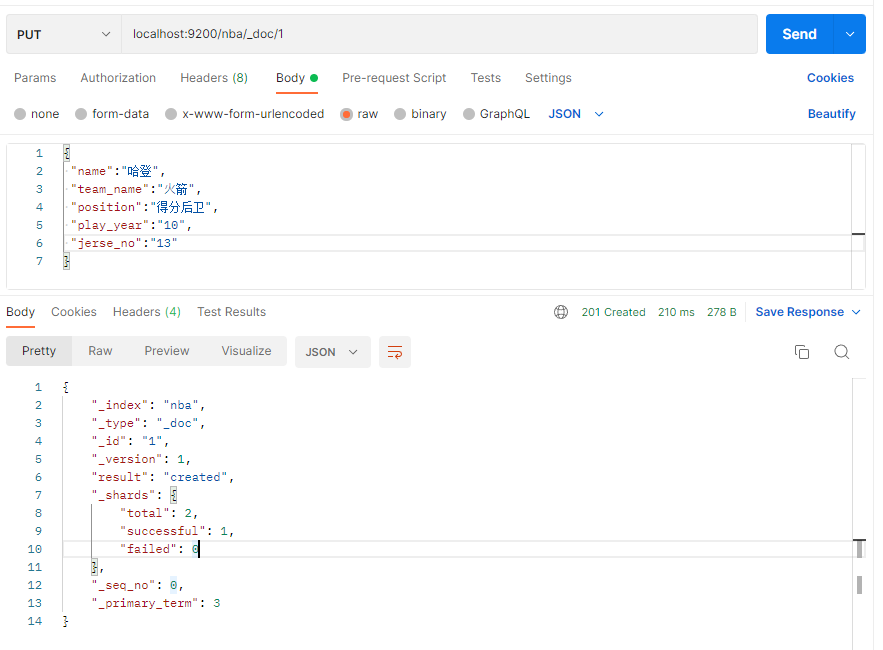


## 5.创建文档(doc/record)

1.新增文档:

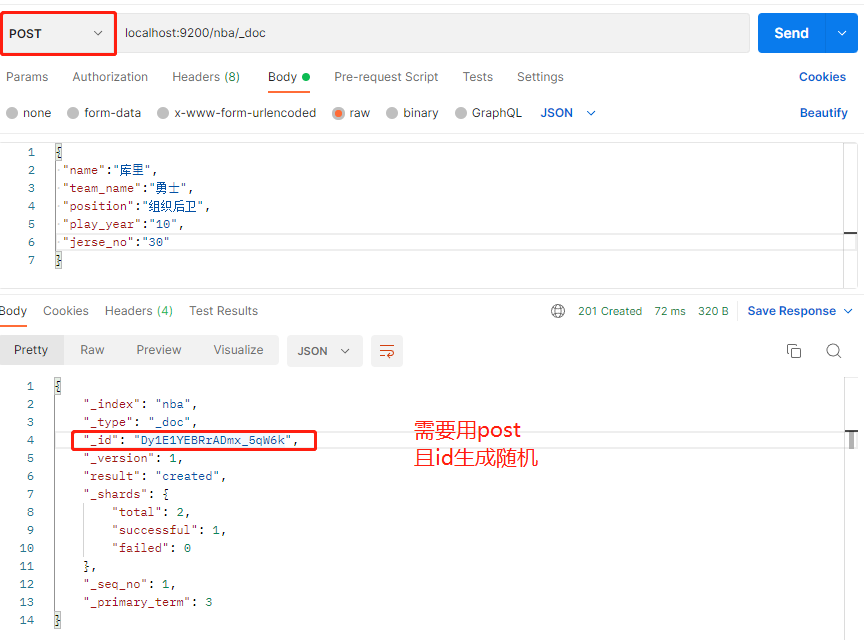
PUT localhost:9200/nba/\_doc/1 (指定id)

Get方式获取，同链接



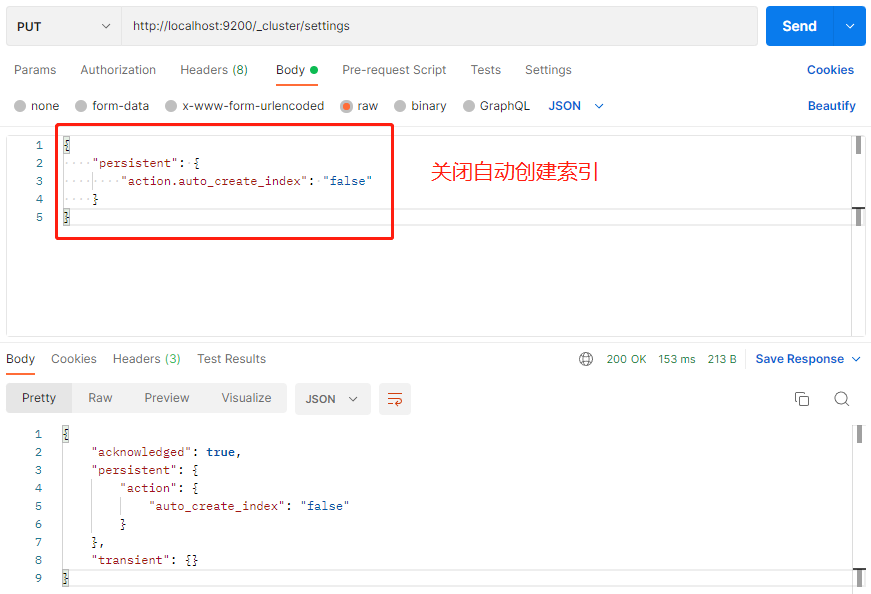
localhost:9200/nba/\_doc (不指定id)

需要用post，且id随机生成

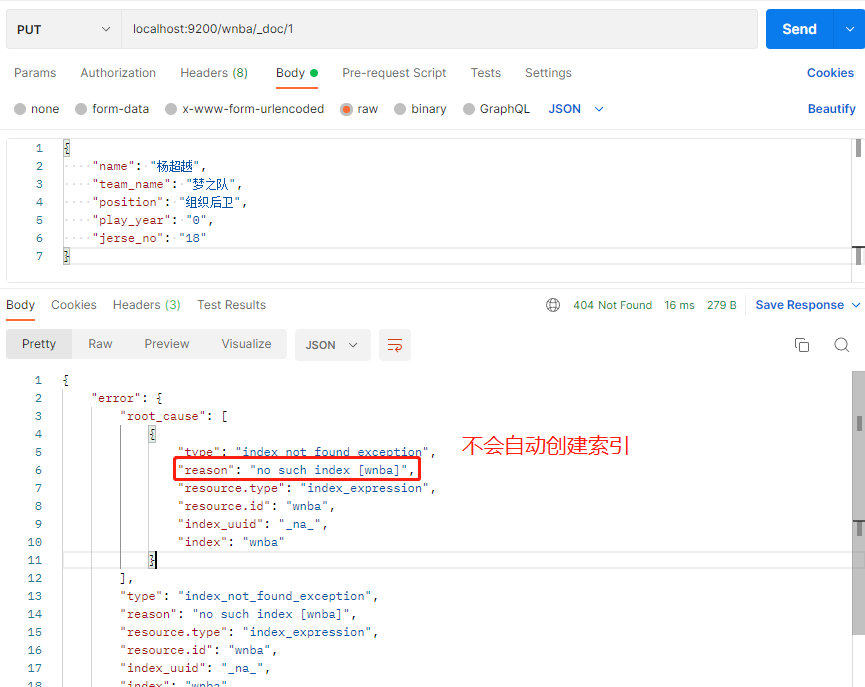


2.开启/关闭自动创建索引

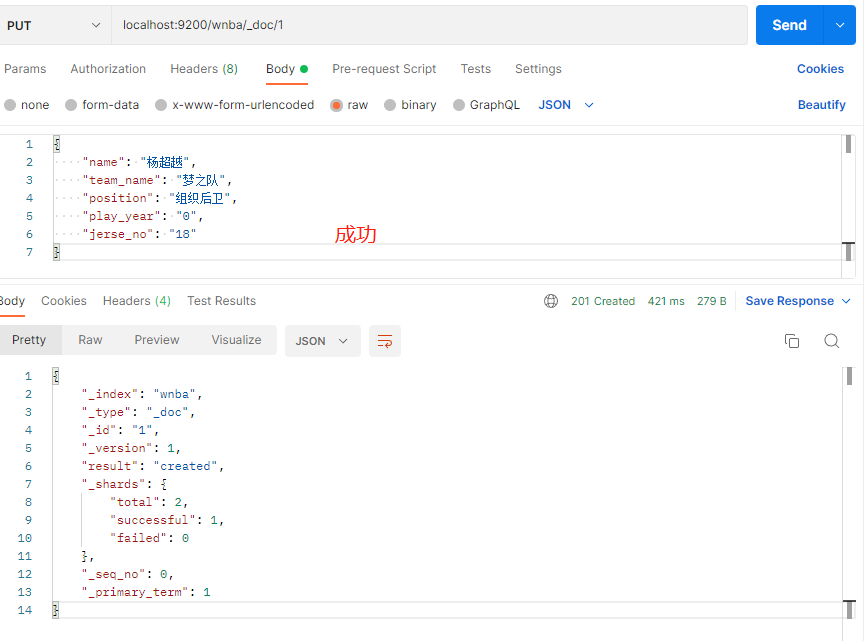




测试,新建一个doc到wnba（之前不存在这个索引）



开启后再测试:



防止覆盖掉已有的doc: 加上参数op\_type=create

3.查询文档:

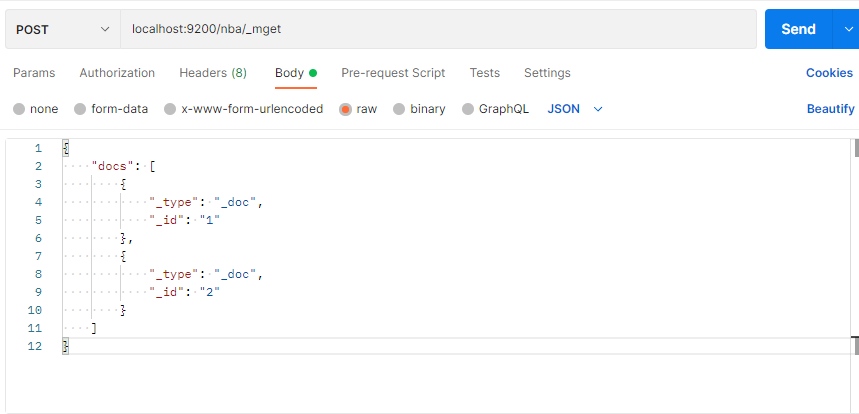


Link越详细，json需要的参数就越少

localhost:9200/nba/\_doc/\_mget

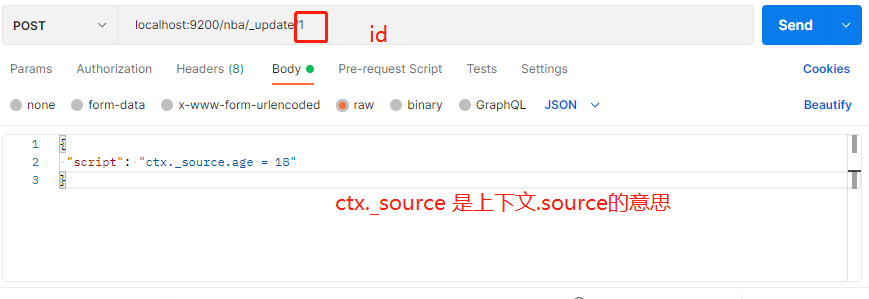
localhost:9200/nba/\_mget

查出来的东西都是一样的



1. 新增一个attribute:

{ "script": "ctx.\_source.age = 18" }



删除:

{ "script": "ctx.\_source.remove(\"age\")" }

带参数的更新：



如果doc不存在，update自动填充的默认值写法:



若存在，就+4了。

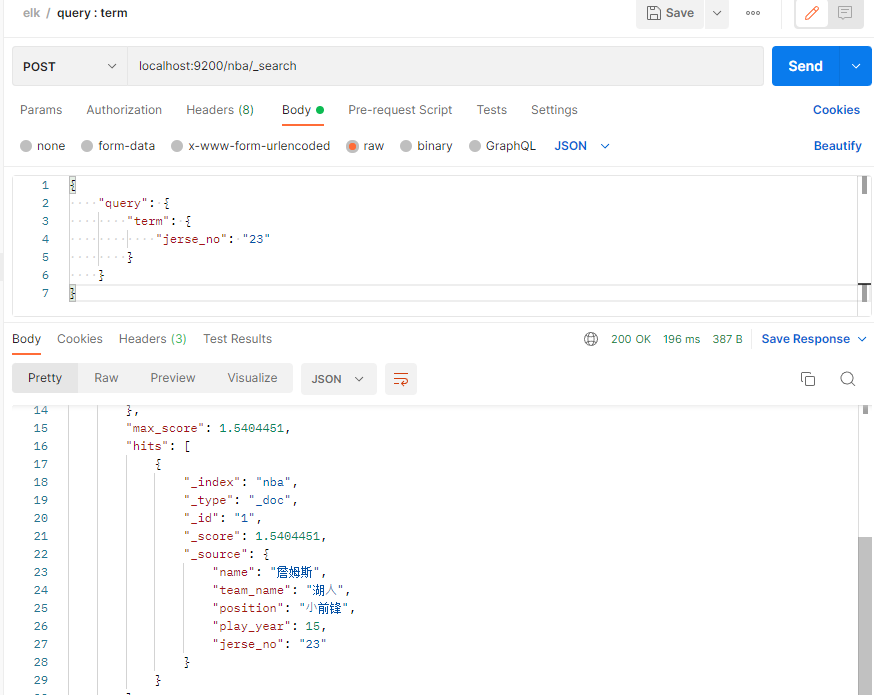
1. term查询:

词条查询：通过某个字段，单点查询

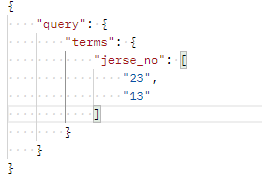
是针对属性值为keyword的查询方式



拿jerse\_no举例：



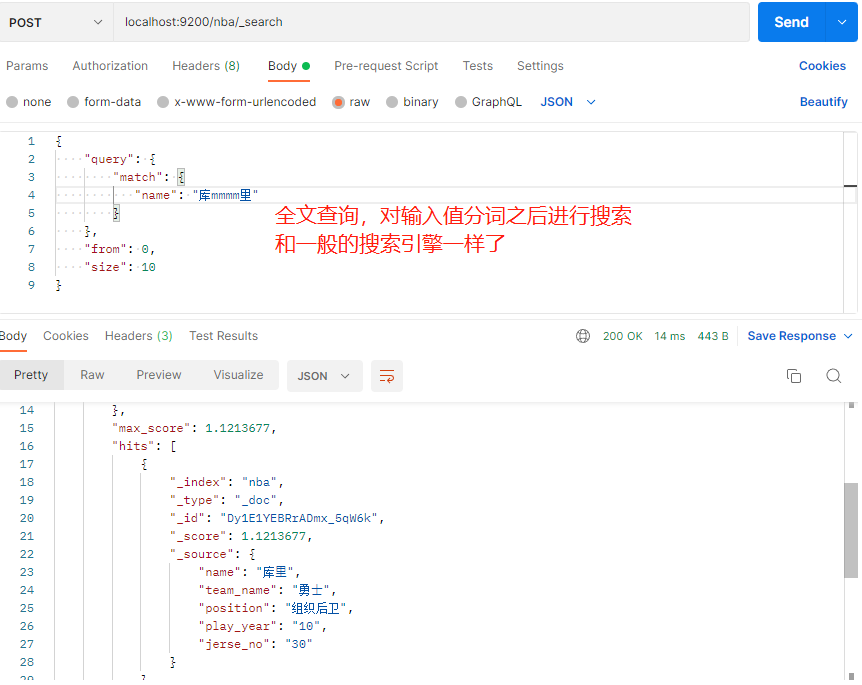
多条一起：



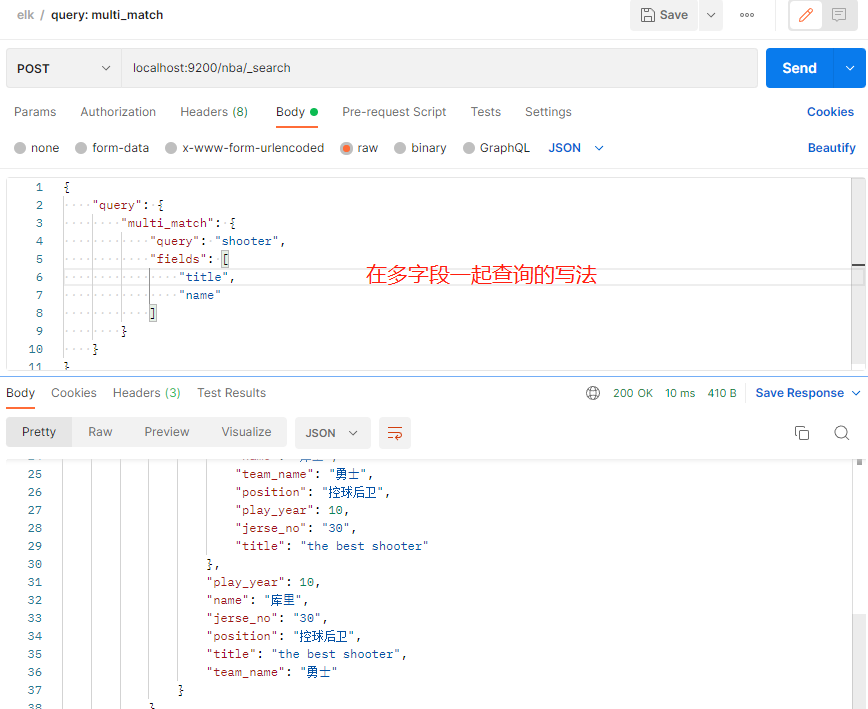
## 6.match查询:

Match\_all +空内容可以查所有

Match部分:



Multi\_match:

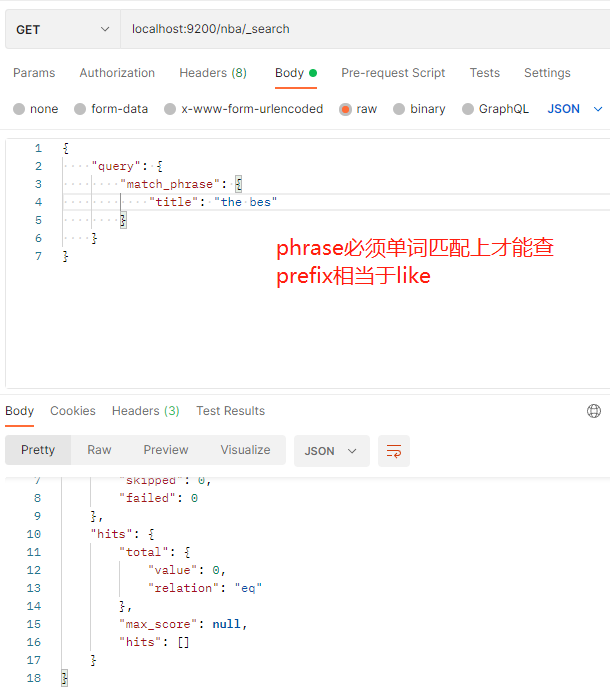


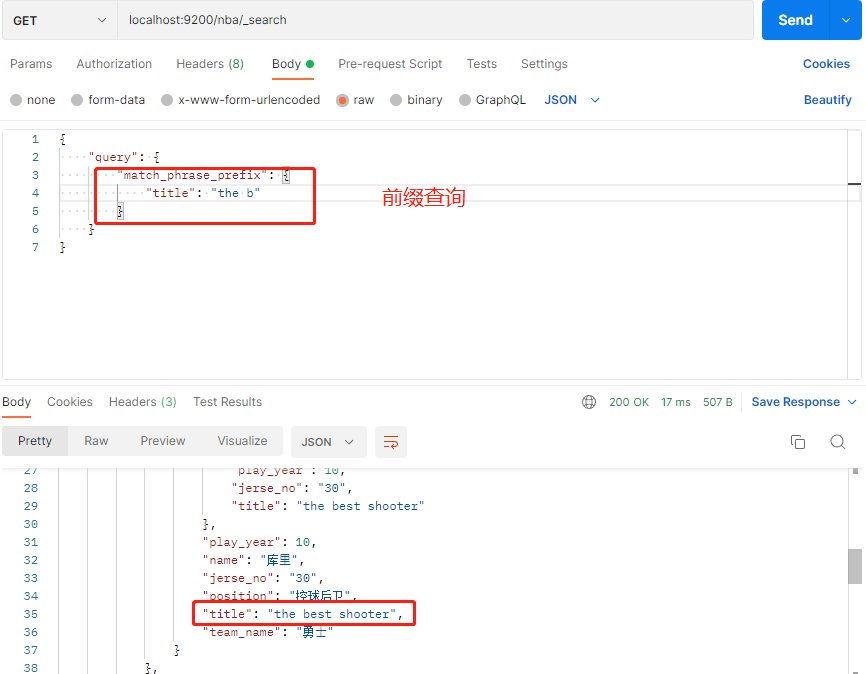
Match\_phrase/match\_phrase\_prefix:

Match\_phrase:

字面理解，只有单词完全匹配上，才会进行查询

Prefix:带前缀的精准查询





分词器：

{

    "analyzer":"standard",

    "text":"the best shoter"

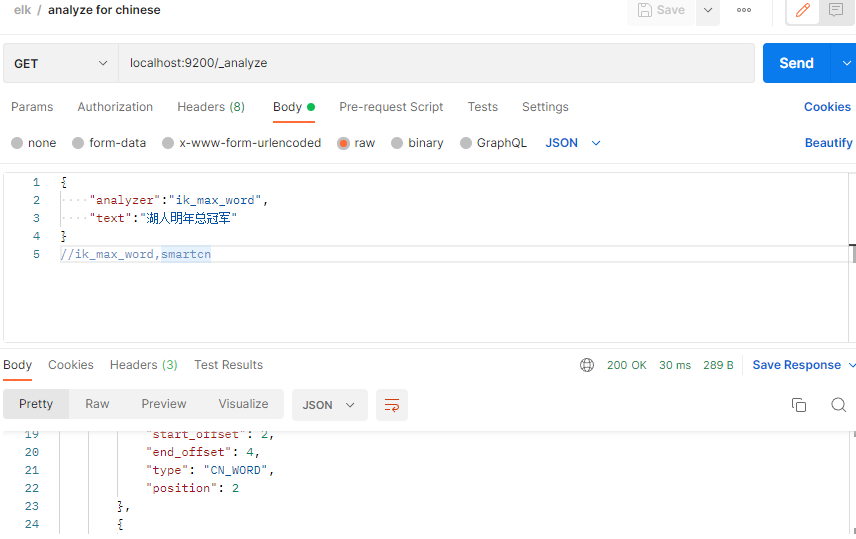
}

//whitespace,english,simple,pattern

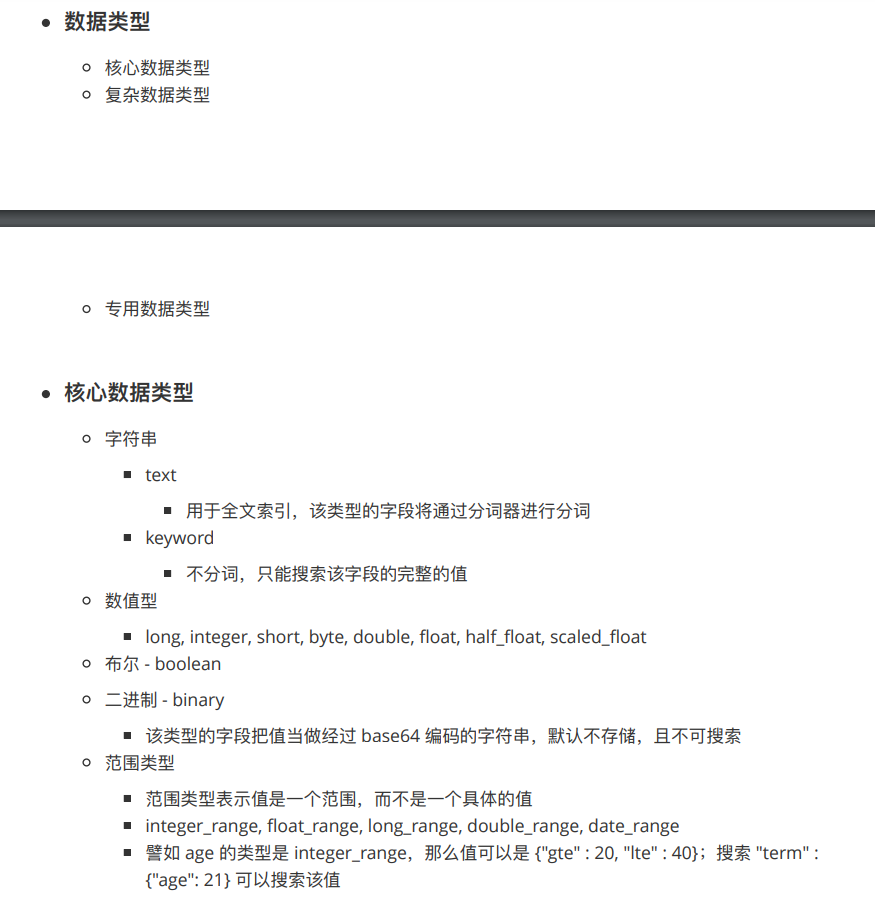
多种分词规则，可以对指定的text字段进行分词规则设定。



支持中分分词:

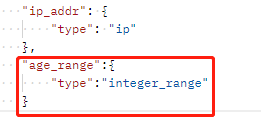


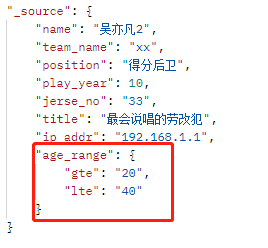
## 7.数据类型



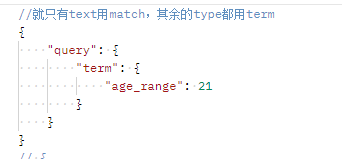
范围类型很好用:

用integer\_range做例子:





指范围20-40



这样就能查出来了，21在范围里面。



Date的三种格式如下:



还有复杂类型:

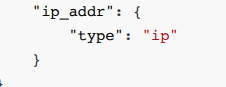
对象中对象，数组



访问规则:



特殊字段:ip



查找:

