使用查询重写字段值（敏感词搜索）

这个功能可以解决敏感词搜索问题。

场景：如果创建一个与Mapping字段相同名称的运行时字段名，那么这个运行时字段会覆盖Mapping字段，并按照你定义的脚本返回这个字段值。

要求：敏感词不能展示，但是可以被搜索。

如，定义敏感词为：“敏感”，此时搜索”敏感“二字可以将这个文本搜索出来，原始文本不变（毕竟敏感词多了全是*已经认不出原文了），但不能将敏感两个字在搜索结果中展示出来。

这条数据 {"text":"我是一个敏感的内容"} 搜索关键词“敏感”时，这条数据应展示为”我是一个**的内容“。

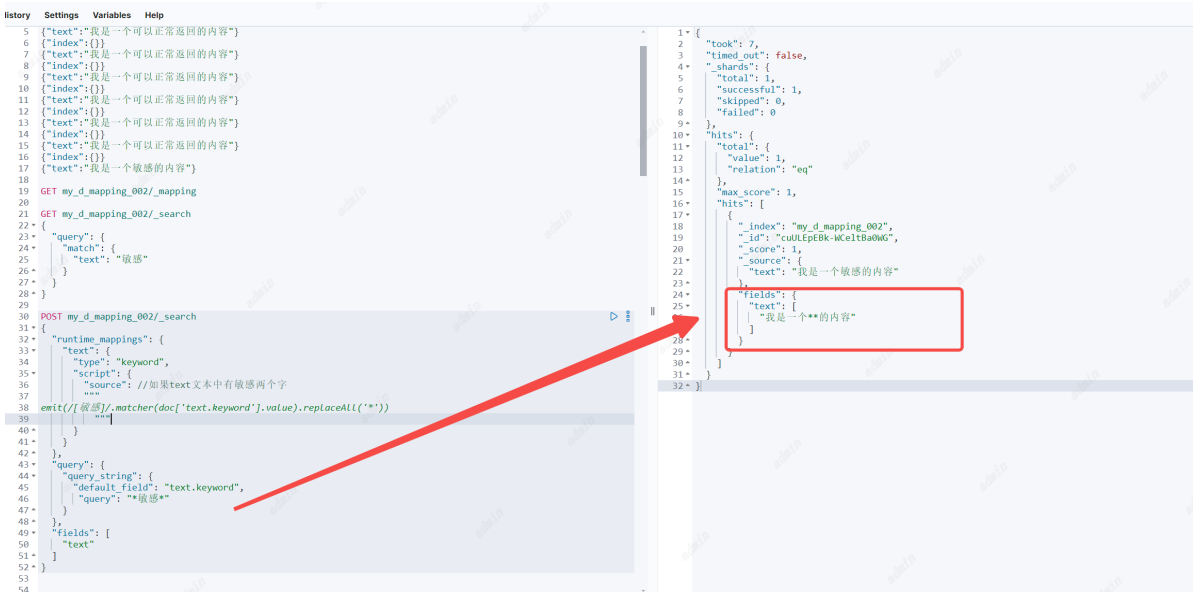
```
POST my_d_mapping_002/_bulk?refresh=true
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个敏感的内容"}  //搜索关键词“敏感”时，这条数据应展示为我是一个**的内容。
```

使用运行时字段在查询时覆盖掉原始原始字段text

```
POST my_d_mapping_002/_search
{
  "runtime_mappings": {
    "text": { //运行时字段与Mapping中需要被覆盖的字段名称相同。
      "type": "keyword",
      "script": {
        "source": //如果text文本中有敏感两个字
        """
emit(/[敏感]/.matcher(doc['text.keyword'].value).replaceAll('*'))  //表示如果
text.keyword中包含“敏感”一词时，将“敏感”词变为”*“
          """
      }
    }
  },
  "query": {
    "query_string": {
      "default_field": "text.keyword",
       "query": "*敏感*"
    }
  },
  "fields": [
    "text"
```

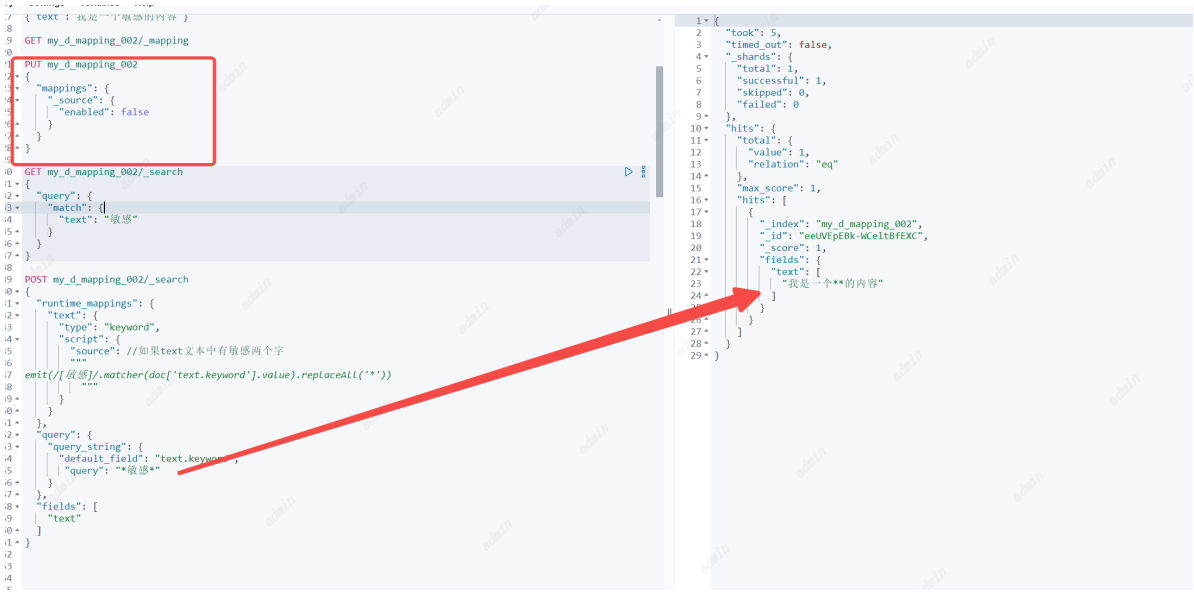
```
    ]
}
```

结果可以看到这条数据被查询出来，且fields里的text已经脱敏展示。原本的_source内容没有改变。



如果需要原始*source不展示，在Mapping中这时*`source`参数为false

```
//注意，需要在创建索引时设置。
PUT my_d_mapping_002
{
  "mappings": {
    "_source": {
      "enabled": false
    }
  }
}
```



对比_update_by_query

脱敏处理还有一个API看起来好像也能做到那就是update_by_query。

这个API，顾名思义在查询时更新。那一起来看看这个区别。

```
PUT my_d_mapping_003/_bulk?refresh
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个可以正常返回的内容"}
{"index":{}}
{"text":"我是一个敏感的内容"}
```

调用_update_by_query,jiang 字段text进行敏感字替换。同样的这条数据 {"text":"我是一个敏感的内容"} 搜索关键词“敏感”时，这条数据应展示为”我是一个**的内容“。

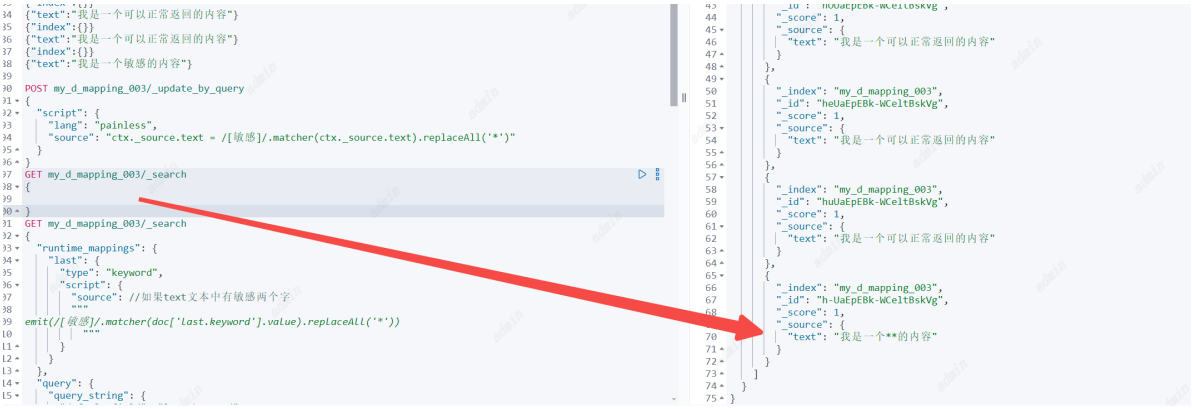
```
POST my_d_mapping_003/_update_by_query
{
  "script": {
    "lang": "painless",
    "source": "ctx._source.text = /[敏
感]/.matcher(ctx._source.text).replaceAll('*')"
  }
}
```

执行查询语句

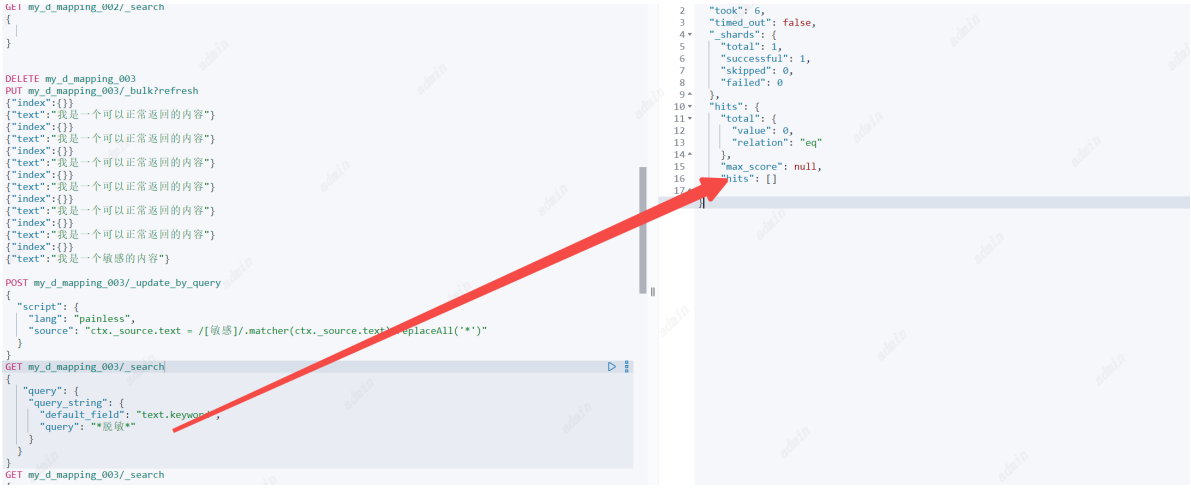
```
GET my_d_mapping_003/_search
{
   "query": {
    "query_string": {
      "default_field": "text.keyword",
      "query": "***"
    }
  }
}
```

看起来数据被查出来了还脱敏了。

实际上仔细想想我们之前的业务需求，除了需要脱敏，还需要保证原始文本内容不变，那么就看一下原始文本内容。



原始文本已经被替换成了**。细心的人已经发现了，我第一次搜索的不是”脱敏“，而是”*“，当我搜索”脱敏“一词是无法将文档搜索出来的。



所以可以看到，_update_by_query这个API实际上是在搜索时将我们的文档进行了替换并查询出来，而使用运行时字段对原Mapping进行替换并不会改变原始的文档内容，还可以根据自己的需要判断是否返回原始内容。

Painless 语法继续扩展

通过刚才查询出来的内容可以看到这条数据 {"text":"我是一个敏感的内容"} 搜索关键词“敏感”时，这条数据应展示为”我是一个**的内容“。有两个*。说明语法将“敏感”一词中的两个字分别进行了替换，但很多时候敏感的不是词是一个词或者一句话。所以我们需要脱敏后展示”我是一个*的内容“

最有可能对结果产生影响的就是脚本里的内容了，让我们看脚本里都有哪些内容。

```
emit(/[敏感]/.matcher(doc['text.keyword'].value).replaceAll('*'))
```

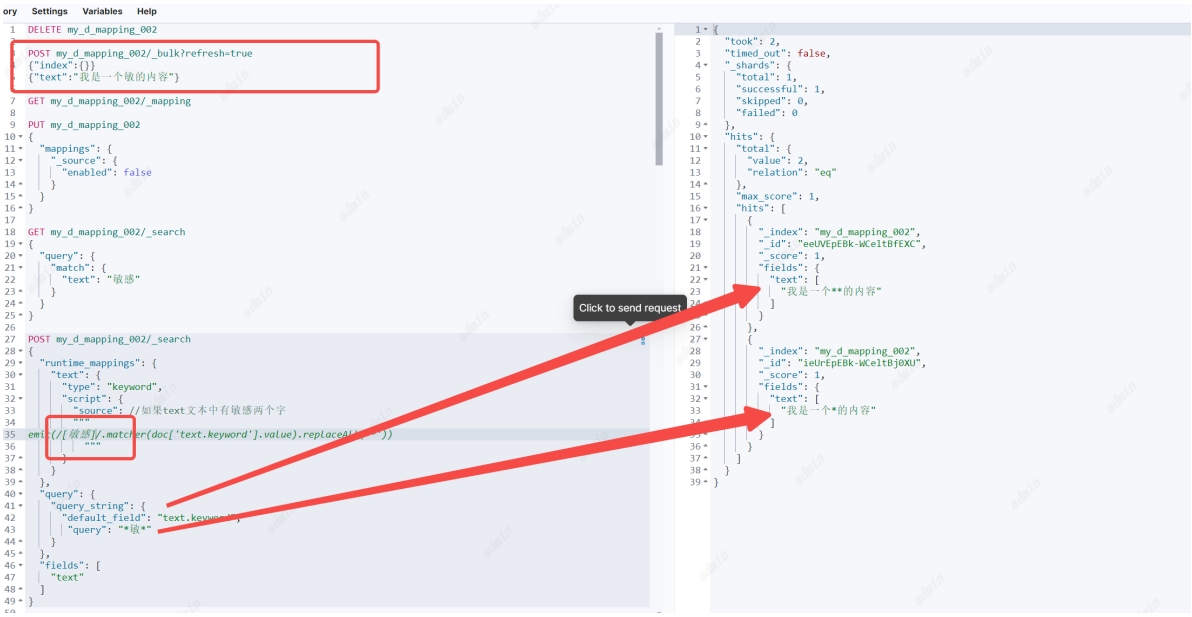
1. emit()：函数处理得到的返回值会赋值给我们定义好的运行时字段
2. /pattern/：是一个匹配的模型，在painless语法中使用双\表示一个模型，这里也支持Java中的正则表达式。
3. matcher()：匹配函数
4. doc['filed'].value：表示获取文档中这个字段的值，这里为什么用keyword呢，因为运行时字段支持的文本类型只有keyword。
5. replaceAl()：替换函数。

看了这些可以看到问题出现在匹配上，`/[敏感]/.matcher`将文本匹配成两个字分别进行后续的替换操作。那为什么不是词呢，关键就是`[]`这个中括号，敏感的程序员一下子就能想到，这表示数组。

没错，这里`/[敏感]/`在Painless 语法中被认为是由`敏`和`感`两个字组成的数组，所以替换后会有两个*

可以验证一下，这里我再加入一个数据，这时我搜索敏，可以看到新的数据也被替换了

```
POST my_d_mapping_002/_bulk?refresh=true
{"index":{}}
{"text":"我是一个敏的内容"}
```

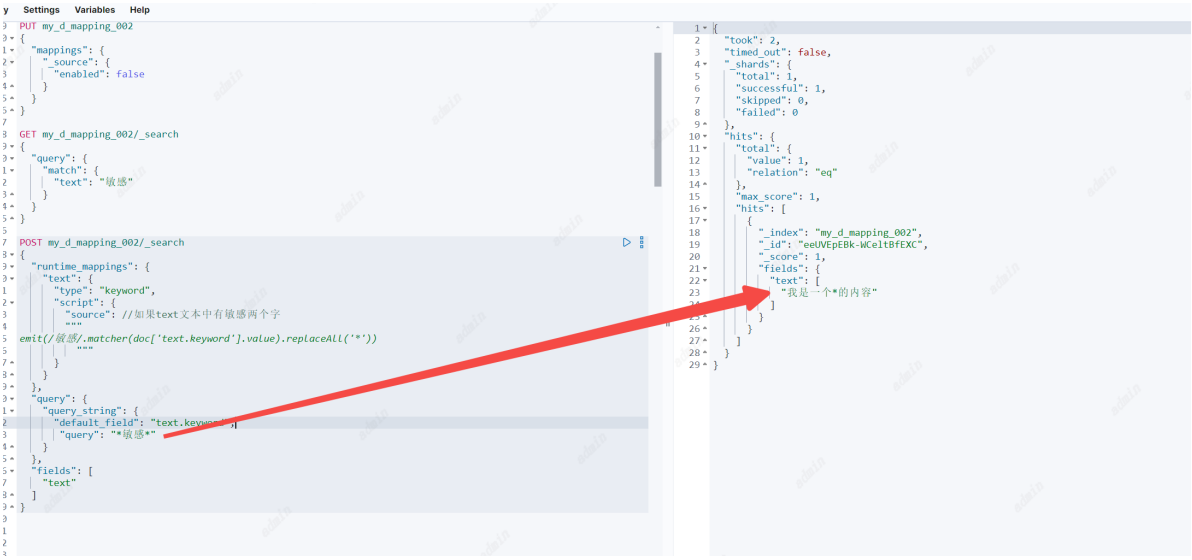


那么如何修改呢，当然就是去掉中括号了

```
POST my_d_mapping_002/_search
{
  "runtime_mappings": {
    "text": {
      "type": "keyword",
      "script": {
        "source": //如果text文本中有敏感两个字
        """
emit(/敏感/.matcher(doc['text.keyword'].value).replaceAll('*'))
          """
      }
    }
  },
```

```
  "query": {
    "query_string": {
      "default_field": "text.keyword",
       "query": "*敏*"
    }
  },
  "fields": [
    "text"
  ]
}
```

再次搜索，新加的数据并没有被脱敏。



问题：如果有多敏感词呢？