term 查询

term 查询是代表完全匹配，搜索之前不会对你搜索的关键字进行分词，直接拿关键字去文档分词库中匹配内容

```
POST /sms_logs_index/sms_logs_type/_search
{
 "from": 0, #limit from,size
 "size": 5,
 "query": {
 "term": {
 "province": {
 "value": "河北"
 }
  }
  }
}
```

terms查询

terms 和 term 查询的机制一样，搜索之前不会对你搜索的关键字进行分词，直接拿 关键字 去文档分词库中匹配内容

是针对一个字段包含多个值得运用，也可针对 text，只是在分词库中查询的时候不会进行分词

```
terms: where province = 河北 or province = ? or province = ?(类似于mysql中的 in)
POST /sms-logs-index/sms-logs-type/_search
{
 "query": {
 "terms": {
 "province": [
 "北京",
 "晋城"
 ]
  }
  }
}
```

match查询

根据你查询字段类型不一样，采用不同的查询方式

实际底层就是多个term查询，将多个term查询的结果进行了封装

```
POST /sms_logs_index/sms_logs_type/_search
{
 "query": {
 "match": {
 "smsContent": "打车"
 }
 }
}
```

1. 查询的如果是日期或者是数值的话，它会根据你的字符串查询内容转换为日期或者是数值对等
2. 如果查询的内容是一个不可被分的内容（keyword），match查询不会对你的查询的关键字进行分词
3. 如果查询的内容是一个可被分的内容（text）,match则会根据指定的查询内容按照一定的分词规则去分词进行查询

布尔match查询

基于一个Filed匹配的内容，采用and或者or的方式进行连接

```
POST /sms_logs_index/sms_logs_type/_search
{
 "query": {
 "match": {
 "smsContext": {
 "query": "打车 女士",
 "operator": "and" #or
 }
 }
  }
 }
```

multi_match查询

match针对一个field做检索，multi_match针对多个field进行检索，多个key对应一个text

```
POST /sms_logs_index/sms_logs_type/_search
{
 "query": {
```

```
  "multi_match": {
  "query": "河北", #指定text
  "fields": ["province","smsContext"] #指定field
  }
  }
}
```

Prefix查询

前缀查询，可以通过一个关键字去指定一个Field的前缀，从而查询到指定的文档

```
POST /sms_logs_index/sms_logs_type/_search
{
  "query": {
  "prefix": {
  "smsContext": {
  "value": "河"
  }
  }
  }
}
```

fuzzy查询

模糊查询，我们可以输入一个字符的大概，ES就可以根据输入的内容大概去匹配一下结果(可以存在一些错别字)查询结果不稳定

```
POST /sms_logs_index/sms_logs_type/_search
{
  "query": {
  "fuzzy": {
  "corpName": {
  "value": "盒马生鲜",
  "prefix_length": 2 # 指定前几个字符要严格匹配
  }
  }
   }
}
```

wildcard查询

通配查询，与mysql中的like查询是一样的，可以在查询时，在字符串中指定通配符*和占位符?

```
POST /sms_logs_index/sms_logs_type/_search
{
 "query": {
 "wildcard": {
 "corpName": {
 "value": "*车" # 可以使用*和？指定通配符和占位符
 }
 }
 }
}
?代表一个占位符 ??代表两个占位符
```

range查询

范围查询，只针对数值类型，对某一个Field进行大于或者小于的范围指定

```
POST /sms-logs-index/sms-logs-type/_search
{
 "query": {
 "range": {
 "fee": {
 "gte": 10,
 "lte": 20
 }
 }
  }
}
```

regexp查询

正则查询，通过你编写的正则表达式去匹配内容

PS: **prefix**,**fuzzy**,**wildcar**和**regexp**查询效率相对比较低,在对效率要求比较高时，避免去使用

```
POST /sms_logs_index/sms_logs_type/_search
{
 "query": {
 "regexp": {
 "moible": "109[0-8]{7}" # 匹配的正则规则
 }
```

```
  }
}
```

深分页Scroll

ES对from+size有限制，from和size两者之和不能超过1w

from+size ES查询数据的方式：

1 先将用户指定的关键词进行分词处理

2 将分词去词库中进行检索，得到多个文档的id

3 去各个分片中拉去指定的数据 **耗时**

4 根据数据的得分进行排序 **耗时**

5 根据from的值，将查询到的数据舍弃一部分，

6 返回查询结果

Scroll+size 在ES中查询方式

1 先将用户指定的关键词进行分词处理

2 将分词去词库中进行检索，得到多个文档的id

3 将文档的id存放在一个ES的上下文中，ES内存

4 根据你指定给的size的个数去ES中检索指定个数的数据，拿完数据的文档id，会从上下文中移除

5 如果需要下一页的数据，直接去ES的上下文中，找后续内容

6 循环进行4.5操作

缺点，Scroll是从内存中去拿去数据的，不适合做实时的查询，拿到的数据不是最新的

```
# 执行scroll查询，返回第一页数据，并且将文档id信息存放在ES的上下文中，指定生存时间
POST /sms_logs_index/sms_logs_type/_search?scroll=1m
{
 "query": {
 "match_all": {}
 },
 "size": 2,
 "sort": [
 {
  "fee": {
  "order": "desc"
```

```
 }
 }
 ]
}
#查询下一页的数据
POST /_search/scroll
{
 "scroll_id": "_scroll_id",
 "scoll" :"1m" #scorll信息的生存时间
}
#删除scroll在ES中上下文的数据
DELETE /_search/scroll/scrill_id
```

delete-by-query

根据term，match等查询方式去删除大量的文档

需要删除的内容，是index下的大部分数据，不建议使用，建议逆向操作，创建新的索引，添加需要保留的数据内容

```
POST /sms_logs_index/sms_logs_type/_delete_by_query
{
 "query": {
 "range": {
 "relyTotal": {
 "gte": 2,
 "lte": 3
 }
 }
 }
}
##中间跟你的查询条件，查到什么，删什么
```

复合查询

bool查询

复合过滤器，可以将多个查询条件以一定的逻辑组合在一起

must:所有条件组合在一起，表示 and 的意思

must_not: 将must_not中的条件，全部都不匹配，表示not的意思

should:所有条件用should 组合在一起，表示or 的意思

```
POST /sms-logs-index/sms-logs-type/_search
{
```

```
"query": {
"bool": {
"should": [
{
"term": {
"province": {
"value": "晋城"
 }
 }
 },
 {
 "term": {
 "province": {
 "value": "北京"
 }
 }
 }
 ],
 "must_not": [
 {
 "term": {
 "operatorId": {
 "value": "2"
 }
 }
 }
 ],
 "must": [
 {
 "match": {
 "smsContent": "战士"
 }
 },
 {
 "match": {
 "smsContent": "的"
 }
 }
 ]
 }
```

```
  }
}
```

boosting 查询

boosting 查询可以帮助我们去影响查询后的score

positive:只有匹配上positive 查询的内容，才会被放到返回的结果集中

negative: 如果匹配上了positive 也匹配上了negative, 就可以 降低这样的文档score.

negative_boost:指定系数,必须小于1　　(0.5)

关于查询时，分数时如何计算的：

搜索的关键字再文档中出现的频次越高，分数越高

指定的文档内容越短，分数越高。

搜索时,指定的关键字也会被分词,这个被分词的内容.被分词库匹配的个数越多,分数就越高。

```
POST /sms-logs-index/sms-logs-type/_search
{
 "query": {
 "boosting": {
 "positive": {
 "match": {
 "smsContent": "孩童"
 }
 },
  "negative": {
  "match": {
  "smsContent": "希尔曼"
  }
  },
  "negative_boost": 0.5
  }
  }
}
```

filter 查询

query 查询：根据你的查询条件，去计算文档的匹配度得到一个分数，并根据分数排序，不会做缓存的。

filter 查询：根据查询条件去查询文档，不去计算分数，而且filter会对经常被过滤的数据进行缓存。

```
POST /sms-logs-index/sms-logs-type/_search
{
 "query": {
 "bool": {
 "filter": [ #json数组的bool复合查询
 {
 "term": {
 "corpName": "海尔智家公司"
 }
  },
  {
  "range":{
  "fee":{
  "lte":50
  }
  }
  }
  ]
  }
  }
}
```

高亮查询

高亮查询就是用户输入的关键字，以一定特殊样式展示给用户，让用户知道为什么这个结果被检索出来

高亮展示的数据，本身就是文档中的一个field,单独将field以highlight的形式返回给用户

ES提供了一个highlight 属性，他和query 同级别。

frament_size: 指定高亮数据展示多少个字符回来

pre_tags:指定前缀标签<front color="red">

post_tags:指定后缀标签

```
POST /sms-logs-index/sms-logs-type/_search
{
 "query": {
 "match": {
```

```
"smsContent": "团队"
}
},
"highlight": {
"fields": {
 "smsContent":{}
 },
 "pre_tags":"<font color='red'>",
 "post_tags":"</font>",
 "fragment_size":10
 }
}
```