1. 查询表达式（query DSL）：赋值给query

GET /\_search

{

"query": YOUR\_QUERY\_HERE

}

1. 需要constant\_score包裹，以便将**查询**转换成**过滤**。Filter必须放在bool中，此时是查询，无法缓存，将bool转成constant\_score，就变成过滤，可以缓存
2. terms 查询和 term 查询一样，但它允许你指定多值进行匹配。如果这个字段**包含了指定值中的任何一个值**，那么这个文档满足条件：

{ "terms": { "tag": [ "search", "full\_text", "nosql" ] }}

1. range:

“gte”:”**now-1h**”=🡺过去一小时

"lt" : "2014-01-01 00:00:00||+1M" 早于2014 年 2 月 1 日

1. missing: null，空数组，不存在的字段

"query" : {

"constant\_score" : { // **constant\_score性能等于bool**

"filter": {

"**missing**" : { "field" : "tags" }

}

}

}

1. match:

有**operator**参数，默认是or；可以设置成and，等同于**minimum\_should\_match**=100%或者must下包含若干个term

有**minimum\_should\_match**，控制匹配程度:可以是整数，或者百分比，一般使用百分比，因为不知道查询词的数量 "75%"

"query": {

"match": {

"title": {

"**query**": "quick brown dog",

"**minimum\_should\_match**": "75%",

“**operator**”:”and”

}

}

}

1. should：如果查询中只有should一个选项，则结果必须匹配should中的一项；否则，should可以忽略。

和match一样，should也可以使用minimum\_should\_match，至少符合should查询中几个项（可以是整数或者百分比）

"query": {

"bool": {

"should": [

{ "match": { "title": "brown" }},

{ "match": { "title": "fox" }},

{ "match": { "title": "dog" }}

],

"minimum\_should\_match": 2

}

}

1. **boost**: 提升或者降低\_score权重

{

"query": {

"bool": {

"must": {

"match": {

"content": {

"query": "full text search",

"operator": "and"

}

}

},

"should": [

{ "match": {

"content": {

"query": "Elasticsearch",

"**boost**": 3

}

}},

{ "match": {

"content": {

"query": "Lucene",

"**boost**": 2

}

}}

]

}

}

}

1. **多字段**查询

bool 查询采取 **more-matches-is-better** 匹配越多越好的方式，所以每条 match 语句的评分结果会被加在一起，从而为每个文档提供最终的分数 \_score 。能与两条语句同时匹配的文档比只与一条语句匹配的文档得分要高。

"query": {

"bool": {

"should": [

{ "match": { "title": "War and Peace" }}, //权重1/3

{ "match": { "author": "Leo Tolstoy" }}, //权重1/3

{ "bool": { //整个bool权重1/3

"should": [

{ "match": { "translator": "Constance Garnett" }},

{ "match": { "translator": "Louise Maude" }}

]

}}

]

}

}

1. **最佳匹配**：dis\_max 查询只会简单地使用 单个 最佳匹配语句的评分 \_score 作为整体评分

{

"query": {

"**dis\_max**": {

"**queries**": [

{ "match": { "title": "Brown fox" }},

{ "match": { "body": "Brown fox" }}

]

}

}

}

**最佳匹配调优**

{

"query": {

"**dis\_max**": {

"**queries**": [

{ "match": { "title": "Quick pets" }},

{ "match": { "body": "Quick pets" }}

],

"**tie\_breaker**": 0.3

}

}

}

tie\_breaker 参数提供了一种 dis\_max 和 bool 之间的折中选择，它的评分方式如下：

获得最佳匹配语句的评分 \_score 。

将其他匹配语句的评分结果与 tie\_breaker 相乘。

对以上评分求和并规范化。

1. **Multi-match**:

{

"**multi\_match**": { //查询类型是match

"**query**": "Quick brown fox", //查询值

"type": "best\_fields", //默认是**best\_fields(dis\_max)**，可换**most\_fields** (bool)和 **cross\_fields(混合多个字段后查询)**

"**fields**": [ "**\*\_**title", "body**^**1.5" ], //应用到那些字段。1.可使用通配符 2.可使用boost

"tie\_breaker": 0.3,

"minimum\_should\_match": "30%"

“operator”:”and”

}

}

1. **多数字段**

"mappings": {

"my\_type": {

"properties": {

"title": {

"type": "string",

"analyzer": "english",

**"fields": {**

**"std": {**

**"type": "string",**

**"analyzer": "standard"**

**}**

}

}

}

}

}

"query": {

"multi\_match": {

"query": "jumping rabbits",

"type": "most\_fields",

"fields": [ "title^10", "title.std" ] //提高相关度

}

}

"query": {

"multi\_match": {

"query": "peter smith",

"type": "**cross\_fields**", //会默认对查询字符进行分析（拆分成token，即使设置了operator为and）

"fields": [ "title^2", "description" ]

}

}

{

"query": {

"**match\_phrase**": {

"title": {

"query": "quick fox", //可以匹配quick brown fox

"**slop**": 1

}

}

}

}

**近似匹配**

1. **Slop**

"query": {

"**match\_phrase**": { //3个word必须相连

"title": "quick brown fox"

}

}

"query": {

"match\_phrase": {

"title": {

"query": "quick fox",

"**slop**": 1 //查询词移动几次能够匹配文档（可以匹配quick xxx fox），在范围内，quick 和 fox越接近，score越高

}

}

}

1. **Gap**

"properties": {

"names": {

"type": "string",

"**position\_increment\_gap**": 100 //为数组中元素认为添加slop

}

}

如果不设置**position\_increment\_gap**，以下查询会匹配数组

"query": {

"match\_phrase": {

"names": "**Abraham Lincoln**"

}

}

"names": [ "John **Abraham**", "**Lincoln** Smith"]