DSL 查询语法

```
GET /索引库名字/_search
{
    "query":{
        "查询类型":{
            "字段名字Field": "要查询的值"
        }
    }
}
//查询类型 match / match_all / multi_match
```

match查询：全文检索查询，对用户输入的内容分词，然后倒排索引库检索

```
GET /索引库名字/_search
{
  "query": {
    "match": {
      "FIELD": "TEXT"
    }
  }
}
//可以在建立索引库的时候，字段使用‘copy to’语法，可以聚合索引
```

multi_match 可以同时查询多个字段

```
GET /索引库名字/_search
{
  "query": {
    "multi_match": {
      "query": "",
      "fields": []
    }
  }
}
Eg:
GET /hotel/_search
{
  "query": {
    "multi_match": {
      "query": "地铁",
      "fields": ["name","business"]
    }
  }
}// 查询name，business中包含地铁的，返回结果--不建议非常多的字段查询
```

精确查询，一般是查找keyword，数值，日期，Boolean等类型字段，不会对搜索条件分词

*term*根据词条精确值查询

*range*根据值的范围查询

```

```

```
//term 查询
GET /索引库名字/_search
{
  "query": {
    "term": {
      "FIELD指定查询字段": {
        "value": "想要查询的值"
      }
    }
  }
}

//range 查询
GET /索引库名字/_search
{
  "query": {
    "range": {
      "FIELD1": {
        "gte": 10,
        "lte": 20
      }
    }
  }
}// 10<=FIELD1<=20(gte,lte), 10<FIELD1<20(gt,lt)
```

经纬度查询，使用场景：酒店查找，呼叫滴滴

geo_bounding_box: 查询geo_point值落在某个矩形范围的所有文档

```
GET /索引库名字/_search
{
  "query": {
    "geo_bounding_box":{
      "FIELD1":{
        "top_left":{
          "lat":31.1,
          "lon":121.5
        },
        "bottom_right":{
          "lat":30.9,
          "lon":121.7
        }
      }
    }
  }
}
```

geo_distance: 指定到中心点小于某个距离值的所有文档

```
GET /hotel/_search
{
  "query": {
    "geo_distance":{
      "distance":"15km",
      "FIELD1": "31.21,121.5"
    }
  }
}
```

复合查询：可以将其他简单的查询组合起来，实现更复杂的搜索逻辑

function score：算分函数查询，可以控制文档相关性算分，控制文档排名。比如百度竞价

$$TF(\text{词条频率}) = \frac{\text{词条出现次数}}{\text{文档中词条总数}}$$

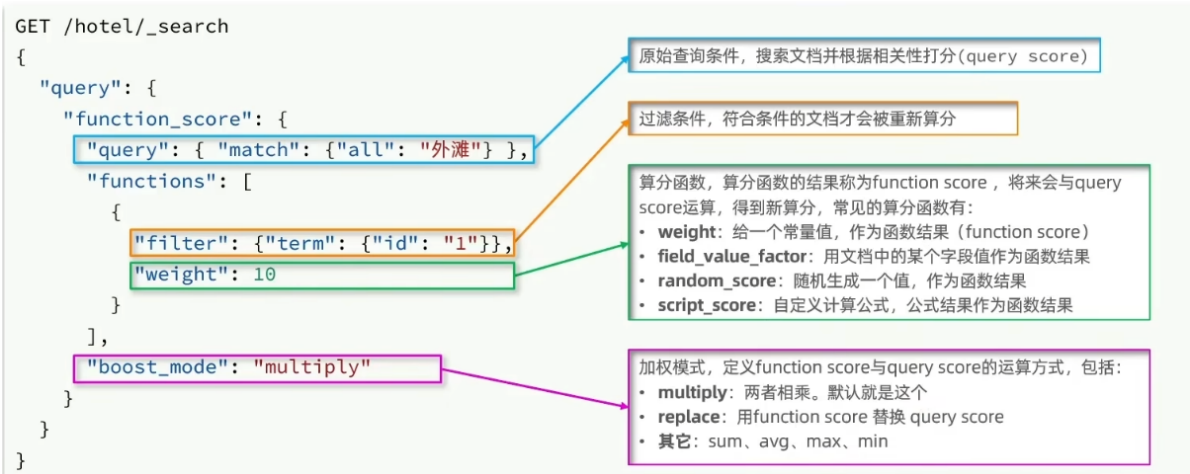
TF-IDF算法

$$IDF(\text{逆文档频率}) = Log(\frac{\text{文档总数}}{\text{包含词条的文档总数}})$$

$$score = \sum_i^n TF(\text{词条频率}) * IDF(\text{逆文档频率})$$

---

function score query，可以修改文档为相关性算分(query score)，根据新得到的算分排序



布尔查询， Boolean Query，一个或者多个子句的组合

must：必须匹配每个子查询，类似“与”

should：选择性匹配子查询，类似于“或”

must_not：必须不匹配，**不参与算分**，类似于“非”

filter：必须匹配，**不参与算分**

搜索结果处理

支持对搜索结果排序，默认是根据相关度算分（score）来排序，可以排序的字段类型有：keyword类型，数值类型，地理坐标类型，日期类型等

```
GET /hotel/_search
{
  "query": {
    "match_all": {}
  },
  "sort": [
    {
      "_geo_distance": {
        "location": {
          "lat": 31.014,
          "lon": 121.612
        },
        "order": "desc"//asc
      }
    }
  ]
}
```

分页（ES默认只返回top10的数据，查询更多就需要分页参数了）

通过修改from，size参数控制返回的分页结果

ES 是分布式的，会有深度分页的问题，对内存和CPU的消耗高，所以设定结果集查询的上限为10000

```
GET /hotel/_search
{
  "query": {
    "match_all": {},
  },
  "from": 50,//分页开始位置，默认为0
  "size": 200,//期望获取的文档总数
  "sort": [
    {
      "price": {
        "order": "asc"
      }
    }
  ]
}
```

关于深度分页，ES有两种解决方案，

search after，分页时需要排序，原理是从上一次排序值开始，查询下一页数据，官方推荐

scroll：原理将排序数据形成快照，保存在内存。官方不推荐

处理结果之 高亮， 把搜索结果中的关键收缩字突出显示

1.将搜索结果中的关键字用标签标记出来

2.在页面中给标签添加css样式

```
GET /hotel/_search
{
  "query": {
    "match": {
      "all": "如家"//搜索字段
    }
  },
  "highlight": {
    "fields": {
      "name": {//“name”这个字段中
        "pre_tags": "<em>",//标记高亮字段的前置标签
        "post_tags": "<em>",//标记高亮字段的后置标签
        "require_field_match": "false"
      }
    }
  }
}
```