Query DSL小结

一 概念以及基本查询

一共有两种clauses“

第一种是leaf query clauses，比如match，term或者range；

第二种是compound query clauses，组合查询方式，比如使用bool或者dis\_max查询，或者使用constant\_score。

1 相关性分数relevance scores。相关性分数越高，和文档的关联性越强。

2 查询内容query context

3 过滤内容filter context

查询内容和过滤内容的例子：

|  |  |
| --- | --- |
| 输入 | 输出 |
| GET hpc\_job/\_search  {  "query": {  "bool": {  "must": [  { "match": { "username": "yinjf" }},  { "match": { "queue": "normal" }}  ],  "filter": [  { "term": { "machine": "pss1" }},  { "range": { "nodes": { "gte": 8 }}}  ]  }  }  } | {  "took" : 624,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 10000,  "relation" : "gte"  },  "max\_score" : 11.772232,  "hits" : [  {  "\_index" : "hpc\_job",  "\_type" : "\_doc",  "\_id" : "20210114010103\_26293114",  "\_score" : 11.772232,  "\_source" : {  "creation\_time" : "2021-01-15 00:36:02",  "queue" : "normal",  "job\_id" : "26293114",  "username" : "yinjf",  "machine" : "pss1",  "nodes" : 8,  "processors" : 256,  "type" : "PARALLEL",  "submit\_time" : "2021-01-13 08:46:35",  "start\_time" : "2021-01-14 01:01:03",  "end\_time" : "2021-01-14 04:29:16",  "modelname" : "WRF",  "amount" : 3198208,  "state" : "3"  }  }, ... ... |

在上面这个例子中，使用query context我们用到了”query“关键词，里面写有bool逻辑，使用了must关键字，表示逻辑”与“的关系，即搜索的hits结果必须满足username为yinjf且queue为normal。过滤内容使用”filter“关键字，表示过滤项term为machine指定为pss1，range指定过滤的范围，即nodes数目不小于8（使用gte关键字表示不小于）。

Compound查询：

1 Bool查询（must/should/must\_not）

2 Boosting查询（negative/positive query）

|  |  |
| --- | --- |
| 输入 | 输出 |
| GET hpc\_job/\_search  {  "query": {  "boosting": {  "positive": {  "term": {  "username": "yinjf"  }  },  "negative": {  "term": {  "machine": "pss1"  }  },  "negative\_boost": 0.5  }  }  } | {  "took" : 99,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 10000,  "relation" : "gte"  },  "max\_score" : 8.451662,  "hits" : [  {  "\_index" : "hpc\_job",  "\_type" : "\_doc",  "\_id" : "20210209042657\_cma20n02.17956039.0",  "\_score" : 8.451662,  "\_source" : {  "creation\_time" : "2021-02-10 01:35:01",  "queue" : "normal",  "job\_id" : "cma20n02.17956039.0",  "username" : "yinjf",  "machine" : "neptune",  "nodes" : 0,  "processors" : 256,  "type" : "PARALLEL",  "submit\_time" : "2021-02-09 04:26:56",  "start\_time" : "2021-02-09 04:26:57",  "end\_time" : "2021-02-09 04:27:05",  "modelname" : "WRF",  "amount" : 2048,  "state" : "3"  }  }, ... ... |

在boosting查询中，positive关键字后面表示用户希望运行的命令。比如在以上这个例子中，我们期望运行username为yinjf的所有字段。Negative关键字后面表示用于降低匹配文档相关性的查询。在这个例子中，我们不太希望查找到machine为pss1的字段。Negative\_boost关键字可定义为0到1之间的一个值，用于降低匹配负查询的文档的相关性得分。

3 Constant\_score查询

|  |  |
| --- | --- |
| 输入 | 输出 |
| GET hpc\_job/\_search  {  "query": {  "constant\_score": {  "filter": {  "term": { "username": "yinjf" }  },  "boost": 1.2  }  }  } | {  "took" : 4,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 10000,  "relation" : "gte"  },  "max\_score" : 1.2,  "hits" : [  {  "\_index" : "hpc\_job",  "\_type" : "\_doc",  "\_id" : "20210114010103\_26293114",  "\_score" : 1.2,  "\_source" : {  "creation\_time" : "2021-01-15 00:36:02",  "queue" : "normal",  "job\_id" : "26293114",  "username" : "yinjf",  "machine" : "pss1",  "nodes" : 8,  "processors" : 256,  "type" : "PARALLEL",  "submit\_time" : "2021-01-13 08:46:35",  "start\_time" : "2021-01-14 01:01:03",  "end\_time" : "2021-01-14 04:29:16",  "modelname" : "WRF",  "amount" : 3198208,  "state" : "3"  }  }, ... ... |

Constant\_score查询表示包装筛选器查询并返回每个匹配的文档，其相关性得分等于boost参数值。在以上这个例子中，我们查找machine为pss1且相关性分数固定为1.2的文档并返回，可知满足条件的有四个。

4 Dis\_max查询

返回匹配一个或多个包装查询的文档，称为查询子句或子句。

如果返回的文档匹配多个查询子句，则dis\_max查询从任何匹配子句中为该文档分配最高的相关性得分，并为任何其他匹配子查询加上一个打破平局的增量。

可以使用dis\_max在具有不同boost因子映射的字段中搜索一个术语。

|  |  |
| --- | --- |
| 输入 | 输出 |
| GET hpc\_job/\_search  {  "query": {  "dis\_max": {  "queries": [  { "term": { "queue": "normal" } },  { "term": { "machine": "pss1" } }  ],  "tie\_breaker": 0.7  }  }  } | {  "took" : 88,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 10000,  "relation" : "gte"  },  "max\_score" : 4.8454175,  "hits" : [  {  "\_index" : "hpc\_job",  "\_type" : "\_doc",  "\_id" : "20210114035031\_26292140",  "\_score" : 4.8454175,  "\_source" : {  "creation\_time" : "2021-01-15 00:36:02",  "queue" : "normal",  "job\_id" : "26292140",  "username" : "wanghong",  "machine" : "pss1",  "nodes" : 32,  "processors" : 1024,  "type" : "PARALLEL",  "submit\_time" : "2021-01-13 08:03:14",  "start\_time" : "2021-01-14 03:50:31",  "end\_time" : "2021-01-14 05:53:57",  "modelname" : "GRAPES",  "amount" : 7583744,  "state" : "3"  }  }, ... ... |

Dis\_max下面的query中的term，返回的文档必须要满足至少一个。在该例子中，queue等于normal或者machine等于pss1必须要满足至少一个，tie\_breaker表示0打1之间的浮点数，表示用于增加匹配多个查询子句的文档的相关性得分。默认为零，在例子中我们指定为0.7.

5 Function\_score查询

二 全文本查询（Full text queries）

全文查询能够搜索分析过的文本字段

1 intervals query一个全文查询，允许细粒度控制匹配术语的顺序和接近度。

根据匹配术语的顺序和接近度返回文档。

间隔查询使用匹配规则，这些规则由一小组定义构造而成。然后将这些规则应用于来自指定字段的术语。

定义产生跨越文本主体中的术语的最小间隔序列。这些间隔可以通过父源进一步组合和过滤。

2 match query用于执行全文查询的标准查询，包括模糊匹配和短语或邻近查询。

3 match\_bool\_prefix query创建一个bool查询，将每个术语作为一个术语查询进行匹配，最后一个术语作为前缀查询进行匹配

4 match\_phrase query类似于匹配查询，但用于匹配准确的短语或单词邻近匹配。

5 match\_phrase\_prefix query类似于match\_phrase查询，但对最后一个单词进行通配符搜索。

6 multi\_match query匹配查询的多字段版本。

7 combined\_fields query匹配多个字段，就好像它们已经被索引到一个组合字段中。

8 query\_string query支持紧凑的Lucene查询字符串语法，允许您在单个查询字符串中指定AND|OR|NOT条件和多字段搜索。仅供专家用户使用。

9 simple\_query\_string query适合直接向用户公开的更简单、更健壮的query\_string语法版本。

三 Geographic queries

1 geo\_bounding\_box query

2 geo\_distance query

3 geo\_polygon query

4 geo\_shape query

四 shape query

查询包含使用形状类型索引的字段的文档。

需要形状映射。

该查询支持两种定义目标形状的方法，一种是提供完整的形状定义，另一种是引用在另一个索引中预先索引的形状的名称或id。下面用示例定义了这两种格式。

五 joining query

1 嵌套查询

文档可能包含嵌套类型的字段。这些字段用于索引对象数组，其中每个对象都可以作为一个独立的文档进行查询(使用嵌套查询)。

2 Has\_child和has\_parent查询

连接字段关系可以存在于单个索引中的文档之间。has\_child查询返回其子文档与指定查询匹配的父文档，而has\_parent查询返回其父文档与指定查询匹配的子文档。

六 span queries

跨度查询是低级的位置查询，它提供对指定术语的顺序和邻近性的专家控制。它们通常用于实现对法律文件或专利的非常具体的查询。

只允许在外跨度查询上设置boost。复合跨度查询，比如span\_near，只使用内部跨度查询的匹配跨度的列表来找到它们自己的跨度，然后使用这些跨度生成分数。分数永远不会在内部跨度查询中计算，这也是为什么不允许提升的原因:它们只影响分数的计算方式，而不是跨度。

七 specialized queries

这一组包含不适合其他组的查询:

distance\_feature query

一个基于起源和文档的date、date\_nanos和geo\_point字段之间动态计算的距离来计算得分的查询。它能够有效地跳过非竞争性的命中。

more\_like\_this query

该查询查找与指定文本、文档或文档集合相似的文档。

percolate query

该查询查找存储为与指定文档匹配的文档的查询。

rank\_feature query

一种基于数值特征值计算分数的查询，能够有效地跳过非竞争性命中。

script query

该查询允许脚本充当过滤器。请参见function\_score查询。

script\_score query

允许用脚本修改子查询得分的查询。

wrapper query

一个以json或yaml字符串形式接受其他查询的查询。

pinned query

将选定的文档提升到与给定查询匹配的其他文档之上的查询。

八 term-level queries

可以使用术语级查询根据结构化数据中的精确值查找文档。结构化数据的例子包括日期范围、IP地址、价格或产品id。

与全文查询不同，术语级查询不分析搜索词。相反，术语级查询匹配存储在字段中的准确的术语。

Types of term-level queries

exists query

返回包含字段的任何索引值的文档。

fuzzy query

返回包含与搜索词相似的词的文档。Elasticsearch使用Levenshtein编辑距离度量相似性或模糊性。

ids query

根据文档id返回文档。

prefix query

返回在提供的字段中包含特定前缀的文档。

range query

返回包含给定范围内的术语的文档。

regexp query

返回包含匹配正则表达式的术语的文档。

term query

返回在提供的字段中包含精确术语的文档。

terms query

返回在提供的字段中包含一个或多个准确术语的文档。

terms\_set query

返回在提供的字段中包含最少数量的准确术语的文档。您可以使用一个字段或脚本定义匹配术语的最小数量。

type query

返回指定类型的文档。

wildcard query

返回包含匹配通配符模式的术语的文档。