# 优质IT资源微信x923713



# 《手写OS操作系统》小班二期招生,全程直播授课,大牛带你掌握硬核技术!



慕课网首页

免费课

实战课

体系课

慕课教程

专栏 手记

企业服务

QĦ

.

我拍

莩

从所有教程的词条中查询・・・

首页 > 慕课教程 > Go工程师体系课全新版 > 8. query dsl查询



1. go最常用的设计模式 – 函数 选项

2. 单例模式和懒加载

3. 测试金字塔

# 第23周 protoc插件开发、cobra命令行

1. protoc调试源码

2. protoc自定义qin插件

#### 第24周 log日志包设计

日志源码

# 第25周 ast代码生成工具开发

错误码

#### 第26周 三层代码结构

通用app项目启动

**bobby**・更新于 2022–11–16

◆ 上一节 7. 批量插入和批... 9. mapping 下一节 →

# 1. 一个完整的查询语句应该如何写

#### 官方文档

## 2. 全文查询 - 分词

#### 1. match查询(匹配查询)

#### 文档

match: 模糊匹配,需要指定字段名,但是输入会进行分词,比如"hello world"会进行拆分为hello和world,然后匹配,如果字段中包含hello或者world,或者都包含的结果都会被查询出来,也就是说match是一个部分匹配的模糊查询。查询条件相对来说比较宽松。



 $\Box$ 

0



### 2. match\_phrase查询 短语查询

#### 官方文档

match\_phase: 会对输入做分词,但是需要结果中也包含所有的分词,而且顺序要求一样。以"hello world"为例,要求结果中必须包含hello和world,而且还要求他们是连着的,顺序也是固定的,hello that word不满足,world hello也不满足条件。

🧪 意见反馈

♡ 收藏教程

□ 标记书签

# 优质IT资源微信x923深13

#### 3. multi\_match查询

#### 官方文档

multi\_match 查询提供了一个简便的方法用来对多个字段执行相同的查询,即对指定的多个字段进行 match查询

```
<> 代码块
    POST resume/_doc/12
2
      "title": "后端工程师",
3
      "desc": "多年go语言开发经验, 熟悉go的基本语法, 熟悉常用的go语言库",
4
5
      "want_learn":"python语言"
6
7
8
    POST resume/_doc/13
9
      "title": "go工程师",
10
     "desc": "多年开发经验",
11
      "want_learn":"java语言"
12
13
14
15
    POST resume/_doc/14
16
17
      "title": "后端工程师",
18
    "desc": "多年开发经验",
19
20
      "want_learn":"rust语言"
21
22
23
24
   GET account/_search
25
     "query": {
26
        "multi_match": {
27
         "query": "go",
28
         "fields": ["title", "desc"]
30
      }
31
32
```

# ### 4. query\_string查询 [官方文档]

(https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html)
`query\_string`: 和match类似,但是match需要指定字段名,query\_string是在所有字段中搜索,范围更 广泛。```shell GET user/\_search { "query":{ "query\_string": { "default\_field": "address", "query":
"Madison AND street" } } } ```

### 3. match all 查询

```
1   GET user/_search
2   {
3     "query": {
4          "match_all": {}
5      }
6   }
```

⊡

?

 $\Box$ 

0

# 优质IT资源微信x923深13

#### 官方文档

## 1. term查询

term: 这种查询和match在有些时候是等价的,比如我们查询单个的词hello,那么会和match查询结果一样,但是如果查询"hello world",结果就相差很大,因为这个输入不会进行分词,就是说查询的时候,是查询字段分词结果中是否有"hello world"的字样,而不是查询字段中包含"hello world"的字样,elasticsearch会对字段内容进行分词,"hello world"会被分成hello和world,不存在"hello world",因此这里的查询结果会为空。这也是term查询和match的区别。

### 2. range查询 - 范围查询

```
<> 代码块
1
   GET user/_search
    "query":{
3
       "range": {
 4
         "age": {
           "gte": 20,
 6
           "lte": 30
7
8
        }
9
      }
10
    }
   }
11
```

# 3. exists查询

## 4. fuzzy模糊查询

### 编辑距离

⊡

?

0

# 优质IT资源微信x923深13

```
"fuzziness": 1
         }
 8
       }
9
10
    }
11 }
<> 代码块
   GET /_search
1
   {
      "query": {
3
       "fuzzy": {
4
         "user.id": {
5
6
          "value": "ki"
7
        }
      }
8
9
    }
10 }
```

# 5. 复合查询

#### 官方文档

Elasticsearch bool查询对应Lucene BooleanQuery, 格式如下

```
<> 代码块
   {
1
        "query":{
 2
          "bool":{
 3
               "must":[
 5
               ],
               "should":[
 6
 7
               ],
               "must_not":[
               ],
9
               "filter":[
10
11
12
           }
      }
13
14 }
```

```
1 must: 必须匹配,查询上下文,加分
2 should: 应该匹配,查询上下文,加分
3 must_not: 必须不匹配,过滤上下文,过滤
4 filter: 必须匹配,过滤上下文,过滤
```

bool查询采用了一种匹配越多越好的方法,因此每个匹配的must或should子句的分数将被加在一起,以 提供每个文档的最终得分

⊡

?

0

2022/12/9 10:21

# 优质IT资源微信\*923713

```
},
10
              {
11
               "range": {
12
                 "age": {
13
                   "gte": 20,
14
                   "lte": 30
15
16
               }
17
             }
18
19
           ],
20
           "must_not": [
             {
21
               "term": {
22
                 "gender": "m"
23
               }
24
             }
25
26
           ],
           "should": [
27
             {
28
               "match": {
29
                 "firstname": "Decker"
30
               }
31
             }
32
33
           ],
           "filter": [
34
             {
35
               "range": {
36
37
                 "age": {
38
                   "gte": 25,
                   "lte": 30
39
                                                                                          40
                 }
               }
41
             }
42
                                                                                          ?
           ]
43
44
                                                                                          45
       }
46
     }
                                                                                          0
                  7. 批量插入和批量查询 ◆ 上一节
                                             下一节 ▶ 9. mapping
                                    ▶ 我要提出意见反馈
```

企业服务 网站地图 网站首页 关于我们 联系我们 讲师招募 帮助中心 意见反馈 代码托管

Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11 京公网安备11010802030151号