**3.查询**

**我们从4块来讲查询：**

* **基本查询**
* **\_source过滤**
* **结果过滤**
* **高级查询**
* **排序**

**3.1.基本查询：**

**基本语法**

**GET /索引库名/\_search**

**{**

**"query":{**

**"查询类型":{**

**"查询条件":"查询条件值"**

**}**

**}**

**}**

**这里的query代表一个查询对象，里面可以有不同的查询属性**

* **查询类型：** 
  + **例如：match\_all， match，term ， range 等等**
* **查询条件：查询条件会根据类型的不同，写法也有差异，后面详细讲解**

**3.1.1 查询所有（match\_all)**

**示例：**

**GET /testindex/\_search**

**{**

**"query":{**

**"match\_all": {}**

**}**

**}**

* **query：代表查询对象**
* **match\_all：代表查询所有**

**结果：**

**{**

**"took": 2,**

**"timed\_out": false,**

**"\_shards": {**

**"total": 5,**

**"successful": 5,**

**"failed": 0**

**},**

**"hits": {**

**"total": 1,**

**"max\_score": 1,**

**"hits": [**

**{**

**"\_index": "testindex",**

**"\_type": "goods",**

**"\_id": "AWsS5Neq-k3yg4WVTNnG",**

**"\_score": 1,**

**"\_source": {**

**"title": "小米手机",**

**"images": "1,jpg",**

**"price": 111**

**}**

**}**

**]**

**}**

**}**

* **took：查询花费时间，单位是毫秒**
* **time\_out：是否超时**
* **\_shards：分片信息**
* **hits：搜索结果总览对象** 
  + **total：搜索到的总条数**
  + **max\_score：所有结果中文档得分的最高分**
  + **hits：搜索结果的文档对象数组，每个元素是一条搜索到的文档信息** 
    - **\_index：索引库**
    - **\_type：文档类型**
    - **\_id：文档id**
    - **\_score：文档得分**
    - **\_source：文档的源数据**

**3.1.2 匹配查询（match）**

* **or关系**

**match类型查询，会把查询条件进行分词，然后进行查询,多个词条之间是or的关系**

**GET /testindex/\_search**

**{**

**"query":{**

**"match":{**

**"title":"小米手机"**

**}**

**}**

**}**

**结果：**

**{**

**"took": 26,**

**"timed\_out": false,**

**"\_shards": {**

**"total": 5,**

**"successful": 5,**

**"failed": 0**

**},**

**"hits": {**

**"total": 2,**

**"max\_score": 0.51623213,**

**"hits": [**

**{**

**"\_index": "testindex",**

**"\_type": "goods",**

**"\_id": "AWsS5Neq-k3yg4WVTNnG",**

**"\_score": 0.51623213,**

**"\_source": {**

**"title": "小米手机",**

**"images": "1,jpg",**

**"price": 111**

**}**

**},**

**{**

**"\_index": "testindex",**

**"\_type": "goods",**

**"\_id": "3",**

**"\_score": 0.25811607,**

**"\_source": {**

**"title": "小米电视",**

**"images": "4.jp",**

**"price": 444**

**}**

**}**

**]**

**}**

**}**

**在上面的案例中，不仅会查询到小米手机，而且与小米相关的都会查询到，多个词之间是or的关系。（小米手机分为小米和手机两个词，由于是or关系，所以只要有小米或者手机两个关键字中的一个就会被查询到）**

* **and关系**

**某些情况下，我们需要更精确查找，我们希望这个关系变成and，可以这样做：**

**GET /testindex/\_search**

**{**

**"query":{**

**"match": {**

**"title": {**

**"query": "小米手机",**

**"operator": "and"**

**}**

**}**

**}**

**}**

**结果：**

**{**

**"took": 26,**

**"timed\_out": false,**

**"\_shards": {**

**"total": 5,**

**"successful": 5,**

**"failed": 0**

**},**

**"hits": {**

**"total": 1,**

**"max\_score": 0.51623213,**

**"hits": [**

**{**

**"\_index": "testindex",**

**"\_type": "goods",**

**"\_id": "AWsS5Neq-k3yg4WVTNnG",**

**"\_score": 0.51623213,**

**"\_source": {**

**"title": "小米手机",**

**"images": "1,jpg",**

**"price": 111**

**}**

**}**

**]**

**}**

**}**

**本例中，只有同时包含小米和手机的词条才会被搜索到。**

* **or和and之间？**

**在 or 与 and 间二选一有点过于非黑即白。 如果用户给定的条件分词后有 5 个查询词项，想查找只包含其中 4 个词的文档，该如何处理？将 operator 操作符参数设置成 and 只会将此文档排除。**

**有时候这正是我们期望的，但在全文搜索的大多数应用场景下，我们既想包含那些可能相关的文档，同时又排除那些不太相关的。换句话说，我们想要处于中间某种结果。**

**match 查询支持 minimum\_should\_match 最小匹配参数， 这让我们可以指定必须匹配的词项数用来表示一个文档是否相关。我们可以将其设置为某个具体数字，更常用的做法是将其设置为一个百分数，因为我们无法控制用户搜索时输入的单词数量：**

**GET /testindex/\_search**

**{**

**"query":{**

**"match":{**

**"title":{**

**"query":"小米无敌手机",**

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

**}**

**}**

**}**

**}**

**本例中，搜索语句可以分为3个词，如果使用and关系，需要同时满足3个词才会被搜索到。这里我们采用最小品牌数：75%，那么也就是说只要匹配到总词条数量的75%即可，这里3\*75% 约等于2。所以只要包含2个词条就算满足条件了。**

**3.1.3 多字段查询（multi\_match）**

**multi\_match与match类似，不同的是它可以在多个字段中查询**

**GET /testindex/\_search**

**{**

**"query":{**

**"multi\_match": {**

**"query": "小米",**

**"fields": [ "title", "image" ]**

**}**

**}**

**}**

**会在title和image两个字段中取查询**

**3.1.4 词条匹配(term)**

**term 查询被用于精确值 匹配，这些精确值可能是数字、时间、布尔或者那些未分词的字符串**

**GET /testindex/\_search**

**{**

**"query":{**

**"term":{**

**"price":111**

**}**

**}**

**}**

**结果：**

**{**

**"took": 15,**

**"timed\_out": false,**

**"\_shards": {**

**"total": 5,**

**"successful": 5,**

**"failed": 0**

**},**

**"hits": {**

**"total": 1,**

**"max\_score": 1,**

**"hits": [**

**{**

**"\_index": "testindex",**

**"\_type": "goods",**

**"\_id": "AWsS5Neq-k3yg4WVTNnG",**

**"\_score": 1,**

**"\_source": {**

**"title": "小米手机",**

**"images": "1,jpg",**

**"price": 111**

**}**

**}**

**]**

**}**

**}**

**3.1.5 多词条精确匹配(terms)**

**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：**

**GET /testindex/\_search**

**{**

**"query":{**

**"terms":{**

**"price":[111,222]**

**}**

**}**

**}**

**3.2.结果过滤**

**默认情况下，elasticsearch在搜索的结果中，会把文档中保存在\_source的所有字段都返回。**

**如果我们只想获取其中的部分字段，我们可以添加\_source的过滤**

**3.2.1.直接指定字段**

**示例：**

**GET /testindex/\_search**

**{**

**"\_source": ["title","price"],**

**"query": {**

**"term": {**

**"price": 111**

**}**

**}**

**}**

**返回的结果：**

**{**

**"took": 28,**

**"timed\_out": false,**

**"\_shards": {**

**"total": 5,**

**"successful": 5,**

**"failed": 0**

**},**

**"hits": {**

**"total": 1,**

**"max\_score": 1,**

**"hits": [**

**{**

**"\_index": "testindex",**

**"\_type": "goods",**

**"\_id": "AWsS5Neq-k3yg4WVTNnG",**

**"\_score": 1,**

**"\_source": {**

**"price": 111,**

**"title": "小米手机"**

**}**

**}**

**]**

**}**

**}**

**这样\_source字段中只有title和price两个字段了**

**3.2.2.指定includes和excludes**

**我们也可以通过：**

* **includes：来指定想要显示的字段**
* **excludes：来指定不想要显示的字段**

**二者都是可选的。**

**示例：**

**GET /testindex/\_search**

**{**

**"\_source": {**

**"includes":["title","price"]**

**},**

**"query": {**

**"term": {**

**"price": 111**

**}**

**}**

**}**

**与下面的结果将是一样的：**

**GET /testindex/\_search**

**{**

**"\_source": {**

**"excludes": ["images"]**

**},**

**"query": {**

**"term": {**

**"price": 2699**

**}**

**}**

**}**

**3.3 高级查询**

**3.3.1 布尔组合（bool)**

**bool把各种其它查询通过must（与）、must\_not（非）、should（或）的方式进行组合**

**GET /testindex/\_search**

**{**

**"query":{**

**"bool":{**

**"must": { "match": { "title": "小米" }},**

**"must\_not": { "match": { "title": "电视" }},**

**"should": { "match": { "title": "手机" }}**

**}**

**}**

**}**

**结果：**

**{**

**"took": 18,**

**"timed\_out": false,**

**"\_shards": {**

**"total": 5,**

**"successful": 5,**

**"failed": 0**

**},**

**"hits": {**

**"total": 1,**

**"max\_score": 0.51623213,**

**"hits": [**

**{**

**"\_index": "testindex",**

**"\_type": "goods",**

**"\_id": "AWsS5Neq-k3yg4WVTNnG",**

**"\_score": 0.51623213,**

**"\_source": {**

**"title": "小米手机",**

**"images": "1,jpg",**

**"price": 111**

**}**

**}**

**]**

**}**

**}**

**range查询允许以下字符：**

| **操作符** | **说明** |
| --- | --- |
| **gt** | **大于** |
| **gte** | **大于等于** |
| **lt** | **小于** |
| **lte** | **小于等于** |

**3.3.3 模糊查询(fuzzy)**

**我们新增一个商品：**

**POST /testindex/goods/4**

**{**

**"title":"apple手机",**

**"images":"apple.jpg",**

**"price":6899.00**

**}**

**fuzzy 查询是 term 查询的模糊等价。它允许用户搜索词条与实际词条的拼写出现偏差，但是偏差的编辑距离不得超过2：**

**GET /testindex/\_search**

**{**

**"query": {**

**"fuzzy": {**

**"title": "appla"**

**}**

**}**

**}**

**上面的查询，也能查询到apple手机**

**我们可以通过fuzziness来指定允许的编辑距离：**

**GET /testindex/\_search**

**{**

**"query": {**

**"fuzzy": {**

**"title": {**

**"value":"appla",**

**"fuzziness":1**

**}**

**}**

**}**

**}**

**3.4 过滤(filter)**

**条件查询中进行过滤**

**所有的查询都会影响到文档的评分及排名。如果我们需要在查询结果中进行过滤，并且不希望过滤条件影响评分，那么就不要把过滤条件作为查询条件来用。而是使用filter方式：**

**GET /testindex/\_search**

**{**

**"query":{**

**"bool":{**

**"must":{ "match": { "title": "小米手机" }},**

**"filter":{**

**"range":{"price":{"gt":2000.00,"lt":3800.00}}**

**}**

**}**

**}**

**}**

**注意：filter中还可以再次进行bool组合条件过滤。**

**无查询条件，直接过滤**

**如果一次查询只有过滤，没有查询条件，不希望进行评分，我们可以使用constant\_score取代只有 filter 语句的 bool 查询。在性能上是完全相同的，但对于提高查询简洁性和清晰度有很大帮助。**

**GET /heima/\_search**

**{**

**"query":{**

**"constant\_score": {**

**"filter": {**

**"range":{"price":{"gt":2000.00,"lt":3000.00}}**

**}**

**}**

**}**

**3.5 排序**

**3.4.1 单字段排序**

**sort 可以让我们按照不同的字段进行排序，并且通过order指定排序的方式**

**GET /testindex/\_search**

**{**

**"query": {**

**"match": {**

**"title": "小米手机"**

**}**

**},**

**"sort": [**

**{**

**"price": {**

**"order": "desc"**

**}**

**}**

**]**

**}**

**3.4.2 多字段排序**

**假定我们想要结合使用 price和 \_score（得分） 进行查询，并且匹配的结果首先按照价格排序，然后按照相关性得分排序：**

**GET /goods/\_search**

**{**

**"query":{**

**"bool":{**

**"must":{ "match": { "title": "小米手机" }},**

**"filter":{**

**"range":{"price":{"gt":200000,"lt":300000}}**

**}**

**}**

**},**

**"sort": [**

**{ "price": { "order": "desc" }},**

**{ "\_score": { "order": "desc" }}**

**]**

**}**