ES进阶篇

# Elasticsearch结构化搜索\_在案例中实战使用term filter来搜索数据

根据用户ID、是否隐藏、帖子ID、发帖日期来搜索帖子

1.1插入一些测试帖子数据

|  |
| --- |
| POST /forum/article/\_bulk  { "index": { "\_id": 1 }}  { "articleID" : "XHDK-A-1293-#fJ3", "userID" : 1, "hidden": false, "postDate": "2017-01-01" }  { "index": { "\_id": 2 }}  { "articleID" : "KDKE-B-9947-#kL5", "userID" : 1, "hidden": false, "postDate": "2017-01-02" }  { "index": { "\_id": 3 }}  { "articleID" : "JODL-X-1937-#pV7", "userID" : 2, "hidden": false, "postDate": "2017-01-01" }  { "index": { "\_id": 4 }}  { "articleID" : "QQPX-R-3956-#aD8", "userID" : 2, "hidden": true, "postDate": "2017-01-02" } |

初步来说，就先搞4个字段，因为整个es是支持json document格式的，所以说扩展性和灵活性非常之好。如果后续随着业务需求的增加，要在document中增加更多的field，那么我们可以很方便的随时添加field。但是如果是在关系型数据库中，比如mysql，我们建立了一个表，现在要给表中新增一些column，那就很坑爹了，必须用复杂的修改表结构的语法去执行。而且可能对系统代码还有一定的影响。

|  |
| --- |
| GET /forum/\_mapping/article  {  "forum": {  "mappings": {  "article": {  "properties": {  "articleID": {  "type": "text",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "hidden": {  "type": "boolean"  },  "postDate": {  "type": "date"  },  "userID": {  "type": "long"  }  }  }  }  }  } |

现在es 5.2版本，type=text，默认会设置两个field，一个是field本身，比如articleID，就是分词的；还有一个的话，就是field.keyword，articleID.keyword，默认不分词，会最多保留256个字符

1.2根据用户ID搜索帖子

|  |
| --- |
| GET /forum/article/\_search  {  "query" : {  "constant\_score" : {  "filter" : {  "term" : {  "userID" : 1  }  }  }  }  } |

term filter/query：对搜索文本不分词，直接拿去倒排索引中匹配，你输入的是什么，就去匹配什么

比如说，如果对搜索文本进行分词的话，“helle world” --> “hello”和“world”，两个词分别去倒排索引中匹配

term，“hello world” --> “hello world”，直接去倒排索引中匹配“hello world”

1.3根据发帖日期搜索帖子

|  |
| --- |
| GET /forum/article/\_search  {  "query" : {  "constant\_score" : {  "filter" : {  "term" : {  "postDate": "2017-01-01"  }  }  }  }  } |

1.4根据帖子ID搜索帖子

|  |
| --- |
| GET /forum/article/\_search  {  "query" : {  "constant\_score" : {  "filter" : {  "term" : {  "articleID" : "XHDK-A-1293-#fJ3"  }  }  }  }  }  {  "took": 1,  "timed\_out": false,  "\_shards": {  "total": 5,  "successful": 5,  "failed": 0  },  "hits": {  "total": 0,  "max\_score": null,  "hits": []  }  }  GET /forum/article/\_search  {  "query" : {  "constant\_score" : {  "filter" : {  "term" : {  "articleID.keyword" : "XHDK-A-1293-#fJ3"  }  }  }  }  }  {  "took": 2,  "timed\_out": false,  "\_shards": {  "total": 5,  "successful": 5,  "failed": 0  },  "hits": {  "total": 1,  "max\_score": 1,  "hits": [  {  "\_index": "forum",  "\_type": "article",  "\_id": "1",  "\_score": 1,  "\_source": {  "articleID": "XHDK-A-1293-#fJ3",  "userID": 1,  "hidden": false,  "postDate": "2017-01-01"  }  }  ]  }  } |

articleID.keyword，是es最新版本内置建立的field，就是不分词的。所以一个articleID过来的时候，会建立两次索引，一次是自己本身，是要分词的，分词后放入倒排索引；另外一次是基于articleID.keyword，不分词，保留256个字符最多，直接一个字符串放入倒排索引中。

所以term filter，对text过滤，可以考虑使用内置的field.keyword来进行匹配。但是有个问题，默认就保留256个字符。所以尽可能还是自己去手动建立索引，指定not\_analyzed吧。在最新版本的es中，不需要指定not\_analyzed也可以，将type=keyword即可。

1.5查看分词

|  |
| --- |
| GET /forum/\_analyze  {  "field": "articleID",  "text": "XHDK-A-1293-#fJ3"  } |

默认是analyzed的text类型的field，建立倒排索引的时候，就会对所有的articleID分词，分词以后，原本的articleID就没有了，只有分词后的各个word存在于倒排索引中。

term，是不对搜索文本分词的，XHDK-A-1293-#fJ3 --> XHDK-A-1293-#fJ3；但是articleI建立索引的时候，XHDK-A-1293-#fJ3 --> xhdk，a，1293，fj3

1.6重建索引

|  |
| --- |
| DELETE /forum  PUT /forum  POST /forum/article/\_bulk  { "index": { "\_id": 1 }}  { "articleID" : "XHDK-A-1293-#fJ3", "userID" : 1, "hidden": false, "postDate": "2017-01-01" }  { "index": { "\_id": 2 }}  { "articleID" : "KDKE-B-9947-#kL5", "userID" : 1, "hidden": false, "postDate": "2017-01-02" }  { "index": { "\_id": 3 }}  { "articleID" : "JODL-X-1937-#pV7", "userID" : 2, "hidden": false, "postDate": "2017-01-01" }  { "index": { "\_id": 4 }}  { "articleID" : "QQPX-R-3956-#aD8", "userID" : 2, "hidden": true, "postDate": "2017-01-02" } |

1.7重新根据帖子ID和发帖日期进行搜索

|  |
| --- |
| GET /forum/article/\_search  {  "query" : {  "constant\_score" : {  "filter" : {  "term" : {  "articleID" : "XHDK-A-1293-#fJ3"  }  }  }  }  } |

1.8总结

term filter：根据exact value进行搜索，数字、boolean、date天然支持

text需要建索引时指定为not\_analyzed，才能用term query

相当于SQL中的单个where条件

select \* from forum.article where articleID='XHDK-A-1293-#fJ3';

# Elasticsearch结构化搜索\_filter执行原理深度剖析（bitset机制与caching机制）

2.1在倒排索引中查找搜索串，获取document list