

# Java客户端下 的增删改查

<https://blog.csdn.net/zhangxin09/article/details/51405016> 挺详细的，可以作为查询资料

# 都有什么查询

散仙在上篇[文章](http://qindongliang.iteye.com/blog/2183142)中，介绍了关于ElasticSearch基本的增删改查的基本粒子，本篇呢，我们来学下稍微高级一点的知识：   
（1）如何在ElasticSearch中批量提交索引 ？   
（2）如何使用高级查询（包括，检索，排序，过滤，分页） ？   
（3）如何组合多个查询 ？   
（4）如何使用翻页深度查询 ？   
（5）如何使用基本的聚合查询 ？

# 一般查询方式

/\*\*

\* 使用QueryBuilder

\* termQuery("key", obj) 完全匹配

\* termsQuery("key", obj1, obj2..) 一次匹配多个值

\* matchQuery("key", Obj) 单个匹配, field不支持通配符, 前缀具高级特性

\* multiMatchQuery("text", "field1", "field2"..); 匹配多个字段, field有通配符忒行

\* matchAllQuery(); 匹配所有文件

\*/

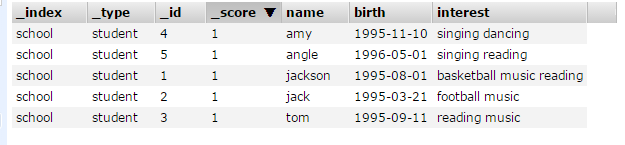
## matchQuery和termQuery的区别

termQuery  
不带分析器，比如说你搜索“中国”，没有分析器你就搜索不到，而  
matchQuery  
就带了分析器，你搜索“中国”的时候他会自动使用自带的中文分析器帮你去检索，那么你就能搜索到关于“中国”的信息。  
2.term是字段的检索，检索时会按照你输入的内容按照完全匹配的模式检索，而match是全文检索，会模糊按照匹配相关度给你找出结果按分值排列。某种意义上来说，term相当于“match\_phrase”。

matchQuery是带分词功能的全文搜索  
而termQuery是不带分词功能的精准匹配  
什么意思呢 matchQuery 会分析词语 比如hello world quick 会拆分这个［hello,wolrd,quick］去搜索  
term只会去精准匹配 不带分词

# 搜索案例

**elasticsearch中存储的全部文档**

****

**1.matchAllQuery()**

**matchAllQuery()方法用来匹配全部文档**

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. **public** **class** QueryTest {
2. **public** **static** **void** main(String[] args) {
3. //创建对象，设置集群名称和IP地址
4. ElasticsearchUtils es = **new** ElasticsearchUtils("im\_shan",
5. "localhost");
6. String indexName = "school";//索引名称
7. String typeName = "student";//类型名称
8. //搜索全部文档
9. QueryBuilder queryBuilder = QueryBuilders.matchAllQuery();
10. SearchResponse searchResponse = es.searcher(indexName, typeName,
11. queryBuilder);
12. SearchHits hits = searchResponse.getHits();
13. SearchHit[] searchHits = hits.getHits();
14. **int** i = 0;
15. **for** (SearchHit searchHit : searchHits) {
16. String name = (String) searchHit.getSource().get("name");
17. String birth = (String) searchHit.getSource().get("birth");
18. String interest = (String) searchHit.getSource().get("interest");
19. System.out.println("-------------" + (++i) + "------------");
20. System.out.println(name);
21. System.out.println(birth);
22. System.out.println(interest);
23. }
25. }
26. }

控制台将输出全部文档

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. 输出：
2. -------------1------------
3. amy
4. 1995-11-10
5. singing dancing
6. -------------2------------
7. angle
8. 1996-05-01
9. singing reading
10. -------------3------------
11. jackson
12. 1995-08-01
13. basketball music reading
14. -------------4------------
15. jack
16. 1995-03-21
17. football music
18. -------------5------------
19. tom
20. 1995-09-11
21. reading music

**2.matchQuery(String name,Object  text)**

matchQuery("filedname","value")匹配单个字段，匹配字段名为filedname,值为value的文档

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. //单个匹配，搜索name为jack的文档
2. QueryBuilder queryBuilder = QueryBuilders.matchQuery("name", "jack");

此时会输出一条记录，因为只有一个name为jack的文档

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. -------------1------------
2. jack
3. 1995-03-21
4. football music

**3.multiMatchQuery(Object text, String... fieldNames)**

多个字段匹配某一个值

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. QueryBuilder queryBuilder = QueryBuilders.multiMatchQuery("music",
2. "name", "interest");//搜索name中或interest中包含有music的文档（必须与music一致）

控制台输出如下

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. -------------1------------
2. jack
3. 1995-03-21
4. football music
5. -------------2------------
6. tom
7. 1995-09-11
8. reading music
9. -------------3------------
10. jackson
11. 1995-08-01
12. basketball music reading

**4.wildcardQuery()模糊查询**

模糊查询，?匹配单个字符，\*匹配多个字符

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. WildcardQueryBuilder queryBuilder = QueryBuilders.wildcardQuery("name",
2. "\*jack\*");//搜索名字中含有jack文档（name中只要包含jack即可）

控制台输出如下

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. -------------1------------
2. jackson
3. 1995-08-01
4. basketball music reading
5. -------------2------------
6. jack
7. 1995-03-21
8. football music

5.使用BoolQueryBuilder进行复合查询  
  
使用must

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. //模糊查询
2. WildcardQueryBuilder queryBuilder1 = QueryBuilders.wildcardQuery(
3. "name", "\*jack\*");//搜索名字中含有jack的文档
4. WildcardQueryBuilder queryBuilder2 = QueryBuilders.wildcardQuery(
5. "interest", "\*read\*");//搜索interest中含有read的文档
7. BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();
8. //name中必须含有jack,interest中必须含有read,相当于and
9. boolQueryBuilder.must(queryBuilder1);
10. boolQueryBuilder.must(queryBuilder2);

控制台输出如下（name中必须含有jack,interest中必须含有read,只有一条文档匹配）：

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. -------------1------------
2. jackson
3. 1995-08-01
4. basketball music reading

使用should

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. WildcardQueryBuilder queryBuilder1 = QueryBuilders.wildcardQuery(
2. "name", "\*jack\*");//搜索名字中含有jack的文档
3. WildcardQueryBuilder queryBuilder2 = QueryBuilders.wildcardQuery(
4. "interest", "\*read\*");//搜索interest中含有read的文档
6. BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();
7. //name中含有jack或者interest含有read，相当于or
8. boolQueryBuilder.should(queryBuilder1);
9. boolQueryBuilder.should(queryBuilder2);

控制台输出（name中含有jack或者interest中含有read的都将被匹配）：

**[java]** [view plain](https://blog.csdn.net/lom9357bye/article/details/52852533) [copy](https://blog.csdn.net/lom9357bye/article/details/52852533)

1. ------------1------------
2. jackson
3. 1995-08-01
4. basketball music reading
5. -------------2------------
6. angle
7. 1996-05-01
8. singing reading
9. -------------3------------
10. jack
11. 1995-03-21
12. football music
13. -------------4------------
14. tom
15. 1995-09-11
16. reading music