1、 constant Query

该查询能够包含一个查询或过滤器，通过该方式将返回的文档的score设置为1， 然后通过设置boost来提高当前查询的权重（官方文档说返回的score和boost相等）， 提高该查询的相关度

QueryBuilder queryBuilder = QueryBuilders.boolQuery().must(QueryBuilders.constantScoreQuery(QueryBuilders.termQuery("hotelName", "酒")).boost(2f))

.should(QueryBuilders.constantScoreQuery(QueryBuilders.termQuery("hotelName", "店")).boost(1f))

应用场景（来源网络）： 比如匹配酒店设备：多个term 泳池，花园，wifi 比如我们要将泳池的的分值放大，则将泳池的term通过该方式包一下，并设置一个你认为合理的权重，提高泳池在酒店匹配过程的占比。

QueryBuilder queryBuilder = QueryBuilders.boolQuery().must(QueryBuilders.constantScoreQuery(QueryBuilders.termQuery("hotelName", "泳池")).boost(2f))

.should(QueryBuilders.constantScoreQuery(QueryBuilders.termQuery("hotelName", "花园")).boost(1f))

.should(QueryBuilders.constantScoreQuery(QueryBuilders.termQuery("hotelName", "wifi")).boost(1f))

2、bool query  
一种复合查询，三种逻辑关系如下 must： AND  
must\_not：NOT should：OR 复合查询： cityCode="chongqing\_city" 并且 hotelName="hotel22"

QueryBuilders.boolQuery().must(QueryBuilders.matchQuery("cityCode","chongqing\_city")).must(QueryBuilders.matchQuery("hotelName", "hotel22"))

3、 Dis Max Query

将子查询union 到一起，没个文档的分数是 子查询中相同文档的得分最大值。 例： 北京大饭店 酒索引得分 0； 店 得分 1； 大得分1.1 最后的结果是 北京大饭店相关度得分1.1

QueryBuilders.disMaxQuery()

.add(QueryBuilders.termQuery("hotelName","酒"))

.add(QueryBuilders.termQuery("hotelName","店"))

.add(QueryBuilders.termQuery("hotelName","大"));

4、boosting query

一种复合查询，分为positive子查询和negitive子查询，两者的查询结构都会返回。 positive子查询的score保持不变，negetive子查询的值将会根据negative\_boost（小于1）的值相乘，做相应程度的降低分值 hotelName 检索酒店，hotelNo 检索123 hotelNo检索获得的分值通过乘以0.3来降低分值，降低相关度

QueryBuilders.boostingQuery().positive(QueryBuilders.matchQuery("hotelName", "酒店")).negative(QueryBuilders.matchQuery("hotelNo","123")).negativeBoost(0.3f)

5、 indices Query

跨索引复合查询，实现跨索引的查询，并提供没有查询到结果是 从其他索引查询（默认是从其他所有索引进行匹配） 如下实例：从 索引1，索引2 中查询hotelName = "7天"的 term 查询。 如果没有查询到数据，从 索引3 查询hotelName = "置" 的term查询，如果没有指明索引3，默认从client下所有索引中进行匹配结果

QueryBuilder queryBuilder = QueryBuilders.indicesQuery(QueryBuilders.termQuery("hotelName", "7天"), "索引1","索引2")

.noMatchQuery(QueryBuilders.termQuery("hotelName", "置"));

System.out.println(queryBuilder);

SearchResponse response = client.prepareSearch("索引3")

.setSearchType(SearchType.DFS\_QUERY\_THEN\_FETCH)

.execute()

.actionGet();

6、 and Query

与 请求查询，要求必须满足所有term查询 如下实例：hotelName 必须同时满足 7 和 天 两个term查询 不过现在该方法再client已经标记为过时，推荐使用boolQuery 的must来实现

QueryBuilder queryBuilder = QueryBuilders.andQuery(QueryBuilders.termQuery("hotelName", "7")).add(QueryBuilders.termQuery("hotelName", "天"));

相当于

QueryBuilder queryBuilder = QueryBuilders.boolQuery().must(QueryBuilders.termQuery("hotelName", "7")).must(QueryBuilders.termQuery("hotelName", "天"));

7、 or query

或 请求查询，要求满足其中一个term查询即可，现在也属于过时的方法，推荐使用boolQuery来代替。 略过，

8、 filtered Query

过滤查询： 根据第一个term来查询结果，通过第二个term进行过滤 已经标记为过时方法：推荐使用：boolQuery的 must来实现 如下实例：查询hotelName 为 "7"的文档，并过滤hotelName为天的作为结果文档

QueryBuilder queryBuilder = QueryBuilders.filteredQuery(QueryBuilders.termQuery("hotelName", "7"),QueryBuilders.termQuery("hotelName","天"));

9、limit Query

没实现出来，通过看源代码提示结合文档，得出如下结论 限制每个分片的检索数量，已经标记为过时 ，通过SearchRequestBuilder#setTerminateAfter(int) 代替实现 如下实例：没个分片查询1个数据，

QueryBuilder queryBuilder = QueryBuilders.termQuery("hotelName","酒");

System.out.println(queryBuilder);

SearchResponse response = client.prepareSearch("titan\_hotel")

.setSearchType(SearchType.DFS\_QUERY\_THEN\_FETCH)

.setQuery(queryBuilder).setTerminateAfter(1)

.execute()

.actionGet();