多个MUST的Query的文档号合并

这种Query组合的文档号合并的代码是在ConjunctionDISI类中实现。本文通过一个例子来介绍文档号合并逻辑，这篇文章中对于每个关键字如何获得包含它的文档号，不作详细描述，大家可以去看我添加了详细注释的ConjunctionDISI类，相信能一目了然。GitHub地址是：https://github.com/luxugang/Lucene-7.5.0/blob/master/solr-7.5.0/lucene/core/src/java/org/apache/lucene/search/ConjunctionDISI.java。

例子

添加10篇文档到索引中。如下图： 图1：

```
Document doc ;
// 0
doc = new Document();
doc.add(new TextField( name: "content",  value: "a", Field.Store.YES));
indexWriter.addDocument(doc);
// 1
doc = new Document();
doc.add(new TextField( name: "content",  value: "b", Field.Store.YES));
indexWriter.addDocument(doc);
// 2
doc = new Document();
doc.add(new TextField( name: "content",  value: "c b", Field.Store.YES));
indexWriter.addDocument(doc);
// 3
doc = new Document();
doc.add(new TextField( name: "content",  value: "a c", Field.Store.YES));
indexWriter.addDocument(doc);
// 4
doc = new Document();
doc.add(new TextField( name: "content",  value: "h", Field.Store.YES));
indexWriter.addDocument(doc);
// 5
doc = new Document();
doc.add(new TextField( name: "content",  value: "c e", Field.Store.YES));
indexWriter.addDocument(doc);
// 6
doc.add(new TextField( name: "content",  value: "c a", Field.Store.YES));
indexWriter.addDocument(doc);
// 7
doc = new Document();
doc.add(new TextField( name: "content",  value: "f e", Field.Store.YES));
indexWriter.addDocument(doc);
// 8
doc = new Document();
doc.add(new TextField( name: "content",  value: "a c d e c e", Field.Store.YES));
indexWriter.addDocument(doc);
// 9
doc = new Document();
doc.add(new TextField( name: "content",  value: "a c e a b c", Field.Store.YES));
indexWriter.addDocument(doc);
indexWriter.commit();
```

使用WhiteSpaceAnalyzer进行分词。 查询条件如下图，**MUST描述了满足查询要求的文档必须包含"a"、"b"、"c"、"e" 四个关键字。** 图2：

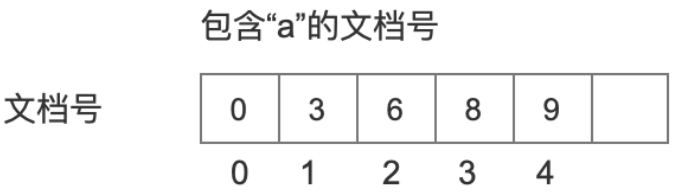
```
BooleanQuery.Builder builder = new BooleanQuery.Builder();
builder.add(new TermQuery(new Term( fld: "content",  text: "a")), BooleanClause.Occur.MUST);
builder.add(new TermQuery(new Term( fld: "content",  text: "b")), BooleanClause.Occur.MUST);
builder.add(new TermQuery(new Term( fld: "content",  text: "c")), BooleanClause.Occur.MUST);
builder.add(new TermQuery(new Term( fld: "content",  text: "e")), BooleanClause.Occur.MUST);
```

文档号合并

将包含各个关键字的文档分别放入到各自的数组中，数组元素是文档号。

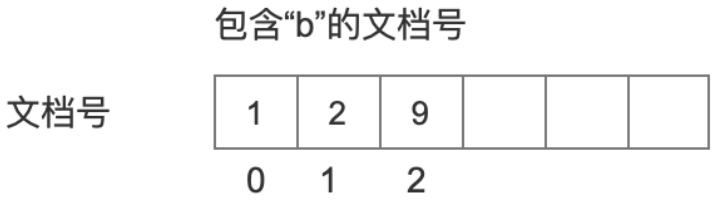
包含“a”的文档号

图3：



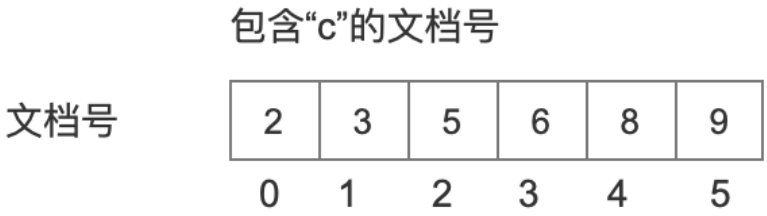
包含“b”的文档号

图4：



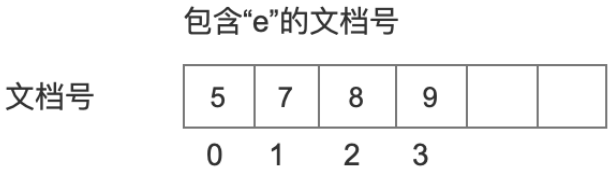
包含“c”的文档号

图5：



包含“e”的文档号

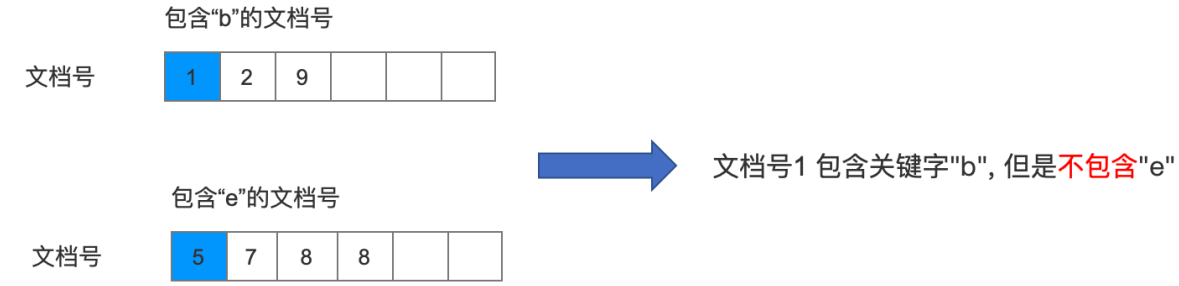
图6：



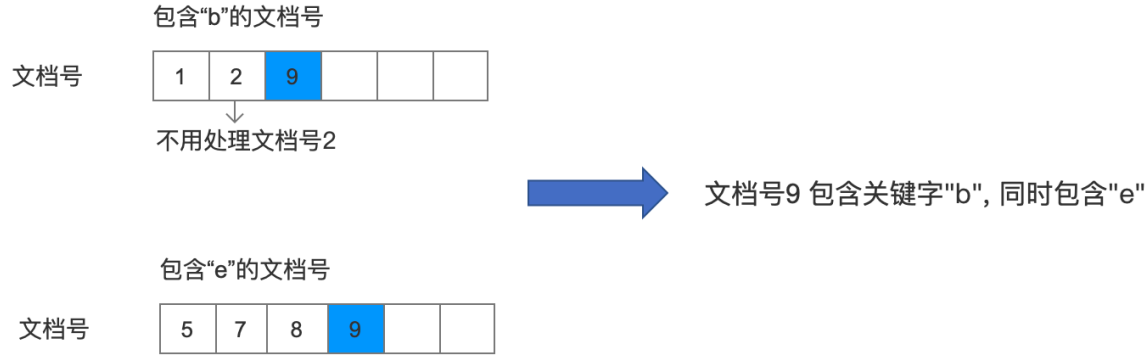
由于满足查询要求的文档中必须都包含"a"、"b"、"c"、"e" 四个关键字，所以满足查询要求的文档个数最多是上面几个数组中最小的数组大小。 所以合并的逻辑即遍历数组大小最小的那个，在上面的例子中，即包含"b"的文档号的数组。每次遍历一个数组元素后，再去其他数组中匹配是否也包含这个文档号。遍历其他数组的顺序同样的按照数组元素大小从小到大的顺序，即包含**"e"的文档号 ---> 包含"a"的文档号 ---> 包含"c"的文档号**。

合并过程

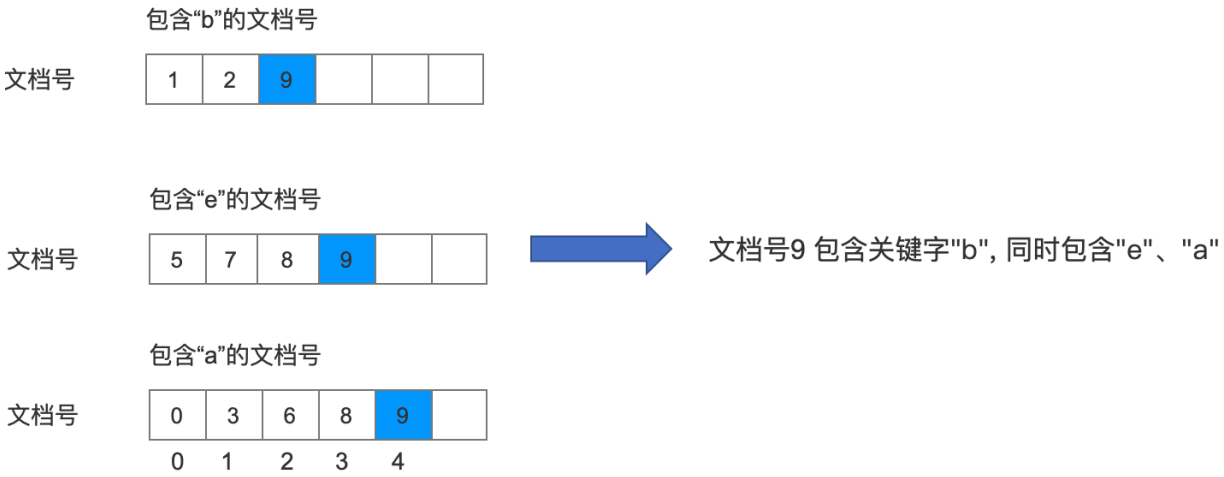
从包含"b"的文档号的数组中取出第一个文档号doc1的值 1，然后从包含"e"的文档号的数组中取出第一个不小于 doc1 (1)的文档号doc2的值，也就是5。 比较的结果就是 doc1 (1) ≠ doc2 (5)，那么没有必要继续跟其他数组作比较了。因为文档号1中不包含关键字"e"。 图7：



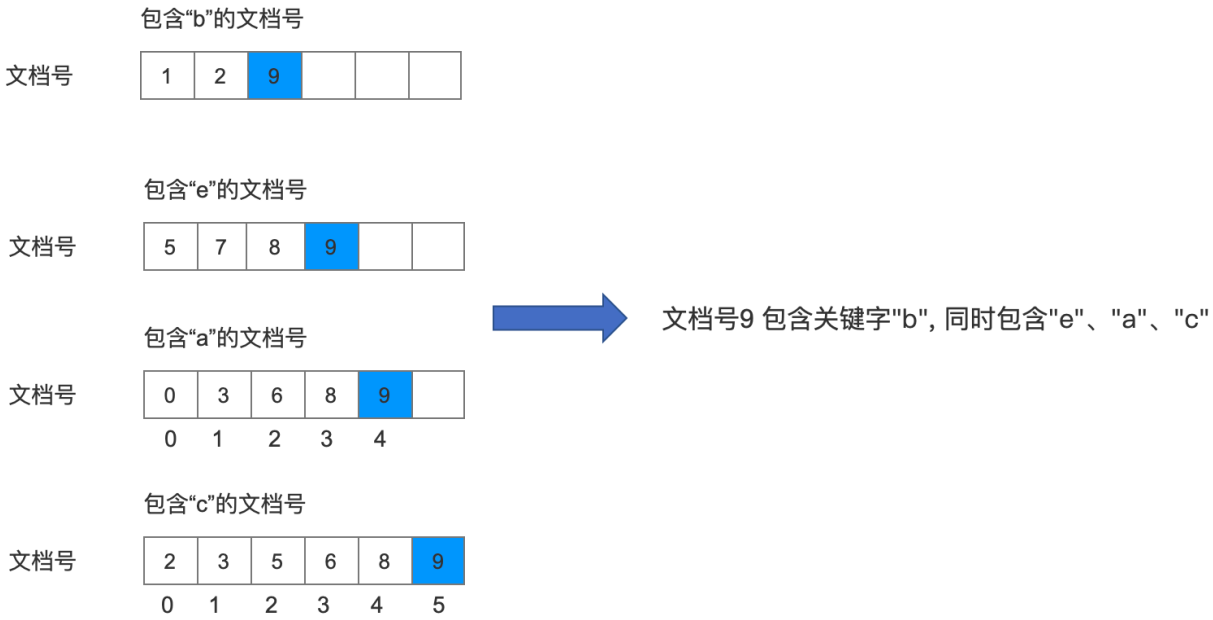
接着继续从包含"b"的文档号的数组中取出不小于doc2 (5)的值（**在图7的比较过程中，我们已经确定文档号1~文档号5中都不同时包含关键字"b"跟"e"，所以下一个比较的文档号并不是直接从包含"b"的文档号的数组中取下一个值，即2，而是根据包含"e"的文档号的数组中的doc2(5)的值，从包含"e"的文档号的数组中取出不小于5的值，即9**），也就是 9，doc1更新为9，然后再包含"e"的文档号的数组中取出不小于doc1(9)，也就是doc2的值被更新为 9： 图8：



比较的结果就是 doc1 (9) = doc2 (9), 那么我们就需要继续跟剩余的其他数组元素进行比较了，从包含"a"的文档号数组中取出不小于doc1 (9)的 文档号doc3的值，也就是 9： 图9：



这时候由于 doc1 (9) = doc3 (9)，所以需要继续跟包含"c"的文档号的数组中的元素进行比较，从包含"c"的文档号的数组中取出不小于doc1 (9)的文档号doc4的值，也就是9: 图10：



至此所有的数组都遍历结束，并且文档号9都在在所有数组中出现，即文档号9满足查询要求。

结语

本文介绍了使用BooleanQuery并且所有的TermQuery之间是MUST关系的文档号合并原理，在后面的文章中会依次介绍 SHOULD、MUST、MUST_NOT、FILTER的TermQuery的文档号合并原理。

点击下载Markdown文档