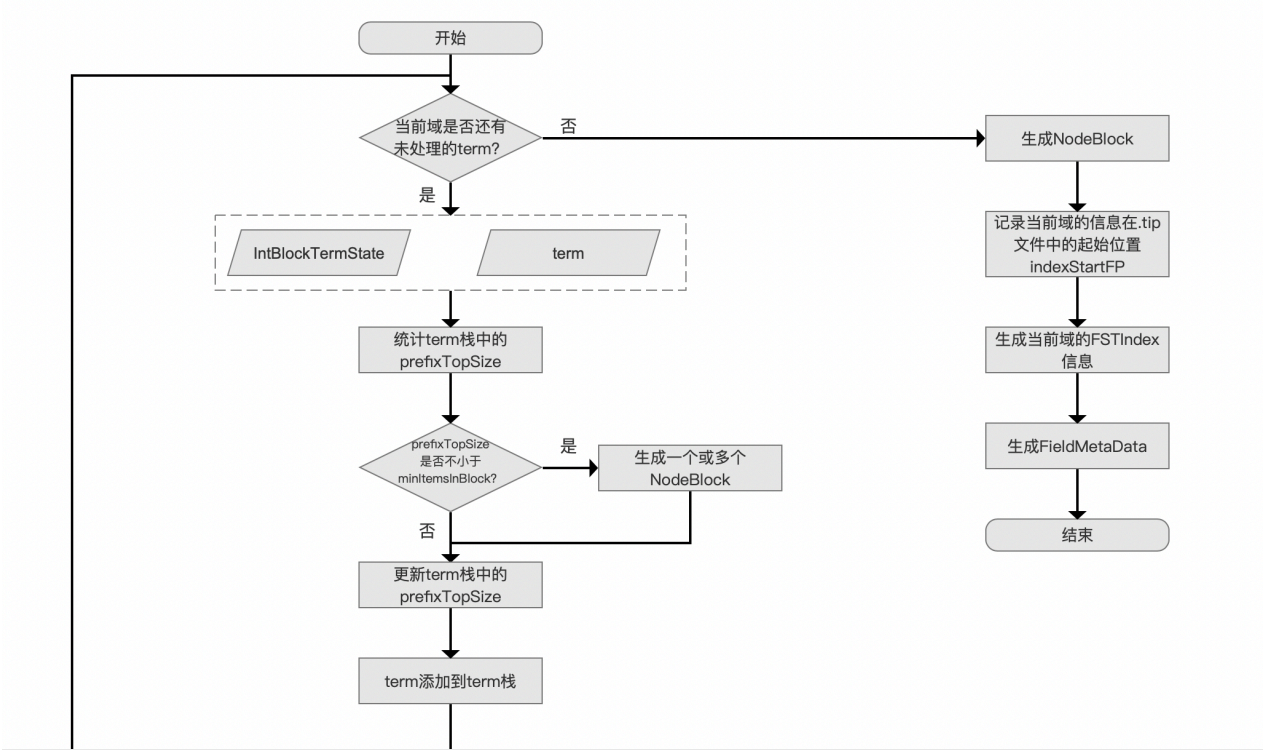
索引文件的生成（七）

本文承接索引文件的生成（六）继续介绍剩余的内容，下面先给出生成索引文件.tim、.tip的流程图。

生成索引文件.tim、.tip的流程图

图1：



统计每一个term的信息

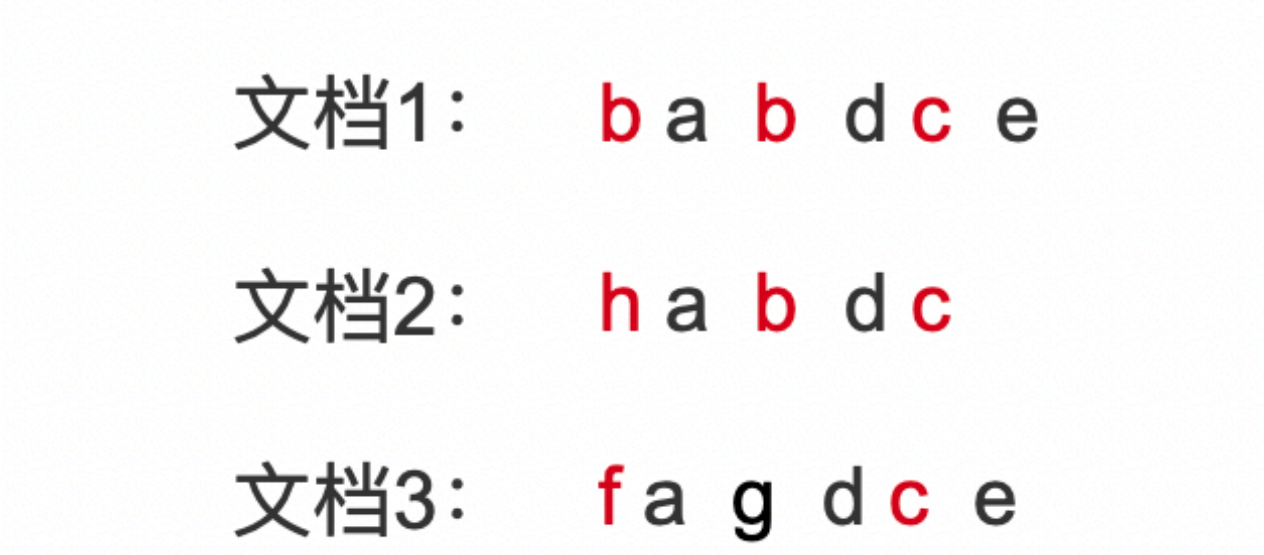
图2：

执行到该流程，我们需要将当前term的一些信息（图1中的IntBlockTermState，见文章索引文件的生成（五））的汇总到所属域的信息中（这里先提一下的是，这些信息在后面使用FieldMetaData封装），图2中出现的字段的含义如下：

- sumDocFreq：包含当前域的所有term的文档数量总和，注意的是当前域可能有多个term在同一文档中
- sumTotalTermFreq：当前域的所有term在所有文档中出现的次数总和
- numTerms：当前域中的term数量
- minTerm：当前域中最小（字典序）的term
- maxTerm：当前域中最大（字典序）的term

例如我们有如下几篇文档：

图3：



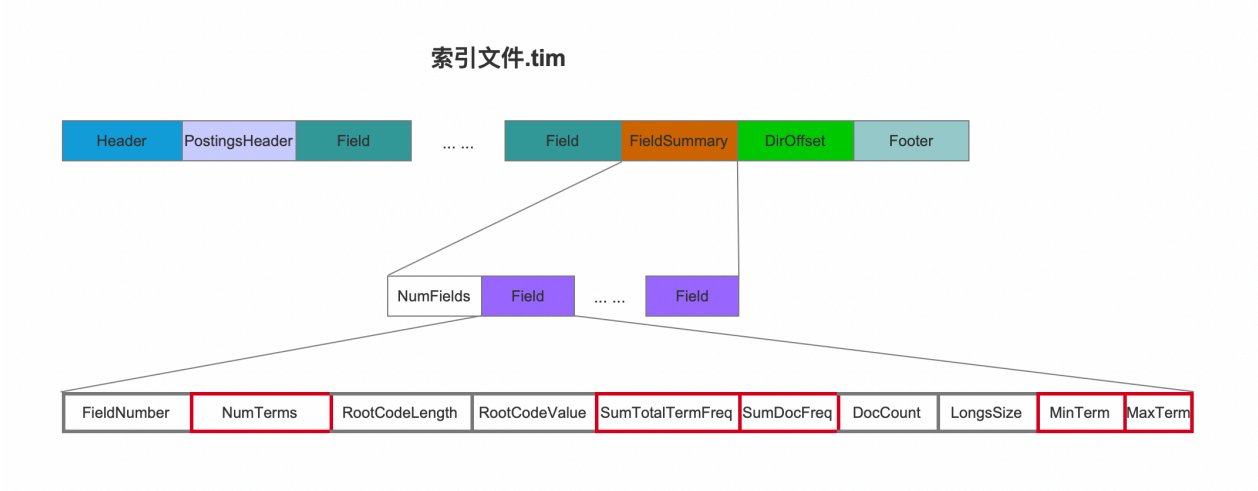
其中用红色标注的term属于域名为"content"的域，那么在处理完"content"之后，图2中的字段的值如下所示：

- sumDocFreq：b（2）+ c（3）+ f（1）+ h（1）= 7
- sumTotalTermFreq：b（3）+ c（3）+ f（1）+ h（1）= 8
- numTerms：b、c、f、h共4个term

- minTerm：b
- maxTerm：h

再处理完所有域之后，上述的信息在索引文件.tim中的位置如下：

图4：



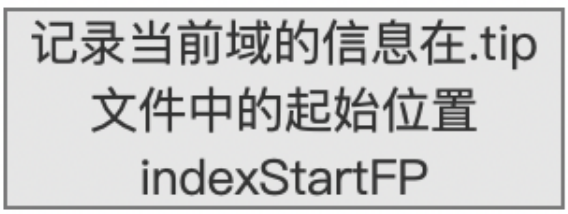
生成NodeBlock

图5：

当前域的所有term处理结束后，那么将term栈中剩余未处理的PendingEntry生成NodeBlock（见文章索引文件的生成（六））。

记录当前域的信息在.tip文件中的起始位置indexStartFP

图6：

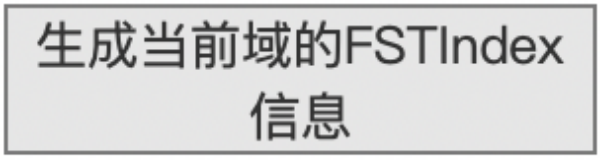


到此流程，Lucene将要在索引文件.tip中写入当前域的FSTIndex信息，在读取阶段，通过读取索引文件.tip中的FSTIndex信息来获取当前域在索引文件.tim的内容，而所有域的FSTIndex信息连续的存储在索引文件.tip中，那么需要indexStartFP来实现"索引"功能，如下图所示：

图7：

生成当前域的FSTIndex信息

图8：



在图5的流程中，当前域的所有term处理结束后，term栈中剩余未处理的PendingEntry会被处理为NodeBlock，最终只会生成一个PendingBlock（没明白？见文章索引文件的生成（六）），并且PendingBlock中的index信息，即FST信息将会被写入到FSTIndex中，由于本人还未对FST在Lucene中的应用有过文章的介绍，即使在本篇文章中列出FSTIndex中包含的字段信息，相信读者也无法理解，故只能通过几句话大概介绍下FSTIndex的内容以及功能：FSTIndex包含了当前域中的term的一些前缀值的信息，根据该信息就可以在索引文件.tip中找到每一种前缀值对应的NodeBlock，该NodeBlock中包含了具有该相同前缀值的所有term的信息。

在文章FST算法（上）中只是简单的介绍了FST的基本原理，而其在Lucene中的应用并没有展开介绍，故当完成应用篇的文章后，到时再来更新本篇文章的内容（先立个flag 😂）。

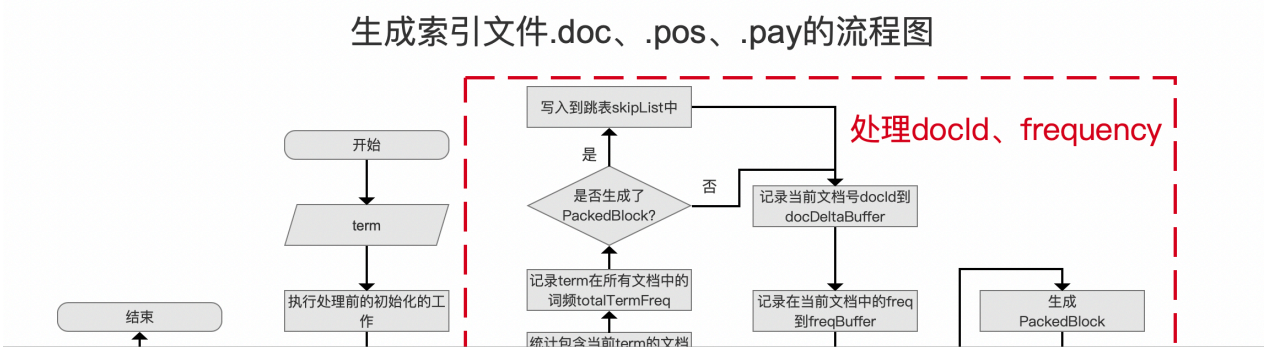
生成FieldMetaData

图9：

由于Lucene的处理逻辑是先处理所有的域，最后把这些域的信息写入一次性到索引文件.tip中，故在处理完一个域后，要将该域的信息通过FieldMetaData来存储下，当所有的域处理结束后，遍历所有的FieldMetaData，将这些信息依次到索引文件.tip中，故有了图7中的数据结构，FieldMetaData中只有一个信息需要介绍下，其他信息可以自行看源码中https://github.com/LuXugang/Lucene-7.5.0/blob/master/solr-7.5.0/lucene/core/src/java/org/apache/lucene/codecs/blocktree/BlockTreeTermsWriter.java 的内部类FieldMetaData：

- docCount：该值描述的是包含当前域的文档号数量，以图3为例，三篇文档都包含了域名为"content"的文档，所以docCount = 3，该值是在生成索引文件.doc、pos、pay（见文章索引文件的生成（一））的过程中统计的，统计的时机点如下图红框标注的流程点：

图10：



至此，生成索引文件.tim、.tip的流程介绍完毕。

结语

相信看完这七篇的系列文章后，大家对于索引文件.doc、.pos、.pay、.tim、tip的生成以及他们之间的关系有了深刻的了解，自己品。

点击下载附件