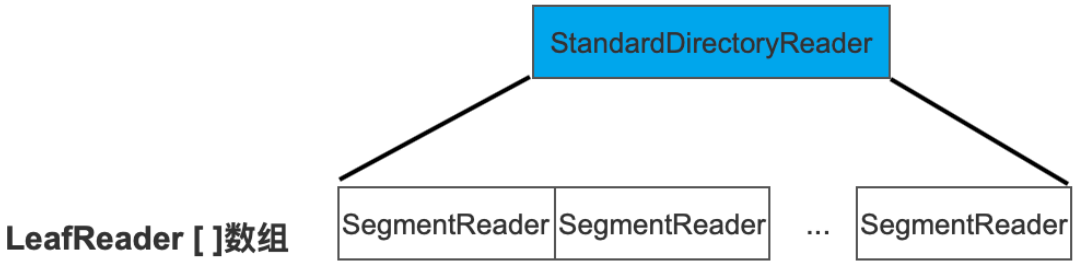
# SegmentReader（一）

在近实时搜索NRT的系列文章中，我们知道用于读取索引目录中所有索引信息的StandardDirectoryReader实际是使用了一个LeafReader数组封装了一个或者多个SegmentReader，而每一个SegmentReader则对应一个段中的索引信息，如下图所示：

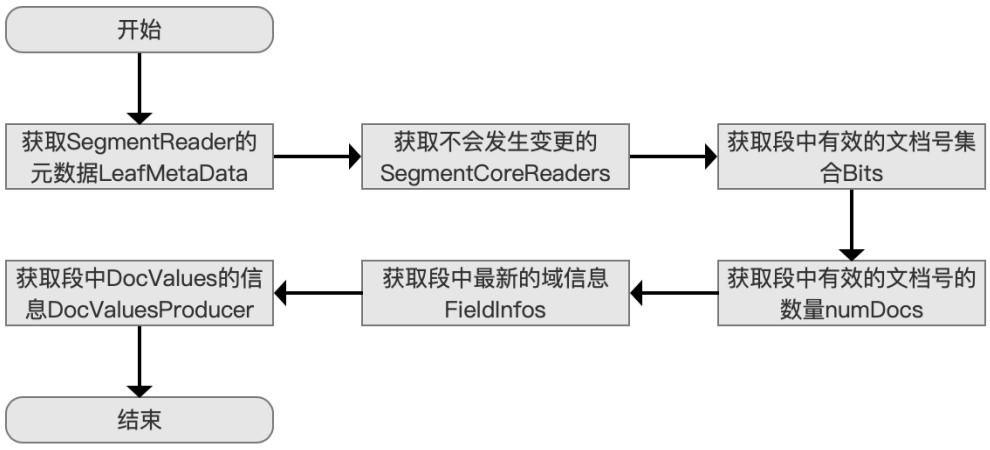
图1：




本篇文章将会介绍在生成SegmentReader的过程中，它获取了哪些具体的索引信息信息，**更重要的是，我们还会了解到为什么通过DirectoryReader.openIfChange()（见近实时搜索NRT（三））重新打开一个StandardDirectoryReader的开销会远远的小于DirectoryReader.open()方法（见近实时搜索NRT（一））**。

# 生成SegmentReader的流程图

图2：




## 获取SegmentReader的元数据LeafMetaData

LeafMetaData包含的信息如下：

- createdVersionMajor：该值描述了SegmentReader读取的段所属的Segment_N文件，它是创建时的Lucene的主版本（major）号，见文章索引文件之segments_N中的Version字段的介绍
- minVersion：该值通过.si索引文件获得，含义见文章索引文件之si中的SegVersion字段的Min.major、Min.minor、Min.bugfix的介绍
- sort：该值同样通过.si索引文件获得，描述了段中的文档的排序规则，其含义见文章索引文件之si中的IndexSort的介绍，不赘述

下图描述了LeafMetaData包含的信息对应在索引文件中的信息：

图3：

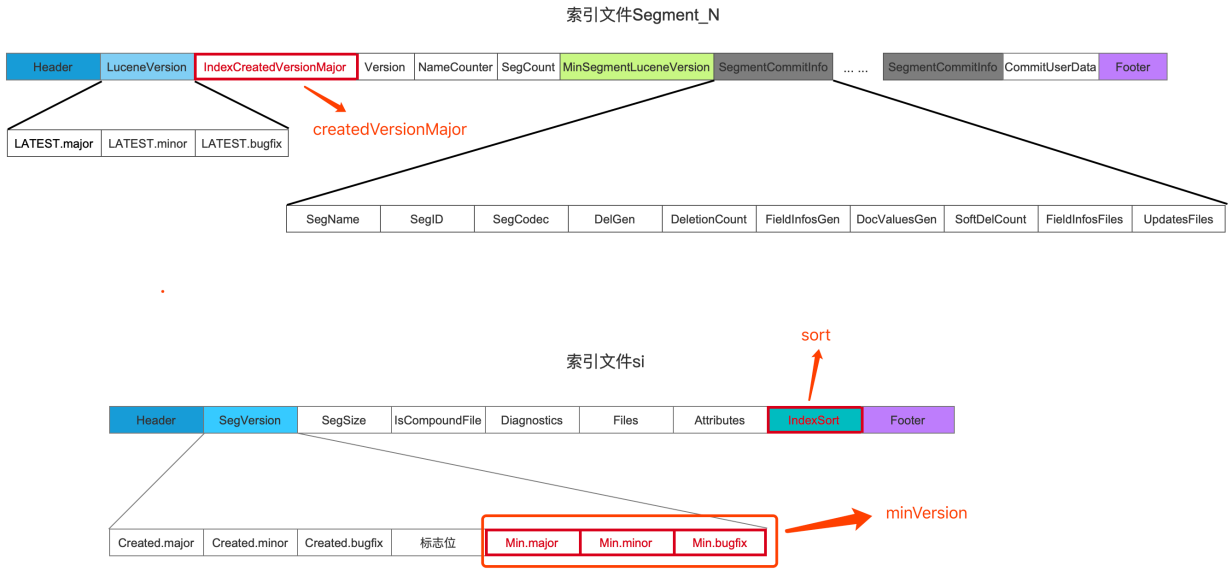



图3中的蓝色箭头描述的是我们可以通过segName找到索引目录中的.si索引文件，可以点击近实时搜索NRT（一）查看segName更详细的介绍。

# 获取不会发生变更的SegmentCoreReaders

我们首先介绍下SegmentCoreReaders包含了哪些主要信息：

- StoredFieldsReader：从索引文件fdx&&fdt中读取存储域的索引信息
- FieldsProducer：从索引文件tim&&tip、索引文件doc、索引文件pos&&pay中读取域的索引信息
- TermVectorsReader：从索引文件tvx&&tvd读取词向量的索引信息
- PointsReader：从索引文件dim&&dii中读取域值为数值类型的索引信息
- NormsProducer：从索引文件nvd&&nvm中读取域的打分信息
- FieldInfos：从索引文件fnm读取域的信息
- segment：段的前缀名

上文中涉及好几个读取索引文件的操作，这里并不会详细展开读取索引文件的过程，因为太简单了。

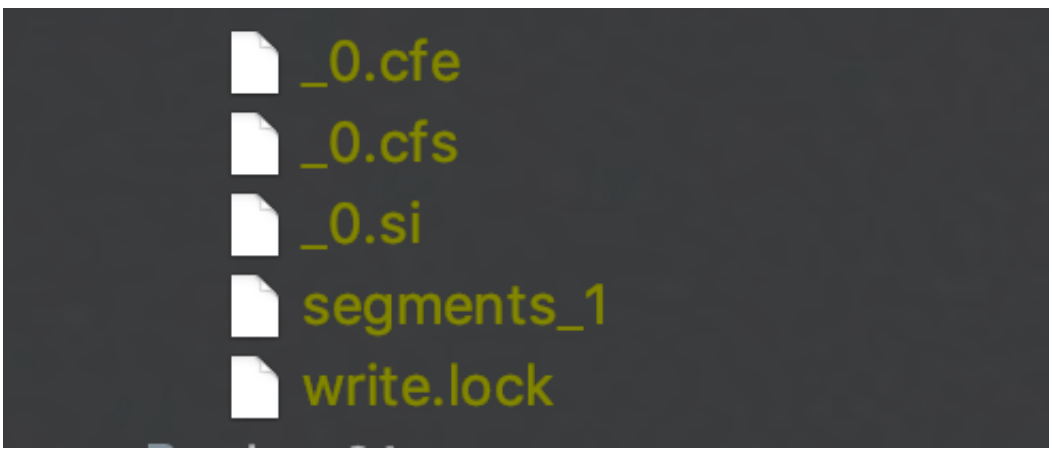
不过这里得提一下，使用和不使用复合索引文件cfs&&cfe时，读取索引文件的区别：

图4、图5分别是不使用复合索引文件和使用复合索引文件时索引目录中的文件列表：

图4：


_0.dii
_0.dim
_0.fdt
_0.fdx
_0.fnm
_0.nvd
_0.nvm
_0.si
_0.tvd
_0.tvx
_0_Lucene50_0.doc
_0_Lucene50_0.pos
_0_Lucene50_0.tim
_0_Lucene50_0.tip
_0_Lucene70_0.dvd
_0_Lucene70_0.dvm
segments_1
write.lock

图5：


_0.cfe
_0.cfs
_0.si
segments_1
write.lock

在不使用复合索引文件的情况下，获得SegmentCoreReaders只需要根据`段的前缀名`从索引目录中分别找到每一个非复合索引文件，随后读取即可，而在使用复合索引文件的情况下，我们需要先根据复合索引文件cfs&&cfe，才能读取每一个非复合索引文件的信息。

根据图4、图5的例子，复合索引文件的数据结构如下图所示：

图6：

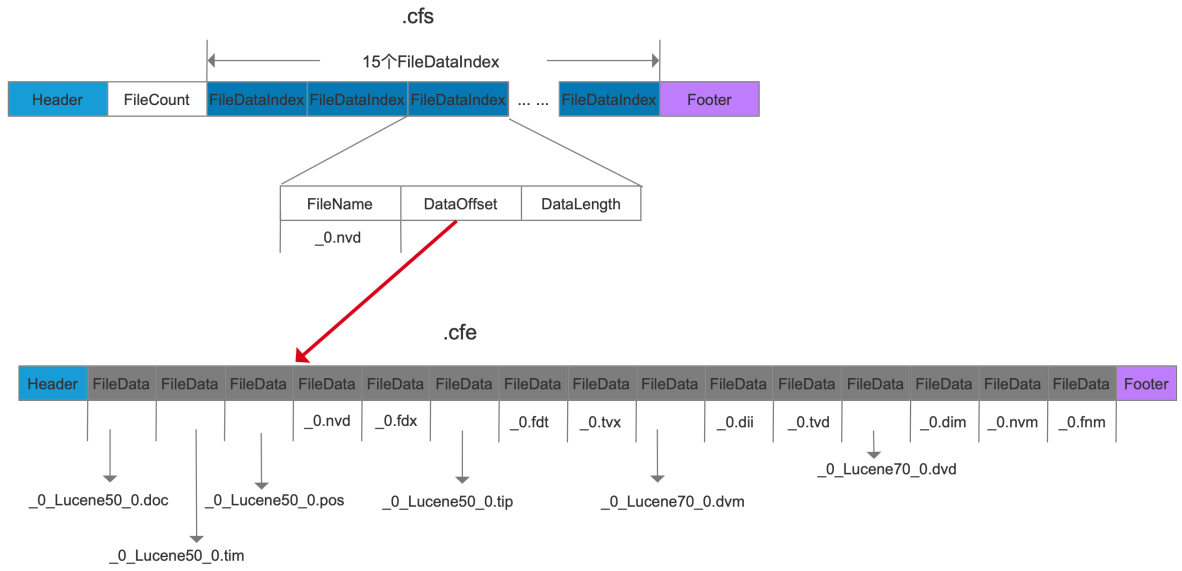



图6中每一个字段的含义已经在索引文件之cfs&&cfe的文章中介绍，不赘述。

SegmentCoreReaders中包含的索引信息中StoredFieldsReader、TermVectorsReader、PointsReader、NormsProducer是不会发生更改的内容，使得SegmentCoreReaders能够被复用，如果在未来的操作，该段中的索引信息发生更改，那么段中变更的索引信息会以其他索引文件来描述，这便是索引文件之liv、索引文件值.dvm、.dvd、索引文件之fnm，索引信息发生变更的情况以及描述变更的方式如下所示：

- 文档被删除：被删除的文档通过索引文件之liv来描述
- 文档被更新：文档的更新实际是先删除，后添加的过程，如果是更新DocValues，那么使用索引文件值.dvm、.dvd、索引文件之fnm来描述

**所以在我们通过DirectoryReader.openIfChange()获取最新的StandardDirectoryReader时，即使StandardDirectoryReader中某个或多个SegmentReader发生变更，我们可以直接复用SegmentCoreReaders中的索引信息，然后只需要更新（读取）相对开销较小的索引文件之liv、索引文件值.dvm、.dvd、索引文件之fnm这些索引文件即可。**

**这里的开销指的是什么：**

- 索引文件的信息在磁盘，这里的开销的大头实际就是读取索引文件的磁盘I/O

# 获取段中有效的文档号集合Bits

Bits中包含的是段中有效的文档号（live document id），使用FixedBitSet存放这些文档号，通过FixedBitSet其实就知道了那些无效的文档号（原因见FixedBitSet的介绍）。

**无效的文档号包含两种情况：**

- 情况一：添加一篇新的文档的过程中，该文档会先被安排一个文档号，然后再执行添加的逻辑，如果在添加的过程中发生错误导致未能正确的添加这篇文档，那么该文档的文档号被视为无效的（见文章文档提交之flush（三）中`处理出错的文档`的流程点）
- 情况二：被删除的文档也是被视为无效的

有效的文档号集合Bits通过索引文件之liv获得。

图7：

```
41
42      Document doc ;
43      // 0
44      doc = new Document();
45      doc.add(new TextField( name: "content",  value: "h", Field.Store.YES));
46      indexWriter.addDocument(doc);
47      // 1
48      doc = new Document();
49      doc.add(new TextField( name: "content",  value: "b", Field.Store.YES));
50      indexWriter.addDocument(doc);
51      // 2
52      doc = new Document();
53      doc.add(new TextField( name: "content",  value: "a c", Field.Store.YES));
54      indexWriter.addDocument(doc);
55      // 3
56      doc = new Document();
57      doc.add(new TextField( name: "content",  value: "a c e", Field.Store.YES));
58      indexWriter.addDocument(doc);
59      // 4
60      doc = new Document();
61      doc.add(new TextField( name: "content",  value: "h", Field.Store.YES));
62      indexWriter.addDocument(doc);
63      // 5
64      doc = new Document();
65      doc.add(new TextField( name: "content",  value: "i", Field.Store.YES));
66      indexWriter.addDocument(doc);
67      // 6
68      doc = new Document();
69      doc.add(new TextField( name: "content",  value: "c a e", Field.Store.YES));
70      indexWriter.addDocument(doc);
71      // 7
72      doc = new Document();
73      doc.add(new TextField( name: "content",  value: "f", Field.Store.YES));
74      indexWriter.addDocument(doc);
75      // 8
76      doc = new Document();
77      doc.add(new TextField( name: "content",  value: "b c d e c e", Field.Store.YES));
78      indexWriter.addDocument(doc);
79      // 9
80      doc = new Document();
81      doc.add(new TextField( name: "content",  value: "a c e a b c", Field.Store.YES));
82      indexWriter.addDocument(doc);
83      // 删除文档
84      indexWriter.deleteDocuments(new Term( fld: "content",  text: "h"));
85      indexWriter.deleteDocuments(new Term( fld: "content",  text: "f"));
86      indexWriter.commit();
```

图7中，代码的第84、85行，删除了包含"h"的文档号，那么文档0、4、7三篇文档会被删除，即上文中的无效的文档号，那么对应的.liv索引文件如下所示：

图8：

索引文件.liv

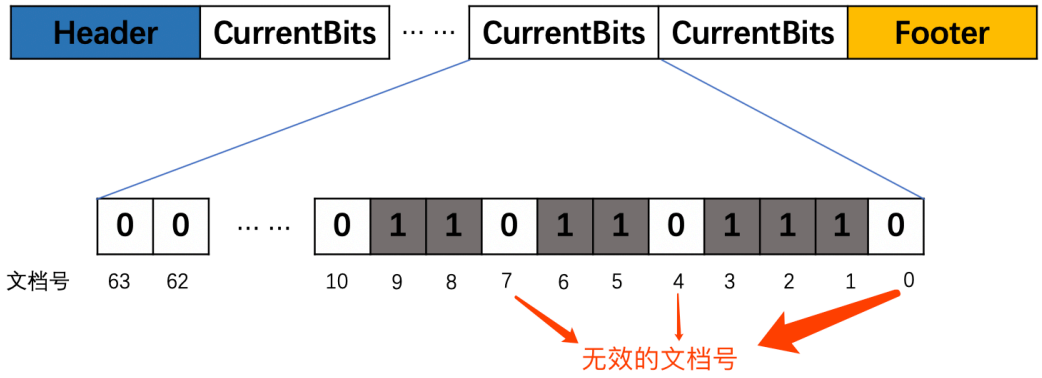



图8中各个字段的名字在文章索引文件之liv中已经介绍，不赘述，图8中的索引信息随后被读取到有效的文档号集合Bits中。

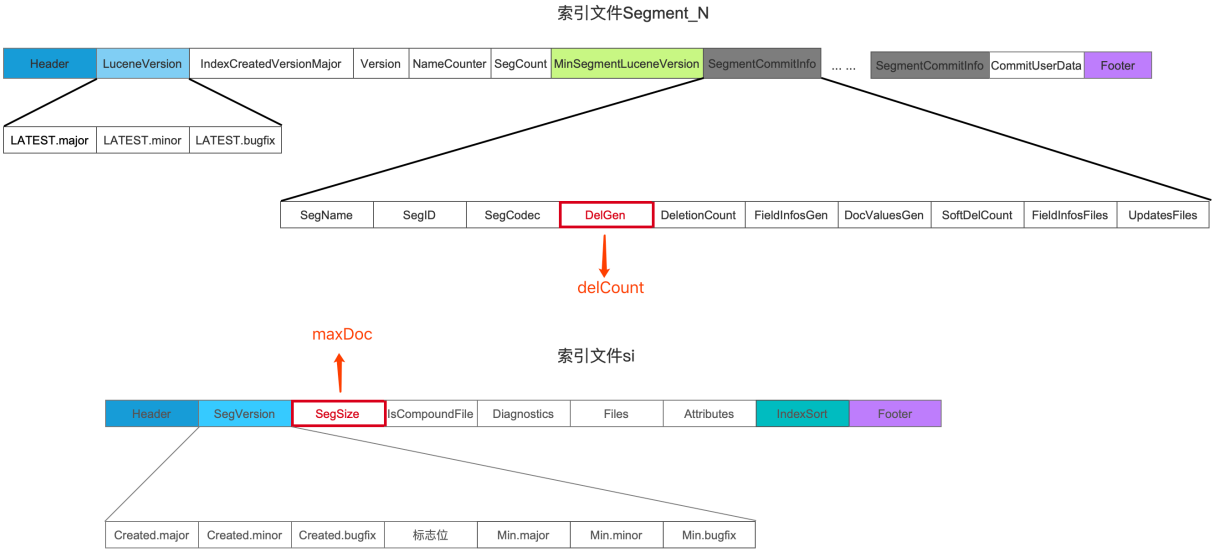
# 获取段中有效的文档号的数量numDocs

numDocs的值描述的是段中有效的文档号的数量，它的计算方式如下：

```
numDocs = maxDoc - delCount;
```

上述的计算方式中，maxDoc描述的是DWPT处理的文档总数（包含添加出错的文档），delCount描述的是无效的文档个数，这两个值在索引文件中的位置如下所示：

图9：




# 结语

基于篇幅，剩余的内容在下一篇文档中展开介绍。

点击下载附件