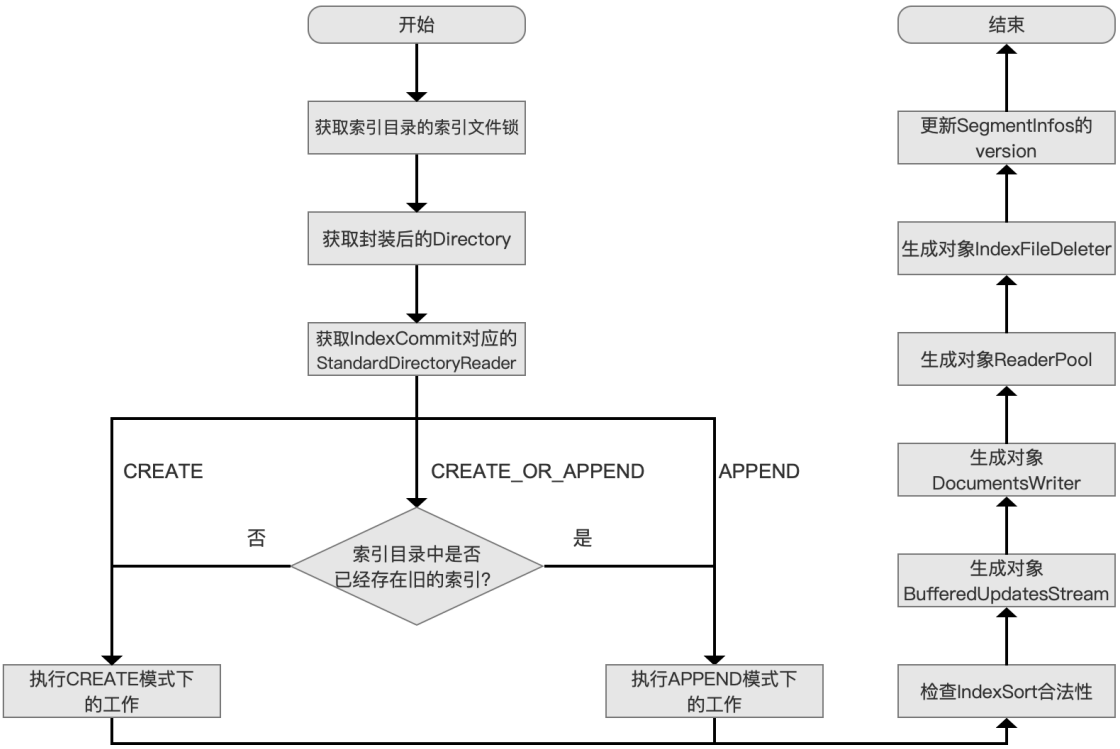
构造IndexWriter对象（五）

本文承接构造IndexWriter对象（四），继续介绍调用IndexWriter的构造函数的流程。

调用IndexWriter的构造函数的流程图

图1：

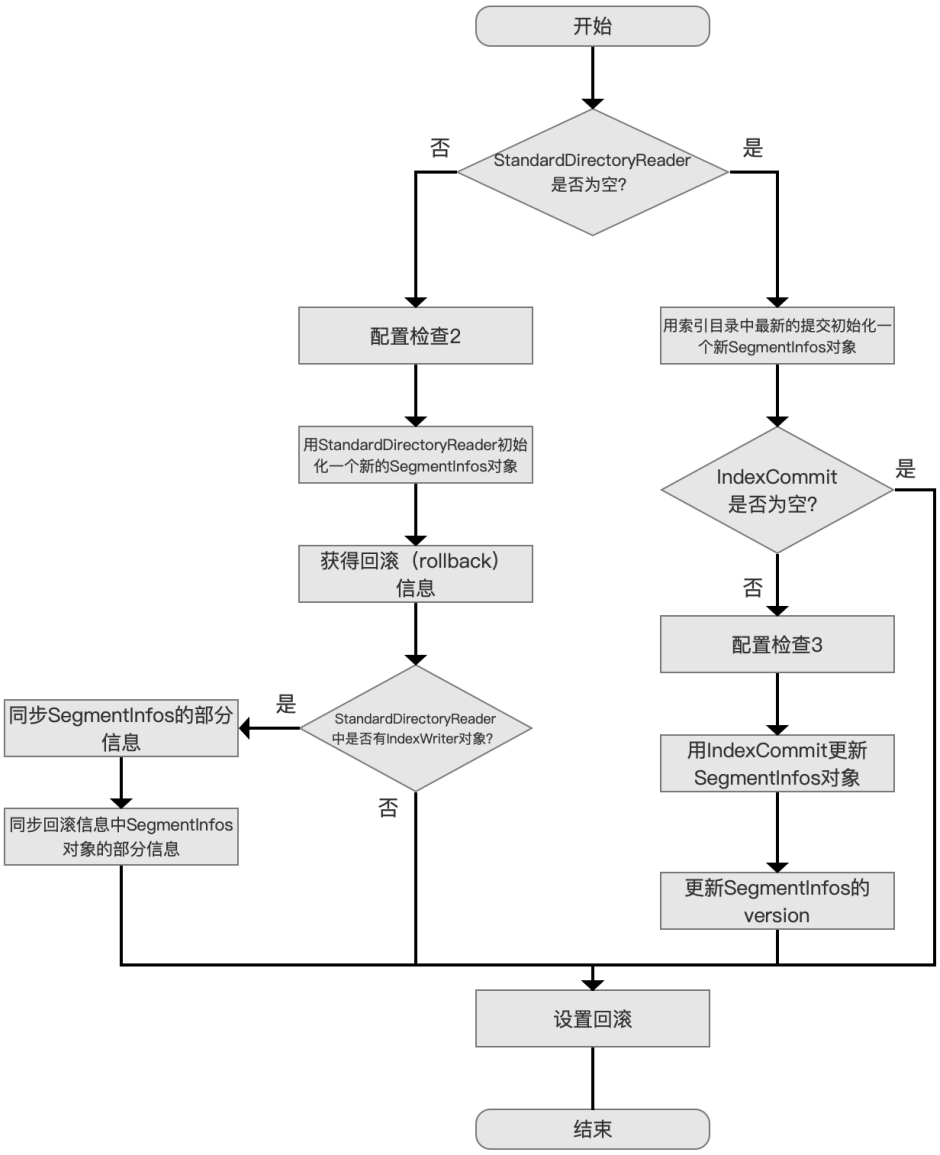


根据不同的OpenMode执行对应的工作

在上一篇文章中，我们介绍了执行APPEND模式下的工作的部分流程点，故继续对剩余的流程进行介绍。

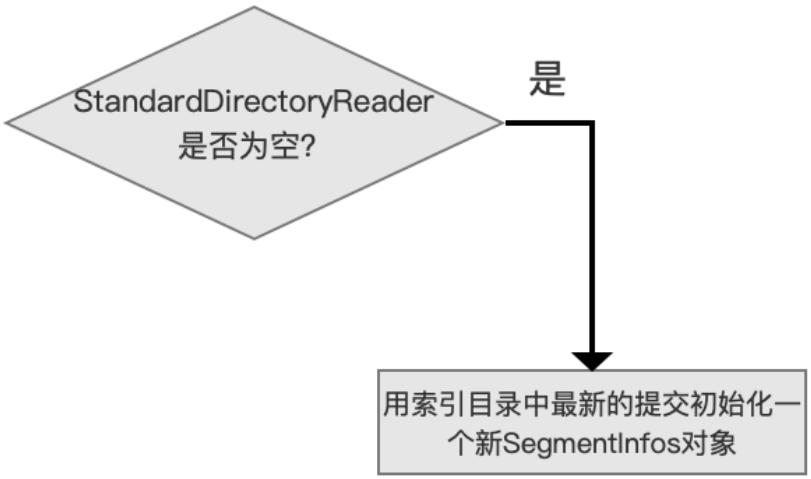
执行APPEND模式下的工作的流程图

图2：



用索引目录中最新的提交初始化一个新SegmentInfos对象

图3：



由于StandardDirectoryReader为空，那么就从索引目录中初始化一个新SegmentInfos对象（见构造IndexWriter对象（三）），即通过找到索引目录中的segment_N文件读取索引信息。

当索引目录中有多个segment_N文件时该如何选择：

- Lucene设定为读取最新的一次提交，即选取segment_N的N值最大的那个，因为N越大意味着更新的提交（commit()操作）

IndexCommit是否为空?

图4：

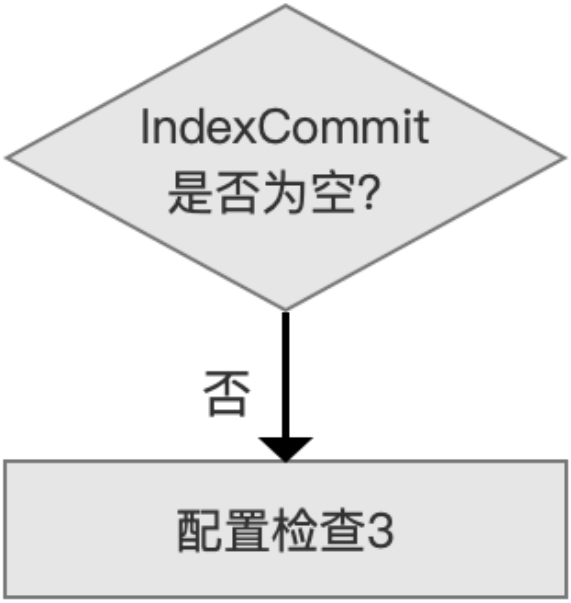
在构造IndexWriter对象（四）文章中我们说到，图2中StandardDirectoryReader为空的情况分为下面两种：

- 用户没有设置IndexCommit
- 用户设置了IndexCommit，但是IndexCommit中没有StandardDirectoryReader对象的信息

如果是第一种情况的进入到当前流程点，那么当前流程点的出口为是，那么以APPEND模式打开的IndexWriter追加的索引信息为索引目录中最新的一次提交。

配置检查3

图5：



如果IndexCommit不为空，那么IndexCommit必定是CommitPoint或者SnapshotCommitPoint对象，接着就需要执行下面的配置检查：

```
if (commit.getDirectory() != directoryOrig) {
            throw new IllegalArgumentException("IndexCommit's directory doesn't
match my directory, expected=" + directoryOrig + ", got=" +
commit.getDirectory());
          }
```

其中commit即IndexCommit对象、directoryOrg为IndexWriter的工作目录，这个配置检查意味着要求当前构造的IndexWriter的工作目录必须和IndexCommit对应的索引信息所在的目录必须一致

用IndexCommit更新SegmentInfos对象

图6：

通过IndexCommit读取对应的索引信息，然后更新到上文中已经完成初始化的SegmentInfos对象中。

更新SegmentInfos的version

图7：

对一个已经初始化的SegmentInfos进行更新操作必然需要更新version，version的概念在构造IndexWriter对象（三）的文章中介绍，这里不赘述。

至此，我们介绍完了分别在CREATE、APPEND、CREATE_AND_APPEND模式下的执行流程，接着我们根据图1介绍剩余的流程点。

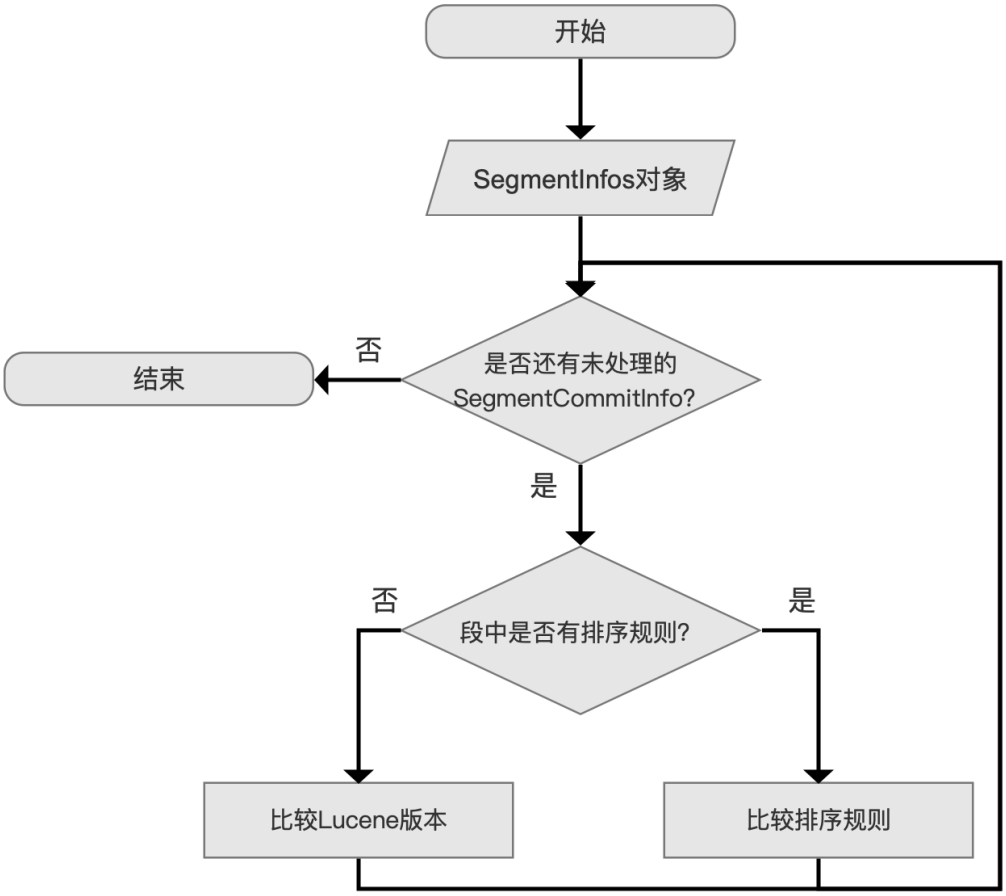
检查IndexSort合法性

图8：

如果设置了IndexSort，那么在生成一个段的过程中，Lucene会根据用户提供的排序规则对段内的文档进行排序，关于IndexSort的详细介绍见文章构造IndexWriter对象（一）， 如果用户通过IndexWriterConfig.setIndexSort(Sort sort)设置了IndexSort配置，那么需要对参数Sort进行合法性检查，检查逻辑如下所示：

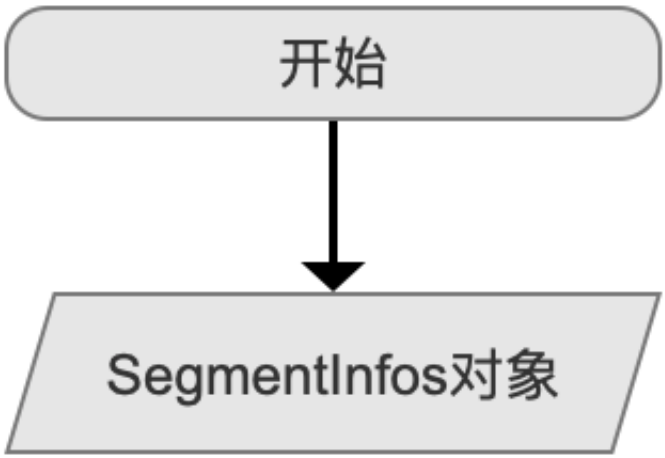
检查IndexSort合法性的流程图

图9：



SegmentInfos对象

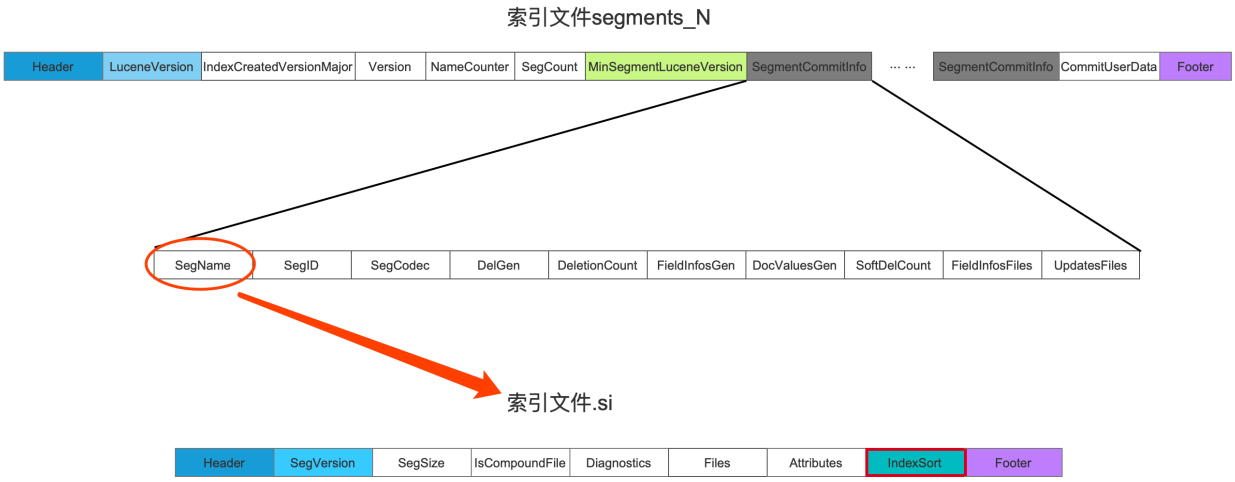
图10：



为什么检查IndexSort合法性的准备数据是SegmentInfos对象：

SegmentInfos对象是索引文件segment_N跟.si文件在内存中的描述，如下图所示：

图11：



由图11可以看出，我们只能通过SegmentInfos找到每一个段（图11中的SegmentCommitInfo）的段内排序规则IndexSort（图11总红色标注）。

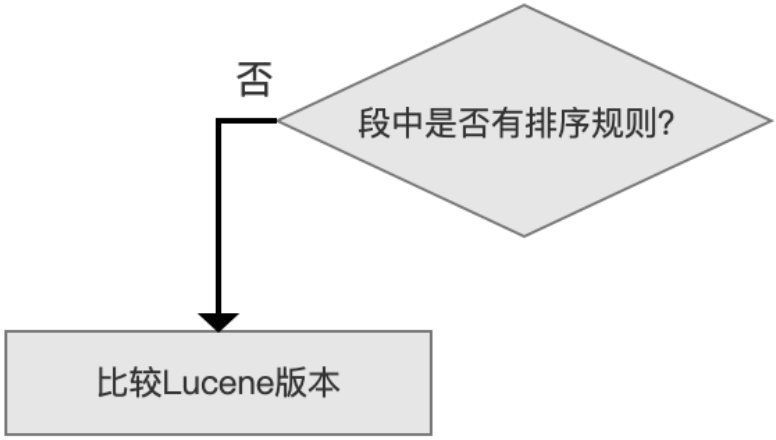
是否还有未处理的SegmentCommitInfo?

图12：

对每一个SegmentCommitInfo（见图11）进行IndexSort合法性检查，只要有一个段判断为非法，那么就抛出异常，即构造IndexWriter对象失败。

比较Lucene版本

图13：



如果通过图11的索引文件.si中的IndexSort字段来判断出段中没有排序规则，那么需要判断生成该段的Lucene版本号，代码如下：

```
if (segmentIndexSort == null &&
info.info.getVersion().onOrAfter(Version.LUCENE_6_5_0)){
    throw new CorruptIndexException("segment not sorted with indexSort=" +
segmentIndexSort, info.info.toString());
}
```

上述代码中，segmentIndexSort为段中的排序规则，info.info.getVersion( )中，第一个info是SegmentCommitInfo，第二个info为segmentInfo对象（即索引文件.si在内存中的描述），getVersion( )获得值即图11中SegVersion。

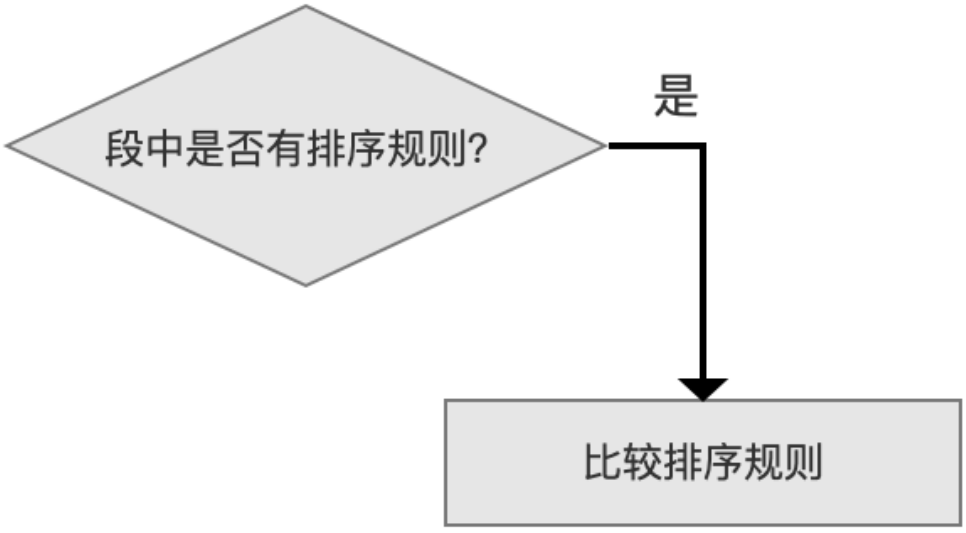
图13的流程描述的是，如果正在构造的IndexWriter对象设置了IndexSort配置，并且旧索引（旧索引指的是图1中执行三种打开模式流程获得的索引）中一个或多个段中没有排序规则，并且生成这些段的版本号大于等于6.5.0，那么就不能通过调用IndexWriter构造函数来读取旧的索引。

如何读取没有排序规则的段，并且生成这些段的Lucene版本号大于等于6.5.0的旧索引：

- 如果旧索引的版本号是Lucene7以上，那么通过IndexWriter.addIndexes(Directory... dirs)方法读取旧索引，该方法必须要求旧索引跟当前读取索引的Lucene主版本（即图11中索引文件segment_N的IndexCreatedVersionMajor字段的值）是一致的。下面的这个demo演示了如何添加上述旧索引：https://github.com/LuXugang/Lucene-7.5.0/blob/master/LuceneDemo/src/main/java/lucene/index/ValidateIndexSort.java 。
- 如果旧索引的版本号是Lucene7以下并且是Lucene6以上，可以通过DirectoryReader.open(Directory directory)的方式读取
- 如果旧索引的版本号是Lucene6以下，那么无法读取

比较排序规则

图14：



如果旧索引中的段包含排序规则，那么需要判断是否与正在构造中的IndexWriter设置的排序规则一致，不一致则抛出异常，如下所示：

```
if (segmentIndexSort != null && indexSort.equals(segmentIndexSort) == false) {
    throw new IllegalArgumentException("cannot change previous indexSort=" +
segmentIndexSort + " (from segment=" + info + ") to new indexSort=" +
indexSort);
}
```

其中segmentIndexSort为段中的排序规则，indexSort为IndexWriter配置的排序规则。

结语

基于篇幅，剩余的流程点将在下一篇文章中展开。

点击下载附件