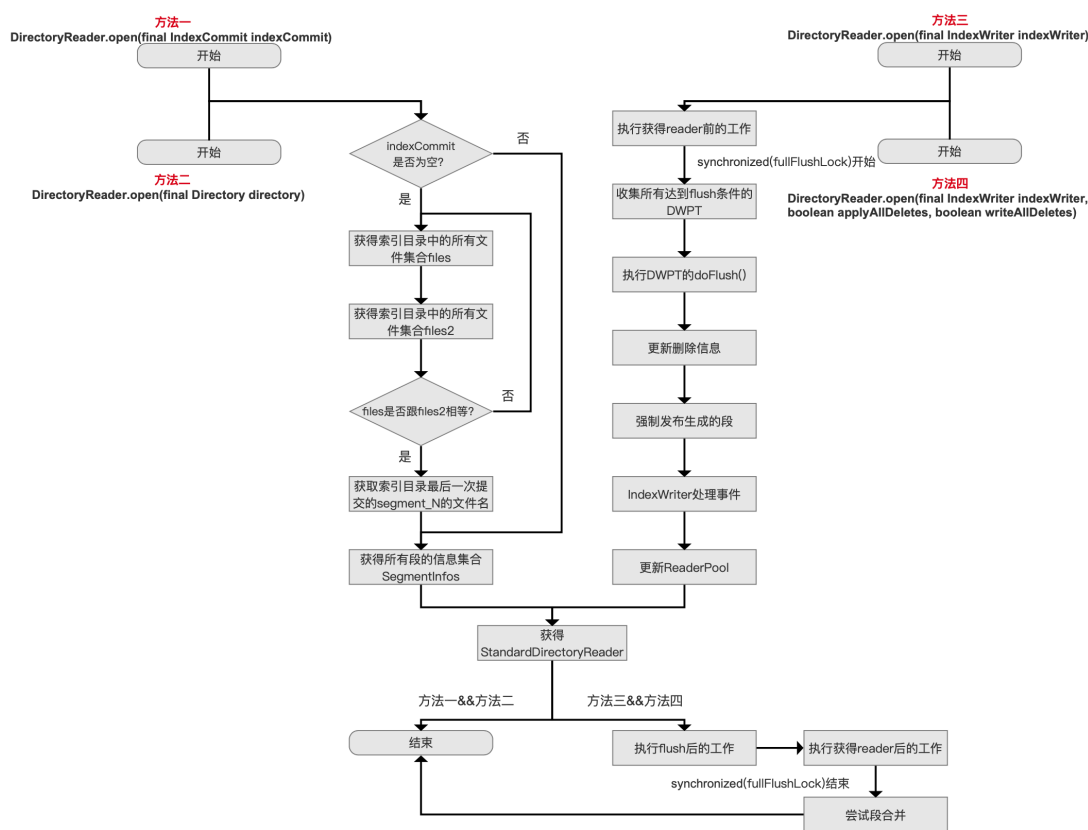


近实时搜索NRT (二)

本文承接[近实时搜索NRT \(一\)](#)，继续依次介绍每一个流程点。

获取StandardDirectoryReader对象的流程图

图1:



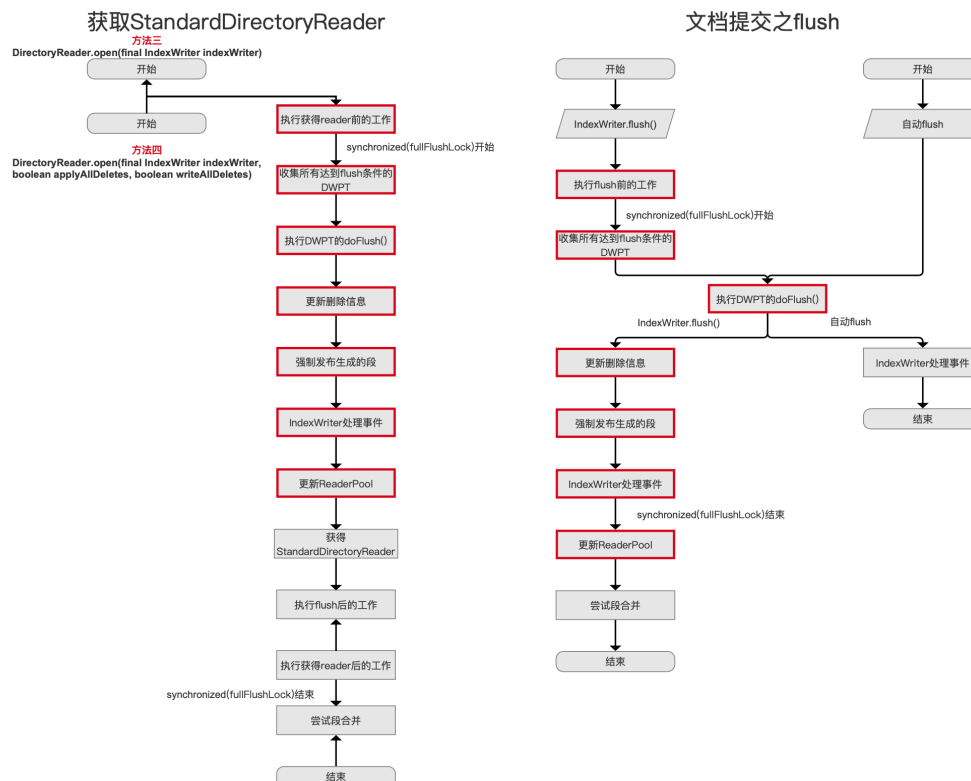
[点击查看大图](#)

我们继续介绍方法三&&方法四的所有流程点。

这两种是业务中最常使用，通过IndexWriter实现NRT功能的方法，在构造IndexWriter对象期间，会读取Directory，即索引目录中的已存在的索引信息（旧的索引信息），而对索引信息的更改（新的索引信息）都需要通过IndexWriter对象，故通过IndexWriter对象，我们能获得Directory中新旧索引信息，实现NRT。

我们将方法三&&方法四的流程从图1中拆解出来，并且跟[文档提交之flush \(一\)](#)的流程进行对比，如下图所示：

图2:



[点击查看大图](#)

图2中两个流程图中用红色标注的描述了他们具有具有相同的流程。

为什么获取StandardDirectoryReader需要执行flush的操作：

- 执行更改索引信息操作之后，其变更的内容并不会马上生成新段或更新旧段，例如文档的添加，在主动flush或者自动flush之前（见[文档提交之flush（一）](#)），新增的文档信息被保存在DWPT（见[文档的增删改（中）](#)）中；文档的删除/更新，其删除信息被保存在删除队列（见[文档的增删改（下）（part 2）](#)）中，只有在flush后，这些变更的信息才会生成新的段（见[文档提交之flush（三）](#)），即生成[近实时搜索NRT（一）](#)中的SegmentCommitInfo，它最终成为StandardDirectoryReader的一个LeafReader（见[近实时搜索NRT（一）](#)），即变更的索引信息能在搜索阶段能被读取，即NRT机制

为什么执行更改索引信息操作之后，其变更的内容并不马上生成新段或更新旧段：

- 假设我们每次添加一篇文档，就执行flush操作，那么一篇文档就会对应生成一个段，我们按照下面的条件分别介绍其导致的后果
 - 不使用段合并：索引目录中的段的个数跟文档个数相同，在[查询原理（二）](#)的文章中我们知道，查询阶段，我们分别从每一个段中执行查询操作，其性能可想而知
 - 使用段合并：根据段的合并策略LogMergePolicy或者TieredMergePolicy（默认策略），会导致及其频繁的段的合并操作，合并操作最可怕的地方就是在合并结束后需要跟磁盘同步，其磁盘同步性能影响在前面的文章已经介绍（见[文档提交之commit（一）](#)）

获得StandardDirectoryReader

图3：



在图2中红色标注的流程执行结束后，新旧索引信息都生成了SegmentCommitInfo，那么我们就可以获得StandardDirectoryReader了，其获得过程跟方法一&&方法二的 获得StandardDirectoryReader 是一致的，不赘述。

执行flush后的工作

图4：



该流程在前面的文章已经介绍，在源码中调用[DocumentsWriterFlushControl.finishFullFlush\(\)](#)的方法，详细的介绍见[文档提交之flush \(六\)](#)文章中的IndexWriter处理事件章节的内容。

执行获得reader后的工作

图5：

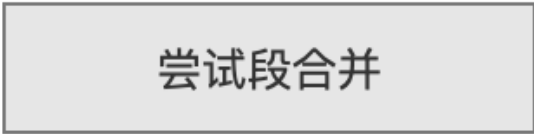


Lucene在当前流程点提供一个钩子函数doAfterFlush()方法，用户可以实现自己的业务逻辑，定义如下：

```
/**
 * A hook for extending classes to execute operations after pending added and
 * deleted documents have been flushed to the Directory but before the change
 * is committed (new segments_N file written).
 */
protected void doAfterFlush() throws IOException {}
```

尝试段合并

图6：



尝试段合并

由于执行了flush的操作，故索引可能发生了变化，在每一次索引发生变化后，都需要尝试判断是否需要执行段的合并操作，其判断条件依据不同的合并策略而有所不同，合并策略的文章可以看这里：[LogMergePolicy](#)、[TieredMergePolicy](#)。

结语

我们通过调用图1中的四个方法来获得索引目录中最新的索引信息，无论哪一种方法，目的就是索引目录中每一个段的信息生成一个LeafReader，最后将LeafReader封装为StandardDirectoryReader，然而这四种方法还存在性能问题，故Lucene提供了openIfChange的方法来提高NRT的性能，具体内容将在下篇文章中展开介绍。

[点击](#)下载附件