



目录

1

学习目标

2

IndexWriter详解

3

Document详解

4

索引更新



■ 学完本课题，你应达成如下目标：

1. 明白索引的过程
2. 掌握Lucene索引相关的概念
3. 熟练使用索引API创建、更新索引。
4. 会使用索引查看工具Luke



目录

1

学习目标

2

IndexWriter详解

3

Document详解

4

索引更新

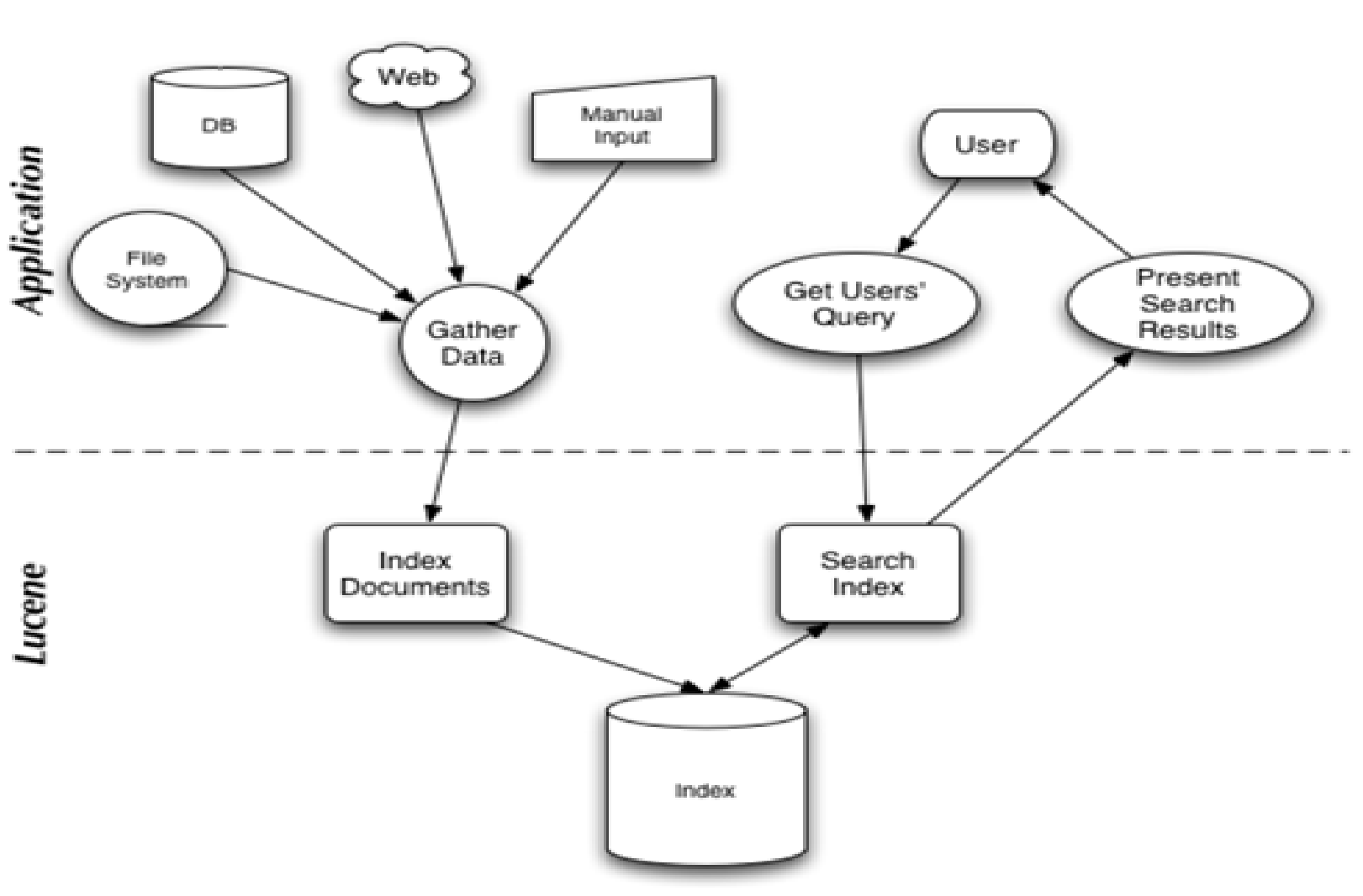


■ 回顾

问题1：索引创建过程完成什么事？

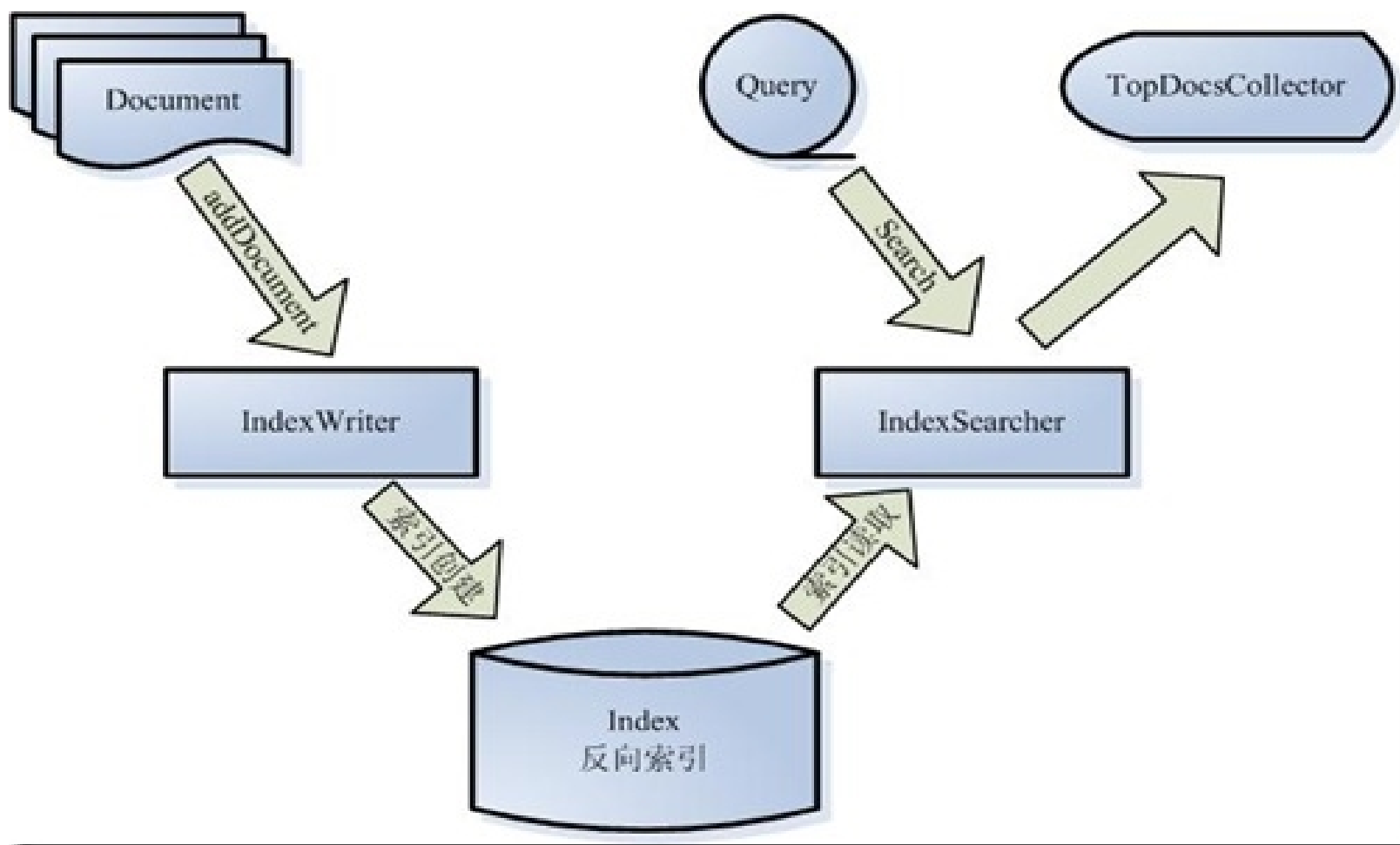


■ 回顾架构图





■ Lucene索引创建API 图示





■ Lucene索引创建代码示例

```

public static void main(String[] args) throws IOException {
    // 创建使用的分词器
    Analyzer analyzer = new IKAnalyzer4Lucene7(true);
    // 索引配置对象
    IndexWriterConfig config = new IndexWriterConfig(analyzer);
    // 设置索引库的打开模式：新建、追加、新建或追加
    config.setOpenMode(OpenMode.CREATE_OR_APPEND);

    // 索引存放目录
    // 存放到文件系统中
    Directory directory = FSDirectory
        .open((new File("f:/test/indextest")).toPath());

    // 存放到内存中
    // Directory directory = new RAMDirectory();

    // 创建索引写对象
    IndexWriter writer = new IndexWriter(directory, config);

    // 创建document
    Document doc = new Document();
    // 往document中添加 商品id字段
    doc.add(new StoredField("prodId", "p0001"));

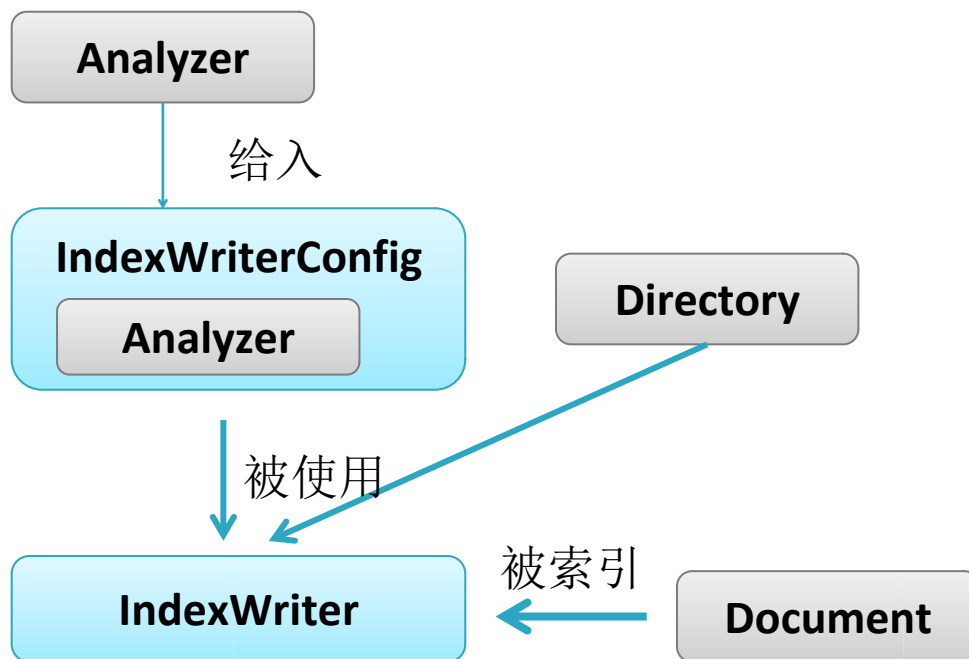
    // 往document中添加 商品名称字段
    String name = "ThinkPad X1 Carbon 20KH0009CD/25CD 超极本轻薄笔记本电脑联想";
    doc.add(new TextField("name", name, Store.YES));
}

```





■ IndexWriter涉及类图示





■ IndexWriterConfig 写索引配置：

- 使用的分词器，
- 如何打开索引（是新建，还是追加）。
- 还可配置缓冲区大小、或缓存多少个文档，再刷新到存储中。
- 还可配置合并、删除等的策略。

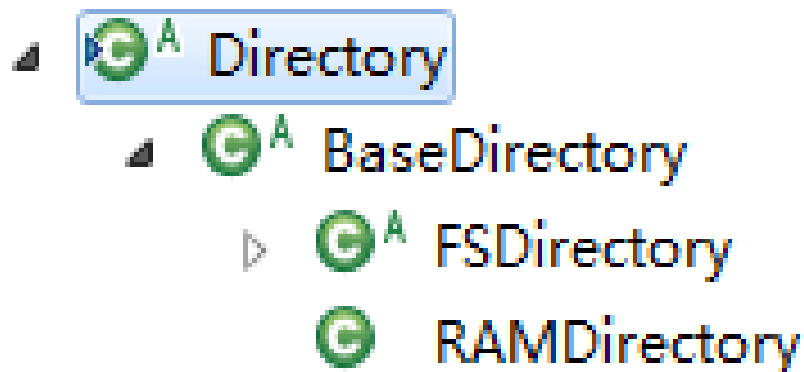
注意：用这个配置对象创建好IndexWriter对象后，再修改这个配置对象的配置信息不会对IndexWriter对象起作用。

如要在indexWriter使用过程中修改它的配置信息，通过 indexWriter的 getConfig()方法获得 LiveIndexWriterConfig 对象，在这个对象中可查看该 IndexWriter使用的配置信息，可进行少量的配置修改（看它的setter方法）



■ Directory 指定索引数据存放的位置：

- 内存
- 文件系统
- 数据库



保存到文件系统用法：

`Directory directory = FSDirectory.open(Path path);` // path指定目录



- **IndexWriter** 用来创建、维护一个索引。它的API使用流程：

```
// 创建索引写对象
IndexWriter writer = new IndexWriter(directory, config);

// 创建document

// 将文档添加到索引
writer.addDocument(doc);

// 删除文档
//writer.deleteDocuments(terms);

//修改文档
//writer.updateDocument(term, doc);

// 刷新
writer.flush();

// 提交
writer.commit();
```

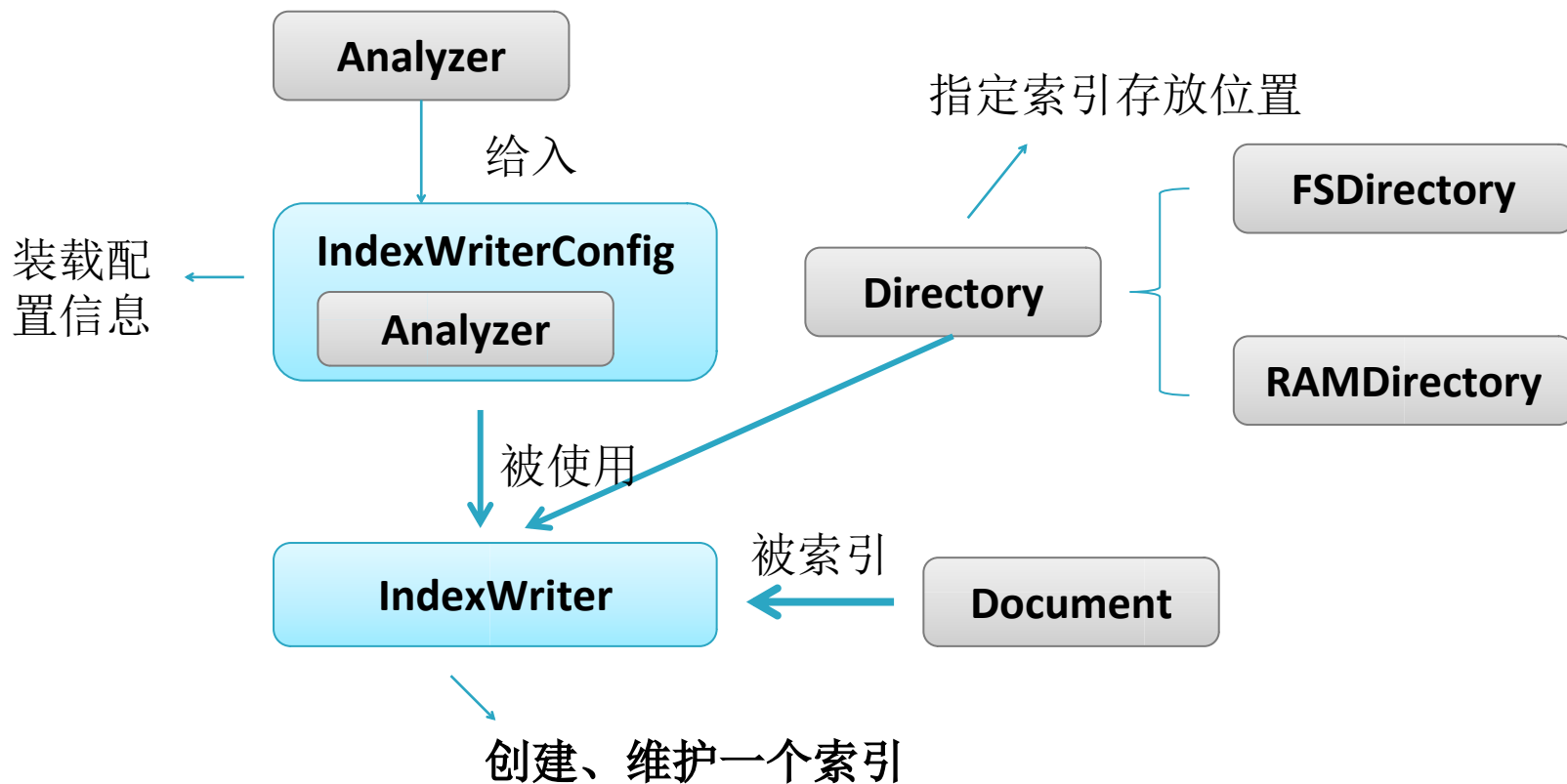
注意：IndexWriter是线程安全的。如果你的业务代码中有其他的同步控制，请不要使用IndexWriter作为锁对象，以免死锁。

请查看IndexWriter还提供了哪些：

- add方法
- delete方法
- update方法
- 其他方法

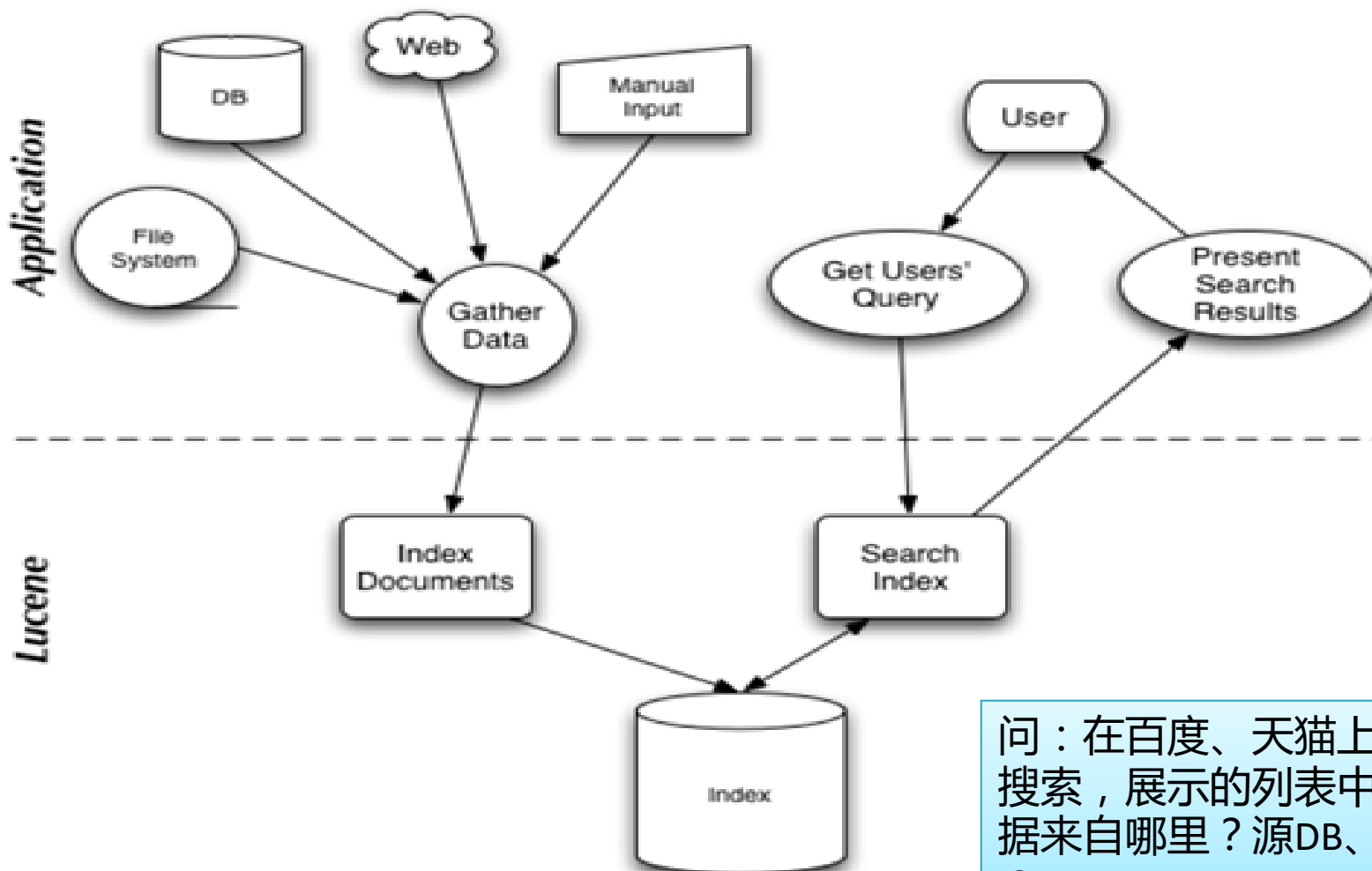


IndexWriter涉及类图示





■ 问题2：索引库中会存储反向索引数据，会存储document吗？



问：在百度、天猫上进行搜索，展示的列表中的数据来自哪里？源DB、FS吗？



■ 问题3：document会以什么结构存储？

网页会存储哪些信息？



目录

1

学习目标

2

IndexWriter详解

3

Document详解

4

索引更新



■ Document 文档

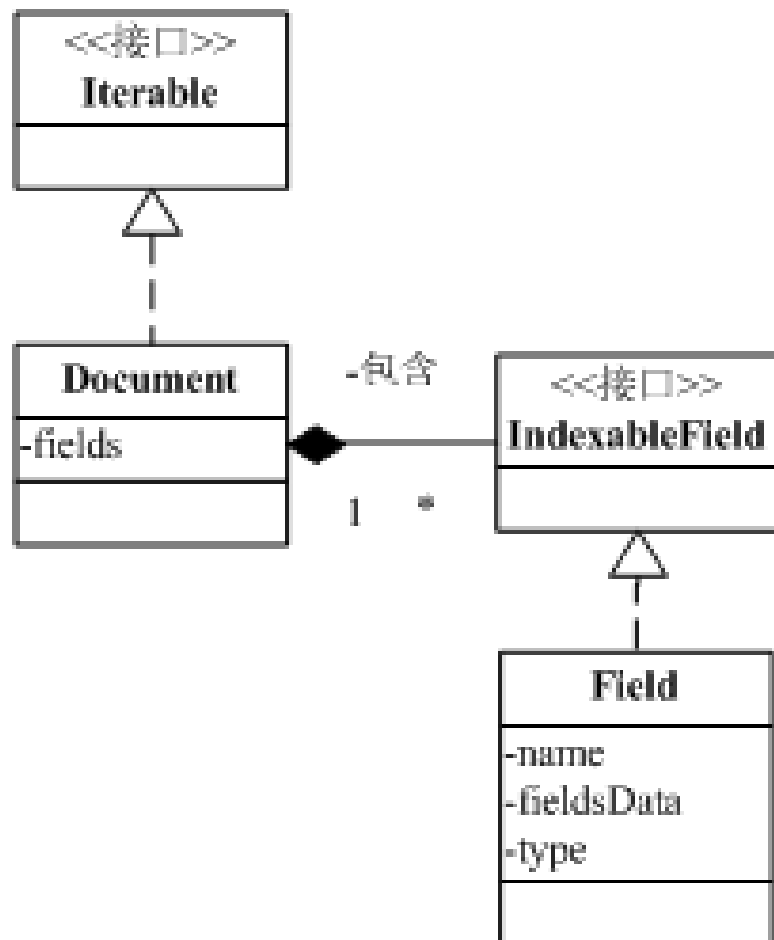
要索引的数据记录、文档在lucene中的表示，是索引、搜索的基本单元。一个Document由多个字段Field构成。就像数据库的记录-字段。

IndexWriter按加入的顺序为Document指定一个递增的id（从0开始），称为文档id。反向索引中存储的是这个id，文档存储中正向索引也是这个id。业务数据的主键id只是文档的一个字段。

请查看Document的源码，找出操作字段的API



Document API



- Document**
- fields** : List<IndexableField>
 - Document()**
 - iterator()** : Iterator<IndexableField>
 - add(IndexableField)** : void
 - removeField(String)** : void
 - removeFields(String)** : void
 - getBinaryValues(String)** : BytesRef[]
 - getBinaryValue(String)** : BytesRef
 - getField(String)** : IndexableField
 - getFields(String)** : IndexableField[]
 - getFields()** : List<IndexableField>
 - NO_STRINGS** : String
 - getValues(String)** : String[]
 - get(String)** : String
 - toString()** : String
 - clear()** : void



■ Field

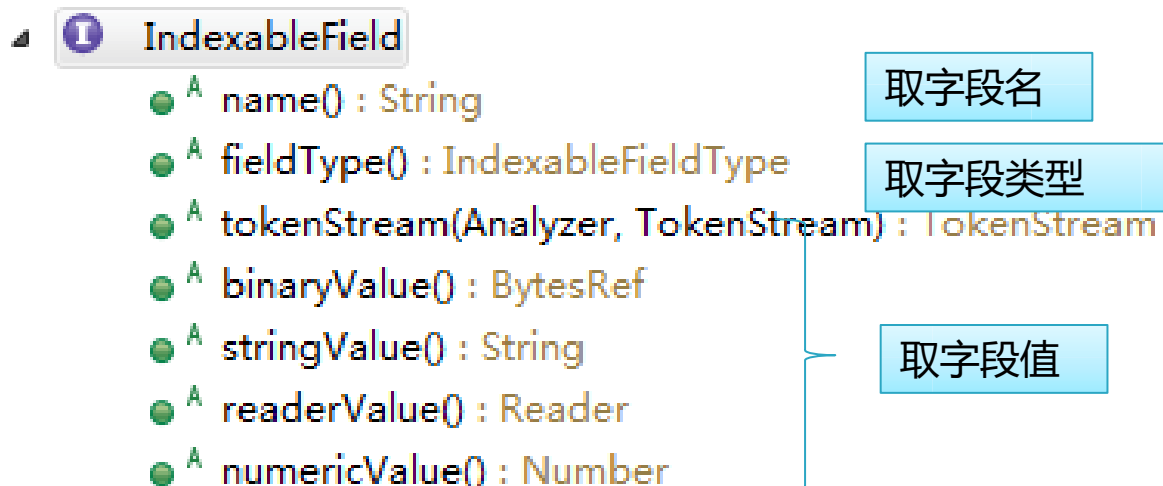
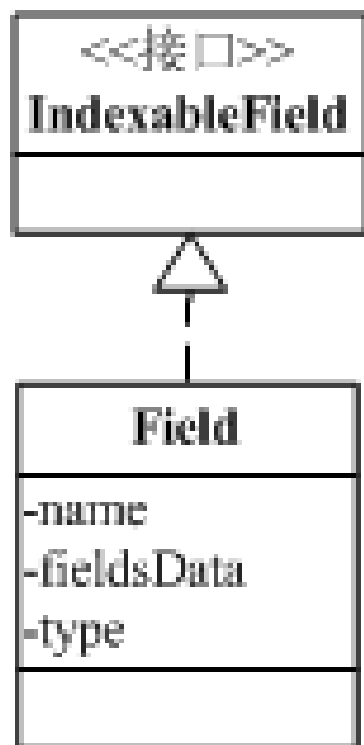
字段：由字段名name、字段值value（fieldsData）、字段类型 type 三部分构成。

字段值可以是文本（String、Reader 或 预分析的 TokenStream）、二进制值（byte[]）或数值。

请查看Field的源码，找出这三个属性
查看它提供了哪些构造方法供我们使用。



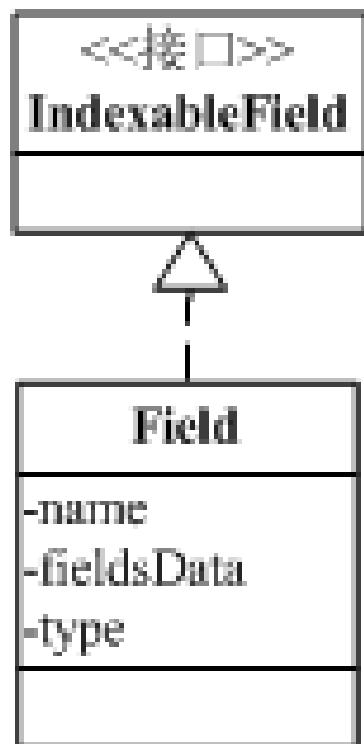
■ IndexableField Field API



Field的API见下页



IndexableField Field API



Field

Attributes:

- type : IndexableFieldType
- name : String
- fieldsData : Object
- tokenStream : TokenStream

属性

Constructors:

- Field(String, IndexableFieldType)
- Field(String, Reader, IndexableFieldType)
- Field(String, TokenStream, IndexableFieldType)
- Field(String, byte[], IndexableFieldType)
- Field(String, byte[], int, int, IndexableFieldType)
- Field(String, BytesRef, IndexableFieldType)
- Field(String, String, IndexableFieldType)

构造方法

Methods:

- stringValue() : String
- readerValue() : Reader
- tokenStreamValue() : TokenStream
- setStringValue(String) : void
- setReaderValue(Reader) : void
- setBytesValue(byte[]) : void
- setBytesValue(BytesRef) : void
- setByteValue(byte) : void
- setShortValue(short) : void
- setIntValue(int) : void
- setLongValue(long) : void
- setFloatValue(float) : void
- setDoubleValue(double) : void
- setTokenStream(TokenStream) : void
- name() : String
- numericValue() : Number
- binaryValue() : BytesRef
- toString() : String



■ Document—Field 数据举例

➤新闻：新闻id，新闻标题、新闻内容、作者、所属分类、发表时间

➤网页搜索的网页：标题、内容、链接地址

➤商品： id、名称、图片链接、类别、价格、库存、商家、品牌、月销量、详情...

问题1：我们收集数据创建document对象来为其创建索引，数据的所有属性是否都需要加入到document中？如数据库表中的数据记录的所有字段是否都需要放到document中？哪些字段应加入到document中？



问题2：是不是所有加入的字段都需要进行索引？是不是所有加入的字段都要保存到索引库中？什么样的字段该被索引？什么样的字段该被存储？

请就网页、商品进行思考？

网页：标题、内容、链接地址

商品：id、名称、图片链接、类别、价格、库存、商家、品牌、月销量、详情...



问题3：各种要被索引的字段该以什么样的方式进行索引，全都是分词进行索引，还是有不同区别？

请就网页、商品进行思考？

网页：标题、内容、链接地址

商品：id、名称、图片链接、类别、价格、库存、商家、品牌、月销量、详情...

从问题2、3得出：不同的字段会有不同的索引设置信息。

这些信息通过字段的类型属性`type:IndexableFieldType`对象来定义



■ IndexableFieldType

字段类型：描述该如何索引存储该字段

字段可选择性地保存在索引中，这样在搜索结果中，这些保存的字段值就可获得。

一个Document应该包含一个或多个存储字段来唯一标识一个文档。为什么？

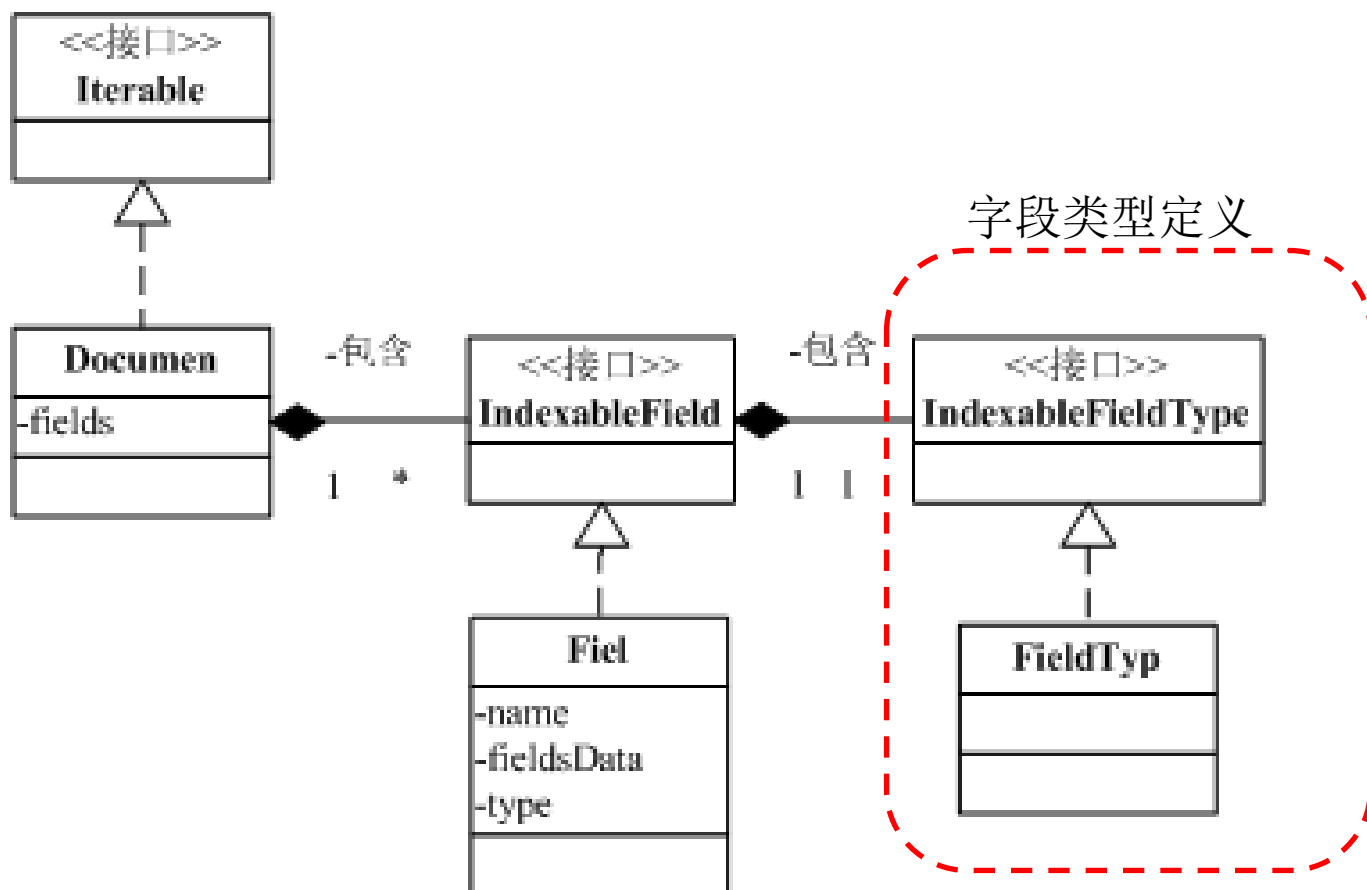
注意：未存储的字段，从索引中取得的document中是没有这些字段的。

请查看IndexableFieldType 的源码，找到存储、分词、索引信息的定义

请查看IndexableFieldType的实现类有哪些？

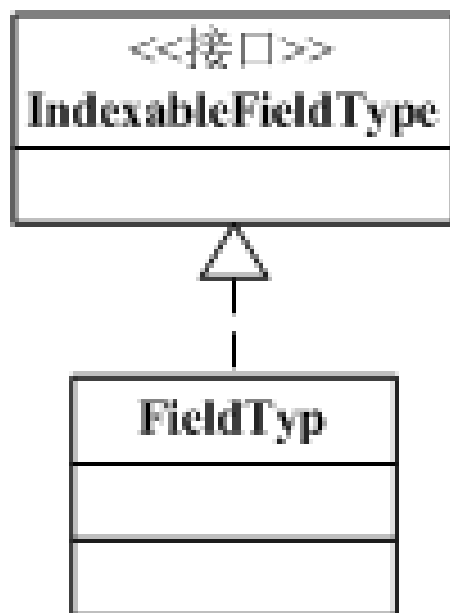


Document 类关系





■ IndexableFieldType API 说明



- I IndexableFieldType**
- ^A `stored() : boolean` 是否存储
 - ^A `tokenized() : boolean` 是否分词
 - ^A `storeTermVectors() : boolean`
 - ^A `storeTermVectorOffsets() : boolean`
 - ^A `storeTermVectorPositions() : boolean`
 - ^A `storeTermVectorPayloads() : boolean`
 - ^A `omitNorms() : boolean` 是否忽略标准化
 - ^A `indexOptions() : IndexOptions` 如何索引
 - ^A `docValuesType() : DocValuesType`
 - ^A `pointDimensionCount() : int`
 - ^A `pointNumBytes() : int`



■ IndexOptions 索引选项说明：

➤NONE

Not indexed 不索引

➤DOCS

反向索引中只存储了包含该词的 文档id，没有词频、位置

➤DOCS_AND_FREQS

反向索引中会存储 文档id、词频

➤DOCS_AND_FREQS_AND_POSITIONS

反向索引中存储 文档id、词频、位置

➤DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS

反向索引中存储 文档id、词频、位置、偏移量

请参照备注的代码示例，结合后两页介绍的luke工具来看不同索引选项的结果！



■ Luke 索引查看工具安装

为了查看我们的代码创建的索引情况，请安装工具 **luke**:

下载地址:


<https://github.com/DmitryKey/luke/releases>


➤当前最新版 7.2.0 可用于lucene7.3.0版

Luke 7.2.0

 mocobeta released this on 7 Jan · [2 commits](#) to master since this release

Assets




 [luke-7.2.0-luke-release.zip](#)

 [Source code \(zip\)](#)

 [Source code \(tar.gz\)](#)

Upgrade to lucene 7.2.0

开箱即用:

 **target**
 **luke.bat**
 **luke.sh**



■ luke Document 查看说明：

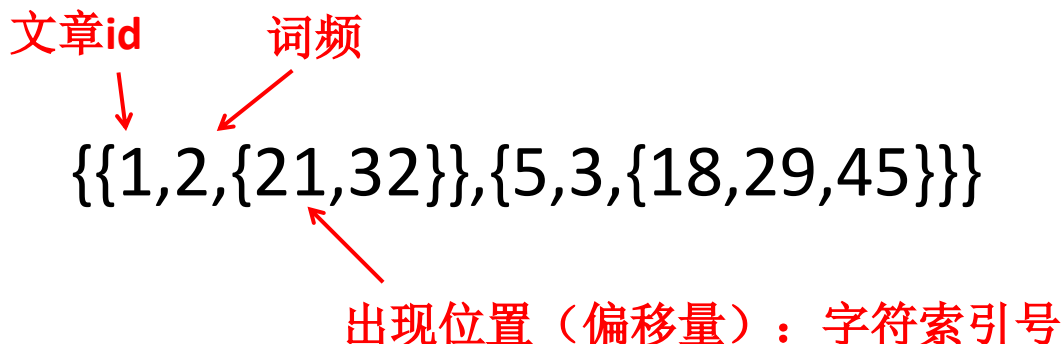
I - Indexed (docs,freqs,pos,offsets) P - Payloads S - Stored; Flags: B - Binary; N - Norms (type/precision); #bxx - Numeric (type/precision); Dbxxxx - DocValues (type/precision); Tx/x - PointValues (numBytes/dimension);		
SVBN#bxxDbxxxxTx/x	Norm	Value
-S-----	---	http://www.dongnao.com/aaa
-S--N-----	11	ThinkPad X1 Carbon 20KH0009CD/25CD 超极本轻薄笔记本电脑联想
-#i32DnumberT4/1	---	999900
-S-----	---	p0001
-----Dsorted----	---	<not present or not stored>
SV-N-----	6	集成显卡 英特尔 酷睿 i5-8250U 14英寸



问题4：如果要在搜索结果中做关键字高亮，需要什么信息？如果要实现短语查询、临近查询（跨度查询），需要什么信息？

如 要搜索包含“张三” “李四”，且两词之间跨度不超过5个字符。

问题5：位置、偏移数据在反向索引中占的存储量占比大不大？





问题6：如果某个字段不需要进行短语查询、临近查询，那么在反向索引中就不需要保存位置、偏移数据。这样是不是可以降低反向索引的数据量，提升效率？但是如果该字段要做高亮显示支持，该怎么办？。

为了提升反向索引的效率，这样的字段的位置、偏移数据是不应该保存到反向索引中的。这也你前面看到 IndexOptions为什么有那些选项的原因。

在lucene4.0以前，反向索引中总会存储这些数据，4.0后改进为可选择的。

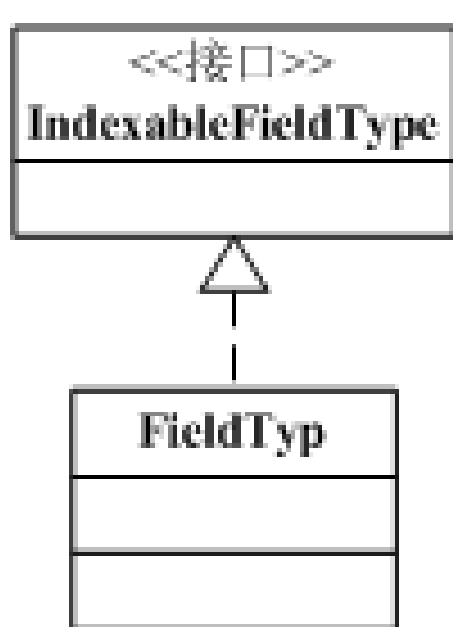
那对于只做高亮显示（或得到搜索结果后需要使用这些信息）的字段怎么办？

一个字段分词器分词后，每个词项会得到一系列属性信息，如 出现频率、位置、偏移量等，这些信息构成一个词项向量 **termVectors**

请查看IndexableFieldType、FieldType中有没有设置保存termVectors的方法。



■ IndexableFieldType API 说明



I IndexableFieldType

● ^A stored() : boolean

是否存储

● ^A tokenized() : boolean

是否分词

● ^A storeTermVectors() : boolean

是否存储词项向量

● ^A storeTermVectorOffsets() : boolean

词项向量中是否存储偏移

● ^A storeTermVectorPositions() : boolean

词项向量中是否存储位置

● ^A storeTermVectorPayloads() : boolean

词项向量中是否存储负载

● ^A omitNorms() : boolean

是否忽略标准化

● ^A indexOptions() : IndexOptions

如何索引

● ^A docValuesType() : DocValuesType

● ^A pointDimensionCount() : int

● ^A pointNumBytes() : int



■ storeTermVectors

对于不需要在搜索反向索引时用到，但在搜索结果处理时需要的位置、偏移量、附加数据(payload) 的字段，我们可以单独为该字段存储（文档id→词项向量）的正向索引。

- `boolean storeTermVectors()` 是否存储词项向量
- `boolean storeTermVectorPositions()` 是否在词项向量中存储位置
- `boolean storeTermVectorOffsets()` 是否在词项向量中存储偏移量
- `boolean storeTermVectorPayloads()` 是否在词项向量中存储附加信息

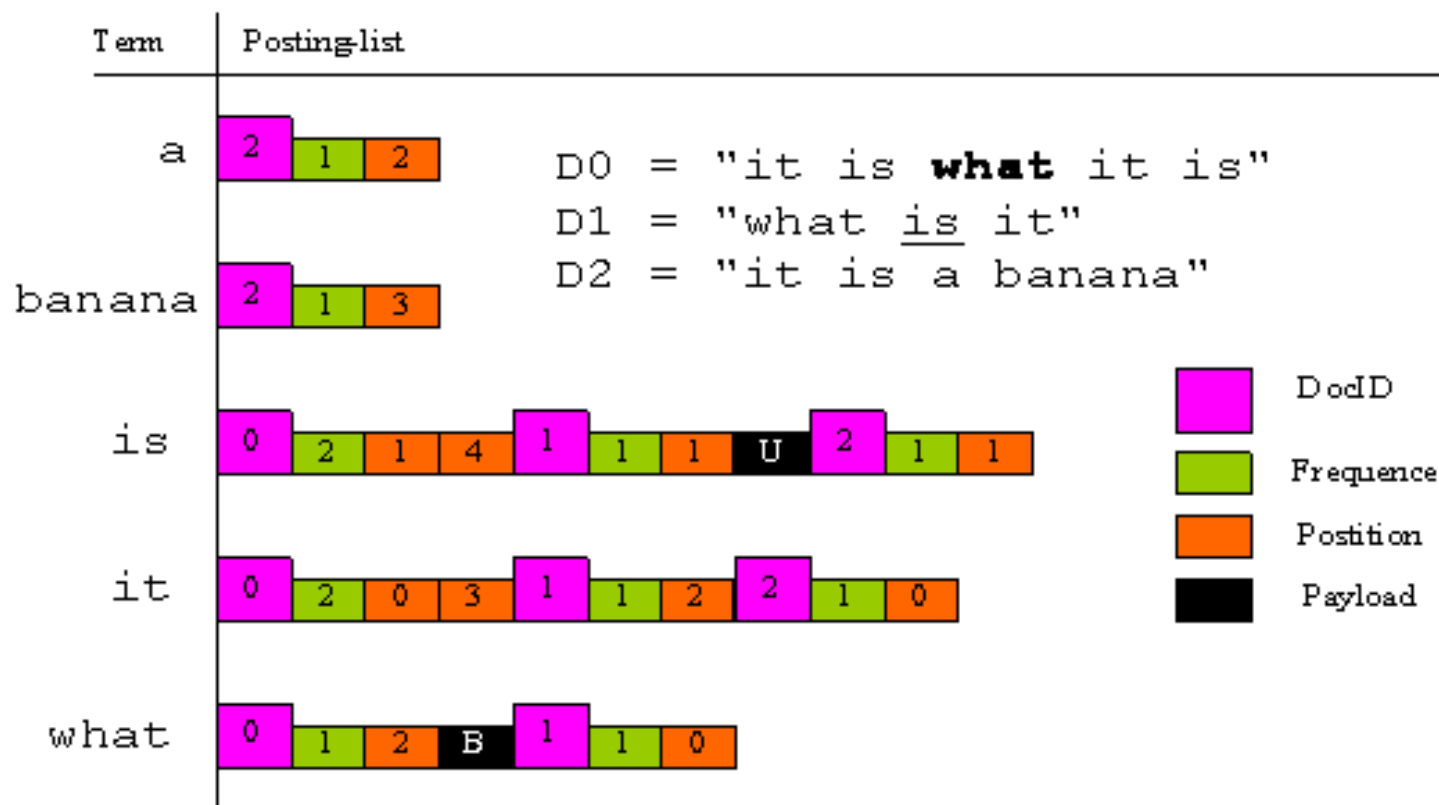
FieldType实现类中有对应的set方法

动手：在前面的示例代码基础上增加词项向量的存储，看看索引结果。
示例代码，请参考 `IndexTermVectorsDemo`

概念说明：Token trem 词条： 分词得到的词项



■ 什么是附加信息Payloads



附加信息非常有用，可用它来存储特殊信息，及减少词项数等



■ 练习1

请为商品记录建立索引，字段信息如下：

➤商品id：字符串，不索引、但存储

```
String prodId = "p0001";
```

➤商品名称：字符串，分词索引(存储词频、位置、偏移量)、存储

```
String name = "ThinkPad X1 Carbon 20KH0009CD/25CD 超极本轻薄笔记本电脑";
```

➤图片链接：仅存储

```
String imgUrl = "http://www.dongnao.com/aaa";
```

➤商品简介：字符串，分词索引（不需要支持短语、临近查询）、存储，结果中支持高亮显示

```
String simpleIntro = "集成显卡 英特尔 酷睿 i5-8250U 14英寸";
```

➤品牌：字符串，不分词索引，存储

```
String brand = "ThinkPad";
```

代码示例：ProductIndexExercise1



问题7：我们往往需要对搜索的结果支持按不同的字段进行排序，如商品搜索结果按价格排序、按销量排序等。以及对搜索结果进行按某字段分组统计，如按品牌统计。

假如我们按关键字“娃娃”搜索后得到相关的文档id列表

{10,21,18,48,29,.....}

要对它们进行按价格排序

有的人想看销量排序

有时需要按品牌统计数量...

反向索引对排序有用吗？

需得到每个id对应的价格或销售是多少、品牌是什么，再进行排序、统计。

这个价格、销量、品牌数据在哪里？

如果搜到的文档列表量很大，排序会有什么问题没？



■ 空间换时间

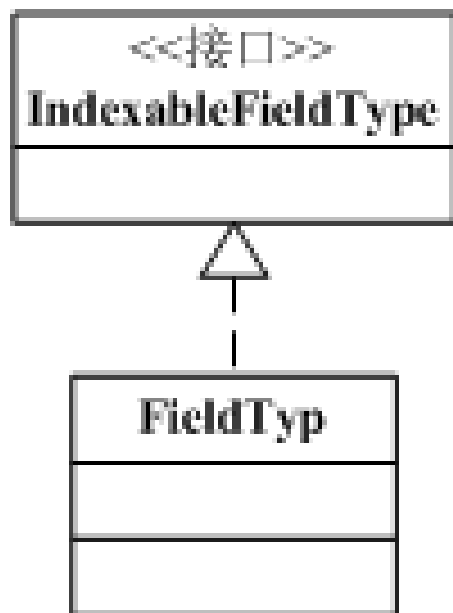
对这种需要排序、分组、聚合的字段，为其建立独立的文档->字段值的正向索引、列式存储。这样我们要加载搜中文档的这个字段的数据就快很多，耗内存少。

■ docValueType

IndexableFieldType 中的 docValueType方法 就是让你来为需要排序、分组、聚合的字段指定如何为该字段创建文档->字段值的正向索引的。



■ IndexableFieldType API 说明



I IndexableFieldType

● ^A stored() : boolean

是否存储

● ^A tokenized() : boolean

是否分词

● ^A storeTermVectors() : boolean

是否存储词项向量

● ^A storeTermVectorOffsets() : boolean

词项向量中是否存储偏移

● ^A storeTermVectorPositions() : boolean

词项向量中是否存储位置

● ^A storeTermVectorPayloads() : boolean

词项向量中是否存储负载

● ^A omitNorms() : boolean

是否忽略标准化

● ^A indexOptions() : IndexOptions

如何索引

● ^A docValuesType() : DocValuesType

指定如何为该字段创建正向索引

● ^A pointDimensionCount() : int

● ^A pointNumBytes() : int



■ DocValueType 选项说明

➤NONE 不开启docvalue

➤NUMERIC 单值、数值字段，用这个

➤BINARY 单值、字节数组字段用

➤SORTED 单值、字符字段用，会预先对值字节进行排序、去重存储

➤SORTED_NUMERIC 单值、数值数组字段用，会预先对数值数组进行排序

➤SORTED_SET 多值字段用，会预先对值字节进行排序、去重存储

**DocValueType是强类型要求的：
字段的值必须保证同类型。**

具体使用选择：

➤字符串+单值 会选择SORTED作为docvalue存储

➤字符串+多值 会选择SORTED_SET作为docvalue存储

➤数值或日期或枚举字段+单值 会选择NUMERIC 作为docvalue存储

➤数值或日期或枚举字段+多值 会选择SORTED_SET作为docvalue存储

强调：需要排序、分组、聚合、分类查询（面查询）的字段才创建docValues



■ 练习2

➤1、修改品牌字段：支持统计查询

➤2、增加商品类别字段：字符串（类别名），索引不分词，不存储、支持分类统计，多值（一个商品可能属于多个类别）。

`type = {"电脑","笔记本电脑"}`

多值如何加入到document?

同字段多次加入

➤3、增加价格字段：整数，单位分，不索引、存储，需要支持排序



■ 如何加入数值字段

请查看Field类中提供了对应的构造方法或其他方法没？

Field的构造方法和set方法：

- Field
 - Field(String, Reader, IndexableFieldType)
 - Field(String, TokenStream, IndexableFieldType)
 - Field(String, byte[], IndexableFieldType)
 - Field(String, byte[], int, int, IndexableFieldType)
 - Field(String, BytesRef, IndexableFieldType)
 - Field(String, String, IndexableFieldType)
 - setStringValue(String) : void
 - setReaderValue(Reader) : void
 - setBytesValue(byte[]) : void
 - setBytesValue(BytesRef) : void
 - setByteValue(byte) : void
 - setShortValue(short) : void
 - setIntValue(int) : void
 - setLongValue(long) : void
 - setFloatValue(float) : void
 - setDoubleValue(double) : void
 - setTokenStream(TokenStream) : void

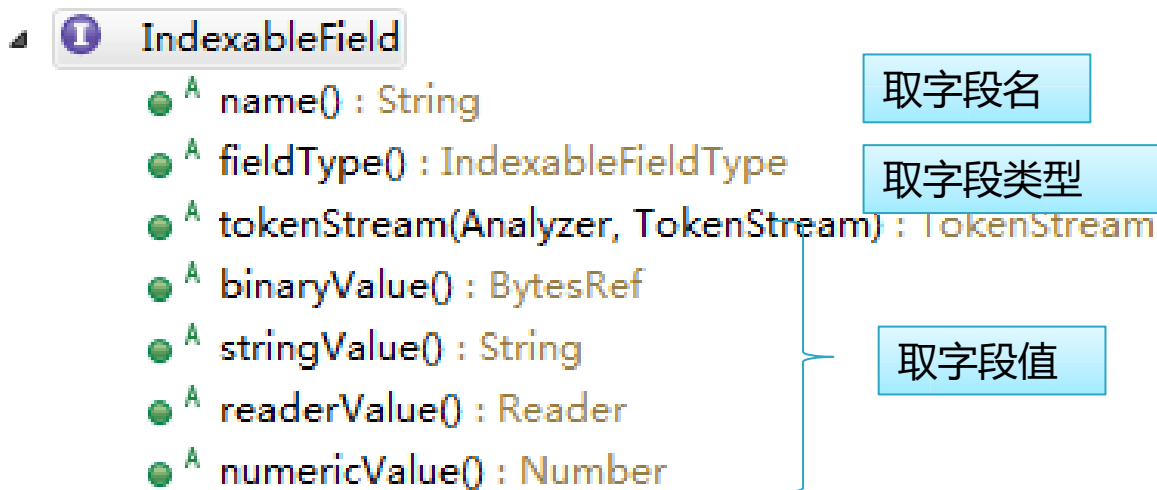
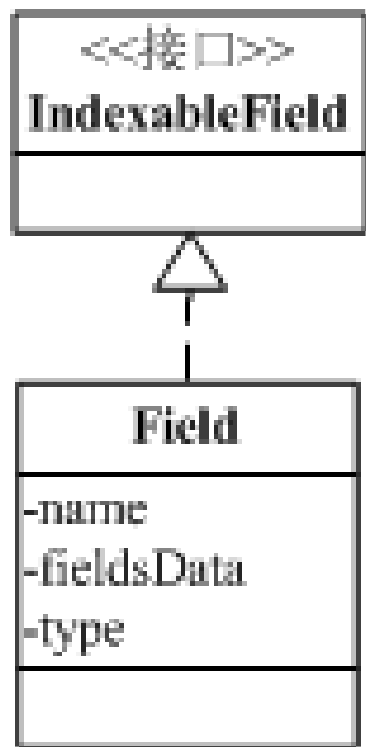
构造方法没有对应的，set方法倒是有，请看下setIntValue方法的源码，看它是如何将字段值设为一个整数值的，你有疑惑吗？ 再看下其他的set方法。

只有一种解释：数值字段需要扩展Field 。该如何扩展呢？ 先看后页



■ 如何加入数值字段

请再查看 IndexableField 的API：



请再查看**Field**类中对应的实现

你是否发现它们的实现和那些**setValue**方法的逻辑是一样的。

疑问1：它们为什么返回**null**，而不是进行转换后返回？

疑问2：为什么有这五个取值方法？你是否有答案？



■ 回顾前面 Field 的定义

字段：由字段名name、字段值value、字段类型 type 三部分构成。

字段值可以是文本（String、Reader 或 预分析的 TokenStream）、二进制值（byte[]）或数值。

这两个疑问你解开了吗？

➤五个方法对应五种不同的值类型

➤返回null表示不是该类型的值

➤还有一点待发现总结

不然，IndexWriter在索引字段时，如何知道字段的值类型。但在IndexWriter中它是以怎样的顺序来进行判断的呢？

查看 IndexWriter在索引、存储字段、为字段建立docValues正向索引时的代码，看看它是如何使用这些方法的，找出答案。



- **问1：反向索引时是如何使用这五个方法的？**
- **问2：存储时是如何使用这五个方法的？**
- **问3：docValues时是如何使用这五个方法的？**
- **问4：如何正确加入价格字段？**



■ 如何加入数值字段

补充疑惑答案一点：

- 五个方法对应五种不同的值类型
- 返回null表示不是该类型的值
- 即使不是binary类型值，也要提供正确的binaryValue()值获取

加入数值字段方式：

- 扩展Field，提供构造方法传入数值类型值，赋给字段值字段；
- 改写binaryValue() 方法，返回数值的字节引用。

代码参考：MyIntField



问题8： IndexableFieldType中最后定义的pointDimensionCount(), pointNumBytes() 是做什么用的？

Lucene6以后引入了点的概念来表示数值字段，废除了原来的IntField等。在Point字段类中提供了精确、范围查询的便捷方法。

注意：只是引入点的概念，并未改变数值字段的本质。

既然是点，就有空间概念：维度。一维：一个值，二维：两个值的；

pointDimensionCount() 返回点的维数

pointNumBytes() 返回点中数值类型的字节数。



■ Lucene预定义的字种子类，你可灵活选用

➤ **TextField**: Reader or String indexed for full-text search

➤ **StringField**: String indexed verbatim as a single token

➤ **IntPoint**: int indexed for exact/range queries.

➤ **LongPoint**: long indexed for exact/range queries.

➤ **FloatPoint**: float indexed for exact/range queries.

➤ **DoublePoint**: double indexed for exact/range queries.

➤ **SortedDocValuesField**: byte[] indexed column-wise for sorting/faceting

➤ **SortedSetDocValuesField**: SortedSet<byte[]> indexed column-wise for
sorting/faceting

➤ **NumericDocValuesField**: long indexed column-wise for sorting/faceting

➤ **SortedNumericDocValuesField**: SortedSet<long> indexed column-wise for
sorting/faceting

➤ **StoredField**: Stored-only value for retrieving in summary results

请仔细看它们的源码是怎么设置字段的值、类别的。

注意：这里没有设置存储词项向量的。

如果单个子类不满足需要，可多个组合。请参考 示例代码：**IndexWriteDemo**

如果组合不了，就直接用**Field + FieldType**

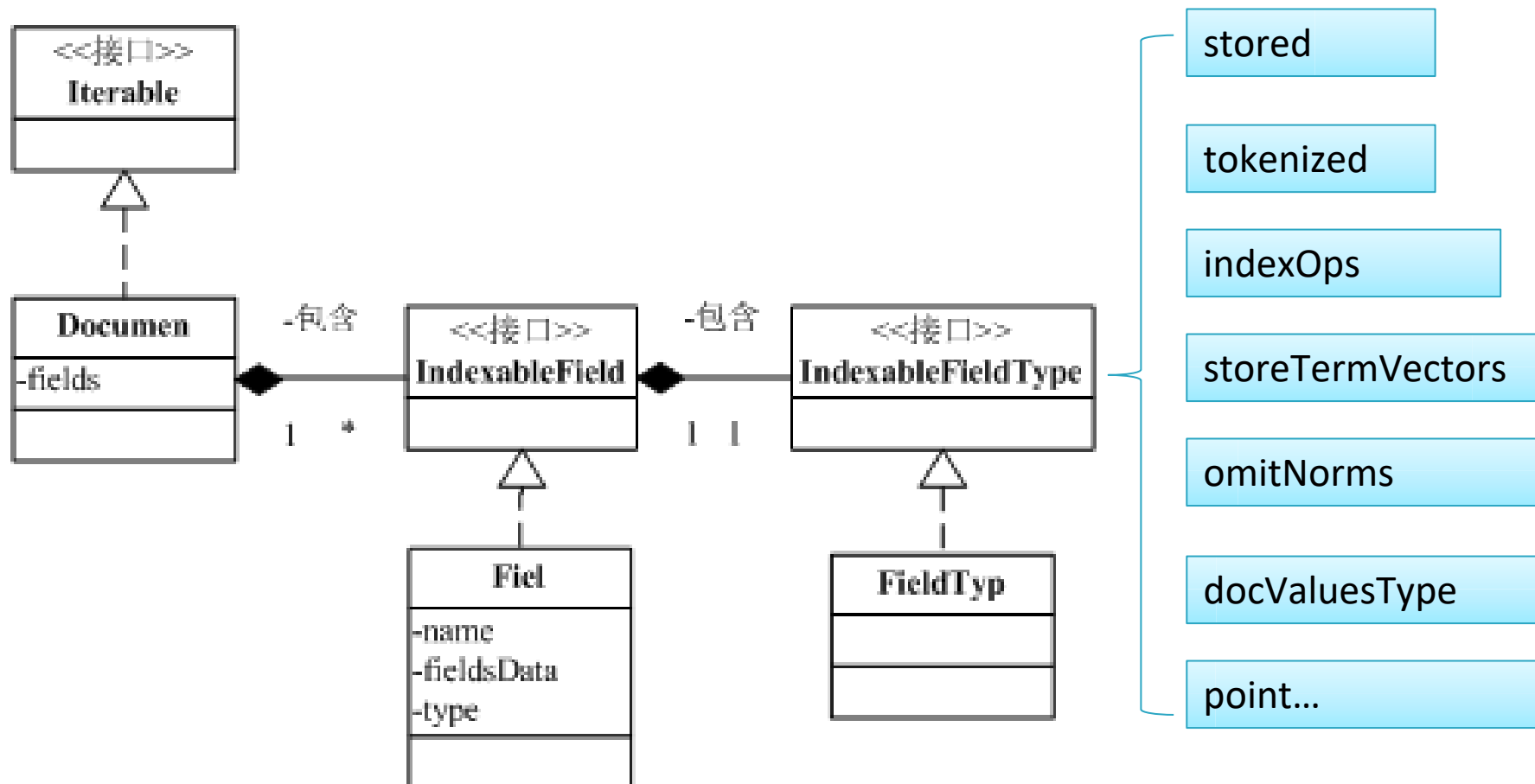


问题9：Field中提供那么多的setXXValue()方法，是什么意图？

问题10：加入索引时，每个数据记录需要都创建一个Document吗？



■ 小结





目录

1

学习目标

2

IndexWriter详解

3

Document详解

4

索引更新



■ IndexWriter 索引更新 API

- `deleteDocuments(Term...) : long`
- `deleteDocuments(Query...) : long`
- `updateDocument(Term, Iterable<? extends IndexableField>) : long`
- `updateNumericDocValue(Term, String, long) : long`
- `updateBinaryDocValue(Term, String, BytesRef) : long`
- `updateDocValues(Term, Field...) : long`

■ 说明：

➤Term 词项 指定字段的词项

➤删除流程：根据Term、Query找到相关的文档id、同时删除索引信息，再根据文档id删除对应的文档存储。

➤更新流程：先删除、再加入新的doc

➤注意：只可根据索引的字段进行更新。

示例代码：IndexUpdateDemo