

# Estructura de Datos y Algoritmos

ITBA 2024-Q2

# Algoritmos para texto

El algoritmo KMP busca en forma eficiente la aparición exacta de un query en un texto, tomando ventaja del procesamiento del “query”.

Pero si quisiéramos buscar un query en un “grupo de documentos”, nos conviene generar un índice de dicha colección para luego buscar a través del índice.

# Algoritmos para texto

Es decir, preprocesar los documentos sobre los que se va a realizar la búsqueda (el corpus). Los términos que integran al corpus se denomina Vocabulario.

Eso es lo que hacen las bibliotecas de **Fulltext Retrieval**.

Un buscador como Google/Bing generan un índice de los documentos para facilitar la búsqueda. La búsqueda se realiza a través del índice.

Existe una biblioteca de código abierto para esto:  
Lucene

Está escrita en Java (<https://lucene.apache.org/>) => escrita por Douglass Cutting en 1999, en su primera versión. Doug es uno de los fundadores de Apache Hadoop.

Nos permite armar nuestro propio “search engine”!!!  
(Ej: el buscador de Amazon vs el buscador de Jumbo)

## ¿Qué es un índice? ¿Qué tipo de índice usa Lucene?

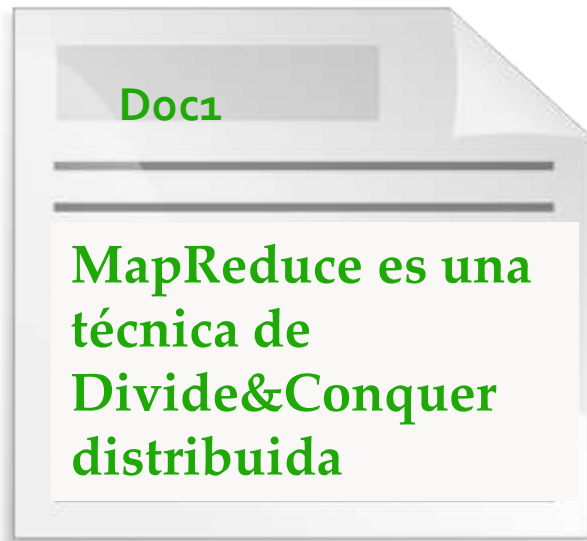
Un índice es una estructura que permite llegar rápidamente al dato buscado. Si se agrega/elimina/actualiza un documento en la colección debe actualizarse. Su ventaja está en la búsqueda.



¿Qué índice resulta conveniente? ¿Cuál es el que usa Lucene?

Lucene usa un **Archivo Invertido**: “conjunto de términos que dicen a qué documento pertenece”

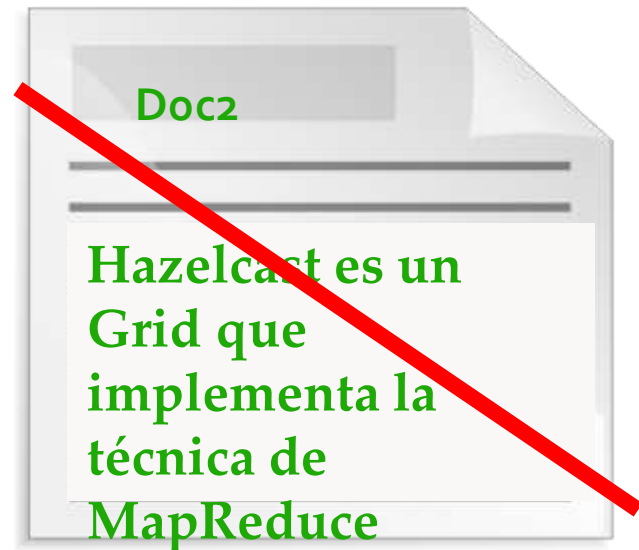
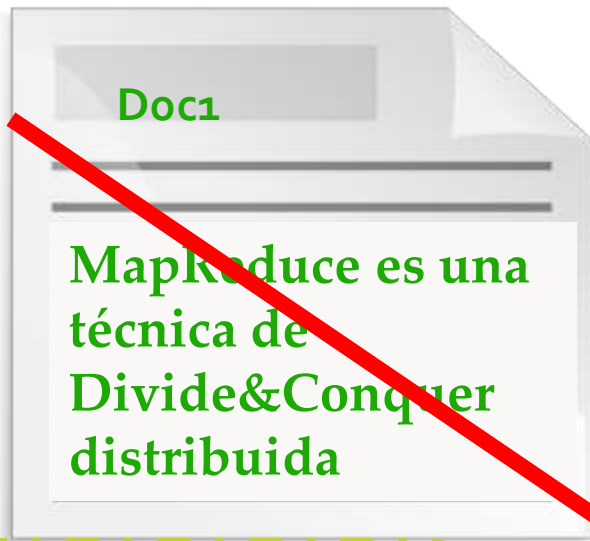
Es un mapping: **término → documento.**



*MapReduce => { Doc1, Doc2 }*  
*es => {Doc1, Doc2 }*

...  
*técnica => {Doc1, Doc2}*  
*Grid => {Doc2 }*

...



MapReduce => { Doc1, Doc2 }  
es => {Doc1, Doc2 }

...  
técnica => {Doc1, Doc2}  
Grid => {Doc2 }

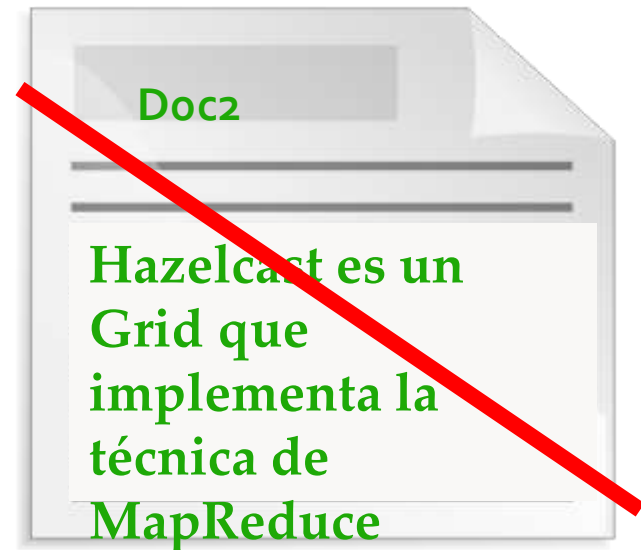
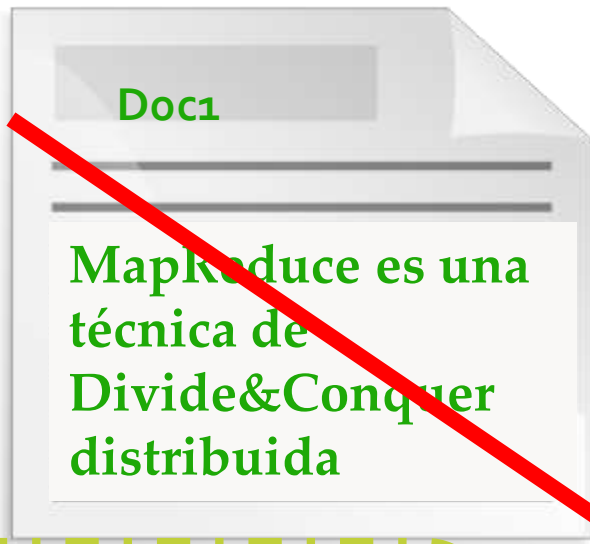
...  
*¿Cómo se usa el índice para  
responder las consultas?*



"MapReduce"?

Rta:  
{Doc1, Doc2}





MapReduce  $\Rightarrow \{ \text{Doc1}, \text{Doc2} \}$

Es  $\Rightarrow \{ \text{Doc1}, \text{Doc2} \}$

...

Técnica  $\Rightarrow \{ \text{Doc1}, \text{Doc2} \}$

Grid  $\Rightarrow \{ \text{Doc2} \}$

...

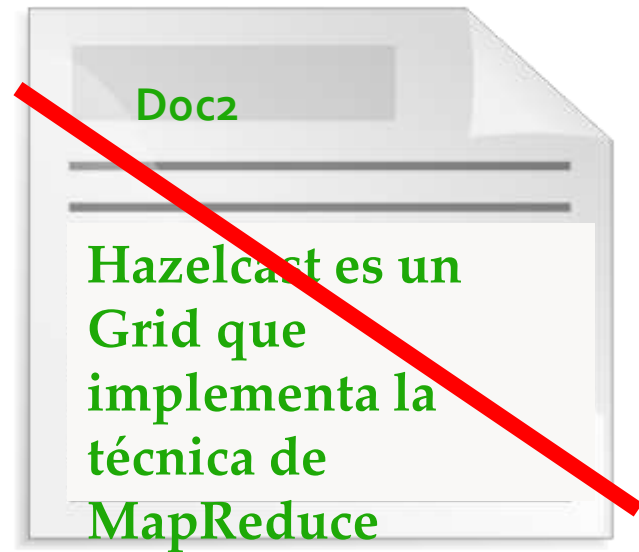
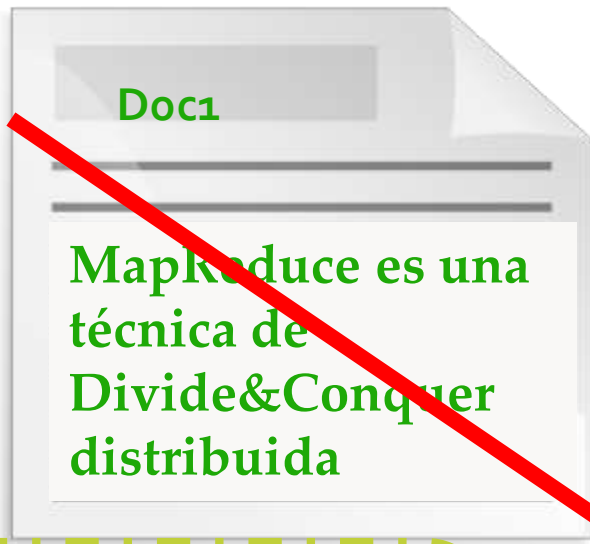
"MapReduce" && "Grid"?

¿Cómo se usa el índice para  
responder las consultas?



Rta:

$\{ \text{Doc1}, \text{Doc2} \} \cap \{ \text{Doc2} \}$   
o sea, sólo  $\{ \text{Doc2} \}$



MapReduce => { Doc1, Doc2 }

Es => {Doc1, Doc2 }

...

Técnica => {Doc1, Doc2}

Grid => {Doc2 }

...

"MapReduce" - "Grid"?

¿Cómo se usa el índice para  
responder las consultas?



Rta:

{Doc1, Doc2} - { Doc2}  
o sea, sólo {Doc1}

# Lucene

- Concepto de documento, campos.
- Almacenamiento en Lucene: en el índice y fuera del índice
- Aplicaciones
  - IndexBuilder (creación de los documentos)
  - TheSearcher (búsqueda de documentos)
- Queries:
  - API
  - QueryBuilder
- Formas de separar en tokens
- Ranking de documentos

# Lucene

- Concepto de documento, campos.
- Almacenamiento en Lucene: en el índice y fuera del índice
- Aplicaciones
  - IndexBuilder (creación de los documentos)
  - TheSearcher (búsqueda de documentos)
- Queries:
  - API
  - QueryBuilder
- Formas de separar en tokens
- Ranking de documentos

## Documento en Lucene (definición)

Un *Documento Lucene* es una secuencia de *Campos (fields)*. Cuando se ingresa un documento en Lucene, automáticamente se le asociará un ID (docid).

## Campo/Field en Lucene (definición)

Un *Campo Lucene* es un par *nombre y secuencia de 1 o más términos*.  
Ej: creo un field que se llama "author" y contiene "Leticia Gomez". Según las configuraciones que use podré hacer que "Leticia Gomez" sea un único término (token) o dos términos (2 tokens)

## Término / Term (definición)

Un *Término Lucene* es una secuencia de bytes (podrían interpretarse como String, números, etc) asociada a cierto campo. Dos secuencias de bytes con igual contenido pero asociadas a 2 campos diferentes se consideran diferentes.

# Lucene

- *Concepto de documento, campos.*
- *Almacenamiento en Lucene: en el índice y fuera del índice*
- Aplicaciones
  - IndexBuilder (creación de los documentos)
  - TheSearcher (búsqueda de documentos)
- Query:
  - API
  - QueryBuilder
- Formas de separar en tokens
- Ranking de documentos

# Creación de un Campo

Al crear un campo (nombre y términos) y asociarlo a cierto documento, puedo optar por:

- Almacenarlo en el **índice** o en el **archivo invertido**. Eso significa que se puede **buscar** o **no se lo busca**.  
El **índice** almacena los términos y los separa en palabras. El **archivo invertido** almacena los documentos y los separa en palabras. Los signos de puntuación se eliminan y los términos se convierten en minúsculas.  
El **índice** **está indexado** y el **archivo invertido** **no está indexado**.  
Un field puede ser solo **almacenable**, solo **indexable** o **ambas cosas**.  
Obligatorio que sea, por lo menos, alguna de ellas => sino, para qué está Lucene
- Indexarlo en el **archivo invertido** o **no**. Eso significa que **tokenizado** o **no** (según la configuración).  
El **archivo invertido** **tokeniza** los términos y los separa en palabras. Los signos de puntuación se eliminan y los términos se convierten en minúsculas.  
El **archivo invertido** **tokenizado** y sus términos (tokens) **participan de las búsquedas**.



Como crear un Field?

```
FieldType aField = new FieldType();
```

- Sobre su almacenamiento literal, fuera del archivo invertido

```
aField.setStored(  );
```

true: se almacena

false: no se almacena

- Sobre su indexación en el archivo invertido (alguna de estas)

`aField.setIndexOptions(●);`

**IndexOptions.NONE** : no se indiza

**IndexOptions.DOCS**: se indiza cada término solo con los doc ids en los que participa

**IndexOptions.DOCS\_AND\_FREQS**: se indiza cada término con los doc ids en los que participa y frecuencia en ellos

**IndexOptions.DOCS\_AND\_FREQS\_AND\_POSITIONS**: se indiza cada término con los doc ids en los que participa, junto con frecuencia y posiciones ordinales dentro de ellos

**IndexOptions.DOCS\_AND\_FREQS\_AND\_POSITIONS\_AND\_OFFSETS**: se indiza cada término con los doc ids en los que participa, frecuencia en ellos, posiciones ordinales y offsets dentro de ellos

Salvo que use IndexOptions.NONE, el archivo invertido está preparado para recibir la siguiente información que se completará según las opciones elegidas

Por cada campo tenemos una tabla ordenada ascendentemente por valor del término (token).

Hay tantas tablas como Fields indexados hayamos creado.

Por eso la búsqueda se realiza por campos.

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] :[startOffsets - endOffsets in docid) ]

### *Ejemplo 1:*

Creo un documento con un único field llamado “**content**”. No lo almaceno fuera del índice invertido, pero sí lo indizo sólo a nivel **IndexOptions.DOCS** y tokenizo

```
// itero creando documentos que van a Lucene
for( ... ) {
    Document aDoc = new Document();

    String text= “bla bla bla”;

    FieldType fieldD = new FieldType();
    fieldD.setStored(false);
    fieldD.setIndexOptions(IndexOptions.DOCS);

    aDoc.add(new Field("content", text, fieldD ) );
    ....
}
```

De toda esta información se va a completar solo esto:

Value term	Freq en docs	[ docid: <del>freqs in docid</del> <del>positions in docid</del> <del>:startOffsets - endOffsets in docid</del> ) ]

De acuerdo a lo anterior, tengo una colección de 4 documentos formados por solo ese field "content". Lo que se indiza es precisamente el contenido de los archivos a.txt, b.txt, c.txt, d.txt

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

Como es el índice invertido?

Docid 0 (a.txt)

store,, game

Value term	Freq en docs	[ docid]

Docid 1 (b.txt)

video

Value term	Freq en docs	[ docid ]				
game	1	<table><tr><td>0</td><td></td><td></td><td></td></tr></table>	0			
0						
store	1	<table><tr><td>0</td><td></td><td></td><td></td></tr></table>	0			
0						



Docid 3 (d.txt)

Game video,  
review game.

Value term	Freq en docs	[ docid]
game	1 2	0
store	1	0
video	1 2	1

Docid 2 (c.txt)

game

Value term	Freq en docs	[ docid ]			
game	2 <div>3</div>				
		0			
		3			
review	1	3			
store	1	0			
video	2	1			
		3			

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

Es decir, en forma compacta el archivo invertido para el campo “content” tiene:

Value term	Freq en docs	[ docid ]												
game	3	<table><tr><td>0</td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td></tr></table>	0				3				2			
0														
3														
2														
review	1	<table><tr><td>3</td><td></td><td></td><td></td></tr></table>	3											
3														
store	1	<table><tr><td>0</td><td></td><td></td><td></td></tr></table>	0											
0														
video	2	<table><tr><td>1</td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td></tr></table>	1				3							
1														
3														

### *Ejemplo 2:*

Creo un documento con un único field llamado “**content**”. No lo almaceno fuera del índice invertido, pero si lo indizo solo a nivel **IndexOptions.DOCS\_AND\_FREQS\_AND\_POSITIONS** y tokenizo

```
// itero creando documentos que van a Lucene
```

```
for( ... ) {
```

```
    Document aDoc = new Document();
```

```
    String text= “bla bla bla”;
```

```
    FieldType fieldD = new FieldType();
```

```
    fieldD.setStored(false);
```

```
    fieldD.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS);
```

```
    aDoc.add(new Field("content", text, fieldD ) );
```

```
    ....
```

De toda esta información se va a completar solo esto:

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] <del>:[startOffsets - endOffsets in docid) ]</del>

De acuerdo a lo anterior, tengo una colección de 4 documentos formados por solo ese field “content”. Lo que se indiza es precisamente el contenido de los archivos a.txt, b.txt, c.txt, d.txt

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

Como es el índice invertido?

Docid 0 (a.txt)

store,, game

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] ]

Docid 1 (b.txt)

video

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] ]				
game	1	<table><tr><td>0</td><td>1</td><td>[1]</td><td></td></tr></table>	0	1	[1]	
0	1	[1]				
store	1	<table><tr><td>0</td><td>1</td><td>[0]</td><td></td></tr></table>	0	1	[0]	
0	1	[0]				



Docid 3 (d.txt)

Game video,  
review game.

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] ]			
game	1 2	0	1	[1]	
store	1	0	1	[0]	
video	1 2	1	1	[0]	

Docid 2 (c.txt)

game

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] ]			
game	2 / 3	0	1	[1]	
		3	2	[0, 3]	
review	1	3	1	[2]	
store	1	0	1	[0]	
video	2	1	1	[0]	
		3	1	[1]	

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

Es decir, en forma compacta el archivo invertido para el campo “content” tiene:

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] ]			
game	3	0	1	[1]	
		3	2	[0, 3]	
		2	1	[0]	
review	1	3	1	[2]	
store	1	0	1	[0]	
video	2	1	1	[0]	
		3	1	[1]	

### *Ejemplo 3:*

Si lo indizo a nivel **IndexOptions.DocsAndFreqsAndPositionsAndOffsets** y tokenizo con los mismos documentos, cómo hubiera quedado el índice archivo invertido?

La lista de offsets de cada término tiene el formato de intervalo [startOffset, endOffset)  
¿NO se puede calcular conociendo el término (y su longitud) y el ordinal que ocupa??

Rta: no

Solo mostramos como queda el token game. (Completen Uds. Todo lo demás)

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

s t o r e , , g a m e

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid]:[startOffsets - endOffsets in docid] ]			
game	3	0	1	[1]	
		3	2	[0, 3]	
		2	1	[0]	
Etc etc etc					

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

G	a	m	e		v	i	d	e	o	,		\r	\n
r	e	v	i	e	w				g	a	m	e	.

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid]:[startOffsets - endOffsets in docid] ]			
game	3	0	1	[1]	[ [8-12) ]
		3	2	[0, 3]	
		2	1	[0]	
Etc etc etc					

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

g a m e

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid]: <b>startOffsets - endOffsets in docid</b> ]			
game	3	0	1	[1]	[ [8-12) ]
		3	2	[0, 3]	[ [0-4), [23-27) ]
		2	1	[0]	
Etc etc etc					

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid]: <b>[startOffsets - endOffsets in docid]</b> ]			
game	3	0	1	[1]	[ [8-12) ]
		3	2	[0, 3]	[ [0-4), [23-27) ]
		2	1	[0]	[ [0, 4) ]
Etc etc etc					



#### *Ejemplo 4:*

¿ Qué se hubiera guardado en Lucene si hubiéramos solicitado (más allá del índice y sus configuración) : **aField.setStored( yes );** ?

```
// itero creando documentos que van a Lucene
for( ... ) {
    Document aDoc = new Document();

    String text= "bla bla bla";

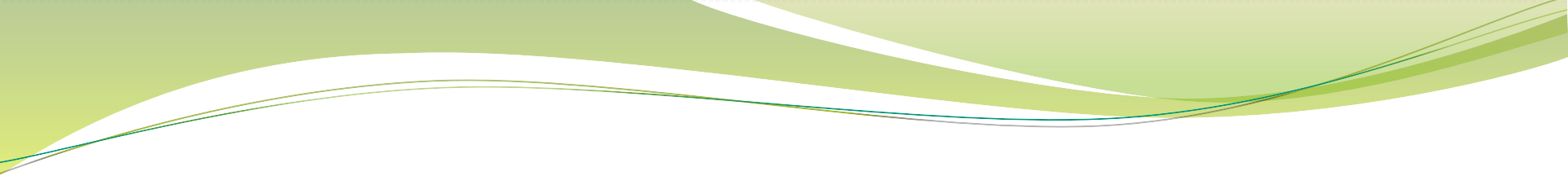
    FieldType fieldD = new FieldType();
    fieldD.setStored(false);
    fieldD.setIndexOptions(IndexOptions.DOCS);
    aField.setStored( yes );
    aDoc.add(new Field("content", text, fieldD ) );
    ....
}
```

Rta: fuera del archivo invertido se almacena literal la información en formato tradicional (no invertido)

Docid 0 (a.txt)

store,, game

Doc Id	Field	
0	content	store,, game
Etc etc etc		



Resumiendo: la base de la indexación corresponde a definir fields con ciertas características => configurable

Lucene viene equipado con un Field muy especial que corresponden a cierta configuración/combinación muy usada. Muchas veces es suficiente y en vez de generar un field y asociarles todas sus características podemos usarlo (como si fueran un shortcut a cierta configuración).

¿Cuál es el field predefinido?

- **TextField:** Maneja tipo de datos texto. Se indexa y por default se tokenizada (se pasa a minúscula, se separa por signos de puntuación)=> Es como usar  
IndexOptions.DOCS\_AND\_FREQS\_POSITIONS

Como muchas veces este tipo de campos no se lee desde un string (de pocos caracteres) sino que se lo indiza desde un “stream” (archivo grande), Lucene solo ofrece: almacenarlo él por fuera del índice si no proviene de un stream. Caso contrario estaría triplicado el espacio: en el stream (file system), en el índice (para buscar) y además otra vez literal fuera del archivo invertido. Demasiado!

Es decir, la opción de almacenar solo se podrá hacer si no viene de un stream.

Usar un TextField es como usar todo esto:

```
FieldType fieldD = new FieldType();  
fieldD.setStored( ?? ); // DEPENDE DE DONDE VENGA LA FUENTE  
fieldD.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS); // SIN OFFSET  
aDoc.add(new Field("content", text, fieldD ) );
```

Como setStored(false)

*Pero solo una sentencia creo esa configuración*

```
String text="bla bla bla";
```

```
aDoc.add(new TextField("content", text, Field.Store.NO ) );
```

*O bien*

```
aDoc.add(new TextField("content", text, Field.Store.YES ) );
```

*O bien si tengo un reader sobre un archivo:*

```
aDoc.add(new TextField("content", aReader) );
```

Como setStored(true)

### *Ejemplo 5:*

¿ Qué se hubiera guardado en Lucene si hubiéramos solicitado el siguiente código completando text desde los 4 archivos anteriores?

```
// itero creando documentos que van a Lucene
for( ... ) {
    Document aDoc = new Document();

    String text= "...";

    aDoc.add(new TextField("content", text, Field.Store.NO ) );
    ....
}
```

Docid 0 (a.txt)

store,, game

Docid 1 (b.txt)

video

Docid 3 (d.txt)

Game video,  
review game.

Docid 2 (c.txt)

game

Es decir, en forma compacta el archivo invertido para el campo “content” tiene:

Value term	Freq en docs	[ docid:freqs in docid:[positions in docid] ]			
game	3	0	1	[1]	
		3	2	[0, 3]	
		2	1	[0]	
review	1	3	1	[2]	
store	1	0	1	[0]	
video	2	1	1	[0]	
		3	1	[1]	

# Lucene

- *Concepto de documento, campos.*
- *Almacenamiento en Lucene: en el índice y fuera del índice*
- *Aplicaciones*
  - IndexBuilder (creación de los documentos)
  - TheSearcher (búsqueda de documentos)
- Query:
  - API
  - QueryBuilder
- Formas de separar en tokens
- Ranking de documentos





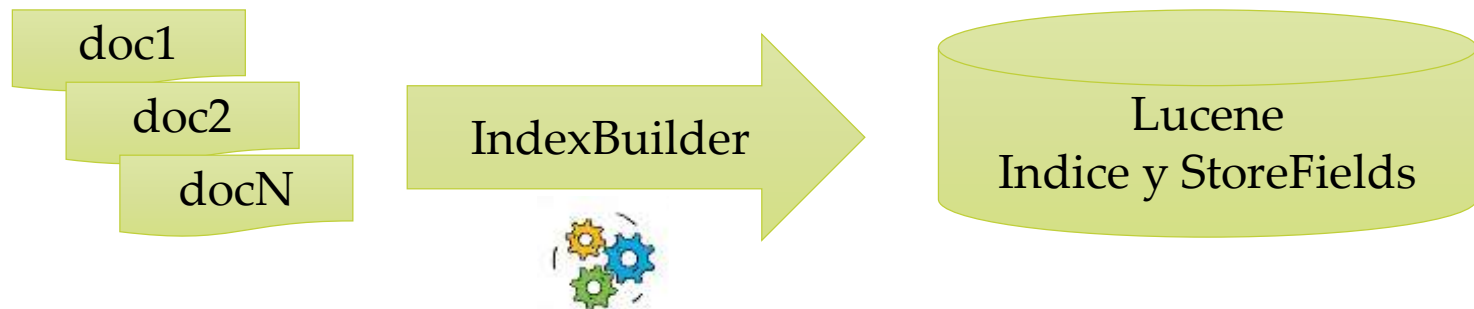
Las 2 aplicaciones independientes que precisamos realizar con Lucene son: IndexerBuilder y Searcher

# Lucene

- *Concepto de documento, campos.*
- *Almacenamiento en Lucene: en el índice y fuera del índice*
- *Aplicaciones*
  - IndexBuilder (creación de los documentos)
  - TheSearcher (búsqueda de documentos)
- Query:
  - API
  - QueryBuilder
- Formas de separar en tokens
- Ranking de documentos

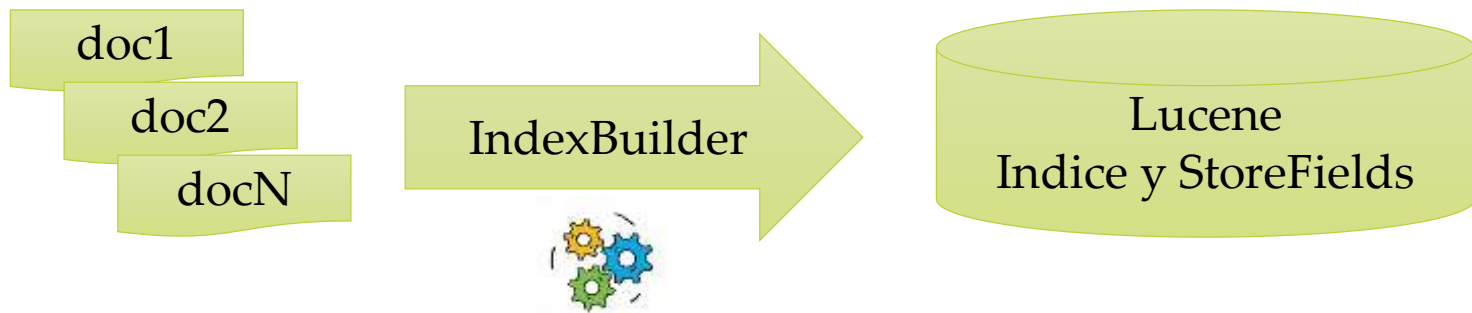
## 1. IndexBuilder

Aplicación que se encarga de generar el índice a partir de un conjunto de documentos Lucene y lo deja almacenado en cierto directorio que le indiquemos para tal efecto.



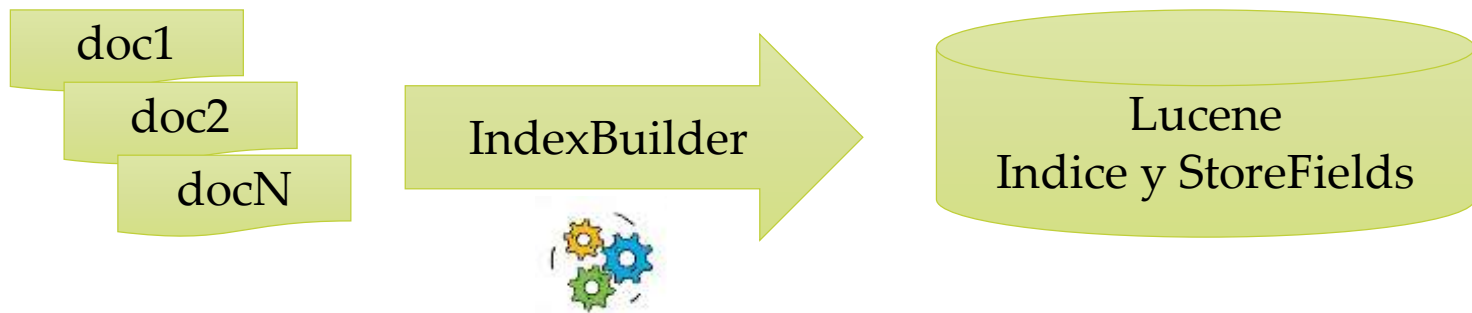
## 1. IndexBuilder (continuación)

Típicamente, algunos de los datos de los documentos se toman de algún archivo de disco (txt, pdf, doc), aunque podrían crearse from scratch (desde un String). Si seguimos esa estrategia, le debemos indicar uno/varios directorio/s dónde residen los documentos físicos.



## 1. IndexBuilder (continuación)

Es nuestra responsabilidad mantener el índice actualizado, es decir, re ejecutar si queremos agregar documentos, o re generar el índice si los documentos en disco fueron modificados.



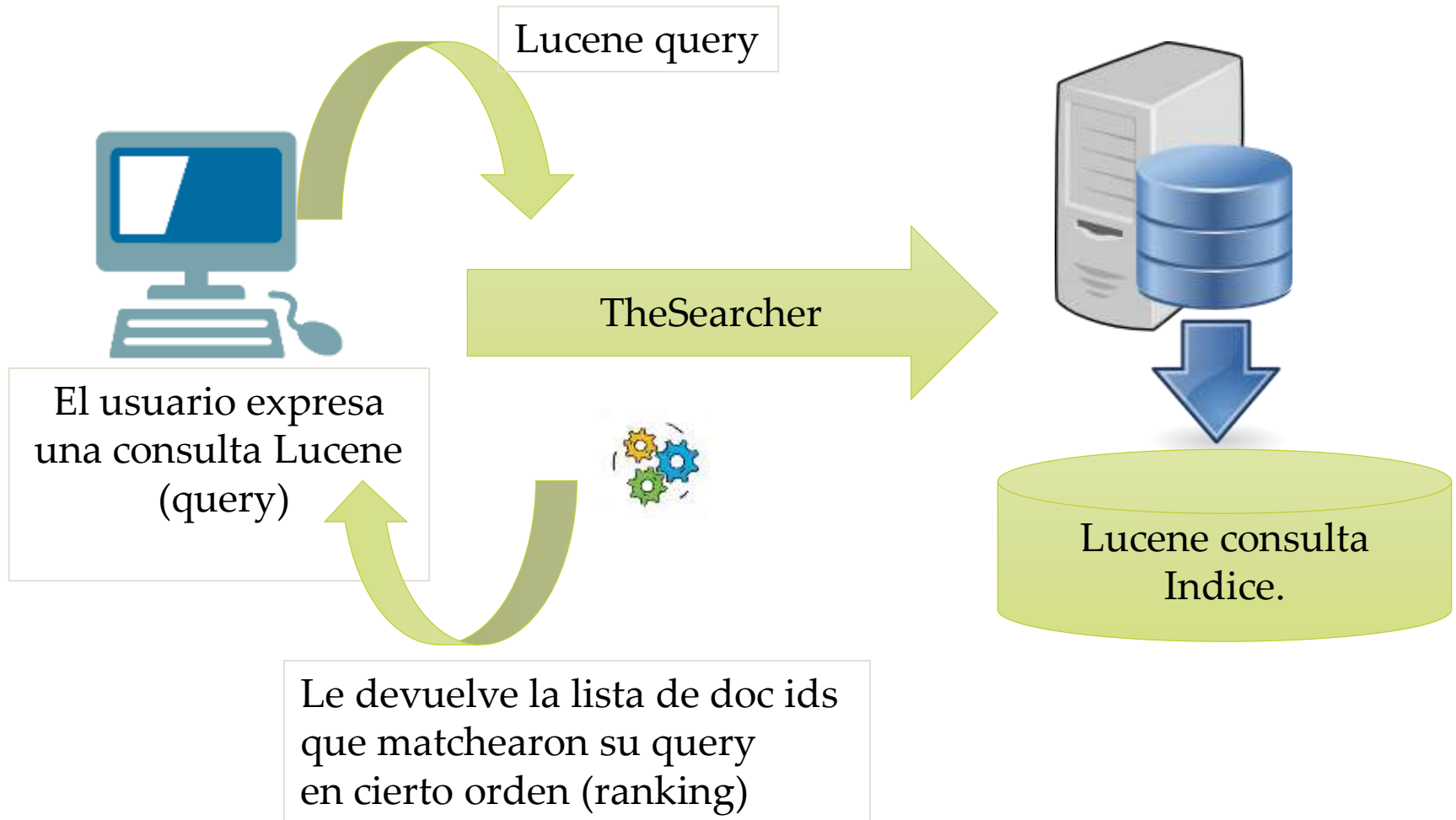
# Lucene

- *Concepto de documento, campos.*
- *Almacenamiento en Lucene: en el índice y fuera del índice*
- *Aplicaciones*
  - *IndexBuilder (creación de los documentos)*
  - *TheSearcher (búsqueda de documentos)*
- Query:
  - API
  - QueryBuilder
- Formas de separar en tokens
- Ranking de documentos

## 2. TheSearcher

Aplicación que se encarga de aceptar consultas y utiliza el índice construido para retornar los documentos (ids) que “matchearon” la consulta rankeados en cierto orden.

# Proceso de búsqueda





# Proceso de búsqueda mejorado

