# Inteligência evolutiva

Marcelo de Rezende Martins
Marcio Vinicius dos Santos

Busca é

Ubíquo

Busca é

Um negócio de bilhões de doláres

# Fundamentos da Busca

Dada uma consulta, busca é o processo de retornar documentos relevantes para a consulta.

# Fundamentos da Busca

Dada uma consulta, busca
é o processo de retornar
documentos relevantes
para a consulta.

# Relevância

Não existe uma fórmula
ou regra para relevância.

# O Apache Lucene

É uma uma biblioteca de buscas de alta performance em Java

# O Lucene

Possui dois serviços principais

# 1- Serviço de Indexação Assíncrona

Responsável por criar um índice de busca, através da indexação de documentos.

# 11- serviço de busca

Através de uma consulta,
retorna os documentos
relevantes a busca.

# Um exemplo de uso

```
public static void main(String [] args) throws Exception {

    BlogSearchExample bs = new BlogSearchExample();
    String tag = "collective intelligence";
    String luceneIndexPath = "blogSearchIndex";
    BlogQueryResult blogQueryResult = bs.getBlogsFromTechnorati(tag);
    Directory indexDirectory = bs.createSearchIndex(luceneIndexPath,
blogQueryResult);
    bs.searchForBlogs(indexDirectory, tag);

}
```

Pegando informações do blog

Indexando Entradas do Blog

Buscando por entradas no blog

# Indexando Entradas do Blog

```
    public Directory createSearchIndex(String path, BlogQueryResult
blogQueryResult) throws Exception{

        IndexWriter indexWriter = new IndexWriter(path, getAnalyzer(), true);

        indexBlogEntries(indexWriter, blogQueryResult.getRelevantBlogs());

        indexWriter.optimize();

        indexWriter.close();

        return indexWriter.getDirectory();
    }
```

carrega o analisador de sinônimos

Responsável por criar e atualizar o índice de busca

Adiciona no índice as novas entradas

Atualiza o índice com as novas entradas

Salva as atualizacões

Responsável por retornar objetos que manipula o índice de busca

# Adicionando novas linguagens no índice

```
private void indexBlogEntries(IndexWriter indexWriter, List<RetrievedBlogEntry> blogEntries) throws Exception {
    for (RetrievedBlogEntry blogEntry: blogEntries) {
        indexWriter.addDocument( getDocument(blogEntry));
    }
}

private Document getDocument(RetrievedBlogEntry blogEntry) {
    Document document = new Document();
    BlogDataSetCreatorImpl dataSetCreator = new BlogDataSetCreatorImpl();
    String completeText = dataSetCreator.composeTextForAnalysis(blogEntry);

    addField(document,"completeText",completeText, Field.Store.YES,Field.Index.TOKENIZED , Field.TermVector.YES);
    addField(document,"name",blogEntry.getName(), Field.Store.YES,Field.Index.TOKENIZED , Field.TermVector.YES);
    addField(document,"title",blogEntry.getTitle(), ...
    addField(document,"excerpt",blogEntry.getExcerpt(), ...
    addField(document,"url",blogEntry.getUrl(), ...
    addField(document,"author",blogEntry.getAuthor(),...
    return document;
}

private void addField(Document document, String fieldName,String value,
    Field.Store fieldStore, Field.Index fieldIndex,Field.TermVector fieldTermVector) {
    Field field = new Field(fieldName, getNotNullValue(value), fieldStore, fieldIndex, fieldTermVector);
    document.add(field);
}
```

Cada field armazenado no documento pode ser recuperado na busca(Se não for tokenizado)

# Buscando por entradas no blog

```
public void searchForBlogs(Directory directory, String queryString) {

    IndexSearcher indexSearcher = new IndexSearcher(directory);

    QueryParser queryParser = new QueryParser("completeText",getAnalyzer());

    Query query = queryParser.parse(queryString);

    Hits hits = indexSearcher.search(query);

    Iterator iterator = hits.iterator();
    while (iterator.hasNext()) {
        Hit hit = (Hit) iterator.next();
        Document document = hit.getDocument();
        System.out.println(document.get("completeText"));
    }
        indexSearcher.close();
}
```

Responsável por busca no índice

O parser otimizará a query para o field ”completeText” do Document

Transforma a query de busca num objeto de busca

Busca no índice usando a query o documentos relevantes

# Indexação Incremental

```
    public void updateIndex(Directory indexDirectory, List<Term> deletionTerms,
List<Document> addDocuments){

        IndexReader indexReader = IndexReader.open(indexDirectory);

        for (Term deletionTerm: deletionTerms) {
            indexReader.deleteDocuments(deletionTerm);
        }
        indexReader.close();
       IndexWriter iWriter = new IndexWriter(indexDirectory,getAnalyzer(),false);
        for (Document document: addDocuments) {
            iWriter.addDocument(document);
        }
        iWriter.optimize();
        iWriter.close();

        }
```

Responsável por deleção no índice

Um termo é um dupla contendo o field e o conteúdo do field a ser deletado

Salva Atualizações

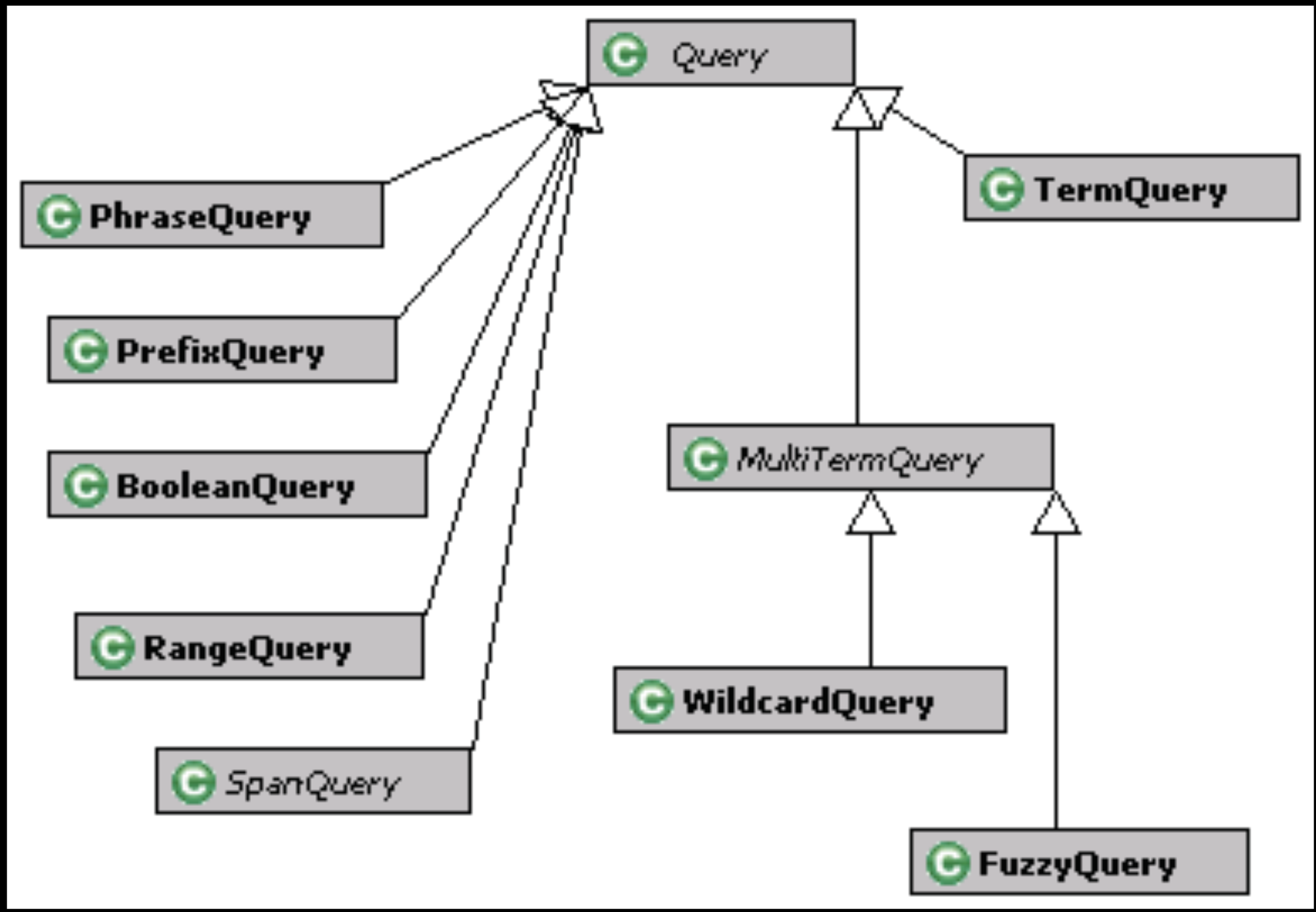
Atualiza Índex

Salva Atualizações

# Indexação Incremental

É recomendado sempre que possível recalcular o índice todo ao invés de atualizá-lo.

# Buscando no Lucene



# Buscando no Lucene

```
public Hits illustrateQueryCombination(Directory indexDirectory){
    IndexSearcher indexSearcher = new IndexSearcher(indexDirectory);

  PhraseQuery phraseQuery = new PhraseQuery();
    phraseQuery.add(new Term("completeText","collective"));
    phraseQuery.add(new Term("completeText","intelligence"));

  PrefixQuery prefixQuery = new PrefixQuery( new Term("completeText","web"));

  BooleanQuery booleanQuery = new BooleanQuery();
    booleanQuery.add(phraseQuery, BooleanClause.Occur.MUST);
    booleanQuery.add(prefixQuery, BooleanClause.Occur.SHOULD);

  return indexSearcher.search(booleanQuery);
}
```

Consulta por frases

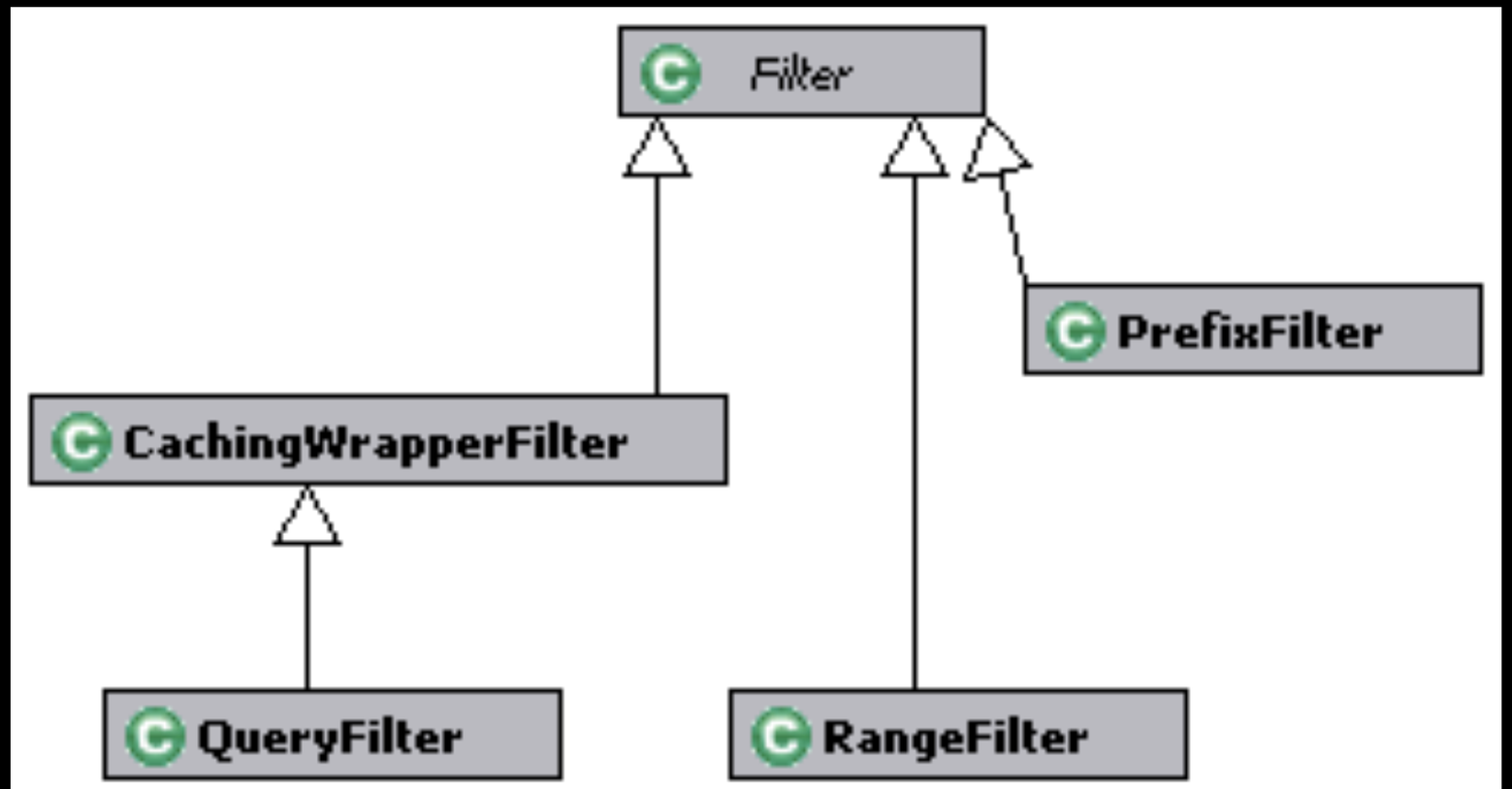
Consulta por prefixos

Conjunção de consultas

# Buscando em multiplos campos

```
    public Hits getMultiFieldAndQuery(String queryString, IndexSearcher
indexSearcher){

        String [] fields = {"name", "title", "excerpt"};
        BooleanClause.Occur[] flags = {
                BooleanClause.Occur.SHOULD,
                BooleanClause.Occur.MUST,
                BooleanClause.Occur.MUST_NOT
            };
         Query query = MultiFieldQueryParser.parse(queryString, fields,
flags, getAnalyzer());

        return indexSearcher.search(query);
         }
```

# Filtros


Filter
PrefixFilter
CachingWrapperFilter
QueryFilter
RangeFilter

# Filtrando resultados

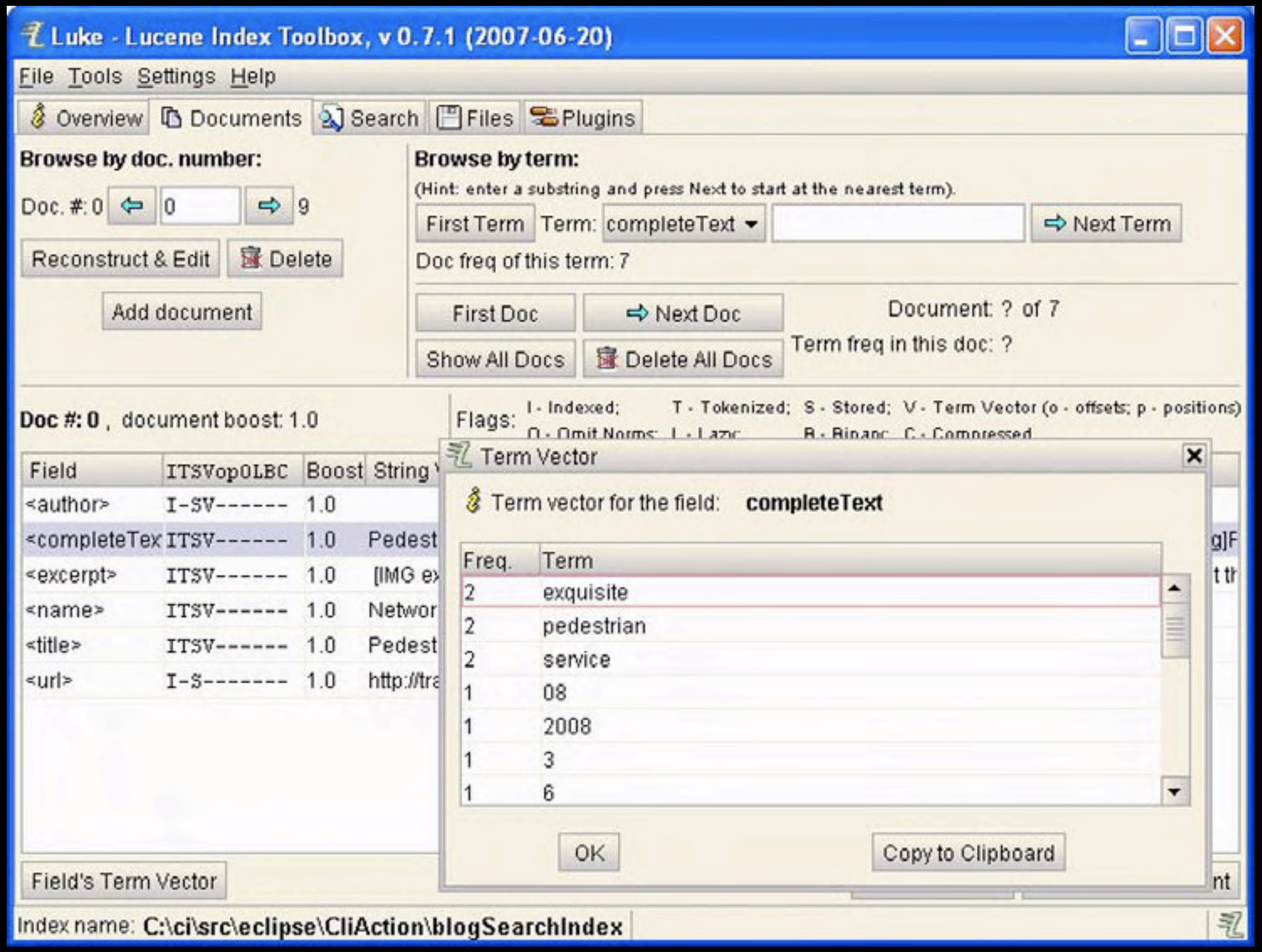
```
    public void illustrateFilterSearch(IndexSearcher indexSearcher,
Query query, Sort sort){

        Filter rangeFilter =
        new RangeFilter("modifiedDate", "20080101","20080131", true, true);

        CachingWrapperFilter cachedFilter =
        new CachingWrapperFilter(rangeFilter);

        Hits hits = indexSearcher.search(query, cachedFilter, sort);
    }
```

Filtra consultas por range

Põe em cache resultados da filtragem

# Ferramentas e frameworks Úteis

# Luke


Luke - Lucene Index Toolbox, v 0.7.1 (2007-06-20)
File Tools Settings Help
Overview Documents Search Files Plugins
Browse by doc. number:
Doc. #: 0 0 9
Reconstruct & Edit Delete
Add document
Browse by term:
(Hint: enter a substring and press Next to start at the nearest term).
First Term Term: completeText Next Term
Doc freq of this term: 7
First Doc Next Doc
Show All Docs Delete All Docs
Document: ? of 7
Term freq in this doc: ?
Doc #: 0 , document boost: 1.0
Flags: I - Indexed; T - Tokenized; S - Stored; V - Term Vector (o - offsets; p - positions)
Field ITSVopOLBC Boost String
<author> I-SV------ 1.0
<completeTex ITSV------ 1.0 Pedest
<excerpt> ITSV------ 1.0 [IMG ex
<name> ITSV------ 1.0 Networ
<title> ITSV------ 1.0 Pedest
<url> I-S------- 1.0 http://tra
Field's Term Vector
Term Vector
Term vector for the field: completeText
Freq. Term
2 exquisite
2 pedestrian
2 service
1 08
1 2008
1 3
1 6
OK
Copy to Clipboard
Index name: C:\ci\src\eclipse\CliAction\blogSearchIndex

# Solr

Servidor de busca que fornece API's JSON e XML/HTTP para o acesso, usa como motor o Lucene.

# Hibernate Search

Indexa o banco de dados no Lucene, habilitando consultas de alta performance no bd.

# Outros tipos de busca

Busca linguistica(Semântica):

http://www.hakia.com/,

http://www.powerset.com/

http://www.powerset.com/

# Outros tipos de busca

## Busca Customizada

http://www.google.com/coop/

http://www.eurekster.com/

http://rollyo.com/

# Outros tipos de busca

Busca Customizada

Projeto Genoma

# Outros tipos de busca

Sugestões para implementação do Lucene no seu sistema

-Criar indices
-Analisar tabelas e desnormaliza-las
-Criar analisador sintático ou de sinônimos
-Definir a busca