Integrando recuperação de informação em banco de dados com Hibernate Search

Alunos

Gustavo Kendi Tsuji

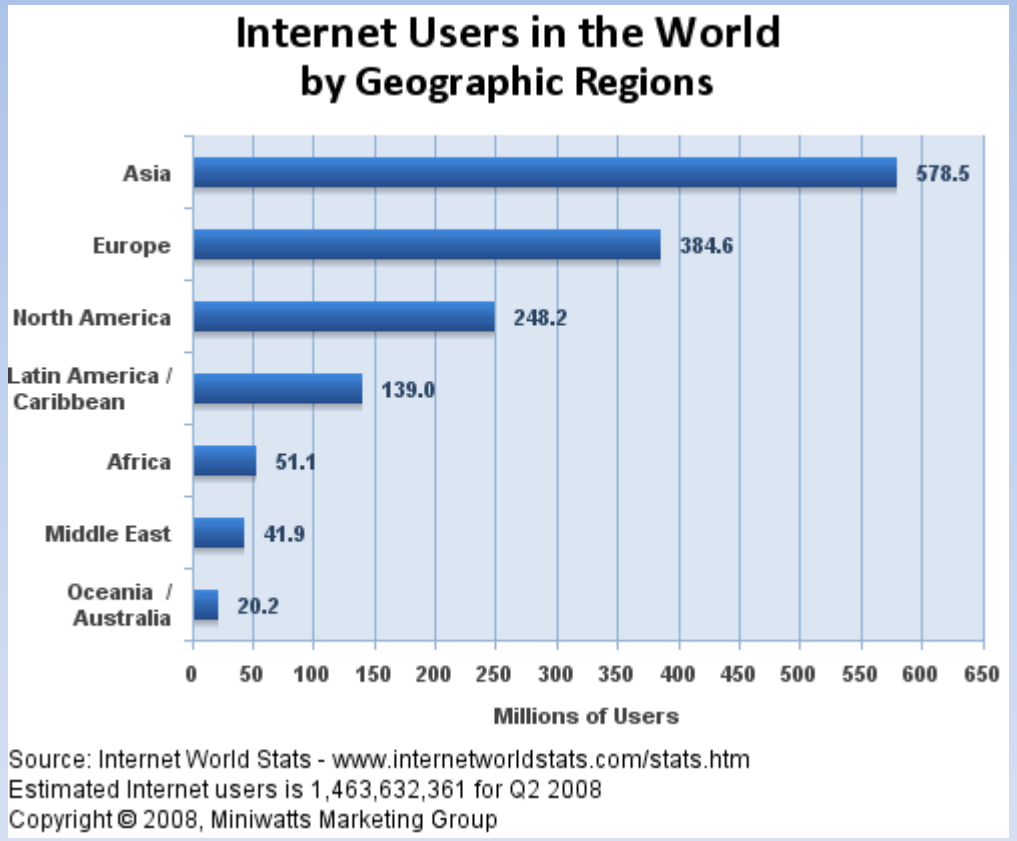
Leonardo Tadashi Kamaura

Orientador

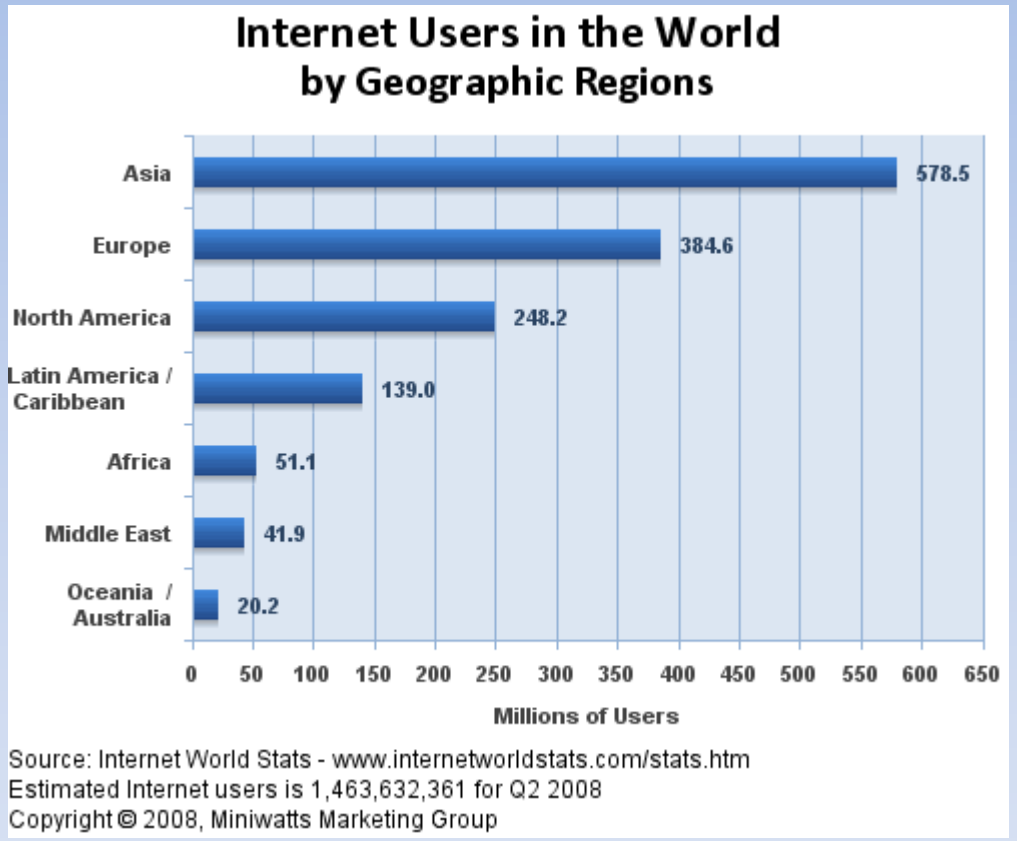
João Eduardo Ferreira

Introdução

Introdução

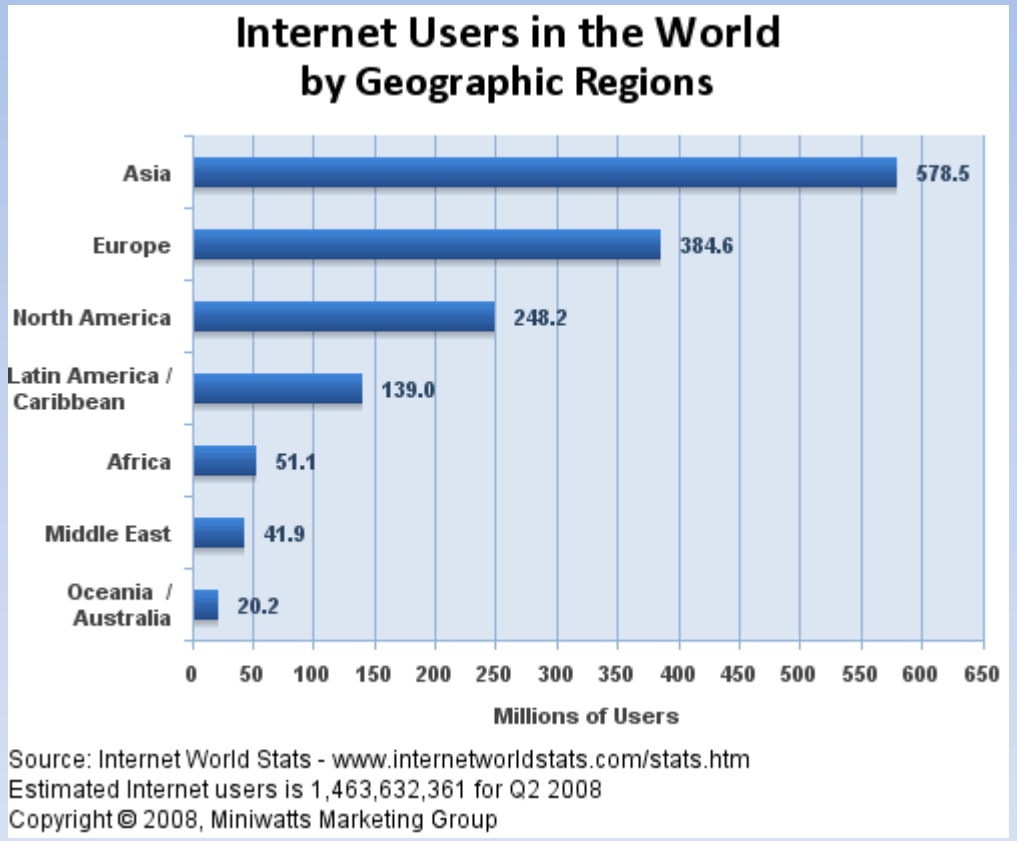

Internet Users in the World
by Geographic Regions
Asia
578.5
Europe
384.6
North America
248.2
Latin America / Caribbean
139.0
Africa
51.1
Middle East
41.9
Oceania / Australia
20.2
0
50
100
150
200
250
300
350
400
450
500
550
600
650
Millions of Users
Source: Internet World Stats - www.internetworldstats.com/stats.htm
Estimated Internet users is 1,463,632,361 for Q2 2008
Copyright © 2008, Miniwatts Marketing Group

Introdução



Média de 305% de crescimento de usuários na internet

Introdução



Média de 305% de crescimento de usuários na internet

=

Aumento do conteúdo na internet produzido pelos usuários

Introdução

A Google registrou em 29 de julho de 2008 a marca de mais de 1 trilhão de páginas disponíveis para consulta a partir do seu site de busca

Google's search index hits one trillion page mark

http://en.wikinews.org/wiki/Google%27s_search_index_hits_one_trillion_page_mark

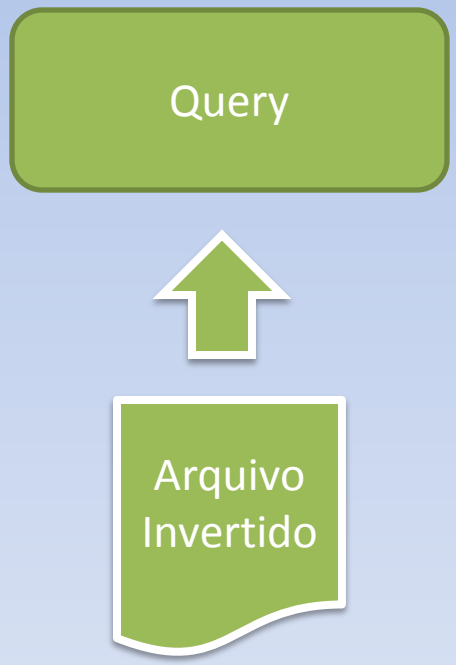
Objetivos

- Estudar conceitos de Recuperação de Informação (RI);

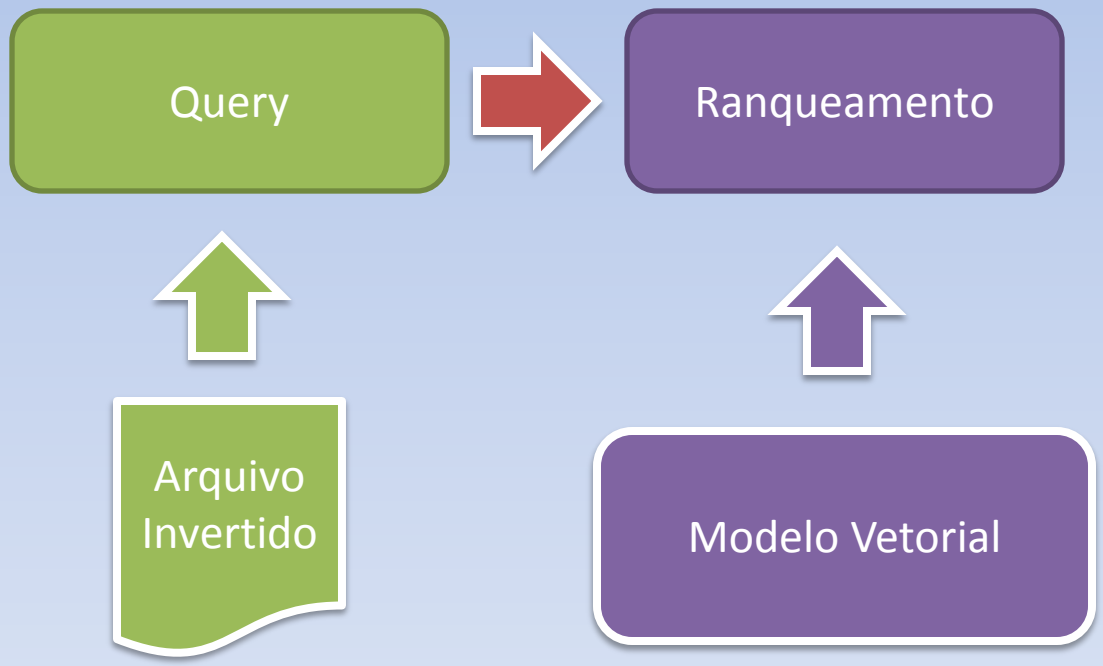
Objetivos

- Estudar conceitos de Recuperação de Informação (RI);
- Integrar uma solução de RI no banco de dados do Projeto Colméia;

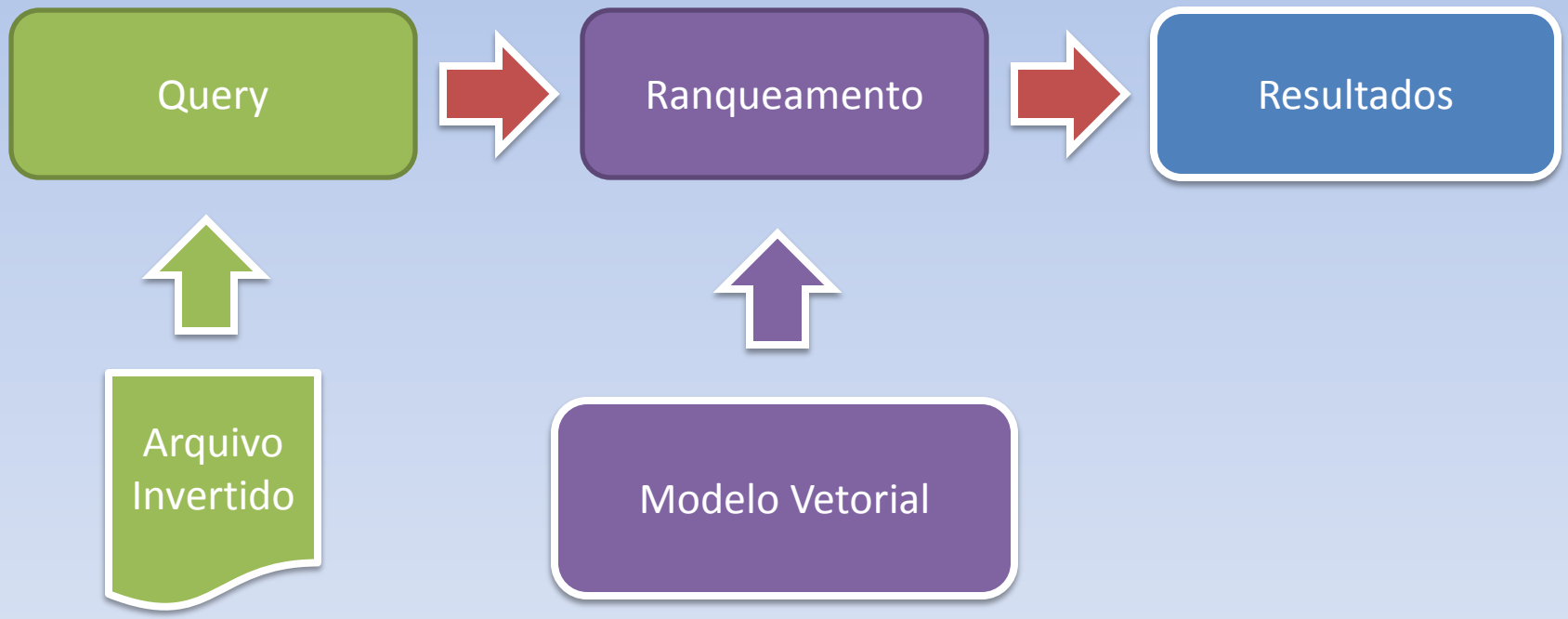
Visão Geral de RI



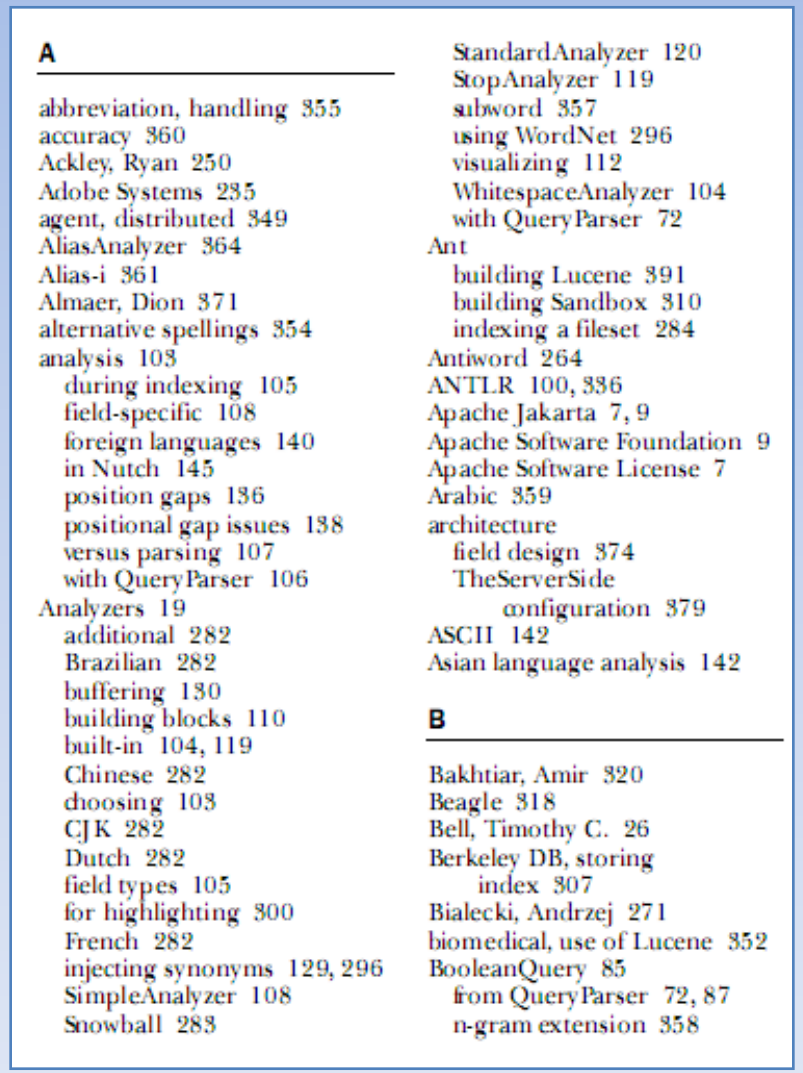
Visão Geral de RI



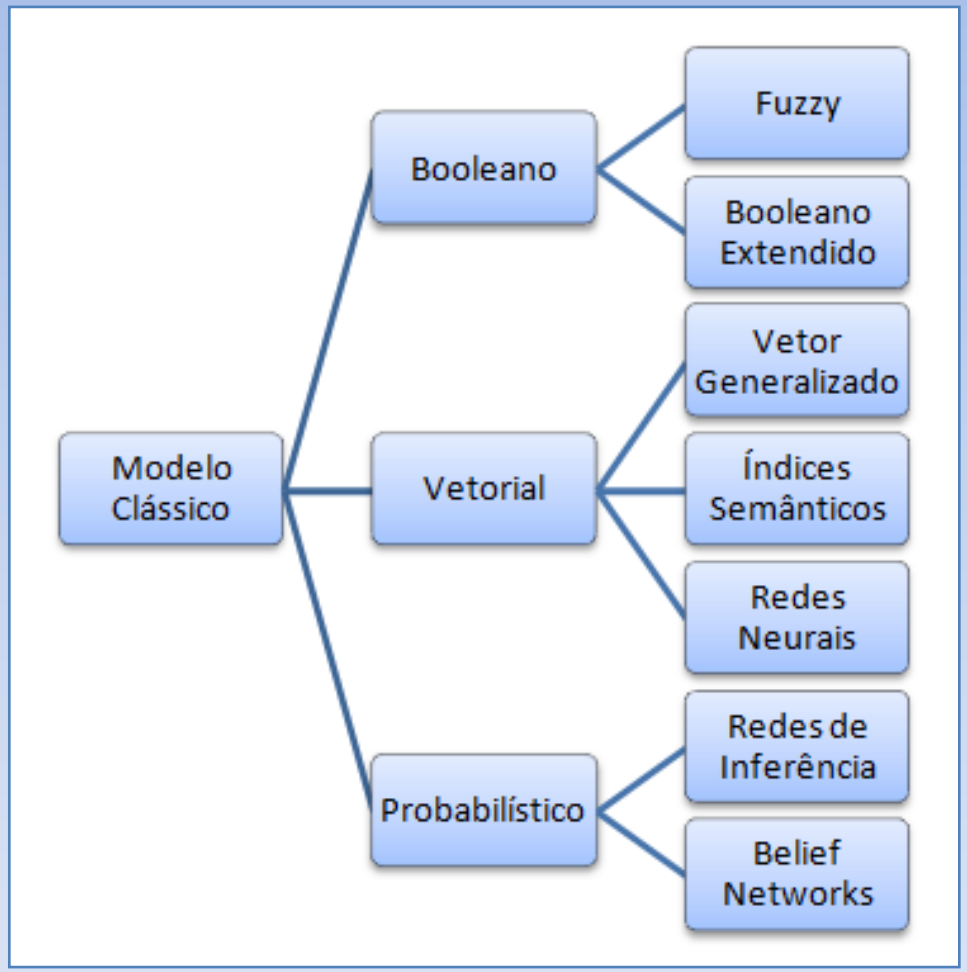
Visão Geral de RI



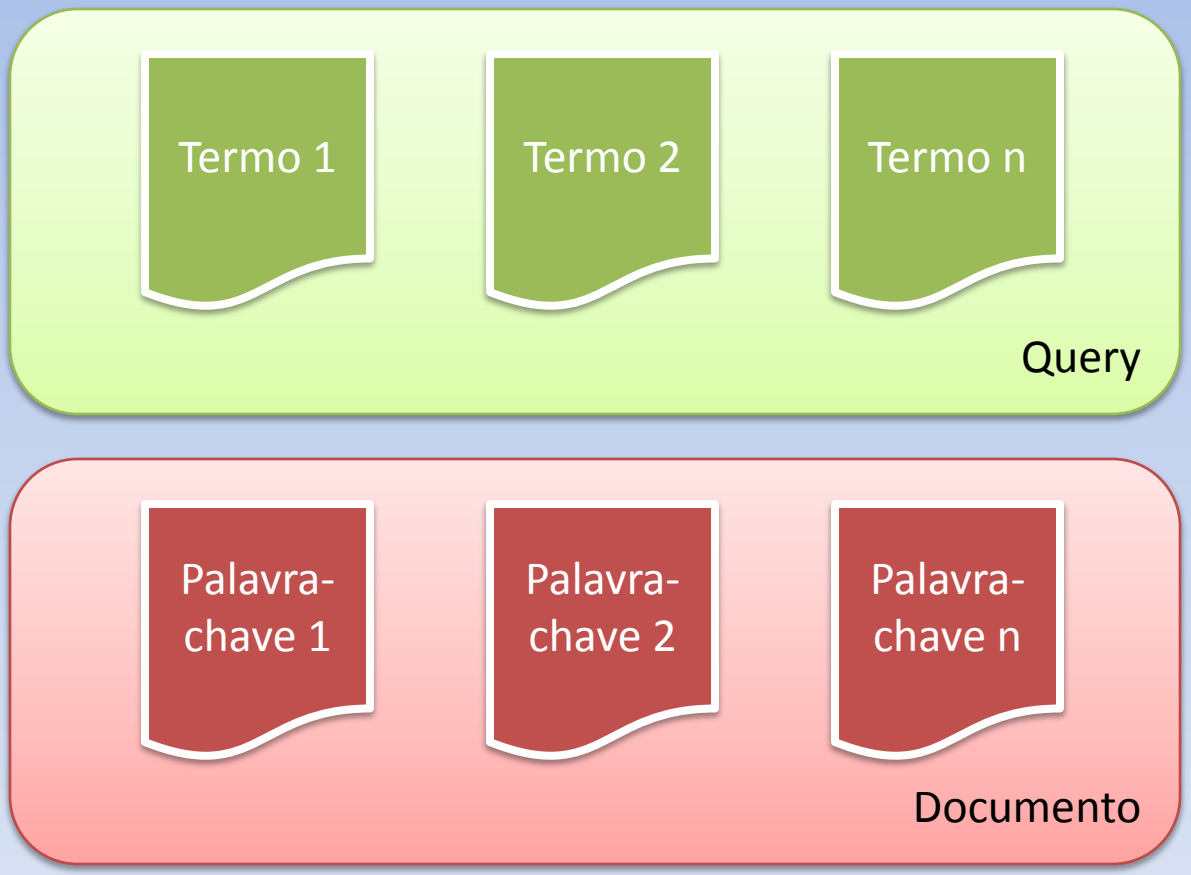
Arquivo Invertido



Modelos


Modelo Clássico
Booleano
Vetorial
Probabilístico
Fuzzy
Booleano Extendido
Vetor Generalizado
Índices Semânticos
Redes Neurais
Redes de Inferência
Belief Networks

Modelo Vetorial



Lucene

- projeto de código aberto da Apache implementado em Java
- gera índices que serão usados na busca
- não é um programa final de busca
- não depende da fonte dos dados
- portável (Perl, Python, C++, .NET, Ruby)

Hibernate Search

- Hibernate Annotations + Lucene
- busca full-text

Hibernate / Hibernate Annotations

Hibernate / Hibernate Annotations

```
@Entity
@Indexed
@Table(name = "livro", schema = "public")
public class Livro implements java.io.Serializable {

   private Long idLivro;
   private String tituloOriginal;
   ...

   @Id
   @DocumentId
  @Column(name = "id_livro", nullable = false)
   public Long getIdLivro() {
      return this.idLivro;
   }

   @Field(index=Index.TOKENIZED, store=Store.NO)
   @Column(name = "titulo_original", length = 300)
   public String getTituloOriginal() {
      return this.tituloOriginal;
   }
```

Hibernate / Hibernate Annotations

```
@Entity
@Indexed
@Table(name = "livro", schema = "public")
public class Livro implements java.io.Serializable {

   private Long idLivro;
   private String tituloOriginal;
   ...

   @Id
   @DocumentId
   @Column(name = "id_livro", nullable = false)
   public Long getIdLivro() {
      return this.idLivro;
   }

   @Field(index=Index.TOKENIZED, store=Store.NO)
   @Column(name = "titulo_original", length = 300)
   public String getTituloOriginal() {
      return this.tituloOriginal;
   }
```

Hibernate - Relacionamentos

```
@Entity
@Indexed
@Table(name = "livro", schema = "public")
public class Livro implements java.io.Serializable {
   ...
   private ItemAcervo itemAcervo;
   private Set<Subtitulo> subtitulos = new HashSet<Subtitulo>();
   ...

   @IndexedEmbedded
   @OneToOne
   @PrimaryKeyJoinColumn
   public ItemAcervo getItemAcervo() {
      return itemAcervo;
   }

   @IndexedEmbedded
   @OneToMany(fetch=FetchType.LAZY, mappedBy = "livro")
   public Set<Subtitulo> getSubtitulos() {
      return subtitulos;
   }
```

Hibernate Search

Hibernate Search

```
@Entity
@Indexed
@Table(name = "livro", schema = "public")
public class Livro implements java.io.Serializable {

   private Long idLivro;
   private String tituloOriginal;
   ...

   @Id
   @DocumentId
   @Column(name = "id_livro", nullable = false)
   public Long getIdLivro() {
      return this.idLivro;
   }

   @Field(index=Index.TOKENIZED, store=Store.NO)
   @Column(name = "titulo_original", length = 300)
   public String getTituloOriginal() {
      return this.tituloOriginal;
   }
```

Hibernate Search - Relacionamentos

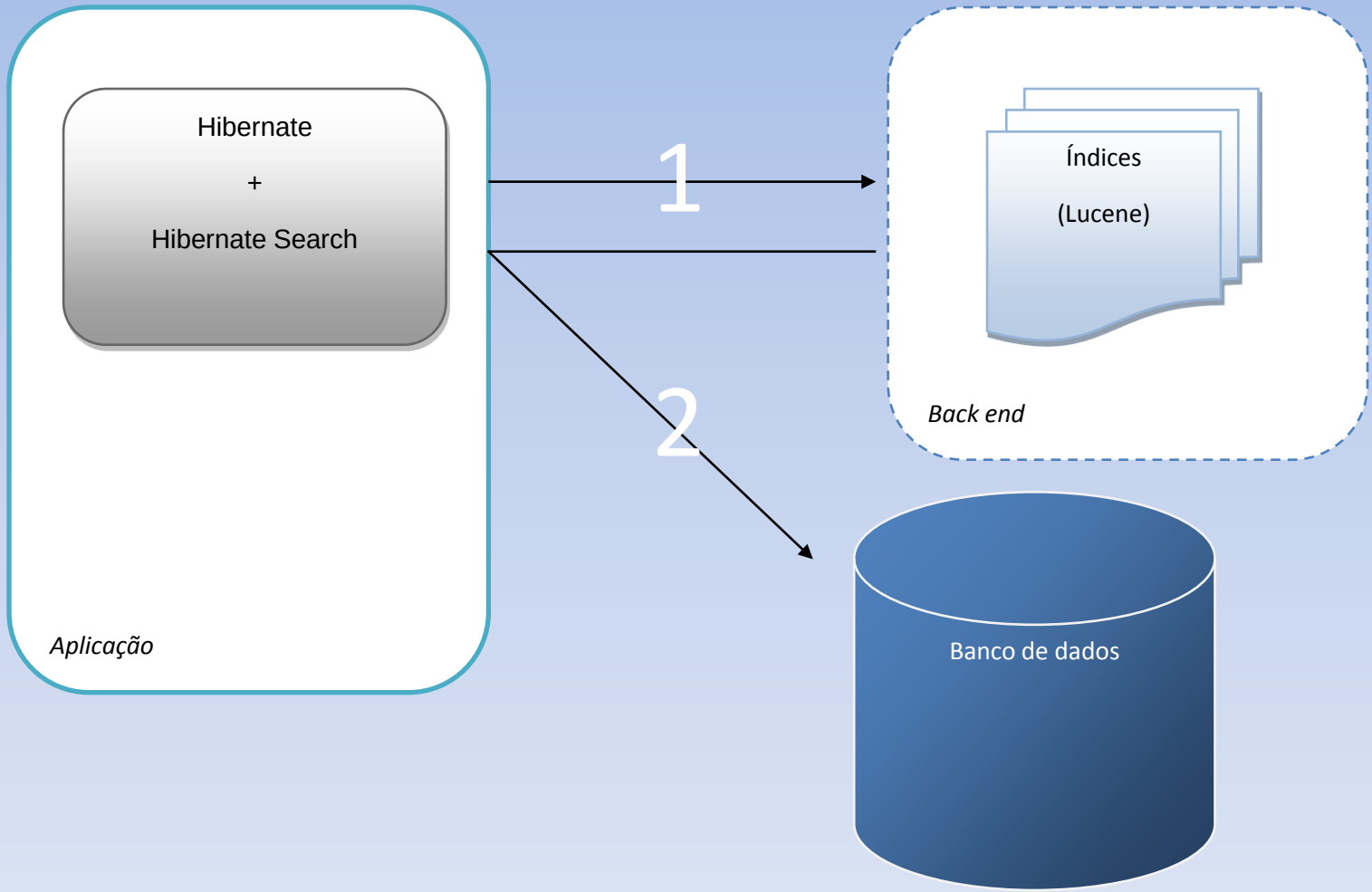
```
@Entity
@Indexed
@Table(name = "livro", schema = "public")
public class Livro implements java.io.Serializable {
   ...
   private ItemAcervo itemAcervo;
   private Set<Subtitulo> subtitulos = new HashSet<Subtitulo>();
   ...

   @IndexedEmbedded
   @OneToOne
   @PrimaryKeyJoinColumn
   public ItemAcervo getItemAcervo() {
      return itemAcervo;
   }

   @IndexedEmbedded
   @OneToMany(fetch=FetchType.LAZY, mappedBy = "livro")
   public Set<Subtitulo> getSubtitulos() {
      return subtitulos;
   }
```

Hibernate Search - Relacionamentos

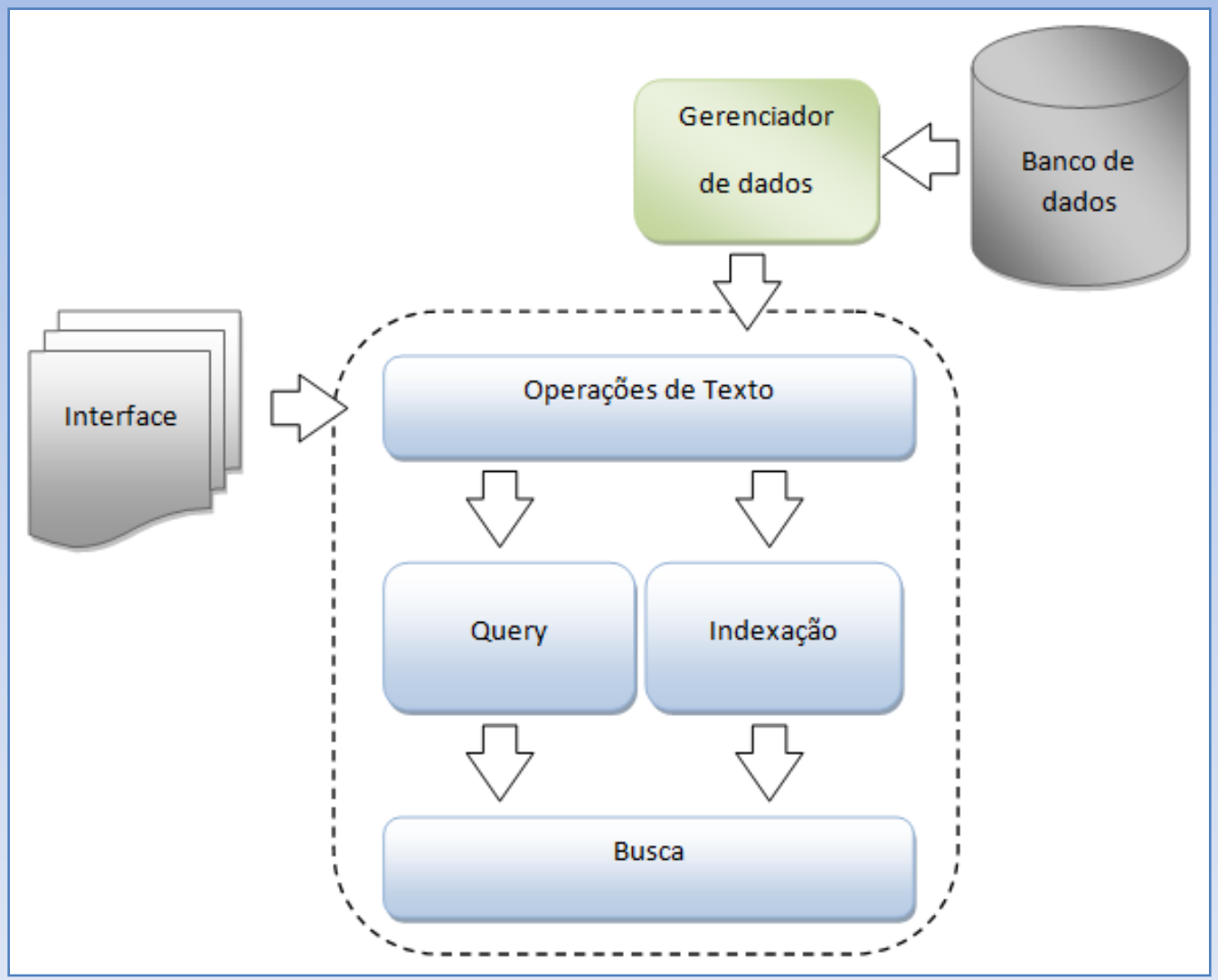
```
@Entity
@Table(name = "subtitulo", schema = "public")
@Embeddable
public class Subtitulo implements java.io.Serializable {

    private long idSubtitulo;
    private String subtitulo;
    private Livro livro;

    @Id
    @Column(name = "id_subtitulo", nullable = false)
    public long getIdSubtitulo() {
        return this.idSubtitulo;
    }

    @Field(index=Index.TOKENIZED, store=Store.NO)
    @Column(name = "subtitulo", nullable = false, length = 100)
    public String getSubtitulo() {
        return this.subtitulo;
    }
```

Utilizando Hibernate Search na Aplicação

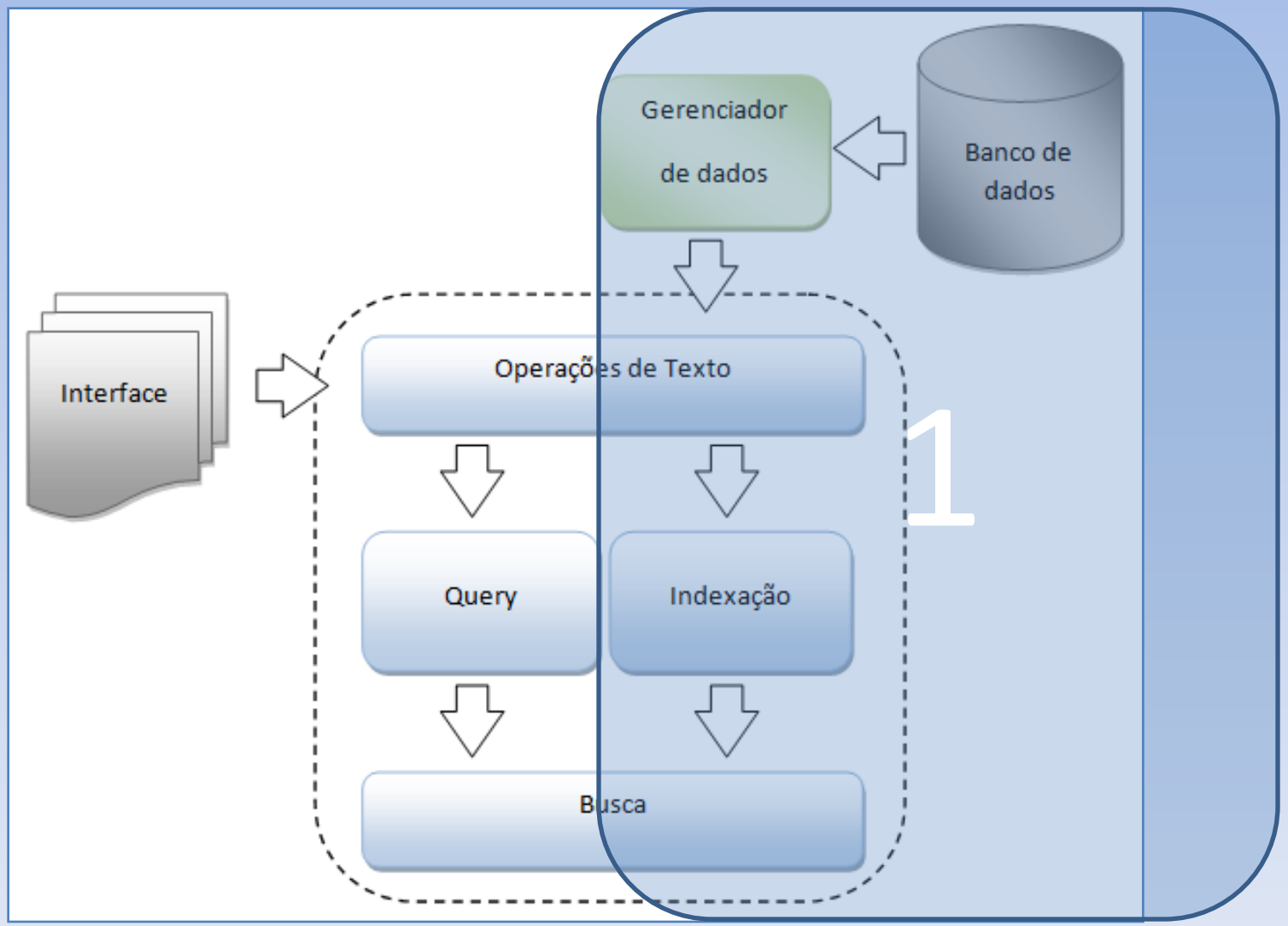
Processos


Hibernate
+
Hibernate Search
Aplicação
1
2
Índices
(Lucene)
Back end
Banco de dados

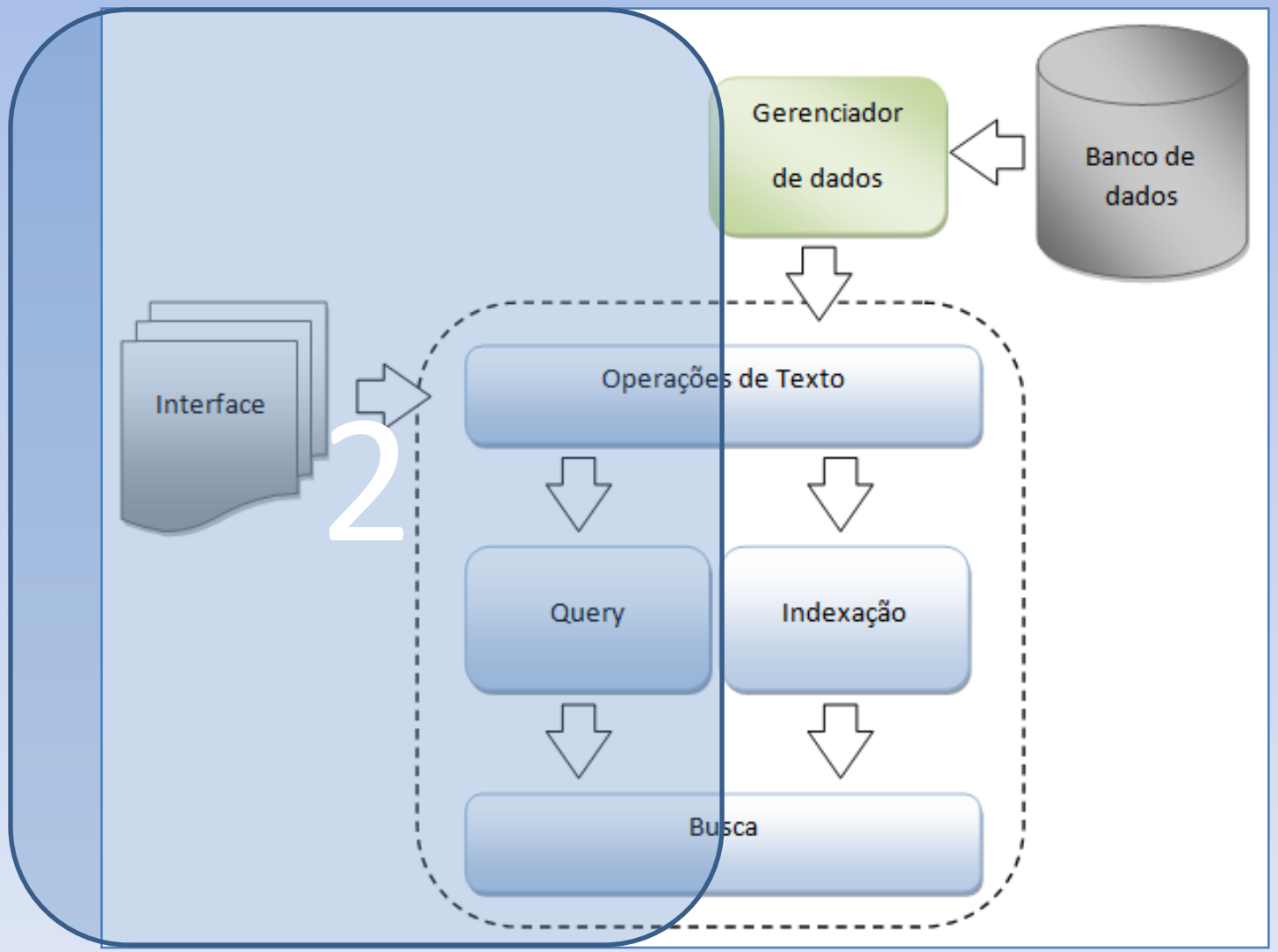
Processos


Gerenciador
de dados
Banco de
dados
Interface
Operações de Texto
Query
Indexação
Busca

Processos


Gerenciador
de dados
Banco de
dados
Interface
Operações de Texto
Query
Indexação
Busca
1

Processos


Gerenciador
de dados
Banco de
dados
Interface
2
Operações de Texto
Query
Indexação
Busca

Indexação

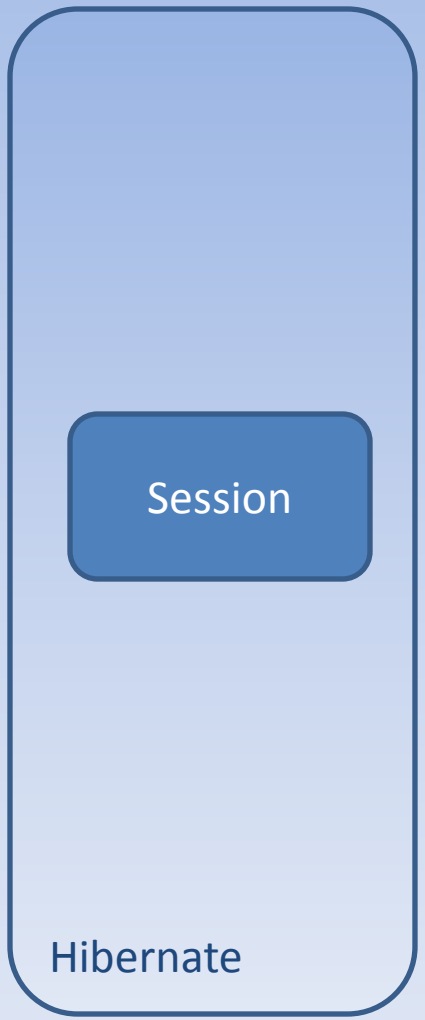
Hibernate Search - Indexação

```
public void indexaLivros(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    FullTextSession fullTextSession = Search.createFullTextSession(session);
    Transaction tx = fullTextSession.beginTransaction();

    List<?> livros = session.createCriteria(Livro.class).list();
    for (Object livro : livros) {
        fullTextSession.index(livro);
    }
    tx.commit();
}
```

Hibernate Search - Indexação

```
public void indexaLlivros(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    FullTextSession fullTextSession = Search.createFullTextSession(session);
    Transaction tx = fullTextSession.beginTransaction();

    List<?> livros = session.createCriteria(Livro.class).list();
    for (Object livro : livros) {
       fullTextSession.index(livro);
    }
    tx.commit();
}
```

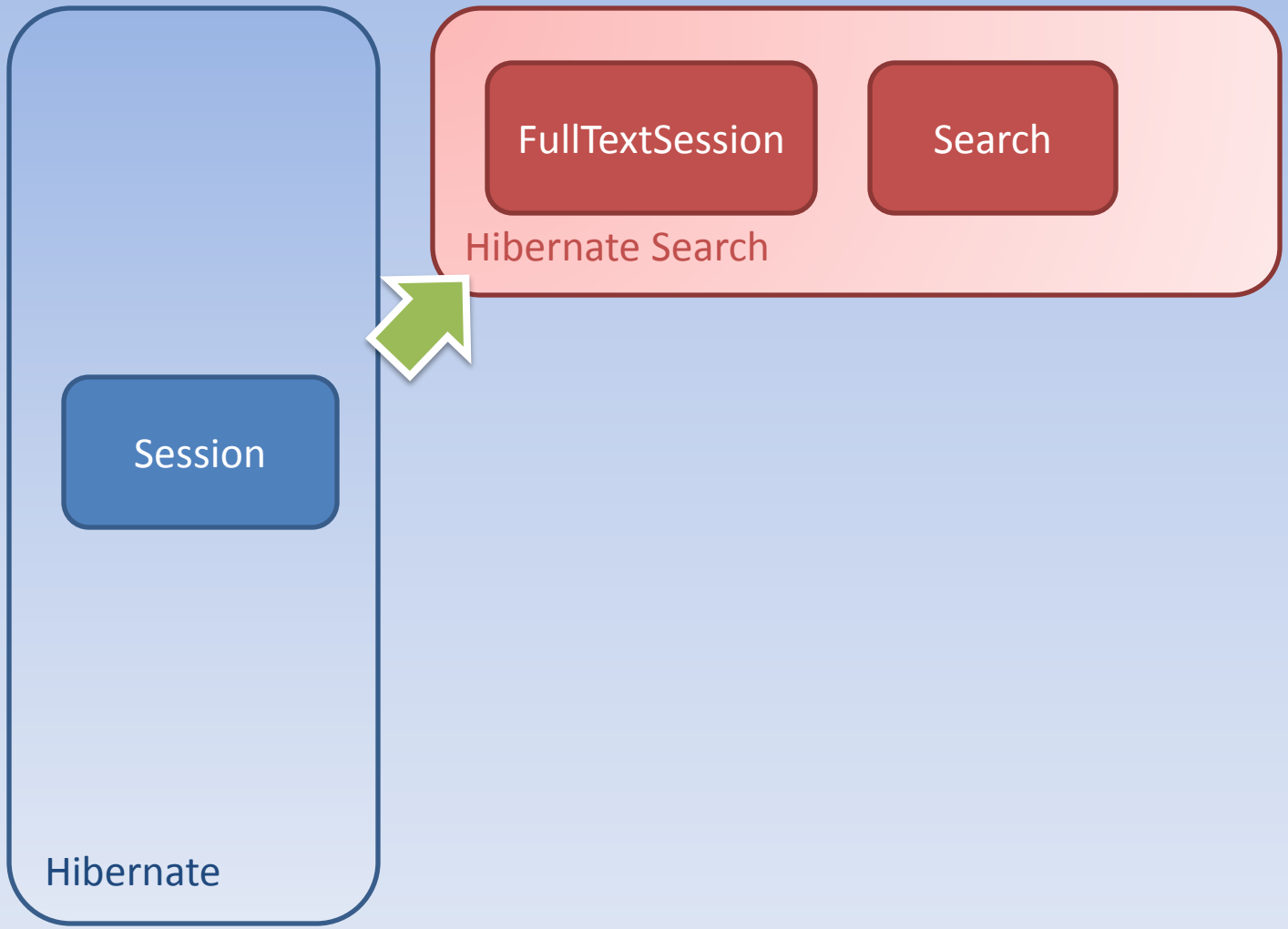
Hibernate Search - Indexação



Hibernate Search - Indexação

```
public void indexaLlivros(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    FullTextSession fullTextSession = Search.createFullTextSession(session);
    Transaction tx = fullTextSession.beginTransaction();

    List<?> livros = session.createCriteria(Livro.class).list();
    for (Object livro : livros) {
       fullTextSession.index(livro);
    }
    tx.commit();
}
```

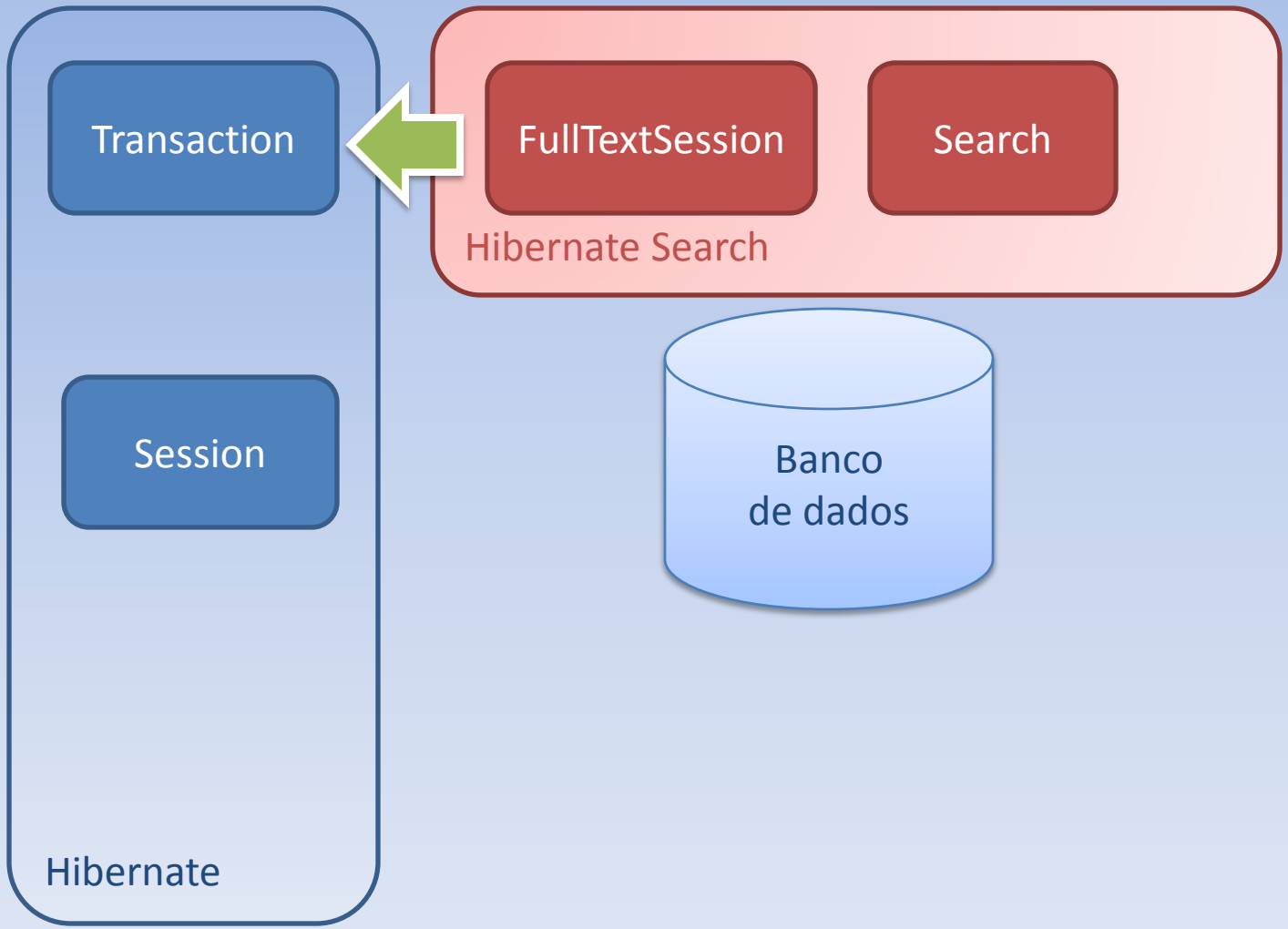
Hibernate Search - Indexação



Hibernate Search - Indexação

```
public void indexaLivros(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    FullTextSession fullTextSession = Search.createFullTextSession(session);
    Transaction tx = fullTextSession.beginTransaction();

    List<?> livros = session.createCriteria(Livro.class).list();
    for (Object livro : livros) {
       fullTextSession.index(livro);
    }
    tx.commit();
}
```

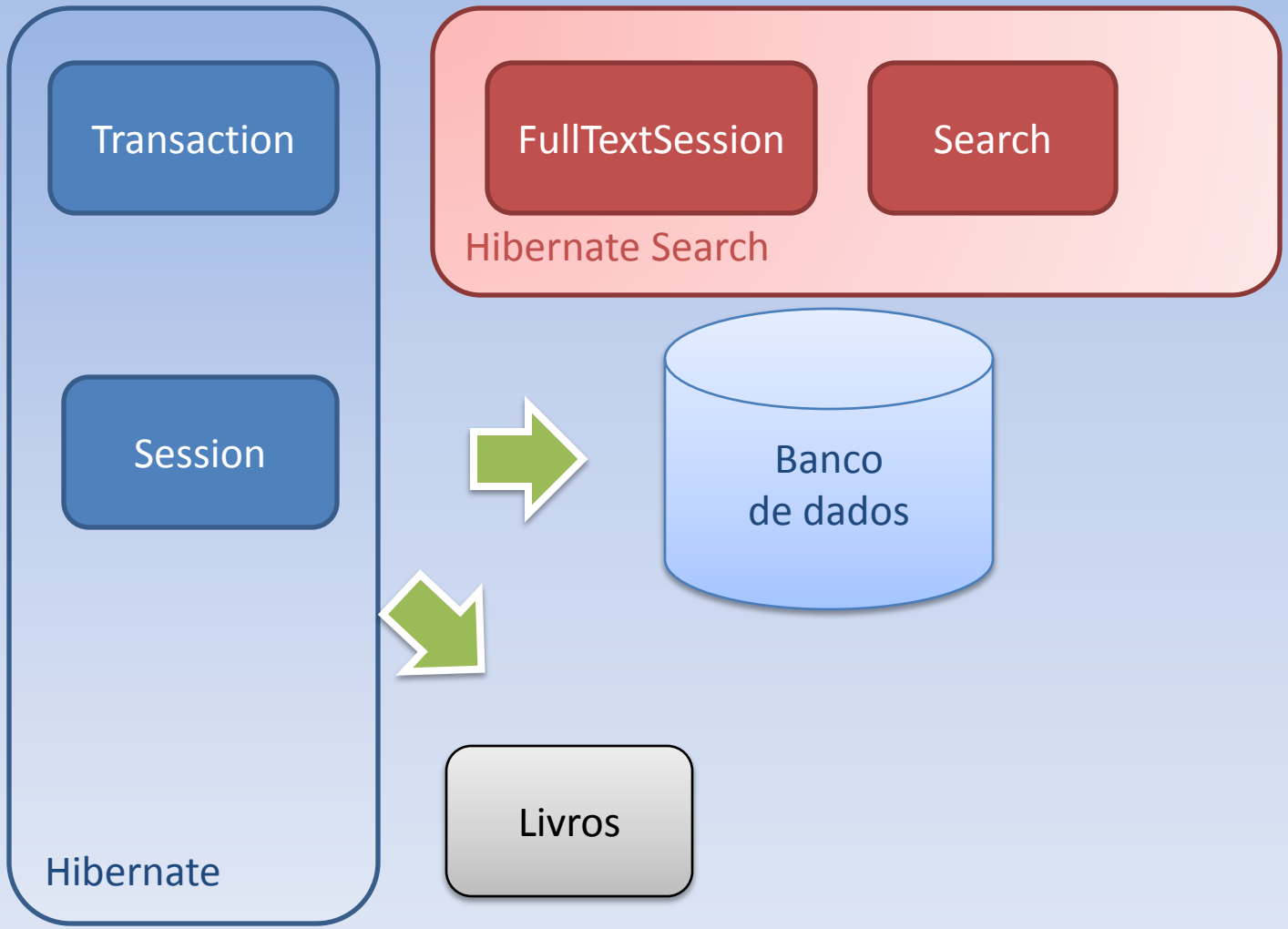
Hibernate Search - Indexação



Hibernate Search - Indexação

```
public void indexaLivros(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    FullTextSession fullTextSession = Search.createFullTextSession(session);
    Transaction tx = fullTextSession.beginTransaction();

    List<?> livros = session.createCriteria(Livro.class).list();
     for (Object livro : livros) {
        fullTextSession.index(livro);
     }
    tx.commit();
}
```

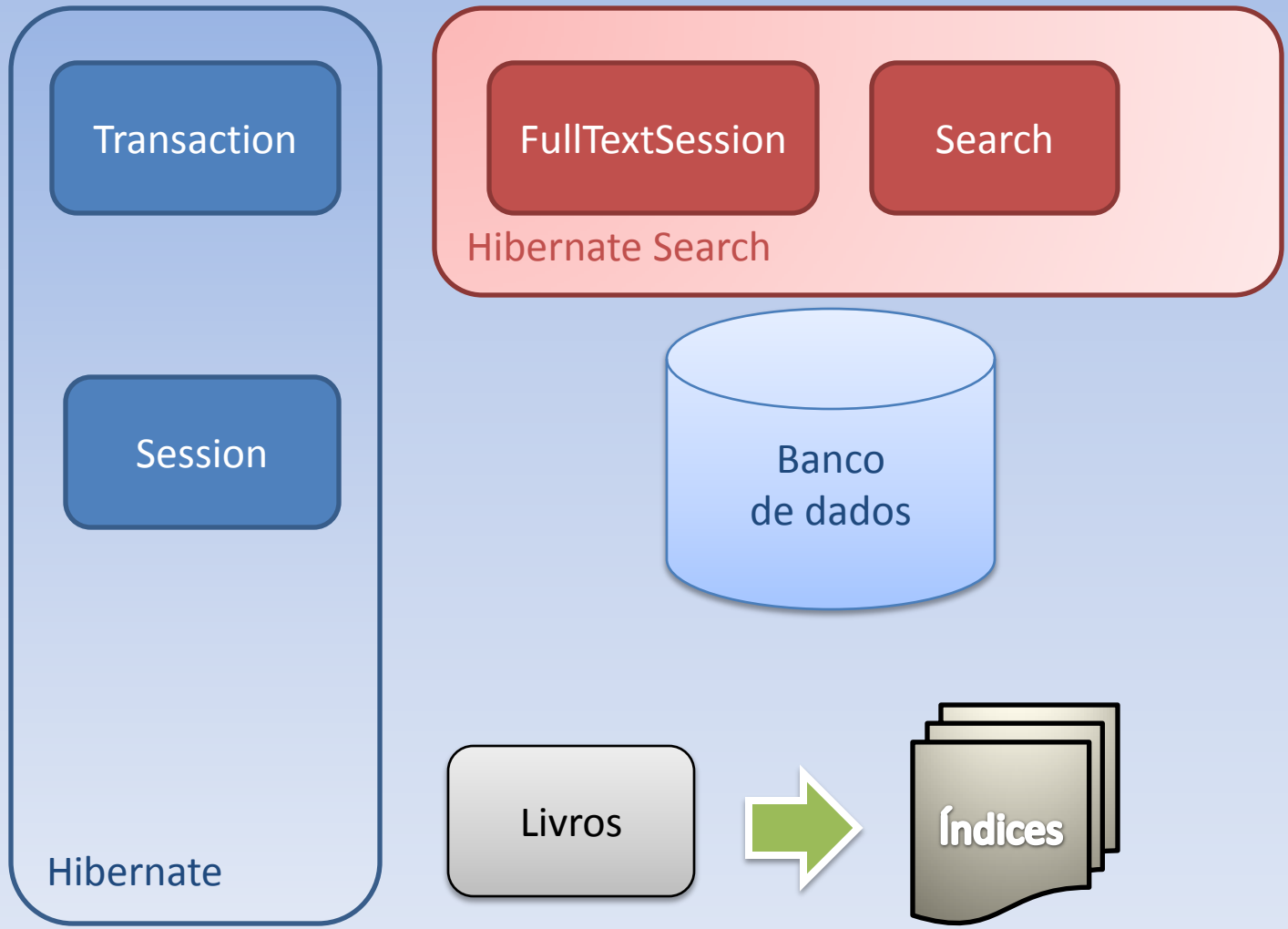
Hibernate Search - Indexação



Hibernate Search - Indexação

```
public void indexaLivros(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    FullTextSession fullTextSession = Search.createFullTextSession(session);
    Transaction tx = fullTextSession.beginTransaction();

    List<?> livros = session.createCriteria(Livro.class).list();
     for (Object livro : livros) {
        fullTextSession.index(livro);
     }
    tx.commit();
}
```

Hibernate Search - Indexação



Busca

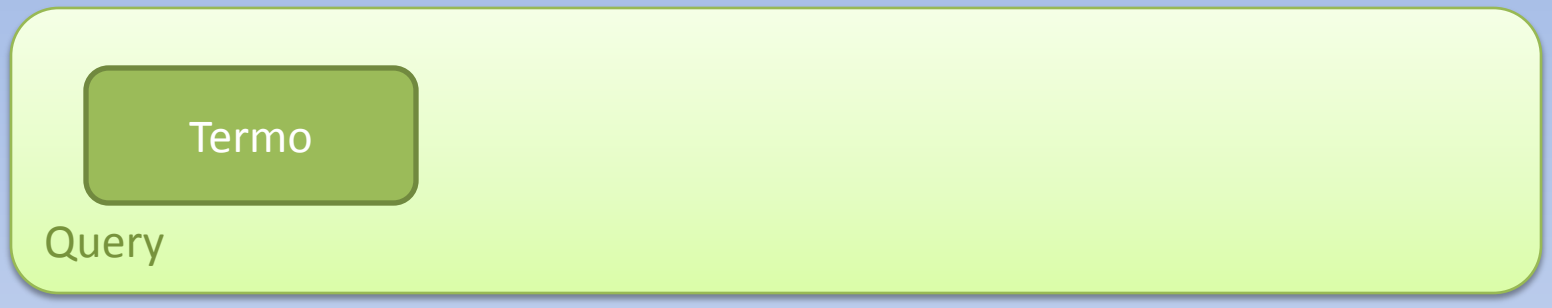
Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
      String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome",
            "livroEditoraCidade.editora.editora", "assuntosDedalus"};
      MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
      org.apache.lucene.search.Query luceneQuery = null;
      try {
         luceneQuery = parser.parse(termo);
      } catch (ParseException e) {
         e.printStackTrace();
      }
       org.hibernate.search.FullTextQuery fullTextQuery =
      fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
      return ajustaPaginacao(session, indice, maximo);
}
```

Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
      String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome",
            "livroEditoraCidade.editora.editora", "assuntosDedalus"};
      MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
      org.apache.lucene.search.Query luceneQuery = null;
      try {
         luceneQuery = parser.parse(termo);
      } catch (ParseException e) {
         e.printStackTrace();
      }
       org.hibernate.search.FullTextQuery fullTextQuery =
      fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
      return ajustaPaginacao(session, indice, maximo);
}
```

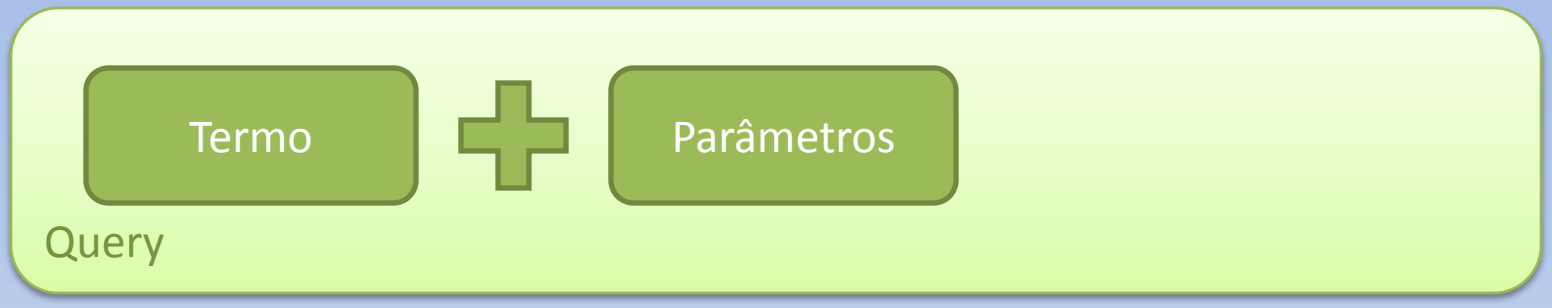
Hibernate Search – Busca



Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
    String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome",
        "livroEditoraCidade.editora.editora", "assuntosDedalus"};
    MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
    org.apache.lucene.search.Query luceneQuery = null;
    try {
       luceneQuery = parser.parse(termo);
    } catch (ParseException e) {
       e.printStackTrace();
    }
     org.hibernate.search.FullTextQuery fullTextQuery =
    fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
    return ajustaPaginacao(session, indice, maximo);
}
```

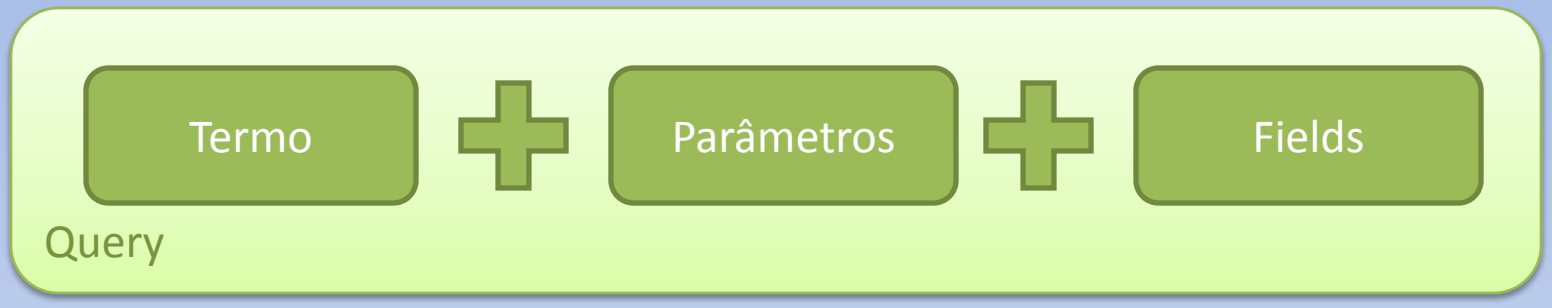
Hibernate Search – Busca



Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
     String[] fields = {"tituloOriginal", "subtitulos.subtitulo",
    "livrosAutores.autor.nome ",  "livroEditoraCidade.editora.editora",
    "assuntosDedalus"};
    MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
    org.apache.lucene.search.Query luceneQuery = null;
    try {
       luceneQuery = parser.parse(termo);
    } catch (ParseException e) {
       e.printStackTrace();
    }
     org.hibernate.search.FullTextQuery fullTextQuery =
    fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
    return ajustaPaginacao(session, indice, maximo);
}
```

Hibernate Search – Busca



Hibernate Search – Busca

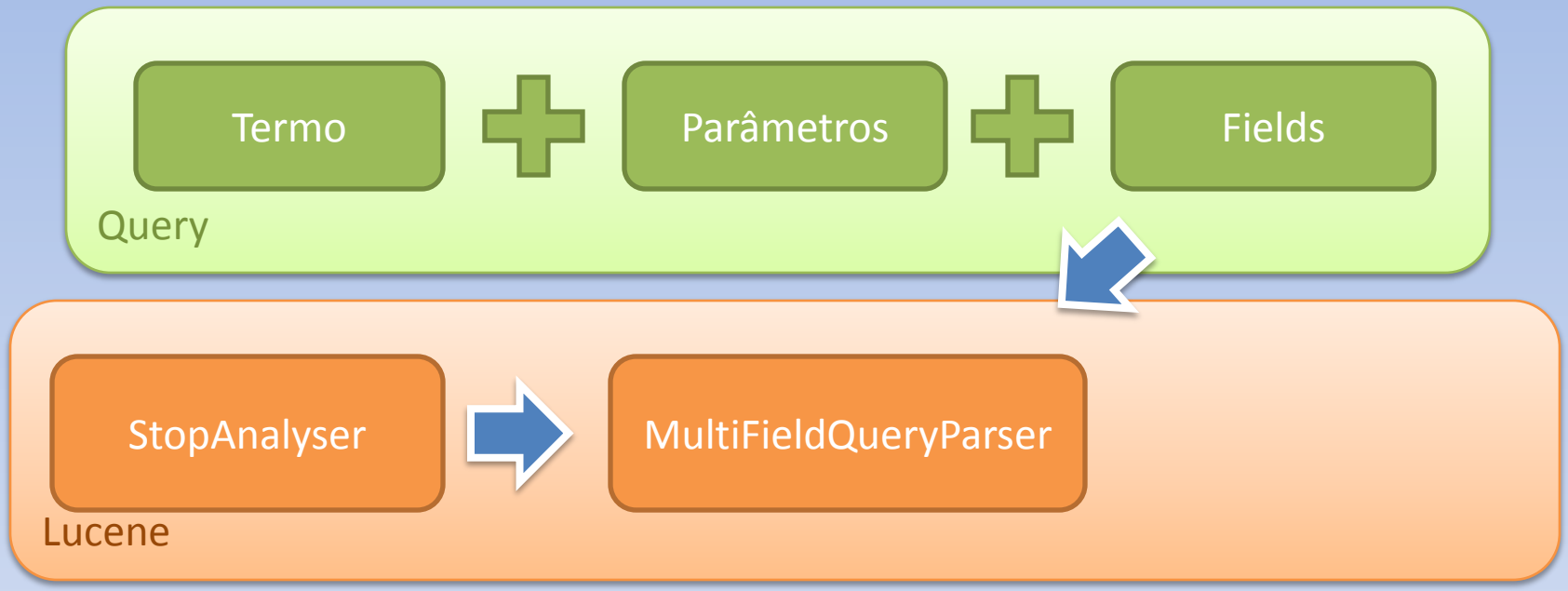
```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
      String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome ",
            "livroEditoraCidade.editora.editora", "assuntosDedalus"};
      MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
      org.apache.lucene.search.Query luceneQuery = null;
      try {
         luceneQuery = parser.parse(termo);
      } catch (ParseException e) {
         e.printStackTrace();
      }
      org.hibernate.search.FullTextQuery fullTextQuery =
      fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
      return ajustaPaginacao(session, indice, maximo);
}
```

Hibernate Search – Busca

Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
    String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome ",
            "livroEditoraCidade.editora.editora", "assuntosDedalus"};
    MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new
    StopAnalyzer());
    org.apache.lucene.search.Query luceneQuery = null;
    try {
        luceneQuery = parser.parse(termo);
    } catch (ParseException e) {
        e.printStackTrace();
    }
    org.hibernate.search.FullTextQuery fullTextQuery =
    fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
    return ajustaPaginacao(session, indice, maximo);
}
```

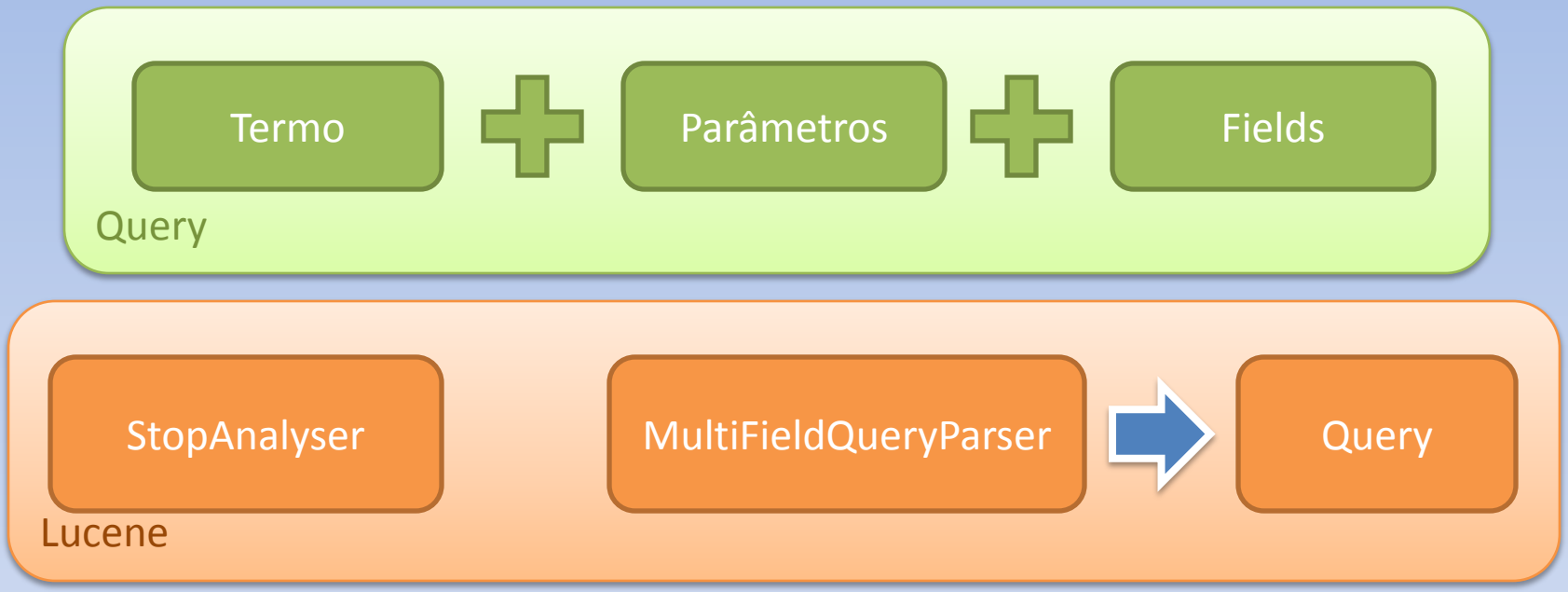
Hibernate Search – Busca



Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
    String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome ",
            "livroEditoraCidade.editora.editora", "assuntosDedalus"};
    MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
    org.apache.lucene.search.Query luceneQuery = null;
     try {
        luceneQuery = parser.parse(termo);
     } catch (ParseException e) {
        e.printStackTrace();
     }
    org.hibernate.search.FullTextQuery fullTextQuery =
    fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
    return ajustaPaginacao(session, indice, maximo);
}
```

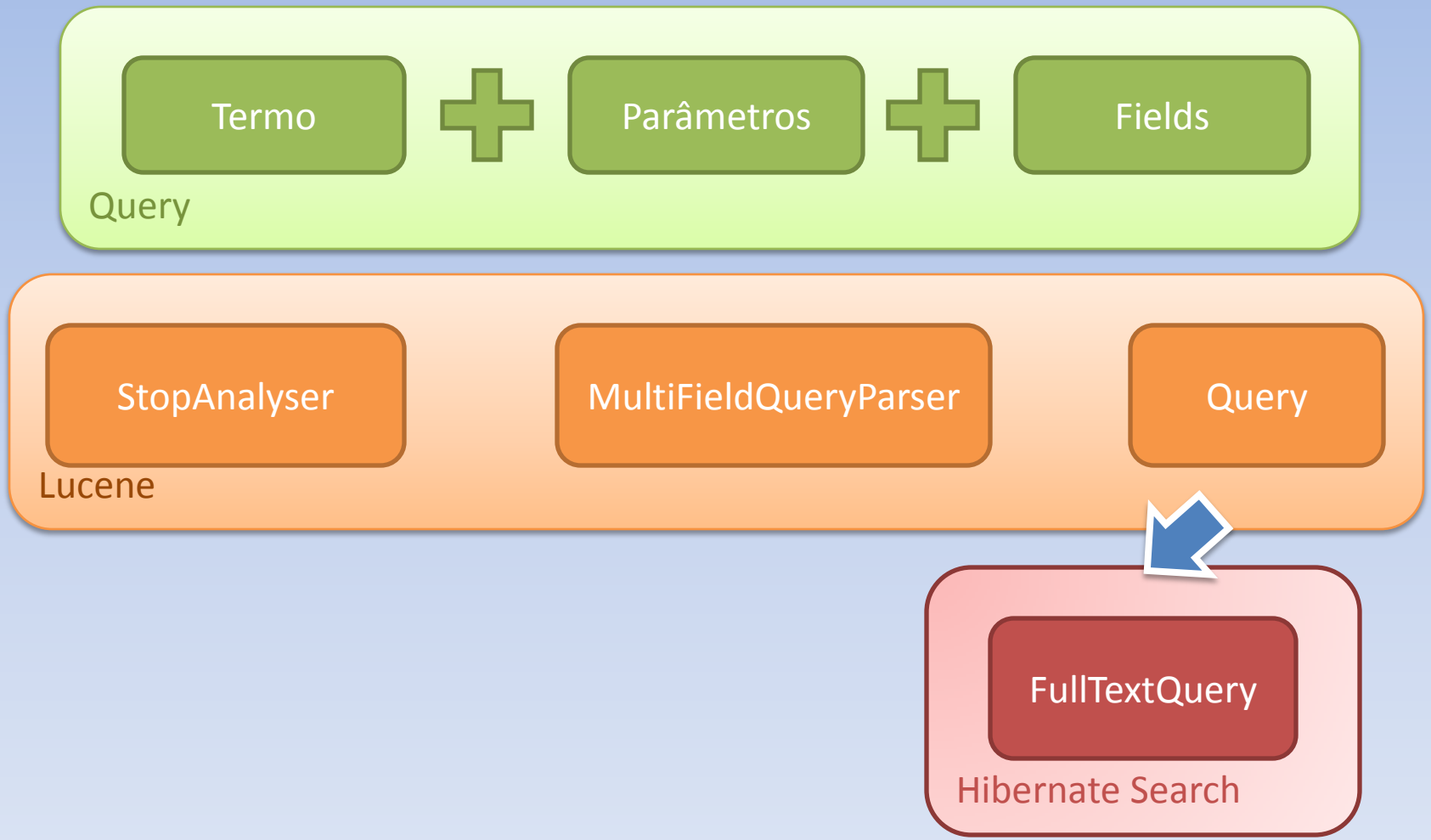
Hibernate Search – Busca



Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
      String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome ",
            "livroEditoraCidade.editora.editora", "assuntosDedalus"};
     MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
      org.apache.lucene.search.Query luceneQuery = null;
      try {
         luceneQuery = parser.parse(termo);
      } catch (ParseException e) {
         e.printStackTrace();
      }
      org.hibernate.search.FullTextQuery fullTextQuery =
       fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
     return ajustaPaginacao(session, indice, maximo);
}
```

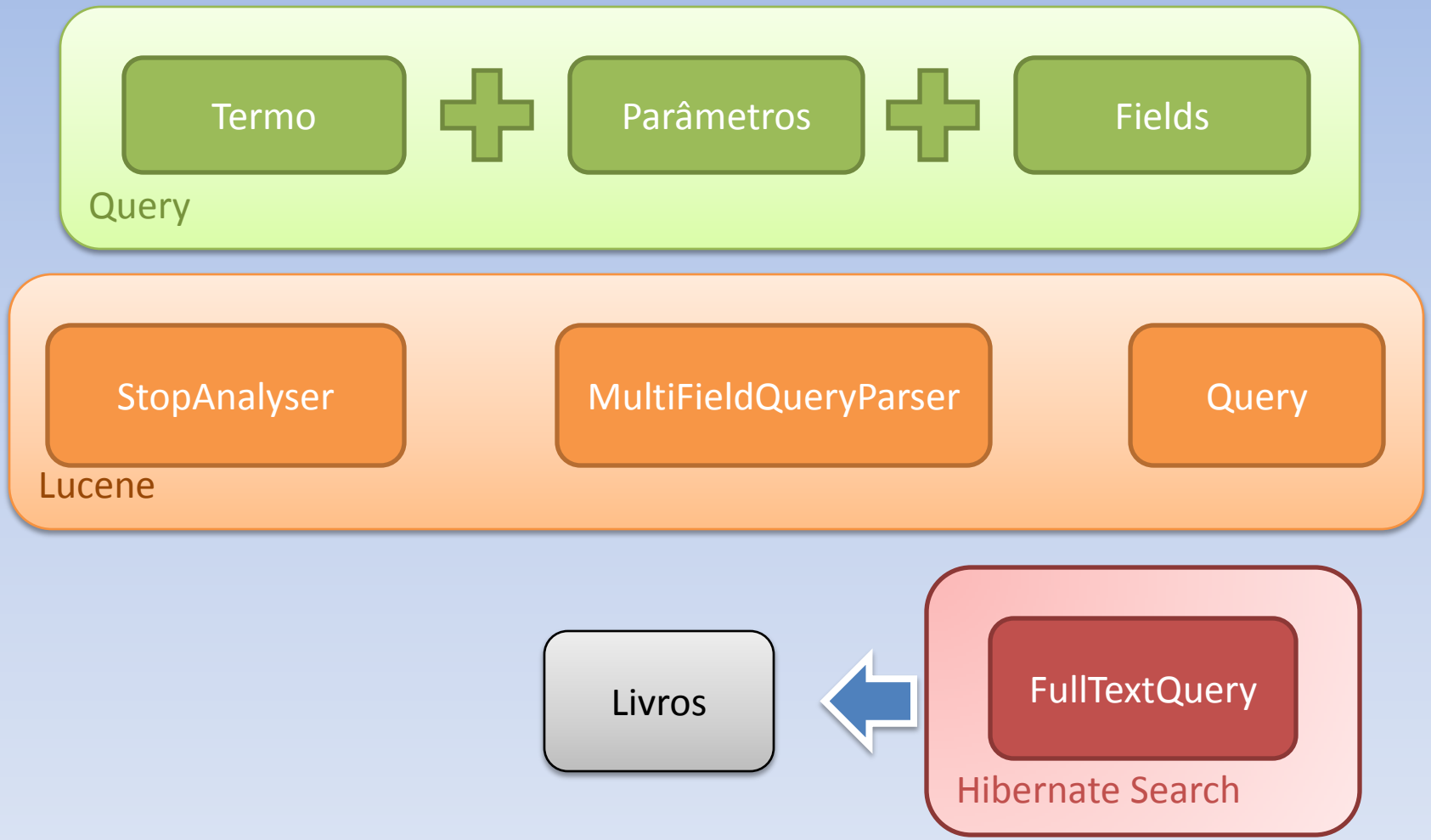
Hibernate Search – Busca



Hibernate Search – Busca

```
public List<Livro> buscaLivro(String termo, int indice, int maximo) throws Exception {
        String[] fields = {"tituloOriginal", "subtitulos.subtitulo", "livrosAutores.autor.nome ",
                "livroEditoraCidade.editora.editora", "assuntosDedalus"};
      MultiFieldQueryParser parser = new MultiFieldQueryParser(fields, new StopAnalyzer());
       org.apache.lucene.search.Query luceneQuery = null;
       try {
          luceneQuery = parser.parse(termo);
       } catch (ParseException e) {
          e.printStackTrace();
       }
       org.hibernate.search.FullTextQuery fullTextQuery =
       fullTextSession.createFullTextQuery( luceneQuery , Livro.class);
      return ajustaPaginacao(session, indice, maximo);
}
```

Hibernate Search – Busca



Score

Lucene - Score

- É uma combinação do **Modelo de Espaço Vetorial** e do **Modelo Booleano**.
- O score de uma query q para um documento d correlaciona à distância do coseno ou ao produto escalar entre vetores de documentos e queries no Modelo de Espaço Vetorial.

Fórmula:

$$score(q,d) = coord(q,d) \cdot queryNorm(q,d) \cdot \sum_{t \text{ in } d} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost(\ ) \cdot norm(t,d) \right)$$

Lucene - Score

$$score(q,d) = coord(q,d) \cdot queryNorm(q,d) \cdot \sum_{t \text{ in } d} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost(\ ) \cdot norm(t,d) \right)$$

***tf(t in d)*: term frequency**

O número de vezes que o termo *t* ocorre no documento *d*

***idf(t in d)*: inverse document frequency**

O número de documentos que contêm o termo *t*

***coord(q, d)*: fator**

baseado na quantidade de termos da query *q* que são encontrados no documento *d*

***queryNorm(q, d)*: fator de normalização**

usada para permitir a comparação entre os scores das queries

***t.getBoost()*:**

atribui peso para o termo *t* na query *q* em tempo de busca

***norm(t, d)*:**

encapsula alguns fatores de boost em tempo de indexação

Lucene - Score

$$score(q,d) = coord(q,d) \cdot queryNorm(q,d) \cdot \sum_{t \text{ in } d} \left(tf(t \text{ in } d) \cdot \mathbf{idf}(\mathbf{t})^2 \cdot t.getBoost(\ ) \cdot norm(t,d) \right)$$

***tf(t in d)*: term frequency**

O número de vezes que o termo *t* ocorre no documento *d*

***idf(t in d)*: inverse document frequency**

O número de documentos que contêm o termo *t*

***coord(q, d)*: fator**

baseado na quantidade de termos da query *q* que são encontrados no documento *d*

***queryNorm(q, d)*: fator de normalização**

usada para permitir a comparação entre os scores das queries

***t.getBoost()*:**

atribui peso para o termo *t* na query *q* em tempo de busca

***norm(t, d)*:**

encapsula alguns fatores de boost em tempo de indexação

Lucene - Score

$$score(q,d) = \mathbf{coord}(q,d) \cdot queryNorm(q,d) \cdot \sum_{t \text{ in } d} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost(\ ) \cdot norm(t,d) \right)$$

***tf(t in d)*: term frequency**

O número de vezes que o termo *t* ocorre no documento *d*

***idf(t in d)*: inverse document frequency**

O número de documentos que contêm o termo *t*

***coord(q, d)*: fator**

baseado na quantidade de termos da query q que são encontrados no documento d

***queryNorm(q, d)*: fator de normalização**

usada para permitir a comparação entre os scores das queries

***t.getBoost()*:**

atribui peso para o termo t na query q em tempo de busca

***norm(t, d)*:**

encapsula alguns fatores de boost em tempo de indexação

Lucene - Score

$$score(q,d) = coord(q,d) \cdot \mathbf{queryNorm(q,d)} \cdot \sum_{t \text{ in } d} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost(\ ) \cdot norm(t,d) \right)$$

***tf(t in d)*: term frequency**

O número de vezes que o termo *t* ocorre no documento *d*

***idf(t in d)*: inverse document frequency**

O número de documentos que contêm o termo *t*

***coord(q, d)*: fator**

baseado na quantidade de termos da query *q* que são encontrados no documento *d*

***queryNorm(q, d)*: fator de normalização**

usada para permitir a comparação entre os scores das queries

***t.getBoost()*:**

atribui peso para o termo *t* na query *q* em tempo de busca

***norm(t, d)*:**

encapsula alguns fatores de boost em tempo de indexação

Lucene - Score

$$score(q,d) = coord(q,d) \cdot queryNorm(q,d) \cdot \sum_{t \text{ in } d} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot \mathbf{t.getBoost(\ )} \cdot norm(t,d) \right)$$

***tf(t in d)*: term frequency**

O número de vezes que o termo *t* ocorre no documento *d*

***idf(t in d)*: inverse document frequency**

O número de documentos que contêm o termo *t*

***coord(q, d)*: fator**

baseado na quantidade de termos da query *q* que são encontrados no documento *d*

***queryNorm(q, d)*: fator de normalização**

usada para permitir a comparação entre os scores das queries

***t.getBoost()*:**

atribui peso para o termo *t* na query *q* em tempo de busca

***norm(t, d)*:**

encapsula alguns fatores de boost em tempo de indexação

Lucene - Score

$$score(q,d) = coord(q,d) \cdot queryNorm(q,d) \cdot \sum_{t \text{ in } d} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost(\ ) \cdot \mathbf{norm(t,d)} \right)$$

***tf(t in d)*: term frequency**

O número de vezes que o termo *t* ocorre no documento *d*

***idf(t in d)*: inverse document frequency**

O número de documentos que contêm o termo *t*

***coord(q, d)*: fator**

baseado na quantidade de termos da query *q* que são encontrados no documento *d*

***queryNorm(q, d)*: fator de normalização**

usada para permitir a comparação entre os scores das queries

***t.getBoost()*:**

atribui peso para o termo *t* na query *q* em tempo de busca

***norm(t, d)*:**

encapsula alguns fatores de boost em tempo de indexação

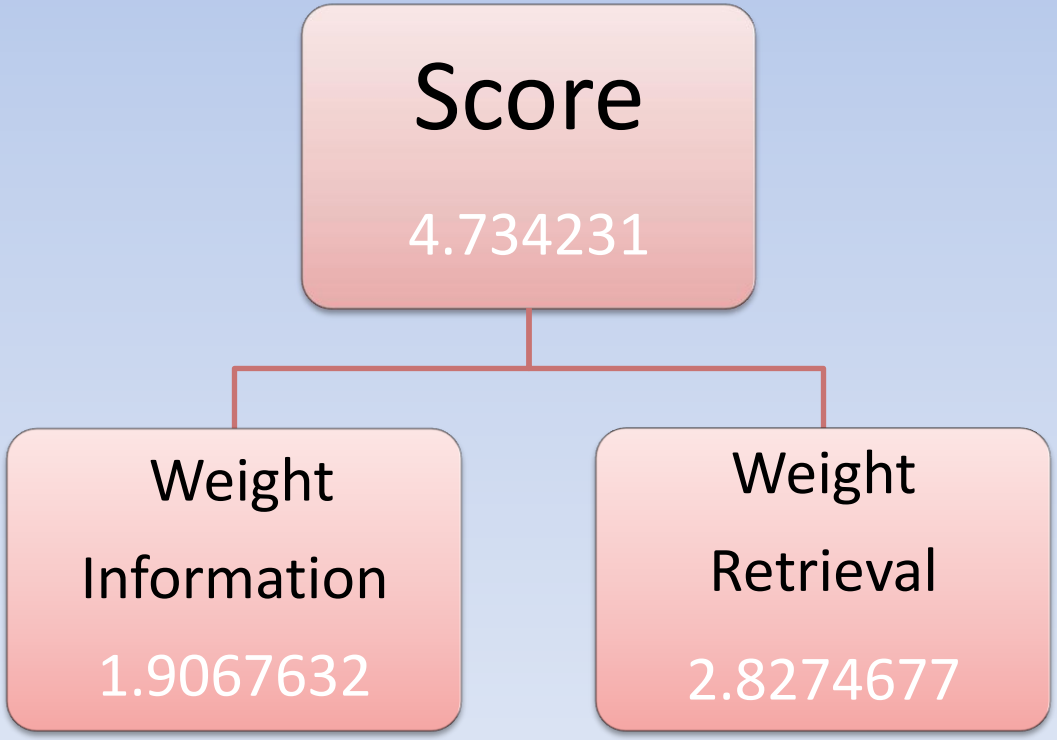
Lucene - Score

Query: Information Retrieval
Resultado: Information analysis and retrieval

4.734231 = (MATCH) sum of:
  1.9067632 = (MATCH) weight(tituloOriginal:information in 2099), product of:
    0.6346345 = queryWeight(tituloOriginal:information), product of:
      6.009012 = idf(docFreq=36, numDocs=5541)
      0.105613776 = queryNorm
    3.004506 = (MATCH) fieldWeight(tituloOriginal:information in 2099), product of:
      1.0 = tf(termFreq(tituloOriginal:information)=1)
      6.009012 = idf(docFreq=36, numDocs=5541)
      0.5 = fieldNorm(field=tituloOriginal, doc=2099)
  2.8274677 = (MATCH) weight(tituloOriginal:retrieval in 2099), product of:
    0.7728124 = queryWeight(tituloOriginal:retrieval), product of:
      7.317345 = idf(docFreq=9, numDocs=5541)
      0.105613776 = queryNorm
    3.6586726 = (MATCH) fieldWeight(tituloOriginal:retrieval in 2099), product of:
      1.0 = tf(termFreq(tituloOriginal:retrieval)=1)
      7.317345 = idf(docFreq=9, numDocs=5541)
      0.5 = fieldNorm(field=tituloOriginal, doc=2099)

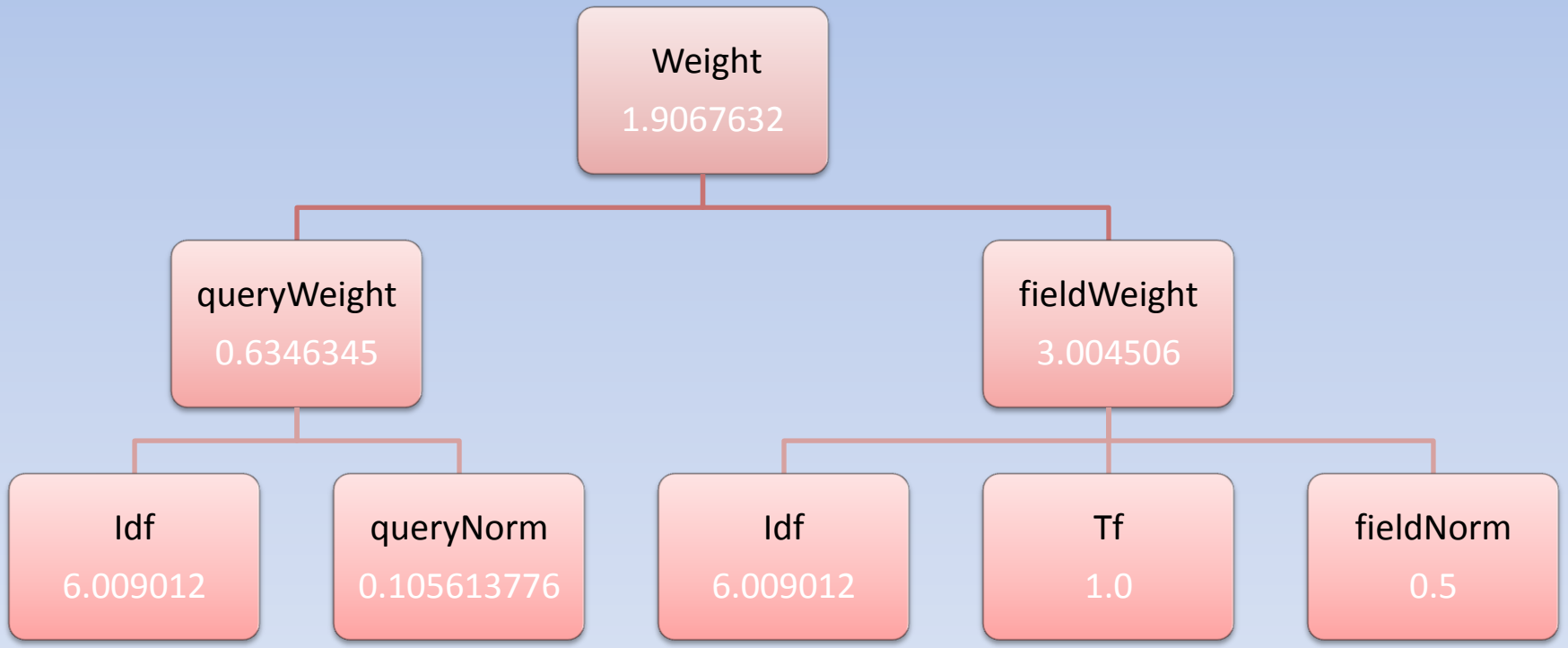
Lucene - Score

Query: tituloOriginal: Information Retrieval
Resultado: **Information analysis and retrieval**



Lucene - Score

Information



Lucene - Score

Query: **Information Retrieval**

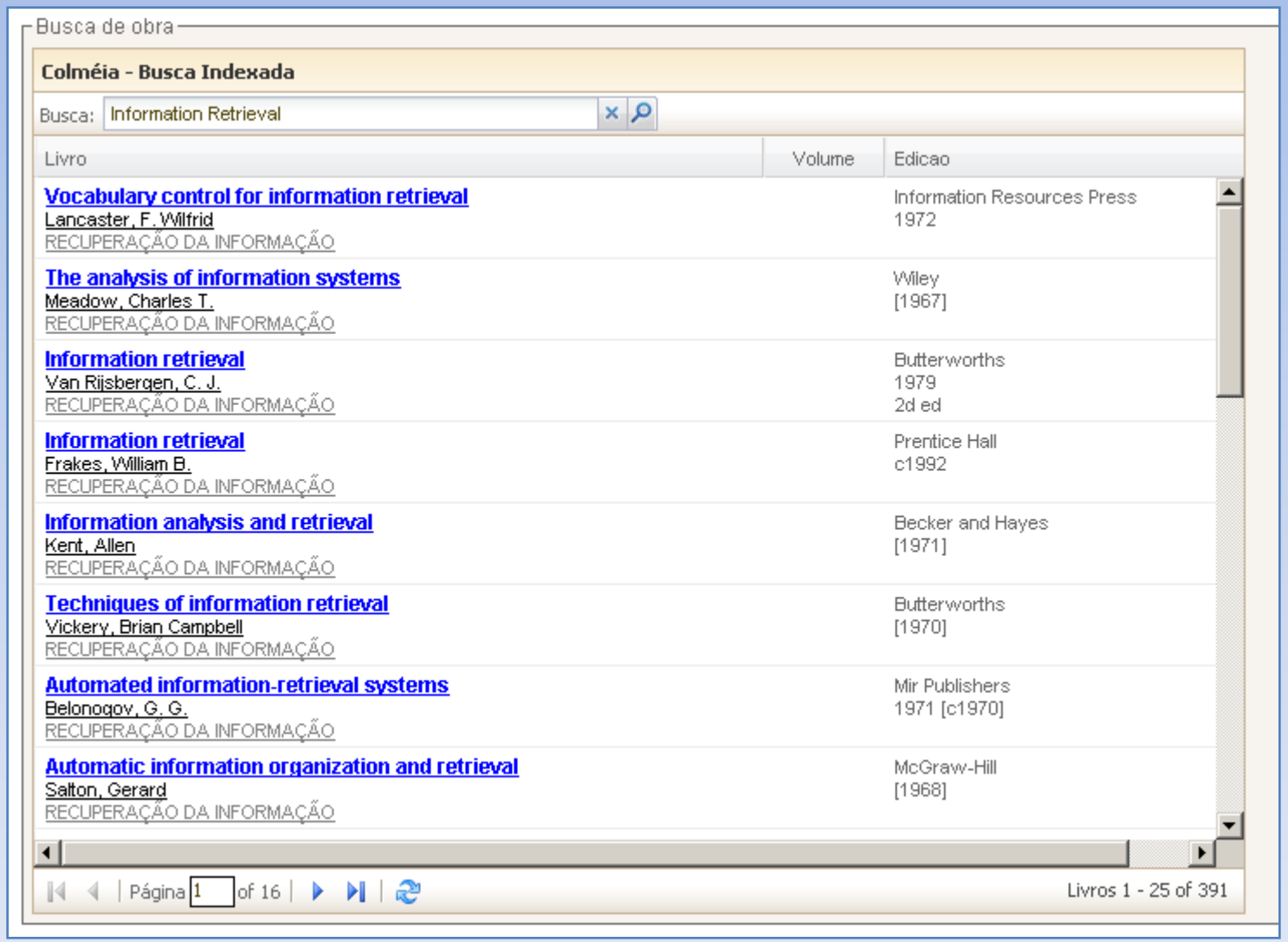
R1: Information analysis and retrieval

R2: Information storage and retrieval: tools, elements, theories

R3: The SMART retrieval system

		Information			Retrieval		
	Score	Weight	Query Weight	Field Weight	Weight	Query Weight	Field Weight
R1	**4.734231**	**1.9067632**	0.6346345	3.004506	**2.8274677**	0.7728124	3.6586726
R2	**3.550673**	**1.4300724**	0.6346345	2.2533796	**2.1206007**	0.7728124	2.7440045
R3	**1.4137338**				**2.8274677**	0.7728124	3.6586726

Resultados


Busca de obra
Colméia - Busca Indexada
Busca: Information Retrieval
Livro
Volume
Edicao
Vocabulary control for information retrieval
Lancaster, F. Wilfrid
RECUPERAÇÃO DA INFORMAÇÃO
Information Resources Press
1972
The analysis of information systems
Meadow, Charles T.
RECUPERAÇÃO DA INFORMAÇÃO
Wiley
[1967]
Information retrieval
Van Rijsbergen, C. J.
RECUPERAÇÃO DA INFORMAÇÃO
Butterworths
1979
2d ed
Information retrieval
Frakes, William B.
RECUPERAÇÃO DA INFORMAÇÃO
Prentice Hall
c1992
Information analysis and retrieval
Kent, Allen
RECUPERAÇÃO DA INFORMAÇÃO
Becker and Hayes
[1971]
Techniques of information retrieval
Vickery, Brian Campbell
RECUPERAÇÃO DA INFORMAÇÃO
Butterworths
[1970]
Automated information-retrieval systems
Belonogov, G. G.
RECUPERAÇÃO DA INFORMAÇÃO
Mir Publishers
1971 [c1970]
Automatic information organization and retrieval
Salton, Gerard
RECUPERAÇÃO DA INFORMAÇÃO
McGraw-Hill
[1968]
Página 1 of 16
Livros 1 - 25 of 391

Resultados

Query: assunto:banco de dados AND titulo:database*

Livros encontrados: 125 livros

Hibernate sem indexação:

media: 89 ms

Hibernate com indexação:

media: 13 ms

Resultados

Query: assunto:matematica*

Livros encontrados: 4763 livros

Hibernate sem indexação:

media: 730 ms

Hibernate com indexação:

media: 265 ms