SỬ DỤNG LUCENE

I. GIỚI THIỆU
II. QUY TRÌNH LẬP CHỈ MỤC
III. QUY TRÌNH TRUY XUẤT

GIỚI THIỆU

1. LUCENE

- Là thư viện mã nguồn mở của Apache.
- Phục vụ nhu cầu xây dựng hệ thống truy xuất thông tin theo mô hình Boolean kết hợp với Vector Space.
- Cho phép xử lý tập tài liệu lớn có khoảng 2 tỉ tài liệu với khoảng 270 tỉ term.
- Được phát triển trên Java (có bản Lucene.NET để dùng trên môi trường .NET).

GIỚI THIỆU

1. LUCENE

- Lucene cung cấp các chức năng của một hệ thống truy xuất thông tin thông qua hai quy trình gồm quy trình lập chỉ mục và quy trình tìm kiếm
- Thư viện lucene áp dụng cho tài liệu là văn bản gồm các file chính như sau (x.y.z là số hiệu phiên bản):
  - lucene-core-x.y.z.jar: chứa các lớp xử lý chung của một hệ thống truy xuất thông tin
  - lucene-analyzers-common-x.y.z.jar: chứa các lớp phân tích tài liệu và truy vấn thành các term.
  - lucene-queryparser-x.y.z.jar: chứa các lớp phân tích input thành truy vấn dưới dạng biểu thức luận lý.

GIỚI THIỆU

2. PYLUCENE

- Là wrapper của Lucene, được sử dụng để thực thi Lucene trong môi trường Python.
- Sử dụng JCC để thực hiện các lời gọi hàm đến thư viện Java Lucene.
- Cần có:
  - Jre để thực thi các lớp trong Java Lucene.
  - JCC để làm cầu nối giữa Python và Java
  - lucene là giao diện trên python của các lớp trong Lucene.
  - Các VisualC++ runtime để thực thi các DLL

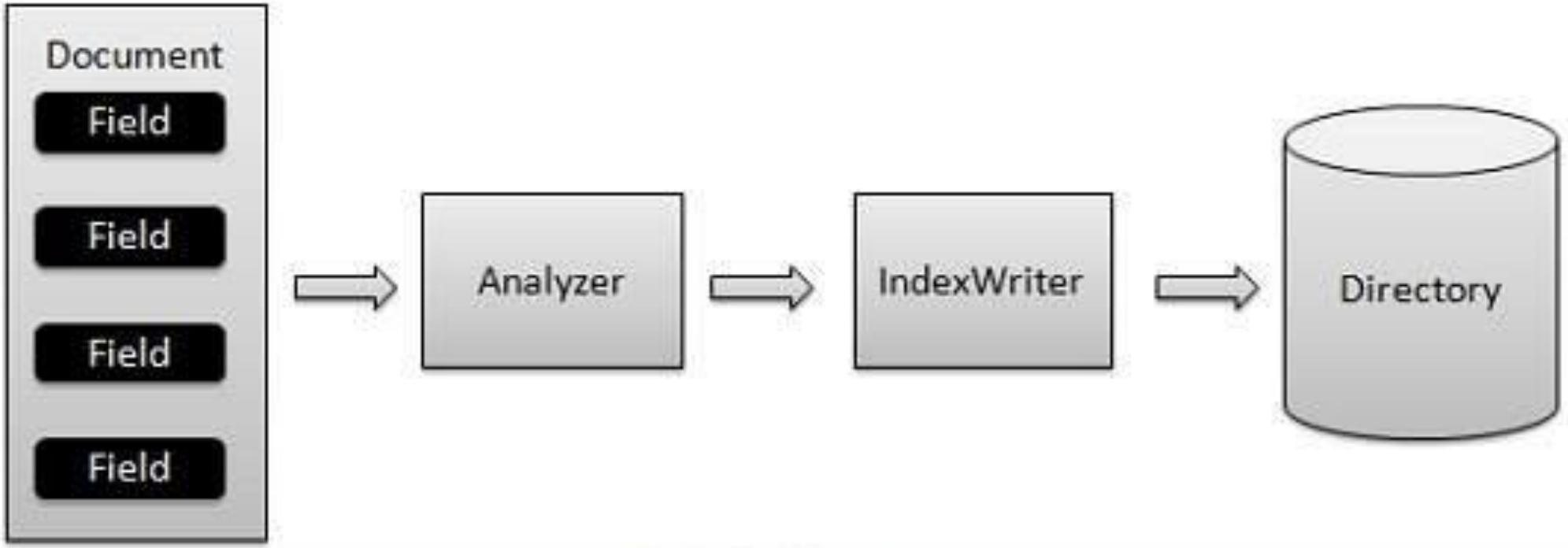
GIỚI THIỆU

2. PYLUCENE

Cài đặt PyLucene:

- Giả sử thư mục chứa JRE là C:\Program Files\jre8. Thêm các đường dẫn sau vào biến môi trường Path của hệ thống:
  - C:\Program Files\jre8\bin
  - C:\Program Files\jre8\bin\server
- Giải nén các thư mục lucene-* và JCC-* vào thư mục Lib\sitepackages của Python
- Cài đặt VC++ runtime cá phiên bản từ 2015 đến 2017

QUY TRÌNH LẬP CHỈ MỤC



QUY TRÌNH LẬP CHỈ MỤC

1. DIRECTORY

- **org.apache.lucene.store**
- Là đối tượng có nhiệm vụ lưu trữ và truy xuất dữ liệu chỉ mục đã được lập khi phân tích tài liệu.
- Có 2 dạng lưu chỉ mục:
  - RAMDirectory: lưu chỉ mục trên bộ nhớ chính. Tạo đối tượng RAMDirectory như sau:

```
RAMDirectory directory = new RAMDirectory();
```

QUY TRÌNH LẬP CHỈ MỤC

1. DIRECTORY

- FSDirectory: lưu chỉ mục trên bộ nhớ ngoài. Tạo đối tượng FSDirectory:

```
FSDirectory directory = FSDirectory.open(Paths.get(f));
```

f (kiểu String) là đường dẫn đến thư mục sẽ lưu dữ liệu chỉ mục (dữ liệu chỉ mục là một hệ thống file có cấu trúc do Lucene định nghĩa để tối ưu việc truy xuất)

Lưu ý: Paths, Path được đặt trong package java.nio.file. Khi sử dụng Paths.get() có thể sinh lỗi không tìm thấy thư mục

QUY TRÌNH LẬP CHỈ MỤC

2. INDEXWRITER

- **org.apache.lucene.index**
- Là đối tượng có nhiệm vụ xử lý các term, tạo từ điển term, tính trọng số term của tài liệu và kết xuất kết quả đến đối tượng DIRECTORY.
- Có hai kiểu xử lý trong IndexWriter là thêm mới và cập nhật. Hai kiểu xử lý này được chỉ định thông qua đối tượng thuộc lớp IndexWriterConfig như sau::

```
IndexWriterConfig cf;
// khởi tạo đối tượng cf;
cf.setOpenMode(openMode);
```

openMode có 3 giá trị: **OpenMode.CREATE** – xóa và tạo mới, **OpenMode.APPEND** – cập nhật, **OpenMode.CREATE_OR_APPEND** – nếu chưa có thì tạo mới, ngược lại cập nhật

QUY TRÌNH LẬP CHỈ MỤC

2. INDEXWRITER

- Tạo đối tượng IndexWriter với các tham số từ đối tượng cf và directory như sau:

```
IndexWriter writer = new IndexWriter(directory, cf);
```

QUY TRÌNH LẬP CHỈ MỤC

3. ANALYZER

- **org.apache.lucene.analysis.core**
- Phân tích tài liệu hoặc truy vấn thành những term theo mong muốn của người xây dựng hệ thống. Đối tượng này cần phải có để tạo đối tượng IndexWriterConfig.
- Được tạo từ những lớp được xây dựng riêng để phù hợp với cách phân tích tài liệu. Đối với tiếng Anh, đối tượng này có thể được tạo từ các lớp như sau:

```
Analyzer analyzer = new WhiteSpaceAnalyzer();
Analyzer analyzer = new KeywordAnalyzer();
Analyzer analyzer = new SimpleAnalyzer();
Analyzer analyzer = new StopAnalyzer();
```

QUY TRÌNH LẬP CHỈ MỤC

3. ANALYZER

- WhiteSpaceAnalyzer: Tách nội dung thành những term dựa trên khoảng trắng.
- KeywordAnalyzer: Toàn bộ nội dung là một term.
- SimpleAnalyzer: Tách nội dung thành những term dựa trên những ký tự không phải chữ cái.
- StopAnalyzer: tương tự như SimpleAnalyzer nhưng loại bỏ những stopword như the, a, an, ...

QUY TRÌNH LẬP CHỈ MỤC

3. ANALYZER

- **org.apache.lucene.analysis.standard**

  Analyzer analyzer = new StandardAnalyzer();
- StandardAnalyzer: tương tự như StopAnalyzer nhưng có nhận dạng các đối tượng đơn giản như email, tên riêng, ….

QUY TRÌNH LẬP CHỈ MỤC

3. ANALYZER

- Từ đối tượng analyzer, tạo đối tượng IndexWriterConfig như sau:

```
IndexWriterConfig cf;
cf = new IndexWriterConfig(analyzer);
```

Sau đó sẽ tạo đối tượng IndexWriter như mục 2.

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- **org.apache.lucene.document**
- Là đối tượng chứa các nội dung được lập chỉ mục.
- Document trong Lucene bao gồm các trường (Field) để phục vụ cho việc tìm theo các tiêu chí khác nhau, chẳng hạn: tìm theo tựa của tài liệu, tìm theo tác giả của tài liệu, tìm theo nội dung tài liệu, ...

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- Field có một số dạng thường sử dụng như sau:
  - TextField: thường được sử dụng để chứa nội dung bằng văn bản phục vụ cho full-text search. Nội dung của TextField sẽ được tách thành term và xử lý bởi đối tượng Analyzer.
  - StringField: chứa văn bản. Nội dung của StringField được xử lý như 1 term (không tách thành term và xử lý bởi đối tượng Analyzer).
  - StoredField: chứa văn bản hoặc số nhưng chỉ dùng để lưu trữ, không dùng để tìm kiếm.
  - Các dạng Field còn lại gồm: IntPoint, LongPoint, FloatPoint, DoublePoint (lập chỉ mục cho số); SortedDocValuesField, SortedSetDocValuesField,...

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- Tạo một số đối tượng Field thường dùng:
  - **Tạo TextField từ chuỗi**

    TextField field = new TextField(name, content, b);

- name (kiểu String): cho biết tên của trường (sẽ dùng khi tìm kiếm)
- content (kiểu String): nội dung của trường.
- b (kiểu boolean): cho biết nội dung chỉ cần được lập chỉ mục (Field.Store.NO) hay còn để lưu trữ (Field.Store.YES).

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- Tạo một số đối tượng Field thường dùng:
  - **Tạo TextField từ đối tượng Reader**

    TextField field = new TextField(name, reader);

- name (kiểu String): cho biết tên của trường (sẽ dùng khi tìm kiếm)
- reader (kiểu Reader): đối tượng đọc nội dung từ file hoặc trang web.

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- Tạo một số đối tượng Field thường dùng:
  - **Tạo StringField**

StringField field = new StringField(name, content, b);

- name (kiểu String): cho biết tên của trường (sẽ dùng khi tìm kiếm)
- content (kiểu String): nội dung của trường.
- b (kiểu boolean): cho biết nội dung chỉ cần được lập chỉ mục (Field.Store.NO) hay còn để lưu trữ (Field.Store.YES).

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- Tạo một số đối tượng Field thường dùng:
  - **Tạo StoredField**

    StoredField field = new StoredField(name, value);

- name (kiểu String): cho biết tên của trường (sẽ dùng khi tìm kiếm)
- value (kiểu bất kỳ như String, int, long, float, ...): nội dung của trường.

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- Để có đối tượng doc là một Document cần:
  - Khởi tạo đối tượng doc:

```
Document doc = new Document();
```

  - Đưa các field cần thiết vào đối tượng doc

```
doc.add(field);
```

(đối tượng field đã được tạo trước)

- Để truy xuất nội dung một trường của đối tượng doc:

```
doc.get(name);
```

  - name là tên trường cần truy xuất nội dung

QUY TRÌNH LẬP CHỈ MỤC

4. DOCUMENT

- Xử lý tài liệu doc bằng cách gọi hành vi addDocument của đối tượng writer (đã được tạo như trong phần IndexWriter):

```
writer.addDocument(doc);
```

- Sau khi xử lý tất cả tài liệu, cần đóng hệ thống chỉ mục lại bằng cách gọi hành vi close của đối tượng writer.

```
writer.close();
```

BÀI TẬP 1

Cho các tài liệu sau:

Tài liệu 1:

- Tựa: Modern Computer Architecture
- Tác giả: James Harrison
- Nội dung: computer architecture, modern RAM, CPU speed, hard drive capacity, easy to use.

Tài liệu 2:

- Tựa: Fashion in Use
- Tác giả: James Smith
- Nội dung: white shirt, hard hat, modern hair styles, easy to work

Tài liệu 3:

- Tựa: Safety in Transportation
- Tác giả: Smith Johnson
- Nội dung: drunk driver, high speed vehicle architectures, carelessly drive, modern designed cars, smith

BÀI TẬP 1

Yêu cầu:

Lập chỉ mục cho các tài liệu trên sao cho:

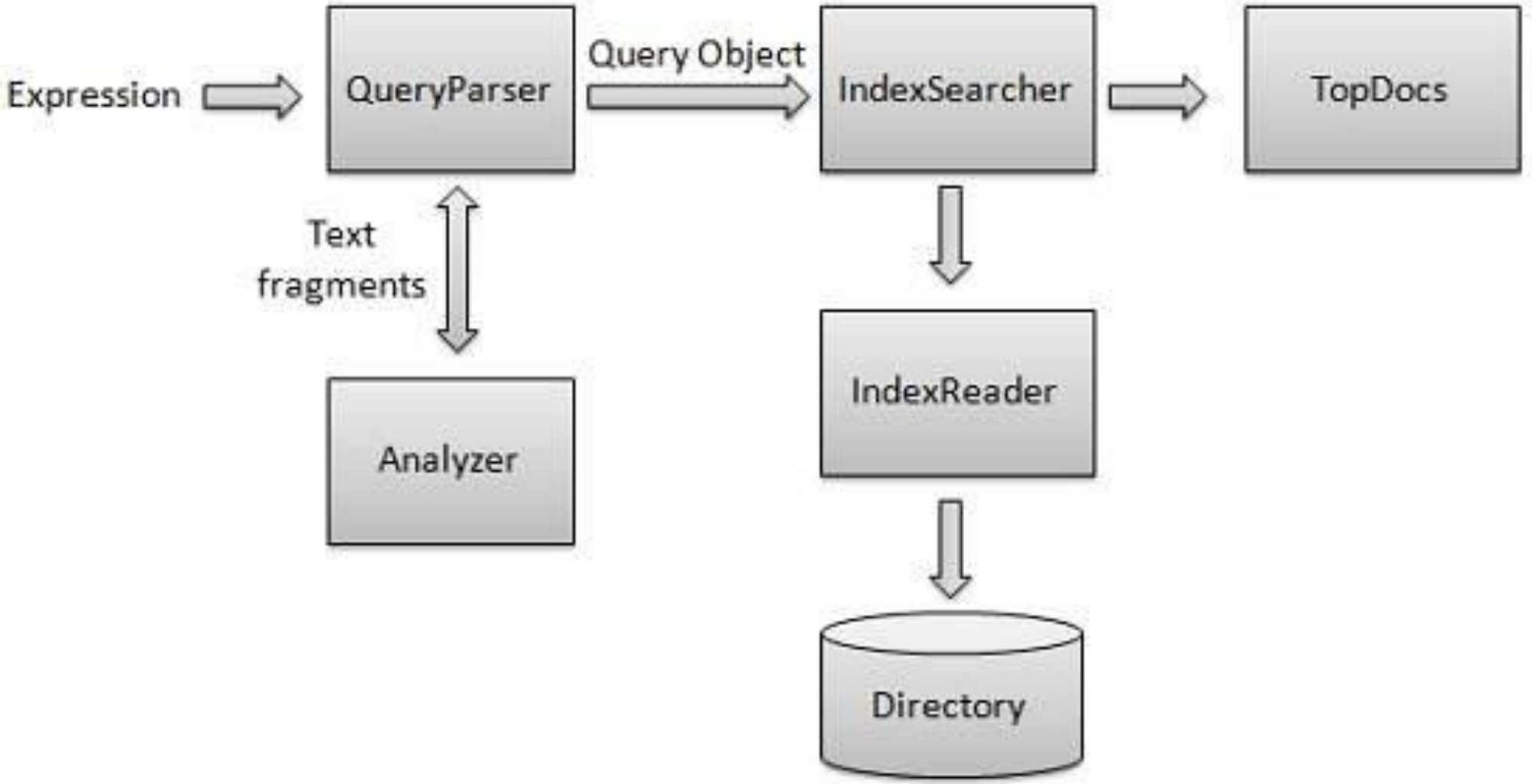
- Cho phép tìm tài liệu theo nội dung và tác giả.
- Kết quả trả về có thể là tựa của tài liệu hoặc tác giả của tài liệu.
- Chỉ mục được lưu trên đĩa cứng

BÀI TẬP 1

Gợi ý:

- Tạo một class tên **LuceneIndexer** với các thuộc tính là các thành phần trong quy trình lập chỉ mục.
- Khởi tạo các đối tượng **Directory**, **IndexWriter**, **Analyzer** trong constructor của lớp
- Viết hàm **index** nhận một tham số là một chuỗi đường dẫn đến thư mục cần lưu chỉ mục. Trong hàm **index** sẽ tạo các **Document** với các **Field** phù hợp và gọi các phương thức của đối tượng thuộc lớp **IndexWriter** để xử lý và lưu hệ thống chỉ mục

QUY TRÌNH TRUY XUẤT



QUY TRÌNH TRUY XUẤT

1. DIRECTORY

- Là đối tượng lưu hệ thống chỉ mục
- Việc tạo đối tượng DIRECTORY tương tự như quy trình lập chỉ mục, trong đó đường dẫn đến thư mục (trong trường hợp dùng FSDirectory) chính là đường dẫn đã dùng khi lập chỉ mục.

QUY TRÌNH TRUY XUẤT

2. INDEXREADER

- **org.apache.lucene.index**
- Là đối tượng có nhiệm vụ truy xuất đến hệ thống chỉ mục để lấy các dữ liệu cần thiết cho việc so khớp và xếp hạng khi tìm kiếm.
- Đối tượng IndexReader được khởi tạo như sau:

```
IndexReader reader = DirectoryReader.open(directory);
```

QUY TRÌNH TRUY XUẤT

3. INDEXSEARCHER

- **org.apache.lucene.search**
- Là đối tượng có nhiệm vụ thực hiện việc so khớp truy vấn với dữ liệu chỉ mục để truy tìm các tài liệu liên quan và xếp hạng chúng.
- Đối tượng IndexSearcher được khởi tạo như sau:

```
IndexSearcher searcher = new IndexSearcher(reader);
```

- Đối tượng searcher của IndexSearcher còn được dùng để truy cập đến tài liệu có chỉ số n trong hệ thống chỉ mục như sau:

```
Document doc = searcher.doc(id);
```

  - id (kiểu int) là chỉ số của tài liệu trong hệ thống

QUY TRÌNH TRUY XUẤT

3. ANALYZER

- Là đối tượng có nhiệm vụ phân tích truy vấn thành các term theo mong muốn của người xây dựng hệ thống.
- Đối tượng này phải được tạo cùng lớp với đối tượng Analyzer trong quy trình lập chỉ mục để đảm bảo việc xử lý tài liệu là thống nhất. (chẳng hạn trong quy trình lập chỉ mục tạo đối tượng thuộc lớp SimpleAnalyzer thì trong quy trình truy xuất cũng phải tạo đối tượng thuộc lớp SimpleAnalyzer)
- Đối tượng thuộc Analyzer được khởi tạo như trong quy trình lập chỉ mục.

QUY TRÌNH TRUY XUẤT

4. QUERYPARSER

- **org.apache.lucene.queryparser.classic**
- Phân tích chuỗi đầu vào mà người dùng đã nhập để tìm kiếm tài liệu thành biểu thức luận lý tương ứng.
- Kết quả xử lý của QueryParser là một đối tượng query thuộc lớp Query. Đối tượng query này mới thực sự được sử dụng để tìm kiếm.
- Đối tượng parser thuộc lớp QueryParser được khởi tạo như sau:

```
QueryParser parser = new QueryParser(name,
                                     analyzer);
```

  - name là tên trường mặc định chứa nội dung cần tìm
  - analyzer là đối tượng được tạo trong mục 3.

QUY TRÌNH TRUY XUẤT

5. QUERY

- **org.apache.lucene.search**
- Đối tượng query thuộc lớp Query được tạo như sau:

Query query = parser.parse(input);

  - input (kiểu String) là nội dung mà người dùng nhập vào để tìm kiếm tài liệu
  - input có cú pháp giống như Google. Chẳng hạn: *inurl:computer +speed*

QUY TRÌNH TRUY XUẤT

6. TOPDOCS

- **org.apache.lucene.search**
- Là đối tượng chứa danh sách các tài liệu tìm được đã được xếp hạng.
- Đối tượng thuộc docs thuộc lớp TopDocs được tạo từ việc thực hiện truy vấn của đối tượng searcher (thuộc lớp IndexSearcher) như sau:

```
TopDocs docs = searcher.search(query, n);
```

  - query (kiểu String) là đối tượng được tạo từ thông tin cần tìm nhờ đối tượng parser trong mục 3
  - n: số lượng tài liệu tối đa cần lấy (nếu muốn lấy tất cả tài liệu tìm được, cần xác định n lớn hơn hoặc bằng số lượng tài liệu liên quan có thể có)

QUY TRÌNH TRUY XUẤT

6. TOPDOCS

- Để truy xuất đến danh sách xếp hạng tài liệu (ScoreDoc) từ đối tượng docs thuộc lớp TopDocs:

```
ScoreDoc score[] = docs.scoreDoc();
```

  - score là một mảng các phần tử có các trường doc (kiểu int) cho biết chỉ số tài liệu, trường score (kiểu float) cho biết giá trị ưu tiên của tài liệu đã được sử dụng để xếp hạng.

BÀI TẬP 2

Yêu cầu:

Từ kết quả lập chỉ mục trong Bài tập 1:

- Tìm kiếm tài liệu theo tên tác giả và trả về danh sách tựa tài liệu
- Tìm kiếm tài liệu theo nội dung và trả về danh sách tựa tài liệu và tác giả tương ứng
- Nhận xét kết quả tìm kiếm theo nội dung với nội dung tìm kiếm:
  - architecture
  - driver
  - carelessly drive
  - modern car

BÀI TẬP 2

Gợi ý:

- Tạo một class tên **LuceneSearcher** với các thuộc tính là các thành phần trong quy trình truy xuất.
- Khởi tạo các đối tượng **Directory**, **IndexReader**, **IndexSearcher**, **QueryParser** trong constructor của lớp
- Viết hàm **searchAuthor** và **searchContent** có một tham số lần lượt là chuỗi chứa tên tác giả và nội dung cần tìm trả về danh sách. Trong hai hàm này sẽ dùng đối tượng thuộc lớp **TopDocs**, **ScoreDoc** và phương thức **doc** của lớp **IndexSearcher** để lấy thông tin theo yêu cầu.

BÀI TẬP 3

Xây dựng ứng dụng cho phép chọn một thư mục chứa các file text để lập chỉ mục cho toàn bộ các file trong thư mục đó. Sau đó, người sử dụng sẽ nhập nội dung cần tìm vào một TextField và hệ thống sẽ tìm trong nội dung các file đó để xác định danh sách các file có liên quan và trả về danh sách tên file trong một List