**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ – ĐHQGHN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

TÌM HIỂU ELASTICSEARCH

-------\*--------

Sinh viên: **Thái Tiến Dũng**

MSSV: 15022052

MỤC LỤC

[1. Mở đầu 3](#_Toc513335260)

[2. Cài đặt 3](#_Toc513335261)

[3. Khái niệm 6](#_Toc513335262)

[3.1. Elasticsearch 6](#_Toc513335263)

[3.1.1. Mục đích và đối tượng sử dụng 7](#_Toc513335264)

[3.1.2. Một số tập đoàn, tổ chức lớn sử dụng ES phổ biến như: 7](#_Toc513335265)

[3.2. Các thuật ngữ 7](#_Toc513335266)

[3.3. Mô hình dữ liệu 8](#_Toc513335274)

[3.3.1. Data field 9](#_Toc513335275)

[3.3.2. JSON 9](#_Toc513335276)

[3.3.3. Data Type 9](#_Toc513335277)

[3.4. Query DSL 9](#_Toc513335278)

[3.4.1. Full text query 9](#_Toc513335279)

[3.4.2. Term level queries 10](#_Toc513335280)

[3.4.3. Compound queries 11](#_Toc513335281)

[3.4.4. Joining queries 11](#_Toc513335282)

[3.4.5. Geo queries 11](#_Toc513335283)

[3.4.6. Specialized queries 11](#_Toc513335284)

[3.4.7. Span queries 12](#_Toc513335285)

[3.5. Indexting, analyzing 12](#_Toc513335286)

[3.5.1. REST API 12](#_Toc513335287)

[3.5.2. Indexing 13](#_Toc513335288)

[3.5.3. Analyzing 13](#_Toc513335289)

[3.6. Phân tán, mở rộng, tăng cường hiệu suất 14](#_Toc513335290)

[3.6.1. Phân tán 14](#_Toc513335291)

[3.6.2. ES hỗ trợ mở rộng theo chiều ngang. 17](#_Toc513335292)

[3.6.3. Tăng hiệu suất 17](#_Toc513335293)

[4. So sánh - Ứng dụng 18](#_Toc513335294)

[4.1. So sánh 18](#_Toc513335295)

[4.2. Ứng dụng 20](#_Toc513335296)

[5. Kết luận 21](#_Toc513335297)

DANH MỤC HÌNH ẢNH

[Figure 1: Download Elasticsearch 5](#_Toc513335988)

[Figure 2: Giải nén Elasticsearch 5](#_Toc513335989)

[Figure 3: Chạy Elasticsearch 6](#_Toc513335990)

[Figure 4: Hoàn thành chạy Elasticsearch 6](#_Toc513335991)

[Figure 5: Kiểm tra Elasticsearch đã cài đặt 7](#_Toc513335992)

[Figure 6: Ví dụ về JSON 10](#_Toc513335993)

[Figure 7: Tìm kiếm thông qua terms 14](#_Toc513335994)

[Figure 8: Các bước xử lý dữ liệu bằng máy xử lý 15](#_Toc513335995)

[Figure 9: Tổ chức dữ liệu Cluster 16](#_Toc513335996)

[Figure 10: Công thức xác định shard 16](#_Toc513335997)

[Figure 11: Quá trình lưu dữ liệu 16](#_Toc513335998)

[Figure 12: Quá trình lấy dữ liệu 17](#_Toc513335999)

[Figure 13: Mở rộng theo chiều ngang 18](#_Toc513336000)

[Figure 14: Đồ thị so sánh xếp hạng Elasticsearch với MySQL 20](#_Toc513336001)

[Figure 15: Search với từ khóa "Học phí" 21](#_Toc513336002)

[Figure 16: Kết quả tìm kiếm của "Học phí" 22](#_Toc513336003)

# **Mở đầu**

Cuộc sống hàng ngày của chúng ta bây giờ, internet là điều không thể thiếu. Cả thế giới đang phát sốt với cuộc cách mạng công nghiệp 4.0, với IOT - Interner vạn vất càng cho thấy tầm quan trọng của nó. Và khi bạn lên bất kỳ một trang web nào, một ứng dụng công nghệ nào bạn đều sẽ nhìn thấy một công cụ tìm kiếm. Giả sử, chúng ta nghe qua một bài hát rất hay, rất cảm xúc nhưng chúng ta vô tình quên mất tên bài hát, chỉ có thể ngân nga 1-2 câu trong vô vàn những câu hát, bài hát tồn tại. Và khi chúng ta muốn tìm lại rất đơn giản. Vào ô search, gõ những gì mình nhớ và nhận được câu trả lời gần như đáp ứng hoàn toàn nhu cầu. Đó là sự tuyệt diệu của Elasticsearch - tìm kiếm nội dung.

Elasticsearch là Search Engine được Shay Banon tạo ra năm 2010 sau khi sản phẩm trước đó là Compass không hỗ trợ được giải pháp tìm kiếm và khả năng mở rộng, Elastic là một giải pháp được xây dựng trên nguyên lý phân tán sử dụng giao diện Json và HTTP. [1]

Báo cáo này được chia làm 2 phần, phần đầu là giới thiệu về Elasticsearch, các khái niệm, kiến trúc hệ thống, mô hình dữ liệu. Các DLS query và Indexing cũng là nội dung của phần một, cuối cùng là điểm mà Elasticsearch có thể tạo ra những ưu việt của nó - Phân tán. Phần 2 là so sánh tính ưu việt của Elasticsearch và thực hiện ứng dụng Elasticsearch.

# **Cài đặt**

Step 1: Cài đặt JDK và JAVA\_HOME

Step 2: Dowload và giải nén

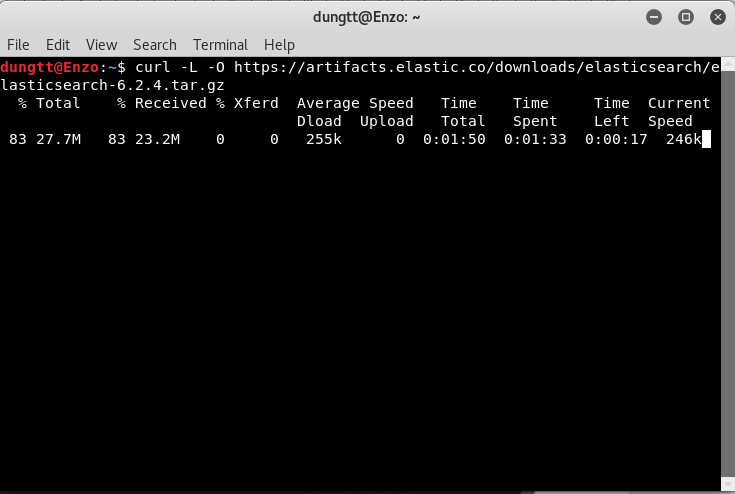


Figure 1: Download Elasticsearch

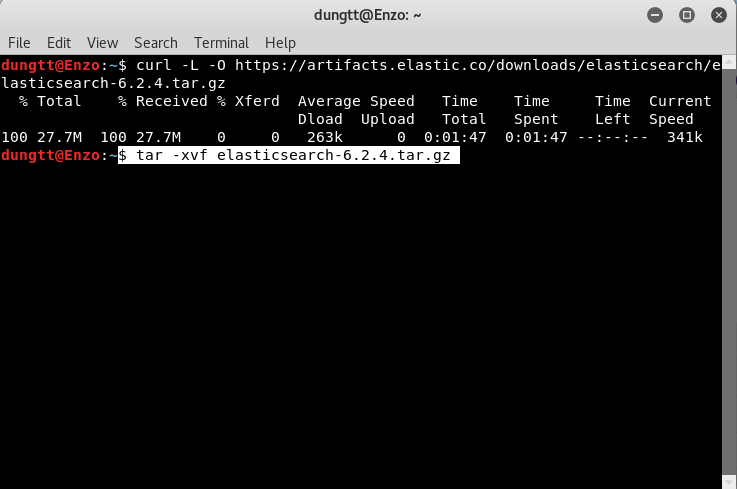


Figure 2: Giải nén Elasticsearch

Step 3: Chạy file elasticsearch trong thư mục elasticsearch/bin

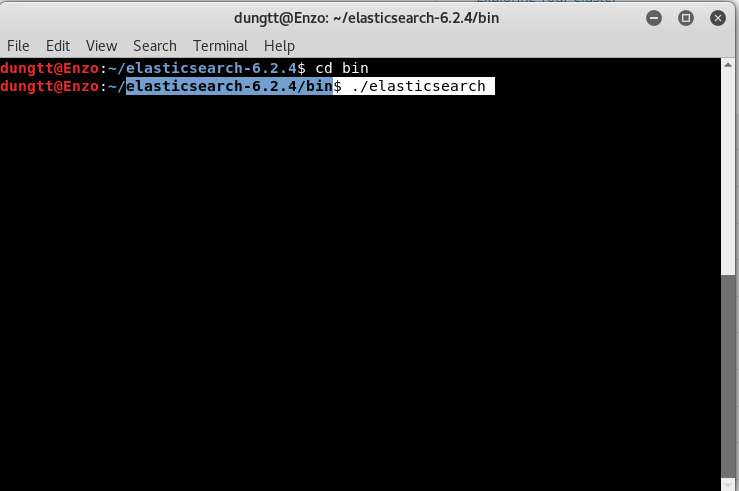


Figure 3: Chạy Elasticsearch

Step 4: Hoàn thành, kiểm tra với http://localhost:9200/

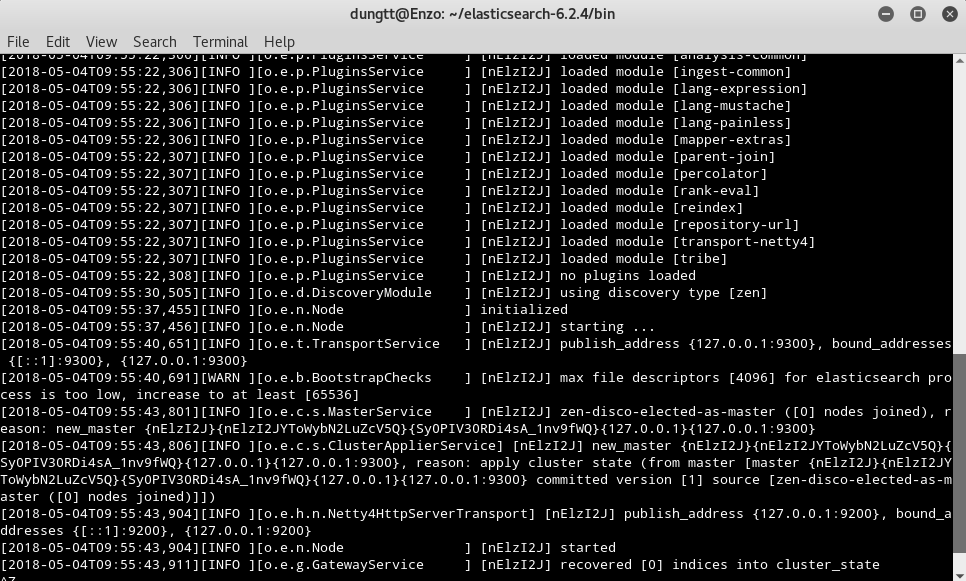


Figure 4: Hoàn thành chạy Elasticsearch

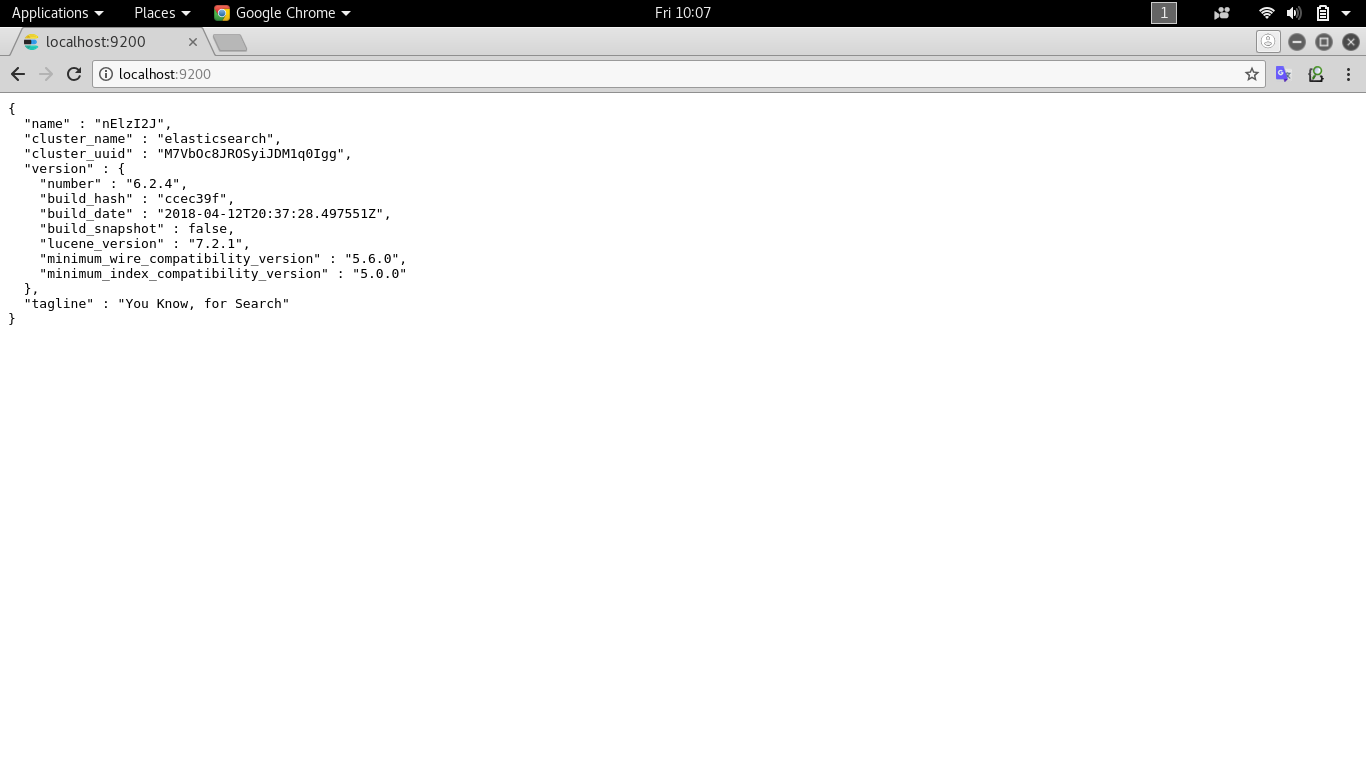


Figure 5: Kiểm tra Elasticsearch đã cài đặt

* Ngoài ra, để có thể hỗ trợ khả năng TEST và truy vấn Elasticsearch ta có thể cài đặt thêm:
  + Kibana <https://www.elastic.co/downloads/kibana>
  + Addon chrome Elasticsearch Head <https://chrome.google.com/webstore/detail/elasticsearch-head/ffmkiejjmecolpfloofpjologoblkegm/>

# **Khái niệm**

## Elasticsearch

Elasticsearch là một search engine kế thừa nền tảng Apache Lucene. Thực chất hoạt động như một web server, có khả năng tìm kiếm nhanh chóng (near realtime) thông qua giao thức RESTful. Là một open-source viết bằng Java Elasticsearch có khả năng phân tích và thống kê dữ liệu, hỗ trợ nhiều ngôn ngữ để có thể tích hợp. Là một hệ thống phân tán với khả năng mở rộng theo chiều ngang cũng như phục hồi dữ liệu. [2]

Là một kho lưu trữ dữ liệu, ES không được sử dụng độc lập trong ứng dụng.

- ES là một phần backend của hệ thống. Ta có hệ thống đầy đủ các chức năng và sử dụng ES để lưu trữ và thao tác với dữ liệu trực tiếp

- ES được tích hợp vào hệ thống có sẵn, các chức năng của hệ thống được xây dựng nhưng cần yêu cầu cao cho việc tìm kiếm và ES được sử dụng ở đấy

- ES được sử dụng với thư viện và công cụ có sẵn. Với các công cụ có sẵn của mình, ES có thể phân tích log của hệ thống mà không cần xây dựng ứng dụng phân tích. [3]

### Mục đích và đối tượng sử dụng

Elasticsearch là một documents oriented databases, do đó nó đóng vai trò chính là một kho chứa và truy xuất dữ liệu. Vậy khi nào ta nên sử dụng Elasticsearch. Không phải với bất kỳ SQL Databases nào cũng nên chuyển đổi sang ES, chúng có mục đích, ưu, nhược điểm khác nhau. Một số trường hợp nên sử dụng ES như:

- Tìm kiếm Text thông thường,

- Tìm kiếm Text và dữ liệu có cấu trúc,

- Tìm kiếm theo tọa độ,

- Tổng hợp dữ liệu,

- Lưu trữ dữ liệu dạng JSON. [2]

### Một số tập đoàn, tổ chức lớn sử dụng ES phổ biến như:

Cisco, eBay, Goldman Sachs, NASA, Microsoft, Mayo Clinic, The New York Times, Wikipedia, Verizon, Adobe System, Facebook, Github, Netflix,... [4]

## Các thuật ngữ

### Near realtime

- Elasticsearch là một nền tảng tìm kiếm nhanh chóng gần như tức thì. Chỉ một độ trễ rất nhỏ từ khi một chỉ mục dữ liệu được lập tới khi nó có thể tìm kiếm được [2]

### Documents

* Là một JSON object với một số dữ liệu,
* Đơn vị nhỏ nhất để lưu trữ dữ liệu, tức thông tin cơ bản có thể lập chỉ mục [2]

### Index

- Một chỉ mục tập hợp tất cả các Documents có cùng những đặc điểm tương tự.

- Một chỉ mục phải được xác định bằng tên bao gồm các chữ thường

- Elasticsearch có thể lưu trữ một hoặc nhiều Index và sử dụng Apache Lucene Library để đọc và ghi dữ liệu. [2]

### Nodes

- Là một máy chủ duy nhất

- Lưu trữ toàn bộ dữ liệu để tiến hành truy vấn, tìm kiếm.

- Có tên được xác định bởi Random by UUID hoặc cũng có thể tự xác định tên. [2]

### Cluster

- Một Cluster bao gồm một hoặc nhiều Node, cùng lưu trữ toàn bộ dữ liệu, cung cấp khả năng lập chỉ mục và tìm kiếm trên tất cả các Node.

- Mỗi Cluster phải được xác định bằng tên duy nhất, mặc định là “elasticsearch”. [2]

### Type

- Tập các Documents được phân đoạn, lập danh mục trong Index được gọi là Type.

- Định nghĩa một trường trong mỗi Type được gị là mapping, nếu trường chưa được định nghĩa thì Elasticsearch sẽ auto mapping type cho trường đó. [2] [1]

### Shards

- Một index có thể có dung lượng lớn vượt qua giới hạn của phần cứng. Do đó, Elasticsearch cung cấp một giải pháp chia nhỏ các index thành các mảnh được gọi là Shards.

- Khi tạo Index có thể xác định được số lượng Shards, mỗi Shards là một index độc lập, có đầy đủ chức năng và có thể được lưu trữ trên bất kỳ node nào.

- Elasticsearch quản lý toàn bộ việc giao tiếp với Shards.

* Primary shards: Dữ liệu lưu ở Primary shards được đánh chỉ số rồi chuyển tới Replica shards, mặc định mỗi Index sẽ có 5 primary shards và khi đã được tạo, số lượng này không thể thay đổi.
* Replica shards: Mặc định mỗi Primary shards có 1 Replica shards, số lượng này có thể được nhân lên bất kỳ lúc nào. Vai trò của Replica shards là đảm bảo toàn vẹn dữ liệu nếu Primary shards gặp sự cố, hơn nữa khi nhiều Replica shards lưu ở nhiều nơi sẽ tăng tốc độ đọc lên. [2]

## Mô hình dữ liệu

Là một Documents oriented databases, ES sử dụng mô hình dữ liệu phẳng. Dữ liệu được tổ chức theo định dạng JSON.

### Data field

* field là đơn vị dữ liệu nhỏ nhất trong ES, mỗi field được xác định type.
* Documents là tập hợp các field.

### JSON

JSON - **JavaScript Object Notation** là định dạng dữ liệu mà ES sử dụng. JSON là một định dạng dữ liệu gồm các cặp Key - Value theo một quy luật mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. [5]

Ví dụ:

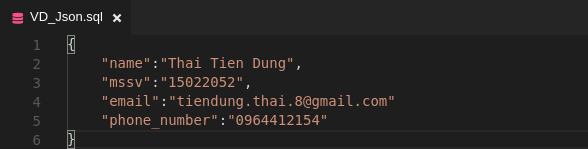


Figure 6: Ví dụ về JSON

### Data Type

- Dữ liệu trong ES sẽ được mapping với 1 type

- Một số type trong Elastic: String, int, long, double, date,...`

## Query DSL

ES cung cấp đầy đủ Query DSL (Domain Specific Language) dựa trên JSON để xác định các truy vấn.các truy vấn. Có 2 loại mệnh đề truy vấn.

- Leaf query clauses: Các mệnh đề truy vấn tìm kiếm một giá trị cụ thể trong một trường hợp cuj thể như: match, term, hoặc range. Các truy vấn này có thể sử dụng độc lập.

- Compound query clauses: Các mệnh đề truy vấn kết hợp các leaf query clauses hoặc các compound query clauses

Mỗi truy vấn là một API request trả về một JSON kết quả. [6]

### Full text query

Là các truy vấn tìm kiếm các đoạn văn bản trên các trường văn bản.

- Match query: Dạng chuẩn full text query, bao gồm các truy vấn cụm từ và kết hợp văn bản mờ hoặc các truy vấn với các cụm từ có mối liên kết với nhau.

- Match\_phrase query: Giống như match query nhưng được sử dụng để đối sánh cụm từ chính xác hoặc kết hợp các từ gần đúng.

- Match\_phrase\_prefix query: Giống như match\_phrase nhưng tìm kiếm theo ký tự đại diện cuối cùng.

- Multi\_match query: Match query với nhiều trường dữ liệu.

- Common\_terms query: Truy vấn chuyên biệt dành cho các từ không phổ biến.

- Query\_string query: Hỗ trợ của Lucene cho phép kết hợp các điều kiện AND, OR, NOT và tìm kiếm nhiều trường cho một chuỗi truy vấn độc lập

- Simple\_query\_string: Truy vấn với cú pháp đoan giản, mạnh mẽ hơn phù hợp để hiển thị trực tiếp cho người dùng. [6]

### Term level queries

Term level query hoạt động trên các cụm từ chính xác được trích xuất từ inverted index và sẽ chuẩn hóa các cụm từ trước khi thức hiện. Các truy vấn này thường được sử dụng cho các dữ liệu cấu trúc như số, ngày tháng hơn là full text. Nó cũng cho phép tạo các truy vấn cấp thấp. Các truy vấn gồm:

- term query: Tìm tài liệu chứa cụm từ chính xác trong trường được chỉ định.

- terms query: Tìm tài liệu chứa bất kỳ cụm từ chính xác nào trong trường được chỉ định.

- terms\_set query: Tìm tài liệu phù hợp với một hoặc nhiều thuật ngữ được chỉ định.

- range query: Tìm tài liệu với giá trị trong phạm vi được chỉ định

- exists query: Tìm tài liệu mà trường được chỉ định chứa bất kỳ giá trị khác null

- prefix query: Tìm tài liệu với trường được chỉ định chứa cụm từ bắt đầu bằng tiền tố chính xác được yêu cầu.

- wildcard query: Tìm tài liệu trong đó trường được chỉ định khớp với mẫu được chỉ định.

- regexp: tìm tài liệu trong đó trường được chỉ định chứa cụm từ khớp với cụm từ thông dụng được chỉ định.

- fuzzy query: Tìm tài liệu trong đó trường được chỉ định chứa cụm từ mở tương tự với cụm từ được chỉ định.

- type query: Tìm tài liệu thuộc type được chỉ định

-ids query: Tìm tài liệu có loại và ID được chỉ định [6]

### Compound queries

* contant\_score query: Tài liệu tìm kiếm được đưa ra cùng một hằng số
* bool query: Truy vấn mặc định cho cách kết hợp nhiều lá hoặc truy vấn hợp chất điều khoản, như must, should, must\_not, hoặc filter.
* dis\_max query: Nhiều truy vấn cùng lúc trả về bất kỳ kết quả của truy vấn nào.
* function\_score query: Sửa đổi trọng số được trả về bởi các hàm bao gồm yếu tố thay đổi
* boosting query: Trả về kết quả phù hợp với mệnh đề truy vấn nhưng giảm trọng số của tài liệu khớp với phủ định truy vấn. [3]

### Joining queries

Thực hiện Join kiểu SQL trong ES rất tốn kém, thay vào đó ES cung cấp hai dạnh Join được thiết kế để mở rộng theo chiều ngang.

- nested query: Tài liệu có thể có thêm trường kiểu nested. Trường này thường dùng để index mảng các object có thể là một query độc lập

- has\_child và has\_parent: Dùng để mô tả quan hệ cha con của 2 tài liệu với chỉ một index. has\_child trả về tài liệu cha và ngược lại với has\_parent. [3]

### Geo queries

ES hỗ trợ 2 kiểu dữ liệu geo: geo\_point trường hỗ trợ cặp lat/lon và geo\_shape hỗ trợ point, line, circles, polygons, multi-polygons

- geo\_shape query: Tìm tài liệu có hình dạng địa lý giao nhau với hình dạng được chỉ định.

- geo\_bounding\_box: Tìm tài liệu thuộc hình chữ nhật chỉ định

- geo\_distance: Tìm tài liệu thuộc đường tròn chỉ định

- geo\_polygon query: Tìm tài liệu thuộc đa giác chỉ định [2]

### Specialized queries

Là nhóm các truy vấn mà không thể từ các nhóm khác

- more\_like\_this query: Tìm tài liệu tương tự như văn bản, tài liệu hoặc bộ sưu tập tài liệu được chỉ định

- script query: Cho phép tập lệnh hoạt động như một bộ lọc.

- percolate query: Tìm các truy vấn được lưu trữ phù hợp

- wrapper query: Chấp nhận các truy vấn khác dưới dạng Json hoặc yaml [2]

### Span queries

Đây là low-level query cho phép can thiệp ở mức sâu hơn vào thứ tự và phân bố của các mệnh đề truy vấn. Nó thường được dùng để thực hiện các truy vấn yếu cầu phải truy vấn chính xác như văn bản pháp luật, bằng sáng chế. Các truy vấn loại này không được để lẫn với các dạng truy vấn khác ngoại trừ khi dùng từ khóa "span\_multi".

- span\_term: Tương đương với term nhưng được dùng trong các

span queries khác.

- span\_multi: Thay cho term,range, prefix, wildcard, regexp, fuzzy.

- span\_first: Hỗ trợ tìm kiếm cho span queries khác với giá trị khớp

phải nằm sau một vị trí cho trước.

- span\_near: Hỗ trợ nhiều span queries mà các kết quả trả về có

khoảng cách với nhau nhỏ hơn một giá trị và có thể cùng thứ tự.

- span\_or: Hỗ trợ nối nhiều queries và trả về kết quả khớp với ít

nhất 1 truy vấn.

- span\_containing: Hỗ trợ danh sách span queries, nhưng chỉ trả

về kết quả mà khớp cả với query thứ 2.

- span\_within: Kết quả trả về của một span query nếu query đấy

nằm trong một danh sách cho trước.

- field\_masking\_span: Hỗ trợ span\_near và span\_or cho những

trường khác nhau. [2]

## Indexting, analyzing

### REST API

ES sử dụng một REST API cho việc tìm kiếm và lưu trữ documents.

Ví dụ:

*$ curl -XPUT 'http://localhost:9200/blog/post/1' -d '{*

*"author": "lucas",*

*"tags": ["java", "web"],*

*"title": "A Fancy Title",*

*"context": "A nice post content..."*

*}'*

Request lấy về một document:

*$ curl -XGET 'http://localhost:9200/blog/post/\_search' -d '{*

*"query" : {*

*"term" : { "author" : "lucas" }*

*}*

*}'*

Khi một query gửi đến ES cluster, query sẽ nhận một trong các node rồi xác định shard nào nên được truy vấn. Sau đó, "query coordinator" node gửi query cho mỗi shard để thự hiện truy vấn song song. Sau khi lần lượt các node trả lời, các truy vấn với kết qủa từng phần, node sẽ merge kết quả và gửi lại cho User. [2]

### Indexing

Khi ES nhận được một êu cầu indexing, nó sẽ chọn 1 shard để index tài liệu, theo mặc định, tài liệu được phân bố đều trên các shard. Khi shard đích được xác định, node sẽ chuyển documents tới node đang giữ shard đó, hoạt đột indexing sẽ được lặp lại trên tất cả các bản sao shard. Indexing hoàn thành khi tất cả các bản sao hoàn thành indexing tài liệu.

Khi lưu trữ document, ES tạo ra inverted index, map các thuật ngữ/từ khóa xuất hiện trong document tới chính document đó.

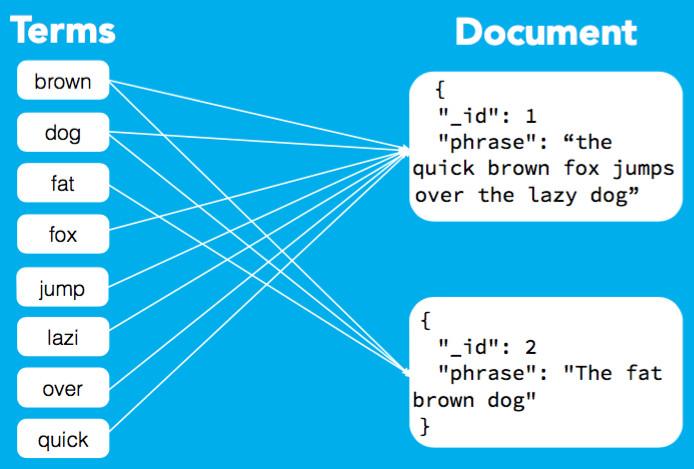


Figure 7: Tìm kiếm thông qua terms

Lúc này, ES có thể tìm kiếm thông qua terms như một binary tree làm giảm thời gian tìm kiếm.

### Analyzing

Phân tích là quá trình chuyển đổi văn bản thành các tokens hoặc terms mà sẽ được thêm vào inverted index cho việc tìm kiếm. Việc phân tích được thực hiện bởi một máy phân tích.

Một máy phân tích là một package chứa 3 khối:

- Character filters: Bộ lọc ký tự chuyển đổi các dòng của văn bản gốc bằng cách thêm, xóa hoặc thay đổi ký tự. Một máy phân tích có thể có nhiều hoặc không có bộ lọc ký tự.

- Tokenizer: Một tokenizer nhận một dòng ký tự và chia thành các thẻ riêng biệt (thường là các từ riêng lẻ) và xuất ra một luông tokens. Nó cũng ghi lại thứ tự hoặc vị trí của từng cụm từ và các ký tự bắt đầu và kết thúc của văn bản gốc. Máy phân tích phải có 1 tokenizer.

- Token filters: Một token filter nhận một luồng token và có thể thêm, xóa và sửa các tokens.

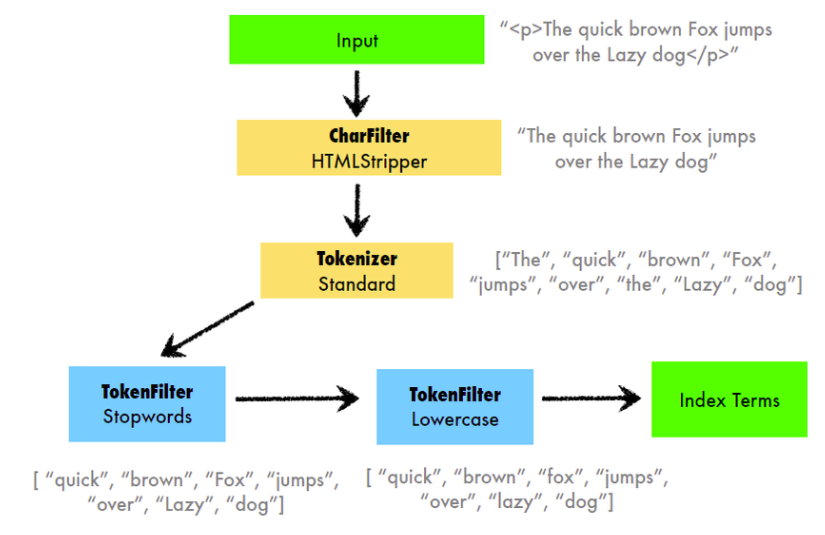


Figure 8: Các bước xử lý dữ liệu bằng máy xử lý

Chúng ta có thể thấy rõ các bước xử lý dữ liệu đầu vào trong elasticsearch như sau

- Loại bỏ thẻ tag

- Tách từ

- Loại bỏ stop word (những từ thường xuyên xuất hiện nhưng không ảnh hưởng nhiều đến kết quả tìm kiếm như *and, then, the, this, that ...* những từ này sẽ có idf nhỏ

- Cuối cùng là đánh index cho từng từ trong nội dung dữ liệu của chúng ta

## Phân tán, mở rộng, tăng cường hiệu suất

### Phân tán

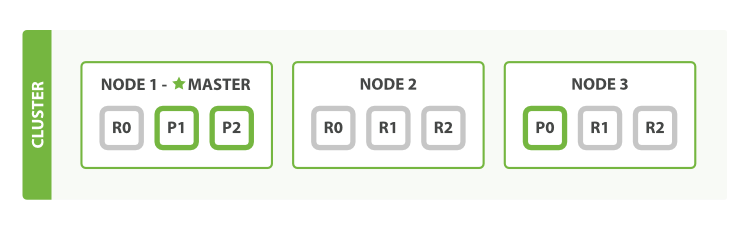
ES lưu dữ liệu trên các shard bao gồm các primary shard và replica shard. Các shard này phân bố đều trên tất cả các node một cách ngẫu nhiên. Replica shard không nằm cùng node với primary shard của nó, điều này đảm bảo khi một node gặp vấn đề, primary shard bị tổn hại thì vẫn có thể sử dụng dữ liệu từ các replica để thay thế, nó cũng cho phép bạn phân chia theo chiều ngang / mở rộng khối lượng nội dung của bạn. Ngoài ra các primary shard không bắt buộc phải nằm trên master node , Nó cho phép bạn phân phối và hoạt động song song trên các phân đoạn (có khả năng trên nhiều node) do đó tăng hiệu suất / thông lượng.

Figure 9: Tổ chức dữ liệu Cluster

Ví dụ: chúng ta có Cluster với 3 node và 3 primary shard, mỗi primary shard có 2 replica shard.

Với một documents mới được thêm vào, nó sẽ được xác định shard để lưu trữ bằng hàm hash:



Figure 10: Công thức xác định shard

Trong đó, hash là hàm tính toán của ES, routing là một "text" đi kèm với document, thường là id, number\_of\_primary\_shards là số lượng primary shard của cluster. Giá trị shard này sẽ đóng vai trò như là id của documents để rounting và xác định shard lưu trữ. Đây cũng là lý do số lượng primary shard luôn phải cố định sau khi khởi tạo.

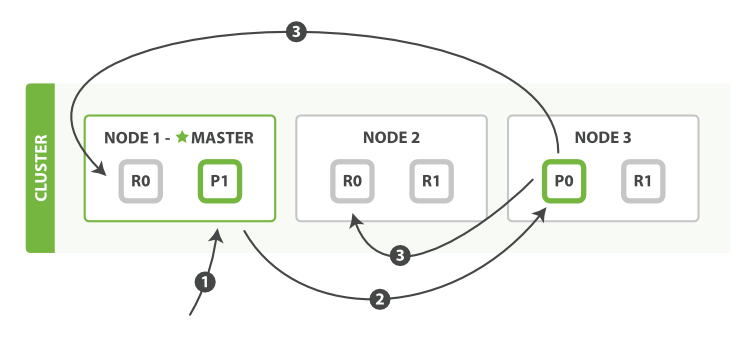


Figure 11: Quá trình lưu dữ liệu

* Quá trình lưu dữ liệu vào ES được chia làm 3 bước như sau:
* Request được gửi đến Master node ( node 1) để tính toán xem đâu sẽ là primary shard lưu trữ. Kết quả là P0.
* Request được gửi đến Node 3 để thực hiện và xử lý dữ liệu.
* Sau khi lưu trữ vào P0, request tiếp tục được gửi đến các node 1, 2 nơi có replication của P0, xử lý và lưu trữ tại đó để đảm bảo đồng nhất dữ liệu.

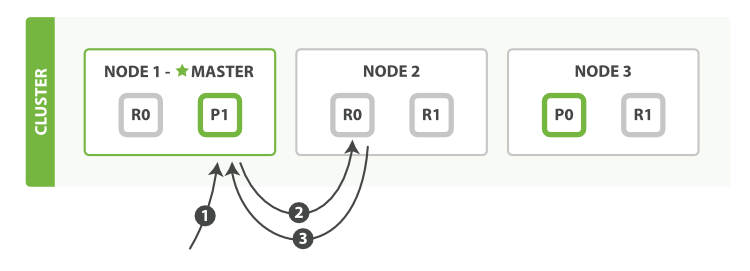


Figure 12: Quá trình lấy dữ liệu

* Quá trình lấy dữ liệu cũng được mô tả trong 3 bước.

- Request được gửi đến Master node, nó xác định được primary shard của document là P0 (bằng id của document)

- Master node sẽ chọn 1 node (node 2) để lấy dữ liệu dựa trên thuật toán Round Robin.

- Dữ liệu sẽ được lấy từ shard tương ứng với primary 0 trả về Master node.

* Quá trình tìm kiếm:
  + Query:
    - Node (trung gian) nhận request và gửi broadcast request đến tất cả các Node khác. Tại mỗi node sẽ chỉ định shard thực hiện tìm kiếm
    - Mỗi shard tìm kiếm trả về id và score của document. Score là giá trị dùng để sắp xếp danh sách kết quả trả về.
    - Node trung gian nhận kết quả trả về từ các node sẽ thực hiện merge kết quả và trả về client.
  + Lấy dữ liệu: Sau khi truy vấn và có được danh sách id, score. Node trung gian lấy nội dung bản ghi dựa vào id đó. [2]

### ES hỗ trợ mở rộng theo chiều ngang.

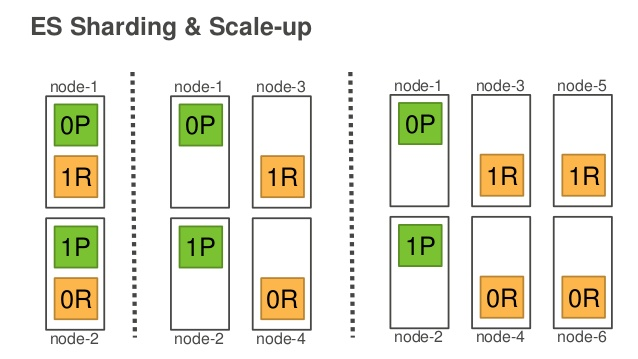


Figure 13: Mở rộng theo chiều ngang

Khi thêm một node vào cluser hiện tại, ES sẽ tự động cân bằng các shard giữa tất cả các node. Điều này giúp tăng tính sẵn có và hiệu năng. Khi các replica enabled, ES sẽ tự động gán 1 replica shard cho 1 primary shard, trong trường hợp không tìm thấy primary shard trong node.

### Tăng hiệu suất

- Để tăng hiệu suất hiệu quả, ES thực hiện gộp các Request. Việc đó giúp tết kiệm thời gian và hạn chế độ trễ gây ra.

- Tối ưu xử lý segment: Mỗi lần ES nhận document từ ứng dụng, nó lập chỉ mục chúng trong RAM và các chỉ mục ngược. Các segments được ghi vào ổ cứng. Các segment nhỏ được tổng hợp một cách định kỳ thành các segment lớn hơn. ES cho phép việc cấu hình linh động cho các thông số liên quan tới xử lý segment để phù hợp với từng trường hợp sử dụng

- Xác định giới hạn thông lượng I/O: Việc tạo các segments lớn có thể gây chậm quá trình lập chỉ mục và tìm kiếm. ES cho phép giới hạn thông lượng I/O để điều tiết quá trình này. [3]

# **So sánh - Ứng dụng**

## So sánh

Để thấy rõ những nổi bật về hiệu năng của ES, ta tiến hành so sánh nó với một RDBMS là MySql. Đầu tiên là một số thông tin cơ bản về 2 DBMS này. [4]

|  |  |  |
| --- | --- | --- |
| Tên | Elasticsearch | MySQL |
| Mô tả | Công cụ phân tích và tìm kiếm hiện đại, phân tán với RESTful dựa trên Apache Lucene | RDBMS mã nguồn mở được sử dụng rộng rãi. |
| Mô hình Cơ sở dữ liệu | Máy tìm kiếm - Store tài liệu | DBMS quan hệ - Store tài liệu, lưu trữ khóa, giá trị |
| Xếp hạng | Điểm: 130.44  Hạng 9 DBMS, hạng 1 Search Engine | Điểm 1223.34  Hạng 2 DBMS, hạng 2 RDBMS |
| Phát triển | Elastic - Java | Oracle - C, C++ |
| License | Open source | Open source |
| Phương pháp phân vùng | Sharding | MySQL cluster, MySQL Fabric |
| Sao chép | Replica | Master - master, Master - slave |

Như đã đề cập, mỗi DBMS đều có một điểm mạnh và mục đích riêng của mình, tuy nhiên, dù ra đời chưa lâu nhưng ES đang dần khẳng định được vị thế của mình trên bảng xếp hạng với tốc độ tăng số lượng người dùng một cách chóng mặt.

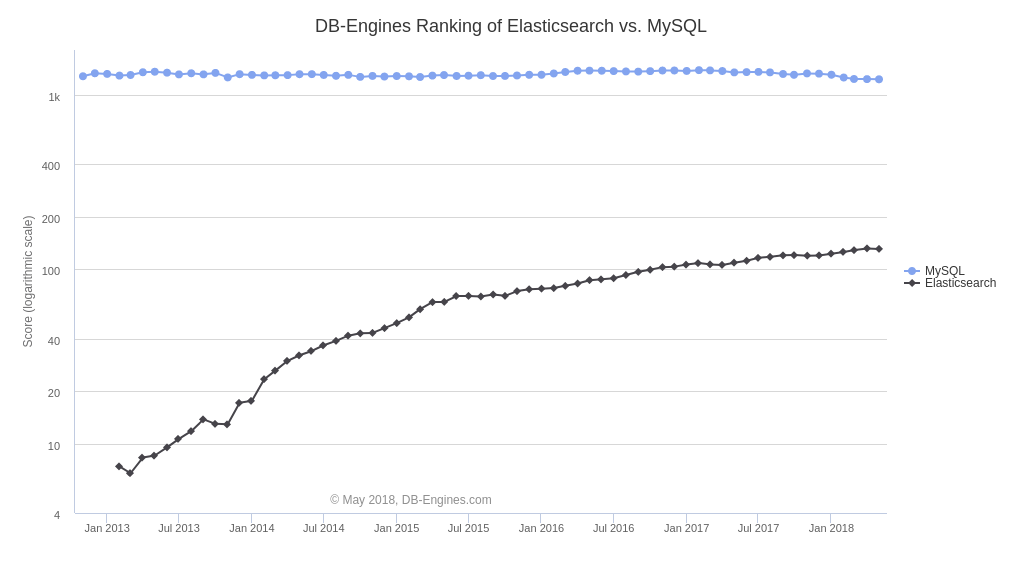


Figure 14: Đồ thị so sánh xếp hạng Elasticsearch với MySQL

Để so sánh về hiệu năng của 2 DBMS, tham khảo kết quả thực nghiệm. Thực nghiệm được tiến hành với ES và MySQL được cài đặt trên Server với cấu hình sau: [3]

CPU : Intel Xeon E5-2690 12 core 24 thread

RAM: 16GB DRR3 STORAGE: HDD 1TB

Bandwidth: 100Mb OS: Ubuntu server 16.04 x64

Dữ liệu test: IMDB - http://www.imdb.com/interfaces

|  |  |  |
| --- | --- | --- |
|  | Elasticsearch | MySQL |
| Đếm số bản ghi | 1ms | 4ms |
| Lấy dữ liệu có điều kiện | 87ms | 118ms |
| Lấy dữ liệu có điều kiện và sắp xếp | 43ms | 8982ms |
| Nối 2 bảng ~ 10 triệu bản ghi | 27ms | 3500ms |
| Nối nhiều bảng đã sắp xếp ~ 100 triệu bản ghi | 27 ms | >3 phút |
| Truy vấn con | 43ms | 3789ms |
| Thêm dữ liệu 1 bản ghi | 5ms | 202ms |

Từ bảng trên ta có thể thấy sự ưu việt về hiệu năng của Elasticsearch so với MySQL. ES nhanh hơn MySQL lớn nhất là gần 7000 lần với việc nối nhiều bảng với xấp xỉ 100 triệu bản ghi. Các truy vấn đơn giản như đếm số bản ghi hay lấy dữ liệu có điều kiện thì sự khác biệt không quá lớn.

## Ứng dụng

Với những kiến thức tìm hiểu được về Elasticsearch. Tôi đã ứng dụng nó vào một project tra cứu thông tin. Cơ sở dữ liệu được tổng hợp sau mỗi câu hỏi chưa có đáp án của người dùng. Link sản phẩm thực tế: [sguet.com](http://sguet.com/)

Công nghệ sử dụng chính:

- Lavarel PHP: Framework để xây dựng hệ thống

- Elasticsearch: Kho lưu trữ và truy vấn dữ liệu

- Jquery và Boostrap

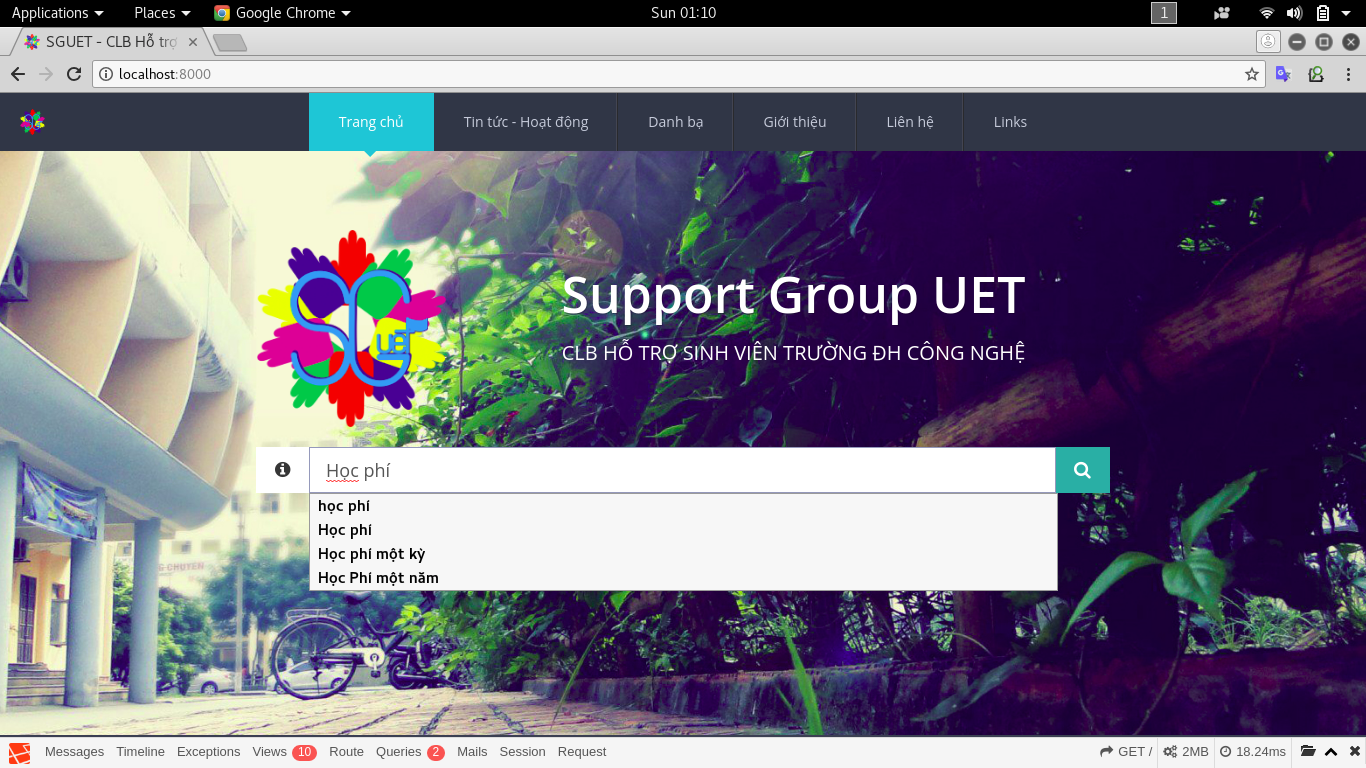


Figure 15: Search với từ khóa "Học phí"

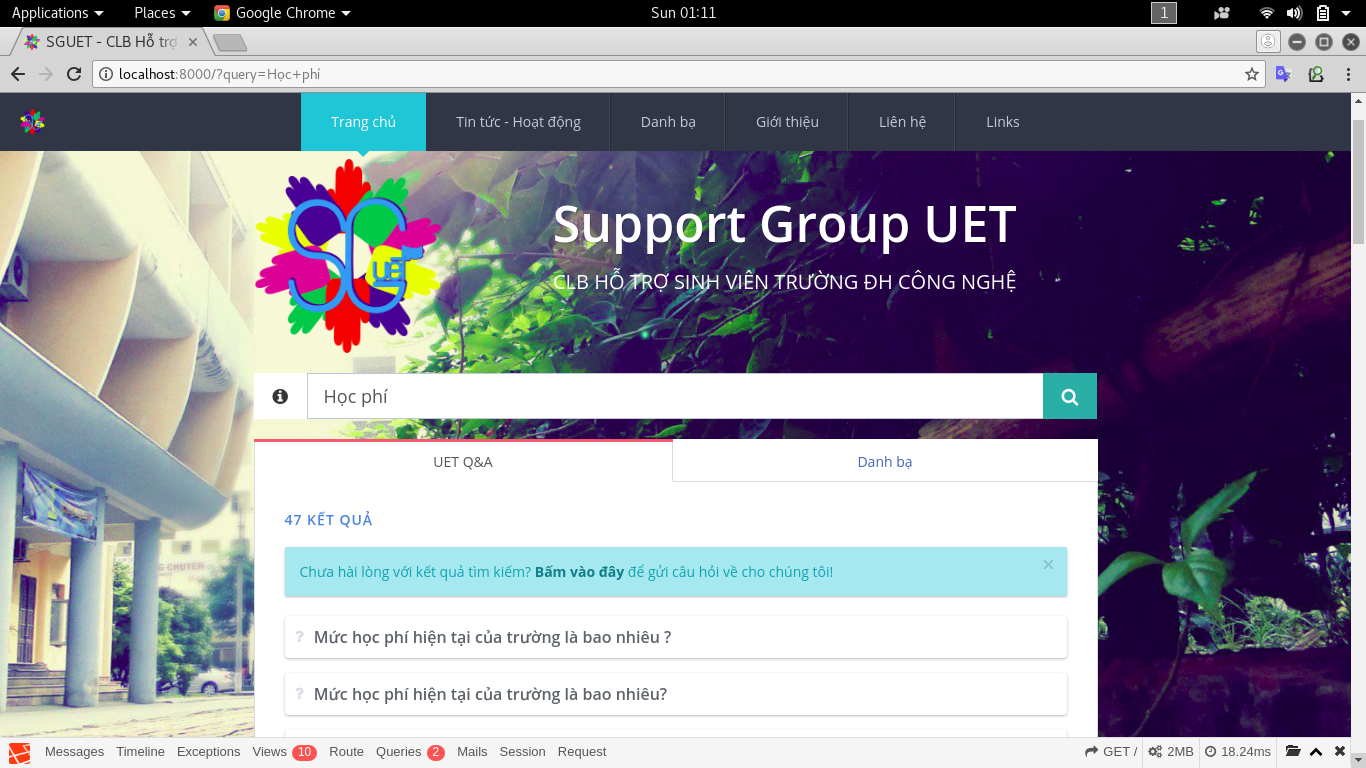


Figure 16: Kết quả tìm kiếm của "Học phí"

# **Kết luận**

* 1. Ưu điểm
* Elasticsearch có tốc độ tìm kiếm tốt
* Có nhiều tài liệu hỗ trợ chi tiết
* Dễ dàng trong việc scaling mở rộng dữ liệu
* Hỗ trợ tìm kiếm "fuzzy" search cho việc mình nhập vào từ có lỗi chính tả hay nhập thiếu một phần từ khóa nhưng vẫn có thể trả ra kết quả mong muốn
* Hỗ trợ cả cơ chế tính điểm đối với kết quả trả về dựa trên tf.idf (term frequent.inverse document frequent)
* Hỗ trợ nhiều loại truy vấn như match query, full text search, fuzzy search
* Phù hợp làm *second data store* cho việc lưu trữ những dữ liệu cần được đánh indexed và tìm kiếm dạng full-text search [7]
  1. Điểm chưa tối ưu
     + Không hỗ trợ bất kỳ cơ chế xác thực hay quyền truy cập nào
     + Không hỗ trợ transaction, là việc khi chúng ta thực hiện thay đổi nhiều bản ghi nếu xảu ra lỗi thì sẽ làm cho logic của mình bị sai hay dẫn tới mất mát dữ liệu
     + Phức tạp hơn khi tích hợp với ứng dụng và chưa có nhiều framework hỗ trợ
     + Không phù hợp với hệ thống có dữ liệu thường xuyên thay đổi, vì thường xuyên thay đổi dẫn tới chi phí bỏ ra đễ đánh chỉ mục là nhiều Data Availability khiến dữ liệu *near realtime* chứ không hẳn *realtime* nếu như mà có yêu cầu cập nhật dữ liệu ngay.
     + Tránh việc sử dụng elasticsearch như là *primary data store* Durability: tính bền vững dữ liệu. Tính bền vững không có độ ưu tiên cao như là các hệ cơ sở dữ liệu khác.
     + Elasticsearch không phải hệ thống ACID (Atomicity, Consistency, Isolation, và Durability)
     + Mất mát dữ liệu, do cơ chế single node partitions trong elasticsearch [7]

Mỗi DBMS đều có điểm mạnh, điểm yếu của nó, lựa chọn DBMS một cách phù hợp với mục đích sử dụng để tối đa hóa sự hiệu quả mà nó mang lại. Công nghệ ngày ngày thay đổi, việc một công nghệ mới được sinh ra giải quyết các bài toán mà các công nghệ cũ đặt ra là điều hiển nhiên. Do đó chúng ta phải biết nắm bắt công nghệ một cách kịp thời, áp dụng chúng vào trong thực tế công việc, cải thiện hiệu suất làm việc. Tuy nhiên việc nắm bắt phải vững vàng dựa trên tính kế thừa từ các công nghệ cũ để không bị lạc lối, đi vào một cách hiểu sai vấn đề hời hợt.

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | S. BANON, The Future of Compass & Elasticsearch, 2010. |
| [2] | "elastic.co," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-gateway.html. |
| [3] | Trần Minh Tuấn, Nguyễn Văn Nhật, Nguyễn Thị Lan, "Tìm hiểu Elasticsearch," Đại học Công nghệ - Đại học Quốc gia Hà Nội, 2017. |
| [4] | "db-engines," [Online]. Available: https://db-engines.com/en/system/Elasticsearch. |
| [5] | w3schools, "w3schools.com," [Online]. Available: https://www.w3schools.com/js/js\_json\_intro.asp. |
| [6] | elastic, "elastic.co," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/\_basic\_concepts.html. |
| [7] | N. Đ. Đức, "Giải pháp xếp hạng và tính toán song song trên nền tảng Apache spark," Đại học Công ngệ - Đại học Quốc gia Hà Nội, 2016. |