Searching có 2 cách:

* Tìm kiếm bằng console của kibana
* Thực hiện thông qua REST API của Elasticsearch

1. Searching thông qua REST API

Một số câu lệnh có thể sử dụng trong việc tìm kiếm:

GET /products/\_doc/100

GET /products/\_search?

GET /products/\_search?q=name:Maker

GET /products/\_search?q=name:tenet

GET /products/\_search?q=name:Maker AND price:65

**Query DSL**

|  |  |
| --- | --- |
| GET /products/\_search {  "query": {  "match": {"in\_stock":10}  }  } | GET /products/\_search {  "query": {  "bool": {  "must": [  { "match": {"in\_stock":10}},  { "match": {"name":"Maker"}}  ]  }  } } |

Trong query DSL được chia làm 2 loại:

* Leaf query: tìm kiếm trên một trường thông tin duy nhất
* Bool query: truy vấn tổng hợp có thể dùng nhiều trường khác nhau

**How searching work**

Như đã trình bày ở trên thì một index sẽ được chia thành các shard và lưu trữ trên nhiều node khác nhau. Vậy nên khi có một query tới thì Elasticsearch không chỉ tìm kiếm trên 1 node mà sẽ tìm kiếm trên nhiều node khác nhau để trả về cho client.

Chính vì vậy mà trong elasticsearch sẽ có 2 port sử dụng với 2 mục đích khác nhau:

* Port 9200: nhận request từ bên ngoài cluster
* Port 9300: trao đổi thông tin giữa các node trong cùng cluster

**Match query**

|  |  |
| --- | --- |
|  |  |
|  |  |

Với query này, elasticseach sử dụng việc analyzer và tokenizer để chia nhỏ văn bản phục vụ cho việc tìm kiếm các document có nội dung trường content bao gồm cả “review” và “index”

Ngoài match query thì elasticsearch còn có multi-match query giúp truy vấn trên nhiều trường khác nhau.

**Term query**

Trong elasticsearch, term query là một loại truy vấn được sử dụng để tìm kiếm các tài liệu dựa trên giá trị cụ thể của một trường (field).

Lưu ý tránh sử dụng truy vấn này cho trường văn bản (text field). Theo mặc định, elasticsearch thay đổi giá trị trường văn bản như một phần của phân tích. Điều này có thể làm cho việc tìm kiếm kết quả khớp chính xác cho trường văn bản trở nên khó khăn. Để tìm kiếm theo trường văn bản thì ta có thể sử dụng match query.

|  |  |
| --- | --- |
| Term query | Match query |
| {  "query": {  "term": {  "title.keyword":"Elasticsearch Guide"  }  }  } | {  "query": {  "match": {  "content": "Elasticsearch Guide"  }  }  } |

**Query và Filter context**

Trong Elasticsearch, query và filter là hai khái nhiệm quan trọng và được sử dụng để thực hiện các tìm kiếm khác nhau trong các ngữ cảnh khác nhau.

Query context:

* Chức năng: truy vấn được sử dụng để tìm kiếm và đánh giá độ phù hợp của tài liệu với một điều kiện cụ thể.
* Đánh giá độ phù hợp: Truy vấn xác định mức độ phù hợp của tài liệu với điều kiện tìm kiếm bằng cách sử dụng các điểm số phù hợp (relevance scores)
* Analyzer và Tokenizer: trong query context, elasticsearch thường được sử dụng Analyzer và Tokenizer để xử lý văn bản, phân tích và chia nhỏ văn bản thành các từ để tìm kiếm.

Filter context:

* Chức năng: Bộ lọc được sử dụng để giới hạn tập hợp các tài liệu dựa trên điều kiện nhất định mà không ảnh hưởng đến độ phù hợp.
* Optimization: Bộ lọc có thể được tối ưu hóa hơn so với truy vấn vì chúng không cần phải tính toán điểm số phù hợp.
* Caching: Bộ lọc có thể được lưu vào bộ nhớ đệm để tái sử dụng, làm giảm tải cho hệ thống.

Trong một truy vấn hoặc bộ lọc có thể tồn tại cả hai ngữ cảnh. Sử dụng kết hợp chúng có thể cung cấp cách linh hoạt để thực hiện các tìm kiếm và giới hạn tập hợp kết quả dựa trên các điều kiện phức tạp.



Trong ví dụ này, truy vấn yêu cầu tìm kiếm các tài liệu có "Elasticsearch Guide" trong tiêu đề và đồng thời giới hạn tập hợp kết quả bằng cách chỉ chấp nhận các tài liệu có giá trị của trường "price" nằm trong khoảng từ 20 đến 50.