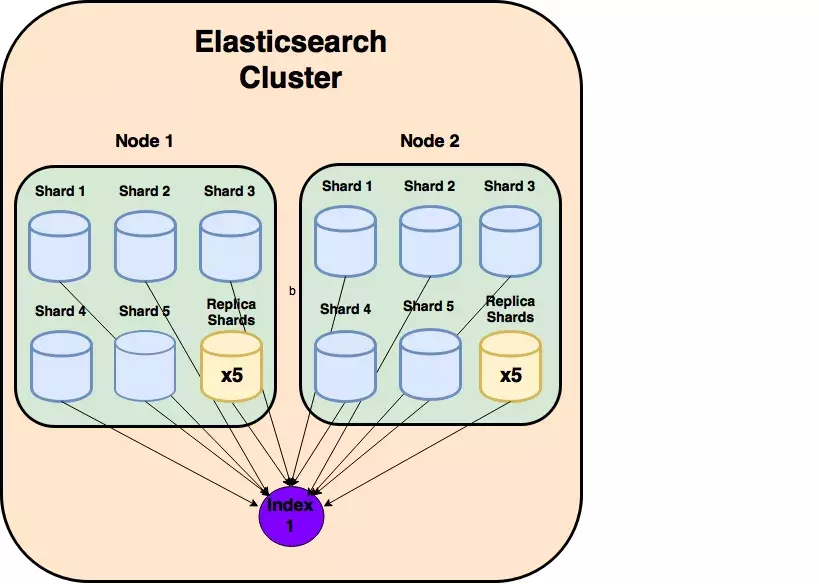
# Khái niện cơ bản

## Thuật ngữ

Elasticsearch như là 1 search engine



### Node

* Là nơi lưu trữ dữ liệu cũng như tham gia vào việc đánh index và thực hiện các thao tác tìm kiếm
* Việc cho phép search cùng lúc trên các shard ở các node khách nhau giúp tăng tốc độ tìm kiếm

### Index

* ĐỊnh danh như 1 table, chính là tập hợp các shard -là tập hợp các document có chung thuộc tính
* ĐỊnh danh bằng tên duy nhất, thực hiện các hoạt động lập chỉ mục, tìm kiếm, cập nhật và xóa các document

### Shard

* Tập hợp các document có chung thuộc tính. 1 shard chính là 1 tập con của index, được chia ra và lưu dữ liệu (document) trên các node bất kỳ. Một node có thể chứa nhiều shard của nhiều index khách nhau
* 2 loại shard

#### Primary Shard

Mặc định mỗi index có 5 primary shard. Dây chính là nơi lưu trữ dũ liệu và được đánh index

#### Replica Shard

Là nơi lưu dữ liệu sao chép của Primary Shard. Mặc định 1 Primary Shard có 1 Replica Shard để đản bảo không bị mất dữ liệu khi xảy ra sự cố. Sẵn sàng thau thế Primary shard khi tìm kiếm bị lỗi

### Document

* Đơn vị lưu trữ dữ liệ cơ bản trong elasticsearch ⬄ tương đương với 1 dòng trong table

### Field

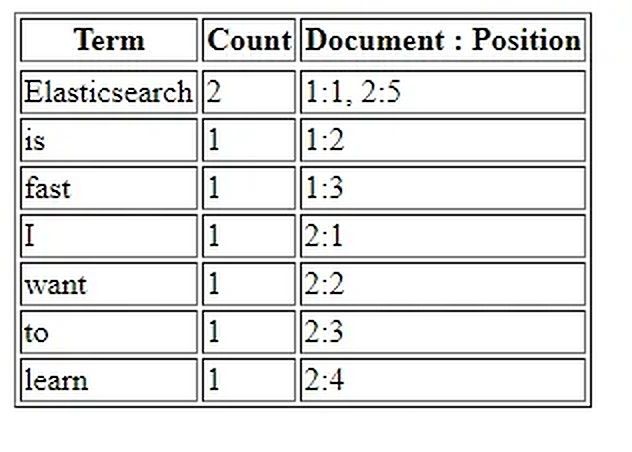
* Tương đương với 1 column trong table

## Inverted Index

Document 1: “Elasticsearch is fast”

Document 2: “I want to learn elasticsearch”

Here is what it would like in the inverted index:



Bây giờ mỗi khi tìm kiếm từ “elastichsearch” thì từ ấy sẽ được tìm kiếm ở trong inverted index

## https://codecurated.com/content/images/2020/11/1_qfZMzJQSKm92K7Qv1eoYew.pngKeyword

* set nó khi cài đặt hoặc tìm kiếm với giá trị trị trường sẽ được phân tích tích theo [Inverted Index](#_Inverted_Index) nhưng với cả cụm dữ liệu chứ không tách từng từ. tìm kiếm như các cơ sở dữ liệu Khác
* ĐIều này sẽ làm tăng hiệu suất trong quá trình truy vấn

### Term

### match

## Text

* Set sẽ đực phân tích như theo [Inverted Index](#_Inverted_Index)

### term

POST /shakespeare/\_search

{

"query": {

"term": {

"text\_entry": {

"value": "The chidden billow seems to pelt the clouds;" }} }

}

{

"query": {

"term": {

"text\_entry": {

"value": "The " }} }

}

Cả 2 truy vấn đều không có kết quả vì

* Cái 1: document đã được phân tích tích theo [Inverted Index](#_Inverted_Index) thành từng từ nên không có 1 term nào thỏa mãn giá trị đàu vào là cả cụm thế kia
* Cái 2: vì document đã được phâ tích theo [Inverted Index](#_Inverted_Index) thì tấ cả các từ sẽ được viết thường

### match

POST /shakespeare/\_search

{

"query": {

"match": {

"text\_entry": {

"query": "The ERROR billow seems to pelt the clouds;" } } }

}

{

"query": {

" match ": {

"text\_entry": {

" query ": "The " }} }

}

* Cả 2 Truy vấn có kết quả :cả query và document được phân tích thành [Inverted Index](#_Inverted_Index) để so khớp với từng từ. Tất cả trở thành viết thường.
* Kết quả trả về chỉ cần khớp ít nhất 1 ký tự là được.
* Sẽ có **\_score** đánh giá độ khớp với các từ trong [Inverted Index](#_Inverted_Index)

# Lệnh cơ bản

## Lệnh show tổng thể thông tin

Get /\_cat/health?v

## Lệnh show các index

Get /\_cat/indices?v

## Tạo index

PUT /<tên index>?pretty

PUT /product?pretty

Pretty cho dễ đọc

## Xóa 1 index

DELETE /<tên index>?pretty

DELETE /product?pretty

## Thêm document

PUT /student/\_doc/<id> {

…

}

-example:

PUT /student/\_doc/3?pretty

{

"name" : "ten3",

"age": 3,

"subject":{

"name": "math",

"year": [2021, 2022]

}

}

## Lấy doc theo id

GET /<tên index>/\_doc/3?pretty

GET /student/\_doc/3?pretty

## Sửa doc theo id

POST /<tên index> /\_doc/<id>?pretty

{

…

}

* example

POST /student/\_doc/2?pretty

{

"name":"ten2 (update)",

"age": 15

}

## Xóa tài liệu

DELETE /<tên index> /\_doc/<id>?pretty

DELETE /student/\_doc/2?pretty

## Tạo mới và cập nhật nhiều doc

POST /\_bulk  
{“index” :{“\_index” : “<tên index>”, “\_id”: <id}}

{…}

{“index” :{“\_index” : “<tên index>”, “\_id”: <id}}

{…}

….

-vd:

POST /\_bulk

{"index": {"\_index": "student", "\_id":4}}

{"name" : "ten4", "age" : 4, "subject" : {"name" : "math", "year" : [2022,2023]}}

{"index": {"\_index": "student", "\_id":5}}

{"name" : "ten5","age" : 5, "subject" : {"name" : "math","year" : [2024,2025]}}

{"index": {"\_index": "student", "\_id":6}}

{"name" : "ten6", "age" : 6,"subject" : {"name" : "math","year" : [2026,2027]}}

# Các dạng truy vấn

## size

* lấy ra số lượng document

{

“query” :{…},

….,

“size” : <số lượng document muốn mấy ra>,

…  
}

## from

* lấy ra bản ghi bắt đầu từ vị trí

{

“query” :{…},

….,

“from” : <lấy từ vị trí thứ >,

…  
}

## sort

* sắp xết bản ghi với field

{

“query” :{…},

….,

“sort” :[

{

“<tên field/\_id/\_score>” : {

“order” : “<desc/asc>”

}

}

,{ “<tên field>” : “<desc/asc”} <// cách 2>

,…

],

…

}

## \_source

* Chỉ định field được lấy ra khi GET

{

…

“\_source”: [“<tên field 1>” ,... ,…]

,…

}

## match , match\_phrase , prefix, term , multi\_match , bool

* truy vấn được sử dụng với dạng giá trị văn bản, số , ngày tháng
* truy vấn áp dụng trực tiếp cho 1 field 1 lần
* Nội dung phân tích được áp dụng
  + “Phong cách” => [“Phong cách”, “Phong”, “cách”]
  + “[Chiminhtoi0112@gmail.com](mailto:Chiminhtoi0112@gmail.com)” => [“[Chiminhtoi0112@gmail.com](mailto:Chiminhtoi0112@gmail.com)”, “Chiminhtoi0112”, “Chiminhtoi”, “0112”, “ gmail.com” , “gmail”, “com”]
* Mẫu:

…{

“<loại so sánh>” :{ “<field>” : “<nội dung>”}

}…

Hoặc:

…{

“<loại so sánh>” :{ “<field>” :

{

“query”:“<nội dung>”,…

}}

}…

Với multi\_match:

…{

“multi\_match” : {

“query” : “<nội dung>”,  
 “field” : [“<tên field 1>,…] / [“\_all”] <// so sánh theo tất cả các trường

….

}

}…

### operate:

* Áp dụng cho nội dung query khi phân tích thành mảng
* Vd ; capital of hungy sẽ được phân tích thành [“capital”, “of”, “hungry"]

Match: {“<tên field>” : { “query” : “capital of hungy”}}

Mặc định operator của match là or có nghĩa là: có câu có thể chỉ chứa 1 trong 3 ý tự-

“capital” OR “of” OR “hungry"

Nếu set Match: {“<tên field>” : { “query” : “capital of hungy”, “operator “: “and”}}

Thì trong kết quả trả về phải trả ra cả 2 giá trị

* Không sử dụng operator trong phrase\_match

### match

* Nội dung phân tích : trả về kết quả tìm kiếm không cần theo thứ tự, operator mặc định là or , có thể set là and
* Vd: mặc định là or

..{

"match": {

"text\_entry":"The desperate"

}

}…

* Dùng với and:

…{

"match": {

"text\_entry": {

“query” : "The desperate",

“operator” : “and”

}

}}…

### match\_phare

* Trả về kết quả tìn kiếm với các term bắt buộc phải có ( = operater (and)) và cùng thứ tự và liên tiếp nhau
* Không dùng với operator

### Term

* Trả về kết ủa truy vấn không sử dụng analyzer
* Không dùng với các trường text
* Với mẫu thay “query” thành “value”
* Mẫu:

…{

“term” :{ “<field>” : “<nội dung>”}

}…

Hoặc:

…{

“term” :{ “<field>” :

{

“value”:“<nội dung>”,…

}}

}…

### multi\_match

* Dùng để search với nhiều trường 1 lúc

…{

“multi\_match” : {

“query” : “<nội dung>”,  
 “field” : [“<tên field 1>,…] / [“\_all”] <// so sánh theo tất cả các trường

“operator” : “or” <// mặc định là or  
 “type” ;’’<type của multi\_match>”

}

}…

#### Type

##### best\_fields

Sử dụng tốt khi muốn tìm kiếm nhiều từ trong 1 field. Ví dự như “đèn ông sao” đi vs nhau trong 1 trường sẽ có nghĩa hơn là mỗi từ ở 1 trường khác nhau

##### most\_fields

Hữu ích khi tìm ở nhiều filed có chứa cùng 1 loại văn bản . Tổng hợp điểm ở các trường khác nhau để sắp xếp

Giống như: GET /\_search

{

"query": {

"bool": {

"should": [

{ "match": { "title": "quick brown fox" }},

{ "match": { "title.original": "quick brown fox" }},

{ "match": { "title.shingles": "quick brown fox" }}

]

}

}

}

## Bool

* Dùng để kết hợp so sánh với nhiều câu truy vấn khác
* Có 4 kiểu sử dụng với boolen

### must

* Tất cả các điều kiện phải trả về true
* Mô tả dạng sql : <điều kiện>--AND-<điều kiện>- AND -<điều kiện>- AND --<điều kiện>
* Mẫu

"bool": {

"must": [

{ <Điều kiện> },

…

]

}…

* Vd:

…"bool": {

"must": [

{ "match": { "speech\_number": 8 } },

{ "term": { "line\_id": 102 } }

] }…

### must\_not

* Tất cả các điều kiện phải trả về true
* Mô tả dạng sql : Not (<điều kiện>--AND-<điều kiện>- AND -<điều kiện>- AND --<điều kiện>)
* Mẫu

"bool": {

"must\_not": [

{ <Điều kiện> },

…

]

}…

* Vd:

…"bool": {

"must\_not": [

{ "match": { "speech\_number": 8 } },

{ "term": { "line\_id": 102 } }

] }…

### should

* Một trong những cái đúng thì trả về đúng
* Mô tả dạng sql : <điều kiện>--OR -<điều kiện>- OR -<điều kiện>- OR --<điều kiện>
* Mẫu

"bool": {

"should": [

{ <Điều kiện> },

…

]

}…

* Vd:

…"bool": {

"must": [

{ "match": { "speech\_number": 8 } },

{ "term": { "line\_id": 102 } }

] }…

#### Should\_not

Thực chất không có should\_not nhưng ta có thể dưa vào tính chất

=

…{“must” : {

}}…

### filter

## Inside Field search

* Các cấu hình lọc với field – đươn vị cấu hình lọc nhỏ nhất
* Mẫu

…{ “<tên field>” :{

“query” : “<nội dung>” <//có thể bỏ “” nếu nội dung là số

< Inside Field search>

}}…

### slop

### zero\_terms\_query

### boost