Querying & Analysis

JP Toto
http://jptoto.jp
@jptoto

Elasticsearch Basic Queries

"Search lite"

```
(GET) http://localhost:9200/my_blog/_search?q=post_text:awesome
```

Elasticsearch Query DSL

```
(GET) http://localhost:9200/my_blog/_search
```

```
(BODY)

{
    "query": {
        "match": {
            "post_text": "awesome"
        }
    }
}
```

Query DSL

Elasticsearch Query DSL

```
{
    "query": {
        "match": {
            "post_text": "awesome"
        }
    }
}
```

Elasticsearch Match Phrase

```
{
    "query": {
        "match_phrase": {
            "post_text": "awesome"
        }
    }
}
```

- Finds exact phrases
- Good for full text searching
- More predictable results

Elasticsearch Query Filters

```
{
    "query": {
        "filtered": {
            "filter": {
                "range": {
                    "post_date": {
                        "gt": "2014-10-18"
                    }
                }
            },
            "query": {
                "match": {
                    "post_text": "awesome"
                }
            }
        }
    }
}
```

- Allows for more complex searching
- Efficiently reduces results

Elasticsearch Highlighting

shard 1 of 4

History [edit]

Shay Banon created Compass in 2004.[1] While thinking about the third version of Compass he realized that it would be necessary to rewrite big parts of Compass to "create a scalable search solution".[1] So he created "a solution built from the ground up to be distributed" and used a common interface, JSON over HTTP, suitable for programming languages other than Java as well.[1] Shay Banon released the first version of Elasticsearch in February 2010.[2]

In June 2014, the company announced raising $70 million in a Series C funding round, just 18 months after forming the company. The round was led by New Enterprise Associates (NEA). Additional funders include Benchmark Capital and Index Ventures. This round brings total funding to $104M.[3]

Shay Banon talking about Elasticsearch at Berlin Buzzwords 2010

Overview [edit]

Elasticsearch can be used to search all kinds of documents. It provides scalable search, has near real-time search, and supports multitenancy.[4] "ElasticSearch is distributed, which means that indices can be divided into shards and each shard can have zero or more replicas. Each node hosts one or more shards, and acts as a coordinator to delegate operations to the correct shard(s). Rebalancing and routing are done automatically [...]".[4]

It uses Lucene and tries to make all features of it available through the JSON and Java API. It supports facetting and percolating,[5] which can be useful for notifying if new documents match for registered queries.

Another feature is called "gateway" and handles the long term persistence of the index;[6] for example, an index can be recovered from the gateway in a case of a server crash. Elasticsearch supports real-time GET requests, which makes it suitable as a NoSQL solution,[7] but it lacks distributed transactions.[8]

Elasticsearch Highlighting

```
{
    "query": {
        "match": {
            "post_text": "awesome"
        }
    },
    "highlight": {
        "fields": {
            "post_text": {}
        }
    }
}
```

- Calls out the matches for you
- Returns html

Elasticsearch Analytics

```
{
    "aggs": {
        "all_words": {
            "terms": {
                "field": "post_text"
            }
        }
    }
}
```

Aggregations

- Like GROUP BY but a lot better
- Can do hierarchical rollups, min/max, percentiles, histograms, calculating the distance between geographical points… TONS of built in aggregations.
- http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/search-aggregations.html

What Is an Elasticsearch Index?

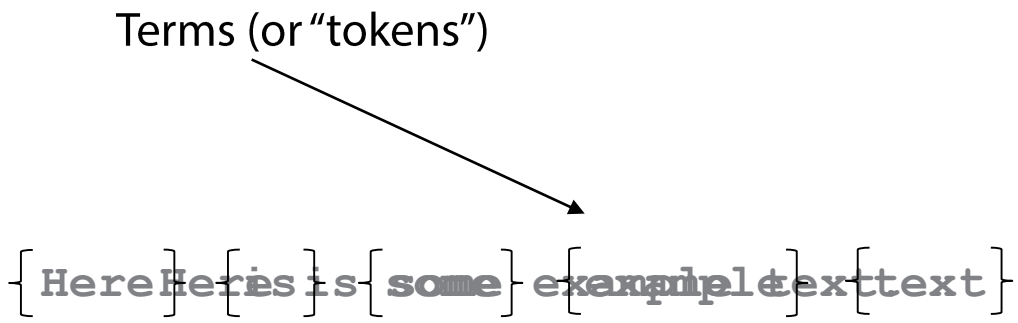
1 `i can not wait to drive my new car`

2 `my new car looks great today`

Term	Document 1	Document 2
i	✓	
can	✓	
not	✓	
looks		✓
to	✓	
today		
my	✓	✓
new	✓	✓
car	✓	✓
wait	✓	
great		✓
drive	✓	

Inverted Index

What Is Analysis?



- Character filter. Remove html, convert "9" to "nine"
- Tokenize by whitespace, period, comma, etc.
- Token filter to remove stop words like "and" or "the"

Elasticsearch Analyzers

- Comes with a lot of built-in analyzers
- You can even make your own

Some example built-in Elasticsearch analyzers:

- standard: Default. Good general purpose analyzer for multiple languages. Breaks up text strings by natural word boundaries, takes away punctuation, and lower-cases terms
- whitespace: Breaks up text by whitespace only. More useful for strings like computer code or logs
- simple: Breaks up strings by anything that isn't a number, Lowercases terms

Elasticsearch Analyzers: Standard

"Convert the title-case text using the ToLower(string) command."

"tokenized"

convert	the	title	case	text	using	the	tolower	string	command

- Dropped hyphenation and parentheses
- Converted to lower case
- Separated by natural word endings

Elasticsearch Analysis: API

Elasticsearch comes with an API for testing analysis.

(GET) http://localhost:9200/_analyze?analyzer=standard

(BODY)
string to be searched

Elasticsearch Analysis: Setting

```
(POST) http://localhost:9200/my_blog

{
    "mappings": {
        "post": {
            "properties": {
                "user_id": {
                    "type": "integer"
                },
                "post_text": {
                    "type": "string",
                    "analyzer": "standard"
                },
                "post_date": {
                    "type": "date"
                }
            }
        }
    }
}
```

Elasticsearch Analysis: Setting

```
"user_id": {
    "type": int,
    "index": "not_analyzed"
}
```

- Whatever is inserted, is what's indexed without any analysis
- Useful for user IDs, states of types using fields like "status"

Elasticsearch Summary

What did we do?

- Learned how to perform basic and complex queries in Elasticsearch
- Talked about how Elasticsearch indexes data
- Figured out how Elasticsearch analyzes data and learned how to choose the best analyzers for our work

What's next?

- Using Elasticsearch with .NET
- NEST and ElasticLINQ