Advanced Search with Elasticsearch

Contents:

What is Elasticsearch

Helping everyone find what they need faster

Elasticsearch

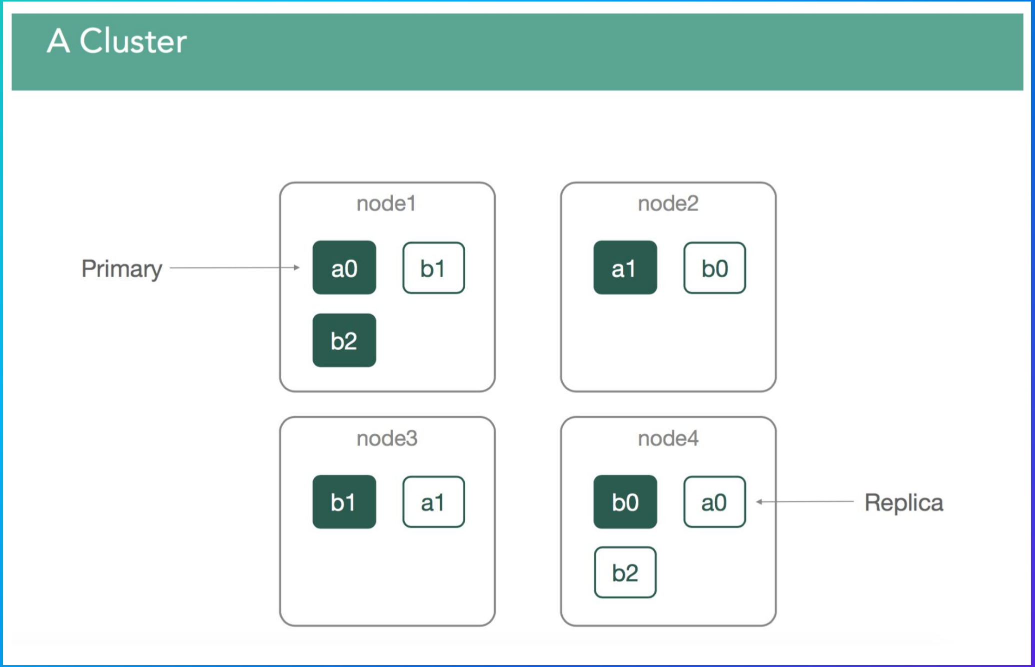
- Built on Apache Lucene
- Distributed
- Scalability
- Free
- Search and analytics engine
- First released in 2010
- Other Tools
  - Kibana
  - Logstash
  - Beats

Types of data

- Textual
- Numerical
- Geospatial
- Structured
- Unstructured

Elasticsearch cluster

- Node
- Primary Shard
- Replica
- Segments
- Inverted index



Text analysis

- *Text analysis* is the process of converting unstructured text, like the body of an email into a structured format that's optimized for search.
- Tokenization - enables matching on individual terms
  - `the quick brown fox jumps`
- Normalization - allows to match tokens that are not exactly the same as the search terms
  - Lowercase
  - Synonyms
  - Stemmed

Built-in analyzers

- Standard Analyzer
  - Divides text into terms, removes most punctuation, lowercases terms and supports removing stop words.
- Simple Analyzer
  - Breaks text into tokens at any non-letter character, discards non-letter characters and changes uppercase to lowercase.
- Whitespace Analyzer
  - Divides text into terms whenever it encounters any whitespace character. It does not lowercase terms.
- Keyword Analyzer
  - Accepts text and outputs the exact same text as a single term.
- Pattern Analyzer
  - uses a regular expression to split the text into terms
- Language Analyzers

2

Full Text Queries

The full text queries can search analyzed text fields such as the body of an email. The query string is processed using the same analyzer that was applied to the field during indexing.

Match Queries

Returns documents that match a provided text, number, date or boolean value. The provided text is analyzed before matching.

Example

```
curl -XGET "http://localhost:9200/book_index/_search"
-H 'Content-Type: application/json' -d'
{
  "query": {
    "match": {
      "chapter": {
        "query": "door closed behind dorian."
      }
    }
  }
}'
```

```
curl -XGET "http://localhost:9200/book_index/_search"
-H 'Content-Type: application/json' -d'
{
  "query": {
    "match": {
      "chapter": {
        "query": "door closed behind dorian",
        "operator": "and"
      }
    }
  }
}'
```

query_string Query Type

- The query string is processed using the same analyzer that was applied to the field during indexing.
- default_field index level setting, which has a default value of *

Example

```
curl -XGET "http://localhost:9200/book_index/_search" -H 'Content-Type: application/json' -d'
{
  "query": {
    "query_string": {
      "query": "door AND closed AND behind AND dorian",
      "default_field": "chapter"
    }
  }
}'

curl -XGET "http://localhost:9200/book_index/_search" -H 'Content-Type: application/json' -d'
{
  "query": {
    "query_string": {
      "query": "door closed behind dorian",
      "default_field": "chapter",
      "default_operator": "and"
    }
  }
}'


GET book_index/_search
{
  "query": {
    "query_string": {
      "query": "door AND closed AND (behind OR dorian)",
      "default_field": "chapter"
    }
  }
}
```

Match Phrase Query

- A phrase query matches terms up to a configurable slop (which defaults to 0) in any order. Transposed terms have a slop of 2. This is also called proximity search

Example

```
curl -XGET "http://localhost:9200/book_index/_search"
-H 'Content-Type: application/json' -d'
{
 "query": {
  "match_phrase": {
   "chapter": {
    "query": "the door closed behind dorian",
    "slop": 3
   }
  }
 }
}'
```

3

Term-level Queries

Term-level queries are used to find documents based on precise values in structured data. Unlike full-text queries, term-level queries do not analyze search terms. Instead, term-level queries match the exact terms stored in a field.

Term/Terms Query

Term query returns documents that contain an exact term in a provided field.

The terms query is the same as the term query, except you can search for multiple values.

Example

```
curl -XGET "http://localhost:9200/book_index/_search"
-H 'Content-Type: application/json' -d'
{
 "query": {
  "term": {
   "chapter": {
    "value": "door"
   }
  }
 }
}'
```

```
curl -XGET "http://localhost:9200/book_index/_search"
-H 'Content-Type: application/json' -d'
{
 "query": {
  "terms": {
   "chapter": [
    "door",
    "closed",
    "behind"
   ]
  }
 }
}'
```

Prefix Query

Returns documents that contain a specific prefix in a provided field.

Example

```
curl -XGET "http://localhost:9200/book_index/_search"
-H 'Content-Type: application/json' -d'
{
  "query": {
    "prefix": {
      "chapter": {
        "value": "tyrianconvol"
      }
    }
  }
}'
```

Range query

Returns documents that contain terms within a provided range. Used for dates field.

Example

```
GET data_index/_search
{
  "query": {
    "range": {
      "date": {
        "gte": "now-10d",
        "lte": "now-3d"
      }
    }
  }
}

GET data_index/_search
{
  "query": {
    "range": {
      "date": {
        "gte": "1656655078715",
        "lte": "1656655078717"
      }
    }
  }
}
```

Exists query

Returns documents that contain an indexed value for a field. Useful for finding documents without values for some field.

Example

```
GET data_index/_search
{
 "query": {
  "bool": {
   "must_not": {
    "exists": {
     "field": "area_code"
    }
   }
  }
 }
}
```

Wildcard query

The * wildcard operator matches zero or more characters

Example

```
GET book_index/_search
{
 "query": {
   "wildcard": {
     "chapter": {
       "value": "tyrian*lvulus"
     }
   }
 }
}
```

Fuzzy query

Returns documents that contain terms similar to the search term, as measured by a Levenshtein edit distance.

The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

These changes can include:

- Changing a character
- Removing a character
- Inserting a character
- Transposing two adjacent characters

Example

```
GET book_index/_search
{
 "query": {
  "wildcard": {
   "chapter": {
    "value": "tyrian*lvulus"
   }
  }
 }
}
```

4

Span Queries

Span query is typically used to implement very specific queries on legal documents.

Span Near Query

Matches spans which are near one another. One can specify *slop*, the maximum number of intervening unmatched positions, as well as whether matches are required to be in-order.

Example

```
GET book_index/_search
{
 "query": {
  "span_near": {
   "clauses": [
    {
     "span_term": {
      "chapter": "men"
     }
    },
    {
     "span_term": {
      "chapter": "and"
     }
    },
    {
     "span_term": {
      "chapter": "women"
     }
    }
   ],
   "slop": 4,
   "in_order": true
  }
 }
}
```

Span Not Query

The `include` and `exclude` clauses can be any span type query. The `include` clause is the span query whose matches are filtered, and the `exclude` clause is the span query whose matches must not overlap those returned.

Example

```
GET book_index/_search
{
 "query": {
  "span_not": {
   "include": {
    "span_near": {
     "clauses": [
      {
       "span_term": {
        "chapter": "reached"
       }
      },
      {
       "span_term": {
        "chapter": "library"
       }
      }
     ],
     "slop": 1,
     "in_order": true
    }
   },
   "exclude": {
    "span_term": {
     "chapter": "a"
    }
   }
  }
 }
}
```

5

Geo Queries

Elasticsearch supports two types of geo data: geo_point fields which support lat/lon pairs, and geo_shape fields, which support points, lines, circles, polygons, multi-polygons, etc.

Geo-bounding box query

Matches geo_point and geo_shape values that intersect a bounding box.

Example

```
GET data_index/_search
{
  "size": 40,
  "query": {
    "bool": {
      "must": {
        "term": {
          "city": {
            "value": "New Jersey Bmc"
          }
        }
      },
      "filter": {
        "geo_bounding_box": {
          "location": {
            "top_left": {
              "lat": 40.73,
              "lon": -74.1
            },
            "bottom_right": {
              "lat": 40.01,
              "lon": -73.12
            }
          }
        }
      }
    }
  }
}
```

Geo-distance query

Matches geo_point and geo_shape values within a given distance of a geopoint.

Example

```
GET data_index/_search
{
 "query": {
  "bool": {
   "must": {
    "match_all": {}
   },
   "filter": {
    "geo_distance": {
     "distance": "3km",
     "location": {
      "lat": 40.73,
      "lon": -74.1
     }
    }
   }
  }
 }
}
```

Geo-shape query

The geo_shape query uses the same grid square representation as the geo_shape mapping to find documents that have a shape that is related to the query shape, using a specified spatial relationship: either intersects, contained, within or disjoint. It will also use the same Prefix Tree configuration as defined for the field mapping.

GeoJSON Types

- `Point`
- `LineString`
- `Polygon`
- `MultiPoint`
- `MultiLineString`
- `MultiPolygon`

Example

```
GET data_index/_search
{
 "query": {
  "bool": {
   "must": {
    "match_all": {}
   },
   "filter": {
    "geo_shape": {
     "location": {
      "shape": {
       "type": "Polygon",
       "coordinates": [
        [
         [
          -74.168701171875,
          41.14143302653628
         ],
         [
          -74.6136474609375,
          40.697299008636755
         ],
         [
          -74.0753173828125,
          40.509622849596695
         ],
         [
          -73.114013671875,
          40.46784549077255
         ],
         [
          -73.333740234375,
          40.896905775860006
         ],
         [
          -74.168701171875,
          41.14143302653628
         ]
        ]
       ]
      },
      "relation": "within"
     }
    }
   }
  }
 }
}
```

LINKS

Simple app for indexing data

- https://github.com/stevicaum/qil-demo

How to install Elasticsearch and Kibana

- https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html

Query documentation

- https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

Contact

Stevica Arsic

https://www.linkedin.com/in/stevica-arsic-99b4a4139/