Demystifying the bool query in Elasticsearch

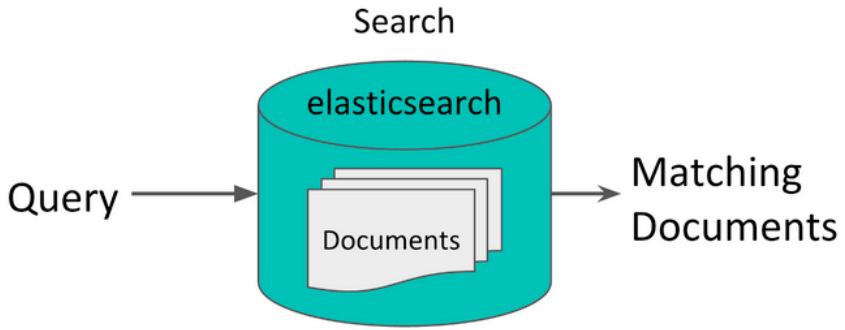
Jay Chow

Software Engineer, Cisco eCommerce, IT

Cisco, San Jose

11th May 2020

Agenda



Source: https://underthehood.meltwater.com

1. Elasticsearch and bool query
2. Must clause
3. Must not clause
4. Should clause
5. Filter clause
6. Real world scenario
7. Reference

ElasticSearch and bool query

- ElasticSearch is one of the most popular and leading industry search engine
- A bool query is a query that matches documents matching boolean combinations of other queries
- The bool query maps to Lucene BooleanQuery; it is built using one or more boolean clauses, each clause with a typed occurrence
- 4 kinds of occurrences: must, should, must_not, filter

Must clause

- The ***must*** clause (query) must appear in matching documents, and its functionality mimics the boolean “AND”.The highest priority for the clause is to score the documents
- For instance, you are searching for “Audi A6" model car and do not want to see other cars in the search result.

```
{
  "query" : {
    "bool" : {
      "must": [{
          "match": {
              "make": "Audi"
          }
      }, {
          "match": {
              "model": "A6"
          }
      }]
    }
  }
}
```

Must not clause

- In the must_not clause, query must not appear in the matching documents.
- For example, if we do not want the car make to consist of Toyota:

```
{
  "query" : {
    "bool" : {
      "must_not": [{
          "match": {
              "make": "Toyota"
          }
      }]
    }
  }
}
```

Should clause

- Should corresponds to the Boolean “OR”
- For example, if you want to search for a car make of either “Audi” or “Kia”:

```
{
  "query" : {
    "bool" : {
      "should": [{
          "match": {
              "make": "Audi"
          }
      }, {
          "match": {
              "make": "Kia"
          }
      }]
    }
  }
}
```

Filter clause

- It is similar to the must clause, if a *filter* clause is used, then the query must also appear in the matching documents, but does not contribute to the score
- For instance, if you want to filter cars that have an ANCAP safety rating of 5:

```
{
  "query" : {
    "bool" : {
      "should": [{
          "match": {
              "make": "Audi"
          }
      }, {
          "match": {
              "make": "Kia"
          }
      }],
      "filter" : {
          "term": {
              "ancap_rating": "5"
           }
       }
    }
  }
}
```

Real world scenario

- If you want to search used “Audi”, “Kia” and “ Toyota” for models “A6”, “Camry” and “Optima”:

```
{
    "query": {
        "bool": {
            "must": [{
                "bool": {
                    "should": [{
                        "match": {
                            "make": "Audi"
                        }
                    }, {
                        "match": {
                            "make": "Kia"
                        }
                    }, {
                        "match": {
                            "make": "Toyota"
                        }
                    }]
                }
            }, {
                "bool": {
                    "should": [{
                        "match": {
                            "model": "A6"
                        }
                    }, {
                        "match": {
                            "model": "Camry"
                        }
                    }, {
                        "match": {
                            "model": "Optima"
                        }
                    }]
                }
            }, {
                "bool": {
                    "must": [{
                        "match": {
                            "condition": "used"
                        }
                    }]
                }
            }]
        }
    }
}
```

Reference

https://engineering.carsguide.com.au/elasticsearch-demystifying-the-bool-query-11da737a4efb