Querying with boolean logic

Query

```
GET /products/_search
{
  "query": {
    "bool": {
      "must": [
        { "term": { "tags.keyword": "Alcohol" } } 🔵
      ],
      "must_not": [
        { "term": { "tags.keyword": "Wine" } }
      ],
      "should": [
        { "term": { "tags.keyword": "Beer" } }, 🔴
        { "match": { "name": "beer" } }, 🟡
        { "match": { "description": "beer" } } 🟢
      ]
    }
  }
}
```

Document #1

```
{
  "name": "Beer - Corona",
  "tags": [
    "Alcohol",
    "Beverage",
    "Beer"
  ]
}
```

🔵 1.9903715	13.395043
🔴 5.450562	
🟡 5.9541097	
🟢 N/A	

Document #2

```
{
  "name": "Heineken",
  "tags": [
    "Alcohol",
    "Beverage",
    "Beer"
  ]
}
```

🔵 2.067343	6.827667
🔴 4.760324	
🟡 N/A	
🟢 N/A	

Document #3

```
{
  "name": "Schnappes - Peach Walkers",
  "tags": [
    "Alcohol",
    "Beverage"
  ]
}
```

🔵 2.067343	2.067343
🔴 N/A	
🟡 N/A	
🟢 N/A	

CodingExplained
CODE. SLEEP. REPEAT.

Important things about `should`

- If a `bool` query only contains `should` clauses, **at least one must match**
- Useful if you just want *something* to match and reward matching documents
    - If nothing were required to match, we would get irrelevant results
- If a query clause exists for `must`, `must_not`, or `filter`, no `should` clause is required to match
    - Any `should` clauses are only used to boost relevance scores

CodingExplained
CODE. SLEEP. REPEAT.

One `should` clause must match

```
GET /products/_search
{
  "query": {
    "bool": {
      "should": [
        {
          "term": {
            "tags.keyword": "Beer"
          }
        },
        {
          "match": {
            "name": "beer"
          }
        }
      ]
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

`should` clauses are optional

```
GET /products/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "tags.keyword": "Alcohol"
          }
        }
      ],
      "should": [
        {
          "term": {
            "tags.keyword": "Beer"
          }
        },
        {
          "match": {
            "name": "beer"
          }
        }
      ]
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

`minimum_should_match` parameter

```
GET /products/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "tags.keyword": "Alcohol"
          }
        }
      ],
      "should": [
        {
          "term": {
            "tags.keyword": "Beer"
          }
        },
        {
          "match": {
            "name": "beer"
          }
        }
      ],
      "minimum_should_match": 1
    }
  }
}
```

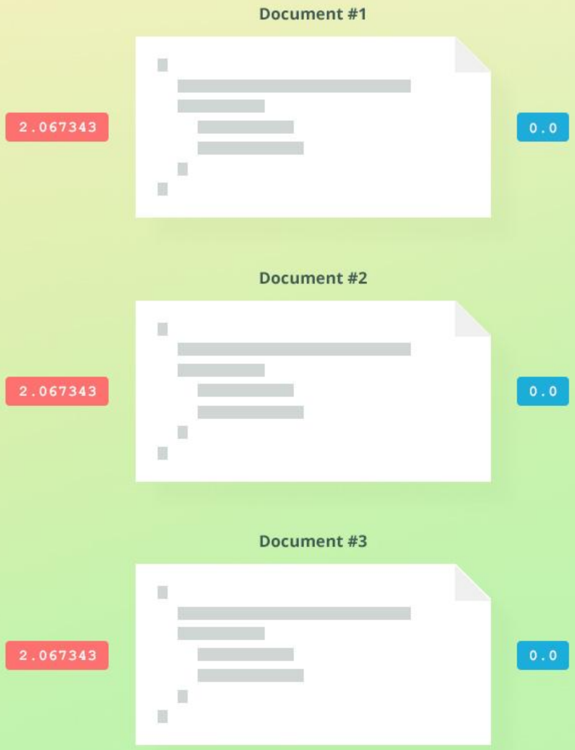
CodingExplained
CODE. SLEEP. REPEAT.

The `filter` occurrence type

- Query clauses must match
- Similar to the `must` occurrence type
- Ignores relevance scores
  - This improves the performance of the query ⚡
  - Query results can be cached and reused

Query ●

```
GET /products/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "tags.keyword": "Alcohol"
          }
        }
      ]
    }
  }
}
```



Query ●

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "tags.keyword": "Alcohol"
          }
        }
      ]
    }
  }
}
```

Occurrence types

Occurrence type	Description
`must`	Query clauses are required to match and will contribute to relevance scores.
`filter`	Query clauses are required to match, but will *not* contribute to relevance scores. Query clauses may therefore be cached for improved performance.
`must_not`	Query clauses must *not* match and do not affect relevance scoring. Query clauses may therefore be cached for improved performance.
`should`	Query clauses *should* match. Relevance scores of matching documents are boosted for each matching query clause. Behavior can be adjusted with `minimum_should_match`.

CodingExplained
CODE. SLEEP. REPEAT.

Occurrence types

Occurrence type	Required to match?	Affects relevance scores?	Can be cached?
`must`	Yes	Yes	No
`filter`	Yes	No	Yes
`must_not`	No	No	Yes
`should`	Conditional	Yes	No

CodingExplained
CODE. SLEEP. REPEAT.

The `match` query, revisited.

```
GET /products/_search
{
  "query": {
    "match": {
      "name": "PASTA"
    }
  }
}
```

```
GET /products/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "name": "pasta"
          }
        }
      ]
    }
  }
}
```

Document #1

```
POST /products/_doc
{
  "name": "Pasta with chicken"
}
```

Term	Document #1
"chicken"	X
"pasta"	X
"with"	X

CodingExplained
CODE. SLEEP. REPEAT.

```
GET /products/_search
{
  "query": {
    "match": {
      "name": "PASTA CHICKEN"
    }
  }
}
```

$\longrightarrow$

```
GET /products/_search
{
  "query": {
    "bool": {
      "should": [
        {
          "term": {
            "name": "pasta"
          }
        },
        {
          "term": {
            "name": "chicken"
          }
        }
      ]
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

```
GET /products/_search
{
  "query": {
    "match": {
      "name": {
        "query": "PASTA CHICKEN",
        "operator": "and"
      }
    }
  }
}
```

$\longrightarrow$

```
GET /products/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "name": "pasta"
          }
        },
        {
          "term": {
            "name": "chicken"
          }
        }
      ]
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

Query examples

CodingExplained
CODE. SLEEP. REPEAT.

Example 1

SQL query

```
WHERE (tags IN ("Beer") OR name LIKE '%Beer%') AND in_stock <= 100
```

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "in_stock": {
              "lte": 100
            }
          }
        }
      ]
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

Example 1

SQL query

```
WHERE (tags IN ("Beer") OR name LIKE '%Beer%') AND in_stock <= 100
```

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      ...
      "must": [
        {
          "bool": {
            "should": [
              { "term": { "tags.keyword": "Beer" } },
              { "match": { "name": "Beer" } }
            ]
          }
        }
      ]
    }
  }
}
```

↔

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "in_stock": {
              "lte": 100
            }
          }
        }
      ],
      "should": [
        { "term": { "tags.keyword": "Beer" } },
        { "match": { "name": "Beer" } }
      ],
      "minimum_should_match": 1
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

Example 2

SQL query

```
WHERE tags IN ("Beer") AND (name LIKE '%Beer%' OR description LIKE '%Beer%') AND in_stock <= 100
```

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "in_stock": {
              "lte": 100
            }
          }
        }
      ]
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

Example 2

SQL query

```
WHERE tags IN ("Beer") AND (name LIKE '%Beer%' OR description LIKE '%Beer%') AND in_stock <= 100
```

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        ...
        {
          "term": {
            "tags.keyword": "Beer"
          }
        }
      ]
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

Example 2

SQL query

```
WHERE tags IN ("Beer") AND (name LIKE '%Beer%' OR description LIKE '%Beer%') AND in_stock <= 100
```

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        ...
        ...
      ],
      "should": [
        { "match": { "name": "Beer" } },
        { "match": { "description": "Beer" } }
      ],
      "minimum_should_match": 1
    }
  }
}
```

CodingExplained
CODE. SLEEP. REPEAT.

Example 2

SQL query

```
WHERE tags IN ("Beer") AND (name LIKE '%Beer%' OR description LIKE '%Beer%') AND in_stock <= 100
```

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        . . .
        . . .
      ],
      "should": [
        { "match": { "name": "Beer" } },
        { "match": { "description": "Beer" } }
      ],
      "minimum_should_match": 1
    }
  }
}
```

Elasticsearch query

```
GET /products/_search
{
  "query": {
    "bool": {
      "filter": [
        . . .
        . . .
      ],
      "must": [
        {
          "multi_match": {
            "query": "Beer",
            "fields": ["name", "description"]
          }
        }
      ]
    }
  }
}
```

** The two queries differ slightly in terms of relevance scoring.*

CodingExplained
CODE. SLEEP. REPEAT.

Lecture summary (1/2)

- The `bool` query is one of the most important queries in Elasticsearch
- Occurrence types:
  - `must`: Must match. Affects relevance scores.
  - `must_not`: Must not match.
  - `should`: Boosts relevance scores for matching documents. Often used in combination with `must` and/or `filter`.
  - `filter`: Must match. Ignores relevance scores. Cacheable.

CodingExplained
CODE. SLEEP. REPEAT.

Lecture summary (2/2)

- If a `bool` query *only* contains `should` clauses, at least one is required to match (otherwise they are not required to match)
  - This can be adjusted with the `minimum_should_match` parameter
- `match` queries are *usually* translated into `bool` queries internally
  - What the translated query looks like depends on parameters, etc.