

**Zadanie 5 – Elastic Search**

**10.12.2022**

**Matej Delinčák**

**Github classroom:** <https://github.com/FIIT-DBS/zadanie-pdt-mateju25>

## 1. Rozbehajte si 3 inštalácie Elasticsearch-u

Pomocou docker compose som si rozbehal 3 inštalácie elasticu. Compose yml súbor bude priložený pri dokumentácií. Vytvoril som teda 3 inštalácie do jedného clustru a jeden z nich označil aj ako master.

GETlocalhost:9200/\_cat/nodes?v=trueSend

ParamsAuthHeaders (7)BodyPre-reqTestsSettingsCookies

Body

200 OK209 ms297 BSave Response

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	v	true			
	Key	Value	Description		

PrettyRawPreviewVisualizeText

1ipheap.percentram.percentcpuload\_1mload\_5mload\_15mnode.rolemastername

2172.19.0.34310000.160.060.01d-es02

3172.19.0.23210000.160.060.01dm-es01

4172.19.0.46210000.160.060.01d-es03

5

## 2. Vytvorte index pre Tweety, ktorý bude mať “optimálny” počet shardov a replík pre 3 nody (aby tam bola distribúcia dotazov vo vyhľadávaní, aj distribúcia uložených dát)

V dokumentácii som sa dočítal, že tri shardy je ideálne na 3 nody ak nemám veľa pamäte. Tak to budem mať aj paralelne spracovávanie. Počet replík som nastavil na 1, ktorá je aj defaultná hodnota. Počet replík je dôležitý pre dostupnosť systému. Ale nemyslím, že u mňa systém bude tak zaťažný (aby sa stala nejaká nehoda), aby som potreboval až dve repliky.

```
PUT localhost:9200/tweets

{
  "settings": {
    "index": {
      "number_of_shards": 3,
      "number_of_replicas": 1
    }
  }
}
```

```
{
  "acknowledged": true,
  "shards_acknowledged": true,
  "index": "tweets"
}
```

Overenie:

index	shard	prirep	state	docs	store	ip	node
.geoip_databases	0	r	STARTED	41	38.9mb	172.25.0.4	es03
.geoip_databases	0	p	STARTED	41	39mb	172.25.0.3	es02
tweets	0	r	STARTED	0	225b	172.25.0.4	es03
tweets	0	p	STARTED	0	225b	172.25.0.2	es01
tweets	1	p	STARTED	0	225b	172.25.0.4	es03
tweets	1	r	STARTED	0	225b	172.25.0.3	es02
tweets	2	p	STARTED	0	225b	172.25.0.3	es02
tweets	2	r	STARTED	0	225b	172.25.0.2	es01

**3. Vytvorte mapping pre normalizované dáta z Postgresu (denormalizujte ich) – Každý Tweet teda musí obsahovať údaje rovnaké ako máte už uložené v PostgreSQL (všetky tabuľky). Dbajte na to, aby ste vytvorili polia v správnom dátovom type (polia ktoré má zmysel analyzovať analyzujte správne, tie ktoré nemá, aby neboli zbytočne analyzované (keyword analyzer)) tak aby index nebol zbytočne veľký, pozor na nested – treba ho použiť správne. Mapovanie musí byť striktné. Čo sa týka väzieb cez referencies – pre ne zaindexujte type vstáhu, id, autor (id, name, username), content a hashtags.**

Vytvoril som nasledovné mapovanie. Pre čísla som použil integer, pre string typ text. Typ keyword som použil pre id stĺpce a hastagy, keďže v dokumentácii som sa dočítal, že sa najčastejšie používajú na tento typ. Typ nested som použil pre author, content\_domains a context\_entities. Z dokumentácie: „*The nested type is a specialised version of the object data type that allows arrays of objects to be indexed in a way that they can be queried independently of each other.*“ Mapping bude priložený pri dokumentácii.

```
{
  "dynamic": "strict",
  "properties": {
    "source_id": {"type": "keyword"},
    "author": {
      "type": "nested",
      "properties": {
        "id": {"type": "keyword"},
        "username": {"type": "text"},
        "name": {"type": "text"},
        "description": {"type": "text"},
        "followers_count": {"type": "integer"},
        "following_count": {"type": "integer"},
        "listed_count": {"type": "integer"},
        "tweet_count": {"type": "integer"}
      }
    },
    "content": {"type": "text"},
    "language": {"type": "text"},
    "source": {"type": "text"},
    "retweet_count": {"type": "integer"},
    "reply_count": {"type": "integer"},
    "like_count": {"type": "integer"},
    "quote_count": {"type": "integer"},
    "possibly_sensitive": {"type": "boolean"},
    "created_at": {"type": "date"},
    "hashtags": {"type": "keyword"},
    "links": {
      "properties": {
        "title": {"type": "text"},
        "description": {"type": "text"},
        "url": {"type": "text"}
      }
    }
  }
},
  "annotations": {
    "properties": {
      "value": {"type": "text"},
      "type": {"type": "text"},
      "probability": {"type": "double"}
    }
  },
  "context_annotations": {
    "type": "nested",
    "properties": {
      "domain": {
        "type": "nested",
        "properties": {
          "name": {"type": "text"},
          "description": {"type": "text"}
        }
      },
      "entity": {
        "type": "nested",
        "properties": {
          "name": {"type": "text"},
          "description": {"type": "text"}
        }
      }
    }
  }
}
```

```
{
  "conversation_references": {
    "properties": {
      "id": {"type": "keyword"},
      "type": {"type": "text"},
      "hashtags": {"type": "keyword"},
      "content": {"type": "text"},
      "author": {
        "properties": {
          "id": {"type": "keyword"},
          "name": {"type": "text"},
          "username": {"type": "text"}
        }
      }
    }
  }
}
```

PUT

localhost:9200/tweets/\_mapping

Params

Auth

Headers (9)

Body

Pre-req.

Tests

Settings

Cookies

Body

Cookies

Headers (4)

Test Results

raw

JSON

Beautify

```
1 {
2   "dynamic": "strict",
3   "properties": {
4     "source_id": {"type": "keyword"},
```

Pretty

Raw

Preview

Visualize

JSON

⌵

```
1 {
2   "acknowledged": true
3 }
```

4. Pre index tweets vytvorte 3 vlastné analyzéry (v settings) nasledovne:

a. Analyzátor "englando". Tento analyzátor bude obsahovať nasledovné:

- i. filtre: english\_possessive\_stemmer, lowercase, english\_stop, english\_stemmer,
- ii. char\_filter: html\_strip
- iii. tokenizer: štandardný - ukážku nájdete na stránke elastic.co pre anglický analyzátor

Vytvoril som nasledovný analyzátor podľa zadania. Išlo to ľahko podľa dokumentácie. Všetky filtre okrem lowercase som musel ešte dotatočne customizovať, lebo taký názov neexistoval.

PUT localhost:9200/tweets/\_settings

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON Beautify

```
1 {
2   "settings": {
3     "analysis": {
4       "analyzer": {
5         "englando": {
6           "type": "custom",
7           "tokenizer": "standard",
8           "char_filter": [
9             "html_strip"
10          ],
11           "filter": [
12             "english_possessive_stemmer",
13             "lowercase",
14             "english_stop",
15             "english_stemmer"
16          ]
17         },
18         "english_possessive_stemmer": {
19           "language": "possessive_english",
20           "type": "stemmer"
21         },
22         "english_stop": {
23           "type": "stop",
24           "stopwords": "_english_"
25         },
26         "english_stemmer": {
27           "type": "stemmer",
28           "language": "english"
29         }
30       }
31     }
32   }
33 }
34
35 }
```

Body Cookies Headers (4) Test Results

200 OK 251 ms 176 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "acknowledged": true
3 }
```

Ale predtým, než som mohol daný analyzátor vytvoriť, tak som musel najprv vypnúť index.

POST localhost:9200/tweets/\_close

POST localhost:9200/tweets/\_open

**b. Analyzátor custom\_ngram:**

- i. filtre: lowercase, ascii-folding, filter\_ngrams (definujte si ho sami na rozmedzie 1- 10)
- ii. char\_filter: html\_strip
- iii. tokenizer: štandardný

Vytvorenie analyzátoru custom\_ngram bolo taktiež priamočiare. Ale keď som vytváral filter\_ngrams, tak som dostal error hlášku, že maximálna hodnota ngramu pre index je 9. Tak som ešte musel nastaviť max. prístupnú hodnotu na 10.

The screenshot shows a REST client interface with a PUT request to `localhost:9200/tweets/_settings`. The request body is a JSON configuration for Elasticsearch settings. The response body shows `"acknowledged": true`.

```
1 {
2   "settings": {
3     "index": {
4       "max_ngram_diff": 10
5     },
6     "analysis": {
7       "analyzer": {
8         "custom_ngram": {
9           "type": "custom",
10          "tokenizer": "standard",
11          "char_filter": [
12            "html_strip"
13          ],
14          "filter": [
15            "asciifolding",
16            "lowercase",
17            "filter_ngrams"
18          ]
19        },
20      },
21      "filter": {
22        "filter_ngrams": {
23          "type": "ngram",
24          "min_gram": 1,
25          "max_gram": 10
26        }
27      }
28    }
29  }
30 }
```

```
1 {
2   "acknowledged": true
3 }
```

**c. Analyzér custom\_shingles:**

- i. filtre: lowercase, asciifolding, filter\_shingles (definujte si ho sami a dajte token\_separator: "")
- ii. char\_filter: html\_strip
- iii. tokenizer: štandardný

Pri vytvorení analyzéra custom\_shingles bolo potrebné vytvoriť custom filter pre typ shingle s parametrom token\_separator.

The screenshot displays a REST client interface with a PUT request to `localhost:9200/tweets/_settings`. The request body is a JSON object defining custom shingles and filters. The response body shows `"acknowledged": true`.

**Request Body (JSON):**

```
1 {
2   "settings": {
3     "analysis": {
4       "analyzer": {
5         "custom_shingles": {
6           "type": "custom",
7           "tokenizer": "standard",
8           "char_filter": [
9             "html_strip"
10          ],
11          "filter": [
12            "asciifolding",
13            "lowercase",
14            "filter_shingles"
15          ]
16        },
17      },
18      "filter": {
19        "filter_shingles": {
20          "type": "shingle",
21          "token_separator": ""
22        }
23      }
24    }
25  }
26 }
```

**Response Body (JSON):**

```
1 {
2   "acknowledged": true
3 }
```

d. Do mapovania pridajte:

i. každý anglický text (rátajme že každý tweet a description u autora je primárne v angličtine) nech je analyzovaný novým analyzátom "englando"

ii. Priradte analyzery

1. a. author.name nech má aj mapovania pre custom\_ngram, a custom\_shingles
2. b. author.screen\_name nech má aj custom\_ngram,
3. c. author.description nech má aj custom\_shingles. Toto platí aj pre mentions, ak tam tie záznamy máte.

iii. Hashtagy indexujte ako lowercase

Pre zadefinovanie viacerých analyzátorov, som musel použiť fields parameter. Aby sa hashtagy indexovali ako lowercase, tak som si vytvoril normalizér s menom custom\_normalizer, ktorý má filter lowercase v sebe.

PUT localhost:9200/tweets/\_mapping

Params Auth Headers (9) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

```
5 ..... "author": {
6 .....   "type": "nested",
7 .....   "properties": {
8 .....     "id": { "type": "keyword",
9 .....       "username": { "type": "text", "analyzer": "englando",
10 .....     "fields": {
11 .....       "ngram": {
12 .....         "type": "text",
13 .....         "analyzer": "custom_ngram"
14 .....       },
15 .....     },
16 .....   },
17 .....   "name": { "type": "text", "analyzer": "englando",
18 .....     "fields": {
19 .....       "ngram": {
20 .....         "type": "text",
21 .....         "analyzer": "custom_ngram"
22 .....       },
23 .....       "shingles": {
24 .....         "type": "text",
25 .....         "analyzer": "custom_shingles"
26 .....       }
27 .....     }
28 .....   },
29 .....   "description": { "type": "text", "analyzer": "englando",
30 .....     "fields": {
31 .....       "shingles": {
32 .....         "type": "text",
33 .....         "analyzer": "custom_shingles"
34 .....       }
35 .....     }
36 .....   },
37 .....   "followers_count": { "type": "integer"},
38 ..... },
39 ..... "created_at": { "type": "date",
40 ..... },
41 ..... "hashtags": { "type": "keyword", "normalizer": "custom_normalizer"},
42 ..... "links": {
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize

```
1 ..... "acknowledged": true
2 .....
3 .....
```



## 5. Vytvorte bulk import pre vaše normalizované Tweety.

## 6. Importujete dáta do Elasticsearchu prvych 5000 tweetov

V súbore export.txt sa nachádza query podľa, ktorej som si vytiahol json pre každú konverzáciu. Tento json obsahuje donormalizované dáta z databázy. Pri query som využil funkcie ako `jsonb_build_object`, ktorá vyskladala z dát json. Ďalej `json_agg`, čo je window funkcia na agregáciu jsonov do poľa. Funkcia `row_to_json` vytvorí s viacerých stĺpcov jeden stĺpec, v ktorom je json. Novou vecou pre mňa bola funkcionality LATERAL join, čo je v podstate niečo ako for cyklus. Túto novinku mi poradil cvičiaci, bez nej by som nevedel vytvoriť sql tak jednoducho. Ďalej pri importe do elasticu našiel nejaké znaky, čo robili problémy, tak som pomocou regexu takéto opravil. Export mi trval 13 hodín.

Následne som si vytvoril script v jave, ktorý pomocou bulk api elasticu vložil 5000 tweetov. Overil som si import pomocou vytiahnutia všetkých dát z elasticu. A boli tam už všetky.

The screenshot shows a REST client interface with a GET request to `localhost:9200/tweets/_search`. The request body is a JSON query:

```
1 {
2   "query": {
3     "match_all": {}
4   }
5 }
```

The response body is a JSON object showing search statistics:

```
1 {
2   "took": 1890,
3   "timed_out": false,
4   "_shards": {
5     "total": 3,
6     "successful": 3,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 5000,
13      "relation": "eq"
14    },
15    "max_score": 1.0,
16    "hits": [
17      {
18        "_index": "tweets",
19        "type": "tweet"
20      }
21    ]
22  }
23 }
```

**7. Experimentujte s nódami, a zistite koľko nódov musí bežať (a ktoré) aby vám Elasticsearch vedel pridávať dokumenty, mazať dokumenty, prezerať dokumenty a vyhľadávať nad nimi? Dá sa nastaviť Elastic tak, aby mu stačil jeden nód? Čo je dôvodom toho že existuje nejaké kvórum?**

Táto úloha sa správala u mňa zvlášťne. Teda čakal som, že čím viac nodov pobeží, tým rýchlejšie budú veci. Ale pravdou bolo, že keď bežal práve jeden, tak som mal najlepšie výsledky. Cvičiaci mi poradil, aby som si nastavil pre jeden shard aspoň 1 cpu. Táto zmena trochu zlepšila rozdiely. Avšak túto skutočnosť pripisujem tomu, že ak je viac nodov, treba omnoho viac času na manažment vecí okolo a pri takto malej vzorke to prevýši čas potrebný na vyhľadávanie. Samozrejme, keď nebežal žiaden node, tak to nenašlo nič a dostal som error.

Pridávanie záznamu nevrátilo took ale uviedol som aspoň čas, ktorý sa zobrazil v postmanovi. Ak by sme chceli elastic len s jedným nodom, tak musí byť typu master aj data a je vhodné nastaviť na index len jeden shard.

	Pridávanie	Mazanie	Prezeranie	Vyhľadávanie
Bežia 3 node-y	X (39ms)	37	5	7
Bežia 2 node-y	X (30ms)	34	6	10
Beží 1 nod-y (len master)	X (25ms)	21	5	7
Beží 0 nodov	X	X	X	X

Tabuľka trvaní (took)

#### Pridávanie

PUT localhost:9200/tweets/\_doc/527

Params Auth Headers (9) Body Pre-req. Tests

raw JSON

```

1 {
2   "source_id": "1499084337626722310",
3   "author": {
4     "id": "1378",
5     "name": "Michael (मुकेश) पाठक",
6     "username": "MParekh",
7     "description": "Tech OODA Looper.us
8     IN.Tech Investor/Builder.LT.Tec
9     Ex-GS Partner/Internet Ax. Syn
10    "tweet count": 31855.

```

#### Prezeranie

GET localhost:9200/tweets/\_search

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON

```

1 {
2   "query": {
3     "match": {
4       "source_id": "1499084337626722310"
5     }
6   }
7 }

```

#### Mazanie

POST localhost:9200/tweets/\_delete\_by\_query

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON

```

1 {
2   "query": {
3     "match": {
4       "source_id": "1499084337626722310"
5     }
6   }
7 }

```

#### Vyhľadávanie

POST localhost:9200/tweets/\_search

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON

```

1 {
2   "query": {
3     "bool": {
4       "must": [
5         {
6           "range": {
7             "retweet_count": {
8               "gte": 20000
9             }
10        }
11      ]
12    }
13  }
14 }

```

Ukážka ako sa rozdelili shardy a repliky pri **dvoch** nodoch:

```
1 index shard prirep state docs store ip node
2 tweets 0 p STARTED 10925 2.8mb 172.19.0.2 es01
3 tweets 0 r UNASSIGNED
4 tweets 1 p STARTED 10969 2.8mb 172.19.0.4 es03
5 tweets 1 r UNASSIGNED
6 tweets 2 r STARTED 10818 2.8mb 172.19.0.2 es01
7 tweets 2 p STARTED 10818 2.8mb 172.19.0.4 es03
8
```

---

```
1 ip heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
2 172.19.0.2 23 98 12 0.53 0.81 0.75 dm * es01
3 172.19.0.4 39 95 12 0.53 0.81 0.75 d - es03
4
```

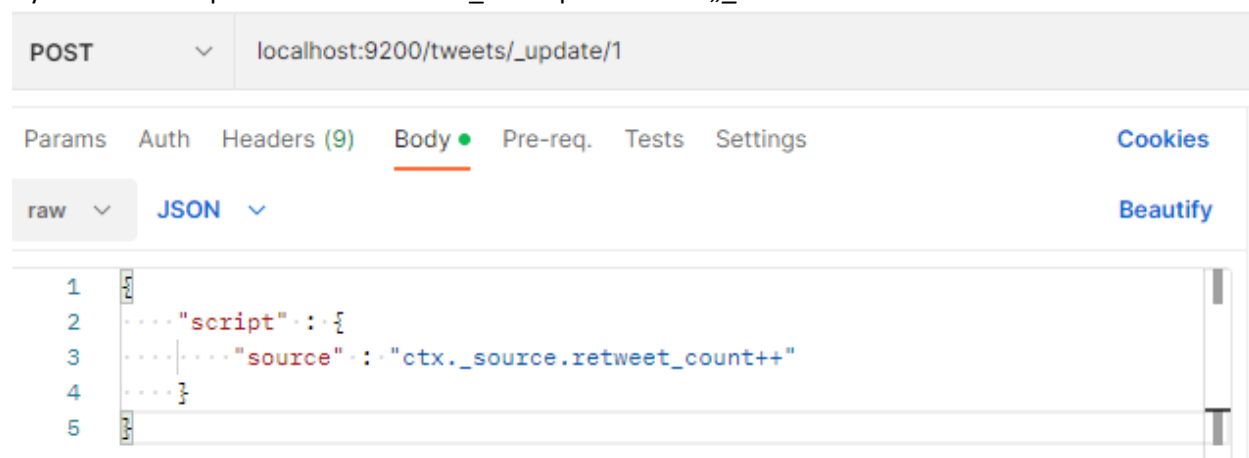
Ukážka ako sa rozdelili shardy a repliky pri **jednom** node:

```
1 index shard prirep state docs store ip node
2 tweets 0 p STARTED 10925 2.8mb 172.19.0.2 es01
3 tweets 0 r UNASSIGNED
4 tweets 1 p UNASSIGNED
5 tweets 1 r UNASSIGNED
6 tweets 2 p STARTED 10818 2.8mb 172.19.0.2 es01
7 tweets 2 r UNASSIGNED
8
```

```
1 ip heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
2 172.19.0.2 40 99 2 0.46 0.63 0.69 dm * es01
3
```

8. Upravujte počet retweetov pre vami vybraný tweet pomocou vašeho jednoduchého scriptu (v rámci Elasticsearchu) a sledujte ako sa mení `_seq_no` a `_primary_term` pri tom ako zabíjate a spúšťate nódy.

Vytvoril som skript na menení `retweet_count` pre tweets s „`id`“ 1.



Vždy keď updatnem tento dokument, alebo aj ktorýkoľvek iný. Tak `_seq_no` sa zvýši o jedno. Je to preto, lebo toto číslo predstavuje počet operácií nad indexom. A teda keď som opätovne posielal rovnaký request, tak sa toto číslo zvyšovalo o jeden.

Čo sa týka `_primary_term`, tak tento sa nezvyšoval obvyčajným posielaním requestu. Pre jeho zmenu som skúšal vypnúť es-03 node. Ale nič sa nezmenilo a teda tento dokument nebol uložený na primary sharde na tomto node. Zmena nastala pri vypínaní es-02, es-01 a celého klastra. Vždy pri týchto procesoch sa záznam zapísal na iný shard a vtedy sa zmenil aj `_primary_term`.

Tieto dve čísla elastic používa na to aby zabezpečil Optimistic concurrency control. V skratke ide o riešenie konfliktov pri lost update problému.

Pred vypnutím es-02

```
1 {
2   "_index": "tweets",
3   "_id": "1",
4   "_version": 18,
5   "result": "updated",
6   "_shards": {
7     "total": 2,
8     "successful": 1,
9     "failed": 0
10  },
11   "_seq_no": 1677,
12   "_primary_term": 10
13 }
```

Po vypínaní es-02, es-01 a celého klastra

```
1 {
2   "_index": "tweets",
3   "_id": "1",
4   "_version": 29,
5   "result": "updated",
6   "_shards": {
7     "total": 2,
8     "successful": 2,
9     "failed": 0
10  },
11   "_seq_no": 1688,
12   "_primary_term": 13
13 }
```

## 9. Zrušte repliky a importujte všetky tweety

Vytvoril som si nový index bez replík a starý som vymazal. Pomocou rovnakého java skriptu ako v úlohe 5 som si naimportoval všetky dáta, ktoré som mal exportnuté do jsonu. Import trval cca 6 hodín.

GETlocalhost:9200/tweets/\_count

ParamsAuthHeaders (7)BodyPre-req. TestsSettingsCookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (4)Test Results

PrettyRawPreviewVisualize

```
1  {
2    "count": 32347011,
3    "_shards": {
4      "total": 3,
5      "successful": 3,
6      "skipped": 0,
7      "failed": 0
8    }
9  }
```

## 10. Vyhľadajte vo vašich tweetoch, kde použite function\_score pre jednotlivé medzikroky nasledovne

V query na vyhľadanie „put1n christian fake jew“ som použil multi\_match, keďže trebalo viacero fieldov. Type som nastavil na most\_fields, aby sa dokumenty zoradili podľa počtu výrazov. Boost pre jednotlivé fieldy som spravil cez ^. Ďalej do must som pridal term pre hashtag a pre conversation\_references.content.

```
{
  "size": 10,
  "query": {
    "function_score": {
      "query": {
        "bool": {
          "must": [
            {
              "multi_match": {
                "query": "putin christian fake jew",
                "type": "most_fields",
                "fields": [
                  "author.description.shingles^10",
                  "content^6"
                ],
                "fuzziness": "auto",
                "operator": "or"
              }
            },
            {
              "term": {
                "hashtags": "ukraine"
              }
            },
            {
              "term": {
                "conversation_references.content": "nazi"
              }
            }
          ]
        }
      }
    }
  }
}
```

Čo sa týka filtrov, tak som potreboval urobiť nested query pre autora, kde som naraz vyfiltroval dané počty pre following\_count a followers\_count. Ako posledný filter som ešte pridal to, aby dokument obsahoval aspoň jeden link.

```
{
  "filter": [
    {
      "nested": {
        "path": "author",
        "query": {
          "bool": {
            "must": [
              {
                "range": {
                  "author.following_count": {
                    "gt": 100
                  }
                }
              },
              {
                "range": {
                  "author.followers_count": {
                    "gt": 100
                  }
                }
              }
            ]
          }
        }
      },
      {
        "exists": {
          "field": "links.uri"
        }
      }
    ]
  }
}
```

Do časti should som pridal 4 pravidiel. Dva nested query pre contex\_domain a context\_entity, ktoré som boostol podľa zadania.

```
"should": [
  {
    "nested": {
      "path": "context_annotations",
      "query": {
        "nested": {
          "path": "context_annotations.entity",
          "query": {
            "match": {
              "context_annotations.entity.name": {
                "query": "Soros",
                "boost": 10
              }
            }
          }
        }
      }
    }
  },
  {
    "nested": {
      "path": "context_annotations",
      "query": {
        "nested": {
          "path": "context_annotations.domain",
          "query": {
            "match": {
              "context_annotations.domain.name": {
                "query": "Person",
                "boost": 5
              }
            }
          }
        }
      }
    }
  }
],
{
  "nested": {
    "path": "author",
    "query": {
      "match_phrase": {
        "author.description": {
          "query": "putin christian fake jew",
          "slop": 1,
          "boost": 5
        }
      }
    }
  }
},
{
  "match_phrase": {
    "content": {
      "query": "putin christian fake jew",
      "slop": 1,
      "boost": 5
    }
  }
}
```

Ďalej nested query pre autora a pre content, kde som boostol daný dokument pokiaľ obsahoval frázu „putin christian fake jew“ so slopom 1 (výmena slov).

```
{
  "nested": {
    "path": "author",
    "query": {
      "match_phrase": {
        "author.description": {
          "query": "putin christian fake jew",
          "slop": 1,
          "boost": 5
        }
      }
    }
  }
},
{
  "match_phrase": {
    "content": {
      "query": "putin christian fake jew",
      "slop": 1,
      "boost": 5
    }
  }
}
```

Nakoniec som pridal agregáciu, kde som agregoval dokumenty podľa týždňov. Spočítal som celkový počet dokument, ktoré spadajú do tohto týždňa a potom som vytvoril bucket pro-russia, ktorý zrátal, koľko z nich obsahovalo hashtagy zo zadania.

```
"aggs": {
  "hashtags": {
    "date_histogram": {
      "field": "created_at",
      "calendar_interval": "week",
      "format": "yyyy-MM-dd"
    },
    "aggs": {
      "hashtags": {
        "filters": {
          "filters": {
            "pro-russia": {
              "terms": {
                "hashtags": [
                  "istandwithputin",
                  "racism",
                  "itrillion",
                  "istandwithrussia",
                  "isupportrussia",
                  "blacklivesmatter",
                  "racistukraine",
                  "africansinukraine",
                  "palestine",
                  "israel",
                  "freepalestine",
                  "istandwithpalestine",
                  "racisteu",
                  "putin"
                ]
              }
            }
          }
        }
      }
    }
  }
}
```

Nakoniec som vyfiltroval také buckety, ktoré neobsahovali ani jeden dokument, kvôli lepšej prehľadnosti vo výsledku.

```
.....:}
.....:},
.....:"sales_bucket_filter":{
.....:"bucket_selector":{
.....:"buckets_path":{
.....:"docCount": "_count"
.....:},
.....:"script": "params.docCount > 0"
.....:}
.....:}
.....:?
```

Daná query mi našla tento dokument, ktorý má najväčšie skóre:

```
.....:"hits": {
.....:  "total": {
.....:    "value": 1371,
.....:    "relation": "eq"
.....:  },
.....:  "max_score": 142.47322,
.....:  "hits": [
.....:    {
.....:      "_shard": "[tweets][1]",
.....:      "_node": "S-BeD_JVRSCqzpucc@moyQ",
.....:      "_index": "tweets",
.....:      "_id": "10477411",
.....:      "_score": 142.47322,
.....:      "_source": {
.....:        "source_id": 1497698752907390979,
.....:        "author": {
.....:          "id": 1450394162,
.....:          "name": "us Reel Conquest us",
.....:          "username": "ReelConquest",
.....:          "description": "us US Army Veteran us Patriot, 2nd
.....:            Amendment advocate, MAGA #TrumpWon 🇺🇸🇵🇸",
.....:          "us 🇺🇸🇵🇸-🇷🇺🇵🇸🇷🇺🇵🇸",
.....:          "tweet_count": 36466,
.....:          "listed_count": 3,
.....:          "followers_count": 906,
.....:          "following_count": 1115
.....:        }
.....:      }
.....:    },
.....:  ],
.....:}
```

A agregácie vyzerali takto:

```
.....:
.....:  "key_as_string": "2022-03-07",
.....:  "key": 1646611200000,
.....:  "doc_count": 86,
.....:  "hashtags": {
.....:    "buckets": {
.....:      "pro-russia": {
.....:        "doc_count": 48
.....:      }
.....:    }
.....:  },
.....:},
.....:{
.....:  "key_as_string": "2022-03-14",
.....:  "key": 1647216000000,
.....:  "doc_count": 206,
.....:  "hashtags": {
.....:    "buckets": {
.....:      "pro-russia": {
.....:        "doc_count": 104
.....:      }
.....:    }
.....:  },
.....:},
.....:},
.....:{
.....:  "key_as_string": "2022-03-21",
.....:  "key": 1647820800000,
.....:  "doc_count": 201,
.....:  "hashtags": {
.....:    "buckets": {
.....:      "pro-russia": {
.....:        "doc_count": 112
.....:      }
.....:    }
.....:  },
.....:},
.....:}
```