Pokročilé databázové technológie

Zadanie 5 – Elasticsearch

Marek Adamovič

Cvičenie: Piatok 11:00 – 12:40

Ing. Ján Balažia, PhD. 2022/2023

Obsah

[1. Rozbehanie inštancií 3](#_Toc122728829)

[2. Index pre Tweety 5](#_Toc122728830)

[3. Mapping pre normalizované dáta 6](#_Toc122728831)

[4. Vlastné analyzéry 8](#_Toc122728832)

[5. Bulk import 12](#_Toc122728833)

[6. Import 14](#_Toc122728834)

[7. Uzly 16](#_Toc122728835)

[8. Script 17](#_Toc122728836)

[9. Veľký import 20](#_Toc122728837)

[10. Vyhľadávanie 23](#_Toc122728838)

# 1. Rozbehanie inštancií

**Otázka:**

Rozbehajte si 3 inštancie Elasticsearch-u.

**Odpoveď:**

Inštancie sme vytvárali na windowse. Stiahli sme si Elasticsearch a rozbalili ho do troch zložiek (každá zložka tvorí jednu inštanciu). Následne sme nakonfigurovali elasticsearch.yml súbor (obrázok č.1) v každej z troch zložiek a postupne spustili jednotlivé inštancie pomocou súboru elasticsearch.bat. Konfigurácie sa líšia minimálne, len v názve uzlu a portoch. Na obrázku č.2 vidíme náš cluster, ktorý obsahuje 3 uzly.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Konfigurácia prvého uzlu v clusteri

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Kontrola clusteru

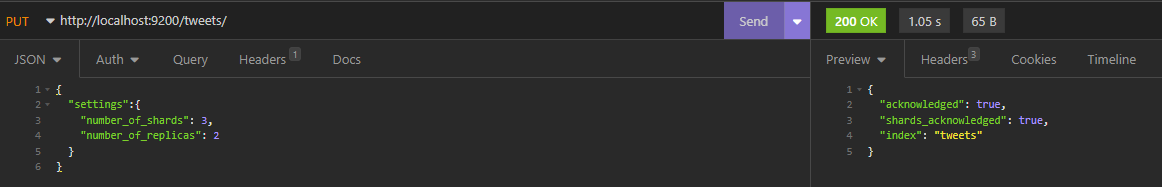
# 2. Index pre Tweety

**Otázka:**

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

**Odpoveď:**

Rozhodli sme sa zvoliť počet shardov 3, aby sme vedeli naplno využiť paralelizmus pri 3 uzloch. Počet replík sme zvolili 2, keďže nemá zmysel dávať väčšie číslo ako počet uzlov mínus jedna (keďže jedny dáta máme vždy defaultne).



Obrázok Vytvorenie indexu

# 3. Mapping pre normalizované dáta

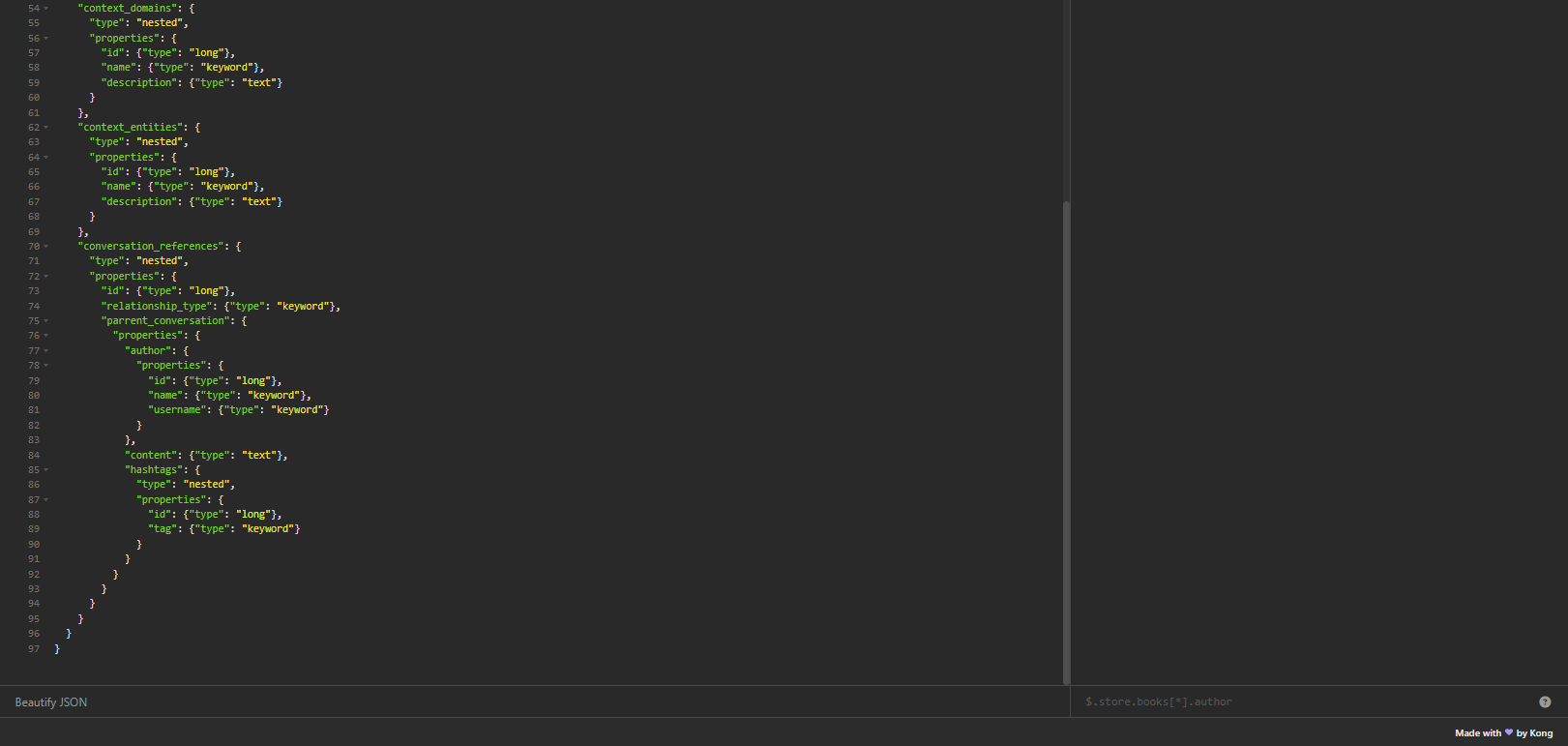
**Otázka:**

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 vzťahu, id, autor (id, name, username), content a hashtags.

**Odpoveď:**

Pre striktné mapovanie sme použili dynamic -> false, čo nám zakazuje pridávať nové polia do indexu. Typ nested sme použili len pre také polia, ktoré majú typ vzťahu ku tweetom many-to-one (to znamená, že na jednu konverzáciu ich môže byť viacero). Pri vzťahu one-to-one alebo one-to-many nebolo potrebné využívať typ nested. Niekoľkokrát sme použili typ keyword (v úlohe 4 sme ho potom niekde vymenili za text kvôli vlastným analyzérom), keďže tento typ má už daný analyzer keyword a jeho jediné obmedzenie je maximálna dĺžka stringu 32766 znakov. Preto sme ho využívali pre polia, ktoré túto dĺžku nemôžu presiahnúť a zároveň ich nechceme analyzovať (keďže typ text sa automaticky analyzuje). Ak sme mali polia, ktoré môžu presahovať spomenutú dĺžku a napriek tomu ich nechceme analyzovať, použili sme typ text a explicitne sme nastavili analyzer na hodnotu keyword. Na nasledujúcich obrázkoch uvádzame naše mapovanie.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Ukážka mapovania

# 4. Vlastné analyzéry

**Otázka:**

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

a. Analyzér "englando". Tento analyzér bude obsahovať nasledovné:

i. fitre: english\_possessive\_stemmer, lowercase, english\_stop, english\_stemmer,

ii. char\_filter: html\_strip

iii. tokenizer: štandardný - ukážku nájdete na stránke elastic.co pre anglický analyzér

b. Analyzér custom\_ngram:

i. filtre: lowercase, asciifolding, filter\_ngrams (definujte si ho sami na rozmedzie 1- 10)

ii. char\_filter: html\_strip

iii. tokenizer: štandardný

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

d. Do mapovania pridajte:

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

ii. Priraďte analýzery

1. a. author.name nech má aj mapovania pre custom\_ngram, a custom\_shingles

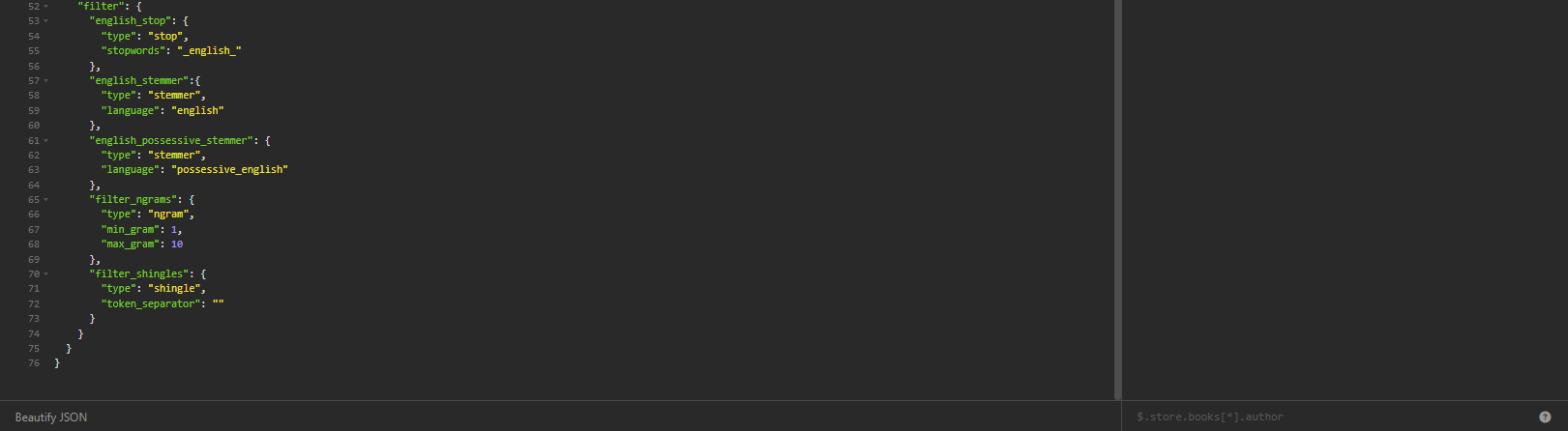
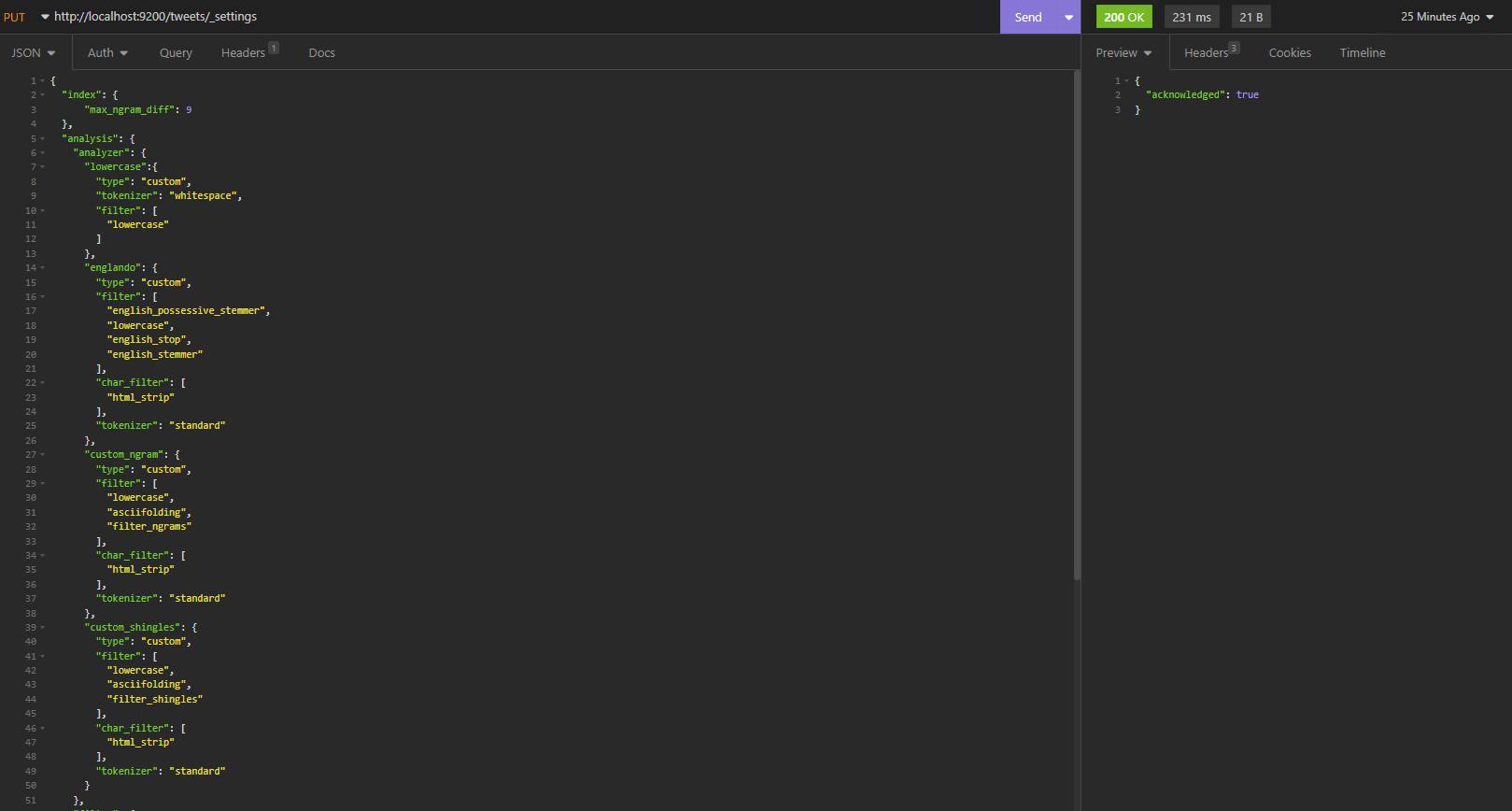
2. b. author. screen\_name nech má aj custom\_ngram,

3. c. author.description nech má aj custom\_shingles. Toto platí aj pre mentions, ak tam tie záznamy máte.

iii. Hashtagy indexujte ako lowercase

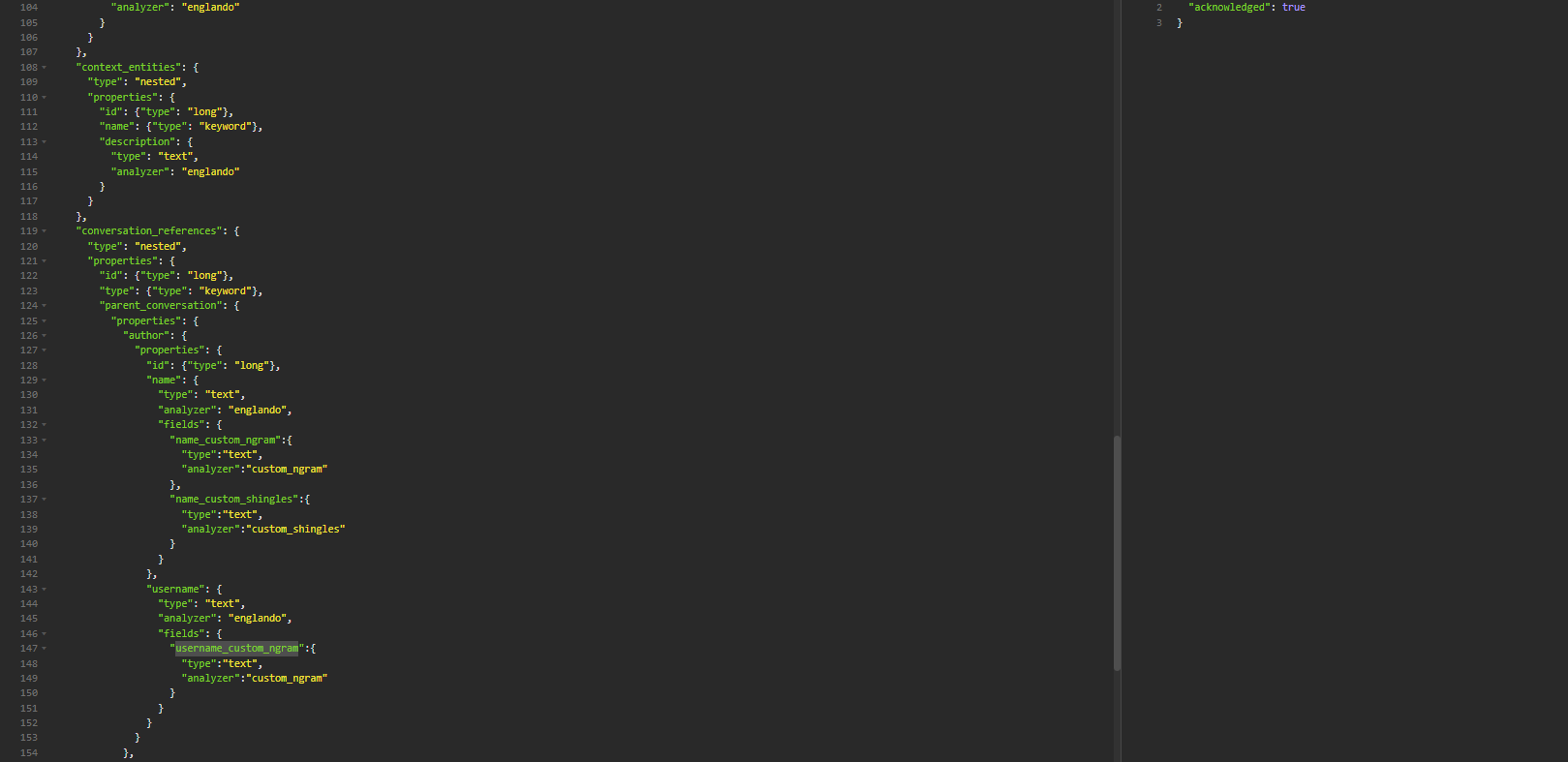
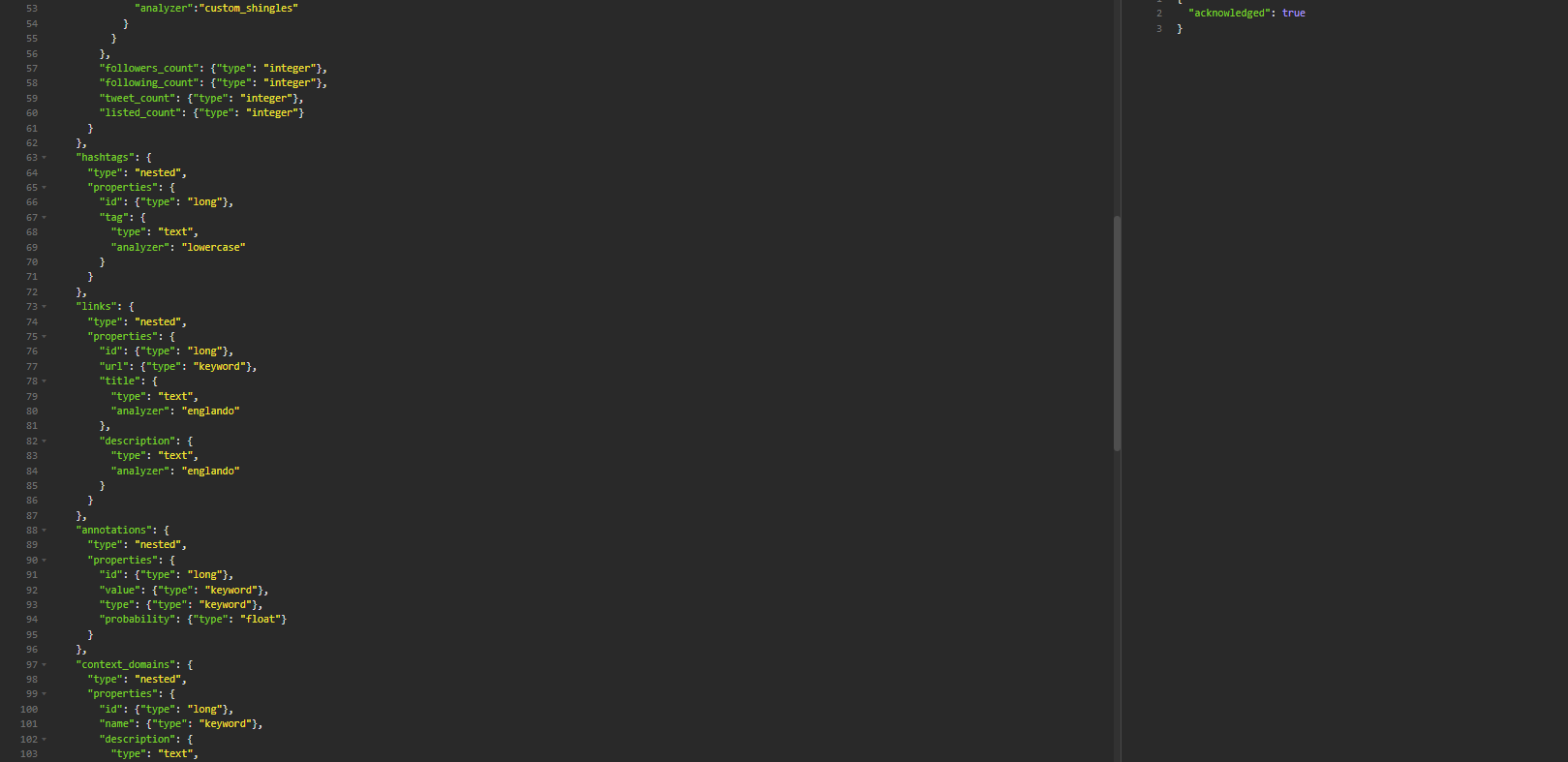
**Odpoveď:**

Najskôr sme vytvorili analyzéry v settings (obrázok č.5). Aby sme ich vytvorili, museli sme pozastaviť index pomocou POST dopytu http://localhost:9200/tweets/\_close. Po vytvorení sme ho znovu spustili pomocou http://localhost:9200/tweets/\_open. Následne sme upravili pôvodný mapping, aby využíval dané analyzéry tam, kde treba (obrázok č.6). Tým pádom mapping musíme vytvárať až po vytvorení našich analyzérov.



Obrázok Vytvorenie custom analyzérov

Obrázok, na ktorom je text, snímka obrazovky, elektronika

Automaticky generovaný popis

Obrázok Upravený mapping

# 5. Bulk import

**Otázka:**

Vytvorte bulk import pre vaše normalizované Tweety.

**Odpoveď:**

Pre bulk import sme si najskôr vytvorili indexy (obrázok č.7) pre všetky cudzie kľúče. Následne sme napísali dopyt (obrázok č.8), ktorý denormalizuje dáta zo všetkých tabuliek a spojí ich do jedného JSON stringu na jeden riadok.

Obrázok, na ktorom je text, obrazovka, snímka obrazovky

Automaticky generovaný popis

Obrázok vytvorenie indexov

Obrázok, na ktorom je text

Automaticky generovaný popis Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Dopyt denormalizujúci dáta z tabuliek

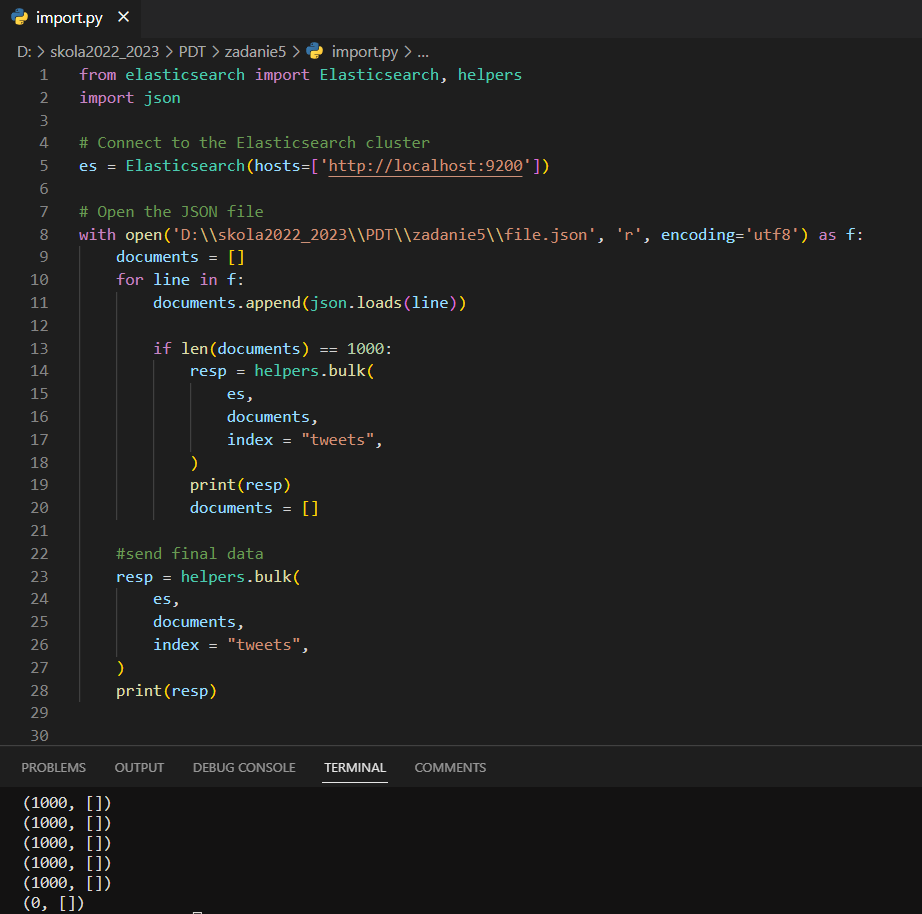
# 6. Import

**Otázka:**

Importujete dáta do Elasticsearchu prvych 5000 tweetov.

**Odpoveď:**

Keď sme mali tieto dáta v json súbore, importovali sme ich do elasticu pomocou python scriptu (obrázok č.9) využívajúcom bulk api po batchoch o veľkosti 1000. Následne sme skontrolovali (obrázok č.10), či sedí počet dát v elasticsearch pomocou search dopytu.



Obrázok Python script na importovanie záznamov do Elasticsearch

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Kontrola počtu vložených dát

# 7. Uzly

**Otázka:**

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?

**Odpoveď:**

3 uzly – všetko ide

2 uzly – skúsili sme vymazať aj master uzol, aj nie-master uzol .. v oboch prípadoch všetko išlo. Keď sme vypli uzol, ktorý bol master (obrázok č.11), tak iný uzol bol vybraný za mastera (obrázok č.12)

Obrázok, na ktorom je text, snímka obrazovky, monitor, obrazovka

Automaticky generovaný popis

Obrázok Vypnutie master uzla

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Iný uzol zvolený za master

1 uzol – ak spustíme samostatne jeden uzol, tak nefunguje nič, nevieme spraviť ani dopyt na cluster health. Ak spustíme 2 uzly a potom jeden vypneme, cluster taktiež prestáva byť funkčný. Keď sa pozrieme na logy uzlu, vidíme, že na zvolenie mastera treba aspoň 2 uzly.

0 uzlov – nefunguje nič 🐸

Aby zostal cluster k dispozícií, nesmieme stratiť polovicu alebo viacej uzlov, ktoré môžu byť mastermi. V našej konfigurácií to znamená to, na čo sme prišli -> maximálne môžeme stratiť len jeden uzol, 2 musia byť vždy dostupné. Elastic sa dá nastaviť aj bez clusteru, aby fungoval len na jednom uzle. Kvórum v clusteri je potrebné, aby sme zachovali funkčnosť clusteru aj v prípade, že niektoré uzly spadnú. Vďaka tomu, že kvórum musí odsúhlasiť dôležité akcie, vieme zabezpečiť konzistenciu dát a zabrániť prípadnej strate dát. Pri distribuovanej architektúre potrebujeme takýto alebo obdobný mechanizmus pre jej správne fungovanie.

# 8. Script

**Otázka:**

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.

**Odpoveď:**

Najskôr sme si vytvorili script (obrázok č. 13), ktorý budeme používať na inkrementovanie dokumentu. Keďže na funkčnosť clusteru potrebujeme aspoň 2 uzly z 3, vykonáme nasledujúce testy, počas ktorých budeme sledovať zmeny \_seq\_no a \_primary\_term:

1. test -> všetky uzly aktívne

2. test -> vypneme prvý uzol

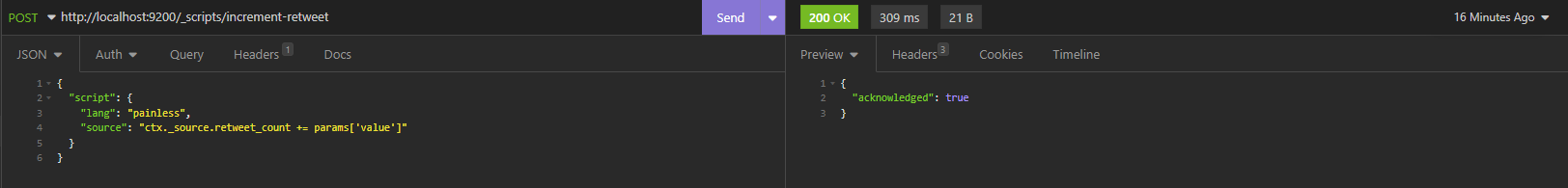
3. test -> zapneme prvý uzol

4. test -> vypneme druhý uzol

5. test -> zapneme druhý uzol

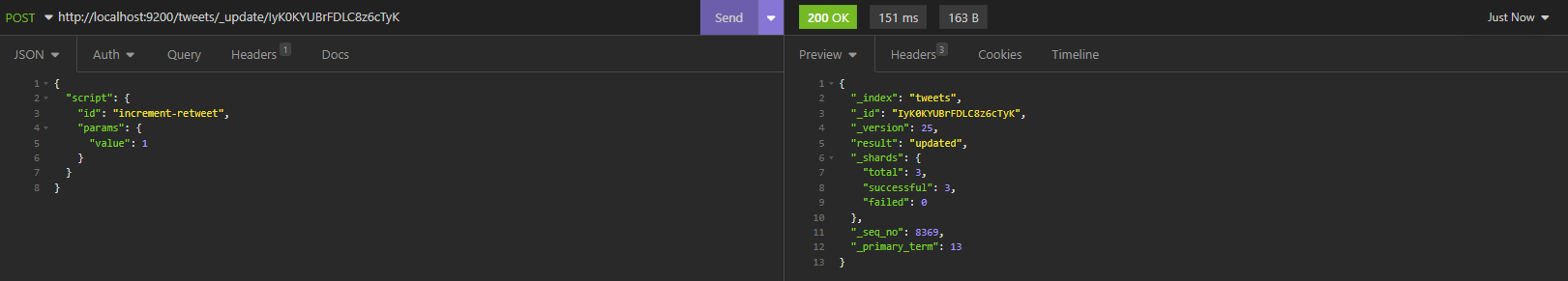
6. test -> vypneme tretí uzol

7. test -> zapneme tretí uzol



Obrázok Vytvorenie scriptu na inkrementáciu

1. test (obrázok č. 14) - \_seq\_no sa nám zväčšuje o jedna, zatiaľ čo \_primary\_term ostáva rovnaké



Obrázok Test 1

2. test (obrázok č.15) - \_seq\_no sa nám zvýšilo o jedna a \_primary\_term sa nám taktiež zvýšilo o jedna, avšak len jedenkrát (od vypnutia)

Obrázok, na ktorom je text, obrazovka, snímka obrazovky

Automaticky generovaný popisObrázok, na ktorom je text, vnútri, obrazovka, snímka obrazovky

Automaticky generovaný popis

Obrázok Test 2

3. test (obrázok č.16) – \_seq\_no sa nám zväčšilo o jedna, zatiaľ čo \_primary\_term ostáva rovnaké

Obrázok, na ktorom je text, prenosný počítač, obrazovka, snímka obrazovky

Automaticky generovaný popis

Obrázok Test 3

4. test (obrázok č.17) – \_seq\_no sa nám zvýšilo o jedna a \_primary\_term sa nám taktiež zvýšilo o jedna, avšak len jedenkrát (od vypnutia)

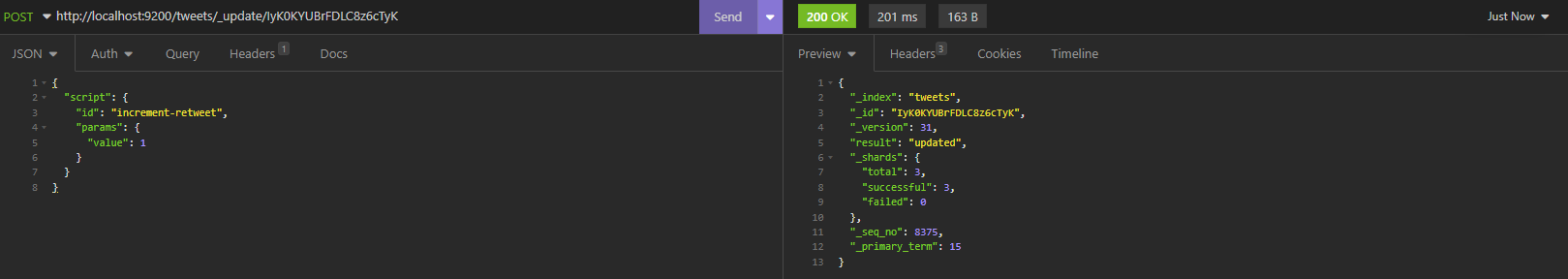
Obrázok, na ktorom je text, obrazovka, snímka obrazovky

Automaticky generovaný popisObrázok, na ktorom je text, obrazovka, snímka obrazovky

Automaticky generovaný popis

Obrázok Test 4

5. test (obrázok č.18) - \_seq\_no sa nám zväčšilo o jedna, zatiaľ čo \_primary\_term ostáva rovnaké



Obrázok Test 5

6. test (obrázok č.19) – \_seq\_no sa nám zvýšilo o jedna a \_primary\_term sa nám taktiež zvýšilo o jedna, avšak len jedenkrát (od vypnutia)

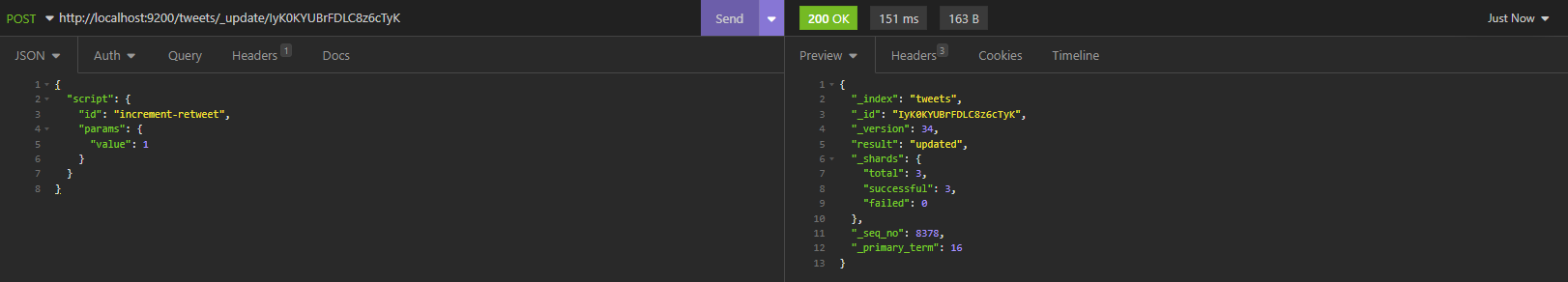
Obrázok, na ktorom je text, prenosný počítač, snímka obrazovky

Automaticky generovaný popis Obrázok, na ktorom je text, vnútri, obrazovka, snímka obrazovky

Automaticky generovaný popis

Obrázok Test 6

7. test (obrázok č.20) - \_seq\_no sa nám zväčšilo o jedna, zatiaľ čo \_primary\_term ostáva rovnaké



Obrázok Test 7

Tieto dve hodnoty nám slúžia ako počítadlá, vďaka ktorým zabezpečíme kontrolu konkurencie (napríklad, aby nám starší update neprepísal novší update). Primary\_term sa inkrementuje vždy keď sa z iného shardu stane primárny, teda pri páde pôvodného primárneho shardu. Seq\_no je jednoduché počítadlo, ktoré sa inkrementuje vždy pri vykonanej operácií.

# 9. Veľký import

**Otázka:**

Zrušte repliky a importujete všetky tweety.

**Odpoveď:**

Bohužiaľ, nepodarilo sa nám vytvoriť query, ktorá by zvládla denormalizovať všetky dáta konverzácií v prijateľnom čase. Vyskúšali sme 3 hlavné typy queries, a to query pomocou subselectov, query pomocou left joinov a na koniec query využívajúcu kombináciu lateral joinov a klasických joinov (obrázok č.21). Prvotné vytvorenie indexov na všetky cudzie kľúče bolo samozrejmosťou. Nakoniec sa nám podarilo vyexportovať viac ako 2 milióny denormalizovaných záznamov do json súboru. Neskôr sme však zistili, že veľmi **úzke hrdlo je pri tomto importe pravdepodobne autorove HDD** .. skúšali sme zbehnúť túto query (pre 5000 záznamov) na inom stroji, patriacemu autorovmu spolužiakovi, kde je schéma uložená na SSD disku .. z pôvodných 4 minút, čo trebalo starému stroju na denormalizáciu 5000 záznamov, sme sa dostali na 10 sekúnd, čo je markantné zrýchlenie okolo 96%. Pri importe všetkých dát to znamená rozdiel medzi 17 hodinami a 430 hodinami.

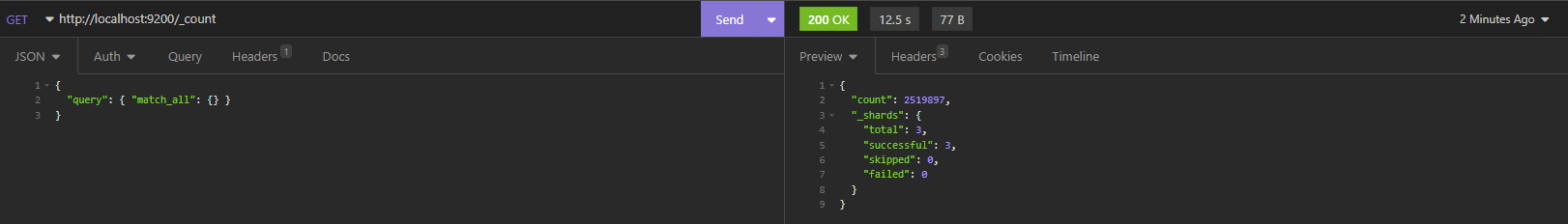
Obrázok, na ktorom je text

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Query pre denormalizáciu dát

V ďalšom kroku sme využili python script z úlohy č.6, aby sme dáta z json súboru dostali na elasticsearch cluster. Museli sme upraviť zopár parametrov, aby viacej zodpovedali importu veľkých dát. Konkrétne sa jednalo o veľkosť posielaného batchu a request timeout (keďže defaultne je nastavený na 10 sekúnd, čo nebolo dostačujúce). Po veľkom importe máme v indexe 2 519 897 dokumentov (obrázok č.22).



Obrázok Počet dokumentov v indexe

# 10. Vyhľadávanie

**Otázka:**

Vyhľadajte vo vašich tweetoch, kde použite function\_score pre jednotlivé medzikroky nasledovne:

a. Must:

i. Vyhľadajte vo viacerých poliach naraz (konkrétne: author.description.shingles (pomocou shingle) – boost 10, content (cez analyzovaný anglický text) spojenie – boost 6 "put1n chr1stian fake jew", zapojte podporu pre preklepy, operátor je OR.

ii. V poly referencies.content slovo “nazi”

iii. Hashtag “ukraine”

b. Filter:

i. vyfiltrujte len tie, ktoré majú author.following\_count > 100, tie ktoré majú author.followers\_count > 100 a tie, ktoré majú nejakú linku

c. Should:

i. Ak sa v context\_annotations.domain.name nachádza “Person” boostinte o 5

ii. Ak sa v context\_annotations.entity.name nachádza “Soros” boostnite o 10

iii. Ak je vyhľadaný string “put1n chr1stian fake jew” aj fráza s tým ze sa môže stat jedna výmena slov boostnite o 5

d. Agregácie:

i. Vytvorte bucket pro-russia ktorý obsahuje hastagy používane Kremľom na propagandu: istandwithputin, racism, 1trillion, istandwithrussia, isupportrussia, blacklivesmatter, racism, racistukraine, africansinukraine, palestine, israel, freepalestine, istandwithpalestine, racisteu, putin

1. Pre neho spravte týždňový histogram, kde pre každý týždeň zobrazte štatistiky

**Odpoveď:**

Naša query (obrázok č.23) je rozdelená na 2 časti -> bool query a agregácie. V bool query máme podľa zadania vyhľadávanie v Must, Filter a Should. V agregácií vytvárame buckety podľa hashtagov. Pri tvorbe bucketov sme si museli pridať do mappingu na hashtagy *"fielddata": true,* keďže ich máme uložené ako text (kvôli analyzérom). Týždňový histogram sa nám nepodarilo spraviť. Skóre relevancie sme menili pomocou ^(číslo) – notácie (pri jednotlivých slovách) a boostu (celých dokumentov). Pre operácie vykonávané nad poľami (respektíve nad nested fieldami) sme používali nested operátor.

Obrázok, na ktorom je text

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popis

Obrázok Vyhľadávacia query