

Leonard Puškáč

PDT - Zadanie #5 - Elasticsearch

Úloha 1

Na vytvorenie troch inštancií elasticu som použil docker. Najprv som si stiahol docker image s príkazom:

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:8.5.3
```

Potom som si vytvoril docker-compose.yml súbor podľa stránky

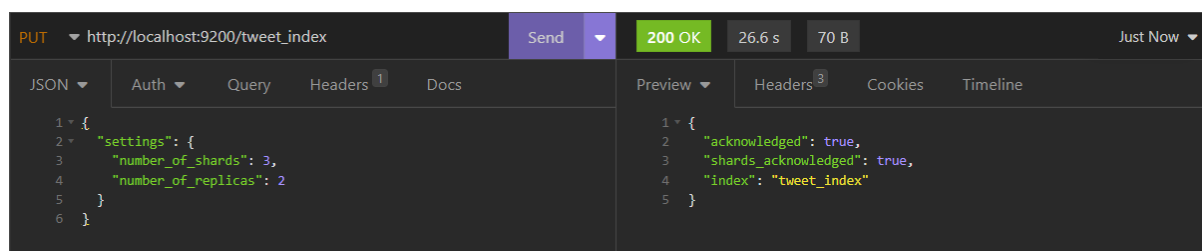
<https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html> a vytvoril

kontainery príkazom:

```
docker compose up
```

Toto vytvorilo jeden kontainer so štyrmi pod-kontainermi - jeden pre kibana a tri pre elastic search.

Úloha 2



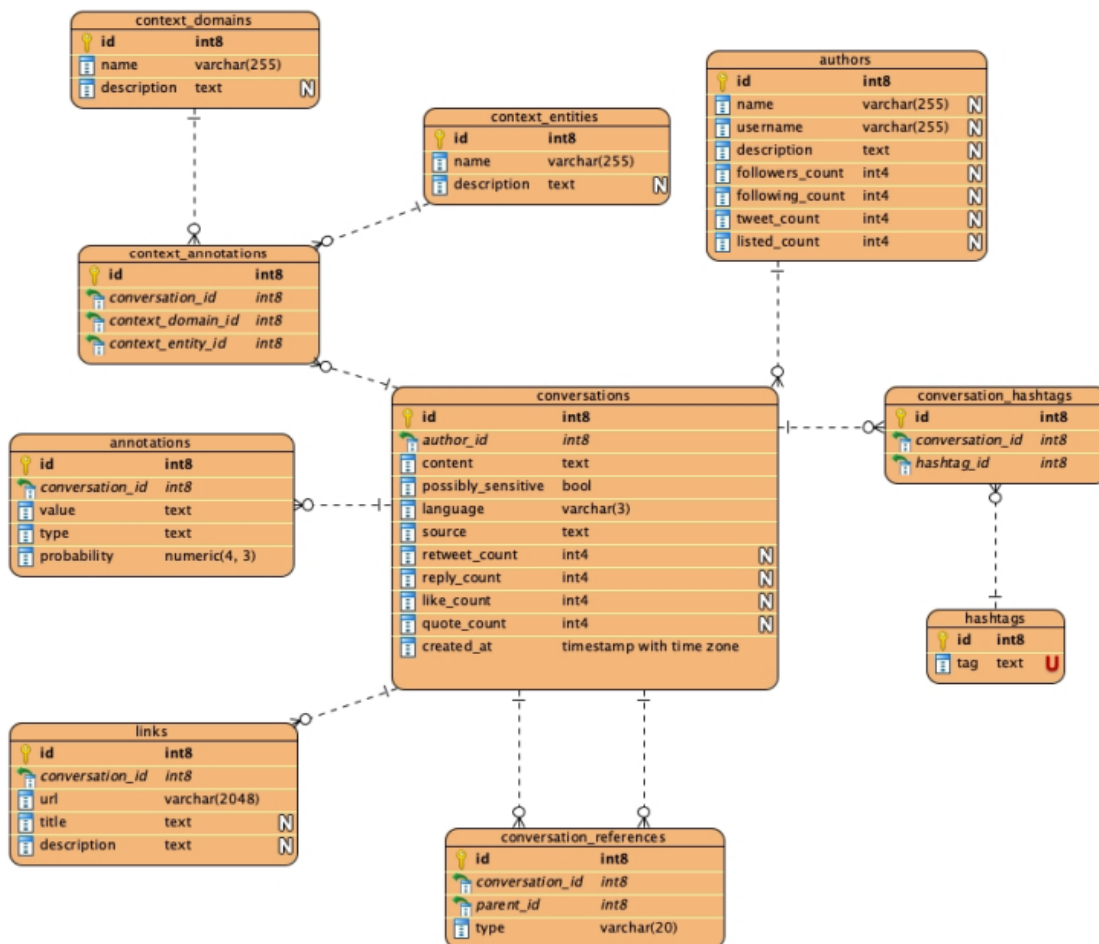
Na vytvorenie indexu som použil metódu PUT s tým, že som request poslal na port 9200.

Tento request som definoval v json. Počet shardov som nastavil na 3, tak aby sa rovnal počtu nodov - nepotrebujeme viac shardov - takýmto spôsobom každý node má jeden shard. Väčší počet shardov by znamenal, že niektorý z nodov by mal viacero shardov, a musel by pri requestoch na ne spájať výsledky zo všetkých svojich shardov.

Počet replík som nastavil na 2.

Úloha 3

Pôvodná schéma tabuliek zo zadania 1:



Podľa tejto schémy som vytvoril mapping zo všetkých tabuliek. Tento mapping obsahuje properties - každá má názov jednej z tabuliek, okrem tabuľky conversation references (ta je nahradená property s názvom "parent_tweet"), a okrem context annotations, aby sme sa zbavili čisto relačnej tabuľky - môžeme proste uložiť všetky relevantné entities a domains do daného dokumentu.

Tie tabuľky ktoré sú v one-to-many relationship s daným tweetom, sú definované v mappingu ako "nested" property.

Celý post request je uložený v súbore create_mapping.json

```

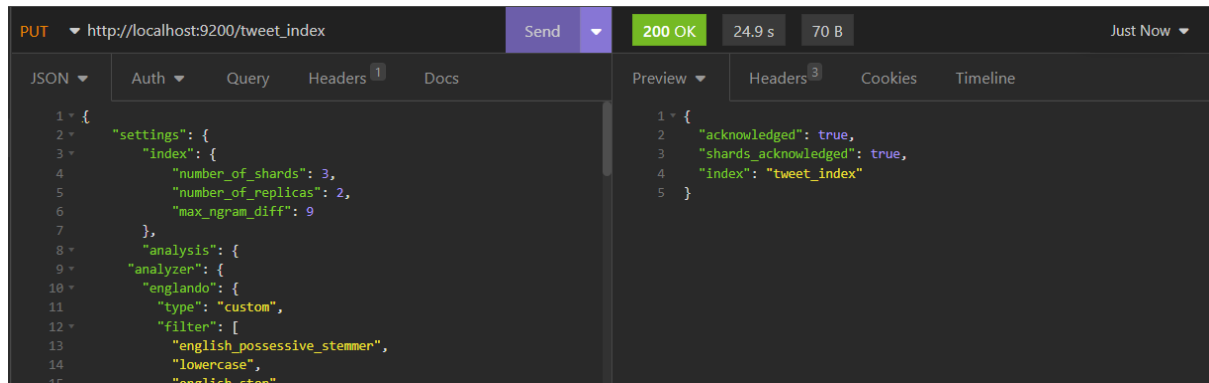
POST http://localhost:9200/tweet_index/_mapping
200 OK 15.7 s 21 B Just Now

JSON
1 {
2   "dynamic": "strict",
3   "properties": {
4     "author": {
5       "properties": {
6         "id": {
7           "type": "long"
8         },
9         "name": {
10          "type": "text",
11          "fields": {
12            "keyword": {
13              "type": "keyword"
14            }
15          }
16        },
17        "username": {
18          "type": "text"
19        }
20      }
21   }
22 }

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

Úloha 4

Keďže treba pridať analyzéry, index vytvorený v úlohe 2 spolu s mapping som vymazal, a vytvorím nový s rovnakým mappingom (plus analyzery na niektore fields).



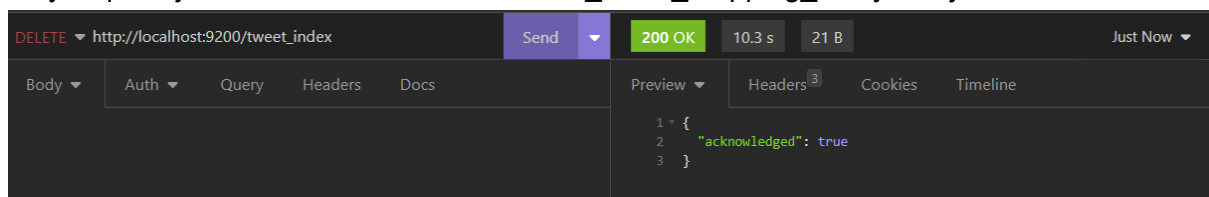
```
PUT http://localhost:9200/tweet_index
{
  "settings": {
    "index": {
      "number_of_shards": 3,
      "number_of_replicas": 2,
      "max_ngram_diff": 9
    }
  },
  "analysis": {
    "analyzer": {
      "englando": {
        "type": "custom",
        "filter": [
          "english_possessive_stemmer",
          "lowercase",
          "english_stop"
        ]
      }
    }
  }
}
```

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

Následne som vytvoril nový index s rovnakým názvom, s tým že som pridal požadované analyzéry, použitím PUT requestu. Pri author.description a tweet_info.description som pridal analyzér englando, a pri hashtags som pridal normalizer na lowercase. author name, a username teraz majú field pre custom_ngram, a name a description majú field pre custom_shingles (name má tým pádom obidva).\

Oproti úlohe 3 som ešte spravil zmenu, čo sa týka conversation_references - uvedomil som si, že tam je treba tiež spraviť nested pretože pre tweet existuje viacero referencií.

Celý request je možné vidieť v súbore create_index_mapping_analysers.json



```
DELETE http://localhost:9200/tweet_index
```

```
{
  "acknowledged": true
}
```

Úloha 5

Na denormalizovanie dát z tabuliek som si vytvoril jeden veľký sql dopyt ktorý spája všetky tabuľky do jednej - tento sql dopyt je v súbore denormalize_tables.sql.

Bohužiaľ som toto zadanie začal robiť veľmi neskoro, a pri zbiehání denormalizácie som zisťil, že nemám dostatočný disk space... takže rip.