

DerbOven Maxim

IAO3

Elasticsearch

data persistency

Inhoud

[2 Wat is Elasticsearch 2](#_Toc118297029)

[2.1 Inleiding 2](#_Toc118297030)

[2.2 Hoe werk de Elastic stack 2](#_Toc118297031)

[2.3 Opzetten van Elasticsearch & Kibana 2](#_Toc118297032)

[2.4 Match Query 2](#_Toc118297033)

[2.5 Multi-Match Query 3](#_Toc118297034)

[2.6 Match Phrase 3](#_Toc118297035)

[2.7 Exists Queries 4](#_Toc118297036)

[2.8 Range Queries 4](#_Toc118297037)

[2.9 Ids Queries 5](#_Toc118297038)

[2.10 Wildcard Queries 6](#_Toc118297039)

[2.11 Fuzzy 6](#_Toc118297040)

[2.12 3. Boosting 8](#_Toc118297041)

[2.13 4. Sorting 8](#_Toc118297042)

[2.13.1 4.1 Default Sorting 8](#_Toc118297043)

# Wat is Elasticsearch

## Inleiding

Elasticsearch kan voortdurend grote volumes informatie behandelen en automatisch schalen. Het is een nosql, json based data store. Je communiceert met elasticsearch met een RESTFUL API.

|  |  |
| --- | --- |
| Relationeel Database | ElasticSearch |
| Database | Indexes |
| Tables | Patterns/Types |
| Rows | Documents |
| Columns | Fields |

## Hoe werk de Elastic stack (Inverted Index)

Inverted Indexes onderscheiden elastic search van andere sezarch engines. Een inverted index mapt tekst op basis van zijn inhoud.

Om dit in een voorbeeld te verwerken nemen we even volgende 3 documenten:

1. Food is great
2. It is raining
3. Wind is strong

De index voor volgende documenten zou als volgt zijn

|  |  |  |
| --- | --- | --- |
| **Term** | **Frequency** | **Documents** |
| Food | 1 | 1 |
| Great | 1 | 1 |
| is | 3 | 1,2,3 |

De termen worden alfabetisch opgeslagen zodat ze gemakkelijk gevonden kunnen worden.

Zoeken op meerdere termen is gedaan door een lookup te doen in de index. Het voert een UNION of INTERSECTION uit en haalt de relevante documenten op. Zulke indexes zijn verspreid over verschillende shards.

De ES index bestaat uit verschillende Lucene indexes, waarop deze weer bestaan uit index segmenten. De index files zijn niet aanpasbaar (behalve voor het verwijderen).

### Analyzers

Analyseren is het proces om tekst te om te zetten in tokens of termen die worden toegevegd aan de inverted index. Een analyzer bestata uit volgende delen:

Character filter: Krijgt de originele tekst als stream en bewerkt deze door het toevoegen, verwijderen of veranderen van individuele characters.

Tokenizers: krijt een stream van charachter ens breekt deze op in verschildende tokens

Token filters: krijgt de tokens en veranderd hier ook nog een laatste keer iets waar nodig.

Soorten analyzers:

Standaard: Split tekst in termen op basis van woord in lower case.

Keyword: geeft de gehele tekst terug zoals hij hem kreeg.

### Basis Concepten

**Index**

Collectie vzn JSON documents. Elk vzan deze documents zitin een index.

**Shards**

Omdat er geen limiet staat op het aantal documenten dat zich in een index kan bevinden, worden indexes vaak horizontaal gepartitooneerd over shards. Deze bevinden zich als nodes in een cluster.

**Analyzer**

Worden gebruikt zijdens het zoeken van documenten. Deze bevatten tokenizers die zinnen en tekst opsplitsen in tokens en token filters

**Mapping**

Combinatie van velden en analyzers Het bepaald hoe velden gestockeerd en geindexed worden.

# Opzetten van Elasticsearch & Kibana

## Containers opzetten

Op de website staat duidelijk aangegeven hoe je het kan opzetten met docker.

1. Elasticsearch container opzetten met de volgende commands;

docker network create elastic

docker pull docker.elastic.co/elasticsearch/elasticsearch:8.4.3

docker run --name elasticsearch --net elastic -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" -t docker.elastic.co/elasticsearch/elasticsearch:8.4.3

1. Start Kibana

docker pull docker.elastic.co/kibana/kibana:8.4.3

docker run --name kibana --net elastic -p 5601:5601 docker.elastic.co/kibana/kibana:8.4.3

1. Vervolgens naar de kibana webGUI surfen via localhost:5601. Daar moet je de enrollment  token en password ingeven die in de terminal verscheen bij het instellen van de elasticsearch container.

## Data inladen

Data inladen kan gemakkelijk via de kibana interface met een JSON of CSV.

# Soorten Queries

Elasticsearch heeft een ruim aanbod aan zoekfunctionaliteiten. Ik bespreek hieronder de voornaamste en pas ze in het voorbeeld toe op onze hierboven gemaakte dataset.

## Match Query

We gaan op zoek naar een bepaalde waarde in een bepaald veld met volgende query

|  |
| --- |
| POST test/\_search  {  "query": {  "match": {  "phrase": {  "query" : "testzoekopdracht"  }  }  }  } |

Wanneer er bij query meerdere woorden voorkomen gaat hij automatisch de OR operator gebruiken. Om er voor te zorgen dat alle woorden moeten voorkomen kan dat met volgende query:

|  |
| --- |
| POST test/\_search  {  "query": {  "match": {  "phrase": {  "query" : "testzoekopdracht tweede derde",  "operator" : "AND"  }  }  }  } |

Met "minimum\_should\_match" kunnen we bepalen hoeveel woorden uit de query er minstens moeten voorkomen in het veld.

|  |
| --- |
| POST test/\_search  {  "query": {  "match": {  "phrase": {  "query" : "testzoekopdracht tweede derde",  "minimum\_should\_match": 2  }  }  }  } |

## Multi-Match Query

Met een multimatch query kunnen we de match functionaliteiten toevoegen om te zoeken op meerdere velden.

|  |
| --- |
| POST test/\_search  {  "query": {  "multi\_match": {  "query" : "zoekopdracht1 zoekopdracht2"  , "fields": ["field1","field2"] }  }  } |

## Match Phrase

Met Match phrase kan je een volledige zin, inclusief de volgorde van de woorden matchen.

Optioneel: met de SLOP parameter kunnen we bepalen hoeveel de woorden mogen verwisselen van plaats om alsnog een match te vinden.

|  |
| --- |
| POST test/\_search  {  "query": {  "match\_phrase": {  "phrase": {  "query": "dit is een zin”,  "slop": 1  }} |

## Exists Query

Als we alle documenten willen zoeken die een bepaald veld bevatten kunnen we gebruiken maken van ‘exists’

|  |
| --- |
| GET employees/\_search  {  "query": {  "exists": {  "field": "company"  }  }  } |

## Range Query

Met een range query kun je bepaalde velden filteren op range. In het voorbeeld wordt er gezocht op een veld met een waarde tussen 5 en 10

|  |
| --- |
| POST employees/\_search  {  "query": {  "range" : {  "experience" : {  "gte" : 5,  "lte" : 10  }  }  }  } |

### Ranges (GTE, GT, LTE, LT)

|  |  |
| --- | --- |
| GTE | Groter dan of gelijk aan (Greater Than or Equal to) |
| GT | Groter dan (Greater Than) |
| LTE | Kleinder dan of gelijk aan (Less Than or Equal to) |
| LT | Kleinder dan (Less Than) |

## Ids Query

Soms willen we documenten verkrijgen op basis van hun ID, dit kan gemakkelijk met volgende query

|  |
| --- |
| GET indexname/typename/documentId |

Hiermee kunnen we echter maar 1 document met dat Id opvragen, waneer we op meerdere Ids willen matchen moeten we gebruik maken van volgende query.

|  |
| --- |
| POST employees/\_search  {  "query": {  "ids" : {  "values" : ["1", "4"]  }  }  } |

## Wildcard Query

Deze query staat toe om te zoeken op bepaalde velden met een wildcard.

|  |
| --- |
| GET employees/\_search  {  "query": {  "wildcard": {  "country": {  "value": "c\*a"  }  }  }  } |

### Wildcard Opties

|  |  |
| --- | --- |
| \* | Staat voor 1,0 of meer characters |
| . | Staat voor 1 charachter |

## Fuzzy

Een fuzzy query kan gebruikt worden om documenten terug te geven die hard lijken op de zoekterm. Dit kan gebruikt worden om spelling fouten recht te zetten.

|  |
| --- |
| GET employees/\_search  {  "query": {  "fuzzy": {  "country": {  "value": "Chnia",  "fuzziness": "2"  }  }  }  } |

## Filter Query

Elastic search haalt documenten op die aan de criteria voldoenen sorteert deze op basis van de score. Wanneer de score niet relevant is kunnen we gebruiken van een filter query.

|  |
| --- |
|  |

## Boosting

## 3. Boosting

While querying, it is often helpful to get the more favored results first. The simplest way of doing this is called boosting in Elasticsearch. And this comes in handy when we query multiple fields. For example, consider the following query:

POST employees/\_search

{

"query": {

"multi\_match" : {

"query" : "versatile Engineer",

"fields": ["position^3", "phrase"]

}

}

}

This will return the response with the documents matching the “position” field to be in the top rather than with that of the field “phrase”.

Points

## 4. Sorting

### 4.1 Default Sorting

When there is no sort parameter specified in the search request, Elasticsearch returns the document based on the descending values of the “\_score” field. This “\_score” is computed by how well the query has matched using the default scoring methodologies of Elasticsearch. In all the examples we have discussed above you can see the same behavior in the results.  
It is only when we use the “filter” context there is no scoring computed, so as to make the return of the results faster.