Advanced Elastic search Tips and Tricks

This is Heavy tech post, I will talk about most common tips and tricks of elasticsearch , putting together different points which are very important when you are using elastic search for your enterprise projects.

## Disable Doc\_Values when you don’t need them

Doc\_Values is column-oriented store in elastic search and it is enabled by default, created at index-time when a field is indexed, Elasticsearch adds the tokens to the inverted index for search.In another word contains same value as the \_source [Original Json Document] with different structure [column-oriented].

Doc\_values are used in several places such as Sorting, Aggregation and script which Column-oriented is way more efficient.

[\*] If you don’t need to sort or aggregation, or access the field value from a script, you HAVE to disable doc values in order to save disk space.

How to:

PUT demo\_index

{

"mappings": {

"\_doc": {

"properties": {

"username": {

"type": "keyword"

},

"session\_id ": {

"type": "keyword",

"doc\_values": false

}

}

}

}

}

## Synonymous

The concept of Synonymous is very interesting, handy and enriches the document in order to be retrieved with different combination of terms.

For example:

USA -> U.S.A

Sr. -> Senior

Dev -> Developer, Software Developer

…………………………………………………………………………..

Now Let’s imagine we would like to search “USA” and get all documents contains either U.S.A or USA, in order to solve this problem we have few options

1. convert a query for USA into a query for USA OR “U.S.A”. at search time
2. we replace USA with the two terms usa and “U.S.A, Using the synonym token filter in index time, then at search time we can find those terms.

Both solutions are valid but the first will drive us to have multiple responsibilities at application level and in my opinion it is not good practice.

With the second option we have centralized search engine will manage everything for us, converting the terms and make it available for search ☺

How to

PUT /vergnaty\_index

{

"settings": {

"analysis": {

"filter": {

"special\_synonyms": {

"type": "synonym",

"synonyms": [

"USA,U.S.A",

"Sr.,Senior"

]

}

},

"analyzer": {

"my\_synonyms": {

"tokenizer": "standard",

"filter": [“lowercase”, "special\_synonyms"]

}

}

}

}

}

GET /vergnaty\_index/\_analyze

{

"analyzer" : "my\_synonyms",

"text" : "Sr. Software developer in USA"

}

## Using Filtered Aliases with Routing

Aliases with filters provide an easy way to create different "views" of the same index, all documents

Are routed to a particular shard in an index and by default is used the document’s \_id.

Having Filtered Aliases combined with \_routing can reduce the impact of searches. Instead of having to fan out a search request to all the shards in an index, the request can be sent to just the shard that matches the specific routing values.

Ex

[]

## Document Relevance using Functional Score and Boosting

The theory behind the document relevance in elasticsesarch is managed by Lucene using Boolean Model, TF/IDF (Term Frequency / Inverse Document Frequency) ,the vector space model and field-length norm.

Using all thus Models Lucene’s Practical scoring function will calculate the score of each document that are retrieved by the query.

Why Functional Score or Boosting?

Both of them allow having custom made document relevance, in some case scenarios they could be useful to be applied.

Let’s say we are looking for Senior Software Developers and (Contractor or Permanent), but we would like to give priority to Permanent then to the Contractor, here comes useful having custom Score and this can be done using functional score.

By definition The function\_score allows you to modify the score of documents that are retrieved by a query. A score function is computationally expensive and it is sufficient to compute the score on a filtered set of documents.

Example []

Boosting is applied in different manner which makes prioritizing clauses at search time to give one query more importance than another.

Let’s imagine that we have documents with Title, Author and Content, using Boost we can give important to the Title than author and content. Remember there is no formula for deciding on the correct boost value for particular clause instead change the boost and check again since you don’t get what you want.

There is another factor you need to care as well, which is field-length norm, for instance, in the above example, let’s say we have just Title and Content so no more author, the title field will probably already have a “natural” boost over the content field thanks to the [field-length norm](https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html#field-norm) (titles are usually shorter than the related content), so don’t blindly boost fields just because you think they should be boosted.