Fast Searching using OpenSearch with FastAPI

Aldion Amirrul

Surabaya 08 February 2025

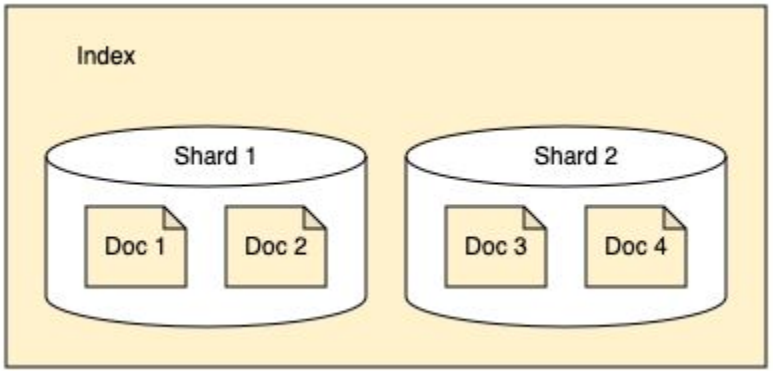
surabaya.py

What ?

- An open-source project, distributed search and analytics engine.
- Using for do a **full-text search**, **aggregations**, and **real-time analytics.**
- First initialize developed by AWS.
- Is not Elasticsearch but originally forked from it !
- 100% Free !!

surabaya.py

What ?



surabaya.py

Why ?

- Opensearch itself is designed to **handle millions of records efficiently.**
- Full-text search with **ranking & relevance scoring**.
- Autocomplete, typo tolerance, **fuzzy search**.
- Can also handle **filters, aggregations, and complex queries**.
- It’s free.

surabaya.py

Hmm.. another reason is:

surabaya.py

Let’s see by an example:

Approximately data +10 Million

```
SELECT
    p.title,
    MATCH(p.title) AGAINST ('oknum polisi' IN NATURAL LANGUAGE MODE) AS score
FROM posts p
LEFT JOIN post_tag pt ON p.id = pt.post_id
LEFT JOIN tags t ON pt.tag_id = t.id
WHERE
    p.description IS NOT NULL
    AND p.title IS NOT NULL
    AND MATCH(p.title) AGAINST ('oknum polisi' IN NATURAL LANGUAGE MODE)
    AND t.slug IN ('viral', 'polisi')
ORDER BY score DESC;
```

surabaya.py

Let’s see by an example:

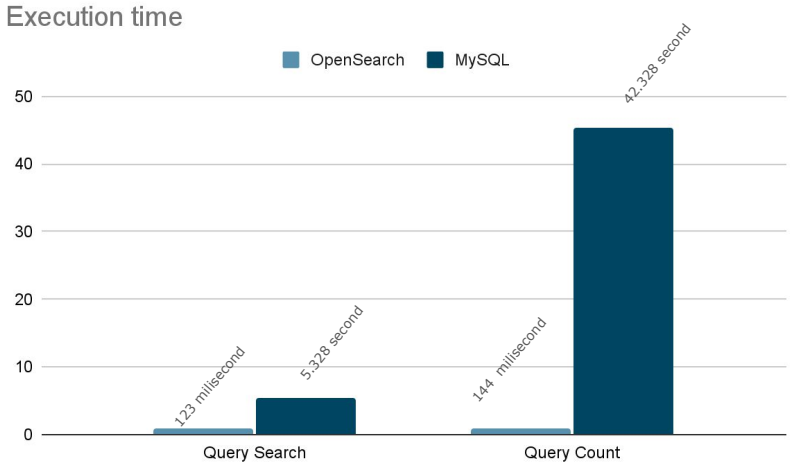
Against

```
GET /articles/_search
{
    "_source": "title",
    "size": 100,
    "query": {
        "bool": {
            "must": [
                {"exists": {"field": "description"}},
                {"exists": {"field": "title"}},
                {
                    "multi_match": {
                        "query": "oknum polisi",
                        "fields": ["title^2"],
                        "type": "best_fields"
                    }
                }
            ],
            "filter": [
                {"terms": {"tags.slug": ["viral", "polisi"]}}
            ]
        }
    },
    "sort": [
        {"_score": "desc"}
    ]
}
```

surabaya.py

Let’s see by an example:

Searching performance:



surabaya.py

When ?

Common study cases:

- E-commerce, for product catalogue search.
- News or Wiki portal, for article search.
- General - Application, log or website monitoring.
- BI (Business Intelligence) - for tracking sales trend ? or maybe customer insight, etc.

surabaya.py

Fast API ?

Fast API ?

- Easy implementation.
- Built-in documentation.
- They said it fast.

OpenSearch Python Library

- Low-level Python client
- High-level Python client
- Machine-learning Python

Low-level Python client

“The OpenSearch low-level Python client (`opensearch-py`) provides wrapper methods for the OpenSearch REST API so that you can interact with your cluster more naturally in Python.”

surabaya.py

Low-level Python client

```python
from opensearchpy import OpenSearch

q = 'Voluptates'
query = {
  'size': 5,
  'query': {
    'multi_match': {
      'query': q,
      'fields': ['title^2', 'description']
    }
  }
}

response = client.search(
    body = query,
    index = 'article-index'
)
```

main.py

High-level Python client

“The OpenSearch high-level Python client (`opensearch-dsl-py`) provides wrapper classes for common OpenSearch entities, like documents, so you can work with them as Python objects.”

surabaya.py

High-level Python client

```python
from opensearch_dsl import Document, Text, Keyword

class Article(Document):
    title = Text(fields={'raw': Keyword()})
    description = Text()

    class Index:
        name = "article-index"

    def save(self, ** kwargs):
        return super(Article, self).save(** kwargs)
```

main.py

```python
from opensearch_dsl import connections
from models.article import Article

query = Article.search(using="default").query("match", title="Voluptates")
response = query.execute()
```

models/article.py

surabaya.py

Lets try ..

surabaya.py

Conclusion

surabaya.py

Long story short ….

~ ~ ~ ~Thank you ……

surabaya.py

References:

https://opensearch.org/docs/latest/getting-started/intro/
https://fastapi.tiangolo.com/#example
https://ctaverna.github.io/opensearch-data-compression/