9 - Term-level search

- Buscas de match exato
- Para dados estruturados.

OVERVIEW OF TERM-LEVEL SEARCH

- E.g.procura datas e números
- Essas buscas não envolvem o analyzer:
- Computacionalmente mais barato
- A não ser que usemos um normalizer.
- Engine fará a busca pelo termo exato no índice invertido, sem modificações.
- Ideal para busca tipo keyword
    - Busca de termos exatos.

THE TERM QUERY

- Term não é analisado, é procurado direto no índice invertido.

Listing 9.1 Fetching movies with a given rating

```
GET movies/_search
{
  "query": {
    "term": {
      "certificate": "R"
    }
  }
}
```

The term query declaration

- Maiúsculo ou minúsculo é importante, pode alterar resultado (pois não passa por analyzer)
- Pouco recomendado para buscar em text fields, exceto se o campo já for previsto para isso.
- Sintaxe não-simplificada:

Listing 9.3 Full syntax of a query

```
GET movies/_search
{
  "query": {
    "term": {
      "certificate": {
        "value": "R",
        "boost": 2
      }
    }
  }
}
```

The certificate field has values enclosed in an object.

The certificate's search criteria

In addition to the value, we can provide parameters like boost.

THE TERMS QUERY

- Busca múltiplos critérios de uma vez (operador OR)

Listing 9.4 Searching for multiple search criteria in a field

```
GET movies/_search
{
  "query": {
    "terms": {
      "certificate": ["PG-13","R"]
    }
  }
}
```

The terms query expects an array of search criteria.

Multiple search criteria against a single field

-

- Há um setting (pode ser alterado dinamicamente, sem parar o index) para alterar o número máximo de termos permitidos na busca
    - default é 2^16.
- Exemplo de busca a partir de um resultado de um documento no lugar de um termo pré-definido:

Listing 9.8 A `terms` lookup search

```
GET classic_movies/_search
{
  "query": {
    "terms": {
      "director": {
        "index":"classic_movies",
        "id":"3",
        "path":"director"
      }
    }
  }
}
```

A terms query (with a twist!)

The field that we are interested in searching against

The index field denotes the name of the index where the document resides.

Name of the field containing the terms for the query

Search field in the current document

-

THE IDS QUERY

- Busca direto por Ids dos documentos.

Listing 9.9 Fetching multiple documents using an `ids` query

```
GET movies/_search
{
  "query": {
    "ids": {
      "values": [10,4,6,8]
    }
  }
}
```

Name of the query

Provides the document IDs as an array

-

- Podemos fazer a mesma busca fazendo uma terms query que busca por dados "*_id*"

THE EXISTS QUERY

- Verifica se um campo existe antes de fazer a busca de fato
  - Pode economizar tráfego.
  - Só retorna documentos em que um dado campo existe.

Listing 9.11 Running an `exists` query to check whether a field exists

```
GET movies/_search
{
  "query": {
    "exists": {
      "field": "title"
    }
  }
}
```

Defines the query type as exists

Provides the field that we want to check in the document

- Também pode ser útil para buscar documentos que **não** contêm um dado campo, usando *must_not*

THE RANGE QUERY

- Procura valores dentro de um range

Listing 9.13 Fetching movies with a specified range of ratings

```
GET movies/_search
{
  "query": {
    "range": {
      "rating": {
        "gte": 9.0,
        "lte": 9.5
      }
    }
  }
}
```

- Podemos fazer operações com datas:
  -
```
GET movies/_search
{
  "query": {
    "range": {
      "release_date": {
          "gte": "now-1y"
      }
    }
  }
}
```

Fetches all the movies from last year: the current date (represented as now) minus one year

THE WILDCARD QUERY

- Busca por padrões

Listing 9.15 Wildcard search in action

```
GET movies/_search
{
  "query": {
    "wildcard": {
      "title": {
        "value": "god*"
      }
    }
  }
}
```

The wildcard query type

Searches the value field with a wildcard operator

-
- Queries aqui também não são analisadas. Lowercase é importante.
- Para buscar wildcard que subsituem apenas um único caractere, usamos "?" no lugar de "*".:

Table 9.2 Types of wildcards

Character	Description
* (asterisk)	Searches for zero or more characters
? (question mark)	Searches for a single character

-
- Este tipo de query é computacionalmente cara
  - Este tipo pode ser desabilitado nos settings no cluster.:

```
PUT _cluster/settings
{
  "transient": {
    "search.allow_expensive_queries": "false"
  }
}
```

THE PREFIX QUERY

- Como wildcard mas apenas para prefixos

Listing 9.17 Using a `prefix` query

```
GET movies/_search
{
  "query": {
    "prefix": {
      "actors.original": {
        "value": "Mar"
      }
    }
  }
}
```

Specifies the prefix type query

Queries for words that start with "Mar"

- Versão abreviada:

Listing 9.18 Shortening `prefix` query usage

```
GET movies/_search
{
  "query": {
    "prefix": {
      "actors.original":  "Leo"
    }
  }
}
```

- Também é uma operação cara.
    - Para mitigar isso, podemos definir "index_prefixes" quando criamos um index.
        - Escolhemos qual campo vai ser o alvo dos index_prefixes
        - ES então indexa prefixos daquele campo de tamanho de 2 a 5 (por default)
            - E.g. title="Gladiador" preindexa os prefixos ["Gl","Gla","Glad","Gladi"]
            - Acelera a busca de prefixos posteriores
            - Tamanho dos prefixo preindexados pode ser customizado no mappings
                - Límite absoluto mínimo é maior que 0 e máximo menor que 20.

FUZZY QUERIES

- Buscas não-exatas de termos.
- Perdoa erros de ortografia e digitação.
- Baseado na distância de Levenshtein
    - A.k.a "Edit Distance"
- Em queries fuzzy, podemos definir a *fuzziness* (edit distance) desejada, para saber quão flexível a busca será feita.

Listing 9.22 A `fuzzy` query in action

```
GET movies/_search
{
  "query": {
    "fuzzy": {
      "genre": {
        "value": "rama",
        "fuzziness": 1
      }
    }
  },
  "highlight": {
    "fields": {
      "genre": {}
    }
  }
}
```

- Podemos também deixar o fuzziness no "auto":
  - É o default quando não definimos nenhum *fuzziness*
  - ES define o grau de fuzziness dependendo do tamanho do termo procurado.
    - 0 a 2 - fuzziness=0
    - 3 a 5 - fuzziness=1
    - >5 - fuzziness=2

Summary

- Term-level searching is carried out on structured data such as numbers, keywords, Booleans, and dates.
- A term-level search produces a binary result: the search leads to a result or none. There is no *likely match* scenario.
- Term-level queries are not analyzed, meaning that when applied to `text` fields undergoing text analysis during indexing, they can yield incorrect or no results.
- There are many term-level search queries: `term`, `terms`, `prefix`, `range`, `fuzzy`, and others.
- A `term` query searches for a single term against a field, while a `terms` (plural) query search for multiple values against a single field.
- A `range` query helps search data within a set of bounds: for example, searching for crimes that happened in London in the last month.
- A `wildcard` query uses `*` and `?` operators to fetch results.
- Fuzziness employs Levenshtein's edit distance to fetch similar-looking words. Elasticsearch employs `fuzzy` queries to support the user's spelling inconsistencies.
- A `prefix` query retrieves results given a prefix (no need to specify a wildcard operator). As prefix operations are expensive to execute on a live index, we can ask Elasticsearch to pre-create them at index time to avoid finding them during a live query phase.