11 - Compound Queries

- Fazer queries mais complexas com múltiplos critérios simultâneos não-triviais

COMPOUND QUERIES

- Cinco tipo de compound query:

Compound query	Description
Boolean (`bool`) query	A combination of conditional clauses that wraps individual leaf (term and full-text) queries. Works similarly to the `AND`, `OR`, and `NOT` operators. Example: products= TV AND color = silver NOT rating < 4.5 AND brand = Samsung OR LG
Constant score (`constant_score`) query	Wraps a `filter` query to set constant scores on results. Also helps boost the score. Example: Search for all TVs with user ratings higher than 5 but set a constant score of 5 for each result regardless of the search engine's calculated score.
Function score (`function_score`) query	User-defined functions to assign custom scores to result documents Example: Search for products, and if the product is from LG and is a TV, boost the score by three (via a `script` or `weight` function).
Boosting (`boosting`) query	Boosts the score for positive matches while negating the score for non-matches Example: Fetch all TVs but lower the score of those that are expensive.
Disjunction max (`dis_max`) query	Wraps several queries to search for multiple words in multiple fields (similar to a `multi_match` query) Example: Search for smart TVs in two fields (say, `overview` and `description`) and return the best match.

THE BOOLEAN (BOOL) QUERY

- junta critérios booleanos:
- Quatro termos possíveis

Clause	Description
`must`	An `AND` query where all the documents must match the query criteria Example: Fetch TVs (`product` = `TV`) that fall within a specific price range
`must_not`	A `NOT` query where none of the documents match the query criteria Example: Fetch TVs (`product` = `TV`) that fall within a specific price range *but* with an exception, such as not being a certain color
`should`	An `OR` query where one of the documents must match the query criteria Example: Search for fridges that are frost-free *or* energy rated above C grade.
`filter`	A `filter` query where the documents must match the query criteria (similar to the `must` clause), but the `filter` clause does not score the matches Example: Fetch TVs (`product` = `TV`) that fall within a specific price range (but the score of the documents returned will be zero)

- É possível fazer compound queries (ou queries leaf (simples))dentro deo compound queries:

```
GET books/_search
{
  "query": {
    "bool": {
      "must": [
          { "match": {"FIELD": "TEXT"}},
{ "term": {"FIELD": {"value": "VALUE"}}}
      ],
      "must_not": [
        {"bool": { "must": [{}]}}}
      ]
      "should": [
          { "range": { "FIELD": {"gte": 10,"lte": 20}}},
{ "terms": { "FIELD": [ "VALUE1", "VALUE2" ]}}
      ]
    }
  }
}
```

  - Match query é um caso específico da compound query bool
  - bool { must { match }}
- Vários exemplos
- Podemos usar o should para boost de score, junto com must e must_not
  - A princípio should não filtra nada quando junto de must
  - Podemos definir um threshold mínimo de matches para o should retornar um resultado, neste caso o shuld acaba filtrando sim.
    - *minimum_should_match*
  - Sem nenhum must, should serve de filtro (default de minimum_should_match vai para 1, em vez de 0)
- Filter clause é como o must, mas pula a etapa de scoring, mais rápido.
  - É comum combinar filter com must
    - Não altera o score que o must determinou
- Podemos atribuir nomes para as leaf queries dentro de uma compound query
  - Resultado retorna então quais queries foram usadas para o resultado, e exclui quais não foram

CONSTANT SCORES

- Serve para atribuir um score fixo não-zero aos resultados, sem passar pela função de scoring. Por exemplo para atribuir um score não-zero aos resultados de um filter, que a princípio não gera score.
- Pode servir para compor com os resultados de uma query com must e fazer o ranqueamento melhor.

THE BOOSTING QUERY

- altera os scores dos resultados de uma query

- Tem duas partes:

  - Positiva
    - Retorna os matches da parte positiva

    - Negativa
        - Diminui os scores que atendem um certo critério

- Multiplica score por um coeficiente.
- e.g. diminuir pela metade o score de TVs com preço maior que $2500
  THE DISJUNCTION MAX (DIS_MAX) QUERY
- São várias leaf queries em paralelo
- multi_match usa esse por trás.
- Também suporta tiebreaker para usar os não-best fields como desempate

THE FUNCTION_SCORE QUERY

- Cria scoring customizado, como random ou baseado em um script, ignore o scoring default da busca.
- random_score
    - randômico
    - suporta seed (junto de field) para reprodutibilidade
- script_score
    - Podemos criar um score baseado nos valores dos campos, não necessariamente relacionado com a relevância da query em si.

Listing 11.28 Multiplying the field's value by an external parameter

```
GET products/_search
{
  "query": {
    "function_score": {
      "query": {
        "term": {
          "product": "tv"
        }
      },
      "script_score": {
        "script": { #B
          "source":"_score * doc['user_ratings'].value * params['factor']",
          "params": {
            "factor":3
          }
        }
      }
    }
  }
}
```

The script_score function holds the key to generating a score based on its defined script.

The script object

Passes the external parameters to the script

The source is where we define our logic.

- field_value_factor
    - Simplesmente atribui um outro campo como score (e.g. "user rating")
    - Dispensa script
    - Podemos modificar o score por um coeficiente ou elevando ao quadrado, por exemplo
- Podemos combinar diversos scores diferenets para compor um score final.
    - Por default esses scores se multiplicam
        - Podemos alterar com parâmetro "score_mode"
    - Igualmente para boost, boost default é de multiplicação

- "boost_mode" pode ser alterado para min, max, replace, avg ou soma.

Summary

- Compound queries combine leaf queries to create advanced queries that satisfy multiple search criteria.
- The `bool` query is the most popular compound query, consisting of four clauses: `must`, `must_not`, `should`, and `filter`.
- The queries in `must_not` and `filter` clauses do not contribute to the overall relevance score. On the other hand, queries in `must` and `should` clauses always improve the scoring.
- The `constant_score` query wraps a `filter` query and produces a constant score set by the user.
- The `boosting` query increases the score of a `positive` clause while suppressing the score on the queries that aren't a match (the `negative` clause).
- The `dis_max` query, used by the `multi_match` query, wraps queries and executes them individually.
- The `function_score` query sets a custom score based on a user-defined function, such as a field's value or weight or a random value.