Databáze, úvod do fulltextového vyhledávání

Fulltextové vyhledávání a ElasticSearch

Tomáš Urbanec

Katedra informatiky, Univerzita Palackého v Olomouci

ZS 2025/2026

Fulltextové vyhledávání

- umožňuje hledat v textu podle slov, frází nebo jejich částí.
- Klíčové techniky:
  - **Tokenizace**
    - rozdělení textu na jednotlivé tokeny (slova).
  - **Stemming**
    - ořez slova na jeho kořen (často už ne slovo)
  - **Lemmatizace**
    - převod slova na základní tvar.
    - v češtině náročné
  - **Invertovaný index**
    - struktura umožňující velmi rychlé vyhledávání.
    - udržuje informaci o dokumentech, ve kterých je dané slovo
- Použití:
  - dokumentové systémy,
  - analýza logů,
  - webové vyhledávače, e-shopy, doporučovací systémy, ...

Elastic search

- Open-source distribuovaný vyhledávací a analytický engine.
- Postaven na knihovně **Apache Lucene**.
  - výkonná open-source knihovna pro fulltextové vyhledávání a indexaci textu.
  - základ pro vyhledávací systémy (např. Elastic search, Solr)
  - invertované indexy, analyzéry, efektivní dotazovací struktury, ...
- Hlavní vlastnosti:
  - Velmi rychlé fulltextové vyhledávání.
  - Horizontální škálování (clustery, shardy, repliky).
  - REST API pro snadnou integraci.
  - Vhodné pro logy, monitoring a analýzu dat (ELK stack).
- Vhodný i pro rychlou analýzu velkých objemů dat.
- www.elastic.co/elasticsearch
- Pro instalaci vizte návod dr. Laštovičky.

Elastic search - základní použití

- Elasticsearch poskytuje jednoduché REST API (HTTP + JSON).
- Základní operace:
  - Vyhledávání:
    GET /index/_search
  - Získání dokumentu:
    GET /index/_doc/1
  - Vytvoření dokumentu:
    POST /index/_doc + JSON
  - Aktualizace dokumentu:
    PUT /index/_doc/1 nebo POST /index/_update/1
  - Mazání dokumentu:
    DELETE /index/_doc/1
- Vše postupně projdeme

Základní pojmy

- **Dokument**
  - textový řetězec, zapisujeme v uvozovkách
  - Např. " Televize Tesla "
- **Token**
  - neprázdný řetězec z malých písmen
  - Např. televize , tesla
- **Záznam**
  - dvojice $\langle d, D \rangle$, kde
  - kde d je identifikátor a D dokument

Tokenizace

- převod dokumentu na posloupnost tokenů $t_1, ..., t_n$
- dokument rozdělíme na bílých znacích a získaná slova převedeme na malá písmena
- značíme T, tj.

$$T(D) = (t_1, \ldots, t_n)$$

(Ukázka)

Index

- konečná množina záznamů s po dvou různými identifikátory
- Tj.

$$I = \{\langle d_1, D_1\rangle, \ldots, \langle d_n, D_n\rangle\}$$

  - kde $i \neq j \Rightarrow d_i \neq d_j$

Vytváření a přidávání do indexu

- PUT /index vytvoří nový prázdný index $\emptyset$
- Dokument v indexu lze vytvořit pomocí REST API:
  - POST /index/_doc – vytvoření dokumentu s automatickým ID.
  - PUT /index/_doc/ID – vytvoření nebo přepsání dokumentu s vlastním ID.
- Tělo požadavku je JSON reprezentující data dokumentu.
- Přepsání dokumentu v indexu modelujeme jako odstranění původního a přidání nového záznamu s daným ID
  - $Del(I, d)$ v doprovodných textech dr. Laštovičky

(Ukázky)

Mazání indexu a z indexu

- DELETE /index odstraní celý index (nelze vrátit zpět).
- DELETE /index/_doc/ID smaže dokument s daným ID.
- Pokud dokument neexistuje, Elasticsearch vrátí informaci not_found.
- Elasticsearch při mazání nevykonává fyzické odstranění ihned:
  - dokument je označen jako smazaný a fyzicky je odstraněn až při tzv. segment merge.
- Mazání dokumentu v modelu lze chápat jako operaci
  - $Del(I, d)$ v doprovodných textech dr. Laštovičky

(Ukázky)

Získávání záznamů

- GET /index/_doc/ID vrátí dokument s daným ID v daném indexu.
- Tj. pro $d \in Dom(index)$ dostaneme $\langle d, index(d) \rangle$
- Pokud dokument neexistuje, Elasticsearch vrátí found: false .
- Získaný dokument obsahuje metadata:
  - _id, _index, _version, _source
- Pro hromadné načtení více dokumentů lze použít:
  - POST /_mget – multi-get s výčtem ID.
- V modelu práce s indexem jde o operaci přístupu k prvku

(Ukázky)

Fulltextové vyhledávání

- GET /index/_search provádí dotaz nad dokumenty v indexu.
- Tělo požadavku je JSON s definicí dotazu:
  - jednoduchý match: hledání konkrétního slova nebo fráze,
  - bool dotaz: kombinace podmínek (must, should, must_not),
  - filtrování podle polí, rozsahů, dat apod.
- Výsledek obsahuje:
  - hits . total – počet nalezených dokumentů,
  - hits . hits – seznam odpovídajících dokumentů.
- V modelu práce s indexem jde o dotaz na data (splňující podmínky)

(Ukázka)

Základní podmínky

- **Podmínka** - je výraz, o kterém lze u každého dokumentu říct, jestli jej splňuje.
- Term t jako podmínka
  - Nechť $Set(T(D))$ je množina termů v dokumentu D
  - dokument D splňuje t, pokud $t \in Set(T(D))$
- Pokud v podmínce uvedeme více termů, vyhodocují se v disjunkci
  - Tj. dokument podmínku splňuje, pokud obsahuje alespoň jeden z nich.
- Pokud v podmínce uvedeme více termů a uvedeme operator: AND, vyhodnocují se v konjunkci.
  - Tj. dokument podmínku splňuje, pokud obsahuje všechny.

(Příklady)

Složené podmínky

- Dotaz může kombinovat více poddotazů s logickými vztahy.
- Hlavní podmínky:
  - must – všechny tyto poddotazy **musí** být splněny (AND).
  - should – alespoň jeden z těchto poddotazů **by měl** být splněn (OR), zvyšuje skóre.
  - must_not – tyto poddotazy **nesmí** být splněny (NOT).

(Příklady)

Podmínky - zápis

```
{
  ...
  "match": { "text": "..." },
  "match": { "text": { "query": "...", "operator": "AND"} },
  "bool": {
    "must": [ { "match": { ... } } ],
    "should": [ { "match": { ... } } ],
    "must_not": [ { "match": { ... } } ]
  }
}
```

Restrikce indexu

- Restrikce (filtrování) umožňuje omezit výsledky dotazu na konkrétní podmnožinu dokumentů, dle zadané podmínky.
- Je-li theta podmínka a I index, pak

$$\sigma_\theta(I) = \{\langle d, D\rangle \in I \mid D \text{ splňuje } \theta\}$$

```
GET /index/_search
{
  "query": {
    podminka
  }
}
```

(Ukázky)

Vyhodnocování kvality odpovědí

- Cílem je měřit, jak dobře systém odpovídá na dotazy.
- Zohledňuje se:
  - **relevance** dokumentů k dotazu,
  - úplnost a přesnost výsledků,
  - pořadí výsledků v seznamu.
- Nastíníme jen základní pojmy.
- Všechny pojmy jsou řádně definovány v doprovodných textech, projděte si je.

Základní statistiky

- Počet výskytů termů v dokumentu:

$$f(t, D) = \text{počet výskytů termu t v D}$$

- Délka dokumentu

$$dl(D) = \text{počet termů v dokumentu}$$

- Průměrná délka dokumentu v indexu I

$$avgdl(I) = \frac{\sum dl(D_i)}{N(I)}$$

- Počet záznamů v I: $N(I) = |I|$
- Počet dokumentů v I obsahujících term t

$$n(I, t) = |\{\langle d, D\rangle \in I \mid t \in \mathrm{Set}(T(D))\}|$$

IDF — inverse document frequency

$$IDF(l,t) = \ln\left(\frac{1 + N(l) - n(l,t) + 0.5}{n(l,t) + 0.5}\right)$$

Vyjadřuje vzácnost termu.

TF — term frequency

$$tf(I,t,D) = \frac{f(t,D)}{f(t,D) + k_1(1 - b + b\frac{dl(D)}{avgdl(I)})}$$

- term saturation parameter $k_1 = 1.2$
- length normalization parameter $b = 0.75$

Skóre dokumentu

- Pokud je podmínka term:

$$score(I, D, t) = boost \cdot IDF(I, t) \cdot tf(I, t, D)$$

  - parametr $boost = k_1 + 1 = 2.2$.
- Pokud je podmínka disjunkce:

$$score(I, D, \theta_1 \vee \cdots \vee \theta_n) = \sum_{i=1}^{n} score(I, D, \theta_i)$$

- Pokud je podmínka konjunkce

$$\text{score}(I, D, \theta_1 \wedge \cdots \wedge \theta_n) = \sum_{i=1}^{n} \text{score}(I, D, \theta_i)$$

  - V čem je rozdíl?
- Pro hodnocení složitějších podmínek vizte studijní materiály.

Vysvětlení výpočtu skóre systémem

- Přidáme-li k dotazu klíček explain s hodnotou true, pak
- Získáme položku explanation, která popíše
  - IDF
  - TF
  - výsledné score
  - podrobné mezivýpočty

Shrnutí

- Elastic search slouží k fulltextovému vyhledávání v dokumentech
- Index: množina záznamů $\langle d, D \rangle$
- Podmínka se vyhodnocuje nad dokumentem
- Restrikce na podmínku vrací podindex splňující podmínku
- Elasticsearch provádí:
  - GET /_doc
  - PUT /_doc
  - DELETE /_doc
  - GET /_search
- Kvalita splnění podmínky (score) vychází z TF–IDF modelu
- Dotazy vrací dokumenty v sestupném pořadí podle score

Dotazy, zpětná vazba a diskuze