Elasticsearch

Vad är Elasticsearch?

En sökmotor och datastore

Baserat på **Lucene**

Vad är Lucene?

Ett Java-bibliotek för sökning

Tar dokument och indexerar dem i ett format som är effektivt och välfungerande för textsökning.

(Elasticsearch index != Lucene index då Elasticsearch delar upp datan i **shards** som kan distribueras över noder i ett kluster.)

Elasticsearch utöver Lucene

- Rest API
- Distributed
- QueryDSL i JSON
- Monitorering
- ... allt utom sökningen?

Solr?

Solr är också baserat på Lucene.

Och kan göra samma saker.

Är möjligtvis lättare att anpassa till specialfall.

Sökning

- Alla dokument som innehåller något av termerna “elasticsearch”, “lucene” eller “solr”
- “hästar” träffar även på “häst”
- Sök på felstavade ord, fuzzy matching
- Resultaten är ordnade efter **relevans** (score i Elasticsearch)

Kräver att texter i dokumenten analyseras och att metadata om termfrekvens och dokumentfrekvens skapas.

Det perfekta scenariot?

- Datan passar bra i JSON-format
- Resultatet ska sorteras efter relevans
- Dokumenten innehåller text som behöver analyseras
- Användaren söker i fritextfält
- Datan ändras inte
- Datan kan återskapas från en annan källa
- (Behov av hög tillgänglighet)

Det sämsta scenariot?

- Det finns relationer mellan dokument
- Transaktioner behövs
- Det finns constraints som behöver upprätthållas
- Dokumenten sorteras alltid på något fält, t.ex. datum
- Textanalys behövs ej
- Datan är enkel att “platta ut”
- Datan ändras och får ej förloras
- Textsökningen är “Korplik”

Mittemellanscenariot

Kombinera Elasticsearch med en annan databas.

Använd Elasticsearch för sökning.

Använd den andra databasen för det som Elasticsearch är dålig på.

> Det går att indexera och söka i Elasticsearch, men stänga av datalagring för att spara diskutrymme.

Alternativ: Relationsdatabaser

JSON-kolumner med index på de fält man vill söka i

Fulltext index på textinnehåll

Många relationsdatabaser kan ge tillbaka resultatet i relevansordning samt göra highlighting

> Om man ändå vill använda t.ex. Postgres så kan man undersöka hur bra ovanstående funktioner fungerar innan man blandar in Elasticsearch.

Andra alternativ

- En annan NoSQL-databas
- Lucene, om man bara behöver sökningen

Annat?

Elasticsearch basics

Nod

Kluster

Index

Shards (och replicas)

Mapping

Elasticsearch basics

Indexera JSON-dokument.

Gör sökningar och få tillbaka samma JSON-dokument (om dom träffar).

Inverted index

Mappa termer till dokument

Termfrekvens för att kunna skapa **score** för sortering

Positioner och offsets för att kunna göra **highlighting**.

Inverted index

1: Winter is coming.
2: Ours is the fury.
3: The choice is yours.

term	freq	documents
choice	1	3
coming	1	1
fury	1	2
is	3	1, 2, 3
ours	1	2
the	2	2, 3
winter	1	1
yours	1	3

Dictionary

Postings

Mappning

Dokumentet som skickar in sparas som **_source**

Indexera inte fält som inte ska sökas i

Mappa varje fält i dokumentet till en datatyp

> **Typer:** text (analyseras), keyword (analyseras inte), olika typer av nummer, bool, objekt och geografisk data.

Mappning

Det dokument man skickar in är det man får tillbaka.

Med undantaget att det går att exkludera ett fält ifrån **_source**.
Användbart för fält som enbart är till för sökning.

```
{
  "code": 3300,
  "name": "Abbekås",
  "info": "Ett trevligt ställe.",
  "municipality": {
    "code": 1264,
    "name": "Skurups kommun"
  }
}
```

```
"mappings": {
  "_doc": {
    "properties": {
      "code": { "type": "integer"  },
      "name": { "type": "keyword"  },
      "info": { "type":   "text" },
      "municipality": {
        "properties": {
          "code": { "type": "integer },
          "name": { "type": "text" }
        }}}}}
```

Textanalys i Elasticsearch

- Character filters
- Tokenization
- Token filters

Token filters

Får output från tokenizern och bearbetar varje token

- Skiftläge
- Böjningsmönster
- Synonymer
- Stoppord
- Sammansatta ord
- Expansion av mening (äpple -> äpple, frukt)
- Skiljetecken (standard tokenizer tar bort skiljetecken)

fields

Ett fält kan ha flera subfält och analyseras på flera olika sätt.

```
{
  “aField”: “With DaTa”
}
```

```
aField -> [“with”, “data”],
aField.raw -> “With DaTa”
```

Listor

Array har ingen dedikerad datatyp i Elasticsearch.

Alla fält kan ta emot en lista av värden, bara typen är rätt.

(Men det finns en speciell datatyp som heter **nested**, som kan behövas för listor av objekt...)

Några viktiga querytyper

- **term** - ingen analys, söker efter exakta träffar
- **match** - analyserar söksträngen med samma analyzer som fältet man söker i
- **bool** - kombinerar queries med **must** eller **should**
- **match_phrase** - analyserar söksträngen, kräver att termerna är i ordning och ett visst avstånd ifrån varandra
- **multi_match** - som match, men söker i flera fält

Sökning

Fokus bör ligga på att träffarna är ordnade från mest relevant till minst relevant, inte vilka dokument som träffar.

Filtrering

Använd **filter** parametern i **bool**, eller **constant_score**.

Ingen scoring sker, oönskade dokument försvinner från träffmängden.

Kombinera med fritextsökning, som **multi_match**.

Relevans igen

- Elasticsearch beräknar score med hjälp av **BM25** funktionen

term frequency, inverse document frequency, field-length normalization, nonlinear term-frequency saturation

- **explain: true**
- Boosts
- Förväntningen är nog oftast att man vill ha dokument där flest termer träffade, men det sker inte alltid.

Relevans igen

Sökning på fler termer i flera fält

```
Boolean should = [
        text=”elastic”,
        text=”kibana”,
        titel=”elastic”,
        titel=”kibana”
]
```

score blir summan av de fyra termernas score

Relevans igen

Sökning på fler termer i flera fält

```
Multi-match
   fields= [“text”, “titel”]
   query=”elastic kibana”
]
```

score beräknas på olika sätt beroende på vilken typ av **multi_match** man gör: t.ex. **best_fields** slänger bort **score** från alla fält utom det som har högst värde.

Sökning

Inga felmeddelanden om man söker i fält som inte finns

multi_match, **query_string** m.fl. hoppar över fält som det inte går att söka i och det enda sättet man kan veta är m.h.a. **explain: true**

Tänk igenom vilka fält du vill söka i och hur de analyseras och använd explain för att kolla att det sker.

https://github.com/majsan/esworkshop