Semi-Hack-a-thon OV France, Paris, 5 avril 2019

Elasticsearch

Retour d’expérience dans le cadre de SsODNet

Jonathan Normand

SsODNet

- Système d’information dédié aux objets des systèmes solaire et extrasolaire : planètes, satellites, astéroïdes, comètes, sondes spatiales et débris, exoplanètes
- 846213 objets pour ~3.5M de désignations
- Plusieurs modules dont :
  - Quaero : API REST de recherche et d’identification des corps

Quaero – Objectifs

- Rechercher un objet en fournissant dans un champ unique des informations pouvant correspondre à plusieurs propriétés : *name*, *aliases*, *class*, *type*...
- Contrôler les résultats en favorisant certaines propriétés
- Filtrer les résultats

> Fonctionnalités avancées de recherche en plein texte

Quaero – Ressource

- Identification
  - Identifiant unique
  - Nom
  - Désignations alternatives
- Autres propriétés
  - Relation parent-enfant
  - Système dynamique
  - Type
  - Classe

```
{
  name: "Io",
  id: "Io_(Asteroid)",
  aliases: ["85","1899 LA"...],
  ephemeris: true,
  class: ["MB","Middle"],
  system: "Sun",
  parent: "Sun",
  type: "Asteroid",
}
```

Quaero – Cas d’utilisation

Mars, 1, 1P, I99L00A, J-1, P/Halley, C/1987 A1, P/Shoemaker-Levy 9 (D), SCOUT B R/B, 2013 AB, S/2002 (2001 QC298) 1, CoRoT-18 b, SL-1 R/B

- Exemples de recherche :
  - P/Shoemaker-Levy via « P/Shoemaker-Levy », « Shoemaker » ou « levy »
  - S/2002 (2001 QC298) 1 via « QC298 »
  - « corot-1 » doit uniquement retourner CoRot-1 (pas de CoRot*)
  - par numéro
  - filtrage par type d’objet
- Recherche avancée par combinaison de propriétés
- Mécanisme d’auto-complétion à partir des premiers caractères d'une désignation ou d'un de ses tokens

Elasticsearch

- Pourquoi
  - Langage de requête simple – Lucene
  - Données facilement accessibles (API REST) et lisibles (conservation de la structure des documents)
  - Outils simples et performants
    - *elasticsearch-head* pour inspecter les données
    - API *_analyze* pour analyser comment sont indexées les propriétés des documents
    - API *_validate* pour comprendre comment est effectuée une recherche
- Infrastructure
  - Version 2.4
  - Cluster contient un nœud. Pas d'utilisation des aspects « distribué » ou « haute disponibilité »
  - Un index *ssodnet*, un type de document *sso*
  - Fonctionnalité « snapshot and restore »

Index – mapping

- Comment les propriétés d'un document sont indexées
- Nécessaire pour éviter les risques
  - *_all* – propriétés par défaut
  - dynamic mapping
- Configuration de *ssodnet*
  - *_all* désactivé – les propriétés par défaut sont définies au moment de la recherche
  - Pour toutes les propriétés :
    - *type*
    - *index* : no (propriété non exposée à la recherche on ne l'indexe pas) ou not_analyzed
    - *analyzer* et *search_analyzer* – Attention à la cohérence
  - Une même propriété peut être indexée de différentes façons mais sous des noms différents

Index – Analyseur

- Transformation d'une chaîne de caractères en tokens. Les tokens permettent de construire un index inversé.
- Fonctionnement d'un analyseur :
  - Filtrage des caractères (char_filter)
  - Découpage en tokens (tokenizer)
  - Traitement et/ou transformation des tokens (token filters)
- Configuration de *ssodnet* – 4 analyseurs pour les besoins suivants :
  - char_filter
    - Supprimer les caractères non significatifs (pattern_replace)
  - Tokenizers
    - Conserver la chaîne entière (keyword)
    - Découpage en tokens (whitespace)
  - Token filters
    - Rendre la recherche insensible à la casse et aux accents (lowercase, asciifolding)
    - Conserver les tokens alpha-numérique (word_delimiter)
    - Découper les tokens en suite de caractères (edgeNGram)

Recherche standard

- Tous les tokens d'une requête doivent apparaître dans les propriétés donnant les désignations

```
{
  multi_match: {
    query: "1950 TT2",
    type: "best_fields",
    fields: ["name.raw^2","name",
      "aliases.raw^2", "aliases"],
    operator: "and"
  }
}
```

- *multi_match* : recherche dans plusieurs propriétés et la requête est analysée avant son exécution
- Favoriser la correspondance avec la chaîne entière dans le calcul du score
- Propriété multivaluée (ex : 2000 JW8)

Recherche standard avec filtre

- Tous les tokens d'une requête doivent apparaître dans les propriétés donnant les désignations et seuls les documents correspondant au type demandé sont retournés

```
{
  bool: {
    must: [{
      multi_match: {
        query: "io",
        fields: ["name.raw^2","name",
          "aliases.raw^2","aliases"],
        operator: "and"
    }}],
    filter: [
      {term: {"type": "Asteroid"}}
    ]
  }
}
```

- *filter* : on se place dans un context filter (oui / non) car on veut uniquement obtenir des résultats correspondant au type demandé
  - term : pas d'analyse donc attention à la casse, performance
- Utilisation *match* : retourne aussi des documents du mauvais type mais avec un score plus faible

Recherche avancée

```
{
  query_string: {
    query: "1950 tt2",
    fields: ["name.raw", "name", "aliases.raw", "aliases"]
  }
}
```

- *query_string*
  - contraintes sous forme de paires clef:valeur
  - wildcards, expressions régulières et similarités (fuzziness)
- *fields* : précise les propriétés utilisées en l'absence de clef
- OR opérateur par défaut

Auto-complétion

- Recherche à partir du début de la chaîne recherchée ou d’un de ses tokens
- Analyseurs d’indexation et de recherche différents :
  - indexation : chaîne entière, découpage en tokens et lowercase, asciifolding, edgeNGram
  - recherche : lowercase et asciifolding

Elasticsearch – Outils

- API _*analyze* permet de tester :
  - analyzer
  - tokenizer avec ou sans token filters
  - champs
- API _*validate* permet de :
  - Valider la syntaxe d'une recherche
  - Comprendre comment la recherche est effectuée

```
explanations: [{
  index: "ssodnet", valid: true,
  explanation: "+(
    (+aliases:1950 +aliases:tt2) | aliases.raw:1950 tt2^2.0
    | (+name:1950 +name:tt2) | name.raw:1950 tt2^2.0)
    #ConstantScore(_type:sso)"
}]
```

Ressources en ligne

- SsODNet

  `http://vo.imcce.fr/webservices/ssodnet/`

- Documentation de l'API Quaero

  `https://doc.ssodnet.imcce.fr/quaero.html`

- Exemples et cas d'utilisation

  `http://vo.imcce.fr/webservices/ssodnet/?quaero`