ElasticSearch

Elasticsearch

https://www.elastic.co/fr/

- Moteur de recherche qui indexe des documents.
- Propose une API WEB qui analyse un texte pour son indexation + pour la recherche
- Elasticsearch est utilisé par:
  Github, Docker, OUI.sncf, Orange ...

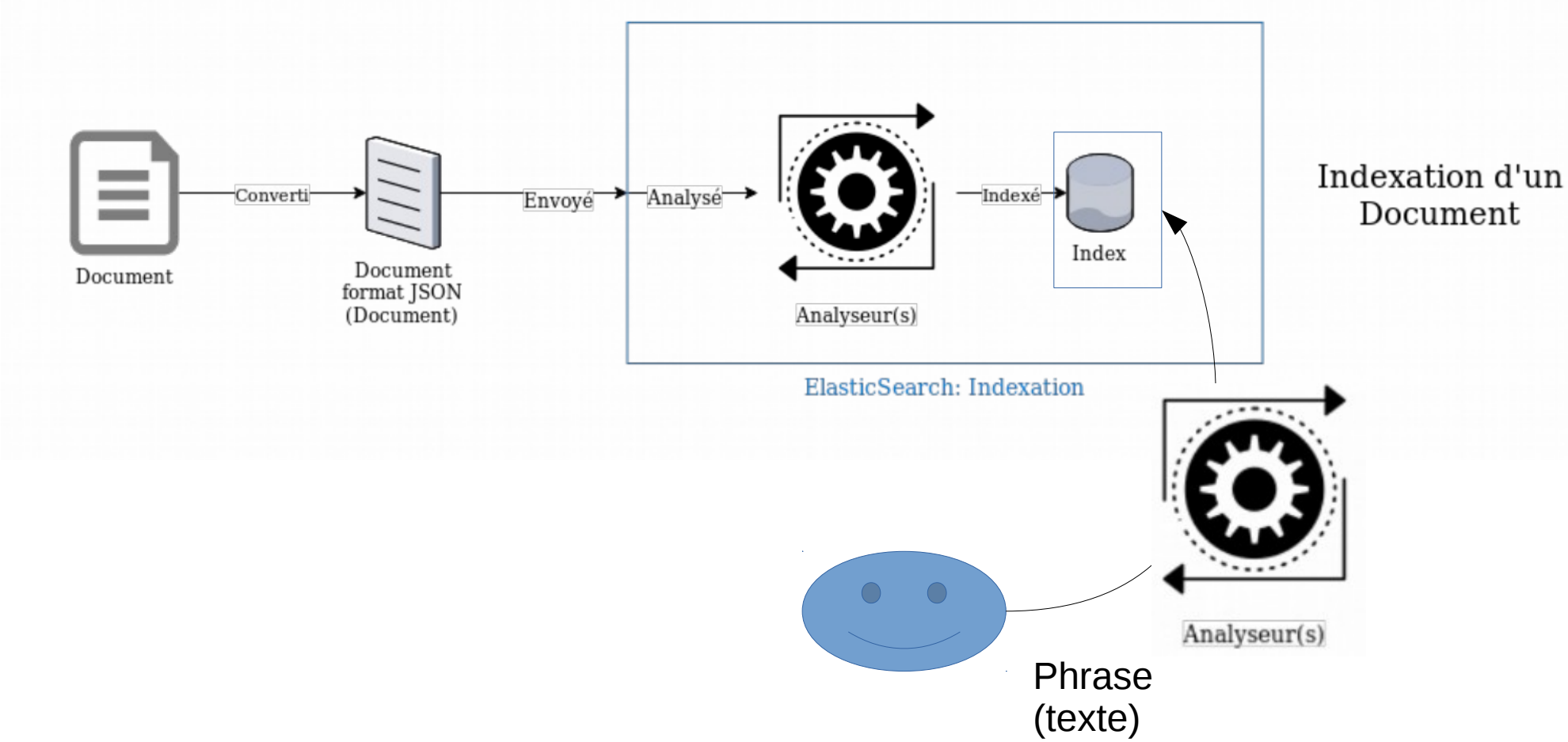
Technique

- Base de données **NOSQL**
- Basé sur le moteur de recherche **Apache Lucene**
- Demon Java , open source sous licence Apache
- Peut contenir jusqu'à 2G de documents
- Dispose d'un architecture qui peut être parallèlisée
- Architecture plutôt lourde : elasticSearch utilise beaucoup de mémoire (mémoire conventionnelle + partagée)

Indexation d'un document

L'indexation d'un document dans une base ElasticSearch

- Format JSON en entrée



Mise en forme du document JSON

Fichier ReadMe de VizieR

```
B/assocdata     Associated data in VizieR                    (G.Landais, 2016)
================================================================================
Spectra, Time-series and Images in VizieR gathered into a ObsCore Table
    G.Landais, Laurent Michel, Pierre Ocvirk et al.
   <EPJWC 186, 02002>
   =2018EPJWC.186020020
================================================================================
ADC_Keywords: Observatory log
Keywords: spectroscopy

Abstract:
    The ObsCore VizieR table gather FITS images, spectra and time-series into a
    single table. The contents are the VizieR associated data, publilished in
    original article with the tables.
    The metadata comes from the ObsCore Data Model. ObsCore (Tody et al. 2011)
    is a standard of the Virtual Observatory used to map images,spectra or
    time-series resources with standardized metadata.


File Summary:
--------------------------------------------------------------------------------
 FileName    Lrecl    Records    Explanations
--------------------------------------------------------------------------------
ReadMe          80          .   this file
obscore.dat    1110       1000  VizieR Spectra, images gathered in a table


See also:
    http://cdarc.u-strasbg.fr/assocdata/ : the VizieR associated data
    http://saada.unistra.fr : Saada Database generator software
    B/cfht/obscore : ObsCore CFHT logs observation (CADC)
    B/hst/obscore : ObsCore HST Logs observation (CADC)
    B/gemini/obscore : Gemini observation logs (CADC)
    B/jcmt/obscore :  Jame Clerk Maxwell Telescope Science Archive (CADC)

Byte-by-byte Description of file: obscore.dat
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
                                                a product
                                                ibration level
                                                ervation ID
                                                e of the data collection
                                                aset identifier given by the
                                                lisher
                                                ronomical object observed
 277- 290   F14.10 deg RAdeg              Central right ascension (J2000) (s_ra)
 292- 306   F15.11 deg DEdeg              Central declination (J2000) (s_dec)
 308- 324   E17.12 ---   s_fov            ? Diameter of the covered region
 326- 581   A256   ---   s_region         Region covered as specified in
                                           STC or ADQL
 583- 602   F20.15 ---   s_resolution     ? Spatial resolution of data as FWHM
 604- 618   F15.7  d     t_min            ? Start time in MJD
 620- 634   F15.7  d     t_max            ? Stop time in MJD
 636- 649   F14.8  ---   t_exptime        ? Total exposure time
 651- 652   F2.0   ---   t_resolution     ? Temporal resolution FWHM
 654- 670   E17.12 ---   em_min           ? Start in spectral coordinates
 672- 688   E17.12 ---   em_max           ? Stop in spectral coordinates
 690- 705   A16    ---   em_band          Spectral band (1)
 707- 722   A16    ---   o_ucd            UCD of observable
 724- 730   A7     ---   pol_states       List of polarization states
 732- 763   A32    ---   facility_name    Name of the facility used for this
                                           observation
 765- 796   A32    ---   instrument_name  Name of the instrument used for this
                                           observation
 798-1053   A256   ---   access_url       URL used to access dataset
1055-1074   A20    ---   access_format    File content format
1076-1085   I10    ---   access_estsize   Estimated size of dataset in
                                           kilo bytes
1087-1106   A20    ---   oidsaada         internal Saada OID (non persistent)
1108-1108   I1     ---   has_wcs          WCS detected (1) (2)
1110-1110   I1     ---   extension        Extension (1)
--------------------------------------------------------------------------------

Note (1):
     columns which is not a IVOA standard

Note (2):
     WCS detection flag
     1=contains positional WCS
     2=contains spectral WCS

History:
    Prepared via OCR at CDS.

Acknowledgements:
    the VizieR documentalists M.Brouty, S.Guehenneux, E.Perret, T.Pouvreau, P.Vannier

References:
    ObsCore Data Model. ObsCore (Tody et al. 2011)

================================================================================
(End)                                   Gilles Landais [CDS]    02-fevr.-2017
```

Fichier JSON pour ElasticSearch

```
{
    "year": "2016",
    "id": "B/assocdata",
    "title_cds": "Associated data in VizieR",
    "author": "G.Landais",
    "bibcode": ["2018EPJWC.186020020", "2019Ycat....."],
    "keyword_cds": "Observatory log",
    "keyword_pub": "spectroscopy",
    "abstract": "The ObsCore VizieR table gather FITS images
spectra and time-series into a single table. The contents are the
VizieR associated data publilished in original article with the
tables. The metadata comes from the ObsCore Data Model. ObsCore
(Tody et al. 2011) is a standard of the Virtual Observatory used
to map imagesspectra or time-series resources with standardized
metadata.",
    "curator": "Gilles Landais [CDS]",
    "last_update": "2020-05-01",
    "tables": [{
        "name": "obscore.dat",
        "desc": "VizieR Spectra, images gathered in a table "
      },
,       {"name": "obscore.dat",
,       "desc": "VizieR Spectra, images gathered in a table "
,     }],
,   ......
,}
```

Structure ElasticSearch

Structure ElasticSearch – comparaison avec le vocabulaire SGBD

Entité ElasticSearch	Description	Equivalent SGBD
document	document	Record/ligne
field	Information décrivant une propriété d'un document	colonne
shard	Unites incluant un ensemble de documents	Schémas ?
index	Collection de documents avec des propriétés similaire	Database / table

L'index ElasticSearch est dynamique

Ie : la structure des json injectés dans la base (avec ces mots clés) ne sont pas figés

Ainsi, pour chaque document injectés (en json)

- Aucun field n'est obligatoire
- Un nouveau field est possible (nouvelle clé json)
- Attention à respecter le type de chaque field (ex : field date)

Type de donnees

Les fields peuvent prendre plusieurs types

- boolean
- Integer
- double
- Keyword
- Text
- Date
- Ip
- ....

Analyser

L'analyse de texte ElasticSearch

- Les "phrases" sont analysés par ElasticSearch à l'aide de « l'Analyser"
- l'analyser ElasticSearch :

  1) supprime/remplace les caractères "indésirable" :
     ex : remplacer & par et, suppression de HTML ….

  2) découpe une phrase selon des stopwords : le **tokenizer**
     le tokenizer est unique

  3) analyse les mots pour y extraire des radicaux: le **token-filter**
     ex : en anglais : construction => construct
     en francais : construction => construction

l'analyser = tokenizer + token-filter

Exemple :

« les travaux de construction de ALMA ont commencé en 1998 »

french analyser → traval, construction, alma, commenc, 1998

Analyser

Elasticsearch propose un certain nombre d'anlyser

- Standard analyser : enlève la ponctuation, met en lowercase
- Simple analyser : découpe un texte des qu'un caractère n'est pas une lettre et met en lowercase
- Whitespace analyser : idem – découpe avec les blancs
- Keyword analyser : prend un texte sans rien modifier dans un unique terme
- Language anlayser : anglais, francais..

Exemple d'analyse

Voir kibana

Analyser

Personnalisation des analysers

- Ajout de stopwords
- Ajout de synonymes
- Composer son analyser
  - Sélectionner son stopwords (tokenizer)
  - Ajouter autant de filtre que l'on veut : ex stemmer_english+lowercase
- Créer ses propres filtres

Injestion des documents

l'API ElasticSearch

→ API HTTP REST base sur JSON

- La liste des index : GET /_cat/indices
  curl 127.0.0.1:9200/_cat/indices
- Creer un nouvel index : PUT /new_index
  ex: curl -X PUT localhost:9200/new_index
- La liste des fields d'un index: GET /new_index
  → renvoie du JSON
- Supprimer un index: DELETE /new_index
- Ajouter un document : POST /new_index/
  curl -XPOST "http://localhost:9200/new_index/_doc/" -H 'Content-Type: application/json' -d'{"user" : "landais", "message" : "coucou"}'
- Mais encore : on peut update un document, supprimer, faire un bulk (copie massive)....

Mapping

Faire un mapping ElasticSearch

On peut lors de la creation ajouter un mapping qui détermine pour chaque fields l'analyser que l'on va utiliser.

Fichier json utilisé lors de la création de l'index

Requetes ElasticSearch

… Encore une API HTTP (JSON)

ElasticSearch dispose de plusieurs mode interrogation:

- Recherche dans tous les fields
- Recherche dans un field ou une liste de field déterminée
- Syntaxe
  - avec wildcard: ex: telesco*
  - exacte : “telescopic”
  - Fuzzy search : “telesco~”
- Pour une phrase, on peut specifier la logique AND, OR, NOT
- Faire des suggester: “schmi” → repond “schmitt”, “schmidt”, “schmit” …

Caclul de scores (et tri)

Pour chaque requette ElasticSearch calcul un score pour chaque document en fontion de sa pertinence:

- nombre de fois que le mot est trouvé
- mais il prend aussi en compte la rareté de certain mots !

Exemple de requete

- Exemple de requetes:

Voir aussi la recherche VizieR textuelle: http://cdsarc.unistra.fr/viz-bin/cat

```
GET /readme/_search
{
  "query": {
    "query_string": {
      "query":"author:schmit~ AND year:>2000"
    }
  }
}
```

Recherche des documents pour un auteur et à partir d'une date

→ utilise seulement les fields author + year

```
GET /readme/_search
{
  "query": {
    "query_string": {
      "query":"radial velocity",
      "default_operator":"AND",
      "fields": ["title_cds","abstract"]
    }
  },
    "_source":["first_author","author","title_cds", "year"]
}
```

Recherche d'une phrase dans toute la base

Plus loin?

ElasticSearch comme architecture bigdata

- Une base peut être découpée en shards
- Les shards peuvent être dupliqués sur plusieurs noeuds
- Les shards peuvent être répartis sur plusieurs noeuds (parallelisation)