



Elasticsearch : Retour d'expérience

Mise en oeuvre dans le cadre du projet SSHADE

Philippe Bollard (CNRS/IPAG/OSUG)

`philippe.bollard@univ-grenoble-alpes.fr`

05/04/2019 - Hackaton ASOV 2019 - Paris

1. Contexte
2. Problématique
3. Indexer efficacement
4. Rechercher efficacement
5. Mise en oeuvre
6. Debug
7. Conclusion

Contexte

<https://www.sshade.eu/>

Projet

- Entrepôt de données spectrales issues d'analyses de solides
- Données et toutes les métadonnées du contexte expérimental
- Stockage, indexation, diffusion (VO), publication (DOI)
- SNO labellisé par le CNRS/INSU

Équipe active

- Bernard Schmitt (porteur du projet scientifique)
- Philippe Bollard, Damien Albert (développement)
- L. Bonal, O. Poch (contributeurs scientifiques)



20 bases actives

- France (10)
- Pologne (2)
- Italie (2)
- Suisse (1)
- Allemagne (1)
- Espagne (1)
- Hongrie (1)
- UK/Inde (1)



Et d'autres à venir !

 SQLAlchemy

 PostgreSQL

 Pyramid™

 python™


elasticsearch

 **jQuery**
write less, do more.

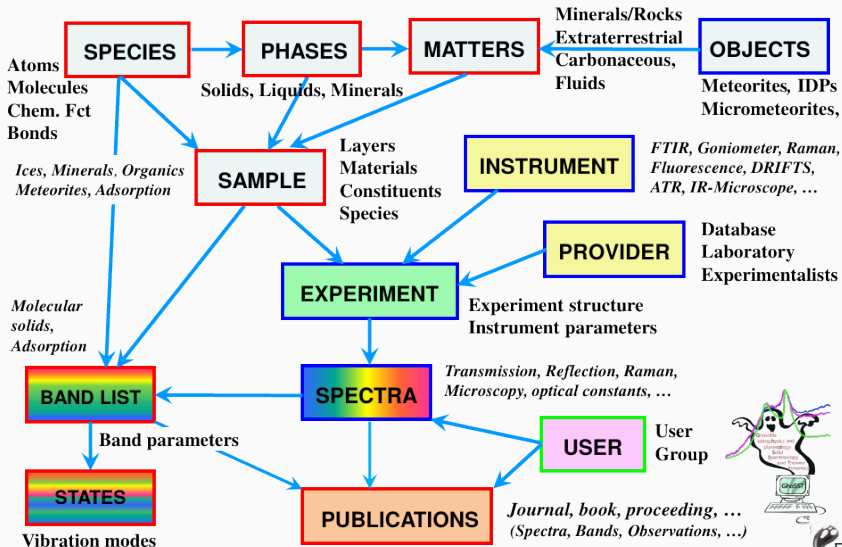
 **B** Bootstrap

 plotly.js

et GAVO DaCHS pour la partie VO / EPN-TAP

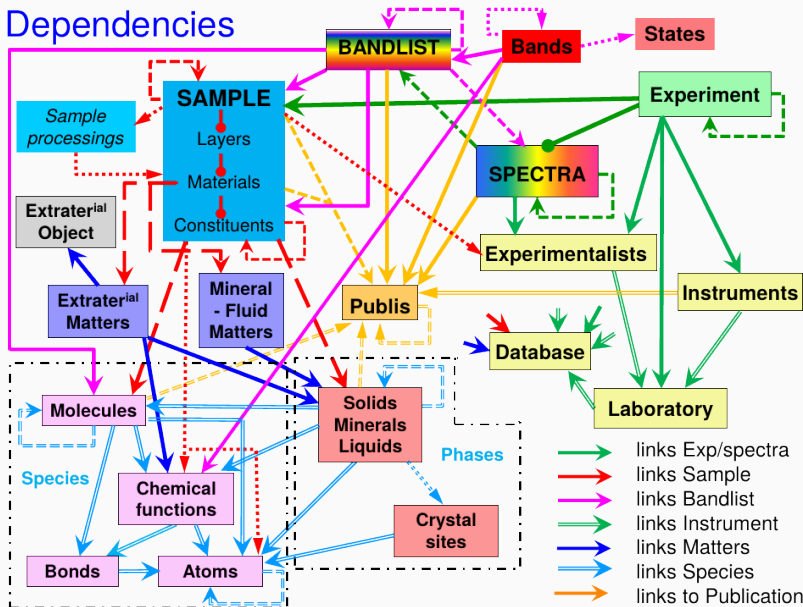
Problématique

SSDM: Solid Spectroscopy Data Model

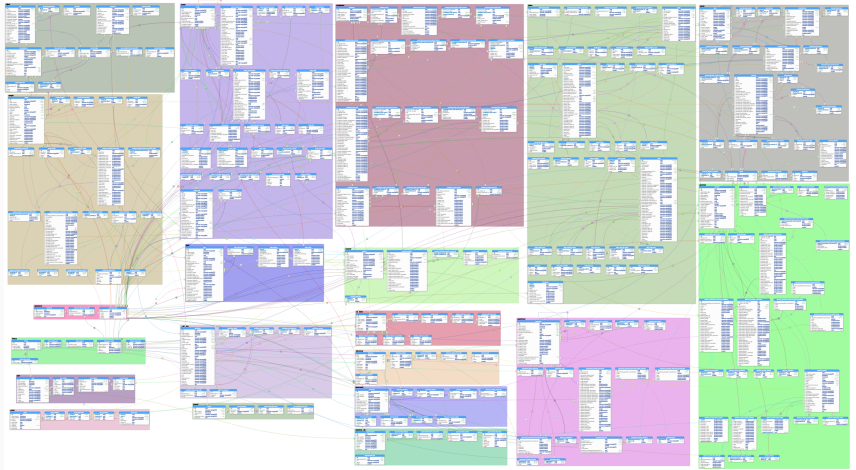


SSDM: Solid Spectroscopy Data Model

Dependencies



Implémentation PostgreSQL



≈ 250 tables réparties sur 20 schémas

Problématique

Moteur de recherche

Problème : Recherches ciblant de nombreuses tables

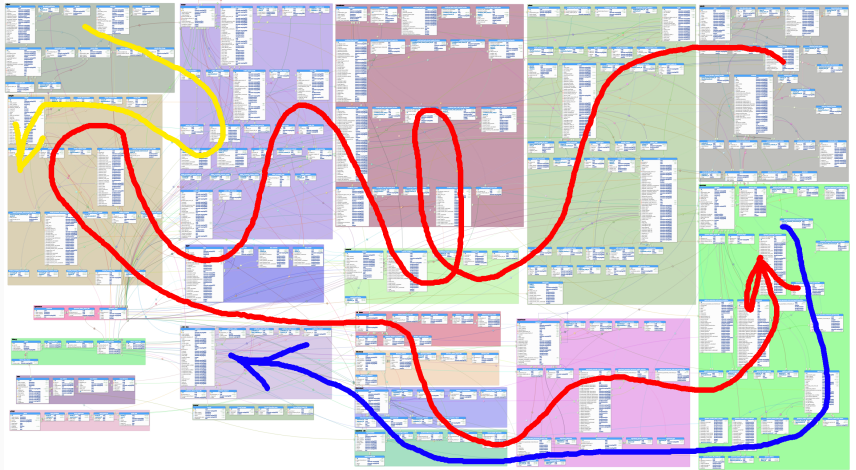
Exemples de recherche

- Spectres de météorites contenant du fer analysées à l'IPAG
- Publications qui utilisent les spectres de Bernard Schmitt
- Échantillons provenant de la météorite Allende

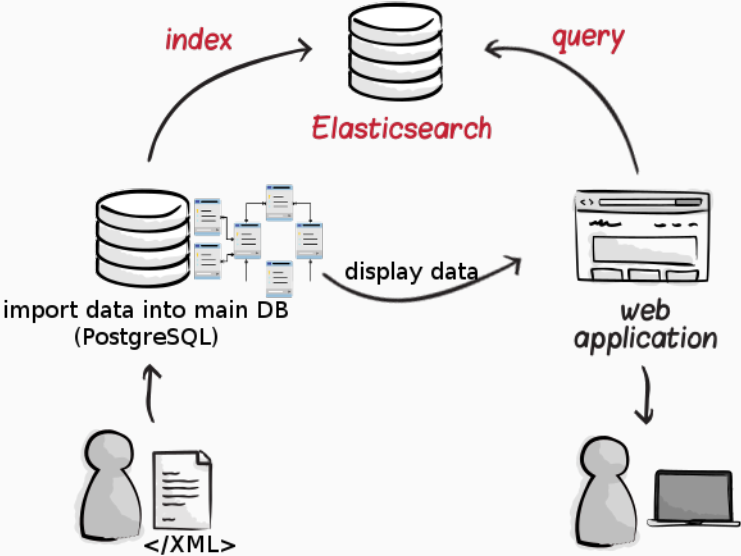
Complexité du modèle de données relationnel

- SSDM \approx 850 pages A4
- Nombreuses jointures pour assembler les résultats
- Chemins de recherche non déterministes
- Récursivité de certaines parties du modèle

Problème : Recherches ciblant de nombreuses tables



Solution : Utiliser Elasticsearch comme moteur de recherche



Problématique

Formulaire de recherche

Pour l'utilisateur

- Nombreux champs = ouhlala, trop compliqué pour moi
- Un seul champ = facile à coder, même Google sait le faire !

Pour le développeur (SQL)

- Nombreux champs = c'est faisable avec plusieurs JOIN mais...
- Un seul champ = quoi, tu veux vraiment ça... ? Pour hier ?

Solution : Combiner les deux approches



Spectrum ▾

Write your keywords here ...



Help ▾ Log In / Register

Spectra search

Reset

Write your keywords here ...

Search

Filters

Reset

By experiment

Owner

Database acronym

in ▾

Nothing selected ▾

Laboratory acronym

contains all words ▾

Experiment

Type

in ▾

Nothing selected ▾

Version

= ▾

Spectrum

Type

in ▾

Nothing selected ▾

Title

contains all words ▾

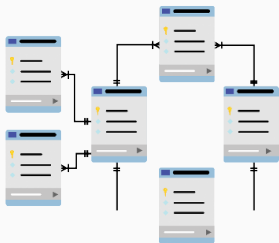
Date recorded

= ▾

YYYY-MM-DD

Indexer efficaciment

Comment traduire un schéma relationnel en structure JSON ?



Relational model



```
1 [
2   {
3     "name": "Top Level",
4     "parent": "null",
5     "children": [
6       {
7         "name": "Level 2: A",
8         "parent": "Top Level",
9         "children": [
10        {
11          "name": "Son of A",
12          "parent": "Level 2: A"
13        },
14        {
15          "name": "Daughter of A",
16          "parent": "Level 2: A"
17        }
18      ]
19    },
20   {
21     "name": "Level 2: B",
22     "parent": "Top Level"
23   }
24 ]
25 }
26 ]
```

JSON document

Indice

- équivalent d'une base de données SQL
- contient les documents JSON indexés

Mapping

- équivalent d'une structure de table SQL (colonnes, types)
- structure un indice avec une arborescence de propriétés typées

Type (de document)

- équivalent d'une table SQL

Attention, un indice ne peut pas avoir plusieurs mappings

⇒ créer un indice (et donc mapping) par type de donnée

Indexer efficacement

Définir le mapping d'un indice

Résoudre tous les chemins de recherche pour une donnée

Sélectionner les propriétés pertinentes...

... de l'objet à indexer

- UID
- title, name, formula, ...
- type, family, origin, ...

... des objets parents ou affiliés

- spectrum: experiment, sample, ...
- spectrum: parent spectra, ...

... des objets enfants

- sample: layer
- layer: materials, matters, ...

Assigner un type de donnée à chaque propriété

- text
- keyword
- float
- integer
- boolean
- date
- array
- nested
- object
- ...

www.elastic.co/guide/en/elasticsearch/reference/current/mapping-types.html

Quel type choisir pour les objets embarqués ?

Object

- pour un objet unique
- www.elastic.co/guide/en/elasticsearch/reference/current/object.html

Nested

- pour les objets multiples (dans un 'array')
- chaque objet est indexé séparément dans un document caché
- www.elastic.co/guide/en/elasticsearch/reference/current/nested.html

Quel type choisir pour les chaînes de caractères ?

Text

- la chaîne est découpée en une série de termes (\approx mots)
- chaque terme subit des transformations (casse, ponctuation)
- chaque terme ainsi nettoyé est indexé individuellement

Keyword

- la chaîne entière est indexée sans aucune transformation
- utile pour les valeurs énumérées et correspondances exactes

Astuce

combinaison de Text et Keyword en indexant les deux versions

www.elastic.co/guide/en/elasticsearch/reference/current/multi-fields.html

Quel 'analyzer' choisir pour un 'text' ?

Standard \propto espaces, -ponctuation, +minuscules

Simple \propto non-lettre, +minuscules

Whitespace \propto espaces

Pattern \propto regex, (+minuscules)

Keyword ne fait rien !

Custom traitement personnalisé

Custom

char_filter remplacements optionnels de caractères

tokenizer mode de découpage de la chaîne en tokens

filter traitements optionnels des tokens (lowercase)

www.elastic.co/guide/en/elasticsearch/reference/current/analysis-analyzers.html

Rechercher efficacement

Combiner les recherches “full-text” et multi-critères



Spectrum ▾

Write your keywords here ...



Help • Log in / Register

Spectra search

Reset

Write your keywords here ...

FULL-TEXT

Search

Filters

Reset

By experiment

Owner

Database acronym

In ▾

Nothing selected ▾

Laboratory acronym

contains all words ▾

Experiment

Type

In ▾

Nothing selected ▾

Version

= ▾

Spectrum

Type

In ▾

Nothing selected ▾

Title

contains all words ▾

Date recorded

= ▾

YYYY-MM-DD

FILTRES

Full-text

- trouve tous les documents contenant globalement au moins une occurrence des termes recherchés
- ne permet pas de restreindre la recherche d'un terme dans une propriété précise

Multi-critères

- trouve tous les documents contenant précisément le terme cherché pour la propriété souhaitée
- filtres additionnels
- permet d'affiner les résultats d'une recherche full-text

Comment se fait l'indexation full-text ?

Champ masqué '_all'

- concaténation de toutes les valeurs de tous les champs
- analyzer 'standard'
- www.elastic.co/guide/en/elasticsearch/reference/current/mapping-all-field.html

Astuce

- créer un autre champ '_all' personnalisé
- y copier les champs souhaités avec l'option 'copy_to'
- www.elastic.co/guide/en/elasticsearch/reference/current/copy-to.html

Rechercher efficacement

Recherche full-text

Opérateur de recherche 'query_string'

Syntaxe riche

- AND, OR, (), ""
- wildcard '*' (0 ou + car.), '?' (1 car.), '~' (similarité)
- expressions régulières
- www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html

Astuce

- utiliser l'opérateur 'simple_query_string'
- version plus robuste et sans expressions régulières
- www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-simple-query-string-query.html

Rechercher efficacement

Filtres de recherche

match contient un des termes recherchés

match_phrase contient la phrase cherchée (même ordre)

match_phrase_prefix contient la phrase cherchée + wildcard

Attention

- appliquent des transformations sur les valeurs recherchées
- ne permettent pas de chercher des valeurs exactes

www.elastic.co/guide/en/elasticsearch/reference/current/full-text-queries.html

Opérateurs sur les valeurs exactes

term contient exactement la valeur indexée

terms contient exactement une des valeurs indexées

prefix contient une valeur indexée commençant par

wildcard contient une valeur indexée respectant le masque

range contient une valeur entre deux bornes ($<$, \leq , $>$, \geq)

exists n'est pas vide

Attention

- travaillent sur les valeurs indexées après transformation
- n'appliquent pas les transformations sur les valeurs cherchées

www.elastic.co/guide/en/elasticsearch/reference/current/term-level-queries.html

Grouper des conditions avec l'opérateur 'bool'

ET

filter toutes les conditions

must toutes les conditions + impacte le score

OU

should au moins une condition

Attention

les conditions d'un objet 'nested' doivent être au même niveau

www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html

Rechercher efficacement

Aggrégations

- équivalent à un 'GROUP BY'
- filtrage/traitement complémentaire sur les résultats
- www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html

Spectra search Reset

water schmitt Search

simulated NIR bidirectional reflection spectra ($\theta=0-70^\circ/\phi=0-70^\circ/\alpha_z=0^\circ$) of Na-Montmorillonite SWy-1 (20 μm grains) at 293K 📊 📄

2 spectra ▼ MIR optical constants spectrum of H2O Ia at 15K and Ih at 60K 📊 📄

MIR optical constants spectrum of H2O Ia at 15K 📊 📄

MIR optical constants spectrum of H2O Ih at 60 K 📊 📄

2 spectra ▼ FIR optical constants spectrum of H2O Ia at different temperatures and annealing T 📊 📄

Mise en oeuvre

Paquet Debian

- les dépôts Debian ont une version trop ancienne
- utiliser plutôt le dépôt officiel Elastic
- nécessite la présence d'un Java récent (8 ?)

www.elastic.co/guide/en/elasticsearch/reference/current/deb.html

Installation des dépendances Python

elasticsearch-dsl

- client haut-niveau
- wrapper objet pour les recherches et les docs JSON
- <https://elasticsearch-dsl.readthedocs.io/>

elasticsearch-py

- client bas-niveau (utilisé par le client haut-niveau)
- fonctionnalités avancées (manipulation des indices)
- <https://elasticsearch-py.readthedocs.io/>

Attention

- installer les versions des lib compatibles avec le serveur ES
- module 'pyramid_es' trop ancien (bloqué sur ES version 1)

Exemple de code

- manipulation des indexes
- manipulation des documents
- définition des mappings
- définition des documents
- définition des requêtes
- définition des conditions de requêtes
- affichage des résultats

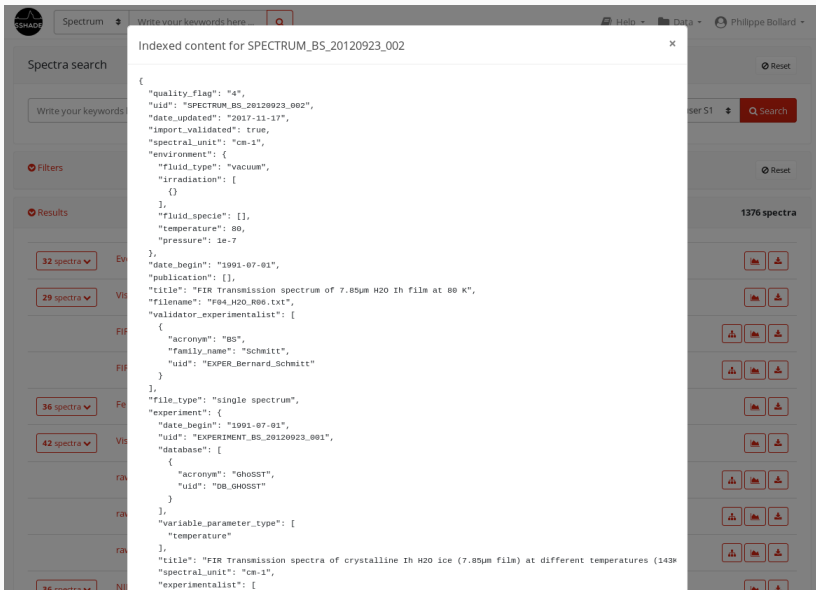
Debug

```
GET http://localhost:9200/ssshade_spectrum/spectrum/_count  
[status:200 request:0.013s]
```

```
{"query": {"bool": {"filter": [{"terms": {"type": ["transmission"]}}]},  
"must": [{"simple_query_string": {"query": "water", "default_operator"  
: "AND", "fields": ["_all_provider1"]}}]}}
```

```
{"count":112,"_shards":{"total":5,"successful":5,"skipped":0,"failed":0
```

Affichage du JSON indexé pour chaque résultat de recherche



The screenshot shows a web application interface for searching spectra. A modal window is open, displaying the indexed JSON content for a specific search result. The JSON data is as follows:

```
{
  "quality_flag": "4",
  "uid": "SPECTRUM_BS_20120923_002",
  "date_updated": "2017-11-17",
  "import_validated": true,
  "spectral_unit": "cm-1",
  "environment": {
    "fluid_type": "vacuum",
    "irradiation": {
      {}
    },
  },
  "fluid_specie": [],
  "temperature": 80,
  "pressure": 1e-7
},
"date_begin": "1991-07-01",
"publication": [],
"title": "FIR Transmission spectrum of 7.85µm H2O 1h film at 80 K",
"filename": "F04_H2O_R00.txt",
"validator_experimentalist": [
  {
    "acronym": "BS",
    "family_name": "Schmitt",
    "uid": "EXPER_Bernard_Schmitt"
  }
],
"file_type": "single spectrum",
"experiment": {
  "date_begin": "1991-07-01",
  "uid": "EXPERIMENT_BS_20120923_001",
  "database": [
    {
      "acronym": "GHOSST",
      "uid": "DB_GHOSST"
    }
  ]
},
"variable_parameter_type": [
  "temperature"
],
"title": "FIR Transmission spectra of crystalline 1h H2O ice (7.85µm film) at different temperatures (143",
"spectral_unit": "cm-1",
"experimentalist": [
```

Elasticsearch-Head : Interface de gestion d'Elasticsearch

The screenshot displays the Elasticsearch-Head interface. At the top, the status is 'Santé du cluster: yellow (120 240)'. Below the navigation bar, a search results table is visible. A modal window titled 'Résultat au format JSON' is open, showing a detailed JSON document for a 'spectrum' entry. The document includes fields like 'index', 'type', 'id', 'version', 'score', 'source', 'quality flag', 'validator', 'acronym', 'family name', 'uid', 'date updated', 'import validated', 'experiment', 'date begin', 'titles', 'type', 'laboratory', and 'database'.

Index	Type	ID	Score	Source	Quality Flag	Validator	Acronym	Family Name	UID	Date Updated	Import Validated	Experiment	Date Begin	Titles	Type	Laboratory	Database
sshade_liquid	spectrum	46e90a63-b016-4f7d-83ab-3deabe72de58	1														
sshade_material	spectrum	6b987cc7-a895-41a6-87c3-29ed519717dd	1														
sshade_matter	spectrum	713b2d35-1474-4e18-9ffe-22a11b6ae192	1														
sshade_mineral	spectrum	8725d63e-79a0-4369-b935-c47f414ae9c4	1														
sshade_molecule	spectrum	b7b67aa3-b361-41e3-ad36-072a941fd26	1														
sshade_object_idp	spectrum	0c9fbf04-fd4b-486d-a291-5d0c2173cfa1	1														
sshade_object_meteorite	spectrum	1-a-4f7b-53-4-31-887-01b-4438018f	1														
sshade_object_micrometeorite	spectrum	480c2042-20f1-4239-b9ae-300921e9030	1														
sshade_openium	spectrum	24c70b0b-2287-4e36-bfe2-90aff74df1e1	1														
sshade_publication	spectrum	84d61841-4ef2-4c15-b14a-5fb7d663cdd8	1														
sshade_sample	spectrum	6bc574aa-5bf5-4e09-9d82-8c18d9ff4549	1														
sshade_solid	spectrum	c3510482-9b25-4d8c-afae-42c5a95f17db	1														
sshade_spectrum	spectrum	ce64c4d3-4239-4caf-8fdb-6b26605c77e4	1														
atom	spectrum	e3764ab7-cbe0-4887-9447-badc66ccf16	1														
chemical_bond	spectrum	f0ee68f1-0e6d-480a-9d18-a7789ef14a2e	1														
chemical_function	spectrum	76caf1d6-9591-4c00-a6fe-13cf7d4970ba	1														
constituent	spectrum	41243975-a113-4409-9c9a-d20de775b2d5	1														
database	spectrum	5ea321cc-fa0d-4c16-8705-862343eaa51	1														
experiment	spectrum	9754e47e-04b0-b6fa-c24908199c67	1														
experimentalist	spectrum	b9d898eb-86db-4a11-9c16-7126b9666e9d	1														
instrument	spectrum	c06cef62-12b3-40a3-9cfd-10007e253267	1														
journal	spectrum	eca34e6c-e3b9-496f-b606-3f8230d266fd	1														
laboratory	spectrum	92c7c1fb-b63d-439c-9576-7918ddeb3e8	1														
layer	spectrum	92c7c1fb-b63d-439c-9576-7918ddeb3e8	1														
liquid	spectrum	92c7c1fb-b63d-439c-9576-7918ddeb3e8	1														
material	spectrum	543e4cfd-0d25-4dc9-9ee2-51ea44d656e6	1														
matter	spectrum	044577c9-c7ff-499e-8f82-02019d3aae9	1														
mineral	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
molecule	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
object_idp	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
object_meteorite	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
object_micrometeorite	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
openium	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
publication	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
sample	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
solid	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
spectrum	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
CHAMPS	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														
all_providero	spectrum	0199750-d23f-40a1-a28b-625537a67613	1														

```
{
  "index": "sshade_spectrum",
  "type": "spectrum",
  "id": "c3510482-9b25-4d8c-afae-42c5a95f17db",
  "version": 1,
  "score": 1,
  "source": {
    "acronym": "SP",
    "family_name": "Potin",
    "uid": "EXPER_Sandra_Potin",
    "date updated": "2018-05-14",
    "import validated": true,
    "experiment": {
      "date begin": "2018-04-03",
      "titles": "NMR inelastic neutron spectroscopy of the spin Hall effect of light and e60" of 4 lunar soils (Apollo 15, 16, 17) under ambient conditions",
      "variable parameter type": "emergence angle"
    },
    "laboratory": "I",
    "database": "I"
  },
  "quality flag": "gim",
  "validator": {
    "experimentalist": "76caf1d6-9591-4c00-a6fe-13cf7d4970ba"
  }
}
```

Conclusion

Recommandations

- n'indexer que les infos réellement pertinentes
- définir les mappings en fonction des données/recherches
- définir des '_all' personnalisés
- personnaliser les analyzers
- combiner 'text' et 'keyword'
- combiner 'simple_query_string' et des filtres

Attention

- données dupliquées car dénormalisation du schéma
- CRON pour tout effacer et réindexer

Questions?