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

Plan

Contexte

SSHADE: Solid Spectroscopy Hosting Architecture of DBs

`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)

Projet européen (Europlanet 2020-RI / Workpackage VESPA)

20 bases actives

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

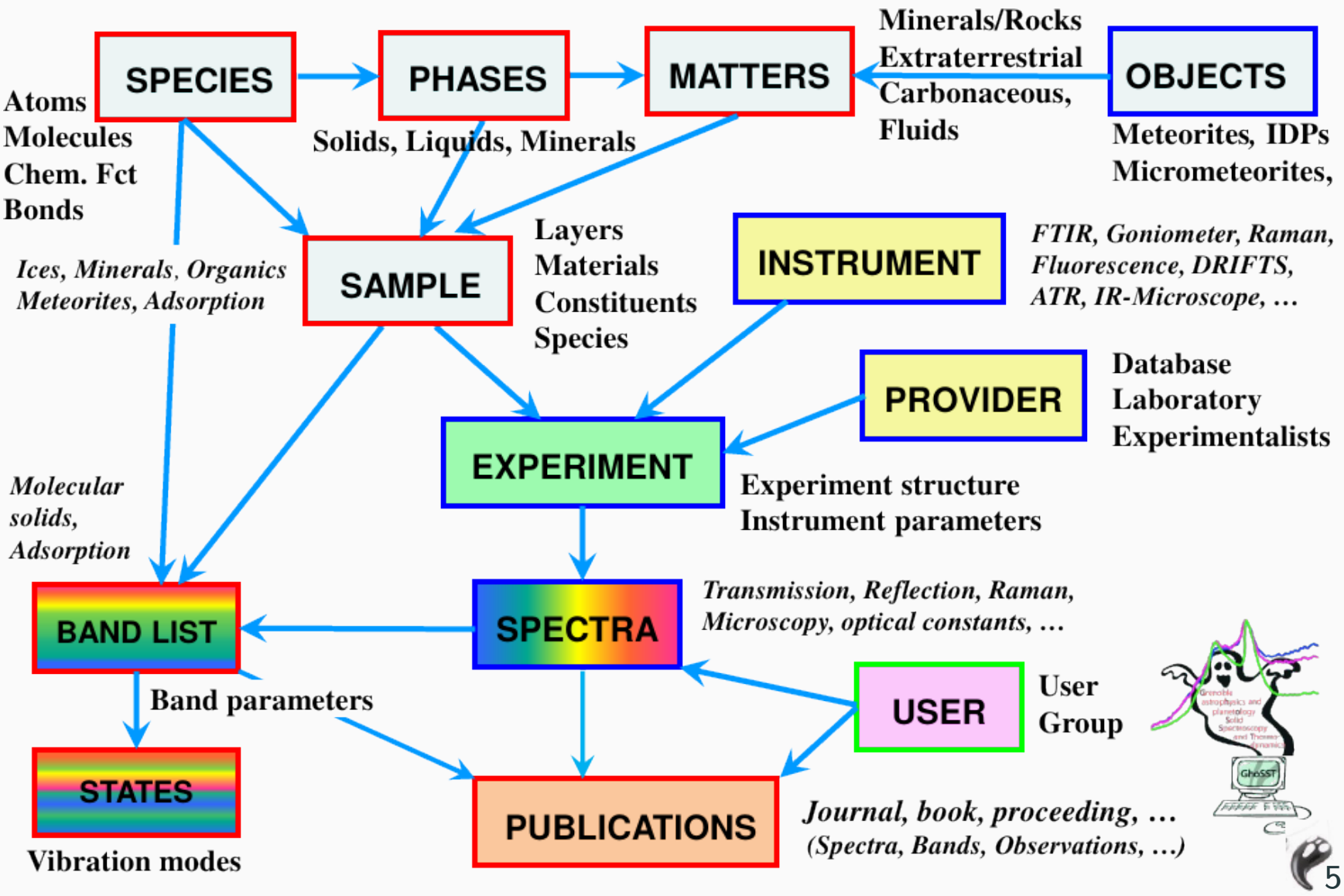
Et d'autres à venir !

Choix techniques

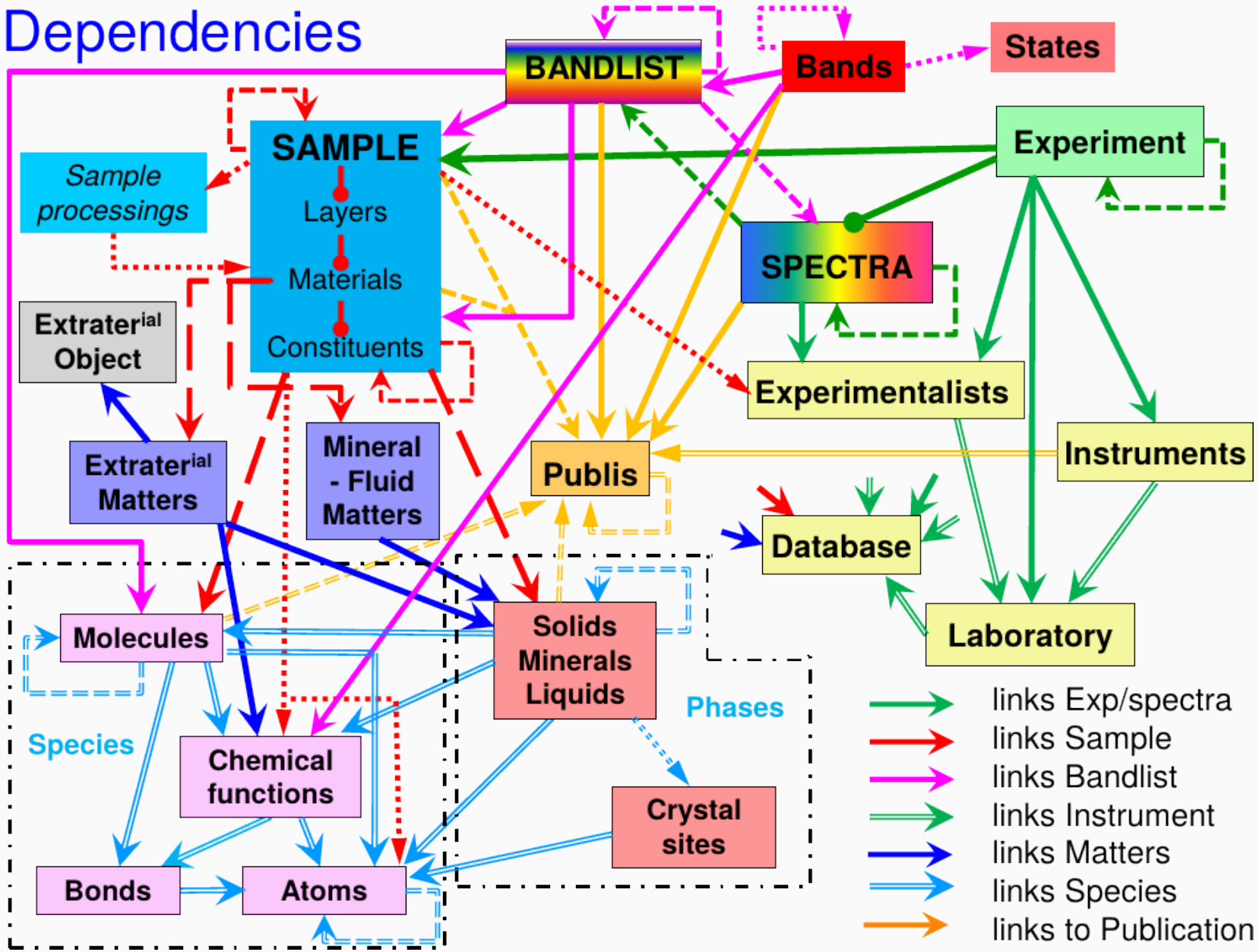
et GAVO DaCHS pour la partie VO / EPN-TAP

Problématique

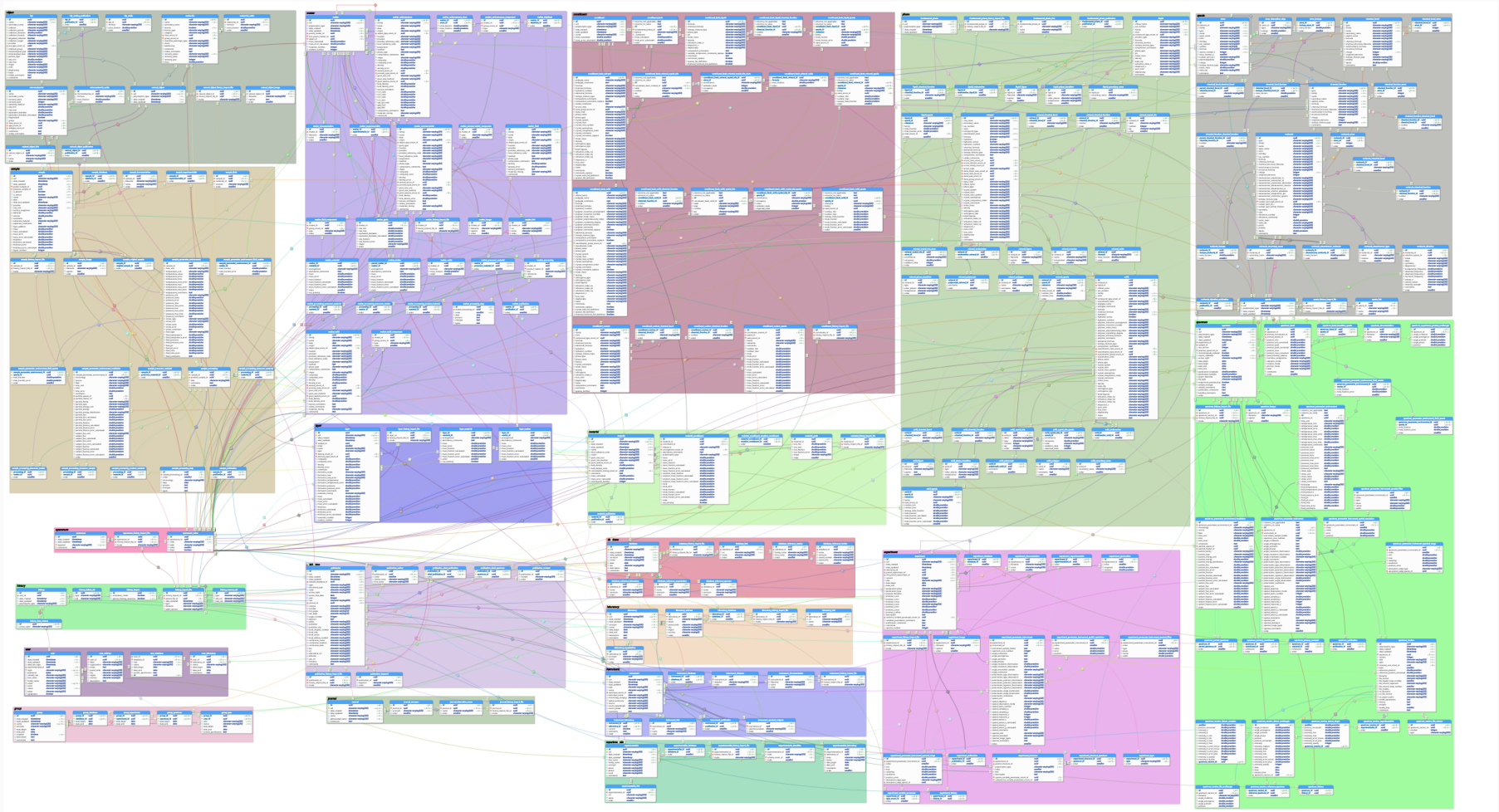
SSDM: Solid Spectroscopy Data Model



SSDM: Solid Spectroscopy Data Model



Implémentation PostgreSQL



≈ 250 tables réparties sur 20 schémas

Problématique

Moteur de recherche

Problème : Recherches ciblant de nombreuses tables

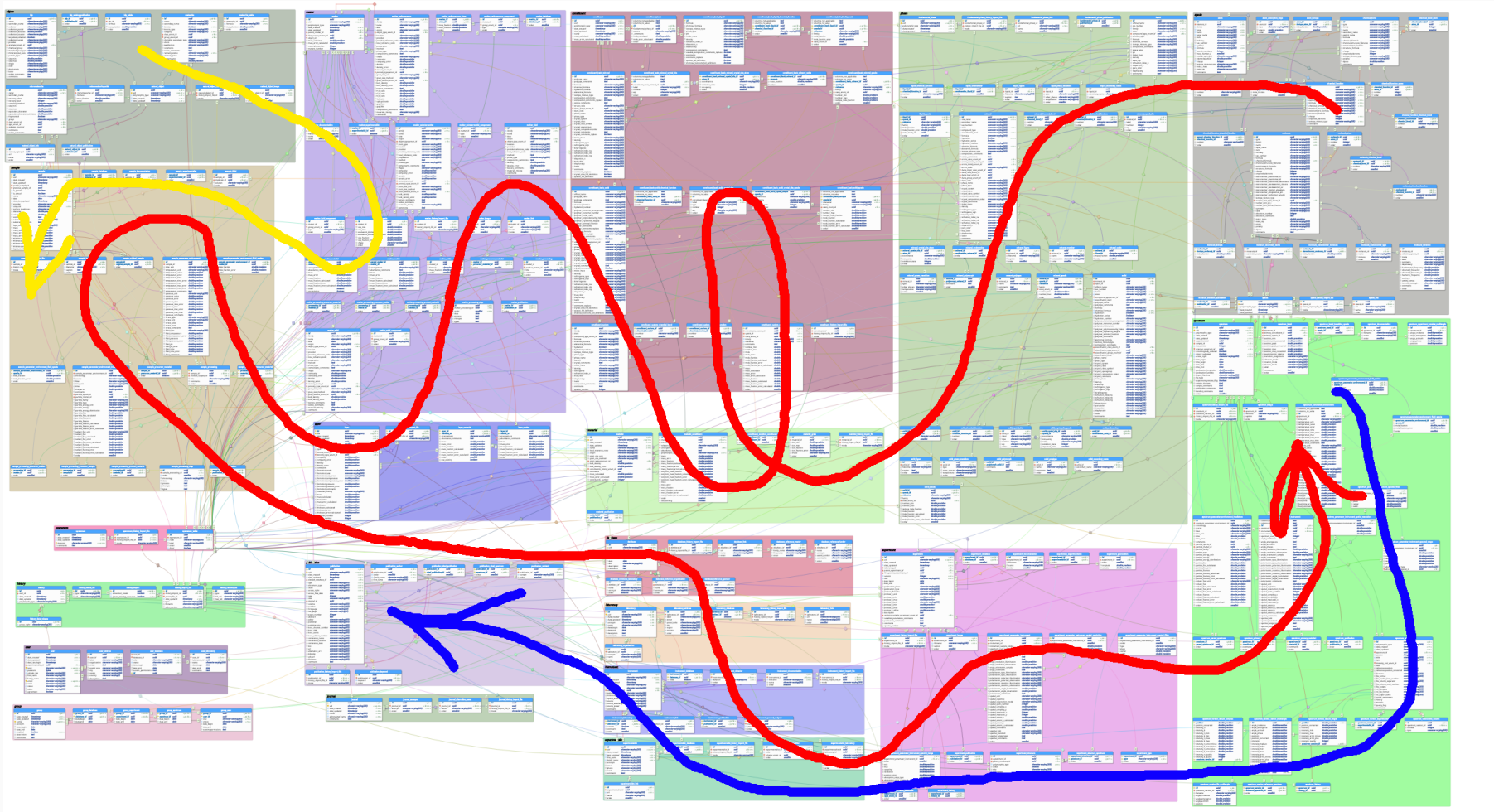
Exemples de recherche

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

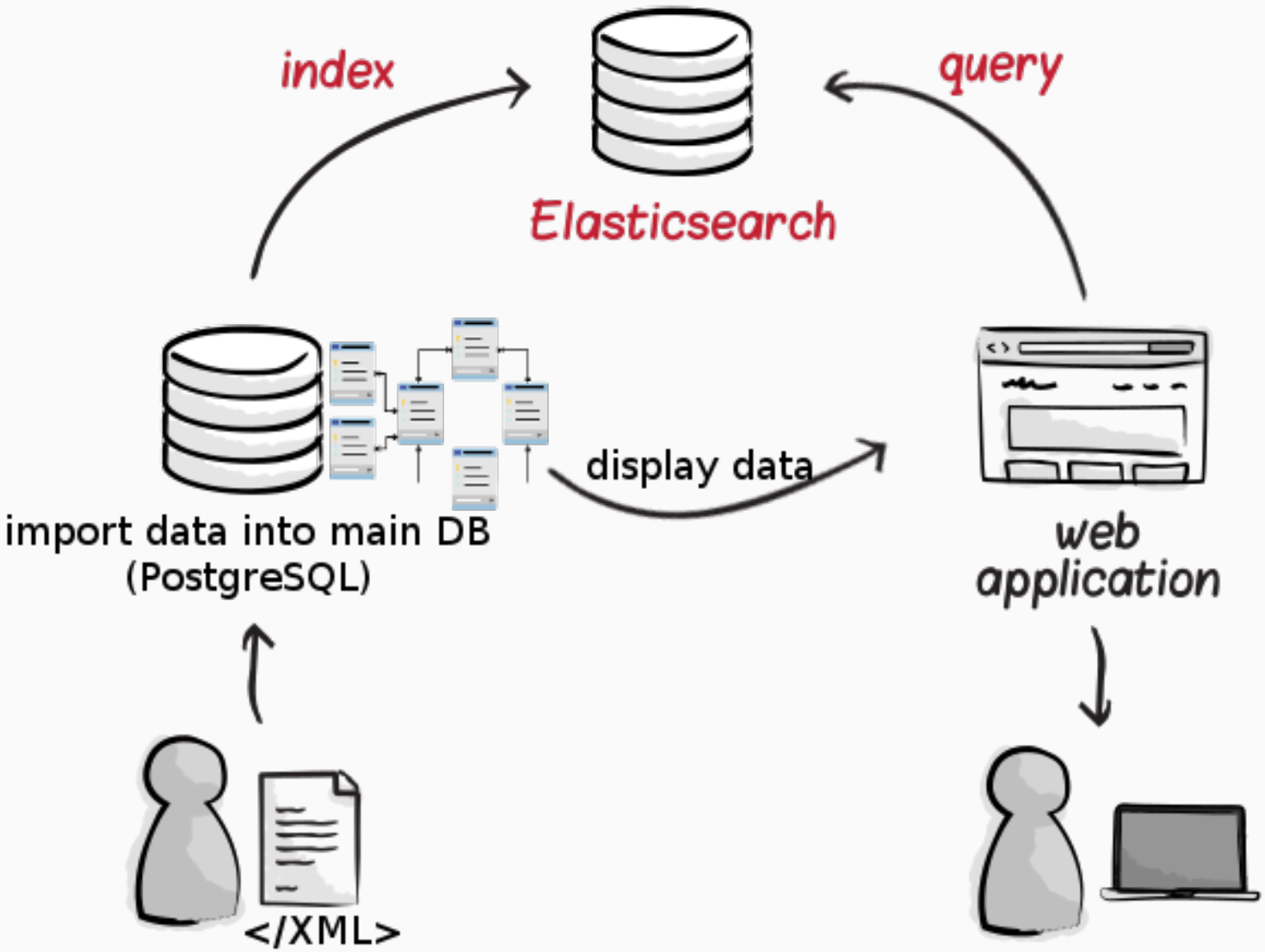
Complexité du modèle de données relationnel

- 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

Problème : Complexité du formulaire de recherche

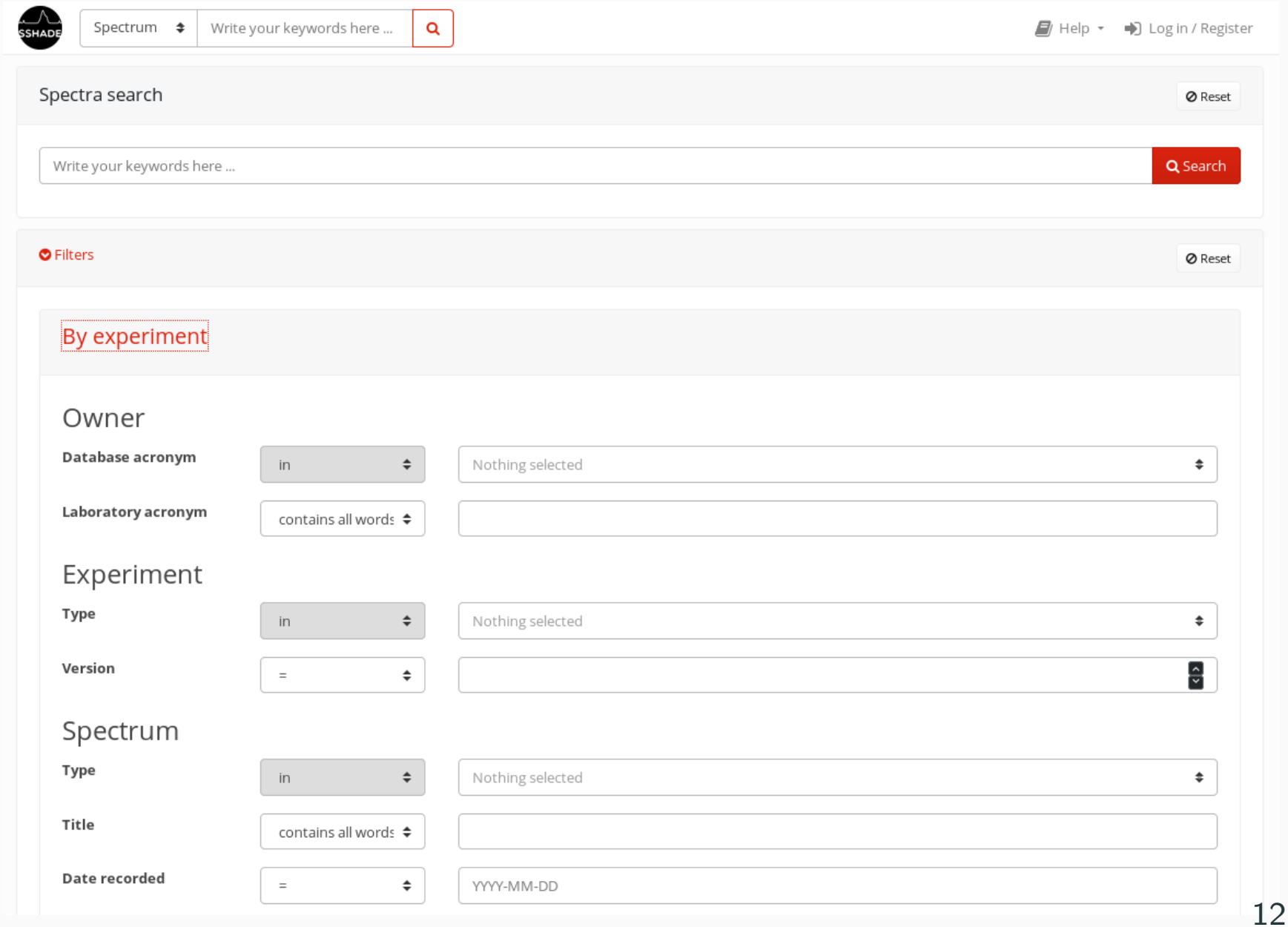
Pour l'utilisateur

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

Pour le développeur (SQL)

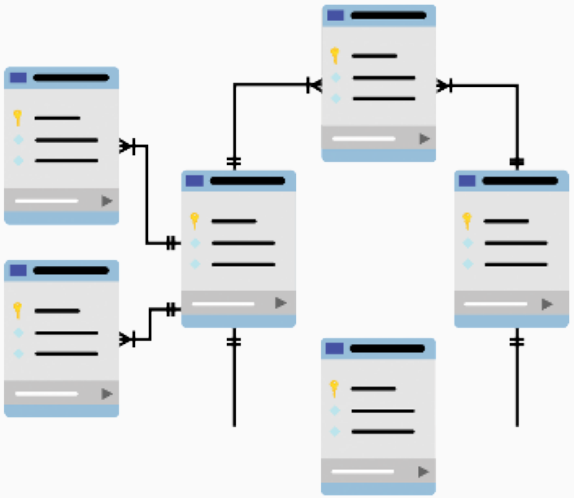
- Nombreux champs = c'est faisable avec plusieurs JOIN mais...
- Un seul champ = quoi, tu veux vraiment ça... ? Pour hier ?

Solution : Combiner les deux approches



Indexer efficacement

Comment traduire un schéma relationnel en structure JSON ?



Relational model

```json
[
    {
        "name": "Top Level",
        "parent": "null",
        "children": [
            {
                "name": "Level 2: A",
                "parent": "Top Level",
                "children": [
                    {
                        "name": "Son of A",
                        "parent": "Level 2: A"
                    },
                    {
                        "name": "Daughter of A",
                        "parent": "Level 2: A"
                    }
                ]
            },
            {
                "name": "Level 2: B",
                "parent": "Top Level"
            }
        ]
    }
]
```

JSON document

Indice, mapping et type

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

$\Longrightarrow$ 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 séparemment 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 tranformation
- utile pour les valeurs énumérées et correspondances exactes

Astuce

combiner 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

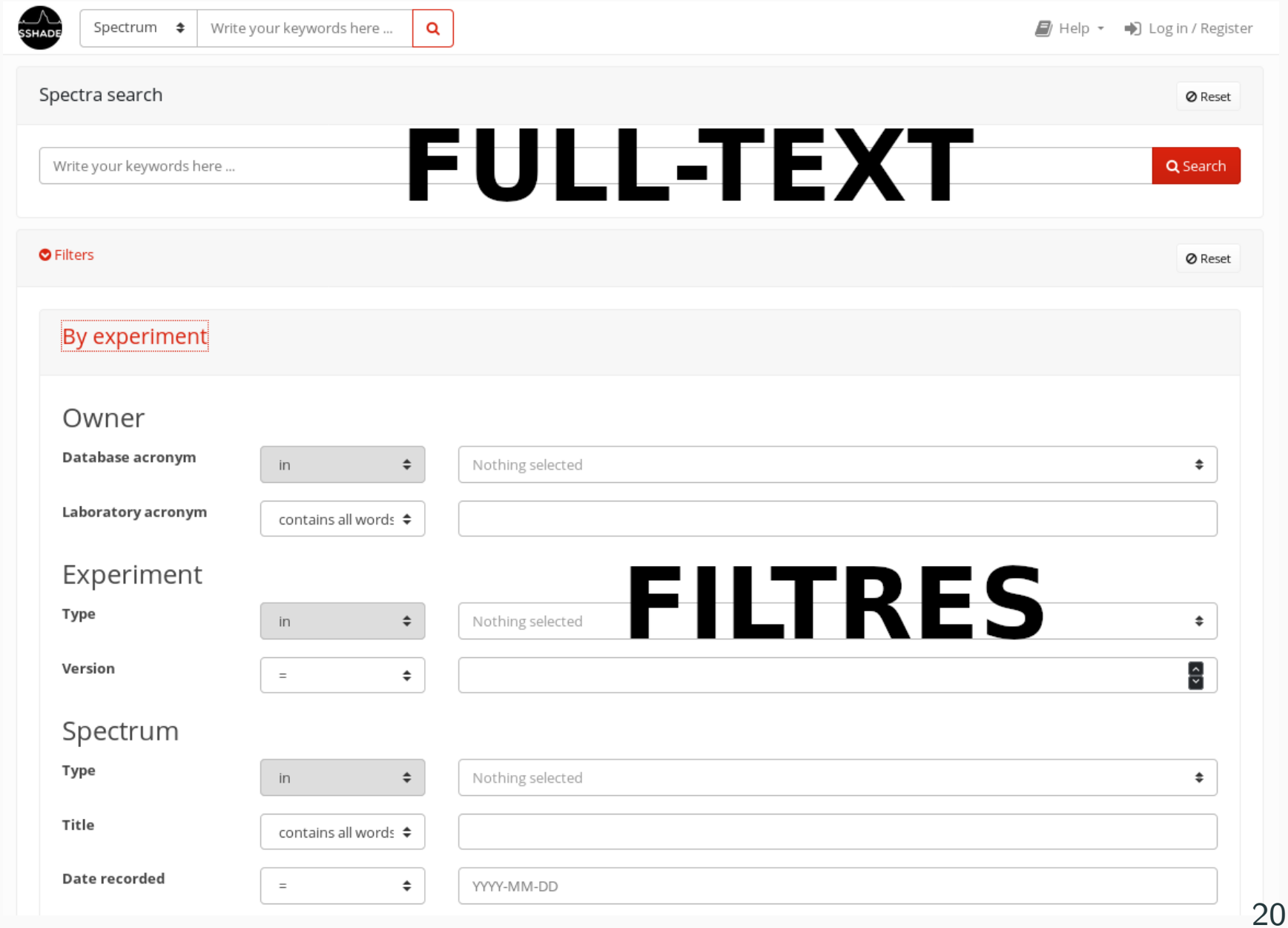
Custom

char_filter	remplacements optionnels de caractères
tokenizer	mode de découpage de la chaîne en tokens
filter	traitements optionnels des tockens (lowercase)

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

Rechercher efficacement

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



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

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

Opérateurs full-text

match contient un des termes cherchés

match_phrase contient la phrase cherchée (même ordre)

match_phrase_prefix contient la phrase cherchée + wildcard

Attention

- appliquent des transformations sur les valeurs cherchées
- 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

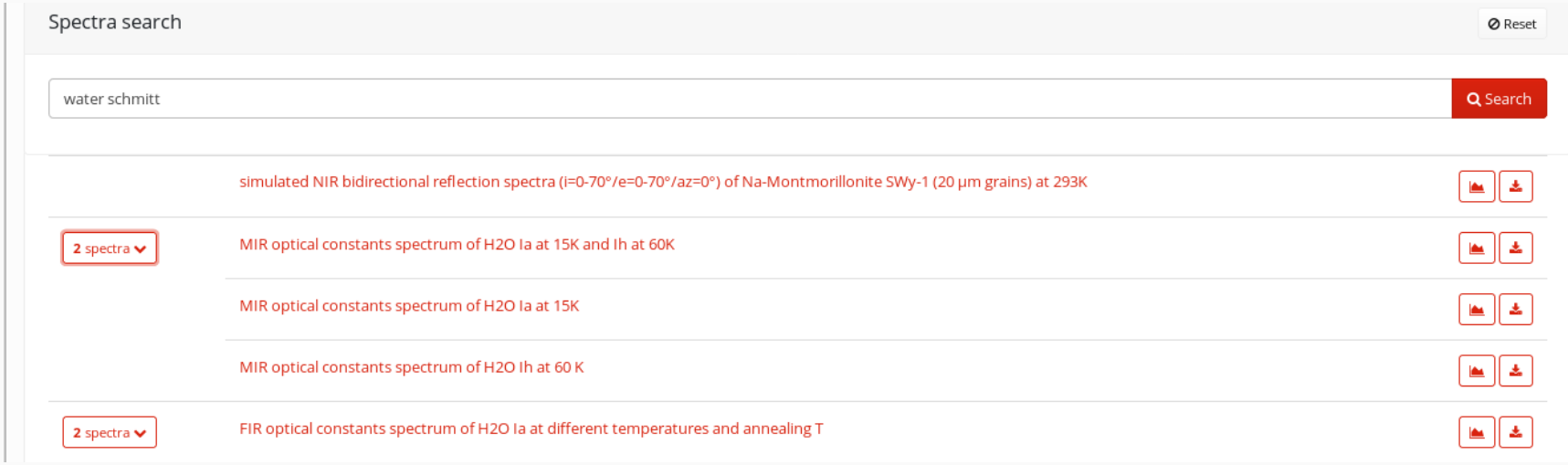
`www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html`

Rechercher efficacement

Aggrégations

Aggrégations

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



Mise en oeuvre

Installation du serveur Elasticsearch

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)

Implémentation SSHADE

Exemple de code

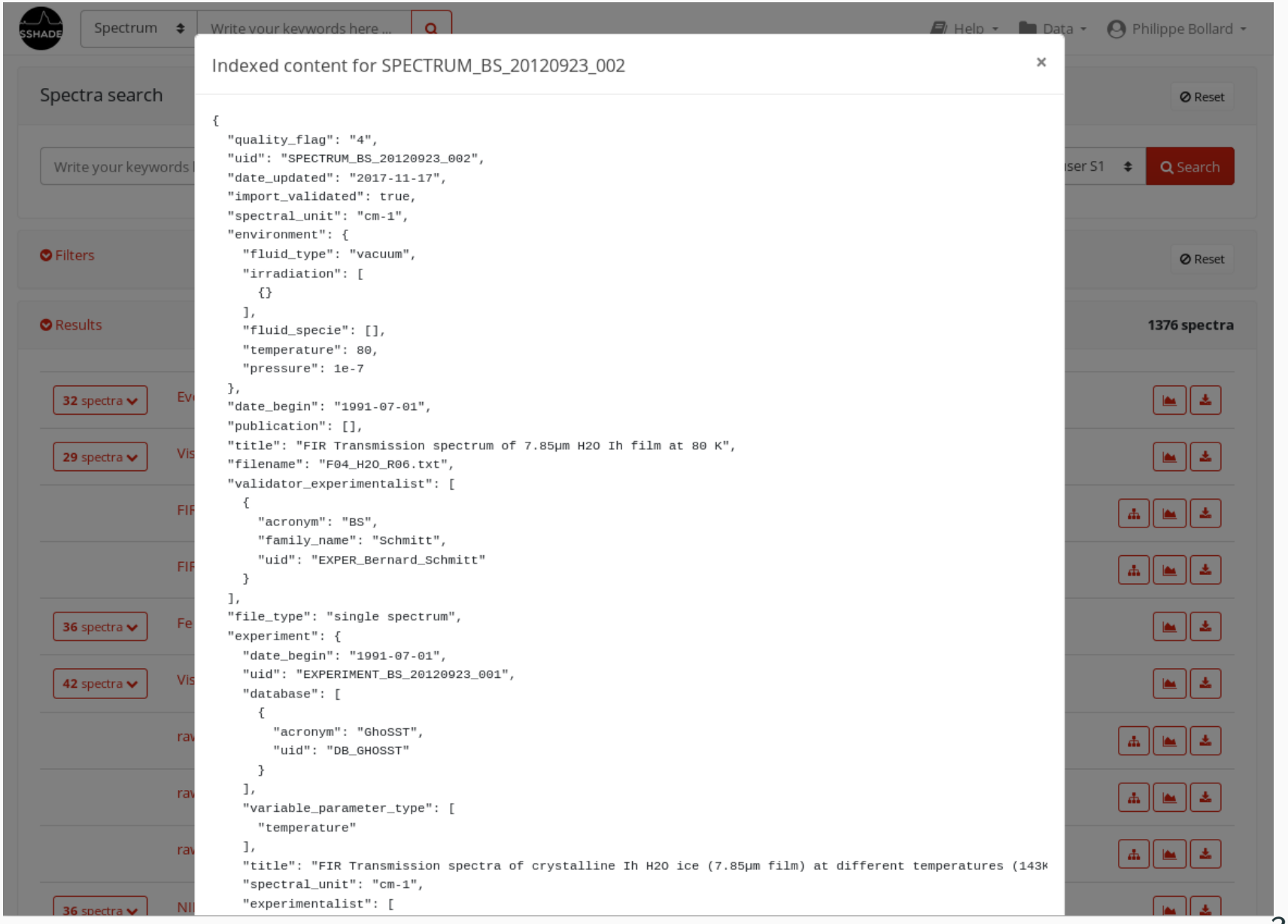
- manipulation des indexes
- manipulation des documents
- définition des mappings
- définition des documents
- définition des requêtes
- définition des conditions de requêtes
- affichage des résultats

Debug

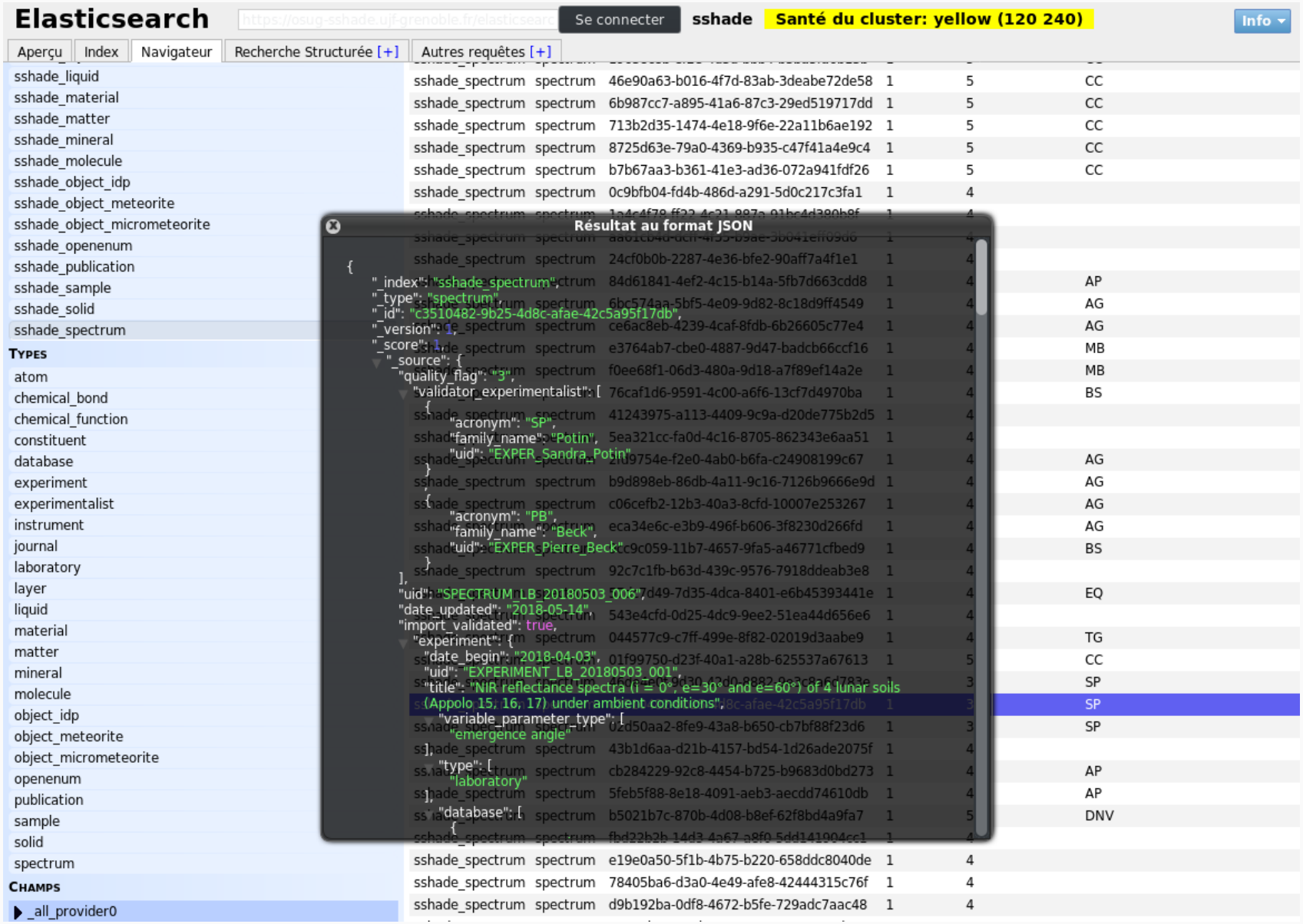
Logs

```
GET http://localhost:9200/sshade_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



Elasticsearch-Head : Interface de gestion d'Elasticsearch



https://mobz.github.io/elasticsearch-head/

Conclusion

En résumé

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?