

Mathrice ANF 2018

Module BD4

**Implémentation d'une Plate-forme d'exploration interactive de données complexes et massives
(Preuve de concept)**

*Christophe Cancé
Univ. Grenoble Alpes
UMS GRICAD*



I. Approche interactive

- les couches : collecte, requête, visualisation, couche élaborée d'informations

II. Calibrage

- Ordres de grandeurs et enseignements de l'Analyse de logs avec ELK
- Introduction aux ETL

III. Collecte et organisation de l'information

Acheminement de l'information vers un modèle cible

IV. Bases graphes

- Choix d'une solution
- structure, propriétés
- exploitation, intégration

V. IHM de navigation dans l'information

Apports des SIG et webmapping avec Openlayers.

VI. Moteur de recherche et Visualisation

VII : production d'information métier.

Exemple : trajectoire

I. Contexte, objectifs et moyens

Problématique :

- Réunir l'information pertinente sous une forme organisée et interopérable.
- L'interroger de façon interactive sur des volumes significatifs
- Observer les résultats intermédiaires
- Permettre d'élaborer de nouvelles couches d'information métier

Contexte : économie de code, briques interchangeable

Moyens : intégration, généricité du code et cohérence de la donnée

- Approche graphes : intérêt et points d'attention
- Focus collecte de données ETL
- Lac de données et requêtes: base graphe
- IHM
- Moteur de recherche et data-visualisation

Conception d'un lac de données: préalable à l'intégration des données

- Contexte de données potentiellement hétérogènes, complexes et massives stockées dans des bases de données disparates (cas de silos multiples)
- Objectif: rassembler les informations collectées par la couche ETL sous une forme organisée et exploitable
 - => concevoir le schéma cible de données, au plus près du sens intrinsèque du phénomène qui doit être observé.

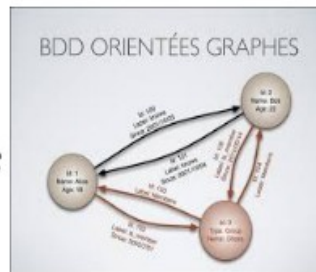


Extraire, Enrichir
transformer

ETL



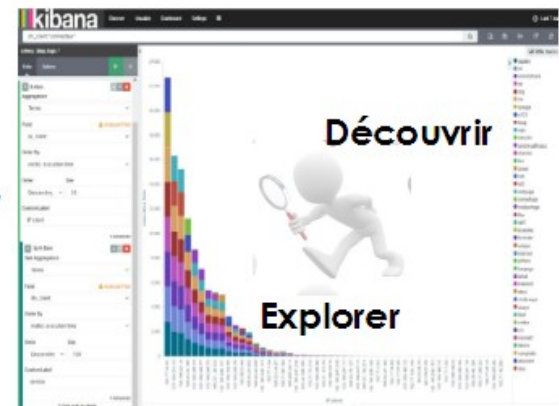
Intégrer



Lac de données



Forer



Moteur de recherche, visualisation

Figure 3: Alimentation du lac de données, gisement d'information

ETL : collecte de l'information vers une structure cible

- Hétérogénéité des sources
- Workflow de traitements
- Réutilisation, industrialisation de process type « Business process Management »
- Evolutivité du workflow

- Cas d'usages courants
 - Collecte massive d'information de type logs ou de données web type twitter => peut nécessiter aménagement d'un contrôle de flux (tampon)
ex : RabbitMQ, beats - logstash
 - Collecte multi-sources des bases de données en mode « pull »

Quelques solutions ETL open source

- Nifi : logiciel libre né à la NSA en 2006, intègre la fondation apache en 2014
<https://nifi.apache.org/>
- Logstash : brique d'ingestion de données de la pile Elastic
<https://www.elastic.co/fr/products/logstash>
- Talend : membre de la fondation Apache depuis
<https://fr.talend.com/>

II. Calibrage et Analyse de logs LDAP avec Elastic contexte

- **LDAP : Usage massif**
 - Annuaires : cœur du contrôle d'accès aux services numériques (niveaux local, global)
 - COMUE GA : ~66 millions de traces d'activité LDAP/j
- **Structure des logs LDAP**
 - Traces enchevêtrées dans le temps
 - Service, serveur, utilisateur ne sont jamais simultanément présents dans une entrée du fichier de logs

Exemple de Log d'annuaire LDAP

- ▶ [29/Mar/2016:14:46:42 +0200] **conn=5294995** op=-1 msgId=-1 - **fd=166 slot=166 LDAP connection from 152.77.188.30:56127 to 152.77.18.142**
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=0 msgId=1 - **BIND**
dn="cn=sarad,ou=machines,ou=ujf,dc=agalan,dc=org" method=128 version=3
- ▶ [29/Mar/2016:14:46:42 +0200] conn=-1 op=-1 msgId=-1 - **SRCH**
base="cn=sarad,ou=machines,ou=ujf,dc=agalan,dc=org" scope=0 filter="(|(objectclass=*)(objectclass=ldapsubentry))" attrs=ALL
- ▶ [29/Mar/2016:14:46:42 +0200] conn=-1 op=-1 msgId=-1 - ENTRY dn="cn=sarado, ou=machines, ou=ujf, dc=agalan,dc=org"
- ▶ [29/Mar/2016:14:46:42 +0200] conn=-1 op=-1 msgId=-1 - **RESULT** err=0 tag=101 nentries=1
etime=0.000250
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=0 msgId=1 - RESULT err=0 tag=97 nentries=0
etime=0.000690 dn="cn=sarad,ou=machines,ou=ujf,dc=agalan,dc=org"
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=1 msgId=2 - SRCH
base="ou=ujf,dc=agalan,dc=org" scope=1 filter="(cn=lock)" attrs="agInINE"
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=1 msgId=2 - ENTRY
dn="cn=lock,ou=ujf,dc=agalan,dc=org"
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=1 msgId=2 - **RESULT** err=0 tag=101 nentries=1
etime=0.000700
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=2 msgId=3 - **SRCH**
base="ou=groupes,ou=ujf,dc=agalan,dc=org" scope=1 filter="(cn=ujf-pers-etab)" attrs="cn
gidNumber member"
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=2 msgId=3 - ENTRY dn="cn=ujf-pers-
etab,ou=groupes,ou=ujf,dc=agalan,dc=org"
- ▶ [29/Mar/2016:14:46:42 +0200] conn=5294995 op=2 msgId=3 - **RESULT** err=0 tag=101 nentries=1
etime=0.050670

Problématique

Comment,

- connaître la part d'activité de l'annuaire consacrée à un service ? à un serveur ?
- Produire des indicateurs d'usages des annuaires ?
- Parcourir les données ?
- Naviguer / Requêter
- Présenter des vues synthétiques de ces données

...en limitant l'écriture de code ?

Test de la pile Elastic (ELK)

Elasticsearch

moteur de recherche et d'analyse HTTP REST JSON distribué (éclats répartis sur des noeuds)

Technologie d'indexation fondée sur Apache Lucene
tokenization

Recherche full text, fuzzy,..

Logstash (Extract Transform Load)

Collecte l'information issue de différentes sources, la transforme, l'enrichit, l'injecte dans des index ES par exemple. Permet de générer des index performants (structure, redondance).

http://logstash-docs.elasticsearch.org.s3.amazonaws.com/_logstash_config_language.html

Kibana requête ES, visualisation, navigation, management

Logstash - Paramétrage 3 sections : INPUT , FILTER, OUTPUT

ex : input depuis des fichier :

```
input {  
  file {  
    path => "/var/log/messages"  
    type => "syslog"  
  }  
}
```

```
file {  
  path => "/var/log/apache/access.log"  
  type => "apache"  
}  
}
```

ex : input depuis un port UDP

```
input {  
  syslog {  
    port => 514  
  }  
}
```

Traitement : le parser décortique, retient, modifie, complète

```
filter {  
  
  grok {  
    patterns_dir => "/home/logstash/conf/patterns"  
    match => ["message", "\\[%{HTTPDATE:log_date}\\]\\sconn=%{INT:connexion}\\sop=%  
{INT:op}\\smsgid=%{INT:msgid}\\s-\\s%{OPERATION:operation}\\s*%{GREEDYDATA:suite}"]  
  }  
  
  if [operation] == "fd=" {  
  
    grok {  
      match => ["suite", "%{INT:fd}\\slot=%{INT:slot:int}\\sLDAP(S)?\\sconnection\\sfrom\\s%  
{IP:ip_client}:%{INT}\\sto\\s%{IP:ip_serveur}"]  
    }  
  
    aggregate {  
      task_id => "%{connexion}"  
      code => "map['ip_client'] = event['ip_client']"  
      map_action => "create"  
    }  
  }  
}
```

Traitement : le parser décortique, retient, modifie, complète

```
if [operation] == "BIND" {  
  
  grok {  
    match => ["suite", "dn=%{QS:dn}\\smethod=%{INT:method}\\sversion=%  
{INT:version}"]  
  }  
  
  aggregate {  
    task_id => "%{connexion}"  
    code => "map['dn_client'] = event['dn']"  
    map_action => "update"  
  }  
  
  aggregate {  
    task_id => "%{connexion}"  
    code => "event['ip_client'] = map['ip_client']"  
    map_action => "update"  
  }  
  
  mutate {  
    rename => {"dn" => "dn_client"}  
  }  
}
```

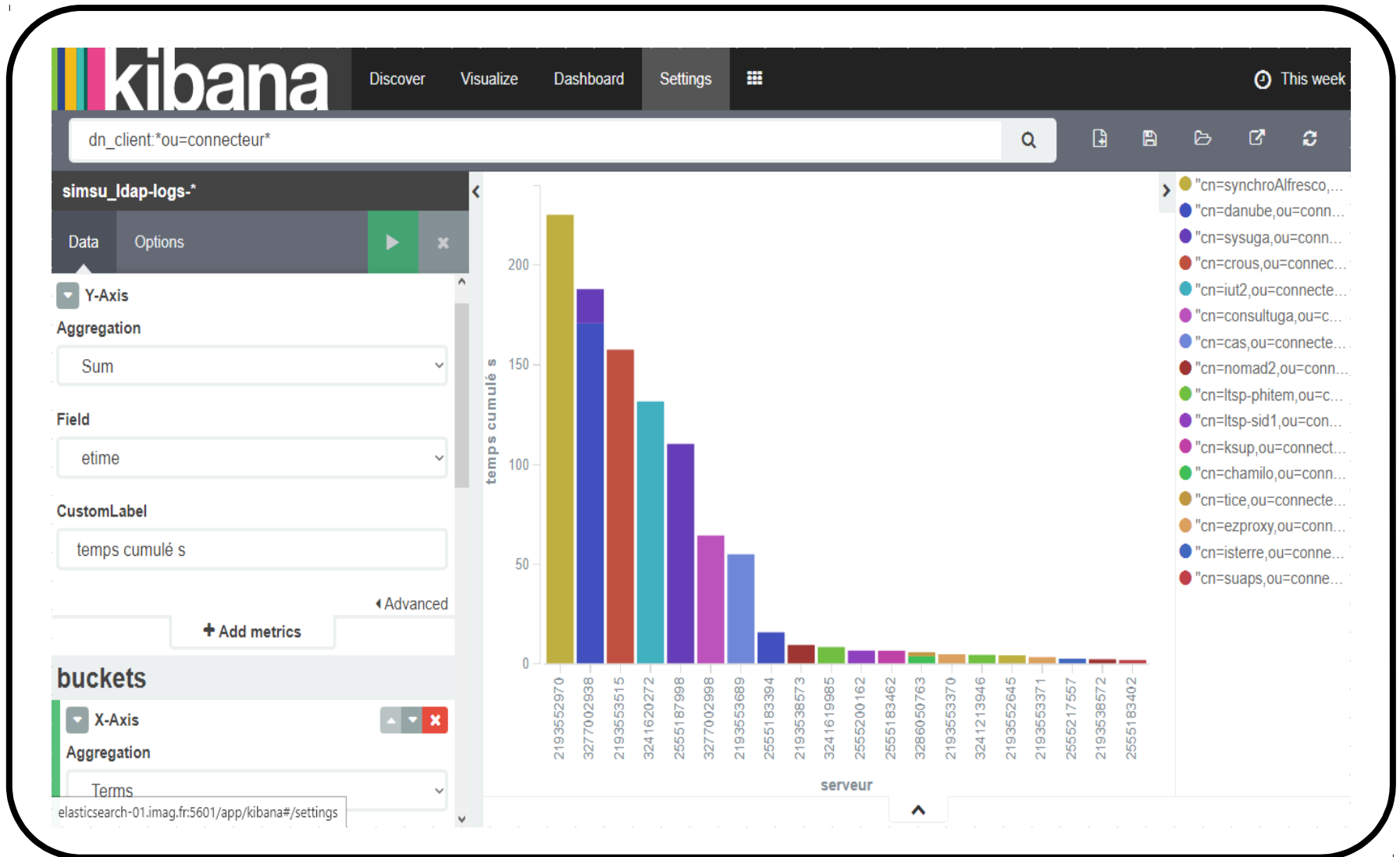
Traitement : le parser décortique, retient, modifie, complète

```
if [operation] == "RESULT" {  
  
  grok {  
    match => ["suite", "err=%{INT:err}\stag=%{INT:tag}\snentries=%{INT:nentries}\setime=%  
{NUMBER:etime}(\sdn=%{QS:dn})?"]  
  }  
}  
  
if [operation] in ["RESULT", "SRCH"] {  
  
  aggregate {  
    task_id => "%{connexion}"  
    code => "event['ip_client'] = map['ip_client'];  
           event['dn_client'] = map['dn_client']"  
    map_action => "update"  
  }  
}
```

Paramétrage de la sortie, ici vers Elasticsearch

```
output {
  if "_grokparsefailure" in [tags] {
    file { path => ["/home/logstash/data/REJET"] }
  }
  else if [operation] in ["fd=", "BIND", "RESULT", "SRCH", "UNBIND"] {
    elasticsearch {
      hosts => ["http://elasticsearch-01.imag.fr:9200"]
      index => "simstu_ldap-tp-logs-%{+YYYY.MM.dd}"
      template => "/home/logstash/conf/logs-ldap_simstu_mapping.json"
      template_name => "logs-ldap_mapping"
    }
  }
  else {
    file { path => ["/home/logstash/data/AUTRES"] }
  }
}
```

Synthèse d'usage du LDAP



Calibrage - Analyse de logs LDAP avec Elastic - Bilan

- 1 VM « standard » avec 4 coeurs 16Go RAM, 1 To Disque, QoS 2000 IOps
- 66 millions de lignes par jour en moyenne sur la COMUE GA
- Maximum admissible observé ~1000/lignes/sec en lecture sur un port UDP
- Trop juste pour le temps réel => lecture depuis un fichier => dérive 8h/j
- Agrégations concentriques sur 500 000 000 d'entrées en ~1 seconde
- Pile Elastic search Kibana adaptée pour forage interactif dans les données
 - Organisation de la redondance
 - Projection dans des index
 - Usage en lecture
- <https://www.jres.org/fr/presentation?id=90>

III. Collecte et organisation de l'information

- Extract Transform Load – Talend - <https://fr.talend.com/>
- Logiciel d'intégration de données open source
2006 : https://fr.wikipedia.org/w/index.php?title=Talend_Open_Studio_for_Data_Integration&action=edit&redlink=1
- Talend membre de la fondation Apache de 2010
- Générateur de code Java
- Business model open Core : fonctionnalités supplémentaires payantes
support, outils de développement enrichis, ...
- 900 composants et connecteurs utilisables dans des jobs
- Sequenceur de jobs
- A un projet correspond un espace de travail, des jobs

Job : graphe d'exécution de traitements choisis parmi 900 composants

The screenshot displays the Talend Open Studio for Big Data interface (version 6.2.1.20160704_1411) connected to a local LDAP2Elastic source. The main workspace shows a job design for 'Job cim10' with the following components and data flow:

- tFileInputDelimited_1**: Reads 40519 rows in 0,33s at 122413,9 rows/s. It feeds into **row4 (Lookup)**.
- tRowGenerator_1**: Generates 1500 rows in 0,5s at 3000 rows/s. It feeds into **row1 (Main)**.
- tMap_1**: Processes 1500 rows in 0,5s at 3000 rows/s. It feeds into **diag2elastic (Main)**.
- tLogRow_2**: Logs 1500 rows in 0,5s at 2935,42 rows/s. It feeds into **row2 (Main)**.
- tMap_2**: Processes 1500 rows in 0,5s at 2935,42 rows/s. It feeds into **diag (Main)**.
- tLogRow_1**: Logs 1500 rows in 0,51s at 2923,98 rows/s. It feeds into **row3 (Main)**.
- FileOutputJSON_1**: Writes 1500 rows in 0,51s at 2923,98 rows/s.

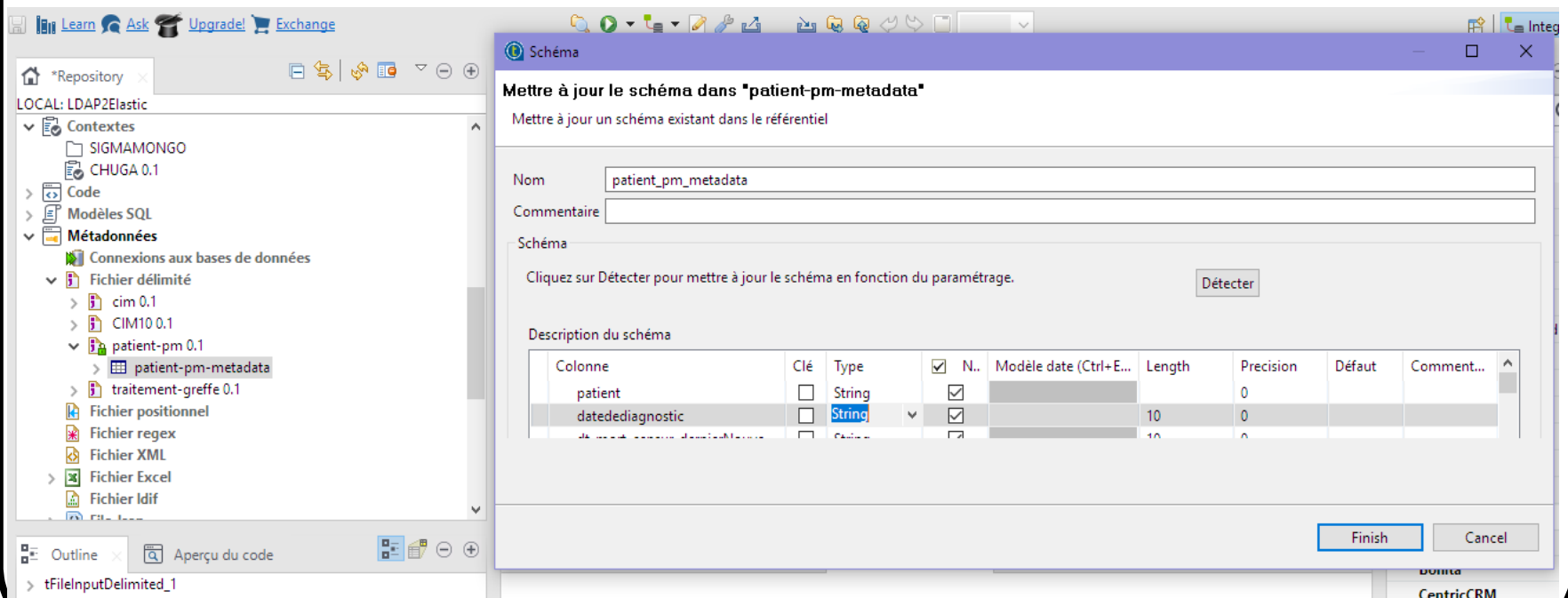
The interface includes a Repository pane on the left showing job designs, an Outline pane at the bottom left, a Palette on the right with Favorites and Applications Métier sections, and a bottom status bar with 'Designer' and 'Code' tabs and a 'tMap_1' component selected.

6 Décembre 2018

C. Cancé - C. Lenne ANF-MATHRICE

Paramétrage: Intérêt de l'usage des métadonnées

- Description réutilisable de la structure
- Propagation automatique des changements dans les jobs
=> adaptation quasi-automatique du code aux évolutions de l'alimentation.



Composant tmap : jointures, transformations ex:construction de liens

The screenshot displays the Talend Open Studio interface for configuring a tMap component. The main workspace shows three tables: 'row1', 'row2', and 'a_subi'. 'row1' has columns 'patient', 'greffe', and 'dt_greffe'. 'row2' has a primary key 'patient' and other columns. The 'Var' table defines two expressions: 'patient' + '/' + row1.patient (String, Variable checked) and 'greffe' + '/' + row2.patient (String, Variable checked). The 'a_subi' table shows the resulting columns: 'patient', '_from', and '_to'. The bottom section shows the 'Éditeur de Schéma' and 'Éditeur d'expression' tabs with detailed column metadata for 'row1' and 'a_subi'.

| Clé d'expr. | Column |
|--------------|---------|
| row1.patient | patient |

| Expression | Type | Variable |
|------------------------------|--------|---|
| "patient"+"/" + row1.patient | String | <input checked="" type="checkbox"/> _from |
| "greffe/" + row2.patient | String | <input checked="" type="checkbox"/> _to |

| Property | Value |
|--------------------------------|----------|
| Catch output reject | false |
| Catch lookup inner join rej... | false |
| Schema Type | Built-In |

| Expression | Column |
|--------------|---------|
| row1.patient | patient |
| Var_from | _from |
| Var_to | _to |

| Colonne | Clé | Type | <input checked="" type="checkbox"/> N.. | Modèle dat... | Length | Preci... | Déf... | Com... |
|-----------|-------------------------------------|---------|---|---------------|--------|----------|--------|--------|
| patient | <input checked="" type="checkbox"/> | Integer | <input checked="" type="checkbox"/> | | 2 | 0 | | |
| greffe | <input type="checkbox"/> | String | <input checked="" type="checkbox"/> | | 3 | 0 | | |
| dt_greffe | <input type="checkbox"/> | String | <input checked="" type="checkbox"/> | | 10 | 0 | | |

| Colonne | Clé | Type | <input checked="" type="checkbox"/> N.. | Modèle dat... | Length | Preci... | Déf... | Com... |
|---------|--------------------------|---------|---|---------------|--------|----------|--------|--------|
| patient | <input type="checkbox"/> | Integer | <input checked="" type="checkbox"/> | | 2 | 0 | | |
| _from | <input type="checkbox"/> | String | <input checked="" type="checkbox"/> | | | | | |
| _to | <input type="checkbox"/> | String | <input checked="" type="checkbox"/> | | | | | |

Composant tmap: exemple d'enrichissement depuis un référentiel (CIM10)

The screenshot displays the configuration of a tmap component. On the left, the 'row2' table is defined with columns 'patid' and 'diagid'. Below it, the 'row4' table properties are shown: Lookup Model (Charge une fois), Match Model (Correspondance unique), Join Model (Left Outer Join), and Store temp data (false). The central 'Var' table lists expressions and their types: 'row2.patid' (String), 'row2.diagid' (String), 'row4.code_cim' (String), and 'row4.lib_cim_long' (String). On the right, the 'diag' table is configured with columns 'patid', 'diagid', 'code_cim', and 'lib_cim_long'. The 'code_cim' column is marked as a primary key. The 'Auto map !' button is visible at the top right of the 'diag' configuration. At the bottom, two tables show the column definitions for 'row2' and 'diag'.

| Colonne | Clé | Type | <input checked="" type="checkbox"/> | N.. | Modèle date (...) | Length | Precision | Défaut | Comm... |
|---------|--------------------------|--------|-------------------------------------|-----|-------------------|--------|-----------|--------|---------|
| patid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |
| diagid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |

| Colonne | Clé | Type | <input checked="" type="checkbox"/> | N.. | Modèle date (...) | Length | Precision | Défaut | Comm... |
|--------------|-------------------------------------|--------|-------------------------------------|-----|-------------------|--------|-----------|--------|---------|
| patid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |
| diagid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |
| code_cim | <input checked="" type="checkbox"/> | String | <input checked="" type="checkbox"/> | | | 6 | 0 | | |
| lib_cim_long | <input type="checkbox"/> | String | <input checked="" type="checkbox"/> | | | 78 | 0 | | |

Composant tmap: exemple d'enrichissement depuis un référentiel (CIM10)

The screenshot displays the tmap component interface, which is used for data transformation and enrichment. It is divided into several panels:

- Left Panel (row2):** Shows the source data structure with columns 'patid' and 'diagid'. Below it, the 'row4' configuration is visible, showing join and match models.
- Center Panel (Var):** A table for defining variables used in expressions.
- Right Panel (diag):** Shows the target data structure with columns 'patid', 'diagid', 'code_cim', and 'lib_cim_long'. It includes an 'Auto map!' button.

The 'Var' table is currently empty:

| Expression | Type | Variable |
|------------|------|----------|
|------------|------|----------|

The 'diag' table shows the mapping of source expressions to target columns:

| Expression | Column |
|-------------------|--------------|
| row2.patid | patid |
| row2.diagid | diagid |
| row4.code_cim | code_cim |
| row4.lib_cim_long | lib_cim_long |

At the bottom, the 'Éditeur de Schéma' (Schema Editor) shows the detailed schema for 'row2' and 'diag':

| Colonne | Clé | Type | <input checked="" type="checkbox"/> | N.. | Modèle date (...) | Length | Precision | Défaut | Comm... |
|---------|--------------------------|--------|-------------------------------------|-----|-------------------|--------|-----------|--------|---------|
| patid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |
| diagid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |

| Colonne | Clé | Type | <input checked="" type="checkbox"/> | N.. | Modèle date (...) | Length | Precision | Défaut | Comm... |
|--------------|-------------------------------------|--------|-------------------------------------|-----|-------------------|--------|-----------|--------|---------|
| patid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |
| diagid | <input type="checkbox"/> | String | <input type="checkbox"/> | | | | | | |
| code_cim | <input checked="" type="checkbox"/> | String | <input checked="" type="checkbox"/> | | | 6 | 0 | | |
| lib_cim_long | <input type="checkbox"/> | String | <input checked="" type="checkbox"/> | | | 78 | 0 | | |

Organiser l'information

Conception d'un lac de données - Méthode

- Concevoir avec qui ? Les ingénieurs SI et experts métiers du domaine
- Pour produire quoi ? Identification des concepts pertinents, leurs propriétés, leurs relations.

Phase cognitive : reconnaissance des ensembles, éléments, propriétés, relations.
- Comment ? Interview experts métiers, modélisation collaborative par itération

Définition des éléments de la structure => thésaurus

Définir une représentation partagée de l'organisation de l'information

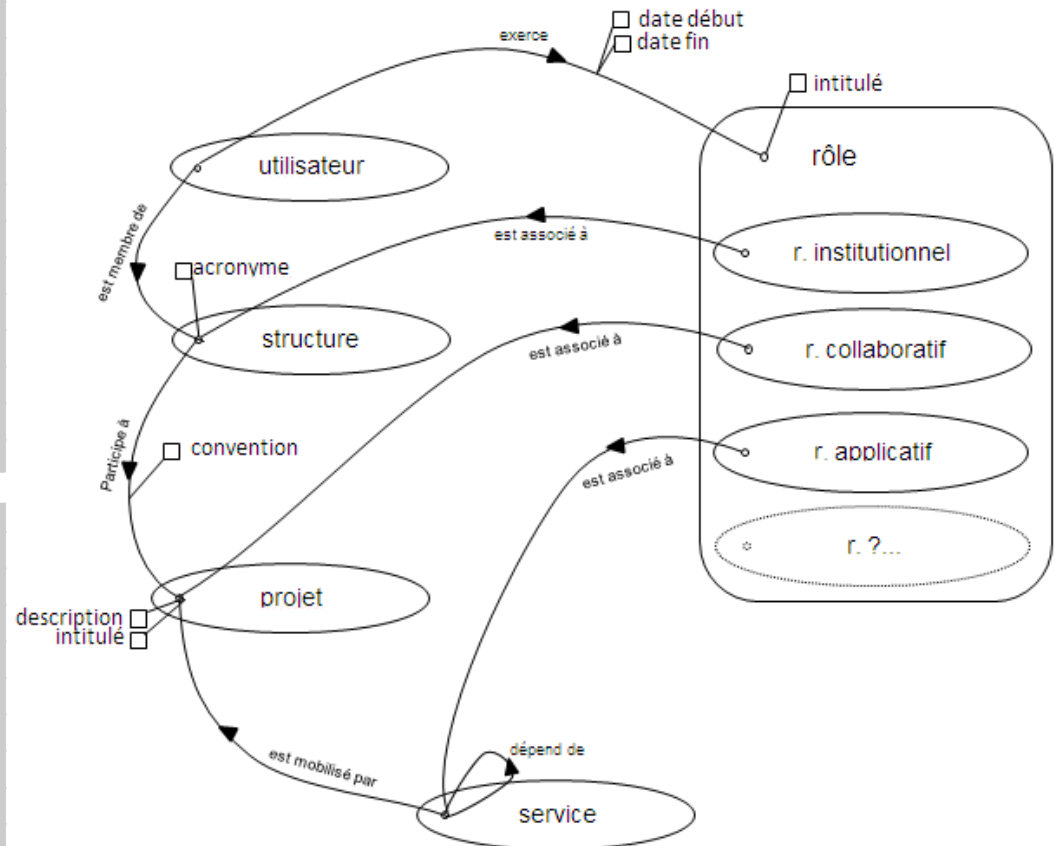
CHUGA : choix d'une méthode de type entité association fondée sur les hypergraphes

[Hypergraph Based Data Structure, F. Bouillé – 77]
- Vérification de la capacité de la structure à supporter les algorithmes nécessaires aux traitements des use-cases exprimés par les experts métiers.

Exemple de structuration « entité-association » de type graphe – (HBDS)

- Sommets (classes, d'objets) : concepts majeurs
- Arcs orientés, nommés : liens entre les concepts
- Attributs (de lien, classe, hyperclasse)

- Modèle évolutif
- Représentation propice au dialogue
- Rassemble les concepts proches



Du modèle à la base orientée graphe

- Similitude des concepts
- Objets sont représentés par des documents qui rassemblent les attributs sous formes de clefs-valeurs.
- **les réalisations (instances) des relations entre les objets sont matérialisées « physiquement »** par des liens, nommés, orientés et persistants.
=> pas d'évaluation de correspondance de clefs, pas de table intermédiaires pour les relations => **rapide.**
- Exploitation : requêtes SQL-LIKE spécifiques, fonctions de propagation dans un graphes, chemin le + court

Des foncteurs permettent de **naviguer par bonds** d'ensemble d'objets à d'autres à mesure que l'on se propage le long des arcs en appliquant d'éventuels filtres sur les propriétés et orientations.

http://cours-fad-public.ensg.eu/pluginfile.php/1270/mod_imscp/content/1/res/10_2_-_Les_foncteurs_de_base.pdf

Résoudre un problème peut revenir à parcourir un chemin dans une structure graphe.

• Intérêt, points d'attention des bases graphes (property graphs)

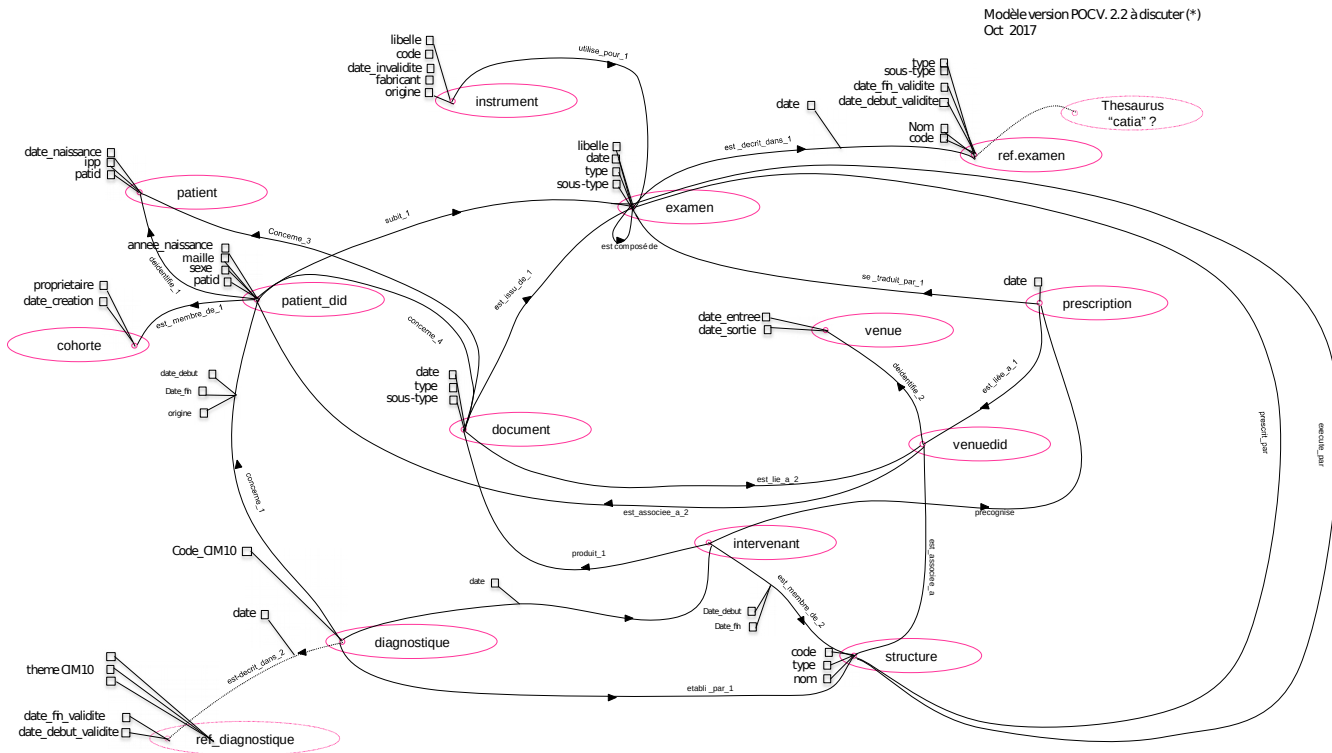
- Capacité à rassembler des concepts proches : adaptation à l'exploitation de structures souples, évolutives (possibilités de variantes dans les structures d'un même ensemble.
- Capacité à naviguer rapidement dans des organisations complexes de données massives fortement connectées.
- Intéropérabilité facilitée par les possibilités d'interconnexions entre les graphes notamment avec les référentiels.
- JSON : permet de prendre en compte des attributs complexes (listes, matrices,..) et la capacité à distribuer les traitements sur différents applicatifs au delà du clustering classique.
- Complémentarités entre les formes de bases orientées graphes et RDF triplestore. Maturité des langages et outils différentes ex : SPARQL (standardisé) vs GREMLIN et d'autres
- Warnings :

Fragilité de l'intégrité référentielle => bases graphe davantage adaptée à l'observation qu'à la gestion des données. Contrôles nécessaires (ex : dédoublonnage, topologie...)
Implémentation de l'héritage : jouer sur la souplesse ou relier les sous ensembles ?
Distinguer le graphe du modèle (structure du phénomène), du graphe des données (réalisations).

Fonctionnalités cibles nécessaires à la plate-forme

- Compatibilité d'échange web services **HTTP REST JSON**
- intégrer une **API customisable** permettant de faire abstraction des spécificité du langage de requête interne, fréquemment spécifique.
=> développer un moyen graphique d'interroger le lac.
- Capacité à interroger de l'ordre de $x.10^7$ à $x.10^8$ objets et leurs relations de façon interactive.

Du modèle conceptuel au POC : choix d'implémentation



Réunion des objets dans les ensembles du plus haut niveau du modèle.
Distinction des classes (sous ensembles) avec un attribut commun « Type »
=> simplification de la structure, généricité nécessaire du code, évolutivité

Solutions open source envisagées

- Neo4J : <https://neo4j.com>

+ lisibilité du langage de requête « cypher » : <https://www.opencypher.org/>

+ Compatibilité Apache's TinkerPop API

-structuration de l'information ?

- coûts potentiels de la montée en charge

```
MATCH (me:User),(me)-[rating:RATED]->(movie)
WHERE me.name = 'Me'
RETURN movie.title, rating.stars, rating.comment;
```

- Orientdb : <https://orientdb.com/>

base document multi-modèles (document, graphe, clé/valeur,,) , sous licence Apache

+ Compatibilité Apache's TinkerPop API

+ clustering, sharding

+scalabilité

- documentation API, personnalisation

- Arangodb : <https://www.arangodb.com/>

base multimodèle (document, graphe, clé/valeur), JSON natif (nested any depht, arrays) , sous licence Apache

+ simplicité de la structure et lisibilité des informations

+ API HTTP REST JSON personnalisable microservices foxx

+ documentation+++

+ AQL puissant mais spécifique

+ GraphQL (facebook 2015)

Focus sur ArangoDB

- Base orientée graphe « in memory » clusterisable (sharding)
- Base graphe, clef-valeur, document (les nœuds et arcs du graphe peuvent avoir des attributs)
- Format natif JSON
- Unités d'indexation sous forme de collections (sommets ou arcs)

- Accès depuis le service HTTP intégré <http://localhost:8529>

depuis la console : `unix> arangosh`

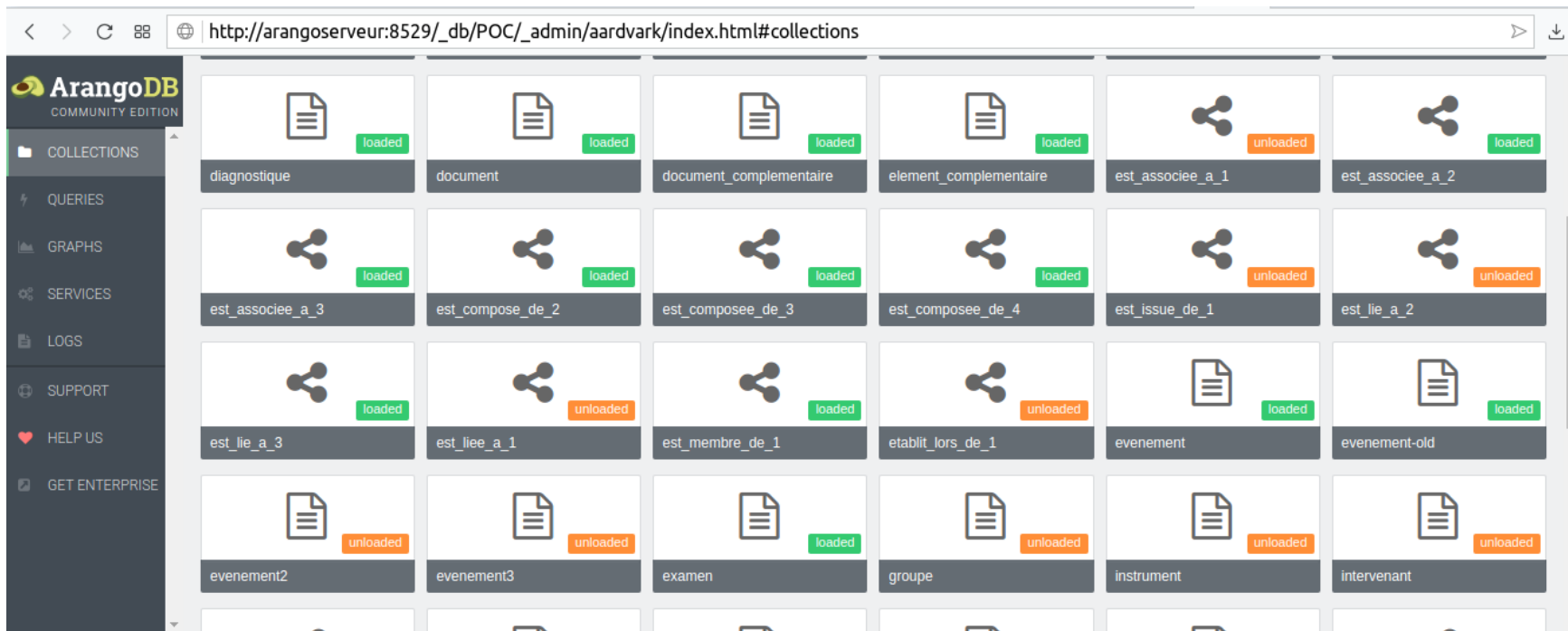
shell de commandes javascript

- <https://www.arangodb.com/documentation/>

ArangoDB - structure graphe des données

- Collections rassemblent les objets de types : « Document » (Sommet)
« Edge » (arc)

Un index par collection



The screenshot shows the ArangoDB web interface with a grid of collections. The left sidebar contains navigation options: COLLECTIONS, QUERIES, GRAPHS, SERVICES, LOGS, SUPPORT, HELP US, and GET ENTERPRISE. The main area displays a grid of collection cards, each with an icon representing its type (document or edge) and a status indicator (loaded or unloaded).

| Collection Name | Type | Status |
|-------------------------|----------|----------|
| diagnostique | Document | loaded |
| document | Document | loaded |
| document_complementaire | Document | loaded |
| element_complementaire | Document | loaded |
| est_associee_a_1 | Edge | unloaded |
| est_associee_a_2 | Edge | loaded |
| est_associee_a_3 | Edge | loaded |
| est_compose_de_2 | Edge | loaded |
| est_composee_de_3 | Edge | loaded |
| est_composee_de_4 | Edge | loaded |
| est_issue_de_1 | Edge | unloaded |
| est_lie_a_2 | Edge | unloaded |
| est_lie_a_3 | Edge | loaded |
| est_liee_a_1 | Edge | unloaded |
| est_membre_de_1 | Edge | loaded |
| etablit_lors_de_1 | Edge | unloaded |
| evenement | Document | loaded |
| evenement-old | Document | loaded |
| evenement2 | Document | unloaded |
| evenement3 | Document | unloaded |
| examen | Document | loaded |
| groupe | Document | unloaded |
| instrument | Document | unloaded |
| intervenant | Document | unloaded |

Contenu d'une collection : documents JSON

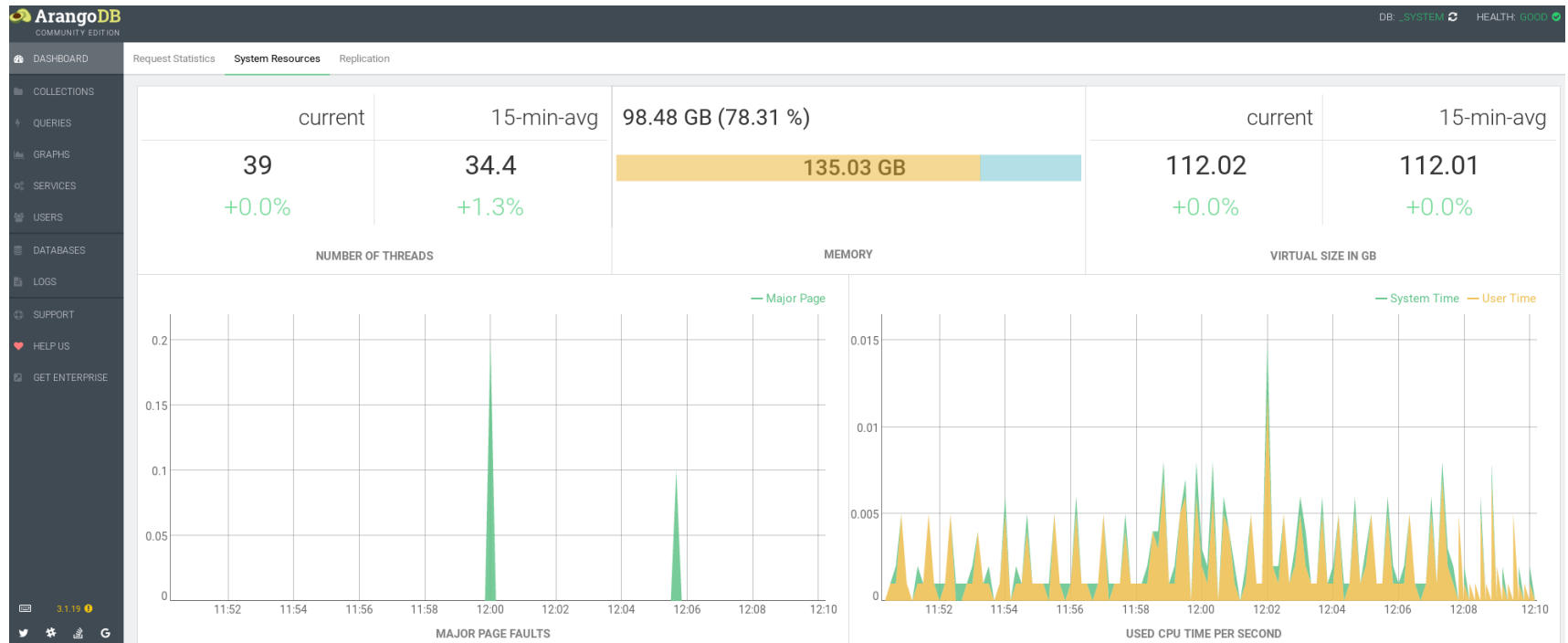
The screenshot shows the ArangoDB interface for a collection named 'est_composee_de_3'. The main view displays a table of documents with columns for 'Content' and '_key'. A filter is applied: 'age >= 20'. The table shows several documents, each with a unique '_key' and a JSON document structure. A modal window is open, showing a detailed view of a document with fields like 'date_creation', 'description', 'format', and 'hdfs_path'.

| Content | _key |
|--|------------|
| { "_from": "trajectoire/1006739C92", "_to": "evenement/Z88", "age": "20.60035842293907", "context": "C92", "date": "2014-08-07T08:00:00", "... | 4116069691 |
| { "_from": "trajectoire/932172C92", "_to": "evenement/A41", "age": "82.98127053464755", "context": "C92", "date": "2014-12-25T00:47:00", "d... | 4116124997 |
| { "_from": "trajectoire/771422C92", "_to": "evenement/C92", "age": "6 | |
| { "_from": "trajectoire/1571108C92", "_to": "evenement/K81", "age": "7 | |
| { "_from": "trajectoire/823812C92", "_to": "evenement/C92", "age": "7 | |
| { "_from": "trajectoire/1611601C92", "_to": "evenement/Z94", "age": "7 | |
| { "_from": "trajectoire/1006739C92", "_to": "evenement/C92", "age": "7 | |
| { "_from": "trajectoire/771422C92", "_to": "evenement/Z94", "age": "6 | |
| { "_from": "trajectoire/775873C92", "_to": "evenement/Z04", "age": "6 | |
| { "_from": "trajectoire/1505951C92", "_to": "evenement/C92", "age": "6 | |

7,603 edge(s)

10,166,510 doc(s)

Monitoring



Document JSON d'une collection Arangodb

- Formulaire d'un edge, modifiable, sauf les attributs built-in (préfixés par « _ »)

```
_id: est_composee_de_3/4116114473
_rev: _Xg4QBFu---   _from: trajectoire/1571108C92
_key: 4116114473   _to: evenement/K81
```

☰ ☰ Code ▾

```
1 {
2   "type": "clinique",
3   "context": "C92",
4   "tagval": "cim_cod3",
5   "val": "K81",
6   "unit": "unit",
7   "date": "2014-08-26T09:16:00",
8   "daterel": "115.75069444444445",
9   "age": "60.65157556750299"
10 }
```

- `_rev` : versionning de document utilisé par la base
 - => Choix des documents à répliquer entre des serveurs.
 - => Fournir au client la dernière version d'un document à modifier
- Amélioration : prévoir un intitulé dans le sens inverse du lien

ArangoDB Query Language : AQL

- Langage propriétaire comparable à SQL, avec des fonctions supplémentaires de traitement des graphes <https://docs.arangodb.com/3.3/AQL/>
Utilisation depuis l'interface web Arangodb, le dhell arangosh, l'API HTTP ou encore les microservices Foxx
- Jeu d'instructions pour les Requêtes « SQL-LIKE » <https://www.arangodb.com/why-arangodb/sql-aql-comparison/>

FOR: Iterate over all elements of an array.

RETURN: Produce the result of a query.

FILTER: Restrict the results to filtered elements

SORT: Force a sort of the array

LIMIT: Reduce the number of elements in the result

LET: Assign an arbitrary value to a variable.

COLLECT: Group an array by one or multiple group criteria.

REMOVE: Remove documents from a collection.

UPDATE: Partially update documents in a collection.

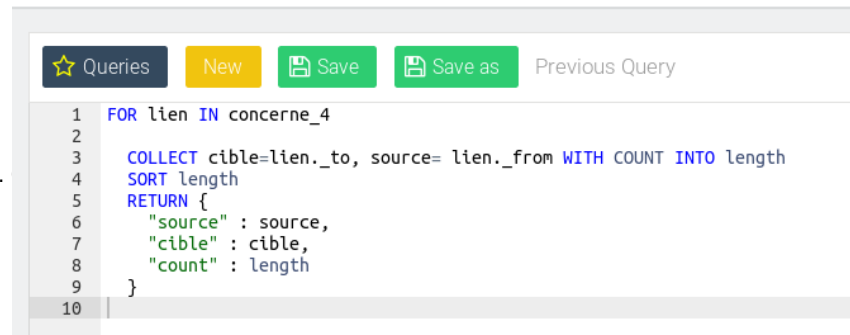
REPLACE: Completely replace documents in a collection.

INSERT: Insert new documents into a collection.

UPSERT: Update/replace an existing document, or create it in the case it does not exist.

WITH: Specify collections used in a query (at query begin only).

Query: detect_doublon_link



```
1 FOR lien IN concerne_4
2
3 COLLECT cible=lien._to, source= lien._from WITH COUNT INTO length
4 SORT length
5 RETURN {
6   "source" : source,
7   "cible" : cible,
8   "count" : length
9 }
10
```

- Fonctions courantes chaînes, dates...

ArangoDB Query Language : AQL

- Liste des objets connectés à un objet de la collection «cohorte», selon les arcs contenus dans la collection « est_membre_de » (ici les membres d'un groupe de patients)

```
FOR x IN ANY @cohorte est_membre_de_1 OPTIONS {bfs: true, uniqueVertices: 'global'} RETURN x
```

- Requête de traversée de graphe multi-chemin depuis un objet (patient) calculant le nombre d'objets de chaque classe à une distance de 1 arc

```
FOR v IN 1 ANY 'patient_did/501959' GRAPH "patient_did360"
```

```
COLLECT classe=LEFT(v._id, FIND_FIRST(v._id, '/')) WITH COUNT INTO length
```

```
RETURN {
```

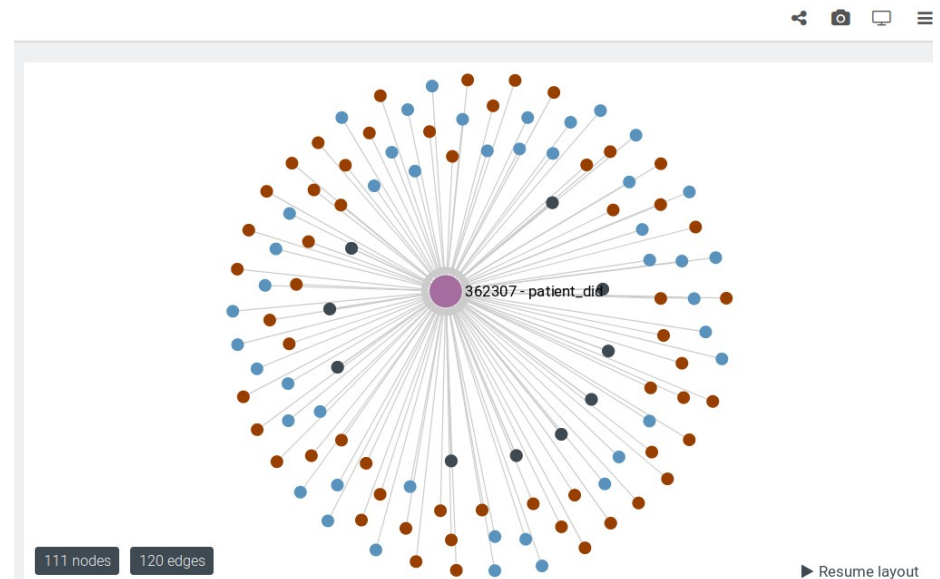
- "classe" : classe,

```
"count" : length
```

réponse en JSON:

```
[[  
  {"classe": "diagnostique",  
   "count": 148  
  },  
  {"  
   "classe": "document",  
   "count": 80  
  },  
  {"  
   "classe": "venue_did",  
   "count": 60  
  }  
]]
```

réponse graphique :



Focus sur ArangoDB - Intégration

- intégration - option 1 : Utiliser l'API HTTP standard dans le code
curl --data @- -X POST --dump - http://localhost:8529/_api/cursor

```
{ "query" : "FOR u IN users LIMIT 2 RETURN u", "count" : true, "batchSize" : 2
```

<https://docs.arangodb.com/3.3/HTTP/AqlQueryCursor/AccessingCursors.html>

- **Option 2 : Définir sa propre API pour faire abstraction de l'AQL dans les applications clientes.**

Solution choisie pour implémenter un langage de navigation dans le graphe utilisable par utilisateurs métiers (graphique et cypher-like)

ArangoDB - Interface de développement Foxx

- API Foxx coté server HTTP REST JSON , accès à la data « in-memory »

cloisonnement par contexte

JS8 multi-threads

- Exemple

```
router.get('/patdidsfromcohort', function (req, res) {
```

```
  const stmt = db._createStatement({ "query": "FOR x IN ANY @cohort est_membre_de_1 OPTIONS {bfs: true, uniqueVertices: 'global'} RETURN DOCUMENT(x._id)" });
```

```
  stmt.bind('cohort', req.param('cohort'));
```

```
  const c = stmt.execute();
```

```
  res.json(c.toArray());
```

```
  })
```

```
  .summary("returns patdids who are member of cohort identifier parameter");
```

- Apport :

encapsulation de l'API AQL => pas de diffusion dans l'écosystème applicatif Python, NodeJS, JS

robustesse aux évolutions (un changement de base graphe par exemple)

peu de code à maintenir

Tests de l'API



```
ent/diagnosticseventsdatesfrompatdid?patdid=patient_did 120 %  
Les plus visités BIPER Groupes Pictionary gratuit en li... EOSC-Pillar - PartB-Se... RNR V01 : Cadre de p...  
JSON Données brutes En-têtes  
Enregistrer Copier Tout réduire Tout développer Filtre  
▼ 0:  
  diag_id: "diagnostique/4596034"  
  cim_cod3: "Z90"  
  date: "2014-02-27T09:30:00"  
  age: 63.157897834274955  
▼ 1:  
  diag_id: "diagnostique/4596033"  
  cim_cod3: "Z51"  
  date: "2014-02-27T09:30:00"  
  age: 63.157897834274955  
▼ 2:  
  diag_id: "diagnostique/4596032"
```

IHM

Apport des SIG et du Webmapping

- Intégration d' Outils de Web mapping (Open Layer) pour générer la souche de l'interface interactive de navigation dans le Lac à partir de sa méta-structure définie dans QGIS (classe des classes, classe des liens, attributs associés).
- Même solution pour générer la carte de répartition de cohorte à la volée.
- Economie de code

V. IHM : Navigation interactive dans le lac

Parcourir le lac

Démarrer diagnostique[cim_libelle_court=DIABETE]-concerne_1[*]->patient_id[Naviguer diagnostiqu 94539 Dashboard Map

Modèle POC CIUGA V. 2.5 à discuter (*)

- lien
- classe
- POC_graph_model-2.6_c

name: patient_id
annee_naissance
maille
sexe

patient

instrument

examen

document

intervention

structure

venue

prescription

venueid

traitement

diagnostique

ref_diagnostique

cohortes

groupe

element

date_naissance
ipp
patid

libelle
code
date_invaldite
fabricant
origine

libelle
date
type
sous-type

date
type
sous-type
date_fin_valdite
date_debut_valdite
Nom
code

date_entree
date_sortie

date
type
sous-type

date_debut
date_fin

code_cim10
cim_10_libelle_court

code
type
nom

date_debut
date_fin

date
type
sous-type

date_debut
date_fin

theme CIM10
date_fin_valdite
date_debut_valdite

date_naissance
annee_naissance
maille
sexe
patid

titre
auteur
date_creation

date_debut
date_fin
probabilite

date

Map data © OpenV

Sauvegarde de la

Entrez le titre

Entrez les caractéristiques

Enregistrer Chemin diagnostique[cim_libelle_court=

L'API développée autorise l'interactivité de l'utilisateur

The screenshot shows a REST API documentation interface for a service named 'thingsfrompatient'. The interface includes a top navigation bar with 'Info', 'API', 'Readme', and 'Settings' tabs. The 'API' tab is active, displaying a list of endpoints. Each endpoint is represented by a blue button with the HTTP method 'GET' and a description of the endpoint's function. The endpoints listed are:

- `/patdidfromdocument`: returns patdid concerned by document identifier parameter
- `/venuesfrompatdid`: returns structures informations labels and dates provided by venues associated to patdid identifier parameter
- `/structuresfrompatdidvenues`: returns structures informations labels and dates provided by venues associated via venue_id to patdid identifier parameter
- `/selectioncount`: returns structures informations labels and dates provided by venues associated via venue_id to patdid identifier parameter
- `/filteredobjectsfromclasse`: returns selection of grouped objects
- `/filteredobjectsfromselection`: returns new selection of filtered objects
- `/linkedobjectsfromselection`: returns selection of namedlink objects connected to selection
- `/selectedobjects`: returns selected objects (single list)
- `/firstdiagnosticdatefrompatdid`: returns first specified diagnostic event date
- `/diagnosticseventsdatesfrompatdid`: returns chronological diagnostic list events dates and patient age from patient_id
- `/diagnosticseventsrelativedatesfrompatdid`: returns chronological diagnostic list events relative dates and patient age from patient_id
- `/clearsaveselection`: clear selection saved in lastselection
- `/clearselections`: clear selections, current and last

Calcul des effectifs par maille de l'index spatial et cartographie à la volée (souche Openlayer)

- // Foxx map selected patients to spatial mesh (appel depuis l'IHM)

```
router.get('/mapselectedpatients2mesh', function (req, res) {
```

```
  const stmt = db._createStatement({ "query": "FOR x IN ANY @groupesel  
  regroupe OPTIONS {bfs: true, uniqueVertices: 'global'} COLLECT  
  maille=x.maille WITH COUNT INTO effmaille RETURN {\"maille\":  
  maille, \"effectif\": effmaille}" });
```

```
  stmt.bind('groupesel', 'groupe/selection');
```

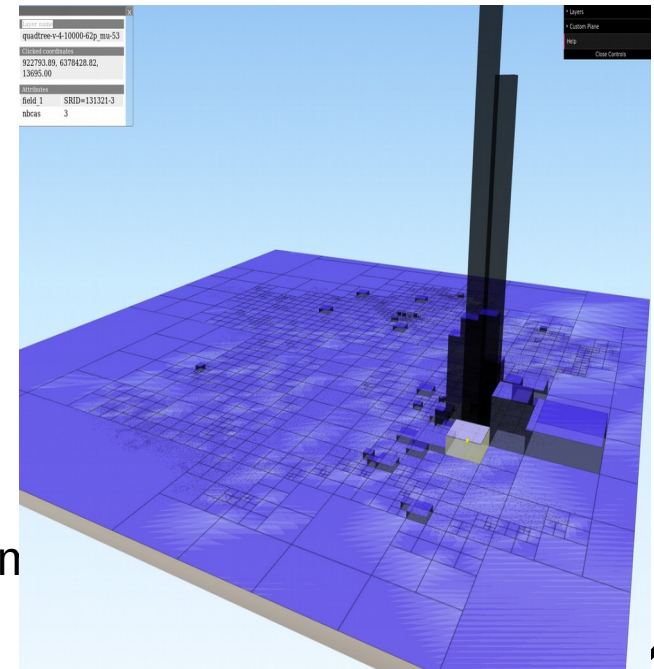
```
  const c = stmt.execute();
```

```
  res.json(c.toArray());
```

```
  // fin mapping
```

```
  })
```

- .summary("map selected patients to spatial n



Exemples d'échanges avec d'autres applications

- Echanges avec les briques applicatives de traitement

usage de web services HTTP REST JSON

- ETL : ex composant talent « trest »

```
curl -X POST --data-binary @- --dump - http://datalaketest-11.imag.fr:8529/_api/document/diagnostic <<EOF
{
  "ven_id" : "xxxx4564",
  "pat_id" : "yyyy1684",
  "type_diagnostic" : "DIAGNOSTIC PRINCIPAL",
  "cim_cod" : "zz.52"
}
```

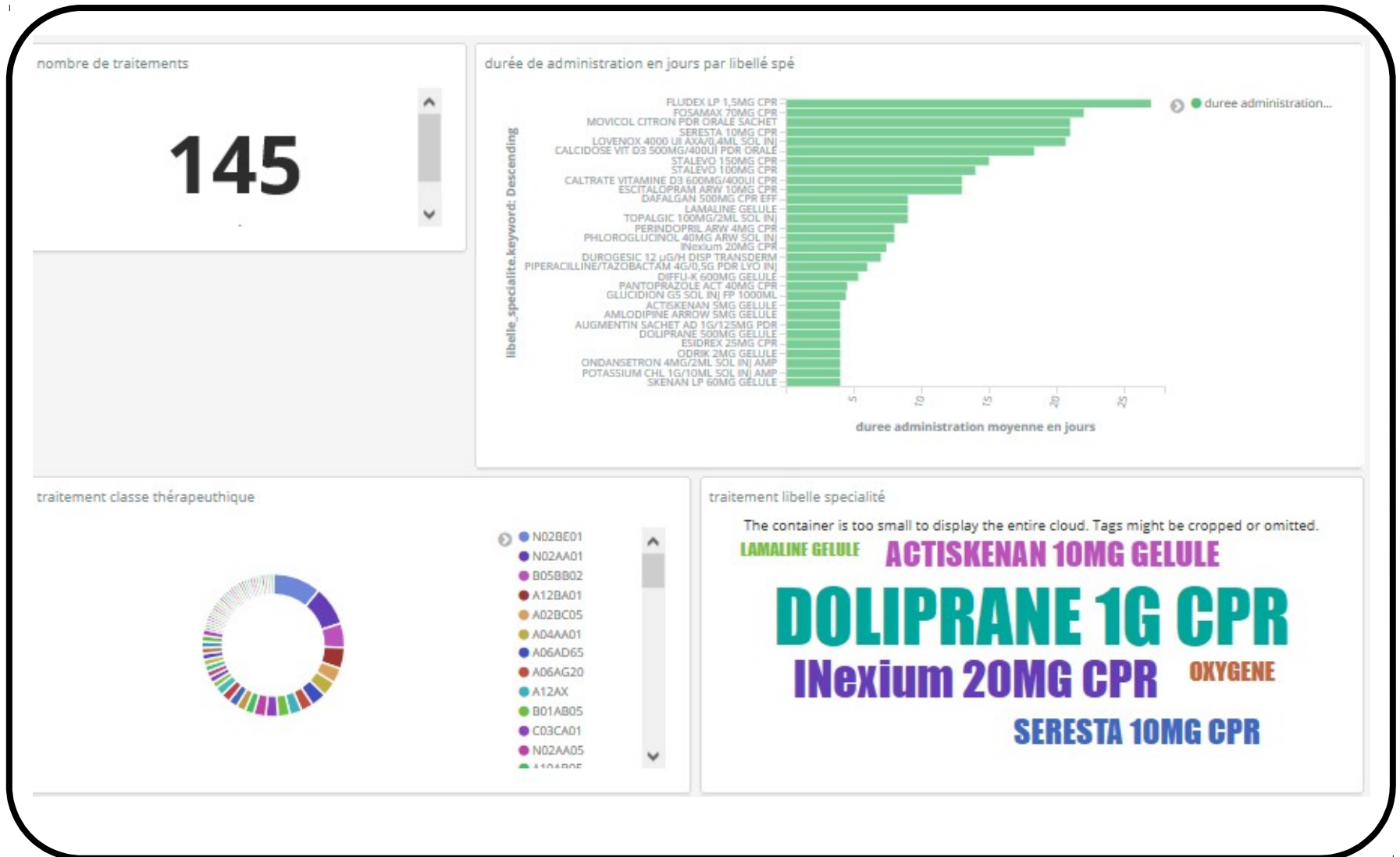
- Moteur de recherche type Elastic : moteur de recherche, dataviz

```
var url = "http://elastic-13.imag.fr:9200/arango-"+jump.initialclassname+"/"+jump.initialclassname
req1.open("POST", url,false);
req1.setRequestHeader("Access-Control-Allow-Origin","*");
req1.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
req1.setRequestHeader("X-Requested-With","XMLHttpRequest");
req1.send(objtpost);
```

- Environnement Spark - Hadoop ex : classification

http://160.8.x,x:8529/.../thingsfrompatient/addobjectingroup/?group=groupe/6385&object=patient_did/599571

VI. Projection à la volée dans la pile Elastic visualisation Kibana



Points d'attention sur Elastic

- Mapping : définition du types des champs contenus dans un indice
(Préalable au chargement de l'index)

<https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html>

- Les agrégations

<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html>

- Bool query (must, should, must not...), leur impact sur le score des documents retournés:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html>

Elaboration de données métier

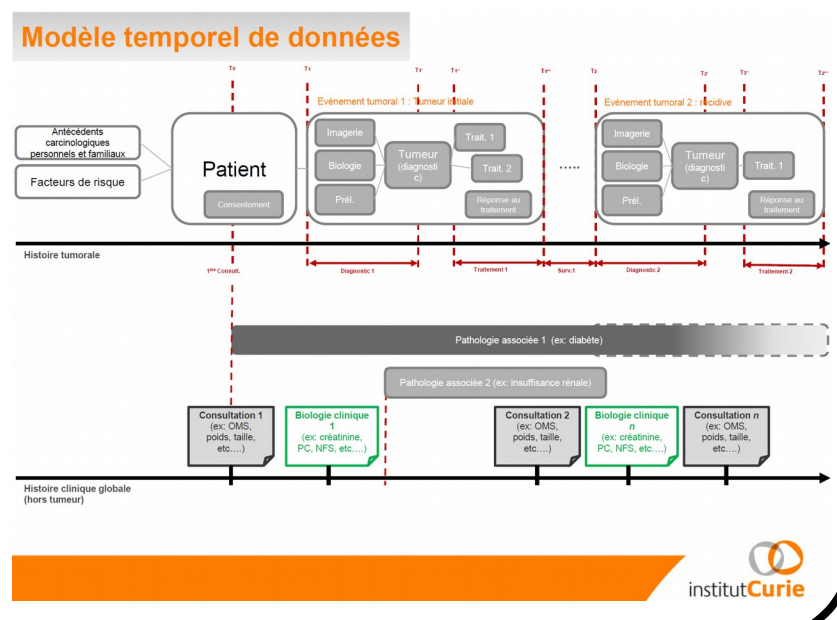
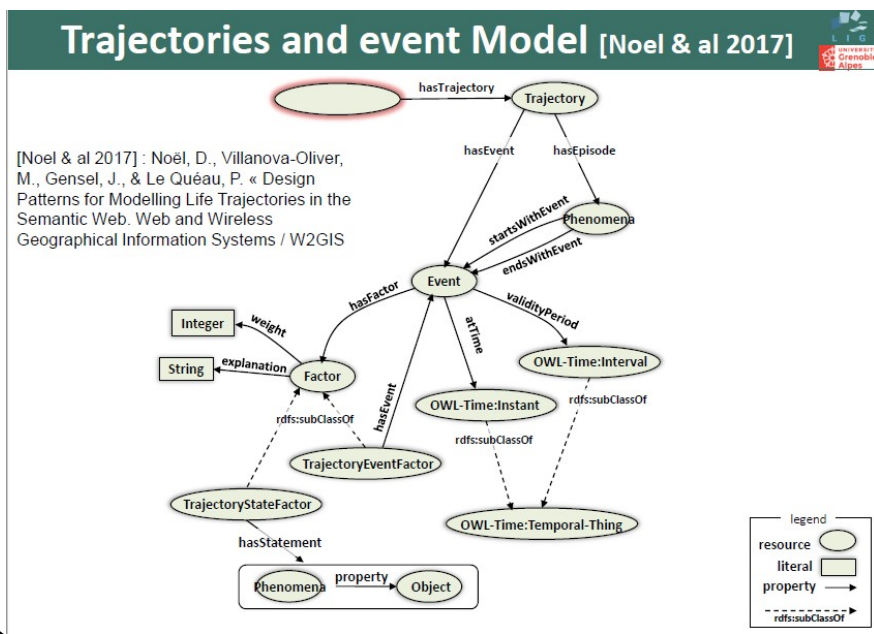
VII. Construction de données métiers élaborées :trajectoires

- Modélisation sous ArangoDB de « **trajectoires** » composées d'**événements** et de **phases**. Concept transversal, en cours d'exploration avec le CHUGA-IAB.

A un patient est associé à une trajectoire d'un point de vue clinique.

Une trajectoire clinique est composée d'évènements cliniques (d'autres pourraient être pharmacologique, biologique, ou entrelacer des évènements de différents types) et d'éventuelles phases identifiables par les experts métiers.

- références



Elaboration de trajectoires cliniques dans la base graphe

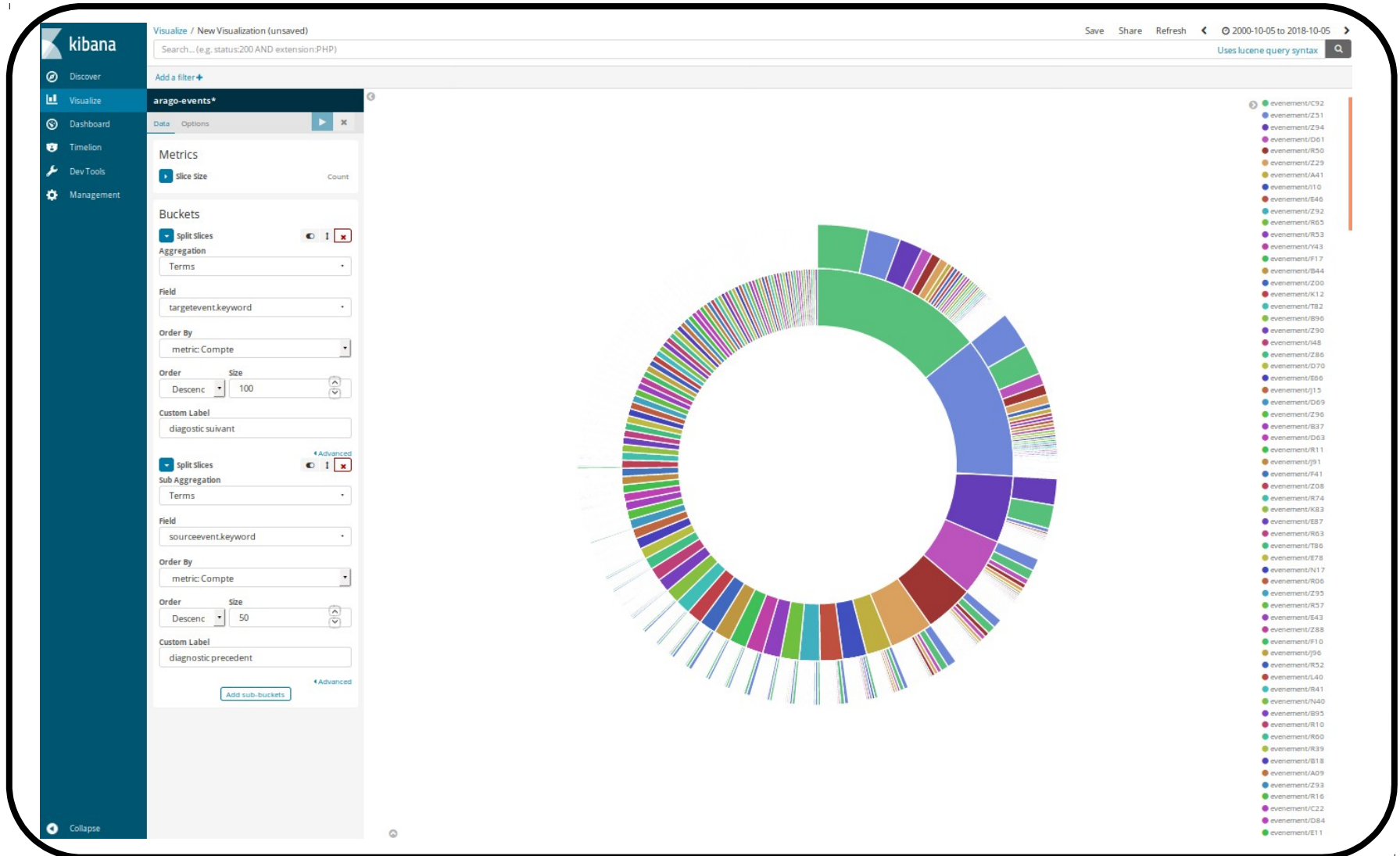
- Développement Python + API foxx
- Phase 1 : Générer 1 trajectoire par patient et la relier aux évènements que l'on peut générer à partir de l'existant clinique dans du lac.
- Phase 2 : Générer des arcs dans une collection « précède » reliant chaque évènement d'une même trajectoire à son successeur → 80 000 Arcs générés. Quel outil de représentation ?
- Projection du résultat vers la pile Elastic

```
// python
```

```
def insertEventsLinkInElastic (sindex, svaleur) :
```

- ```
url="https://simu:9200/"+sindex
header = {"Access-Control-Allow-Origin" : "*", "X-Requested-With" : "XMLHttpRequest", "Content-Type" :
"application/json"}
print (svaleur)
params=svaleur
req = requests.post (url, headers=header, data=params,auth=auth,verify=False)
return
```

Résultat : agrégations concentriques sur les arcs « précède » sur l'évènement suivant (sommet final de l'arc) - anneau intérieur - puis sur l'évènement précédent (sommet initial de l'arc) - anneau extérieur



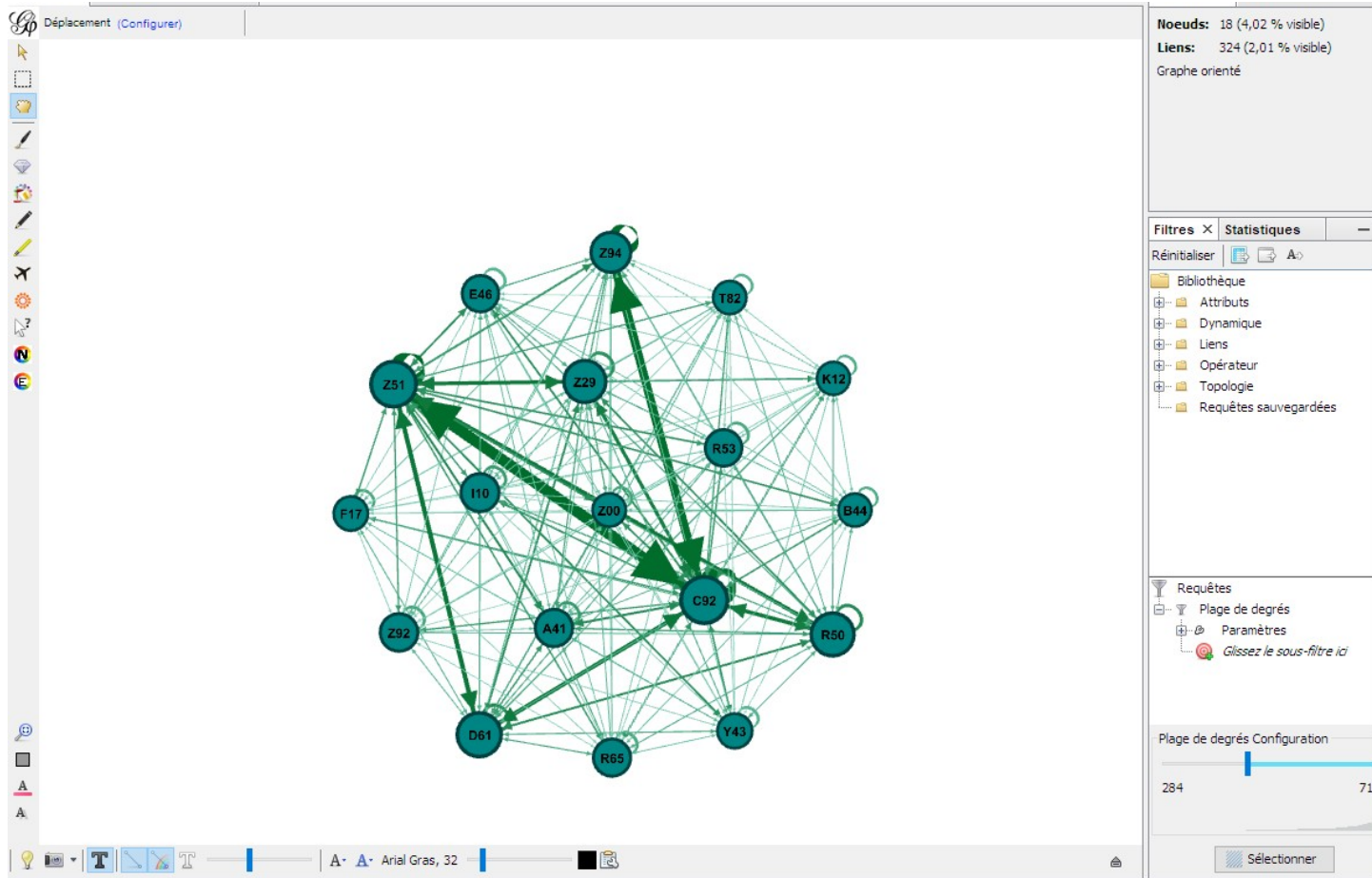
# Visualisation et analyse de graphe première approche





# Agrégation des arcs reliant les mêmes évènements

- Représentation des relations fréquentes de précédences entre des évènements



# Annexes

# Infrastructures supportant les bases graphes

- Ecosystème applicatif « in memory » clusterisable

Spark – GraphX <https://spark.apache.org/graphx/>

Arangodb, Orientdb, Neo4j,....

- Langages de requête : non standards

exemples :

- cypher (Neo4J) : orienté graphes, optimisé pour la lisibilité

- gremlin (spark.apache Tinkerpop...):

```
g.V().as("a").out("knows").as("b").
```

```
select("a","b").
```

```
by("name").
```

```
by("age")
```

AQL (arangodb):

```
FOR x IN ANY "patient_did/14xyz14" concerne OPTIONS {bfs: true, uniqueVertices: 'global'} RETURN DOCUMENT(x._id)
```

- Initiatives de standardisation, intégration : graph QL facebook

« propose une alternative aux REST API sur les graphes dont le stockage est éventuellement distribué. Il propose au client de formuler la structure de données dans la requête, alors cette même structure est retournée par le serveur. »

<https://www.arangodb.com/2016/02/using-graphql-nosql-database-arangodb/>

<https://fr.wikipedia.org/wiki/GraphQL>

Gremlin: apache2 tinkerpop framework <https://github.com/tinkerpop/gremlin/wiki/SPARQL-vs.-Gremlin> , <http://tinkerpop.apache.org/> ,

<http://tinkerpop.apache.org/gremlin.html>

- Exemples de Triple stores Jena fuseki, Virtuoso, allegrograph... usage GAFA
- Infrastructures mixtes naissantes orientées graphes et RDF

# Bibliographie

- [10] Théorie des graphes, Stéphane Pelle ENSG  
[http://cours-fad-public.ensg.eu/pluginfile.php/1525/mod\\_resource/content/1/Theorie\\_des\\_graphes.pdf](http://cours-fad-public.ensg.eu/pluginfile.php/1525/mod_resource/content/1/Theorie_des_graphes.pdf)
- [11] Éléments de théorie des graphes - Alain Bretto, Alain Faisant, François Hennecart
- [12] François Bouillé. Le modèle HBDS. ENSG 2013  
<http://cours-fad-public.ensg.eu/mod/imscp/view.php?id=254>
- [13] Pelle Stéphane. Documentation HBDS et UML 2006.<http://pelle.stephane.free.fr/>
- [14] Qwant et le machine learning, JRES 2017 - Sylvain Peyronnet  
<https://www.jres.org/fr/videotheque?mode=replay&id=189&resolution=360>
- [15] BERGE C, "Graphes et Hypergraphes", Edition Dunod, 1970
- [16] Ph. GENOUD, Web des données: *Les Principes-Les Standards du W3C* –Journée Interopérabilité et Innovation –IGN-BRGM-OGC -7 Octobre 2014 -Paris

# Rappel Workflow de la donnée : intégration de l'existant, élaboration d'information métier, forage - exploration

