Auffray Baptiste
Boutin Loïc
Dromigny--Chevreuil Ivan

# Rapport de projet

## Web des données / Web sémantique

## 2017-2018

## Table des matières

# I- Introduction

Ce projet avait pour objectif de transformer les données ouvertes de l'enseignement supérieur de la Recherche et de l'Innovation (https://data.esr.gouv.fr/FR/) en données sémantiques. En ce qui nous concerne nous avons choisi le dataset suivant : https://data.enseignementsup-recherche.gouv.fr/explore/dataset/fr-esr-rd-moyens-entreprises-intensite/table/?sort=valeur.
Celui-ci concerne les moyens consacrés, qu'ils soient financiers ou humains à la Recherche et au Développement dans les entreprises par zone géographique, année, intensité technologique et type de personnel.
Ce projet est composé de 5 étapes et celles-ci seront expliquées dans les différentes parties qui suivent. Nous terminerons avec une conclusion sur le travail fourni, les difficultés rencontrées et les éventuelles améliorations possibles.
Comparé au 4-5 pages demandé ce rapport peut paraître beaucoup plus long, mais celui-ci contient beaucoup d'annexes, car nous voulions pouvoir illustrer de manière efficace le travail que nous avons fourni.
La totalité de notre travail est également disponible sur GitHub : https://github.com/Yaronn44/WebSem-WebData

# II- Transformation des données

La première étape était de choisir un ensemble de données ouvertes et de les transformer en RDF en utilisant si possible des vocabulaires partagés. Il fallait également proposer deux requêtes SPARQL sur les données une fois celles-ci transformées en RDF.
Ainsi afin de transformer nos données, nous avons utilisé un des logiciels recommandés sur le sujet : Tarql. Celui-ci fut simple d'installation et d'utilisation et nous transformons nos données au moyen de la commande suivante :

```
~/Application/tarql-master/target/appassembler/bin/tarql -e utf-8 -d ";" Construct/construct.sparql > Data/project.ttl
```

l'option `-e` permet de définir l'encodage tandis que `-d` permet de définir les délimiteurs du fichier en entrée.
Pour visualiser la différence entre les données avant et après leur transformation en RDF, voir l'Annexe 1 et l'Annexe 2

Pour représenter au mieux nos données nous avons cherché un vocabulaire adapté et grâce à l'utilisation du site "lov.okfn.org" nous avons découvert "frapo". Frapo est une ontologie permettant de représenter différents types de budgets, donations et autres dons pécuniaires mais également définir l'administration des recherches et des projets.
Nous utilisons aussi l'ontologie de dbpedia pour représenter les zones géographiques par région, ainsi que les namespaces "rdf" pour définir les types de nos données ou encore "owl" pour lier nos données à certains datasets déjà existants. De même, nous avons créés un namespace "ns" pour créer nos propres types et prédicats lorsqu' aucun autre type de prédicat ne semblait convenir.
Nous utilisons la classe frapo:Funding pour définir le type de nos atomes. Ceux-ci vont correspondre à une URI que nous construisons à partir de la chaîne de caractère suivante `http://ex.org/fr-esr-rd-moyens-entreprises-intensite/` à laquelle nous concaténons le code de la zone géographique, l'année, le code de l'indicateur, le code du type de personnel s'il existe, et le code de l'intensité.
Cette URI va également être composé de plusieurs attributs. L'un d'eux va être de type dbo:Place, et va être constitué de deux autres attributs : une zone géographique de type frapo:hasLocation et d'une URI que nous construisons à partir de la chaîne de caractère suivante `http://ex.org/fr-esr-rd-moyens-entreprises-intensite/codeRegion/` à laquelle nous concaténons le code de la région concerné par les moyens consacrés. Cette dernière URI nous sera utile plus tard pour la liaison avec d'autres datasets (ref. III- Liaison des données).

Pour tester si notre fichier RDF était bien construit et cohérent, ainsi que pour tester nos requêtes SPARQL nous avons utilisé Fuseki. Ce dernier est un serveur SPARQL, que l'on lance sur notre

machine. Nous y accédons donc en local et une fois notre RDF uploadé dessus nous avons pu tester nos requêtes (voir Annexe 3, Annexe 4, Annexe 5 et Annexe 6).
Nous avons également créé une multitude de requêtes basiques pour récupérer tous les attributs distincts un à un afin de vérifier que le fichier de données de base ne contenait pas de redondances pouvant provenir de fautes de frappe ou autre. Cela nous a permis de corriger à la main notre fichier .csv d'origine et ainsi de normaliser nos données.

# III- Liaison des données

La seconde étape était de lier nos données aux données d'autres groupes, et de proposer une requête sur cet ensemble de données.
Plusieurs groupes avaient des datasets pouvant potentiellement être liés au nôtre, ceux-ci se référant à des régions administratives françaises. C'est donc au moyen de ce dernier attribut que nous avons décidé de lier nos datasets ensemble. Nous avons donc pour cela créé une URI contenant le code des régions, nous permettant ainsi au moyen d'un owl:sameAs de lier nos données à celles d'autres groupe ayant construit des URI de la même manière dans leurs datasets.
Nos données ont été lié aux groupes suivant :

- MAHIER Loïc, PHALAVANDISHVILI Demetre, COUILLEROT Carol (budget de recherche et de transfert de technologie (R&T) des collectivités territoriales)
- Duclos Romain, JEHANNO Clément, CAILLAUD Pierre (effectifs d'étudiants inscrits dans les établissements et les formations de l'enseignement supérieur, recensés pour les années 2001-2002 à 2015-16)

Cette étape a demandé beaucoup de communication entre les groupes, afin de s'assurer du bon fonctionnement des liaisons, bien qu'en temps normal, il faut partir du principe que l'on lie notre dataset à un dataset déjà existant et bien formé. C'est donc normalement à nous de nous adapter à ceux déjà existants, ce qui n'a pas réellement été le cas ici car nous avons travaillé à plusieurs groupes pour cette étape.
Nous avons ainsi pu créer une requête portant sur plusieurs datasets (voir Annexe 7 et Annexe 8). Ainsi en utilisant le mot clé SERVICE <http://localhost:3030/OtherSet1> et en pointant sur le dataset du second groupe que nous avons ajouté en local sur Fuseki et au moyen de owl:sameAs nous arrivons à obtenir les informations appartenant aux deux datasets.
On se rend compte très rapidement que contrairement à une requête sur notre dataset seul qui peut prendre jusqu'à quelques minutes (jamais plus de 3 ou 4 minutes), une requêtes sur plusieurs datasets va être beaucoup plus longue (10~15 mins). Cela semble logique, car lorsque nous allons récupérer le résultat d'une ligne sur le dataset de l'autre groupe avec SERVICE, cela vérifie ensuite que la ligne match également avec notre dataset, et ainsi de suite pour toutes les lignes qui vont matcher dans SERVICE.
On peut également supposer que le fait que les requêtes soient exécutés en local rende le temps total d'exécution plus long. En effet, sur certains SPARQL endpoints il est tout à fait probable que

l'implémentation des opérateurs soient différentes et que l'exécution de notre « SERVICE » soit répartie sur plusieurs endpoints qui rendraient des résultats simultanément, accélérant ainsi la vitesse de la requête.

# IV- Création d'inférences

Dans l'étape numéro quatre, il est question de mettre en place des règles d'inférence et un raisonnement sur nos données.
Afin de continuer notre travail sur la même technologie, nous avons décidé de mettre en place nos règles et de les exploiter grâce à Fuseki.
Pour cela nous avons trouvé un tutoriel portant sur les inférences avec Fuseki, le voici : tutoriel. Il explique comment mettre en place notre propre inférence sans utiliser celle de owl ou de rdfs.
Dans un premier temps, nous créons un fichier model.ttl. Ce fichier va définir les inférences que nous souhaitons mettre en place. Pour cela nous utilisons les vocabulaires de owl et de rdfs, nous redéfinirons par la suite le fonctionnement de certains d'entre eux car nous n'utilisons pas les méthodes de raisonnements de ceux-ci.
Ce fichier nous permet de définir le fonctionnement de certaines des propriétés et des classes que nous avons créé dans <http://example.org/ns#>. Comme pour les triples suivants de notre fichier construct.sparql :

```
frapo:Funding ns:hasIndicator [frapo:hasCode ?code_indi;
                               frapo:BudgetInformation ?indicateur];,
```

nous avons créé les inférences suivantes dans model.ttl : Voir Annexe 9.
Dans le cas présent, nous définissons ns:hasIndicator comme une propriété fonctionnelle vers un objet (cela signifie qu'un sujet ne peut avoir qu'une instance d'un objet). Nous indiquons respectivement par le biais de rdfs:range et rdfs domain le type du sujet et de l'objet concerné par ce prédicat. La classe ns:Indicator y est aussi définie, elle est le sujet des prédicats frapo:hasCode et frapo:BudgetInformation.
On reproduit ce même modèle d'inférence pour chaque prédicat que nous avons créé. Pour le cas des objets qui contiennent des valeurs booléennes ou des chaînes de caractères, nous les définissons respectivement comme étant de type xsd:boolean et xsd:string.
Sachant que nous n'utilisons pas les inférences du vocabulaire owl et rdfs, nous redéfinissons basiquement les classes et les propriétés des vocabulaires utilisés dans notre dataset. Ceci va nous permettre de construire des règles dans notre fichier model.rules, afin de mettre en place une forme de raisonnement sur les données. Nous définissons principalement les classes ainsi que les sous-classes qui leur correspondent, comme pour :

```
frapo:Funding rdf:type owl:Class;
          rdfs:subClassOf frapo:FinancialEntity.
```

Les méthodes de raisonnement désactivées, nous allons définir des règles de raisonnement dans model.rules. Pour créer une règle de raisonnement, il faut respecter cette mise en forme :
« [rulename : (condition sur un triple) → (triple engendré)] ».
Si la condition sur le triple est respectée alors le triple engendré est créé. Prenons un exemple avec :

```
[rule3:  (frapo:Funding  rdfs:subClassOf  frapo:FinancialEntity)  (?s
rdf:type frapo:Funding) -> (?s rdf:type frapo:FinancialEntity)]
```

Cette règle se traduit comme ceci : Si frapo:Funding est une sous-classe de frapo:FinancialEntity et si un sujet est de type frapo:Funding, alors le sujet est de type frapo:FinancialEntity. On observe qu'il peut y avoir plusieurs triples de conditions.
Nous avons défini quatre règles qui permettent de lier une classe et sa classe parente, de façon que si un objet est d'un certain type, alors il est aussi de type parent.
Une fois ces différents fichiers créés, il faut configurer Fuseki de sorte que nos inférences et raisonnements soient pris en compte par celui-ci. Pour cela, il faut modifier le fichier de configuration de notre dataset Fuseki, et d'y ajouter ceci : Voir Annexe 10.
Les lignes **#1** et **#2** permettent de définir l'emplacement des fichiers contenant les triples de notre dataset ainsi que les inférences. Ensuite, la ligne **#3** permet de donner l'emplacement de nos règles de raisonnements. Une fois ceci fait, les données du dataset, de l'inférence et de raisonnement seront directement chargées au lancement de Fuseki. Nous avons aussi appliqué ceci pour les datasets des groupes avec lesquels nous avons lié nos données.
Dès lors il suffit de lancer des requêtes SPARQL, afin de tester nos inférences, voici quelques exemples : Annexe 11, Annexe 12.
La requête en Annexe 11 retourne bien tous les sujets de type Funding, ceci est possible grâce à la rule3 que nous avons créée. La requête en Annexe 12 retourne bien les résultats en Annexe 13, On observe bien l'apparition de frapo:BudgetInformation et frapo:BudgetedAmount en prédicat dans les résultats.

# V- Liaison des données au cloud de linked data

Dans cette étape il était question de proposer des liens et des propriétés afin de lier nos data au cloud de linked data.
Avant toute chose, il existe plusieurs principes que l'on se doit de respecter si l'on veut que nos datas soient lié au cloud de linked data :

1) Use URIs as names for things
2) Use HTTP URIs so that people can look up those names
3) When someone looks up a URI, provide useful RDF information
4) Include RDF statements that link to other URIs so that they can discover related things

En ce qui nous concerne, nos données sont bien au format RDF et notre mapping est bien effectué sur des ressources que nous identifions comme des URI.

Cependant, nous possédons des attributs ayant un préfixe ns: que nous avons créé nous même, il faudrait donc changer ces préfixes par certains déjà existants.
Nous avons lié notre dataset à ceux d'autres groupes au moyen d'une URI créée à partir du code de région car cela nous arrangeait et parce que nous trouvions cela plus simple vis à vis des noms des régions qui pouvaient différer en fonction des datasets. Néanmoins afin de pouvoir lier notre dataset au cloud de linked data, nous avons créé une URI de la même manière que les anciennes mais avec le nom de la région cette fois là, nous permettant ainsi de lier notre dataset avec dbpedia via les régions.

# VI- Vocabulaire VoID

Dans cette dernière étape, il nous a été demandé d'utiliser les vocabulaire VoID (pour Vocabulary of Interlinked Dataset) afin de fournir à notre dataset des métadonnées. Les informations qui vont être fournies par le VoID ont pour but de permettre aux potentiels utilisateurs du dataset de connaître les différentes informations que l'on peut y trouver.
Notre fichier VoID va ainsi contenir des informations telles que les noms des personnes qui ont créé le dataset sémantifié (nous pour ce cas précis), ou plus important des liens vers les datasets auxquels celui-ci est lié.

# VII- Annexes

## I. Annexe 1


```
"records": [
   {
      "datasetid": "fr-esr-rd-moyens-entreprises-
      intensite",
      "recordid":
       "17c9f35e6c52f571ba04ba38de591d4c3086ad62",
      "fields": {
         "indicateur": "Effectifs de R&D",
         "type_de_personnel": "Tous types de personnel",
         "code_intensite": "0",
         "code_indicateur": "pers",
         "intensite_technologique": "Toutes intensités
         technologiques",
         "zone_geographique": "France",
         "code_type_pers": "0",
         "secret": "non",
         "code_zone_geo": "100",
         "valeur": 251446.28,
         "etat_des_donnees": "semi-définitif",
         "annee": "2013",
         "non_disponible": "non"
      },
      "record_timestamp": "2016-04-30T09:20:33+00:00"
   },
```


*Illustration 1: Données en format JSON avant Tarql*

## II. Annexe 2


```
<http://ex.org/fr-esr-rd-moyens-entreprises-intensite/codeRegion/100>
        rdf:type    frapo:hasCode ;
        owl:sameAs  <http://ex.org/budget/codeRegion/100> ;
        owl:sameAs  <http://ex.org/nous/codeRegion/100> .

<http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France>
        rdf:type    frapo:hasLocation ;
        owl:sameAs  <http://fr.dbpedia.org/resource/France> .

<http://ex.org/fr-esr-rd-moyens-entreprises-intensite/1002013pers00>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  frapo:Funding ;
        dbo:Place  _:b2328 .

_:b2328  frapo:hasLocation  <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France> ;
        frapo:hasCode      <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/codeRegion/100> .

<http://ex.org/fr-esr-rd-moyens-entreprises-intensite/1002013pers00>
        frapo:hasDeliveryDate  "2013"^^xsd:gYear ;
        ns:hasIndicator        _:b2329 .

_:b2329  frapo:hasCode           "pers" ;
        frapo:BudgetInformation  "Effectifs de R&D" .

<http://ex.org/fr-esr-rd-moyens-entreprises-intensite/1002013pers00>
        ns:hasStaff  _:b2330 .

_:b2330  frapo:hasCode           0 ;
        frapo:BudgetInformation  "Tous types de personnel" .

<http://ex.org/fr-esr-rd-moyens-entreprises-intensite/1002013pers00>
        ns:hasIntensity  _:b2331 .

_:b2331  frapo:hasCode           0 ;
        frapo:BudgetInformation  "Toutes intensités technologiques" .

<http://ex.org/fr-esr-rd-moyens-entreprises-intensite/1002013pers00>
        ns:hasSecret          "0"^^xsd:boolean ;
        ns:hasNotAvailable    "0"^^xsd:boolean ;
        ns:hasDataState       "semi-définitif" ;
        frapo:BudgetedAmount  "251446.28"^^xsd:double .
```


*Illustration 2: Données en format Turtle après Tarql*

# III. Annexe 3

```
#Requete n°1
PREFIX frapo: <http://purl.org/cerif/frapo/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX ns: <http://example.org/ns#>

Select  ?zone_geo ?annee ?type
Where { ?x frapo:BudgetedAmount ?valeur;
          frapo:hasDeliveryDate ?annee;
          dbo:Place ?place;
          ns:Staff ?pers.
       ?place frapo:hasLocation ?zone_geo.
       ?pers frapo:BudgetInformation ?type.
     FILTER(?valeur > 25000)}
ORDER BY ?annee
```

Cette première requête permet de récupérer les zones géographique, l'année et le type de personnel pour lesquels les moyens accordés sont supérieurs à 25 000 M €.

# IV. Annexe 4

| | zone_geo | annee | type |
|---|---|---|---|
| 1 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France> | "2001"^^xsd:gYear | "Chercheurs" |
| 2 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2001"^^xsd:gYear | "Soutien à la recherche" |
| 3 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2001"^^xsd:gYear | "Tous types de personnel" |
| 4 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2001"^^xsd:gYear | "Chercheurs" |
| 5 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France> | "2001"^^xsd:gYear | "Tous types de personnel" |
| 6 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France> | "2001"^^xsd:gYear | "Soutien à la recherche" |
| 7 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France> | "2002"^^xsd:gYear | "Chercheurs" |
| 8 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France> | "2002"^^xsd:gYear | "Soutien à la recherche" |
| 9 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2002"^^xsd:gYear | "Chercheurs" |
| 10 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2002"^^xsd:gYear | "Tous types de personnel" |

*Illustration 3: Résultat via Fuseki pour la première requête*

Ne sont présents que les 16 premiers résultats, cela permet néanmoins de donner un aperçu.

# V. Annexe 5

```
#Requete n°2
PREFIX frapo: <http://purl.org/cerif/frapo/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

Select DISTINCT ?zone_geo (SUM(?valeur) as ?sum)
Where { ?x frapo:BudgetedAmount  ?valeur;
           frapo:hasDeliveryDate ?annee;
         dbo:Place ?place.
      ?place frapo:hasLocation ?zone_geo.
      FILTER(?annee = "2010"^^xsd:gYear && ?zone_geo != "France")}
GROUP BY ?zone_geo
ORDER BY DESC(?sum)
LIMIT 5
```

Cette seconde requête permet de récupérer les 5 zones géographiques ayant eu le plus de moyens leur ayant été attribuées lors de l'année 2010 par ordre décroissant (nous ne prenons ici pas en compte la "région" France).

# VI. Annexe 6

| | zone_geo | sum |
|---|---|---|
| 1 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/France> | "997261.04e0"^^xsd:double |
| 2 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "398090.1800000001e0"^^xsd:double |
| 3 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Rh%C3%B4ne-Alpes> | "127462.7e0"^^xsd:double |
| 4 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Midi-Pyr%C3%A9n%C3%A9es> | "72621.27999999997e0"^^xsd:double |
| 5 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Provence-Alpes-C%C3%B4te%20d'Azur%20+%20Corse> | "60418.85e0"^^xsd:double |

Showing 1 to 5 of 5 entries

*Illustration 4: Résultat via Fuseki de la seconde requête*

# VII. Annexe 7

```
PREFIX frapo: <http://purl.org/cerif/frapo/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

Select DISTINCT ?zoneGeo ?annee (SUM(?valeur) as ?sum) (SUM(?
otherValeur1) as ?otherSum1)
Where {
  ?funding a frapo:Funding;
            frapo:BudgetedAmount ?valeur;
          frapo:hasDeliveryDate ?annee;
          dbo:Place ?place.

  ?place frapo:hasCode ?codeReg;
      frapo:hasLocation ?zoneGeo.

  ?codeReg owl:sameAs ?otherCodeReg1.

  SERVICE <http://localhost:3030/OtherSet1> {
  ?otherFunding a frapo:Funding;
                frapo:BudgetedAmount ?otherValeur1;
                dbo:year ?annee1;
                dbo:code ?otherCodeReg1.
  }
  FILTER(?annee = ?annee1)
}
GROUP BY ?zoneGeo ?annee
ORDER BY ?annee ?zoneGeo
LIMIT 10
```

Cette requête va récupérer dans les deux datasets les valeurs des fonds attribués par région et par année par ordre croissant des années.

# VIII. Annexe 8

| | zoneGeo | annee | sum | otherSum1 |
|---|---|---|---|---|
| 1 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2007"^^xsd:gYear | "1.3954477039999953E7"^^xsd:double | "13366.310000000018e0"^^xsd:double |
| 2 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Bretagne> | "2007"^^xsd:gYear | "1908672.4799999732e0"^^xsd:double | "9413.930000000018e0"^^xsd:double |
| 3 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Centre> | "2007"^^xsd:gYear | "1517356.3599999999e0"^^xsd:double | "12637.470000000001e0"^^xsd:double |
| 4 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Pays%20de%20la%20Loire> | "2007"^^xsd:gYear | "1030984.9199999959e0"^^xsd:double | "9392.080000000004e0"^^xsd:double |
| 5 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2008"^^xsd:gYear | "1.3130281800000042E7"^^xsd:double | "16751.15999999996e0"^^xsd:double |
| 6 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Bretagne> | "2008"^^xsd:gYear | "1935467.999999992e0"^^xsd:double | "10058.219999999974e0"^^xsd:double |
| 7 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Centre> | "2008"^^xsd:gYear | "1548510.6000000008e0"^^xsd:double | "11254.269999999995e0"^^xsd:double |
| 8 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Pays%20de%20la%20Loire> | "2008"^^xsd:gYear | "1057364.6399999934e0"^^xsd:double | "14960.41000000001e0"^^xsd:double |
| 9 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/%C3%8Ele-de-France> | "2009"^^xsd:gYear | "1.3519360439999957E7"^^xsd:double | "17404.38000000001e0"^^xsd:double |
| 10 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/region/Bretagne> | "2009"^^xsd:gYear | "1971155.9999999914e0"^^xsd:double | "10519.160000000025e0"^^xsd:double |

Showing 1 to 10 of 10 entries

*Illustration 5: Résultat via Fuseki de la requête portée sur plusieurs datasets*

# IX. Annexe 9

```
ns:Indicator rdf:type owl:Class, frapo:BudgetInformation.

ns:hasIndicator rdf:type owl:ObjectProperty, owl:FunctionalProperty;
                rdfs:range ns:Indicator;
                rdfs:domain frapo:Funding.
```

Exemple d'inférences du fichier model.ttl.

# X. Annexe10

```
:dataset         rdf:type ja:RDFDataset ;
rdfs:label "inference" ;
ja:defaultGraph
[ rdfs:label "inference" ;
a ja:InfModel ;

    #Reference to model.ttl file
    ja:content [ja:externalContent
<file:/home/bobby/Documents/Fuseki/run/databases/inference/model.ttl> ]
; #1
```

```
    #Reference to data.ttl file
    ja:content [ja:externalContent
<file:/home/bobby/Documents/Fuseki/run/databases/inference/project.ttl>
] ; #2

    #Disable OWL-based reasoner
    #ja:reasoner [ja:reasonerURL
<http://jena.hpl.hp.com/2003/OWLFBRuleReasoner>] ;

    #Disable RDFS-based reasoner
    #ja:reasoner [ja:reasonerURL
<http://jena.hpl.hp.com/2003/RDFSExptRuleReasoner>] ;

    #Enable Jena Rules-based reasoner and we point the location of
myrules.rules file
    ja:reasoner [
        ja:reasonerURL <http://jena.hpl.hp.com/2003/GenericRuleReasoner>
;
        ja:rulesFrom
<file:/home/bobby/Documents/Fuseki/run/databases/inference/model.rules>
; #3
    ] ;
  ] ;
 .
```

Configuration de Fuseki.

# XI. Annexe11

```
PREFIX frapo: <http://purl.org/cerif/frapo/>

SELECT ?s
WHERE {
  ?s rdf:type frapo:FinancialEntity.

}
```

Requête pour vérifier le fonctionnement de l'inférence et du raisonnement, retourne tous les sujet de type FinancialEntity

# XII. Annexe12

```
PREFIX frapo: <http://purl.org/cerif/frapo/>

SELECT ?s ?p ?o
WHERE {
  ?s frapo:BudgetInformation ?o.
  ?s ?p  ?o

}
```

Requête pour vérifier le fonctionnement de l'inférence et du raisonnement, retourne tous les triples avec un sujet et un objet qui sont dans le triple «?s frapo:BudgetInformation?o ».

# XIII. Annexe13

| | s | p | o |
|---|---|---|---|
| 1 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/742012pers18> | frapo:BudgetInformation | "376.95"^^xsd:double |
| 2 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/742012pers18> | frapo:BudgetedAmount | "376.95"^^xsd:double |
| 3 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/732003dird5> | frapo:BudgetInformation | "39.03"^^xsd:double |
| 4 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/732003dird5> | frapo:BudgetedAmount | "39.03"^^xsd:double |
| 5 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/522007dird6> | frapo:BudgetInformation | "49.38"^^xsd:double |
| 6 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/522007dird6> | frapo:BudgetedAmount | "49.38"^^xsd:double |
| 7 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/832008pers09> | frapo:BudgetInformation | "394.92"^^xsd:double |
| 8 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/832008pers09> | frapo:BudgetedAmount | "394.92"^^xsd:double |
| 9 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/412011dird6> | frapo:BudgetInformation | "37.72"^^xsd:double |
| 10 | <http://ex.org/fr-esr-rd-moyens-entreprises-intensite/412011dird6> | frapo:BudgetedAmount | "37.72"^^xsd:double |

Résultats de la requête en Annexe 12.