Acronyme	**Campus AAR**
Titre du projet	**Campus « Archives Audiovisuelles de la Recherche ».** *Le démonstrateur d'un environnement numérique pour la production, description/indexation et publication d'archives audiovisuelles.* *Domaine d'application : les humanités numériques.*
Proposal title	**ARA Campus.** The Audiovisual Research Archive Campus – a demonstrator for the production, description and publishing of digital audiovisual archives. Domain of expertise: Digital humanities

Tâche 3.3 : Outil de gestion des règles de saturation

Partenaire(s) concerné(s)	Armadillo
Référence convention/décision	**ANR-13-CORD-0016-01**
Rédacteur(s) du rapport	Rania Soussi
Téléphone de contact	0141230214
Adresse électronique (de contact)	rania@armadillo.fr
Date :	01/04/2015

ID document	Campus AAR – Rapport
Distribution	☒ Public ☐ Restreint ☐ Interne
Description	Description de l'algorithme assurant la gestion des règles de saturation

Résumé général du rapport

Ce document décrit le processus utilisé pour saturer la base Armadillo.
Dans la première section, nous présentons un bref état de l'art des règles d'inférences qui permettent de saturer les graphes RDF. Ensuite la deuxième section présente la technique utilisée pour saturer la base Armadillo et les premiers résultats obtenus.

Table des matières

Introduction

Dans le cadre du projet Campus AAR, les données traitées en tant que analyse ou méta données audiovisuelle ont été modélisées sous format RDF/OWL2.

Cette modélisation a été choisie pour faciliter l'enrichissement des modèles de données et les modèles d'analyses ainsi que de permettre une interrogation rapide et pertinente des données.

Dans ce même contexte, le SGBD NoSQL ArmadilloDB doit assurer de son côté un mécanisme qui facilite le traitement de ces données et son enrichissement.

Ainsi, la saturation de la base avec des nouvelles données (métadonnées ou liaisons entre contenus) permettra d'enrichir les corpus des partenaires et améliorer la qualité des résultats de recherche.

Vu que dans le contexte du web sémantique les règles d'inférences permettant de déduire des nouveaux triplets, l'opération de saturation d'un graphe RDF consiste à appliquer ces règles d'inférence afin de matérialiser tous les triplets pouvant être déduits. On obtient ainsi un graphe saturé.

Dans ce rapport, nous allons décrire les algorithmes et les solutions proposées pour saturer la base Armadillo.

La première partie du rapport est consacrée à une présentation des notions de base des inférences en web sémantique et une brève présentation de l'état de l'art.

La deuxième partie du rapport est consacrée à la description de la solution adoptée et les premiers résultats obtenus.

1. Notions de base

Dans cette section, nous décrivons en premier temps le modèle de données RDF les vocabulaires RDF et OWL. Ensuite, nous présentons la définition des règles d'inférence applicable à un graphe RDF et la liste des règles proposées par le W3C.

1.1 Les modèles de données

Dans cette section, nous décrivons en premier temps le modèle de données RDF les vocabulaires RDF et OWL. Ensuite, nous présentons la définition des règles d'inférence applicable à un graphe RDF et la liste des règles proposées par le W3C.

1.1.1 Le modèle de donnés RDF

RDF[1] est un langage assertionnel conçu pour exprimer des propositions à l'aide de vocabulaires formels précis, notamment ceux définis avec RDFS (RDF Schema) ou par OWL (Conen *et al.* 2000). RDF est un standard de W3C qui est basé sur les graphes. Un graphe RDF est un ensemble de triplets (s, p, o) avec s est le sujet ayant comme propriété p, et la valeur de cette propriété est l'objet o.

Les ensembles des triplets construisent un graphe orienté dont les nœuds et les arcs sont étiquetés par des URIs (uniform resource identifiers). RDF intègre un ensemble de classes et de propriétés qui ont des espaces de noms commençant par rdf : ou rdfs : Par exemple rdf :type spécifie le type de classe à laquelle la ressource appartient.

1.1.2. RDFS

RDFS[2] fait partie de la recommandation RDF pour la description de son vocabulaire.
Il a été conçu pour mettre des contraintes entre les classes et les propriétés dans un graphe RDF.
RDFS permet de déclarer, en RDF, les propriétés et les classes utilisées pour décrire des données RDF et les relations hiérarchiques qui existent entre propriétés et entre classes. Il permet de définir :
– une ressource comme étant une classe de type rdfs:Class. Une classe est une ressource représentant un ensemble de ressources dont la valeur de la propriété rdf:type est cette classe ;
– une ressource comme étant une propriété de type rdf:Property ;
– des hiérarchies de classes avec la propriété rdfs:subClassOf ;
– des hiérarchies de propriétés avec la propriété rdfs:subPropertyOf ;
– le domaine et la portée d'une propriété avec les propriétés rdfs:domain et rdfs:range ;
– les labels des classes et propriétés avec la propriété rdfs:label ;
– les définitions en langue naturelle des classes et propriétés avec la propriété rdfs:comment.

[1] https://www.w3.org/RDF/
[2] http://www.w3.org/TR/2014/REC-rdf-schema-20140225/Overview.html

1.1.3. OWL

Cependant, Pour définir des ontologies riches, l'expressivité de RDFS devient insuffisante.

En effet RDFS ne permet pas d'exprimer l'unicité, l'union, l'intersection de classes, ni certaines propriétés algébriques de propriétés telles que la transitivité, la symétrie, etc. C'est ce qui a motivé la recommandation de Web Ontology Language (OWL) qui permet non seulement de déclarer des propriétés et des classes mais aussi de les définir.
OWL[3] a été recommandé par le W3C en 2004, et sa version 2 en 2009.
– C'est un méta-vocabulaire (comme RDFS) inspiré des logiques de descriptions avec valeurs concrètes (littéraux).
– Il définit plusieurs profils offrant des compromis différents en termes d'expressivité et de complexité.
– Il mime les capacités de méta-modélisation de RDFS.

La liste des principaux éléments que permet d'exprimer OWL :
– Equivalence de classes, en utilisant la propriété owl:equivalentClass,
– Equivalence de propriétés, en utilisant la propriété owl:equivalentProperty,
– Relations inverses, en utilisant la propriété owl:inverseOf,
– Relations symétriques, en utilisant la classe owl:SymmetricProperty,
– Relations transitives, en utilisant la classe owl:TransitiveProperty,
– Restrictions de propriétés, en utilisant la classe owl:Restriction et la propriété owl:onProperty,
– Contraintes de valeurs, en utilisant les propriétés owl:allValuesFrom, owl:someValuesFrom, owl:hasValue

1.2 Les inférences dans le web sémantique

Pour enrichir les graphes RDF, des règles d'inférences ont été créées pour exploiter la sémantique implicite de ces graphes. Dans cette section, nous présentons les règles d'inférences appliquées sur les données RDF.

1.2.1 Règle d'Inférence en RDF (entailment rules)

Plus généralement, une règle d'inférence est une fonction qui prend n-uplet de formules et rend une formule. Ses arguments sont appelés « les prémisses » et sa valeur la « conclusion ». Les règles d'inférence peuvent également être vues comme des relations liant prémisses et conclusions par lesquelles une conclusion est dite « déductible » ou « dérivable » des prémisses. Si l'ensemble des prémisses est vide, alors la conclusion est appelée un « théorème » ou un « axiome » de la logique.

Dans le domaine du web sémantique, l'inférence est une action ou un processus qui permet de découvrir des nouvelles expressions. Les triplets implicites sont considérés dans le graphe.

Une caractéristique importante de RDF est la modélisation de triplets implicites : ils sont considérés comme faisant partie d'un graphe, même s'ils ne sont pas explicitement présents dans celui-ci. Le W3C nomme RDF entailment le mécanisme par lequel les triplets implicites sont dérivés (ou engendrés) à partir des triplets explicites d'un graphe et de règles d'entailment (ou aussi règles d'inférences).

[3] http://www.w3.org/TR/2004/REC-owl-features-20040210/

En RDF, il existe quatre régimes d'entailment (ter et al., 2005)

- entailment Simple : elle n'interprété pas le vocabulaire RDF et RDFS

- entailment RDF : interprété le vocabulaire RDF

- entailment RDFS : interprété les vocabulaires RDF et RDFS

- entailment- D : utilise en plus les informations sur les types de données

Dans ce qui suit nous allons présenter ces techniques.

1.2.1.1 Entailment Simple

L'inférence immédiate (dite entailment simple) est le processus de créer de nouveaux triplets à travers l'application d'une seule règle d'inférence. D'une façon générale, un triplet (s,p,o) est impliqué par un graphe G si seulement si il y a une séquence d'application immédiate des règles d'inférences qui mène de G à (s,p,o).

Les règles d'inférences simples peuvent être résumé par le tableau 1, où

- E : fragment d'un graph RDF
- aaa, bbb : n'importe quel URI.
- uuu, vvv : n'importe quel URI ou nœud anonyme (un nœud anonyme c'est une ressource ou nœud d'un graphe RDF, qui n'est pas identifiée par une URI. elle peut être sujet ou objet d'un triplet)
- xxx, yyy: n'importe quel URI, nœud anonyme ou littéral
- lll: n'importe quel littéral
- eee,ddd : type de données
- ttt,sss :forme lexicale
- _:nnn: un nœud anonyme

Règle	**Si E contient**	**ajouter**
Se1	uuu aaa xxx .	uuu aaa _:nnn . où _:nnn présente une ressource anonyme allouée à xxx par la règle se1 ou se2.
Se2	uuu aaa xxx .	:nnn aaa xxx . où _:nnn présente une ressource anonyme allouée à uuu par la règle se1 ou se2.
lg	uuu aaa lll .	uuu aaa _:nnn . où _:nnn présente une ressource anonyme allouée au littéral lll par cette règle.
gl	uuu aaa _:nnn . where _:nnn identifie une ressource anonyme allouée au littéral lll par la règle lg.	uuu aaa lll .

Tableau1. Règles d'inférence simple

Par exemple en appliquant la règle s1 sur le triplet en entrée

Triplets entrées	**Eléments ajoutés**
fmsh:institution rdfs:label "institution"@en	fmsh:institution rdfs:label - :node

1.2.1.2 RDF Entailment

L'inférence RDF satisfait ce lemme : un ensemble de graphe G de type RDF implique un graphe E si seulement si un graphe qui peut dériver de G ainsi que des triplets axiomatique RDF en appliquant la règle lg (voir tableau 11) et les règles d'inférence RDF et qui implique simplement E.

Dans le tableau 2, on détaille ces règles.

Régle	**Si E contient**	**ajouter**
rdf1	uuu aaa yyy .	aaa rdf:type rdf:Property .
rdf2	uuu aaa lll . où lll est un littéral XML bien typé.	_:nnn rdf:type rdf:XMLLiteral . où _:nnn identifie une ressource anonyme allouée au littéral lll par la règle lg.

Tableau2. Règles d'inférence RDF

Par exemple en appliquant la règle rdf1 sur le triplet ayant comme sujet <http://www.ina.fr/core#snap> on peut ajouter un nouveau triplet à l'ontologie core.

Triplets entrées	**Eléments ajoutés**
<http://www.ina.fr/core#snap> rdfs:domain <http://www.ina.fr/core#TemporalLayer>	<http://www.ina.fr/core#snap> rdf :type rdf :Property

1.2.1.3 RDFS Entailement

Les inférences RDFS satisfait le lemme : un ensemble de graphe G de type RDF implique un graphe E si seulement s'il existe un graphe qui peut dériver de G et les triplets axiomatique de RDF et RDFS en appliquant les règles *lg* et les règles *gl* et les règles d'inférences RDF et RDFS et qui implique simplement E. ces règles peuvent être présentées dans le tableau3.

Règle	**Si E contient**	**ajouter**
rdfs1	uuu aaa lll	_:nnn rdf:type rdfs:Literal . où _:nnn présente un nœud anonyme allouée à lll par la règle lg.
rdfs2	aaa rdfs:domain xxx . uuu aaa yyy .	uuu rdf:type xxx .

rdfs3	aaa rdfs:range xxx . uuu aaa vvv .	vvv rdf:type xxx .
rdfs4a	uuu aaa xxx .	uuu rdf:type rdfs:Resource .
rdfs4b	uuu aaa vvv.	vvv rdf:type rdfs:Resource .
rdfs5	uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .	uuu rdfs:subPropertyOf xxx .
rdfs6	uuu rdf:type rdf:Property .	uuu rdfs:subPropertyOf uuu .
rdfs7	aaa rdfs:subPropertyOf bbb. uuu aaa yyy .	uuu bbb yyy .
rdfs8	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf rdfs:Resource
rdfs9	uuu rdfs:subClassOf xxx. vvv rdf:type uuu .	vvv rdf:type xxx
rdfs10	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf uuu .
rdfs11	uuu rdfs:subClassOf vvv . vvv rdfs:subClassOf xxx .	uuu rdfs:subClassOf xxx
rdfs12	uuu rdf:type rdfs:ContainerMembershipProperty	uuu rdfs:subPropertyOf rdfs:member
rdfs13	uuu rdf:type rdfs:Datatype	uuu rdfs:subClassOf rdfs:Literal

Tableau3. Règles d'inférence RDFS

Par exemple pour le sujet <http://www.ina.fr/core#TemporalSegment>, en appliquant la règle rdfs9, on peut ajouter un nouveau triplet à la base

Triplets entrées	**Eléments ajoutés**
<http://www.ina.fr/core#TemporalSegment> rdfs:subClassOf <http://www.ina.fr/core#Segment> <http://www.ina.fr/core#Segment> rdf :type owl:Class	<http://www.ina.fr/core#Segment> rdf :type owl:Class

1.2.1.4 Entailment en utilisant les types de donnes (exemple D-entailment)

Les types d'inférences précédentes ne prennent pas en considération les types de données. En effet, rdf:XMLLiteral est le seul type de données considéré. Les spécifications de la sémantique RDF (Hayes, 2004) définissent la notion de D entailment (Datatype entailment) qui étend les RDFS entailment en supportant les types des données.

Ter Horst (2005) définit la notion de D* entailment qui est une autre extention de RDFS entailment.

Le D-entailment étend les RDFS entailment. En donnant un ensemble de datatype mapping D, un ensemble S de graphe RDF généralisé si chaque D-interpretation I qui satisfait S satisfait aussi G.

On présente dans la table 3, un exemple des règles de cette catégorie :

Règle	**Si E contient**	**ajouter**
rdfD1	ddd rdf:type rdfs:Datatype . uuu aaa "sss"^^ddd .	_:nnn rdf:type ddd . où _:nnn présente un noeud annonyme attribué à "sss"^^ddd par la règle lg.
rdfD2	ddd rdf:type rdfs:Datatype . uuu aaa "sss"^^ddd .	uuu aaa "ttt"^^ddd .
rdfD3	ddd rdf:type rdfs:Datatype . eee rdf:type rdfs:Datatype . uuu aaa "sss"^^ddd .	uuu aaa "ttt"^^eee .

Tableau4. Règles d'inférence D

1.2.3La saturation des graphes

SPARQL est le standards W3C pour requêter des graphes RDF. SPARQL se base sur des graphs pattern basique (BGP). BGP est un ensemble de pattern de triplets. L'évaluation d'une requête q dans un graphe RDF *G* en utilisant seulement les requêtes explicites du graphe donne généralement une information non complète. Pour cela, il y a deux méthodes pour obtenir une réponse pertinente : la première c'est la reformulation des requête qui consiste à étendre la requête d'origine pour pouvoir détecter les triplets implicites (voir par exemple les techniques utiliser dans (Calvanese et al,2007) (Gottlob et al,2011).

La deuxième technique c'est la saturation des graphes. Comme définit dans (Bursztyn et al.,2015), Les règles d'inférences immédiates permettent de définir une saturation finie d'un graph RDF G qui est un graph G^{∞} définie comme un point fixe, en appliquant un ensemble n de règles d'inférences. La saturation d'un graphe RDF est unique et ne contient plus des triplets implicites. Il est important de noter que le RDF entailment fait partie de la norme RDF. Par conséquent, en RDF, tout graphe G est (sémantiquement) équivalent à sa saturation G^{∞}.

2. Algorithme de saturation

2.1 Description de l'architecture et des données initiales

Conformément à ce qui a été détaillé dans le livrable 1.2 de spécification de l'architecture fonctionnelle, le SGBD NoSQL d'Armadillo, avec ses fonctionnalités d'indexation, de recherche avancée et de traitement des médias, est utilisé aux différents modules du campus AAR.

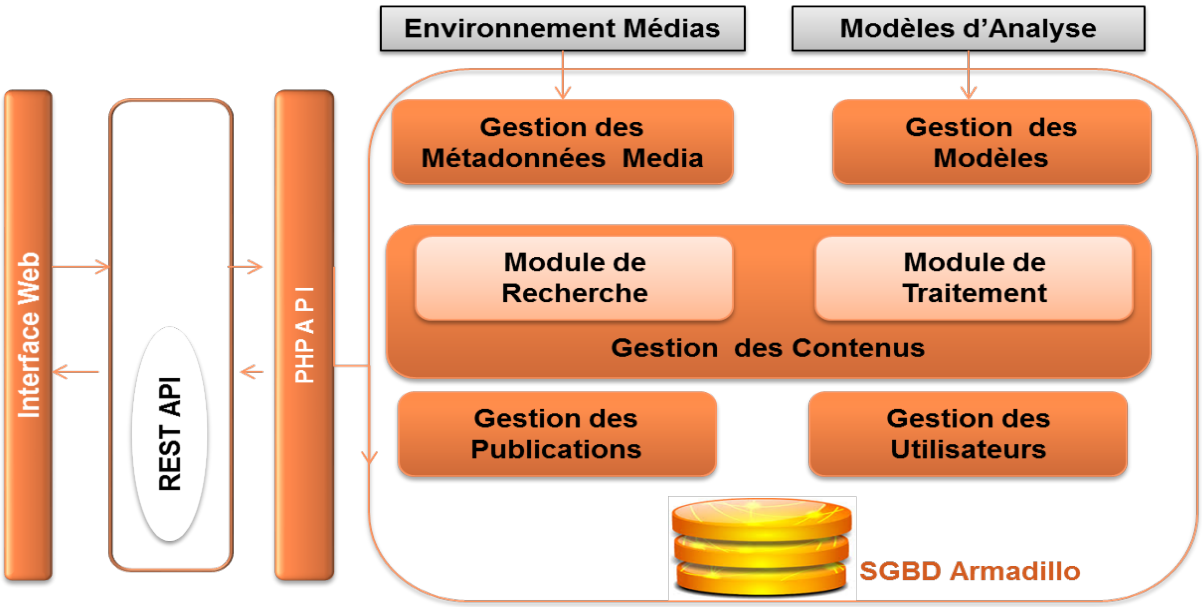

Figure 1.Architecture Fonctionnelle de Campus AAR

Ainsi, l'enrichissement de la base de données s'effectue directement à travers les différents modules tels que des analyses et la gestion des publications.

Afin de maximiser la sécurité des données et pour saturer les données en utilisant un moteur d'inférence adapté au modèle de données de campusaar, nous avons accosté au SGBD Armadillo le triplestore openRDF Sésame. Ainsi les données initialement stockées dans le SGBD Armadillo sont dupliquées par la suite dans la partie base de Sésame.

Les requêtes SPARQL envoyées à la base Armadillo à travers le module de traitement de requêtes est communiqué à Sésame pour obtenir des résultats enrichie avec les règles d'inférences.

La communication entre le SGBD Armadillo et la base sésame est assurée à travers des requêtes http.

La figure 2 présente l'architecture fonctionnelle de cet accostage et un exemple de communication avec le module d'analyse.

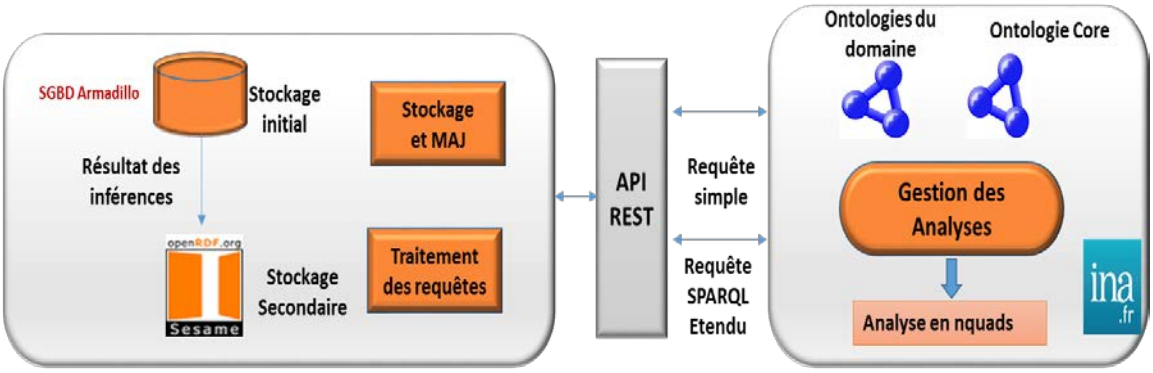

Figure 2. couche SGBD de la base Armadillo

Sésame stocke les graphes RDF dans des entrepôts qui peuvent avoir des types différents

- memory – un dépôt RDF in-memory
- memory-rdfs –un référentiel principale- in memory avec application de l'inférence RDFS
- memory-rdfs-dt -- un référentiel principale avec inférence RDFS et hiérarchie de type directe
- native -- un référentiel qui stocke la structure de données sur le disque
- native-rdfs -- un référentiel natif avec RDFS entailment
- native-rdfs-dt -- un référentiel natif avec inférence RDFS et hiérarchie de type directe
- remote – un référentiel qui sert comme un proxy pour un référentiel sur un serveur de sésame

Pour pouvoir appliquer plus de règles d'inférences nous avons choisi des dépôts de type native-rdfs-dt.

Dans ce qui suit nous allons présenter les premiers résultats de cette saturation.

2.2 Procédure de saturation et premiers résultats

Dans un premier temps, les analyses et les ontologies créées par l'INA (décrites dans le livrable 3.2) sont importées dans la base native Armadillo et par la suite dans le triplestore Sesame.

Les données importées sans appliquer des règles d'inférences forment 310395 triplets et 5563 graphes.

Pour assurer la saturation de la base de données, on utilise la couche inférence proposée en open source par openRDF sésame.

En appliquant les règles d'inférence sur l'ontologie http://escom.msh-paris.fr/_staticTypes.owl qui est composé initialement par 4096 triplets, on obtient 13950 triplets.

Nous allons décrire la technique de saturation avec un exemple basé sur une ressource média de l'ESCoM-AAR.

Initialement le sujet fmsh:004cf88f-358c-4645-8440-f97f0778d1a0 est rattaché comme subclasse fmsh:56383bec-f47d-4a5a-bee1-0e23685c40be

Les n-quads correspondant est le suivant :

Sujet	Prédicat	Objet	Contexte
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:56383bec-f47d-4a5a-bee1-0e23685c40be	http://escom.msh-paris.fr/_staticTypes.owl

Ce triplet est représenté par le graphe RDF du figure3.

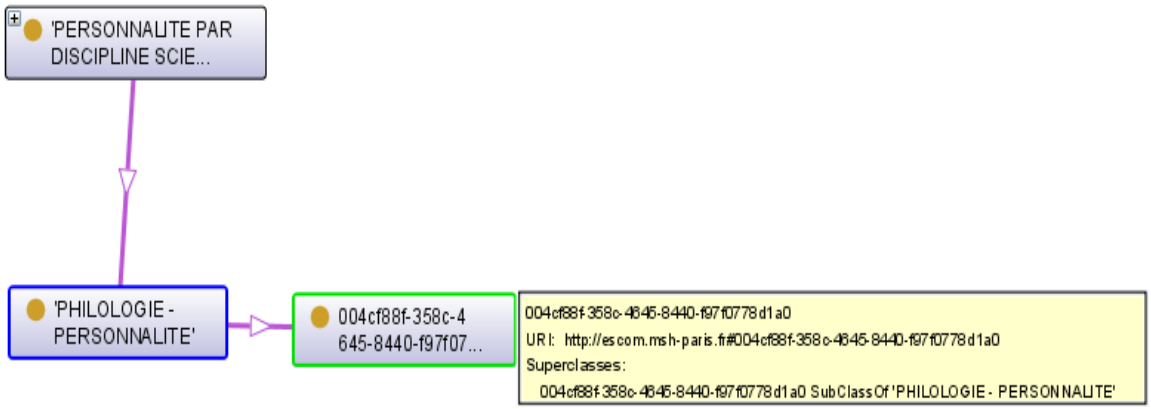

Figure 3.graphe RDF du sujet fmsh:004cf88f-358c-4645-8440-f97f0778d1a0

En appliquant les règles d'inférences RDFS, on obtient :

Sujet	Prédicat	Objet	Contexte
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdf:type	rdfs:Class	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdf:type	rdfs:Resource	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	rdfs:Resource	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:e2e5b80a-12b2-4cc5-b21e-fc0296c01cac	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:e3af27ff-b718-4d42-bdcb-cec4ba9853b2	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:f177b9ba-8317-4828-aedb-1e6d3159c41e	

Sujet	Prédicat	Objet	Contexte
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:56383bec-f47d-4a5a-bee1-0e23685c40be	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:060fb5f2-cf6a-4a00-a729-d40ed211dba6	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:6b0e162c-af26-4754-a6fa-fed3ca352ef3	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	<http://www.openrdf.org/schema/sesame#directType>	rdfs:Class	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	<http://www.openrdf.org/schema/sesame#directSubClassOf>	fmsh:56383bec-f47d-4a5a-bee1-0e23685c40be	
fmsh:004cf88f-358c-4645-8440-f97f0778d1a0	rdfs:subClassOf	fmsh:56383bec-f47d-4a5a-bee1-0e23685c40be	http://escom.msh-paris.fr/_staticTypes.owl

Ainsi le graphe résultant est le graphe RDF décrit par la figure 4, ce graphe comporte 12 nouvelles relations qui permettront par la suite d'enrichir les résultats des requêtes SPARQL et améliorer les temps de réponses.

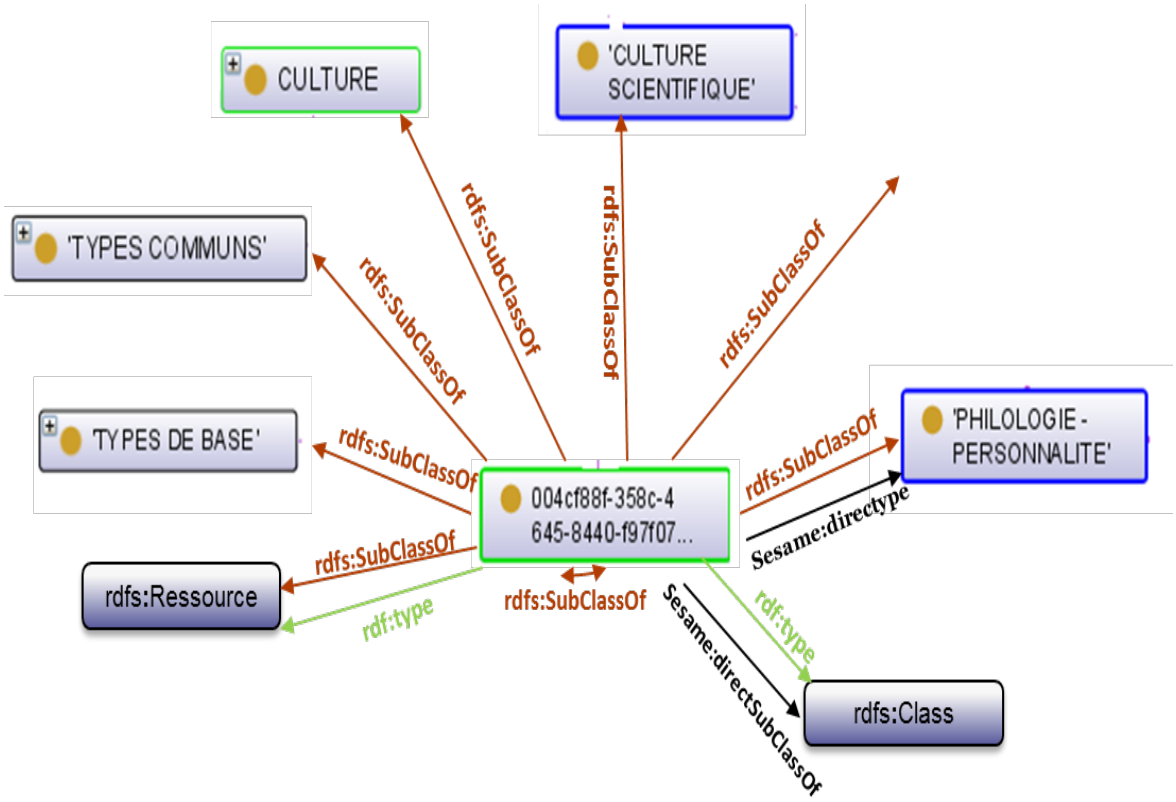

Figure 4.graphe RDF du sujet fmsh:004cf88f-358c-4645-8440-f97f0778d1a0 obtenu après inférence

2.3 Création des nouvelles règles d'inférences

Appliquer des règles d'inférences permet d'ajouter des nouvelles relations au graphe initial qui permettent d'enrichir la sémantique des données.

En étudiant, les relations existantes entres les différents concepts et les besoins des applications qui utilisent les graphes RDF, des nouvelles règles d'inférences personnalisées peuvent être proposées.

Par exemple dans le cas de campus-aar le graphe décrit par la figure 5 (extrait du graphe RDF comportant l'ensemble des ontologies), représente la relation core :about qui relie le concept Segment avec le concept Entité.

Cette relation générique qui peut relier différents type d'entité (qui peut être des sujets, des lieux ou des objets) peut devenir plus spécifique à chaque type d'instances. Puis, les nouvelles relations peuvent être définit sous forme des règles d'inférences qui permettra de relier les différents instances des graphes RDF déjà stockées dans la base.

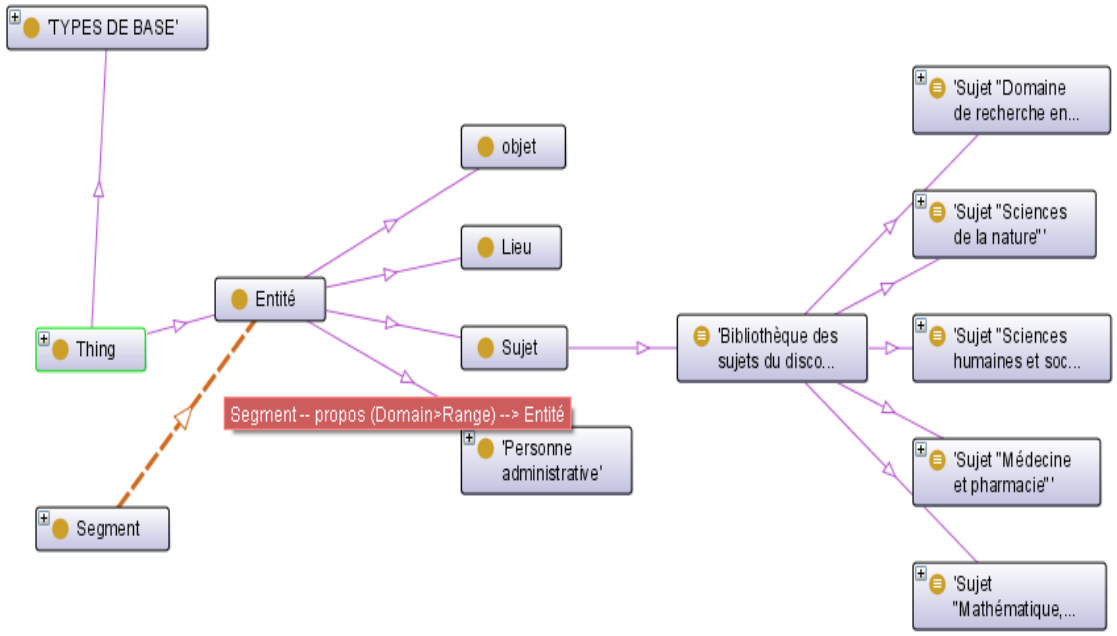

Figure 5.graphe RDF de la relation entre Segment et Entité

3. Références

Bursztyn, Damian, et al. "Reasoning on Web Data: Algorithms and Performance." *ICDE-31st International Conference on Data Engineering*. 2015.

Calvanese, D., Giacomo, G. D. , D. Lembo, M. Lenzerini, and R. Rosati, “Tractable Reasoning and Efficient Query Answering in escription
Logics: The DL-Lite Family,” J. of Automated Reasoning (JAR), 2007.

Conen, Wolfram, and Reinhold Klapsing. "A logical interpretation of RDF."*Journal of Electronic Transactions on Artificial Intelligence (ETAI), Area: The Semantic Web (SEWEB)* 5 (2000).

G. Gottlob, G. Orsi, and A. Pieris, “Ontological Queries: Rewriting and Optimization,” in ICDE, 2011, keynote.

ter Horst, Herman J. "Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary."*Web Semantics: Science, Services and Agents on the World Wide Web* 3.2 (2005): 79-115.