Interroger intuitivement le Web des Données avec SimplePARQL

Thomas Raimbault Sonia Djebali

DE VINCI RESEARCH CENTER — ESILV
Pôle Universitaire Léonard De Vinci, Paris – La Défense

{thomas.raimbault, sonia.djebali}@devinci.fr

Résumé

SimplePARQL est une approche intuitive d'interrogation du Web des Données ne nécessitant pas une connaissance préalable de la structure des bases RDF ni des identifiants des ressources. SimplePARQL est une généralisation du langage de requêtes SPARQL utilisant, en plus des mots-clés SPARQL, des termes du langage naturel au sein des triplets. Nous proposons la plateforme `universal-endpoint.com`*, qui permet d'interroger des bases RDF du Web des Données avec des requêtes SimplePARQL. Les requêtes SPARQL sont aussi acceptées.*

Mots Clefs

SimplePARQL, SPARQL intuitif, Web des Données, Linked Data, RDF

Abstract

SimplePARQL is an intuitive approach to query the Web of Data, where knowing the ontology (classes and properties) and resources' identifiers from an RDF base are not required. SimplePARQL is a generalisation of SPARQL query langage using keywords *in addition to SPARQL elements. We implemented our approach on the platform* `universal-endpoint.com`*, where SimplePARQL (and SPARQL) queries can be written and executed.*

Keywords

Intuitive SPARQL, Web of Data, Linked Data, RDF bases

1 Introduction

Le travail de recherche présenté et mis en application dans cet article est tiré de l'article théorique [3], publié en anglais en 2015, portant sur la nouvelle approche SimplePARQL. L'approche SimplePARQL offre la possibilité d'interroger intuitivement les bases RDF du Web des Données, à travers les endpoints SPARQL [7] existants, sans nécessiter une connaissance préalable de la structure ontologique de ces bases et/ou des identifiants des ressources.

Le Web des Données – encore appelé Linked Data ou Web Sémantique [2] – est constitué de centaines de bases RDF [5] inter-liées formant un vaste réseau de milliards de triplets RDF. Une base RDF est composée d'un ensemble de *triplets*, où chaque triplet s'exprime de la forme (sujet, prédicat, objet). Le sujet représente la *ressource* à décrire, le prédicat représente une *propriété* applicable à cette ressource, enfin l'objet représente la valeur. Chaque *ressource* est identifiée de manière unique, par un IRI [1] pour un accès à travers le Web, au sein de la base RDF où elle est stockée. Pour récolter et manipuler les données d'une base RDF, SPARQL [2] est le langage de requêtes recommandé par le W3C [3]. L'écriture d'une requête SPARQL est cependant complexe, et reste inaccessible à la plupart des utilisateurs potentiels du Web des Données. En effet (i) l'agencement des ressources entre elles n'est généralement pas connu d'avance ; (ii) les identifiants des ressources ne sont en principe pas connus (les IRI n'ont qu'un rôle pratique d'identification unique).

Face à la difficulté d'écrire des requêtes SPARQL, il existe dans la littérature plusieurs outils d'aide à l'*exploration* du Linked Data sans la production de requêtes SPARQL. On peut citer les travaux Visual data Web [6], Sparklis [4], ou encore Freebase Easy [1]. Hélas, ces approches limitent l'exploration des bases aux ressources préalablement indexées (et souvent stockées) en interne. Seule la plateforme Freebase Easy propose aussi une interface (API) d'*exploitation* des données mais de manière *ad hoc* non SPARQL. On notera enfin que la plateforme Sparklis renseigne la requête SPARQL théorique associée à l'exploration en cours des données.

Notre contribution est double, et est exploitable de manière centralisée au sein d'une plateforme : `universal-endpoint.com`. D'une part, nous permettons l'écriture de pseudo requêtes SPARQL, que nous appelons requêtes *SimplePARQL*, plus intuitives à écrire que des requêtes SPARQL (ces dernières sont aussi acceptées), sans nécessiter une connaissance préalable des bases interrogées. D'autre part nous permettons la diffusion automatique d'une requête aux différentes bases RDF du Linked Open Data.

1. *Internationalized Resource Identifier*
2. *Sparql Protocol and RDF Query Language*
3. World Wide Web Consortium, `http://www.w3.org/`.

Cet article est organisé comme suit. La Section 2 présente notre approche et la plateforme permettant l'écrire intuitive des requêtes SimplePARQL. La Section 3 conclue cet article.

2 L'approche SimplePARQL

Dans les bases RDF, les ressources sont décrites par leurs liens aux autres ressources et leurs liens à des valeurs littérales. La sémantique des bases RDF est donc contenue dans ces relations. Cependant, il y a un écart entre la représentation structurée que perçoit l'utilisateur et celle physiquement présente dans une base RDF. Par exemple, l'information (Paris, capitale-de, France) pour l'utilisateur est concrètement stockée sous la forme (ressource1,propriété2,ressource3), (ressource1,label,"Paris"), (ressource3,label,"France"), (propriété2,label,"capitale-de"). Le but de notre approche est de réduire cet écart. Avec SimplePARQL les utilisateurs construisent des requêtes structurées (toujours en triplets) d'une façon intuitive et sans nécessiter une connaissance préalable du vocabulaire de la base et des IRI.

Supposons que nous souhaitons, via le endpoint SPARQL de *DBpedia*, connaître les peintres nés entre le `01-01-1700` et le `01-01-1900` en Allemagne et éventuellement ceux en plus décédés en Allemagne. La requête SPARQL suivante permet d'y arriver (résultats en FIGURE 1) [4].

```
Requête 1 :
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX dbr:<http://dbpedia.org/resource/>
SELECT DISTINCT *
WHERE {
 ?x rdf:type dbo:Painter.
 ?x dbo:birthDate ?birth.
 ?x dbo:birthPlace ?cityBirth.
 ?cityBirth dbo:country dbr:Germany.
 OPTIONAL {
  ?x dbo:deathPlace ?cityDeath.
  ?cityDeath dbo:country dbr:Germany. }
 FILTER(str(?birth)>="1700-1-1"&&str(?birth)<"1900-1-1")
}
```

On constate sur cet exemple que sans connaître l'IRI des ressources `<http://dbpedia.org/resource/Painter>`, `<http://dbpedia.org/ontology/birthDate>`, *etc.*, l'utilisateur aurait du mal à formuler sa requête SPARQL ; et d'autant plus de mal avec d'autres bases RDF du Linked Data où les IRI n'ont pas de signification et sont juste des identifiants.

2.1 Des requêtes proches de l'humain

SimplePARQL fournit un système d'interrogation structuré en triplets des bases RDF en utilisant des requêtes plus naturelles et intuitives pour l'utilisateur. L'idée est de permettre aux utilisateurs de raisonner sur des données concrètes – du texte, des nombres, des dates, *etc.* – contenues au sein des littéraux, et rarement dans les IRI. Ainsi l'utilisateur peut écrire ses requêtes comme une conjonction de triplets, mais où cette fois-ci une ou plusieurs composantes d'un triplet peut être autre chose qu'une variable, un IRI, un nœud blanc, ou un littéral. Nous avons appelé ce nouvel élément de la requête une *ressource imprécise*.

Concrètement, notre système « élargit » la requête SimplePARQL pour permettre de faire coïncider les *ressources imprécises* aux données concrètes associées aux ressources recherchées. Ainsi la requête SimplePARQL d'origine est transformée en N requêtes SPARQL possibles pour faire correspondre les *ressources imprécises* avec des ressources pertinentes dans la base RDF. La FIGURE 2 illustre notre approche et l'écart d'interprétation entre l'utilisateur et la structure physique en base.

Reprenons l'exemple de la requête SPARQL précédente "Requête 1", mais maintenant écrite en SimplePARQL :

```
Requête 2 :
SELECT *
WHERE { ?x a Painter
?x (birth Date) ?birth
?x (birth Place) "Germany"
OPTIONAL {?x (death Place) "Germany"}
FILTER(str(?birth)>="1700-1-1" &&str(?birth)<"1900-1-1")
}
```

Cette requête contient finalement une seule variable, `?x` (l'objectif de la requête), et cinq *ressources imprécises* : Painter, (birth Date), (birth Place), (death Place) et "Germany". Les résultats d'une requête SimplePARQL sont présentés avec les justifications de "matching" des *ressources imprécises* sur des données concrètes, comme cela présenté en FIGURE 3.

Une requête SimplePARQL peut être composée de mots-clés SPARQL, de variables précédées par le symbole ' ?', d'IRI entre deux chevrons (< ... >), d'un IRI court [5], de l'élément `a` [6], de nœuds blancs, de littéraux entre double quotes, et de *ressources imprécises*. Une *ressource imprécise* peut être :

1. **Un mot unique** (*e.g.* Peinter), ce qui signifie que toutes les ressources dans la base interrogée qui sont liées à un littéral contenant ce mot doivent être retournées.
2. **Plusieurs mots entre parenthèses** (*e.g.* (birth Place)), toutes les ressources liées à un littéral possédant ces mots (ordre et casse insensibles) doivent être retournées.
3. **Un ou plusieurs mots entre double quotes** (*e.g.* "Germany"), toutes les ressources liées à cette expression exacte doivent être retournées [7].

2.2 Différents cas de conversion en SPARQL

Dans notre approche, une requête SimplePARQL est réécrite en un ensemble de requêtes SPARQL offrant la possibilité de matcher la *ressource imprécise* avec des ressources RDF dans la base interrogée. En fonction de la position de la *ressource imprécise* dans le triplet, des

4. Pour plus d'informations sur le langage SPARQL `https://www.w3.org/TR/sparql11-overview/`

5. Utilisation du mot-clé `PREFIX` pour les IRI courts

6. a ⇔ rdf:type ⇔ http://www.w3.org/1999/02/22-rdf-syntax-ns#type

7. Dans ce cas, si le *ressource imprécise* est en position d'objet, il est équivalent à un littéral en SPARQL.

x	birth	cityBirth	cityDeath
dbr:Johann_Christian_Eberlein	"177"	dbr:Göttingen	dbr:Göttingen
dbr:Hanns_Bolz	"1885-01-22"	dbr:Aachen	dbr:Munich
dbr:Ludwig_von_Löfftz	"1845-06-21"	dbr:Darmstadt	
dbr:Paul_Weber_(artist)	"1823"	dbr:Darmstadt	dbr:Munich
...			

FIGURE 1 – Extrait des résultats à la requête SPARQL "Requête 1" en interrogeant *DBpedia*.

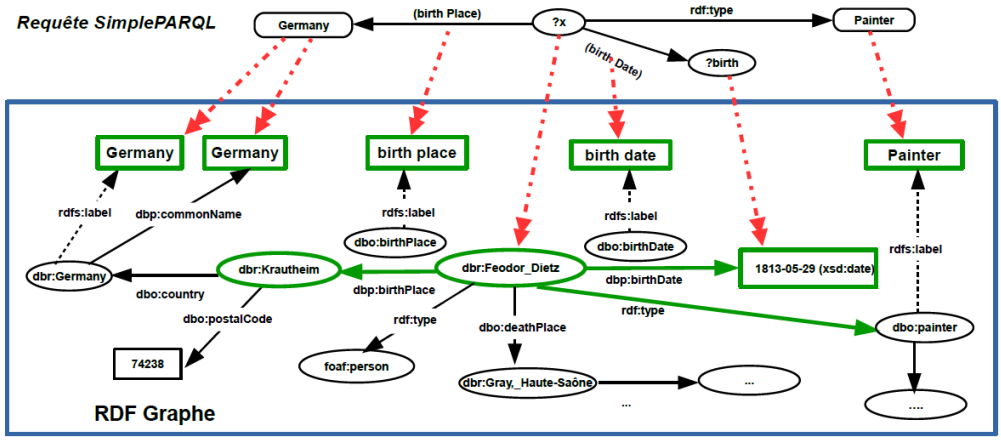

FIGURE 2 – Matching entre une requête SimplePARQL et la partie retournée dans la base RDF.

x	such as Painter is	such as (birth Date) is	birth	such as (birth Place) is	such as "Germany" is	such as (death Place) is	such as "Germany" is
http://dbpedia.org/resource/Hermann_Nestel	http://dbpedia.org/ontology/Painter **because has label** "painter"@en	http://dbpedia.org/property/birthDate **because has label** "birth date"@en	"1858-05-10"^^http://www.w3.org/2001/XMLSchema#date	http://dbpedia.org/property/birthPlace **because has label** "birth place"@en	"Stuttgard, Germany"@en		
http://dbpedia.org/resource/Otto_Ackermann_(painter)	http://dbpedia.org/ontology/Painter **because has label** "painter"@en-gb	http://dbpedia.org/property/birthDate **because has label** "birth date"@en	1872	http://dbpedia.org/property/birthPlace **because has label** "birth place"@en	"Berlin, Germany"@en	http://dbpedia.org/property/placeOfDeath **because has label** "PLACE OF DEATH"@en	"Düsseldorf, Germany"@en
http://dbpedia.org/resource/Wihelm_Amberg	http://dbpedia.org/ontology/Painter **because has label** "painter"@en	http://dbpedia.org/property/birthDate **because has label** "birth date"@en	"1822-02-25 "^^http://www.w3.org/2001/XMLSchema#date	http://dbpedia.org/property/birthPlace **because has label** "birth place"@en	"Berlin, Germany"@en		
...	...	...	...	...	...	...	...

FIGURE 3 – Extrait des résultats de la requête SimplePARQL "Requête 2" en interrogeant *DBpedia*.

requêtes SPARQL sont générées en se basant sur les règles de réécriture suivantes :

1. La *ressource imprécise* est en position du sujet ⇒ réécriture en trois requêtes SPARQL.

Exemple :
```
SELECT ?objet ?relation
WHERE { (Feodor Dietz) ?relation ?objet. }
```

Cas 1.1 : Chercher toutes les ressources liées par rdfs:label à un littéral contenant la *ressource imprécise*.
```
SELECT ?objet ?relation
WHERE {
?RessourceImp ?relation ?objet.
?RessourceImp rdfs:label ?label.
FILTER(CONTAINS(UCASE(STR(?label)),UCASE("Feodor"))
&& CONTAINS(UCASE(STR(?label)),UCASE("Dietz"))) }
```

Cas 1.2 : Chercher les ressources liées par une propriété (pas uniquement rdfs:label) à un littéral contenant la *ressource imprécise*.
```
SELECT ?objet ?relation
WHERE {
?RessourceImp ?relation ?objet.
?RessourceImp ?tmp_var1 ?tmp_var2
FILTER(CONTAINS(UCASE(STR(?tmp_var2)),UCASE("Feodor"))
&& CONTAINS(UCASE(STR(?tmp_var2)),UCASE("Dietz"))) }
```

Cas 1.3 : Chercher toutes les IRI contenant la *ressource imprécise* (à du sens avec DBpedia)
```
SELECT ?objet ?relation
WHERE {
?RessourceImp ?relation ?objet.
FILTER(CONTAINS(UCASE(STR(?RessourceImp)),UCASE("Feodor"))
&& CONTAINS(UCASE(STR(?RessourceImp)),UCASE("Dietz"))) }
```

2. La *ressource imprécise* est en position de prédicat ⇒ réécriture en deux requêtes SPARQL.

```
Exemple : SELECT ?sujet ?objet
          WHERE { (?sujet (birth Place) ?objet }
```

Cas 2.1 : Chercher toutes les propriétés liées par rdfs:label à un littéral contenant la *ressource imprécise*.

```
SELECT ?sujet ?objet
WHERE {
?sujet ?RessourceImp ?objet.
?RessourceImp rdfs:label ?label.
FILTER(CONTAINS(UCASE(STR(?label)),UCASE("birth"))
&& CONTAINS(UCASE(STR(?label)),UCASE("Place"))) }
```

Cas 2.2 : Chercher toutes les IRIs contenant la *ressource imprécise* (à du sens avec DBpedia)

```
SELECT ?sujet ?objet
WHERE {
?sujet ?RessourceImp ?objet.
FILTER(CONTAINS(UCASE(STR(?RessourceImp)),UCASE("birth"))
&& CONTAINS(UCASE(STR(?RessourceImp)),UCASE("Place"))) }
```

3. La *ressource imprécise* est en position d'objet ⇒ réécriture en trois requêtes SPARQL.

```
Exemple : SELECT ?sujet ?objet
          WHERE { (?sujet ?relation "Germany" }
```

Cas 3.1 : Chercher les littéraux (et les IRI pour *DBpedia*) contenant la *ressource imprécise*

```
SELECT ?sujet ?relation
WHERE {
 ?sujet ?relation ?RessourceImp.
 FILTER(CONTAINS(STR(?RessourceImp),"Germany")) }
```

Cas 3.2 : Chercher toutes les ressources liées par rdfs:label à un littéral contenant la *ressource imprécise*.

```
SELECT ?sujet ?relation
WHERE {
?sujet ?relation ?RessourceImp.
?RessourceImp rdfs:label ?label.
FILTER(CONTAINS(STR(?label),"Germany")) }
```

Cas 3.3 : Chercher les ressources liées par une propriété (pas uniquement rdfs:label) à un littéral contenant la *ressource imprécise*.

```
SELECT ?sujet ?relation
WHERE {
?sujet ?relation ?RessourceImp.
?RessourceImp ?tmp_var1 ?tmp_var2.
FILTER(CONTAINS(STR(?tmp_var2),"Germany")) }
```

Si une requête SimplePARQL est composée de plusieurs *ressources imprécises* et\ou de plus d'un triplet, toutes les combinaisons possibles des cas de réécriture sont effectuées. Exemple, si un triplet est composé de deux *ressources imprécises*, le premier en position de sujet et le second en position de prédicat, alors le nombre de réécritures est de six : trois requêtes pour la réécriture du sujet, deux requêtes pour la réécriture du prédicat.

3 Conclusion et travaux futurs

Face à la difficulté d'interroger les bases RDF du Web des Données par des requêtes SPARQL non triviales à écrire, du fait d'une organisation complexe entre les ressources et de leurs manipulations via des IRI, nous proposons la plateforme `universal-endpoint.com`. Avec cette plateforme, l'utilisateur a la possibilité d'interroger de manière intuitive des bases du Linked Open Data – sans nécessiter d'en être un expert – en rédigeant des requêtes SimplePARQL pouvant contenir des *ressources imprécises* en plus des éléments du langage SPARQL.

Concrètement, l'interrogation d'une requête SimplePARQL se fait via une réécriture transparente en plusieurs requêtes SPARQL à travers les endpoints des bases du Linked Data. Ces multiples interrogations SPARQL, nécessaires pour déterminer les éléments imprécis de la requête SimplePARQL formulée par l'utilisateur, ont un impact direct sur le temps d'exécution. Cependant aujourd'hui, des travaux récents comme le framework [8] permettent de réduire fortement les temps d'attente face à de multiples requêtes sur les serveurs SPARQL des bases du Linked Data.

Le endpoint SimplePARQL disponible sur notre plateforme n'offre pas aujourd'hui toute l'expressivité de SPARQL. Si cela était un choix délibéré au départ en terme de simplification d'écriture de requêtes, il s'avère à l'usage que c'est une limitation qu'il faut lever pour l'utilisateur expert du Web des Données.

Références

[1] Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haußmann. Easy access to the freebase dataset. In *Proc. of WWW'14 Companion*, pages 95–98, 2014.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 279(5) :34–43, 2001.

[3] Sonia Djebali and Thomas Raimbault. Simpleparql : A new approach using keywords over sparql to query the web of data. In *Proc. of SEMANTICS'15*, pages 188–191. ACM, 2015.

[4] Sébastien Ferré. Sparklis : a SPARQL Endpoint Explorer for Expressive Question Answering. In *Proc. of ISWC Posters & Demonstrations Track*, 2014.

[5] G. Klyne, J. J. Carroll, and B. McBride. RDF 1.1 Concepts and Abstract Syntax, 2014. `http://www.w3.org/TR/rdf-concepts/`.

[6] Steffen Lohmann, Philipp Heim, Timo Stegemann, and Jürgen Ziegler. The relfinder user interface : Interactive exploration of relationships between objects of interest. In *Proc. of Intelligent User Interfaces (IUI'10)*, pages 421–422. ACM, 2010.

[7] E. Prudhommeaux and A. Seaborne. SPARQL Query Language for RDF, 2008. `www.w3.org/TR/rdf-sparql-query/`.

[8] R. Verborgh, M. Vander Sande, O Hartig, J Van Herwegen, L De Vocht, B De Meester, G Haesendonck, and P Colpaert. Triple Pattern Fragments : a low-cost knowledge graph interface for the Web. *Journal of Web Semantics*, 37–38 :184–206, March 2016.