# Langage de recherche d'associations sémantiques à partir d'une base de connaissances

**Thabet Slimani – Boutheina Ben Yaghlane – Khaled Mellouli**

*Laboratoire LARODEC, ISG de Tunis*
*41 rue de la liberté, Bouchoucha Bardo 2000, Tunisia*
*thabet.slimani@issatm.rnu.tn, {boutheina.yaghlane, khaled.mellouli}@ihec.rnu.tn*

RÉSUMÉ. *Dans le cadre de l'extraction d'associations sémantiques (liens complexes/chemins), nous définissons des requêtes d'extraction des chemins comme sorte d'extraction des relations complexes dans un graphe RDF reliant deux entités spécifiées. Plusieurs besoins d'extraction des chemins peuvent être formulés. Néanmoins, les langages d'interrogation RDF existants n'offrent pas de mécanisme adéquat pour formuler des requêtes destinées à l'extraction des chemins. Afin de pallier à cette défaillance, nous proposons un nouveau langage (PmSPARQL) permettant d'étendre le langage d'interrogation actuel de W3C (SPARQL). Les principaux objectifs de PmSPARQL consistent à inclure des possibilités d'extraction d'associations sémantiques et pouvoir les évaluer. Une évaluation des performances de notre approche sur des données du monde réel et synthétiques est également présentée.*

ABSTRACT. *In the context of Semantic Association extraction, we define path extraction queries as paths in RDF graph between two entities which represent a kind of associations between them. According to the needs of path extraction queries, such requests can extract paths which connect a source entity with target entity. Several paradigms of path extractions can be formulated. Nevertheless, the current RDF query languages have the lack to formulate requests intended for the extraction of semantic associations. In this paper, we propose the PmSPARQL query language, which extends the current W3C query language SPARQL. We compare PmSPARQL with some other query languages which have similar goals*

MOTS-CLÉS : *Association sémantique, langage de requêtes, requête sémantique, extraction des chemins, liens complexes.*

KEYWORDS: *Semantic Association, language request, semantic request, path extraction, complex links.*

## 1. Introduction

Les travaux qui portent sur la recherche sémantique visent à améliorer les méthodes conventionnelles de recherche et d'extraction des informations et à faciliter l'acquisition, le traitement, le stockage et la découverte des informations dans le domaine du Web sémantique. La recherche sémantique de l'information complète la recherche de l'information conventionnelle en fournissant des services de recherche centrés sur des entités, des relations et des connaissances. En plus, le développement du Web sémantique exige également des paradigmes améliorés de recherche afin de faciliter l'extraction des informations sémantiques. Les méthodes de recherche actuelles du Web sémantique étalent la portée du paradigme de recherche documentaire traditionnel de la simple recherche documentaire à l'extraction des entités, des liens complexes (à travers l'extraction des relations indirectes ou des chemins dans un graphe RDF) et des connaissances. Les techniques de recherche d'associations sémantiques accordent un intérêt important pour répondre à des requêtes de type "existe t'il un rapport sémantique entre deux entités A et B ?". Une association sémantique se rapporte à un lien indirect qui relie deux entités contenues dans une base de connaissances (Aleman-Meza *et al.,* 2003)(Kemafor *et al.,* 2005)(Boanerges *et al.,* 2005). L'origine des associations sémantiques est apparue avec les théories et les méthodes de recherche inventées par le laboratoire LSDIS à l'université de la Géorgie[1].

Les langages d'interrogation à partir des bases de données RDF, telles que RDQL (Seaborne, 2004) et son successeur SPARQL (Prud'hommeaux *et al.,* 2005) aident à la définition des motifs de graphe et permettent des restrictions concernant des entités et/ou des propriétés qui participent à certains modèles définis. Néanmoins, aucun support n'est fourni par ces langages en ce qui concerne l'extraction des chemins ou des associations sémantiques dans des graphes RDF. Récemment, deux langages proposés (Kemafor *et al.,* 2007) et (Kochut *et al.,* 2007) s'intéressant à l'extraction des chemins dans un graphe RDF ont été développés. Notre langage proposé, dénommé PmSPARQL, s'inspire de la combinaison des avantages de ces deux langages. En plus, PmSPARQL, qui est considéré comme une extension du langage d'interrogation courant de W3C (SPARQL), vise à identifier des relations complexes aussi bien que l'évaluation des chemins produits. Nous discutons quelques problèmes de complexité d'exécution pour ce langage. De même, nous comparons notre langage proposé avec certains langages d'interrogation qui ont des buts semblables. La syntaxe et la sémantique de notre langage sont présentées dans le travail de (Slimani *et al.,* 2008).

Le reste de ce document est organisé comme suit. La section 2 présente un état de l'art sur les langages de requêtes du Web sémantique. La section 3 définit le principe d'une association sémantique. Les objectifs du langage PmSPARQL, ainsi qu'un exemple d'application dans des domaines du monde réel sont présentés dans la section 4. La section 5 présente des exemples d'application pour des requêtes d'associations sémantiques, ainsi qu'une comparaison avec d'autre langages. Les résultats expéri-

1. http ://lsdis.cs.uga.edu/

mentaux sont présentés dans la section 6. Finalement, la section 7 présente un résumé et les perspectives des travaux futurs.

## 2. Etat de l'art sur les langages de requêtes du Web sémantique

L'objectif principal du Web sémantique est de permettre aux machines d'exploiter effectivement les connaissances contenues dans le Web d'une manière décentralisée. Pour atteindre ce but, le contenu (données) du Web sémantique est accompagné des métadonnées. Les métadonnées offrent des informations concernant le contenu afin d'aider les applications à gérer et traiter leur contenu, par exemple en capturant des informations sur la spécification sémantique de certaines données. Le standard RDF (Resource Description Framework) inventé par W3C qui sert comme langage de métadonnées pour le Web sémantique et son extension OWL (Web Ontology Language) constituent les langages de base du Web sémantique. Afin d'employer des métadonnées pour l'interopérabilité entre applications, il n'est pas suffisant d'avoir uniquement un langage de description des métadonnées. Un langage pour décrire des requêtes sur les métadonnées est également nécessaire. Plusieurs travaux de recherches que se soit industriels ou académiques sont actuellement impliqués en proposant plusieurs langages d'interrogation des métadonnées RDF. Nous classifions ces langages en cinq catégories : Langages SQL-Like, langages de requêtes fonctionnels, des langages basées sur les règles, langages de parcours des graphes et des langages d'extraction des chemins :

– langages SQL-Like : SPARQL(Prud'hommeaux *et al.,* 2005), RDQL(Seaborne, 2004), SeRQL(Broekstra *et al.,* 2003) and Inkling, Squish[2].

– langage de requêtes fonctionnels : RQL(Karvounarakis *et al.,* 2002).

– langages basés sur les règles : Triple(Sintek *et al.,* 2001),RuleML[3] N3[4] et Inkling.

– langages de parcours de graphes : RDFQ[5], Versa[6], RxPath (Souzis, 2004) (est un langage de requêtes basé sur XPath), G, G+, Gram.

– Langages d'extraction des chemins : SPARQ2L (Anyanwu *et al.,* 2007), SPARQLer (Kochut *et al.,* 2007), SPARQL-ST (Perry *et al.,* 2008) et PSPARQL (Faisal *et al.,* 2005).

PSPARQL est une extension de SPARQL qui permet l'appariement flexible des graphes en se basant sur des bases de données RDF. Ce langage manque des caractéristiques des chemins variables. Notre langage PmSPARQL est classifié sous les langages d'extraction des chemins. Les langages de requêtes qui ont des buts plus si-

2. http ://ilrt.org/discovery/2001/02/squish/
3. http ://www.ruleml.org/
4. http ://www.w3.org/DesignIssues/Notation3.html
5. http ://sw.nokia.com
6. http ://www.xml.com/lpt/a/1601

milaires à PmSPARQL sont SPARQLer et SPARQ2L. La différence entre ces deux langages et PmSPARQL sera présentée dans la section 5.3.

## 3. Association sémantique

RDF (Resource Description Framework) est basé sur un modèle du triplet (sujet : propriété : objet) dont les ressources (sujet/objet :littéraux ou bnodes (blank nodes)) et les propriétés combinées ensemble forment un triplet. Chaque élément (sujet ou objet) du triplet constitue une entité ou une ressource d'un graphe RDF. Une propriété (predicate/IRI) est définie comme relation binaire entre deux entités (ressources). Soit I, B et L (IRIs, blank nodes, et littéraux) des paires disjointes d'ensembles finis. Le triplet (sujet, prédicat, objet)$\in$(I$\cup$B)xIx(I$\cup$B$\cup$L) est appelé triplet RDF. Un graphe RDF (RDF dataset) est un ensemble de triplets RDF. Nous définissons une association sémantique comme un chemin direct ou indirect reliant deux entités contenues dans un graphe RDF incluant une série de relations explicites. Nous présentons dans ce qui suit les formalismes qui définissent la distinction entre une association sémantique directe et indirecte.

**Definition 1** *Chemin direct :*

*Deux entités $e_0$ et $e_n$ sont sémantiquement connectées via une association directe s'il existe un chemin direct qui les relient de la forme suivante : ($e_0 \xrightarrow{p_1} e_1 \xrightarrow{p_2} e_2 ... \xrightarrow{p_{n-1}} e_{n-1} \xrightarrow{p_n} e_n$).*

où $e_i$ ($1 \leq i \leq n$) est une entité d'ordre i et $P_j$($1 \leq j \leq n$) est une propriété d'ordre j. L'ensemble des propriétés $p_1$, $p_2$,..., $p_{n-1}$ représente *une séquence des propriétés* (SP) connectant *la séquence des entités* SE ($e_1$,$e_2$,..., $e_{n-1}$, $e_n$). La longueur de l'association n désigne le nombre des triplets dans l'association. Dans ce travail, nous exploitons uniquement les associations directes simples (les entités qui apparaissent dans une association sémantique sont distinctes) pour éviter des boucles infinies.

**Definition 2** *Chemin indirect : Deux entités $e_0$ et $e_n$ sont indirectement associées s'il existe une connectivité sémantique entre elles, indépendamment de la direction des propriétés (arcs menant de $e_0$ à $e_n$).*

L'exemple de la figure 1 montre une association sémantique indirecte entre deux entités de domaine génomique "Cockayne SyndromeXeroderma" et "Li-Fraumeni Syndrome".

## 4. Présentation du langage PmSPARQL

PmSPARQL est une extension de SPARQL conçu pour découvrir des associations sémantiques contenues dans des données sémantiques (ontology OWL, RDF graph).

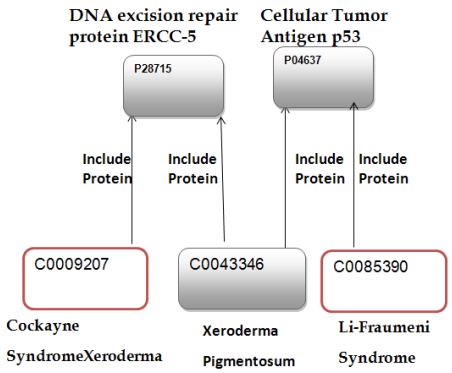


**Figure 1.** *Exemple d'associations sémantiques indirectes.*

L'approche est définie par deux caractéristiques fondamentales. Premièrement, les requêtes de PmSPARQL sont entièrement compatibles avec des requêtes de SPARQL. Deuxièmement, une requête PmSPARQL permet l'extraction d'associations en se basant sur quelques critères spécifiques définis par un utilisateur.

L'extraction d'associations sémantiques par PmSPARQL est basée sur la construction d'un motif de chemin (Path Pattern : motif de triplet créé avec l'utilisation d'une variable de chemin (path variable) à la place de la variable de propriété adoptée dans le langage SPARQL) incluant des chemins directs ou indirects. La syntaxe de PmSPARQL présente une modification légère de la grammaire du langage SPARQL. Les nouvelles fonctionnalités ajoutées par PmSPARQL rendent les opérations suivantes possibles :

– Localisation des chemins directs entre deux entités spécifiées (intérêt pour la découverte d'association sémantiques sans spécification de la longueur ou d'importance).

– Localisation des chemins à travers le filtrage des associations sémantiques à l'aide des expressions régulières concernant les propriétés intermédiaires d'une d'association.

– Localisation des chemins en imposant des contraintes sur la longueur du chemin.

– Identification de tous les chemins qui incluent la présence ou l'absence d'une propriété au niveau d'une position définie dans l'association sémantique.

– Identification de tous les chemins ordonnés (directs) en évaluant les associations sémantiques selon un critère spécifique à l'utilisateur (richesse, importance,...etc. Voir (Slimani *et al.,* 2008)).

Les objectifs derrière le développement du langage PmSPARQL se présentent comme suit :

– Soutenir la formulation d'expressions que ce soit par des requêtes traditionnelles d'extraction des entités ou des requêtes d'extraction des chemins.

– Eviter le développement d'un nouveau langage à partir de zéro mais plutôt en se basant sur les langages de requêtes existants et qui sont déjà validés et des méthodologies ayant un certain aspect standardisé (SPARQ2L et SPARQLeR).

– Consolider l'expressivité d'un langage de requêtes, ainsi que sa simplicité au niveau de la formulation des requêtes.

– Formuler des expressions de requêtes permettant facilement l'évaluation des chemins extraits, quand l'utilisateur est confronté à un ensemble de chemins limités (rangement ou restriction des chemins).

## 5. Requêtes d'associations sémantiques avec PmSPARQL

A cause d'une limitation d'espace, le détail concernant la syntaxe et la sémantique du langage PmSPARQL est présenté dans (Slimani *et al.,* 2008). Néanmoins, nous rappelons quelques définitions utiles pour la compréhension du reste du papier.

**Definition 3** *Motif de triplet. Un motif de triplet est un triplet où un ou plusieurs éléments peuvent être des variables : ( ?x, Prof :name, ?y).*

**Definition 4** *Motif de graphe. Un motif de graphe est un ensemble de motifs de triplets combinés en utilisant certains opérateurs binaires AND, FILTER, UNION, OPTIONAL.*

A titre d'exemple, l'expression ((( ?x, Prof :name, ?y) AND ( ?x, Prof :age, ?z)) FILTER ( ?z > 60)) est un exemple de motif de graphe utilisant les opérateurs AND et FILTER. D'une manière similaire, l'expression d'un motif de triplet-chemin est semblable au motif de triplet décrit dans SPARQL (Pérez *et al.,* 2006) sauf qu'au niveau de la position de la propriété sera placée une entité de type chemin (expression d'une association sémantique : <subject : chemin : object>). Une *entité de type chemin* est une séquence de plusieurs entités RDF. Une séquence se compose d'une liste de propriétés non ordonnées reliant les entités du sujet (entité source) et l'objet (entité cible) constitue la définition de l'association sémantique présentée ultérieurement. Formellement, l'expression du motif de triplet-chemin se compose d'un ou plusieurs motifs de triplet de SPARQL (Pérez *et al.,* 2006) exprimée à travers une variable de chemin (*Path Variable* :PV) à la position d'attribut/propriété et certaines conditions intégrées pour le filtrage de chemin. Un élément dans l'ensemble PV est dénoté par p⊂@P (@P est une variable de chemin dont le nom est P et commençant toujours par le caractère @). Par exemple, le motif triplet-chemin suivant $\{<$?x$>$ : $\xrightarrow{advises}$St0$\xrightarrow{EnrolledIn}$C0$\xrightarrow{CourseIn}$ : $<$?y$>\}$ inclut un motif triplet-chemin avec deux variables, respectivement, dans les positions sujet et objet.

**Definition 5** *Motif de triplet-chemin. Nous dénotons par IL l'union I ∪ L, et par T l'union I ∪ B ∪ L. Un motif de triplet-chemin est une combinaison de 3 tuples : (IL∪V)×PV×(IL∪V).*

D'une façon semblable aux contraintes spécifiées sur des termes variables (langage SPARQL), il est possible de spécifier des contraintes sur des path-variables qui ont besoin de certaines spécificités additionnelles. Ces contraintes peuvent être exprimées par la fonction de construction des chemins contenus dans l'ensemble PV. Les expressions régulières à travers un motif de triplet sont décrites dans les travaux de (Faisal *et al.,* 2005) et (Kochut *et al.,* 2007). Un motif peut impliquer des noeuds/entités et/ou des arcs/propriétés.

Soit T l'ensemble de tous les motifs de triplets possibles et RT l'ensemble d'expressions régulières appliquées sur T. Soit RP et PV (respectivement, les chemins réguliers et les variables de chemin) deux paires d'ensembles qui sont disjoints de T. Un motif de triplet-chemin est un tuple dans $(IL \cup RP)\times(I \cup RP)\times(IL \cup RP)$.

Dans le paragraphe suivant, nous présentons quelques fonctions développées pour faciliter l'extraction d'associations sémantiques :

– MaxPropertyFrequence (PV, p) $\longrightarrow N$ : Retourne une valeur (type réel) d'association qui présente la fréquence maximale d'un prédicat/arc spécifié ("p"$\subset I$).

– Size (PV) $\longrightarrow N$ : est une fonction (type entier) qui détermine la taille d'une association sémantique (nombre de propriétés/arcs).

– MatchResource (PV, R$e$) $\longrightarrow$boolean : Cette fonction retourne si une ressource specifiée par une expression régulière sur un triplet est contenue dans une association sémantique ou non ( $Re\subset$IL).

– MatchPattern (PV, *RT*) $\longrightarrow$boolean : Cette fonction recherche l'existence des chemins qui incluent un motif de triplet *RT*.

– MatchProperty (PV, p, pos) $\longrightarrow$boolean : Cette fonction retourne si une propriété/arc est contenu dans une association sémantique au niveau d'une position pos.

– MatchProperty (PV, *p*+) $\longrightarrow$boolean : Cette fonction retourne si un chemin inclut une propriété/arc p bien spécifique.

– MatchAdjEdges(@P,R$e$)$\longrightarrow$String : Cette fonction retourne tous les chemins incluant des propriétés entrantes pour une ressource R$e$.

– MatchNodeDegree(@P,R$e$)$\longrightarrow N$ : Cette fonction retourne le nombre d'arcs/-prédicats qui impliquent une ressource R$e$ dans une association sémantique.

– MatchAdjNodes(@P,R$e$)$\longrightarrow$String : Cette fonction retourne tous les noeuds/ressources adjacents à une ressource/noeud R$e$.

– MinDistance(@P) $\longrightarrow N$ : Cette fonction retourne la distance la plus courte entre deux entités/noeuds (nombre d'arcs).

– RichnessRank(@P, asc/desc)$\longrightarrow$String, ConnectivityRank(@P,asc/desc)$\longrightarrow$String, ImportanceRank(@P, asc/desc)$\longrightarrow$String, FrequenceRank(@P, p)$\longrightarrow$String : Chaque fonction retourne des chemins rangés selon le choix de rangement spécifié.

– IndPath(PV,[*p1-p2*]+)⟶boolean : Cette fonction retourne des chemins indirects incluant une paire des séquences de propriétés : *p1* suivie par l'inverse de la propriété *p2*.

Le motif de graphe de PmSPARQL se compose des motifs de triplet-chemin combinés en utilisant les opérateurs standard de composition de SPARQL AND, FILTER, OPTIONAL, UNION et PFILTER. Le nouveau opérateur PFILTER permet la formulation des expressions de filtrage de chemin c-à-d des contraintes sur des variables de chemin.

### 5.1. *Les requêtes PmSPARQL par l'exemple*

Dans cette section, nous présentons la syntaxe de quelques requêtes concrètes de PmSPARQL par des exemples :

**Query1** (Requête simple d'extraction des chemins) : La requête de sélection suivante, implique l'extraction de toutes les associations sémantiques possibles entre deux noeuds (deux publications) bien fixés :

```
SELECT @P WHERE
{
<http://dblp.uni-trier.de/rec/bibtex/books/aw/AbiteboulH
V95>  @P
<http://dblp.uni-trier.de/rec/bibtex/conf/vldb/AroraC78>
};
```

**Query2** (Requête définie pour un chemin dont on spécifie une ressource unique) : Identifier tous les chemins connectant l'entité s="AbiteboulHV95" avec certaines autres ressources/entités inconnues :

```
SELECT @P WHERE
  {<http://dblp.uni-trier.de/rec/bibtex/books/aw/Abitebou
  lHV95>
   @P  ?x};
```

Cette requête définie une requête de chemins avec une source unique qui localise des associations sémantiques à partir d'une source s. La forme analogue de la requête précédente est reliée à l'inverse de la forme du motif de graphe extensible { ?x @P <r>}. Ce motif recherche tous les chemins dirigés à partir de la source s.

**Query3** (Requête des chemins avec une contrainte sur les noeuds intermédiaires) : Trouver tous les chemins entre deux ressources inconnues ?x and ?y, étant donné le noeud intermédiaire ?e. Dans l'exemple de la requête Query3, le noeud intermédiaire représente une molécule biologique "cbo.400.8q21.1" reliée à travers l'arc BAND-OF :

```
SELECT @P WHERE
{?x @P ?y
  ?e cbo:BAND-OF <cbo.400.8q21.1>
  PFilter (MatchResource(@P,?e)};
```

**Query4** (Requête des chemins pour des associations sémantiques indirectes) : Les chemins localisés ne peuvent pas être entièrement orientés, mais guidés par différentes propriétés spécifiques. Ceci peut être formulé par une expression appropriée au niveau de la variable de chemin, comme dans l'exemple de la requête suivante :

```
SELECT @P  WHERE
{
<r1> @P <r2>
PFilter(size(@P)>=6 AND size(@P)<=8 AND IndPath(@P,
[p1-P2]+))};
```

### 5.2. *Complexité de calcul de PmSPARQL*

La complexité de calcul du langage de requêtes SPARQL est présentée dans (Pérez *et al.,* 2006). En général, le problème de trouver des chemins dans des graphes simples est difficile et certaines requêtes construites avec PmSPARQL ont également cette propriété. Nous commenterons la complexité de calcul en se basant sur quelques fonctions proposées :

1) Pour l'extraction des chemins avec la fonction *MatchAllProp(p1,p2,..,pn)*, les requêtes sont NP-Hard avec une taille k. Alors, pour plusieurs applications, la taille k peut être beaucoup plus faible que le nombre des noeuds dans le graphe RDF qui le rendent pratique pour ces applications.

2) Pour l'extraction des chemins avec la fonction *MatchProperty(x1,x2,..,xn)* ou *MatchResource(x1,x2,..,xn)*, les requêtes posent un problème de complexité pour trouver des paires d'associations sémantiques disjointes à partir d'une entité source xi et de xi vers une entité de destination. Ce problème est également comparable au problème DISJOINTPATH connu par la complexité NP-Hard.

3) Pour l'extraction des chemins avec la fonction *MatchPattern(pattern)*, le problème de complexité est similaire aux résultats présentés dans (Milo *et al.,* 1999) où les auteurs présentent le problème de découverte des chemins simples imposés par certaines contraintes fixées par le langage. L'utilisation de ce type de requêtes pose une condition de temps polynômial, comme exprimé dans le travail (Milo *et al.,* 1999).

### 5.3. *Comparaison avec d'autres langages de requêtes*

La possibilité des requêtes appliquées sur des graphes flexibles au-dessus des bases de données RDF est envisageable par certains langages tels que RxPath, Versa, PSPARQL, SPARQLer (L1 dans tableau 1) et SPARQ2L (L2 dans tableau 1). Néanmoins, seulement SPARQLer et SPARQL2L sont destinés à l'extraction des chemins (associations sémantiques).

| Propriétés | G | G+ | Gram | RxPath | L2 | L1 | PmSPARQL |
|---|---|---|---|---|---|---|---|
| Noeuds adjacents | $\pm$ | $*$ | $*$ | $\times$ | $\times$ | $\times$ | $*$ |
| Arcs adjacents | $\pm$ | $*$ | $*$ | $\times$ | $\times$ | $\times$ | $*$ |
| Degré du noeud | $\times$ | $*$ | $\times$ | $\times$ | $\times$ | $\times$ | $*$ |
| Chemin orienté | $*$ | $*$ | $*$ | $\pm$ | $*$ | $*$ | $*$ |
| Distance entre 2 noeuds | $\times$ | $*$ | $\times$ | $\times$ | $*$ | $\times$ | $*$ |
| Diamètre d'un graphe | $\times$ | $*$ | $\times$ | $\times$ | $*$ | $\times$ | $\times$ |
| Chemin avec entité unique | ? | ? | ? | ? | $\times$ | $*$ | $*$ |
| Chemin non orienté | $\times$ | $\times$ | $\times$ | $\times$ | $\times$ | $*$ | $\pm$ |
| Degré d'un chemin | $\times$ | $\times$ | $\times$ | $\times$ | $\times$ | $\times$ | $*$ |
| Classer les chemins | $\times$ | $\times$ | $\times$ | $\times$ | $\times$ | $\times$ | $*$ |
| Chemin de longueur fixée | $*$ | $*$ | $*$ | $\pm$ | $*$ | $*$ | $*$ |

**Tableau 1.** *Capacité de certains langages de requêtes sémantiques vis à vis du support pour certaines propriétés ($*$ support complet, $\pm$ support partiel, $\times$ aucun support, et ? indique la non disponibilité de l'information).*

Il existe certaines différences entre notre approche et les approches décrites dans SPARQLer et SPARQ2L. La première, mais légère différence est que SPARQLer utilise le filtre standard de SPARQL (FILTER) pour exprimer des contraintes sur les variables de chemin. D'une façon semblable au langage SPARQ2L, nous avons opté pour la proposition d'une nouvelle catégorie de clause de filtrage (PFILTER) pour des variables de chemin afin d'augmenter les possibilités de formuler sémantiquement des expressions régulières. La plus importante contribution est exprimée par l'inclusion de nouvelles classes de requêtes qui peuvent être formulées uniquement avec PmSPARQL pour classer les chemins extraits. D'une manière similaire à SPARQLer, l'aptitude des requêtes pour l'extraction des chemins non orientés est actuellement possible avec PmSPARQL, mais cette caractéristique n'est pas permise dans le langage SPARQ2L. En plus, l'expression des requêtes d'extraction des chemins qui incluent des contraintes d'inclusion (motifs impliquant des entités et des propriétés) n'est pas permise dans SPARQLer qui présente seulement des contraintes au-dessus des propriétés, mais qui est possible par PmSPRQL et SPARQ2L. Pour illustrer la

| Dataset et kBase | Nombre de RDF Statement | Nombre de Classes | Nombre de relat-ions | Taille du fichier sur disque |
|---|---|---|---|---|
| SwetoDBLP-Aug-2007 | 305000 | 21 | 22 | 25M |
| CBO | 31893 | 4069 | 8463 | 3.02M |
| Glyco-0-9-full-enzyo | 15259 | 338 | 81 | 1.25M |
| SwetoDBLP-apr-2008-part1 | 610360 | 21 | 22 | 50M |

**Tableau 2.** *Propriétés des graphes RDF et bases de connaissances.*

comparaison des langages de requêtes appliqués sur des graphes RDF, nous avons choisi sept langages d'interrogation et onze propriétés de graphe fournies dans (voir (Angles *et al.,* 2004))(Tableau 1).

## 6. Résultats expérimentaux et évaluation

Dans cette section, nous décrivons l'implémentation de PmSPARQL à travers une évaluation empirique des requêtes formulées via notre approche ainsi qu'une évaluation des performances des requêtes d'extraction des associations sémantiques selon divers paradigmes. Les tests ont été appliqués sur une machine Intel (R) Core (TM) 2 DUO 2.2GHZ CPUS et 3 GB de mémoire.

### 6.1. *Dataset utilisé*

Nous avons adopté l'ensemble des données suivantes pour tester notre approche : SwetoDBLP-August-2007, SwetoDBLP-april-2008-part1[7], un extrait de CBO[8] et GlycO-0-9-full-enzyo[9].

Dans le Tableau 2, les propriétés de certains graphes RDF choisis sont présentées. Le graphe Glyco représente des informations sur des glycans et inclut un schéma complet aussi bien que des instances. Le graphe CBO (Clinical Bioinformatics Ontology) est un vocabulaire contrôlé et sémantiquement structuré pour des diagnostics moléculaires qui satisfait le besoin de la représentation consistante des entités moléculaires biologiques et cytogénétiques médicalement appropriées. Pour examiner l'implémentation à grande échelle de notre approche, il est utile d'employer un ensemble de données beaucoup plus large. Pour cela nous avons utilisé la version de l'ontologie disponible dans SwetoDBLP-april-2008-part1. Ces graphes RDF contiennent des informations sur des auteurs, des documents publiés, des articles, l'année de la pu-

7. http ://lsdis.cs.uga.edu/projects/semdis/swetodblp/
8. http ://clinbioinformatics.org/cbopublic
9. http ://lsdis.cs.uga.edu/projects/glycomics/2006/GlycO-0-9-full-enzyo.Owl

blication, etc. La propriété "*cite*" est utile pour dériver des associations sémantiques (chemins), mais elle est assignée pour un nombre réduit de publications (3067 occurrences), ce qui rend cet ensemble peu convenable pour le test d'évolutivité.

Pour pouvoir rechercher des associations sémantiques longues et significatives, nous avons ajouté à l'ensemble de données SwetoDBLP-april-2008-part1 une nouvelle liste de citations aléatoirement créées (1 à 12 citations aléatoires pour des papiers cités pendant les années ultérieurs dans la base de connaissances, en utilisant une distribution normale). Le nombre total des citations aléatoirement insérées dans l'ensemble de données SwetoDBLP-april-2008-part1 a atteint presque 1000000 triplets. L'ensemble de données SwetoDBLP-april-2008-part1 contient plus de 80657 publications éditées pendant des années spécifiques. Pour tester l'évolutivité de notre algorithme, nous avons employé des publications entre 1980 et 2008. Le test contient 78080 publications subdivisées en 28 sous-ensembles, le premier inclut des publications qui commencent en 2008 et le dernier inclut des publications dans l'intervalle [2007-1980] (le plus petit sous-ensemble inclus seulement les publications en 2008 (20 publications) et le plus grand inclus des publications pour les années dans l'intervalle [1980-2008] (78060 publications)). La figure 2.A présente le nombre de publications dans le graphe SwetoDBLPapril-2008 de chaque sous-ensemble relatif aux dates entre 1980 et 2008, en plus du pourcentage de chaque sous-ensemble en terme de certaines publications comparées à la totalité des publications (1980 à 2008).

### 6.2. *Test de performance sur les données de SwetoDBLP-april-2008-part1*

Nous évaluons l'exécution du langage PmSPARQL en observant deux propriétés :

– la taille des associations retournées c-à-d. le nombre de fp-expressions (expression finale d'un chemin obtenue par un processus de décomposition des triplets appropriés et successivement recherchés (t-expression) dans la base de données RDF) construites à partir des triplets du graphe RDF.

– la durée de traitement d'une requête qui dépend du coût de construction des fp-expressions.

Les instructions dans Q5 présentent un exemple d'une requête PmSPARQL avec une entité unique. Les résultats obtenus dans la figure 2 montrent que le nombre des chemins augmente exponentiellement d'une manière proportionnelle à l'accumulation des publications pour des années précédentes (figure 2.A).

```
Q5: SELECT @P WHERE
{<http://dblp.uni-trier.de/rec/bibtex/books/sp/Helmert2008> @P ?x
PFilter(Size(@P) <=28 AND MatchProperty(@P, "opus:cites"+))};
```

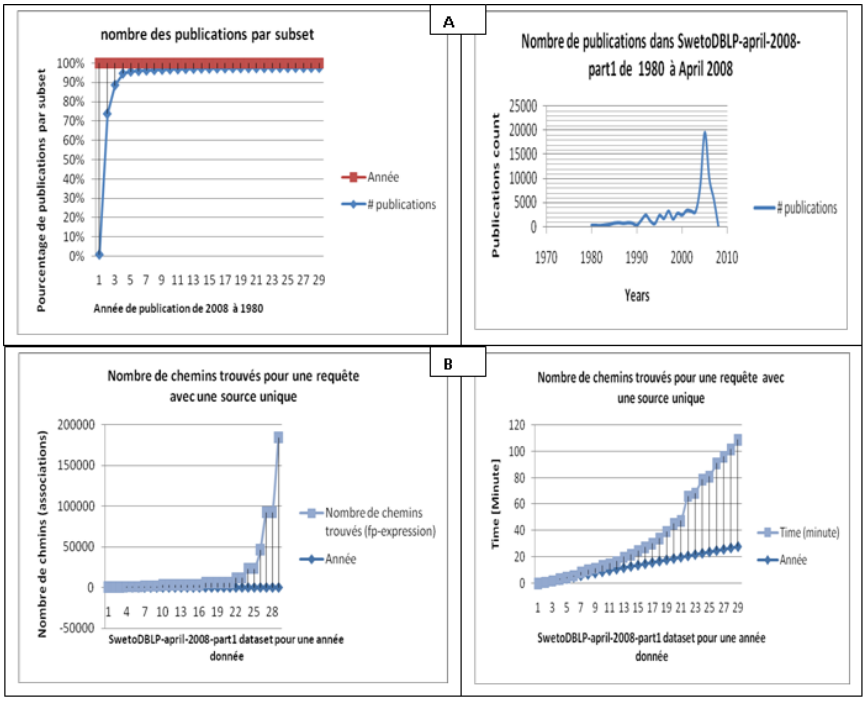


**Figure 2.** *A) Nombre et pourcentage de publications dans le graphe SwetoDBLP-april-2008-part1 entre l'année 1980 et 2008 ; B) Temps d'exécution et nombre des chemins trouvés dans SwetoDBLP-april-2008-part1 pour une requête PmSPARQL avec une entité source fixe et une entité cible variable.*

Le temps d'exécution présenté dans la figure 2.B a également suivi une croissance exponentielle, mais pour le plus long chemin (longueur=28) n'a pas dépassé 81 minutes pour produire 184320 chemins.

Nous comparons le temps d'exécution des tests que nous avons réalisés par rapport aux tests décrits dans (Kochut *et al.,* 2007) sur une requête appliquée aux articles publiés uniquement en 2006 et une requête finale qui cherche à trouver des chemins entre des articles publiés entre 1981-2006. Le nombre maximum des chemins dans tous ces tests est égal à 6. La figure 3 fournit des informations sur le temps d'exécution de notre algorithme. La complexité de notre algorithme pour l'extraction des chemins présente une exécution semblable comparée au temps d'exécution présenté dans (Kochut *et al.,* 2007).

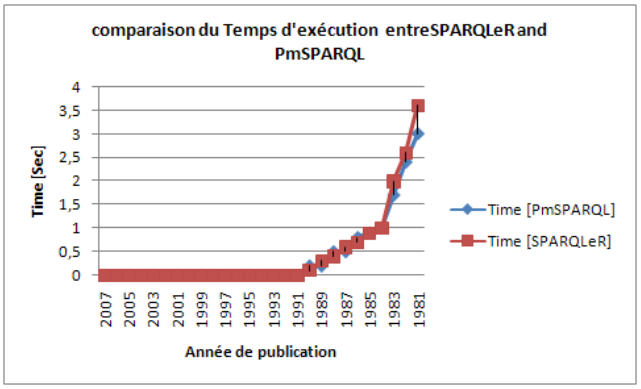


**Figure 3.** *Temps d'exécution pour une requête avec une entité source unique SwetoDBLP-april-2008-part1.*

### 6.3. *Test de PmSPARQL dans le domaine biomédical*

Nous avons exécuté des tests sur de nombreuses bases de données RDF afin d'évaluer la fonctionnalité de PmSPARQL pour trouver des chemins longs et complexes. Parmi ces tests, nous choisissons de présenter une requête significative dans le domaine biomédical qui cherche des réactions dans la voie de biosynthèse de Glycan. Nous proposons pour ce test, une requête de recherche des réactions entre deux familles de glycans N-Glycan-G00020 et N-Glycan-G00022. Un exemple de requête PmSPARQL est présentée dans la requête Q6 :

```
Q6=SELECT @P WHERE
{<glyco:N-Glycan-G00020>
  @P <r2>
PFilter(size(@P)<=8 AND IndPath(@P,["has-product"-"has-
reactant "]+))};
```

Le glycan N-Glycan-G00020 est un prédécesseur de N-Glycan-G00022. Ces entités sont sémantiquement associées, mais elles n'avaient aucun chemin direct les reliant. En fait, le chemin final (voie) relie la substance source N-Glycan-G00020 et un noeud non spécifique en utilisant une séquence répétitive de réactions spécifée par la direction inverse des deux propriétés "has-product" et "has-reactant". Cette requête localise une voie de longueur 8, se composant de plusieurs réactions contenues dans les données du graphe GlycO-0-9-full-enzyo. Les résultats de ces tests ont démontré l'efficacité du langage proposé et son applicabilité à plusieurs datasets.

En plus de l'exemple de Q6, nous avons testé une autre requête (Q7) qui permet d'identifier tous les chemins indirects entre deux entités spécifiées ("http ://lsdis.cs.uga.edy/EnzyO.owlR05991", "http ://lsdis.cs.uga.edy/EnzyO.owlR05986").

```
Q7=SELECT @P WHERE
{<http://lsdis.cs.uga.edy/EnzyO.owl#R05991>
  @P <http://lsdis.cs.uga.edy/EnzyO.owl#R05986>
PFilter(IndPath(@P,[has_substrate-has_product])
};
```

## 7. Conclusion

Dans cet article, nous avons présenté un langage de requêtes sémantiques pour l'identification d'associations sémantiques dans des bases de données RDF. Pour ce faire, nous avons présenté PmSPARQL, une extension du langage SPARQL. Le langage proposé est conçu pour trouver des associations sémantiques selon plusieurs paradigmes dépendants des besoins de l'utilisateur. L'inclusion des motifs de chemin dans des requêtes PmSPARQL a démontré des résultats encourageants motivés par l'extraction de longs chemins dans un temps raisonnable. En plus de l'extraction de longs chemins, notre implémentation démontre l'importance des mesures d'évaluation conçues pour ranger des associations selon divers critères (richesse, connectivité, importance,...) (Slimani *et al.,* 2008). Comme perspective de ce travail, les requêtes de PmSPARQL présentent un temps d'exécution favorable, pour cela nous pensons à optimiser les requêtes de PmSPARQL pour l'extraction des chemins ce qui présente un intérêt important pour l'usage pratique de ce langage dans des applications réelles.

## 8. Bibliographie

Aleman-Meza B., Halaschek-Wiener C., Arpinar B., Sheth A., « Context-Aware Semantic Association Ranking », *First International Workshop on Semantic Web and Databases.*, Berlin, Germany, p. 33-50, 2003.

Angles R., Gutierrez C., Hayes J., RDF Query Languages Need Support for Graph Properties (Technical Report TR/DCC-2004-3), Technical report, Department of Computer Science, University of Chile, 2004.

Anyanwu K., Maduko A., Sheth A., « SPARQ2L : Towards Support for Subgraph Extraction Queries in RDF Databases », *WWW 2007, Banf, Alberta, Canada*, p. 797-806, 2007.

Balmin A., Papakonstantinou Y., Hristidis V., Srivastava D., Koudas N., Wang T., « A System for Keyword Proximity Search on XML Databases », *Proceeding of 29th VLDB Conference, Berlin, Germany*, p. 1069-1072, 2003.

Bernd A., Michel S., « Gram : A Graph Data Model and Query Language », *European Conference on Hypertext Technology, Milan, Italy*, ACM Press, p. 201-211, 1992.

Boanerges A., HalaschekWiener C., Arpinar I., Ramakrishnan C., Sheth. A., « Ranking Complex Relationships on the Semantic Web », *IEEE Internet Computing. 9 (3)*, p. 37-44, May/-Jun, 2005.

Broekstra J., Kampman A., « SeRQL :A Second Generation RDF Query Language », *SWAD-Europe Workshop on Semantic Web Storage and Retrieval, Amsterdam, Netherlands*, p. 13-14, 2003.

Cruz I. F., Mendelzon A., Wood P. T., « A Graphical Query Language Supporting Recursion », *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, California,*, p. 323-330, 1987.

Donninger H., Bonome T., Radonovich M., Pise-Masison C., Brady J., Shih J., Barrett J., JBirrer M., « Whole genome expression profiling of advance stage papillary serious ovarian cancer reveals activated pathways », *Oncogene*, vol. volume 23, p. pp. 8065-8077, 2004.

Faisal A., Baget J., Euzenat J., « Complex Path Queries for RDF », *ISWC2005, Galway, Ireland*, 2005.

Helenius A., Aebi M., « Roles of N-Linked Glycans in the Endoplasmic Reticulum », *Annual Review of Biochemistry*, vol. volume 73, number 1, p. pp. 1019-1049, 2004.

Karvounarakis G., Alexaki S., Christophides V., Plexousakis D., Scholl M., « RQL : A Declarative Query Language for RDF », *WWW2002, Honolulu, Hawaii, USA*, ACM Press, p. 592-603, 2002.

Kemafor A., Angela M., Amit S., « SPARQ2L : Towards Support for Subgraph Extraction Queries in RDF Databases », *WWW 2007*, Banff, Alberta, Canada, p. 797- 806, May 8 12, 2007.

Kemafor A., Angela M., Sheth A., « Sem-Rank : Ranking Complex Relationship Search Results on the Semantic Web », *International World Wide Web Conference*, 14, ACM, Chiba, Japan, p. 117-127, 2005.

Kochut K., Janik M., « SPARQLeR : Extended SPARQL for Semantic Association Discovery », *16th International World Wide Web Conference, Innsbruck, Austria*, Springer-Verlag, p. 797-806, 2007.

Miki T., Nomura S., Ishida T., « Semantic web link analysis to discover social relationships in academic communities », *Symposium on Applications and the Internet, SAINT'05*, p. 38-45, 2005.

Milo T., Suciu D., « Index structures for Path Expressions », *Proceeding of the 7th Internatioanl Conference on Database Theory, Jerusalem*, Springer-Verlag, p. 277-295, 1999.

Pérez J., Arenas M., Gutierrez C., « Semantics and Complexity of Sparql », *International Semantic Web Conference 06, Athens, GA, USA*, Springer-Verlag, p. 30-43, 2006.

Perry M., Sheth A., « SPARQL-ST : Extending SPARQL to Support Spatiotemporal Queries », *Kno.e.sis Center Technical Report. KNOESIS-TR-2009-01*, 2008.

Prud'hommeaux E., Seaborne A., *SPARQL :Query Language for RDF*, W3C. 2005.

Seaborne A., *«RDQL A Query Language for RDF, WWWConsortium, Member Submission SUBM-RDQL-20040109»*, W3C. January, 2004.

Sintek M., Gmbh D., Decker S., « TRIPLE-An RDF Query, Inference, and Transformation Language », *In Proceedings of INAP'2001, Tokyo, Japan*, Springer-Verlag, p. 47-56, 2001.

Slimani T., Ben Yaghlane B., Mellouli K., « PmSPARQL : Extended SPARQL for Multi-paradigm Path Extraction », *International Journal of Computer, Information, and Systems Science, and Engineering.*, vol. 2 (3), p. pp. 179-190, 2008.

Souzis A., *«RxPath Specification Proposal»*, W3C. 2004.