# Un modèle d'index pour la résolution distribuée de requêtes sur un nombre restreint de bases d'annotations RDF.

Fabien Gandon, Moussa Lo, Cheikh Niang

# Un modèle d'index pour la résolution distribuée de requêtes sur un nombre restreint de bases d'annotations RDF

Fabien Gandon[1], Moussa Lo[2], Cheikh Niang[1]

[1] Edelweiss, Inria, Méditérannée, France
Fabien.Gandon@sophia.inria.fr
[2] LANI, UFR SAT, Université Gaston Berger, Saint-Louis, Sénégal
lom@ugb.sn

**Résumé** : En utilisant une décomposition des annotations RDF inspirée par la structure des connaissances modélisées nous proposons un mécanisme d'indexation et de requêtes sur un petit nombre de bases distribuées.

**Mots-clés** : RDF, requêtes distribuées.

## 1 Introduction

Un web sémantique, qu'il soit public ou sur l'intranet d'une entreprise, repose généralement sur plusieurs serveurs web qui proposent chacun différentes ontologies et différentes bases d'annotations utilisant ces ontologies pour décrire des ressources. Les scénarios d'usage amènent souvent un utilisateur à formuler des requêtes dont les réponses combinent des éléments d'annotation distribués entre plusieurs de ces serveurs. Ceci demande alors d'être capable :
(1) d'identifier les serveurs susceptibles d'avoir des éléments de réponse ;
(2) d'interroger des serveurs distants sur les éléments qu'ils connaissent sans surcharger le réseau;
(3) de décomposer la requête et router les sous-requêtes vers les serveurs idoines ;
(4) de recomposer les résultats à partir des réponses partielles.

Notre scénario motivant ici, vient du projet européen *SevenPro*[1] visant à développer des outils et des technologies afin d'aider un ingénieur à concevoir de nouveaux objets en lui fournissant des rendus 3D de l'objet à concevoir ou d'objets similaires conçus précédemment. Il utilise ces vues pour accéder aux informations sur ces objets quelque soit la provenance de ces informations. Le scénario inclut notamment l'extraction d'annotations à partir de documents textuels et de fichiers CADs (conception et dessin industriel assistés par ordinateur). Les sources d'information et d'annotations étant potentiellement distribuées, c'est dans ce cadre que nous nous intéresserons à la résolution de requêtes dont les réponses combinent des éléments d'annotation distribués entre plusieurs serveurs.

[1] http://www.sevenpro.org/

La recommandation SPARQL[2] du W3C fournit trois briques de base pour construire une telle solution:

- un langage de requête permettant de décrire les graphes RDF[1] recherchés sous forme de triplets (arcs du graphe) et de contraintes sur ces triplets (filtres) ;
- un format de réponse en XML donnant les valeurs des variables sélectionnées dans la requête pour chaque graphe trouvé ;
- un protocole permettant d'échanger les requêtes et leurs réponses avec un serveur distant, notamment en utilisant une architecture de services web.

Si la recommandation SPARQL permet effectivement d'interroger un serveur web sémantique distant, elle ne fournit pas de mécanismes permettant d'orchestrer la résolution collective d'une requête par plusieurs serveurs.

Dans cet article nous allons donc nous intéresser à un scénario particulier où les standards du web sémantique sont utilisés pour implanter un système d'information dans une organisation et où chaque serveur possède le même jeu d'ontologies (RDFS[1], OWL[1]) que les autres mais une base d'annotations (RDF[1]) différente qu'il met à disposition sous la forme d'un service web (SPARQL). Autrement dit, dans ce système à base de connaissances, le vocabulaire conceptuel est commun à tous les serveurs mais les bases de faits sont spécifiques à chaque serveur.

Dans un premier temps, nous présentons l'architecture du système qui repose sur des serveurs pairs appelés *hubs*. Nous décrivons ensuite le modèle d'index proposé pour caractériser les contributions possibles d'un *hub* ainsi que la méthode de résolution distribuée d'une requête.

## 2 Contexte technique et identification des serveurs

Notre système repose sur des *hubs* ou serveurs pairs dont l'architecture est la même partout où ils sont déployés. Un *hub* contient systématiquement:

- une interface utilisateur : un serveur web proposant des applications accessibles aux utilisateurs par leur navigateur web ;
- une interface programmatique : des services web proposant un accès distant aux applications du *hub* pour d'autres applications.

Dans cette architecture, illustrée en figure 1, un utilisateur peut se connecter à n'importe quel *hub* pour utiliser une application web et notamment pour soumettre une requête. Pour cet utilisateur et sa demande, ce *hub* est alors chargé d'identifier les autres *hubs* susceptibles de l'aider à répondre à la requête et d'orchestrer la résolution distribuée de la requête avec les *hubs* identifiés. Pour ce scénario, nous avons fixé deux hypothèses fortes:

- chaque *hub* a les mêmes ontologies que les autres *i.e.* les mêmes schémas RDFS et OWL sont répliqués sur chaque *hub*.
- les *hubs* sont en nombre restreint et leur connectivité est stable.

[2] W3C Semantic Web Activity http://www.w3.org/2001/sw/ et http://www.w3.org/2002/ws/sawsdl/

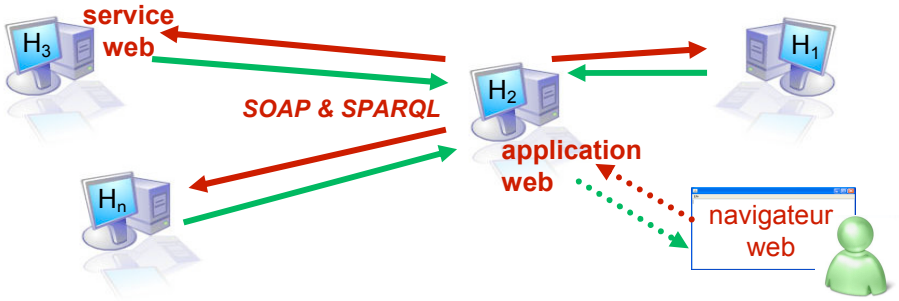


**Fig. 1** – Architecture globale d'un système à base de *hubs*.

Un *hub* implante les interfaces SPARQL ainsi que les interfaces nécessaires à la gestion de la distribution et décrites dans les sections suivantes. Chaque *hub* propose aussi une application web permettant de déclarer les autres *hubs* et en particulier l'URL du point d'accès à leurs services web. Cette description est elle même une annotation RDF du *hub* qui pourrait, dans une extension de ce travail, être générée à partir d'une description SAWSDL[3] par exemple en utilisant le mécanisme GRDDL[4].

Avec cette architecture chaque *hub* peut donc envoyer des requêtes à n'importe quel autre *hub*. Afin d'éviter l'envoi systématique de toutes les requêtes à tous les *hubs* (*broadcast*) et afin de trouver les réponses à une requête dont les triplets sont distribués sur différents serveurs nous allons nous intéresser dans les sections suivantes à sélectionner les *hubs* pouvant contribuer à la résolution d'une requête (*multicast* sélectif). Le premier problème à résoudre est de pouvoir caractériser le contenu de chaque serveur afin de pouvoir prédire s'il peut ou non contribuer à la résolution d'une requête donnée.

## 3 Modèle d'index d'un *hub*

La caractérisation du contenu d'un *hub* est un compromis entre la précision de la description (plus précise est la description, meilleure sera la sélection) et la concision de la description (la meilleure description du contenu, c'est le contenu lui même ; une description doit avoir une taille qui permet de l'échanger et de l'exploiter efficacement).

Nous appellerons index d'un *hub* la caractérisation du contenu de sa base d'annotation. Notre domaine d'application, la conception, va nous faire considérer la décomposition suivante ; une annotation d'un objet de conception contient souvent deux types de structures:

- des étoiles: une ressource qui est le sujet de plusieurs triplets, par exemple: un boulon qui a un diamètre, une longueur, une référence, un acier, etc.;

[3] http://www.w3.org/2002/ws/sawsdl/
[4] http://www.w3.org/2001/sw/grddl-wg/

- un chemin: une chaine de propriété qui relie plusieurs ressources par exemple: un chemin de partonomie où une voiture inclut une portière, qui inclut une charnière, qui inclut un boulon, qui inclut une vis.

Tout graphe a évidement ces structures, mais ce qui nous intéressera ici c'est de considérer le typage des ressources et des liens dans ces deux structures pour abstraire dans l'index les types de ressources et les types de liens impliqués et récurrents.

**Définition 1: un chemin**. Soient $x$ et $y$ deux ressources annotées dans une base. On dit qu'il existe un chemin $C(x, y) = <r_0, p_0, r_1, p_1, r_2,..., p_{n-1}, r_n>$ de $x$ vers $y$, si et seulement si il existe un nombre fini $n+1$ de ressources $r_0, r_1, ..., r_n$ avec $r_0=x$ et $r_n=y$ telles que pour tout $0 \leq i \leq n-1$ il existe dans les annotations un triplet $(r_i, p_i, r_{i+1})$. Notons que $r_n=y$ peut être un littéral.

**Définition 2: un chemin d'index.** Un chemin d'index $CI(x,y) = <t_0, p_0, t_1, p_1, t_2..., p_{n-1}, t_n>$ est un chemin mis dans l'index d'un *hub* pour indiquer qu'il existe dans la base d'annotations de ce *hub* au moins un chemin $C(x, y) = <r_0, p_0, r_1, p_1, r_2,..., p_{n-1}, r_n>$ tel que $r_i$ est du type $t_i$ avec $0 \leq i \leq n$.

**Définition 3 : une étoile.** Soit $x$ une ressource annotée dans une base. On dit qu'il existe une étoile $E(x) = ((x, p_0, r_0), (x, p_1, r_1),..., (x, p_n, r_n))$ autour de $x$, si et seulement si il existe un nombre fini $n+1$ de ressources ou littéraux $r_0, r_1, ..., r_n$ tels que pour tout $0 \leq i \leq n$ et $n \geq 2$ il existe dans les annotations un triplet $(x, p_i, r_i)$.

**Définition 4: une étoile d'index.** Une étoile d'index $E(x) = ((t_x, p_0, t_0), (t_x, p_1, t_2)... (t_x, p_n, t_n))$ est une étoile mise dans l'index d'un *hub* pour indiquer qu'il existe dans la base d'annotations de ce *hub* au moins une étoile $E(x) = ((x, p_0, r_0), (x, p_1, r_1),..., (x, p_n, r_n))$ telle que $r_i$ est du type $t_i$ avec $0 \leq i \leq n$.

Le calcul des chemins d'index (resp. étoiles) se fait par requêtes récursives à partir de chacun des chemins d'index (resp. étoiles) de taille 1 trouvés dans la base. Etant donné que RDF est un modèle de graphes orientés étiquetés qui peut contenir des cycles, nous contrôlons l'exploration récursive avec une profondeur de recherche maximale.

La figure 2 donne deux exemples de la génération d'un index respectivement pour les chemins (partie haute) et pour les étoiles (partie basse). Les exemples sont issus de bases d'annotations du projet *SevenPro*. En regardant cet exemple, on peut s'interroger sur la concision de l'index puisqu'ici il est plus grand que l'annotation de départ. Cependant, le "facteur de compression" de l'index est directement lié à la spécialisation des bases: si une base contient des descriptions concernant un même domaine (par exemple: les stocks de pièces d'assemblage) ces descriptions vont rapidement utiliser les mêmes structures d'étoiles et de chemins ; dès lors le nombre d'instances utilisant les mêmes structures d'index peut augmenter sans que l'index lui ne change de taille.

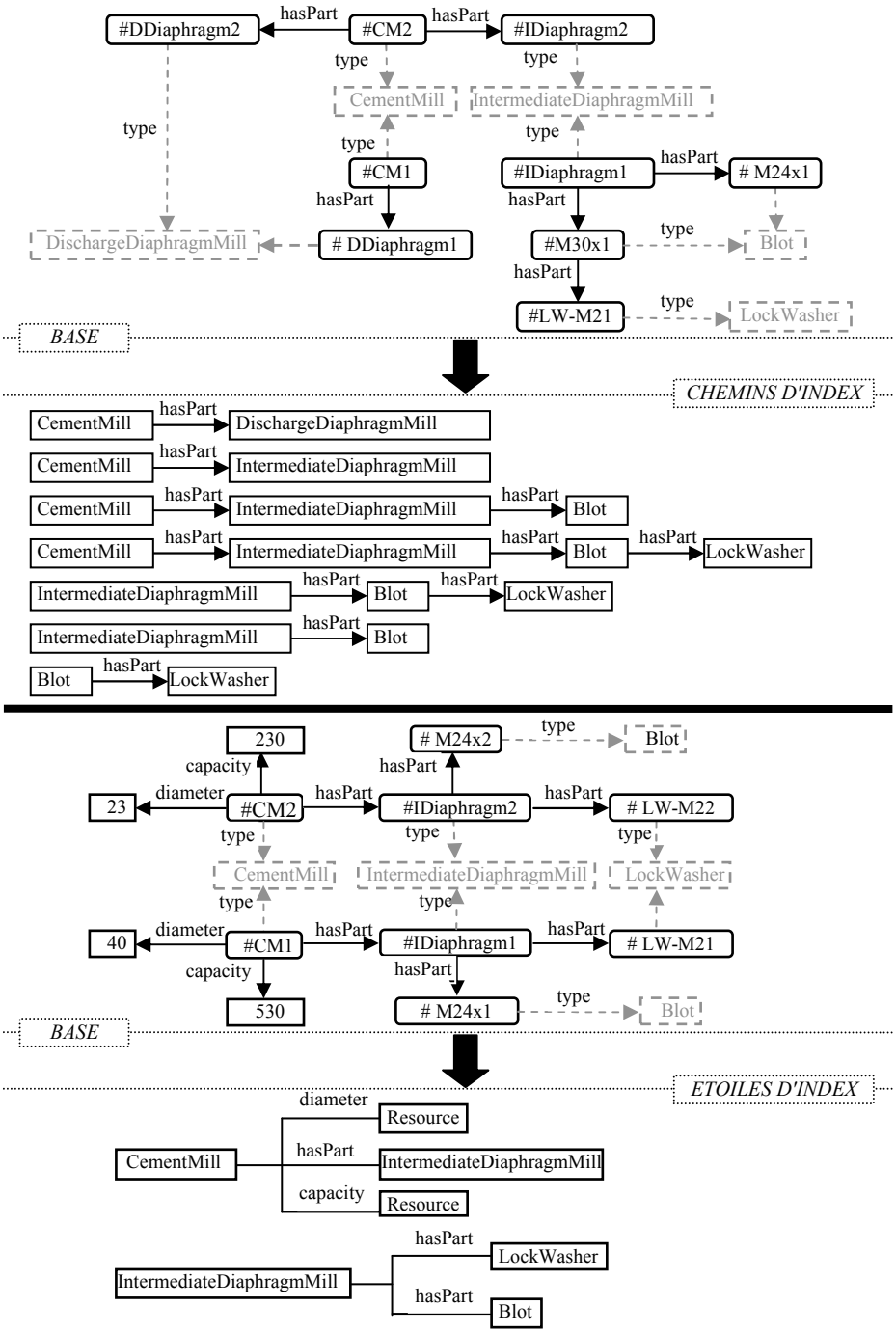


**Fig. 2** – Exemples de chemins et d'étoiles d'index générés à partir d'une annotation.

L'index, calculé par un *hub* à partir de ses annotations, est lui même sauvé comme une annotation RDF (Fig 3) en utilisant les primitives d'une ontologie système pour représenter les notions d'index, d'étoile et de chemin. Chaque chemin (resp. étoile) d'index est représenté par un chemin (resp. étoile) où des nœuds anonymes (*blank nodes*) sont typés à l'identique du chemin (resp. étoile) à représenter. Ainsi le mécanisme de la projection de graphe et le langage de requête SPARQL peuvent être utilisés directement pour interroger ces index.

```
<rdf:RDF (...) >
<CoreseServer rdf:about="#serverA12">
<contains>
 <Path>
  <numberInstance>18</numberInstance>
  <length>3</length>
  <content>
   <es:ShortLifter><es:partOf><es:IntermediateDiaphragmMill>
    <es:partOf><es:CementMill><es:capacityMill><rdfs:Resource/>
     </es:capacityMill></es:CementMill></es:partOf>
   </es:IntermediateDiaphragmMill></es:partOf></es:ShortLifter>
  </content>
 </Path>
</contains>
(...)
</CoreseServer>
</rdf:RDF>
```

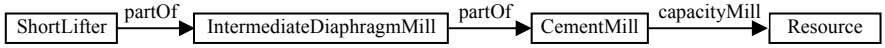


**Fig. 3** – Exemple d'annotation représentant un chemin d'index de longueur 3.

Les index sont calculés par des requêtes au moteur d'inférence de Corese (Corby et al., 2004). Ce moteur implante la sémantique de RDF, RDFS, ainsi que les *datatypes*, les propriétés transitives, symétriques et inverses de OWL Lite. Corese embarque aussi un moteur de règles en chainage avant, utilisé en particulier pour implanter la vérification des restrictions en OWL DL. Ces inférences, et les règles de production éventuellement ajoutées par l'application utilisant Corese, sont appliquées jusqu'à saturation, avant le calcul des index et la résolution des requêtes. Tous les traitements présentés ici se font donc sur une base saturée par ces inférences.

Les annotations représentant les index étant ajoutées à la base, le lecteur pourrait penser qu'elles risquent de polluer les résultats des requêtes sur cette base en ajoutant de nouveaux chemins et étoiles. En utilisant les mécanismes de provenance[5] associant une source à chaque annotation, nous pouvons identifier les annotations ayant pour source le calcul des index et les ignorer dans les traitements autres que le pilotage de la résolution distribuée d'une requête.

[5] http://www.w3.org/Submission/rdfsource/

Les sérialisations en RDF/XML des index sont échangées entre les *hubs* à chaque fois que l'un d'entre eux se déclare auprès des autres, par l'intermédiaire d'un service web dédié à ces échanges. Une fois échangés, ces index permettent à un *hub* de connaître le type de contributions qu'il peut attendre d'un autre *hub* ; chaque chemin (resp. étoile) d'index inclus dans l'index d'un *hub* indique que ce *hub* possède au moins un chemin (resp. étoile) de triplets utilisant des ressources et des propriétés de types donnés par ce chemin (resp. étoile). A ce point, nous pouvons donc aborder le problème de la résolution d'une requête.

## 4 Méthode de résolution distribuée d'une requête

Lorsqu'une requête est soumise à un *hub* par un utilisateur ou une application cliente celui-ci devient responsable de l'orchestration de sa résolution distribuée. Les grandes étapes de la résolution sont les suivantes:

1. décomposer la requête en chemins et en étoiles. Le principe est le même que pour les bases d'annotation: une construction récursive à partir de chacun des chemins (resp. étoiles) de taille 1 trouvés dans la requête.
2. projeter sur les index des *hubs* connus chaque chemin et étoile de la requête pour savoir quel *hub* est susceptible de contribuer et pour quelle partie de la requête. Les index étant des annotations comme les autres, la projection d'une étoile (resp. chemin) sur l'index est une requête SPARQL classique dont le résultat est une liste de *hubs* archivant des annotations susceptibles de contenir les étoiles (resp. chemin) recherchées.
3. émettre les sous-requêtes correspondant à chaque étoile (resp. chemin) vers les *hubs* identifiés comme des contributeurs pour cette étoile (resp. chemin) en utilisant les services web publiés par chacun de ces *hubs*.
4. recevoir les résultats partiels et les charger dans une base temporaire sur laquelle sera résolue la requête globale. Ce chargement dans une même base réalise l'union des résultats temporaires et leur jointure sur les URI communs.

La génération de sous requêtes peut être optimisée, comme nous le discuterons en conclusion. La version actuelle prend déjà en compte le problème des *blank nodes* aux extrémités des sous-graphes requêtes: un *blank node* est un sommet anonyme dans le graphe RDF ; un sommet anonyme ne peut être joint à aucun autre sommet sauf cas particuliers en OWL non traités ici (ex: propriétés fonctionnelles).

Par conséquent les sous-requêtes construites incluent des contraintes pour éviter de renvoyer des résultats partiels qui ne pourraient pas être joints aux autres lors de la résolution de la requête globale. En d'autres termes tout sommet d'un chemin ou d'une étoile devant être joint à d'autres sommets ne peut être un sommet anonyme.

En SPARQL un corps de requête (Fig 4 - partie gauche) va donc être décomposé en plusieurs sous-requêtes SPARQL correspondant aux étoiles et chemins qu'elle contient (Fig 4 - les lignes 1-6 et 1;7-9 génèrent respectivement des chemins et des étoiles). Toutes les opérations de résolutions de requêtes sur une base ou un index sont implantées en utilisant le moteur CORESE (Corby *et al.*, 2004).

```
1  ?x rdf:type es:CementMill .
2  ?x es:includes ?y .
3  ?y rdf:type es:Diaphragm .
4  ?y es:includes ?z .
5  ?z rdf:type es:Lifter .
6  ?z es:fixedBy  ?u .
7  ?x es:diameter ?d .
8  ?x es:capacity ?c .
9  ?x es:hasPart  ?t .
```

un des chemins

```
?x rdf:type es:CementMill .
?x es:includes ?y .
?y rdf:type es:Diaphragm .
?y es:includes ?z .
?z rdf:type es:Lifter .
filter(!isBLANK(?x))
filter(!isBLANK(?z))
```

une des étoiles

```
?x rdf:type es:CementMill .
?x es:diameter ?d .
?x es:capacity ?c .
?x es:hasPart  ?t .
filter(!isBLANK(?x))
```

**Fig. 4** – Exemples de corps d'une requête et de deux des sous requêtes générées.

## 5 Tests dans le projet *SevenPro*

Les tests actuels de notre prototype, se font sur trois *hubs* ayant chacun toutes les ontologies et une partie des annotations du projet *SevenPro*. La figure 3 montre une copie d'écran de l'interface de visualisation des index. La figure 4 montre une interface de suivi et débogage de la résolution distribuée.

Le cas d'usage de *SevenPro* visualisé ici est celui d'une entreprise de fabrication de moulins ayant plusieurs sources d'information sur chaque produit: bases de CAO, documents textuels techniques, catalogues numériques, base de normes publiques, etc.

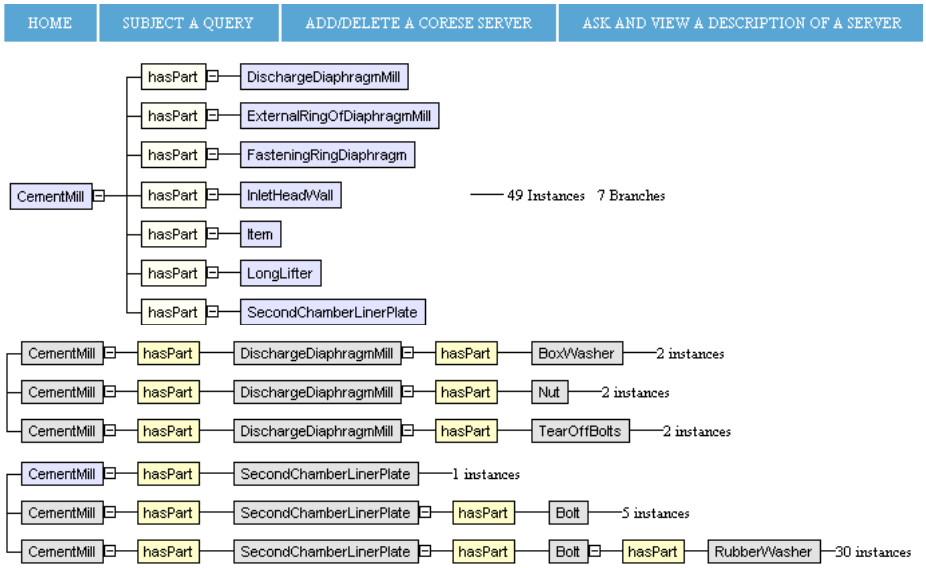


**Fig. 3** – Copie d'écran de l'interface de visualisation des index dans *SevenPro*.

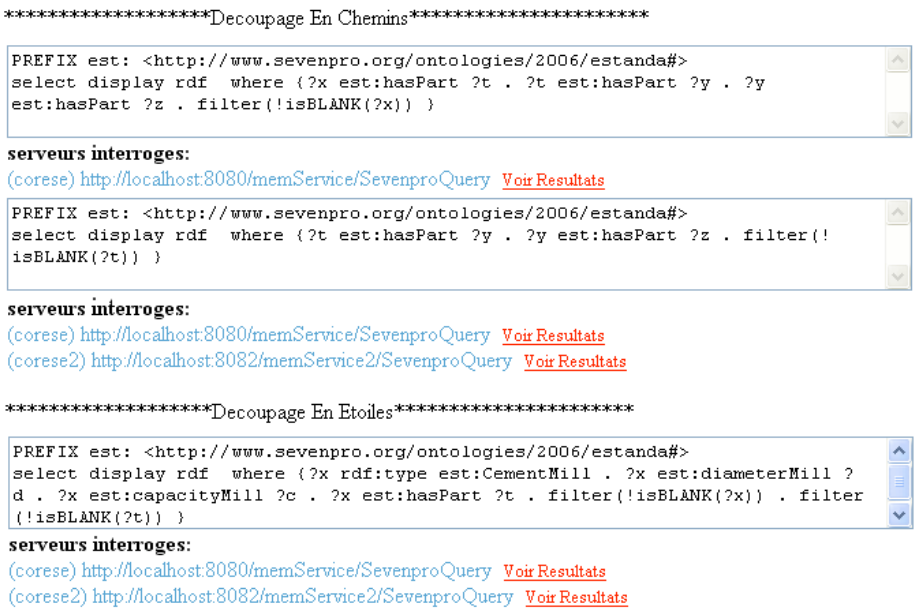


**Fig. 4** – Copie d'écran de l'interface de suivi de la résolution distribuée.

La figure 3 montre une décomposition en étoiles et en chemins correspondant à l'index d'une base d'annotations de documents de CAO. L'un des objectifs de *SevenPro* est d'obtenir par ailleurs ces annotations à partir de la fouille de la structure des documents de CAO. La figure 4 montre le suivi de la résolution distribuée d'une requête sur les trois *hubs* et la sélection des serveurs pertinents à chaque étape de la décomposition. La requête interroge les bases sur les grands types de composants utilisés dans le moulin (ex: diaphragme) en fonction des caractéristiques du moulin (ex: diamètre, produit mouliné, capacité). Les réponses peuvent être intégrées à des visualisations 3D (Fig. 5) pour l'aide à la conception, les tests, l'aide au montage, etc.

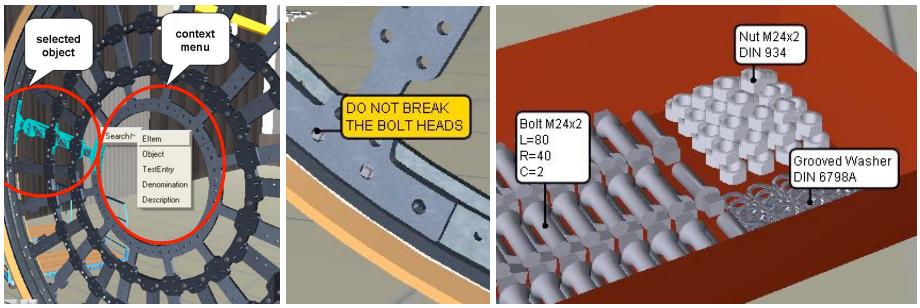


**Fig. 5** – Visualisation 3D enrichie par des requêtes SPARQL dans *SevenPro*.

## 6 Positionnement et discussion

Le travail présenté ici est une extension d'un travail antérieur (Gandon, 2003) motivée par l'article de (Stuckenschmidt *et al.*, 2004) qui, par sa structure hiérarchique d'index et ses définitions a servi de base aux définitions et extensions faites ici. Notre principal ajout est l'introduction de la structure d'étoile comme deuxième motif de décomposition des bases. Le travail présenté dans (Stuckenschmidt *et al.*, 2004) capitalisait déjà sur des travaux sur les structures d'index de modèles objets de (Bertino, 1991), (Bertino & Foscoli, 1995) et (Shidlovsky & Bertino, 1996). Ces travaux sont les plus pertinents en ce sens qu'ils bâtissent à la fois sur les résultats dans le domaine des bases de données distribuées et dans le domaine des bases de données ayant un schéma objet.

ARQtick[6] propose une clause supplémentaire de SPARQL permettant de distribuer des parties prédéterminées d'une requête à des serveurs prédéterminés ; chaque partie n'étant déléguée qu'à un seul serveur pré-choisi. L'extension "Federated SPARQL"[7] propose le complémentaire d'ARQtick en permettant d'associer à une requête des *bindings i.e.* des listes de candidats pré-calculés pour des variables de la requête. Ainsi des requêtes partiellement résolues peuvent transiter entre des serveurs. DARQ[8] permet de fédérer des bases RDF et utilise un routage au niveau des triplets en se basant sur le type de prédicat demandé. Ces trois initiatives se placent à un niveau plus élémentaire de la fédération de bases et fournissent des briques élémentaires pour des algorithmes de distribution de requêtes ou des cas de fédération très spécifiques.

Nous ne considèrerons pas ici les approches pair-à-pair telles que (Cai & Frank, 2004) ou (Kokkinidis *et al.*, 2006) car leurs échelles de déploiement et la nature éphémère de leurs réseaux ne leurs permettent pas de reposer sur un échange préliminaire d'index comme nous le faisons ici et les orientent vers des problèmes de routage de requêtes entre pairs et des structures comme les tables de hash distribuées. Nous ne considérons ici que des réseaux de *hubs* petits et stables comme on pourrait en trouver dans un système d'information d'entreprise.

Enfin l'optimisation des requêtes est un domaine connexe où les index sont aussi des structures centrales. Dans le cadre de RDF nous citerons en particulier (Harth & Decker, 2005) qui proposent aussi un ensemble de structures d'index de littéraux et de graphes mais dédiées à l'accélération des accès à une base de triplets.

Les travaux présentés ici ont montré plusieurs perspectives:

1. il existe une redondance entre les index de chemins et les index d'étoiles notamment pour les tailles 1 et 2. Nous étudions une meilleure intégration des algorithmes d'indexation pour obtenir un index unique sans redondance.
2. il existe une redondance entre les réponses à une sous-requête correspondant à un chemin (resp. étoile) et les réponses à une sous-requête correspondant aux chemins (resp. étoiles) de taille inférieure. Nous cherchons actuellement à combiner une génération plus concise des index et des contraintes supplémentaires dans les sous-requêtes de façon à éviter cette redondance.

[6] http://seaborne.blogspot.com/2007/07/basic-federated-sparql-query.html
[7] http://www.w3.org/2007/05/SPARQLfed/
[8] http://darq.sourceforge.net/

3. les valeurs littérales sont correctement traitées dans la résolution, mais ne participent pas à l'optimisation de l'émission des sous-requêtes car leurs particularités ne sont pas exploitées par les index au contraire de (Gandon, 2003) et (Harth & Decker, 2005).
4. il serait intéressant d'exploiter des résultats partiels déjà reçus pour contraindre les sous-requêtes restantes en envisageant une extension comme ARQtick.

A plus long terme, nous envisageons l'extension de l'hypothèse de la décomposition en étoiles et chemins des index pour considérer d'autres structures par exemple en cherchant à détecter les graphes minimaux des bases *i.e.* le graphe en étoile autour d'une ressource dont toutes les branches se terminent soit sur une ressource nommée soit sur un littéral soit sur une ressource anonyme sans autre relation ; voir, par exemple, les MSG de (Tummarello *et al.*, 2005). Nous envisageons aussi l'utilisation de statistiques sur les motifs composant un index afin d'ordonner les priorités de routage entre les *hubs* par exemple en favorisant les requêtes les plus contraintes pour diminuer la combinatoire. Une autre extension consisterait à permettre des résultats paginés pour contrôler le flot des données sur le réseau. Enfin nous envisageons d'étendre l'infrastructure à des producteurs d'annotations dynamiques par exemple en incluant des *wrappers* de base de données correspondant à des *hubs* ayant un index fixé et spécialisé.

**Remerciements:** La commission et le projet européens SevenPro FP6-027473

# Références

BERTINO, E., (1991), An indexing technique for object-oriented databases. In Proc. International Conference on Data Engineering, 160-170, IEEE Computer Society.

BERTINO, E. & FOSCOLI, P., (1995) Index organizations for object-oriented database systems. TKDE, 7(2):193–209.

CAI, M. & FRANK, M., (2004) RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network, In Proc. International World Wide Web Conference.

CORBY, O., DIENG-KUNTZ, R., FARON-ZUCKER, C., (2004) Querying the Semantic Web with the CORESE search engine. In R. Lopez de Mantaras and L. Saitta eds, Proc. of the 16th European Conference on Artificial Intelligence, subconference PAIS, 22-27, IOS Press, p. 705-709.

GANDON, F., (2003) Agents handling annotation distribution in a corporate semantic Web, Web Intelligence and Agent Systems, IOS Press Intemational Journal, (Eds) Jiming Liu, Ning Zhong, Volume 1, Number 1, p 23-45, ISSN: 1570-1263

HARTH, A. & DECKER, S., (2005) Optimized index structures for querying rdf from the web. In Proceedings of LA-WEB the 3rd Latin American Web Congress, IEEE Press

KOKKINIDIS, G., SIDIROURGOS, L., CHRISTOPHIDES, V., (2006) Book Chapter in Semantic Web and Peer-to-Peer, S. Staab, H. Stuckenschmidt (eds.), Springer-Verlag

SHIDLOVSKY, B., BERTINO, E., (1996) A graph-theoretic approach to indexing in object-oriented databases In S. Y. W. Su, editor, Proceedings of the Twelfth International Conference on Data Engineering, 230–237, IEEE Computer Society.

STUCKENSCHMIDT, H., VDOVJAK, R., HOUBEN, G.J., BROEKSTRA, J., (2004) Index Structures and Algorithms for Querying Distributed RDF Repositories, in Proc. WWW Conference, 17-22

TUMMARELLO, G. , MORBIDONI, C., PULITI, P., PIAZZA, F., (2005) Signing individual fragments of an RDF graph, Special interest tracks and posters of the World Wide Web Conf., 10-14