TD3 : SPARQL et Inférences avec JENA et ARQ

STI4A – Cours d'IA

B. NGUYEN

Ressources : à télécharger sur http://www.benjamin-nguyen.fr/ENS/IA/SPARQL/ lorsque c'est nécessaire.

Prérequis : avoir correctement configuré JENA dans votre IDE (e.g. Eclipse). ARQ est l'IDE permettant l'interrogation SPARQL de données RDF.

Objectif du TD : interrogation de données avec JENA/ARQ (API SPARQL). Inférences.

Exercice 1 : Vérification de l'installation. Téléchargez le programme Test1.java et les données wine.rdf et exécutez le. Vérifiez que vous en comprenez bien le fonctionnement. Modifiez la requête SPARQL de telle sorte que vous déclarez en préambule le préfixe de l'URI de la propriété « p », en utilisant la commande `PREFIX loc: <urn:x-hp-jena:eg/>` dans la suite, on supposera que les préfixes sont définis correctement.

Exercice 2 : Créez un nouveau programme Test2 dans lequel vous poserez la requête SPARQL suivante : `SELECT ?a WHERE {?a loc:q "foo"}` A quel résultat vous attendez-vous, et êtes vous surpris de la réponse ? Utilisez un moteur de raisonnement `Reasoner r`, construit avec l'un des constructeurs suivants : `ReasonerRegistry.getRDFSReasoner()`, `ReasonerRegistry.getOWLReasoner()`, `ReasonerRegistry.getOWLMicroReasoner()`, et générez le nouveau modèle basé sur les inférences possibles en utilisant le constructeur : `InfModel inf = ModelFactory.createInfModel(r, rdfsExample)`

Quel moteur d'inférence faut-il utiliser ? Qu'est ce qui change si on en utilise un autre ? Exécutez la requête sur ce nouveau modèle, est-ce que vous observez ce à quoi vous vous attendiez ?

Exercice 3 : Téléchargez le fichier Test3.java, le schéma OWL (en format rdf/xml) `owlSchema.owl` et les données (en format rdf/xml) `data.rdf`. Essayez de comprendre la signification de l'ontologie, est décrivez la « en français ».

Que fait la méthode printStatements ? Que fait la requête SPARQL qui est exécutée ? En regardant les données, est-ce que ce résultat vous parait normal ? Modifiez le programme (voir ligne commentée) pour utiliser le modèle avec les inférences. Qu'observez-vous ? Expliquez. Testez les autres types de Reasoner (RDFS, OWLMini, OWLMicro). Qu'en déduisez-vous ?

Exercice 4 : Téléchargez le fichier wine.owl. Il s'agit d'une ontologie OWL et de données. Ouvrez le et essayer de comprendre sa structure. Ecrivez une requête SPARQL permettant de lister les URI des ressources qui se trouvent dans (`wine:locatedIn`) une région française (`wine:FrenchRegion`). Combien trouvez-vous de régions ? Posez une autre requête affichant toutes les régions. Votre réponse vous parait-elle satisfaisante ? Avez-vous utilisé du raisonnement ? Posez une requête SPARQL permettant de trouver tous les vins rouges (`wine:hasColor wine:Red`). Listez également le cépage utilisé (`wine:WineGrape`).

Exercice 5 : requêtes SPARQL, à poser sans raisonnement, sur les données www2012.ttl (RDF format Turtle, il se charge exactement de la même manière). N'oubliez pas de bien configurer les préfixes. Ce fichier représente les données d'un certain nombre de conférences sur le web sémantique.

R1 : Donnez le titre (dc:title) de tous les workshops (swc :WorkshopEvent)

R2 : Donnez la liste ordonnée de tous les sujets (dc:subject) de tous les workshops, sans doublons.

R3 : Donnez le titre de tous les workshops ayant comme sujet « Linked Data ». Est-ce que le workshop USEWOD fait partie des solutions ? Consultez le fichier www2012.ttl pour essayer de comprendre.

R4 : En utilisant une expression régulière (voir : http://www.w3.org/TR/rdf-sparql-query/#funcex-regex) modifiez votre requête précédente pour que le workshop apparaisse.

R5 : Chaque workshop a eu lieu dans un endroit précis (propriété swc:hasLocation). Donnez le rdfs:label de chacun de ces endroits, associé au titre du workshop qui s'y est déroulé.

R6 : Donnez les URI de tous les évènements qui ont eu lieu au même endroit que le workshop « USEWOD » de 2012. Ce workshop ne devra pas apparaitre dans l'ensemble des solutions.

R7 : Donnez les évènements qui se sont déroulés avant USEWOD. Vous pourrez utiliser les propriétés ical:dtstart et ical:dtend.

Est-ce que poser ces même requêtes avec du raisonnement changerait quelque chose ? Si oui, quoi ?