Práctica 5.
Creación y consulta de grafos RDF.

Escuela de Ingeniería y Arquitectura
Depto. de Informática e Ingeniería de Sistemas

1. Introducción

Los objetivos de esta quinta práctica son los siguientes:

1. Aprender a utilizar la librería de Java JENA para la creación y consulta de grafos de RDF
2. Aprender a ejecutar consultas SPARQL sobre los grafos creados

2. Jena

JENA es un framework de Java de Apache que permite la creación y consulta de grafos RDF (`http://www.apache.org/dist/jena/binaries` (apache-jena)). Proporciona un API para lectura, procesado y escritura de RDF en XML, N-triples, y Turtle. Proporciona un sistema de almacenamiento para gestionar grandes cantidades de triples RDF, y un motor de consulta de SPARQL para la publicación de servicios de acceso a los datos.

Esta práctica muestra un resumen de las operaciones básicas de manejo de RDF con JENA y propone ejercicios para que probar su funcionamiento. Una descripción más detallada y diversos tutoriales son proporcionados en la página del proyecto (`http://jena.apache.org/tutorials/rdf_api.html`).

3. Creación de un modelo de RDF usando Jena

En el proyecto de Eclipse proporcionado junto con este enunciado, podéis encontrar un conjunto de programas de prueba que muestran la funcionalidad básica de JENA.

En la clase CreacionRDF se muestra cómo construir un modelo de Jena y añadir nuevos recursos mediante la clase Model. Esto se puede realizar añadiendo recursos (clase Resource), y a estos propiedades (clase Property) o mediante la adición de tripletas completas (sujeto, predicado, objeto). La creación de un nuevo modelo se realiza a través de la clase ModelFactory.

Ejercicio

- Crea el código necesario para añadir que el recurso ya definido tiene la relación `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` con valor `http://xmlns.com/foaf/0.1/person`
- Crea dos recursos más de tipo persona con la URI y valores que querais. Añadir entre ellos la relación `http://xmlns.com/foaf/0.1/knows`
- ¿Cual es la diferencia entre las relaciones rdf:resource y rdf:nodeID?
- ¿Cual es la diferencia entre <vcard:N rdf:nodeID="A0"/>y <vcard:N>A0< vcard:N>
- Añade una nueva tripleta al modelo utilizando alguno de los métodos add de la clase model.

4. Carga y almacenamiento de modelos de RDF existentes

La clase PersistenciaRDF_Fichero muestra cómo almacenar un modelo de JENA en un fichero de texto RDF en diferentes formatos de representación y como volver a cargarlo en memoria.

La clase PersistenciaRDF_Fichero usa métodos que cargan el modelo RDF en memoria. Dichos métodos no son adecuados para el almacenamiento de cientos de miles de tripletas. Para modelos grandes se utilizan bases de datos de tripletas (triple stores). La clase PersistenciaRDF_TDB2 muestra cómo crear un triple store (TDB2), como añadir un nuevo modelo y como recuperarlo. La clase utilizada para ello se llama Dataset y representa el triplestore. Un dataset contiene un grafo RDF por defecto y se le pueden añadir nuegos grafos identificados por una URI.

Ejercicio

- Modifica la clase PersistenciaRDF_TDB2 para que importe el fichero "nombre.rdf" generado por la clase CreacionRDF en un grafo con nombre. Usa para ello la clase de Jena RDFDataMgr y el método apropiado de la clase Dataset.

5. Acceso a los elementos dentro de un modelo de Jena (API y SPARQL)

Aparte de la creación y almacenamiento de los modelos es necesario poder acceder a su contenido. Esto se puede hacer de dos maneras: mediante el API del modelo, o a través del API de consultas de SPARQL. Esta sección se centra en el primer caso.

La clase AccesoRDF muestra cómo acceder a los distintos elementos del modelo contenido el el fichero "card.rdf" mediante el API de Jena. La clase usada en el acceso a los recursos RDF es la clase Statement. Esta clase representa a una tripleta de información compuesta por un Sujeto, Predicado y Objeto. Estos elementos pueden ser Recursos (Resource), Propiedades (Property) o Literales (Literal).

Ejercicio

- Crea un programa que dada una URI cualquiera del fichero card.rdf muestre todos los recursos del modelo que comparten al menos una de sus propiedades.

El acceso al contenido del modelo a través del API es adecuado para aquellas situaciones en las que el modelo es pequeño o sabemos exactamente lo que estamos buscando (ej., propiedades de un recurso con URI conocida). En cualquier otro caso deberemos recurrir al módulo de consulta de SPARQL.

La clase AccesoSPARQL muestra un ejemplo de uso para los diferentes tipos de consulta en SPARQL (query, describe, ask, constuct).

Ejercicio

- Crea una consulta que devuelva todos los literales que contengan el texto "Berners-Lee".
- Crea una consulta que devuelva el título de todos los documentos creados por Tim Berners-Lee.
- Crea un nuevo modelo que solo contenga la información de los documentos creados por Tim Berners-Lee.

6. Inferencia

En RDFS existen 13 reglas de inferencia lógica que permiten deducir nueva información de un modelo de forma automática. Por ejemplo, las propiedades subClassOf y supPropertyOf estan definidas como transitivas. En la clase InferenciaRDFS se muestra como crear un modelo inferido de acuerdo a las reglas de RDFS.

Ejercicio

- Observa el código fuente de la clase InferenciaRDFS y los resultados mostrados.
- ¿Que nueva información se ha deducido y que regla de inferencia la ha generado?. (Ver: https://www.w3.org/TR/rdf-mt/ sección 7.3)

En el tema de OWL veremos propiedades adicionales que se pueden añadir a los recursos y propiedades para poder deducir más información. La clase InferenciaOWL muestra como aplicar al mismo modelo usado en la clase InferenciaRDFS las reglas de inferencia de OWL2.

Ejercicio

- Observa el código fuente de la clase InferenciaOWL2 y los resultados mostrados. Al usar como ejemplo un modelo pensado para RDFS, el razonador de OWL, prácticamente deduce lo mismo que el razonador de RDFS.
- Descomenta las dos líneas que definen la relación padreDe como inversa de la relación de hijo en OWL, y vuelve a generar el modelo inferido. Observa la nueva información que se añade.