Versión beta del módulo de consultas federadas

A continuación, se redacta la información necesaria para entender cómo es que fue construido el módulo de consultas federadas geoespaciales.

Definiciones

Traductor: Es un programa que convierte un programa fuente en un programa objeto.

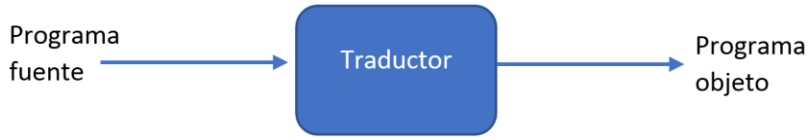

Fig. 1 Traductor.

Compilador: Es un traductor que convierte lenguaje de medio o alto nivel a lenguaje de máquina o ensamblador. Por ejemplo: C, Java, Pascal, Fortran, etcétera)

Intérprete: Es un programa que traduce y ejecuta al mismo tiempo lenguaje de alto nivel. Por ejemplo: LISP, Basic, Logo entre otros.

Jerarquía de lenguajes: La jerarquía de lenguajes se refiere al nivel de dependencia que tiene un cierto lenguaje con la máquina o procesador. Los lenguajes se clasifican de la siguiente forma:

- Lenguajes de máquina: Son los nativos de cada computadora y describen la arquitectura de esta. Las instrucciones en lenguajes de máquina se representan mediante códigos numéricos. La comunicación entre los lenguajes de máquina y la máquina no son necesarios.
- Lenguajes ensambladores: Son conjuntos de símbolos que representan unívocamente a cada código numérico del lenguaje de máquina asociado. El nombre que reciben las instrucciones de este tipo de lenguajes es nemónico.
- Lenguajes de nivel medio: Son lenguajes que permiten el uso eficiente de lenguajes ensambladores y también sobre las estructuras de control de flujo, de iteración y el orden de un lenguaje de alto nivel.
- Lenguajes de alto nivel de uso general: Este tipo de lenguajes permiten abstraerse de la estructura interna de la computadora al manipular las estructuras de control de flujo, de iteración y secuencia.
- Lenguajes de propósito específico: Son lenguajes de alto nivel que solucionan problemas especiales en determinadas tareas como PostgreSQL para manejar bases de datos.
- Lenguajes de inteligencia artificial: Estos lenguajes permiten declarar explícitamente conocimiento y las estructuras de control de flujo, de iteración y secuencia no son expresadas en el programa por lo que está abstraída la lógica del control de la computadora.

Partes de un compilador

Si bien los compiladores pueden clasificarse en función a su construcción como de una, dos, múltiples pasadas, en función de su construcción o de alguna característica especial. Sin embargo, generalmente se dividen en 2 partes: análisis y síntesis.

La parte de análisis está compuesta por:

- Analizador léxico o *scanner*: Encargado de analizar el código fuente y separar las palabras mediante token los cuales son un conjunto de caracteres como pueden ser palabras reservadas, variables, símbolos aritméticos, de relación entre otros. Elimina comentarios y los tokens son almacenados en una tabla de símbolos.
- Analizador sintáctico *parser*: Encargado de analizar la correcta agrupación de los tokens en frases gramaticales las cuales pueden ser representadas mediante un *parse tree*.
- Analizador semántico: Encargado de verificar la consistencia de las expresiones y/o significado del código mediante la localización de errores.

La parte de síntesis está compuesta por:

- Generador de código intermedio: Encargado de llevar acabo equivalencias entre el código fuente y código máquina. Usualmente se usan como representación la notación de postfijos, árbol de direcciones de código o un árbol de sintaxis.
- Optimizador de código: Encargado de reducir el código generado por el generador de código intermedio para reducir el número de recursos y mejora la velocidad de ejecución.
- Generador de código: Genera código ensamblador o código ejecutable de tal forma que la máquina pueda interpretar el código fuente.

Identificación de elementos en Apache Marmotta

Para llevar a cabo la implementación del módulo de consultas federadas geoespaciales, se tuvieron que considerar las definiciones y conceptos anteriores para saber cómo se realizaría la implementación del código puesto que está basado en el código RDF4J que está construido sobre Java y cuya licencia es *Eclipse Distribution License (EDl)* por lo que el código estará bajo la licencia *Apache License 2.0* y no la de Eclipse.

Analizador léxico: Los primeros archivos con los que la consulta tiene contacto es con el analizador de léxico ubicado en

Marmotta/libraries/kiwi/kiwi-sparql/src/main/java/org/apache/marmotta/kiwi/sparql/evaluation/KiWiEvaluationStrategy.java

Y con el evaluador de expresiones

Marmotta/libraries/kiwi/kiwi-sparql/src/main/java/org/apache/marmotta/kiwi/sparql/builder/eval/ValueExpressionEvaluator.java

Analizador sintáctico: Posteriormente, la consulta es evaluada sintácticamente por un conjunto de archivos alojados en las carpetas:

Marmotta/libraries/kiwi/kiwi-sparql/src/main/java/org/apache/marmotta/kiwi/sparql/builder/collect/*

Analizador semántico: Después es evaluada la consulta para corroborar que no haya errores en la consistencia de las expresiones o significado del código. El archivo encargado de esta tarea está en:

Marmotta/libraries/kiwi/kiwi-sparql/src/main/java/org/apache/marmotta/kiwi/sparql/exception/UnstisfiableQueryExecution.java

Generador de código intermedio: Dándole el contexto adecuado al proyecto, el programa fuente es SPARQL, el traductor el Java y el lenguaje objeto es SQL, que por razones de funcionalidad dicho código es ejecutado sobre PostgreSQL, se tiene que generador de código intermedio está compuesto por el siguiente archivo:

Marmotta/libraries/kiwi/kiwi-sparql/src/main/java/org/apache/marmotta/kiwi/sparql/model/SQLBuilder.java

Optimizador de código: Con el objetivo que la consulta sea la óptima, se hace uso de 3 archivos alojados en:

Marmotta/libraries/kiwi/kiwi-sparql/src/main/java/org/apache/marmotta/kiwi/sparql/optimizer/*

Generador de código: Finalmente, si no se tuvo problemas para en el análisis y síntesis de la consulta, se procede a ejecutar la consulta en PostgreSQL puesto que dicha tecnología es usada para el almacenamiento de datos geoespaciales. El archivo responsable de esta tarea es:

Marmotta/libraries/kiwi/kiwi-triplestore/src/main/java/or/apache/marmotta/kiwi/persistance/pgsql/PostgreSQLDialect.java

A manera de analogía, el siguiente diagrama mostrado en la figura 2 se muestra cómo es que conviven las 3 tecnologías

Fig. 2 Interacción entre las 3 tecnologías principales.

Cabe decir que Java es un lenguaje de alto nivel y funge la función de traductor en el proyecto. SPARQL, el cual es ejecutado en Apache Marmotta, es el programa fuente mientras que el programa objeto es SQL el cual es ejecutado sobre PostgreSQL.

Pruebas

Se probaron 2 consultas federadas a DBpedia de las cuales una fue normal y la otra fue con datos geoespaciales. Las consultas son las siguientes:

La primera consulta, mostrada en la figura 3, solamente recuperó las coincidencias de tipos de datos entre DBpedia y la máquina local. Como comentario, esta consulta sirve para medir de una manera vaga la completitud del *triple store* disponible de forma local.

```
select * {

  #Tipos de clases extraidos desde dbpedia mediante consulta federada
    service <https://dbpedia.org/sparql> {
      select distinct ?t {
        [] a ?t .
      } limit 1000
    }
    #Recursos locales que pertenecen a tipos de Dbpedia
    ?a a ?t .
} limit 100
```

Fig. 3 Consulta #1.

En la figura 4 se muestra la consulta y resultados de la figura 3 en ejecución sobre la plataforma de Apache Marmotta.

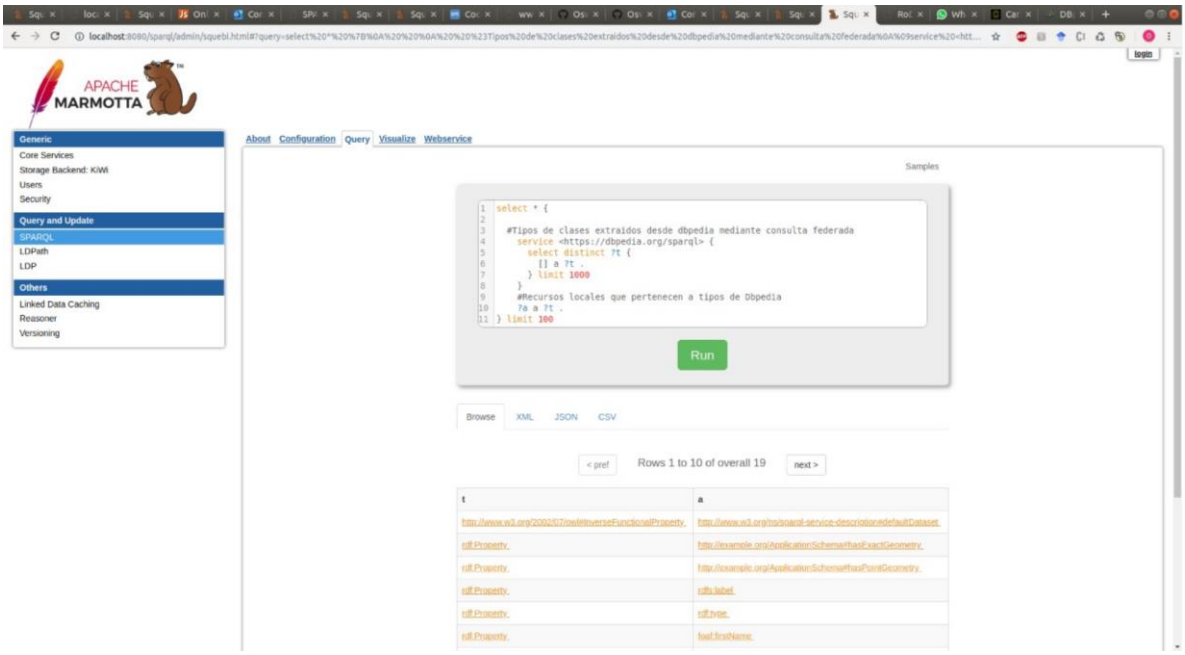

Fig. 4 Ejecución y recepción de resultados de la consulta #1.

Pasando a la siguiente consulta, mostrada en la figura 5, se hace una consulta federada que recopile datos geoespaciales sobre los teatros que estén en la ciudad de Paris, Francia.

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geos: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX : <http://www.semanticweb.org/frubi/ontologies/2017/10/puntsWIFI#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>


SELECT *
WHERE {
  SERVICE <http://dbpedia.org/sparql/> {
    SELECT ?teatreName ?lat ?long
    WHERE {
      ?teatre rdf:type dbpedia:Theatre .
      ?teatre foaf:name ?teatreName .
      ?teatre geo:lat ?lat .
      ?teatre geo:long ?long .
      ?teatre dbp:city "Paris"^^<http://www.w3.org/1999/02/22-rdf-syntax-ns#langString>
    }
  }
}
```

Fig. 5 Consulta #2.

La figura 6 muestra los resultados de la consulta número 2.

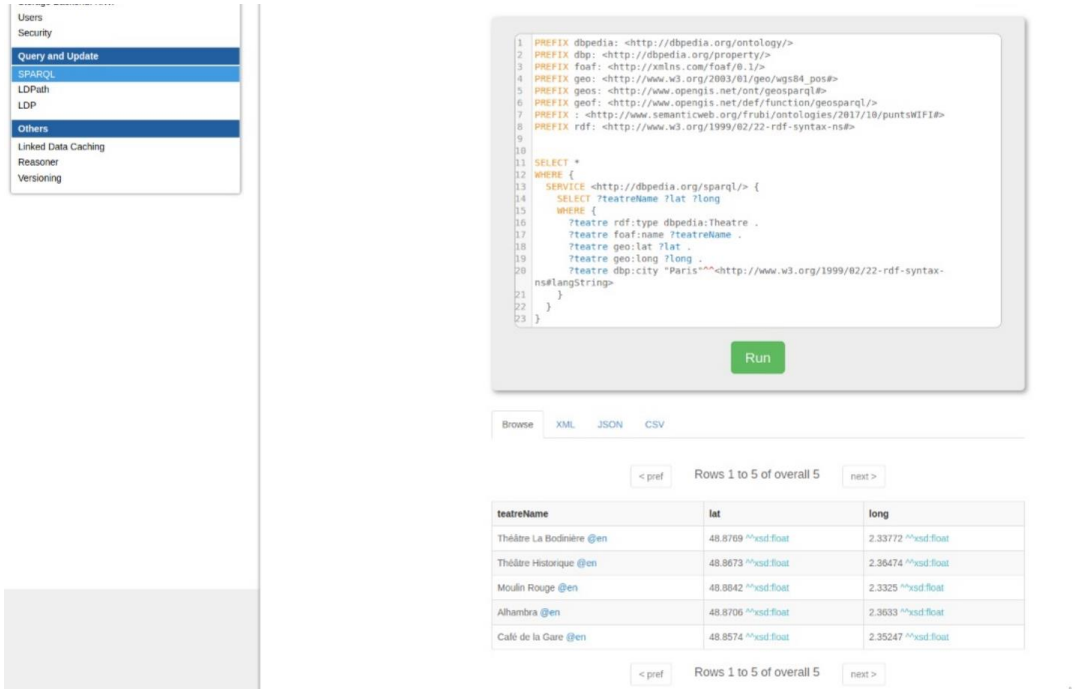

Fig. 6 Resultados de consulta #2.

A manera de comprobación, se hizo el rastreo de la petición de la consulta número 2 mediante WireShark. Como primer paso, se checó cuál es la IP de la máquina de donde provenía la consulta tal y como se muestra en la figura 7.

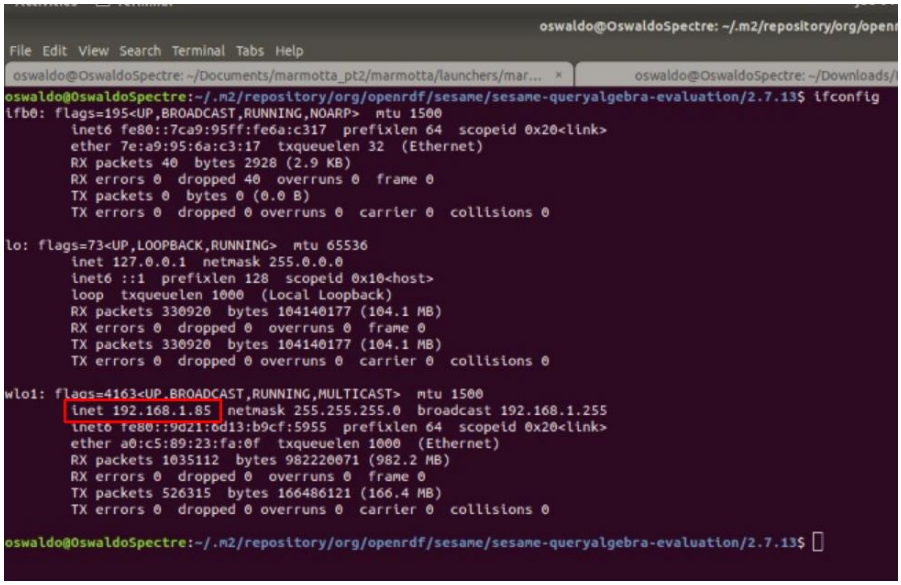

Fig. 7 Datos de la máquina local.

De la figura 7 se tiene que la interfaz es la wlo1 con la IP 192.168.1.85. Una vez obtenidos estos datos, se procedió a registrar el tráfico con WireShark y se corrió la consulta número 2 y así corroborar que la consulta se estuviera haciendo de forma federada. La figura 8 y 9 muestran que la consulta se dio de manera federada.

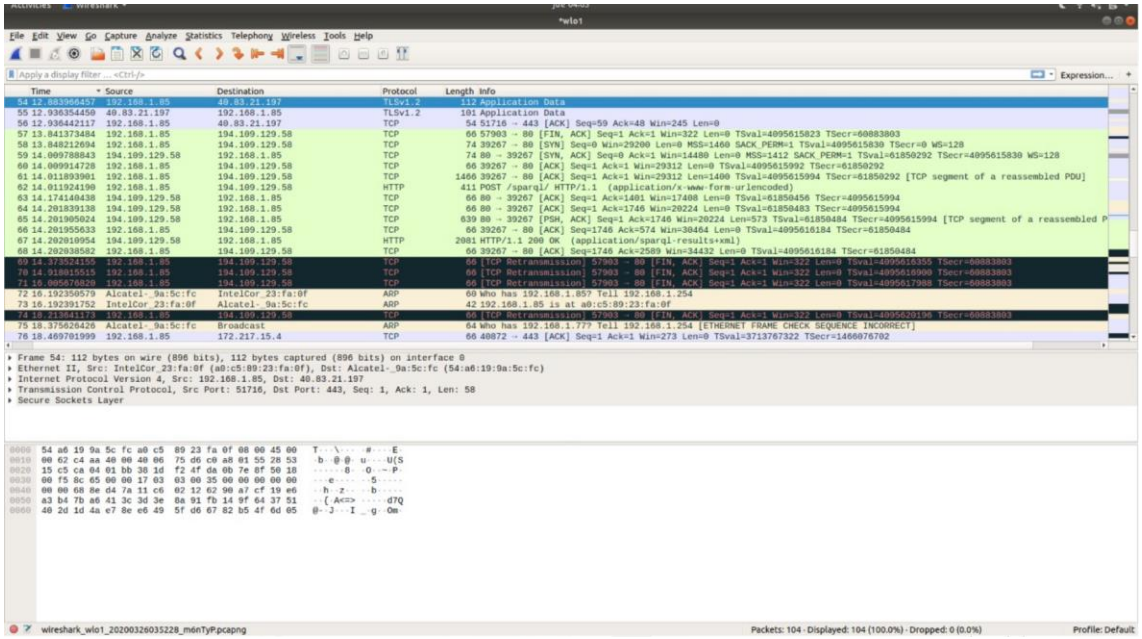

Fig. 8 Tráfico de consulta federada.

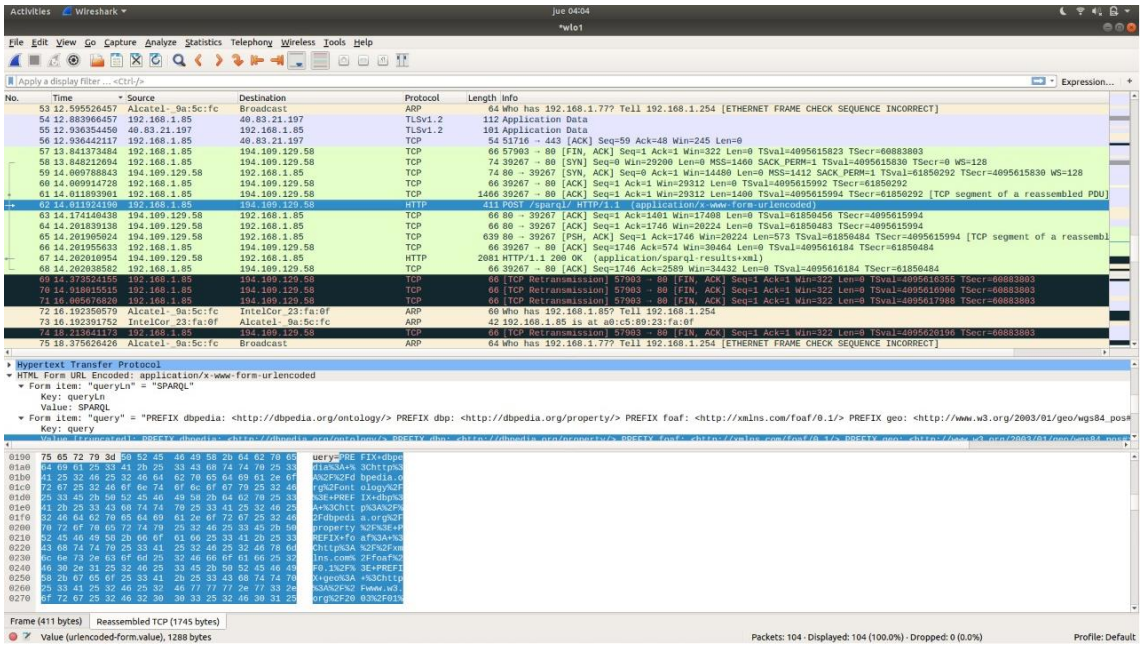

Fig. 9 Tráfico de consulta federada.

Y también para corroborar que el *triple store* si consultado si es el que está alojado en DBpedia, se procedió a insertar la dirección IP 194.109.129.58 en el buscador, tal y como se muestra en la figura 10.

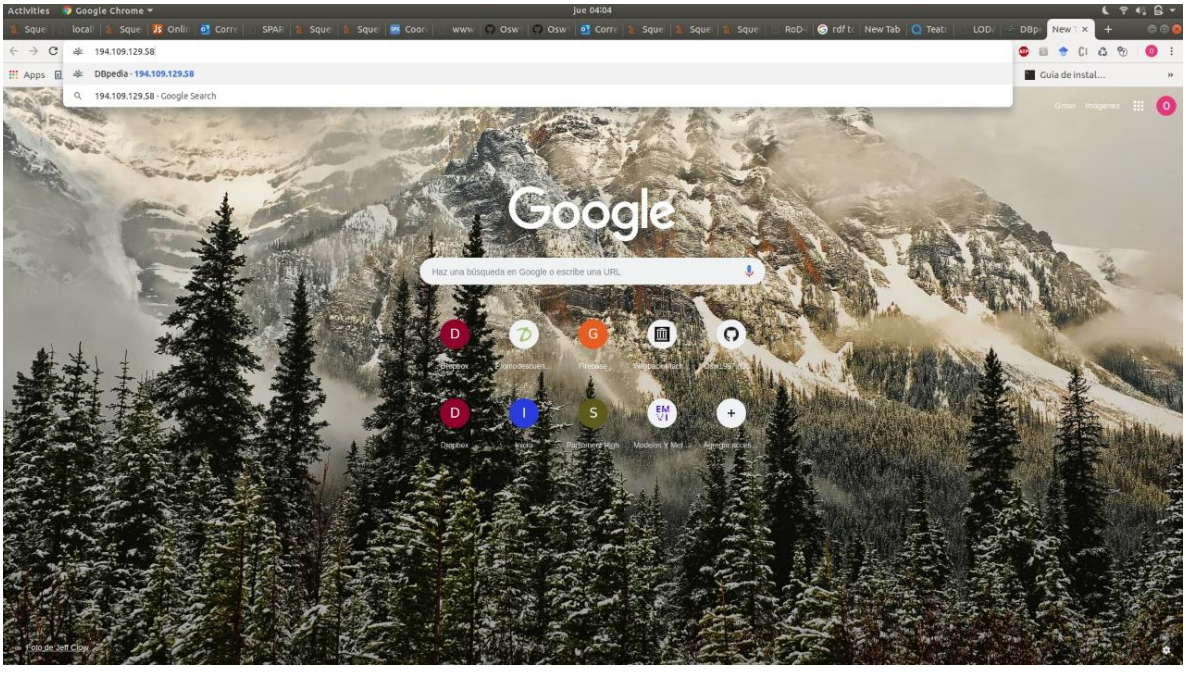

Fig. 10 Buscando la IP de DBpedia en el buscador Web.

Después de unos segundos, la página de DBpedia fue cargada, así como la figura 11 lo muestra.

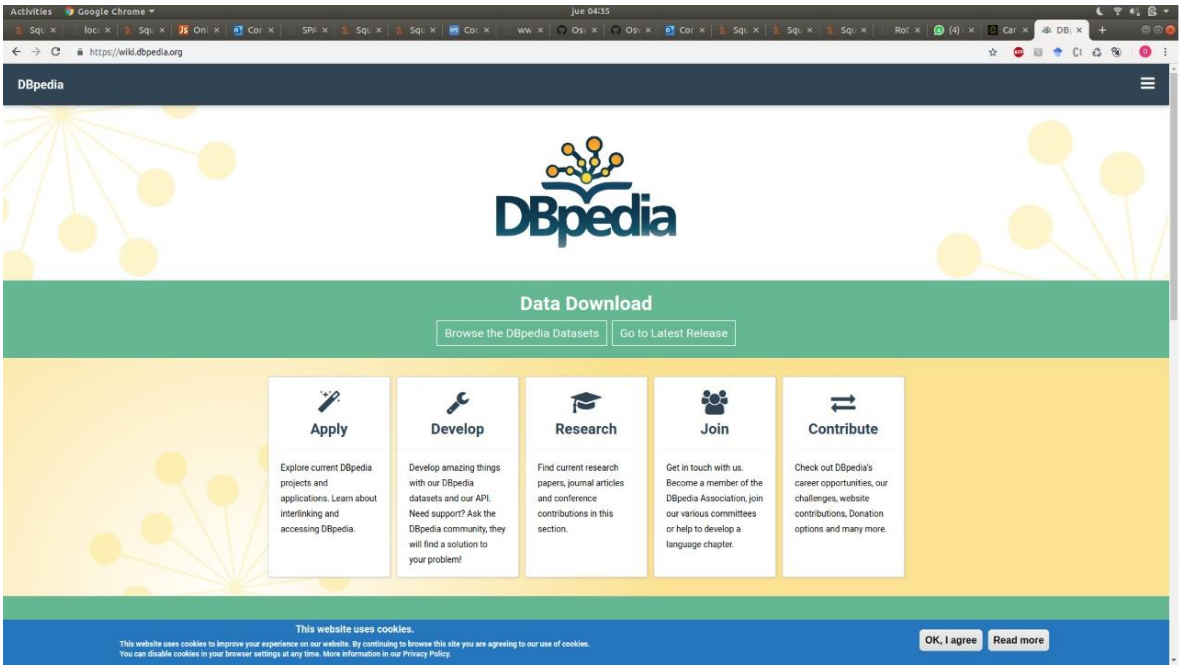

Fig. 11 Página de DBpedia cargada mediante la IP mostrada por Wireshark después de consulta federada.

Si bien el código de Apache Marmotta se encuentra en el siguiente enlace:

https://github.com/Osw1997/marmotta/tree/MARMOTTA-584

Aun no se ha hecho el *commit* ni el *push* al repositorio ya que se decidió hacer hasta culminar el proyecto y consecutivamente hacer el *pull request.*