Analizar los estándares SPARQL y GeoSPARQL para la implementación de consultas.

Objetivo: Comprender mediante diagramas o resumen cómo es que se lleva a cabo las consultas federadas y geoespaciales.

Resultados: Documento donde se describa cómo hacer una consulta en SPARQL y GeoSPARQL.
En el presente documento se tratan los estándares *SPARQL 1.1 Federated Query* y el *OGC GeoSPARQL – A Geographic Language for RDF Data* los cuales son documentos base para el desarrollo de la herramienta a implementar en el *triple store* Apache Marmotta.

SPARQL 1.1 Federated Query

En la Web Semántica se usan datos RDF para describir y representar la información en la Web mediante el uso de grafos etiquetados y dirigidos. El lenguaje SPARQL puede ser usado para realizar consultas sobre distintas fuentes de información ya sea por una base de datos RDF local o consultando la base de datos RDF mediante por medio de un *middleware*. El documento *SPARQL 1.1 Federated Query*, elaborado por la W3C, plantea la sintaxis y semántica del cómo debe de ser implementada la extensión para la ejecución de consultas sobre diferentes SPARQL *endpoints*.

Se usa la palabra *service* como palabra reservada para indicar al *triple store* que se va a llevar a cabo una consulta federada.

Los objetivos, planteados por la W3C en dicho documento, son

- La unión de datos federados en la Web.
- La capacidad de que usuario pueda dirigir una porción de una consulta a un SPARQL *endpoint* específico.
- Los datos retornados de la consulta parcial al procesador de consultas federadas deben ser combinados con los resultados de la consulta restante.

De manera gráfica, en la figura 1 se muestra lo anterior

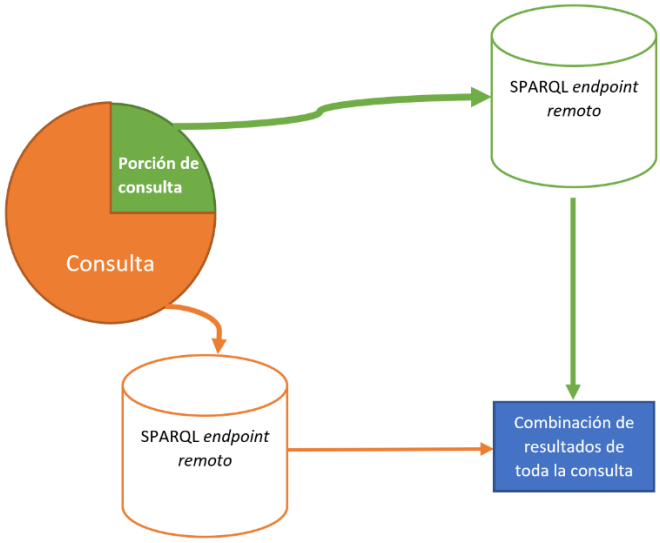

Fig. 1 Diagrama a bloques de una consulta federada

Ahora, con base al documento de esta sección, se presenta un análisis de la consulta federada propuesta en él. Las URIs solo son de fines demostrativos, es decir, los identificadores no redirigen a ninguna página real.

Base de datos local, la cual solo tiene una tripleta y está alojada en http://example.org/myfoaf.rdf

```
<http://example.org/myfoaf/I>
<http://xmlns.com/foaf/0.1/knows>
<http://example.org/people15> .
```

Fig. 2 Base de datos local (Una tripleta)

Base de datos remota la cual contiene 4 tripletas y está alojada en http://people.example.org/sparql

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

:people15  foaf:name    "Alice" .
:people16  foaf:name    "Bob" .
:people17  foaf:name    "Charles" .
:people18  foaf:name    "Daisy" .
```

Fig. 3 Base de datos remota (4 tripletas)

La consulta es la siguiente: Obtener los nombres de los conocidos (base de datos remota) por la persona #15(base de datos local)

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.org/myfoaf.rdf>
WHERE
{
  <http://example.org/myfoaf/I> foaf:knows ?person .
  SERVICE <http://people.example.org/sparql> {
    ?person foaf:name ?name . }
}
```

Fig. 4 Consulta SPARQL federada

Por lo que se entiende en la consulta

- Prefijo: http://xmlns.com/foaf/0.1/
- Columna por consultar: *name.*
- Base de datos local: http://example.org/myfoaf.rdf
- Sujeto a tratar: http://example.org/myfoaf/I
- Objeto a tratar: *foaf:knows*
- Variable por hacer coincidir: *?person*
- Base de datos remota: http://people.example.org/sparql

A grandes rasgos lo que se hace es consultar todos los nombres disponibles de la base de datos local y devolver los nombres asociados a dicha persona en cuestión. Cabe decir que los nombres de las personas conocidas están alojados en la base de datos remota.

Puesto que la base de datos local solo contiene una persona alojada en el Sistema, solo se hace la comparación de esa persona contra todos los de la base de datos remota. Tal y como se observa en la figura 2 y 3, la persona #15 solo conoce a una sola persona quien es “Alice”. Por tanto, el resultado que retorna la consulta es el siguiente.

Query Result:

name
"Alice"

Fig. 5 Resultado de la consulta federada

De manera ilustrativa, la obtención de nombres a consultar sería el bloque naranja de la consulta en la figura 1 mientras que la recuperación de los nombres conocidos por la consulta del bloque naranja corresponde a la consulta parcial, es decir, la porción verde.

Este análisis será de suma importancia en el desarrollo de la extensión para Apache Marmotta ya que será una referencia para saber que funciona de manera básica.

Por último, este análisis es solo una guía de cómo Apache Marmotta en conjunto con la extensión de consultas federadas podría funcionar. Esto implica que puede existir una ligera modificación debido al hecho que el documento *SPARQL 1.1 Federated Query* es solo una referencia de cómo es que los *triple store* deben de implementar esta extensión.

OGC GeoSPARQL – A Geographic Language for RDF Data

Este documento fue desarrollado por el *Open Geospatial Consortium* con el fin de servir como un estándar para la representación de y consultas de datos geoespaciales en la Web Semántica. GeoSPARQL define un vocabulario para representar los datos geoespaciales en RDF y la extensión para el lenguaje de consultas SPARQL.

El estándar GeoSPARQL se basa en un diseño modular

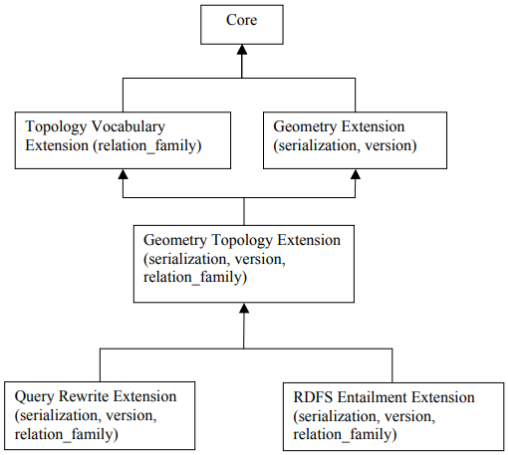

Fig. 6 Grafo de dependencias de requerimientos de clase para GeoSPARQL

Descripción de las clases que los confirman

- *Core:* Define clases RDFS/OWL para objetos espaciales.
- *Topology Vocabulary:* Define propiedades del RDF para consultas de relaciones topológicas entre objetos espaciales.
- *Geometry Extension:* Define un vocabulario de geometría y funciones de consultas no topológicas.
- *Geometry Topology Extension:* Define funciones de consulta topológicas para objetos geométricos.
- *RDFS entailment extension:* Define un mecanismo de coincidencia para tripletas RDF implícitas basados en RDF y semánticas RDFS.
- *Query rewrite extension:* Define reglas de transformación para determinar relaciones espaciales entre objetos espaciales según sus geometrías asociadas.

Así como en SPARQL, con GeoSPARQL se usan prefijos para las consultas. Ejemplo de prefijos son los siguientes

- ogc: http://www.opengis.net/
- geo: http://www.opengis.net/ont/geosparql#
- geof: http://www.opengis.net/def/function/geosparql/
- rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
- rdfs: http://www.w3.org/2000/01/rdf-schema#
- owl: http://www.w3.org/2002/07/owl#

En GeoSPARQL se hacen uso de características básicas para definir relaciones topológicas entre datos geoespaciales. Estas relaciones son

- *equals*
- *disjoint*
- *intersects*
- *touches*
- *within*
- *contains*
- *overlaps*
- *crosses*

Tal y como se menciona en los *papers Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL* y en el documento *OGC GeoSPARQL – A Geographic Language for RDF Data*, existen familias de relaciones que abordan las relaciones geoespaciales: *Egenhofer* y *RCC8*. La siguiente tabla comparativa muestra las equivalencias entre ellos

Características simples	*RCC8*	*Egenhofer*
Equals	Equals	Equals
Disjoint	disconnected	Disjoint
Intersects	~ disconnected	~ Disjoint
Touches	externally connected	Meet
Within	non-tangential proper part + tangential proper part	inside + coveredBy
Contains	non-tangential proper part inverse + tangential proper part inverse	contains + covers
Overlaps	partially overlapping	overlap

Tabla 1 - Equivalencias de características simples, RCC8 y Egenhofer

En GeoSPARQL se deben manejar propiedades estándar para características, geometrías.

Propiedades estándar para **geo:Feature**

- **geo:hasGeometry**
- **geo:hasDefaultGeometry**

Propiedades estándar para **geo:Geometry**

- **geo:dimension**
- **geo:coordinateDimension**
- **geo:spatialDimension**
- **geo:isEmpty**
- **geo:isSimple**
- **geo:hasSerialization**

Propiedades estándar para **RDFS *datatypes***

Y de igual manera, los requerimientos para serialización WKT también son plasmados

Tipo de datos RDFS

- **geo:wktLiteral**

Propiedades de serialización

- **geo:asWKT**

En otro caso, los requerimientos para serialización GML son

Tipo de datos RDFS

- **geo:gmlLiteral**

Propiedades de serialización

- **geo:asGML**

Así como se mencionó anteriormente, existen funciones de consulta no topológicas

Las siguientes funciones producirán un error si los argumentos de estas son inválidos. Un argumento inválido puede ser cualquiera de éstos:

- Tipo de dato incorrecto
- Valor de la variable geométrica es inválida
- Literal geométrica de un sistema de referencia espacial no es compatible con el sistema de referencia espacial usado para los cálculos.

Las funciones son las siguientes

- **geof:dist**
- **geof:buffer**
- **geof:convexHull**
- **geof:intersection**
- **geof:union**
- **geof:difference**
- **geof:symDifference**
- **geof:envelope**
- **geof:boundary**
- **geof:getsrid**

Para concluir el resumen de este estándar, al final, se presenta un anexo donde se muestran diversos ejemplos de consultas GeoSPARQL usando diferentes funciones y propiedades mencionadas anteriormente y sus resultados.