Semantic Web

In the last few years, the Web is moving from a "Web of documents" to a "Web of data". This is happening because big search engines such as Google are making use of not only the textual information that appears on a Web page (e.g., "James Bond has been played by Sean Connery. Sean Connery comes from Scotland.") but also the meaning of the terms that constitute this information (e.g., "James Bond", "Sean Connery", "Scotland", "plays", "comes from") in order to answer user queries. In the past, if we googled "Sean Connery", we would get back a list of URLs pointing us to Web pages that contain the string "Sean Connery". These Web pages would be ordered by their importance using various algorithms such as PageRank that Google employed. Today if google "Sean Connery" (try it!), Google understands that we are looking for information about the person called Sean Connery" and it will also give us a window where structured data about this person appear (his name, his occupation, his age, his Wikipedia entry, his date of birth, pictures of him, etc.). Thus, search engines like Google have started understanding the "meaning" or the "semantics" of the natural language text that appears on Web pages.

The technologies that have allowed search engines to improve in this exciting way have been under development since around 2000 under the terms "Semantic Web" and "Linked Data". On this page, you can read the influential paper "The Semantic Web" that kick-started this research area and was co-authored by the inventor of the Web and Turing Award Winner Sir Tim-Berners Lee. Europe has been a major player in this area with many scientists from Universities, Research Institutes and Companies contributing the the Semantic Web and Linked Data.

Semantic Web and Linked Data are based on the following technologies:

- URIs as a way to name entities, relationships etc. in the world.
- HTTP as the protocol to access information about these entities etc.
- RDF as the data model for representing information about the world.
- SPARQL as the query language for querying this information.

The Resource Description Framework

The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.

RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a "triple"). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications.

This linking structure forms a directed, labelled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations.

An RDF graph is made up of triples consisting of a subject, predicate and object. The simplest triple statement is a sequence of (subject, predicate, object) terms, separated by whitespace and terminated by '.' after each triple.

<http://example.org/#spiderman> <http://www.perceive.net/schemas/relationship/enemyOf> <http://example.org/#green-goblin> .

The subject of an RDF triple is either a uniform resource identifier (URI) or a blank node, both of which denote resources. The predicate is a URI which also indicates a resource, representing a relationship. The object is a URI, blank node or a Unicode string literal. As of RDF 1.1 resources are identified by IRI's. IRI is a generalization of URI.

The following example is describing a resource with statements "there is a Person identified by http://www.w3.org/People/EM/contact#me, whose name is Eric Miller, whose email address is e.miller123(at)example, and whose title is Dr.

The resource "http://www.w3.org/People/EM/contact#me" is the subject.

The objects are:

"Eric Miller" (with a predicate "whose name is"), mailto:e.miller123(at)example (with a predicate "whose email address is"), and "Dr." (with a predicate "whose title is"). The subject is a URI.

The predicates also have URIs. For example, the URI for each predicate:

"whose name is" is http://www.w3.org/2000/10/swap/pim/contact#fullName, "whose email address is" is http://www.w3.org/2000/10/swap/pim/contact#mailbox, "whose title is" is http://www.w3.org/2000/10/swap/pim/contact#personalTitle. In addition, the subject has a type (with URI http://www.w3.org/1999/02/22-rdf-syntax-ns#type), which is person (with URI http://www.w3.org/2000/10/swap/pim/contact#Person).

Therefore, the following "subject, predicate, object" RDF triples can be expressed:

<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/2000/10/swap/pim/contact#fullName> "Eric Miller" .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/2000/10/swap/pim/contact#mailbox> <mailto:e.miller123(at)example> .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/2000/10/swap/pim/contact#personalTitle> "Dr." .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2000/10/swap/pim/contact#Person> .

For more information on the RDF model you can visit the following links:

- RDF Primer
- Turtle
- Introduction to RDF lecture

Ontologies

What is a Vocabulary?

On the Semantic Web, vocabularies define the concepts and relationships (also referred to as "terms") used to describe and represent an area of concern. Vocabularies are used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms. In practice, vocabularies can be very complex (with several thousands of terms) or very simple (describing one or two concepts only). There is no clear division between what is referred to as "vocabularies" and "ontologies". The trend is to use the word "ontology" for more complex, and possibly quite formal collection of terms, whereas "vocabulary" is used when such strict formalism is not necessarily used or only in a very loose sense. Vocabularies are the basic building blocks for inference techniques on the Semantic Web.

What are Vocabularies Used For?

The role of vocabularies on the Semantic Web are to help data integration when, for example, ambiguities may exist on the terms used in the different data sets, or when a bit of extra knowledge may lead to the discovery of new relationships. Consider, for example, the application of ontologies in the field of health care. Medical professionals use them to represent knowledge about symptoms, diseases, and treatments. Pharmaceutical companies use them to represent information about drugs, dosages, and allergies. Combining this knowledge from the medical and pharmaceutical communities with patient data enables a whole range of intelligent applications such as decision support tools that search for possible treatments; systems that monitor drug efficacy and possible side effects; and tools that support epidemiological research.

Another type of example is to use vocabularies to organize knowledge. Libraries, museums, newspapers, government portals, enterprises, social networking applications, and other communities that manage large collections of books, historical artifacts, news reports, business glossaries, blog entries, and other items can now use vocabularies, using standard formalisms, to leverage the power of linked data.

It depends on the application how complex vocabularies they use. Some applications may decide not to use even small vocabularies, and rely on the logic of the application program. Some application may choose to use very simple vocabularies like the one described in the examples section below, and let a general Semantic Web environment use that extra information to make the identification of the terms. Some applications need an agreement on common terminologies, without any rigor imposed by a logic system. Finally, some applications may need more complex ontologies with complex reasoning procedures. It all depends on the requirements and the goals of the applications.

To satisfy these different needs, W3C offers a large palette of techniques to describe and define different forms of vocabularies in a standard format. These include RDF and RDF Schemas, Simple Knowledge Organization System (SKOS), Web Ontology Language (OWL),

and the Rule Interchange Format (RIF). The choice among these different technologies depend on the complexity and rigor required by a specific application.

Examples

A general example may help. A bookseller may want to integrate data coming from different publishers. The data can be imported into a common RDF model, eg, by using converters to the publisher's databases. However, one database may use the term "author", whereas the other may use the term "creator". To make the integration complete, and extra definition should be added to the RDF data, describing the fact that the relationship described as "author" is the same as "creator". This extra piece of information is, in fact, a vocabulary (or an ontology), albeit an extremely simple one. In a more complex case the application may need a more detailed ontology as part of the extra information. This may include formal description on how authors are to be uniquely identified (eg, in a US setting, by referring to a unique social security number), how the terms used in this particular application relate to other datasets on the Web (eg, Wikipedia or geographic information), how the term "author" (or "creator") can be related to terms like "editors", etc.

For more information on Ontologies you can visit the following links:

- RDF Schema
- Introduction to RDF Schema

SPARQL

SPARQL 1.1

RDF is a directed, labeled graph data format for representing information in the Web. This specification defines the syntax and semantics of the SPARQL query language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

Most forms of SPARQL query contain a set of triple patterns called a basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph.

The example below shows a SPARQL query to find the title of a book from the given data graph. The query consists of two parts: the SELECT clause identifies the variables to appear in the query results, and the WHERE clause provides the basic graph pattern to match against the data graph. The basic graph pattern in this example consists of a single triple pattern with a single variable (?title) in the object position.

Data:

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
<http://example.org/book/book1> <http://example.org/ns#price> 28 .
```

Query:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Results:

```
"SPARQL Tutorial"
```

Graph pattern matching produces a solution sequence, where each solution has a set of bindings of variables to RDF terms. SPARQL FILTERs restrict solutions to those for which the filter expression evaluates to TRUE.

SPARQL FILTERs can for example restrict on arithmetic expressions.

Data:

<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
<http://example.org/book/book1> <http://example.org/ns#price> 28 .
<http://example.org/book/book2> <http://purl.org/dc/elements/1.1/title> "Semantic Web" .
<http://example.org/book/book2> <http://example.org/ns#price> 32 .

Query:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
        FILTER (?price < 30.5)
        ?x dc:title ?title . }
```

Results:

"SPARQL Tutorial" | 28

For more information on SPARQL 1.1 you can visit the following links:

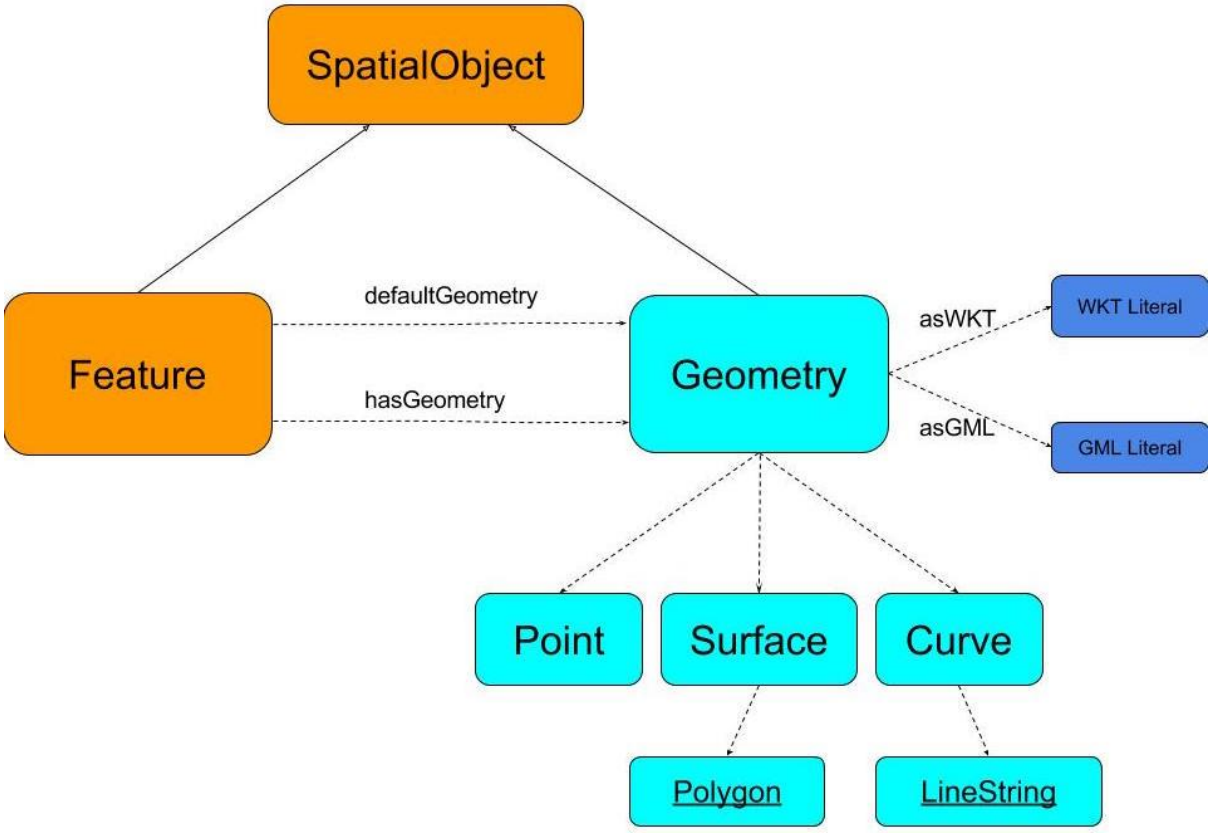
- SPARQL 1.1
- SPARQL Lecture 1
- SPARQL Lecture 2

GeoSPARQL

The OGC GeoSPARQL standard supports representing and querying geospatial data on the Semantic Web. GeoSPARQL defines a vocabulary for representing geospatial data in RDF, and it defines an extension to the SPARQL query language for processing geospatial data. In addition, GeoSPARQL is designed to accommodate systems based on qualitative spatial reasoning and systems based on quantitative spatial computations.

In particular, GeoSPARQL provides for:

- A small topological ontology in RDFS/OWL for representation using Geography Markup Language (GML) and well-known text (WKT) literals as shown in the Figure below.
- Simple Features, RCC8, and DE-9IM (a.k.a. Egenhofer) topological relationship vocabularies and ontologies for qualitative reasoning.
- A SPARQL query interface using a set of topological SPARQL extension functions for quantitative reasoning.
- A set of Rule Interchange Format (RIF) Core inference rules for query transformation and interpretation.



The GeoSPARQL vocabulary can be used to construct SPARQL graph patterns for querying appropriately modeled geospatial data. RDFS and OWL vocabulary have both been used so that the vocabulary can be understood by systems that support only RDFS entailment and by systems that support OWL-based reasoning. Two main classes are defined: geo:SpatialObject and geo:Feature. The class geo:SpatialObject is defined by the following:

```
geo:SpatialObject a rdfs:Class,
                owl:Class;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "Spatial Object"@en;
   rdfs:comment "The class Spatial Object represents everything that
            can have a spatial representation. It is superclass
            of feature and geometry"@en .
```

The class geo:Feature is defined by the following:

```
geo:Feature a rdfs:Class,
           owl:Class;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "Feature"@en;
   rdfs:subClassOf geo:SpatialObject;
   owl:disjointWith geo:Geometry;
   rdfs:comment "This class represents the top-level feature type.
            This class is equivalent to GFI_Feature defined in
            ISO 19156, and it is superclass of all feature
            types."@en .
```

Also, a single root geometry class is defined: geo:Geometry with properties for describing geometry data and associating geometries with features. The class geo:Geometry is defined by the following:

```
geo:Geometry a rdfs:Class,
            owl:Class;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "Geometry"@en;
   rdfs:subClassOf geo:SpatialObject;
   owl:disjointWith geo:Feature;
   rdfs:comment "The class represents the top-level geometry type.
            This class is equivalent to the UML class GM_Object
            defined in ISO 19107, and it is superclass of all
            geometry types."@en .
```

The property geo:hasGeometry is used to link a feature with a geometry that represents its spatial extent. A given feature may have many associated geometries.

```
geo:hasGeometry a rdf:Property,
                owl:ObjectProperty;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "has Geometry"@en;
   rdfs:comment "A spatial representation for a given feature."@en;
   rdfs:domain geo:Feature;
   rdfs:range geo:Geometry .
```

The property geo:hasDefaultGeometry is used to link a feature with its default geometry.

The default geometry is the geometry that should be used for spatial calculations in the absence of a request for a specific geometry (e.g. in the case of query rewrite).

```
geo:hasDefaultGeometry a rdf:Property,
                owl:ObjectProperty;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "has Default Geometry"@en;
   rdfs:comment "The default geometry to be used in spatial
          calculations,usually the most detailed geometry."@en;
   rdfs:subPropertyOf geo:hasGeometry;
   rdfs:domain geo:Feature;
   rdfs:range geo:Geometry .
```

GeoSPARQL does not restrict the cardinality of the geo:hasDefaultGeometry property. It is thus possible for a feature to have more than one distinct default geometry or to have no default geometry. This situation does not result in a query processing error; SPARQL graph pattern matching simply proceeds as normal. For representing geometry data in RDF we can use the WKT or GML serializations. The WKT serialization is defined by Simple Features [ISO 19125-1], while GML is defined by Geography Markup Language Encoding Standard [OGC 07-036].

```
geo:wktLiteral a rdfs:Datatype;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "Well-known Text Literal"@en;
   rdfs:comment "A Well-known Text serialization of a geometry
          object."@en .
geo:gmlLiteral a rdfs:Datatype;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "GML literal"@en;
   rdfs:comment "The datatype of GML literal values"@en .
```

All RDFS Literals of type geo:wktLiteral shall consist of an optional URI identifying the coordinate reference system followed by Simple Features Well Known Text (WKT) describing a geometric value. Valid geo:wktLiterals are formed by concatenating a valid, absolute URI as defined in [RFC 2396], one or more spaces (Unicode U+0020 character) as a separator, and a WKT string as defined in Simple Features [ISO 19125-1]. For geo:wktLiterals, the beginning URI identifies the spatial reference system for the geometry. The OGC maintains a set of CRS URIs under the http://www.opengis.net/def/crs/ namespace. This leading spatial reference system URI is optional. In the absence of a leading spatial reference system URI, the following spatial reference system URI will be assumed: <http://www.opengis.net/def/crs/OGC/1.3/CRS84>. This URI denotes WGS 84 longitude-latitude. The URI <http://www.opengis.net/def/crs/OGC/1.3/CRS84> shall be assumed as the spatial reference system for geo:wktLiterals that do not specify an explicit spatial reference system URI. Valid geo:gmlLiterals are formed by encoding geometry information as a valid element from the GML schema that implements a subtype of GM_Object. For example, in GML 3.2.1 this is every element directly or indirectly in the substitution group of the element {http://www.opengis.net/ont/gml/3.2}AbstractGeometry. In GML 3.1.1 and GML 2.1.2 this is every element directly or indirectly in the substitution group of the element {http://www.opengis.net/ont/gml}_Geometry. All geo:gmlLiterals shall consist of a

valid element from the GML schema that implements a subtype of GM_Object as defined in [OGC 07-036]. To link a geometry with a WKT/GML serialization we use the properties geo:asWKT and geo:asGML. Those two properties are defined below:

```
geo:asWKT a rdf:Property,
        owl:DatatypeProperty;
   rdfs:subPropertyOf geo:hasSerialization;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "as WKT"@en;
   rdfs:comment "The WKT serialization of a geometry."@en;
   rdfs:domain geo:Geometry;
   rdfs:range geo:wktLiteral .
geo:asGML a rdf:Property;
   rdfs:subPropertyOf geo:hasSerialization;
   rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
   rdfs:label "as GML"@en;
   rdfs:comment "The GML serialization of a geometry."@en;
   rdfs:domain geo:Geometry;
   rdfs:range geo:gmlLiteral .
```

Below we showcase two examples of how we can represent a point geometry in both serialization formats:

```
"<http://www.opengis.net/def/crs/EPSG/0/4326>
Point(33.95 -83.38)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
```

```
"<gml:Point
   srsName=\"http://www.opengis.net/def/crs/OGC/1.3/CRS84\"
   xmlns:gml=\"http://www.opengis.net/ont/gml\">
   <gml:pos>-83.38 33.95</gml:pos> </gml:Point>"^^<http://www.opengis.net/ont/geosparql#gmlLiteral>
```

To create spatial filters, GeoSPARQL offers the following functions:

```
geof:sfEquals(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean

geof:sfDisjoint(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean

geof:sfIntersects(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean

geof:sfTouches(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean

geof:sfCrosses(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean

geof:sfWithin(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean
```

```
geof:sfContains(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean

geof:sfOverlaps(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral
): xsd:boolean
```

The following example SPARQL query could help model the question "What is within the bounding box defined by 38.913574°N 77.089005°W and 38.886321°N 77.029953°W?"

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?what
WHERE {
  ?what geo:hasGeometry ?geometry .
  ?geometry geo:asWKT ?wkt .

  FILTER(geof:sfWithin(?wkt,
    "POLYGON((
       -77.089005 38.913574,
       -77.029953 38.913574,
       -77.029953 38.886321,
       -77.089005 38.886321,
       -77.089005 38.913574
    ))"^^geo:wktLiteral))
}
```

For more information on GeoSPARQL you can visit the following links:

- GeoSPARQL standard
- Linked Geospatial Data Lecture