Lenguajes de Ontologías

Web Semántica

Javier Béjar

ECSDI - 2023/2024 2Q

CS-GEI-FIB 

Objetivos del tema

- La Web Semántica
- Recursos como información, XML como lenguaje de representación
- RDF, RDFS - Tripletas - Grafos Semánticos
- OWL como lenguaje de ontologías en la Web
- Linked Data, Triplestores, SPARQL

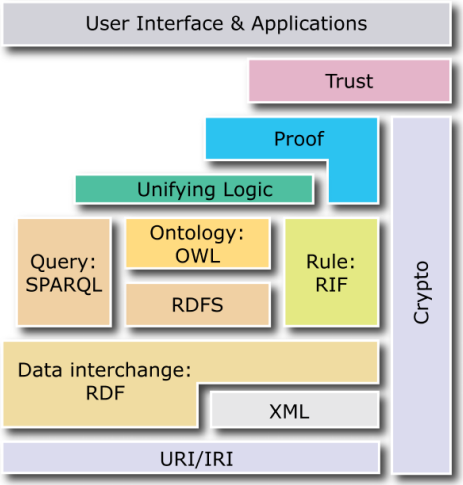
Información y la WWW

- La WWW es una inmensa fuente de información
- **Problema**: está pensada para ser utilizada por personas
  - Lenguaje orientado a la presentación (HTML)
- Asume que los usuarios finales pueden:
  - Reconocer el significado del contenido y sacar conclusiones
  - Inferir nuevo conocimiento utilizando el contexto
  - Entender el conocimiento relacionado
- La evolución de la Web 2.0 a la Web 3.0 pretende que las máquinas aprovechen también esa información

- El que servicios/agentes compartan/adquieran información requiere lenguajes para expresarla
- Este lenguaje:
  - Debe permitir representar el conocimiento ontológico de manera sencilla
  - Debe tener una semántica axiomatizable (eg. DL)
  - Debe permitir el razonamiento automático
  - El coste computacional del razonamiento debe ser razonable

La Web Semántica

- El poder utilizar la información de la web necesita de un lenguaje de representación uniforme
- El proyecto **W3C Semantic Web** pretende establecer el conjunto de estándares necesarios para hacerlo realidad
- Todo el conocimiento en la web podrá ser descrito usando este esquema
- Utilizados para la representación, consumo y compartición de información entre servicios y agentes
- El objetivo es construir los servicios web semánticos sobre esta tecnología


User Interface & Applications
Trust
Proof
Unifying Logic
Query:
SPARQL
Ontology:
OWL
RDFS
Rule:
RIF
Crypto
Data interchange:
RDF
XML
URI/IRI

- Un primer paso para poder representar de manera uniforme la información en la web es poder referenciarla
- En terminología de la web semántica, todo lo que se puede describir en la web es un recurso
- Todo recurso está identificado mediante un **URI** (Uniform Resource Identifier)

- Un URI junta dos conceptos:
  - URN (Uniform Resource Name): Identificador único que permite referenciar un recurso (pero no dice donde esta) (p.ej.: un ISBN de un libro)
  - URL (Uniform Resource Locator): Un identificador único que permite indicar como acceder al recurso

- URI, sintaxis

  `scheme:[//authority][/path][?query][#fragid]`

- Elementos:
  - `scheme`, tipo de URI (http, https:, mailto, imap...)
  - `authority`, habitualmente un servidor
  - `path`, ruta de acceso dentro del servidor
  - `query`, parámetros adicionales
  - `fragid`, una parte constituyente del recurso

- XML es un lenguaje para definir lenguajes
- Estándar para interoperación/representación (W3C)
- Mecanismo de almacenamiento de información uniforme
- Lenguaje de intercambio entre aplicaciones (eg: servicios/agentes)
- Puede definir esquemas de representación del conocimiento (lenguajes de ontologías)
- Permite su traducción a otros esquemas (XSLT)

- XML se ha extendido para referenciar definiciones
- Permite construir repositorios de definiciones reutilizables (namespaces)
- Se pueden tomar como vocabularios para dominios concretos

Ejemplo

```
<direccion xmlns="http://definicion_de_direccion">
  <nombre> Juan </nombre>
  ...
</direccion>
```

- Extension de XML usado como lenguaje de definición tipos de datos
- Basado en un conjunto de tipos primitivos (XML Schema Datatypes, XSD)
- Podemos definir nuevos tipos de datos a partir de estos

ejemplo.xml

```
<xsd:complexType name="direccion" >
<xsd:sequence>
<xsd:element name="nombre"   type="xsd:string"/>
<xsd:element name="calle" type="xsd:string"/>
<xsd:element name="ciudad"   type="xsd:string"/>
...
</xsd:sequence>
</xsd:complexType>
```

Web Semántica - RDF

- XML puede servir como base para un lenguaje de representación del conocimiento
- Los **namespaces** permiten la posibilidad de definiciones compartibles
- Necesitamos además una semántica asociada a esas definiciones (separar la estructura de los datos de su significado)
- RDF es un estándar del W3C definido sobre XML que permite representar información sobre recursos

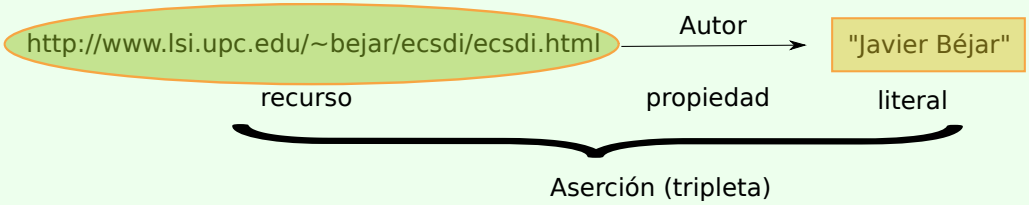
- El elemento principal de RDF es la afirmación (statement)
- Una afirmación define lo que conocemos sobre un recurso:
  - Relacionándolo con otro recurso
  - Declarando los valores de las propiedades de un recurso
- Una afirmación recibe también el nombre de **tripleta** (triplet)

- Una tripleta está formada por:
  - **Sujeto**, un recurso identificado por su URI
  - **Predicado**, una propiedad (también denotada por una URI)
  - **Objeto**, un recurso o literal con el que se define la relación

- **Recursos** (resources): Cualquier cosa que se pueda referenciar (tiene un URI)
- **Propiedades** (properties): Características, atributos o relaciones que se pueden usar para describir recursos, formarán parte de un vocabulario identificado por una URI
- **Literales**: Valores que pertenecen a un tipo de datos primitivo (habitualmente uno de los definidos en xsd)

- **Contenedores/colecciones**: permiten referenciar grupos de recursos
- **Nodos Blancos**: recursos sin identificador usados para agrupar información o como variables en ciertas expresiones

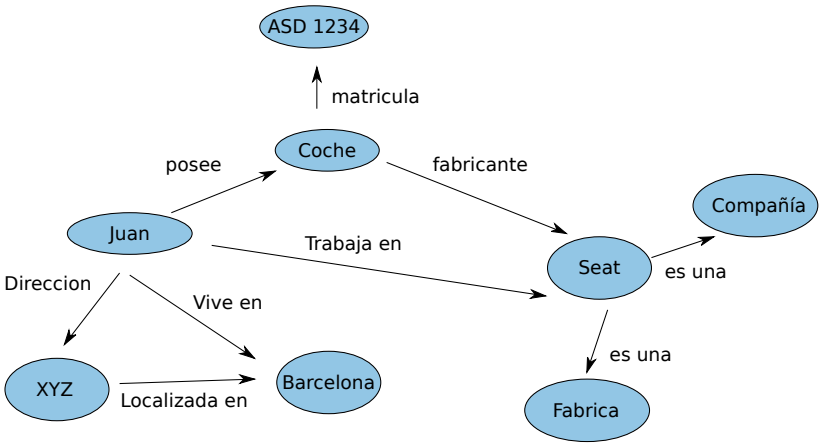
Gráficamente:



en RDF:

```
<rdf:RDF xmlns:s="URI definiciones (propiedad Autor)">
 <rdf:Description about="http:\\ ...\ecsdi.html">
  <s:Autor> Javier Bejar</s:Autor>
 </rdf:Description>
</rdf:RDF>
```

- Las tripletas representan el conocimiento como un grafo
- Los diferentes conceptos representados están interconectados mediante sus relaciones



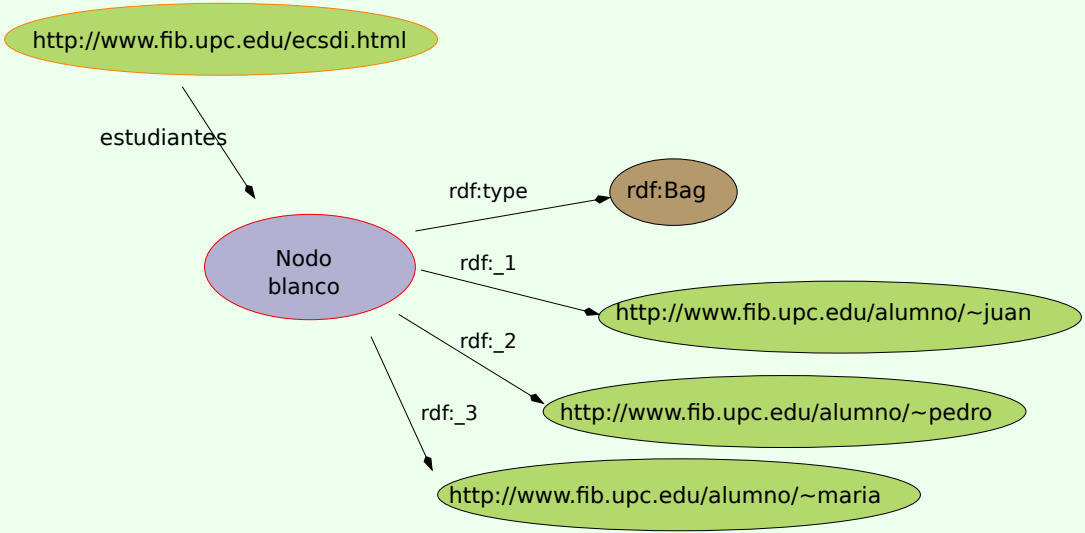
- Tipos y propiedades serán recursos que tendrán un URI
- Indicamos el tipo de un recurso usando la etiqueta `rdf:type`

```
juan rdf:type estudiante.
```

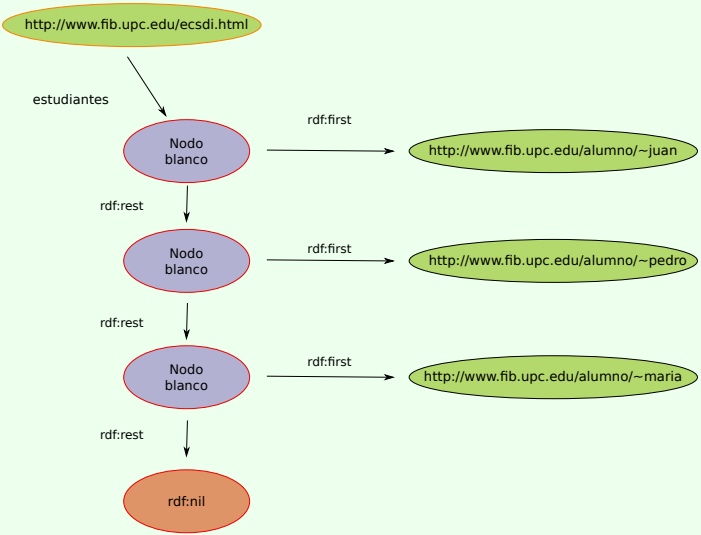
- Podemos indicar que algo es una propiedad con `rdf:property` (que usaremos para enlazar recursos/literales)

```
vive_en rdf:type rdf:property.
juan vive_en Barcelona.
```

- RDF tiene además la capacidad de definir contenedores y colecciones (grupos de recursos)
- **Contenedores** (no cerrados)
  - `rdf:Bag`: Conjunto no ordenado de recursos o literales
  - `rdf:Sequ`: lista ordenada de recursos o literales
  - `rdf:Alt`: Alternativas para el valor de una propiedad
- **Colecciones** (cerradas)
  - `rdf:List`: Listas enlazadas y cerradas de recursos (`rdf:first`, `rdf:next`, `rdf:nil`)


http://www.fib.upc.edu/ecsdi.html
estudiantes
Nodo
blanco
rdf:type
rdf:Bag
rdf:_1
http://www.fib.upc.edu/alumno/~juan
rdf:_2
http://www.fib.upc.edu/alumno/~pedro
rdf:_3
http://www.fib.upc.edu/alumno/~maria

RDF - ejemplo de coleccion



- El estándar W3C de RDF define su vocabulario

Clases:	`rdf:Property`, `rdf:Statement`, `rdf:XMLLiteral`, `rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdf:List`
Propiedades:	`rdf:type`, `rdf:subject`, `rdf:predicate`, `rdf:object`, `rdf:first`, `rdf:rest`, `rdf:_n`, `rdf:value`
Recursos:	`rdf:nil`

https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/

- La forma de serializar RDF es habitualmente XML, pero hay otras alternativas que lo hacen más legible
- En la práctica se ha de leer código en RDF y obviamente no es fácil leerlo directamente en XML
- Varias notaciones alternativas:
  - **TURTLE**(Terse RDF Triple Language)
  - N-triples, JSON-LD, RDFa. TriG, N-quads

- Utiliza definición de prefijos para aumentar la legibilidad

  `@prefix rdf:<http://www.w3.org/...`

- Permite definir una base común para todos los recursos

  `@base rec:<http://mis.recursos.org/>`

- Substitute `rdf:type` por `a`

- Permite encadenar aserciones sobre un mismo sujeto (`;`)

- Posee una sintaxis simple para los `xsd` y la denotación del idioma de las etiquetas (`@`)

```
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix pers:<http://personas.org>.
@prefix org:<http://organizacion.org>.

pers:juan a pers:persona.

pers:juan pers:edad "33"^^xsd:integer;
          pers:nombre "Juan"^^xsd:string.

org:UPC a org:universidad;
     rdf:label "Technical University of Catalonia"@en.

pers:juan org:estudia_en org:UPC.
```

Web Semántica - RDFS

- RDF solo nos permite establecer afirmaciones sobre la información
- No permite definir la estructura de la información
- Por ejemplo, podemos decir:

```
pers:juan rdf:type pers:estudiante.
```

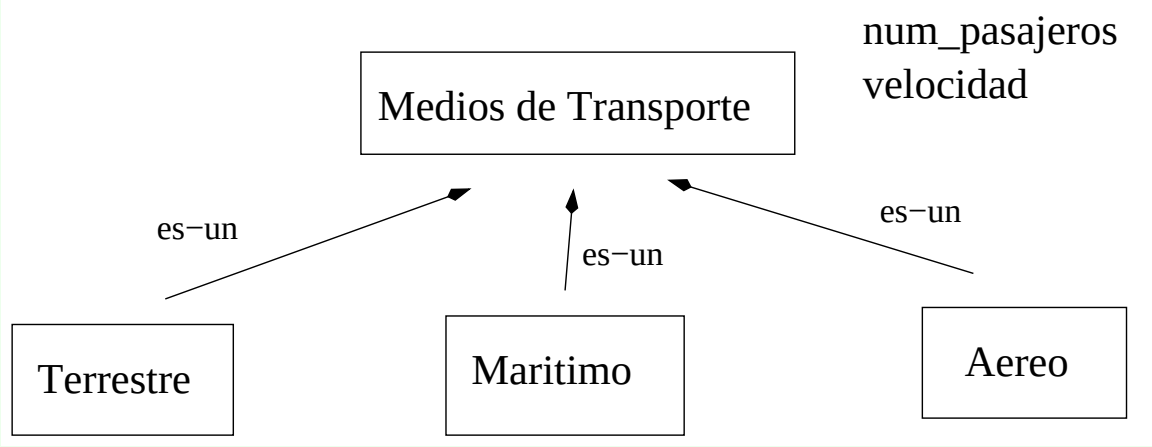
- Pero no decimos qué es un `estudiante`
- Asumimos implícitamente que es una clase

- RDF Schema (RDFS) es una extensión de RDF que permite declarar clases, propiedades de clases y relaciones entre clases
- RDFS es una colección de recursos RDF que permite describir propiedades de otros recursos RDF (espacio de nombres **rdfs**)
- Es un sistema de clases a partir de las cuales se pueden definir otras clases vía instanciación
- Podemos extenderlo, añadiendo nuevas definiciones
- Estándar W3C: http://www.w3.org/TR/rdf-schema/

- `rdfs:Resource`: Todo lo expresable en RDF es de esta clase
- `rdfs:Class`: Define conjuntos de recursos
- `rdfs:Literal`: Toda expresión que pertenece a un tipo primitivo (números, strings...)
- `rdfs:Datatype`: Clase de los tipos de datos primitivos
- `rdfs:Container`: Clase de los contenedores
- `rdf:Property`: Toda propiedad de una clase es una instancia de esta clase
- `rdf:type`: Indica que un recurso es un miembro de una clase (instancia-de)

- `rdfs:subClassOf`: Es una propiedad que permite definir la relacion clase/subclase. Su rango es siempre una clase. Es transitiva. Permite herencia de propiedades
- `rdfs:Subproperty`: Indica que una propiedad es una especialización de otra (instancia de `rdf:Property`)
- `rdfs:domain`: Dominio de una propiedad
- `rdfs:range`: Rango de una propiedad

- Metadatos
  - `rdfs:label`: Etiqueta asignada a un recurso
  - `rdfs:comment`: Comentario descriptivo
  - `rdfs:SeeAlso`: Información adicional sobre el recurso


num_pasajeros
velocidad
Medios de Transporte
es−un
es−un
es−un
Terrestre
Maritimo
Aereo

```
@prefix rdf:<RDF-vocabulario>.
@prefix rdf:<RDFS-vocabulario>.
@prefix xsd:<XML-Datatype>.
@base mt:<http://transporte.org/>.

mt:Medios_de_Transporte a rdfs:Class.

mt:Maritimo rdfs:SubClassOf mt:Medios_de_Transporte.
mt:Terrestre rdfs:SubClassOf mt:Medios_de_Transporte.
...
mt:num_pasajeros a rdf:Property.

mt:num_pasajeros rdfs:domain mt:Medios_de_Transporte;
                 rdfs:range xsd:integer.
```

Web Semántica - OWL

- RDF+RDFS aún no tienen la potencia expresiva necesaria
- No hay semántica para contenedores, colecciones
- No permite hacer comprobaciones para las propiedades de dominio y rango
- Solo se pueden expresar relaciones binarias

- No permite definir las características de las propiedades (transitiva, reflexiva...)
- No permite restricciones de cardinalidad
- No permite definir clases a partir de otras (unión, intersección) o indicar restricciones sobre clases (disjuntas)

- Diferentes esfuerzos para definir un lenguaje de ontologías sobre RDFS llevan hasta **OWL 2**
- Extienden los elementos de RDFS para:
  - Tener primitivas de frames y Description Logic
  - Tener una semántica definida (DL/Primer Orden)
  - Poder construir demostradores para soportar razonamiento automático
- Define diferentes sintaxis para representar las ontologías
- Estandar W3C: http://www.w3.org/TR/owl2-overview/

- La ontología pasa a ser un objeto de primer orden $\Rightarrow$ un recurso
- Las ontologías son importables y extensibles
- Las ontologías se pueden anotar con metadatos
- Podemos establecer restricciones sobre clases y propiedades
- Podemos construir clases a partir de otras clases
- Podemos declarar axiomas de deducción sobre la ontología

- La extensión esta definida en un nuevo espacio de nombres (`owl`)
- Las ontologias se declaran como objetos de tipo ontología: `owl:Ontology`
- Podemos importar otras ontologias: `owl:imports`
- Podemos utilizar las definiciones de tipos en XML-Schema-Datatype (namespace `xsd`)
- Se definen propiedades para anotar ontologías

- Definición de clases: `owl:Class`
- Objeto inicial de la jerarquía `owl:Thing`
- Restricciones sobre clases:
  - `owl:ComplementOf`: Complementario de otra clase
  - `owl:DisjointWith`: Declaración de clases disjuntas
  - `owl:UnionOf`: Clase declarada como unión de clases
  - `owl:IntersectionOf`: Clase declarada como intersección de clases
  - `owl:SameClassAs`: Sinónimo de otra clase

```
Animales a owl:Ontology.

Animal a owl:Class.

ATerrestre rdfs:SubClassOf Animal.
AAcuatico rdfs:SubClassOf Animal.
AAereo rdfs:SubClassOf Animal.
AAereo owl:DisjointWith ATerrestre.

AAnfibio rdfs:SubClassOf Animal.
AAnfibio owl:IntersectionOf [rdfs:first ATerrestre;
                             rdfs:rest [rdfs:first AAcuatico;
                             rdf:rest rdfs:nil]].
```

- Definicion de relaciones: `owl:ObjectProperty`
- Definicion de atributos: `owl:DataTypeProperty`
- Características de las relaciones/propiedades/instancias:
  - `owl:UniqueProperty`: Cardinalidad 1
  - `owl:TransitiveProperty`, `owl:SymmetricProperty`, `owl:InverseOf`: Transitividad, simetría, inversa
  - `owl:restriction`: Restricciones (p. ej: de cardinalidad `owl:Cardinality`)
  - `owl:sameAs`, `owl:differentFrom`: Individuos iguales/diferentes
  - ...

```
Hombre a owl:Class.
Mujer a owl:Class.
Persona a owl:Class;  owl:UnionOf (Hombre Mujer).

Nombre a rdfs:DataProperty; rdfs:domain Persona;
        rdfs:range xsd:string.
Edad a rdfs:DataProperty; rdfs:domain Persona;
        rdfs:range xsd:int.

Progenitor a owl:ObjectProperty; rdfs:domain Persona;
            rdfs:range Persona.

CardProg2 a owl:Restriction;
          owl:onProperty Progenitor;
          owl:cardinality 2.

Hijo_de a owl:ObjectProperty; owl:InverseOf Progenitor.
```

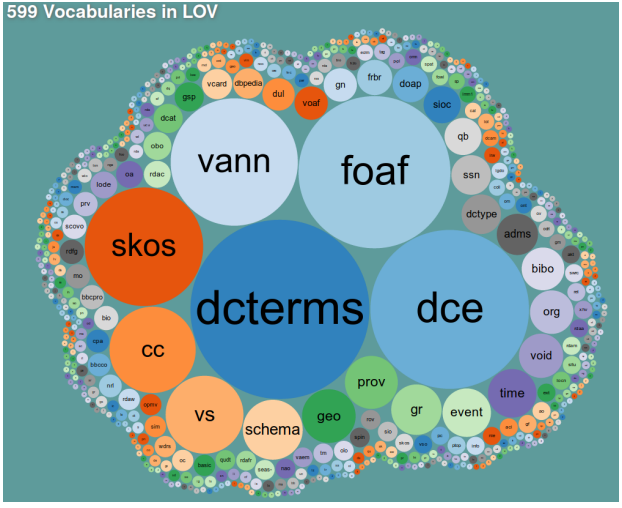
Las instancias se construyen a partir de clases y propiedades

```
juan a Hombre;
     rdfs:comment "Juan es el padre de Luisa";
     Edad 38;
     Nombre "Juan"
     Progenitor luisa.

luisa a Mujer;
     Edad 12;
     Nombre "Luisa"
     Hijo_de juan.
```

- La extensión del uso de la web semántica ha llevado a popularizar (y a veces a estandarizar) vocabularios
- Ejemplos:
  - SKOS (Simple Knowledge Organization System) (W3C)
  - FOAF (Friend of a Friend)
  - Dublin Core
  - Good Relations
  - Organization Ontology
  - DOAP (Description of a Project)
  - Basic Geo Vocabulary

- DAML (http://www.daml.org/ontologies/)
  - 282 Ontologías públicas escritas en DAML+OIL/OWL
  - academic department, Actors, address book, airport, Bibliography, Biology, Chemistry, Clothing, Weather, ...
- AberOWL ontology repository (http://aber-owl.net/)
- BioPortal (http://bioportal.bioontology.org/)
  - Más de 300 ontologías en biología
- Ontohub (http://ontohub.org/ontologies)
  - Cerca de 22.000 ontologías
- http://schema.org (Google, Microsoft, Yahoo, ...)

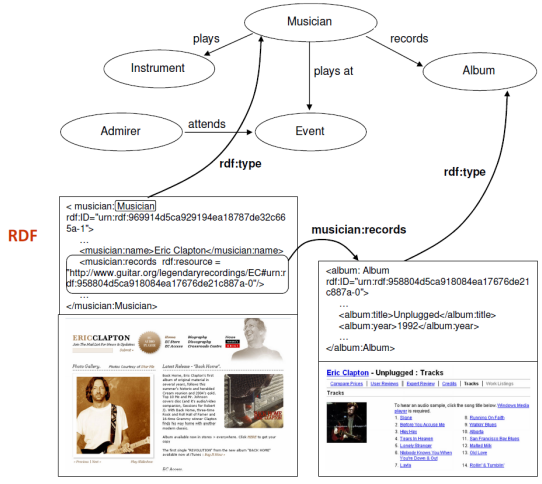


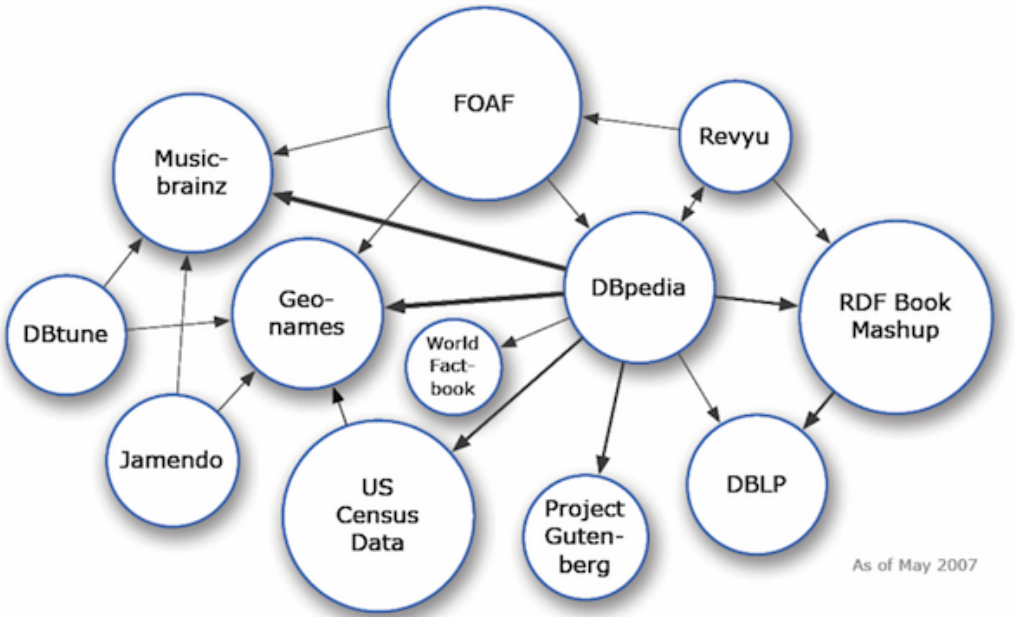
https://lov.linkeddata.es/dataset/lov/

- Friend of a Friend vocabulary
- Description of a Project (DOAP)
- European Skills, Competences, qualifications and Occupations (ESCO)
- Air Traffic Management Vocabulary
- Drammar: a comprehensive ontology of drama
- domOS Common Ontology (IoT)

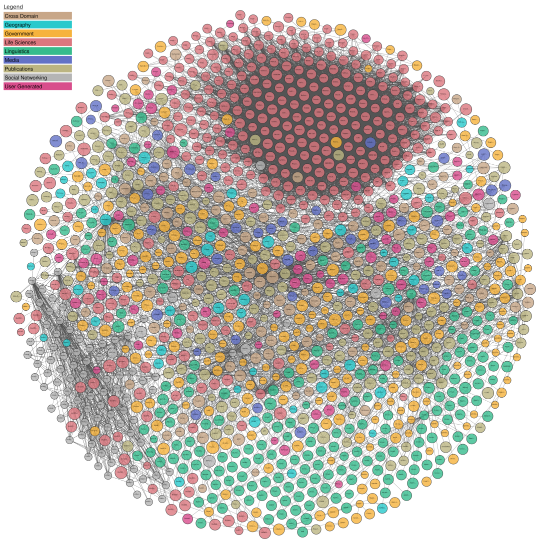
Linked Data

- El uso de ontologías/vocabularios comunes permite publicar datos de manera abierta
- Todo tipo de conocimiento puede ser descrito para su acceso
- Diferentes organizaciones pueden describir su información y enlazarla usando las mismas clases y atributos
- Disponer de esta información permite usarla en nuevas aplicaciones y de nuevas formas


Musician
plays
records
Instrument
plays at
Album
Admirer
attends
Event
rdf:type
rdf:type
RDF
< musician:Musician
rdf:ID="urn:rdf:969914d5ca929194ea18787de32c665a-1">
…
<musician:name>Eric Clapton</musician:name>
<musician:records rdf:resource =
"http://www.guitar.org/legendaryrecordings/EC#urn:rdf:958804d5ca918084ea17676de21c887a-0"/>
…
</musician:Musician>
musician:records
<album: Album
rdf:ID="urn:rdf:958804d5ca918084ea17676de21c887a-0">
…
<album:title>Unplugged</album:title>
<album:year>1992</album:year>
…
</album:Album>
ERIC CLAPTON
Eric Clapton - Unplugged : Tracks
Tracks


FOAF
Revyu
Music-
brainz
DBpedia
Geo-
names
DBtune
World
Fact-
book
RDF Book
Mashup
Jamendo
US
Census
Data
Project
Guten-
berg
DBLP
As of May 2007

Linked Data 2021 - http://lod-cloud.net/



The Linked Open Data Cloud from lod-cloud.net

- Los datos forman una gigantesca base de datos distribuida
- Cada elemento (recurso) esta descrito mediante un URI que le sirve de identificador
- Las propiedades/aserciones enlazan los recursos entre si
- El lenguaje de consulta **SPARQL** (similar a SQL) permite acceder a esta información
- Los puntos de conexión para lanzar las consultas se denominan **SPARQL endpoints**
  - eg.: DBPedia (`http://dbpedia.org/sparql`)

- Una **Triple Store** equivale a un RDBMS para linked data
- Almacena la información representada en grafos RDF
- Las operaciones que se pueden hacer sobre una TS son similares a las de una BD relacional
- El lenguaje de manipulación (**SPARQL**) está definido como un estándar por el W3C (version 1.1)
- Estándar W3C: http://www.w3.org/TR/sparql11-query/

- **SPARQL** es un lenguaje de consultas para grafos RDF
- Su sintaxis es similar a SQL
- No requiere que haya un esquema tabular para los datos
- Hace más sencilla la consulta a los datos a través sus relaciones
- Las consultas se resuelven mediante coincidencia de patrones (pattern matching) en el grafo

- SPARQL define cuatro tipos de consultas:
  - `SELECT`: permite obtener una lista de tripletas que coinciden con unas propiedades
  - `ASK`: permite saber si existe alguna instancia que cumpla unas propiedades
  - `DESCRIBE`: retorna algunas propiedades del URI que corresponde a la consulta
  - `CONSTRUCT`: permite construir un grafo RDF a partir de los resultados de la consulta

- SPARQL define cuatro tipos de operaciones de modificación:
  - `INSERT DATA`: Permite insertar nuevas tripletas
  - `INSERT`: Permite insertar nuevas tripletas o mover tripletas de un grafo RDF a otro
  - `DELETE DATA`: Permite borrar tripletas
  - `DELETE`: Permite borrar tripletas según un patrón
- Existen otras operaciones, pero el estándar no obliga a implementarlas

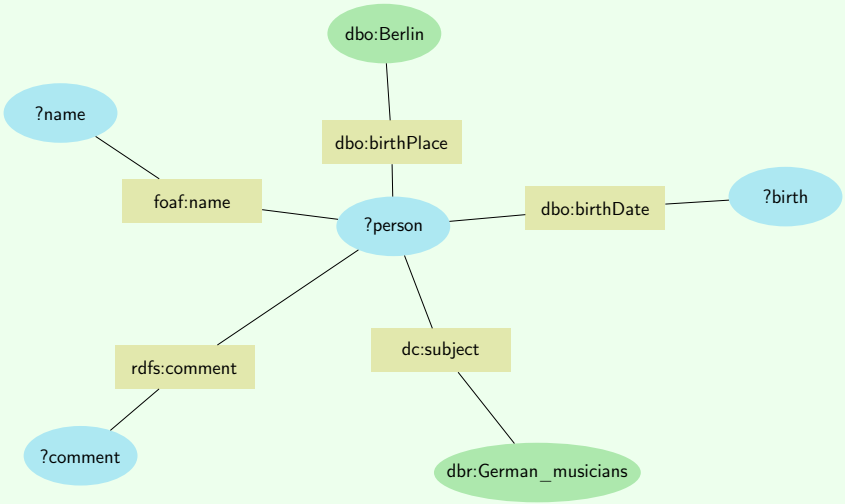
Añadimos una persona a un grafo con algunas propiedades

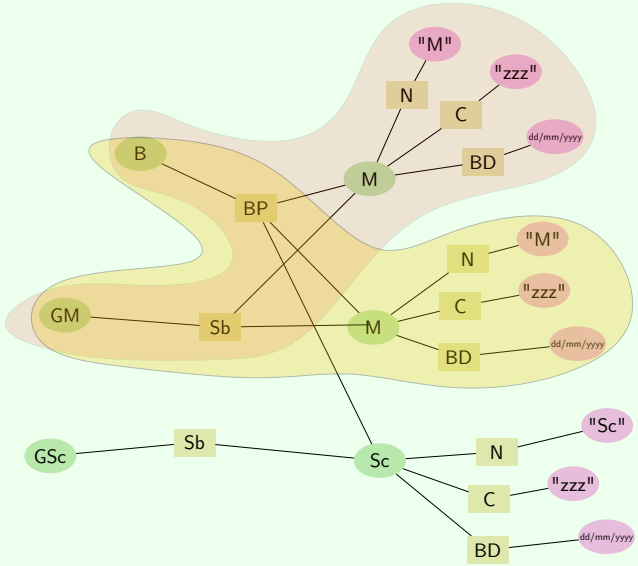
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ejemplo: <http://ejemplo.org/>
INSERT DATA
         {
          ejemplo:maria a foaf:person;
            foaf:name "Maria";
            foaf:age 27 .
}
```

- Una consulta SELECT esta compuesta por:
  - BASE: URI de la TS sobre la que hacer la consulta
  - PREFIX: Lista de prefijos a usar para simplificar la referencia a los espacios de nombre
  - SELECT: descripción del resultado de la consulta (qué variables) (podemos usar el modificador DISTINCT)
  - FROM: Grafo en el que hacer la consulta
  - WHERE: Patrón de la consulta
  - Modificadores: ORDER BY, GROUP BY, LIMIT, OFFSET, ...

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?birth ?city ?person ?name ?description
WHERE {
   ?person dct:subject dbc:German_musicians.
   ?person  dbo:birthDate ?birth.
   ?person dbo:birthPlace ?city.
    ?person foaf:name ?name.
    ?person rdfs:comment ?description .
}
```

http://factforge.net/sparql


dbo:Berlin
?name
dbo:birthPlace
foaf:name
?person
dbo:birthDate
?birth
rdfs:comment
dc:subject
?comment
dbr:German_musicians


"M"
N
"zzz"
C
dd/mm/yyyy
BD
M
B
BP
"M"
N
"zzz"
C
dd/mm/yyyy
BD
M
GM
Sb
"Sc"
N
"zzz"
C
dd/mm/yyyy
BD
Sc
GSc
Sb

- En la cláusula se pueden especificar otros elementos:
  - `optional`: Indicando que una parte de la consulta es opcional (queremos el resultado aunque no se cumpla)
  - `filter`: queremos aplicar un filtro a los valores de las variables de la consulta mediante una condición sobre su valor o una expresión regular
  - `union`: queremos que la consulta coincida con alguno de los patrones que indicamos

Personas con su nombre, correo (si es un .com) y fecha de nacimiento (si esta)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX myont: <http://my.ontology.org/my-ont#>
SELECT DISTINCT *
FROM <http://mi.grafo.org/personas.rdf>
WHERE {
        ?p myont:nombre ?n .
        ?p foaf:mbox ?mail.
        optional {
            ?p myont:fnacim ?fn.
        }
        filter (regexp(str(?mail),".com"))
}
```

100 Personas con su nombre, que tengan correo y/o teléfono y fecha de nacimiento posterior a 1/1/1990

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX myont: <http://my.ontology.org/my-ont#>
SELECT DISTINCT *
FROM <http://mi.grafo.org/personas.rdf>
WHERE {
   ?p myont:nombre ?n .
   { {?p foaf:mbox ?mail.}
     union
     {?p myont:telefono ?tf.}}
   ?p myont:fnacim ?fn.
   filter (xsd:date(str(?fn)) > "1990-1-1"^^xsd:date).}
LIMIT 100
```

Personas con su nombre y ordenadas por edad

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX myont: <http://my.ontology.org/my-ont#>
SELECT  ?n ?e
FROM <http://mi.grafo.org/personas.rdf>
WHERE {
       ?p myont:nombre ?n .
       ?p myont:edad ?e.
       filter (?e > 18).
}
ORDER BY ?e
```

- La sentencia `CONSTRUCT` permite añadir nuevas tripletas a partir de una consulta
- La cláusula `construct` permite indicar qué aserciones se han de construir a partir de las variables instanciadas en la consulta
- Esto permite definir reglas de deducción arbitrarias que permiten explicitar nueva información a partir de la consulta

Transformar datos de FOAF a mi ontología

```
PREFIX myont: <http://my.ontology.org/my-ont>
CONSTRUCT {
    ?p myont:nombre ?n.
    ?p myont:correo ?m.
}
WHERE {
    ?p foaf:name ?n.
    ?p foaf:mbox ?m.
}
```

Dos personas que viven en la misma dirección son vecinos

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX myont: <http://my.ontology.org/my-ont>
CONSTRUCT {
    ?p1 myont:vecino ?p2.
}
WHERE {
    ?p1 myont:direccion ?d.
    ?p2 miont:direccion ?d.
}
```