Tema 1

“Grafos de conocimiento”

Año académico 2024/25

Índice

- Introducción a la Web Semántica
- Resource Description Framework (RDF)
- Consulta de información en la Web Semántica
  - SPARQL
- RDF Schema

Bibliografía

Bibliografía:

- W3C Semantic Web Activity (http://www.w3.org/2001/sw/)
- A Semantic Web Primer. Grigoris Antoniou y Frank van Harmelen. The MIT Press. 2012 (3ª Ed.)
- Semantic Web: Concepts, Technologies and Applications. K. Breitman, M. A. Casanova, W. Truszkowski. Springer. 2007
- Foundations of Semantic Web Technologies. P. Hitzler, M. Krötzsch, S. Rudolph. CRC Press. 2009.

  (Libro disponible online a través de biblioteca)
- Inteligencia Artificial. Ejercicios Resueltos. H. Billhardt, A. Fernández, S. Ossowski. Areces. 2015

Web actual

- ¿Qué es la Web Actual?
  - Conjunto de documentos
  - Basada en HTML
  - Navegación basada en hipervinculos
  - Diseñada para que sea legible para los humanos

Web actual

- Problemas de la web actual
  - Falta de estructura semántica
  - Datos aislados
  - Dependencia de los motores de búsqueda

- Solución
  - Web Semántica

Web actual

Web Actual	Web Semántica
Documentos enlazados por hipervínculos.	Datos conectados con significado semántico.
Los datos están pensados para humanos.	Los datos están pensados para humanos y máquinas.
No hay contexto o significado asociado a los datos.	Los datos tienen un significado que las máquinas pueden entender.
Dificultad para integrar datos de múltiples fuentes.	Los datos pueden ser compartidos y reutilizados fácilmente entre diferentes fuentes.

Web Semántica

- La Web Semántica es una extensión de la Web actual
- Fue propuesta por **Tim Berners-Lee**, el creador de la World Wide Web.
- Diferencia clave
- ¿Por qué es importante la Web Semántica?
  - Interoperabilidad de datos

Web Semántica

- Interoperabilidad de los datos
  - Interoperabilidad de aplicaciones
  - Interoperabilidad de diferentes sistemas y organizaciones

- Ejemplos de iniciativas:
  - OBO (Open Biological and Biomedical Ontology)
  - Linked Universities
  - GoodRelations Ontology

La Web Semántica

- Motivación (histórica): el problema con la Web

  Millones de documentos y datos disponibles online.

  – Recuperación de los documentos adecuados
  – Extracción de los datos relevantes (de dichos documentos)
  – Combinación de información de diferentes fuentes

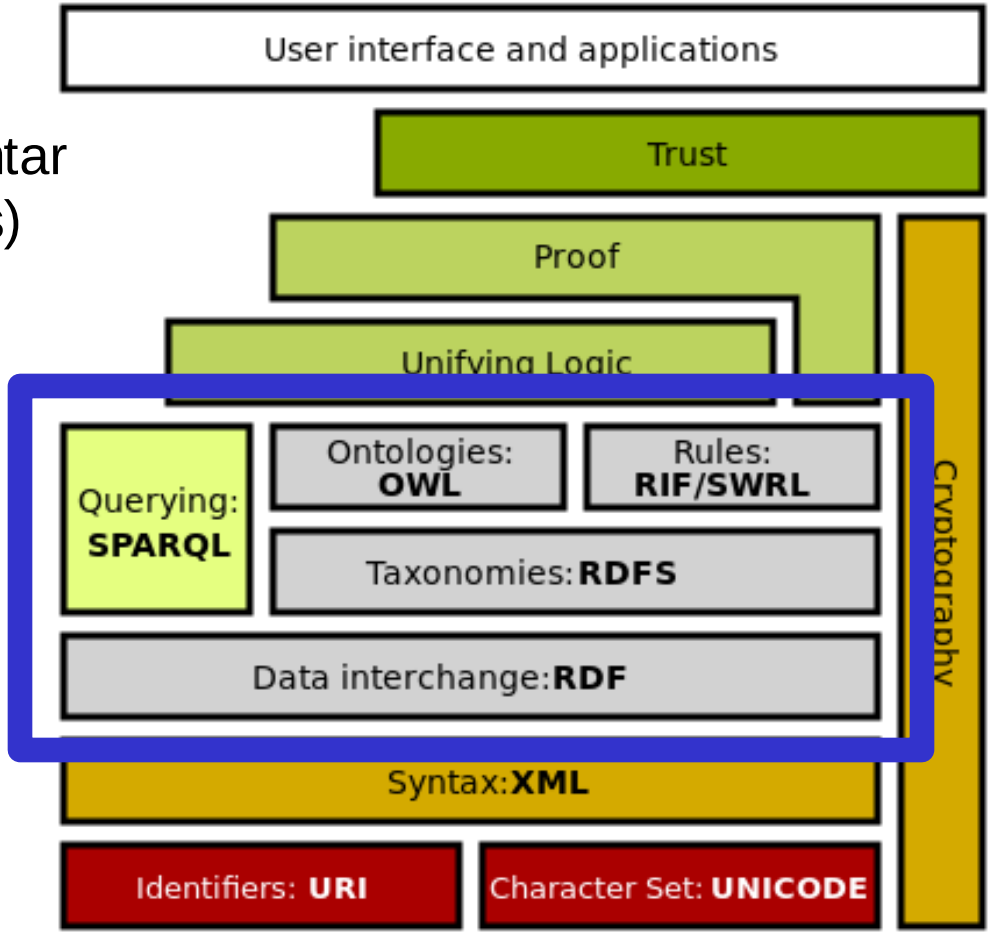
- Propuesta
  – Publicar datos procesables y entendibles por los ordenadores
  – Consultar información en términos entendibles por los ordenadores

La Web Semántica

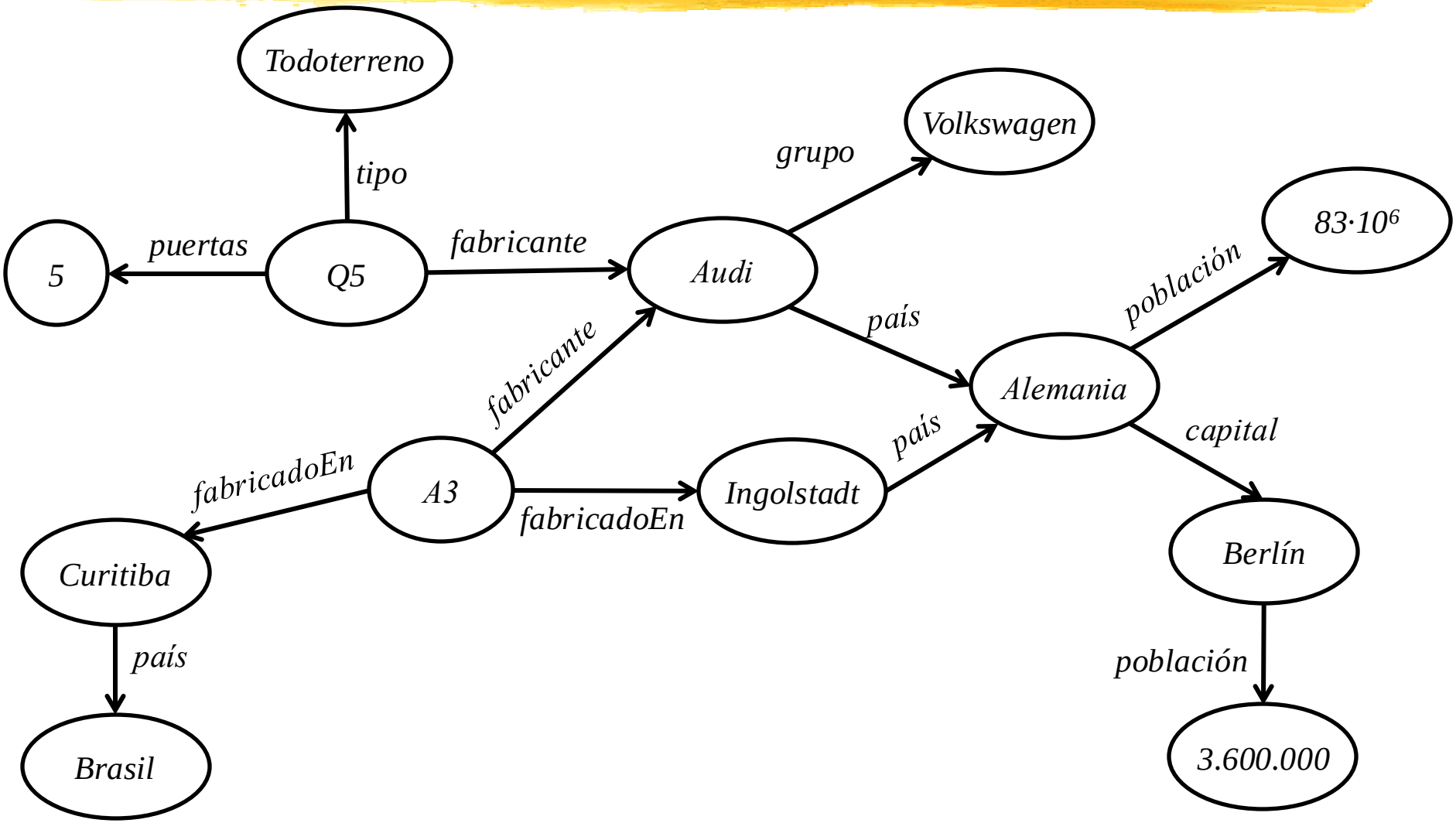
- Objetivo
  - Tecnologías para facilitar la compartición y reutilización de datos entre agentes, tanto de forma automática como manual
- Enfoque
  - Extender los principios de la Web (de documentos) a los datos. Relacionar unos datos con otros (igual que las webs se relacionan entre ellas)
  - ¿Cómo?
    - usando vocabularios/ontologías estandarizadas
      - Lenguajes para construir ontologías
      - Ontologías concretas

La Web Semántica

- Proporciona
  - Lenguajes para representar conocimiento (ontologías)
    - RDF, RDF(S), OWL
  - Lenguaje de consulta
    - SPARQL
  - Herramientas



Grafos de Conocimiento (Redes Semánticas)



Grafos de Conocimiento

- Modelo flexible de organizar la información
  - Más que, por ejemplo, modelo relacional (rigidez de tablas)
  - Fácil de extender y mejorar la información
- Puede mediar entre diferentes modelos
- Usados internamente por grandes compañías
  - Google Knowledge Graph
  - Microsoft: Bing knowledge graph
  - Facebook
  - eBay
  - IBM (Watson)
  - …
- Hay muchos abiertos

Índice

- Introducción a la Web Semántica
- Resource Description Framework (RDF)
- Consulta de información en la Web Semántica
  - SPARQL
- RDF Schema (RDFS)

RDF

- Resource Description Framework
- Recomendación del W3C
  - http://www.w3.org/RDF/
  - http://www.w3.org/TR/rdf11-primer/
- Expresar información sobre recursos en la Web
  - documentos, personas, objetos físicos y abstractos, etc.
  - se identifican con IRIs ("International Resource Identifier")
    - http://www.urjc.es/
    - http://dbpedia.org/resource/King_Juan_Carlos_University
    - http://sws.geonames.org/2510769/
    - http://example.org/bob#me
- Modelo de datos equivalente a una red semántica

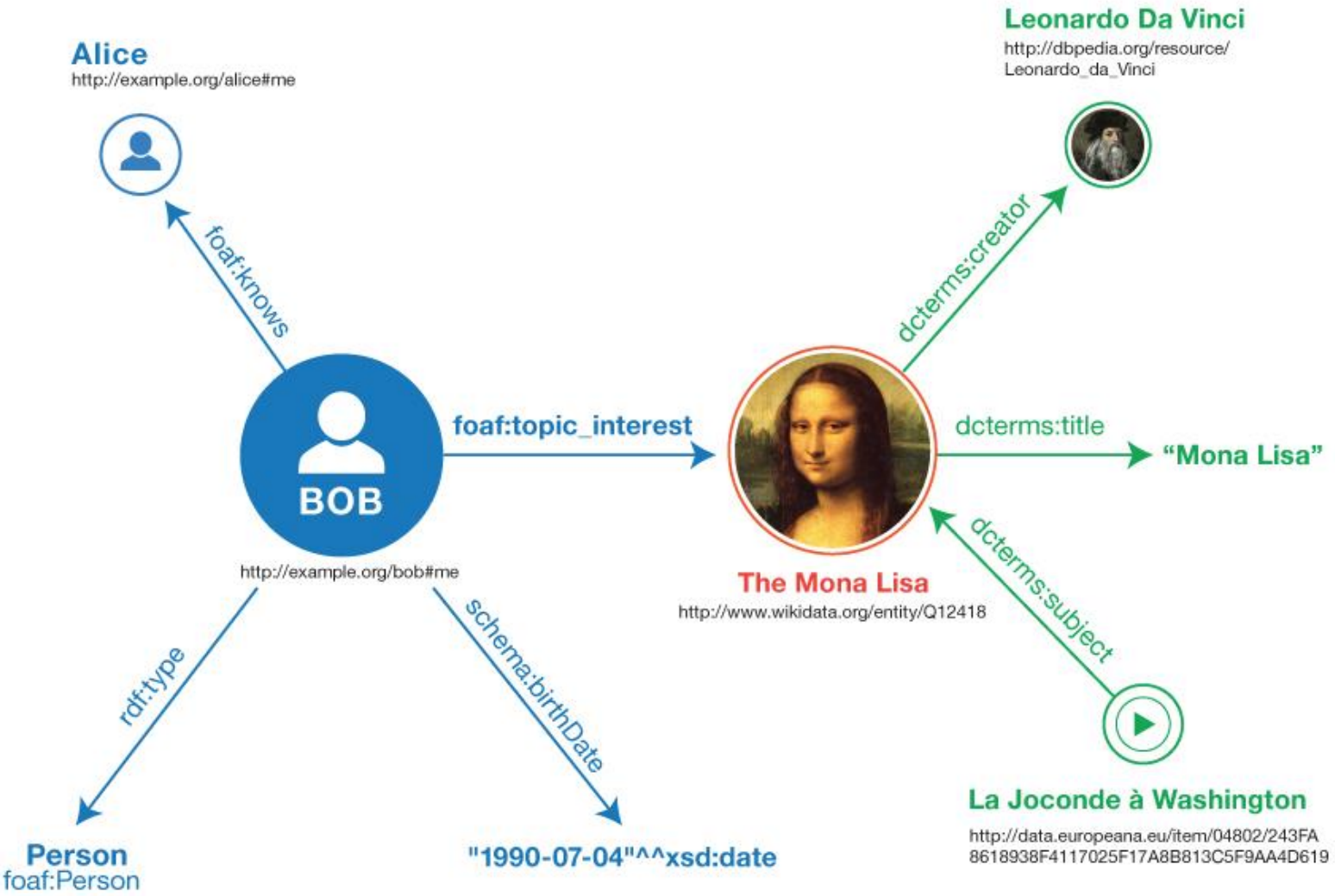
RDF

- RDF como modelo de datos equivalente a una red semántica

- RDF organiza la información en una estructura de red, lo que se conoce como un grafo de triples.

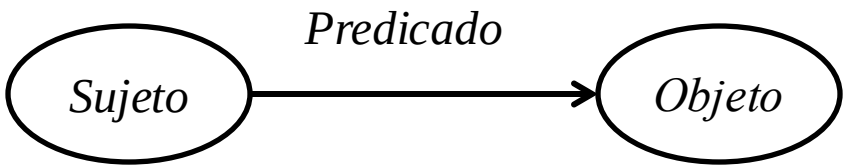
- Los triples tienen tres partes:
  - Sujeto
  - Predicado
  - Objeto
- Ejemplo: "La Universidad Rey Juan Carlos (sujeto) está en (predicado) Madrid (objeto)".

RDF: Ejemplo

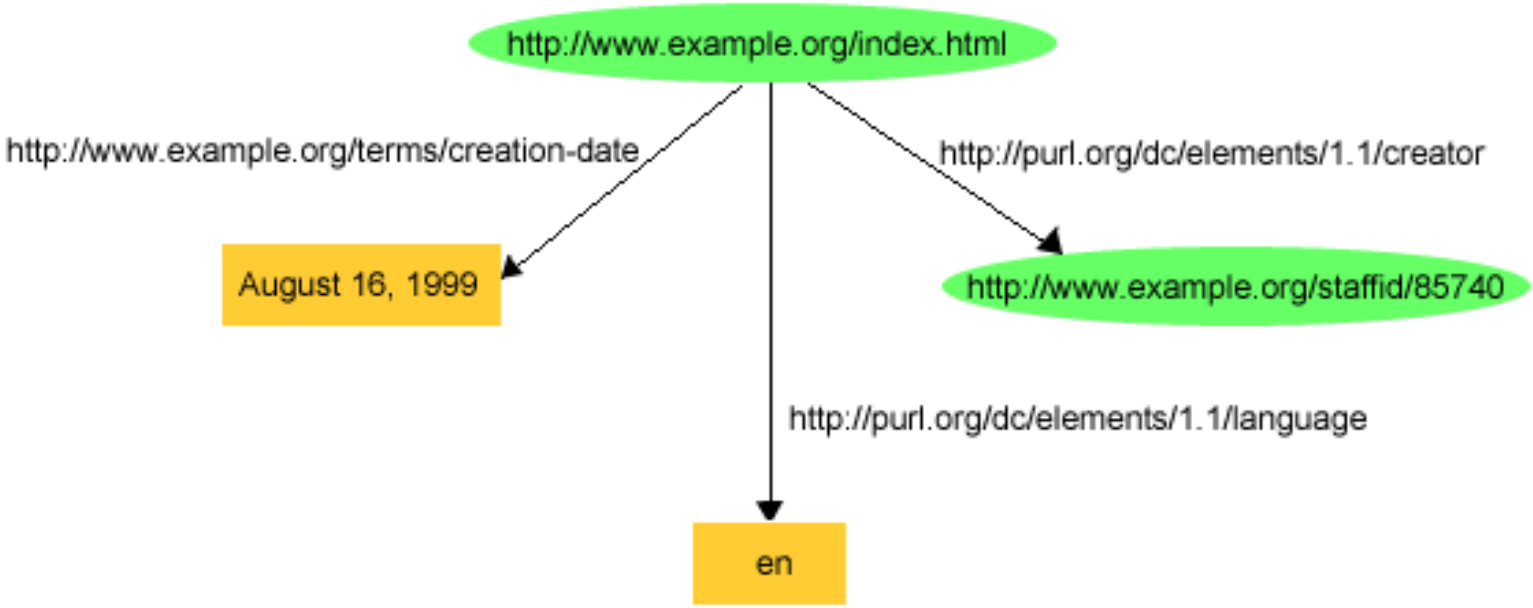

Alice
http://example.org/alice#me
Leonardo Da Vinci
http://dbpedia.org/resource/
Leonardo_da_Vinci
foaf:knows
dcterms:creator
BOB
http://example.org/bob#me
foaf:topic_interest
dcterms:title
"Mona Lisa"
The Mona Lisa
http://www.wikidata.org/entity/Q12418
dcterms:subject
rdf:type
schema:birthDate
La Joconde à Washington
http://data.europeana.eu/item/04802/243FA
8618938F4117025F17A8B813C5F9AA4D619
Person
foaf:Person
"1990-07-04"^^xsd:date

RDF

- Un grafo RDF es un conjunto de Sentencias (arcos)
- Sentencia = tripla (Sujeto, Predicado, Objeto)
  - Sujeto: recurso (IRI) o “blank node”
  - Predicado/Propiedad: relación binaria (IRI)
  - Objeto: IRI, literal o “blank node”



RDF: Ejemplo


http://www.example.org/index.html
http://www.example.org/terms/creation-date
http://purl.org/dc/elements/1.1/creator
August 16, 1999
http://www.example.org/staffid/85740
http://purl.org/dc/elements/1.1/language
en

RDF: Serialización

- Varios formatos de representación (serialización)
  - N-Triples
  - **Turtle**
  - TriG
  - N-Quads
  - JSON-LD
  - RDFa (para incluir en HTML)
  - RDF/XML

- ¿Qué formato elegirías?

RDF: Serialización. Turtle

- Turtle es un formato para serializar datos RDF (Resource Description Framework).
- Es un formato legible para humanos y más conciso que N-Triples o RDF/XML.
- Características:
  - Utiliza prefijos para simplificar las IRIs largas.
  - Permite expresar tripletas de forma compacta.
  - Soporta estructuras complejas mediante listas y colecciones.
  - Es ampliamente usado debido a su simplicidad y legibilidad.

RDF: Serialización. Turtle

<sujeto> <predicado> <objeto> .

<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/creator> <http://www.example.org/staffid/85740> .
<http://www.example.org/index.html> <http://www.example.org/terms/creation-date> "August 16, 1999" .
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/language> "en" .

O más compacto:

```
@prefix ex: <http://www.example.org/>.
@prefix exstaff: <http://www.example.org/staffid/>.
@prefix exterms: <http://www.example.org/terms/>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
ex:index.html dc:creator exstaff:85740 .
ex:index.html exterms:creation-date "August 16, 1999" .
ex:index.html dc:language "en" .
```

O

```
ex:index.html dc:creator exstaff:85740 ;
              exterms:creation-date "August 16, 1999" ;
              dc:language "en" .
```

Abreviaciones

- @prefix
- ; (compartir sujeto)
- , (compartir sujeto-pred)
- a (rdf:type)
- [] (blank nodes)
- () (listas)

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

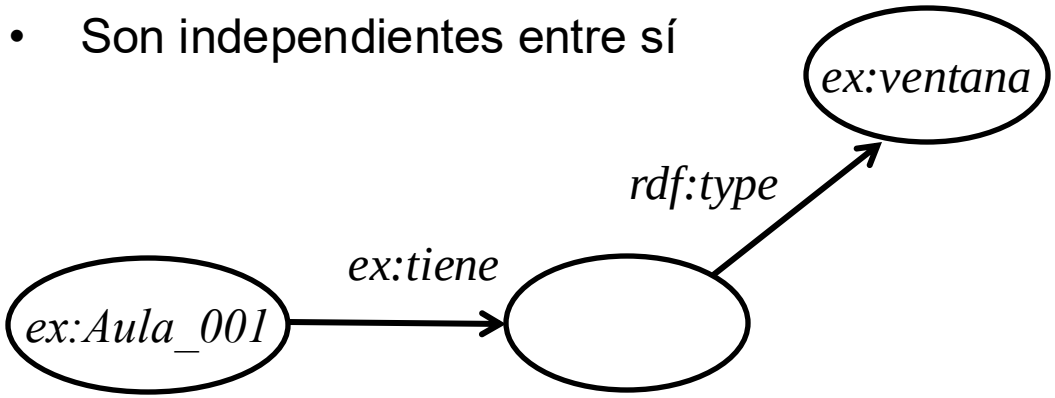
RDF: Sintaxis XML. Ejemplo

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:exterms="http://www.example.org/terms/">
  <rdf:Description rdf:about="http://www.example.org/index.html">
      <exterms:creation-date>August 16, 1999</exterms:creation-date>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.example.org/index.html">
      <dc:language>en</dc:language>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.example.org/index.html">
      <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
  </rdf:Description>
</rdf:RDF>
```

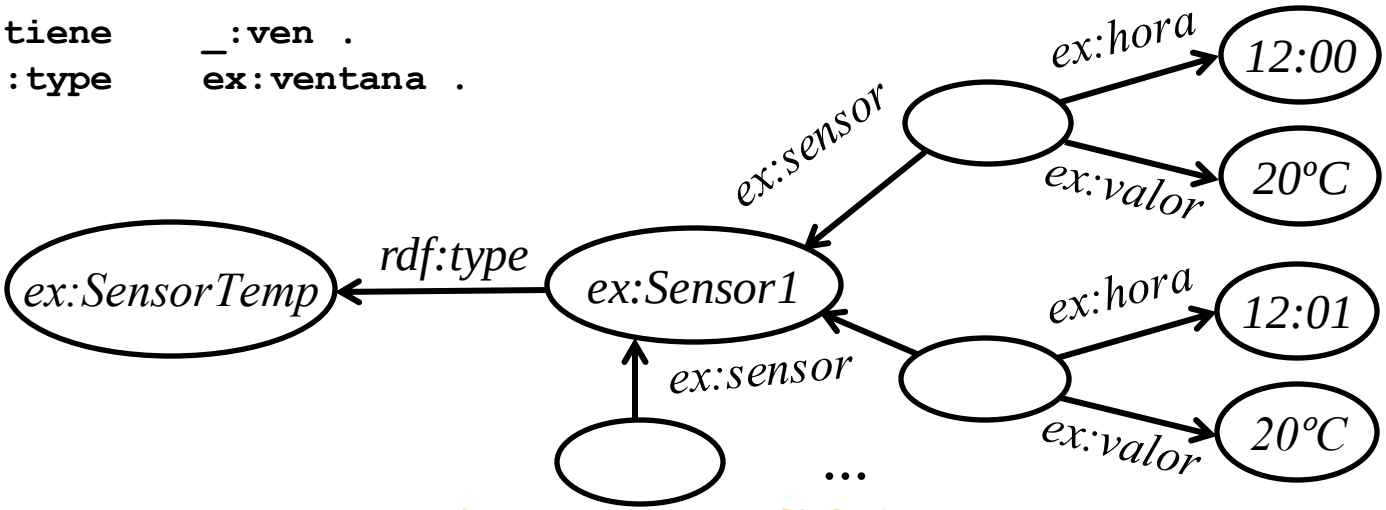
```
<rdf:Description rdf:about="http://www.example.org/index.html">
      <exterms:creation-date>August 16, 1999</exterms:creation-date>
      <dc:language>en</dc:language>
      <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
 </rdf:Description>
```

RDF: Nodos Anónimos (Blank Nodes)

- Nodos sin IRI (no es necesaria)
- Son independientes entre sí




```
ex:Aula_001 ex:tiene    _:ven .
_:ven       rdf:type    ex:ventana .
```



RDF: Literales

- Tipos de datos
  - Se puede asociar un tipo a los literales: “valor”^^<tipo>
  - Se recomienda usar XML Schema datatypes
    (xsd=http://www.w3.org/2001/XMLSchema#):
    - xsd:string, xsd:integer, xsd:date,…
  - Se pueden etiquetar strings indicando el idioma

- Ejemplo

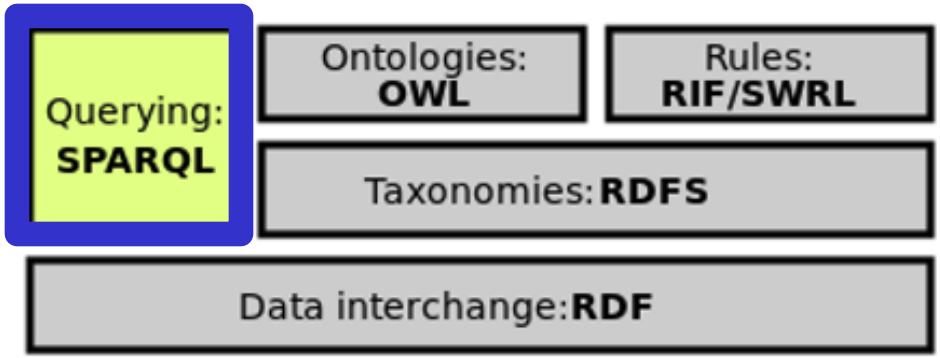
```
ex:index.html exterms:creation-date "1999-08-16"^^xsd:date .
exstaff:85740 exterms:postalCode "01730"^^xsd:integer .
exstaff:85740 exterms:birthPlaceName "Boston"@en .
```

Índice

- Introducción a la Web Semántica
- Resource Description Framework (RDF)
- Consulta de información en la Web Semántica
  - SPARQL
- RDF Schema

SPARQL

- Query Language for RDF
- Recomendación del W3C
  - http://www.w3.org/TR/sparql11-query/
- Lenguaje de consulta de contenidos RDF
- Sintaxis estilo SQL



SPARQL

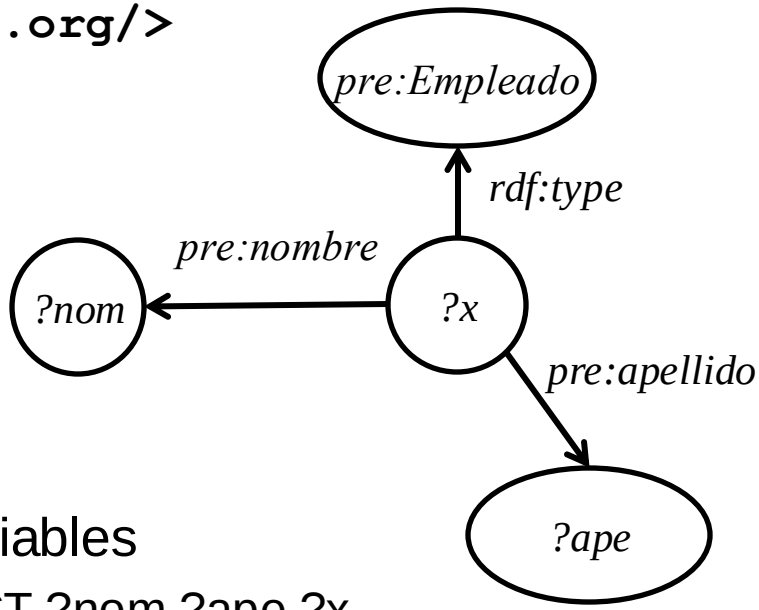
Estructura de una consulta:

```
[
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  ...
]
SELECT ...
[FROM ...]
WHERE
{
        ...
}
[Modificadores]
```

SPARQL

- Patrones básicos

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX pre: <http://www.example.org/>
SELECT ?nom ?ape
WHERE {
   ?x rdf:type pre:Empleado.
  ?x pre:nombre   ?nom.
   ?x pre:apellido ?ape.
   }
```



- Observaciones
  - SELECT * selecciona todas las variables
    - En el ejemplo: SELECT * = SELECT ?nom ?ape ?x
  - También:

```
{ ?x pre:nombre   ?nom ;
     pre:apellido ?ape ;
     a pre:Empleado.
}
```

SPARQL

- Literales
  - Etiquetas de idioma
    - “Madrid”, “Madrid”@es y “Madrid”@en son valores distintos
  - Tipos numéricos
    - “123"^^<http://www.w3.org/2001/XMLSchema#integer> es lo mismo que 123 (y que "123"^^xsd:integer) con:

      PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
  - Tipos arbitrarios
    - "abc"^^ex:TipoT

```
SELECT ?s
WHERE
{
        ?s ?p "abc"^^ex:TipoT.
}
```

SPARQL

- Restricciones (FILTER)
  - Numéricas

```
SELECT ?nom ?ape
WHERE
{ ?x pre:nombre    ?nom.
  ?x pre:apellido ?ape.
  ?x pre:nacido_en ?pais.
  ?pais pre:poblacion ?hab.
  FILTER (?hab >= 200000).}
```

Algunos Operadores
Lógicos: !, &&, ||
*Aritméticos: +, -, *, /*
Comparación: =, !=, >, <,>=,<= ...

  - Strings
    - `FILTER (regex(?nom, "Alberto", "i"))`
    - `FILTER (regex(str(?nom), "^Al"))`
    - *regex* como en XQuery 1.0 y XPath 2.0

da igual mayúsculas/ minúsculas

  - Otros
    - `FILTER ( ?fecha > "1980-05-1"^^xsd:date && ?fecha < "1980-06-30"^^xsd:date )`

SPARQL

- Patrones opcionales (OPTIONAL)

```
SELECT ?nom ?ape ?nac ?email
WHERE
{ ?x pre:nombre    ?nom.
  ?x pre:apellido ?ape.
  ?x rdf:type pre:Empleado.
  OPTIONAL {?x pre:añoNac  ?nac.
            ?x pre:email ?email.}
}
```

- Ojo! es distinto a:

```
SELECT ?nom ?ape ?nac ?email
WHERE
{ ?x pre:nombre    ?nom.
  ?x pre:apellido ?ape.
  ?x rdf:type pre:Empleado.
  OPTIONAL {?x pre:añoNac  ?nac.}
  OPTIONAL {?x pre:email ?email.}
}
```

SPARQL

- Patrones alternativos (UNION)

```
SELECT ?nom ?a
WHERE
{
   {
      ?x pre:nombre ?nom.
      ?x rdf:type pre:Empleado.
   }
   UNION
   {
      ?x pre:name ?nom.
      ?x rdf:type pre:Employee.
   }

   ?x pre:añoNac ?a.
}
```

SPARQL

- Negación
  - Comprobando ausencia

```
SELECT ?nom ?ape
WHERE
{ ?x pre:nombre   ?nom.
  ?x pre:apellido ?ape.
  FILTER NOT EXISTS {?x rdf:type pre:Empleado}
}
```

  - Eliminando posibles soluciones

```
SELECT ?nom ?ape
WHERE
{ ?x pre:nombre   ?nom.
  ?x pre:apellido ?ape.
  MINUS {?x rdf:type pre:Empleado}
}
```

SPARQL

- Agregados
  - COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT, SAMPLE

```
SELECT ?equi (SUM(?sueldo) AS ?totalSueldos)
WHERE {
  ?x rdf:type pre:Futbolista .
  ?x pre:equipo ?equi.
  ?x pre:salario ?sueldo .
}
GROUP BY ?equi
HAVING (SUM(?sueldo) > 20)
```

Filtra soluciones

SPARQL

- Agregados
  - **COUNT**: Devuelve el número total de elementos en un grupo.
    - Ejemplo: `COUNT(?x)` cuenta el número de futbolistas en cada equipo.
  - **AVG**: Calcula el promedio de un conjunto de valores.
    - Ejemplo: AVG(?sueldo) calcula el salario promedio de los futbolistas por equipo.
  - **MIN** y **MAX**: Devuelven el valor mínimo o máximo en un grupo.
    - Ejemplo: MIN(?sueldo) devuelve el salario más bajo entre los futbolistas de un equipo.
    - Ejemplo: MAX(?sueldo) devuelve el salario más alto.
  - **GROUP_CONCAT**: Concatenar los valores de un grupo en un solo resultado.
    - Ejemplo: GROUP_CONCAT(?nombre) podría concatenar los nombres de todos los futbolist de un equipo en una cadena de texto.
  - **SAMPLE**: Devuelve un valor aleatorio de entre los resultados de un grupo.
    - Ejemplo: SAMPLE(?nombre) devolvería un futbolista aleatorio por equipo.

SPARQL

- Otras
  - Expresiones agregadas en SELECT
  - Subqueries
  - Consultas federadas
  - Modificar los grafos del dataset (UPDATE)
  - …

SPARQL

- Modificadores de resultados
  - ORDER BY

```
SELECT ?nom ?ape
WHERE
{ ?x pre:nombre    ?nom.
  ?x pre:apellido ?ape.
}
ORDER BY ?ape DESC(?nom)
```

  - DISTINCT: evita soluciones duplicadas

```
SELECT DISTINCT ?nom ?ape
```

  - OFFSET / LIMIT

```
SELECT  ?nom ?ape
WHERE
{ ?x pre:nombre ?nom.
  ?x pre:apellido ?ape.}
ORDER BY ?ape
LIMIT 5
OFFSET 3
```

SPARQL

- RDF Datasets
  - Un *RDF dataset* representa una colección de grafos RDF
    - Grafo por defecto (*default graph*)
    - Cero o más grafos nombrados (*named graphs*)

```
PREFIX ex: <http://example.org/>
PREFIX pre: <http://www.example.org/>
SELECT ?nom ?ape
FROM <http://example.org/datos1>
FROM NAMED <http://example.org/datos2>
FROM NAMED <http://example.org/datos3>
WHERE
{
GRAPH ex:datos2 {
?x pre:nombre   ?nom.
?x pre:apellido ?ape.
}
}
```

default graph

SPARQL

- Consultas Federadas

```
SELECT ?nom ?ape
WHERE {
   ?x rdf:type pre:Empleado.
   SERVICE <http://datos.personas.com/sparql> {
     ?x pre:nombre   ?nom.
     ?x pre:apellido ?ape.
   }
}
```

SPARQL

- Formas de consulta
  - SELECT
  - ASK
    - devuelve si existe o no alguna solución
  - DESCRIBE
    - Grafo RDF con datos de los objetos indicados
    - depende del servidor

```
DESCRIBE ?pais
WHERE
{ ?x pre:nombre    ?nom.
  ?x pre:nacido_en ?pais.
  ?pais pre:poblacion ?hab.
  FILTER (?hab >= 200000).
}
```

  - CONSTRUCT
    - genera un grafo RDF

SPARQL

- CONSTRUCT

```
CONSTRUCT {?x pre:tieneTio ?her.
           ?her pre:tieneSobrino ?x.}
WHERE
{ ?x pre:tienePadre ?padre.
  ?padre pre:tieneHermano ?her.
}

CONSTRUCT {?x o2:hasParent ?padre.
           ?padre o2:hasBrother ?her.}
WHERE
{ ?x o1:tienePadre ?padre.
  ?padre o1:tieneHermano ?her.
}
```

SPARQL

Property Paths

- **Property Paths** es un elemento del lenguaje de consulta SPARQL que permite al usuario consultar rutas complejas entre nodos en lugar de limitarse cada vez a los vecinos colindantes.

- Los **Property Paths** amplían las capacidades de SPARQL para emparejar patrones de grafos de longitud indeterminada y ofrecer múltiples alternativas

- W3C. SPARQL 1.1Property Paths -> https://www.w3.org/TR/sparql11-property-paths/

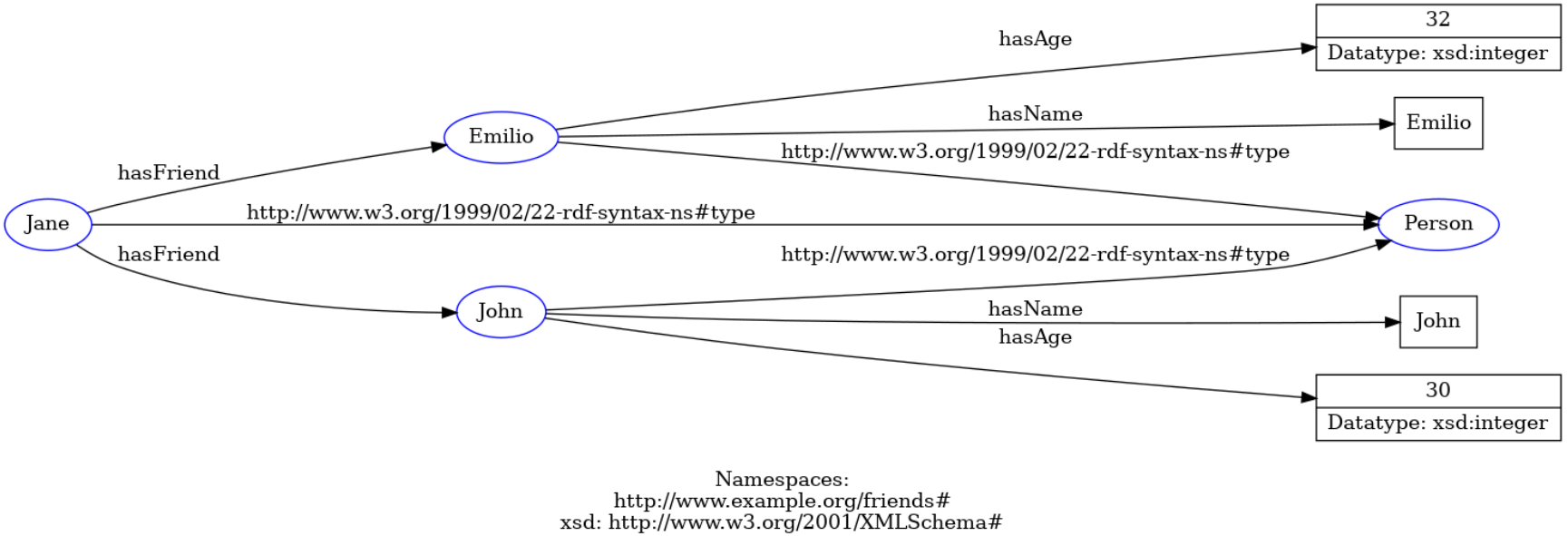
SPARQL

Property Paths

- Principales operadores:

property1/property2	Sequenceproperty	Matches on forward property--property1 followed by property2.
property1|property2	AlternativeProperty	Matches on either property1 or property2.
property1*	ZeroOrMoreProperty	Connects the subject and object of the propertyby zero or more matches of property1, i.e., property1 repeated zero or more times.
property1+	OneOrMoreProperty	Connects the subject and object of the propertyby one or more matches of property1, i.e., property1 repeated one or more times.
property1?	ZeroOrOneProperty	Connects the subject and object of the propertyby zero or one matches of property1, i.e., property1 is optional.

SPARQL



Namespaces:
http://www.example.org/friends#
xsd: http://www.w3.org/2001/XMLSchema#

Property Paths

- Si queremos obtener las edades de todos los amigos de Jane sería:

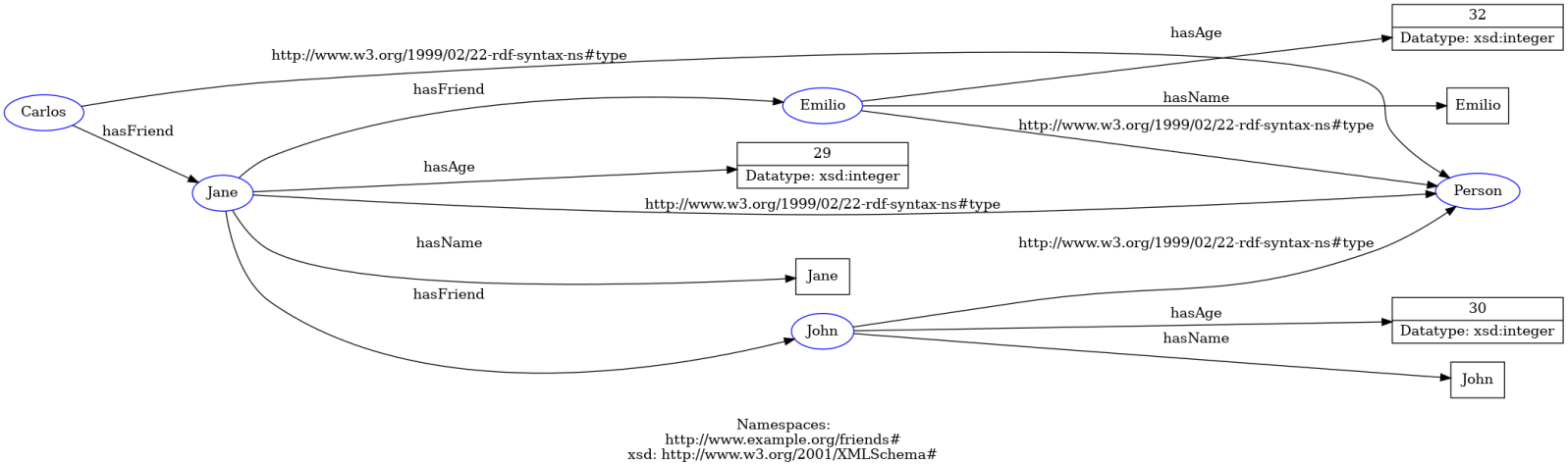
Sin usar property paths:

```
SELECT ?edad WHERE {
   ex:Jane ex:hasFriend ?amigo .

?amigo ex:hasAge ?edad .}
```

Usando property paths:

```
SELECT ?edad WHERE {
    ex:Jane ex:hasFriend/ex:hasAge ?edad .
}
```

SPARQL



Property Paths

Si queremos obtener todos los amigos de Carlos:

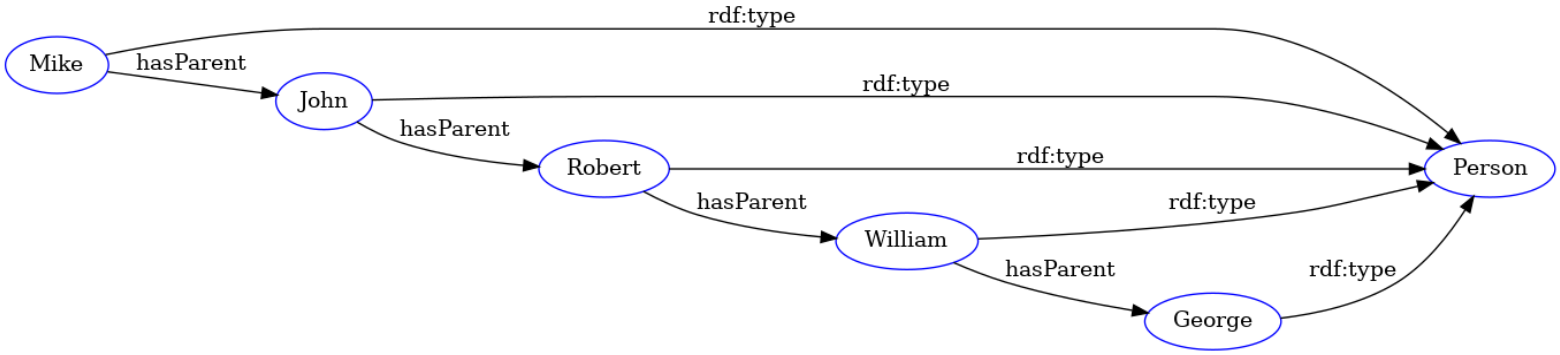
Usando property paths

```
SELECT ?nombre WHERE {
   ex:Carlos ex:hasFriend* ?amigo .
   ?amigo ex:hasName ?nombre .
}
```

Usando property paths:

```
SELECT ?nombre WHERE{
   ex:Carlos ex:hasFriend+ ?amigo .
   ?amigo ex:hasName ?nombre .
}
```

SPARQL



Namespaces:
http://www.example.org/family#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

Property Paths

Si queremos obtener todos los parientes de Mike sería:

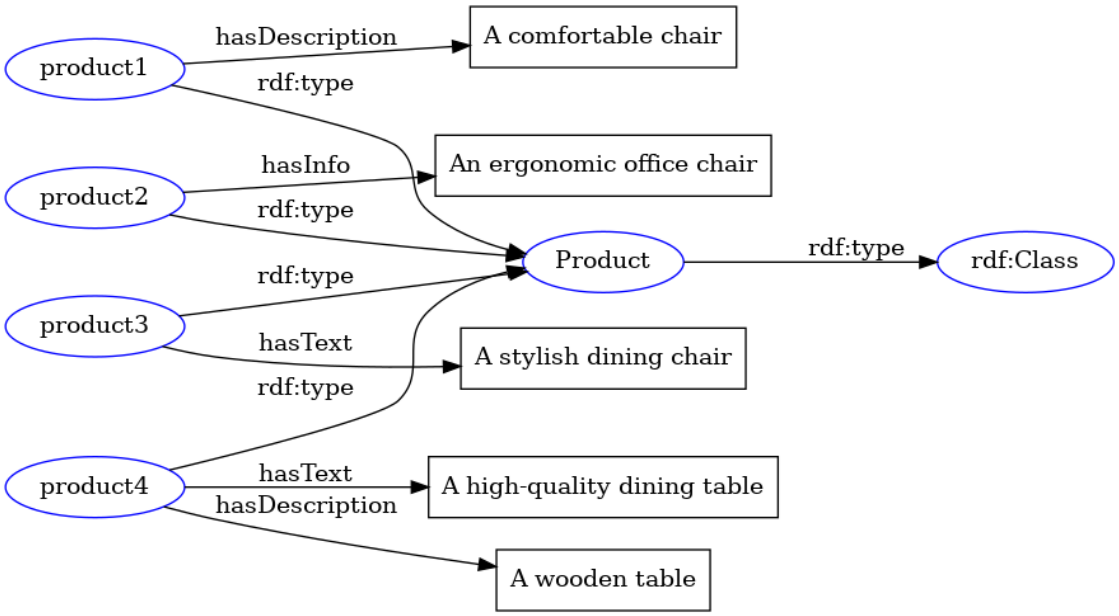
```
SELECT * WHERE {
   :Mike :hasParent+ ?ancestor .
}
```

SPARQL

El código turtle sería:

```
@prefix : <http://www.example.org/family#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
      .
# Definición de personas
:Mike a :Person ;
    :hasParent :John .
:John a :Person ;
    :hasParent :Robert .
:Robert a :Person ;
    :hasParent :William .
:William a :Person ;
    :hasParent :George .
:George a :Person .
```

SPARQL



Namespaces:
http://example.org/products#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

Property Paths

Obtener las descripciones de todos los productos, incluso si nuestros datos proceden de diferentes fuentes que utilizan diferentes etiquetas de predicado.

```
SELECT * WHERE {
   ?product a :Product;
   ?product :hasDescription|:hasInfo|:hasText ?description
}
```

SPARQL

El código turtle sería:

```
@prefix : <http://example.org/products#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
# Definición de clases y productos
:Product a rdf:Class .
# Instancias de productos
:product1 a :Product ;
    :hasDescription "A comfortable chair" .
:product2 a :Product ;
    :hasInfo "An ergonomic office chair" .
:product3 a :Product ;
    :hasText "A stylish dining chair" .
:product4 a :Product ;
    :hasDescription "A wooden table" ;
    :hasText "A high-quality dining table" .
```

SPARQL

- Otras
  - Subqueries
  - Asignaciones
    - BIND (?precio*(1+?iva) AS ?total)
  - Modificar los grafos del dataset (UPDATE)
    - http://www.w3.org/TR/sparql11-update/
  - …

SPARQL Endpoints

- Aceptan consultas SPARQL y devuelven los resultados por HTTP
- Formatos de resultados
  - XML, JSON, RDF (RDF/XML, N-Triples, Turtle, etc.), HTML, CSV, TSV,…
- Tipos
  - Genéricos
    - http://www.openlinksw.com/sparql
    - http://sparql.org/sparql.html
    - http://librdf.org/query/
    - …
  - Específicos

Índice

- Introducción a la Web Semántica
- Resource Description Framework (RDF)
- Consulta de información en la Web Semántica
  - SPARQL
- RDF Schema (RDFS)

RDF Schema (RDFS)

- En RDF hablamos de *objetos individuales* (recursos)
- Nos gustaría razonar sobre *clases* que definen tipos de objetos
  - Por ejemplo, para evitar sentencias como (válidas en RDF):
    - *BD es impartida por BD* (restricción de rango)

      `ex:BD ex:impartidaPor ex:BD.`
    - *Casa es impartida por Juan* (restricción de domino)

      `ex:Casa ex:impartidaPor ex:Juan.`
- Solución
  - Clases, relaciones, restricciones de dominio y rango, …
  - Ejemplo:
    - *Las asignaturas deben ser impartidas por miembros del personal docente*

RDF Schema (RDFS)

- Recomendación del W3C
  - http://www.w3.org/TR/rdf-schema/
- RDFS extiende RDF con nuevas primitivas (IRIs) con semántica definida
- Define a un lenguaje básico para describir ontologías
  - Clases e Instancias
  - Propiedades
    - Restricciones de dominio y rango
  - Jerarquías de clases y propiedades
    - Define la semántica de “subclase de” y “subpropiedad de”

RDF Schema (RDFS)

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

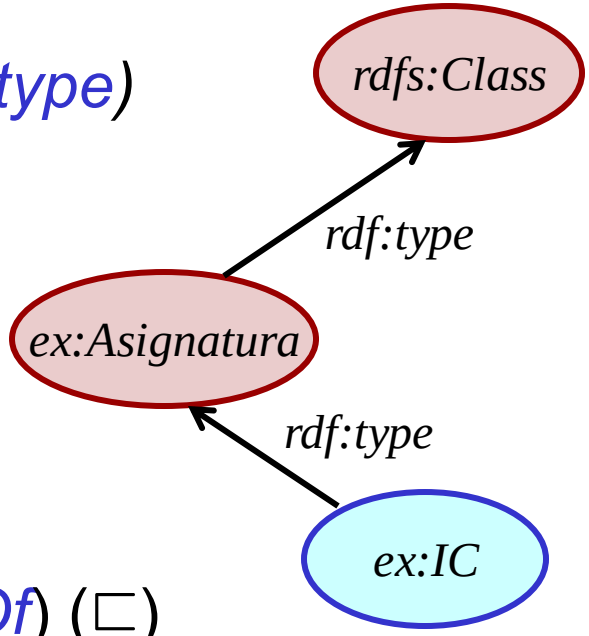
- Clases (*rdfs:Class*) *e* Instancias (*rdf:type*)

  Definición de Clases

  ```
  ex:Asignatura rdf:type rdfs:Class.
  ```

  Instancias

  ```
  ex:IC rdf:type ex:Asignatura.
  ```



- Jerarquías de clases (*rdfs:subClassOf*) (⊑)

  ```
  ex:Catedratico rdfs:subClassOf ex:PersonalDocente.
  ```

RDF Schema (RDFS)

- Propiedades (*rdf:Property*)

```
ex:impartidaPor rdf:type rdf:Property .
```

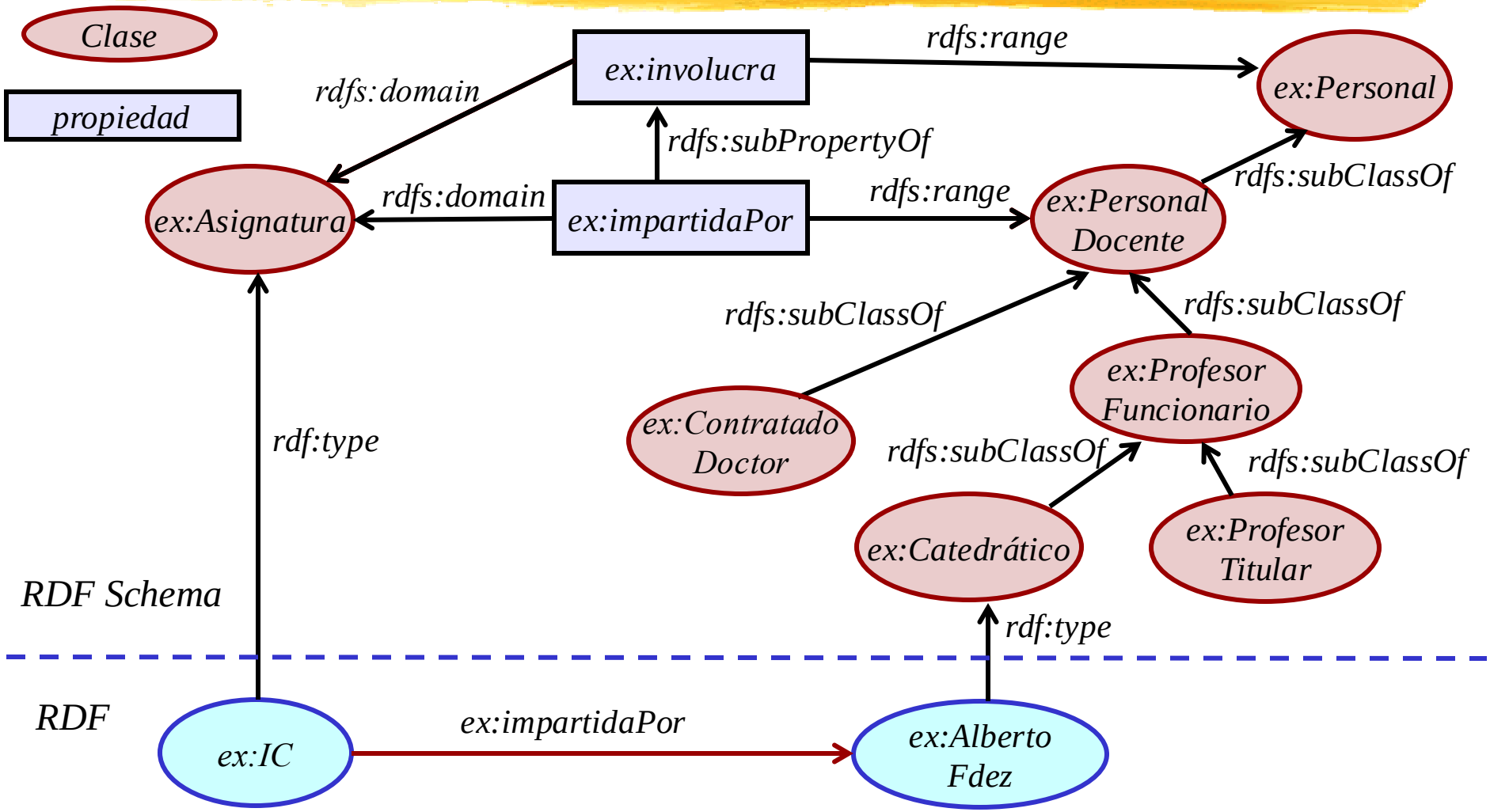
- Restricciones propiedades (*rdfs:domain* ($\exists r.\top \sqsubseteq C$), *rdfs:range* ($\top \sqsubseteq \forall r.C$))

```
ex:impartidaPor rdfs:domain ex:Asignatura;
                rdfs:range ex:PersonalDocente.
```

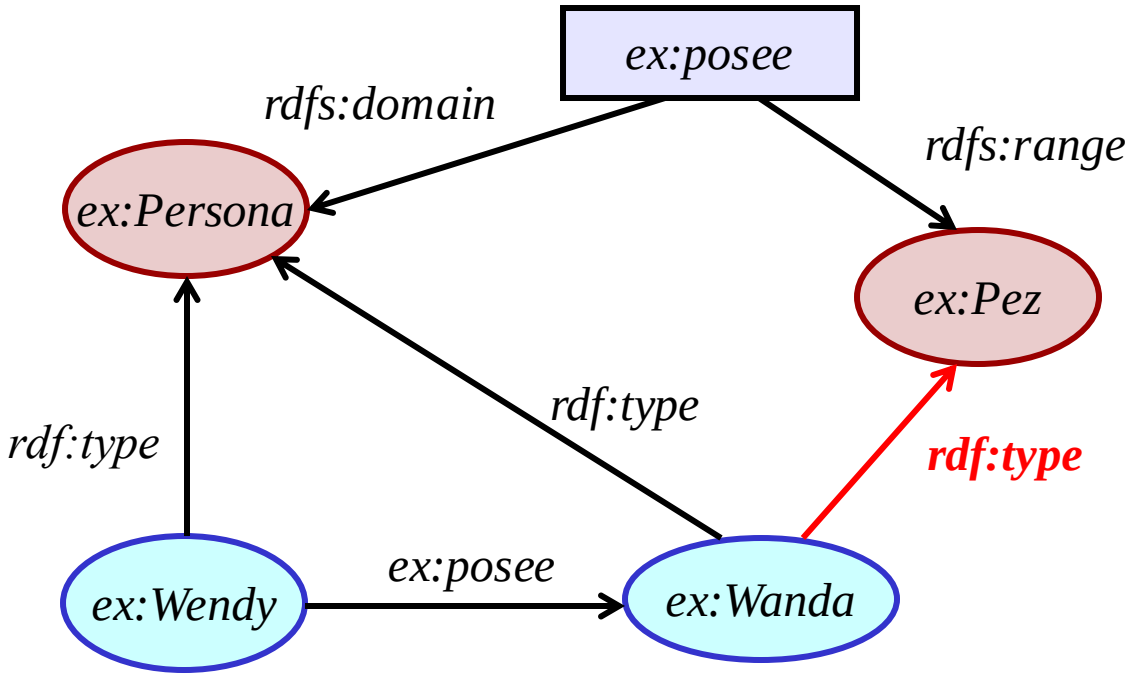
- Jerarquías de propiedades (*rdfs:subPropertyOf*)

```
ex:impartidaPor rdfs:domain ex:Asignatura;
                rdfs:range ex:PersonalDocente;
                rdfs:subPropertyOf ex:involucra.
```

RDF Schema (RDFS)



RDF Schema (RDFS)



¿Violación de la restricción: El rango de posee es Pez?

No hay inconsistencia: Wanda es un pez! *Sirena?*

RDF Schema (RDFS)

- Algunas otras propiedades
  - *rdfs:comment*: descripción legible (por humanos) del recurso
  - *rdfs:label*: versión legible del nombre del recurso
  - *rdfs:seeAlso*: para indicar un recurso con información adicional

```
dbpedia:Spain rdfs:comment "España es un país...";
              rdfs:label "España";
              rdfs:seeAlso <http://www.spain.info/>
```

RDF Schema (RDFS)

- Algunas limitaciones de RDFS
  - Básicamente permite la organización de vocabularios en jerarquías
  - Ámbito local de las propiedades
    - Las restricciones de rango no se pueden aplicar a algunas clases solamente (ej: las vacas sólo comen hierba)
  - No permite expresar:
    - Clases disjuntas
      - Ejemplo: *masculino* y *femenino*
    - Combinación booleana de clases
      - Ejemplo: *Persona* = *Hombre* $\cup$ *Mujer*
    - Restricciones de cardinalidad
    - Características especiales de las propiedades
      - *Transitiva, simétrica, inversa de*, …