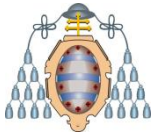
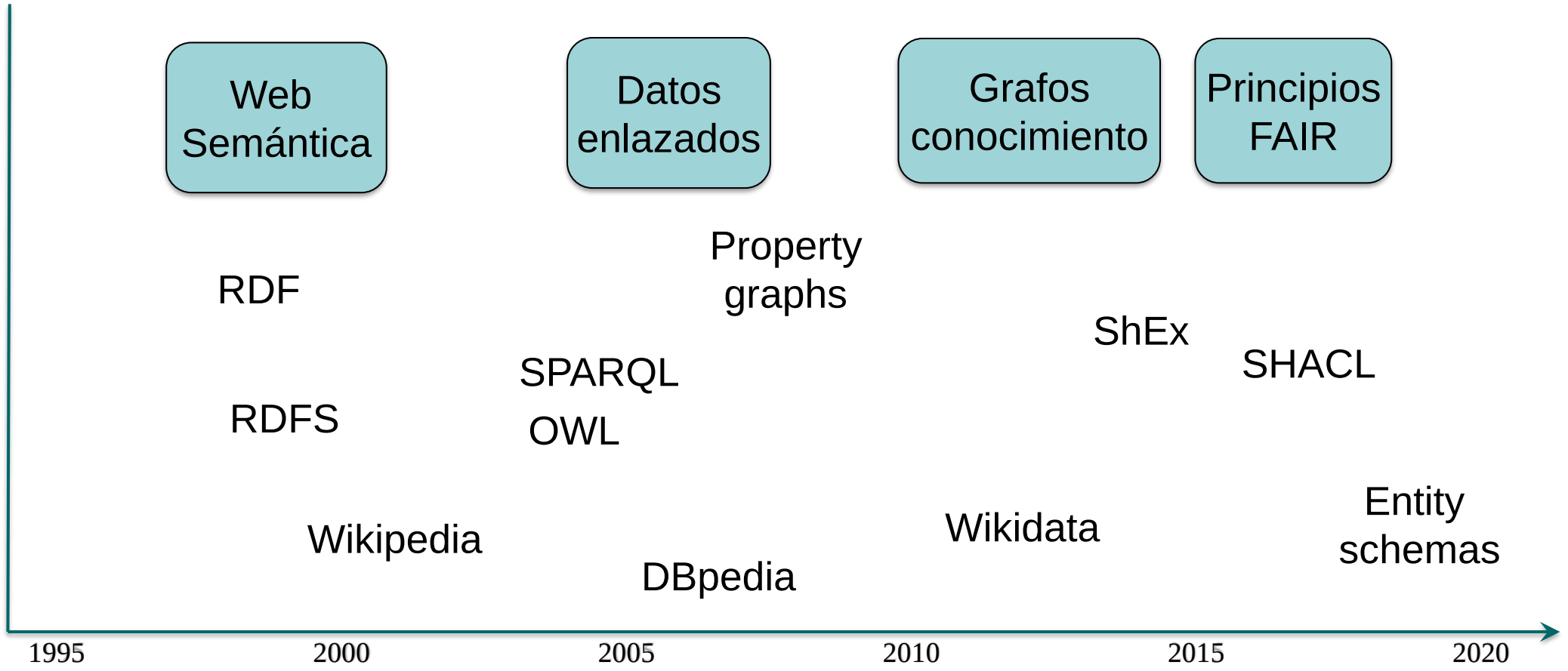


Introducción a RDF

Jose Emilio Labra Gayo

Departamento de Informática
Universidad de Oviedo

Línea temporal



RDF

RDF = Resource Description Framework

Se basa en tripletas y URIs que representan propiedades y nodos

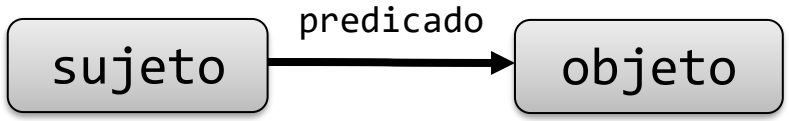
Breve historia

- Hacia 1997 - PICS, Dublin core, Meta Content Framework
- 1997 1st Working draft https://www.w3.org/TR/WD-rdf-syntax-971002, RDF/XML
- 1999 1st W3C Rec https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, XML Syntax, first applications RSS, EARL
- 2004 - RDF Revised https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, SPARQL, Turtle, Linked Data
- 2014 - RDF 1.1 https://www.w3.org/TR/rdf11-concepts/, SPARQL 1.1, JSON-LD
- 2024 - RDF 1.2 https://www.w3.org/TR/rdf12-concepts Enunciados sobre enunciados (RDF-Star)

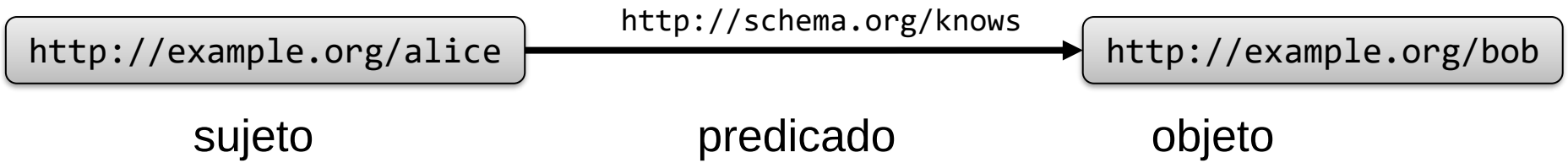
Modelo de datos RDF

RDF está formado por enunciados (statement)

Enunciado = tripleta (sujeto, predicado, objecto)



Ejemplo:



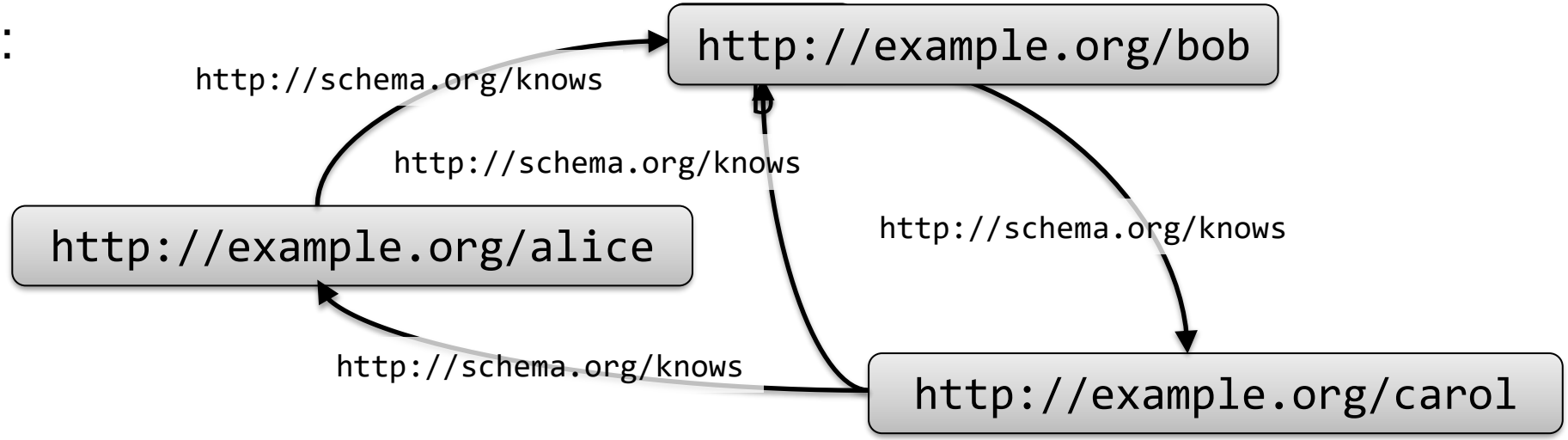
Representación N-Triples

```
<http://example.org/alice> <http://schema.org/knows> <http://example.org/bob> .
```

Conjunto de enunciados = Grafo RDF

Modelo de datos RDF = grafo dirigido

Ejemplo:



Representación en N-triples

```
<http://example.org/alice>   <http://schema.org/knows>   <http://example.org/bob> .
<http://example.org/bob>     <http://schema.org/knows>   <http://example.org/carol> .
<http://example.org/carol>   <http://schema.org/knows>   <http://example.org/alice> .
<http://example.org/carol>   <http://schema.org/knows>   <http://example.org/bob> .
```

sujeto predicado objeto

Notación Turtle

Notación legible por seres humanos que simplifica N-Triples

Permite, por ejemplo, declarar alias de espacios de nombres

N-Triples

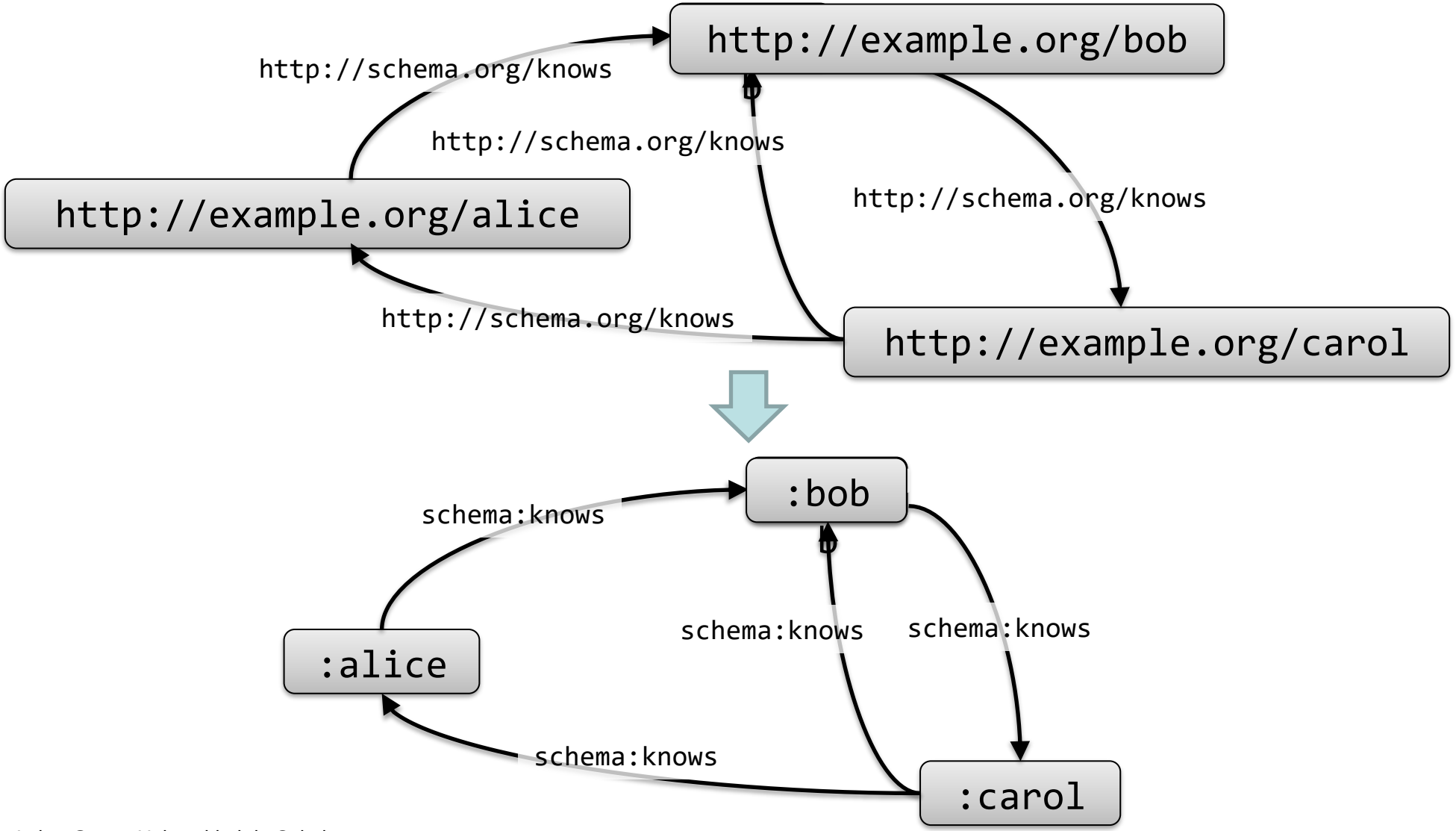
```
<http://example.org/alice>  <http://schema.org/knows>   <http://example.org/bob> .
<http://example.org/bob>    <http://schema.org/knows>   <http://example.org/carol> .
<http://example.org/carol>  <http://schema.org/knows>   <http://example.org/alice> .
<http://example.org/carol>  <http://schema.org/knows>   <http://example.org/bob> .
```

Turtle

```
prefix :       <http://example.org/>
prefix schema: <http://schema.org/>

:alice schema:knows :bob .
:bob   schema:knows :carol .
:carol schema:knows :bob .
:carol schema:knows :alice .
```

Nota:
Más adelante veremos más simplificaciones

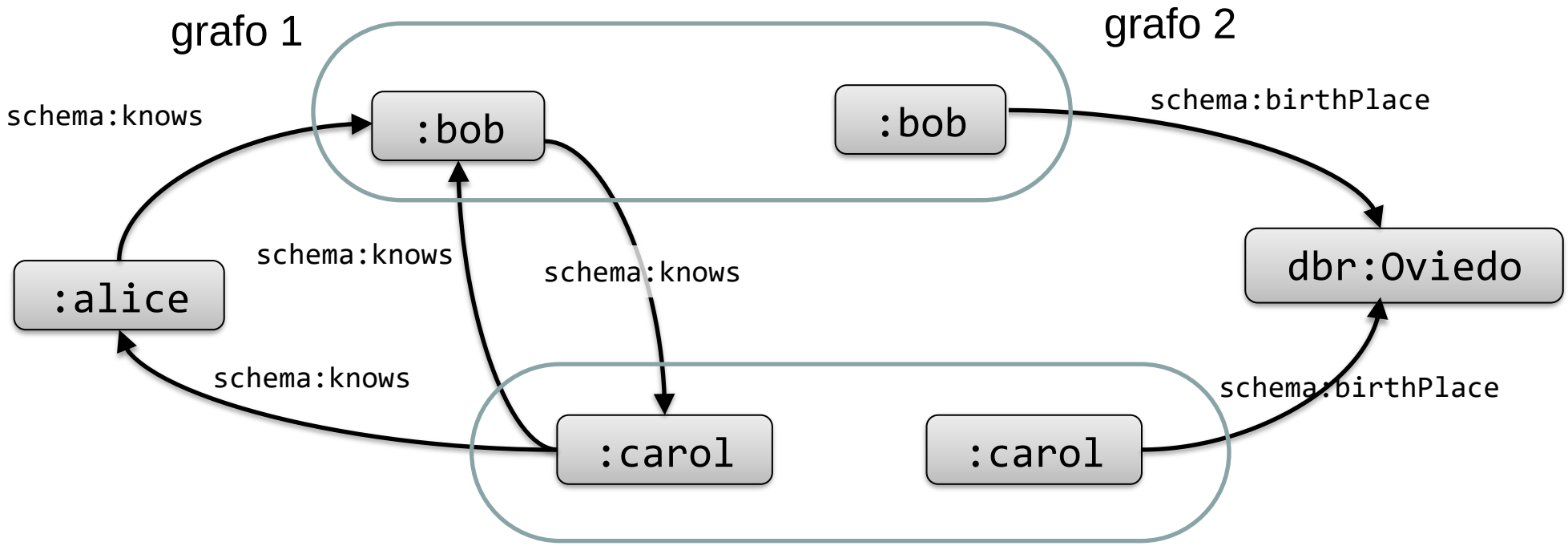
Simplificaciones de espacios de nombres



Propiedad: RDF es composicional

Grafos RDF pueden mezclarse para obtener un grafo más grande

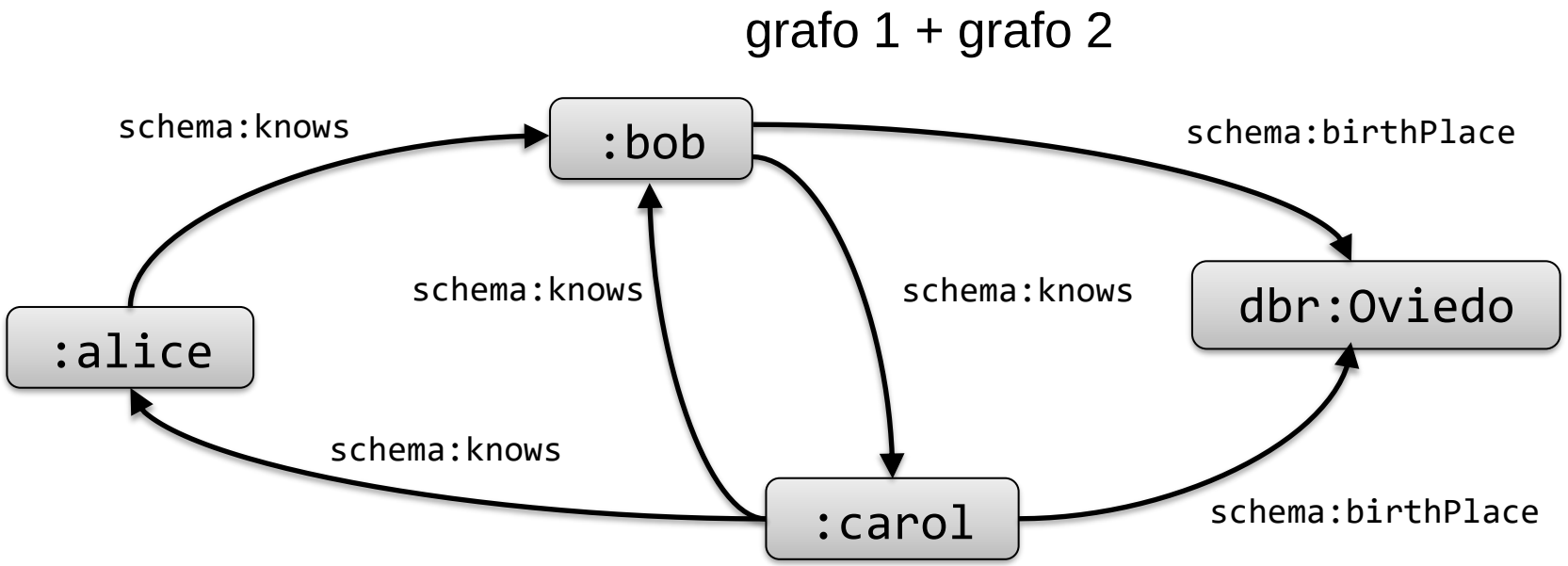
Integración de datos automática



Propiedad: RDF es composicional

Grafos RDF pueden mezclarse para obtener un grafo más grande

Integración de datos automática



Sintaxis Turtle

Algunas simplificaciones

Declaraciones de prefijos

; cuando las tripletas comparten el sujeto

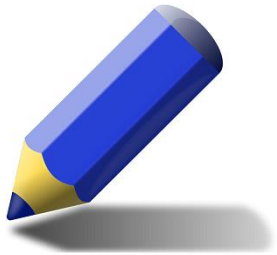
```
:alice schema:birthPlace dbr:Oviedo .
:alice schema:knows      :bob .
```

```
:alice schema:birthPlace dbr:Oviedo ;
       schema:knows      :bob .
```

, cuando las tripletas comparten sujeto y predicado

```
:carol  schema:knows      :alice .
:carol  schema:knows      :bob .
```

```
:carol schema:knows :alice, :bob .
```



Sintaxis Turtle

Ejercicio: simplificar

```
prefix :       <http://example.org/>
prefix schema: <http://schema.org/>
prefix dbr:    <http://dbpedia.org/resource>


:alice schema:knows      :bob .
:bob   schema:knows      :carol .
:carol schema:knows      :bob .
:carol schema:knows      :alice .
:bob   schema:birthPlace dbr:Spain .
:carol schema:birthPlace dbr:Spain .
```

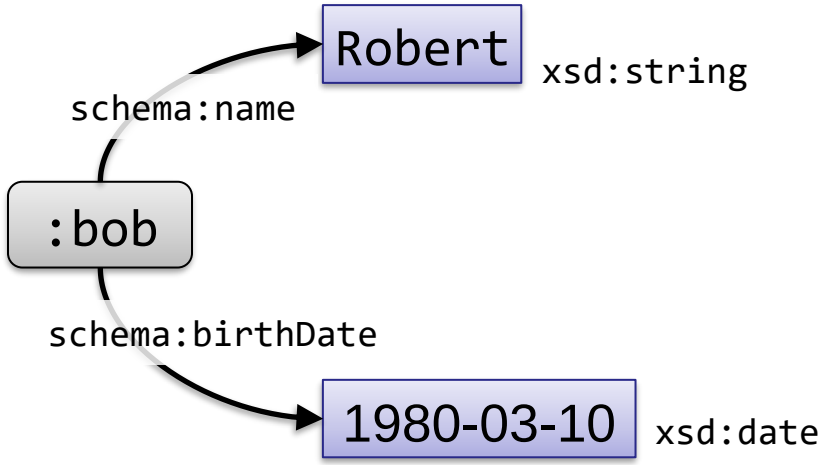
```
prefix ex:     <http://example.org/>
prefix schema: <http://schema.org/>
prefix dbr:    <http://dbpedia.org/resource>

:alice schema:knows      :bob , :carol.
:bob   schema:knows      :carol ;
       schema:birthPlace dbr:Spain .
:carol schema:knows      :bob, :alice ;
       schema:birthPlace dbr:Spain .
```

Try it: https://rdfshape.weso.es/link/16629704380

Literales RDF

Los objetos también pueden ser literales

- Los literales pueden contener una forma léxica y un tipo de datos
- Tipos de datos típicos = Tipos de datos primitivos de XML Schema
  - Si no se especifica, un literal tiene tipo de datos `xsd:string`

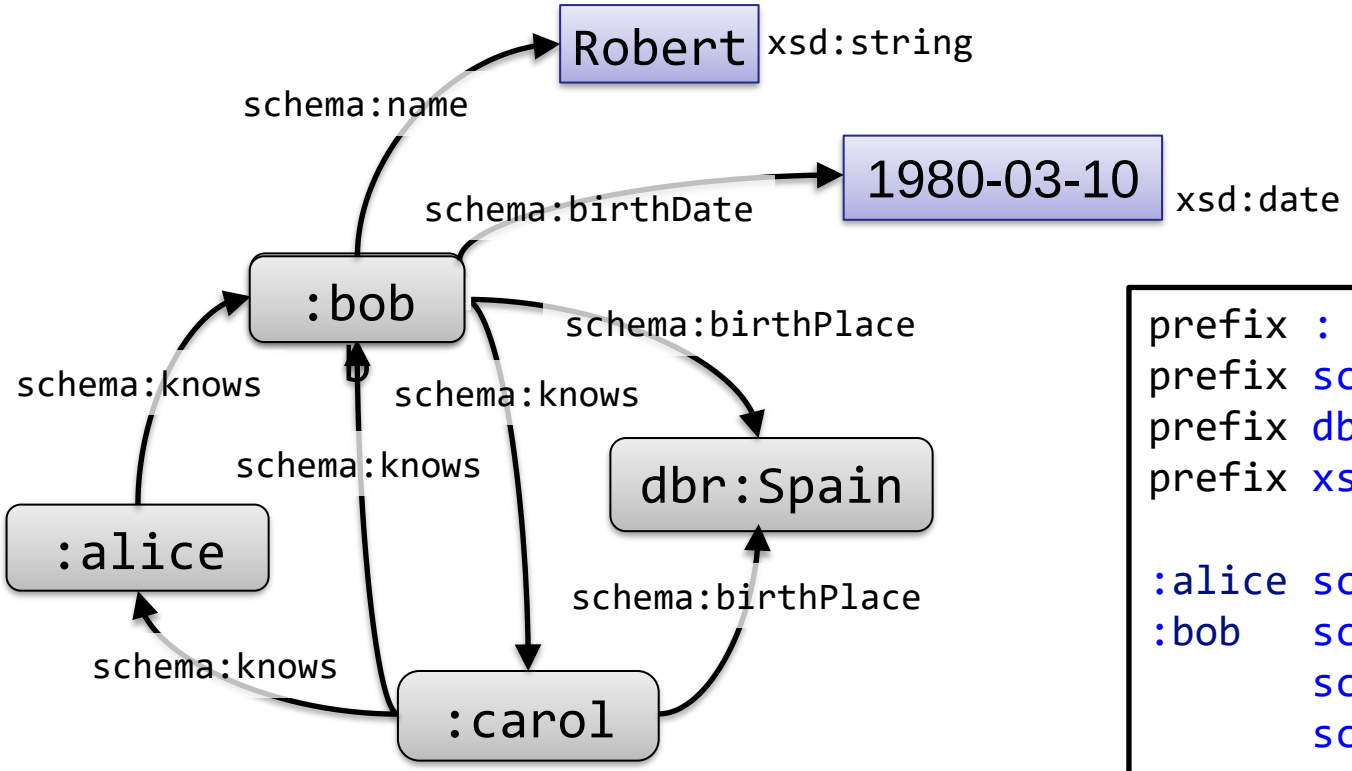


Turtle notation

```
:bob schema:name      "Robert" ;
     schema:birthDate "1980-03-10"^^<xsd:date>.
```

Recordad...RDF es composicional

Mezclando con los datos anteriores




```
prefix :       <http://example.org/>
prefix schema: <http://schema.org/>
prefix dbr:    <http://dbpedia.org/resource>
prefix xsd:    <http://www.w3.org/2001/XMLSchema#>

:alice schema:knows      :bob, :carol .
:bob   schema:knows      :carol    ;
       schema:birthPlace dbr:Spain ;
       schema:name       "Robert"  ;
       schema:birthDate  "1980-03-10"^^<xsd:date> .
:carol schema:knows      :bob, :alice ;
       schema:birthPlace dbr:Spain .
```

https://rdfshape.weso.es/link/16629705496

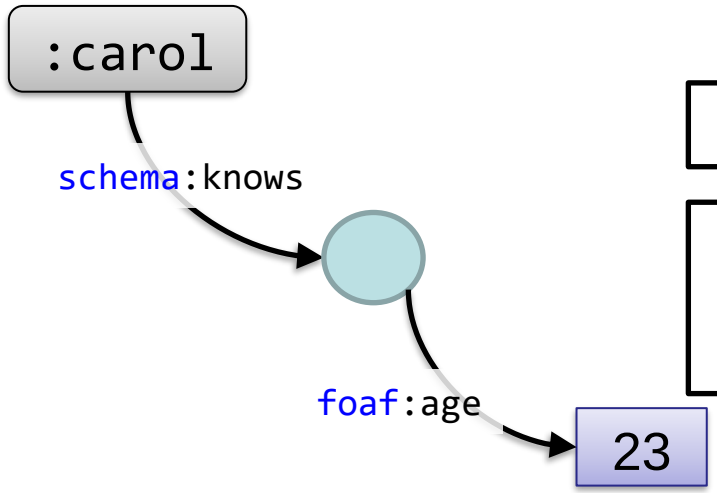
Nodos anónimos

Los sujetos y los objetos también pueden ser nodos anónimos

"Carol conoce a alguien cuya edad es 23"

Notación Turtle con identificador local

```
:carol schema:knows    _:x .
_:x    foaf:age        23  .
```



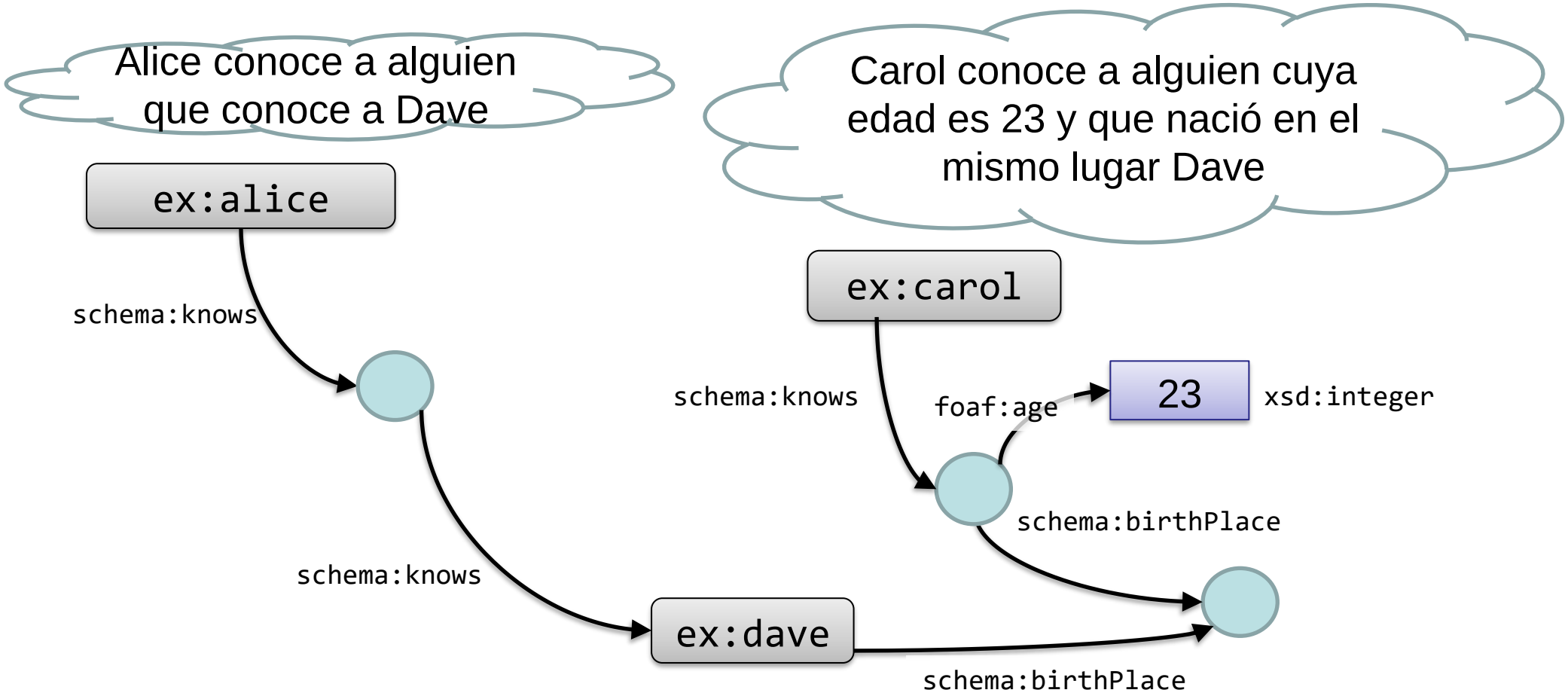
Notación Turtle con corchetes

```
:carol schema:knows [foaf:age 23 ] .
```

Significado matemático:

$\exists x(\texttt{schema:knows}(\texttt{:carol},x) \wedge \texttt{foaf:age}(x, \texttt{23}))$

Nodos anónimos

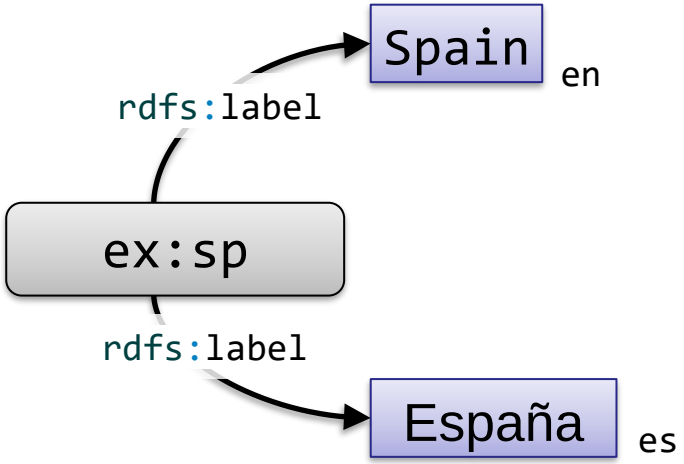



```
:alice schema:knows     [ schema:knows :dave ]    .
:carol schema:knows     [ :age                23    ;
                          schema:birthPlace _:p ] .
:dave schema:birthPlace _:p .
```

Strings con etiqueta de idioma

Los literales de tipo String pueden estar cualificado con una etiqueta de idioma

Tienen un tipo de datos `rdfs:langString`

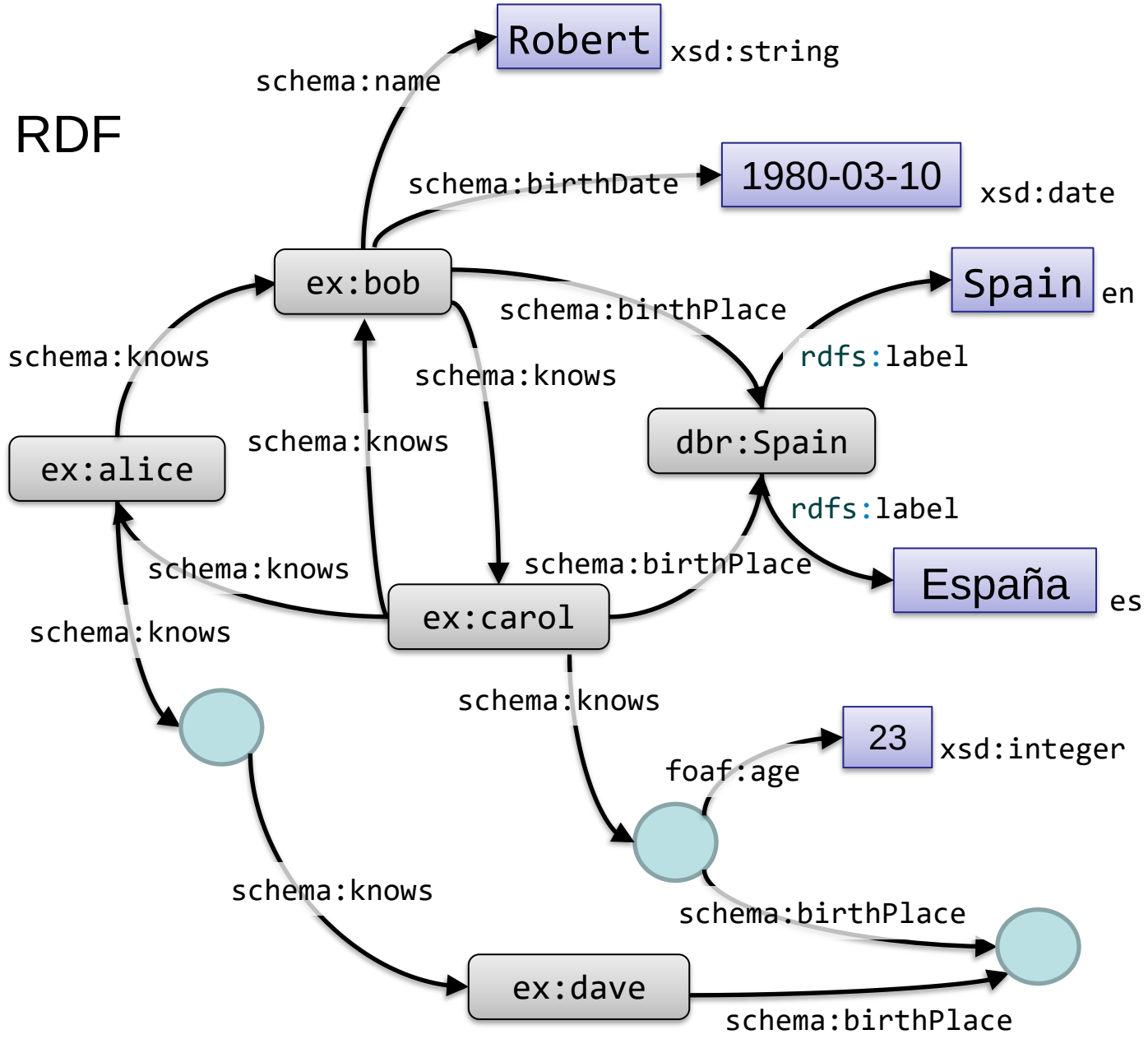


Notación Turtle

```
ex:sp rdfs:label "Spain"@en .
ex:sp rdfs:label "España"@es .
```

Modelo de datos RDF

Ejemplo de datos RDF



3 tipos de nodos

- IRIs
- Nodos anónimos
- Literales

Sujetos: URIs ó Nodos anónimos
Objetos: URIs, nodos anónimos o literales
Predicados siempre son URIs

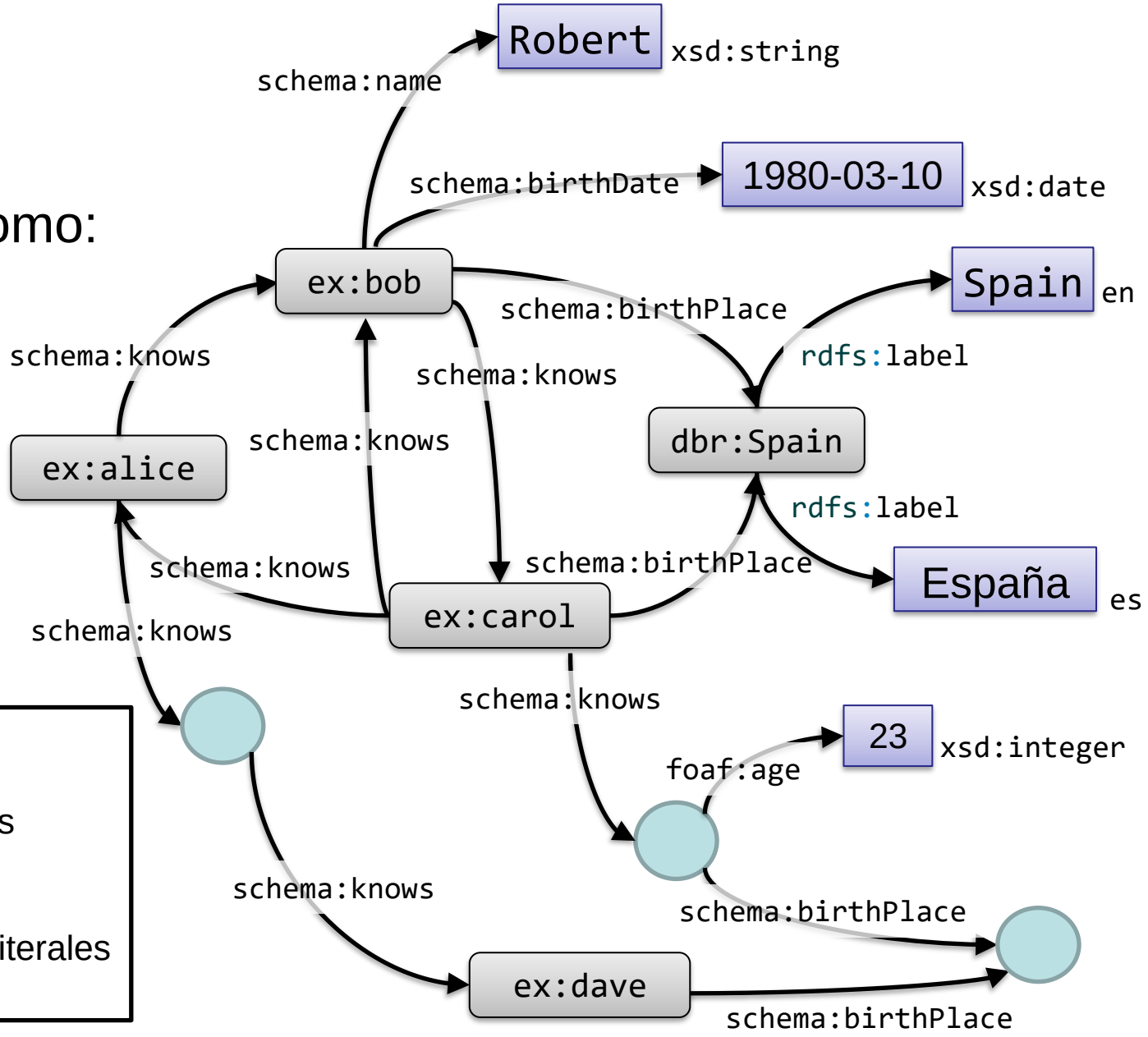
Definición formal del modelo de datos RDF

La mayoría de artículos tienen algo como:

Given a set of IRIs $\mathcal{I}$,
a set of blank nodes $\mathcal{B}$
and a set of literals Lit
an *RDF graph* is a tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{P}, \mathcal{O}, \rho \rangle$
where
$\mathcal{S} = \mathcal{I} \cup \mathcal{B}$,
$\mathcal{P} = \mathcal{I}$,
$\mathcal{O} = \mathcal{I} \cup \mathcal{B} \cup Lit$
$\rho \subseteq \mathcal{S} \times \mathcal{P} \times \mathcal{O}$

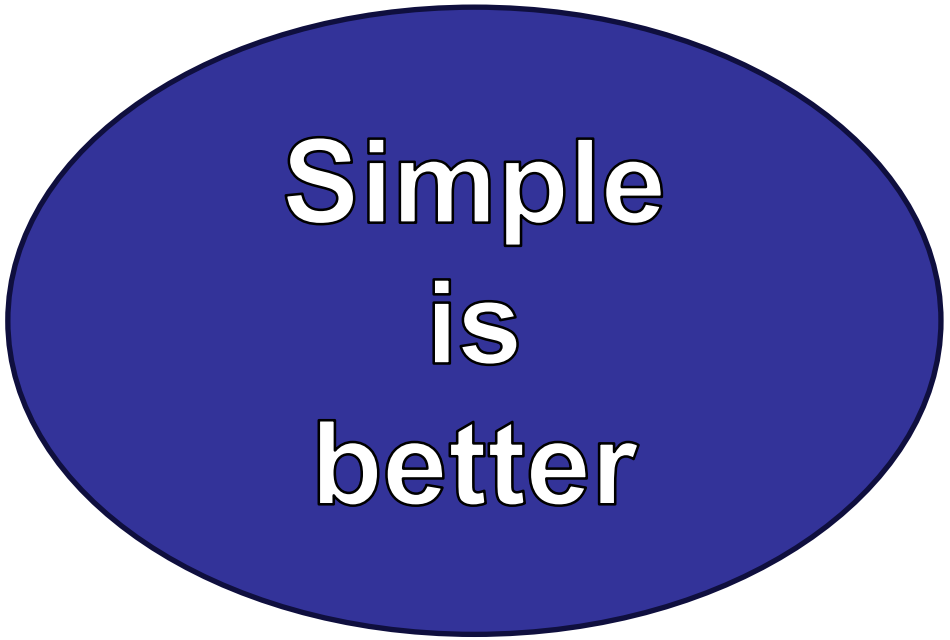
3 tipos de nodos
- IRIs
- Nodos anónimos
- Literales

Sujetos: URIs ó Nodos anónimos
Objetos: URIs, nodos anónimos o literales
Predicados siempre son URIs



...y eso es todo sobre el modelo de datos RDF

El modelo de datos RDF es muy simple



Ecosistema RDF

Sintaxis RDF (RDF/XML, N-Triples, Turtle, JSON-LD,...)

Entidades compartidas, vocabularios, ontologías (RDFS, OWL, SKOS,...)

Lenguaje de consultas (SPARQL, TripleStores...)

Descripción y validación RDF (ShEx, SHACL)

Sintaxis RDF

Primera sintaxis se basaba en XML: RDF/XML

N-Triples (enumera todas las tripletas separadas por puntos)

Turtle (legibilidad humans)

JSON-LD

...otras sintaxis...

....muchas sintaxis pero un modelo de datos único

RDF/XML

Primera sintaxis

No muy popular

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://example.org/"
         xmlns:schema="http://schema.org/">
<rdf:Description rdf:about="http://example.org/carol">
 <schema:knows>
  <rdf:Description rdf:about="http://example.org/bob">
   <schema:knows rdf:resource="http://example.org/carol"/>
   <schema:name>Robert</schema:name>
   <schema:birthDate rdf:datatype="xsd:date">1980-03-10</schema:birthDate>
  </rdf:Description>
 </schema:knows>
 <schema:knows>
  <rdf:Description rdf:about="http://example.org/alice">
   <schema:knows rdf:resource="http://example.org/bob"/>
   <schema:knows rdf:resource="http://example.org/carol"/>
  </rdf:Description>
 </schema:knows>
 <schema:knows rdf:parseType="Resource">
  <age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">23</age>
 </schema:knows>
 </rdf:Description>
</rdf:RDF>
```

N-Triples

Para realizar pruebas y facilitar el análisis sintáctico

...simplemente tripletas separadas por puntos

```
<http://example.org/carol> <http://schema.org/knows>     <http://example.org/bob> .
<http://example.org/carol> <http://schema.org/knows>     <http://example.org/alice> .
<http://example.org/carol> <http://schema.org/knows>     _:x .
_:x                        <http://example.org/age>      "23"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://example.org/alice> <http://schema.org/knows>     <http://example.org/bob> .
<http://example.org/alice> <http://schema.org/knows>     <http://example.org/carol> .
<http://example.org/bob>   <http://schema.org/knows>     <http://example.org/carol> .
<http://example.org/bob>   <http://schema.org/name>      "Robert" .
<http://example.org/bob>   <http://schema.org/birthDate> "1980-03-10"^^<http://www.w3.org/2001/XMLSchema#date> .
```

Turtle

Conciso

Diseñado para ser legible por humanos

```
prefix :       <http://example.org/>
prefix schema: <http://schema.org/>

:alice schema:knows     :bob, :carol .
:bob   schema:knows     :carol                    ;
       schema:name      "Robert"                  ;
       schema:birthDate "1980-03-10"^^<xsd:date>.
:carol schema:knows     :bob, :alice ;
       schema:knows     [ :age 23 ]  .
```

JSON-LD

Json for Linked Data

```
{
"@context" : {
"knows" : { "@id" : "http://schema.org/knows", "@type" : "@id" },
"age" : { "@id" : "http://example.org/age",
          "@type" : "http://www.w3.org/2001/XMLSchema#integer" },
"name" : { "@id" : "http://schema.org/name" },
"birthDate" : { "@id" : "http://schema.org/birthDate", "@type" : "xsd:date" },
"@vocab" : "http://example.org/",
"schema" : "http://schema.org/"
},
"@graph" : [
 { "@id" : "http://example.org/alice",
  "knows" : [ "http://example.org/bob", "http://example.org/carol" ] },
 { "@id" : "http://example.org/bob",
  "birthDate" : "1980-03-10",
  "knows" : "http://example.org/carol",
  "name" : "Robert" },
 { "@id" : "http://example.org/carol",
  "knows" : [ "http://example.org/bob", "http://example.org/alice", "_:x" ] },
 { "@id" : "_:x",
  "http://example.org/age" : 23 }
 ]
}
```

Otras simplificaciones de Turtle

Propiedad RDF type

Constantes

Colecciones

Propiedad RDF type

La propiedad `rdf:type` declara el tipo de un recurso

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

e:alice rdf:type schema:Person .
e:bob   rdf:type schema:Person .
```

`rdf:type` puede simplificarse como `a`

```
@prefix schema: <http://schema.org/> .

:alice a schema:Person .
:bob   a schema:Person .
```

Constantes

Números y valores booleanos pueden representarse sin comillas

Son analizados como tipos de datos XML Schema

Tipo de datos	Ejemplo simplificado	Representación interna
xsd:integer	3	"3"^^xsd:integer
xsd:decimal	-3.14	"-3.14"^^xsd:decimal
xsd:double	3.14e2	"3.14e2"^^xsd:double
xsd:boolean	true	"true"^^xsd:boolean

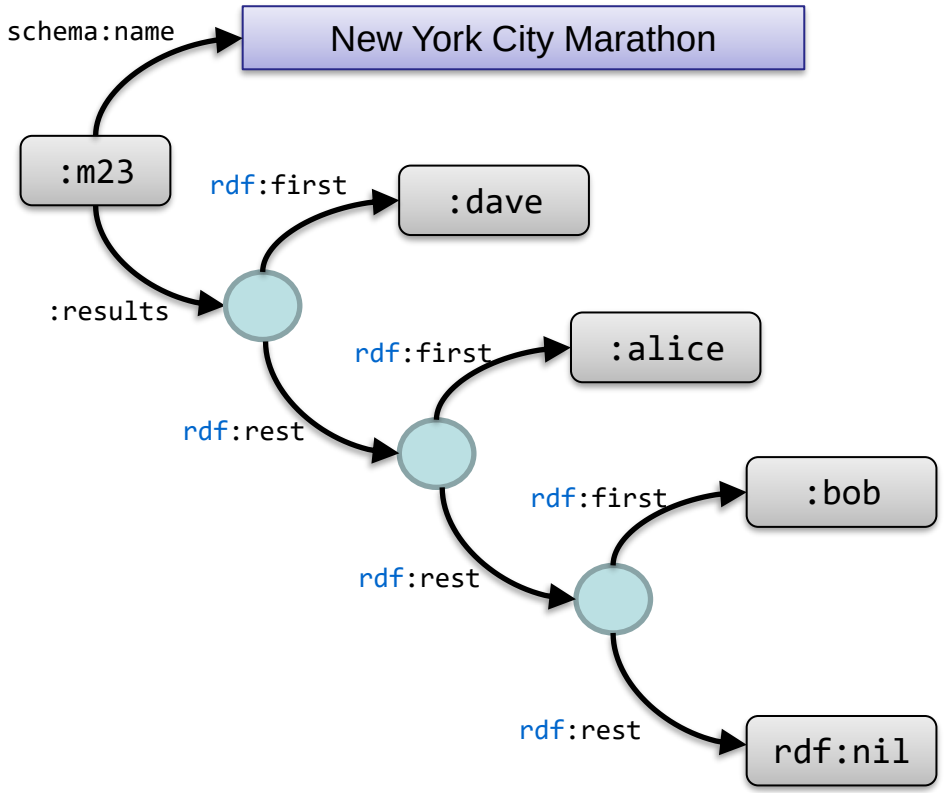
Colecciones

Listas ordenadas

```
:m23 schema:name "New York City Marathon ";
     :results    ( :dave :alice :bob ) .
```

Internamente, se representan como listas ordenadas

```
:m23 schema:name "New York City Marathon ";
     :results _:1 .
_:1 rdf:first :dave ;
    rdf:rest _:2 .
_:2 rdf:first :alice ;
    rdf:rest _:3 .
_:3 rdf:first :bob ;
    rdf:rest rdf:nil .
```



SPARQL

SPARQL Protocol And RDF Query Language

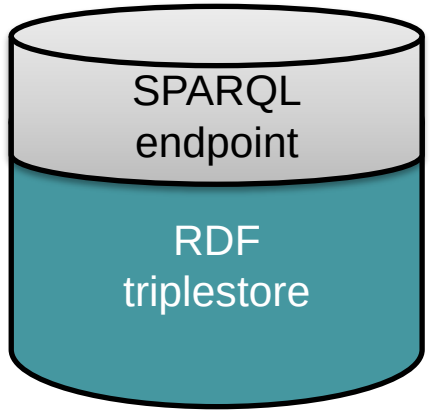
SPARQL 1.0 (2008), 1.1 (2013)

Syntaxis inspirada por Turtle

Semántica basada en patrones de grafos

SPARQL endpoints = servicios que implementan el protocolo SPARQL

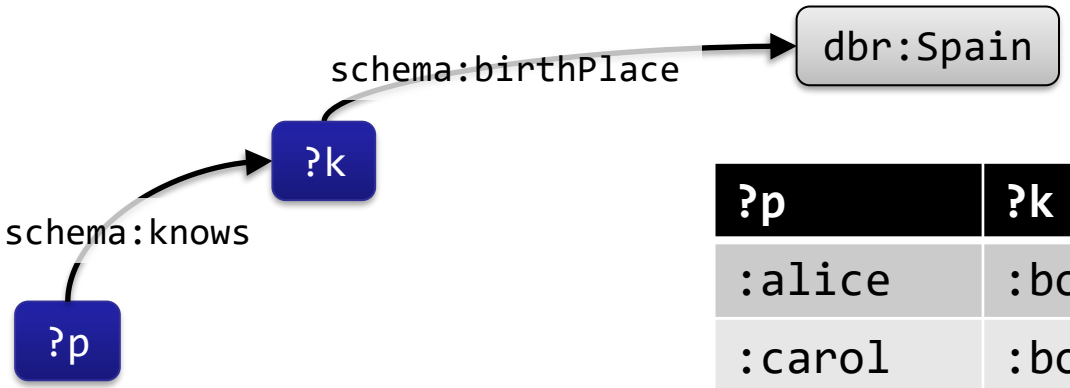
Triplestores = Bases de datos RDF



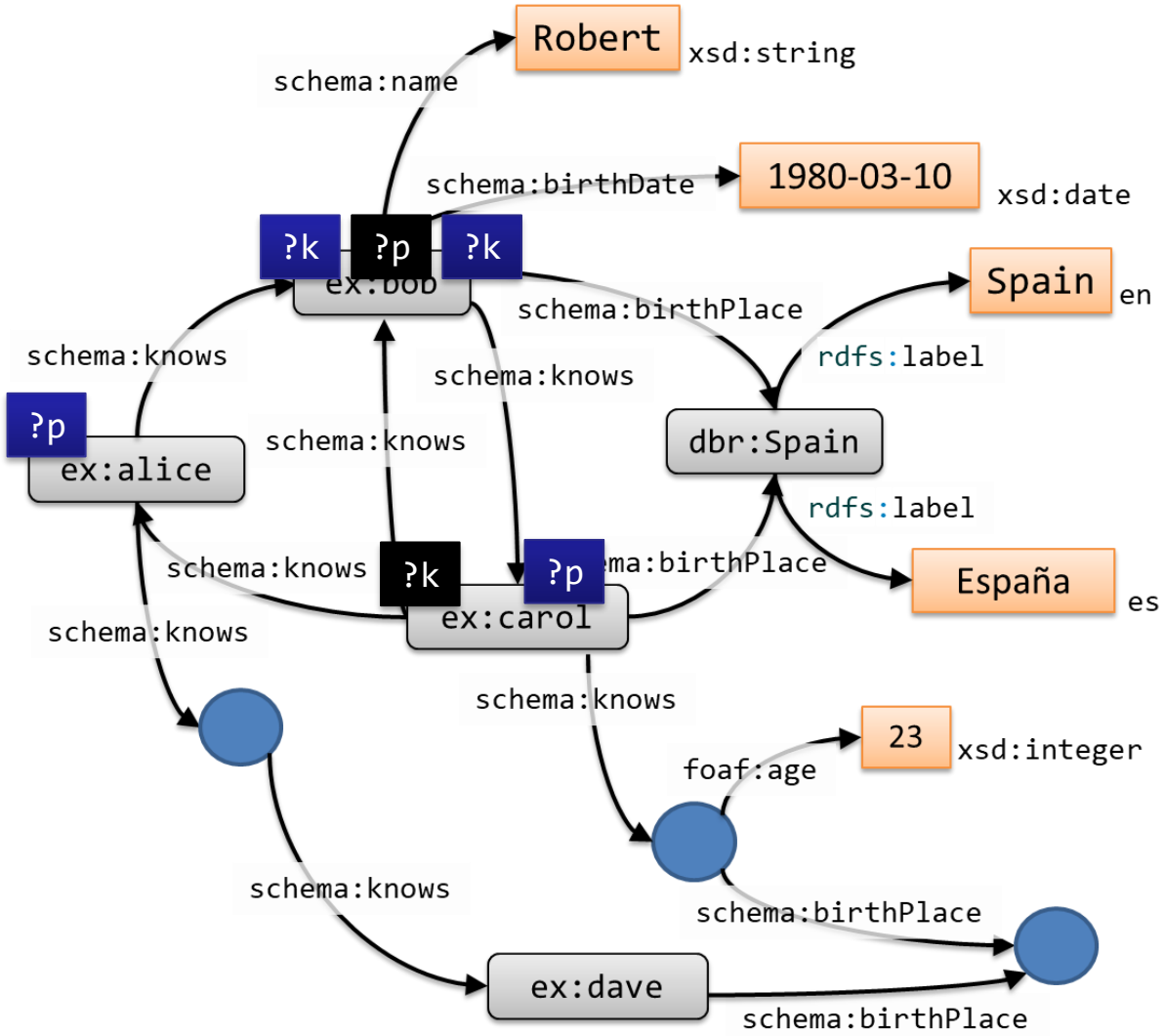
Ejemplo SPARQL

"Gente que conoce alguien cuyo lugar de nacimiento es España"

```
prefix schema: <http://schema.org/>
prefix dbr:    <http://dbpedia.org/resource/>

select ?p ?k where {
  ?p schema:knows       ?k  .
  ?k schema:birthPlace dbr:Spain
}
```



?p	?k
:alice	:bob
:carol	:bob
:bob	:carol



Entidades compartidas y vocabularios

El uso de URIs en lugar de cadenas de texto facilita:

- Combinar datos de fuentes heterogéneas
- Evitar ambigüedad

Reto: Ponerse de acuerdo sobre entidades y propiedades compartidas

Algunos vocabularios populares:

- schema.org: Esfuerzo conjunto de Google, Yahoo, Microsoft, Yandex
- Proyecto Linked open vocabularies: http://lov.okfn.org/

Algunos vocabularios y espacios de nombres populares

Alias	URL	Nombre	Algunas propiedades
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#	RDF	type, subject, predicate, object,…
rdfs:	http://www.w3.org/2000/01/rdf-schema#	RDF Schema	domain, range, Class, Property, subClassOf,…
owl:	http://www.w3.org/2002/07/owl#	OWL Ontologies	sameAs, intersectionOf, unionOf, …
dc:	http://purl.org/dc/elements/1.1/	Dublin Core	author, date, creator, …
Schema:	http://schema.org/	Schema.org	name, knows, etc.
skos:	http://www.w3.org/2008/05/skos#	SKOS	broader, narrower, …

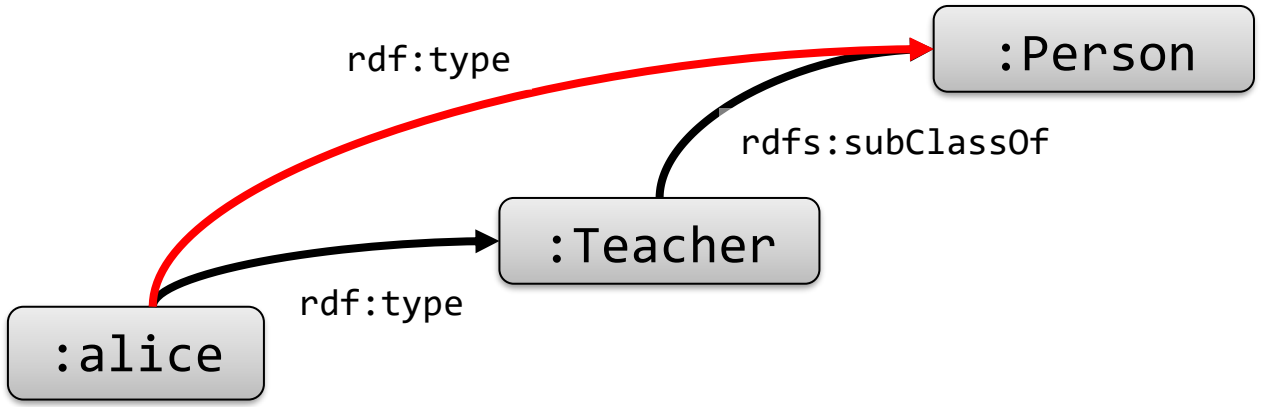
El servicio http://prefix.cc puede usarse para encontrar el alias más popular para una URI

El poder de los vocabularios compartidos

RDFS (Inicialmente RDF Schema) define conceptos comunes

- Clases: `rdfs:Class`, `rdfs:Property`, `rdfs:Literal`
- Propiedades: `rdfs:domain`, `rdfs:range`, `rdfs:subClassOf`, ...

Los procesadores RDFS pueden inferir nuevas tripletas



IF x `rdf:type` A AND
A `rdfs:subClassOf` B
THEN
x `rdf:type` B

De vocabularios compartidos a ontologías

OWL = Web Ontology Language.

OWL 1 (2004), OWL 2 (2009)

Se basa en lógica descriptiva

Describe clases, propiedades, individuos y sus relaciones

Muy expresivo con un mecanismo de inferencia muy potente

Ejemplo OWL

Ontología simple, parte Terminológica (TBox)

$$Person \sqsubseteq = 2\, hasParent$$
$$Person \sqsubseteq \exists\, hasParent\, Male$$
$$Person \sqsubseteq \exists\, hasParent\, Female$$

$$\forall x(Person(x) \rightarrow \exists y(hasParent(x,y) \wedge Male(y)))$$

$$Male \sqsubseteq \neg Female$$
$$Female \sqsubseteq \neg Male$$

```
:Person rdf:type owl:Class ;
  rdfs:subClassOf [
   rdf:type owl:Restriction ;
   owl:onProperty :hasParent ; owl:cardinality 2
  ], [
  rdf:type owl:Restriction ;
  owl:onProperty :hasParent ; owl:someValuesFrom :Male
  ], [
  rdf:type owl:Restriction ;
  owl:onProperty :hasParent ; owl:someValuesFrom :Female
  ] .

:Female owl:disjointWith :Male .
```

Datos de instancias, parte de enunciados = ABox

$$Person(alice)$$
$$hasParent(alice, bob)$$
$$hasParent(alice, carol)$$
$$Female(carol)$$

```
:alice rdf:type :Person ;
       :hasParent :bob,
                  :carol .
:carol rdf:type :Female .
```

Datos Inferidos

$$Male(bob)$$

```
:bob rdf:type :Male .
```

El ejemplo no es completo...
¿Qué falta?

Necesitamos declarar que `:bob` ≠ `:carol`

```
[ rdf:type owl:AllDifferent ;
  owl:distinctMembers ( :bob
                        :carol
                      ) ] .
```

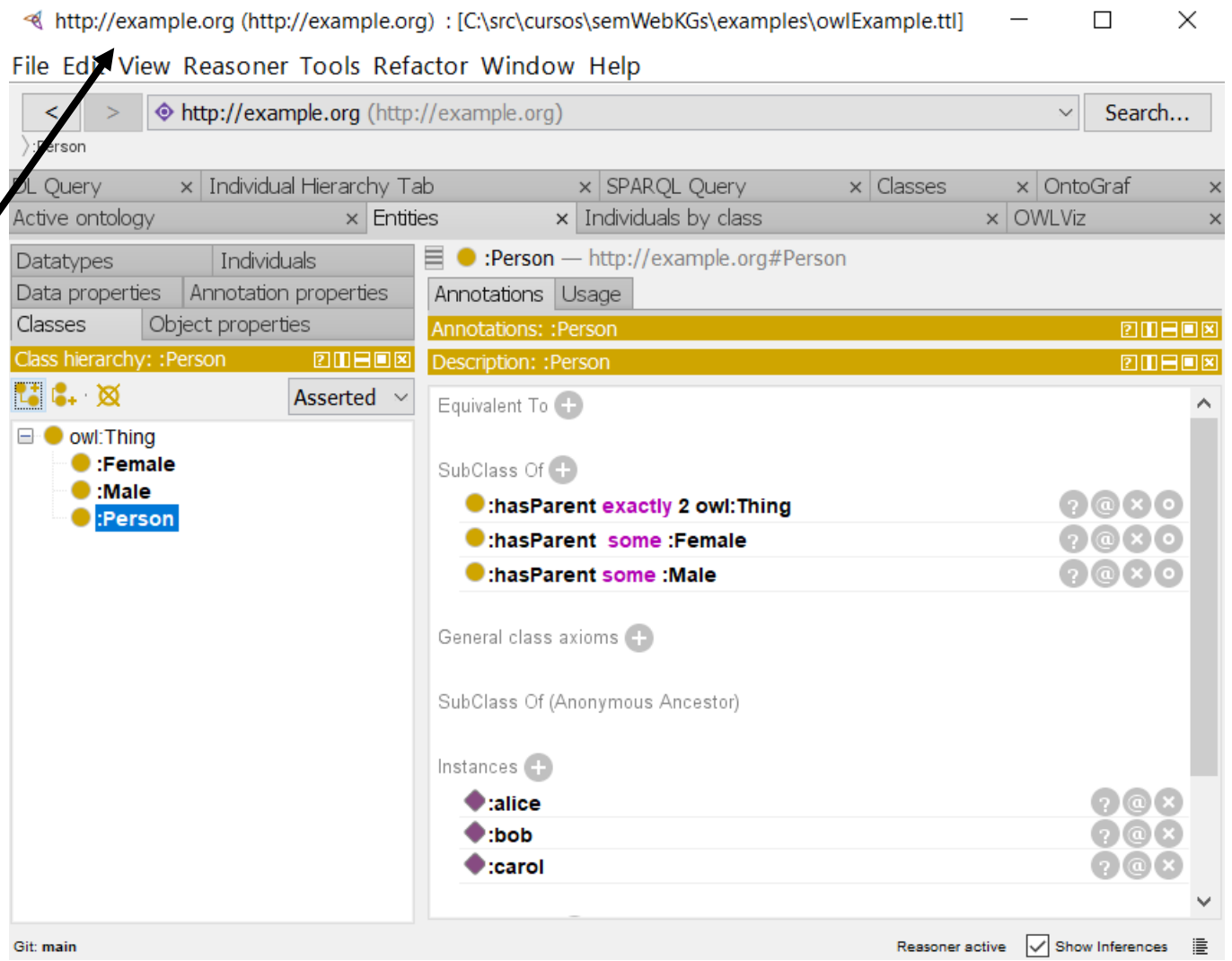
OWL

OWL = Lenguaje para describir ontologías

Diferentes tipos de ontologías:

- Ontologías de alto nivel (SUMO, BFO, ...)
- Ontologías de dominio específico

Editores de ontologías como Editor Protégé



Los conceptos de ontología tienen una URI
Pueden ser definidos y almacenados en ficheros locales
¿Cómo podrían ser publicados?

Modelado de datos en RDF

Representando información en RDF

RDF = modelo de datos para intercambiar información en Web

Algunas consideraciones & compromisos

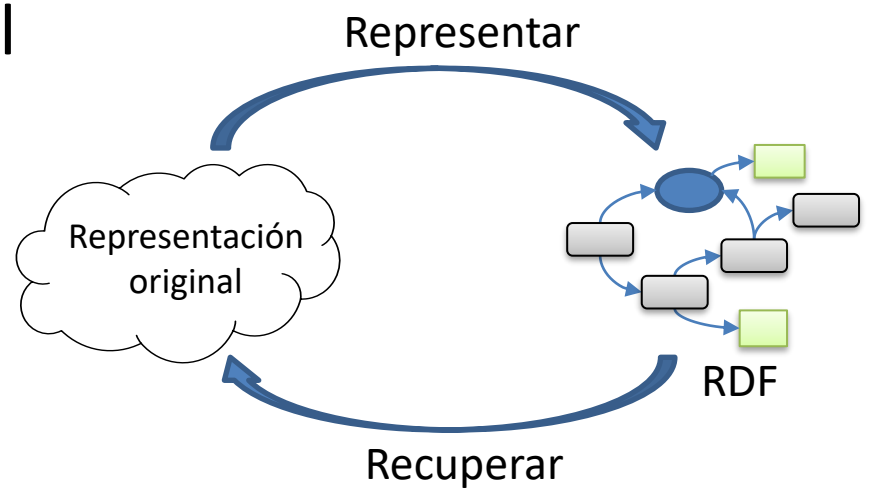
- Precisión semántica (*Semantic accuracy*)
- Legibilidad humana
- Flexibilidad y falta de esquema
- Interoperabilidad & rendimiento

Precisión semántica (*semantic accuracy*)

Evitar pérdidas semánticas

Ida y vuelta (*Round-tripping*)

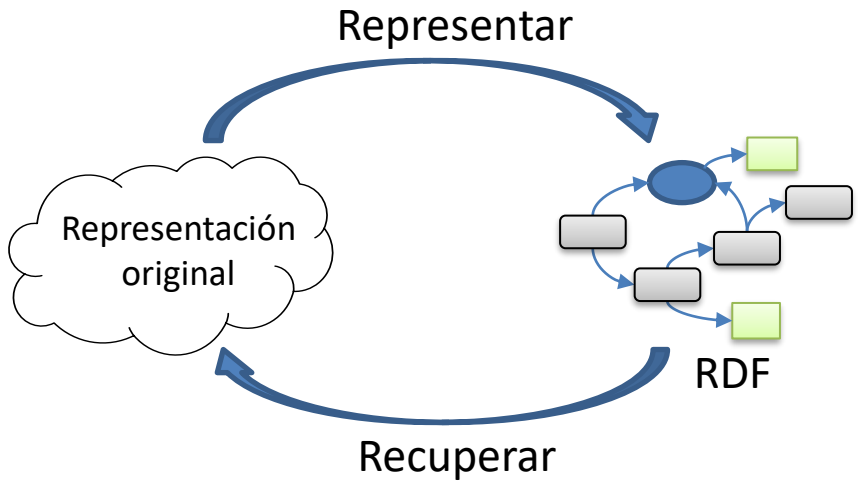
- De la represetación original a RDF
- De RDF recuperar representación original



Precisión semántica

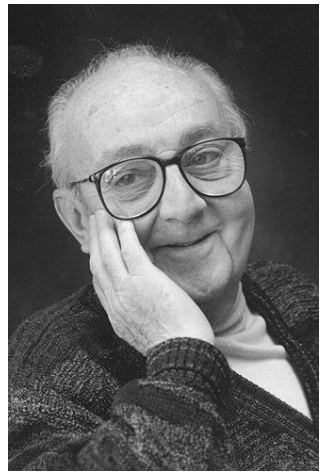
Debemos tener cuidado sobre

Relación mapa-territorio



"*Todos los modelos están equivocados, pero algunos son útiles*"

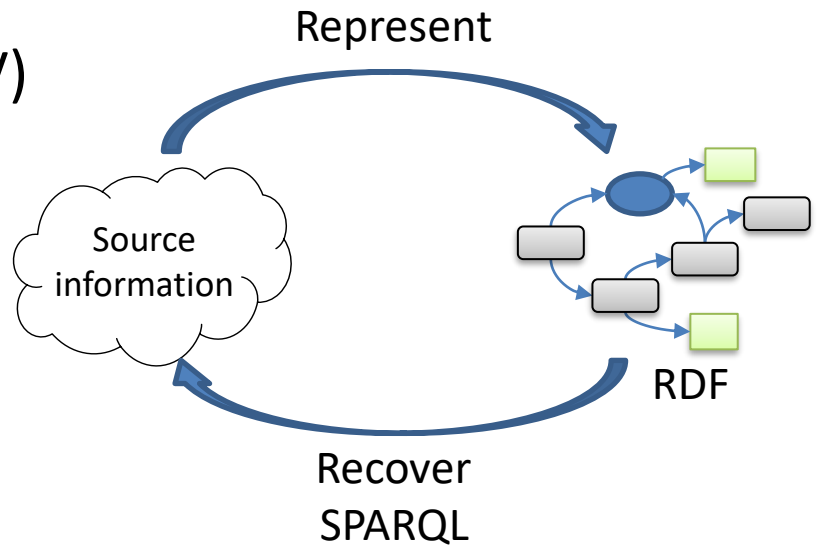
G. Box aphorism



George Box, fuente: Wikipedia

Precisión semántica

- Representar en RDF
  - Convertir de datos existentes
    - Bases de datos relacionales y datos tabulares (CSV)
    - Datos jerárquicos: XML, JSON
- Recuperar a partir de RDF
  - Consultas SPARQL



Legibilidad humana

- RDF como lenguaje de comunicación
  - Turtle puede ser legible por humanos
  - Útil para depuración
- Grandes conjuntos de datos RDF = ilegibles
  - Metáfora de grafo puede no ser útil para datos grandes en la práctica

Flexibilidad y falta de esquema

- RDF es muy flexible
  - Varias formas de modelar/representar la misma información
  - Sin esquema: No es necesario comprometerse a un esquema estricto
- ¿Demasiada libertad?
  - Normalmente tenemos algún esquema implícito
  - Conocer la estructura de los datos puede ser útil
    - Mejorar la comunicación y la documentación
    - Menos necesidad de programación defensiva
    - Posibles optimizaciones y más seguridad

Interoperabilidad

- Datos RDF deberían ser procesables por las máquinas
  - Adoptar vocabularios y URIs comunes
  - No reinventar la rueda
  - Evitar ambigüedad
  - Proporcionar context y procedencia de las declaraciones
- Verborrea (*verbosity*)
  - Demasiada información puede decrecer la legibilidad/rendimiento
    - Ejemplo: contenido audiovisual

Hacia una metodología de modelado de datos RDF

Antes de modelar

Identificar *stakeholders*

Crear preguntas de competencia

Recoger ejemplos

Licencias y procedencia

Modelando datos

Seleccionar vocabularios

Diseñar URIs

Definir shapes de datos

Preparar infraestructura

Convertir fuentes existentes

Después de modelar

Mantener *pipelines*

Documentar *endpoint*

- Ejemplos, consultas, *shapes*,...

Hacer partícipes a usuarios

- Apps de ejemplo y APIs
- Hackathones
- Visualizaciones de datos

Fase de modelado de datos RDF

Seleccionar vocabularios

Diseño de URIs

Definir shapes de datos

Preparar infraestructura

Conversión de fuentes existentes

Seleccionar vocabularios

- Encontrar vocabularios existentes
  - Ejemplos:
    - LOV: Linked open vocabularies: https://lov.linkeddata.es/dataset/lov/
    - Bioportal: https://bioportal.bioontology.org/
- ¿Crear nuevos vocabularios?
  - Algunas veces puede ser necesario
    - Tus conceptos no son exactamente los mismos que los existentes
    - No quieres tener demasiadas dependencias externas
  - Intentar siempre mapear a vocabularios existentes
    - Propiedades para mapear: `owl:sameAs`, `skos:related`, `rdfs:seeAlso`

Seleccionar vocabularios

Tipos de vocabularios

Tesauros (esquemas de conceptos)

- Definen términos y relaciones básicas
- Semántica ligera
- Objetivo: Facilitar búsquedas
- Lenguaje habitual: SKOS
- Ejemplo: https://agclass.nal.usda.gov/

Ontologías

- Definen clases, propiedades, individuos, jerarquías en un dominio
- Semántica puede ser compleja
- Objetivo: Facilitar razonamientos
- Lenguaje habitual: OWL
- Ejemplos: https://obofoundry.org/

Diseño de URIs

- URIs buenas (*Cool URIs*)
  - Cool URIs don't change: https://www.w3.org/Provider/Style/URI
  - Cool URIs for the semantic web: https://www.w3.org/TR/cooluris/
- Algunas decisiones típicas: Opacas vs descriptivas
  - Opaca: http://www.wikidata.org/entity/Q14317
  - Descriptiva: http://dbpedia.org/resource/Oviedo
- Utilizar patrones de URIs
  - Ejemplo: UK URI patterns
    - http://ukgovld.github.io/ukgovldwg/recommendations/uri-patterns.html

Definir shapes de datos

Comprender los datos existentes y *posibles*

Definir topología de grafo RDF

- Esquemas implícitos vs explícitos
- Datos abiertos vs cerrados

Shapes de datos: ShEx, SHACL

¿Patrones de shapes?

Preparar infraestructura

- ¿Dónde almacenamos los datos?
  - No solamente triple-stores de datos RDF nativos
  - Otras posibilidades: bases de datos en grafo, relacionales, etc.
    - RDF como capa de comunicación: Conversión a RDF bajo demanda
  - Habilitar endpoint SPARQL
- Seguir principios de datos enlazados
  - Negociación de contenido
  - Habilitar vistas HTML de los datos

Convertir de fuentes existentes

- Diferente fuentes y modelos de datos
  - Datos jerárquicos: JSON, XML
  - Datos tabulares: Excel, CSV
    - http://shexml.herminiogarcia.com/spec
  - Bases de datos relacionales:
    - Mapeos directos: https://www.w3.org/TR/rdb-direct-mapping/
    - Mapeos más específicos: R2RML https://www.w3.org/TR/r2rml/
  - Lenguaje natural: FRED (http://wit.istc.cnr.it/stlab-tools/fred/)
- Tecnologías de mapeo
  - ¿Mantener información de esquema?

Algunos patrones de modelado de RDF

Relaciones de aridad-N

Datos tabulares

Representación de orden

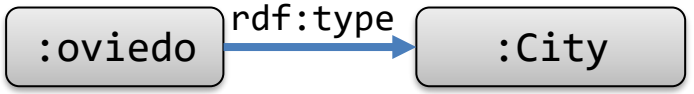
Reificación y procedencia

Agrupando tripletas RDF y conjuntos de datos

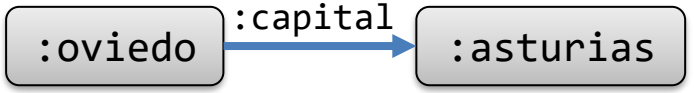
Relaciones de aridad N

Expresar relaciones entre 1, 2,... elementos en RDF

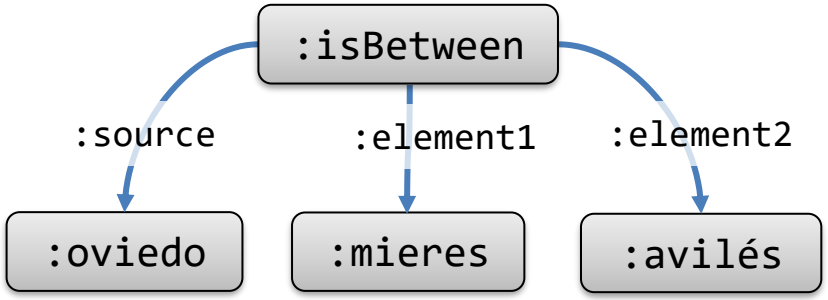
Aridad 1: *Oviedo es una ciudad* `city(Oviedo)`



Aridad 2: *Oviedo es la capital de Asturias* `capital(Oviedo,Asturias)`



Aridad 3: *Oviedo está entre Mieres y Avilés* `isBetween(Oviedo,Mieres,Aviles)`



Técnica habitual (*reificar la relación*)

- Crear un nodo auxilirar que representa la relación
- Añadir nuevas relaciones entre nodos y nodo auxiliar

Defining N-ary Relations on the Semantic Web:
https://www.w3.org/TR/swbp-n-aryRelations/

Representar orden

RDF puede representar conjuntos, pero no listas

Varias soluciones

- Listas enlazadas (Colecciones RDF)
- Propiedades que indican orden (RDF containers)
- Añadir anotaciones de orden a los valores
- Ignorar el orden
- Combinar varias soluciones

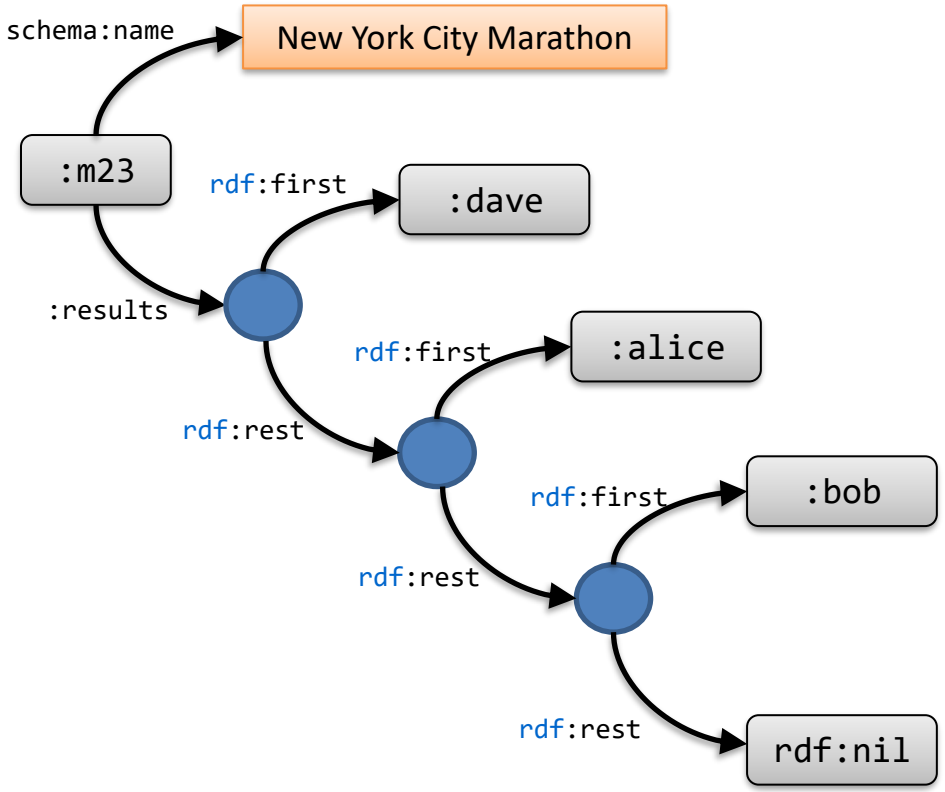
Solución 1
Representar orden con listas enlazadas

Listas ordenadas

```
:m23 schema:name "New York City Marathon ";
     :results    ( :dave :alice :bob ) .
```

Internamente, se represeta como listas enlazadas

```
:m23 schema:name "New York City Marathon ";
     :results _:1 .
_:1 rdf:first :dave ;
    rdf:rest _:2 .
_:2 rdf:first :alice ;
    rdf:rest _:3 .
_:3 rdf:first :bob ;
    rdf:rest rdf:nil .
```



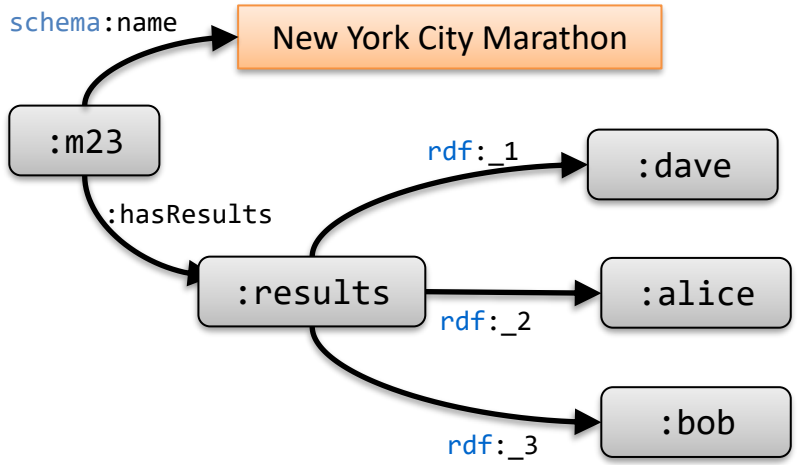
Pros: Representación elegante, fácil para insertar/borrar, marcar fin de lista
Cons: Acceso ineficiente a un elemento dado

Solución 2
Representar orden con propiedades

Utilizar propiedades que indiquen el orden

RDF dispone de algunas propiedades específicas: `rdf:_1, rdf:_2, ...`

```
:m23 schema:name "New York City Marathon ";
     :hasResults :results .
:results rdf:_1 :dave ;
         rdf:_2 :alice ;
         rdf:_3 :bob   .
```



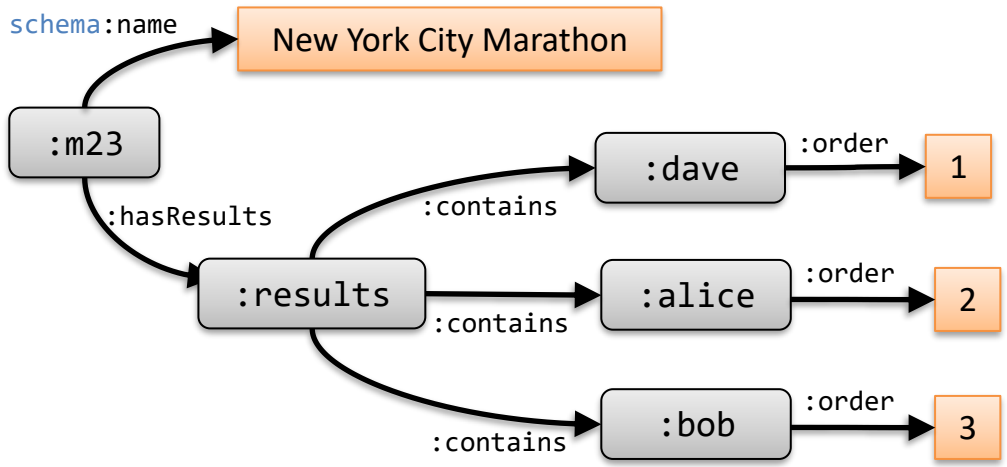
Pros: Acceso directo a cada elemento

Cons: No es fácil detectar la estructura de la lista (longitud de la lista, valores que faltan, ...)
Difícil insertar/borrar elementos

Solución 3
Representar orden con valores anotados

Anotar los elementos con un valor que indique el orden

```
:m23 schema:name "New York City Marathon ";
     :hasResults :results .
:results :contains :dave, :alice, :bob.
:dave   :order 1 .
:alice  :order 2 .
:bob    :order 3 .
```



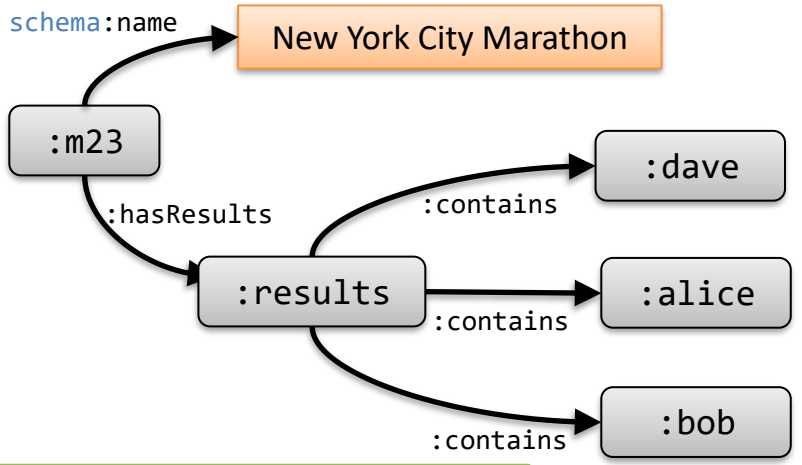
Pros: Es posible acceso directo a cada elemento, longitud de lista disponible
Cons: Es posible crear inconsistencias (ej. elementos con el mismo orden)
Más difícil insertar/borrar elementos

Solución 4
Ignorar el orden

A veces el orden realmente no es necesario

Abandonar el orden y represetar listas como conjuntos

```
:m23 schema:name "New York City Marathon ";
     :hasResults :results .
:results :contains :dave, :alice, :bob.
```



Pros: Fácil de hacer en RDF, puede ser suficiente en muchos casos
Cons: No hay orden

Solución 5
Combinar varias técnicas

RDF es muy versátil

Es posible combinar varias técnicas

```
:m23 schema:name "New York City Marathon ";
     :hasResults :results .
:results :contains :dave, :alice, :bob.
         :order    ( :dave, :alice, :bob ) .
```

Pros: Puede ofrecer las ventajas de las diferentes técnicas
Cons: Volumen de datos mayor, redundancias y posibles inconsistencias

Datos tabulares

Ejemplo

Curso

CID	Código	Título	Clase	Profesor
23	CS101	Programming	A1	144
34	A102	Algebra	B2	144

Profesor

PID	Nombre	Apellidos
144	Alice	Cooper

Cada tabla puede verse como una relación de aridad-N

Posible representación

```
prefix : <http://example.org/>

:23 a          :Course ;
    :code      "cs101" ;
    :title     "Programming"@en ;
    :room      "A1" ;
    :teacher   :144 .
:34 a          :Course ;
    :code      "A102" ;
    :title     "Algebra"@en .
    :room      "B2" ;
    :teacher   :144 .

:144 a            :Teacher ;
     :firstName   "Alice" ;
     :lastName    "Cooper" .
```

RDB2RDF: A Direct Mapping of Relational Data to RDF.
https://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/

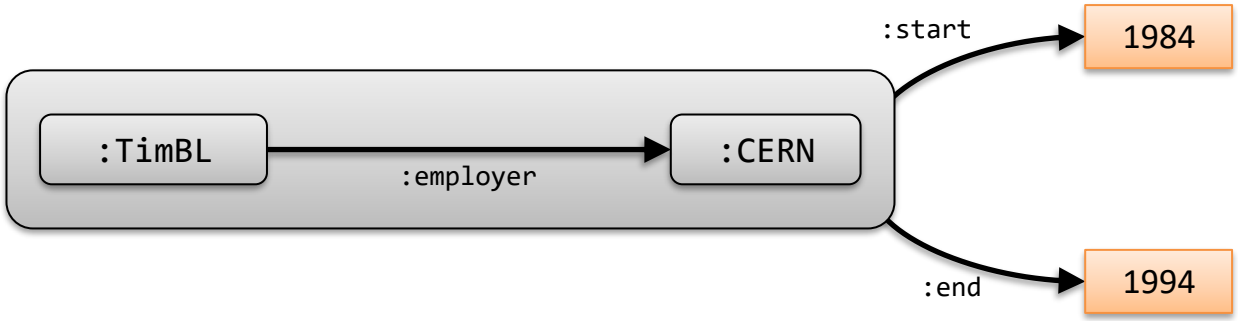
Reificación

Reificación: añadir enunciados sobre enunciados

Ejemplo: *Tim Berners-Lee trabajó en el CERN* *(entre 1984 y 1994)*

Varias técnicas

- Reificación estándar de RDF
- Relaciones de aridad N
- RDF-*
- Grafos con nombre



Técnica 1 para reificación
Reificación estándard de RDF

Ya se introdujo en RDF 1.0

Predicados `rdf:subject, rdf:predicate, rdf:subject`

Clase `rdf:Statement`

```
:s1 a rdf:Statement ;
    rdf:subject   :TimBl ;
    rdf:predicate :employer ;
    rdf:object    :CERN ;
    :start        "1984"^^xsd:gYear ;
    :end          "1994"^^xsd:gYear .
```

Pros: Es parte de RDF, desde RDF 1.0
Cons: No es fácil de gestionar y no es muy flexible.
No es compatible con OWL DL

Técnica 2 para reificación
Enunciados como relaciones de aridad-N

Crear un nodo auxiliar para representar un enunciado

Añadir propiedades para relacionar los nodos con el nodo auxiliar

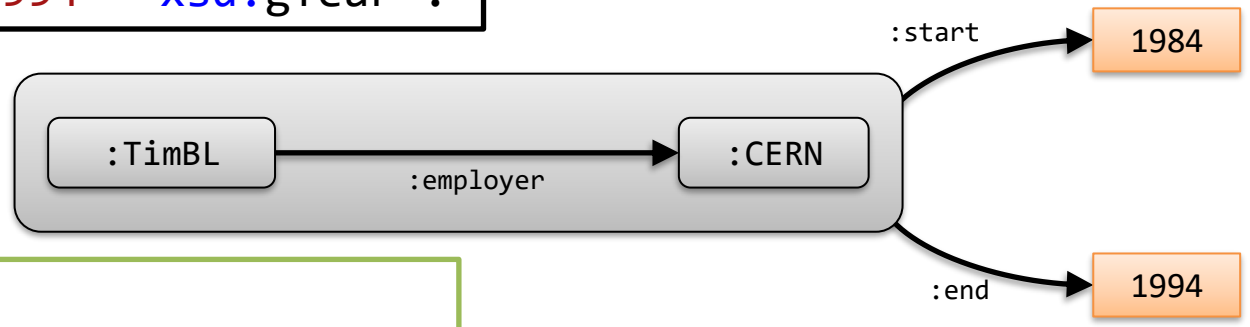
```
:timBl :employer :e .

:e :organization :CERN ;
   :start        "1984"^^xsd:gYear ;
   :end          "1994"^^xsd:gYear .
```

Pros: Puede expresarse directamente en RDF
Cons: Requiere la creación de nodos y propiedades auxiliares

Técnica 3 para reificación
RDF-*

RDF-* = Extensión de RDF en la que los grafos pueden ser sujetos u objetos de un enunciado

```
<< :timBl :employer :CERN >> :start "1984"^^xsd:gYear ;
                              :end   "1994"^^xsd:gYear .
```



Pros: Expresa la reificación directamente

Cons: Todavía no está siendo ampliamente adoptada.
Podría requerir herramientas para convertir a RDF

RDF-* Community group specification: https://w3c.github.io/rdf-star/cg-spec
RDF-* Working group: https://www.w3.org/groups/wg/rdf-star

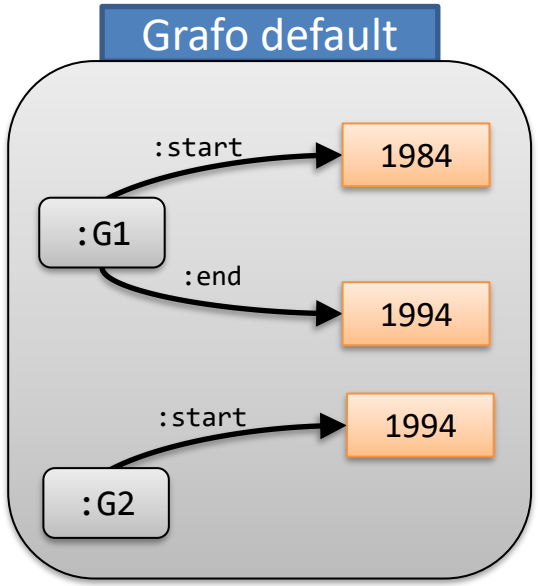
Técnica 4 para reificación
Grafos con nombre

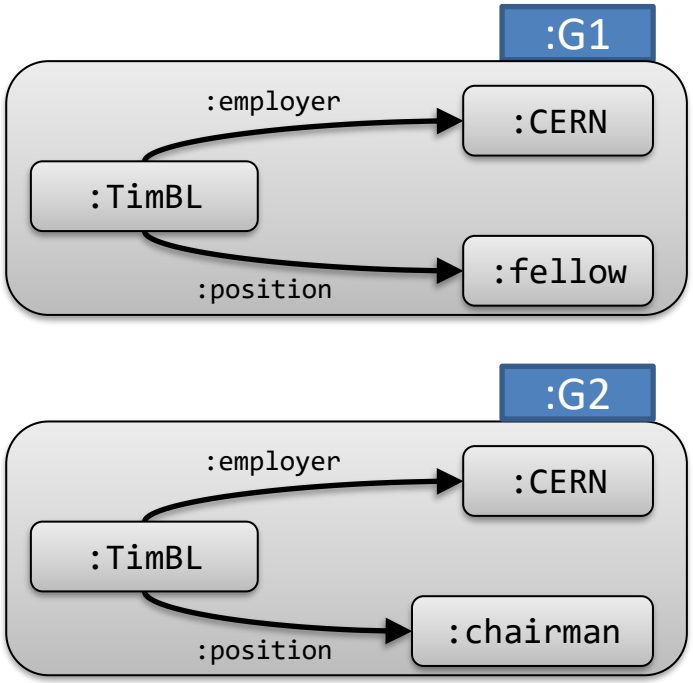
RDF datasets = colecciones de grafos RDF (se soportan en SPARQL)

- Un grafo por defecto
- Cero o más grafos con nombre (nombre = IRI/Blank node)

TRIG = Extensión de Turtle que puede expresar *RDF datasets*

```
:G1 :start "1984"^^xsd:gYear .
:G1 :end   "1994"^^xsd:gYear .
:G2 :start "1994"^^xsd:gYear .

:G1 {
  :timBl :employer :CERN ;
         :position :fellow
}
:G2 {
  :timBl :employer :W3C .
         :position :chairman
}
```





Algunas referencias

Linked data patterns

https://patterns.dataincubator.org/

Ontology design patterns

http://ontologydesignpatterns.org/

Working ontologist book

http://workingontologist.org/

Best Practices for Publishing Linked Data.

https://www.w3.org/TR/ld-bp/

Data on the Web Best Practices

https://www.w3.org/TR/dwbp/

Fin de la Presentación