

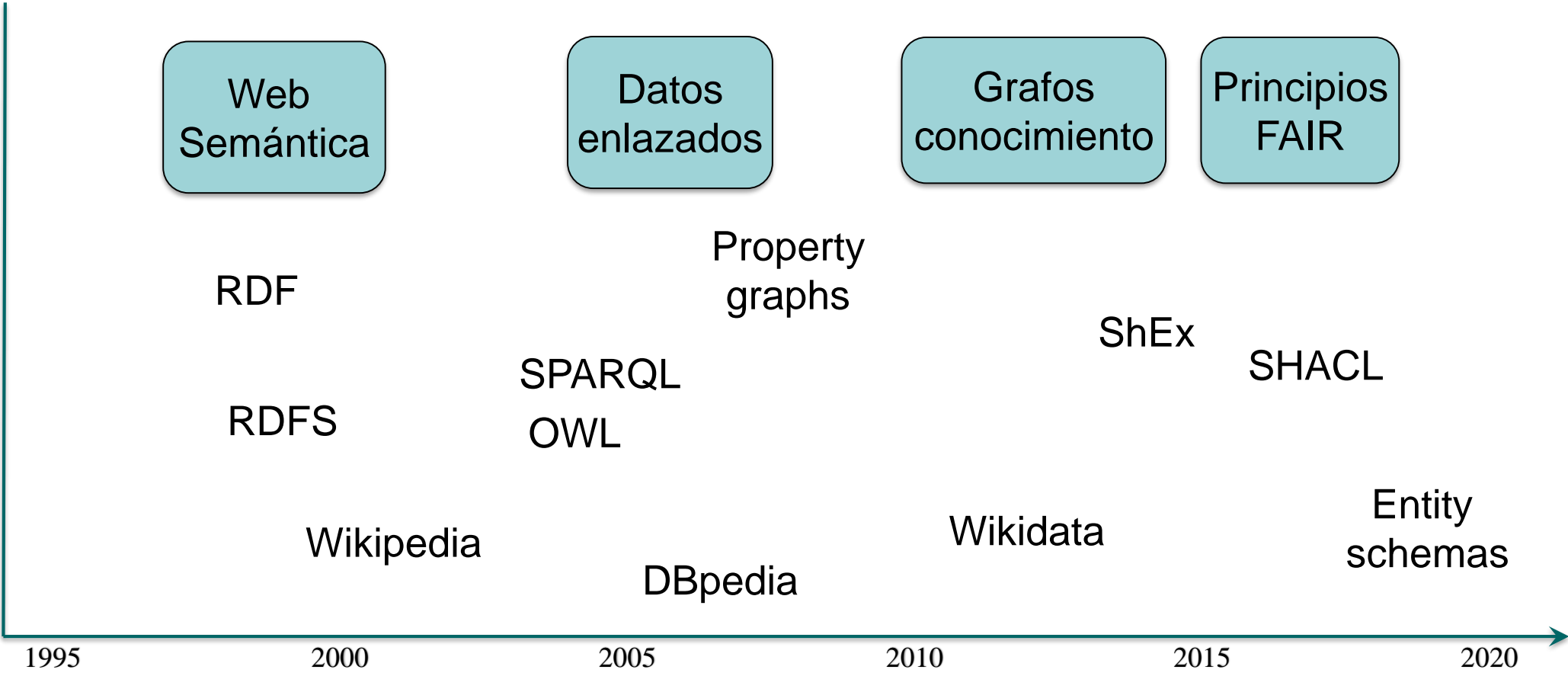
Introducción a RDF

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

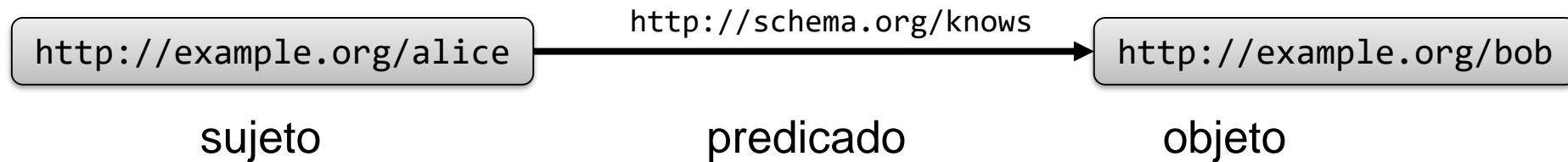
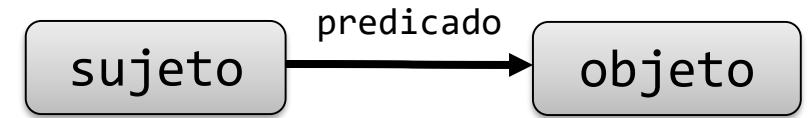
1999 1st W3C Rec <https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>, XML Syntax, first applications RSS, EARL

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

Modelo de datos RDF

RDF está formado por enunciados (statement)
Enunciado = tripleta (sujeto, predicado, objeto)
Ejemplo:



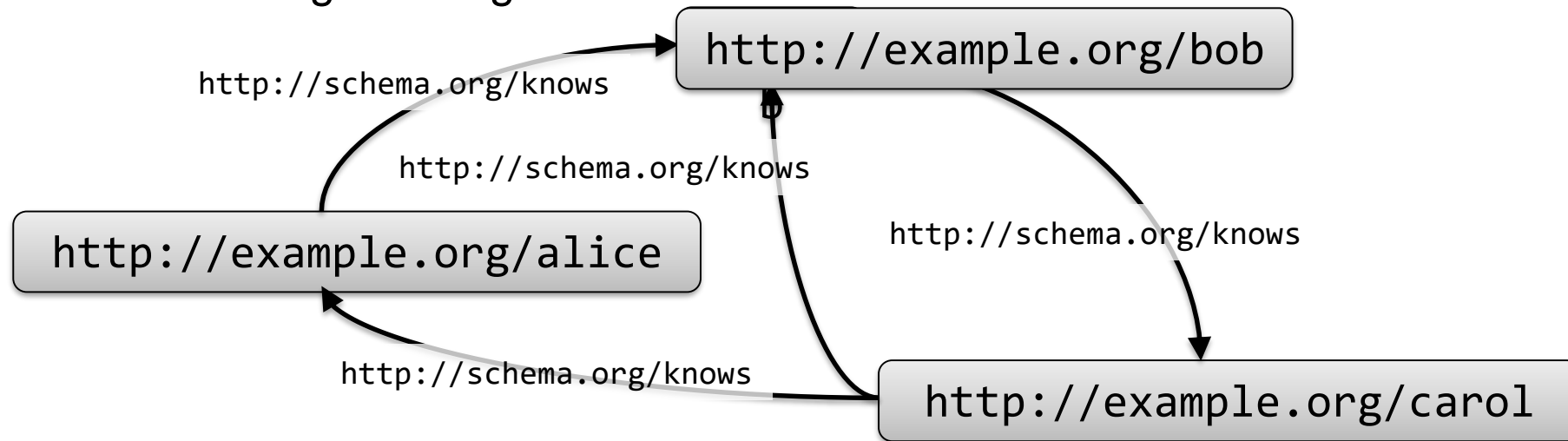
Representación N-Triples

```
<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

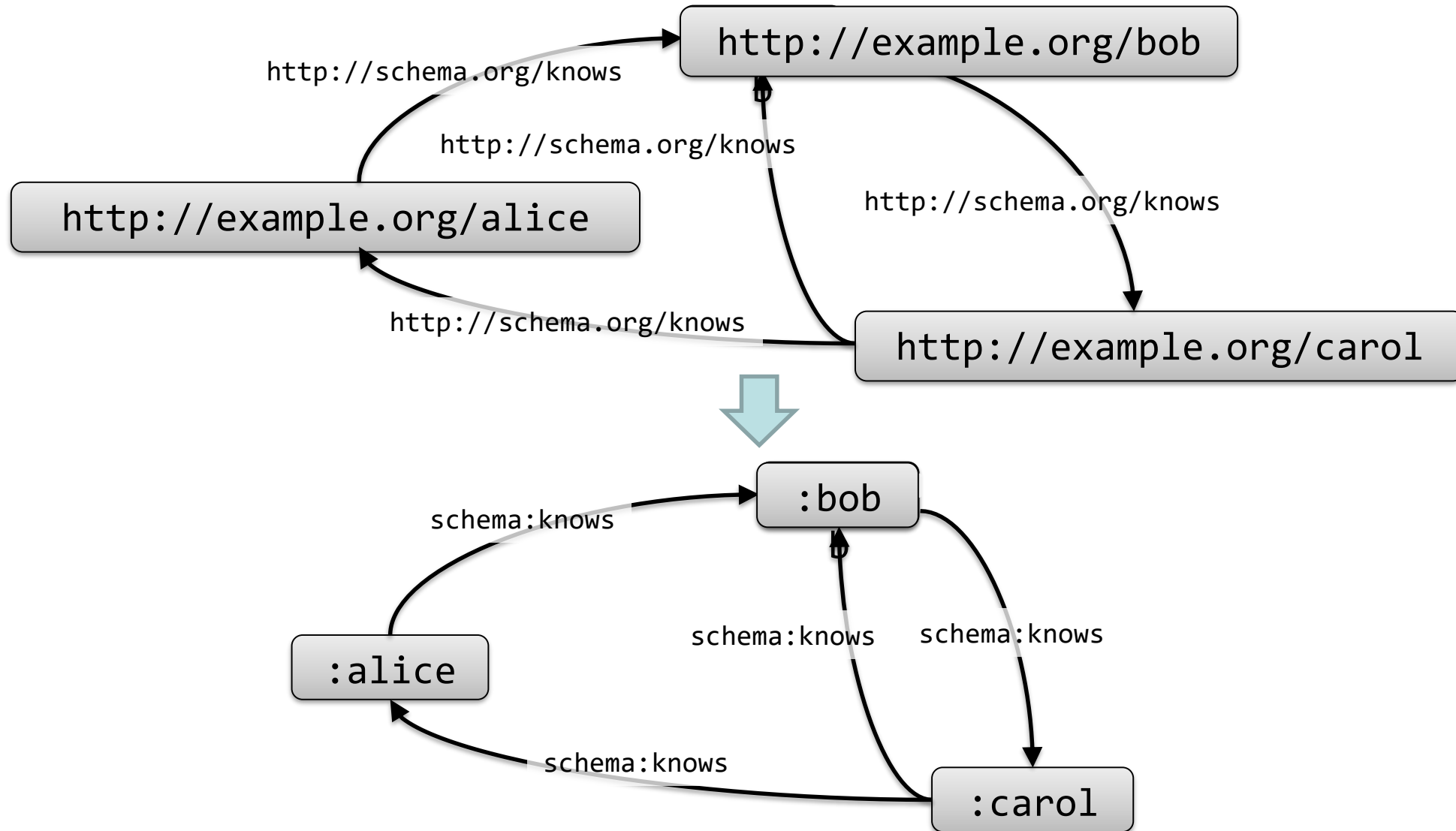


```
prefix : <http://example.org/>  
prefix schema: <http://schema.org/>  
  
:alice schema:knows :bob .  
:bob schema:knows :carol .  
:carol schema:knows :bob .  
:carol schema:knows :alice .
```

Nota:

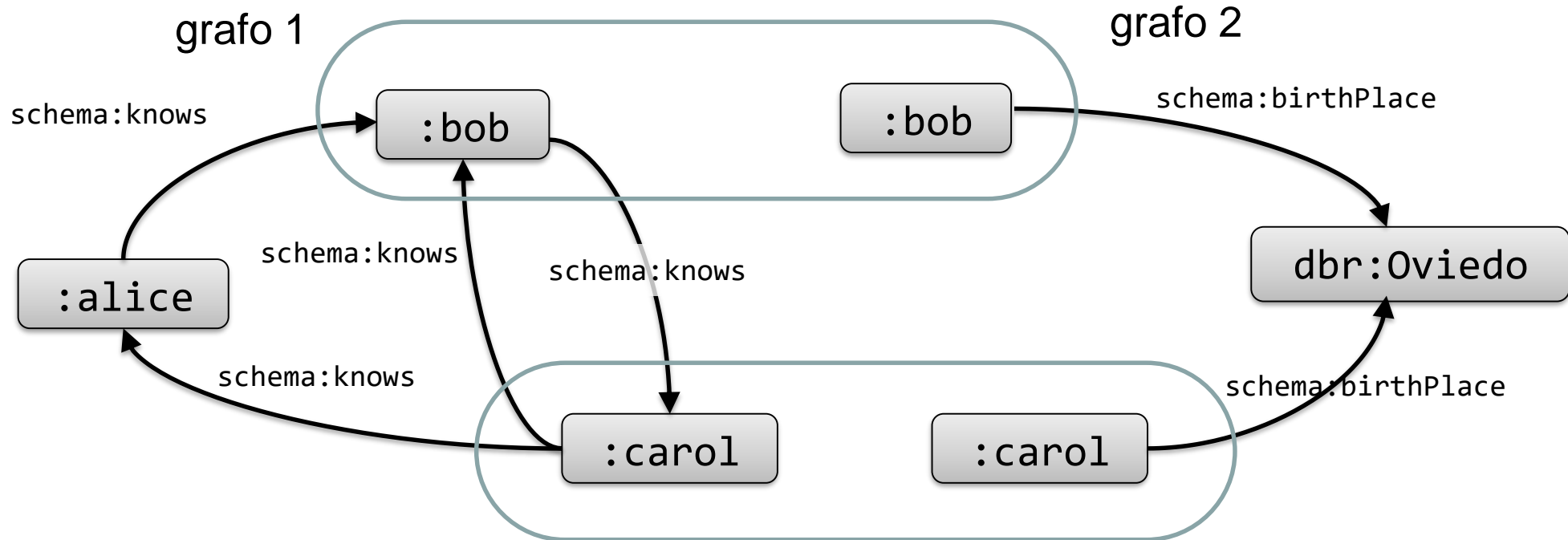
Más adelante veremos más simplificaciones

Simplificaciones de espacios de nombres



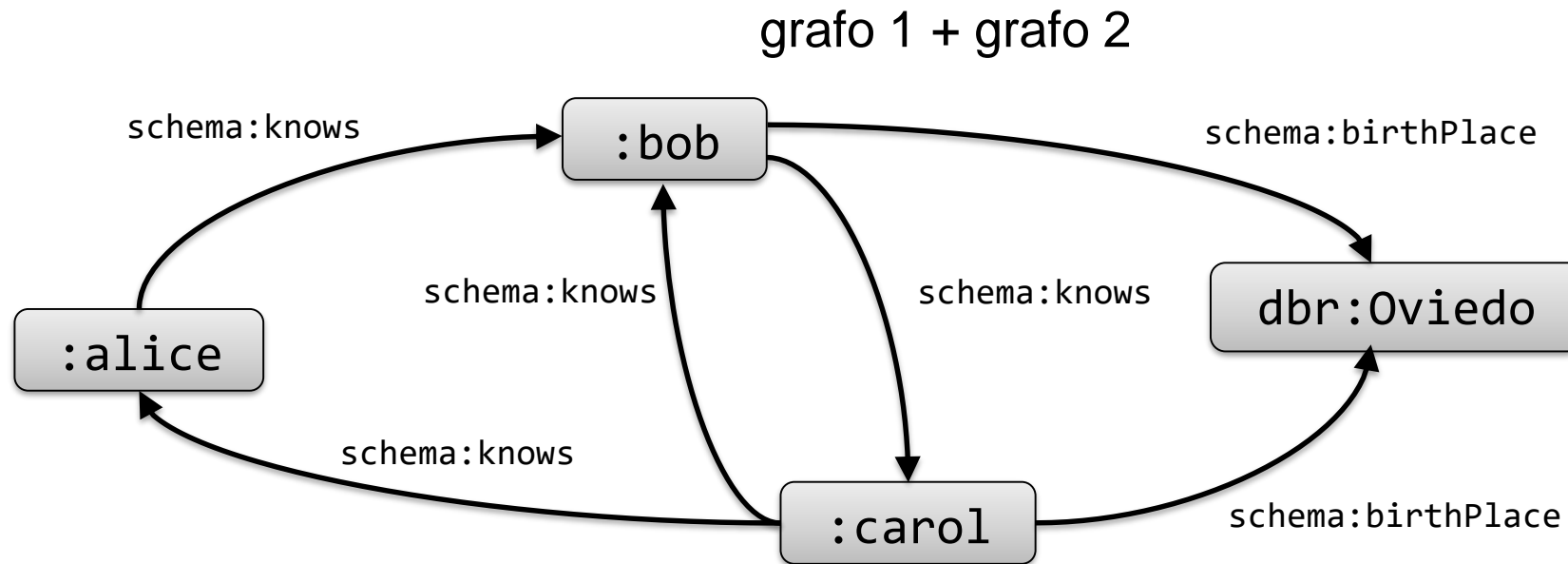
Propiedad: RDF es composicional

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 .
```



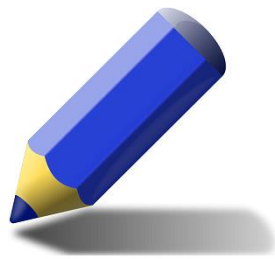
```
:alice schema:birthPlace dbr:Oviedo ;  
      schema:knows      :bob .
```

, 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>

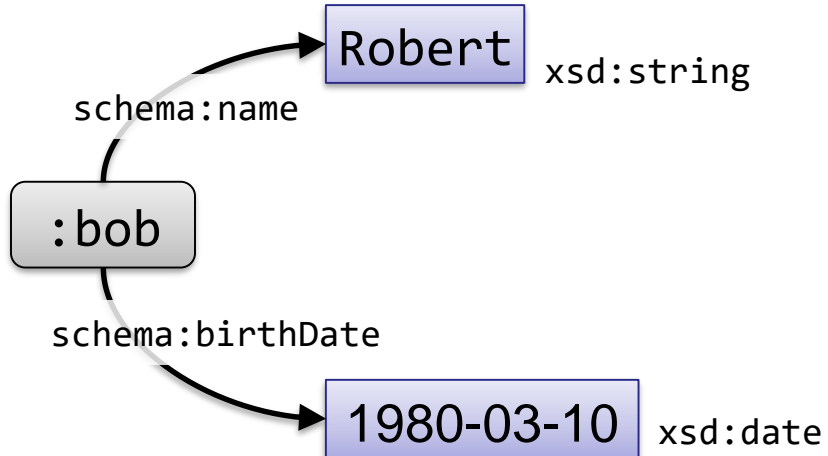
Literales RDF

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`

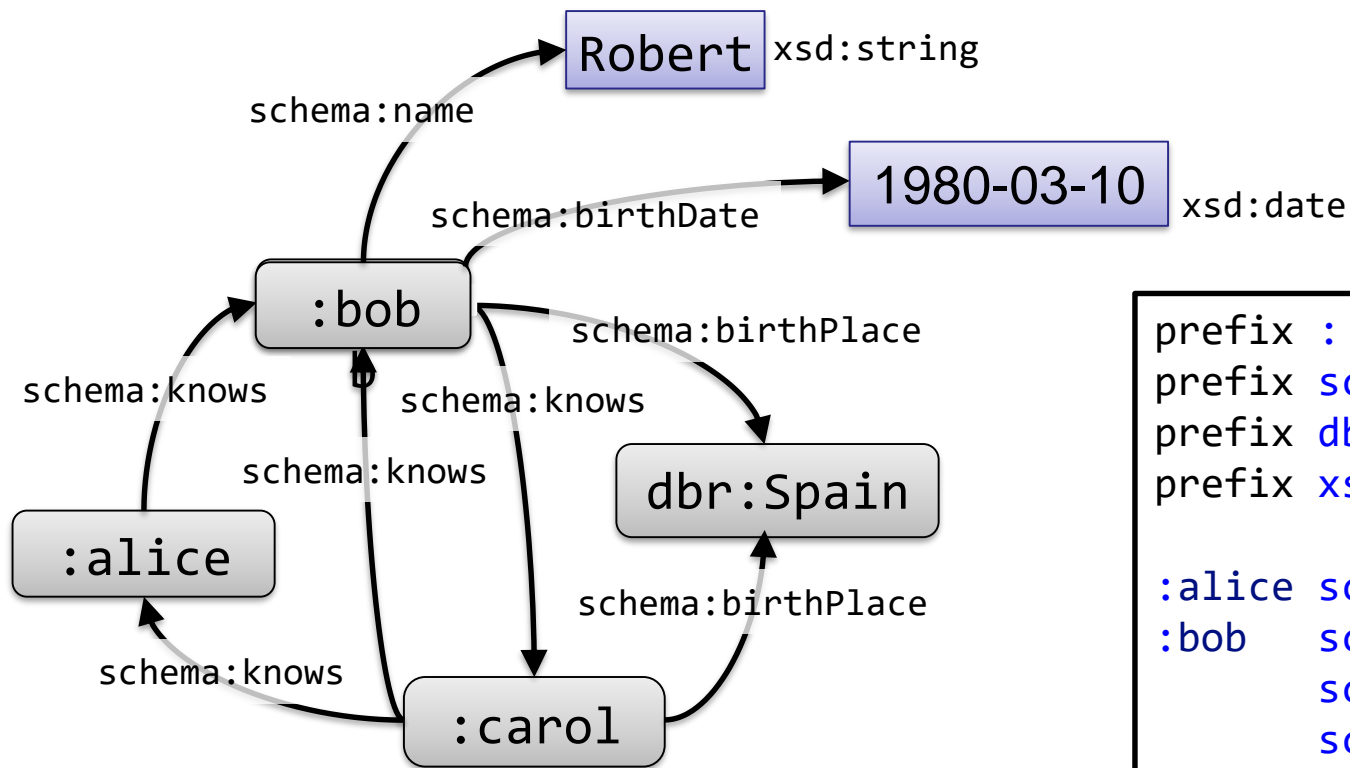


Turtle notation

```
: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"

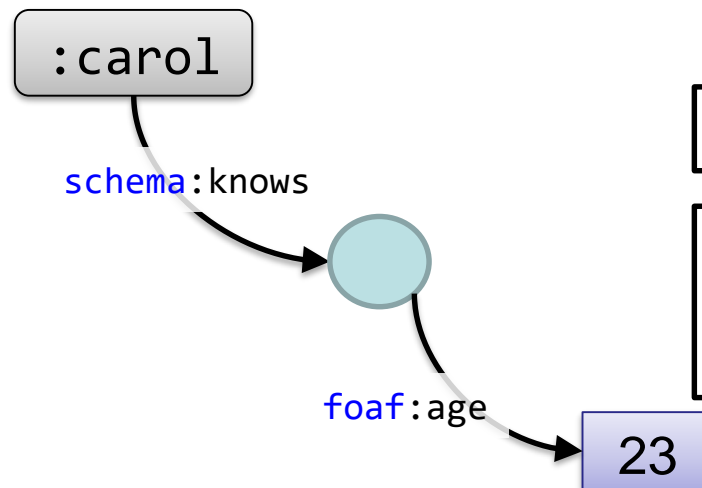
Notación Turtle con identificador local

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

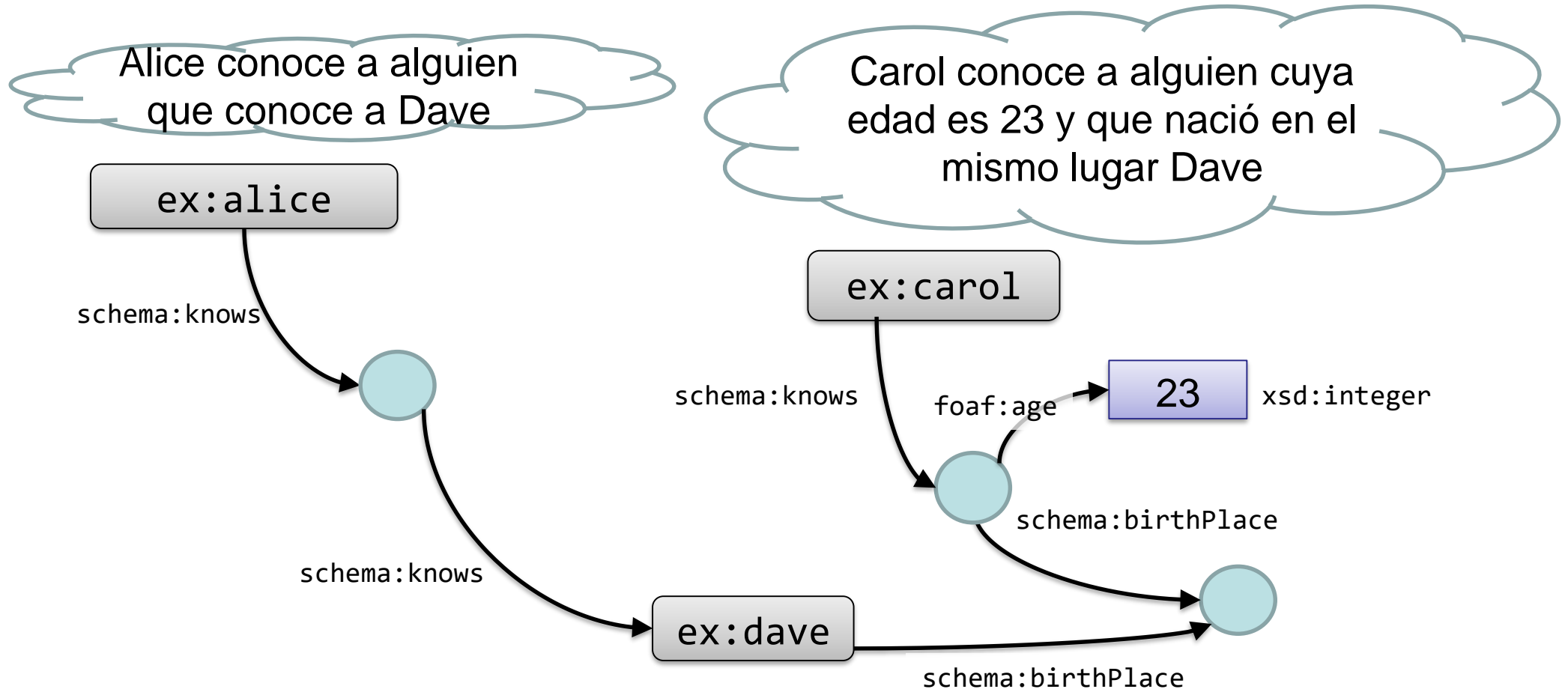
Notación Turtle con corchetes

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

Significado matemático:

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


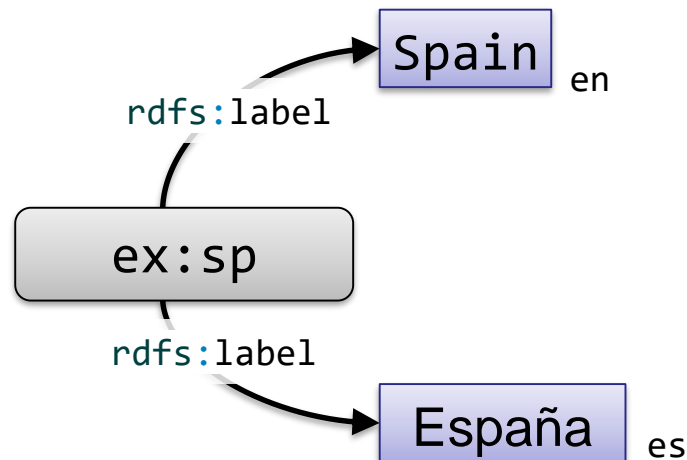
Nodos anónimos



```
: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`

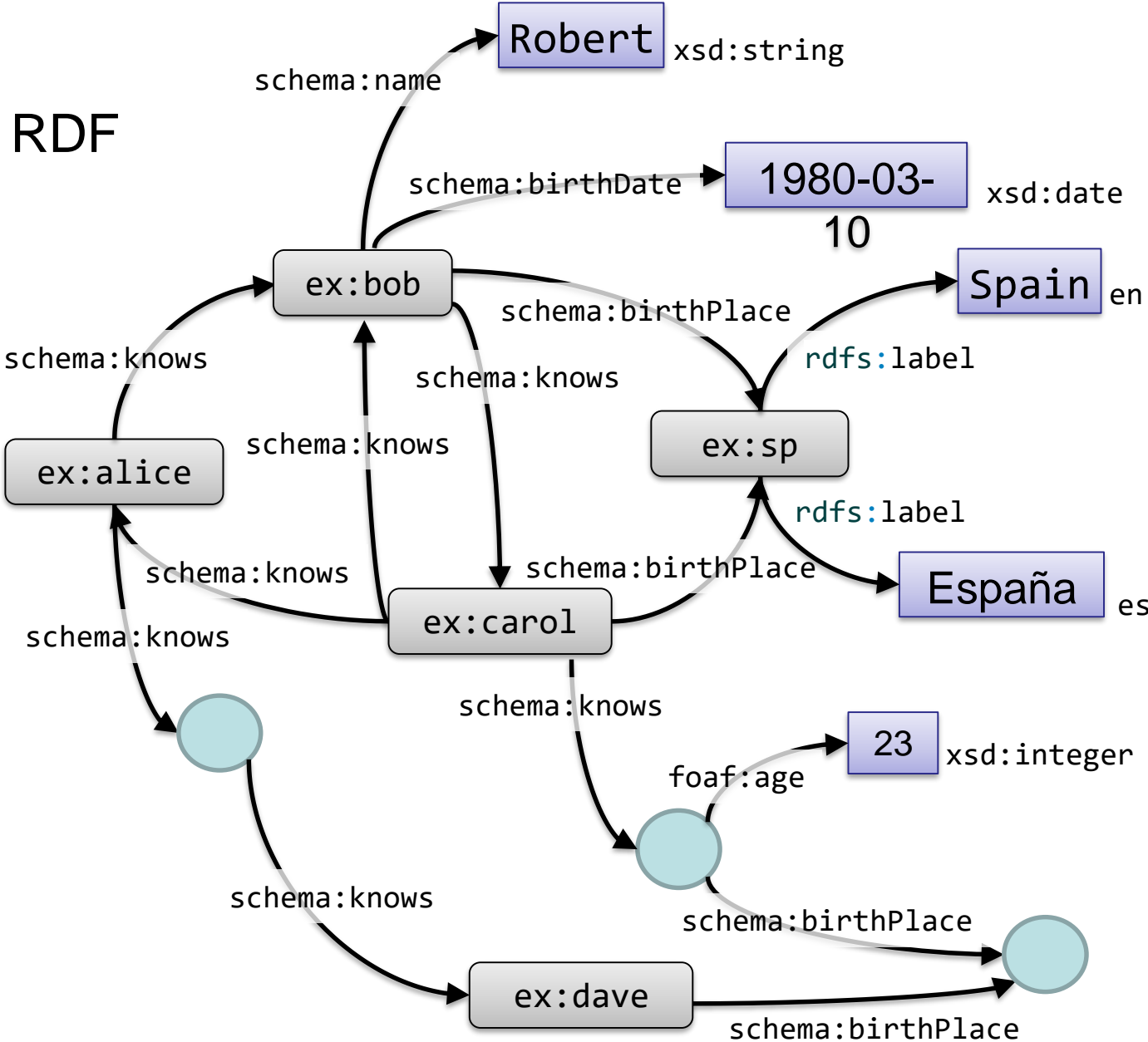


Notación Turtle




```
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

$$\mathcal{S} = \mathcal{I} \cup \mathcal{B},$$

$$\mathcal{P} = \mathcal{I},$$

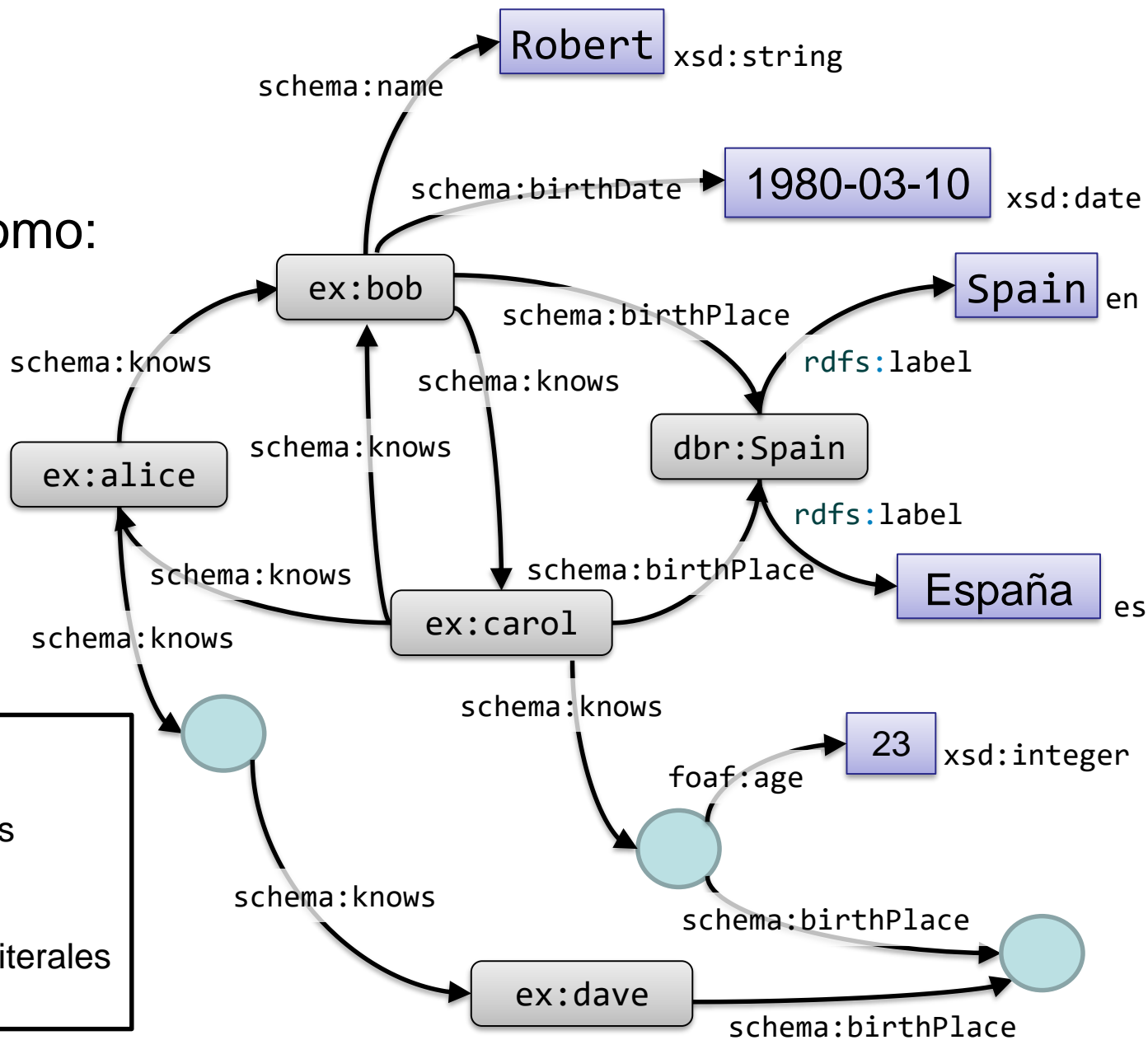
$$\mathcal{O} = \mathcal{I} \cup \mathcal{B} \cup Lit$$

$$\rho \subseteq \mathcal{S} \times \mathcal{P} \times \mathcal{O}$$

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



Simple
is
better

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

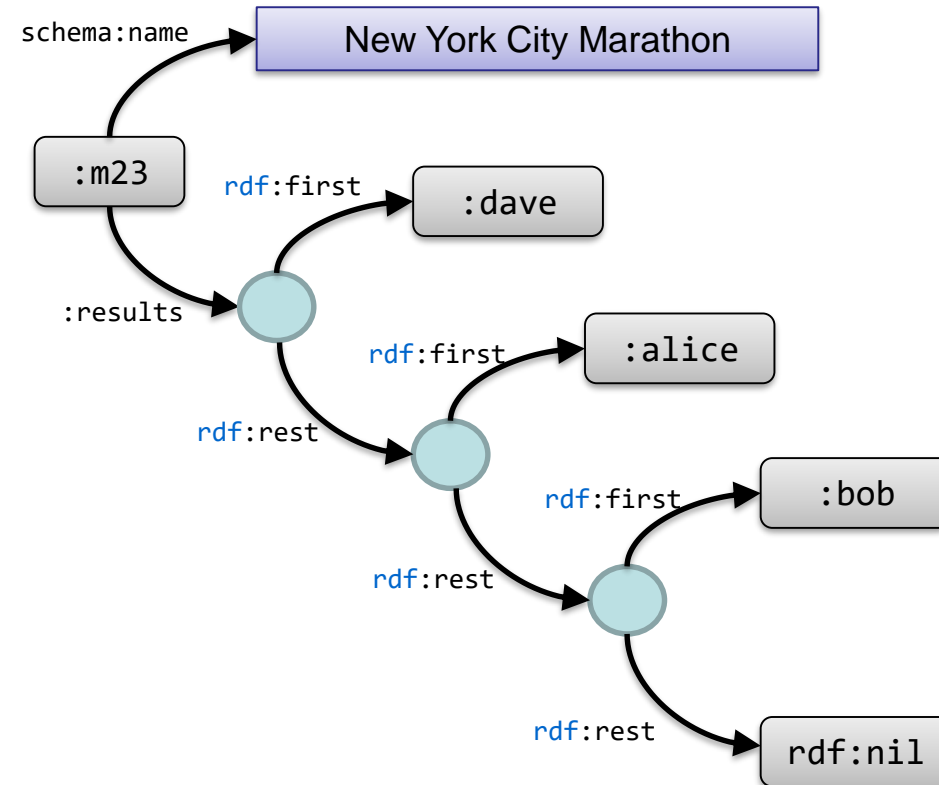
Colecciones

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

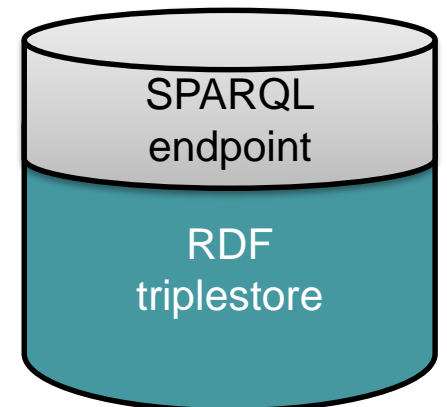
SPARQL 1.0 (2008), 1.1 (2013)

Syntaxis inspirada por Turtle

Semántica basada en patrones de grafos

SPARQL endpoints = servicios que implementan el protocolo SPARQL

Triplestores = Bases de datos RDF



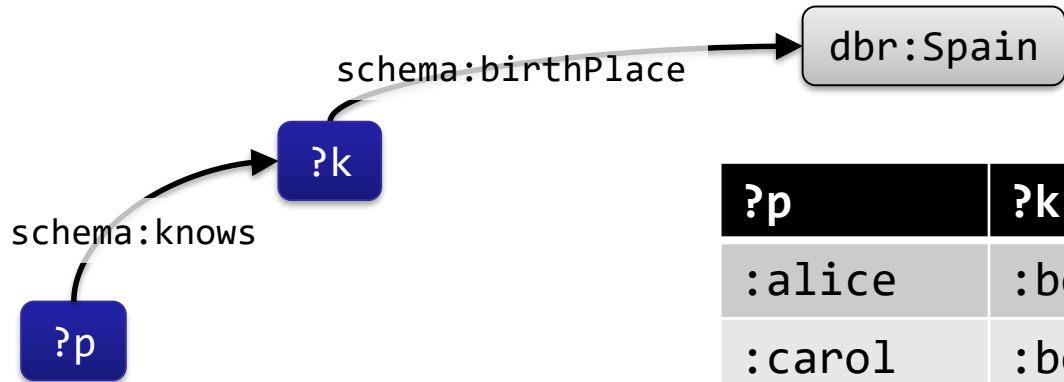
Ejemplo SPARQL

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

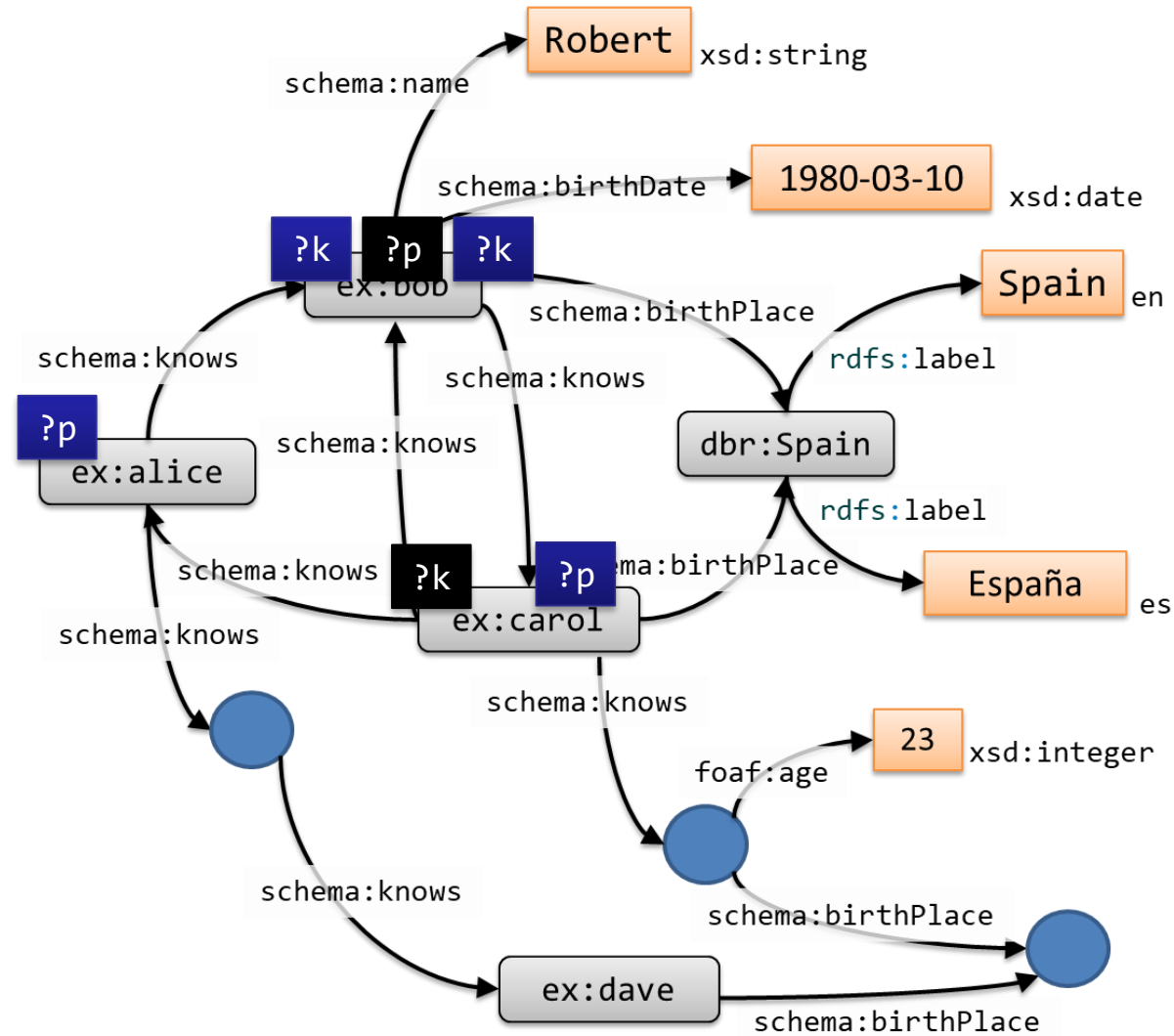
```

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

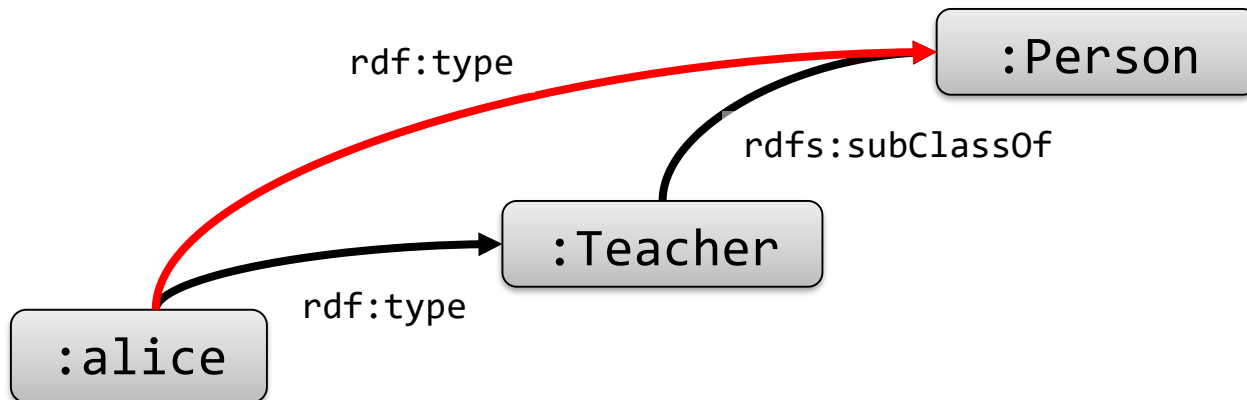
El poder de los vocabularios compartidos

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)

$Person \sqsubseteq = 2 \text{ hasParent}$
 $Person \sqsubseteq \exists \text{ hasParent Male}$
 $Person \sqsubseteq \exists \text{ hasParent Female}$

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

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

```
: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

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

Datos Inferidos

$Male(bob)$

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

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



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

Necesitamos declarar que $:bob \neq :carol$

```
[ 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?

The screenshot displays the Protégé OWL editor interface. The main window shows the class hierarchy for the ontology `http://example.org`. The class `:Person` is selected, and its hierarchy is shown as follows:

- `owl:Thing`
 - `:Female`
 - `:Male`
 - `:Person`

The right-hand pane shows the class axioms for `:Person`. The axioms are:

- Equivalent To
- SubClass Of
 - `:hasParent exactly 2 owl:Thing`
 - `:hasParent some :Female`
 - `:hasParent some :Male`
- General class axioms
- SubClass Of (Anonymous Ancestor)
- Instances
 - `:alice`
 - `:bob`
 - `:carol`

The interface includes a menu bar (File, Edit, View, Reasoner, Tools, Refactor, Window, Help) and a toolbar with navigation and editing icons. The status bar at the bottom indicates "Git: main" and "Reasoner active".

Fin de la Presentación

