Modelado de datos en RDF

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain

Representando información en RDF

RDF = modelo de datos para intercambiar información en Web

Algunas consideraones & compromisos

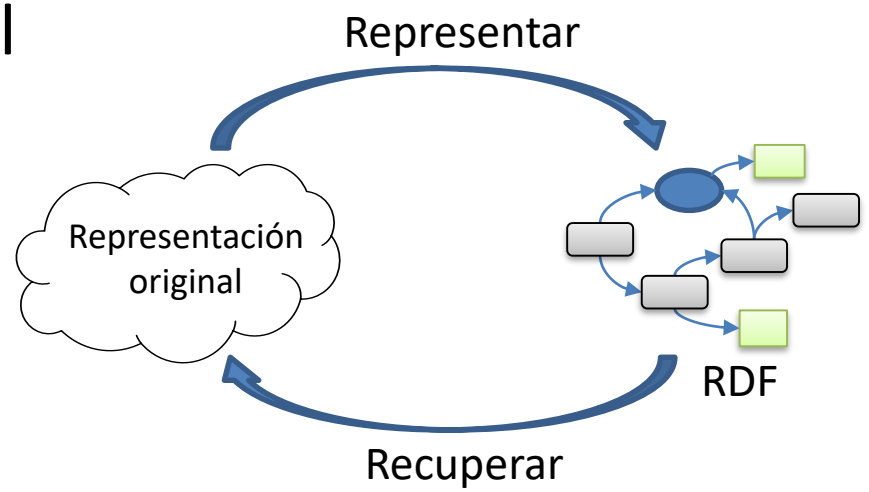
- Precisión semántica (*Semantic accuracy*)
- Legibilidad humana
- Flexibilidad y falta de esquema
- Interoperabilidad & rendimiento

Precisión semántica (*semantic accuracy*)

Evitar pérdidas semánticas

Ida y vuelta (*Round-tripping*)

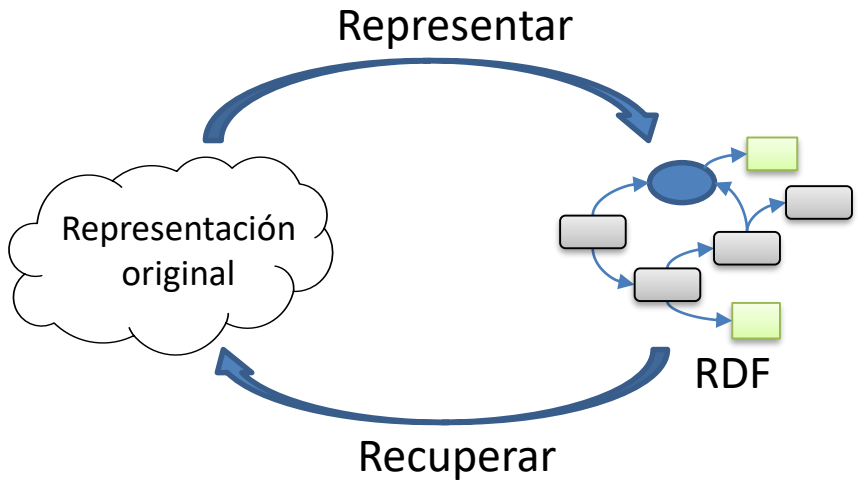
- De la represetación original a RDF
- De RDF recuperar representación original



Precisión semántica

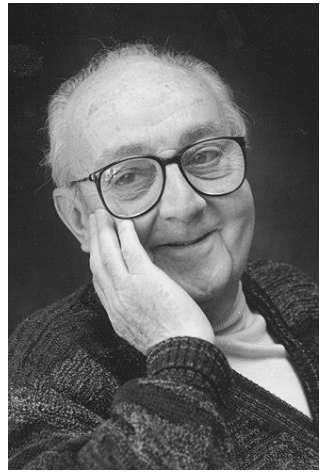
Debemos tener cuidado sobre

Relación mapa-territorio



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

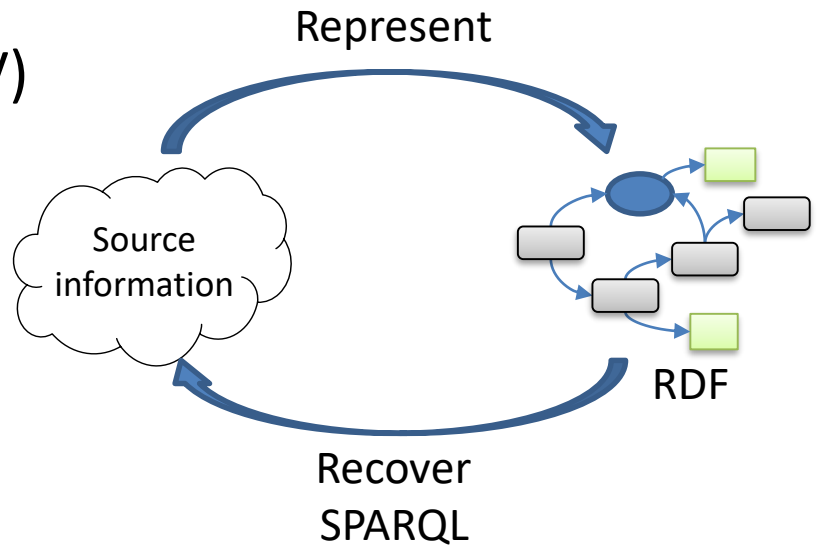
G. Box aphorism



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 vocabularies 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

WESO

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`

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

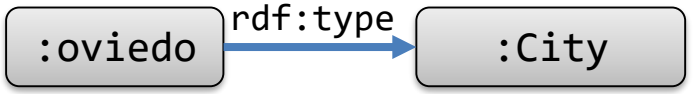
Reificación y procedencia

Agrupando tripletas RDF y conjuntos de datos

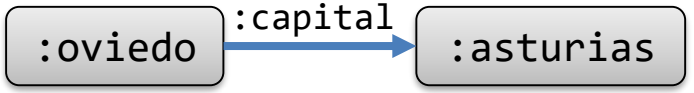
Relaciones de aridad N

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

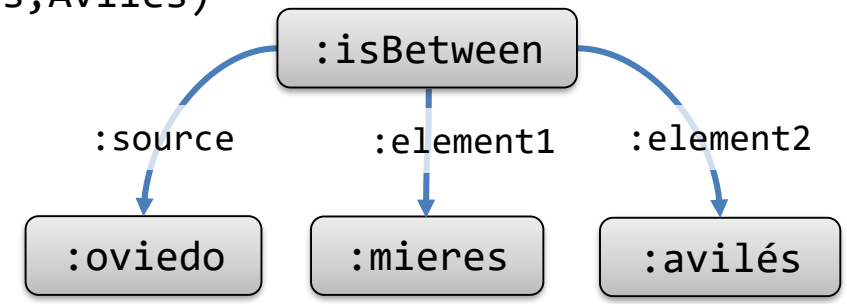
Aridad 1: *Oviedo es una ciudad* `city(Oviedo)`



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

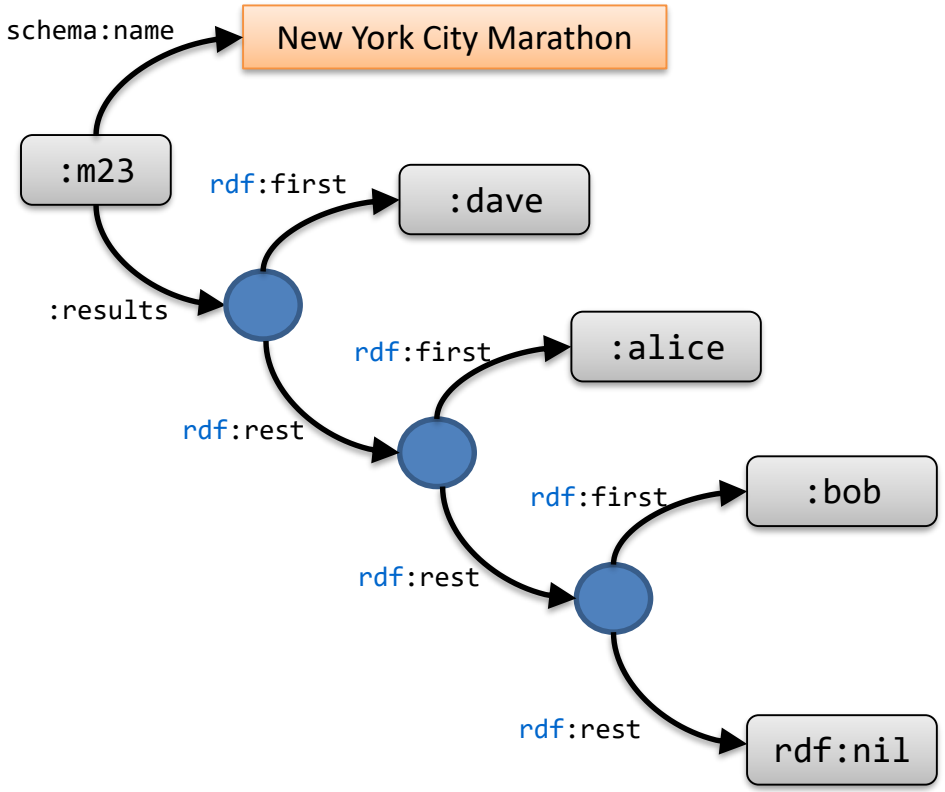
Solución 1
Representar orden con listas enlazadas

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 .
```



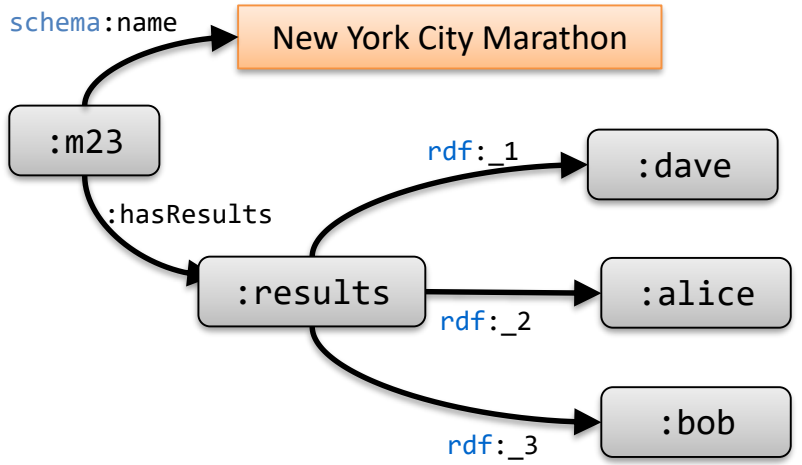
Pros: Representación elegante, fácil para insertar/borrar, marcar fin de lista
Cons: Acceso ineficiente a un elemento dado

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

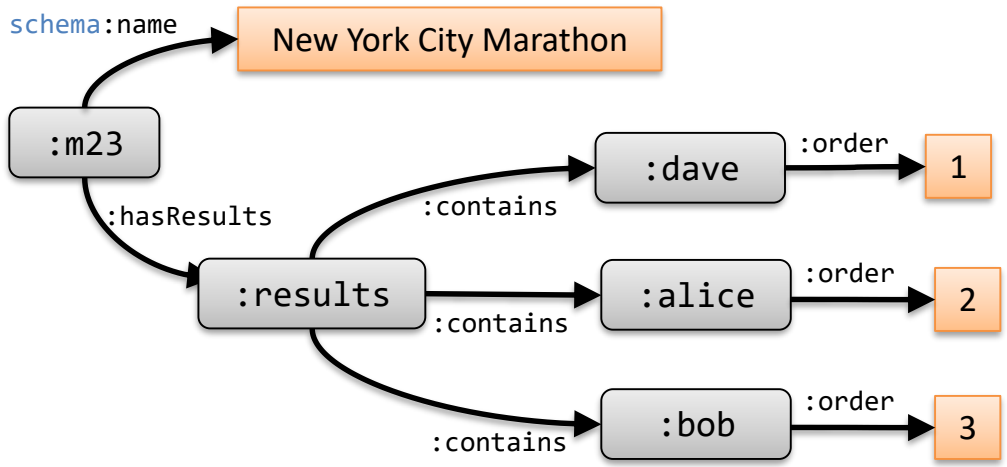
Cons: No es fácil detectar la estructura de la lista (longitud de la lista, valores que faltan, ...)

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 .
```



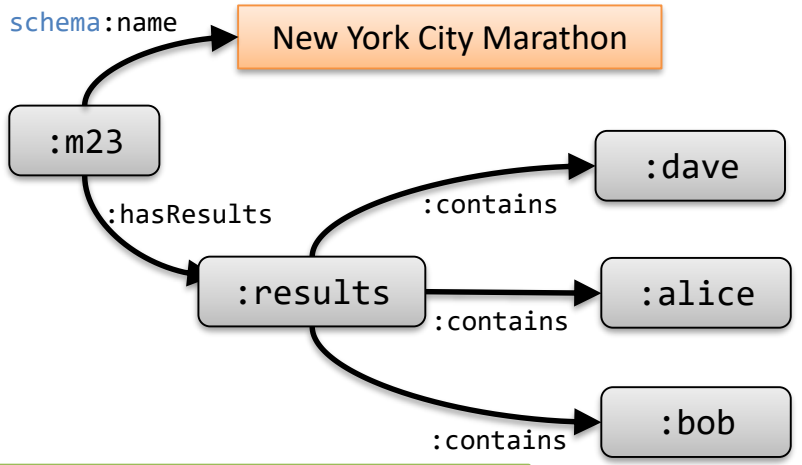
Pros: Es posible acceso directo a cada elemento, longitud de lista disponible
Cons: Es posible crear inconsistencias (ej. elementos con el mismo orden)
Más difícil insertar/borrar elementos

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/

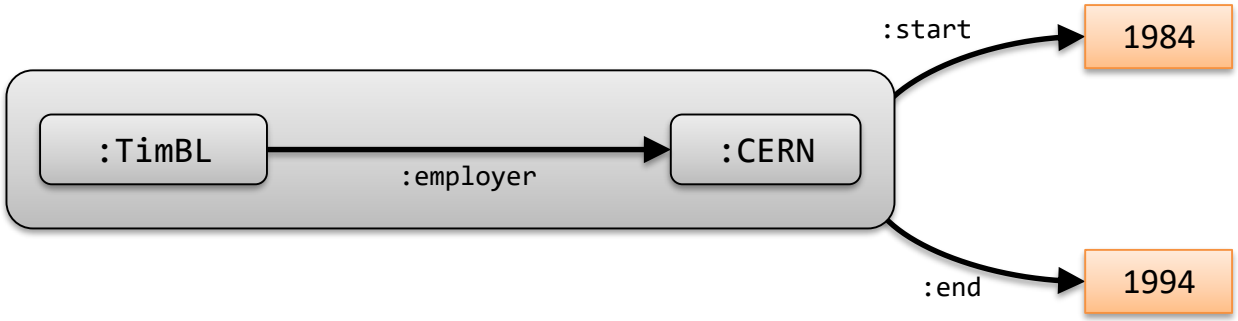
Reificación

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

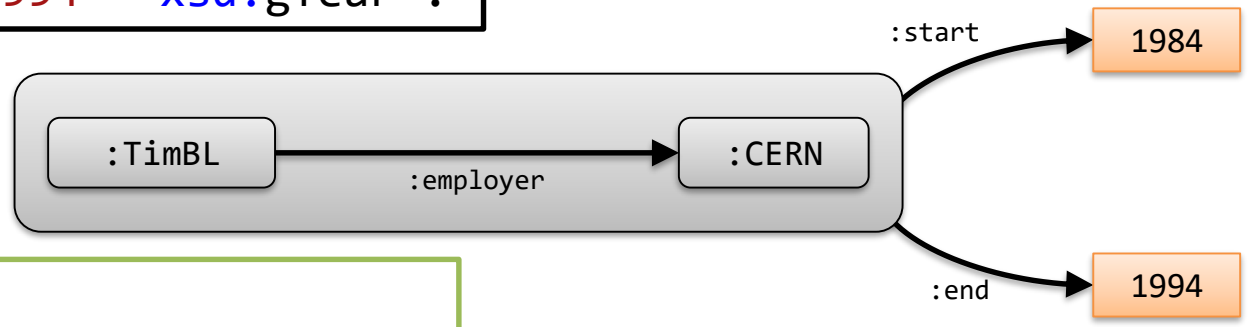
```
:timBl :employer :e .

: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

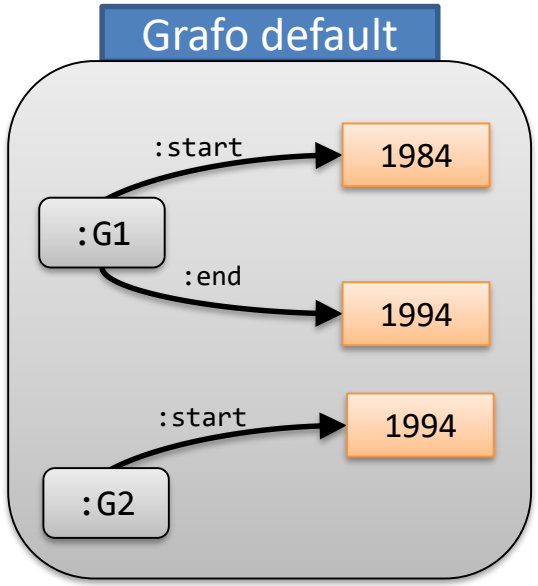
Técnica 4 para reificación
Grafos con nombre

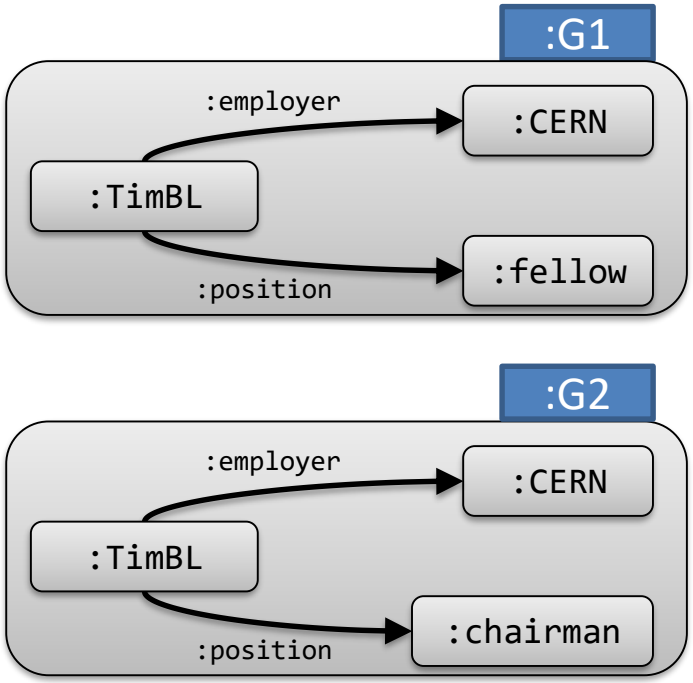
RDF datasets = colecciones de grafos RDF (se soportan en SPARQL)

- 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/