Dados Ligados na Web

Veruska Zamborlini (veruska.zamborlini@ufes.br)

Professora do Departamento de Informática

Núcleo de Estudos em Modelagem Conceitual e Ontologias - NEMO

Departamento de Informática

Centro Tecnológico

Universidade Federal do Espírito Santo

Agenda

1. Introdução
2. História e Fundamentos básicos
   a. Mãos nos dados

      PAUSA
3. Fundamentos RDF, RDFS
   a. Mãos nos dados

      PAUSA
4. Fundamentos SPARQL
   a. Mãos nos dados

      PAUSA
5. Discussão
6. Extras

NEMO - Modelagem Conceitual e Ontologias


Camila


João Paulo


Monalessa


Ricardo (in memoriam)


Veruska


Vítor


Giancarlo


Renata

MEMBROS SÊNIOR

- ➔ 18 anos de existência
- ➔ +600 publicações nacionais e internacionais
- ➔ +70 alunos formados (graduação e pós)
- ➔ +20 orientações em andamento

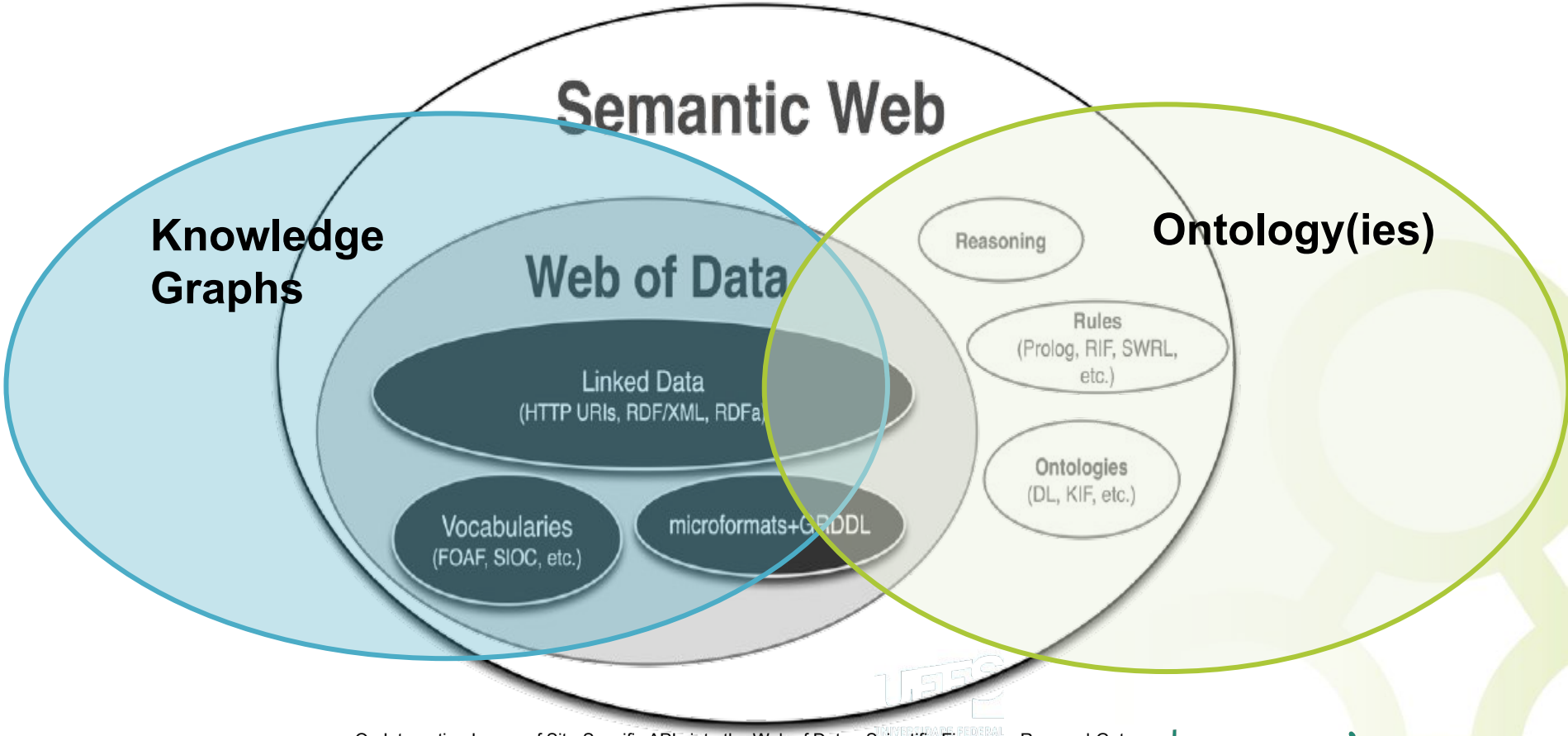
MEMBROS EXTERNOS

https://nemo.inf.ufes.br

Introdução
Dados Ligados na Web

Quem já ouviu falar de … ?

- Dados Ligados (*Linked Data*)
- Web Semântica
- Ontologia(s)
- Grafos de conhecimento (*Knowledge Graph*)



On Integration Issues of Site-Specific APIs into the Web of Data - Scientific Figure on ResearchGate.
https://www.researchgate.net/figure/The-Web-of-Data-structured-and-interlinked-data-in-RDF_fig1_242700809

Quem já ouviu falar de … ?

- *SPARQL* - acrônimo recursivo para

  SPARQL Protocol and RDF Query Language

  (mais que) linguagem de consulta para BD de "grafo"

- *SQL* - acrônimo para

  Structured Query Language

  (mais que) linguagem de consulta para BD relacionais

* Employee

EmpId	EmpName	DeptId
101	Jane Doe	17
102	Graham Kerr	40
103	Sydney Green	17
104	Jason Morgan	40

* Department

DeptId	DeptName	DeptMngrId
17	IT	103
40	Accounting	104

```
select EmpName,DeptName
from  Employee join Department
on Employee.DeptId = Department.DeptId
```

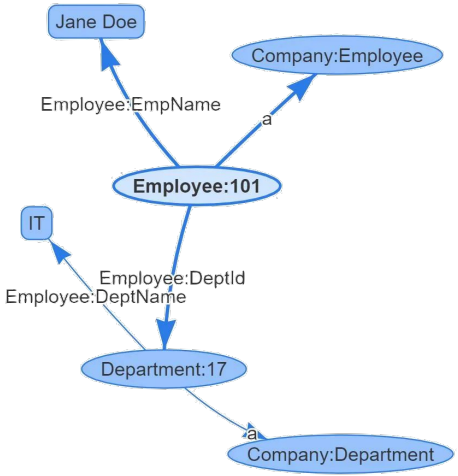
```
select EmpName,DeptName
from  Employee, Department
where Employee.DeptId = Department.DeptId
```

SQL

SPARQL

* Results

empName	deptName
"Jane Doe"	"IT"
"Graham Kerr"	"Accounting"
"Sydney Green"	"IT"
"Jason Morgan"	"Accounting"



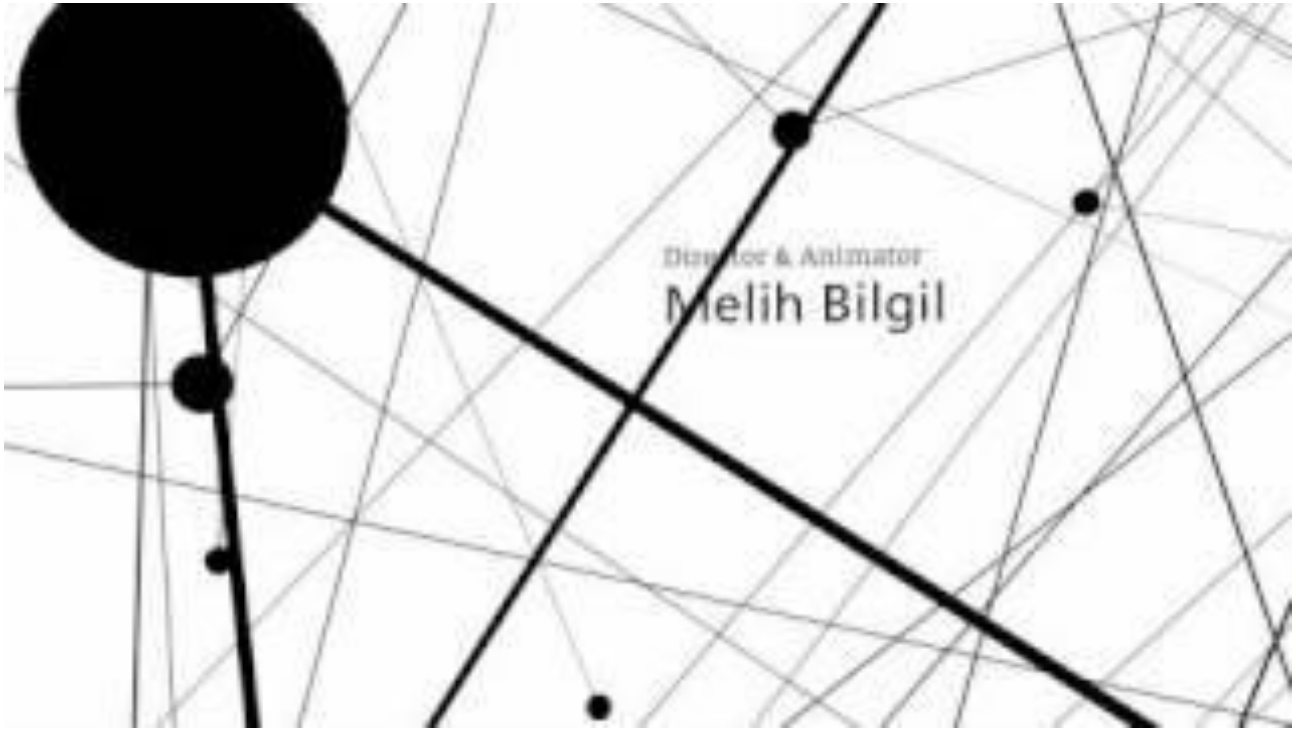

```
select ?empName ?deptName
from graph:Company
where {
 ?employee a Company:Employee.
 ?department a Company:Department.
 ?employee Employee:deptId ?department.
 ?employee Employee:dmpName ?empName.
 ?department Department:deptName ?deptName.
}
```

https://medium.com/metaphorical-web/from-sql-tables-to-sparql-graphs-4a...

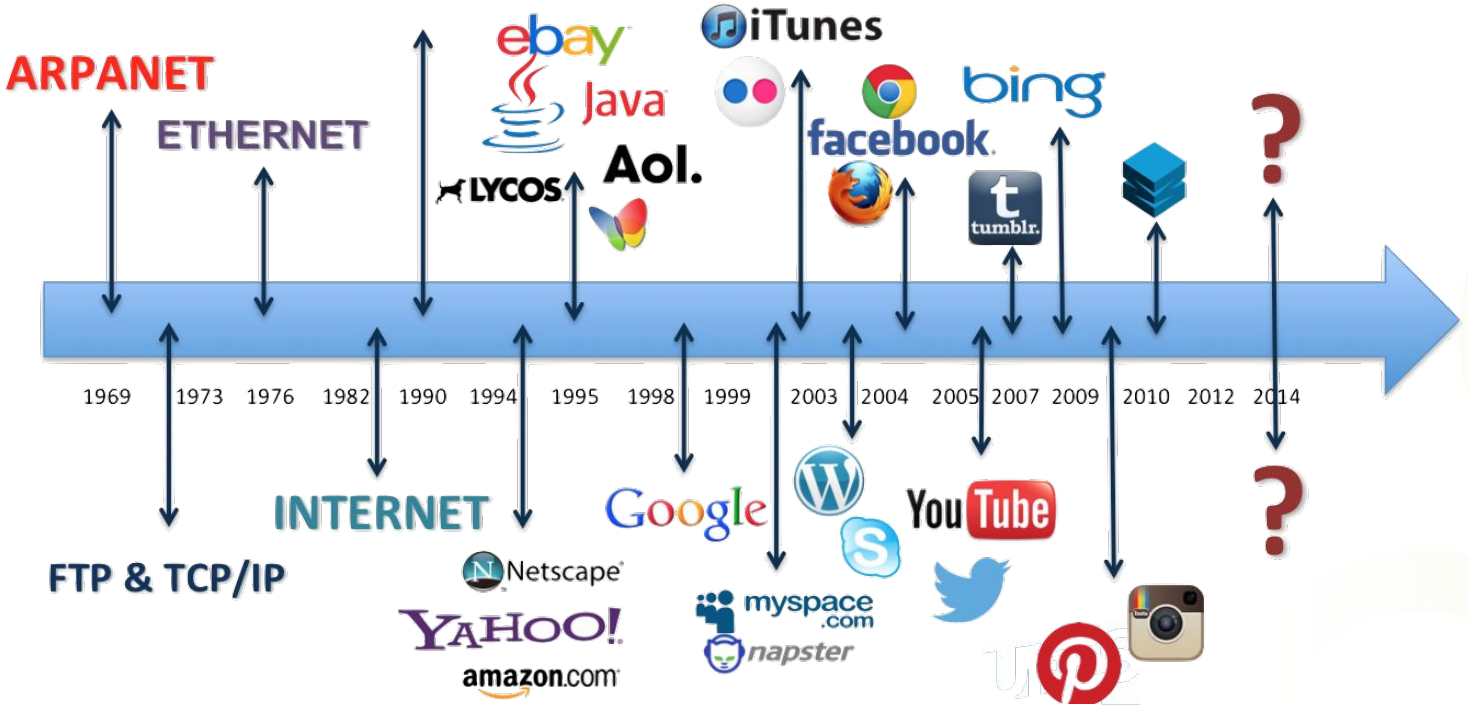
História & Fundamentos básicos

Escola Regional de Informática ES

História da Internet



História da Internet



https://ahmadfaizar.blogspot.com/2018/08/evolution-of-web-web-10-web-20-web-30.html

Evolução da Internet Web

Web 1.0

read-only
static

Web 2.0

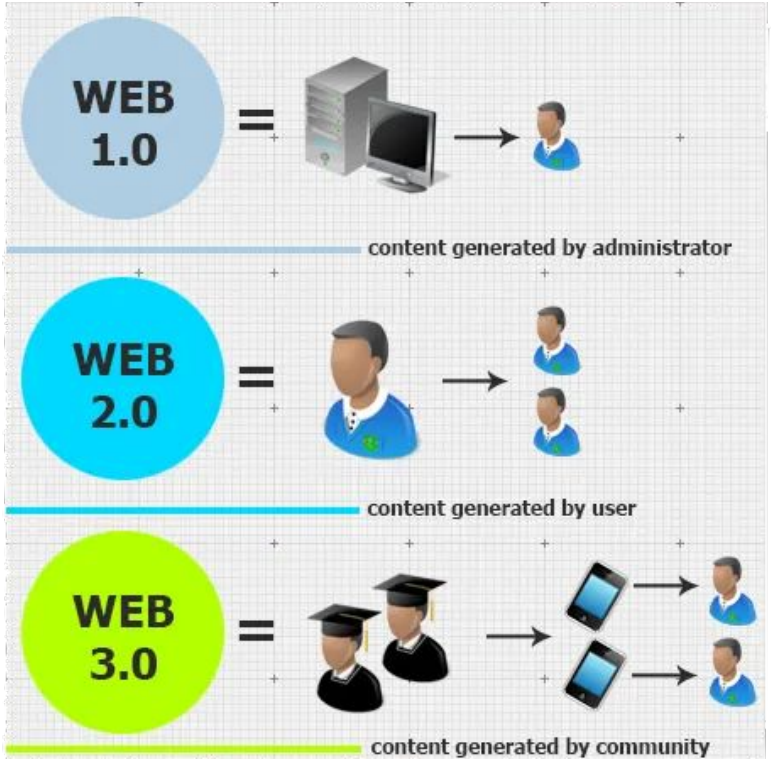
read-write
interactive

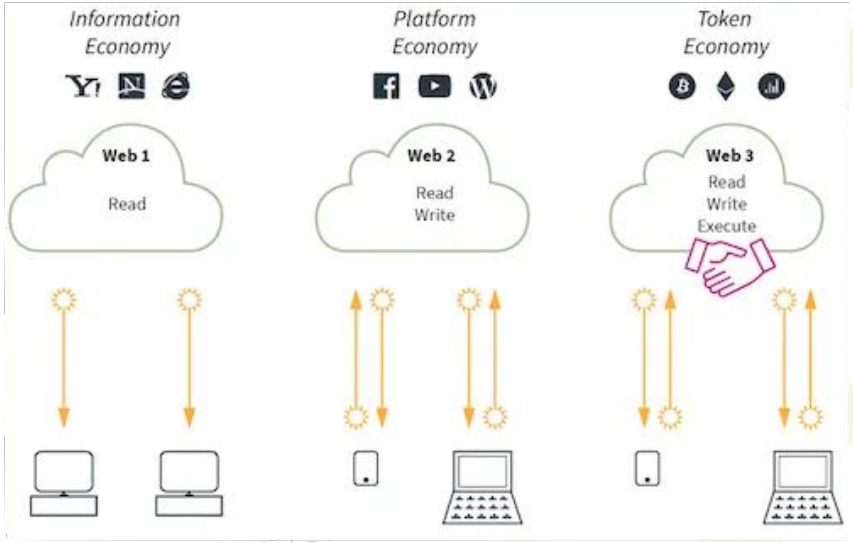
Web 3.0

read-write-trust
verifiable

https://www.linkedin.com/pulse/www-evolution-web10-web20-web30-web40-prasenjit-singh

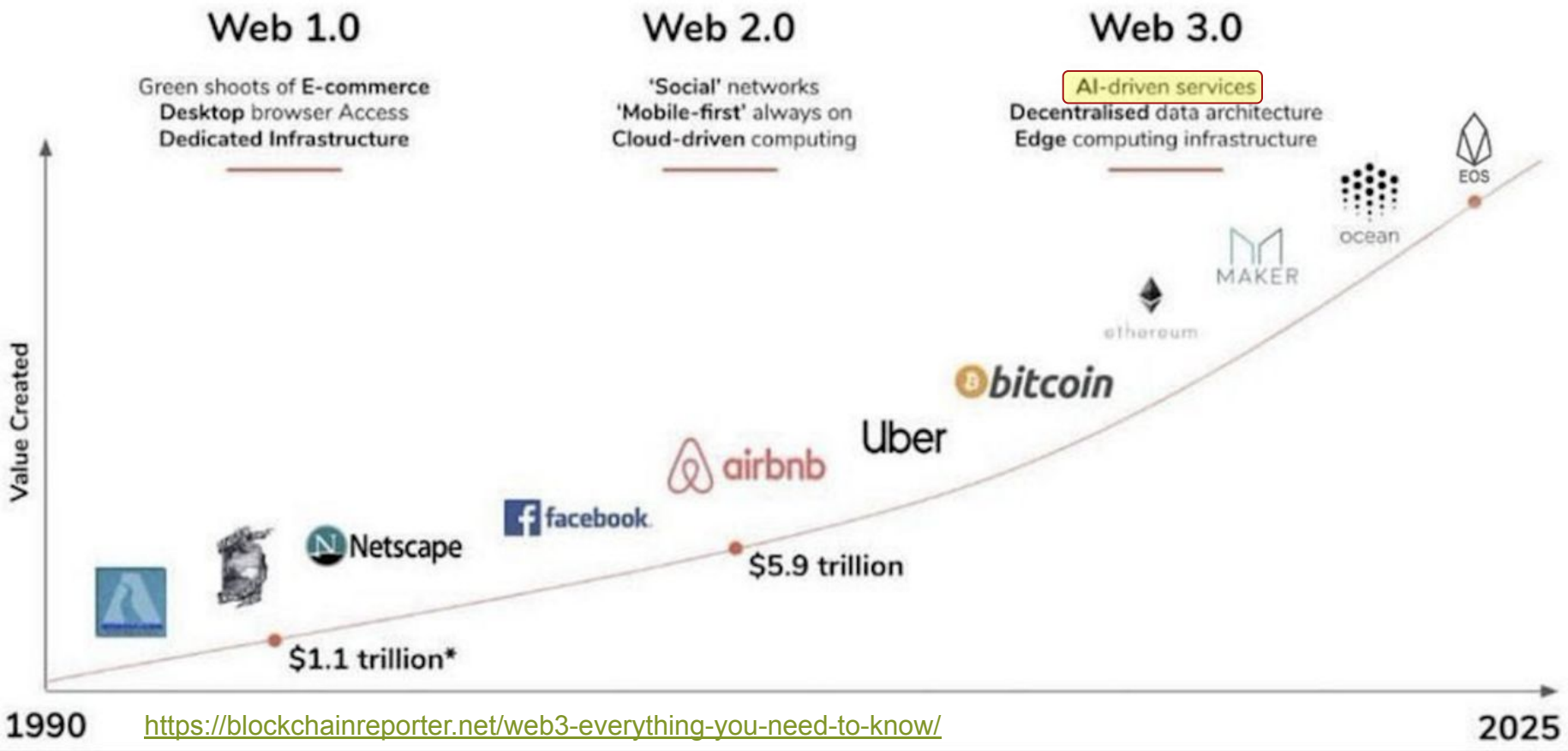
Evolução da Internet Web





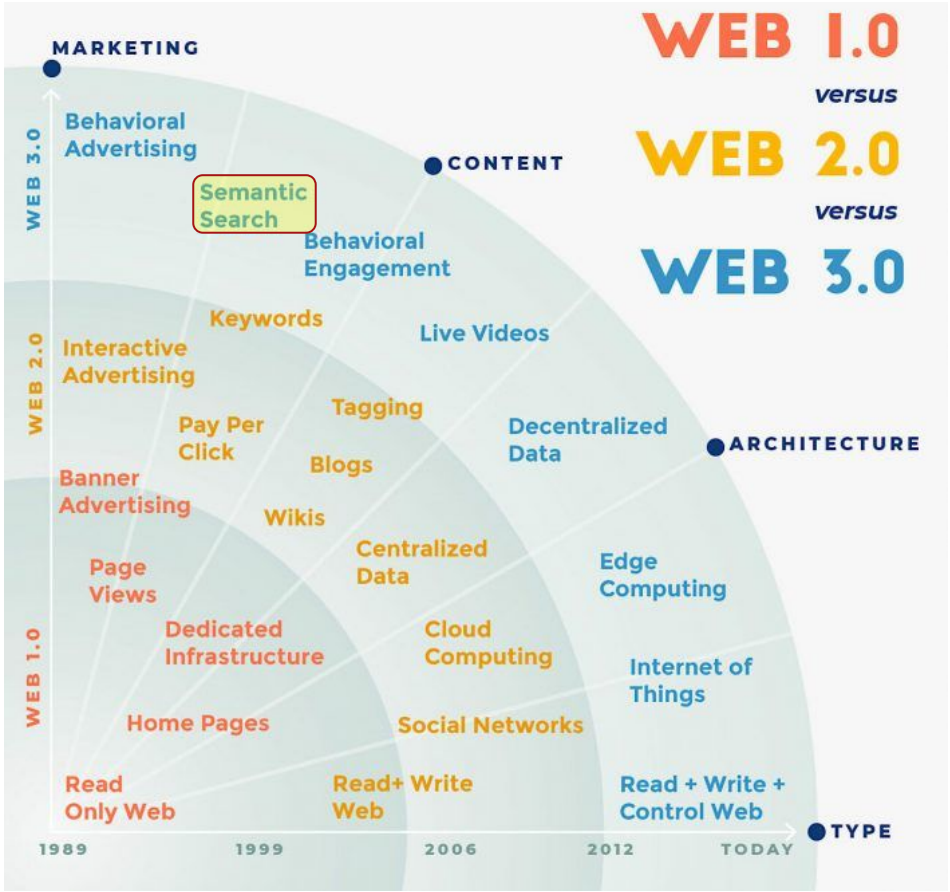
https://medium.com/@kalyanjit/the-evolution-of-the-internet-web-web-1-0-web-2-0-and-web-3-0-2aea0b302278

Evolução da Internet Web



https://blockchainreporter.net/web3-everything-you-need-to-know/

Evolução da Internet Web



https://tinyurl.com/rer228vn

Evolução da Internet Web

https://tinyurl.com/4e8h9s72

World Wide Web:

- Principalmente para que **humanos** naveguem e acessem informações.
- Conteúdo **não estruturado ou semiestruturado**, geralmente formatado em HTML para leitura humana.
- As **máquinas** entregam o conteúdo aos usuários, mas **não entendem o significado** do conteúdo.

ERIES Escola Regional de Informática ES

a beautiful mind - Google Sea

google.com/search?q=a+beautiful+mind...

Guest

Google

a beautiful mind

Wikipedia
https://pt.wikipedia.org › wiki › U... · Translate this page

Uma Mente Brilhante – Wikipédia, a enciclopédia livre

A Beautiful Mind (bra/prt: Uma Mente Brilhante) é um filme estadunidense de 2001, do gênero drama biográfico, dirigido por Ron Howard para a Universal ...

IMDb
https://www.imdb.com › title

A Beautiful Mind (2001)

A mathematical genius, John Nash made **an** astonishing discovery early in his career **and** stood on **the** brink **of** international acclaim. But **the** handsome **and** ...

8.2/10 ★★★★☆ (1,001,903)

Plot · Awards · Full Cast & Crew · Official Trailer

Wikipedia
https://en.wikipedia.org › wiki › A_Beautiful_Mind_(film)

A Beautiful Mind (film)

A Beautiful Mind is a 2001 American biographical drama film about the mathematician John Nash, a Nobel Laureate in Economics, played by Russell Crowe.

John Forbes Nash Jr. · Vivien Cardone · Michael Laudor

A Beautiful Mind (film) - Wiki

en.wikipedia.org/wiki/A_Beautiful_Mind_(film)

Guest

WIKIPEDIA
The Free Encyclopedia

Donate · Create account · Log in

A Beautiful Mind (film)

73 languages

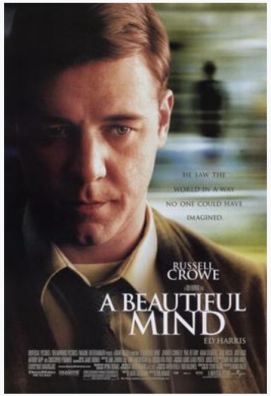
Article · Talk · Read · Edit · View history · Tools

From Wikipedia, the free encyclopedia

A Beautiful Mind is a 2001 American biographical drama film about the mathematician John Nash, a Nobel Laureate in Economics, played by Russell Crowe. The film is directed by Ron Howard based on a screenplay by Akiva Goldsman, who adapted the 1998 biography by Sylvia Nasar. In addition to Crowe, the film's cast features Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles. The story begins in Nash's days as a brilliant but asocial mathematics graduate student at Princeton University. After Nash accepts secretive work in cryptography, he becomes liable to a larger conspiracy, through which he begins to question his reality.

A Beautiful Mind was released theatrically in the United States on December 21, 2001 by Universal Pictures and internationally by DreamWorks Pictures through United International Pictures. It went on to gross over $313 million worldwide and won four Academy Awards, for Best Picture, Best Director, Best Adapted Screenplay and Best Supporting Actress for Connelly. It was also nominated for Best Actor, Best Film Editing, Best Makeup, and Best Original Score.

A Beautiful Mind



Theatrical release poster

Directed by	Ron Howard
Screenplay by	Akiva Goldsman
Based on	*A Beautiful Mind*

Web de Dados:

- Projetada para que **máquinas** acessem dados para reutilização em aplicações.
- **Dados estruturados e interligados em formato RDF**.
- Ainda **carecem de uma descrição semântica para entendimento humano e de máquina**.

a beautiful mind - Google Se

google.com/search?q=a+beautiful+mind&oq=A+Bea&gs_lcrp=EgZjaHJvbWUqBwgBEAAYjwIyBggAEEUYOTIHCAEQABiPAjIHCAIQABiPAjIHCA...

Guest

Google a beautiful mind Sign in

Cast


Russell Crowe
John Nash


Jennifer Connelly
Alicia Nash


Paul Bettany
Charles


Ed Harris
Parcher


Alicia Nash


Josh Lucas
Hansen

People also ask

O que se trata o filme Uma Mente Brilhante?

Quantos Oscars ganhou o filme Uma Mente Brilhante?

Como termina o filme Uma Mente Brilhante?

Onde passa o filme Mente Brilhante?

Feedback

Wikipedia
https://pt.wikipedia.org › wiki › U... · Translate this page

Uma Mente Brilhante – Wikipédia, a enciclopédia livre

A Beautiful Mind (bra/prt: Uma Mente Brilhante) é um filme estadunidense de 2001, do gênero drama biográfico, dirigido por Ron Howard para a Universal ...

Watch movie

EDIT SERVICES

Already watched

Want to watch

About


A Beautiful Mind (2001) Theatrical Trailer
2:29

8.2/10	74%	72%
IMDb	Rotten Tomatoes	Metacritic

84% liked this film
Google users

John Nash, a brilliant but asocial mathematical genius, finds himself in pain when he encounters a cruel disorder. He ultimately overcomes his struggles and emerges free of any trauma.

Release date: February 15, 2002 (Brazil)

Director: Ron Howard

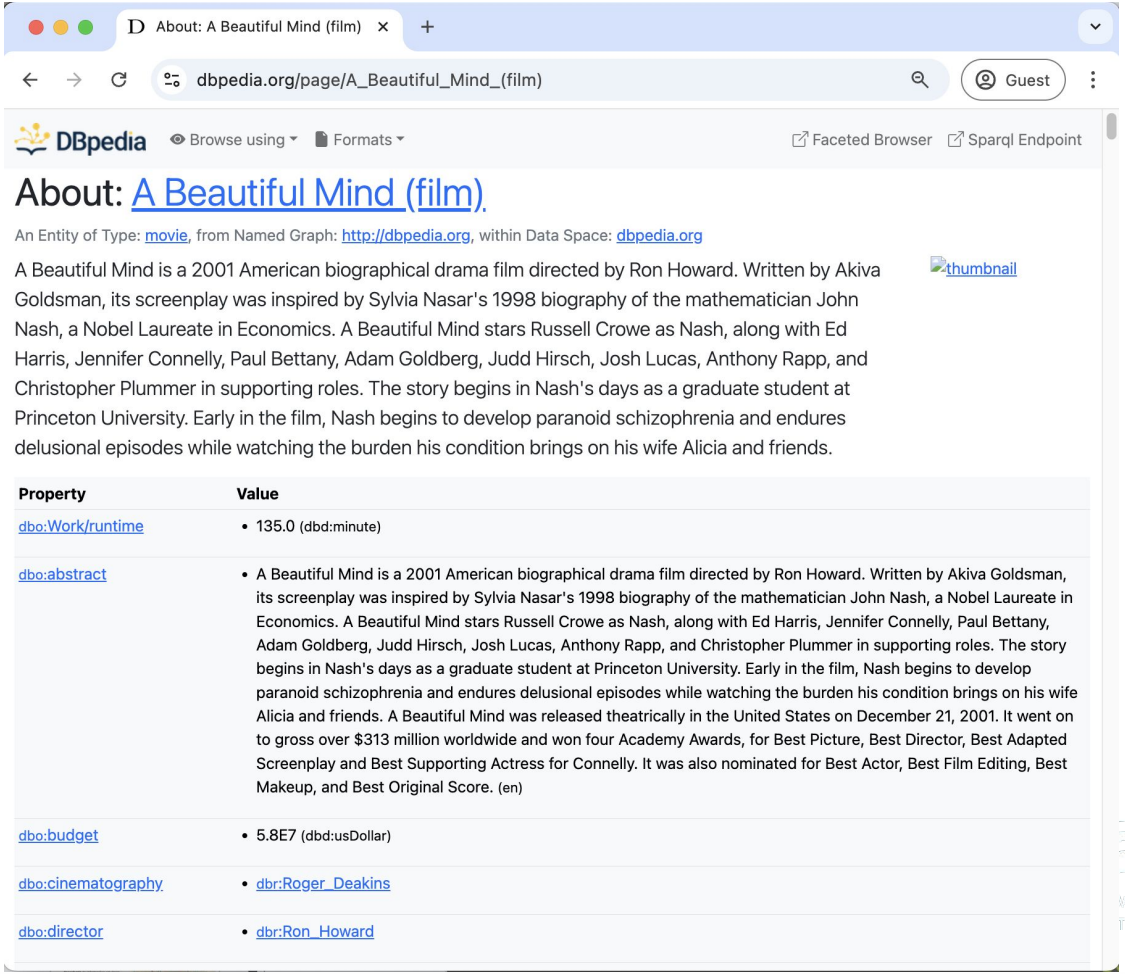
Distributed by: Universal Pictures, DreamWorks Pictures

Adapted from: A Beautiful Mind

Budget: $58 million

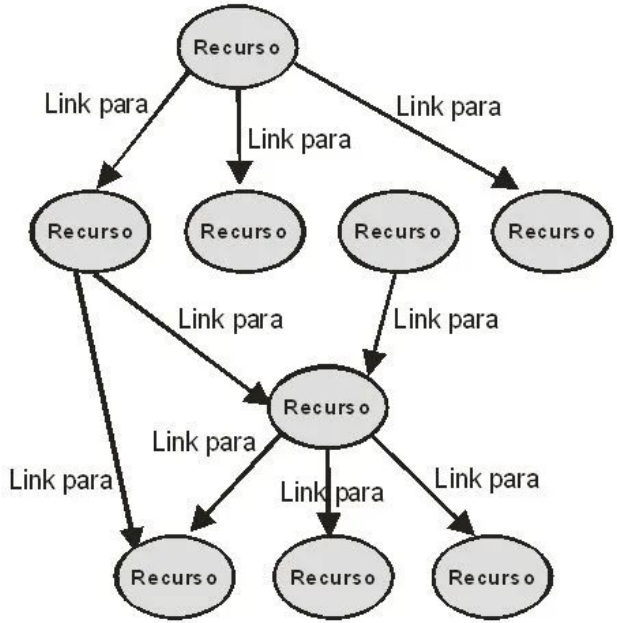
Cinematography: Roger Deakins

Feedback

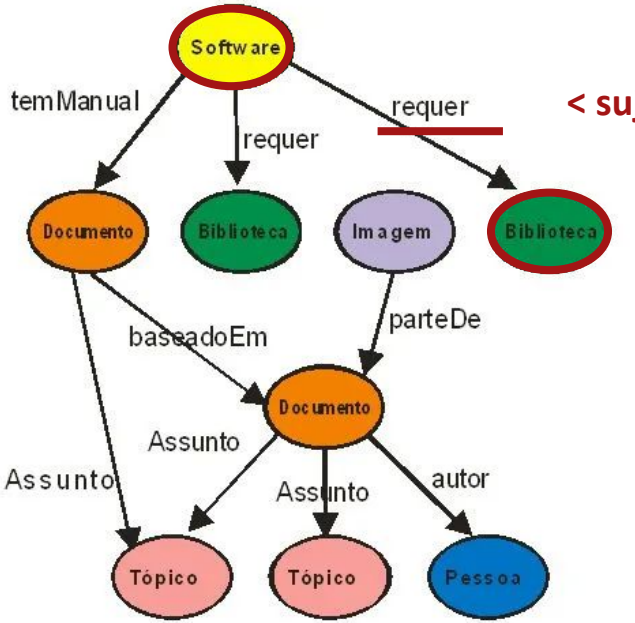

About: A Beautiful Mind (film)
dbpedia.org/page/A_Beautiful_Mind_(film)
Guest
DBpedia
Browse using
Formats
Faceted Browser
Sparql Endpoint
About: A Beautiful Mind (film)
An Entity of Type: movie, from Named Graph: http://dbpedia.org, within Data Space: dbpedia.org
A Beautiful Mind is a 2001 American biographical drama film directed by Ron Howard. Written by Akiva Goldsman, its screenplay was inspired by Sylvia Nasar's 1998 biography of the mathematician John Nash, a Nobel Laureate in Economics. A Beautiful Mind stars Russell Crowe as Nash, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles. The story begins in Nash's days as a graduate student at Princeton University. Early in the film, Nash begins to develop paranoid schizophrenia and endures delusional episodes while watching the burden his condition brings on his wife Alicia and friends.
thumbnail
Property
Value
dbo:Work/runtime
135.0 (dbd:minute)
dbo:abstract
A Beautiful Mind is a 2001 American biographical drama film directed by Ron Howard. Written by Akiva Goldsman, its screenplay was inspired by Sylvia Nasar's 1998 biography of the mathematician John Nash, a Nobel Laureate in Economics. A Beautiful Mind stars Russell Crowe as Nash, along with Ed Harris, Jennifer Connelly, Paul Bettany, Adam Goldberg, Judd Hirsch, Josh Lucas, Anthony Rapp, and Christopher Plummer in supporting roles. The story begins in Nash's days as a graduate student at Princeton University. Early in the film, Nash begins to develop paranoid schizophrenia and endures delusional episodes while watching the burden his condition brings on his wife Alicia and friends. A Beautiful Mind was released theatrically in the United States on December 21, 2001. It went on to gross over $313 million worldwide and won four Academy Awards, for Best Picture, Best Director, Best Adapted Screenplay and Best Supporting Actress for Connelly. It was also nominated for Best Actor, Best Film Editing, Best Makeup, and Best Original Score. (en)
dbo:budget
5.8E7 (dbd:usDollar)
dbo:cinematography
dbr:Roger_Deakins
dbo:director
dbr:Ron_Howard

World Wide Web x Web de Dados

wikipedia



dbpedia



< suj, pred, obj >

https://www.deviante.com.br/noticias/web-semantica-a-informacao-tornando-se-conhecimento/

Princípios de Linked Open Data

★

Estar **disponível na Web**, em **qualquer formato**, desde que seja com uma **licença aberta**, sendo assim Dado Aberto

★★

Estar disponível num **formato estruturado legível por máquina** (e.g. excel ao invés de uma foto de uma tabela

★★★

Estar num **formato não-proprietário** (e.g. CSV ao invés de excel)

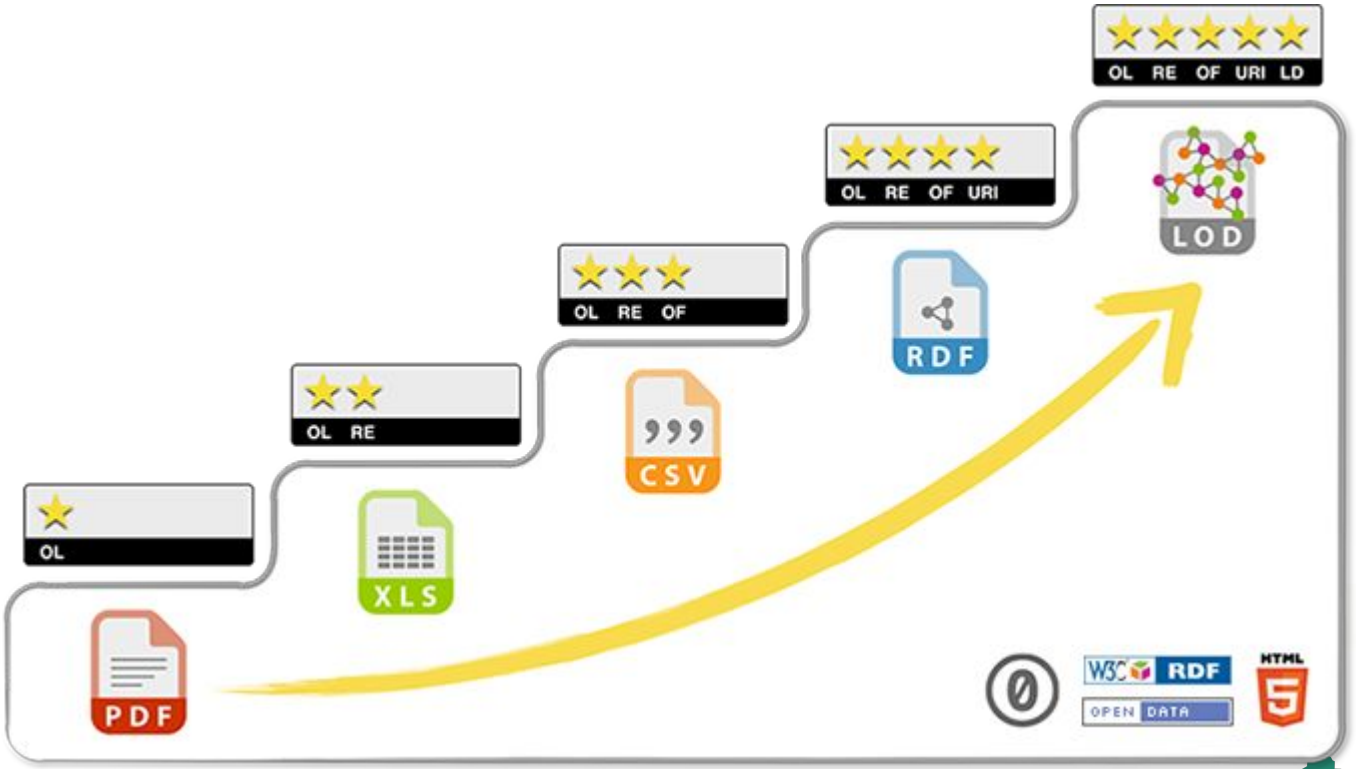
★★★★

Usar **padrões do W3C (URI, RDF, etc)** para identificar recursos, de forma que outros possam "apontar para" seus recursos

★★★★★

Ligar seus dados ao de outros para prover contexto

Princípios de Linked Open Data



Nuvem de Dados Ligados

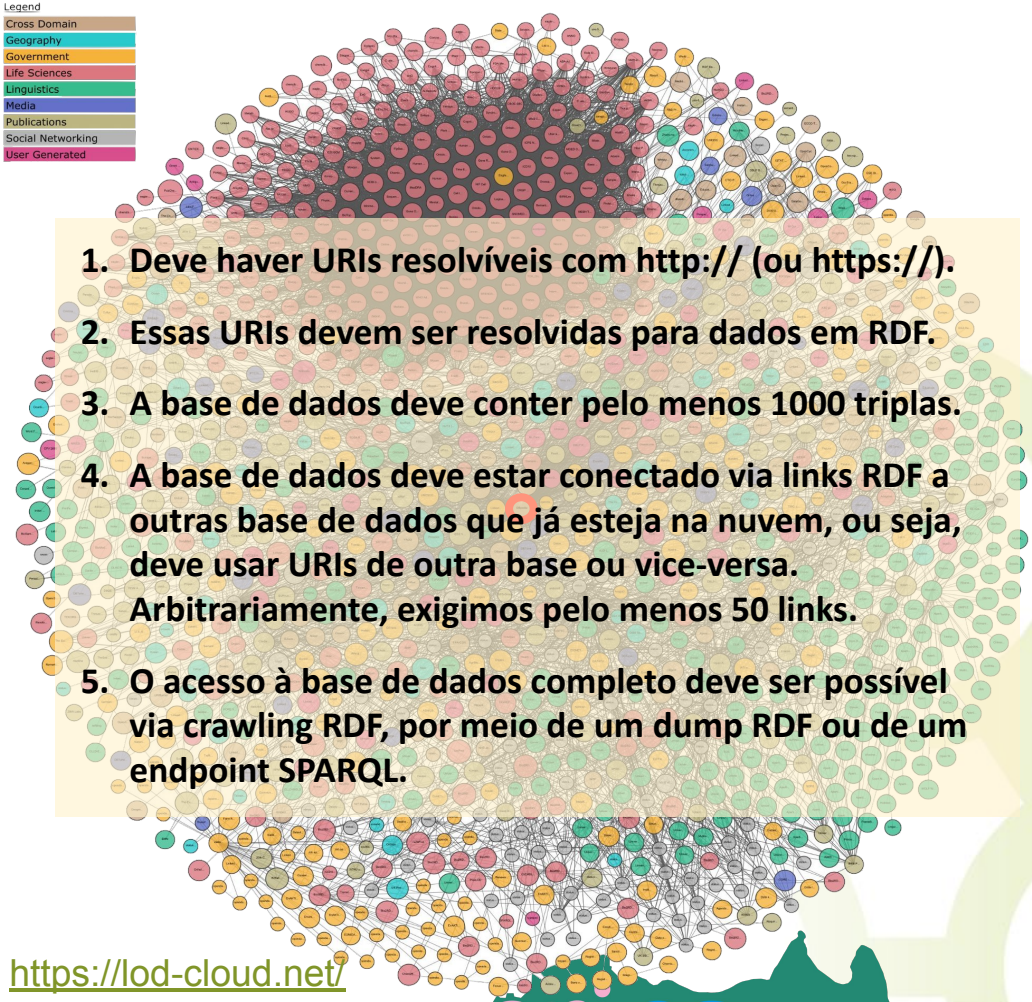
2007: **12** bases de dados

2014: **570** bases de dados

2020: **1255** bases de dados
16174 links

2023-2024 **+1.400** bases de dados,
+16 bilhões de triplas RDF
milhões de links RDF.

Links facilitam a **integração** e a **descoberta** de informações em áreas, como ciências biológicas, geografia, governo e patrimônio cultural.



1. **Deve haver URIs resolvíveis com http:// (ou https://).**
2. **Essas URIs devem ser resolvidas para dados em RDF.**
3. **A base de dados deve conter pelo menos 1000 triplas.**
4. **A base de dados deve estar conectado via links RDF a outras base de dados que já esteja na nuvem, ou seja, deve usar URIs de outra base ou vice-versa. Arbitrariamente, exigimos pelo menos 50 links.**
5. **O acesso à base de dados completo deve ser possível via crawling RDF, por meio de um dump RDF ou de um endpoint SPARQL.**

https://lod-cloud.net/

Web Semântica:

- Permite que as **máquinas** não apenas acessem, mas também **interpretem (logicamente) os dados**.
- Enriquece os **dados estruturados** com **significado semântico** os conceitos, propriedades e instâncias, possibilitando inferência de **novas informações** ou verificação de **inconsistências**.

Se der tempo…

- owl:sameAs: 558,9 M
- rdfs:subClassOf: 4,4 M
- owl:equivalentClass: 1 M
- owl:disjointWith: 450 K
- rdfs:domain: 206 K
- rdfs:range: 197,5 K
- rdfs:subPropertyOf: 80 K
- owl:equivalentProperty: 8,4 K
- owl:differentFrom 3,6 K

So in total that is just over 565M statements that would classify as “rules”. The total size of the LOD-a-lot crawl is 28.3B unique statements (the crawl is all deduciplicated). So that would make it **just under 2% of the entire LOD cloud**). (Notice also the very skewed frequency distribution of these statements; without owl:sameAs it would only by 0.02% of the entire LOD cloud).

https://frankvanharmelen.home.blog/

Quem usa Linked Data?

Empresas:

- Google
- Yahoo
- Microsoft

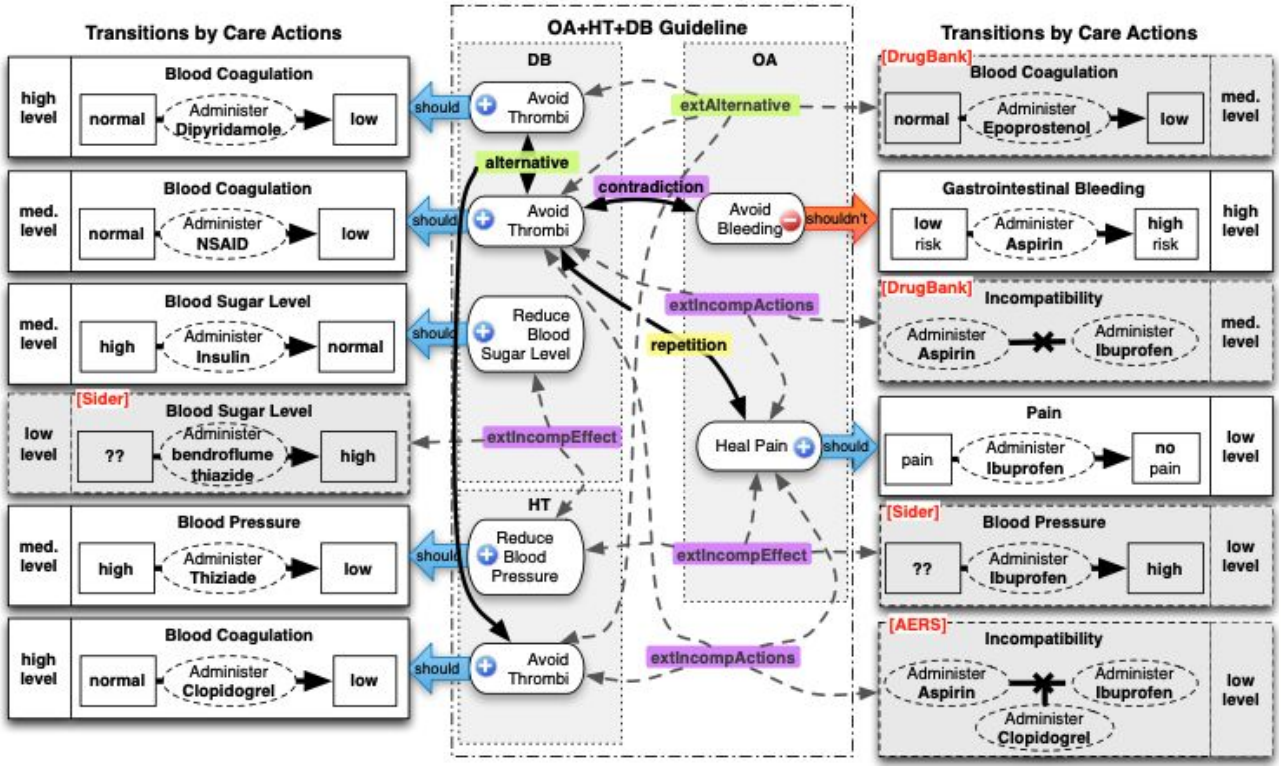
Governos

Indústria Farmacêutica

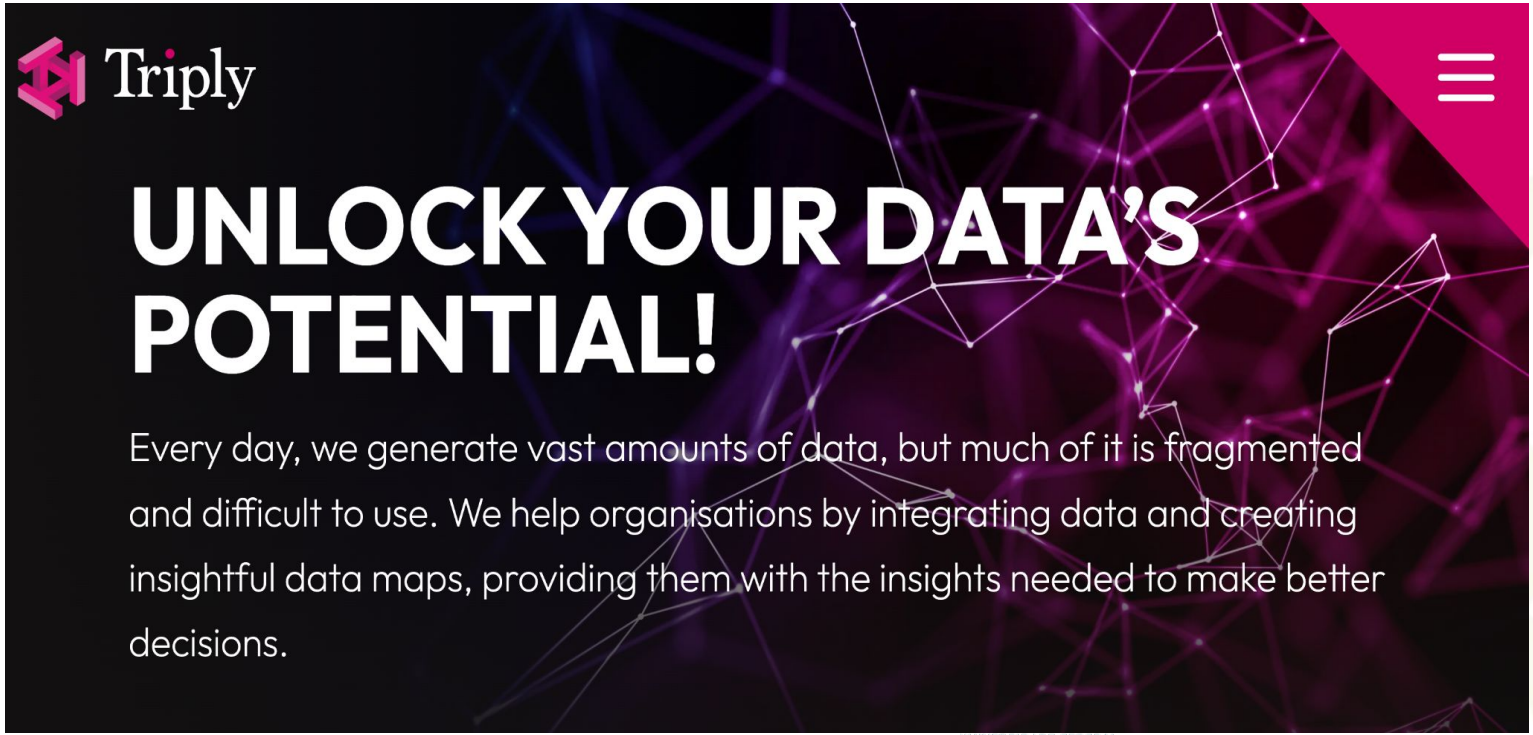
Academia

- http://dbpedia.org/
- http://datahub.io/
- http://datadryad.org/
- http://datacatalogs.org/
- http://www.openstreetmap.org/
- http://www.freebase.com/
- http://data.gov.uk/
- http://www.data.gov/
- http://publicdata.eu/
- http://doc.metalex.eu/
- http://linkeddata.few.vu.nl/
- …

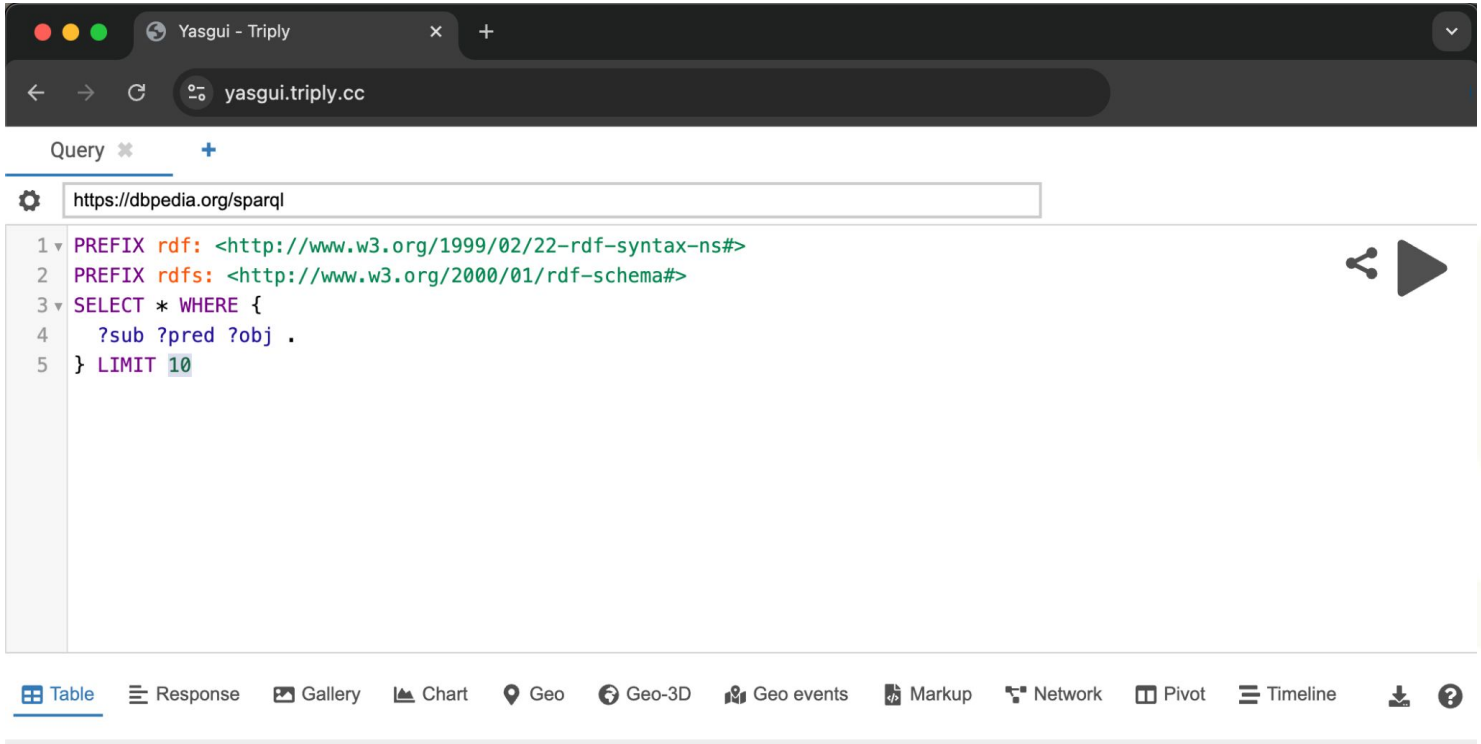
Quem usa Linked Data?



triply.cc → 22 funcionários e expandindo



yasgui.triply.cc



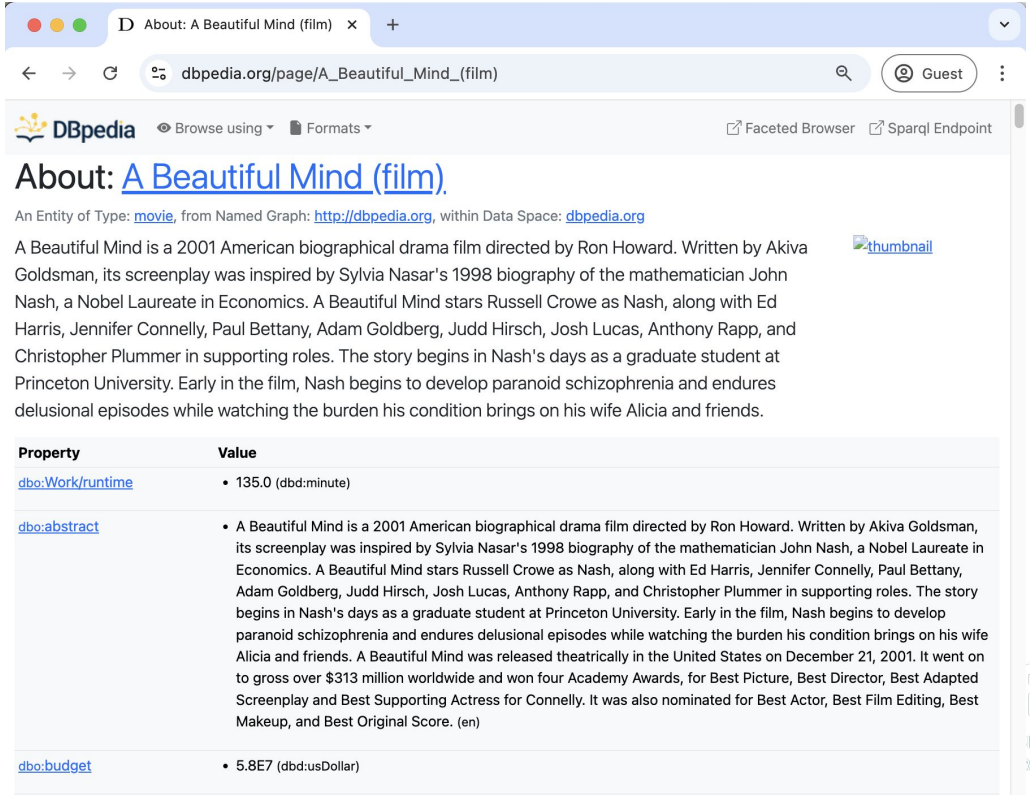
Mãos nos dados

ERIES Escola Regional de Informática ES

Meu 1o recurso RDF - Dbpedia

- Abra um navegador
- Pesquise por algo como "dbpedia beautiful mind film"
- Verifique se aparece um link para o filme começando com:
  - http://dbpedia.org/…
- Navegue pelos recursos ligados ao seu recurso

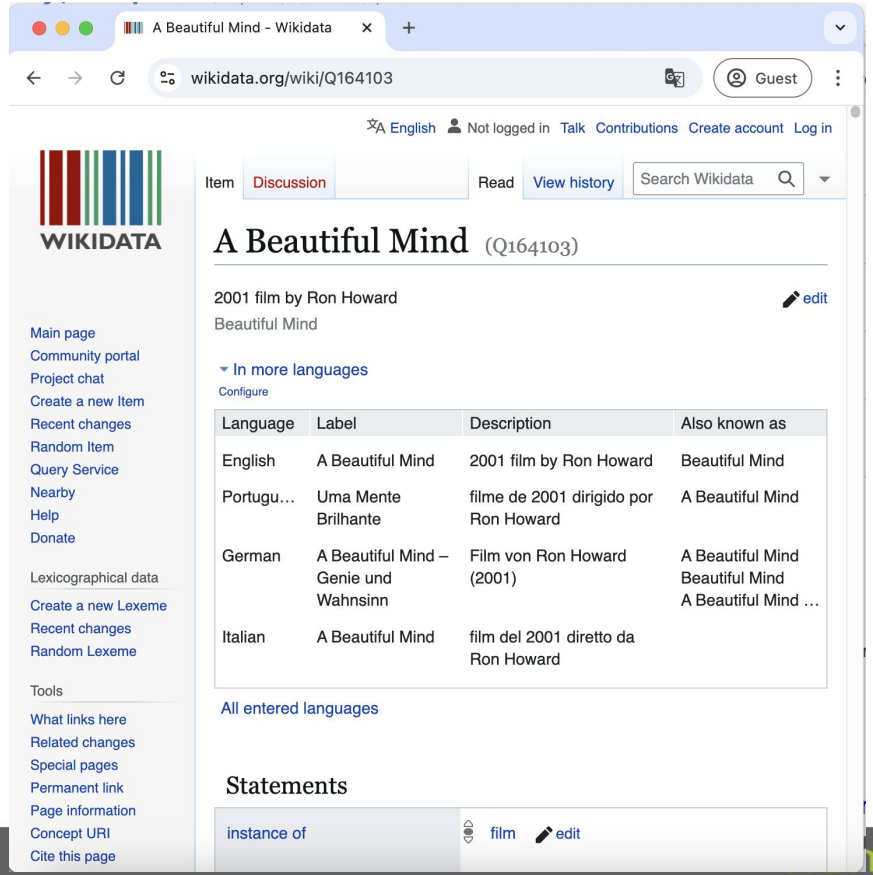
Meu 1o recurso RDF



Meu 2o recurso RDF - Wikidata

- Abra um navegador
- Pesquise por algo como "wikidata beautiful mind film"
- Verifique se aparece um link para o filme começando com:
  - https://wikidata.org/…
- Navegue pelos recursos ligados ao seu recurso

Meu 2o recurso RDF - Wikidata



Eles são mesmo diferentes?

Minha 1a consulta SPARQL

- Abra o site
  - https://yasgui.triply.cc/
- Aparecerá a seguinte query
-

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
 ?sub ?pred ?obj .
}
LIMIT 10
```

- Execute-a 
  - ***endpoint*** padrão: http://dbpedia.org/sparql

Esta consulta retorna as 10 primeiras "triplas" (`LIMIT 10`) encontradas no endpoint indicado (http://dbpedia.org/sparql), que satisfaçam o padrão indicado na consulta. Neste caso, (`?sub ?pred ?obj`) onde a ? indica variáveis que serão "casadas" com os valores das triplas que "casarem" com o padrão. Um ponto (.) deve ser usado ao final de cada padrão-tripla. Nesta consulta, qualquer tripla casará com este padrão, pois ele é "aberto", não especificando nenhuma parte da tripla, apenas variáveis.

Minha 2a consulta SPARQL

- Modifique o endpoint e execute-a de novo 
  - ***endpoint***: escreva wikidata e selecione

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
 ?sub ?pred ?obj .
}
LIMIT 10
```

Esta consulta retorna as 10 primeiras "triplas" (`LIMIT 10`) encontradas no endpoint indicado (wikidata endpoint), que satisfaçam o padrão indicado na consulta. Neste caso, (`?sub ?pred ?obj`) onde a ? indica variáveis que serão "casadas" com os valores das triplas que "casarem" com o padrão.
Um ponto (.) deve ser usado ao final de cada padrão-tripla.
Nesta consulta, qualquer tripla casará com este padrão, pois ele é "aberto", não especificando nenhuma parte da tripla, apenas variáveis.

Porque os resultados são diferentes?

Como eu consigo achar os meus recursos?

Minha 1a consulta SPARQL

- Substitua a variável ?sub pela URI do recurso dbpedia

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
 <http://dbpedia.org/resource/A_Beautiful_Mind_(film)> ?pred ?obj .
}
LIMIT 10
```

- Execute-a 
  - ***endpoint***: http://dbpedia.org/sparql

Esta consulta retorna as 10 primeiras "triplas" (`LIMIT 10`) encontradas no endpoint indicado (http://dbpedia.org/sparql), que satisfaçam o padrão indicado na consulta. Neste caso, os resultados serão os 10 primeiros pares (predicado, objeto) que estejam ligados ao recurso indicado.

Minha 1a consulta SPARQL

- Substitua a variável ?sub pela URI do recurso dbpedia

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
   <http://www.wikidata.org/entity/Q164103> ?pred ?obj .
}
LIMIT 10
```

- Execute-a 
  - ***endpoint***: wikidata

Esta consulta retorna as 10 primeiras "triplas" (`LIMIT 10`) encontradas no endpoint indicado (wikidata endpoint), que satisfaçam o padrão indicado na consulta. Neste caso, os resultados serão os 10 primeiros pares (predicado, objeto) que estejam ligados ao recurso indicado.

Adobe Stock

Fundamentos básicos

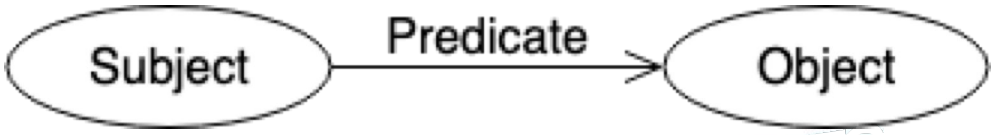
slides adaptados de Tiago Sales,
inspirados no livro
The Web of Data de Aidan Hogan
https://aidanhogan.com/wodata/book.pdf

ERIES Escola Regional de Informática ES

Como funciona isso?
(versão simplificada)

RDF: Resource Description Framework

- padrão do W3C projetado para ser o modelo de dados padrão para a web.
- prescreve um modelo de dados estruturado em grafo, que é inerentemente mais flexível e fácil de integrar do que árvores ou tabelas
- foi concebido com a Web em mente, apresentando um esquema de nomeação global que utiliza identificadores nativos da Web.



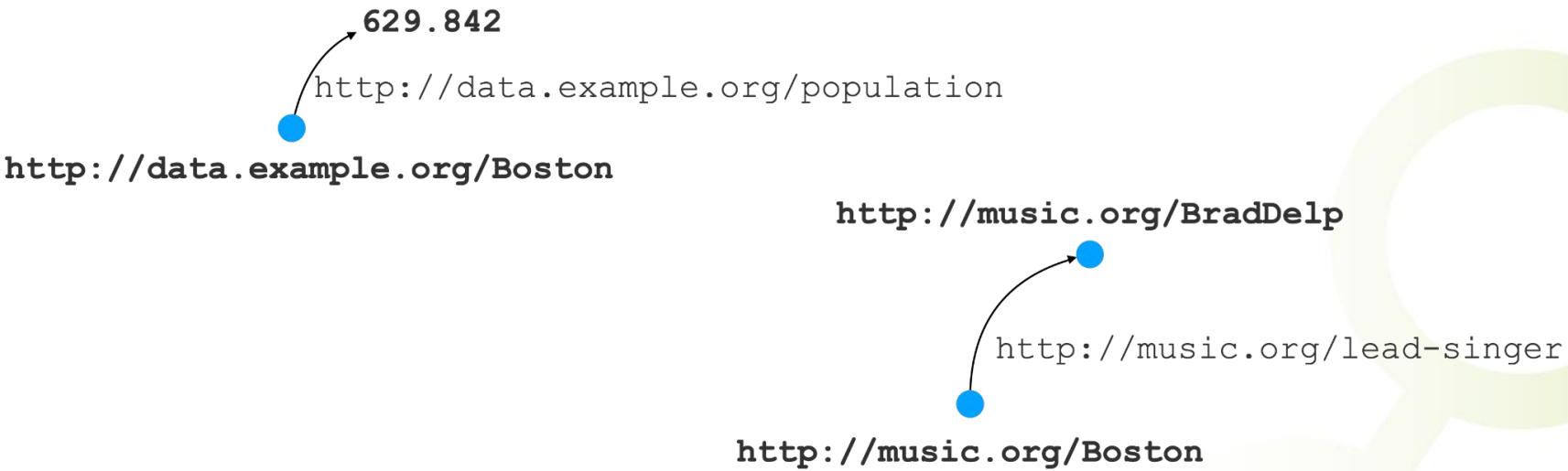
RDF: Resource Description Framework

- Framework para descrição de **recursos.**
- **Recursos** são qualquer coisa com **identidade** que se possa considerar descrever em dados, incluindo:
  - **entidades virtuais**: páginas da web, sites, arquivos de desktop;
  - **entidades concretas**: livros, pessoas, lugares, lojas;
  - **entidades abstratas**: categorias, espécies de animais, pontos no tempo, relações de ancestralidade

RDF: Identificador

- **URL**: Uniform Resource Locator
  - Localiza **documentos** na web
  - E.g. páginas HTML, PDFs
- **URI**: Uniform Resource Identifier
  - **Generalização de URLs para qualquer tipo de recurso**
  - E.g. páginas HTML, PDFs, cidades, cantores
  - Não precisa ser resolvível
  - Limitado a caracteres ASCII: Québec seria Qu%C3%A9bec
- **IRI**: Internationalised Resource Identifier
  - **Generalização de URIs para acomodar qualquer caracter**

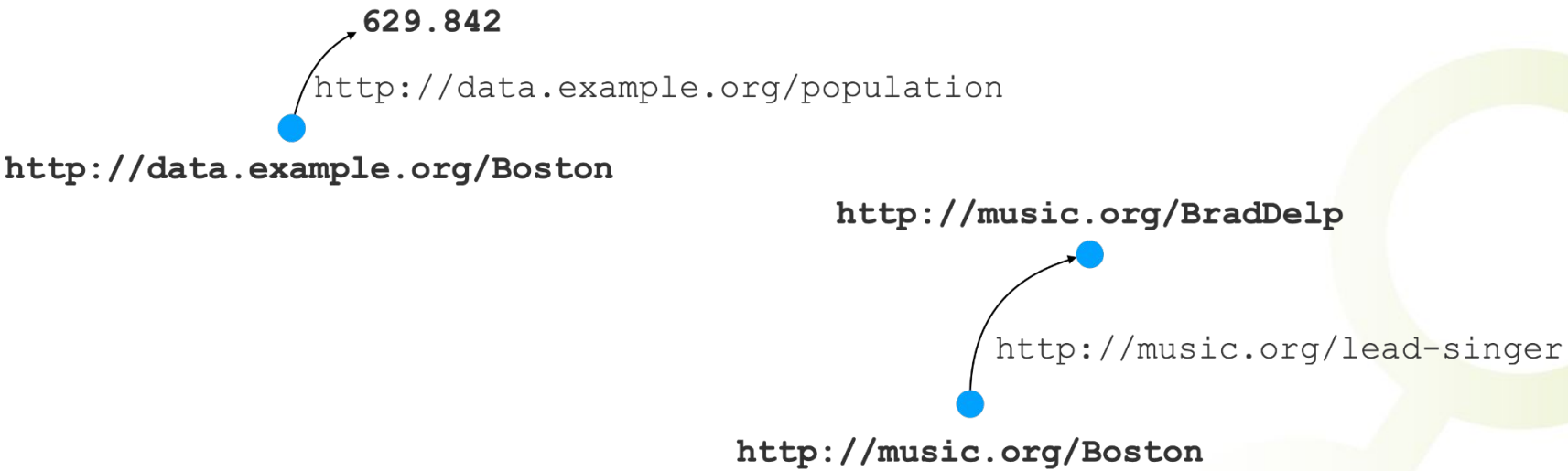
RDF: Resource Description Framework



RDF: Prefixos

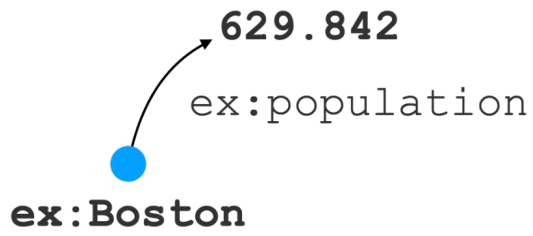
- Muitas sintaxes suportarão o uso de prefixos para **IRIs**.
- Se definirmos o prefixo
  - ex: como http://data.example.org/city/ ,
- podemos usar
  - http://data.example.org/city/Boston como ex:Boston

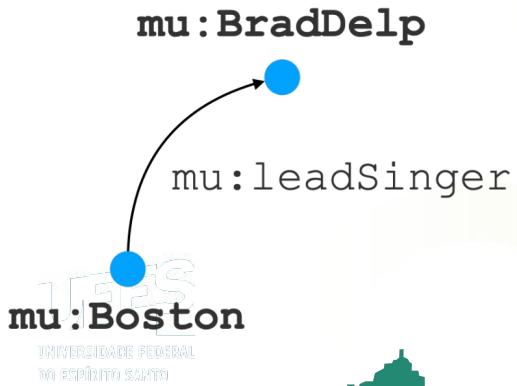
RDF: Resource Description Framework



RDF: Resource Description Framework

ex:http://data.example.org/
mu:http://music.org/





RDF: Literais

- São usados para fornecer informações legíveis por humanos, como títulos, descrições, datas, valores numéricos.
- Um literal pode ser:
  - **forma lexical**: uma string unicode;
  - **IRI de tipo de dado**: identifica o 'tipo' do literal;
  - **tag de idioma**: indicando a língua humana do texto (de acordo com o BCP-47).

RDF: Literais - Exemplos

- **Literais simples**:
  - "Strawberry Cheesecake"
  - 15
- **Lang strings**:
  - "Strawberry Cheesecake"@en
  - "Torta de queijo doce de morango"@pt-br
- **Literais tipados**:
  - "Strawberry Cheesecake"^^xsd:string
  - "-8.5"^^xsd:decimal

RDF: Tipos de Dados suportados

- **Booleano:** Um único tipo de dado verdadeiro/falso.
- **Numérico:** Um conjunto de tipos de dados que se referem a valores numéricos.
  - Valores numéricos precisos (classificados como xsd:decimal).
  - Aproximações de valores numéricos em ponto flutuante IEEE (xsd:double ou xsd:float).
- **Temporal:** Um conjunto de tipos de dados para definir datas, horários, durações e datas recorrentes/parciais de acordo com o calendário gregoriano.
- **Texto:** Um conjunto de tipos de dados que geralmente se referem a sequências de caracteres, possivelmente codificados ou seguindo uma sintaxe preestabelecida.

RDF: Tipos de Dados suportados

Datatype IRI	Example Lexical Forms	Note
	BOOLEAN	
xsd:boolean	"true", "false", "1", "0"	Case sensitive
	NUMERIC	
xsd:decimal	"-2.320"	Any precision
└xsd:integer	"-3"	Any precision, $x \in \mathbb{Z}$
└xsd:long	"-9223372036854775808"	$-2^{63} \leq x < 2^{63}$
└xsd:int	"+2147483647"	$-2^{31} \leq x < 2^{31}$
└xsd:short	"-32768"	$-2^{15} \leq x < 2^{15}$
└xsd:byte	"127"	$-2^{7} \leq x < 2^{7}$
└xsd:nonNegativeInteger	"0", "4"	$0 \leq x < \infty$
└xsd:positiveInteger	"+1", "3152"	$1 \leq x < \infty$
└xsd:unsignedLong	"18446744073709551615"	$0 \leq x < 2^{64}$
└xsd:unsignedInt	"+4294967295"	$0 \leq x < 2^{32}$
└xsd:unsignedShort	"65535"	$0 \leq x < 2^{16}$
└xsd:unsignedByte	"+255"	$0 \leq x < 2^{8}$
└xsd:nonPositiveInteger	"0", "-4"	$x \leq 0$
└xsd:negativeInteger	"-3152"	$x < 0$
xsd:double	"1.7e308" "-4.9E-324", "NaN", "INF", "-INF"	IEEE 64-bit floating point
xsd:float	"3.4E38", "-1.4e-45", "NaN", "INF", "-INF"	IEEE 32-bit floating point

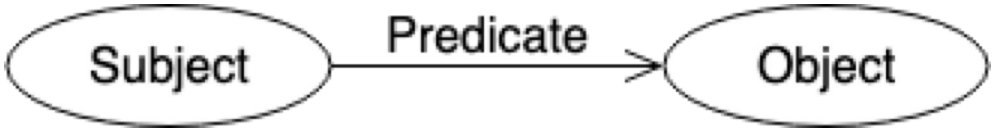
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

RDF: Tipos de Dados suportados

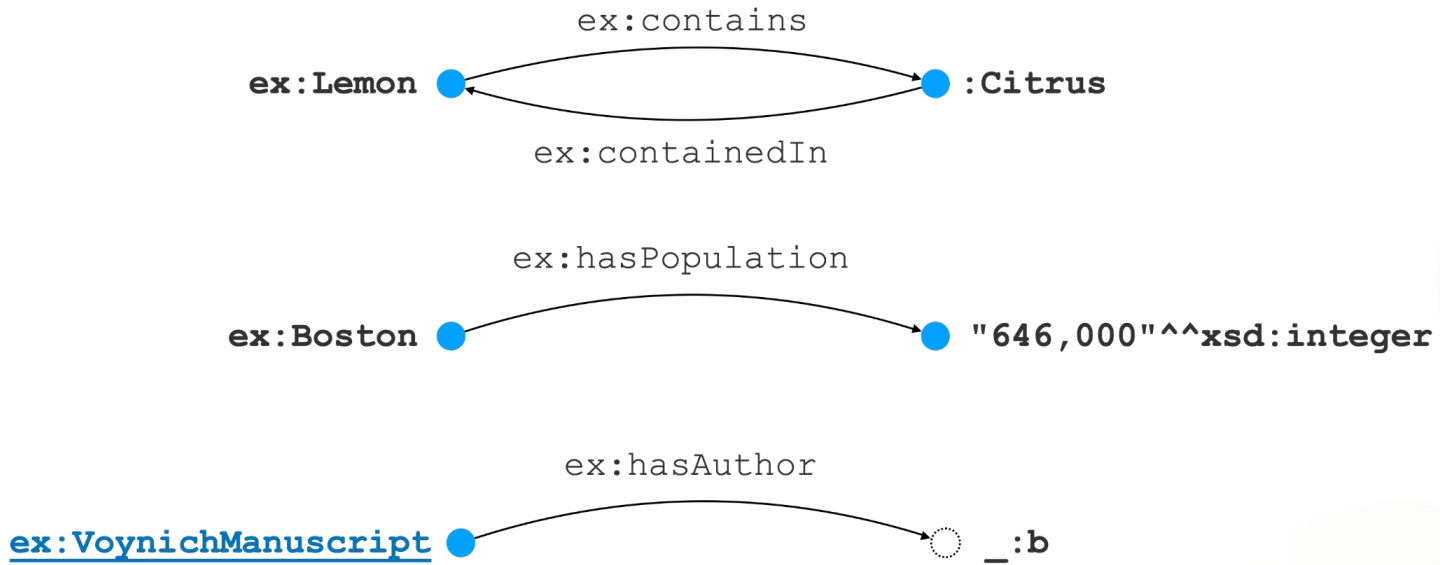
Temporal		
xsd:time	"05:04:12", "05:04:12Z", "05:04:12.00-10:00"	Z indicates +00:00 time-zone
xsd:date	"2012-02-29", "2012-12-31+04:00"	Time-zone optional
xsd:dateTime	"2012-12-31T00:01:02.034"	Time-zone optional
└xsd:dateTimeStamp	"2012-12-31T00:01:02+04:00"	Time-zone required
xsd:duration	"P6Y9M15DT25H61M4.2S", "P6Y4.2S"	6 Years (…) 4.2 Seconds
└xsd:dayTimeDuration	"P2DT8H14S"	No month or year
└xsd:yearMonthDuration	"-P89Y13M"	No days or time
xsd:gDay	"---15", "---01-13:59"	Day recurring every month
xsd:gMonth	"--12", "--01+14:00"	Month recurring every year
xsd:gMonthDay	"--02-29", "--03-01Z"	Date recurring every year
xsd:gYear	"1985", "-0005"	A year ($-y$ indicates B.C.)
xsd:gYearMonth	"1985-05", "-0005-02"	A specific month
Text		
xsd:string	" tab-> <-tab "	Most Unicode characters
└xsd:normalizedString	" multiple-> <-spaces "	No \r, \n, \t
└xsd:token	"one-> <-space"	No leading or double spaces
└xsd:language	"en", "en-UK", "en-uk", "zh-yue-Hant"	Generalises BCP47 [316]
└xsd:name	"ns:some_name"	XML names
└xsd:NCName	"some_name"	XML names: no colons
└xsd:NMTOKEN	"1some_name"	XML names: 1st char relaxed
xsd:base64Binary	"QS5ILiBuZWVkcyBhIHNtb2tlLg=="	Base-64 encoded strings
xsd:hexBinary	"2e2e2e20616e6420616c636f686f6c2e"	Hexadecimal strings
xsd:anyURI	"http://example.com/",	Full IRI strings
rdf:HTML	" some data "	Well-formed HTML content
rdf:XMLLiteral	"<flavour><fruit>apple</fruit></flavour>"	Well-formed XML content

RDF: Formando triplas

- **sujeitos (*subjects*)** podem apenas conter IRIs ou blank nodes;
- **predicados (*predicates*)** podem apenas conter IRIs;
- **objetos (*objects*)** podem conter IRIs, blank nodes ou literais.



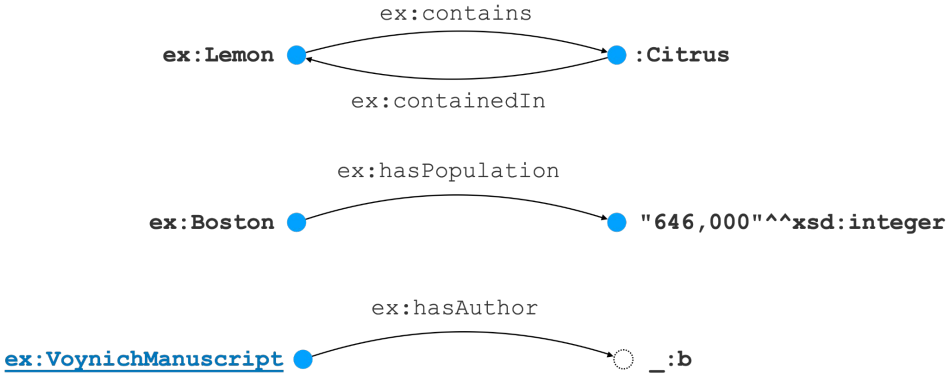
RDF: Formando triplas - Exemplo

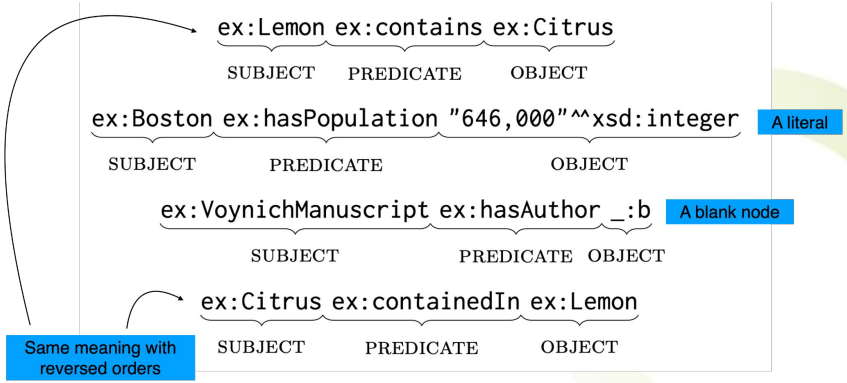


RDF: Sintaxes

- RDF/XML
- N-Triples
- Turtle
- JSON-LD

RDF: Turtle - Exemplo





RDF: Classes

- **Classes** são grupos de recursos com algumas semelhanças conceituais
  - agrupam zero ou muitos recursos;
    - frequentemente chamada de **tipo** desses recursos.
  - um recurso pode ser membro de várias classes,
    - frequentemente chamado de **instância** dessa classe.
  - são termos usados **principalmente** na posição de **objeto**;
- RDF fornece uma propriedade incorporada para denotar **instâncias de classes**
  - **rdf:type** uma propriedade para relacionar uma instância a seu tipo.
    - em turtle e sparql pode ser usada simplesmente como **a**, abreviação para ***is-a*** (é um/a)

RDF: Classes e Propriedades

ex:LemonCheesecake	~~rdf:type~~ a	ex:DessertRecipe
ex:LemonCheesecake	ex:contains	ex:Lemon
ex:LemonCheesecake	ex:rating	"4.3"^^xsd:decimal
ex:Lemon	~~rdf:type~~ a	ex:Ingredient
ex:Lemon	~~rdf:type~~ a	ex:Fruit
ex:Lemon	rdfs:label	"Lemon"@en
ex:LemonCheesecake	ex:contains	ex:Cheese
ex:Cheese	~~rdf:type~~ a	ex:Ingredient
ex:Cheese	~~rdf:type~~ a	ex:Dairy
ex:LemonCheesecake	ex:contains	_:b
_:b	~~rdf:type~~ a	ex:Ingredient
_:b	~~rdf:type~~ a	ex:BakedGoods

RDF: Classes e Propriedades

- É convencional no RDF que:
  - **Classes** recebam nomes singulares com a primeira letra **maiúscula**.
    - *ex:Person*
  - **Predicados** recebam nomes singulares com a primeira letra **minúscula**.
    - *ex:name, ex:father*
  - Predicados entre dois recursos é chamado de ***object-property*** e entre um recurso e um literal é chamado de ***data-property***
    - É importante que ***object-properties*** reflitam sua **direção** de aplicação
    - Ao invés de *ex:father*, é preferível *ex:fatherOf* ou *ex:isFatherOf*

RDF: Turtle - Exemplo

```
@prefix ex: <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .

ex:Department1 a ex:Department ;
   dcterms:title "Human Resources"@en, "HR" ;
   ex:hasManager ex:Employee1 .

ex:Department2 a ex:Department ;
   dcterms:title "Software Engineering"@en, "SE" .

ex:Employee1 a foaf:Person ;
   foaf:name "Alice Smith" ;
   ex:worksIn ex:Department1 .

ex:Employee2 a foaf:Person ;
   foaf:name "Bob Johnson" ;
   ex:worksIn ex:Department2 .

ex:Employee3 a foaf:Person ;
   foaf:name "Carol White" ;
   ex:worksIn ex:Department2 .
```

Mãos nos dados

ERIES Escola Regional de Informática ES

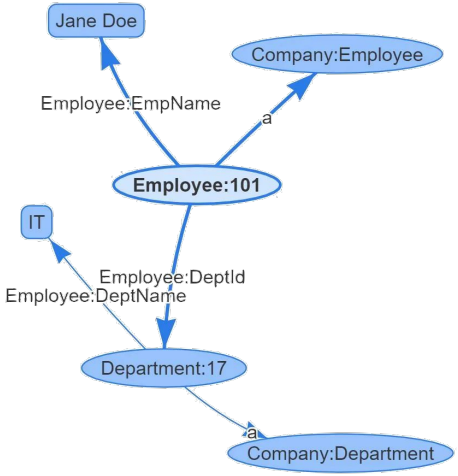
Meu 1o arquivo grafo RDF

* Employee

EmpId	EmpName	DeptId
101	Jane Doe	17
~~102~~	~~Graham Kerr~~	~~40~~
103	Sydney Green	17
~~104~~	~~Jason Morgan~~	~~40~~

* Department

DeptId	DeptName	DeptMngrId
17	IT	103
~~40~~	~~Accounting~~	~~104~~



Meu 1o arquivo RDF-Turtle (.ttl)

- Abra o ***Visual Studio Code***
- Instale as extensões:
  - ***Stardog RDF Grammars***
  - ***RDF Sketch***

Adobe Stock

Fundamentos básicos

slides adaptados de Tiago Sales,
inspirados no livro
The Web of Data de Aidan Hogan
https://aidanhogan.com/wodata/book.pdf

ERIES Escola Regional de Informática ES

RDFS: RDF Schema

- É uma extensão semântica de RDF que fornece mecanismos para descrever a estrutura e o significado dos dados RDF.
- Introduz um mínimo de semântica-lógica, permanecendo simples, especialmente em comparação com extensões mais complexas, como o OWL (Web Ontology Language).

RDF: Classes e Propriedades

- RDF já introduziu um reduzido **vocabulário de alto nível** – um conjunto de termos RDF – para favorecer a estruturação e interpretação dos dados.
- além da propriedade **rdf:type**, incorporada para denotar instâncias de classes, oferece também uma classe incorporada para capturar todas as propriedades.
- **rdf:Property** a classe de todas as propriedades.

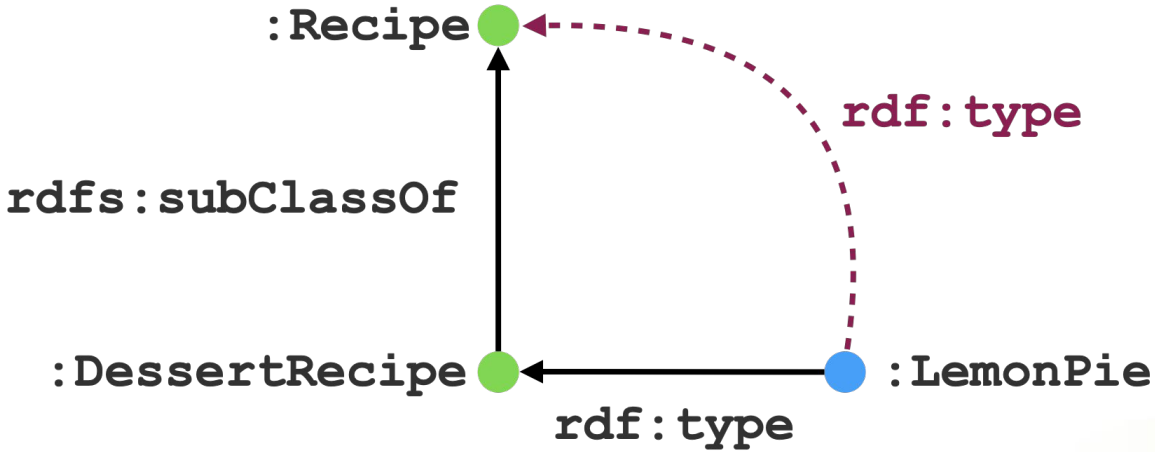
RDF: Classes e Propriedades

ex:LemonCheesecake	rdf:type	ex:DessertRecipe
ex:LemonCheesecake	ex:contains	ex:Lemon
ex:LemonCheesecake	ex:rating	"4.3"^^xsd:decimal
ex:Lemon	rdf:type	ex:Ingredient
ex:Lemon	rdf:type	ex:Fruit
ex:Lemon	rdfs:label	"Lemon"@en
ex:LemonCheesecake	ex:contains	ex:Cheese
ex:Cheese	rdf:type	ex:Ingredient
ex:Cheese	rdf:type	ex:Dairy
ex:LemonCheesecake	ex:contains	_:b
_:b	rdf:type	ex:Ingredient
_:b	rdf:type	ex:BakedGoods
ex:contains	rdf:type	rdf:Property
ex:rating	rdf:type	rdf:Property
ex:price	rdf:type	rdf:Property

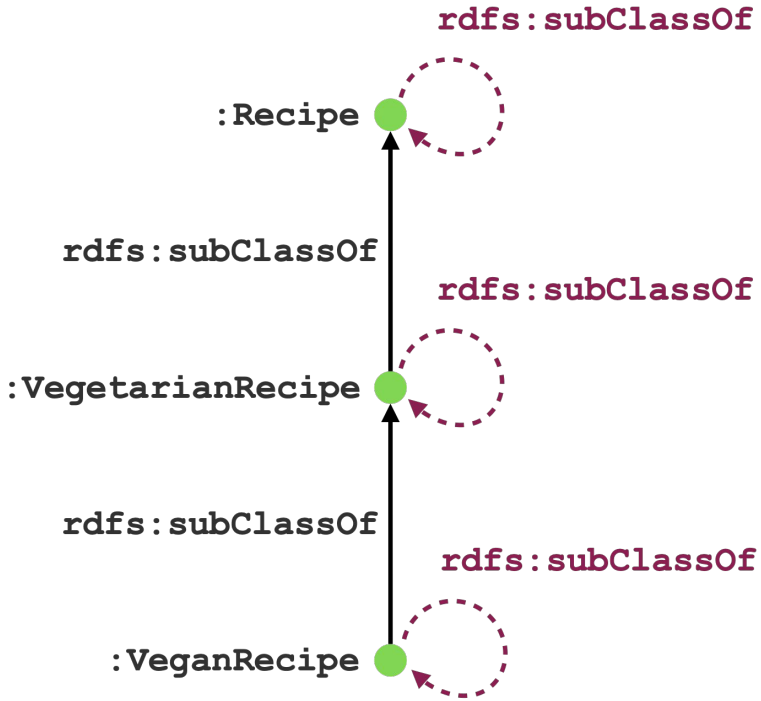
RDFS: Principais características

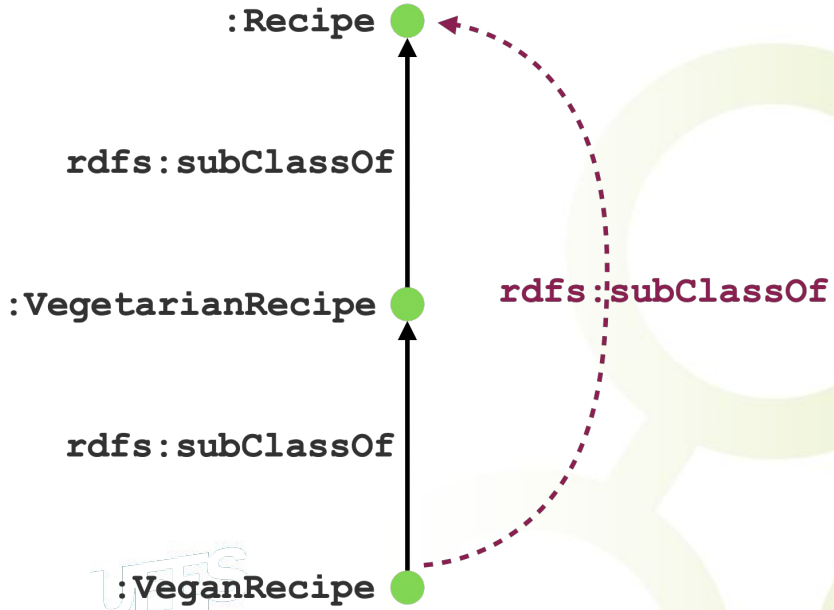
- **Classes e Subclasses:** Permite definir classes (tipos de recursos) e organizá-las em uma hierarquia usando **rdfs:subClassOf**.
  - Exemplo: "Cão" é subclasse de "Animal".
- **Propriedades e Subpropriedades**: Define propriedades (relações entre recursos) e permite criar hierarquias de propriedades usando **rdfs:subPropertyOf**.
  - Exemplo, "temIrmão" poderia ser uma subpropriedade de "relacionadoA".
- **Restrições de Domínio e Imagem**: Especifica os tipos de recursos aos quais uma propriedade pode se relacionar usando **rdfs:domain** e **rdfs:range**.
  - Exemplo, afirmando que a propriedade "temIdade" se aplica a "Pessoa" (domínio) e seus valores devem ser inteiros (imagem).
- **Rótulos e Comentários:** Suporta a adição de rótulos legíveis por humanos (**rdfs:label**) e comentários (**rdfs:comment**) aos recursos para melhor documentação.

rdfs:subClassOf

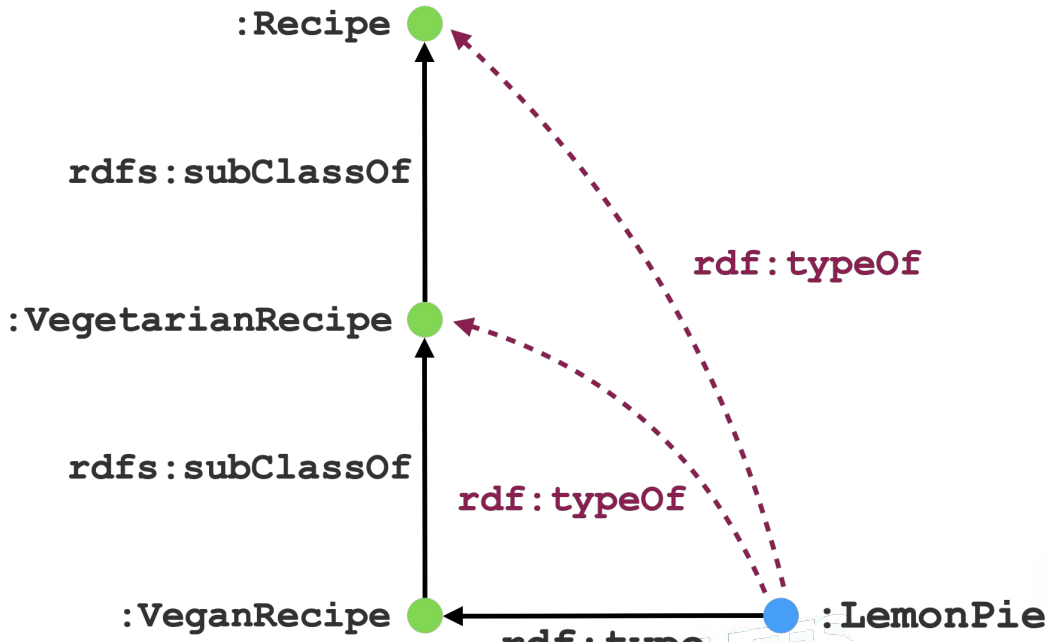


rdfs:subClassOf é reflexiva e transitiva

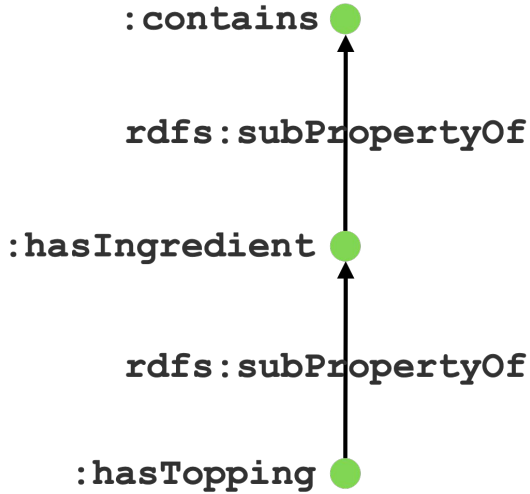


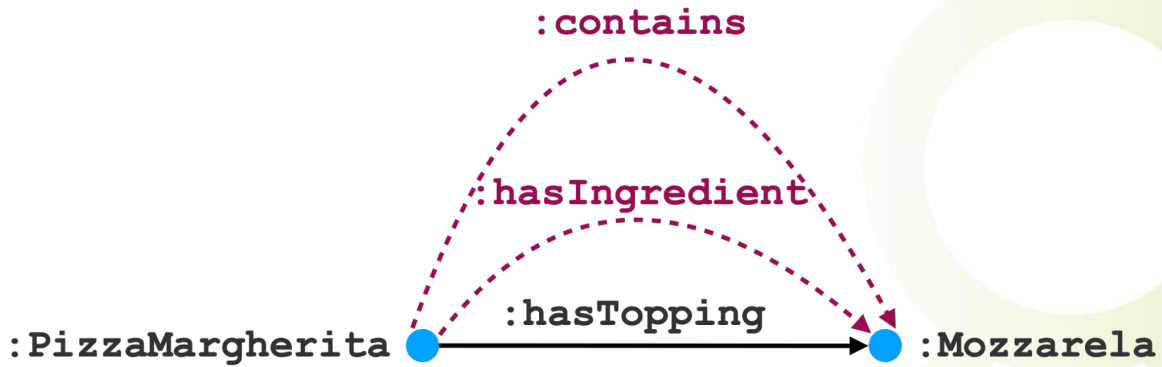


rdfs:subClassOf

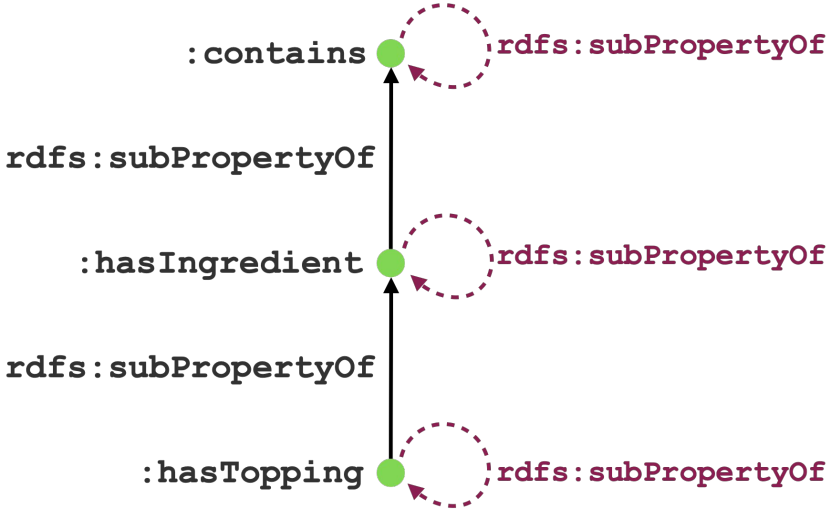


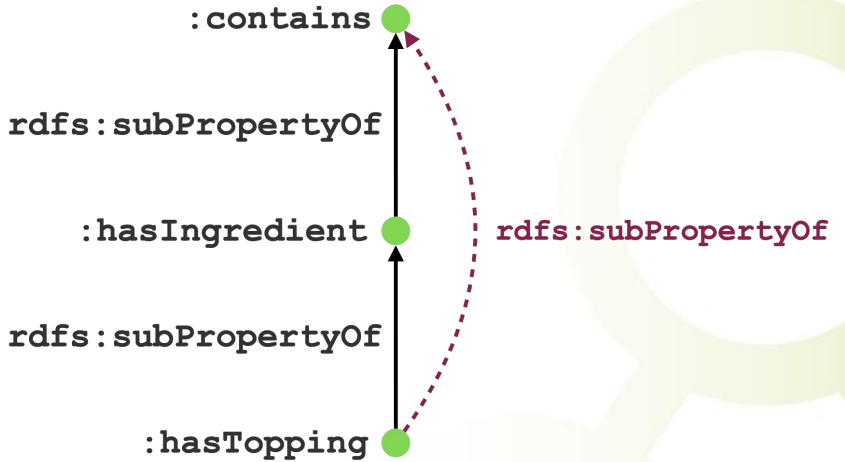
rdfs:subPropertyOf



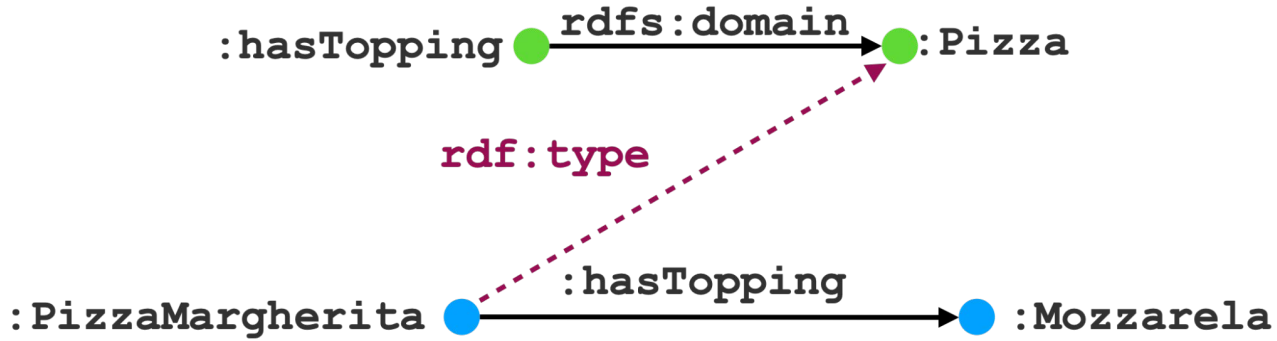


rdfs:subPropertyOf é reflexiva e transitiva

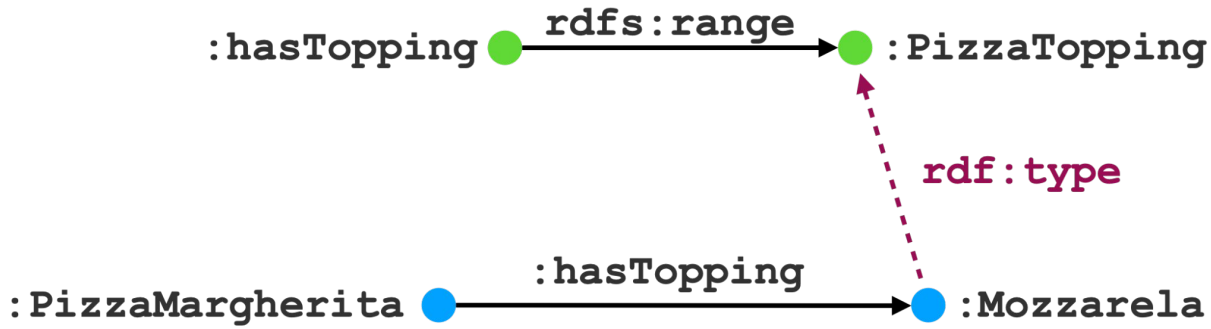




rdfs:domain


:hasTopping
rdfs:domain
:Pizza
rdf:type
:PizzaMargherita
:hasTopping
:Mozzarela

rdfs:range


:hasTopping
rdfs:range
:PizzaTopping
rdf:type
:PizzaMargherita
:hasTopping
:Mozzarela

rdfs:domain e rdfs:range - Perigo!

```
ex:hasBakingTime      rdfs:domain          ex:CakeRecipe
ex:hasBakingTime      rdfs:domain          ex:CasseroleRecipe
ex:hasBakingTime      rdfs:domain          ex:PieRecipe
```

```
ex:hasBakingTime      rdfs:domain          ex:OvenRecipe
ex:CakeRecipe         rdfs:subClassOf      ex:OvenRecipe
ex:CasseroleRecipe    rdfs:subClassOf      ex:OvenRecipe
ex:PieRecipe          rdfs:subClassOf      ex:OvenRecipe
```

RDFS: Anotações

- RDFS provê quatro propriedades de anotação que relacionam um recurso com … :
  - its name → **rdfs:label**
  - a comment describing it in more detail → **rdfs:comment**
  - a web location that contains information about it → **rdfs:seeAlso**
  - a web location that provides a definition of it → **rdfs:isDefinedBy**

RDF & RDFS

- RDF e RDFS fornecem um **vocabulário de alto nível** – um conjunto de termos – de **uso geral** para favorecer a estruturação e interpretação dos dados.
- **Vocabulários** de alto nível ou não podem ser **reutilizados** de maneira simples em diferentes fontes RDF independentes.
- Conjuntos de dados que concordam em seus vocabulários são (***supostamente***) mais fáceis de integrar, pois 'falam a mesma linguagem'.
- Por vezes são usados vocabulários mais ou menos "gerais" com propósitos similares aos termos de RDF, RDFS e também OWL, porém normalmente com uma semântica ligeiramente diferente.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

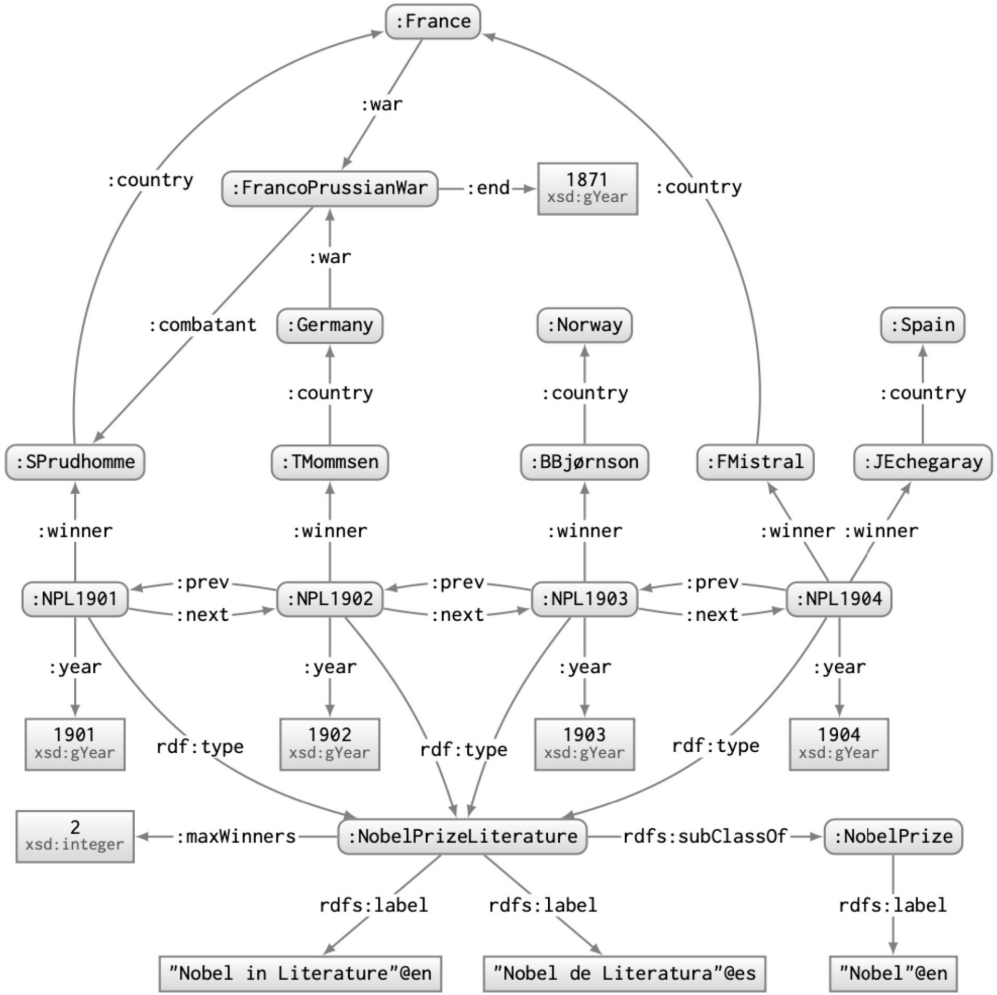
Fundamentos básicos

slides adaptados de Tiago Sales,
inspirados no livro
The Web of Data de Aidan Hogan
https://aidanhogan.com/wodata/book.pdf

ERIES Escola Regional de Informática ES

Como consultar um grafo RDF?

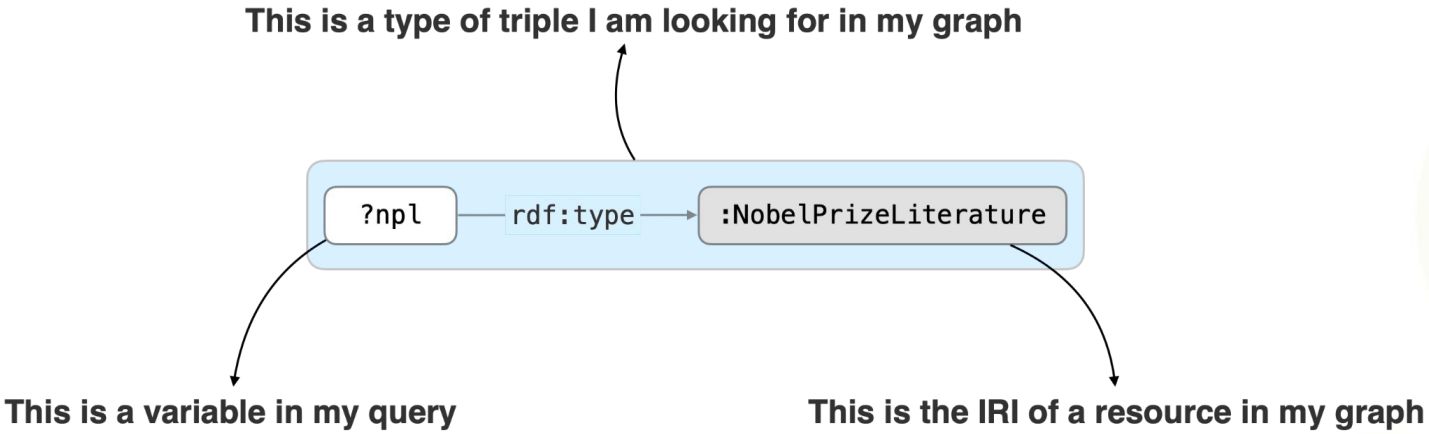
Buscando por "padrões" no grafo!

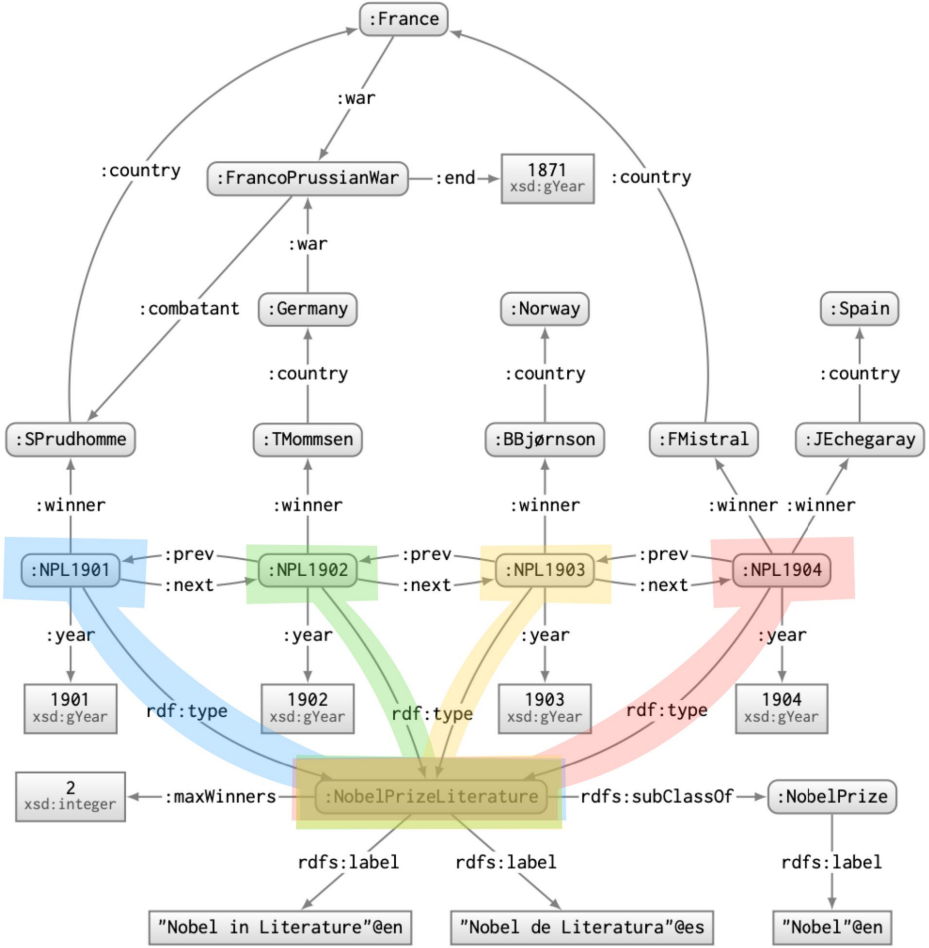


SPARQL
Caso de Uso

- Quem são os laureados com o Prêmio Nobel de Literatura que lutaram em guerras?
- Dê-me o IRI do laureado, o ano do prêmio e o ano em que a guerra terminou.

Quais Prêmios Nobel de Literatura foram concedidos?

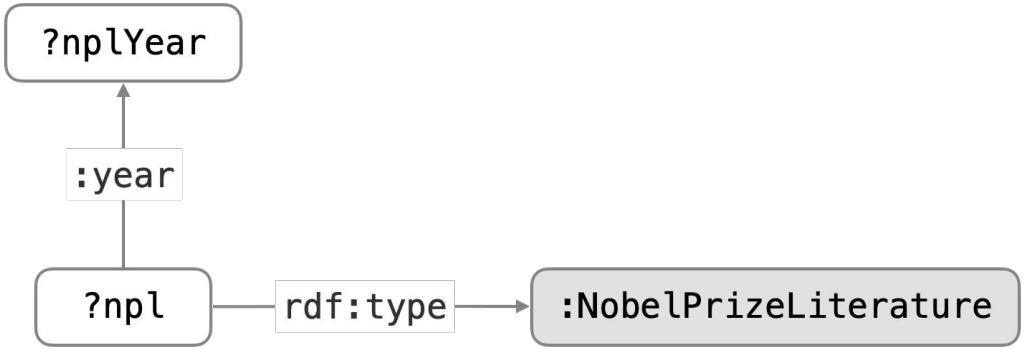


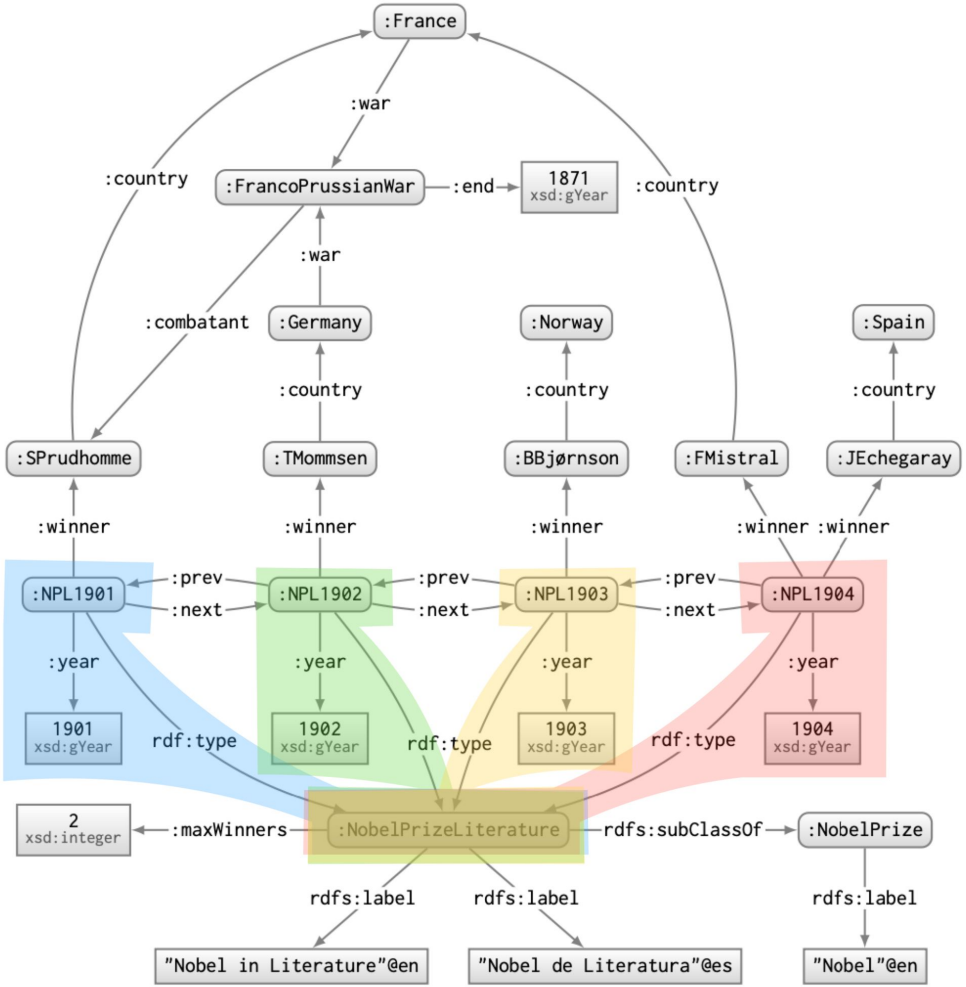


SPARQL
Caso de Uso

- Quais Prêmios Nobel de Literatura foram concedidos?
- 4 casamentos do padrão!!!

E em que ano foram concedidos?

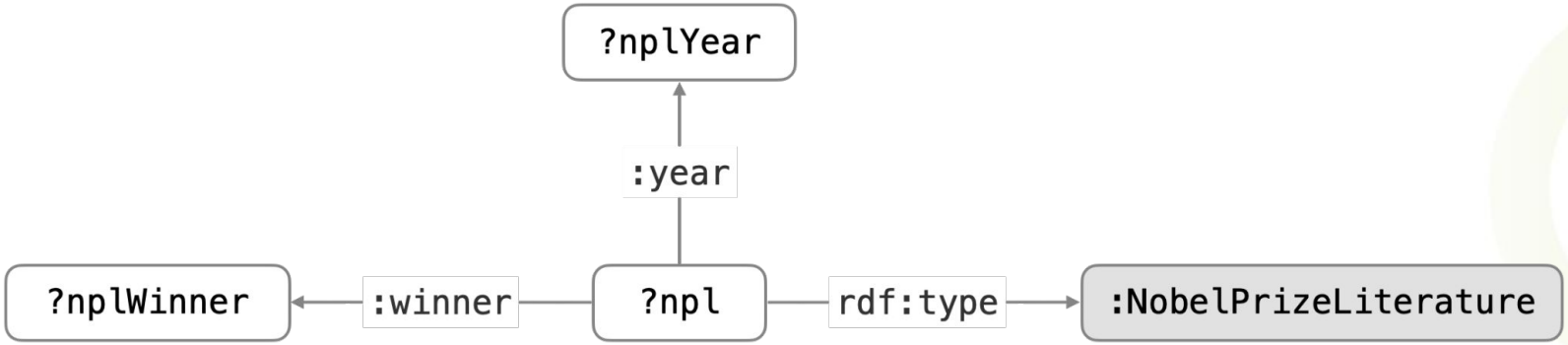


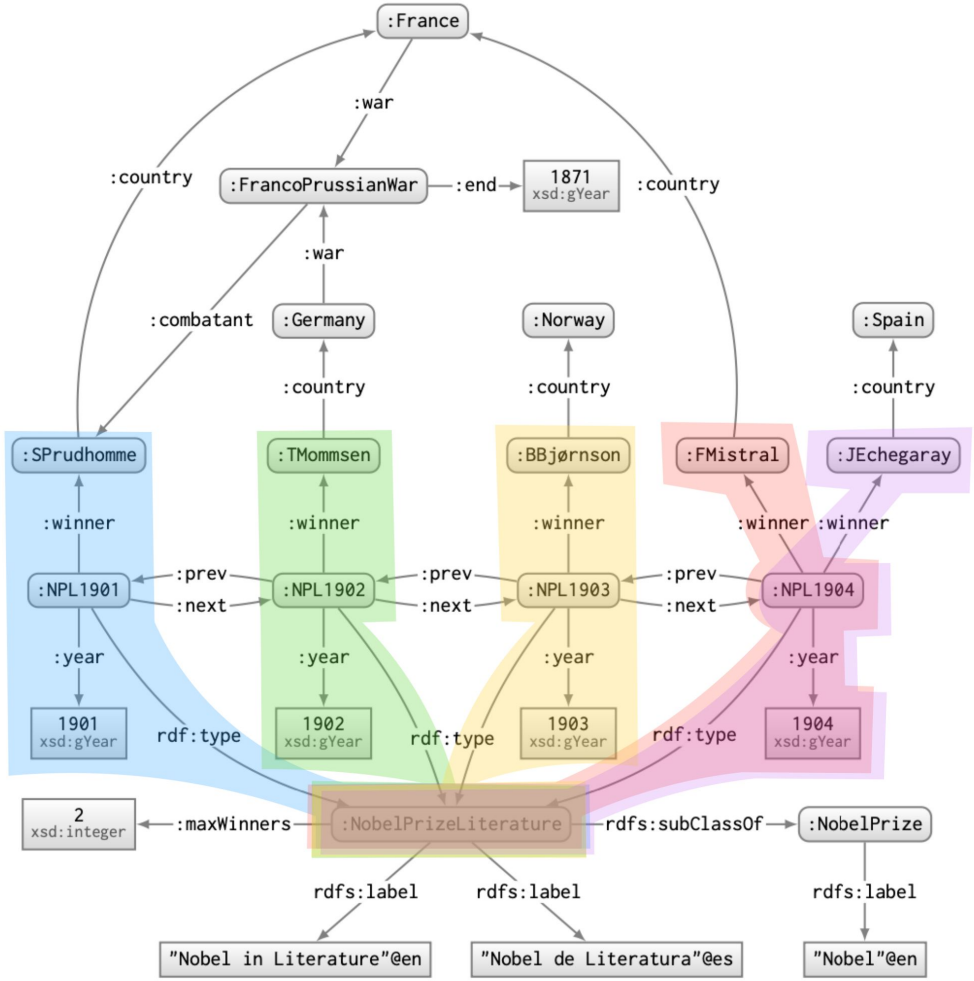


SPARQL
Caso de Uso

- Quais Prêmios Nobel de Literatura foram concedidos em que ano?
- 4 casamentos do padrão!!!

Quem foram os vencedores?

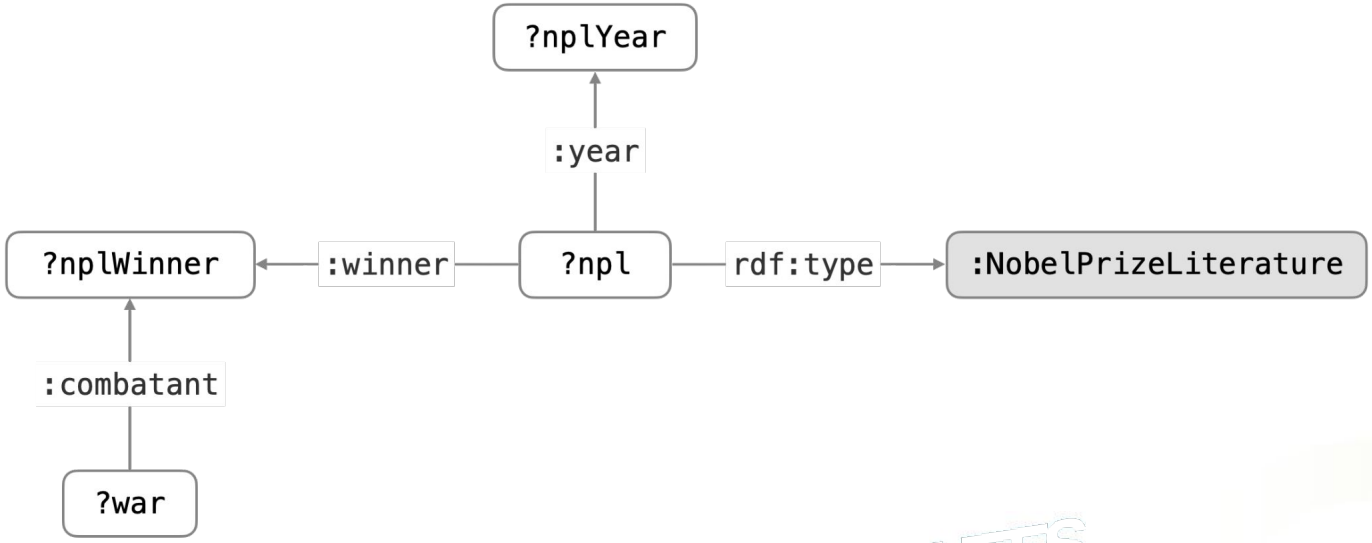


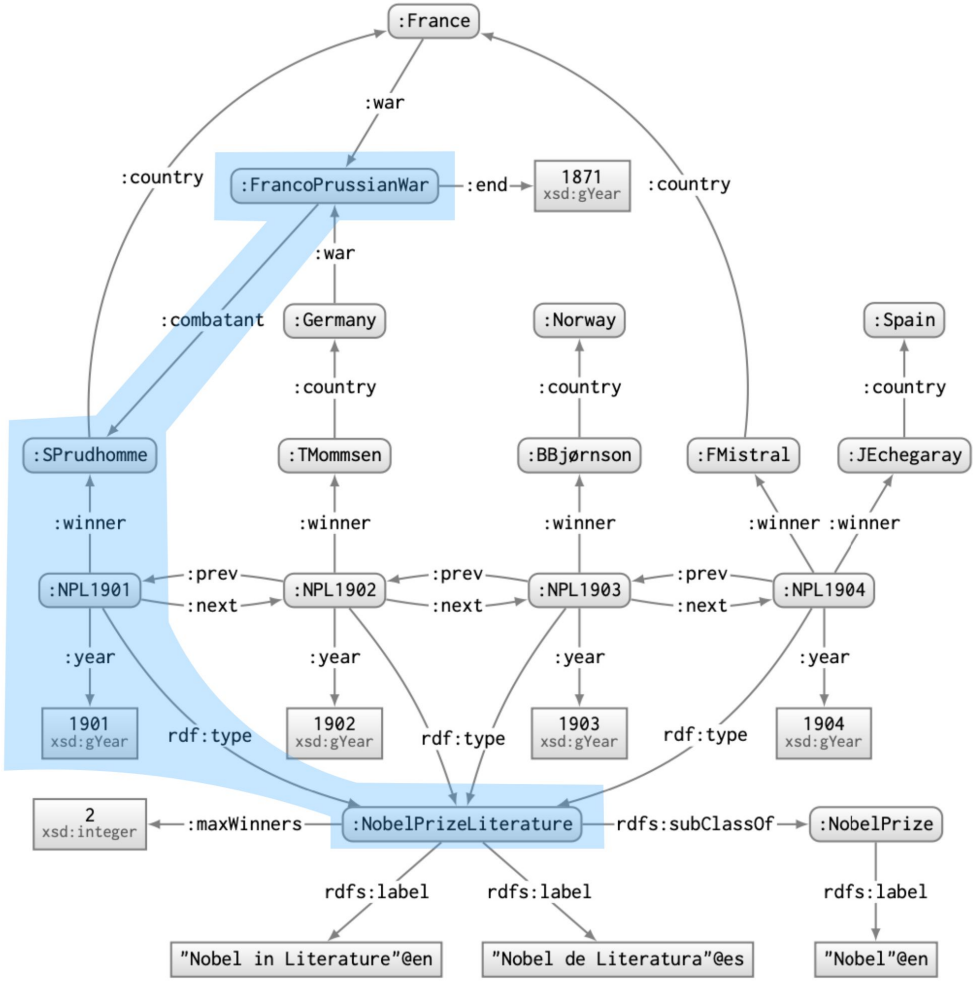


SPARQL
Caso de Uso

- Quem são os laureados com o Prêmio Nobel de Literatura, em que ano e quem foram os vencedores?
- **5 casamentos do padrão!!!**

Quem foram os vencedores que lutaram em uma guerra?

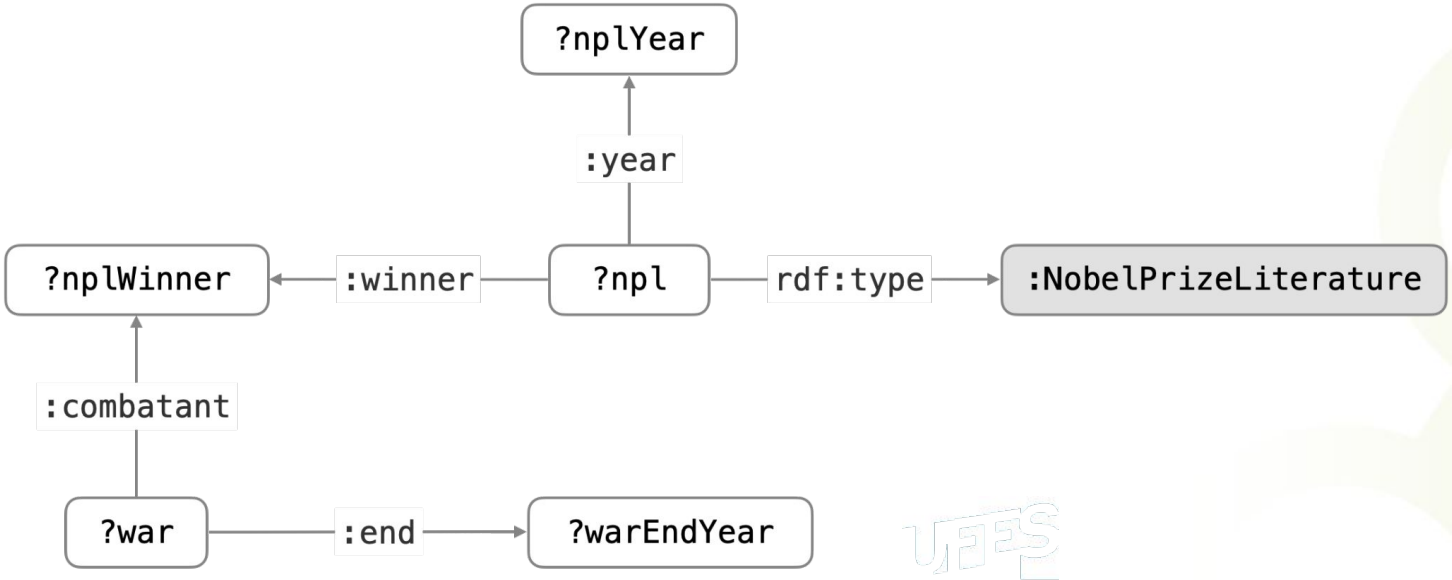


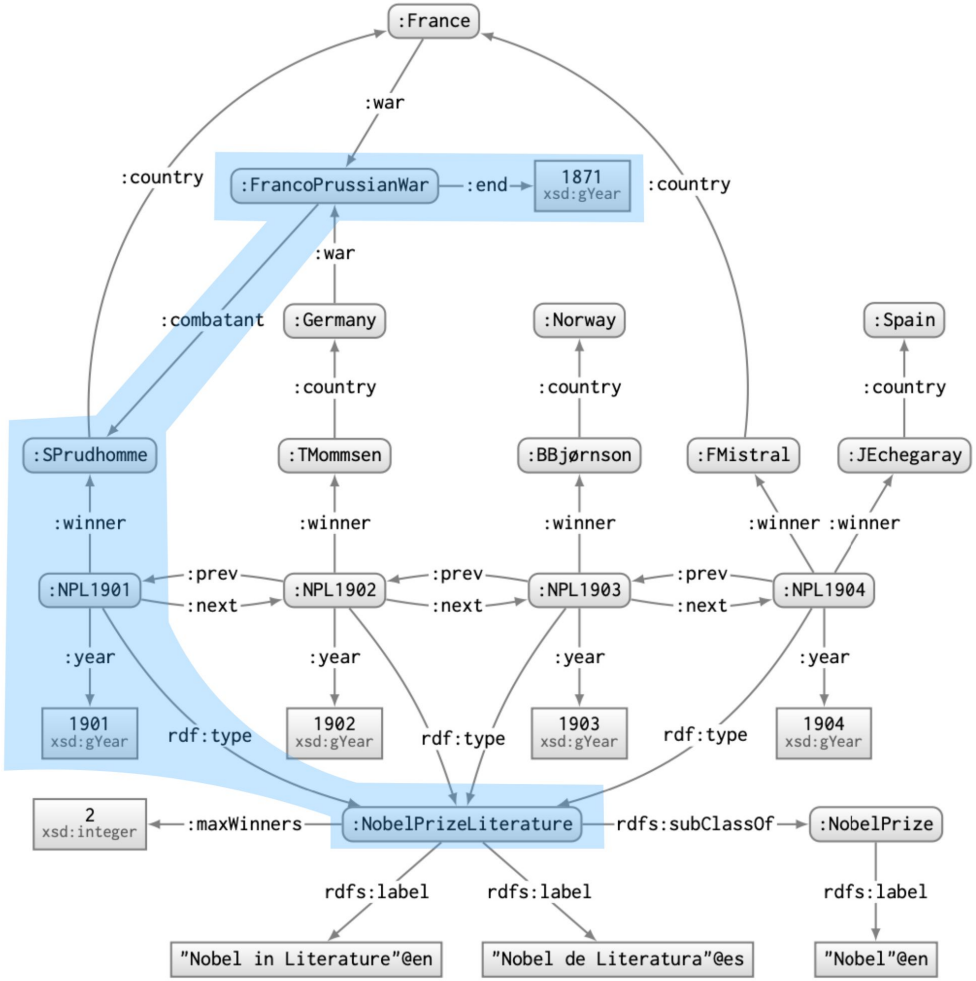


SPARQL
Caso de Uso

- Quem são os laureados com o Prêmio Nobel de Literatura, em que ano e quem foram os vencedores que lutaram em alguma guerra?
- **1 casamento do padrão!!!**

Quem foram os vencedores que lutaram em uma guerra e em que ano a guerra acabou?

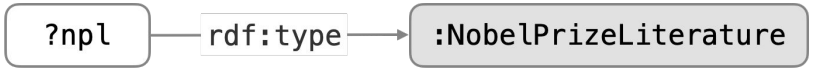




SPARQL
Caso de Uso

- Quem são os laureados com o Prêmio Nobel de Literatura, em que ano e quem foram os vencedores que lutaram em alguma guerra e em que ano a guerra acabou?
- **1 casamento do padrão!!!**

Padrão → SPARQL



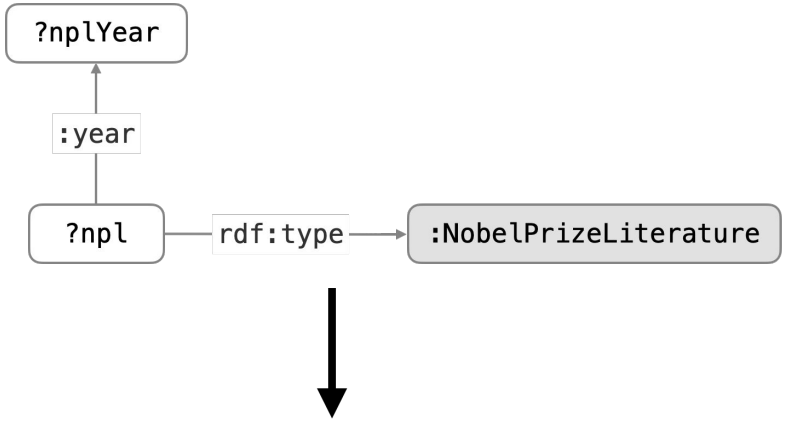
Just like in Turtle, we can add prefixes to make our query more compact

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

SELECT *
WHERE {
  ?npl a :NobelPrizeLiterature .
}
```

?npl
:NPL1901
:NPL1902
:NPL1903
:NPL1904

Padrão → SPARQL

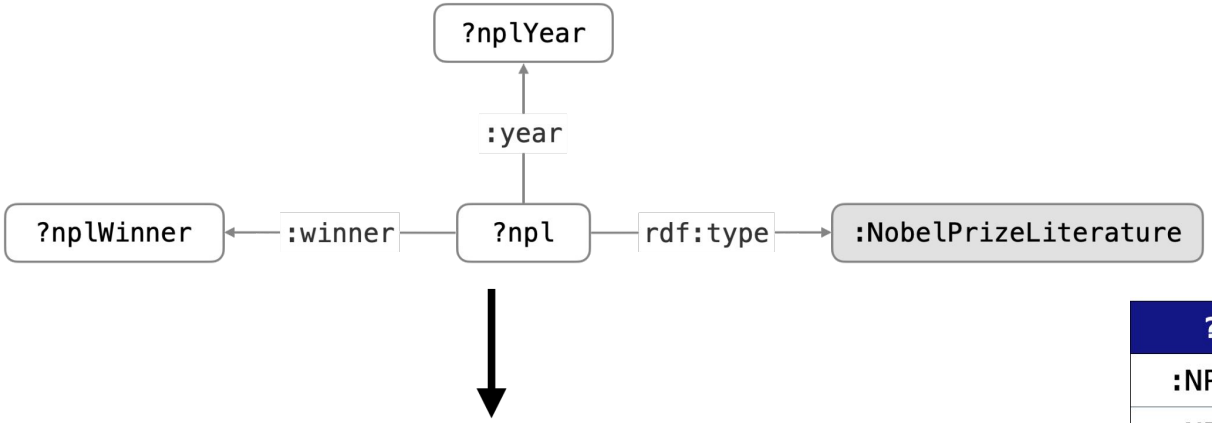



```
PREFIX : <http://example.org/>

SELECT *
WHERE {
  ?npl a :NobelPrizeLiterature .
  ?npl :year ?nplYear .
}
```

?npl	?nplYear
:NPL1901	"1901"^^xsd:gYear
:NPL1902	“1902"^^xsd:gYear
:NPL1903	“1903"^^xsd:gYear
:NPL1904	“1904"^^xsd:gYear

Padrão → SPARQL

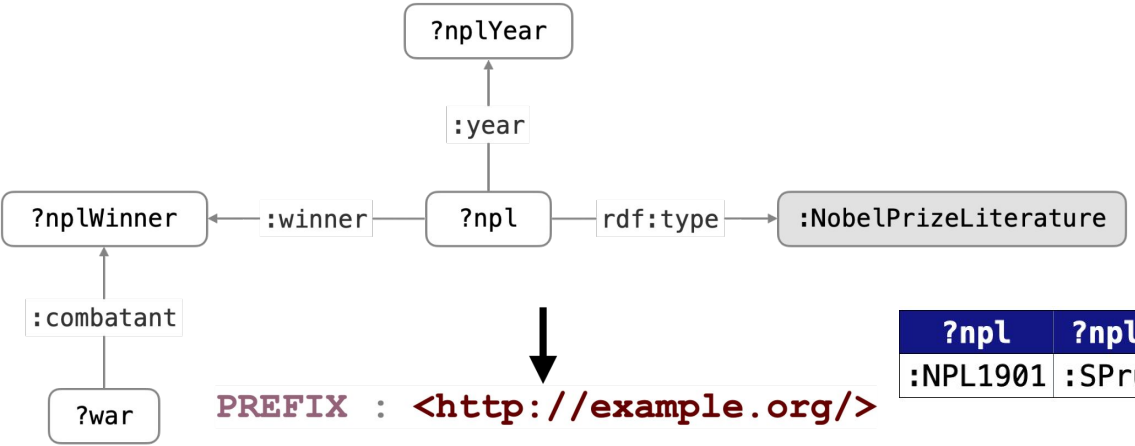



```
PREFIX : <http://example.org/>

SELECT *
WHERE {
   ?npl a :NobelPrizeLiterature .
   ?npl :year ?nplYear .
   ?npl :winner ?nplWinner .
}
```

?npl	?nplWinner	?nplYear
:NPL1901	:SPrudhomme	"1901"^^xsd:gYear
:NPL1902	:TMommsen	“1902"^^xsd:gYear
:NPL1903	:BBjornsøn	“1903"^^xsd:gYear
:NPL1904	:FMinstral	“1904"^^xsd:gYear
:NPL1904	:JEchegaray	“1904"^^xsd:gYear

Padrão → SPARQL

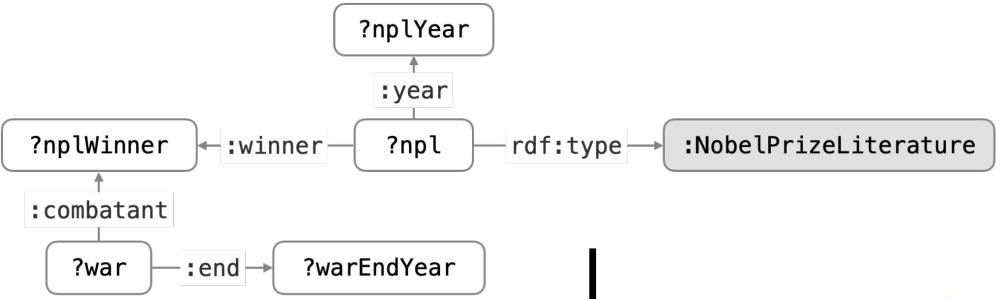



```
PREFIX : <http://example.org/>

SELECT *
WHERE {
   ?npl a :NobelPrizeLiterature .
   ?npl :year ?nplYear .
   ?npl :winner ?nplWinner .
   ?war :combatant ?nplWinner .
}
```

?npl	?nplWinner	?nplYear	?war
:NPL1901	:SPrudhomme	"1901"^^xsd:gYear	:FrancoPrussianWar

Padrão → SPARQL




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

SELECT ?nplWinner ?nplYear ?warEndYear
WHERE {
  ?npl a :NobelPrizeLiterature .
  ?npl :year ?nplYear .
  ?npl :winner ?nplWinner .
  ?war :combatant ?nplWinner .
  ?war :end ?warEndYear .
}
```

?nplWinner	?nplYear	?warEndYear
:SPrudhomme	"1901"^^xsd:gYear	"1871"^^xsd:gYear

Mãos nos dados

ERIES Escola Regional de Informática ES

Minha 1a consulta SPARQL

- Abra o site
  - https://yasgui.triply.cc/
- Aparecerá a seguinte query
-

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
 ?sub ?pred ?obj .
}
LIMIT 10
```

- Execute-a 
  - ***endpoint*** padrão: http://dbpedia.org/sparql

Esta consulta retorna as 10 primeiras "triplas" (`LIMIT 10`) encontradas no endpoint indicado (http://dbpedia.org/sparql), que satisfaçam o padrão indicado na consulta. Neste caso, (`?sub ?pred ?obj`) onde a ? indica variáveis que serão "casadas" com os valores das triplas que "casarem" com o padrão. Um ponto (.) deve ser usado ao final de cada padrão-tripla. Nesta consulta, qualquer tripla casará com este padrão, pois ele é "aberto", não especificando nenhuma parte da tripla, apenas variáveis.

Consultas SPARQL

A consulta pode ser mais específica ao fixar alguma parte do padrão. Esta retorna 100 recursos que são "instâncias de" Film (?sujeito [is] `a Film`) no vocabulario defindo em `http://dbpedia.org/ontology/`

```
SELECT DISTINCT *
WHERE { ?sujeito a <http://dbpedia.org/ontology/Film> .
} LIMIT 100
```

O nome "completo" pode ser substituído por um "abreviado", desde que um prefixo esteja definido: `PREFIX dbo: <http://…>`

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT *
WHERE { ?sujeito a dbo:Film .
} LIMIT 100
```

Todos os recursos devem ser um nome "completo" ou "abreviado". Exceção para `rdf:type` que, por ser muito usado, pode ser representado como "`a`" ("sugar syntax")

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT *
WHERE { ?sujeito rdf:type dbo:Film .
} LIMIT 100
```

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Consultas SPARQL

Combinando 2 padrões, esta consulta seleciona todos os recursos que são instâncias de Filme **E** que têm um título especificado via predicado `label` definido em http://www.w3.org/2000/01/rdf-schema#

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
select distinct *
where { ?sujeito a dbo:Film .
        ?sujeito rdfs:label ?titulo .
}
```

Em particular, alguns valores são definidos com tipos de dados específicos que podem ser filtrados. O idioma, em particular, pode ser filtrado usando `FILTER (lang(?titulo) = 'en')`, por exemplo, para resultados em língua inglesa.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
select distinct *
where { ?sujeito a dbo:Film .
        ?sujeito rdfs:label ?titulo .
        FILTER (lang(?titulo) = 'en')
} LIMIT 100
```

Consultas SPARQL

Quando o sujeito se repete em vários padrões, basta usar um ponto e vírgula (ao invés do ponto) para omiti-lo na linha seguinte.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dboW: <http://dbpedia.org/ontology/Work/>

SELECT DISTINCT *
WHERE { ?sujeito a dbo:Film ;
                rdfs:label ?titulo ;
                dboW:runtime ?tempo .
        FILTER (lang(?titulo) = 'en')
} LIMIT 100
```

Consultas SPARQL

As variáveis a serem retornadas podem ser especificadas no `SELECT`.
Elas também podem ser ordenadas na cláusula `ORDER BY`
Também existem funções que permitem fazer a conversão para um tipo de dado específico (`xsd:float()`). Esta pode ser feita usando um `BIND .. as` para associar o valor resultante a uma variável.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dboW: <http://dbpedia.org/ontology/Work/>

SELECT DISTINCT ?titulo ?minutos
WHERE { ?sujeito a dbo:Film ;
                 rdfs:label ?titulo ;
                 dboW:runtime ?tempo .
        BIND ( xsd:float(?tempo) as ?minutos)
        FILTER (lang(?titulo) = 'en')
}
ORDER BY DESC(?minutos) ?titulo
LIMIT 100
```

Consultas SPARQL

Esta consulta retorna 100 recursos do tipo filme que têm título em inglês e têm uma duração e um orçamento, ordenando primeiro pela maior duração, e depois em ordem alfabética.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dboW: <http://dbpedia.org/ontology/Work/>

SELECT DISTINCT ?titulo ?minutos ?orcamento
WHERE { ?sujeito a dbo:Film ;
                 rdfs:label ?titulo ;
                 dboW:runtime ?tempo ;
                 dbo:budget ?orcamento .
        BIND (xsd:float(?tempo) as ?minutos)
        FILTER (lang(?titulo) = 'en')
}
ORDER BY DESC(?minutos) ?titulo
LIMIT 100
```

Resultado

	titulo	minutos	budget
1	"Anchorman (film series)"@en	"1636.0"^^xsd:float	"7.6E7"^^<http://dbpedia.org/datatype/usDollar>
2	"The Clock (2010 film)"@en	"1440.0"^^xsd:float	"100000.0"^^<http://dbpedia.org/datatype/usDollar>
3	"Harry Potter (film series)"@en	"1179.0"^^xsd:float	"1.2E9"^^<http://dbpedia.org/datatype/usDollar>
4	"Puppet Master (film series)"@en	"1160.0"^^xsd:float	"4230000.0"^^<http://dbpedia.org/datatype/usDollar>
5	"The Diamond from the Sky"@en	"900.0"^^xsd:float	"800000.0"^^<http://dbpedia.org/datatype/usDollar>
6	"Berlin Alexanderplatz (miniseries)"@en	"894.0"^^xsd:float	"1.3E7"^^<http://dbpedia.org/datatype/usDollar>
7	"American Pie (film series)"@en	"788.0"^^xsd:float	"1.47E8"^^<http://dbpedia.org/datatype/usDollar>
8	"Mission: Impossible (film series)"@en	"768.0"^^xsd:float	"8.28E8"^^<http://dbpedia.org/datatype/usDollar>
9	"Pirates of the Caribbean (film series)"@en	"726.0"^^xsd:float	"1.274"^^<http://dbpedia.org/datatype/usDollar>
10	"The Million Dollar Mystery"@en	"690.0"^^xsd:float	"125000.0"^^<http://dbpedia.org/datatype/usDollar>

Showing 1 to 10 of 10 entries

Consultas SPARQL

Se ter um orçamento for uma característica opcional (nem todos têm), esse padrão em particular deverá ser usado dentro de uma cláusula `OPTIONAL ()`

Se houver duas ou mais características opcionais desconectadas, então várias cláusulas `OPTIONAL ()` devem ser usadas.

Ao colocar vários padrões em uma única cláusula `OPTIONAL ()`, então os resultados, mesmo opcionais, só serão retornados se todos forem encontrados (tudo ou nada).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dboW: <http://dbpedia.org/ontology/Work/>

SELECT DISTINCT ?titulo ?minutos ?orcamento
WHERE { ?sujeito a dbo:Film ;
                 rdfs:label ?titulo ;
                 dboW:runtime ?tempo .
        OPTIONAL {?sujeito dbo:budget ?orcamento}
        BIND (xsd:float(?tempo) as ?minutos)
        FILTER (lang(?titulo) = 'en')
}
ORDER BY DESC(?minutos) ?titulo
LIMIT 100
```

Consultas SPARQL

Se o valor para a variável título puder ser obtida a partir de duas propriedades, os padrões correspondentes deve estar conectados via uma cláusula `{} UNION {}` (ou mais de uma, se forem mais propriedades)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dboW: <http://dbpedia.org/ontology/Work/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?titulo ?minutos ?orcamento
WHERE { ?sujeito a dbo:Film ;
                 dboW:runtime ?tempo .
              {?sujeito rdfs:label ?titulo }
        UNION {?sujeito foaf:name ?titulo }
        OPTIONAL {?sujeito dbo:budget ?orcamento}
        BIND (xsd:float(?tempo) as ?minutos)
        FILTER (lang(?titulo) = 'en')
}
ORDER BY DESC(?minutos) ?titulo
LIMIT 100
```

Consultas SPARQL

Se quisermos também saber o nome dos diretores de cada filme, então precisamos primeiro, ligar cada filme a um diretor:

```
?sujeito dbo:director ?diretor
```

Depois buscar a(s) propriedade(s) de diretor que descreve(m) o seu nome:

```
?diretor rdfs:label ?nomeDir
```

Observem que se um filme tem mais de um diretor, ou se há mais de um nome por diretor, haverá "repetições" no resultado (um filme aparecerá várias vezes, uma para cada diretor e para cada nome de diretor.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dboW: <http://dbpedia.org/ontology/Work/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?titulo ?minutos ?orcamento ?nomeDir
WHERE { ?sujeito a dbo:Film ;
                 dboW:runtime ?tempo ;
                 dbo:director ?diretor .
              {?sujeito rdfs:label ?titulo }
        UNION {?sujeito foaf:name ?titulo }
        OPTIONAL {?sujeito dbo:budget ?orcamento}
        ?diretor rdfs:label ?nomeDir
        BIND (xsd:float(?tempo) as ?minutos)
        FILTER (lang(?titulo) = 'en')
}
ORDER BY DESC(?minutos) ?titulo
LIMIT 100
```

Atividade

Vamos escolher um outro tema e modificar as queries para realizar consultas semelhantes.

1. Busquem na wikipedia por um tema de interesse: livros, filmes, carros, jogos
2. Tentem modificar as queries para o seu tema escolhido (difícil conseguir de primeira, mas a tentativa, por mais que seja frustrada, vai ajudar no aprendizado)

Adobe Stock

Discussões

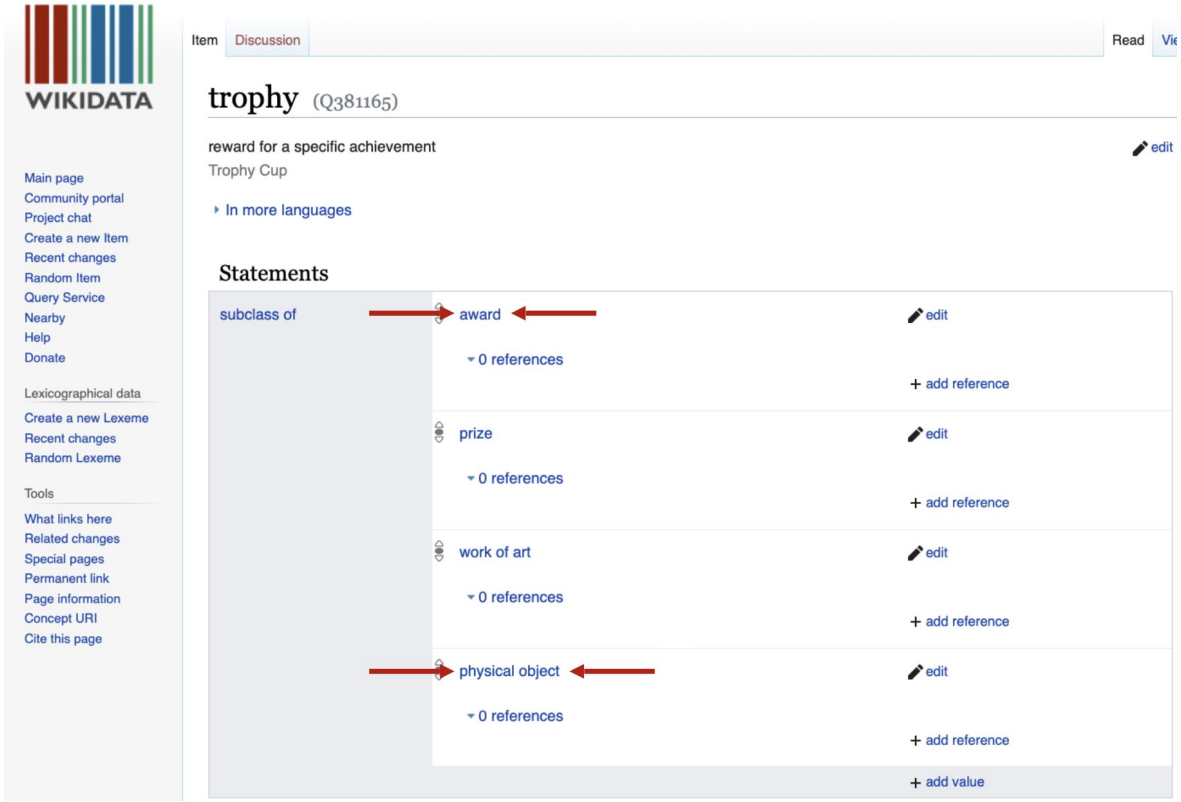
slides adaptados de Tiago Sales, Giancarlo Guizzardi e João Paulo Almeida

Problemas:

1. Definição "fraca" de vocabulários/ontologias
2. Muita liberdade na instanciação desses vocabulários

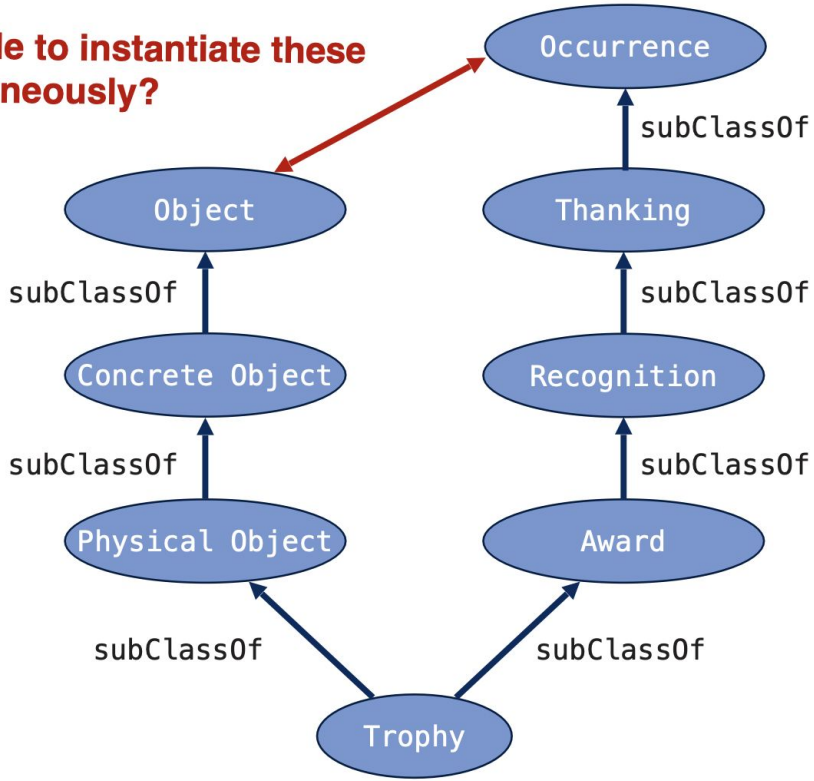
O problema não está na tecnologia, mas no seu (mal) uso!

Problemas:


WIKIDATA
Item
Discussion
Read
trophy (Q381165)
reward for a specific achievement
Trophy Cup
edit
In more languages
Statements
subclass of
award
edit
0 references
add reference
prize
edit
0 references
add reference
work of art
edit
0 references
add reference
physical object
edit
0 references
add reference
add value
Main page
Community portal
Project chat
Create a new Item
Recent changes
Random Item
Query Service
Nearby
Help
Donate
Lexicographical data
Create a new Lexeme
Recent changes
Random Lexeme
Tools
What links here
Related changes
Special pages
Permanent link
Page information
Concept URI
Cite this page

Problemas:

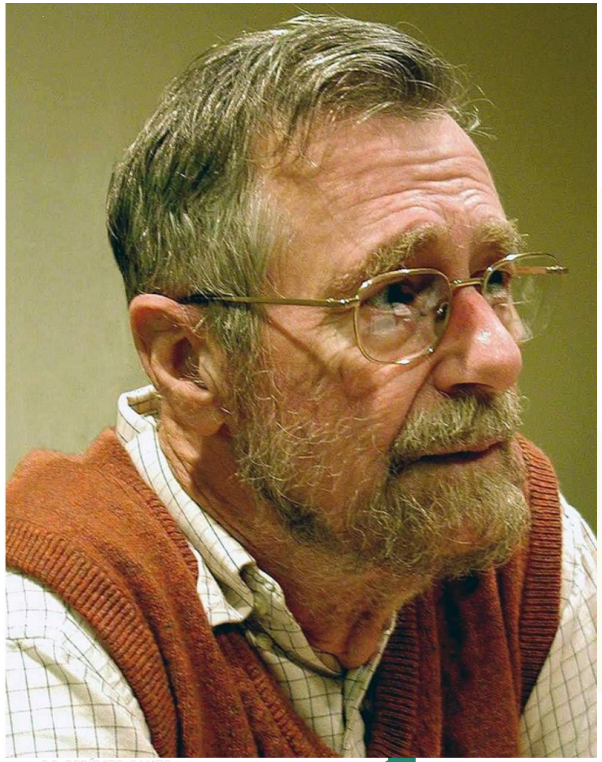
Should it be possible to instantiate these two classes simultaneously?



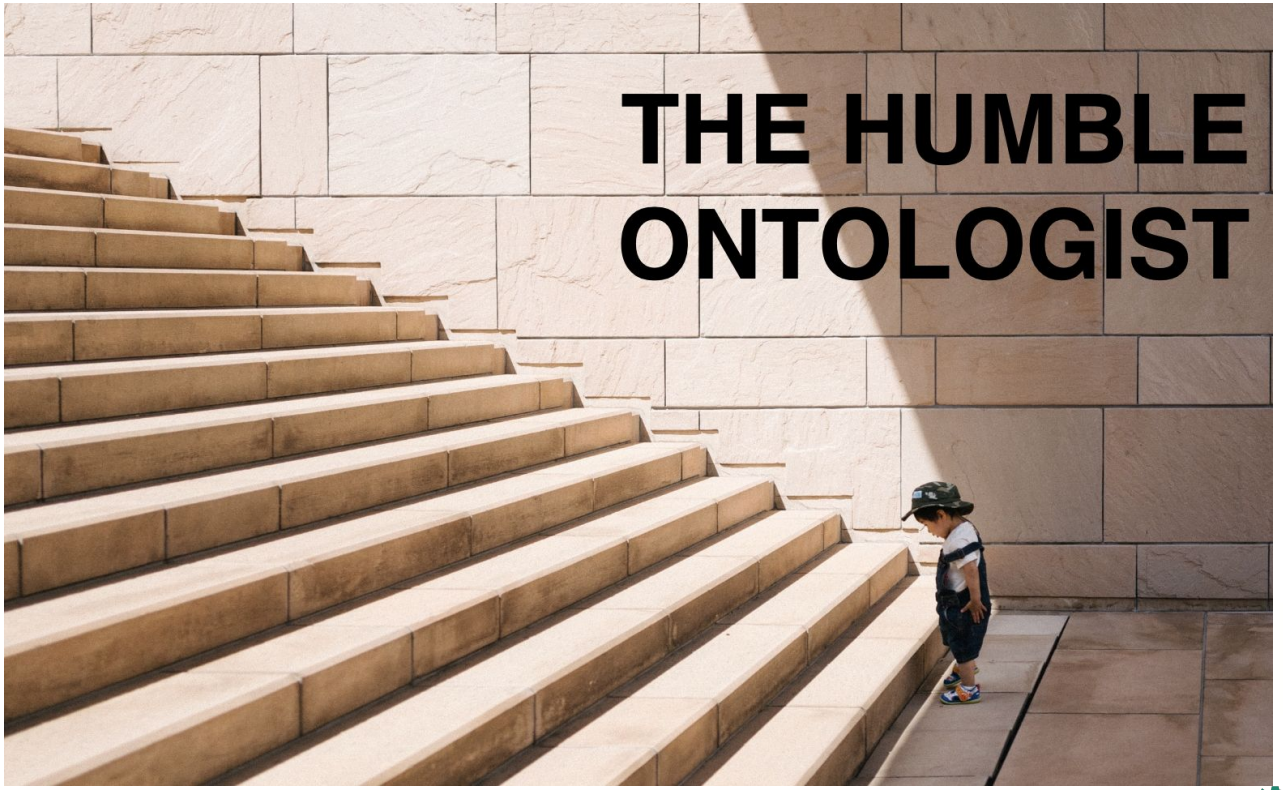
Problema Análogo:

"We shall do a much better programming job, provided that we approach the task with a **full appreciation of its tremendous difficulty**, […] we stick to modest and elegant programming languages, […] we respect the **intrinsic limitations of the human mind** and approach the task as Very Humble Programmers."

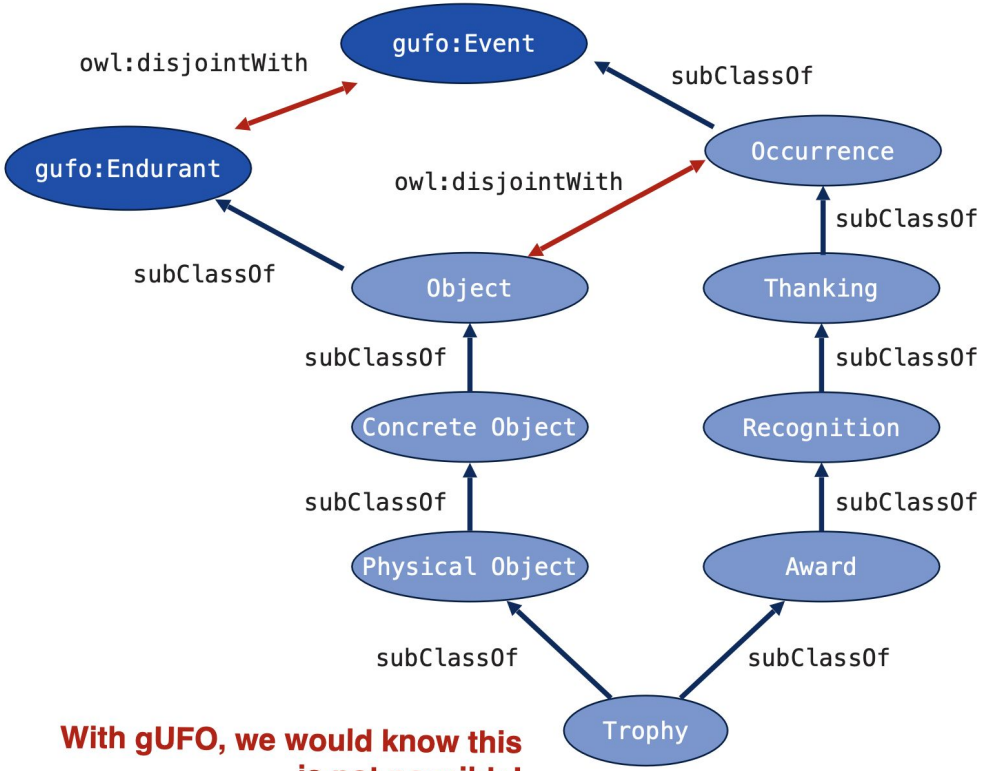
Edsger W. Dijkstra (1972). "The Humble Programmer", ACM Turing Award Lecture



Solução Análoga:



Solução:


gufo:Event
owl:disjointWith
subClassOf
gufo:Endurant
Occurrence
owl:disjointWith
subClassOf
subClassOf
Object
Thanking
subClassOf
subClassOf
Concrete Object
Recognition
subClassOf
subClassOf
Physical Object
Award
subClassOf
subClassOf
Trophy
With gUFO, we would know this is not possible!

Evolução da Internet Web

https://tinyurl.com/4e8h9s72

Extras

Como usar com outras linguagens?

1. Baixar o resultado de uma query em csv
2. Rodar queries em um endpoint (local ou externo) usando bibliotecas das linguagens, exemplos
   a. **SWI-Prolog:** *semweb package* -> *fatos*
   b. **Python:** *rdflib e outras* -> *listas*
   c. ***Java, Javascript, R, Ruby e outras***

Dados Ligados na Web

Escola Regional de Informática ES

OBRIGADA!!

Veruska Zamborlini (veruska.zamborlini@ufes.br)

Núcleo de Estudos em Modelagem Conceitual e Ontologias - NEMO

Departamento de Informática

Centro Tecnológico

Universidade Federal do Espírito Santo

nemo

UFES UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO