Engineering of ontologies with Description Logics

4. semantic technologies – representating and querying knowledge

Nicolas Troquard

We are going to use:

- Protégé: install software `https://protege.stanford.edu/`
- Ontop: install (Docker `https://www.docker.com/` and) the docker instance from `https://bitbucket.org/troquard/ontop-aquila`
- DBpedia SPARQL endpoint: `https://dbpedia.org/sparql` (no instalation needed)

This is a very partial guide and tutorial about languages and technolologies for semantic science.

- Writing and querying ontologies
- Languages for writing ontologies: RDF(S), OWL
- Languages for querying ontologies: DL, SPARQL
- Ontology editing environment: Protégé
- OBDA, and the VKG system Ontop

Outline

Mutually defined vocabularies

- RDF `http://www.w3.org/1999/02/22-rdf-syntax-ns`:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
```

- RDFS `https://www.w3.org/2000/01/rdf-schema`:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
```

- OWL `https://www.w3.org/2002/07/owl`:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix grddl: <http://www.w3.org/2003/g/data-view#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

RDF

```
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> a owl:Ontology ;
dc:title "The RDF Concepts Vocabulary (RDF)" ;
dc:date "2019-12-16" ;
dc:description "This is the RDF Schema for the RDF vocabulary terms in the RDF Namespace, defined in RDF 1.1 Concepts." .

...
```

RDFS: Classes

```
<http://www.w3.org/2000/01/rdf-schema#> a owl:Ontology ;
dc:title "The RDF Schema vocabulary (RDFS)" .



rdfs:Resource a rdfs:Class ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "Resource" ;
rdfs:comment "The class resource, everything." .

rdfs:Class a rdfs:Class ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "Class" ;
rdfs:comment "The class of classes." ;
rdfs:subClassOf rdfs:Resource .

rdfs:Literal a rdfs:Class ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "Literal" ;
rdfs:comment "The class of literal values, eg. textual strings and integers." ;
rdfs:subClassOf rdfs:Resource .

rdfs:Datatype a rdfs:Class ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "Datatype" ;
rdfs:comment "The class of RDF datatypes." ;
rdfs:subClassOf rdfs:Class .

...
```

RDFS: Properties

```
rdfs:subClassOf a rdf:Property ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "subClassOf" ;
rdfs:comment "The subject is a subclass of a class." ;
rdfs:range rdfs:Class ;
rdfs:domain rdfs:Class .

rdfs:subPropertyOf a rdf:Property ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "subPropertyOf" ;
rdfs:comment "The subject is a subproperty of a property." ;
rdfs:range rdf:Property ;
rdfs:domain rdf:Property .

rdfs:comment a rdf:Property ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "comment" ;
rdfs:comment "A description of the subject resource." ;
rdfs:domain rdfs:Resource ;
rdfs:range rdfs:Literal .

rdfs:label a rdf:Property ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "label" ;
rdfs:comment "A human-readable name for the subject." ;
rdfs:domain rdfs:Resource ;
rdfs:range rdfs:Literal .

...
```

OWL RDF vocabulary

https://www.w3.org/TR/owl-ref/

Concepts (owl:Class) and roles (owl:ObjectProperties) are rdfs:Class. And so is owl:Ontology.

```
owl:Class a rdfs:Class ;
     rdfs:label "Class" ;
     rdfs:comment "The class of OWL classes." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf rdfs:Class .

owl:ObjectProperty a rdfs:Class ;
     rdfs:label "ObjectProperty" ;
     rdfs:comment "The class of object properties." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf rdf:Property .

owl:Ontology a rdfs:Class ;
     rdfs:label "Ontology" ;
     rdfs:comment "The class of ontologies." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf rdfs:Resource .
```

...

OWL RDF vocabulary

The $\top$ top concept `owl:Thing` and the $\bot$ bottom concept `owl:Nothing` are OWL classes.

```
owl:Thing a owl:Class ;
     rdfs:label "Thing" ;
     rdfs:comment "The class of OWL individuals." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> .

owl:Nothing a owl:Class ;
     rdfs:label "Nothing" ;
     rdfs:comment "This is the empty class." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf owl:Thing .
```

...

OWL RDF vocabulary

```
owl:complementOf a rdf:Property ;
     rdfs:label "complementOf" ;
     rdfs:comment "The property that determines that a given class is the complement of another class." ;
     rdfs:domain owl:Class ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:range owl:Class .

owl:unionOf a rdf:Property ;
     rdfs:label "unionOf" ;
     rdfs:comment "The property that determines the collection of classes or data ranges that build a union." ;
     rdfs:domain rdfs:Class ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:range rdf:List .

owl:Restriction a rdfs:Class ;
     rdfs:label "Restriction" ;
     rdfs:comment "The class of property restrictions." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf owl:Class .

owl:TransitiveProperty a rdfs:Class ;
     rdfs:label "TransitiveProperty" ;
     rdfs:comment "The class of transitive properties." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf owl:ObjectProperty .

owl:FunctionalProperty a rdfs:Class ;
     rdfs:label "FunctionalProperty" ;
     rdfs:comment "The class of functional properties." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf rdf:Property .

owl:allValuesFrom a rdf:Property ;
     rdfs:label "allValuesFrom" ;
     rdfs:comment "The property that determines the class that a universal property restriction refers to." ;
     rdfs:domain owl:Restriction ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:range rdfs:Class .
```

OWL RDF vocabulary

```
owl:DatatypeProperty a rdfs:Class ;
     rdfs:label "DatatypeProperty" ;
     rdfs:comment "The class of data properties." ;
     rdfs:isDefinedBy <http://www.w3.org/2002/07/owl#> ;
     rdfs:subClassOf rdf:Property .

...
```

Functional-style syntax for OWL ontologies (1)

`https://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/`

Some concepts (Class Expressions):

OWL functional	RDF	DL
ObjectIntersectionOf(C1 ... Cn)	`x rdf:type owl:Class. x owl:intersectionOf ( C1 .. Cn )`	$C_1 \sqcap \ldots \sqcap C_n$
ObjectUnionOf(C1 .. Cn)	`x owl:unionOf ( C1 .. Cn )`	$C_1 \sqcup \ldots \sqcup C_n$
ObjectComplementOf(C)	`x owl:complementOf C`	$\neg C$
ObjectOneOf(a1 .. an)	`x owl:oneOf ( a1 .. an )`	$\{a_1\} \sqcup \ldots \sqcup \{a_n\}$
ObjectAllValuesFrom(P C)	`x rdf:type owl:Restriction. x owl:onProperty P. x owl:allValuesFrom C`	$\forall P.C$
ObjectSomeValuesFrom(P C)	`x owl:someValuesFrom C`	$\exists P.C$
ObjectExactCardinality(n P C)	`x owl:qualifiedCardinality n. x owl:onClass C`	$(= n\ P.C)$
ObjectMaxCardinality(n P C)	`x owl:maxQualifiedCardinality n. x owl:onClass C`	$(\leq n\ P.C)$
ObjectMinCardinality(n P C)	`x owl:minQualifiedCardinality n. x owl:onClass C`	$(\geq n\ P.C)$
...		

Functional-style syntax for OWL ontologies (2)

Some axioms:

OWL functional	RDF(S)	DL
ClassAssertion(C a)	`a rdf:type C`	$C(a)$
ObjectPropertyAssertion(PN a1 a2)	`a1 PN a2`	$PN(a_1, a_2)$
NegativeObjectPropertyAssertion(P a1 a2)	`x rdf:type owl:NegativePropertyAssertion. x owl:sourceIndividual a1. x owl:assertionProperty P. x owl:targetIndividual a2`	$\neg PN(a_1, a_2)$
SameIndividual(a1 .. an)	`aj owl:sameAs aj+1. j=1..n-1`	$a_j = a_{j+1}, 1 \leq i < n$
DifferentIndividuals(a1 a2)	`a1 owl:differentFrom a2`	$a_1 \neq a_2$
SubClassOf(C1 C2)	`C1 rdfs:subClassOf C2`	$C_1 \sqsubseteq C_2$
EquivalentClasses(C1 ... Cn)	`Cj owl:equivalentClass Cj+1. j=1..n-1`	$C_j \equiv C_{j+1}, 1 \leq j < n$
DisjointClasses(C1 C2)	`C1 owl:disjointWith C2`	$C_1 \sqsubseteq \neg C_2$
DisjointUnionOf(CN C1 .. Cn)	`CN owl:disjointUnionOf ( C1 .. Cn ).`	$CN \equiv \sqcup_{1 \leq j \leq n} C_j;\ C_j \sqsubseteq \neg C_{j+i}, 1 \leq j < n$
...		
FunctionalObjectProperty(P)	`P rdf:type owl:FunctionalProperty`	$\top \sqsubseteq (\leq 1\, R.\top)$
SymmetricObjectProperty(P)	`P rdf:type owl:SymmetricProperty`	$P^- \sqsubseteq P$
TransitiveObjectProperty(P)	`P rdf:type owl:TransitiveProperty`	$P \circ P \sqsubseteq P$
SubDataPropertyOf(R1 R2)	`R1 rdfs:subPropertyOf R2`	$R_1 \sqsubseteq R_2$
DataPropertyDomain(R C)	`R rdfs:domain C`	$\exists R.\top \sqsubseteq C$
DataPropertyRange(R D)	`R rdfs:range D`	$\exists R^-.\top \sqsubseteq C$
...		

OWL 2 profiles

`https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/`

Different profiles that correspond to different Description Logics.

profile	$\mathcal{DL}$	why?
OWL 2 DL	$\mathcal{SROIQ}(\mathcal{D})$	expressivity and decidable reasoning
OWL 2 EL	$\mathcal{EL}^{++}$	reasoning in PTIME
OWL 2 QL	$\mathcal{DL}$-Lite$_{\mathcal{R}}$	query answering in AC^0 wrt. data
OWL 2 RL	$\mathcal{RL}$	rule-based reasoning

Outline

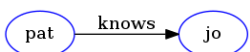
See https://www.w3.org/2000/10/swap/Primer (Subject, verb and object)

RDF triples

A collection of statements, each with:

- a subject,
- a verb,
- an object.

```
<#pat> <#knows> <#jo> .
```



Namespaces:
http://www.ldf.fi/service/rdf-grapher/#

The object can be a literal.

```
<#pat> <#knows> <#jo> .
<#pat> <#age> 24 .
```

Namespaces:
http://www.ldf.fi/service/rdf-grapher/#

Literals (1)

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

Core types:

xsd:string	Character strings (but not all Unicode character strings)
xsd:boolean	true, false
xsd:decimal	Arbitrary-precision decimal numbers
xsd:integer	Arbitrary-size integer numbers

Floating-point:

xsd:double	64-bit floating point numbers incl. ±Inf, ±0, NaN
xsd:float	32-bit floating point numbers incl. ±Inf, ±0, NaN

Time and date:

xsd:date	Dates (yyyy-mm-dd) with or without timezone
xsd:time	Times (hh:mm:ss.sss...) with or without timezone
xsd:dateTime	Date and time with or without timezone
xsd:dateTimeStamp	Date and time with required timezone

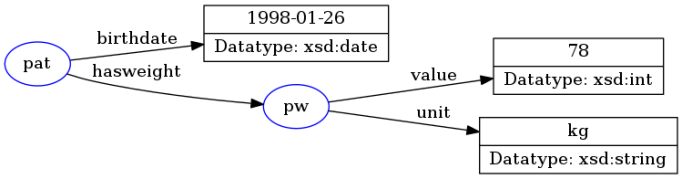
And more:

xsd:duration	Duration of time
xsd:byte	-128...+127 (8 bit)
xsd:int	-2147483648...+2147483647 (32 bit)
...	...

Literals (2)

```
@prefix : <#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#pat> <#birthdate> "1998-01-26"^^xsd:date .
<#pat> <#hasweight> <#pw> .
<#pw> <#value> "78"^^xsd:int .
<#pw> <#unit> "kg"^^xsd:string .
```



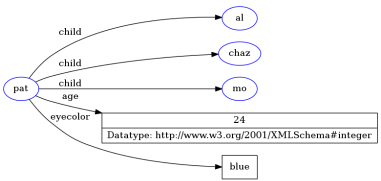
Namespaces:
http://www.ldf.fi/service/rdf-grapher/#
xsd: http://www.w3.org/2001/XMLSchema#

Shortcuts

When talking about the same subject

- a semicolon ";" introduces another property of the same subject
- a comma introduces another object with the same predicate and subject

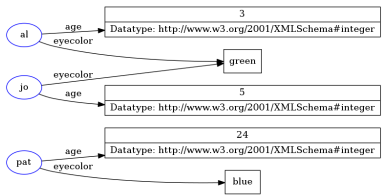
```
<#pat> <#child>  <#al>, <#chaz>, <#mo> ;
       <#age>    24 ;
       <#eyecolor> "blue" .
```



	age	eyecolor
pat	24	blue
al	3	green
jo	5	green

becomes

```
<#pat>   <#age> 24;  <#eyecolor> "blue" .
<#al>    <#age>  3;  <#eyecolor> "green" .
<#jo>    <#age>  5;  <#eyecolor> "green" .
```



Anonymous individuals

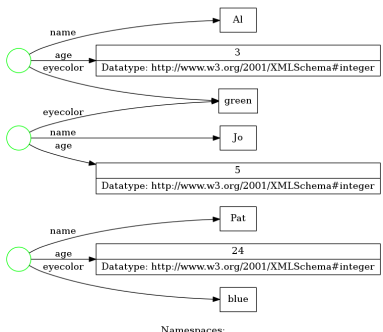
("Named") individual `#pat` has a child aged 4 and a child aged 3.

```
<#pat> <#child> [ <#age> 4 ] , [ <#age> 3 ].
```

name	age	eyecolor
Pat	24	blue
Al	3	green
Jo	5	green

Anonymous individuals with a `#name` attribute:

```
[ <#name> "Pat"; <#age> 24;
                 <#eyecolor> "blue"  ].
[ <#name> "Al"; <#age>  3;
                <#eyecolor> "green" ].
[ <#name> "Jo"; <#age>  5;
                <#eyecolor> "green" ].
```



Visualize with `https://www.ldf.fi/service/rdf-grapher`

Outline

General strategy to write a formal ontology

- Identify the primitive classes and relations of the domain.
- Establish the taxonomy of the classes and of the relations.
- Perform an ontological analysis.
  - Provide definitions, and identify precise relations between classes.
  - Leverage upon the theories of essence, identity, space, time, constitution, mereology, etc.
- Write formal axioms.

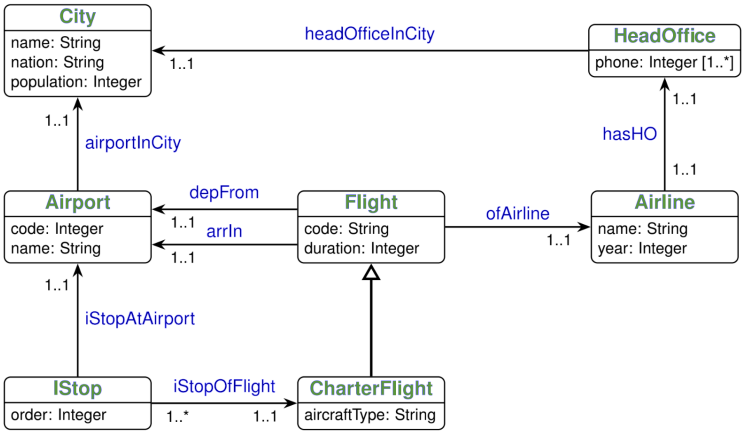
See https://www.w3.org/2000/10/swap/Primer (Making vocabularies)

Example: Flights domain

- The information about each flight includes: code, duration in minutes, airline, airport of departure, and airport of arrival.
- The information about each airport includes: code, name, city (with name and population), and nation.
- The information about each airline includes: name, year of establishment, and city of the head office with its phone number. It is assumed that every office has at least one phone number.
- Charter flights are a special kind of flights that foresee intermediate stops at airports. The order of the stops is of interest The type of the aircraft is also of interest.

Chosen classes, relations, taxonomy:

- Classes: Flight, Airline, HeadOffice, Airport, City, IStop, CharterFlight
- Class taxonomy: CharterFlight is a Flight; the rest of the taxonomy is flat.
- Relations: depFrom, arrIn, airportInCity, ofAirline, iStopOfFlight, iStopAirport, hasHO, headOfficeInCity
- Relation taxonomy: flat.


City
name: String
nation: String
population: Integer
headOfficeInCity
1..1
HeadOffice
phone: Integer [1..*]
1..1
1..1
airportInCity
hasHO
1..1
Airport
code: Integer
name: String
depFrom
1..1
arrIn
1..1
Flight
code: String
duration: Integer
ofAirline
1..1
Airline
name: String
year: Integer
1..1
iStopAtAirport
IStop
order: Integer
iStopOfFlight
1..*
1..1
CharterFlight
aircraftType: String

Prefixes, class declarations and taxonomy

We use the Terse RDF Triple Language (Turtle). https://www.w3.org/TR/turtle/

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix flights: <http://www.example.org/flights#> .

<http://www.example.org/flights> rdf:type owl:Ontology .

flights:City rdf:type owl:Class .
flights:HeadOffice rdf:type owl:Class .
flights:Airport rdf:type owl:Class .
flights:Flight rdf:type owl:Class.
flights:Airline rdf:type owl:Class .
flights:CharterFlight rdf:type owl:Class ;
  rdfs:subClassOf flights:Flight .
flights:IStop rdf:type owl:Class .
```

Properties declarations

```
flights:headOfficeInCity rdf:type owl:ObjectProperty .
flights:airportInCity rdf:type owl:ObjectProperty .
flights:hasHO rdf:type owl:ObjectProperty .
flights:depFrom rdf:type owl:ObjectProperty .
flights:arrIn rdf:type owl:ObjectProperty .
flights:ofAirline rdf:type owl:ObjectProperty .
flights:iStopAtAirport rdf:type owl:ObjectProperty .
flights:iStopOfFlight rdf:type owl:ObjectProperty .
```

Domains and ranges

```
flights:headOfficeInCity rdfs:domain flights:HeadOffice ;
  rdfs:range flights:City .
flights:airportInCity rdfs:domain flights:Airport ;
  rdfs:range flights:City .
flights:hasHO rdfs:domain flights:Airline ;
  rdfs:range flights:HeadOffice .
flights:depFrom rdfs:domain flights:Flight ;
  rdfs:range flights:Airport .
flights:arrIn rdfs:domain flights:Flight ;
  rdfs:range flights:Airport .
flights:ofAirline rdfs:domain flights:Flight ;
  rdfs:range flights:Airline .
flights:iStopAtAirport rdfs:domain flights:IStop ;
  rdfs:range flights:Airport .
flights:iStopOfFlight rdfs:domain flights:IStop ;
  rdfs:range flights:CharterFlight .
```

Date type properties

```
flights:name rdf:type owl:DatatypeProperty ; rdfs:range xsd:string .
flights:nation rdf:type owl:DatatypeProperty ; rdfs:range xsd:string .
flights:population rdf:type owl:DatatypeProperty ; rdfs:range xsd:integer .
flights:phone rdf:type owl:DatatypeProperty ; rdfs:range xsd:integer .
flights:year rdf:type owl:DatatypeProperty ; rdfs:range xsd:integer .
flights:duration rdf:type owl:DatatypeProperty ; rdfs:range xsd:integer .
flights:order rdf:type owl:DatatypeProperty ; rdfs:range xsd:integer .
flights:aircraftType rdf:type owl:DatatypeProperty ; rdfs:range xsd:string .

flights:code rdf:type owl:DatatypeProperty .
flights:codeAirport rdf:type owl:DatatypeProperty ;
  rdfs:subPropertyOf flights:code ;
  rdfs:domain flights:Airport ;
  rdfs:range xsd:integer .
flights:codeFlight rdf:type owl:DatatypeProperty ;
  rdfs:subPropertyOf flights:code ;
  rdfs:domain flights:Flight ;
  rdfs:range xsd:string .
```

Mandatory participation (objects)

```
flights:HeadOffice rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:headOfficeInCity ;
  owl:someValuesFrom flights:City
] .
flights:Airport rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:airportInCity ;
  owl:someValuesFrom flights:City
] .
flights:Airline rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:hasHO ;
  owl:someValuesFrom flights:HeadOffice
] .
```

```
flights:Flight rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:depFrom ;
  owl:someValuesFrom flights:Airport
] , [
  rdf:type owl:Restriction ;
  owl:onProperty flights:arrIn ;
  owl:someValuesFrom flights:Airport
] .
flights:Flight rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:ofAirline ;
  owl:someValuesFrom flights:Airline
] .
flights:IStop rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:iStopAtAirport ;
  owl:someValuesFrom flights:Airport
] , [
  rdf:type owl:Restriction ;
  owl:onProperty flights:iStopOfFlight ;
  owl:someValuesFrom flights:Flight
] .
```

Mandatory participation (data)

```
flights:City rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:name ;
  owl:someValuesFrom xsd:string
] , [
  rdf:type owl:Restriction ;
  owl:onProperty flights:nation ;
  owl:someValuesFrom xsd:string
] , [
  rdf:type owl:Restriction ;
  owl:onProperty flights:population ;
  owl:someValuesFrom xsd:integer
] .
flights:HeadOffice rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:phone ;
  owl:someValuesFrom xsd:integer
] .
```

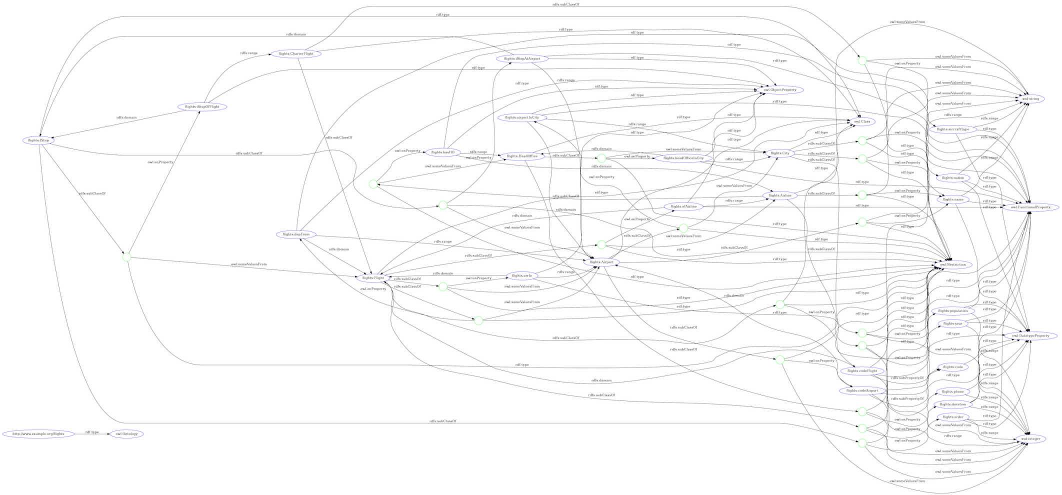
```
flights:Airport rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:codeAirport ;
  owl:someValuesFrom xsd:integer
] , [
  rdf:type owl:Restriction ;
  owl:onProperty flights:name ;
  owl:someValuesFrom xsd:string
] .
flights:Flight rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty flights:codeFlight ;
  owl:someValuesFrom xsd:string
] , [
  rdf:type owl:Restriction ;
  owl:onProperty flights:duration ;
  owl:someValuesFrom xsd:integer
] .

...
.
```

Max arity 1

```
flights:headOfficeInCity rdf:type owl:FunctionalProperty .
flights:airportInCity rdf:type owl:FunctionalProperty .
flights:hasHO rdf:type owl:FunctionalProperty .
flights:depFrom rdf:type owl:FunctionalProperty .
flights:arrIn rdf:type owl:FunctionalProperty .
flights:ofAirline rdf:type owl:FunctionalProperty .
flights:iStopAtAirport rdf:type owl:FunctionalProperty .

flights:name rdf:type owl:FunctionalProperty .
flights:nation rdf:type owl:FunctionalProperty .
flights:population rdf:type owl:FunctionalProperty .
flights:year rdf:type owl:FunctionalProperty .
flights:duration rdf:type owl:FunctionalProperty .
flights:order rdf:type owl:FunctionalProperty .
flights:aircraftType rdf:type owl:FunctionalProperty .
flights:codeAirport rdf:type owl:FunctionalProperty .
flights:codeFlight rdf:type owl:FunctionalProperty .
```



Namespaces:
xsd: http://www.w3.org/2001/XMLSchema#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
owl: http://www.w3.org/2002/07/owl#
flights: http://www.example.org/flights#

Outline

`https://protege.stanford.edu/`

On an example...

We create a small ontology of pizzas, where:

- types of pizza (margherita, diavola, 4 stagioni, capricciosa, ...) are individuals of the ontology.
- ingredients (mozzarella, anchovies, ...) are individuals, too.

(We could have chosen them to be classes.)

Under tabs ‘Entities’/‘Classes’, below owl:Thing, create two classes:

- :Pizza
- :Topping

Roles / Object Properties

In tab ‘Entities’/‘Object properties’, below owl:topObjectProperty, create the role:

- :hasTopping

Individuals

In tab ‘Entities’/‘Individuals’, create:

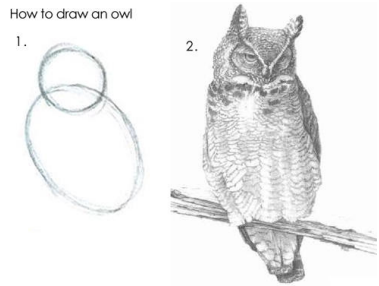
- :margherita, and under ‘Types’/‘Class hierarchy’, place it in the class :Pizza
- :tomato, and place it in the class :Topping

Create the rest of the OWL ontology

Create taxonomy:

- owl:Thing
  - :Pizza
    - :VegetarianPizza
  - :Topping
    - :FishBased
    - :MeatBased
    - :OtherTopping
    - :Vegetarian
      - :Cheese
      - :Vegan
        - :Vegetable

Add more stuff: next slides.



Add more individuals

- :diavola in :Pizza
- (:margherita in Pizza) done
- :mozzarella in Cheese
- :salami in MeatBased
- (:tomato in Vegetable) half done

Disjoint classes

$$\text{Vegetarian} \sqsubseteq \neg\text{MeatBased}$$

When vizuzaling the class :Vegetarian, under 'Disjoint With', select :MeatBased

Equivalent classes

$$\text{VegetarianPizza} \equiv \text{Pizza} \sqcap \forall \text{hasTopping}.\text{Vegetarian}$$

Under ‘Equivalent to’, add: :hasTopping only :Vegetarian

Classes equivalent to disjoint union of classes

- Topping $\equiv$ OtherTopping $\sqcap$ FishBased $\sqcap$ MeatBased $\sqcap$ Vegetarian
- OtherTopping $\sqsubseteq \neg$Vegetarian
- OtherTopping $\sqsubseteq \neg$FishBased
- OtherTopping $\sqsubseteq \neg$MeatBased
- FishBased $\sqsubseteq \neg$MeatBased
- FishBased $\sqsubseteq \neg$Vegetarian
- Vegetarian $\sqsubseteq \neg$MeatBased

Under tab 'Entities'/'Classes', while vizualizing :Topping, under 'Disjoint Union Of' select:

- :OtherTopping
- :FishBased
- :MeatBased
- :Vegetarian

Range and domain restrictions

$$\top \sqsubseteq \forall \text{hasTopping}.\text{Topping}$$

Under tabs ‘Entities’/‘Object properties’, while vizualizing :hasTopping, under ‘Ranges’/‘Class hierarchy’ select :Topping.

Outline

Conjunctive queries

Queries with k free variables:

$$q(x_1, \ldots, x_k) \leftarrow \exists x_{k+1} \ldots x_m . Pred_1(\overline{x}) \wedge \ldots \wedge Pred_n(\overline{x})$$

Special case, concept queries:

$$q(x_1) \leftarrow C(x_1)$$

Simple concept-based queries with DL in Protégé

Reasoner > Start Reasoner.

Then, in 'DL Query' tab:

- :Pizza
- :Pizza and :hasTopping some :MeatBased
- not :VegetarianPizza
- :Pizza and not :VegetarianPizza
- ...

Surprises with the Open-World Assumption

The query :VegetarianPizza does not return margherita.

Open world assumption:

- hasTopping(margherita, mozzarella)
- hasTopping(margherita, tomato)
- both Vegetarian(mozzarella) and Vegetarian(tomato)
- but in some models, margherita has some additional toppping that aren't vegetarian.

Solutions:

- add
  - that margherita has exactly two toppings, i.e., ABox axiom $((= 2\, \text{hasTopping}.\top))(\text{margherita})$:
    - under tabs 'Entities'/'Individuals' while vizualizing margherita...
    - under 'Types' select 'Object restriction creator', ...
    - restricted property: hasTopping
    - restriction filler: owl:Thing
    - restriction type: exactly 2
  - that tomato is different from mozzarella, i.e., ABox axiom tomato $\neq$ mozzarella:
    - under tabs 'Entities'/'Individuals' while vizualizing tomato...
    - under 'Different Individuals' select mozzarella
- or consider pizzas and toppings as classes and define: Margherita $\equiv$
  - $\exists \text{hasTopping}.\text{Mozzarella} \sqcap \exists \text{hasTopping}.\text{Tomato} \sqcap \forall \text{hasTopping}.(\text{Mozzarella} \sqcup \text{Tomato})$

Complex queries in SPARQL

Some queries cannot be expressed using a concept.

E.g., the toppings that are on a pizza alongside a topping that does not go well together.

$$q(x) \leftarrow \text{hasTopping}(y, x) \wedge \text{hasTopping}(y, z) \wedge \text{doesNotGoWellTogether}(x, z)$$

We can use SPARQL to flexibly query the **ABox**. (No TBox reasoning if using a plugin with simple *entailment regime*! No handling of complex queries if using the SNAP plugin...)

```
PREFIX pizza: <http://www.example.org/lil-pizza#>
PREFIX new: <http://www.example.org/extra-pizza#>

SELECT DISTINCT ?topping WHERE {
  {
    SELECT DISTINCT ?topping WHERE {
      ?pizza pizza:hasTopping ?topping .
      ?pizza pizza:hasTopping ?othertopping .
      ?topping new:doesNotGoWellTogether ?othertopping
    }
  }
}
```

Complex queries in SPARQL

E.g., the list of toppings alongside the pizza they go on.

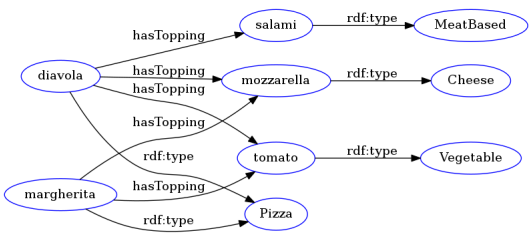
$$q(x, y) \leftarrow \mathsf{hasTopping}(y, x)$$

```
PREFIX pizza: <http://www.example.org/lil-pizza#>

SELECT ?topping ?pizza WHERE {
  ?pizza pizza:hasTopping ?topping .
}
```

Result:

?topping	?pizza
salami	diavola
tomato	margherita
mozzarella	margherita
tomato	diavola
mozzarella	diavola



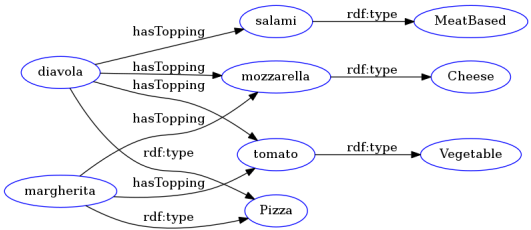
Complex queries in SPARQL

Some queries can benefit from SQL-like querying.

E.g., top two toppings in meat-based pizzas, ordered by name:

```
PREFIX pizza: <http://www.example.org/lil-pizza#>

SELECT DISTINCT ?topping WHERE {
  {
    SELECT DISTINCT ?pizza WHERE {
      ?pizza a pizza:Pizza .
      ?pizza pizza:hasTopping ?topping .
      ?topping a pizza:MeatBased
    }
  }
  ?pizza pizza:hasTopping ?topping
} ORDER BY ?topping LIMIT 2
```



Result:

?topping
mozzarella
salami

Exercise

- *Add a role :doesNotGoWellTogether; make it symmetric.*
- *Add an individual pineapple*
- *Add the ABox axiom Vegetarian(pineapple)*
- *Add the ABox axiom doesNotGoWellTogether(pineapple, mozzarella)*
- *Specify :Topping as the domain and range of doesNotGoWellTogether(pineapple, mozzarella)*
- *Create a Hawaiian pizza with tomato, mozzarella, salami, and pineapple toppings.*
- *Find the toppings that are on a pizza alongside a topping that does not go well together*
- *See that making the role :doesNotGoWellTogether symmetric has no effect (if using a plugin with simple entailment regime). You can add doesNotGoWellTogether(mozzarella, pineapple) explicitly.*
- *Find top two ingredients in meat-based pizzas, ordered by name*

SPARQL keywords

- SELECT, CONSTRUCT, ASK, DESCRIBE
- UNION, MINUS
- FILTER, VALUES
- OPTIONAL
- ORDER BY, GROUP BY
- LIMIT

Down the DBpedia rabbit hole: Zooming in on Ibrahim M. and "young interesting trumpeters"

DBPedia ontology: `https://dbpedia.org/ontology/`

- `http://dief.tools.dbpedia.org/server/ontology/classes/`
- `http://dief.tools.dbpedia.org/server/ontology/dbpedia.owl`

SPARQL endpoint: `https://dbpedia.org/sparql`

Simple queries:

```
SELECT * where {
  ?mf a dbo:MusicFestival .
}

DESCRIBE <http://dbpedia.org/resource/Atlanta_Trumpet_Festival>

SELECT * WHERE {
  ?x dbo:wikiPageWikiLink dbr:Trumpet .
}
```

Down the DBpedia rabbit hole: Zooming in on Ibrahim M. and “young interesting trumpeters”

```
SELECT * WHERE {
  ?person a dbo:MusicalArtist .
  ?person ?relation dbr:Trumpet .
}

SELECT * WHERE {
  ?person dbo:instrument dbr:Trumpet .
}

SELECT * WHERE {
  ?person dbo:instrument dbr:Trumpet; dbo:birthPlace ?birthPlace
}

SELECT * WHERE {
  ?person dbo:instrument dbr:Trumpet; dbo:birthPlace ?birthPlace .
  ?birthPlace rdf:type dbo:Country .
}

SELECT * WHERE {
  ?person dbo:instrument dbr:Trumpet; dbo:birthPlace ?birthPlace .
  ?birthPlace rdf:type dbo:Country .
  ?birthPlace dbp:areaKm ?akm .
  FILTER (?akm <= '15000'^^xsd:integer)
}

SELECT * WHERE {
  ?person dbo:instrument dbr:Trumpet; dbo:birthPlace ?birthPlace .
  ?birthPlace rdf:type dbo:Country .
  ?birthPlace dbp:areaKm ?akm .
  FILTER (?akm <= '15000'^^xsd:integer) .
  ?birthPlace rdf:type yago:WikicatMiddleEasternCountries .
}

DESCRIBE <http://dbpedia.org/resource/Ibrahim_Maalouf>
```

Down the DBpedia rabbit hole: Zooming in on Ibrahim M. and “young interesting trumpeters”

```
SELECT * WHERE {
  ?person dbo:instrument dbr:Trumpet; dbo:birthDate ?date .
  FILTER (?date >= '1980-01-01'^^xsd:date) .
}

SELECT * WHERE {
  ?person dbo:instrument dbr:Trumpet; dbo:birthDate ?date .
  FILTER (?date >= '1980-01-01'^^xsd:date) .
  ?person dbp:occupation ?occupation .
}

SELECT * WHERE {
{
  SELECT ?person (COUNT(?occupation) AS ?numOccupations) WHERE {
    ?person dbo:instrument dbr:Trumpet; dbo:birthDate ?date .
    FILTER (?date >= '1980-01-01'^^xsd:date) .
    ?person dbp:occupation ?occupation .
  } GROUP BY ?person
}
  FILTER (?numOccupations > 2) .
}

DESCRIBE <http://dbpedia.org/resource/Mika_Horiuchi>
```

Outline

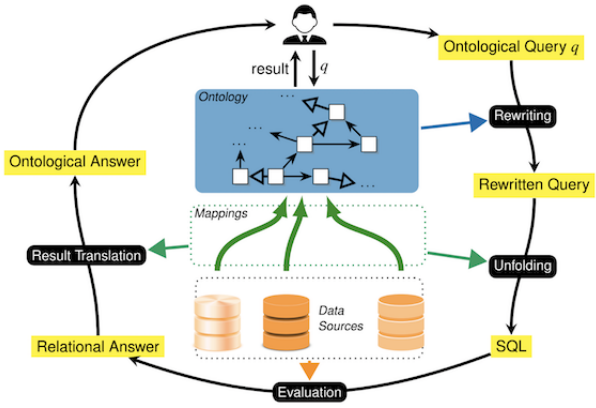
"We advocate that for OBDA, i.e., all for those contexts where ontologies are used to access large amounts of data, a suitable DL should be used, specifically tailored to capture all those constructs that are used typically in conceptual modeling, while keeping query answering efficient. Specifically, efficiency should be achieved by delegating data storage and query answering to a relational data management systems (RDBMS), which is the only technology that is currently available to deal with complex queries over large amounts of data. The chosen DL should include the main modeling features of conceptual models, which are also at the basis of most ontology languages. These features include cyclic assertions, ISA and disjointness of concepts and roles, inverses on roles, role typing, mandatory participation to roles, functional restrictions of roles, and a mechanisms for identifying instances of concepts." [Calvanese et al. 2009][1]

[1] Diego Calvanese et al. "Ontologies and Databases: The DL-Lite Approach". In: *Reasoning Web, Tutorial Lectures*. 2009, pp. 255–356.

Languages

The choice of the right languages needs to take into account the tradeoff between expressive power and efficiency of query answering.
Data plays a prominent role: efficiency with respect to the data is the key factor.
The W3C has standardized languages that are suitable for VKGs:

- Knowledge graph: expressed in RDF
- Ontology: expressed in OWL 2 QL
- Mapping: expressed in Ontop (mapping) (fully interoperable with R2RML)
- Query: expressed in SPARQL

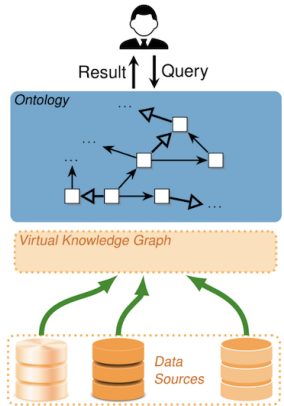

result
q
Ontological Query q
Ontology
Rewriting
Ontological Answer
Rewritten Query
Mappings
Result Translation
Unfolding
Data
Sources
Relational Answer
SQL
Evaluation

Virtual knowledge graph (VKG)

In VKGs, the mapping encodes how the data in the sources should be used to populate the knowledge graph.

Virtual knowledge graph $\mathcal{V} = \mathcal{M}(\mathcal{D})$:

- $\mathcal{D}$: data.
- $\mathcal{M}$: mapping.
- Queries are answered wrt. $\mathcal{O}$ and $\mathcal{V}$.
- $\mathcal{V}$ is not materialized.
- Instead, the information in $\mathcal{O}$ is used to translate queries over $\mathcal{O}$ into queries formulated over the data sources.
- Advantage of non-materialization: always up-to-date wrt. the data sources.



Ontop mapping: RDB to RDF mapping language

A mapping is a set of assertions:

$$\Phi(\vec{x}) \mapsto \Psi(\vec{x}, \vec{t})$$

where

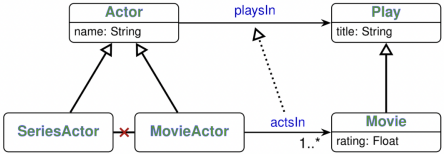
- $\Phi(\vec{x})$ is the source query in SQL over a database
- $\Psi(\vec{x}, \vec{t})$ is the target query, consisting of ripple patters over the classes and properties of the ontology.

In the Ontop mapping language, each assertion is made of:

- an identifier
- a source; regular SQL query
- a target: a set of triples making use of IRI-templates. The answer variables are enclosed in $\{\ldots\}$.

Example

Ontology:



Mapping:

```
mappingID   class-Movie
target      :m/{mcode} rdf:type :Movie ; :title {mtitle} .
source      SELECT mcode, mtitle FROM Movie
            WHERE type = "m"

mappingID   property-actsIn
target      :a/{acode} :actsIn :m/{mcode} .
source      SELECT M.mcode, A.acode FROM Movie M, Actor A
            WHERE M.mcode = A.pcode AND M.type = "m"
```

Database:

Movie

mcode	mtitle	myear	mtype	...
5118	The Matrix	1999	m	...
8234	Altered Carbon	2018	s	...
2281	Blade Runner	1982	m	...

Actor

pcode	acode	aname	...
5118	438	K. Reeves	...
5118	572	C.A. Moss	...
2281	271	H. Ford	...

Virtual Knowledge Graph: the mapping applied to the database generates the VKG:

```
:m/5118 rdf:type :Movie .  :m/5118 :title "The Matrix" .
:m/2281 rdf:type :Movie .  :m/2281 :title "Blade Runner" .
:a/438 :actsIn :m/5118 .   :a/572 :actsIn :m/5118 .        :a/271 :actsIn :m/2281 .
```

Ontop

`https://ontop-vkg.org/`

L'Aquila – Virtual Knowledge Graph

See `https://bitbucket.org/troquard/ontop-aquila` for a docker instance of Ontop, data on the reconstruction of L'Aquila, an ontology of the reconstruction domain, mappings, and queries.

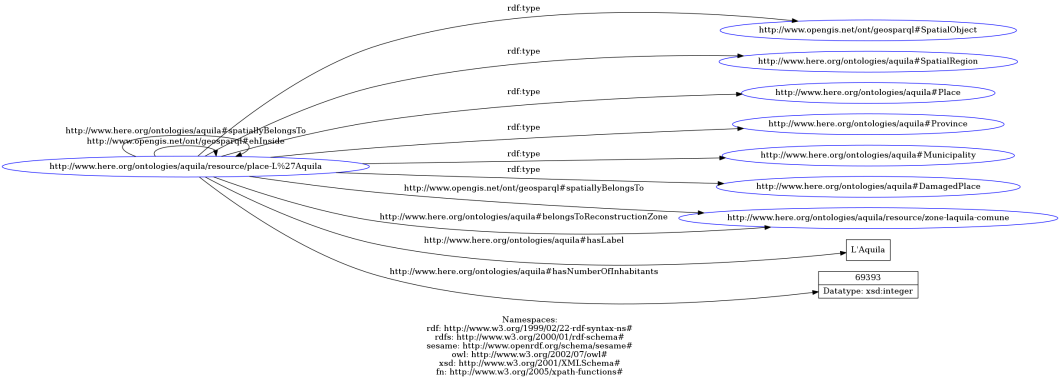
- Data sources, and DB scripts:
  `https://bitbucket.org/troquard/ontop-aquila/src/master/db/`
- Ontology: `https://bitbucket.org/troquard/ontop-aquila/src/master/vkg/vkg.ttl`
- Mappings: `https://bitbucket.org/troquard/ontop-aquila/src/master/vkg/vkg.obda`
- Some queries: `https://bitbucket.org/troquard/ontop-aquila/src/master/vkg/vkg.toml`

DESCRIBE place-L%27Aquila

```
DESCRIBE <http://www.here.org/ontologies/aquila/resource/place-L%27Aquila>
```

```
<http://www.here.org/ontologies/aquila/resource/place-L%27Aquila> a <http://www.opengis.net/ont/geosparql#SpatialObject>,
    <http://www.here.org/ontologies/aquila#SpatialRegion>, <http://www.here.org/ontologies/aquila#Place>,
    <http://www.here.org/ontologies/aquila#Province>, <http://www.here.org/ontologies/aquila#Municipality>,
    <http://www.here.org/ontologies/aquila#DamagedPlace>;
  <http://www.opengis.net/ont/geosparql#ehInside> <http://www.here.org/ontologies/aquila/resource/place-L%27Aquila>;
  <http://www.here.org/ontologies/aquila#spatiallyBelongsTo> <http://www.here.org/ontologies/aquila/resource/place-L%27Aquila>;
  <http://www.opengis.net/ont/geosparql#spatiallyBelongsTo> <http://www.here.org/ontologies/aquila/resource/zone-laquila-comune>;
  <http://www.here.org/ontologies/aquila#belongsToReconstructionZone> <http://www.here.org/ontologies/aquila/resource/zone-laquila-com
  <http://www.here.org/ontologies/aquila#hasLabel> "L'Aquila";
  <http://www.here.org/ontologies/aquila#hasNumberOfInhabitants> 69393 .
```

VKG: place-L%27Aquila



Credits

Many slides and examples based on Diego Calvanese's tutorials `http://www.inf.unibz.it/~calvanese/`.