Master in Artificial Intelligence 2022-23

Fundamentals of Artificial Intelligence

Practice 2: Ontologies.

Group 2
Guijas Bravo, Pedro
Miguélez Millos, Ángel
Vila de la Cruz, Daniel

January 17, 2023

Contents

1 Editing an Ontology 1

1.1 Importing the ontology 1

1.2 Adding allergens to the ontology 1

1.3 Defining that some health outcomes are caused by inflammation 2

1.4 Creating an instance of a sandwich 5

2 Querying in SPARQL 6

2.1 Cats and monkey 6

2.2 Cristiano Ronaldo 6

2.3 Cristian Munteanu 7

2.4 Height of the people 7

1 Editing an Ontology

The following subsections will report on the modifications made to the **ontology of fast food facts (OFFF)** [1], using the WebProtégé tool [2]. Note that these modifications are requested in the *P2-AIF-Tasks.pdf* file provided, and will be made with the main objective of learning about ontologies.

1.1 Importing the ontology

First of all, the ontology of fast food facts must be imported. For doing this, the *.owl* file [1] that contains the information about this ontology must be added when creating a project in WebProtégé as shown in the Figure 1.

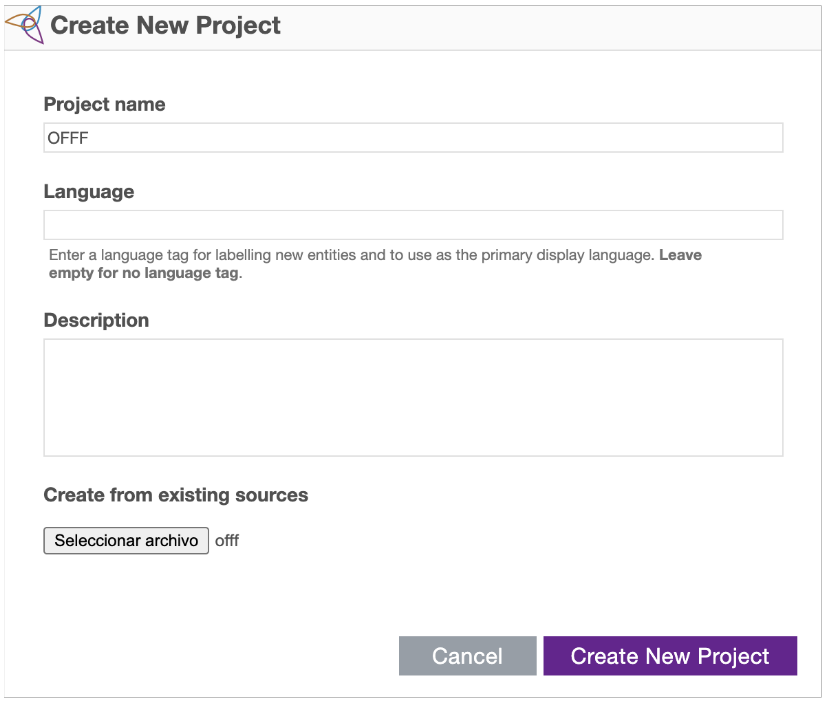
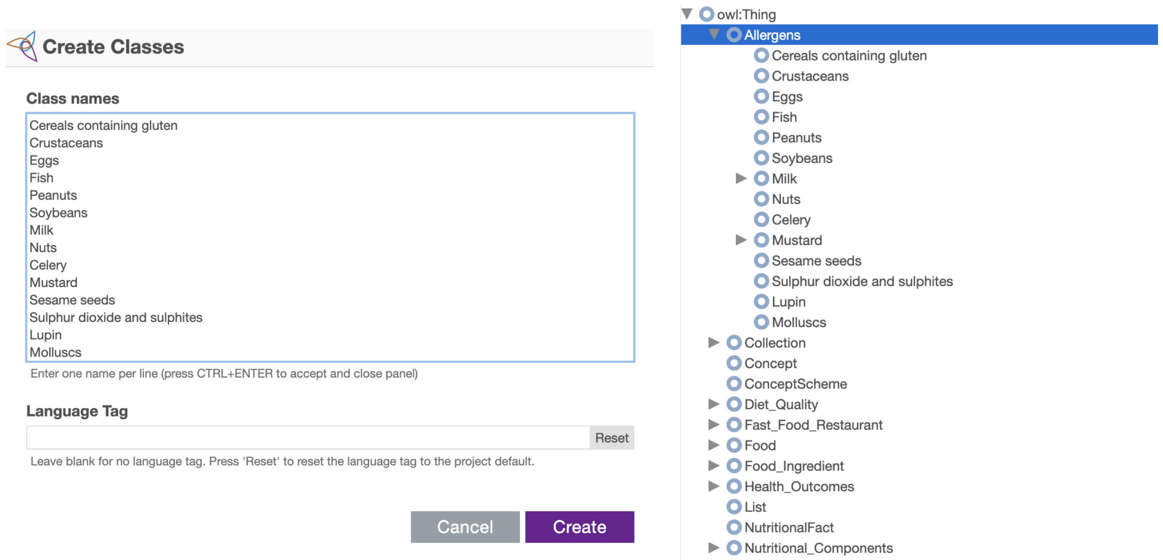

Create New Project
Project name
OFFF
Language
Enter a language tag for labelling new entities and to use as the primary display language. Leave empty for no language tag.
Description
Create from existing sources
Seleccionar archivo
offf
Cancel
Create New Project

Figure 1: Importing OFFF at WebProtégé

1.2 Adding allergens to the ontology

Once the ontology has been established, the first step is to add the 14 main EU allergens [3] to the ontology. To do this, go to the "Classes" tab, select the *Allergens* class, click on the "create" icon, and then paste the list of allergens to be added. See graphically the process in the Figure 2.

[1] https://github.com/UTHealth-Ontology/OFFF



Cereals containing gluten
Crustaceans
Eggs
Fish
Peanuts
Soybeans
Milk
Nuts
Celery
Mustard
Sesame seeds
Sulphur dioxide and sulphites
Lupin
Molluscs

Enter one name per line (press CTRL+ENTER to accept and close panel)

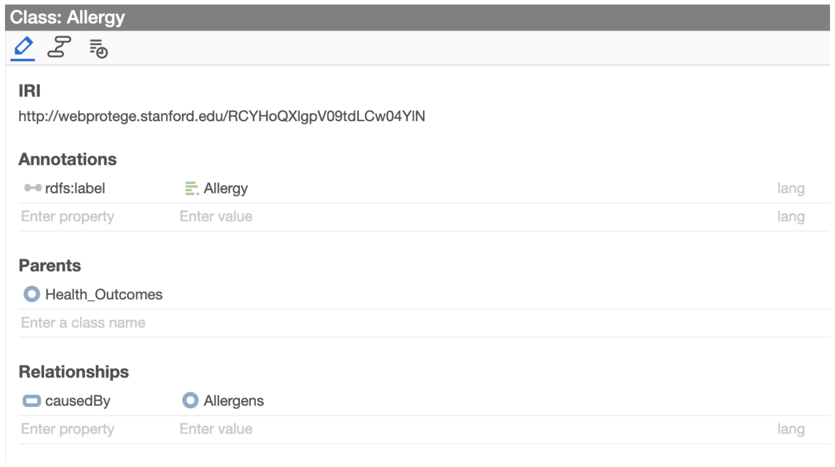
Leave blank for no language tag. Press 'Reset' to reset the language tag to the project default.

owl:Thing
- Allergens
  - Cereals containing gluten
  - Crustaceans
  - Eggs
  - Fish
  - Peanuts
  - Soybeans
  - Milk
  - Nuts
  - Celery
  - Mustard
  - Sesame seeds
  - Sulphur dioxide and sulphites
  - Lupin
  - Molluscs
- Collection
- Concept
- ConceptScheme
- Diet_Quality
- Fast_Food_Restaurant
- Food
- Food_Ingredient
- Health_Outcomes
- List
- NutritionalFact
- Nutritional_Components

(a) Creation process (b) Result

Figure 2: Creating the allergens in the ontology

In the same way, the class *Allergy* must be created to the ontology, this must be a subclass of *Health_Outcomes*. Next, to provide the ontology with the information that allergies are caused by allergens, a new relationship will be added to *Allergy* that relates it to *Allergens* by using the property *causedBy*. Again, Figure 3 graphically details the process of adding this relationship.



IRI
http://webprotege.stanford.edu/RCYHoQXlgpV09tdLCw04YIN

Annotations

rdfs:label	Allergy	lang
Enter property	Enter value	lang

Parents

Health_Outcomes
Enter a class name

Relationships

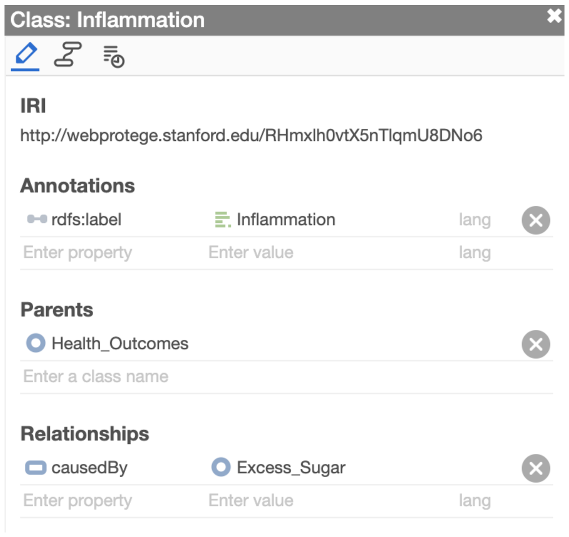
causedBy	Allergens	
Enter property	Enter value	lang

Figure 3: Relating *Allergy* with *Allergens*

1.3 Defining that some health outcomes are caused by inflammation

The third step consists to establish that heart disease, hypertension, obesity and type II diabetes are caused by inflammation. For doing so, it is necessary to create the *Inflammation* class as a new subclass of *Health_Outcomes*. After that, we have to add the relationship between the above health outcomes classes and *Inflammation*, using the *causedBy* property. Then, we also define the relationship that links *Inflammation* with *Excess_Sugar* class, using again the property *causedBy*.

Following the format, Figure 4 shows the result of adding the *Inflammation* class, while Figures 5, 6, 7 and 8 reflect the result of relating *Inflammation* as a cause of different *Health_Outcomes* by means of the *causedBy* relationship.


Class: Inflammation

IRI
http://webprotege.stanford.edu/RHmxlh0vtX5nTlqmU8DNo6

Annotations

rdfs:label	Inflammation	lang
Enter property	Enter value	lang

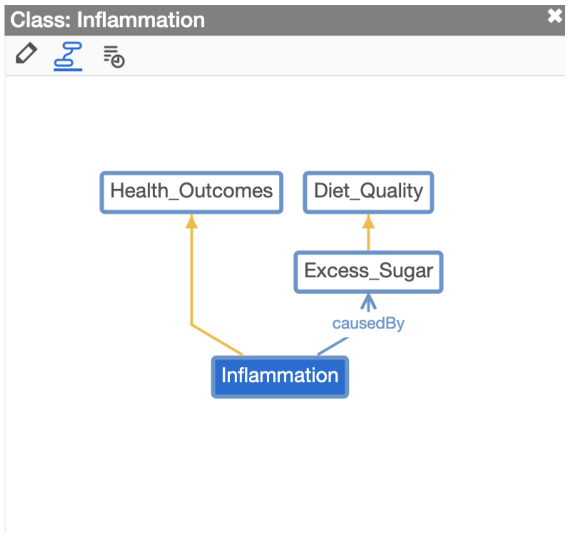
Parents

Health_Outcomes
Enter a class name

Relationships

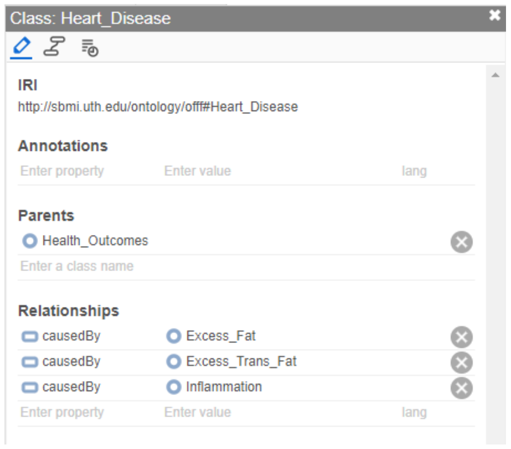
causedBy	Excess_Sugar	
Enter property	Enter value	lang

(a) Class details



(b) Entity graph

Figure 4: Excess sugar


Class: Heart_Disease

IRI
http://sbmi.uth.edu/ontology/offf#Heart_Disease

Annotations

Enter property	Enter value	lang

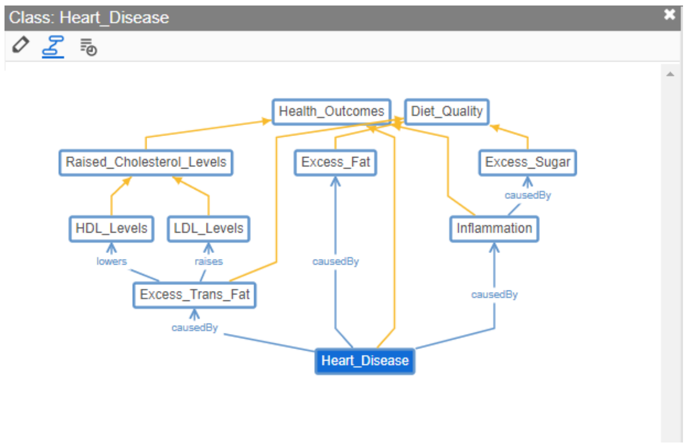
Parents

Health_Outcomes
Enter a class name

Relationships

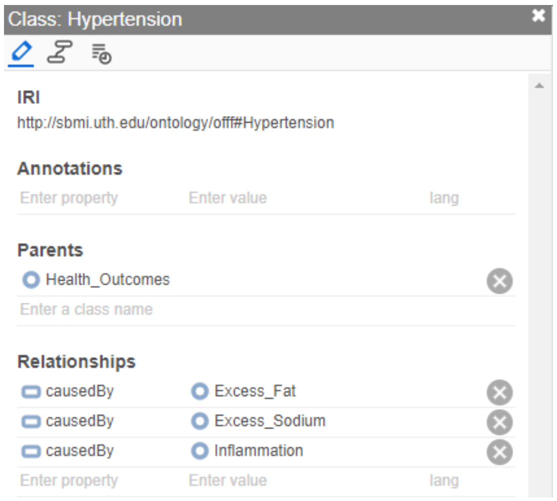
causedBy	Excess_Fat	
causedBy	Excess_Trans_Fat	
causedBy	Inflammation	
Enter property	Enter value	lang

(a) Class details



(b) Entity graph

Figure 5: Heart Disease


Class: Hypertension

IRI
http://sbmi.uth.edu/ontology/offf#Hypertension

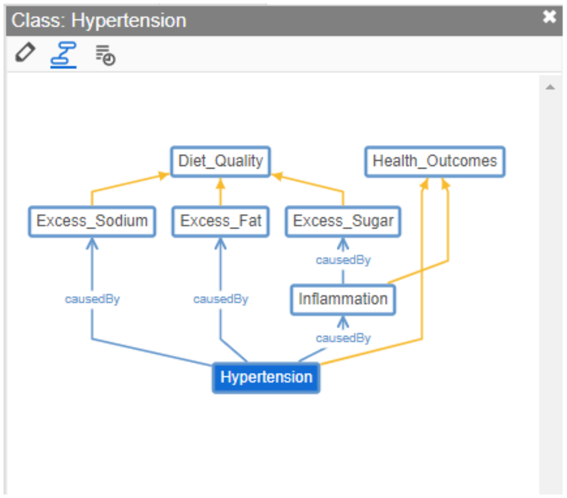
Annotations
Enter property | Enter value | lang

Parents
Health_Outcomes
Enter a class name

Relationships

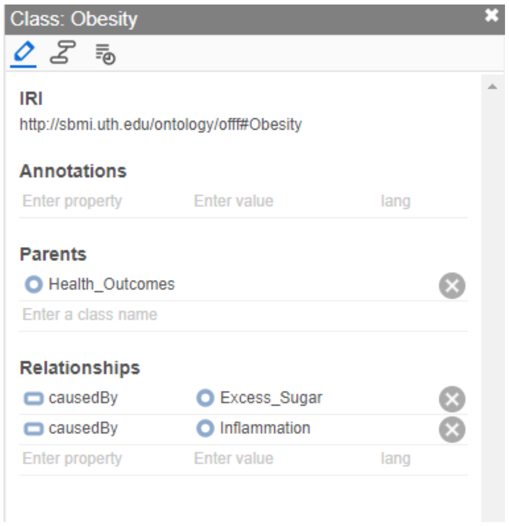
causedBy	Excess_Fat
causedBy	Excess_Sodium
causedBy	Inflammation
Enter property	Enter value

(a) Class details



(b) Entity graph

Figure 6: Hypertension


Class: Obesity

IRI
http://sbmi.uth.edu/ontology/offf#Obesity

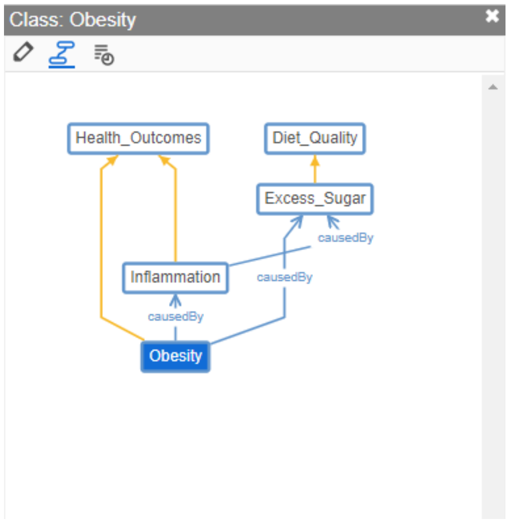
Annotations
Enter property | Enter value | lang

Parents
Health_Outcomes
Enter a class name

Relationships

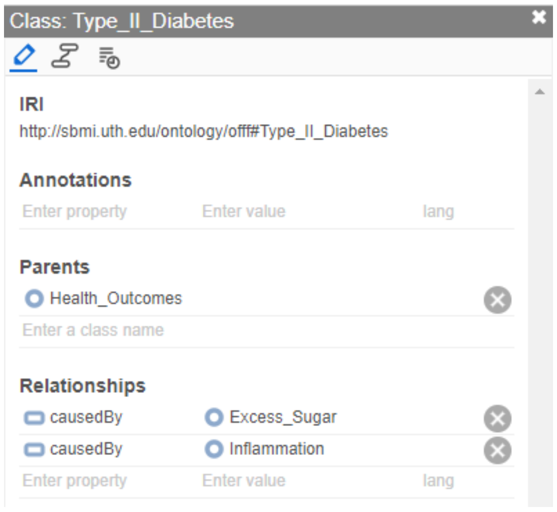
causedBy	Excess_Sugar
causedBy	Inflammation
Enter property	Enter value

(a) Class details



(b) Entity graph

Figure 7: Obesity


Class: Type_II_Diabetes

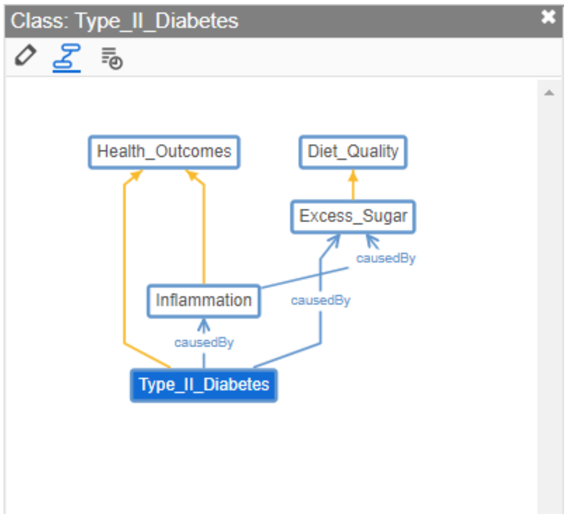
IRI
http://sbmi.uth.edu/ontology/offf#Type_II_Diabetes

Annotations
Enter property | Enter value | lang

Parents
Health_Outcomes
Enter a class name

Relationships
causedBy | Excess_Sugar
causedBy | Inflammation
Enter property | Enter value | lang

(a) Class details

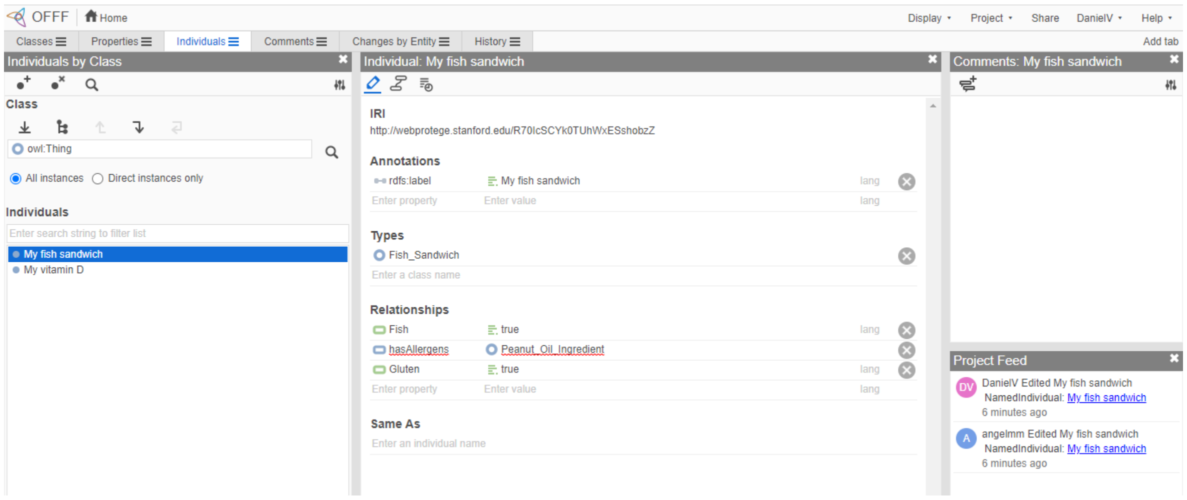


(b) Entity graph

Figure 8: Diabetes

1.4 Creating an instance of a sandwich

The last step involves the creation of an individual, particularly an instance of the class *Fish Sandwich* with 3 data properties. To do that, go to "Individuals" tab, create a new individual called "My fish sandwich", establish the type to *Fish_Sandwich* and set the relationships *"Fish"* - *"true"*, *"Gluten"* - *"true"* and *"hasAllergens"* - *"Peanut_Oil_Ingredient"*. The result is shown in the Figure 9.



Individuals by Class

Class
owl:Thing
All instances ○ Direct instances only

Individuals
Enter search string to filter list
My fish sandwich
My vitamin D

Individual: My fish sandwich

IRI
http://webprotege.stanford.edu/R70IcSCYk0TUhWxESshobzZ

Annotations
rdfs:label | My fish sandwich | lang
Enter property | Enter value | lang

Types
Fish_Sandwich
Enter a class name

Relationships
Fish | true | lang
hasAllergens | Peanut_Oil_Ingredient
Gluten | true | lang
Enter property | Enter value | lang

Same As
Enter an individual name

Comments: My fish sandwich

Project Feed
DanielV Edited My fish sandwich
NamedIndividual: My fish sandwich
6 minutes ago
angelmm Edited My fish sandwich
NamedIndividual: My fish sandwich
6 minutes ago

Figure 9: my_fish_sandwich

2 Querying in SPARQL

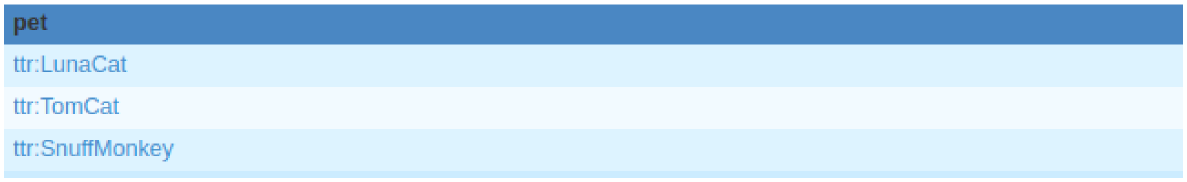
The following sections present the four queries asked for completing the part of the task corresponding to the SPARQL queries on ontology databases. The information asked, the query, its logic and the results are shown for every one of them.

2.1 Cats and monkey

This first query requires to extract all the cats and the monkeys (in fact monkey, since there is only one) using the SPARQL Playground[2]. This can be achieved with the following query:

```
SELECT * WHERE {
    {
        ?pet a tto:Cat .
    } UNION {
        ?pet a tto:Monkey .
    }
}
```

We can see that an `UNION` operator is applied to get all the subjects `?pet` that are either a `tto:Cat` or a `tto:Monkey`. The keyword `a` means, in fact, the relationship `rdf:type`. The subject `?pet` is then printed in the results using the `*` wildcard that prints all the variables in the `SELECT` clause, as shown in the Figure 10.



pet
ttr:LunaCat
ttr:TomCat
ttr:SnuffMonkey

Figure 10: Results from the cats and monkeys query

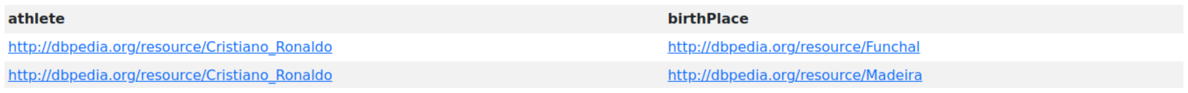
2.2 Cristiano Ronaldo

This second query asks for the birth place of an athlete called Cristiano Ronaldo in the DBpedia SPARQL ontology[3]. This is done with the following query:

```
SELECT * WHERE {
    ?athlete rdfs:label "Cristiano Ronaldo"@en ;
             dbo:birthPlace ?birthPlace .
}
```

The triple whose subject `?athlete` is labeled as ”Cristiano Ronaldo” is matched by checking that its predicate `rdfs:label` is `"Cristiano Ronaldo"@en`, where the `@en` specifies the language tag English. Then, the birth place is retrieved with the relationship `dbo:birthPlace`. It should be pointed out that this second clause lacks the subject because it is the same subject of the previous clause, and that the enumeration of the clauses are implicitly an `AND` operator. As in the previous query, all the variables `?athlete` and `birthPlace` are printed with the `*` wildcard. The results are shown in the Figure 11.

[2]https://sparql-playground.sib.swiss/
[3]https://dbpedia.org/sparql/



athlete	birthPlace
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Funchal
http://dbpedia.org/resource/Cristiano_Ronaldo	http://dbpedia.org/resource/Madeira

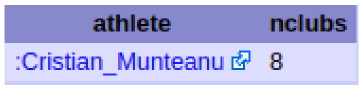
Figure 11: Results from the Cristiano Ronaldo query

2.3 Cristian Munteanu

The third query asks for the number of clubs in which an athlete called "Cristian Munteanu" has played, using the DBpedia SNORQL[4]. The query has the form of:

```
SELECT ?athlete (COUNT(?club) as ?nclubs) WHERE {
    ?athlete rdfs:label "Cristian Munteanu"@en ;
             dbp:clubs ?club .
}
```

where the first clause retrieves the athlete "Cristian Munteanu" in a similar way to the previous query, and the clubs are got by checking the `dbp:clubs` relationship. To group all the clubs into a single value that represents the amount of clubs, the aggregate `COUNT` is used in the `SELECT` clause, so every club `?club` counts as 1 and the result of the addition is passed to the `?nclubs` variable, which is printed along with the athlete name `?athlete`, as shown in the Figure 12.



athlete	nclubs
:Cristian_Munteanu	8

Figure 12: Results from the Cristian Munteanu query

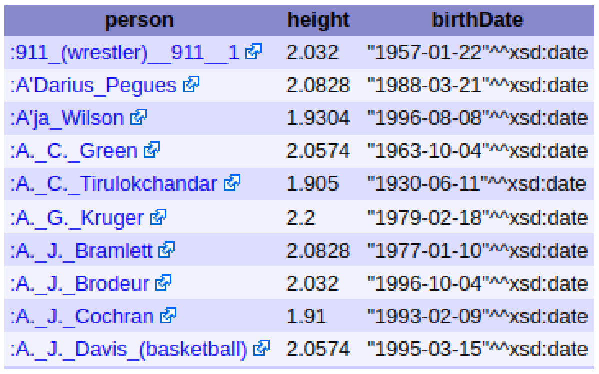
2.4 Height of the people

Finally, using the same tool as in the previous query, the first 10 people with their height greater than 1.9m that have a birth date are retrieved and ordered by name with the following query:

```
SELECT * WHERE {
    ?person a dbo:Person ;
            dbo:height ?height ;
            dbo:birthDate ?birthDate .
    FILTER (?height > "1.9"^^xsd:double)
}
ORDER BY ?person
LIMIT 10
```

First of all, all the people is retrieved by checking the subjects that are a `dbo:Person`. After that, the height (`?height`) and the birth date (`?birthDate`) relationships are consulted (people with no height or birth place would be removed). To filter by height, we have used the `FILTER` clause so it checks that the height variable `?height` is greater than 1.9. It should be pointed out that we use `"1.9"^^xsd:double` to specifically indicate that the value is a double, but if we use `1.9` instead it will work fine too. The results are also limited to a maximum of 10 rows/people with the clause `LIMIT 10` and ordered by the person name with `ORDER BY ?person`. The Figure 13 shows the results for this query.

[4]https://dbpedia.org/snorql/



person	height	birthDate
:911_(wrestler)__911__1	2.032	"1957-01-22"^^xsd:date
:A'Darius_Pegues	2.0828	"1988-03-21"^^xsd:date
:A'ja_Wilson	1.9304	"1996-08-08"^^xsd:date
:A._C._Green	2.0574	"1963-10-04"^^xsd:date
:A._C._Tirulokchandar	1.905	"1930-06-11"^^xsd:date
:A._G._Kruger	2.2	"1979-02-18"^^xsd:date
:A._J._Bramlett	2.0828	"1977-01-10"^^xsd:date
:A._J._Brodeur	2.032	"1996-10-04"^^xsd:date
:A._J._Cochran	1.91	"1993-02-09"^^xsd:date
:A._J._Davis_(basketball)	2.0574	"1995-03-15"^^xsd:date

Figure 13: Results from the people heights query

Bibliography

[1] Muhammad Amith et al. “The ontology of fast food facts: conceptualization of nutritional fast food data for consumers and semantic web applications”. In: *BMC Medical Informatics and Decision Making* 21.7 (Nov. 2021), p. 275. ISSN: 1472-6947. DOI: 10.1186/s12911-021-01636-1. URL: https://doi.org/10.1186/s12911-021-01636-1.

[2] Matthew Horridge et al. “WebProtégé: A Cloud-Based Ontology Editor”. In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW ’19. San Francisco, USA: Association for Computing Machinery, 2019, pp. 686–689. ISBN: 9781450366755. DOI: 10.1145/3308560.3317707. URL: https://doi.org/10.1145/3308560.3317707.

[3] *List of 14 allergens*. URL: https://www.fsai.ie/legislation/food_legislation/food_information/14_allergens.html.