Artificial Intelligence Fundamentals

Practice 2: Ontologies

Master in Artificial Intelligence

2025-26

Santiago Delgado Ferreiro
santiago.delgado@udc.es

David Carballo Rodríguez
david.carballo1@udc.es

October 2025

WebProtégé Username: santi_david

Contents

1 Introduction

Ontologies are formal representations of knowledge that allow structuring and organizing information in a semantic way. In the context of Artificial Intelligence, ontologies play a fundamental role by providing:

- **Knowledge representation:** Formal structuring of concepts and their relationships.
- **Interoperability:** They facilitate information exchange between heterogeneous systems.
- **Automated reasoning:** They allow inferring new knowledge from existing facts.
- **Knowledge reuse:** They promote the creation of shared knowledge bases.

1.1 Practice Objectives

The main objectives of this practice are:

1. **Understanding ontology structure** through analysis of the OFFF ontology (Ontology of Fast Food Facts).
2. **Editing ontologies** using the WebProtégé tool, adding classes, properties, and instances.
3. **Performing SPARQL queries** on knowledge bases available on the Semantic Web (DBpedia).
4. **Applying theoretical knowledge** to practical cases related to nutrition and food health.

1.2 Tools Used

- **WebProtégé:** Web-based collaborative ontology editor (https://webprotege.stanford.edu/)
- **OFFF:** Fast food ontology that integrates nutritional information
- **SPARQL:** Query language for RDF data
- **DBpedia:** Knowledge base extracted from Wikipedia

2 OFFF Ontology Editing

2.1 Context: The OFFF Ontology

The Ontology of Fast Food Facts (OFFF) is a formal representation of knowledge about fast food, developed by UTHealth. This ontology integrates:

- Nutritional information of fast food products
- Food components (carbohydrates, proteins, fats, vitamins, minerals)
- Allergens present in foods
- Specific ingredients
- Relationships with health outcomes

The hierarchical structure of OFFF allows establishing causal relationships between diet quality and various health problems, facilitating the analysis and reasoning about eating habits.

2.2 Allergen Selection

For this practice, 10 allergens have been selected from the official list of internationally recognized food allergens (https://en.wikipedia.org/wiki/List_of_allergens). The selection was based on the following criteria:

1. **Prevalence:** Most common allergens in the general population
2. **Severity:** Potential to cause serious allergic reactions
3. **Regulation:** Mandatory labeling in multiple jurisdictions
4. **Presence in fast food:** Frequency of occurrence in fast food products

2.2.1 Selected Allergens

1. **Peanut**
   - One of the most common and potentially fatal allergens
   - Present in oils, sauces, and processed products
   - Causes anaphylactic reactions in sensitive individuals
2. **Tree Nuts**
   - Includes almonds, walnuts, hazelnuts, cashews, pistachios
   - Common in salads, desserts, and baked goods
   - High risk of cross-reactions between different tree nuts
3. **Milk**
   - Most common allergen in young children
   - Present in dairy products, ice cream, cream sauces, baked goods
   - Different from lactose intolerance
4. **Egg**
   - Especially egg white (albumin)

- Used in mayonnaise, baked goods, breading
- Common in breakfast fast food products

5. **Fish**
   - Includes all finned fish
   - Present in fish sandwiches, salads, sauces
   - Reactions usually persist throughout life
6. **Shellfish**
   - Includes crustaceans (shrimp, crab, lobster) and mollusks
   - Common in salads, wraps, "premium" options
   - One of the most common allergies in adults
7. **Soy**
   - Present in vegetable proteins, oils, lecithin
   - Extensively used in processed products
   - Common component in vegetarian burgers
8. **Wheat**
   - Contains gluten (allergy-causing protein)
   - Present in bread, dough, breading, thickeners
   - Different from celiac disease
9. **Sesame**
   - Increasingly recognized as an important allergen
   - Present in breads (buns), sauces (tahini), oils
   - Recently added to the list of major allergens in the USA
10. **Celery**
    - Common allergen in Europe
    - Used in broths, soups, salads, condiments
    - Can cause serious reactions even in small amounts

2.3 Creating Allergen Subclasses

2.3.1 Modification Process

The process of adding the 10 allergens as subclasses of the `Allergens` class in WebProtégé consisted of:

1. **Navigation to parent class:** Locate the `Allergens` class in the OFFF class hierarchy

2. **Subclass creation:** Use the "Create subclass" function (+) for each allergen
3. **Nomenclature:** Use English names following OFFF convention (CamelCase or snake_case as appropriate)
4. **Verification:** Check that subclasses appear correctly in the hierarchy

2.3.2 Resulting Hierarchical Structure

The created hierarchy is as follows:

```
Allergens
|- Peanut
|- TreeNuts
|- Milk
|- Egg
|- Fish
|- Shellfish
|- Soy
|- Wheat
|- Sesame
‘- Celery
```

2.3.3 Graphical Evidence

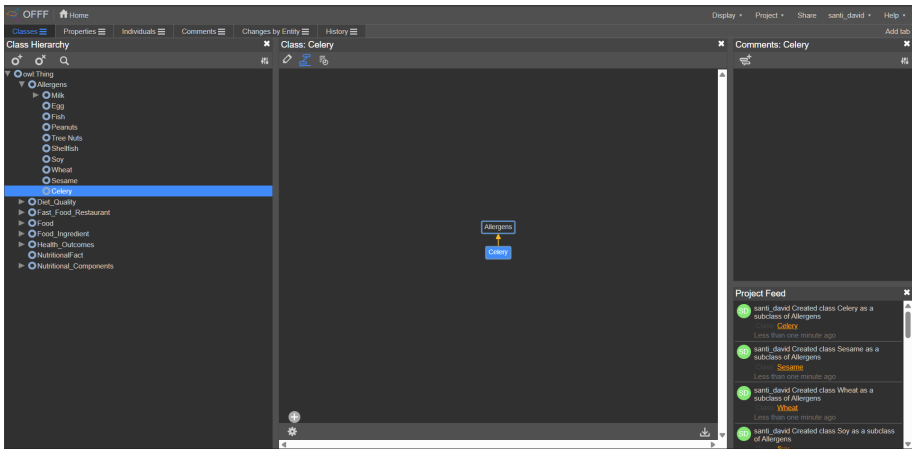


Figure 1: Class hierarchy showing the 10 new Allergens subclasses. Username `santi_david` is visible in the WebProtégé interface.

2.3.4 Technical Explanation

- **Taxonomic relationship:** A `SubClassOf` relationship is established between each allergen and the `Allergens` class
- **Inheritance:** Each subclass inherits the properties and restrictions of the parent class
- **Reasoning:** A reasoner can infer that any instance of `Peanut` is also an instance of `Allergens`

- **Extensibility:** The structure allows adding new allergens in the future without modifying the base ontology

2.4 Allergy Class in Health_Outcomes

2.4.1 Objective

Create a new `Allergy` class as a subclass of `Health_Outcomes` and establish causal relationships between allergens and this new health condition.

2.4.2 Creation Process

1. **Locate parent class:** Navigate to the `Health_Outcomes` class
2. **Create subclass:** Use "Create subclass" with the name `Allergy`
3. **Define causal relationships:** Establish that `Allergy` is caused by each of the 10 allergens

2.4.3 `causedBy` Relationships

Ten class axioms were established using the object property `causedBy`:

```
Allergy SubClassOf causedBy some Peanut
Allergy SubClassOf causedBy some TreeNuts
Allergy SubClassOf causedBy some Milk
Allergy SubClassOf causedBy some Egg
Allergy SubClassOf causedBy some Fish
Allergy SubClassOf causedBy some Shellfish
Allergy SubClassOf causedBy some Soy
Allergy SubClassOf causedBy some Wheat
Allergy SubClassOf causedBy some Sesame
Allergy SubClassOf causedBy some Celery
```

2.4.4 Graphical Evidence

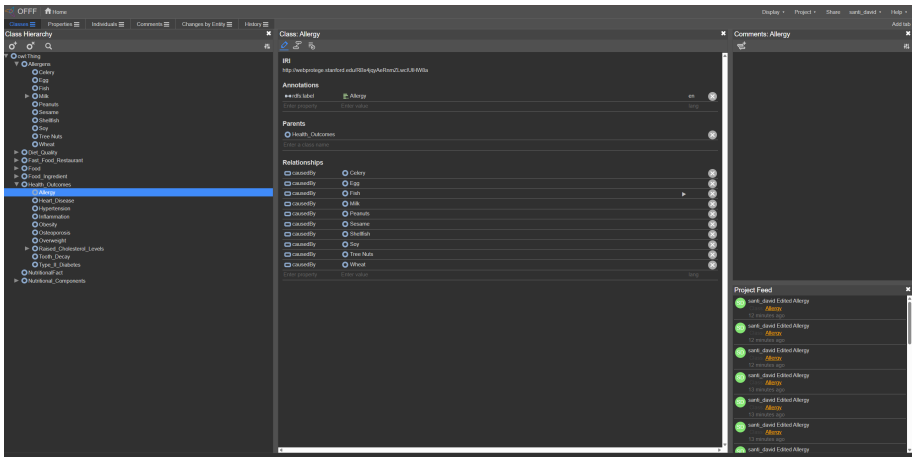


Figure 2: Allergy class showing the list of `causedBy` axioms in WebProtégé's class description view. All 10 allergen relationships are visible. Username `santi_david` visible.

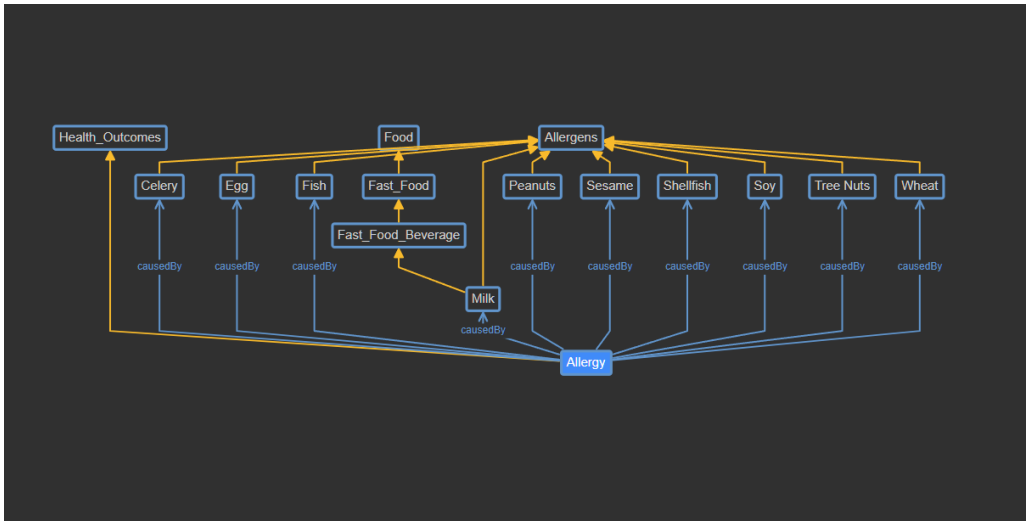


Figure 3: Entity graph visualization of the Allergy class showing the network of causal relationships. The graph displays Allergy (center) connected to all 10 allergen subclasses through `causedBy` relationships, providing a visual representation of the ontology structure. Username `santi_david` visible.

2.4.5 Technical Explanation

- **Object property:** `causedBy` is an existing object property in OFFF that relates `Health_Outcomes` to `Diet_Quality` or food components
- **Existential restriction (some):** Indicates that there exists at least one causal relationship
- **Semantics:** "An allergy is a health outcome that can be caused by any of the defined allergens"

- **Reasoning:** A reasoner can infer that if a food contains `Peanut`, it can cause `Allergy`

2.5 Inflammation Class in Health_Outcomes

2.5.1 Objective

Create the `Inflammation` class and establish that various chronic diseases are caused by inflammatory processes.

2.5.2 Creation Process

Similar to the previous process:

1. Create `Inflammation` as a subclass of `Health_Outcomes`
2. Establish inverse causal relationships with 4 existing health conditions

2.5.3 Causal Relationships

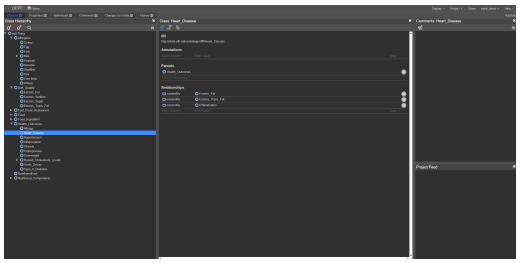
It was established that the following conditions are caused by inflammation:

1. `Heart_Disease`: Cardiovascular diseases
2. `Hypertension`: Arterial hypertension
3. `Obesity`: Obesity
4. `Type_II_Diabetes`: Type 2 diabetes

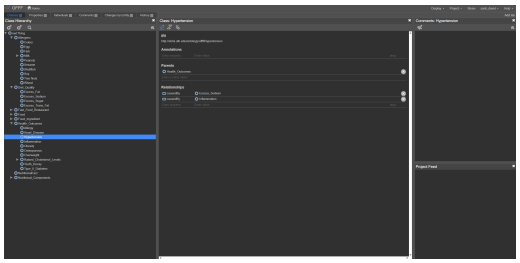
The established axioms were:

```
Heart_Disease SubClassOf causedBy some Inflammation
Hypertension SubClassOf causedBy some Inflammation
Obesity SubClassOf causedBy some Inflammation
Type_II_Diabetes SubClassOf causedBy some Inflammation
```

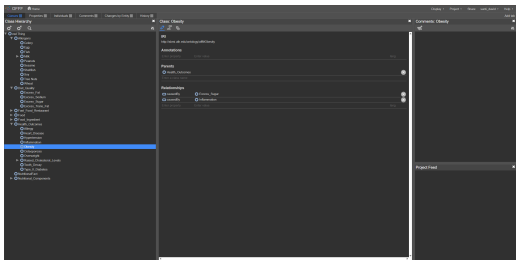
2.5.4 Graphical Evidence



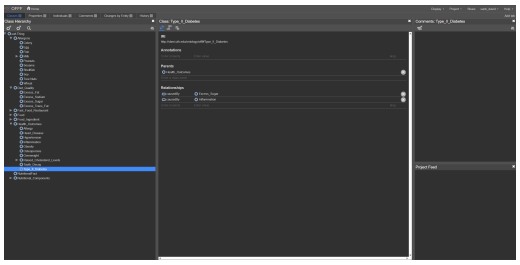
(a) Relationship with Heart Disease



(b) Relationship with Hypertension



(c) Relationship with Obesity



(d) Relationship with Type II Diabetes

Figure 4: Inflammation class and its causal relationships with chronic diseases. Username `santi_david` visible in all screenshots.

2.5.5 Technical Explanation

- **Relationship direction:** Established from diseases toward `Inflammation` using `causedBy`
- **Interpretation:** These diseases are results of chronic inflammatory processes
- **Causal chain:** Allows reasoning about chains: Diet → Inflammation → Disease

2.6 Individual "My fish sandwich"

2.6.1 Objective

Create a concrete instance of a food product (`Fish_Sandwich`) with specific properties related to allergens.

2.6.2 Creation Process

1. **Locate class:** Find `Fish_Sandwich` in the hierarchy (subclass of `Fast_Food`)
2. **Create instance:** Go to the "Individuals" tab and create a new individual
3. **Assign type:** Set `Fish_Sandwich` as the class type
4. **Add data properties:** Assign values to data properties

2.6.3 Assigned Data Properties

1. **containsGluten:** `true`
   - Type: `xsd:boolean`
   - Meaning: The sandwich contains gluten (from bread)
2. **hasAllergen:** `''Peanut Oil''`
   - Type: `xsd:string`
   - Meaning: Peanut oil was used in preparation

2.6.4 Graphical Evidence

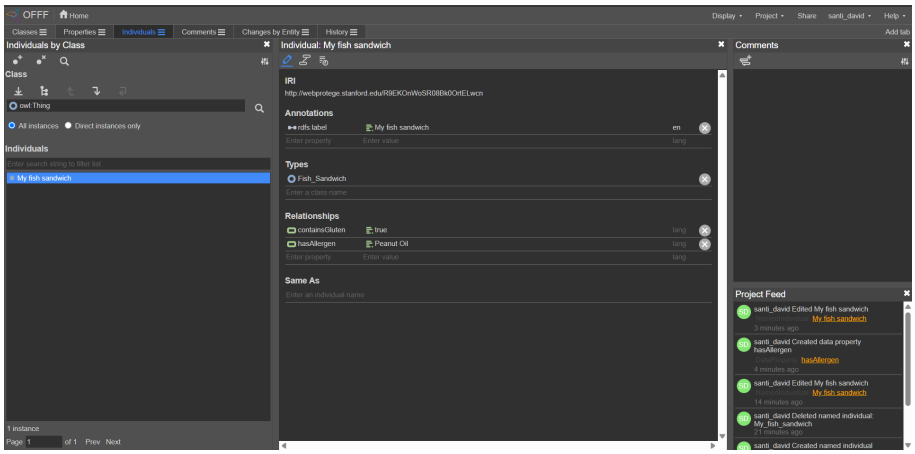


Figure 5: Individual "My fish sandwich" showing its data properties in WebProtégé's property values view. The two assigned properties (`containsGluten` and `hasAllergen`) are clearly visible. Username `santi_david` visible.

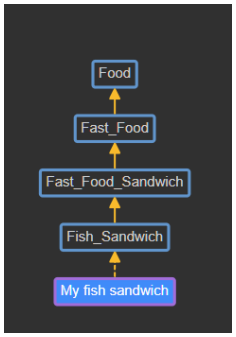


Figure 6: Entity graph visualization of the "My fish sandwich" individual. The graph shows the relationships between the individual, its class type (`Fish_Sandwich`), and its connections to allergen-related entities, providing a network view of how this instance relates to the ontology structure. Username `santi_david` visible.

2.6.5 Technical Explanation

- **Instance vs Class:** This is a concrete individual, not a category

- **ABox vs TBox:** Instances are part of the ABox (data), while classes form the TBox (terminology)
- **Data properties:** Connect individuals with literal values (strings, numbers, booleans)
- **Inference:** A reasoner can infer that this sandwich may cause allergies in people sensitive to peanuts or gluten
- **Practical utility:** Allows labeling specific products with allergen information for recommendation systems

2.6.6 Implications

This individual exemplifies how ontologies can be used to:

- Label products with allergen information
- Facilitate searches ("find gluten-free sandwiches")
- Warn consumers about potential risks
- Comply with food labeling regulations

3 SPARQL Queries

SPARQL (SPARQL Protocol and RDF Query Language) is the standard language for querying RDF data on the Semantic Web. This section presents three queries performed on DBpedia, the semantic version of Wikipedia.

3.1 Query 1: Artists in DBpedia

3.1.1 Objective

Obtain 10 artist names whose names begin with "Mado" using the DBpedia SPARQL endpoint.

3.1.2 Endpoint

- **URL:** https://dbpedia.org/sparql/
- **Endpoint URI:** `http://dbpedia.org`

3.1.3 SPARQL Query

Listing 1: Query 1: Artists starting with "Mado"

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

SELECT DISTINCT ?artist ?name
WHERE {
```

```
  ?artist a dbo:Artist .
  ?artist rdfs:label ?name .
  FILTER (lang(?name) = 'en')
  FILTER (STRSTARTS(?name, "Mado"))
}
LIMIT 10
```

3.1.4 Detailed Explanation

1. **PREFIX declarations:**
   - `dbo:` DBpedia Ontology (`http://dbpedia.org/ontology/`)
   - `rdfs:` RDF Schema (`http://www.w3.org/2000/01/rdf-schema#`)
2. **SELECT DISTINCT:** Selects unique values of variables `?artist` and `?name`, avoiding duplicates
3. **Triple patterns:**
   - `?artist a dbo:Artist`: Filters only entities of type `Artist`
   - `?artist rdfs:label ?name`: Gets the label (name) of the artist
4. **FILTER (lang(?name) = 'en'):** Filters only labels in English language
5. **FILTER (STRSTARTS(?name, "Mado")):** String function that verifies if the name starts with "Mado"
6. **LIMIT 10:** Limits results to 10 artists

3.1.5 Results

SPARQL | HTML5 table

artist	name
http://dbpedia.org/resource/Mado_Maurin	"Mado Maurin"@en
http://dbpedia.org/resource/Madoka_Asahina	"Madoka Asahina"@en
http://dbpedia.org/resource/Madoka_Sugai	"Madoka Sugai"@en
http://dbpedia.org/resource/Madoka_Takagi	"Madoka Takagi"@en
http://dbpedia.org/resource/Madoka_Yonezawa	"Madoka Yonezawa"@en
http://dbpedia.org/resource/Madokoro_Akutagawa_Saori	"Madokoro Akutagawa Saori"@en
http://dbpedia.org/resource/Madonna_Fitta_de_Milano	"Madonna Fitta de Milano"@en
http://dbpedia.org/resource/Madonna_Tassi	"Madonna Tassi"@en
http://dbpedia.org/resource/Madonna_Wayne_Gacy	"Madonna Wayne Gacy"@en
http://dbpedia.org/resource/Madoka_Kimura	"Madoka Kimura"@en

Figure 7: Results of Query 1 executed on DBpedia SPARQL Endpoint

Table 1: Artists starting with "Mado"

Artist URI	Name
http://dbpedia.org/resource/Mado_Maurin	Mado Maurin
http://dbpedia.org/resource/Madoka_Asahina	Madoka Asahina
http://dbpedia.org/resource/Madoka_Sugai	Madoka Sugai
http://dbpedia.org/resource/Madoka_Takagi	Madoka Takagi
http://dbpedia.org/resource/Madoka_Yonezawa	Madoka Yonezawa
http://dbpedia.org/resource/Madokoro_Akutagawa_Saori	Madokoro Akutagawa Saori
http://dbpedia.org/resource/Madonna_Fitta_de_Milano	Madonna Fitta de Milano
http://dbpedia.org/resource/Madonna_Tassi	Madonna Tassi
http://dbpedia.org/resource/Madonna_Wayne_Gacy	Madonna Wayne Gacy
http://dbpedia.org/resource/Madoka_Kimura	Madoka Kimura

3.1.6 Results Analysis

The results show various artists whose names begin with "Mado". The query demonstrates:

- Text pattern filtering capability in SPARQL
- Use of string functions (`STRSTARTS`)
- Language filtering in multilingual data
- Access to DBpedia's structured ontology

3.2 Query 2: Persons by Height

3.2.1 Objective

Obtain the first 10 persons with height between 1.8 and 2.3 meters, born after 1980, ordered by birth date.

3.2.2 Endpoint

- **URL:** https://dbpedia.org/snorql/

3.2.3 SPARQL Query

Listing 2: Query 2: Persons by height and birth date

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

SELECT DISTINCT ?person ?name ?height ?birthDate
WHERE {
  ?person a dbo:Person .
  ?person rdfs:label ?name .
  ?person dbo:height ?height .
  ?person dbo:birthDate ?birthDate .

```

```
  FILTER (lang(?name) = 'en')
  FILTER (?height >= 1.8 && ?height <= 2.3)
  FILTER (YEAR(?birthDate) > 1980)
}
ORDER BY ?birthDate
LIMIT 10
```

3.2.4 Detailed Explanation

1. **Selected variables:** person, name, height, and birth date
2. **Triple patterns:**
   - `?person a dbo:Person`: Filters entities of type Person
   - `?person dbo:height ?height`: Gets the height property
   - `?person dbo:birthDate ?birthDate`: Gets birth date
3. **FILTER (?height ¿= 1.8 && ?height ¡= 2.3):**
   - Logical AND operators (`&&`)
   - Filters heights in the range [1.8m, 2.3m]
   - Heights in DBpedia are in meters
4. **FILTER (YEAR(?birthDate) ¿ 1980):**
   - `YEAR()` function extracts the year from a date
   - Filters persons born after 1980
5. **ORDER BY ?birthDate:**
   - Orders results by birth date
   - Default is ascending (oldest first)
6. **LIMIT 10:** First 10 persons meeting the criteria

3.2.5 Results

SPARQL | HTML5 table

person	name	height	birthDate
http://dbpedia.org/resource/Hüzeyfe_Doğan	"Hüzeyfe Doğan"@en	1.81	1981-01-01
http://dbpedia.org/resource/Ju_Ho-jin	"Ju Ho-jin"@en	1.82	1981-01-01
http://dbpedia.org/resource/Vladimir_Loginov_(ice_hockey)	"Vladimir Loginov (ice hockey)"@en	1.8288	1981-01-01
http://dbpedia.org/resource/Marek_Tomica	"Marek Tomica"@en	1.8288	1981-01-01
http://dbpedia.org/resource/Mladen_Petrić	"Mladen Petrić"@en	1.85	1981-01-01
http://dbpedia.org/resource/Moumouni_Dagano	"Moumouni Dagano"@en	1.86	1981-01-01
http://dbpedia.org/resource/Zenon_Konopka	"Zenon Konopka"@en	1.8288	1981-01-02
http://dbpedia.org/resource/Alexander_Krysanov	"Alexander Krysanov"@en	1.8288	1981-01-02
http://dbpedia.org/resource/Ivan_Babić_(footballer,_born_1981)	"Ivan Babić (footballer, born 1981)"@en	1.83	1981-01-02
http://dbpedia.org/resource/Bill_Nicholls_(Australian_footballer)	"Bill Nicholls (Australian footballer)"@en	1.84	1981-01-02

Figure 8: Results of Query 2 executed on DBpedia Snorql

Table 2: Persons by height (1.8–2.3 m) born after 1980

Person URI	Name	Height (m)	Birth Date
http://dbpedia.org/resource/Hzeyfe_Doan	Hüzeyfe Doğan	1.81	1981-01-01
http://dbpedia.org/resource/Ju_Ho-jin	Ju Ho-jin	1.82	1981-01-01
http://dbpedia.org/resource/Vladimir_Loginov_(ice_hockey)	Vladimir Loginov (ice hockey)	1.8288	1981-01-01
http://dbpedia.org/resource/Marek_Tomica	Marek Tomica	1.8288	1981-01-01
http://dbpedia.org/resource/Mladen_Petri	Mladen Petrić	1.85	1981-01-01
http://dbpedia.org/resource/Moumouni_Dagano	Moumouni Dagano	1.86	1981-01-01
http://dbpedia.org/resource/Zenon_Konopka	Zenon Konopka	1.8288	1981-01-02
http://dbpedia.org/resource/Alexander_Krysanov	Alexander Krysanov	1.8288	1981-01-02
http://dbpedia.org/resource/Ivan_Babi_(footballer,_born_1981)	Ivan Babić (footballer, born 1981)	1.83	1981-01-02
http://dbpedia.org/resource/Bill_Nicholls_(Australian_footballer)	Bill Nicholls (Australian footballer)	1.84	1981-01-02

3.2.6 Results Analysis

This query demonstrates:

- **Numeric filters:** Value ranges for numeric properties
- **Date functions:** Extraction of temporal components
- **Ordering:** Results ordered chronologically
- **Complex queries:** Combination of multiple criteria

The results show relatively tall persons from the post-1980 generation, likely athletes or models.

3.3 Query 3: Persons by Height/Weight and Clubs

3.3.1 Objective

Obtain persons with height $\geq$ 2.10m OR weight $\geq$ 95kg, showing the number of associated clubs, grouped by person, ordered by number of clubs (ascending), limited to 7 results.

3.3.2 Endpoint

- **URL:** https://dbpedia.org/snorql/

3.3.3 SPARQL Query

Listing 3: Query 3: Persons by height/weight and number of clubs

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

SELECT ?person ?name (COUNT(?club) as ?numberOfClubs)
WHERE {
  ?person a dbo:Person .
  ?person rdfs:label ?name .
  OPTIONAL { ?person dbo:team ?club . }

  {
    { ?person dbo:height ?height . FILTER (?height >= 2.10) }
    UNION
```

```
13      { ?person dbo:weight ?weight . FILTER (?weight >= 95000)
            }
14    }
15
16    FILTER (lang(?name) = 'en')
17  }
18  GROUP BY ?person ?name
19  ORDER BY ASC(?numberOfClubs)
20  LIMIT 7
```

3.3.4 Detailed Explanation

1. **SELECT with aggregation:**
   - `COUNT(?club) as ?numberOfClubs`: Counts the number of clubs per person
   - Aggregate function creates a new variable
2. **OPTIONAL { ?person dbo:team ?club . }:**
   - Optional pattern: doesn't discard persons without clubs
   - Allows including persons with 0 clubs
3. **UNION between height and weight:**
   - Logical OR operator: meet ANY of the conditions
   - First branch: height $\geq$ 2.10 meters
   - Second branch: weight $\geq$ 95000 grams (95 kg)
   - Note: DBpedia stores weights in grams
4. **GROUP BY ?person ?name:**
   - Groups results by person
   - Necessary for aggregation functions (COUNT)
   - Allows counting clubs per person
5. **ORDER BY ASC(?numberOfClubs):**
   - Orders by number of clubs ascending
   - `ASC`: ascending order (explicit)
   - Persons with fewer clubs appear first
6. **LIMIT 7:** First 7 persons according to the ordering criterion

3.3.5 Results


SPARQL | HTML5 table

person	name	numberOfClubs
http://dbpedia.org/resource/Ben_Lawson_(basketball)	"Ben Lawson (basketball)"@en	0
http://dbpedia.org/resource/Blair_Rasmussen	"Blair Rasmussen"@en	0
http://dbpedia.org/resource/Bob_Kurland	"Bob Kurland"@en	0
http://dbpedia.org/resource/Brian_Zoubek	"Brian Zoubek"@en	0
http://dbpedia.org/resource/John_Shasky	"John Shasky"@en	0
http://dbpedia.org/resource/Jon_Kreft	"Jon Kreft"@en	0
http://dbpedia.org/resource/Jordan_Dickerson	"Jordan Dickerson"@en	0

Figure 9: Results of Query 3 executed on DBpedia Snorql

Table 3: Persons by height/weight and number of associated clubs

Person URI	Name	Number of Clubs
http://dbpedia.org/resource/Ben_Lawson_(basketball)	Ben Lawson (basketball)	0
http://dbpedia.org/resource/Blair_Rasmussen	Blair Rasmussen	0
http://dbpedia.org/resource/Bob_Kurland	Bob Kurland	0
http://dbpedia.org/resource/Brian_Zoubek	Brian Zoubek	0
http://dbpedia.org/resource/John_Shasky	John Shasky	0
http://dbpedia.org/resource/Jon_Kreft	Jon Kreft	0
http://dbpedia.org/resource/Jordan_Dickerson	Jordan Dickerson	0

3.3.6 Results Analysis

This advanced query demonstrates:

- **Aggregation functions:** COUNT to count relationships
- **Optional patterns:** OPTIONAL to handle missing data
- **Logical operators:** UNION for alternative conditions
- **Grouping:** GROUP BY for entity-based analysis
- **Custom ordering:** ORDER BY with explicit direction

The results likely include professional athletes (basketball, American football) given the physical characteristics. The number of clubs may indicate career mobility.

4 Conclusions

4.1 Learnings about Ontologies

Through this practice, fundamental knowledge has been acquired about:

- **Ontology structure:** Understanding of class hierarchy, properties, and relationships
- **Knowledge modeling:** Ability to formally represent real-world concepts

- **Axioms and restrictions:** Use of descriptive logic to express semantic relationships
- **TBox vs ABox:** Distinction between terminology (classes) and assertions (instances)
- **Automated reasoning:** Understanding how reasoners can infer new knowledge

4.2 Experience with WebProtégé

WebProtégé proved to be a powerful tool for:

- **Collaborative editing:** Accessible web interface without local installation
- **Visualization:** Entity graphs that facilitate understanding of relationships
- **Change management:** Complete history of modifications
- **Interoperability:** Import/export of standard formats (OWL)

Difficulties encountered:

- Need to remove problematic import lines
- Learning curve in axiom syntax
- Management of existing properties vs. creating new ones

4.3 Learnings about SPARQL

SPARQL proved to be a powerful query language:

- **Expressiveness:** Ability to formulate complex queries with multiple criteria
- **Functions:** Extensive library of functions for strings, numbers, dates, aggregation
- **Optional patterns:** Flexibility to handle incomplete data
- **Federation:** Possibility to query multiple endpoints

Key concepts learned:

- Triple patterns as the basis of queries
- Importance of FILTERs to refine results
- Use of aggregation functions (COUNT, SUM, AVG, etc.)
- Logical operators (UNION, OPTIONAL, FILTER)
- Result ordering and limitation

4.4 Practical Applications

The acquired knowledge has applications in:

1. **Food recommendation systems:**
   - Suggest foods based on allergen restrictions
   - Warn about potential risks
   - Personalization according to health profiles
2. **Intelligent labeling:**
   - Automation of nutritional labels
   - Compliance with allergen regulations
   - Ingredient traceability
3. **Public health analysis:**
   - Correlation between diet and diseases
   - Identification of epidemiological patterns
   - Evidence-based health policies
4. **Semantic Web:**
   - Integration of heterogeneous data
   - Federated queries over multiple sources
   - Linked Data applications

4.5 Final Reflection

Ontologies represent a fundamental tool for knowledge representation and management in intelligent systems. The combination of well-designed ontologies (such as OFFF) with expressive query languages (such as SPARQL) allows creating sophisticated applications that can reason about complex data and provide valuable information to users.

The practical experience with WebProtégé and DBpedia has provided a tangible understanding of theoretical concepts, demonstrating the viability and utility of Semantic Web technologies in real-world applications.

5 References

1. Ding, Y., Pramanik, S., Muppala, S., et al. (2021). *The ontology of fast food facts: conceptualization of nutritional fast food data for consumers and semantic web applications*. BMC Medical Informatics and Decision Making, 21, 275. https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-016

2. UTHealth Ontology. (2021). *OFFF - Ontology of Fast Food Facts*. GitHub Repository.
https://github.com/UTHealth-Ontology/OFFF/

3. Wikipedia. (2025). *List of allergens*.
https://en.wikipedia.org/wiki/List_of_allergens

4. Stanford University. (2025). *WebProtégé: A Free, Open-Source Ontology Editor*.
https://webprotege.stanford.edu/

5. DBpedia Association. (2025). *DBpedia SPARQL Endpoint*.
https://dbpedia.org/sparql/

6. DBpedia Association. (2025). *DBpedia Snorql - SPARQL Query Interface*.
https://dbpedia.org/snorql/

7. W3C. (2013). *SPARQL 1.1 Query Language*. W3C Recommendation.
https://www.w3.org/TR/sparql11-query/

8. W3C. (2012). *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation.
https://www.w3.org/TR/owl2-overview/

9. Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). *From SHIQ and RDF to OWL: The making of a web ontology language*. Journal of Web Semantics, 1(1), 7-26.

10. Bizer, C., Heath, T., & Berners-Lee, T. (2009). *Linked Data - The Story So Far*. International Journal on Semantic Web and Information Systems, 5(3), 1-22.

A Modified Ontology File

The modified ontology is found in the file `offf_modified.owl` delivered together with this report.

A.1 Modification Summary

- **Classes added:** 12 (10 allergens + Allergy + Inflammation)
- **Relationships added:** 14 causal axioms
- **Individuals added:** 1 (My fish sandwich)
- **Data properties:** 2 assigned values

B Complete SPARQL Code

All SPARQL query files are found in the `sparql_queries/` folder:

- `query1_artists.sparql` - Artists starting with "Mado"
- `query2_persons_height.sparql` - Persons by height and birth date
- `query3_persons_clubs.sparql` - Persons by height/weight and number of clubs

C Additional Screenshots

All screenshots include the username `santi_david` from WebProtégé to validate authorship of the work.

C.1 Screenshot List

1. `Allergens.png` - Allergen hierarchy
2. `allergy.png` - Allergy class
3. `inflammation_1.png` - Relationship with Heart Disease
4. `inflammation_2.png` - Relationship with Hypertension
5. `inflammation_3.png` - Relationship with Obesity
6. `inflammation_4.png` - Relationship with Type II Diabetes
7. `My fish sandwich.png` - Individual with data properties
8. `Query1.png` - Query 1 results
9. `Qiery2.png` - Query 2 results
10. `Query3.png` - Query 3 results