Assignment 1 – Animal Scouter

University of Aveiro

Web Semântica

2022/2023

Group nº 2:

- João Bernardo Coelho Leite - 115041
- João Pedro dos Reis - 115513
- Luís Miguel Gomes Batista - 115279

Index

INTRODUCTION **2**

DATA, SOURCES AND TRANSFORMATIONS **3**

DATASET 3
Selection of serialisation format *3*
Transformations *3*
OVERVIEW 3

DATA OPERATIONS (SPARQL) **4**

SELECT 4
Get all animals from a given class *4*
Get all animals that produce a given nurturing *4*
Get all animals that have a certain amount of legs *5*
Get all stored attributes of an animal *6*
ASK 6
Get veracity of an attribute *6*
UPDATE 7
Insert *7*
Delete *7*

APPLICATION FUNCTIONALITY **8**

HOME 8
CONFIGS 8
QUERIES 9
ASK 9

CONCLUSIONS **10**

APPLICATION CONFIGURATION **11**

REQUIREMENTS 11
CREATING THE DATABASE 11
RUNNING WITH PYCHARM 11
RUNNING WITH COMMAND LINE 11

REFERENCES **12**

Introduction

The Semantic Web is an extension of the World Wide Web that allows data to be shared and reused across different applications, platforms, and systems. It is based on the principle of creating machine-readable data that can be easily understood and processed by computers. One of the key components of the Semantic Web is the use of linked data, which refers to a set of best practices for publishing and connecting structured data on the web.

In this report, we will discuss the conversion of a database from CSV (Comma Separated Variables) to N-Triples, a *Triplestore* database format, using a Python script. We will also explore the use of SPARQL, a query language used to retrieve and manipulate data stored in RDF formats. Additionally, we will showcase the features of a website made with Django that has been created to demonstrate the capabilities of the Semantic Web, including the ability to query and retrieve data using SPARQL.

The report will begin by describing the data and process of converting a database from CSV to N-Triples, and the advantages of using a *Triplestore* database. It will also be explained the basics of SPARQL and how it can be used to query and retrieve data from *Triplestore* databases.

Finally, the website that has been created to showcase the features of the Semantic Web will be discussed. The website will include a demonstration of how to use it, as well as examples of how the data can be visualized and analysed.

Overall, this report aims to provide an introduction to the Semantic Web, and to demonstrate the benefits of using linked data and *Triplestore* databases for managing and sharing structured data on the web.

Data, sources and transformations

Dataset

The Zoo Animals dataset used in this Semantic Web analysis was sourced from Kaggle, a popular platform for data science and machine learning projects. The dataset consists of two separate CSV files, each containing 18 columns of data related to various attributes of different zoo animals.

Selection of serialisation format

Before any transformations there was an important decision to make regarding what type of data should be used: NT, N3 or RDF/XML. It ultimately was decided that NT (N-Triples) was the right choice for this project. The NT format is a widely accepted and standardized format for representing data in Semantic Web, it is designed to represent structured data providing ease of querying and analysis. Lastly, it's very optimised for scalability being able to handle large amounts of data efficiently.

Transformations

In order to prepare the dataset for use in the Semantic Web, several transformations were applied to the data. Firstly, the two CSV files were merged into a single file to simplify the data and make it easier to work with. Next, a script was created to automatically convert the merged dataset into the NT (N-Triples) format, which is a standard format for representing data in the Semantic Web. The script generates unique IDs for the "class type", "animal name", and "type of nurturing" attributes in order to be able to name them separately to one another. These IDs served to create triples consisting of a subject, predicate, and object as per the example:

<http://zoo.org/animal/id/turtle> <http://zoo.org/pred/name> "Turtle" .

Example 1 Triple where a subject "id" has "name" which is "Turtle"

In addition, among the list of possible attributes present in the original csv dataset format, some interesting characteristics were chosen to enrich the sematic of the processed file: *legs*, *tail*, *fins*, *feathers*, *hair*, *domestic*, *venomous*, *toothed*, *airborne* and *aquatic*.

Overview

After the data was converted to the NT format, it was loaded into a *Triplestore* stored in GraphDB, which is a type of database designed for storing RDF data. This allowed for efficient querying using SPARQL and analysis of the data, as well as integration with other Semantic Web tools and technologies.

Overall, the use of the Zoo Animals dataset allowed for a robust and interesting analysis of the data using Semantic Web techniques. By applying transformations to the data and converting it into the NT format, it was possible to unlock valuable insights and information about the various attributes of different zoo animals.

Data Operations (SPARQL)

SELECT

Get all animals from a given class

Given a class ID as an object related to a subject animal ID, this same ID is used to retrieve the name of the animal names related to this class type.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix class: <http://zoo.org/class/id/>
select ?animal_name
where {
    ?animal_id pred:class class:_class_id.
    ?animal_id pred:name ?animal_name.
}
```

Figure 1 Select query to get all animals from a category

Get all animals that produce a given nurturing

Given a nurturing ID as an object related to a subject animal ID, this same ID is used to retrieve the name of the animal names related to this nurture type of either “Eggs” or “Milk”.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix nurt: <http://zoo.org/nurt/id/>
select ?animal_name
where {
    ?animal_id pred:nurt nurt:_nurt_id.
    ?animal_id pred:name ?animal_name.
}
```

Figure 2 Select query to get all animals that produce milk or eggs

If there are animals able to produce both types of nurturing a new query was made to intersect the previously mentioned queries and retrieve the name of all animals within that group.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix nurt: <http://zoo.org/nurt/id/>
select ?animal_name
where {
    {
        ?animal_id pred:nurt nurt:1.
        ?animal_id pred:name ?animal_name.
    }
    {
        ?animal_id pred:nurt nurt:2.
        ?animal_id pred:name ?animal_name.
    }
}
```

Figure 3 Select query to get all animals that produce eggs and milk

Get all animals that have a certain amount of legs

Given a number of legs as an object related to a subject animal ID, this same ID is used to retrieve the name of the animal names with the given number of legs.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix class: <http://zoo.org/class/id/>
select ?animal_name
where {
    ?animal_id pred:legs ?legs_number.
    ?animal_id pred:name ?animal_name.
    filter(?legs_number = "_legs_number").
}
```

Figure 4 Select query to get all animals that have a specific number of legs

Since 0 values have not been stored, in order to select the animals that have no legs, all of them are selected and then a filter is used to find only the animals that don't have that feature in relation to their respective ID. This, however, also retrieved class names and another filter had to be add to exclude those results.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix class: <http://zoo.org/class/id/>
select ?animal_name
where {
    ?animal_id pred:name ?animal_name.
    filter NOT EXISTS { ?animal_id pred:legs ?legs_number. }.
    filter EXISTS { ?animal_id pred:class ?class_name. }.
}
```

Figure 5 Select query to get all animals with no legs

Get all stored attributes of an animal

Given the animals name its ID was used to retrieve all related data to that animal.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix class: <http://zoo.org/class/id/>
select ?animal_id ?p ?o
where {
    ?animal_id pred:name "_animal_name".
    ?animal_id ?p ?o.
}
```

Figure 6 Select query to get all attributes of a specific animal

For both nurture and class predicates, two extra queries have to be made to retrieve their specific names.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix class: <http://zoo.org/class/id/>
select ?class_name
where {
    class:_class_id pred:name ?class_name.
}
```

Figure 7 Select query to retrieve name of class through ID

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
prefix nurt: <http://zoo.org/nurt/id/>
select ?nurt_name
where {
    nurt:_nurt_id pred:name ?nurt_name.
}
```

Figure 8 Select query to retrieve name of nurture through ID

ASK

Get veracity of an attribute

Given an animal name and an attribute or feature it will return a boolean result of whether that animal has a certain attribute or if it is of a certain type.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
ask {
    ?animal_s pred:name "_animal_name".
    ?animal_s pred:_pred "_animal_attribute".
}
```

Figure 9 Ask query to verify or deny existence of an attribute

UPDATE

Insert

With an input of an animal name and a category selected, multiple other attributes can be selected as well to add a new entry to the dataset.

```
base <http://zoo.org/>
prefix id: <http://zoo.org/animal/id/>
prefix pred: <http://zoo.org/pred/>
prefix class: <http://zoo.org/class/id/>
prefix nurt: <http://zoo.org/nurt/id/>
insert data {
    id:_name_id pred:name "_animal_name".
    id:_name_id pred:class class:_class_id.
    id:_name_id pred:is "Domestic".
    id:_name_id pred:is "Airborne".
    id:_name_id pred:nurt nurt:1.
    id:_name_id pred:has "Feathers".
    # (...)
}
```

Figure 10 Insert query to add a new entry to dataset

Delete

This method will look for an animal ID in relation to a provided name, which, once identified, any and all predicates and respective objects associated with this ID will be deleted.

```
base <http://zoo.org/>
prefix pred: <http://zoo.org/pred/>
delete { ?s ?p ?o }
where {
    ?s pred:name "_animal_name".
    ?s ?p ?o.
}
```

Figure 11 Delete query to delete all entries related to an animal name

Application Functionality

Home

The "home" tab provides users with an introduction to the website's purpose and content. It includes a brief description of the three other tabs: configs, queries, and ask, along with links to each tab. This makes it easy for users to navigate to the tab that best suits their needs. The "home" tab serves as a landing page that welcomes users and gives them a high-level understanding of the website's functionalities.

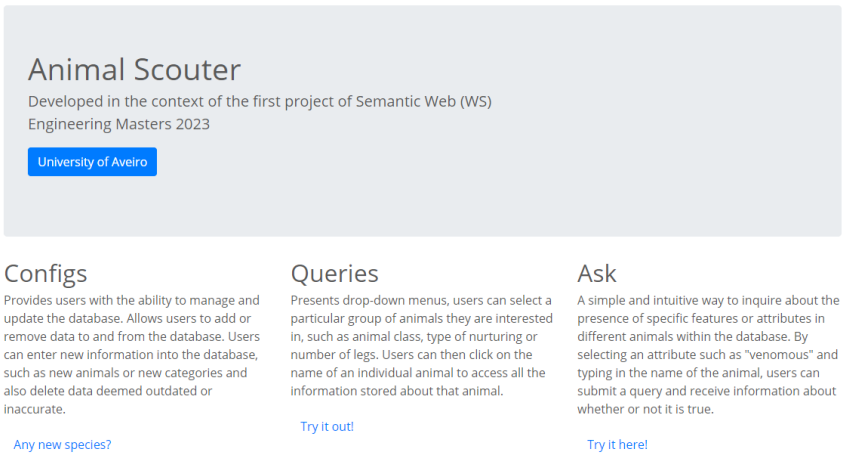

Figure 12 Application home page

Configs

The configs tab on our website is designed for admin users and allows them to update the database by deleting or adding entries to it. This functionality is crucial for maintaining an up-to-date and accurate database.

Admin users can add new entries to the database, along with relevant metadata and annotations, ensuring that the new data is properly integrated and can be easily queried by users.

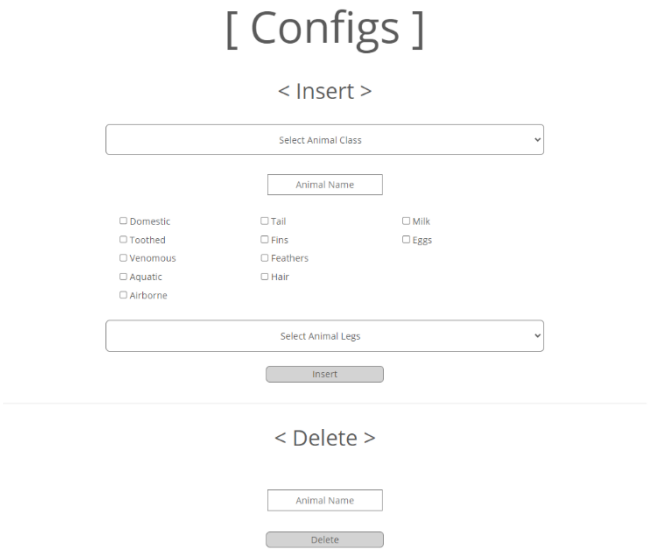

Figure 13 Application configs page

Queries

The queries tab on our website is designed to help users retrieve information from the database using select queries. It allows users to select a category of animals, features, or the number of legs to get a list of animals with that respective trait. This feature enables users to easily filter through the database and retrieve the specific information they are looking for.

Additionally, if a user is looking for information on a specific animal, they can click on the animal's name in the list, and the website will display all information available for that animal.

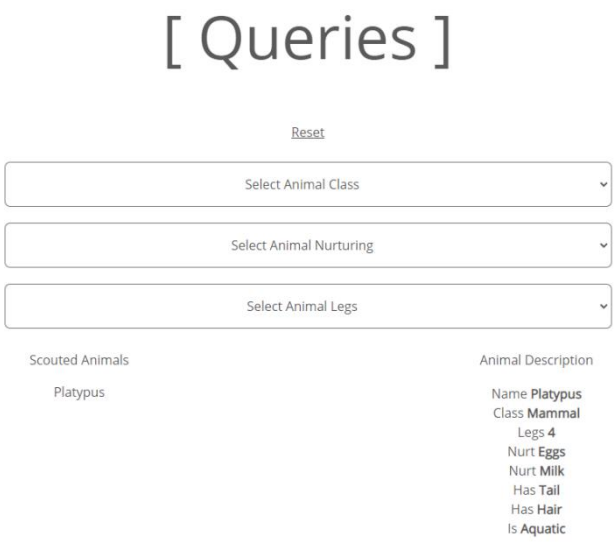

Figure 14 Application queries page

Ask

The Ask tab on our website is a powerful feature that enables users to ask questions about the animals in the database by using ASK queries. The user can type in the name of an animal and select a feature, and the website will show whether that feature is true or false for that animal.

By using ASK queries, the Ask tab allows for efficient and accurate retrieval of information from the database. This feature can be particularly helpful for users who need to verify specific information quickly.

Figure 15 Application ask page

Conclusions

In conclusion, using SPARQL in our project has been an incredibly helpful experience in expanding our knowledge and understanding of the Semantic Web. By transforming our dataset from CSV to N-Triples and uploading it to *GraphDB*, we were able to utilize SPARQL to query the data in a structured and efficient manner.

Developing a website that leverages SPARQL has allowed us to explore the full potential of the Semantic Web, making it easier for users to access and retrieve information from the database. The tabs we created on our website, including the configs, queries, and ask tabs, have enabled users to perform a variety of tasks, such as updating the database, filtering through animals based on features, and retrieving specific information through ASK queries.

Through this project, we have gained a deeper understanding of the power and versatility of SPARQL in accessing, querying, and managing large-scale datasets. This experience has given us a strong foundation in the principles and practices of the Semantic Web, which we can apply in future projects and research endeavours.

In conclusion, our project has not only expanded our knowledge of SPARQL and the Semantic Web but has also allowed us to develop a practical tool that can be used by others to access and retrieve information about animals in an efficient and structured manner.

Application Configuration

Requirements

Installation requirements to set the developed application up and running:

- Python (preferably 3.8.10 or higher)
- GraphDB
- s4api (pip install s4api)

Creating the database

In GrapDB control panel, it is needed to create a database named “zoo” and import the provided “zoo.nt” N-Triples file.

Optionally, for the base url, *http://zoo.org/* may be used.

Running with PyCharm

To run the application with PyCharm, simply open the wsproject folder and press the run button. Then, a localhost link should appear in the console which just needs to be opened with the web browser.

Running with command line

For running the application using the command line, open a new command line in the “/wsproject/” directory and type the command “py manage.py runserver”. Once again, a localhost link should appear in the console which just needs to be opened with the web browser.

References

Dataset

https://www.kaggle.com/datasets/agajorte/zoo-animals-extended-dataset

Slides

Representação do Conhecimento, Standards da Web Semântica, WS, DETI, UA

Representação do Conhecimento, A Linguagem SPARQL, WS, DETI, UA

Representação do Conhecimento, A *Triplestore* GraphDB, WS, DETI, UA

Web references

https://docs.djangoproject.com/en/4.1/ref/forms/widgets/

https://docs.djangoproject.com/en/4.1/ref/forms/fields/