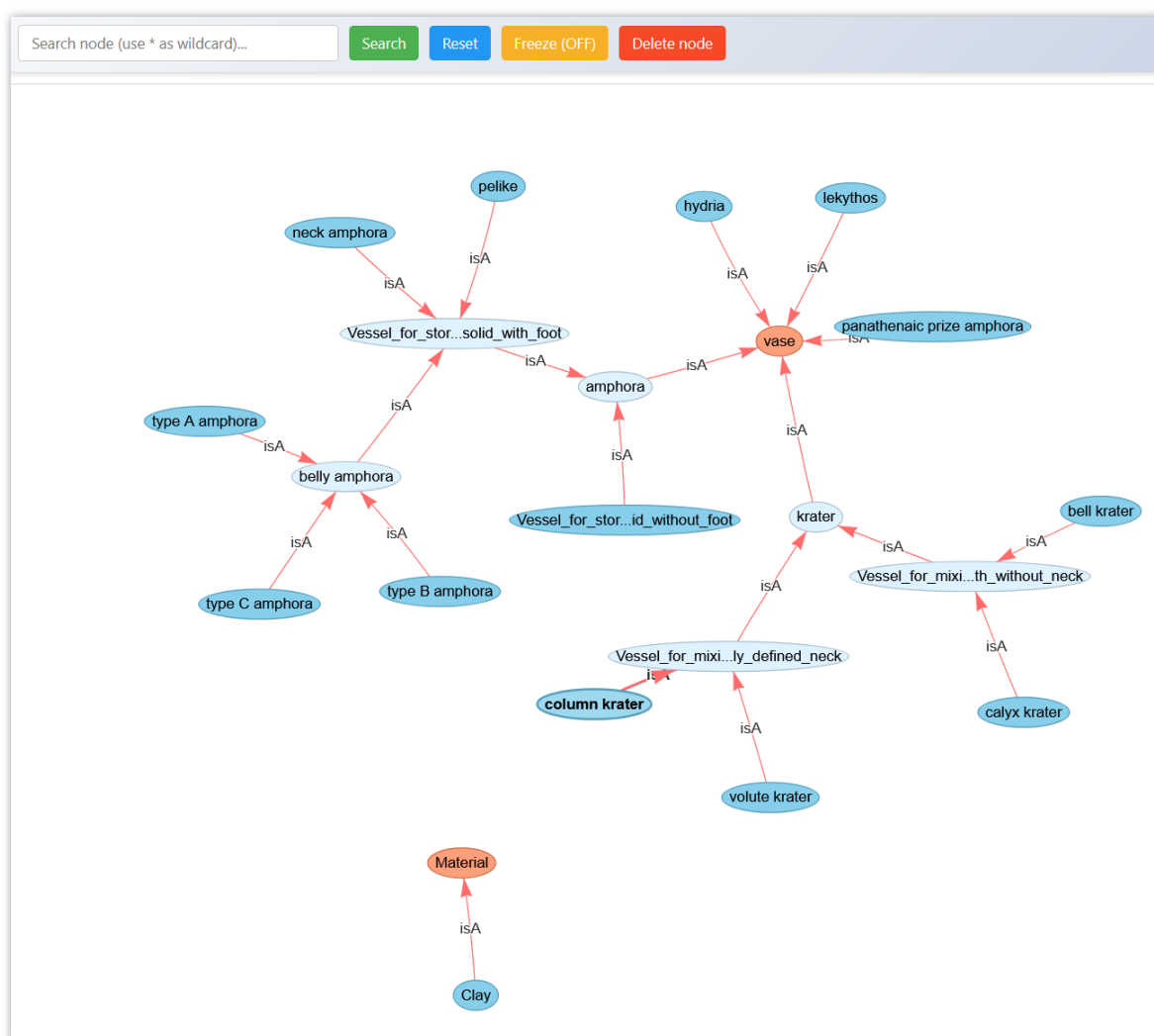


# TALOS Software

## “RDF GRAPH Viewer”

### Installation and Usage Guide



Version 1.0

17/08/2025

## Contents

|   |    |
|---|----|
| 1. Software Identification .....                      | 3  |
| 2. General Description .....                          | 3  |
| 3. Core Functionalities .....                         | 3  |
| 4. Installation & Launch .....                        | 3  |
| 4.1. System Requirements .....                        | 3  |
| 4.2 Install Python .....                              | 4  |
| Windows: .....  | 4  |
| macOS: .....  | 4  |
| Linux (Ubuntu/Debian): .....                          | 4  |
| 4.3 Install Required Python Libraries .....           | 4  |
| Core Libraries (Required) .....                       | 4  |
| Alternative: Install Everything at Once .....         | 4  |
| 4.4 Download the Talos_Text_Analyzer.py Program ..... | 4  |
| 4.5 Launch the Program TALOS_Text_Analyzer.py .....   | 5  |
| 4.5 Troubleshooting .....                             | 5  |
| Common Issues: .....                                  | 5  |
| Performance Tips: .....                               | 5  |
| 5. Updates .....                                      | 6  |
| 6. How to Use Talos_RDF_Viewer.py .....               | 6  |
| 6.2 Basic Workflow: .....                             | 6  |
| 6.3 Supported File Formats: .....                     | 8  |
| 7. Use Case .....                                     | 8  |
| 7.1 Example .....                                     | 8  |
| 7.2 Select RDF File .....                             | 9  |
| 7.2 Upload and Analyze .....                          | 10 |
| 7.3 Show Metadata .....                               | 10 |
| 7.4 Select Properties .....                           | 12 |
| 7.5 View Graph .....                                  | 13 |
| Node Labelling Priority .....                         | 16 |
| Label Truncation .....                                | 16 |
| Node Tooltips .....                                   | 16 |
| Node Coloring System .....                            | 16 |
| Edge Visualization .....                              | 16 |
| Search Scope .....                                    | 16 |
| Pattern Matching .....                                | 16 |
| Search Results .....                                  | 17 |
| Freeze/Unfreeze Toggle .....                          | 17 |
| Reset Function .....                                  | 17 |
| Node Deletion .....                                   | 17 |
| Responsive Design .....                               | 17 |
| 7.6 SPARQL Endpoint .....                             | 18 |
| 8. Technical Implementation Details .....             | 19 |
| 9. Best Practices .....                               | 19 |
| 10. 5-Minute Quick Start Guide .....                  | 20 |
| 11. Developer .....                                   | 21 |

|                          |    |
|--------------------------|----|
| 12. License .....        | 21 |
| 13. Download links ..... | 21 |
| 14. Support .....        | 21 |

## 1. Software Identification

- **Name:** RDF Graph Viewer
- **Program:** Talos\_RDF\_Viewer.py
- **Version:** 1.0
- **Date:** 17/08/2025
- **Language:** Python 3.12
- **Type:** Software (standalone)
- **Topic:** RDF Graph Visualization
- **Web site:** [http://talos-ai4ssh.eu/RDF\\_Viewer/](http://talos-ai4ssh.eu/RDF_Viewer/)
- **Author:** Christophe Roche

## 2. General Description

Talos RDF Graph Viewer is a lightweight, browser-based application for visualizing and exploring RDF graphs.

It allows users to upload RDF files in XML format, automatically extracts metadata and graph statistics. It also lets users select which RDF properties to visualize, and provides an interactive network graph with navigation and search tools.

Talos RDF Graph Viewer includes specific features that allow the display of ontoterminologies<sup>1</sup> built with the TEDI<sup>2</sup> environment.

It also includes a built-in SPARQL endpoint for querying RDF data directly from the interface.

## 3. Core Functionalities

- Graph Visualisation
- SPARQL Graph Endpoint

## 4. Installation & Launch

### 4.1. System Requirements

- **Operating System:** Windows 10 or later, macOS 10.15 or later, or Linux (Ubuntu/Debian recommended)
- **Python Version:** Python 3.8 or later
- **RAM:** At least 4 GB (8 GB recommended for large RDF graphs)

---

<sup>1</sup> <https://ontoterminology.com/>

<sup>2</sup> TEDI is a software environment dedicated to ontoterminology building. TEDI is freely distributed by University of Crete <https://ontoterminology.com/tedi>

- **Disk Space:** At least 2GB free space for Python and libraries
- **Web Browser:** Any modern browser (Chrome, Firefox, Edge, Safari)

## 4.2 Install Python

### Windows:

1. Download the latest Python installer from: <https://www.python.org/downloads/>
2. Run the installer and **check the box “Add Python to PATH”** before clicking **Install Now**.
3. Verify installation: `python --version`

### macOS:

1. Install using Homebrew (recommended): `brew install python`  
Or download from <https://www.python.org/downloads/>
2. Verify installation: `python3 --version`

### Linux (Ubuntu/Debian):

1. Update your package index: `sudo apt update`
2. Install Python: `sudo apt install python3 python3-pip`
3. Verify installation: `python3 --version`

## 4.3 Install Required Python Libraries

Open Terminal/Command Prompt and run the following **commands**:

### Core Libraries (Required)

```
pip install flask rdflib pyvis
```

### Alternative: Install Everything at Once

Create a file named `requirements.txt` with the following content:

```
flask
```

```
rdflib
```

```
pyvis
```

Then run:

```
pip install -r requirements.txt
```

## 4.4 Download the Talos\_Text\_Analyzer.py Program

- Save the `Talos_RDF_Viewer.py` file from the TALOS official repository to your computer
- Choose a dedicated folder (e.g., Documents/TALOS/RDF\_Viewer)

## 4.5 Launch the Program TALOS\_Text\_Analyzer.py

### Method 1 - Command Line:

1. Open a Terminal
2. Change the current folder: `cd /path/to/your/TALOS/RDF_Viewer`
3. Run the program: `python Talos_RDF_Viewer.py`

### Method 2 - Double-click (Windows):

- Right-click on `Talos_RDF_Viewer.py`
- Select “Open with” → “Python”

### Method 3 - Create a Shortcut (Windows):

- Create a .bat file with:

```
@echo off
cd "C:\path\to\your\TALOS\RDF_Viewer"
python Talos_RDF_Viewer.py
pause
```

After a few seconds, your default browser will open automatically at: `http://127.0.0.1:5000`

If it does not open automatically, manually copy-paste the above URL into your browser.

## 4.5 Troubleshooting

### Common Issues:

#### 1. “Python is not recognized”

Solution: Reinstall Python and check “Add Python to PATH”

#### 2. “Module not found” errors - A required library is missing

Solution: Run `pip install [module_name]` for each missing module

#### 3. Check Dependencies:

Solution: Ensure all required libraries are installed

#### 4. Update Libraries: Run `pip install --upgrade [library_name]`

#### 5. Port already in use:

Solution: If port 5000 is occupied, modify the last line in the script:

```
app.run(port=NEW_PORT, debug=False)
```

Then launch the program again and access it at `http://127.0.0.1:NEW_PORT`.

### Performance Tips:

- Use smaller RDF files if visualization becomes slow
- Select the properties to visualize
- For ontoterminologies generated by TEDI, select only the necessary vocabularies when exporting to RDF

- Close unused browser tabs while running the viewer.
- Increase available RAM for large datasets.

## 5. Updates

To update `Talos_RDF_Viewer.py` :

1. Replace the `Talos_RDF_Viewer.py` file with the latest version
2. Update dependencies: `pip install --upgrade flask rdflib pyvis`

## 6. How to Use Talos\_RDF\_Viewer.py

The next chapter, dedicated to a use case, details each of the software's functions


### 6.2 Basic Workflow:

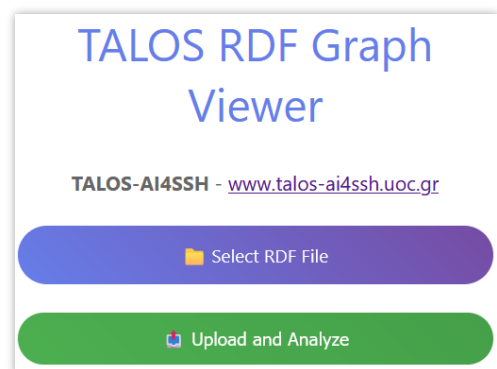
#### 1. Home Page

#### 2. Select RDF File

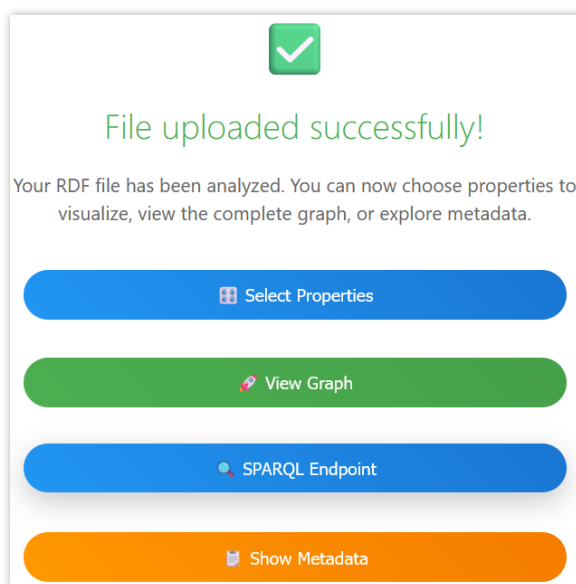
- Click  **Select RDF File** to choose an .rdf, .xml, .ttl, or .jsonld file.





#### 3. Upload and Analyse

- Click  Upload and Analyze.





#### 4. Upload Success Page



-  **Select Properties** to choose which RDF properties to visualize.
-  **View Graph** to directly generate the graph with all properties (in general, this is not necessary and can be extremely time-consuming).
-  **SPARQL Endpoint** to run SPARQL queries on your data.
-  **Show Metadata** to view basic graph statistics, Dublin Core metadata, and declared namespaces

## 5. Select Properties

- Tick the properties you want to include in the visualization.
- Click  **Generate Graph** at the bottom of the window

 **Object Properties (42)** Select All

☐ <http://www.ontologia.fr/OTB/KraterAll#foundAt>

☐ <http://www.ontologia.fr/OTB/otv#allonym>


☐ <http://www.ontologia.fr/OTB/otv#attribute>

☐ <http://www.ontologia.fr/OTB/otv#belongsToAxis>

☐ <http://www.ontologia.fr/OTB/otv#containsDifference>

☐ <http://www.ontologia.fr/OTB/otv#denotedByProperName>

☐ <http://www.ontologia.fr/OTB/otv#denotedByTerm>


 **Annotation Properties (4)** Select All

☐ <http://www.w3.org/2000/01/rdf-schema#comment>

☐ <http://www.w3.org/2000/01/rdf-schema#label>

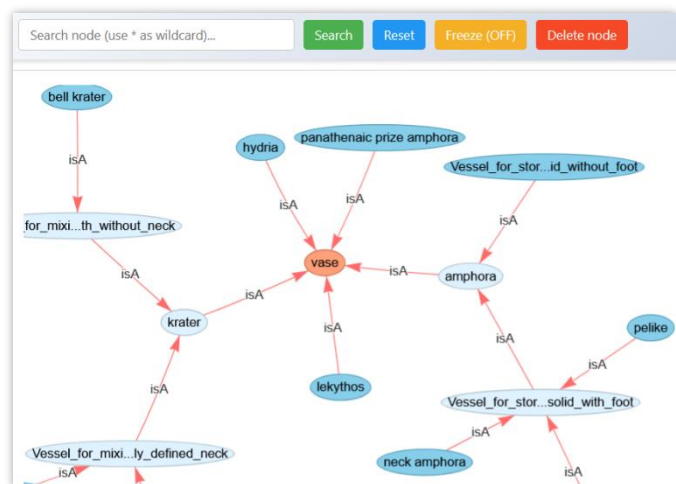
☐ <http://www.w3.org/2004/02/skos/core#altLabel>

☐ <http://www.w3.org/2004/02/skos/core#prefLabel>

 **Generate Graph**

## 6. View Graph

- Interactive **visualization** with draggable nodes.
- **Search** nodes containing a substring
- **Reset** to return to the default view.
- **Freeze** to lock node positions.
- **Delete node** to remove a node and its edges.



## 7. SPARQL Endpoint

- Write and execute SPARQL queries.
- View results in a table format.
- Use example queries provided in the interface.


**SPARQL Query:**

SELECT DISTINCT ?label WHERE { ?s a ?type. ?s rdfs:label ?label }

Execute Query

Clear

Examples

 **Query Results (301 rows)**

| label                 |
|-----------------------|
| bell krater           |
| cratère en cloche     |
| κρατήρας κωδωνόσχημος |

## 6.3 Supported File Formats:

- RDF files in various formats (.rdf, .xml, .ttl, .jsonld)

## 7. Use Case

The TALOS RDF Viewer is an interactive web application designed to upload, explore, and query RDF graphs through a clear and user-friendly interface. This tool provides comprehensive functionality for analyzing ontological structures, relationships, and metadata within RDF documents. It is organized into several functional pages, each corresponding to a stage in the workflow. Below we present the functionality of each page in detail.

### 7.1 Example

We will take as an example the ontoterminology<sup>3</sup> of kraters, vases from ancient Greece intended for mixing water and wine. The data, both for building the ontology and populating it, as well as for constructing the terminology, are drawn from the Beazley Archive<sup>4</sup>.

Below, we provide the definitions of the four types of kraters together with a graphical representation of the ontoterminology, illustrating the three levels: concepts, terms, and objects.

*The term 'krater' suggests a mixing-vessel (compare Greek kerannumi - to mix), and we know that the wine served at the symposium was mixed with water:*



**Column-krater:** Named for its column-like handles, the column-krater is first known from Corinthian examples dated to the late seventh century. It is regularly produced by Athenian potters from the first half of the sixth-century until the third quarter of the fifth. It seems from graffiti on Athenian red-figure examples that the vessel was referred to as *Korinthios* or *Korinthiourges*.



**Volute-krater:** The volute-krater is named after its handles. The François Vase is a famous and early example, but the typical Athenian form occurs only later in the sixth century, with the handles tightly curled so that they look like the volutes on Ionic columns. The shape is also found in metal. Over the course of the fifth and fourth centuries, examples become slimmer, and Apulian volute-kraters from South Italy are particularly elaborate.



**Calyx-krater:** The handles of the calyx-krater are placed low down on the body, at what is termed the *cul*. Their upward curling form lends the shape an appearance reminiscent of the calyx of a flower, hence the name. The earliest known example was possibly made by Exekias in the third quarter of the sixth century. It continues to be produced, mainly in red-figure, becoming more elongated over the course of the fifth and fourth centuries.



**Bell-krater:** The latest of the four krater-types, it first occurs in the early fifth century, and is not found decorated in black-figure. It is named for its bell-like shape, perhaps originating in wood. It has small horizontal upturned handles just over halfway up the body. Some do not have a foot, and earlier examples may have lugs for handles.

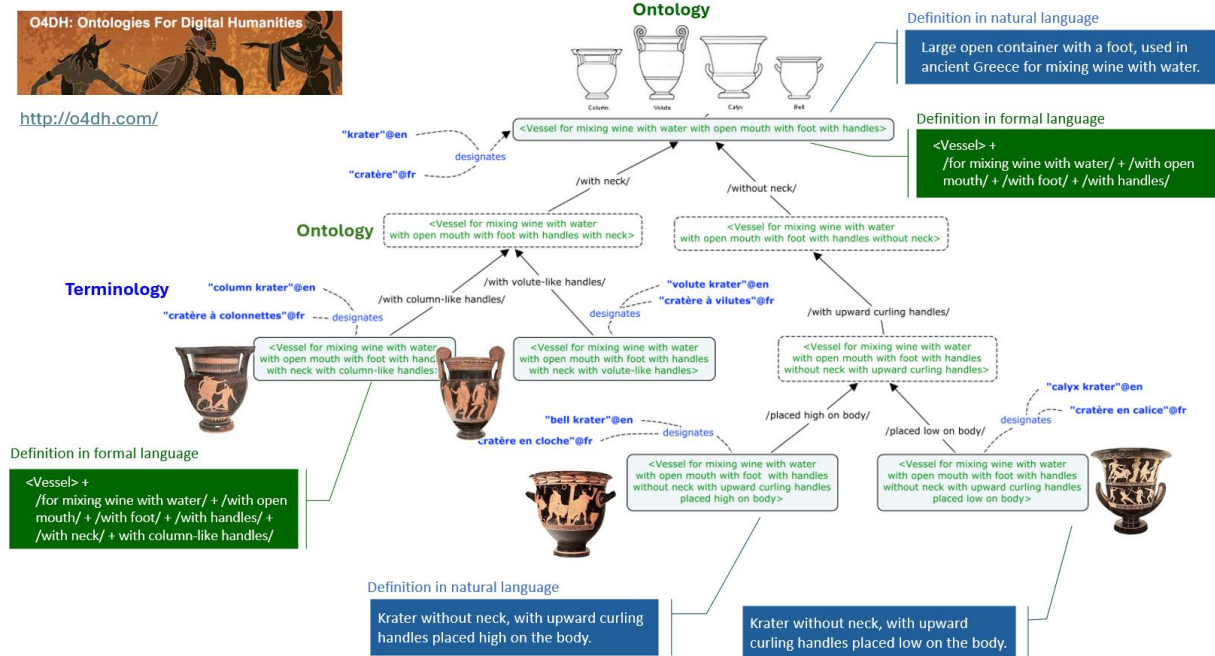
<sup>3</sup> An ontoterminology is a terminology whose conceptual system is a formal ontology: <https://ontoterminology.com/>

<sup>4</sup> Beazley Archive: <https://www.carc.ox.ac.uk/carc/Home>

The ontoterminology of kraters was built with TEDI<sup>5</sup>, an ontoterminology development environment freely distributed by the University of Crete, and exported in RDF format. Its representation relies on the RDF, RDFS, SKOS, OWL, OntoLex-Lemon, and OTV vocabularies. The richness of such an ontology, which moreover includes “punning”<sup>6</sup>, makes its visualization difficult with standard RDF graph visualization tools, thus motivating the development of the TALOS\_RDF\_Viewer program.

The Krater ontoterminology can be downloaded at: <http://ontologia.fr/OTB/Krater/krater.rdf>

More information is available at: <http://o4dh.com/>



## 7.2 Select RDF File

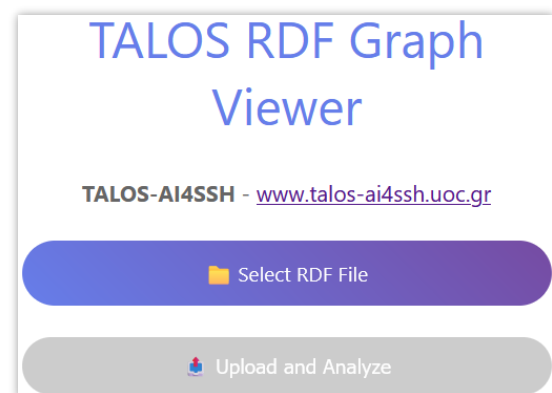
### Select your RDF file using the file picker

The system automatically detects the file format based on the extension

### Supported Formats

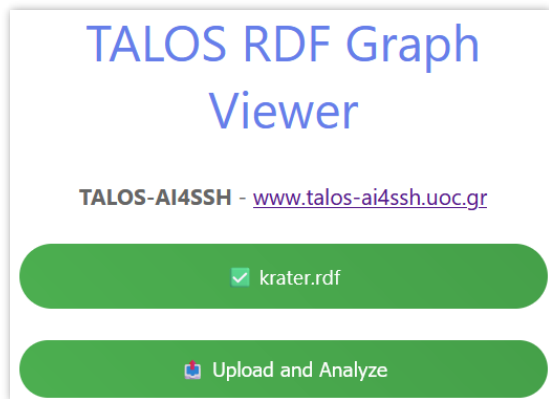
The application supports the following RDF serialization formats:

- RDF/XML (.rdf, .xml)
- Turtle (.ttl)
- JSON-LD (.jsonld)



<sup>5</sup> <https://ontoterminology.com/tedi>

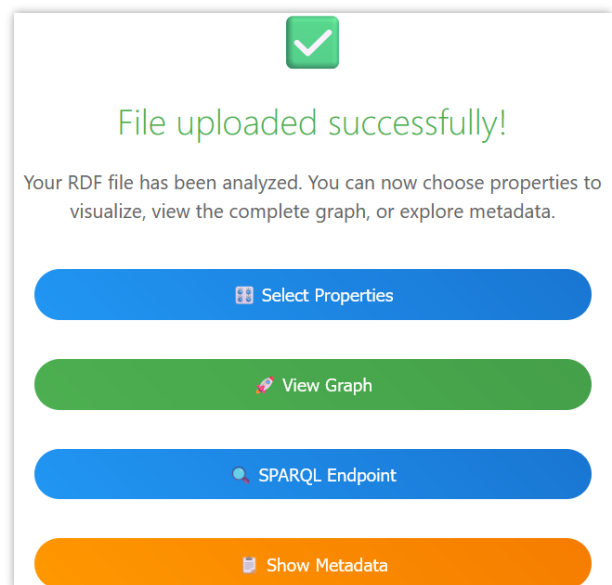
<sup>6</sup> Punning allows to treat classes as instances: <https://www.w3.org/2007/OWL/wiki/Punning>



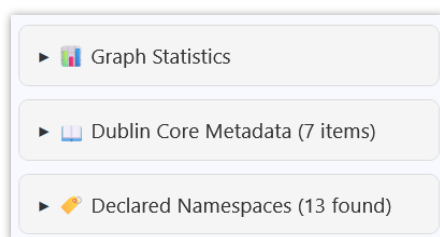
## 7.2 Upload and Analyze

Upon successful upload, the file is parsed and analyzed to extract:

- Graph statistics (total triples, property counts)
- Dublin Core metadata from owl:Ontology declarations
- Declared namespaces from the RDF header.



## 7.3 Show Metadata

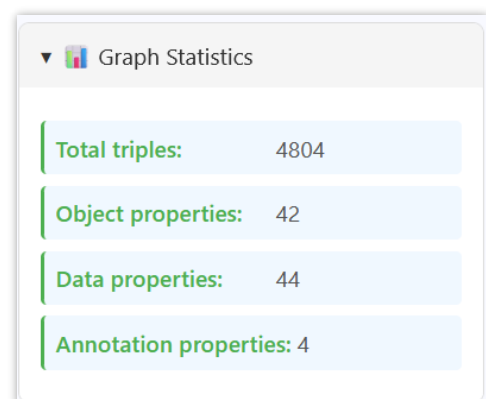


The metadata includes:

### Graph Statistics

Automatic calculation of:

- Total number of triples
- Object properties count (relationships between URIs)
- Data properties count (relationships to literals)
- Annotation properties count (labels, comments, etc.)



## Dublin Core Metadata

▼ Dublin Core Metadata (7 items)

**dc:creator**  
Christophe Roche

**dc:date**  
2019-04-28

**dc:description**  
Archaic and Classical Periods, from -700 BC to -323. The concepts of the ontology are linked with a skos:exactMatch to Getty Vocabularies, Wikidata and Kerameikos. The ontology of kraters is populated with all the kraters of the collection of the National Archeological Museum (NAM) in Athens representing medata from the Beazley Archive Pottery Database. Each object is linked with a skos:ExactMatch to its web description in Beazley.

**dc:issued**  
16 août 2025

The system specifically extracts Dublin Core metadata only from owl:Ontology declarations.

It searches for properties under the <http://purl.org/dc/elements/1.1/> namespace and displays them with the dc: prefix format.

**dc:modified**  
2025-08-13

**dc:publisher**  
Condillac

**dc:title**  
Ontoterminology of Vases

## Namespace Declaration

The application extracts namespaces declared in the RDF/XML header using pattern matching on xmlns attributes within the <rdf:RDF> tag. This ensures only explicitly declared namespaces are displayed, not inferred ones.

▼ Declared Namespaces (13 found)

**dc:** <http://purl.org/dc/elements/1.1/>

**default:** <http://www.ontologia.fr/OTB/krater#>

**foaf:** <http://xmlns.com/foaf/0.1/>

**ontolex:** <http://www.w3.org/ns/lemon/ontolex#>

**otv:** <http://www.ontologia.fr/OTB/otv#>

**owl:** <http://www.w3.org/2002/07/owl#>

## 7.4 Select Properties

### Property Classification

The system automatically categorizes properties into three types:

#### Object Properties

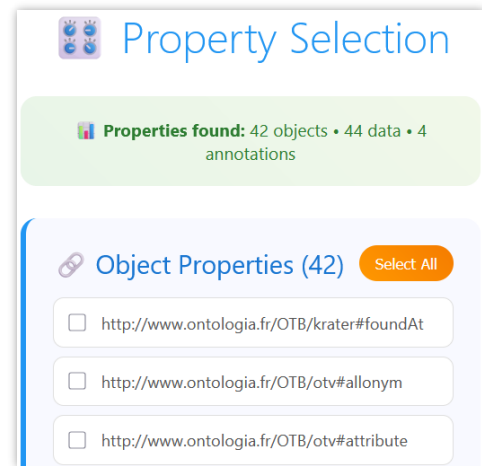
- Properties where the object is a URI reference (URIRef)
- Represent relationships between resources
- Displayed with a "🔗" icon

#### Data Properties

- Properties where the object is a literal value
- Represent attributes with string, numeric, or other literal values
- Displayed with a "📄" icon

#### Annotation Properties

- Properties identified by containing "label" or "comment" in their URI
- Used for human-readable descriptions and labels
- Displayed with a "💬" icon

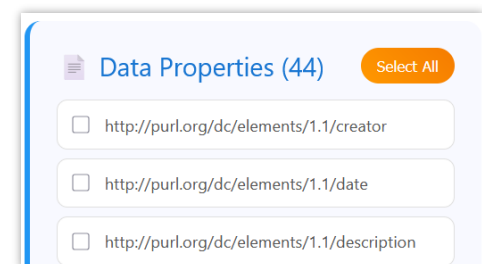


**Property Selection**

Properties found: 42 objects • 44 data • 4 annotations

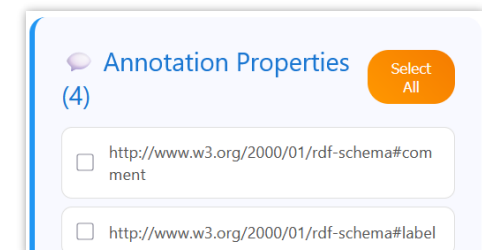
**Object Properties (42)** Select All

- ☐ <http://www.ontologia.fr/OTB/krater#foundAt>
- ☐ <http://www.ontologia.fr/OTB/otv#allonym>
- ☐ <http://www.ontologia.fr/OTB/otv#attribute>



**Data Properties (44)** Select All

- ☐ <http://purl.org/dc/elements/1.1/creator>
- ☐ <http://purl.org/dc/elements/1.1/date>
- ☐ <http://purl.org/dc/elements/1.1/description>



**Annotation Properties (4)** Select All

- ☐ <http://www.w3.org/2000/01/rdf-schema#comment>
- ☐ <http://www.w3.org/2000/01/rdf-schema#label>

### Selection Features

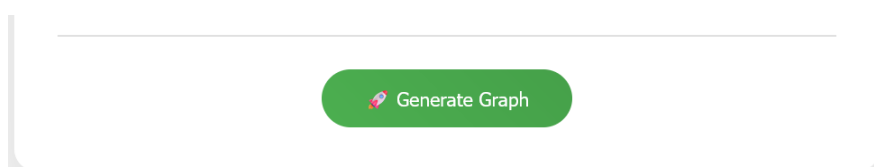


Selecting all properties can cause **significant computation time**.

Select only the necessary properties.

- **Individual selection:** Check/uncheck specific properties
- **Global selection:** "Select All" or "Deselect All" button for each category

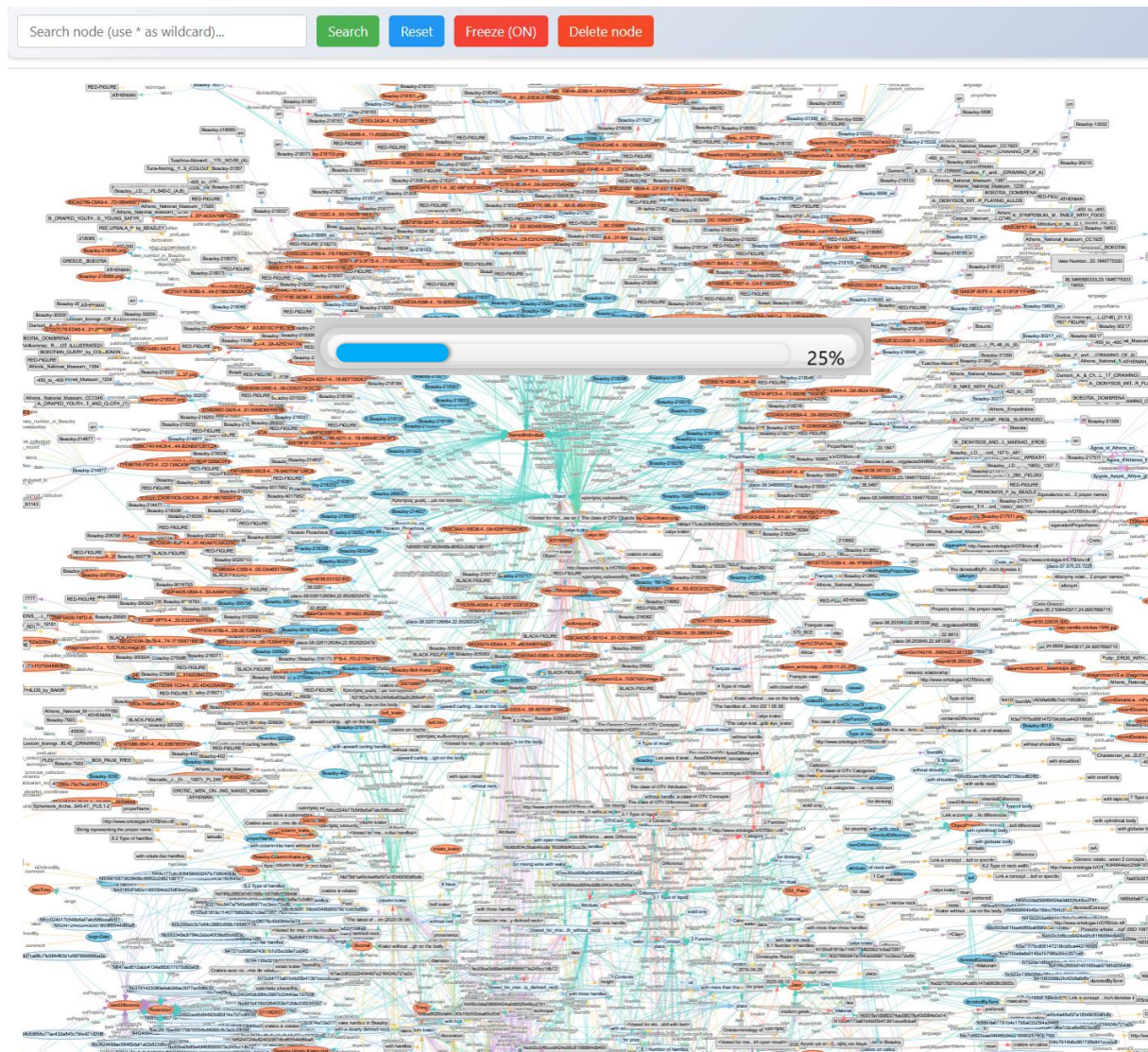
The graph generation button is at the bottom of the page



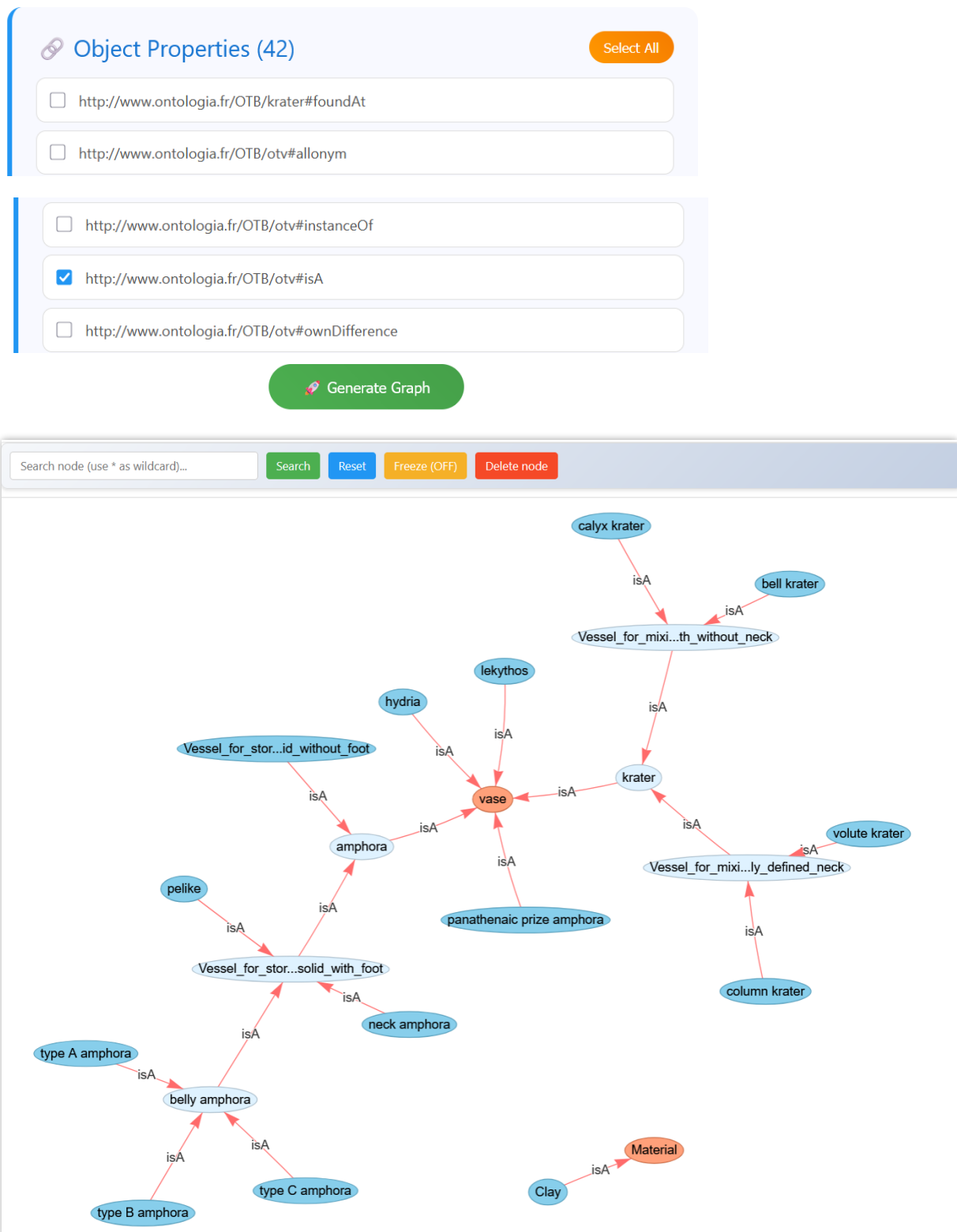
## 7.5 View Graph



It is important to select only the relationships you want to visualize **BEFORE** launching the graph visualization.



Example of displaying the ontology of the vase by selecting the property `otv:isA`<sup>7</sup>



<sup>7</sup> `isA` is the generic relationship between two concepts. `isA` is defined in the OTV vocabulary:  
<http://www.ontologia.fr/OTB/otv.rdf>

Example of displaying the ontology of the vase by selecting the properties `otv:isA` and `otv:denotedByTerm`<sup>8</sup>

☐ <http://www.ontologia.fr/OTB/otv#denotedByProperName>

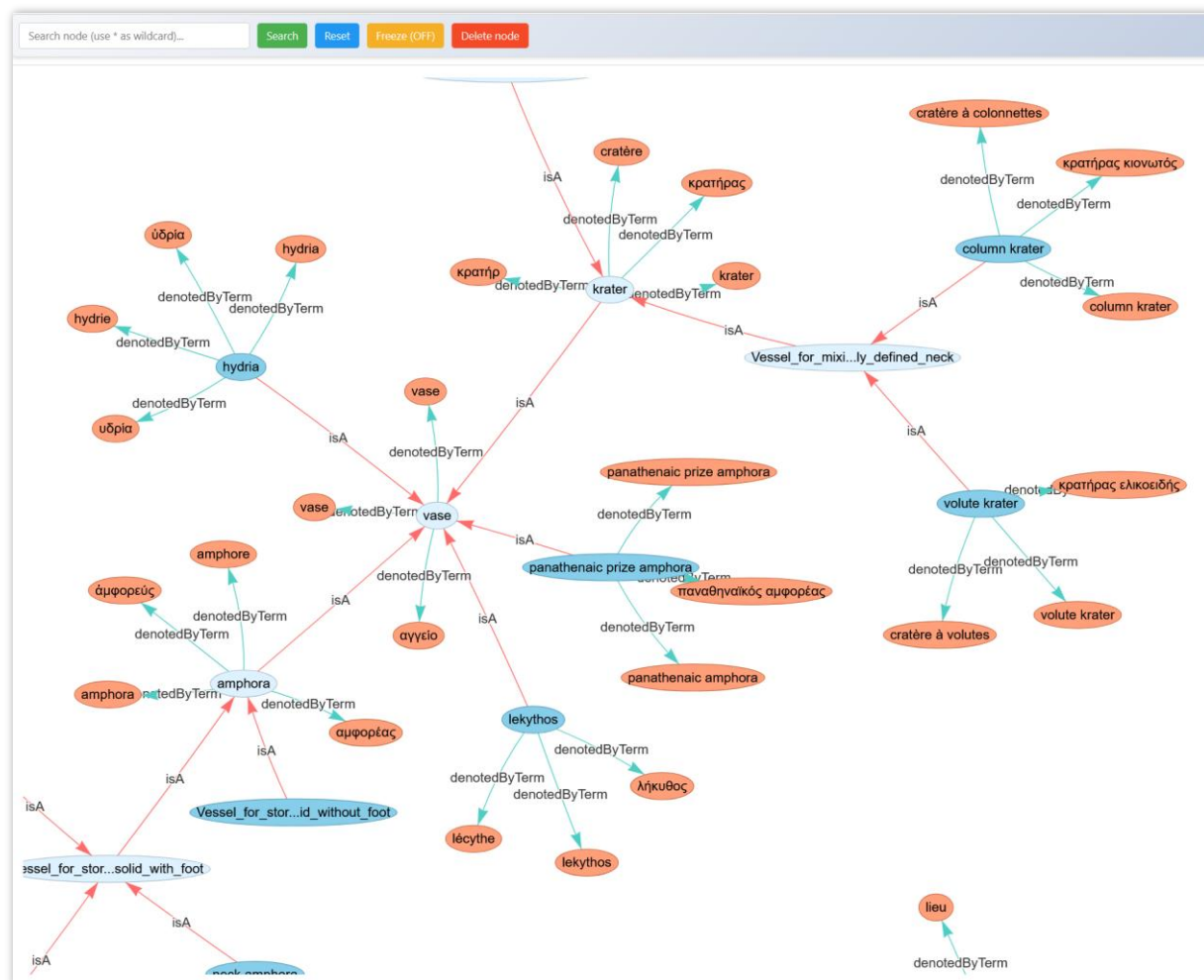
☒ <http://www.ontologia.fr/OTB/otv#denotedByTerm>

☐ <http://www.ontologia.fr/OTB/otv#denotedConcept>

☐ <http://www.ontologia.fr/OTB/otv#instanceOf>

☒ <http://www.ontologia.fr/OTB/otv#isA>

☐ <http://www.ontologia.fr/OTB/otv#ownDifference>



<sup>8</sup> denotedByTerm is the relationship between a concept and the term denoting the concept. denotedByTerm is defined in the OTV vocabulary: <http://www.ontologia.fr/OTB/otv.rdf>

## Node Labelling Priority

The system uses a hierarchical approach to determine node labels:

1. **First Priority:** otv:shortConceptName from the OTV vocabulary
2. **Second Priority:** rdfs:label (standard RDF Schema label)
3. **Fallback:** Final part of the URI (after # or /, whichever comes last)

## Label Truncation

- Labels longer than 35 characters are truncated
- Format: First 15 characters + "..." + Last 15 characters
- This ensures readability while preserving important information




## Node Tooltips

Comprehensive tooltips display:

- **Concept Name:** If otv:conceptName is available
- **Full URI:** Complete resource identifier
- **Literal Values:** Full content for data properties

## Node Coloring System

The visualization uses different colors to represent node types in the graph structure:

- **Root Nodes** (Light Salmon  #FFA07A): Resources that appear as objects but never as subjects
- **Terminal Nodes** (Light Steel Blue  #87CEEB): Resources that appear as subjects but never as objects
- **Intermediate Nodes** (Light Blue  #DFF2FF): Resources that appear both as subjects and objects
- **Literal Nodes** (Light Gray #E0E0E0): Data values (strings, numbers, dates, etc.)

## Edge Visualization

- **Color Coding:** Each property type gets a unique color from a predefined palette
- **Labels:** Property names are simplified (final URI fragment)
- **Tooltips:** Full property URIs on hover
- **Direction:** Arrows indicate the direction of relationships

**The search feature provides powerful node discovery capabilities:**

## Search Scope

The search operates on both:

- **Node labels** (visible text in the graph)
- **Node titles** (tooltip content including URIs and concept names)

## Pattern Matching

- **Wildcard Support:** Use \* as a wildcard character (e.g., concept\* matches all nodes starting with "concept")
- **Case Insensitive:** Search is not case-sensitive
- **Regular Expression:** Wildcards are converted to regex patterns for flexible matching

## Search Results

- **Visual Highlighting:** Matching nodes are highlighted in yellow with bold text (click inside the window)
- **Selection:** All matching nodes are automatically selected
- **Auto-Focus:** The view automatically centers on the found nodes
- **Result Count:** Displays the number of matches found

## Physics and Layout Control

### Freeze/Unfreeze Toggle

- **Physics ON (Freeze OFF):**
  - Automatic layout calculation and node positioning
  - Dynamic movement and stabilization
  - Automatic centering after operations
  - Button shows "Freeze (OFF)" in orange
- **Physics OFF (Freeze ON):**
  - Static layout with manual node positioning
  - Drag-and-drop node movement
  - No automatic recentering
  - Button shows "Freeze (ON)" in red

### Reset Function

- Clears all search terms
- Reloads the entire page to restore original graph state
- Resets all visual modifications (colors, selections, positions)

## Node Management

### Node Deletion


- **Selection Required:** Must select exactly one node
- **Confirmation Dialog:** Prevents accidental deletions
- **Cascade Removal:** Automatically removes all connected edges
- **Network Update:** Redraws the graph after deletion

### Responsive Design

- **Auto-Resize:** Graph adapts to browser window size changes
- **Full-Screen:** Optimized for maximum viewing area
- **Mobile Compatible:** Responsive controls and layout

## 7.6 SPARQL Endpoint

Talos\_RDF\_Viewer provides a SPARQL Endpoint for querying the uploaded RDF graph. It includes examples of SPARQL queries.



### SPARQL Endpoint

**SPARQL Query:**

```


PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX otv: <http://www.ontologia.fr/OTB/otv#>

SELECT ?shortLabel ?definition

WHERE {
  ?cpt rdf:type otv:Concept.
  ?cpt otv:shortConceptName ?shortLabel.
  ?cpt skos:definition ?definition
  FILTER (lang(?definition)='en')
}

ORDER BY ?shortLabel
          
```

▶ Execute Query
🗑 Clear
💡 Examples



### Query Results (16 rows)

| shortLabel    | definition   |
|---------------|--|
| amphora       | Vase with neck and two handles for storing and transport liquids and solids.                             |
| bell krater   | Krater without neck, with upward curling handles placed high on the body.                                |
| belly amphora | Amphora with foot, without a clearly defined neck.   |
| calyx krater  | Krater without neck, with upward curling handles placed low on the body.                                 |
| column krater | Krater with a clearly defined neck and column-like handles.  |
| hydria        | Vase with neck and three handles, two for carrying and one for pouring, for storing and transport water. |
| krater        | Large open container with a foot and handles, used in ancient Greece for mixing wine with water.         |

## 8. Technical Implementation Details

### Graph Processing

- **RDFLib Integration:** Uses Python's RDFLib for robust RDF parsing
- **Memory Efficiency:** Temporary file handling for uploaded content
- **Format Detection:** Automatic format recognition based on file extensions

### Visualization Engine

- **PyVis Network:** Leverages PyVis for interactive graph generation
- **HTML5 Output:** Generates standalone HTML files with embedded JavaScript
- **Physics Simulation:** Configurable physics engine for dynamic layouts

### Web Framework

- **Flask Backend:** Lightweight Python web server
- **Template Engine:** Dynamic HTML generation with Jinja2
- **File Handling:** Secure temporary file management
- **Cross-Platform:** Compatible with Windows, macOS, and Linux

## 9. Best Practices

### File Preparation

- Ensure RDF files are well-formed and valid
- Use standard namespaces for better compatibility
- Include `owl:Ontology` declarations for metadata extraction

### Performance Optimization

- For large graphs (>1000 nodes), consider using property filtering
- Use the SPARQL endpoint for targeted data exploration
- Leverage the freeze function for manual layout optimization

### Visualization Tips

- Start with a subset of properties to understand graph structure
- Use search functionality to locate specific concepts
- Combine freeze mode with manual positioning for presentation layouts
- Utilize color coding to understand node relationships and hierarchy

## 10. 5-Minute Quick Start Guide

This Quick Start Guide will help you install, launch, and use the TALOS RDF Graph Viewer in under 5 minutes.

### 1. Install Python 3.8 or Later

- Windows: Download from python.org, check 'Add to PATH'.
- macOS: Install with Homebrew (brew install python).
- Linux: sudo apt install python3 python3-pip

### 2. Install Required Libraries

Open a terminal and run:

```
pip install flask rdflib pyvis
```

### 3. Download the Program

Place Talos\_RDF\_Viewer.py in a folder of your choice.

### 4. Launch the Program

In a terminal, navigate to the folder and run:

```
python Talos_RDF_Viewer.py
```

Your browser will open automatically at <http://127.0.0.1:5000>

### 5. Using the Viewer

- Select and upload an RDF file (.rdf, .xml, .ttl, .jsonld).
- View metadata, namespaces, and statistics.
- Select properties or view all.
- Interact with the graph: search, freeze, delete nodes.
- Run SPARQL queries in the SPARQL endpoint.

### 6. Troubleshooting

- Missing module: `pip install .`
- Port conflict: Change port in last line of script.
- Large files: Use smaller datasets for better performance.

## 11. Developer

- Prof Christophe Roche – TALOS ERA Chair Holder – University of Crete  
<https://talos-ai4ssh.uoc.gr/>
- Contact: [roche.university@gmail.com](mailto:roche.university@gmail.com)

## 12. License

- Creative Commons CC-BY-NC<sup>9</sup> 

## 13. Download links

- Web site : [http://talos-ai4ssh.eu/RDF\\_View/](http://talos-ai4ssh.eu/RDF_View/)
- Source code: [http://talos-ai4ssh.eu/RDF\\_View/Talos\\_RDF\\_View.zip](http://talos-ai4ssh.eu/RDF_View/Talos_RDF_View.zip)
- Documentation:  
[http://talos-ai4ssh.eu/RDF\\_View/TALOS\\_RDF\\_View\\_Documentation.pdf](http://talos-ai4ssh.eu/RDF_View/TALOS_RDF_View_Documentation.pdf)

## 14. Support

- Prof Christophe Roche – TALOS ERA Chair Holder – University of Crete  
<https://talos-ai4ssh.uoc.gr/>
- Contact: [roche.university@gmail.com](mailto:roche.university@gmail.com)

*End of document*

---

---

<sup>9</sup> This license enables reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator.  
<https://creativecommons.org/share-your-work/cclicenses/>