Team 1: Knowledge Graphs

Final Presentation, Fall'23, CS-5604, Information Storage and Retrieval

Presented On - 11/30/2023

Students

- Daniel Gaugler (danielg19@vt.edu)
- Ashish Aggarwal (ashish1702@vt.edu)

SME

- Satvik Chekuri (satvikchekuri@vt.edu)

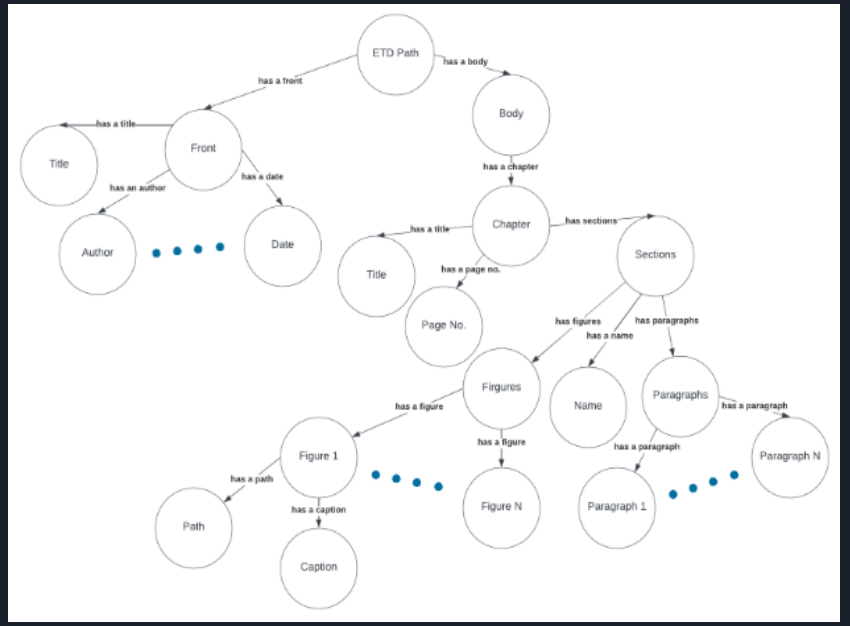
Virginia Tech, Blacksburg, VA 24061

Introduction

- Within an ETD, there are many data elements such as:
  - Document metadata
  - Chapters
  - Tables
  - Figures
  - etc.
- Currently even with these objects stored in the database there is no effective way to determine the relationships between them.
  - There is not process to iterate through these elements even if they are closely related.
  - For example: the caption of a given figure, the chapter the figure and caption are located in, or even the ETD the elements originated from
- Our team will be creating knowledge graphs to help build these relationships

What are Knowledge Graphs

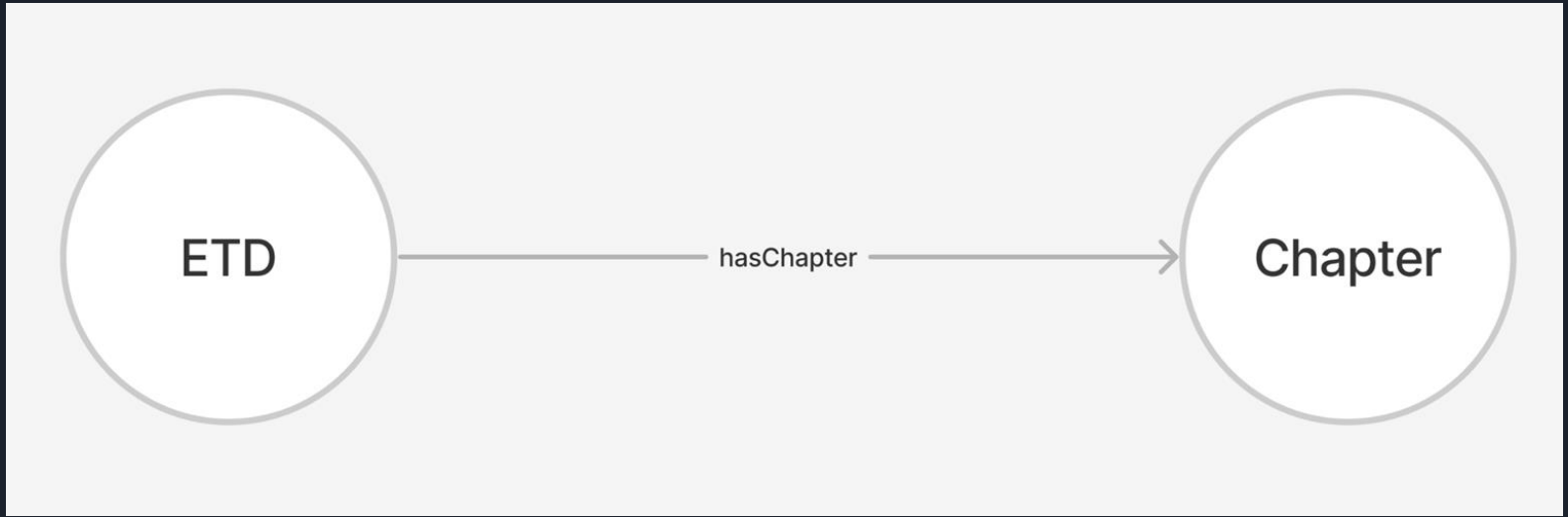
- Knowledge graphs are structured data representations for computer processing.
- Knowledge graphs are composed of nodes (entities) and edges (relationships).
  - Nodes and edges have unique identifiers (e.g., URIs).
- Knowledge graphs utilize RDF (Resource Description Framework) for interconnected data representation.



RDF triple

- RDF (Resource Description Framework) triples are the basic building blocks of data in the Semantic Web.
- Each RDF triple consists of three parts:
  - Subject: This represents the resource being described. In other words the subject is what the information is about.
  - Predicate: This represents the property or relationship between the subject and the object. Like the subject, it is represented by a URI within our system. The predicate defines the kind of information being expressed such as title of, chapter of, cited by, etc.
  - Object: This represents the value of the property or relationship. It can be a URI or a literal value (like a string or a number). The object is what the subject's property is related to.

RDF Triple Example



RDF Triple Schema

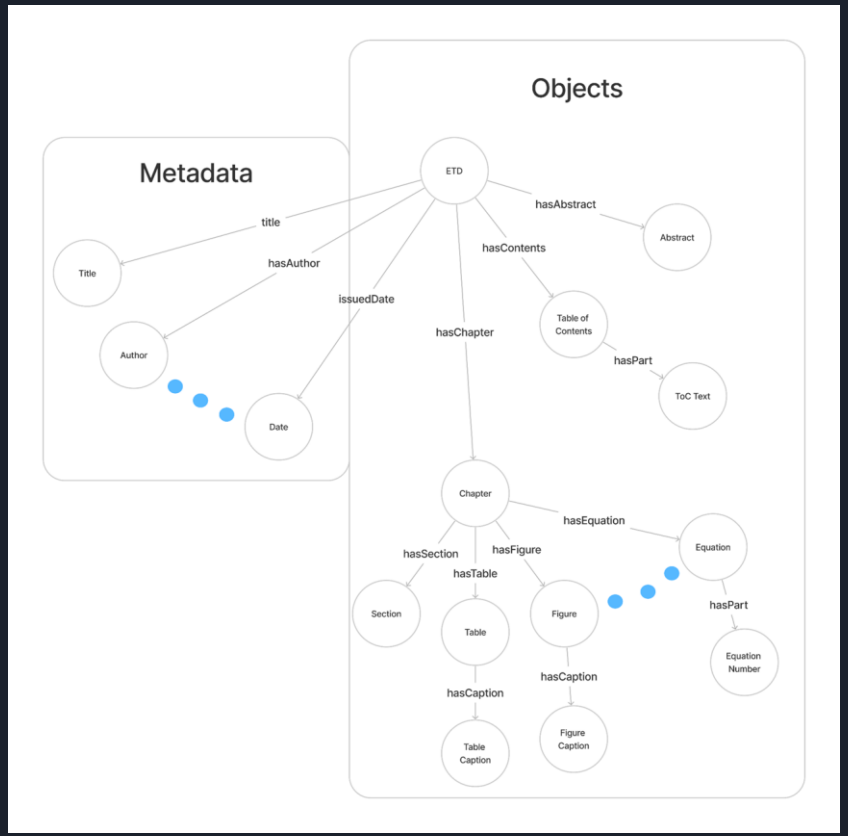
- Total of 49 types of unique RDF triples
  - Number of RDF triples per ETD roughly 600 unique entries
- Schema can essentially be broken down into 3 types:
  - Firstly, the ETD storing metadata (simplified):

Subject	**Predicate**	**Object**
ETD	hasTitle	Title
ETD	hasAuthor	Author
ETD	publishedOn	Date
ETD	publishedBy	University
ETD	committeeMember	Committee Member
ETD	degreeType	Degree
ETD	academicDiscipline	Department

RDF Triple Schema (continued)

- Secondly, the ETD storing chapters, abstracts, and table of contents
  - These values are stored as related to the ETD and contain other elements as well
- Thirdly, the chapters, abstracts, and table of contents storing elements that are located within them.
  - For the abstract and table of contents this is mostly just storing the text of the abstract and table of contents.
  - For the chapter this includes storing:
    - Sections
    - Paragraphs
    - Figures
    - Equations
    - Tables
    - Footnotes
    - etc.

Schema Diagram



ETD to RDF Triples Script

- Currently Python script that parses XML file representing the ETD
  - Extracts the necessary triples from the file
- Used on 200 ETDs provided to us from the object detection of Team 3
- Stores objects in the triple as URIs; example of a chapter being stored:

http://etdkb.endeavour.cs.vt.edu/v1/objects/6384265	http://etdkb.endeavour.cs.vt.edu/v1/predicate/chapterOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228

- The first box is the URI of the ETD
- The second box is the URI of the “hasChapter” predicate
- The third box is the URI of the Chapter itself

Top-down and Bottom-up Triples

- The queries we perform can filter for Subject and Predicate but not for the Object.
- Example: We want to find the chapters of a specific ETD.
  - We could query using the ETD ID as the subject and “hasChapter” as the predicate.
  - This would return all triples matching this convention, meaning we could see all the URIs of the chapters in this ETD.
- Now say we have a chapter and want to find the entire ETD it belongs to.
  - We cannot query based on Object value.
  - So we must create an inverse Triple with a “bottom-up” format.

Subject	Predicate	Object
ETD ID	hasChapter	Chapter ID
Chapter ID	chapterOf	ETD ID

Top-down and Bottom-up Triples

- This means for every relationship we must store two sets of triples:

Analysis of Generated Triples

- Total of 95k triples generated for ~160 ETDs
- These are the counts for each type of predicate that can occur in an ETD.

Term	Count
abstractOf	145
algorithmOf	143
authorOf	158
captionOf	6777
chapterOf	1403
committeeOf	138
degreeType	156
equationOf	5891
figureOf	4895
hasAbstract	145
hasAuthor	158
hasCaption	6777
hasChapter	1403
hasCommittee	138
hasContents	475
hasEquation	5891
hasAlgorithm	143
hasFigure	4895
hasPart	42609
hasPublished	133
hasTable	1882
issuedDate	146
partOf	8534
publishedBy	133
hasTitle	156
titleOf	156
tableOf	1882

Virtuoso - Graph Database

- Virtuoso is a high-performance, scalable, and extensible triple store and graph database. It's designed to handle large-scale RDF data and is widely used in the context of the Semantic Web and Linked Data.
- For our purposes we will use it for RDF Triple Store: Virtuoso is primarily known for its ability to store and query RDF triples, making it a crucial tool for managing Semantic Web data.

Store Triples in Virtuoso

- Virtuoso deployed on Endeavour container
- Successfully uploaded RDF triples into Virtuoso
- Allows for uploaded triples to be queryable via SPARQL
  - We can query by subject, predicate, or object or any combination of them using *filters* in SPARQL.
- Successfully uploaded RDF triples of 200 ETDs into Virtuoso

SPARQL | HTML5 table

http://etdkb.endeavour.cs.vt.edu/v1/objects/6388059	http://etdkb.endeavour.cs.vt.edu/v1/predicate/chapterOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6387955
http://etdkb.endeavour.cs.vt.edu/v1/objects/6388085	http://etdkb.endeavour.cs.vt.edu/v1/predicate/chapterOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6387955
http://etdkb.endeavour.cs.vt.edu/v1/objects/6388235	http://etdkb.endeavour.cs.vt.edu/v1/predicate/chapterOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6387955
http://etdkb.endeavour.cs.vt.edu/v1/objects/6388365	http://etdkb.endeavour.cs.vt.edu/v1/predicate/chapterOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6387955
http://etdkb.endeavour.cs.vt.edu/v1/etd/6384228/meta/committee	http://etdkb.endeavour.cs.vt.edu/v1/predicate/committeeOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228
http://etdkb.endeavour.cs.vt.edu/v1/etd/6385235/meta/committee	http://etdkb.endeavour.cs.vt.edu/v1/predicate/committeeOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6385235
http://etdkb.endeavour.cs.vt.edu/v1/etd/6387955/meta/committee	http://etdkb.endeavour.cs.vt.edu/v1/predicate/committeeOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6387955
http://etdkb.endeavour.cs.vt.edu/v1/etd/6404135/meta/committee	http://etdkb.endeavour.cs.vt.edu/v1/predicate/committeeOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6404135
http://etdkb.endeavour.cs.vt.edu/v1/etd/6405958/meta/committee	http://etdkb.endeavour.cs.vt.edu/v1/predicate/committeeOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6405958
http://etdkb.endeavour.cs.vt.edu/v1/etd/6450634/meta/committee	http://etdkb.endeavour.cs.vt.edu/v1/predicate/committeeOf	http://etdkb.endeavour.cs.vt.edu/v1/objects/6450634

SPARQL Examples

```
SELECT ?subject ?predicate ?object
WHERE {
  ?subject ?predicate ?object
FILTER regex(str(?subject), "3039fa8f-d5f9-4a0f-953e-bbd7f20a3254", "i")
}
```

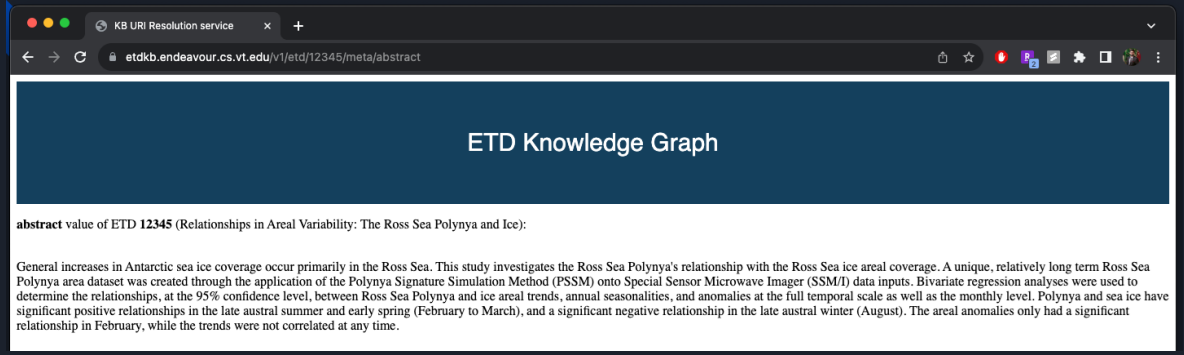
Fetch all relationships for the mentioned ETD ID

```
SELECT ?subject ?predicate ?object
WHERE {
  ?subject ?predicate ?object
FILTER regex(str(?predicate), "title", "i")
}
```

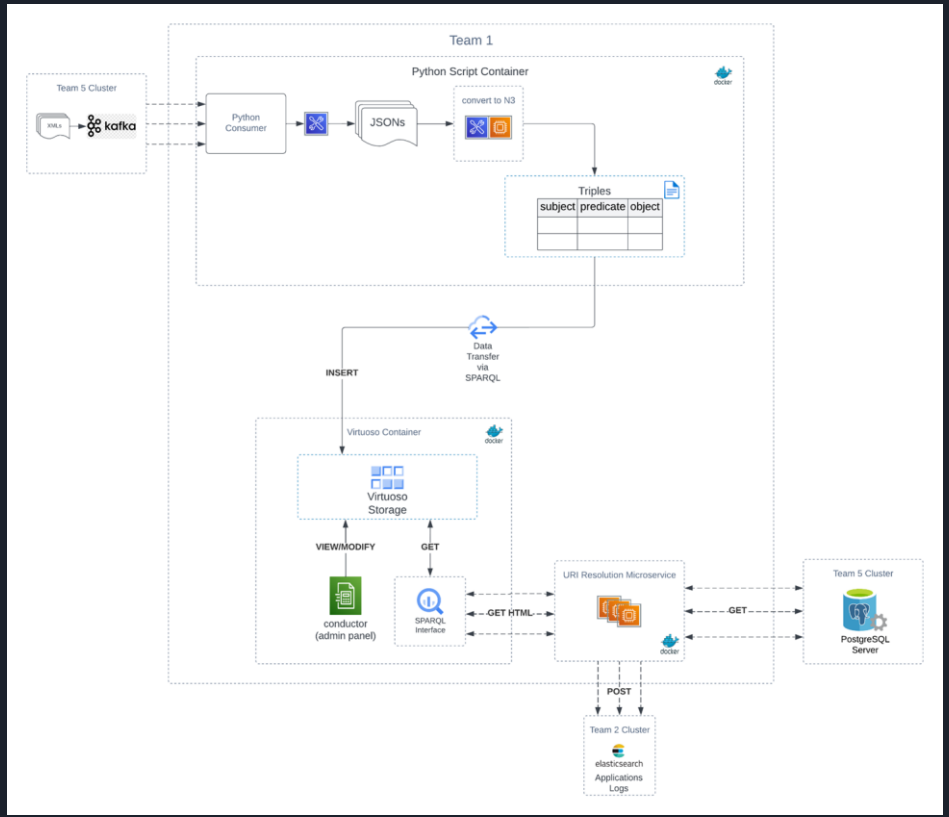
Fetch all ETDs that have “title” predicate

URI Resolution Service

- This web service resolves URIs which are returned as part of the SPARQL query. The user can click on the URI and they will be redirected to a webpage that contains supplemental information.
- This is a RESTful service with 3 main routes:
  - v1/etd/:etdId/meta/:metaProp
    - This route fetches data from the “metadata” table of PostgreSQL.
  - v1/objects/:objectId
    - This route fetches data from the “objects” table of PostgreSQL.
  - v1/predicate/:predicateName
    - This route fetches data from a JSON stored in server.
- The domain of the service is - https://etdkb.endeavour.cs.vt.edu/



System Architecture

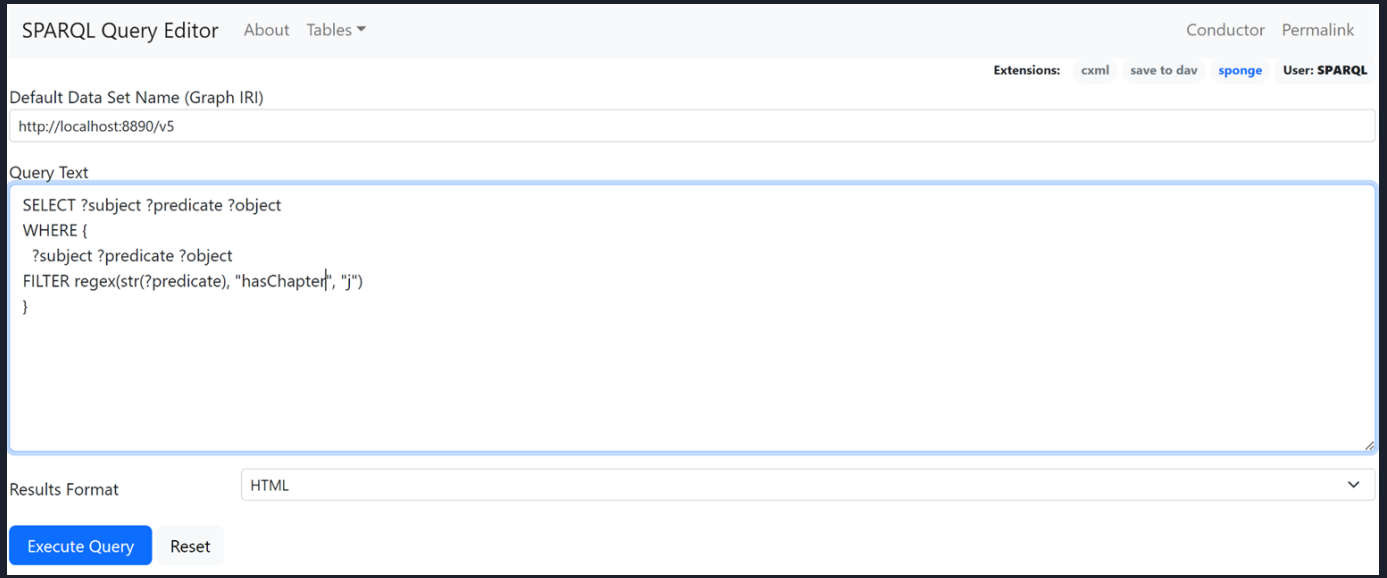


Future Work

- Object detection was only completed for a set of 200 ETDs; future researchers shall work on performing OD and generating XML files for all the 500K ETDs.
    - To also consider the page images being generated for each ETD by Team3 and later their XML files.
- Expand the triple extraction and ingestion process to Virtuoso DB for the remaining ETD XML files in the collection (500K).
- Enhance the URI resolution service UI.
- Build a custom UI integrated with Search interface to query the ETD RDF graph without the end-user needing to enter SPARQL queries in Virtuoso interface.
- Extract entities from the textual objects of ETDs using NER methods to build a finer grained KG.
- The Kafka implementation could not be completed due to delays within other teams. We manually utilised a set of dummy 200 ETD XMLs provided to us.
    - Automate the pipeline for triple extraction and ingestion into Virtuoso DB.

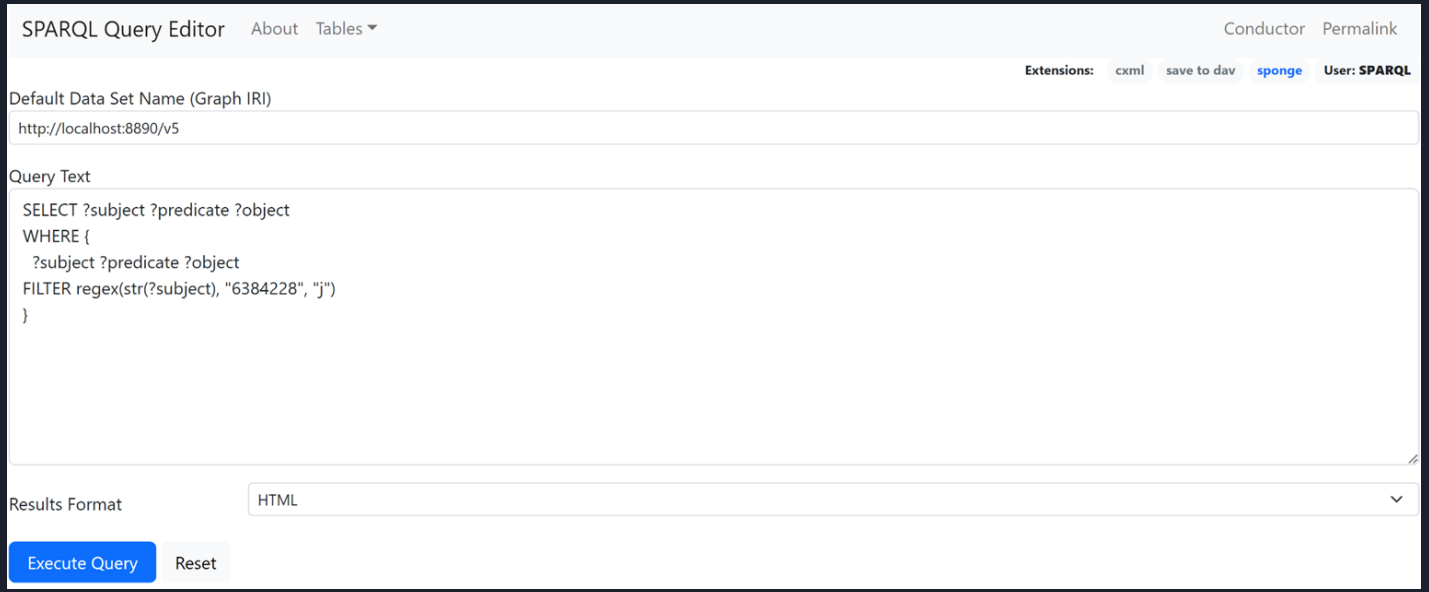
Demo Step 1

Example SPARQL query that queries by predicate



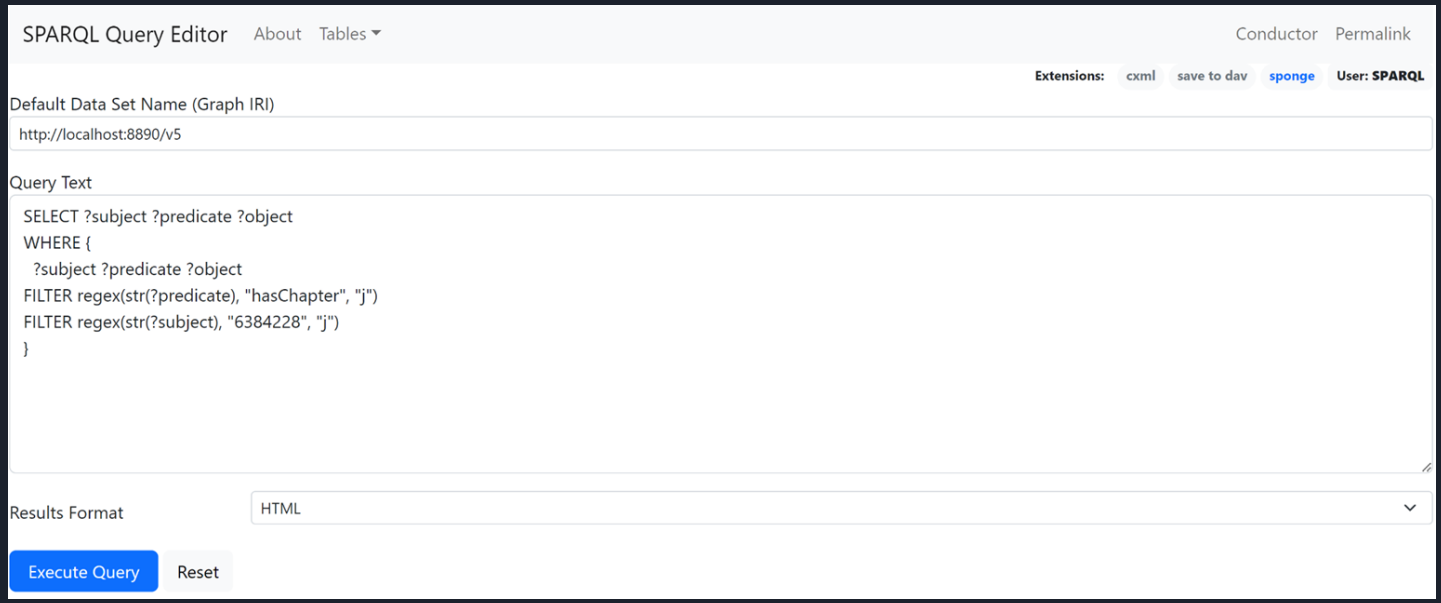
Demo Step 2

Example SPARQL query that queries by subject



Demo Step 3

Example SPARQL query that queries by both predicate and subject



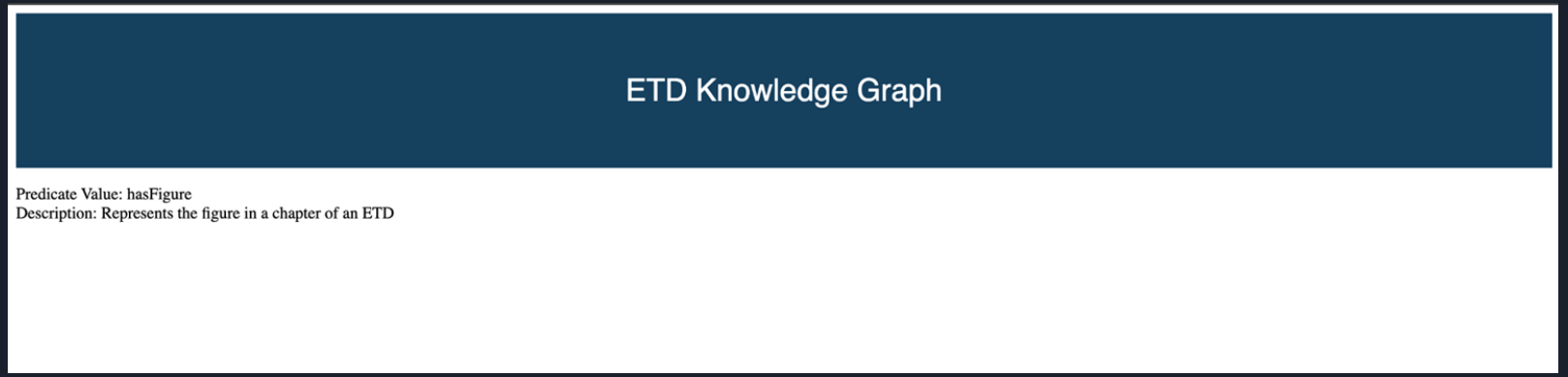
Demo Step 4

Results of the query by both subject and predicate

subject	predicate	object
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384265
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384267
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384312
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384340
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384411
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384456
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384552
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384562
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384624
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384631
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384633
http://etdkb.endeavour.cs.vt.edu/v1/objects/6384228	http://etdkb.endeavour.cs.vt.edu/v1/predicate/hasChapter	http://etdkb.endeavour.cs.vt.edu/v1/objects/6384666

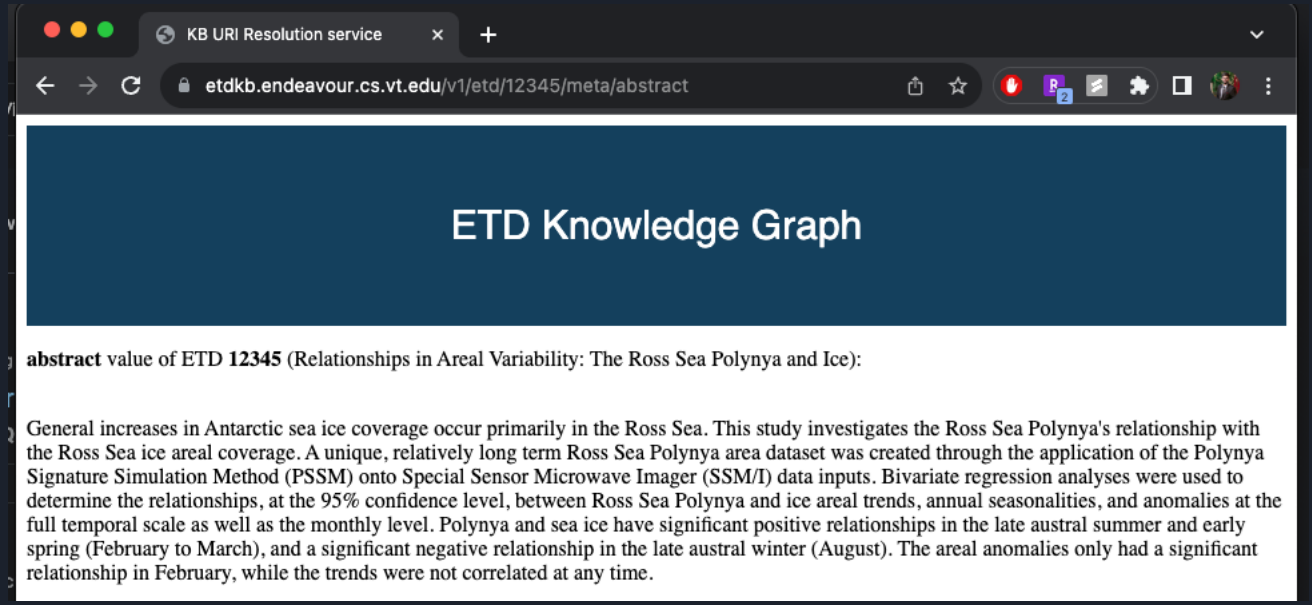
Demo Step 5

URI resolution of an example predicate: “hasFigure”



Demo Step 6

URI resolution of an example abstract



Thank you!