

2024



CSE488: ontologies and the semantic web

AHMED ASHOUR 20P4222

Habiba yasser abdelhalim 20P3072

nadine hisham 20P9880

Sohayla Ihab Abdelmawgoud Ahmed Hamed 19P7343

Milestone I: A Design and Implementation for Movies Ontology

# Part I: Modeling the Ontology

Protégé Editor was used to create this ontology of movies. Basically, we aim to create a blueprint for the online referencing and storage of movie datasets. First, the following classes and properties are required to be set:

1. **Class:**
2. Movie: defines basic Movie class
3. Genre: defines basic Genre class
4. Person: defines basic Person class
5. Actor: basic Actor class is a subclass of Person class
6. Writer: basic Writer class is a subclass of Person
7. Director: basic Director class is a subclass of Person
8. **Object Property:**
9. hasGenre: Each Movie has one or more instances of class Genre
10. hasActor: Each Movie has at least one instance of class Actor
11. hasWriter: Each Movie has at least one instance of class Writer
12. hasDirector: Each Movie has at least one instance of class Director
13. isGenreOf: Inverse of hasGenre
14. isActorOf: Inverse of hasActor
15. isDirectorOf: Inverse of hasDirector
16. isWriterOf: Inverse of hasWriter
17. **Data Property:**
18. name: Name of Person
19. age: Age of Person
20. nationality: Nationality(s) of Person
21. gender: Gender of Person
22. genre: Genre(s) of Movie
23. title: Title of Movie
24. country: Country(s) of Production of Movie
25. language: Language(s) of Movie
26. year: Year of Copyright of Movie

In order to create restrictions and proper relationships between classes and properties, the following assumptions have been observed:

1. Datasets all have films dating from the early 1900s till the current documentation
2. An individual can be an actor or a director or a writer. They can also play multiple roles (ie: an actor can be a director or a writer as well)
3. A Movie cannot have the same values of the Genre as Director, Writer, Actor
4. A Genre cannot have name of Person (Actor, Writer, Director) or Movie
5. hasGenre cannot be equivalent to hasActor, hasDirector, hasWriter between the same individuals.
6. Disjointedness between data properties is always between properties belonging to the same class.
7. Language and Country of a Movie are not equivalent values (ie: A Movie whose country is Canada can have the language France (in English tags only), but France should not be filtered as a country among others in a query searching for

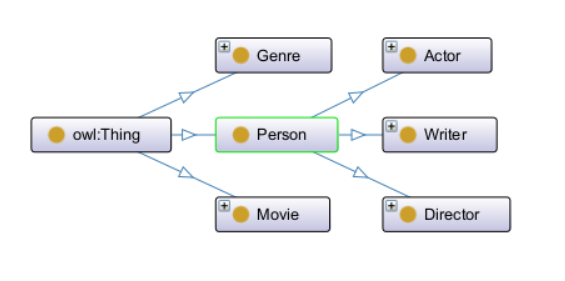
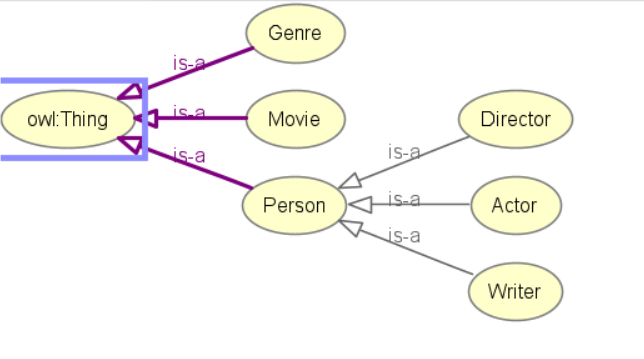
country == “France”).

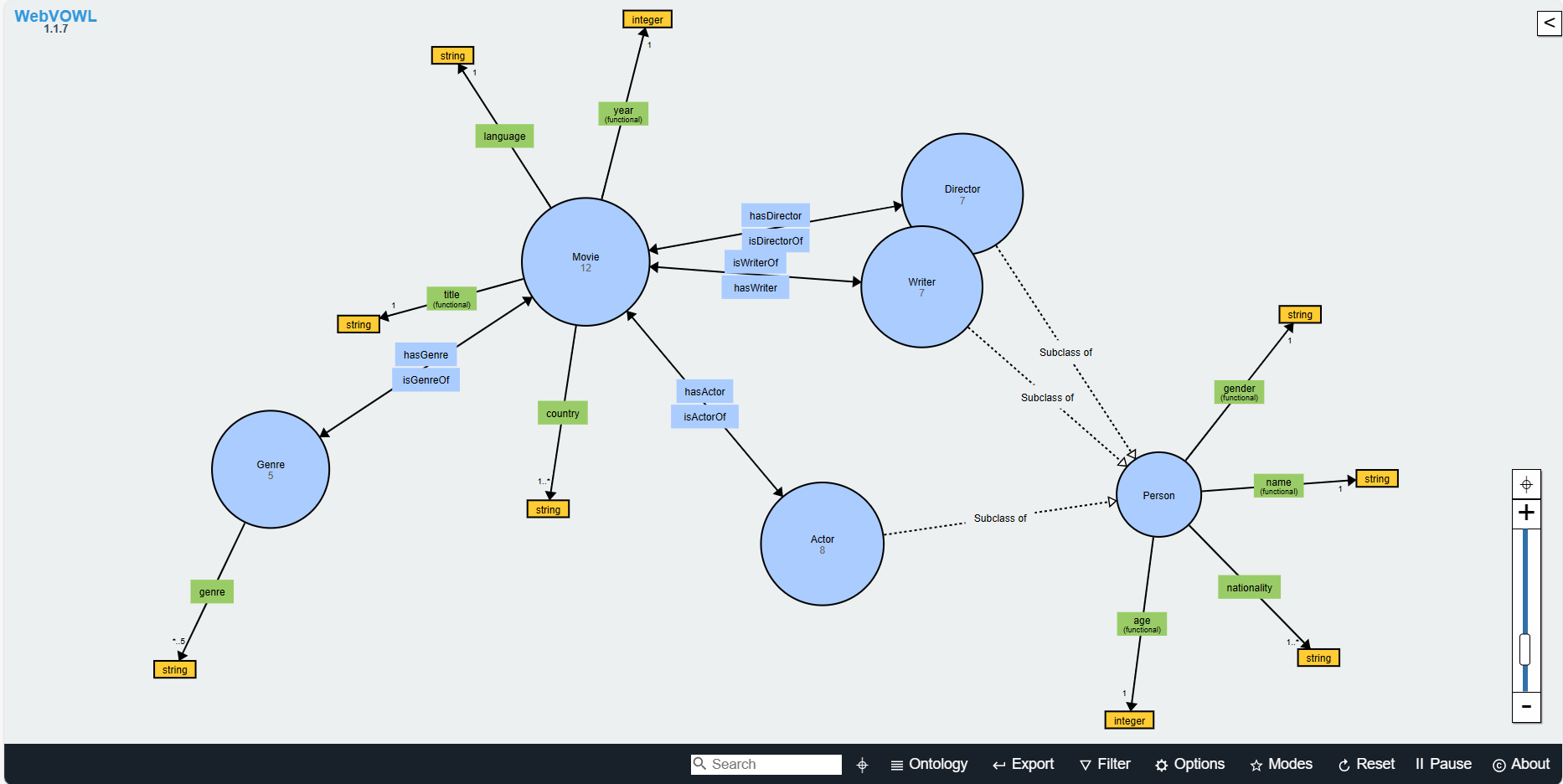
1. Only one name and one title for a Person or a Movie should be registered.
2. There are only 2 genders, male or female.
3. A Movie in a different language is considered a different movie (ie: Sleeping Beauty exists in languages: English, Arabic and

Al-Jameela Al-Na’ema also exists in the languages: Arabic, English).

1. Only 1 type of language tags were used, the default English tags. There weren’t explicitly declared.
2. Age 0 means the Person is deceased.
3. A Person (Actor, Director, Writer) can have multiple nationalities.

Finally, view the ontology:

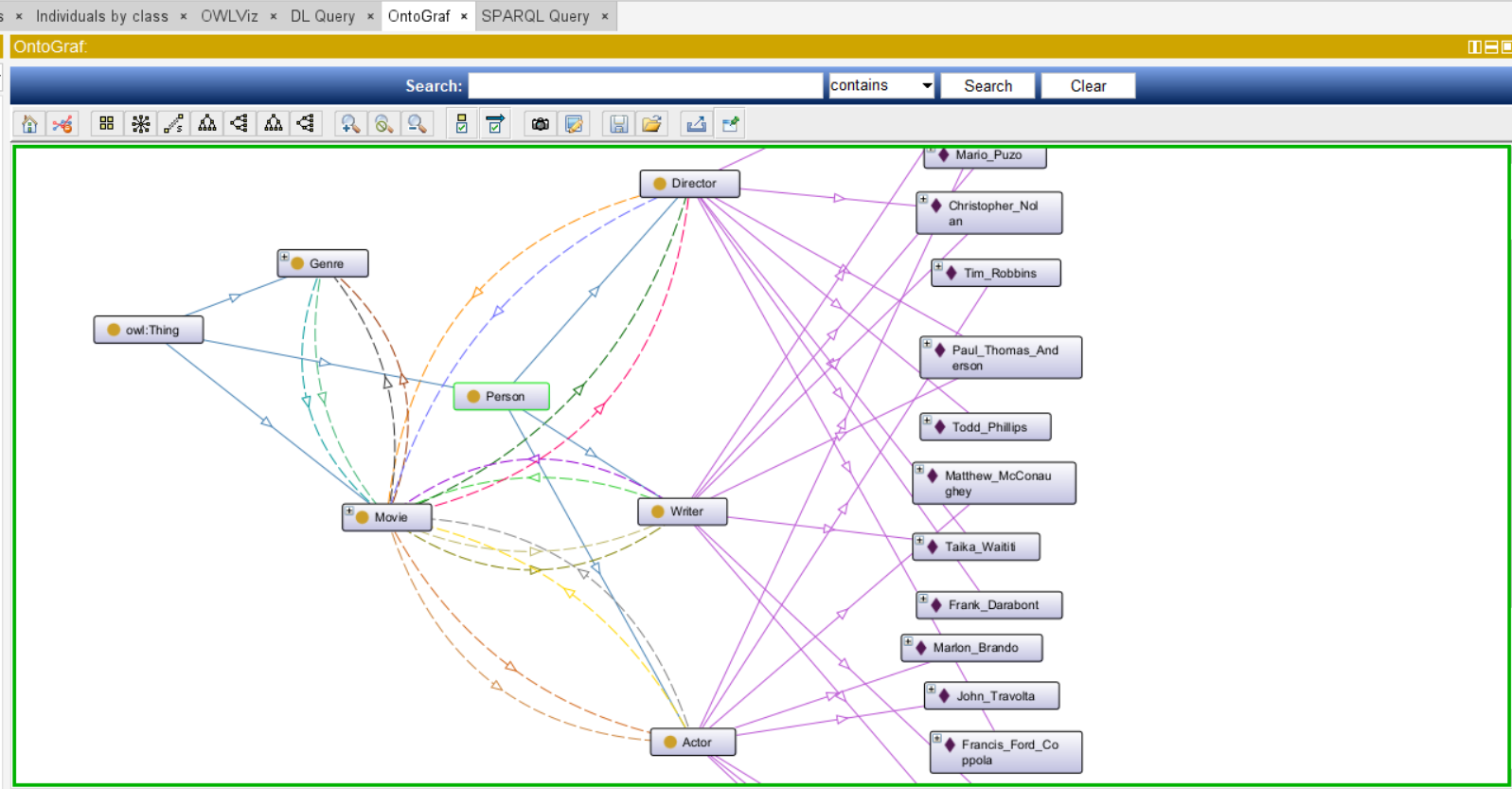




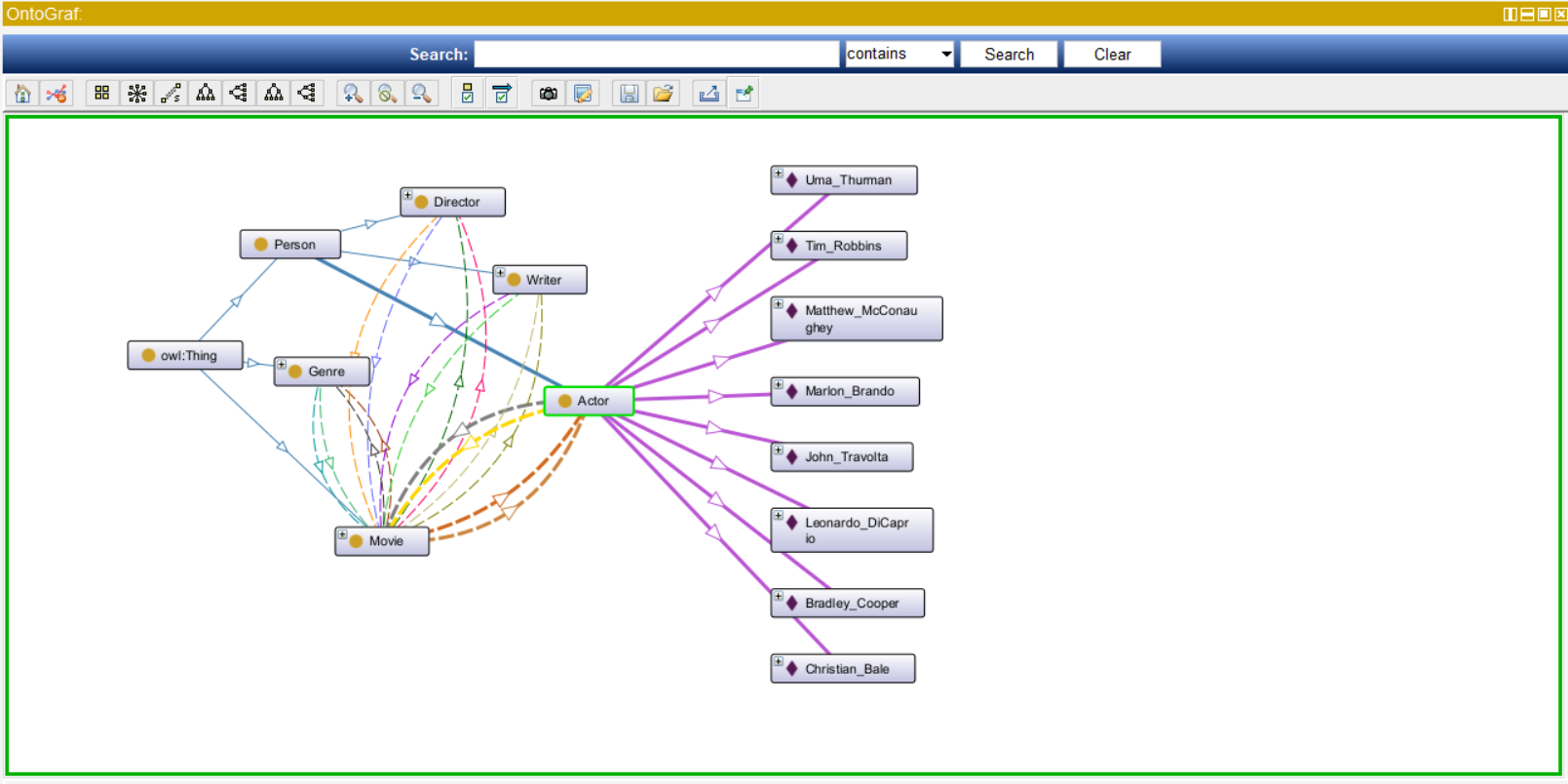
# Part II: Populating the Ontology

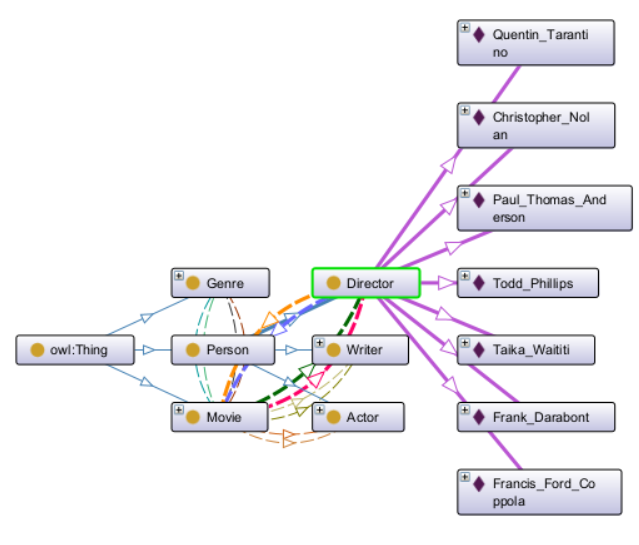
Now, having designed the ontology, it is time to populate it. Here are some individuals:

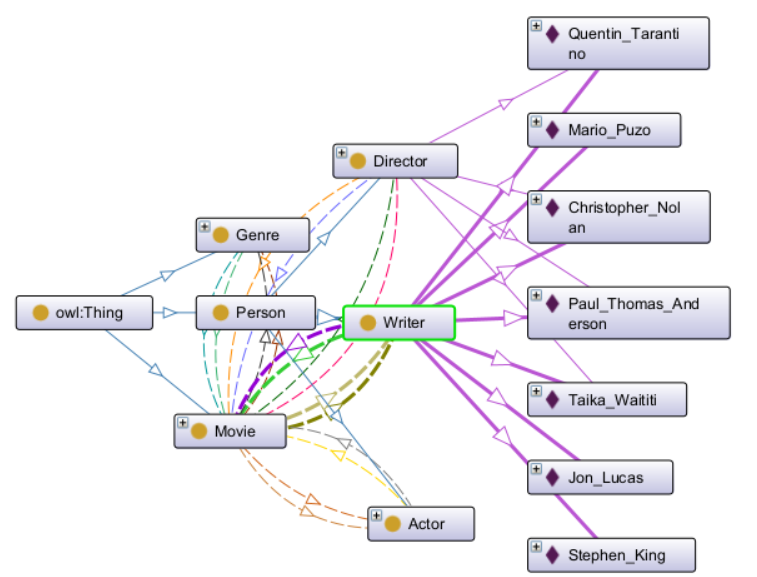
1. **Person (Actor, Director, Writer)**



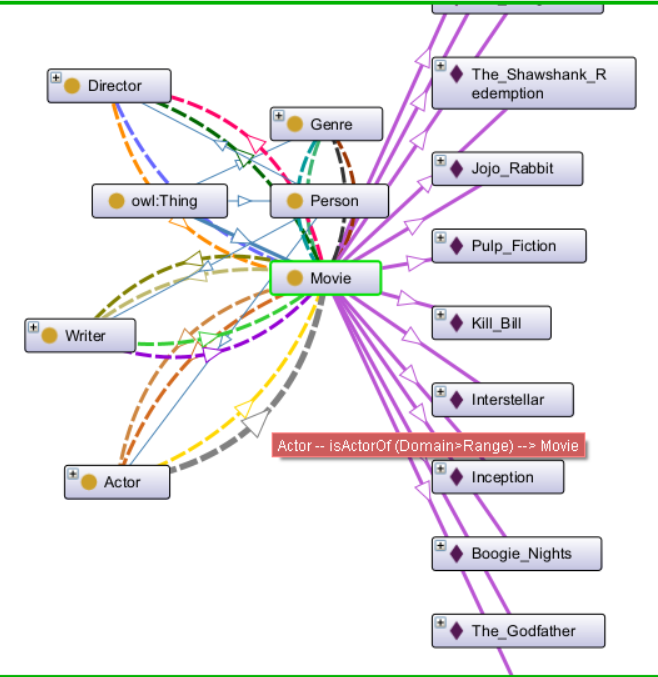
1. **Actor**



1. **Director**
2. **Writer**

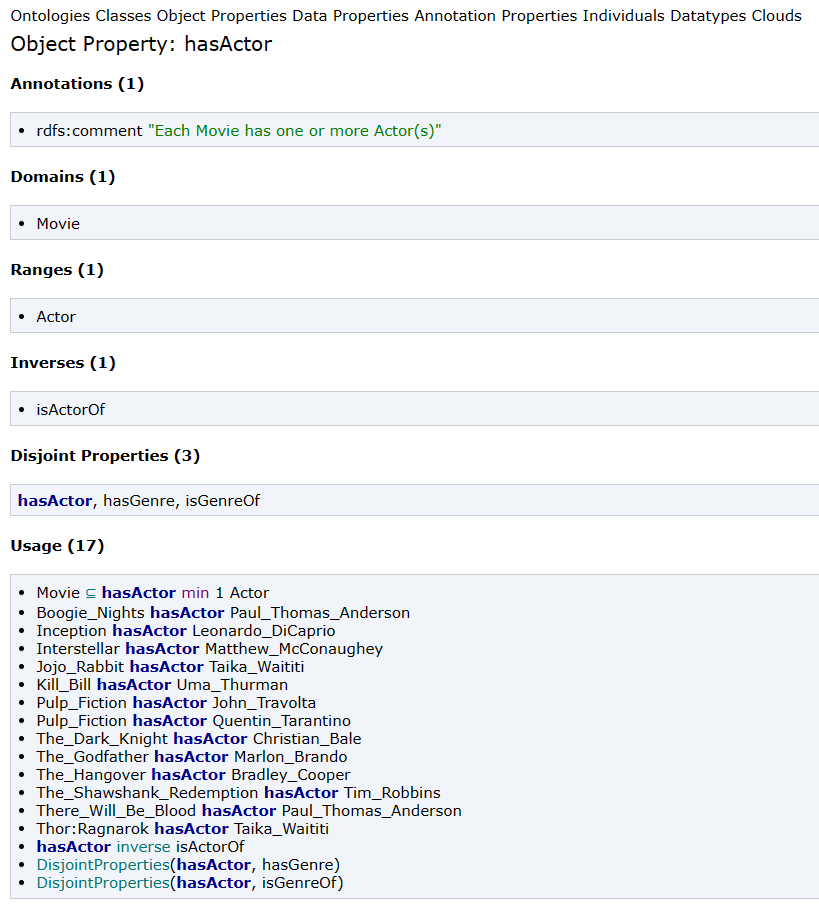


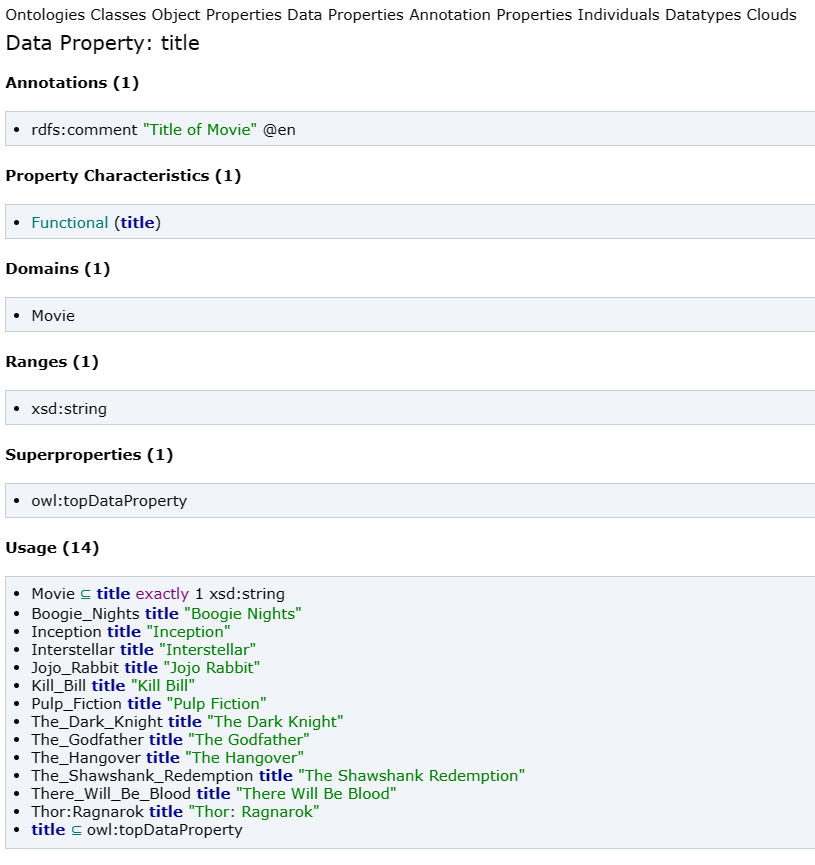
1. **Movie**



Finally, test the consistency of the ontology with PELLET reasoner (Turn on Pellet and press CTRL+R)

Below, the restrictions of some object properties and datatype properties are summarized:





# Part III: SPARQL Queries on the Ontology

Start querying your ontology with sparql, use different **types** and nests of queries**.** Each **type of query** is listed with its output:

1. **Query 1: A query that contains at least 2 Optional Graph Patterns and uses a FILTER with regular expressions**

Query 1: Extracts the names, ages, and nationalities of actors whose names start with letters A-M.

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

SELECT DISTINCT **?name** **?age** **?nationality**

WHERE {

?actor rdf:type ont:Actor.

?actor ont:name **?name**.

OPTIONAL {

?actor ont:age **?age**

}

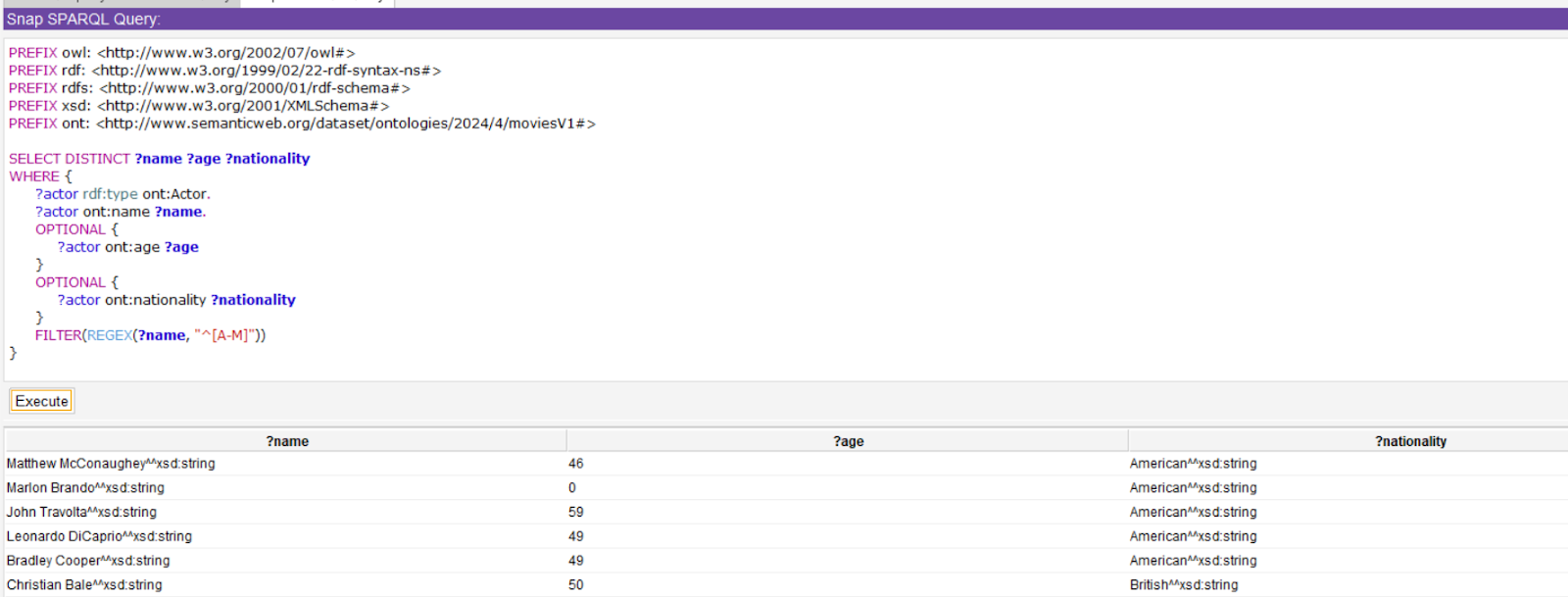
OPTIONAL {

?actor ont:nationality **?nationality**

}

FILTER(REGEX(**?name**, "^[A-M]"))

}



1. **Query 2: A query that contains at least 2 alternatives and conjunctions and uses aggregate functions (COUNT)**

Query 2: Retrieves titles, years, genre names, actor names, and director names for movies released before 2010 and categorized as Action or Thriller, along with the count of genres for each movie.

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

SELECT **?title** **?year** **?genre\_name** **?actor\_name** **?director\_name** (COUNT(?genre) AS **?genre\_count**)

WHERE {

?movie rdf:type ont:Movie.

?movie ont:title **?title**.

?movie ont:year **?year**.

?movie ont:hasGenre ?genre.

?genre ont:genre **?genre\_name**.

{

?movie ont:hasActor **?actor\_name**.

}

UNION

{

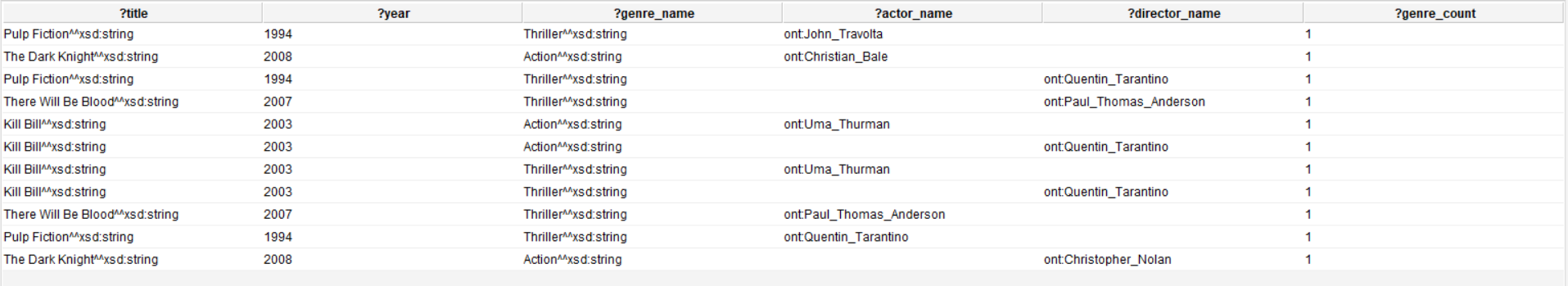
?movie ont:hasDirector **?director\_name**.

}

FILTER(**?year** < 2010 && (**?genre\_name** = "Action" || **?genre\_name** = "Thriller"))

}

GROUP BY **?title** **?year** **?genre\_name** **?actor\_name** **?director\_name**



1. **Query 3: A query that contains a CONSTRUCT query form with nested patterns**

Query 3: Constructs a new RDF graph containing individuals who are both actors and directors.

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

CONSTRUCT {

**?person** rdf:type ont:Actor;

rdf:type ont:Director .

}

WHERE {

{

**?person** rdf:type ont:Actor .

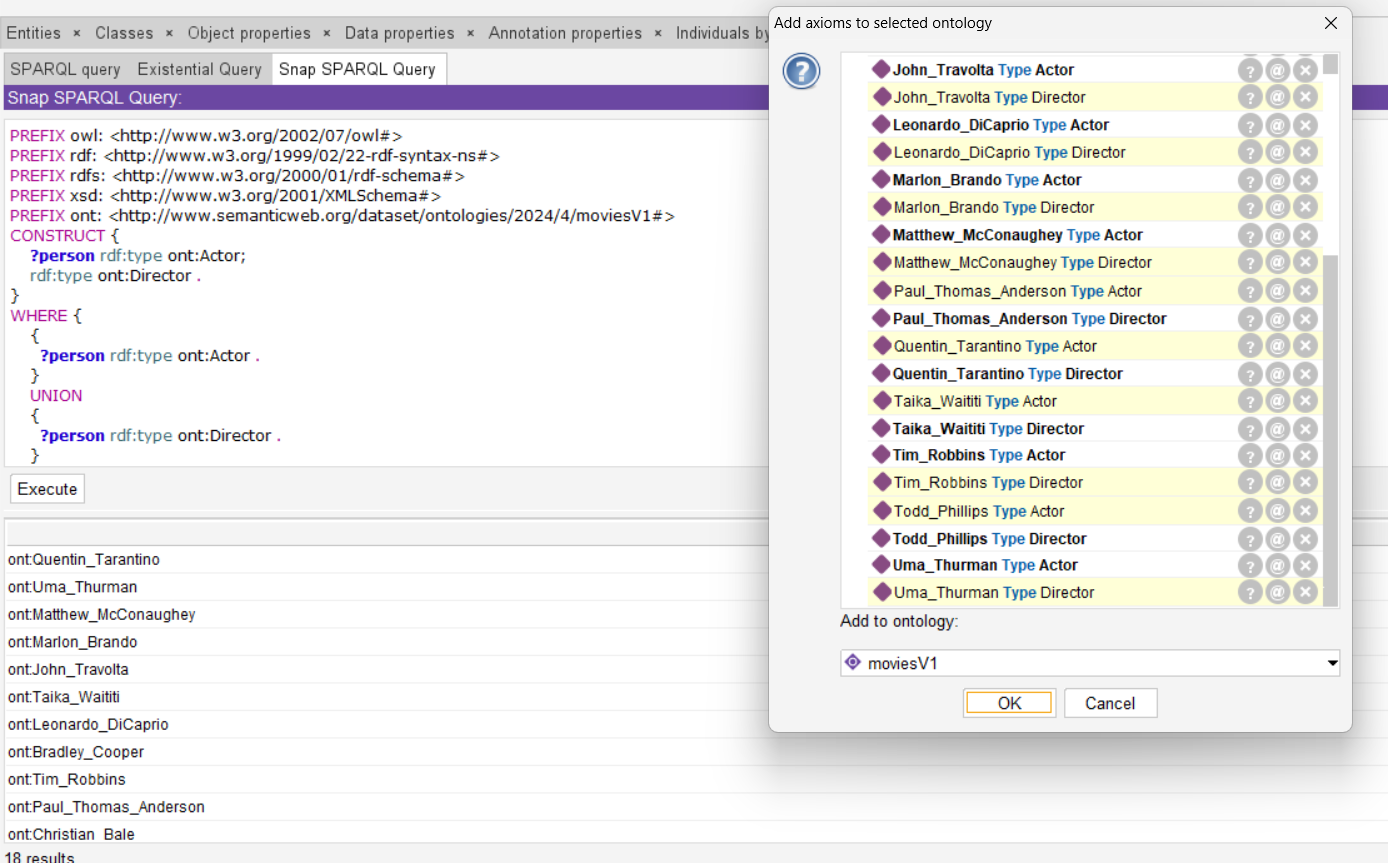
}

UNION

{

**?person** rdf:type ont:Director .

}

}

1. Query 4: Count movies with both "Comedy" and "Drama" genres, excluding those released before 2005.

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

SELECT (COUNT(?movie) AS **?count**)

WHERE {

?movie rdf:type ont:Movie ;

ont:hasGenre ?genre ;

ont:year ?year .

FILTER((?genre = ont:Comedy) || (?genre = ont:Drama) && ?year >= 2005)

}

A screenshot of a computer

Description automatically generated

1. **Query 5: A query that contains a FILTER with date comparison**

Query 5: Fetches titles and release dates of movies released after January 1, 2000.

Query 7: Fetches titles and release dates of movies released after January 1, 2000.

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

SELECT **?title** **?release\_date**

WHERE {

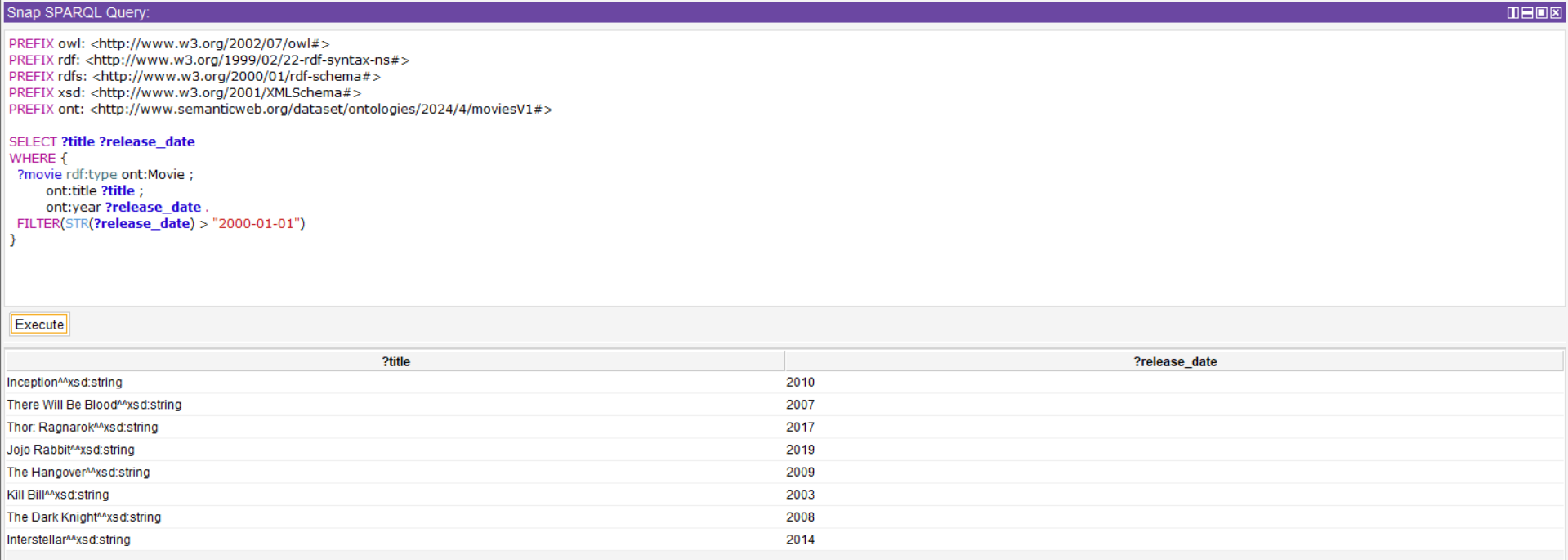
?movie rdf:type ont:Movie ;

ont:title **?title** ;

ont:year **?release\_date** .

FILTER(STR(**?release\_date**) > "2000-01-01")

}



1. **Query 6: Retrieve all movies written by writers who are also actors.**

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

SELECT **?movie** **?writer\_actor**

WHERE {

**?movie** rdf:type ont:Movie ;

ont:hasWriter ?writer ;

ont:hasActor **?writer\_actor** .

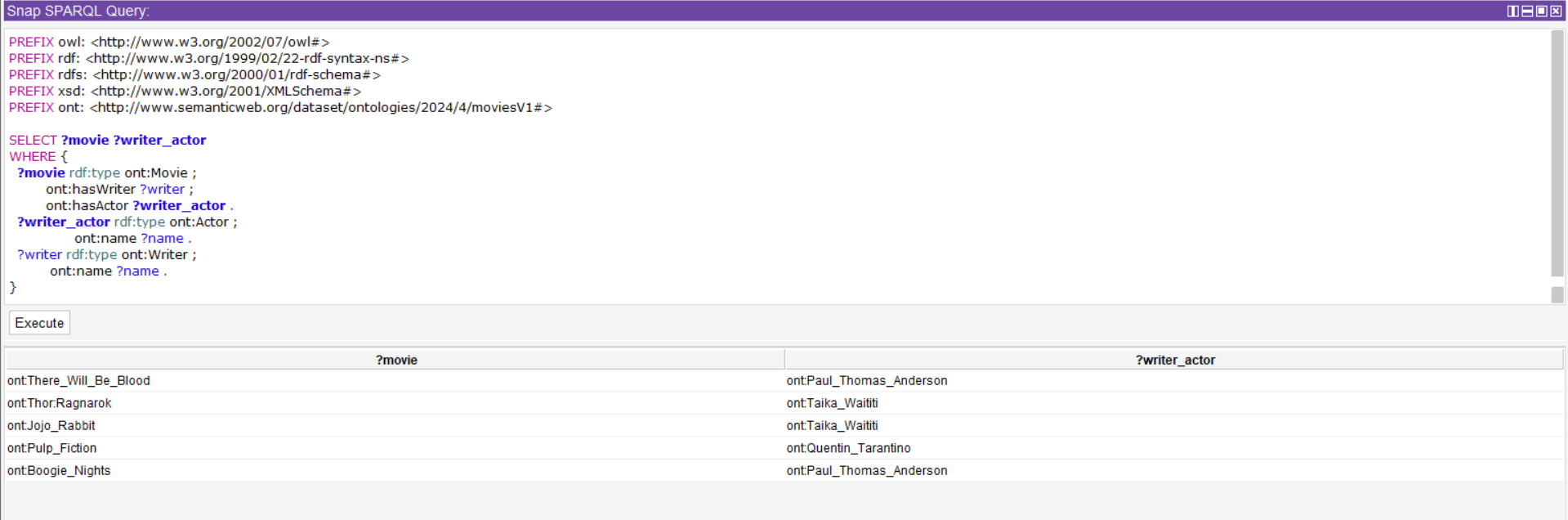
**?writer\_actor** rdf:type ont:Actor ;

ont:name ?name .

?writer rdf:type ont:Writer ;

ont:name ?name .

}



1. **Query 7: Retrieve all movies released in the 21st century along with their titles and genres.**

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

SELECT **?movie** **?title** **?genre**

WHERE {

**?movie** rdf:type ont:Movie ;

ont:title **?title** ;

ont:hasGenre **?genre** ;

ont:year ?year .

FILTER(?year >= 2000 && ?year < 2100)

}

A screenshot of a computer

Description automatically generated

1. **Query 8: List all actors who have appeared in movies directed by themselves, along with their names.**

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

SELECT DISTINCT **?actor ?name**

WHERE {

**?actor** rdf:type ont:Actor ;

         ont:name **?name** ;

         ont:isActorOf **?movie** .

**?movie** ont:hasDirector **?actor** .

}

A screenshot of a computer

Description automatically generated

1. **Query 9: A query that contains an ASK query form with negation (MINUS)**

Query 9: Checks if there are movies without the genre "Comedy."

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

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

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

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX ont: <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#>

ASK

WHERE {

**?movie** rdf:type ont:Movie .

  MINUS {

**?movie** ont:hasGenre **?genre** .

    FILTER(**?genre** = "Comedy")

  }

}

A screenshot of a computer code

Description automatically generated

1. **Query 10: *A query that contains a DESCRIBE query form with nested properties***

Query 10: Describes information about the Actor class in the ontology.

DESCRIBE <http://www.semanticweb.org/dataset/ontologies/2024/4/moviesV1#Actor>

A screenshot of a computer code

Description automatically generated