Práctica 4: Creación de una ontología de libros con Protégé

---

1. Estructura de la ontología

1.1. Jerarquías

Las clases representan conceptos principales en el dominio, como libros, géneros y ediciones. Algunas de ellas tienen subclases que especifican categorías más concretas.

Clases principales

- **Libro**. Clase base para representar libros. Tiene subclases específicas:
  - ***Ebook***: representa libros electrónicos (disjunta de *LIbroFísico*).
  - ***LibroFísico***: representa libros físicos (disjunta de *Ebook*).
  - ***LibroDeLujo***: con precios superiores a 50.
  - ***LibroAnónimo***: libros que no tienen un autor asociado.
  - ***LibroCienciaFicciónBarato***: de ciencia ficción con precios menores a 20.
  - ***LibroColaboración***: escritos por al menos dos autores.
  - ***LibroEdiciónLimitada***: libros con edición limitada.
  - ***LibroPopular***: con precios menores a 20.
- **Género**. Clase que agrupa categorías de libros según su contenido:
  - Subclases: ***CienciaFicción, Fantasía, Ficción, NoFicción, Terror, Biografía*** (siendo *Ficción* y *NoFicción* disjuntas).
- **Edición**. Clase que agrupa los tipos de ediciones, en este caso tres:
  - ***Estándar***, ***Limitada***, ***Primera****.*
- **Autor** y **Editorial**: representan los participantes en la creación y publicación de los libros, respectivamente.

1.1. Propiedades

Las propiedades conectan clases y permiten modelar relaciones y atributos de los individuos.

Propiedades de objeto

- ***escritoPor*** (dominio: Libro, rango: Autor): relaciona un libro con su(s) autor(es).
- ***haEscrito*** (inversa de `escritoPor`): relaciona un autor con el(los) libro(s) que ha escrito.
- ***publicadoPor*** (dominio: Libro, rango: Editorial): relaciona un libro con su editorial.
- ***haPublicado*** (inversa de `publicadoPor`): relaciona una editorial con los libros que ha publicado.
- ***tieneGénero*** (dominio: Libro, rango: Género): relaciona un libro con su género.
- ***tieneEdición*** (dominio: Libro, rango: Edición): relaciona un libro con su tipo de edición.

Propiedades de datos

- ***tieneAñoPublicación*** (dominio: Libro, rango: xsd:integer): representa el año de publicación del libro.
- ***tienePrecio*** (dominio: Libro, rango: xsd:decimal): representa el precio del libro

1.3. Restricciones

Las restricciones especifican condiciones para las clases mediante expresiones OWL. Se utilizan para definir clases equivalentes y asegurar la consistencia de la ontología.

Clases definidas por restricciones

- ***LibroAnónimo***:
  - Restricción: no tiene un autor asociado.
  - Expresión: `escritoPor cardinalidad máxima = 0`
- ***LibroCienciaFicciónBarato***:
  - Restricciones:
    - Género: Ciencia Ficción.
    - Precio: menor a 20.
  - Expresión:
    - `tieneGénero value géneroCienciaFicción`
    - `tienePrecio < 20`
- ***LibroColaboración***:
  - Restricción: tiene al menos dos autores.
  - Expresión: `escritoPor cardinalidad mínima = 2`
- ***LibroDeLujo***:
  - Restricción: precio mayor o igual a 50.
  - Expresión: `tienePrecio ≥ 50`
- ***LibroEdiciónLimitada***:
  - Restricción: el tipo de edición es *Limitada*.
  - Expresión: `tieneEdición value Limitada`.
- ***LibroPopular***:
  - Restricción: precio menor a 20.
  - Expresión: `tienePrecio < 20`

Propiedades con restricciones

- Como se ha visto anteriormente, las propiedades tienen dominios y rangos definidos para garantizar su correcto uso. Por ejemplo: ***escritoPor*** conecta libros con autores, y su rango es ***Autor***.

1.4. Individuos

Individuos específicos que representan entidades concretas:

- **Autores**: `Gabriel_García_Márquez`, `Rosa_Montero`.
- **Editoriales**: `Santillana`.
- **Libros**: `EjemploLibro`, `LibroCFBarato`, `LibroTerrorEdLimitada`…
- **Géneros**: `géneroCienciaFicción`, `géneroTerror`, `géneroFantasía`…
- **Ediciones**: `Estándar`, `Limitada`, `Primera`.

1.5. Representación gráfica de la ontología

Se pueden visualizar las jerarquías mediante las herramientas de ***OWLViz*** y ***OntoGraf***.

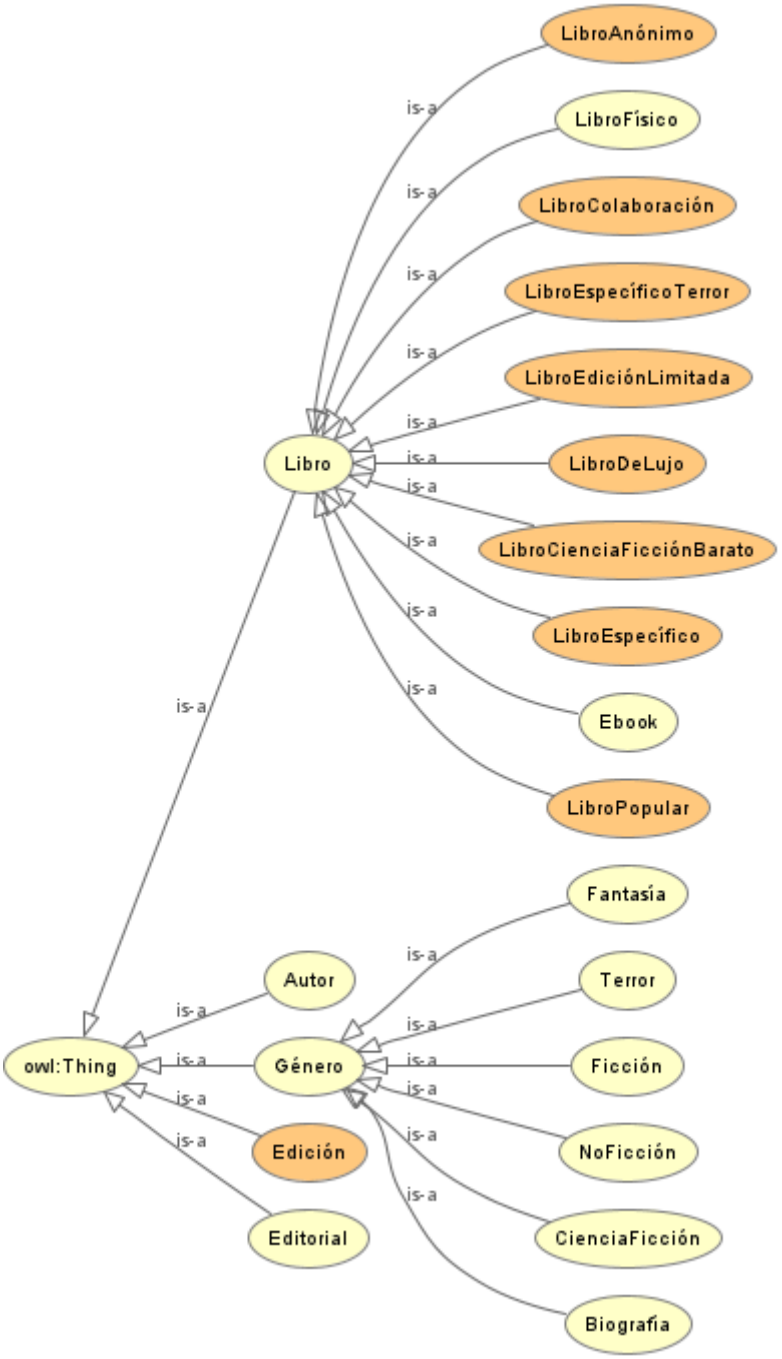


Figura 1: representación en OWLViz de la jerarquía *asserted*.

Figura 2: representación en OWLViz de la jerarquía *inferred*.

Se explicarán en mayor detalle las inferencias en el apartado siguiente.

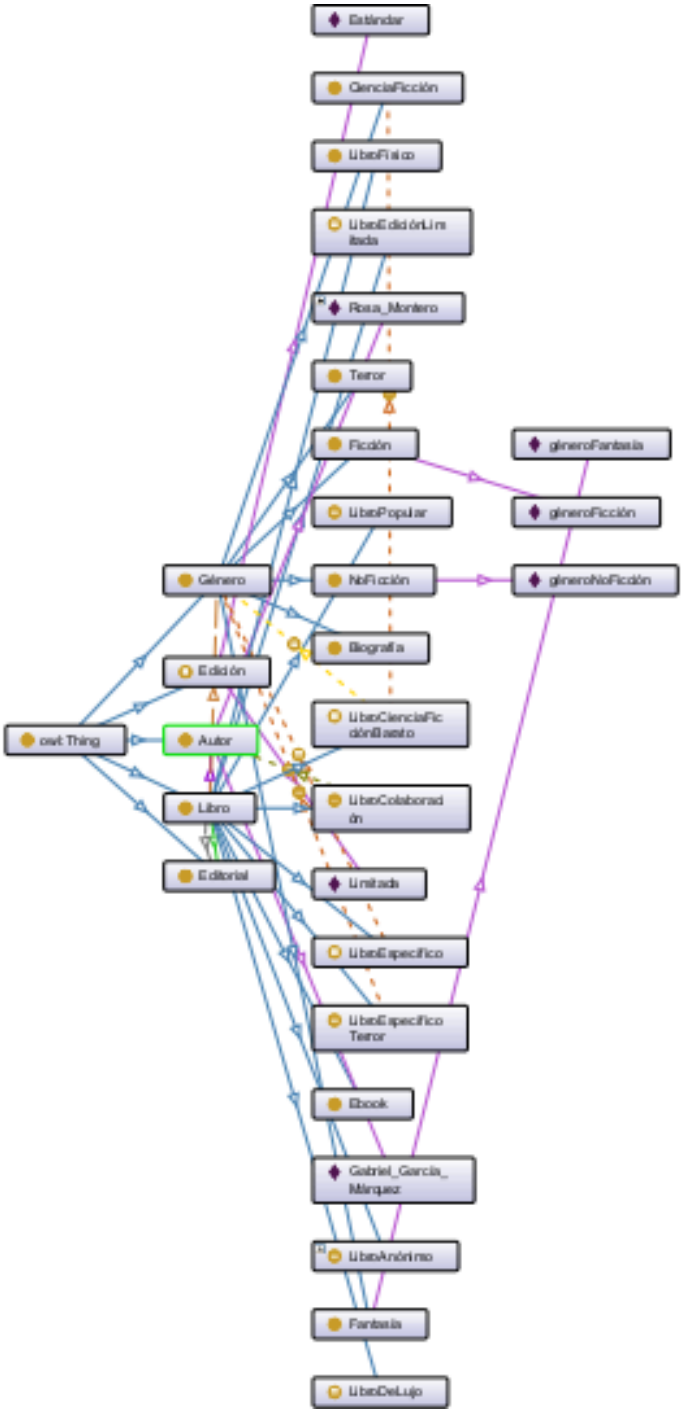
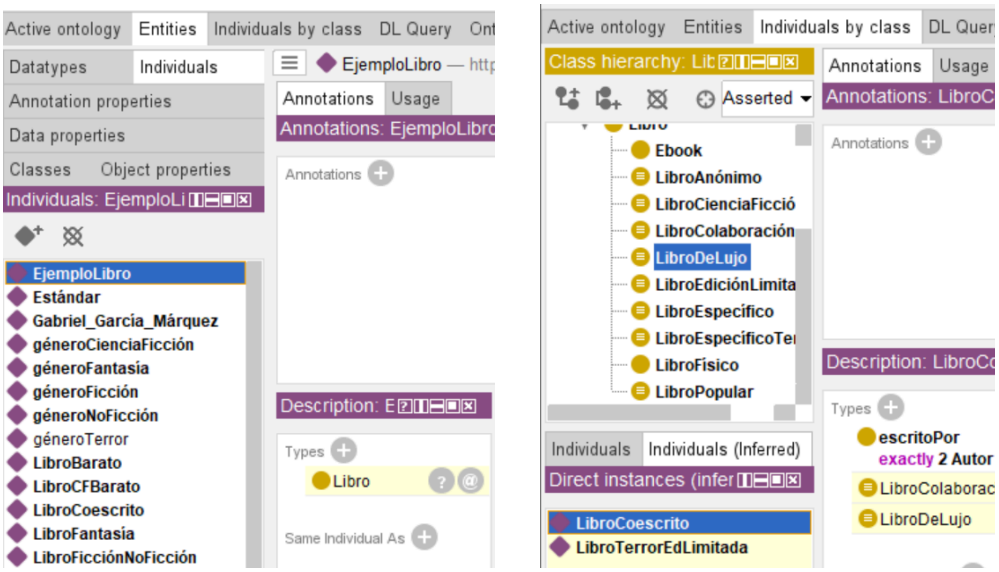


Figura 3: representación de las relaciones con OntoGraf.

2. Clasificaciones e inferencias del razonador

Las inferencias se pueden verificar de distintas formas: desde la ventana *Individuals by class*, desde *Entities* → *Individuals* y desde el panel de *Classification*.

Las inferencias destacan por estar resaltadas en **amarillo**, y se puede pulsar el icono de la interrogación para consultar la explicación de la inferencia.



Figuras 4 y 5: Formas de consultar los tipos inferidos para cada individuo.

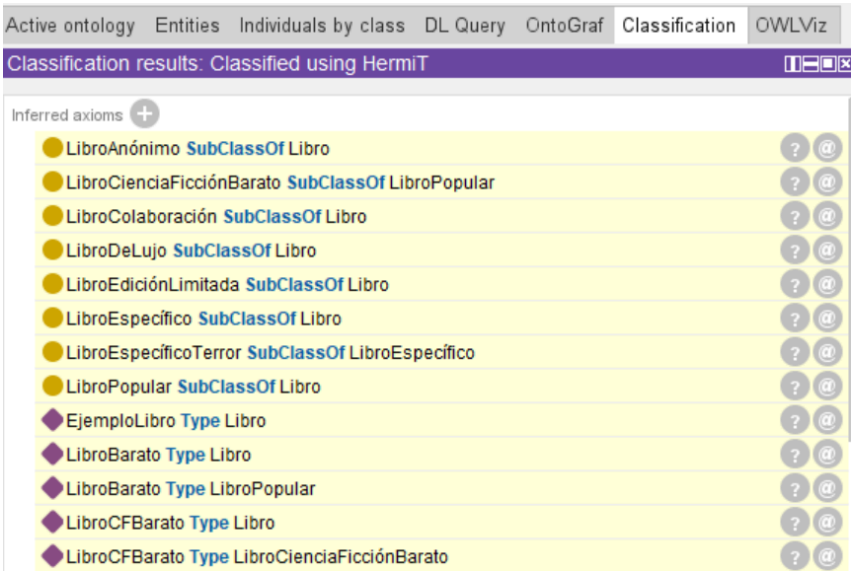


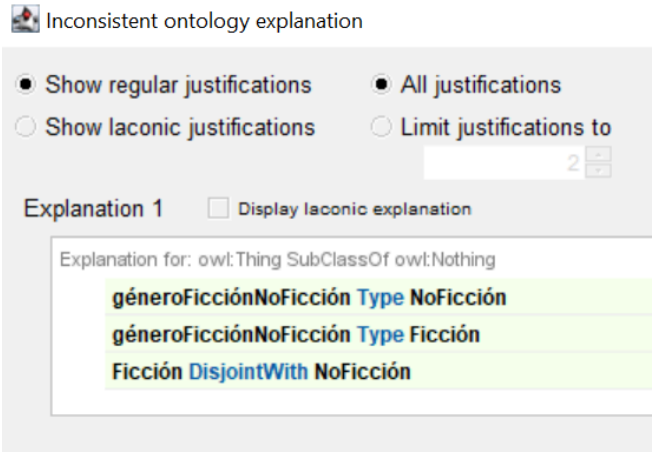
Figura 6: Panel de resultados de clasificación del razonador.

2.1. Pruebas simples (PS-X)

Para cada prueba, se ha creado un individuo con determinadas aserciones de propiedad (tanto para objetos como datos) y se ha iniciado el razonador ***HermiT 1.4.3.456***, para ver las inferencias que realiza sobre los tipos de cada individuo.

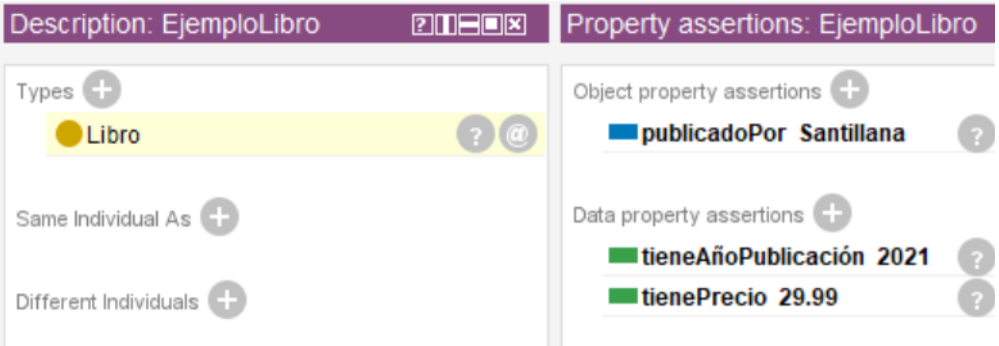
- **PS-1**: Libro con géneros inconsistentes (disjuntos).

Cuando se intenta iniciar el razonador, salta un aviso de error, dando la opción de explicar dónde se encuentran las inconsistencias.

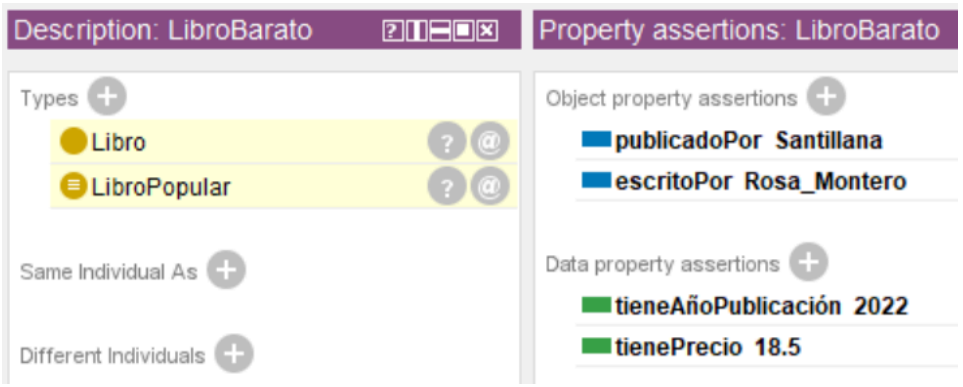


Se ha eliminado esta inconsistencia para poder realizar el resto de pruebas con el razonador.

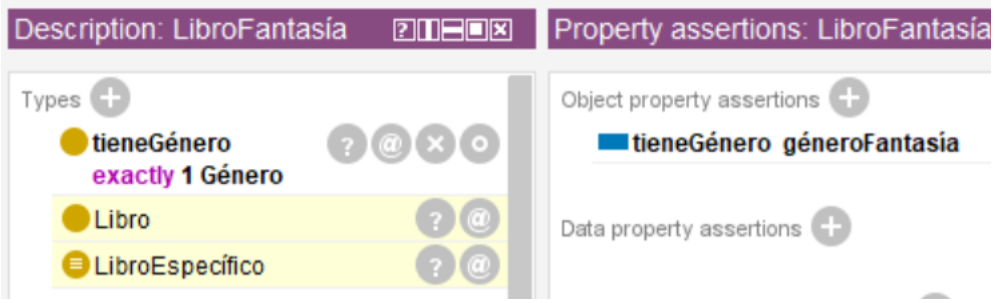
- **PS-2**: Ejemplo de libro.



➢ **PS-3**: Libro popular (barato).



➢ **PS-4**: Libro de un género específico.

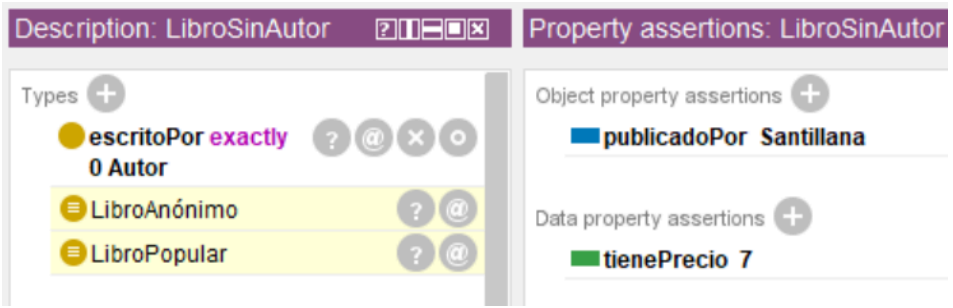


Tras una serie de pruebas, se ha observado que para que clasifique correctamente una instancia de LibroEspecífico, se debe concretar que el individuo va a tener exactamente un género, ya que este razonador no puede asumir por su cuenta que, aunque se haya definido un solo género (en este caso fantasía), este vaya a ser el único asociado a la propiedad tieneGénero.

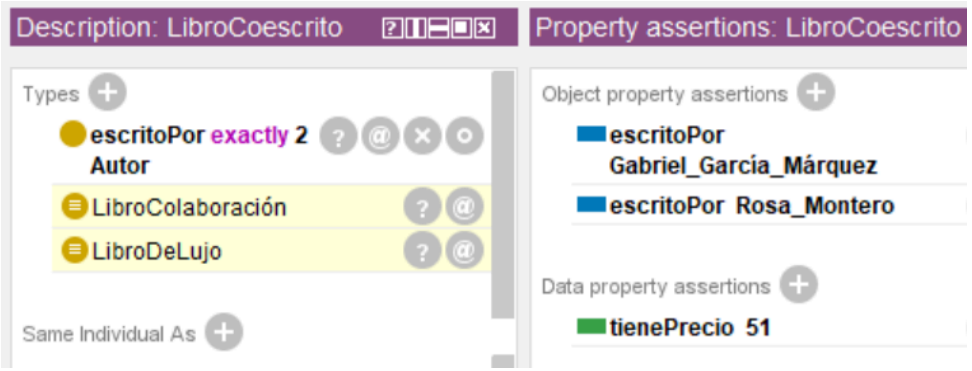
Lo mismo sucederá para instancias que tengan cardinalidades concretas para las propiedades de objetos, tal y como se observa con las subclases de libros con restricciones en cuanto al número de autores, que se van a ser vistas a continuación.

2.2. Pruebas más avanzadas (PA-X)

➢ **PA-1**: Libro anónimo (sin autor) y popular.

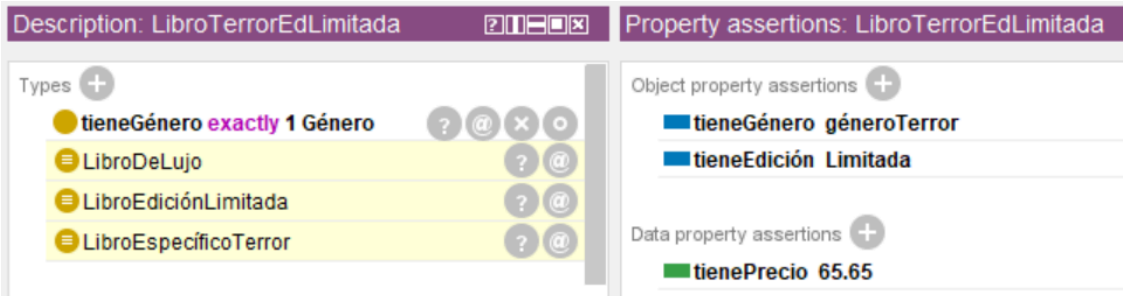


➢ **PA-2**: Libro con varios autores y de lujo.



➢ **PA-3**: Libro de ciencia ficción barato.

➢ **PA-4**: Libro específico de terror, edición limitada y de lujo.

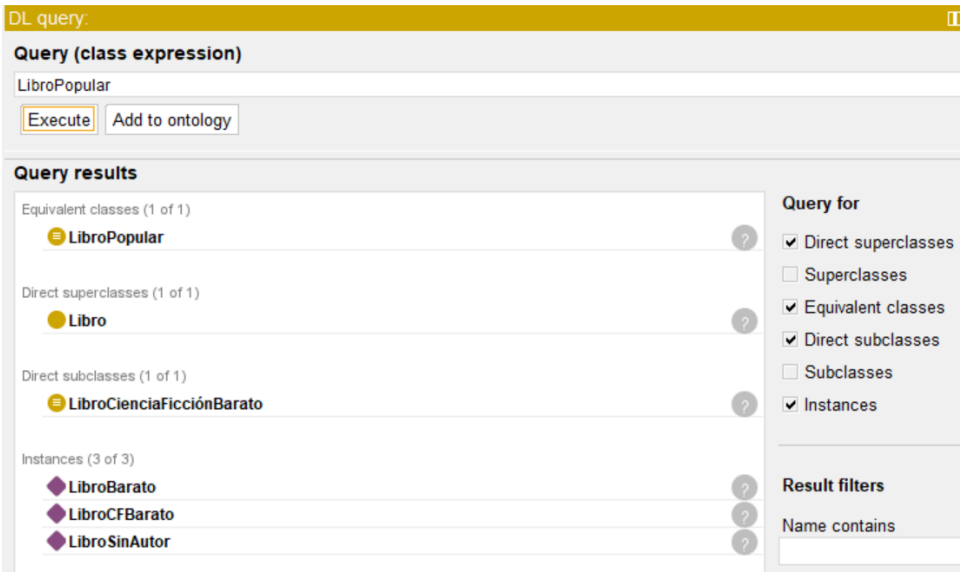


3. Otros complementos en Protégé

3.1. Consultas en DL query

En Protégé OWL, la herramienta DL Query permite verificar la consistencia y funcionalidad de la ontología, pudiendo realizar consultas basadas en lógica descriptiva. Facilita la comprobación de coherencia en clases, individuos y axiomas, la exploración de jerarquías de clases según condiciones específicas, y la validación de restricciones en individuos.

Se podrían consultar las superclases, subclases e instancias (tanto inferidas como establecidas) de una clase concreta, tomando como ejemplo ***LibroPopular***:



Nota: se han desmarcado algunos de los filtros (*Superclasses* y *Subclasses*) para evitar que devuelva como resultado la superclase `owl:Thing` y la subclase `owl:Nothing`.

Otro ejemplo, si quiero obtener las clases equivalentes, superclases directas e instancias de ***libros que tienen edición limitada***, se puede realizar la siguiente consulta:

3.2. Consultas en SPARQL

SPARQL es una herramienta útil y versátil para consultar y explorar datos estructurados en formato *RDF*, ampliamente utilizada en el ámbito de la web semántica.

Su capacidad para trabajar con ontologías permite explorar clases, propiedades y patrones complejos. Sin embargo, una de sus limitaciones es que no incluye automáticamente las clases e instancias inferidas por los razonadores. Por ello, se han creado algunas instancias explícitas de *Libro* (*Libro1*, *Libro2* y *Libro3*) previamente.

Una consulta simple es la recuperación de los autores, junto con los libros que han escrito, lo que resulta útil para ver todas las ***relaciones entre libros y autores***. Para ello, busca triples en el modelo RDF donde un libro tiene la relación *:escritoPor* hacia un autor.

SPARQL query:

```
PREFIX : <http://www.bookstore.com/ontology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?autor ?libro
WHERE {
  ?libro :escritoPor ?autor .
}
```

autor	
Gabriel_García_Márquez	LibroCoescrito
Rosa_Montero	LibroCoescrito
Rosa_Montero	LibroBarato
Gabriel_García_Márquez	Libro1
Gabriel_García_Márquez	Libro3
Rosa_Montero	Libro2

También se han realizado algunas consultas utilizando operadores de SPARQL, como para obtener la ***media del precio de todos los libros*** mediante el operador *AVG*.

SPARQL query:

```
PREFIX : <http://www.bookstore.com/ontology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT (AVG(?precio) AS ?precioPromedio)
WHERE {
  ?libro rdf:type :Libro .
  ?libro :tienePrecio ?precio .
}
```

precioPromedio
"32.333333333333333333333333"^^<http://www.w3.org/2001/XMLSchema#decimal>

Otra consulta interesante podría ser recuperar todos los ***libros que han sido publicados después del año 2000***, aprovechando la propiedad de datos que se ha definido anteriormente.

SPARQL query:

```
PREFIX : <http://www.bookstore.com/ontology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?libro ?año
WHERE {
  ?libro rdf:type :Libro .
  ?libro :tieneAñoPublicación ?año .
  FILTER(?año >= "2000"^^xsd:integer)
}
```

libro	año
Libro2	"2001"^^<http://www.w3.org/2001/XMLSchema#decimal>
Libro3	"2002"^^<http://www.w3.org/2001/XMLSchema#decimal>

La última consulta, algo más avanzada, combina algunos elementos de las consultas anteriores, consiste en listar los ***autores en orden descendente en cuanto a la suma total del precio de sus libros, mientras esta supere los 30***.

SPARQL query:

```
PREFIX : <http://www.bookstore.com/ontology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?autor (SUM(?precio) AS ?precioTotal)
WHERE {
  ?libro rdf:type :Libro .
  ?libro :escritoPor ?autor .
  ?libro :tienePrecio ?precio .
}
GROUP BY ?autor
HAVING (SUM(?precio) > 30)
ORDER BY DESC(?precioTotal)
```

autor	precioTotal
Rosa_Montero	"60"^^<http://www.w3.org/2001/XMLSchema#decimal>
Gabriel_García_Márquez	"37"^^<http://www.w3.org/2001/XMLSchema#decimal>