

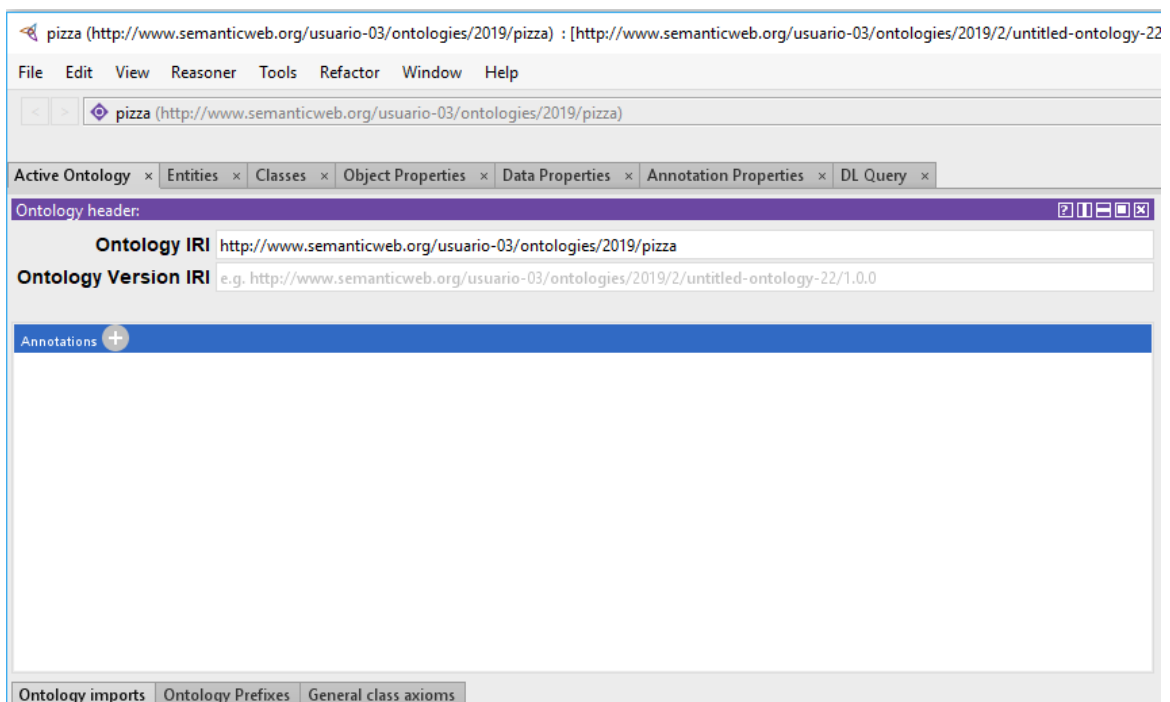
PRACTICA OWL

Introducción

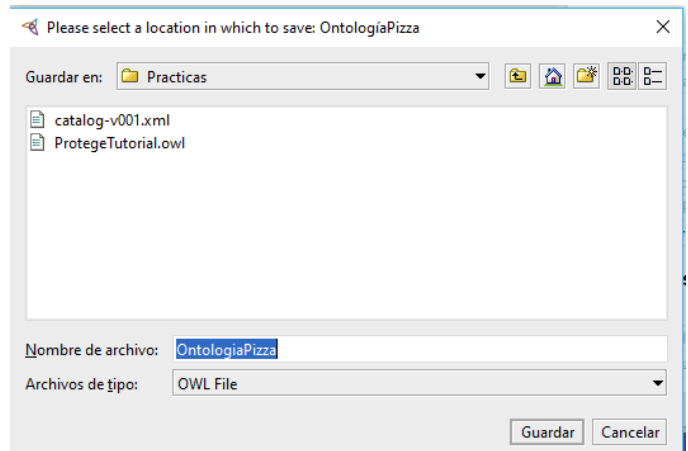
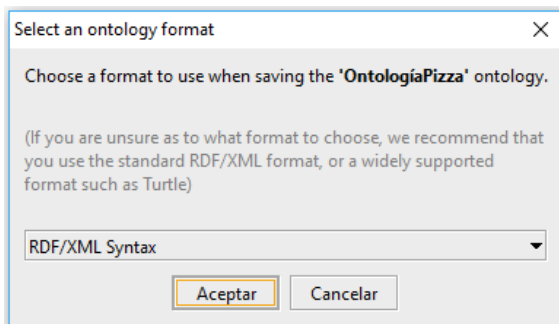
El objetivo de esta práctica es crear una ontología usando el editor Protégé. El documento muestra a través de una serie de ejercicios cómo crear una ontología de pizzas.

Ejercicio 1 – Crear una nueva ontología OWL

- Iniciar el editor protégé.
- Cada ontología se nombra usando un único Identificador de recursos (IRI). En el tab Active Ontology reemplace si lo desea el IRI predeterminado con <http://www.pizza.com/ontologias/pizza.owl>.

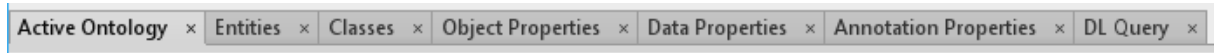


- También guarde la Ontología en un archivo en su computador. Puede navegar en el disco y guarde su ontología en un archivo nuevo, es posible que desee nombrar su archivo 'OntologíaPizza.owl'. Una vez que elija un archivo, presione "Finalizar"

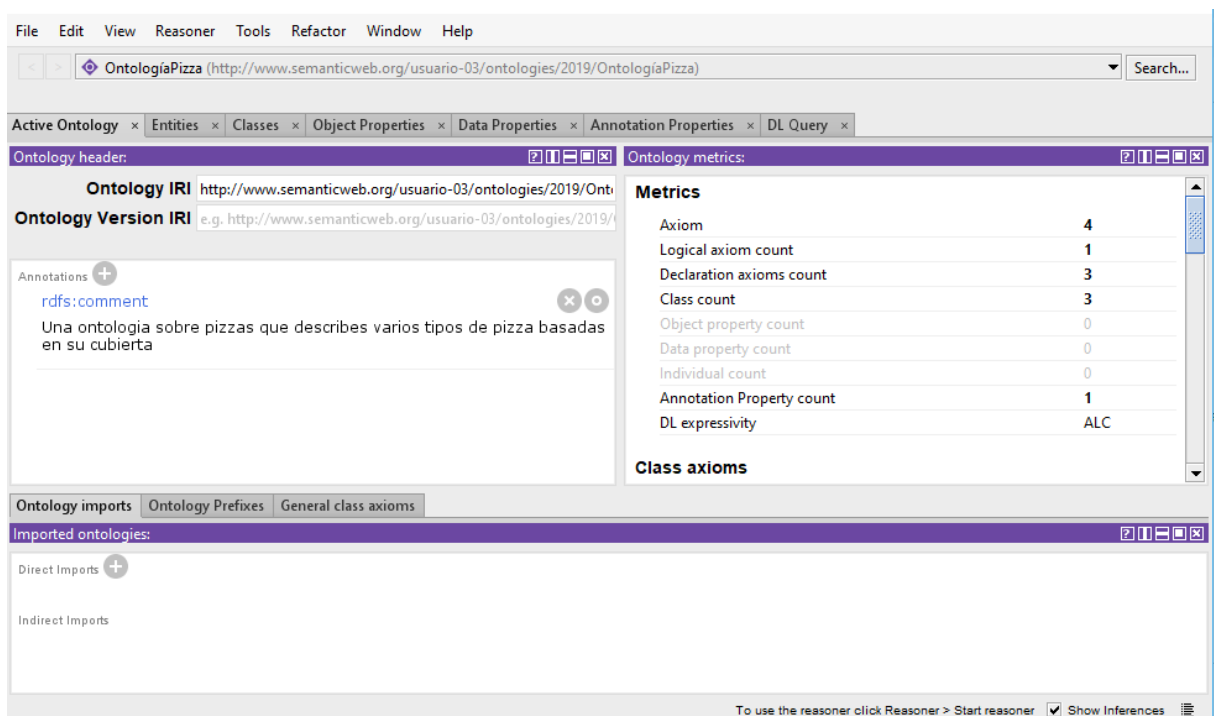


Ejercicio 2 – Agregar un comentario a la ontología OWL

- Asegúrese de que 'Active Ontology Tab' esté seleccionado.



- ▶ Active Ontology – Información general de la ontología
 - ▶ Entities: Muestra todas las entidades de la ontología
 - ▶ Classes - jerarquía de clases y definiciones
 - ▶ Object Properties y Data Properties - jerarquías de propiedades y definiciones
 - ▶ Annotation Properties - ontología de gestión y anotación.
 - ▶ DL Query: Ejecutar consultas sobre los elementos de la ontología
- En la vista "Annotations ", haga clic en el icono "Add" (+).
 - Aparecerá una ventana de edición en la tabla. Seleccione 'rdfs:comment' de la lista y escriba su comentario en el cuadro de texto en el panel de la derecha. Ingrese un comentario tal como “Una ontología sobre pizzas que describes varios tipos de pizza basadas en su cubierta” y pulse OK para asignar el comentario.



Ejercicio 3 – Crear las clases Pizza, PizzaCubierta y PizzaBase e indique que son clases disjuntas

- Cree la jerarquía de clases
 - ▶ Seleccione la pestaña **Entities**
 - ▶ Asegúrate de que **owl:Thing** este seleccionado en la jerarquía de clases
 - ▶ Seleccione **Tools | Create Class hierarchy...**

- Cree la jerarquía para denotar subclases
- ▶ Finalice el asistente y deje marcada la casilla **Make classes disjoint**.


Enter hierarchy

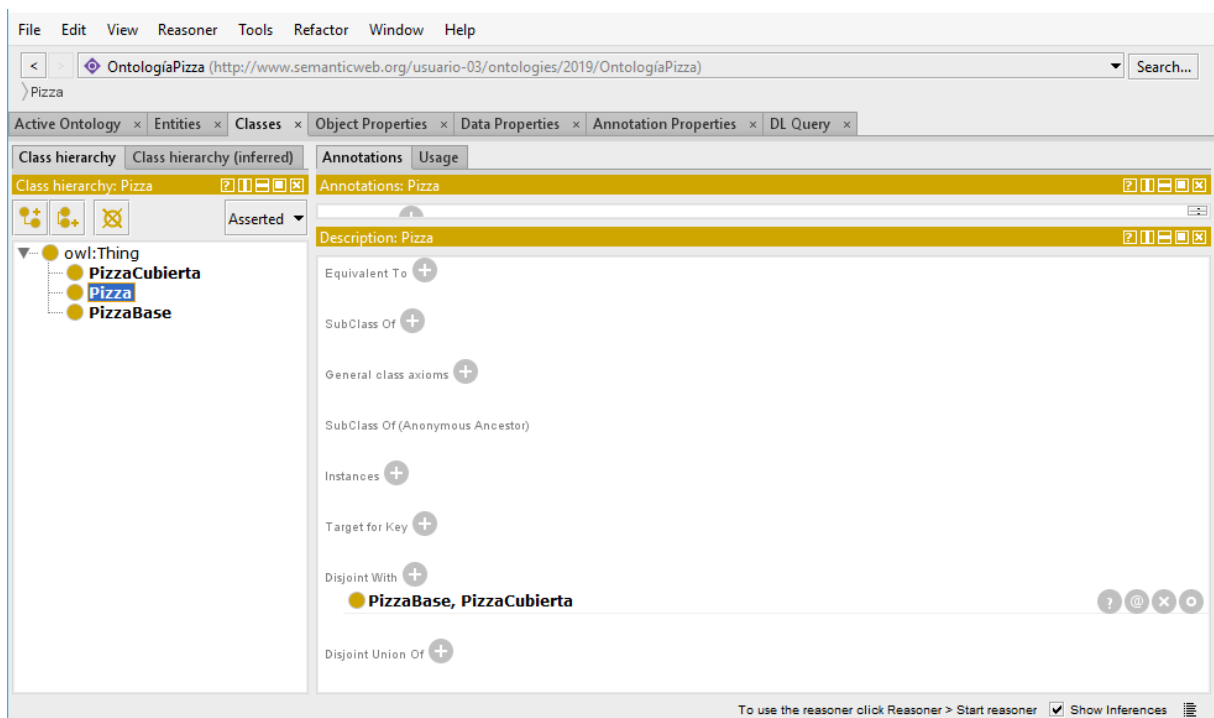
Please enter one name per line. You can use tabs to indent names to create a hierarchy.

```
Pizza
PizzaBase
PizzaCubierta
```

Prefix

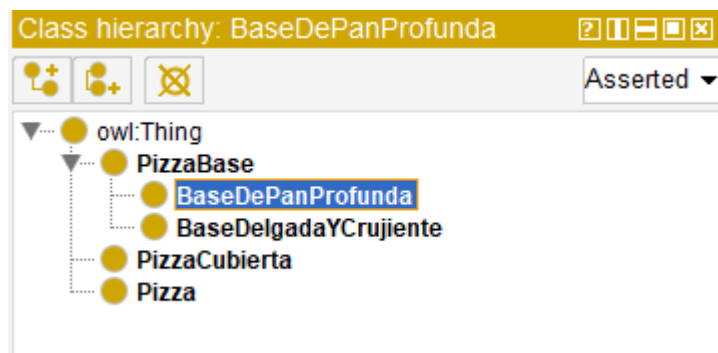
Suffix

- Borre las clases creadas y repita el proceso usando un método alternativo
 - ▶ Asegúrese de que la Ficha 'Classes' esté seleccionada.
 - ▶ Presione el ícono 'Add Subclass' (). Este botón crea una nueva clase como subclase de la clase seleccionada (en este caso queremos crear una subclase de Owl:Thing).
 - ▶ Aparecerá un cuadro de diálogo para que nombre su clase, ingrese Pizza y presione Enter.
 - ▶ Repita los pasos anteriores para agregar las clases PizzaCubierta y también PizzaBase, asegurándose de que Owl:Thing esté seleccionado antes de presionar el botón 'Add Subclass' para que las clases se creen como subclases de Owl:Thing.
 - ▶ Seleccione la clase Pizza en la jerarquía
 - ▶ Presione el botón 'Disjoint With' en la vista 'class description', esto abrirá un diálogo donde puede seleccionar múltiples clases para separarlas. Esto hará que PizzaBase y PizzaCubierta (las clases hermanas de Pizza) se separen de Pizza.
- Usando cualquiera de los métodos descritos anteriormente el resultado debe mostrarse como se indica en la figura



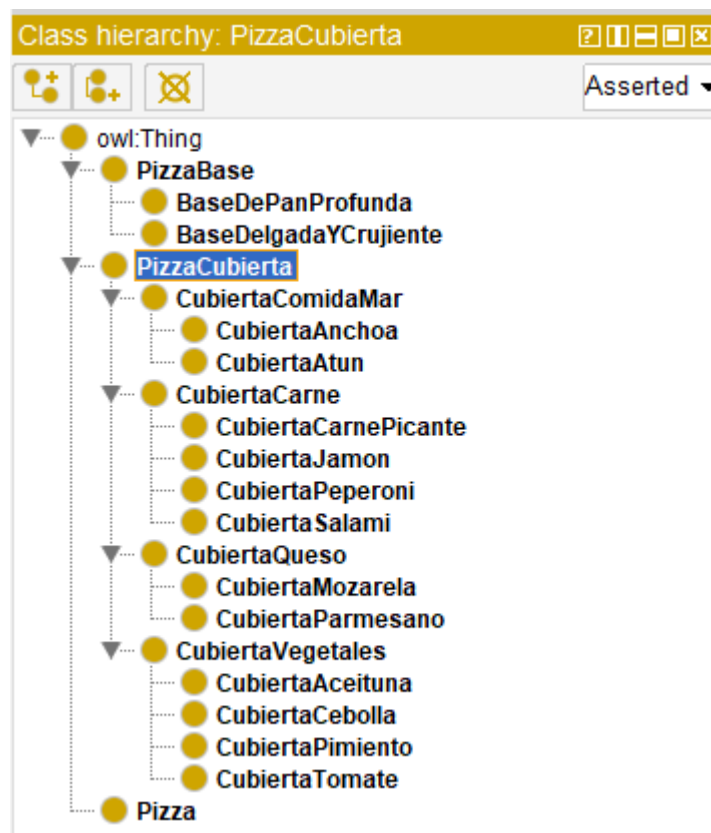
Ejercicio 4 – Crear la jerarquía de clases restante

- Crear la jerarquía de clases de la clase PizzaBase
 - ▶ Seleccione PizzaBase como clase raíz
 - ▶ Agregue las clases
 - BaseDelgadaYCrujiente
 - BaseDePanProfunda



- Crear la jerarquía de clases de la clase PizzaCubierta
 - ▶ Usando Tools->“Create class Hierarchy...” cree la jerarquía
 - CubiertaQueso
 - ▶ CubiertaMozarela
 - ▶ CubiertaParmesano
 - CubiertaCarne
 - ▶ CubiertaJamon
 - ▶ CubiertaPeperoni
 - ▶ CubiertaSalami
 - ▶ CubiertaCarnePicante

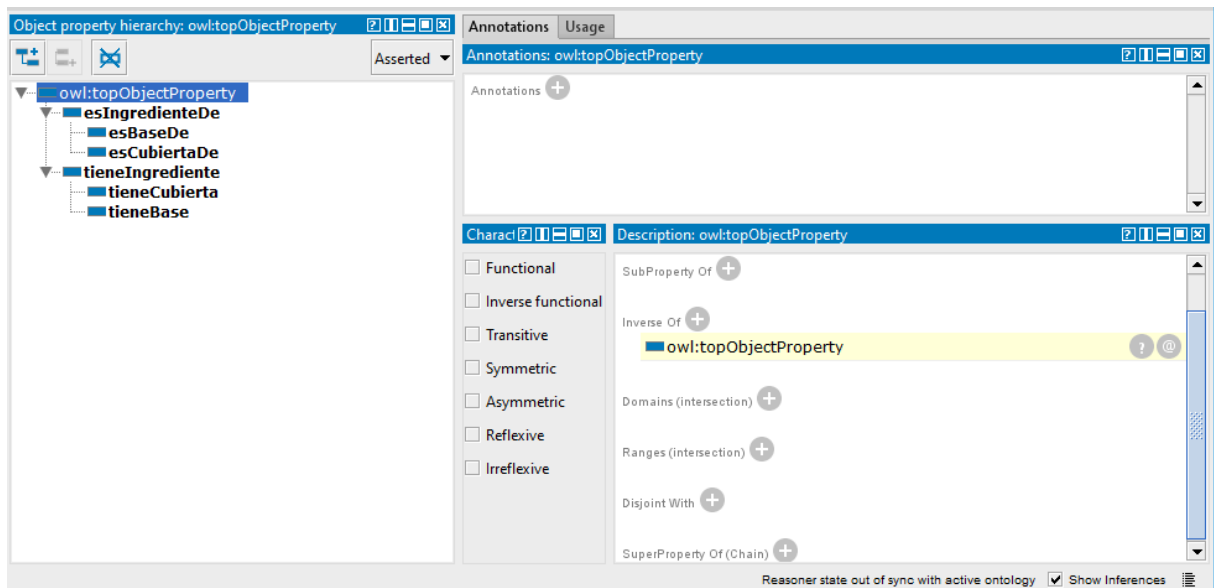
- CubiertaComidaMar
 - ▶ CubiertaAnchoa
 - ▶ CubiertaAtun
 - CubiertaVegetales
 - ▶ CubiertaAceituna
 - ▶ CubiertaCebolla
 - ▶ CubiertaPimiento
 - ▶ CubiertaTomate
- ▶ **Nota:** Esta jerarquía indica que todos los individuos que son miembros de la clase CubiertaTomate son miembros de la clase CubiertaVegetales y miembros de la clase PizzaCubierta.



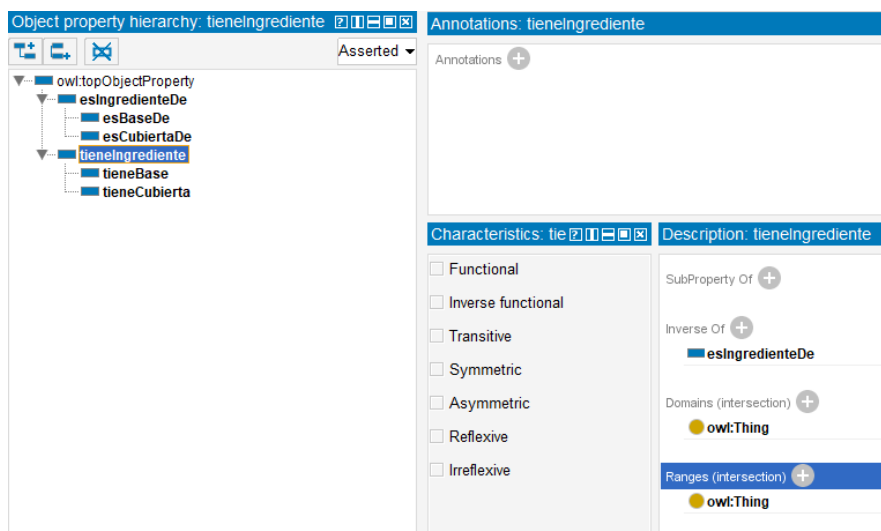
Ejercicio 5: Crear propiedades objeto (Object Property) y subpropiedades

- ▶ Recuerde que OWL soporta dos tipos de propiedades
 - Object Properties: Relaciones entre dos individuos.
Corresponde a relaciones entre clases
 - Data Properties: Relaciones entre un individuo y un dato
Corresponde a atributos de una clase
- Cambie a la pestaña "Object Properties". Utilice el botón "Add sub property" para crear una nueva propiedad de Objeto.
- Nombre la propiedad como tieneIngrediente usando el cuadro de dialogo "Create a new object property"
- Usando el mismo procedimiento cree las subpropiedades tieneBase, tieneCubierta

- Crear otra propiedad denominada esIngredienteDe y las subpropiedades esBaseDe y esCubiertaDe



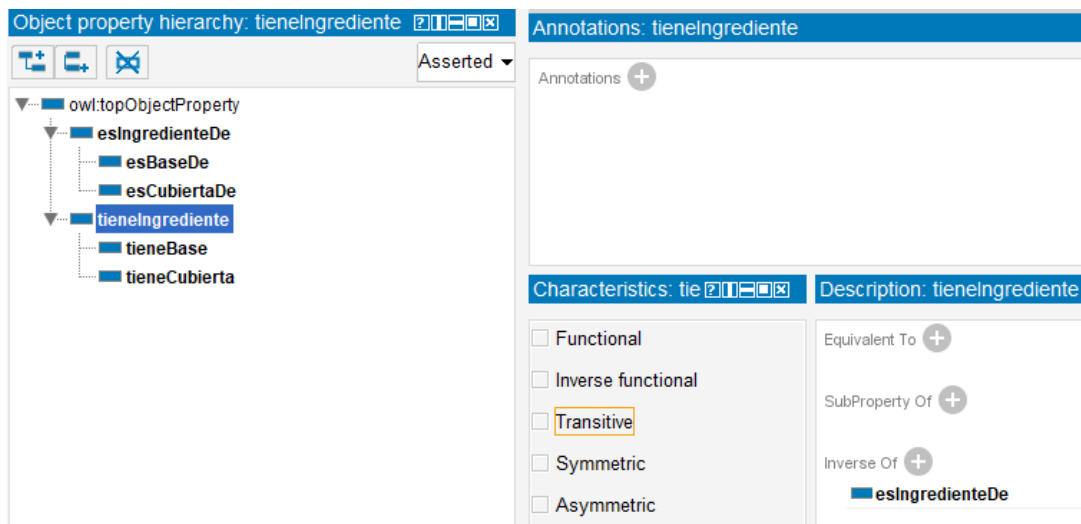
- Defina los dominios y rangos de las propiedades creadas (ObjectProperties)
 - ▶ Recuerde que en OWL las propiedades tienen Dominio (Domain) y Rango (Range)
En el caso de las propiedades ObjectProperties, el dominio y rango son:
 - Domain: Clases de individuos
 - Range: Clases de individuos
- Especifique el dominio y rango de la propiedad tieneIngrediente como owl:Thing
 - ▶ Nota: Esto indica que la propiedad tieneIngrediente puede tener un individuo de cualquier tipo como dominio y rango
 - ▶ Seleccione la propiedad tieneIngrediente
 - ▶ Cambie a la ventana Description y seleccione como Domains (intersection) and Ranges (intersection) la clase owl:Thing



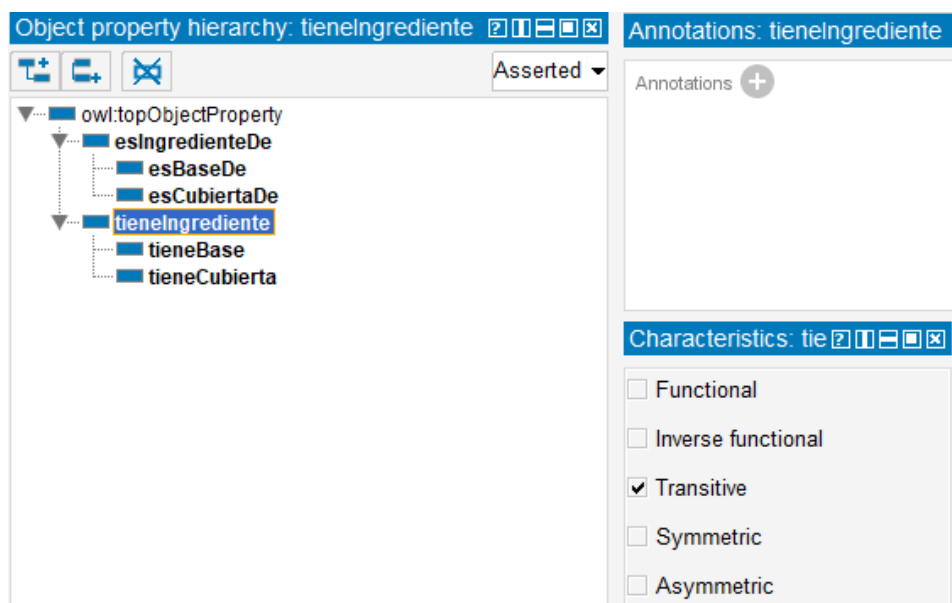
- Siga el mismo procedimiento para especificar el dominio para tieneCubierta a Pizza y rango PizzaCubierta
 - ▶ Nota: Esto indica que la propiedad tieneCubierta pertenece a la clase Pizza y tiene como rango cualquier elemento de la clase PizzaCubierta

The screenshot shows the Protégé interface with the 'tieneCubierta' property selected. The 'Object property hierarchy: tieneCubierta' panel on the left shows the hierarchy: owl:topObjectProperty > esIngredienteDe > esBaseDe > esCubiertaDe > tieneIngrediente > tieneBase > tieneCubierta. The 'Annotations: tieneCubierta' panel is empty. The 'Characteristics: tieneCubierta' panel shows several unchecked options: Functional, Inverse functional, Transitive, Symmetric, Asymmetric, Reflexive, and Irreflexive. The 'Description: tieneCubierta' panel shows the property's configuration: 'Equivalent To' is empty, 'SubProperty Of' is 'tieneIngrediente', 'Inverse Of' is empty, 'Domains (intersection)' is 'Pizza', and 'Ranges (intersection)' is 'PizzaCubierta'.

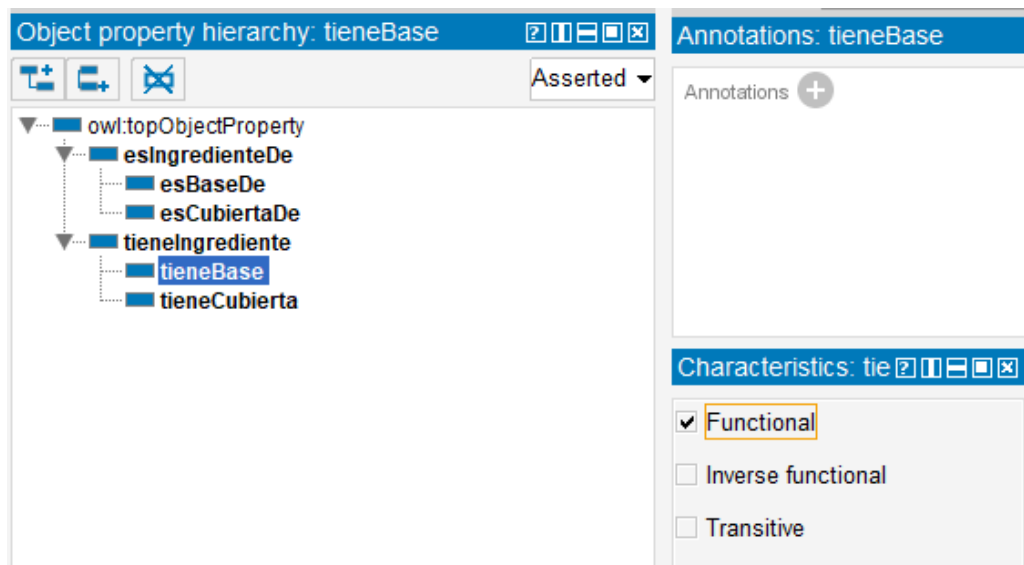
- **Determine y especifique el dominio y rango de la propiedad tieneBase**
- Defina las propiedades inversas de otras propiedades (ObjectProperties)
 - ▶ Algunas propiedades tienen la característica de ser la inversa de otra. (Inverse Of)
 - Si alguna propiedad vincula al individuo A con el individuo B, entonces su propiedad inversa vinculará al individuo B con el individuo A
 - E.j. tieneIngrediente versus esIngredienteDe
 - ▶ Especifique que la propiedad tieneIngrediente es inversa de esIngredienteDe. Para ello seleccione la propiedad tieneIngrediente.
 - ▶ Cambie a la ventana Description y seleccione como Inverse Of la propiedad esIngredienteDe



- Defina las características de las propiedades (ObjectProperties)
 - ▶ Algunas características pueden ser aplicadas las propiedades ObjectProperties
 - Functional
 - Inverse functional
 - Transitive
 - Symmetric
 - Antisymmetric
 - Reflexive
 - Irreflexive
- Haga la propiedad tieneIngrediente transitiva
 - ▶ Nota: Un ingrediente de un ingrediente, es además un ingrediente de una Pizza
 - ▶ Seleccione la propiedad tieneIngrediente
 - ▶ En la ventana Characteristics seleccione la opción Transitive



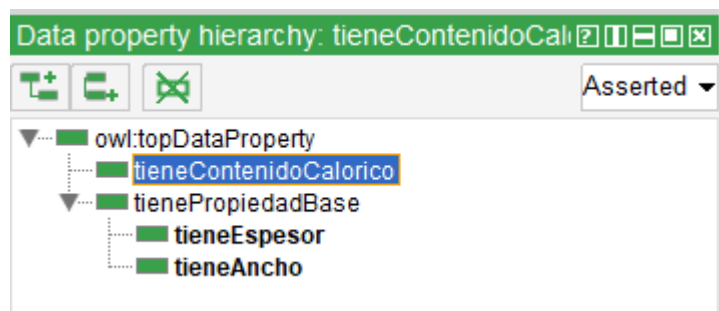
- Haga que la propiedad tieneBase sea funcional
 - ▶ Nota: Una Pizza tiene solamente una base
 - ▶ Seleccione la propiedad tieneBase
 - ▶ En la ventana Characteristics seleccione la opción Functional



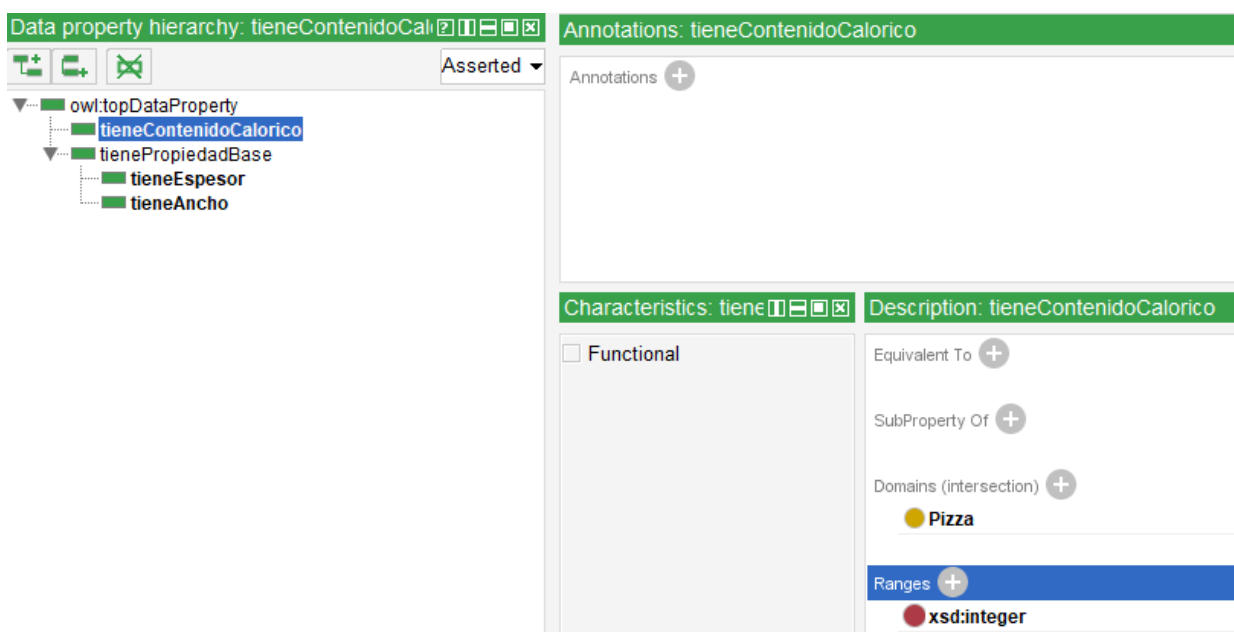
- Especifique que la propiedad tieneBase es inversa de esBaseDe.
- Especifique que la propiedad tieneCubierta es inversa de esCubiertaDe.

Ejercicio 6: Crear propiedades valor (Data Properties) y subpropiedades

- ▶ Recuerde que OWL soporta dos tipos de propiedades
 - Object Properties: Relaciones entre dos individuos.
Corresponde a relaciones entre clases
 - Data Properties: Relaciones entre un individuo y un dato
Corresponde a atributos de una clase
- Cambie a la pestaña "Data Properties". Utilice el botón "Add sub property" para crear una nueva propiedad de tipo dato.
- Nombre la propiedad como tienePropiedadBase usando el cuadro de dialogo "Create a new data property"
- Usando el mismo procedimiento cree las subpropiedades tieneAncho, tieneEspesor
- Crear otra propiedad denominada tieneContenidoCalorico



- Defina los dominios y rangos de las propiedades creadas (DataProperties)
 - ▶ Recuerde que en OWL las propiedades tienen Dominio (Domain) y Rango (Range)
En el caso de las propiedades DataProperties, el dominio y rango son:
 - Domain: Clases de individuos
 - Range: Tipos de datos: Ejemplo: Integer, String, Boolean, etc.
- Especifique el dominio de la propiedad tieneContenidoCalorico como la clase Pizza y como rango Integer
 - ▶ Nota: Esto indica que la propiedad tieneContenidoCalorico pertenece a la clase Pizza y sus valores serán enteros
 - ▶ Seleccione la propiedad tieneContenidoCalorico
 - ▶ Cambie a la ventana Description y seleccione como Domains (intersection) la clase Pizza
 - ▶ En la misma ventana Description seleccione como Ranges en la pestaña Built in datatypes el tipo xsd:integer

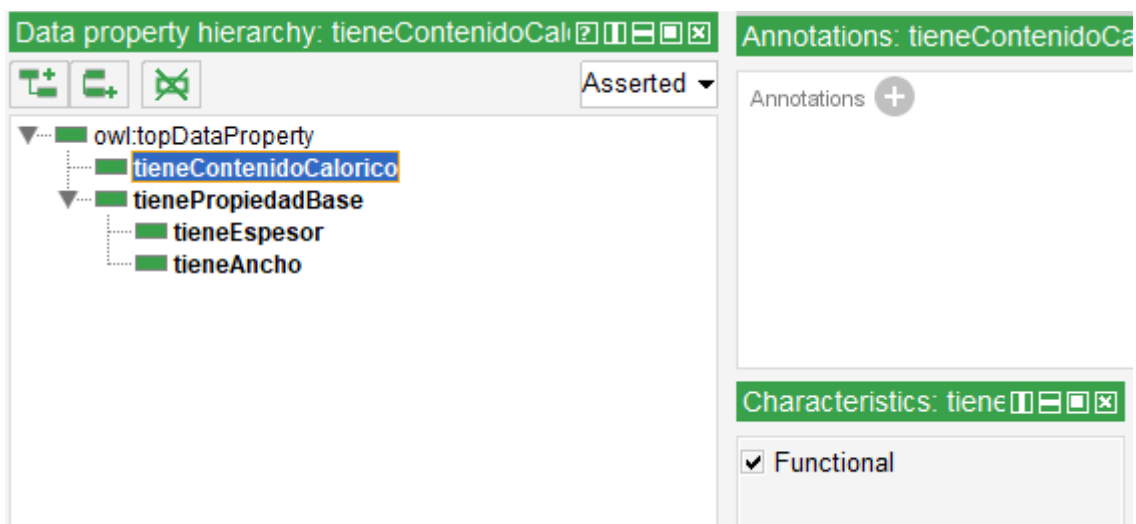


- Especifique el dominio de la propiedad tienePropiedadBase como la clase PizzaBase
- Especifique que el rango de la subpropiedad tieneAncho es xsd:Integer
- Especifique que el rango de la subpropiedad tieneEspesor es xsd:Float
- Responda las siguientes preguntas

Porqué la propiedad tienePropiedadBase no requiere rango?

Porqué las subpropiedades tieneAncho y tieneEspesor no tiene dominio?

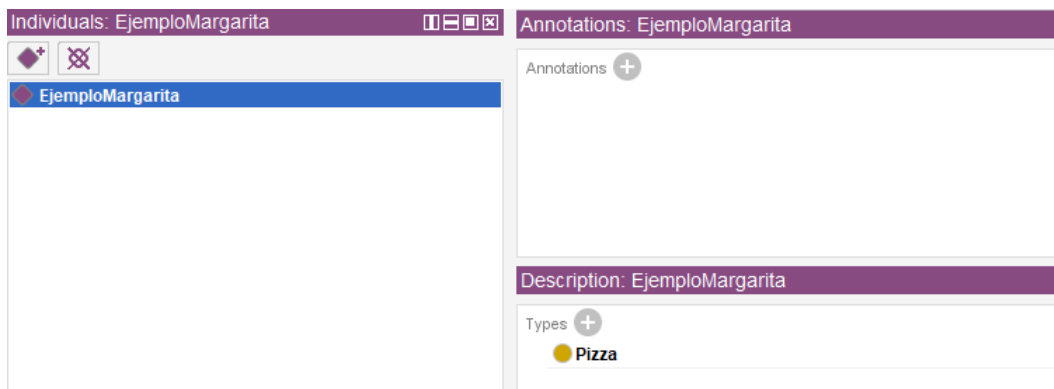
- ▶ Recuerde que las propiedades DataProperties NO pueden tener propiedades invertidas
- Defina las características de las propiedades (DataProperties)
 - ▶ Estas propiedades pueden tener como características únicamente
 - Functional
- Especifique que las propiedades tieneContenidoCalorifico, tieneAncho y tieneEspesor son funcionales
 - ▶ Nota: Un pizza base tiene un solo ancho, un solo espesor y un solo contenido calórico
 - ▶ Seleccione la propiedad tieneContenidoCalorifico
 - ▶ En la ventana Characteristics seleccione la opción Functional



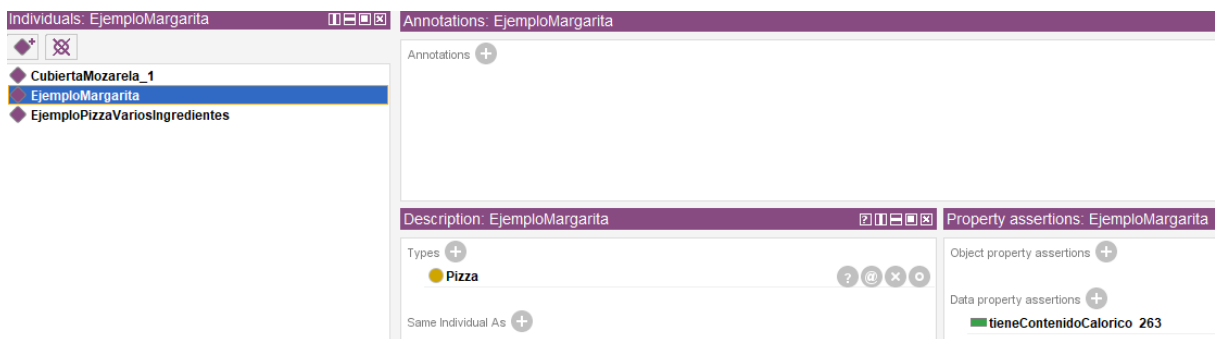
- Repita este mismo procedimiento para las propiedades tieneAncho y tieneEspesor

Ejercicio 7: Crear varios individuos (Individuals)

- ▶ Recuerde que un 'individuo' es otro nombre para 'Instancia' en la terminología ontológica. Las instancias forman el ABox de la base de Conocimiento.
- Agregue como instancia de Pizza el individuo EjemploMargarita con contenido calórico de 263
 - ▶ Cambie a la pestaña "Individuals"
 - ▶ Presione el botón "Add Individual" y agregue la instancia "EjemploMargarita"
 - ▶ En la ventana Description vaya a la sección "Types" e indique que la clase a la que pertenece esta instancia es Pizza



- ▶ Para agregar el contenido calórico, esto se corresponde con la DataProperty denominada tieneContenidoCalorico
- ▶ Teniendo seleccionada la instancia EjemploMargarita, en la ventana Property Assertions vaya a la sección “Data property assertions” y presione el botón “+”



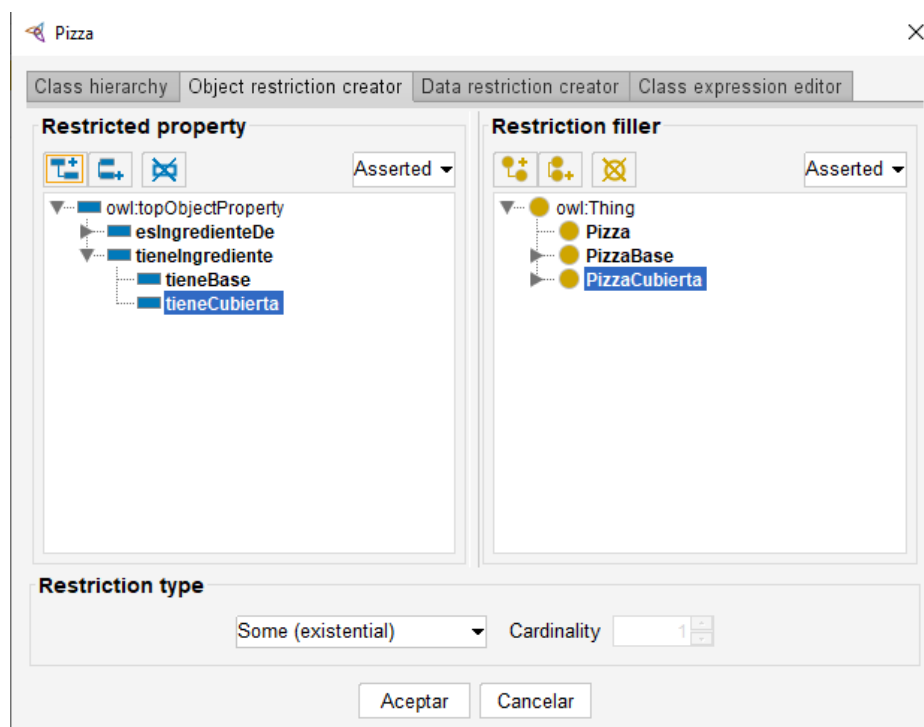
- ▶ En la ventana escoja del lado izquierdo la propiedad “tieneContenidoCalorico” y en la parte derecha agregue el valor 263



- Agregue una instancia para Pizza denominado “EjemploPizzaVariosIngredientes” con contenido calórico 723
- Agregue una instancia para CubiertaMozarela denominado “CubiertaMozarela 1”
- Cree una clase llamada País (hija de Owl:Thing) y agregue valores como individuos a Italia, EEUU, Inglaterra, Francia, y Alemania

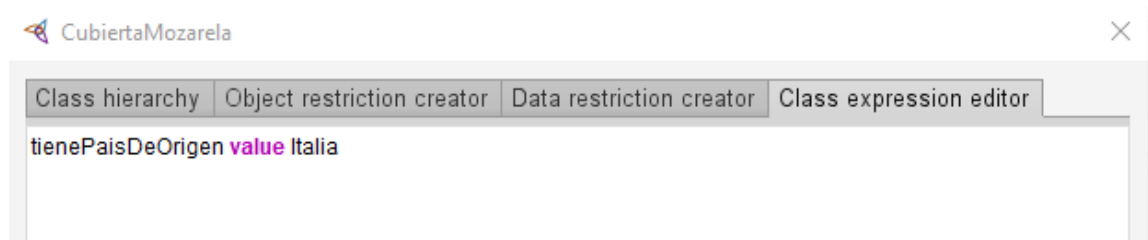
Ejercicio 8: Crear algunas restricciones

- ▶ Una restricción describe una clase anónima
El conjunto de **subclases e individuos** que satisfacen la restricción dada.
Aplica a: Classes, Object Properties y Datatype Properties
- ▶ Existen dos tipos de restricciones
 - Restricciones de cardinalidad
 - Restricciones de valor
- Cree una restricción de cardinalidad para especificar que Pizza tiene al menos una PizzaCubierta
 - ▶ Agregue una restricción existencial a Pizza sobre tieneCubierta alguna PizzaCubierta
 - ▶ En lógica descriptiva: $\text{Pizza} \sqsubseteq \exists \text{tieneCubierta.PizzaCubierta}$
 - ▶ Seleccione la clase Pizza al que se le va a agregar una restricción
 - ▶ En la ventana Description, agregue a la sección SubClassOf "tieneCubierta some PizzaCubierta" presionando el botón "+".
 - ▶ En la ventana cambie a la pestaña Object restriction creator pues tieneCubierta es una propiedad ObjectProperty.
 - ▶ En la sección Restricted Property elija la propiedad tieneCubierta.
 - ▶ En la sección Restriction Filter elija la clase PizzaCubierta.
 - ▶ En la sección Restriction Type elija la restricción Some(existential).



- Cree una restricción de valor para especificar que CubiertaMozarela tiene a "Italia" como su país de origen
 - ▶ **Como no existe la propiedad entonces cree previamente la propiedad tienePaisDeOrigen con País como rango. Responda las siguientes preguntas:**
 - **Qué tipo de propiedad es?**

- Al no definir el dominio de esta propiedad de manera explícita, responda cuál es su dominio?
 - ▶ Seleccione la clase CubiertaMozarela al que se le va a agregar una restricción
 - ▶ En la ventana Description, agregue a la sección SubClassOf "tienePaisDeOrigen value Italia" presionando el botón "+".
 - ▶ En la ventana cambie a la pestaña Class expression editor
 - ▶ Agregue la restricción tienePaisDeOrigen value Italia



- Cree una restricción de cardinalidad para especificar que Pizza tiene exactamente 1 PizzaBase

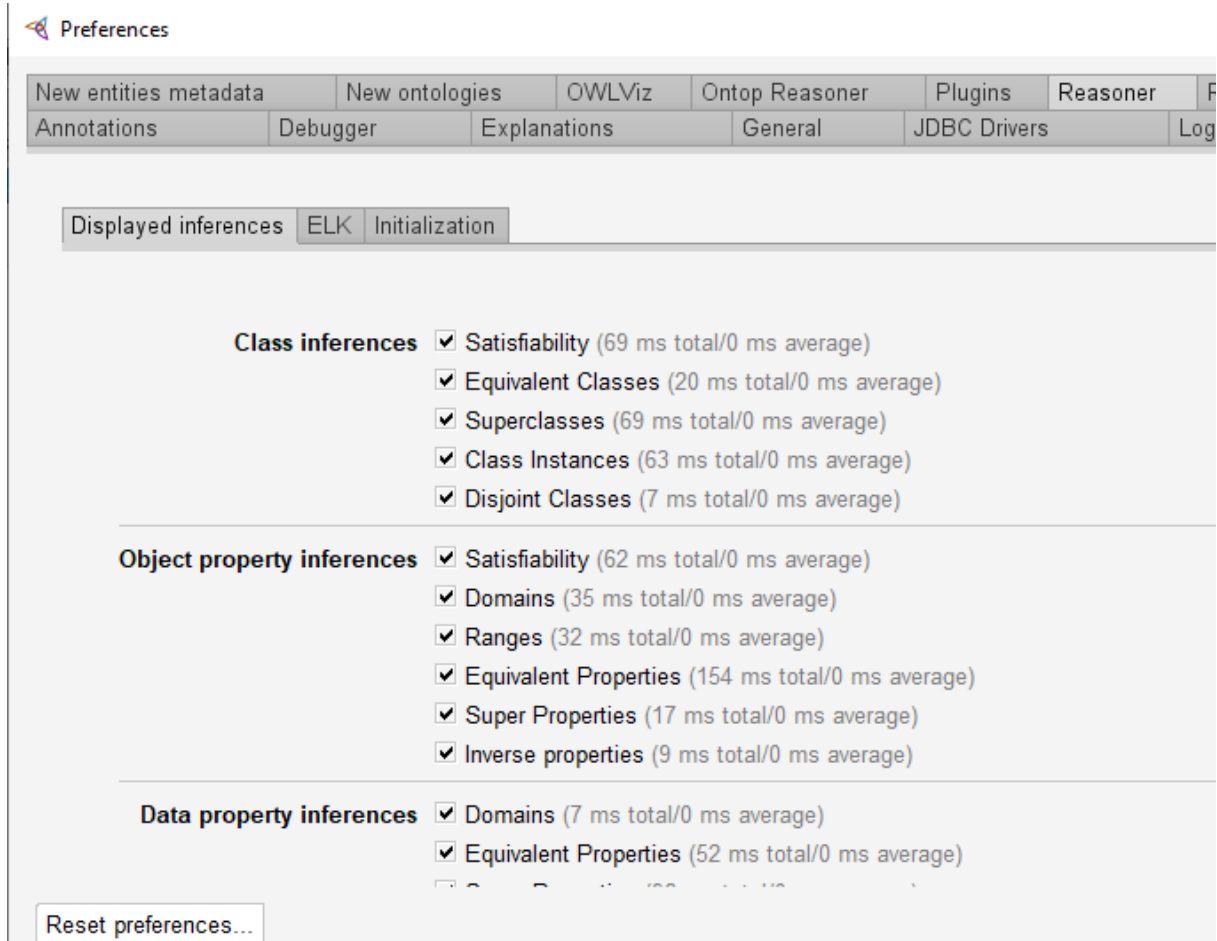
Ejercicio 9: Crear algunas restricciones adicionales y clases complejas

- Usando como información toda la explicación de los puntos previos ejecutar las siguientes acciones
 - ▶ Crear una clase llamada "**Picante**"
 - ▶ Crear tres nuevas clases (**Caliente**, **Mediano**, y **Suave**) como subclases de la clase "**Picante**"
 - ▶ Haga todas las subclases disjuntas
 - ▶ Crear una propiedad objeto llamada **tienePicante** con **Picante** como rango
 - ▶ Crear un axioma de equivalencia para la clase **Picante** agregando a "equivalentTo" el axioma "**Caliente or Mediano or Suave**"
- Usando como información toda la explicación de los puntos previos ejecutar las siguientes acciones
 - ▶ Defina la clase **Pais** como una clase enumerada
 - ▶ En la sección equivalentTo agregue el axioma {EEUU, Inglaterra, Francia, Alemania, Italia}

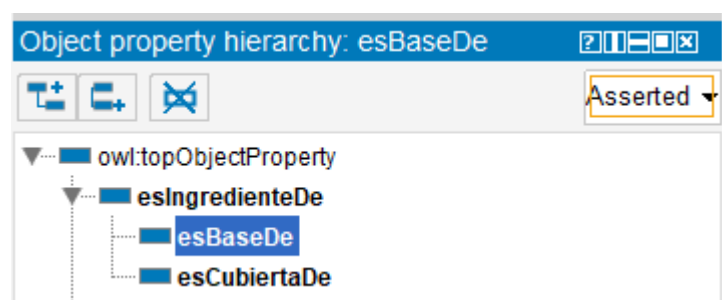
Ejercicio 10: Usar un razonador para determinar los axiomas inferidos

- ▶ Recuerde que un razonador es un software capaz de inferir consecuencias lógicas de un conjunto de hechos o axiomas afirmados
- Seleccionar y ejecutar un razonador para inferir nuevo conocimiento

- ▶ Vaya al menú “Reasoner” y seleccione “Pellet” como razonador
- ▶ Antes de ejecutar el razonador en el menú “Reasoner” seleccione la opción Configure... y en la pestaña Reasoner y luego en la opción Displayed Inferences verifique que todas las opciones están marcadas. Esto permitirá que el razonador puede inferir sobre cualquier elemento de la ontología.



- ▶ En el mismo menú, seleccione la opción “Start Reasoner...”
- Revisar el conocimiento inferido
 - ▶ En la pestaña Object Properties seleccione la propiedad esBaseDe
 - ▶ Para observar el conocimiento inferido escoja la opción Inferred en el menú que consta en la parte superior derecha de esta ventana.



- Explique porqué para la propiedad *esBaseDe* aparece como dominio (domains) y rango (range) los valores inferidos
- Explique porqué para la propiedad *esCubiertaDe* aparece los valores del dominio y rango inferidos
- Explique porqué para las propiedades *tieneAncho* y *tieneEspesor* aparece los valores de dominio inferidos.

Ejercicio 11: Determinar las inferencias ejecutadas por el razonador

- ▶ Es necesario recordar que un razonador puede determinar **inconsistencia**
- Explique porqué se produce una inconsistencia al ejecutar las siguientes acciones
 - ▶ Agregue una clase de prueba llamada **CubiertaInconsistente** la cual es una subclase de ambos **CubiertaQueso** y **CubiertaVegetales**
 - ▶ Ejecute el razonador
- Explique porqué se logra consistencia al ejecutar las siguientes acciones
 - ▶ Remueva que las clases **CubiertaQueso** y **CubiertaVegetales** son disjuntas
 - ▶ Ejecute el razonador
 - ▶ **NOTA:** Luego de explicar las razones de la consistencia vuelva a colocar que las clases **CubiertaQueso** y **CubiertaVegetales** como disjuntas y elimine la clase **CubiertaInconsistente**

Ejercicio 12: Determinar las inferencias ejecutadas por el razonador

- ▶ Es necesario recordar que un razonador puede ejecutar una **clasificación**
- Explique cómo se clasifican las clases **Margarita**, **Americana**, **AmericanaCaliente** y **Soho** al ejecutar las siguientes acciones
 - ▶ Crear una subclase de Pizza llamada "PizzaConNombre"
 - ▶ Crear como subclases de PizzaConNombre las clases **Margarita**, **Americana**, **AmericanaCaliente** y **Soho**
 - ▶ Haga a las pizzas creadas disjuntas
 - ▶ Describa que la clase **Margarita** tiene alguna cubierta de **Mozarela** y tiene alguna cubierta de **Tomate**
 - ▶ Describa que la clase **Americana** tiene alguna cubierta de **Mozarela** y tiene alguna cubierta de **Tomate** y tiene alguna cubierta de **Peperoni**
 - ▶ Describa que la clase **AmericanaCaliente** tiene alguna cubierta de **Mozarela** y tiene alguna cubierta de **Tomate** y tiene alguna cubierta de **Peperoni** y tiene alguna cubierta de **CarnePicante**
 - ▶ Especifique que la clase **Soho** tiene alguna cubierta de **Mozarela** y tiene alguna cubierta de **Tomate** y tiene alguna cubierta de **Aceituna** y tiene alguna cubierta de **Parmesano**
 - ▶ Defina que la clase **PizzaConQueso** es una pizza que tiene alguna cubierta de queso

- ▶ Ejecute el razonador
- Explique porqué las clases Margarita y Soho no son clasificadas como PizzaVegetariana al ejecutar las siguientes acciones?
 - ▶ Defina que una PizzaVegetariana es una pizza que tiene solamente queso o vegetales en la cubierta
 - ▶ Ejecute el razonador

Ejercicio 13: Determinar las inferencias ejecutadas por el razonador

- Explique cómo se clasifican las clases Margarita y Soho al ejecutar las siguientes acciones
 - ▶ La descripción actual de la clase Margarita indica que tiene alguna cubierta de Mozarela y tiene alguna cubierta de Tomate, pero podría tener más cubiertas.
 - ▶ Para que la clase Margarita solo pueda aceptar estos dos tipos de cubierta hay que agregar un axioma de cierre.
 - ▶ NOTA: Un axioma de cierre en una propiedad consiste en una restricción universal que actúa a lo largo de la propiedad para decir que solo puede ser llenado por los rellenos especificados. La restricción tiene un relleno que es la unión de los rellenos que se dan en las restricciones existenciales para la propiedad
 - ▶ Agregue un axioma de cierre a la clase Margarita indicando que tiene solamente una cubierta de Mozarela o Tomate

Class hierarchy: Margarita

owl:Thing

- └─ Pais
 - └─ Picante
 - └─ Pizza
 - └─ PizzaConNombre
 - Americana
 - AmericanaCaliente
 - Margarita**
 - Soho
 - PizzaConQueso
 - PizzaVegetariana
 - PizzaBase
 - BaseDelgadaYCrujiente
 - BaseDePanProfunda
 - PizzaCubierta
 - CubiertaCarne
 - CubiertaCarnePicante

Annotations: Margarita

Annotations +

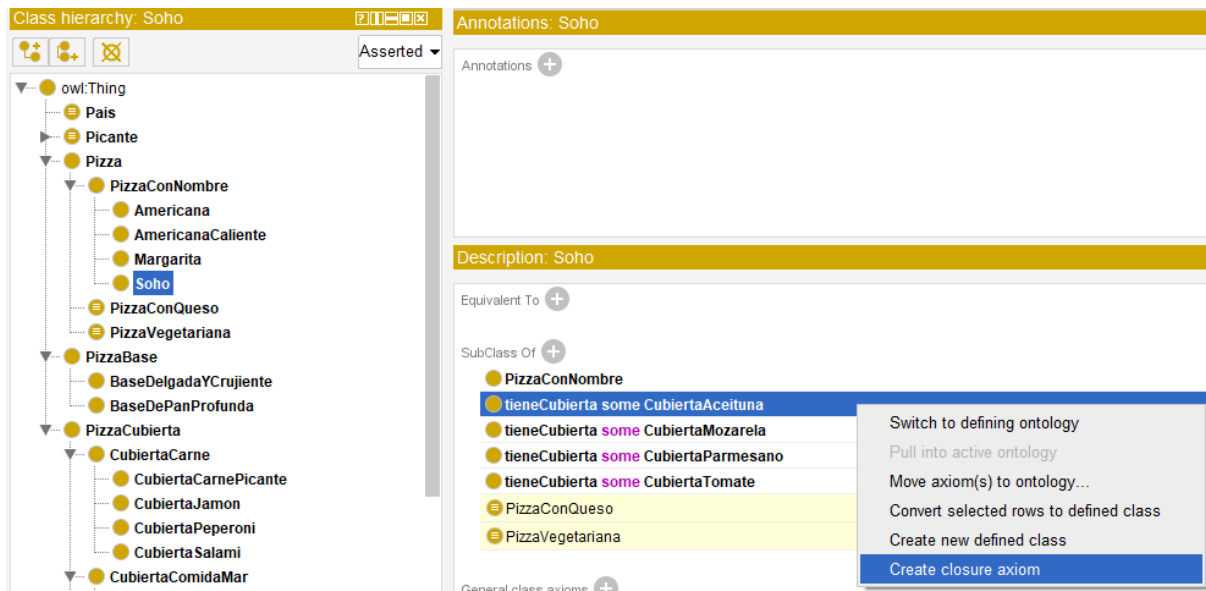
Description: Margarita

Equivalent To +

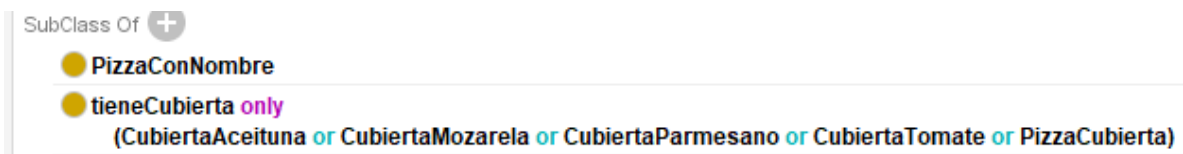
SubClass Of +

- PizzaConNombre
- tieneCubierta **only** (CubiertaMozarela **or** CubiertaTomate)
- tieneCubierta **some** CubiertaMozarela
- tieneCubierta **some** CubiertaTomate

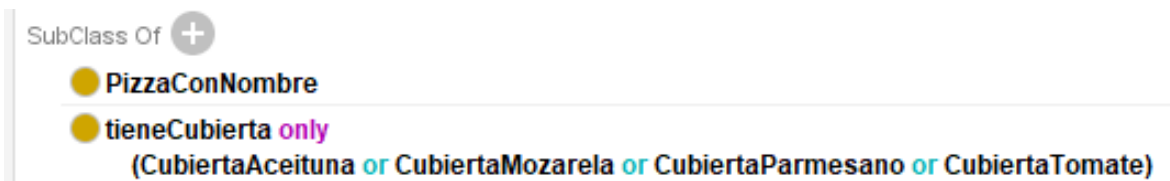
- ▶ También se puede agregar a la especificación de la clase Soho un axioma de cierre, usando el método siguiente en lugar de hacerlo manualmente
- ▶ Seleccione la clase Soho. En la ventana Description seleccione alguna de las restricciones existenciales y con el botón derecho escoja la opción “Create closure axiom”



- ▶ Edite el axioma de cierre creado, eliminando la clase PizzaCubierta pues caso contrario el axioma se entendería como que la clase Soho acepta cubiertas de Aceituna, Mozzarella, Parmesano, Tomato o cualquier otra cubierta (PizzaCubierta)



- ▶ El axioma de cierre para la clase Soho debería quedar de la siguiente forma



- ▶ Describa axiomas de cierre para las clases Americana y AmericanaCaliente siguiendo cualquiera de los métodos descritos previamente
- ▶ Ejecute el razonador

Ejercicio 14: Determinar las inferencias ejecutadas por el razonador

- **Crear una clase pizza no vegetariana e identificar qué clases son clasificadas como PizzaNoVegetariana al ejecutar las siguientes acciones**
 - ▶ Defina que una PizzaNoVegetariana es una pizza que no es una PizzaVegetariana
 - ▶ Esta nueva clase deber ser disjunta de la clase PizzaVegetariana
 - ▶ Ejecute el razonador
- **Crear una clase que tenga al menos tres cubiertas e identificar qué clases son clasificadas como PizzaInteresante al ejecutar las siguientes acciones**
 - ▶ Defina que “PizzaInteresante” es una pizza que tiene como mínimo 3 tipos de cubiertas
 - ▶ Ejecute el razonador