# Глава 5 Ограничения типов данных (Datatype Properties)

В разделе 4.4 главы 4 мы представили свойства в OWL, но только описали свойства объекта — то есть, отношения между индивидами. В этой главе мы обсудим и покажем примеры свойств типа данных. Свойства Datatype связывают индивида со значением типа данных схемы XML или rdf. Другими словами они описывают связи между индивидом и значениями данных. Большая часть описанных в главе 4 характеристик свойств не может использоваться с свойством Datatype. Далее в этой главе мы будем описывать характеристики свойств, которые применимы к свойствам данных.

Свойства Datatype могут быть созданы в закладках, показанных на рисунке 5.1.

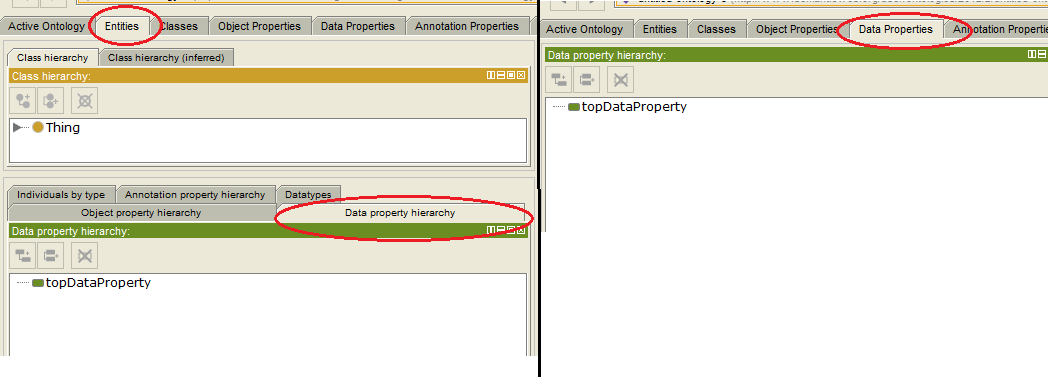


Рисунок 5.1 Варианты задания ограничений DataType

Мы будем использовать свойства datatype для описания содержания калорий в пицце. Затем мы будем использовать некоторые числовые диапазоны для широкой классификации конкретных пицц, как высококалорийных или низкокалорийных. Для того чтобы это сделать нам нужно выполнить следующие действия:

Создать свойство типа данных hasCalorificContentValue, которое будет использоваться для описания калорийности пиццы.

Создать несколько примеров экземпляров пицц с конкретным содержимым калорий.

Создать два класса для категоризации пицц с высоким и низким содержанием калорий.

Теперь давайте сделаем это в Protege.

Свойство datatype может использоваться для индивида, относительно конкретных значений данных, которые могут быть типизированными и нетипизированными.

**Упражнение 46: Создайте свойство DataType с именем hasCalorificContentValue**

|  |  |
| --- | --- |
|  | 1. Перейти на вкладку «Свойства Datatype» использовать «Добавить свойства Datatype» |

Нет ничего в свойствах данных для предотвращения выполнения классификации уровня класса, но мы создадим некоторые индивиды как примеры для классификации, поскольку это, вероятно, слишком сильно будет сказано, что все MargheritaPizzas имеют точно 263 калории.

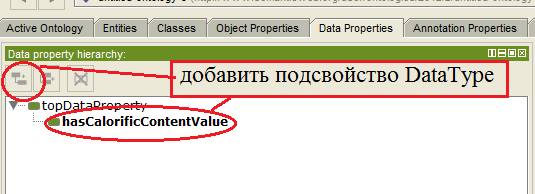


Рисунок 5.1а Добавление свойства DataType

☺- инструменты для работы со свойствами DataType аналогичны тем, которое мы использовали при работе со свойствами объектов.

**Упражнение 47: Создание примера экземпляра пиццы**

☺- здесь мы пытаемся создать некоторое множество индивидов, которые будут обладать нашим свойством **hasCalorificContentValue.** В дальнейшем мы этих индивидов распределим по именованным классам пицц. В отличии от Protégé – фрейма, в Protégé 4 индивиды имеют изначально при создании только имя, а потом при описании у них появляются свойства из тех свойств, которые мы зададим также отдельно. При этом набор свойств может быть абсолютно индивидуальным.

|  |  |
| --- | --- |
|  | 1.Выбрать закладку «Entities Tab» или «Individuals Tab» или «Classes». |
|  | Нажмите на кнопку Добавить индивида и создайте экземпляр под названием в любом классе/подклассе Pizza одним из способов (рисунки 5.1б,в,г). |
|  | В диалоговом окне, изображенном на рисунке 5.2 связать индивида Margherita свойством hasCalorificContentValue со значением типа integer 263. |
|  | Добавить аналогично несколько индивидов с различными значениями, включая экземпляр QuattroFormaggio с 723 калориями. |

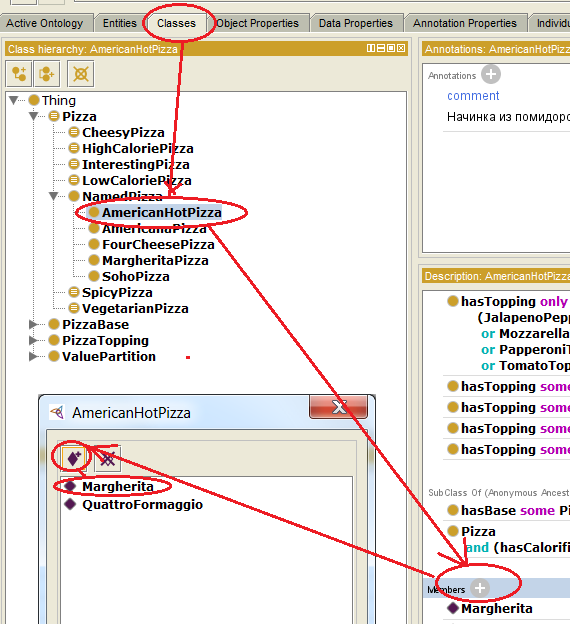


Рисунок 5.1б. 1-способ создания экземпляра класса

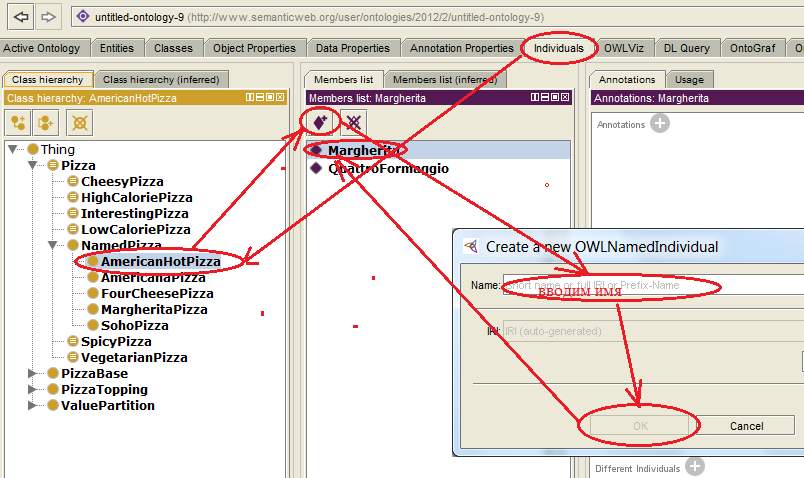


Рисунок 5.1в. 2й-способ создания экземпляра класса

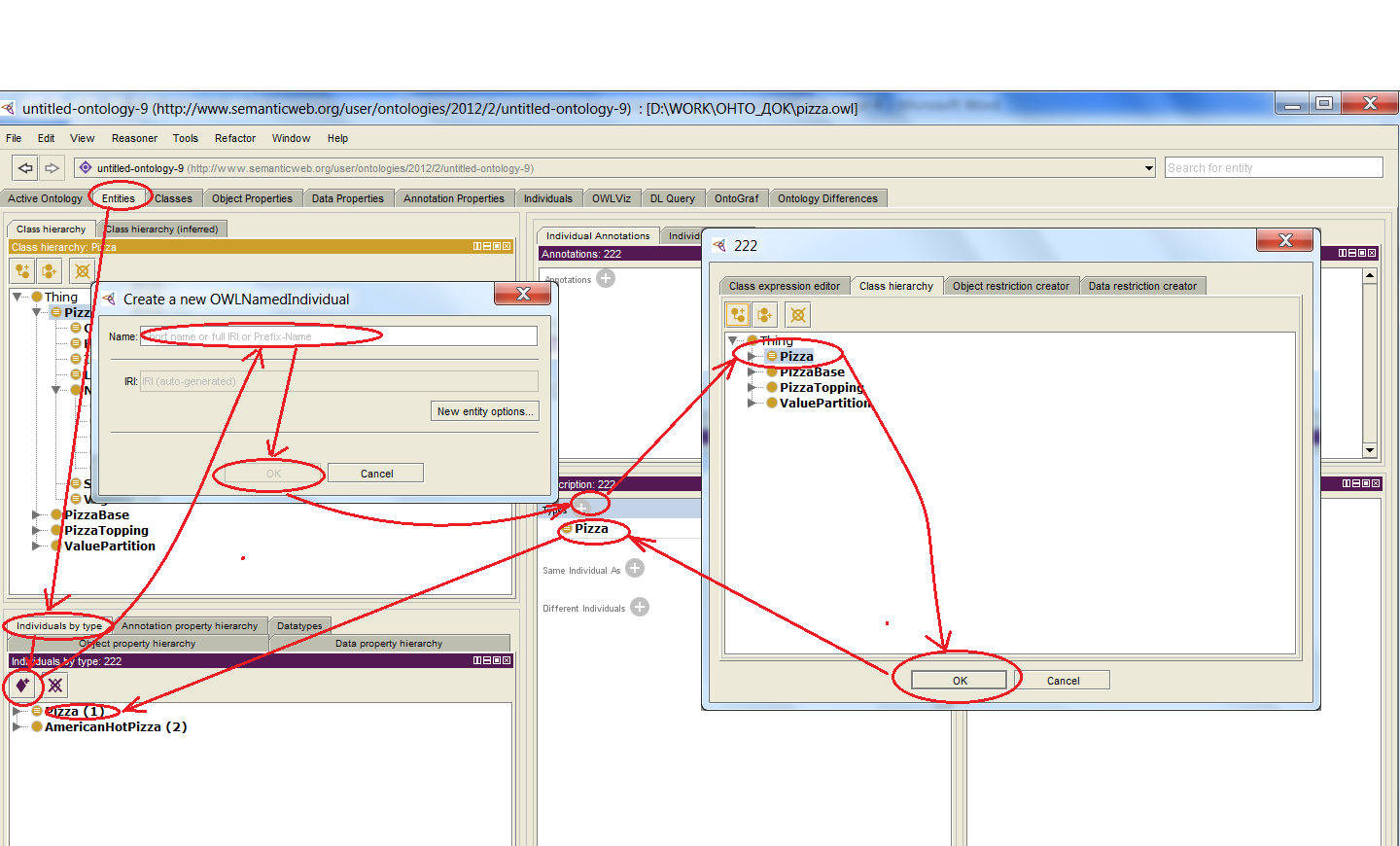


Рисунок 5.1г. 3й-способ создания экземпляра класса

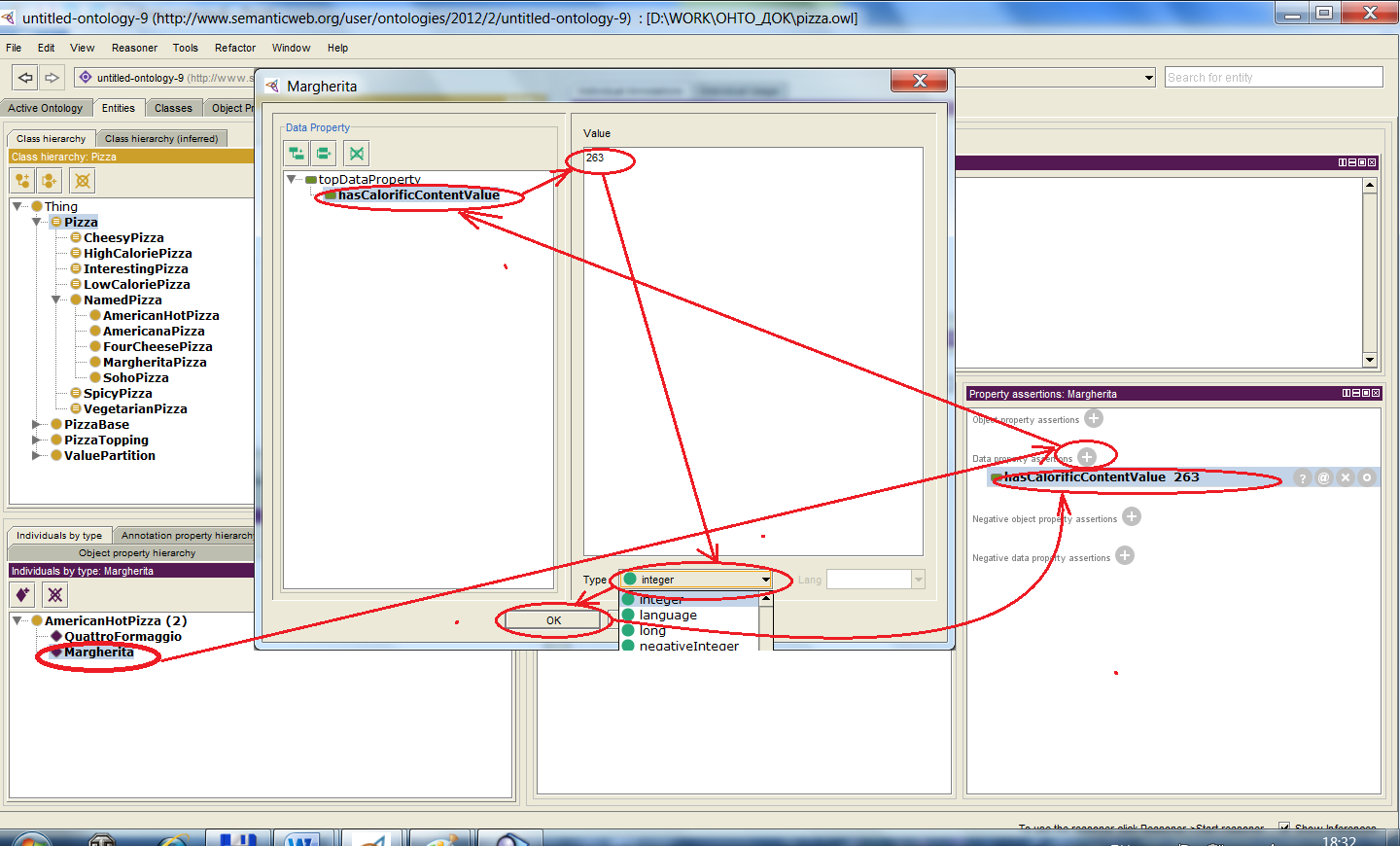


Рисунок 5.2 Связывание индивида свойством hasCalorificContentValue со значением типа integer 263.

Свойство datatype может также использоваться в ограничених относительно индивидов с известным типом данных. Построенные типы данных указываются в словарь схемы XML и включают в себя целые числа, с плавающей точкой, строки, логические значения и т.д.

**Упражнение 48: Создание ограничения DataType которое объявит, что все пиццы имеют энергетическую ценность (калорийность)**

|  |  |
| --- | --- |
|  | 1.Выбрать закладку «Classes» или «Entities». |
|  | 2. Выбрать Pizza и на панели описания класса выбрать «+» в строке «SubClass Of». Это вызовет диалоговое окно редактора ограничений. |
|  | 3. Выбрать закладку «data restriction creator» и ввести «hasCalorificContentValue some integer», как показано на рисунке 5.3, затем нажать “Ok” |

Ограничение будет введено, как показано на рисунке 5.4

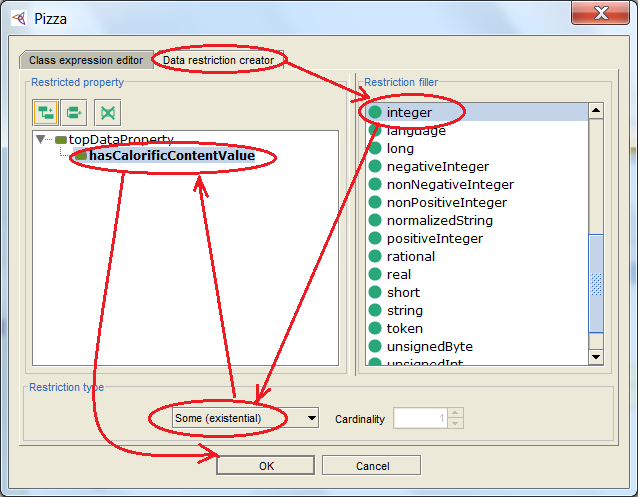


Рисунок 5.3. Задание ограничения DataType

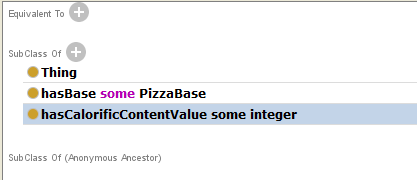


Рисунок 5.4. Вид экзистенциального ограничения DataType на класс Pizza

|  |  |
| --- | --- |
|  | Теперь мы заявили что все пиццы имеют по крайней мере одно значение энергетической ценности (и что значение должно быть целым числом). |

В дополнение к использованию предопределенного набора типов данных можно также специализировать использование DataType путем указания ограничений для возможных значений. Например, это применимо для указания диапазона численных значений.

С помощью свойства datatype, созданного нами, мы будем теперь создавать определенные классы, которые задают диапазон интересующих калорийностей. Мы создадим определения, которые используют **minlnclusive** и **maxExclusive** фасеты, которые могут применяться для числовых типов данных. Класс HighCaloriePizza будет создан для любых пицц с калорийностью не меньше чем 400.

**Упражнение 49: Создание класса HighCaloriePizza высококалорийных пицц, имеющих энергетическую ценность не меньше чем 400.**

|  |  |
| --- | --- |
|  | 1.Выбрать закладку «Classes» или «Entities». |
|  | 2.Выбрать класс Pizza в иерархии классов |
|  | 3.Создать подкласс с именем HighCaloriePizza |
|  | 4. Выбрать HighCaloriePizza и на панели описания класса выбрать «+» в строке «SubClass Of». Это вызовет диалоговое окно редактора ограничений. |
|  | 5. Выбрать закладку «data restriction creator» и ввести «hasCalorificContentValue some integer», как показано на рисунке 5.3, затем нажать “Ok”. В окне описания появится ограничени, аналогичное тому, что мы задали для Pizza |
|  | 6. Выбрать кнопку «Edit» в строке ограничения, откроется окно, как на рисунке 5.4. Добавьте фасет [>=400] в строку ограничения, и введите Ok |
|  | 7.Конвертируйте primitive класс в definied (Ctrl+d). Описание класса будет, как на рисунке 5.5 |
|  | 8. Создайте LowCaloriePizza так же, но определите его как эквивалентный пицце и имеющий ограничение калорийности hasCalorificContentValue менее 400 (любая пицца имеет энергетическую ценность [< 400]). Обратите внимание на то, что определение не перекрывается с HighCaloriePizza. |

Теперь у вас есть две категории пиццы, которые должны охватывать любого индивида, который имеет энергетическую ценность. Мы теперь должны проверить, как осуществится классификация индивидов, которых мы создали.

Мы будем использовать реазонер для выполнения классификации.

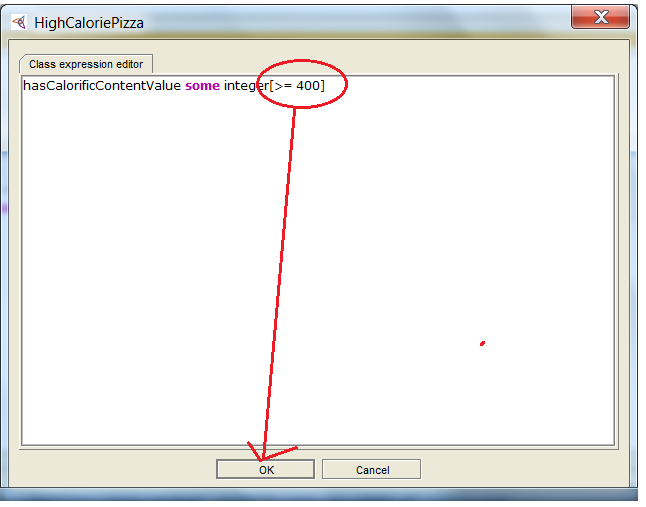


Рисунок 5.4 Добавление фасета диапазона

**Упражнение 50: Классифицировать индивидов класса Pizza на основе их ограничений по hasCalorificContentValue**

|  |  |
| --- | --- |
|  | 1. Выберите start в меню Reasoner или нажмите «Synchronize reasoner», если он активирован. Реазонер проведет классификацию и покажет иерархию выводимых классов. |
|  | 2. Выберите HighCaloriePizza. Вы сможете увидеть результаты рассуждений, показанные на панели описания класса на желтом фоне пунктирной границей. |
|  | 3. Раздел «members» (рисунок 5.5а) должен включать экземпляр «QuattroFormaggio» и возможно других индивидов, если мы их введем и опишем, что они имеют значение 400 или более калорий. |
|  | 4. Выберите LowCaloriePizza. Раздел «members» должен включать экземпляр «ExampleMargherita» и возможно другие индивиды, указанные как имеющие значение энергетической ценности меньше 400 калорий. |

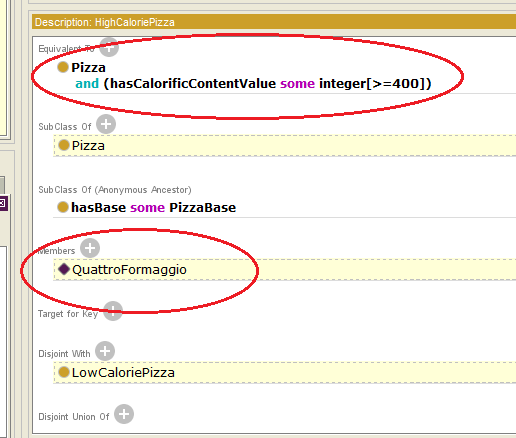


Рисунок 5.5а Класс ограничен по диапазону значений

Наконец, подумайте о том, как много различных значений калорийности могут быть связаны с отдельными пиццами. Конечно, ответ только одно значение. Помните, что это свойство характеризуется тем, что индивид по нему может вступать в отношение только единожды. Описывая свойства объекта как функциональные, мы заявляем, что любой индивид может быть связан не более чем с одним индивидом по этому свойству. Мы также можем использовать функциональную характеристику на свойства DataType (В настоящее время это единственная характеристика, которую можно использовать у свойств данных) Делая hasCalorificContentValue функциональным мы объявим, что любая пицца можно иметь только одно значение энергетической ценности.

**Упражнение 51: Объявление свойства DataType hasCalorificContentValue функциональным**

|  |  |
| --- | --- |
|  | 1. Перейдите на закладку «Data Properties» и выберите пункт hasCalorificContentValue |
|  | 2. На панели «Характеристики» отметьте галочкой характеристику «Functional». |
|  | 3. Тест, что это работает, при создании для экземпляра пиццы двух индивидуальных значений калорий. Это должно привести к онтологии стать несовместимыми. |

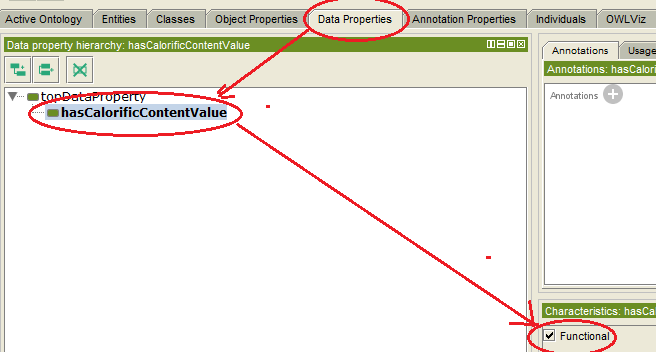


Рисунок 5.5б. Объявление свойства DataType функциональным

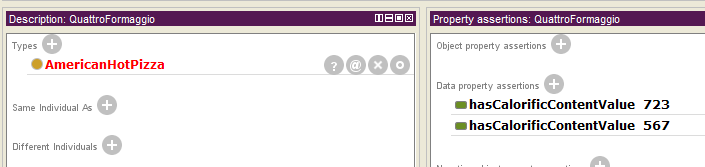


Рисунок 5.5в Тест функционального свойства: класс объявлен несовместимым

|  |  |
| --- | --- |
|  | Существует несколько способов для моделирования классификации по численным значениям в OWL, но мы для понимания решили использовать простой пример. В этом примере классы являются неявными по ограничению свойства (на количество калорий) и используются повсеместно в нашей онтологии. |

# Глава 6

**Подробнее о рассуждении об открытости мира**

В этой главе примеры нюансов рассуждений об открытости мира.

Мы создадим NonVegetarianPizza для дополнения наших категорий пиццы в VegetarianPizzas. Класс NonVegetarianPizza должен содержать все пиццы, которые не являются VegetarianPizzas. Для этого мы создадим класс, который является дополнением VegetarianPizza. Класс-дополнение содержит всех индивидов, которые не содержатся в классе, к которому он является дополнением. Таким образом, если мы создаем NonVegetarianPizza, как подкласс Pizza и делаем его дополнением класса VegetarianPizza, то он должен содержать все пиццы, которые не являются членами VegetarianPizza.

**Упражнение 52: Создание NonVegetarianPizza, как подкласса Pizza и объявление его непересекающимся с VegetarianPizza.**

|  |  |
| --- | --- |
|  | 1. Выберите Pizza в иерархии классов на вкладке «Classes» и нажмите кнопку 'Добавить подкласс' для создания нового класса, как подкласса Pizza. |
|  | 2. Имя нового класса NonVegetarianPizza. |
|  | 3. Объявите NonVegetarianPizza несвязанным с VegetarianPizza — для этого выберите NonVegetarianPizza и нажмите кнопку «+» в строке Disjoint with на панели описания классов. Укажите VegetarianPizza. |

Теперь мы хотим определить NonVegetarianPizza для всех пицц, которые не являются VegetarianPizza.

**Упражнение 53: Сделать VegetarianPizza дополнением к VegetarianPizza**

|  |  |
| --- | --- |
|  | 1. Убедитесь, что в иерархии классов на закладке классы выбрано NonVegetarianPizza. |
|  | 2. Выберите “edit” в строке Pizza в разделе «SubClass Of» на панели описания класса и введите «Pizza and not VegetarianPizza» в закладе «class expression editor» |
|  | 3. Нажмите кнопку OK для создания и назначения выражения. Если всё было введено правильно, то затем редактор выражений закроется, и будет создано выражение. (Если есть ошибки, проверьте правописание «VegetarianPizza»). |

|  |  |
| --- | --- |
|  | Очень полезной функцией редактора выражений является способность «автозаполнения» имен классов, имен свойств и отдельных названий. Автозаполнение для встроенного редактора выражений активируется с помощью клавиши Тab. В приведенном выше примере, если мы введем «Vege» в редакторе и нажмем клавишу Тab, то редактор предоставит выбор завершить слово Vege списком слов, начинающихся с Vege, как показано на рисунке 6.1. Клавишами «стрелки» вверх и вниз можно выбрать VegetarianPizza и нажать клавишу Enter для завершения слова за нас. |

Представление описания классов теперь должно напоминать рисунок 6.2. Однако нам нужно добавить пиццу в необходимые и достаточные условия, так как на данный момент наше определение NonVegetarianPizza говорит, что индивид, который не является членом класса VegetarianPizza (т.е. все остальное!) является NonVegetarianPizza.

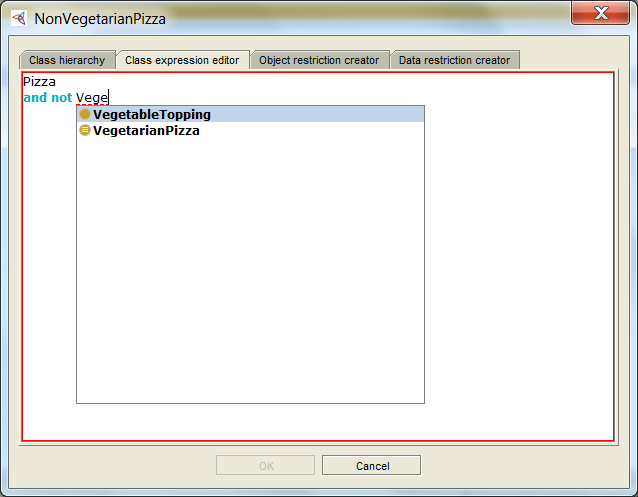


Рисунок 6.1. Автозаполнение в редакторе выражений

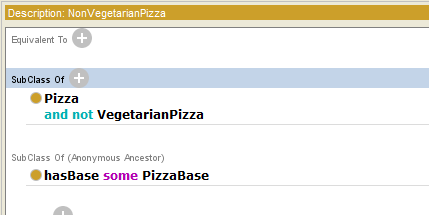


Рисунок 6.2: Панель описания класса: вид primitive-ограничения NonVegetarianPizza

**Упражнение 54: Добавьте пиццы в необходимые и достаточные условия для NonVegetarianPizza**

|  |  |
| --- | --- |
|  | 1. Убедитесь, что выбран NonVegetarianPizza в иерархии классов на вкладке 'Clases'. |
|  | 2. Преобразуйте SubClass Of в эквивалентный класс. Для этого либо выберите закладку «Edit» в головном меню и в ней функцию преобразовать в defenied- класс, или просто нажмите Ctrl+d |

Панель описания классов должна выглядеть, как показано на рисунке 6.3.

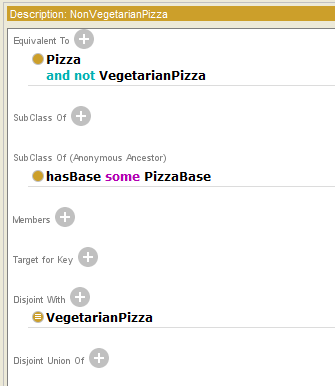


Рисунок 6.3: Панель описания классов: Отображение описания NonVegetarianPizza

|  |  |
| --- | --- |
|  | В состав класса входят все индивиды, которые не являются членами класса. Создавая NonVegetarianPizza – подкласса класса Pizza в качестве дополнения к классу VegetarianPizza мы заявили, что индивиды, которые являются пиццами, и которые не являются членами VegetarianPizza, должны быть членами класса NonVegetarianPizza. Обратите внимание, что мы также сделали классы VegetarianPizza и NonVegetarianPizza непересекающимися так, что если индивид является членом VegetarianPizza, то он не может быть членом NonVege-tarianPizza. |

**Упражнение 55: Используйте реазонер для классификации онтологии**

|  |  |
| --- | --- |
|  | 1. Нажмите кнопку 'synchronize reasoner’ в панели инструментов Reasoner. Через некоторое время реазонер вычислит выводимую иерархию классов, и панель иерархий класса inferred будет открыта сначала. |

Выводимая иерархия классов должна напоминать рисунок 6.4. Как можно видеть, Margher-itaPizza и SohoPizza были классифицированы как подклассы VegetarianPizza. AmericanaPizza и AmericanHotPizza были классифицированы как NonVegetarianPizza. Ограничение, как представляется, работает. Однако давайте добавим пиццу, которая не содержит аксиому закрытия на свойстве hasTopping.

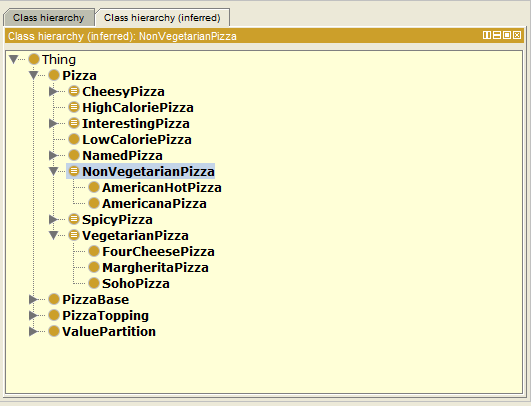


Рисунок 6.4: Выведенная иерархия класса: выведены подклассы VegetarianPizza и NonVegetarianPizza

**Упражнение 56: Создайте подкласс NamedPizza с начинкой MozzarellaTopping**

|  |  |
| --- | --- |
|  | 1.Создание подкласса NamedPizza под названием UnclosedPizza. |
|  | 2. Убедившись, что UnclosedPizza выбран, на панели описания класса выберите заголовок «SubClass Of». |
|  | 3. Нажмите кнопку «+» для отображения редактора ограничений. |
|  | 4. Введите hasTopping в качестве свойства, используемого для ограничения. |
|  | 5. Введите «some» в целях создания экзистенциального ограничения. |
|  | 6. Введите MozzarellaTopping в текстовое поле для указания, что начинки должны быть из класса MozzarellaTopping. |
|  | 7.Нажмите Enter для создания ограничения |

|  |  |
| --- | --- |
|  | Если индивид является членом UnclosedPizza, это значит, что он должен быть членом класса NamedPizza и иметь по крайней мере одно hasTopping отношение к индивиду, который является членом класса MozzarellaTopping. Помните, что действует предположение об открытости мира и тот факт, что мы не добавили аксиому закрытия на свойство hasTopping, следовательно, UnclosedPizza может иметь дополнительные начинки, которые не являются видами MozzarellaTopping. |

**Упражнение 57: Используйте реазонер для классификации онтологии**

|  |  |
| --- | --- |
|  | 1. Нажмите кнопку 'synchronize reasoner’ в панели инструментов Reasoner. Через некоторое время реазонер вычислит выводимую иерархию классов, и панель иерархий класса inferred будет открыта сначала. |

|  |  |
| --- | --- |
|  | Как ожидалось (из-за открытого мира рассуждения) UnclosedPizza не классифицируется как VegetarianPizza. Реазонер не может определить, что UnclosedPizza относится VegetarianPizza, потому что нет аксиомы закрытия на hasTopping, и пицца может иметь другие начинки. Мы поэтому могли рассчитывать, что UnclosedPizza классифицируется как NonVegetarianPizza, так как она не классифицируется, как VegetarianPizza. Однако, предположение об открытости мира не позволяет делать такой вывод потому, что UnclosedPizza не может быть определена ни как VegetarianPizza ни как NonVegetarianPizza - она может быть VegetarianPizza и также она может не быть VegetarianPizza (☺- при отсутствии аксиомы закрытия на hasTopping отношения, предполагается, что возможны и другие hasTopping отношения нашего индивида, которые оставляют вопрос открытым). Таким образом UnclosePizza не может быть классифицирована и как NonVegetarianPizza. |

# Глава 7

**Создание других OWL конструкций в Protege 4**

В этой главе обсуждается создание других OWL-конструкций с использованием **Protege 4.2**. Эти конструкции не являются частью основного учебника и могут быть созданы при необходимости в новом проекте **Protege 4.2**.

## Создание индивидов

OWL позволяет нам определить индивидов и задавать им свойства. Индивиды могут также использоваться в описаниях класса, а именно в hasValue ограничении и перечислимых классах, как это будет объяснено в разделе 7.2 и разделе 7.3 соответственно. Для создания индивидов в **Protege 4.2** используется закладка individuals.

Предположим, что мы хотели описать страны происхождения различных начинок пиццы. Сначала мы должны добавить различные страны нашей онтологии. Страны, например, «Англия», «Италия», «Америка», обычно рассматриваются как индивиды (было бы неверно иметь класс Англии к примеру, тогда как его индивиды будут считаться, «вещи, которые являются экземплярами Англии»). Для создания этого в нашей онтологии мы создадим класс Country (страны) и затем наполним его индивидами:

**Упражнение 58: Создайте класс с именем Country и его наполнение некоторыми индивидами**

|  |  |
| --- | --- |
|  | 1. Создайте Country как подкласс Thing. |
|  | 2. Выберите закладку 'Individuals ' показанную на рисунке 7.1. |
|  | 3. Выберите Country в иерархии классов |
|  | 4. Нажмите на кнопку «добавить индивид», показанную на рисунке 7.2. (Помните, что «индивид» другое имя для экземпляра в терминологии онтологии). |
|  | 5. Имя нового индивида Italy. |
|  | 6. Используйте описанные выше шаги для создания экземпляров индивидов, которые являются членами класса Country под названием America, England, France, и Germany. |

Вспомните из раздела 3.1.1, что OWL не использует уникальное имя предположения (УНА). Отдельные индивиды могут ссылаться на других таким образом, чтобы быть «тем же» или «отличным от». В Protégé 4 эти утверждения можно сделать с помощью «SameAs» и «DifferentFrom» характеристик.

Создав некоторые экземпляры, мы можем использовать их в описании класса как это описано в п. 7.2 и п. 7.3.

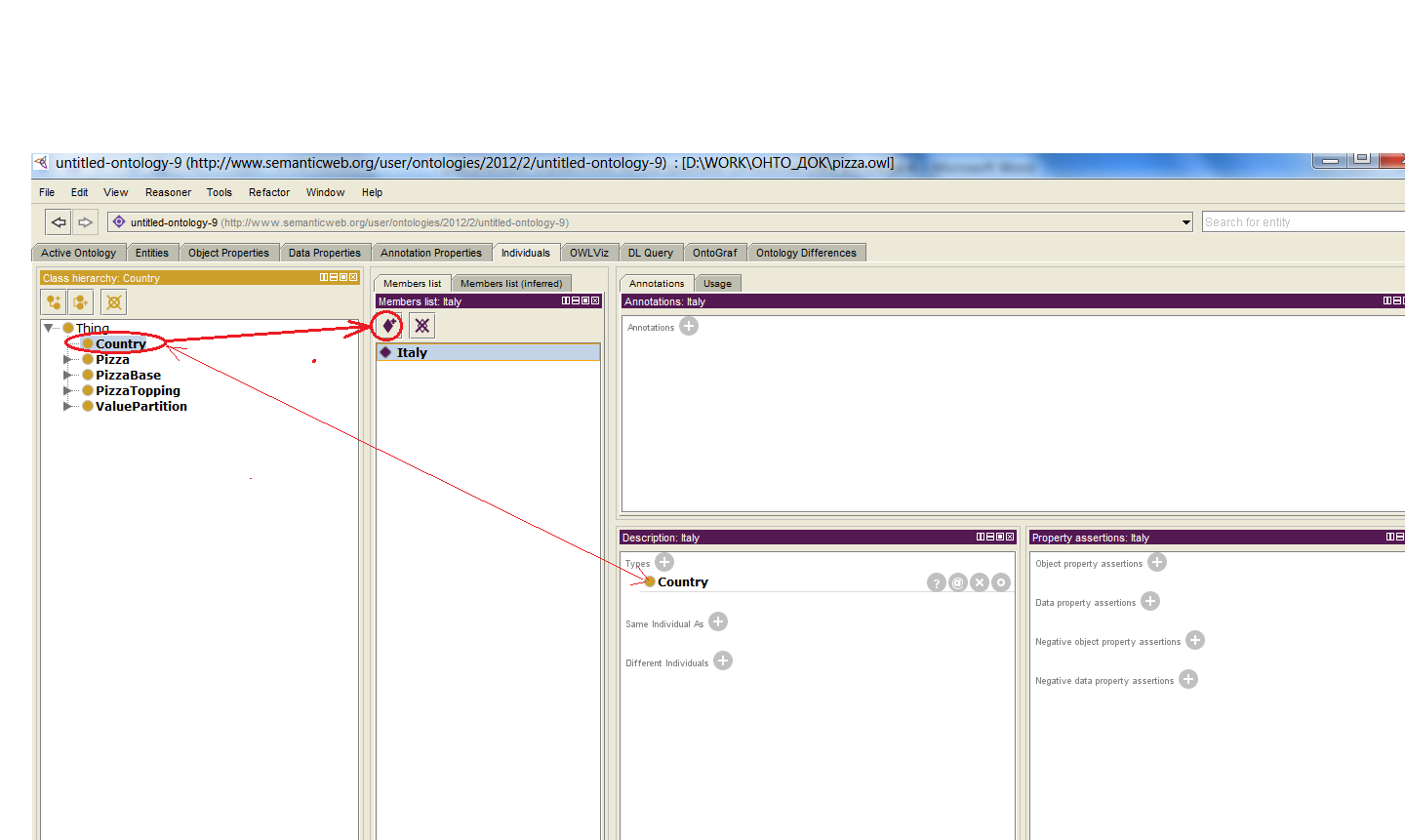


Рисунок 7.1: Закладка индивиды

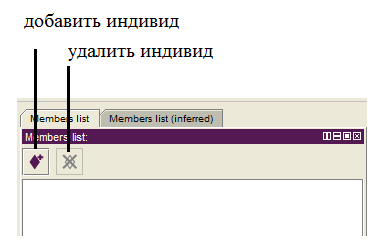


Рисунок 7.2: Экземпляры: назначение кнопок

## Ограничение hasValue

Ограничение по свойству hasValue, обозначается символом **϶**. Оно описывает набор отдельных лиц, которые имеют по крайней мере одно отношение по указанному свойству для конкретного лица. Пример hasValue ограничения: hasCountryOfOrigin **϶** Italy (где Italy это индивид) - описывает набор индивидов (анонимный класс индивидов), которые имеют по крайней мере одно отношение по свойству hasCountryOfOrigin к конкретному индивиду Italy. Дополнительные сведения о hasValue ограничении можно найти приложение A.2.

Предположим, что мы хотим указать происхождение ингредиентов в нашей онтологии пиццы. Например, мы могли бы сказать, что сыр моцарелла (MozzarellaTopping) из Италии. У нас уже есть класс Country в нашей онтологии пиццы, в которой представлены некоторые страны в виде отдельных индивидов, включая Италию. Мы можем использовать ограничение hasValue для указания страны происхождения начинок MozzarellaTopping – в частности Италии.

**Упражнение 59: Создание hasValue ограничения для указания, что MozzarellaTopping имеет Италию, как страну ее происхождения.**

|  |  |
| --- | --- |
|  | 1. Перейдите на вкладку «Свойства объекта» - Создание нового свойства объекта и назовите его hasCountryOfOrigin. |
|  | 2. Перейдите на закладку «Классов» и выберите класс MozzarellaTopping. |
|  | 3. Нажмите на кнопку «+» в строке «SubClass Of» панели описания класса, чтобы открыть редактор ограничений «class expression editor». |
|  | 4. Выберите по автозаполнению hasCountryOfOrigin свойство, которое должно быть ограничено |
|  | 5. Введите value, как тип ограничения, которое необходимо создать. |
|  | 6. Введите Italy как индивид для завершения ограничения. Можно либо ввести это или перетащить из окна индивидов при автозаполнении. |

Панель описания классов теперь должна выглядеть, как показано на рисунке 7.3.



Рисунок 7.3: Описание класса: отображение ограничения hasValue для MozzarellaTopping

|  |  |
| --- | --- |
|  | Условия, которые мы определили для MozzarellaTopping, теперь говорят, что: индивиды, которые являются членами класса MozzarellaTopping также являются членами класса CheeseTopping и связаны с индивидом Italy через свойство hasCountryOfOrigin и связаны по крайней мере с одним из членов класса Mild через свойство hasSpiciness. В более естественном изложении: индивиды, которые являются видом MozzarellaTopping и видом CheeseTopping и из Италии, и из Mild. |
|  | Текущая выводимая классификация не является полной для индивидов. Используйте индивидов в описании класса с осторожностью — неожиданные результаты могут быть выданы реазонером. |

## Перечислимые классы

Кроме описания именованных классов через суперклассы и анонимных через ограничения, OWL позволяет определять классы списком индивидов, которые являются членами класса. Например, мы могли бы определить класс DaysOfTheWeek, содержащий индивидов (и только индивидов): воскресенье, понедельник, вторник, среда, четверг, пятницу и субботу. Такие классы известны как перечислимые классы.

В Protégé 4 перечислимые классы определяются с помощью редактора выражений «class expression editor» - индивидов, которые составляют перечислимый класс, задаются списком имен индивидов (через запятую и разделенных пробелами) внутри фигурных скобок. Например, {Воскресенье, Понедельник, Вторник, Среда, Четверг, Пятница, Суббота}. Индивиды сначала должны быть созданы в онтологии. Перечислимые классы, описанные таким образом, являются анонимными классами - это класс индивидов (и только индивидов), перечисленных в списке. Эти индивиды можно присоединить к именованному классу в Protégé 4, создавая перечисление, как условие в «Эквивалентных классах».

**Упражнение 60: Преобразуйте класс Country в перечислимый класс**

|  |  |
| --- | --- |
|  | 1. Перейдите на вкладку «Classes» и выберите класс Country. |
|  | 2. Выберите заголовок «Эквивалентные классы» на панели описания класса. |
|  | 3. Нажмите кнопку «+», и выберите вкладку «class expression editor» - появится текстовое поле. |
|  | 4. В текстовом поле введите {America, England, France, Germany, Italy}. (Не забудьте окружить список фигурными скобками). Помните, что доступна функция автозаполнения типов. Набирая первые несколько букв индивидуума, нажмите клавишу tab для того, чтобы получить список возможных вариантов. |
|  | 5.Нажмите клавишу enter, чтобы принять перечисление и закрыть редактор выражений. |

Панель описания классов теперь должна выглядеть подобно рисунку 7.4.



Рисунок 7.4: Панель описания классов: отображение перечислимого класса

|  |  |
| --- | --- |
|  | Это означает, что индивид, который является членом класса Country, должен быть одним из включенных в перечень индивидов (то есть быть либо Америкой, либо Англией, либо Францией, либо Германией, либо Италией). Более формально, класс Country эквивалентен (содержит те же индивиды, как), определенный в перечислении анонимный класс — это показано на рисунке 7.5. Это явно не полный перечень стран, но для целей данной онтологии (и этого примера!) он отвечает нашим потребностям. |

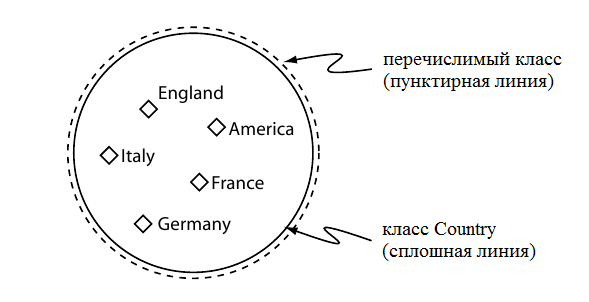


Рисунок 7.5: Принципиальная схема класса Country, эквивалентного перечислимому классу

## Свойства аннотации

OWL позволяет аннотации с различной информацией и мета-данными классов, свойств, индивидов и онтологии (технически говоря: онтологии заголовка). Эти части информации могут принимать форму аудита или редакционной информации. Например, комментарии, дату создания, автора, или ссылки на ресурсы, такие как веб-страницы и др. OWL Full не ставит каких-либо ограничений на использование свойства аннотации. Однако, OWL DL накладывает ряд ограничений на использование свойства аннотации — двумя из наиболее важных ограничений являются:

* Наполнителем для свойства аннотации должны быть символьные данные, ссылки URI или индивидуумы.
* В свойствах аксиом нельзя использовать свойства аннотации, например они не могут использоваться в иерархии свойств, поэтому они не могут иметь подсвойства, или быть подсвойством другого свойства. Также для них не должны устанавливаться домен и диапазон.

Литерные данные являются строкой символов, DataType – значением: например, «Мэтью», 25, 3.11.

OWL имеет пять предопределенных свойств аннотации, которые могут быть использованы для аннотирования классов (включая анонимные классы, такие как ограничения), свойств и индивидов:

OWL: versionInfo — в целом диапазоном этого свойства является строка.

RDFS:Label — имеет диапазон строки. Это свойство может использоваться для добавления пояснений человеку при чтении в онтологии имен таких элементов, как классы, свойства и индивиды. RDFS:Label может также использоваться для предоставления многоязычных имен для элементов онтологии.

RDFS:comment — имеет диапазон строки.

RDFS:seeAlso — имеет значение URI, которое может использоваться для обозначения соответствующих ресурсов.

RDFS:isDefinedBy — имеет значение URI-ссылка, которую можно использовать для ссылки на онтологию, определяющую онтологию элементов, таких как классы, свойства и физические лица.

К примеру, свойство аннотации rdfs:comment используется для хранения комментариев для классов в Protégé 4. Свойства аннотации Rdfs:label могут быть использованы для предоставления альтернативных имен для классов, свойств и др.

Есть также несколько свойств аннотации, которые могут быть использованы для аннотирования онтологии. Свойства аннотации онтологии, перечисленные ниже, имеют значение URI-ссылка, которая используется для обозначения другой онтологии. Это свойство аннотации OWL: VersionInfo можно также использовать для аннотирования онтологии.

OWL: priorVersion — определяет предыдущие версии онтологии.

OWL: backwardsCompatibleWith — определяет предыдущую версию онтологии, совместимую с текущей онтологией. Это означает, что все идентификаторы из предыдущей версии имеют такое же значение в текущей версии. Таким образом в любой онтологии или приложении, которые ссылаются на предыдущую версию можно безопасно переключить ссылки на новую версию.

OWL: incompatibleWith — определяет предыдущую версию онтологии не совместимой c текущей онтологией.

Для создания свойства используют вкладки «Классы», «Свойства объекта» и «Свойства Datatype», соответствующие описаниям свойств аннотации в каждой из активных онтологий. Вы можете управлять вашей аннотацией, используя вкладку «Свойства аннотации», на которй могут быть созданы новые свойства аннотацииэ Для этого нужно нажать на кнопку «создать аннотацию свойства» на вкладке «Аннотация свойства». Чтобы создавать описания свойств аннотации используется панель, представленная на рисунке 7.6. Представление аннотации расположено на всех вкладках: классы, свойства, индивиды и активные онтологии, для создания аннотации классов, свойств, индивидов и онтологии соответственно. Можно также добавить аннотации к ограничениям и другим анонимным классам, выбрав команду «@»-«Изменить свойства аннотации...» на панели описания классов в строке выбранного ограничения.

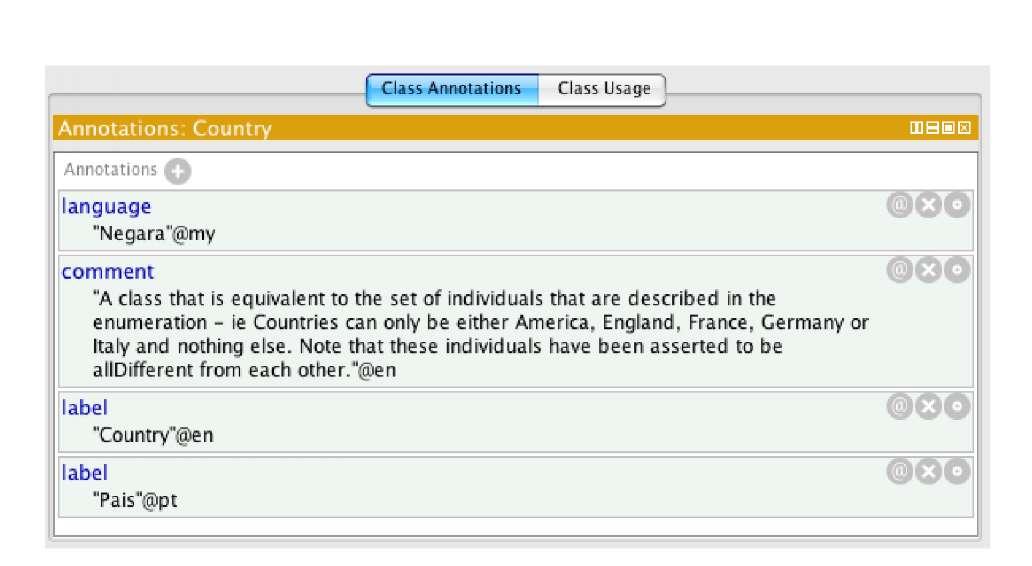


Рисунок 7.6: Представление аннотации

## Несколько наборов необходимых и достаточных условий (Multiple Sets Of Necessary & Sufficient Conditions)

В OWL можно иметь несколько наборов необходимых и достаточных условий (эквивалентных классов), как показано на рисунке 7.7. На панели описания класса представлен набор из нескольких необходимых и достаточных условий в виде заголовков в «Эквивалентных классах», где необходимые и достаточные условия перечисленные в каждом заголовке, как показано на рисунке 7.8. Для создания нового набора необходимых и достаточных условий, используйте значок «Add» (+) рядом с заголовком «Эквивалентные классы». Эквивалентные классы могут быть написаны в редакторе class expression editor.

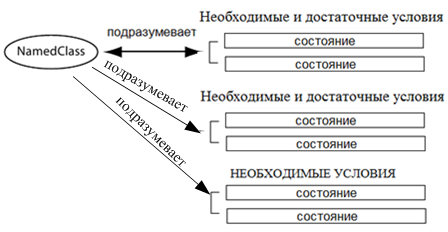


Рисунок 7.7: Необходимые условия и несколько наборов необходимых и достаточных условий

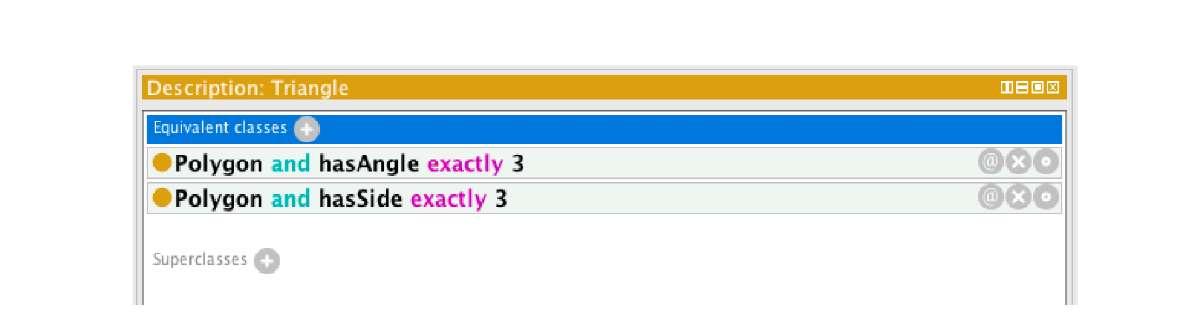


Рисунок 7.8: Описание треугольника с помощью нескольких необходимых и достаточных условий

# Глава 8

**Выполнение SPARQL-запросов средствами Protege 4**

В этой главе рассмотрено несколько примеров по выполнению SPARQL-запросов встроенными средствами **Protege 4.2**.

**1. Выполнение запроса “по умолчанию”**

В случае если закладка “SPARQL Query” не отображается, выполните Window/Tabs/SPARQL Query, как показано на рис. 8.1.

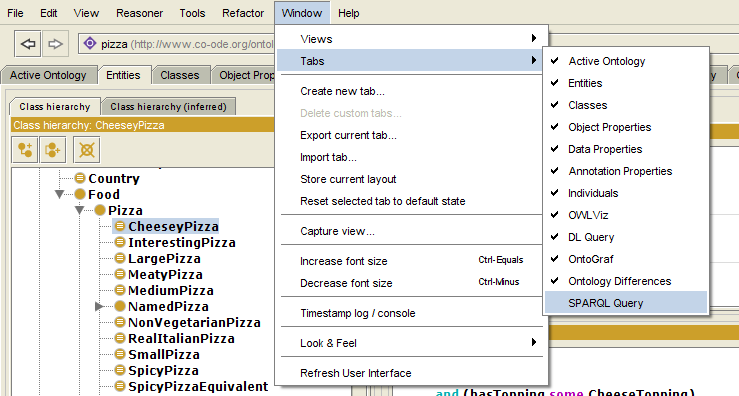


Рис. 8.1. Подключение закладки “SPARQL Query”

Перейдите на закладку «SPARQL Query». На закладке представлен образец запроса по умолчанию:

SELECT ?subject ?object

WHERE { ?subject rdfs:subClassOf ?object }

Нажмите кнопку “Execute” для выполнения запроса. Результат запроса показан на рис. 8.2.

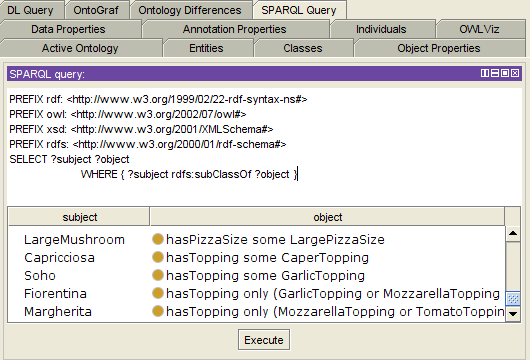


Рис 2. Результаты запроса по умолчанию

В результате выполнения запроса переменной ?subject в соответствие будут установлены классы, для которых характерно наличие свойства subClassOf. То есть классы, которые являются потомками для других классов (например класс, определяющий пиццу MediumSoho, является подклассом по отношению к классу, определяющему все пиццы Soho (см. рис. 8.3)) и классов, которые являются подклассами ограничений (см. рис. 8.4).

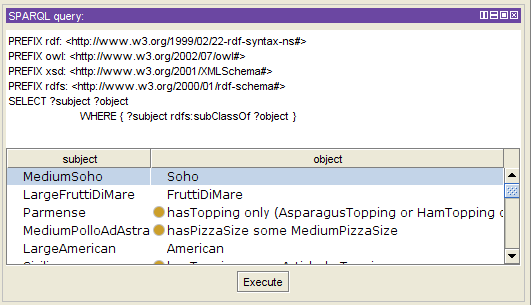


Рис. 8.3. Образцы потомков класса пицц Soho

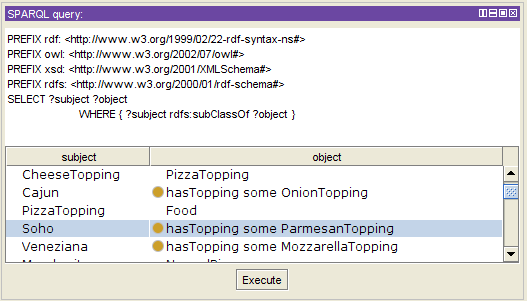


Рис. 8.4. Образцы потомков для ограничения классов пицц, содержащих слой с пармезаном

**2. Поиск подклассов Pizza с применением данных о литералах**

В предыдущем запросе были возвращены все классы, которые являются потомками любых других классов и ограничений. Выполним запрос, который вернет только те классы, которые являются подклассами Pizza:

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

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

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

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

PREFIX : <http://www.co-ode.org/ontologies/pizza/pizza.owl#>

SELECT ?subject ?class

WHERE { ?subject rdfs:subClassOf ?class.

?class rdfs:label "Pizza"@en}

Обратите внимание на выделенный фрагмент кода:

PREFIX : <http://www.co-ode.org/ontologies/pizza/pizza.owl#>,

который необходим для выполнения специализированных запросов к онтологии пиццы.

В результате выполнения запроса будут возвращены потомки класса, для которого установлена метка "Pizza"@en.

Результаты выполнения запроса представлены на рис. 8.5:

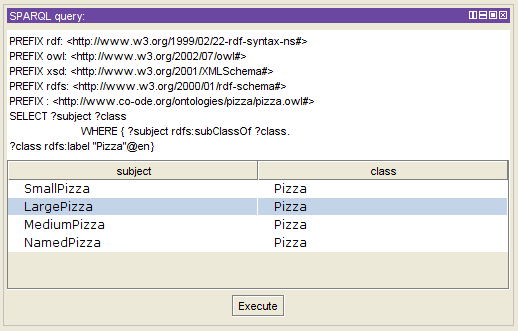


Рис. 8.5. Потомки класса Pizza

**3. Поиск данных о пиццах, представленных в ресторанных меню, с применением данных о литералах**

Выполним запрос для получения перечня “именованных” пицц, так как именно эта информация представлена в ресторанных меню.

Попытаемся выполнить запрос по аналогии с предыдущем, заменив метку «Pizza» на «NamedPizza». Результат выполнения запроса представлен на рис. 8.6:

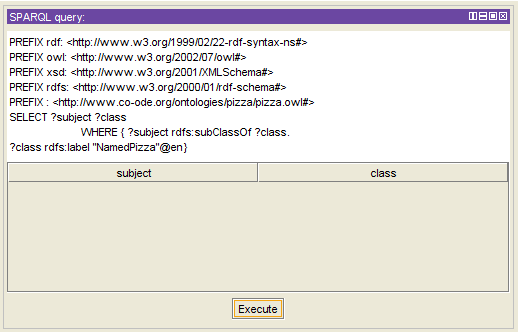


Рис. 8.6. Результат поиска именованных пицц по метке “NamedPizza"@en

С целью исключения опечатки в информации о метке “NamedPizza"@en выполним анализ свойств класса NamedPizza (см. 8.7).

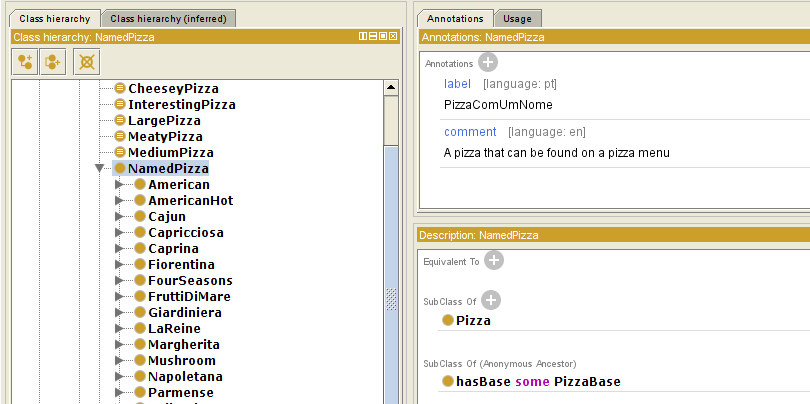


Рис. 8.7. Свойства класса NamedPizza

Согласно рис. 8.7 для класса NamedPizza не установлена метка “NamedPizza"@en. Для корректировки данной ситуации возможны 2 варианта:

1. Добавить метку “NamedPizza"@en.
2. Использовать имеющиеся данные.

Выполним запрос, используя информацию о комментарии (см. рис. 8.8):

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

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

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

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

PREFIX : <http://www.co-ode.org/ontologies/pizza/pizza.owl#>

SELECT ?subject ?class

WHERE { ?subject rdfs:subClassOf ?class.

?class rdfs:comment "A pizza that can be found on a pizza menu"@en}

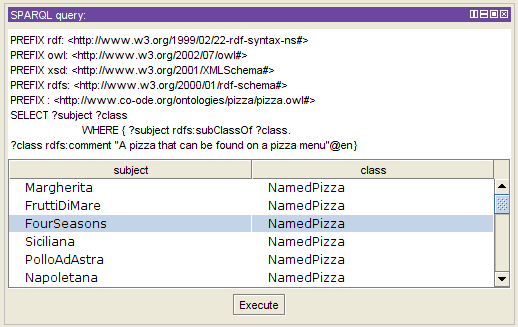


Рис. 8.8. Потомки класса NamedPizza

Выполните самостоятельно запрос, используя информацию о метке на португальском языке.

**4. Запрос на основе данных об экземплярах классов**

Выполним запрос, который вернет информацию о традиционно итальянских видах пицц, результат выполнения запроса представлен на рис. 8.9:

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

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

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

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

PREFIX : <http://www.co-ode.org/ontologies/pizza/pizza.owl#>

SELECT ?food

WHERE { ?food rdfs:subClassOf ?fromItaly.

?fromItaly owl:onProperty :hasCountryOfOrigin.

?fromItaly owl:hasValue :Italy.

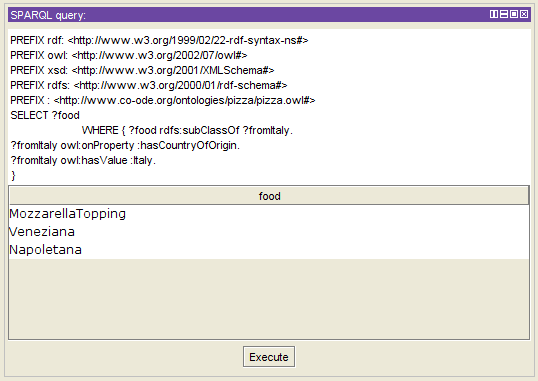


Рис. 8.9. Традиционные пиццы Италии

**5. Запрос на основе данных об объектных свойствах классов**

Пример запроса, возвращающего виды пицц с моцареллой, представлен по адресу:<http://protegewiki.stanford.edu/wiki/File:P4_sparql_query.png.>

Результат выполнения запроса представлен на рис. 8.10.

Текст запроса:

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

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

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

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

PREFIX : <http://www.co-ode.org/ontologies/pizza/pizza.owl#>

SELECT ?withMozarella

WHERE { ?withMozarella rdfs:subClassOf ?object .

?object owl:onProperty :hasTopping.

?object owl:someValuesFrom :MozzarellaTopping.

}

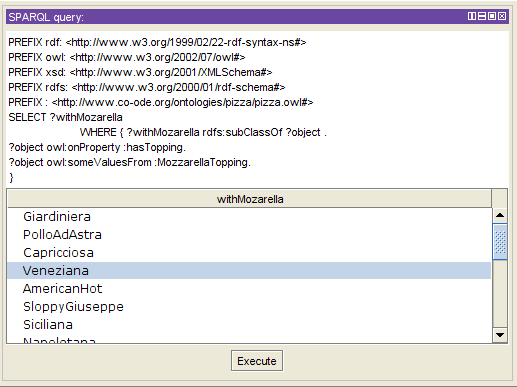


Рис. 8.10. Пиццы с моцареллой

Выполните самостоятельно запрос, который вернет название пиццы с красным луком (RedOnionTopping).