Онтологии

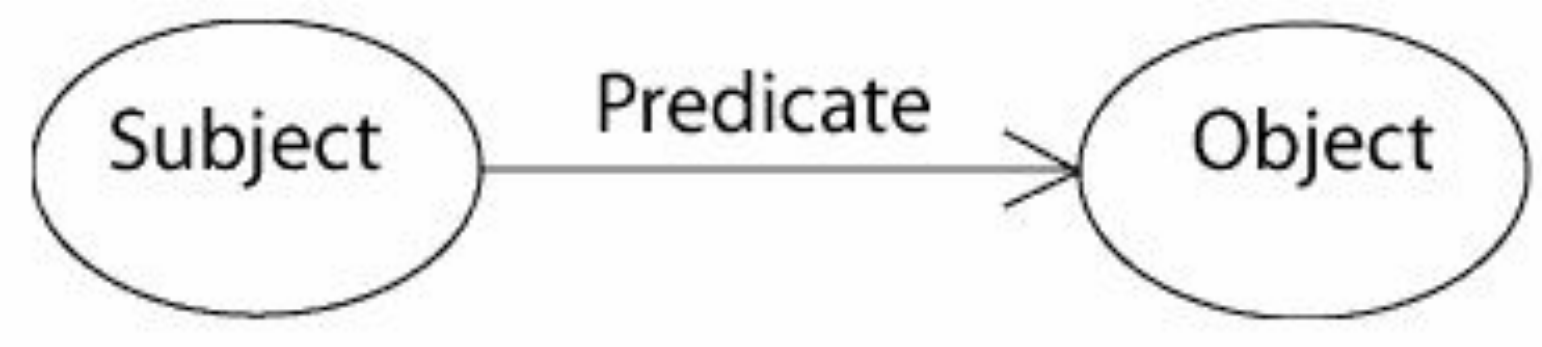
RDF

- RDF - язык представления информации о ресурсах WWW. В частности, RDF служит для представления метаданных, связанных с ресурсами Сети, таких как "заголовок", "автор", "дата последнего изменения страницы". Но RDF может использоваться и для представления информации о ресурсах "второго типа", на которые можно только ссылаться (или идентифицировать в Сети при помощи URI), но невозможно непосредственно получить к ним доступ через Сеть.

- Может оказаться, что в некоторых случаях для управления метаданными достаточно использовать XML и XML Schema (либо вообще ограничиться подэлементом HEAD элемента HTML). Но этот подход слабо масштабируется: при увеличении объема метаданных, усложнении их структуры управление метаданными, построенными на основе XML Schema, становится трудоемкой задачей, для решения которой и предназначен RDF.

Модель данных RDF. RDF-граф

- Базовой структурной единицей RDF является коллекция троек (или триплетов), каждая из которых состоит из субъекта, предиката и объекта (S,P,O). Набор триплетов называется RDF-графом. В качестве вершин графа выступают субъекты и объекты, в качестве дуг - предикаты (или свойства). Направление дуги, соответствующей предикату в данной тройке (S,P,O), всегда выбирается так, чтобы дуга вела от субъекта к объекту. RDF



- Каждая тройка представляет некоторое высказывание, увязывающее S, P и O.
- Первые два элемента RDF-тройки (субъект и предикат) идентифицируются при помощи URI. Объектом же может быть как ресурс, идентифицируемый при помощи URI, так и RDF-литерал (значение).

Архитектура метаданных в World Wide Web
Документы, метаданные, связи

- Когда вы переходите по ссылке URL, то получаете нечто. Мы будем называть это нечто ресурсом Сети. Часто под ресурсом понимается документ, поскольку в Сети много читабельных (удобных для чтения человеком) документов - HTML-страниц, PDF-документов и т.п. Иногда ресурс - это просто некий объект, когда полученный ресурс имеет машинопонятный вид или обладает скрытым внутренним состоянием.
- термины "ресурс", "объект" и "документ" являются синонимами.
- Неотъемлемой характеристикой любого ресурса Сети является сопровождающая его информация. Эту "сверхинформацию", или информацию об информации (о ресурсе), принято называть метаданными.
- Под метаданными будем понимать машинопонятную информацию о веб-ресурсах и других сущностях.
- Термин "машинопонятная" является ключевым. Речь идет о понимании информации программными агентами. Причем "понимании" с одной целью - использовать информацию для решения задач, возложенных на них (агентов) пользователем.
- Метаданные должны иметь хорошо определенную ясную структуру и семантику.

- Пример 1. Метаданные.

- Объект, извлеченный из сети по протоколу HTTP, может иметь дополнительную информацию (метаданные):

- дата создания или дата прекращения действия;
- владелец;
- другая информация.

- Таким образом, в Сети есть данные - ресурсы, есть метаданные - информация о ресурсах. Эта информация, в свою очередь, тоже может рассматриваться как данные (ресурс).

- Приведем два постулата (A1 и A2), на которых основана архитектура метаданных Сети.

- A1. Метаданные - это данные (другими словами, информация об информации - это тоже информация).

- Поскольку метаданные - это данные, то они могут храниться в ресурсе (могут быть представлены как ресурс). То есть любой ресурс Сети может хранить как данные, так и метаданные о себе или о других ресурсах. На практике в Сети существует 3 способа передачи/получения метаданных:

- метаданные хранятся и передаются внутри документа (тег HEAD в HTML, данные о документе MS Word);
- сообщение метаданных происходит во время HTTP (GET, POST или PUT) передачи;
- метаданные хранятся в каком-то другом документе.

- Итак, метаданные могут храниться внутри самого документа, внутри другого документа либо передаваться вместе с документом средствами протокола HTTP.
- Форма метаданных

- Метаданные состоят из высказываний о данных и при представлении имеют форму имени (или типа высказывания) и набора параметров.

- A2. Архитектура, представляемая метаданными, является набором независимых высказываний (утверждений).

- Как следствие, при группировке двух и более высказываний об одном ресурсе они объединятся логическим " И ". Альтернативные высказывания являются независимыми, а их наборы представляют собой неупорядоченные множества.

- Конечно, высказывания можно комбинировать и другим способом, используя сложные синтаксические правила, но основной формой представления является неупорядоченный список, элементы которого связаны логическим " И ".

- Наиболее распространенной формой высказывания является следующая модель:

- Ресурс - атрибут - значение

- ресурс - это объект, о котором фиксируется высказывание, атрибут - некоторое свойство или параметр объекта, значение представляет некоторое значение из области значений атрибута (или диапазона значений атрибута данного объекта).

- Пример 2. Модель "Ресурс - атрибут - значение".

- E-mail - Date - 01.01.2006
- E-mail - From - Vasya

- В общем виде высказывание может быть выражено так:

- (A u1 p q ...),

- где A - имя (или идентификатор) типа высказывания (такие как Author, Date и т.п.), u1 - URI ресурса, о котором делается высказывание, p, q, \dots - другие параметры, зависящие от типа высказывания, в том числе и представляющие значение атрибута.

- Связи
- Отношение между двумя ресурсами будем называть связью. Связь представляется тройкой (A u1 u2),
- где A - тип отношения, U1 - URI первого ресурса, U2 - URI второго ресурса.
- Связи являются основой навигации в Сети. Они могут использоваться для построения структур внутри WWW, а также для создания семантической Сети, в которой могут быть представлены знания об окружающем мире.
- Одна из основных задач, решаемых при проектировании архитектуры метаданных Сети, состоит в том, чтобы сделать информацию самоописывающейся (self-describing).
- Однако узким местом системы всегда является способ определения семантики метаданных и данных, применяемых внутри системы. Например, семантика метаданных заголовков e-mail и HTTP-сообщений определяется вручную на английском языке в виде спецификаций соответствующих протоколов. Эта семантика понятна людям (конечно, тем, кто знает английский). Чтобы теперь перейти к семантике, понятной машине, нужно использовать подходящий логический язык или язык представления знаний. Тогда семантика (точное значение) некоторого высказывания может быть выражена в терминах других отношений (более абстрактных концептов логического языка).
- Преимущество самоописывающейся информации состоит в том, что нет необходимости согласовывать значение каждого термина централизованно, стандартизировать семантику высказываний. Язык RDF позволяет описывать метаданные о любых ресурсах Сети (и даже о сущностях, находящихся за ее пределами).

RDF-литералы (или символьные константы)

- RDF-литералы бывают двух видов: типизированные и нетипизированные.
- Каждый литерал в RDF-графе содержит одну или две именованные компоненты.
- Все литералы имеют лексическую форму в виде строки символов Unicode.
- Простые литералы состоят из лексической формы и необязательной ссылки на язык (ru, en, :).
- Типизированные литералы состоят из лексической формы и URI-ссылки на тип данных, задаваемой в формате RDF URI.

- Замечание. Язык литерала не нужно путать с идентификатором (языком) локали. Язык относится только к текстам, написанным на естественном языке. Все трудности, возникающие при представлении данных на конкретном компьютере (при определении локали), должны решаться конечным потребителем метаданных.

Сравнение литералов

- Два литерала равны тогда и только тогда, когда выполняются все перечисленные ниже условия.
- Строки обеих лексических форм совпадают посимвольно.
- Либо оба литерала имеют теги языка, либо оба не имеют.
- Теги языка, если они имеются, совпадают.
- Либо оба литерала имеют URI типа данных, либо оба не имеют.
- При наличии URI типа данных эти URI совпадают посимвольно.

Определение значения типизированного литерала

- Рассмотрим следующий пример. Пусть множество {T, F} - множество значений истинности в математической логике. В различных приложениях элементы этого множества могут представляться по-разному. В языках программирования {1, 0} (1 соответствует T, 0 соответствует F), либо {true, false}, либо {истина, ложь}.
- Фактически задается некоторое отображение множества значений истинности на множество чисел или строк символов. Теперь значениями логического типа (bool или boolean) становятся строковые значения или спецсимволы. Чтобы получить значения истинности, необходимо воспользоваться обратным отображением.
- Таким же образом происходит получение значения типизированного RDF-литерала. За лексической формой стоит некоторое значение, которое определяется применением отображения. Это отображение определяется по URI типа данных и зависит от самого типа.

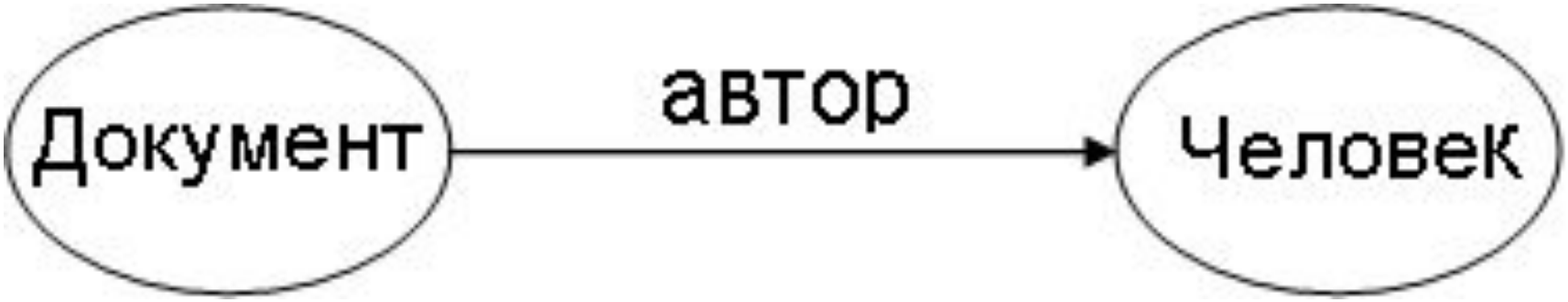
- **Языки представления онтологий: RDFS, OWL. Язык запросов SPARQL**

- Для того чтобы реализовывать различные онтологии, необходимо разработать языки их представления, имеющие достаточную выразительную мощность и позволяющие пользователю избежать "низкоуровневых" проблем. Ключевым моментом в проектировании онтологии является выбор соответствующего языка спецификации онтологий. Цель таких языков - дать возможность указывать дополнительную машинно-интерпретируемую семантику ресурсов, сделать машинное представление данных более похожим на положение вещей в реальном мире, существенно повысить выразительные возможности концептуального моделирования слабо структурированных Web-данных.

- Распространение онтологического подхода к представлению знаний оказало содействие при создании разнообразных языков представления онтологии и инструментальных средств, предназначенных для их редактирования и анализа. Существуют традиционные языки спецификации онтологий: Ontolingua, CycL, языки, основанные на дескриптивных логиках (такие как LOOM), языки, основанные на фреймах (OKBC, OCML, F-Logic). Более поздние языки основаны на Web-стандартах (XOL, SHOE, UPML). Специально для обмена онтологиями через Web были созданы языки RDF, RDFS, DAML+OIL, OWL.

RDFS

- Каждый из элементов триплета определяется ссылкой на тип элемента и URI. Предикат (в контексте RDF его обычно называют свойством) может пониматься либо как атрибут, либо как бинарное отношение между двумя ресурсами. Но RDF сам по себе не предоставляет никаких механизмов ни для описания атрибутов ресурсов, ни для определения отношений между ними. Для этого предназначен язык RDFS (RDF Schema) - язык описания словарей для RDF. RDFS определяет классы, свойства и другие ресурсы.
- RDF-тройка "субъект-предикат-объект"



- RDFS является семантическим расширением RDF. Он предоставляет механизмы для описания групп связанных ресурсов и отношений между этими ресурсами. Все определения RDFS выражены на RDF (поэтому RDF называется "самоописывающимся" языком). Новые термины, вводимые RDFS, такие как "домен", "диапазон" свойства, являются ресурсами RDF.

- Система классов и свойств языка описания RDF-словарей похожа на систему типов объектно-ориентированных языков программирования, например, Java. Но RDF отличается от большинства таких систем тем, что здесь центральным аспектом является определение свойства, а не класса. Свойства в RDF определяются как пары (домен, диапазон). При этом домен представляет некоторое множество классов RDF, к которым данное свойство применимо, диапазон определяет допустимое множество ресурсов - значений свойства. Для сравнения: в Java определение класса имеет законченную форму (свойства класса выражаются в полях и методах класса). В RDF, напротив, описание класса всегда остается открытым (набор свойств класса определяется вне самого класса).

- Пример. Определим свойство "автор" с доменом "Документ" и диапазоном "Человек". В случае появления дополнительной информации о свойствах "Документа" нет необходимости изменять описание класса "Документ". Достаточно добавить новое свойство с соответствующим доменом.
- Пример "a-la RDF":
- Класс ("Документ");
- Класс ("Человек");
- Свойство ("Автор", "Документ", "Человек").
- Пример "a-la Java":
- Класс "Документ"
- {
- "Человек" "Автор"
- }
- Можно заметить, что при изменении смысла свойств изменять придется именно их. При этом все классы, зависящие от изменяемых свойств, косвенно изменят свою семантику.

- Основное преимущество такого подхода - в легкой расширяемости: добавление/удаление свойств интуитивно проще, чем управление множеством классов, обладающих каждый своим индивидуальным набором свойств (как в ООП). Фактически, любой может расширять описание существующих ресурсов (лозунг Web: "Кто угодно может сказать что угодно о чем угодно!").

Классы

- Ресурсы могут объединяться в группы, называемые классами. Члены класса (здесь наиболее близкий термин - "экземпляры" или "объекты" ООП) называются экземплярами класса. Сами классы также являются ресурсами и идентифицируются ссылками RDF-URI. Чтобы указать, что ресурс является экземпляром класса, используется свойство rdf:type ("rdf" здесь применен как префикс пространства имен).
- В RDF определение класса или свойства (т.н. интенсионал) отделено от множества экземпляров класса и значений свойства (т.н. экстенсионала). Так, два класса с одинаковыми экстенсионалами считаются различными, если они имеют разные наборы свойств (интенсионалы).

Экстенсионал и интенсионал

- Рассмотрим множества
- A = {0, 2, 4, 6, 8},
- B = {x | x = 2k, k = 0..4, k - целое},
- C - множество неотрицательных четных чисел, меньших 10.
- В этом примере множество A полностью описывается своим экстенсионалом, множества B и C описываются интенсионалами, т.е. с использованием характеристических свойств данного множества. Множества, имеющие бесконечное число элементов, могут быть описаны только своим интенсионалом. Однако при использовании интенсионала могут возникнуть парадоксы. Чтобы избежать их, в теории множеств вводятся дополнительные аксиомы. Примечательно, что RDF нарушает эти аксиомы. Классу RDF не запрещено быть экземпляром самого себя.

Парадокс Рассела (иногда парадокс Рассела — Цермело)

- открытый в 1901 году[1] Бертраном Расселом и позднее независимо переоткрытый Э. Цермело теоретико-множественный парадокс, демонстрирующий противоречивость логической системы Фреге, являвшейся ранней попыткой формализации наивной теории множеств Г. Кантора.

- Пусть K — множество всех множеств, которые не содержат себя в качестве своего элемента. Содержит ли K само себя в качестве элемента? Если предположить, что содержит, то мы получаем противоречие с "Не содержат себя в качестве своего элемента". Если предположить, что K не содержит себя как элемент, то вновь возникает противоречие, ведь K — множество всех множеств, которые не содержат себя в качестве своего элемента, а значит должно содержать все возможные элементы, включая и себя.

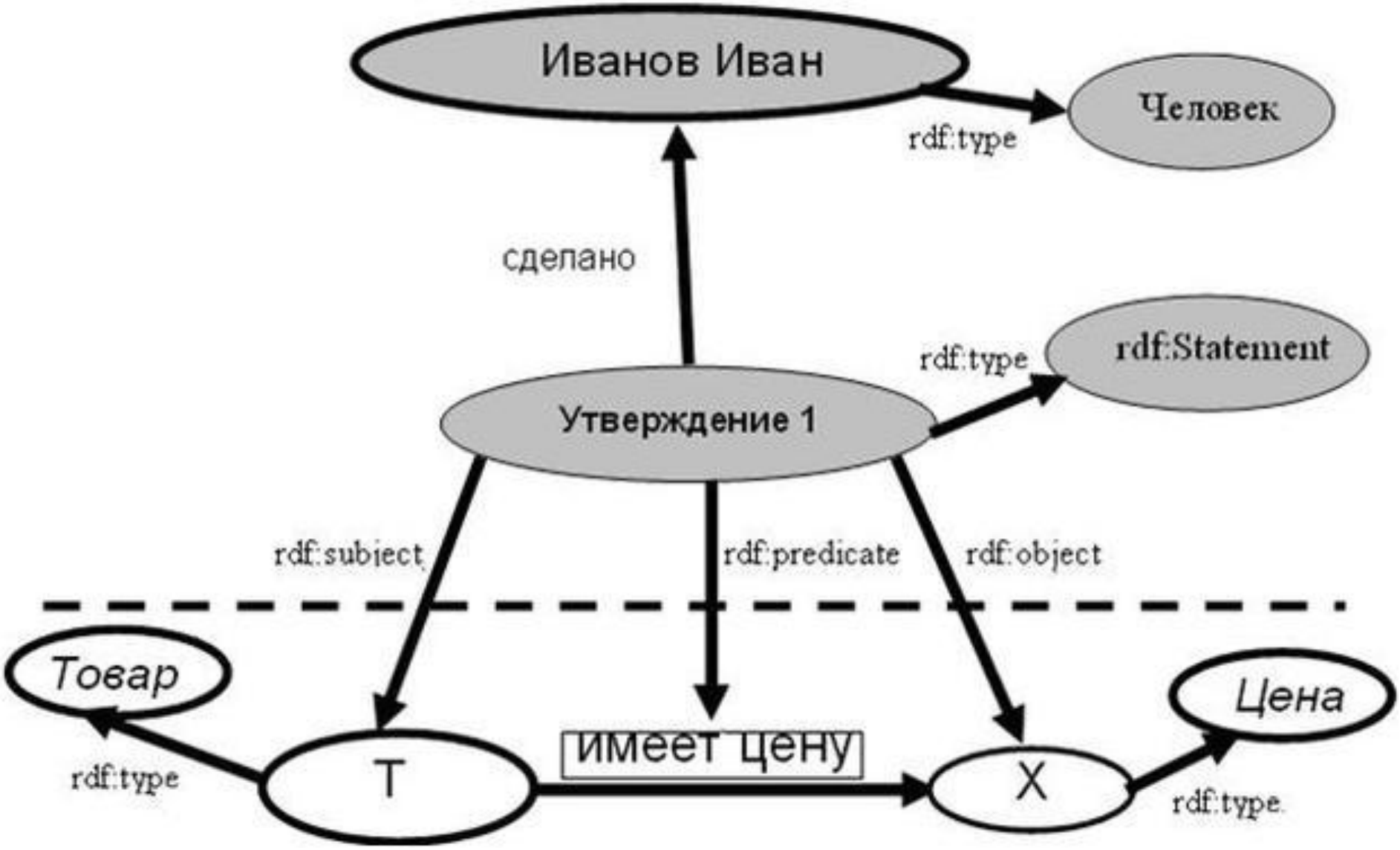
- Группа ресурсов, являющихся классами, в RDFS описывается термином rdfs:Class.
- На множестве классов определено отношение ПОДКЛАСС-НАДКЛАСС, описываемое RDFS-свойством rdfs:subClassOf. Семантика данного отношения состоит в том, что экстенсионал любого подкласса данного класса C целиком содержится (как множество) в экстенсионале самого класса C. Другими словами, если ресурс i является экземпляром класса C*, а класс C* является подклассом класса C, то i является экземпляром класса C.
- Любой класс RDFS по определению является подклассом самого себя.
- В спецификации по RDFS определены также списки, коллекции и контейнеры ресурсов, текстовые пометки и комментарии для создания удобных для чтения примечаний к ресурсам.

Реификация (материализация, овеществление утверждений)

- В случае, когда необходимо сделать утверждение об утверждении RDF, прибегают к так называемой реификации, или материализации утверждений. В этом случае само утверждение (или высказывание) выступает в роли объекта.
- Для этого используется специальный класс rdf:Statement и его свойства rdf:subject, rdf:predicate и rdf:object. Каждое RDF-утверждение является экземпляром класса rdf:Statement. По свойствам (и их значениям) можно однозначно идентифицировать само утверждение. Обладая этой информацией, возможно фиксировать утверждения об утверждениях.

- Пример. В базе данных электронного магазина хранится информация о том, что некий товар (T) имеет цену x. Данное утверждение (1) (товар T имеет цену x) может быть выражено Ивановым Иваном Ивановичем на языке RDF. Если далее потребуется высказать утверждение (2) о том, кто именно сделал утверждение (1), можно использовать механизм реификации

Товар	T	#
rdf:Property	Имеет цену	
Цена	X	#
----------		
rdf:Statement	Утверждение 1	*
rdf:subject	T	*
rdf:predicate	имеет_цену	*
rdf:object	x	*
rdf:Statement	Утверждение 1	+
rdf:Property	сделано	+
Человек	Иванов	+


Иванов Иван
Человек
rdf:type
сделано
rdf:type
rdf:Statement
Утверждение 1
rdf:subject
rdf:predicate
rdf:object
Товар
Цена
имеет цену
rdf:type
T
X
rdf:type

- Отметим один важный момент: фиксация только тех утверждений, которые помечены " * ", не означает, что товар T действительно имеет цену x. Даже вместе с утверждениями, помеченными " + ", вся информация, которую мы узнаём - это: "некто Иванов Иван Иванович сделал утверждение о товаре T, что он имеет цену x ". Но не более того! Значение x цены товара T фиксируется тройкой строк, помеченных " # ".
- Понятно, что новое утверждение (высказывание об Утверждении 1) также может быть подвергнуто реификации, поскольку синтаксически не отличается от Утверждения 1 (оно также является экземпляром класса rdf:Statement).

полный список классов и свойств RDF/RDFS. Классы RDFS

rdfs:Resource	Класс-ресурс, включает "всё"
rdfs:Literal	Класс литеральных значений, текстовых строк или чисел
rdf:XMLLiteral	Класс XML-литералов
rdfs:Class	Класс классов
rdf:Property	Класс RDF-свойств
rdfs:Datatype	Класс типов данных RDF
rdf:Statement	Класс утверждений
rdf:Bag	Класс неупорядоченных контейнеров
rdf:Seq	Класс упорядоченных контейнеров
rdf:Alt	Класс контейнеров-альтернатив
rdfs:Container	Класс RDF-контейнеров
rdfs:ContainerMembership	Класс свойств "членства" в контейнерах: rdf:_1, rdf:_2, ..., все они являются подсвойствами свойства rdfs:member
rdf:List	Класс RDF-списков

Свойства RDFS

Имя свойства	Пояснение	Домен	Диапазон
rdf:type	Субъект является экземпляром класса	rdfs:Resource	rdfs:Class
rdfs:subClassOf	Субъект является подклассом класса	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	Субъект является подсвойством свойства	rdf:Property	rdf:Property
rdfs:domain	Домен свойства субъекта	rdf:Property	rdfs:Class
rdfs:range	Диапазон свойства субъекта	rdf:Property	rdfs:Class
rdfs:label	Человекочитаемое название субъекта	rdfs:Resource	rdfs:Literal

rdfs:comment	Текстовое описание ресурса	rdfs:Resource	rdfs:Literal
rdfs:member	Член ресурса субъекта	rdfs:Resource	rdfs:Resource
rdf:first	Первый элемент списка	rdf:List	rdfs:Resource
rdf:rest	Оставшийся за первым элементом "хвост" списка	rdf:List	rdf:List
rdfs:seeAlso	Дополнительная информация о субъекте	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	Определение ресурса субъекта	rdfs:Resource	rdfs:Resource

rdf:value	Свойство, используемое для структурированных значений	rdfs:Resource	rdfs:Resource
rdf:subject	Субъект RDF-утверждения (см. "реификация")	rdf:Statement	rdfs:Resource
rdf:predicate	Предикат RDF-утверждения (см. "реификация")	rdf:Statement	rdfs:Resource
rdf:object	Объект RDF-утверждения (см. "реификация")	rdf:Statement	rdfs:Resource

- Возможности и ограничения языка RDF (RDF Schema)
- Сам по себе RDF не является стандартом метаданных, как, например, Dublin Core, FOAF, vCard. Все, что он "умеет", - это фиксировать утверждения о ресурсах, их свойствах и значениях этих свойств.
- Важные свойства языка:
- обобщенный способ работы с метаданными;
- ориентация на программное обеспечение в качестве конечного потребителя информации;
- возможность осуществлять автоматическую обработку Web-ресурсов:
- поиск;
- каталогизацию;
- генерацию иерархических карт сайтов.

- Недостатки RDF
- Открытость и расширяемость RDF ведет к тому, что "кто угодно (т.е. любой пользователь RDF) может сказать что угодно (т.е. фиксировать произвольное утверждение) о чем угодно (т.е. о любом ресурсе)", используя RDF. RDF не запрещает делать бессмысленных утверждений или утверждений, не согласующихся с другими. Следовательно, нет никакой гарантии целостности и непротиворечивости RDF-описаний. Вся ответственность за проверку ложится на получателей (конечных пользователей) метаданных, т.е. на разработчиков приложений, обрабатывающих RDF-данные.

Способы представления RDF-описаний

Ниже приводится пример двух способов представления RDF графов: в форме XML-документа (часто более удобной для автоматической обработки) и в форме последовательностей троек - так называемый N Triple или N3 синтаксис (удобный для восприятия человеком).

XML-синтаксис

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:exterms="http://www.example.org/terms/">
<rdf:Description rdf:about="http://www.example.org/index.html">
<exterms:creation-date>August 16, 1999</exterms:creation-date>
</rdf:Description>
<rdf:Description rdf:about="http://www.example.org/index.html">
<dc:language>en</dc:language>
</rdf:Description>
</rdf:RDF>
```

N3-синтаксис (удобный для чтения человеком и расширяющий исходную модель данных RDF)

```
<ex:index.html> <dc:creator> exstaff:85740 .
<ex:index.html> <exterms:creation-date> "August 16, 1999" .
<ex:index.html> <dc:language> "en".
```

- На этих примерах можно заметить "тяжеловесность" XML-синтаксиса RDF по сравнению с N3-синтаксисом. Но он более удобен для сериализации RDF-графов.
- Из вышесказанного о RDF и метаданных можно сделать вывод, что RDF имеет довольно слабые (по объему) выразительные средства и не основан на каком-либо логическом формализме. Это язык описания метаданных, причем метаданных в широком смысле слова: имеющих произвольную структуру и смысл. Пожалуй, единственный принцип, которому следует RDF, это основной лозунг Web. RDF - универсальный инструмент и поэтому требует настройки для решения конкретных специализированных задач. Способ такой "настройки" состоит в расширении RDF при помощи словарей.

- OWL (Ontology Web Language) - это язык, базирующийся на направлении Semantic Web, служащий для представления web-онтологий предметных областей, одобренный консорциумом W 3 C . Под онтологией понимается некоторый набор терминов предметной области и связей между этими терминами.
- OWL предоставляет три подмножества, имеющие различную степень детализации:
- OWL Lite предназначено для пользователей или приложений, которым необходима лишь классификационная иерархия сущностей и некоторые простые условия согласованности сущностей.
- OWL DL (Description Logic) рассчитано на пользователей, которым необходима максимальная степень выразительных возможностей языка без потери вычислительной полноты, без потери ни одного из семантических воплощений - содержательных толкований выводов, полученных формально-логическим путем и разрешимости, что означает что вычисления, будут закончены за конечное время. Уровень OWL DL ориентирован на существующие сегодня системы описания знаний и системы логического программирования.
- OWL Full рассчитано на пользователей, которым необходимы максимальные выразительные возможности языка и свобода выбора конечного формата Resource Description Framework (RDF), но без каких-либо гарантий вычислительной полноты и разрешимости. OWL Full позволяет расширить смысл термина, взятого из какого-либо заданного словаря, и добавить его в онтологию.

- В качестве своего синтаксиса OWL использует язык XML. Основными элементами языка являются свойства, классы и ограничения. Эти элементы позволяют реализовать представление о мире, как о множестве сущностей (объектов), характеризуемых некоторым набором свойств. Эти сущности состоят между собой в определенных отношениях и объединяются по определенным признакам (свойствам и ограничениям) в группы (классы).
- В языке OWL свойства подразделяются на два вида: свойства-характеристики (DatatypeProperty) и свойства-связи (ObjectProperty). Первые характеризует объекты (классы) и принимают в качестве своих значений данные определенных типов. Вторые ассоциирует объекты (классы) друг с другом и соответственно принимают в качестве своих значений объекты (классы).

- Кроме того, на свойства могут накладываться ограничения. Ограничения подразделяются на два вида: глобальные и локальные. К глобальным ограничениям относятся домены (domain) (классы, объекты которых могут обладать этими свойствами) и диапазоны (range) (классы, объекты которых могут выступать в качестве значений этих свойств). Локальные ограничения накладываются на свойства в рамках определенного класса и могут еще более сужать диапазоны для свойств в рамках этого класса, определять мощность свойств и их виды.
- Также язык OWL имеет механизмы описания версий онтологии и механизмы агрегирования данных, содержащихся в онтологиях.

- Структура OWL-онтологии
- Любая онтология имеет заголовок и тело. В заголовке содержится информация о самой онтологии (версия, примечания), об импортируемых онтологиях. За заголовком следует тело онтологии, содержащее описания классов, свойств и экземпляров.
- Базовые элементы OWL
- Классы
- В OWL введен новый термин - класс (owl:Class). Необходимость этого объясняется тем, что не все классы диалектов OWL DL и OWL Lite являются RDFS-классами (в этом случае owl:Class является подклассом rdfs:Class). В диалекте OWL Full подобных ограничений нет, и owl:Class фактически является синонимом rdfs:Class.
- Для организации классов в иерархию используется свойство rdfs:subClassOf.

- Особое место занимают два взаимодополняющих класса - owl:Thing и owl:Nothing. Первый из них является надклассом любого класса OWL, второй - подклассом любого класса OWL. Экземпляр любого класса OWL входит в экстенсионал класса owl:Thing. Экстенсионал класса owl:Nothing является пустым множеством.
- OWL-класс может быть описан шестью способами:
- идентификатором класса (URI);
- перечислением всех экземпляров класса;
- ограничением на значение свойства;
- пересечением 2-х и более определений классов;
- объединением 2-х и более определений классов;
- дополнением (логическим отрицанием) определения класса.

- Только первый способ определяет именованный класс OWL. Все оставшиеся определяют анонимный класс через ограничение его экстенсионала. Способ 2 явно перечисляет экземпляры класса, способ 3 ограничивает экстенсионал только теми экземплярами, которые удовлетворяют данному свойству. Способы 4-6 используют теоретико-множественные операции (объединение, пересечение и дополнение) над экстенсионалами соответствующих классов, чтобы определить экстенсионал нового класса.
- Описания класса являются строительными блоками для определения классов посредством аксиом.
- Простейшая аксиома, определяющая именованный класс:
- <owl:Class rdf:ID="Human"/>
- Всё, что постулирует эта аксиома, - существование класса с именем Human.

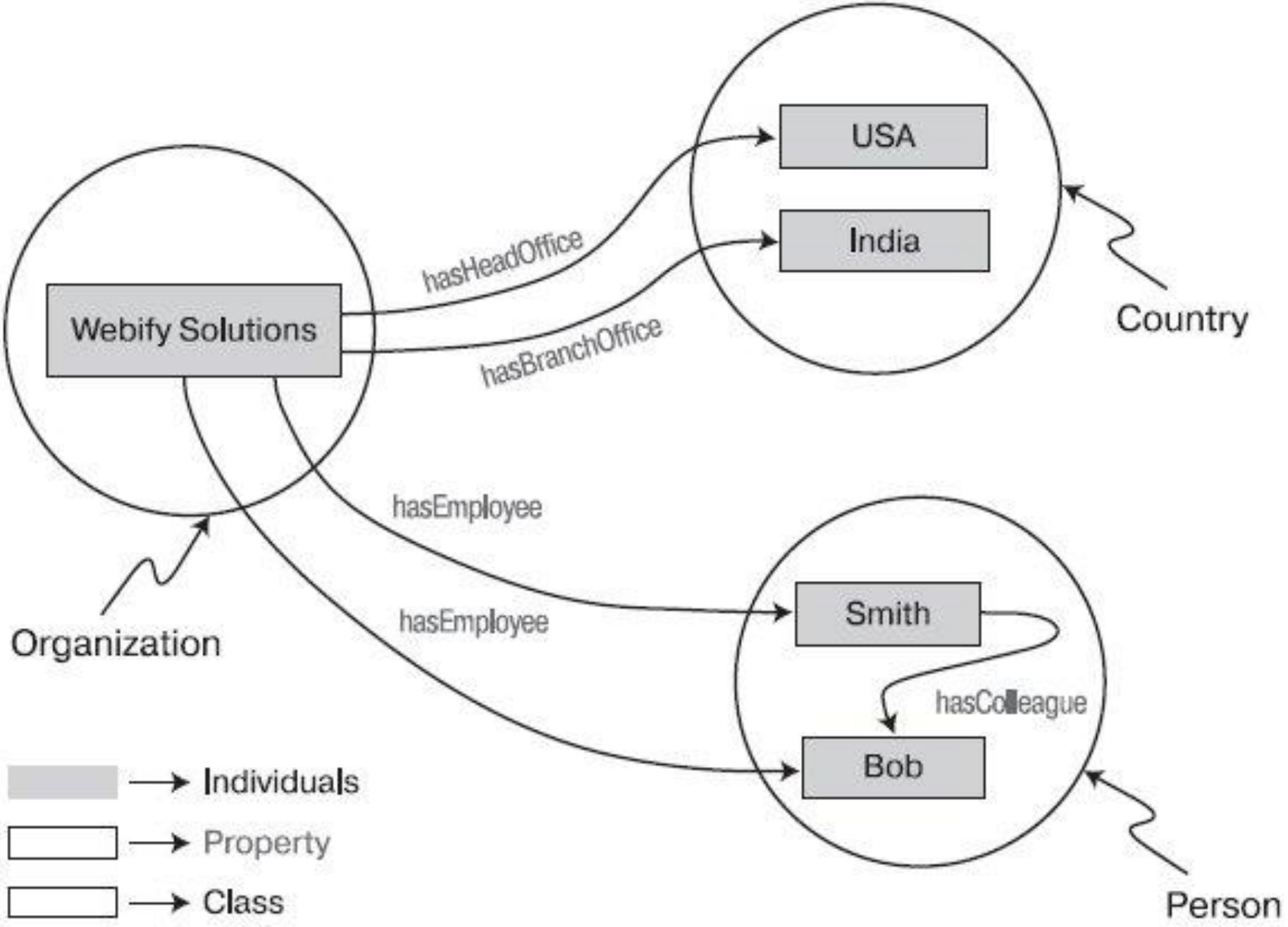
- В OWL определены еще 3 конструкции, комбинируя которые, можно определять более сложные аксиомы классов:
- rdfs:subClassOf говорит о том, что экстенсионал одного класса (подкласса) полностью входит в экстенсионал другого (надкласса);
- owl:equivalentClass говорит о том, что экстенсионалы двух классов совпадают;
- owl:disjointWith говорит о том, что экстенсионалы двух классов не пересекаются. Иногда говорят, что таким образом определяются дизъюнктивные классы.

- Свойства
- В OWL выделяют две категории свойств: свойства-объекты (или объектные свойства) и свойства-значения. Первые связывают между собой индивиды (экземпляры классов). Вторые связывают индивиды со значениями данных. Оба класса свойств являются подклассами класса rdf:Property.
- Для определения новых свойств как экземпляров owl:ObjectProperty или owl:DatatypeProperty используются аксиомы свойств.
- Пример аксиомы:
- <owl:ObjectProperty rdf:ID="hasParent"/>
- Все, что постулирует данная аксиома, - существование некоторого свойства hasParent, связывающего экземпляры класса owl:Thing друг с другом.

- Кроме того, OWL поддерживает следующие конструкции для построения аксиом свойств:
- Конструкции RDFS: rdfs:subPropertyOf (определяет подсвойство данного свойства), rdfs:domain (определяет домен свойства) и rdfs:range (определяет диапазон свойства)
- Отношения между свойствами: owl:equivalentProperty (определяет эквивалентное свойство) и owl:inverseOf (определяет обратное свойство).
- Ограничения глобальной кардинальности: owl:FunctionalProperty (определяет однозначное свойство - однозначное отображение домена свойства на диапазон) и owl:InverseFunctionalProperty (обратно функциональное свойство, т.е. определяет, что свойство, обратное данному свойству, является однозначным).
- Логические характеристики свойства: owl:SymmetricProperty (определяет свойство как симметричное) и owl:TransitiveProperty (определяет транзитивное свойство).

- Индивиды определяются при помощи аксиом индивидов (т.н. фактов). Рассмотрим два вида фактов:
- факты членства индивидов в классах и факты о значениях свойств индивидов;
- факты идентичности/различности индивидов.
- Пример аксиом индивидов первого вида:
- <Балет rdf:ID="ЛебединоеОзеро">
- <имеетКомпозитора rdf:resource="#Чайковский"/>
- </Балет>
- Данная аксиома постулирует сразу 2 факта: (1) существует некоторый индивид класса Балет, имеющий имя ЛебединоеОзеро ; (2) этот индивид связан свойством имеетКомпозитора с индивидом Чайковский (который определен где-то в другом месте). Первый факт говорит о членстве в классе, второй - о значении свойства индивида.

- Аксиомы второго вида необходимы для суждения об идентичности индивидов. Дело в том, что в OWL не делается никаких предположений ни о различии, ни о совпадении двух индивидов, имеющих различные идентификаторы URI. Подобные утверждения выражаются аксиомами идентичности с помощью следующих конструкций:
- owl:sameAs постулирует, что две ссылки URI ссылаются на один и тот же индивид;
- owl:differentFrom постулирует, что две ссылки URI ссылаются на разные индивиды;
- owl:AllDifferent предоставляет средство для определения списка попарно различных индивидов.


USA
India
Country
Webify Solutions
hasHeadOffice
hasBranchOffice
hasEmployee
hasEmployee
Organization
Smith
hasColleague
Bob
Individuals
Property
Class
Person

- SPARQL
- Вероятно, сами по себе языки представления онтологий не были бы так сильно востребованы, если бы не возникало необходимости автоматически обрабатывать онтологии, наполнять их содержимым и выполнять к ним запросы. Наиболее популярными среди языков запросов к RDF-хранилищам на сегодняшний день являются языки RDQL и SPARQL.
- Рассмотрим несколько упрощенный синтаксис SPARQL-запроса:
- SELECT <список_перем>
- FROM <URI_онтологии>
- WHERE { <список_шаблонов>.
- FILTER <фильтр>
- }

- Где: список_перем - список имен переменных; URI_онтологии - URI-ссылка на онтологию; список_шаблонов - список шаблонов; фильтр - ограничения на значения переменных.
- Допустим, онтология содержит следующие RDF-триплеты:
- (Foo1, category, "Total Members")
- (Foo1, rdf:value, 199)
- (Foo2, category, "Total Members")
- (Foo2, rdf:value, 200)
- (Foo2, category, "CATEGORY X")
- (bar, category, "CATEGORY X")
- (bar, rdf:value, 358)

- Проследим за ходом выполнения запроса (имена переменных предваряются знаком " ?")
- SELECT ?cat ?val
- FROM <URI_онтологии>
- WHERE { ?x rdf:value ?val.
- ?x category ?cat.
- FILTER (?val>=200)
- }
- Семантика запроса: "Выдайте все объекты cat предиката category, субъект которого (x) является также субъектом предиката rdf:value со значением объекта val, не меньшим 200. Вместе со значениями cat выдайте соответствующие значения val ".

- Ход выполнения запроса:
- На место переменной x могут быть подставлены Foo1, Foo2 и bar (из исходной онтологии), причем Foo2 может быть подставлен дважды, поскольку имеет два свойства category.
- При подстановке Foo1 значение переменной val не удовлетворяет ограничению в предложении FILTER SPARQL-запроса. Во всех остальных случаях все условия запроса выполнены.
- Результат выполнения запроса - 3 пары значений (cat, val):
- [
- ["Total Members", 200],
- ["CATEGORY X", 200],
- ["CATEGORY X", 358]