Языки представления онтологий: RDFS, OWL. Язык запросов SPARQL

- **RDFS**
- **OWL**
- **Запросы к RDF/OWL: SPARQL.**

RDFS

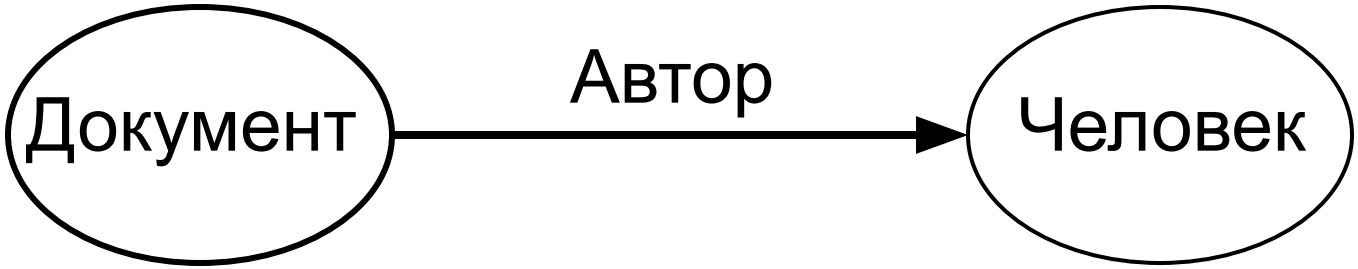
- RDFS – язык описания словарей для RDF.
- RDF Schema определяет классы, свойства и другие ресурсы.
- RDFS является семантическим расширением RDF.

Система классов и свойств языка RDFS

- класс
- свойство
  - -домен
  - -диапазон
- сравнение с системой классов ООП.

Пример (начало)

Определим свойство «автор» с доменом «Документ» и диапазоном «Человек».



Пример (окончание)

***a-la RDF** (дескриптивная логика):*

Класс («Документ»);

Класс («Человек»);

Свойство («автор», «Документ», «Человек»).

a-la Java:

```
Класс «Документ»
{
«Человек» «автор»
}
```

В случае появления дополнительной информации о свойствах «Документа», нет необходимости изменять описание класса «Документ». Достаточно добавить новое свойство с соответствующим доменом.

Классы (1)

- Ресурсы могут объединяться в группы называемые *классами.*
- Члены класса называются экземплярами класса.
- Экземпляры и классы являются ресурсами RDF.
- Свойство rdf:type используется для того, чтобы указать, что ресурс является экземпляром класса

Классы (2)

- RDF отделяет класс от множества его экземпляров (т.н. экстенсионала).
- Два класса с одинаковыми экстенсионалами считаются различными, если они имеют разные наборы свойств (интенсионалы).

Пример (интенсионал и экстенсионал)

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 описывается своим экстенсионалом, множества B и C описываются интенсионалами, т.е. используя характеристические свойства данного множества.

Парадокс Рассела.

Примечательно, что RDF нарушает одну из основных аксиом теории множеств: классу RDF **не** запрещено быть экземпляром самого себя.

Классы (3)

- Группа ресурсов, являющихся классами в RDFS описывается термином *rdfs:Class*
- Отношение «подкласс-надкласс», описывается RDFS свойством *rdfs:subClassOf.*
- Любой класс 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.

Перечень классов RDFS

Имя класса	Пояснение
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:ContainerMembershipProperty	Класс свойств «членства» в контейнерах, rdf:_1, rdf:_2, ..., подсвойствами всех свойства является
rdf:List	Класс RDF списков.

Перечень свойств 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 сеноъекта.	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 утверждения (см. реификация).	rdf:Statement	rdfs:Resource
rdf:object	Объект RDF утверждения (см. реификация).	rdf:Statement	rdfs:Resource

Реификация
или материализация утверждений

- К реификации прибегают, когда необходимо сделать утверждение об утверждении RDF
- Для этого используется специальный класс *rdf:Statement* и его свойства *rdf:subject*, *rdf:predicate*, *rdf:object*.
- Каждое RDF утверждение является экземпляром класса *rdf:Statement*

Пример (начало)

Утверждение об авторстве исходного утверждения:

Утверждение 1: «товар Т имеет цену х». Допустим, что оно сделано Ивановым Иваном Ивановичем на языке RDF.

Требуется высказать утверждение 2 о том, что именно Иванов И.И. сделал утверждение 1.

Пример (продолжение)

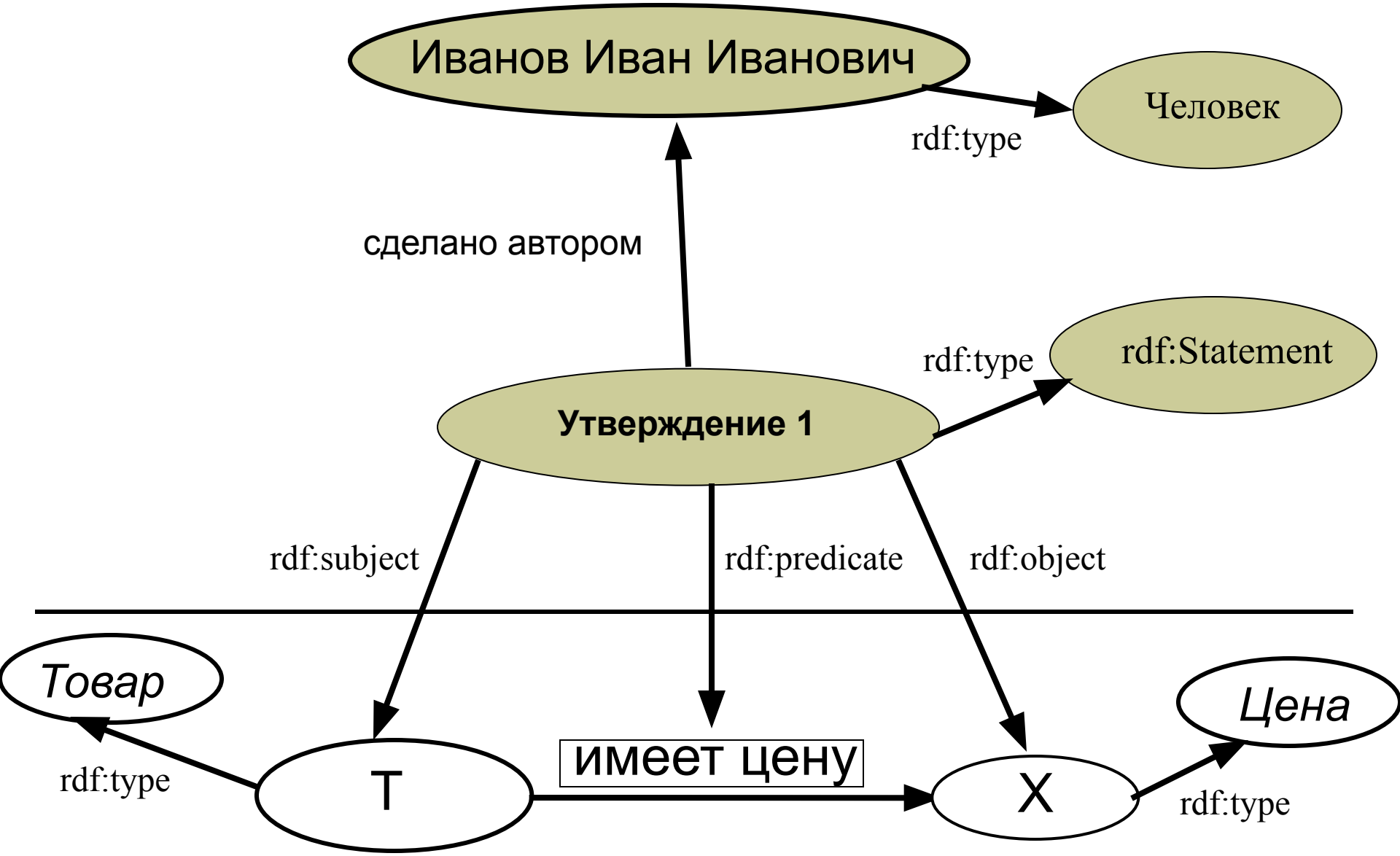
Товар T
rdf:Property имеет цену
Цена x

*rdf:Statement Утверждение 1 **
*rdf:subject T **
*rdf:predicate имеет цену **
*rdf:object x **

rdf:Statement Утверждение 1 +
rdf:Property сделано автором +
Человек Иванов Иван Иванович +

- ***Важный момент:* Косвенные утверждения.**

Пример (окончание)



Возможности RDF, RDF Schema

обобщенный способ работы с метаданными

- ориентирован на программное обеспечение в качестве конечного потребителя информации
- позволяет осуществлять автоматическую обработку Web-ресурсов
  - поиск
  - каталогизацию
  - генерацию иерархических карт сайтов

Ограничения языка RDF, RDF Schema

- 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 Сети)» 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 целостности ложится на получателей (конечных пользователей) метаданных, т.е. на разработчиков приложений обрабатывающих RDF.

Способы представления RDF-описаний (1)

- 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>
```

Способы представления RDF-описаний (2)

- N3 (N-Triples) синтаксис (удобный для чтения человеком, но еще и расширяющий исходную модель RDF)

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

Краткие итоги (RDF)

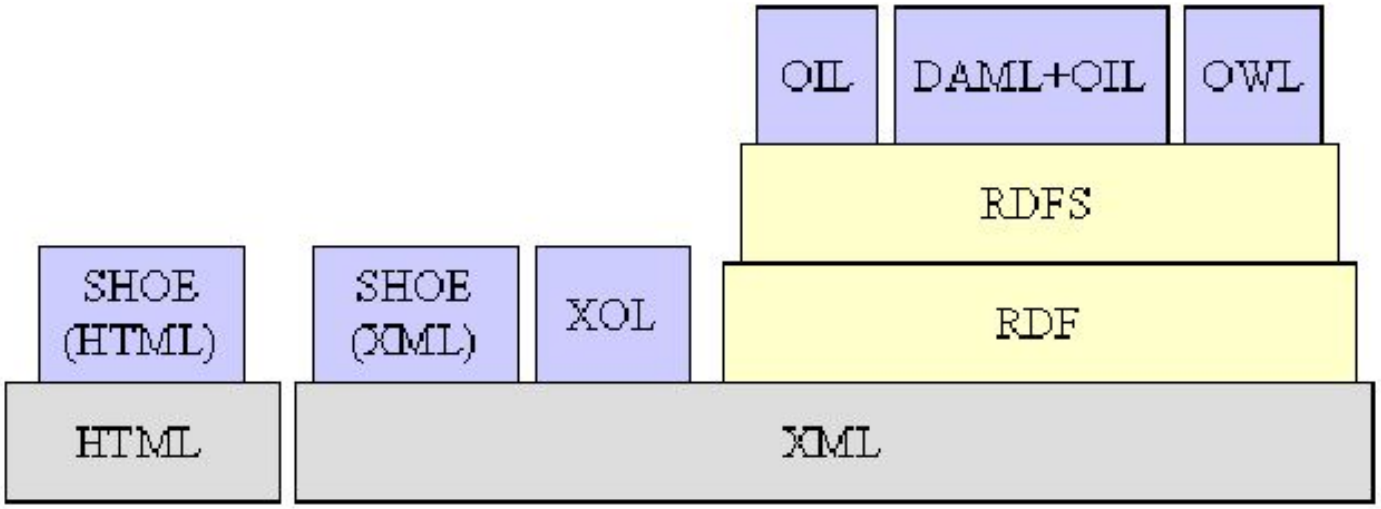
- RDF – язык описания метаданных в Сети
- Модель данных RDF – ориентированный граф
- RDF граф строится на основе элементарных высказываний (триплетов)
- Форма высказываний – бинарное отношение (S,P,O)
- RDF чрезвычайно выразителен (кто угодно может сказать что угодно о чем угодно)
- RDFS служит для определения словарей RDF.

OWL

- OWL (Web Ontology Language) – язык представления онтологий в Web. Фактически это словарь расширяющий набор терминов определенных RDFS.
- OWL-онтологии могут содержать описания классов, свойств и их экземпляров.

Исторические предшественники OWL

- DAML, OIL, SHOE, XOL
- OWL является рекомендацией W3C и объединяет лучшие черты своих предшественников



Три диалекта OWL

- OWL Lite (простота)
- OWL DL (полнота и разрешимость)
- OWL Full (выразительная мощь)

- Каждый из этих диалектов (кроме Lite) является расширением предыдущего.
- Как следствие:

  Любая OWL Lite онтология является OWL DL онтологией, а любая OWL DL онтология является OWL Full онтологией.

Структура OWL онтологии

- Заголовок
  - версия
  - примечания
  - импортируемые онтологии
- Тело
  - описания классов, свойств и индивидов в форме аксиом

OWL. Базовые элементы. Классы (1)

- Различия в смысле owl:Class для диалекта Full и DL
- Специальные классы
  - Thing
  - Nothing

OWL. Базовые элементы. Классы (2)

- 6 способов определения класса
  1. идентификатором класса (URI)
  2. перечислением всех экземпляров класса
  3. ограничением свойства
  4. пересечением 2 и более определений классов
  5. объединением 2 и более определений классов
  6. дополнением определения класса

OWL. Базовые элементы. Классы (3)

- Простейшая аксиома, определяющая именованный класс:
  - <owl:Class rdf:ID="Human"/>
  - Все что постулирует эта аксиома – существование класса с именем “Human”.
- Конструкции OWL для определения сложных аксиом классов:
  - **rdfs:subClassOf** - говорит о том, что экстенсионал одного класса (подкласс) полностью входит в экстенсионал другого (надкласс).
  - **owl:equivalentClass** - говорит о том, что экстенсионалы двух классов совпадают.
  - **owl:disjointWith** - говорит о том, что экстенсионалы двух классов не пересекаются.

OWL. Базовые элементы. Свойства (1)

- Две основные категории OWL свойств:
  - объектные свойства (owl:ObjectProperty)
    - связывают между собой индивиды
  - свойства-значения (owl:DatatypeProperty)
    - связывают индивиды со значениями данных

Оба класса свойств являются подклассами класса rdf:Property

Простейший пример аксиомы свойства:

<owl:ObjectProperty rdf:ID="hasParent"/>

Всё что постулирует эта аксиома – существование некоторого свойства "hasParent" связывающего экземпляры owl:Thing друг с другом.

OWL. Базовые элементы. Свойства (2)

- Конструкции для построения аксиом свойств (начало):
  - Конструкции RDF Schema:

    `rdfs:subPropertyOf` (определяет подсвойство данного свойства),

    `rdfs:domain` (определяет домен) и

    `rdfs:range` (определяет диапазон)
  - Отношения между свойствами:

    `owl:equivalentProperty` (определяет эквивалентное свойство) и

    `owl:inverseOf` (определяет обратное свойство)

OWL. Базовые элементы. Свойства (3)

- Конструкции для построения аксиом свойств (окончание)
  - Ограничения глобальной кардинальности: `owl:FunctionalProperty` (определяет однозначное свойство – однозначное отображение домена свойства на диапазон) и
    `owl:InverseFunctionalProperty` (взаимоднозначное отображение домена свойства на его диапазон, т.е. определяет существование однозначного свойства обратного данному свойству)
  - Логические характеристики свойства: `owl:SymmetricProperty` (определяет свойство как симметричное данному) и
  - `owl:TransitiveProperty` (определяет транзитивное свойство)

OWL. Базовые элементы. Индивиды (1)

- Индивиды определяются при помощи аксиом индивидов (фактов)
- 2 вида фактов:
  - (1) Факты о членстве индивидов в классах и о значении свойств индивидов.
  - (2) Факты об идентичности индивидов

OWL. Базовые элементы. Индивиды (2)

Пример аксиомы первого вида:

```
<Балет rdf:ID="ЛебединоеОзеро">
    <имеетКомпозитора rdf:resource="#Чайковский"/>
</Балет>
```

Данная аксиома постулирует сразу 2 факта:

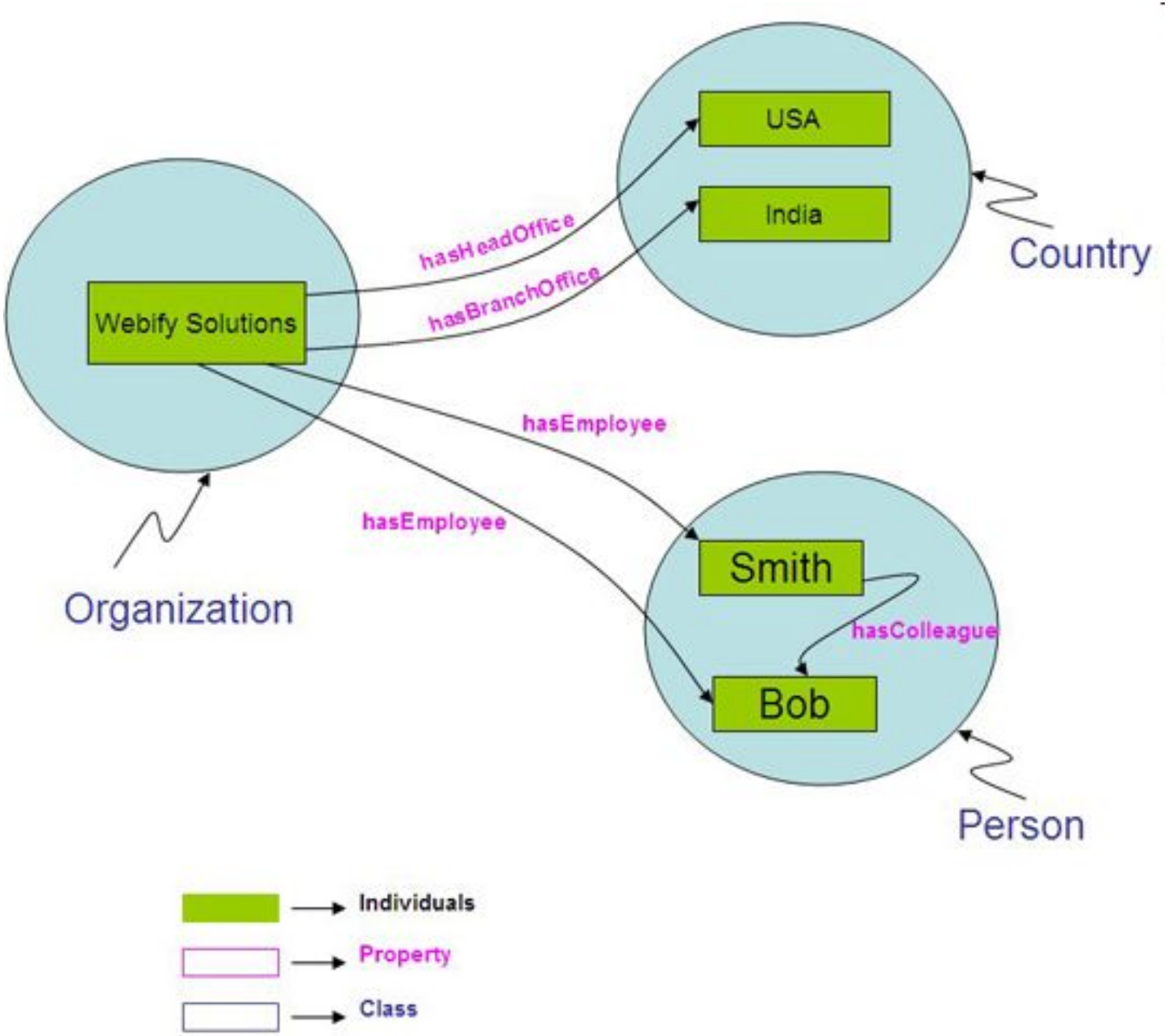
- (а) существует некоторый индивид класса “Балет” имеющий имя “ЛебединоеОзеро”;
- (б) этот индивид связан свойством “имеетКомпозитора” с индивидом: “Чайковский” (определенным где-то в другом месте).

Первый факт говорит о членстве в классе, второй – о значении свойства индивида.

OWL. Базовые элементы. Индивиды (3)

- Для описания фактов об идентичности индивидов используются аксиомы идентичности
- Вспомогательные конструкции OWL:
  - *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.

OWL. Базовые элементы. Пример.



Языки запросов к RDF хранилищам. Логический вывод над RDF-графами и онтологиями

Представление знаний в машинопонятном формате не имело бы никакого смысла, если бы к этим знаниям нельзя было обращаться, автоматически их обрабатывать и пополнять.

Имеются две близкие задачи:

- Первая задача связана с извлечением имеющихся в хранилище знаний – запросами к хранилищу (asking, querying).
- Вторая задача связана с применением логического вывода над имеющимися знаниями (reasoning, entailment).

SPARQL

- Синтаксис запроса (упрощенный)

```
SELECT    <v_list>
FROM   <onologyURI>
WHERE    {   <template_list>.
            FILTER <filter_expr>
        }
```

- v_list – список имен переменных
- onologyURI – ссылка на онтологию
- template_list – список шаблонов
- filter_expr – ограничения на значения переменных

Пример SPARQL (1)

Данные RDF в виде триплетов (S, P, O):

```
(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).
```

Пример SPARQL (2)

- **SPARQL - Запрос:**

```
SELECT ?cat ?val
WHERE {?x rdf:value ?val. ?x category ?cat.
FILTER(?val>=200).}
```

Предложение **SELECT** описывает переменные `cat` и `val`, значение которых необходимо вычислить.

Предложение **WHERE** накладывает шаблоны вида (S, P, O).

Внутри шаблона можно использовать связанные переменные - `x`

Предложение **FILTER** накладывает ограничение на значения переменной `val` извлеченных значения

Семантика запроса: выдайте все объекты `cat` предиката category, субъект которого (`x`) является также субъектом предиката `rdf:value` со значением `val`, не меньшим `200`. Вместе со значениями `cat` выдать соответствующие значения `val`.

Пример SPARQL (3)

Ход выполнения запроса:

- *На место переменной* `x` *могут быть подставлены Foo1, Foo2 и bar, причем Foo2 может быть подставлен дважды.*
- *При подстановке Foo1 значение переменной val не удовлетворяет ограничению в предложении FILTER. Во всех других случаях все условия запроса выполнены (см. результат).*
- Результат выполнения запроса:

```
[["Total Members", 200], ["CATEGORY X", 200],
["CATEGORY X", 358]]
```

Заключение. Основные вехи на пути к Semantic Web

Широкое распространение Web стандартов (рекомендаций W3C)

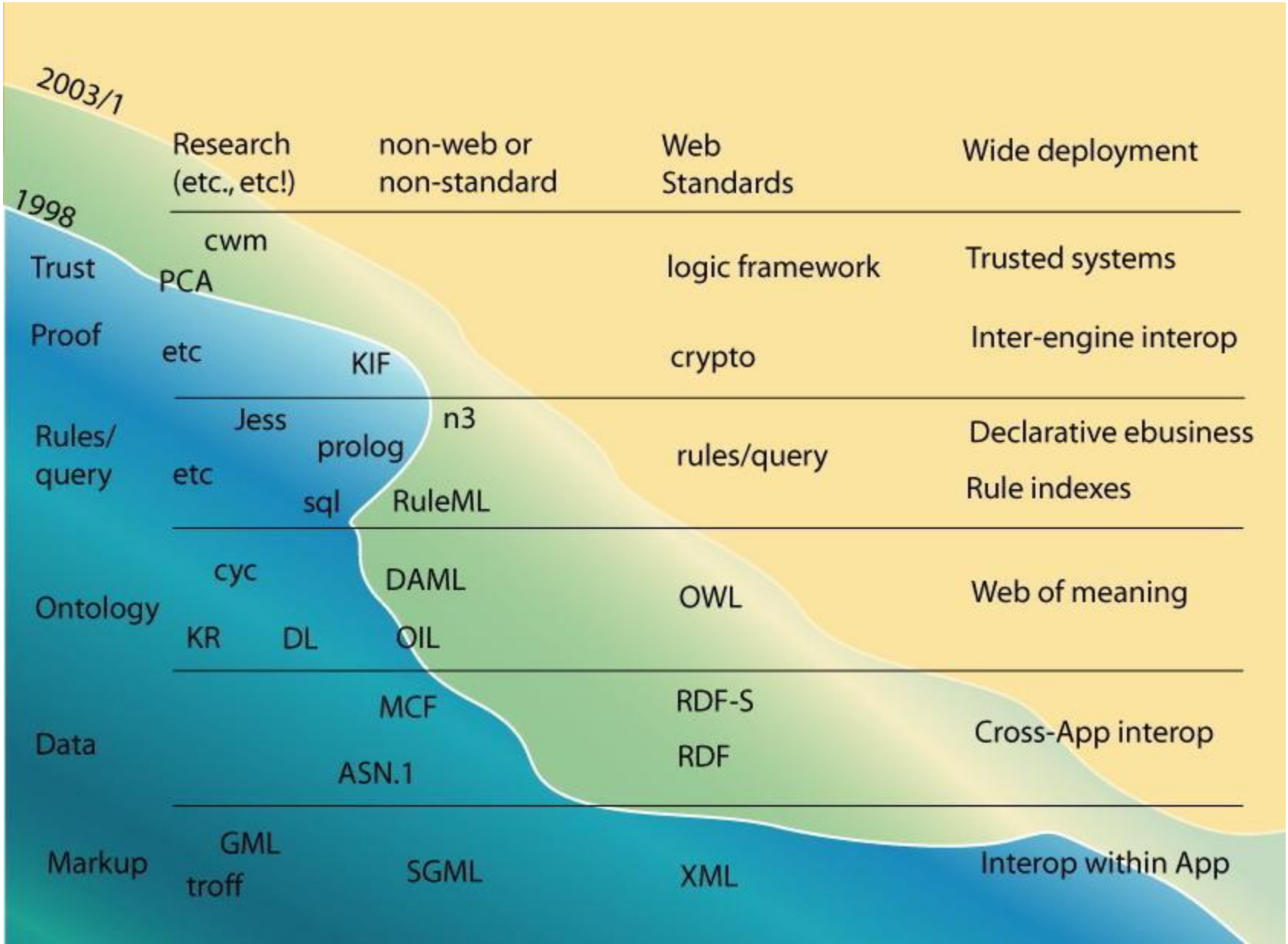
- XML
- RDF/RDFS
- OWL
- SPARQL
- RIF (Rule Interchange Format)

Наличие свободно распространяемых каркасов для разработки Semantic Web приложений

- Jena Framework (Java)
- Drive RDF Parser (C#)

Массовая разработка и использование онтологий!!!

Заключение. Semantic Web «прилив»



Вопросы к лекции

- Для чего нужен RDFS?
- Что такое реификация?
- Чем отличается класс RDFS от класса OWL?