# Графы знаний

Лекция 2 - Представление знаний в графах

М. Галкин, Д. Муромцев

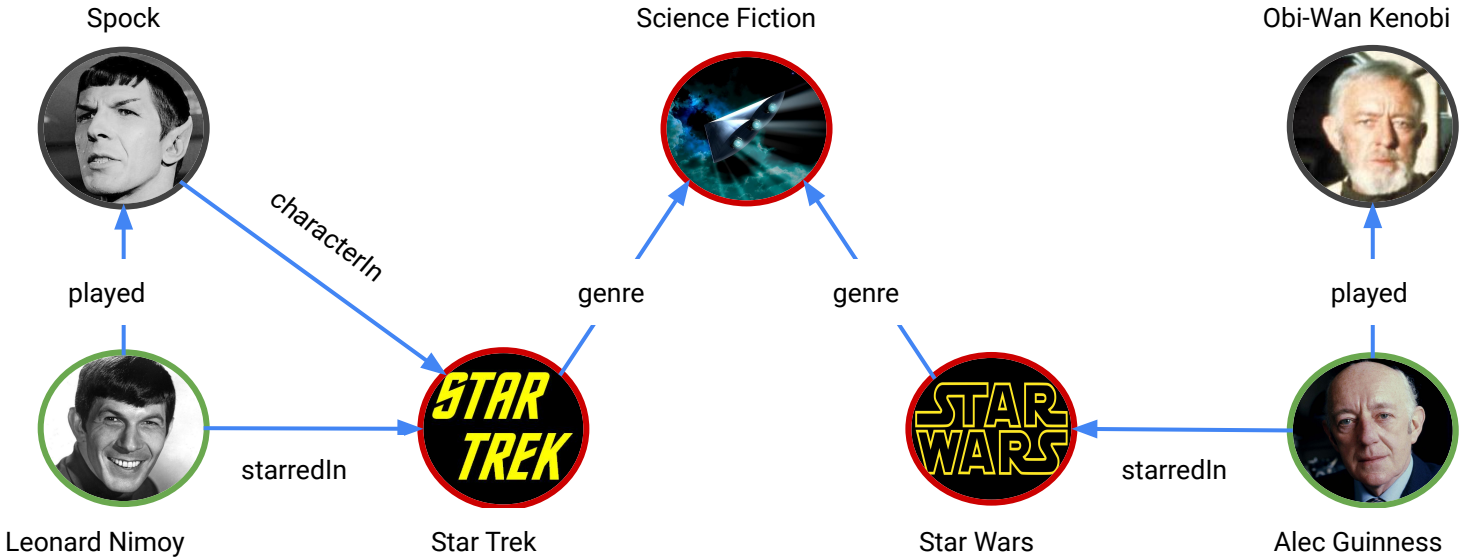
# Сегодня

1. Introduction
2. **Представление знаний в графах - RDF & RDFS & OWL**
3. Хранение знаний в графах - SPARQL & Graph Databases
4. Однородность знаний - Reification & RDF* & SHACL & ShEx
5. Интеграция данных в графы знаний - Semantic Data Integration
6. Векторные представления графов - Knowledge Graph Embeddings
7. Введение в теорию графов - Graph Theory Intro
8. Машинное обучение на графах - Graph Neural Networks & KGs
9. Некоторые применения - Question Answering & Query Embedding

# Содержание

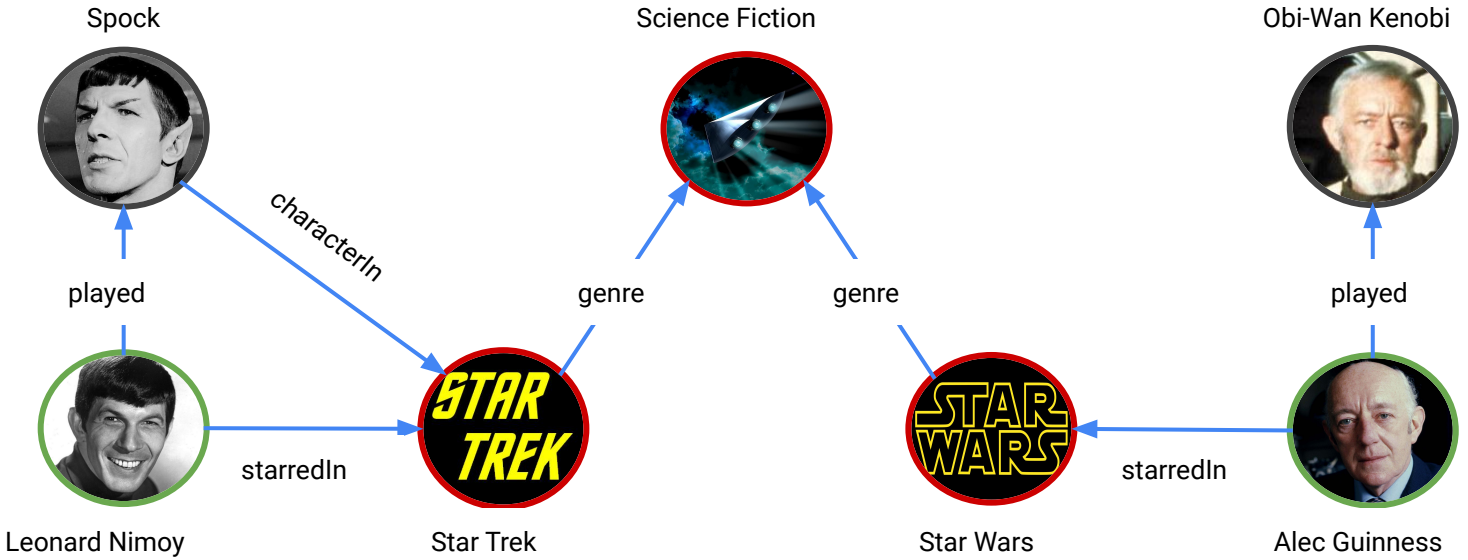
- Графы знаний = графы + знания
- Как представлять знания? Логика и семантика
- Модель RDF и RDFS
- Сериализации RDF
- OWL, классы, экземпляры, аксиомы
- Онтологии как схемы графов знаний

# Представление знаний - онтологическое




```
LeonardNimoy   starredIn     StarTrek;        AlecGuinness   starredIn     StarWars;
               played        Spock.                          played        Obi-Wan.
Spock          characterIn   StarTrek .       StarWars       genre         SciFi .
```

# Представление знаний - статистическое



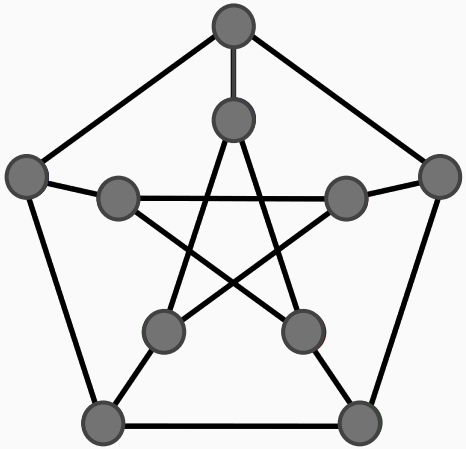

Spock = [0.1, 0.2, 0.3]
Leonard Nimoy = [0.4, 0.8, 0.1]
Star Trek = [0.22, 0.34, 0.87]

characterIn = [0.1, 0.1, 0.6]

Obi-Wan = [0.05, 0.25, 0.37]
Alec Guinness = [0.33, 0.5, 0.3]
Star Wars = [0.18, 0.4, 0.9]

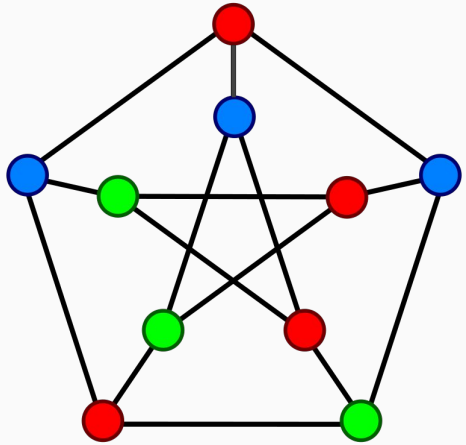
# Граф знаний

$G = (V,E) \mid E \subseteq \mathbb{R}^{|V| \times |V|}$



# Граф знаний

$G = (V,E) \mid E \subseteq \mathbb{R}^{|V|x|V|}$

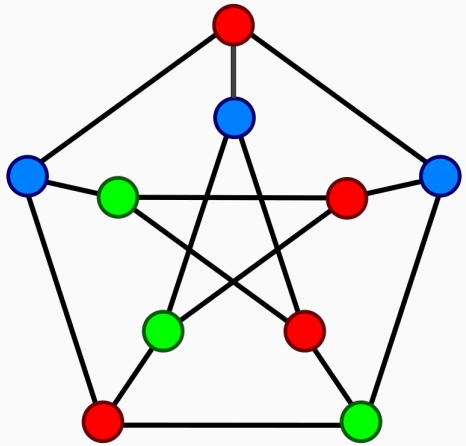


$\tau(v) = \gamma,\ v \in V$

$\tau(e) = \delta,\ e \in E$

# Граф знаний

$G = (V,E) \mid E \subseteq \mathbb{R}^{|V|x|V|}$



$\tau(v) = \gamma,\ v \in V$

$\tau(e) = \delta,\ e \in E$

- Вершины и ребра имеют типы (классы)
- Ребра (отношения) могут иметь разную значимость
- Классы и отношения имеют логический смысл
- Сетевая структура графа - более естественный способ представлять знания о мире

- Формальная семантика

# Формальная семантика

`I(Musician writes songs)`

- Семантика передает смысл символов некоторого формального или естественного языка

# Формальная семантика

$I(\text{Musician writes songs})$

$\text{Musician} \sqsubseteq \exists \text{writes.Song}$

- Семантика передает смысл символов некоторого формального или естественного языка
- Формальная семантика передает смысл языков в математических терминах

# Формальная семантика

`I(Musician writes songs)`

`Musician ⊑ ∃writes.Song`

`C(x), ⊑, ∀, ∃, ⊓, ⊔, ∃R.C, ∀R.C`

- Семантика передает смысл символов некоторого формального или естественного языка
- Формальная семантика передает смысл языков в математических терминах
- Формальная логика предоставляет математический аппарат для интерпретации языков

# Формальная семантика

`I(Musician writes songs)`

`Musician ⊑ ∃writes.Song`

`C(x), ⊑, ∀, ∃, ⊓, ⊔, ∃R.C, ∀R.C`

- Семантика передает смысл символов некоторого формального или естественного языка
- Формальная семантика передает смысл языков в математических терминах
- Формальная логика предоставляет математический аппарат для интерпретации языков
  - Пропозициональная логика
  - Логика предикатов
  - Модальная логика
  - Логика первого порядка
  - Дескрипционные логики

# Формальная семантика

`I(Musician writes songs)`

`Musician ⊑ ∃writes.Song`

`C(x), ⊑, ∀, ∃, ⊓, ⊔, ∃R.C, ∀R.C`

- Семантика передает смысл символов некоторого формального или естественного языка
- Формальная семантика передает смысл языков в математических терминах
  - **Теоретико-модельная семантика (семантика Тарского)**
- Формальная логика предоставляет математический аппарат для интерпретации языков
  - Пропозициональная логика
  - Логика предикатов
  - Модальная логика
  - **Логика первого порядка**
  - **Дескрипционные логики**

# Выразительность vs Сложность

Высокая выразительность

✓ Комплексные утверждения
✓ Богатая логика

– Часто алгоритмически неразрешимы

# Выразительность vs Сложность

Высокая выразительность

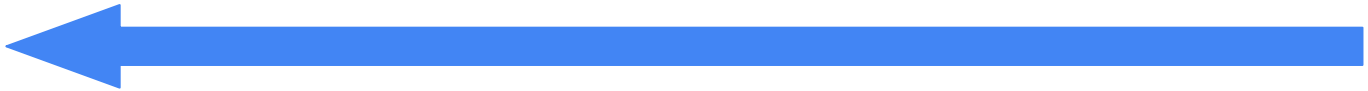
✓ Комплексные утверждения
✓ Богатая логика

– Часто алгоритмически неразрешимы

Низкая выразительность

✓ Для простых задач
✓ Полиномиальная сложность

– Сложные утверждения недоступны



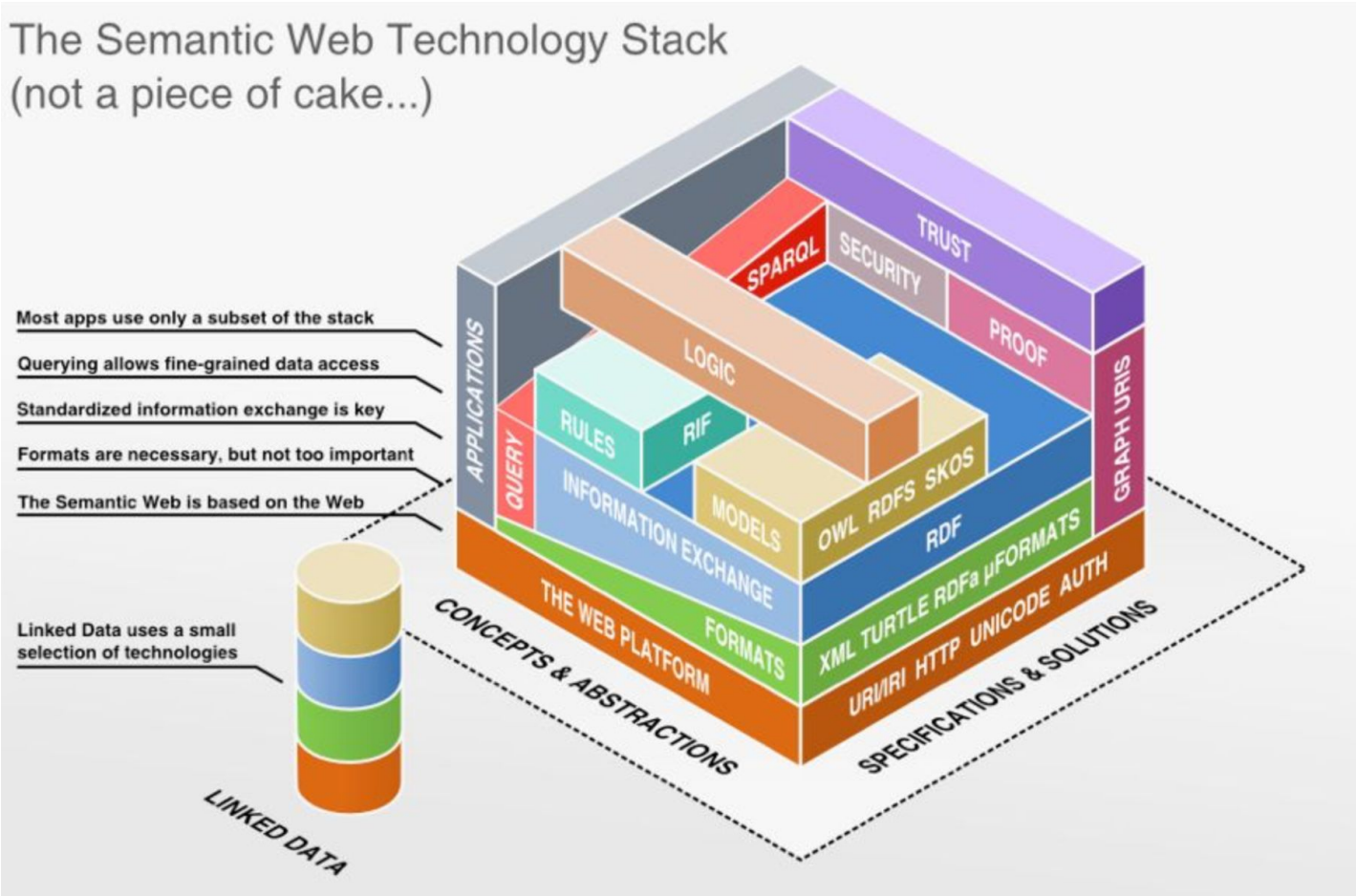
**First-order logic (FOL)**

**Description logics (DL)**

**Propositional logic (PL)**

# Semantic Web Layer Cake

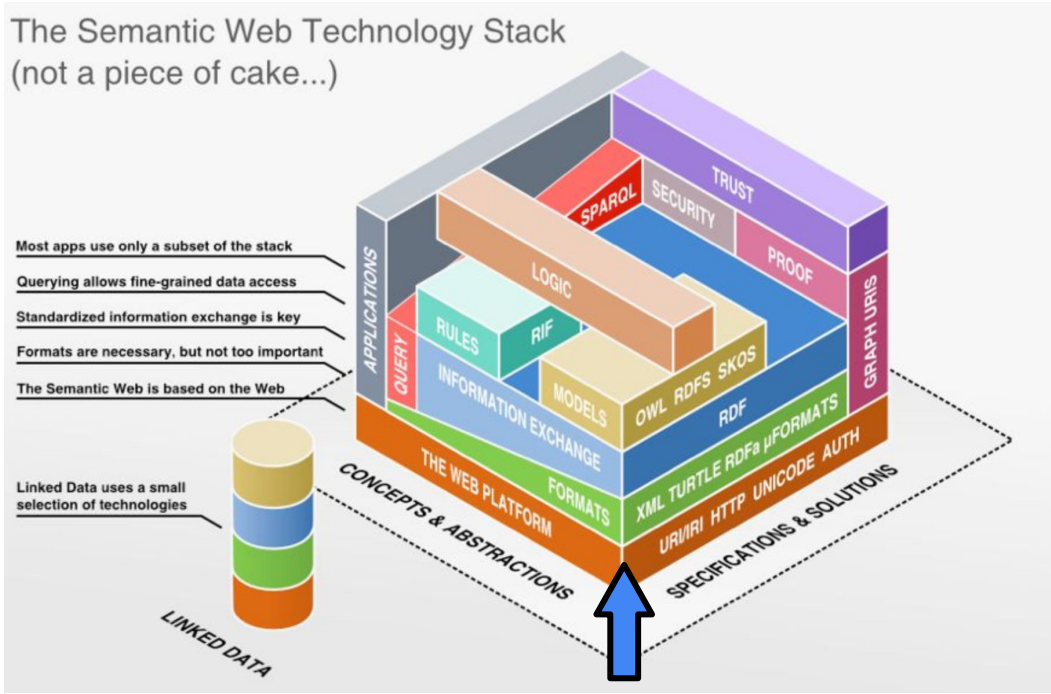
Идея логического представления знаний в Сети легла в основу концепции Semantic Web



# Semantic Web Layer Cake

The Web Platform

- URI / IRI
- HTTP
- Unicode
- Auth




Базовые средства передачи информации в сети.
Все сущности являются URI

`http://example.com/ITMO_University`

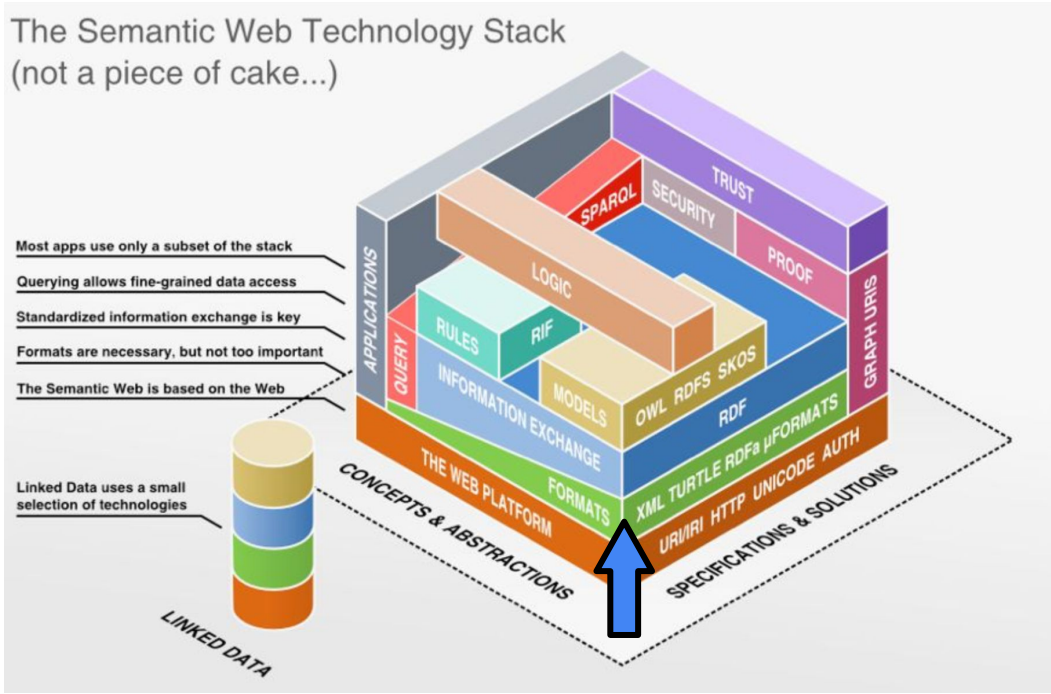
`https://www.wikidata.org/wiki/Q1342013`

# Semantic Web Layer Cake

Formats

- XML
- **Turtle**
- **JSON-LD**
- RDFa

Форматы сериализации моделей высших уровней

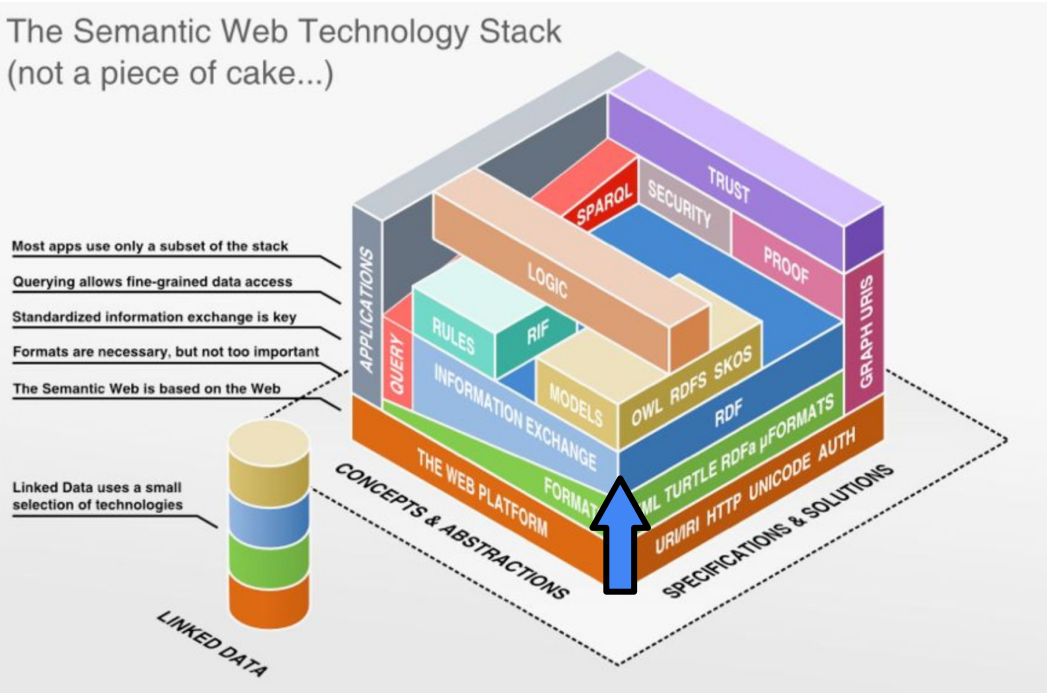


# Semantic Web Layer Cake

Information Exchange

- **RDF (Resource Description Framework)**

Вводит базовую графовую модель представления знаний в виде триплетов



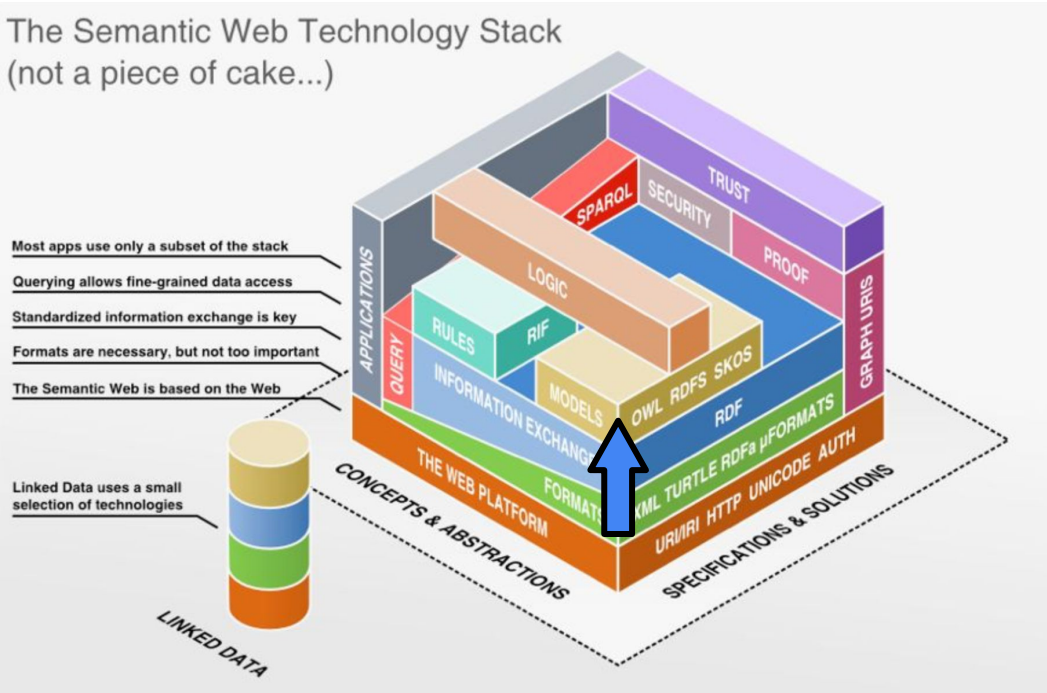

<subject> <predicate> <object>

# Semantic Web Layer Cake

Models & Rules & Logic

- **RDFS**
- **OWL**
- SKOS
- RIF , SWRL, SPIN
- Description Logics SROIQ(D), SHOIN(D)

Логические расширения RDF с большей выразительностью

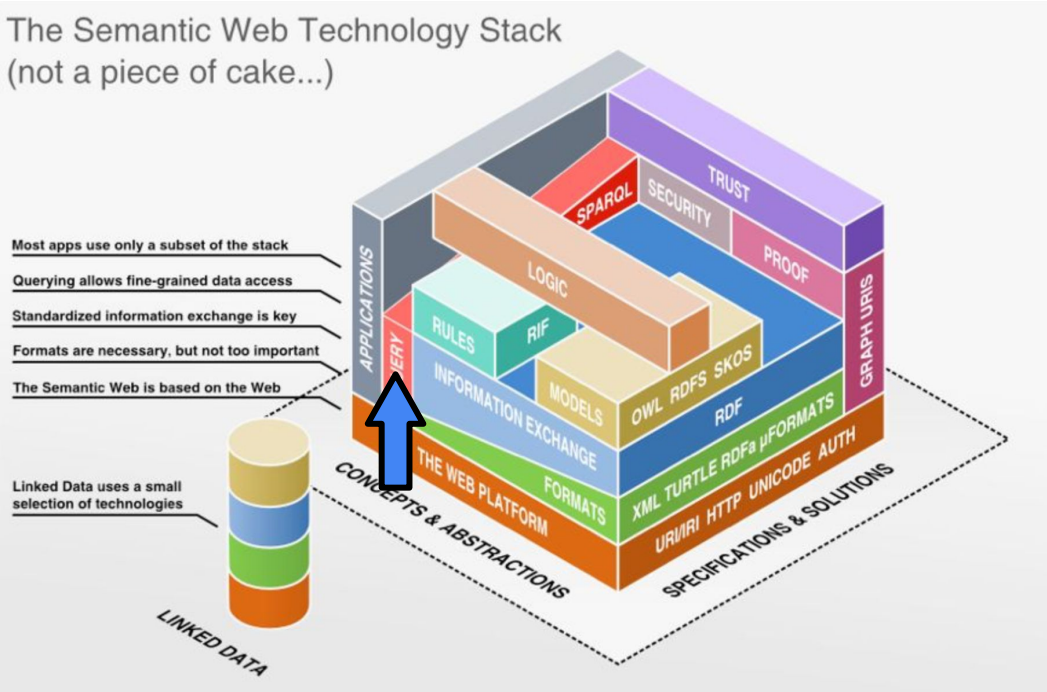


# Semantic Web Layer Cake

Querying

- **SPARQL**
- Cypher
- Gremlin

Язык запросов к RDF графам

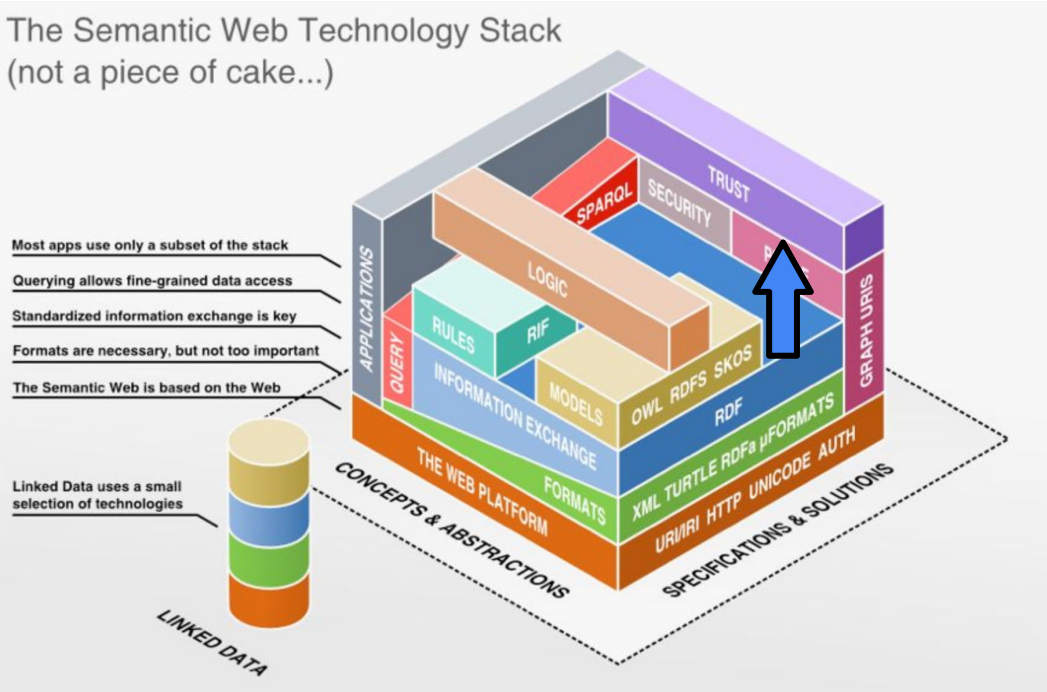


# Semantic Web Layer Cake

Security & Proof & Trust

- **SHACL, ShEx**
- VoID
- ProvO

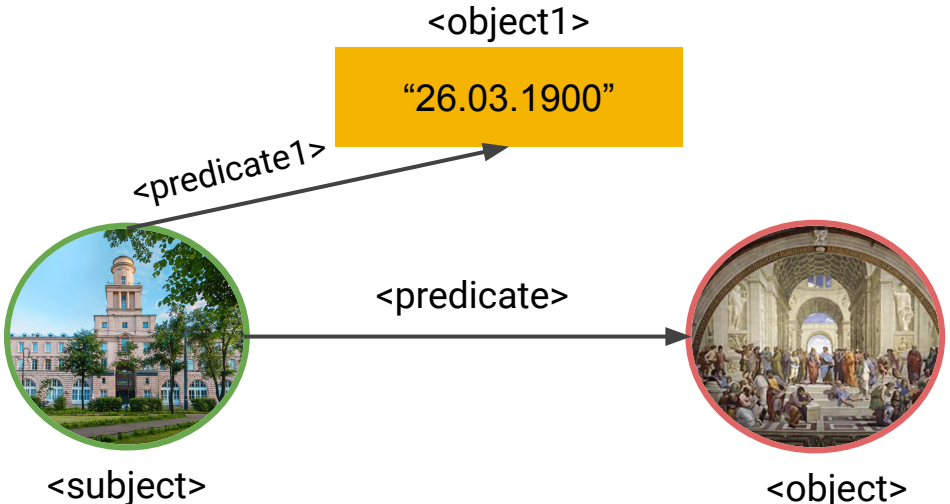
Средства обеспечения достоверности и корректности знаний



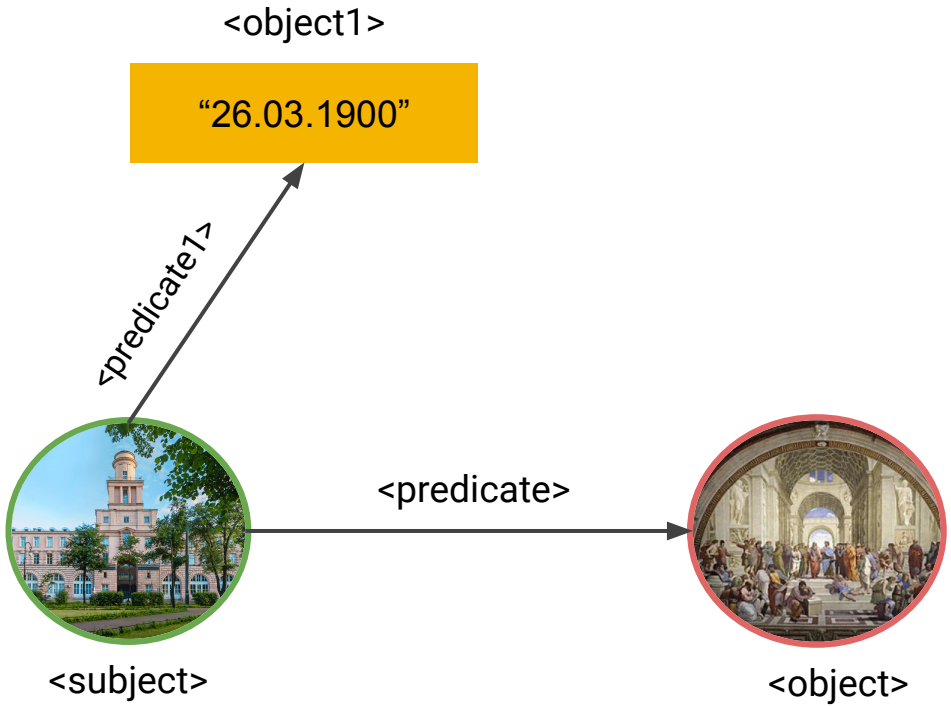
# RDF - Resource Description Framework

RDF Triple

**Definition 2.1.1** *Let $\mathcal{U}, \mathcal{B}, \mathcal{L}$ be disjoint infinite sets of URIs, blank nodes, and literals, respectively. A tuple $(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times (\mathcal{U}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is denominated as an RDF triple, where $s$ is called the subject, $p$ the predicate, and $o$ the object. (s,p,o) is a generalized RDF triple when $(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times (\mathcal{U}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. An element $\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$ is called an RDF term.*

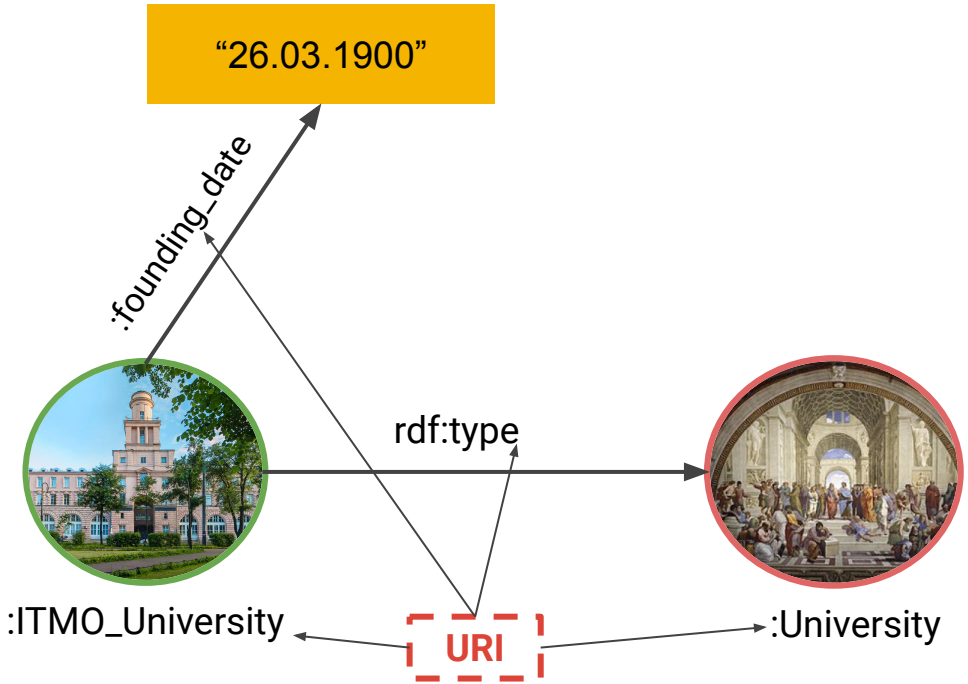


# RDF - Resource Description Framework




- Модель описания ресурсов
- Ресурсы уникально идентифицируются URI или литералами

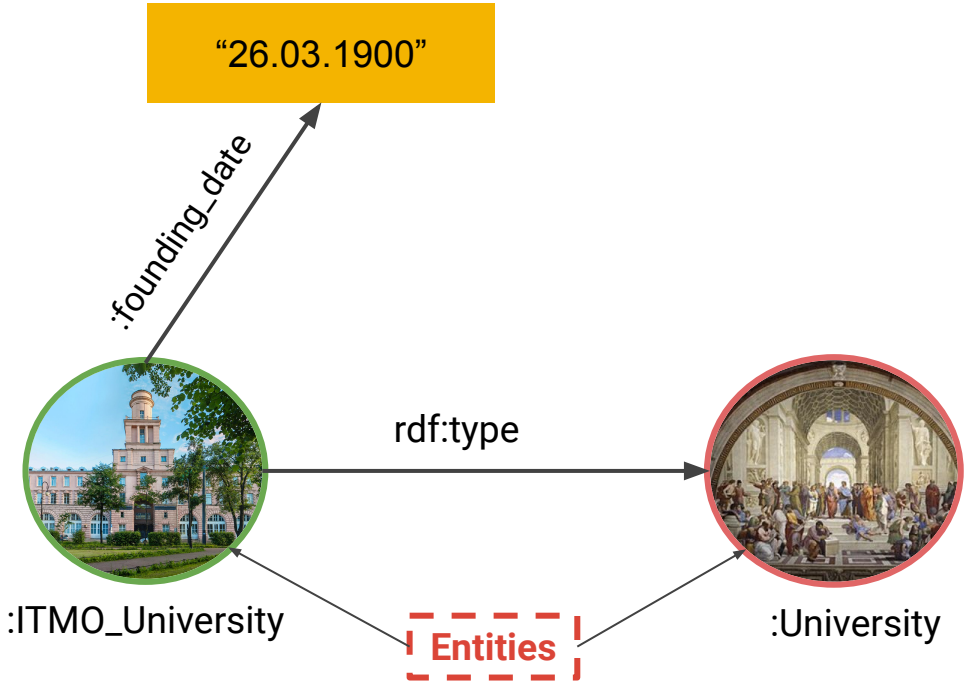
# RDF - Resource Description Framework




**URIs:**

- :ITMO_University
- :University
- rdf:type
- :founding_date

# RDF - Resource Description Framework

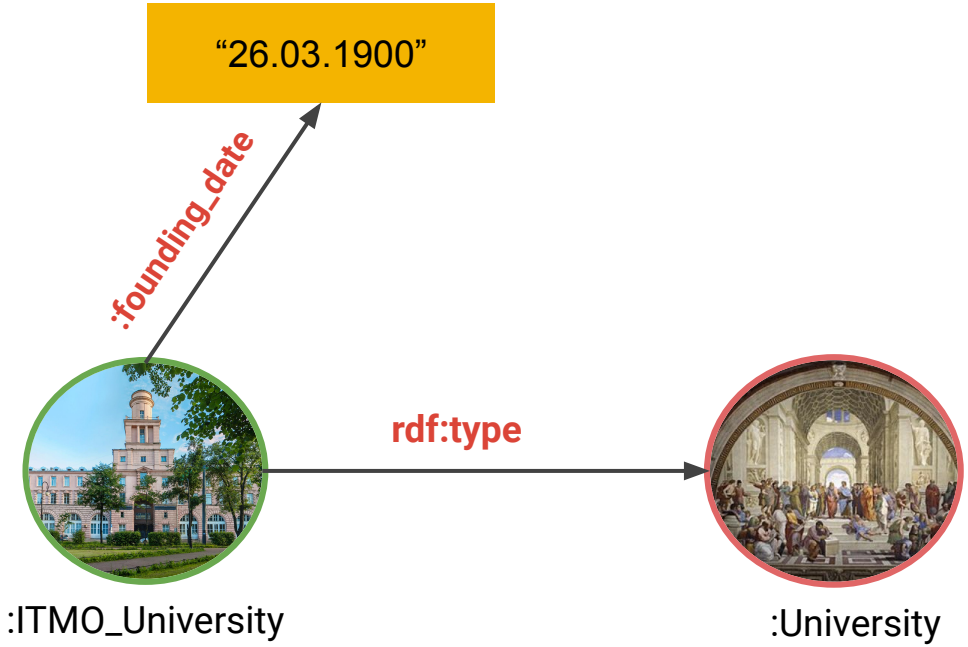



URIs:

- :ITMO_University
- :University
- rdf:type
- :founding_date

**Entities:**

# RDF - Resource Description Framework




URIs:

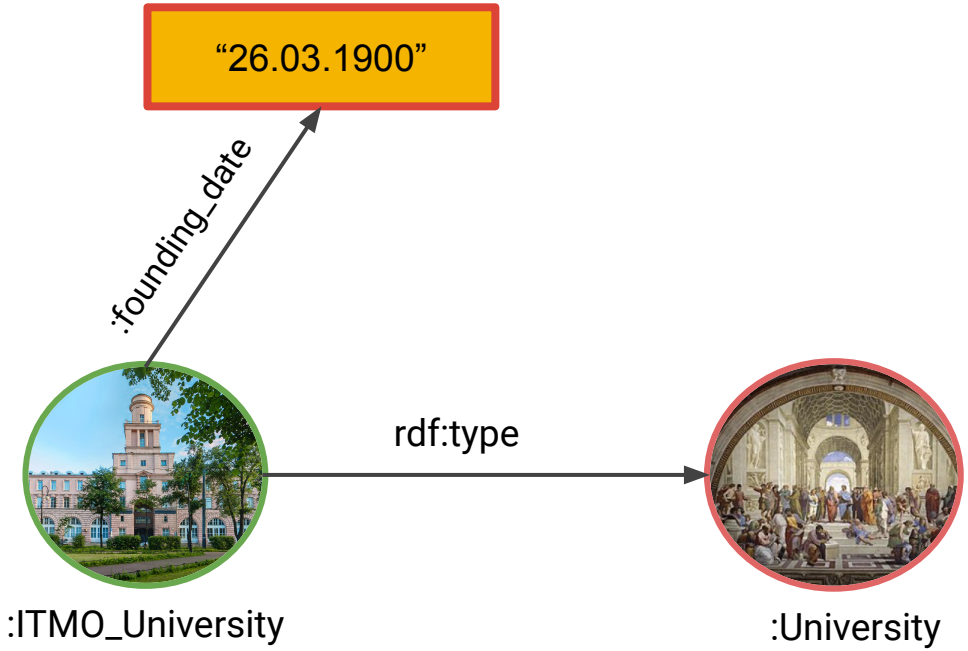
- :ITMO_University
- :University
- rdf:type
- :founding_date

Entities:

**Predicates:**

- rdf:type
- :founding_date

# RDF - Resource Description Framework




URIs:

- :ITMO_University
- :University
- rdf:type
- :founding_date

Entities:

Predicates:

- rdf:type
- :founding_date

**Literals:** “26.03.1900”

# RDF: URI

- Uniform Resource Identifier - способ назначения уникальных идентификаторов абстрактным и реальным сущностям и предикатам (RFC 3986)
- URL-подобные идентификаторы для Content Negotiation
- Префиксы для сокращения строк (http://prefix.cc)

```
                    https://en.ifmo.ru/en    ifmo:ITMO
          http://example.org/ITMO_University  ex:ITMO_University
http://dbpedia.org/resource/ITMO_University   dbr:ITMO_University
     https://www.wikidata.org/wiki/Q1342013   wd:Q1342013
```

- **Одна и та же сущность может быть по-разному представлена в разных графах (interlinking & alignment problem)**

# RDF: Literals

- Литералы - строки, числа и XSD-определенные типы

| “текст” | 100 | True | “5.5”^^xsd:float |
|---|---|---|---|

- Хороший тон и рекомендация - назначать строкам тэг языка

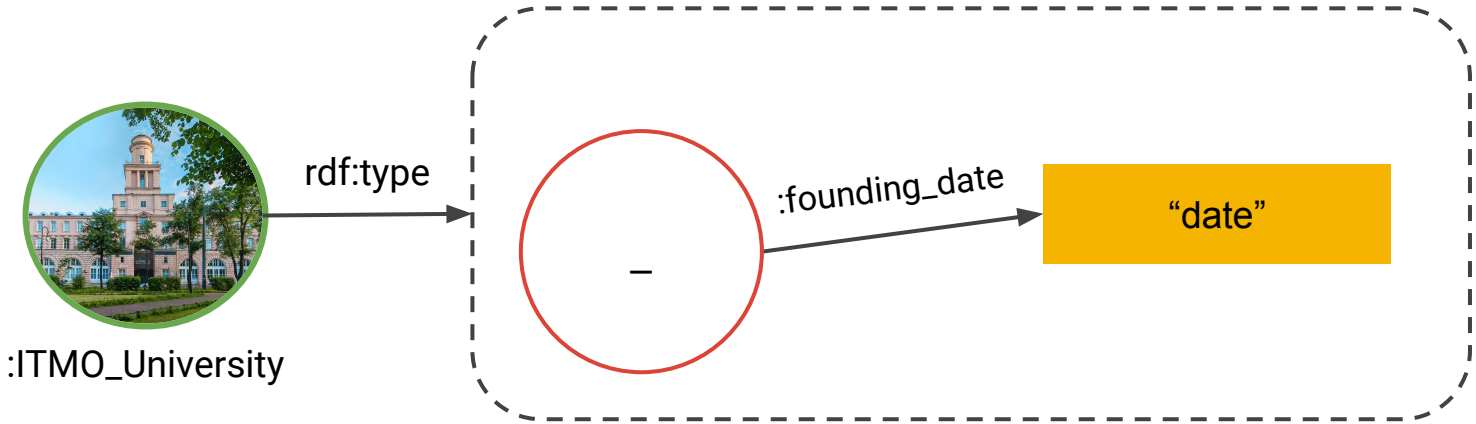
| “строка”@ru | “string”@en | “Die Zeichenfolge”@de | “chaîne”@fr |
|---|---|---|---|

- В расширениях RDF можно создавать собственные литеральные типы (“числа, делящиеся на 2”)

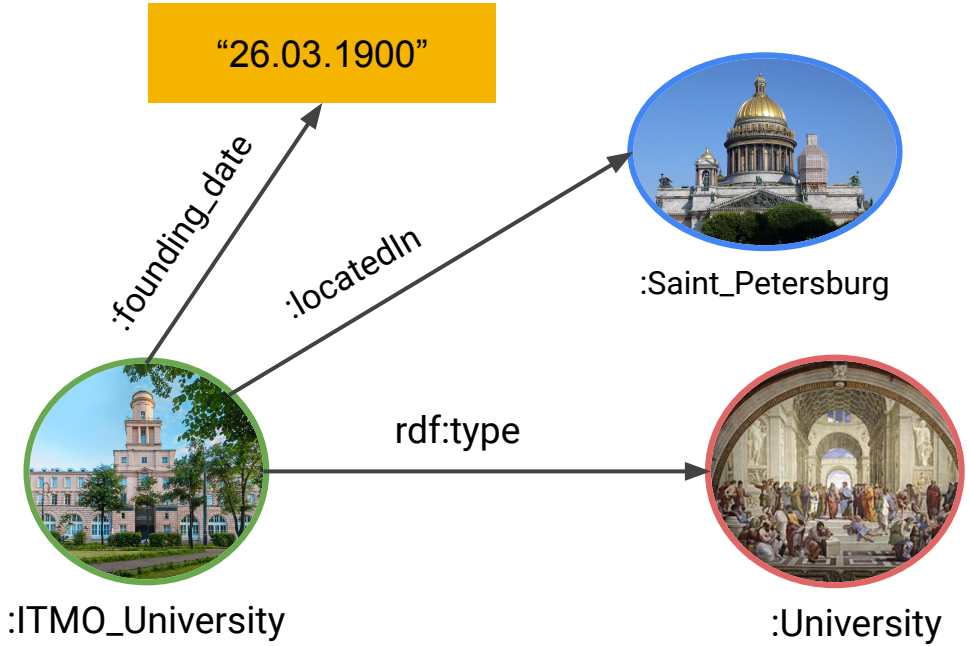
- Литералы не могут быть субъектом RDF-триплета

# RDF: Blank Nodes

- Неименованные вершины - анонимно заданные сущности без URI или литерала
- Используются в т.ч. для логических аксиом и реификации



# RDF Graph




```
<http://example.org/ITMO_University> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://example.org/University> .
<http://example.org/ITMO_University> <http://example.org/founding_date>         “26.03.1900” .
<http://example.org/ITMO_University> <http://example.org/locatedIn>             <http://example.org/Saint_Petersburg> .
```

# RDF Vocabulary

**RDF Vocabulary**

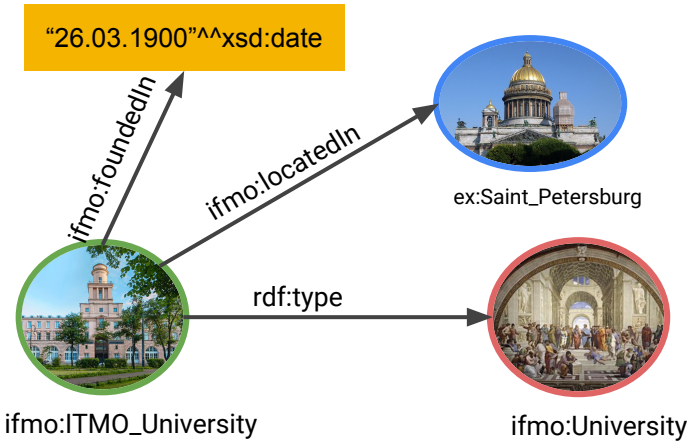
```
rdf:type
rdf:property
rdf:subject
rdf:predicate
rdf:object
rdf:first
rdf:rest
rdf:value
rdf:nil
rdf:list
```

- RDF позволяет:
  - Делать утверждения о принадлежности ресурса к множеству
  - Назначать ресурсам атрибуты-литералы
- Набор ключевых слов - предикатов (RDF vocabulary)
  - Формальная семантика определена в стандарте
- Логическая модель, не зависящая от синтаксиса
  - Но есть популярные сериализации
    - Turtle (.ttl)
    - JSON-LD (.jsonld)
    - XML/RDF (.rdf)
    - N-Triples (.nt)
    - N3 (.n3)
    - TriG (.trig), TriX (.trix)

# Сериализации RDF: Turtle

https://www.w3.org/TR/turtle/

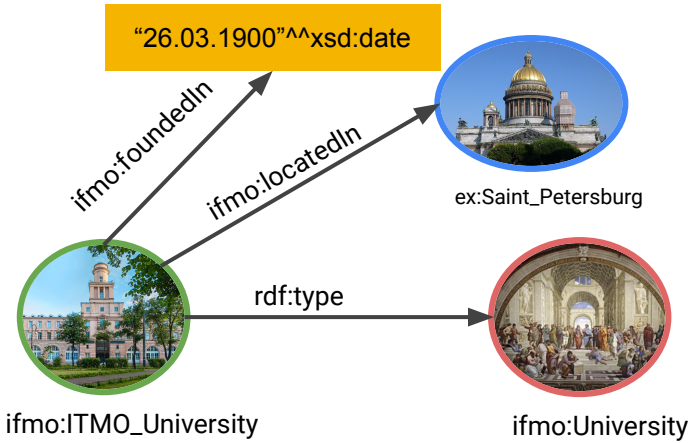
```
@prefix ifmo:  <http://en.ifmo.ru/> .
@prefix ex: <http://example.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ifmo:ITMO_University    rdf:type          ifmo:University ;
                        ifmo:locatedIn    ex:Saint_Petersburg ;
                        ifmo:foundedIn    “26.03.1900”^^xsd:date  .
```



# Сериализации RDF: JSON-LD

https://json-ld.org/spec/latest/json-ld/

```
{
        "@graph" : [ {
          "@id" : "ifmo:ITMO_University",
          "@type" : "ifmo:University",
          "foundedIn" : "26.03.1900",
          "locatedIn" : "ex:Saint_Petersburg"
        } ],
        "@id" : "urn:x-arq:DefaultGraphNode",
        "@context" : {
          "foundedIn" : {
            "@id" : "http://en.ifmo.ru/foundedIn",
            "@type" : "http://www.w3.org/2001/XMLSchema#date"
          },
          "locatedIn" : {
            "@id" : "http://en.ifmo.ru/locatedIn",
            "@type" : "@id"
          },
          "ex" : "http://example.org/",
          "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
          "xsd" : "http://www.w3.org/2001/XMLSchema#",
          "ifmo" : "http://en.ifmo.ru/”
             }
}
```



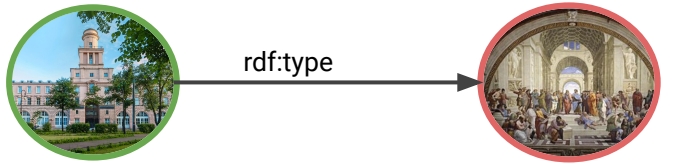
# RDF Schema (RDFS)

**RDFS Vocabulary**

`rdfs:Class`

- RDFS позволяет:
  - Объявлять классы с помощью **`rdfs:Class`**
  - Создавать экземпляры классов с помощью **`rdf:type`**

```
:University   rdf:type  rdfs:Class .
:ITMO         rdf:type :University .
```



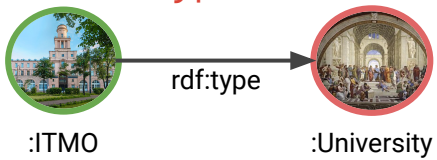
# RDF Schema (RDFS)

**RDFS Vocabulary**

```
rdfs:Class

rdf:Property
rdfs:range
rdfs:domain
```

- RDFS позволяет:
  - Объявлять классы с помощью **rdfs:Class**
  - Создавать экземпляры классов с помощью **rdf:type**

```
:University   rdf:type   rdfs:Class .
:ITMO         rdf:type   :University .
```




  - Объявлять предикаты, их область определения и область значений с помощью **rdf:Property, rdfs:domain, rdfs:range**

```
:locatedIn    rdf:type      rdf:Property .
:locatedIn    rdfs:range    :Place .
:locatedIn    rdfs:domain   :University .
```

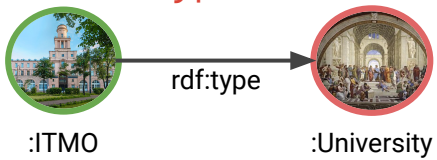
# RDF Schema (RDFS)

**RDFS Vocabulary**

```
rdfs:Class

rdf:Property
rdfs:range
rdfs:domain
```

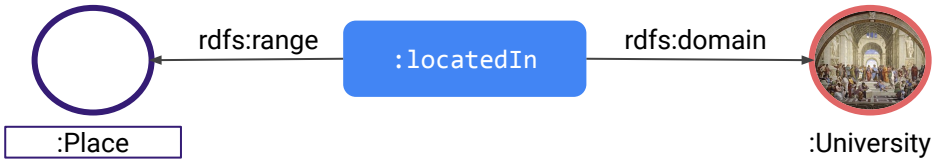
- RDFS позволяет:
  - Объявлять классы с помощью **rdfs:Class**
  - Создавать экземпляры классов с помощью **rdf:type**

```
:University   rdf:type   rdfs:Class .
:ITMO         rdf:type   :University .
```




  - Объявлять предикаты, их область определения и область значений с помощью **rdf:Property, rdfs:domain, rdfs:range**

```
:locatedIn    rdf:type      rdf:Property .
:locatedIn    rdfs:range    :Place .
:locatedIn    rdfs:domain   :University .
```



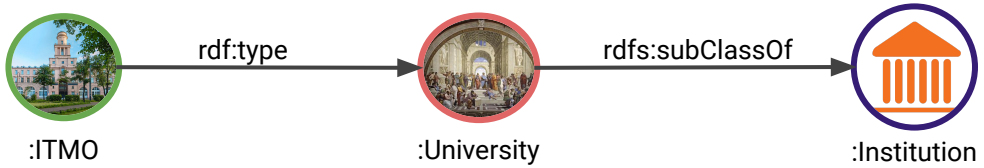
# RDF Schema (RDFS)

**RDFS Vocabulary**

```
rdfs:Class

rdf:Property
rdfs:range
rdfs:domain

rdfs:subClassOf
rdfs:subPropertyOf
```

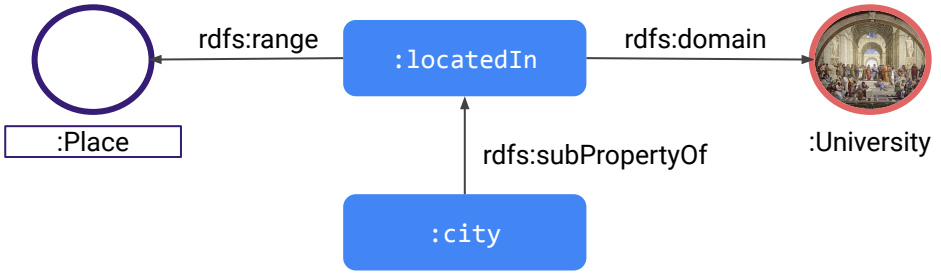
- RDFS позволяет:
  - Создавать иерархию классов с помощью **rdfs:subClassOf**

```
:University   rdfs:subClassOf   :Institution .
```




  - Создавать иерархию предикатов с помощью **rdfs:subPropertyOf**

```
:city     rdfs:subPropertyOf     :locatedIn .
```



# RDF Schema (RDFS)

**RDFS Vocabulary**

```
rdfs:Class

rdf:Property
rdfs:range
rdfs:domain

rdfs:subClassOf
rdfs:subPropertyOf

rdfs:label
rdfs:comment
rdfs:seeAlso
rdfs:isDefinedBy
```

- RDFS позволяет создавать аннотации (не участвуют в логическом выводе):
  - **rdfs:label** - человекочитаемое имя ресурса

    ```
    :ITMO    rdfs:label  “ITMO University”@en .
    :ITMO    rdfs:label  “Университет ИТМО”@ru .
    ```

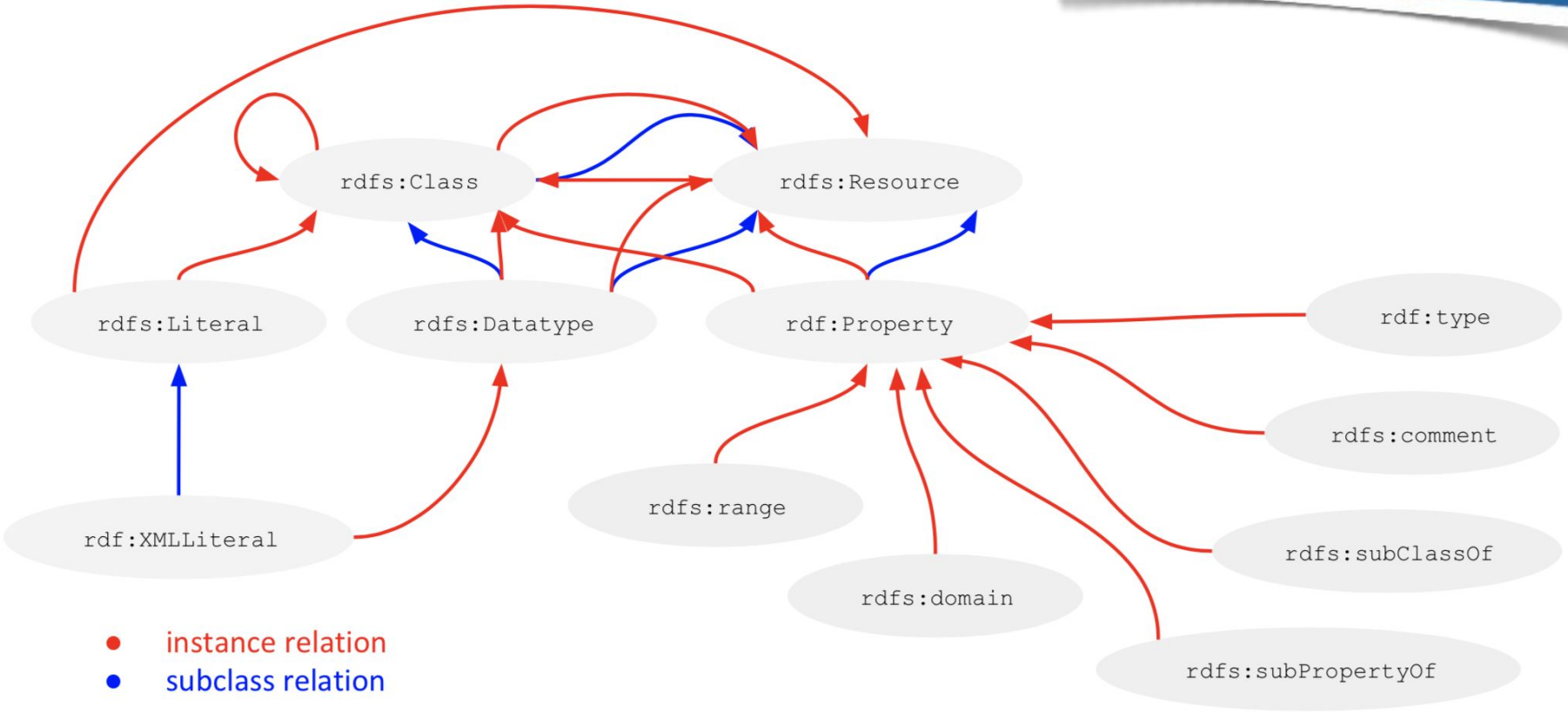
  - **rdfs:comment** - текстовый комментарий

    ```
    :ITMO    rdfs:comment  “A university in Russia” .
    ```

  - **rdfs:seeAlso** - ссылка на объясняющий ресурс

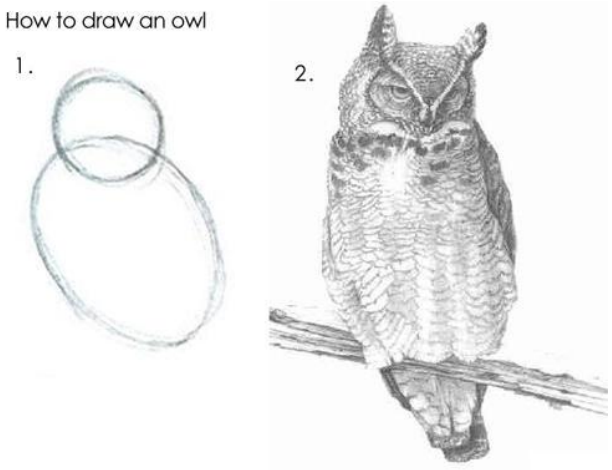
    ```
    :ITMO    rdfs:seeAlso  :Universities_in_Russia .
    ```

# RDFS Language Model




http://www.w3.org/TR/rdf-schema/#ch_classes

# OWL - Web Ontology Language




- OWL - еще более выразительное надмножество RDFS
  - Основан на дескрипционных логиках
  - OWL 1 (2004) $\mathcal{SHOIN}(\mathcal{D})$
  - OWL 2 (2009) $\mathcal{SROIQ}(\mathcal{D})$
  - Классы, предикаты и экземпляры классов (почти как в RDFS)
  - Гипотеза об открытом мире - отсутствие информации не говорит о ее истинности или ложности

    `:Alice :knows :Bob .`

    Не значит, что **только** Алиса знает Боба
  - Гипотеза уникальных имен не выполняется - различия нужно указывать отдельно

# Description Logics

- Логика первого порядка алгоритмически полуразрешима
  - Существует алгоритм, способный за *конечное время* подтвердить истинность некоторого высказывания в данной теории, а иначе может работать бесконечно долго.
- Экспрессивность ↑ - разрешимость ↓
- Получение разрешимых логик - хотя бы $O(e^n)$ - требует уменьшать экспрессивность
- Дескрипционные логики позволяют получить довольно экспрессивные, но алгоритмически разрешимые теории

# Description Logics

- Логика первого порядка алгоритмически полуразрешима
  - Существует алгоритм, способный за *конечное время* подтвердить истинность некоторого высказывания в данной теории, а иначе может работать бесконечно долго.
- Экспрессивность ↑ - разрешимость ↓
- Получение разрешимых логик - хотя бы $O(e^n)$ - требует уменьшать экспрессивность
- Дескрипционные логики позволяют получить довольно экспрессивные, но алгоритмически разрешимые теории

  OWL EL, OWL RL, OWL QL ⊆ OWL 2 DL ⊆ OWL 2 Full ⊆ FOL

# OWL 1 $\mathcal{SHOIN}(\mathcal{D})$

- Аксиомы
  - **TBox**: подклассы $C \sqsubseteq D$ **(H)**
  - **RBox**: иерархия предикатов $R \sqsubseteq S$ **(H)**,
    инверсные предикаты $R^-$ **(I)**, транзитивность предикатов $\sqsubseteq^+$ **(S)**
  - **ABox**: факты о классах $C(a)$, предикатах $R(a, b)$,
    эквивалентность $(a=b)$, различие $(a \neq b)$
- Составные классы
  - Конъюнкция $C \sqcap D$, дизъюнкция $C \sqcup D$, отрицание $\neg C$ классов
  - Кванторы существования $\exists R.C$ и всеобщности $\forall R.C$
  - Ограничения на количество предикатов $\leq n\ R$, $\geq n\ R$ **(N)**
  - Номинальные классы $C = \{a\}$ **(O)**
- Типы данных **(D)**

# OWL 2 $\mathcal{SROIQ}(\mathcal{D})$

- Составные классы
  - Имена классов $A, B$
  - Конъюнкция $C \sqcap D$,
  - Дизъюнкция $C \sqcup D$,
  - Отрицание $\neg C$ классов
  - Кванторы существования $\exists R.C$
  - Кванторы всеобщности $\forall R.C$
  - Self $\exists S.Self$
  - Количество предикатов $\leq n\ R.C$ **(Q)**
  - Количество предикатов $\geq n\ R.C$ **(Q)**
  - Номинальные классы $C = \{a\}$ **(O)**

- Предикаты
  - Имена предикатов $R, S, T$
  - Простые предикаты $S, T$
  - Инверсные предикаты $R^-$
  - Универсальные предикаты $U$

- Аксиомы о классах (**TBox**)
  - Подклассы $C \sqsubseteq D$
  - Эквивалентность $C \equiv D$

- Аксиомы о предикатах (**RBox**)
  - Иерархия предикатов $R_1 \sqsubseteq R_2$
  - Цепи $R^{(-)}_1 \circ R^{(-)}_2 \circ \ldots \circ R^{(-)}_n \sqsubseteq R$
  - Транзитивность $\sqsubseteq^+$ **(S)**
  - Симметричность
  - Рефлексивность **(R)**
  - Иррефлексивность **(R)**
  - Несовместность

- Факты (**ABox**)
  - Факты о классах $C(a)$,
  - Факты о предикатах $R(a, b)$,
  - Негативные факты о предикате $\neg R(a, b)$
  - Эквивалентность $(a=b)$,
  - Различие $(a \neq b)$

# OWL 2 Manchester Syntax

$$\text{Musician} \equiv \exists \text{writes.Song} \sqcap \forall \text{writes.Song}$$

```
Class: Musician
    EquivalentTo:
        (writes some Song)
         and (writes only Song)
```

# OWL 2 Turtle

$$\text{Musician} \equiv \exists \text{writes.Song} \sqcap \forall \text{writes.Song}$$

```
:Musician rdf:type owl:Class ;
          owl:equivalentClass [ owl:intersectionOf    (
                    [    rdf:type owl:Restriction ;
                         owl:onProperty :writesSongs ;
                         owl:someValuesFrom :Song ]
                    [    rdf:type owl:Restriction ;
                         owl:onProperty :writesSongs ;
                         owl:allValuesFrom :Song]     ) ;
                    rdf:type owl:Class ] .
```

Blank node

# Классы в OWL

:University

- Простые или сложные (задаются аксиомой)

```
:University rdf:type owl:Class
```

- Два уже определенных класса
  - `owl:Thing` - все экземпляры всех классов
  - `owl:Nothing` - пустое множество

$\top \equiv C \sqcup \neg C$

$\bot \equiv C \sqcap \neg C$

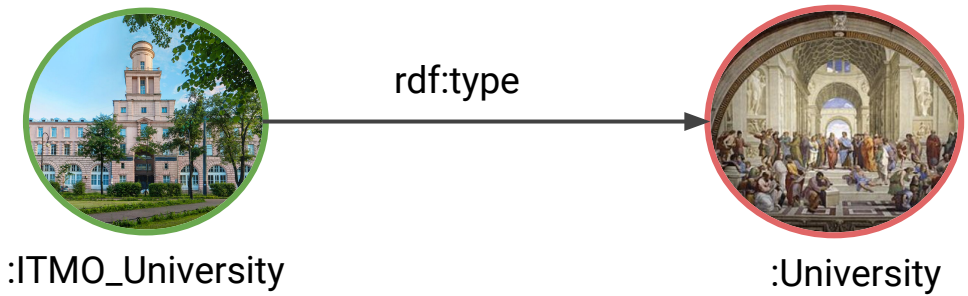
- Сложные классы

$$Musician \equiv \exists writes.Song \sqcap \forall writes.Song$$

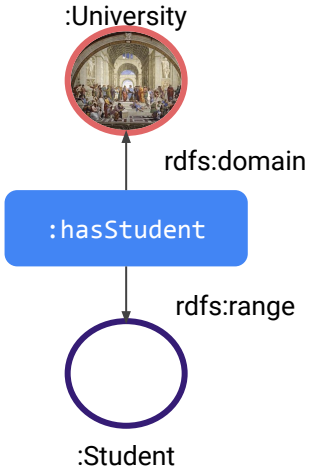
# Экземпляры в OWL

- Задаются через `owl:NamedIndividual`

```
:ITMO_University rdf:type :University .
:ITMO_University rdf:type owl:NamedIndividual .
```



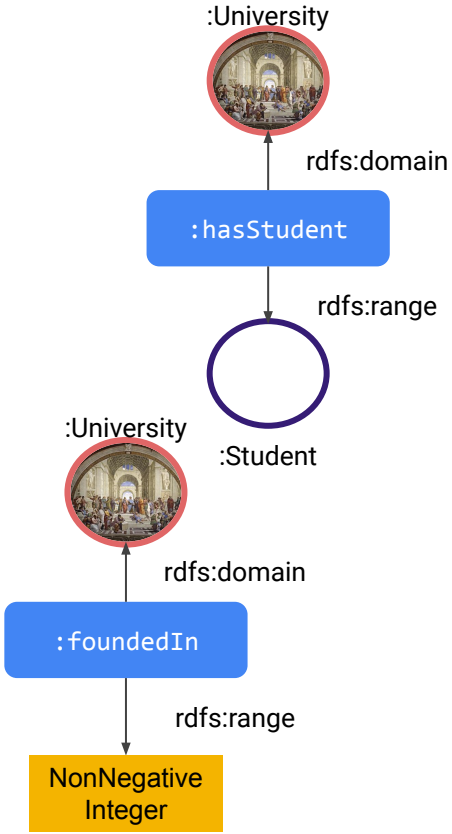
# Предикаты в OWL




- Объектные: значение - ресурс

```
:hasStudent rdf:type     owl:ObjectProperty ;
            rdfs:range   :Student ;
            rdfs:domain  :University .
```

# Предикаты в OWL



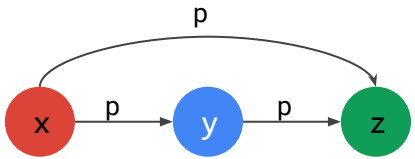

- Объектные: значение - ресурс

```
:hasStudent rdf:type     owl:ObjectProperty ;
            rdfs:range   :Student ;
            rdfs:domain :University .
```

- Литеральные: значение - литерал

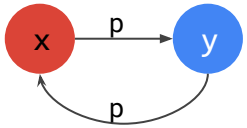
```
:foundedIn rdf:type     owl:DatatypeProperty ;
           rdfs:range   xsd:NonNegativeInteger ;
           rdfs:domain :University .
```

# Объектные предикаты в OWL



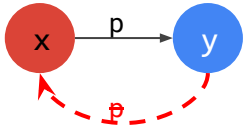

- Транзитивность `owl:TransitivePropery`

  `TransitiveProperty(p),`
  `p(x,y) ⊓ p(y,z) ⊨ p(x,z)`



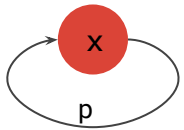

- Симметричность `owl:SymmetricProperty`

  `SymmetricProperty(p), p(x,y) ⊨ p(y,x)`




- Асимметричность `owl:AsymmetricProperty`

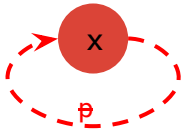
  `AsymmetricProperty(p), p(x,y) ⊭ p(y,x)`



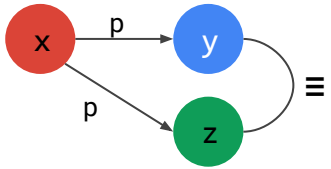

- Рефлексивность `owl:ReflexiveProperty`

  `ReflexiveProperty(p) ⊨ p(x,x)`

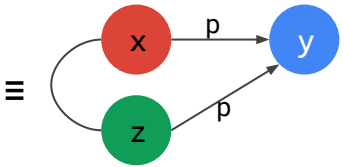
# Объектные предикаты в OWL



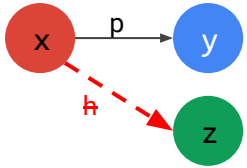

- Иррефлексивность `owl:IrreflexiveProperty`
  `IrreflexiveProperty(p)` ⊨ `p(x,x)`




- Функциональность `owl:FunctionalProperty`
  `FunctionalProperty(p),`
  `p(x,y)` ⊓ `p(x,z)` ⊨ `y≡z [sameAs(y,z)]`

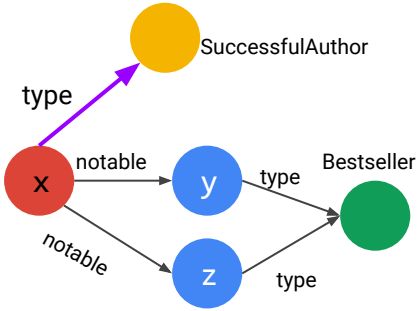



- Инверсная функциональность `owl:InverseFunctionalProperty`
  `InverseFunctionalProperty(p),`
  `p(x,y)` ⊓ `p(z,y)` ⊨ `x≡z [sameAs(x,z)]`




- Несовместность `owl:propertyDisjointWith`
  `propertyDisjointWith(p,h),`
  `p(x,y)` ⊓ `h(x,z)` ≡ ⊥

# OWL Restrictions & Rules

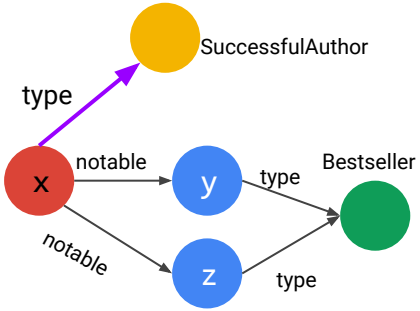



- Квалифицированные ограничения

  $\mathtt{SuccessfulAuthor} \sqsubseteq \geq 1\ \mathtt{notableWork.Bestseller}$

```
:SuccessfulAuthor a owl:Class ;
      rdfs:subClassOf [
            a owl:Restriction;
            owl:onProperty :notableWork;
            owl:minQualifiedCardinality 1;
            owl:onClass :Bestseller ] .
```

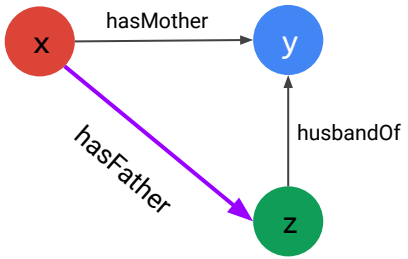
# OWL Restrictions & Rules




- Квалифицированные ограничения

  $SuccessfulAuthor \sqsubseteq \geq 1\ notableWork.Bestseller$

```
:SuccessfulAuthor a owl:Class ;
      rdfs:subClassOf [
            a owl:Restriction;
            owl:onProperty :notableWork;
            owl:minQualifiedCardinality 1;
            owl:onClass :Bestseller ] .
```

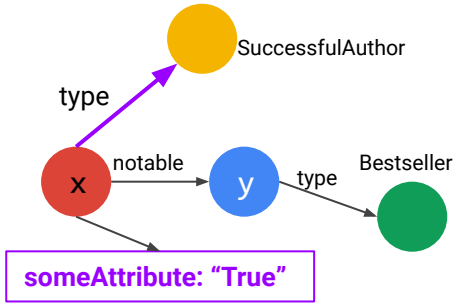



- Отношения над предикатами

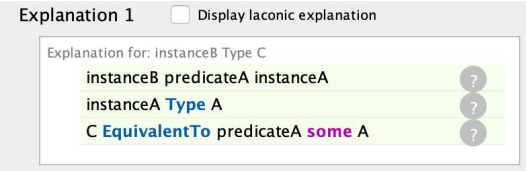
  $hasMother(x,y) \sqcap husbandOf(z,y) \models hasFather(x,z)$

- Правила “если-то” (продукции)
  - Стандарты SWRL, SPIN

# Reasoning - логический вывод новых фактов

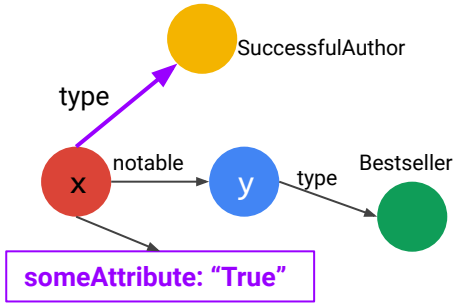



- Вход: RDF-граф
  Выход: RDF-граф с новыми триплетами
  - Новые атрибуты вершин
  - Новые ребра между вершинами

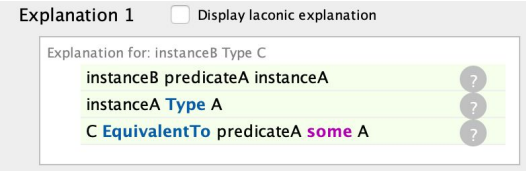



- Логическое обоснование каждому выведенному факту (чего не может статистическо-вероятностный ML)

# Reasoning - логический вывод новых фактов




- Вход: RDF-граф
  Выход: RDF-граф с новыми триплетами
  - Новые атрибуты вершин
  - Новые ребра между вершинами




- Логическое обоснование каждому выведенному факту (чего не может статистическо-вероятностный ML)
- Время работы быстро растет от размера графа
- Актуальный ресерч:
  Объединить объяснимость символьных вычислений и скорость вероятностных

# Граф знаний

ABox (Assertion Box)
Наполнение графа
Факты графа знаний

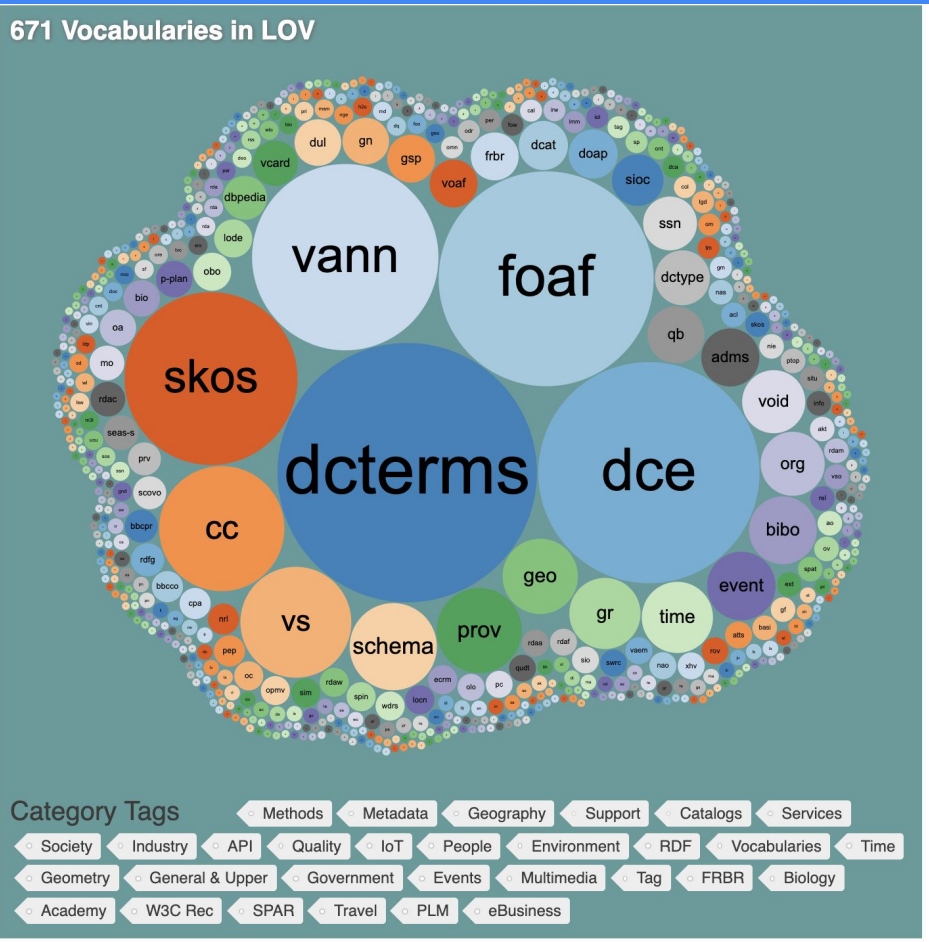
➔ Создание как правило через семантическую интеграцию существующих источников (СУБД, неструктурированные данные) с помощью схемы графа

TBox (Terminology Box)
Схема данных графа
Модель предметной области

Онтология:

➔ “формализованная модель некоторой области знаний, согласованная с экспертами этой области”
➔ Итеративное (вручную) или автоматизированное создание

# Онтологии в Linked Open Data




Крупнейшие онтологии:

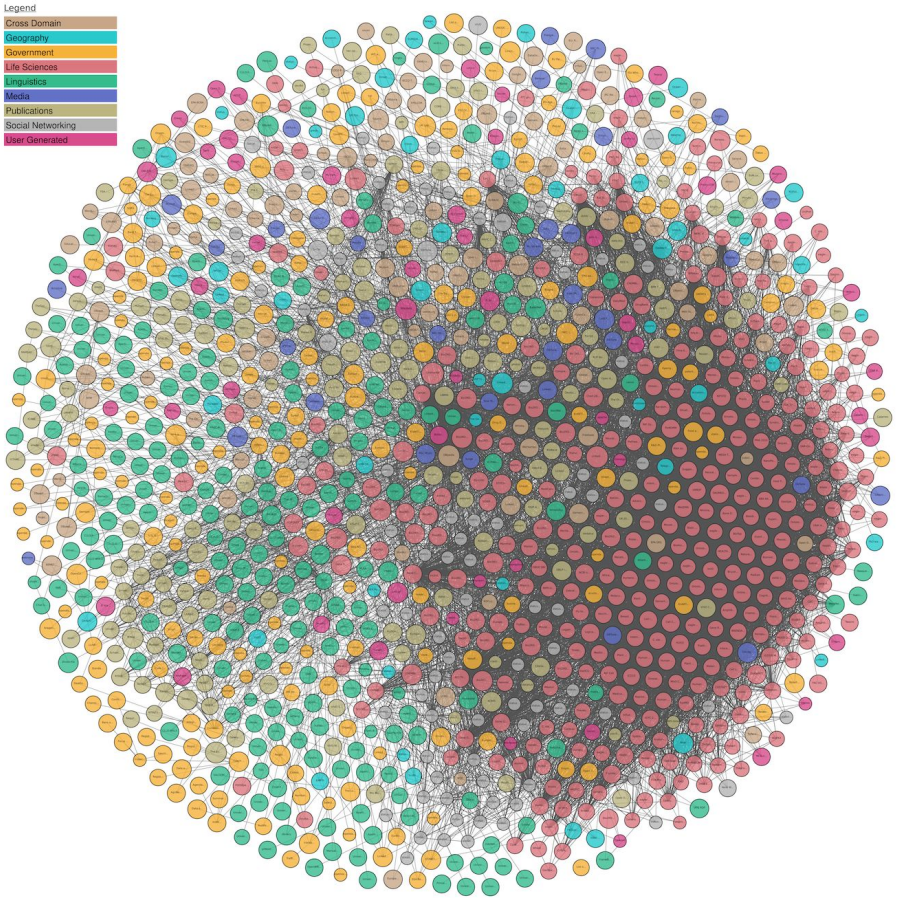
- SNOMED-CT - медицина
- FIBO - финансы
- **DBpedia Ontology - из Википедии**
- **YAGO - из Википедии**
- **Wikidata Ontology ~ из Википедии**

Мета-онтологии:

- Dublin Core Terms (dcterms)
- SKOS
- DCAT
- RDF DataCube

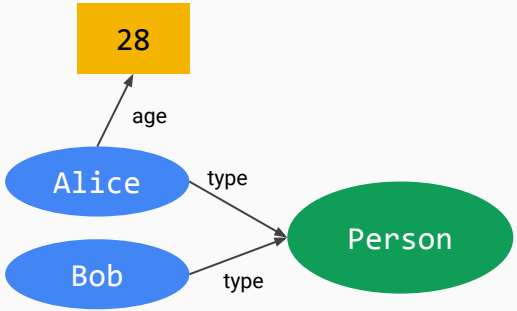
https://lov.linkeddata.es/dataset/lov

# Linked Open Data - 2020



# Open World Assumption

- Изначально неполная картина мира
- Все, что не задано (как истина) - *возможно*, истина
- Онтологии могут быть неполными by design для расширения




# Closed World Assumption

- Задается все возможное знание
- Все, что не задано (как истина) - ложь
- Нужен источник истины, например
  - Колонка в БД
  - Поле объекта
  - Слот во фрейме

| Person | Age |
|---|---|
| Alice | 28 |
| Bob | N/A |

# В следующей серии

1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. **Хранение знаний в графах - SPARQL & Graph Databases**
4. Однородность знаний - Reification & RDF* & SHACL & ShEx
5. Интеграция данных в графы знаний - Semantic Data Integration
6. Векторные представления графов - Knowledge Graph Embeddings
7. Введение в теорию графов - Graph Theory Intro
8. Машинное обучение на графах - Graph Neural Networks & KGs
9. Некоторые применения - Question Answering & Query Embedding