

# Графы знаний

Лекция 4 - Однородность знаний в графах

М. Галкин, Д. Муромцев



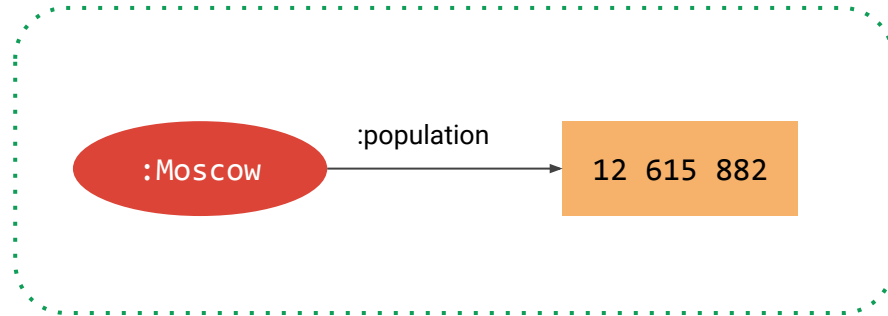
# Сегодня

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

# Содержание

- Часть 1: Trust & Provenance
- Часть 1: Реификация
  - 6 схем, RDF\*
  - Сравнение производительности
- Часть 1: Модель данных Wikidata
- Часть 2: Валидация
  - SHACL
  - ShEx
  - Сходства и различия

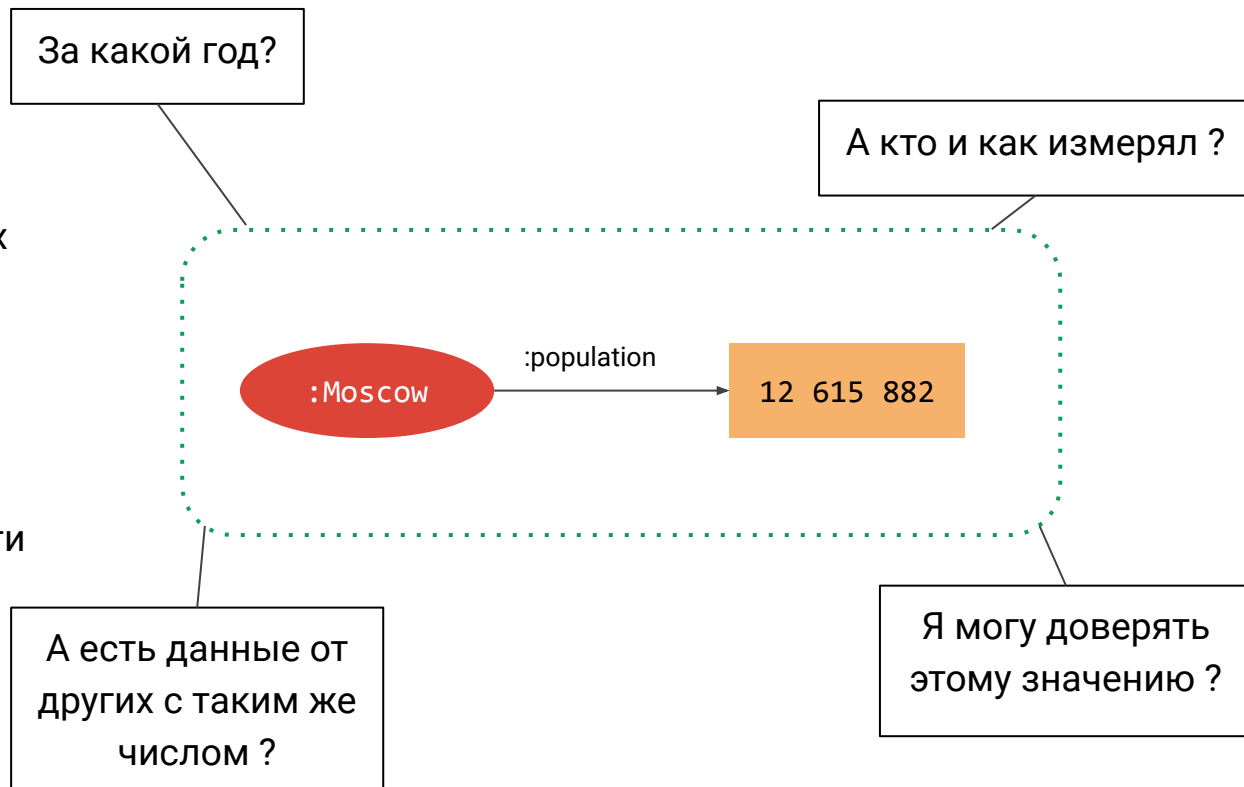
# Trust & Provenance



# Trust & Provenance

**Метаданные** в графах знаний важны для обеспечения:

- Достоверности
- Целостности
- Доверительности



# Trust & Provenance - Wikidata example

Предикат может  
детализироваться  
дополнительными  
значениями

N-арные  
предикаты?

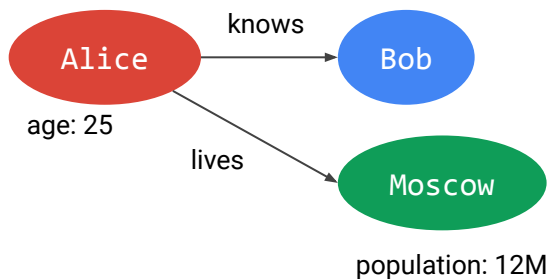
population	12,615,882	
point in time		2019
▼ 2 references		
reference URL	<a href="http://www.gks.ru/free_doc/new_site/population/demo/Popul2019.xls">http://www.gks.ru/free_doc/new_site/population/demo/Popul2019.xls</a>	
imported from Wikimedia project	Russian Wikipedia	
Wikimedia import URL	<a href="https://ru.wikipedia.org/?oldid=98816600">https://ru.wikipedia.org/?oldid=98816600</a>	

population	1,174,700±100	
point in time		1902
▼ 0 references		

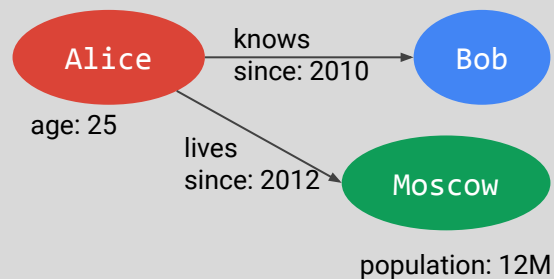
# Графовые СУБД: RDF vs LPG

## RDF



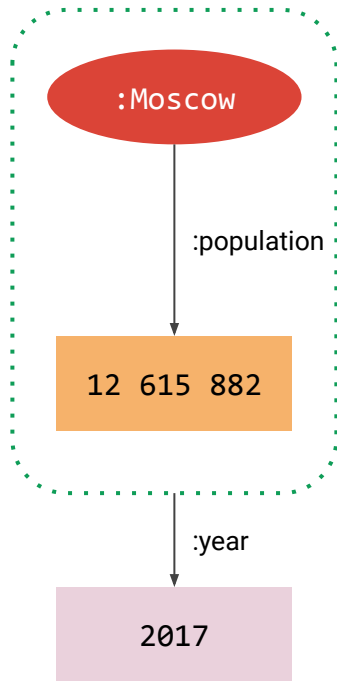
- Атрибуты предикатов ограничены RDFS/OWL
- Схема графа - семантическая
- Задание метаданных о конкретном триплете нетривиально

## LPG (Labeled Property Graph)



- Атрибуты предикатов не ограничены
- Схема графа - не семантическая
- В атрибуты предикатов можно поместить метаданные

# Реификация (RDF Reification)

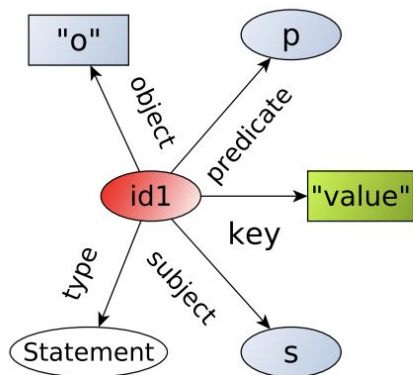


Реификация высказывания - описание высказывания (триплета) с помощью других высказываний

- Высказывания о высказываниях - логика высших порядков
- Тривиально для LPG
- Несколько способов для RDF



# RDF Reification - Standard Reification

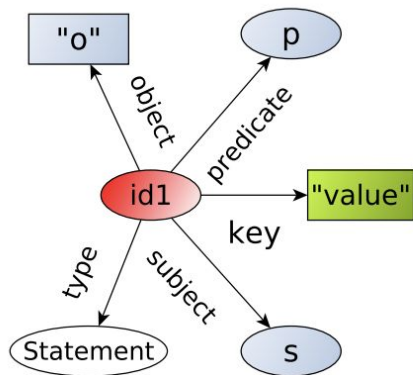


Стандарт RDF Reification предоставляет набор специальных предикатов:

- `rdf:Statement` - тип новой вершины
- `rdf:subject`, `rdf:predicate`, `rdf:object` - ссылки на составляющие триплета

```
<statementID> a rdf:Statement ;  
    rdf:subject    dbr:Moscow;  
    rdf:predicate  dbo:population;  
    rdf:object     1174700;  
    dbo:year       1902 .
```

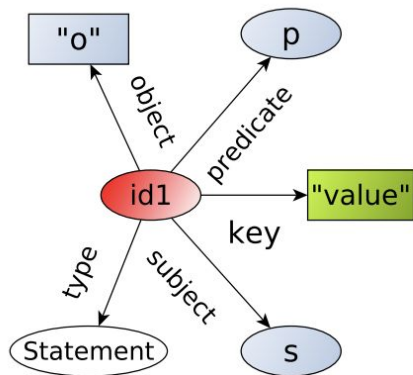
# RDF Reification - Standard Reification - Issues



```
<statementID> a rdf:Statement ;  
    rdf:subject    dbr:Moscow;  
    rdf:predicate  dbo:population;  
    rdf:object     1174700;  
    dbo:year       1902 .
```

- **Высказывание о триплете не объявляет существование триплета**  
**dbr:Moscow dbo:population 1174700 .**

# RDF Reification - Standard Reification - Issues



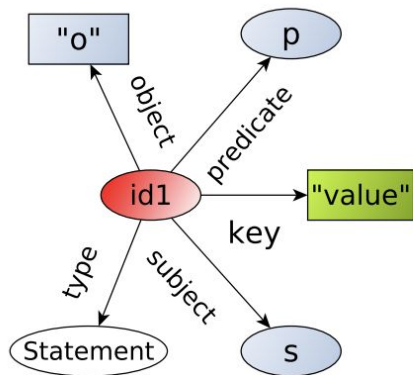
```
<statementID> a rdf:Statement ;  
    rdf:subject    dbr:Moscow;  
    rdf:predicate  dbo:population;  
    rdf:object     1174700;  
    dbo:year       1902 .
```

- **Высказывание о триplete не объявляет существование триплета**  
**dbr:Moscow dbo:population 1174700 .**

```
SELECT ?population WHERE {  
    dbr:Moscow dbo:population ?population }  
empty
```

```
SELECT ?population WHERE {  
    ?statement    rdf:subject    dbr:Moscow ;  
    rdf:predicate  dbo:population ;  
    rdf:object     ?population . }  
1174700
```

# RDF Reification - Standard Reification - Issues



- Высказывание о триплете не объявляет существование триплета  
`dbr:Moscow dbo:population 1174700 .`
- Явное объявление высказываний без реификации может нарушить логическую целостность

`dbr:Moscow dbo:population 1174700 . # year 1902`  
`dbr:Moscow dbo:population 12615882 . # year 2017`

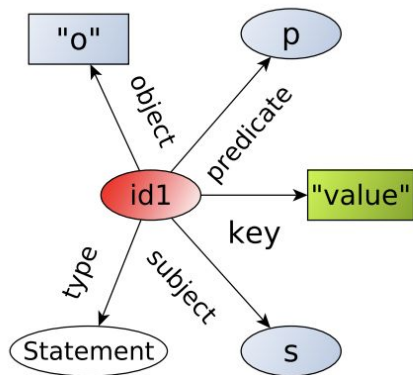
```
<statementID1> a rdf:Statement ;
```

```
    rdf:subject      dbr:Moscow;  
    rdf:predicate    dbo:population;  
    rdf:object       1174700;  
    dbo:year         1902 .
```

```
<statementID2> a rdf:Statement ;
```

```
    rdf:subject      dbr:Moscow;  
    rdf:predicate    dbo:population;  
    rdf:object       12615882;  
    dbo:year         2019 .
```

# RDF Reification - Standard Reification - Issues



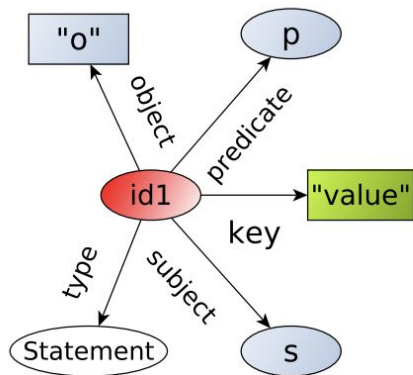
- Высказывание о триплете не объявляет существование триплета  
dbr:Moscow dbo:population 1174700 .
- Явное объявление высказываний без реификации может нарушить логическую целостность
  - Измененный шаблон SPARQL-запросов

```
SELECT ?population ?year WHERE {  
  ?statement      rdf:subject  
                  rdf:predicate  
                  rdf:object  
                  dbo:year  
ORDER BY ASC (?year)
```

```
dbr:Moscow ;  
dbo:population ;  
?population ;  
?year . }
```

?population	?year
1174700	1902
12615882	2019

# RDF Reification - Standard Reification - Issues



- Высказывание о триплете не объявляет существование триплета  
`dbr:Moscow dbo:population 1174700 .`
- Явное объявление высказываний без реификации может нарушить логическую целостность
  - Измененный шаблон SPARQL-запросов
- Обозначение актуальных значений

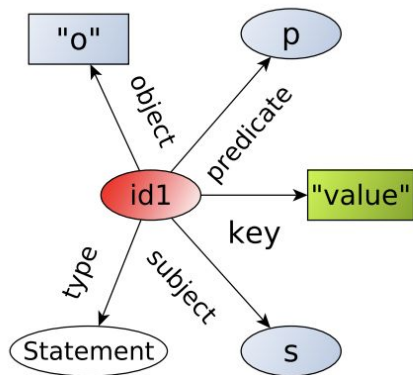
```
<statementID1> a rdf:Statement ;
```

```
    rdf:subject      dbr:Moscow;  
    rdf:predicate    dbo:population;  
    rdf:object       1174700;  
    dbo:year         1902 .
```

```
<statementID2> a rdf:Statement ;
```

```
    rdf:subject      dbr:Moscow;  
    rdf:predicate    dbo:population;  
    rdf:object       12615882;  
    dbo:year         2019 .
```

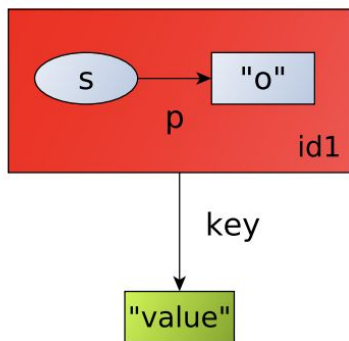
# RDF Reification - Standard Reification - Issues



- Высказывание о триплете не объявляет существование триплета  
`dbr:Moscow dbo:population 1174700 .`
- Явное объявление высказываний без реификации может нарушить логическую целостность
  - Измененный шаблон SPARQL-запросов
- Обозначение актуальных значений
- Физический размер графа быстро растет
  - 4+ триплета на высказывание
  - Искусственные вершины в графе (statement ids)

# RDF Reification - Named Graphs

Стандарты RDF и SPARQL 1.1 поддерживают именованные графы



- Каждый триплет оборачивается в собственный named graph
- Атрибуты и пары key - value ассоциируются с именованным графом

```
<graph1> { dbr:Moscow dbo:population 1174700 . }
```

```
<graph2> { dbr:Moscow dbo:population 12615882 . }
```

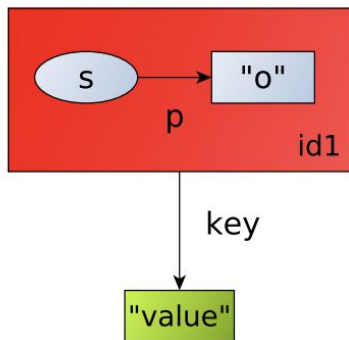
```
<graph1> dbo:year 1902 .
```

```
<graph2> dbo:year 2019 .
```



# RDF Reification - Named Graphs

Стандарты RDF и SPARQL 1.1 поддерживают именованные графы



- Каждый триплет оборачивается в собственный named graph
- Атрибуты и пары key - value ассоциируются с именованным графом

```
<graph1> { dbr:Moscow dbo:population 1174700 . }
```

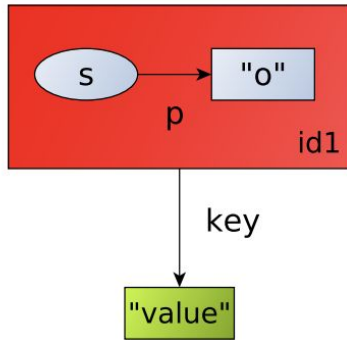
```
<graph2> { dbr:Moscow dbo:population 12615882 . }
```

```
<graph1> dbo:year 1902 .
```

```
<graph2> dbo:year 2019 .
```

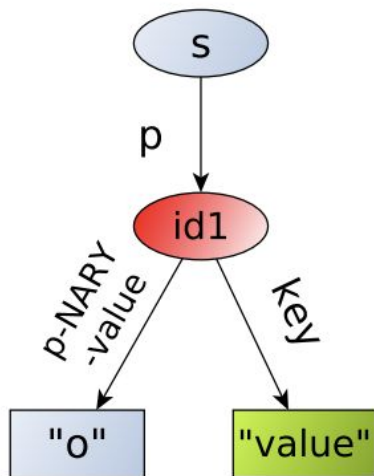
```
SELECT ?population ?year WHERE {  
  GRAPH ?g { dbr:Moscow dbo:population ?population } .  
  ?g dbo:year ?year .  
} ORDER BY ASC (?year)
```

# RDF Reification - Named Graphs - Issues



- + Именованные графы - часть стандартов
- + Экономия места без введения новых вершин в граф
- Невозможность работы с графами, которые в своей логической модели уже используют named graphs

# RDF Reification - N-ary Predicates

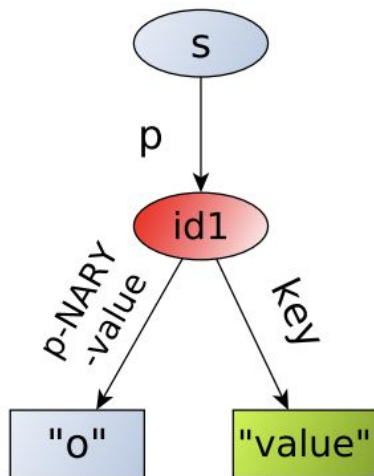


- Объект триплета заменяется новой вершиной, которая содержит оригинальное значение
- Искусственно создается p-nary-value предикат и другие пары предикат-значение

```
dbr:Moscow dbo:population <node1> .  
<node1> dbo:population-value 1174700 .  
<node1> dbo:year 1902 .
```

```
dbr:Moscow dbo:population <node2> .  
<node2> dbo:population-value 12615882 .  
<node2> dbo:year 2019 .
```

# RDF Reification - N-ary Predicates



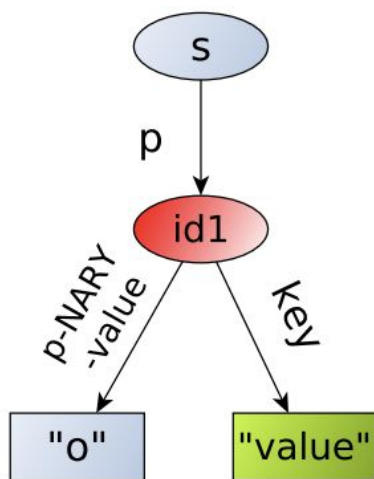
- Объект триплета заменяется новой вершиной, которая содержит оригинальное значение
- Искусственно создается p-nary-value предикат и другие пары предикат-значение

```
dbr:Moscow dbo:population <node1> .  
<node1> dbo:population-value 1174700 .  
<node1> dbo:year 1902 .
```

```
dbr:Moscow dbo:population <node2> .  
<node2> dbo:population-value 12615882 .  
<node2> dbo:year 2019 .
```

```
SELECT ?population ?year WHERE {  
  dbr:Moscow dbo:population ?temp .  
  ?temp dbo:population-value ?population .  
  ?temp dbo:year ?year .} ORDER BY ASC (?year)
```

# RDF Reification - N-ary Predicates

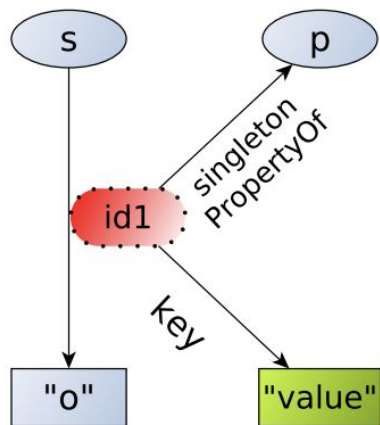


- + Один новый триплет на факт
- + Можно работать с именованными графами в исходном графе

- Ввод новой вершины и предиката может нарушить семантическую целостность:  
dbo:population становится объектным свойством
- Каждый предикат дублируется схожим predicate-value

```
dbr:Moscow dbo:population <node1> .  
<node1> dbo:population-value 1174700 .  
<node1> dbo:year 1902 .
```

# RDF Reification - Singleton Property

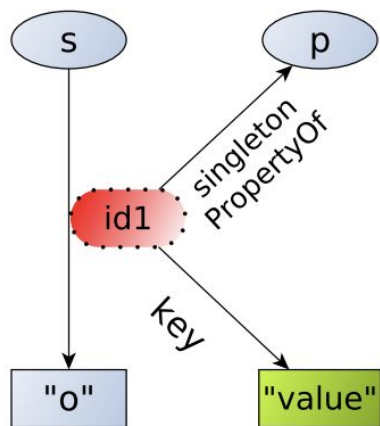


- Для каждого высказывания создается уникальный предикат (с собственным идентификатором)
- Уникальные предикат содержит ссылку на оригинальный предикат (`rdf:singletonPropertyOf`) и пары предикат-значение

```
dbr:Moscow <predicate1> 1174700 .  
<predicate1> rdf:singletonPropertyOf dbo:population .  
<predicate1> dbo:year 1902 .
```

```
dbr:Moscow <predicate2> 12615882 .  
<predicate2> rdf:singletonPropertyOf dbo:population .  
<predicate2> dbo:year 2019 .
```

# RDF Reification - Singleton Property



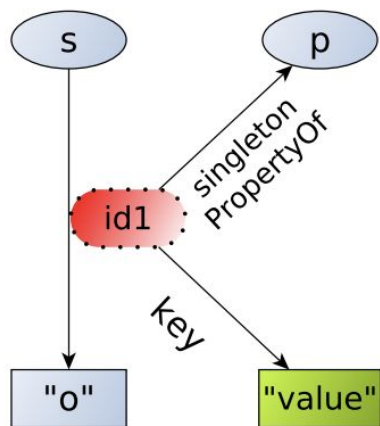
- Для каждого высказывания создается уникальный предикат (с собственным идентификатором)
- Уникальный предикат содержит ссылку на оригинальный предикат (`rdf:singletonPropertyOf`) и пары предикат-значение
- `rdf:singletonPropertyOf` необходим для логического вывода оригинального высказывания (через RDFS).

```
dbr:Moscow <predicate1> 1174700 .  
<predicate1> rdf:singletonPropertyOf dbo:population .  
<predicate1> dbo:year 1902 .
```

```
SELECT ?population ?year WHERE {  
  dbr:Moscow ?singleton ?population .  
  ?singleton rdf:singletonPropertyOf dbo:population .  
  ?singleton dbo:year ?year .} ORDER BY ASC (?year)
```

# RDF Reification - Singleton Property

+ Можно работать с именованными графами

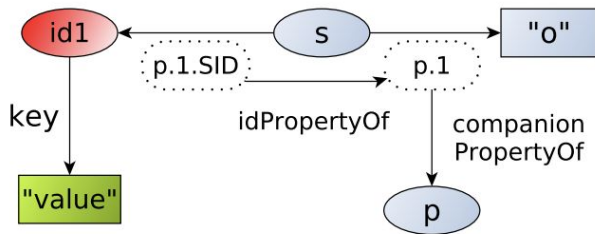


- У каждого триплета уникальный предикат
- Усложнение индексов для SPARQL
- Усложнение логического вывода
- Изменение распределения предикатов по сущностям - усложнение задачи предсказания связи между двумя сущностями



# RDF Reification - Companion Property

- Вводит уникальные предикаты для уникальных субъектов
- Соглашение об именах:
  - Каждому использованию предиката  $p$  с одинаковым  $s$  назначается растущий  $id$
  - У каждого уникального предиката есть  $sID$  с метаданными и парами предикат-значение



```
dbr:Moscow      dbo:population.1 1174700 ;  
                dbo:population.1.SID <statement1> ;  
                dbo:population.2 12615882 ;  
                dbo:population.2.SID <statement2> .
```

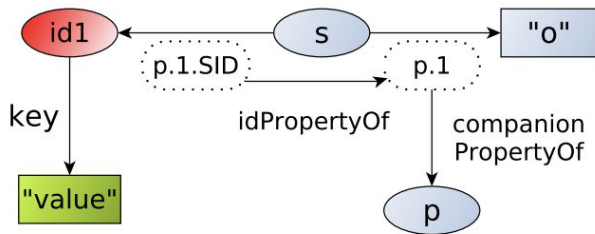
```
<statement1> dbo:year 1902 .
```

```
<statement2> dbo:year 2019 .
```

```
dbo:population.1.SID rdf:idPropertyOf dbo:population.1 .  
dbo:population.1 rdf:companionPropertyOf dbo:population .  
dbo:population.2.SID rdf:idPropertyOf dbo:population.2 .  
dbo:population.2 rdf:companionPropertyOf dbo:population .
```

# RDF Reification - Companion Property

- Вводит уникальные предикаты для уникальных субъектов
- Соглашение об именах:
  - Каждому использованию предиката  $p$  с одинаковым  $s$  назначается растущий  $id$
  - У каждого уникального предиката есть  $sID$  с метаданными и парами предикат-значение



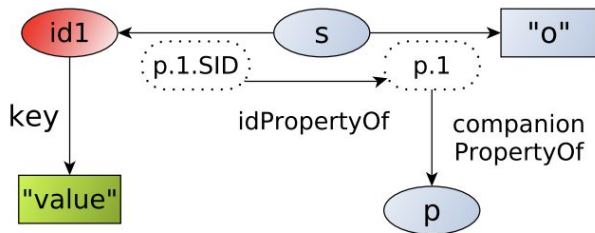
```
SELECT ?population ?year WHERE {
  { dbr:Moscow ?companion ?population .
    ?companion rdf:companionPropertyOf dbo:population
    ?companionID rdf:idPropertyOf ?companion . }

  { ?companionID dbo:year ?year . }
} ORDER BY ASC (?year)
```

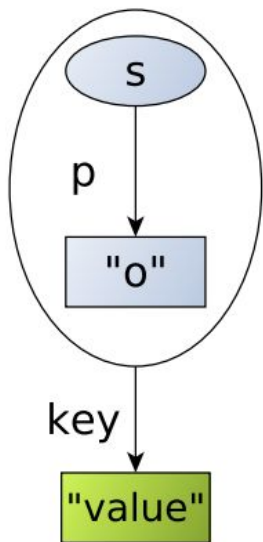
# RDF Reification - Companion Property

+ По сравнению с Singleton Property сгенерирует меньше уникальных предикатов

- Сложность моделирования
- Нарушения распределения предикатов и однородности графа
- Большое число вспомогательных триплетов для логической целостности



# Reification Done Right (RDR) -> RDF\*



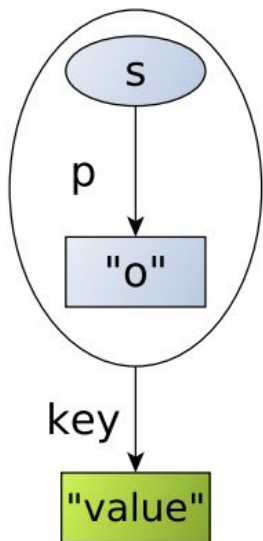
- Простая схема реификации, предложенная разработчиками СУБД Blazegraph
- Использует новый синтаксис RDF\* и SPARQL\*
- Транслируется в любую предыдущую схему
- Имеет формальную модель RDF\*-графа
- Новейший драфт спецификации от 18.02.2021

```
<<db: Moscow dbo:population 1174700>> dbo:year 1902 .  
<<db: Moscow dbo:population 12615882>> dbo:year 2019 .
```

<https://w3c.github.io/rdf-star/rdf-star-cg-spec.html>

<https://blog.liu.se/olafhartig/2019/01/10/position-statement-rdf-star-and-sparql-star/>

# RDF\* / SPARQL\*

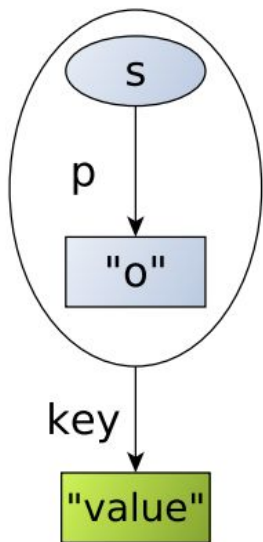


- Простая схема реификации, предложенная разработчиками СУБД Blazegraph
- Использует новый синтаксис RDF\* и SPARQL\*
- Транслируется в любую предыдущую схему
- Имеет формальную модель RDF\*-графа
- Новейший драфт спецификации от 13.02.2021

```
<<dbr:Moscow dbo:population 1174700>> dbo:year 1902 .  
<<dbr:Moscow dbo:population 12615882>> dbo:year 2019 .
```

```
SELECT ?population ?year WHERE {  
  << dbr:Moscow dbo:population ?population >> dbo:year ?year .  
}
```

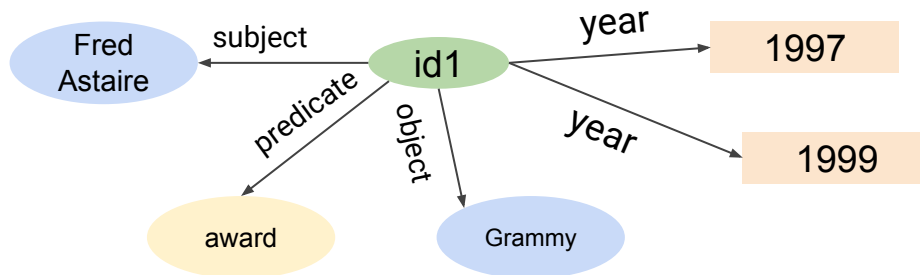
# RDF\* / SPARQL\*



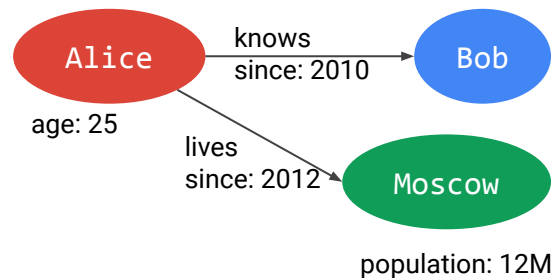
- + Получил распространение в разных СУБД (Stardog, GraphDB, Jena)
- + Концептуально связывает модели RDF и LPG
- + Существуют инструменты преобразования RDF  $\leftrightarrow$  RDF\*

- Реификация одного и того же триплета несколькими свойствами транслируется в один `rdf:Statement`
- Триплет в угловых скобках не существует как ребро между узлами

```
<<dbr:Fred_Astaire dbo:award dbr:Grammy>> dbo:year 1997 .  
<<dbr:Fred_Astaire dbo:award dbr:Grammy>> dbo:year 1999 .
```



# RDF\* / SPARQL\*



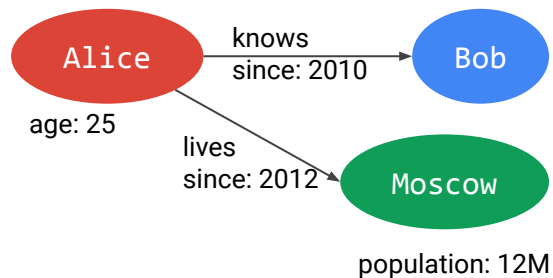
<< Alice knows Bob >> since 2010 .

<< Alice lives Moscow >> since 2012 .

Alice age 25 .

Moscow population 12M .

# RDF\* / SPARQL\*



```
<< Alice knows Bob >> since 2010 .  
<< Alice lives Moscow >> since 2012 .
```

```
Alice age 25 .  
Moscow population 12M .
```

С какого года Алиса знает Боба?

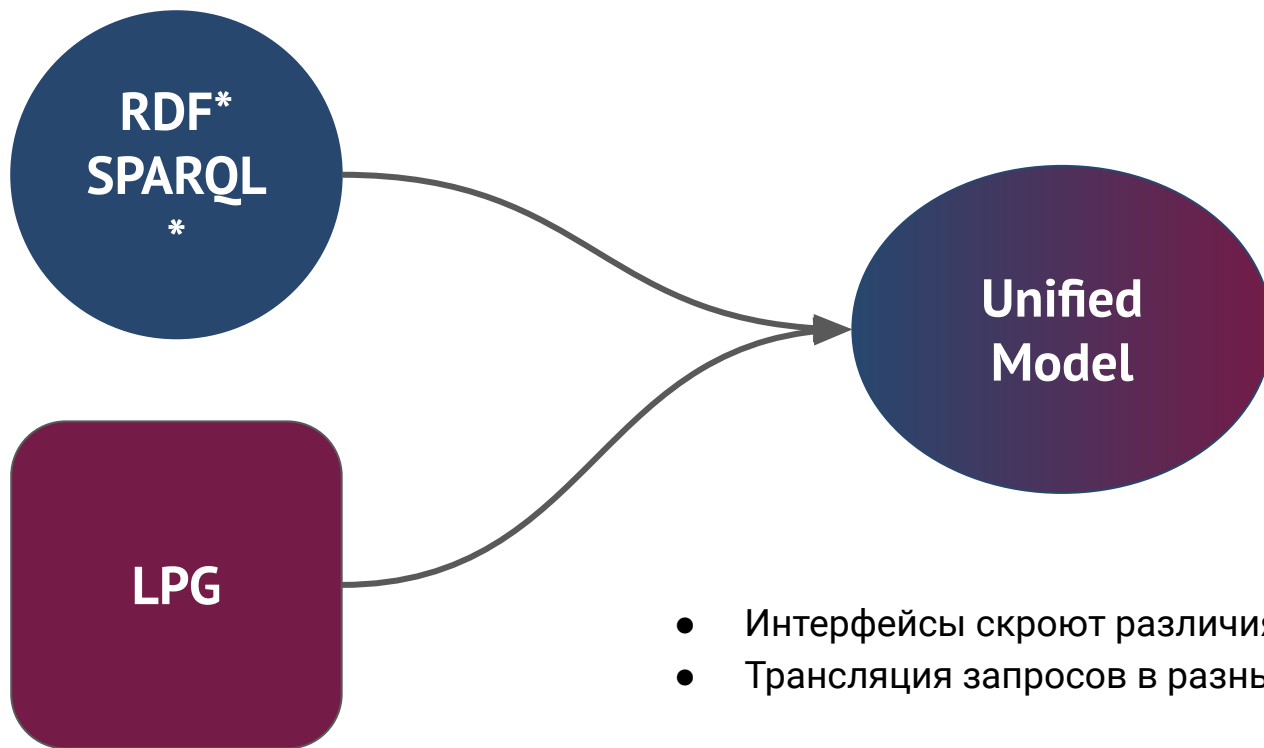
```
SELECT ?date WHERE {  
  << Alice knows Bob >> since ?date . }
```

Сколько человек живет в городе, в котором живет Алиса с 2012 года?

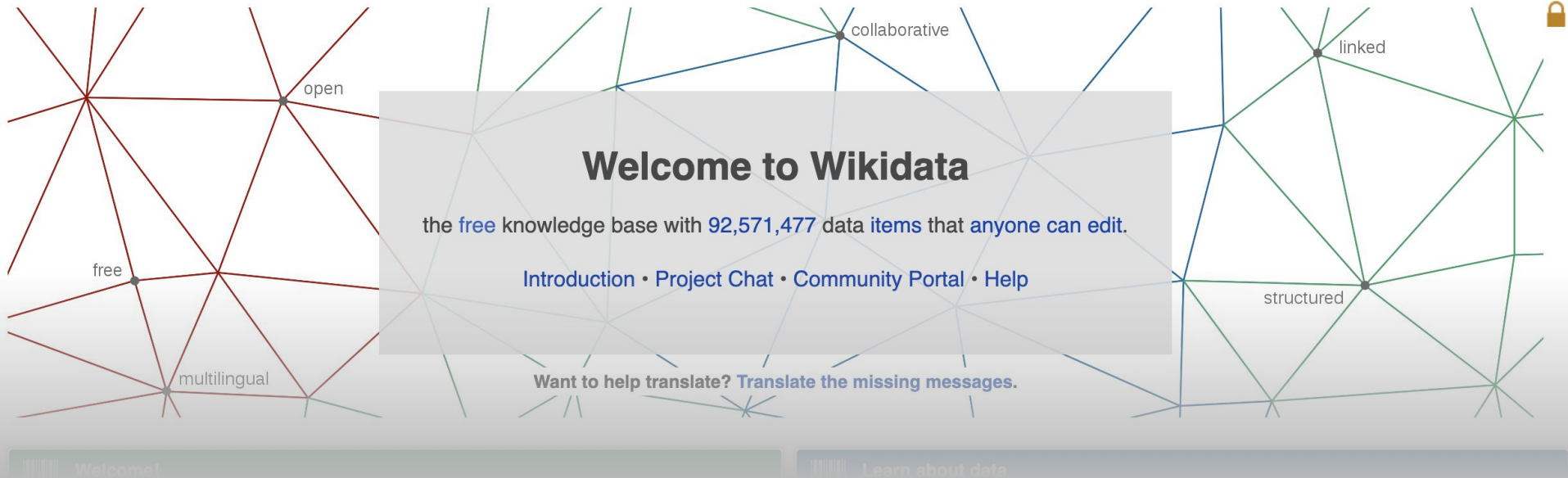
```
SELECT ?population WHERE {  
  << Alice lives ?city >> since 2012 .  
  ?city population ?population . }
```



# RDF\* / SPARQL\* + LPG Convergence

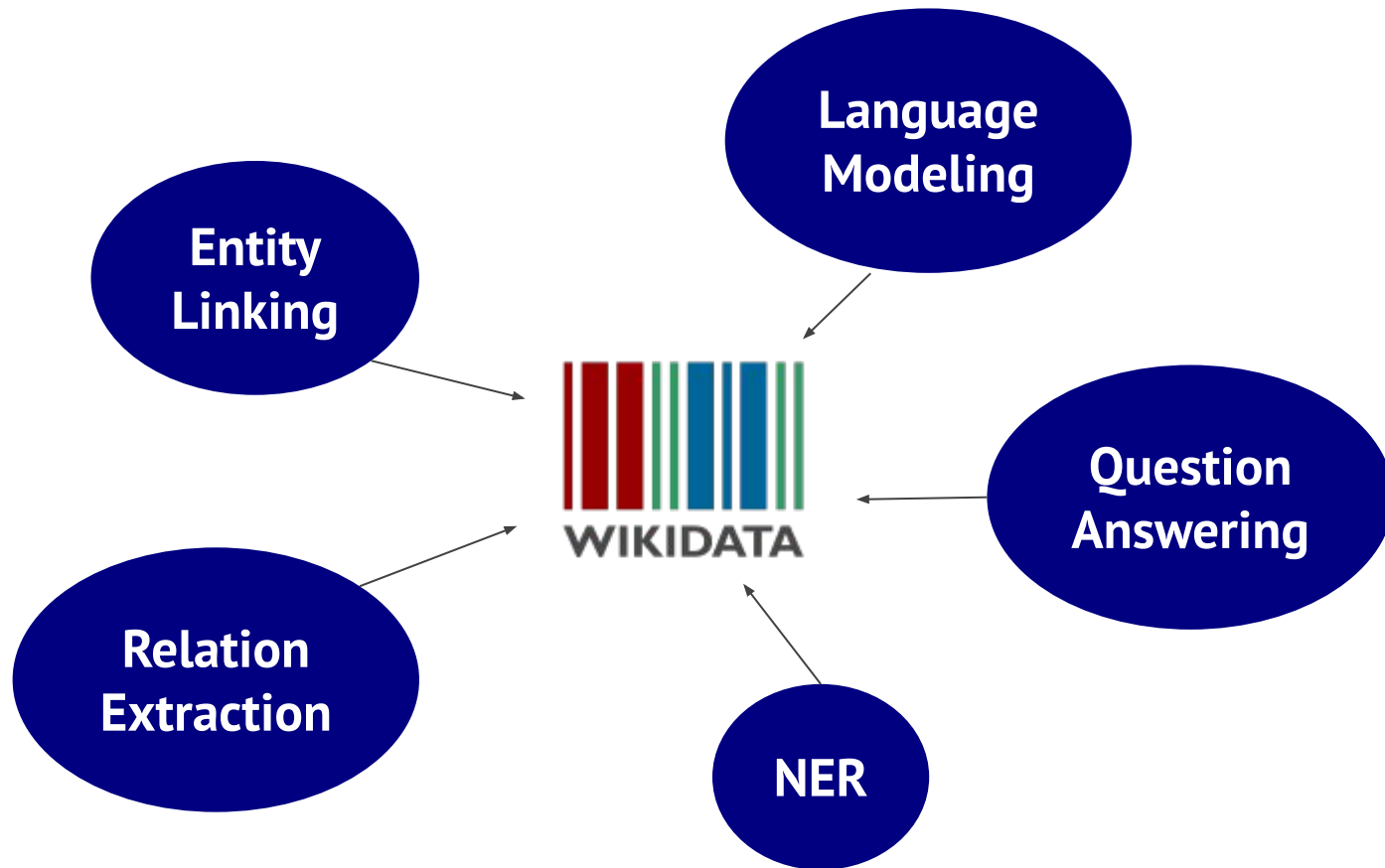


- Интерфейсы скроют различия в реализации
- Трансляция запросов в разные языки



- Один из самых больших графов общего назначения в открытом доступе
  - Около 100 миллионов сущностей
  - Около 12 миллиардов фактов (ребер)
- В Wikidata загружают данные как люди, так и компании
- Все больше научных публикаций используют Wikidata

# Wikidata в центре ML и NLP исследований

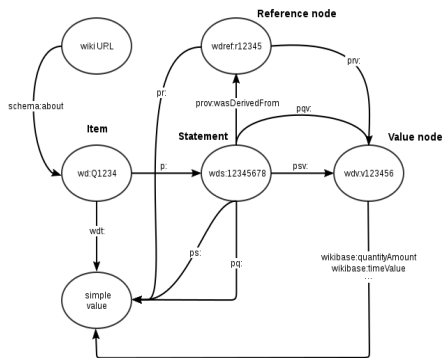


# Модель данных Wikidata



Открытый, поддерживаемый сообществом граф знаний, из которого Wikipedia получает данные для инфобоксов

- Модель изначально основана не на RDF, но может быть экспортирована в RDF
- Использует собственную модель данных и реификации Wikidata Statement Model (WSM)
  - Утверждения (claims) могут иметь квалификаторы (qualifiers)
  - Квалификаторы упрощенно имеют вид ключ-значение
  - Квалификаторы задают контекст правдивости утверждения
  - Утверждения могут иметь явный источник происхождения значения (references)
- Есть SPARQL-endpoint



# Wikidata Entities

Сущности делятся на 4 вида:

Item (Q)

- Предмет (Item), префикс Q:
  - Q649 (Moscow), Q656 (Saint Petersburg)

Property (P)

- Предикат (Property), префикс P:
  - P1082 (population)

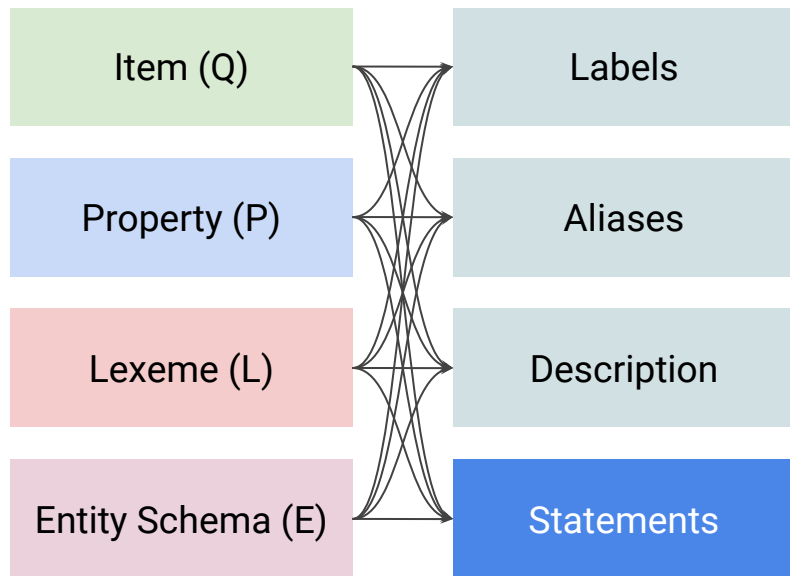
Lexeme (L)

- Лексема (Lexeme), префикс L:
  - L10 - to describe

Entity Schema (E)

- Классовые формы (Entity Schema), префикс E:
  - E11 - film festival schema

# Wikidata Entities

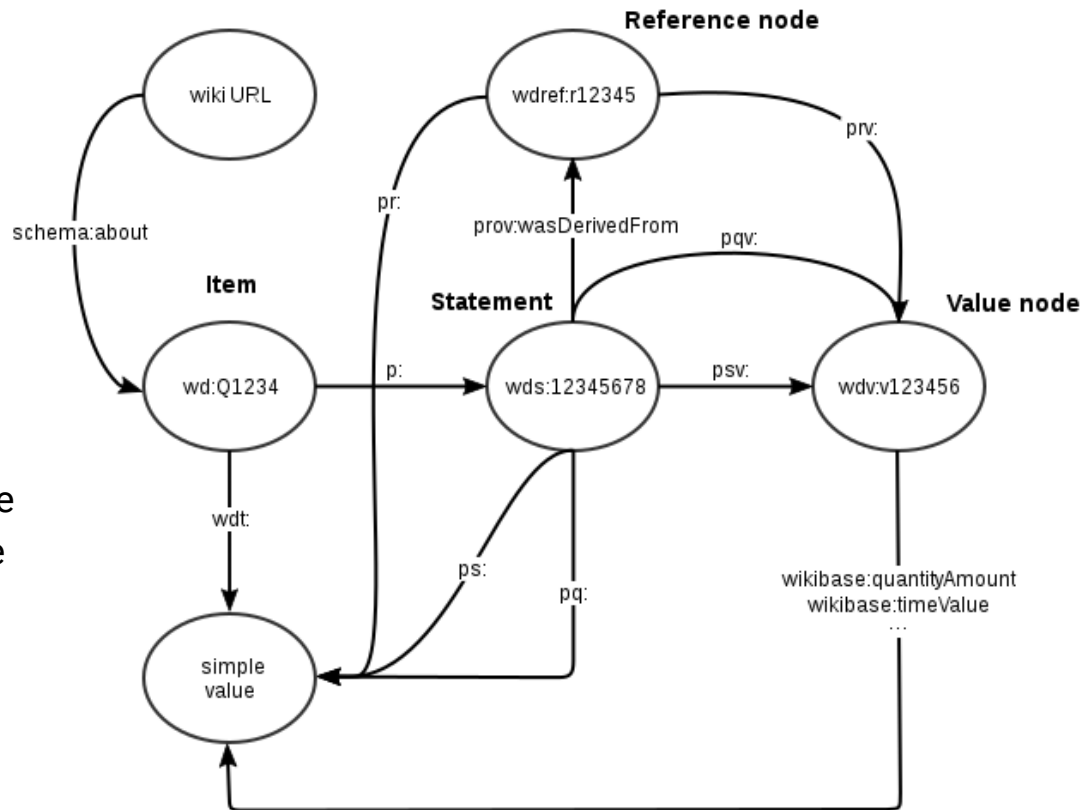


Каждая сущность может иметь

- Каноническое название (label) на разных языках (литералы)
- Синонимы (aliases) на разных языках (литералы)
- Текстовое описание (description) на разных языках (литералы)
- Высказывания (statements) (объекты)

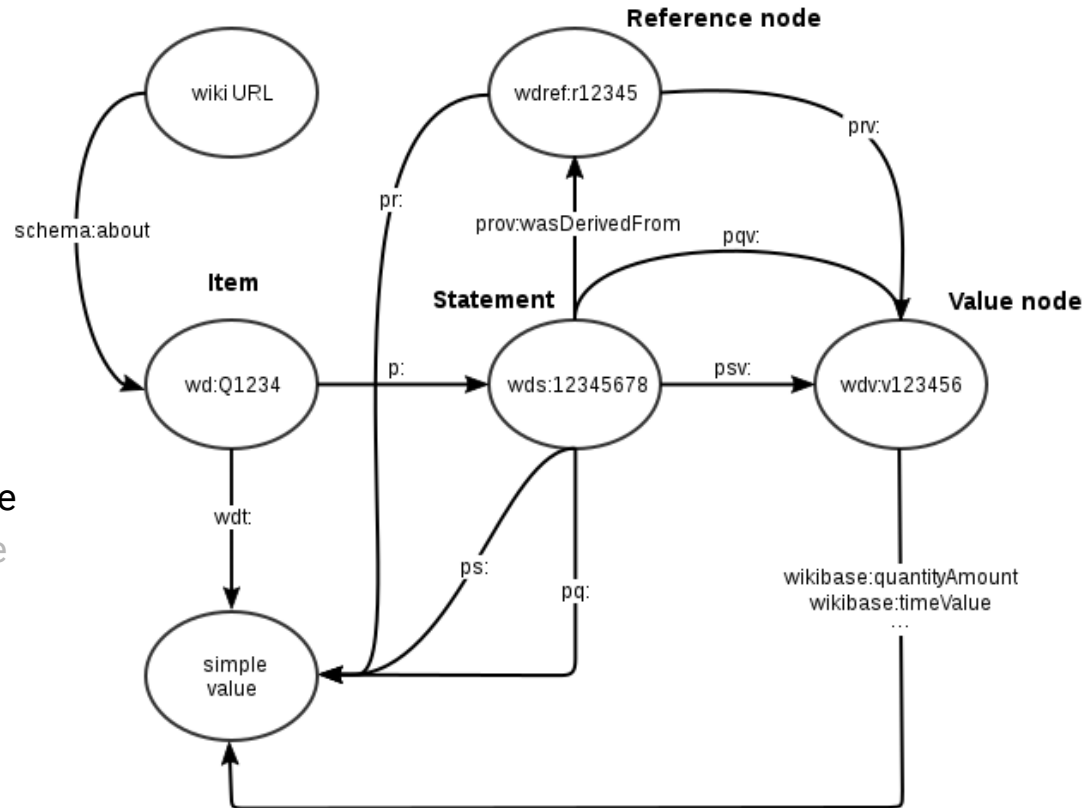
# Wikidata Statement Model

- **wd:** - основной префикс
  - **wdt:** - item -> truthy value
  - **wds:** - statement
  - **wdv:** - value node
  - **wdref:** - reference node
- 
- **p:** - predicate -> statement
  - **pq:** - statement -> qualifier
  - **ps:** - statement -> simple value
  - **psv:** - statement -> value node
  - **pqv:** - qualifier -> value node
  - **prov:** - statement -> reference
  - **prv:** - reference -> value node
  - **pr:** - reference -> simple value



# Wikidata Statement Model - Important Prefixes

- **wd:** - основной префикс
  - **wdt:** - item -> truthy value
  - **wds:** - statement
  - **wdv:** - value node
  - **wdref:** - reference node
- 
- **p:** - predicate -> statement
  - **pq:** - statement -> qualifier
  - **ps:** - statement -> simple value
  - **psv:** - statement -> value node
  - **pqv:** - qualifier -> value node
  - **prov:** - statement -> reference
  - **prv:** - reference -> value node
  - **pr:** - reference -> simple value





# Wikidata Statement Model

label

Moscow

(Q649)

item identifier

description

capital city and the largest city of Russia; separate federal subject of Russia

aliases

Moskva | Москва | Moscow, Russia | Moskva Federal City, Russia | Moscow, USSR | Moskva, Russia | City of Moscow | Moscow, Russian Federation | Moscow, Soviet Union | Moscow, Russian SFSR

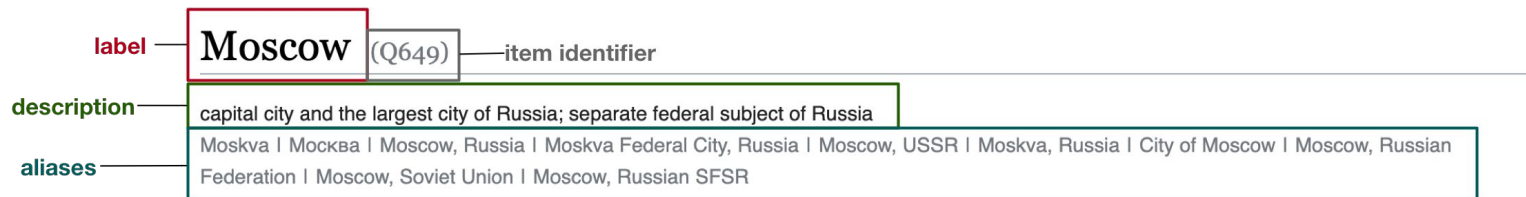
▼ In more languages

Configure

Language	Label	Description	Also known as
English	Moscow	capital city and the largest city of Russia; separate federal subject of Russia	Moskva Москва Moscow, Russia Moskva Federal City, Russia Moscow, USSR Moskva, Russia City of Moscow Moscow, Russian Federation Moscow, Soviet Union Moscow, Russian SFSR
language, label, description, aliases			
Russian	Москва	столица и крупнейший город России; город федерального значения; административный центр Московской области (не входит в её состав)	Первопрестольная Порт пяти морей Москва (город) Москва, Россия Москва (Россия) Москва Златоглавая Третий Рим
German	Moskau	Hauptstadt von Russland	
French	Moscou	capitale de la Russie	

All entered languages

# Wikidata Statement Model

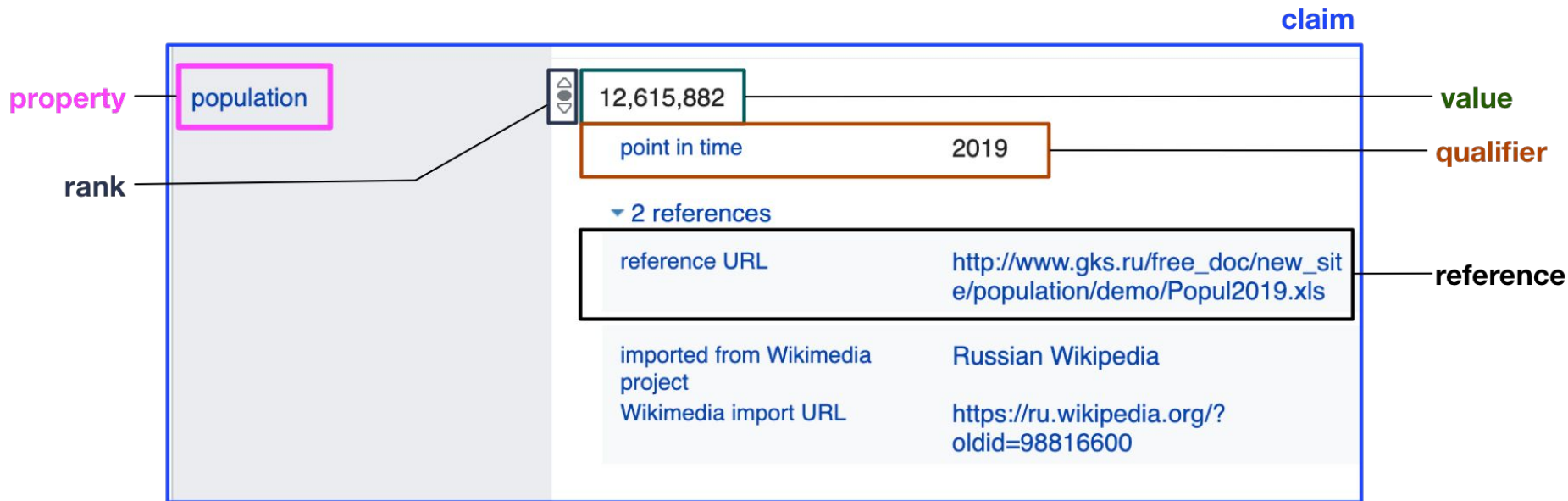


`wd:Q649`     **`rdfs:label`** “Moscow”@en ;

**`schema:description`** “capital city and the largest city of  
Russia ...”@en ;

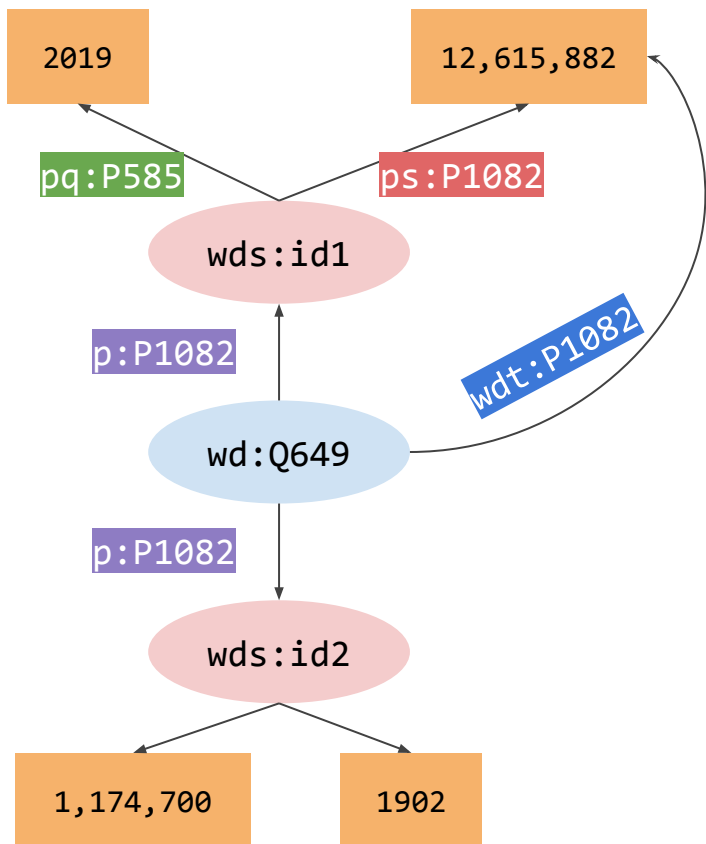
**`skos:altLabel`** “Moskva | Москва | Moscow, Russia | ...”

# Wikidata Statement Model



- Ранг (rank) - текущий статус утверждения: предпочитаемое, обычное, устаревшее
- Истинное высказывание: префикс wdt :, указывает на значение, квалификаторы опускаются

# Wikidata Statement Model



# содержит утверждение о населении Москвы за 2019 год

wd:Q649 `p:P1082` wds:id1 .

wds:id1 `ps:P1082` 12,615,882 ;

`pq:P585` 2019 .

# содержит утверждение о населении Москвы за 1902 год

wd:Q649 `p:P1082` wds:id2 .

wds:id2 `ps:P1082` 1,174,700 ;

`pq:P585` 1902 .

# truthy statement о текущем населении Москвы

wd:Q649 `wdt:P1082` 12,615,882 .

# Wikidata Data Model

- В Wikidata предметы (Items) могут быть и (под)классами, и экземплярами одновременно

IBM Watson (Q12253) - инстанс

- artificial intelligence (Q11660),
- supercomputer (Q121117),
- one-of-a-kind computer (Q28542014),

IBM Watson (Q12253) - подкласс

- computer (Q68)

## Watson (Q12253)








artificial intelligence computer system made by IBM

IBM Watson

 edit

[In more languages](#)

### Statements

instance of		artificial intelligence	 edit
		▾ 0 references	+ add reference
		supercomputer	 edit
		▸ 1 reference	
		one-of-a-kind computer	 edit
		▾ 0 references	+ add reference
	+ add value		
	subclass of		computer
		▾ 0 references	+ add reference
+ add value			

# Wikidata Data Model

rdf:type

wd:P31  
(instanceOf)

rdfs:sub  
ClassOf

wd:P279  
(subclass of)

owl:inve  
rseOf

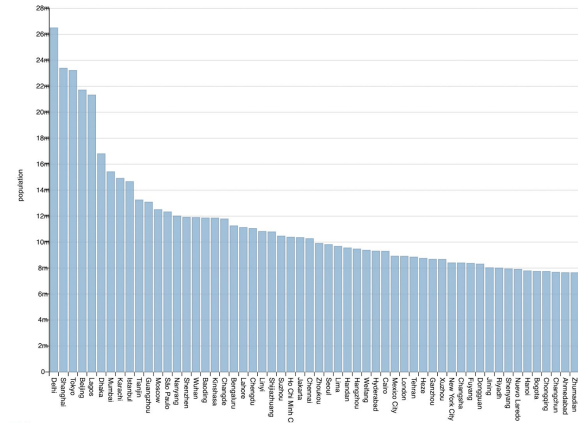
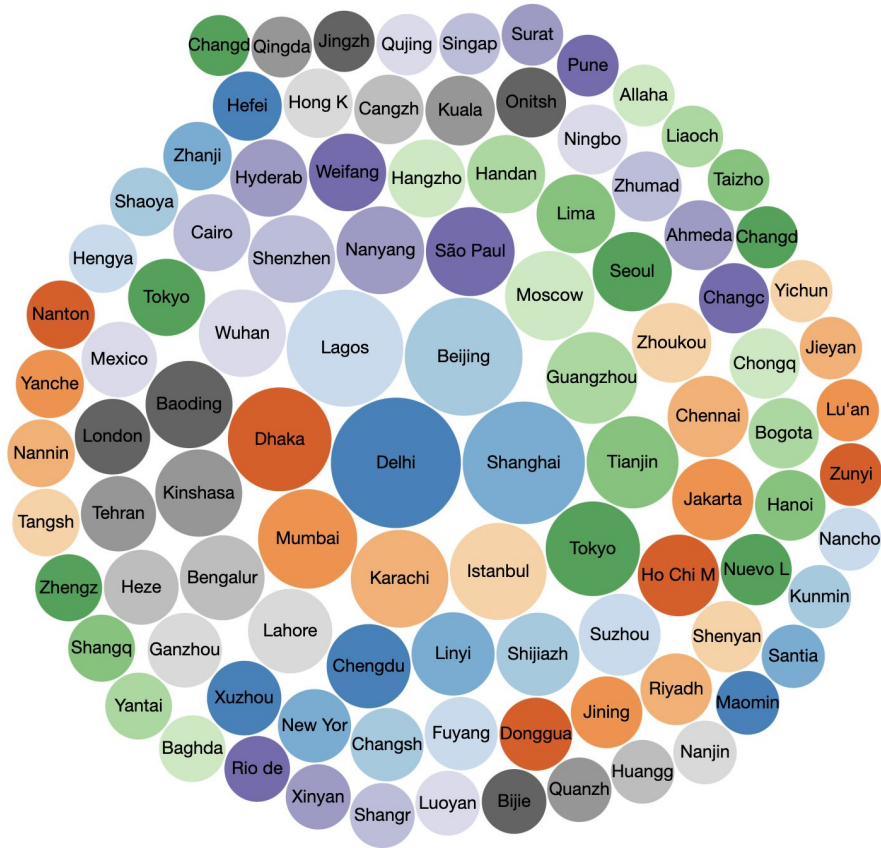
wd:P1696  
(inverse of)

- В Wikidata предметы (Items) могут быть и (под)классами, и экземплярами одновременно
- По этой причине не используются предикаты из RDF(S) и OWL (только rdfs:label)
- Вводятся собственные предикаты без строгой семантики
- Не существует онтологии Wikidata
  - Есть таксономии классов и предикатов



```
1 #Крупнейшие города мира
2 #defaultView:BubbleChart
3 SELECT DISTINCT ?cityLabel ?population ?gps
4 WHERE
5 {
6   ?city wdt:P31/wdt:P279* wd:Q515 .
7   ?city wdt:P1082 ?population .
8   ?city wdt:P625 ?gps .
9   SERVICE wikibase:label {
10     bd:serviceParam wikibase:language "en" .
11   }
12 }
13 ORDER BY DESC(?population) LIMIT 100
```

<https://query.wikidata.org/>





# Производительность СУБД при реификации

Table 2

**MRM Comparison (factorization, backward compatibility & usability):** The level of factorization of each MRM is measured for a set of 100 DBpedia entities and its revision metadata (95,864 meta facts, 950 on avg. per entity). Furthermore, the query complexity between different MRMs is compared. Note: The first number in the statements count column of rdr MRMs corresponds with the number of rdr (nested) statements in the database, whereas the second represents the number of triples, obtained by unnesting the rdr statements. The file format for rdr is `.ntx`, an extension of N-Triples.

	# statements	.nq file size (MB)	backward comp.	#{triple patterns)	#{overhead variables) & #{sparql elements)
raw data	8,444	1.16	/	/	/
cpprop	115,822	19.76	yes - w/ rdfs reasoning	5	3
naryrel	11,202,942	2161.53	no	3	2 & 1 BIND + 3 string-functions
naryrel (shared)	122,812	21.36	no	4	3 & 1 BIND + 3 string-functions
ngraphs	104,308	19.34	quads: no / triples: yes	2	1 & 1 GRAPH
rdr	11,126,845 / 22,169,250	2856.60 (.ntx)	quads: no / triples: yes	2	1 & 1 BIND
rdr (shared)	246,947 / 314,499	44.55 (.ntx)	quads: no / triples: yes	3	2 & 1 BIND
sgprop	11,202,942	2193.75	yes - w/ rdfs reasoning	3	1
sgprop (shared)	122,812	21.52	yes - w/ rdfs reasoning	4	2
stdreif	11,354,934	2188.17	yes - w/ custom reasoning	5	1
stdreif (shared)	141,316	24.63	yes - w/ custom reasoning	6	2

# Производительность СУБД при реификации

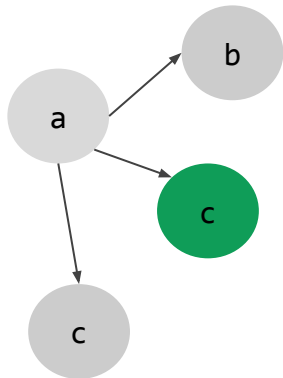
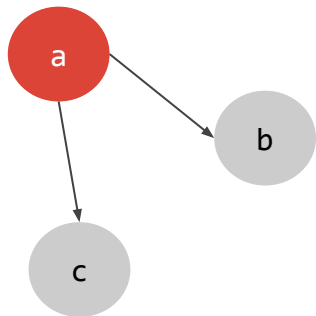
Table 3

**Wikidata experiment:** The number of statements for the Wikidata dataset, its respective loading times and the final database size for the different MRMs. All loading times are in hours, whereas database sizes are in GiB.

	<b>naryrel</b>	<b>ngraphs</b>	<b>sgprop</b>	<b>stdreif</b>	<b>rdr</b>
<b>#statements</b>	563,678,588	482,371,357	563,676,547	644,981,737	563,676,547
<b>loading time Virtuoso (hours)</b>	3.39	3.04	3.22	3.15	-
<b>db size Virtuoso (GiB)</b>	45.94	46.83	46.25	45.21	-
<b>loading time Blazegraph (hours)</b>	14.57	13.03	7.12	7.09	10.40
<b>db size Blazegraph (GiB)</b>	60.73	98.25	60.73	60.73	66.87
<b>loading time Stardog (hours)</b>	1.12	1.35	1.07	0.85	-
<b>db size Stardog (GiB)</b>	32.34	56.52	32.31	32.19	-

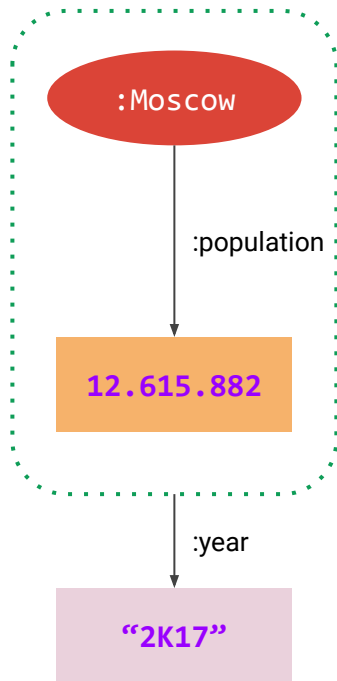
# Валидация графов знаний

# Валидация графов знаний



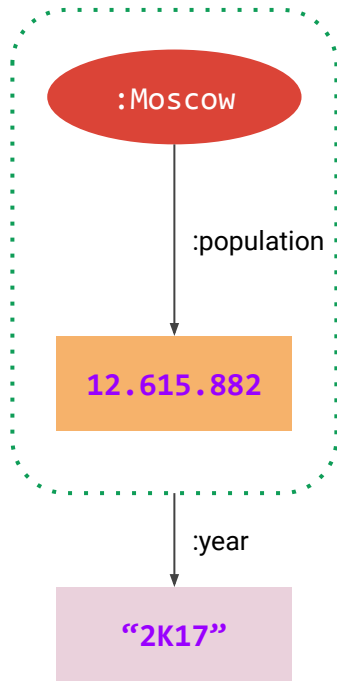
- В практических применениях нужно оценивать качество информационной системы
- Для графов знаний стандартные параметры еще (на 2021 г.) не определены
- Но у нас есть построенные онтологии!
  - Логически непротиворечивые (hopefully)

# Валидация графов знаний - Критерии



- Синтаксическая корректность
  - Оценивает физическое представление графов
  - При загрузке и индексировании проверки выполняют СУБД
  - Пример: целое число `12.615.882` не может содержать несколько точек
- Семантическая корректность
  - Оценивает логическую непротиворечивость
  - Рекомендации SHACL и ShEx
  - Пример: значение предиката `year` не может быть строкой
- Управление знаниями в целом (data governance)
  - Важно, когда граф разрабатывается и поддерживается коллективно
  - Игруют роль внутренние стандарты качества данных

# Валидация графов знаний - Graph Shape



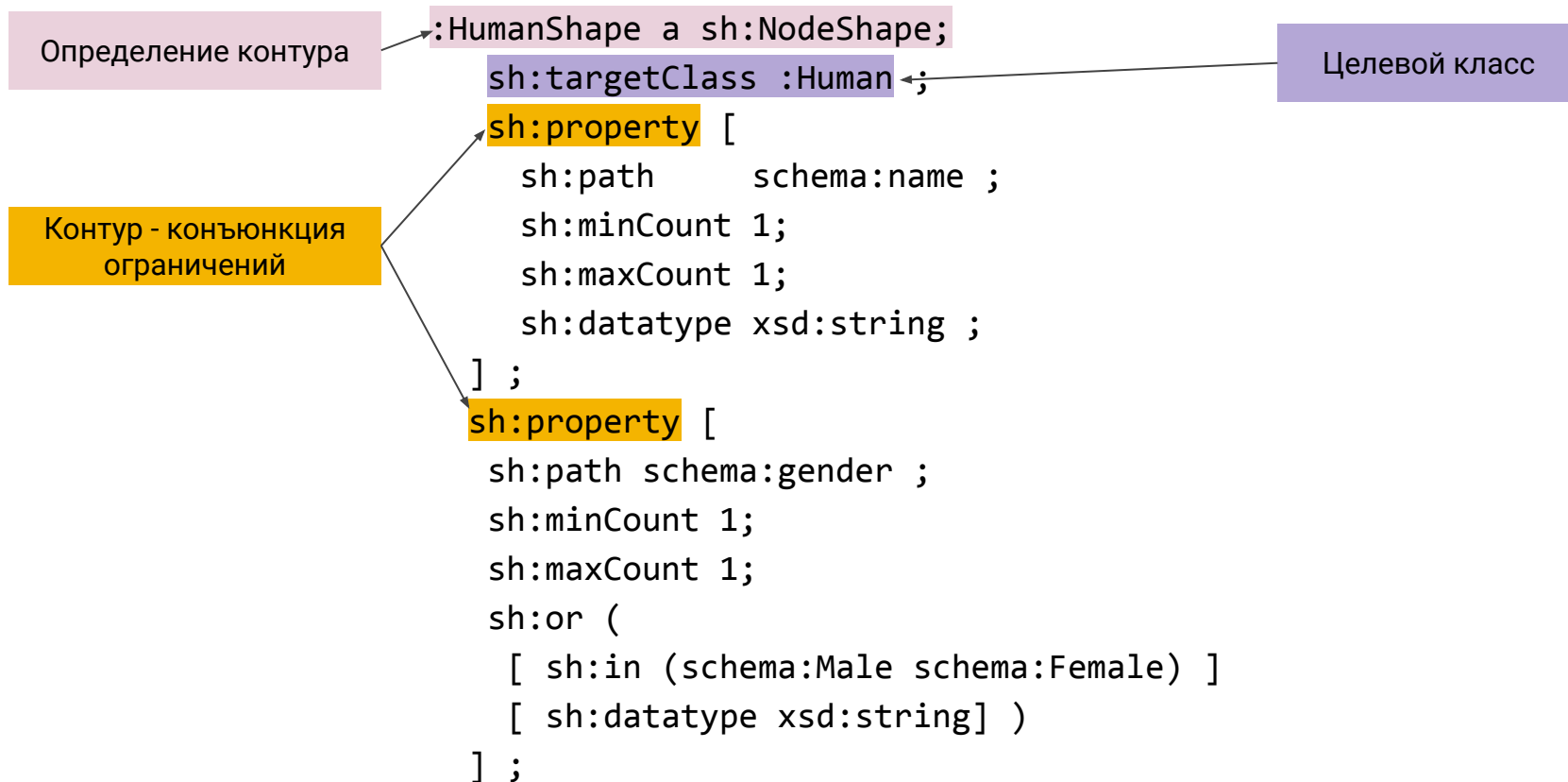
- Контур графа (graph shape) - набор ограничений на компоненты графа знаний:
  - На экземпляры заданного класса
  - На тип значения предиката
  - На разрешенное количество предикат на экземпляр
  - На сочетания предикатов
  - ... и более сложные

Shapes Constraint Language (SHACL) - язык описания контурных ограничений - официальная рекомендация W3C к валидации графов

SHACL

ShEX

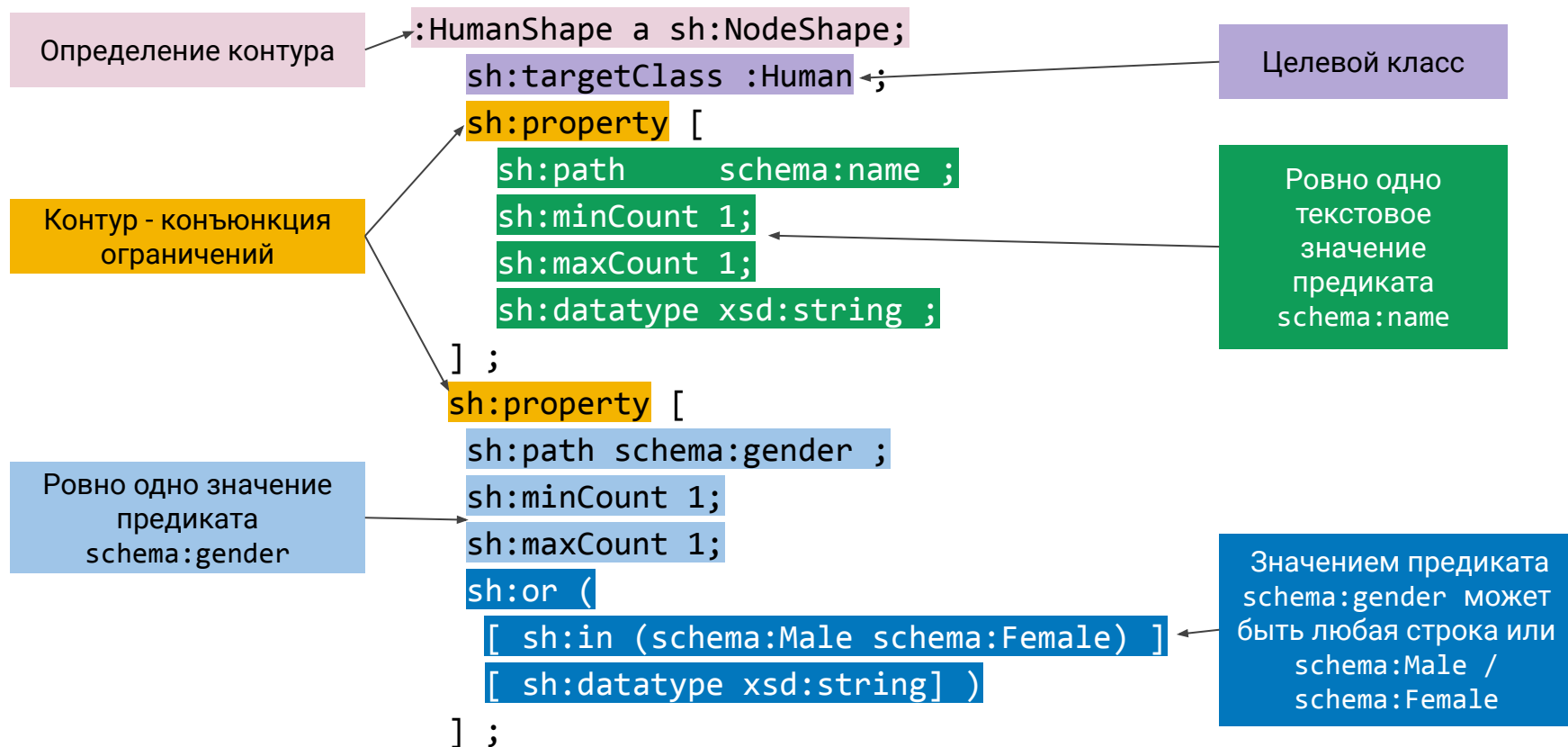
- Первый драфт: 2015 год, рекомендация: 2017 год
- Задумывался для объединения существующих подходов к валидации (ShEx и другие)
- SHACL Core - основная часть, определяет набор RDF-терминов для описания контуров и базовую семантику валидации
- SHACL-SPARQL - надмножество SHACL Core, использует семантику стандарта SPARQL 1.1, вводит более продвинутые механизмы
- Поддерживается TopQuadrant, реализации на Java, JS, Python





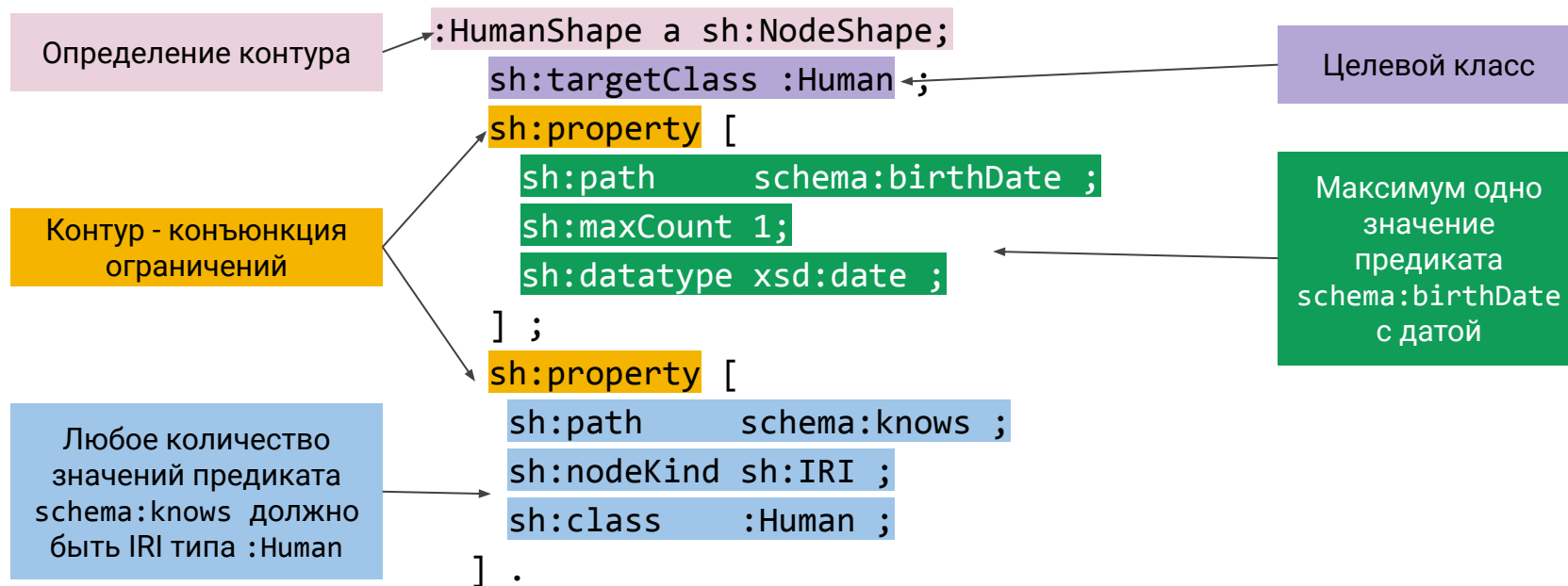
# SHACL Example

<https://www.w3.org/TR/shacl/>



# SHACL Example

<https://www.w3.org/TR/shacl/>



```
:alice a :Human;  
  schema:name      "Alice" ;  
  schema:gender    schema:Female ;  
  schema:knows     :bob .
```

SHACL  
Validator

```
[ a sh:ValidationReport ;  
  sh:conforms true  
]
```

Shapes graph

```
:emily a :Human;  
  schema:name      "Emily",  
                  "Emilee" ;  
  schema:gender    schema:Female .
```

SHACL  
Validator

Shapes graph

```
[ :report a sh:ValidationReport ;  
  sh:conforms false ;  
  sh:result  
    [ a      sh:ValidationResult ;  
      sh:resultSeverity    sh:Violation ;  
      sh:sourceConstraintComponent  
sh:MaxCountConstraintComponent ;  
      sh:sourceShape ... ;  
      sh:focusNode       :emily ;  
      sh:resultPath      schema:name ;  
      sh:resultMessage  
        "More than 1 values" ] .  
]
```

```
:HumanShape a sh:NodeShape;  
  sh:targetClass :Human
```

```
sh:property [  
  sh:path      schema:name      ]
```

```
sh:property [  
  sh:path      schema:gender    ]
```

```
sh:property [  
  sh:path      schema:birthDate ]
```

```
sh:property [  
  sh:path      schema:knows     ]
```

- Node Shape
  - Применяется к целевому узлу
  - :HumanShape
- Property Shape
  - Применяется к одному предикату или их комбинации
  - Комбинация предикатов - property paths в SPARQL 1.1
  - Выразительный механизм ограничений

Контур содержит SPARQL запрос

Запрос как значение предиката

`$this` указывает на текущий проверяемый инстанс

`FILTER` объявляет ограничения на значения

```
:UserShape a sh:NodeShape;
  sh:targetClass :Human ;
  sh:sparql [
    a sh:SPARQLConstraint ;
    sh:message "schema:name must equal schema:givenName+schema:familyName";
    sh:prefixes [
      sh:declare [ sh:prefix    "schema" ;
                   sh:namespace "http://schema.org/"^^xsd:anyURI ; ]] ;
    sh:select
      """SELECT $this (schema:name AS ?path) (?name as ?value)
      WHERE {
        $this schema:name      ?name .
        $this schema:givenName ?givenName .
        $this schema:familyName ?familyName .
        FILTER (!isLiteral(?value) ||
                !isLiteral(?givenName) ||
                !isLiteral(?familyName) ||
                concat(str(?givenName), ' ', str(?familyName))!=?name
                )}""" ;
  ] .
```

Shapes Expressions (ShEx) - язык описания структуры RDF графов

SHACL

- Разработан в 2013 году, актуальная версия ShEx 2.1 - 2018 год
- ShEx был отправлен в W3C, но рекомендацией стал SHACL
- ShEx - больше грамматика для создания RDF графов, но также позволяет объявлять контуры графа и производить валидацию

ShEX

- ShExC - компактный синтаксис
- ShExJ - синтаксис на основе JSON-LD
- ShExR - RDF-граф на основе JSON-LD документа
- Реализации на множестве платформ, в тч Python, Java, JS, Ruby

# ShEx Example

<https://www.w3.org/TR/shacl/>

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

Определение контура

```
:Human {
  schema:name          xsd:string ;
  schema:birthDate     xsd:date? ;
  schema:gender
    [ schema:Male schema:Female ] OR xsd:string ;
  schema:knows          IRI @:Human*
}
```

Максимум одно  
значение  
предиката  
schema:birthDate  
с датой

Ровно одно  
текстовое  
значение  
предиката  
schema:name

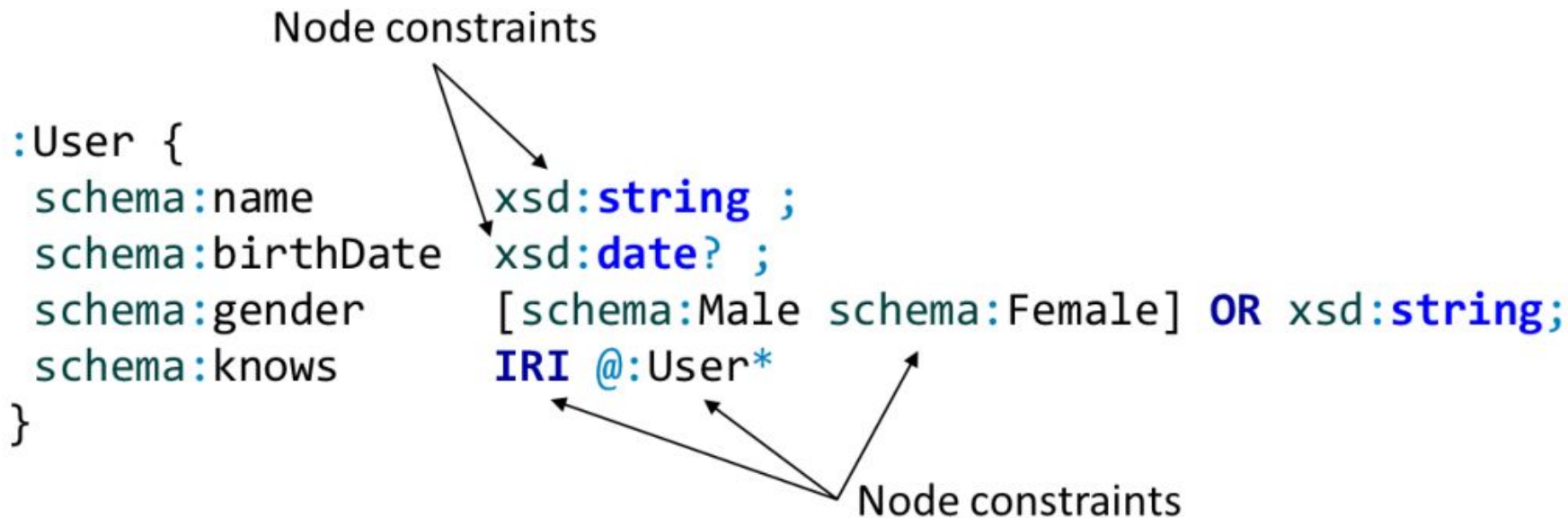
Любое количество  
значений предиката  
schema:knows должно  
быть IRI типа :Human

Значением предиката  
schema:gender может  
быть любая строка или  
schema:Male /  
schema:Female



# ShEx Node Constraints

<https://www.w3.org/TR/shacl/>



# ShEx Node Constraints Facets

<https://www.w3.org/TR/shacl/>

Фасет и аргумент	Корректные значения	Некорректные значения
MinInclusive 1	"1"^^xsd:decimal, 1, 2, 98, 99, 100	"1"^^xsd:string, -1, 0
MinExclusive 1	2, 98, 99, 100	-1, 0, 1
MaxInclusive 99	1, 2, 98, 99	100
MaxExclusive 99	1, 2, 98	99, 100
TotalDigits 3	"1"^^xsd:integer, 9, 999, 0999, 9.99, 99.9, 0.1020	"1"^^xsd:string, 1000, 01000, 1.1020, .1021, 0.1021
FractionDigits 3	"1"^^xsd:decimal, 0.1, 0.1020, 1.1020	"1"^^xsd:integer, 0.1021, 0.10212
Length 3	"123"^^xsd:string, "123"^^xsd:integer, "abc"	"12"^^xsd:string, "12"^^xsd:integer, "ab", "abcd"
MinLength 3	"abc", "abcd"	"", "ab"
MaxLength 3	"", "ab", "abc"	"abcd", "abcde"
/^ab+/ Regex pattern	"ab", "abb", "abbcd"	"", "a", "acd", "cab" , "AB", "ABB", "ABBCD"
/^ab+/i	"ab", "abb", "abbcd"	"", "a", "acd"

# ShEx Triple Constraints

<https://www.w3.org/TR/shacl/>

```
:User {
```

```
  schema:name      xsd:string ;
```

```
  schema:birthDate xsd:date ? ;
```

```
  schema:gender    [schema:Male schema:Female] OR xsd:string;
```

```
  schema:knows     IRI @:User *
```

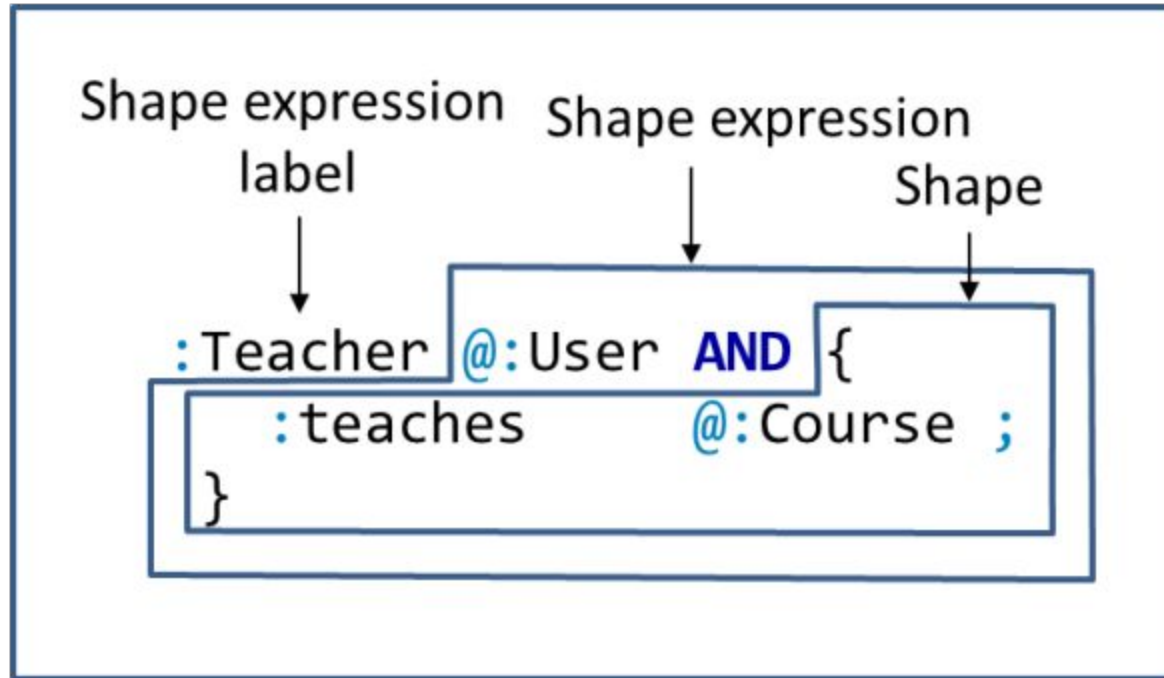
```
}
```

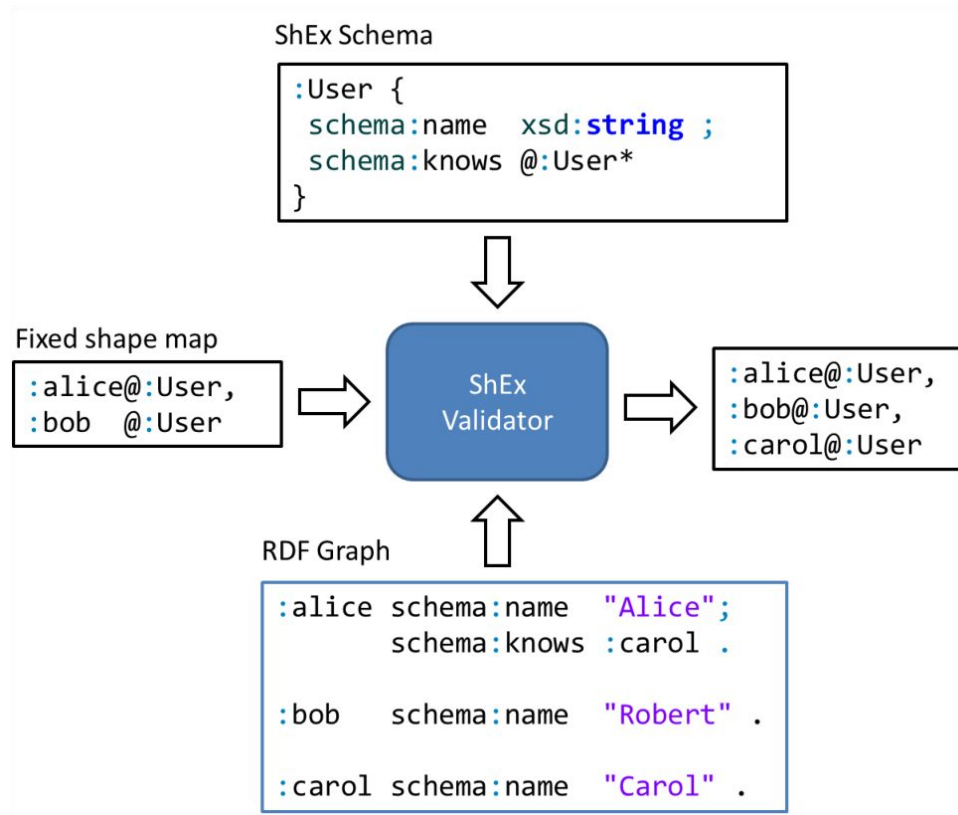
Triple  
constraints

- |   |  |
|---|--|
| <code>schema:name xsd:string</code>       | ● по умолчанию: ровно 1 раз              |
| <code>schema:name xsd:string *</code>     | ● * - ноль раз или больше                |
| <code>schema:name xsd:string ?</code>     | ● ? - ноль или один раз                  |
| <code>schema:name xsd:string +</code>     | ● + - один и более раз                   |
| <code>schema:name xsd:string {3,4}</code> | ● {i, j} - не менее i, но не более j раз |
| <code>schema:name xsd:string {3}</code>   | ● {m} - ровно m раз                      |
| <code>schema:name xsd:string {3,}</code>  | ● {m, } - m и более раз                  |

# ShEx Complex Shapes

<https://www.w3.org/TR/shacl/>





# ShEx 2.0 Shape Maps

<https://www.w3.org/TR/shacl/>

## Fixed shape map

```
:alice@:User,  
:bob @:User
```

## Query shape map

```
{FOCUS schema:worksFor _ }@:User,  
{FOCUS rdf:type schema:Person}@:User,  
{_ schema:worksFor FOCUS }@:Company
```

- Fixed Shape Map
  - Содержит селекторы узлов графа для валидации
- Query Shape Map
  - Содержит набор triple patterns, которые могут содержать константы или переменные для поиска пути в графе
- Result Shape Map
  - Содержит результаты работы валидатора

node	shape	result	reason
<node1>	<Shape1>	pass	
<node2>	<Shape2>	fail	"msg"

# ShEx & Wikidata

[https://www.wikidata.org/wiki/Wikidata:WikiProject\\_ShEx](https://www.wikidata.org/wiki/Wikidata:WikiProject_ShEx)

human <sup>(E10)</sup>

language code	label	description	aliases	edit
en	human	simple schema for humans	person   human being	<a href="#">✎ edit</a>
ca	humà	schema per a éssers humans	persona   ésser humà	<a href="#">✎ edit</a>
cs	osoba	jednoduché schéma pro člověka	člověk   osoba	<a href="#">✎ edit</a>
da	menneke		person	<a href="#">✎ edit</a>
de	Mensch	einfaches Schema für Objekte über Menschen	Person	<a href="#">✎ edit</a>
el	άνθρωπος			<a href="#">✎ edit</a>
eo	homo	simpla skemo por homoj	persono	<a href="#">✎ edit</a>
es	ser humano	esquema simple para una persona	persona	<a href="#">✎ edit</a>
et	inimene	lihtne skeem inimese jaoks		<a href="#">✎ edit</a>
fi	ihminen	yksinkertainen skeema kohteelle ihminen	henkilö	<a href="#">✎ edit</a>
fr	humain	schéma simple pour un être humain	personne	<a href="#">✎ edit</a>
fy	minske		persoan	<a href="#">✎ edit</a>
gl	ser humano	esquema simple para definir unha persoa	persoa	<a href="#">✎ edit</a>
hu	ember		személy	<a href="#">✎ edit</a>
it	umano	schema per descrivere un essere umano	persona   individuo   essere umano	<a href="#">✎ edit</a>
ja	ヒト	ヒト記述用のスキーマ	人間	<a href="#">✎ edit</a>
ko	사람	사람을 위한 간단한 스키마	인간   인물	<a href="#">✎ edit</a>
lv	cilvēks		persona	<a href="#">✎ edit</a>
nl	mens	simpel schema voor mensen	persoon	<a href="#">✎ edit</a>
pt	humano	esquema simples para humanos		<a href="#">✎ edit</a>
pt-br	humano	esquema para descrever seres humanos		<a href="#">✎ edit</a>
ru	человек	простая схема для людей	личность   существование	<a href="#">✎ edit</a>
sk	osoba	schéma pre dátové položky ľudí	človek	<a href="#">✎ edit</a>
sq	njeri	skema e thjeshtë për njerëzit	personi   qenie njerëzore	<a href="#">✎ edit</a>
sr	osoba	prosta shema za osobe	човек	<a href="#">✎ edit</a>
sv	människa	ett enkelt schema för människor		<a href="#">✎ edit</a>
tr	insan	insanlar için basit şema	kişi	<a href="#">✎ edit</a>

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
```

```
start = {human}
```

```
<human> EXTRA wdt:P31 (
  wdt:P101 [wd:Q5];
  wdt:P118 . * ; # image (portrait)
  wdt:P21 [wd:Q6581097 wd:Q6581072]? ; # gender
  wdt:P19 . ? ; # place of birth
  wdt:P20 . ? ; # place of death
  wdt:P569 . ? ; # date of birth
  wdt:P570 . ? ; # date of death
  wdt:P735 . * ; # gives name
  wdt:P734 . * ; # family name
  wdt:P104 . * ; # occupation
  wdt:P1559 . ? ; # name in native language
  wdt:P27 {<country> * ; # country of citizenship
  wdt:P22 {<human> * ; # father
  wdt:P25 {<human> * ; # mother
  wdt:P337? {<human> * ; # sibling
  wdt:P26 {<human> * ; # spouse
  wdt:P40 {<human> * ; # children
  wdt:P103 {<human> * ; # relatives
  wdt:P103 {<language> * ; # native language
  wdt:P1412 {<language> * ; # languages spoken, written or signed
  wdt:P4886 {<language> * ; # writing language
  rdfs:label rdf:langString;
}

<country> EXTRA wdt:P31 (
  wdt:P31 [wd:Q4256 wd:Q3024240 wd:Q3624078] * ;
)
```

- ShEx официально поддерживается в Wikidata с мая 2019 года
- Четвертый глобальный тип (префикс E - EntitySchema)
- Инструменты валидации сущностей Wikidata
- Можно создавать контуры вручную или автоматически через SPARQL-запросы



start = @<human>

```
<human> EXTRA wdt:P31 {  
  wdt:P31 [wd:Q5];  
  wdt:P18 . * ;                # image (portrait)  
  wdt:P21 [wd:Q6581097 wd:Q6581072]?; # gender  
  wdt:P19 . ?;                  # place of birth  
  wdt:P20 . ?;                  # place of death  
  wdt:P569 . ? ;                # date of birth  
  wdt:P570 . ? ;                # date of death  
  wdt:P735 . * ;                # given name  
  wdt:P734 . * ;                # family name  
  wdt:P106 . * ;                # occupation  
  wdt:P1559 . ? ;              #name in native language  
  wdt:P27 @<country> *;         # country of citizenship  
  wdt:P22 @<human> *;           # father  
  wdt:P25 @<human> *;           # mother  
  wdt:P3373 @<human> *;         # sibling  
  wdt:P26 @<human> *;           # spouse  
  wdt:P40 @<human> *;           # children  
  wdt:P1038 @<human> *;         # relatives  
  wdt:P103 @<language> *;       # native language  
  wdt:P1412 @<language> *;      # languages spoken, written or signed  
  wdt:P6886 @<language> *;      # writing language  
  rdfs:label rdf:langString+;  
}
```

# SHACL vs ShEx

	SHACL	ShEx
Model	Constraints over RDF graphs	Grammar-based
Validating RDF graphs	Yes	Yes
Reporting	Yes	Yes
Syntax	RDF with its vocabulary	Compact, JSON-LD, RDF
Property Paths	Yes	No
Default cardinality	{0, *}	{1, 1}
Recursion & Cycles	No	Yes

# В следующей серии

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