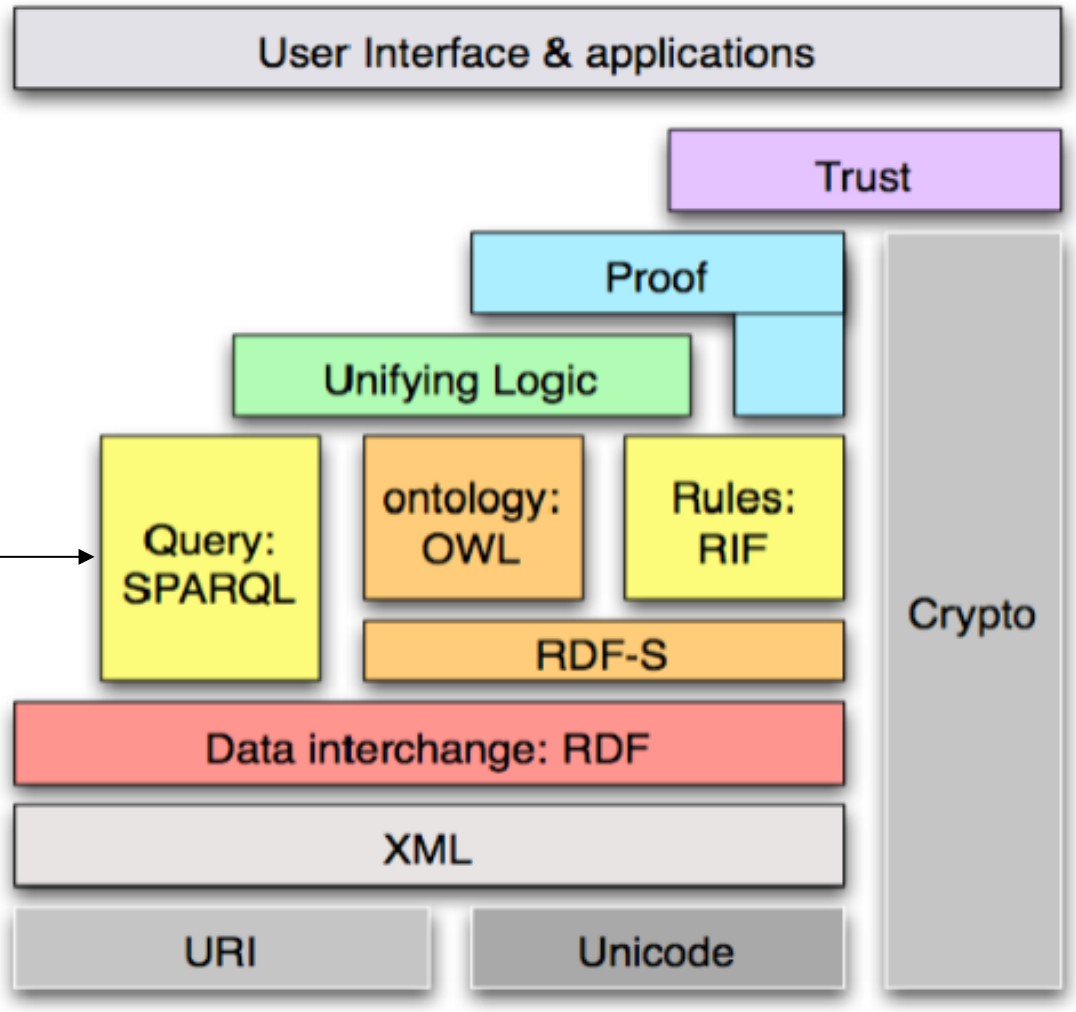
Język zapytań SPARQL

Agnieszka Ławrynowicz

Instytut Informatyki
Politechniki Poznańskiej
Poznań, 2018

Stos języków Sieci Semantycznej

Język zapytań SPARQL



Turtle

Turtle (Terse RDF Triple Language):

- **serializacja RDF**
- **reprezentacja „trójkowa” „trójek” <Subject, Predicate, Object>**
- **przyjazna dla człowieka alternatywa dla składni RDF/XML**
  - często wykorzystywana w przykładach, tutorialach, literaturze

Turtle

Turtle (Terse RDF Triple Language):

- **serializacja RDF**
- **reprezentacja „trójkowa” „trójek” <Subject, Predicate, Object>**
- **przyjazna dla człowieka alternatywa dla składni RDF/XML**
  - często wykorzystywana w przykładach, tutorialach, literaturze

```
@prefix osoba: <http://przyklad/osoba/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
osoba:a foaf:name "Alicja" .
osoba:a foaf:mbox <mailto:alicja@example.net> .
osoba:b foaf:name "Jan" .
```

SPARQL to…

SPARQL Protocol And Rdf Query Language - język zapytań Sieci Semantycznej

- Wymowa: jak „sparkle”
- Język zapytań dla danych w formacie RDF
- Protokół (HTTP, SOAP)

SPARQL pozwala na…

- Zapytania do źródeł danych strukturalnych i semistrukturalnych
- Eksplorowanie danych przez zadawanie zapytań dotyczących nieznanych związków
- Wykonywanie złożonych złączeń z różnorodnych baz danych w pojedynczym zapytaniu
- Transformację danych RDF z jednego słownika do innego

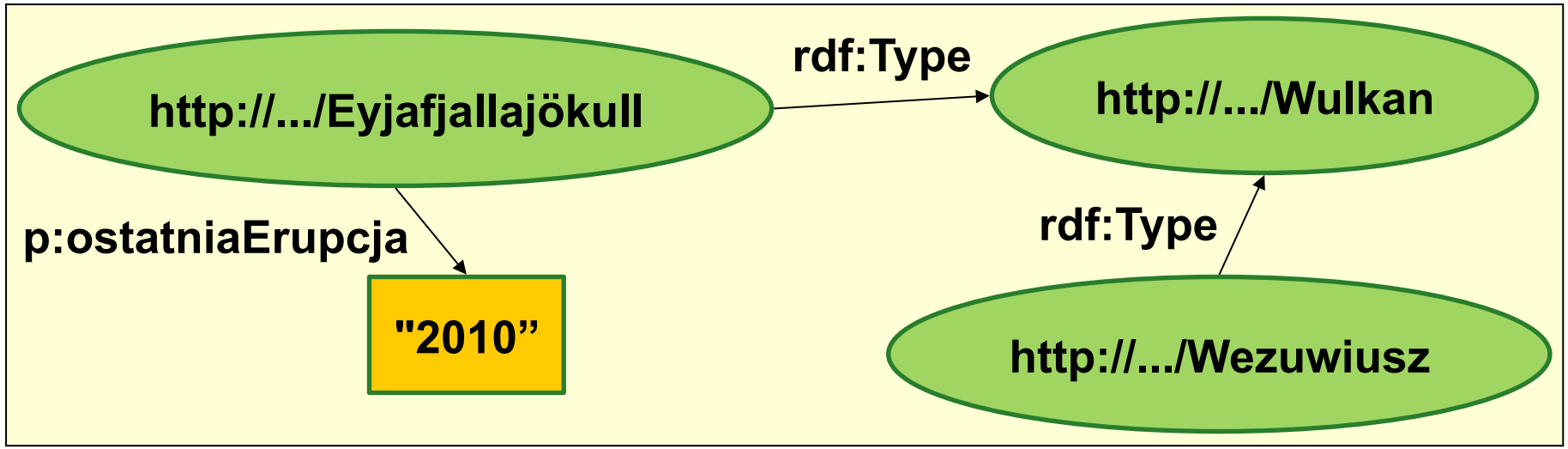
SPARQL – główna idea

Dopasowywanie wzorców

- Opisz podgrafy tych grafów RDF, do których wydawane jest zapytanie
- Podgrafy, które da się dopasować do Twojego opisu konstruują wynik
- Wzorce grafowe - grafy RDF zawierające zmienne

SPARQL – główna idea (przykład)

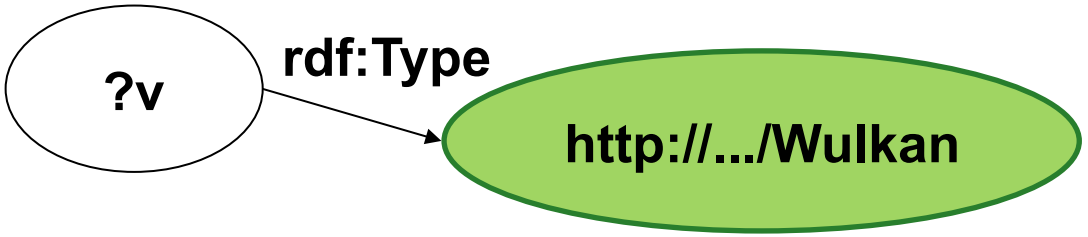
Graf, do którego wydajemy zapytanie:



Wyniki:

?v
http://.../Eyjafjallajökull
http://.../Wezuwiusz

Wzorzec zapytania:



Końcówki SPARQL

Końcówka SPARQL*:*

- **na wejściu przyjmuje zapytania**
- **na wyjściu zwraca wyniki poprzez HTTP**

Rodzaje końcówek:

- **Ogólne – umożliwiają zadawanie zapytań do dowolnych danych RDF dostępnych przez sieć WWW**
- **Dedykowane – związane z konkretnymi zbiorami danych**

Dostęp do końcówki SPARQL

- końcówka SPARQL: usługa sieciowa typu REST
- wydawanie zapytań SPARQL do zdalnej końcówki jest generalnie wydawaniem żądania HTTP GET do tej końcówki z parametrem **query** (przesyłany tekst zapytania)

Zapytanie SPARQL składa się z…

Deklaracji prefiksów dla przestrzeni nazw do skracania URI

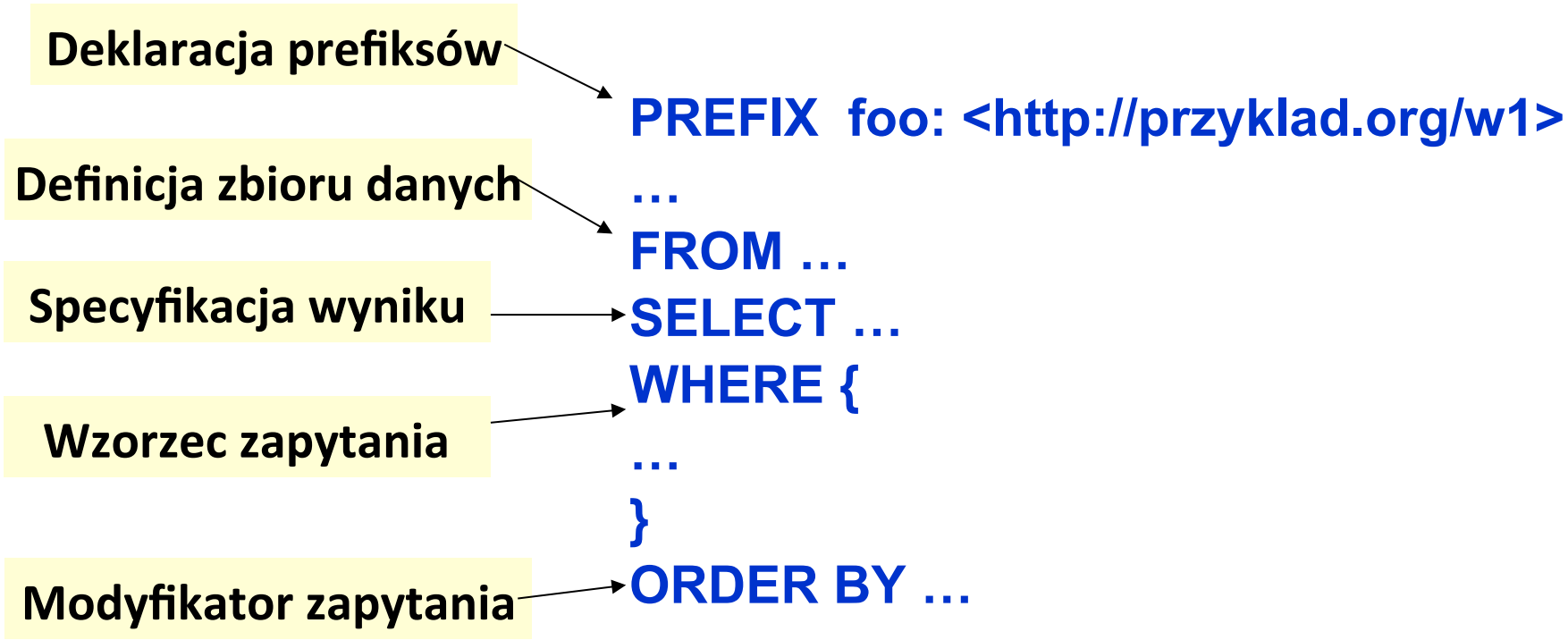
Definicji zbioru danych do wskazania, do którego grafu (grafów) RDF będą wydawane zapytania

Klauzuli specyfikującej wynik do identyfikacji, jaką informację zwrócić z zapytania

Wzorca zapytania do specyfikacji o co pyta zapytanie wydane do zbioru danych

Modyfikatorów zapytania do specyfikacji operatorów np. sortowania wyników i innych sposobów re-aranżacji wyników zapytania

Zapytanie SPARQL składa się z…



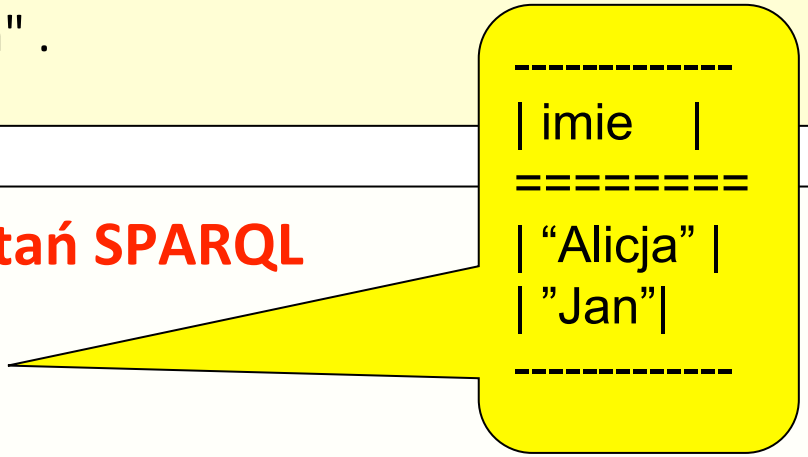
SPARQL – wzorzec trójki

Wzorzec trójki = Trójka RDF z możliwością występowania w niej (nazwanych) zmiennych

```
@prefix osoba: <http://przyklad/osoba/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
osoba:a foaf:name “Alicja" .
osoba:a foaf:mbox <mailto:alicja@example.net> .
osoba:b foaf:name “Jan" .
```

"Hello world" zapytań SPARQL

```
SELECT  ?imie
WHERE
{ ?x  foaf:name  ?imie }
```



SPARQL – podstawowy wzorzec grafu

Podstawowy wzorzec grafu – zbiór wzorców trójek, definiuje kształt grafu

```
@prefix osoba: <http://przyklad/osoba/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
osoba:a foaf:name “Alicja" .
osoba:a foaf:mbox <mailto:alicja@example.net> .
osoba:b foaf:name “Jan" .
```

```
PREFIX osoba: <http://przyklad/osoba/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?imie
WHERE
{ ?osoba foaf:mbox <mailto:alicja@example.net> .
?osoba foaf:name ?imie . }
```

```
------------
| imie     |
========
| “Alicja” |
------------
```

Wieloznacznik

```
@prefix osoba: <http://przyklad/osoba/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
osoba:a foaf:name “Alicja" .
osoba:a foaf:mbox <mailto:alicja@example.net> .
osoba:b foaf:name “Jan" .
```

```
PREFIX osoba: <http://przyklad/osoba/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE
{ ?osoba foaf:mbox <mailto:alicja@example.net> .
?osoba foaf:name ?imie . }
```

```
----------------------
| osoba    | imie     |
==========
| osoba:a | “Alicja” |
----------------------
```

Rozdzielanie trójek w klauzuli WHERE

- zapis standardowy

  a : b c .

  d : e f .
- wspólny podmiot

  a : b c ;

  : e f .
- wspólny podmiot i predykat

  a : b c , f .

Rozdzielanie trójek w klauzuli WHERE - przykład

```
PREFIX osoba: <http://przyklad/osoba/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE
{ ?osoba foaf:mbox <mailto:alicja@example.net> .
?osoba foaf:name ?imie . }
```

```
PREFIX osoba: <http://przyklad/osoba/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE
{ ?osoba foaf:mbox <mailto:alicja@example.net> ,
          foaf:name ?imie . }
```

Zmienne anonimowe

- **nazwane**: głównie różnią się tym od zminennych nieanonimowych tym, że nie są uwzględniane w SELECT * (jednak możliwość odczytania na żądanie)
- **nienazwane** oznaczane przez []

SPARQL – modyfikatory zapytania

Procedury dopasowywania wzorców produkują zbiór rozwiązań.

Zbiór rozwiązań może być zmodyfikowany na wiele sposobów:

- projekcja
- ORDER BY
- LIMIT/OFFSET
- DISTINCT

FILTER

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix magazyn: <http://example.org/magazyn#> .
@prefix inw: <http://example.org/inwentarz#> .
magazyn:ksiazka1 dc:title "SPARQL Query Language Tutorial" .
magazyn:ksiazka1 inw:cena 10 .
magazyn:ksiazka1 inw:ilosc 3 .

magazyn:ksiazka2 dc:title "SPARQL Query Language (2nd ed)" .
magazyn:ksiazka2 inw:cena 20 ; inw:ilosc 5 .

magazyn:ksiazka3 dc:title "Moving from
magazyn:ksiazka3 inw:cena 5 ; inw:ilosc

magazyn:ksiazka4 dc:title "Applying XQu
magazyn:ksiazka4 inw:cena 20 ; inw:ilos
```

```
---------------------------------------------------------------------
| ksiazka                | tytul                                    |
=====================================================================
| magazyn:ksiazka1 | "SPARQL Query Language Tutorial" |
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX magazyn: <http://example.org/magazyn#>
PREFIX inw: <http://example.org/inwentarz#>
SELECT ?ksiazka ?tytul
WHERE {
?ksiazka dc:title ?tytul .
?ksiazka inw:cena ?cena . FILTER ( ?cena < 15 )
?ksiazka inw:ilosc ?ilosc . FILTER ( ?ilosc > 0 ) }
```

FILTER - wbudowane funkcje

Logiczne*:* **!**, **&&**, **||**

Arytmetyczne*:* **+**, **-**, ***, **/**

Porównania*:* **=**, **!=**, **>**, **<**, ...

Testy SPARQL*:* **isURI**, **isBlank**, **isLiteral**, **bound**

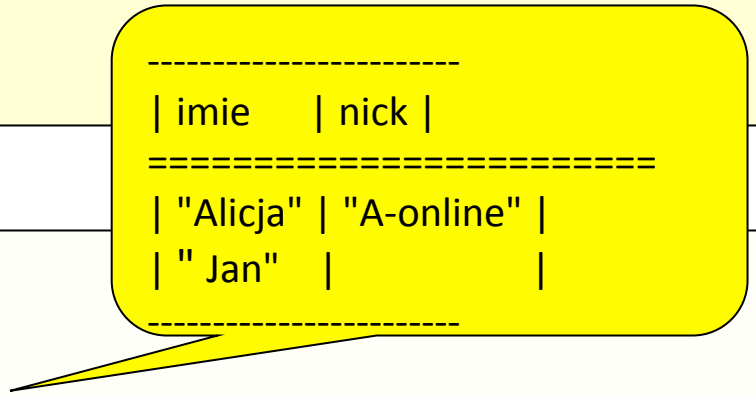
Akcesory SPARQL*:* **str**, **lang**, **datatype**

Inne*:* **sameTerm**, **langMatches**, **regex**

OPTIONAL

```
@prefix osoba: <http://przyklad/osoba/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
osoba:a foaf:name "Alicja" .
osoba:a foaf:nick "A-online" .
osoba:b foaf:name "Jan" .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?imie ?nick
{ ?x foaf:name ?imie .
 OPTIONAL {?x foaf:nick ?nick }
}
```



UNION

```
@prefix ksiazka: <http://przyklad/ksiazka/> .
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
ksiazka:a dc10:title "SPARQL Query Language Tutorial" .
ksiazka:b dc11:title "SPARQL Query Language (2nd
ksiazka:c dc10:title "SPARQL" .
ksiazka:c dc11:title "SPARQL" .
```

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.
SELECT DISTINCT ?tytul
{
{ ?ksiazka dc10:title ?tytul } UNION { ?ksiazka dc11:title ?tytul }
}
```

```
---------------------------------
| tytul |
=================================
| "SPARQL Query Language Tutorial" |
| "SPARQL" |
| "SPARQL Query Language (2nd ed)" |
---------------------------------
```

Źródła danych

- zapytania SPARQL są wykonywane do **zbiorów danych RDF**, składających się z grafów RDF
- jak dotąd zapytania wydawane do **domyślnego grafu**
- zbiory danych RDF składają się z domyślnego grafu i zero lub więcej **nazwanych grafów**, identyfikowanych przez URI
- Nazwane grafy: specyfikowane poprzez klauzule **FROM NAMED**, lub zaszyte w danej końcówce SPARQL
- **GRAPH**: pozwala porcjom zapytania dopasować się do nazwanych grafów w zbiorze RDF, wszystko poza klauzulą GRAPH – dopasowanie do domyślnego grafu

Klauzula FROM - przykład

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alicja" .
_:a foaf:mbox <mailto:alicja@work.example> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?imie
FROM <http://przyklad.org/foaf/alicjaFoaf>
WHERE { ?x foaf:name ?imie }
```

Formaty zapytań i odpowiedzi SPARQL

SELECT

zwraca **tablicę wyników** (wszystkie lub podzbiór zmiennych związanych podczas dopasowywania wzorca zapytania)

ASK

zwraca **wartość logiczną** wskazującą na to czy wzorzec zapytania znajduje dopasowanie czy nie

DESCRIBE

zwraca **graf RDF**, który opisuje strukturę znalezionych zasobów (w sposób zależny od konfiguracji danego procesora– „niespodzianka”)

CONSTRUCT

zwraca **graf RDF** skonstruowany poprzez podstawienie zmiennych w zbiorze szablonów trójek (translacja pomiędzy różnymi grafami RDF)

ASK

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name “Alicja" .
 _:a foaf:homepage <http://work.example.org/Alicja/> .
 _:b foaf:name " Jan" .
_:b foaf:mbox <mailto:bob@work.example> .
```

Yes

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
 ASK    { ?x   foaf:name    “Alicja" }
```

```
<?xml version="1.0"?>
 <sparql xmlns="http://www.w3.org/2005/sparql-results#">
            <head>
            </head>
            <results>
                        <boolean>true</boolean>
            </results>
</sparql>
```

DESCRIBE

```
PREFIX ksiazki:   <http://przyklad.org/ksiazka/>
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
DESCRIBE ?ksiazka
WHERE
  { ?ksiazka dc:title "Harry Potter i więzień Azkabanu" }
```

```
<rdf:RDF>
     <rdf:Description rdf:about="http://przyklad.org/ksiazka/ksiazka3">
          <dc:creator rdf:parseType="Resource">
               <vcard:N rdf:parseType="Resource">
                    <vcard:Given>Joanna</vcard:Given>
                    <vcard:Family>Rowling</vcard:Family>
               </vcard:N>
              <vcard:FN>J.K. Rowling</vcard:FN>
          </dc:creator>
          <dc:title>Harry Potter i więzień Azkabanu</dc:title>
     </rdf:Description>
</rdf:RDF>
```

CONSTRUCT

```
@prefix osoba: <http://przyklad/osoba/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
osoba:a foaf:name "Alicja" .
osoba:a foaf:mbox <mailto:alicja@example.net> .
osoba:b foaf:name " Jan" .
```

```
PREFIX osoba: <http://przyklad/osoba/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT { ?osoba vcard:FN ?imie }
WHERE
{?osoba foaf:name ?imie . }
```

```
@prefix osoba: <http://przyklad/osoba/> .
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
osoba:a vcard:FN "Alicja" .
osoba:b vcard:FN "Jan" .
```

SPARQL 1.0 a SPARQL 1.1

- SPARQL 1.0 – styczeń 2008
  - **SPARQL 1.0 język zapytań**
  - **SPARQL 1.0 protokół**
  - **SPARQL format XML wyników**
- SPARQL 1.1 – listopad 2012
  - **Zaktualizowane wersje SPARQL Query i SPARQL Protocol**
  - **SPARQL 1.1 Update**
  - **SPARQL 1.1 jednolity protokół HTTP do zarządzania grafami RDF**
  - **SPARQL 1.1 opisy usług**
  - **SPARQL 1.1 wnioskowanie**
  - **SPARQL 1.1 zapytania sfederowane**

SPARQL 1.1

Wyrażenia w liście SELECT - wyniki zawierające wartości wyprowadzone ze stałych, wywołań funkcji lub innych wyrażeń

Ścieżki własności – zapytania o dowolnej długości ścieżki w grafie poprzez wyrażenia regularne

Agregacja – grupowanie wyników i obliczanie zagregowanych wartości (np. count, min, max, avg, sum, ...).

Podzapytania – zapytania w zapytaniu

Zapytania sfederowane - rozdzielanie pojedynczego zapytania do wielu końcówek SPARQL i następnie łączenie wyników

Negacja –a) filtrowanie wyników zależne od dopasowania wzorca grafu w kontekście znalezionych wyników b) usuwanie wyników będących w relacji z innym wzorcem

Wnioskowanie (Entailment Regimes) – RDFS, OWL, reguły

Wyrażenia w liście SELECT

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:givenName   "Jan" .
_:a  foaf:surname  „Kowalski" .
```

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ( fn:concat(?G, " ", ?S) AS ?imie_nazw )
WHERE  { ?P foaf:givenName ?G ; foaf:surname ?S }
```

```
----------------------
| imie_nazw |
======================
| „Jan Kowalski" |
----------------------
```

Ścieżki własności

- Dopasuj jedną lub drugą możliwość:

{ :ksiazka1 dc:title | rdfs:label ?lancuch }

- Znajdź imię jakiejkolwiek osoby, którą zna Alicja

{ ?x foaf:mbox <mailto:alicja@przyklad> . ?x foaf:knows/foaf:name ?imie . }

- Imiona ludzi w odległości 2 „linków” "foaf:knows"

{ ?x foaf:mbox <mailto:alicja@przyklad> . ?x foaf:knows/foaf:knows/
foaf:name ?imie . }

{ ?x foaf:mbox <mailto:alicja@przyklad> . ?x foaf:knows{2}/foaf:name ?imie . }

- Dowolna długość ścieżki – imiona wszystkich ludzi, których URI można osiągnąć startując z foaf:knows Alicji:

{ ?x foaf:mbox <mailto:alicja@przyklad> . ?x foaf:knows+/foaf:name ?imie . }

Agregacja

```
@prefix : <http://ksiazki.przyklad/> .
:org1 :afiliuje :aut1, :aut2 .
:aut1 :piszeKsiazke :ksiazka1, : ksiazka2 .
:ksiazka1 :cena 9 .
:ksiazka2 :cena 5 .
:aut2 :piszeKsiazke :ksiazka3 .
:ksiazka3 :cena 7 .
:org2 :afiliuje :aut3 .
:aut3 :piszeKsiazke :ksiazka4 .
:ksiazka4 :cena 7 .
```

```
PREFIX : <http://ksiazki.przyklad/>
SELECT (SUM(?lcena) AS ?total)
WHERE {
  ?org :afiliuje ?aut .
  ?aut :piszeKsiazke ?ksiazka .
  ?ksiazka :cena ?lcena . }
GROUP BY ?org
HAVING (SUM(?lcena) > 10)
```

```
----------------------
| ?total |
======================
| 21 |
----------------------
```

Podzapytania

```
@prefix : <http://people.example/> .

:alicja :name "Alicja", "Alicja Foo", "A. Foo" .
:alicja :knows :jan, :karol .
:jan :name „Jan", „Jan Bar", „J. Bar" .
:karol :name „Karol", „Karol Baz", „K. Baz" .
```

```
PREFIX : <http://people.example/>
SELECT ?y ?minImie
WHERE {
  :alicja :knows ?y .
  {   SELECT ?y (MIN(?imie) AS ?minImie)
    WHERE {
      ?y :name ?imie .
    } GROUP BY ?y
  }
}
```

?y	?minImie
:jan	„J. Bar„
:karol	„K. Baz„

Zapytania sfederowane

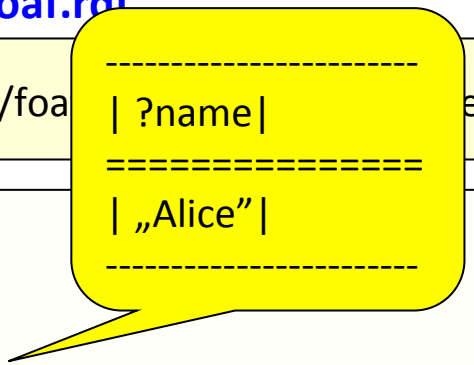
Końcówka SPARQL: http://people.example.org/sparql

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .
   :people15  foaf:name    "Alice" .
   :people16  foaf:name    "Bob" .
   :people17  foaf:name    "Charles" .
   :people18  foaf:name    "Daisy" .
```

Lokalny plik FOAF: http://example.org/myfoaf.rdf

```
<http://example.org/myfoaf/I> <http://xmlns.com/foa                    example.org/people15> .
```

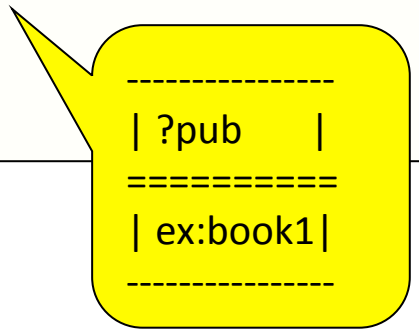
```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.org/myfoaf.rdf>
WHERE {
  <http://example.org/myfoaf/I> foaf:knows ?person .
  SERVICE <http://people.example.org/sparql> {
    ?person foaf:name ?name . }
}
```



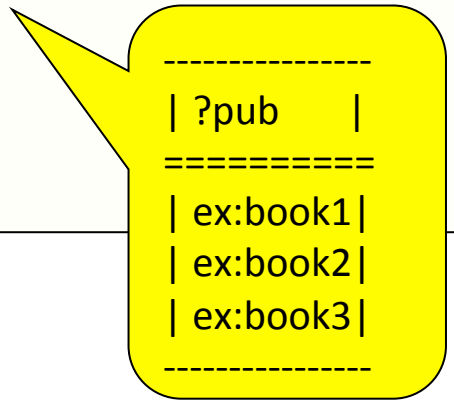
Wnioskowanie

```
@prefix ex: <http://publications.example/> .

ex:book1 rdf:type ex:Publication .
ex:book2 rdf:type ex:Article .
ex:Article rdfs:subClassOf ex:Publication .
ex:publishes rdfs:range ex:Publication .
ex:MITPress ex:publishes ex:book3 .
```

```
@prefix ex: <http://publications.example/> .

SELECT ?pub
WHERE {
  ?pub rdf:type ex:Publication
}
```



RDF-entailment



RDFS-entailment

Silniki zapytań/repozytoria trójek

- **Jena**
- **AllegroGraph**
- **Sesame**
- **OpenLink Virtuoso**
- **Blazegraph**

Repozytoria trójek

- **Specjalizowana baza danych do przechowywania trójek RDF**
  - Trójki wprowadzane w różnych formatach
- **Wspiera język zapytań do RDF**
  - SPARQL (rekomendacja W3C)
  - Inne języki (np. ARQ, RDQL)
  - Może, ale nie musi wykonywać wnioskowanie

Architektury

- **Implementacja: *In-memory*, Natywna, Nie-natywna**
- **Natywna: JENA TDB, Sesame Native, Virtuoso, AllegroGraph, Oracle 11g**
- **Nie-Natywna: Np. Używająca RDBMS jako *backend***

Materiały:

SPARQL specification document, http://www.w3.org/TR/rdf-sparql-query/

SPARQL By Example, A Tutorial, Lee Feigenbaum,
http://www.cambridgesemantics.com/semantic-university/sparql-by-example

„Querying semantic data“:
http://www.linkeddatatools.com/querying-semantic-data

Quads i Grafy Nazwane (Named Graphs)

- Wiele repozytoriów trójek wspiera
  "czwórki" (quads)
  dla nazwanych grafów (named graphs)

A named graph is just an RDF with a
  URI name often called the *context*

Such a triple store divides its data a default graph and
  zero or more additional named graphs

SPARQL has support for named graphs

De facto standards exist for representing quad data,
  e.g., n-quads and TriG (a turtle/N3 variant)

AllegroGraph stores quints (S,P,O,C,ID), the ID can be
  used to attach metadata to a triple