第3回 RDF 講習会

SPARQLの基礎

山本 泰智
@yayamamo
ライフサイエンス統合データベースセンター

2018.11.1 @ JST

SPARQL?
(スパークル)

RDFデータを検索する
問い合わせ言語

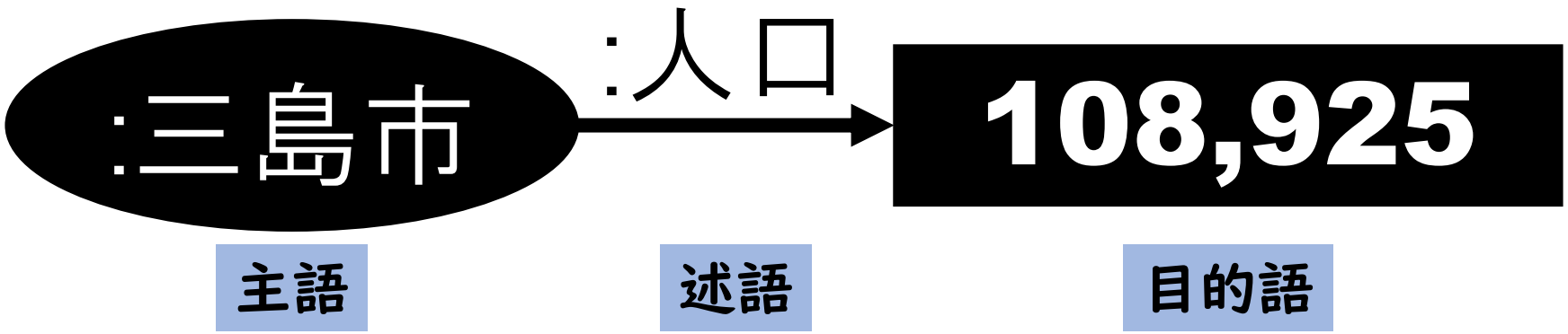
穴埋め問題+α

RDFおさらい

RDFは文

三島市の人口は108,925人です。

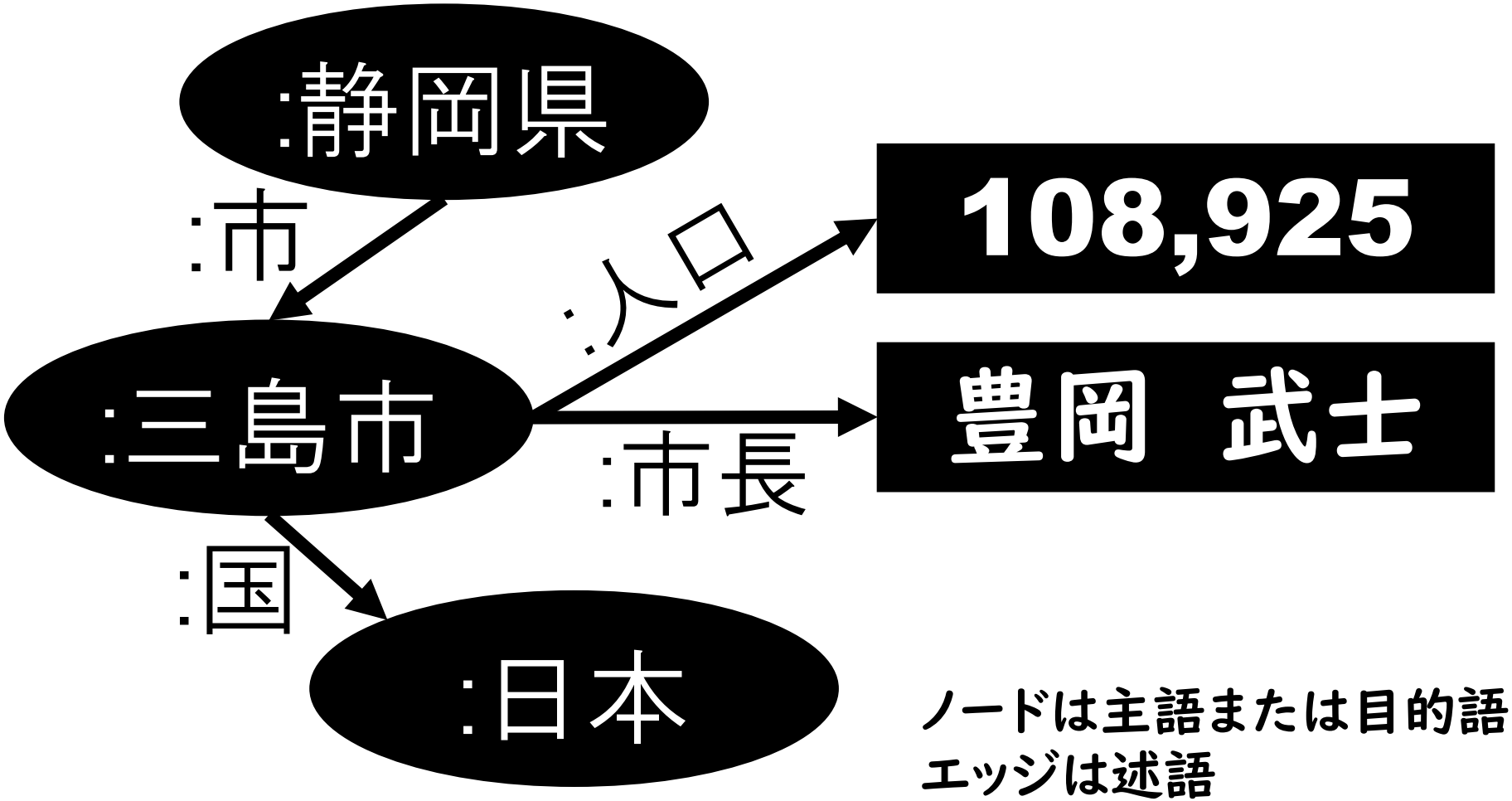
平成30年4月1日現在



RDF: Resource Description Framework

全てのデータを主語、述語、目的語からなる文（トリプル）で表現

RDFはグラフ



ノードは主語または目的語
エッジは述語

全てURIかリテラル

:三島市

URI

Uniform Resource Identifier

コロン（:）で始まるものはURIの省略表記

ネット上での識別子

http://ja.dbpedia.org/resource/三島市

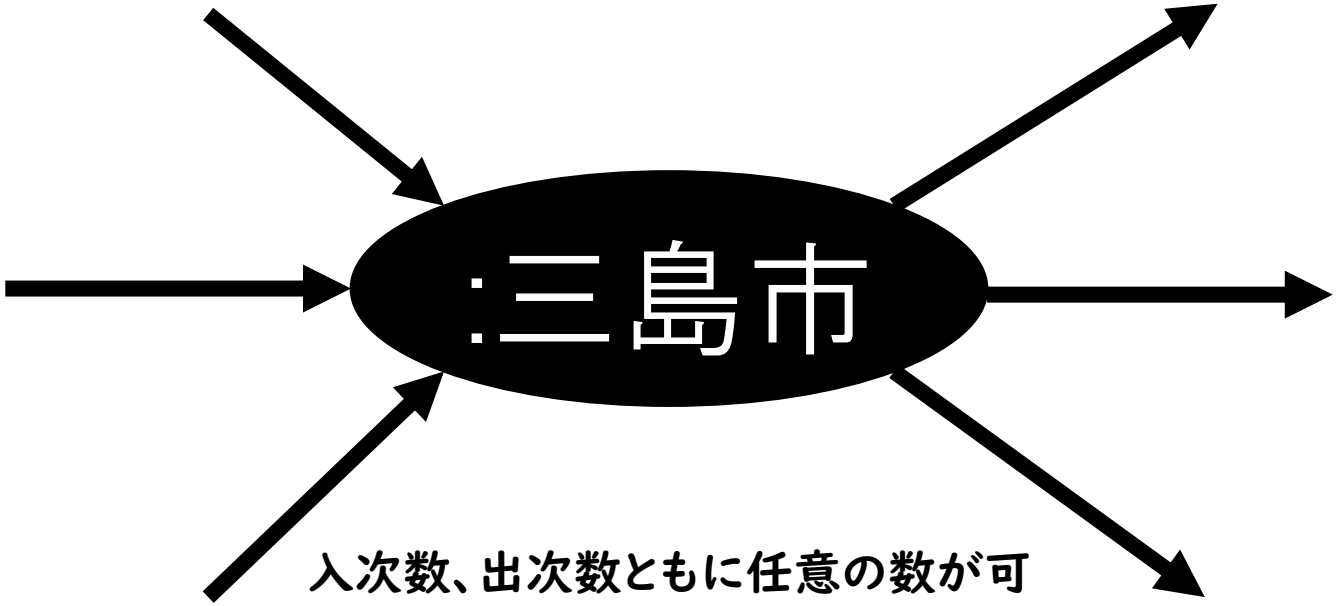
http://www.wikidata.org/entity/Q653478

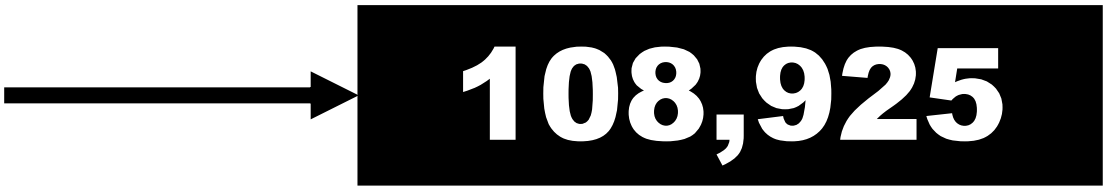
108,925 リテラル

具体的な文字や数値などの値

目的語はURIかリテラル、主語と述語はURIのみ

URIとリテラル





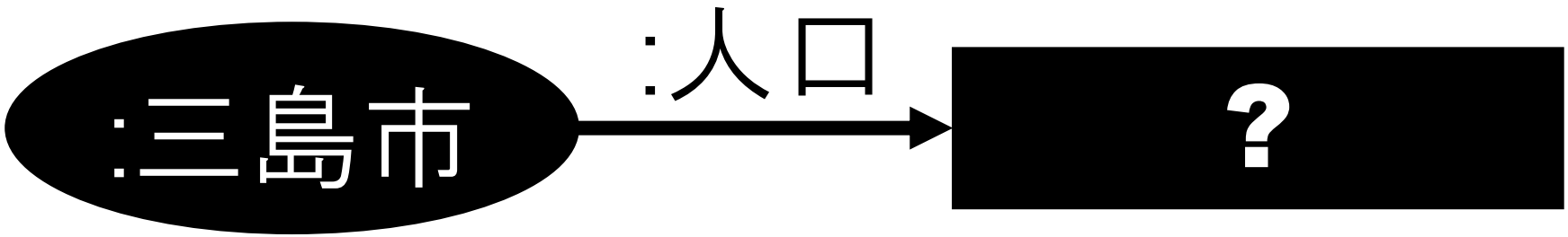
入次数=1、出次数=0のみ

いよいよSPARQL

穴埋め問題+α

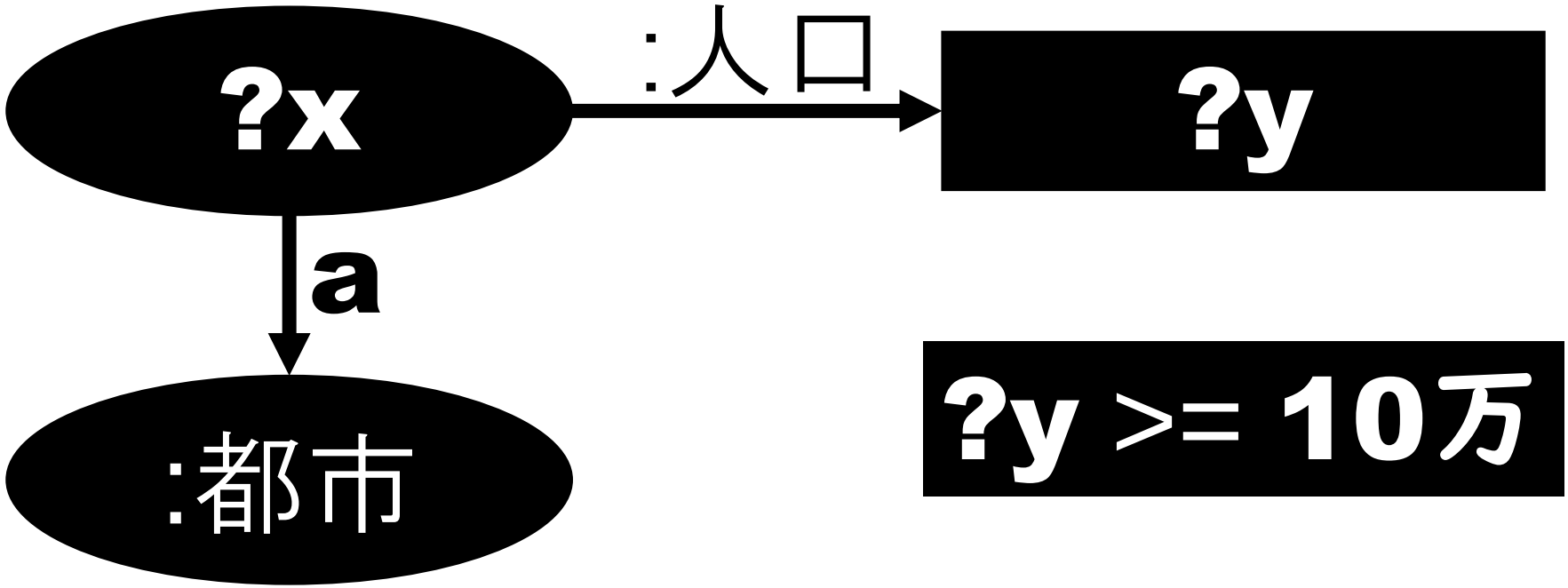
RDFを検索

三島市の人口は ？ 人です。



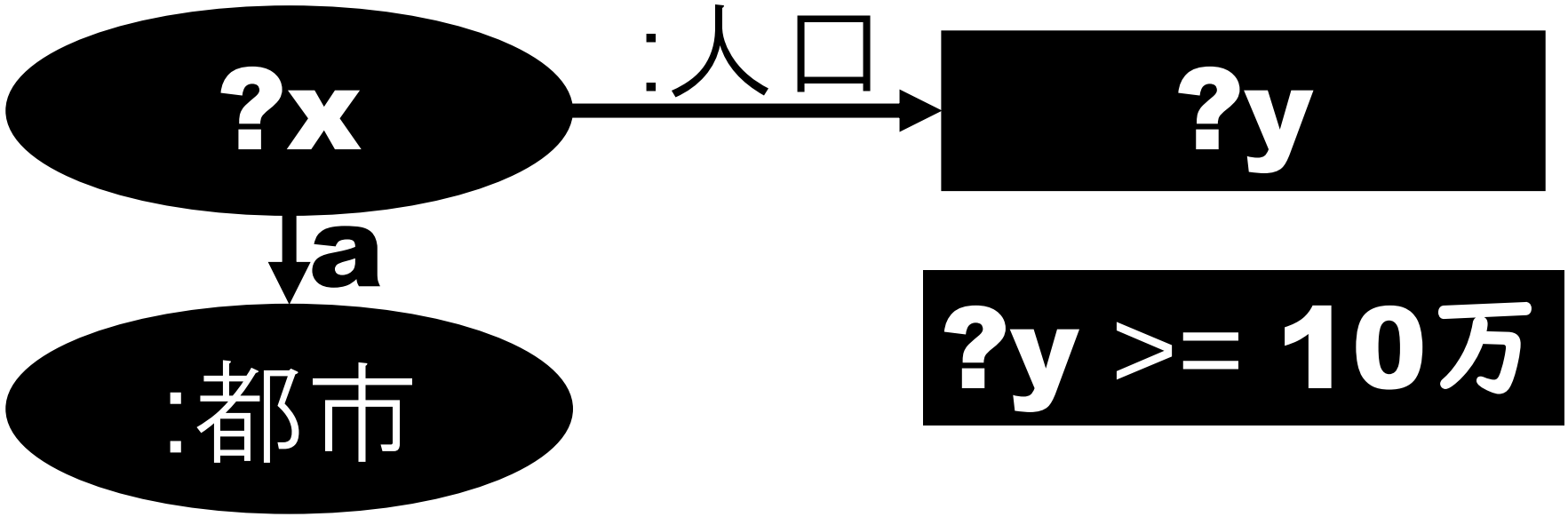
RDFを検索

人口が10万人以上の都市は?



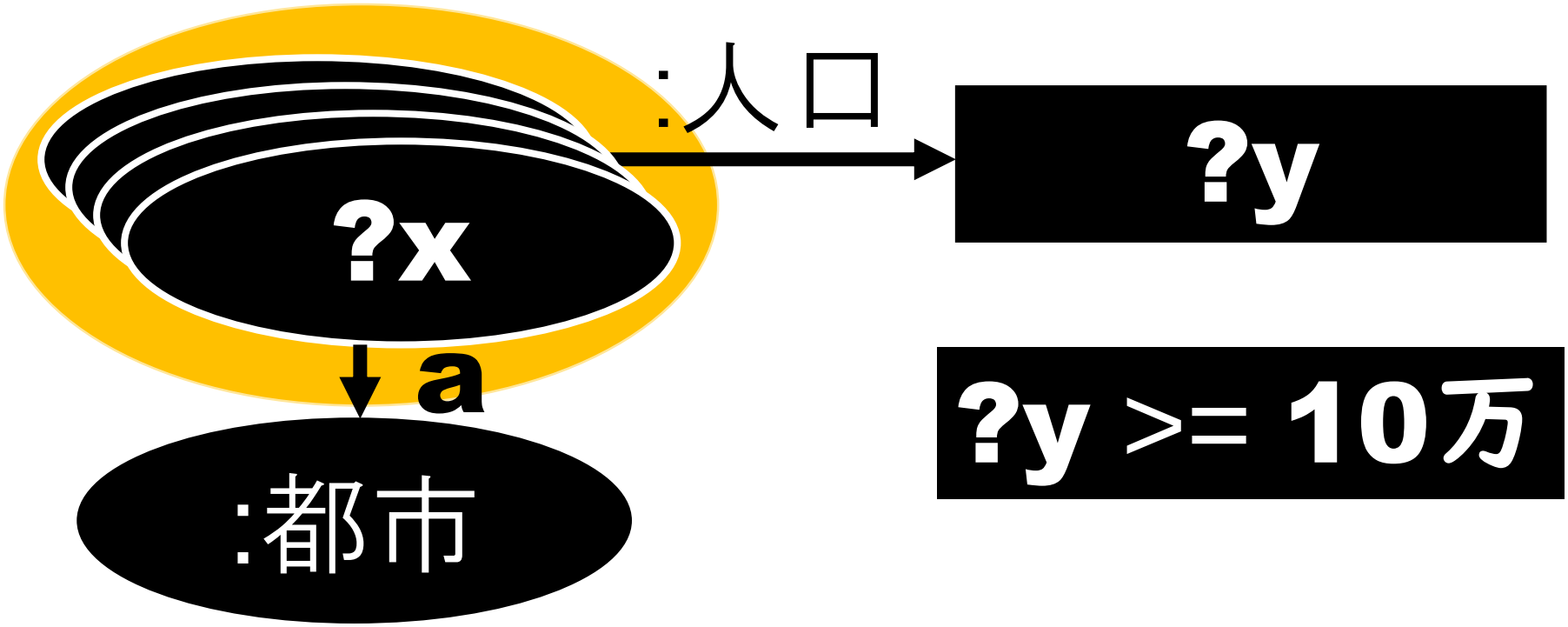
RDFを検索

?x は都市で、人口は ?y 人です。
ただし、人口は10万人以上です。



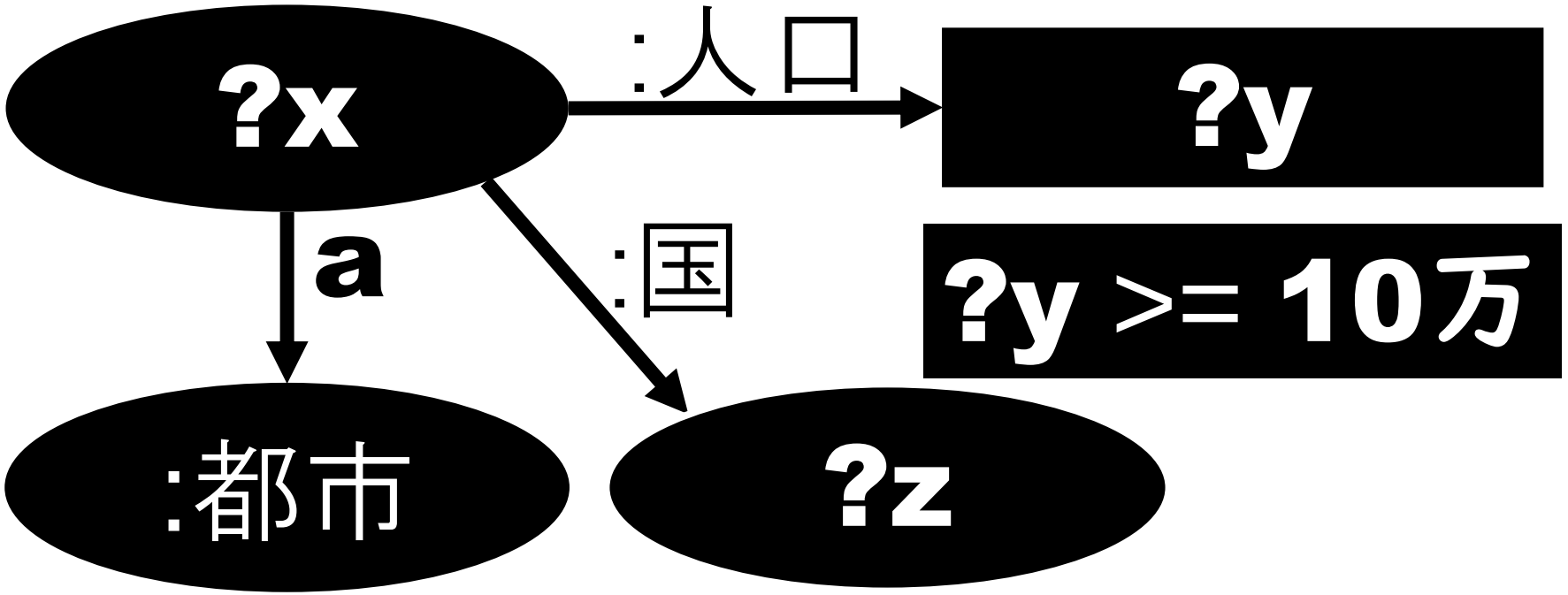
RDFを検索

人口が10万人以上の都市はいくつある？



RDFへ質問

人口が10万人以上の都市がある国は?



SPARQLで表現する

SPARQL1.1

W3C

Recommendation

仕様は11の文書

1. SPARQL 1.1 Overview
2. SPARQL 1.1 Query Language
3. SPARQL 1.1 Update
4. SPARQL 1.1 Service Description
5. SPARQL 1.1 Federated Query
6. SPARQL 1.1 Query Results JSON Format
7. SPARQL 1.1 Query Results CSV and TSV Formats
8. SPARQL Query Results XML Format (Second Edition)
9. SPARQL 1.1 Entailment Regimes
10. SPARQL 1.1 Protocol
11. SPARQL 1.1 Graph Store HTTP Protocol

問い合わせ言語

- 参照系（RDFストアの内容を変えない）
- 更新系（RDFストアの内容を更新する）
- 大文字小文字の違いを無視

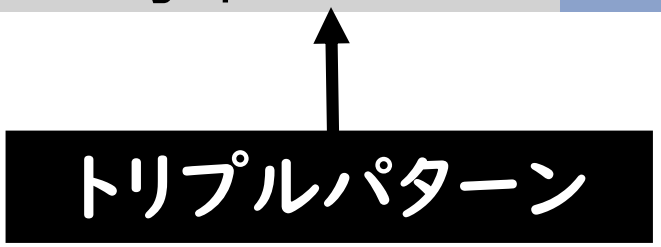
参照系

- **SELECT** ←――― 今日扱う範囲
- **ASK**
- **DESCRIBE**
- **CONSTRUCT**

SELECT

欲しいデータのパターンを記述
結果は変数に関連付けられる

```
SELECT ?o WHERE
{
    :三島市  :人口  ?o .
}
```



?o: 変数, $oでも同じ

FILTER

トリプルパターンと併せて
より詳細なマッチ条件を記述できる

```
SELECT ?c
WHERE {
   ?c :人口 ?o .
   ?c a    :都市 .
   FILTER(?o >= 100000)
}
```

aは**rdf:type**の簡易表記

COUNT / AS

検索にマッチした件数を数えられる

```
SELECT (COUNT(?c) AS ?n)
WHERE {
   ?c :人口 ?o .
   ?c a     :都市 .
   FILTER(?o >= 100000)
}
```

DISTINCT

検索結果の重複を取り除ける

```
SELECT DISTINCT ?n
WHERE {
   ?c :人口 ?o .
   ?c a     :都市 .
   ?c :国    ?n .
   FILTER(?o >= 100000)
}
```

UNION

検索条件に、「または」を含められる

```
SELECT ?c ?o
WHERE {
   ?c :人口 ?o .
   { ?c a :町 . }
   UNION
   { ?c a :村 . }
   FILTER(?o >= 1000)
}
```

OPTIONAL

柔軟な検索条件を与えられる

```
SELECT DISTINCT ?n ?m
WHERE {
  ?c :人口 ?o ;
     a     :都市 ;
     :国 ?n .
  OPTIONAL {
    ?c :首長 ?m .
  }
  FILTER(?o >= 100000)}
```

GROUP BY

検索結果を集計できる

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
   ?c :標高 ?h ;
      a    :山 ;
      :国   ?n .
   FILTER(?h > 2000)
} GROUP BY (?n)
```

HAVING

集計結果を条件にできる

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
   ?c :標高 ?h ;
      a    :山 ;
      :国   ?n .
   FILTER(?h > 2000)
} GROUP BY ?n
HAVING (?nc > 1)
```

ORDER BY

検索結果の表示順を決められる

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
   ?c :標高 ?h ;
      a    :山 ;
      :国   ?n .
   FILTER(?h > 2000)
} GROUP BY ?n
HAVING (?nc > 1)
ORDER BY DESC(?nc)
```

DESC: 降順
ASC: 昇順
省略時は**ASC**

LIMIT / OFFSET

検索結果を切り出せる

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
   ?c :標高 ?h ;
      a    :山 ;
      :国   ?n .
   FILTER(?h > 2000)
} GROUP BY (?n)
ORDER BY DESC(?nc)
LIMIT 10 OFFSET 0
```

GROUP BYなど解修飾子の順序

GROUP BY / HAVING / ORDER BY / LIMIT, OFFSET

すべて省略できるが、順序は変えられない

ただし、**LIMIT, OFFSET**は入れ替え可
（**OFFSET**を先にしても**OK**）

注意点

リテラルの型と言語タグ

```
SELECT ?c
WHERE {
   ?c :標高 "1234"^^xsd:integer;
      a     :山 ;
      :国    "日本"@ja. }
```

"1234"^^xsd:integer は 1234 と同値

"123.4"^^xsd:decimal は 123.4 と同値

"日本"@ja のjaは言語タグ

文字列の扱い

●単純リテラル

"日本"

●プレインリテラル

"日本"@ja

単純リテラルはxsd:string型リテラルの簡易表現
"日本" と "日本"^^xsd:string は同値

単純リテラルとプレインリテラルは互いに異なるデータとして扱われることに注意

PREFIX

RDFはURIもしくはリテラルで表記される。
URIは長くなり易く人可読性が悪くなりがち。

特定のIDを示す部分を除く共通部分をくくりだして
予めPREFIXとして宣言。
パターンを表記する際の可読性を上げる。

PREFIX : <http://example.com/>

:三島市 は <http://example.com/三島市> と同値

よく使われるPREFIXと語彙

- **rdf**: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  - **rdf:type**
- **rdfs**: <http://www.w3.org/2000/01/rdf-schema#>
  - **rdfs:label**
  - **rdfs:seeAlso**
- **owl**: <http://www.w3.org/2002/07/owl#>
  - **owl:sameAs**
- **foaf**: <http://xmlns.com/foaf/0.1/>
  - **foaf:name**
  - **foaf:homepage**
- **geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>**
  - **geo:lat**
  - **geo:long**

省略可能な部分

WHEREは省略OK*

```
SELECT ?c {
  ?c :人口 ?o .
  ?c a     :都市 .
}
```

最後のトリプルのパターンのピリオドは省略OK

```
SELECT ?c {
  ?c :人口 ?o .
  ?c a     :都市
}
```

* CONSTRUCT（発展編）の一形態を除く

評価順序

SPARQLクエリは論理式。
複数のトリプルのパターンの列挙はそれらのパターンの論理積を意味する。
したがって順序は問わない。

```
{
  ?c :人口 ?o .
  ?c a :都市 .
  ?c :国 ?n .
}
```

```
{
  ?c a :都市 .
  ?c :国 ?n .
  ?c :人口 ?o .
}
```

DBpedia Japaneseで試す

http://ja.dbpedia.org/sparql

上記ページに書かれているサンプルを試す。
その後、適宜書き換えて試す。

Linked Dataとの関係

http://ja.dbpedia.org/resource/%E4%B8%89%E5%B3%B6%E5%B8%82
へのアクセスと

http://ja.dbpedia.org/sparql で以下のクエリを発行

DESCRIBE <http://ja.dbpedia.org/resource/三島市>

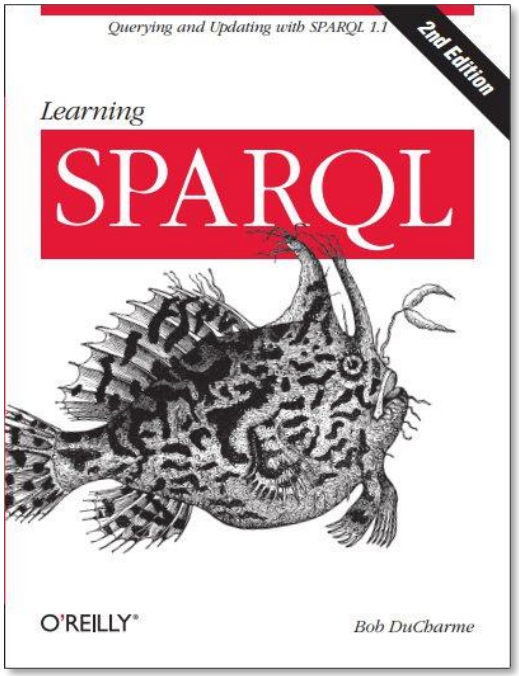
関連ツール

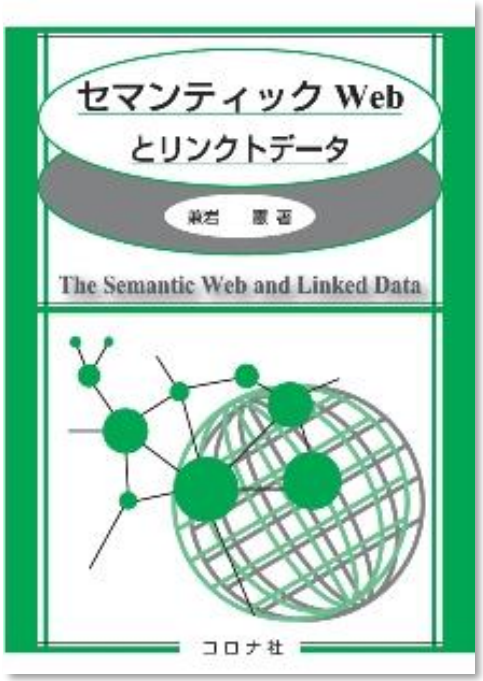
- **YASGUI**
  クエリ構築を支援
  http://yasgui.org/
- **Sparklis**
  クエリを明示的に書かずにデータを閲覧
  http://www.irisa.fr/LIS/ferre/sparklis/osparklis.html
- **Fuseki**
  お手元でSPARQLを試すのに最適
  http://jena.apache.org/documentation/fuseki2/index.html
- **SPARQLES**
  様々なエンドポイントを複数の見地から定期的に調査
  http://sparqles.ai.wu.ac.at/

生命科学分野での応用例

- **http://togogenome.org/**
- **http://wikigenomes.org/**

参考文献


Querying and Updating with SPARQL 1.1
2nd Edition
Learning
SPARQL
O'REILLY
Bob DuCharme


セマンティック Web
とリンクトデータ
The Semantic Web and Linked Data
コロナ社


インプレスR&D【NextPublishing】
オープンデータ時代の標準Web API
SPARQL
加藤 文彦
川島 秀一
岡野原 陽子
山本 泰智
片山 俊明
データのマッシュアップが加速する!
RDF用クエリ言語 SPARQL
日本初の解説書登場

情報共有の場

DBCLSはハッカソンを定期的に開催しています

参加自由ですので、ぜひご検討ください!

http://wiki.lifesciencedb.jp/mw/SPARQLthon

発展編

SERVICE

他のエンドポイントに検索を依頼し、結果を統合できる

```
SELECT ?h ?lat ?long
WHERE {
   ?c :標高 ?h ;
      a     :山 .
   SERVICE <location> {
     ?c :緯度 ?lat ;
        :経度 ?long .
   }
   FILTER(?h > 2000)}
```

BIND / VALUES

特定の変数に特定の値を代入できる

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
  ?c :標高 ?h ;
     a     :山 ;
     :国   ?n .
  FILTER(?h > 2000)
  BIND(?n AS "日本"@ja)
}
```

VALUES

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
  ?c :標高 ?h ;
     a     :山 ;
     :国   ?n .
  FILTER(?h > 2000)
  VALUES ?n
    { "日本"@ja "ブラジル"@ja }
}
```

VALUES

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
  ?c :標高 ?h ;
     a     ?t ;
     :国   ?n .
  FILTER(?h > 2000)
  VALUES (?t ?n)
    { (:山 "日本"@ja)
      (:都市 "ネパール"@ja) }
}
```

MINUS

特定の条件にマッチするデータを取り除ける

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
   ?c :標高 ?h ;
      a     :山 ;
      :国   ?n .
   FILTER(?h > 2000)
   MINUS {
      ?c :国 "日本"@ja .
   } }
```

MINUSの中のパターンに、外のパターンと共通する変数が一つは必要

MINUSとNOT EXISTS

```
SELECT (COUNT(?n) AS ?nc) ?n
WHERE {
   ?c :標高 ?h ;
      a     :山 ;
      :国   ?n .
   FILTER(?h > 2000)
   FILTER NOT EXISTS {
      ?c :国 "日本"@ja .
   } }
```

内外で共通する変数がある限りMINUSと同じ作用

サブセレクト

SELECTは入れ子にできる

```
SELECT ?n ?c WHERE {
   ?c a :山 ; :標高 ?mh ; :国 ?n .
   {SELECT ?n (MAX(?h) AS ?mh)
      WHERE {
       [] :標高 ?h ;
          a     :山 ;
          :国   ?n .
      } GROUP BY ?n
   }}
```

国ごとに最高峰の標高のみを取得

各国の最高峰をリストアップする

プロパティパス

述語のパターンを柔軟に記述できる

```
SELECT ?who
WHERE {
   {
      :私 :友 ?who .
   } UNION {
      :私 :友/:友 ?who .
   }
}
```

```
SELECT ?who
WHERE {
   :私 :友+ ?who .
}
```

? + * / ! ^ | ()を利用して構築

よく使われる文字列関連関数

- **CONTAINS**
- **STRSTARTS**
- **STRENDS**
- **REGEX**

```
SELECT DISTINCT ?c ?n
WHERE {
   ?c :人口 ?o ;
      :国 ?n .
   FILTER(STRSTARTS(?n,"ab"))
}
```

REGEX（正規表現）

FILTERと共に利用して文字列マッチ

```
SELECT ?c
WHERE {
   ?c :標高 "1234"^^xsd:integer ;
      a     :山 ;
      :国    "日本"@ja .
   FILTER REGEX(str(?c),"岳$","s")
}
```

ここでは不必要だが、REGEXの記法を紹介するため

REGEX REPLACE

REPLACEと共に利用して文字列置換

```
SELECT ?c ?label
WHERE {
  ?c :標高 "1234"^^xsd:integer ;
     a     :山 ;
     :国   "日本"@ja .
  BIND(
    REPLACE(str(?c),"岳$","嶽")
        AS ?label)
}
```

ASK

欲しいデータのパターンを与える
結果はパターンにマッチするデータがあるか否か
TRUE / FALSE

```
ASK WHERE
{
  :三島市 :人口 ?○ .
}
```

CONSTRUCT

欲しいデータと生成されるデータのパターンを与える
結果は生成パターンに基づく新たなRDFデータ

```
CONSTRUCT {
   ?c a      :高山 ;
      :標高 ?h .
} WHERE {
   ?c :標高 ?h ;
      a     :山 .
   FILTER(?h > 2000)
}
```

DESCRIBE

欲しいデータのURIを与える
結果は当該URIに関する有益なRDFデータ

DESCRIBE ：三島市

RDFデータのURIにウェブブラウザなどでアクセスした際に表示させるデータの生成に最適

トリプルからクワッドへ

一連のRDFトリプルの集合を一つのグラフとして扱い、名前を付けられる

グラフ名もURIなので、主語、述語、目的語、グラフ名の四つ組、すなわちクワッドをデータの最小単位とする

SPARQL1.1からグラフの概念が導入された

シリアライズ方式としてN-quadがある
（.nq 形式のファイル）

GRAPH / FROM

```
SELECT ?o WHERE {
   GRAPH <graph> {
   :三島市  :人口  ?o .
}}
```

```
SELECT ?o
FROM <graph>
WHERE {
   :三島市  :人口  ?o .
}
```

GRAPHの扱いについて注意

グラフ名を指定しない場合のトリプルは
DEFAULT GRAPHに含まれる

グラフ名を陽に指定しないで検索した場合には

1. 全てのグラフを検索対象にする
2. DEFAULT GRAPHだけを検索対象にする

のいずれかで実装により異なる

プログラムからアクセス

- **Python**: SPARQLWrapper
  https://rdflib.github.io/sparqlwrapper/
- **R**: SPARQL Package
  https://cran.r-project.org/web/packages/SPARQL/index.html
- **Ruby**: SPARQL Client for RDF.rb
  https://github.com/ruby-rdf/sparql-client
- **Java**: Apache Jena
  http://jena.apache.org/
- **Java**: Eclipse RDF4J
  http://rdf4j.org/
- **JavaScript**: SPARQL.js etc.
  https://github.com/RubenVerborgh/SPARQL.js

curlでアクセス

SPARQLの仕様はHTTPを用いたAPIも含む

HTTP GETによるアクセス例

```
$ curl -gLH 'Accept: text/tab-separated-values'
'http://ja.dbpedia.org/sparql?
query=select+distinct+*+where+{+%3Chttp%3A%2F%2Fja.dbpedia.org%2Fr
esource%2F%E6%9D%B1%E4%BA%AC%E9%83%BD%3E+%3Fp+%3Fo+.++}+LIMIT+100'
```

仕様ではクエリ無しでHTTP GETするとエンドポイントのメタデータが得られることになっているが…

curlでアクセス

HTTP POSTでのアクセス例

```
$ echo ¥
'select distinct * where { <http://ja.dbpedia.org/resource/東京都> ?p ?o .  } LIMIT 100' ¥
| curl -gLH 'Accept: text/tab-separated-values' ¥
--data-urlencode query@- 'http://ja.dbpedia.org/sparql'
```