UE 803 - Data Science

Session 6: Linked Data

Yannick Parmentier - Université de Lorraine / LORIA

Introduction

- What we have seen so far:
  - **retrieving** data from webpages or webservices
  - **storing** data in (distributed or relational) databases
  - **querying** data (using javascript or SQL)
- Today:
  - yet another data+storage type: **Linked Open Data**

Linked (Open) Data

Linked (Open) Data

> **Structured data** which is **interlinked** with other data so it becomes more useful through **semantic queries**.

(Source: wikipedia)

Linked (Open) Data

> **Structured data** which is **interlinked** with other data so it becomes more useful through **semantic queries**.

(Source: wikipedia)

- builds upon **standard Web technologies** such as HyperText Transfer Protocol (HTTP) and Unified Resource Identifiers (URIs)
- instead of merely serving data (as web pages) for human readers, **sharing information in a way that can be read automatically by computers**

Principles of Linked Data (Berners-Lee, 2006)

Principles of Linked Data (Berners-Lee, 2006)

1. Use URIs to **name / identify things** (data).

Principles of Linked Data (Berners-Lee, 2006)

1. Use URIs to **name / identify things** (data).
2. Use **HTTP URIs** so that these things can be **looked up** / *interpreted*.

Principles of Linked Data (Berners-Lee, 2006)

1. Use URIs to **name / identify things** (data).
2. Use **HTTP URIs** so that these things can be **looked up** / *interpreted*.
3. **Provide useful information** about what a name identifies **using open standards**.

Principles of Linked Data (Berners-Lee, 2006)

1. Use URIs to **name / identify things** (data).
2. Use **HTTP URIs** so that these things can be **looked up** / *interpreted*.
3. **Provide useful information** about what a name identifies **using open standards**.
4. **Refer to other things** using their HTTP URI-based names **when publishing data** on the Web.

Outline

Linked data

Some linked data warehouses

Some linked data projects

- **DBpedia**
  - extracted data from Wikipedia
  - about 3.4 million concepts
- **FOAF** (*Friend Of A Friend*)
  - descriptions of persons, their properties and relationships
- **GeoNames**
  - descriptions of more than 7,500,000 geographical features worldwide

Some linked data projects (continued)

- **Wikidata**
  - a collaborative knowledge base
  - central storage for the structured data of its Wikimedia Foundation sister projects
- **GRID** (*Global Research Id Database*)
  - 89,506 academic institutions
- **YAGO** (*Yet Another Great Ontology*)
  - 10 million entities and 120+ million facts about these entities

Representing / structuring linked data

The RDF framework

The RDF framework

Resource Description Framework

> Family of World Wide Web Consortium **specifications** originally designed as a **metadata *data model***.

(Source: wikipedia)

The RDF framework

Resource Description Framework

> Family of World Wide Web Consortium **specifications** originally designed as a **metadata *data model***.

(Source: wikipedia)

- **Data model** for representing information about **resources** available on the **web** and their **cross-references**
- Model primarily **intended for applications**

Identifying resources

- **Uniform Resource Identifier (URI)**: a string that unambiguously identifies a particular resource

Identifying resources

- **Uniform Resource Identifier (URI)**: a string that unambiguously identifies a particular resource

- A URI is written using the following **syntactic pattern**:

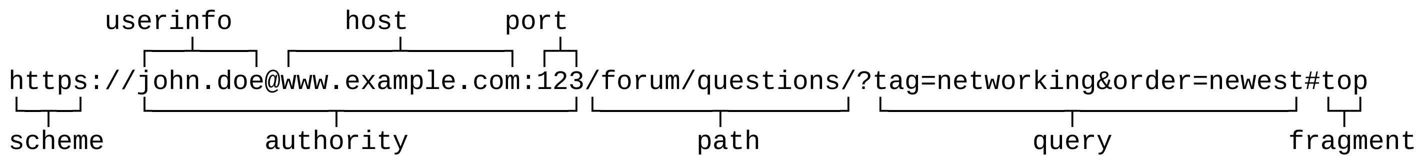
```
scheme:[//authority]path[?query][#fragment]
```

Identifying resources

- **Uniform Resource Identifier (URI)**: a string that unambiguously identifies a particular resource
- A URI is written using the following **syntactic pattern**:

```
scheme:[//authority]path[?query][#fragment]
```

- Example:


userinfo
host
port
https://john.doe@www.example.com:123/forum/questions/?tag=networking&order=newest#top
scheme
authority
path
query
fragment

Identifying resources

- **Uniform Resource Identifier (URI)**: a string that unambiguously identifies a particular resource

- A URI is written using the following **syntactic pattern**:

  `scheme:[//authority]path[?query][#fragment]`

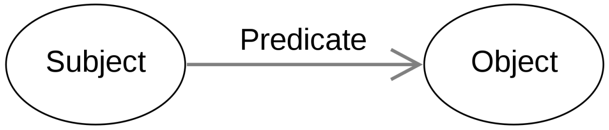
- Example:

```
          userinfo       host      port
          ┌───┴──┐ ┌──────┴──────┐ ┌┴┐
https://john.doe@www.example.com:123/forum/questions/?tag=networking&order=newest#top
└─┬─┘   └─────────────┬────────────┘└───────┬───────┘ └────────────┬────────────┘ └┬┘
scheme          authority                  path                  query           fragment
```

- A **URI reference** may be absolute or relative (to another URI)

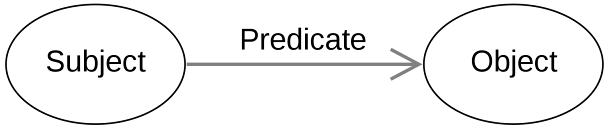
Linking resources

- Each **statement** about a resource (i.e., piece of knowledge) is of the form ***subject-predicate-object*** and represented by a triple of **URIs** (and / or **literals**) and is included in a so-called **RDF graph**:



Linking resources

- Each **statement** about a resource (i.e., piece of knowledge) is of the form ***subject-predicate-object*** and represented by a triple of **URIs** (and / or **literals**) and is included in a so-called **RDF graph**:



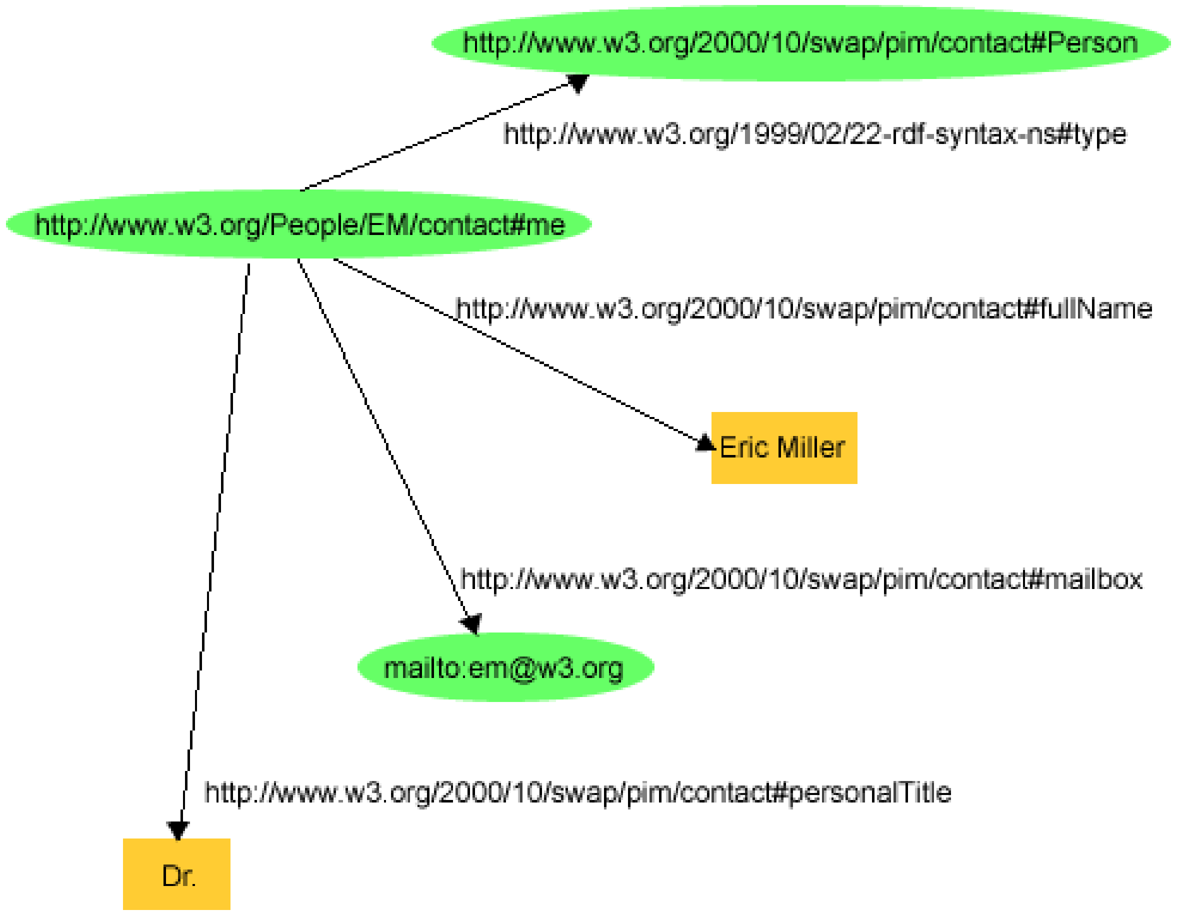
- **Example**:

  "Michel Gagnon works at Polytechnique Montréal."

```
( <https://www.polymtl.ca/Profs#MichelGagnon>,
  <https://www.polymtl.ca/vocab#worksAt>,
  <https://www.polymtl.ca/gigl/> )
```

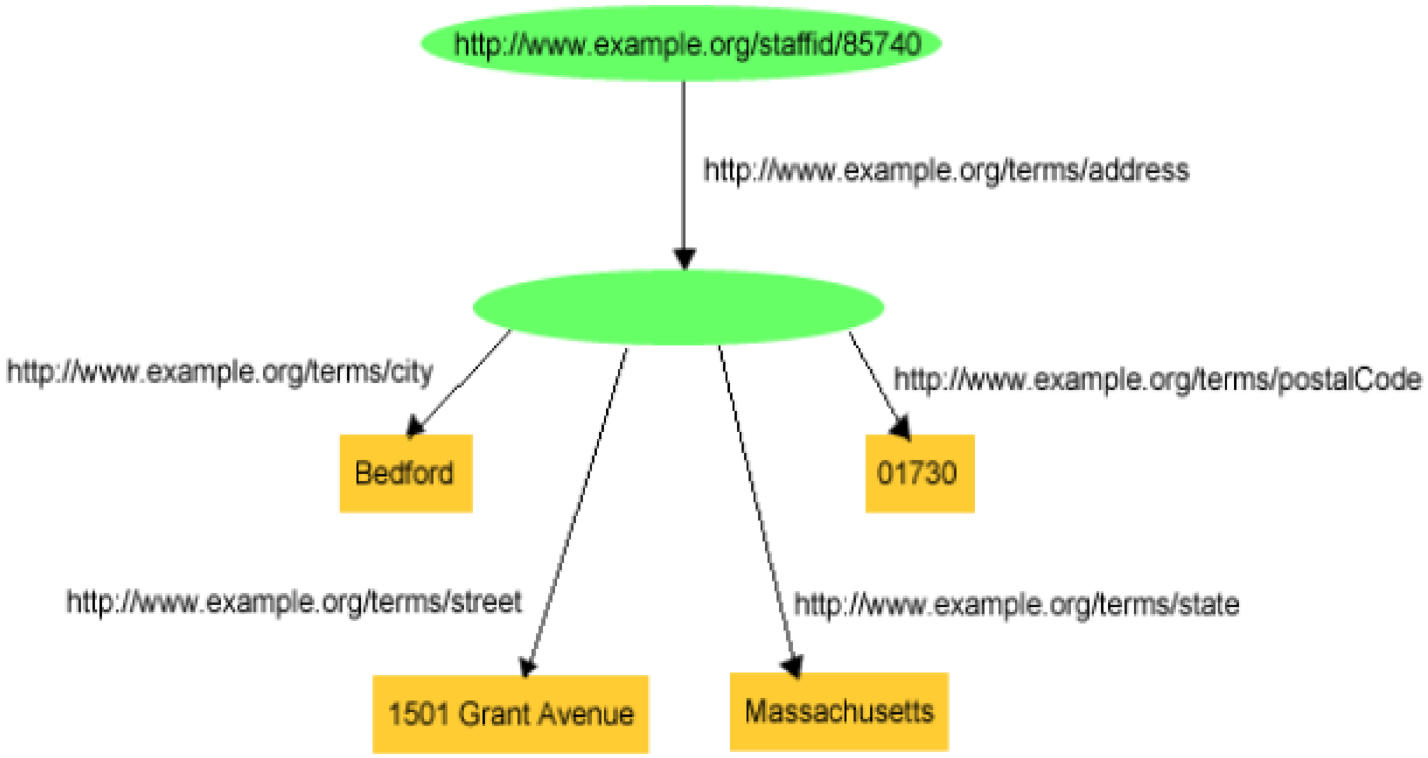
RDF graph: example

- Resource: http://www.w3.org/People/EM/contact#me


http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

About nodes

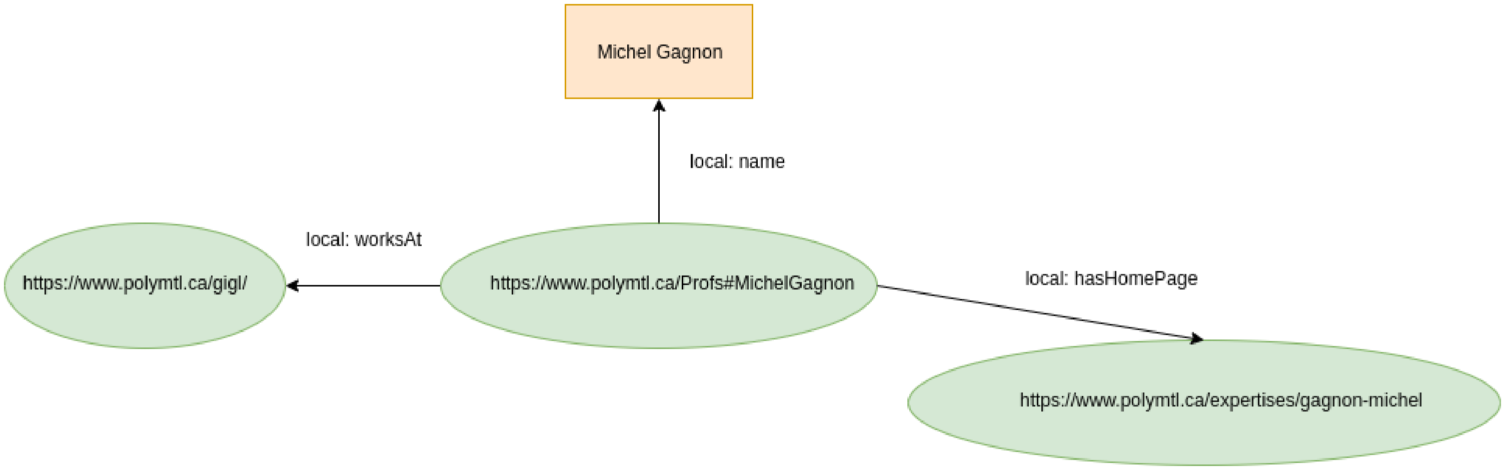
- Nodes may be a URI, left **blank** or be a typed literal (e.g. `"10"^^xsd:integer`)


http://www.example.org/staffid/85740
http://www.example.org/terms/address
http://www.example.org/terms/city
http://www.example.org/terms/postalCode
Bedford
01730
http://www.example.org/terms/street
http://www.example.org/terms/state
1501 Grant Avenue
Massachusetts

RDF *formats*

- RDF is a **modelling framework**
- It is **implemented** through various **data formats**
- **Common RDF formats** include:
    - XML RDF
    - Turtle RDF
    - JSON-LD RDF
    - N-Triple RDF
- These rely (among others) on the concepts of **namespaces**, **grouping**, **ids** and **literals**

XML RDF

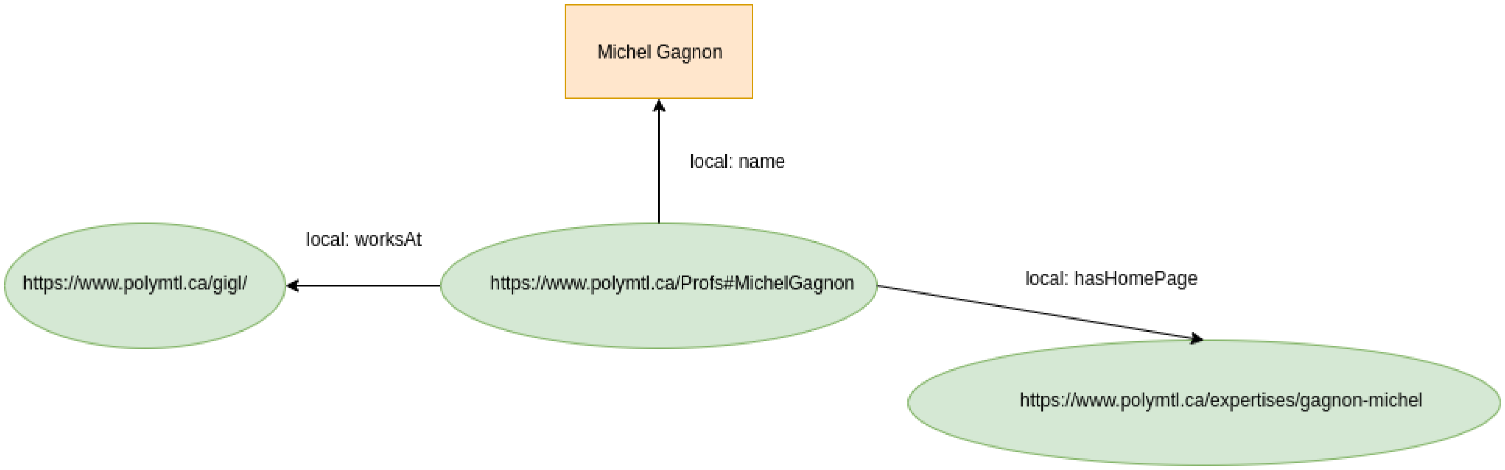

Michel Gagnon
local: name
local: worksAt
https://www.polymtl.ca/gigl/
https://www.polymtl.ca/Profs#MichelGagnon
local: hasHomePage
https://www.polymtl.ca/expertises/gagnon-michel

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:local="http://www.polymtl.ca/vocab#">
    <rdf:Description
           rdf:about="https://www.polymtl.ca/Profs#MichelGagnon">
        <local:hasHomePage
           resource="https://www.polymtl.ca/expertises/gagnon-miche
        <local:worksAt
           resource="https://www.polymtl.ca/gigl/"/>
        <local:name>Michel Gagnon</local:name>
  </rdf:Description>
</rdf:RDF>
```

XML RDF (continued)

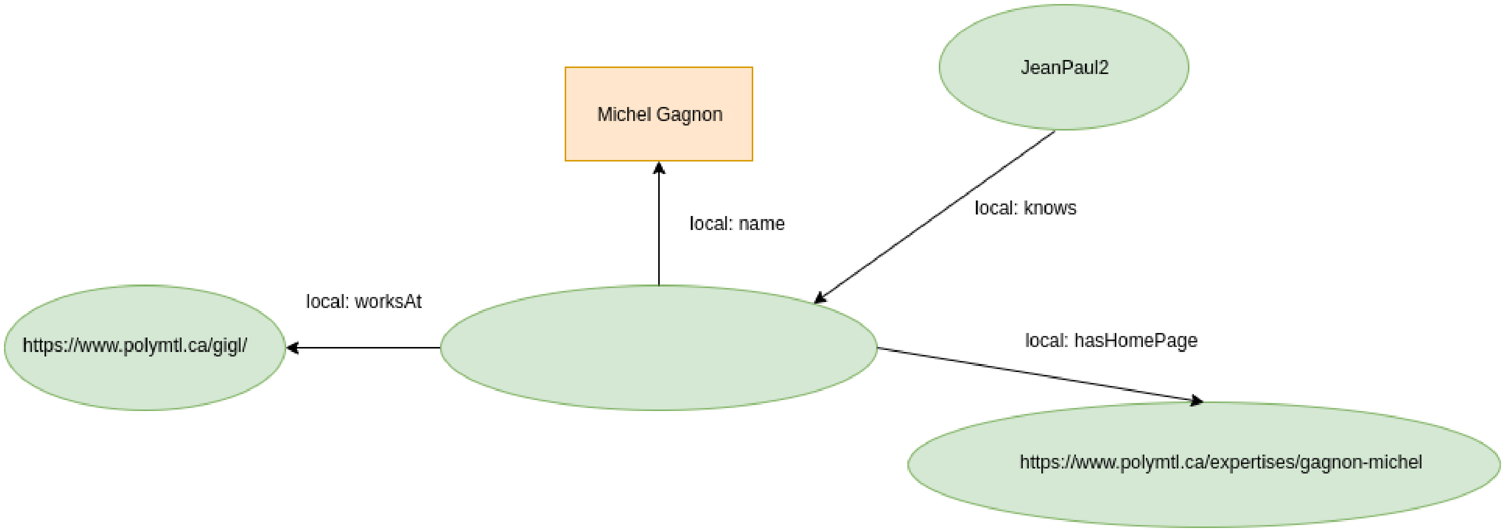
- Remarks:
  - **Composite resources** are embedded in a `rdf:Description` tag
  - **Blank nodes** use the `rdf:nodeID` tag and do not contain `about` attribute
  - **Literals** may be **typed**, if so an `rdf:datatype` attribute is used in the predicate linking a resource to that path
  - There may be **several ways of encoding** a given resource

Turtle RDF


Michel Gagnon
local: name
local: worksAt
https://www.polymtl.ca/gigl/
https://www.polymtl.ca/Profs#MichelGagnon
local: hasHomePage
https://www.polymtl.ca/expertises/gagnon-michel

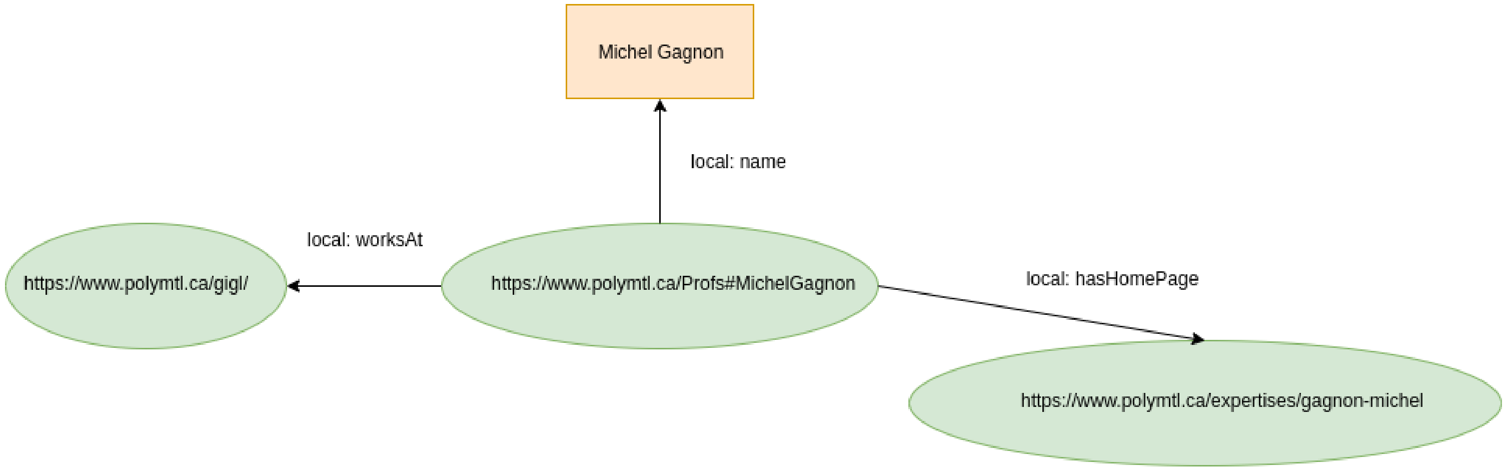
```
@prefix local: <http://www.polymtl.ca/vocab#>.
@prefix prof: <https://www.polymtl.ca/Profs#>.

prof:MichelGagnon
   local:hasHomePage <https://www.polymtl.ca/expertises/gagnon-mich
   local:worksAt <https://www.polymtl.ca/gigl/>;
   local:name "Michel Gagnon".
```

Turtle RDF with blank nodes



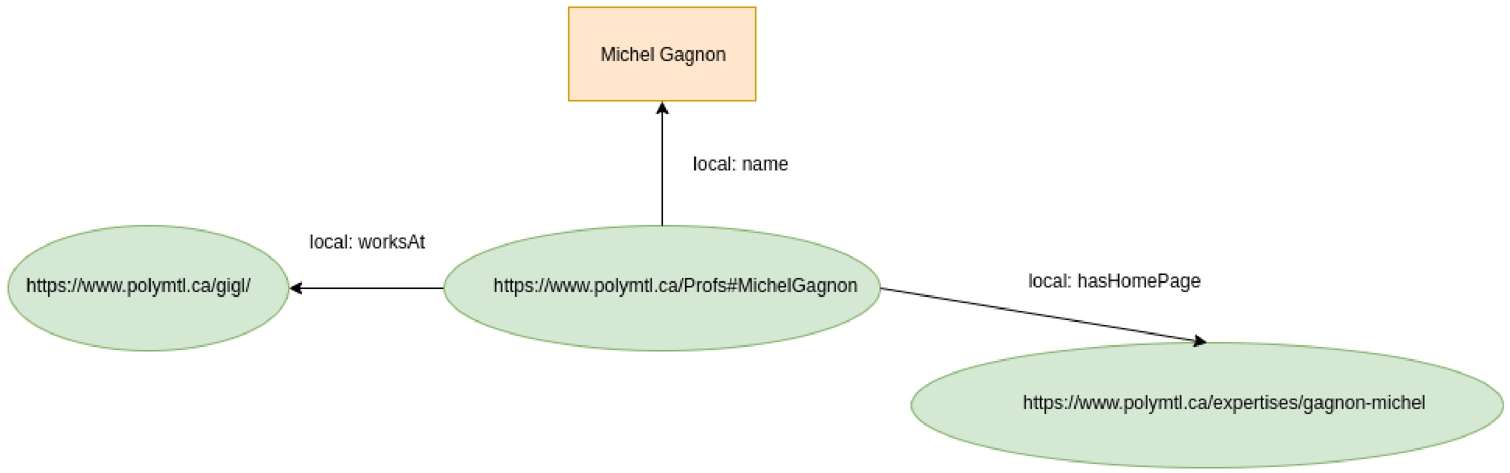
```
@prefix local: <http://www.polymtl.ca/vocab#>.

_:n1
   local:hasHomePage <https://www.polymtl.ca/expertises/gagnon-miche
   local:worksAt <https://www.polymtl.ca/gigl/>;
   local:name "Michel Gagnon".

local:JeanPaul2
   local:knows _:n1.
```

JSON-LD: an example


Michel Gagnon
local: name
local: worksAt
https://www.polymtl.ca/gigl/
https://www.polymtl.ca/Profs#MichelGagnon
local: hasHomePage
https://www.polymtl.ca/expertises/gagnon-michel

```
[ {
    "@id" : "https://www.polymtl.ca/Profs#MichelGagnon",
    "http://www.polymtl.ca/vocab#hasHomePage" :
        [{"@id" : "https://www.polymtl.ca/expertises/gagnon-michel"
    "http://www.polymtl.ca/vocab#worksAt":
        [{"@id":"https://www.polymtl.ca/gigl/"}],
    "http://www.polymtl.ca/vocab#name":
        [{"@value":"Michel Gagnon"}]
   },
   { "@id":"https://www.polymtl.ca/expertises/gagnon-michel" },
   { "@id":"https://www.polymtl.ca/gigl/" }
]
```

N-triples: an example


Michel Gagnon
local: name
local: worksAt
https://www.polymtl.ca/gigl/
https://www.polymtl.ca/Profs#MichelGagnon
local: hasHomePage
https://www.polymtl.ca/expertises/gagnon-michel

```
<https://www.polymtl.ca/Profs#MichelGagnon>
  <http://www.polymtl.ca/vocab#hasHomePage>
    <https://www.polymtl.ca/expertises/gagnon-michel> .

<https://www.polymtl.ca/Profs#MichelGagnon>
  <http://www.polymtl.ca/vocab#worksAt>
    <https://www.polymtl.ca/gigl/> .

<https://www.polymtl.ca/Profs#MichelGagnon>
  <http://www.polymtl.ca/vocab#name>
    "Michel Gagnon" .
```

The SPARQL Protocol and RDF Query Language

SPARQL

- **Query language** for RDF data

SPARQL

- **Query language** for RDF data
- Based on **pattern matching** on graphs

SPARQL

- **Query language** for RDF data
- Based on **pattern matching** on graphs
- Has a similar syntax to **SQL**

SPARQL

- **Query language** for RDF data
- Based on **pattern matching** on graphs
- Has a similar syntax to **SQL**
- Does not require local software when used via a webservice providing **endpoints** (URLs)

SPARQL

- **Query language** for RDF data
- Based on **pattern matching** on graphs
- Has a similar syntax to **SQL**
- Does not require local software when used via a webservice providing **endpoints** (URLs)
- Depending on the endpoints, queries can either be:
  - **entered in an online form**
    (e.g. with the Virtuoso SPARQL online editor)
  - or **encapsulated in HTTP GET/POST queries**
    (e.g., when the webservice uses a RESTful API)

SPARQL queries

- **Extract raw values** from RDF stores:

```
[PREFIX <key:value>]

SELECT <variables>

[FROM <URIs>]

[WHERE { <relation_constraints aka graph patterns> }]

[ORDER BY <variable>]
```

SPARQL queries (continued)

- Note that **variable** names start with `?` and **values** are:

  - **regular URIs**
  - **prefixed URIs**, e.g. `local:worksAt`, or
  - **plain** literals, e.g. `"27"`, `"Hello World"@en`, ..., or
  - **typed** literals, e.g. `"27"^^http://www.w3.org/2001/XMLSchema#integer`

- Graph patterns are **triples** such as `value1` `value2` `value3`
  where `value1`, `value2` and `value3` refer to subject, predicate and object respectively, and may be **merged**:

```
value1 value2 value3 .
value1 value4 value5 .
```

```
value1 value2 value3 ;
       value4 value5 .
```

Other SPARQL queries

- **Check if there is at least one result** for a given query pattern:

```
ASK ... WHERE ...
```

- **Return an RDF graph that describes a resource**:

```
DESCRIBE ... WHERE ...
```

- **Return an RDF graph that is created from a template specified as part of the query itself**:

```
CONSTRUCT ... WHERE ...
```

Application

Querying the DBpedia store

DBpedia

- Community effort to **extract structured information from Wikipedia**
- Started in 2006, participated in the **Linked Open Data** initiative
- Allows for **semantic queries** on Wikipedia content **and linking with other datasets**
- DBpedia database **is thus aligned** with Wikipedia content, see e.g.

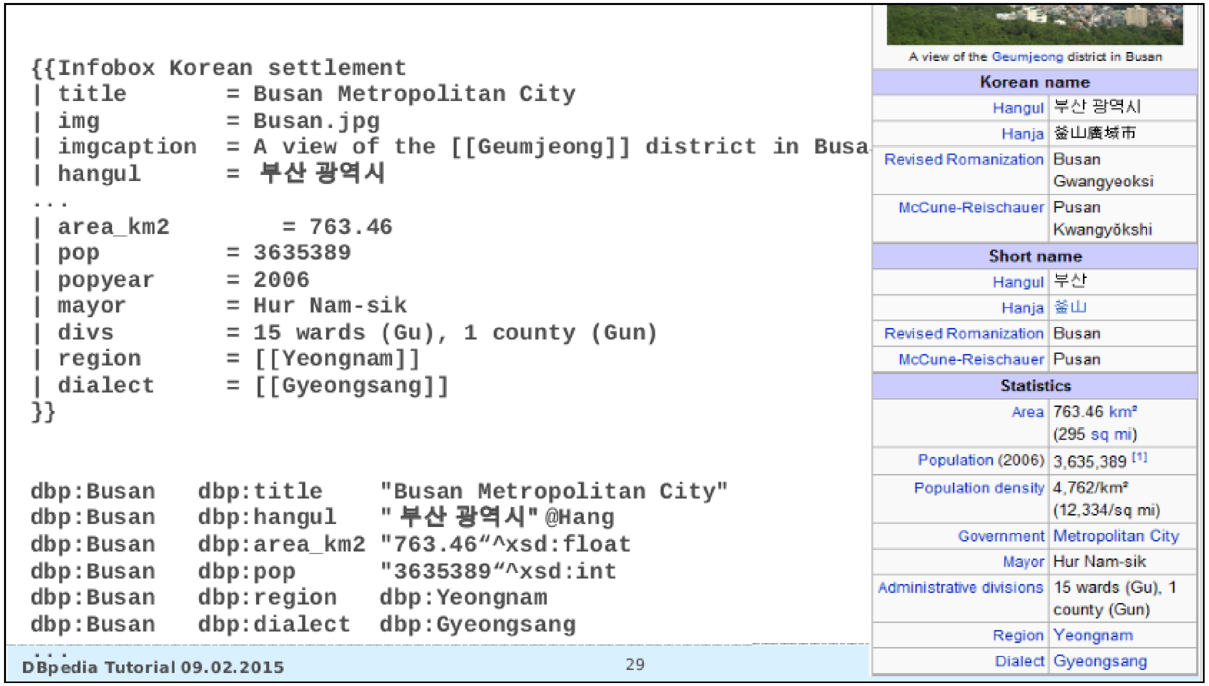
https://en.wikipedia.org/wiki/Monty_Python

⬍

https://dbpedia.org/page/Monty_Python

DBpedia (continued)

- From wikipedia to DBpedia



```
{{Infobox Korean settlement
| title        = Busan Metropolitan City
| img          = Busan.jpg
| imgcaption   = A view of the [[Geumjeong]] district in Busa
| hangul       = 부산 광역시
...
| area_km2         = 763.46
| pop          = 3635389
| popyear      = 2006
| mayor        = Hur Nam-sik
| divs         = 15 wards (Gu), 1 county (Gun)
| region       = [[Yeongnam]]
| dialect      = [[Gyeongsang]]
}}


dbp:Busan   dbp:title    "Busan Metropolitan City"
dbp:Busan   dbp:hangul   "부산 광역시"@Hang
dbp:Busan   dbp:area_km2 "763.46"^xsd:float
dbp:Busan   dbp:pop      "3635389"^xsd:int
dbp:Busan   dbp:region   dbp:Yeongnam
dbp:Busan   dbp:dialect  dbp:Gyeongsang
...
```

DBpedia Tutorial 09.02.2015 29

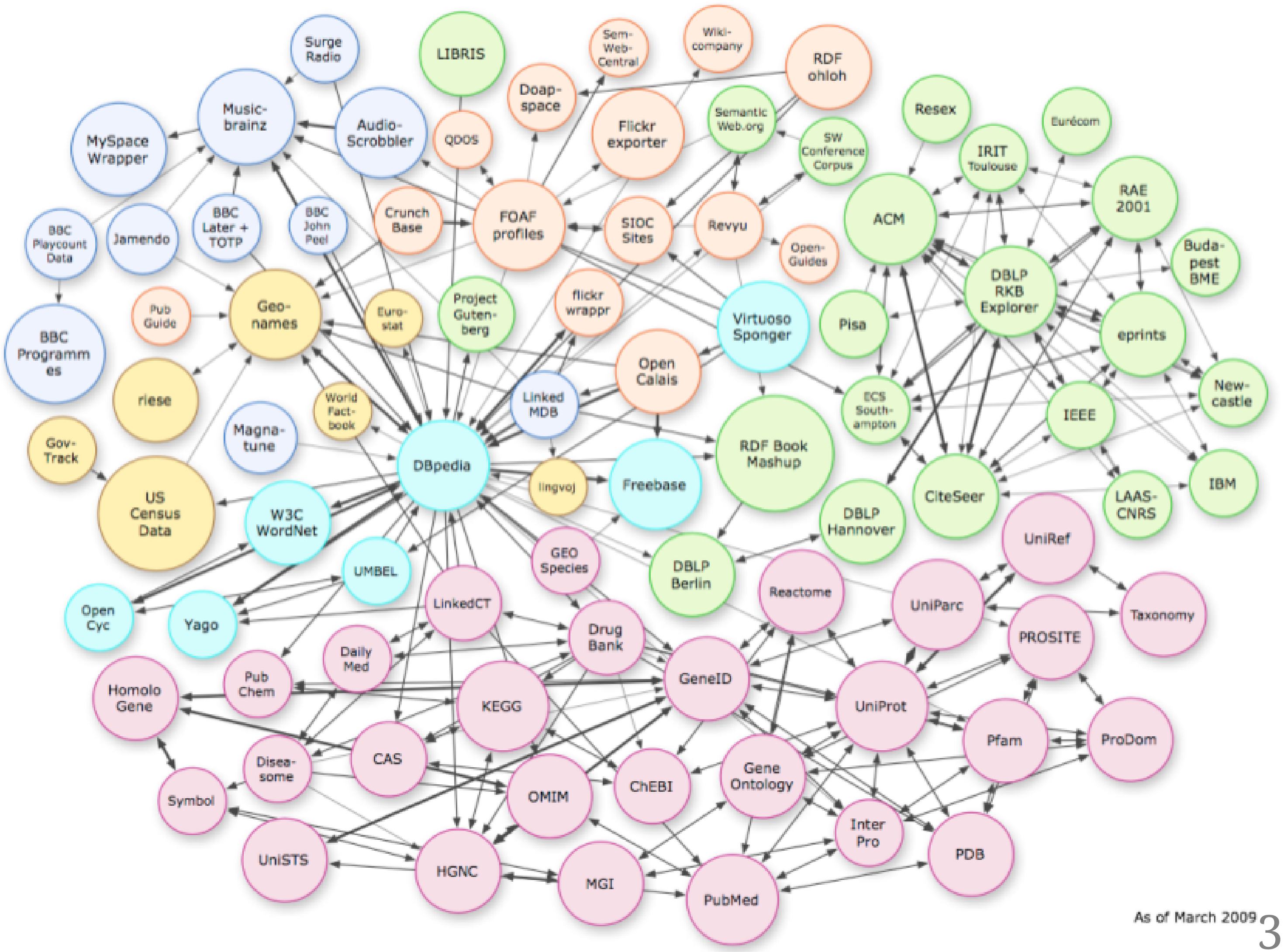
A view of the Geumjeong district in Busan

Korean name	
Hangul	부산 광역시
Hanja	釜山廣域市
Revised Romanization	Busan Gwangyeoksi
McCune-Reischauer	Pusan Kwangyŏkshi
Short name	
Hangul	부산
Hanja	釜山
Revised Romanization	Busan
McCune-Reischauer	Pusan
Statistics	
Area	763.46 km² (295 sq mi)
Population (2006)	3,635,389 [1]
Population density	4,762/km² (12,334/sq mi)
Government	Metropolitan City
Mayor	Hur Nam-sik
Administrative divisions	15 wards (Gu), 1 county (Gun)
Region	Yeongnam
Dialect	Gyeongsang

DBpedia (continued)

- The 2016-04 release of the DBpedia data set describes **6.0 million entities**, out of which 5.2 million are classified in a consistent ontology, including:
    - 1.5M persons,
    - 810k places,
    - 135k music albums,
    - 106k films,
    - 20k video games,
    - 275k organizations,
    - 301k species ,
    - 5k diseases.
- 1.5 billion RDF triples extracted from the English Wikipedia

DBpedia (continued)



Querying DBpedia: example #1

- DBpedia endpoint: http://dbpedia.org/sparql (online form)
- Retrieving all cities in Texas:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT *
WHERE {
  ?city
    rdf:type
      <http://dbpedia.org/class/yago/WikicatCitiesInTexas>
}
```

Querying DBpedia: example #2

- Retrieving all cities in Texas, together with their population:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT *
WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> .
?city
  dbp:populationTotal
    ?popTotal .
}
```

Querying DBpedia: example #3

- Retrieving all cities in Texas with their total and metropolitan populations:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal ;
  dbp:populationMetro ?popMetro .
}
```

OPTIONAL clause

- To deal with missing values
- Example:
  - Retrieve all cities that are in Texas and their total population and optionally the metropolitan population, if it exists.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal .

OPTIONAL {?city dbp:populationMetro ?popMetro . }
}
```

Ordering results

- Keywords: `asc` and `desc`
- Example:
  - Retrieve all cities that are in Texas and their total population and optionally the metro population, if it exists, by decreasing number of inhabitants.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal .

OPTIONAL {?city dbp:populationMetro ?popMetro . }
}
ORDER BY desc(?popTotal)
```

Limit and offset

- **`LIMIT`** puts an upper bound on the number of results
- **`OFFSET`** causes the results to start after the specified number
- Example:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal .
OPTIONAL {?city dbp:populationMetro ?popMetro. }
}
ORDER BY desc(?popTotal)
LIMIT 10
OFFSET 5
```

Filtering data

- Available criteria:
  - Logical filters: `&&`, `||`, `!`
  - Mathematical filters: `+`, `-`, `*`, `/`
  - Comparisons: `=`, `!=`, `<`, `>`, `<=`, `>=`
  - SPARQL tests: `isURI`, `isBlank`, `isLiteral`, `bound`
  - SPARQL accessors: `str`, `lang`, `datatype`
  - Other filters: `sameTerm`, `langmatches`, `regex`

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>
SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal .
OPTIONAL {?city dbp:populationMetro ?popMetro . }
FILTER (?popTotal > 50000)
} ORDER BY desc(?popTotal)
```

Filtering data (continued)

- List cities with their URI and **name**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal ;
  rdfs:label ?name
OPTIONAL {?city dbp:populationMetro ?popMetro. }
FILTER (?popTotal > 50000 && langmatches(lang(?name), "EN"))
}
ORDER BY desc(?popTotal)
```

NB: `rdfs:label` is a RDFS predicate commonly used to represent the human-readable name of a resource.

Filtering data (continued)

- Retrieving cities having "El" in their English names.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal ;
  rdfs:label ?name
OPTIONAL {?city dbp:populationMetro ?popMetro. }
FILTER (?popTotal > 50000 &&
      langmatches(lang(?name), "EN") &&
      regex(str(?name), "El"))
}
ORDER BY desc(?popTotal)
```

Negation

- **bound** is a boolean test that returns whether or not a specific property is bound in the result being returned.
- Example:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal ;
  rdfs:label ?name
OPTIONAL {?city dbp:populationMetro ?popMetro. }
FILTER (?popTotal > 50000 && langmatches(lang(?name), "EN") )
FILTER(!bound(?popMetro))
}
ORDER BY desc(?popTotal)
```

Union

- A disjunction between two basic graph patterns.
- Retrieve cities that are in Texas or in California.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/ontology/>

SELECT * WHERE {
?city dbp:populationTotal ?popTotal ;
      rdfs:label ?name
{ ?city
    rdf:type
      <http://dbpedia.org/class/yago/WikicatCitiesInTexas> . }
UNION
{ ?city
    rdf:type
      <http://dbpedia.org/class/yago/CitiesInCalifornia>. }

OPTIONAL {?city dbp:populationMetro ?popMetro. }
FILTER (?popTotal > 50000 && langmatches(lang(?name), "EN"))
}
ORDER BY desc(?popTotal)
```

ASK query: example #1

- Is Austin a city in Texas?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ASK WHERE {
 <http://dbpedia.org/resource/Austin,_Texas>
 rdf:type
 <http://dbpedia.org/class/yago/WikicatCitiesInTexas> .
}
```

ASK query: example #2

- Is there a city in Texas that has a total population greater than 600,000 and a metro population less than 1,800,000?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>

ASK WHERE {
?city
  rdf:type
     <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal ;
  dbp:populationMetro ?popMetro.
FILTER (?popTotal > 600000 && ?popMetro < 1800000)
}
```

Describing RDF graphs

- RDF graph that describes Austin?

```
DESCRIBE <http://dbpedia.org/resource/Austin,_Texas>
```

NB: returns the triples where the resource is in the subject or in the object position.

- Other example:

```
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/ontology/>
DESCRIBE ?city WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal ;
  dbp:populationMetro ?popMetro.
FILTER (?popTotal > 600000 && ?popMetro < 1800000)
}
```

Constructing RDF graphs

- Construct a new RDF graph for cities in Texas that have a metro population greater than 500,000.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/ontology/>

CONSTRUCT {
?city
  rdf:type
    <http://myvocabulary.com/LargeMetroCitiesInTexas> ;
  <http://myvocabulary.com/cityName> ?name ;
  <http://myvocabulary.com/totalPopulation> ?popTotal ;
  <http://myvocabulary.com/metroPopulation> ?popMetro .
} WHERE {
?city
  rdf:type
    <http://dbpedia.org/class/yago/WikicatCitiesInTexas> ;
  dbp:populationTotal ?popTotal ;
  rdfs:label ?name ;
  dbp:populationMetro ?popMetro .
FILTER (?popTotal > 500000 && langmatches(lang(?name), "EN"))
}
```

Python wrapping

Introducing the SPARQLwrapper and RDFlib libraries

SPARQLwrapper

```
from SPARQLWrapper import SPARQLWrapper, JSON

sparql = SPARQLWrapper("http://dbpedia.org/sparql/")
sparql.setQuery("""
  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  SELECT ?label
  WHERE {
  <http://dbpedia.org/resource/Asturias> rdfs:label ?label }
""")
sparql.setReturnFormat(JSON)
results = sparql.query().convert()

for result in results["results"]["bindings"]:
    print(result["label"]["value"])
```

RDFlib

```python
import rdflib

g = rdflib.Graph()
result = g.parse("http://www.w3.org/People/Berners-Lee/card")
# NB: it retrieves and parses an RDF file!

print("graph has %s statements." % len(g))
# prints graph has 79 statements.

for subj, pred, obj in g:
    print(subj,pred,obj)
```

Thank you!

Slideshow created using remark.