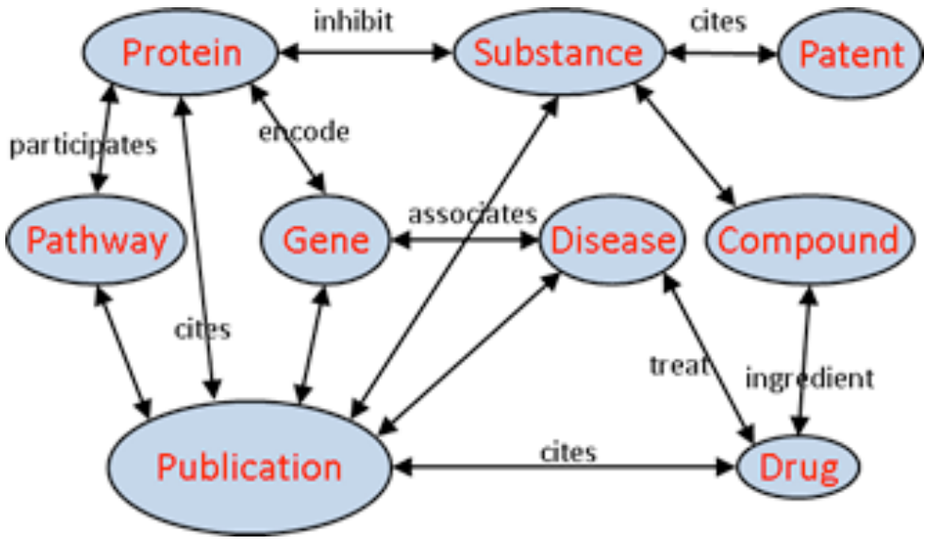
SPARQL tutorial

BIME 550: Knowledge Representation
January 24, 2018

Lucy Lu Wang
lucylw@uw.edu

SPARQL is a query language for RDF

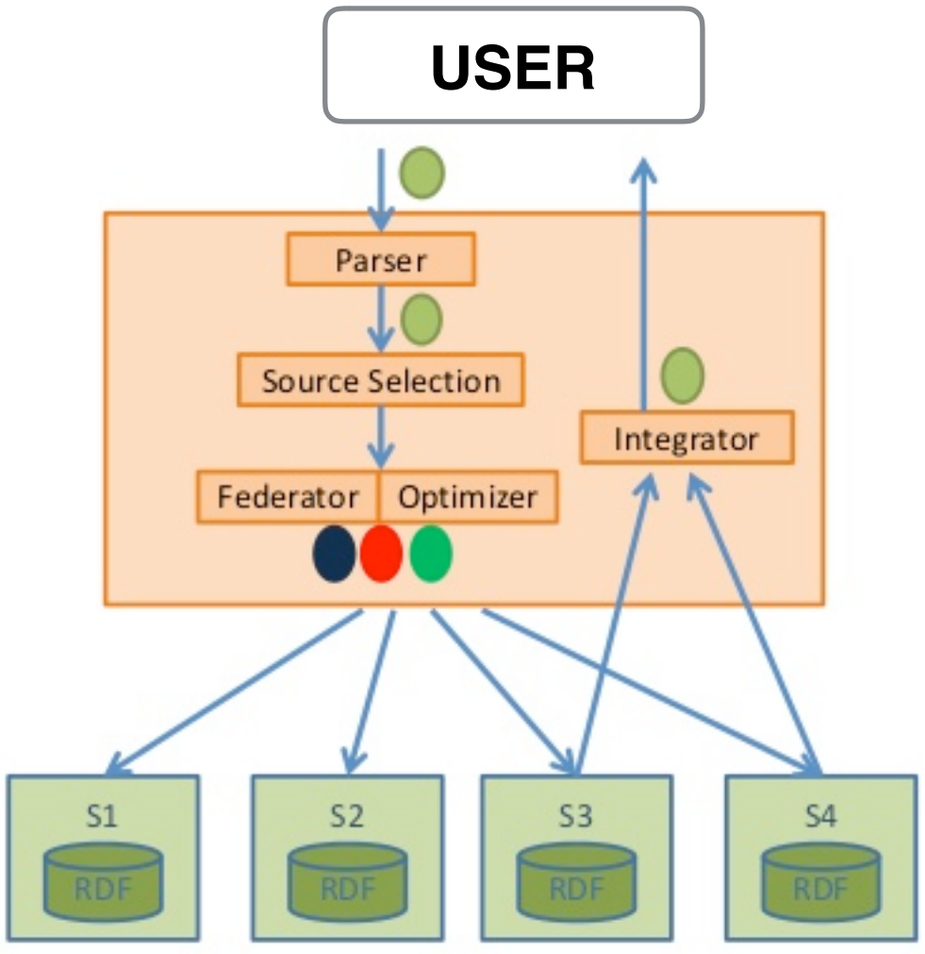


Gene *encodes* Protein
Protein *participates_in* Pathway
Pathway *cites* Publication
Gene *cites* Publication
Gene *associated_with* Disease
Drug *treats* Disease
Drug *has_ingredient* Compound

Entities and properties are identified by a *Unique Reference Identifier* (URI)

SPARQL pattern matches over these triples

RDF stores can be accessed through a SPARQL endpoint



https://www.slideshare.net/muhammad_saleem/hi-bis-cuseswc2014

Some notable endpoints:

https://www.ebi.ac.uk/rdf/services/sparql

http://yasgui.org/

http://dbpedia.org/sparql

PREFIX allows you to abbreviate URIs

Entity URIs:

<http://www.biopax.org/release/biopax-level3.owl#Pathway>
<http://www.biopax.org/release/biopax-level3.owl#Protein>

After abbreviation:

PREFIX bp: <http://www.biopax.org/release/biopax-level3.owl#>

bp:Pathway
bp:Protein

There are four types of SPARQL queries

SELECT	Retrieve matches
ASK	Is there a match?
DESCRIBE	Describe match
CONSTRUCT	Create an RDF graph from matches

Note: not all endpoints support all query types of keywords

SELECT retrieves query matches

SELECT and WHERE are analogous to keywords in SQL

Get all entities of type pathway

```
SELECT ?pathway WHERE {
  ?pathway rdf:type bp:Pathway .
}
```

Get 50 distinct entities of type pathway

```
SELECT DISTINCT ?pathway WHERE {
  ?pathway a bp:Pathway .
} LIMIT 50
```

To match literals, specify either language suffix or datatype URI

“Heart”@en

“100”^^xsd:integer

“Glycolysis”^^xsd:string

Note: “Glycolysis” is not equal to “Glycolysis”^^xsd:string

Get entities of type pathway with name “Glycolysis”

```
SELECT ?pathway WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName “Glycolysis”^^xsd:string
}
```

ASK & **DESCRIBE** can be used to get more information

Does something of type pathway exist?

```
ASK {
  ?pathway a bp:Pathway .
}
```

Tell me everything about the pathway named “Glycolysis”

```
DESCRIBE ?pathway WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName “Glycolysis”^^xsd:string
}
```

CONSTRUCT creates a graph as output

Create a graph of all the components of pathways named “Glycolysis”

```
CONSTRUCT {
  ?pathway bp:pathwayComponent ?component .
}
WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName "Glycolysis"^^xsd:string .
  ?pathway bp:pathwayComponent ?component .
}
```

FILTER can be used to restrict the outputs

Fetch all pathways that have the word “signaling” in their name; “i” flag denotes case-insensitivity

```
SELECT DISTINCT ?pathway ?name
WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName ?name .
  FILTER regex(?name, "signaling", "i")
}
```

OPTIONAL provides additional information when it exists

Return a sub-pathway and sub-pathway name if they exist in the knowledgebase

```
SELECT DISTINCT ?pathway ?name ?subpath ?subpathname
WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName ?name .
  OPTIONAL {
    ?pathway bp:pathwayComponent ?subpath .
    ?subpath a bp:Pathway .
    ?subpath bp:displayName ?subpathname .
  }
```

UNION joins the results of multiple matches

Union of three BioPAX terms for properties for name

* Bind to the same variable -> ?name

```
SELECT DISTINCT ?pathway ?name
WHERE {
  ?pathway a bp:Pathway .
  { ?pathway bp:name ?name . }
  UNION
  { ?pathway bp:displayName ?name . }
  UNION
  { ?pathway bp:standardName ?name . }
  FILTER regex(?name, "notch1", "i")
}
```

ORDER BY sorts the output

Order pathway output alphabetically by pathway name

```
SELECT DISTINCT ?pathway ?name
WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName ?name .
} ORDER BY ?name
```

ASC vs DESC: e.g. ORDER BY DESC(?name)

GROUP BY and **COUNT** can be used to aggregate over properties

Aggregate over pathway names and count how many pathways with each name; useful in combination with ORDER BY

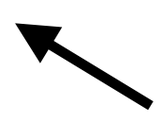
```
SELECT ?name (COUNT(?name) as ?pathnum)
WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName ?name .
}
GROUP BY ?name
```

Construct federated queries using **SERVICE**

For a protein in Reactome, query for the label and sequence of the corresponding UniProt entity

```
PREFIX uniprot: <http://purl.uniprot.org/core/>

SELECT DISTINCT ?protein ?entref ?label ?sequence
WHERE {
  ?protein a bp:Protein .
  ?protein bp:displayName "BRCA1"^^xsd:string .
  ?protein bp:entityReference ?entref .
  SERVICE <http://sparql.uniprot.org/sparql> {
    ?entref rdfs:label ?label .
    ?entref uniprot:sequence ?sequence .
    }}
LIMIT 5
```



This part is being executed at the UniProt SPARQL endpoint!

Arbitrary path length matches using *

Get all components of components of “Glycolysis” that are of type BiochemicalReaction

```
SELECT DISTINCT ?reaction ?rxname
WHERE {
  ?pathway a bp:Pathway .
  ?pathway bp:displayName "Glycolysis"^^xsd:string .
  ?pathway bp:pathwayComponent* ?reaction .
  ?reaction a bp:BiochemicalReaction .
  ?reaction bp:displayName ?rxname .
}
```

Wildcards are much less performant, but can match arbitrary properties

When you are uncertain about all the possible paths leading to your entities of interest, you can use a <wildcard> construct

?reaction **<wildcard>*** ?member .
?member ?property ?value .
?value **<wildcard>*** ?protein .
?protein a bp:Protein .

Should only use this when uncertain of properties; otherwise, use UNION construct