Lecture 4

Knowledge Base Queries & SPARQL

COMP 474/6741, Winter 2022

René Witte
Department of Computer Science
and Software Engineering
Concordia University

Outline

René Witte

Slides Credit

- Includes slides by Ivan Herman, W3C [Her]

Outline

René Witte

René Witte

Vocabularies

- Data integration needs agreements on
  - terms
    - “translator”, “author”
  - categories used
    - “Person”, “literature”
  - relationships among those
    - “an author is also a Person…”, “historical fiction is a narrower term than fiction”
    - ie, new relationships can be deduced

René Witte
Concordia University

Vocabularies

- There is a need for “languages” to define such vocabularies
  - to define those vocabularies
  - to assign clear “semantics” on how new relationships can be deduced

René Witte
Concordia University

Classes, resources, …

- Think of well known traditional vocabularies:
  - use the term “novel”
  - “every novel is a fiction”
  - “«The Glass Palace» is a novel”
  - etc.
- RDFS defines resources and classes:
  - everything in RDF is a “resource”
  - “classes” are also resources, but…
  - …they are also a collection of possible resources (i.e., “individuals”)
    - “fiction”, “novel”, …

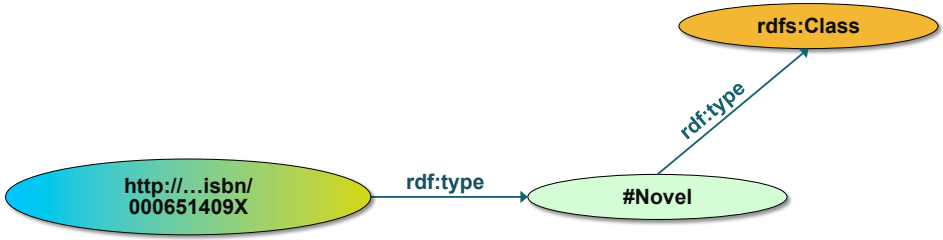
René Witte
Concordia University

Classes, resources, … (cont.)

- Relationships are defined among resources:
  - “typing”: an individual belongs to a specific class
    - “«The Glass Palace» is a novel”
    - to be more precise: “«http://.../000651409X» is a novel”
  - “subclassing”: all instances of one are also the instances of the other (“every novel is a fiction”)
- RDFS formalizes these notions in RDF

René Witte
Concordia University

Classes, resources in RDF(S)



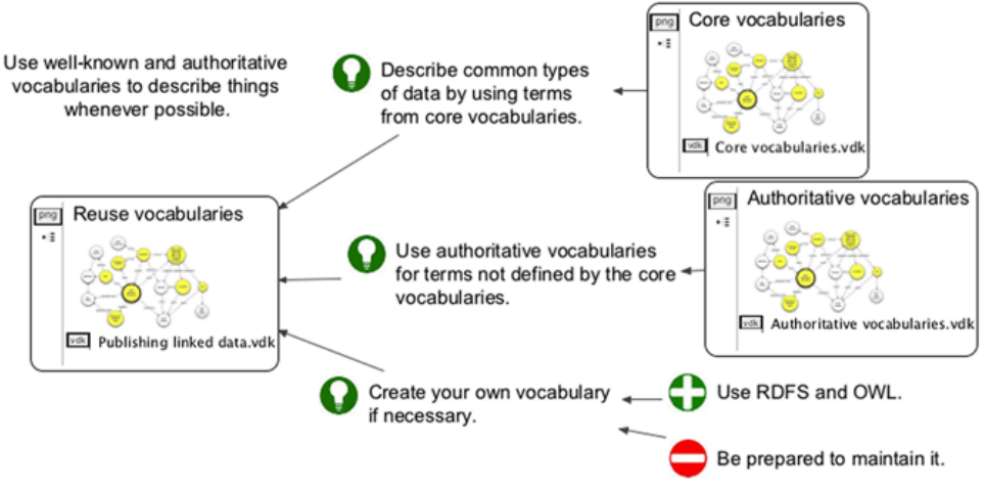
- RDFS defines the meaning of these terms
  - (these are all special URI-s, we just use the namespace abbreviation)

→ Worksheet #3: Tasks 1 & 2

René Witte
Concordia University

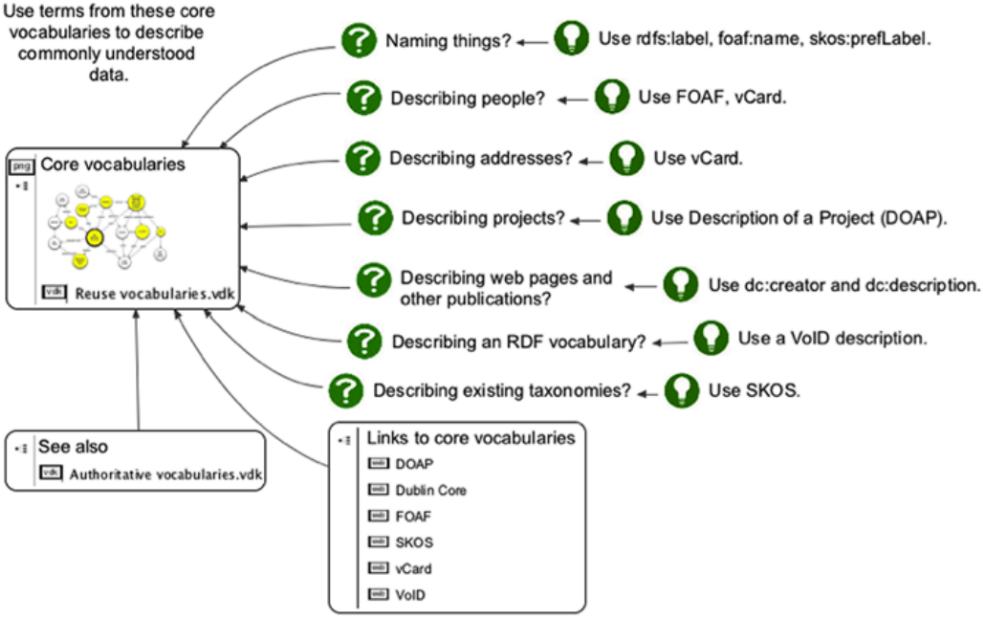
Reuse vocabularies whenever possible

René Witte

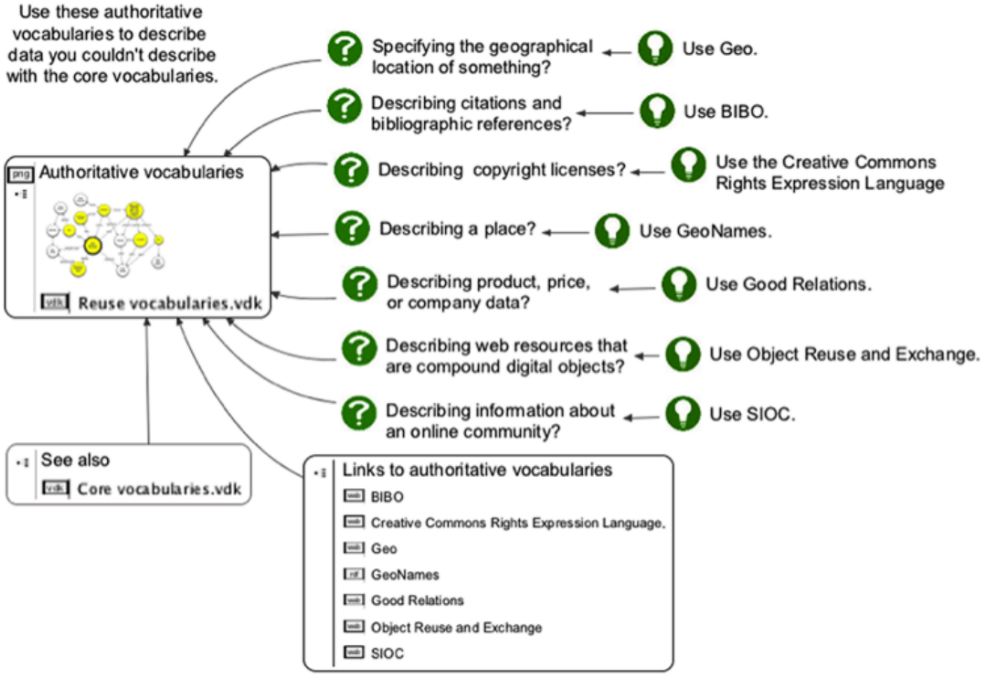


Core Vocabularies

René Witte



More authoritative vocabularies



René Witte

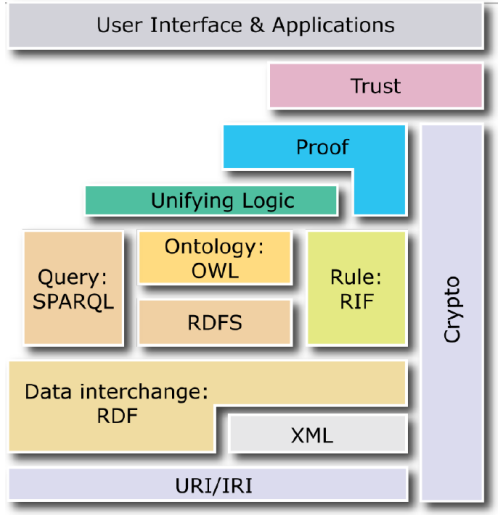
Outline

René Witte

René Witte
Concordia University

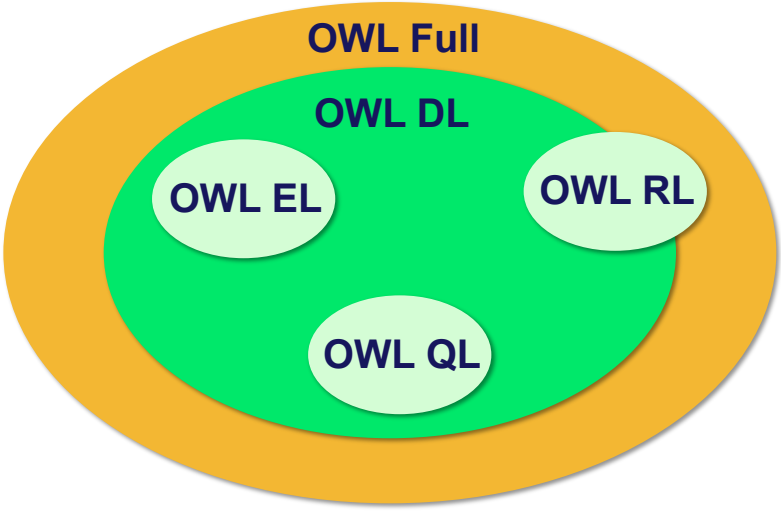
The Web Ontology Language (OWL)

- Current version OWL2 (2009)
- Different OWL2-Profiles (lite, full, etc.)
- Ontology language based on Description Logics (DL)
- Enables logic-based reasoning



http://www.w3.org/TR/owl2-overview/

OWL Species



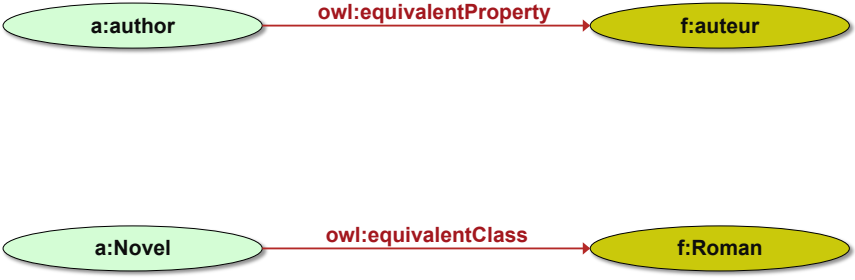
René Witte

Term equivalences

- For individuals:
  - owl:sameAs: two URIs refer to the same concept (“individual”)
  - owl:differentFrom: negation of owl:sameAs

René Witte
Concordia University

Other example: connecting to French



René Witte

Typical usage of owl:sameAs

- Linking our example of Amsterdam from one data set (DBpedia) to the other (Geonames):

```
<http://dbpedia.org/resource/Amsterdam>
  owl:sameAs <http://sws.geonames.org/2759793>;
```

- This is the main mechanism of “Linking” in the Linked Open Data project

→ **Worksheet #3: Task 3**

René Witte
Concordia University

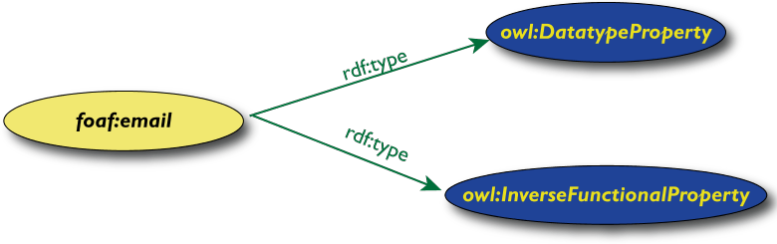
Property characterization

- In OWL, one can characterize the behavior of properties (symmetric, transitive, functional, inverse functional, reflexive, irreflexive, …)
- OWL also separates data and object properties
  - “datatype property” means that its range are typed literals

René Witte
Concordia University

Characterization example

- “foaf:email” may be defined as “inverse functional”
  - i.e., two different subjects cannot have identical objects



René Witte
Concordia University

What this means is…

- If the following holds in our triples:

```
:email rdf:type owl:InverseFunctionalProperty.
```

René Witte
Concordia University

What this means is…

- If the following holds in our triples:

```
:email rdf:type owl:InverseFunctionalProperty.
<A> :email "mailto:a@b.c".
<B> :email "mailto:a@b.c".
```

René Witte
Concordia University

What this means is…

- If the following holds in our triples:

```
:email rdf:type owl:InverseFunctionalProperty.
<A> :email "mailto:a@b.c".
<B> :email "mailto:a@b.c".
```

then, processed through OWL, the following holds, too:

```
<A> owl:sameAs <B>.
```

René Witte
Concordia University

Inverse properties

- There may be an inverse relationship among properties, eg:

```
<somebook> ex:author <somebody>.
ex:author owl:inverseOf ex:authorOf.
```

yields, in OWL:

```
<somebody> ex:authorOf <somebook>.
```

René Witte
Concordia University

Classes in OWL

- In RDFS, you can subclass existing classes… that's all
- In OWL, you can construct classes from existing ones:
  - enumerate its content
  - through intersection, union, complement
  - etc

René Witte
Concordia University

OWL DL and Description Logic

- OWL DL can be interpreted as a variant of Description Logic
  - for connoisseurs: OWL (2) DL $\approx \mathcal{SROIQ}$(D)
- Hence the results of this particular area of logic are directly applicable

René Witte
Concordia University

Description Logic Formalism

- There is also a compact mathematical notation for axioms, assertions, etc:
  - Literature $\equiv$ Novel $\sqcup$ Short_Story $\sqcup$ Poetry
  - Listed_Price $\sqsubseteq \forall$ currency.Currencies
- You may see these in papers, books…

René Witte

Ontologies examples

- eClassOwl: eBusiness ontology for products and services, 75,000 classes and 5,500 properties
- National Cancer Institute's ontology: about 58,000 classes
- Open Biomedical Ontologies Foundry: a collection of ontologies, including the Gene Ontology to describe gene and gene product attributes in any organism or protein sequence and annotation terminology and data (UniProt)
- BioPAX: for biological pathway data

René Witte
Concordia University

Outline

René Witte

Querying RDF graphs

- Remember the Python+RDFLib idiom:

```
for (s,p,o) in graph.triples((subject,None,None)) :
  do_something(p,o);
```

René Witte
Concordia University

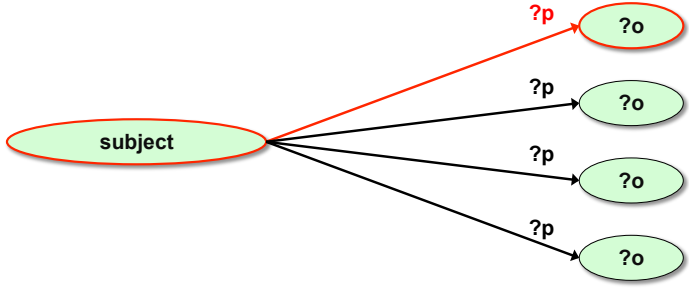
Querying RDF graphs

- In practice, more complex queries into the RDF data are necessary
  - something like: “give me the (a,b) pair of resources, for which there is an x such that (x parent a) and (b brother x) holds” (ie, return the uncles)
    - these rules may become quite complex
- The goal of SPARQL (Query Language for RDF)

René Witte
Concordia University

Analyze the Python+RDFLib example

```
for (s,p,o) in graph.triples((subject,None,None)) :
  do_something(p,o);
```



René Witte

General: graph patterns

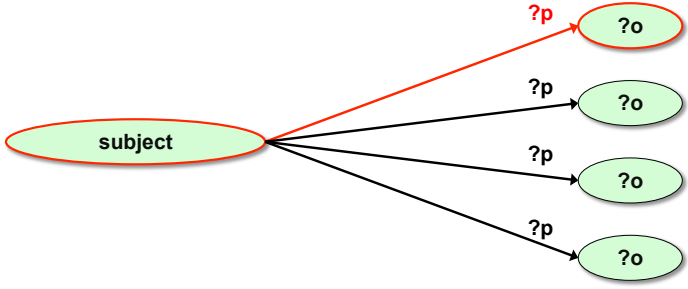
- The fundamental idea: use graph patterns
  - the pattern contains unbound symbols
  - by binding the symbols, subgraphs of the RDF graph are selected
  - if there is such a selection, the query returns the bound resources

René Witte
Concordia University

Our Python example in SPARQL

```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

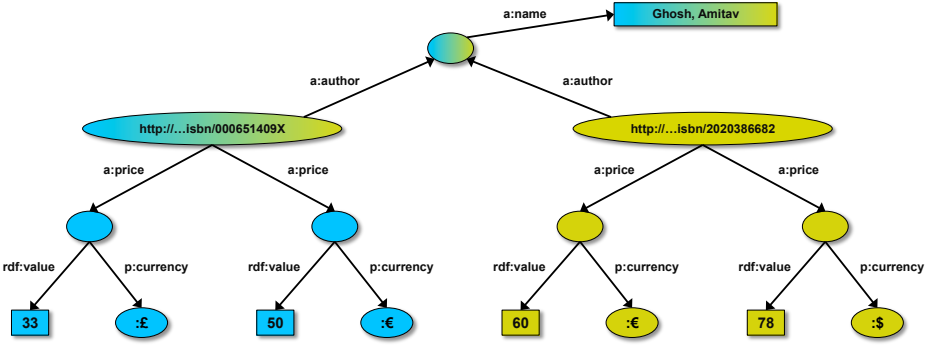
- The triples in WHERE define the graph pattern, with ?p and ?o “unbound” symbols
- The query returns all p,o pairs



René Witte

Simple SPARQL example

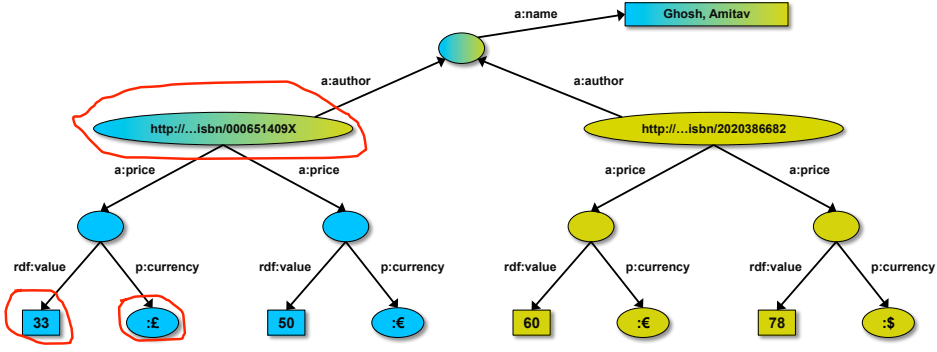
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

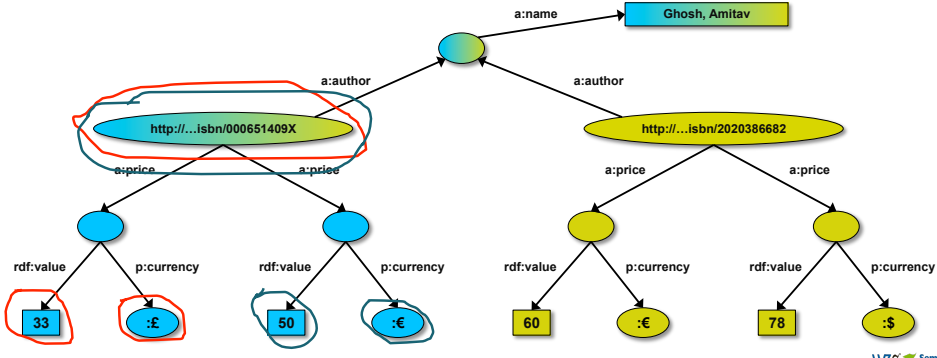
Returns: [<…409X>,33,:£]



Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

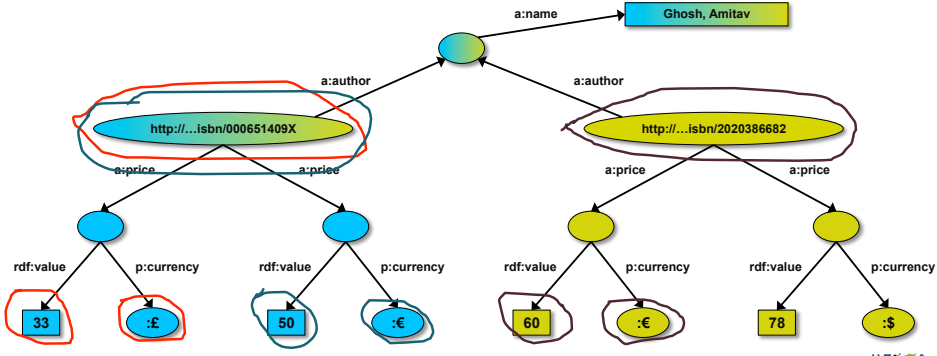
Returns: [<…409X>,33,:£], [<…409X>,50,:€]



Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

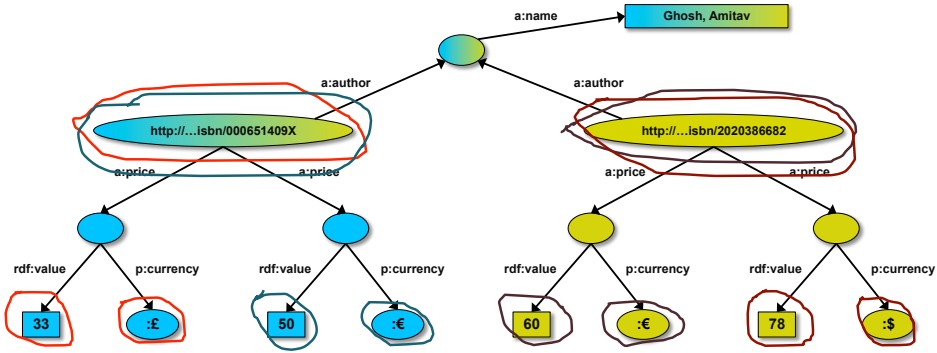
Returns: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€]



Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

Returns: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€], [<…6682>,78,:$]



→ **Worksheet #3: Task 4**

Outline

René Witte

Query RDF: (SPARQL)

Photo credit "reedster", Flickr

René Witte
Concordia University

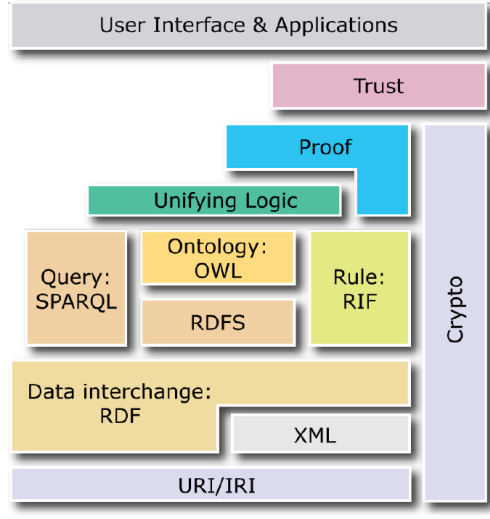
René Witte
Concordia University

SPARQL W3C Standard

SPARQL stands for...

"SPARQL Protocol And RDF Query Language"

- Current version 1.1 (like RDF, RDFS, etc.)
- Language for querying graphs
- and a protocol for doing this over the web using a SPARQL endpoint
- Major difference between 1.0 and 1.1: modifying graphs via SPARQL (insert, delete etc.)



https://www.w3.org/TR/sparql11-query/

Describe

René Witte

Describing a resource

Simple query that can be used when no information about a graph's content is available.

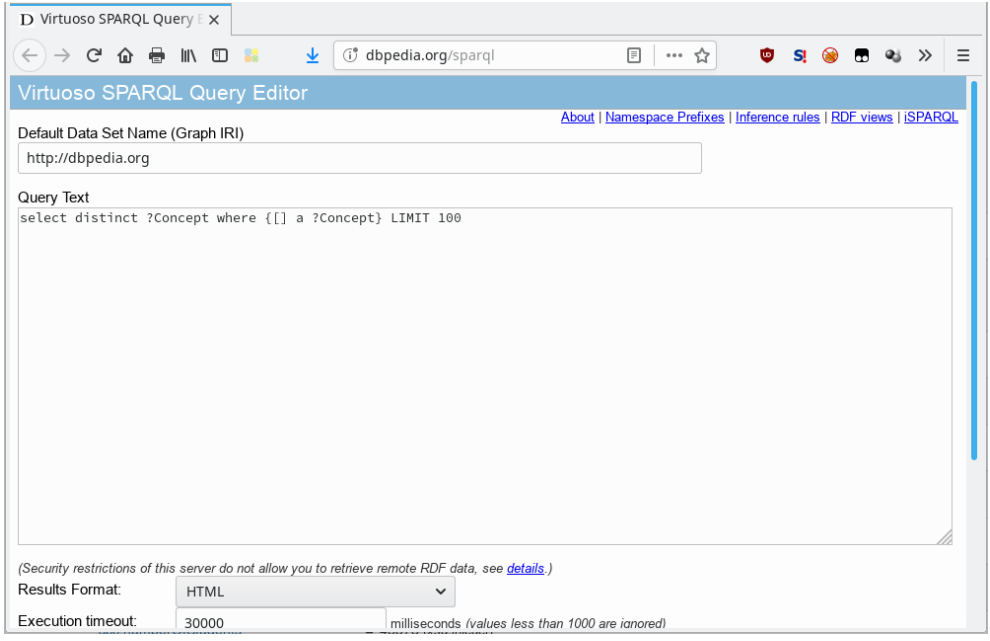
Example 1

```
DESCRIBE <http://dbpedia.org/resource/Concordia_University>
```

Example 2

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
DESCRIBE ?s
WHERE { ?s geo:lat "45.497002"^^xsd:float .
        ?s geo:long "-73.578003"^^xsd:float . }
```

Public SPARQL Endpoint



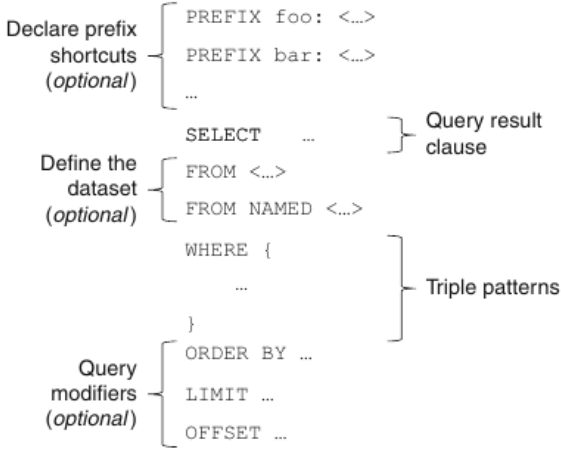
→ **Worksheet #3: Task 5**

René Witte

Selecting triples from a graph

Probably the most widely used type of SPARQL query

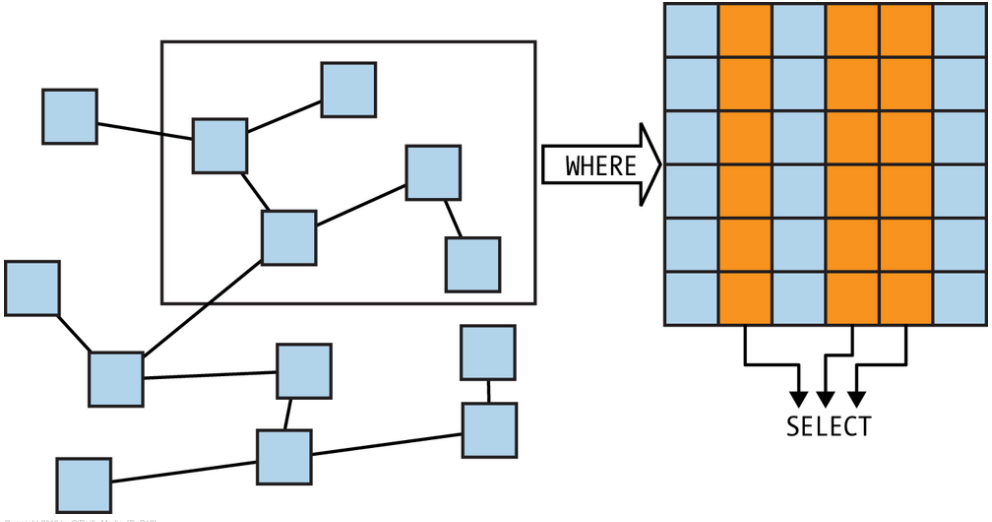
- Select triples from a graph that match a given triple pattern
- Like an RDF triple, except subject, predicate, and/or object may be a variable



René Witte

Select. . . where

René Witte



Select: some details

DISTINCT

Use `SELECT DISTINCT` to remove redundant triples

ORDER BY

Use `ORDER BY` to sort the result triples (e.g., `ORDER BY ?amount`)

LIMIT

Use `LIMIT` to restrict the number of results (e.g., `LIMIT 10`)

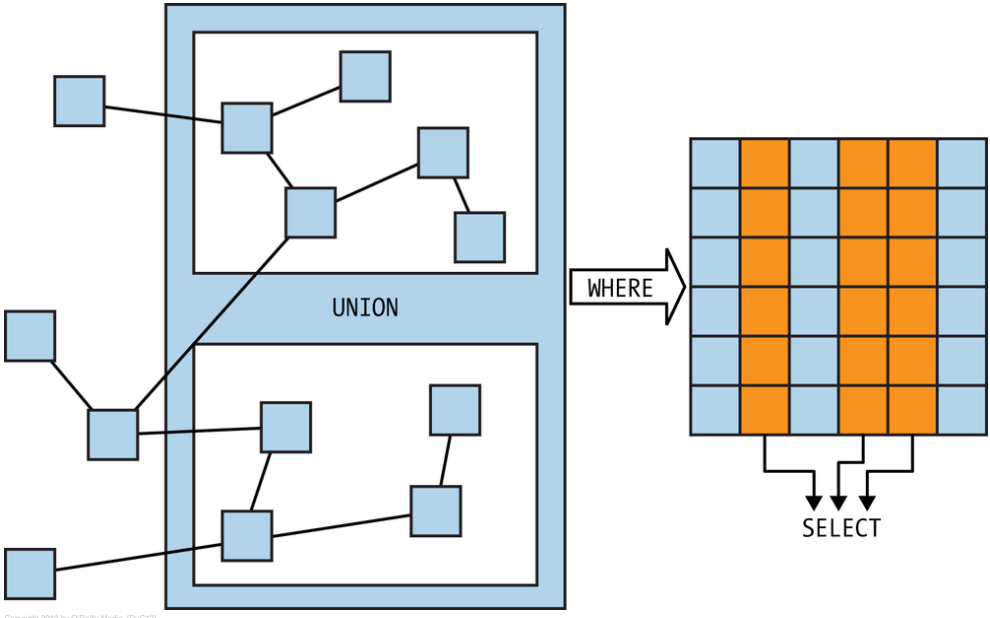
Functions

You can use functions like `AVG(), MIN(), MAX(), COUNT(), SUM()`, e.g.,

```
SELECT (MAX(?amount) as ?maxAmount)
WHERE { ?meal e:amount ?amount . }
```

→ **Worksheet #3: Task 6**

Union



René Witte

Union: Querying Alternatives

René Witte
Concordia University

Example (using the Gene Ontology)

"Find me the cellular processes that are either integral to or a refinement of signal transduction."

```
PREFIX go: <http://purl.org/obo/owl/GO#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX obo: <http://www.obofoundry.org/ro/ro.owl#>

SELECT DISTINCT ?label ?process
WHERE {
  { ?process obo:part_of go:GO_0007165 }      # "integral to"
      UNION
  { ?process rdfs:subClassOf go:GO_0007165 } # "refinement of"
  ?process rdfs:label ?label
}
```

René Witte
Concordia University

Optional Information

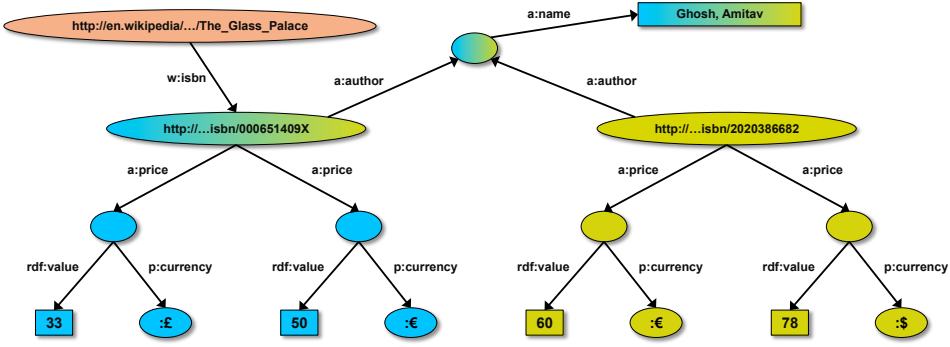
Use the `OPTIONAL` keyword to match optional information, e.g.,

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix foaf: <http://xmlns.com/foaf/0.1/>

select ?name ?url
where {
    ?person foaf:name ?name .
    OPTIONAL { ?person rdfs:seeAlso ?url }
}
```

- This will return a person's URL, if there is one
- Without the `OPTIONAL`, persons without URLs would not have been matched

Optional pattern

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

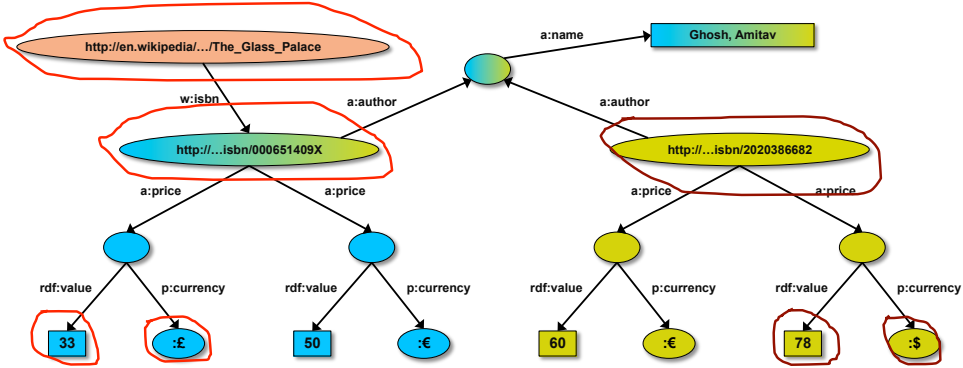


René Witte

Optional pattern

```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

Returns: [[<..09X>,33,:£,<…Palace>], … , [<..6682>,78,:$,]]



→ **Worksheet #3: Task 7**

René Witte

René Witte
Concordia University

Filtering Information

Use a `FILTER` to remove results that were matched by `WHERE`, e.g.:

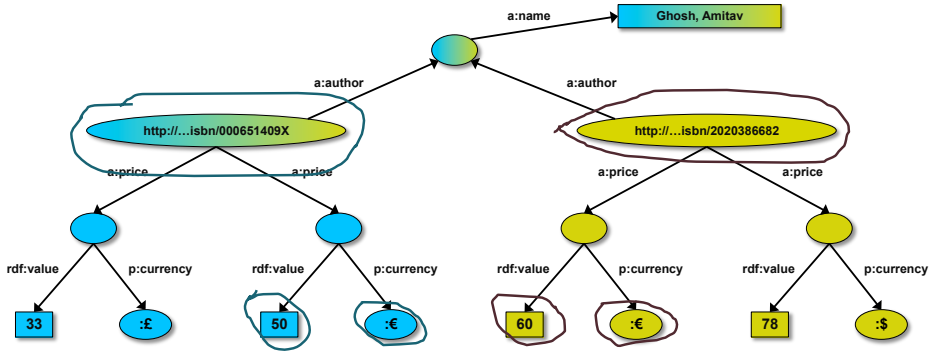
```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?comment
WHERE {
    dbr:Linked_data rdfs:comment ?comment .
    FILTER (lang(?comment) = "en")
}
```

- Here, we restrict all matched abstracts to those with an English language tag.
- `FILTER`s can operate on numbers, strings, dates, URIs, or other data types and support regular expressions.

Pattern constraints

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == :€) }
```

Returns: [<…409X>,50,:€], [<…6682>,60,:€]



→ **Worksheet #3: Task 8**

René Witte
Concordia University

Construct

René Witte

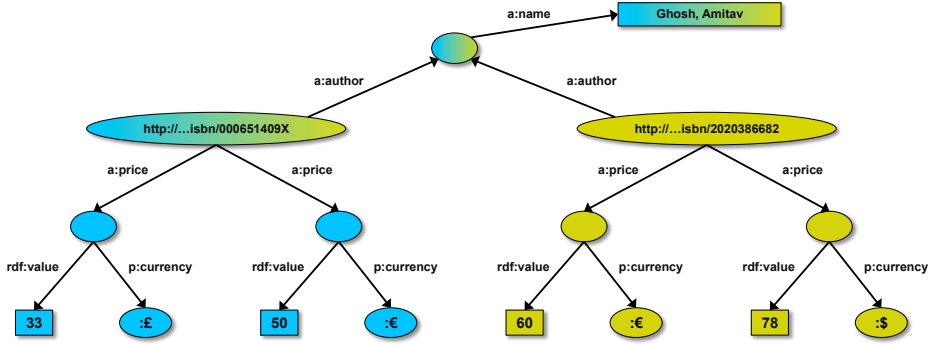
Constructing a new graph

Can be used to re-construct a new graph from an existing one.

- For example, re-write triples from one vocabulary into another

Construct a new graph

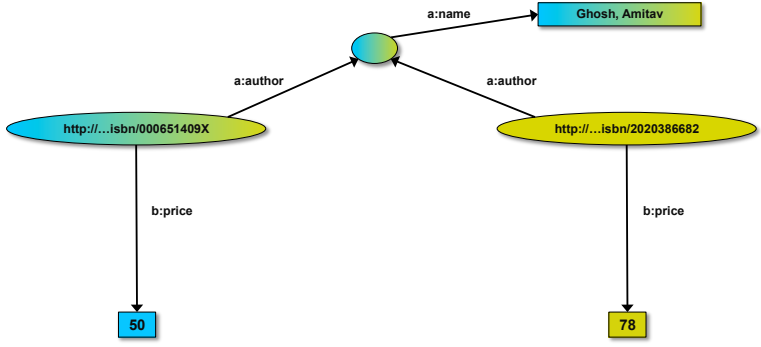
```
CONSTRUCT { ?isbn b:price ?price.
            ?isbn a:author ?y. ?y a:name ?name . }
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        ?isbn a:author ?y. ?y a:name ?name .
        FILTER(?currency == :€) }
```



René Witte

Construct a new graph

```
CONSTRUCT { ?isbn b:price ?price.
            ?isbn a:author ?y. ?y a:name ?name . }
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        ?isbn a:author ?y. ?y a:name ?name .
        FILTER(?currency == :€) }
```



René Witte
Concordia University

Ask

Asking a true/false question

ASK <graph pattern>

Returns `true` if the pattern can be matched in the graph, otherwise `false`

Example

Is Concordia University located in Mexico?

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

ASK {
    dbr:Concordia_University dbo:country dbr:Mexico
}
```

→ **Worksheet #3: Task 9**

Other SPARQL features

- Limit the number of returned results; remove duplicates, sort them, …
- Specify several data sources (via URI-s) within the query
- Construct a graph combining a separate pattern and the query results
- Use datatypes and/or language tags when matching a pattern
- Aggregation of the results (min, max, average, etc.)
- Path expressions (a bit like regular expressions)

René Witte
Concordia University

SPARQL usage in practice

- SPARQL is usually used over the network
  - http request is sent to a SPARQL endpoint
  - result is the result of the SELECT, the CONSTRUCT,…
- Separate documents define the protocol and the result format
  - SPARQL Protocol for RDF with HTTP and SOAP bindings
  - SPARQL results in XML or JSON formats
- Big datasets usually offer “SPARQL endpoints” using this protocol

René Witte
Concordia University

Remote query/reply example

```
GET /qps?&query=SELECT+:…+WHERE:+… HTTP/1.1
User-Agent: my-sparql-client/0.0
Host: my.example

HTTP/1.1 200 OK
Server: my-sparql-server/0.0
Content-Type: application/sparql-results+xml

<?xml version="1.0" encoding="UTF-8"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#>
  <head>
    <variable name="a"/>
    ...
  </head>
  <results>
    <result ordered="false" distinct="false">
      <binding name="a"><uri>http:…</uri></binding>
      ...
    </result>
    <result> ... </result>
  </results>
</sparql>
```

René Witte
Concordia University

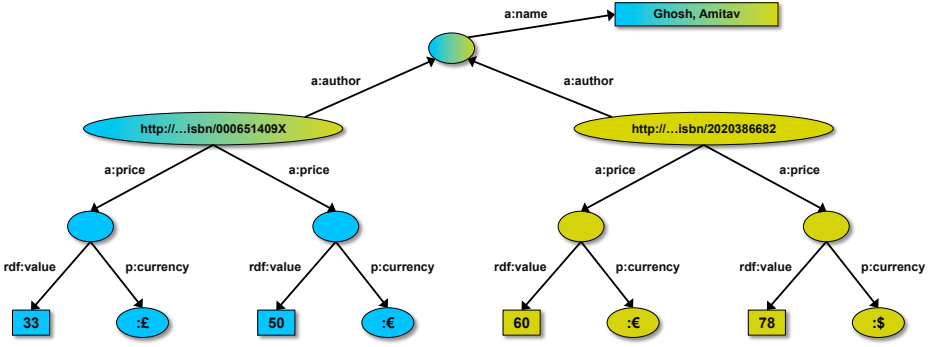
SPARQL 1.1 Update

- SPARQL CONSTRUCT returns a new, modified graph
  - the original data remains unchanged!
- SPARQL 1.1 Update *modifies the original dataset!*

René Witte
Concordia University

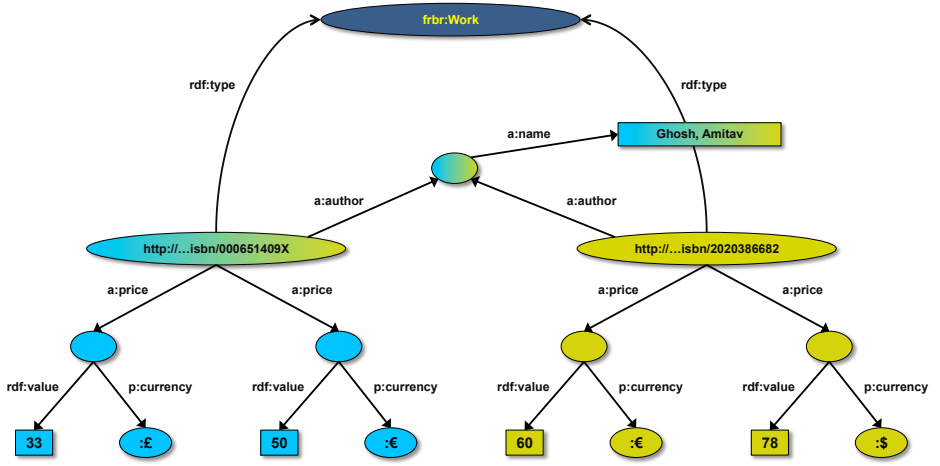
Update: insert

```
INSERT {?isbn rdf:type frbr:Work}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Update: insert

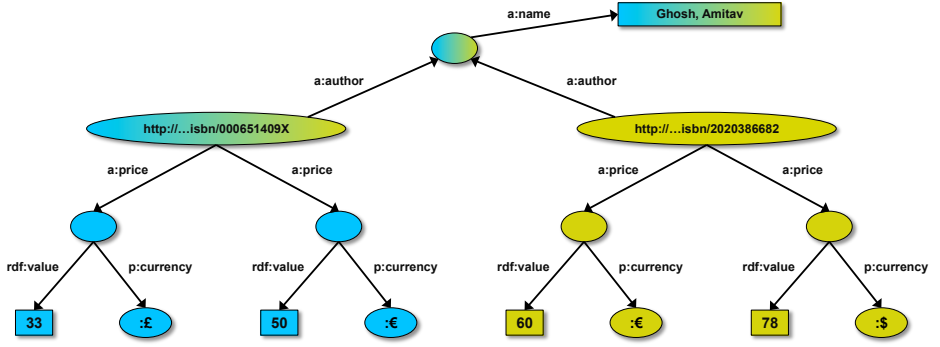
```
INSERT {?isbn rdf:type frbr:Work}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



René Witte
Concordia University

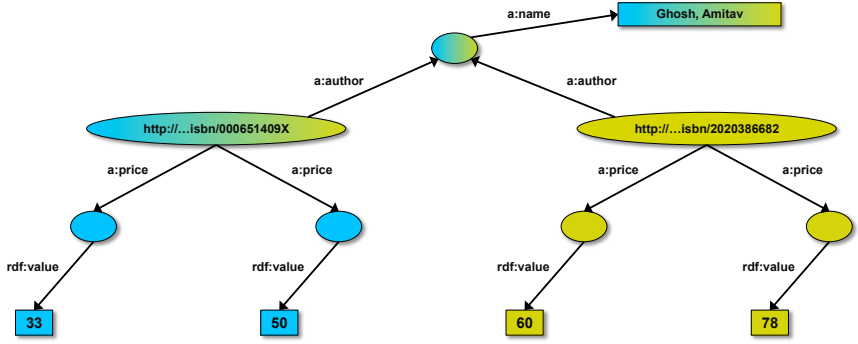
Update: delete

```
DELETE {?x p:currency ?currency}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



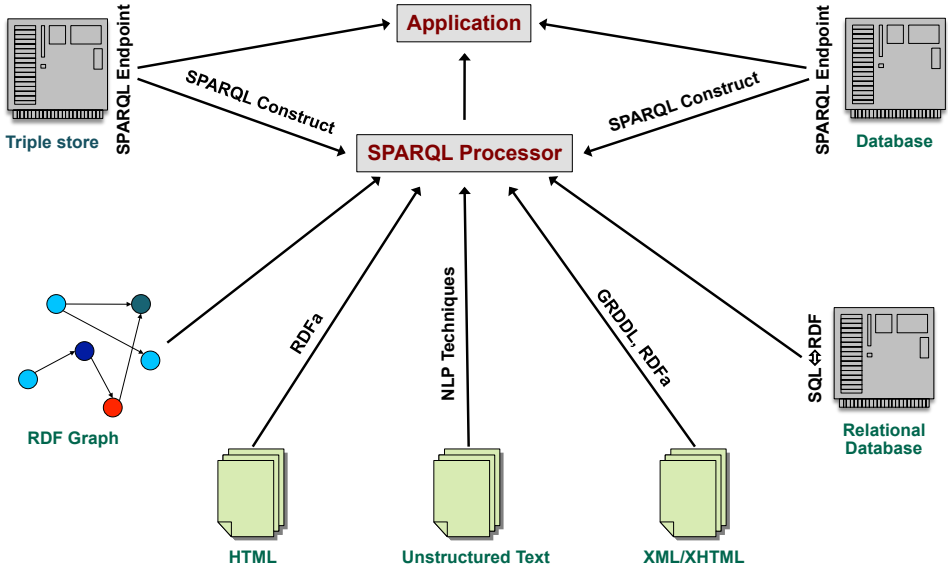
Update: delete

```
DELETE {?x p:currency ?currency}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



René Witte
Concordia University

SPARQL as a unifying point



René Witte

Outline

Named Graphs

RDF Dataset

An RDF dataset may have multiple named graphs and at most one unnamed ("default") graph.

Serialization

TriG:	Extension of Turtle for named graphs See https://www.w3.org/TR/trig/
N-Quads:	Extension of N-Triples for named graphs See https://www.w3.org/TR/n-quads/

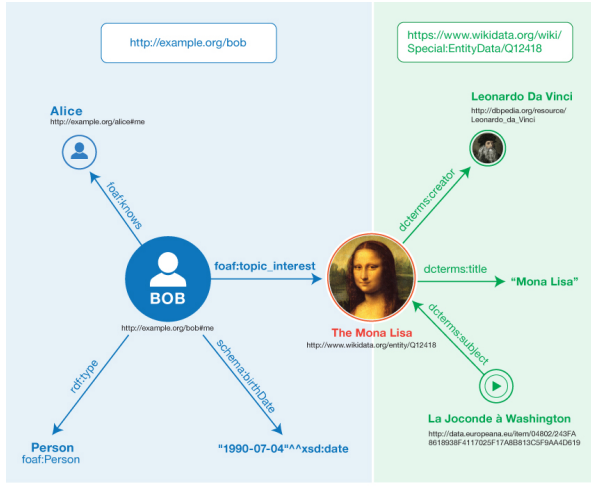
TriG Example

René Witte
Concordia University


```
BASE   <http://example.org/>
PREFIX ...

GRAPH <http://example.org/bob>
  {
    <bob#me>
        a foaf:Person ;
        foaf:knows <alice#me> ;
        schema:birthDate "1990-07-04"^^xsd:date ;
        foaf:topic_interest wd:Q12418 .
  }

GRAPH <https://www.wikidata.org/wiki/Special:EntityData/Q12418>
  {
    wd:Q12418
        dcterms:title "Mona Lisa" ;
        dcterms:creator <http://dbpedia.org/resource/Leonardo_da_Vinci> .

    <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>
       dcterms:subject wd:Q12418 .
  }

<http://example.org/bob>
    dcterms:publisher <http://example.org> ;
    dcterms:rights <http://creativecommons.org/licenses/by/3.0/> .
```

See https://www.w3.org/TR/rdf11-primer/

Named Graphs Example



https://www.w3.org/TR/rdf11-primer/

René Witte

Named Graphs: N-Quads

N-Quads

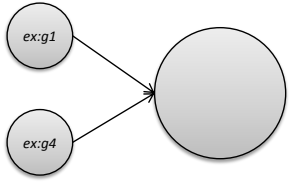
N-Quads add a fourth element to a line, capturing the graph IRI of the triple described on that line

Example

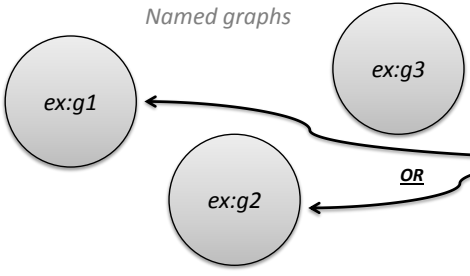
```
<http://example.org/bob#me> ↩
      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ↩
         <http://xmlns.com/foaf/0.1/Person> <http://example.org/bob> .
...
<http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/title> ↩
      "Mona Lisa" <https://www.wikidata.org/wiki/Special:EntityData/Q12418> .
...
<http://example.org/bob> <http://purl.org/dc/terms/rights> ↩
      <http://creativecommons.org/licenses/by/3.0/> .
```

A SPARQL queries a *default graph* (normally) and zero or more *named graphs* (when inside a **GRAPH** clause).

Default graph
(the merge of zero or more graphs)



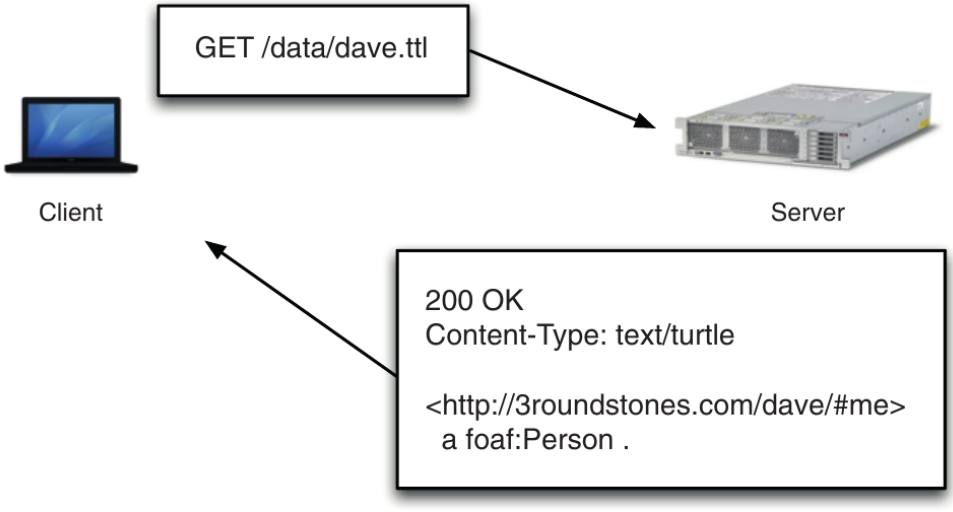
Named graphs




```
PREFIX ex: <…>
SELECT …
FROM ex:g1
FROM ex:g4
FROM NAMED ex:g1
FROM NAMED ex:g2
FROM NAMED ex:g3
WHERE {
   … A …
   GRAPH ex:g3 {
      … B …
   }
   GRAPH ?g {
      … C …
   }
}
```

René Witte

Simple HTTP Request



René Witte

SPARQL Over HTTP (the SPARQL Protocol)

`http://host.domain.com/sparql/endpoint?`*<parameters>*

where *<parameters>* can include:

`query=`*<encoded query string>*

e.g. `SELECT+*%0DWHERE+{…`

`default-graph-uri=`*<encoded graph URI>*

e.g. `http%3A%2F%2Fexmaple.com%2Ffoo…`

n.b. zero of more occurrences of `default-graph-uri`

`named-graph-uri=`*<encoded graph URI>*

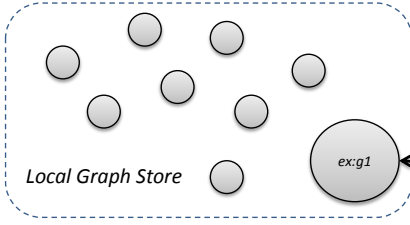
e.g. `http%3A%2F%2Fexmaple.com%2Fbar…`

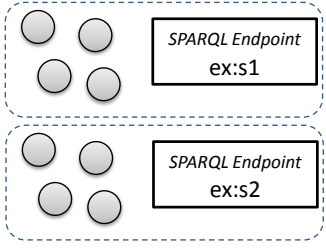
n.b. zero of more occurrences of `named-graph-uri`

HTTP `GET` or `POST`. Graphs given in the protocol override graphs given in the query.

René Witte
Concordia University

Federated Query *(SPARQL 1.1)*






```
PREFIX ex: <…>
SELECT …
FROM ex:g1
WHERE {
    … A …
    SERVICE ex:s1 {
        … B …
    }
    SERVICE ex:s2 {
        … C …
    }
}
```

René Witte

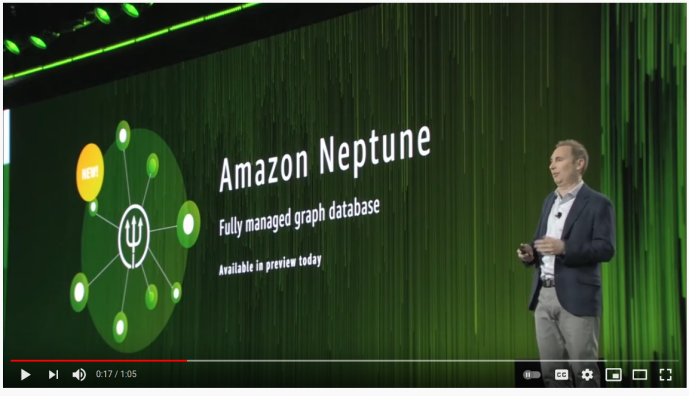
SPARQL Server Engines

Examples

Commercial: Virtuoso (OpenLink Software); has “open source edition” at https://github.com/openlink/virtuoso-opensource

Cloud: Amazon AWS Neptune, see https://aws.amazon.com/neptune/

Open Source: Apache Jena, see https://jena.apache.org/



AWS re:Invent 2017 - Amazon Neptune: Fast, Reliable Graph Database Built for the Cloud

https://www.youtube.com/watch?v=Rl6UwE7kLio

René Witte
Concordia University

Apache Jena Fuseki

Apache Jena Fuseki is a SPARQL server. It can run as a operating system service, as a Java web application (WAR file), and as a standalone server. It provides security (using Apache Shiro) and has a user interface for server monitoring and administration.

It provides the SPARQL 1.1 protocols for query and update as well as the SPARQL Graph Store protocol.

Fuseki is tightly integrated with TDB to provide a robust, transactional persistent storage layer, and incorporates Jena text query. It can be used to provide the protocol engine for other RDF query and storage systems.

Contents

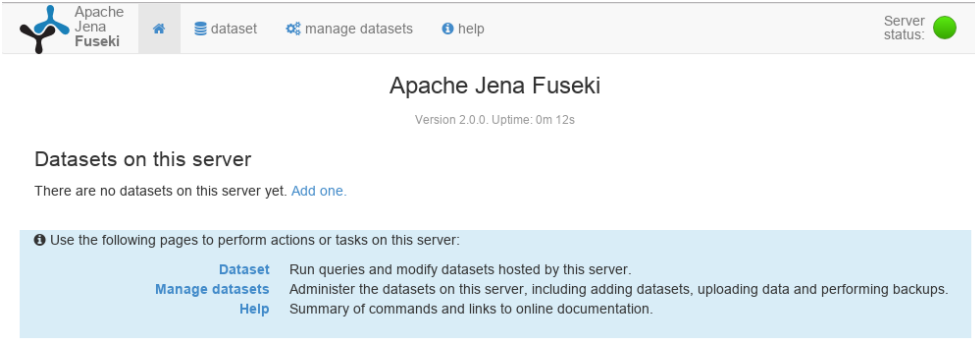
The Jena users mailing is the place to get help with Fuseki.

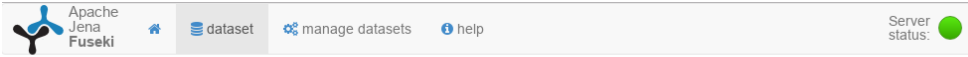
See https://jena.apache.org/documentation/fuseki2/index.html

René Witte
Concordia University

Apache Fuseki (Standalone Server Mode)

René Witte
Concordia University





Dataset: /foo

query | upload files | edit | info

SPARQL query

To try out some SPARQL queries against the selected dataset, enter your query here.

EXAMPLE QUERIES

Selection of triples | Selection of classes

PREFIXES

rdf | rdfs | owl | xsd

SPARQL ENDPOINT	CONTENT TYPE (SELECT)	CONTENT TYPE (GRAPH)
http://localhost:3030/foo/sparql	JSON	Turtle

```
1
2
3  SELECT ?subject ?predicate ?object
4  WHERE {
5    ?subject ?predicate ?object
6  }
7  LIMIT 25
```

René Witte
Concordia University

Inferencing

René Witte

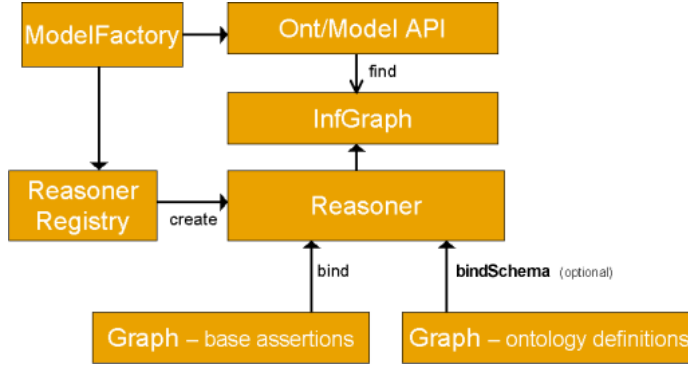
Remember these...

```
ex:Student rdfs:subClassOf foaf:Person
ex:Joe a ex:Student
```

What happens when you query for all `foaf:Person`s?

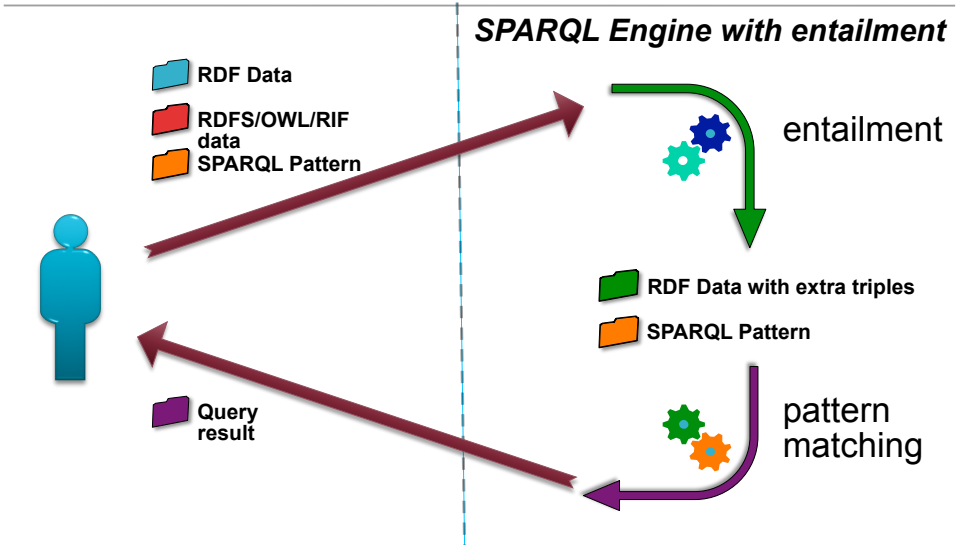
Reasoning Engine

- RDFlib will return empty result
- Requires inference support (e.g., RDFS reasoner, OWL reasoner)



See https://jena.apache.org/documentation/inference/

SPARQL 1.1 and RDFS/OWL/RIF



René Witte
Concordia University

Outline

René Witte

Reading Material

Required

- [Yu14, Chapter 6] (SPARQL)

Supplemental

- [DuC13] (Learning SPARQL)
- [WZRH14, Chapter 5] (SPARQL)
- SPARQL 1.1 Overview, https://www.w3.org/TR/sparql11-overview/

References I

[DFH11] John Domingue, Dieter Fensel, and James A. Hendler, editors.
Handbook of Semantic Web Technologies.
Springer, 2011.
https://concordiauniversity.on.worldcat.org/oclc/769756125.

[DuC13] Bob DuCharme.
Learning SPARQL: Querying and Updating with SPARQL 1.1.
O'Reilly, 2nd edition, 2013.
https://concordiauniversity.on.worldcat.org/oclc/853679890.

[Her] Ivan Herman.
Tutorial on Semantic Web Technologies.
http://www.w3.org/People/Ivan/CorePresentations/RDFTutorial/.

[WZRH14] David Wood, Marsha Zaidman, Luke Ruth, and Michael Hausenblas.
Linked Data: Structured Data on the Web.
Manning, 2014.
https://concordiauniversity.on.worldcat.org/oclc/871683907.

References II

[Yu14] Liyang Yu.
A Developer's Guide to the Semantic Web.
Springer-Verlag Berlin Heidelberg, 2nd edition, 2014.
https://concordiauniversity.on.worldcat.org/oclc/897466408.