Knowledge Representation & Reasoning
&
Introduction To Knowledge Graphs

Week 6 & 7 | Fall 2022

Dr. Amna Basharat

Knowledge Is Power?

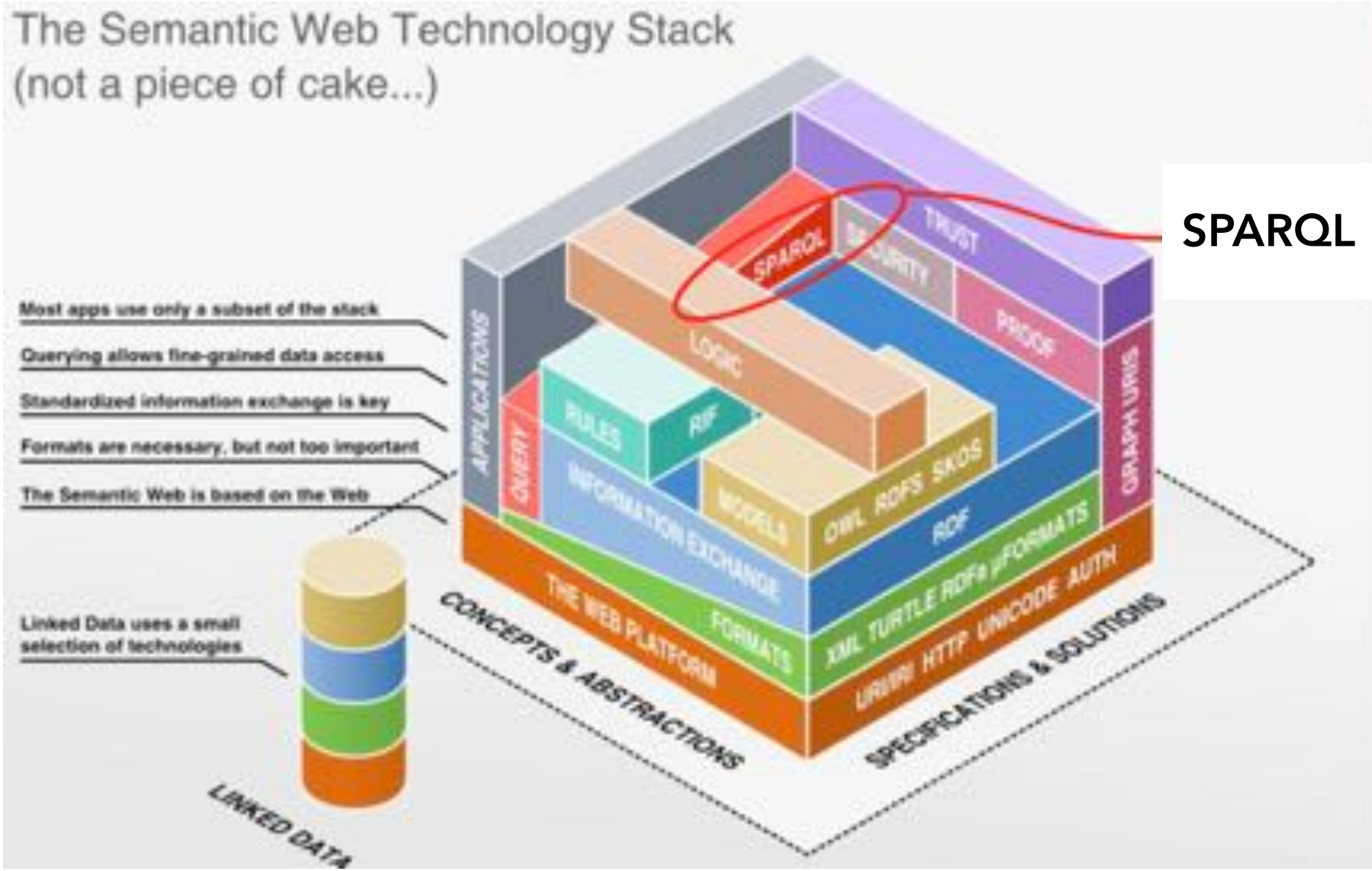
Agree/Disagree?

Knowledge Is the Key to Ultimate Success?

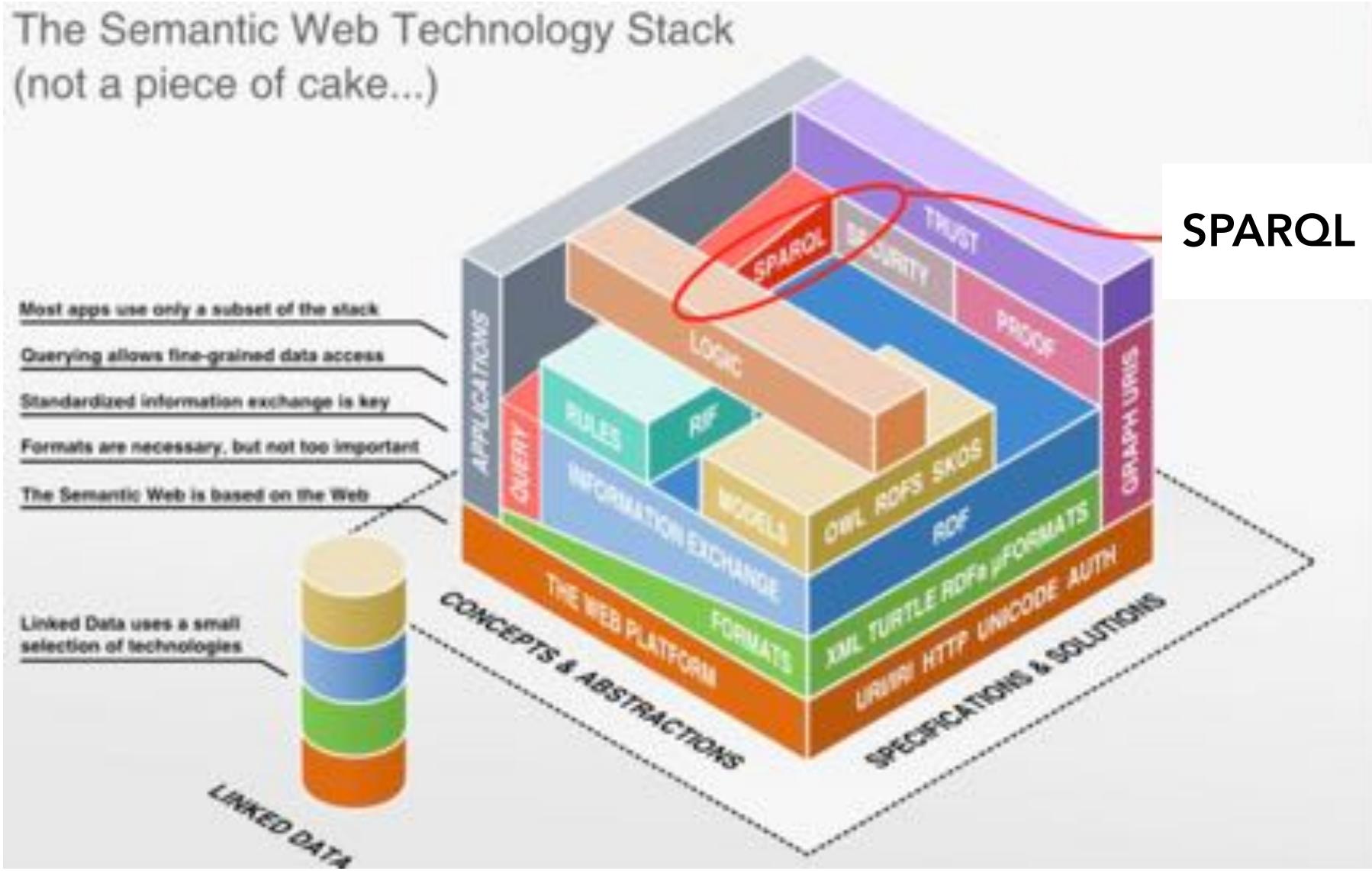
Agree/Disagree

If yes, to what extent?

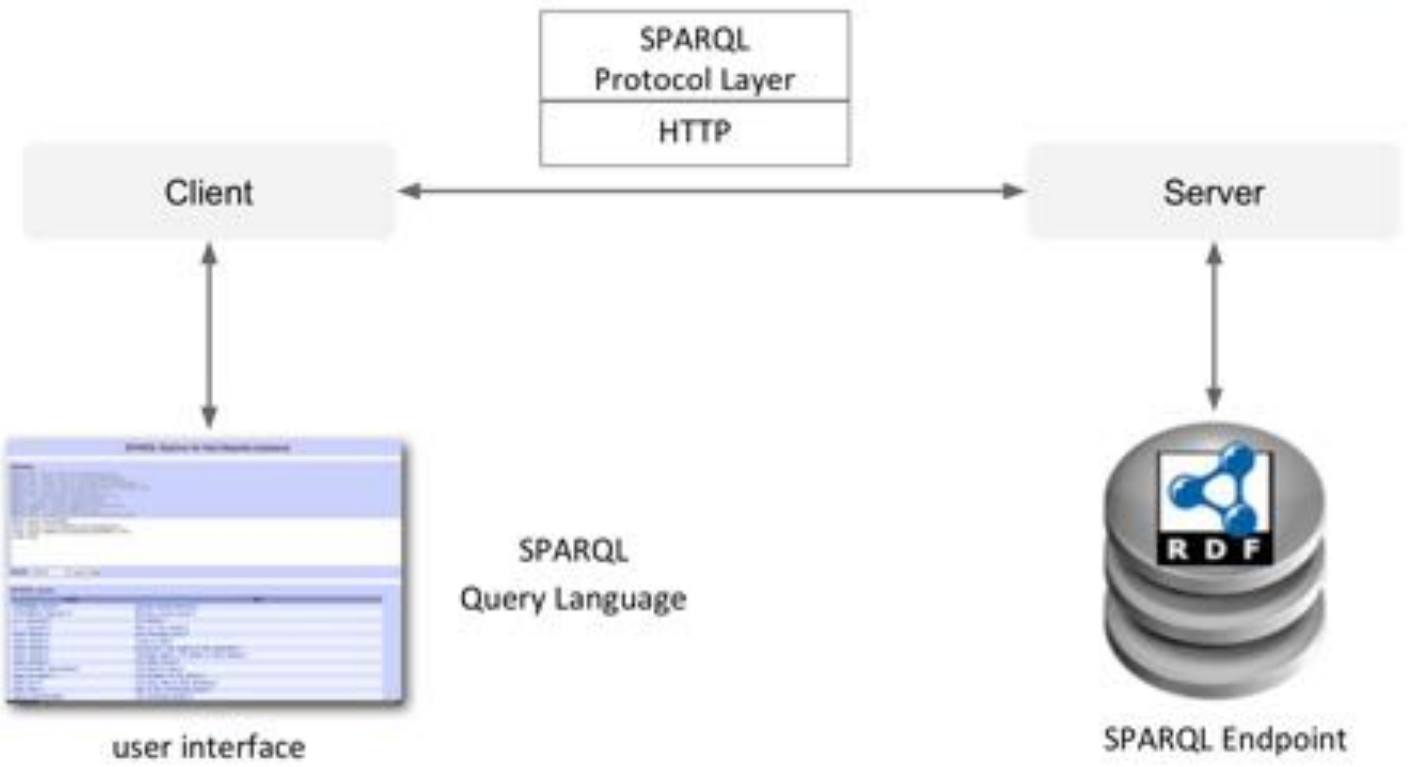
How To Query RDF(S)? - SPARQL

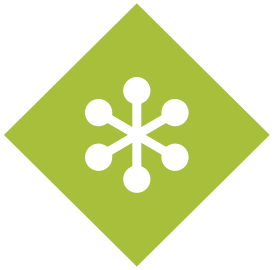

The Semantic Web Technology Stack
(not a piece of cake...)
SPARQL
Most apps use only a subset of the stack
Querying allows fine-grained data access
Standardized information exchange is key
Formats are necessary, but not too important
The Semantic Web is based on the Web
Linked Data uses a small selection of technologies
LINKED DATA
CONCEPTS & ABSTRACTIONS
SPECIFICATIONS & SOLUTIONS
APPLICATIONS
QUERY
LOGIC
RULES
RIF
INFORMATION EXCHANGE
VOCABS
OWL RDFS SKOS
RDF
TRUST
PROOF
GRAPH URIS
THE WEB PLATFORM
FORMATS
XML TURTLE RDFa μFORMATS
URI/IRI HTTP UNICODE AUTH

How To Query RDF(S)? - SPARQL


The Semantic Web Technology Stack
(not a piece of cake...)
SPARQL
Most apps use only a subset of the stack
Querying allows fine-grained data access
Standardized information exchange is key
Formats are necessary, but not too important
The Semantic Web is based on the Web
Linked Data uses a small selection of technologies
LINKED DATA
CONCEPTS & ABSTRACTIONS
SPECIFICATIONS & SOLUTIONS
APPLICATIONS
QUERY
LOGIC
RULES
RIF
INFORMATION EXCHANGE
VOCABS
OWL RDFS SKOS
RDF
TRUST
PROOF
GRAPH URIS
THE WEB PLATFORM
FORMATS
XML TURTLE RDFa μFORMATS
URI/IRI HTTP UNICODE AUTH

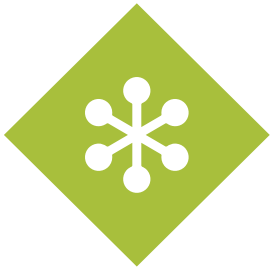
SPARQL - A Query Language for RDF





SPARQL - A Query Language for RDF

- **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage is
  - a Query Language for RDF graph traversal (SPARQL Query Language Specification)
  - a Protocol Layer, to use SPARQL via http (SPARQL Protocol for RDF Specification)
  - an XML Output Format Specification for SPARQL queries (SPARQL Query XML Results Format)
  - W3C Standard (SPARQL 1.1, Mar 2013)
  - inspired by SQL



SPARQL - A Query Language for RDF

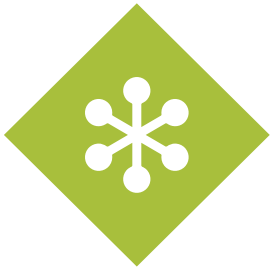
- SPARQL Features:
  - Extraction of Data as
    - *RDF Subgraphs, URIs, Blank Nodes, typed and untyped Literals*
    - *with aggregate functions, subqueries, complex joins, property paths*
  - Exploration of Data via Query for unknown relations
  - Transformation of RDF Data from one vocabulary into another
  - Construction of new RDF Graphs based on RDF Query Graphs
  - Updates of RDF Graphs as full data manipulation language
  - Logical Entailment for RDF, RDFS, OWL, and RIF Core entailment.
  - Federated Queries distributed over different SPARQL endpoints

For Queries We Need Variables

- SPARQL **Variables** are bound to RDF terms
  - e.g. **?title, ?author, ?address**
- In the same way as in SQL, a **query for variables** is performed via **SELECT statement**
  - e.g. **SELECT ?title ?author ?published** SPARQL Query
- A SELECT statement returns Query Results as a **table**

?title	?author	?published
1984	George Orwell	1948
Brave New World	Aldous Huxley	1932
Fahrenheid 451	Ray Bradbury	1953

SPARQL Result



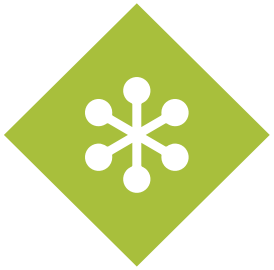
SPARQL - Graph Pattern Matching

- SPARQL is based on **RDF Turtle serialization** and **basic graph pattern matching.**
- A **Graph Pattern (Triple Pattern)** is a RDF Triple that contains variables at any arbitrary place (Subject, Property, Object).
  (Graph) Triple Pattern = Turtle + Variables

- Example:
  Look for countries and their capitals:
  - `?country dbo:capital ?capital .`

- A **Basic Graph Pattern (BGP)** is a set of Triple Pattern

SPARQL - Graph Pattern Matching



SPARQL - Complex Query Patterns

- SPARQL Graph Pattern can be combined to form **complex (conjunctive) queries** for RDF graph traversal

- Find countries, their capitals, and their population count:
  - **?country dbo:capital ?capital .**
    ?country dbo:population ?population .

- Given a FOAF URI, find the name of a person and her friends:

```
<http://hpi-web.de/id#haraldsack> foaf:name ?surname ;
                                  foaf:knows ?friend .
?friend foaf:name ?friend_surname .
```

SPARQL - General Query Format

- *search all authors and the titles of their notable works:*

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

SELECT ?author_name ?title

FROM <http://dbpedia.org/>

WHERE {
     ?author rdf:type dbo:Writer .
     ?author rdfs:label ?author_name .
     ?author dbo:notableWork ?work .
     ?work rdfs:label ?title .
}
```

specifies namespaces

specifies output variables

specifies graph to be queried

specifies graph pattern to be matched

SPARQL - General Query Format

- Search all authors and the titles of their notable works ordered by authors in ascending order and limit the results to the first 100 results starting the list at offset 10 position:

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

SELECT ?author_name ?title

FROM <http://dbpedia.org/>

WHERE {
      ?author rdf:type dbo:Writer .
      ?author rdfs:label ?author_name .
      ?author dbo:notableWork ?work .
      ?work rdfs:label ?title .
}
ORDER BY ASC (?author_name)
LIMIT 100
OFFSET 10
```

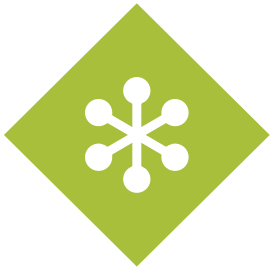
solution sequence modifiers

SPARQL - Filter Constraints

- FILTER expressions contain operators and functions
- FILTER can NOT assign/create new values

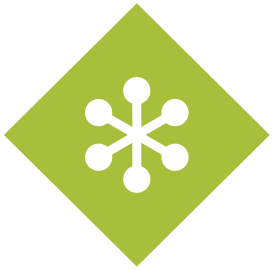
```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer .
        ?author rdfs:label ?author_name .
        ?author dbo:notableWork ?work .
        ?work dbo:numberOfPages ?pages
        FILTER (?pages > 500) .
        ?work rdfs:label ?title .
} LIMIT 100
```

specifies constraints for the result



SPARQL - Unary Operator Constraints

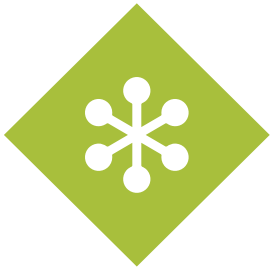
Operator	Type(A)	Result Type
`!A`	xsd:boolean	xsd:boolean
`+A`	numeric	numeric
`-A`	numeric	numeric
`BOUND(A)`	variable	xsd:boolean
`isURI(A)`	RDF term	xsd:boolean
`isBLANK(A)`	RDF term	xsd:boolean
`isLITERAL(A)`	RDF Term	xsd:boolean
`STR(A)`	literal/URL	simple literal
`LANG(A)`	literal	simple literal
`DATATYPE(A)`	literal	URI



SPARQL - Filter Constraints

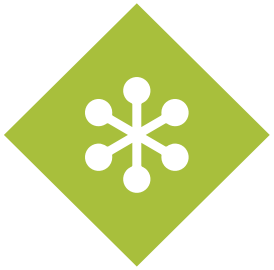
- Example: Filter results only for English labels

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer .
        ?author rdfs:label ?author_name
        FILTER (LANG(?author_name)="en").
        ?author dbo:notableWork ?work .
        ?work dbo:numberOfPages ?pages
        FILTER (?pages > 500) .
        ?work rdfs:label ?title .
        FILTER (LANG(?title)="en").
} LIMIT 100
```



SPARQL - First Hands on

- From Wikipedia to DBpedia
  e.g. fromhttps://en.wikipedia.org/wiki/Muhammad_Ali_Jinnah to http://dbpedia.org/page/Muhammad_Ali_Jinnah
- Browsing DBpedia
  e.g. usinghttp://dbpedia.org/page/Muhammad_Ali_Jinnah as a starting point to learn more about DBpedia structure and DBpedia ontologies
- Using DBpedia Sparql Endpoint with http://dbpedia.org/sparql and query DBpedia via SPARQL

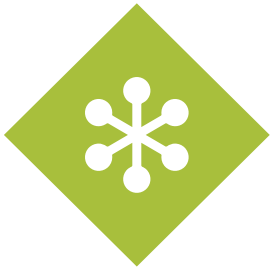


Exercise 1:

- Write a query to retrieve the names of parents of Muhammad Ali Jinnah
- Write a query to answer: Who are the successors of Muhammad Ali Jinnah?

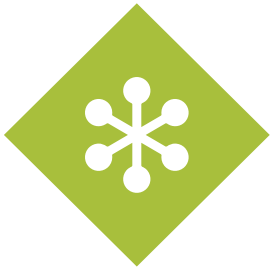
SPARQL - A Query Language for RDF

SPARQL Result Format



SPARQL - A Query Language for RDF

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF graph traversal (SPARQL Query Language Specification)
  - a **Protocol Layer**, to use SPARQL via http (SPARQL Protocol for RDF Specification)
  - an **XML Output Format Specification** for SPARQL queries (SPARQL Query XML Results Format)



SPARQL Result Format

- SPARQL results are given as well formed and valid XML documents

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

- In a <head> element all variables of the SPARQL query are listed

```
<head>
    <variable name="x"/>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="mbox"/>
    <variable name="blurb"/>
</head>
```

SPARQL Result Format

- For each SPARQL Query result exists a <result> element

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="x"/>
    ...
  </head>
  <results>
    <result>
      <binding name="x"> ... </binding>
      <binding name="hpage"> ... </binding>
    </result>

    <result> ... </result>
    ...
  </results>
</sparql>
```

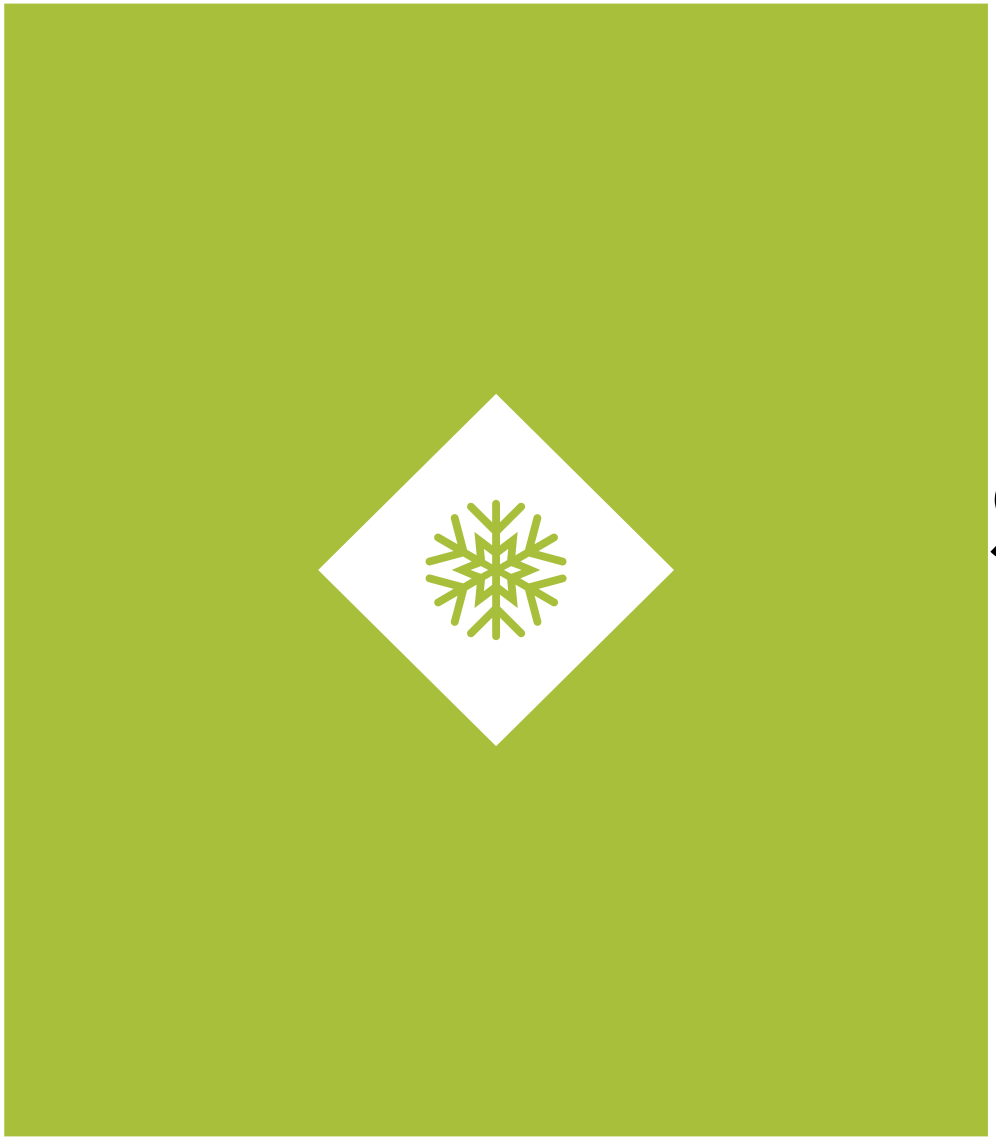
single SPARQL query result

SPARQL Result Format

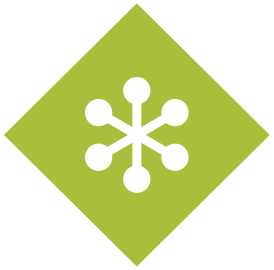
- Within a <binding> element a <head> variable is bound to a result

```
<result>
      <binding name="x">
        <bnode>r2</bnode>
      </binding>
      <binding name="hpage">
        <uri>http://work.example.org/bob/</uri>
      </binding>
      <binding name="name">
        <literal xml:lang="en">Bob</literal>
      </binding>
      <binding name="age">
        <literal datatype="http://www.w3.org/2001/XMLSchema#integer">
         30
        </literal>
      </binding>
      <binding name="mbox">
        <uri>mailto:bob@work.example.org</uri>
      </binding>
    </result>
```

variable bound to result

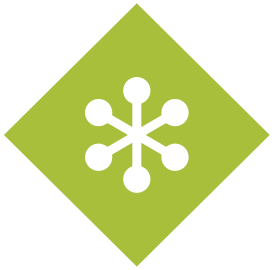


SPARQL Protocol



SPARQL - A Query Language for RDF

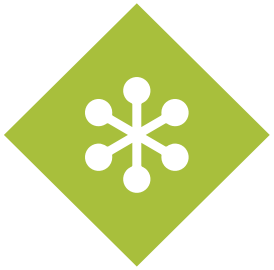
- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF graph traversal (SPARQL Query Language Specification)
  - a **Protocol Layer**, to use SPARQL via http (SPARQL Protocol for RDF Specification)
  - an **XML Output Format Specification** for SPARQL queries (SPARQL Query XML Results Format)



SPARQL Protocol

- Method to query/respond of SPARQL queries via http
- A SPARQL URI consists out of 3 parts:
  - (1) URL of a SPARQL endpoint (e.g. http://example.org/sparql)
  - (2) RDF Graph(s) to be queried
    (optional, part of the query string,
    e.g. named-graph-uri=http://example.org/testrdf.rdf)
  - (3) SPARQL query
    (part of the query string, e.g. query=SELECT...)

```
http://example.org/sparql?named-graph-uri=http%3A%2F%2Fexample.org%2Ftestrdf&
query=SELECT+%3Freview_graph+WHERE+%7B%0D%0A++GRAPH+%3Frev
iew_graph+%7B%0D%0A+++++%3Freview+rev%3Arating+10+.%0D%0A++%7D%0D%
0A%7D
```



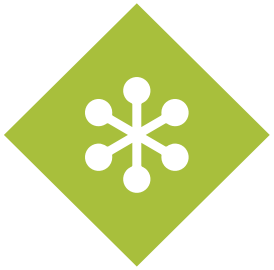
SPARQL Protocol - Example

- Simple SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author  ?work
WHERE {
       ?author rdf:type dbo:Writer ;
               dbo:notableWork ?work .
} LIMIT 100
```

- HTTP Trace of the SPARQL query

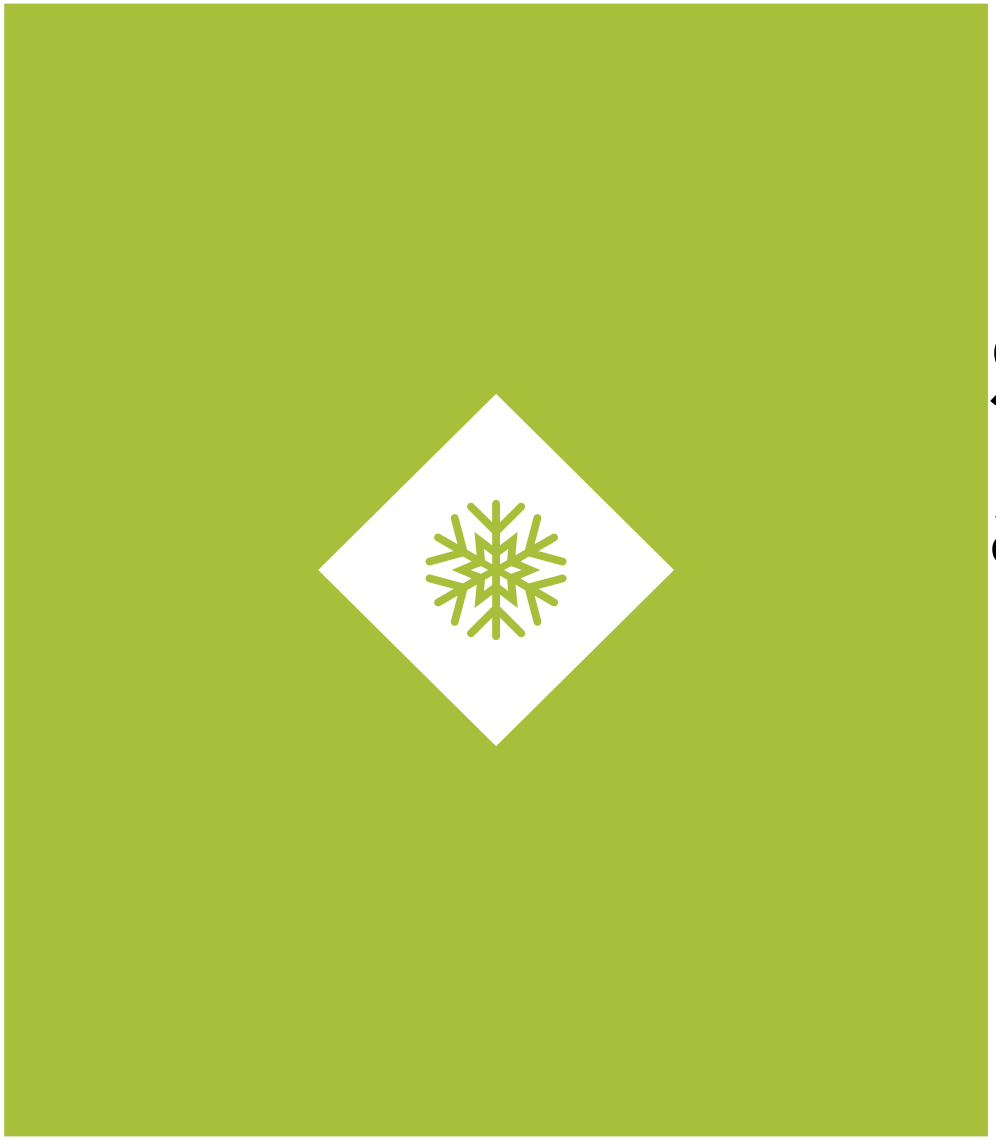
```
GET
http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+rdf%3A+%
3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns%23%3E%0D%0APREFIX+dbo%
3A+%3Chttp%3A%2F%2Fdbpedia.org%2Fontology%2F%3E%0D%0ASELECT+%3Fauthor++%3Fwork%
0D%0AWHERE+%7B%0D%0A+++++++%3Fauthor+rdf%3Atype+dbo%3AWriter+%3B%0D%
0A+++++++++++++++dbo%3AnotableWork+%3Fwork+.%0D%0A%7D+LIMIT+100%0D%0A
Host: dbpedia.org
User-agent: Mozilla/5.0 ...
Accept:text/html,application/xhtml+xml,application/xml
```



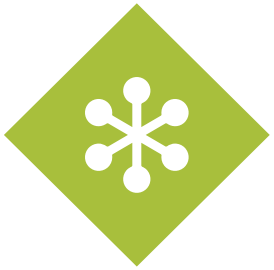
SPARQL Protocol - Example

- HTTP Trace of the SPARQL response

```
HTTP/1.1 200 OK
Date: Tue, 18 Aug 2015 09:55:07 GMT
Content-Type: application/sparql-results+xml; charset=UTF-8
Content-Length: 21055
Connection: keep-alive
Server: Virtuoso/07.20.3214 (Linux) x86_64-redhat-linux-gnu  VDB
X-SPARQL-default-graph: http://dbpedia.org
...
<sparql xmlns="http://www.w3.org/2005/sparql-results#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://www.w3.org/2001/sw/DataAccess/rf1/result2.xsd">
 <head>
  <variable name="author"/>
  <variable name="work"/>
 </head>
 <results distinct="false" ordered="true">
  <result>
   <binding name="author"><uri>http://dbpedia.org/resource/Ding_Ling</uri></binding>
   <binding name="work"><uri>http://dbpedia.org/resource/Miss_Sophia&#39;s_Diary</uri></binding>
  </result>
  ...
</results>
</sparql>
```

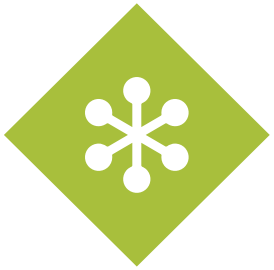


SPARQL Is Not Only a Query Language



SPARQL - A Query Language for RDF

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF graph traversal (SPARQL Query Language Specification)
  - a **Protocol Layer**, to use SPARQL via http (SPARQL Protocol for RDF Specification)
  - an **XML Output Format Specification** for SPARQL queries (SPARQL Query XML Results Format)



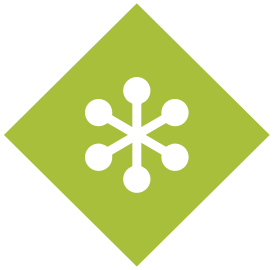
SPARQL Is Not Only a Query Language

- In addition to SELECT queries SPARQL allows:
- ASK
  - Check whether there is at least one result
  - Result: true or false
  - Result is delivered as XML or JSON

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

ASK
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
}
```

- Example: Is there an author with a notable work?

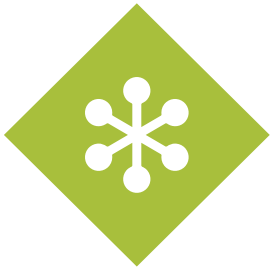


SPARQL Is Not Only a Query Language

- In addition to SELECT queries SPARQL allows:
- DESCRIBE
  - Result: an RDF graph with data about resources
  - Result is RDF/XML or Turtle

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

DESCRIBE ?author ?work
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
} LIMIT 10
```



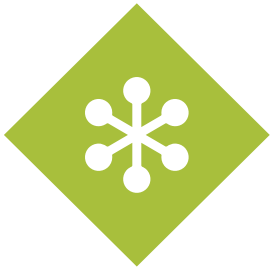
SPARQL Is Not Only a Query Language

- In addition to SELECT queries SPARQL allows:
- CONSTRUCT
  - Result: an RDF graph constructed from a template
  - Template: graph pattern with variables from the query pattern
  - Result is RDF/XML or Turtle

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

CONSTRUCT { ?author <http://example.org/hasWritten> ?work .}
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
} LIMIT 10
```

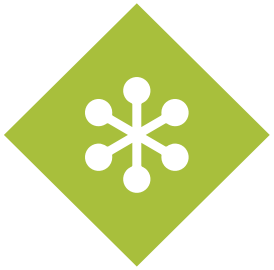
Complex Queries With SPARQL



SPARQL - Filter Constraints

- Example: Filter results only for English labels

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author rdfs:label ?author_name
       FILTER (LANG(?author_name)="en").
       ?author dbo:notableWork ?work .
       ?work dbo:numberOfPages ?pages
       FILTER (?pages > 500) .
       ?work rdfs:label ?title .
       FILTER (LANG(?title)="en").
} LIMIT 100
```



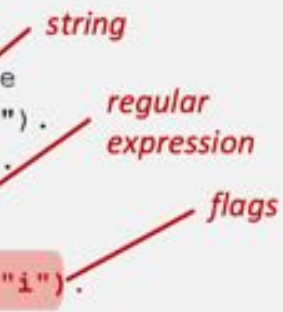
More SPARQL Operators

- Logical connectives && and || for xsd:boolean
- Comparison operators =, !=, <, >, <=, and >= for numeric datatypes, xsd:dateTime, xsd:string, and xsd:boolean
- Comparison operators = and != for other datatypes
- Arithmetic operators +, -, *, and / for numeric datatypes
- and in addition:
  - REGEX(String,Pattern)orREGEX(String,Pattern,Flags)
  - sameTERM(A,B)
  - langMATCHES(A,B)

SPARQL - Filter Constraints

- Example: Book titles that contain the word “love”

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer .
        ?author rdfs:label ?author_name
        FILTER (LANG(?author_name)="en").
        ?author dbo:notableWork ?work .
        ?work rdfs:label ?title .
        FILTER (LANG(?title)="en")
        FILTER REGEX (?title, "love", "i").
} LIMIT 100
```



https://regexone.com

SPARQL - Filter Constraint Evaluation

- SPARQL Filter Constraints are evaluated in 3-valued Logic
- truth values: true, false, and error

A	B	A || B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	E	T	E
E	T	T	E
F	E	E	F
E	F	E	F
E	E	E	E

binary operators

A	!A
T	F
F	T
E	E

unary operators

SPARQL - Filter Constraints OPTIONAL

- Example: Retrieve also the German book title, if available

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?en_title ?de_title
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author rdfs:label ?author_name
       FILTER (LANG(?author_name)="en").
       ?author dbo:notableWork ?work .
       ?work rdfs:label ?en_title .
       FILTER (LANG(?en_title)="en") .
       OPTIONAL {?work rdfs:label ?de_title
                 FILTER (LANG(?de_title)="de") .
      }
} LIMIT 100
```

optional constraint

- The keyword OPTIONAL selects optional elements from the RDF graph
- complies to a Left Outer Join

SPARQL - Alternative Results via UNION

- Example: Retrieve all influencers of and people influenced by Jules Verne
- The keyword UNION allows for alternatives (logical disjunction)

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

SELECT ?influencer ?influenced
FROM <http://dbpedia.org/>
WHERE {
   { :Jules_Verne dbo:influenced ?influenced . }
  UNION
   { :Jules_Verne dbo:influencedBy ?influencer . }
}
```

logical disjunction

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

SELECT ?influencer ?influenced
FROM <http://dbpedia.org/>
WHERE {
   { :Alexander_Belyaev dbo:influenced ?influenced . }
  UNION
   { :Alexander_Belyaev dbo:influencedBy ?influencer . }
}
```

SPARQL - Negation

- Example: Retrieve authors that don't have an entry for "notable work"
- Negation in SPARQL
- complies to "NOT EXISTS" in SQL

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

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       OPTIONAL {?author dbo:notableWork ?work . }
       FILTER (!BOUND(?work)) .
} LIMIT 100
```

no variable binding

SPARQL - Negation (2)

- Example: Retrieve authors that don't have an entry for "notable work"
- SPARQL 1.1 also provides FILTER expressions NOT EXISTS and EXISTS

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

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer
        FILTER NOT EXISTS {?author dbo:notableWork ?work .}

} LIMIT 100
```

filter query result for existency

SPARQL - Negation (3)

- Example: Retrieve authors that don't have an entry for "notable work"
- Filtering of query solutions by removing possible solutions with MINUS.

- Difference to NOT EXIST:
  - MINUS changes the graph pattern
  - query result is dependent on position of MINUS

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

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer
        MINUS {?author dbo:notableWork ?work .}

} LIMIT 100
```

remove from query result

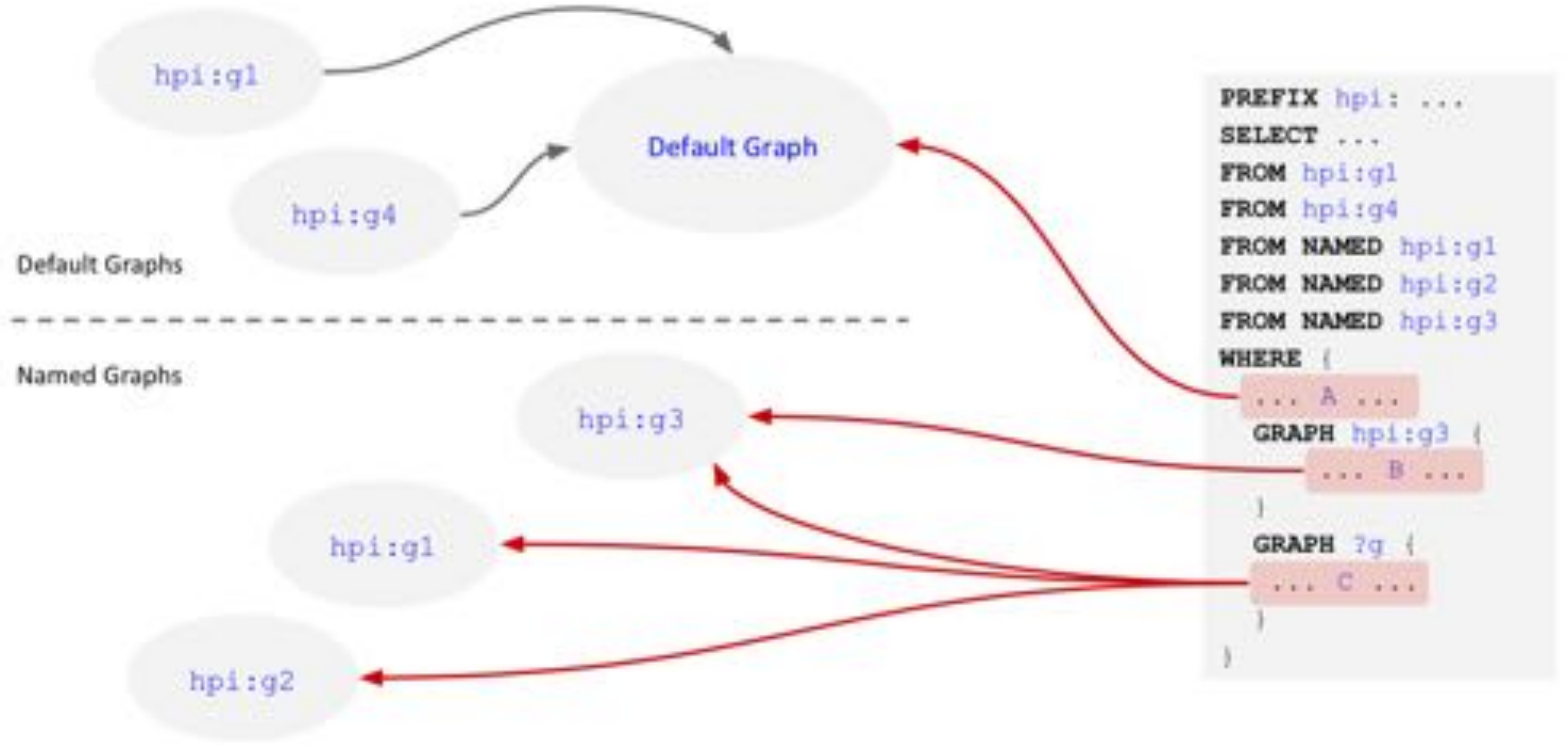
SPARQL - RDF Graphs

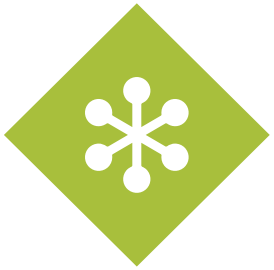
- SPARQL queries are executed over an RDF dataset
  - one (or more) default RDF graph (FROM)
  - zero or more named RDF graphs (FROM NAMED, GRAPH)

```
SELECT ...
WHERE {
...
GRAPH <http://example.org/graph1.rdf> {
      ?x foaf:mbox ?mbox
}
...
}
```

- Named Graphs can be explicitly addressed via the keyword GRAPH and the URI of the named graph

SPARQL - RDF Graphs

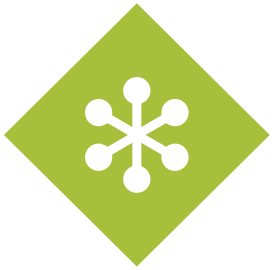




SPARQL - RDF Graphs

- How to ask a SPARQL Endpoint which RDF Graphs are available?

```
SELECT DISTINCT ?g
WHERE {
  GRAPH ?g { ?s ?p ?o . }
}
```



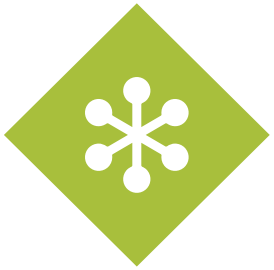
SPARQL - Federated Queries

- SPARQL enables federated queries over several RDF datasets or SPARQL endpoints via the SERVICE objective
- Example: connect the Linked Movie Database with DBpedia
  - only possible, if SPARQL endpoints permit federation

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dcterms: <http://purl.org/dc/terms/>

SELECT ?film ?label ?subject WHERE {
    SERVICE <http://data.linkedmdb.org/sparql> {
        ?film a movie:film .
        ?film rdfs:label ?label .
        ?film owl:sameAs ?dbpediaLink
        FILTER regex(STR(?dbpediaLink), "dbpedia", "i")
    }
    SERVICE <http://dbpedia.org/sparql> {
        ?dbpediaLink dcterms:subject ?subject .
    }
}
LIMIT 100
```

More Complex SPARQL Queries



SPARQL - Variable Assignments

- Example: Select all authors with their notable works and year of publication

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

SELECT ?author ?work ?date
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
       ?work dbp:releaseDate ?date
} ORDER BY ?date
LIMIT 100
```

SPARQL - Variable Assignments

- Example: Select all authors with their notable works and year of publication

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?author ?work xsd:integer(?date) AS ?year
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
       ?work dbp:releaseDate ?date
} ORDER BY DESC(?year)
LIMIT 100
```

new variable assignment

SPARQL - Variable Assignments

- Example: Select all authors with their notable works and year of publication

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?author ?work SUBSTR((REPLACE(STR(?date),"[^0-9]", "")), 1, 4) AS ?year
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
       ?work dbp:releaseDate ?date
       FILTER REGEX (?date, "[0-9]{4}") .
} ORDER BY DESC(?year)
LIMIT 100
```

new variable assignment with string manipulation and regular expression

SPARQL - Aggregate Functions

- Example: How many authors are there in DBpedia?

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

SELECT COUNT(?author) AS ?num
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
}
```

aggregate function

SPARQL - Aggregate Functions

- Example: How many distinct authors are there in DBpedia who have entries for notable works?

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

SELECT COUNT(DISTINCT(?author)) AS ?num
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer ;
                dbo:notableWork ?work .
}
```

aggregate function

SPARQL - Aggregate Functions

- Example: Which author wrote how many notable works?

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

SELECT ?author COUNT(?work) AS ?num_works
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer ;
               dbo:notableWork ?work .
} GROUP BY ?author
ORDER BY DESC (?num_works)
```

aggregate function

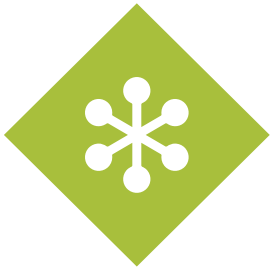
SPARQL - Aggregate Functions

- Example: Which author wrote exactly 3 notable works (according to DBpedia)?

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

SELECT ?author COUNT(?work) AS ?num_works
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer ;
                dbo:notableWork ?work .
} GROUP BY ?author
HAVING COUNT(?work) = 3
ORDER BY ?author
```

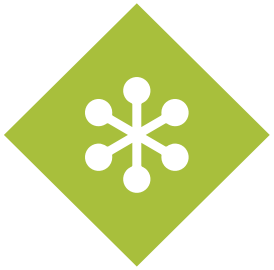
aggregate function



SPARQL - Aggregate Functions

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

SELECT ?author COUNT(?work) AS ?num_works
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer ;
               dbo:notableWork ?work .
} GROUP BY ?author
HAVING (COUNT(?work) = 3)
ORDER BY ?author
```



SPARQL - Aggregate Functions

- SPARQL 1.1 provides more aggregate functions
  - SUM
  - AVG
  - MIN
  - MAX
  - SAMPLE -- „pick“ one non-deterministically
  - GROUP_CONCAT -- concatenate values with a designated string separator

SPARQL - Aggregate Functions

- Example: Compare an arbitrary title and non-english titles of notable Works by authors

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

SELECT (SAMPLE(?title) AS ?name)
       (GROUP_CONCAT(?alt_titles; SEPARATOR=", ") AS ?alt_names)
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
       ?work rdfs:label ?title .
       ?work rdfs:label ?alt_titles
       FILTER (LANG(?alt_titles)!="en") .
} GROUP BY ?author
```

aggregate function

SPARQL Subqueries and Property Paths

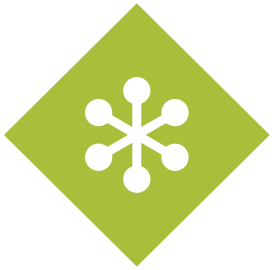
SPARQL - Subqueries

- Example: Select all authors, by whom they are influenced, and all the influencers' notable works
- Subqueries are a way to embed SPARQL Queries within other queries
- Result is achieved by first evaluating the inner query

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

SELECT ?author ?influencer ?work
FROM <http://dbpedia.org/>
WHERE {
{ SELECT ?author ?influencer
  FROM <http://dbpedia.org/>
  WHERE {
       ?author rdf:type dbo:Writer ;
               dbo:influencedBy ?influencer .
  }
}
 ?influencer dbo:notableWork ?work .
}
 LIMIT 100
```

subquery



SPARQL - Property Paths

- A **property path** is a possible route through an RDF graph between two graph nodes.
  - Trivial case: property path of length 1, i.e. a triple pattern
  - **Alternatives**: match one or both possibilities

```
{ :book1 dc:title|rdfs:label ?displayString }
```

  - **sequence:** property path of length > 1

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:knows/foaf:name ?name . }
```

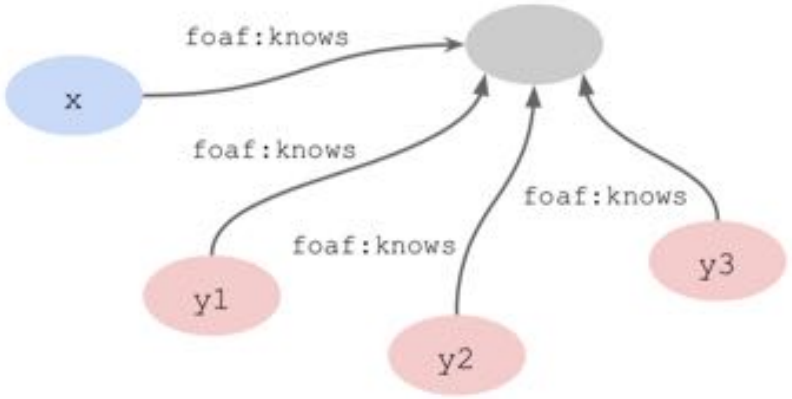
  - Inverse property paths: reversing the direction of the triple

```
{ ?x foaf:mbox <mailto:alice@example> }
                    =
{ <mailto:alice@example> ^foaf:mbox ?x }
```

SPARQL - Property Paths

- Inverse path sequences

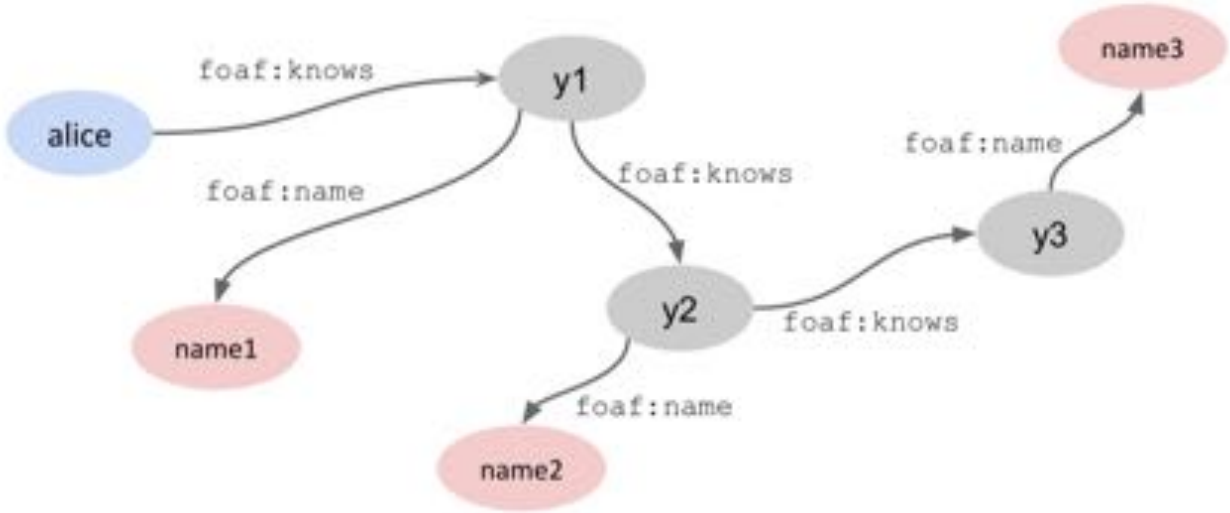
```
{ ?x foaf:knows/^foaf:knows ?y
    FILTER (?x != ?y) . }
```

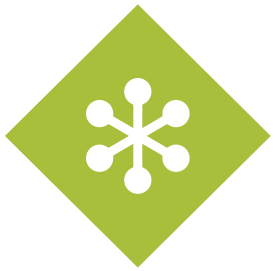


SPARQL - Property Paths

- Arbitrary length match

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name .      }
```





SPARQL - Property Paths

- Inverse path sequences

  { ?x **foaf:knows/^foaf:knows** ?y

  FILTER (?x != ?y) . }

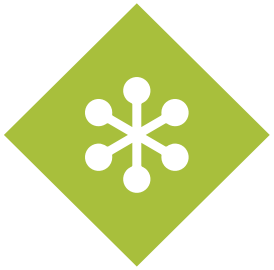
- Arbitrary length match

  { ?x foaf:mbox <mailto:alice@example> .

  ?x foaf:knows+/foaf:name ?name . }

- Negated property paths

  { ?x !(rdf:type|^rdf:type) ?y . }

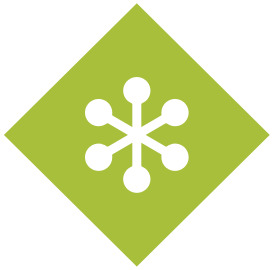


SPARQL - Property Paths

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

SELECT ?influencedByInfluencers
FROM <http://dbpedia.org/>
WHERE {
   :Jules_Verne dbo:influencedBy/^dbo:influencedBy ?influencedByInfluencers
   FILTER (?influencedByInfluencers!= :Jules_Verne).
}
```

property path expression



SPARQL - Property Paths

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

SELECT ?influencedByInfluencers
FROM <http://dbpedia.org/>
WHERE {
   :Vladimir_Nabokov dbo:influencedBy/^dbo:influencedBy ?
influencedByInfluencers
   FILTER (?influencedByInfluencers!= :Vladimir_Nabokov).
}
```