



Seminar: SPARQL Querying RDF Data

Christian Chiarcos
Goethe University Frankfurt, Germany
University of Cologne, Germany



SPARQL

- SPARQL Protocol and RDF Query Language
- W3C Recommendation
 - <http://www.w3.org/TR/rdf-sparql-query/>
 - defines syntax and semantics for querying RDF
 - independent from underlying DB implementation
 - retrieves results as variable bindings (table) or RDF triples



SPARQL

- „SQL meets Turtle“
 - extends Turtle-like triple syntax with
 - variables (marked with *?name*), and
 - specification of return values

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
    ?a a dbpedia-owl:Person .
    ?a rdfs:label ?l .
}
```



SPARQL

- PREFIX
 - namespace declaration (cf. Turtle)
- SELECT
 - specifies return values: variable binding
- WHERE
 - query
- triples
 - with variables

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
    ?a a dbpedia-owl:Person .
    ?a rdfs:label ?l .
}
```

Variables



SPARQL

- PREFIX
 - namespace declaration (cf. Turtle)
- SELECT
 - specifies return value
- WHERE
 - query
- triples
 - with variables

returns the selected variables
(additional variables can be used in WHERE)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
    ?a a dbpedia-owl:Person .
    ?a rdfs:label ?l .
}
```



Example

- data

```
ns1:Aad_Stoop    rdf:type      dbpedia-owl:Person ;
                  rdfs:label   "Ad Stoop" ,
                  .....
                  "Aad Stoop" ;
                  dcterms:provenance "The original data was retrieved from http
                  did the RDF transformation, please refer to the original site
                  skos:prefLabel "Aad Stoop" ;
                  ns1:hasId      "634034" ;
                  dc:license     <http://langtech.jrc.it/JRC-Names.html> .
```

- query

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
    ?a a dbpedia-owl:Person .
    ?a rdfs:label ?l .
}
```



Example

- data

```
ns1:Aad_Stoop    rdf:type      dbpedia-owl:Person ;
                 rdfs:label    "Ad Stoop" ,
                 .....
                 "Aad Stoop" ;
                 dct:provenance "The original data was retrieved from http
                 did the RDF transformation, please refer to the original site
                 skos:prefLabel "Aad Stoop" ;
                 ns1:hasId      "634034" ;
                 dc:license     <http://langtech.jrc.it/JRC-Names.html> .
```

- results*

?a	?l
ns1:Aad_Stoop	„Ad Stoop“
ns1:Aad_Stoop	„Aad Stoop“
...	...

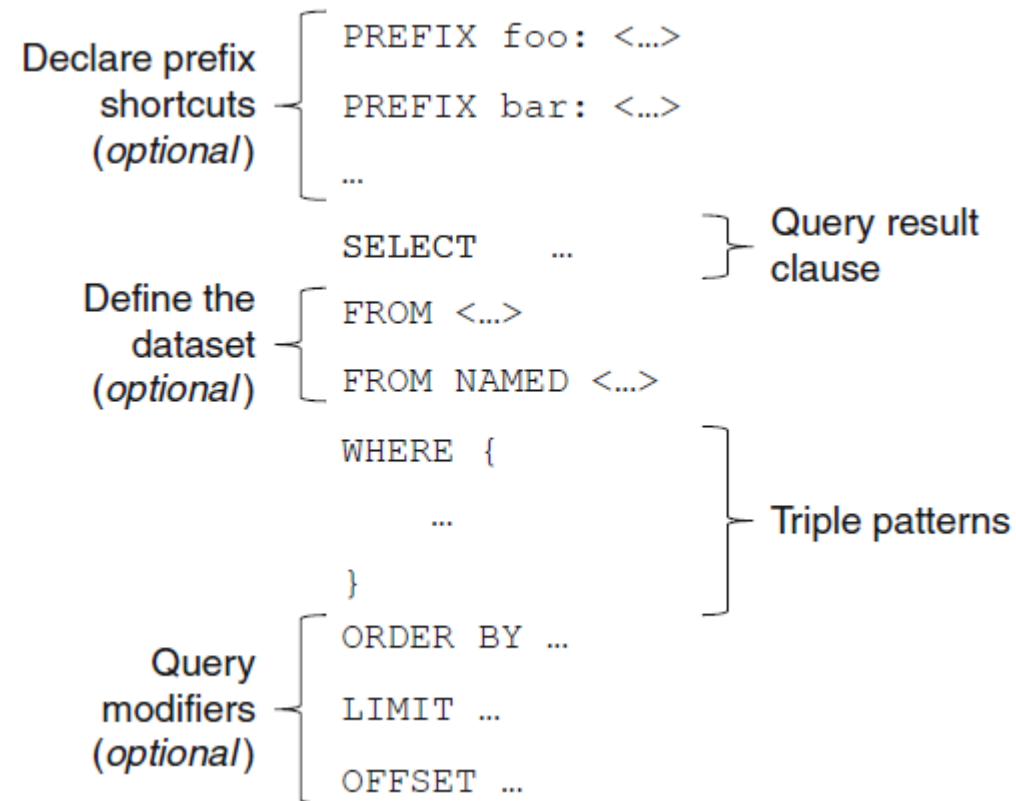
* output format can be specified, can be table, triples, html, etc.



Simple queries with filters and modifiers



General structure of SPARQL queries





SPARQL

- reasoning
 - default: RDF semantics
 - basically plain graph queries
 - external external regimes *can* be specified
 - RDFS, OWL
 - <https://www.w3.org/TR/sparql11-service-description/>
- triple notation similar to Turtle
- variables
 - start with ?



restricting a result set

- *after* a result set is initialized with a series of statements, it can be filtered
- **FILTER(...)**
 - filter conditions aren't triples, but functions over variable values

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
    ?a a dbpedia-owl:Person .
    ?a rdfs:label ?l .
    FILTER(strstarts(str(?l), "Peter"))
}
```



Functions (selection)

<i>general</i>	<i>math</i>	<i>comparison</i>	<i>string</i>
DATATYPE	ABS	=	STRLEN
STR	ROUND	<	SUBSTR
IRI	CEIL	>	UCASE
LANG	FLOOR	!=	LCASE
BOUND	+	<i>boolean</i>	STRSTARTS
IN	-	&&	STREND
NOT IN	*		STRBEFORE
isBLANK	/	!	STRAFTER
			CONCAT
			REGEX

SPARQL allows adding custom functions

e.g., strSplit (Apache Jena)

<https://jena.apache.org/documentation/query/library-propfunc.html>



typed RDF literals

- untyped strings „*Georga W Busha*“ „1“
- typed „*Georga W Busha*“^^*xsd:string* “1”^^*xsd:int*
- strings can also carry a language tag „*Georga W Busha*“@*pl*

Keep track of the type when querying!

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns1:    <http://mlope.nlp2rdf.org/resource/jrc-names/>
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
ns1:George_W_Bush  rdf:type      dbpedia-owl:Person .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
ns1:George_W_Bush  rdfs:label   "\u00B8\u00E7\u00FC\u00B8\u00FB
\u0044\u002C\u0048\u0031\u002C \u0028\u0048\u0034"@ar
\u0048\u002C\u0048\u0031\u002C \u002F\u0028\u0044\u0064
"Bush Junior" ,
"Georgem W Bushem"@sl ,
"George Walker Bushi"@et ,
"Georga W Busha"@pl ,
"\u005AC\u006BB \u0083\u0051B \u00E0\u00B8A" ,
"President Bush" ,
```

```
SELECT ?p
WHERE {
  ?p rdfs:label "Georga W Busha"
}
```

What will happen?



typed RDF literals

- untyped strings „*Georga W Busha*“ „1“
- typed „*Georga W Busha*“^^*xsd:string* “1”^^*xsd:int*
- strings can also carry a language tag „*Georga W Busha*“@*pl*

Keep track of the type when querying!

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns1:    <http://mlope.nlp2rdf.org/resource/jrc-names/>
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
ns1:George_W_Bush  rdf:type      dbpedia-owl:Person .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
ns1:George_W_Bush  rdfs:label    "\u00B8\u00E7\u00FC\u00B8\u00FB
\u0044\u002C\u0048\u0031\u002C \u0028\u0048\u0034"@ar
\u0048\u002C\u0048\u0031\u002C \u002F\u0028\u0044\u0064
"Bush Junior" ,
"Georgem W Bushem"@sl ,
"George Walker Bushi"@et ,
"Georga W Busha"@pl ,
"\u0055AC\u006CBB \u00683\u00514B \u005E03\u006B8A" ,
"President Bush" ,
```

```
SELECT ?p
WHERE {
  ?p rdfs:label "Georga W Busha"
}
```

FAILS

„Georga W. Busha“ has a language tag



typed RDF literals

- untyped strings „*Georga W Busha*“ „1“
- typed „*Georga W Busha*“^{^^xsd:string} “1”^{^^xsd:int}
- strings can also carry a language tag „*Georga W Busha*“@pl

Keep track of the type when querying!

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns1:    <http://mlope.nlp2rdf.org/resource/jrc-names/>
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
ns1:George_W_Bush  rdf:type      dbpedia-owl:Person .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
ns1:George_W_Bush  rdfs:label    "\u00B8\u00E7\u00FC\u00B8\u00FB
\u0044\u002C\u0048\u0031\u002C \u0028\u0048\u0034"@ar
\u0048\u002C\u0048\u0031\u002C \u002F\u0028\u0044\u0064
"Bush Junior" ,
"Georgem W Bushem"@sl ,
"George Walker Bushi"@et ,
"Georga W Busha"@pl ,
"\u0055\u00AC\u00BB \u0083\u0051\u004B \u00E0\u00B8A" ,
"President Bush" ,
```

```
SELECT ?p
WHERE {
  ?p rdfs:label "Georga W Busha"@pl
}
```

OK
return „Georga W Busha“ as possible result



type-free matches with FILTER

```
SELECT ?p
WHERE {
  ?p rdfs:label ?label
  FILTER(str(?label) = "Georga W Busha")
}
```



type-free matches with FILTER

```
SELECT ?p
WHERE {
  ?p rdfs:label ?label
  FILTER(str(?label) = "Georga W Busha")
}
```

... or with BIND

```
SELECT ?p
WHERE {
  ?p rdfs:label ?label.
  BIND(str(?label) as ?plainLabel)
  FILTER(?plainlabel = "Georga W Busha")
}
```



modifiers: ORDER BY

- sort the results of SELECT

```
SELECT ?p
WHERE {
  ?p a dbpedia-owl:Person.
  ?p rdfs:label ?l
}
ORDER BY ?l
```

p
http://mlode.nlp2rdf.org/resource/jrc-names/A_Flod
http://mlode.nlp2rdf.org/resource/jrc-names/Abdul_Kalam
http://mlode.nlp2rdf.org/resource/jrc-names/A_Petersen
http://mlode.nlp2rdf.org/resource/jrc-names/A_Thorbjørnsen
http://mlode.nlp2rdf.org/resource/jrc-names/Abdul_Kalam
http://mlode.nlp2rdf.org/resource/jrc-names/Aabid_Hussain_Khan
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_de_Mos
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Goudriaan



modifiers: LIMIT and OFFSET

- LIMIT n : return max n matches
- OFFSET m : skip the first n matches
 - combine with ORDER to ensure constant order of return values

```
SELECT ?p
WHERE {
  ?p a dbpedia-owl:Person.
  ?p rdfs:label ?l
} ORDER BY ?l
LIMIT 5
OFFSET 5
```

p
http://mlode.nlp2rdf.org/resource/jrc-names/Aabid_Hussain_Khan
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_de_Mos
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Goudriaan
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Jacobs
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Koster



Property Paths

instead of writing triple by triple, create paths of properties



SPARQL 1.1 Property Paths

- Instead of writing all statements individually, you can define sequences of properties that connect two variables

?entry a ontalex:LexicalEntry.

?entry ontalex:sense ?sense.

?entry ontalex:lexicalForm ?form.

?form ontalex:writtenRep ?string.

as property path:

*?entry **ontalex:lexicalForm/ontalex:writtenRep** ?string.*

X/Y (sequence): object of the first property X is the subject of the next Y



SPARQL 1.1 Property Paths

- Instead of writing all statements individually, you can define sequences of properties that connect two variables

?entry a ontalex:LexicalEntry.

?entry ontalex:sense ?sense.

?entry ontalex:lexicalForm ?form.

?form ontalex:writtenRep ?string.

as property path:

*?entry **ontalex:lexicalForm/ontalex:writtenRep** ?string.*

X/Y (sequence): object of the first property X is the subject of the next Y

X|Y (alternative): predicate is either X or Y

X* (iteration): any sequence of zero to *n* iterations of X

^X (reverse): switch subject and object

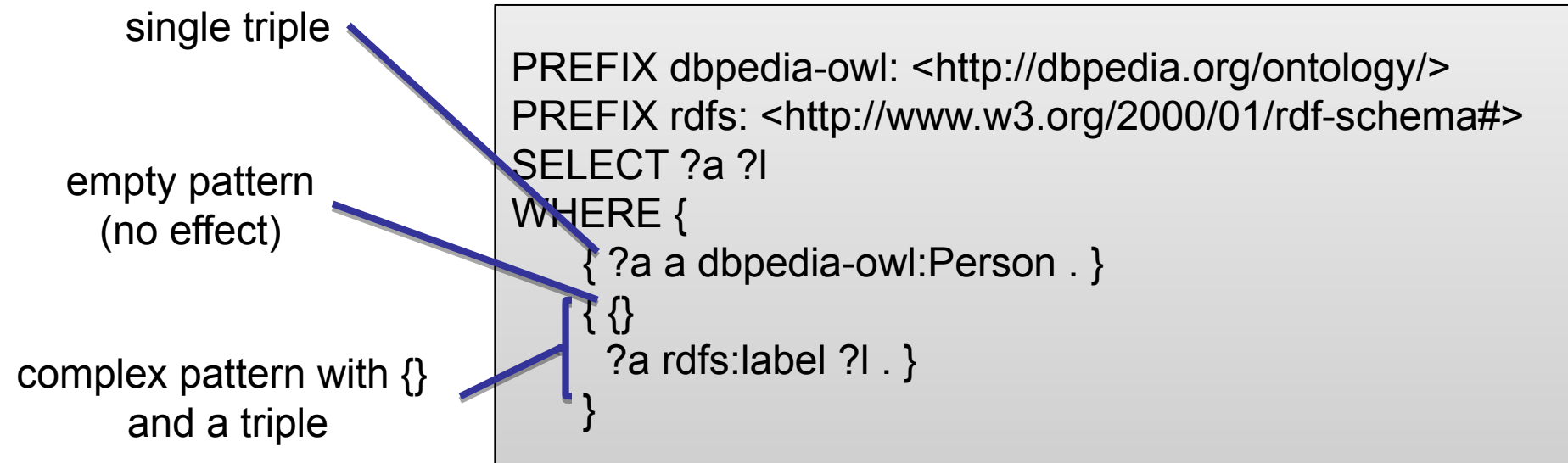


Graph patterns and logical operators



graph patterns

- complex graph patterns
 - in { ... }
 - without key words, these just serve to group together statements





OPTIONAL { ... }

- *optional* statements in parentheses
 - *bottom-up evaluation*
 - establish a result set for the optional graph pattern
 - if not an empty result, then
 - *integrate with context*
 - if a result is found, try join with result set
 - if no result is found *or* the join fails, some variables may remain unbound



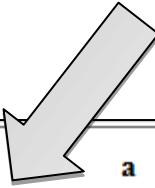
OPTIONAL { ... }

```

PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
    ?a a dbpedia-owl:Person .
    OPTIONAL {
        ?a rdfs:label ?l.
        FILTER regex(?l, '^G.* Bush$') .
    }
}

```

(optional)
graph pattern:
tripel
FILTER



a	l
http://mlode.nlp2rdf.org/resource/jrc-names/A_Thorbjørnsen	
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Meijboom	
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Stoop	
http://mlode.nlp2rdf.org/resource/jrc-names/Aad_Taal	
http://mlode.nlp2rdf.org/resource/jrc-names/Aage_Borchgrevink	



{ ... } UNION { ... }

- logical disjunction („or“)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
  { ?a a dbpedia-owl:Person . } UNION
  { ?a rdfs:label ?l. }
}
```

- both conjuncts are processed independently,
then they are joined into a single result set
 - if both legs return the same variable binding, it is returned twice



MINUS { ... }

~ logical negation („and not“)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
  ?a a dbpedia-owl:Person .
  ?a rdfs:label ?l.
  MINUS {
    FILTER(?l = "Homer Simpson"@en)
  }
}
```

WHAT WILL HAPPEN?



MINUS { ... }

~ logical negation („and not“)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
    ?a a dbpedia-owl:Person .
    ?a rdfs:label ?l.
    MINUS {
        FILTER(?l = "Homer Simpson"@en)
    }
}
```

This works on OpenLink Virtuoso (<https://dbpedia.org/sparql>)
On other end points, this may fail, because no result set is established
in the MINUS clause when the filter is applied



MINUS { ... }

~ logical negation („and not“)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?a ?l
WHERE {
  ?a a dbpedia-owl:Person .
  ?a rdfs:label ?l.
  MINUS {
    ?a rdfs:label ?l.
    FILTER(?l = "Homer Simpson"@en)
  }
}
```

This should work everywhere.



Next Steps



Next Steps

- Hands-on (May 31)
 - 11:30: SPARQL (with dictionary data)
 - *DBnary*
 - *Linking Latin*
 - 14:30: LLOD generation
 - *Fintan*
- June 1 onwards: more applications ;)



Links

- <http://www.sparql.org/>
 - links to authoritative information
 - Online validator and processor
 - query public data without a local endpoint ;)
 - loads data from FROM clause
- If you prefer prose
 - <https://en.wikibooks.org/wiki/SPARQL> is quite usable



Advanced: Return values for queries

Think of these slides as a reference and consult if needed ;)



Return values for queries

- **SELECT**
 - table of variable bindings
 - SELECT * for all variables
- **ASK**
 - return a boolean (true, false)
- **CONSTRUCT**
 - return RDF triples
- **DESCRIBE**
 - return RDF tripels „about“ the variables in the returned matches*

* implementation-specific

SELECT

```
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns1:      <http://mlode.nlp2rdf.org/resource/jrc-names/> .
@prefix dbpedia-owl:  <http://dbpedia.org/ontology/> .
ns1:George_W_Bush   rdf:type      dbpedia-owl:Person .
@prefix rdfs:      <http://www.w3.org/2000/01/rdf-schema#> .
ns1:George_W_Bush   rdfs:label    "\u30B8\u30E7\u30FC\u30B8\u30FB\u30A6\u30F3
    "\u0644\u062C\u0648\u0631\u062C \u0628\u0648\u0634\u0627"@ar ,
    "\u0648\u062C\u0648\u0631\u062C \u0628\u0628\u0648\u0628 \u0628\u0648\u0627"@pl ,
    "Bush Junior" ,
    "Georgem W Bushem"@sl ,
    "George Walker Bushi"@et ,
    "Georgia W Busha"@pl ,
    "\u55AC\u6CBB \u6C83\u514B \u5E03\u6B8A" ,
    "President Bush" ,
```

SELECT DISTINCT

- remove all duplicats

SELECT REDUCED

- remove most („best-effort“) duplicats

```
SELECT ?p
WHERE {
    ?p rdfs:label "President Bush".
}
```

P
http://mlode.nlp2rdf.org/resource/jrc-names/George_W_Bush



ASK

```
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns1:      <http://mlode.nlp2rdf.org/resource/jrc-names/> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
ns1:George_W_Bush  rdf:type      dbpedia-owl:Person .
@prefix rdfs:      <http://www.w3.org/2000/01/rdf-schema#> .
ns1:George_W_Bush  rdfs:label    "\u30B8\u30E7\u30FC\u30B8\u30FB\u30A6\u30F3
    "\u0644\u062C\u0648\u0631\u062C \u0628\u0648\u0634\u0648\u0634\u0648"@ar ,
    "\u0648\u062C\u0648\u0631\u062C \u0628\u0648\u0628\u0648\u0644\u0648\u0648\u0648" ,
    "Bush Junior" ,
    "Georgem W Bushem"@sl ,
    "George Walker Bushi"@et ,
    "Georgia W Busha"@pl ,
    "\u55AC\u6CBB \u6C83\u514B \u5E03\u688A" ,
    "President Bush" ,
```

can be used to test whether
some information is provided
by an end point

```
ASK
WHERE {
    ?p rdfs:label "President Bush".
}
```

true



CONSTRUCT

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns1:    <http://mlode.nlp2rdf.org/resource/jrc-names/> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
ns1:George_W_Bush rdf:type    dbpedia-owl:Person .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
ns1:George_W_Bush rdfs:label  "\u00B8\u00E7\u00FC\u00B8\u00FB\u00A6\u00E4\u0062C\u00648\u00631\u0062C \u00628\u00648\u00634"@ar ,
    "\u00648\u0062C\u00648\u00631\u0062C \u0062F\u00628\u00644\u0064A\u00648 \u0062C\u00648\u00631\u0062C \u0062F\u00628\u00644\u0064A\u00648" ,
    "Bush Junior" ,
    "Georgem W Bushem"@sl ,
    "George Walker Bushi"@et ,
    "Georga W Busha"@pl ,
    "\u0055AC\u006CBB \u006C83\u00514B \u005E03\u006B8A" ,
    "President Bush" ,
```

return RDF triples
can be used for pulling
triples out of an end point

```
CONSTRUCT {
  ?p owl:sameAs
    <http://my-db.org/Bush>.
  ?p rdfs:comment "durch Label-Match
    gefunden"@de
}
WHERE {
  ?p rdfs:label "President Bush".
}
```

graph pattern defines output

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ns1:    <http://mlode.nlp2rdf.org/resource/jrc-names/> .
ns1:George_W_Bush rdfs:comment  "durch Label-Match gefunden"@de .
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix ns3:    <http://my-db.org/> .
ns1:George_W_Bush owl:sameAs ns3:Bush .
```

DESCRIBE

```
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns1:     <http://mlode.nlp2rdf.org/resource/jrc-names/> .
@prefix dbpedia-owl:    <http://dbpedia.org/ontology/> .
ns1:George_W_Bush   rdf:type       dbpedia-owl:Person .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
ns1:George_W_Bush   rdfs:label    "\u30B8\u30E7\u30FC\u30B8\u30FB\u30A6\u30F3
        \"\u0644\u062C\u0648\u0631\u062C \u0628\u0648\u0634\"@ar ,
        \"\u0648\u062C\u0648\u0631\u062C \u0628\u0648\u0644 \u0628\u0648\u0634\"@pl ,
        "Bush Junior" ,
        "Georgem W Bushem"@sl ,
        "George Walker Bushi"@et ,
        "Georgia W Busha"@pl ,
        "\u55AC\u6CB8 \u6C83\u514B \u5E03\u6B8A" ,
        "President Bush" ,
```

return context (RDF triples)

can be used for exploring
an end point

```
DESCRIBE ?p
WHERE {
    ?p rdfs:label "President Bush"@en.
}
```

Used here to retrieve the fragment above from DBpedia.

<https://dbpedia.org/sparql>



Advanced: FROM and SERVICE

Think of these slides as a reference and consult if needed ;)



Named Graphs

- data can be split in different subsets
 - e.g., source documents
 - identified by an URI => *GRAPH*
 - if no graph is defined, use the default (unnamed) graph
 - e.g., merged graph of all imported documents



Named Graphs: FROM

- FROM <URI>
 - set default graph to the graph identified by the URI

```
SELECT ?p
FROM <http://thedatahub.org/dataset/jrc-
names>
WHERE {
    ?p a dbpedia-owl:Person.
    ?p rdfs:label ?l
}
ORDER BY ?l LIMIT 5 OFFSET 5
```



Named Graphs: FROM NAMED

- instead of operating within a single graph, we can also access multiple graphs independently
 - FROM NAMED => graph data will become available, but not added to default graph



Named Graphs: GRAPH

Use *GRAPH* to access a named graph within a WHERE block

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
{
  GRAPH ?src
  { ?x foaf:mbox <mailto:bob@work.example> .
    ?x foaf:nick ?bobNick
  }
}
```

src	bobNick
<http://example.org/foaf/aliceFoaf>	"Bobby"
<http://example.org/foaf/bobFoaf>	"Robert"



Federation: SERVICE

- Similar to accessing a named graph in your DB with *GRAPH*, you can access a remote end point with *SERVICE*

The screenshot shows a web browser window with the address bar at www.sparql.org/sparql.html. The browser's toolbar includes various icons and links such as 'uni', 'DFG', 'dfg-form', 'NIF 2.0 Core Ontology', 'URL', 'FAMPLAN', 'T-AP', and 'LIDER-RefC'. Below the toolbar, a light blue header bar is present. The main content area contains the text 'General SPARQL query : input query, set any options and press "Get Results"'. A text input field contains the following SPARQL query:

```
SELECT *  
WHERE {  
  SERVICE <https://dbpedia.org/sparql> {  
    SELECT *  
    WHERE {  
      ?a ?b ?c } LIMIT 10  
    }  
  }  
}
```



Advanced: SPARQL Update

Think of these slides as a reference and consult if needed ;)



INSERT DATA

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
INSERT DATA {
```

```
<http://example/book1> dc:title "A new book";
```

```
dc:creator "A.N.Other" .
```

```
}
```

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
```

```
@prefix ns: <http://example.org/ns#> .
```

```
<http://example/book1> ns:price 42 .
```

```
<http://example/book1> dc:title "A new book" . <http://example/book1> dc:creator  
"A.N.Other" .
```

- you can also define a *GRAPH*
INSERT DATA { GRAPH <g> { ... } }
- otherwise, use the default graph
- no variables !



DELETE DATA

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
DELETE DATA {
```

```
<http://example/book1> dc:title "A new book";
```

```
dc:creator "A.N.Other" .
```

```
}
```

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
```

```
@prefix ns: <http://example.org/ns#> .
```

```
<http://example/book1> ns:price 42 .
```

```
<http://example/book1> dc:title "A new book". <http://example/book1> dc:creator  
"A.N.Other".
```

- analogous to INSERT DATA
also

```
DELETE DATA { GRAPH <g> { ... } }
```



DELETE/INSERT + WHERE

- add or remove triples in accordance with conditions

WITH *graphIRI*

DELETE { *Triples* }

INSERT { *Triples* }

USING [NAMED] *graphIRI*

WHERE { ... }

(optional) restrict to one named graph

DELETE and INSERT can be
combined *in that order*

restrict to one or
multiple named graphs

conventional WHERE block



DELETE/INSERT + WHERE

- add or remove triples in accordance with conditions

WITH *graphIRI*

DELETE { *Triples* }

INSERT { *Triples* }

USING [NAMED] *graphIRI*

WHERE { ... }

Rename every „Bill“
to „William“

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

WITH <http://example/addresses>
DELETE { ?person foaf:givenName 'Bill' }
INSERT { ?person foaf:givenName 'William' }
WHERE
  { ?person foaf:givenName 'Bill'
  }
```



mass operations

load / delete all triples (of a named graph)

LOAD *uri* [INTO GRAPH *uri*]

CLEAR (GRAPH *uri* | DEFAULT | NAMED | ALL)

Use LOAD to pull local *or remote*
data into your end point



managing graphs

- The CREATE operation creates a new graph in stores that support empty graphs.
- The DROP operation removes a graph and all of its contents.
- The COPY operation modifies a graph to contain a copy of another.
- The MOVE operation moves all of the data from one graph into another.
- The ADD operation reproduces all data from one graph into another.

CREATE GRAPH *graphIRI*

create an empty graph

DROP (GRAPH *uri* | DEFAULT | NAMED | ALL)

remove a graph and its content

DROP DEFAULT: clear default graph



Appendix:

More advanced stuff

Aggregates



Aggregate functions in SELECT

- SELECT can not just return variable bindings, but also assign new variables to aggregates

```
SELECT (COUNT(*) AS ?triples)  
WHERE { ?a ?b ?c }
```

- COUNT: count results
- SUM, AVG, MIN, MAX: math
- GROUP_CONCAT: concatenation



Aggregate functions in SELECT

- SELECT can not just return variable bindings, but also assign new variables to aggregates

```
SELECT (COUNT(*) AS ?triples)  
WHERE { ?a ?b ?c }
```

- COUNT: count results
- SUM, AVG, MIN, MAX: math
- GROUP_CONCAT: concatenation



Aggregate functions in SELECT

- If an aggregate is required as part of the query, you can call SELECT in a WHERE block
 - combine with GROUP_BY

```

SELECT ?a ?b
WHERE {
  ?a ?b ?maxC.
  { SELECT ?a (MAX(?c) AS ?maxC)
    WHERE { ?a ?tmp ?c }
  } GROUP BY ?a
}

```

for every ?a:
which property ?b
has the highest value?

only the SELECT variables
(?a, ?maxC) are unified with
context variables, local variables
can have the same name



Have fun!

← → ↺ ⚠ Nicht sicher | sparql.org/sparql.html

SPARQLer - General purpose processor

General SPARQL query : input query, set any options and press "Get Results"

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ontolox: <http://www.w3.org/ns/lemon/ontolox#>
PREFIX vartrans: <http://www.w3.org/ns/lemon/vartrans#>
SELECT DISTINCT ?translation
WHERE {
  SERVICE <http://kaiko.getalp.org//sparql?default-graph-uri> {
    ?entry fun ontolox:canonicalForm/ontolox:writtenRep "fun"@en.
    ?t (vartrans:translatableAs|^vartrans:translatableAs) ?entry fun.
    ?t ontolox:canonicalForm/ontolox:writtenRep ?translation.
  }
  BIND(lang(?translation) as ?lang)
} ORDER BY ?lang ?translation LIMIT 20
```

Target graph URI (or use FROM in the query)

If no dataset is provided, the query will execute against an empty one.

The query can contain use VALUES to set some variables.

Output:

XSLT style sheet (blank for none):

☐ Force the accept header to text/plain regardless

SPARQLer Query Results

translation
"веселие" @bg
"забава" @bg
"смешен" @bg
"Amusement" @de
"Gaudi" @de
"Spaß" @de
"Vergnügen" @de
"lustig" @de
"spaßig" @de
"verlustieren" @de
"κέφι" @el
"diversión" @es
"divertido" @es
"embullarse" @es
"gracia" @es
"placer" @es
"hauska" @fi
"hauskanpito" @fi
"hauskuus" @fi
"huvi" @fi



Acknowledgments

The original version of these slides has been created for EUROLAN-2021, Romania (online), Feb 2021 by Christian Chiarcos, Applied Computational Linguistics, GU Frankfurt, Germany

- ~ Thanks to tutors and participants of EUROLAN-2021 for feedback and corrections
- ~ Use with Attribution to author, EUROLAN-2021 and SD-LLOD 2022
- ~ Licensed as [CC BY-NC-SA 3.0 Unported](https://creativecommons.org/licenses/by-nc-sa/3.0/)