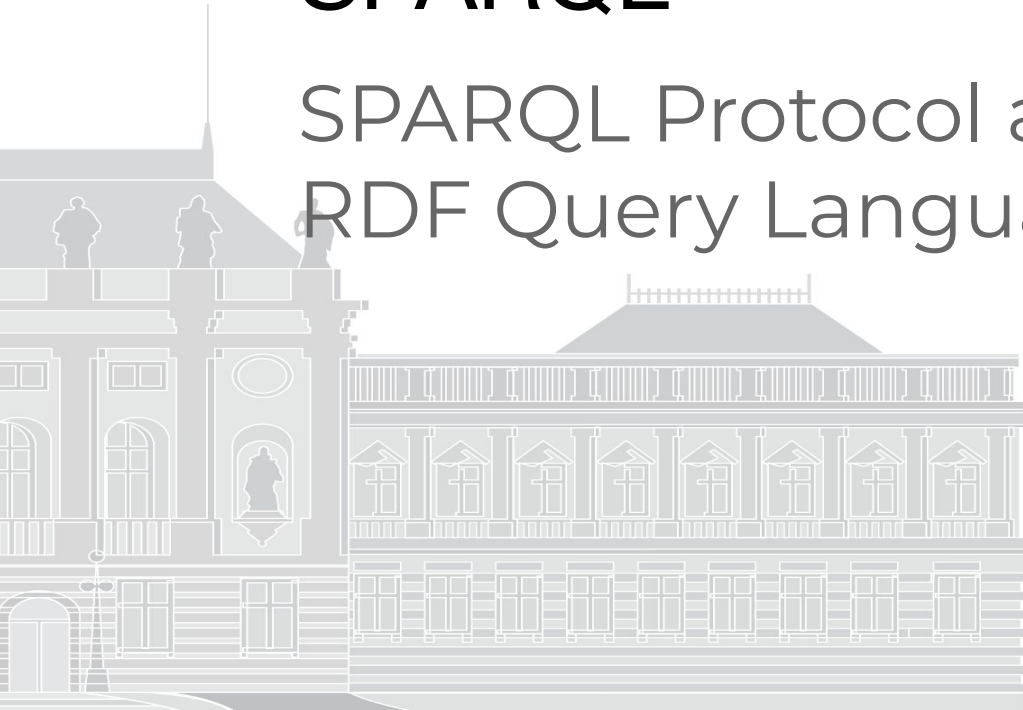


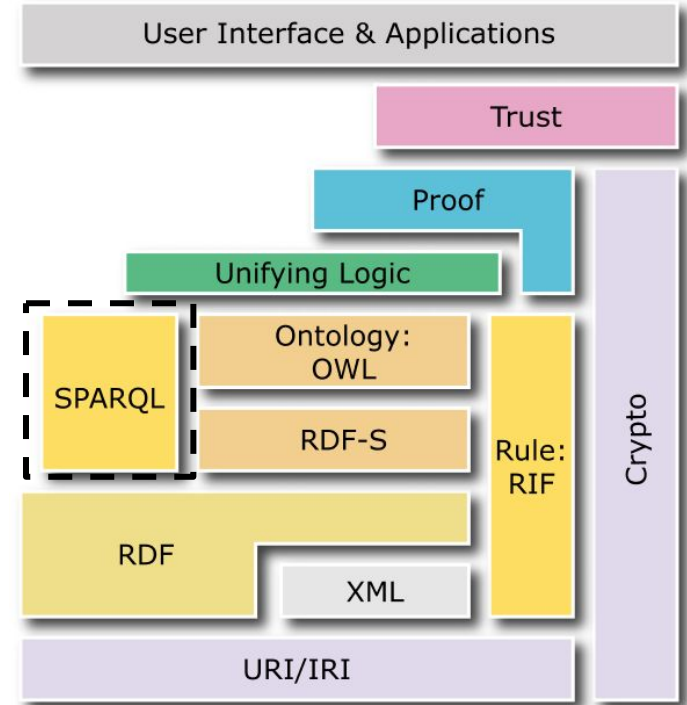
SPARQL

SPARQL Protocol and RDF Query Language



SPARQL Protocol and RDF Query Language 1.1 ist ...

- ... in RDF Turtle serialisiert
- ... eine Abfragesprache für “*RDF graph traversal*”
- ... ein spezifisches Protokoll das über HTTP läuft:
Client vs. Server (SPARQL-Endpoints)
- ... ein Spezifikation für ein XML Output Format
- ... ein W3C Standard
- ... “wie SQL” und davon inspiriert



Was wir bisher wissen...

- Graphs / Namespaces / Serialisation

object-property

data-property

Subject

Property

Object

emp3

title

"Vice President"

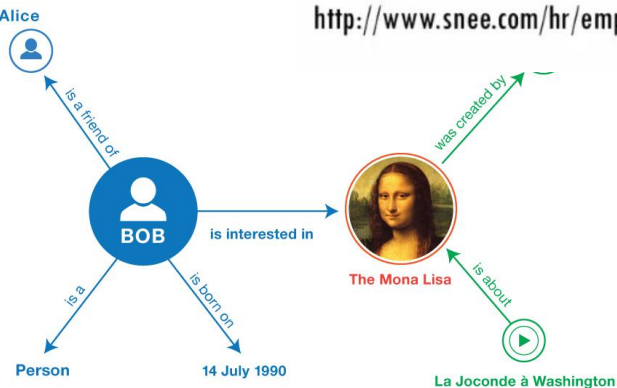
Literal

<http://www.snee.com/hr/emp3>

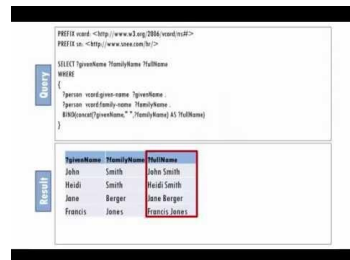
<http://www.w3.org/2006/vcard/ns#title>

"Vice President".

URI



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



SPARQL in 11 minutes

Einführung in SPARQL

Graph Pattern

PREFIX dbp:<http://dbpedia.org/property/>

```
SELECT ?hero ?status
WHERE {

    ?hero dbp:hero ?status.

}
```

SPARQL basiert auf RDF Turtle Serialisierung und auf “*basic graph pattern matching*”

Ein ***triple pattern*** ist ein RDF-Triple, das Variablen beinhaltet, wie:

?country dbo:capital ?capital

Ein Basic ***graph pattern*** ist eine Menge von *triple patterns*.

→ SPARQL gibt uns also Ergebnisse auf der Basis, ob ein bestimmtes Graphmuster in den Daten gefunden wird.

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

specifies namespaces

```
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 ——— specifies output variables
```

```
FROM <http://dbpedia.org/> ——— specifies graph to be queried
```

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

*specifies graph pattern
to be matched*

**Konjunktive
Verknüpfung
von graph
patterns**

DISTINCT / LIMIT / OFFSET

SELECT *

#SELECT DISTINCT *

WHERE {

?s ?p ?o .

}

OFFSET 100

LIMIT 200

* ... Alles wird ausgegeben

?s ?p ?o ... Variablen

LIMIT ...

Damit wird das Ergebnis auf eine Anzahl von Treffern limitiert. Gut zum “ausprobieren”.

OFFSET ...

nicht das erste, sondern ab dem ‘100. Treffer’

... Kommentar

DISTINCT ... alle Duplikate rauswerfen

Übung

Wie könnte eine Query aussehen, damit wir einen Überblick über alle Properties in unserer Datenbank bekommen?

Oder Ressourcen, die mit einer `rdfs:label` Property verknüpft sind.

Verwende dabei **DISTINCT**.

Übung

Wie könnte eine Query aussehen, damit wir einen Überblick über alle Properties in unserer Datenbank bekommen?

Oder Ressourcen, die mit einer `rdfs:label` Property verknüpft sind.

Verwende dabei **DISTINCT**.

```
SELECT DISTINCT ?property
WHERE {
    ?a ?property ?b.
}
```

```
SELECT DISTINCT ?property ?b
WHERE {
    ?a ?property ?b.
}
```

```
SELECT DISTINCT *
WHERE {
    ?entity rdfs:label ?name .
}
```

Namespaces / Discover with rdfs:label

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT DISTINCT *
```

```
WHERE {  
    ?entity rdfs:label ?name .  
}
```

```
LIMIT 200
```

[WIKIDATA - Query](#)

```
PREFIX rdfs: ...
```

Wir definieren einen Namespace “rdfs”, hinter dem die URI für das RDF Schema steckt. Wir können beliebige Namespaces definieren.

```
rdfs:label ...
```

Ist die Property “label” aus dem RDFS-Datenschema.

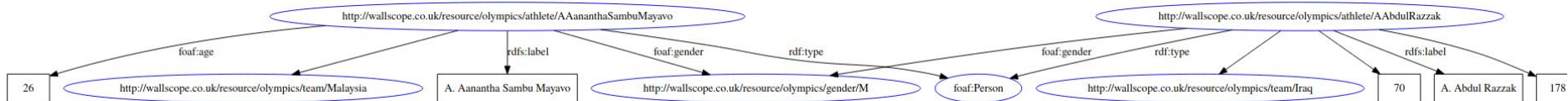
Wir fragen alle Triples ab, bei denen eine Ressource (?entity) mit der Property rdfs:label mit einer andere Ressource (?name) verknüpft ist.

Olympics RDF-Dataset

```
@prefix ns0:    <http://wallscope.co.uk/resource/olympics/team/> .
@prefix ns1:    <http://dbpedia.org/ontology/> .
@prefix ns2:    <http://wallscope.co.uk/resource/olympics/athlete/> .
@prefix ns3:    <http://xmlns.com/foaf/0.1/> .
@prefix ns4:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ns5:    <http://www.w3.org/2001/XMLSchema#> .
@prefix ns6:    <http://wallscope.co.uk/resource/olympics/gender/> .
ns2:AAananthaSambuMayavo ns1:team ns0:Malaysia .
ns2:AAananthaSambuMayavo a ns3:Person .
ns2:AAananthaSambuMayavo ns4:label "A. Aanantha Sambu Mayavo"@en .
ns2:AAananthaSambuMayavo ns3:age "26"^^ns5:int .
ns2:AAananthaSambuMayavo ns3:gender ns6:M .
ns2:AAbdulRazzak ns1:height 29;
ns1:team ns0:Iraq ;
ns1:weight 70 ;
a ns3:Person ;
ns4:label "A. Abdul Razzak"@en ;
ns3:gender ns6:M .
```

Mit <http://www.easyrdf.org/converter>
kann man RDF in andere Serialisation
transformieren.

So auch in das graphviz Format,
welches man dann auf
<https://dreampuf.github.io/GraphvizOnline>
visualisieren kann.



Übung

Schreibe eine Query, die
für die Athleten, die
Namen, das Team und
das Alter ausgibt.

```
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
prefix ns0: <http://wallscope.co.uk/resource/olympics/team/>  
prefix ns1: <http://dbpedia.org/ontology/>
```

```
prefix ns2: <http://wallscope.co.uk/resource/olympics/athlete/>  
prefix ns3: <http://xmlns.com/foaf/0.1/>  
prefix ns4: <http://www.w3.org/2000/01/rdf-schema#>  
prefix ns5: <http://www.w3.org/2001/XMLSchema#>  
prefix ns6: <http://wallscope.co.uk/resource/olympics/gender/>
```

```
SELECT  
WHERE  
{  
  
}
```

Template

Übung

Schreibe eine Query, die für die Athleten, die Namen, das Team und das Alter ausgibt.

```
Prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix ns0: <http://wallscope.co.uk/resource/olympics/team/>
prefix ns1: <http://dbpedia.org/ontology/>
prefix ns2: <http://wallscope.co.uk/resource/olympics/athlete/>
prefix ns3: <http://xmlns.com/foaf/0.1/>
prefix ns4: <http://www.w3.org/2000/01/rdf-schema#>
prefix ns5: <http://www.w3.org/2001/XMLSchema#>
prefix ns6: <http://wallscope.co.uk/resource/olympics/gender/>
```

```
SELECT ?athlete ?name ?team ?age
WHERE
{
    ?athlete a ns3:Person.
    #?athlete rdf:type ns3:Person.
    ?athlete ns4:label ?name.
    ?athlete ns1:team ?team.
    ?athlete ns3:age ?age.
}
```

*Alle Triples sind **konjunktive verknüpft**:
Athleten, die kein 'ns3:age' haben,
werden damit nicht gefunden.*

OPTIONAL

```
prefix ns0: <http://wallscope.co.uk/resource/olympics/team/>
prefix ns1: <http://dbpedia.org/ontology/>
prefix ns2: <http://wallscope.co.uk/resource/olympics/athlete/>
prefix ns3: <http://xmlns.com/foaf/0.1/>
prefix ns4: <http://www.w3.org/2000/01/rdf-schema#>
prefix ns5: <http://www.w3.org/2001/XMLSchema#>
prefix ns6: <http://wallscope.co.uk/resource/olympics/gender/>
```

```
SELECT ?athlete ?name ?team ?age
```

```
WHERE
```

```
{
```

```
  ?athlete a ns3:Person.
```

```
  ?athlete ns4:label ?name.
```

```
  ?athlete ns1:team ?team.
```

```
  OPTIONAL{?athlete ns3:age ?age.}
```

```
}
```

OPTIONAL {} ...

dient zur disjunktiven Verknüpfung:
es kann oder kann keine Verbindung
bestehen (wie ein *outer join* in SQL).

OPTIONAL

Data is here:

```
# filename: ex054.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
SELECT ?first ?last ?workTel ?nick
WHERE {
  ?s ab:firstName ?first ;
     ab:lastName ?last .
  OPTIONAL {
    ?s ab:workTel ?workTel ;
       ab:nick ?nick .
  }
}
```

```
# filename: ex059.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
```

```
SELECT ?first ?last ?workTel ?nick
WHERE
{
  ?s ab:firstName ?first ;
     ab:lastName ?last .
  OPTIONAL {
    ?s ab:workTel ?workTel ;
       ab:nick ?nick .
  }
}
```

first	last	workTel	nick
Richard	Mutt		
Craig	Ellis		
Cindy	Marshall		

```
# filename: ex061.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
```

```
SELECT ?first ?last ?workTel ?nick
WHERE
{
  ?s ab:firstName ?first ;
     ab:lastName ?last .
  OPTIONAL { ?s ab:workTel ?workTel . }
  OPTIONAL { ?s ab:nick ?nick . }
}
```

first	last	workTel	nick
Richard	Mutt		
Craig	Ellis	(245) 315-5486	
Cindy	Marshall		

Jedes **OPTIONAL** {} definiert wieder ein graph pattern: darin wird mit “UND verknüpft”

OPTIONAL : *order of the graph pattern matters*

```
# filename: ex059.rq
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
```

```
SELECT ?first ?last
WHERE
{
  ?s ab:lastName ?last .
    OPTIONAL { ?s ab:nick ?first . }
    OPTIONAL { ?s ab:firstName ?first . }
}
```

first	last
Richy300	Mutt
Chrisi	Pollin
Craig	Ellis
Cindy	Marshall

Alle Ressourcen mit einem ab:lastName.

Gibt es ein ab:nick wird das in ?first gebunden, wenn nicht,

dann wird ab:firstName in ?first gebunden, wenn es denn existiert.

“When querying large datasets, overuse of OPTIONAL graph patterns can slow down your queries”!

Übung

Was machen
die 3
Queries?

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

```
SELECT ?athlete
```

```
WHERE{?athlete rdfs:label "Ali Bourai"@en}
```

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

```
prefix ns3: <http://xmlns.com/foaf/0.1/>
```

```
prefix ns5: <http://www.w3.org/2001/XMLSchema#>
```

```
SELECT ?athlete
```

```
WHERE{?athlete ns3:age "29"^^ns5:int.}
```

```
PREFIX walls: <http://wallscope.co.uk/ontology/olympics/>
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT ?instance ?medal ?name
```

```
WHERE
```

```
{
```

```
    ?instance walls:athlete <http://wallscope.co.uk/resource/olympics/athlete/LidaPeytonElizaPollockMcMillen> ;  
    walls:medal ?medal .
```

```
    <http://wallscope.co.uk/resource/olympics/athlete/LidaPeytonElizaPollockMcMillen>  
    rdfs:label ?name.
```

```
}
```

FILTER

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix ns1: <http://dbpedia.org/ontology/>
prefix ns3: <http://xmlns.com/foaf/0.1/>
```

```
SELECT distinct ?team_name ?athlet ?age
WHERE
```

```
{
```

```
    ?athlet ns1:team ?team.
    ?team rdfs:label ?team_name.
```

```
    FILTER(REGEX(?team_name, '^J', 'i'))
    ?athlet ns3:age ?age.
    FILTER(?age < 30)
```

```
}
```

FILTER ...

Beinhalten Operatoren & Funktionen. Sind Restriktionen für das Query-Ergebnis. Können nach jedem *graph pattern* verwendet werden.

REGEX ... REGEX(input, pattern, flag)

https://en.wikibooks.org/wiki/SPARQL/Expressions_and_Functions#REGEX

Schränkt das *graph pattern* bis hierhin ein. Alle Einträge, die deren Team-Name mit “J oder j” beginnt.
^ ... Wortanfang
i ... ignore case sensitivity

Heißt, das nur für alle Athleten aus einem “J-Team” die ?age ermittelt werden.

UNION

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

```
prefix ns1: <http://dbpedia.org/ontology/>
```

```
prefix ns3: <http://xmlns.com/foaf/0.1/>
```

```
SELECT distinct ?team_name ?athlete ?age
```

```
WHERE{
```

```
{
```

```
  ?athlete ns1:team ?team.
```

```
  ?team rdfs:label ?team_name.
```

```
  FILTER(REGEX(?team_name, '^J', 'i'))
```

```
  ?athlete ns3:age ?age.
```

```
  FILTER(?age < 20)
```

```
}
```

```
UNION
```

```
{
```

```
  ?athlete ns1:team ?team.
```

```
  ?team rdfs:label ?team_name.
```

```
  FILTER(REGEX(?team_name, '^S', 'i'))
```

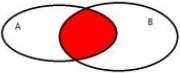
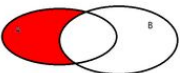
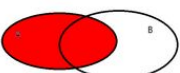
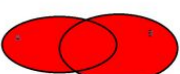
```
  ?athlete ns3:age ?age.
```

```
  FILTER(?age > 40)
```

```
}
```

UNION ...

Erlaubt es unterschiedliche *graph pattern* miteinander zu kombinieren / zu vereinen.

Venn diagram	Mathematical		SPARQL
	$A \cap B$	And	A. B.
	$A \setminus B$		A. FILTER NOT EXISTS{ B. }
	A		A. OPTIONAL{ B. }
	$A \cup B$	Or	{ A. } UNION { B. }

<https://en.wikibooks.org/wiki/SPARQL/UNION>

Operators and Functions

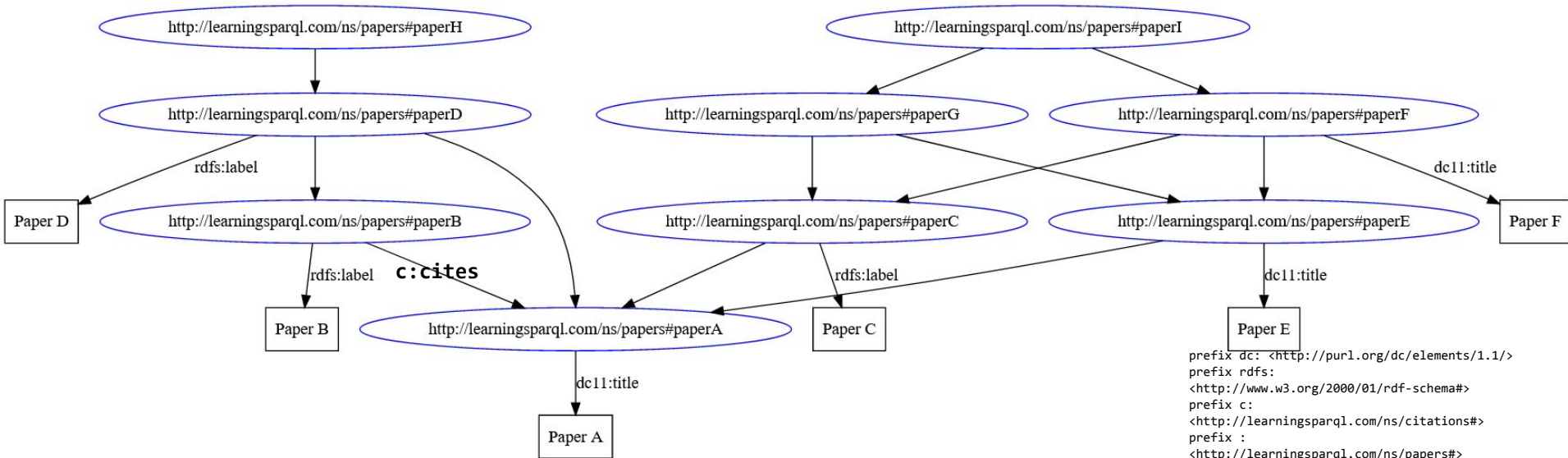
Operator	Type(A)	Result Type
!A	xsd:boolean	xsd:boolean
+A	numeric	numeric
-A	numeric	numeric
BOUND (A)	variable	xsd:boolean
isURI (A)	RDF term	xsd:boolean
isBLANK (A)	RDF term	xsd:boolean
isLITERAL (A)	RDF Term	xsd:boolean
STR (A)	literal/URL	simple literal
LANG (A)	literal	simple literal
DATATYPE (A)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

https://en.wikibooks.org/wiki/SPARQL/Expressions_and_Functions

Session 2

More complex queries

Property Paths - Who cites whom?



```
prefix dc: <http://purl.org/dc/elements/1.1/>
prefix rdfs:
  <http://www.w3.org/2000/01/rdf-schema#>
prefix c:
  <http://learningsparql.com/ns/citations#>
prefix :
  <http://learningsparql.com/ns/papers#>
```

```
:paperA dc:title "Paper A" .
:paperC dc:title "Paper C" .
:paperE dc:title "Paper E" .
:paperF dc:title "Paper F" .
:paperB rdfs:label "Paper B" ;
  c:cites :paperA .
:paperC c:cites :paperA .
:paperD c:cites :paperA , :paperB .
:paperE c:cites :paperA .
:paperF c:cites :paperC , :paperE .
:paperG c:cites :paperC , :paperE .
:paperH c:cites :paperD .
:paperI c:cites :paperF , :paperG .
```

<https://www.w3.org/TR/sparql11-property-paths/>

Data is here:

Property Paths

Übung

Worin unterscheiden
sich diese 4 triple
patterns?

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://learningsparql.com/ns/papers#>
PREFIX c: <http://learningsparql.com/ns/citations#>
```

```
SELECT ?s ?title
WHERE {
  ?s c:cites :paperA .
  #?s c:cites+ :paperA .
  #?s c:cites/c:cites/c:cites :paperA
  # :paperA ^c:cites ?s
  OPTIONAL{
    ?s (dc:title | rdfs:label) ?title.
  }
}
```

Property Paths

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://learningsparql.com/ns/papers#>
PREFIX c: <http://learningsparql.com/ns/citations#>
```

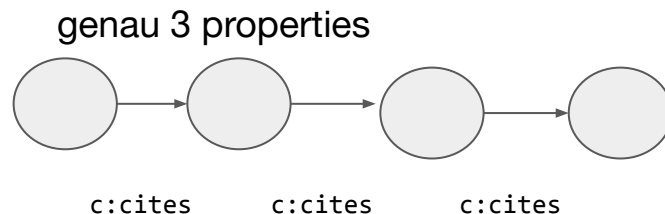
```
SELECT ?s ?title
WHERE {
  ?s c:cites :paperA .
  #?s c:cites+ :paperA .
  #?s c:cites/c:cites/c:cites :paperA
  # :paperA ^c:cites ?s
  OPTIONAL{
    ?s (dc:title | rdfs:label) ?title.
  }
}
```

dc:title | rdfs:label
~Shortcut für UNION; bzw. ODER

?s c:cites+ :paperA

+	...	“one or more”	(8 results)
*	...	“none or more”	(9 results)
?	...	“none or one”	(5 results)

c:cites/c:cites/c:cites



:paperA ^c:cites ?s
inverse property path operator

gleiches Ergebnis;
nur die andere Richtung im Graphen

^ - Inverse Property Paths

filename: ex084.rq

PREFIX : <http://learningsparql.com/ns/papers#>

PREFIX c: <http://learningsparql.com/ns/citations#>

SELECT ?s

WHERE

{

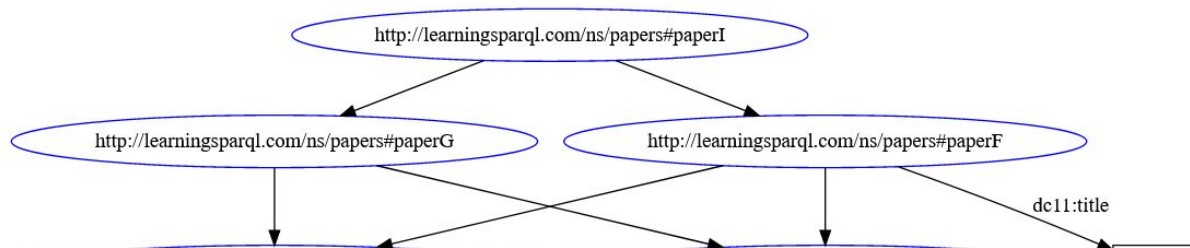
?s c:cites/^c:cites :paperF .

FILTER(?s != :paperF)

}

FILTER(?s != :paperF)

Filter des Ergebnisses:
ist **nicht** Paper F.



Übung

Schreibe eine Query, die alle Namen von Athleten zurückgibt, die mindestens eine Medaille gewonnen haben und aus einem Team kommen, das mit “O” beginnt.

Gib die Gesamtzahl der Medaillen, sortiert nach der Anzahl der Medaillen, aus.

Dafür brauchst du folgende SPARQL - Konstrukte

COUNT() ...

AS ...

GROUP BY ...

ORDER BY DESC...

```
ns130:AnjaSofiaTessPrson    ns13:athlete    ns2:AnjaSofiaTessPrson ;

    ns13:event    ns8:AlpineSkiingWomensCombined ;

    ns13:games    ns44:Winter ;

    ns13:medal    ns50:Bronze .
```

```
ns2:AnjaSofiaTessPrson    ns1:height    "170"^^ns5:int ;
    ns1:team    ns0:Sweden ;
    ns1:weight    81 ;
    a    ns3:Person ;
    ns4:label    "Anja Sofia Tess Prson"@en ;
    ns3:age    "20"^^ns5:int ,
        "24"^^ns5:int ,
        "28"^^ns5:int ;
    ns3:gender    ns6:F .
```

```
PREFIX ns13: <http://wallscope.co.uk/ontology/olympics/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix ns1: <http://dbpedia.org/ontology/>
prefix ns3: <http://xmlns.com/foaf/0.1/>
```

```
SELECT (COUNT(?name) As ?noOfMedals)
WHERE
```

```
{
```

```
    ?instance ns13:athlete ?athlete ;
              ns13:medal ?medal .
```

```
}
```

```
GROUP BY ?name
```

```
ORDER BY DESC(?noOfMedals)
```

```
PREFIX walls:
<http://wallscope.co.uk/ontology/olympics/>
```

```
walls:athlete
```

```
walls:medal
```

<https://codyburleson.com/sparql-examples-count-all-statements/>

Counting?!

```
PREFIX ns13: <http://wallscope.co.uk/ontology/olympics/>
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
prefix ns1: <http://dbpedia.org/ontology/>
```

```
prefix ns3: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?name (COUNT(?name) As ?noOfMedals)
```

```
WHERE
```

```
{
```

```
    ?instance ns13:athlete ?athlete ;
```

```
        ns13:medal    ?medal .
```

```
    ?athlete rdfs:label    ?name .
```

```
    ?athlete ns1:team ?team.
```

```
    ?team rdfs:label ?team_name.
```

```
    FILTER(REGEX(?team_name, '^O', 'i'))
```

```
}
```

```
GROUP BY ?name
```

```
ORDER BY DESC(?noOfMedals)
```

Read oriented query types

4 Types of SPARQL Queries

SELECT queries

Project out specific variables and expressions:

```
SELECT ?c ?cap (1000 * ?people AS ?pop)
```

Project out all variables:

```
SELECT *
```

Project out distinct combinations only:

```
SELECT DISTINCT ?country
```

Results in a table of values (in [XML](#) or [JSON](#)):

?c	?cap	?pop
ex:France	ex:Paris	63,500,000
ex:Canada	ex:Ottawa	32,900,000
ex:Italy	ex:Rome	58,900,000

ASK queries

Ask whether or not there are any matches:

```
ASK
```

Result is either "true" or "false" (in [XML](#) or [JSON](#)):

```
true, false
```

CONSTRUCT queries

Construct RDF triples/graphs:

```
CONSTRUCT {  
  ?country a ex:HolidayDestination ;  
  ex:arrive_at ?capital ;  
  ex:population ?population .  
}
```

Results in RDF triples (in any RDF serialization):

```
ex:France a ex:HolidayDestination ;  
  ex:arrive_at ex:Paris ;  
  ex:population 635000000 .  
ex:Canada a ex:HolidayDestination ;  
  ex:arrive_at ex:Ottawa ;  
  ex:population 329000000 .
```

DESCRIBE queries

Describe the resources matched by the given variables:

```
DESCRIBE ?country
```

Result is RDF triples (in any RDF serialization):

```
ex:France a geo:Country ;  
  ex:continent geo:Europe ;  
  ex:flag <http://.../flag-france.png> ;  
  ...
```

```
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/>
```

```
DESCRIBE ?sport  
WHERE {  
  {  
    ?sport rdf:type dbo:Sport .  
  }  
}
```

```
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/>
```

```
ASK { ?sport rdf:type dbo:Sport . }
```

<http://www.iro.umontreal.ca/~lapalme/ift6281/sparql-1-1-cheat-sheet.pdf>

CONSTRUCT

... ermöglicht es neue Triple in der Ausgabe zu erzeugen.

“The CONSTRUCT query form returns a single RDF graph specified by a graph template.”

Die Tripel können direkt aus einer Datenquelle gezogen werden, ohne sie zu ändern, oder können Werte herausziehen und diese Werte zur Erstellung neuer Tripel verwenden.

PREFIX dbo: <<http://dbpedia.org/ontology/>>

```
CONSTRUCT {  
    ?athlete dbo:height ?height  
}  
WHERE  
{  
    ?athlete dbo:height ?height .  
}
```

subject	predicate	object
http://wallscope.co.uk/resource/olympics/athlete/ViktoriyaPavlivnaKarpenko	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/YelenaViktorovnaDavydovaFilatova	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/YevgeniyaVasilyevnaShishkova	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/YolandaVegaAlbo	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/YuMinobe	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/ZhangNan	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/ZhangShaojie	http://dbpedia.org/ontology/height	148

Construct

PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?athlete ?height

WHERE {

?athlete dbo:height ?height .

}

PREFIX dbo: <<http://dbpedia.org/ontology/>>

CONSTRUCT {

?athlete dbo:height ?height

}

WHERE

{

?athlete dbo:height ?height .

}

subject	predicate	object
http://wallscope.co.uk/resource/olympics/athlete/ViktoriyaPavlivnaKarpenko	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/ViktorovnaDavydovaFilatova	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/ViyaVasilyevnaShishkova	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/LaVegaAlbo	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/Abey	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/Alan	http://dbpedia.org/ontology/height	148
http://wallscope.co.uk/resource/olympics/athlete/Abey	http://dbpedia.org/ontology/height	148

hero	height
http://wallscope.co.uk/resource/olympics/athlete/LytonLevisonMphande	127
http://wallscope.co.uk/resource/olympics/athlete/RosarioBriones	127
http://wallscope.co.uk/resource/olympics/athlete/HelmanPalije	128
http://wallscope.co.uk/resource/olympics/athlete/BostonSimbeye	130
http://wallscope.co.uk/resource/olympics/athlete/SalvadorMiranda	130
http://wallscope.co.uk/resource/olympics/athlete/NadiaFezzani	131
http://wallscope.co.uk/resource/olympics/athlete/AnaOlvidoMansoGallego	132
http://wallscope.co.uk/resource/olympics/athlete/KhamisMohamedSaifAlSubhi	132
http://wallscope.co.uk/resource/olympics/athlete/MichaelConway	132
http://wallscope.co.uk/resource/olympics/athlete/SaidMubarakMarhoonAlKhatry	132

SERVICE

```
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 ?sportName (lang(?sportName) as
?lang) ?teamSize
```

```
WHERE
```

```
{
  SERVICE <http://dbpedia.org/sparql>
  {
    ?dbsport rdfs:label ?sportName;
    dbo:teamSize ?teamSize .
  }
}
```

SERVICE...

Erlaubt eine Query an einen angegebenen SPARQL Endpoint, damit dieser die Abfrage ausführen und dann das Ergebnis zurücksenden kann.

Es können komplette Abfragen und einzelne Graphpattern übergeben werden.

Übung

Erweitere die Query dahingehend, dass nur die englischen Namen der Sportarten ausgegeben werden.

Hinweis:

Verwende dazu FILTER und die Funktion LANG()

```
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 ?sportName ?teamSize
WHERE
{
  SERVICE <http://dbpedia.org/sparql>
  {
    ?dbsport rdfs:label ?sportName;
    dbo:teamSize ?teamSize .
  }
}
```

Übung

Erweitere die Query dahingehend, dass nur die englischen Namen der Sportarten ausgegeben werden.

Hinweis: Verwende dazu FILTER

```
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 ?sportName ?teamSize
WHERE
{
  SERVICE <http://dbpedia.org/sparql>
  {
    ?dbsport rdfs:label ?sportName;
    dbo:teamSize ?teamSize .

    FILTER (LANG(?sportName) = "en")
  }
}
```

SERVICE - Federated Query

```
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 ?sportName ?teamSize
```

```
WHERE
```

```
{
    ?sport rdf:type    dbo:Sport ;
    rdfs:label ?sportName .
}
```

```
SERVICE <http://dbpedia.org/sparql>
```

```
{
    ?dbsport rdfs:label ?sportName ;
             dbo:teamSize ?teamSize .
}
}
```

Eine *Federated Query* ermöglicht es, mehrere verteilte Datasets anzufragen und zu verbinden.

Teile der Query können gegen andere, remote SPARQL Endpoints aufgerufen werden.

Listet alle Sportarten, deren Teamgröße in DBpedia und gibt diese Zahlen, neben der Sportart zurück.

Übung

Konstruiere aus unserer Federated Query neue Triple mit dem CONSTRUCT Keyword,

die jedem Sport die Teamgröße aus DBpedia zuweist.

Ergebnis:

```
<http://wallscope.co.uk/resource/olympics/sport/Baseball>  
    dbo:teamSize 9 .
```

```
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 ?sportName ?teamSize  
WHERE  
{  
    ?sport rdf:type    dbo:Sport ;  
    rdfs:label ?sportName .  
  
    SERVICE <http://dbpedia.org/sparql>  
    {  
        ?dbsport rdfs:label ?sportName ;  
                  dbo:teamSize ?teamSize .  
    }  
}
```

VALUES

```
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 ?sportName ?teamSize
WHERE {
{
VALUES ?sportName {"Curling"@en "Polo"@en "Baseball"@en}
    ?sport rdf:type    dbo:Sport ;
           rdfs:label ?sportName .
SERVICE <http://dbpedia.org/sparql>
{
    ?dbsport rdfs:label ?sportName ;
             dbo:teamSize ?teamSize .
}
}
}
```

... ermöglicht mir, mehrere Werte
bzw. Tabellen von Werten an eine
Variable zu binden.

VALUES

```
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/>
```

```
CONSTRUCT {?sport dbo:teamSize ?teamSize. }
```

```
WHERE {
```

```
{
```

```
    ?sport rdf:type    dbo:Sport ;
```

```
    rdfs:label ?sportName .
```

```
    SERVICE <http://dbpedia.org/sparql>
```

```
    {
```

```
        ?dbsport rdfs:label ?sportName ;
```

```
        dbo:teamSize ?teamSize .
```

```
    }
```

```
} UNION
```

```
{
```

```
    VALUES (?sport ?teamSize) {
```

```
        (<http://wallscope.co.uk/resource/olympics/sport/Soccer> 11)
```

```
        (<http://wallscope.co.uk/resource/olympics/sport/Icehockey> 6)
```

```
    }
```

```
}
```

```
;
```

Query optimization und Performance

- OPTIONAL minimieren/vermeiden
- Suchraum einschränken durch Reihenfolge der verwendeten Pattern
 - Pattern mit drei Variablen ist weniger selektiv
 - Pattern mit gegebenem Prädikat, aber Subjekt und Objekt als Variable mehr Ergebnisse als z.B. mit einem gegebenen Subjekt
- So wenig Variablen wie möglich in SELECT, * vermeiden
- FILTER vermeiden
- UNION wenn möglich durch VALUES ersetzen
- Wenn möglich einfache String Funktionen verwenden anstatt von regex()
- Suchraum einschränken durch die Verwendung von *Named Graphs*
- Mit Query optimization des jeweiligen Triple Stores vertraut machen

<https://wiki.blazegraph.com/wiki/index.php/QueryOptimization>

https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/query_optimization

FILTER

```
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 db: <http://dbpedia.org/resource/>
```

```
SELECT DISTINCT ?pacinoFilmName
WHERE {
    SERVICE <http://dbpedia.org/sparql>
    {
        ?pacinoFilm dbo:starring db:Al_Pacino .
        ?deNiroFilm dbo:starring db:Robert_De_Niro .

        FILTER (?pacinoFilm = ?deNiroFilm)

        ?pacinoFilm rdfs:label ?pacinoFilmName.

        FILTER (LANG(?pacinoFilmName) = "en")
    }
}
```

FILTER

```
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 db: <http://dbpedia.org/resource/>
```

```
SELECT DISTINCT ?pacinoFilmName
WHERE {
    SERVICE <http://dbpedia.org/sparql>
    {
        ?pacinoFilm dbo:starring db:Al_Pacino .
        ?deNiroFilm dbo:starring db:Robert_De_Niro .

        FILTER (?pacinoFilm = ?deNiroFilm)

        ?pacinoFilm rdfs:label ?pacinoFilmName.

        FILTER (LANG(?pacinoFilmName) = "en")
    }
}
```

```
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 db: <http://dbpedia.org/resource/>
```

```
SELECT DISTINCT ?pacinoFilmName
WHERE {
    SERVICE <http://dbpedia.org/sparql>
    {
        ?pacinoFilm dbo:starring db:Al_Pacino .
        ?pacinoFilm dbo:starring db:Robert_De_Niro .

        ?pacinoFilm rdfs:label ?pacinoFilmName.

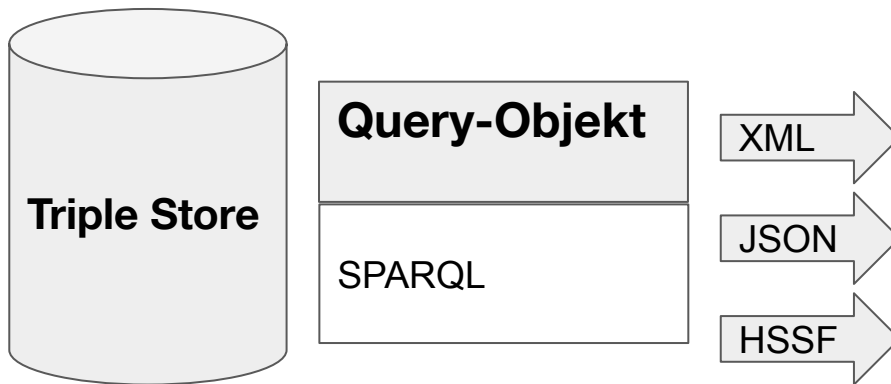
        FILTER (LANG(?pacinoFilmName) = "en")
    }
}
```

Session 3

SPARQL und GAMS

SPARQL-Mini-Hackathon

GAMS



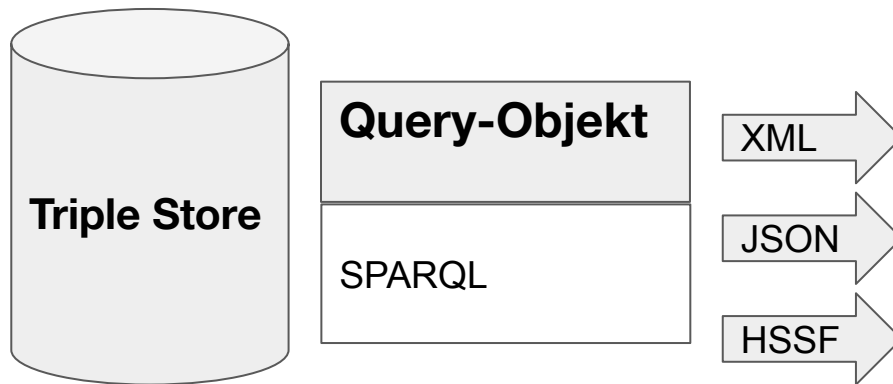
```
<szd:Agent rdf:about="https://gams.uni-graz.at/o:szd.personen#SZDPER.2">
  <szd:forename>Paul</szd:forename>
  <szd:surname>Adam</szd:surname>
  <foaf:page rdf:resource="https://de.wikipedia.org/wiki/Paul_Adam_%28Schl">
  <szd:birth>1862</szd:birth>
  <szd:death>1920</szd:death>
  <gams:textualContent>
    Adam Paul Adan Pol' Adam Paul Auguste Marie Plowert Jacques http://www.1
  </gams:textualContent>
  <gams:isMemberOfCollection rdf:resource="https://gams.uni-graz.at/o:szd
</szd:Agent>
```

<http://stefanzweig.digital/o:szd.bibliothek/RDF>

<http://stefanzweig.digital/o:szd.personen/RDF>

The screenshot shows the GAMS search interface. At the top, there is a navigation bar with links: NACHLASS, SAMMLUNGEN, THEMEN, BIOGRAPHIE, INDEX, GLOSSAR, ABOUT, and a search bar labeled VOLLTEXTSUCHE. Below the navigation bar, the search results are displayed under the heading 'SUCHERGEBNISSE'. The results show 'PERSONENSUCHE: Abraham, Pierre' and 'SUCHERGEBNISSE: 1'. There are buttons for 'ALLE ÖFFNEN', 'DRUCKEN', and 'ALLE ZUM DATENKORB HINZUFÜGEN'. A 'FILTER' section shows 'ALL' selected and 'BIBLIOTHEK' as an option. Below the search results, there is a 'BIBLIOTHEK' section showing a result for 'Abraham, Pierre: Proust: recherches sur la création intellectuelle' with the identifier 'SZDBIB.365'.

http://stefanzweig.digital/archive/objects/query:szd.person_search/methods/sdef:Query/get?params=%241%7C%3Chttps%3A%2F%2Fgams.uni-graz.at%2Fo%3Aszd.personen%23SZDPER.1%3E%3B%242%7Cde&locale=de



Eine Fotocollage aller Personen im Kontext des Nachlasses von Stefan Zweig

```

prefix owl: <http://www.w3.org/2002/07/owl#>
PREFIX szd:<https://gams.uni-graz.at/o:szd.ontology#>
PREFIX bds:<http://www.bigdata.com/rdf/search#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX gams:<https://gams.uni-graz.at/o:gams-ontology#>
PREFIX wde:<http://www.wikidata.org/entity/>
PREFIX wdp:<http://www.wikidata.org/prop/direct/>
SELECT distinct ?re ?wikidata_id ?forename ?surname
?img
where
{
  ?re a szd:Agent.
  ?re szd:wikidata ?wikidata_id.
  ?re szd:forename ?forename.
  ?re szd:surname ?surname.
  SERVICE <https://query.wikidata.org/sparql>
  {
    ?wikidata_id wdp:P18 ?img.
  }
}

```

- **Datenstrom im Query-Object**

<http://glossa.uni-graz.at/archive/objects/query:szd.collage/datastreams/QUERY/content>

- **XML - RESULT**

<http://glossa.uni-graz.at/archive/objects/query:szd.collage/methods/sdef:Query/getXML?params=>

- **HTML**

<http://glossa.uni-graz.at/archive/objects/query:szd.collage/methods/sdef:Query/get>

Anhang

Linked Data Triplestores

- **Blazegraph**
- **Stardog**
- **Virtuoso**
- **GraphDB**
- **AnzoGraph**
- **AllegroGraph**
- **MarkLogic**
- **Apache Rya**

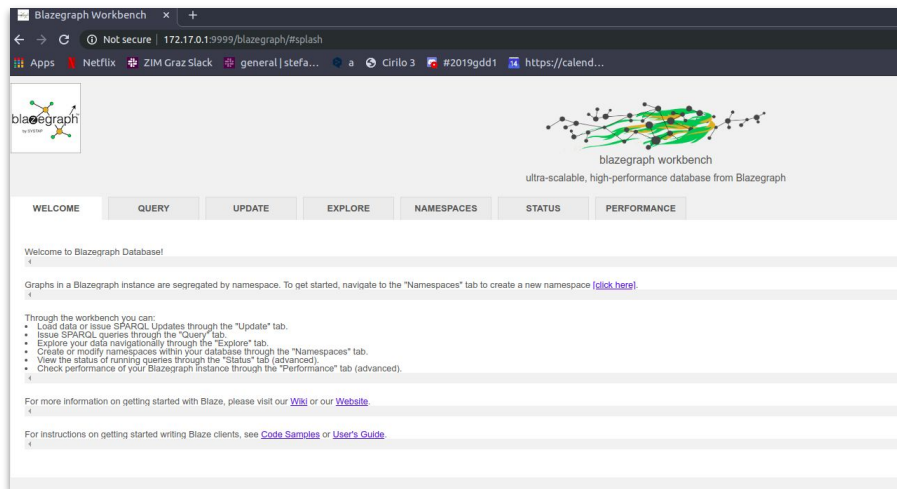
Blazegraph

Download: <https://sourceforge.net/projects/bigdata/files/bigdata/2.1.5/blazegraph.jar/download>

Quick Start: https://wiki.blazegraph.com/wiki/index.php/Quick_Start

JAR starten: `java -server -Xmx4g -jar blazegraph.jar`

Im Browser erreichbar unter : <http://localhost:9999/blazegraph/#splash>



https://wiki.blazegraph.com/wiki/index.php/Main_Page

Blazegraph (Windows 10)

Download: <https://sourceforge.net/projects/bigdata/files/bigdata/2.1.5/blazegraph.jar/download>

Quick Start: https://wiki.blazegraph.com/wiki/index.php/Quick_Start

- Das .jar in einen Ordner ablegen.
- Java SDK muss installiert sein.
- Über die Eingabeaufforderung (CMD) in diesen Ordner wechseln.
- `java -jar blazegraph.jar`

Im Browser erreichbar unter der Adresse, die im Terminal steht.
(<http://localhost:9999/blazegraph/>)

```
WARN : NanoSparqlServer.java:517: Starting NSS
WARN : ServiceProviderHook.java:171: Running.
serviceURL: http://192.168.56.1:9999

Welcome to the Blazegraph(tm) Database.

Go to http://192.168.56.1:9999/blazegraph/ to get started.
WARN : MapgraphServiceProxy.java:67: Running without GPU Acceleration.
lated/.
```

Beispieldaten

Addlesee Angus

Jeweils die .ttl Files herunterladen und in die Blazegraph ingestieren. Unter Update drag and drop rein.

[wallscope/olympics-rdf](#)

(data/olympics-ttl-nodup/olympics.ttl)

[wallscope/superhero-rdf](#)

(data/superhero.ttl)

Blazegraph mit Daten befüllen

Im Reiter *QUERY* werden die SPARQL formuliert.

Mit folgender Query kann man überprüfen, was sich in der DB befindet:

```
SELECT ?a ?b ?c WHERE {?a ?b ?c} LIMIT 100
```

Sollte zu Beginn leer sein. Über den Reiter *UPDATE* kann man Daten in die DB einspielen.

Alle RDF Daten also z.B. .rdf oder .ttl Files können per Drag and Drop, Copy-Paste oder Path ausgewählt und ingestiert werden.

Mit `SELECT ?a ?b ?c WHERE {?a ?b ?c} LIMIT 100` sollte dann was in der DB stehen

FRAGEN

Gibt es eine Möglichkeit datentyp-agnostisch abzufragen?

Ja, indem ich der Query mitteile, dass sie alles als String bewerten soll und die `str()` Funktion dazu verwende. Ein Beispiel

Dataset:

```
@prefix http://learningsparql.com/ns/data#> .  
@prefix dm: <http://learningsparql.com/ns/demo#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
@prefix mt: <http://learningsparql.com/ns/mytypesystem#> .  
  
d:item2a dm:prop "two" .  
d:item2b dm:prop "two"^^xsd:string .  
d:item2c dm:prop "two"^^mt:potrzebies .  
d:item2d dm:prop "two"@en .
```

Diese Query gibt alle 4 Items zurück, unabhängig von deren Datentyp:

```
SELECT ?s ?o  
WHERE  
{  
    ?s ?p ?o .  
    FILTER (str(?o) = "two")  
}
```