SPARQL

Pedro Szekely
Jose Luis Ambite
University of Southern California

Architecture of Semantic Web Applications

slide by Pedro Szekely

Motivation: Music!

EUCLID
EdUcational Curriculum for the usage of LInked Data

Application

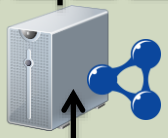
Analysis & Mining Module

RDFa

Visualization Module

Access

SPARQL Endpoint



Publishing

LD Dataset

Vocabulary Mapping

Interlinking

Cleansing

Integrated Dataset

Data acquisition

Physical Wrapper

LD Wrapper

D2R Transf.

LD Wrapper

Streaming providers

Downloads

Musical Content

Metadata

Other content

RDF/ XML

Basic SPARQL

slide by Pedro Szekely

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

SPARQL

SELECT

Get data

ASK

Yes/No questions

CONSTRUCT

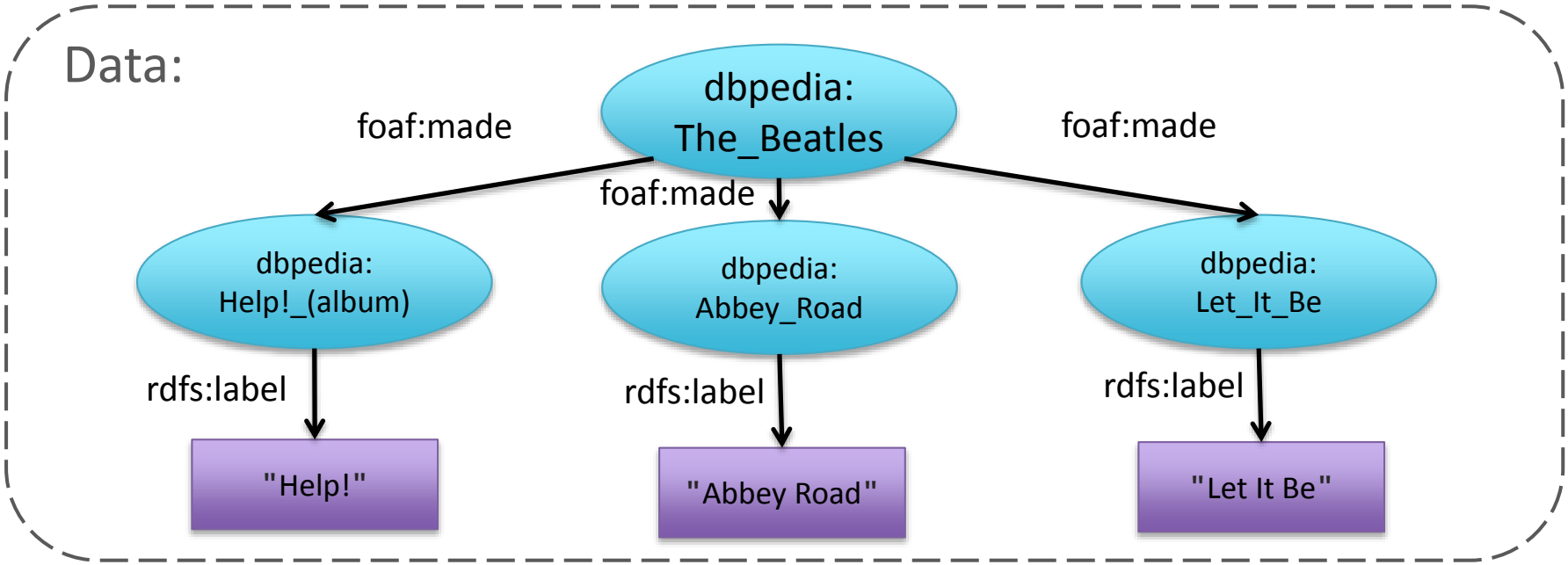
Create RDF

DESCRIBE

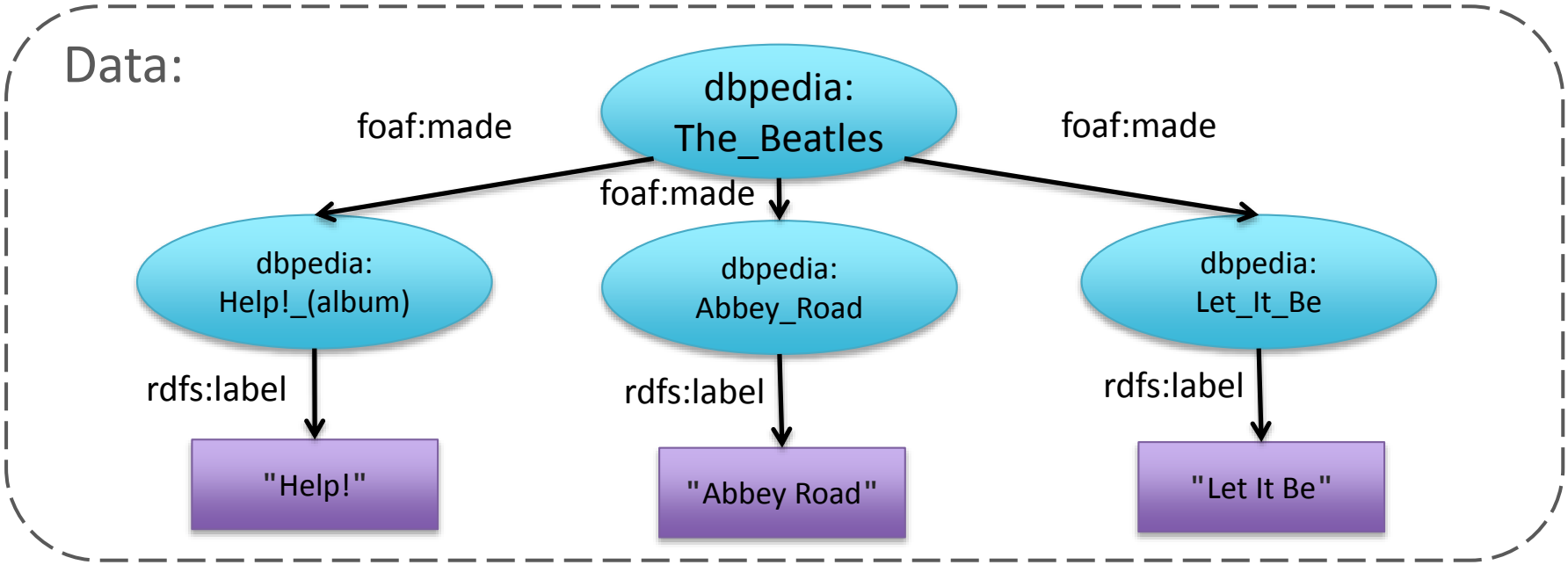
Get some information

slide by Pedro Szekely

SPARQL Query



SPARQL Query



Graph patterns:

Results:

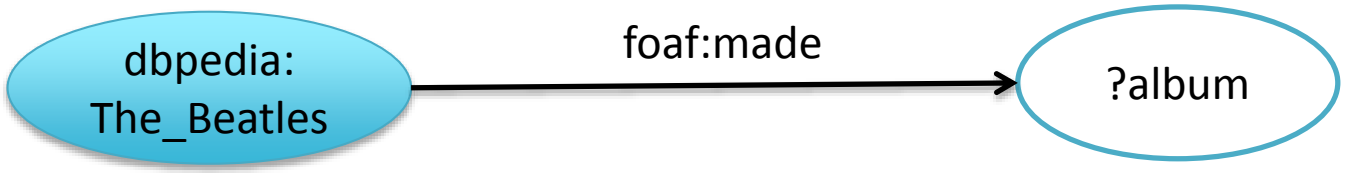
?album
dbpedia:Help!_(album)
dbpedia:Abbey_Road
dbpedia:Let_It_Be

SPARQL Query

EUCLID
EdUcational Curriculum for the usage of LInked Data

Main idea: **Pattern matching**

- Queries describe sub-graphs of the queried graph
- **Graph patterns** are RDF graphs specified in Turtle syntax, which contain variables (prefixed by either “?” or “$”)



- Sub-graphs that match the graph patterns yield a **result**

Simple Query

Data

```
<http://example.org/book/book1>
<http://purl.org/dc/elements/1.1/title>
"SPARQL Tutorial" .
```

Query

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1>
  <http://purl.org/dc/elements/1.1/title>
  ?title .
}
```

Result

title

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

Multiple Matches

Data

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

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

Result

name	mbox

Blank Nodes

Data

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

_:a  foaf:name   "Alice" .
_:b  foaf:name   "Bob" .
```

Query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?x ?name
WHERE  { ?x foaf:name ?name }
```

Result

x	name
_:c	
_:d	

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

Creating Values with Expressions

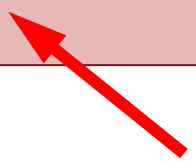
Data

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

_:a  foaf:givenName   "John" .
_:a  foaf:surname  "Doe" .
```

Query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE  {
   ?P foaf:givenName ?G ;
      foaf:surname ?S
   BIND(CONCAT(?G, " ", ?S) AS ?name)
}
```



Result

name

Selection: Restricting the Value of Strings

Data

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

Query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "^SPARQL")
        }
```

Result

title

Selection: Restricting Numeric Values

Data

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

Query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5)
          ?x dc:title ?title . }
```

Result

title	price
"The Semantic Web"	

"SPARQL Tutorial" too expensive

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

Some Syntax (Prefix)

Query

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { <http://example.org/book/book1> dc:title ?title }
```

URIs in angle brackets as <http://…>

Query

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

SELECT  ?title
WHERE   { :book1  dc:title  $title }
```

Empty prefix

Query

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

SELECT  ?title
WHERE   { <book1>  dc:title  ?title }
```

Define BASE: no need to write long URIs

More Syntax (Blank Nodes)

Query Fragment

```
?x  a  :Class1 .
[ a :appClass ] :p "v" .
```

Short form

Query Fragment

```
?x    rdf:type  :Class1 .
_:b0  rdf:type  :appClass .
_:b0  :p        "v" .
```

Long form

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

Graph Patterns

- Basic Graph Patterns,
  - where a set of triple patterns must match
- Group Graph Pattern: {}
  - where a set of graph patterns must all match
- Optional Graph patterns: OPTIONAL
  - where additional patterns may extend the solution
- Alternative Graph Pattern: UNION
  - where two or more possible patterns are tried
- Patterns on Named Graphs: GRAPH
  - where patterns are matched against named graphs

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

Group Graph Patterns

Query

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {
          ?x foaf:name ?name .
          ?x foaf:mbox ?mbox .
       }
```

One basic graph pattern

Query

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { { ?x foaf:name ?name . }
         { ?x foaf:mbox ?mbox . }
       }
```

Two group graph patterns

Scope of Filters

Query Fragment

```
{   ?x foaf:name ?name .
    ?x foaf:mbox ?mbox .
    FILTER regex(?name, "Smith")
}
```

Query Fragment

```
{   FILTER regex(?name, "Smith")
    ?x foaf:name ?name .
    ?x foaf:mbox ?mbox .
}
```

Query Fragment

```
{   ?x foaf:name ?name .
    FILTER regex(?name, "Smith")
    ?x foaf:mbox ?mbox .
}
```

Scope is whole group where filter appears

Optional Pattern Matching

Data

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a  rdf:type        foaf:Person .
_:a  foaf:name       "Alice" .
_:a  foaf:mbox       <mailto:alice@example.com> .
_:a  foaf:mbox       <mailto:alice@work.example> .

_:b  rdf:type        foaf:Person .
_:b  foaf:name       "Bob" .
```

Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x  foaf:mbox  ?mbox }
       }
```

Result

name	mbox
"Alice"	<mailto:alice@example.com>
"Alice"	<mailto:alice@work.example>

Multiple Optional Graph Patterns

Data

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

_:a  foaf:name       "Alice" .
_:a  foaf:homepage   <http://work.example.org/alice/> .

_:b  foaf:name       "Bob" .
_:b  foaf:mbox       <mailto:bob@work.example> .
```

Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x foaf:mbox ?mbox } .
         OPTIONAL { ?x foaf:homepage ?hpage }
       }
```

Result

name	mbox	hpage
"Alice"		<http://work.example.org/alice/>
"Bob"	<mailto:bob@work.example>	

UNION

Data

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .

_:a   dc10:title     "SPARQL Query Language Tutorial" .
_:a   dc10:creator   "Alice" .
_:b   dc11:title     "SPARQL Protocol Tutorial" .
_:b   dc11:creator   "Bob" .
_:c   dc10:title     "SPARQL" .
_:c   dc11:title     "SPARQL (updated)" .
```

Query

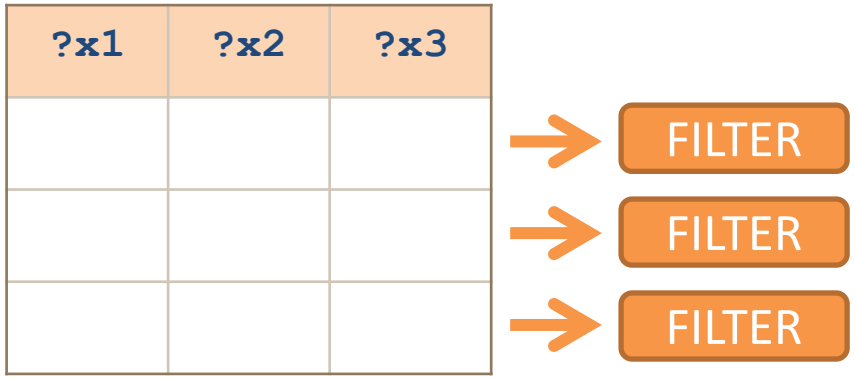
```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION
         { ?book dc11:title  ?title }
       }
```

Result

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

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

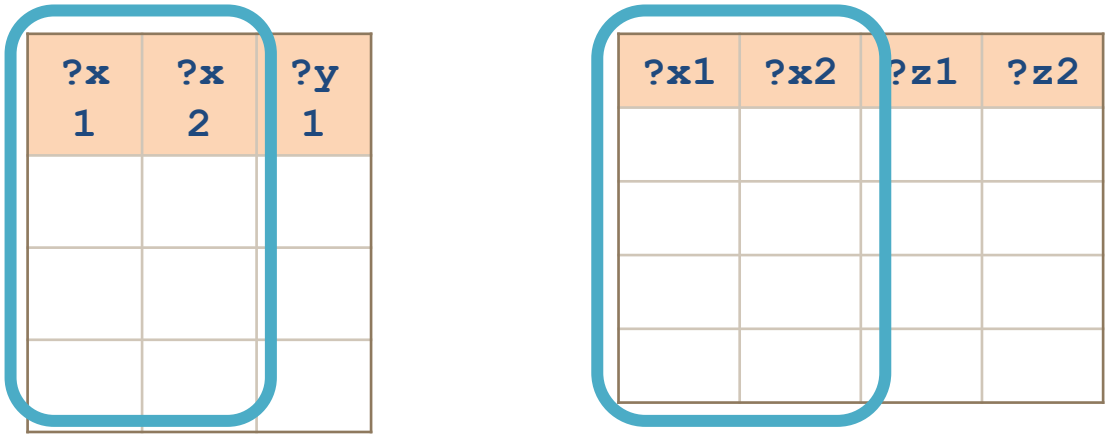
FILTER NOT EXISTS



testing whether a pattern exists
in the data,
given the bindings already
determined by the query pattern

MINUS

Graph Pattern MINUS Graph Pattern



evaluates both its arguments,
then calculates solutions in
the left-hand side that are not
compatible with the solutions
on the right-hand side

Negation: Absence of a Pattern

Data

```
@prefix  :       <http://example/> .
@prefix  rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  foaf:   <http://xmlns.com/foaf/0.1/> .

:alice  rdf:type   foaf:Person .
:alice  foaf:name  "Alice" .
:bob    rdf:type   foaf:Person .
```

Query

```
PREFIX  rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  foaf:   <http://xmlns.com/foaf/0.1/>

SELECT ?person
WHERE
{
    ?person rdf:type  foaf:Person .
    FILTER NOT EXISTS { ?person foaf:name ?name }
}
```

Result

person

Can also do FILTER

slide by Pedro Szekely

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

Negation: Removing Possible Solutions

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

:alice  foaf:givenName "Alice" ;
        foaf:familyName "Smith" .

:bob    foaf:givenName "Bob" ;
        foaf:familyName "Jones" .

:carol  foaf:givenName "Carol" ;
        foaf:familyName "Smith" .
```

Data

Query

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

SELECT DISTINCT ?s
WHERE {
   ?s ?p ?o .
    MINUS {
       ?s foaf:givenName "Bob" .
    }
}
```

Result

s

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

See section 8.3

FILTER vs MINUS, Example 1

Query "FILTER"

```
SELECT *
{
  ?s ?p ?o
  FILTER NOT EXISTS { ?x ?y ?z }
}
```

Data

```
@prefix : <http://example/> .
:a :b :c .
```

Result "FILTER"

FILTER vs MINUS, Example 1

Query "FILTER"

```
SELECT *
{
  ?s ?p ?o
  FILTER NOT EXISTS { ?x ?y ?z }
}
```

Data

```
@prefix : <http://example/> .
:a :b :c .
```

Result "FILTER"

s	p	o

Query "MINUS"

```
SELECT *
{
   ?s ?p ?o
   MINUS
     { ?x ?y ?z }
}
```

No shared variables!

Result "MINUS"

FILTER vs MINUS, Example 2

Query "FILTER"

```
PREFIX : <http://example/>
SELECT *
{
  ?s ?p ?o
  FILTER NOT EXISTS { :a :b :c }
}
```

Data

```
@prefix : <http://example/> .
:a :b :c .
```

Result "FILTER"

?s	?p	?o

FILTER vs MINUS, Example 2

Query "FILTER"

```
PREFIX : <http://example/>
SELECT *
{
  ?s ?p ?o
  FILTER NOT EXISTS { :a :b :c }
}
```

Data

```
@prefix : <http://example/> .
:a :b :c .
```

Result "FILTER"

s	p	o

Query "MINUS"

```
PREFIX : <http://example/>
SELECT *
{
  ?s ?p ?o
  MINUS { :a :b :c }
}
```

No shared variables!

Result "MINUS"

Brain Teaser: Inner Filter

Data

```
@prefix : <…
:a :p 1 .
:a :q 1 .
:a :q 2 .

:b :p 3.0 .
:b :q 4.0 .
:b :q 5.0 .
```

Query "FILTER"

```
PREFIX : <http://example.com/>
SELECT * WHERE {
        ?x :p ?n
        FILTER NOT EXISTS {
                ?x :q ?m .
                FILTER(?n = ?m)
        }
}
```

Result "FILTER"

x	n

Brain Teaser: Inner Filter

Data

```
@prefix : <…
:a :p 1 .
:a :q 1 .
:a :q 2 .

:b :p 3.0 .
:b :q 4.0 .
:b :q 5.0 .
```

Query "FILTER"

```
PREFIX : <http://example.com/>
SELECT * WHERE {
        ?x :p ?n
        FILTER NOT EXISTS {
                ?x :q ?m .
                FILTER(?n = ?m)
        }
}
```

Result "FILTER"

x	n
<http://example.com/b>	3.0

Query "MINUS"

```
PREFIX : <http://example/>
SELECT * WHERE {
        ?x :p ?n
        MINUS {
                ?x :q ?m .
                FILTER(?n = ?m)
        }
}
```

Result "MINUS"

x	n

Property Paths

Query Fragment

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

Alternatives: Match one or both possibilities

Query Fragment

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

Sequence: Find the name of any people that Alice knows.

Query Fragment

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

Sequence: Find the names of people 2 "foaf:knows" links away.

Query Fragment

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

Arbitrary length match:

Find the names of all the people that can be reached from Alice by “foaf:knows”:

Property Path Semantics

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

=

```
{
  ?x foaf:mbox <mailto:alice@example> .
  ?x  foaf:knows ?a1 .
  ?a1 foaf:knows ?a2 .
  ?a2 foaf:name ?name .
}
```

=

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

BIND: Assigning to Variables

Data

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title     "SPARQL Tutorial" .
:book1  ns:price     42 .
:book1  ns:discount  0.2 .

:book2  dc:title     "The Semantic Web" .
:book2  ns:price     23 .
:book2  ns:discount  0.25 .
```

Query

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

SELECT  ?title ?price
{   ?x ns:price ?p .
    ?x ns:discount ?discount
    BIND (?p*(1-?discount) AS ?price)
    FILTER(?price < 20)
    ?x dc:title ?title .
}
```

Result

title	price
"The Semantic Web"	17.25

Aggregation

Data

```
@prefix : <http://books.example/> .

:org1 :affiliates :auth1, :auth2 .
:auth1 :writesBook :book1, :book2 .
:book1 :price 9 .
:book2 :price 5 .
:auth2 :writesBook :book3 .
:book3 :price 7 .
:org2 :affiliates :auth3 .
:auth3 :writesBook :book4 .
:book4 :price 7 .
```

Query

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org :affiliates ?auth .
  ?auth :writesBook ?book .
  ?book :price ?lprice .
}
GROUP BY ?org
HAVING (SUM(?lprice) > 10)
```

Bindings

?org	?auth	?book	?1price
:org1	:auth1	:book1	9
:org1	:auth1	:book2	5
:org1	:auth2	:book3	7
:org2	:auth3	:book4	7

21

7

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

Aggregation

Query

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org :affiliates ?auth .
  ?auth :writesBook ?book .
  ?book :price ?lprice .
}
GROUP BY ?org
HAVING (SUM(?lprice) > 10)
```

Bindings

?org	?auth	?book	?lprice
:org1	:auth1	:book1	9
:org1	:auth1	:book2	5
:org1	:auth2	:book3	7
:org2	:auth3	:book4	7

21

7

Result

totalPrice
21

Aggregation

Data

```
@prefix : <http://books.example/> .

:org1 :affiliates :auth1, :auth2 .
:auth1 :writesBook :book1, :book2 .
:book1 :price 9 .
:book2 :price 5 .
:auth2 :writesBook :book3 .
:book3 :price 7 .
:org2 :affiliates :auth3 .
:auth3 :writesBook :book4 .
:book4 :price 7 .
```

Query

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org :affiliates ?auth .
  ?auth :writesBook ?book .
  ?book :price ?lprice .
}
GROUP BY ?org
HAVING (SUM(?lprice) > 10)
```

Result

totalPrice
21

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

Subqueries

Data

```
@prefix : <http://people.example/> .

:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```

Query

```
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
  :alice :knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y :name ?name .
    } GROUP BY ?y
  }
}
```

Result

y	minName
???	???
???	???
???	???

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

Subqueries

Data

```
@prefix : <http://people.example/> .

:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```

Query

```
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
  :alice :knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y :name ?name .
    } GROUP BY ?y
  }
}
```

Subqueries

Data

```
@prefix : <http://people.example/> .

:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```

Query

```
SELECT ?y (MIN(?name) AS ?minName)
WHERE {
  ?y :name ?name .
} GROUP BY ?y
```

Bindings

?y	?name
:alice	"Alice"
:alice	"Alice Foo"
:alice	"A. Foo"
:bob	"Bob"
:bob	"Bob Bar"
:bob	"B. Bar"
:carol	"Carol"
:carol	"Carol Baz"
:carol	"C. Baz"

Result

y	minName
:alice	"A. Foo"
:bob	"B. Bar"
:carol	"C. Baz"

slide by Pedro Szekely

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

Subqueries

Data

```
@prefix : <http://people.example/> .

:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```

Query

```
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
  :alice :knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y :name ?name .
    } GROUP BY ?y
  }
}
```

Subquery Result

y	minName
:alice	"A. Foo"
:bob	"Bob Bar"
:carol	"C. Baz"

slide by Pedro Szekely, Jose Luis Ambite

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

Subqueries

Data

```
@prefix : <http://people.example/> .

:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```

Query

```
PREFIX : <http://people.example/>
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
  :alice :knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y :name ?name .
    } GROUP BY ?y
  }
}
```

y
:bob
:carol

Subquery Result

Subquery Result

y	minName
:alice	"A. Foo"
:bob	"Bob Bar"
:carol	"C. Baz"

slide by Pedro Szekely, Jose Luis Ambite

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

Subqueries

Query

```
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
  :alice :knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y :name ?name .
    } GROUP BY ?y
  }
}
```

Return a name (the one with the lowest sort order) for all the people that know Alice and have a name.

Subquery Result

y
:bob
:carol

JOIN

Subquery Result

y	minName
:alice	"A. Foo"
:bob	"Bob Bar"
:carol	"C. Baz"

=

Result

y	minName
:bob	"B. Bar"
:carol	"C. Baz"

Subqueries

Data

```
@prefix : <http://people.example/> .

:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```

Query

```
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
  :alice :knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y :name ?name .
    } GROUP BY ?y
  }
}
```

Result

y	minName
:bob	"B. Bar"
:carol	"C. Baz"

slide by Pedro Szekely, Jose Luis Ambite

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

RDF Dataset =

default graph

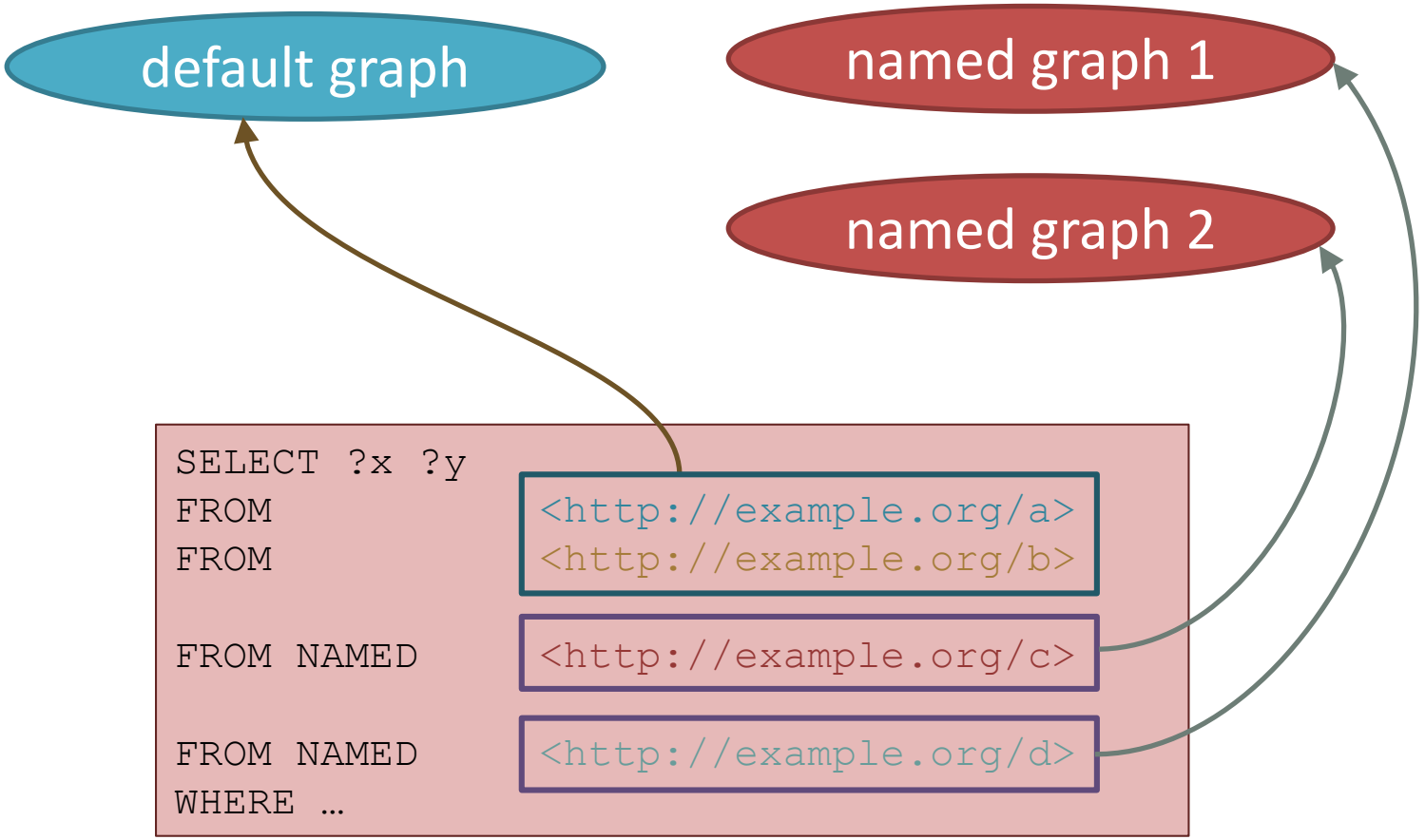
+ named graph 1

+ named graph 2

+ …

… the SPARQL queries seen so far target the default graph

Specifying Datasets Explicitly



Default graph = “RDF merged” graphs in FROM clauses

RDF merge = union N-triples, renaming blank nodes to not conflict

RDF Datasets

Default Graph

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

<http://example.org/bob>    dc:publisher  "Bob" .
<http://example.org/alice>  dc:publisher  "Alice" .
```

Provenance

Named Graph 1: http://example.org/bob

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

_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .
```

Named Graph 2: http://example.org/alice

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

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example.org> .
```

Graphs can be merged

[Note that blank nodes _:a represent different objects in each of the named graphs!]

Separate graphs enable you to reason about who said what and when (provenance)

Provenance Reasoning

prefixes omitted to save space

Default Graph

```
g:graph1 dc:publisher "Bob" .
g:graph1 dc:date "2004-12-06"^^xsd:date .

g:graph2 dc:publisher "Bob" .
g:graph2 dc:date "2005-01-10"^^xsd:date .
```

Named Graph 1 RDF collected on 2004-12-06

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@oldcorp.example.org> .
```

Named Graph 2 RDF collected on 2005-01-10

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@newcorp.example.org> .
```

Default Graph

```
g:graph1 dc:publisher "Bob" .
g:graph1 dc:date "2004-12-06"^^xsd:date .

g:graph2 dc:publisher "Bob" .
g:graph2 dc:date "2005-01-10"^^xsd:date .
```

Named Graph 1

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@oldcorp.example.org> .
```

Named Graph 2

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@newcorp.example.org> .
```

?

Result

name	mbox	date
"Bob"	<mailto:bob@oldcorp.example.org>	"2004-12-06"^^xsd:date
"Bob"	<mailto:bob@newcorp.example.org>	"2005-01-10"^^xsd:date

Default Graph

```
g:graph1 dc:publisher "Bob" .
g:graph1 dc:date "2004-12-06"^^xsd:date .
g:graph2 dc:publisher "Bob" .
g:graph2 dc:date "2005-01-10"^^xsd:date .
```

Named Graph 1

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@oldcorp.example.org> .
```

Named Graph 2

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@newcorp.example.org> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
SELECT ?name ?mbox ?date
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE
  {  ?g dc:publisher ?name ;
        dc:date ?date .
     GRAPH ?g
       { ?person foaf:name ?name ; foaf:mbox ?mbox }
  }
```

from Default Graph

from Named Graphs

slide by Pedro Szekely, Jose Luis Ambite

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

Default Graph

```
g:graph1 dc:publisher "Bob" .
g:graph1 dc:date "2004-12-06"^^xsd:date .
g:graph2 dc:publisher "Bob" .
g:graph2 dc:date "2005-01-10"^^xsd:date .
```

Named Graph 1

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@oldcorp.example.org> .
```

Named Graph 2

```
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@newcorp.example.org> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
SELECT ?name ?mbox ?date
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE
  {  ?g dc:publisher ?name ; dc:date ?date .
    GRAPH ?g
      { ?person foaf:name ?name ; foaf:mbox ?mbox }
  }
```

from Default Graph

from Named Graphs

name	mbox	date
"Bob"	<mailto:bob@oldcorp.example.org>	"2004-12-06"^^xsd:date
"Bob"	<mailto:bob@newcorp.example.org>	"2005-01-10"^^xsd:date

slide by Pedro Szekely, Jose Luis Ambite

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

Take the following four named graphs...

<http://grapha.com> = { <a1> <p> <a2> . }
<http://graphb.com> = { <b1> <p> <b2> . }
<http://graphc.com> = { <c1> <p> <c2> . }
<http://graphd.com> = { <d1> <p> <d2> . }

SELECT ?s WHERE { ?s <p> ?o }
will often give <a1>, <b1>, <c1>, <d1>, but this depends on what the default graph is implicitly defined as.

http://answers.semanticweb.com/questions/11509/what-is-the-difference-between-from-and-from-named

Take the following four named graphs...

<http://grapha.com> = { <a1> <p> <a2> . }
<http://graphb.com> = { <b1> <p> <b2> . }
<http://graphc.com> = { <c1> <p> <c2> . }
<http://graphd.com> = { <d1> <p> <d2> . }

SELECT ?s WHERE { ?s <p> ?o }
will often give <a1>, <b1>, <c1>, <d1>, but this depends on what the default graph is implicitly defined as.

FROM <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
should give <a1>.

http://answers.semanticweb.com/questions/11509/what-is-the-difference-between-from-and-from-named

Take the following four named graphs...

<http://grapha.com> = { <a1> <p> <a2> . }
<http://graphb.com> = { <b1> <p> <b2> . }
<http://graphc.com> = { <c1> <p> <c2> . }
<http://graphd.com> = { <d1> <p> <d2> . }

SELECT ?s WHERE { ?s <p> ?o }
will often give <a1>, <b1>, <c1>, <d1>, but this depends on what the default graph is implicitly defined as.

FROM <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
should give <a1>.

FROM NAMED <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
should give nothing.

http://answers.semanticweb.com/questions/11509/what-is-the-difference-between-from-and-from-named

Take the following four named graphs...

<http://grapha.com> = { <a1> <p> <a2> . }
<http://graphb.com> = { <b1> <p> <b2> . }
<http://graphc.com> = { <c1> <p> <c2> . }
<http://graphd.com> = { <d1> <p> <d2> . }

SELECT ?s WHERE { ?s <p> ?o }
will often give <a1>, <b1>, <c1>, <d1>, but this depends on what the default graph is implicitly defined as.

FROM <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
should give <a1>.

FROM NAMED <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
should give nothing.

FROM <http://grapha.com>
FROM <http://graphb.com>
FROM NAMED <http://graphc.com>
FROM NAMED <http://graphd.com>
SELECT ?s WHERE { ?s <p> ?o }
should give <a1>, <b1>.

http://answers.semanticweb.com/questions/11509/what-is-the-difference-between-from-and-from-named

Take the following four named graphs...

<http://grapha.com> = { <a1> <p> <a2> . }
<http://graphb.com> = { <b1> <p> <b2> . }
<http://graphc.com> = { <c1> <p> <c2> . }
<http://graphd.com> = { <d1> <p> <d2> . }

SELECT ?s WHERE { ?s <p> ?o }
will often give <a1>, <b1>, <c1>, <d1>, but this depends on what the default graph is implicitly defined as.

FROM <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
should give <a1>.

FROM NAMED <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
should give nothing.

FROM <http://grapha.com>
FROM <http://graphb.com>
FROM NAMED <http://graphc.com>
FROM NAMED <http://graphd.com>
SELECT ?s WHERE { ?s <p> ?o }
should give <a1>, <b1>.

FROM <http://grapha.com>
FROM <http://graphb.com>
FROM NAMED <http://graphc.com>
FROM NAMED <http://graphd.com>
SELECT ?s WHERE { GRAPH ?g { ?s <p> ?o }}
should give <c1>, <d1>.

http://answers.semanticweb.com/questions/11509/what-is-the-difference-between-from-and-from-named

Take the following four named graphs...

<http://grapha.com> = { <a1> <p> <a2> . }
<http://graphb.com> = { <b1> <p> <b2> . }
<http://graphc.com> = { <c1> <p> <c2> . }
<http://graphd.com> = { <d1> <p> <d2> . }

```
SELECT ?s WHERE { ?s <p> ?o }
```
will often give <a1>, <b1>, <c1>, <d1>, but this depends on what the default graph is implicitly defined as.

```
FROM <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
```
should give <a1>.

```
FROM NAMED <http://grapha.com>
SELECT ?s WHERE { ?s <p> ?o }
```
should give nothing.

```
FROM <http://grapha.com>
FROM <http://graphb.com>
FROM NAMED <http://graphc.com>
FROM NAMED <http://graphd.com>
SELECT ?s WHERE { ?s <p> ?o }
```
should give <a1>, <b1>.

```
FROM <http://grapha.com>
FROM <http://graphb.com>
FROM NAMED <http://graphc.com>
FROM NAMED <http://graphd.com>
SELECT ?s WHERE { GRAPH ?g { ?s <p> ?o }}
```
should give <c1>, <d1>.

```
FROM <http://grapha.com>
FROM NAMED <http://graphb.com>
SELECT ?s WHERE {
  GRAPH <http://grapha.com> { ?s <p> ?o }}
```
should give nothing. ...etc.

http://answers.semanticweb.com/questions/11509/what-is-the-difference-between-from-and-from-named

Controlling the Output

```
SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY ?name DESC(?emp)
```

Ordering the solutions

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name
WHERE { ?x foaf:name ?name }
```

Eliminating duplicates

```
SELECT  ?name
WHERE   { ?x foaf:name ?name }
LIMIT   5
OFFSET  10
```

Selecting a range of results

SPARQL

SELECT

Get data

ASK

Yes/No questions

CONSTRUCT

Create RDF

DESCRIBE

Get some information

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

SELECT vs CONSTRUCT

SELECT

Result: table of bindings

x	n
<http://example.com/b>	3.0
<http://example.com/a>	1

CONSTRUCT

Result: RDF

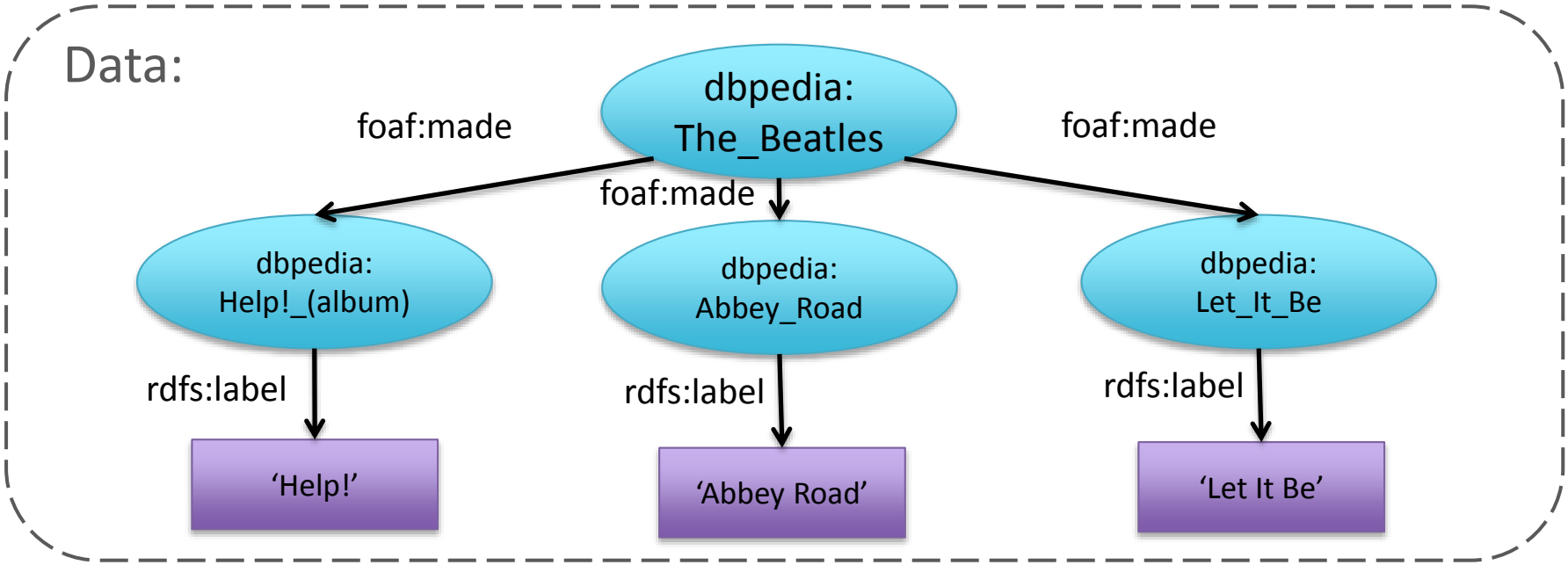
```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .

_:v1 vcard:N          _:x .
_:x vcard:givenName   "Alice" .
_:x vcard:familyName "Hacker" .

_:v2 vcard:N          _:z .
_:z vcard:givenName   "Bob" .
_:z vcard:familyName "Hacker" .
```


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

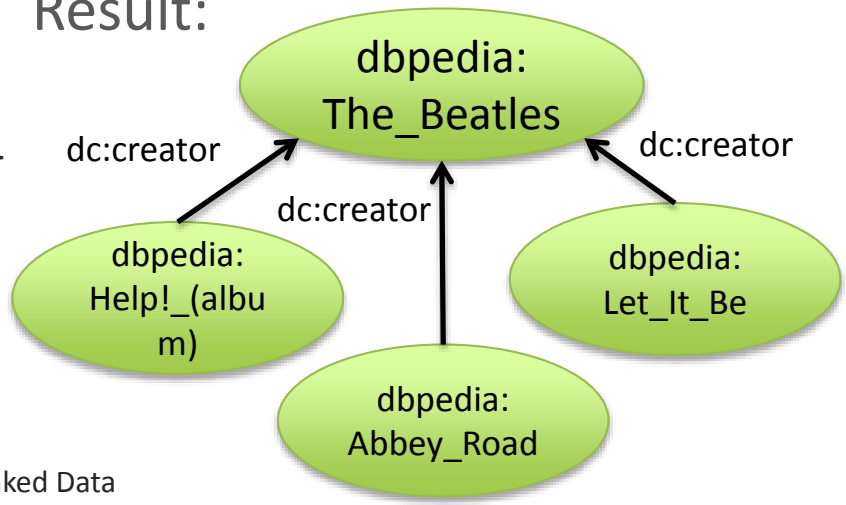
Query Form: CONSTRUCT



Query:

```
CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .}
```

Result:



Query Form: CONSTRUCT

EUCLID
EdUcational Curriculum for the usage of LInked Data

Subsets of results

- It is possible to combine the query with **solution modifiers** (`ORDER BY`, `LIMIT`, `OFFSET`)

Query: *Create the dc:creator descriptions for the 10 most recent albums and their tracks recorded by 'The Beatles'.*

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX music-ont: <http://purl.org/ontology/mo/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
CONSTRUCT {
  ?album dc:creator dbpedia:The_Beatles .
  ?track dc:creator dbpedia:The_Beatles .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .
  ?album music-ont:track ?track ;
         dc:date ?date .
} ORDER BY DESC(?date)
  LIMIT 10
```

Query Form: CONSTRUCT

EUCLID
EdUcational Curriculum for the usage of LInked Data

Assigning Variables

- The value of an expression can be added to a solution mapping by binding a new variable (which can be further used and returned)
- The BIND form allows to assign a value to a variable from a BGP

Query: *Calculate the duration of the tracks from ms to s, and store the value using the dbpedia-ont:runtime property .*

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX music-ont: <http://purl.org/ontology/mo/>
PREFIX dbpedia-ont: <http://dbpedia.org/ontology/>
CONSTRUCT { ?track dbpedia-ont:runtime ?secs .}
WHERE {
  dbpedia:The_Beatles foaf:made ?album .
  ?album music-ont:track ?track .
  ?track music-ont:duration ?duration .
  BIND((?duration/1000) AS ?secs) .}
```

CONSTRUCTing a Graph

Data

```
_:a    foaf:givenname   "Alice" .
_:a    foaf:family_name "Hacker" .
_:b    foaf:firstname   "Bob" .
_:b    foaf:surname     "Hacker" .
```

Query

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT { ?x  vcard:N _:v .
            _:v vcard:givenName ?gname .
            _:v vcard:familyName ?fname }
WHERE {
    { ?x foaf:firstname ?gname } UNION { ?x foaf:givenname   ?gname } .
    { ?x foaf:surname   ?fname } UNION { ?x foaf:family_name ?fname } .
 }
```

Result Data

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .

_:v1 vcard:N          _:x .
_:x vcard:givenName   "Alice" .
_:x vcard:familyName "Hacker" .

_:v2 vcard:N          _:z .
_:z vcard:givenName   "Bob" .
_:z vcard:familyName "Hacker" .
```

Convert from
“foaf”
to
“vcard”

slide by Pedro Szekely

Query Form: ASK

EUCLID
EdUcational Curriculum for the usage of LInked Data

- Namespaces are added with the 'PREFIX' directive
- Statement patterns that make up the graph are specified between brackets ("{}")

Query: *Is Paul McCartney member of 'The Beatles'?*

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpedia-ont: <http://dbpedia.org/ontology/>
ASK WHERE { dbpedia:The_Beatles dbpedia-ont:bandMember
                          dbpedia:Paul_McCartney.}
```

Results:

true

Query: *Is Elvis Presley member of 'The Beatles'?*

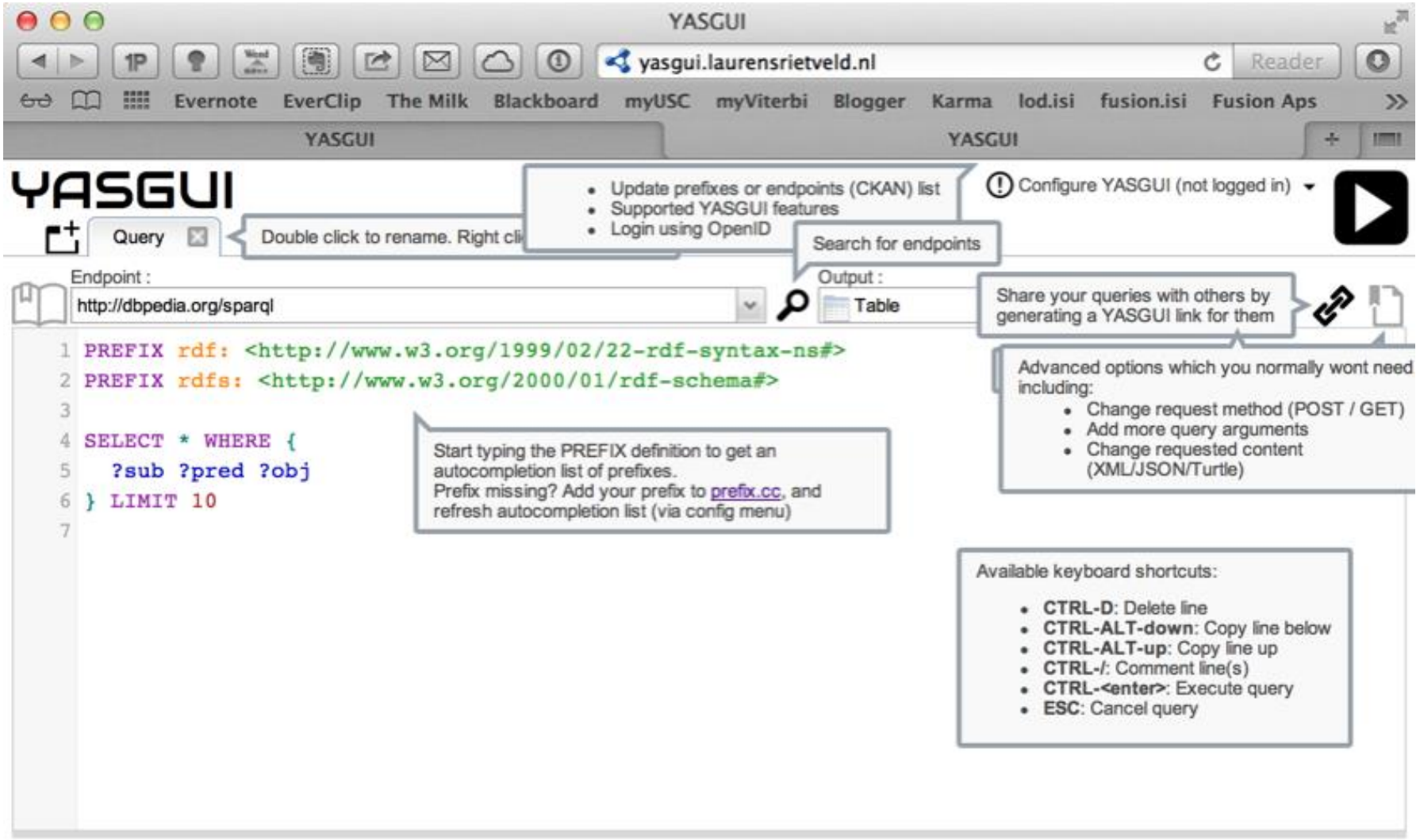
```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpedia-ont: <http://dbpedia.org/ontology/>
ASK WHERE { dbpedia:The_Beatles dbpedia-ont:bandMember
                          dbpedia:Elvis_Presley.}
```

Results:

false

Executing SPARQL Queries

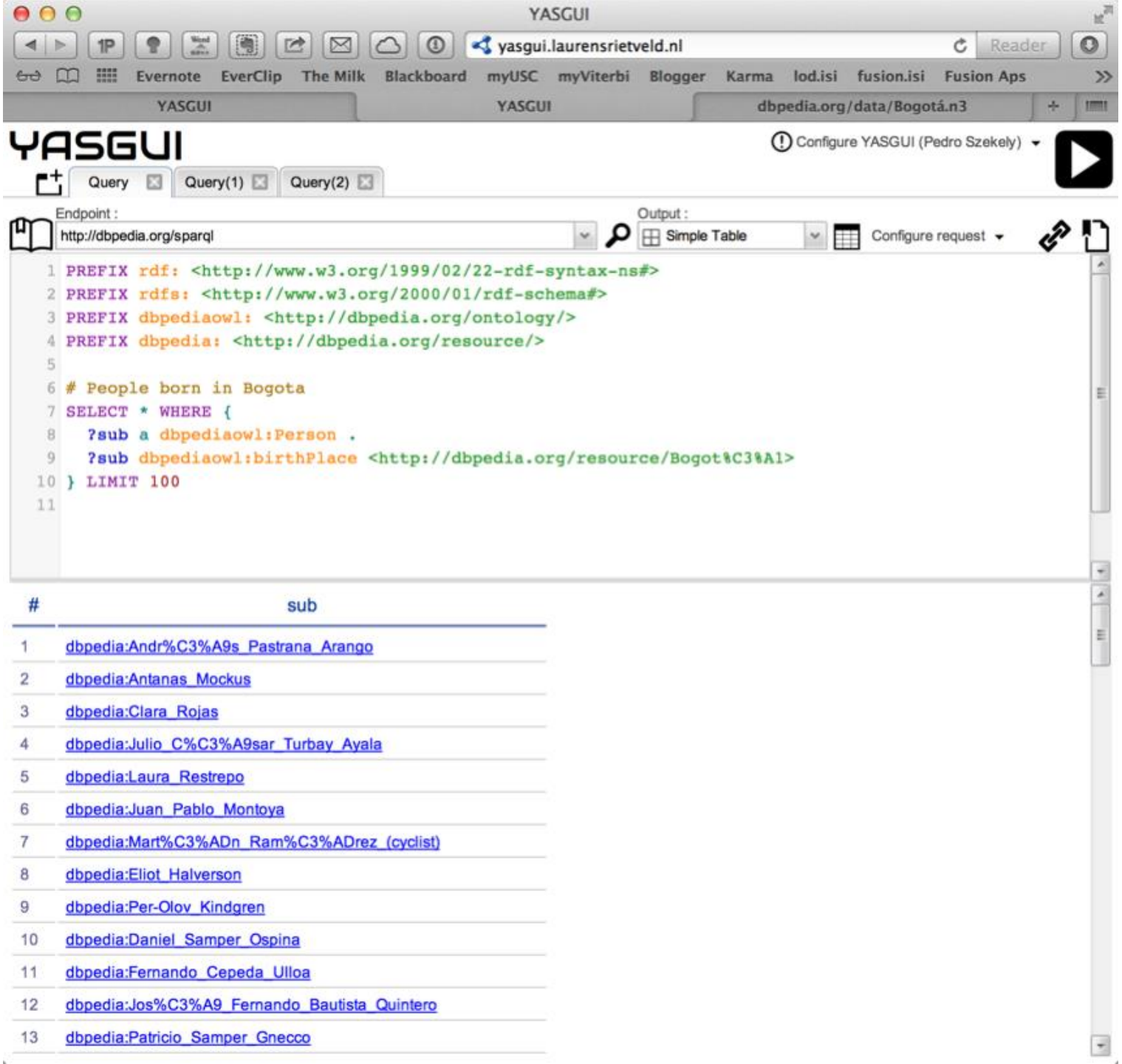
http://yasgui.laurensrietveld.nl



Example Query

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

# People born in Bogota
SELECT * WHERE {
  ?sub a dbpediaowl:Person .
  ?sub dbpediaowl:birthPlace <http://dbpedia.org/resource/Bogot%C3%A1>
} LIMIT 100
```


YASGUI
yasgui.laurensrietveld.nl
Reader
Evernote EverClip The Milk Blackboard myUSC myViterbi Blogger Karma lod.isi fusion.isi Fusion Aps
YASGUI
YASGUI
dbpedia.org/data/Bogotá.n3
YASGUI
Configure YASGUI (Pedro Szekely)
Query
Query(1)
Query(2)
Endpoint :
http://dbpedia.org/sparql
Output :
Simple Table
Configure request
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX dbpediaowl: <http://dbpedia.org/ontology/>
4 PREFIX dbpedia: <http://dbpedia.org/resource/>
5
6 # People born in Bogota
7 SELECT * WHERE {
8 ?sub a dbpediaowl:Person .
9 ?sub dbpediaowl:birthPlace <http://dbpedia.org/resource/Bogot%C3%A1>
10 } LIMIT 100
11
sub
1 dbpedia:Andr%C3%A9s_Pastrana_Arango
2 dbpedia:Antanas_Mockus
3 dbpedia:Clara_Rojas
4 dbpedia:Julio_C%C3%A9sar_Turbay_Ayala
5 dbpedia:Laura_Restrepo
6 dbpedia:Juan_Pablo_Montoya
7 dbpedia:Mart%C3%ADn_Ram%C3%ADrez_(cyclist)
8 dbpedia:Eliot_Halverson
9 dbpedia:Per-Olov_Kindgren
10 dbpedia:Daniel_Samper_Ospina
11 dbpedia:Fernando_Cepeda_Ulloa
12 dbpedia:Jos%C3%A9_Fernando_Bautista_Quintero
13 dbpedia:Patricio_Samper_Gnecco

How can I access CORS-disabled endpoints which are installed locally?

Using this proxy is a great solution in accessing all **public** SPARQL endpoints. However, when you have a CORS-disabled endpoint installed locally on your computer, the YASGUI server is not able to access this endpoint from the outside. To access these endpoints, choose one of the options below

Enable CORS for your endpoint

Install a simple CORS proxy

Use a browser extension

Possible the easiest way of accessing CORS-disabled endpoints is by installing a browser extension. Such browser extensions can modify the response headers, and add the CORS headers automatically for all responses the browser receives.

Firefox

Download and install the firefox browser addon here. After restarting the browser, make sure you have the addons bar enabled (via `View → Toolbars → Add-on Bar`). To enable cross-domain access to all endpoints, click on the *CORS* button in the bottom-right of your screen.

Chrome

Download and install the chrome browser extension here. To enable cross-domain access to all endpoints, click on the *CORS* button in the top-right of your screen.

Forcing cross-domain access for all responses, reduces your browser security. Only enable these plugins during the period you use the CORS-disable endpoints

https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi