Big Graph Processing Systems

Part III: RDF and SPARQL

▶ **Chapter 2:** SPARQL in a Nutshell

Christopher Spinrath

CNRS – LIRIS – Lyon 1 Université

DISS Master 2025

This presentation is based on slides by E. Della Valle

`http://emanueledellavalle.org` – @manudellavalle

What is SPARQL

What is SPARQL?

- SPARQL is short for **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage
- It is a query Language and a protocol

What is SPARQL

What is SPARQL?

- SPARQL is short for **S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage
- It is a query Language and a protocol

Example (Query: Find Drugs in DBpedia)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?s
WHERE { ?s a dbpedia-owl:Drug . }
```

Example (Protocol: Submit a query)

```
http://factforge.net/sparql?query=PREFIX+dbpedia-owl%3A+%3Chttp%3A%2F%2Fdbpedia.
org%2Fontology%2F%3E%0D%0ASELECT+%3Fs+%0D%0AWHERE+{+%3Fs+a+dbpedia-owl%3ADrug+.+
}%0D%0ALIMIT+10&_implicit=false&_equivalent=false&_form=%2Fsparql
```

Why SPARQL?

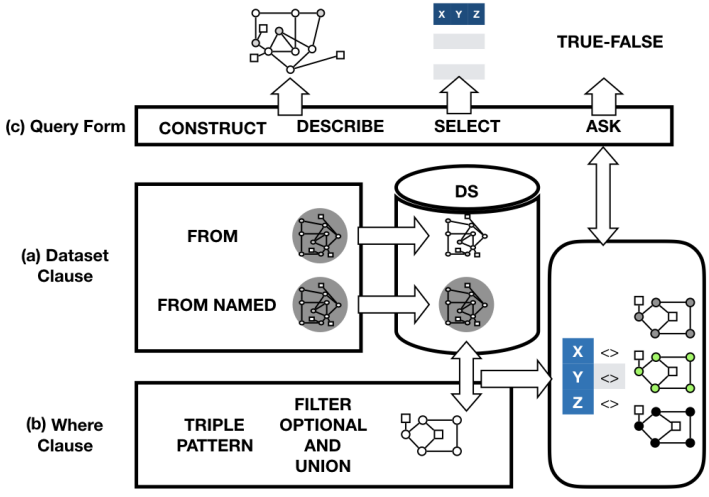
Why SPARQL?

SPARQL lets us

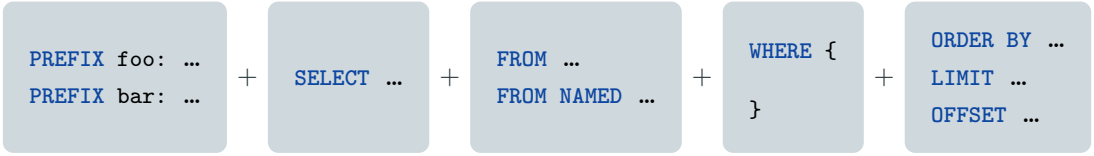
- Pull values from structured and semi-structured data represented in RDF
- Explore RDF data by querying unknown relationships
- Perform complex joins of disparate RDF repositories in a single query
- Transform RDF data from one vocabulary to another
- Develop higher-level cross-platform applications
- It is a W3C standard (since 2008)

Anatomy of a SPARQL Query

[Source: Arenas et al., *On the Semantics of SPARQL*,. 2010]



Anatomy of a SPARQL Query (SELECT)



Declare prefix shortcuts (optional)

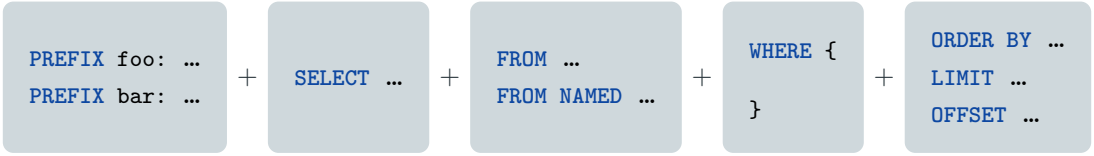
Query form (or result clause)

Define the dataset(s) (optional)

Triple patterns (Graph Pattern)

Query modifiers (optional)

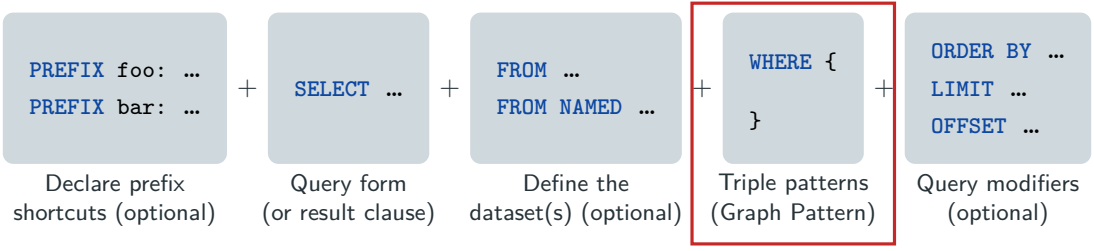
Anatomy of a SPARQL Query (SELECT)



Example (Query: Find Drugs in DBpedia)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?s
FROM <http://dbpedia.org/>
WHERE { ?s a dbpedia-owl:Drug . }
ORDER BY DESC(?s)
```

Anatomy of a SPARQL Query (SELECT)



Example (Query: Find Drugs in DBpedia)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?s
FROM <http://dbpedia.org/>
WHERE { ?s a dbpedia-owl:Drug . }
ORDER BY DESC(?s)
```

Triple Pattern Syntax in WHERE Clauses

Triple Pattern Syntax

- Turtle-like: IRIs, literals, convenience syntax, ...
- In addition: Variables `?var`
  - Variable names are prefixed with a question mark ?
  - Variable names are XML NCNames (can contain a-z and A-Z but not ".", ":", ...)

Example (Simple Triple Pattern)

```
?s rdf:type dbo:Artwork .
```

SPARQL – A Simple Query

Writing a Simple Query

- Query

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s
WHERE { ?s rdf:type dbo:Artwork . }
```

- Result

?s
http://dbpedia.org/resource/St._John_the_Baptist_(Leonardo)
http://dbpedia.org/resource/The_Magpie_(Monet)
...

Try it out at

https://dbpedia.org/sparql

Basic Graph Patterns in SPARQL

- A basic graph pattern is a set of triple patterns, **all of which must match graph**

Example (Basic Graph Pattern)

```
?s rdf:type dbo:Artwork .
?s dbo:author ?o .
?s ?p dbr:Paris .
```

SPARQL – Basic Graph Pattern

Basic Graph Patterns in SPARQL

- A basic graph pattern is a set of triple patterns,
  all of which must match graph
- In this case "matches the graph" means finding a **variable binding** such that
  - the substitution of variables for values in the triples patterns
  - creates triples
  - which are in the database

Example (Basic Graph Pattern)

```
?s rdf:type dbo:Artwork .
?s dbo:author ?o .
?s ?p dbr:Paris .
```

SPARQL – A Bit More Complex Query

Writing a Bit More Complex Query

- Query

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s rdf:type dbo:Artwork .
        ?s dbo:author ?o . }
```

Try it out at
https://dbpedia.org/sparql

- Result

?s	?o
https://dbpedia.org/resource/St._John_the_Baptist_(Leonardo)	https://dbpedia.org/resource/Leonardo_da_Vinci
https://dbpedia.org/resource/The_Magpie_(Monet)	https://dbpedia.org/resource/Claude_Monet
https://dbpedia.org/resource/Fishing_(Carracci)	https://dbpedia.org/resource/Annibale_Carracci
…	…

SPARQL – Query Properties

Writing a Query that Asks for Properties

- Query

```
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?p
WHERE { dbr:Auguste_Rodin ?p dbr:Paris . }
```

- Result

?drug
http://dbpedia.org/ontology/birthPlace
http://dbpedia.org/property/birthPlace
http://dbpedia.org/ontology/wikiPageWikiLink

Try it out at
https://dbpedia.org/sparql

Matching RDF Literals

- The query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?s
WHERE { ?s rdfs:label "The Thinker" . }
```

Matching RDF Literals – Language Tags

Matching RDF Literals

- The query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?s
WHERE { ?s rdfs:label "The Thinker" . }
```

- returns zero(!) results on `https://dbpedia.org/sparql/`
- But the resource `http://dbpedia.org/resource/The_Thinker` exists in dbpedia

Matching RDF Literals – Language Tags

Matching RDF Literals

- The query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?s
WHERE { ?s rdfs:label "The Thinker" . }
```

- returns zero(!) results on `https://dbpedia.org/sparql/`
- But the resource `http://dbpedia.org/resource/The_Thinker` exists in dbpedia

Explanation

- All `rdfs:label` in dbpedia have a language tag
- `"The Thinker"` is not the same as `"The Thinker"@en`

Matching RDF Literals – Language Tags

Matching RDF Literals

- The query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?s
WHERE { ?s rdfs:label "The Thinker"@en . }
```

- returns

?s
http://dbpedia.org/resource/The_Thinker

Matching RDF Literals – Types

Matching Typed Literals

- As is the case for language tags, if the literals are typed, they do not match if the types are not specified explicitly.

Example

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s
WHERE { ?s dbo:birthDate "1840-11-12"^^xsd:date . }
```

Matching RDF Literals – Types

Matching Typed Literals

- As is the case for language tags, if the literals are typed, they do not match if the types are not specified explicitly.

Example

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s
WHERE { ?s dbo:birthDate "1840-11-12"^^xsd:date . }
```

Result

?s
http://dbpedia.org/resource/Auguste_Rodin
http://dbpedia.org/resource/Thomas_Collier_(painter)
http://dbpedia.org/resource/William_McKay_Wright
http://dbpedia.org/resource/John_J._Valentine_Sr.

Filtering Results

Filtering Results

Filter Clause

- SPARQL allows restricting answers by applying the `FILTER` clause

Example

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s dbo:birthDate ?o .
        FILTER (?o > "1840-01-01"^^xsd:date && ?o < "1840-12-31"^^xsd:date) .
}
```

Filtering Results

Filter Clause

- SPARQL allows restricting answers by applying the `FILTER` clause

Example

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s dbo:birthDate ?o .
        FILTER (?o > "1840-01-01"^^xsd:date && ?o < "1840-12-31"^^xsd:date) .
}
```

Result

?s	?o
http://dbpedia.org/resource/George_R._Davis	1840-01-03
http://dbpedia.org/resource/Father_Damien	1840-01-03
...	...

Filtering Results

Filter by REGEX

- `FILTER` clauses also allow restricting values of strings using the `regex()` function

Example

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s dbo:birthDate ?o .
        FILTER (regex(?o, "1880")) }
```

Filtering Results

Filter by REGEX

- `FILTER` clauses also allow restricting values of strings using the `regex()` function

Example

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s dbo:birthDate ?o .
        FILTER (regex(?o, "1880")) }
```

Result

?s	?o
http://dbpedia.org/resource/Douglas_MacArthur	1880-1-26
http://dbpedia.org/resource/H._L._Mencken	1880-9-12
...	...

Filtering Matches – Operators

Operators to Filter/Test Values

- Comparison of values: `<`, `>`, `=`, `<=`, `>=` and `!=`
- Test functions: `isIRI`, `isLITERAL`, ...
- Accessing accessories: LANG, DATATYPE
- Comparison of strings: REGEX, langMatches
- Logical operators: `||` and `&&`
- Constructor functions: `bool`, `dbl`, `flt`, `dec`, `int`, `dT`, `str`, `IRI`
- Extensible value testing:
  - For example, `FILTER ( aGeo:distance(?axLoc, ?ayLoc, ?bxLoc, ?byLoc) < 10 ) .`
  - see `http://www.w3.org/TR/rdf-sparql-query/#extensionFunctions`

Composing Graph Patterns

Composing Graph Patterns

- RDF is “semi structured” and has no integrity constrains
- SPARQL addresses this issue with ...

Composing Graph Patterns

- RDF is "semi structured" and has no integrity constrains
- SPARQL addresses this issue with ...

Group Patterns

- ...match if all subpatterns match and all constraints are satisfied
- SPARQL syntax: groups are enclosed in curly braces `{ ... }`

Composing Graph Patterns

- RDF is "semi structured" and has no integrity constrains
- SPARQL addresses this issue with ...

Group Patterns

- ...match if all subpatterns match and all constraints are satisfied
- SPARQL syntax: groups are enclosed in curly braces `{ ... }`

Optional Graph Patterns

- ...accommodate the need to add information to a result but without the query failing just because some information is missing
- SPARQL syntax: `OPTIONAL { ... }`

Composing Graph Patterns

- RDF is "semi structured" and has no integrity constrains
- SPARQL addresses this issue with ...

Group Patterns

- ...match if all subpatterns match and all constraints are satisfied
- SPARQL syntax: groups are enclosed in curly braces `{ ... }`

Optional Graph Patterns

- ...accommodate the need to add information to a result but without the query failing just because some information is missing
- SPARQL syntax: `OPTIONAL { ... }`

Union Graph Patterns

- ...allow to match alternatives
- SPARQL syntax: `{ ... } UNION { ... }`

Composing Graph Patterns – Example

Example (Optional Graph Pattern)

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s a dbo:Artwork .
      OPTIONAL { ?s dbp:heightMetric ?o }
}
```

Composing Graph Patterns – Example

Example (Optional Graph Pattern)

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s a dbo:Artwork .
      OPTIONAL { ?s dbp:heightMetric ?o }
}
```

Result

?s	?o
http://dbpedia.org/resource/The_Death_of_Actaeon	178.4
http://dbpedia.org/resource/Bust_of_Amenemhat_V	
...	...

Composing Graph Patterns – Example

Example (Union of Graph Patterns)

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s a dbo:Artwork
      { ?s dbo:nearestCity ?o } UNION { ?s dbp:city ?o }
}
```

Composing Graph Patterns – Example

Example (Union of Graph Patterns)

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s a dbo:Artwork
      { ?s dbo:nearestCity ?o } UNION { ?s dbp:city ?o }
}
```

Result

?s	?o
http://dbpedia.org/resource/Andrews_Geyser	http://dbpedia.org/resource/Asheville
...	...
http://dbpedia.org/resource/All_Is_Well_(sculpture)	"Salt Lake City, Utah, U.S."@en

Subqueries

Example

```
PREFIX dbpp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?drug ?mp
WHERE {
  ?drug a dbo:Drug .
  ?drug dbpp:meltingPoint ?mp .
  {
    SELECT (MIN(?m) AS ?mm)
    WHERE { ?d dbpp:meltingPoint ?m }
  }
  FILTER (?mp = ?mm)
}
```

Result

?drug	?mp
http://dbpedia.org/resource/Diisopropyl_fluorophosphate	−82

Negation

Negation

- two forms – stylistic and conceptual differences
- Nearly the same, but not quite

`MINUS` clause

- absence of a pattern
- based on testing whether a pattern exists in the data given the bindings already determined by the query pattern

`FILTER NOT EXISTS` clause

- Removing rows from a result set
- Removes matches based on the evaluation of two patterns

Negation – The MINUS clause

Example

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbcat: <http://dbpedia.org/resource/Category:>
SELECT ?drug
WHERE { ?drug a dbo:Drug .
  ?drug dcterms:subject dbcat:Anxiolytics .
  MINUS { ?drug dbp:excretion dbpedia:Kidney }
}
```

Result

?drug
http://dbpedia.org/resource/Eglumegad
http://dbpedia.org/resource/Oxaflozane
...

Negation – NOT EXISTS

Example

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbcat: <http://dbpedia.org/resource/Category:>
SELECT ?drug
WHERE { ?drug a dbo:Drug .
  ?drug dcterms:subject dbcat:Anxiolytics .
  FILTER NOT EXISTS { ?drug dbp:excretion dbpedia:Kidney }
}
```

Result

?drug
http://dbpedia.org/resource/Eglumegad
http://dbpedia.org/resource/Oxaflozane
...

Negation – Difference between Negation Clauses

RDF Data

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

Negation via `MINUS` clause

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

Negation via `FILTER NOT EXISTS` clause

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

Negation – Difference between Negation Clauses

RDF Data

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

Negation via `MINUS` clause

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

Result

?s	?p	?o
:a	:b	:c

- there is no shared variable between the first part (`?s ?p ?o`) and the second (`?x ?y ?z`)
- Hence, no bindings are eliminated

Negation via `FILTER NOT EXISTS` clause

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

Result

- 0 answers
- `{ ?x ?y ?z }` matches given any `?s ?p ?o`
- Hence, `NOT EXISTS { ?x ?y ?z }` eliminates any solutions

Navigational Queries with Property Paths

Property Paths

Matching Variable-length Path with Property Paths

- Variable-length paths can be matched using property paths
- Property paths are an extended form of 2RPQs

Property Paths

Matching Variable-length Path with Property Paths

- Variable-length paths can be matched using property paths
- Property paths are an extended form of 2RPQs

Example (Finding Recommendations)

Finding posts which Alex might like

```
PREFIX : <http://example.org/social-network/>
SELECT ?p
WHERE { :Alex :knows+/:likes ?p . ?p rdf:type :Post . }
```

Property Paths – Syntax

[https://www.w3.org/TR/sparql11-query/#propertypaths]

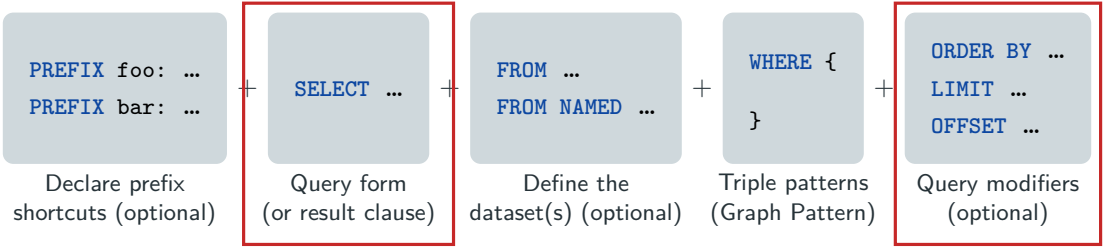
Path Constructs and Operators

- The basic building block are IRIs (paths of length one)
- If p_1 and p_2 are path constructs, then so are
  - $\hat{}p_1$ (Inverse path: from object to subject)
  - p_1/p_2 (Concatenation of paths)
  - $p_1|p_2$ (Alternative/Disjunction)
  - p_1* (Kleene-star, zero or more repetitions)
  - p_1+ (One or more repetitions)
  - $p_1?$ (Optional)
- If $i_1, \ldots, i_n$ are IRIs, then
  - $!i_1$ and $!(i_1|\ldots|i_n)$; (Negation)
  - $!\hat{}i_1$ and $!(\hat{}i_1|\ldots|\hat{}i_n)$; (Negation of inverse edges)
  - $!(i_1|\ldots|i_j|\hat{}i_{j+1}|\ldots|\hat{}i_n)$ (Negation of forward and inverse edges)

  are path constructs

Query Modifiers

Anatomy of a SPARQL Query (SELECT)



Example (Query: Find Drugs in DBpedia)

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?s
FROM <http://dbpedia.org/>
WHERE { ?s a dbpedia-owl:Drug . }
ORDER BY DESC(?s)
```

Obtaining Ordered Results

ORDER BY Clauses

- Answers can be returned ordered using `ORDER BY`

Example

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s a dbo:Artwork . ?s dbp:heightMetric ?o }
ORDER BY DESC(?o)
```

Result

?s	?o
http://dbpedia.org/resource/White_Crucifixion	15462
http://dbpedia.org/resource/The_Death_of_Germanicus	14796
...	...

Limiting the Number of Answers

LIMIT Clauses

- The number of answers in the result can be limited using `LIMIT`

Example

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?s ?o
WHERE { ?s a dbo:Artwork . ?s dbp:heightMetric ?o }
ORDER BY DESC(?o)
LIMIT 2
```

Result

?s	?o
http://dbpedia.org/resource/White_Crucifixion	15462
http://dbpedia.org/resource/The_Death_of_Germanicus	14796

Obtaining Unique Answers

The DISTINCT Keyword

- Results without duplicates can be obtained using `DISTINCT`

Example

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?p
WHERE { ?s a dbo:Artwork .
        ?s ?p ?o
}
```

Note

- Without the `DISTINCT` clause, the above query returns each properties as many times as it is used to describe artworks

Aggregations

Aggregates

The built-in aggregate expressions are

- `AVG(expr)`
- `COUNT(*)` and `COUNT(expr)`
- `GROUP_CONCAT(expr)` with optional `;separator = "string"`
- `MAX(expr)`
- `MIN(expr)`

Aggregations

Aggregates

The built-in aggregate expressions are

- `AVG(expr)`
- `COUNT(*)` and `COUNT(expr)`
- `GROUP_CONCAT(expr)` with optional `;separator = "string"`
- `MAX(expr)`
- `MIN(expr)`

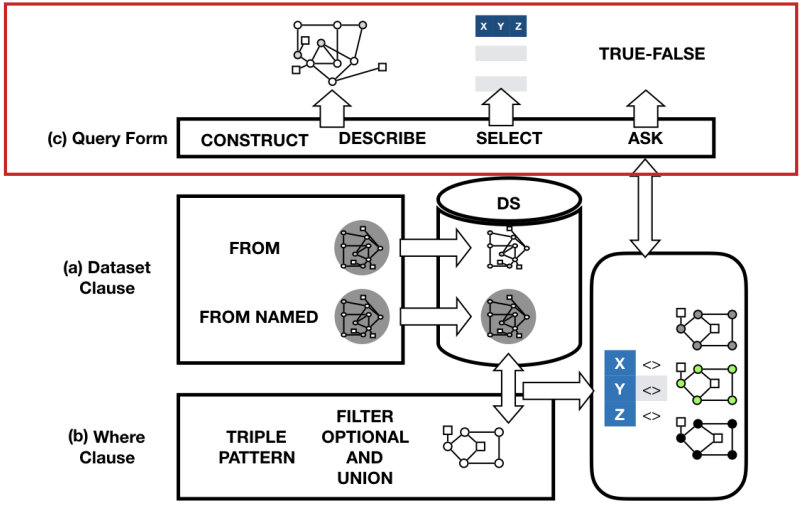
Example

```
PREFIX dbpp: <http://dbpedia.org/property/>
SELECT (MIN(?m) AS ?mm)
WHERE { ?d dbpp:meltingPoint ?m }
```

Query Forms

Anatomy of a SPARQL Query

[Source: Arenas et al., *On the Semantics of SPARQL*,. 2010]



Query Forms

Query Forms

- A `SELECT` query always returns a table of values
- SPARQL allows three other types of queries

ASK queries return a Boolean answer (true/false, yes/no)

CONSTRUCT queries use variable bindings to return new RDF triples

DESCRIBE queries return server-determined RDF about the queried resources (meta information)

Query Forms

Query Forms

- A `SELECT` query always returns a table of values
- SPARQL allows three other types of queries

ASK queries return a Boolean answer (true/false, yes/no)

CONSTRUCT queries use variable bindings to return new RDF triples

DESCRIBE queries return server-determined RDF about the queried resources (meta information)

Note

- Results of `SELECT` and `ASK` queries can be returned as XML or JSON
- Results of `CONSTRUCT` and `DESCRIBE` queries can be returned via any RDF serialization (e.g. Turtle or RDF/XML).

Boolean Queries – Example

Example (ASK Query)

```
PREFIX dbo: <http://dbpedia.org/ontology/>
ASK
WHERE { ?s a dbo:Artwork . }
```

Result

The query will return yes if and only if the endpoint knows about artworks (that is, if the pattern is matched at least once)

Constructing Results in RDF

Example (CONSTRUCT Query)

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX : <http://ex.org/>
CONSTRUCT { ?m :exposes ?s }
WHERE { ?s a dbo:Artist .
        ?s dbp:works ?w .
        ?w dbo:museum ?m . }
```

Result

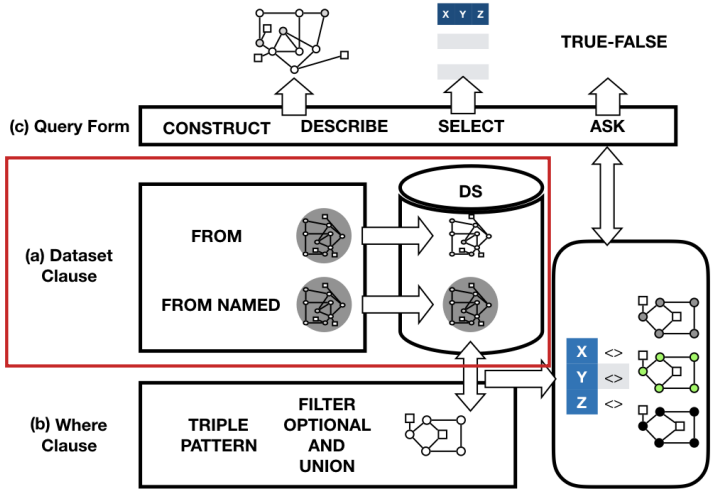
The query construct a new triple for each pair of museum and artist

- The new triples can be added back to the dataset reducing its variety

Dataset Clauses

Anatomy of a SPARQL Query

[Source: Arenas et al., *On the Semantics of SPARQL*,. 2010]



Multiple Graphs and the Dataset

Specifying the Dataset and RDF Graph(s)

- All the queries we have seen so far have operated on a single RDF graph
  - Namely, the default graph
  - If the query does not specify it, is up to the query engine/repository, what the default graph is
- A SPARQL query actually runs on an RDF dataset which may consist of multiple RDF graphs
- RDF graphs are identified, like everything else, by IRIs

Specifying the Default Graph

Specifying the Dataset

- The default graph can be specified explicitly using the `FROM` clause

Example

```
PREFIX dbpprop: <http://dbpedia.org/property/>
SELECT ?drug ?mp
FROM <http://factforge.net/dbpedia>
WHERE { ?drug dbpprop:meltingPoint ?mp }
```

Specifying the Default Graph

Specifying the Dataset

- The default graph can be specified explicitly using the `FROM` clause

Example

```
PREFIX dbpprop: <http://dbpedia.org/property/>
SELECT ?drug ?mp
FROM <http://factforge.net/dbpedia>
WHERE { ?drug dbpprop:meltingPoint ?mp }
```

Result

?drug	?mp
http://dbpedia.org/resource/Olanzapine	195
...	...

Specifying the Default Graph

Specifying the Dataset

- The default graph can be specified explicitly using the `FROM` clause

Example

```
PREFIX dbpprop: <http://dbpedia.org/property/>
SELECT ?drug ?mp
FROM <http://factforge.net/dbpedia>
WHERE { ?drug dbpprop:meltingPoint ?mp }
```

Note

- If the dataset is not in the RDF repository
  the SPARQL engine will try to download it from the Web
- If it is in the repository it will limit the scope of the query to the declared graph

Combining Multiple Graphs in the Dataset

Combine Several Graphs in the Dataset

- Using multiple `FROM` clauses, several graphs can be combines in a dataset

Example

```
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
SELECT ?drug
FROM <http://dbpedia.org>
FROM <http://factforge.net/dbpedia>
WHERE { ?drug dbpprop:metabolism dbpedia:Kidney }
```

Combining Multiple Graphs in the Dataset

Combine Several Graphs in the Dataset

- Using multiple `FROM` clauses, several graphs can be combines in a dataset

Example

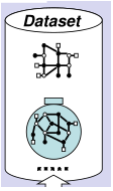
```
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
SELECT ?drug
FROM <http://dbpedia.org>
FROM <http://factforge.net/dbpedia>
WHERE { ?drug dbpprop:metabolism dbpedia:Kidney }
```

Result

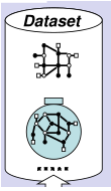
?drug
http://dbpedia.org/resource/Calcitriol
...

Named Graphs

- The dataset can be composed of
  - one (optional) default graph
  - any number of named graphs



Named Graphs



- The dataset can be composed of
  - one (optional) default graph
  - any number of named graphs
- The `FROM` keyword specifies the default graph
  - If several `FROM` clauses are used, the specified graphs are merged to create the default graph
- The `FROM NAMED` keyword adds named graphs to the dataset
- The `GRAPH` keyword is required to match patterns in a named graph by explicitly stating which graph they must match in

- Warning: If the query contains `FROM NAMED` clauses, but no `FROM` clause, the default graph is empty (but some Endpoints/Webinterfaces always add a `FROM` clause to the query, for instance `https://dbpedia.org/sparql`)

Named Graphs – Example

Example

```
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
SELECT ?drug
FROM <http://dbpedia.org>
FROM NAMED <http://factforge.net/dbpedia>
WHERE { ?drug a dbpedia-owl:Drug .
        { ?drug dbpprop:metabolism dbpedia:Kidney . }
        UNION
        { GRAPH <http://factforge.net/dbpedia>
                { ?drug dbpprop:excretion dbpedia:Kidney . } } }
```

Result

?drug
http://dbpedia.org/resource/Calcitriol
...

Abbreviation Graph Names Using Prefixes

Example

```
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX ff: <http://factforge.net/>
SELECT ?drug
FROM <http://dbpedia.org>
FROM NAMED ff:dbpedia
WHERE { ?drug a dbpedia-owl:Drug .
        { ?drug dbpprop:metabolism dbpedia:Kidney . }
        UNION
        { GRAPH ff:dbpedia
                { ?drug dbpprop:excretion dbpedia:Kidney . } } }
```

Result

?drug
http://dbpedia.org/resource/Calcitriol
...

Beyond the Nutshell

Federated Queries

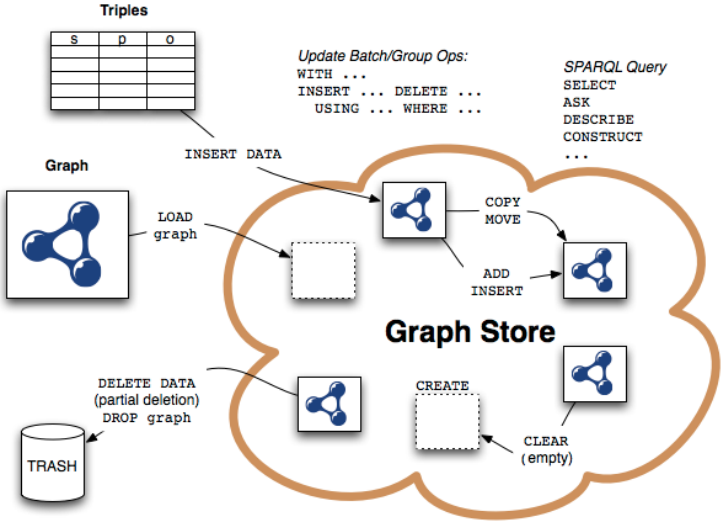
Basic Federated Queries

A graph pattern that

- invokes a SPARQL protocol call and
- remote query returning the usual result formats

```
SERVICE <endpoint-uri> { graph pattern }
```

Updates and Data Integration



SPARQL Resources

- SPARQL implementations – community maintained list of open source and commercial SPARQL engines and tools
  - `https://www.w3.org/wiki/SparqlImplementations`
- Public SPARQL endpoints – community maintained list
  - `https://www.w3.org/wiki/SparqlEndpoints`
- SPARQL extensions – collection of SPARQL extensions implemented in various SPARQL engines
  - `https://www.w3.org/wiki/SPARQL/Extensions`

Credits

This presentation is adapted from and contains slides by
E. Della Valle – http://emanueledellavalle.org – @manudellavalle

It contains slides adapted from

- www.dcs.warwick.ac.uk/~acristea/courses/CS411/2010/SPARQL.ppt, and
- http://www.emse.fr/~zimmermann/Teaching/WebSem/SlidesSPARQL.html

References

 Arenas, Marcelo, Claudio Gutierrez, and Jorge Pérez (2010). *On the Semantics of SPARQL*. Semantic Web Information Management: A Model Based Perspective. Springer. Chap. 1.