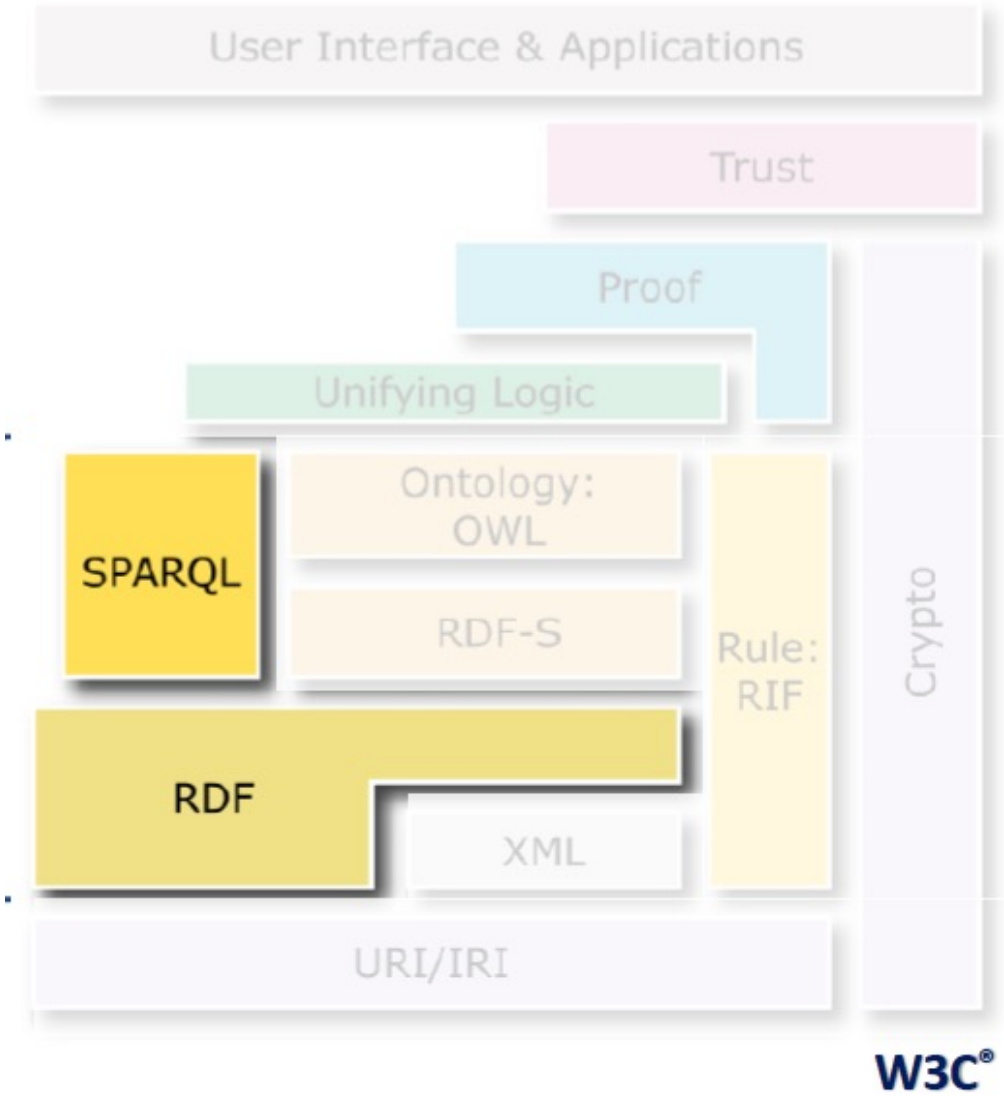
SPARQL
Query Language for RDF

Introduction

The Semantic Web Standard Stack

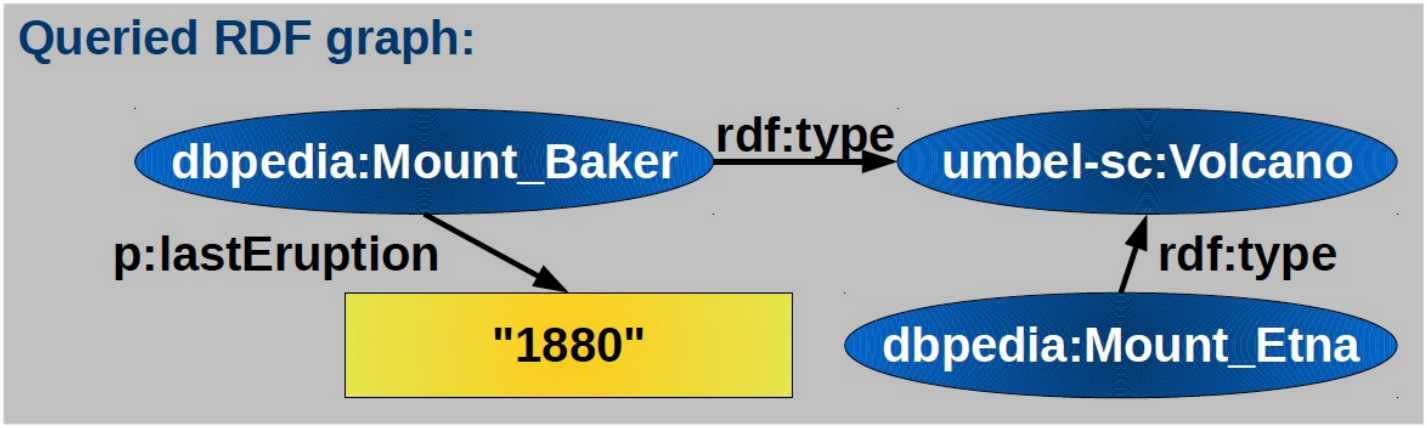


SPARQL

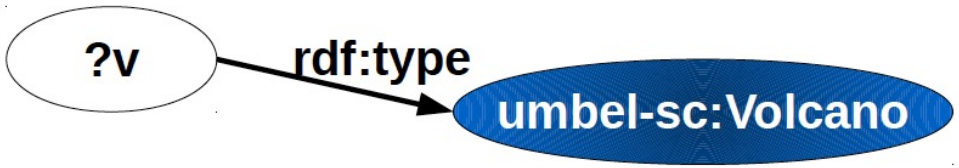
- Query language for getting information from RDF graphs
- It provides facilities to:
  - extract information in the form of URIs, blank nodes, plain and typed literals
  - extract RDF subgraphs
  - construct new RDF graphs based on information in the queried graphs
- Query terms and data description format: Turtle

Main idea of SPARQL

- SPARQL queries are defined starting from triple patterns subject-property-object
  - similar to an RDF Triple (subject, predicate, object), but any component can be a query variable (denoted by ?<name> where <name> is the variable name)



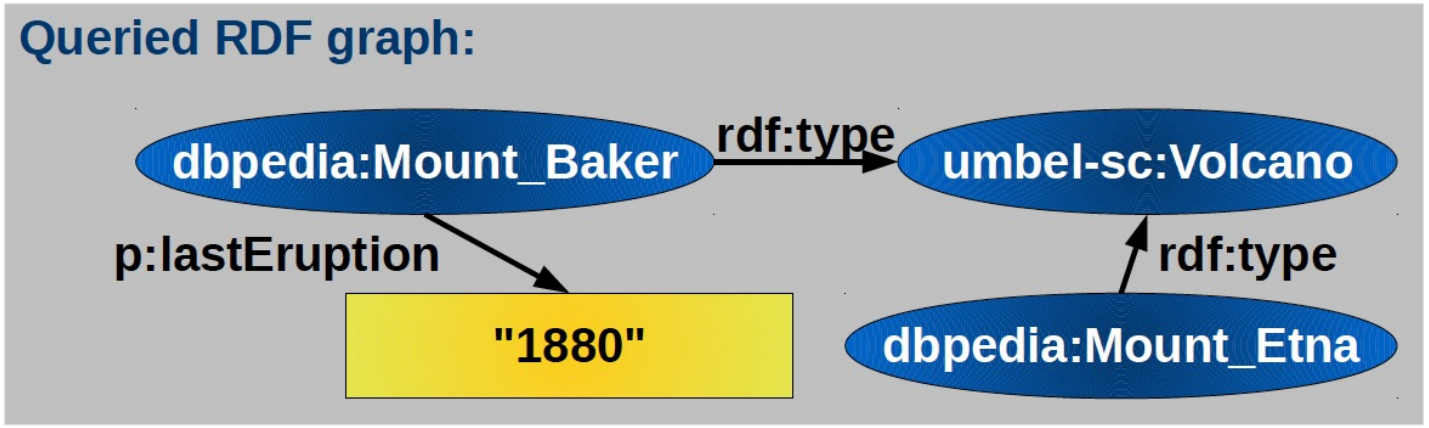
Variable

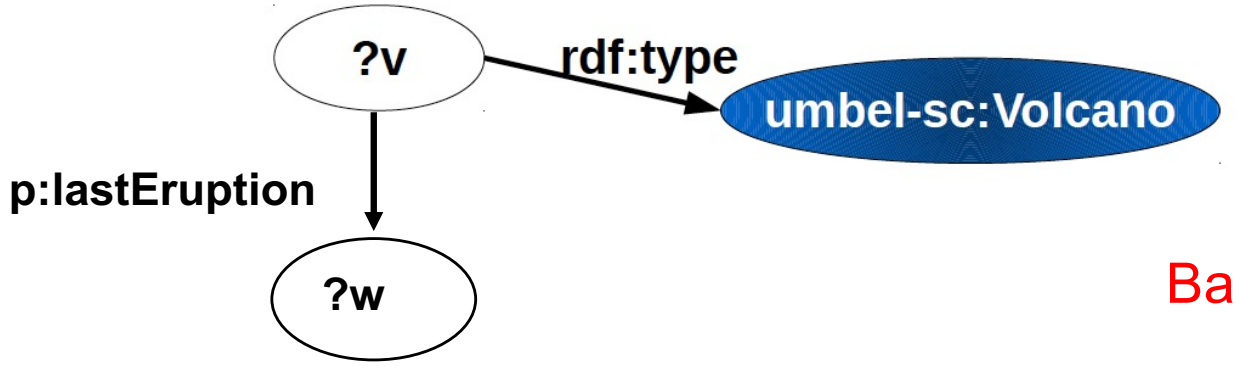


Triple pattern

Main idea of SPARQL

- Combining triple patterns gives a basic graph pattern (RDF graphs with variables)





Basic graph pattern

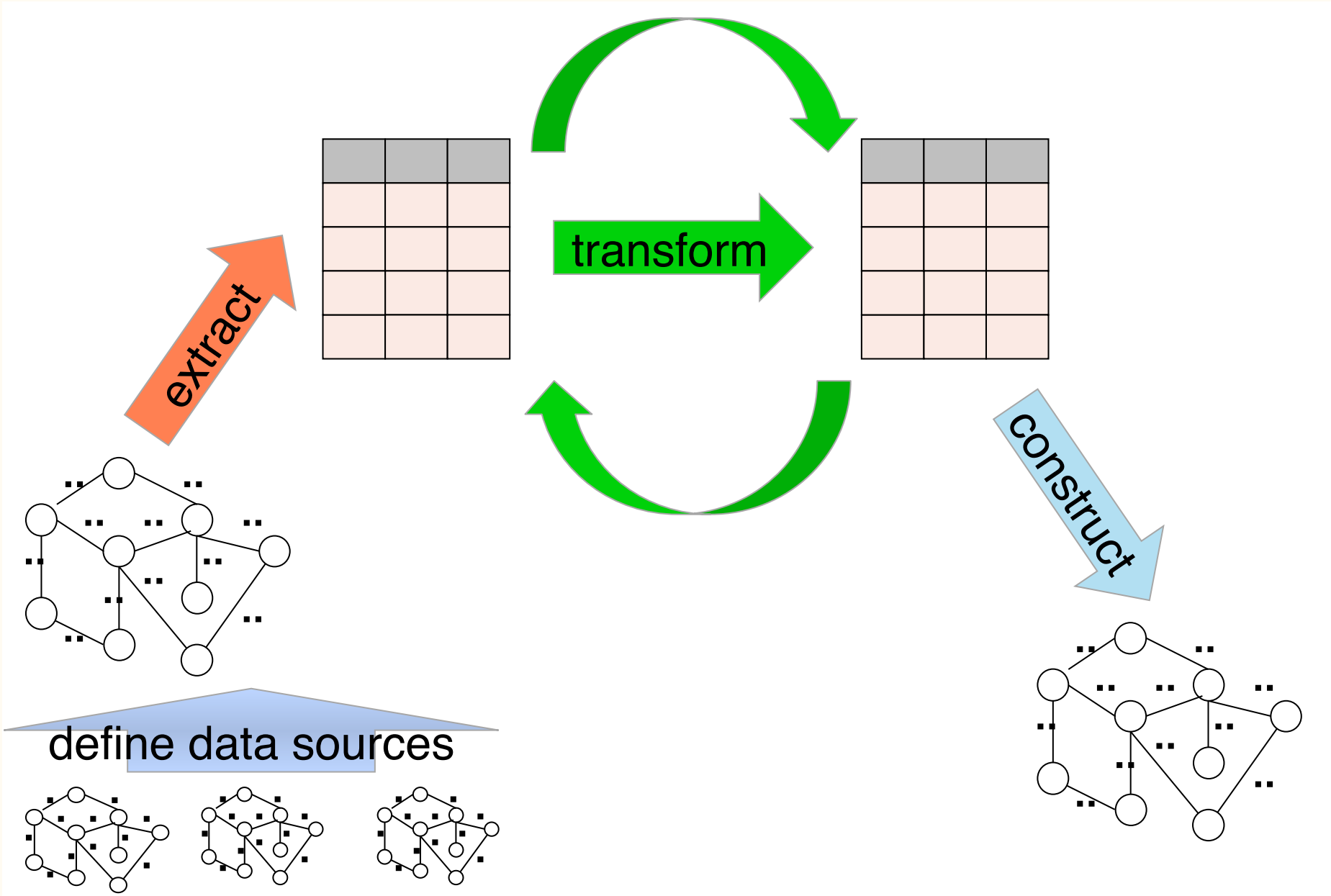
Main idea of SPARQL

- Matching a basic graph pattern to an RDF graph:
  - bindings between variables in graph pattern and RDF Terms (either resources or literals)

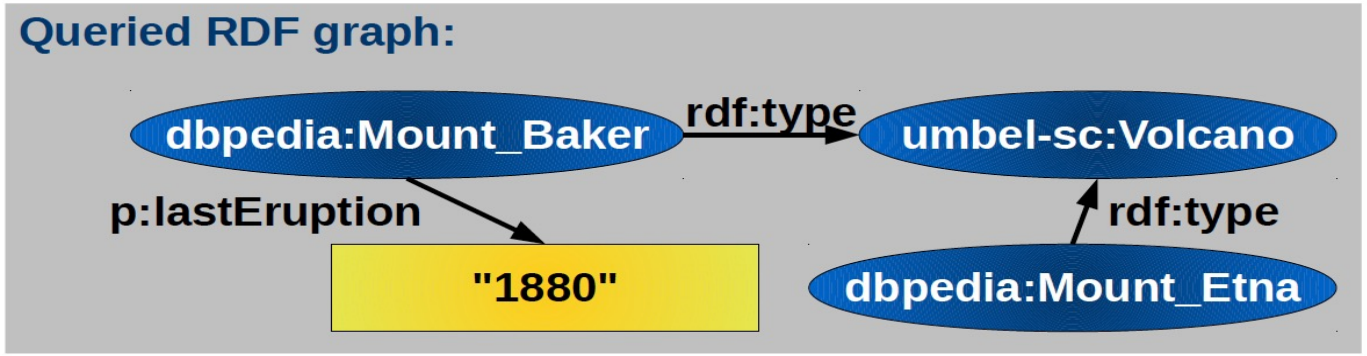
- A Pattern Solution of a Basic Graph Pattern GP on an RDF graph G is any substitution S of variables with RDF terms such that S(GP) is a subgraph of G

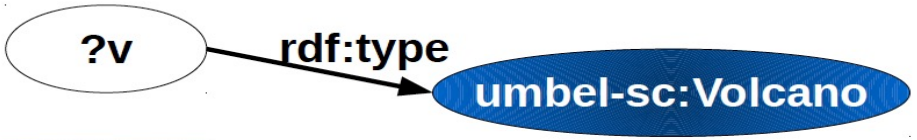
- The results of SPARQL queries can be either a tabular result set or an RDF graph

SPARQL is a Graph Query Language



Example





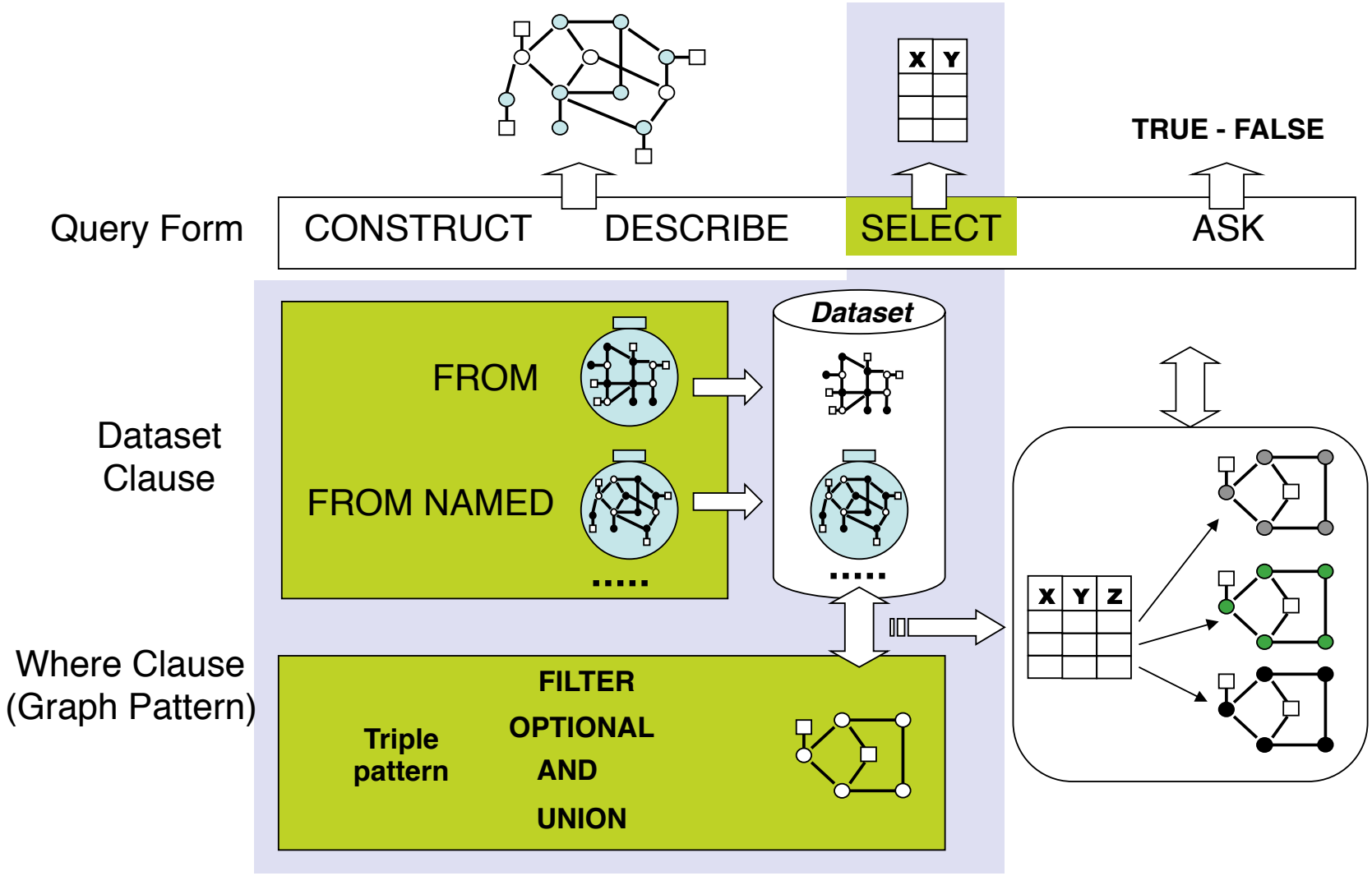
Result:

?v
dbpedia:Mount_Baker
dbpedia:Mount_Etna

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
SELECT ?v
FROM <http://example.org/myGeoData>
WHERE {
  ?v rdf:type umbel-sc:Volcano .
}
ORDER BY ?v
```

Main SPARQL query forms

SPARQL Query



Query forms

- SPARQL has four query forms. These query forms use the solutions from pattern matching to return either a result set or an RDF graph
- SELECT
  - Returns all, or a subset of, the variables bounds by pattern matching
- CONSTRUCT
  - Returns an RDF graph constructed by substituting variables with the corresponding solution terms in a set of triple templates
- ASK
  - Returns a boolean indicating whether a query pattern matches or not the input RDF dataset
- DESCRIBE
  - Returns an RDF graph that represents all the triples describing the resources found

SELECT

SELECT specifies the projection: the number and the order of retrieved data

FROM is used to specify the source (the RDF dataset) being queried (optional, if not specified a default RDF dataset is considered)

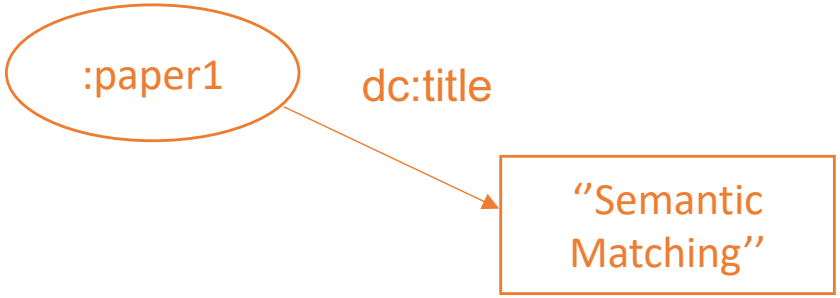
WHERE imposes constraints on solutions in form of graph pattern templates and boolean constraints

Data

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

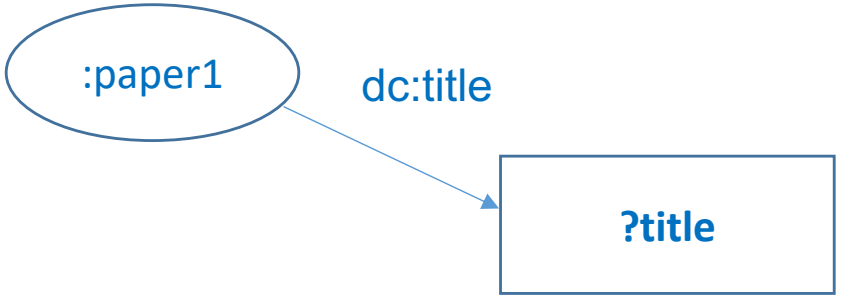
@prefix : <http://example.org/book/> .

:paper1 dc:title "Semantic Matching"



Query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
FROM <http://example.org/book/>
WHERE { :paper1 dc:title ?title . }
```



title
"Semantic Matching"

SELECT (multiple matches and variables)

Data

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

_:a foaf:name "Tim Berners-Lee" .

_:a foaf:homepage <http://www.w3.org/People/Berners-Lee/> .

_:b foaf:name ”Barbara Catania" .

_:b foaf:homepage <http://www.dibris.unige.it/catania-barbara> .

Query

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

SELECT ?name ?homepage

WHERE { ?x foaf:name ?name .

?x foaf:homepage ?homepage . }

Result

name	homepage
Tim Berners-Lee	<http://www.w3.org/People/Berners-Lee/>
Barbara Catania	<http://www.dibris.unige.it/catania-barbara>

SELECT (expressions)

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, (?p*(1-?discount) AS ?price)
WHERE
{ ?x ns:price ?p .
  ?x dc:title ?title .
  ?x ns:discount ?discount .
 }
```

Result

title	price
"The Semantic Web"	17.25
"SPARQL Tutorial"	33.6

SELECT (filtering, term restriction)

FILTER specifies how solutions are restricted to those RDF terms which match with the filter expression (only the bindings satisfying the FILTER condition are returned)

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 ?p
WHERE
{ ?x ns:price ?p .
  ?x dc:title ?title .
FILTER (?p >= 25)
}
```

Result

title	p
"The Semantic Web"	42

Joins

(Implicit join) Retrieve all the titles and the associated prices

```
SELECT  ?title ?p
WHERE
{ ?x ns:price ?p .
  ?x dc:title ?title .
}
```

(Explicit join) Retrieve all the titles and the associated prices

```
SELECT  ?title ?p
WHERE
{ ?x ns:price ?p .
  ?Y dc:title ?title .
   FILTER (?x = ?y)
}
```

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	“Alice”
_:d	“Bob”

Alternative Graph Patterns

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" .
_:b dc11:title "SPARQL Protocol Tutorial" .
_: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 ?x ?y
WHERE { { ?book dc10:title ?x } UNION { ?book dc11:title ?y } }
```

Result

x	y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

CONSTRUCT

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

- **The result.** The result is an RDF graph formed by
  - taking each query solution in the solution sequence,
  - substituting for the variables in the graph template,
  - combining the triples into a single RDF graph by set union
- **Ignored triples.** If any such instantiation produces a triple containing an unbound variable or an illegal RDF construct (e.g., a literal in subject or predicate position) then that triple is not included in the output RDF graph
- **Ground triples.** The graph template can contain triples with no variables (known as ground or explicit triples), and these also appear in the output RDF graph returned by the CONSTRUCT query form

CONSTRUCT

Data

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

_:a foaf:name "Alice" .

_:a foaf:mbox <mailto:alice@example.org> .

Query

PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>

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

CONSTRUCT { <http://example.org/person#Alice> vcard:FN **?name** }

WHERE { **?x** foaf:name **?name** }

Result

It creates vcard properties from the FOAF information:

@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
<http://example.org/person#Alice> vcard:FN "Alice" .

ASK

- Applications can use the ASK form to test whether or not a query pattern has a solution.
- No information is returned about the possible query solutions, just whether or not a solution exists (a Boolean value)

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/>
ASK  { ?x foaf:name "Alice" }
```

Result

```
true
```

DESCRIBE

The DESCRIBE form returns a single RDF graph containing RDF data about resources

- The query pattern is used to create a result set
- The DESCRIBE form takes each of the resources identified in a solution and returns a "description" of such resources, in the form of an RDF graph, which can come from any information available including the target RDF Dataset
- The description is determined by the query service
- The syntax DESCRIBE * is an abbreviation that describes all of the variables in a query.

Query 1

```
DESCRIBE <http://example.org/>
```

Query 2

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

Other clauses and modifiers

OPTIONAL

OPTIONAL allows binding variables to RDF terms to be included in the solution in case of availability (basically it allows for empty cells in the result set)

Data

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

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

_:a dc:creator "Tim Berners-Lee" .

_:a foaf:age 53 .

_:a foaf:homepage <http://www.w3.org/People/Berners-Lee/> .

_:b dc:creator “Alice Smith" .

Query

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

SELECT **?author ?age**

WHERE { **?x** dc:creator **?author** .

OPTIONAL {**?x** foaf:age **?age**}}

Result

author	Age
"Tim Berners-Lee"	53
“Alice Smith"	

ORDER BY

ORDER BY is a facility to order a solution sequence

Data

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

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

_:a dc:creator ” Alice Smith" .

_:a foaf:age 48 .

_:b dc:creator "Tim Berners-Lee" .

_:b foaf:age 53.

Query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?author
 WHERE { ?x dc:creator ?author .
         ?x foaf:age ?age }
ORDER BY ?author DESC
```

Result

author
"Tim Berners-Lee"
“Alice Smith"

DISTINCT modifier

The DISTINCT solution modifier eliminates duplicate solutions. Only one solution that binds the same variables to the same RDF terms is returned from the query.

Data

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

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

_:a dc:creator ”Alice Smith" .

_:a foaf:age 22 .

_:b dc:creator ”Alice Smith" .

_:b foaf:age 48.

Result

creator
“Alice Smith"

Query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?creator
 WHERE { ?x dc:creator ?creator}
```

LIMIT

The LIMIT clause puts an upper bound on the number of solutions returned. If the number of actual solutions, after OFFSET is applied, is greater than the limit, then at most the limit number of solutions will be returned. A LIMIT of 0 would cause no results to be returned.

Data

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

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

_:a dc:creator ”Alice Smith" .

_:a foaf:age 48 .

_:b dc:creator "Tim Berners-Lee" .

_:b foaf:age 53.

Query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT ?author
 WHERE { ?x dc:creator ?author }
ORDER BY ?author  LIMIT 1
```

Result

auhor
”Alice Smith"

OFFSET

The OFFSET clause causes the solutions generated to start after the specified number of solutions. An OFFSET of zero has no effect.

Data

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

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

_:a dc:creator ”Alice Smith" .

_:a foaf:age 48 .

_:b dc:creator "Tim Berners-Lee" .

_:b foaf:age 53.

Query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?author
WHERE { ?x dc:creator ?author }
ORDER BY ?author OFFSET 1
```

Result

auhor
"Tim Berners-Lee"

Additional features

- **New features of SPARQL 1.1 Query:**
  - Aggregate functions (e.g. COUNT, SUM, AVG)
  - Subqueries
  - Negation (EXISTS, NOT EXISTS, MINUS)
  - Assignments (e.g. BIND, SELECT expressions)
  - Property paths
  - Basic query federation (SERVICE, BINDINGS)
- **SPARQL 1.1 Update:**
  - Graph update (INSERT DATA, DELETE DATA, INSERT, DELETE, DELETE WHERE, LOAD, CLEAR)
  - Graph management (CREATE, DROP, COPY, MOVE, ADD)