SPARQL

Fausto Giunchiglia and Mattia Fumagallli

University of Trento

Roadmap

- Introduction
- Basic query forms
  - SELECT
  - CONSTRUCT
  - ASK
  - DESCRIBE
- Other clauses and modifiers
- SPARQL Federated Query

- Exercises

Introduction

SPARQL

What is SPARQL

- A language for expressing queries to retrieve information from various datasets represented in RDF [SPARQL Spec.]
- A query language with the capability to search graph patterns [SPARQL Spec.]

Queries

- SPARQL queries typically contain triple graph patterns: subject-property-object
- Combining triple patterns gives a basic graph pattern, where an exact match to a graph is needed to fulfill a pattern
- RDF terms in each pattern can be substituted with variables

Results

- The results of SPARQL queries can be results sets or RDF graphs

IRIs and URIs

- An URI (Uniform Resource Identifier) includes a subset of the ASCII character set
- An IRI (Internationalized Resource Identifier) can include UNICODE characters

Turtle

What is Turtle

- A terse RDF triple language
- A textual syntax for RDF that facilitates writing RDF graphs
  - in a compact and natural language text form
  - with abbreviations for common usage patterns and datatypes
  - compatible with triple pattern syntax of SPARQL (and N-Triples)

Triple lists

- A triple is a sequence of (subject, property, object) terms separated by whitespace
- Each triple is terminated by dot '.' after each triple

```
<http://www.w3.org/…/Weaving/>
<http://purl.org/dc/elements/1.1/creator>
<http://www.w3.org/People/Berners-Lee> .
```

- In compact form, subsequent triples referring to the same subject are separated by semicolon ';'

```
<http://www.w3.org/…/Weaving>
<http://purl.org/dc/elements/1.1/creator>
<http://www.w3.org/People/Berners-Lee> ;
<http://purl.org/dc/elements/1.1/title> "Weaving the Web".
```

Datasets in Turtle syntax

RDF DATASET

NEW GRAPH

IRI

EXTERNAL NAMED GRAPHS

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<#green-goblin>
   rel:enemyOf <#spiderman> ;
   a foaf:Person ;    # in the context of the Marvel universe
   foaf:name "Green Goblin" .

<#spiderman>
   rel:enemyOf <#green-goblin> ;
   a foaf:Person ;
   foaf:name "Spiderman", "Человек-паук"@ru .
```

RELATIVE (to the current dataset) IRI

RELATIVE STATEMENT

LITERAL

LANGUAGE TAG

Datasets in Turtle syntax

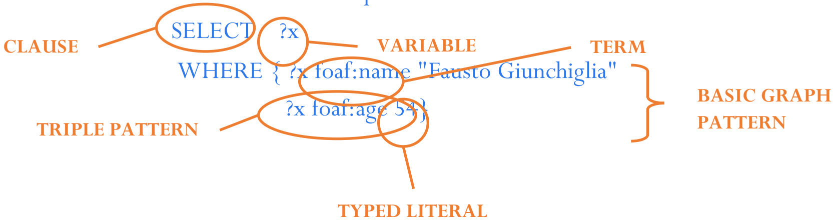
```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Tim Berners-Lee" .
_:a foaf:homepage <http://www.w3.org/People/Berners-Lee/> .
_:b foaf:name "Fausto Giunchiglia" .
_:b foaf:homepage <http://disi.unitn.it/~fausto/> .
_:b foaf:age 54 .
```

BLANK NODE

BLANK NODE

SPARQL Query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT   ?x
 WHERE { ?x foaf:name "Fausto Giunchiglia"
         ?x foaf:age 54}
```



Example of SPARQL query

Retrieve all classes from the RDF data

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?c
WHERE
{
        ?c rdf:type rdfs:Class .
}
```

Here the basic graph pattern is constituted by one triple pattern where:

-the **subject** is given by the variable ?c

-the **property** is rdf:type

-the **object** is rdfs:Class

Example of SPARQL query (II)

Retrieve all instances of the class “course”

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uni: <http://www.mydomain.org/uni-ns#>
SELECT ?c
WHERE
{
        ?c rdf:type uni:course .
}
```

Here the basic graph pattern is constituted by one triple pattern where:

-the **subject** is given by the variable ?c

-the **property** is rdf:type

-the **object** is uni:course

Basic query forms

Query forms

SPARQL has four query forms. These query forms use the solutions from pattern matching to form result sets or RDF graphs. The query forms are:

SELECT

Returns all, or a subset of, the variables bound in a query pattern match

CONSTRUCT

Returns an RDF graph constructed by substituting variables in a set of triple templates

ASK

Returns a boolean indicating whether a query pattern matches or not

DESCRIBE

Returns an RDF graph that describes the resources found

SELECT

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

FROM is used to specify the source being queried (optional)

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       "The Semantic Web"
```

Query

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

Result

title
"The Semantic Web"

SELECT (multiple matches)

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 "Fausto Giunchiglia" .
_:b foaf:homepage <http://disi.unitn.it/~fausto/> .
```

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/>
Fausto Giunchiglia	<http://disi.unitn.it/~fausto/>

SELECT (multiple variables)

The SELECT returns a result set.

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

Joins

(Implicit join) Retrieve all lecturers and their phone numbers:

```
SELECT ?x ?y
WHERE
{
  ?x rdf:type uni:Lecturer .
  ?x uni:phone ?y .
}
```

(Explicit join) Retrieve the name of all courses taught by the lecturer with ID 949352

```
SELECT ?n
WHERE
{
   ?x rdf:type uni:Course .
   ?x uni:isTaughtBy :949352 .
   ?c uni:name ?n .
   FILTER (?c = ?x) .
}
```

CONSTRUCT

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

- **Triples in the graph**: The result is an RDF graph formed by taking each query solution in the solution sequence, substituting for the variables in the graph template, and combining the triples into a single RDF graph by set union.
- **Unbound variables**: If any such instantiation produces a triple containing an unbound variable or an illegal RDF construct, such as 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 foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT { <http://example.org/person#Alice> vcard:FN ?name }
WHERE {<http://example.org/person#Alice> 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.

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 result 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, together with any resources directly named by an IRI (Internationalized Resource Identifier, written in UNICODE), and assembles a single RDF graph by taking a "description" 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" }
```

DESCRIBE

Query

```
PREFIX ent:  <http://org.example.com/employees#>
DESCRIBE ?x WHERE { ?x ent:employeeId "1234" }
```

Result

```
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix vcard:  <http://www.w3.org/2001/vcard-rdf/3.0> .
@prefix exOrg:  <http://org.example.com/employees#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#>

_:a    exOrg:employeeId    "1234" ;
       foaf:mbox_sha1sum   "bee135d3af1e418104bc42904596fe148e90f033" ;
       vcard:N
        [ vcard:Family      "Smith" ;
          vcard:Given       "John"  ] .
foaf:mbox_sha1sum  rdf:type  owl:InverseFunctionalProperty .
```

Other clauses and modifiers

FILTER: term restriction

FILTER specifies how solutions are restricted to those RDF terms which match with the filter expression

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 .
_:b dc:creator "Fausto Giunchiglia" .
_:b foaf:age 54.
```

Query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT ?author
 WHERE { ?x dc:creator ?author .
                  FILTER regex(?author, “tim“, “i”) . }
```

Result

author
"Tim Berners-Lee"

FILTER: term restriction

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 .
_:b dc:creator "Fausto Giunchiglia" .
_:b foaf:age 54.
```

Query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/> .
SELECT ?author ?age
WHERE { ?x dc:creator ?author .
            ?x foaf:age ?age .
            FILTER (?age > 53) }
```

Result

author	Age
"Fausto Giunchiglia"	54

OPTIONAL

OPTIONAL allows binding variables to RDF terms to be included in the solution in case of availability

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 "Fausto Giunchiglia" .
```

Query

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

Result

author	Age
"Tim Berners-Lee"	53
"Fausto Giunchiglia"	

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 "Fausto Giunchiglia" .
_:a foaf:age 54 .
_: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 (?age)
```

Result

author
"Tim Berners-Lee"
"Fausto Giunchiglia"

DISTINCT modified

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 "Fausto Giunchiglia" .
_:a foaf:age 53 .
_:b dc:creator "Fausto Giunchiglia" .
_:b foaf:age 54.
```

Query

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

Result

creator
"Fausto Giunchiglia"

REDUCED modifier

While the DISTINCT modifier ensures that duplicate solutions are eliminated from the solution set, **REDUCED** simply permits them to be eliminated. The cardinality of the elements in the solution set is at least one and no more than the cardinality without removing duplicates.

Data

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a dc:creator "Fausto Giunchiglia" .
_:b dc:creator "Fausto Giunchiglia" .
_:c dc:creator "Fausto Giunchiglia" .
```

Query

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

Result

creator
"Fausto Giunchiglia"
"Fausto Giunchiglia"

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 "Fausto Giunchiglia" .
_:a foaf:age 54 .
_: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"

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 "Fausto Giunchiglia" .
_:a foaf:age 54 .
_: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  OFFSET 1
```

Result

auhor
"Tim Berners-Lee"

SPARQL Federated Query

SPARQL Federated Query

SPARQL endpoints

- Each endpoint typically contains one (unnamed) slot holding a default graph and zero or more named slots holding named graphs

SPARQL Federated query can be used to issue queries across different data sources if:

- data is stored natively as RDF or data is viewed as RDF via middleware
- queries are executed over different SPARQL endpoints

The SERVICE keyword

- allows the author of a query to direct a portion of the query to a particular SPARQL endpoint
- supports merge and joins of SPARQL queries over data distributed across the Web

Example: query to a remote SPARQL endpoints

Remote http://people.example.org/sparql endpoint:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .
:people15  foaf:name     "Alice" .
:people16  foaf:name     "Bob" .
:people17  foaf:name     "Charles" .
:people18  foaf:name     "Daisy" .
```

Local FOAF file http://example.org/myfoaf.rdf :

```
<http://example.org/myfoaf/I> <http://xmlns.com/foaf/0.1/knows>
<http://example.org/people15> .
```

Query: "find the names of the people I know"

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.org/myfoaf.rdf>
WHERE { <http://example.org/myfoaf/I> foaf:knows ?person .
            SERVICE <http://people.example.org/sparql> { ?person foaf:name ?name . } }
```

Result

name
"Alice"

Example: query to two remote SPARQL endpoint

http://people.example.org/sparql:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .
:people15  foaf:name     "Alice" .
:people16  foaf:name     "Bob" .
:people17  foaf:name     "Charles" .
:people17  foaf:interest
<http://www.w3.org/2001/sw/rdb2rdf/> .
```

http://people2.example.org/sparql:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .
:people15  foaf:knows    :people18 .
:people18  foaf:name     "Mike" .
:people17  foaf:knows    :people19 .
:people19  foaf:name     "Daisy" .
```

Query: find information of the people I know.

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?person ?interest ?known
WHERE {
 SERVICE <http://people.example.org/sparql> {
   ?person foaf:name ?name .
   OPTIONAL { ?person foaf:interest ?interest .
    SERVICE <http://people2.example.org/sparql> {
     ?person foaf:knows ?known . } }
 }
}
```

Result

person	interest	known
"Alice"		
"Bob"		
"Charles"	<http://www.w3.org/2001/sw/rdb2rdf/>	<http://example.org/people1

Exercises

You can try the queries here: http://www.sparql.org/query.html

Exercise 1

Suppose that an RDF model represents information about real world entities of unknown types. The entities can be persons, locations, books, monuments, organizations, etc.

(i) Write a SPARQL query to return all possible information about all kinds of entities.

(ii) Write a SPARQL query that can return at most 5 triples representing information

Solution 1

(i)

```
SELECT ?x ?y ?z
WHERE
      { ?x ?y ?z }
```

(ii)

```
SELECT ?x ?y ?z
WHERE
 { ?x ?y ?z }
 LIMIT 5
```

Exercise 2

Given that an RDF model represents information about books and the model is created using standard vocabularies:

i. Write a SPARQL query that can return the authors of the books. Note that books can be represented as URIs.

ii. Write a SPARQL query that can return the titles and authors of the books.

iii. Write a SPARQL query that can return the titles and the date of publication of the books.

Solution 2

(i)

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

(ii)

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

Solution 2

(iii
)

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

Exercise 3

Given that an RDF model represents information about books and the model is created using standard vocabularies:

i. Write a SPARQL query that returns the authors and publishers of the books for which publisher information is available.

ii. Write a SPARQL query that returns the authors and publishers of the books for which publisher might or might not be (or optionally) available.

Solution 3

(i)

```
PREFIX dc:     <http://purl.org/dc/elements/1.1/>
SELECT?author ?publishingHouse
WHERE
  {   ?book dc:creator ?author.
      ?book dc:title ?bookTitle.
      ?book dc:publisher ?publishingHouse }
```

(ii)

```
PREFIX dc:     <http://purl.org/dc/elements/1.1/>
SELECT?author ?publishingHouse
WHERE
  {   ?book dc:creator ?author.
      ?book dc:title ?bookTitle.
      OPTIONAL {?book dc:publisher ?publishingHouse . } }
```

Exercise 4

Given that an RDF model represents information about books and the model is created using standard vocabularies:

i. Write a SPARQL query that returns the authors of the books in descending order.

ii. Write a SPARQL query that returns the authors of the books whose title starts with "Harry Potter".

iii. Write a SPARQL query that returns the authors of the books whose title contains the term "deathly" or "Deathly" or similar variations.

Solution 4

(i)

```
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
SELECT ?bookTitle ?author
WHERE
 {    ?book dc:creator ?author.
      ?book dc:title ?bookTitle. }
ORDER BY DESC (?author)
```

(ii)

```
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
SELECT ?bookTitle ?author
WHERE
  {   ?book dc:creator ?author.
      ?book dc:title ?bookTitle.
      FILTER regex(?bookTitle, “^Harry Potter") }
```

Solution 4

(iii)

```
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
SELECT ?bookTitle ?author
WHERE
  {   ?book dc:creator ?author.
      ?book dc:title ?bookTitle.
      FILTER regex(?bookTitle, "deathly", "i") }
```

Exercise 5

Given that an RDF model represents information about various entities including books and the model is created using standard vocabularies:

i. Write a SPARQL query that extracts title and author of books and creates another RDF model that is a subset of the original one.

ii. Write a SPARQL query that extracts title, author and publisher (if any) of books and creates another RDF model that is a subset of the original one.

Solution 5

(i)

```
PREFIX dc:    <http://purl.org/dc/elements/1.1/>
CONSTRUCT {?book dc:creator ?author.
           ?book dc:title ?bookTitle}
WHERE
  {  ?book dc:creator ?author.
     ?book dc:title ?bookTitle. }
```

(ii)

```
PREFIX dc:    <http://purl.org/dc/elements/1.1/>
CONSTRUCT {?book dc:creator ?author.
       ?book dc:title ?bookTitle.
       ?book dc:publisher ?pub }
WHERE
  {  ?book dc:creator ?author.
     ?book dc:title ?bookTitle.
       OPTIONAL {?book dc:publisher ?pub}  }
```

References

- SPARQL 1.1 Overview (W3C): http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/
- SPARQL 1.1 Update (W3C): http://www.w3.org/TR/sparql11-update/
- SPARQL Query Language for RDF (W3C): www.w3.org/TR/rdf-sparql-query/
- SPARQL 1.1 Query Language (W3C): http://www.w3.org/TR/sparql11-query/
- SPARQL 1.1 Federated Query (W3C): http://www.w3.org/TR/sparql11-federated-query/
- RDF 1.1 Turtle (W3C): http://www.w3.org/TR/turtle/
- FOAF: http://xmlns.com/foaf/spec/
- G. Antoniou & F. van Harmelen (2004). A Semantic Web Primer (Cooperative Information Systems). MIT Press, Cambridge MA, USA.
- D. Allemang and J. Hendler. Semantic web for the working ontologist: modeling in RDF, RDFS and OWL. Morgan Kaufmann Elsevier, Amsterdam, NL, 2008.