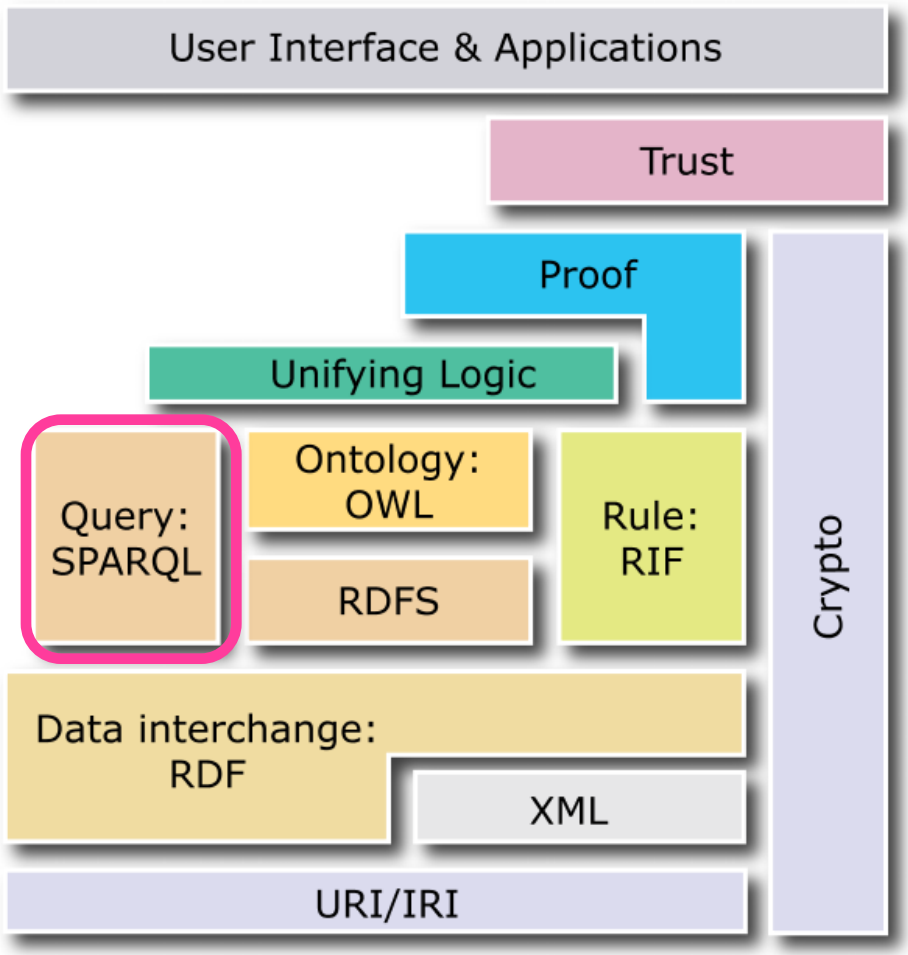
SPARQL

UROŠ KRČADINAC

EMAIL: uros@krcadinac.com

URL: www.krcadinac.com

Semantic Web Layer Cake



What is SPARQL?

- SPARQL - SPARQL Protocol and RDF Query Language
- Enables for query execution over RDF graph or database (triple store)
- SPARQL ≈ SQL

What is SPARQL?

SPARQL enables:

- Value extraction from structured or semi-structured data
- Exploration of data structure
- Joining data coming in as results of a single query executed over different datastores at the same time
- Transforming RDF data from one vocabulary into another

SPARQL in action

SPARQL query defines a pattern of resulting triplets, called **graph pattern**

Graph pattern variables

SPARQL usually starts with symbols **?** or **$** and can represent any part of a triplet (subject, predicate or object).

Graph patterns are the same as RDF triples, just some parts of a triplet are replaced with a variable.

RDF triplet:

```
ex:John          schema:name          “John” .
```

Graph pattern:

```
?person          schema:name          “John” .
```

Types of SPARQL queries

SELECT

ASK

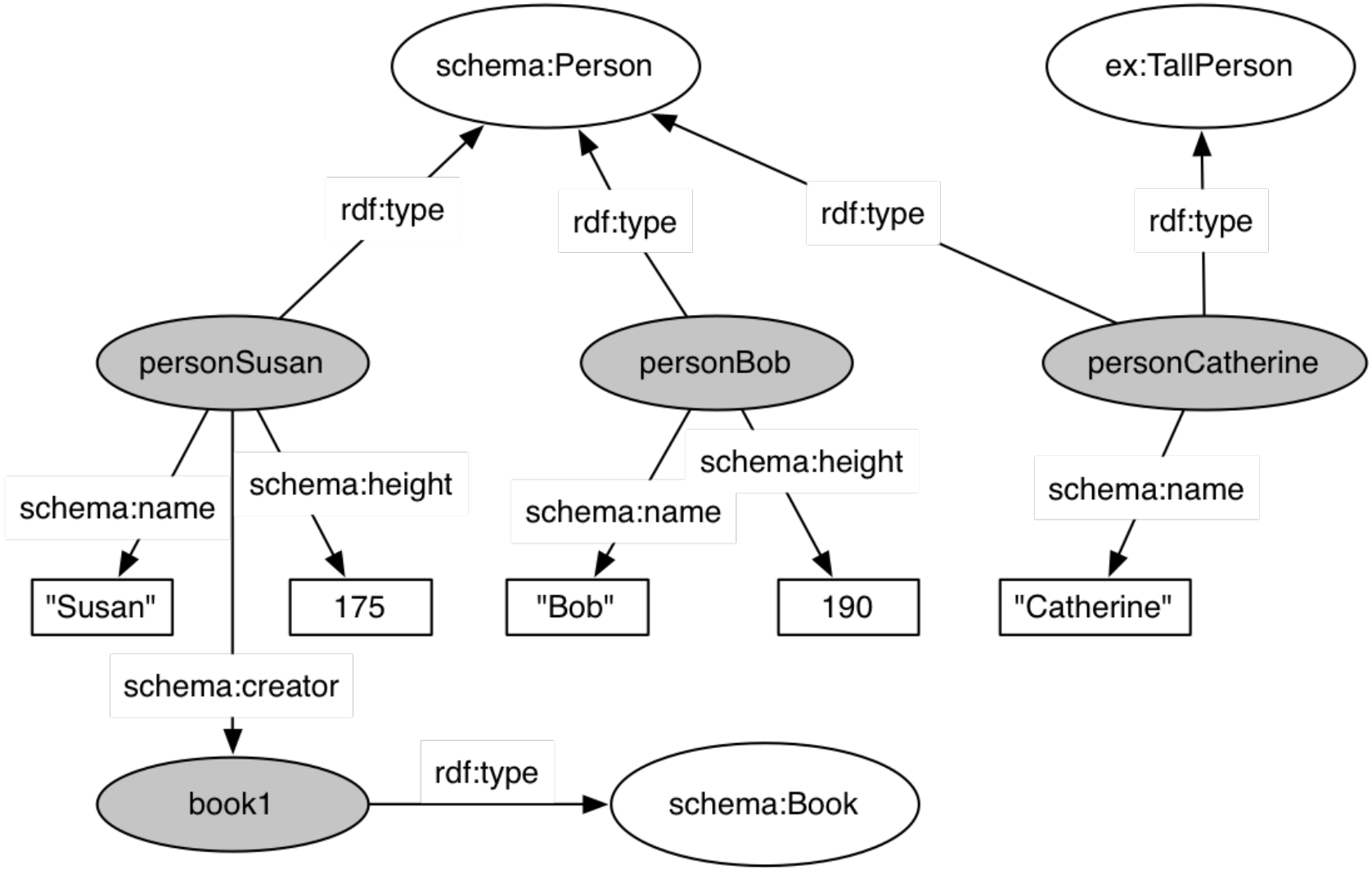
CONSTRUCT

DESCRIBE

Query structure

```
# defining prefixes
PREFIX foo: <http://example.com/resources/>
...

# data sources over which a query will be executed
FROM ...

# return values of a query
SELECT ...

# graph pattern
WHERE {
    …
}

# query modifiers
ORDER BY ...
```

Example – Graph 1



Example – Graph 1 – textual form

personSusan	rdf:type	schema:Person .
personSusan	schema:name	“Susan” .
personSusan	schema:height	175 .
personSusan	schema:autor	book1 .
personBob	rdf:type	schema:Person .
personBob	schema:name	“Bob” .
personBob	schema:height	190.
personCatherine	rdf:type	schema:Person .
personCatherine	rdf:type	ex:TallPerson .
personCatherine	schema:name	“Catherine” .
book1	rdf:type	schema:Book .
book1	schema:name	“My Bestseller” .

Assumptions

In all examples we will use the following prefixes:

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

Q1: Persons and their names

```
SELECT ?person ?name
WHERE {
        ?person        rdf:type              schema:Person .
        ?person        schema:name           ?name .
}
```

Result of SELECT query

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#" >
    <head>
        <variable name="person"/>
        <variable name="name"/>
    </head>
    <results ordered="false" distinct="false">
        <result>
            <binding name="person”>
                <uri>personSusan</uri>
            </binding>
            <binding name="name”>
                <literal>Susan</literal>
            </binding>
        </result>
        ...
    </results>
</sparql>
```

Result of query Q1 over Graph 1 data

person	name
personSusan	“Susan”
personBob	“Bob”
personCatherine	“Catherine”

FILTER

- Defining constrains over values in the graph pattern (e.g. numerical values constrains X>180)
- Boolean operators can be used, as well as regular expressions (regex)

Q2: Persons taller than 180 cm

```
SELECT ?person ?name
WHERE {
        ?person        rdf:type              schema:Person .
        ?person        schema:name           ?name .
        ?person        schema:height         ?height .

        FILTER ( ?height > 180 )
}
```

Result of query Q2 over Graph 1 data

person	name
personBob	“Bob”

OPTIONAL

- Part of a graph pattern can be declared as optional by using keyword OPTIONAL.

Q3: Return person name and height if exists

```
SELECT ?person ?name ?height
WHERE {
        ?person        rdf:type              schema:Person .
        ?person        schema:name           ?name .
        OPTIONAL { ?person    schema:height    ?height }
}
```

Result of query Q3 over Graph 1 data

person	name	height
personSusan	“Susan”	175
personBob	“Bob”	190
personCatherine	“Catherine”	

UNION

- Defining alternatives in graph pattern. Query result consists of results of all alternatives.

Q4: Tall persons declared explicitly or implicitly

```
SELECT ?person ?name
WHERE {
        ?person        rdf:type       schema:Person .
        ?person        schema:name    ?name .
        {
                {
                        ?person      rdf:type      ex:TallPerson .
                }
                UNION
                {
                        ?person        schema:height        ?height .
                        FILTER ( ?height > 180 )
                }
        }
}
```

Result of query Q4 over Graph 1 data

person	name
personBob	“Bob”
personCatherine	“Catherine”

Sorting and paging

ORDER BY for sorting

LIMIT limiting number of results

OFFSET offset of the first result

Q5: Results from position 21 to 30 sorted by name

```
SELECT ?person ?name
WHERE {
        ?person        rdf:type              schema:Person .
        ?person        schema:name           ?name .
}
ORDER BY ?name
LIMIT 10
OFFSET 20
```

BOUND

- Checks whether a variable has value or not. Can also be used with negation.

Q6: Persons that haven't wrote any book

```
SELECT ?person ?name
WHERE {
        ?person        rdf:type              schema:Person .
        ?person        schema:name           ?name .

        FILTER ( ?person    schema:creator      ?x )
        FILTER ( !bound(?x) )
}
```

Result of query Q6 over Graph 1 data

person	name
personBob	“Bob”
personCatherine	“Catherine”

ASK

- Used to check whether query has any result
- Returns true or false. We don't retrieve any data about query results, only information whether there are results or not

Q7: Are there persons taller that 180 cm?

```
ASK
{
        ?person schema:height ?height .
        FILTER (?height > 180)
}
```

Result of query Q7 over Graph 1 data

true

CONSTRUCT

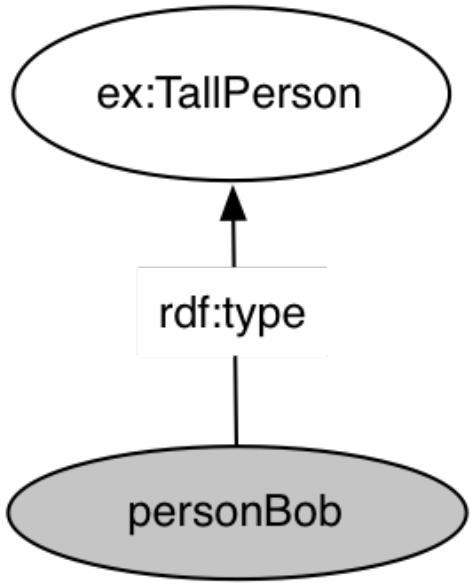
- Creates new RDF graph based on the query results.
- CONSTRUCT query for RDF provides similar functionalities as XSLT provides for XML

Q8: Return a graph with instances of ex:TallPerson Vraća graf sa instancama klase TallPerson za osobe više od 180 cm

```
CONSTRUCT
{
        ?person rdf:type ex:TallPerson .
}
WHERE
{
        ?person        rdf:type              schema:Person .
        ?person        schema:height         ?height .
        FILTER (?height > 180)
}
```

Result of query Q8 over Graph 1 data

personBob rdf:type ex:TallPerson .



DESCRIBE

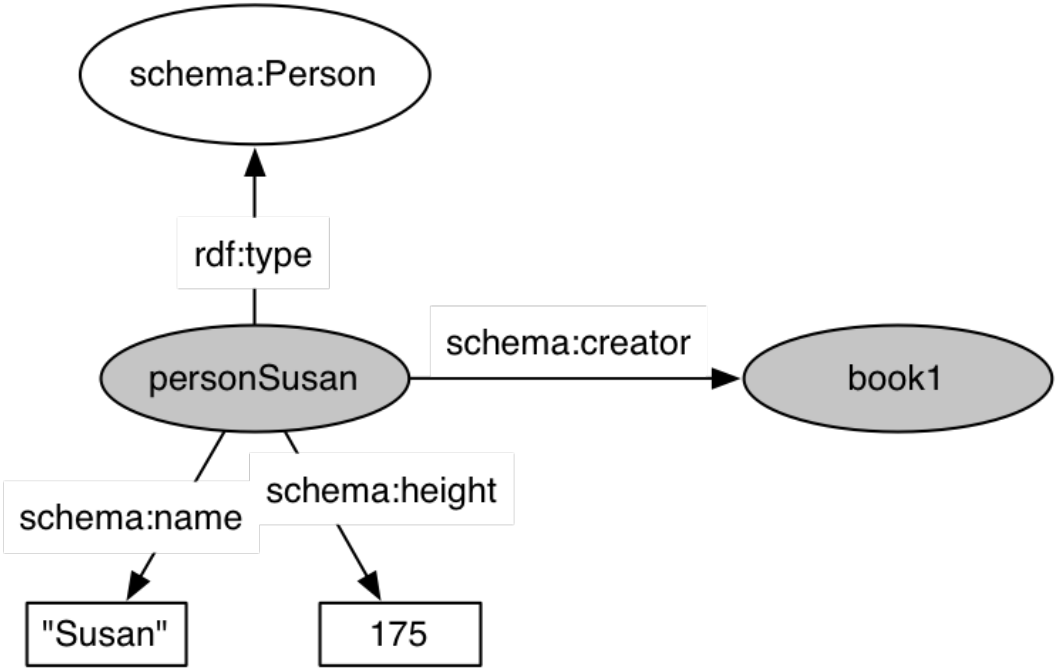
- Returns a graph that contains all edges (properties)of resources described with a graph pattern

Q9: Return all information we have on person named “Susan”

```
DESCRIBE ?person
WHERE {
        ?person        rdf:type              schema:Person .
        ?person        schema:name           “Susan” .
}
```

Result of query Q9 over Graph 1 data

personSusan	rdf:type	schema:Person .
personSusan	schema:name	“Susan” .
personSusan	schema:height	175 .
personSusan	schema:autor	book1 .



Useful links

“SPARQL By Example”, Lee Feigenbaum, Eric Prud'hommeaux, http://www.cambridgesemantics.com/semantic-university/sparql-by-example

“Bee Node: A FOAF Tale”, Leigh Dodds, http://www.ldodds.com/blog/2008/01/bee-node-a-foaf-tale/

SPARQL Query Language for RDF - specification http://www.w3.org/TR/rdf-sparql-query/

Credits

Part of the content is taken from the presentation:

- "SPARQL in a nutshell", Fabien Gandon, http://www.slideshare.net/fabien_gandon/sparql-in-a-nutshell

(Anonymous) survey for your comments and suggestions:
http://goo.gl/cqdp3I

Questions?

UROŠ KRČADINAC

EMAIL: uros@krcadinac.com

URL: www.krcadinac.com