U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Class 4: Learning Objectives

- Review RDF principles;
- RDF Schema.

- Exercises
- Practical Work

- Introduction to SPARQL

Querying with SPARQL

Now that we know how to represent resources with RDF, we will explore how we can interrogate RDF with the SPARQL query language.

We will focus on retrieving information.

Updating RDF datasets with SPARQL are left as an exercise for your projects.

The SPARQL Query Language

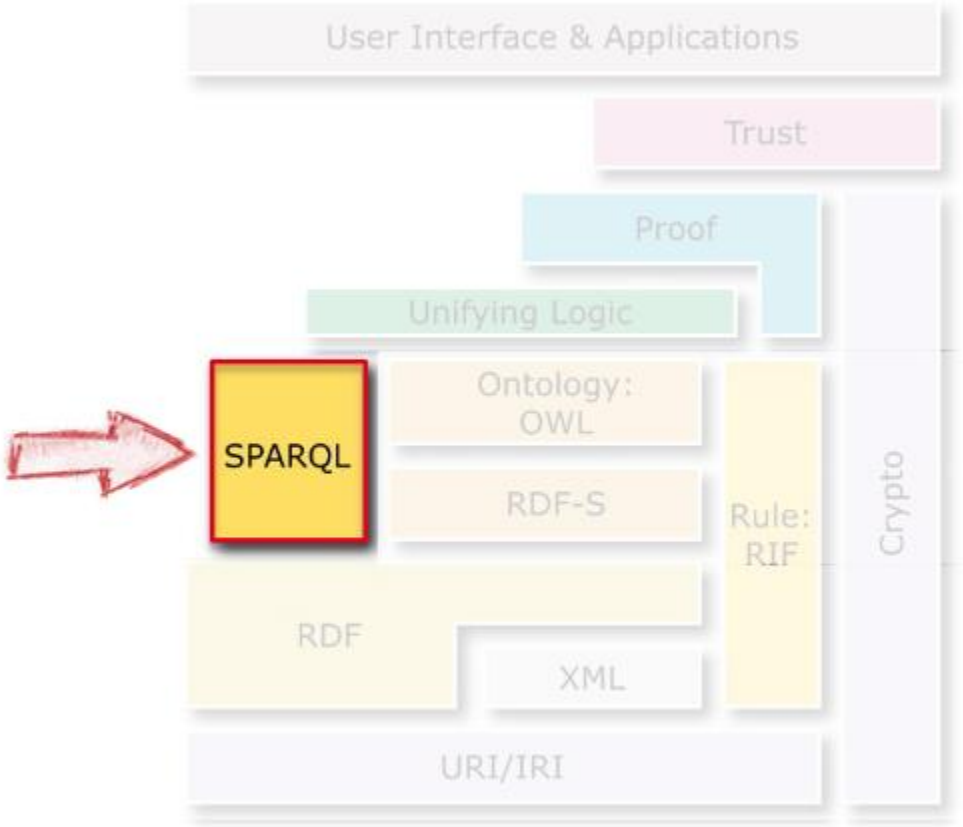
Query RDF triple stores published on the Web

- SPARQL (pronounced sparkle) is a query language that enables users to perform information retrieval by querying RDF datasets on the Web of Data.
- It also enables users to update RDF datasets.

- SPARQL stands for SPARQL Protocol And RDF Query Language

SPARQL Query Language

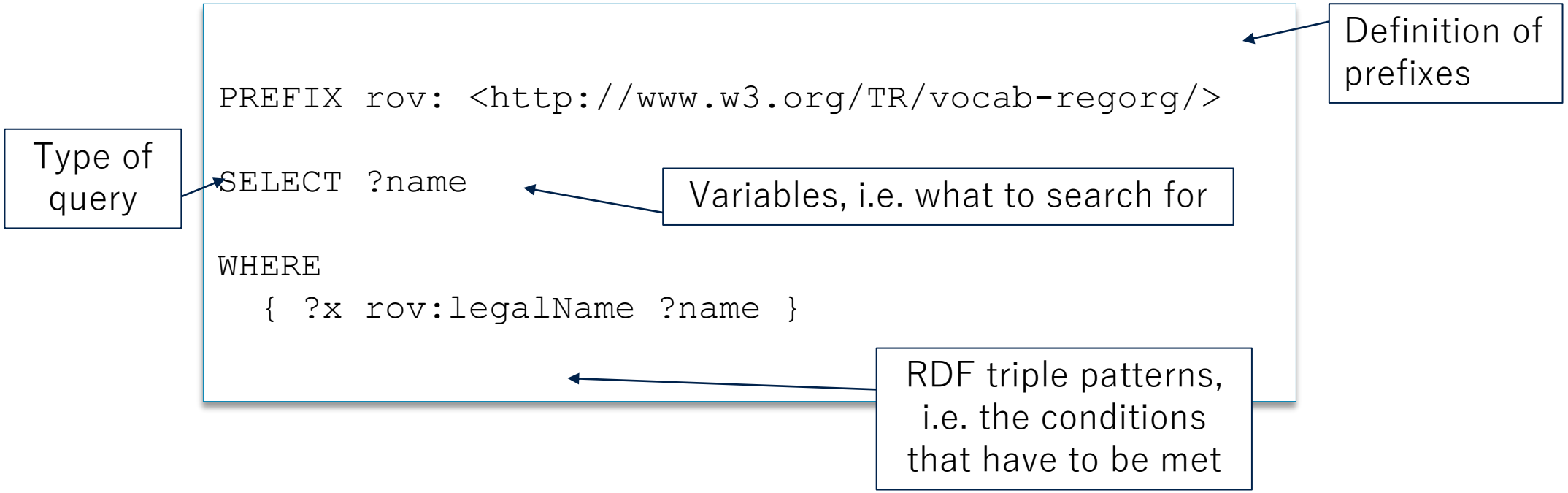
- Query RDF triple stores published on the Web



SPARQL

- SPARQL allows us to:
  - Pull values from structured and semi-structured data;
  - Explore data by querying unknown relationships;
  - Perform complex joins over different graphs in a single, simple query;
  - Including over different endpoints in a single, simple query
  - Transform RDF data from one vocabulary to another

Structure of a SPARQL Query



SPARQL triples

- Turtle syntax with question marks for **variables**:

```
?x rdf:type ex:Person
```

SPARQL triples

- Turtle syntax with question marks for **variables**:

```
?x rdf:type ex:Person
```

- Specify graph pattern to be found:

```
SELECT ?subject ?property ?value
WHERE { ?subject ?property ?value }
```

Structure of a SPARQL Query

- Variables start with a question mark and can match any term (resource or literal);
- Triple patterns are just like triples, except that any of the parts of a triple can be replaced with a variable;
- The SELECT result clause returns a table of variables and values that satisfy the query.
- “a” is syntactic sugar for “rdf:type”

SPARQL triples

- A basic graph pattern is a conjunction of triples

```
SELECT ?x WHERE
{ ?x rdf:type ex:Person .
?x ex:name ?name . }
```

SPARQL triples

in Turtle

- Triples with common subject:

```
SELECT ?name ?fname
WHERE {?x a ex:Person;
      ex:name ?name ;
      ex:firstname ?fname ;
      ex:author ?y . }
```

Declare Prefixes and Namespaces

- Declare prefixes for vocabularies used in the query:

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE {
?student mit:registeredAt ?x .
}
```

Declare Prefixes and Namespaces

- Declare prefixes for vocabularies used in the query:

```
PREFIX mit: <http://www.mit.edu#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student
WHERE {
?student mit:registeredAt ?x .
?x foaf:homepage <http://www.mit.edu> .
}
```

Specify Language and Datatype of Literals

- ```
  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
  SELECT ?x ?f WHERE {
  ?x foaf:name “Nuno"@pt ; foaf:knows ?f .
  }
  ```
  ```

Specify Language and Datatype of Literals

- ```
  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
  SELECT ?x ?f WHERE {
  ?x foaf:name “Nuno"@pt ; foaf:knows ?f .
  }
  ```

- ```
  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
  SELECT ?x WHERE {
  ?x foaf:name “Nuno"@pt ;
  foaf:age "21"^^xsd:integer .
  }
  ```

Optional Pattern

- When part of graph pattern is not mandatory

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
?person foaf:homepage <http://nuno.info> .
OPTIONAL { ?person foaf:name ?name . }
}
```

Optional Pattern

- `OPTIONAL` tries to match a graph pattern but doesn't fail the whole query if the optional match fails.
- If an `OPTIONAL` pattern fails to match for a particular solution, any variables in that pattern remain unbound for that solution.

Alternative Patterns

- Union of results of graph patterns

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person foaf:name ?name .
  {
   ?person foaf:homepage <http://nuno.info> .
  }
UNION
  {
   ?person foaf:homepage <http://nunes.org> .
  }
}
```

Alternative Patterns

- UNION is useful to match alternatives. By combining two or more *graph patterns*.
- It is thus not restricted to triples patterns.

Negation

- Remove results that match a pattern

```
PREFIX ex: <http://www.example.abc#>
SELECT ?x
WHERE {
  ?x a ex:Person
  MINUS { ?x a ex:Man }
}
```

Predefined Variable Values

- Results where part of the bindings are predefined

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
?person foaf:name ?name .
}
VALUES ?name { "Peter" "Pedro" "Pierre" }
```

Variable Binding

- Results where part of the bindings are computed

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
  ?person ex:fname ?fname ;
     ex:lname ?lname .
BIND (concat(?fname, ?lname) AS ?name)
}
```

Property path

- Regular expressions on property path between resources

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

  ?x foaf:name “Nuno Nunes" ;
       foaf:knows+ ?friend .
}
```

Keep Distinct Results

- Keep one occurrence of similar results with same values for same variables

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

Filter Results by Values

Declare constraints on variable values

- `select` = select values to be returned
- `where` = graph pattern
- `filter` = constraints in the `where` clause with expressions and functions

E.g. Person At Least 18 Years Old

```
PREFIX ex: <http://fe.up.pt/schema#>
SELECT ?person ?name
WHERE {
?person rdf:type ex:Person ;
ex:name ?name ;
ex:age ?age .
FILTER (xsd:integer(?age) >= 18)
}
```

Usual test: if... then... else...

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * where {
?x foaf:name ?name ;

foaf:age ?age .
FILTER ( if (langMatches(lang(?name), "PT"),
?age >= 18, ?age >= 21) )
}
```

Verify Presence / Absence of a Pattern

- *exists* checks whether a pattern **occurs** in the graph
- *not exists* checks whether a pattern **does not occur** in the graph

```
SELECT ?name
WHERE {
?x foaf:name ?name .
FILTER NOT EXISTS { ?x foaf:age -1 }
}
```

Order and Limit Results

- E.g.: sort results by name from n°21 to n°40

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name . }
ORDER BY ?name
LIMIT 20
OFFSET 20
```

Order and Limit Results

- E.g.: sort results by name from n°21 to n°40

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name . }
ORDER BY ?name
LIMIT 20
OFFSET 20
```

Solution modifiers
– `LIMIT` - limits the number of rows returned
– `ORDER BY` - sorting
– `OFFSET` - used together with `LIMIT` and `ORDER BY` for slicing, e.g., for paging

Aggregate Results

- Group results by variable(s) values: `group by`
- Aggregate values: `count, sum, min, max, avg, group_concat, sample`
- Filter aggregated values: `having`

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE { ?student mit:score ?score . }
GROUP BY ?student
HAVING(AVG(?score) >= 10)
```

Check the Existence of a Solution

- Do not enumerate all solutions, just answer true or false

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?person foaf:age 111 . }
```

Construct a Result Graph

- Result of query is a fresh new RDF graph

```
PREFIX mit: <http://www.mit.edu#>
PREFIX corp: <http://mycorp.com/schema#>

CONSTRUCT { ?student a corp:FuturExecutive . }

WHERE { ?student a mit:Student . }
```

Querying SPARQL Endpoints

- DBpedia is an effort to “triplify” Wikipedia
- http://dbpedia.org/sparql

- “Find me 50 distinct ‘things’ with names in DBpedia”

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Further reading SPARQL

- SPARQL 1.1 Overview W3C Recommendation
- SPARQL 1.1 Query Language W3C Recommendation
- SPARQL 1.1 Update W3C Recommendation
- SPARQL 1.1 Federated Query W3C Recommendation
- DBpedia SPARQL endpoint