NSWI144 – Linked Data – Lecture 5 – 15th November 2011

SPARQL

Martin Svoboda

Faculty of Mathematics and Physics

Charles University in Prague

SPARQL

- SPARQL 1.1 Query Language
  - W3C Working Draft – version 1.1 (2011)
    - http://www.w3.org/TR/sparql11-query/
    - …
  - W3C Recommendation – version 1.0 (2008)
    - http://www.w3.org/TR/rdf-sparql-query/
    - …
- Language for querying RDF data

Basic Idea

- Graph patterns
  - Based on triples
  - URIs, literals, blank nodes
  - Variables
    - ?name
    - $name
- Query result
  - Based on matching subgraphs and substitutions

Simple Query

- Data
  - ```
    @prefix sis: <http://is.cuni.cz/studium/sis#> .
    sis:stud1 sis:name "John" ;
              sis:age "26" ;
              rdf:type sis:person .
    sis:stud2 sis:name "Peter" ;
              sis:age "30" ;
              rdf:type sis:person .
    sis:stud3 sis:name "Martin" ;
              sis:age "20" .
    ```
    ```

Simple Query

- Query
  - ```
    PREFIX sis: <http://is.cuni.cz/studium/sis#>
    SELECT ?name ?age
    WHERE {
      ?stud rdf:type sis:person ;
            sis:name ?name ; sis:age ?age .
    }
    ```
- Result
  - ```
    ?name     ?age
    "John"    "26"
    "Peter"   "30"
    ```

Basic Idea

- Query result
  - `SELECT ?name ?age FROM … WHERE {…}`
  - Variable binding = (variable, value)
    - `(?name, "John")`
  - Result item = set of variable bindings
    - `{ (?name, "John"), (?age, "26") }`
  - Query result = sequence of result items
    - `{ (?name, "John"), (?age, "26") },`
      `{ (?name, "Peter"), (?age, "30") }`

Matching Literals

- Language tags
  - "Praha"
  - "Praha"@cs
  - "Prague"@en
- Typed literals
  - 1 = "1"^^xsd:integer
  - 1.5 = "1.5"^^xsd:decimal
  - 1.0E6 = "1"^^xsd:double
  - true = "true"^^xsd:boolean

Blank Nodes

- Data
  - Distinct nodes within the document scope
- Query
  - Expression
    - Blank nodes act as non-selectable variables
  - Result
    - Distinct nodes within the result scope
      - Blank node identifiers in the result may not correspond to original blank node identifiers from the given document

Query Syntax

- Query components
  - `PREFIX …`
    `SELECT | DESCRIBE | ASK | CONSTRUCT …`
    `FROM …`
    `WHERE { … }`
    `ORDER BY … LIMIT … OFFSET …`

Prologue Component

- PREFIX
  - Without "." separator
  - Example
    - `PREFIX name: <http://www.my.cz/>`
      `name:local`
- Relative URIs
  - Example
    - `BASE <http://www.my.cz/>`
      `<sis>`
    - `<http://www.my.cz/sis>`

Graph Patterns

- Graph patterns
  - Basic graph patterns
  - Group graph patterns
  - Optional graph patterns
  - Alternative graph patterns
  - Patterns on named graphs
- Inductive construction

Graph Patterns

- Basic graph patterns
  - Set of triples
  - Interpretation
    - Required matching
      - Conjunction of all individual triple patterns
      - All variables need to be bound
      - Otherwise the result item candidate is thrown away

Graph Patterns

- Group graph patterns
  - `{ … }`
  - Empty group
    - `{ }`

Graph Patterns

- Optional graph patterns
  - `pattern OPTIONAL { pattern }`
  - Interpretation
    - Optional matching
      - If the optional part does not match, it creates no bindings but does not eliminate the solution

Graph Patterns

- Optional graph patterns
  - Example
    - ```
      SELECT ?name ?age ?type
      WHERE {
        ?stud sis:name ?name ; sis:age ?age .
        OPTIONAL {
          ?stud rdf:type ?type.
        }
      }
      ```
    - ```
      ?name     ?age   ?type
      "John"    "26"   sis:person
      "Peter"   "30"   sis:person
      "Martin"  "20"
      ```

Graph Patterns

- Optional graph patterns
  - Left-associative
    - `{ OPTIONAL { pattern1 } }`
      `{ { } OPTIONAL { pattern1 } }`
    - `p1 OPTIONAL { p2 } OPTIONAL { p3 }`
      `{ p1 OPTIONAL { p2 } } OPTIONAL { p3 }`
  - Multiple patterns
    - `{ s1 p1 o1 .`
      `OPTIONAL { pattern1 } .`
      `OPTIONAL { pattern2 }`
      `}`

Graph Patterns

- Alternative graph patterns
  - `{ pattern1 } UNION { pattern2 }`
  - Interpretation
    - Union of results from both patterns

Graph Patterns

- Patterns on named graphs
  - `FROM NAMED <http://...>`
  - `GRAPH <http://...> { pattern }`
  - `GRAPH ?graph { pattern }`
  - Usage
    - `GRAPH … {…}`
  - Interpretation
    - SPARQL queries are executed over a dataset:
      - One default graph (`FROM …`)
      - Zero or more named graphs (`FROM NAMED …`)

Term Constraints

- FILTER component
  - Filters result item candidates
  - Contains operators and functions
  - Applied on entire group graph patterns
  - Example
    - `FILTER (?age < 20)`
  - 3 value logic
    - True, false, error

Term Constraints

- Arithmetic operators
  - Unary + –
  - Binary + – * /
- Comparison operators
  - < <= >= >
  - = !=
- Logical connectives
  - ! && ||

Term Constraints

- Other operators
  - bound, isUri, isBlank, isLiteral
- Access functions
  - str, lang, dataType

Result Forms

- Query components
  - `PREFIX …`
    `SELECT | DESCRIBE | ASK | CONSTRUCT …`
    `FROM …`
    `WHERE { … }`
    `ORDER BY … LIMIT … OFFSET …`

Result Forms

- SELECT
  - Result
    - Sequence of sets of variable bindings
  - Variables are separated by space character
  - Asterisk (*) selects all variables
- ASK
  - Checks whether at least one result item exists
  - Result
    - True or false

Result Forms

- DESCRIBE
  - Result
    - An RDF graph with data about resources
  - Non-deterministic
  - Examples
    - `DESCRIBE <http://www.my.cz/>`
    - ```
      DESCRIBE ?s
      FROM <http://is.cuni.cz/studium>
      WHERE { ?s rdf:type sis:person }
      ```
      ```

Result Forms

- CONSTRUCT
  - Result
    - An RDF graph constructed from a template
  - Example
    -
```
CONSTRUCT
  { ?s sis:name concat(?n1, " ", ?n2) }
FROM <http://is.cuni.cz/studium>
WHERE
  { ?s sis:firstName ?n1; sis:lastName ?n2 }
```

Solution Modifiers

- Modifiers
  - `SELECT DISTINCT …`
    `FROM … WHERE …`
    `ORDER BY … LIMIT … OFFSET …`
  - Only for SELECT queries
  - Modify the entire result sequence
- DISTINCT
  - Removes duplicates from the result sequence

Solution Modifiers

- ORDER BY
  - Orders items in the result sequence
  - Hierarchical ordering criteria
    - ASC = ascending (default), DESC = descending
    - Unbound variable < blank node < URI < literal
  - Example
    - `ORDER BY ASC(?name), DESC(?age)`

Solution Modifiers

- LIMIT
  - Limits the number of items in the result sequence
  - Always should be preceded by ORDER BY modifier
  - Example
    - `ORDER BY ?name LIMIT 10`
- OFFSET
  - Index of the first reported item from the sequence
  - Example
    - `ORDER BY ?name LIMIT 10 OFFSET 20`