NSWI144 – Linked Data – Lecture 7 – 26 November 2012

SPARQL

Martin Svoboda

Faculty of Mathematics and Physics

Charles University in Prague

Outline

- SPARQL 1.1
  - **Query**
    - Negation
    - Property paths
    - Aggregates
  - **Update**

SPARQL Query

- **SPARQL 1.1 Query Language**
  - Query language for RDF graphs
  - Proposed recommendation
    - 8 November 2012
    - http://www.w3.org/TR/sparql11-query/
  - Motivation
    - Extension of SPARQL 1.0

Negation

- **EXISTS** constraint
  - **… when existence of solutions should be tested**
  - Syntax
    - *Pattern1* `FILTER [NOT] EXISTS { Pattern2 }`
  - Notes
    - Does not generate any additional bindings

Negation

- **MINUS** graph pattern
  - **… when compatible solutions should be removed**
  - Syntax
    - *Pattern1* **`MINUS`** { *Pattern2* }
  - Idea
    - Solutions of the left-hand pattern are preserved if and only if they are not compatible with any solution of the right-hand pattern
    - Corresponds to $\text{minus}_{\text{rdf}}$ operation

Assignment 7.1

- Find other equivalent query statements for the problem from Assignment 6.7.
  - I.e. select all courses that are **not** taught on Mondays and nor on Fridays
  - Assume only courses in winter semester 2011/12
  - Return course references and codes
  - Use NOT EXISTS and MINUS

Assignment 6.7

- Statement with **OPTIONAL and BOUND**
  - ```
    PREFIX i: <http://is.cuni.cz/is#>
    SELECT DISTINCT ?c ?code
    FROM <http://is.cuni.cz/teaching/>
    WHERE {
      ?c i:code ?code .
      OPTIONAL {
        ?e i:course ?c ; i:day ?day ;
           i:term t:term1112W .
        FILTER ( (?day = "MON") || (?day = "FRI") )
      }
      FILTER ( ! bound(?e) )
    }
    ```
    ```

Assignment 7.1

- Statement with **NOT EXISTS**
  - ```
    PREFIX i: <http://is.cuni.cz/is#>
    SELECT DISTINCT ?c ?code
    FROM <http://is.cuni.cz/teaching/>
    WHERE {
      ?c i:code ?code .
      FILTER NOT EXISTS {
        ?e i:course ?c ; i:day ?day ;
           i:term t:term1112W .
        FILTER ( (?day = "MON") || (?day = "FRI") )
      }
    }
    ```
    ```

Assignment 7.1

- Statement with **MINUS**
  - ```
    PREFIX i: <http://is.cuni.cz/is#>
    SELECT DISTINCT ?c ?code
    FROM <http://is.cuni.cz/teaching/>
    WHERE {
      ?c i:code ?code .
      MINUS {
        ?e i:course ?c ; i:day ?day ;
           i:term t:term1112W .
        FILTER ( (?day = "MON") || (?day = "FRI") )
      }
    }
    ```
    ```

Property Paths

- Motivation
  - **Describing routes between two nodes in a graph**
    - … routes of arbitrary lengths
    - … and even routes satisfying more complex conditions
- Example
  - All friends of student s:s4 and their friends

```
PREFIX i: <http://is.cuni.cz/is#>
SELECT DISCTINCT ?friend
FROM <http://is.cuni.cz/students/>
WHERE { s:s4 foaf:knows{1,2} ?friend . }
```

Property Paths

- Notes
  - **Lengths**
    - Path of length 1 is an ordinary triple
    - Path of length 0 connects a graph node to itself
  - **Variables**
    - Variables cannot be used inside paths
      - But they can be used at their ends
  - **Cycles**
    - Matching of cycles is possible
      - But each cycle is considered at most once

Property Paths

- Syntax
  - **Edge**
    - *Predicate*
  - **Sequences**
    - *Path1* / *Path2*
  - **Alternatives**
    - *Path1* | *Path2*
  - **Groups**
    - (*Path*)

Property Paths

- Syntax
  - **Occurrences**… particular number of edges is expected
    - *Path**
    - *Path*+
    - *Path*?
    - *Path*{*n*,*m*}
    - *Path*{*n*}
    - *Path*{*n*,}
    - *Path*{,*m*}

Property Paths

- Syntax
  - **Inverse**… roles of subject and object are swapped
    - ^*Path*
  - **Negation**… any predicate except the specified ones
    - ! *Predicate*
    - ! (*Predicate1* | *Predicate2* | …)
    - ! (^*Predicate1* | ^*Predicate2* | …)
    - ! (*PredicateP1* | *PredicateP2* | … | ^*PredicateR1* | ^*PredicateR2* | …)

Assignment 7.2

- Find all pre-requisites or co-requisites for a course with code *NPRG036*
  - Search these dependencies recursively
  - Return course references and codes
  - Order solutions using course codes

Assignment 7.2

- Statement
  - ```
    PREFIX i: <http://is.cuni.cz/is#>
    SELECT DISTINCT ?c ?code
    FROM <http://is.cuni.cz/teaching/>
    WHERE {
      ?s i:code "NPRG036" .
      ?s (i:prerequisity|i:corequisity)+ ?c .
      ?c i:code ?code .
    }
    ORDER BY ?code
    ```
- Solutions

?c	?code
t:c3	"NPRG030"
t:c5	"NSWI096"
    ```

Aggregates

- Motivation
  - Aggregation of groups of solutions
- Example
  - Total capacity of all rooms in each building
    -
```
PREFIX i: <http://is.cuni.cz/is#>
SELECT ?b (SUM(?c) AS ?capacity)
FROM <http://is.cuni.cz/faculty/>
WHERE { ?r i:building ?b ; i:capacity ?c . }
GROUP BY ?b
```

Aggregates

- Functions
  - COUNT
  - SUM, AVG, MIN, MAX
- Syntax
  - **SELECT** …
    FROM …
    WHERE …
    GROUP BY …
    HAVING …

Assignment 7.3

- Return average results for all students over their enrolled courses
  - Assume only courses in winter semester 2011/12
  - Ignore enrollments with undefined results
  - Describe students by their full names
  - Assume only students with at least 10 courses

Assignment 7.3

- Statement
  - ```
    PREFIX i: <http://is.cuni.cz/is#>
    PREFIX t: <http://is.cuni.cz/teaching/>
    SELECT ?first ?last (AVG(?r) AS ?average)
    FROM <http://is.cuni.cz/students/>
    WHERE {
      ?s i:name [ i:first ?first ; i:last ?last ] ;
         i:enroll [ i:course ?c ; i:result ?r ;
                    i:term t:term1112W ] .
    }
    GROUP BY ?s ?first ?last
    HAVING (COUNT(?c) >= 10)
    ```
    ```

SPARQL Update

- **SPARQL 1.1 Update Language**
  - Update language for RDF graphs
  - Proposed recommendation
    - 8 November 2012
    - http://www.w3.org/TR/sparql11-update/
  - Motivation
    - Update, create and remove RDF graphs

Terminology

- **Graph Store**
  - Mutable container of RDF graphs
    - One (unnamed) slot with a **default graph**
      - Used when operations do not specify any particular graph
    - Zero or more (named) slots for **named graphs**
- Functionality
  - **Updates** – addition and removal of triples
  - **Management** – creation and deletion of graphs…

Terminology

- Updates
  - **Operation**
    - Action resulting from a single command
  - **Request**
    - Sequence of requested operations

Updates

- **INSERT DATA**
  - Adds triples into a given graph
  - Syntax
    - **`INSERT DATA`** { *Triples* }
    - **`INSERT DATA`** { **`GRAPH`** *Graph* { *Triples* } }
  - Notes
    - Variables are not allowed in provided triples
    - Blank nodes are inserted as new distinct nodes
    - Insertion of already existing triples has no effect
    - Destination graph should be created if required

Updates

- **DELETE DATA**
  - Removes triples from a given graph
  - Syntax
    - **DELETE DATA** { *Triples* }
    - **DELETE DATA** { **GRAPH** *Graph* { *Triples* } }
  - Notes
    - Variables and blank nodes are not allowed
    - Deletion of non-existing triples has no effect

Updates

- **DELETE / INSERT**
  - Removes and/or adds triples from/to a given graph
  - Idea
    - **Selection**
      - Group graph pattern leading to the solution sequence
    - **Deletion**
      - Quad pattern of triples to be deleted
    - **Insertion**
      - Quad pattern of triples to be inserted

Updates

- **DELETE / INSERT**
  - Syntax
    - `WITH` *Graph*

      `DELETE` { *QuadPattern* }

      `INSERT` { *QuadPattern* }

      `USING` *Graph*

      `USING NAMED` *Graph*

      `WHERE` *GroupGraphPattern*

Updates

- **DELETE / INSERT**
  - Explicit specification of graphs
    - **Matched graph**
      - » … those considered in `WHERE`
      - Definitions of graphs via `USING` / `USING NAMED`
        - » They act exactly as standard `FROM` / `FROM NAMED`
      - Then we can use standard `GRAPH` graph patterns
    - **Modified graphs**
      - » … those considered in `DELETE` and `INSERT`
      - Quad patterns with specified `GRAPH`

Updates

- **DELETE / INSERT**
  - Universal specification of graphs
    - **WITH**
      - Defines one single graph for WHERE, DELETE and INSERT
      - `WITH G`
        `DELETE { … } INSERT { … }`
        `WHERE { … }`
      - `DELETE { GRAPH G { … } }`
        `INSERT { GRAPH G { … } }`
        `USING G WHERE { … }`
      - Explicit definitions locally override this universal one

Updates

- **DELETE / INSERT**
  - Notes
    - Deletion and insertion parts are optional
      - However at least one of them must be provided
    - Solution sequence is evaluated only once
    - Deletion always happens before insertion
    - Illegal triples (unbound or invalid) are ignored
      - Not deleted or not inserted
    - Special shortcut `DELETE WHERE { … }`
      - Removes all matched triples

Assignment 7.4

- Add full names to all students
  - Consider only students having both names
    - I.e. those with specified `i:first` and `i:last`
  - Use CONCAT function

Assignment 7.4

- Statement
  - ```
    PREFIX i: <http://is.cuni.cz/is#>
    WITH <http://is.cuni.cz/students/>
    INSERT {
      ?s i:fullName CONCAT(?first, " ", ?last) .
    }
    WHERE {
      ?s i:name [ i:first ?first ; i:last ?last ] .
    }
    ```
    ```

Assignment 7.4

- Solutions

?s	?first	?last
s:s2	"Martin"	"Svoboda"
s:s4	"Tomas"	"Knap"
s:s6	"Jakub"	"Klimek"

- Insertions
  - ```
    s:s2 i:fullName "Martin Svoboda" .
    s:s4 i:fullName "Tomas Knap" .
    s:s6 i:fullName "Jakub Klimek" .
    ```
    ```

Updates

- **LOAD**
  - Inserts triples from a document into a graph
  - `LOAD` *Document*
  - `LOAD` *Document* `INTO GRAPH` *Graph*
- **CLEAR**
  - Removes all triples in the specified graphs
  - `CLEAR GRAPH` *Graph*
  - `CLEAR ( DEFAULT | NAMED | ALL )`

Management

- **CREATE**
  - Creates a new empty graph
  - `CREATE GRAPH` *Document*
- **DROP**
  - Removes the specified graphs
  - `DROP GRAPH` *Graph*
  - `DROP ( DEFAULT | NAMED | ALL )`

Management

- **COPY**
  - Copies data from a source into a target graph
    - First, all triples in the target graph are removed
    - All triples from the source are inserted into the target
    - The source graph is not affected
  - `COPY ( GRAPH` *Graph* `| DEFAULT )`
    `TO ( GRAPH` *Graph* `| DEFAULT )`

Management

- **MOVE**
  - Moves data from a source into a target graph
    - First, all triples in the target graph are removed
    - All triples from the source are inserted into the target
    - Finally, the source graph is removed
  - `MOVE ( GRAPH` *Graph* `| DEFAULT )`
    `TO ( GRAPH` *Graph* `| DEFAULT )`

Management

- **ADD**
  - Copies data from a source into a target graph
    - All initial triples in the source are kept
    - All triples from the source are inserted into the target
    - The source graph is not affected
  - `ADD ( GRAPH` *Graph* `| DEFAULT )`
    `TO ( GRAPH` *Graph* `| DEFAULT )`

Conclusion

- SPARQL 1.1
  - **Query**
    - `EXISTS` constraint
    - `MINUS` pattern
    - Property paths
    - Aggregates
  - **Update**
    - `INSERT DATA`, `DELETE DATA`, `DELETE` / `INSERT`
    - …