Κεφάλαιο 5

Η γλώσσα SPARQL

Σε αυτό το κεφάλαιο αναφερόμαστε διεξοδικά σε μια από τις πιο διαδεδομένες τεχνολογίες των Συνδεδεμένων Δεδομένων που δεν είναι άλλη από την *SPARQL Protocol and RDF Query Language (SPARQL)*. Η SPARQL αποτελεί μια οικογένεια προδιαγραφών για αλληλεπίδραση πάνω σε δεδομένα RDF. Πιο συγκεκριμένα, αυτές οι προδιαγραφές μας δίνουν τη δυνατότητα να ανακτήσουμε και να διαχειριστούμε δεδομένα που είναι αποθηκευμένα σε αποθετήρια τριάδων RDF. Διασταλτικά, θα μπορούσαμε να πούμε πως η SPARQL για τα αποθετήρια τριάδων RDF είναι ό,τι η SQL για τις σχεσιακές βάσεις δεδομένων (RDBMS). Αρχικά, αναφέρουμε τις επιμέρους προδιαγραφές που συνιστούν τη SPARQL και σχολιάζουμε περιληπτικά καθεμία από αυτές. Στη συνέχεια, αναλύουμε τις προδιαγραφές της γλώσσας ερωτημάτων SPARQL, οι οποίες αποτελούν ίσως το πιο σημαντικό και ευρέως διαδεδομένο σύνολο προδιαγραφών της SPARQL. Κάθε δυνατότητα που παρουσιάζουμε, θα έχετε την ευκαιρία να τη δοκιμάζετε στην πράξη απευθύνοντας το αντίστοιχο ερώτημα σε κατάλληλα διαμορφωμένα τελικά σημεία SPARQL (SPARQL endpoints).

5.1 Ανάλυση του προτύπου

Όπως αναφέρεται και στη σύνοψη του προτύπου, η SPARQL είναι ένα σύνολο προδιαγραφών που παρέχει γλώσσες και πρωτόκολλα ερωτήσεων και χειρισμού γράφων RDF τόσο στον παγκόσμιο ιστό όσο και σε αποθετήρια RDF. Η πρώτη επίσημη έκδοση του προτύπου δημοσιεύτηκε τον Ιανουάριο του 2008, ενώ η πιο πρόσφατη έκδοση τη στιγμή δημοσίευσης αυτού του βιβλίου είναι η SPARQL 1.1, η οποία δημοσιεύτηκε τον Μάρτιο του 2013. Το πρότυπο SPARQL 1.1 συνίσταται από τις ακόλουθες προδιαγραφές:

1. *Γλώσσα ερωτημάτων SPARQL 1.1 (SPARQL 1.1 Query Language)* - Μια γλώσσα ερωτημάτων για δεδομένα RDF.

2. *Μορφότυπα αποτελεσμάτων αναζήτησης σε SPARQL 1.1 (SPARQL 1.1 Query Results JSON Format και SPARQL 1.1 Query Results CSV and TSV Formats)* - Πέρα από το παραδοσιακό μορφότυπο αποτελεσμάτων σε ερωτήματα RDF σύμφωνα με τις προδιαγραφές της XML (SPARQL-XML-Result), το πρότυπο SPARQL 1.1 επιτρέπει την έκφραση των αποτελεσμάτων αυτών σε τρία εναλλακτικά μορφότυπα: JavaScript Object Notation - JSON), comma-separated values - CSV και tab-separated values - TSV.

3. *Ομόσπονδα ερωτήματα SPARQL 1.1 (SPARQL 1.1 Federated Query)* - Προδιαγραφές που ορίζουν μια επέκταση στη SPARQL 1.1 Query Language για εκτέλεση ερωτημάτων σε διαφορετικά τελικά σημεία SPARQL με κατανεμημένο τρόπο.

4. *Συστήματα συμπερασματολογίας SPARQL 1.1 (SPARQL 1.1 Entailment Regimes)* - Προδιαγραφές που ορίζουν τη σημασιολογία ερωτημάτων SPARQL υπό το πρίσμα συστημάτων συμπερασματολογίας, όπως τα RDF Schema, OWL, Rule Interchange Format - RIF.

5. *Γλώσσα ενημερώσεων SPARQL 1.1 (SPARQL 1.1 Update Language)* - Μια γλώσσα ενημερώσεων (update) για γράφους RDF.

6. *Πρωτόκολλο SPARQL 1.1 για RDF (SPARQL 1.1 Protocol for RDF)* - Ένα πρωτόκολλο που ορίζει τρόπους μετάδοσης ερωτημάτων αλλά και ενημερώσεων SPARQL σε μια υπηρεσία SPARQL.

7. *Περιγραφή υπηρεσιών SPARQL 1.1 (SPARQL 1.1 Service Description)* - Προδιαγραφές που ορίζουν μια μέθοδο για ανακάλυψη και ένα λεξιλόγιο για περιγραφή υπηρεσιών SPARQL.

8. *Πρωτόκολλο HTTP για αποθετήρια γράφου SPARQL 1.1 (SPARQL 1.1 Graph Store HTTP Protocol)* - Σε αντίθεση με το πλήρες πρωτόκολλο SPARQL, οι προδιαγραφές αυτές ορίζουν ένα μινιμαλιστικό τρόπο απευθείας χειρισμού δεδομένων σε ένα αποθετήριο τριάδων RDF μέσω κοινών λειτουργιών HTTP.

9. *Σουίτα ελέγχων SPARQL 1.1 (SPARQL 1.1 Test Cases)* - Μια σουίτα ελέγχων, χρήσιμη για την κατανόηση βασικών στοιχείων των προδιαγραφών αλλά και ικανή να διαπιστώσει ότι ένα σύστημα πληροί τις προδιαγραφές SPARQL 1.1.

Ας κάνουμε μια εισαγωγή στη χρήση των γλωσσών, πρωτοκόλλων και λοιπών προδιαγραφών της SPARQL 1.1 μέσα από ένα μικρό παράδειγμα. Έτσι λοιπόν, έστω ότι ο παρακάτω γράφος RDF περιέχει προσωπικές πληροφορίες σχετικές με την Alice και τις κοινωνικές της επαφές (στο παράδειγμά μας χρησιμοποιούμε τη σύνταξη Turtle).

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
<http://example.org/alice#me> a foaf:Person .
<http://example.org/alice#me> foaf:name "Alice" .
<http://example.org/alice#me> foaf:mbox <mailto:alice@example.org> .
<http://example.org/alice#me> foaf:knows <http://example.org/bob#me> .
<http://example.org/bob#me> foaf:knows <http://example.org/alice#me> .
<http://example.org/bob#me> foaf:name "Bob" .
<http://example.org/alice#me> foaf:knows
↪  <http://example.org/charlie#me> .
<http://example.org/charlie#me> foaf:knows
↪  <http://example.org/alice#me> .
<http://example.org/charlie#me> foaf:name "Charlie" .
<http://example.org/alice#me> foaf:knows <http://example.org/snoopy> .
<http://example.org/snoopy> foaf:name "Snoopy"@en .
```

Μέσω της SPARQL 1.1, μπορούμε να απευθύνουμε ερωτήματα στον παραπάνω γράφο, να τον τοποθετήσουμε σε αποθετήρια RDF και να τον χειριστούμε με διάφορους τρόπους. Στις αμέσως επόμενες ενότητες θα αναφερθούμε περιληπτικά στις επιμέρους προδιαγραφές που συνιστούν το πρότυπο SPARQL 1.1.

5.1.1 Γλώσσα ερωτημάτων SPARQL 1.1

Θεωρώντας ότι τα δεδομένα RDF της προηγούμενης ενότητας είναι διαθέσιμα από μια υπηρεσία SPARQL (δηλαδή, ένα τελικό σημείο SPARQL ικανό να χειριστεί ερωτήματα SPARQL), μπορούμε να χρησιμοποιήσουμε τη γλώσσα ερωτημάτων SPARQL 1.1 για να διατυπώσουμε ερωτήματα που αφορούν: από απλά ταιριάσματα στον γράφο RDF που σχηματίζουν τα δεδομένα αυτά έως πιο σύνθετες εκφράσεις. Για παράδειγμα, θα μπορούσαμε να ρωτήσουμε τα ονόματα των ατόμων και το πλήθος των φίλων τους χρησιμοποιώντας το παρακάτω ερώτημα SPARQL (σύμφωνα με το συντακτικό της SPARQL, οι μεταβλητές εμφανίζονται να έχουν ως πρόθεμα τον χαρακτήρα `?` ή τον χαρακτήρα `$`):

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/ (Επιλέξτε `Control Panel` και μετά `Select` προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name (COUNT(?friend) AS ?count)
WHERE {
    ?person foaf:name ?name .
    ?person foaf:knows ?friend .
} GROUP BY ?person ?name
```

Το έγγραφο SPARQL 1.1 Query Language ορίζει τη σύνταξη και τη σημασιολογία των ερωτημάτων SPARQL 1.1, ενώ ταυτόχρονα προσφέρει διάφορα παραδείγματα χρήσης τους. Σε επόμενη ενότητα θα επιχειρήσουμε μία λεπτομερή παρουσίαση των προδιαγραφών αυτών.

5.1.2 Μορφότυπα αποτελεσμάτων αναζήτησης σε SPARQL 1.1 (XML, JSON, CSV, TSV)

Τα αποτελέσματα σε ερωτήματα τύπου SELECT στη SPARQL δεν είναι τίποτα άλλο παρά σύνολα αντιστοίχησης μεταβλητών σε δεδομένα RDF, τα οποία πολλές φορές αναπαρίστανται ως πίνακες. Για παράδειγμα, το ερώτημα της προηγούμενης ενότητας επιστρέφει τα ακόλουθα αποτελέσματα:

Πίνακας 5.1: Αποτελέσματα απλού ερωτήματος SPARQL

?name	?count
"Alice"	3
"Bob"	1
"Charlie"	1

Για να μπορέσουμε να εκφράσουμε τα παραπάνω αποτελέσματα σε μηχαναγνώσιμη μορφή, η SPARQL 1.1 υποστηρίζει τέσσερα διαδεδομένα μορφότυπα ανταλλαγής δεδομένων: Extensible Markup Language - XML, JavaScript Object Notation - JSON, comma-separated values - CSV και tab-separated values - TSV. Τα μορφότυπα αυτά περιγράφονται σε τρία διαφορετικά έγγραφα:

- SPARQL Query Results XML Format
- SPARQL 1.1 Query Results JSON Format
- SPARQL 1.1 Query Results CSV and TSV Formats

Τα παραπάνω έγγραφα περιγράφουν όλες τις περιπτώσεις έκφρασης αποτελεσμάτων σε ερωτήματα SPARQL. Τα αποτελέσματα στον πίνακα 5.1 εκφράζονται στα υποστηριζόμενα μορφότυπα της SPARQL ως ακολούθως:

XML

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>
   <variable name="name"/>
   <variable name="count"/>
 </head>
 <results>
   <result>
     <binding name="name">
       <literal>Alice</literal>
     </binding>
     <binding name="count">
```

```
      <literal datatype="http://www.w3.org/2001/XMLSchema#integer">3</literal>
    </binding>
  </result>
  <result>
    <binding name="name">
      <literal>Bob</literal>
    </binding>
    <binding name="count">
      <literal datatype="http://www.w3.org/2001/XMLSchema#integer">1</literal>
    </binding>
  </result>
  <result>
    <binding name="name">
      <literal>Charlie</literal>
    </binding>
    <binding name="count">
      <literal datatype="http://www.w3.org/2001/XMLSchema#integer">1</literal>
    </binding>
  </result>
 </results>
</sparql>
```

JSON

```
{
  "head": {
    "vars": [ "name" , "count" ]
  } ,
  "results": {
    "bindings": [
      {
        "name": { "type": "literal" , "value": "Alice" } ,
        "count": { "datatype": "http://www.w3.org/2001/XMLSchema#integer"
↪   , "type": "typed-literal" , "value": "3" }
      } ,
      {
        "name": { "type": "literal" , "value": "Bob" } ,
        "count": { "datatype": "http://www.w3.org/2001/XMLSchema#integer"
↪   , "type": "typed-literal" , "value": "1" }
      } ,
      {
        "name": { "type": "literal" , "value": "Charlie" } ,
        "count": { "datatype": "http://www.w3.org/2001/XMLSchema#integer"
↪   , "type": "typed-literal" , "value": "1" }
      }
```

```
    ]
  }
}
```

CSV

```
name,count
Alice,3
Bob,1
Charlie,1
```

TSV

```
?name<TAB>?count
"Alice"<TAB>3
"Bob"<TAB>1
"Charlie"<TAB>1
```

Σημείωση

Ο χαρακτήρας που αντιστοιχεί στην εσοχή αναπαρίσταται ως `<TAB>` στο παραπάνω παράδειγμα.

5.1.3 Ομόσπονδα ερωτήματα SPARQL 1.1

Το έγγραφο SPARQL 1.1 Federated Query περιγράφει μια επέκταση των βασικών προδιαγραφών SPARQL 1.1 Query Language έτσι ώστε να είναι δυνατή η αποστολή υποερωτημάτων σε διαφορετικά τελικά σημεία SPARQL. Έτσι λοιπόν, στο παράδειγμά μας, θα μπορούσαμε να ρωτήσουμε αν μεταξύ των φίλων της Alice υπάρχει κάποιος/-α που να έχει το ίδιο όνομα με το URI `<http://dbpedia.org/resource/Snoopy>` στην DBpedia. Αυτό επιτυγχάνεται συνδυάζοντας ένα τοπικό ερώτημα για τα ονόματα των φίλων με ένα απομακρυσμένο ερώτημα στο τελικό σημείο SPARQL της DBpedia (http://dbpedia.org/sparql) σχετικό με το όνομα του URI `<http://dbpedia.org/resource/Snoopy>` χρησιμοποιώντας τη λέξη-κλειδί `SERVICE`:

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/ (Επιλέξτε `Control Panel` και μετά `Select` προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    <http://example.org/alice#me> foaf:knows ?x .
```

```
    ?x foaf:name ?name .
    SERVICE <http://dbpedia.org/sparql> {
↪   <http://dbpedia.org/resource/Snoopy> foaf:name ?name }
}
```

Το αποτέλεσμα που θα μας επιστρέψει είναι το εξής:

Πίνακας 5.2: Αποτελέσματα ερωτήματος SPARQL με τη λέξη-κλειδί `SERVICE`

?name
"Snoopy"@en

Εδώ, το πρώτο μέρος του ταιριάσματος στην εμβέλεια του `WHERE` αναζητείται στο τοπικό τελικό σημείο SPARQL, ενώ το δεύτερο μέρος (αυτό που έπεται της λέξης `SERVICE`) αναζητείται στο απομακρυσμένο τελικό σημείο SPARQL.

5.1.4 Συστήματα συμπερασματολογίας SPARQL 1.1

Θα μπορούσαμε να χρησιμοποιήσουμε τη SPARQL πάνω σε οντολογικές πληροφορίες της μορφής `RDF Schema` ή σε αξιώματα της `OWL` (γνωστά ως συστήματα συμπερασματολογίας). Για παράδειγμα, ας υποθέσουμε πως, εκτός των δεδομένων σχετικά με την Alice, δηλώνεται επιπρόσθετη οντολογική πληροφορία για το λεξιλόγιο `FOAF` στην υπηρεσία SPARQL του παραδείγματός μας (στη συνέχεια παρατίθεται μόνο ένα απόσπασμα):

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
...
foaf:name rdfs:subPropertyOf rdfs:label .
...
```

Το παρακάτω ερώτημα αναζητά ετικέτες για άτομα:

```
SELECT ?label
WHERE { ?person rdfs:label ?label }
```

Μια υπηρεσία SPARQL που δεν λαμβάνει υπόψη κάποιο ιδιαίτερο σύστημα συμπερασματολογίας δεν θα επιστρέψει αποτελέσματα στο παραπάνω ερώτημα. Αντίθετα, μια υπηρεσία SPARQL που λαμβάνει υπόψη το σύστημα συμπερασματολογίας `RDF Schema` θα επέστρεφε τα παρακάτω αποτελέσματα:

Πίνακας 5.3: Αποτελέσματα ερωτήματος SPARQL με σύστημα συμπερασματολογίας

?label
"Alice"
"Bob"
"Charlie"
"Snoopy"@en

καθώς η ιδιότητα `foaf:name` είναι υπο-ιδιότητα της ιδιότητας `rdfs:label`. Το έγγραφο SPARQL 1.1 Entailment Regimes ορίζει τις απαντήσεις που πρέπει να επιστρέφονται ανάλογα με το εκάστοτε σύστημα συμπερασματολογίας (RDF, RDF Schema, D-Entailment, OWL και RIF).

5.1.5 Γλώσσα ενημερώσεων SPARQL 1.1

Το έγγραφο SPARQL 1.1 Update Language ορίζει το συντακτικό και τη σημασιολογία των αιτήσεων ενημέρωσης της SPARQL 1.1. Παράλληλα, παρέχει διάφορα παραδείγματα για τη χρήση τους. Οι λειτουργίες ενημέρωσης μπορεί να συνίστανται από αρκετές συνεχόμενες αιτήσεις, ενώ εκτελούνται σε μία συλλογή από γράφους RDF που βρίσκονται σε ένα αποθετήριο RDF.

Στο παράδειγμά μας η παρακάτω αίτηση εισάγει στον εξ ορισμού γράφο RDF μία νέα φίλη της Alice με το όνομα Dorothy και στη συνέχεια διαγράφει όλα τα ονόματα των φίλων της Alice που έχουν την ετικέτα της αγγλικής γλώσσας:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT DATA { <http://www.example.org/alice#me> foaf:knows ?x .
              ?x foaf:name "Dorothy" . } ;
DELETE { ?person foaf:name ?mbox }
WHERE { <http://www.example.org/alice#me> foaf:knows ?person .
        ?person foaf:name ?name FILTER ( lang(?name) = "EN" ) .}
```

Όπως αναδεικνύεται από τη δεύτερη λειτουργία στο παραπάνω παράδειγμα, οι εισαγωγές και οι διαγραφές μπορεί να εξαρτώνται από τα αποτελέσματα ερωτημάτων στον γράφο. Η σύνταξη που χρησιμοποιείται στην εμβέλεια του `WHERE` περιγράφεται στο έγγραφο SPARQL 1.1 Query Language.

5.1.6 Πρωτόκολλο SPARQL 1.1 για RDF

Το έγγραφο SPARQL 1.1 Protocol for RDF ορίζει τον τρόπο μεταφοράς ερωτημάτων και ενημερώσεων SPARQL 1.1 σε υπηρεσίες SPARQL μέσω του πρωτοκόλλου HTTP.

Επίσης, το έγγραφο αυτό ορίζει την αντιστοιχία αιτήσεων/αποκρίσεων SPARQL σε αιτήσεις/αποκρίσεις HTTP POST και GET. Για παράδειγμα, το ερώτημα:

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/ (Επιλέξτε `Control Panel` και μετά `Select` προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name (COUNT(?friend) AS ?count)
WHERE {
    ?person foaf:name ?name .
    ?person foaf:knows ?friend .
} GROUP BY ?person ?name
```

αν ήταν να το απευθύνουμε σε μια υπηρεσία SPARQL διαθέσιμη στο `http://www.example.org/sparql`, σύμφωνα με τις αντίστοιχες προδιαγραφές θα έπρεπε να μετατραπεί ως εξής:

```
GET /sparql/?query=PREFIX%20foaf%3A%20%3Chttp%3A%2F%2Fxmlns.com%2Ffoaf
    %2F0.1%2F%3E%0ASELECT%20%3Fname%20%28COUNT%28%3Ffriend%29%20AS
    %20%3Fcount%29%0AWHERE%20%7B%20%0A%20%20%20%20%3Fperson
    %20foaf%3Aname%20%3Fname%20.%20%0A%20%20%20%20%3Fperson
    %20foaf%3Aknows%20%3Ffriend%20.%20%0A%7D
    %20GROUP%20BY%20%3Fperson%20%3Fname
HTTP/1.1
Host: www.example.org
User-agent: my-sparql-client/0.1
```

Το παραπάνω ερώτημα είναι `URL-encoded` έτσι ώστε να μπορούμε να το γράψουμε στη `διεύθυνση` του φυλλομετρητή μας. Για να κωδικοποιούμε/αποκωδικοποιούμε URLs μπορούμε να χρησιμοποιήσουμε πλειάδα online εργαλείων όπως URL Encode/Decode: http://www.url-encode-decode.com/.

5.1.7 Περιγραφή υπηρεσιών SPARQL 1.1

Το έγγραφο SPARQL 1.1 Service Description περιγράφει μια μέθοδο ανακάλυψης αλλά και ένα λεξιλόγιο RDF περιγραφής υπηρεσιών SPARQL που είναι διαθέσιμες μέσω του πρωτοκόλλου SPARQL για RDF. Σύμφωνα με το έγγραφο αυτό, ένα τελικό σημείο SPARQL, χωρίς περαιτέρω παραμέτρους (ερωτημάτων ή ενημερώσεων), πρέπει να επιστρέψει μία περιγραφή RDF για την παρεχόμενη υπηρεσία. Για παράδειγμα, η παρακάτω αίτηση HTTP:

```
GET /sparql/ HTTP/1.1
Host: www.example.org
```

στο τελικό σημείο SPARQL που βρίσκεται στη διεύθυνση `http://www.example.org/sparql/` πρέπει να επιστρέψει μία περιγραφή με βάση το λεξιλόγιο RDF περιγραφής URI υπηρεσιών SPARQL. Η απόκριση οφείλει να παρέχει πληροφορίες σχετικά με τον εξ ορισμού γράφο ή πληροφορίες σχετικά με τα γνωρίσματα της γλώσσας SPARQL και τα συστήματα συμπερασματολογίας που υποστηρίζονται.

5.1.8 Πρωτόκολλο HTTP για αποθετήρια γράφου SPARQL 1.1

Πολλές εφαρμογές και υπηρεσίες που σχετίζονται με δεδομένα RDF δεν έχουν ανάγκη από το πλήρες σύνολο προδιαγραφών του εγγράφου SPARQL 1.1 Update. Έτσι λοιπόν το έγγραφο SPARQL 1.1 Graph Store HTTP Protocol υποδεικνύει τρόπους εκτέλεσης συγκεκριμένων λειτουργιών διαχείρισης γράφων RDF μέσω αιτήσεων HTTP. Για παράδειγμα, η ενημέρωση SPARQL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
INSERT DATA { <http://www.example.org/alice#me> foaf:knows [ foaf:name
↪  "Dorothy" ]. }
```

ουσιαστικά προσθέτει μία τριάδα σε έναν γράφο RDF. Εναλλακτικά, θα μπορούσαμε να επιτύχουμε την πρόσθεση αυτή μέσω της ακόλουθης αίτησης HTTP POST:

```
POST /rdf-graphs/service?graph=http%3A%2F%2Fwww.example.org%2Falice
↪  HTTP/1.1
Host: example.org
Content-Type: text/turtle

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://www.example.org/alice#me> foaf:knows [ foaf:name "Dorothy" ] .
```

Άλλες αιτήσεις HTTP που τροποποιούν (π.χ. χρήση HTTP PUT για αντικατάσταση ολόκληρου γράφου RDF, ή HTTP DELETE για διαγραφή ολόκληρου γράφου RDF) ή ανακτούν (μέσω HTTP GET) γράφους RDF, περιγράφονται στο έγγραφο SPARQL 1.1 Graph Store HTTP Protocol. Οι αιτήσεις αυτές μαζί με τις γλώσσες SPARQL 1.1 Query και SPARQL 1.1 Update μπορούν να θεωρηθούν ως ένα πιο «ελαφρύ» πρωτόκολλο SPARQL 1.1.

Στην επόμενη ενότητα θα εστιάσουμε στις προδιαγραφές της γλώσσας ερωτημάτων SPARQL 1.1.

5.2 Μια πιο αναλυτική ματιά στη γλώσσα ερωτημάτων SPARQL 1.1

Ίσως το πιο σημαντικό στάδιο στην ανάκτηση πληροφοριών μέσω του προτύπου SPARQL είναι η διατύπωση του κατάλληλου ερωτήματος. Ας αρχίσουμε την περιπλάνησή μας σε αυτό το πρότυπο αναλύοντας τη δομή ενός ερωτήματος SPARQL [1]. Έτσι λοιπόν, ένα ερώτημα SPARQL αποτελείται από:

- *δηλώσεις προθεμάτων*, που χρησιμεύουν στη συντομογραφία των χρησιμοποιούμενων URIs
- *δηλώσεις συνόλων δεδομένων*, που δηλώνουν τους γράφους RDF που χρησιμοποιούνται,
- *πρόταση αποτελέσματος*, που προσδιορίζει το αποτέλεσμα που θέλουμε να επιστρέψει το ερώτημα
- *μοτίβο ερωτήματος*, που ορίζει το ζητούμενο από το υποκείμενο σύνολο δεδομένων
- *μετατροπείς ερωτημάτων*, που μετασχηματίζουν τα αποτελέσματα αναζήτησης.

```
# Δηλώσεις προθεμάτων
PREFIX foo: <http://example.com/resources/>
...
# Δηλώσεις συνόλων δεδομένων
FROM ...
# Πρόταση αποτελέσματος
SELECT ...
# Μοτίβο ερωτήματος
WHERE {
    ...
}
# Μετατροπείς ερωτημάτων
ORDER BY ...
```

Τα ερωτήματα SPARQL απευθύνονται σε σύνολα δεδομένων RDF, που απαρτίζονται από γράφους RDF (παρακάτω θα δούμε περισσότερες λεπτομέρειες). Επίσης, ένα τελικό σημείο SPARQL δέχεται ερωτήματα SPARQL και επιστρέφει τα αντίστοιχα αποτελέσματα μέσω του πρωτοκόλλου HTTP. Όπως έχουμε ήδη αναφέρει, τα αποτελέσματα SPARQL εκφράζονται σε διάφορες μορφές (π.χ., json, csv, tsv, κτλ.).

5.2.1 Ένα απλό ερώτημα SPARQL

Έστω ότι σε ένα αποθετήριο τριάδων RDF έχουμε φορτώσει πληροφορίες σχετικές με το κοινωνικό δίκτυο του Tim Berners-Lee, όπως αυτές διατίθενται μέσω του λεξιλογίου foaf: http://dig/csail.mit.edu/2008/webdav/timbl/foaf.rdf. Όσον αφορά το πρώτο μας ερώτημα, ας υποθέσουμε ότι ψάχνουμε να βρούμε τα ονόματα των ανθρώπων που αναφέρονται στα παρακάτω δεδομένα RDF (δηλ. Timothy Berners-Lee, Henry Story, Lee Feigenbaum, Amy van der Hiel, κτλ.):

```
@prefix card: <http://www.w3.org/People/Berners-Lee/card#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

card:i rdf:type foaf:Person .
card:i foaf:name "Timothy Berners-Lee" .
<http://bblfish.net/people/henry/card#me> foaf:name "Henry Story" .
<http://www.cambridgesemantics.com/people/about/lee> foaf:name "Lee
↪  Feigenbaum" .
card:i foaf:knows card:amy .
card:amy foaf:name "Amy van der Hiel" .
...
```

Ουσιαστικά, στον γράφο http://dig/csail.mit.edu/2008/webdav/timbl/foaf.rdf αναζητούμε όλα τα υποκείμενα (?person) και τα αντικείμενα (?name) που συνδέονται μέσω του κατηγορήματος foaf:name. Τέλος, θέλουμε να μας επιστραφούν όλες οι τιμές της μεταβλητής ?name. Με άλλα λόγια, θέλουμε όλα τα ονόματα που αναφέρονται στο foaf αρχείο του Tim Berners-Lee:

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/ (Επιλέξτε `Control Panel` και μετά `Select` προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`).

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

Όπως έχουμε ήδη αναφέρει, οι μεταβλητές SPARQL αρχίζουν με το ερωτηματικό ? και ταυτίζονται με οποιοδήποτε κόμβο (URI ή/και λεκτικό) στο σύνολο δεδομένων RDF. Τα μοτίβα τριάδων είναι τριάδες που στη θέση κάποιου ή κάποιων από τα μέρη τους (δηλ. υποκείμενο, κατηγόρημα ή αντικείμενο) υπάρχει μεταβλητή. Η πρόταση αποτελέσματος SELECT επιστρέφει έναν πίνακα μεταβλητών ή/και τιμών που ικανοποιούν το ερώτημα.

5.2.2 Πολλαπλά μοτίβα τριάδων

Έστω ότι θέλουμε να ανακτήσουμε όλα τα άτομα στο αρχείο foaf του Tim Berners-Lee που έχουν email και όνομα. Έτσι, θα μας επιστραφούν για τα άτομα αυτά το URI τους (?person), το όνομα (?name) και το email τους (?email):

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/ (Επιλέξτε `Control Panel` και μετά `Select` προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`).

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

Αν θέλουμε να μας επιστραφούν όλες οι δηλωμένες μεταβλητές του ερωτήματός μας, μπορούμε να χρησιμοποιήσουμε το *.

5.2.3 Πολλαπλά μοτίβα τριάδων που διασχίζουν τον γράφο RDF

Έστω ότι θέλουμε να βρούμε την προσωπική ιστοσελίδα όλων των γνωστών του Tim Berners-Lee:

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/ (Επιλέξτε `Control Panel` και μετά `Select` προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`).

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX card: <http://www.w3.org/People/Berners-Lee/card#>
SELECT ?homepage

WHERE {
    card:i foaf:knows ?known .
    ?known foaf:homepage ?homepage .
}
```

Χρησιμοποιώντας τη μεταβλητή ?known ως αντικείμενο στο πρώτο μοτίβο τριάδας και υποκείμενο στο δεύτερο μοτίβο τριάδας, διασχίζουμε πολλαπλούς κόμβους στον γράφο RDF, όπως φαίνεται στο σχήμα 5.1.

Σχήμα 5.1: Διάσχιση γράφου RDF

5.2.4 Μετατροπείς ερωτημάτων

Προκειμένου να παρουσιάσουμε τους μετατροπείς ερωτημάτων της SPARQL, θα χρειαστούμε να απευθύνουμε ερωτήματα στο αποθετήριο τριάδων της DBpedia. Ας πούμε λίγα πράγματα γι' αυτό το αποθετήριο.

Αποθετήριο τριάδων RDF: DBpedia

Η DBpedia αποτελεί μια έκδοση της Wikipedia σε μορφή RDF. Τα δεδομένα της DBpedia προέρχονται από συγκεκριμένες δομές της Wikipedia όπως infoboxes, κατηγορίες, περιλήψεις λημμάτων και διάφορους εξωτερικούς συνδέσμους. Η DBpedia περιλαμβάνει πάνω από 100 εκατομμύρια τριάδες. Αν θέλαμε να ανακτήσουμε 100 κλάσεις στο σύνολο δεδομένων της DBpedia, θα έπρεπε να γράψουμε το εξής ερώτημα:

Δοκιμάστε το ερώτημα: http://dbpedia.org/sparql

```
SELECT DISTINCT ?concept
WHERE {
    ?s a ?concept .
} LIMIT 100
```

Η λέξη `LIMIT` αντιστοιχεί σε έναν μετατροπέα που περιορίζει το πλήθος των αποτελεσμάτων που επιστρέφονται. Η SPARQL υποστηρίζει άλλους δύο μετατροπείς:

- `ORDER BY`, για την κατάταξη των αποτελεσμάτων σύμφωνα με την τιμή μίας η περισσότερων μεταβλητών
- `OFFSET`, ο οποίος μαζί με τους μετατροπείς `LIMIT` και `ORDER BY` επιστρέφει ένα τμήμα του συνόλου αποτελεσμάτων (ιδιαίτερα χρήσιμο στην περίπτωση που θέλουμε να επιτύχουμε σελιδοποίηση αποτελεσμάτων).

Όπως έχουμε ήδη αναφέρει σε προηγούμενο κεφάλαιο, η λέξη-κλειδί `a` αποτελεί σύντμηση για το κατηγόρημα rdf:type, το οποίο επιστρέφει την κλάση ενός πόρου.

Η λέξη `DISTINCT` αντιστοιχεί σε έναν μετατροπέα που εξαλείφει τις διπλοεγγραφές από τα αποτελέσματα.

5.2.5 Βασικά φίλτρα SPARQL

Έστω ότι θέλουμε να βρούμε όλες τις χώρες με πληθυσμό άνω των 15 εκατομμυρίων κατοίκων.

Δοκιμάστε το ερώτημα: http://dbpedia.org/sparql

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT DISTINCT ?country_name ?population
WHERE {
    ?country a type:LandlockedCountries ;
             rdfs:label ?country_name ;
             prop:populationEstimate ?population .
    FILTER (?population > 15000000) .
}
```

Η λέξη `FILTER` επιτρέπει τη χρήση δυαδικών συνθηκών για το φιλτράρισμα μη επιθυμητών αποτελεσμάτων. Υπενθυμίζουμε ότι η χρήση του `;` σημαίνει τη χρήση του υποκειμένου της προηγούμενης τριάδας στην επόμενη. Επίσης, το κατηγόρημα rdfs:label αποτελεί ένα ευρέως διαδεδομένο κατηγόρημα για την απόδοση μίας ανθρωποαναγνώσιμης ετικέτας σε έναν πόρο. Με μια προσεκτική ματιά στα επιστρεφόμενα αποτελέσματα, διαπιστώνουμε την ύπαρξη πολλών διπλοεγγραφών σε διάφορες γλώσσες. Αυτό είναι ένα ζήτημα που αντιμετωπίζεται με τη χρήση της κατάλληλης συνάρτησης. Η SPARQL υποστηρίζει πλήθος συναρτήσεων, τις οποίες παραθέτουμε ανά είδος:

- λογικές: `!, &&, ||`
- μαθηματικές: `+, -, *, /, abs, round, ceil, floor, RAND`
- συγκριτικές: `=, !=, >, <, IN, NOT IN...`
- ελέγχου: `isURI, isBlank, isLiteral, isNumeric, bound`
- εκτίμησης: `str, lang, datatype`
- άλλες: `sameTerm, langMatches, regex, REPLACE`
- συνθήκης: `IF, COALESCE, EXISTS, NOT EXISTS`
- δημιουργίας: `URI, BNODE, STRDT, STRLANG, UUID, STRUUID`
- αλφαριθμητικές: `STRLEN, SUBSTR, UCASE, LCASE, STRSTARTS, STRENDS, CONTAINS, STRBEFORE, STRAFTER, CONCAT, ENCODE_FOR_URI`
- ημερομηνίας: `now, year, month, day, hours, minutes, seconds, timezone, tz`

- κατακερματισμού: `MD5, SHA1, SHA256, SHA384, SHA512`

Στο παρακάτω παράδειγμα, θα χρησιμοποιήσουμε τη συνάρτηση `langMatches` προκειμένου να φιλτράρουμε την αγγλική απόδοση των χωρών με πληθυσμό άνω των δεκαπέντε εκατομμυρίων κατοίκων. Η συνάρτηση `lang` φιλτράρει την ετικέτα γλώσσας από το ταίριασμα μίας μεταβλητής. Τέλος, θα τις ταξινομήσουμε με βάση τον πληθυσμό:

Δοκιμάστε το ερώτημα: http://dbpedia.org/sparql

```
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT DISTINCT ?country_name ?population

WHERE {
    ?country a type:LandlockedCountries ;
             rdfs:label ?country_name ;
             prop:populationEstimate ?population .
    FILTER (?population > 15000000 &&
            langMatches(lang(?country_name), "EN")) .
} ORDER BY DESC(?population)
```

5.2.6 Ο ρόλος των λέξεων OPTIONAL και UNION στα ερωτήματα SPARQL

Για να παρουσιάσουμε καλύτερα τη χρήση της λέξης OPTIONAL στα ερωτήματα SPARQL, θα χρειαστεί να χρησιμοποιήσουμε το αποθετήριο τριάδων RDF Bio2RDF. Ας κάνουμε μια μικρή παρουσίαση αυτού του αποθετηρίου.

Αποθετήριο τριάδων RDF: Bio2RDF

Ο οργανισμός bio2rdf.org φιλοξενεί 19 σύνολα δεδομένων των ανθρωπιστικών επιστημών που αριθμούν πάνω από ένα δισεκατομμύριο τριάδες. Τα σύνολα αυτά (http://download.bio2rdf.org/release/2/release.html) περιλαμβάνουν διασυνδεδεμένα δεδομένα για γονίδια, πρωτεΐνες, βιολογικά μοντέλα, φάρμακα, αλληλεπιδράσεις γονιδίων/πρωτεϊνών, γονιδιωματική, κ.ά. Ας δοκιμάσουμε να απευθύνουμε ένα ερώτημα στο τελικό σημείο SPARQL: DrugBank (http://drugbank.bio2rdf.org/sparql) που θα μας επιστρέψει όλα τα φάρμακα μαζί με τη δοσολογία τους αλλά και τα αντίστοιχα συμπτώματα. Μια πρώτη σκέψη είναι να απευθύνουμε το εξής ερώτημα:

Δοκιμάστε το ερώτημα: http://drugbank.bio2rdf.org/sparql

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://bio2rdf.org/drugbank_vocabulary:>
SELECT ?drug_name ?dosage ?indication
```

```
WHERE {
    ?drug a db:Drug .
    ?drug rdfs:label ?drug_name .
    ?drug db:dosage ?dosage .
    ?drug db:indication ?indication .
}
```

Το παραπάνω ερώτημα θα μας επιστρέψει γύρω στα 3.000 φάρμακα. Αν αφαιρέσουμε από το παραπάνω ερώτημα τα δύο μοτίβα τριάδων για τη δοσολογία και τα συμπτώματα αντίστοιχα, το τροποποιημένο ερώτημα:

Δοκιμάστε το ερώτημα: http://drugbank.bio2rdf.org/sparql

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://bio2rdf.org/drugbank_vocabulary:>
SELECT ?drug_name
WHERE {
    ?drug a db:Drug .
    ?drug rdfs:label ?drug_name .
}
```

θα μας επιστρέψει γύρω στα 7.700 αποτελέσματα. Αυτή η απόκλιση στο πλήθος των επιστρεφόμενων αποτελεσμάτων οφείλεται στο γεγονός ότι δεν έχουν όλα τα φάρμακα πληροφορίες για δοσολογία και συμπτώματα. Η προσθήκη της λέξης `OPTIONAL` στο μοτίβο ερωτήματος επιτρέπει το ταίριασμα φαρμάκων που δεν συμμετέχουν αναγκαστικά σε τριάδες οι οποίες αναφέρονται σε δοσολογία ή/και συμπτώματα:

Δοκιμάστε το ερώτημα: http://drugbank.bio2rdf.org/sparql

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://bio2rdf.org/drugbank_vocabulary:>
SELECT ?drug_name ?dosage ?indication
WHERE {
    ?drug a db:Drug .
    ?drug rdfs:label ?drug_name .
    OPTIONAL { ?drug db:dosage ?dosage . }
    OPTIONAL { ?drug db:indication ?indication . }
}
```

Αν ένα μοτίβο τριάδων που αρχίζει με τη λέξη `OPTIONAL` αποτύχει να ταυτιστεί με κάποια τριάδα στον γράφο, οι μεταβλητές που συμμετέχουν σε αυτό παραμένουν χωρίς τιμή (`unbound`).

Έστω ότι θέλουμε να βρούμε όλα τα ένζυμα και τα φάρμακα που υπάρχουν στο αποθετήριο τριάδων Bio2RDF:

Δοκιμάστε το ερώτημα: http://drugbank.bio2rdf.org/sparql

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://bio2rdf.org/drugbank_vocabulary:>
SELECT distinct ?label
WHERE {
{?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 ↪  <http://bio2rdf.org/drugbank_vocabulary:Enzyme>}
UNION
{?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 ↪  <http://bio2rdf.org/drugbank_vocabulary:Drug>}
?s rdfs:label ?label}
```

Η λέξη `UNION` επιτρέπει να ζητήσουμε τριάδες οι οποίες να ικανοποιούν και τα δύο μοτίβα τριάδων που συνδέονται μέσω της λέξης αυτής. Με άλλα λόγια, αποτελεί την «ένωση» αυτών των δύο μοτίβων.

5.2.7 Αποθετήρια RDF και γράφοι RDF

Όπως έχουμε ήδη αναφέρει σε αυτό το κεφάλαιο, τα ερωτήματα SPARQL απευθύνονται σε αποθετήρια τριάδων RDF που αποτελούνται από γράφους RDF. Κάθε ερώτημα SPARQL μπορεί να απευθύνεται στον εξ ορισμού γράφο RDF ή/και σε έναν ή περισσότερους επώνυμους γράφους RDF.

Επώνυμοι γράφοι

Ο επώνυμος γράφος (named graph) αναφέρεται μέσω της χρήσης της λέξης `FROM NAMED`. Όπως είδαμε στο δεύτερο κεφάλαιο, επώνυμους γράφους έχουμε μόνο σε αποθετήρια RDF που περιλαμβάνουν τετράδες RDF. Πιο συγκεκριμένα, σε ένα αποθετήριο τετράδων RDF το URI που ακολουθεί τη λέξη `FROM NAMED` αντιπροσωπεύει τις τετράδες του αποθετηρίου που έχουν στην τέταρτη θέση αυτό το URI. Σε ένα αποθετήριο τριάδων RDF δεν έχουμε επώνυμους γράφους.

Εξ ορισμού γράφος

Αντίστοιχα, σε ένα ερώτημα SPARQL, ο εξ ορισμού γράφος (default graph) αναφέρεται μέσω της χρήσης της λέξης `FROM`. Πιο συγκεκριμένα, σε ένα αποθετήριο τετράδων RDF το URI που ακολουθεί τη λέξη `FROM` αντιπροσωπεύει τις τετράδες που έχουν στην τέταρτη θέση αυτό το URI. Σε ένα αποθετήριο τριάδων RDF ο εξ ορισμού γράφος περιλαμβάνει όλες τις τριάδες του αποθετηρίου. Μέχρι τώρα όλα τα ερωτήματα που παραθέσαμε απευθύνονταν στον εξ ορισμού γράφο RDF.

Με τις λέξεις `FROM` και `FROM NAMED` σε ένα ερώτημα SPARQL προσδιορίζουμε το υποσύνολο των τριάδων του αποθετηρίου στο οποίο θα προσπαθήσουν να ταιριάξουν τα μοτίβα του ερωτήματος. Με τη λέξη `GRAPH` σε ένα ερώτημα SPARQL προσδιορίζουμε τον γράφο (εξ ορισμού ή επώνυμο) στον οποίο θα προσπαθήσει να ταιριάξει ένα συγκεκριμένο μοτίβο τριάδας. Οποιοδήποτε μοτίβο τριάδων εκτός της εμβέλειας της λέξης `GRAPH` προσπαθεί να ταιριάξει στον εξ ορισμού γράφο όπως φαίνεται παρακάτω παράδειγμα:

```
PREFIX ex: <http://www.example.org/>
SELECT #…
FROM ex:g1
FROM ex:g4
FROM NAMED ex:g1
FROM NAMED ex:g2
FROM NAMED ex:g3
WHERE {
 # … A …
GRAPH ex:g3 {
 # … B …
}
GRAPH ?graph {
 # … C …
}
}
```

Αν ένα URI αναφέρεται τόσο σε δήλωση `NAMED` όσο σε δήλωση `FROM NAMED`, αυτό σημαίνει ότι οι αντίστοιχες τετράδες στο αποθετήριο συμμετέχουν τόσο στον εξ ορισμού όσο και σε έναν επώνυμο γράφο. Έτσι λοιπόν, όπως φαίνεται στο παράδειγμα, τα μοτίβα που βρίσκονται στη θέση `... A ...` θα προσπαθήσουν να ταιριάξουν με τριάδες του εξ ορισμού γράφου που αποτελείται από τα URI `ex:g1` και `ex:g4`, ενώ τα μοτίβα που βρίσκονται στη θέση `... C ...` θα προσπαθήσουν να ταιριάξουν με τριάδες όλων των επώνυμων γράφων του αποθετηρίου που αντιστοιχούν στα URI `ex:g1`, `ex:g2` και `ex:g3`.

Για να εξηγήσουμε καλύτερα τον ρόλο της λέξης GRAPH στα ερωτήματα SPARQL, θα χρειαστεί να παρουσιάσουμε το αποθετήριο τριάδων `semanticweb.org`.

Αποθετήριο τριάδων RDF: semanticweb.org

Ο ιστότοπος data.semanticweb.org φιλοξενεί δεδομένα RDF που σχετίζονται με εργαστήρια, γεγονότα, προγράμματα και ομιλητές των δύο βασικών συνεδρίων του σημασιολογικού ιστού: International Semantic Web Conference (ISWC) και European Semantic Web Conference (ESWC). Τα δεδομένα αυτά παρουσιάζονται μέσω των οντολογιών FOAF (http://xmlns.com/foaf/0.1/), SWRC (http://ontoware.org/swrc/) και iCal (rfc2445). Τα δεδομένα για κάθε ξεχωριστό συνέδριο ISWC/ESWC είναι αποθηκευμένα σε ξεχωριστούς επώνυμους γράφους. Δεδομένα που δεν σχετίζονται αποκλειστικά με ένα συνέδριο (π.χ. ονόματα συμμετεχόντων) είναι αποθηκευμένα στον εξ ορισμού γράφο. Περισσότερες λεπτομέρειες γιαυτό το αποθετήριο θα αναφέρουμε σε μία ενότητα στο τελευταίο κεφάλαιο του βιβλίου. Έστω ότι θέλουμε να βρούμε τα άτομα που έχουν συμμετάσχει σε τρία συνέδρια ESWC (2013,2014,2015):

Δοκιμάστε το ερώτημα: http://data.semanticweb.org/snorql/

```
SELECT DISTINCT ?name
WHERE {
```

```
    ?person foaf:name ?name .
    GRAPH <http://data.semanticweb.org/conference/eswc/2015/complete> {
↪   ?person a foaf:Person }
    GRAPH <http://data.semanticweb.org/conference/eswc/2014/complete> {
↪   ?person a foaf:Person }
    GRAPH <http://data.semanticweb.org/conference/eswc/2013/complete> {
↪   ?person a foaf:Person }
}
```

Η λέξη GRAPH εξασφαλίζει την ταύτιση μοτίβων τριάδων με τριάδες του αντίστοιχου γράφου RDF. Το περιβάλλον διεπαφής αυτού του τελικού σημείου SPARQL έχει ορίσει εκ των προτέρων το λεξιλόγιο foaf, γι' αυτό και δεν το συμπεριλαμβάνουμε στα προθέματα του παραπάνω ερωτήματος.

5.2.8 Μετασχηματισμός λεξιλογίων

Η λέξη-κλειδί CONSTRUCT αποτελεί εναλλακτική πρόταση αποτελεσμάτων της λέξης SELECT. Αντί να επιστρέφεται ένας πίνακας με τιμές αποτελεσμάτων, η CONSTRUCT επιστρέφει έναν γράφο RDF (δηλ. τριάδες). Ο γράφος αυτός δημιουργείται ως εξής: τα αποτελέσματα του ισοδύναμου ερωτήματος SELECT χρησιμοποιούνται έτσι ώστε να συμπληρωθούν οι τιμές των μεταβλητών στη δομή CONSTRUCT. Δε δημιουργούνται τριάδες στον γράφο όταν υπάρχουν μεταβλητές χωρίς τιμή. Για να μετασχηματίσουμε δεδομένα foaf σε δεδομένα VCard απευθύνουμε το εξής ερώτημα:

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/. (Επιλέξτε `Control Panel` και μετά `Select`, προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`.)

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT { ?X vCard:FN ?name .
            ?X vCard:URL ?url .
            ?X vCard:TITLE ?title . }
WHERE {
  OPTIONAL { ?X foaf:name ?name . FILTER isLiteral(?name) . }
  OPTIONAL { ?X foaf:homepage ?url . FILTER isURI(?url) . }
  OPTIONAL { ?X foaf:title ?title . FILTER isLiteral(?title) . }
}
```

5.2.9 Ερωτήματα Ναι/Όχι

Έστω ότι θέλουμε να μάθουμε αν ο Αμαζόνιος είναι μακρύτερος από τον Νείλο:

Δοκιμάστε το ερώτημα: http://dbpedia.org/sparql

```
PREFIX prop: <http://dbpedia.org/property/>
ASK
{
  <http://dbpedia.org/resource/Amazon_River> prop:length ?amazon .
  <http://dbpedia.org/resource/Nile> prop:length ?nile .
  FILTER(?amazon > ?nile) .
}
```

Η λέξη ASK επιστρέφει true/false ανάλογα με το αν το ερώτημα ταυτίζεται με μία ή περισσότερες τριάδες στον γράφο RDF. Όπως σε κάθε ερώτημα SPARQL, η λέξη WHERE είναι προαιρετική.

5.2.10 Πληροφορίες γύρω από έναν πόρο

Έστω ότι θέλουμε να αντλήσουμε πληροφορίες γύρω από την Paris Hilton. Θα πρέπει να απευθύνουμε το εξής ερώτημα:

Δοκιμάστε το ερώτημα: http://dbpedia.org/sparql

```
DESCRIBE <http://dbpedia.org/resource/Paris_Hilton>
```

Η πρόταση αποτελεσμάτων DESCRIBE δίνει τη δυνατότητα στο τελικό σημείο SPARQL να επιστρέψει οτιδήποτε κρίνει πως είναι σχετικό με τον δοσμένο πόρο. Ακριβώς επειδή κάθε τελικό σημείο SPARQL είναι ελεύθερο να ερμηνεύσει το ερώτημα DESCRIBE κατά το δοκούν, τα ερωτήματα αυτού του τύπου δεν είναι διαλειτουργικά.

5.2.11 Συναθροίσεις (Aggregates)

Οι συναθροίσεις εφαρμόζουν συγκεκριμένες εκφράσεις πάνω σε σύνολα αποτελεσμάτων. Εξ ορισμού, ένα σύνολο αποτελεσμάτων αποτελείται από ένα ακριβώς σύνολο που περιέχει όλα τα αποτελέσματα. Η ομαδοποίηση προσδιορίζεται μέσα από τη χρήση της φράσης GROUP BY. Οι συναθροίσεις που ορίζονται στο πρότυπο της SPARQL 1.1 είναι οι εξής: COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT και SAMPLE. Χρησιμοποιούμε συναθροίσεις όταν θέλουμε να δούμε ακριβώς ένα αποτέλεσμα που υπολογίζεται μέσα από ένα σύνολο αποτελεσμάτων, όπως τη μέγιστη τιμή μίας μεταβλητής, αντί την τιμή κάθε μεταβλητής ξεχωριστά.

Έστω τα παρακάτω δεδομένα RDF:

```
@prefix books: <http://books.example/> .

books:org1 books:affiliates books:auth1, books:auth2 .
books:auth1 books:writesBook books:book1, books:book2 .
```

```
books:book1 books:price 9 .
books:book2 books:price 5 .
books:auth2 books:writesBook books:book3 .
books:book3 books:price 7 .
books:org2 books:affiliates books:auth3 .
books:auth3 books:writesBook books:book4 .
books:book4 books:price 7 .
```

Αν απευθύνουμε το εξής ερώτημα:

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/. (Επιλέξτε `Control Panel` και μετά `Select`, προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`.)

```
PREFIX books: <http://books.example/>

SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org books:affiliates ?auth .
  ?auth books:writesBook ?book .
  ?book books:price ?lprice .
}
GROUP BY ?org
HAVING (SUM(?lprice) > 10)
```

θα μας επιστραφεί το αποτέλεσμα `totalPrice:21`. Στο παραπάνω παράδειγμα, η φράση `GROUP BY` χρησιμοποιείται για την ομαδοποίηση αποτελεσμάτων σύμφωνα με την έκφραση `?org`, η οποία θα μπορούσε να είναι και πιο περίπλοκη. Επίσης, η λέξη `HAVING` είναι ανάλογη με τη λέξη `FILTER` που είδαμε πιο πριν, μόνο που χρησιμοποιείται όταν έχουμε ομάδες αντί για μεμονωμένα αποτελέσματα. Πιο συγκεκριμένα, στο παραπάνω παράδειγμα τοποθετούμε στην ίδια ομάδα όλα τα αποτελέσματα που έχουν την ίδια τιμή στη μεταβλητή ?org και υπολογίζουμε τη συνάρτηση `SUM` για την ομάδα αυτή. Οι ομάδες που σχηματίζονται, φιλτράρονται μέσω της έκφρασης `HAVING`, η οποία αφαιρεί όλες τις ομάδες όπου η τιμή `SUM(?lprice)` δεν είναι μεγαλύτερη από 10.

5.2.12 Υποερωτήματα

Τα υποερωτήματα αποτελούν έναν τρόπο να συμπεριλάβουμε ερωτήματα μέσα σε άλλα ερωτήματα. Κάτι τέτοιο είναι χρήσιμο σε διάφορες περιπτώσεις, όπως στον περιορισμό του αριθμού των αποτελεσμάτων από κάποια υποέκφραση μέσα στο ερώτημα. Στη SPARQL τα υποερωτήματα εκτελούνται πρώτα και τα αντίστοιχα αποτελέσματα προσφέρονται στο «εξωτερικό» ερώτημα.

Έστω τα παρακάτω δεδομένα RDF:

```
@prefix people: <http://people.example/> .

people:alice people:name "Alice", "Alice Foo", "A. Foo" .
people:alice people:knows people:bob, people:carol .
people:bob people:name "Bob", "Bob Bar", "B. Bar" .
people:carol people:name "Carol", "Carol Baz", "C. Baz" .
```

αν απευθύνουμε το εξής ερώτημα:

Δοκιμάστε το ερώτημα: http://sparql.lodbook.org/. (Επιλέξτε `Control Panel` και μετά `Select`, προκειμένου να φορτώσετε το σύνολο δεδομένων `ds`.)

```
PREFIX people: <http://people.example/>
SELECT ?y ?minName
WHERE {
  people:alice people:knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y people:name ?name .
    } GROUP BY ?y
  }
}
```

θα μας επιστραφούν τα αποτελέσματα:

Πίνακας 5.4: Αποτελέσματα συνολικού ερωτήματος SPARQL

y	minName
people:bob	"B. Bar"
people:carol	"C. Baz"

Τα αποτελέσματα αυτά προκύπτουν ως εξής: Αρχικά, υπολογίζεται το "εσωτερικό" ερώτημα:

```
SELECT ?y (MIN(?name) AS ?minName)
WHERE {
  ?y people:name ?name .
} GROUP BY ?y
```

το οποίο παρέχει τα εξής αποτελέσματα:

Πίνακας 5.5: Αποτελέσματα «εσωτερικού» ενθυλακωμένου ερωτήματος SPARQL

y	minName
people:alice	"A. Foo"
people:bob	"B. Bar"
people:carol	"C. Baz"

Κατόπιν, υπολογίζεται το «εξωτερικό» ερώτημα:

```
PREFIX people: <http://people.example/>
SELECT ?y ?minName
WHERE {
  people:alice people:knows ?y .
}
```

το οποίο παρέχει τα εξής αποτελέσματα:

Πίνακας 5.6: Αποτελέσματα «εξωτερικού» ενθυλακωμένου ερωτήματος SPARQL

y	minName
people:bob	
people:carol	

Η συναλήθευση (αλλιώς, η τομή) αυτών των δύο συνόλων αποτελεσμάτων συνιστά τα τελικά αποτελέσματα του ερωτήματος.

5.2.13 Η άρνηση στη SPARQL

Η SPARQL υποστηρίζει δύο στιλ άρνησης: Το πρώτο βασίζεται στο φιλτράρισμα αποτελεσμάτων έχοντας ως κριτήριο την ταύτιση ή όχι ενός μοτίβου γραφήματος. Το δεύτερο στιλ βασίζεται στην αφαίρεση αποτελεσμάτων έχοντας ως κριτήριο κάποιο μοτίβο.

- Το πρώτο στιλ άρνησης επιτυγχάνεται μέσω της χρήσης των λέξεων `EXISTS` και `NOT EXISTS` μέσα σε μία έκφραση `FILTER`. Για παράδειγμα, στα δεδομένα:

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

```
:alice  rdf:type   foaf:Person .
:alice  foaf:name  "Alice" .
:bob    rdf:type   foaf:Person .
```

το ερώτημα:

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

SELECT ?person
WHERE
{
    ?person rdf:type  foaf:Person .
    FILTER NOT EXISTS { ?person foaf:name ?name }
}
```

θα μας επιστρέψει το:

```
person: <http://example/bob>.
```

- Το δεύτερο στιλ άρνησης στη SPARQL βασίζεται στη λέξη `MINUS`. Η λέξη αυτή υπολογίζει αρχικά και τα δύο ορίσματά της, ενώ στη συνέχεια υπολογίζει τα αποτελέσματα στην αριστερή πλευρά που δεν είναι συμβατά με τα αποτελέσματα στη δεξιά πλευρά. Για παράδειγμα, στα δεδομένα:

```
@prefix :       <http://example/> .
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .

:alice  foaf:givenName "Alice" ;
        foaf:familyName "Smith" .

:bob    foaf:givenName "Bob" ;
        foaf:familyName "Jones" .

:carol  foaf:givenName "Carol" ;
        foaf:familyName "Smith" .
```

το ερώτημα:

```
PREFIX :       <http://example/>
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?s
```

```
WHERE {
   ?s ?p ?o .
   MINUS {
      ?s foaf:givenName "Bob" .
   }
}
```

θα μας επιστρέψει:

Πίνακας 5.7: Αποτελέσματα ερωτήματος SPARQL με τη λέξη-κλειδί `MINUS`

?s
http://example/carol
http://example/alice

5.2.14 Μονοπάτια κατηγορημάτων (Property paths)

Ένα μονοπάτι κατηγορημάτων (property path) ορίζει μια πιθανή διαδρομή μεταξύ δύο κόμβων μέσα σε ένα γράφο RDF. Το υποκείμενο και το αντικείμενο ενός ερωτήματος SPARQL που χρησιμοποιεί μονοπάτι κατηγορημάτων μπορεί να είναι ένας πόρος ή μια μεταβλητή. Δεν μπορούμε να χρησιμοποιήσουμε μεταβλητές στην έκφραση του μονοπατιού. Πρακτικά, ένα ερώτημα SPARQL που χρησιμοποιεί μονοπάτι κατηγορημάτων, περιέχει μια τριάδα που αποτελείται από το γνωστό μας υποκείμενο και αντικείμενο, ενώ στη θέση του κατηγορήματος υπάρχει το μονοπάτι κατηγορημάτων. Ένα μονοπάτι κατηγορημάτων μοιάζει με μια κανονική έκφραση (regular expression) μόνο που, αντί να αναφέρεται σε χαρακτήρες, αναφέρεται σε κατηγορήματα. Για παράδειγμα, το ερώτημα:

```
PREFIX  foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?name
WHERE
  { ?x foaf:mbox <mailto:alice@example> .
    ?x foaf:knows/foaf:name ?name
  }
```

επιστρέφει τα ονόματα των ατόμων που ξέρει η Alice. Εναλλακτικά, αυτό το ερώτημα θα μπορούσε να διατυπωθεί, χωρίς τη χρήση μονοπατιού κατηγορημάτων, ως εξής:

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

```
  { ?x foaf:mbox <mailto:alice@example> .
    ?x foaf:knows ?z .
    ?z foaf:name ?name
  }
```

Πολλές φορές θέλουμε να εντοπίσουμε ένα μονοπάτι μεταξύ δύο κόμβων στον γράφο RDF που να περνάει από κάποιο κατηγόρημα χωρίς να ξέρουμε εκ των προτέρων πόσες φορές επαναλαμβάνεται το κατηγόρημα αυτό στο μονοπάτι. Για παράδειγμα, έστω ότι θέλουμε τα ονόματα των ατόμων που συνδέονται με την Alice μέσω ενός η περισσότερων ατόμων:

```
PREFIX  foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?name
WHERE
  { ?x foaf:mbox <mailto:alice@example> .
    ?x foaf:knows+ ?z .
    OPTIONAL { ?z foaf:name ?name }
  }
```

Στο παραπάνω παράδειγμα, χρησιμοποιήσαμε τον τελεστή +, ο οποίος μεταφράζεται ως: «μια ή περισσότερες φορές». Ομοίως, ο τελεστής * (Kleene star) μεταφράζεται ως: «μηδέν ή περισσότερες φορές».

Στο σημείο αυτό πρέπει να τονίσουμε ότι η χρήση * ή + σε μονοπάτια κατηγορημάτων καταναλώνει μεγάλη υπολογιστική ισχύ, προκειμένου να βρεθούν τα μονοπάτια στον αντίστοιχο γράφο RDF. Έτσι η χρήση των συμβόλων αυτών σε μονοπάτια κατηγορημάτων δεν συνίσταται σε σύνολα δεδομένων RDF μεγάλου όγκου.

Τέλος, στα μονοπάτια κατηγορημάτων μπορούμε να χρησιμοποιήσουμε τον τελεστή |, ο οποίος ταυτίζει το ένα ή και τα δύο εναλλακτικά μονοπάτια που εκφράζει. Για παράδειγμα, το παρακάτω ερώτημα θα επιστρέψει τον τίτλο ενός βιβλίου χωρίς να ξέρει εκ των προτέρων αν αυτός έχει εκφραστεί σε τριάδα με το κατηγόρημα rdfs:label ή το κατηγόρημα dc:title:

```
SELECT  ?displayString
WHERE
  {
    :book1 dc:title|rdfs:label ?displayString
  }
```

5.3 Ελέγξτε τις γνώσεις σας

1. Ποια από τις παρακάτω προτάσεις είναι σωστή;

i. Η SPARQL 1.1 είναι ένα σύνολο προδιαγραφών που παρέχει γλώσσες και πρωτόκολλα ερωτήσεων και χειρισμού γράφων RDF στον παγκόσμιο ιστό.

ii. Η SPARQL 1.1 είναι ένα σύνολο προδιαγραφών που παρέχει γλώσσες και πρωτόκολλα ερωτήσεων και χειρισμού γράφων RDF σε αποθετήρια RDF.

iii. Η SPARQL 1.1 είναι ένα σύνολο προδιαγραφών που παρέχει γλώσσες και πρωτόκολλα ερωτήσεων και χειρισμού γράφων RDF τόσο στον παγκόσμιο ιστό όσο και σε αποθετήρια RDF.

2. Ποια λέξη-κλειδί χρησιμοποιούμε για να δημιουργήσουμε ομόσπονδα ερωτήματα στη SPARQL;

   i. SERVICE.

   ii. HAVING.

   iii. BIND.

3. Ποια είναι η σωστή σειρά αναφοράς των παρακάτω λέξεων-κλειδιών σε ένα ερώτημα SPARQL;

   i. (α) FROM ... (β) PREFIX ... (γ) SELECT ... (δ) WHERE ... (ε) ORDER BY ...

   ii. (α) PREFIX ... (β) FROM ... (γ) SELECT ... (δ) WHERE ... (ε) ORDER BY ...

   iii. (α) PREFIX ... (β) SELECT ... (γ) FROM ... (δ) WHERE ... (ε) ORDER BY ...

4. Ποιο ή ποια από τα παρακάτω σύμβολα μπορεί να χρησιμοποιηθεί ως πρόθεμα σε μεταβλητές στη SPARQL;

   i. $

   ii. ?

   iii. %

5. Με ποια λέξη-κλειδί μπορούμε να περιορίσουμε το πλήθος αποτελεσμάτων που θα μας επιστραφεί από ένα ερώτημα SPARQL;

   i. ONLY.

   ii. LIMIT.

   iii. JUST.

6. Με ποιο μοτίβο τριάδας είναι ισοδύναμο το: `?var rdf:type foaf:person`;

   i. `?var a foaf:person`

ii. `?var isA foaf:person`

iii. `?var rdf:class foaf:person`

7. Ποια η χρησιμότητα της λέξης-κλειδί OPTIONAL σε ένα ερώτημα SPARQL;

   i. Αν ένα μοτίβο τριάδων που αρχίζει με τη λέξη OPTIONAL αποτύχει να ταυτιστεί με κάποια τριάδα στον γράφο, οι μεταβλητές που συμμετέχουν σε αυτό λαμβάνουν μία τυχαία τιμή.

   ii. Αν ένα μοτίβο τριάδων που αρχίζει με τη λέξη OPTIONAL αποτύχει να ταυτιστεί με κάποια τριάδα στον γράφο, τότε αποτυγχάνει όλο το ερώτημα.

   iii. Αν ένα μοτίβο τριάδων που αρχίζει με τη λέξη OPTIONAL αποτύχει να ταυτιστεί με κάποια τριάδα στον γράφο, οι μεταβλητές που συμμετέχουν σε αυτό παραμένουν χωρίς τιμή.

8. Ποια από τις παρακάτω προτάσεις είναι σωστή;

   i. Η λέξη-κλειδί GRAPH επιτρέπει να εστιάσουμε το ερώτημα SPARQL σε κάποιο επώνυμο γράφο RDF.

   ii. Η λέξη-κλειδί GRAPH επιτρέπει να εστιάσουμε το ερώτημα SPARQL στον εξ ορισμού γράφο RDF.

   iii. Η λέξη-κλειδί GRAPH επιτρέπει να αποκλείσουμε το ερώτημα SPARQL από κάποιο επώνυμο γράφο RDF.

9. Τι είδους αποτελέσματα επιστρέφει η λέξη-κλειδί CONSTRUCT;

   i. Η λέξη-κλειδί CONSTRUCT επιστρέφει δεδομένα σε μορφή JSON.

   ii. Η λέξη-κλειδί CONSTRUCT επιστρέφει έναν πίνακα αποτελεσμάτων.

   iii. Η λέξη-κλειδί CONSTRUCT επιστρέφει έναν γράφο RDF (τριάδες).

10. Τι είδους αποτελέσματα επιστρέφει η λέξη-κλειδί ASK;

    i. Η λέξη-κλειδί ASK επιστρέφει δεδομένα σε μορφή JSON.

    ii. Η λέξη-κλειδί ASK επιστρέφει δεδομένα σε μορφή γράφου (τριάδες).

    iii. Η λέξη-κλειδί ASK επιστρέφει δεδομένα σε μορφή Ναι/Όχι.

11. Ποια από τις παρακάτω προτάσεις είναι σωστή;

    i. Για συγκεκριμένο πόρο, η πρόταση αποτελεσμάτων DESCRIBE επιστρέφει πάντα τις ίδιες πληροφορίες όταν απευθύνεται σε διαφορετικά τελικά σημεία SPARQL.

    ii. Για συγκεκριμένο πόρο, η πρόταση αποτελεσμάτων DESCRIBE δίνει τη δυνατότητα στο τελικό σημείο SPARQL να επιστρέψει οτιδήποτε κρίνει αυτό σχετικό με τον συγκεκριμένο πόρο.

iii. Η πρόταση αποτελεσμάτων DESCRIBE ευνοεί τη διαλειτουργικότητα μεταξύ διαφορετικών τελικών σημείων SPARQL.

12. Ποια λέξη-κλειδί χρησιμοποιούμε για να ονομάσουμε μία συνάθροιση σε ένα ερώτημα SPARQL;

    i. IS.
    ii. AS.
    iii. EQUALS.

13. Ποια λέξη-κλειδί δεν αναφέρεται στην άρνηση σε ένα ερώτημα SPARQL;

    i. MINUS.
    ii. NOT EXISTS.
    iii. EXCEPT.

14. Ποια από τις παρακάτω προτάσεις δεν είναι σωστή;

    i. Μπορούμε να χρησιμοποιήσουμε μεταβλητές στην έκφραση ενός μονοπατιού κατηγορημάτων.
    ii. Τα μονοπάτια κατηγορημάτων επιτρέπουν να διαπιστώσουμε τη συνδεσιμότητα δύο κόμβων σε έναν γράφο RDF.
    iii. Ένα μονοπάτι κατηγορημάτων μήκους 1 ταυτίζεται με ένα μοτίβο τριάδας.

15. Όταν χρησιμοποιούμε το σύμβολο ^ σε ένα μονοπάτι μονοπατιών, υποδηλώνουμε:

    i. αντίστροφο μονοπάτι.
    ii. σειριακό μονοπάτι.
    iii. εναλλακτικό μονοπάτι.

5.4 Σύνοψη

Στο κεφάλαιο αυτό παρουσιάσαμε το σύνολο προδιαγραφών του προτύπου SPARQL 1.1, ενώ δώσαμε ιδιαίτερη βαρύτητα στις προδιαγραφές εκείνες που αφορούν τη γλώσσα ερωτημάτων του προτύπου. Η γνώση αυτή δίνει τη δυνατότητα να απευθύνουμε ερωτήματα στα τελικά σημεία SPARQL διαφόρων αποθετηρίων τριάδων RDF που είναι διαθέσιμα online στη διεύθυνση http://lod-cloud.net. Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι η σχετικά πρόσφατη ανακοίνωση αυτού του προτύπου δεν έχει επιτρέψει στα συστήματα που το υποστηρίζουν να υλοποιήσουν το σύνολο των προδιαγραφών. Όσο περνάει ο καιρός είναι σίγουρο πως η υποστήριξη που θα παρέχουν τα συστήματα στο πρότυπο θα βελτιώνεται. Μπορείτε να ανατρέξετε στο αντίστοιχο λήμμα της Wikipedia προκειμένου να παρακολουθείτε την εξέλιξη των συστημάτων αυτών.

5.5 Βιβλιογραφικές αναφορές

[1] *SPARQL by Example*, διαθέσιμο από http://www.cambridgesemantics.com/semantic-university/sparql-by-example, ημερομηνία πρόσβασης: 13.7.2015.