

Εργαστήριο Σημασιολογικού Ιστού

Ενότητα 8: Εισαγωγή στη SPARQL – Βασική Χρήση

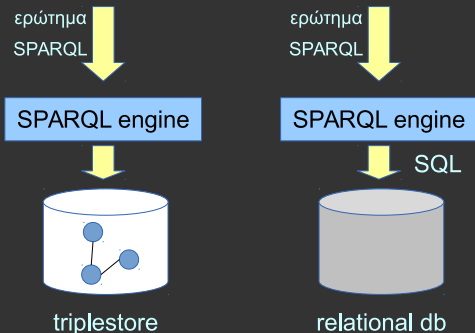
Μ.Στεφανιδάκης

5-4-2017

Η γλώσσα ερωτημάτων SPARQL

- ▶ Ερωτήσεις (και ενημερώσεις) σε σετ δεδομένων RDF
 - ▶ Και σε δεδομένα άλλης μορφής που μπορούν να αναπαρασταθούν ως τριάδες, **χωρίς απαραίτητα να είναι δεδομένα RDF!**
 - ▶ Δεδομένα από σχεσιακές βάσεις
 - ▶ Δεδομένα σε μορφή spreadsheet
 - ▶ **Και οτιδήποτε άλλο περικλείει σημασιολογική πληροφορία**
- ▶ Ένα “παράθυρο” σε βάσεις σημασιολογικών δεδομένων
 - ▶ Διεπαφή (interface) για αλληλεπίδραση με τις εφαρμογές χρήστη
 - ▶ Υλοποίηση από **“SPARQL engines”**

SPARQL engines



- ▶ Συνοδεύουν απαραίτητα βάσεις τριάδων RDF (**triplestores**)
- ▶ Και υπάρχουν συχνά ως ενδιάμεσο λογισμικό (**middleware**) σε σχεσιακές βάσεις ή άλλα σετ (μη RDF) δεδομένων
- ▶ Όταν μπορούν να προσπελαστούν μέσω Web, μιλάμε για **SPARQL endpoints**

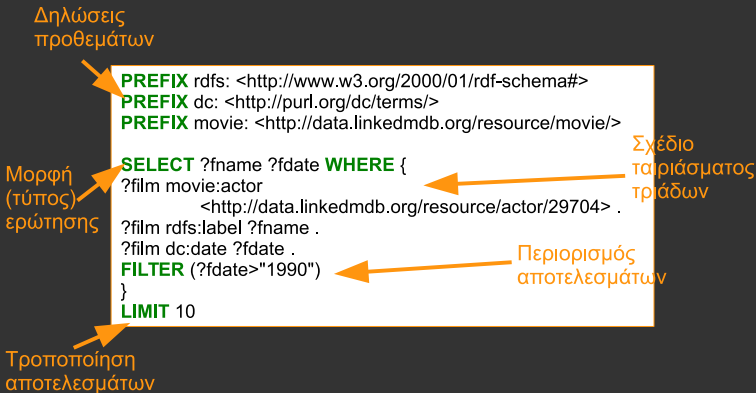
Το πρότυπο της γλώσσας SPARQL

- ▶ Τρέχουσα έκδοση: **SPARQL 1.1** (2013)
- ▶ Μια σειρά από προδιαγραφές που περιγράφουν
 - ▶ Τη σύνταξη και σημασιολογία των **ερωτημάτων** SPARQL
 - ▶ Τη μορφή των επιστρεφόμενων **απαντήσεων**
 - ▶ Τον τρόπο **αποστολής** ερωτημάτων μέσω HTTP
 - ▶ Τη σύνταξη των αιτημάτων **ενημέρωσης** των δεδομένων (update)
 - ▶ Πρόσθετα θέματα όπως
 - ▶ Τη μέθοδο περιγραφής **προσφερόμενων υπηρεσιών**
 - ▶ Τη λειτουργία διεξαγωγής **κατανεμημένων ερωτημάτων**
 - ▶ Τον τρόπο **συνεπαγωγής** πρόσθετης γνώσης (entailment regimes)

Η παλαιότερη έκδοση SPARQL 1.0 (2008)

- ▶ Περιγράφει μέρος μόνο της λειτουργικότητας της νεώτερης SPARQL 1.1
- ▶ Αλλά παραμένει σημαντική:
 - ▶ Είναι η ελάχιστη εγγυημένη λειτουργικότητα από κάθε SPARQL engine που θα συναντήσετε on-line...
 - ▶ Υπάρχει ακόμα λογισμικό που δεν υποστηρίζει τα νέα χαρακτηριστικά της SPARQL 1.1
 - ▶ Ακόμα κι αν έχουν βγει ενημερώσεις, κανείς δεν εγγυάται ότι αυτές έχουν εφαρμοστεί σε κάθε βάση RDF στο Web
 - ▶ Μια ευέλικτη εφαρμογή θα πρέπει να μπορεί να λειτουργήσει ικανοποιητικά χωρίς να βασίζεται απόλυτα στα νέα χαρακτηριστικά της SPARQL 1.1
 - ▶ Παρά μόνο χρησιμοποιώντας τα προσθετικά, ως βελτιώσεις, όταν είναι διαθέσιμα

Παράδειγμα ερωτήματος SELECT της SPARQL



- ▶ Η σύνταξη γενικά αγνοεί κενά και newlines, κεφαλαία-πεζά
- ▶ μόνο το ειδικό κατηγορημα **a** (rdf:type) πρέπει να γραφεί με πεζά
- ▶ Δοκιμάστε κι εσείς το πιο πάνω ερώτημα!

Βοηθητικές Συντομογραφίες

- ▶ Η SPARQL βασίζεται στη σύνταξη των τριάδων σύμφωνα με την **Turtle**, μια πιο συνοπτική γλώσσα περιγραφής τριάδων RDF από τη μορφή NTriples
 - ▶ **Συντομογραφίες τριάδων**: όταν έχουν **κοινό υποκείμενο** μπορείτε να γράψετε

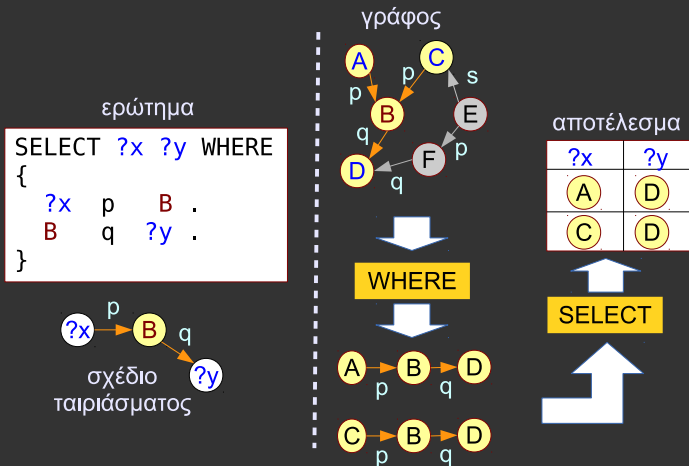
```
?x foaf:name ?name ;      αντί για      ?x foaf:name ?name .  
  foaf:mbox ?mbox .        ?x foaf:mbox ?mbox .
```
 - ▶ **και όταν έχουν κοινό υποκείμενο και κατηγορήμα** μπορείτε να γράψετε

```
?x foaf:nick "Alice" , "Alice_" .  αντί για      ?x foaf:nick "Alice" .  
                                         ?x foaf:nick "Alice_" .
```
 - ▶ **Σταθερές τιμές (Literals)**: μπορείτε να γράψετε τις αριθμητικές τιμές χωρίς εισαγωγικά και χωρίς τύπο δεδομένων
 - ▶ π.χ. το σκέτο 57 είναι ισοδύναμο με το `"57"^^xsd:integer`

Μορφές ερωτημάτων (Query Forms)

- ▶ Η SPARQL (ήδη από την έκδοση 1.0) παρέχει 4 τύπους ερωτημάτων:
 - ▶ **SELECT**: όπως είδαμε στο βασικό παράδειγμα, ταιριάζει σχέδια (patterns) τριάδων και επιστρέφει τις τιμές που ανατέθηκαν στις μεταβλητές του ερωτήματος
 - ▶ **CONSTRUCT**: ταιριάζει σχέδια (patterns) τριάδων και χρησιμοποιεί τις τιμές που ανατέθηκαν στις μεταβλητές για να κατασκευάσει έναν νέο γράφο
 - ▶ **ASK**: επιστρέφει TRUE ή FALSE αν υπάρχουν ταιριάσματα για τη συγκεκριμένη ερώτηση ή όχι
 - ▶ **DESCRIBE**: επιστρέφει έναν νέο γράφο με “ενδιαφέρουσα πληροφορία” για τις οντότητες που ταιριάζει η ερώτηση
 - ▶ Το πρότυπο της SPARQL δεν περιγράφει τι πρέπει να επιστρέφεται, συνήθως οι τριάδες που περιέχουν τις οντότητες ως υποκείμενα ή αντικείμενα

SELECT και WHERE



- ▶ **WHERE**: επιλέγει τριάδες που ταιριάζουν
- ▶ **SELECT**: επιλέγει δεδομένα από τις τριάδες που ταίριαξαν
 - ▶ Προσοχή: η έξοδος του SELECT **δεν είναι τριάδες** αλλά **πίνακας λύσεων**!

Τροποποιώντας τα αποτελέσματα

- ▶ Όταν η SELECT έχει έτοιμα τα αποτελέσματα, μπορείτε να αλλάξετε τη σειρά τους ή το πλήθος τους:
 - ▶ **DISTINCT**: απαλοιφή πανομοιότυπων λύσεων
 - ▶ `SELECT DISTINCT ... WHERE { ... }`
 - ▶ Η απαλοιφή γίνεται μετά την ταξινόμηση αλλά πριν την εφαρμογή LIMIT και OFFSET
 - ▶ Το γράφετε αμέσως μετά το SELECT
 - ▶ **ORDER BY**: ταξινόμηση αποτελεσμάτων με βάση κάποια κριτήρια
 - ▶ `SELECT ... WHERE { ... } ORDER BY ?name DESC(?age)`
 - ▶ Το γράφετε πριν τα LIMIT και OFFSET, μετά το WHERE
 - ▶ **LIMIT** και **OFFSET**: ρυθμίζουν το πλήθος και το σημείο έναρξης λήψης των αποτελεσμάτων
 - ▶ `SELECT ... WHERE { ... } LIMIT 10 OFFSET 20`
 - ▶ Προφανώς, για να τα χρησιμοποιήσετε για σελιδοποίηση (pagination), θα πρέπει πρώτα να τα ταξινομήσετε!

Δοκιμάζοντας τη SPARQL off-line

- ▶ Για να θέσετε δοκιμαστικά ερωτήματα SPARQL σε δεδομένα ενός εγγράφου RDF μπορείτε να χρησιμοποιήσετε στο εργαστήριο:
 - ▶ Το command-line εργαλείο `arq`, το οποίο περιλαμβάνεται στο **Apache Jena Framework**
 - ▶ `arq --query test.rq --data test.nt`
 - ▶ Το command-line εργαλείο `roqet`, από τη βιβλιοθήκη **Redland RDF**
 - ▶ `roqet -D test.nt test.rq`

Η σειρά σας!

- ▶ Χρησιμοποιήστε το σετ δεδομένων RDF για το ωρολόγιο πρόγραμμα με το `arg` για να θέσετε τα παρακάτω ερωτήματα (εκφράστε τα σε SPARQL):
 1. Ποιες οι διαλέξεις την Τρίτη;
 2. Ποιες οι διαλέξεις την Τρίτη και σε ποιες ώρες;
 3. Όπως το προηγούμενο αλλά ταξινομήστε με την ώρα έναρξης!
 4. Ημέρες που γίνονται μαθήματα –θυμηθήκατε το `DISTINCT`;
 5. Ποια τα κατηγορήματα που χρησιμοποιεί το σετ δεδομένων;
 6. Ποιοι διδάσκοντες ξεκινάνε στις 10 το πρωί και ποιες μέρες;
 7. Τα δεδομένα κάθε διάλεξης (μάθημα, ημέρα, έναρξη, λήξη) με ταξινόμηση κατά μάθημα!
 8. Ημέρες και ώρες των διαλέξεων ενός συγκεκριμένου μαθήματος;
 9. Μοναδικά ζεύγη (διδάσκων, μάθημα);
 10. Διδάσκοντες ενός συγκεκριμένου μαθήματος;

Η χρήση του FILTER

- ▶ Το FILTER συνοδεύει προαιρετικά ένα σχέδιο ταιριάσματος τριάδων και **περιορίζει** το σύνολο των προσφερόμενων λύσεων:

```
SELECT DISTINCT ?course WHERE {  
  ?s exv:startTime ?start .  
  ?course exv:hasLecture ?s .  
  FILTER (?start < "15:00:00"^^xsd:time)  
}
```

- ▶ Ακολουθείται από μια λογική έκφραση (true/false) που καθορίζει αν η συγκεκριμένη λύση θα περιληφθεί στην απάντηση
 - ▶ Η έκφραση μπορεί να συγκρίνει αριθμητικές τιμές (τελεστές =, !=, <, <=, > και >=)
 - ▶ Ή γενικά όρους για ισότητα και ανισότητα
 - ▶ Να καλεί συναρτήσεις της SPARQL (θα δούμε μερικές αργότερα)
 - ▶ Και να συνδυάζει όλα τα προηγούμενα με λογικούς τελεστές (&&, || και !)

Αναζητώντας αυτό που δεν υπάρχει

- ▶ Η ορθότερα: αναζητώντας λύσεις που δεν ταιριάζουν στην ερώτηση
- ▶ Η SPARQL διαθέτει 3 (!) τρόπους για να το κάνετε, εδώ φαίνεται ο απλούστερος
 - ▶ Αλλά μόνο στη SPARQL 1.1!

```
SELECT ?s WHERE {  
  ?s a exn:Lecture .  
  FILTER NOT EXISTS {  
    ?s exn:hasTeacher ?t .  
  }  
}
```

- ▶ Λαμβάνοντας υπόψη τις τιμές των μεταβλητών μετά το κάθε ταίριασμα, το FILTER NOT EXISTS ελέγχει ένα δεύτερο σχέδιο ταιριάσματος
 - ▶ Αν το τελευταίο δεν έχει λύση, μόνο τότε το πρώτο ταίριασμα γίνεται αποδεκτό

Η σειρά σας!

- ▶ Χρησιμοποιήστε το σετ δεδομένων RDF για το ωρολόγιο πρόγραμμα με το `arg` για να θέσετε τα παρακάτω ερωτήματα (εκφράστε τα σε SPARQL):
 - ▶ Μαθήματα που έχουν (και) διαλέξεις πριν το μεσημέρι
 - ▶ Διδάσκοντες που διδάσκουν (και) πριν το μεσημέρι
 - ▶ Ποιοι διδάσκουν στις 13:00, ποια μέρα και ποιο μάθημα;
 - ▶ Ημέρες που υπάρχουν διαλέξεις αλλά όχι στις 10:00

rdflib και ερωτήσεις SELECT της SPARQL

```
import rdflib

# create an RDF graph object
g = rdflib.Graph()

# load a dataset from a file into graph
g.parse("schedule.nt",format='nt')

# the SPARQL query string
qstr = """
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ex: <http://ex.com/resource/>
PREFIX exv: <http://ex.com/myvocab#>

select ?course ?start ?end where {
  ?s exv:imera "triti" .
  ?s exv:mathima ?course .
  ?s exv:enarksii ?start .
  ?s exv:liksi ?end .
}
"""

# query the graph and get results
results = g.query(qstr)

# iterate over result rows
for row in results:
    ^^Iprint("{} {} {}".format(row['course'],row['start'],row['end']))
    ^^I# can also access results by index number
    ^^Iprint("{} {} {}".format(row[0],row[1],row[2]))
```


Η σειρά σας!

- ▶ Κατασκευάστε με τη βοήθεια της `rdflib` απλό πρόγραμμα για υποβολή ερωτήσεων SPARQL στα δεδομένα RDF του ωρολογίου προγράμματος
 - ▶ Λειτουργία Α': ο χρήστης δίνει ημέρα της εβδομάδας και το πρόγραμμα επιστρέφει τα μαθήματα (διαλέξεις) για τη μέρα αυτή
 - ▶ Λειτουργία Β': ο χρήστης δίνει ημέρα και ώρα και το πρόγραμμα επιστρέφει τα μαθήματα (διαλέξεις) για την ημέρα και ώρα αυτή (αν υπάρχουν)
 - ▶ Προσοχή: εδώ πρέπει να χειριστείτε διαστήματα ωρών!
- ▶ Μπορείτε να δοκιμάσετε τις ερωτήσεις σας πρώτα με τη βοήθεια του `arg`