SPARQL

- SPARQL Protocol and RDF Query Language
  - Είναι σαν την SQL
  - Κάνει ερωτήματα πάνω σε RDF δεδομένα
  - Εξάγει συμπεράσματα βάσει των τριπλετών του RDF

Δήλωση	Επεξήγηση
PREFIX	Καθορισμός ονόματος για ένα URI
SELECT	Επιστρέφει όλες ή κάποιες από τις μεταβλητές της WHERE
FILTER	Χρησιμοποιείται για υπόδειξη ενός λογικού περιορισμού
WHERE	Μια συνδυαστική λίστα με patterns σε τριάδες ή γράφο
OPTIONAL	Μια συνδυαστική λίστα με προαιρετικά patterns σε λίστα ή γράφο
AND	Λογική έκφραση που μπορεί να εφαρμοστεί στο αποτέλεσμα

SPARQL

παράδειγμα:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/22-rdf-schema#>
SELECT ?c
WHERE
{
?c rdf:type rdfs:Class .
}
```

(τα υποδείγματα τριάδων, όπου rdf:type είναι η ιδιότητα και rdfs:Class το αντικείμενο →ανάκτηση όλων των κλάσεων)

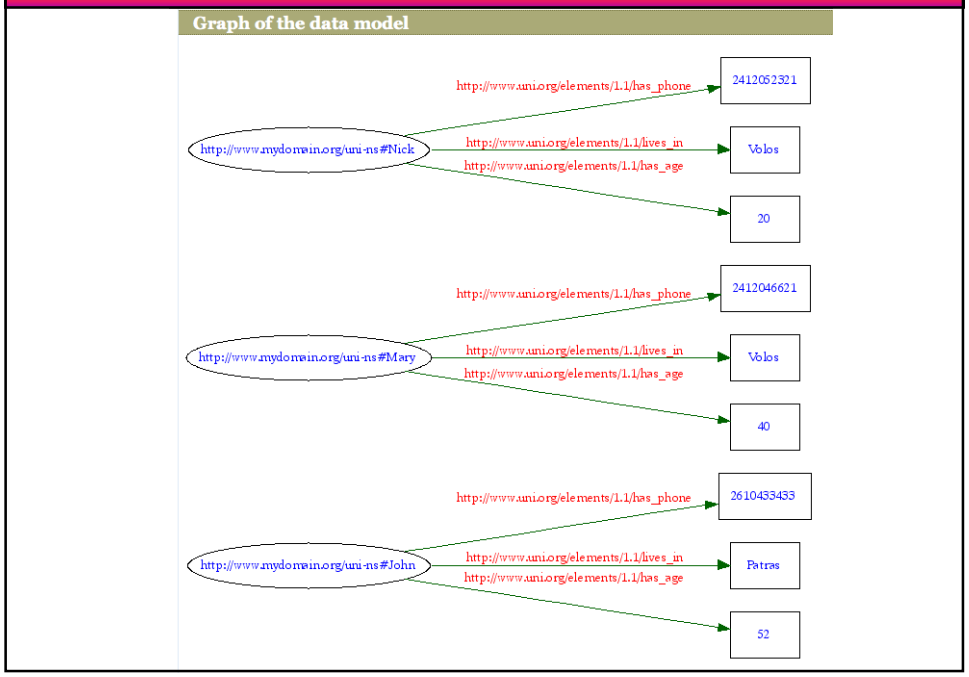
→ *το format εξαγωγής των αποτελεσμάτων μπορεί να οριστεί κατά το δοκούν και ανάλογα πάντα με την εφαρμογή (e.g. HTML, CSV, RDF/XML, TURTLE, JSON, N3)*

Έστω ότι θέλουμε να κάνουμε SPARQL Query σε μια οντολογία που περιέχει και τα παρακάτω instances

```
<rdf:Description rdf:about="http://www.mydomain.org/uni-ns#John">
    <uni:has_phone> 2610433433 </uni:has_phone>
    <uni:lives_in> Patras </uni:lives_in>
    <uni:has_age>52</uni:has_age>
</rdf:Description>

<rdf:Description rdf:about="http://www.mydomain.org/uni-ns#Mary">
    <uni:has_phone> 2412046621</uni:has_phone>
    <uni:lives_in> Volos </uni:lives_in>
    <uni:has_age>40</uni:has_age>
</rdf:Description>

<rdf:Description rdf:about="http://www.mydomain.org/uni-ns#Nick">
    <uni:has_phone> 2412052321</uni:has_phone>
    <uni:lives_in> Volos </uni:lives_in>
    <uni:has_age>20</uni:has_age>
</rdf:Description>
```



Ερώτημα για να πάρουμε τα ονόματα και τα τηλέφωνα όσων μένουν στον Βόλο:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX uni: <http://www.uni.org/...#>
SELECT ?name, ?phone
WHERE {
?name uni:lives_in "Volos" .
?name uni:has_phone ?phone .
}
```

Θα επιστραφεί:

name	phone
http://www.mydomain.org/uni-ns#Mary	2412046621
http://www.mydomain.org/uni-ns#Nick	2412052321

SPARQL

Ερώτημα για να πάρουμε τα ονόματα και τα τηλέφωνα όσων μένουν στον Βόλο **και** είναι κάτω από 30 ετών:

```
...
SELECT ?name, ?phone
WHERE {?name uni:lives_in "Volos" .
       ?name uni:has_phone ?phone .
       ?name uni:has_age ?age .
       FILTER (?age < 30) .
}
```

Θα επιστραφεί:

name	phone
http://www.mydomain.org/uni-ns#Nick	2412052321

SPARQL

Άλλο παράδειγμα:
- SELECT: καθορίζει τον αριθμό και τη σειρά των προς ανάκτηση δεδομένων
- WHERE: επιβάλλει περιορισμούς στις δυνατές απαντήσεις

```
...
SELECT ?x ?y
WHERE
{
?x uni:has_phone ?y .
}
```

(ανάκτηση όλων των τηλεφώνων των μελών του προσωπικού)

SPARQL (implicit join)

```
.....
SELECT ?x ?y
WHERE
{
?x rdf:type uni:lecturer ;
uni:phone ?y .
}
```

```
.....
SELECT ?x ?y
WHERE
{
?x rdf:type uni:lecturer .
?x uni:phone ?y .
}
```

(ανάκτηση όλων των διδασκόντων και των τηλεφώνων τους)

1. Ο όρος ?x rdf:type uni:Lecturer συγκεντρώνει όλα τα στιγμιότυπα της κλάσης Lecturer και συνδέει το αποτέλεσμα με την μεταβλητή ?x
2. Το uni:phone ?y συγκεντρώνει όλες τις τριάδες με κατηγόρημα phone
3. Η έμμεση συνένωση (implicit join) (λόγω του «;») περιορίζει τις τριάδες αυτές σε κείνες με κοινό υποκείμενο με τις πρώτες (?x)

SPARQL (explicit join)

```
…..
SELECT ?n
WHERE
{
?x rdf:type uni:course;
uni:isTaughtBy "janag" .
?c uni:has_title ?n .
FILTER (?c = ?x)  .
}
```

(ανάκτηση όλων των ονομάτων των μαθημάτων που διδάσκονται από τον "janag").

1. Ο όρος ?x rdf:type uni:Course συγκεντρώνει όλα τα στιγμιότυπα της κλάσης Course και συνδέει το αποτέλεσμα με την μεταβλητή ?x
2. Το FILTER χρησιμοποιείται για υπόδειξη ενός λογικού περιορισμού. Εδώ ο περιορισμός είναι η άμεση συνένωση (explicit join) των μεταβλητών ?c και ?x (χρήση τελεστή ισότητας «=»).

SPARQL (εξωτερικοί τελεστές)

```
…..
SELECT ?department (COUNT(?x) AS ?count) (AVG(?age) AS ?age)
WHERE
{
?department rdf:type uni:Department ;
            uni:hasProfessor ?x .
?x uni:age ?age .
}
```

(ανάκτηση όλων των τμημάτων του πανεπιστημίου μαζί με τον αριθμό και το μέσο όρο ηλικίας των καθηγητών για το καθένα από τα τμήματα).

- Άλλοι τελεστές: SUM, MIN, MAX etc.

SPARQL

- Μέχρι τώρα επιστρέφεται απάντηση αν υπάρχει πλήρης ταύτιση υποδείγματος στη βάση γνώσης
- Συχνά όμως απαιτείται μεγαλύτερη ευελιξία

```
<uni:lecturer rdf:id="#janag">
    <uni:name>I. Anagnostopoulos</uni:name>
</uni:lecturer>

<uni:lecturer rdf:id="#vpp">
    <uni:name>V. Plagianakos</uni:name>
    <uni:email>vpp@ucg.gr</uni:email>
</uni:lecturer>
```

```
...
SELECT ?name ?email
WHERE
{
?x rdf:type uni:lecturer ;
    uni:name ?name ;
    uni:email ?email .
}
```

Τι επιστρέφει;

SPARQL (OPTIONAL)

- Μέχρι τώρα επιστρέφεται απάντηση αν υπάρχει πλήρης ταύτιση υποδείγματος στη βάση γνώσης
- Συχνά όμως απαιτείται μεγαλύτερη ευελιξία

```
<uni:lecturer rdf:about="#janag">
    <uni:name>I. Anagnostopoulos</uni:name>
</uni:lecturer>

<uni:lecturer rdf:about="#vpp">
    <uni:name>V. Plagianakos</uni:name>
    <uni:email>vpp@ucg.gr</uni:email>
</uni:lecturer>
```

```
...
SELECT ?name ?email
WHERE
{
?x rdf:type uni:lecturer ;
    uni:name ?name ;
OPTIONAL {?x uni:email ?email}
}
```

SPARQL (UNION)

- Είναι πιθανό να θελήσουμε να δεσμεύσουμε διαφορετικά resources στην ίδια μεταβλητή
- Για παράδειγμα, θεωρήστε την περίπτωση που ρωτάμε δύο γράφους που χρησιμοποιούν διαφορετικά properties για το όνομα ενός ανθρώπου
- πχ. uni:name, foaf:name

```
<uni:lecturer rdf:about="#janag">
    <uni:name>I. Anagnostopoulos</uni:name>
</uni:lecturer>

<uni:lecturer rdf:about="#vpp">
    <foaf:name>V. Plagianakos</ foaf:name>
</uni:lecturer>
```

```
...
SELECT ?name
WHERE
{
{?x uni:name ?name}
UNION
{?x foaf:name ?name}
}
```

SPARQL (τελεστές αποτελεσμάτων)

- Όπως και στην SQL, συχνά θέλουμε να ταξινομήσουμε τα αποτελέσματα βάσει κάποιων κριτηρίων
- ORDER BY
- GROUP BY
- HAVING
- LIMIT

SPARQL (ORDER BY)

- Κάνει κατάταξη των αποτελεσμάτων με αύξουσα (ASC) ή φθίνουσα (DESC) σειρά

```
...
SELECT ?name
WHERE
{
?x foaf:name ?name
}
ORDER BY ASC(?name)
```

```
...
SELECT ?age
 WHERE
{
?x uni:age ?age
}
ORDER BY DESC(?age)
```

SPARQL (GROUP BY)

- Ομαδοποιεί τα αποτελέσματα βάσει μίας μεταβλητής για να επιτρέψει την εφαρμογή υπολογισμών σε κάθε μία από τις ομάδες αυτές

```
...
SELECT ?position (AVG(?age) AS ?averageAge)

WHERE
{
?x uni:position ?position ;
    uni:age ?age .
}
GROUP BY ?position
```

SPARQL (HAVING)

- Επιτρέπει την εισαγωγή υπολογισμών στις τελικές συνθήκες κατάταξης

```
...
SELECT ?position
WHERE
{
?x uni:position ?position ;
    uni:age ?age .
}
GROUP BY ?position
HAVING (AVG(?age) > 30)
```

- Η συνθήκη του HAVING δε μπορεί να μπει εντός του ερωτήματος με FILTER διότι πρέπει πρώτα να ανακτηθούν όλες οι ηλικίες για να υπολογιστεί ο μέσος όρος

SPARQL (LIMIT)

- Περιορίζει τον αριθμό των επιστρεφόμενων αποτελεσμάτων

```
...
SELECT ?position
WHERE
{
?x uni:position ?position ;
    uni:age ?age .
}
ORDER BY ASC(?age)
LIMIT 10
```

References

http://www.w3.org/TR/rdf-sparql-query/
http://www.w3.org/TR/sparql11-protocol/

Apache ARQ - A SPARQL Processor for Jena,
http://incubator.apache.org/jena/documentation/query/
http://incubator.apache.org/jena/

Other:
Sesame (http://openrdf.org/),
JRDF (http://jrdf.sourceforge.net/),
CubicWeb (www.cubicweb.org)

ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ, Ι. ΧΑΤΖΗΛΥΓΕΡΟΥΔΗΣ
http://aigroup.ceid.upatras.gr/undergrad/krweb/docs/RDF_Schema.pdf
http://ebookbrowse.com/fr-3-rdf-sparql-ppt-d209157358