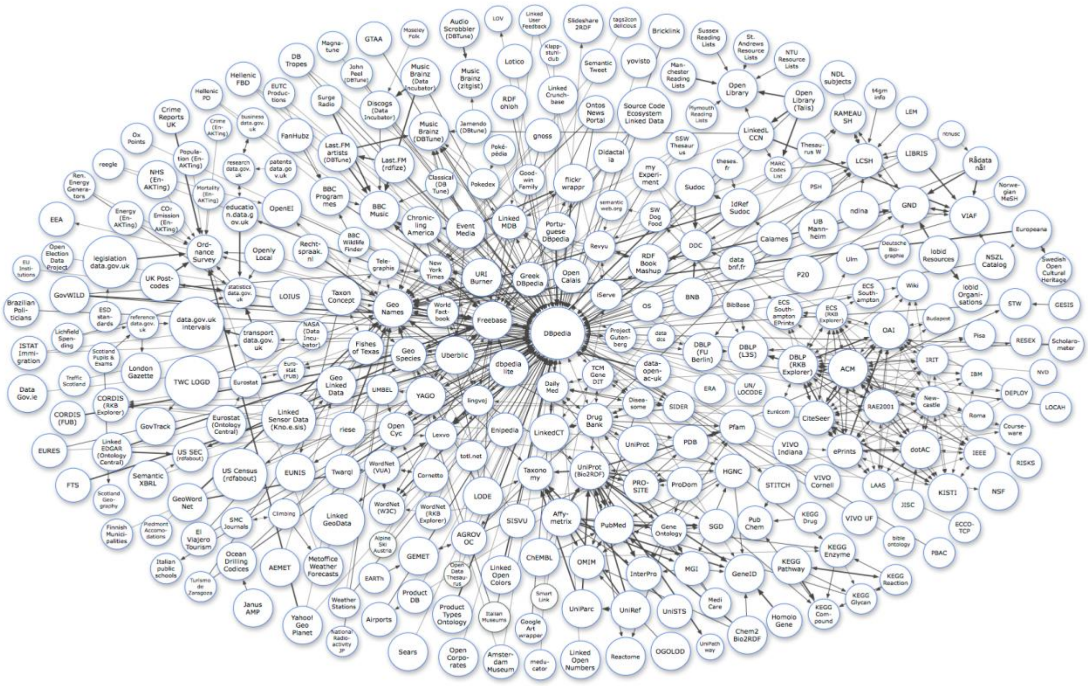
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΥΝΔΕΔΕΜΕΝΑ ΔΕΔΟΜΕΝΑ ΚΑΙ Η ΠΛΑΤΦΟΡΜΑ D2RQ



Του φοιτητή

Κυρατζόπουλου Νικόλαου

Αρ. Μητρώου: 08/3324

Επιβλέπων καθηγητής

Δημήτριος Α. Δέρβος

Θεσσαλονίκη 2014

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στα πλαίσια του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής, της Σχολής Τεχνολογικών Εφαρμογών, του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης, υπό την επίβλεψη του κ. Δημητρίου Α. Δέρβου, καθηγητή του τμήματος.

Ιδιαίτερες ευχαριστίες απευθύνονται στον επιβλέποντα καθηγητή για την καθοδήγησή του, για την παροχή βιβλιογραφικών πηγών καθώς και για τη βοήθεια που μου παρείχε όπου αυτή του ζητήθηκε.

ΕΠΙΤΟΜΗ

Στο Διαδίκτυο υπάρχει ένας τεράστιος όγκος δεδομένων και πληροφοριών, όπου η διαχείρισή τους και η αναζήτηση συγκεκριμένων πληροφοριών γίνεται όλο και πιο δύσκολη υπόθεση, καθώς καθημερινά αυτός ο όγκος των δεδομένων και των πληροφοριών αυξάνεται με ταχύ ρυθμό. Σημαντική επιβάρυνση στη διαδικασία αυτή προκύπτει από το γεγονός ότι τα δεδομένα και οι πληροφορίες αυτές, είναι αποθηκευμένες σε διαφορετικές πηγές, καθώς υπάρχει και το ενδεχόμενο να έχουν και διαφορετική μορφοποίηση. Φυσικά, η κατανόηση του είδους των δεδομένων και των πληροφοριών είναι δύσκολη έως αδύνατη, καθώς χρειάζεται σε μεγάλο βαθμό η επεξεργασία των πληροφοριών από τους ανθρώπους. Έτσι χρειαζόταν η ανάπτυξη νέων τεχνολογιών και προτύπων που διευκολύνουν τη διαχείριση, την πρόσβαση και την κατανομή των δεδομένων και πληροφοριών που υπάρχουν στο Διαδίκτυο. Γι' αυτό το λόγο δημιουργήθηκε το Semantic Web, το οποίο έχει στόχο να προσδώσει σημασιολογία στα δεδομένα και στις πληροφορίες που υπάρχουν, έτσι ώστε οι τελευταίες να γίνονται κατανοητές από τους υπολογιστές οι οποίοι θα τις επεξεργάζονται πιο εύκολα. Μία από τις τεχνολογίες που χρησιμοποιεί το Semantic Web είναι τα Linked Data, τα οποία είναι διασυνδεδεμένα δεδομένα, για την ευκολότερη εύρεση συσχετισμένων δεδομένων. Αυτό επιτυγχάνεται με την χρήση της τεχνολογίας RDF, η οποία είναι ένα μοντέλο δεδομένων που χρησιμοποιείται για την αναπαράσταση των πληροφοριών του Διαδικτύου, καθώς και των ιδιοτήτων τους και των σχέσεων που υπάρχουν μεταξύ τους. Ακόμα μία βοήθεια σε αυτόν τον στόχο είναι και τα διάφορα λεξιλόγια και οντολογίες που υπάρχουν όπως το RDFS, το FOAF κτλ, τα οποία χρησιμοποιούνται για την αναπαράσταση των ιδιοτήτων, τύπων και οντολογιών. Επίσης, υπάρχουν διάφορες μορφές σειριοποίησης των δεδομένων αυτών, όπως η N-Triples και η N3 κτλ, για να γίνονται εύκολα κατανοητά και από τους ανθρώπους. Επιπλέον, υπάρχει και η γλώσσα ερωτημάτων SPARQL. Με την SPARQL μπορούμε να εκτελούμε ερωτήματα πάνω στα δεδομένα που έχουμε σε μορφή RDF και έτσι μπορούμε να αναζητούμε πιο συγκεκριμένη πληροφορία, τη στιγμή που, μπορούμε να φιλτράρουμε και να αναζητούμε ακριβώς την πληροφορία που θέλουμε. Με όλες αυτές τις τεχνολογίες και τα πρότυπα, υπάρχει η δυνατότητα οι εφαρμογές να ανταλλάζουν πληροφορίες μεταξύ τους, καθώς και η δυνατότητα να προτείνονται αντίστοιχες πληροφορίες που ίσως να ενδιαφέρουν τον χρήστη.

Τέλος, υπάρχουν και ειδικά εργαλεία που έχουν δημιουργηθεί για να μας βοηθούν στη δημοσίευση των δεδομένων μας στο Διαδίκτυο σε μορφή RDF, όπως η D2RQ πλατφόρμα και το OpenLink Virtuoso. Αυτά, είτε είναι «open source» άρα δωρεάν, είτε επί πληρωμή. Με τη βοήθεια αυτών των εργαλείων μπορούμε να καταστήσουμε τα δεδομένα μας διαθέσιμα προς όλους.

ABSTRACT

On the Internet there is a huge amount of data and information, the their management and the search of which becomes more and more difficult every day as the volumes of data and information is growing rapidly. Significant burden in this process is the fact that the data and the information are stored in different sources, and often in different formats. In this respect, understanding and interpreting raw data comprises a difficult and often impossible task, calling for high processing overhead, when traditional data processing technologies are used. Considering the above, new technologies and standards need be developed that facilitate the management, access and sharing of data and information on the Internet. For this reason the Semantic Web(SW) has been created. SW, aims to give semantics to data and information in a way that they can be understood and be processed efficiently by computers. One of the technologies used by the Semantic Web is Linked Data. The latter are interconnected data, facilitating the finding of correlated data. This is achieved with the use of RDF technology, which is a data model used for representing information on the Internet, and their properties and relationships between them. Another goal is the establishment of vocabularies and ontologies like RDFS, FOAF etc, which are used for the representation of data properties, types and ontologies. Also, there are various forms of data serialization, such as the N-Triples, N3 etc. Furthermore, there is the SPARQL query language. With SPARQL we can execute queries on data stored in RDF format in order to conduct specific searching tasks. With these technologies and standards, applications can exchange information between them making possible the extraction of information that is of interest to user.

Finally, there are tools that have been developed to help us publish our data on the Internet in the RDF form, such as D2RQ platform and the OpneLInk Virtuoso. Both are available either as “open source”, or as commercial products. Using such tools we can publish our data on the Web, making it freely available to all interested parties: users and/or Internet applications.

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Όταν δημιουργήθηκε το Διαδίκτυο ονομαζόταν ARPANET και ήταν ένα δίκτυο που αναπτύχθηκε από το υπουργείο άμυνας των ΗΠΑ και συνέδεε στρατιωτικές μονάδες με πανεπιστήμια που εκτελούσαν ερευνητικά προγράμματα. Από τότε άρχισε να εξαπλώνεται και να παίρνει την σημερινή μορφή του. Με την ανάπτυξη αυτή του Διαδικτύου, όλο και περισσότερος κόσμος είχε πρόσβαση στα δεδομένα και τις πληροφορίες που υπήρχαν. Τα τελευταία χρόνια, το Διαδίκτυο έχει αναπτυχθεί σε τεράστιο βαθμό και συνεχίζει να αναπτύσσεται συνέχεια με γρήγορους ρυθμούς. Ο όγκος των δεδομένων και των πληροφοριών που υπάρχουν στο Διαδίκτυο, είναι τεράστιος και η διαχείρισή τους είναι πολύ δύσκολη.

Λόγω της χρησιμοποίησης του Διαδικτύου από διάφορους και πολλούς ανθρώπους, το Διαδίκτυο χρειάστηκε να αλλάξει μορφή ανάλογα με τις ανάγκες των χρηστών. Στην αρχική του μορφή, οι ιστοσελίδες ήταν απλά υπερκείμενα που μπορούσες να πλοηγηθείς μεταξύ τους μόνο με συνδέσμους. Όμως με την χρησιμοποίησή του από όλο και περισσότερους ανθρώπους, άρχισε να προστίθεται περισσότερη λειτουργικότητα στις ιστοσελίδες, ώστε οι χρήστες να μπορούν να αλληλεπιδρούν πιο πολύ με τις ιστοσελίδες. Έτσι άρχισαν να αναπτύσσονται εφαρμογές διαδικτύου, κάνοντας το Διαδίκτυο ακόμα πιο δημοφιλές και πιο ελκυστικό για τους χρήστες. Πλέον, επειδή υπάρχει ένας τεράστιος όγκος δεδομένων στο Διαδίκτυο που είναι πολύ δύσκολος στη διαχείρισή του, γίνεται μία προσπάθεια να δοθεί μία σημασιολογία στα δεδομένα και τις πληροφορίες που υπάρχουν στο Διαδίκτυο. Έτσι ώστε, να μπορούν να γίνονται κατανοητά και από τους υπολογιστές και όχι μόνο από τους ανθρώπους. Επίσης, αυτό βοηθάει στην αναζήτηση και στην εύρεση πιο εύκολα των πραγμάτων που ψάχνουμε.

Στην παρούσα πτυχιακή εξετάζεται η αναπαράσταση και η διαχείριση των δεδομένων και των πληροφοριών, με νέες τεχνολογίες που μας βοηθούν στο έργο αυτό. Στα κεφάλαια που ακολουθούν, εξετάζονται και αναπτύσσονται οι τεχνολογίες και τα πρότυπα που χρησιμοποιούνται, καθώς επίσης παρουσιάζονται και κάποια εργαλεία που μας βοηθούν στην χρήση τους. Οι στόχοι της πτυχιακής αυτής είναι η δημοσίευση του περιεχομένου μίας σχεσιακής βάσης δεδομένων σε μορφή Linked Data, καθώς και της δυνατότητας να εκτελούμε SPARQL ερωτήματα στα δεδομένα της βάσης αυτής. Τέλος, για τη δημοσίευση του περιεχομένου της σχεσιακής βάσης δεδομένων χρησιμοποιήθηκαν δύο εργαλεία, ο D2R Server και το OpenLink Virtuoso (open source), επίσης έγινε και μία σύγκριση μεταξύ των δύο εργαλείων στους χρόνους εκτέλεσης των ερωτημάτων SPARQL.

ΚΕΦΑΛΑΙΟ 1

RDF

1.1 ΕΙΣΑΓΩΓΗ

Το RDF (Resource Description Framework) είναι ένα μοντέλο δεδομένων, το οποίο περιγράφει το πως θα πρέπει να αναπαριστώνται οι πληροφορίες σχετικά με τους πόρους που υπάρχουν στο Διαδίκτυο. Η βασική μονάδα πληροφορίας του RDF μοντέλου ονομάζεται τριπλέτα (triple).

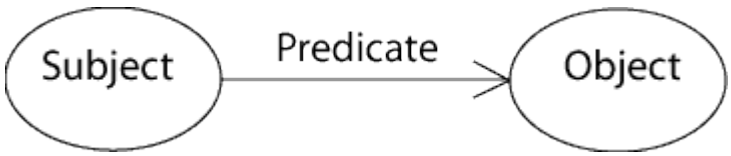
1.2 Τι είναι το RDF μοντέλο

Το W3C (World Wide Web Consortium) δημοσίευσε μια προδιαγραφή του μοντέλου δεδομένων RDF το 1999 μαζί με την XML σύνταξη. Η RDF γλώσσα, στην ουσία το RDF είναι ένα μοντέλο δεδομένων αλλά αρκετές φορές αναφέρεται και σαν γλώσσα, βασίζεται στην XML και σχεδιάστηκε για να αναπαριστά και να περιγράφει τις πληροφορίες και τους πόρους του Διαδικτύου. Δημιουργήθηκε επίσης με στόχο τη διαχείριση των πληροφοριών από διάφορες εφαρμογές, οι οποίες θα μπορούν να ανταλλάσουν τις πληροφορίες μεταξύ τους και όχι μόνο για την αναπαράστασή τους.

Ο τρόπος με τον οποίο αναπαριστούνται οι πληροφορίες στο RDF μοντέλο είναι οι τριπλέτες (triples). Με τις τριπλέτες έχουμε τη δυνατότητα να περιγράψουμε έναν πόρο, δηλαδή τις ιδιότητες που έχει ο πόρος αυτός και τις τιμές που έχουν οι ιδιότητές του. Π.χ. σε μία φυσική γλώσσα όπως η Ελληνική θα λέγαμε το εξής: Η ιστοσελίδα «http://www.example.com/index.html» έχει έναν δημιουργό, ο οποίος είναι ο George Smith. Όμως για να περιγράψουμε τις ιδιότητες ενός πόρου πρέπει πρώτα να ονομάσουμε ή να προσδιορίσουμε κάποιες έννοιες όπως:

1. Τον πόρο που θέλουμε να περιγράψουμε
2. Τις ιδιότητες τις οποίες θα έχει
3. Τις τιμές που θα έχουν οι ιδιότητες

Η ονομασία ή ο προσδιορισμός των εννοιών αυτών στο RDF μοντέλο γίνεται με τη βοήθεια των URIs. Οι τριπλέτες αποτελούνται από τρία στοιχεία, το υποκείμενο (Subject) που είναι αυτό που περιγράφεται, το κατηγόρημα (Predicate) που είναι η ιδιότητα ή το χαρακτηριστικό του υποκειμένου (Subject) και το αντικείμενο (Object) που είναι η τιμή του κατηγορήματος (Predicate).



Σχήμα 1.1: Παράδειγμα τριπλέτας

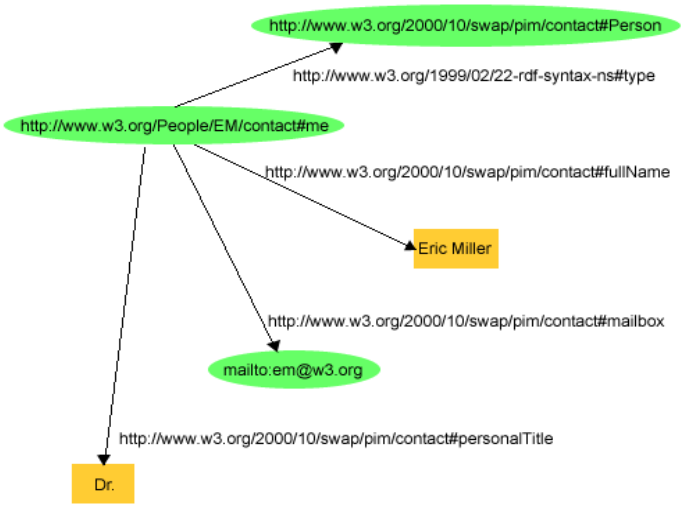
Οπότε, με τους RDF όρους, για την παραπάνω πρόταση του παραδείγματος έχουμε:

- Υποκείμενο το URL http://www.example.com/index.html

- Κατηγόρημα την λέξη «δημιουργός»
- Αντικείμενο το όνομα «George Smith»

Το υποκείμενο μπορεί να είναι είτε ένα URL είτε ένας ανώνυμος κόμβος ή κενός κόμβος. Το κατηγόρημα είναι υποχρεωτικά URI που αντιπροσωπεύει έναν πόρο ο οποίος δίνει ένα γνώρισμα στο υποκείμενο, συνήθως σαν κατηγόρημα χρησιμοποιούνται γνωστά λεξιλόγια που είναι ευρέως αποδεκτά. Τέλος το αντικείμενο μπορεί να είναι είτε ένα URI είτε μία σταθερή τιμή (Literal) είτε ένας ανώνυμος κόμβος ή κενός κόμβος.

Στους RDF γράφους το υποκείμενο και το αντικείμενο είναι κόμβοι και το κατηγόρημα σαν τόξο που ενώνει τους κόμβους μεταξύ τους και έχει φορά προς το αντικείμενο. Οι κόμβοι που είναι URI αναπαρίστανται σαν ελλείψεις ενώ οι σταθερές τιμές (Literals) σαν ορθογώνια παραλληλόγραμμα. Με αυτόν τον τρόπο δημιουργείται ένας κατευθυνόμενος γράφος που αναπαριστά τις πληροφορίες.



Σχήμα 1.2: Τριπλέτες σε γράφο

Όμως, ο γράφος μπορεί να γίνει πολύπλοκος, επειδή είναι μεγάλος ή επειδή αναπαριστά πολλούς πόρους που συνδυάζονται μεταξύ τους, με αποτέλεσμα να δυσκολευόμαστε να τον κατανοήσουμε και να αποσπάσουμε τις πληροφορίες που θέλουμε.

Έτσι υπάρχουν και άλλοι τρόποι αναπαράστασης των πληροφοριών και των πόρων. Μερικοί τρόποι αναπαράστασης που χρησιμοποιούνται πιο συχνά εκτός από την RDF/XML είναι η N-Triples, Turtle, Notation3 ή N3, JSON, N-Quads. Η πιο απλή μορφή αναπαράστασης είναι η N-Triples.

Επίσης στα RDF αρχεία έχουμε τη δυνατότητα να γράφουμε συντομότερα τα URIs χρησιμοποιώντας κάποιο πρόθεμα (prefix) που έχει οριστεί για ένα URI namespace, έτσι ώστε να μην χρειάζεται να γράφουμε όλο το URI πολλές φορές. Επίσης, οργανώνονται καλύτερα και τα URIs καθώς βλέποντας ένα πρόθεμα, στο οποίο έχουμε δώσει ένα κατάλληλο όνομα, μπορούμε να καταλάβουμε αμέσως που αναφέρεται. Τα προθέματα ορίζονται με τον εξής τρόπο:

```
@prefix owl: <http://www.w3.org/2002/07/owl#>
```

Έτσι όταν έχουμε ένα URI για το οποίο έχουμε ορίσει ένα πρόθεμα για το namespace του, τότε μπορούμε να το αντικαταστήσουμε με το πρόθεμά του και μετά να ακολουθήσει το τοπικό όνομα που προσδιορίζει το συγκεκριμένο URI π.χ. έχουμε το namespace «http://xmlns.com/foaf/0.1/» και του έχουμε αναθέσει το πρόθεμα «foaf». Τότε το URI «http://xmlns.com/foaf/0.1/name» μπορούμε να το γράψουμε πιο σύντομα «foaf:name». Όμως, αυτός ο τρόπος της σύντομης γραφής των URIs, είναι απλά μια σύμβαση, καθώς το RDF μοντέλο αναγνωρίζει μόνο ολόκληρα URIrefs και δεν αναλύει τη δομή τους, αλλά ούτε και κάνει καμιά υπόθεση αν έχουν μεταξύ τους κάποια σχέση επειδή έχουν ένα κοινό πρόθεμα ή επειδή έχουν διαφορετικά προθέματα ότι ανήκουν σε διαφορετικά λεξιλόγια.

1.5 Ανώνυμοι κόμβοι

Όταν θέλουμε να αναπαραστήσουμε μια σύνθετη πληροφορία χωρίς να χάσει την σημασία της, π.χ. τη διεύθυνση ενός ανθρώπου και θέλουμε να αναπαραστήσουμε ξεχωριστά την οδό, τον ταχυδρομικό κώδικα, την πόλη κτλ, θα μπορούσαμε να δημιουργήσουμε ξεχωριστούς κόμβους για κάθε κομμάτι πληροφορίας και να τα συνδέσουμε με ένα νέο κόμβο ο οποίος συνδέει το αρχικό υποκείμενο και έχει σαν αντικείμενα, τους νέους αυτούς κόμβους. Όμως μπορεί ο νέος κόμβος, ο οποίος συνδέει το αρχικό υποκείμενο, να μην χρησιμοποιηθεί εκτός του γράφου που τα έχει, δηλαδή από κάποιον άλλον γράφο, τότε μπορούμε να δημιουργήσουμε έναν ανώνυμο κόμβο ή κενό κόμβο (blank node), στον οποίο δεν χρειάζεται να δώσουμε ένα URI ώστε να είναι μοναδικός. Επειδή όμως ένας γράφος μπορεί να περιέχει παραπάνω από έναν ανώνυμο κόμβο ή κενό κόμβο, έτσι τους δίνουμε ένα προσδιοριστικό της μορφής «_:name», που υποδεικνύει ότι αντιπροσωπεύει έναν κενό κόμβο. Τα ονόματα δίνονται μόνο για την αναπαράσταση των κενών κόμβων σε τριπλέτες, στην αναπαράσταση σε γράφο δεν έχουν όνομα. Αν θέλουμε να αναφερθούμε στον κενό κόμβο εκτός του γράφου που τον περιέχει,

τότε πρέπει να προσδιοριστεί μέσω ενός URIref που θα δώσουμε στον ανώνυμο κόμβο ή κενό κόμβο[22].

1.4 Literals

Τα Literals είναι σταθερές τιμές, τα οποία χωρίζονται σε δύο κατηγορίες:

- Plain Literals
- Typed Literals

Τα Plain Literals αποτελούνται από μια σταθερή τιμή και προαιρετικά από ένα αναγνωριστικό γλώσσας. Στα Plain Literals δεν μπορούμε να γνωρίζουμε η τιμή τι τύπου είναι, καθώς η επεξεργασία τους γίνεται θεωρώντας τα ως strings π.χ.:

```
<http://example.com/country> <http://example.com/country#name> “Greece”@en.
```

Τα Typed Literals αποτελούνται από μια σταθερή τιμή και ένα URI που προσδιορίζει έναν τύπο δεδομένων. Π.χ.:

```
<http://example.com/George> <http://example.com/age>
“54”^^<http://www.w3.org/2001/XMLSchema#integer>
```

Το RDF δεν έχει κάποιο ενσωματωμένο σύνολο τύπων δεδομένων όπως έχουν οι γλώσσες προγραμματισμού, π.χ. integer, string κτλ. Οπότε, το RDF με τα Typed Literals υποδεικνύει ποιος τύπος δεδομένων θα πρέπει να χρησιμοποιηθεί ώστε να μπορούμε να επεξεργαστούμε το συγκεκριμένο Literal. Οι τύποι δεδομένων που χρησιμοποιούνται στα Typed Literals ορίζονται εξωτερικά από το RDF και προσδιορίζονται βάση με το URI τύπο δεδομένων τους. Ένα πλεονέκτημα της τεχνικής αυτής είναι, ότι δίνει στο RDF τη δυνατότητα να αναπαριστά απευθείας τις πληροφορίες που έρχονται από διαφορετικές πηγές, χωρίς να χρειάζεται να μετατρέπει τους τύπους δεδομένων της πηγής σε τοπικούς RDF τύπους δεδομένων. Φυσικά για να μπορεί να χρησιμοποιηθεί ένας τύπος δεδομένων στο RDF θα πρέπει πρώτα να είναι καλά ορισμένος ανάλογα με τους κανόνες του RDF[22].

1.3 Επιπλέον δυνατότητες του RDF

Το RDF μας προσφέρει κάποιες επιπλέον δυνατότητες, όπως τους ενσωματωμένους τύπους και ιδιότητες για την αναπαράσταση ομάδων πόρων κτλ, καθώς μας προσφέρει και λεξιλόγιο, ώστε να μπορούμε να δώσουμε κάποια δομή στα δεδομένα.

1.3.1 RDF Containers

Αρκετές φορές, υπάρχει η ανάγκη να περιγράψουμε μια ομάδα πραγμάτων, έτσι το RDF μας προσφέρει μερικούς ενσωματωμένους τύπους και ιδιότητες που μπορούν να χρησιμοποιηθούν για να περιγράψουν αυτές τις ομάδες. Το RDF προσφέρει ένα container λεξιλόγιο που αποτελείται από τρείς προκαθορισμένους τύπους, μαζί με κάποιες συσχετισμένες προκαθορισμένες ιδιότητες.

Το container είναι ένας πόρος που περιέχει πράγματα τα οποία καλούνται μέλη (members). Τα μέλη μπορεί να είναι πόροι (ακόμα και ανώνυμοι κόμβοι ή κενοί κόμβοι) ή Literal. Το RDF προσφέρει τους εξής τρείς container τύπους:

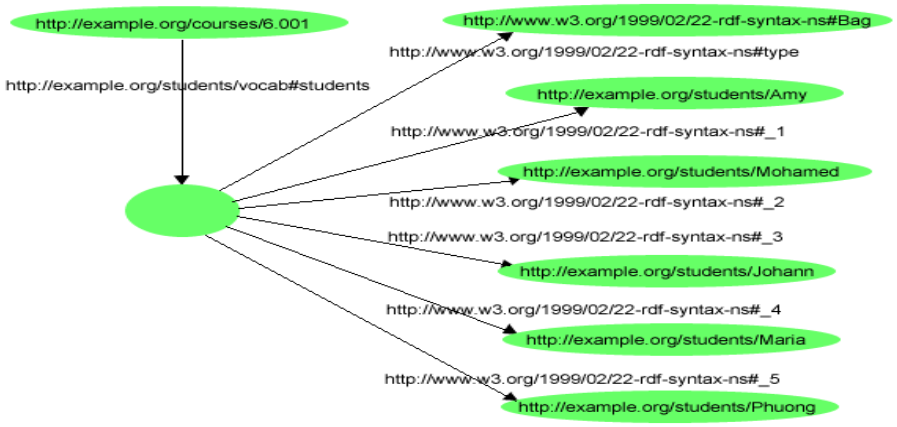
- Rdf:Bag
- Rdf:Seq
- Rdf:Alt

Το Bag αντιπροσωπεύει μια ομάδα από πηγές ή Literals. Πιθανότητα να περιέχει και διπλοεγγραφές από κάποια μέλη, όπου δεν έχει σημασία η σειρά των μελών. Π.χ. όταν θέλουμε να περιγράψουμε μια ομάδα από αριθμούς στους οποίους δεν μας ενδιαφέρει η σειρά εισαγωγής ή επεξεργασίας.

Το Seq ή Sequence αντιπροσωπεύει μια ομάδα από πηγές ή Literals. Πιθανότατα να περιέχει και διπλοεγγραφές από κάποια μέλη, όπου η σειρά των μελών έχει σημασία. Π.χ. όταν θέλουμε να περιγράψουμε μια ομάδα ανθρώπων η οποία θέλουμε να είναι σε αλφαβητική σειρά.

Το alt ή Alternative αντιπροσωπεύει μια ομάδα από πηγές ή Literals, τα οποία είναι εναλλακτικές τιμές για μία έννοια ή ιδιότητα. Π.χ. όταν θέλουμε να περιγράψουμε εναλλακτικές γλώσσες μετάφρασης για τον τίτλο ενός βιβλίου.

Για να περιγραφεί μια πηγή σαν ένας από αυτούς τους τύπους, στην πηγή δίνεται μια «rdf:type» ιδιότητα η οποία έχει σαν τιμή έναν από τους προκαθορισμένους τύπους «rdf:Bag», «rdf:Seq», «rdf:Alt». Η container πηγή, που μπορεί να είναι είτε ένας ανώνυμος κόμβος ή κενός κόμβος είτε μια πηγή με ένα URIfer, δηλώνει την ομάδα σαν σύνολο, όπου τα μέλη της μπορούν να περιγραφούν ορίζοντας μία container membership ιδιότητα για κάθε ένα μέλος, με την container πηγή να είναι το υποκείμενο και τα μέλη της τα αντικείμενα. Οι container membership ιδιότητες έχουν ονόματα της μορφής «rdf:_n», όπου το «n» παίρνει δεκαδικές ακέραιες τιμές μεγαλύτερες του μηδενός. Πρέπει να γίνει κατανοητό ότι αυτοί οι containers τύποι περιγράφονται χρησιμοποιώντας προκαθορισμένους RDF τύπους και ιδιότητες, όπου, οποιαδήποτε ειδική σημασία και αν έχουν πρόκειται απλά για μία κοινή σύμβαση μεταξύ αυτών που θέλουν να περιγράψουν ομάδες πραγμάτων, καθώς το RDF δεν έχει επιπλέον ενσωματωμένη κατανόηση για το τι είδους πηγή είναι ο τύπος π.χ. «rdf:Bag» σε σχέση με μια πηγή με τύπο π.χ. «ex:Tent»[22].

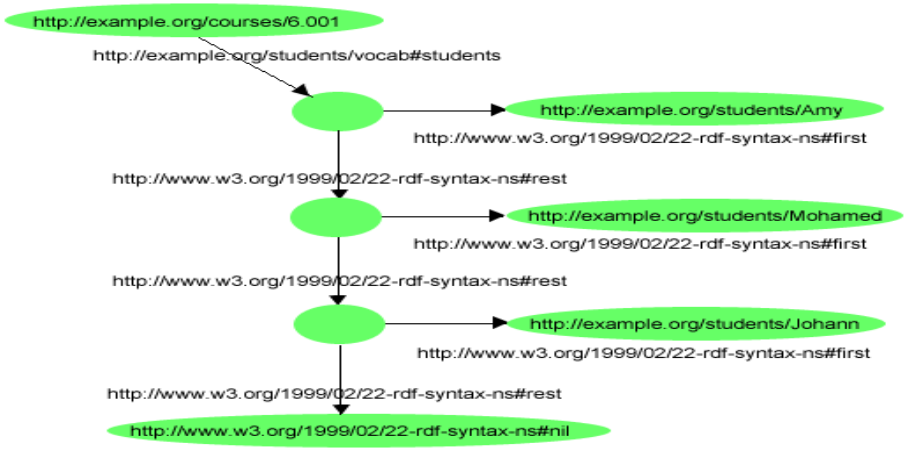


Σχήμα 1.3: Μια απλή Bag Container περιγραφή

1.3.2 RDF Collections

Ένα περιορισμό που έχουν τα RDF Containers είναι ότι δεν υπάρχει τρόπος να ορίσουμε που τελειώνουν. Δηλαδή ένα container απλά αναφέρει κάποιες ορισμένες πηγές σαν μέλη, χωρίς να αναφέρει ότι μπορεί να υπάρχουν και άλλα μέλη, καθώς είναι δυνατόν να υπάρχει και ένας άλλος γράφος που να περιγράφει επιπλέον μέλη. Όμως το RDF προσφέρει και έναν τρόπο για να περιγράψουμε ομάδες που έχουν μόνο συγκεκριμένα μέλη και αυτό το επιτυγχάνει με τα RDF Collections. Τα RDF Collections είναι μια ομάδα πραγμάτων, τα οποία παρουσιάζονται σαν μια λίστα στον RDF γράφο. Η λίστα αυτή δημιουργείται χρησιμοποιώντας ένα προκαθορισμένο collection λεξιλόγιο, που αποτελείται από τον

προκαθορισμένο τύπο «rdf:List», όπου οι προκαθορισμένες ιδιότητες είναι «rdf:first» και «rdf:rest» και τελειώνει με την προκαθορισμένη πηγή «rdf:nil»[22].



Σχήμα 1.4: Μια RDF Collection (δομή λίστας)

1.6 ΕΠΙΛΟΓΟΣ

Με το RDF μοντέλο μπορούμε να αναπαραστήσουμε πληροφορίες οι οποίες είναι κατανεμημένες στο Διαδίκτυο σε διαφορετικές πηγές και να γίνονται κατανοητές από τους υπολογιστές, αφού βασίζεται στην XML. Επίσης για την καλύτερη περιγραφή των δεδομένων έχουν αναπτυχθεί διάφορα λεξιλόγια για το RDF μοντέλο.

ΚΕΦΑΛΑΙΟ 2

ΓΛΩΣΣΕΣ ΠΕΡΙΓΡΑΦΗΣ ΤΥΠΩΝ

2.1 ΕΙΣΑΓΩΓΗ

Με τις γλώσσες περιγραφής τύπων ή λεξιλόγια, μπορούμε να περιγράψουμε τους τύπους των δεδομένων και τις οντότητες, έτσι ώστε να έχουμε μια καλύτερη περιγραφή των πληροφοριών που έχουμε στο RDF μοντέλο. Υπάρχουν αρκετές γλώσσες περιγραφής τύπων, οι οποίες είναι ευρέως αποδεκτές και οι οποίες χρησιμοποιούνται συνήθως όταν θέλουμε να περιγράψουμε συγκεκριμένα πράγματα, καθώς η κάθε μια έχει το δικό της ρόλο.

2.2 RDF Schema

Το RDF Schema προσφέρει ένα λεξιλόγιο μοντελοποίησης δεδομένων για τα RDF δεδομένα. Το RDF Schema είναι μία σημασιολογική (semantic) επέκταση του RDF. Προσφέρει μηχανισμούς για να περιγράφει ομάδες από συσχετισμένους πόρους, καθώς και των σχέσεων που υπάρχουν μεταξύ αυτών των πόρων. Οι περιγραφές του λεξιλογίου RDF Schema γράφονται σε μορφή RDF. Οι πόροι που περιγράφονται με το RDF Schema χρησιμοποιούνται για να προσδιορίσουν χαρακτηριστικά άλλων πόρων, όπως το domain και τα ranges των ιδιοτήτων. Το σύστημα περιγραφής κλάσεων και ιδιοτήτων του RDF Schema, είναι παρόμοιο με τα συστήματα τύπου των αντικειμενοστρεφών γλωσσών προγραμματισμού όπως η java. Το RDF Schema διαφέρει από πολλά τέτοια συστήματα, στο ότι αντί να καθορίζει μια κλάση όσον αφορά τις ιδιότητες που μπορεί να έχει το κάθε ένα στιγμιότυπο (instance), το RDF Schema περιγράφει τις ιδιότητες όσον αναφορά τις κλάσεις των πόρων στους οποίους εφαρμόζονται. Π.χ. θα μπορούσαμε να ορίσουμε μια ιδιότητα «eg:author» να έχει σαν domain το «eg:Document» και range το «eg:Person», ενώ ένα κλασικό αντικειμενοστρεφές σύστημα μπορεί να καθορίσει συνήθως μια κλάση «eg:Book» με ένα χαρακτηριστικό που θα έχει όνομα «eg:author» και θα είναι τύπου «eg:Person». Ένα πλεονέκτημα της προσέγγισης αυτής του RDF είναι ότι επιτρέπει στον καθένα να επεκτείνει την περιγραφή ενός υπάρχοντος πόρου.

Οι πόροι μπορεί να χωριστούν σε ομάδες που ονομάζονται κλάσεις. Τα μέλη μιας κλάσης ονομάζονται στιγμιότυπα της κλάσης. Οι κλάσεις είναι οι ίδιες πόροι, οι οποίες συχνά προσδιορίζονται από URIs και μπορεί να περιγράφονται χρησιμοποιώντας RDF ιδιότητες. Η ιδιότητα «rdf:property» μπορεί να χρησιμοποιηθεί για να δηλώσει ότι ένας πόρος είναι ένα στιγμιότυπο μιας κλάσης. Το RDF διακρίνει μεταξύ τους, την κλάση, με το σύνολο των στιγμιότυπών της. Δύο κλάσεις μπορούν να έχουν το ίδιο σύνολο στιγμιότυπων αλλά να είναι διαφορετικές κλάσεις. Επίσης είναι πιθανό αυτές οι κλάσεις, να έχουν ακριβώς τα ίδια στιγμιότυπα, αλλά να έχουν διαφορετικές ιδιότητες. Η ομάδα των πόρων οι οποίες είναι κλάσεις του RDF Schema είναι από μόνες τους μια κλάση και ονομάζονται «rdfs:Class». Όλοι οι τύποι δεδομένων (datatypes) είναι κλάσεις.

Οι ιδιότητες RDF περιγράφονται σαν μια σχέση ανάμεσα στον πόρο υποκειμένου και τον πόρο αντικειμένου. Αυτός ο προσδιορισμός ορίζει την έννοια της υπο-ιδιότητας (subpoperty). Η «rdfs:subPropertyOf» ιδιότητα μπορεί να δηλώνει ότι μια ιδιότητα είναι μια υπο-ιδιότητα μιας άλλης. Εάν μια ιδιότητα P είναι υπο-ιδιότητα μια ιδιότητας P΄, τότε όλα τα ζευγάρια των πόρων που σχετίζονται με την P σχετίζονται επίσης και με την P΄.

Ο όρος υπερ-ιδιότητα (super-property) χρησιμοποιείται συνήθως σαν αντίστροφο της υπο-ιδιότητας. Εάν μια ιδιότητα P΄ είναι υπερ-ιδιότητα μιας ιδιότητας P, τότε όλα τα ζευγάρια των πόρων τα οποία σχετίζονται με την ιδιότητα P σχετίζονται επίσης και με την ιδιότητα P΄. Αυτός ο προσδιορισμός δεν ορίζει μια αρχική ιδιότητα η οποία είναι υπερ-ιδιότητα όλων των ιδιοτήτων[23].

2.3 OWL

Η Web Ontology Language (OWL) προορίζεται για να χρησιμοποιείται όταν οι πληροφορίες χρειάζονται να επεξεργαστούν από εφαρμογές, σε αντίθεση με καταστάσεις όταν το περιεχόμενο χρειάζεται μόνο να παρουσιαστεί σε ανθρώπους. Η OWL μπορεί να χρησιμοποιηθεί για να αναπαραστήσει ρητά την έννοια των όρων σε λεξιλόγια και των σχέσεων μεταξύ των όρων αυτών. Αυτή η αναπαράσταση των όρων και των σχέσεων τους καλείται οντολογία. Η OWL διαθέτει περισσότερα μέσα για την έκφραση νοημάτων και εννοιών από το RDF και RDFS. Επίσης η OWL πηγαίνει πιο πέρα από αυτές τις γλώσσες-λεξιλόγια στην αναπαράσταση πληροφοριών που μπορούν να διαβαστούν από τους υπολογιστές και υπάρχουν στο Διαδίκτυο. Ακόμα η OWL αποτελεί μια επέκταση του RDF και του RDFS.

Η OWL παρέχει τρείς υπο-γλώσσες για χρήση από συγκεκριμένες κοινότητες και χρήστες:

1. Την OWL Lite, η οποία υποστηρίζει τους χρήστες που πρωτίστως χρειάζονται μια ιεραρχία ταξινόμησης και απλούς περιορισμούς.
2. Την OWL DL, η οποία υποστηρίζει τους χρήστες που θέλουν μέγιστη εκφραστικότητα διατηρώντας την υπολογιστική πληρότητα.
3. Την OWL Full , η οποία προορίζεται για τους χρήστες που θέλουν την μέγιστη εκφραστικότητα και την συντακτική ελευθερία της RDF.

Οι προγραμματιστές οντολογιών που χρησιμοποιούν την OWL πρώτα αποφασίζουν ποία υπο-γλώσσα ταιριάζει με τις ανάγκες τους. Η επιλογή ανάμεσα στην OWL Lite και την OWL DL εξαρτάται από το βαθμό στον οποίο οι χρήστες χρειάζονται μεγαλύτερη εκφραστικότητα των δομών που προσφέρει η OWL DL. Η επιλογή ανάμεσα στην OWL DL και την OWL Full κυρίως εξαρτάται από το βαθμό στον οποίο οι χρήστες χρειάζονται τις meta-modeling ικανότητες του RDF Schema. Επίσης, η OWL Full μπορεί να θεωρηθεί σαν μια επέκταση του RDF, ενώ η OWL Lite

και η OWL DL μπορούν να θεωρηθούν σαν επεκτάσεις μιας περιορισμένης όψης του RDF.

Το Semantic Web είναι μία προοπτική για το μέλλον του Διαδικτύου κατά το οποίο οι πληροφορίες λαμβάνουν σαφή έννοια και «χτίζεται» χρησιμοποιώντας την ικανότητα της XML να προσδιορίζει προσαρμοσμένα σχήματα σήμανσης και την ευέλικτη προσέγγιση του RDF για την αναπαράσταση των δεδομένων. Το πρώτο επίπεδο πάνω από το RDF που απαιτείται για το Semantic Web είναι μία γλώσσα οντολογιών, η οποία μπορεί να περιγράψει επίσημα την ορολογία που χρησιμοποιείται σε έγγραφα του Διαδικτύου. Για να εκτελούν χρήσιμες εργασίες ανάλυσης σε αυτά τα έγγραφα οι μηχανές, η γλώσσα αυτή πρέπει να είναι πέρα από τη βασική σημασιολογία του RDF Schema. Η OWL μας παρέχει αυτή τη δυνατότητα, καθώς επίσης και τα εργαλεία για να προσδιορίζουμε τις διάφορες οντότητες έτσι ώστε να μπορούν να χρησιμοποιηθούν αποτελεσματικά από τις μηχανές[17].

2.4 FOAF

Το Friend of a Friend (FOAF) project πρόκειται για τη δημιουργία ιστοσελίδων ικανών να γίνουν κατανοητές από υπολογιστές, οι οποίες περιγράφουν ανθρώπους, καθώς και τις σχέσεις μεταξύ τους και των πραγμάτων που δημιουργούν και κάνουν. Το FOAF είναι μία απλή τεχνολογία η οποία καθίστα ευκολότερο το μοίρασμα και τη χρησιμοποίηση πληροφοριών σχετικά με τους ανθρώπους και τις δραστηριότητές τους, καθώς και την μεταφορά πληροφοριών ανάμεσα στις ιστοσελίδες και έχει τη δυνατότητα αυτόματα να τις επεκτείνει, να τις συγχωνεύει και να τις επαναχρησιμοποιεί. Το FOAF project ξεκίνησε το 2000 σαν ένα πειραματικό project συνδεδεμένων πληροφοριών από τους Dan Brockley και Libby Miller[36].

Το FOAF λεξιλόγιο περιγράφει ανθρώπους, τις δραστηριότητές τους και τις σχέσεις που έχουν με άλλους ανθρώπους και πράγματα. Το FOAF λεξιλόγιο είναι ένα περιγραφικό λεξιλόγιο το οποίο εκφράζεται χρησιμοποιώντας το RDF και την OWL. Επειδή το RDF περιγράφει με έναν πιο αφαιρετικό μοντέλο πληροφοριών τους όρους στους οποίους εμείς θέλουμε να είμαστε σε θέση να κάνουμε πιο εξειδικευμένες αναζητήσεις, κάνοντας πιο συγκεκριμένες ερωτήσεις έτσι ώστε να αποκτούμε πληροφορίες για διάφορες ομάδες πραγμάτων, το FOAF μας βοηθάει να το επιτύχουμε αυτό για τους ανθρώπους σε μεγάλο βαθμό.

2.5 Άλλα γνωστά λεξιλόγια

2.5.1 SKOS

Το λεξιλόγιο Simple Knowledge Organization System (SKOS) παρέχει έναν τρόπο για την αναπαράσταση των συστημάτων οργάνωσης γνώσης με τη χρήση του RDF. Δηλαδή το SKOS είναι ένα RDF λεξιλόγιο για την αναπαράσταση συστημάτων ημιεπίσημης οργάνωσης γνώσης, όπως οι ιεραρχίες, οι ταξινομήσεις, τα συστήματα ταξινόμησης και οι λίστες θεματικών τίτλων. Επειδή το SKOS είναι βασισμένο στο RDF, αυτές οι αναπαραστάσεις μπορούν να διαβαστούν από τους υπολογιστές και να ανταλλάσσονται μεταξύ των εφαρμογών και να δημοσιεύονται στο Διαδίκτυο. Ο στόχος του SKOS δεν είναι να αντικαταστήσει την αρχική αντίληψη του πλαισίου χρήσης των λεξιλογίων, αλλά να μπορέσουν να μεταφερθούν σε έναν κοινόχρηστο χώρο, βασισμένο σε ένα απλοποιημένο μοντέλο, που επιτρέπει την ευρύτερη επαναχρησιμοποίησή τους και την καλύτερη διαλειτουργικότητά τους[31].

2.5.2 DC

Το Dublin Core (DC) είναι ένα λεξιλόγιο που περιέχει δεκαπέντε ιδιότητες που χρησιμοποιούνται για την περιγραφή πόρων. Το όνομα «Dublin» οφείλεται στην καταγωγή του από ένα workshop του 1995 στο Δουβλίνο (Dublin) του Οχάιο (Ohio), Το «Core» επειδή τα στοιχεία του είναι γενικά, μπορούν να χρησιμοποιηθούν για την περιγραφή ενός ευρέως φάσματος πόρων. Αυτά τα δεκαπέντε στοιχεία που χρησιμοποιούνται στο DC είναι ένα μέρος ενός μεγαλύτερου συνόλου από λεξιλόγια μεταδεδομένων και τεχνικών προδιαγραφών που διατηρούνται από το Dublin Core Metadata Initiative[7].

2.5.3 GeoNames

Το GeoNames λεξιλόγιο καθιστά ικανή τη δυνατότητα να προσθέσουμε γεωγραφική σημασιολογική πληροφορία στο Διαδίκτυο. Υπάρχουν πάνω από 8,3 εκατομμύρια τοπωνύμια geonames που έχουν ένα μοναδικό URL, και αντιστοιχούν σε μία διαδικτυακή υπηρεσία RDF. Επίσης υπάρχουν και άλλες υπηρεσίες που

περιγράφουν τις σχέσεις μεταξύ των τοπωνυμιών. Το GeoNames χρησιμοποιεί την «303 (See other)» ανακατεύθυνση για να ξεχωρίσει την έννοια (το αντικείμενο όπως είναι) από το έγγραφο που αναφέρεται σ' αυτήν. Π.χ. για μία πόλη μπορούμε να έχουμε αυτά τα δύο URIs:

1. http://sws.geonames.org/734077
2. http://sws.geonames.org/734077/about.rdf

Το πρώτο URI αναφέρεται στην πόλη και το χρησιμοποιούμε όταν θέλουμε να αναφερθούμε σε αυτήν.

Το δεύτερο URI είναι το έγγραφο με τις πληροφορίες που έχει το geonames για την πόλη.

Ο web server του geonames έχει ρυθμιστεί, να ανακατευθύνει τις αιτήσεις από το πρώτο URI στο δεύτερο. Αυτή η ανακατεύθυνση πληροφορεί τους Semantic Web Agents, ότι στον server του geonames υπάρχουν πληροφορίες για την πόλη[9].

2.5.4 vCard

Το vCard είναι ένα λεξιλόγιο το οποίο προσπαθεί να καταστήσει δυνατή την κοινή και συνεπή περιγραφή των ατόμων και των οργανισμών-επιχειρήσεων καθώς και να τα κωδικοποιήσει αυτά σε RDF μορφή. Το RDF χρησιμοποιεί το XML Namespace για να προσδιορίσει μοναδικά το σχήμα μεταδεδομένων και της έκδοσης. Για το vCard, έχει οριστεί το εξής URI να είναι το RDF vCard Namespace: «http://www.w3.org/2006/vcard/ns#». Η ρητή χρήση αυτού του XML Namespace στο RDF σημαίνει ότι δεν υπάρχει ανάγκη να υποστηρίζει το προφίλ vCard και την έκδοση των τύπων[27].

2.6 ΕΠΙΛΟΓΟΣ

Οι γλώσσες περιγραφής τύπων, αναπαριστούν τύπους και οντότητες, έτσι ώστε να μπορούμε να περιγράψουμε όσον το δυνατόν καλύτερα τους πόρους που αναπαρίστανται στο RDF μοντέλο. Υπάρχουν πολλά λεξιλόγια που μπορούμε να χρησιμοποιήσουμε για να περιγράψουμε συγκεκριμένους πόρους, έτσι ώστε να έχουμε ακόμα καλύτερη περιγραφή των πόρων. Υπάρχει φυσικά η δυνατότητα να δημιουργήσουμε και δικά μας λεξιλόγια αλλά δεν προτείνεται αυτό, καθώς δεν θα είναι ευρέως γνωστά αυτά, γιατί είναι γενικά αποδεκτό να χρησιμοποιούνται γνωστά λεξιλόγια. Οι πληροφορίες αναπαρίστανται σε μορφή XML, η οποία είναι κατανοητή στους υπολογιστές, αλλά δυσανάγνωστη στους ανθρώπους, οπότε, χρειάζεται να μπορούμε να τις αναπαραστήσουμε και με άλλον τρόπο για να μπορούμε να τις διαχειριστούμε- κατανοήσουμε πιο εύκολα.

ΚΕΦΑΛΑΙΟ 3

ΓΛΩΣΣΕΣ ΣΕΙΡΙΟΠΟΙΗΣΗΣ

3.1 ΕΙΣΑΓΩΓΗ

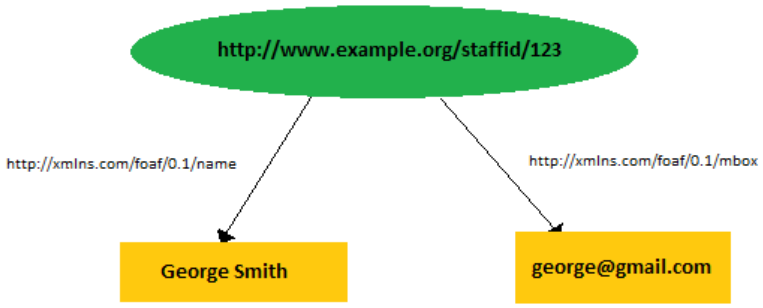
Το RDF μοντέλο βασίζεται στην XML και ο πιο συνηθισμένος τρόπος απεικόνισής του είναι η μορφή RDF/XML. Αν και η μορφή RDF/XML είναι κατανοητή στους υπολογιστές, για τους ανθρώπους δεν είναι ευανάγνωστη. Υπάρχουν και άλλοι τρόποι σειριοποίησης για την αναπαράσταση του RDF μοντέλου, ώστε να γίνεται πιο εύκολα κατανοητό από τους ανθρώπους, ώστε να μπορούν να το επεξεργαστούν πιο εύκολα αν χρειαστεί.

3.2 N-Triples

Η N-Triples σημειογραφία είναι η πιο απλή μορφή σειριοποίησης. Αυτή η απλότητά της είναι που κάνει την N-Triples χρήσιμη όταν γράφουμε εμείς κάποια datasets για εφαρμογές με σκοπό να τις ελέγξουμε.

Κάθε γραμμή σε ένα αρχείο που χρησιμοποιεί τη μορφή σειριοποίησης N-Triples, αντιπροσωπεύει μια ενιαία δήλωση που περιέχει ένα υποκείμενο (subject), ένα κατηγόρημα (predicate) και ένα αντικείμενο (object) που ακολουθείται από μία τελεία. Το υποκείμενο, το κατηγόρημα και το αντικείμενο, εκφράζονται ως απόλυτα URI που περικλείονται ανάμεσα στα σύμβολα «<....>», εκτός από τους ανώνυμους κόμβους ή κενούς κόμβους (blank nodes) και τα Literals τα οποία δεν εκφράζονται με τον τρόπο αυτό. Χρησιμοποιούμε απόλυτα URIs στην N-Triples μορφή, επειδή δεν υποστηρίζει τα prefixes, για να είναι όσο το δυνατόν πιο απλή η μορφή αυτή. Τα υποκείμενα και τα αντικείμενα που είναι ανώνυμοι κόμβοι αναπαριστώνται σαν «_:name». Τα strings Literals μπορούν προαιρετικά να καθορίσουν και την γλώσσα με το «@lang» όπου το «lang» είναι ένας κωδικός ISO 639 language. Επίσης τα Literals μπορούν να προσφέρουν πληροφορίες για τον τύπο δεδομένων με «^^type» όπου το «type» είναι ένας κοινός XSD (XML Schema Definition) τύπος δεδομένων.

Π.χ. έχουμε τον εξής γράφο:



Σχήμα 3.1: Ένας απλός γράφος

Σε μορφή N-Triples θα αναπαρασταθεί έτσι:

```
<http://www.example.org/staffid/123> <http://xmlns.com/foaf/0.1/name> “George Smith” .
```

```
<http://www.example.org/staffid/123>  <http://xmlns.com/foaf/0.1/mbox>
<mailto:george@gmail.com> .
```

3.3 N3

Αν και γενικά η ιδέα του N-Triples είναι πολύ απλή, έχει όμως ένα πρόβλημα, το ότι επαναλαμβάνονται πολλές φορές τα ίδια URIs. Σε μικρά αρχεία αυτό μπορεί να μην θεωρείται πρόβλημα, αλλά όσο αυξάνεται το μέγεθος, το πρόβλημα γίνεται όλο και πιο εμφανές, καθώς σε έναν RDF γράφο, κάθε μία σχέση ανάμεσα σε δύο κόμβους είναι μία τριπλέτα και ένας κόμβος μπορεί να έχει έναν μεγάλο μέγεθος σχέσεων. Με την μορφή σειριοποίησης Notation3 ή N3 μπορούμε να λύσουμε αυτό το πρόβλημα καθώς μας επιτρέπει να χρησιμοποιήσουμε τα prefixes. Επίσης επειδή ένα υποκείμενο (subject) μπορεί να έχει πολλές σχέσεις, οπότε και να εμφανίζεται πολλές φορές, το N3 μας δίνει τη δυνατότητα να μειώσουμε την επαναλαμβανόμενη εμφάνισή του. Αφού μας επιτρέπει να συνδυάσουμε πολλές δηλώσεις στο ίδιο υποκείμενο (subject) χρησιμοποιώντας το ελληνικό ερωτηματικό «;» μετά την πρώτη δήλωση, έτσι μετά δηλώνουμε μόνο το κατηγόρημα (predicate) και το αντικείμενο (object) χρησιμοποιώντας το ίδιο υποκείμενο (subject). Την τελευταία δήλωση την ακολουθεί μια τελεία. Επίσης μας παρέχει μια αντίστοιχη συντόμευση και για τους κενούς κόμβους που έχουν πολλές δηλώσεις.

Π.χ. ο γράφος του σχήματος 3.1 σε N3 μορφή μπορεί να γραφεί ως εξής:

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

<http://www.example.org/staffid/123> foaf:name “George Smith” .

<http://www.example.org/staffid/123> foaf:mbox mailto:george@gmail.com .
```

ή αλλιώς:

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

<http://www.example.org/staffid/123> foaf:name “George Smith” ;

                                     foaf:mbox mailto:george@gmail.com .
```

3.4 Turtle

Η Turtle είναι και αυτή μια μορφή σειριοποίησης, η οποία είναι υπερσύνολο της N-Triples και υποσύνολο της N3. Και η Turtle με την σειρά της επιτρέπει την απεικόνιση του RDF γράφου σε μορφή κειμένου. Όπως και στο N3 έτσι και στην Turtle μπορούμε να χρησιμοποιήσουμε τα prefixes και να συντομεύουμε τις δηλώσεις όταν έχουν το ίδιο υποκείμενο (subject). Επίσης μπορούμε να γράψουμε και σχόλια μετά το σύμβολο «#», το οποίο δεν αποτελεί τμήμα άλλης λεξιλογικής ένδειξης (token) και ακολουθεί μετά το τέλος της σειράς. Και στην Turtle μπορούμε να δηλώσουμε τον τύπο δεδομένων των Literals και την γλώσσα των string Literals.

Όταν υπάρχει η εξής τριπλέτα:

```
exp:George a foaf:Person
```

είναι σαν να γράφουμε:

```
exp:George <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> foaf:Person
```

καθώς η ένδειξη (token) «a» στην θέση του κατηγορήματος (predicate) σε μία Turtle τριπλέτα αντιπροσωπεύει το URI:

«http://www.w3.org/1999/02/22-rdf-syntax-ns#type».

3.5 RDF/XML

Η μορφή σειριοποίησης RDF/XML είναι η αρχική προδιαγραφή του W3C για την μετατροπή του RDF, τόσο ως μία περιγραφή του RDF σαν ένα μοντέλο δεδομένων όσο και ως XML σαν μία έκφραση των μοντέλων RDF. Γι' αυτό τον λόγο πολλοί άνθρωποι συχνά αναφέρονται στο RDF/XML σαν RDF, αλλά είναι σημαντικό να αναγνωρίσουμε ότι είναι μόνο μία δυνατή αναπαράσταση από έναν RDF γράφο.

Η RDF/XML μορφή είναι αρκετά δύσκολη να κατανοηθεί από τους ανθρώπους, αλλά παρόλα αυτά είναι μια από τις πιο συχνά χρησιμοποιούμενες μορφές σειριοποίησης. Οι διαδρομές πάντα περιγράφονται, αρχίζοντας με έναν κόμβο γράφου, χρησιμοποιώντας το στοιχείο (element) «rdf:Description». Η

αναφορά URI για τον κόμβο μπορεί να προσδιοριστεί στο στοιχείο description με την ιδιότητα (attribute) «rdf:about».

Οι κενοί κόμβοι μπορούν να προσδιοριστούν με ένα τοπικό προσδιοριστικό χρησιμοποιώντας την ιδιότητα «rdf:NodeID». Τα κατηγορήματα (predicates) προσδιορίζονται ως στοιχεία-παιδιά (child elements) του «rdf:Description» κόμβου, τα οποία θα έχουν τα δικά τους παιδιά που θα αναπαριστούν κόμβους γράφου. Τα Literals μπορούν να προσδιοριστούν είτε ως κείμενο ενός στοιχείου είτε ως μια ιδιότητα στο «rdf:Description» στοιχείο.

Π.χ. με την μορφή RDF/XML ο γράφος του σχήματος 3.1 θα περιγράφονταν ως εξής:

```
<rdf:RDF
        xmlns:foaf=”http://xmlns.com/foaf/0.1/”>
    <rdf:Description rdf:about=”http://www.example.org/staffid/123”>
        <foaf:name>George Smith</foaf:name>
        <foaf:mbox rdf:resource=”mailto:george@gmail.com”/>
    </rdf:Description>
</rdf:RDF>
```

3.6 RDFa

Το RDFa δεν είναι μία καθαρά μορφή σειριοποίησης για το RDF, αλλά περισσότερο ένας τρόπος να προσθέσουμε RDF δεδομένα σε (X)HTML σελίδες. Η ιδέα είναι να δημοσιεύουμε τα περιεχόμενά μας μία φορά μόνο, με τον συνδυασμό των δεδομένων που είναι κατανοητά από τους ανθρώπους και τους υπολογιστές μαζί.

Το RDFa χρησιμοποιεί ένα μικρό σύνολο από ιδιότητες της XML, οι οποίες ενσωματώνονται στις ετικέτες (tags) του περιεχομένου μίας (X)HTML σελίδας που υπάρχει ήδη, ώστε να προσδιοριστεί η σημασιολογία στις πληροφορίες που προβάλλονται.

Π.χ. έστω ότι έχουμε μία σελίδα, η οποία περιέχει κάποιες πληροφορίες για την ίδια την σελίδα:

```
<html>
<head>
        . . . .
</head>
<body>
        <h2> The title of this web page </h2>
        <p> Date: 2011-11-21 </p>
        . . . .
</body>
```

Οι πληροφορίες αυτές όμως γίνονται κατανοητές μόνο από τους ανθρώπους. Για να γίνουν κατανοητές από τους υπολογιστές χρειάζονται περίπλοκες μέθοδοι. Όμως χρησιμοποιώντας το RDFa, μπορούμε να «υποσημειώσουμε» τα δομημένα δεδομένα και να τα κάνουμε σαφή:

```
<html>
<head>
        . . . .
</head>
<body>
        <h2 property="http://purl.org/dc/terms/title"> The title of this web page
        </h2>
        <p> Date: <span property="http://purl.org/dc/terms/created"> 2011-11-  21
</span> </p>
        . . . .
```

```
</body>

</html>
```

3.7 ΕΠΙΛΟΓΟΣ

Υπάρχουν πολλές μορφές σειριοποίησης, ώστε να μετατρέπουμε τον RDF γράφο σε μορφή κειμένου. Επιπλέον υπάρχουν και οι μορφές σειριοποίησης JSON και N-Quads. Έτσι, έχουμε τη δυνατότητα να χρησιμοποιούμε, όποια μορφή σειριοποίησης θέλουμε, ανάλογα με τις ανάγκες μας, καθώς άλλες είναι πιο κατανοητές από ανθρώπους ενώ άλλες όχι.

ΚΕΦΑΛΑΙΟ 4

SPARQL

4.1 ΕΙΣΑΓΩΓΗ

Η SPARQL είναι μία γλώσσα ερωτημάτων για δεδομένα σε μορφή RDF. Έτσι, μας δίνεται η δυνατότητα να εκτελούμε ερωτήματα (queries) πάνω στα RDF δεδομένα και να παίρνουμε ακριβώς τα αποτελέσματα που θέλουμε.

4.2 Τι είναι η SPARQL

Η SPARQL (Simple Protocol and RDF Query Language) όπως αναφέρει και το όνομά της είναι μία γλώσσα ερωτημάτων και ένα πρωτόκολλο, το οποίο ορίζει τον τρόπο με τον οποίο θα γίνεται η επικοινωνία και τον τρόπο που θα εκτελούνται τα ερωτήματα (queries) και θα επιστρέφονται τα αποτελέσματα. Η SPARQL προφέρεται «sparkle» και είναι ένα πρότυπο που δημιουργήθηκε από την RDF Data Access Working Group (DAWG) και αποτελεί μία από τις βασικές τεχνολογίες του Semantic Web. Στις 15 Ιανουαρίου του 2008 η SPARQL έγινε επίσημα σύσταση του W3C (World Wide Web Consortium) καθώς και η SPARQL 1.1 τον Μάρτιο του 2013[34].

Με την SPARQL εκτελούμε ερωτήματα σε RDF δεδομένα. Δηλαδή με ένα ερώτημα SPARQL, προσδιορίζουμε το μέρος της πληροφορίας που θέλουμε από το σύνολο των δεδομένων, καθώς αν το οπτικοποιήσουμε είναι σαν να παίρνουμε ένα συγκεκριμένο μέρος του γράφου, το οποίο πληροί κάποιες συγκεκριμένες προϋποθέσεις. Οι προϋποθέσεις αυτές, στο ερώτημα SPARQL γράφονται με μορφή τριπλετών όπως οι RDF τριπλέτες. Επίσης, στις τριπλέτες των ερωτημάτων μπορούμε να έχουμε και μεταβλητές, οι οποίες μας δίνουν μια ευελιξία στο ταίριασμα των δεδομένων. Έτσι, με τις μεταβλητές αυτές μπορούμε να παίρνουμε δεδομένα, στα οποία μπορεί να έχουμε θέσει κάποια κριτήρια και θα τα παίρνουμε χωρίς να χρειάζεται να γνωρίζουμε τη δομή τους και το πώς είναι οργανωμένα. Οπότε μας δίνεται η δυνατότητα να εκτελούμε ερωτήματα στα δεδομένα χωρίς απαραίτητα να γνωρίζουμε κάτι γι' αυτά.

4.3 Δομή ενός SPARQL ερωτήματος

Ένα SPARQL ερώτημα έχει ανάλογη δομή με ένα SQL ερώτημα. Αν και η δομή είναι σχεδόν παρόμοια, στην ουσία όμως είναι διαφορετικές γλώσσες ερωτημάτων. Σε ένα SPARQL ερώτημα, προαιρετικά, μπορούμε να παρέχουμε τα prefixes που υπάρχουν και χρησιμοποιούμε στα δεδομένα, έτσι ώστε να γνωρίζουμε τα prefixes που μπορούμε να χρησιμοποιήσουμε στα ερωτήματα. Η δομή ενός SPARQL ερωτήματος είναι η εξής: όπως και στην SQL έτσι και στην SPARQL έχουμε το πεδίο «SELECT» όπου βάζουμε τις μεταβλητές που θέλουμε να εμφανίσουμε. Μετά ακολουθεί προαιρετικά το πεδίο «FROM». Στο πεδίο «FROM» η διαφορά είναι, ότι αντίθετα με την SQL που δηλώνουμε σε ποιους πίνακες θα

εκτελέσουμε τα ερωτήματα, στην SPARQL δίνουμε ένα ή περισσότερα URIs που αναφέρονται σε γράφους που υπάρχουν στο συγκεκριμένο RDF σύνολο δεδομένων. Όταν εκτελούμε ένα SPARQL ερώτημα σε ένα SPARQL endpoint, δηλαδή ένα «σημείο» το οποίο στην ουσία είναι ένα συγκεκριμένο URL όπου μέσω του πρωτοκόλλου HTTP εκτελούμε SPARQL ερωτήματα, διαμέσου μιας διεπαφής που μας παρέχει το ίδιο το SPARQL endpoint, τότε δεν χρειάζεται να βάλουμε το πεδίο «FROM», καθώς όταν στέλνουμε το ερώτημα αυτό θα μετατραπεί και θα συμπληρωθεί αυτόματα το πεδίο με τον default γράφο. Τέλος, έχουμε το πεδίο «WHERE» όπου, όπως και στην SQL ορίζουμε ποιές προϋποθέσεις πρέπει να πληρούν τα δεδομένα που θέλουμε να επιστραφούν. Η διαφορά φυσικά με την SQL είναι ότι εδώ η SPARQL μας δίνει τη δυνατότητα να εφαρμόσουμε και κάποια φίλτρα αν θέλουμε, για να έχουμε ακόμα πιο ακριβή και σωστά αποτελέσματα. Αυτή είναι η πιο συνηθισμένη λειτουργία-μορφή σε ένα SPARQL ερώτημα, καθώς ένα SPARQL ερώτημα έχει και άλλες λειτουργίες-μορφές. Οι λειτουργίες-μορφές ενός SPARQL ερωτήματος είναι οι εξής:

- Η SELECT, η οποία περιγράφεται πιο πάνω, επιστρέφει τα δεδομένα που πληρούν τις προϋποθέσεις που υπάρχουν στο πεδίο «WHERE».
- Η CONSTUCT[33], η οποία έχει ίδια τα υπόλοιπα πεδία με την SELECT, επιστρέφει τριπλέτες τις οποίες εισάγει σε έναν γράφο, για τον οποίο έχουμε δώσει ένα URI. Μας επιτρέπει να παίρνουμε τριπλέτες χωρίς να τις αλλάζουμε και να δημιουργούμε νέες τριπλέτες. Έτσι μπορούμε να δημιουργήσουμε και να μετασχηματίσουμε τα δεδομένα όπως θέλουμε εμείς. Π.χ. έστω ότι έχουμε τις εξής τριπλέτες:

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

_;a foaf:name “George”
_;a foaf:mbox <mailto:george@gmail.com>
```

Και εκτελούμε το εξής SPARQL ερώτημα:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/> .
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .

CONSTRUCT { <http://example.org/person#George> vcard:FN ?name}
WHERE { ?x foaf:name ?name}
```

Θα πάρουμε σαν αποτέλεσμα:

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
```

```
<http://example.org/person#George> vcard:FN “George” .
```

- Η ASK[33], η οποία ρωτάει εάν υπάρχει ή όχι απάντηση σε ένα ερώτημα. Δεν επιστρέφει πληροφορίες σχετικά με τις πιθανές απαντήσεις του ερωτήματος, απλά επιστρέφει εάν υπάρχει απάντηση ή όχι. Δηλαδή ελέγχει αν υπάρχει στο dataset η συγκεκριμένη περιγραφή από τριπλέτες, επιστρέφοντας μία Boolean τιμή «true» ή «false». Π.χ. έστω ότι έχουμε τις εξής τριπλέτες:

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

_:a foaf:name “George” .
_:a foaf:homepage <http://example.org/george/> .
_:b foaf:name “Bob” .
_:b foaf:mbox <mailto:bob@gmail.com> .
```

Και εκτελούμε το εξής SPARQL ερώτημα:

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

ASK { ?x foaf:name “George” . }
```

Θα πάρουμε ως απάντηση: «TRUE».
Ενώ αν εκτελέσουμε το εξής SPARQL ερώτημα:

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

ASK { ?x foaf:name “Georg” ;
        foaf:mbox <mailto:george@gmail.com> . }
```

Θα πάρουμε ως απάντηση: «FALSE».

- Η DESCRIBE[33], η οποία επιστρέφει ως αποτέλεσμα έναν RDF γράφο που περιέχει RDF δεδομένα, τα οποία περιγράφουν τον συγκεκριμένο πόρο. Οι τριπλέτες με τις οποίες θα περιγραφεί ο πόρος προσδιορίζονται από τον επεξεργαστή ερωτημάτων της SPARQL.

Ακόμα, όπως στην SQL έτσι και στην SPARQL μπορούμε να προσθέσουμε ή να τροποποιήσουμε τα δεδομένα. Αυτή η δυνατότητα προστέθηκε με την έκδοση της SPARQL 1.1. Οπότε έχουμε και τις εξής λειτουργίες-μορφές:

- INSERT DATA, η οποία προσθέτει τριπλέτες σε έναν γράφο ενός triplestore, το οποίο είναι μια ειδική κατηγορία βάσεων γράφου και είναι κατασκευασμένα για την αποθήκευση και την ανάκτηση RDF δεδομένων.
- DELETE DATA, η οποία διαγράφει τριπλέτες σε έναν γράφο ενός triplestore.
- LOAD, η οποία διαβάζει ένα RDF έγγραφο από μία URI και φορτώνει τις τριπλέτες του σε ένα γράφο ενός triplestore.
- CLEAR, η οποία διαγράφει όλες τις τριπλέτες ενός γράφου από ένα triplestore.
- CREATE, η οποία δημιουργεί έναν νέο γράφο σε ένα triplestore.
- DROP, η οποία διαγράφει έναν ή περισσότερους γράφους από ένα triplestore.
- COPY, η οποία αντιγράφει τα δεδομένα ενός γράφου σε έναν άλλον, αν ο δεύτερος γράφος είχε δεδομένα τότε διαγράφονται πριν εισαχθούν τα νέα.
- MOVE, η οποία μεταφέρει τα δεδομένα ενός γράφου σε έναν άλλον. Ο πρώτος γράφος μετά διαγράφεται και αν ο δεύτερος γράφος είχε δεδομένα τότε διαγράφονται πριν εισαχθούν τα νέα.
- ADD, η οποία προσθέτει τα δεδομένα ενός γράφου σε έναν άλλον. Τα δεδομένα του πρώτου γράφου παραμένουν όπως ήταν και τα δεδομένα που είχε ο δεύτερος γράφος διατηρούνται ανέπαφα, απλά προστίθενται και τα νέα.

4.4 Παραδείγματα SPARQL ερωτημάτων

Τα παραδείγματα σύνταξης εντολών SPARQL που ακολουθούν στις υποενότητες 4.4.1, 4.4.2, 4.4.3 χρησιμοποιούν τον εξής γράφο δεδομένων RDF:

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

@prefix ex: <http://example.org/staffid#> .

@prefix ab: <http://learningsparql.com/ns/addressbook#> .



ex:123  foaf:name  “George Smith” ;

        foaf:homepage  <http://example.org/george> ;

        ab:mail  “george@gmail.com” ;
```

```
        ab:mail  “george@yahoo.com” .

ex:124  foaf:name  “Ted Mosby” ;

        foaf:homepage  <http://example.org/ted> ;

        ab:mail  “ted@gmail.com” ;

        ab:homeTel  “2310-123-456” .

ex:125  foaf:name  “John Doe” ;

        ab:homeTel  “2310-456-789” ;

        ab:city  “Thessaloniki” .
```

4.4.1 Ένα απλό SPARQL ερώτημα

Ένα θέλουμε να βρούμε τα emails του George μπορούμε να εκτελέσουμε το εξής ερώτημα:

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

PREFIX ex: <http://example.org/staffid#> .

PREFIX ab: <http://learningsparql.com/ns/addressbook#> .


SELECT  ?GeorgeEmail

WHERE

{

        ?person  foaf:name  “George Smith” .

        ?person  foaf:name  ?GeorgeEmail .

}
```

Ο επεξεργαστής ερωτημάτων, καθώς ψάχνει στις τριπλέτες του dataset που έχουμε και βρει ένα ταίριασμα για το πρώτο πρότυπο τριπλέτας του ερωτήματος, που είναι η

τριπλέτα: «ex:123 foaf:name "George Smith"», τότε θα δεσμεύσει την τιμή «ex:123» στην μεταβλητή «?person». Επειδή, το «?person» είναι στην θέση του υποκειμένου στο πρώτο πρότυπο τριπλέτας, τότε με υποκείμενο το «ex:123» ο επεξεργαστής ερωτημάτων θα προσπαθήσει να βρει μία εγγραφή που θα ταιριάζει στο πρότυπο της τριπλέτας αυτής. Οπότε θα επιστρέψει σαν αποτέλεσμα το εξής:

GeorgeEmail
"george@gmail.com>"
"george@yahoo.com"

4.4.2 Αναζητώντας Strings

Όταν θέλουμε να αναζητήσουμε ένα κομμάτι δεδομένων, για το οποίο δεν γνωρίζουμε ούτε τι υποκείμενο ή ιδιότητα μπορεί να έχει, τότε μπορούμε να χρησιμοποιήσουμε ένα φίλτρο, το οποίο θα αναθέσει στον επεξεργαστή ερωτημάτων να επιστρέψει μόνο όσες τριπλέτες πληρούν τις συγκεκριμένες προϋποθέσεις. Έστω, ότι θέλουμε να ψάξουμε ένα συγκεκριμένο string, τότε μπορούμε να εκτελέσουμε το εξής ερώτημα:

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

PREFIX ex: <http://example.org/staffid#> .

PREFIX ab: <http://learningsparql.com/ns/addressbook#> .



SELECT *

WHERE

{

        ?s ?p ?o .

        FILTER (regex(?o, "gmail", "i"))

}
```

Τα αποτελέσματα που θα επιστραφούν είναι:

s	p	o
ex:123	ab:mail	“george@gmail.com>”
ex:124	ab:mail	“ted@gmail.com”

Επειδή στο «FILTER» χρησιμοποιούμαι την μέθοδο «regex», η οποία ελέγχει για strings που ταιριάζουν σε ένα συγκεκριμένο πρότυπο. Δηλαδή ελέγχει για κάθε τριπλέτα που υπάρχει, αν στο αντικείμενο εμφανίζεται το string «gmail».

4.4.3 «Λάθος» ερωτήματα

Αν εκτελέσουμε το εξής ερώτημα:

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

PREFIX ex: <http://example.org/staffid#> .

PREFIX ab: <http://learningsparql.com/ns/addressbook#> .



SELECT  ?GeorgeEmail ?GeorgehomeTel

WHERE

{

        ?person  foaf:name  “George Smith” .

        ?person  ab:mail  ?GeorgeEmail .

        ?person  ab:homeTel  ?GeorgehomeTel .

}
```

Θα πάρουμε σαν απάντηση:

?GeorgeEmail	?GeorgehomeTel

Αυτό, θα γίνει, καθώς θα ψάξει ο SPARQL επεξεργαστής ερωτημάτων για τριπλέτες στο dataset που πληρούν όλες τις προϋποθέσεις που έχουμε θέσει. Έτσι, δεν υπάρχει πόρος που να πληροί όλες τις προϋποθέσεις, οπότε ο SPARQL επεξεργαστής ερωτημάτων δεν θα επιστρέψει κανένα δεδομένο.

Όμως αν θέσουμε το «GeorgehomeTel» σαν «OPTIONAL»:

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

PREFIX ex: <http://example.org/staffid#> .

PREFIX ab: <http://learningsparql.com/ns/addressbook#> .

SELECT  ?GeorgeEmail ?GeorgehomeTel

WHERE

{

        ?person  foaf:name  “George Smith” .

        ?person  ab:mail  ?GeorgeEmail .

        OPTIONAL { ?person  ab:homeTel  ?GeorgehomeTel . }

}
```

Θα μας επιστρέψει σαν απάντηση:

GeorgeEmail	GeorgehomeTel
george@gmail.com	
george@yahoo.com	

Αυτό, θα γίνει, επειδή τα κριτήρια με «OPTIONAL» δεν χρειάζονται να πληρούνται, καθώς πλέον είναι προαιρετικά. Οπότε ο SPARQL επεξεργαστής ερωτημάτων, θα μας επιστρέψει τα δεδομένα που πληρούν τις προϋποθέσεις, που δεν είναι «OPTIONAL» και όσα από αυτά τα δεδομένα πληρούν και τις «OPTIONAL» προϋποθέσεις, θα μας εμφανίσει και τις τιμές που έχουν τα πεδία αυτά.

4.5 ΕΠΙΛΟΓΟΣ

Με την SPARQL, μπορούμε να εκτελούμε ερωτήματα, σε δεδομένα που δεν χρειάζεται να ξέρουμε τη δομή τους και να παίρνουμε αποτελέσματα, ώστε να τα χρησιμοποιήσουμε όπως θέλουμε. Η SPARQL, βοηθά στο να παίρνουμε συλλογή πληροφοριών από δεδομένα που είναι σε μορφή RDF.

ΚΕΦΑΛΑΙΟ 5

LINKED DATA

5.1 ΕΙΣΑΓΩΓΗ

Η ιδέα των Linked Data είναι νεότερη από αυτήν του Semantic Web, αλλά μερικές φορές είναι πιο εύκολο να θεωρούμε πως το Semantic Web δημιουργήθηκε πάνω στις ιδέες των Linked Data. Τα Linked Data δεν είναι μια προδιαγραφή αλλά ένα σύνολο βέλτιστων πρακτικών που προσφέρουν μια υποδομή δεδομένων, κάνοντας ευκολότερη τη διανομή των δεδομένων διά μέσου του διαδικτύου. Επίσης έχουν «χτιστεί» πάνω στις υπάρχουσες τεχνολογίες του διαδικτύου: HTTP και URIs[8].

5.2 Βασικές αρχές

Το Semantic Web δεν πρόκειται απλά μόνο για τη δημοσίευση δεδομένων στο διαδίκτυο αλλά πρόκειται για τη δημιουργία συνδέσμων, έτσι ώστε ένας άνθρωπος ή μια μηχανή να μπορεί να εξερευνήσει το διαδίκτυο των δεδομένων. Με τα Linked Data μπορούμε να βρούμε άλλα συσχετισμένα δεδομένα, όταν έχουμε μερικά. Αυτό επιτυγχάνεται καθώς τα Linked Data μας δίνουν τη δυνατότητα να συνδέουμε δεδομένα από διαφορετικές πηγές μεταξύ τους. Οι κανόνες που προτάθηκαν από τον Tim Berners-Lee για τα Linked Data είναι οι εξής:

1. Χρησιμοποιούμε URIs σαν ονόματα για τα πράγματα.
2. Χρησιμοποιούμε τα HTTP URIs έτσι ώστε οι άνθρωποι να μπορούν να αναζητούν αυτά τα ονόματα.
3. Όταν κάποιος αναζητά ένα URI, να του παρέχονται χρήσιμες πληροφορίες, χρησιμοποιώντας τα πρότυπα (RDF, SPARQL).
4. Στα δεδομένα εμπεριέχονται σύνδεσμοι σε άλλα URIs, έτσι ώστε να μπορούν να εξερευνηθούν και άλλα πράγματα[37].

Ο πρώτος κανόνας αναφέρει ότι τα URIs είναι ο καλύτερος διαθέσιμος τρόπος για να προσδιορίσουμε μοναδικά τα πράγματα, καθώς επίσης και να προσδιορίσουμε τις συνδέσεις μεταξύ τους.

Ο δεύτερος κανόνας αναφέρει ότι μπορεί να υπάρχουν URIs που ξεκινούν με «ftp:», «mailto:», ή με προθέματα που δημιουργήθηκαν από μια συγκεκριμένη κοινότητα. Το να χρησιμοποιούμε αυτά τα άλλα προθέματα μειώνουν τη διαλειτουργικότητα, και η διαλειτουργικότητα είναι όλο το θέμα.

Ο τρίτος κανόνας αναφέρει ότι ένα URI μπορεί να είναι απλά ένα όνομα και όχι πραγματικά μια διεύθυνση μιας ιστοσελίδας, αλλά όπως και να έχει πρέπει να υπάρχει κάτι εκεί. Μπορεί να είναι ένα HTML αρχείο, μια ιστοσελίδα, ή κάτι άλλο, το οποίο χρησιμοποιεί αναγνωρισμένα πρότυπα, όπως το RDFS και OWL, που σου επιτρέπουν να χρησιμοποιήσεις μια λίστα από όρους και πληροφορίες για τους όρους αυτούς και για τις συσχετίσεις τους, σε μια τέτοια μορφή έτσι ώστε να μπορούν να διαβαστούν από υπολογιστές.

Τέλος, ο τέταρτος κανόνας, αναφέρει, ότι τα URIs μπορεί να περιέχουν υπερσυνδέσμους που δείχνουν σε άλλα URIs, έτσι ώστε οι χρήστες να μπορούν να πλοηγούνται ανάμεσα σε διαφορετικές πηγές που περιέχουν πληροφορίες[8].

Οι τέσσερεις βασικές αρχές, είναι στην ουσία μια προσδόκιμη συμπεριφορά, οπότε, παραβιάζοντας έναν κανόνα δεν καταστρέφεται κάτι, αλλά χάνεται η ευκαιρία

να κάνουμε τα δεδομένα μας διασυνδεμένα. Αυτό με την σειρά του περιορίζει τους τρόπους με τους οποίους μπορούν να ξαναχρησιμοποιηθούν χωρίς να προβλέψουμε πως. Αυτή η απρόσμενη επαναχρησιμοποίηση των πληροφοριών είναι η προστιθέμενη αξία στο διαδίκτυο[37].

5.3 Χρησιμοποίηση των URIs ως ονόματα

Για να δημοσιεύσουμε δεδομένα στο διαδίκτυο, πρέπει πρώτα με κάποιο τρόπο να τα προσδιορίσουμε μαζί με κάποια αντικείμενα. Αυτά τα αντικείμενα, είναι οι ιδιότητες και οι σχέσεις τους, που θα περιγράφονται στα δεδομένα, που μπορεί να περιλαμβάνονται και αρχεία διαδικτύου καθώς και οντοτήτων πραγματικού κόσμου και αφηρημένες έννοιες, αφού τα Linked Data «χτίζουν» κατευθείαν πάνω στην αρχιτεκτονική του διαδικτύου, έτσι ο προσδιορισμός των πόρων των δεδομένων γίνεται με τα HTTP URIs. Τα HTTP URIs είναι χρήσιμα στην ονομασία των δεδομένων για δύο λόγους:

1. Παρέχουν έναν απλό τρόπο στη δημιουργία παγκόσμιων μοναδικών ονομάτων.
2. Δεν αποτελούν μόνο ένα όνομα αλλά επίσης ένα μέσω πρόσβασης πληροφοριών, περιγράφοντας την προσδιορισμένη έννοια[39].

5.4 Basic web look-up

Ο πιο απλός τρόπος να δημιουργήσουμε Linked Data είναι να χρησιμοποιήσουμε σε ένα αρχείο, ένα URI το οποίο δείχνει σε ένα άλλο. Όταν γράφουμε σε ένα RDF αρχείο, π.χ. «<http://example.org/smith>», τότε μπορούμε να χρησιμοποιήσουμε τοπικά προσδιοριστικά μέσα στο αρχείο, όπως π.χ. «#albert», «#brian» και «#carol». Σε μορφή N3 μπορεί να γραφεί ως εξής:

```
<#albert> fam:child <#brian>, <#carol>.
```

Ή σε RDF/XML μορφή ως εξής:

```
<rdf:Description about=”#albert”>

 <fam:child rdf:Resource=”#brian”>

 <fam:child rdf:Resource=”#carol”>

</rdf:Description>
```

Η αρχιτεκτονική του παγκόσμιου ιστού μας δίνει ένα παγκόσμιο αναγνωριστικό «http://example.org/smith#albert» για τον Albert. Αυτό είναι ένα πολύτιμο πράγμα, καθώς πλέον καθένας στον πλανήτη μπορεί τώρα να χρησιμοποιήσει αυτό το παγκόσμιο αναγνωριστικό για να αναφέρεται στον Albert και να έχει τη δυνατότητα να προσθέσει περισσότερες πληροφορίες. Π.χ., σε ένα άλλο αρχείο «<http://example.org/jones>» μπορεί κάποιος να γράψει:

```
<#denise> fam:child <#edwin>, <smith#carol>.
```

Ή σε RDF/XML μορφή ως εξής:

```
<rdf:Description about=”#denise”>

 <fam:child rdf:Resource=”#edwin”>

 <fam:child rdf:Resource=”http://example.org/smith#carol”>

</rdf:Description>
```

Σαφώς, είναι λογικό στον οποιονδήποτε που έρχεται σε επαφή με το αναγνωριστικό «http://example.org/smith#carol» να:

1. Σχηματίσει το URI περικόπτοντάς το πριν το σύμβολο «#»
2. Έχει πρόσβαση στο αρχείο και να λάβει τις πληροφορίες σχετικά με το «#carol».

Ονομάζουμε αυτό το URI εύρεση τιμών. Σε αυτό βασίζεται το Semantic Web[37].

5.5 URIs χωρίς hashes και HTTP 303

Υπάρχουν μερικές περιπτώσεις, όπου, το να χωρίζουμε τα προσδιοριστικά μέσα στα αρχεία, δεν λειτουργεί πολύ καλά. Μπορεί λογικά να υπάρχει ένα

παγκόσμιο σύμβολο ανά έγγραφο, και να υπάρχει μια απροθυμία να συμπεριλαμβάνεται το σύμβολο «#» στην URI όπως:

«http://wordnet.example.net/antidisesablishmentarianism#word»

Ιστορικά, τα αρχικά λεξιλόγια Dublin Core και FOAF δεν είχαν το σύμβολο «#» στα URIs τους. Σε κάθε περίπτωση, όταν τα HTTP URIs χωρίς το σύμβολο «#» χρησιμοποιούνται για αφηρημένες έννοιες και υπάρχει ένα έγγραφο που φέρει πληροφορίες σχετικά με αυτές, τότε:

1. Ένα HTTP GET αίτημα για την URI της έννοιας επιστρέφει «303 See Also» και δίνει το «Location: header», το URI του εγγράφου.
2. Το έγγραφο ανακτάται σαν κανονικό.

Αυτή η μέθοδος, έχει το μειονέκτημα ότι πρέπει να γίνει μία αίτηση HTTP ικανή προς πλοήγηση για κάθε ένα. Στην περίπτωση του Dublin Core, για παράδειγμα, το «dc:title» και το «dc:creator» κλπ στην πραγματικότητα εξυπηρετούνται από το ίδιο έγγραφο οντολογίας, αλλά κανείς δεν μπορεί να το γνωρίζει μέχρις ότου το καθένα να έχει επιστραφεί από τις HTTP ανακατευθύνσεις[37].

Το μειονέκτημα των URIs που χρησιμοποιούν το σύμβολο «#», είναι ότι όλοι οι πόροι που μοιράζονται την ίδια URI πριν το σύμβολο «#», θα επιστραφούν στο χρήστη, άσχετα αν ενδιαφερόταν μόνο για ένα URI και όχι για όλα. Οπότε, άμα έχουμε, ένα αρχείο με πολλές τριπλέτες , η τεχνική με τα URIs με το σύμβολο «#» μπορεί να οδηγήσει σε μεγάλη ποσότητα δεδομένων, που άσκοπα διαβιβάζονται στον πελάτη. Συνήθως τα URIs με το σύμβολο «#», χρησιμοποιούνται για να προσδιορίσουν όρους εντός ενός RDF λεξιλογίου, αφού συνήθως είναι μικρά και περιέχουν μερικές χιλιάδες τριπλέτες, οπότε είναι καλύτερο να ανακτούμε όλο το λεξιλόγιο από την αρχή, από το να χρειάζεται να ψάχνουμε τον κάθε όρο χωριστά.

5.6 Περιορισμοί στα περιηγήσιμα δεδομένα

Ένα σημαντικό πρότυπο είναι ένα σύνολο δεδομένων τα οποία μπορείς να εξερευνήσεις καθώς πλοηγείσαι από σύνδεσμο σε σύνδεσμο ανακτώντας δεδομένα. Πρακτικά, όταν τα δεδομένα είναι αποθηκευμένα σε δύο αρχεία, οποιαδήποτε δήλωση (statement) η οποία συνδέει πράγματα μεταξύ αυτών των δύο αρχείων, πρέπει να επαναληφθεί σε κάθε ένα από αυτά. Βέβαια αυτό, είναι ένα θέμα με τα

περιηγήσιμα δεδομένα, καθώς είναι ενάντια του πρώτου κανόνα της αποθήκευσης δεδομένων: «να μην αποθηκεύουμε τα ίδια δεδομένα σε δύο διαφορετικά σημεία», αφού αποθηκεύεται η ίδια πληροφορία σε δύο διαφορετικά σημεία και έτσι θα πρέπει να ενημερώνονται και τα δύο σημεία όταν υπάρχουν αλλαγές. Οπότε, τα δεδομένα πρέπει να αποθηκεύονται αμφίδρομα όταν οι πληροφορίες έχουν σημασία και για τις δύο πλευρές. Έτσι όταν έχουμε δεδομένα από πολλαπλές πηγές τότε πρέπει να υπάρχουν και συμβιβασμοί μεταξύ αυτών των πηγών. Οι συμβιβασμοί αυτοί μπορούν να διευθετηθούν, αν απαντήσουμε στην ερώτηση: «Αν κάποιος έχει το URI ενός αντικειμένου, ποιες σχέσεις με άλλα αντικείμενα θα ήταν χρήσιμες γι' αυτόν να γνωρίζει;». Αν και με αυτόν τον τρόπο χάνεται μέρος της πληροφορίας καθώς τα δεδομένα δεν είναι προσπελάσιμα και από τις δύο πλευρές, όμως διατηρείται το μέρος της πληροφορίας που είναι χρήσιμη και δεν κρατάμε ίδιες πληροφορίες αποθηκευμένες σε πολλαπλές πηγές[37].

5.7 Υπηρεσίες ερωτημάτων

Επειδή, ο όγκος των δεδομένων είναι μεγάλος, τον οποίο τον μοιράζουμε σε πολλά αρχεία, καθιστά τη διαδικασία εκτέλεσης ερωτημάτων στη βάση δεδομένων αναποτελεσματική. Θα ήταν πιο αποτελεσματικό, να παρέχεται μια υπηρεσία για ερωτήματα SPARQL, πάνω στα δεδομένα αυτά. Για να είναι τα δεδομένα αποτελεσματικά συνδεδεμένα, κάποιος ο οποίος έχει μόνο το URI από ένα αντικείμενο θα πρέπει να μπορεί να βρει το δρόμο για το SPARQL endpoint, από το οποίο θα μπορεί να πάρει πληροφορίες γι' αυτό. Αυτό θα μπορούσε να επιτευχθεί αν αυτός που αιτείται πληροφορίες για το συγκεκριμένο URI καθοδηγούνταν σε ένα αρχείο το οποίο θα περιείχε πληροφορίες μεταδεδομένων, όπου εκεί θα υπήρχαν πληροφορίες σχετικά με το ποια endpoint μπορεί να χρησιμοποιήσει και ποιες πληροφορίες για τις κλάσεις των URIs μπορεί να πάρει. Ακόμα όμως λεξιλόγια που να μπορούν να κάνουν κάτι τέτοιο δεν έχουν τυποποιηθεί[37].

5.8 ΕΠΙΛΟΓΟΣ

Τα Linked Data είναι συνδεδεμένα δεδομένα, τα οποία παρέχονται μαζί με πληροφορίες, οι οποίες τα περιγράφουν και είναι προσπελάσιμα, έτσι ώστε να είναι ικανά να χρησιμοποιηθούν από τον οποιονδήποτε. Φυσικά, μπορούν να διαβαστούν και από υπολογιστές και να αναπτυχθούν εφαρμογές που τα χρησιμοποιούν αυτά τα δεδομένα, έτσι ώστε να μπορούμε να αποκτούμε διάφορες πληροφορίες.

ΚΕΦΑΛΑΙΟ 6

SEMANTIC WEB

6.1 ΕΙΣΑΓΩΓΗ

Το Semantic Web ή αλλιώς Web of Data έχει να κάνει με τα δεδομένα, καθώς είναι ένα σύνολο προτύπων και βέλτιστων πρακτικών, για τη διανομή και την σημασιολογία των δεδομένων του διαδικτύου, για την χρησιμοποίησή τους από εφαρμογές. Δηλαδή, προσδιορίζει σημασιολογικά τις έννοιες που αναπαρίστανται στο διαδίκτυο, καθώς περιγράφει τις σχέσεις που έχουν μεταξύ τους και τις ιδιότητές τους. Επειδή υπάρχει μια ραγδαία αύξηση της ποσότητας των δεδομένων, τα οποία γίνονται διαθέσιμα στο Semantic Web, βοηθά στην ανάπτυξη πολλών νέων εφαρμογών[35].

6.2 Τι ακριβώς είναι το Semantic web

Το Semantic Web είναι ένα σύνολο προτύπων, όπως το World Wide Web το οποίο βασίζεται σε ένα σύνολο απλών προτύπων (HTTP, HTML και URIs). Έτσι και το Semantic Web έχει «χτιστεί» πάνω σε ορισμένα πρότυπα όπως το RDF μοντέλο δεδομένων, της SPARQL γλώσσας ερωτημάτων και των RDFSchema και OWL οντολογιών-γλωσσών για την αποθήκευση λεξιλογίων και οντολογιών, αλλά και πάνω σε ήδη υπάρχουσες τεχνολογίες (URI, XML). Είναι ένα σύνολο βέλτιστων πρακτικών για τη διανομή δεδομένων καθώς το Semantic Web χρησιμοποιεί τα Linked Data για να διαμοιράσει τα δεδομένα. Έτσι χρησιμοποιούνται τα URIs για την ονομασία των πραγμάτων, που είναι ένας γνωστός, απλός και εύχρηστος τρόπος. Επίσης προσφέρει τη σημασιολογία των δεδομένων, αφού μας δίνει τη δυνατότητα με την OWL να προσθέσουμε δεδομένα ή μεταδεδομένα που περιγράφουν τα δεδομένα, καθιστώντας τα ικανά να διαβαστούν και να είναι κατανοητά από τους υπολογιστές[8].

Οπότε, η διαφορά του Semantic Web από το World Wide Web, είναι ότι στο World Wide Web έχουμε έναν ιστό από αρχεία που απλά συνδέονται μεταξύ τους με τη βοήθεια των υπερσυνδέσμων, χωρίς να παρέχονται δεδομένα για τα δεδομένα, παρά μόνο ότι πληροφορίες μπορούμε να αποσπάσουμε από τα URIs, έτσι είναι αδύνατο για τους υπολογιστές να διαβάσουν και να ερμηνεύσουν τις πληροφορίες. Ενώ αντιθέτως στο Semantic Web, το οποίο διαχειρίζεται και συνδέει δεδομένα, στα οποία προσφέρει επιπλέον πληροφορίες, οι υπολογιστές μπορούν να ερμηνεύσουν, να επεξεργαστούν και να επιδράσουν στα δεδομένα αυτά.

6.3 Οντολογίες-Λεξιλόγια

Το Semantic Web, για να περιγράψει και να δώσει μια ιεραρχία στα δεδομένα ώστε να γίνονται κατανοητά από τους υπολογιστές χρησιμοποιεί κάποιες γλώσσες οντολογίας. Οι γλώσσες οντολογίας, όπως η OWL και RDFS χρησιμοποιούνται για να αναπαραστήσουν το νόημα των όρων που υπάρχουν στα διάφορα λεξιλόγια (FOAF, SKOS κτλ.) και στις σχέσεις που υπάρχουν μεταξύ των όρων αυτών. Η αναπαράσταση αυτή των όρων και των σχέσεών τους ονομάζεται οντολογία. Επειδή όμως, δεν υπάρχει σαφής διαχωρισμός, μεταξύ του τι αναφέρεται

σαν λεξιλόγιο και τι σαν οντολογία, συνήθως χρησιμοποιούμαι τον όρο οντολογία για να περιγράψουμε μια πιο πολύπλοκη συλλογή όρων.

6.4 Υλοποίηση του Semantic Web

Όπως το World Wide Web αποτελείται από αρχεία HTML, τα οποία συνδέονται με υπερσυνδέσμους, έτσι και το Semantic Web αποτελείται από αρχεία RDF, τα οποία συνδέονται με υπερσυνδέσμους. Μπορούμε να περιηγηθούμε στα δεδομένα με έναν απλό HTML browser όπως γίνεται και στο World Wide Web. Υπάρχουν φυσικά και Link Data browsers ή RDF browsers, για να περιηγηθούμε στα δεδομένα του Semantic Web, οι οποίοι μας προσφέρουν περισσότερες επιλογές. Οι RDF browsers είναι browsers που εκτελούνται σε έναν εξυπηρετητή (server side browsers), οπότε δεν χρειάζεται να τους εγκαταστήσουμε τοπικά και έτσι μπορούμε να τους χρησιμοποιήσουμε μέσω των HTML browsers. Για μερικούς RDF browsers υπάρχουν extensions ή add-ons τα οποία μπορούμε να εγκαταστήσουμε στους HTML browsers. Η διαφορά όταν ακολουθούμε έναν σύνδεσμο σε HTML σελίδες από το να ακολουθούμε έναν σύνδεσμο σε ένα αρχείο RDF, είναι ότι στον πρώτο απλά μεταφερόμαστε από μια σελίδα σε μία άλλη, ενώ στο δεύτερο μεταφερόμαστε σε μία άλλη πηγή δεδομένων, την οποία αν θέλουμε μπορούμε να την εξερευνήσουμε και να αποκτήσουμε περισσότερες πληροφορίες για αυτό που μας ενδιαφέρει. Π.χ. όταν κοιτάμε πληροφορίες για ένα άτομο σε μια πηγή δεδομένων, μπορεί να θέλουμε να μάθουμε περισσότερες πληροφορίες για την πόλη στην οποία ζει, έτσι ακολουθούμε τον σύνδεσμο που υπάρχει για την πόλη και μας μεταφέρει σε μια άλλη πηγή δεδομένων που μας παρέχει περισσότερες πληροφορίες για την πόλη.

Σε συνδυασμό με τα λεξιλόγια-οντολογίες που μας προσφέρει το Semantic Web, τα οποία μας προσφέρουν την σημασιολογία των δεδομένων και τον σχέσεών τους, ομαδοποιούν τα δεδομένα και μας δίνουν τη δυνατότητα να υπάρχει και μια ιεραρχία στα δεδομένα. Έτσι, οι αναζητήσεις μας γίνονται πιο συγκεκριμένες και αποκτούμε περισσότερες πληροφορίες γι' αυτό που πραγματικά μας ενδιαφέρει. Εάν μια πηγή δεδομένων μας παρέχει τη δυνατότητα να εκτελέσουμε και ερωτήματα SPARQL πάνω στα δεδομένα της, τότε μπορούμε να αποσπάσουμε ακριβώς τις πληροφορίες που θέλουμε, αφού είναι σαν να εκτελούμε ερωτήματα πάνω σε μια βάση δεδομένων.

6.5 ΕΠΙΛΟΓΟΣ

Το Semantic Web, δίνει σημασιολογία στα δεδομένα, προσφέροντας επιπλέον πληροφορίες γι' αυτά, καταστώντας τα έτσι δυνατό να τα συνδέουμε λογικά και να μπορούμε να βρίσκουμε παρόμοια δεδομένα. Σε συνδυασμό με τα Linked Data που μας επιτρέπουν να συνδέουμε δεδομένα από διαφορετικές πηγές του διαδικτύου, οι εφαρμογές που αναπτύσσονται για το Semantic Web παρουσιάζουν μια «έξυπνη» συμπεριφορά.

ΚΕΦΑΛΑΙΟ 7

D2RQ PLATFORM

7.1 ΕΙΣΑΓΩΓΗ

Η D2RQ πλατφόρμα είναι ένα σύστημα, το οποίο μας επιτρέπει να έχουμε πρόσβαση σε σχεσιακές βάσεις δεδομένων σαν εικονικό, μόνο προς ανάγνωση (read-only) RDF γράφο, βέβαια εδώ πρέπει να σημειωθεί ότι υπάρχει η D2RQ/Update και ο D2R/Update Server που είναι επεκτάσεις της D2RQ και του D2R Server και μας παρέχουν τη δυνατότητα να εκτελούμε SPARQL/Update δηλώσεις, όπως το INSERT και το DELETE στα χαρτογραφημένα δεδομένα. Προσφέρει RDF-based πρόσβαση στα περιεχόμενα των σχεσιακών βάσεων δεδομένων, χωρίς να χρειάζεται να τα αναπαράγουμε σε ένα RDF store. Χρησιμοποιώντας τη D2RQ, μπορούμε να κάνουμε τα εξής:

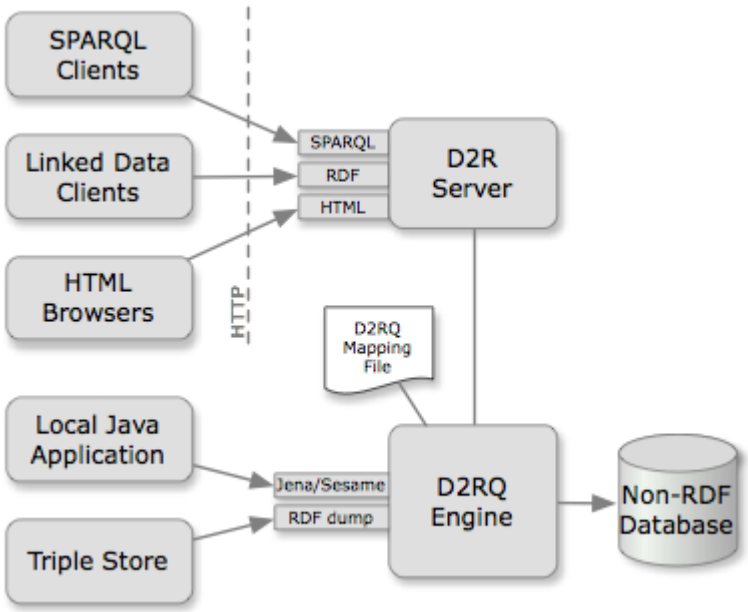
- Να εκτελέσουμε SPARQL ερωτήματα σε μία non-RDF βάση δεδομένων.
- Να έχουμε πρόσβαση στα περιεχόμενα της βάσης δεδομένων σε μορφή Linked Data στο Web.
- Να δημιουργήσουμε δικά μας dumps της βάσης σε διάφορες μορφές του RDF, ώστε να μπορούμε να τα φορτώσουμε σε ένα RDF store.
- Να έχουμε πρόσβαση σε πληροφορίες από non-RDF βάση χρησιμοποιώντας το Apache Jena API.

Επίσης η D2RQ πλατφόρμα είναι open source software και δημοσιεύτηκε σύμφωνα με το Apache license.

7.2 Τα χαρακτηριστικά της D2RQ πλατφόρμας

Η πλατφόρμα D2RQ συνίσταται από τα εξής:

- Τη D2RQ γλώσσα χαρτογράφησης (mapping language), η οποία είναι μία γλώσσα χαρτογράφησης για την περιγραφή των σχέσεων μεταξύ μίας οντολογίας και ενός σχεσιακού μοντέλου δεδομένων.
- Τη D2RQ Engine, η οποία είναι ένα plug-in για το Jena Semantic Web toolkit, το οποίο χρησιμοποιεί τα αρχεία-χάρτες για να ξαναγράψει τα SPARQL ερωτήματα σε SQL ερωτήματα ώστε να εκτελεστούν στη βάση και επιστρέφει τα αποτελέσματα από τα ερωτήματα στα ανώτερα επίπεδα των frameworks.
- Το D2R Server, ο οποίος είναι ένας HTTP Server που παρέχει μία Linked Data αναπαράσταση των δεδομένων, μία HTML αναπαράσταση για debugging και ένα SPARQL Protocol endpoint για να εκτελούμε ερωτήματα στη βάση[4].



Εικόνα 7.1: Η αρχιτεκτονική του D2RQ

7.3 D2R Server

Με το D2R Server μπορούμε να δημοσιεύσουμε το περιεχόμενο μίας σχεσιακής βάσης δεδομένων στο Semantic Web, σαν Linked Data. Επειδή, όμως, στο Semantic Web τα δεδομένα μοντελοποιούνται και αναπαρίστανται σε μορφή RDF, ο D2R Server χρησιμοποιεί ένα τροποποιήσιμο αρχείο-χάρτη για να χαρτογραφήσει τα περιεχόμενα της βάσης σε αυτή τη μορφή και μας επιτρέπει να πλοηγηθούμε και να αναζητήσουμε τα RDF δεδομένα, εκτελώντας SPARQL ερωτήματα, που είναι οι δύο κύριοι τρόποι πρόσβασης του Semantic Web. Οι αιτήσεις, που προέρχονται από το Διαδίκτυο, ξαναγράφονται σε SQL ερωτήματα με την χρησιμοποίηση του αρχείου-χάρτη. Αυτή η επιτόπου (on-the-fly) μετάφραση, μας επιτρέπει να δημοσιεύσουμε βάσεις δεδομένων σε μορφή RDF χωρίς να χρειάζεται να αναπαράγουμε ξανά τα δεδομένα σε μορφή RDF και μετά να τα ανεβάσουμε σε ένα RDF triple store[2].

7.3.1 Χαρακτηριστικά

- Πλοήγηση στα δεδομένα της βάσης: καθώς προσφέρει μία απλή διεπαφή διαδικτύου που μας επιτρέπει να πλοηγηθούμε στα περιεχόμενα της βάσης, δίνοντας μία πιο φιλική προς τον χρήστη προβολή των RDF δεδομένων.
- Πλοηγήσιμα URIs: ακολουθώντας τις αρχές των Linked Data, ο D2R Server αναθέτει ένα URI για κάθε μία οντότητα που περιγράφεται στη βάση. Κάνοντας έτσι, αυτά τα URIs πλοηγήσιμα, καθώς μπορούμε να ανακτήσουμε μία RDF περιγραφή, απλά με την πρόσβαση στα URI της οντότητας.
- Διαπραγμάτευση περιεχομένων: ακολουθώντας τις καλύτερες πρακτικές, η διεπαφή διαδικτύου μαζί με τον πλοηγήσιμο RDF γράφο μοιράζονται τα ίδια URIs.
- SPARQL endpoint και explorer: Η SPARQL διεπαφή επιτρέπει στις εφαρμογές να εκτελούν ερωτήματα στη βάση χρησιμοποιώντας την SPARQL 1.1 γλώσσα, χρησιμοποιώντας το SPARQL πρωτόκολλο. Επίσης παρέχεται και ένας απλός SPARQL explorer.
- Λήψη των περιεχομένων BLOBs/CLOBs: Ο D2R Server, μπορεί να ρυθμιστεί, για να εξυπηρετήσει (serve) δεδομένα που είναι αποθηκευμένα σε BLOBs ή CLOBs. Κάποιες βάσεις, μπορεί να περιέχουν ολόκληρα αρχεία, όπως PDFs ή PNGs, έτσι ο D2r Server μπορεί να κάνει αυτά τα περιεχόμενα ικανά προς λήψη.

- Serving the Vocabulary: αν προστεθούν νέες κλάσεις και ιδιότητες στον D2R για ανάπτυξη, τότε ο server μπορεί να κάνει τα URIs τους πλοηγήσιμα και επιτρέπει τη διαμόρφωση των επικετών τους, των σχολίων τους και των επιπλέον ιδιοτήτων τους.
- Δημοσίευση μεταδεδομένων: τα μεταδεδομένα όπως η αδειοδότηση και οι πληροφορίες προέλευσης μπορούν να συνδεθούν σε κάθε RDF έγγραφο και σελίδα του διαδικτύου που δημοσιεύθηκε από τον D2R Server[2].

7.3.2 Εντολές

Οι εντολές που μπορούμε να χρησιμοποιήσουμε με το D2R Server είναι οι εξής:

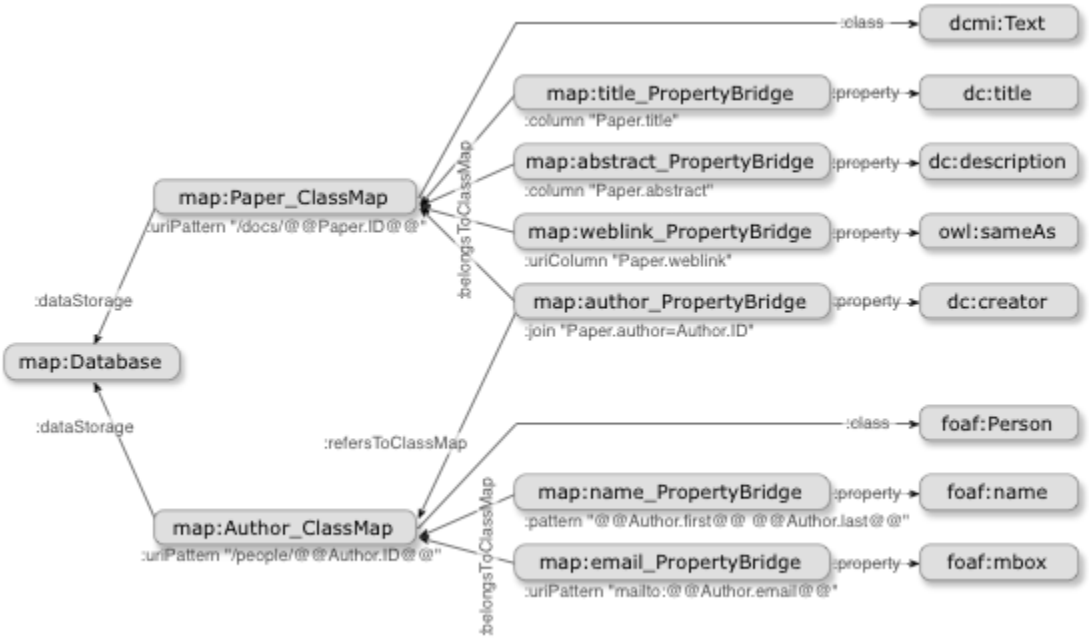
- Η εντολή «generate-mapping -o mapping.ttl -u user -p password jdbc:mysql:///mydatabase», με την οποία δημιουργούμε το αρχείο-χάρτη. Με το «-o mapping.ttl» δίνουμε το όνομα του αρχείου που θα δημιουργηθεί. Μετά δίνουμε το username και password καθώς και το URL που βρίσκεται η βάση.
- Η εντολή «d2r-server -p 8080 C:\...\ mapping.ttl», με την οποία ξεκινάει ο D2R Server. Όπου δίνουμε σαν παραμέτρους σε ποία θύρα θα «ακούει» και το που βρίσκεται το αρχείο-χάρτης που έχουμε.
- Η εντολή «dump-rdf C:\...\mapping.ttl > dump.nt», με την οποία δημιουργούμε ένα αρχείο dump της βάσης δίνοντας το URL της βάσης. Μας δίνει τη δυνατότητα να το δημιουργήσουμε σε διάφορες μορφές.
- Η εντολή «d2r-query mapping.ttl "select * {?s ?p ?o} limit 10"», με την οποία μπορούμε να εκτελέσουμε ένα SPARQL ερώτημα από το command line.
- Η εντολή «d2r-query mapping.ttl @query.sparql», με την οποία φορτώνουμε ένα SPARQL ερώτημα από ένα αρχείο για να εκτελεστεί.

7.4 Η D2RQ γλώσσα χαρτογράφησης

Η D2RQ γλώσσα χαρτογράφησης είναι μία δηλωτική γλώσσα για την περιγραφή των σχέσεων μεταξύ ενός σχήματος μίας σχεσιακής βάσης δεδομένων και των RDF λεξιλογίων και OWL οντολογιών. Ο ίδιος ο D2RQ χάρτης είναι ένα RDF έγγραφο γραμμένο σε μορφή Turtle. Η χαρτογράφηση εκφράζεται χρησιμοποιώντας όρους από το namespace της D2RQ: «http://www.wiwiss.fu-

berlin.de/suhl/bizer/D2RQ/0.1#». Η χαρτογράφηση ορίζει έναν εικονικό RDF γράφο που περιέχει πληροφορίες από τη βάση δεδομένων. Αυτός ο τρόπος είναι παρόμοιος με την ιδέα των SQL όψεων, με τη διαφορά ότι εδώ η εικονική δομή δεδομένων, είναι ένας RDF γράφος αντί για έναν εικονικό σχεσιακό πίνακα.

Οι D2RQ χαρτογραφήσεις μπορούν να γραφτούν από την αρχή με το χέρι σε έναν απλό text editor, αλλά συνήθως είναι πιο γρήγορο να αρχίζουμε με το «generate-mapping» εργαλείο που κατασκευάζει τον σκελετό, δηλαδή το default mapping, από το σχήμα της βάσης δεδομένων. Φυσικά αυτόν τον χάρτη που κατασκευάζει η D2RQ μπορούμε να τον τροποποιήσουμε.



Εικόνα 7.2: Δομή ενός παραδείγματος D2RQ χάρτη

Η βάση δεδομένων, είναι χαρτογραφημένη σε RDF όρους, χρησιμοποιώντας τα «d2rq:ClassMaps» και «d2rq:PropertyBridges». Τα πιο σημαντικά αντικείμενα στον χάρτη είναι τα στοιχεία «classmaps». Ένα «classmap» αναπαριστά μία κλάση ή μία ομάδα ίδιων κλάσεων της οντολογίας. Το «classmap» καθορίζει πως τα URIs ή οι ανώνυμοι κόμβοι ή κενοί κόμβοι θα παράγονται για τα στιγμιότυπα της κλάσης. Επίσης, έχει και ένα σύνολο από «propertybridges», τα οποία καθορίζουν πως θα δημιουργηθούν οι ιδιότητες ενός στιγμιότυπου[5].

7.4.1 Ένα απλό παράδειγμα D2RQ mapping

Έστω ότι έχουμε τον εξής D2RQ χάρτη από μία σχεσιακή βάση δεδομένων:

```
@prefix map: <#> .
@prefix db: <> .
@prefix vocab: <vocab/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d2rq: <http://www.wiwiss.fu-
berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix jdbc: <http://d2rq.org/terms/jdbc/> .

map:database a d2rq:Database;
      d2rq:jdbcDriver "com.mysql.jdbc.Driver";
      d2rq:jdbcDSN "jdbc:mysql:/// mydatabase ";
      d2rq:username "root";
      d2rq:password "password";
      jdbc:autoReconnect "true";
      jdbc:zeroDateTimeBehavior "convertToNull";
      .

# Table persons
map:persons a d2rq:ClassMap;
      d2rq:dataStorage map:database;
      d2rq:uriPattern "persons/@@persons.PerID@@";
      d2rq:class vocab:persons;
      d2rq:classDefinitionLabel "persons";
      .
map:persons_Name a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:persons;
      d2rq:property vocab:persons_Name;
      d2rq:propertyDefinitionLabel "persons Name";
      d2rq:column "persons.Name";
      .
```

Στην αρχή δηλώνουμε τα prefixes που θα χρησιμοποιηθούν στο αρχείο-χάρτη της βάσης δεδομένων. Η D2RQ στο default αρχείο-χάρτη χρησιμοποιεί συγκεκριμένα prefixes όπου μερικά είναι δικά του λεξιλόγια. Φυσικά μπορούμε να προσθέσουμε και

άλλα λεξιλόγια ανάλογα με τις ανάγκες μας, καθώς και να αφαιρέσουμε κάποια λεξιλόγια του D2RQ και να τα αντικαταστήσουμε με γνωστά λεξιλόγια που είναι ευρέως αποδεκτά.

Μετά με το «d2rq:Database» ορίζουμε μία σύνδεση με μία τοπική ή απομακρυσμένη σχεσιακή βάση δεδομένων. Με το «d2rq:jdbcDriver» ορίζουμε την JDBC driver κλάση της βάσης και με το «d2rq:jdbcDSN» ορίζουμε το URL της βάσης δεδομένων. Μετά δίνουμε το «username» και το «password» για να μπορέσουμε να συνδεθούμε στη βάση δεδομένων. Μετά αν θέλουμε μπορούμε να δώσουμε κάποιες επιπλέον επιλογές για την συμπεριφορά της βάσης δεδομένων.

Με το «d2rq:ClassMap» ορίζουμε κλάση, από την οποία θα παραχθούν πολλοί πόροι. Στο «d2rq:dataStorage» αναφέρεται ένα «d2rq:Database», στο οποίο είναι αποθηκευμένο το στιγμιότυπο της βάσης δεδομένων. Με το «d2rq:uriPattern» καθορίζεται το URI πρότυπο, με το οποίο θα προσδιορίζονται τα στιγμιότυπα αυτής της class map. Το «d2rq:class» ορίζει μία RDFS ή OWL κλάση, από την οποία όλοι οι πόροι που θα παραχθούν από αυτό το «ClassMap» θα είναι στιγμιότυπα αυτής της κλάσης. Με το «d2rq:classDefinitionLabel» καθορίζεται ένα label που θα χρησιμοποιείται σαν «rdfs:label» με όλους τους συσχετιζόμενους ορισμούς κλάσης.

Τέλος, με το «d2rq:PropertyBridge» συνδέουμε μία στήλη της βάσης δεδομένων με μία RDF ιδιότητα. Το «d2rq:belongsToClassMap» καθορίζει σε ποιο «d2rq:ClassMap» ανήκει αυτό το property bridge. Με το «d2rq:property» συνδέεται το «ClassMap» με ένα αντικείμενο ή με ένα Literal και προσδιορίζεται και η σχέση μεταξύ τους, χρησιμοποιώντας κάποιο λεξιλόγιο. Στο «d2rq:propertyDefinitionLabel» καθορίζεται ένα label που θα χρησιμοποιείται σαν «rdfs:label» με όλους τους συσχετιζόμενους ορισμούς ιδιοτήτων. Και το «d2rq:column» χρησιμοποιείται για Literal τιμές, όπου δίνουμε και την στήλη της βάσης δεδομένων από όπου θα παίρνει τις τιμές.

Φυσικά μπορούμε να τροποποιήσουμε αυτό το αρχείο-χάρτη ανάλογα με τις ανάγκες μας. Π.χ. μπορεί να γίνει έτσι:

```
@prefix map: <#> .
@prefix db: <> .
@prefix vocab: <vocab/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d2rq: <http://www.wiwiss.fu-
berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix jdbc: <http://d2rq.org/terms/jdbc/> .
```

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


map:database a d2rq:Database;
      d2rq:jdbcDriver "com.mysql.jdbc.Driver";
      d2rq:jdbcDSN "jdbc:mysql:///mydatabase";
      d2rq:username "root";
      d2rq:password "password";
      jdbc:autoReconnect "true";
      jdbc:zeroDateTimeBehavior "convertToNull";
      .

# Table persons
map:persons a d2rq:ClassMap;
      d2rq:dataStorage map:database;
      d2rq:uriPattern "persons/@@persons.PerID@@";
      d2rq:class foaf:Person;
      d2rq:classDefinitionLabel "persons";
      .
map:persons_Name a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:persons;
      d2rq:property foaf:name;
      d2rq:propertyDefinitionLabel "persons Name";
      d2rq:column "persons.Name";
      .
```

Προσθέσαμε το λεξιλόγιο «foaf» και το χρησιμοποιούμε, έτσι ώστε τα στιγμιότυπα που θα δημιουργηθούν από την κλάση να είναι τύπου «foaf:Person» και να έχουν μία ιδιότητα τύπου «foaf:name» που θα περιέχει το όνομα.

7.5 ΕΠΙΛΟΓΟΣ

Με τη D2RQ πλατφόρμα μπορούμε να δημοσιεύσουμε τα δεδομένα μίας σχεσιακής βάσης δεδομένων σε μορφή RDF και να εκτελέσουμε ερωτήματα SPARQL στη βάση δεδομένων. Οι βάσεις δεδομένων που υποστηρίζει η πλατφόρμα D2RQ είναι η Oracle, η MySQL, η PostgreSQL, η SQL Server, η HSQLDB και η InterbaseFirebird. Επίσης, μπορούμε να συνδέσουμε τη D2RQ με μία πηγή δεδομένων όπως η MS Access χρησιμοποιώντας μία ODBC-JDBC bridge, όμως λειτουργεί με κάποιους περιορισμούς. Ακόμα, μπορεί να συνδεθεί και με άλλες βάσεις δεδομένων, αλλά δεν είναι σίγουρο όμως ότι θα λειτουργήσει. Η D2RQ από εξορισμού αλληλεπιδρά με τις βάσεις δεδομένων χρησιμοποιώντας το SQL-92 standard.

ΚΕΦΑΛΑΙΟ 8

OPENLINK VIRTUOSO

8.1 ΕΙΣΑΓΩΓΗ

Το Virtuoso Universal Server είναι ένα middleware και ένα υβρίδιο βάσης δεδομένων, το οποίο συνδυάζει την λειτουργικότητα μίας RDBMS, ORDBMS, virtual database, RDF, XML, free-text, Web application server και file server λειτουργικότητα σε ένα σύστημα. Το Virtuoso έχει έναν «universal server» ο οποίος επιτρέπει μία πολυνηματική διεργασία διακομιστή που υλοποιεί πολλά πρωτόκολλα. Το Virtuoso έχει αναπτυχθεί από την OpenLink Software. Στο κεφάλαιο αυτό θα περιγραφεί η open-source έκδοση του Virtuoso.

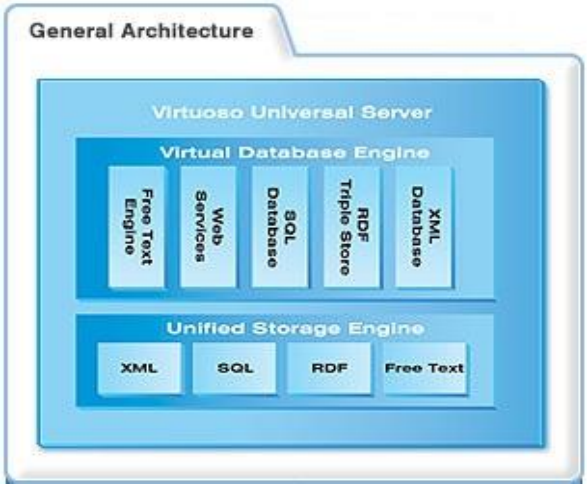
8.2 Virtuoso

Στην ουσία, το Virtuoso είναι μία αντικειμενο-σχεσιακή (object-relational) SQL βάση δεδομένων υψηλής απόδοσης. Το Virtuoso έχει ένα ενσωματωμένο web server, ο οποίος μπορεί να προσφέρει δυναμικές ιστοσελίδες διαδικτύου γραμμένες στην γλώσσα διαδικτύου του Virtuoso (VSP), είτε σε PHP, ASP.net και άλλες. Ο web server του παρέχει SOAP και REST πρόσβαση στις stored procedures του Virtuoso, υποστηρίζοντας ένα ευρύ σύνολο από WS πρωτοκόλλων όπως το WS-Security, το WS-Reliable Messaging και άλλα.

Επίσης, το Virtuoso έχει ένα ενσωματωμένο WebDAV repository, το οποίο μπορεί να φιλοξενήσει στατικό και δυναμικό περιεχόμενο ιστοσελίδων. Το OpenLink Virtuoso υποστηρίζει ενσωματωμένη SPARQL σε SQL για την εκτέλεση ερωτημάτων σε RDF δεδομένα που είναι αποθηκευμένα στη βάση δεδομένων του Virtuoso.

Οπότε, η μοναδική αρχιτεκτονική του υβριδικού server του Virtuoso, δίνει τη δυνατότητα να προσφέρει παραδοσιακά διακριτές λειτουργίες server σε ένα ενιαίο προϊόν που καλύπτει τους εξής τομείς:

- Relational Data Management
- RDF Data Management
- XML Data Management
- Free Text Content Management & Full Text Indexing
- Document Web Server
- Linked Data Server
- Web Application Server
- Web Services Deployment (SOAP ή Rest)



Εικόνα 8.1: Αρχιτεκτονική του Virtuoso

8.3 Οι εκδόσεις του Virtuoso

Το Virtuoso έχει δύο εκδόσεις, την «Open Source» και την «Commercial». Η διαφορές τους είναι ότι η Open Source έκδοση δεν περιλαμβάνει τις εξής λειτουργικότητες που έχει η Commercial έκδοση:

- Την Virtual Database Engine
- Την Data Replication Functionality

Η virtual database στην πιο απλή μορφή της προσφέρει την ικανότητα της αναζήτησης σε πολλές βάσεις δεδομένων με ένα απλό query. Δηλαδή, στην Open Source έκδοση δεν μπορούμε να έχουμε πρόσβαση σε άλλες βάσεις δεδομένων, αλλά μόνο να δουλεύουμε με την ενσωματωμένη βάση δεδομένων του Virtuoso. Επίσης, στην Open Source έκδοση δεν έχουμε τη δυνατότητα να δημιουργούμε και να κρατάμε Snapshots της βάσης.

8.4 R2RML

Το Virtuoso υποστηρίζει την R2RML γλώσσα. Η R2RML είναι μία γλώσσα με την οποία μπορούμε να εκφράσουμε προσαρμοσμένες χαρτογραφήσεις από σχεσιακές βάσεις δεδομένων σε RDF datasets. Αυτές οι χαρτογραφήσεις παρέχουν την ικανότητα να προβάλλουμε τα υπάρχοντα σχεσιακά δεδομένα στο μοντέλο δεδομένων RDF, που εκφράζεται σε μία δομή και σε λεξιλόγιο που επιλέγει ο προγραμματιστής. Οι ίδιες οι R2RML χαρτογραφήσεις είναι εκφρασμένες σαν RDF γράφοι, γραμμένες σε το συντακτικό Turtle. Π.χ. έστω ότι έχουμε τον εξής πίνακα και τα εξής prefixes:

EXAMPLE			
ID (integer)	**Name (varchar(100))**	**Mail (varchar(100))**	**Homepage (varchar(100))**
123	**George Smith**	**george@gmail.com**	**http://example.org/George**

- rr: http://www.w3.org/ns/r2rml#
- foaf: http://xmlns.com/foaf/0.1/

- rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

Έστω ότι θέλουμε να παράγουμε τις εξής RDF τριπλέτες:

```
<http://data.example.com/staffid/123>  rdf:type  foaf:Person .

<http://data.example.com/staffid/123>  foaf:name  “George Smith” .

<http://data.example.com/staffid/123>  foaf:mbox  <mailto:george@gmail.com> .

<http://data.example.com/staffid/123>  foaf:homepage  <http://example.org/George>
.
```

Η χαρτογράφηση σε R2RML των τριπλετών που θέλουμε είναι:

```
@prefix  rr: <http://www.w3.org/ns/r2rml#>

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

@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

<#TriplesMap1>

        rr:logicalTable [ rr:tableName “EXAMPLE” ];

        rr:subjectMap [

                rr:template “http://data.example.com/staffid/{ID}”;

                rr:class foaf:Person;

        ];

        rr:predicateObjectMap [

                rr:predicate foaf:name;

                rr:objectMap [ rr:column “Name” ];

        ];

        rr:predicateObjectMap [

                rr:predicate foaf:mbox;

                rr:objectMap [ rr:column “Mail” ];
```

```
        ];
        rr:predicateObjectMap [
                rr:predicate foaf:homepage;
                rr:objectMap [ rr:column “Homepage” ];
        ];
].
```

8.5 ΕΠΙΛΟΓΟΣ

Το Virtuoso, στην ουσία είναι μία αντικειμενο-σχεσιακή βάση δεδομένων, που όμως μας προσφέρει την λειτουργικότητα, όπως και το D2RQ, να δημοσιεύουμε RDF δεδομένα. Επίσης, μας δίνει τη δυνατότητα να πλοηγηθούμε ή να εκτελέσουμε SPARQL ερωτήματα στα RDF δεδομένα μας, εγκαθιστώντας κάποια επιπλέον πακέτα που χρειάζεται για ορισμένα πράγματα.

ΚΕΦΑΛΑΙΟ 9

ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

9.1 ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα περιγραφεί η βάση δεδομένων που χρησιμοποιήθηκε στα επόμενα κεφάλαια (10, 11, 12) με τα εργαλεία D2R Server και Virtuoso. Επίσης, θα αναφερθούν και οι τροποποιήσεις που έγιναν ώστε να μπορεί να χρησιμοποιηθεί από τα εργαλεία D2R Server και Virtuoso. Ο στόχος ήταν να μετατρέψουμε και να τροποποιήσουμε κατάλληλα τη βάση ώστε να καταστεί δυνατή η χρησιμοποίηση και δημοσίευσή της μέσω των εργαλείων D2R Server και Virtuoso.

9.2 Η βάση δεδομένων

Τα δεδομένα που χρησιμοποιήθηκαν στη βάση, αρχικά ήταν σε μορφή IXF (IBM Exchange Format), οπότε στην DB2 δημιουργήθηκαν οι πίνακες και ύστερα περάστηκαν τα δεδομένα. Η βάση έχει τους εξής πίνακες:

CUSTOMER (ID, NAME)

ARTICLES (ITEM_ID, ARTICLE_CATEGORY)

ARTICLE_CATEGORIES (ID, DESC)

POS_DATA (CUSTOMERID, TRANSID, ITEMID, DATE, TIME, QUANTITY, PRICE, PROMOTION)

Ο πίνακας «CUSTOMER» έχει 3.836 εγγραφές, ο πίνακας «ARTICLES» έχει 2.042 εγγραφές, ο πίνακας «ARTICLE_CATEGORIES» έχει 76 εγγραφές και ο πίνακας «POS_DATA» έχει 247.535 εγγραφές. Αφού δημιουργήθηκαν και περάστηκαν τα δεδομένα στη βάση της DB2, στη συνέχεια εξήχθησαν σε μορφή CSV (comma-separated values), ώστε να περαστούν στη MySQL. Στον πίνακα «POS_DATA» προστέθηκε ακόμα μία στήλη ως κύριο κλειδί, καθώς ο D2R Server πρέπει να προσδιορίσει με ένα URI τα στιγμιότυπα που θα δημιουργηθούν από τον πίνακα και για να γίνει αυτό χρειάζεται να έχει κύριο κλειδί ο πίνακας. Οπότε ο πίνακας «POS_DATA» τροποποιήθηκε:

POS_DATA (ROWID, CUSTID, TRANSID, ITEMID, DATE, TIME, QUANTITY, PRICE, PROMOTION)

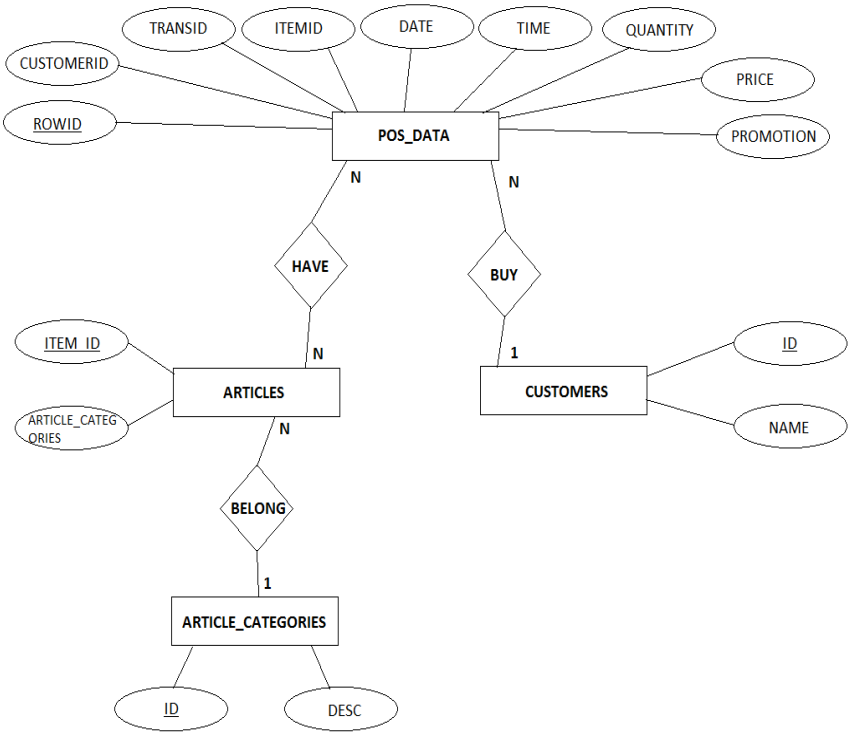
Η νέα στήλη «ROWID» προστέθηκε καθώς κανένας συνδυασμός από τις άλλες στήλες δεν μπορούσε να χρησιμοποιηθεί σαν κύριο κλειδί. Αφού δημιουργήθηκαν οι πίνακες και περάστηκαν τα δεδομένα στη MySQL, μερικοί πίνακες τροποποιήθηκαν και προστέθηκαν σε αυτούς ξένα κλειδιά. Τα ξένα κλειδιά είναι τα εξής:

ARTICLES (ARTICLE_CATEGORY) => **ARTICLE_CATEGORIES** (ID)

POS_DATA (ITEMID) => **ARTICLES** (ITEM_ID)

POS_DATA (CUSTOMERID) => **CUSTOMER** (ID)

Το διάγραμμα ER της βάσης είναι το εξής:



Εικόνα 9.1: Διάγραμμα ER της βάσης δεδομένων

9.3 ΕΠΙΛΟΓΟΣ

Η βάση δεδομένων που περιγράφθηκε στο κεφάλαιο αυτό τροποποιήθηκε και επεξεργάστηκε, με σκοπό να μπορεί να χρησιμοποιηθεί χωρίς προβλήματα από τα εργαλεία D2R Server και Virtuoso. Η βάση στη MySQL έχει σημασία μόνο για το D2R Server, καθώς μόνο αυτός συνδέεται με τη βάση στη MySQL και εκτελεί ερωτήματα στα δεδομένα της. Στο Virtuoso η βάση μεταφέρθηκε με αρχεία CSV καθώς το Virtuoso έχει δικό του DBMS.

ΚΕΦΑΛΑΙΟ 10

ΔΗΜΟΣΙΕΥΣΗ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΟ D2R SERVER

10.1 ΕΙΣΑΓΩΓΗ

Στόχος του κεφαλαίου αυτού είναι να παρουσιάσει τον τρόπο με τον οποίο δημοσιεύτηκε η βάση δεδομένων, που περιγράφθηκε στο προηγούμενο κεφάλαιο (9), με τη χρήση του D2R Server. Επίσης, θα παρουσιαστεί και ο D2R Server στη χρήση και θα εκτελεστούν ερωτήματα στη βάση, τα οποία θα παρουσιαστούν μαζί με τα αποτελέσματα που επιστράφηκαν.

10.2 Παράδειγμα χρήσης του D2R Server

Με τη βάση δεδομένων να βρίσκεται στη MySQL και να έχει τροποποιηθεί κατάλληλα ώστε να μπορεί να χρησιμοποιηθεί από το D2R Server, δημιουργούμε το αρχείο-χάρτη της βάσης με την εξής εντολή:

```
generate-mapping -o retailmap.ttl -u root -p password jdbc:mysql:///retail
```

generate-mapping

Είναι η εντολή με την οποία ο D2R Server δημιουργεί το αρχείο-χάρτη, στην οποία περνάμε και ορισμένες παραμέτρους.

-o retailmap.ttl

Με αυτή την παράμετρο δίνουμε το όνομα του αρχείου-χάρτη που θα δημιουργηθεί, το οποίο είναι σε μορφή Turtle.

-u root

Με αυτή την παράμετρο δίνουμε το όνομα του χρήστη της βάσης.

-p password

Με αυτή την παράμετρο δίνουμε τον κωδικό του χρήστη της βάσης.

jdbc:mysql:///retail

Με αυτή την παράμετρο δίνουμε το URL της JDBC σύνδεσης με τη βάση, όπου το «retail» είναι το όνομα της βάσης.

Το αρχείο-χάρτης που θα δημιουργηθεί θα περιέχει τα default λεξιλόγια που χρησιμοποιεί ο D2R Server, καθώς και το λεξιλόγιο του D2R Server. Το αρχείο-χάρτης τροποποιήθηκε ώστε να περιέχει μερικά από τα γνωστά λεξιλόγια, τα οποία χρησιμοποιήθηκαν στα ερωτήματα SPARQL που έγιναν στη βάση. Ένα μέρος του αρχείου-χάρτη που δημιουργήθηκε και τροποποιήθηκε ώστε να περιέχει μερικά γνωστά λεξιλόγια και περιγράφει τον πίνακα «CUSTOMER»:

```
# Table customer

map:customer a d2rq:ClassMap;

        d2rq:dataStorage map:database;
```

```
        d2rq:uriPattern "customer/@@customer.id@@";
        d2rq:class foaf:Person;
        .
map:customer__label a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:customer;
        d2rq:property rdfs:label;
        d2rq:pattern "customer #@@customer.id@@";
        .
map:customer_id a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:customer;
        d2rq:property dc:identifier;
        d2rq:column "customer.id";
        d2rq:datatype xsd:integer;
        .
map:customer_name a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:customer;
        d2rq:property foaf:name;
        d2rq:column "customer.name";
        .
```

Το «d2rq:ClassMap» αναπαριστά μία κλάση όμοιων πραγμάτων. Το «class map» ορίζει με ποιον τρόπο θα προσδιορίζονται τα στιγμιότυπα αυτής της κλάσης. Συνδέεται με μία βάση «d2rq:dataStorage map:database», η οποία περιγράφεται στο αρχείο-χάρτη, και έχει ένα σύνολο από «d2rq:PropertyBridge». Με το «d2rq:uriPattern "customer/@@customer.id@@"» ορίζουμε το URI πρότυπο με το οποίο θα προσδιορίζονται τα στιγμιότυπα του «class map», όπου εδώ είναι η στήλη «ID» του πίνακα «CUSTOMER». Με το «d2rq:class foaf:Person» ορίζουμε ότι τα στιγμιότυπα που θα δημιουργηθούν θα είναι της κλάσης «foaf:Person».

Το «d2rq:PropertyBridge» συνδέει μία στήλη της βάσης με κάποια RDF ιδιότητα. Με το «d2rq:belongsToClassMap map:customer» ορίζουμε ότι το «property bridge» ανήκει στο «class map» του customer. Με το «d2rq:property foaf:name» ορίζουμε ότι η RDF ιδιότητα που συνδέει το «class map» με το Literal που περιέχει κάποια τιμή από τον πίνακα θα είναι τύπου «foaf:name». Με το «d2rq:column "customer.name"» ορίζουμε ότι τα Literals θα παίρνουν τις τιμές τους από την στήλη «name» του πίνακα «CUSTOMER».

Αφού έχει δημιουργηθεί και τροποποιηθεί κατάλληλα το αρχείο-χάρτης, μπορούμε να ξεκινήσουμε το D2R Server με την εξής εντολή:

```
d2r-server -p 8080 C:\Users\User\Desktop\retailmap.ttl
```

d2r-server

Είναι η εντολή με την οποία ξεκινά ο D2R Server.

-p 8080

Με αυτή την παράμετρο δίνουμε την θύρα στην οποία θα «ακούει» ο D2R Server.

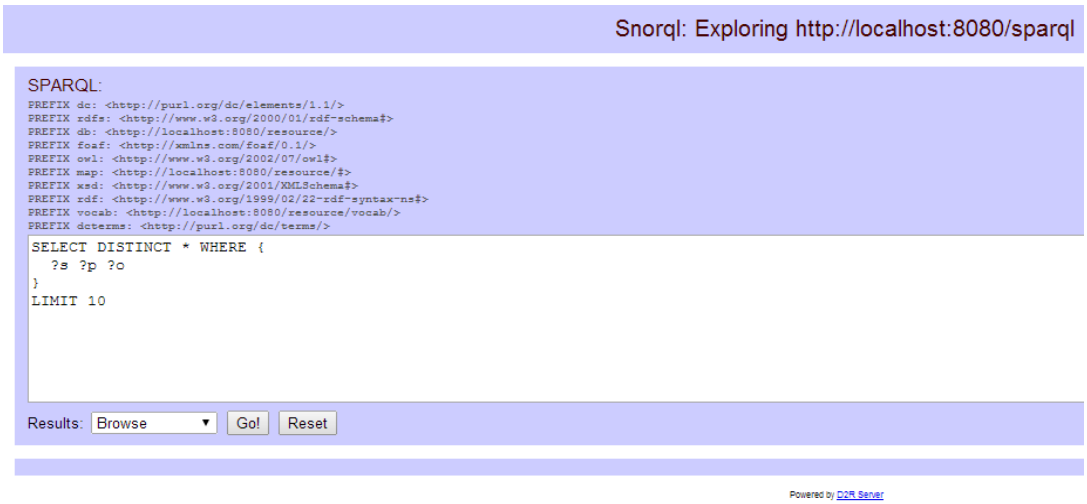
C:\Users\User\Desktop\retailmap.ttl

Με αυτή την παράμετρο δίνουμε τη διαδρομή που βρίσκεται το αρχείο-χάρτης, το οποίο χρησιμοποιείται από το D2R Server για να χαρτογραφηθεί η βάση δεδομένων στη MySQL.

Έπειτα, χρησιμοποιώντας έναν browser δίνουμε το URL «http://localhost:8080/».

Εικόνα 10.1: Αρχική σελίδα του D2R Server

Στην εικόνα 10.1 στην αρχή υπάρχουν οι πίνακες της βάσης στους οποίου μπορούμε να πλοηγηθούμε και να δούμε τα δεδομένα που περιέχουν. Ο D2R Server μας παρέχει και ένα SPARQL endpoint για να μπορούμε να εκτελούμε ερωτήματα SPARQL στη βάση. Στην αρχή αναφέρονται τα λεξιλόγια που μπορούν να χρησιμοποιηθούν στα SPARQL ερωτήματα, στην εικόνα 10.2.



Εικόνα 10.2: SPARQL endpoint του D2R Server

10.2.1 Εκτέλεση SPARQL ερωτημάτων στη βάση

1) Εμφάνιση των ονομάτων και των «id» δέκα πελατών.

```
SELECT ?name ?id WHERE { ?s foaf:name ?name;

                         dc:identifier ?id}LIMIT 10
```

SPARQL results:

name	id
"Bastide D"	9
"Ahles R"	133
"Avgoloupis SI"	382
"BARBATSI K"	1069
"Bauer H"	2249
"Bardou F"	2589
"BATMA A"	3518
"Bazoti FN"	5380
"Batalas N"	7118
"Athanassiou A"	7930

Εικόνα 10.3: Αποτελέσματα ονομάτων και «id» δέκα πελατών

2) Πόσες φορές αγοράστηκε το προϊόν «MILK».

```
select  ?desc (count(?itid) as ?count) where {

  ?rid vocab:pos_data_transid ?transid;

       vocab:pos_data_itemid ?itid.

  ?itid dc:identifier ?art;

       vocab:articles_article_category ?catid.

  ?catid dc:description ?desc.

FILTER(regex(?desc, "MILK", "i"))

}

group by ?desc
```

Εικόνα 10.3: Αποτελέσματα για το πόσες φορές αγοράστηκε το προϊόν «MILK»

10.3 ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο παρουσιάστηκε ο τρόπος με τον οποίο δημοσιεύθηκε η βάση που περιγράφθηκε στο κεφάλαιο εννέα με το εργαλείο D2R Server και εκτελέστηκαν μερικά SPARQL ερωτήματα στα δεδομένα της βάσης.

ΚΕΦΑΛΑΙΟ 11

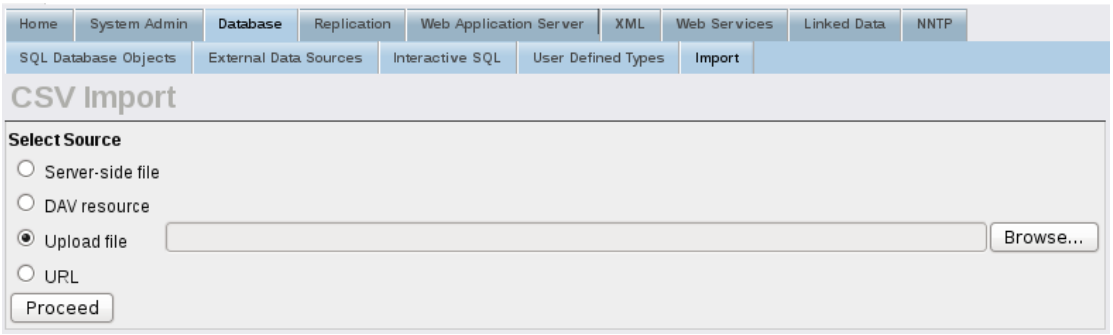
ΔΗΜΟΣΙΕΥΣΗ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΟ VIRTUOSO

11.1 ΕΙΣΑΓΩΓΗ

Στόχος του κεφαλαίου αυτού είναι να παρουσιάσει τον τρόπο με τον οποίο δημοσιεύτηκε η βάση δεδομένων, που περιγράφθηκε στο κεφάλαιο εννέα, με τη χρήση του Virtuoso. Επίσης, θα παρουσιαστεί και το Virtuoso στη χρήση και θα εκτελεστούν ερωτήματα στη βάση, τα οποία θα παρουσιαστούν μαζί με τα αποτελέσματα που επιστράφηκαν.

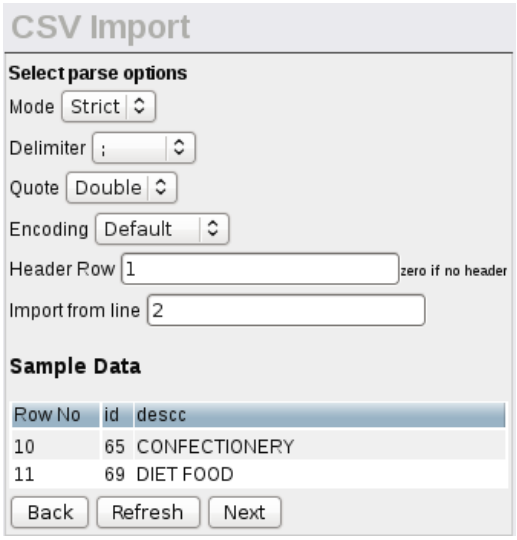
11.2 Μεταφορά της βάσης στο DBMS του Virtuoso

Η «open source» έκδοση του Virtuoso δε μπορεί να συνδεθεί με τη MySQL μέσω ενός ODBC ή JDBC driver, οπότε, η βάση δεδομένων που υπάρχει στη MySQL και περιγράφθηκε στο κεφάλαιο εννέα, μεταφέρθηκε στο DBMS που μας προσφέρει το Virtuoso με CSV αρχεία. Όταν τα αρχεία δεδομένων CSV της βάσης μεταφορτώνονται στο Virtuoso οι πίνακες δημιουργούνται αυτόματα από το Virtuoso, το οποίο προσθέτει μία νέα στήλη ως κύριο κλειδί.



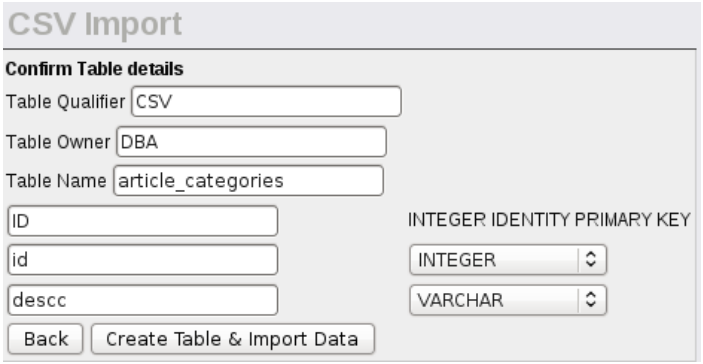
Εικόνα 11.1: Μεταφόρτωση αρχείων CSV στο Virtuoso

Αφού επιλέξουμε το αρχείο που θέλουμε να μεταφορτώσουμε στο Virtuoso, συνεχίζουμε πατώντας το κουμπί «Proceed», εικόνα11.1, και ρυθμίζουμε μερικές επιλογές, ώστε το Virtuoso να μπορεί να «διαβάσει» το CSV αρχείο, εικόνα 11.2. Επιλέγουμε τα δεδομένα με ποίο χαρακτήρα χωρίζονται, αν περικλείονται με μονά ή διπλά εισαγωγικά, αν υπάρχει «Header Row» και σε ποια γραμμή βρίσκεται και από ποία γραμμή να ξεκινήσει να μεταφορτώνει τα δεδομένα.



Εικόνα 11.2: Ρύθμιση επιλογών

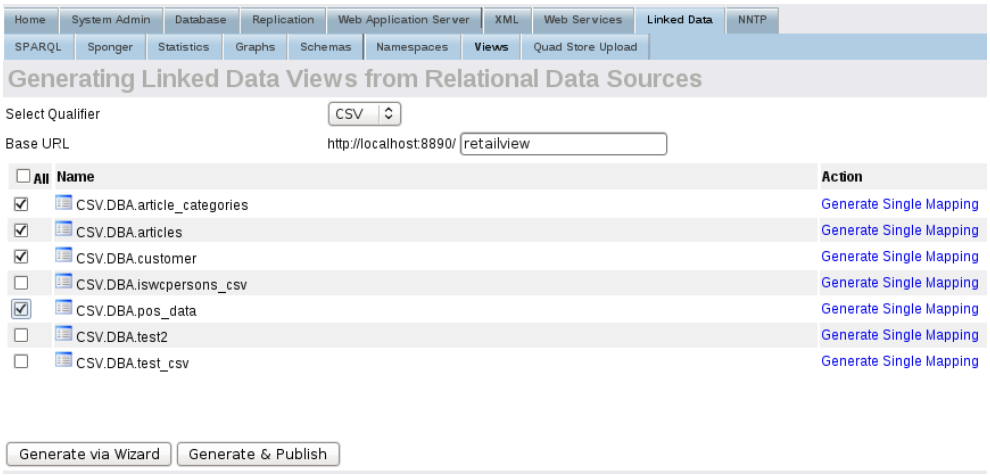
Όταν έχουν ρυθμιστεί και οι επιλογές ώστε το Virtuoso να μπορεί να «διαβάσει» το CSV αρχείο, συνεχίζουμε πατώντας το κουμπί «Next». Έπειτα διαλέγουμε και ρυθμίζουμε το όνομα του πίνακα που θα δημιουργηθεί, τον owner του πίνακα, το schema στον οποίο θα ανήκει ο πίνακας καθώς επίσης τα ονόματα των στηλών και το τι τύπου θα είναι οι στήλες του πίνακα, εικόνα 11.3. Αφού ρυθμιστούν και αυτές οι επιλογές δημιουργούμε τον πίνακα και εισάγουμε τα δεδομένα με το κουμπί «Create Table & Import Data». Όταν δημιουργηθεί ο πίνακας και έχουν εισαχθεί τα δεδομένα, μπορούμε να τροποποιήσουμε τον πίνακα και να αφαιρέσουμε την στήλη που πρόσθεσε το Virtuoso και να ορίσουμε εμείς την στήλη που θέλουμε ως κύριο κλειδί.



Εικόνα 11.3: Ρύθμιση πίνακα

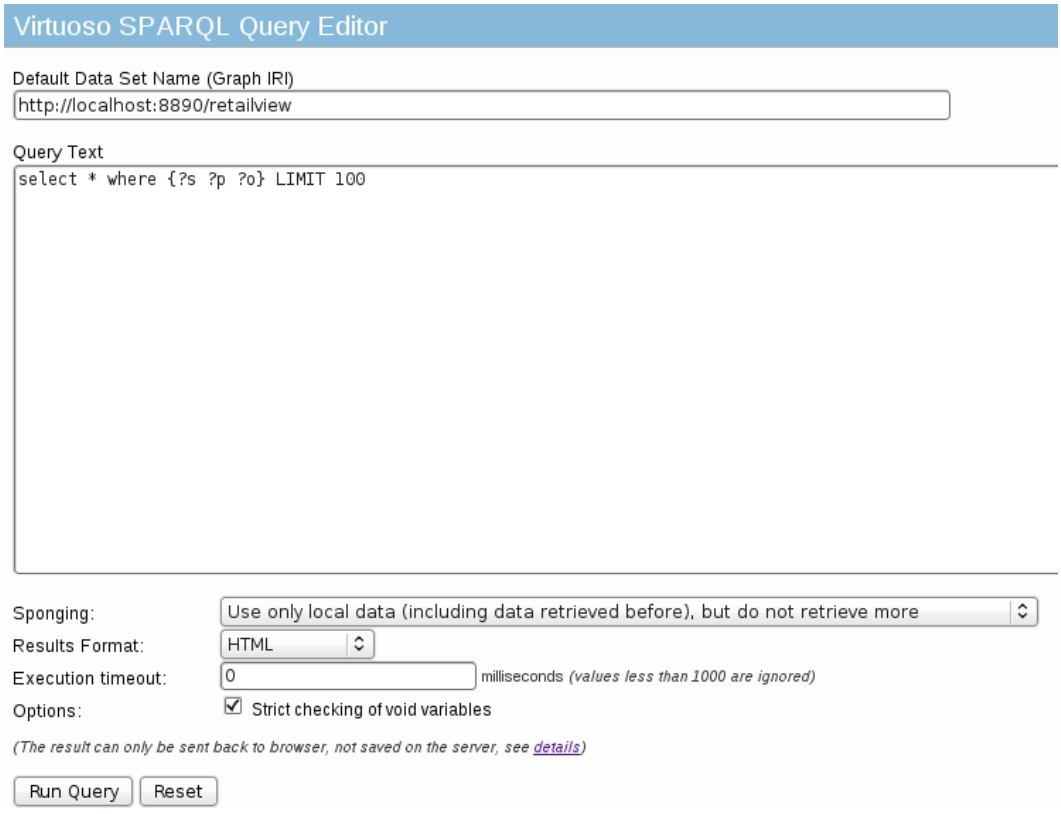
11.3 Δημοσίευση της βάσης στο Virtuoso με RDF Views

Το Virtuoso μας παρέχει τη δυνατότητα να δημοσιεύσουμε τη βάση που δημιουργήσαμε στο DBMS του. Αφού έχουμε μεταφέρει τη βάση δεδομένων στο Virtuoso και έχουμε κάνει τις τροποποιήσεις που θέλουμε, μπορούμε να δημοσιεύσουμε τη βάση με RDF Views. Επιλέγουμε το URL στο οποίο θέλουμε να δημοσιευθεί η βάση και τους πίνακες που θέλουμε να δημοσιεύσουμε από τη βάση και είτε επιλέγουμε το κουμπί «Generate via Wizard» για να ρυθμίσουμε κάποιες επιπλέον επιλογές, δηλαδή για το τι ακριβώς θα δημιουργήσει, είτε επιλέγουμε το κουμπί «Generate & Publish» όπου δημιουργεί τα RDF Views με τις default επιλογές, εικόνα 11.4.



Εικόνα 11.4: Επιλογή URL και πινάκων

Στο τέλος μας εμφανίζει τα αρχεία-χάρτες και οντολογίες που θα δημιουργήσει για να δημοσιεύσει τη βάση, τα οποία μπορούμε να τροποποιήσουμε και να προσθέσουμε λεξιλόγια κτλ. Αφού κάνουμε τις αλλαγές που θέλουμε τότε δημιουργούμε τα αρχεία-χάρτες και οντολογίες και μπορούμε μετά να πλοηγηθούμε στα δεδομένα με συνδέσμους ή να εκτελέσουμε SPARQL ερωτήματα στα δεδομένα από το SPARQL endpoint του Virtuoso, εικόνα 11.5.



Εικόνα 11.5: SPARQL endpoint του Virtuoso

Αν εκτελέσουμε το SPARQL ερώτημα ώστε να βρούμε πόσες φορές αγοράστηκε το προϊόν «MILK»:

```
select  ?desc (count(?transid) as ?count) where {

  ?rid <http://localhost:8890/schemas/testvocabview/transid> ?transid;

       <http://localhost:8890/schemas/testvocabview/itemid> ?itid.

  ?item dc:identifier ?itid;

            <http://localhost:8890/schemas/testvocabview/article_category> ?catid.

  ?cat  dc:identifier ?catid;

           dc:description ?desc.

FILTER(regex(?desc, "MILK", "i"))

}
```

Θα μας επιστραφεί σαν αποτέλεσμα:

desc	count
MILK	49139

Εικόνα 11.6: Αποτελέσματα για το πόσες φορές αγοράστηκε το προϊόν «MILK»

11.4 Δημοσίευση της βάσης στο Virtuoso με RDF αρχεία

Το Virtuoso μας δίνει τη δυνατότητα να δημοσιεύσουμε και RDF αρχεία. Οπότε μπορούμε με το D2R Server να δημιουργήσουμε το RDF dump αρχείο της βάσης, στη συνέχεια να το μεταφορτώσουμε στο Virtuoso και να το δημοσιεύσουμε ώστε να μπορούμε να εκτελούμε SPARQL ερωτήματα στα δεδομένα. Με το D2R Server δημιουργούμε το RDF dump της βάσης που έχουμε στη MySQL με την εξής εντολή:

```
dump-rdf C:\Users\User\Desktop\ retailmap.ttl > dumpretail.nt
```

dump-rdf

Είναι η εντολή με την οποία ο D2R Server δημιουργεί το RDF dump αρχείο της βάσης.

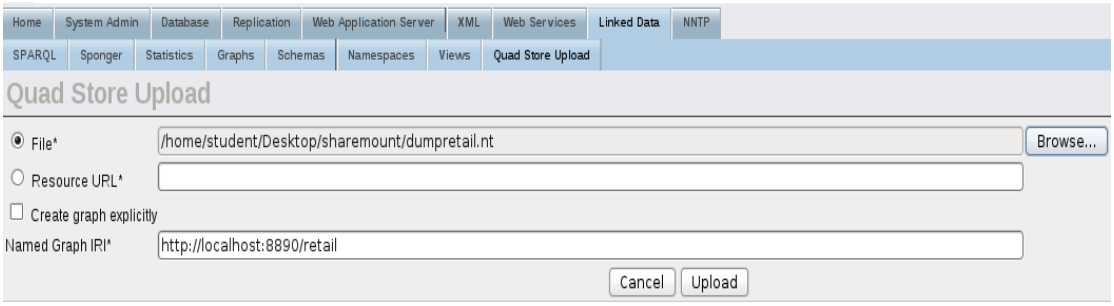
C:\Users\User\Desktop\retailmap.ttl

Με αυτή την παράμετρο δίνουμε τη διαδρομή που βρίσκεται το αρχείο-χάρτης, το οποίο χρησιμοποιείται από το D2R Server για να δημιουργήσει το RDF dump αρχείο.

dumpretail.nt

Με αυτή την παράμετρο δίνουμε το όνομα του RDF dump αρχείου που θα δημιουργηθεί, το οποίο είναι σε μορφή N-Triples η οποία είναι η default επιλογή.

Αφού έχει δημιουργηθεί το RDF dump αρχείο, το μεταφορτώνουμε στο Virtuoso. Όταν το μεταφορτώνουμε δίνουμε και το «Named Graph IRI», το οποίο θα περιέχει τα δεδομένα και στο οποίο θα μπορούμε να εκτελούμε SPARQL ερωτήματα, εικόνα 11.7.



Εικόνα 11.7: Φόρτωση RDF αρχείου στο Virtuoso

Κατά τη μεταφόρτωση του RDF αρχείου μπορεί να υπάρξουν προβλήματα λόγω της μνήμης που χρησιμοποιεί το Virtuoso ή του μεγέθους της μνήμης του υπολογιστή. Αυτά τα προβλήματα μπορούν να λυθούν με τους εξής τρόπους:

1. Τροποποιούμε το αρχείο «virtuoso.ini» και επιλέγουμε πόση μνήμη θα χρησιμοποιεί το Virtuoso. Η default επιλογή του Virtuoso χρησιμοποιεί την ελάχιστη δυνατή μνήμη με αποτέλεσμα να μην έχει καλές επιδόσεις το Virtuoso. Αν θέσουμε όλες τις επιλογές σε σχόλια τότε το Virtuoso θα χρησιμοποιεί όση ελεύθερη μνήμη έχει διαθέσιμη ο υπολογιστής, 11.7.

```
;; When running with large data sets, one should configure the Virtuoso
;; process to use between 2/3 to 3/5 of free system memory and to stripe
;; storage on all available disks.
;;
;; Uncomment next two lines if there is 2 GB system memory free
;       NumberOfBuffers          = 170000
;       MaxDirtyBuffers          = 130000
;; Uncomment next two lines if there is 4 GB system memory free
;       NumberOfBuffers          = 340000
;       MaxDirtyBuffers          = 250000
;; Uncomment next two lines if there is 8 GB system memory free
;       NumberOfBuffers          = 680000
;       MaxDirtyBuffers          = 500000
;; Uncomment next two lines if there is 16 GB system memory free
;       NumberOfBuffers          = 1360000
;       MaxDirtyBuffers          = 1000000
;; Uncomment next two lines if there is 32 GB system memory free
;       NumberOfBuffers          = 2720000
;       MaxDirtyBuffers          = 2000000
;; Uncomment next two lines if there is 48 GB system memory free
;       NumberOfBuffers          = 4000000
;       MaxDirtyBuffers          = 3000000
;; Uncomment next two lines if there is 64 GB system memory free
;       NumberOfBuffers          = 5450000
;       MaxDirtyBuffers          = 4000000
;;
;; Note the default settings will take very little memory
;; but will not result in very good performance
;;
;NumberOfBuffers         = 10000
;MaxDirtyBuffers         = 6000
```

Εικόνα 11.7: Επιλογή της μνήμης που θα χρησιμοποιεί το Virtuoso

2. Χωρίζουμε το αρχείο dump που έχει δημιουργήσει ο D2R Server σε μικρότερα αρχεία με την εντολή «split» των Linux. Έπειτα μπορούμε να μεταφορτώσουμε όλα τα μικρότερα αρχεία στο ίδιο «Named Graph IRI».

Αν εκτελέσουμε το SPARQL ερώτημα ώστε να βρούμε πόσες φορές αγοράστηκε το προϊόν «MILK»:

```
select  ?desc (count(?transid) as ?count) where {

  ?rid <http://localhost:2020/vocab/pos_data_transid> ?transid;

      <http://localhost:2020/vocab/pos_data_itemid> ?itid.

  ?itid dc:identifier ?art;

      <http://localhost:2020/vocab/articles_article_category> ?catid.

  ?catid dc:description ?desc.

FILTER(regex(?desc, "MILK", "i"))

}
```

Θα μας επιστραφεί σαν αποτέλεσμα:

desc	count
MILK	49139

Εικόνα 11.8: Αποτελέσματα για το πόσες φορές αγοράστηκε το προϊόν «MILK»

11.5 Σύνδεση στη βάση του Virtuoso με PHP

Μπορούμε να ενεργοποιήσουμε την PHP στο Virtuoso έτσι ώστε να έχουμε τη δυνατότητα να εκτελούμε PHP αρχεία μέσω του Virtuoso. Για να δέχεται, το Virtuoso, PHP πρέπει, προτού γίνει η εγκατάσταση του Virtuoso, να εγκαταστήσουμε τον iODBC οδηγό και την PHP με κάποιες επιπλέον παραμέτρους που μας δίνει η OpenLink Software. Αφού εγκαταστήσουμε τον iODBC και την PHP με τις επιπλέον παραμέτρους, εγκαθιστούμε το Virtuoso με την παράμετρο ενεργοποίησης της PHP. Έπειτα προσθέτουμε στο «odbcinst.ini» αρχείο τις εξής γραμμές:

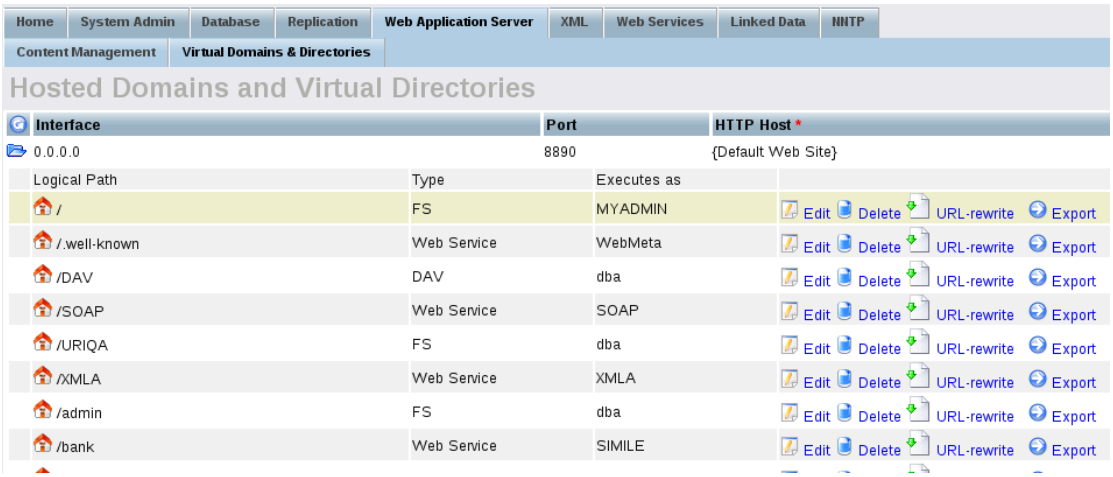
```
[virtuoso-odbc]
Driver = <prefix>/lib/virtodbc.so
```

και στο «odbc.ini» αρχείο τις εξής γραμμές:

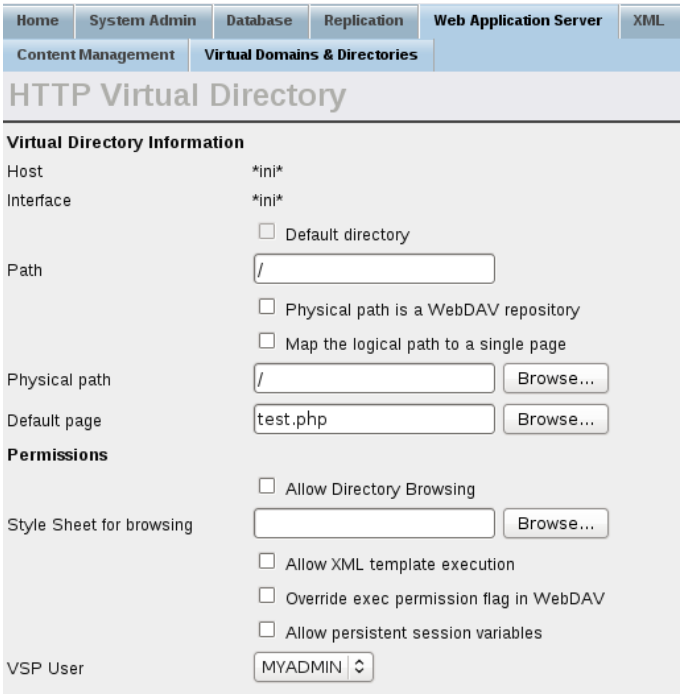
```
[ODBC Data Sources]
VOS = Virtuoso

[VOS]
Driver = virtuoso-odbc
Description = Virtuoso Open-Source Edition
ServerName = localhost:1111
```

Αν δεν υπάρχουν τα αρχεία «odbcinst.ini» και «odbc.ini» τότε τα δημιουργούμε εμείς. Στους πίνακες που έχουμε στη βάση του Virtuoso δίνουμε τα δικαιώματα στο χρήστη που θέλουμε, ώστε να μπορεί να εκτελεί SQL ερωτήματα στη βάση μέσω της PHP. Για να μπορεί να συνδεθεί ο χρήστης όμως πρέπει να του δοθεί και πρόσβαση στο φάκελο που είναι το PHP αρχείο, εικόνα 11.9 και 11.10:



Εικόνα 11.9: Δικαίωμα πρόσβασης στο φάκελο στο χρήστη MYADMIN



Εικόνα 11.10: Δικαίωμα πρόσβασης στο φάκελο στο χρήστη MYADMIN

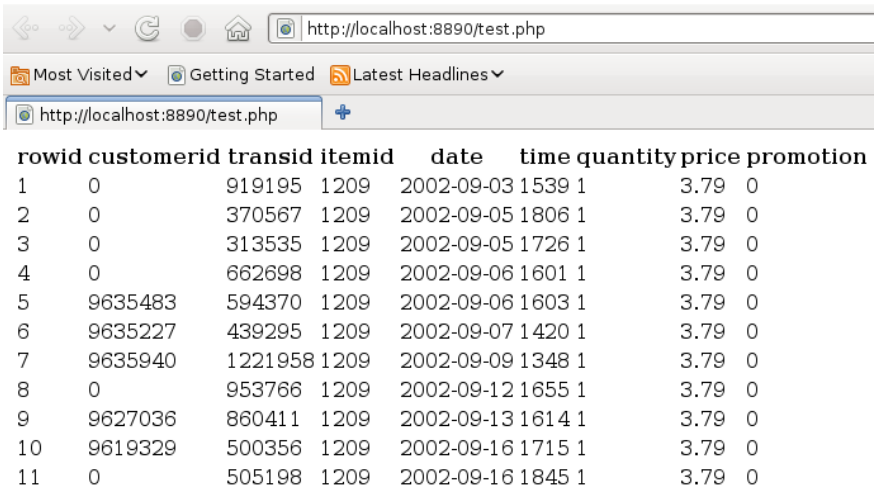
Αφού δοθούν τα δικαιώματα στον χρήστη, τότε το Virtuoso μας δίνει τη δυνατότητα να συνδεθούμε στη βάση με τη μέθοδο «__virt_internal_dsn()» η οποία επιστρέφει ένα ODBC SQLDriverConnect string του τύπου «DSN=use_dsn;UID=uid; PWD=pwd». Με το εξής PHP αρχείο θέλουμε να εμφανίσουμε τον πίνακα «POS_DATA»:

```
<?php
    $db = odbc_connect(__virt_internal_dsn(), null, null);
```

```
    if (!$db)
        error_log ('odbc_connect failed');


    $rs = odbc_exec ($db, 'select * from MYAPP.DBA.pos_data');
    odbc_result_all ($rs);
    odbc_close ($db);
?>
```

Θα μας επιστραφεί το εξής αποτέλεσμα:



rowid	customerid	transid	itemid	date	time	quantity	price	promotion
1	0	919195	1209	2002-09-03	1539	1	3.79	0
2	0	370567	1209	2002-09-05	1806	1	3.79	0
3	0	313535	1209	2002-09-05	1726	1	3.79	0
4	0	662698	1209	2002-09-06	1601	1	3.79	0
5	9635483	594370	1209	2002-09-06	1603	1	3.79	0
6	9635227	439295	1209	2002-09-07	1420	1	3.79	0
7	9635940	1221958	1209	2002-09-09	1348	1	3.79	0
8	0	953766	1209	2002-09-12	1655	1	3.79	0
9	9627036	860411	1209	2002-09-13	1614	1	3.79	0
10	9619329	500356	1209	2002-09-16	1715	1	3.79	0
11	0	505198	1209	2002-09-16	1845	1	3.79	0

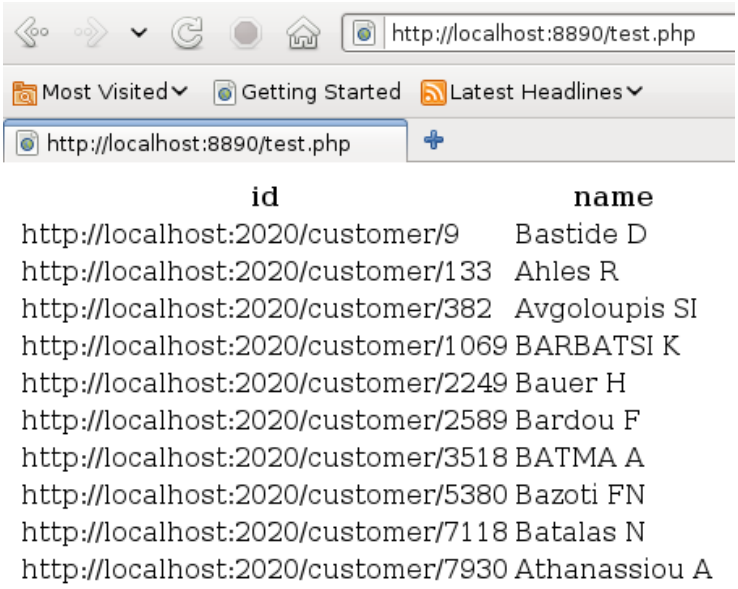
Εικόνα 11.11: Αποτελέσματα εκτέλεσης SQL ερωτήματος με PHP

Επίσης μέσω PHP μπορούμε να κάνουμε και SPARQL ερωτήματα σε κάποιον γράφο που έχουμε στο Virtuoso. Με το εξής PHP αρχείο θέλουμε να εμφανίσουμε δέκα πελάτες από τον γράφο «retail»:

```
<?php
    $conn   = odbc_connect('VOS', 'dba', 'dba');
    $query  = 'SELECT * from <http://localhost:8890/retail> WHERE {?id
foaf:name ?name.}LIMIT 10';


    $result = odbc_exec($conn, 'CALL DB.DBA.SPARQL_EVAL(\'' . $query .
'\', NULL, 0)');
    odbc_result_all ($result);
    odbc_close ($db);
?>
```

Θα μας επιστραφεί το εξής αποτέλεσμα:



Εικόνα 11.12: Αποτελέσματα εκτέλεσης SPARQL ερωτήματος με PHP

11.6 ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο παρουσιάστηκαν οι τρόποι με τους οποίους δημοσιεύθηκε η βάση που περιγράφθηκε στο κεφάλαιο εννέα με το εργαλείο Virtuoso και εκτελέστηκαν μερικά SPARQL ερωτήματα στα δεδομένα της βάσης, καθώς και η εκτέλεση SQL και SPARQL ερωτημάτων στα δεδομένα της βάσης μέσω της PHP.

ΚΕΦΑΛΑΙΟ 12

ΣΥΓΚΡΙΣΗ D2R SERVER ΜΕ VIRTUOSO

12.1 ΕΙΣΑΓΩΓΗ

Και ο D2R Server και το Virtuoso είναι δύο χρήσιμα εργαλεία για τη δημοσίευση RDF δεδομένων στο Διαδίκτυο. Αν και μπορούμε να τα χρησιμοποιήσουμε για τον ίδιο λόγο, στην ουσία είναι τελείως διαφορετικά μεταξύ τους σε πολλά σημεία.

12.2 Διαφορές μεταξύ της D2RQ πλατφόρμας με το OpenLink Virtuoso (open source)

Και με τα δύο εργαλεία μπορούμε να δημοσιεύσουμε RDF αρχεία στο Διαδίκτυο, χαρτογραφώντας μία σχεσιακή βάση δεδομένων και να εκτελέσουμε SPARQL ερωτήματα πάνω σε αυτά τα δεδομένα. Ωστόσο, υπάρχουν πολλές διαφορές μεταξύ τους. Οι κυριότερες είναι:

- Στην ουσία, το Virtuoso είναι μία αντικειμενο-σχεσιακή βάση δεδομένων όπου μας δίνει τη δυνατότητα να δημοσιεύσουμε τα δεδομένα μας σε RDF μορφή. Δηλαδή, εκτός από την ικανότητα δημοσίευσης RDF δεδομένων μας προσφέρει και ένα δικό του DBMS. Ενώ η D2RQ δεν έχει δικό της DBMS, αλλά συνδέεται με άλλα DBMS όπως τη MySQL και δημιουργεί μία χαρτογράφηση της βάσης δεδομένων, δηλαδή είναι μόνο ένα εργαλείο για δημοσίευση RDF δεδομένων.
- Το OpenLink Virtuoso (open source) δεν μας επιτρέπει να συνδεθούμε με άλλα DBMS, σε αντίθεση με τη D2RQ που γίνεται καθώς δεν έχει δικό της DBMS. Όμως μπορούμε να μεταφέρουμε δεδομένα από άλλες βάσεις στο Virtuoso με αρχεία της μορφής CSV.
- Για το Virtuoso υπάρχουν διαθέσιμες δύο εκδόσεις, μία «open source» και μια «commercial», που διαφέρουν ως προς κάποια επιπλέον λειτουργικότητα που υποστηρίζει η «commercial» έκδοση. Ενώ η D2RQ έχει μία έκδοση μόνο που είναι δωρεάν.
- Στο Virtuoso μπορούμε να φορτώσουμε RDF dumps και να εκτελέσουμε SPARQL ερωτήματα πάνω σε αυτά. Ενώ στη D2RQ μπορούμε μόνο να δημιουργήσουμε RDF dumps, οπότε δεν έχουμε τη δυνατότητα να εκτελέσουμε SPARQL ερωτήματα πάνω τους. Αυτό γίνεται γιατί το Virtuoso μας προσφέρει ένα triple store, ενώ η D2RQ όχι.
- Το Virtuoso χρησιμοποιεί την γλώσσα R2RML για να χαρτογραφήσει τη βάση δεδομένων του, ώστε να είναι σε μορφή RDF. Η D2RQ από την άλλη χρησιμοποιεί μία δική της γλώσσα για την χαρτογράφηση μίας βάσης δεδομένων, την D2R mapping language.
- Τα SPARQL ερωτήματα στη D2RQ μετασχηματίζονται σε SQL ερωτήματα για να εκτελεστούν στη βάση, ενώ στο Virtuoso δεν χρειάζεται αυτό.
- Το Virtuoso μπορεί να εκτελέσει SPARQL ερωτήματα είτε πάνω σε δεδομένα της βάσης του που έχει χαρτογραφηθεί, είτε πάνω σε RDF dumps που έχουν φορτωθεί στο Virtuoso. Από την άλλη, η D2RQ εκτελεί ερωτήματα SPARQL μόνο πάνω σε βάσεις με τι οποίες έχει συνδεθεί και τις έχει χαρτογραφήσει.

- Με το Virtuoso και με το D2R Server μπορούμε να εκτελέσουμε ερωτήματα σε όψεις (views) που έχουμε δημιουργήσει στη βάση. Επειδή στη MySQL δεν είναι εφικτό να δημιουργήσουμε index σε μια όψη, κατά τη διάρκεια της δημιουργίας του αρχείου-χάρτη από το D2R Server θα επιστρέφει σφάλμα καθώς δεν υπάρχει κύριο κλειδί, τότε στο αρχείο-χάρτη θα πρέπει να προσθέσουμε εμείς την όψη και να προσδιορίσουμε εμείς με ποία στήλη θα γίνεται ο προσδιορισμός της όψης.
- Ότι αλλαγές γίνουν στα δεδομένα της βάσης, είτε στη MySQL είτε στου Virtuoso, θα εμφανίζονται στα αποτελέσματα και δεν χρειάζεται να γίνει από την αρχή η χαρτογράφηση. Αν όμως αλλάξει το σχήμα της βάσης, προστεθεί κάποιος πίνακας ή μια στήλη σε έναν πίνακα κτλ, η βάση θα πρέπει να χαρτογραφηθεί ξανά από την αρχή και αυτό ισχύει και για τα δύο εργαλεία.

12.3 Πρόσθεση λεξιλογίων στο Virtuoso

Η διαδικασία τροποποίησης του αρχείου-χάρτη στο D2R Server, ώστε να προστεθούν νέα λεξιλόγια, περιγράφθηκε στο κεφάλαιο 7.4. Με την τροποποίηση του αρχείου-χάρτη, τα «rdf dump» αρχεία που δημιουργεί ο D2R Server και τα μεταφορτώνουμε στο Virtuoso θα περιέχουν και αυτά τα λεξιλόγια που προσθέσαμε, οπότε δεν χρειάζεται να γίνουν επιπλέον ενέργειες σε αυτήν την περίπτωση ώστε να προστεθούν νέα λεξιλόγια. Με αυτό τον τρόπο, αν θέλουμε να προσθέσουμε ή να αφαιρέσουμε λεξιλόγια, θα πρέπει να τροποποιήσουμε ξανά το αρχείο-χάρτη και να δημιουργήσουμε από την αρχή τα «rdf dump» αρχεία και να τα μεταφορτώσουμε στο Virtuoso. Στο κεφάλαιο 11.3 αναφέρθηκε ότι το Virtuoso δημιουργεί αρχεία-χάρτες και οντολογίες για να δημοσιεύσει τη βάση. Τα αρχεία-χάρτες που δημιουργεί είναι τρία, στο πρώτο ορίζεται σε ποία στήλη θα ανήκει κάποια συγκεκριμένη rdf ιδιότητα από κάποιο λεξιλόγιο. Προσθέτουμε το λεξιλόγιο που θέλουμε και την ιδιότητα στη στήλη του πίνακα που θέλουμε, στην εικόνα 12.1 προσθέτουμε το λεξιλόγιο «foaf» και την ιδιότητα «foaf:name» στη στήλη «name» του πίνακα «CUSTOMER».

Definitions

```
SPARQL
prefix retaildbview: <http://localhost:8890/schemas/retaildbview/>
prefix retaildbview-stat: <http://localhost:8890/retaildbview/stat#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix void: <http://rdfs.org/ns/void#>
prefix scovo: <http://purl.org/NET/scovo#>
prefix aowl: <http://bblfish.net/work/atom-owl/2006-06-06/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
alter quad storage virtrdf:DefaultQuadStorage
 from "CSV"."DBA"."customer" as customer_s
 from "CSV"."DBA"."pos_data" as pos_data_s
 where (^{pos_data_s.}^."customerid" = ^{customer_s.}^."id")
 {
   create retaildbview:qm-customer as graph iri ("http://^{URIQADefaultHost}^/retaildbview#")
    {
      # Maps from columns of "CSV.DBA.customer"
      retaildbview:customer (customer_s."id")  a retaildbview:customer ;
       a foaf:Person ;
      retaildbview:id customer_s."id" as retaildbview:dba-customer-id ;
      foaf:name customer_s."name" as retaildbview:dba-customer-name ;
      # Maps from foreign-key relations of "CSV.DBA.customer"
```

Εικόνα 12.1: Τροποποίηση των Definitions

Στη συνέχεια προσθέτουμε απλά το λεξιλόγιο «foaf» στο «R2RML Graph» που δημιουργεί το Virtuoso και το οποίο χαρτογραφεί τη βάση, εικόνα 12.2.

R2RML Graph

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix retaildbview: <http://localhost:8890/schemas/retaildbview/> .
@prefix retaildbview-stat: <http://localhost:8890/retaildbview/stat#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix scovo: <http://purl.org/NET/scovo#> .
@prefix aowl: <http://bblfish.net/work/atom-owl/2006-06-06/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .


<#TriplesMaparticle_categories> a rr:TriplesMap; rr:logicalTable [ rr:tableSchema "CSV" ; rr:tableOwner "DBA" ;
rr:tableName "article_categories" ];
rr:subjectMap [ rr:termtype "IRI"  ; rr:template "http://localhost:8890/CSV/article_categories/id={id}"; rr:class
retaildbview:article_categories; rr:graph <http://localhost:8890/retaildbview#> ];
rr:predicateObjectMap [ rr:predicateMap [ rr:constant retaildbview:id ] ; rr:objectMap [ rr:column "id" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant retaildbview:descc ] ; rr:objectMap [ rr:column "descc" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant retaildbview:article_categories_of_articles ] ; rr:objectMap [
rr:parentTriplesMap <#TriplesMaparticles>;  rr:joinCondition [ rr:child "id" ; rr:parent "article_category" ] ; ] ; ] .

<#TriplesMaparticles> a rr:TriplesMap; rr:logicalTable [ rr:tableSchema "CSV" ; rr:tableOwner "DBA" ; rr:tableName
"articles" ];
```

Εικόνα 12.2: Τροποποίηση του R2RML Graph

Τέλος, προσθέτουμε ξανά το λεξιλόγιο «foaf» στις οντολογίες, οι οποίες ορίζουν τις κλάσεις και σε ποιο γράφο ανήκουν οι πίνακες, εικόνα 12.3.

Ontology

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix aowl: <http://bblfish.net/work/atom-owl/2006-06-06/> .
@prefix virtrdf: <http://www.openlinksw.com/schemas/virtrdf#> .
@prefix retaildbview: <http://localhost:8890/schemas/retaildbview/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

retaildbview: a owl:Ontology .

# CSV.DBA.article_categories
retaildbview:article_categories a rdfs:Class .
retaildbview:article_categories rdfs:isDefinedBy retaildbview: .
retaildbview:article_categories rdfs:label "CSV.DBA.article_categories" .
retaildbview:id a owl:DatatypeProperty .
retaildbview:id rdfs:range xsd:int .
retaildbview:id rdfs:domain retaildbview:article_categories .
retaildbview:id rdfs:isDefinedBy retaildbview: .
retaildbview:id rdfs:label "id" .
retaildbview:descc a owl:DatatypeProperty .
```

Εικόνα 12.3: Τροποποίηση του Ontology αρχείου

Οπότε με αυτές τις αλλαγές, δηλαδή που προστέθηκε το λεξιλόγιο «foaf» και στα τρία αρχεία που δημιουργεί το Virtuoso και την πρόσθεση της ιδιότητας «foaf:name», μπορούμε να εκτελέσουμε SPARQL ερωτήματα μέσω του SPARQL endpoint του Virtuoso που περιέχουν το λεξιλόγιο «foaf» και την ιδιότητα «foaf:name».

12.4 Σύγκριση επιδόσεων χρόνου σε SPARQL ερωτήματα

1. Δέκα προϊόντα που αγόρασε ο πελάτης «Atkinson M».

Virtuoso (RDF Views)

```
select Distinct ?name ?desc where {

  ?cid foaf:name ?name.

  ?rid <http://localhost:8890/schemas/testvocabview/has_customer> ?cid;

        <http://localhost:8890/schemas/testvocabview/itemid> ?itid.

  ?item dc:identifier ?itid;

          <http://localhost:8890/schemas/testvocabview/article_category> ?catid.

  ?cat  dc:identifier ?catid;

         dc:description ?desc.

FILTER(regex(?name, "Atkinson M", "i"))

}

LIMIT 10
```

Χρόνος εκτέλεσης: 1s (seconds)

Virtuoso (RDF dump)

```
SELECT Distinct ?name ?desc WHERE {

  ?cid foaf:name ?name.

  ?rid <http://localhost:2020/vocab/pos_data_customerid> ?cid;

      <http://localhost:2020/vocab/pos_data_itemid> ?itid.

  ?itid dc:identifier ?art;
```

```
      <http://localhost:2020/vocab/articles_article_category> ?catid.
  ?catid dc:description ?desc.
FILTER(regex(?name, "Atkinson M", "i"))
}
LIMIT 10
```

Χρόνος εκτέλεσης: 3,5s

D2R Server

```
SELECT Distinct  ?name ?desc WHERE {
  ?cid foaf:name ?name.
  ?rid vocab:pos_data_customerid ?cid;
       vocab:pos_data_itemid ?itid.
  ?itid dc:identifier ?art;
       vocab:articles_article_category ?catid.
  ?catid dc:description ?desc.
FILTER(regex(?name, "Atkinson M", "i"))
}
LIMIT 10
```

Χρόνος εκτέλεσης: 4,3s

2. Προϊόντα που αγόρασε ο πελάτης «Atkinson M» που είτε έχουν «promotion» είτε όχι.

Virtuoso (RDF Views)

```
select Distinct ?name ?desc where {
  ?cid foaf:name ?name.
```

```
  ?rid <http://localhost:8890/schemas/testvocabview/has_customer> ?cid;

        <http://localhost:8890/schemas/testvocabview/itemid> ?itid.

  ?item dc:identifier ?itid;

          <http://localhost:8890/schemas/testvocabview/article_category> ?catid.

  ?cat  dc:identifier ?catid;

         dc:description ?desc.

OPTIONAL{ ?itid <http://localhost:8890/schemas/testvocabview/promotion> ?prom}

FILTER(regex(?name, "Atkinson M", "i"))

}

LIMIT 100
```

Χρόνος εκτέλεσης: 1s

Virtuoso (RDF dump)

```
SELECT Distinct ?name ?desc WHERE {

  ?cid foaf:name ?name.

  ?rid <http://localhost:2020/vocab/pos_data_customerid> ?cid;

       <http://localhost:2020/vocab/pos_data_itemid> ?itid.

  ?itid dc:identifier ?art;

       <http://localhost:2020/vocab/articles_article_category> ?catid.

  ?catid dc:description ?desc.

OPTIONAL{ ?itid <http://localhost:2020/vocab/pos_data_promotion> ?prom}

FILTER(regex(?name, "Atkinson M", "i"))

}LIMIT 100.
```

Χρόνος εκτέλεσης: 4,2s

D2R Server

```
SELECT Distinct  ?name ?desc WHERE {

  ?cid foaf:name ?name.

  ?rid vocab:pos_data_customerid ?cid;

       vocab:pos_data_itemid ?itid.

  ?itid dc:identifier ?art;

       vocab:articles_article_category ?catid.

  ?catid dc:description ?desc.

OPTIONAL{ ?itid <http://localhost:8890/schemas/testvocabview/promotion> ?prom}

FILTER(regex(?name, "Atkinson M", "i"))

}

LIMIT 100
```

Χρόνος εκτέλεσης: 3,3s

3. Εμφάνιση των προϊόντων με την περιγραφή τους.

Virtuoso (RDF Views)

```
select distinct ?itid ?desc where {

  ?rid <http://localhost:8890/schemas/testvocabview/itemid> ?itid.

  ?item dc:identifier ?itid;

          <http://localhost:8890/schemas/testvocabview/article_category> ?catid.

  ?cat  dc:identifier ?catid;

         dc:description ?desc.

} LIMIT 10000
```

Χρόνος εκτέλεσης: 1,3s

Virtuoso (RDF dump)

```
SELECT distinct ?itid ?desc WHERE {

  ?rid <http://localhost:2020/vocab/pos_data_itemid> ?itid.

  ?itid dc:identifier ?art;

       <http://localhost:2020/vocab/articles_article_category> ?catid.

  ?catid dc:description ?desc.

}

LIMIT 10000
```

Χρόνος εκτέλεσης: 1,5s

D2R Server

```
SELECT Distinct  ?itid ?desc WHERE {

  ?rid vocab:pos_data_itemid ?itid.

  ?itid dc:identifier ?art;

       vocab:articles_article_category ?catid.

  ?catid dc:description ?desc.

}LIMIT 10000
```

Χρόνος εκτέλεσης: 13,3s

4. Η συναλλαγή που αγοράζει το μεγαλύτερο αριθμό προϊόντων.

Virtuoso (RDF Views)

```
select  ?transid (count(?itid) as ?count) where {

  ?rid <http://localhost:8890/schemas/testvocabview/transid> ?transid;

       <http://localhost:8890/schemas/testvocabview/itemid> ?itid.
```

```
}

group by ?transid

order by desc(?count)

limit 1
```

Χρόνος εκτέλεσης: 2s

Virtuoso (RDF dump)

```
select  ?transid (count(?itid) as ?count) where {

  ?rid <http://localhost:2020/vocab/pos_data_transid> ?transid;

       <http://localhost:2020/vocab/pos_data_itemid> ?itid.

}

group by ?transid

order by desc(?count)

limit 1
```

Χρόνος εκτέλεσης: 2,5s

D2R Server

```
select  ?transid (count(?itid) as ?count) where {

  ?rid vocab:pos_data_transid ?transid;

       vocab:pos_data_itemid ?itid.}

group by ?transid

order by desc(?count)

limit 1
```

Χρόνος εκτέλεσης: 9s

5. Οι πέντε κατηγορίες προϊόντων που αγοράζονται περισσότερο στις συναλλαγές.

Virtuoso (RDF Views)

```
select  ?desc (count(?transid) as ?count) where {

  ?rid <http://localhost:8890/schemas/testvocabview/transid> ?transid;

       <http://localhost:8890/schemas/testvocabview/itemid> ?itid.

  ?item dc:identifier ?itid;

            <http://localhost:8890/schemas/testvocabview/article_category> ?catid.

  ?cat  dc:identifier ?catid;

           dc:description ?desc.

}

group by ?desc

order by desc(?count)

limit 5
```

Χρόνος εκτέλεσης: 2,4s

Virtuoso (RDF dump)

```
select  ?desc (count(?transid) as ?count) where {

  ?rid <http://localhost:2020/vocab/pos_data_transid> ?transid;

       <http://localhost:2020/vocab/pos_data_itemid> ?itid.

  ?itid dc:identifier ?art;

       <http://localhost:2020/vocab/articles_article_category> ?catid.

  ?catid dc:description ?desc.

}

group by ?desc
```

```
order by desc(?count)

limit 5
```

Χρόνος εκτέλεσης: 2,7s

D2R Server

```
select  ?desc (count(?itid) as ?count) where {

  ?rid vocab:pos_data_transid ?transid;

       vocab:pos_data_itemid ?itid.

  ?itid dc:identifier ?art;

       vocab:articles_article_category ?catid.

  ?catid dc:description ?desc.

}

group by ?desc

order by desc(?count)

limit 5
```

Χρόνος εκτέλεσης: 10s

12.5 ΕΠΙΛΟΓΟΣ

Και με το Virtuoso και με το D2RQ μπορούμε να δημοσιεύσουμε RDF δεδομένα στο Διαδίκτυο. Ανάλογα βέβαια με τις ανάγκες μας και το τι θέλουμε να κάνουμε, χρησιμοποιούμε το σύστημα που χρειαζόμαστε.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Λόγω του τεράστιου όγκου δεδομένων που υπάρχει στις μέρες μας στο Διαδίκτυο, υπάρχει ανάγκη για καλύτερη αναζήτηση και εκμετάλλευση των δεδομένων και των πληροφοριών που υπάρχουν. Μάλιστα η ανάγκη αυτή γίνεται όλο και πιο δύσκολη να υλοποιηθεί, καθώς το Διαδίκτυο συνεχώς αναπτύσσεται. Με την υπάρχουσα μορφή του Διαδικτύου, ο διαχωρισμός των πληροφοριών και η σύνδεση μεταξύ τους είναι σχεδόν αδύνατη καθώς απαιτεί την παρέμβαση του ανθρώπου.

Οπότε, για να λυθεί αυτό το πρόβλημα, αναπτύχθηκαν νέες τεχνολογίες, όπως το Semantic Web και τα Linked Data και άλλες. Με τις τεχνολογίες αυτές, μπορούμε να συνδέουμε πληροφορίες από διαφορετικές πηγές, που υπάρχουν διάσπαρτες στο Διαδίκτυο. Ακόμα, μας επιτρέπουν να δίνουμε μία σημασιολογία στα δεδομένα μας, η οποία θα γίνεται κατανοητή και από τους υπολογιστές, με αποτέλεσμα να συνδέονται μεταξύ τους τα δεδομένα χωρίς να χρειάζεται η παρέμβαση του ανθρώπου. Αυτό είναι ένα τρομερό πλεονέκτημα, καθώς υπάρχει καλύτερη διαχείριση πληροφοριών, αφού η αναζήτηση και η εύρεση δεδομένων και πληροφοριών γίνεται πιο εύκολη και πιο αποτελεσματική, αφού τα δεδομένα μας είναι εμπλουτισμένα με ένα σημασιολογικό περιεχόμενο.

Έτσι το Διαδίκτυο, τείνει, να μετατραπεί σε έναν τεράστιο συνδεδεμένο γράφο, που θα έχει σημασία και για τους υπολογιστές πέρα από τους ανθρώπους. Πλέον υπάρχουν και εργαλεία που μας βοηθούν να το επιτύχουμε αυτό.

Στόχοι της πτυχιακής αυτής ήταν η δημοσίευση του περιεχομένου μίας σχεσιακής βάσης δεδομένων σε μορφή Linked Data, καθώς και της δυνατότητας να εκτελούμε SPARQL ερωτήματα στα δεδομένα της βάσης αυτής, με την χρήση των εργαλείων D2R Server και Virtuoso. Οι στόχοι που τέθηκαν στο ξεκίνημα υλοποιήθηκαν, καθώς η βάση δεδομένων που χρησιμοποιήθηκε δημοσιεύτηκε σε μορφή Linked Data και με τα δύο εργαλεία και εκτελέστηκαν επιτυχώς SPARQL ερωτήματα στα δεδομένα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. D2R Server tutorial. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://sw.cs.technion.ac.il/d2rq/tutorial
2. D2R Server: Accessing Databases with SPARQL and as Linked Data, *Getting Started with D2RQ*. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://d2rq.org/getting-started
3. *D2R Server: Accessing Databases with SPARQL and as Linked Data.* Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://d2rq.org/d2r-server
4. *D2RQ Accessing Relational Databases as Virtual RDF Graphs*. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://d2rq.org
5. D2RQ Mapping Language, *The D2RQ Mapping Language.* Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://d2rq.org/d2rq-language
6. *D2RQ/Update and D2R/Update Server.* Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://d2rqupdate.cs.technion.ac.il
7. Dublin Core, Dublin Core Metadata Initiative, *Dublin Core Metadata Element Set, Version 1.1.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://dyblincore.org/documents/dces/
8. DuCharme Bob, *Learning SPARQL: Quering and Updating with SPARQL 1.1*, O'Reilly Media, 2011, United States of America.
9. GeoNames Ontology. Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://www.geonamew.org/ontology/documentation.html
10. Grigoris Antoniou, Frank van Harmelen, *A Semantic Web Primer,* The MIT Press, 2008, Massachusetts London, Engalnd.
11. Mark Watson, *Practical Semantic Web and Linked Data Applications,* 2010, United States of America.
12. Notation3. Ανακτήθηκε στις 10/4/2014 από τη διεύθυνση: http://en.wikipedia.org/wiki/Notation_3
13. N-Triples. Ανακτήθηκε στις 10/4/2014 από τη διεύθυνση: http://en.wikipedia.org/wiki/N-Triples
14. Openlink Virtuoso OpenSource. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSIntro
15. Openlink Virtuoso R2RML. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtR2RML
16. Openlink Software Virtuoso. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://virtuoso.openlinksw.com
17. *OWL Web Ontology Language Overview.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://www.w3.org/TR/owl-features/

18. Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, *Foundations of Semantic Web Technologies,* CRC Press, 2011, United States of America.
19. *Quick Intro to RDF*. . Ανακτήθηκε στις 7/4/2014 από τη διεύθυνση: http://www.rdfabout.com/quickintro.xpd
20. *R2RML: RDB to RDF Mapping Language.* Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://www.w3.org/TR/r2rml/
21. *RDF 1.1 Primer*. Ανακτήθηκε στις 7/4/2014 από τη διεύθυνση: http://www.w3.org/TR/rdf11-primer/
22. *RDF Primer*. Ανακτήθηκε στις 7/4/2014 από τη διεύθυνση: http://w3.org/TR/2004/REC-rdf-primer-20040210/
23. *RDF Schema 1.1.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://www.w3.org/TR/rdf-schema/
24. *RDF Schema.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://en.wikipedia.org/wiki/RDF_Schema
25. *RDF/XML Syntax Specification (Revised).* Ανακτήθηκε στις 10/4/2014 από τη διεύθυνση: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/
26. *RDFa 1.1 Primer – Second Edition.* Ανακτήθηκε στις 10/4/2014 από τη διεύθυνση: http://www.w3.org/TR/xhtml-rdfa-primer/
27. *Representing vCard Objects in RDF.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://www.w3.org/Submission/vcard-rdf/
28. *Resource Description Framework (RDF): Concepts and Abstract Syntax.* Ανακτήθηκε στις 7/4/2014 από τη διεύθυνση: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
29. *Resource Description Framework.* Ανακτήθηκε στις 7/4/2014 από τη διεύθυνση: http://www.w3.org/RDF/
30. *Resource Description Framework.* Ανακτήθηκε στις 7/4/2014 από τη διεύθυνση: http://en.wikipedia.org/wiki/Resource_Description_Framework
31. *SKOS Simple Knowledge Organization System Primer.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://www.w3.org/TR/skos-preimer/
32. *SKOS: Simple Knowledge Organization fro the Web.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://www.w3.org/2004/02/skos/
33. *SPARQL Query Language for RDF.* Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://www.w3.org/TR/rdf-sparql-query/
34. SPARQL. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://en.wikipedia.org/wiki/SPARQL
35. Standards, in W3C, *Semantic Web*. Ανακτήθηκε στις 4/4/2014 από τη διεύθυνση: http://www.w3.org/standards/semanticweb/
36. *The Friend of a Friend (FOAF) project.* Ανακτήθηκε στις 9/4/2014 από τη διεύθυνση: http://www.foaf-project.org

37. Tim Berners-Lee, Linked Data, in W3C, *Design Issues*, *Architectural and Philosophical Points*. Ανακτήθηκε στις 4/4/2014 από τη διεύθυνση: http://www.w3.org/DesignIssues/LinkedData.html
38. Toby Segaran, Colin Evans, Jamie Taylor, *Programming the Semantic Web*, O'Reilly Media, 2009, United States of America.
39. Tom Heath, Christian Bizer , *Linked Data : Evolving the Web into a Global Data Space*, Morgan & Claypool, 2011, London, England.
40. Tsoukalas Chrysostomos, *Διαχείριση Κατανεμημένων Δεδομένων στο Διαδίκτυο,* Πτυχιακή εργασία, 2011.
41. Turtle. Ανακτήθηκε στις 10/4/2014 από τη διεύθυνση: http://en.wikipedia.org/wiki/Turtle_(syntax)
42. Turtle. Ανακτήθηκε στις 10/4/2014 από τη διεύθυνση: http://www.w3.org/TR/2012/WD-turtle-20120710/
43. Virtuoso Universal Server. Ανακτήθηκε στις 15/4/2014 από τη διεύθυνση: http://en.wikipedia.org/wiki/Virtuoso_Universal_Server