Μια δηλωτική γλώσσα επερώτησης μεταδεδομένων RDF για Πύλες Κοινοτήτων Διαδικτύου

Γρηγόρης Καρβουναράκης

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Πληροφοριακά συστήματα όπως συστήματα μνήμης οργανισμών (organizational memory), κάθετοι συναθροιστές (vertical aggregators), διαμεσολαβητές πληροφορίας (infomediaries) κλπ αναμένεται να παίξουν πρωταρχικό ρόλο στην οικονομία του 21ου αιώνα, επιτρέποντας την ανάπτυξη και συντήρηση συγκεκριμένων κοινοτήτων (π.χ. επιχειρησιακών, επαγγελματικών, εμπορικών) σε επιχειρησιακά κλειστά δίκτυα (intranets) ή στον Παγκόσμιο Ιστό. Τέτοιες Πύλες Κοινοτήτων Διαδικτύου προσφέρουν σε μέλη διαφόρων κοινοτήτων (επιχειρησιακών, δια-επιχειρησιακών, ηλεκτρονικού εμπορίου κλπ) τη δυνατότητα επιλογής, κατηγοριοποίησης και πρόσβασης σε διάφορους πληροφοριακούς πόρους (π.χ. δικτυακούς τόπους, έγγραφα, δεδομένα, υπηρεσίες).

Παρολαυτά, στις εμπορικές εφαρμογές που υποστηρίζουν την ανάπτυξη πυλών για κοινότητες, οι δυνατότητες επερώτησης περιορίζονται σε ανάκτηση πλήρους κειμένου (full-text retrieval) (ή ζεύγη γνωρισμάτων-τιμών) και πιο προηγμένες λειτουργίες είναι δυνατές μόνο με πρόσβαση στους πόρους μέσω πλοήγησης (navigational access). Επίσης, τέτοιες εφαρμογές αγνοούν τελείως πρόσφατα πρότυπα για την περιγραφή πόρων στον Παγκόσμιο Ιστό (βλέπε δραστηριότητα μεταδεδομένων του W3C: RDF/RDF Schema). Επίσης, οι καθιερωμένες (σχεσιακές ή οντοκεντρικές) βάσεις δεδομένων βασίζονται σε μοντέλα που δεν είναι αρκετά ευέλικτα ώστε να μπορούν να αναπαραστήσουν τα ιδιαίτερα χαρακτηριστικά του RDF.

Ορμώμενοι από τα παραπάνω ζητήματα, προτείνουμε ένα νέο μοντέλο δεδομένων και μια δηλωτική γλώσσα επερώτησης για περιγραφές και σχήματα RDF. Η γλώσσα μας, η *RQL* (RDF Query Language) - βασίζεται σε ένα τυπικό μοντέλο γράφων που εμπερικλείει τις βασικές έννοιες μοντελοποίησης του RDF, προσφέροντας και ενα πιο πλούσιο σύστημα τύπων, και επιτρέπει την ερμηνεία περιγρα-

φών RDF από ένα ή περισσότερα σχήματα. Η RQL προσαρμόζει τη λειτουργικότητα των γλωσσών επερώτησης ημιδομημένων εγγράφων στα ιδιαίτερα χαρακτηριστικά του RDF αλλά επίσης την επεκτείνει, προκειμένου να επιτρέψει την επερώτηση σχημάτων RDF. Η πρωτοτυπία της RQL είναι η δυνατότητα ομαλής μετάβασης μεταξύ ερωτήσεων σε σχήματα και σε περιγραφές πόρων, καθώς και η εκμετάλλευση - με διαφανή τρόπο - των ιεραρχιών υπαλληλίας μεταξύ των ετικεττών.

Επόπτης: Πάνος Κωνσταντόπουλος
Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

A declarative RDF metadata query language for Community Web Portals

Gregory Karvounarakis

Master of Science Thesis

Computer Science Department
University of Crete

Abstract

Information systems such as organizational memories, vertical aggregators, infomediaries, etc. are expected to play a central role in the 21st-century economy by enabling the development and maintenance of specific communities of interest (e.g., enterprise, professional, trading) on corporate intranets or the Web. Such Community Web Portals essentially provide the means to select, classify and access, in a semantically meaningful and ubiquitous way various information resources (e.g., sites, documents, data) for diverse target audiences (corporate, inter-enterprise, e-marketplace, etc.).

Yet, in commercial software for deploying Community Portals, querying is still limited to full-text (or attribute-value) retrieval and more advanced information-seeking needs require navigational access. Furthermore, recent Web standards for describing resources (see the W3C Metadata Activity: RDF/ RDF Schema) are completely ignored. Moreover, standard (relational or object) databases are too rigid for capturing the peculiarities of RDF descriptions and schemas.

Motivated by the above issues, we propose a new data model and a query language for RDF descriptions and schemas. Our language, called *RQL*, relies on a formal graph model, that captures the RDF modeling primitives, also providing a richer type system, and permits the interpretation of RDF descriptions by means of one or more schemas. In this context, *RQL* adapts the functionality of semistructured query languages to the peculiarities of RDF but also extends this functionality in order to query RDF schemas. The novelty of *RQL* lies in its ability to smoothly switch between schema and data querying while exploiting - in a

transparent way - the taxonomies of labels and multiple classification of resources.

Supervisor: Panos Constantopoulos
Professor of Computer Science
University of Crete

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον επόπτη μου, κ. Πάνο Κωνσταντόπουλο που μου εμπιστεύθηκε την εργασία αυτή.

Επίσης, θα ήθελα να ευχαριστήσω τα μέλη της εισηγητικής επιτροπής, κ. Δημήτρη Πλεξουσάκη και κ. Ευάγγελο Μαρκάτο για τις εποικοδομητικές τους παρατηρήσεις και διορθώσεις στην εργασία αυτή.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τον επιβλέποντα της εργασίας μου, κ. Βασίλη Χριστοφίδη, που μαζί με τον κ. Πλεξουσάκη ήταν οι άμεσοι επόπτες της εργασίας αυτής, τόσο για την καθοδήγησή του στην επιλογή του θέματος της εργασίας όσο και για τις γνώσεις τις οποίες απέκτησα κοντά του τον τελευταίο χρόνο. Οφείλω να αναγνωρίσω ότι η συνεργασία μεταξύ των τριών μας ήταν κάτι παραπάνω από άψογη, και οι αμέτρητες ώρες των συζητήσεών μας πραγματικά παραγωγικές.

Επίσης θα ήθελα να ευχαριστήσω την Σοφία Αλεξάκη και τον Karsten Tolle, οι μεταπτυχιακές εργασίες των οποίων αποτέλουν τμήματα του συνολικού συστήματος - στο πλαίσιο του οποίου δημιουργήθηκε αυτή η εργασία, με τους οποίους είχαμε μια άψογη συνεργασία. Ακόμα, θα ήθελα να ευχαριστήσω την Ειρήνη Φουντουλάκη, για τη συνεισφορά της σε πολύωρες συζητήσεις στην προσπάθεια κατανόησης και τυπικής περιγραφής της σημασιολογίας του RDF.

Πρέπει επίσης να ευχαριστήσω τις ομάδες Πληροφοριακών Συστημάτων και Παράλληλων και Κατανεμημένων Συστημάτων (Πλειάδες) του Ινστιτούτου Πληροφορικής - Ι.Τ.Ε. στις οποίες απασχολήθηκα τα τελευταία 4,5 χρόνια, τόσο γιατί μου παρείχαν τον απαραίτητο εργαστηριακό εξοπλισμό όσο και για τις γνώσεις που απέκτησα εργαζόμενος σε αυτές. Επίσης, οφείλω να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης, για τις γνώσεις που μου έδωσε την ευκαιρία να αποκτήσω τα τελευταία 6 χρόνια.

Ακόμη θα ήθελα να ευχαριστήσω την Ξένια Μουντρουίδου για την βοήθεια και την αμέριστη συμπαράστασή της, ιδιαίτερα στις δύσκολες στιγμές της συγγραφής αυτής της εργασίας.

Το μεγαλύτερο όμως ευχαριστώ το αξίζουν οι γονείς μου, για τους κόπους τους

και τα όσα θυσίασαν στη ζωή τους για τη μόρφωση και την πνευματική μου καλλιέργεια. Από την πλευρά μου το ελάχιστο που μπορώ να προσφέρω ως ανταμοιβή είναι να τους αφιερώσω αυτή την εργασία, ελπίζοντας ότι κατάφερα να φανώ αντάξιος των προσδοκιών τους.

Γρηγόρης Καρβουναράκης

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Εισαγωγή

1.1 Πύλες Κοινοτήτων Διαδικτύου

Πληροφοριακά συστήματα όπως συστήματα μνήμης οργανισμών (organizational memory), κάθετοι συναθροιστές (vertical aggregators), διαμεσολαβητές πληροφορίας (infomediaries) κλπ αναμένεται να παίξουν πρωταρχικό ρόλο στην οικονομία του 21ου αιώνα, επιτρέποντας την ανάπτυξη και συντήρηση συγκεκριμένων *κοινοτήτων* (π.χ. επιχειρησιακών, επαγγελματικών, εμπορικών) σε επιχειρησιακά κλειστά δίκτυα (intranets) ή στον Παγκόσμιο Ιστό [FA99]. Τέτοιες *πύλες κοινοτήτων διαδικτύου* προσφέρουν σε μέλη διαφόρων κοινοτήτων (επιχειρησιακών, δια-επιχειρησιακών, ηλεκτρονικού εμπορίου κλπ) τη δυνατότητα επιλογής, κατηγοριοποίησης και πρόσβασης σε διάφορους πληροφοριακούς πόρους (π.χ. δικτυακούς τόπους, έγγραφα, δεδομένα, υπηρεσίες). Μάλιστα, οι λειτουργίες αυτές είναι σημασιολογικά πλούσιες, κατανοητές, ενοποιημένες (αναφέρονται δηλ. στο σύνολο των πληροφοριακών πόρων της κοινότητας) και καθολικές (μπορούν δηλ. να εκτελεστούν από παντού). Ένα βασικό τμήμα μιας τέτοιας πύλης είναι ο *Κατάλογος Γνώσης*, ο οποίος περιέχει περιγραφές - δηλ. μεταδεδομένα - σχετικά με τους υπάρχοντες πληροφοριακούς πόρους. Προκειμένου να επιτρέψει την - προσαρμοσμένη στα ενδιαφέροντα και τις ανάγκες κάθε χρήστη (personalized) - πρόσβαση, οι Κατάλογοι Γνώσης οργανωνουν και συγκεντρώνουν πληροφορία με διάφορους τρόπους, που είναι πολύ πιο ευέλικτοι σε σχέση με τα συνηθισμένα (σχεσιακά ή οντοκεντρικά) συστήματα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ). Παρόλαυτα, στις εμπορικές εφαρμογές που υποστηρίζουν την ανάπτυξη πυλών για κοινότητες (π.χ. Epicentric[1], Plumtree[2], Autonomy[3]) οι δυνατότητες επερώτησης πε-

[1] www.epicentric.com

[2] www.plumtree.com

[3] www.autonomy.com

ριορίζονται σε ανάκτηση πλήρους κειμένου (full-text retrieval) (ή ζεύγη γνωρισμάτων-τιμών) και πιο προηγμένες λειτουργίες είναι δυνατές μόνο με πρόσβαση στους πόρους μέσω πλοήγησης (navigational access). Επίσης, τέτοιες εφαρμογές αγνοούν τελείως πρόσφατα πρότυπα για την περιγραφή πόρων στον Παγκόσμιο Ιστό (βλέπε δραστηριότητα μεταδεδομένων του W3C[4]).

Πιο συγκεκριμένα, το **Περιβάλλον Περιγραφής Πόρων** (Resource Description Framework - RDF)[LS99], [BG00], που αποτελεί πρότυπο του W3C, έχει στόχο να επιτρέψει τη δημιουργία και ανταλλαγή περιγραφών πόρων μεταξύ κοινοτήτων. Μάλιστα, λόγω του ευέλικτου μοντέλου του, πολλοί προμηθευτές πληροφορίας (π.χ. ABCNews, CNN, Time Inc., New Media) αλλά και πύλες στον Παγκόσμιο Ιστό (π.χ. Open Directory[5], CNET[6], XMLNews[7]) χρησιμοποιούν ήδη το RDF[LS99] για την περιγραφή των πόρων τους. Πράγματι, η πληροφορία που περιλαμβάνει μια κοινότητα μπορεί να θεωρηθεί ως μια συλλογή από πόρους με αναγνωριστικά URIs[BLFIM98], οι περιγραφές των οποίων μπορούν να μοντελοποιηθούν από επώνυμες ιδιότητες - όπως στο RDF. Έτσι, το RDF είναι κατάλληλο για την δημιουργία μεταδεδομένων διαφόρων ειδών (διαχειριστικών, καταλογοποίησης (cataloguing), αξιολόγησης περιεχομένου (content rating), χάρτες δικτυακών τόπων, προσωποποιημένα κανάλια (push channels) κλπ) για ετερογενείς πόρους (από τελείως αδόμητους όπως έγγραφα RDF ή MSWord, αρχεία video ή ήχου, σε ημι-δομημένα - π.χ. έγγραφα HTML ή συλλογές ηλεκτρονικών μηνυμάτων - ή και δομημένα, όπως έγγραφα XML που ακολουθούν κάποιο γνωστό Ορισμό Τύπου Εγγράφου (DTD)). Επίσης, προκειμένου να επιτρέψει την ερμηνεία των περιγραφών πόρων, μεταξύ κοινοτήτων είτε στο πλαίσιο μίας κοινότητας, το RDF επιτρέπει τον ορισμό σχημάτων[BG00]. Τα σχήματα RDF έχουν διάφορες ιδιαιτερότητες: (α) δεν επιβάλλουν αυστηρούς περιορισμούς τύπων στις περιγραφές (π.χ. ένα πόρος μπορεί να περιγραφεί από ιδιότητες που είναι loosely-coupled από τις κλάσεις), (β) επιτρέπουν επικαλυπτόμενες περιγραφές για τον ίδιο πόρο (π.χ. ένας πόρος μπορεί να κατηγοριοποιηθεί σε διάφορες κλάσεις που πιθανώς εξυπηρετούν διαφορετικούς σκοπούς περιγραφής και πιθανώς προέρχονται από διαφορετικά σχήματα RDF - π.χ. για διαχειριστικά μεταδεδομένα και μεταδεδομένα περιγραφής περιεχομένου), (γ) μπορούν

[4] www.w3.org/Metadata

[5] www.dmoz.org

[6] home.cnet.com

[7] www.xmlnews.org

να επεκταθούν εύκολα (π.χ. με τη σχέση υπαλληλίας τόσο στις κλάσεις όσο και στους τύπους ιδιοτήτων).

Πιο αυστηρά, στις περισσότερες *πύλες κοινοτήτων διαδικτύου*, οι περιγραφές των πόρων, αν και είναι ετερογενείς, μπορούν να αναπαρασταθούν εύκολα και με φυσικό τρόπο στο μοντέλο *κατευθυνόμενων γράφων με ετικέττες* του RDF, στο οποίο οι κόμβοι και οι ακμές αναπαριστούν *πόρους* (ή λεκτικές τιμές) και *ιδιότητες*, αντίστοιχα. Κατά συνέπεια, τα σχήματα RDF ορίζουν ουσιαστικά *λεξιλόγια* από ετικέττες (για κόμβους και ακμές του γράφου) που μπορούν να χρησιμοποιηθούν για την περιγραφή και την επερώτηση πόρων σε διάφορες εφαρμογές. Αυτές οι βασικές αρχές του μοντέλου είναι απαραίτητες για την υποστήριξη περιγραφών πληροφοριακών πόρων μιας κοινότητας για διάφορους σκοπούς. Ταυτόχρονα, διατηρούν την αυτονομία που απαιτείται για λειτουργίες περιγραφής ανεξάρτητων τμημάτων μιας κοινότητας (*υπο-κοινοτήτων*). Αξίζει να σημειωθεί ότι παρόμοιες ανάγκες έχουν διαπιστωθεί και σε άλλες δικτυακές εφαρμογές όπως τα συστήματα υπερκείμενης (superimposed) πληροφορίας[DM99], [MD99] και οι υπηρεσίες ιεραρχικών καταλόγων LDAP [JLM+99], [AYJLS00]. Επίσης, ο συνεχώς αυξανόμενος αριθμός πόρων κοινοτήτων και ο πολλαπλασιασμός των υπηρεσιών περιγραφής έχουν οδηγήσει σε τεράστιους όγκους περιγραφών και σχημάτων RDF (π.χ. η πύλη Open Directory περιέχει περίπου 100MB κατηγοριών/θεμάτων και τα αναγνωριστικά των πόρων που έχουν κατηγοριοποιηθεί σε αυτές φτάνουν τα 700MB). Δυστυχώς όμως, η μόνη λειτουργικότητα που προσφέρεται από τα υπάρχοντα συστήματα [RDF] που υποστηρίζουν το RDF είναι η - βασισμένη σε προγραμματιστικές διεπαφές χαμηλού επιπέδου (low-level APIs) - *ιεραρχική αναδίφυση*, η οποία σε τέτοιες τεράστιες *βάσεις περιγραφών* είναι δύσκολη, χρονοβόρα και πιθανότατα αναποτελεσματική διαδικασία. Η λογική ερώτηση που ανακύπτει λοιπόν είναι αν - και σε πιο βαθμό - μπορούμε να χρησιμοποιήσουμε τεχνολογία βάσεων δεδομένων για να προσφέρουμε *δηλωτική* (declarative) πρόσβαση και διαχείριση αποθήκευση σε περιγραφές RDF.

Δυστυχώς, οι καθιερωμένες (σχεσιακές ή οντοκεντρικές) βάσεις δεδομένων [AHV95] βασίζονται σε μοντέλα που δεν είναι αρκετά ευέλικτα ώστε να μπορούν να αναπαραστήσουν τα ιδιαίτερα χαρακτηριστικά του RDF (π.χ. οι κλάσεις στο RDF δεν ορίζουν τύπους για τα στιγμιότυπά τους ενώ οι ιδιότητες ορίζονται ανεξάρτητα από τις κλάσεις). Εξάλλου, τα περισσότερα ημιδομημένα μοντέλα - όπως το OEM

[CGH+94], [PGW95], [PGU96] και το UnQL [BDS95] - δεν περιέχουν σχήματα (επιτρέποντας οποιαδήποτε ετικέττα σε κόμβους ή ακμές, και μάλιστα όχι και στα δύο - όπως στο RDF). Ακόμα και τα ημιδομημένα συστήματα που παρέχουν δυνατότητες ανάθεσης τύπων (π.χ. αναγνωρίζοντας στιγμιότυπα προτύπων), όπως το YAT [CCS00], [CDSS98], δεν μπορούν να αξιοποιήσουν τις ιεραρχίες κλάσεων και ιδιοτήτων των σχημάτων RDF. Τέλος, τα σχήματα RDF έχουν σημαντικές διαφορές τόσο με τους ορισμούς τύπου εγγράφου XML (DTDs) [BPS98] όσο και με την πιο πρόσφατη πρόταση για σχήματα XML [TBMM00], [MM00]. Για παράδειγμα, οι σχέσεις επέκτασης (extension) και εκλέπτυνσης (refinement) τύπων στοιχείων XML χρησιμοποιούνται καθαρά για συντακτική διευκόλυνση, ενώ ένα στοιχείο δεν μπορεί να έχει περισσότερους από ένα τύπους. Κατά συνέπεια, οι γλώσσες επερώτησης που έχουν προταθεί για ημιδομημένα δεδομένα ή για δεδομένα XML (όπως οι Lorel [AQM+97], StruQL [FFK+97], XML-QL [DFF+99], XQL [Rob99], XML-GL [CCD+99] και Quilt [DC00]) δεν μπορούν να ερμηνεύσουν σωστά τη σημασιολογία των ετικεττών σε κόμβους και ακμές γράφων RDF. Το ίδιο ισχύει και για τις γλώσσες επερώτησης πληροφορίας σχήματος για βάσεις δεδομένων, όπως οι SchemaSQL [LSS96], XSQL [KKS92], Noodle [MR93], οι οποίες βασίζονται στο σχεσιακό ή στο οντοκεντρικό μοντέλο.

1.2 Αντικείμενο της εργασίας

Ορμώμενοι από τα παραπάνω ζητήματα, προτείνουμε ένα νέο μοντέλο δεδομένων και μια δηλωτική γλώσσα επερώτησης για περιγραφές και σχήματα RDF. Η γλώσσα μας, η *RQL* (RDF Query Language) - βασίζεται σε ένα τυπικό μοντέλο γράφων που εμπερικλείει τις βασικές έννοιες μοντελοποίησης του RDF και επιτρέπει την ερμηνεία περιγραφών RDF από ένα ή περισσότερα σχήματα. Αξίζει να σημειωθεί ότι - καθώς η προτυποποίηση του RDF Schema είναι ακόμη υπο εξέλιξη - η δουλειά μας αποσκοπεί να συνεισφέρει προς αυτή την κατεύθυνση. Η *RQL* προσαρμόζει τη λειτουργικότητα των γλωσσών επερώτησης ημιδομημένων εγγράφων [ABS99] στα ιδιαίτερα χαρακτηριστικά του RDF αλλά επίσης την επεκτείνει, προκειμένου να επιτρέψει την επερώτηση σχημάτων RDF. Η *RQL* μπορεί να θεωρηθεί ως ένα γενικής χρήσεως εργαλείο, που μπορεί να αποτελέσει τη βάση για τη δημιουργία διαφόρων εφαρμογών (π.χ. περιγραφή ή ανακάλυψη πόρων) που αποσκοπούν στη δημιουργία και στην προσαρμογή μιας πύλης κοινοτήτων διαδικτύου, σύμφωνα με τις ανάγκες και τα ενδιαφέροντα κάθε μέλους-χρήστη. Τέλος, η ύπαρξη πληροφορίας σχήματος ανοίγει ενδιαφέρουσες προοπτικές για την αποδοτική αποθήκευση περιγραφών RDF και τη βελτιστοποίηση της αποτίμησης των ερωτήσεων της RQL. Πιο συγκεκριμένα, η επιστημονική συνεισφορά αυτής της εργασίας είναι η εξής:

◇ Παρουσιάζουμε ένα μοντέλο γράφων το οποίο γεφυρώνει το χάσμα μεταξύ των προδιαγραφών RDF Model & Syntax [LS99] και RDF Schema [BG00]. Η πρωτοτυπία του μοντέλου μας συνίσταται στην αναπαράσταση των ιδιοτήτων ως *αυτόνομες* οντότητες, καθώς και στον ορισμό ενός μηχανισμού στιγμιοτύπων γράφων δεδομένων, που επιτρέπει την πολλαπλή κατηγοριοποίηση των πόρων. Συγκρινόμενο με την παρούσα κατάσταση του προτύπου, το μοντέλο μας προσφέρει ένα πιο πλούσιο σύστημα τύπων, που περιέχει βασικούς τύπους αλλά και τύπους ένωσης (union types).

◇ Προτείνουμε μια δηλωτική γλώσσα επερώτησης, την RQL, που επιτρέπει την επερώτηση περιγραφών αλλά και σχημάτων RDF. Η RQL υιοθετεί τη συναρτησιακή προσέγγιση της OQL [CB97] ενώ υποστηρίζει γενικευμένες εκφράσεις μονοπατιών, με μεταβλητές στις ετικέττες κόμβων (δηλ. σε κλάσεις) και ακμών (δηλ. τύπων ιδιοτήτων). Η πρωτοτυπία της RQL είναι η δυνατότητα ομαλής μετάβασης μεταξύ ερωτήσεων σε σχήματα και σε περιγραφές πόρων, καθώς και η εκμετάλλευση - με διαφανή τρόπο - των ιεραρχιών υπαλληλίας μεταξύ των ετικεττών. Η RQL είναι η πρώτη γλώσσα που παρέχει αυτή τη λειτουργικότητα.

1.3 Οργάνωση της εργασίας

Η εργασία αυτή έχει οργανωθεί ως εξής:

◇ Στο κεφάλαιο 2 θα παρουσιάσουμε τις βασικές έννοιες του μοντέλου δεδομένων των RDF/RDFS, τονίζοντας τα ιδιαίτερα χαρακτηριστικά του που το καθιστούν κατάλληλο για την περίπτωσή μας και οδηγούν στην ανάγκη της δημιουργίας μιας νέας γλώσσας επερώτησης.

◇ Στο κεφάλαιο 3, θα κάνουμε μια περιγραφή του μοντέλο που σχεδιάσαμε για να αναπαραστήσουμε το RDF, και στο οποίο βασίζεται η γλώσσα επερώτησής μας. Στη συνέχεια, θα αναλύσουμε τη λειτουργικότητα της γλώσσας μας με τη χρήση ενδεικτικών παραδειγμάτων. Στο τέλος του κεφαλαίου, θα περιγράψουμε την τυπική ερμηνεία των διαφόρων κατηγοριών ερωτήσεων της RQL καθώς και το σύστημα τύπων.

◇ Στο κεφάλαιο 4 θα εξηγήσουμε την υλοποίηση της γλώσσας μας, πάνω από ένα οντοκεντρικό-σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων (object-relational DBMS) [SBM99a], παρουσιάζοντας τις διάφορες ενότητες του διερμηνέα ερωτήσεων της RQL. Επίσης, θα αναλύσουμε το *Περιβάλλον Προγραμματισμού Εφαρμογών Επερώτησης* (Query API) το οποίο αναπτύχθηκε για τη σύνδεση του διερμηνέα της γλώσσας με το Ο-Σ ΣΔΒΔ. Τέλος, θα εξηγήσουμε την αποτίμηση των ερωτήσεων της RQL μέσα από μερικές ενδεικτικές μεταφράσεις τέτοιων ερωτήσεων σε αντίστοιχες ερωτήσεις SQL3 [SBM99b].

◇ Στο κεφάλαιο 5 θα συγκρίνουμε τη γλώσσα μας με διάφορες σχετικές εργασίες και θα δείξουμε γιατί αυτές δεν επαρκούσαν, οδηγώντας στην ανάγκη τη δημιουργίας της RQL.

◇ Τέλος, στο κεφάλαιο 6 θα συνοψίσουμε την επιστημονική συνεισφορά αυτής της εργασίας, θα σχολιάσουμε την αποτελεσματικότητά της και θα σκιαγραφήσουμε κάποιες κατευθύνσεις για μελλοντικές επεκτάσεις.

Το πρότυπο RDF

Στο κεφάλαιο αυτό θα δώσουμε μία περιγραφή του RDF (Resource Description Framework - **Περιβάλλον Περιγραφής Πόρων**)[LS99], που αποτελεί πρότυπο του World Wide Web Consortium (W3C). Ο απώτερος στόχος του RDF είναι να ορίσει ένα μηχανισμό για την περιγραφή πληροφοριακών πόρων, ο οποίος είναι κατάλληλος για:

- ◇ Την περιγραφή πληροφοριακών πόρων (δηλ. αντικειμένων του φυσικού ή "δικτυακού" κόσμου, π.χ. δικτυακούς κόμβους), ανεξάρτητα από το συγκεκριμένο (σε κάθε περίπτωση) πεδίο εφαρμογής.
- ◇ Την αναπαράσταση της γνώσης σε ένα πεδίο εφαρμογής, όπως αυτή εκφράζεται μέσω εννοιών και ιδιοτήτων. Η γνώση αυτή θα πρέπει να μπορεί να αξιοποιηθεί για μια σημασιολογικά πλούσια περιγραφή πληροφοριακών πόρων.

Ο πυρήνας του RDF είναι ένα μοντέλο (κατευθυνόμενων) γράφων με ετικέττες, που μπορούν να χρησιμοποιηθούν για να περιγράψουν *σχέσεις* (relationships) μεταξύ πόρων ή γνωρίσματα πόρων (με λεκτικές τιμές), όπως θα δούμε στη συνέχεια του κεφαλαίου. Οι ετικέττες αυτές αφορούν κυρίως *"επώνυμες" ιδιότητες* (named properties) και *κλάσεις*. Το μοντέλο του RDF συνοδεύεται και από μία "γλώσσα" περιγραφής λεξιλογίων, το RDF Schema[BG00]. Η γλώσσα αυτή μπορεί να χρησιμοποιηθεί για τον ορισμό σχημάτων περιγραφής πληροφοριακών πόρων, που ορίζουν τη σημασιολογία των περιεχομένων των μεταδεδομένων για ένα συγκεκριμένο πεδίο εφαρμογής. Τα λεξιλόγια αυτά αποτελούνται (παρόμοια με τα οντοκεντρικά μοντέλα αναπαράστασης γνώσεων) από ιεραρχίες κλάσεων, που μπορούν να επεκταθούν μέσω της σχέσης εξειδίκευσης (isA), καθώς και από τύπους ιδιοτήτων που τις συσχετίζουν, οι οποίοι επίσης μπορούν να εξειδικεύτουν μέσω της αντίστοιχης σχέσης εξειδίκευσης ιδιοτήτων. Μετά από μια σύντομη περιγραφή της ιστορίας της δημιουργίας του RDF, η οποία ακολουθεί στην επόμενη ενότητα, θα προχωρήσουμε σε μια πιο εκτενή ανάλυση του μοντέλου του RDF, καθώς και των δυνατοτήτων που προσφέρει για την μοντελοποίηση και ανταλλαγή μεταδεδομένων, και κατά συνέπεια επερώτηση και ανάκτηση των πόρων που αυτά περιγράφουν,στον Παγκοσμιο Ιστό.

2.1 Ιστορία του RDF

Το RDF προτάθηκε από το World-Wide-Web Consortium (W3C) σαν επέκταση της Πλατφόρμας για Επιλογή Περιεχομένου στο Διαδίκτυο (PICS - Platform for Internet Content Selection[Mil96]), η οποία προτάθηκε, επίσης απο το W3C, το 1995. Το PICS είναι ένας μηχανισμός περιγραφής και ανταλλαγής πληροφορίας αξιολόγησης σελίδων στον Παγκόσμιο Ιστό, από ένα εξυπηρετητή (web server) σε εφαρμογές πελατών (web clients) όπως οι αναδιφυτές (browsers). Τέτοιες πληροφορίες μπορούσαν, π.χ. να αφορούν την καταλληλότητα του περιεχομένου κάποιων σελίδων για παιδία μικρής ηλικίας, και μπορούσαν να χρησιμοποιηθούν π.χ. από γονείς που ήθελαν να προφυλάξουν τα παιδία τους από πληροφορίες/σελίδες που θεωρούσαν ακατάλληλες για αυτά.

Βασισμένο σε αυτή την αρχική ιδέα, της ανταλλαγής πληροφορίας περιγραφής πόρων, η οποία είναι κατανοητή από τον υπολογιστή αλλά ταυτόχρονα και αναγνώσιμη από τον άνθρωπο, και με σκοπό να αντιμετωπίσει τις ευρύτερες ανάγκες της περιγραφής πόρων, το W3C δημιούργησε μία νέα ομάδα εργασίας. Έτσι, όπως συμβαίνει γενικότερα με όλα τα πρότυπα/συστάσεις που παράγονται από το W3C, μέλη από διάφορα ερευνητικά ιδρύματα και εταιρίες (μέλη του W3C) συνεργάστηκαν για να προτείνουν ένα κοινώς αποδεκτό πρότυπο για την περιγραφή πόρων στον Παγκόσμιο Ιστό. Οι βασικές επιρροές (με βάση την προέλευση των μελών της ομάδας εργασίας) στη δημιουργία του RDF προήλθαν από την κοινότητα προτυποποίησης του Παγκοσμίου Ίστου του W3C (με μέλη που είχαν ασχοληθεί με το PICS και τα HTML META-tags[Wei96]), την κοινότητα των ψηφιακών βιβλιοθηκών, την κοινότητα των (ημι-)δομημένων εγγράφων (SGML και κυρίως XML). Επίσης, η σύνθεση της ομάδας εργασίας περιλάμβανε μέλη και άλλων προσπαθειών για ορισμό προτύπων για μεταδεδομένα, όπως το Dublin Core[WMD95], το Warwick Framework[LLJ96] καθώς και πρότυπα που προτάθηκαν από εταιρίες, όπως το MCF[Guh96] της Netscape και το XMLData[LJM+98] της Microsoft.

2.2 Στόχοι του RDF

◇ Ανεξαρτησία (Independence): Πρέπει να παρέχεται η δυνατότητα σε οποιονδήποτε να ορίζει ιδιότητες ή κλάσεις (κατηγορίες) ή και να επαναχρησιμοποιεί ήδη ορισμένες με ένα αυστηρά ορισμένο - σημασιολογικά - τρόπο. Επίσης, οι εφαρμογές που θα βασίζονται στο RDF θα πρέπει να μην επηρεάζονται από την πιθανή μελλοντική εξέλιξη των σχημάτων συγκεκριμένων εφαρμογών.

◇ Ανταλλαγή/Διαμεταφορά (Interchange): Τα μεταδεδομένα RDF θα πρέπει να μεταφέρονται/ ανταλλάσονται εύκολα μεταξύ υπολογιστών και να αποθηκεύονται σε απλά συστήματα αρχείων είτε σε εξειδικευμένα συστήματα βάσεων δεδομένων.

2.3 Πόροι RDF

Το RDF επιτρέπει την περιγραφή πληροφοριακών πόρων που περιγράφονται από ένα URI (Uniform Resource Identifiers - Ομοιόμορφα Αναγνωριστικά Πόρων), όπως περιγράφεται στο [BLFIM98].

Τέτοιοι πόροι μπορεί να είναι:

◇ Δικτυακοί τόποι, όπως το http://www.ics.forth.gr

◇ Συγκεκριμένα έγγραφα/σελίδες (HTML, XML κλπ)

◇ Συγκεκριμένα τμήματα/στοιχεία τέτοιων εγγράφων, καθώς αυτά μπορούν να περιγραφούν με URIs, με την προσθήκη XPaths[CD99]/Xpointers[DJM00].

◇ Φυσικά αντικείμενα, τα οποία δεν είναι καν προσπελάσιμα από τον παγκόσμιο ιστό, π.χ. βιβλία, πρόσωπα, πόλεις κλπ.

Αξίζει να σημειωθεί ότι στο RDF τα URIs χρησιμοποιούνται με τρόπο λίγο διαφορετικό από αυτόν που περιγράφεται στο [BLFIM98]. Πιο συγκεκριμένα, το RDF χρησιμοποιεί τα λεγόμενα qualified URIs, δηλαδή URIs που μπορεί να περιέχουν και ένα αναγνωριστικό τμήματος (fragment identifier), το οποίο διαχωρίζεται από το υπόλοιπο URI με το χαρακτήρα '#'. Αυτό έρχεται σε αντίθεση με το [BLFIM98], όπου το '#' ορίζεται ότι επιστρέφει "μία ιδιότητα των δεδομένων που προκύπτει από μία ενέργεια ανάκτησης". Αντίθετα, στο RDF, κάθε qualified URI θεωρείται ότι χαρακτηρίζει/αναφέρεται σε ένα αυτόνομο πόρο. Στη συνέχεια αυτού του κειμένου θα χρησιμοποιήσουμε τον όρο "πόρος" αναφερόμενοι στην έννοια του πόρου, όπως αυτή θεωρείται στο RDF.

2.4 Έννοιες και ορολογία του RDF

Τα μεταδεδομένα RDF περιγράφουν σχέσεις μεταξύ πόρων (όπως περιγράφηκαν παραπάνω), οι οποίες παριστάνονται ως *επώνυμες ιδιότητες* (named properties). Οι ιδιότητες αυτές είναι εν μέρει παρόμοιες με τα γνωρίσματα αντικειμένων, όπως αυτά ορίζονται στα οντοκεντρικά μοντέλα. Με βάση αυτή τη θεώρηση, οι περιγραφές RDF μπορούν να θεωρηθούν ως απλά ζεύγη γνωρισμάτων-τιμών. Η βασική διαφορά είναι ότι, ενώ στα οντοκεντρικά μοντέλα τα γνωρίσματα χαρακτηρίζονται και από την κλάση στην οποία ορίζονται (και έχουν νόημα μόνο ως τμήμα αυτής), στο RDF τα γνωρίσματα είναι καθολικά και χαρακτηρίζονται μόνο από το αναγνωριστικό τους. Έτσι, αποτελούν αυτόνομες οντότητες (μπορούν να θεωρηθούν και αυτές ως πόροι), και θεωρούνται στο μοντέλο του RDF κι αυτές ως *πολίτες πρώτης κατηγορίας* (first-class citizens). Μια τέτοια ιδιότητα μπορεί να έχει ως πεδίο ορισμού της μία ή περισσότερες κλάσεις, και ως τιμή της μια (μοναδική) κλάση ή την ειδικής χρήσεως κλάση *Literal*, που χρησιμοποιείται για να περιγράψει λεκτικές σταθερές (literals), όπως απλές συμβολοσειρές ή τιμές άλλων βασικών τύπων δεδομένων XML[BPS98]. Μια τέτοια τριάδα (δύο πόρων ή ενός πόρου και μίας τιμής που σχετίζονται μέσω μιας επώνυμης ιδιότητας) ονομάζεται *πρόταση* (statement) και αποτελεί το δομικό στοιχείο του RDF. Μία πρόταση αναπαριστά μια ιδιότητα ενός πληροφοριακού πόρου, ουσιαστικά περιγράφει - εν μέρει - τον πόρο αυτό. Προτάσεις που περιγράφουν ιδιότητες του ίδιου πόρου μπορούν να ομαδοποιηθούν, αποτελώντας μία *περιγραφή* (description) του πόρου αυτού.

Το RDF παρέχει τρεις τρόπους για την περιγραφή ενός μοντέλου μεταδεδομένων:

◇ Οντοκεντρική προσέγγιση: με τη χρήση κατευθυνόμενων γράφων με ετικέττες, που αποτελούν την απεικόνιση του μοντέλου δεδομένων του RDF, με τρόπο ανεξάρτητο από το χρησιμοποιούμενο συντακτικό.

◇ Σχεσιακή προσέγγιση: απεικόνιση μέσω τριάδων (ειδική περίπτωση των πλειάδων του σχεσιακού μοντέλου), καθεμία από τις οποίες αντιστοιχεί με μία πρόταση της περιγραφής. Μία τέτοια τριάδα περιέχει το όνομα της ιδιότητας, το αναγνωριστικό του αντικειμένου που περιγράφεται και την αντίστοιχη τιμή της ιδιότητας.

◇ Σειριοποίηση: Η συντακτική αναπαράσταση του RDF, που βασίζεται στην XML, παρέχει τρόπους σειριοποίησης των μεταδεδομένων σε αρχεία, η μορφή των οποίων είναι κατανοητή από τον άνθρωπο αλλά κυρίως αξιοποιήσιμη από υπολογιστές/προγράμματα που μπορούν να τα διαχειριστούν και να τα ανταλλάξουν.

Δεδομένου ότι στόχος αυτής της εργασίας είναι η δημιουργία μιας γλώσσας επερωτήσεων συμβατής με το μοντέλο του RDF, το συντακτικό είναι δευτερεύουσας σημασίας για την ανάλυσή μας. Γι' αυτό το λόγο, στη συνέχεια του κεφαλαίου θα προχωρήσουμε σε μια πιο εκτενή ανάλυση των χαρακτηριστικών του μοντέλου του RDF, που έχουν ιδιαίτερη σημασία για να προσδιορίσουμε τις ανάγκες αλλά και τις δυνατότητες της γλώσσας επερώτησης. Για τις ανάγκες της παρουσίασης του μοντέλου, θα χρησιμοποιήσουμε την αναπαράσταση με χρήση κατευθυνόμενων γράφων με ετικέττες, καθώς είναι η πιο απλή και εκφραστική για ανθρώπινη κατανάλωση και επιδεικνύει πλήρως τα χαρακτηριστικά του μοντέλου δεδομένων του RDF. Στο τέλος του κεφαλαίου, για λόγους πληρότητας, θα επανέλθουμε σε μια πιο αναλυτική περιγραφή του συντακτικού του RDF.

2.4.1 Κατευθυνόμενοι γράφοι με ετικέττες

Αυτοί οι γράφοι ονομάζονται και *διαγράμματα κόμβων και ακμών* (nodes and arcs diagrams). Οι ακμές αναπαριστούν τις επώνυμες ιδιότητες ενώ οι κόμβοι που εμφανίζονται με ελλειψοειδές σχήμα αναπαριστούν πόρους. Ως ειδική περίπτωση, οι κόμβοι που εμφανίζονται ως παραλληλόγραμμα αναπαριστούν τιμές βασικών τύπων (literals). Δεδομένου ότι το πεδίο ορισμού μιας ιδιότητας είναι πάντα ένας πόρος ενώ η τιμή της είναι είτε ένας άλλος πόρος είτε μία τιμή βασικού τύπου, μία ακμή στο γράφο ξεκινάει πάντα από ένα ελλειψοειδή κόμβο και καταλήγει σε ένα ελλειψοειδή ή ένα παραλληλόγραμμο κόμβο, όπως φαίνεται στο σχήμα 2.1.

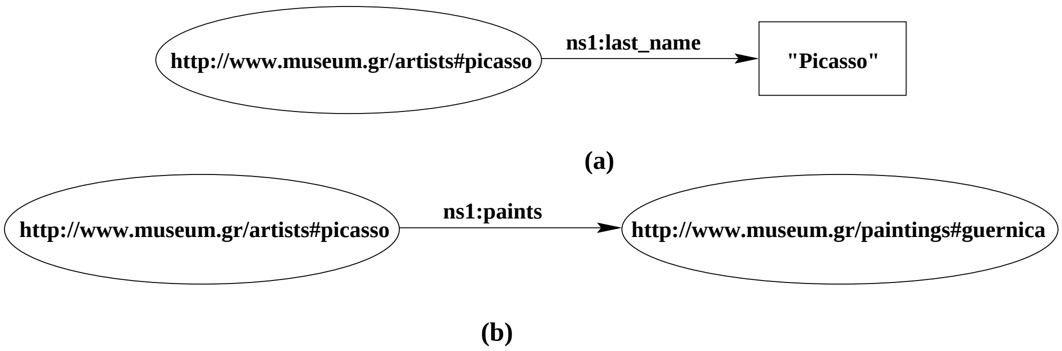
Ένας τέτοιος απλός γράφος δηλώνει ότι ο κόμβος `http://www.museum.gr/-artists#picasso` έχει μία ιδιότητα `http://www.culture.gr/schema.rdf#last-_name`, με τιμή `"Picasso"`. Πιο συγκεκριμένα, ο γράφος του σχήματος 2.1(a) αναπαριστά τη φράση (σε φυσική γλώσσα):

Το επώνυμο (ns1:last_name) του http://www.museum.gr/artists#picasso είναι "Picasso"

Στην ορολογία του RDF, αυτή η τριάδα (ένας πόρος με μία ιδιότητά του και την τιμή της) ονομάζεται *πρόταση* (statement). Τα τρία αυτά μέρη της πρότασης, ανάλογα με τη θέση τους σε αυτή, αποτελούν το υποκείμενο, το κατηγόρημα και το αντικείμενο της πρότασης.

Στην ίδια πρόταση χρησιμοποιήσαμε το πρόθεμα `ns1` για την ιδιότητα `last_name`, στη θέση του πλήρους URI `http://www.culture.gr/schema.rdf-#last_name`. Αυτή η συντομογραφία βασίζεται στη δυνατότητα ονοματοδοσίας της XML (XML Namespace Facility [BHL99]), η οποία επιτρέπει τον ορισμό προθεμάτων που υποδεικνύουν το έγγραφο στο οποίο έχει οριστεί το αντίστοιχο στοιχείο. Στην περίπτωση του RDF αυτό είναι ιδιαίτερα χρήσιμο, στον καθορισμό του σχήματος/-λεξιλογίου (RDF Schema) στο οποίο έχει οριστεί η αντίστοιχη ιδιότητα. Ένα σχήμα ουσιαστικά ορίζει "λέξεις" - και τη σημασία τους - οι οποίες μπορούν να χρησιμοποιηθούν στο σχηματισμό "προτάσεων". Όπως ακριβώς και στη φυσική γλώσσα, η σημασία της πρότασης εξαρτάται από τη σημασία των λέξεων που περιέχει, όπως αυτή καθορίζεται από το λεξιλόγιο που χρησιμοποιείται (π.χ. το λεξιλόγιο της ελληνικής γλώσσας, το λεξιλόγιο ορολογίας των επιστημόνων Η/Υ κλπ). Ένα σχήμα λοιπόν μπορεί να θεωρηθεί ως το αντίστοιχο ενός λεξικού, το οποίο περιέχει όλες τις λέξεις που μπορούμε να χρησιμοποιήσουμε καθώς και τη σημασία καθεμιάς από αυτές.

Είναι προφανές λοιπόν ότι για τη χρήση του RDF για την αναπαράσταση οποιασδήποτε μορφής γνώσης (π.χ. επιστημονικής γνώσης σε κάποιο τομέα κλπ) και την



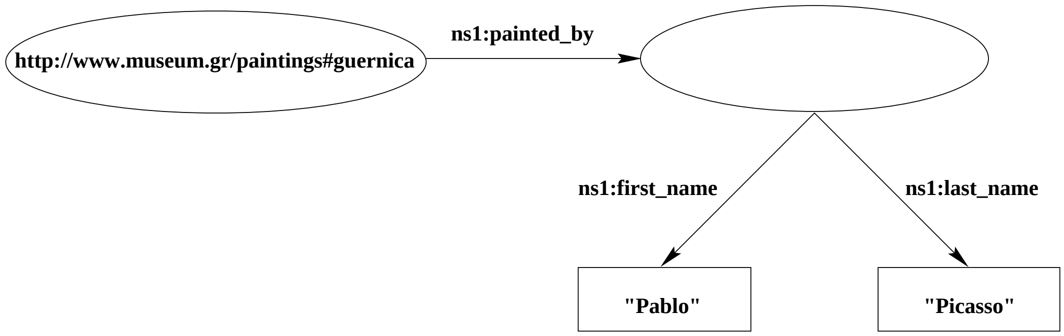
Σχήμα 2.1 Δύο απλά παραδείγματα γράφων RDF

χρήση και ανταλλαγή/διαμοιρασμό της με σημασιολογικά σαφή τρόπο, απαιτείται η ύπαρξη κοινώς αποδεκτών και σαφώς ορισμένων σχημάτων. Επίσης, είναι σαφές ότι η ύπαρξη τέτοιων σχημάτων παρέχει αυξημένες δυνατότητες για ακριβή και σημασιολογικά πλούσια επερώτηση των μεταδεδομένων (αξιοποιώντας τη διαμοιραζόμενη γνώση μιας "κοινότητας", όπως παριστάνεται σε ένα ή περισσότερα σχήματα). Στη συνέχεια του κεφαλαίου θα γίνει πιο εκτενής αναφορά στα βασικά συστατικά στοιχεία ορισμού τέτοιων σχημάτων, οι πλήρεις προδιαγραφές των οποίων περιγράφονται στο [BG00].

Έστω τώρα ότι θέλουμε να αναπαραστήσουμε πιο περίπλοκες φράσεις, όπως π.χ. ότι:

> *Το http://www.museum.gr/paintings#guernica ζωγραφίστηκε (ns1:painted_by) από κάποιον, που έχει όνομα (ns1:first_name) "Pablo" και επώνυμο (ns1:last_name) "Picasso"*

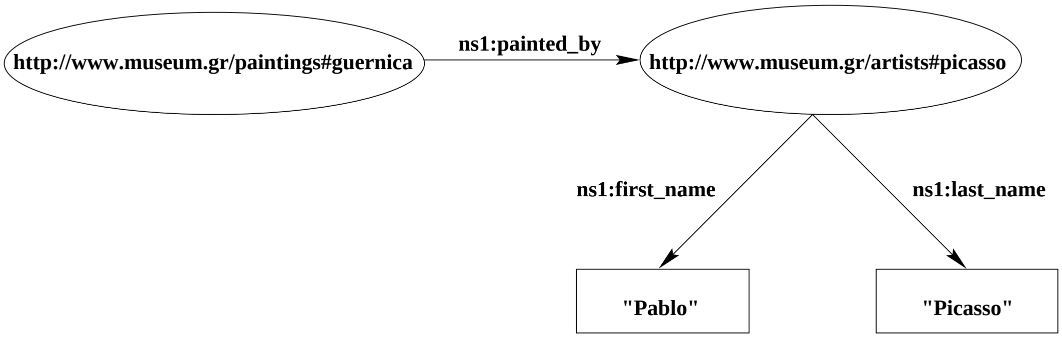
Αυτό σημαίνει πρακτικά ότι το αντικείμενο της πρώτης φράσης είναι μία δομημένη οντότητα, που έχει και άλλες ιδιότητες. Τέτοιες προτάσεις μπορούν να μοντελοποιηθούν εισάγοντας ένα ανώνυμο πόρο, που αναπαριστά αυτή την οντότητα. Ο γράφος που προκύπτει φαίνεται στο σχήμα 2.2



Σχήμα 2.2 Γράφος RDF με δομημένη τιμή

Μια τέτοια δομημένη οντότητα μπορεί επίσης να χαρακτηρίζεται από ένα αναγνωριστικό, π.χ. ένα URI, π.χ. `http://www.museum.gr/artists#picasso`, οπότε θα είχαμε την αναπαράσταση που φαίνεται στο σχήμα 2.3.

Στις επόμενες παραγράφους θα εξηγήσουμε τον τρόπο με τον οποίο αναπαριστούνται με τις δύο άλλες μορφές τέτοιες απλές περιγραφές RDF. Στο υπόλοιπο του κειμένου θα χρησιμοποιήσουμε αυτή την αναπαράσταση για πιο περίπλοκα παραδείγματα γράφων μεταδεδομένων και σχημάτων RDF.



Σχήμα 2.3 Δομημένη τιμή με αναγνωριστικό

2.5 Το μοντέλο δεδομένων του RDF

Μετά από αυτή τη γενική περιγραφή της φιλοσοφίας του RDF, θα προχωρήσουμε σε μια πιο εκτενή ανάλυση όλων των εννοιών που προτείνονται και αποτελούν το μοντέλο αυτό. Για το σκοπό αυτό θα χωρίσουμε τις βασικές έννοιες του RDF σε τρεις κατηγορίες: τις θεμελιώδεις έννοιες, τις έννοιες ορισμού σχημάτων (δηλ. τις βασικές έννοιες του RDF Schema, που μπορεί να χρησιμοποιηθεί για τον ορισμό σχημάτων για περιγραφές μεταδεδομένων RDF) και κάποιες επιπλέον έννοιες, οι οποίες δεν αποτελούν θεμελιώδεις λίθους του μοντέλου αλλά περιλαμβάνονται σε αυτό και μπορεί να φανούν χρήσιμες σε συγκεκριμένα πεδία εφαρμογής.

2.5.1 Θεμελιώδεις Έννοιες

2.5.1.1 rdfs:Resource

Η βασική ιδέα της χρήσης μεταδεδομένων (και ειδικότερα του RDF, όπως φαίνεται και από το όνομά του) είναι η περιγραφή πόρων (resource description). Με δεδομένο ότι - με βάση το [LS99] - οι πόροι χαρακτηρίζονται μοναδικά από ένα URI, και μπορούμε να αναθέσουμε ένα URI σε οτιδήποτε, συμπεραίνουμε ότι τα πάντα μπορούν να θεωρηθούν ως πόροι. Στην περίπτωση της περιγραφής πόρων στο πλαίσιο του Παγκοσμίου Ιστού είναι προφανές ότι ιδιαίτερο ενδιαφέρον παρουσιάζουν οι πόροι που αποτελούν τμήμα του και μπορούν να ανακτηθούν από το Διαδίκτυο.

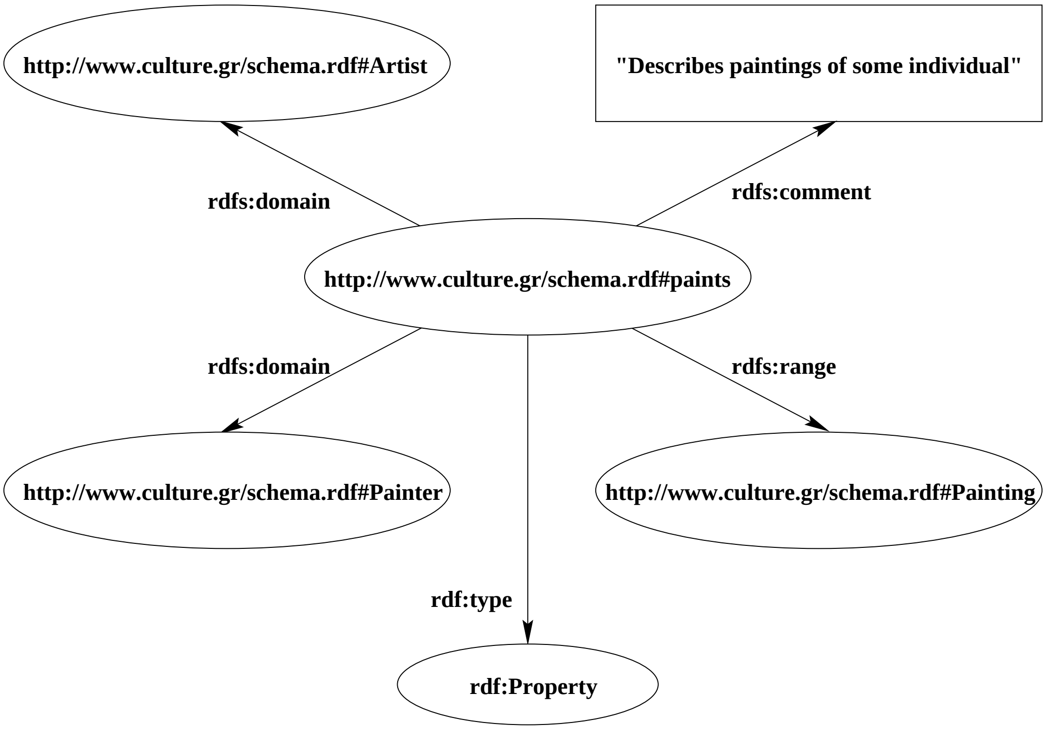
Πρέπει να σημειωθεί ότι οι πόροι αποτελούν μια εννοιολογική απεικόνιση των οντοτήτων στις οποίες αναφέρονται, και όχι σε κάποιο συγκεκριμένο στιγμιότυπο των οντοτήτων αυτών. Έτσι, για παράδειγμα, το URI "http://www.museum.gr/picasso" αναφέρεται σε ένα έγγραφο HTML στο Διαδίκτυο. Η πληροφορία για τον συγγραφέα αυτού του εγγράφου εξαρτάται από το περιεχόμενό του, ενώ το μέγεθός του είναι μια ιδιότητα που εξαρτάται από το συγκεκριμένο στιγμιότυπο του εγγράφου, και μπορεί να προσδιοριστεί τη στιγμή της ανάκτησης από τον εξυπηρετητή HTTP που θα κληθεί να ανακτήσει το έγγραφο. Η ερμηνεία των πόρων είναι διαφορετική στις δύο περιπτώσεις (αντικείμενο πληροφορίας/φυσική μορφή αρχείου). Κάτι τέτοιο ισχύει γενικά για την ερμηνεία των πόρων, το επίπεδο αφαίρεσης της οποίας εξαρτάται κάθε φορά από το σκοπό της συγκεκριμένης εφαρμογής, και πιθανώς αλλάζει με το χρόνο.

2.5.1.2 rdf:Property

Οι ιδιότητες είναι μια ειδική κατηγορία πόρων, όπως είδαμε νωρίτερα, οι οποίες χρησιμοποιούνται ως κατηγορήματα περιγραφών RDF. Ένα από τα πιο χαρακτηριστικά στοιχεία του RDF, σε σχέση με άλλες γλώσσες μοντελοποίησης όπου αποτελούν γνωρίσματα κλάσεων, είναι το γεγονός ότι οι ιδιότητες αποτελούν πολίτες πρώτης κατηγορίας. Αυτό γίνεται ακόμα πιο προφανές από το γεγονός ότι η ύπαρξη μίας τριάδας, που αποτελεί τη στοιχειώδη οντότητα πληροφορίας σε ένα μοντέλο RDF, οφείλεται στον ορισμό μιας ιδιότητας, η οποία ουσιαστικά περιγράφει και τη σημασιολογία της πρότασης. Η έννοια των κλάσεων εξακολουθεί να υφίσταται, όμως οι ιδιότητες μπορούν να οριστούν και να χρησιμοποιηθούν ανεξάρτητα από κλάσεις.

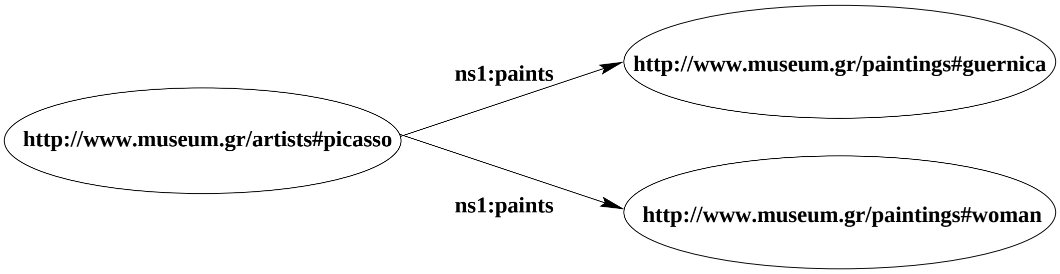
Ο χειρισμός των ιδιοτήτων ως αυτόνομων πόρων, με μοναδικά URIs, μας δίνει επίσης τη δυνατότητα να χρησιμοποιήσουμε το RDF και για την περιγραφή των ίδιων των ιδιοτήτων, γεγονός που συμβάλει σημαντικά στην απλότητα του μοντέλου και στο οποίο στηρίζεται ο "καλός" ορισμός των εννοιών που θα αναλυθούν στη συνέχεια. Δύο ιδιότητες που χαρακτηρίζουν τον ορισμό ενός τύπου ιδιοτήτων είναι το πεδίο ορισμού και το πεδίο τιμών τους. Οι ιδιότητες αναπαριστώνται από τις προκαθορισμένες ιδιότητες `rdfs:domain`, `rdfs:range`, που θα αναλυθούν περαιτέρω στη συνέχεια. Για παράδειγμα, στο σχήμα 2.4 φαίνεται το μοντέλο ενός ορισμού μιας ιδιότητας.

Παρόμοια με άλλα μοντέλα ημιδομημένων δεδομένων (όπως τα OEM [PGW95], YAT [CDSS98], UNQL[BDS95]), οι ιδιότητες στο RDF είναι *προαιρετικές* (optional) και πιθανώς πλειότιμες (multi-valued). Αυτό σημαίνει ότι για κάποιο πόρο που ανήκει στο πεδίο ορισμού μιας ιδιότητας, μπορεί είτε να μην οριστεί κάποιο στιγμιότυπο αυτής της ιδιότητας είτε μπορεί να οριστεί περισσότερες από μία φορές. Π.χ. στο σχήμα



Σχήμα 2.4 Ορισμός της ιδιότητας `paints`

2.5 η ιδιότητα `ns1:first_name` δεν ορίζεται ενώ η `ns1:paints` έχει δύο τιμές.



Σχήμα 2.5 Προεραιτικές και πλειότιμες ιδιότητες

Ως αποτέλεσμα αυτής της ιδιαιτερότητας του RDF, σε στιγμιότυπα μεταδεδομένων δεν εμφανίζονται μηδενικές τιμές (null values), όπως θα συνέβαινε π.χ. για τιμές που δεν έχουν οριστεί στο σχεσιακό μοντέλο. Αυτό το γεγονός παρέχει μεγαλύτερη ευελιξία στην αναπαράσταση αλλά και στην επερώτηση.

2.5.1.3 rdf:Statement

Μία πρόταση είναι ένας πόρος που υποστασιοποιεί (reify) μία τριάδα. Ένας τέτοιος πόρος έχει τουλάχιστον τρείς ιδιότητες, που ορίζονται από το μοντέλο του RDF: τις rdf:subject, rdf:object, rdf:predicate (υποκείμενο, αντικείμενο, κατηγόρημα), με τις αντίστοιχες τιμές από τα μέρη της τριάδας, όπως ορίστηκαν προηγουμένως. Επίσης, ένας τέτοιος πόρος έχει την ιδιότητα rdf:type (για την οποία θα μιλήσουμε πιο αναλυτικά στη συνέχεια), με τιμή rdf:Statement. Η υποστασιοποίηση (reification) των τριάδων αποτελεί θεμελιώδη έννοια στο μοντέλο του RDF, και μπορεί να χρησιμοποιηθεί για τον ορισμό του ίδιου του RDF με περιγραφές RDF.

Η θεώρηση των ίδιων των προτάσεων ως πόρων με αυτές τις ιδιότητες μας επιτρέπει να αναθέτουμε ιδιότητες και σε προτάσεις, είτε να θέτουμε προτάσεις ως τιμές ιδιοτήτων. Έτσι, για παράδειγμα, μπορούμε να εκφράσουμε "φράσεις" όπως:

Ο Gregory δήλωσε ότι: "ο Rodin ζωγράφισε τη Guernica"

Το μοντέλο του RDF που αναπαριστά αυτή την φράση φαίνεται στο σχήμα 2.6.

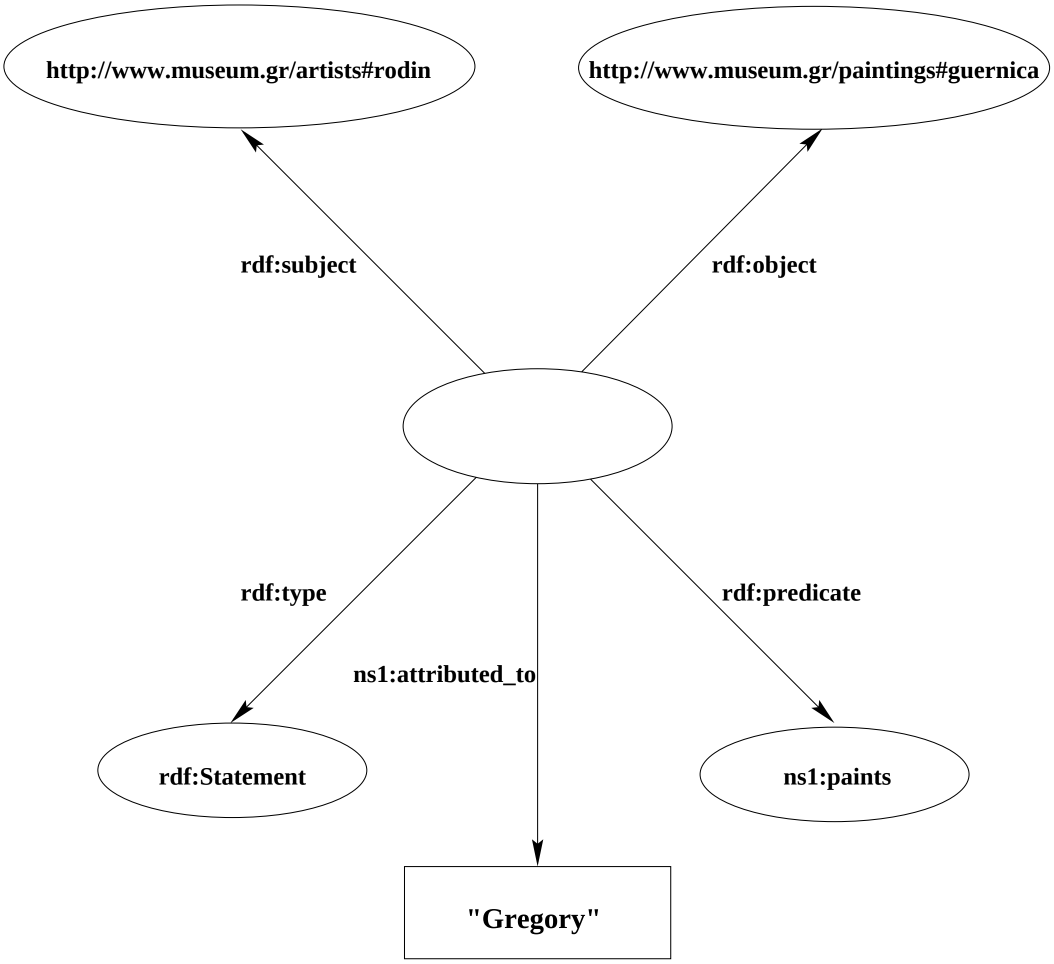
Ουσιαστικά, η δυνατότητα να σχηματίζουμε προτάσεις που έχουν ως μέλη τους άλλες προτάσεις αποτελεί ένα τρόπο για να εκφράζουμε προτάσεις *ανώτερης τάξης* (higher-order) στο *πρωτοβάθμιο* (first-order) μοντέλο του RDF. Μια σημαντική διαφορά αυτών των υποστασιοποιημένων προτάσεων, σε σχέση με τις "κοινές" προτάσεις που περιέχουν μόνο απλούς πόρους - και όχι άλλες προτάσεις, είναι η θεώρησή τους ως γεγονότα: πράγματι, ενώ μια απλή πρόταση όπως:

Ο Rodin ζωγράφισε τη Guernica

θα έπρεπε να εκληφθεί ως γεγονός (γνώση) - έστω κι αν είναι λανθασμένη, από την πρόταση του προηγούμενου παραδείγματος το μόνο γεγονός που μπορεί να εξαχθεί είναι ότι "ο Gregory έκανε τη συγκεκριμένη δήλωση". Για ένα σύστημα που διαχειρίζεται και επερωτά μεταδεδομένα RDF, δεν θα πρέπει σε καμμία περίπτωση η υποστασιοποιημένη πρόταση να εξαχθεί και να χρησιμοποιηθεί ως γεγονός.

2.5.2 Έννοιες ορισμού σχημάτων

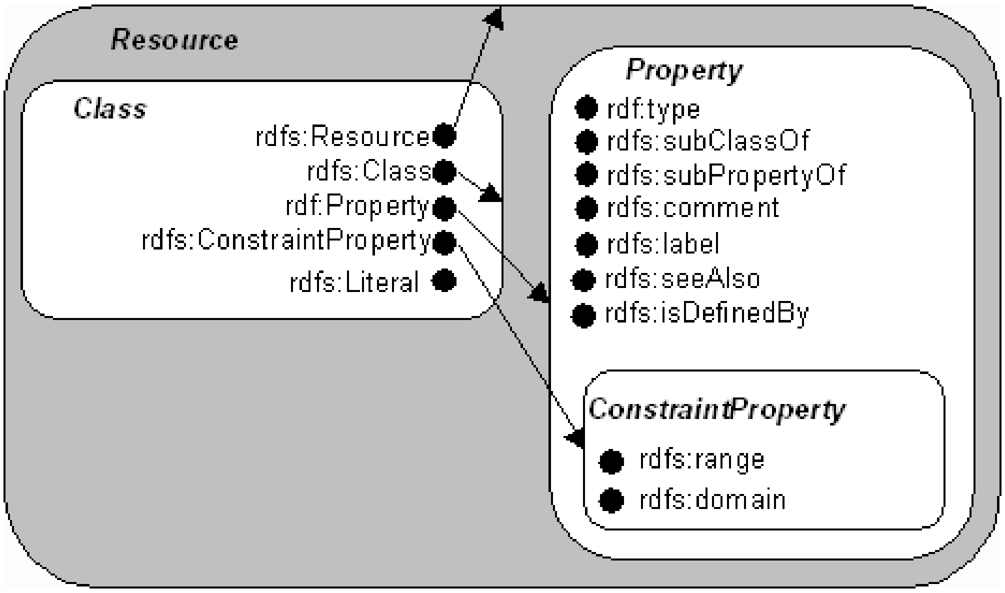
Οι έννοιες που παρουσιάστηκαν ως τώρα ορίστηκαν στο RDF Schema Specification[BG00], με σκοπό να περιγράψουν αυστηρά και με βάση ένα σύστημα τύπων, το



Σχήμα 2.6 Αναπαράσταση μιας υποστασιοποιημένης πρότασης

μοντέλο του RDF. Το RDF Schema προτάθηκε από το W3C για να επιτρέψει τον ορισμό λεξιλογίων για μεταδεδομένα RDF, τα οποία θα είναι επεκτάσιμα και θα μπορούν και τα ίδια να περιγραφούν με τη χρήση του RDF (με σκοπό τη διατήρηση της μεγαλύτερης δυνατής απλότητας και ομοιομορφίας). Λόγω της χρονικής απόστασης που μεσολάβησε ανάμεσα στη μονιμοποίηση των δύο προδιαγραφών και πιθανώς για να καλύψουν τις ανάγκες συγκεκριμένων εφαρμογών, έχουν εμφανιστεί κάποιες εφαρμογές που παραβλέπουν τελείως την αναπαράσταση και την εκμετάλλευση της πληροφορίας που υπάρχει σε σχήματα RDF. Τέτοιες εφαρμογές δεν έχουν ενδιαφέρον στην περίπτωσή μας καθώς χειρίζονται το RDF ως μια απλή περίπτωση ημιδομημένου μοντέλου δεδομένων [ABS99].

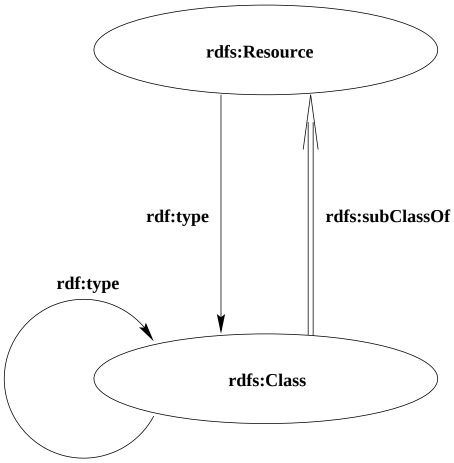
Ο ορισμός των σχημάτων RDF παρέχει τις απαραίτητες έννοιες για τον ορισμό συστημάτων τύπων (λεξιλογίων) για μεταδεδομένα. Στοιχεία αυτών των λεξιλογίων μπορούν να εξειδικευτούν - δημιουργώντας νέους τύπους. Επίσης, τα σχήματα μπορούν να χρησιμοποιηθούν για να περιορίσουν το πεδίο ορισμού ή τιμών διαφόρων ιδιοτήτων και γενικά για να εισάγουν περιορισμούς στα μεταδεδομένα, οδηγώντας στην ανάγκη για έλεγχο εγκυρότητας του ίδιου του σχήματος, ή των δεδομένων σε σχέση με το σχήμα το οποίο τα ερμηνεύει. Στο σχήμα 2.7 φαίνονται οι βασικές κλάσεις και πόροι του RDF Schema, που αποτελούν τις βασικές έννοιες των σχημάτων RDF. Στις επόμενες παραγράφους θα αναλυθεί η σημασιολογία αυτών των εννοιών, καθώς και οι περιορισμοί εγκυρότητας τους οποίους επιβάλλουν στα (μετα)δεδομένα που δημιουργούνται βάσει αυτών.



Σχήμα 2.7 Οι κλάσεις και οι πόροι του RDFS ως Σύνολα και Στοιχεία

2.5.2.1 rdfs:Class

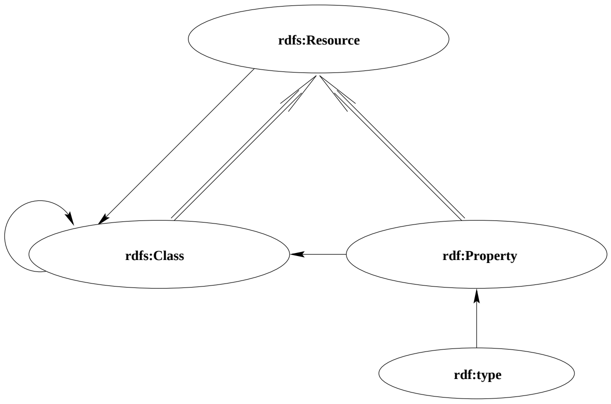
Όπως και στα περισσότερα συνηθισμένα οντοκεντρικά μοντέλα, έτσι και στο RDFS μια βασική έννοια είναι αυτή της *Κλάσης*, ή Τύπου ή Κατηγορίας. Μια κλάση μπορεί να αναπαριστά οτιδήποτε, από τύπους εγγράφων ως αφηρημένες έννοιες και όρους. Η διαφορά των κλάσεων στο RDFS και στα οντοκεντρικά μοντέλα είναι ότι στα τελευταία οι κλάσεις ορίζουν ένα τύπο για την *κατάσταση* (state) των - πιθανών - αντικειμένων τους, η οποία καθορίζεται από τις τιμές των διαφόρων γνωρισμάτων της κλάσης. Αντίθετα, στο RDFS όπου οι ιδιότητες αποτελούν αυτόνομες οντότητες, η κατηγοριοποίηση ενός πόρου σε μία κλάση ουσιαστικά αναθέτει στον πόρο ένα τύπο. Από τον τύπο αυτό - όπως θα δούμε στη συνέχεια - εξαρτάται το αν μπορεί ή όχι να δοθεί μια ορισμένη ιδιότητα σε ένα αντικείμενο αυτής της κλάσης. Ο ορισμός του `rdfs:Class` φαίνεται στο σχήμα 2.8.



Σχήμα 2.8 Δηλωση της κλάσης `rdfs:Class` στον ορισμό του RDFS

2.5.2.2 rdf:type

Η ιδιότητα αυτή χρησιμοποιείται για να δείξει ότι ένας πόρος είναι μέλος μίας κλάσης. Ορίζεται όπως κάθε άλλη επώνυμη ιδιότητα σε ένα σχήμα RDFS, όπως φαίνεται στο σχήμα 2.9. Όταν ένας πόρος έχει ως τιμή της ιδιότητας `rdf:type` του μία συγκεκριμένη κλάση, ο πόρος αυτός είναι ένα στιγμιότυπο της κλάσης αυτής. Για τον ορισμό μιας συγκεκριμένης κλάσης χρησιμοποιούμε ως `rdf:type` τον τύπο `rdfs:Class`. Οι ιδιότητες έχουν επίσης ένα ορισμένο τύπο, τον `rdf:Property`, ενώ οι υποστασιοποιημένες προτάσεις είναι του τύπου `rdf:Statement`. Ένας πόρος μπορεί επίσης να οριστεί ως αντικείμενο περισσότερων από μία κλάσεις, όχι αναγκαστικά συνδεδεμένες με τη σχέση εξειδίκευσης (πολλαπλή κατάταξη σε κλάσεις - multiple classification), δεν μπορεί όμως να οριστεί ταυτόχρονα ως κλάση και ιδιότητα. Στη συνέχεια θα δούμε πως η πληροφορία τύπου που δίνεται με την ιδιότητα `rdf:Property` μπορεί να επεκταθεί /κληρονομηθεί, με την οργάνωση των πόρων του σχήματος (κλάσεων ή τύπων ιδιοτήτων) σε ιεραρχίες υπαλληλίας.

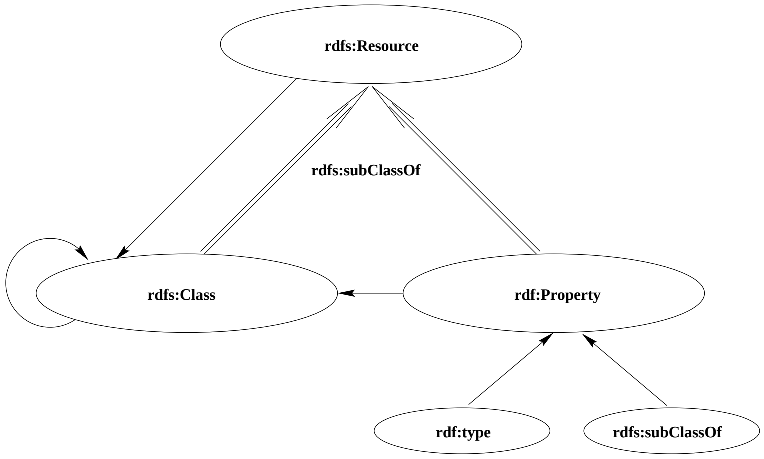


Σχήμα 2.9 Δηλωση της ιδιότητας `rdf:type` στον ορισμό του RDFS

2.5.2.3 rdfs:subClassOf

Αυτή η ιδιότητα καθορίζει μία σχέση υπαλληλίας μεταξύ κλάσεων, η οποία ορίζεται με παρόμοιο τρόπο με την `rdf:type`, όπως φαίνεται στο σχήμα 2.10. Η σχέση αυτή είναι μεταβατική: αν η κλάση Β είναι υποκλάση της Α, και η κλάση Γ είναι υποκλάση της Β, τότε μπορούμε να συμπεράνουμε αυτόματα ότι η Γ είναι υποκλάση και της Α. Αυτό σημαίνει ότι οι πόροι που είναι στιγμιότυπα της κλάσης Γ θα είναι επίσης στιγμιότυπα των Α και Β, αφού η κλάση Γ προσδιορίζει ένα υποσύνολο τόσο της Β όσο και της Α. Επίσης, με βάση τον ορισμό της ιδιότητας `rdfs:subClassOf`, μια κλάση μπορεί να είναι υποκλάση περισσότερων από μίας κλάσης (multiple inheritance) αλλά όχι του εαυτού της ή μιας κλάσης που έχει ήδη οριστεί ως υποκλάση της πρώτης. Κατά συνέπεια, με βάση την ιδιότητα `rdfs:subClassOf`, οι κλάσεις ενός σχήματος μπορούν να οργανωθούν σε μια ιεραρχία (χωρίς κύκλους),

Ο περιορισμός αυτός - της μη ύπαρξης κύκλων στη σχέση εξειδίκευσης - δεν μπορεί να εκφραστεί με κάποιο τρόπο στις προδιαγραφές του RDF Schema, αλλά επιβάλλεται ως κανόνας για τους δημιουργούς τέτοιων σχημάτων. Σε κάποιες περιπτώσεις, μπορεί να θεωρηθεί περιοριστικός (π.χ. δε μας επιτρέπει να χρησιμοποιήσουμε τη σχέση εξειδίκευσης κυκλικά για να δείξουμε ότι δύο κλάσεις είναι ισοδύναμες). Όμως, δεδομένου ότι μπορούμε ελεύθερα να ορίσουμε ξεχωριστές ιδιότητες για τέτοιες ειδικές περιπτώσεις, και δεδομένου ότι η ύπαρξη κύκλων θα πρόσεθετε μεγάλη πολυπλοκότητα (και πιθανώς θα απέκλειε σε κάποιες περιπτώσεις) λειτουργίες όπως αναδρομική διάσχιση ενός τέτοιου γράφου, υπολογισμού όλων των υποκλάσεων μίας κλάσης (transitive closure) κλπ, ένας τέτοιος περιορισμός μπορεί να θεωρηθεί χρήσι-



Σχήμα 2.10 Δηλωση της ιδιότητας `rdfs:subClassOf` στον ορισμό του RDFS

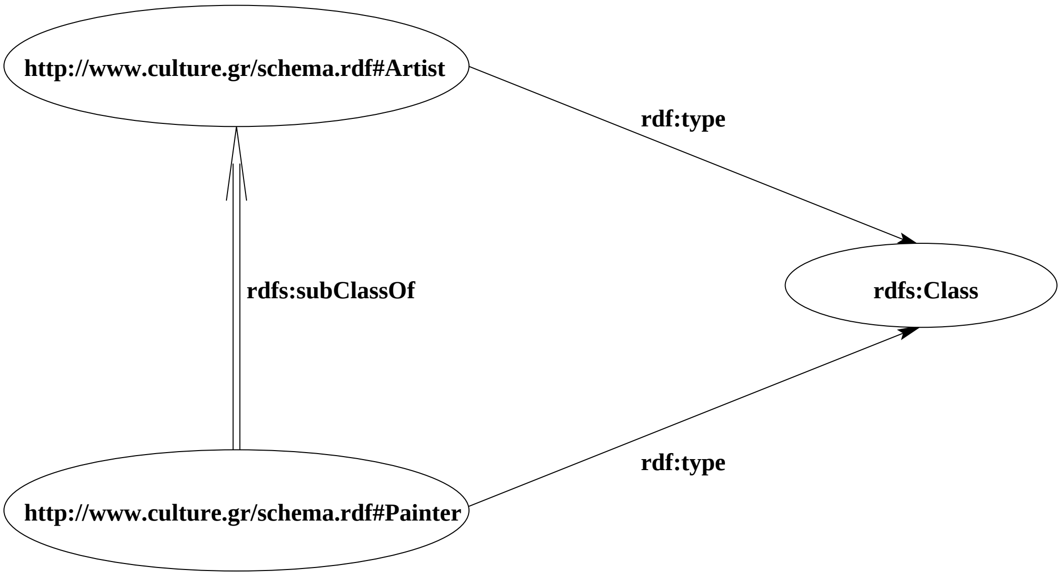
μος και επιθυμητός.

Η περίπτωση της πολλαπλής εξειδίκευσης (multiple inheritance) επίσης παρουσιάζει ιδιαίτερα χαρακτηριστικά, σε σχέση με το οντοκεντρικό μοντέλο. Δεδομένου ότι οι ιδιότητες είναι καθολικές (η εμβέλειά τους δεν περιορίζεται δηλαδή στο εσωτερικό μιας κλάσης), δεν υπάρχει περίπτωση η ταξινόμηση μιας κλάσης ως εξειδίκευσης δύο άλλων να προκαλέσει σύγκρουση ονομάτων (name conflicts), όπως θα συνέβαινε για δύο κλάσεις που έχουν ένα γνώρισμα με ίδιο όνομα, σε μια ανάλογη περίπτωση στο οντοκεντρικό μοντέλο. Περισσότερη συζήτηση για τις συνέπειες της καθολικότητας του ορισμού των ιδιοτήτων γίνεται στην παράγραφο 2.5.2.5.

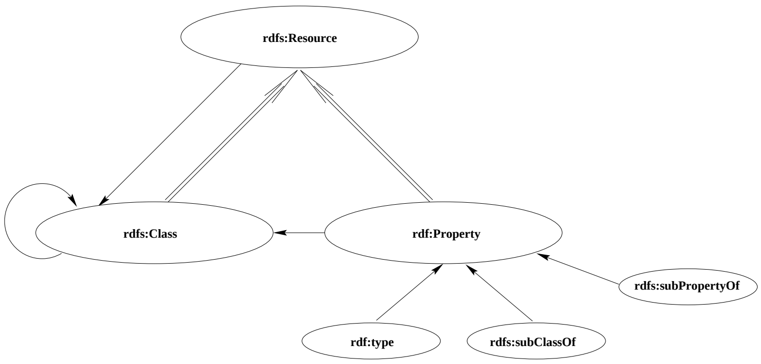
2.5.2.4 rdfs:subPropertyOf

Αντίστοιχα, η ιδιότητα `rdfs:subPropertyOf` (για τον ορισμό της βλ. σχήμα 2.12) προσδιορίζει το γεγονός ότι μια ιδιότητα είναι υπο-ιδιότητα μιας άλλης.

Όπως και η `rdfs:subClassOf` είναι και αυτή μεταβατική, δεν επιτρέπει το σχηματισμό κύκλων, ενώ μία ιδιότητα μπορεί να είναι υπο-ιδιότητα περισσότερων από μίας ιδιοτήτων. Αν π.χ. (βλέπε σχήμα 2.13) η ιδιότητα `paints` έχει οριστεί ως υπο-ιδιότητα της πιο γενικής `creates`, τότε αν σε ένα πόρο έχει οριστεί η `paints`, θα πρέπει να θεωρήσουμε ότι έχει οριστεί και η `creates`, με την ίδια τιμή. Αυτό πρακτικά σημαίνει ότι αν ζητήσουμε σε μία ερώτηση τις τιμές όλων των στιγμιοτύπων της `creates`, θα πρέπει να πάρουμε και τις τιμές των στιγμιοτύπων της `paints`. Επίσης, αν π.χ. μια εφαρμογή δε γνωρίζει τη σημασιολογία της `paints`, θα πρέπει τουλάχιστον να την



Σχήμα 2.11 Η κλάση Artist και η υποκλάση της, Painter

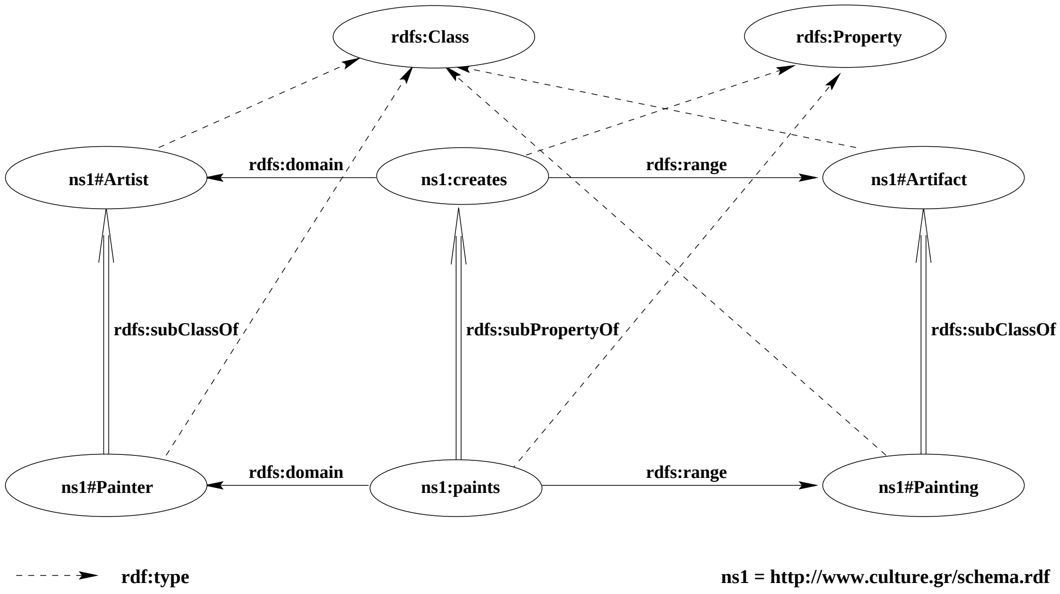


Σχήμα 2.12 Δηλωση της ιδιότητας `rdfs:subPropertyOf` στον ορισμό του RDFS

χειριστεί όπως την `creates`, με δεδομένο ότι γνωρίζει τη σημασιολογία της δεύτερης.

2.5.2.5 rdfs:domain και rdfs:range

Αυτές οι "ειδικής χρήσης" ιδιότητες χρησιμοποιούνται εφαρμόζονται στον ορισμό τύπων ιδιοτήτων και παίρνουν τιμές τύπου `rdfs:Class`. Χρησιμοποιούνται για τον περιορισμό των πόρων που μπορεί να έχουν αυτή την ιδιότητα (πεδίο ορισμού) καθώς και των αντίστοιχων έγκυρων τιμών γι' αυτή η ιδιότητα (πεδίο τιμών). Μία ιδιότητα μπορεί να έχει περισσότερες από μία κλάσεις στο πεδίο ορισμού της, αλλά μόνο μία στο πεδίο τιμών της. Κατά συνέπεια, για να είναι έγκυρη μία πρόταση (τριάδα) θα πρέπει το υποκείμενό της να ανήκει στο πεδίο ορισμού του κατηγορήματος (σε μία

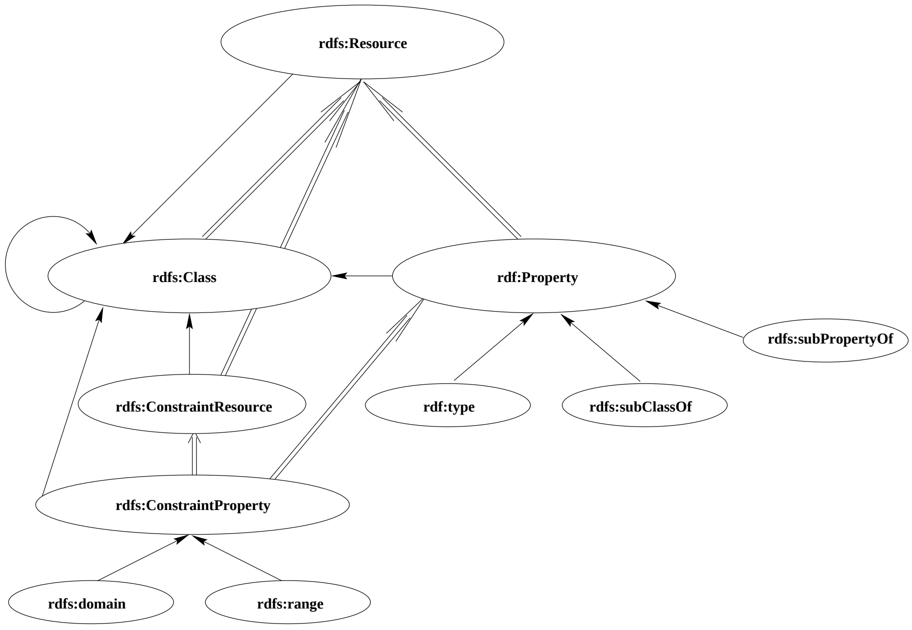


Σχήμα 2.13 Η σχέση creates και η υποσχέση της, paints

από τις κλάσεις που περιέχει, ή σε κάποια από τις υποκλάσεις αυτών), ενώ το αντικείμενο θα πρέπει να ανήκει στην κλάση που ορίζει το πεδίο τιμών (ή σε κάποια από τις υποκλάσεις της).

Για τον ορισμό αυτών των ιδιοτήτων, που χρησιμοποιούνται για να δηλώσουν *περιορισμούς* (constraints), ορίζονται στο RDFS δύο "ειδικής χρήσεως" κλάσεις, οι `rdfs:ConstraintResource` και `rdfs:ConstraintProperty`. Το σύνολο που περιγράφεται από την κλάση `rdfs:ConstraintResource` χρησιμοποιείται για την αναπαράσταση περιορισμών εγκυρότητας. Η υποκλάση του με όνομα `rdfs:ConstraintProperty` μοντελοποιεί ιδιότητες που αναπαριστούν τέτοιους περιορισμούς. Οι ιδιότητες `rdf:domain`, `rdf:range` είναι στιγμιότυπα αυτής της ιδιότητας, όπως φαίνεται στο σχήμα 2.14.

Πρακτικά, αυτή η σύμβαση πηγάζει από τη βασική ιδέα του μοντέλου του RDF, τη σχεδίαση με βάση τις ιδιότητες και όχι τα αντικείμενα/κλάσεις, και το γεγονός ότι ο ορισμός τους είναι καθολικός, για ένα σχήμα. Έστω, για παράδειγμα, ότι έχουμε ορίσει δύο κλάσεις `Artist` και `Museum`, οι οποίες δε συνδέονται μεταξύ τους μέσω της σχέσης εξειδίκευσης, και θέλουμε να ορίσουμε και στις δύο μία ιδιότητα `name`, που αναπαριστά το όνομά του αντίστοιχου αντικειμένου. Αν επιτρεπόταν να έχουμε μόνο μία κλάση στο πεδίο ορισμού της ιδιότητας, οι δύο αυτοί ορισμοί θα έρχονταν σε



Σχήμα 2.14 Δηλωση των ιδιοτήτων `rdfs:domain`, `rdfs:range` στον ορισμό του RDFS

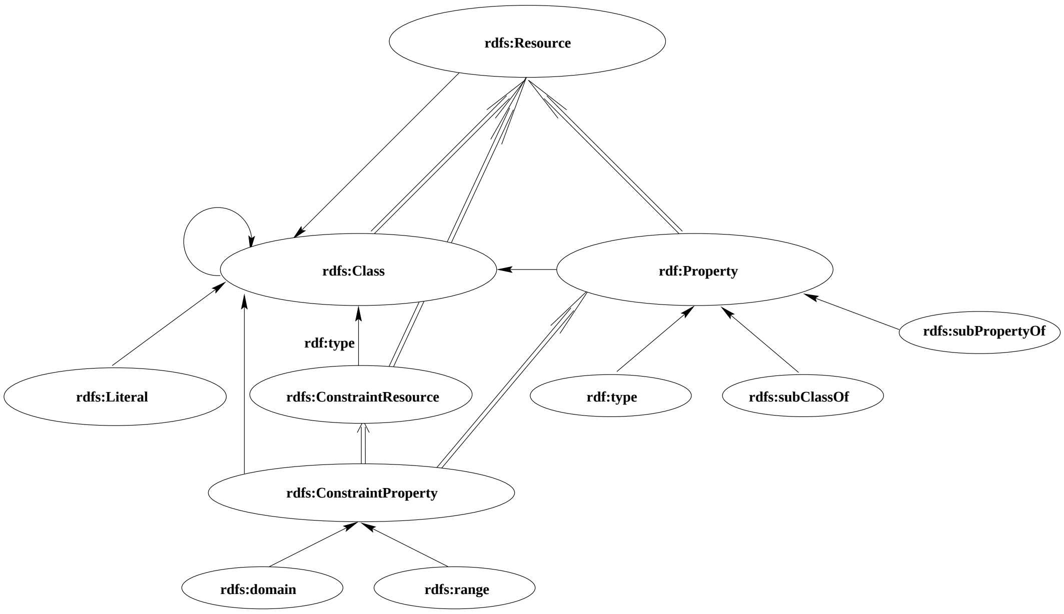
σύγκρουση (αν ορίζονταν στο ίδιο σχήμα). Ουσιαστικά λοιπόν, δίνοντας πολλαπλές τιμές στην ιδιότητα `rdfs:domain` ορίζουμε το πεδίο ορισμού της ιδιότητας ως την ένωση των διαφόρων τιμών/κλάσεων.

Μια λογική απορία που πηγάζει από τα παραπάνω είναι γιατί δεν επιτρέπεται κάτι αντίστοιχο και για την ιδιότητα `rdfs:range`, που υποδεικνύει το πεδίο τιμών της ιδιότητας. Αν και κάτι τέτοιο θα ήταν λογικό σε ένα οντοκεντρικό μοντέλο, όπου ο ορισμός κάθε ιδιότητας έχει εμβέλεια μόνο μέσα στην κλάση που ορίζεται, στην περίπτωση του RDF θα εισήγαγε ασάφειες σχετικά με τη σημασιολογία των ιδιοτήτων. Έστω, π.χ., στο σχήμα του προηγούμενου παραδείγματος ότι το όνομα (`name`) ενός `Person` είχε τιμή τύπου `HumanName` (μια δομημένη οντότητα που περιέχει π.χ. όνομα, επώνυμο, πατρώνυμο κλπ), ενώ για το `Object` η αντίστοιχη τιμή ήταν λεκτική. Θέτοντας ως πεδίο ορισμού το `Person` ∪ `Object` και πεδίο τιμών το `HumanName` ∪ `Literal`, θα μπορούσαμε φυσιολογικά να ορίσουμε, π.χ. μία ιδιότητα `name` σε ένα `Person` με τιμή `Literal`, πράγμα το οποίο δεν είναι συμβατό με την σημασιολογία την οποία αρχικά επιθυμούσαμε να εκφράσουμε με το σχήμα αυτό.

Ένα άλλο "προβληματικό" σημείο αποτελεί η δυνατότητα να παραλείψουμε τον ορισμό του πεδίου τιμών, στον ορισμό μιας ιδιότητας. Πράγματι, δεδομένου ότι το `rdfs:range` αποτελεί κι αυτό μια κοινή ιδιότητα για το μοντέλο του RDF, μπορεί να είναι προαιρετικό, όπως και οι άλλες ιδιότητες. Με βάση τις προδιαγραφές του RDF Schema, κάτι τέτοιο σημαίνει ότι η ιδιότητα μπορεί να πάρει οποιαδήποτε τιμή τύπου `rdfs:Resource` είτε `rdfs:Literal`, πράγμα ασύμβατο με τον ίδιο τον ορισμό του `rdfs:range`, καθώς οι δύο αυτές κλάσεις δε συνδέονται - προφανώς - με τη σχέση εξειδίκευσης. Αναφορικά με τα πεδία ορισμού και τιμών δύο ιδιοτήτων που συνδέονται με τη σχέση εξειδίκευσης, για να είναι έγκυρη η σχέση θα πρέπει τα πεδία ορισμού και τιμών της υπο-ιδιότητας να είναι υποσύνολα (ή ίσα) των αντίστοιχων πεδίων της υπερ-ιδιότητας. Για παράδειγμα, ο ορισμός του σχήματος 2.13 είναι έγκυρος, καθώς το πεδίο ορισμού της υπο-ιδιότητας (`Painter`) είναι υποκλάση του αντίστοιχου της υπερ-ιδιότητας (`Artist`), και το ίδιο ισχύει με το σύνολο τιμών.

2.5.2.6 rdfs:Literal

Παρ'όλο που με βάση το [LS99] οι τιμές βασικών τύπων δεν αποτελούν πόρους, για λόγους ομοιομορφίας του ορισμού σχημάτων ορίζεται η κλάση `rdfs:Literal`(βλ. σχήμα 2.15), για να περικλείσει όλες τις τιμές αυτής της μορφής. Έτσι, το πεδίο τιμών μίας ιδιότητας μπορεί να είναι και το `rdfs:Literal`, κάτι που είναι συμβατό με τον ορισμό του `rdfs:range` που παίρνει μόνο τιμές τύπου `rdfs:Class`.



Σχήμα 2.15 Η κλάση `rdfs:Literal` στον ορισμό του RDFS

2.5.3 Επιπλέον έννοιες

2.5.3.1 rdfs:Container

Η κλάση `rdfs:Container` χρησιμοποιείται για την αναπαράσταση συλλογών από πόρους. Το RDF ορίζει τρεις τύπους συλλογών, που αναπαριστώνται από υποκλάσεις της `rdfs:Container`:

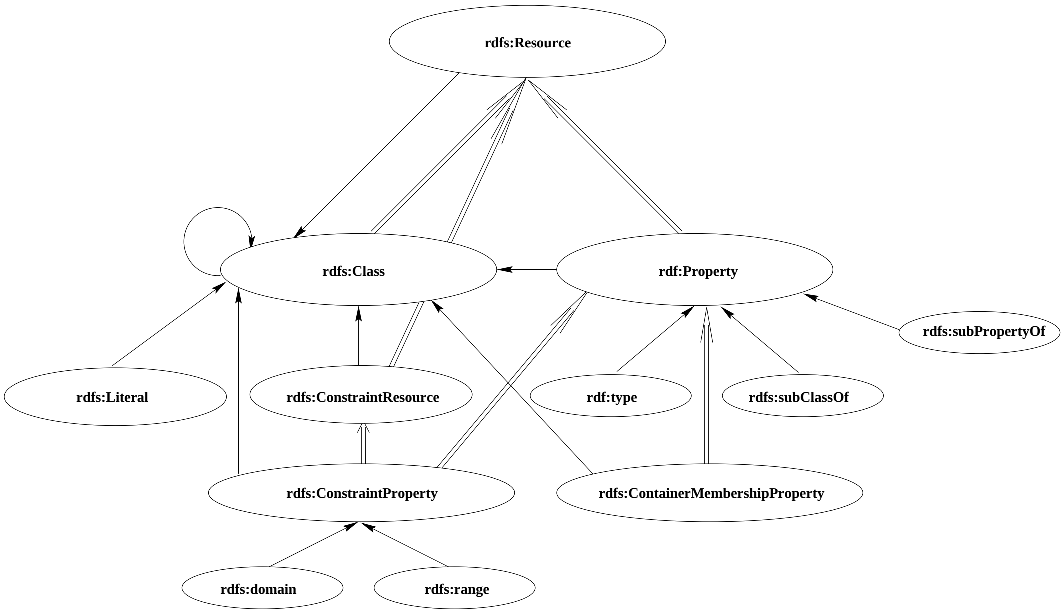
- ◇ `Bag`: Ένα πολυσύνολο από πόρους ή λεκτικές τιμές, χωρίς διάταξη. Τα Bags χρησιμοποιούνται συνήθως για ιδιότητες που παίρνουν πολλές τιμές, και η σειρά των τιμών δεν έχει σημασία. Το Bag δεν επιβάλει περιορισμούς μοναδικότητας για τα μέλη του (οπότε δεν μπορεί να χρησιμοποιηθεί για την αναπαράσταση συνόλων). Περιγράφεται από την κλάση `rdf:Bag`.

- ◇ `Sequence`: Μια λίστα από πόρους ή λεκτικές τιμές. Χρησιμοποιείται για ιδιότητες που παίρνουν πολλές τιμές, όπου όμως η σειρά τους είναι σημαντική. Μπορεί, π.χ. να χρησιμοποιηθεί για να διατηρήσει μια αλφαβητικά ταξινομημένη λίστα τιμών. Επιτρέπει επίσης επαναλαμβανόμενες εγγραφές. Περιγράφεται από την κλάση `rdf:Seq`.

- ◇ `Alternative`: Μια λίστα από πόρους ή λεκτικές τιμές, που αναπαριστά εναλλακτικές τιμές για την μοναδική τιμή μιας ιδιότητας. Για παράδειγμα, μια τέτοια συλλογή μπορεί να χρησιμοποιηθεί για την τιμή μιας ιδιότητας, που είναι μεταφρασμένη σε πολλές γλώσσες. Περιγράφεται από την κλάση `rdf:Alt`.

Ένα χαρακτηριστικό των τριών αυτών τύπων συλλογών είναι ότι δεν επιβάλλουν κάποιο περιορισμό τύπων στα μέλη τους. Έτσι, μια συλλογή μπορεί να περιέχει ταυτόχρονα πόρους και λεκτικές τιμές, καθώς και πόρους που έχουν κατηγοριοποιηθεί σε "ασύμβατες" κλάσεις.

Η ιδιότητα μέλους (membership property) που συνδέει ένα πόρο ή μια λεκτική τιμή με τη συλλογή που το περιέχει μοντελοποιείται (με τρόπο ομοιόμορφο με το υπόλοιπο μοντέλο του RDF) με την αυτόματη δημιουργία ιδιοτήτων της μορφής `rdf:_1`, `rdf:_2`, `rdf:_3` κλπ. Αυτές οι ειδικές ιδιότητες είναι στιγμιότυπα του τύπου ιδιότητας `rdfs:ContainerMembershipProperty`, μίας υποκλάσης του `rdf:Property` (βλ. σχήμα 2.15). Οι συλλογές, όπως και όλοι οι άλλοι πόροι στο RDF, μπορούν επίσης να έχουν κι άλλες ιδιότητες, αλλά υποχρεωτικά έχουν ως `rdf:type` ένα τύπο εκ των `rdf:Bag`, `rdf:Seq`, `rdf:Alt`.



Σχήμα 2.16 Η κλάση `rdfs:ContainerMembershipProperty`, υποκλάση της `rdf:Property`, στον ορισμό του RDFS

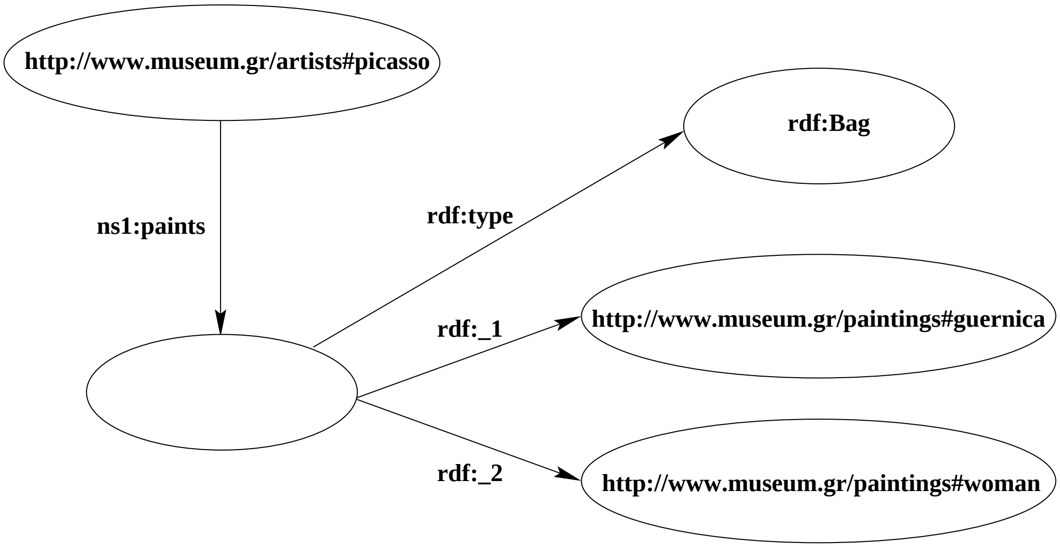
Χρησιμοποιώντας λοιπόν συλλογές, μπορούμε να εκφράσουμε σε RDF την παρακάτω πρόταση, με το μοντέλο RDF που φαίνεται στο σχήμα 2.17.

Τα έργα που ζωγράφισε ο Picasso είναι τα Guernica και Woman

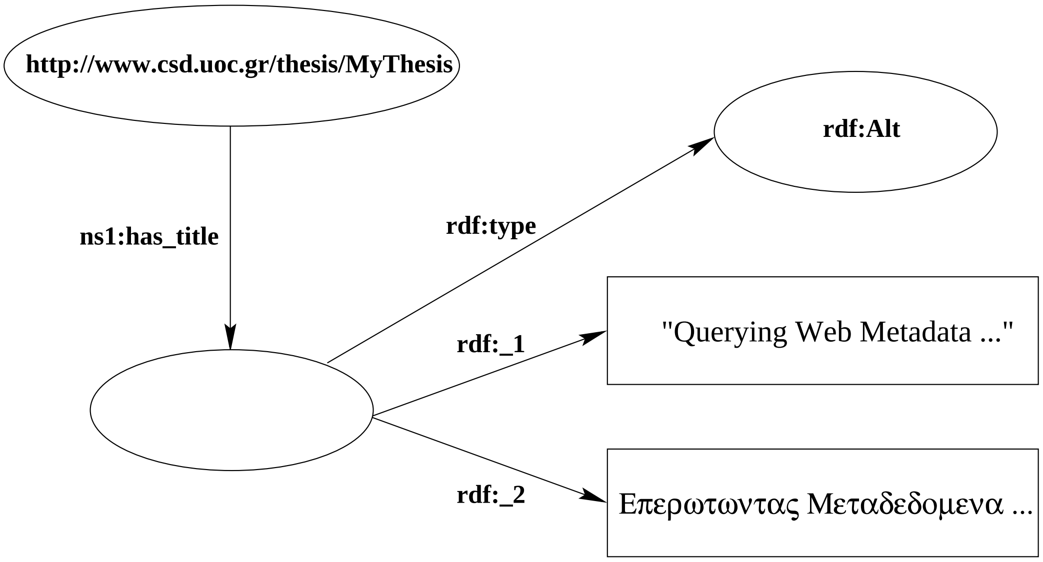
Όμοια, μπορούμε να χρησιμοποιήσουμε τη συλλογή `rdf:Alt` (σχήμα 2.18), για να δηλώσουμε ότι:

Ο τίτλος αυτής της εργασίας, στα αγγλικά και στα ελληνικά είναι "..."

Όπως φαίνεται στο σχήμα, και στην περίπτωση του `rdf:Alt` η αρίθμηση των περιεχομένων γίνεται με τα `rdf:_1`, `rdf:_2` κλπ. Η επιλογή της κατάλληλης εναλλακτικής τιμής κατά περίπτωση (π.χ. της κατάλληλης γλώσσας) είναι καθήκον της εφαρμογής, που πρέπει να γνωρίζει τη "θέση" κάθε εναλλακτικής τιμής. Ουσιαστικά, οι συλλογές αυτού του τύπου χρησιμοποιούνται για να μοντελοποιήσουν τους *απαριθμητούς τύπους δεδομένων* (enumerated types), που συναντάμε σε πολλές γλώσσες προγραμματισμού.



Σχήμα 2.17 Χρήση του τύπου συλλογής Bag

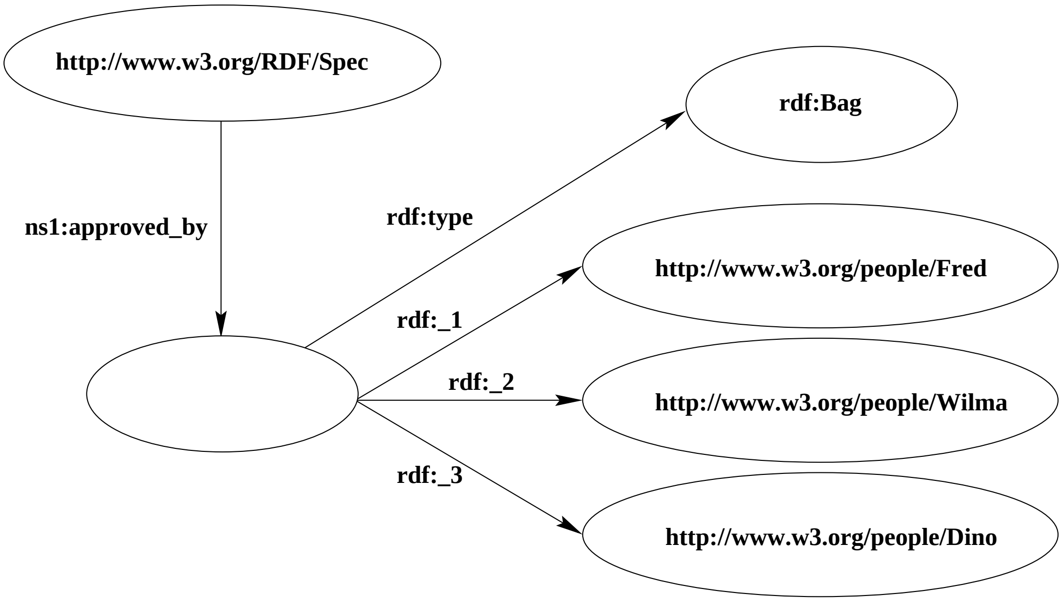


Σχήμα 2.18 Χρήση του τύπου συλλογής Alt, για πολυγλωσσικές τιμές

Αν και παρουσιάσαμε τις συλλογές ώς "επιπλέον" έννοια, υπάρχουν περιπτώσεις που η χρήση τους είναι αναγκαία. Ένα παράδειγμα είναι η παραπάνω "πολυγλωσσική" συλλογή, χωρίς την προκαθορισμένη σειρά γλωσσών της οποίας μια εφαρμογή δε θα μπορούσε να επιλέξει την κατάλληλη γλώσσα κάτα περίπτωση (π.χ. αν χρησιμοποιούσαμε επαναλαμβανόμενες ιδιότητες). Όμως, ούτε τα Bags είναι απόλυτα ισοδύναμα με τις επαναλαμβανόμενες ιδιότητες σε κάποιες περιπτώσεις. Για παράδειγμα, στο σχήμα 2.17 το Bag χρησιμοποιείται για να αναφερθεί συνολικά στα έργα του Picasso. Δεδομένου ότι η ιδιότητα `ns1:paints` αναφέρεται έτσι κι αλλιώς σε κάθε έργο χωριστά, θα μπορούσαμε να χρησιμοποιήσουμε και επαναλαμβανόμενες ιδιότητες. Έστω όμως η παρακάτω πρόταση:

Η επιτροπή που αποτελείται από τους Fred, Wilma και Dino ενέκρινε το ICOM/CIDOC για προτυποποίηση.

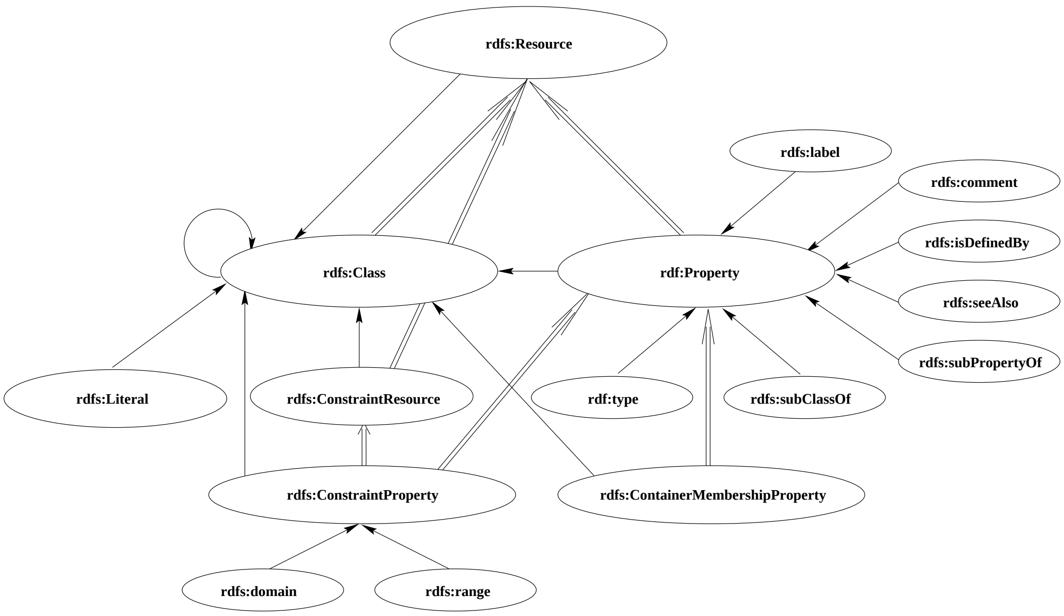
Στην περίπτωση αυτή, δεν γνωρίζουμε συγκεκριμένα ποιά από τα μέλη της επιτροπής ενέκριναν το πρότυπο. Έτσι, η χρήση επαναλαμβανόμενων ιδιοτήτων (που θα σήμαινε ότι καθένα από τα μέλη της επιτροπής ενέκρινε το ICOM/CIDOC[DC98]) θα απέδιδε λανθασμένα τη σημασιολογία της παραπάνω πρότασης. Αντίθετα, με τη χρήση του Bag αποδίδουμε την πραγματική σημασία της πρότασης, όπως φαίνεται στο σχήμα 2.19.



Σχήμα 2.19 Χρήση του τύπου συλλογής Bag

Το RDF παρέχει επίσης τρόπους για να αναθέσουμε ιδιότητες σε καθένα από τα μέλη μιας συλλογής (και όχι μόνο στη συλλογή καθεαυτήν). Έτσι, έχουμε τις λεγόμενες *κατανεμημένες αναφορές*, που διευκολύνουν σε κάποιες περιπτώσεις τη διαδικασία περιγραφής πόρων. Για παράδειγμα, δημιουργώντας μεταδεδομένα για την περιγραφή έργων του Picasso, είναι ευκολότερο να δημιουργήσουμε ένα Bag με τα έργα του και με μία κατανεμημένη αναφορά να δηλώσουμε ότι καθένα από τα μέλη της συλλογής δημιουργήθηκε από τον Picasso, αντί να κάνουμε την ίδια δήλωση για κάθε έργο χωριστά. Οι αναφορές αυτές γίνονται στο RDF με τη χρήση του `rdf:aboutEach`.

Μια άλλη παρόμοια λειτουργικότητα είναι ο έμμεσος ορισμός συλλογών - με σκοπό τη χρήση κατανεμημένων αναφορών - μέσω αντιπαραβολής προτύπων αναγνωριστικών πόρων (URIs), με τη χρήση της αναφοράς `rdf:aboutEachPrefix`. Η αναφορά αυτή επιτρέπει την κατανεμημένη αναφορά σε ιδεατές συλλογές, βάσει του κοινού προθέματος του URI τους. Κατά συνέπεια επιτρέπει π.χ. δηλώσεις που αναφέρονται σε όλες τις σελίδες ενός τμήματος ενός Δικτυακού Τόπου. Βέβαια, μια τέτοια λειτουργία αποτελεί ουσιαστικά, όπως θα δείξουμε σε επόμενο κεφάλαιο, μια διαδικασία επερώτησης σε μεταδεδομένα που έχουν προαποθηκευτεί. Έτσι, δεν είναι δυνατόν να υλοποιηθεί από μια εφαρμογή που "αναγνωρίζει" απλά RDF, χωρίς δυνατότητες επερώτησης.



Σχήμα 2.20 Οι ιδιότητες `rdfs:seeAlso`, `rdfs:isDefinedBy`, `rdfs:label` και `rdfs:comment` στον ορισμό του RDFS

2.5.3.2 rdfs:seeAlso και rdfs:isDefinedBy

Ένας πόρος μπορεί να περιγράφεται σε περισσότερα από ένα "μέρη" στο Διαδίκτυο. Η προκαθορισμένη ιδιότητα `rdfs:seeAlso` (βλ. σχήμα 2.20) έχει σκοπό να υποδεικνύει τέτοιες εναλλακτικές περιγραφές. Η υπο-ιδιότητά της `rdfs:isDefinedBy` είναι έχει πιο εξειδικευμένο σκοπό: να υποδεικνύει την αρχή (authority - κάποιο άλλο πόρο ή φυσικό πρόσωπο κλπ), η οποία όρισε αυτή την πρόταση. Αυτό μπορεί να είναι ιδιαίτερα χρήσιμο στην περίπτωση ορισμού κλάσεων ή ιδιοτήτων (κατασκευών σχήματος), όπου μπορεί π.χ. να χρησιμοποιηθεί για να υποδείξει το σχήμα στο οποίο περιέχεται ο ορισμός. Σε μερικές περιπτώσεις, όπως η χρήση μιας ιδιότητας με ορισμένο πρόθεμα χώρου ονομάτων π.χ. `ns1:creator`, το URI του αντίστοιχου σχήματος είναι προφανές (ns1). Σε άλλες, όπως π.χ. σε μία δήλωση υποκλάσης, όπου χρησιμοποιούμε αναγκαστικά ολόκληρο το URI της υπερκλάσης στην πρόταση RDF, αυτή η πληροφορία δεν μπορεί να εξαχθεί, εκτός αν μοντελοποιηθεί από μία ιδιότητα, όπως η `rdfs:isDefinedBy`.

2.5.3.3 rdfs:label και rdfs:comment

Σε κάποιες περιπτώσεις είναι χρήσιμο να παρέχονται και αναλυτικές (λεκτικές) περιγραφές για πόρους με τρόπο που πιθανώς δεν είναι αξιοποιήσιμος από τον υπολογιστή αλλά είναι κατανοητός από τον άνθρωπο. Για παράδειγμα, όταν κάποιος επιθυμεί να χρησιμοποιήσει ή να επεκτείνει ένα σχήμα, τέτοιες περιγραφές θα τον βοηθούσαν στην κατανόηση της σημασιολογίας του ορισμού των στοιχείων του υπάρχοντος σχήματος. Πιο συγκεκριμένα, η ιδιότητα `rdfs:label` (βλ. σχήμα 2.20) χρησιμοποιείται για να αναθέσει ένα - κατανοητό από ανθρώπους - όνομα σε ένα πόρο, ενώ η `rdfs:comment` για μια πιο αναλυτική περιγραφή του. Με δεδομένο ότι οι ιδιότητες στο RDF μπορούν να έχουν πολλαπλές τιμές, οι ιδιότητες αυτές μπορούν να αποτιμηθούν και σε διάφορες γλώσσες για να καλύψουν τις ανάγκες χρηστών διαφορετικών εθνικοτήτων.

2.6 Αναπαράσταση RDF με τριάδες

Όπως αναφέρθηκε στην αρχή αυτού του κεφαλαίου, ένας άλλος τρόπος για την αναπαράσταση περιγραφών RDF είναι με τη χρήση τριάδων. Δεδομένου ότι η αναπαράσταση αυτή είναι απόλυτα ανάλογη με τον γράφο RDF (μία τριάδα για κάθε ακμή και τα άκρα της) δε θα επεκταθούμε σε αυτή την αναπαράσταση, αλλά θα δώσουμε μόνο ένα ενδεικτικό παράδειγμα, που αντιστοιχεί στο γράφο του σχήματος 2.1(b):

Κατηγόρημα (ιδιότητα)	[http://www.culture.gr/schema.rdf#paints]
Υποκείμενο (πόρος)	[http://www.museum.gr/picasso.html]
Αντικείμενο (πόρος)	[http://www.museum.gr/guernica]

ενώ για το σχήμα 2.1(a) η ισοδύναμη τριάδα θα ήταν:

Κατηγόρημα (ιδιότητα)	[http://www.culture.gr/schema.rdf#last_name]
Υποκείμενο (πόρος)	[http://www.museum.gr/picasso.html]
Αντικείμενο (λεκτική σταθερά)	"Picasso"

Στα παραπάνω παραδείγματα θεωρούμε ότι ένα URI μέσα σε αγκύλες συμβολίζει ένα πόρο, ενώ για π.χ. η συμβολοσειρά "Picasso" δηλώνει μία λεκτική σταθερά.

Η αναπαράσταση αυτή μας επιτρέπει να αποδώσουμε μια σχεσιακή ερμηνεία σε ημιδομημένα δεδομένα και σχήματα (καθώς οι τριάδες είναι απλά μια υποπερίπτωση των πλειάδων). Περισσότερη ανάλυση για την σχεσιακή αναπαράσταση ημιδομημένων δεδομένων και σχημάτων μπορούν να βρεθούν στο [ABS99].

2.7 Συντακτικό RDF/XML

Απαραίτητη προϋπόθεση για να είναι δυνατή η ανταλλαγή μεταδεδομένων RDF μεταξύ προγραμμάτων/υπολογιστών αποτελεί η ύπαρξη και ενός συντακτικού. Για το σκοπό αυτό επιλέχθηκε για την RDF ένα συντακτικό XML[BPS98], με δεδομένα τα πλεονεκτήματα της XML σχετικά με την δομή της και τη δυνατότητα ανάγνωσής της από ανθρώπους (σε αντίθεση με μια πιθανή κωδικοποιημένη δυαδική αναπαράσταση). Επίσης, η XML αποτελεί την ιδανική πλατφόρμα για την επίτευξη της καλύτερης δυνατής διαλειτουργικότητας, μεταξύ ανεξάρτητα ανεπτυγμένων εφαρμογών στον Παγκόσμιο Ιστό.

Το συντακτικό της RDF χρησιμοποιεί το συμβολισμό των χώρων ονομάτων ([BHL99]), όπου τα πλήρη ονόματα προκύπτουν από τη συνένωση του προθέματος (URI χώρου ονομάτων) με το όνομα του αντίστοιχου στοιχείου. Συγκεκριμένα, για τους χώρους ονομάτων των ορισμών των RDF και RDF Schema θα χρησιμοποιούμε στη συνέχεια τα προθέματα `rdf:` και `rdfs:`, όπου:

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

```
rdfs = "http://www.w3.org/2000/01/rdf-schema#"
```

Επίσης, όπου χρησιμοποιούνται άλλα προθέματα θα αναφέρονται σε άλλα σχήματα που έχουν κατασκευαστεί για μια συγκεκριμένη εφαρμογή.

2.7.1 Περιγραφές (Descriptions)

Ένα έγγραφο RDF αποτελείται από μία σειρά από *περιγραφές*. Κάθε περιγραφή αναφέρεται συνήθως σε ένα πόρο, και περιέχει μια σειρά από ιδιότητες του πόρου αυτού. Οι τιμές αυτών των ιδιοτήτων μπορεί να είναι είτε λεκτικές τιμές είτε άλλοι πόροι (στους οποίους αναφερόμαστε μέσω του URI τους) είτε άλλες περιγραφές (οπότε αναφερόμαστε στον πόρο που αυτές περιγράφουν).

Ως έγγραφα XML, τα μεταδεδομένα RDF περιέχονται σε ένα στοιχείο με το όνομα `rdf:RDF`. Αυτό το στοιχείο περιέχει μια σειρά από περιγραφές. Οι περιγραφές, ως στοιχείο XML, έχουν δύο προαιρετικά γνωρίσματα:

◇ `rdf:about`: Χρησιμοποιείται για να αναφερθεί/περιγράψει ένα *πόρο διαδικτύου* στον οποίο αναφέρεται μέσω ενός σχετικού ή απόλυτου URI. Για παράδειγμα:

```
<rdf:Description rdf:about="http://www.museum.gr/artists#picasso">
      <ns1:first_name>Pablo</ns1:first_name>
</rdf:Description>
```

◇ `rdf:ID`: Χρησιμοποιείται για να ορίσει ένα *πόρο περιγραφών*. Το URI του δημιουργούμενου πόρου αποτελείται από τη συνένωση του URI του εγγράφου στο οποίο περιέχεται η περιγραφή με την τιμή του `rdf:ID`, χρησιμοποιώντας και τον ειδικό χαρακτήρα/αναγνωριστικό τμήματος εγγράφου XML "#".

```
<rdf:Description rdf:ID="foobar">
      ...
</rdf:Description>
```

◇ Μια περιγραφή χωρίς κάποιο από τα παραπάνω γνωρίσματα αναφέρεται σε ένα *ανώνυμο* πόρο. Σε τέτοιες περιγραφές ανατίθενται (τυχαία) δημιουργούμενα (generated) URIs, τα οποία χρησιμοποιούνται για "εσωτερικές" αναφορές σε αυτά. Δεδομένου ότι αυτά τα URIs παράγονται τυχαία και δεν είναι μοναδικά αναγνωριστικά, δεν υπαρχει τρόπος αναφοράς σε τέτοιους πόρους από άλλες περιγραφές, πέρα από τα χαρακτηριστικά που τους δίνονται εσωτερικά (inline).

```
<rdf:Description>
    ...
</rdf:Description>
```

Μία περιγραφή περιέχει μια σειρά από στοιχεία XML. Αυτά τα στοιχεία ερμηνεύονται ως ιδιότητες, των οποίων τα URIs προκύπτουν από την επέκταση των ονομάτων των στοιχείων (με τη συνένωση του προθέματος και του ονόματος που αναφέρθηκε παραπάνω). Αν ένα στοιχείο/ιδιότητα είναι άδειο, πρέπει να έχει ορισμένο το γνώρισμα `rdf:resource`, η τιμή του οποίου είναι το URI του πόρου στον οποίο αναφέρεται η ιδιότητα αυτή. Αλλιώς, αν η ιδιότητα έχει λεκτική τιμή, το αντίστοιχο στοιχείο XML περιέχει απλό κείμενο (που μπορεί να ακολουθεί συντακτικό σήμανσης XML). Όλα αυτά τα χαρακτηριστικά επιδεικνύονται στο παράδειγμα εγγράφου RDF του σχήματος 2.21, που αποτελεί τη συντακτική αναπαράσταση του γράφου του σχήματος 2.22.

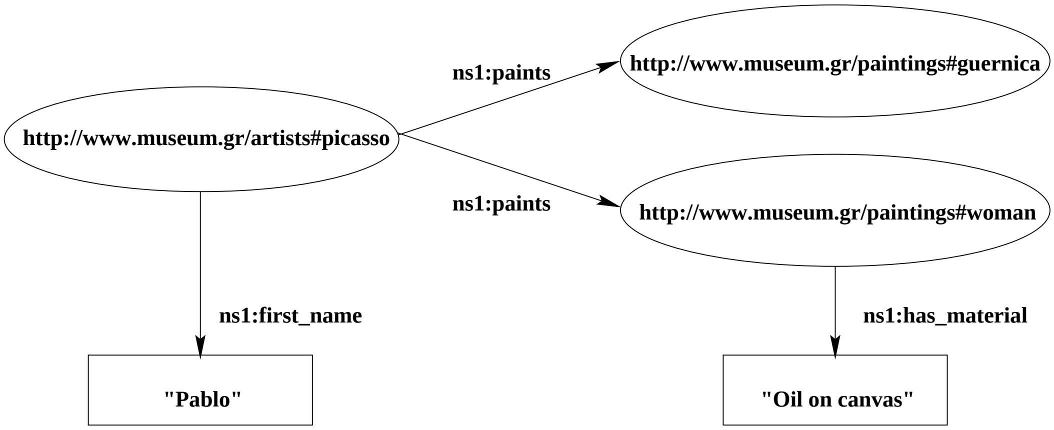
Σε περίπτωση αμφισημίας (όπου δεν μπορούμε να διακρίνουμε αν το περιεχόμενο ενός στοιχείου /ιδιότητας είναι πόρος ή λεκτική τιμή) μπορεί να χρησιμοποιηθεί το γνώρισμα `rdf:ParseType` με τιμή "Resource" ή "Literal" αντίστοιχα. Αυτό είναι ιδιαίτερα χρήσιμο στην περίπτωση που το περιεχόμενο περιέχει σημαδούρες αλλά θέλουμε αυτό να θεωρηθεί και να αναλυθεί ως λεκτική τιμή και όχι ως περιγραφή RDF (rdf:Description).

Οι προδιαγραφές που περιγράφονται στο [LS99] προτείνουν και δύο μετατροπές συντόμευσης στο παρακάτω συντακτικό, για να διευκολύνουν την "χειρωνακτική" συγγραφή του (χωρίς κατάλληλα εξειδικευμένα εργαλεία) αλλά και τον αυτόματο χειρισμό (μερικών) εγγράφων XML ως μεταδεδομένα RDF. Οι συντμήσεις αυτές είναι οι εξής:

- ◇ Το rdf:Description σε μια περιγραφή μπορεί να αντικατασταθεί από ένα URI. Σε αυτή την περίπτωση το URI ερμηνεύεται ως η τιμή της ιδιότητας `rdf:type` για τον περιγραφόμενο πόρο. Για παράδειγμα, η περιγραφή:

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ns1 = "http://www.culture.gr/schema.rdf#">

  <rdf:Description rdf:about =
          "http://www.museum.gr/artists#picasso">
    <ns1:first_name>Pablo</ns1:first_name>
    <ns1:paints rdf:resource =
          "http://www.museum.gr/paintings#guernica" />
    <ns1:paints>
      <rdf:Description rdf:about =
              "http://www.museum.gr/paintings#woman">
        <ns1:has_material>Oil on canvas</ns1:has_material>
      </rdf:Description>
    </ns1:paints>
  </rdf:Description>
</rdf:RDF>
```

Σχήμα 2.21 Συντακτική αναπαράσταση ενός απλού γράφου RDF



Σχήμα 2.22 Ένα ενδεικτικό παράδειγμα γράφου περιγραφών RDF

```
<rdf:Description rdf:about =
        "http://www.museum.gr/artists#picasso">
  <rdf:type rdf:resource =
        "http://www.culture.gr/schema.rdf#Painter" />
  <ns1:first_name>Pablo</ns1:first_name>
  <ns1:paints rdf:resource =
        "http://www.museum.gr/paintings#guernica" />
</rdf:Description>
```

όπου `ns1 = "http://www.culture.gr/schema.rdf#"`, θα μπορούσε να γραφεί ως:

```
<ns1:Painter rdf:about =
      "http://www.museum.gr/artists#picasso">
  <ns1:first_name>Pablo</ns1:first_name>
  <ns1:paints rdf:resource =
        "http://www.museum.gr/paintings#guernica" />
</ns1:Painter>
```

◇ Οι ιδιότητες με λεκτικές τιμές μπορούν να γραφούν ως γνωρίσματα του στοιχείου `rdf:Description` αντί ως περιεχόμενά του. Εφαρμόζοντας και αυτή τη σύντμηση στο παραπάνω παράδειγμα θα παίρναμε το παρακάτω έγγραφο RDF:

```
<ns1:Painter rdf:about = "http://www.museum.gr/artists#picasso"
```

```
        ns1:first_name = "Pablo">
    <ns1:paints rdf:resource =
        "http://www.museum.gr/paintings#guernica" />
  </ns1:Painter>
```

Το συντακτικό αυτό αρκεί για την αναπαράσταση των στοιχείων του μοντέλου που παρουσιάστηκε στην προηγούμενη ενότητα. Στις επόμενες παραγράφους θα γίνει ειδική αναφορά στο συντατικό περιγραφής συλλογών, για τις οποίες προτείνεται μια ειδική συντακτική αναπαράσταση, καθώς και της υποστασιοποίησης.

2.7.2 Συλλογές

Στο συντακτικό του RDF, οι συλλογές (rdf:Bag, rdf:Seq, rdf:Alt) μπορούν να χρησιμοποιηθούν σε όποιο σημείο μπορούν να χρησιμοποιηθούν περιγραφές (στοιχεία rdf:Description). Αυτό είναι λογικό, καθώς και οι περιγραφές δεν είναι τίποτε άλλο παρά συλλογές από ιδιότητες. Σε αντίθεση με τις περιγραφές, οι συλλογές δεν μπορούν να έχουν το γνώρισμα rdf:about, αλλά έχουν είτε το rdf:ID είτε είναι ανώνυμες.

Ειδικά για την περιγραφή των μελών μιας συλλογής, δεν χρησιμοποιούνται οι σειριακές ιδιότητες rdf:_1, rdf:_2 κλπ - στιγμιότυπα της ιδιότητας rdfs:ContainerMembershipProperty - αλλά η "ειδική" ιδιότητα rdf:li, η οποία πρέπει να χειρίζεται κατάλληλα από εφαρμογές που αναγνωρίζουν το συντακτικό του RDF. Έτσι, για παράδειγμα, η συλλογή που φαίνεται στο σχήμα 2.17 θα περιγραφόταν από το έγγραφο RDF του σχήματος 2.23 ενώ αυτή του σχήματος 2.18 από το έγγραφο 2.24.

```
<rdf:Bag rdf:ID = "Picasso's_Paintings">
  <rdf:li rdf:resource = "http://www.museum.gr/paintings#guernica" />
  <rdf:li rdf:resource = "http://www.museum.gr/paintings#woman" />
</rdf:Bag>
```

Σχήμα 2.23 Συντακτική αναπαράσταση μιας συλλογής τύπου Bag

Εξάλλου, για την κατανανεμημένη αναφορά στα μέλη μιας συλλογής χρησιμοποιείται (όπως εξηγήθηκε στην παράγραφο 2.5.3.1) το rdf:aboutEach ως γνώρισμα του στοιχείου που αντιπροσωπεύει τη συλλογή. Έτσι, η αναφορά που δόθηκε

```
<rdf:Alt rdf:ID = "Multilingual_Title">
  <rdf:li>Querying Web Metadata ...</rdf:li>
  <rdf:li>Επερωτώντας Μεταδεδομένα ...</rdf:li>
</rdf:Alt>
```

Σχήμα 2.24 Συντακτική αναπαράσταση μιας συλλογής τύπου Alt

ως παράδειγμα στην ίδια παράγραφο, ότι όλα τα περιεχόμενα της συλλογής έργων του Picasso (που δημιουργήσαμε στο προηγούμενο παράδειγμα) έχουν την ιδιότητα `ns1:created_by` με τιμή το URI που αναφέρεται στον Picasso, περιγράφεται από το έγγραφο του σχήματος 2.25

```
<rdf:Description rdf:aboutEach = "#Picasso's_Paintings">
  <ns1:created\_by rdf:resource = "http://www.museum.gr/artists#picasso" />
</rdf:Description>
```

Σχήμα 2.25 Χρήση του γνωρίσματος aboutEach, για κατανεμημένη αναφορά στα στοιχεία μιας συλλογής

2.7.3 Υποστασιοποίηση

Το συντακτικό του RDF παρέχει ένα επιπλέον τρόπο έκφρασης προτάσεων που αναφέρονται σε άλλες προτάσεις, οι οποίες δεν είναι αναγκαστικά υποστασιοποιημένες: προσθέτοντας ένα γνώρισμα `rdf:ID` στο στοιχείο της ιδιότητα της πρότασης. Έτσι, μια άλλη πρόταση μπορεί να αναφέρεται σε αυτή χρησιμοποιώντας την τιμή αυτού του ID. Επίσης, μπορούμε να αναφερθούμε και σε όλες τις προτάσεις που περιέχονται σε ένα στοιχείο `rdf:Description`, χρησιμοποιώντας το γνώρισμα `rdf:bagID` για να δώσουμε ένα αναγνωριστικό URI στο στοιχείο αυτό (και όχι στον πόρο στον οποίο αναφέρεται η περιγραφή, όπως θα γινόταν με το `rdf:ID`). Στη συνέχεια μπορούμε, π.χ. χρησιμοποιώντας το `rdf:aboutEach` με τιμή το αντίστοιχο bagID, να αναφερθούμε σε όλες τις προτάσεις που περιέχονται σε αυτή την συλλογή/περιγραφή. Οι δυνατότητες αυτές φαίνονται στο παράδειγμα εγγράφου του σχήματος 2.26 (που παριστάνεται από το γράφο του σχήματος 2.27).

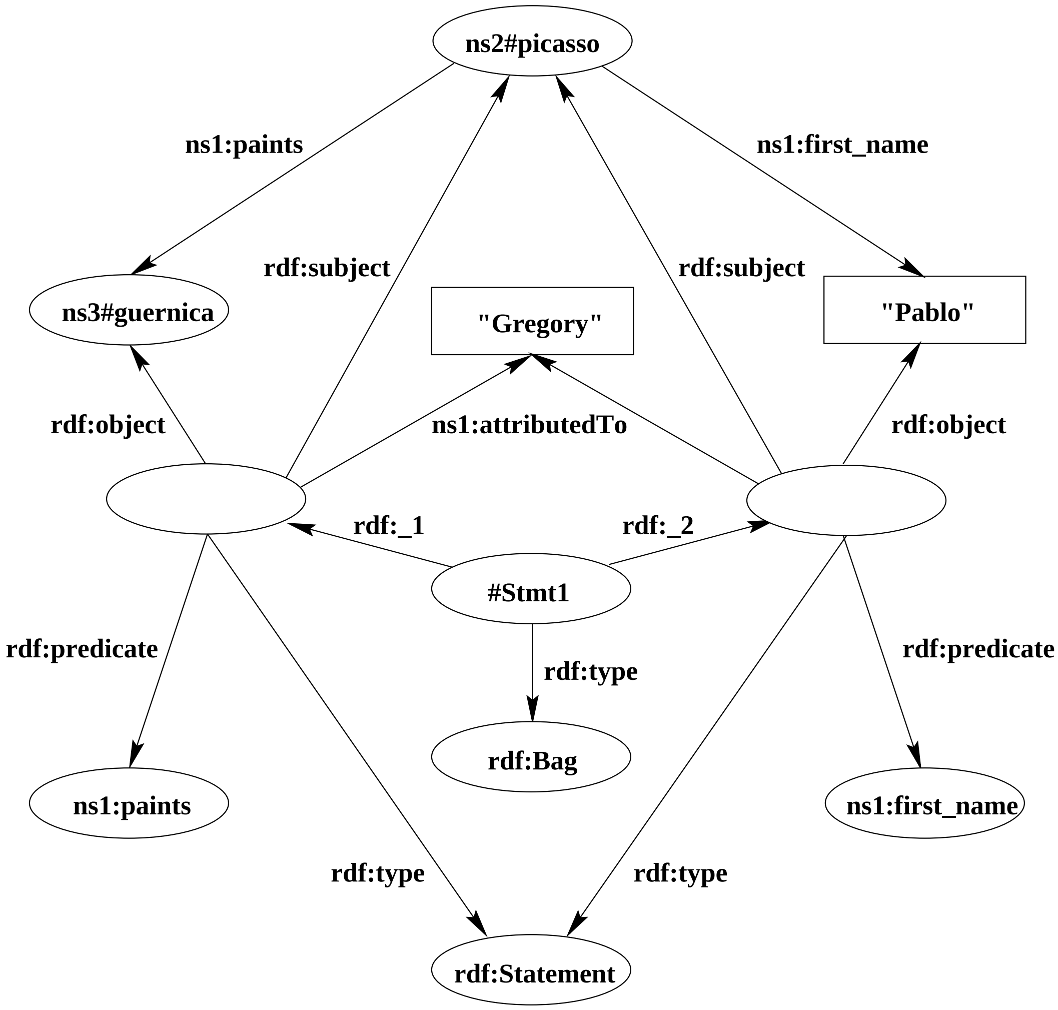
```
    <rdf:Description rdf:about =
            "http://www.museum.gr/artists#picasso" rdf:bagID = "Stmt1">
      <ns1:paints rdf:resource =
            "http://www.museum.gr/paintings#guernica" />
      <ns1:first_name>Pablo</ns1:first_name>
  </rdf:Description>

<rdf:Description rdf:about = "#Stmt1">
    <ns1:attributedTo>Gregory</ns1:attributedTo>
  </rdf:Description>
```

Σχήμα 2.26 Συντακτική περιγραφή προτάσεων υψηλότερης τάξης, με υποστασιοποίηση

2.7.4 RDF vs. XML

Δεδομένου ότι το RDF - προκειμένου να σειριοποιηθεί σε αρχεία - χρησιμοποιεί το συντακτικό της XML, πιθανώς δεν είναι προφανές - με μια πρώτη ματιά - γιατί η ίδια η XML δεν αρκεί για να αναπαραστήσει μεταδεδομένα RDF. Η σύγχυση αυτή προκύπτει αν δούμε το RDF ως συντακτικό και όχι ως μοντέλο, που στην πραγματικότητα είναι, το οποίο είναι εντελώς διαφορετικό από αυτό της XML. Για να γίνει αυτό πιο σαφές, θα επιχειρήσουμε μια σύντομη ανάλυση του μοντέλου της XML και στη συνέχεια θα δείξουμε τις διαφορές του με αυτό του RDF.

Η XML προτείνει ένα μοντέλο δέντρου, με ετικέττες μόνο στους κόμβους (ονόματα στοιχείων-σημαδούρων). Επιπλέον, με τη χρήση XML References ([DMOT00], [DJM00], [CD99]) είναι δυνατό ένα έγγραφο XML να περιγράψει ένα μοντέλο γράφου. Όπως αναλύεται εκτενέστερα στο [ABS99], ένα μοντέλο δέντρου XML μπορεί να μεταφραστεί σε ένα γράφο σύμφωνο με ένα μοντέλο ημιδομημένων δεδομένων (και αντίστροφα), καθώς τα τελευταία είναι μοντέλα γράφων με ετικέττες μόνο στις ακμές (οπότε η αντιστοίχηση κόμβων με ακμές μπορεί να γίνει εύκολα). Στην περίπτωση των γράφων XML, κάτι αντίστοιχο δεν είναι πάντα δυνατό, ειδικά δεδομένου ότι η ίδια πληροφορία μπορεί να αναπαρασταθεί με πολλούς διαφορετικούς τρόπους στην XML. Μάλιστα, το γεγονός ότι ενώ η XML επιβάλλει περιορισμούς όσον αφορά τη σειρά των στοιχείων, όπως θα δούμε παρακάτω, δεν συμβαίνει το ίδιο με τη σειρά των γνω-



Σχήμα 2.27 Υποστασιοποίηση των προτάσεων που περιέχονται σε ένα στοιχείο περιγραφής

ρισμάτων ενός στοιχείου (ενώ τελικά και τα δύο αναπαριστούνται με τον ίδιο τρόπο σε ένα δέντρο XML), περιπλέκει ακόμα περισσότερο την κατάσταση.

Αντίθετα, το RDF είναι ένα μοντέλο κατευθυνόμενων γράφων, με ετικέττες τόσο στους κόμβους (πόροι που έχουν ταξινομηθεί σε κάποια κλάση) όσο και στις ακμές του (επώνυμες ιδιότητες). Έτσι, δεν μπορεί να αναπαρασταθεί ούτε στο μοντέλο της XML (αλλά ούτε και σε κάποιο από τα μοντέλα ημιδομημένων δεδομένων). Επίσης, το RDF προτείνει και κάποια άλλα στοιχεία που δεν εμφανίζονται ούτε και σε άλλα μοντέλα γράφων, όπως είναι οι συλλογές και οι προτάσεις ανωτέρας τάξης.

Ένα άλλο σημαντικό χαρακτηριστικό του μοντέλου της XML είναι το γεγονός ότι θεωρεί σημαντική τη σειρά των στοιχείων σε ένα δέντρο/γράφο, καθώς επίσης και

```
<Painter rdf:about="http://www.museum.gr/picasso.htm"
         first_name="Pablo" last_name="Picasso">
     <paints rdf:resource="http://www.museum.gr/guernica.gif"/>
</Painter>
<Painting rdf:about="http://www.museum.gr/guernica.gif"/>
```

(a)

```
<Painter rdf:about="http://www.museum.gr/picasso.htm">
     <paints>
        <Painting rdf:about="http://www.museum.gr/guernica.gif"/>
     </paints>
     <first_name>Pablo</first_name>
     <last_name>Picasso</last_name>
</Painter>
```

(b)

Σχήμα 2.28 Διαφορετικές συνταχτικές αναπαραστάσεις της ίδιας πληροφορίας σε XML

τον τρόπο που είναι εμφωλευμένα (nested) τα στοιχεία αυτά. Έτσι, τα δύο έγγραφα που φαίνονται στο σχήμα 2.28 είναι διαφορετικά (επεξεργαζόμενα ως έγγραφα XML). Αυτή η επιλογή είναι λογική στην περίπτωση της XML, που προτάθηκε με σκοπό την περιγραφή εγγράφων. Όμως, σε εφαρμογές δεδομένων (π.χ. αποθήκευση και επερώτηση) η σειρά όχι μόνο δεν είναι σημαντική, αλλά αποτελεί και σημαντικό παράγοντα υποβάθμισης των επιδόσεών τους. Γι' αυτό και όλα τα συστήματα που χειρίζονται τέτοιου είδους δεδομένα δεν λαμβάνουν υπ'όψη την σειρά και το φώλιασμα. Εξάλλου, ακόμα και σε ειδικές λειτουργίες όπου η σειρά παίζει ρόλο (π.χ. ταξινόμηση αποτελεσμάτων μιας ερώτησης), η επιβολή σειράς στην αποθήκευση δεν εξασφαλίζει την ανάκτηση των αποτελεσμάτων με την ίδια σειρά. Κατά συνέπεια, ακόμα και τέτοιες λειτουργίες βελτιστοποιούνται καλύτερα από τα συστήματα και δεν έχει νόημα να αποτελούν μέρος της αναπαράστασης.

Η XML επίσης συνοδεύεται από μηχανισμούς ανάθεσης τύπων σε κόμβους. Κατ' αρχήν, οι DTDs (Document Type Definitions) παρέχουν μηχανισμούς ορισμού της δομής εγγράφων XML (μπορούν να θεωρηθούν ως γραμματικές). Δεδομένου ότι οι DTDs επίσης ορίζουν τα πεδία που μπορεί (ή πρέπει) να έχει ένα στοιχείο, μπορούν επίσης να θεωρηθούν ως σχήματα, με περιορισμένη όμως εκφραστική ισχύ [ABS99]. Γι' αυτό προτάθηκε επίσης το XML Schema [TBMM00], [MM00], καθώς και το DCD [BFM98] τα οποία παρέχουν πλήρη συστήματα τύπων, και δυνατότητα ταξινόμησης σε κλάσεις, για τους κόμβους ενός δέντρου (ή γράφου) XML.΄Ομως, κανένα από αυτά δεν λαμβάνει υπόψη τις ιδιότητες (ως γνωρίσματα ή σχέσεις μεταξύ των πόρων), καθώς αναφέρονται σε διαφορετικό μοντέλο, με ετικέττες μόνο στις ακμές Επίσης, πρέπει να σημειωθεί ότι - όπως αναλύεται και στο [KS] - οι σχέσεις επέκτασης (extension) και εκλέπτυνσης (refinement) τύπων στοιχείων XML χρησιμοποιούνται καθαρά για συντακτική διευκόλυνση (τα στιγμιότυπα μιας κλάσης δε θεωρούνται στιγμιότυπα των υπερκλάσεών της), και σκοπός τους είναι η χρησιμοποίηση υπαρχόντων ορισμών τύπων για δημιουργία νέων. Τέλος, στα σχήματα XML ένα στοιχείο δεν μπορεί να κατηγοριοποιηθεί σε περισσότερες από μία κλάσεις). Έτσι, τα μοντέλα αυτά είναι λιγότερο εκφραστικά από το RDF/RDFS, σε σχέση με τις ανάγκες της αναπαράστασης μας.

Η γλώσσα επερωτήσεων *RQL*

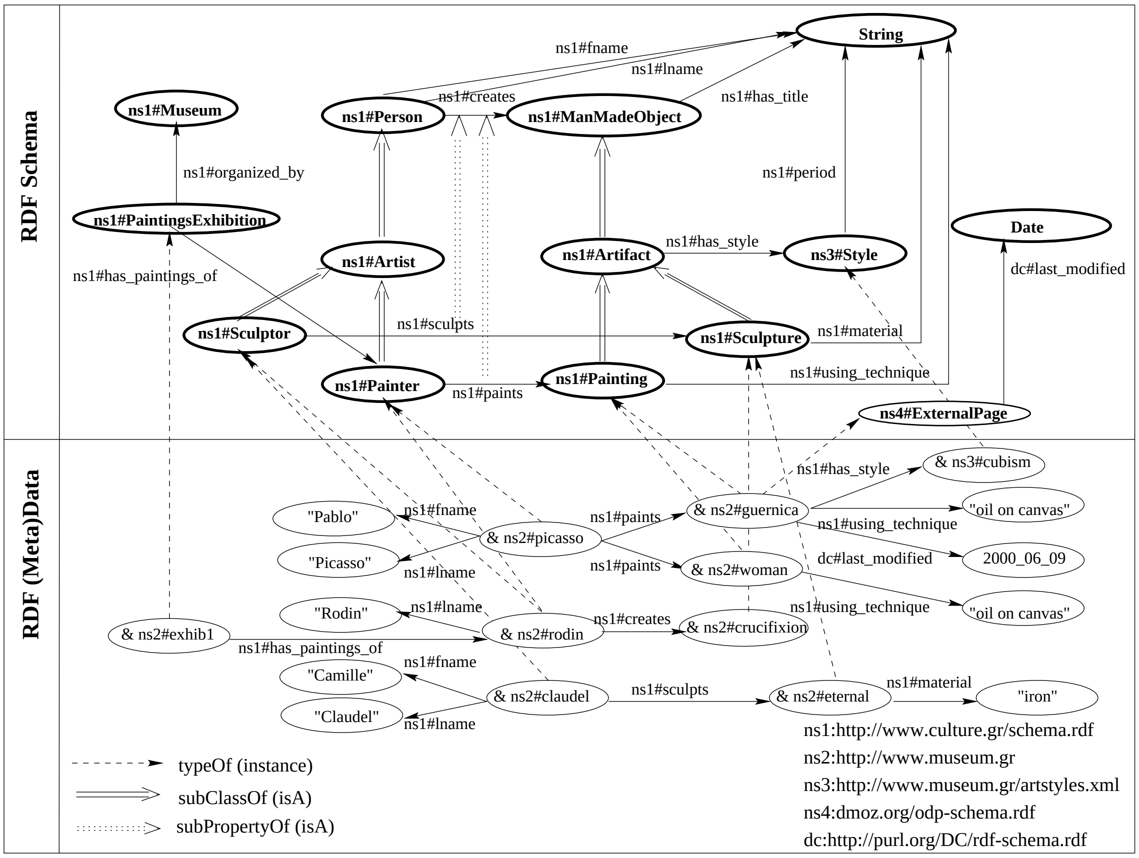
Όπως εξηγήσαμε και στο προηγούμενο κεφάλαιο, οι βασικές έννοιες του RDF έχουν σημαντικές διαφορές από τα συνηθισμένα μοντέλα οντοκεντρικών βάσεων δεδομένων [AK89]:

- ◇ Οι κλάσεις δεν ορίζουν τύπο για τα αντικείμενά τους, δηλ. ένα στιγμιότυπο μιας κλάσης είναι απλά ένα αναγνωριστικό πόρου, χωρίς τιμή/κατάσταση.

- ◇ Οι ιδιότητες, που παίζουν το ρόλο γνωρισμάτων ή σχέσεων, είναι ανεξάρτητες από τους ορισμούς τον κλάσεων, δηλ. διαφορετικοί πόροι που είναι στιγμιότυπα της ίδιας κλάσης μπορεί να έχουν διαφορετικές ιδιότητες.

- ◇ Οι ιδιότητες μπορούν επίσης να εξειδικευτούν, με μερικούς περιορισμούς σχετικά με την ύπαρξη σχέσεων υπαλληλίας μεταξύ των πεδίων ορισμού και τιμών τους.

Τα χαρακτηριστικά αυτά φαίνονται στο παράδειγμα του σχήματος 3.1. Στο πάνω μέρος του σχήματος αυτού βλέπουμε ένα κατάλογο γνώσης που έχει δημιουργηθεί για μια πύλη για πολιτιστικές κοινότητες,Όπως φαίνεται και από τους διαφορετικούς χώρους ονομάτων (*ns1*, *ns3*, *ns4*, *dc*), διάφορα τμήματα του συνολικού αυτού σχήματος προέρχονται από διαφορετικά "φυσικά" σχήματα. Η επεκτασιμότητα του RDF schema μας επιτρέπει - παρολαυτά - να συνδυάσουμε εύκολα πληροφορία που προέρχεται από διάφορα σχήματα σε ένα συνολικό σχήμα. Για τις ανάγκες της παρουσίασης, στη συνέχεια θα αναφερόμαστε στο ολοκληρωμένο (integrated) σχήμα ως ένα μοναδικό σχήμα, και θα παραλείψουμε τα προθέματα χώρου ονομάτων στα ονόματα κλάσεων και ιδιοτήτων. Στο σχήμα αυτό λοιπόν διακρίνουμε μια ιεραρχία από κλάσεις που χρησιμοποιούνται για να περιγράφουν ανθρώπους (και πιο συγκεκριμένα καλλιτέχνες) και μια άλλη με αντικείμενα-κατασκευάσματα. Οι δύο αυτές ιεραρχίες συνδέονται κατ'αρχήν μέσω



Σχήμα 3.1 Ένα ενδεικτικό παράδειγμα ημιδομημένων μεταδεδομένων και σχημάτων RDF

της ιδιότητας creates, και σε χαμηλότερα επίπεδά τους μέσω των υποιδιοτήτων της, paints και sculpts. Στις κλάσεις της ιεραρχίας αντικειμένων ορίζονται επίσης κάποιες ιδιότητες με λεκτικές τιμές (π.χ. has_material) καθώς και η σχέση has_style, που συσχετίζει αντικείμενα των κλάσεων Artifact και Style. Τέλος, στο σχήμα της εφαρμογής μας ορίζεται και η κλάση ExternalPage, στην οποία εφαρμόζεται η ιδιότητα Title, και η οποία δε συνδέεται με κάποια άλλη κλάση του σχήματος.

Το κάτω μέρος του σχήματος περιέχει ένα γράφο μεταδεδομένων, που προκύπτει από περιγραφές πόρων με βάση το παραπάνω σχήμα. Στο γράφο αυτό μπορούμε να διακρίνουμε κάποιες από τις ιδιαιτερότητες του μοντέλου RDF, όπως περιγράφηκαν στο προηγούμενο κεφάλαιο. Για παράδειγμα, ο πόρος $ns2$#picasso έχει δύο ιδιότητες paints, ενώ ο πόρος $ns2$#rodin καμμία. Επίσης, ενώ οι πόροι $ns2$#picasso και $ns2$#claudel έχουν μια ιδιότητα first_name, ως υποκλάσεις της Person, ο πόρος $ns2$#rodin δεν έχει. Ακόμη, μπορούμε να παρατηρήσουμε ότι οι πόροι $ns2$#rodin και $ns2$#`guernica` είναι κατηγοριοποιημένοι σε δύο κλάσεις.

3.1 Το τυπικό μοντέλο για το RDF/RDFS

Σε αυτή την ενότητα θα παρουσιάσουμε το μοντέλο δεδομένων που χειρίζεται τις ιδιαιτερότητες των προτύπων RDF [LS99] και RDF Schema [BG00]. Το μοντέλο αυτό διαχωρίζει σαφώς τους τύπους κλάσεων και σχέσεων (ιδιοτήτων) παρόμοια με το ODMG [AK89], και βασίζεται σε ένα μηχανισμό δημιουργίας στιγμιοτύπων γράφων, εμπνευσμένο από το GRAM [ASR94], [ASR95]. Επίσης, επιτρέπει την θεώρηση στιγμιοτύπων RDF ως ημιδομημένα δεδομένα, που μπορούν να ερμηνευτούν μερικώς από ένα ή περισσότερα σχήματα RDF.

Έστω λοιπόν τα παρακάτω μη πεπερασμένα και ανα δύο ξένα μεταξύ τους σύνολα συμβόλων:

- ♢ *Ονόματα Κλάσεων*, $\mathcal{C} = \{c_1, c_2, \ldots\}$;
- ♢ *Ονόματα Τύπων Ιδιοτήτων*, $\mathcal{P} = \{p_1, p_2, \ldots\}$;
- ♢ *Ατομικοί τύποι*, $\mathcal{L}$ με ατομικές τιμές όπως *string*, *integer*, *date* κλπ. Κάθε ατομικός τύπος έχει ένα αντίστοιχο πεδίο ορισμού, που συμβολίζεται με $dom(t)$. Στη συνέχεια θα χρησιμοποιούμε το $dom(\mathcal{L})$ ως συντομογραφία για το $\bigcup_{t \in \mathcal{L}} dom(t)$ (*rdfs:Literal*).
- ♢ *Αναγνωριστικά Αντικειμένων (object identifiers)*, $\mathcal{O} = \{o_1, o_2, \ldots\}$.

Χωρίς απώλεια της γενικότητας, υποθέτουμε ότι τα σύνολα $\mathcal{C}$ και $\mathcal{P}$ περιέχουν και το όνομα κλάσης `Class` και ιδιότητας `Property` αντίστοιχα. Το πρώτο αναπαριστά τη ρίζα της ιεραρχίας κλάσεων (δηλ. το `rdfs:Class`) ενώ το δεύτερο τη ρίζα της ιεραρχίας ιδιοτήτων (δηλ. το `rdf:Property`), όπως ορίζονται στο μοντέλο RDF/RDFS [LS99], [BG00]. Επίσης, το $\mathcal{P}$ περιέχει τις αριθμητικές ετικέττες ({ `1,2,...`}) που χρησιμοποιούνται ως ονόματα ιδιοτήτων (`rdfs:ContainerMembershipProperties`) για τα μέλη των τιμών συλλογών (`rdfs:Bag`, `rdfs:Sequence`).

Ορισμός 1: *Δοθέντος ενός συνόλου αναγνωριστικών αντικειμένων $O \subseteq \mathcal{O}$, μία τιμή στο O ορίζεται αναδρομικά ως εξής:*

- ◊ *κάθε στοιχείο του* $\mathcal{L}$ *είναι μια τιμή στο* O;
- ◊ *κάθε στοιχείο του* $\mathcal{O}$ *είναι μια τιμή στο* O;
- ◊ *Αν* $v_1, v_2, \ldots, v_n$ *είναι τιμές στο* O, *τότε* $\{1 : v_1, 2 : v_2, \ldots, n : v_n\}$ *είναι μία τιμή πολυσυνόλου (Bag) στο* O
- ◊ *Αν* $v_1, v_2, \ldots, v_n$ *είναι τιμές στο* O, *τότε* $[1 : v_1, 2 : v_2, \ldots, n : v_n]$ *είναι μία τιμή λίστας (Seq) στο* O

Σε ένα τέτοιο γράφο σχήματος, οι κλάσεις αντιστοιχούν στους κόμβους του ενώ οι ιδιότητες αντιστοιχούν στις ακμές του. Κάθε σχήμα RDF περιέχει ένα πεπερασμένο αριθμό από ονόματα κλάσεων $C \subseteq \mathcal{C}$ και ιδιοτήτων $P \subseteq \mathcal{P}$. Στη συνέχεια, οι τύποι ιδιοτήτων ορίζοναι χρησιμοποιόντας τα ονόματα κλάσεων ή τους ατομικούς τύπους έτσι ώστε:

$$\forall p \in P \; domain(p) \in C \text{ και } range(p) \in C \cup \mathcal{L}.$$

Ορισμός 2: *Μια ιεραρχία κλάσεων και τύπων ιδιοτήτων* $H = (N, \prec)$, *όπου* $N = C \cup P$, *είναι* ***καλά δομημένη*** *(well-formed) όταν* $\prec$ *είναι μια ελάχιστη μερική διάταξη (smallest partial ordering) τέτοια ώστε:*

- ◊ *Αν* $c \in C$ *τότε* $c \prec Class$;
- ◊ *Αν* $p \in P$ *τότε* $p \prec Property$;
- ◊ *Αν* $p_1, p_2 \in P$ *και* $p_1 \prec p_2$ *τότε* $domain(p_1) \prec domain(p_2)$ *και* $range(p_1) \prec range(p_2)$.

Αντίστοιχο subtyping ???? δεν μπορεί να οριστεί μεταξύ τύπων συλλογών.

Τα δεδομένα RDF (ουσιαστικά τα είδη των δεδομένων που μπορούν να εμφανιστούν στους κόμβους ενός γράφου RDF) είναι: ατομικές τιμές (π.χ. συμβολοσειρές), αναγνωριστικά αντικειμένων (π.χ. αναγνωριστικά πόρων - URIs) ή τιμές τύπων συλλογών, που χρησιμοποιούνται για τα αποτελέσματα μιας ερώτησης, όπως πολυσύνολα (`rdf:Bag`) και λίστες (`rdf:Sequence`). Η έννοια των συνόλων δεν είναι τόσο σημαντική στο RDF.

Πιο συγκεκριμένα, το σύστημα τύπων του μοντέλου μας είναι το εξής:

$\tau = \{\tau_L \mid \tau_U \mid \{\tau\} \mid [\tau] \mid (1 : \tau_1, 2 : \tau_2, \ldots, n : \tau_n)\}$

όπου: $\tau_L = \{string \mid integer \mid boolean \mid date \mid \ldots\}$[1]
$\tau_U = \{class_U \mid property_U \mid resource_U\}$
$\{.\}$ είναι ο τύπος Bag
$[.]$ είναι ο τύπος Sequence (.) είναι ο τύπος Alternative.

Ο τύπος Alternative χρησιμοποιείται για να αναπαραστήσει τη σημασιολογία της ένωσης (union) ή μεταβλητών (variant) τύπων [Car88], και είναι επίσης διατεταγμένος (δηλ. οι αριθμοί-ετικέττες παίζουν το ρόλο σημαδούρων για τα μέλη της ένωσης). Δεδομένου ότι οι συλλογές τύπου λίστας (Sequence) και ένωσης (Alternative) είναι διατεταγμένες, υπάρχει μία προκαθορισμένη διάταξη των ετικεττών, οπότε αυτές μπορούν να παραληφθούν (για τα πολυσύνολα (Bag) οι ετικέττες δεν έχουν νόημα). Επίσης, όλοι οι τύποι μας είναι *αμοιβαία αποκλειόμενοι* (π.χ. μια τιμή ατομικού τύπου δεν μπορεί να είναι και τύπου πολυσυνόλου). Το σύνολο όλων των ονομάτων τύπων συμβολίζεται με $\mathcal{T}$.

Αυτό το σύστημα τύπων μας παρέχει όλα τα απαραίτητα στοιχεία για την αναπαράσταση συλλογών τόσο με ομογενή όσο και με ετερογενή περιεχόμενα. Επίσης, μας επιτρέπει να ερμηνεύουμε ομοιόμορφα τις κλάσεις και τις ιδιότητες που ορίζονται σε σχήματα RDF. Για παράδειγμα, μια *μη διατεταγμένη πλειάδα* $[v_1, v_2, \ldots, v_n]$ (όπου v_i είναι κάποιου τύπου τ_i) μπορεί να οριστεί ως μία ετερογενής λίστα [2] με τύπο $[(\tau_1, \tau_2, \ldots, \tau_n)]$. Επίσης, οι κλάσεις RDF μπορούν να ερμηνευτούν ως μοναδιαίες σχέσεις τύπου $\{\tau_U\}$, ενώ οι ιδιότητες σαν δυαδικές σχέσεις τύπου $\{[\tau_U, \tau_U]\}$ (σχέσεις) ή $\{[\tau_U, \tau_L]\}$ (γνωρίσματα). Τέλος, οι συλλογές RDF μπορούν να χρησιμοποιηθούν για να αναπαραστήσουν πλειαδικές (n-ary) σχέσεις.

Ορισμός 3: *Έστω $(C, \prec)$ μια ιεραρχία κλάσεων. Μια συνάρτηση απόδοσης μελών στις κλάσεις (population function) με αναγνωριστικά πόρων*[3] *είναι μια συνάρτηση $\pi : C \rightarrow 2^O$, που αναθέτει σε κάθε όνομα κλάσης ένα πεπερασμένο σύνολο από αναγνωριστικά αντικειμένων (URIs) που ανήκουν στο O, με τύπο τ_U τέτοια ώστε:*

[1] All standard SQL3 datatypes (see ie. http://metalogica.com/help/pgsql/user/datatypes.htm)

[2] Μάλιστα, δεδομένου ότι οι πλειάδες είναι διατεταγμένες, για καθε δύο αντιμεταθέσεις $i_1, \ldots, i_n$ και $j_1, \ldots, j_n$ του $1, \ldots, n$, η πλειάδα $[i_1 : v_1, \ldots, i_n : v_n]$ είναι διακριτή από την $[j_1 : v_1, \ldots, j_n : v_n]$.

[3] Θεωρούμε μία non-disjoint ανάθεση αναγνωριστικών σε κλάσεις, λόγω της δυνατότητας πολλαπλής κατηγοριοποίησης ενός πόρου σε περισσότερες από μία κλάσεις.

$[\pi(c) = \cup\{\pi(c') \mid c' \prec c\}]$. *Το σύνολο όλων των τιμών συμβολίζεται με* V.

Ορισμός 4: *Δοθείσας μιας συνάρτησης* π, *η συνάρτηση ερμηνείας (interpretation function)* $[\![.]\!]$ *ορίζεται ως εξής:*

- ♢ *για τους ατομικούς τύπους:* $[\![\mathcal{L}]\!] = dom(\mathcal{L})$,
- ♢ *για τον τύπο συλλογών Bag,* $[\![\{\tau\}]\!] = \{v_1, v_2, \ldots v_n\}$ *όπου* $v_1, v_2, \ldots v_n$ *είναι τιμές τύπου* τ *στο* O.
- ♢ *για τον τύπο συλλογών Seq,* $[\![[\tau]]\!] = [v_1, v_2, \ldots v_n]$ *όπου* $v_1, v_2, \ldots v_n$ *είναι τιμές τύπου* τ *στο* O.
- ♢ *για τον τύπο συλλογών Alt,* $[\![(\tau_1, \tau_2, \ldots, \tau_n)]\!] = v_i$ *όπου* $v_i, 1 < i < n$ *είναι μία τιμή τύπου* τ_i *στο* O.
- ♢ *για κάθε κλάση* $c \in C$, $[\![c]\!] = \pi(c) \cup \{\pi(c') \mid c' \prec c\}$ *(για τη ρίζα της ιεραρχίας των κλάσεων ισχύει βέβαια* $[\![Class]\!] = O$*),*
- ♢ *για κάθε τύπο ιδιότητας* $p \in P$, $[\![p]\!] = \{[v_1, v_2] \mid v_1 \in [\![domain(p)]\!], v_2 \in [\![range(p)]\!]\} \cup \bigcup_{p' \prec p} [\![p']\!]$.

Ειδικά για τους τύπους συλλογών αξίζει να σημειωθεί ότι, ακόμα κι αν θεωρήσουμε ένα μη πεπερασμένο σύνολο από ετικέττες για τη δεικτοδότηση των μελών τους ($\{1, 2, \ldots\}$), μπορούμε παρολαυτά να αναπαραστήσουμε κάθε περίπτωση τέτοιων δεδομένων RDF με πεπερασμένο τρόπο, αφού σε κάθε στιγμιότυπο της Βάσης Δεδομένων μια συλλογή περιλαμβάνει ένα πεπερασμένο σύνολο από στοιχεία - και κατά συνέπεια εμφανίζεται μόνο ένα πεπερασμένο σύνολο από τέτοιες ετικέττες.

Ορισμός 5: *Ένα σχήμα RDF είναι μία πεντάδα* $RS = (V_S, E_S, \psi, \lambda, H)$ *όπου:*

- ♢ V_S *είναι το σύνολο των κόμβων και* E_S *το σύνολο των ακμών,*
- ♢ H *είναι μια καλά δομημένη (well-formed) ιεραρχία κλάσεων και τύπων ιδιοτήτων* $(N, \prec)$, *όπου* N *είναι ένα σύνολο ονομάτων (για τα ονόματα των κλάσεων και των τύπων ιδιοτήτων):* $N = C \cup P$,
- ♢ λ *είναι μια συνάρτηση απόδοσης ετικεττών (labeling function)* $\lambda : V_S \cup E_S \rightarrow N \cup T$,

◊ ψ *είναι μία συνάρτηση πρόσπτωσης (incidence function)* $\psi : E_S \to V_S \times V_S$.

Η συνάρτηση ψ αναπαριστά τα πεδία ορισμού (`rdfs:domain`) και τιμών (`rdfs:-range`) των ιδιοτήτων του RDF. Μάλιστα, οι συναρτήσεις ψ και λ είναι *καθολικές* (total) στα $V_S \cup E_S$ και E_S, αντίστοιχα. Αυτό δεν αποκλείει την περίπτωση κόμβων σχήματος που δεν συνδέονται μέσω κάποιας ακμής (δηλ. δεν εμφανίζονται στα άκρα κάποιας ιδιότητας). Επίσης, διαφορετικοί κόμβοι και ακμές στο RS πρέπει να έχουν διαφορετικές ετικέττες (*υποθέτουμε δηλαδή ότι οι ετικέττες κόμβων και ακμών είναι μοναδικές*).

Ορισμός 6: *Μια βάση περιγραφών RDF, σύμφωνη με ένα σχήμα RS, είναι μία πεντάδα* $RD = (V_D, E_D, \psi, \nu, \lambda)$, *όπου*

◊ V_D *είναι ένα σύνολο από κόμβους και* E_D *ένα σύνολο από ακμές,*

◊ ψ *είναι η συνάρτηση πρόσπτωσης* $\psi : E_D \to V_D \times V_D$,

◊ ν *είναι μια συνάρτηση απόδοσης τιμών* $\nu : V_D \to V$, *που αναθέτει λεκτικές τιμές σε κόμβους ατομικών τύπων (που δεν μπορούν να έχουν εξερχόμενες ακμές) και αναγνωριστικά πόρων σε σύνθετους (complex) κόμβους,*

◊ λ *είναι μία συνάρτηση απόδοσης ετικεττών* $\lambda : V_D \cup E_D \to 2^{N \cup T}$ *που ικανοποιεί την εξής συνθήκη:*

▷ *για κάθε κόμβο* v *στο* V_D, *η* λ *επιστρέφει ένα σύνολο ονομάτων* $n \in C \cup T$ *όπου η τιμή του* v *ανήκει στην ερμηνεία καθενός από τα* n: $\nu(v) \in [\![n]\!]$;

▷ *για κάθε ακμή* ϵ *στο* E_D *που ξεκινάει από τον κόμβο* v *και καταλήγει στον κόμβο* v', *η* λ *επιστρέφει το όνομα ενός τύπου ιδιότητας* $p \in P$, *για το οποίο ισχύουν τα εξής:*

⋆ *Αν* $p \in P \backslash \{1, 2, \ldots\}$, *οι τιμές των* v *και* v' *ανήκουν αντίστοιχα στις ερμηνείες των πεδίου ορισμού και τιμών της* p: $\nu(v) \in [\![domain(p)]\!]$, $\nu(v') \in [\![range(p)]\!]$;

⋆ *Αν* $p \in \{1, 2, \ldots\}$, *οι τιμές των* v *και* v' *ανήκουν αντίστοιχα στην ερμηνεία ενός τύπου* $(Bag \mid Seq \mid Alt)$ *και των αντίστοιχων τύπων των μελών τους:* $\nu(v) \in [\![Bag \mid Seq \mid Alt(\tau)]\!]$, $\nu(v') \in [\![\tau]\!]$.

Ουσιαστικά, η συνάρτηση λ αναπαριστά την ιδιότητα *rdf:type*, η οποία αντιστοιχίζει κάθε κόμβο ενός γράφου μεταδεδομένων RDF σε ένα ή περισσότερα ονόματα κλάσεων του αντίστοιχου σχήματος.

Κλείνοντας αυτή την ενότητα, μπορούμε να συμπεράνουμε ότι το μοντέλο μας εμπερικλείει την πλειοψηφία των εννοιών και των κατασκευών των RDF/RDFS, εκτός από:

◇ Τις ιδιότητες που έχουν *πολλαπλές κλάσεις στη δήλωση του πεδίου ορισμού τους*. Σε κάθε περίπτωση, είναι δυνατόν να θεωρήσουμε μια τέτοια ιδιότητα ως πολλές διαφορετικές, βάζοντας ως πρόθεμα στο κοινό όνομά τους καθεμία από τις κλάσεις που υποδεικνύουν το πεδίο ορισμού της.

◇ Τη δυνατότητα για *υποστασιοποίηση* (reification) ιδιοτήτων/προτάσεων, δεδομένου ότι δεν μπορεί να εκφραστεί σε σχήματα RDF, παρά μόνο αν ο γράφος θεωρηθεί ως υπεργράφος και υπάρχουν υπεραχμές που καταλήγουν σε σύνολα κόμβων.

Τέλος, πρέπει να σημειωθεί ότι και οι υπόλοιποι προκαθορισμένοι τύποι ιδιοτήτων (*rdfs:seeAlso*, *rdfs:isDefinedBy*, *rdfs:comment*, *rdfs:label* μπορούν να αναπαρασταθούν εύκολα στο μοντέλο μας. (??? Αξίζει να γράψουμε τίποτα παραπάνω γι' αυτές ?????)

3.2 Η γλώσσα επερωτήσεων μεταδεδομένων RDF

Στην ενότητα αυτή θα παρουσιάσουμε την *RQL* (RDF Query Language), μια γλώσσα επερωτήσεων για μεταδεδομένα RDF, η οποία επιτρέπει την ομοιόμορφη επερώτηση μεταδεδομένων και σχημάτων RDF, ως ημιδομημένα δεδομένα. Η *RQL* βασίζεται σε μια συναρτησιακή προσέγγιση, που επιτρέπει οποιαδήποτε σύνθεση συναρτήσεων (που δεν έχουν παρενέργειες) και τελεστών σάρωσης (iterators). Επίσης, παρέχει ένα συγκεκριμένο σύνολο από προκαθορισμένους αλγεβρικούς τελεστές, που ονομάζονται *φίλτρα* (filters) και μπορούν να χρησιμοποιηθούν για το σχηματισμό ερωτήσεων.

Η *RQL* βασίζεται στο μοντέλο γράφων που παρουσιάστηκε στην προηγούμενη ενότητα, το οποίο περιλαμβάνει διάφορες μορφές μεταδεδομένων RDF, όπως λεκτικές τιμές, αναγνωριστικά πόρων και σχέσεις (μοναδιαίες μεταξύ κλάσεων ή δυαδικές για τις ιδιότητες). Προκειμένου να επιτρέψει την ομοιόμορφη επερώτηση κόμβων και ακμών σε μεταδεδομένα ή σχήματα RDF, η RQL "κρύβει" τη διάκριση μεταξύ ατομικών τιμών, αναγνωριστικών πόρων και ετικεττών κλάσεων ή ιδιοτήτων. Στη συνέχεια θα παρουσιάσουμε αυτά τα χαρακτηριστικά της RQL μέσω ενδεικτικών ερωτήσεων, που βασίζονται στο παράδειγμα αναπαράστασης πολιτιστικής πληροφορίας του σχήματος 3.1.

3.2.1 Βασικές ερωτήσεις της RQL

Οι βασικές λειτουργίες/συναρτήσεις της RQL παρέχουν τρόπους για την πρόσβαση στοιχειώδους πληροφορίας που αφορά μεταδεδομένα και σχήματα RDF. Για παράδειγμα, η RQL μας δίνει τη δυνατότητα να βρούμε όλες τις κλάσεις και ιδιότητες, αντίστοιχα, που ορίζονται σε ένα σχήμα, μέσω των παρακάτω ερωτήσεων:

```
Class
Property
```

Τα ονόματα `Class` και `Property` χαρακτηρίζουν τις προκαθορισμένες *μετα-συλλογές* που ορίζονται στους χώρους ονομάτων `rdf`, `rdfs` και ουσιαστικά περιλαμβάνουν όλες τις ετικέττες κόμβων και ακμών - αντίστοιχα - που μπορούν να χρησιμοποιηθούν σε ένα γράφο ημιδομημένων (μετα)δεδομένων RDF. Οι παραπάνω ερωτήσεις επιστρέφουν τα ονόματα (ή πιο συγκεκριμένα τα αναγνωριστικά πόρων - URIs) των κλάσεων και τύπων ιδιοτήτων που φαίνονται στο πάνω τμήμα του σχήματος 3.1.

Η RQL προσφέρει επίσης κάποιες βασικές συναρτήσεις που επιτρέπουν τη διάσχιση ιεραρχιών υπαλληλίας κλάσεων/ ιδιοτήτων. Για παράδειγμα, η παρακάτω ερώτηση επιστρέφει όλες τις υποκλάσεις της `Person`, δηλαδή τα ονόματα κλάσεων `Artist`, `Painter` και `Sculptor`.

`subClassOf`(Person)

Επίσης, η RQL προσφέρει τη συνάρτηση `typeOf`, η οποία επιστρέφει τον άμεσο τύπο (ή τύπους, δεδομένης της δυνατότητας για πολλαπλή κατηγοριοποίηση σε κλάσεις) ενός πόρου, π.χ.:

`typeOf`($ns2$#rodin)

Με βάση το πολιτιστικό μας παράδειγμα του σχήματος 3.1, η ερώτηση αυτή επιστρέφει τα ονόματα κλάσεων `Painter` και `Sculptor`, καθώς ο πόρος με αναγνωριστικό $ns2$`#rodin` έχει κατηγοριοποιηθεί ως στιγμιότυπο και των δύο αυτών κλάσεων.

Πρέπει να σημειωθεί ότι και το ίδιο το αναγνωριστικό όνομα ενός πόρου αποτελεί μια βασική συνάρτηση της γλώσσας επερωτήσεων, που επιστρέφει τον εαυτό του. Γενικά, όπως και στην OQL, η γλώσσα επερωτήσεων μας παρέχει πρόσβαση στα στιγμιότυπα μιας κλάσης μέσω του ονόματος της κλάσης αυτής. Έτσι, η ερώτηση:

```
Painter
```

επιστρέφει τους πόρους $ns2$`#picasso` και $ns2$`#rodin`, αφού μόνο αυτοί οι δύο κόμβοι του γράφου μεταδεδομένων RDF έχουν το `Painter` ως ετικέττα. Πρέπει να σημειωθεί ότι στην περίπτωση της RQL θεωρήσαμε πιο χρήσιμο το όνομα μιας κλάσης να επιστρέφει και τα στιγμιότυπα όλων των υποκλάσεων της. Έτσι, η ερώτηση:

```
Artist
```

επιστρέφει τους πόρους $ns2$`#picasso`, $ns2$`#rodin` και $ns2$`#claudel`, κανένας από τους οποίους δεν έχει ταξινομηθεί άμεσα ως στιγμιότυπο της κλάσης `Artist`. Η επιλογή αυτή δικαιολογείται από το γεγονός ότι τα ονόματα των κλάσεων και των ιδιοτήτων μπορούν να θεωρηθούν ως απλοί όροι επερώτησης. Υπό το πρίσμα αυτό, η RQL παρέχει ένα αυτόματο μηχανισμό επέκτασης της ερώτησης σε πιο λιγότερο ευρείς όρους, ανάλογο με τους μηχανισμούς επέκτασης ερωτήσεων που συναντάμε σε πληροφοριακά συστήματα ανακτησης πληροφορίας που βασίζονται σε *θησαυρούς*[Fos97].

Βέβαια, υπάρχουν κάποιες περιπτώσεις ερωτήσεων στις οποίες θέλουμε να πάρουμε μόνο τα *άμεσα στιγμιότυπα* (proper instances), μιας κλάσης (δηλαδή μόνο τους πόρους που έχουν ως ετικέττα το αναγνωριστικό της κλάσης αυτής). Για το σκοπό αυτό η RQL προσφέρει τον ειδικό τελεστή "`^`". Έτσι, στο παράδειγμα του σχήματος 3.1, η ερώτηση `^Artist` επιστρέφει το κενό σύνολο.

Με δεδομένο ότι στο μοντέλο δεδομένων του RDF οι ιδιότητες είναι πολίτες πρώτης κατηγορίας, η RQL επιτρέπει και τη χρήση ονομάτων ιδιοτήτων (ετικεττών ακμών) ως βασικές συναρτήσεις πρόσβασης στα μεταδεδομένα. Αυτό είναι ιδιαίτερα χρήσιμο, ειδικά σε περιπτώσεις όπου τα σχήματα RDF που ερμηνεύουν τα μεταδεδομένα αποτελούνται:

◇ μόνο από ιδιότητες με πεδίο ορισμού τη γενική κλάση `rdf:Resource`, όπως για παράδειγμα στο σχήμα του Dublin Core [WMD95].

◇ από τεράστιες ιεραρχίες κλάσεων που συσχετίζονται μέσω λίγων ιδιοτήτων, οι οποίες ορίζονται μεταξύ κλάσεων που βρίσκονται ψηλά στην ιεραρχία υπαλληλίας. Κάτι τέτοιο είναι πολύ συνηθισμένο για σχήματα που προκύπτουν από την *ολοκλήρωση* (integration) οντολογιών και θησαυρών [AF99].

Κατα συνέπεια, οι ετικέττες των κόμβων (δηλ. τα ονόματα των κλάσεων) μπορεί να μην υπάρχουν είτε οι χρήστες της γλώσσας μπορεί να μην τις γνωρίζουν. Για τέτοιες περιπτώσεις, η RQL επιτρέπει το σχηματισμό ερωτήσεων όπως για παράδειγμα:

```
creates
```

Δεδομένου ότι μια ιδιότητα αναπαριστά μια δυαδική σχέση μεταξύ των πεδίων ορισμού και τιμών της, η ερώτηση αυτή επιστρέφει το σύνολο των διατεταγμένων ζευγών πόρων (ή τιμών) που ανήκουν στην ερμηνεία του `creates` (στον παρακάτω πίνακα τα *source* και *target* απλά υποδεικνύουν τη θέση του αντίστοιχου πόρου ή τιμής στη δυαδική σχέση).

source	*target*
$ns2$#rodin	$ns2$#crucifixion
$ns2$#picasso	$ns2$#guernica
$ns2$#picasso	$ns2$#womanbird
$ns2$#claudel	$ns2$#eternalidol

Όπως φαίνεται και στο παραπάνω αποτέλεσμα, στην ερμηνεία του `creates` συμπεριλαμβάνουμε και τις υπο-ιδιότητες του, όπως ακριβώς και στην αντίστοιχη ερμηνεία των κλάσεων. Ο τελεστής "^" μπορεί να χρησιμοποιηθεί και στην περίπτωση των ιδιοτήτων, για να πάρουμε μόνο τα άμεσα στιγμιότυπα τους (δηλ. τα ζεύγη που συνδέονται μέσω της συγκεκριμένης ιδιότητας). Πρέπει να σημειωθεί ότι η δυνατότητα αυτή, να θεωρήσουμε δηλαδή σε μια ερώτηση το όνομα μιας ακμής ως "σημείο εισόδου" στον γράφο, αποτελεί πρωτότυπο χαρακτηριστικό της RQL. Μάλιστα, όπως θα δείξουμε στη συνέχεια, τα ονόματα ιδιοτήτων αποτελούν το θεμέλιο λίθο για το σχηματισμό των (γενικευμένων ή μη) εκφράσεων μονοπατιών της RQL.

Η RQL υποστηρίζει επίσης πράξεις μεταξύ συνόλων ίδιου τύπου, με τους τελεστές ένωσης (union), τομής (intersect) και διαφοράς (minus)(βλ. και Πίνακα 3.2). Για παράδειγμα, η ερώτηση:

```
Painter intersect Sculptor
```

επιστρέφει το σύνολο που περιέχει μόνο το αναγνωριστικό $ns2$#rodin, αφού αυτός είναι ο μόνος πόρος στο παράδειγμά μας που έχει κατηγοριοποιηθεί και στις δύο αυτές κλάσεις. Ανάλογες πράξεις μπορούν να γίνουν και μεταξύ των ιδιοτήτων. Αξίζει να σημειωθεί ότι η RQL δεν επιτρέπει ανάλογες πράξεις μεταξύ κλάσεων και ιδιοτήτων, καθώς οι πρώτες αναπαριστούνται μέσω μοναδιαίων σχέσεων ενώ οι δεύτερες μέσω δυαδικών.

Εκτός από συλλογές από στιγμιότυπα κλάσεων ή ιδιοτήτων, η RQL χειρίζεται και συλλογές RDF, που μπορεί να είναι τύπου rdf:Bag, rdf:Seq ή rdf:Alt (η σημασιολογία των οποίων εξηγήθηκε στο προηγούμενο κεφάλαιο). Για παράδειγμα, ο Boolean τελεστής "in" μπορεί να χρησιμοποιηθεί για να ελέγξει αν ένας πόρος είναι μέλος μιας τέτοιας συλλογής. Έτσι, η ερώτηση:

$ns2$#picasso in $ns2$#Bag1

επιστρέφει true, αν ο πόρος με αναγνωριστικό $ns2$#picasso περιέχεται στη συλλογή με αναγνωριστικό $ns2$#Bag1. Επίσης, μπορούμε να ρωτήσουμε τον αριθμό των μελών μιας συλλογής, χρησιμοποιώντας τη - μη πρωτοβάθμια - συνάρτηση count. Ειδικά για τη συλλογή τύπου rdf:Seq που αναπαριστά λίστες, μπορούμε να χρησιμοποιήσουμε τον τελεστή "[ ]" με τον κατάλληλο δείκτη θέσης (ή πεδίο τιμών τέτοιων δεικτών). Για παράδειγμα, η παρακάτω ερώτηση επιστρέφει το δεύτερο, το τρίτο και το τέταρτο στοιχείο της συλλογής τύπου rdf:Seq με αναγνωριστικό $ns2$#Seq1. Αν δεν υπάρχει κανένα από αυτά τα στοιχεία, η ερώτηση επιστρέφει το κενό σύνολο, που συμβολίζεται με "{ }".

$ns2$#Seq1 [2:4]

Τέλος, η RQL υποστηρίζει συνηθισμένα Boolean κατηγορήματα όπως =, <, > και like (για ταίριασμα προτύπων). Όλοι αυτοί οι τελεστές μπορούν να εφαρμοστούν τόσο μεταξύ λεκτικών τιμών όσο και μεταξύ αναγνωριστικών πόρων. Μάλιστα, η δεύτερη περίπτωση περιλαμβάνει και συγκρίσεις μεταξύ ονομάτων κλάσεων ή ιδιοτήτων, όπου η σύγκριση δύο ονομάτων αναφέρεται στη διάταξη των αντίστοιχων κλάσεων μέσω της σχέσης υπαλληλίας. Έτσι, η συνθήκη "`Painter < Artist`" επιστρέφει `true`, αφού η κλάση `Painter` είναι υποκλάση της `Artist`. Η επιλογή του κατάλληλου ελέγχου σε κάθε περίπτωση (ισότητα λεκτικών τιμών αντί αντικειμένων, λεξικογραφική διάταξη αντί διάταξη κλάσεων κλπ) γίνεται εξετάζοντας τον τύπο των ορισμάτων. Συγκρίσεις μεταξύ ορισμάτων διαφορετικών τύπων μέσω των τελεστών αυτών δεν είναι αποδεκτές.

3.2.2 Βασικά φίλτρα

Προκειμένου να επιτρέψει συστηματική σάρωση σε συλλογές από μεταδεδομένα RDF και να δώσει τη δυνατότητα για χρήση μεταβλητών σε ερωτήσεις, η RQL παρέχει το βασικό φίλτρο `select-from-where`, που χρησιμοποιεί τις βασικές ερωτήσεις που παρουσιάστηκαν στην προηγούμενη ενότητα. Δεδομένου ότι τα μεταδεδομένα RDF είναι μια περίπτωση ημιδομημένων δεδομένων, τα φίλτρα αυτά χρησιμοποιούν επίσης *εκφράσεις μονοπατιών* (path expressions) για να διασχίσουν γράφους μεταδεδομένων σε αυθαίρετο βάθος. Η τυπική ερμηνεία των φίλτρων και των εκφράσεων μονοπατιών της RQL δίνεται στην ενότητα 3.3. Στη συνέχεια, θα προσπαθήσουμε να παρουσιάσουμε τη χρήση και τη λειτουργικότητα της RQL μέσω παραδειγμάτων που βασίζονται στο παράδειγμα αναπαράστασης πολιτιστικής πληροφορίας του σχήματος 3.1.

Έστω για παράδειγμα η παρακάτω ερώτηση, που είναι ισοδύναμη με τη βασική ερώτηση `creates`:

```
select  X, Y
from    {X}creates{Y}
```

Στο τμήμα του `from` της παραπάνω ερώτησης χρησιμοποιούμε μια απλή έκφραση μονοπατιού με το όνομα της ιδιότητας `creates`. Η έκφραση αυτή περιορίζει το πεδίο τιμών των μεταβλητών κόμβων X και στις αντίστοιχες τιμές από τα διατεταγμένα ζεύγη τιμών που ανήκουν στην ερμηνεία του `creates`, όπως είδαμε στην προηγούμενη ενότητα. Στο συγκεκριμένο παράδειγμα, η μεταβλητή X παίρνει τιμές από το `source` της ιδιότητας `creates` (βλ. παράδειγμα ενότητας 3.2.1) ενώ η Y από το `target`.

Το τμήμα του `select` περιγράφει - ως συνήθως - μια προβολή πάνω στις επιθυμητές μεταβλητές. Στην περίπτωση που επιθυμούμε την προβολή του αποτελέσματος πάνω σε όλες τις μεταβλητές που χρησιμοποιούνται στην ερώτηση, μπορούμε να χρησιμοποιήσουμε το "`select *`". Προκειμένου να εξασφαλίσουμε την κλειστότητα (closure) της γλώσσας, ώστε να μπορούμε να συνθέσουμε τέτοια φίλτρα με οποιοδήποτε τρόπο, επιλέξαμε για τα αποτελέσματα μια μορφή που μπορεί να αναπαρασταθεί σε RDF (και - κατά συνέπεια - να επερωτηθεί με την RQL). Έτσι, τα αποτελέσματα των ερωτήσεων είναι συλλογές τύπου `rdf:Bag`. Εξάλλου, διατεταγμένες πλειάδες μπορούν να αναπαρασταθούν από συλλογές τύπου `rdf:Seq`, στις οποίες έχουμε και πρόσβαση μέσω δεικτών θέσης (όπως π.χ. σε δομές πινάκων σε κοινές γλώσσες προγραμματισμού). Για να γίνει σαφής αυτή η αναπαράσταση, το αποτέλεσμα της προηγούμενης ερώτησης, σειριοποιημένο σε RDF/XML είναι:

```
<rdf:Bag ID="#result1">
  <rdf:li>
     <rdf:Seq>
        <rdf:li rdf:resource= "ns2#picasso"/>
        <rdf:li rdf:resource= "ns2#guernica"/>
     </rdf:Seq> </rdf:li>
  <rdf:li>
     <rdf:Seq>
        <rdf:li rdf:resource= "ns2#picasso"/>
        <rdf:li rdf:resource= "ns2#woman"/>
     </rdf:Seq> </rdf:li>
  <rdf:li>
     <rdf:Seq>
        <rdf:li rdf:resource= "ns2#rodin"/>
        <rdf:li rdf:resource= "ns2#crucifixion"/>
     </rdf:Seq> </rdf:li>
  <rdf:li>
     <rdf:Seq>
        <rdf:li rdf:resource= "ns2#claudel"/>
        <rdf:li rdf:resource= "ns2#eternalidol"/>
```

```
        </rdf:Seq> </rdf:li>
    </rdf:Bag>
```

Όπως αναφέρθηκε και στην ενότητα 3.1, οι τύποι συλλογών RDF μπορούν να αναπαραστήσουν ετερογενείς συλλογές. Επομένως, είναι κατάλληλες αποτελεσμάτων ερωτήσεων. Για παραδειγμα, το αποτέλεσμα της παραπάνω ερώτησης είναι τύπου $\{[\tau_U, \tau_U]\}$.

Η ιδιότητα της κλειστότητας, που αναφέρθηκε προηγουμένως, εξασφαλίζεται από τις βασικές ερωτήσεις που υποστηρίζονται για συλλογές και παρουσιάστηκαν στο τέλος της προηγούμενης ενότητας.

Για τις ανάγκες της παρουσίασης, στο υπόλοιπο του κεφαλαίου δεν θα χρησιμοποιήσουμε την αναπαράσταση συλλογών για τα αποτελέσματα των ερωτήσεων αλλά μια μορφή πίνακα από πλειάδες αποτελεσμάτων, που πιστεύουμε ότι είναι πιο εύκολα κατανοητή από τον αναγνώστη.

Έχοντας παρουσιάσει τα βασικά συστατικά και χαρακτηριστικά των φίλτρων της RQL μπορούμε πλέον να δούμε πως μπορούμε να συνδυάσουμε με οποιοδήποτε τρόπο τέτοια φίλτρα και εκφράσεις μονοπατιών με τις βασικές ερωτήσεις της προηγούμενης ενότητας, για να σχηματίσουμε "χρήσιμες" και πολύπλοκες ερωτήσεις. Για παράδειγμα:

Ερώτηση 1: *Βρες τους πόρους που έχει ζωγραφίσει (*`paints`*) κάποιος με όνομα "Pablo", και είναι φτιαγμένοι από (*`using_technique`*) "oil on canvas".*

```
select  W
from    {X}first_name{Y}, {Z}paints{W}.using_technique{Q}
where   X = Z and Y = "Pablo" and Q = "oil on canvas"
```

Η ερώτηση αυτή επιστρέφει μια συλλογή με τα αναγνωριστικά $ns2$`#guernica` και $ns2$`#woman`, με βάση το παράδειγμα 3.1. Η συνθήκη $=$ (ισότητα μεταξύ αντικειμένων) υποδηλώνει μια σύνδεση (join) ανάμεσα στις ερμηνείες των ιδιοτήτων `first_name` και `paints`, με βάση τις τιμές από τις οποίες ξεκινούν (source values). Η δεύτερη έκφραση μονοπατιού επίσης υπονοεί μια σύνδεση ανάμεσα στις ερμηνείες των ιδιοτήτων `paints` και `using_technique`, πάνω στις τιμές `target` της πρώτης και `source` της δεύτερης.

Αξίζει να σημειωθεί ότι, όπως και στα ημιδομημένα μοντέλα δεδομένων, στο παράδειγμά μας υπάρχουν δύο στιγμιότυπα της ιδιότητα `paints` για τον πόρο

$ns2$`#picasso`, ενώ στον πόρο $ns2$`#rodin` δεν έχει οριστεί η ιδιότητα `first_name`. Έτσι, οι εκφράσεις μονοπατιών της RQL πρέπει να έχουν την ίδια υπαρξιακή σημασιολογία (existential semantics) με την πλοήγηση σε γράφους σε ημιδομημένα μοντέλα και γλώσσες επερωτήσεών τους. Πράγματι, σε κάθε περίπτωση, οι εκφράσεις μονοπατιών της RQL μπορούν να γραφτούν χρησιμοποιώντας υπαρξιακούς[4] ποσοδείκτες.

```
select  W
from    {X}first_name{Y}, {Z}paints{W}
where   X = Z and Y = "Pablo" and
        exists P {select  R
                  from    {R}has_material{Q}
                  where   Q = "oil on canvas"}: P=W
```

Η μεταβλητή P στον υπαρξιακό iterator ??? παίρνει τιμές από το σύνολο των αποτελεσμάτων της εμφωλευμένης ερώτησης.

Στην αποτίμηση της **ερώτησης 1** αγνοήσαμε τελείως τα ονόματα των κλάσεων που χρησιμοποιούνται ως ετικέττες των πόρων που εμφανίζονται στα άκρα των ιδιοτήτων που μας ενδιαφέρουν. Ουσιαστικά λοιπόν, χειριστήκαμε το γράφο σαν κοινό γράφο ημιδομημένων δεδομένων που - όπως αναλύσαμε στην ενότητα 2.7.4 - έχει ετικέττες μόνο στις ακμές. Μάλιστα, δεδομένου ότι κάποιοι πόροι μπορεί να έχουν κατηγοριοποιηθεί σε πολλές κλάσεις (π.χ. ο πόρος με αναγνωριστικό $ns2$`#guernica` ως `Painting` και `ExternalPage`), μπορούμε να επερωτήσουμε έγκυρα μονοπάτια στο γράφο μεταδεδομένων που δεν εμφανίζονται ρητά σε κάποιο σχήμα RDF.

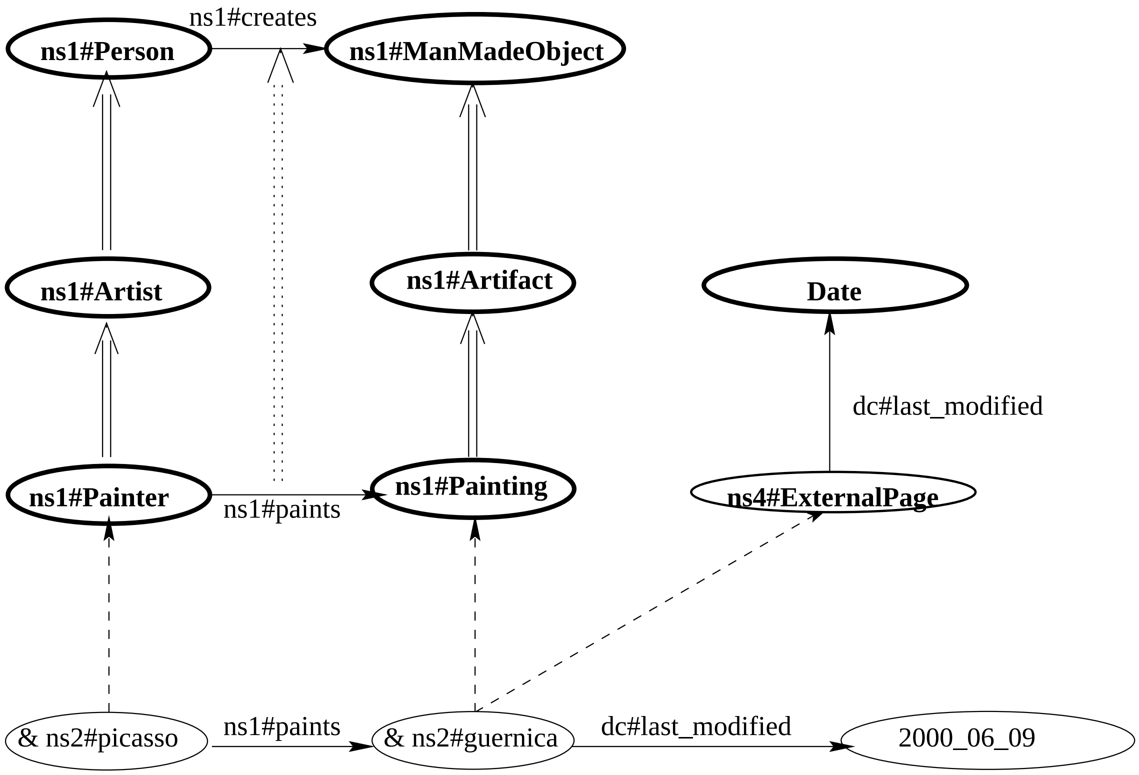
Ερώτηση 2:_Βρες τους πόρους που έχει δημιουργήσει (creates) κάποιος μετά την αρχή του έτους 2000, καθώς και τον δημιουργό τους_

```
select  X, Y
from    {X}creates{Y}.last_modified{Z}
where   Z ¿= 2000-01-01
```

Παρατηρούμε ότι, στο γράφο σχήματος δεν υπάρχει κανένα τέτοιο μονοπάτι, καθώς η ιδιότητα `last_modified` δεν ορίζεται σε καμμία από τις κλάσεις που αποτελούν το πεδίο τιμών της `creates`. Όμως, στο γράφο μεταδεδομένων ορίζεται

[4]Η RQL υποστηρίζει επίσης καθολικούς ποσοδείκτες

ένα τέτοιο μονοπάτι (όπως φαίνεται στο σχήμα 3.2, καθώς ο πόρος $ns2$#guernica στον οποίο καταλήγει μια ακμή paints - θεωρώντας τον ως στιγμιότυπο της κλάσης Painting - έχει μια ιδιότητα last_modified, λόγω της κατηγοριοποίησής του στην κλάση ExternalPage).



Σχήμα 3.2 Το μονοπάτι δεδομένων που διατρέχει η ερώτηση 1

Προκειμένου να διατηρήσουμε μια συντακτική συμβατότητα με την OQL [CB97], οι εκφράσεις μονοπατιών μπορούν να ξεκινούν και από αντικείμενα/πόρους, όπως για παράδειγμα:

```
select  X
from    X Artist
where   X.last_name = "Rodin"
```

Η ερώτηση αυτή δεν είσαγει νέα σημασιολογία στις εκφράσεις μονοπατιών, αλλά είναι απλά μια συντακτική διευκόλυνση που παρέχει η RQL. Ουσιαστικά, είναι ισοδύναμη με την ερώτηση:

```
select  X
```

`from` X Artist, {Y}last_name{Z}
`where` X = Y and Z = "Rodin"

3.2.3 Επερωτώντας (μετα)δεδομένα RDF με ατελή γνώση του σχήματος

Σε αυτή την ενότητα θα παρουσιάσουμε τις *γενικευμένες εκφράσεις μονοπατιών* [CACS94], [CCM96] (generalized path expressions) της RQL, οι οποίες μπορούν να περιέχουν μεταβλητές τόσο στους κόμβους όσο και στις ακμές (ονόματα κλάσεων και ιδιοτήτων αντίστοιχα). Έστω, για παράδειγμα, η ερώτηση:

Ερώτηση 3: *Βρείτε τις τιμές (πόρους ή λεκτικές) όλων των ιδιοτήτων που το όνομά τους ταιριάζει με το πρότυπο "*name".*

`select` Y
`from` $P Property, {X}$P{Y}
`where` $P like "*name"

Προκειμένου να διαχωρίσουμε τις μεταβλητές που αναφέρονται σε σχήματα από αυτές που αναφέρονται σε δεδομένα, χρησιμοποιούμε για τις πρώτες ως πρόθεμα το σύμβολο $. Έτσι, στο παραπάνω παράδειγμα, η $\$P$ είναι μια μεταβλητή σχήματος, που παίρνει τιμές από το σύνολο που ορίζεται από τη μετα-κλάση `Property` και περιέχει όλους τους τύπους ιδιοτήτων. Στη συνέχεια, οι μεταβλητές δεδομένων X και Y - που εξαρτώνται από την $\$P$ - περιορίζονται στα πεδία ορισμού και τιμών της ιδιότητας στην οποία έχει αποτιμηθεί η $\$P$. Συνεπώς, το τμήμα του `from` της ερώτησης 3 ουσιαστικά υποδηλώνει τον υπολογισμό του καρτεσιανού γινομένου των ονομάτων ιδιοτήτων που ορίζονται στο σχήμα RDF, με τα αντίστοιχα στιγμιότυπά τους. Με βάση αυτά, για το παράδειγμα του σχήματος 3.1, η μεταβλητή $\$P$ θα πάρει τις τιμές `first_name` και `last_name` και η μεταβλητή δεδομένων Y τις λεκτικές τιμές "Pablo", "Camille" και "Picasso", "Rodin", "Claudel", αντίστοιχα.

Επιπλέον, οι εκφράσεις μονοπατιών (γενικευμένες ή μη) μπορούν να περιέχουν και μεταβλητές που παίρνουν τιμές από ετικέττες κόμβων (δηλ. ονόματα κλάσεων που εμφανίζονται στα πεδία ορισμού και τιμών μιας ιδιότητας), όπως - για παράδειγμα - στην παρακάτω ερώτηση:

Ερώτηση 4: *Βρες τους πόρους τύπου (με ετικέττα)* `Sculptor` *που έχουν δημιουργήσει* (`creates`) *κάτι.*

```
select  X, Y
from    {X:$Z}creates{Y}
where   $Z = Sculptor
```

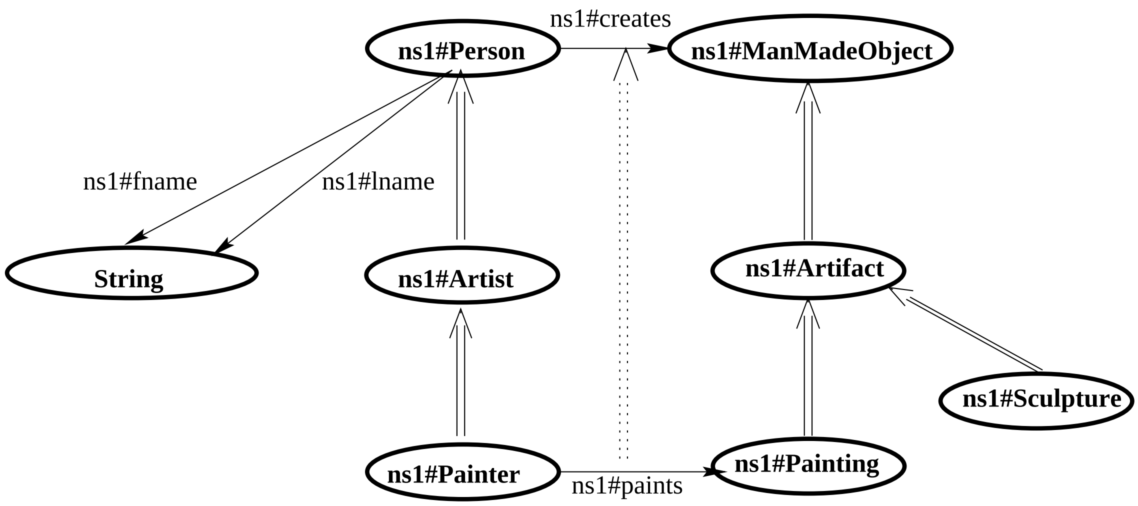
Η μεταβλητή σχήματος $\$Z$ περιορίζεται έμμεσα στα μέλη της μετα-κλάσης Class. Στην ερώτηση 4, η $\$Z$ θα πάρει ως τιμές τις κλάσεις που περιέχονται στο πεδίο ορισμού της ιδιότητας creates, δηλαδή την ίδια την κλάση στην οποία έχει οριστεί (δηλ. την Person) και όλες τις υποκλάσεις της (δηλ. τις κλάσεις Artist, Painter και Sculptor). Στη συνέχεια, ο περιορισμός μέσω της συνθήκης $\$Z$ = Sculptor που εισάγεται στο τμήμα του where της ερώτησης και υποδηλώνει ισότητα ονομάτων κλάσεων, περιορίζει τις αποδεκτές τιμές του $\$Z$ στην τιμή Sculptor. Έπειτα, η έκφραση "$X : \$Z$", περιορίζει τις δυνατές τιμές της μεταβλητής δεδομένων X - που παίρνει τιμές από τα στιγμιότυπα της ιδιότητας creates και των υπο-ιδιοτήτων της - στους πόρους που είναι στιγμιότυπα (άμεσα, λόγω της αυστηρής ισότητας) της κλάσης Sculptor. Έτσι, για το δεδομένο παράδειγμα, η μεταβλητή X στην παραπάνω ερώτηση θα πάρει τις τιμές $ns2$#rodin και $ns2$#claudel. Αντίθετα, αν δεν είχαμε εισάγει τη συνθήκη $\$Z$ = Sculptor, η μεταβλητή $\$Z$ θα έπαιρνε και τις τιμές Person, Artist και Painter (όλες τις κλάσεις στο πεδίο ορισμού της ιδιότητας) και η μεταβλητή δεδομένων θα έπαιρνε όλες τις τιμές που εμφανίζονται στο source όλων των στιγμιοτύπων της creates, συμπεριλαμβάνοντας έτσι και τον κόμβο $ns2$#picasso. Αξίζει επίσης να σημειωθεί ότι αν η κλάση στην παραπάνω συνθήκη δεν είναι υποκλάση της κλάσης ορισμού της ιδιότητας (δηλ. δεν ανήκει στο πεδίο ορισμού της), η παραπάνω ερώτηση θα επιστρέφει το κενό σύνολο για όλες τις μεταβλητές, χωρίς να χρειαστεί να εξετάσει τα στιγμιότυπα της ιδιότητας. Τελικά, το πλήρες αποτέλεσμα της παραπάνω ερώτησης είναι:

X	Y
$ns2$#rodin	$ns2$#crucifixion
$ns2$#claudel	$ns2$#eternalidol

Από τον παραπάνω πίνακα παρατηρούμε ότι στο αποτέλεσμα συμπεριλαμβάνεται και ο πόρος $ns2$#claudel. Πράγματι, ο πόρος αυτός είναι τύπου Sculptor και έχει μια ιδιότητα sculpts. Δεδομένου λοιπόν ότι η ιδιότητα αυτή συνδέεται με την creates μέσω της σχέσης υπαλληλίας (δηλ. είναι υποιδιότητά της) και ότι - όπως αναφέρθηκε προηγουμένως - θεωρούμε στην ερμηνεία μιας ιδιότητας και τα στιγμιότυπα των υποιδιοτήτων τους, ο πόρος αυτός φυσιολογικά συμπεριλαμβάνεται στα αποτελέσματα της παραπάνω ερώτησης, παρόλο που δεν έχει την ίδια την ιδιότητα `creates`.

Προκειμένου να διευκολύνουμε τη σύνταξη τέτοιου είδους εκφράσεων μονοπατιών, η RQL υποστηρίζει επίσης συντομογραφίες όπως "`{X:Sculptor}creates{Y}`", που υποδηλώνει ακριβώς την περίπτωση της ερώτησης 4 (`{X:$Z}creates{Y}` και `$Z = Sculptor`).

Με άλλα λόγια, η RQL επεκτείνει την έννοια των εκφράσεων μονοπατιών σε ολόκληρα μονοπάτια ιεραρχίας κλάσεων (ή ιδιοτήτων), όπως αυτά σχηματίζονται μέσω της σχέσης υπαλληλίας. Αυτό είναι ιδιαίτερα χρήσιμο καθώς οι πόροι μπορεί να είναι κατηγοριοποιημένοι σε διάφορες κλάσεις, και σε ένα πόρο μπορεί να ορίζονται ιδιότητες που προέρχονται από διαφορετικές ιεραρχίες κλάσεων. Παρολαυτά, η RQL επιτρέπει τη διάκριση τέτοιων ιδιοτήτων, επιτρέποντας για παράδειγμα να ξεχωρίσουμε τις ιδιότητες που έχει ένας πόρος μόνο ως αντικείμενο της κλάσης `Painter`, και όχι της κλάσης `Sculptor`, όπως φαίνεται στο σχήμα 3.3.



Σχήμα 3.3 Το σύνολο των ιδιοτήτων που μπορούν να εφαρμοστούν σε ένα αντικείμενο της κλάσης Painter

3.2.4 Επερωτώντας σχήματα RDF ως ημιδομημένα δεδομένα

Σ'αυτή την ενότητα θα παρουσιάσουμε τις δυνατότητες της RQL για επερώτηση σχημάτων RDF, χωρίς να μας απασχολούν τα στιγμιότυπά τους. Έχοντας αυτή τη λειτουργικότητα, η RQL μπορεί να χρησιμοποιηθεί σαν μια γλώσσα υψηλού επιπέδου για αναδίφυση σε σχήματα. Κάτι τέτοιο είναι πολύ χρήσιμο σε πραγματικές εφαρμογές, καθώς τα σχήματα RDF:

◇ μπορεί να είναι υπερβολικά μεγάλα, και έτσι δεν μπορούμε να τα διαχειριστούμε στην κύρια μνήμη του συστήματος.[AF99]

◇ περικλείουν πληροφορία που δεν εκφράζεται ρητά και δεν είναι εύκολο να διακριθεί, λόγω της ύπαρξης ιεραρχιών υπαλληλίας. Έστω για παράδειγμα ότι θέλουμε να βρούμε όλα τα ονόματα ιδιοτήτων που ξεκινούν από ένα κόμβο σε ένα γράφο σχήματος RDF:

Ερώτηση 5: *Βρες όλες τα ονόματα ιδιοτήτων και τα αντίστοιχα πεδία τιμών τους, που μπορούν να οριστούν πάνω σε ένα πόρο τύπου* `Painter` *ή κάποιας υποκλάσης αυτού.*

```
select  $P, $Y
from    {$X}$P{$Y}
where   $X <= Painter
```

Κάποιες από τις ακμές του αποτελέσματος περιγράφονται ρητά στο σχήμα (π.χ. η `paints` με πεδίο τιμών την κλάση `Painting` στο παράδειγμα του σχήματος 3.1). Υπάρχουν όμως και ακμές που εξάγονται από το γεγονός ότι η ιδιότητα `paints` είναι υπο-ιδιότητα της `creates`. Το πλήρες αποτέλεσμα της παραπάνω ερώτησης φαίνεται στον παρακάτω πίνακα:

$\$P$	$\$Y$
creates	Artifact
creates	Painting
paints	Painting
first_name	String
last_name	String

Σε κάποιες περιπτώσεις η παραπάνω αυτόματη επέκταση των ερωτήσεων ίσως δεν είναι επιθυμητή. Για παράδειγμα, δοθέντων δύο ιεραρχιών κλάσεων θέλουμε να βρούμε για κάθε κλάση τις ιδιότητες που έχουν πεδίο ορισμού ακριβώς την κλάση αυτή.

Δε θέλουμε δηλαδή να συμπεριλάβουμε τις ιδιότητες που κληρονομούνται από άλλες υπερκλάσεις ή υπεριδιότητες. Η ερώτηση αυτή μπορεί να γραφεί ως εξής:

Ερώτηση 6: *Για κάθε κλάση μιας ιεραρχίας με κορυφή C_1, βρες όλα τα ονόματα ιδιοτήτων και τα αντίστοιχα πεδία (κλάσεις) τιμών που έχουν ως πεδίο ορισμού την κλάση αυτή*

`select` domain($P), $P, range($P)
`from` $P Property
`where` domain($P) <= C_1 and range($P) <= C_2

Εναλλακτικά, μπορεί σε μια ερώτηση να αναζητούμε τις κλάσεις στις οποίες καταλήγει μια έκφραση μονοπατιών σχήματος, όπως στην ερώτηση:

Ερώτηση 7: *Βρες τις κλάσεις στις οποίες μπορούμε να φτάσουμε σε ένα βήμα από μια κλάση στο πεδίο τιμών της ιδιότητας* `creates`.

`select` $Y, $P, $Z
`from` creates{$Y}.$P{$Z}

Η ερώτηση 7 "φιλτράρει" τα περιεχόμενα της μετα-κλάσης `Property`, περιορίζοντας το πεδίο ορισμού κάθε αποτίμησης του $\$P$ στο $\$Y$. Οι τελικές τιμές για τις μεταβλητές σχήματος $\$Y$, $\$P$ και $\$Z$ είναι οι εξής:

$\$Y$	$\$P$	$\$Z$
Object	has_title	String
Artifact	has_title	String
Artifact	has_style	Style
Painting	has_title	String
Painting	has_style	Style
Painting	using_technique	String
Sculpture	has_title	String
Sculpture	has_style	Style
Sculpture	has_material	String

Στο παραπάνω αποτέλεσμα βλέπουμε ότι - όπως είχαμε αναφέρει και στο σχολιασμό της ερώτησης 2 - δεν υπάρχει μονοπάτι στο σχήμα που να περιλαμβάνει τις ιδιότητες `creates` και `last_modified`. Όμως τέτοια μονοπάτια μπορεί να εμφανιστούν σε γράφους δεδομένων, λόγω της πολλαπλής ταξινόμησης σε κλάσεις. Γι'αυτό η *RQL* παρέχει την απαραίτητη λειτουργικότητα ώστε να καλύψει και τις δύο περιπτώσεις.

Η *RQL* επιτρέπει επίσης τη σύνθεση φίλτρων για το σχηματισμό ερωτήσεων. Για παράδειγμα, στην παρακάτω ερώτηση, το εσωτερικό φίλτρο επιλέγει μερικά μονοπάτια σχήματος, ενώ το εξωτερικό χρησιμοποιεί τα μονοπάτια αυτά σε μια έκφραση μονοπατιού δεδομένων:

Ερώτηση 8: *Δώσε τις τεχνικές με τις οποίες έχουν δημιουργηθεί (creates) πόροι, οι δημιουργοί των οποίων σχετίζονται με κάποια ιδιότητα με μια έκθεση έργων ζωγραφικής (PaintingsExhibition).*

```
select  Z
from    $R (select  $Q
              from    {$X}$P.$Q{$Y}
              where   $X = PaintingsExhibition and $Q < creates),
        $R.using_technique{Z}
```

Στην ερώτηση αυτή το εμφωλευμένο φίλτρο επιστρέφει το δεύτερο τμήμα των μονοπατιών με μήκος δύο που ξεκινούν από την κλάση `PaintingsExhibition`. Έτσι, η μεταβλητή σχήματος $\$Q$ αποτιμάται σε `paints`, αφού το πεδίο τιμών της `has_paintings_of` είναι μόνο η κλάση `Painter` και λόγω του περιορισμού $\$Q <$ creates. Στη συνέχεια, στο εξωτερικό φίλτρο, ζητούνται οι τιμές της `using_technique`, που μπορεί να εφαρμοστεί σε αντικείμενα της κλάσης `Painting`. Έτσι, και αφού υπάρχουν τα αντίστοιχα μονοπάτια `paints.using_technique`, παίρνουμε για το Z τις τιμές "oil on canvas", "oil on canvas" (δύο φορές, για τις δύο ακμές που ξεκινούν από τους πόρους $ns2$`#guernica` και $ns2$`#woman` αντίστοιχα.

Ως τελευταίο παράδειγμα, θα δείξουμε πως η *RQL* μπορεί να χρησιμοποιηθεί για να εκφράσει τη λειτουργία ανάκτησης `rdf:AboutEachPrefix` που περιγράφεται στο [LS99] και επιστρέφει τόσο πληροφορία σχήματος όσο και μεταδεδομένων (περιγραφόμενους πόρους).

Ερώτηση 9: *Δώσε μου ότι πληροφορία υπάρχει σχετικά με τους πόρους του δικτυακού τόπου "http://www.museum.gr".*

```
select  X, $Z, $P, Y, $W
from    {X:$Z}$P{Y:$W}
```

`where` Y `like` "http://www.museum.gr*" `or` X `like` "http://www.museum.gr*"

Αυτή η ερώτηση θα αποτιμηθεί ως εξής: για κάθε τιμή της μεταβλητής σχήματος $\$P$ (που παίρνει ως τιμές όλα τα ονόματα ιδιοτήτων), ορίζω δύο επαναλήψεις με δείκτες \$ και $\$W$ πάνω στην ιεραρχία υποκλάσεων των πεδίων ορισμού και τιμών της αντίστοιχης ιδιότητας. Έπειτα, οι μεταβλητές δεδομένων X και παίρνουν ως τιμές τους κόμβους στα άκρα κάθε ιδιότητας που είναι άμεσα στιγμιότυπα (proper instances) των κλάσεων που υποδεικνύονται από τις μεταβλητές σχήματος $\$Z$, $\$W$. Στο παράδειγμά μας (βλ. σχήμα 3.1), το αποτέλεσμα της παραπάνω ερώτησης φαίνεται στον πίνακα 3.1.

X	$\$Z$	$\$P$	Y	$\$W$
ns2#rodin	Painter	creates	ns2#crucifixion	Painting
ns2#rodin	Sculptor	creates	ns2#crucifixion	Painting
ns2#picasso	Painter	paints	ns2#guernica	Painting
ns2#picasso	Painter	paints	ns2#woman	Painting
ns2#claudel	Sculptor	sculpts	ns2#eternalidol	Sculpture
ns2#picasso	Painter	first_name	"Pablo"	String
ns2#claudel	Sculptor	first_name	"Camille"	String
ns2#picasso	Painter	last_name	"Picasso"	String
ns2#rodin	Painter	last_name	"Rodin"	String
ns2#rodin	Sculptor	last_name	"Rodin"	String
ns2#claudel	Sculptor	last_name	"Claudel"	String
ns2#guernica	Painting	has_style	ns3#cubism	Style
ns2#guernica	Painting	using_technique	"oil on canvas"	String
ns2#woman	Painting	using_technique	"oil on canvas"	String
ns2#eternal	Sculpture	has_material	"iron"	String
ns2#guernica	ExternalPage	Title	2000-06-09	Date

Πίνακας 3.1 Το αποτέλεσμα της ερώτησης 9

3.3 Τυπική ερμηνεία της RQL

Έχοντας παρουσιάσει τη γλώσσα επερωτήσεων RQL μέσω παραδειγμάτων στην προηγούμενη ενότητα, θα δώσουμε σε αυτή την ενότητα την τυπική ερμηνεία των διάφορων κατηγοριών ερωτήσεων που επιτρέπει. Πιο συγκεκριμένα, θα δώσουμε μια αλγεβρική ερμηνεία της RQL, που βασίζεται σε μια σχεσιακή αναπαράσταση των γράφων σχημάτων αλλά και δεδομένων RDF, που περιγράφεται στη συνέχεια. Έτσι, θα μπορέσουμε να αξιολογήσουμε λεπτομερώς τις ιδιότητες της γλώσσας (π.χ. πληρότητα, εκφραστική δύναμη) καθώς και να διακρίνουμε περιπτώσεις στις οποίες υπάρχει δυνατότητα *βελτιστοποίησης* στην αποτίμηση των ερωτήσεων. Όπως θα δείξουμε στη συνέχεια, η RQL μπορεί να ερμηνευτεί με εκφράσεις πρώτης τάξης πάνω από μία πεπερασμένη σχεσιακή δομή πολλών τύπων.

Πράγματι, με εξαίρεση τις λεκτικές τιμές και τα αναγνωριστικά πόρων, η ερμηνεία της RQL βασίζεται σε σχέσεις. Μάλιστα, τα αναγνωριστικά πόρων και τα ονόματα κλάσεων και ιδιοτήτων μπορούν να θεωρηθούν ως τιμές (συμβολοσειρές), τις οποίες συμβολίζουμε με το **val**. Τότε, ένας γράφος (μετα)δεδομένων RDFμπορεί να θεωρηθεί ως στιγμιότυπο του παρακάτω σχήματος (με ανώνυμες πλειάδες):

```
cls(val)
prop(val, val)
cont(val, val, val)
```

Εδώ, τα `cls` και `prop`αντιστοιχούν σε συγκεκριμένες κλάσεις και ιδιότητες κάποιου σχήματος και το `cont` στους τύπους συλλογών Bag, Seq και Alt αντίστοιχα. Τότε, η πλειάδα prop($r1$,$r2$) υποδεικνύει ότι τα $r1$, $r2$ είναι αναγνωριστικά πόρων, οι οποίοι συνδέονται μέσω μιας ακμής με ετικέττα `prop`. Επίσης, η πλειάδα cont($s1$, 1,$r1$) υποδεικνύει ότι υπάρχει μια συλλογή με αναγνωριστικό $s1$, που το πρώτο της μέλος είναι ο πόρος με αναγνωριστικό $r1$.

Σε αντίθεση με τα μοντέλα ημιδομημένων δεδομένων[ABS99] (που δεν περιγράφονται από σχήματα)[5], η ερμηνεία της RQL εκμεταλλεύται στο έπακρο τη σημασιολογία των ετικεττών κλάσεων ή ιδιοτήτων, που ορίζεται σε σχήματα RDF. Επίσης, και

[5] Το ίδιο ισχύει και γράφους του μοντέλου RDF που δεν περιγράφονται από κάποιο σχήμα και μπορούν να ερμηνευτούν μόνο με βάση τη μοναδική σχέση της μορφής(βλ. και αναπαράσταση με τριάδες, ενότητα 2.6): `statement(val, val, val)`

τα σχήματα RDF μπορούν να αναπαρασταθούν με σχεσιακό τρόπο, χρησιμοποιώντας τις σχέσεις `Class` και `Property` (βλ. και ενότητα 3.2) και τις σχέσεις `subclass` και `subproperty`, που περιέχουν την πληροφορία της μερικής διάταξης ($\prec$) πάνω στα προηγούμενα σύνολα, όπως αυτή ορίζεται από τις αντίστοιχες σχέσεις (`rdfs:subClassOf` και `rdfs:subPropertyOf`).

Με βάση τα παραπάνω, η ερμηνεία των βασικών ερωτήσεων της RQL σε σχήματα και μεταδεδομένα φαίνεται στον πίνακα 3.2 . Στον πίνακα αυτό, τα c, c_1, c_2 είναι ονόματα κλάσεων, το p είναι το όνομα μιας ιδιότητας, τα o, o_1, o_2 αναγνωριστικά πόρων, τα col, col_1, col_2 συλλογές και το $< v_1, v_2, \ldots >$ διατεταγμένες πλειάδες.

Στη συνέχεια θα προσπαθήσουμε να αναλύσουμε τις ερμηνείες των φίλτρων και των εκφράσεων μονοπατιών της RQL, που φαίνονται στον παραπάνω πίνακα. Για ευκολία, θα διακρίνουμε τις ερωτήσεις σε διάφορες κατηγορίες, με βάση:

◇ Την αναφορά τους σε μεταδεδομένα η σχήματα RDF.

◇ Την αναφορά τους σε κόμβους ή ακμές (μονοπάτια) σε ένα γράφο RDF.

Φίλτρο Συγκεκριμένης Κλάσης:

`select` X
`from` $X\ c$ $\qquad \{v_i \mid v_i \in [\![c]\!], v_i = l\}$
`where` $X = O$

Αυτό το είδος ερωτήσεων της RQL περιορίζει τα στιγμιότυπα που περιλαμβάνονται στην ερμηνεία της κλάσης c, σε αυτά που ικανοποιούν τη συνθήκη που περιγράφεται στο τμήμα του `where` της ερώτησης.

Φίλτρο Συγκεκριμένης Ιδιότητας (απλή έκφραση μονοπατιού): Στην κατηγορία αυτή μπορούμε να διακρίνουμε τρείς περιπτώσεις:

◇ *Απλή έκφραση μονοπατιού (μετα)δεδομένων*:

`select` X, Y
`from` $\{X\}p\{Y\}$ $\qquad \{< v_1, v_2 > \mid < v_1, v_2 > \in [\![p]\!], v_2 = l\}$
`where` $Y = O$

Ένα τέτοιο φίλτρο περιορίζει τα ζεύγη τιμών (μετα)δεδομένων (τιμές source/target) που αποτελούν την ερμηνεία της ιδιότητας p, θέτοντας συνθήκες στις τιμές/πόρους που εμφανίζονται στο δεύτερο μέλος της ερμηνείας (δηλ. στους κόμβους στους οποίους καταλήγει μια τέτοια ακμή). Κάτι αντίστοιχο μπορεί να γίνει βέβαια και για τους κόμβους από τους οποίους ξεκινά μια τέτοια ακμή.

◇ *Απλή έκφραση μονοπατιού σχήματος*:
select $\$X, \$Y$
from $\{\$X\}p\{\$Y\}$
where $\$Y = c$

$$\{< c_1, c_2 >| \; c_1, c_2 \in C, c_1 \prec domain(p), c_2 \prec range(p), c_1 = c\}$$

Η έκφραση αυτή ορίζει δύο μεταβλητές που παίρνουν τιμές από τις ιεραρχίες κλάσεων που αποτελούν τα πεδία ορισμού και τιμών της ιδιότητας p. Χωρίς τον περιορισμό στο τμήμα του where, το φίλτρο αυτό επιστρέφει το καρτεσιανό γινόμενο των δύο συνόλων-ιεραρχιών. Με την υπάρχουσα συνθήκη, οι πλειάδες του αποτελέσματος περιορίζονται σε αυτές που ικανοποιούν την αντίστοιχη συνθήκη.

◇ *Απλή έκφραση "μικτού" μονοπατιού (σχήματος και δεδομένων)*:
select $X, Y, \$Z, \$W$
from $\{X : \$Z\}p\{Y : \$W\}$
where $Y = O$ and $\$W = c$

$$\begin{aligned}\{< v_1, v_2 >| \quad & \exists p \in P, c_1, c_2 \in C, c_1 \prec domain(p), v_1 \in [\![c_1]\!], \\ & c_2 \prec range(p), v_2 \in [\![c_2]\!], < v_1, v_2 > \in [\![p]\!]\}\end{aligned}$$

Αυτή η έκφραση αποτελεί ουσιαστικά συνδυασμό των δύο προηγούμενων. Ως αποτέλεσμα επιστρέφει τα μονοπάτια σχήματος τα οποία θα επέστρεφε και η αντίστοιχη έκφραση μονοπατιού σχήματος, καθώς και τα αντίστοιχα μεταδεδομένα-στιγμιότυπα των κλάσεων. Ως επιπλέον περιορισμό, στις πλειάδες που επιστρέφονται από μια τέτοια έκφραση τα μεταδεδομένα στα άκρα της ακμής "οφείλουν" να είναι άμεσα στιγμιότυπα των αντίστοιχων κλάσεων στην ίδια πλειάδα. Αυτό βέβαια είναι ισοδύναμο με μια απλή έκφραση μονοπατιού (μετα)δεδομένων, αφού όλα τα έγκυρα ζεύγη τιμών της ιδιότητας αυτής θα πρέπει να αποτελούνται από στιγμιότυπα κλάσεων που συνδέονται μέσω της δεδομένης ιδιότητας στο σχήμα. Όμως, όπως θα εξηγήσουμε αναλυτικά στη συνέχεια, δεν ισχύει το ίδιο στην περίπτωση της σύνθεσης απλών εκφράσεων μονοπατιών. Επίσης, η ύπαρξη μεταβλητών σχήματος και η "εξάρτηση" των δεδομένων από αυτές μας επιτρέπει να "φιλτράρουμε" τα (μετα)δεδομένα με βάση πληροφορία σχήματος (ζητώντας, π.χ. μόνο τις πλειάδες στις οποίες ο πόρος σε κάποιο άκρο της ακμής είναι άμεσο στιγμιότυπο κάποιας κλάσης c, όπως στο παραπάνω παράδειγμα).

3.3.1 Γενικευμένες εκφράσεις μονοπατιών

Σε όλα τα προηγούμενα παραδείγματα μονοπατιών χρησιμοποιήσαμε το όνομα μιας συγκεκριμένης ιδιότητας ως κύριο συνθετικό τους. Η RQL, με τη χρήση των γενικευμένων εκφράσεων μονοπατιών, επιτρέπει επίσης τη χρήση μεταβλητών, οι οποίες παίρνουν ονόματα ιδιοτήτων ως τιμές. Ακολουθώντας την προσέγγιση του ίδιου του μοντέλου του RDF, σε τέτοιες περιπτώσεις δίνουμε προτεραιότητα στην αποτίμηση των μεταβλητών ιδιοτήτων. Στη συνέχεια, για κάθε τιμή αυτών των μεταβλητών, οι υπόλοιπες μεταβλητές (κόμβων) του μονοπατιού αποτιμούνται όπως ακριβώς αναλύθηκε κατά περίπτωση προηγουμένως, στα μονοπάτια που περιέχουν μια συγκεκριμένη ιδιότητα. Έτσι, ένα παράδειγμα γενικευμένης έκφρασης μονοπατιού δεδομένων είναι:

`select` X, Y
`from` $\{X\}\$P\{Y\}$

$$\{< v_1, v_2 >\mid \exists p \in P, < v_1, v_2 >\in \hat{} [\![p]\!]\}$$

3.3.2 Σύνθεση εκφράσεων μονοπατιών

Ένα από τα πιο ισχυρά χαρακτηριστικά της RQL είναι η δυνατότητα σύνθεσης απλών εκφράσεων μονοπατιών. Συνδυάζοντας τις παραπάνω απλές εκφράσεις μέσω του τελεστή "." μπορούμε να σχηματίσουμε σύνθετες εκφράσεις μονοπατιών. Εκτός από την προφανή δυνατότητα συνδυασμού εκφράσεων "ίδιου τύπου" (με βάση την προηγούμενη κατηγοριοποίηση), η RQL επιτρέπει και τη σύνθεση μονοπατιών διαφορετικών τύπων, με τις κατάλληλες συνθήκες σε κάθε περίπτωση:

◇ *Σύνθεση μονοπατιών μεταδεδομένων*:
`select` X, Y, Z
`from` $\{X\}p\{Y\}.q\{Z\}$

$$\{< v_1, v_2, v_3 > | < v_1, v_2 > \in [\![p]\!], < v_2, v_3 > \in [\![q]\!]\}$$

Οι εκφράσεις αυτές αναφέρονται μόνο στο γράφο μεταδεδομένων (και όχι σε μονοπάτια στο σχήμα που τα περιγράφει). Η ερμηνεία τέτοιων εκφράσεων περιλαμβάνει την ένωση των ερμηνειών των δύο εκφράσεων, με βάση την ισότητα των τιμών στα κατάλληλα άκρα των ακμών που "συνενώνονται". Ουσιαστικά η έκφραση αυτή είναι ισοδύναμη με την παρακάτω ερώτηση:

`select` $X, Y, P[2]$
`from` $\{X\}p\{Y\}$
`where` `exists` P {`select` W, Z
`from` $\{W\}q\{Z\}$
}: $P[1] = Y$

◇ *Σύνθεση μονοπατιών σχήματος*:
`select` $\$X, \$Y, \$Z$
`from` $\{\$X\}p\{\$Y\}.q\{\$Z\}$

$$\begin{aligned}\{< c_1, c_2, c_3 > | \quad & c_1, c_2 \in C, c_1 \prec domain(p), c_2 \prec range(p), \\ & c_3 \prec range(q), c_4 \prec domain(q), c_2 = c_4\}\end{aligned}$$

Η ερμηνεία αυτών των εκφράσεων σχηματίζεται από την σύνθεση των ερμηνειών των δύο απλών εκφράσεων, με τη συνθήκη ότι η κλάση στο δεξί άκρο της πρώτης έκφρασης πρέπει να ισούται με την αντίστοιχη στο αριστερό άκρο της δεύτερης. Η έκφραση αυτή είναι ισοδύναμη με την παρακάτω ερώτηση:

```
select  $X, $Y, P
from    {$X}p{$Y}
where   exists P {select  $W, $Z
                  from    {$W}q{$Z}
                  }: P[1] = $Y
```

◇ *Σύνθεση μικτών μονοπατιών*:

select $X, \$C_1, Y, \$C_2, Z, \$C_3$

from $\{X : \$C_1\}p\{Y : \$C_2\}.q\{Z : \$C_3\}$

$$\{< v_1, c_1, v_2, c_2, v_3, c_3 >| \quad c_1, c_2, c_3, c_4 \in C, \\ c_1 \prec domain(p), v_1 \in\hat{} [\![c_1]\!], c_2 \prec range(p), v_2 \in\hat{} [\![c_2]\!], \\ c_3 \prec range(q), v_3 \in\hat{} [\![c_3]\!], c_4 \prec domain(q), v_4 \in\hat{} [\![c_4]\!], \\ < v_1, v_2 >\in [\![p]\!], < v_4, v_3 >\in [\![q]\!], c_2 = c_4, v_2 = v_4\}$$

Και σε αυτή την περίπτωση η ερμηνεία των εκφράσεων προκύπτει από τη σύνθεση των ερμηνειών των δύο απλών μονοπατιών. Όμως, η συνθήκη "ένωσής" τους προκύπτει τόσο από την ένωσή τους ως μονοπάτια σχήματος όσο και από την αντίστοιχη ένωση ως μονοπάτια δεδομένων. Ουσιαστικά η έκφραση αυτή είναι ισοδύναμη με την παρακάτω ερώτηση:

```
select  X, $C1, Y, $C2, P[3], P[4]
from    {X : $C1}p{Y : $C2}
where   exists P {select  W, $C4, Z, $C3
                  from {W : $C4}q{Z : $C3}
                  }: P[1] = Y and P[2] = $C2
```

◇ *Σύνθεση μονοπατιών διαφορετικών τύπων*:

Όπως αναφέρθηκε προηγουμένως, η RQL επιτρέπει επίσης τη σύνθεση απλών μονοπατιών με διαφορετικούς. Με βάση την παραπάνω κατηγοριοποίηση μπορούμε να διακρίνουμε τρεις δυνατούς συνδυασμούς:

▷ **Μικτό μονοπάτι/Μονοπάτι δεδομένων**: Η σύνδεση γίνεται με βάση μόνο τις τιμές δεδομένων, δηλαδή με την επιπλέον συνθήκη $v_2 = v_3$, όπως φαίνεται και στην ερμηνεία της ερώτησης:

select $X, \$C_1, Y, \$C_2, Z, W$

from $\{X : \$C_1\}p\{Y : \$C_2\}.\{Z\}q\{W\}$

$$\{< v_1, c_1, v_2, c_2, v_3, v_4 >| \quad \begin{aligned}[t] & c_1, c_2 \in C, \\ & c_1 \prec domain(p), v_1 \in\hat{} [\![c_1]\!], c_2 \prec range(p), v_2 \in\hat{} [\![c_2]\!], \\ & < v_1, v_2 > \in [\![p]\!], < v_3, v_4 > \in [\![q]\!], v_2 = v_3\} \end{aligned}$$

Παρατηρούμε ότι το σύνθετο μονοπάτι που προκύπτει δίνει στο αριστερό άκρο του τόσο τιμές δεδομένων (v_1) όσο και σχήματος (c_1), συμπεριφέρεται δηλαδή - όσον αφορά τη σύνθεση με άλλα μονοπάτια - ως μικτό μονοπάτι από αριστερά και ως μονοπάτι δεδομένων από δεξιά.

▷ **Μικτό μονοπάτι/Μονοπάτι σχήματος**: Η σύνδεση γίνεται με βάση μόνο τις τιμές σχήματος (ονόματα κλάσεων) για τους αντίστοιχους κόμβους, δηλαδή με την επιπλέον συνθήκη $c_2 = c_4$, όπως φαίνεται και στην ερμηνεία της ερώτησης:

select $X, \$C_1, Y, \$C_2, \$C_3, \$C_4$

from $\{X : \$C_1\}p\{Y : \$C_2\}.\{\$C_3\}q\{\$C_4\}$

$$\{< v_1, c_1, v_2, c_2, c_3, c_4 >| \quad \begin{aligned}[t] & c_1, c_2 \in C, \\ & c_1 \prec domain(p), v_1 \in\hat{} [\![c_1]\!], c_2 \prec range(p), v_2\hat{} \in [\![c_2]\!], \\ & < v_1, v_2 > \in [\![p]\!], c_3 \prec domain(q), c_4 \prec range(q), \\ & c_2 = c_3\} \end{aligned}$$

▷ **Μονοπάτι δεδομένων/Μονοπάτι σχήματος**: Στην περίπτωση αυτή δεν μπορεί να χρησιμοποιηθεί καμμία από τις συνθήκες που χρησιμοποιήσαμε στη συνένωση μονοπατιών ίδιου τύπου. Η κατάλληλη συνθήκη, αν v_2 είναι οι τιμές δεδομένων για τους κόμβους προορισμού του μονοπατιού δεδομένων και c_4 οι τιμές σχήματος για τους κόμβους προέλευσης του μονοπατιού δεδομένων, είναι η $v_2 \in\hat{} [\![c_4]\!]$.

select $X, Y, \$C_3, \$C_4$

from $\{X\}p\{Y\}.\{\$C_3\}q\{\$C_4\}$

$$\begin{aligned}\{< v_1, v_2, c_3, c_4 >| \quad & c_1, c_2 \in C, \\ & < v_1, v_2 > \in [\![p]\!], c_3 \prec domain(q), c_4 \prec range(q), \\ & v_2 \in\hat{} [\![c_3]\!]\}\end{aligned}$$

3.4 Τύποι των εκφράσεων της RQL

Στην ενότητα αυτή θα δείξουμε την εξαγωγή τύπων (type inference) στις εκφράσεις της RQL που παρουσιάσαμε προηγουμένως, με βάση το σύστημα τύπων που παρουσιάσαμε στην ενότητα 3.1. Έτσι, οι τύποι των απλών εκφράσεων - η ερμηνεία των οποίων παρουσιάστηκε στον πίνακα 3.2 - φαίνεται στους πίνακες 3.4 (για τις μοναδιαίες εκφράσεις) και 3.5 (για τις δυαδικές εκφράσεις). Για να εκφράσουμε τους κανόνες εξαγωγής τύπων χρησιμοποιούμε τον εξής συμβολισμό: οι κανόνες πάνω από την οριζόντια γραμμή αποτελούν τις προϋποθέσεις (συνθήκες) που πρέπει να ισχύουν, ώστε η έκφραση να είναι έγκυρη. Στο κάτω μέρος φαίνεται η έκφραση και ο τύπος της. Έτσι, για παράδειγμα, για τη λειτουργία $target(e)$ η έκφραση $e : property_U$ σημαίνει ότι ο τύπος της παραμέτρου e είναι $property_U$ (η οποία είναι και η συνθήκη εγκυρότητας της έκφρασης). Αν αυτή η συνθήκη ισχύει, τότε ο τύπος της έκφρασης είναι $(resource_U, \tau_U)$. Χρησιμοποιούμε το συμβολισμό που εισάγαμε στην παρουσίαση του συστήματος τύπων, στην ενότητα 3.1, για πολυσύνολα $(\{.\})$ και τύπους ένωσης $(.)$.

Ιδιαίτερο ενδιαφέρον εμφανίζει η περίπτωση της ερμηνείας μιας ιδιότητας (και προφανώς η λειτουργία `target(p)` που επιστρέφει τις τιμές προορισμού μιας ιδιότητας): δεδομένου ότι ο τύπος του πεδίου τιμών της μπορεί να είναι είτε ένα αναγνωριστικό πόρου είτε ατομικός τύπος, εισάγουμε την έννοια της *ένωσης τύπων*. Δεδομένου ότι τέτοιες εκφράσεις εμφανίζονται κυρίως σε εκφράσεις μονοπατιών, θα παρουσιάσουμε ένα χαρακτηριστικό παράδειγμα στην επόμενη ενότητα.

3.4.1 Βασικά φίλτρα - απλές εκφράσεις μονοπατιών

Στην ενότητα αυτή θα παρουσιάσουμε την εξαγωγή τύπων για το βασικό φίλτρο της RQL και τα φίλτρα που περιλαμβάνουν απλές εκφράσεις μονοπατιών. Οι εκφράσεις μονοπατιών μπορούν να χωριστούν σε τρεις κατηγορίες: μονοπάτια δεδομένων, μονοπάτια σχήματος και μικτά μονοπάτια - που συνδυάζουν επερώτηση δεδομένων και σχήματος. Οι τύποι αυτών των εκφράσεων φαίνονται στον πίνακα 3.6.

Όπως βλέπουμε στον πίνακα αυτό, ο τύπος για ένα μονοπάτι δεδομένων είναι το ζεύγος των τύπων των πεδίων ορισμού και τιμών της ιδιότητας που περιέχει. Ιδιαίτερο ενδιαφέρον παρουσιάζουν εκφράσεις που περιέχουν μεταβλητή στη θέση της ιδιότητας, όπως στο παρακάτω παράδειγμα:

select $X, \$P, Y$
from $\{X\}\$P\{Y\}$

Το αποτέλεσμα αυτής της ερώτησης φαίνεται στον πίνακα 3.3. Στον πίνακα αυτό, η στήλη X έχει τύπο $class_U$, ενώ η δεύτερη $property_U$. Όμως, η τρίτη στήλη περιέχει και στοιχεία τύπου $resource_U$ και κάποιων από τους ατομικούς τύπους $(string, date, ...)$. Γι' αυτό, το αποτέλεσμα της παραπάνω ερώτησης έχει τύπο: $[resource_U, property_U, (resource_U \mid \tau_B)]$.

Εξάλλου, ο τύπος για ένα μονοπάτι σχήματος είναι ένα ζεύγος που αποτελείται από ένα όνομα κλάσης (για το πεδίο ορισμού της ιδιότητας) και ένα όνομα κλάσης ή ένα από τα ονόματα των βασικών τύπων, τ_B (που υποδηλώνει το πεδίο τιμών της ιδιότητας).

Στην περίπτωση των μικτών μονοπατιών, ορίζουμε μεταβλητές τόσο στο σχήμα όσο και στα δεδομένα. Έτσι, ο τύπος ενός τέτοιου μονοπατιού έχει τέσσερα στοιχεία: το πρώτο και το τρίτο αναφέρονται στις μεταβητές σχήματος (όπως σε ένα απλό μονοπάτι σχήματος) ενώ το δεύτερο και το τέταρτο στις μεταβλητές δεδομένων.

Για την περίπτωση της συνένωσης μονοπατιών, ο έλεγχος τύπων ανάγεται στον έλεγχο τύπων των εκφράσεων στις οποίες μεταφράζεται. Πράγματι, όπως εξηγήσαμε στην προηγούμενη ενότητα, η ερώτηση:

select X, Y, Z
from $\{X\}p\{Y\}.q\{Z\}$

είναι ισοδύναμη με την:

select X, Y, Z
from $\{X\}p\{Y\}$
where exists P $\{$select W
from $\{W\}q\{Z\}$
$\}$: $P = Y$

Έτσι, ουσιαστικά ο έλεγχος τύπων που απαιτείται για τη σύνθεση των δύο εκφράσεων είναι ο έλεγχος της εγκυρότητας της σύγκρισης $P = Y$, που γίνεται με βάση τον αντίστοιχο κανόνα που φαίνεται στον πίνακα 3.5.

3.5 Συμπεράσματα

Ως συμπέρασμα, θα εξηγήσουμε πως η RQL παρέχει την απαραίτητη λειτουργικότητα για να καλύψει τις ανάγκες μιας πύλης κοινοτήτων διαδικτύου. Πράγματι, η δυνατότητα απλής αναδίφυσης, χωρίς διάσχιση ιεραρχιών - όπως αυτή που παρέχεται από τις υπάρχουσες πύλες διαδικτύου, δίνεται από τις βασικές ερωτήσεις της RQL. Εξάλλου, η περίπτωση της προσαρμογής στις ανάγκες και προτιμήσης κάθε μέλους της κοινότητας, π.χ. με τη δημιουργία προσωποποιημένων καναλιών, απαιτεί πληροφορία που μπορεί να συγκεντρωθεί ως το αποτέλεσμα μιας ερώτησης RQL. Πράγματι, τέτοια πληροφορία υπάρχει με τη μορφή περιγραφών των πόρων, πιθανώς με βάση περισσότερα από ένα σχήματα, για διαφορετικά είδη πληροφορίας (διαχειριστική, περιεχομένου κλπ). Η RQL είναι επίσης ιδιαίτερα χρήσιμη για τη διαλογική πλοήγηση σε πύλες κοινοτήτων διαδικτύου. Πράγματι, ενώ το μέγεθος των ιεραρχιών κατηγοριών/θεμάτων καθιστά τέτοιες λειτουργίες απαγορευτικές στις υπάρχουσες πύλες διαδικτύου, η RQL παρέχει τη δυνατότητα σχηματισμού πιο περίπλοκων και πλούσιων ερωτήσεων, που την καθιστούν εφικτή. Τέλος, η RQL μπορεί να χρησιμοποιηθεί σε ένα κατανεμημένο περιβάλλον, για την ανταλλαγή πληροφορίας μεταξύ διαφορετικών εξυπηρετητών υποκοινοτήτων της ίδιας κοινότητας ή διαφορετικών κοινοτήτων. Ως παράδειγμα παραθέτουμε την ερώτηση 9 - που παρουσιάστηκε προηγουμένως - η οποία επιστρέφει όλες τις περιγραφές που σχετίζονται με ένα δικτυακό τόπο (οι οποίες μπορεί να "ανήκουν" σε μια άλλη κοινότητα).

Επίσης, δείξαμε ότι οι ερωτήσεις της RQL ερμηνεύονται τυπικά με ομοιόμορφο τρόπο. Πράγματι, οι ερωτήσεις της RQL μπορούν να ερμηνευτούν με τη χρήση λογικής πρώτης τάξης πάνω από μια δομή που μοιάζει με μια πεπερασμένη, σχεσιακή δομή πολλών τύπων. Συνεπώς, όπως θα εξηγήσουμε και πιο αναλυτικά στο κεφάλαιο 4, η υλοποίηση της αποθήκευσης με βάση το μοντέλο μας καθώς και της RQL πάνω από σχεσιακό σύστημα βάσεων δεδομένων είναι εφικτή.

Expression	Interpretation
`Class`	$C \subseteq \mathcal{C}$
`Property`	$P \subseteq \mathcal{P}$
`Domain(p)`	$domain(p)$
`Range(p)`	$range(p)$
`subClassOf(c)`	$\{c' \in C \mid c' \prec c\}$
`subClassOf^(c)`	$\{c' \in C \mid c' \prec c, \nexists c'' \in C, c'' \prec c \& c' \prec c''\}$
`superClassOf(c)`	$\{c' \in C \mid c \prec c'\}$
`superClassOf^(c)`	$\{c' \in C \mid c \prec c', \nexists c'' \in C, c \prec c'' \& c'' \prec c'\}$
`subPropertyOf(p)`	$\{p' \in P \mid p' \prec p\}$
`subPropertyOf^(p)`	$\{p' \in P \mid p' \prec p, \nexists p'' \in P, p'' \prec p \& p' \prec p''\}$
`superPropertyOf(p)`	$\{p' \in P \mid p \prec p'\}$
`superPropertyOf^(p)`	$\{p' \in P \mid p \prec p', \nexists p'' \in P, p \prec p'' \& p'' \prec p'\}$
`o`	o
`typeOf(o)`	$\tau(v)$, where $\nu(v) = o$
`c`	$[\![c]\!]$
`^c`	$[\![c]\!] - \bigcup_{c' \prec c}([\![c']\!])$
`p`	$[\![p]\!]$
`^p`	$[\![p]\!] - \bigcup_{p' \prec p}([\![p']\!])$
`source(p)`	$\{v_1\}, < v_1, v_2 > \in [\![p]\!]$
`target(p)`	$\{v_2\}, < v_1, v_2 > \in [\![p]\!]$
col_1 `intersect` col_2	$[\![col_1]\!] \cap [\![col_2]\!]$
col_1 `union` col_2	$[\![col_1]\!] \cup [\![col_2]\!]$
col_1 `minus` col_2	$[\![col_1]\!] - [\![col_2]\!]$
`count(`col`)`	n, if $\exists i_1, \ldots, i_n, \exists v_{i_1}, \ldots, v_{i_n}$, $\forall j, i_1 \leq j \leq i_n \ (j, v_j) \in [\![col]\!]$
o `in` col	$\exists i, 1 \leq i \leq count(col), (i, o) \in [\![col]\!]$
$seq[n:m]$	$< v_n, \ldots, v_m >$ where $\exists j_1, \ldots, j_l, 1 \leq n \leq m \leq l$, $\exists v_1, \ldots, v_{(n-1)}, \ldots, v_{(m+1)}, \ldots, v_l, \forall i, 1 \leq i \leq l, (j_i, v_i) \in [\![seq]\!]$

Πίνακας 3.2 Ερμηνεία των βασικών ερωτήσεων της RQL

Domain:	$$\frac{e : property_U}{domain(e) : class_U}$$
Range:	$$\frac{e : property_U, \tau = (class_U \mid literal)}{range(e) : \tau}$$
Sub/SuperClassesOf:	$$\frac{e : class_U}{subClassesOf(e) : \{class_U\}}$$
Sub/SuperPropertiesOf:	$$\frac{e : property_U}{subPropertiesOf(e) : \{property_U\}}$$
Property type p :	$\{[resource_U, (resource_U, \tau_B)]\}$
Class c :	$\{resource_U\}$
Source:	$$\frac{e : property_U}{source(e) : \{resource_U\}}$$
Target:	$$\frac{e : property_U, \tau = (resource_U \mid \tau_B)}{target(e) : \{\tau\}}$$
Proper instances of a class:	$$\frac{e : class_U}{(\hat{\ } e) : \{resource_U\}}$$
Proper instances of a property:	$$\frac{e : property_U}{(\hat{\ } e) : \{[resource_U, (resource_U, \tau_B)]\}}$$
	$$\frac{\text{count: } e : \tau, \tau = (\{\tau\} \mid [\tau])}{count(e) : int}$$

Σχήμα 3.4 Μοναδιαίες λειτουργίες

ith: $$\frac{e : [\tau_1, \tau_2, \ldots, \tau_n], \quad i : integer, i \in [1..n]}{(e[i]) : \tau_i}$$

comp: $$\frac{e : \tau, e' : \tau', \tau = \tau', \tau \neq union(\tau), \theta =, <, >, \leq, \geq}{(e\ \theta\ e') : boolean}$$

in: $$\frac{x : \tau, e : \{\tau\}}{(x\ in\ e) : boolean}$$

like: $$\frac{x : \tau, e : string, \tau = (string \mid \tau_u)}{(x\ like\ e) : boolean}$$

set operations : $$\frac{col_1 : \{\tau\}, col_2 : \{\tau'\}, \tau = \tau', \theta = (intersect \mid union \mid minus)}{(col_1\ \theta\ col_2) : \{\tau\}}$$

$$\frac{col_1 : [\tau], col_2 : [\tau'], \tau = \tau', \theta = (intersect \mid union \mid minus)}{(col_1\ \theta\ col_2) : [\tau]}$$

Σχήμα 3.5 Δυαδικές λειτουργίες

Βασικό φίλτρο:	$$\frac{e : bag(\tau), e' : \tau', p : boolean}{(\text{select } e' \text{ from } x\ e \text{ where } p) : bag(\tau'), x : \tau}$$
Μονοπάτι δεδομένων:	$$\frac{e : property_U, source(e) : \{\tau_1\}, target(e) : \{\tau_2\}}{(\text{select } x, y \text{ from } \{x\}e\{y\}) : \{[\tau_1, \tau_2]\}, x : \tau_1, y : \tau_2}$$
Μονοπάτι σχήματος:	$$\frac{e : property_U, domain(e) : c_{\tau_1}, range(e) : c_{\tau_2}}{\begin{array}{c}(\text{select } \$c_1, \$c_2 \text{ from } \{\$c_1\}e\{\$c_2\}) : \\ \{[c_{\tau_1}, c_{\tau_2}]\}, \\ c_1 : c_{\tau_1}, c_2 : c_{\tau_2}\end{array}}$$
Μικτό μονοπάτι:	$$\frac{\begin{array}{c}e : property_U, domain(e) : c_{\tau_1}, source(e) : \{\tau_1\}, \\ range(e) : c_{\tau_2}, target(e) : \{\tau_2\}\end{array}}{\begin{array}{c}(\text{select } \$c_1, x, \$c_2, y \text{ from } \{x : \$c_1\}e\{y : \$c_2\}) : \\ \{[c_{\tau_1}, \tau_1, c_{\tau_2}, \tau_2]\}, \\ c_1 : c_{\tau_1}, x : \tau_1, c_2 : c_{\tau_2}, y : \tau_2\end{array}}$$

Σχήμα 3.6 Βασικά φίλτρα

X	$\$P$	Y
ns2#rodin	creates	ns2#crucifixion
ns2#rodin	creates	ns2#crucifixion
ns2#picasso	paints	ns2#guernica
ns2#picasso	paints	ns2#woman
ns2#claudel	sculpts	ns2#eternalidol
ns2#picasso	first_name	"Pablo"
ns2#claudel	first_name	"Camille"
ns2#picasso	last_name	"Picasso"
ns2#rodin	last_name	"Rodin"
ns2#rodin	last_name	"Rodin"
ns2#claudel	last_name	"Claudel"
ns2#guernica	has_style	ns3#cubism
ns2#guernica	using_technique	"oil on canvas"
ns2#woman	using_technique	"oil on canvas"
ns2#eternal	has_material	"iron"
ns2#guernica	Title	2000-06-09

Πίνακας 3.3 Το αποτέλεσμα της ερώτησης

Υλοποίηση του μεταφραστή της RQL

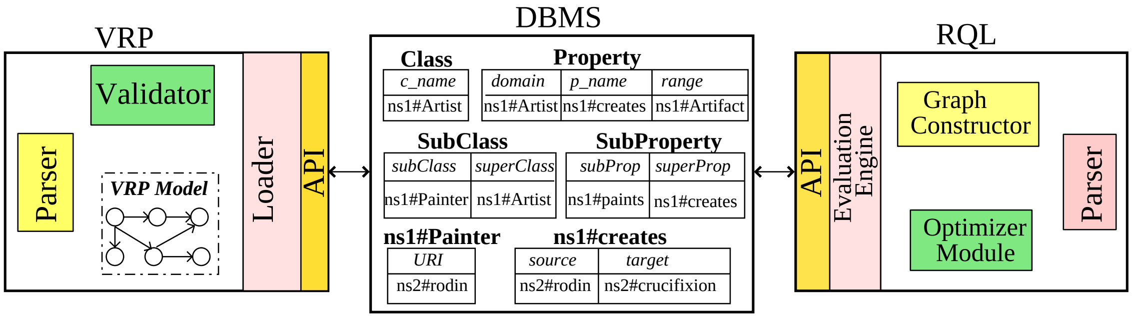
4.1 Αποθήκευση μεταδεδομένων - Δομή του συστήματος

Όπως εξηγήσαμε στο προηγούμενο κεφάλαιο, η άλγεβρα της RQL αποδεικνύει ότι η γλώσσα μας μπορεί να υλοποιηθεί πάνω από μία πεπερασμένη σχεσιακή δομή πολλών τύπων. Έτσι, επιλέξαμε και για λόγους απλότητας, να υλοποιήσουμε την αποθήκευση και επερώτηση των ένα RDF πάνω από ένα κοινό σχεσιακό σύστημα βάσεων δεδομένων. Συγκεκριμένα, χρησιμοποιήσαμε το σύστημα Postgres [SJGP90], [Mom00], που είναι και το πιο διαδεδομένο σύστημα που διανέμεται ελεύθερα (open-source). Η Postgres, όπως και σχεδόν όλα τα σύγχρονα σχεσιακά συστήματα, είναι ένα σχεσιακό-οντοκεντρικό [SBM99a] σύστημα διαχείρισης βάσεων δεδομένων, που επιτρέπει να ορίσουμε σχέσεις υπαλληλίας μεταξύ πινάκων. Στη συνέχεια, παρέχει τρόπους ανάκτησης πληροφορίας από τέτοιες ιεραρχίες, γεγονός απαραίτητο για περιπτώσεις όπως η επέκταση ερωτήσεων της RQL. Ακολουθώντας τη λογική της σχεσιακής αναπαράστασης που παρουσιάσαμε στο προηγούμενο κεφάλαιο, ορίσαμε για την αποθήκευση το σχεσιακό σχήμα που φαίνεται στο σχήμα 4.1.

Το σχεσιακό αυτό σχήμα - σε αντίθεση με τα υπόλοιπα συστήματα αποθήκευσης μεταδεδομένων RDF που χρησιμοποιούν μία σχέση για την αποθήκευση περιγραφών RDF με τη μορφή τριάδων (βλ. ενότητα 2.6) - εξαρτάται από το χρησιμοποιούμενο σχήμα RDF και δημιουργείται κατά τη διάρκεια της φόρτωσης στη βάση δεδομένων. Η φόρτωση αυτή γίνεται από μία εφαρμογή που έχει υλοποιηθεί ως επέκταση του VRP[1] (Συντακτικός αναλυτής RDF με έλεγχο εγκυρότητας - Validating RDF Parser), που είναι ο μοναδικός συντακτικός αναλυτής για RDF που ελέγχει τόσο τη συντακτική όσο και τη σημασιολογική εγκυρότητα των περιγραφών, σε σχέση με το χρησιμο-

[1] www.ics.forth.gr/proj/isst/RDF

ποιούμενο σχήμα. Πιο συγκεκριμένα, το σχήμα αυτό περιλαμβάνει τις σχέσεις Class,



Σχήμα 4.1 Το σχεσιακό σχήμα αποθήκευσης μεταδεδομένων και σχημάτων RDF

Property, subClass, subProperty που λειτουργούν ως κατάλογος της βάσης δεδομένων (καθώς περιέχουν πληροφορία για το υπόλοιπο σχήμα αποθήκευσης). Στη συνέχεια, κάθε κλάση παριστάνεται από μία μοναδιαία σχέση (που περιλαμβάνει τα αναγνωριστικά πόρων (uri) των στιγμιοτύπων της κλάσης αυτής) ενώ κάθε ιδιότητα από μία δυαδική (που περιέχει τα ζεύγη τιμών προέλευσης (source) και προορισμού (target) της ιδιότητας αυτής). Για παράδειγμα, στο σχήμα 4.1, ο πίνακας ns1#Painter περιέχει τα άμεσα στιγμιότυπα της κλάσης Painter, ενώ ο πίνακας ns1#creates τα ζεύγη πόρων που συνδέονται μέσω της ιδιότητας creates. Μάλιστα, οι πίνακες που αναπαριστούν κλάσεις και ιδιότητες αντίστοιχα είναι οργανωμένοι σε ιεραρχίες *υποπινάκων*, με βάση τη σχέση υπαλληλίας μεταξύ τους, που παρέχουν τα οντοκεντρικά-σχεσιακά ΣΔΒΔ. Περισσότερες λεπτομέρειες για την αποθήκευση μεταδεδομένων RDF υπάρχουν στο [Ale00].

Στη συνέχεια, ο διερμηνέας της RQL μπορεί να υλοποιηθεί ως εξωτερική εφαρμογή της βάσης αυτής, χρησιμοποιώντας τα προγραμματιστικά περιβάλλοντα διεπαφής που παρέχει, για την επερώτηση μεταδεδομένων RDF. Προκειμένου να διαχωριστεί η υλοποίηση του διερμηνέα της γλώσσας από την επιλογή της συγκεκριμένης βάσης για την υλοποίηση του συστήματος, ορίσαμε ένα *Περιβάλλον Προγραμματισμού Εφαρμογών Επερώτησης* (Query API) που περιλαμβάνει όλες τις στοιχειώδεις λειτουργίες που χρειάζεται ο διερμηνέας της RQL για να χρησιμοποιηθεί πάνω από μια βάση μεταδεδομένων. Οι λειτουργίες αυτές και η υλοποίησή τους θα περιγραφεί στη συνέχεια αυτού του κεφαλαίου.

4.2 Δομή του Διερμηνέα Ερωτήσεων

Ο διερμηνέας ερωτήσεων της *RQL* ακολουθεί μια συναρτησιακή προσέγγιση (functional approach), που επιτρέπει τη σύνθεση φίλτρων και συναρτήσεων για το σχηματισμό πολύπλοκων ερωτήσεων. Βασίζεται αρκετά στη συναρτησιακή προσέγγιση αλλά και στη σχεδίαση του διερμηνέα ερωτήσεων της OQL [AK89]. Εσωτερικά, χρησιμοποιεί μια σημασιολογική αναπαράσταση με τα εξής χαρακτηριστικά:

◇ Μια ερώτηση παριστάνεται από ένα γράφο. Οι κόμβοι του γράφου αυτού αναπαριστούν *λειτουργίες* (operations) ενώ οι ακμές αναπαριστούν *σχέσεις εξάρτησης* (dependency links). Ο γράφος αυτός χωρίζεται σε ζώνες που ορίζουν την ιεραρχία των εξαρτήσεων.

◇ Για τις σταθερές εκφράσεις, καθώς και γι'αυτές που εμφανίζονται περισσότερες από μία φορά φροντίζουμε να μην αποτιμούνται περισσότερες φορές από όσες χρειάζεται. Αυτό επιτυγχάνεται μέσω της *παραγοντοποίησης κοινών εκφράσεων* (factorization).

◇ Κάθε στοιχείο της αναπαράστασης έχει ένα τύπο και μια αυτόνομη σημασιολογία (όσον αφορά την πληροφορία που το συνοδεύει αλλά και τις διάφορες λειτουργίες που μπορούμε να επιτελέσουμε σε αυτό). Οι iterators παριστάνονται από αλγεβρικούς τελεστές που μπορούν να αναδιαταχθούν κατά τη φάση της βελτιστοποίησης μιας ερώτησης. Μετά το τέλος και αυτής της φάσης, κάθε λειτουργία "γνωρίζει" τη συνάρτηση που πρέπει να χρησιμοποιηθεί για την αποτίμησή της.

Ο διερμηνέας της *RQL* αποτελείται από 6 διακριτά τμήματα-ενότητες (modules), τα οποία ουσιαστικά είναι υπεύθυνα για τις διάφορες αντίστοιχες φάσης μετάφρασης μιας ερώτησης, όπως θα δείξουμε στη συνέχεια του κεφαλαίου:

◇ Ο *συντακτικός αναλυτής* (syntax analyser) παίρνει ως είσοδο μια συμβολοσειρά που αναπαριστά μια ερώτηση και - αφού ελέγξει τη συντακτική της ορθότητα - επιστρέφει το συντακτικό δέντρο της ερώτησης αυτής. Οι λογικοί τελεστές (and, or, not) σε αυτό το δέντρο μετατρέπονται πάντα σε συζευκτική κανονική μορφή.

◇ Ο *κατασκευαστής γράφου* (graph constructor), ο οποίος παίρνει ως είσοδο το συντακτικό δέντρο που επιστρέφει ο συντακτικός αναλυτής και επιστρέφει ένα γράφο που αντιστοιχεί στη σημασιολογική αναπαράσταση που παρουσιάσαμε παραπάνω. Η ενότητα αυτή περιέχει λειτουργίες κατασκευής γράφου, παραγοντοποίησης κοινών εκφράσεων και υπολογισμού των συσχετίσεων μεταξύ λειτουργιών, ενώ χρησιμοποιεί συναρτήσεις των δύο επόμενων για τα υπόλοιπα, δηλ. την ανάθεση τύπων και την βελτιστοποίηση των λειτουργιών του γράφου.

◇ Η ενότητα *ανάθεσης τύπων* (type inference) παρέχει συναρτήσεις που μπορούν να εξάγουν τον τύπο μιας λειτουργίας (με βάση της παραμέτρους ή της εξαρτήσεις της) καθώς και να ελέγξουν τη συμβατότητα μεταξύ τύπων. Με βάση τον εξαγόμενο τύπο, αναθέτουν και την κατάλληλη συνάρτηση αποτίμησης σε κάθε λειτουργία (εκτός από τους αλγεβρικούς τελεστές).

◇ Η ενότητα *βελτιστοποίησης* (optimization) που αναδιατάσει τις λειτουργίες που περιέχονται σε ένα αλγεβρικό τελεστή, με σκοπό τη βελτιστοποίηση της αποτίμησης της ερώτησης. Αναθέτει και αυτή μια συνάρτηση αποτίμησης σε κάθε λειτουργία που βελτιστοποιεί.

◇ Η ενότητα συναρτήσεων *αποτίμησης* (evaluation) αποτελείται από μια σειρά από πολύ συγκεκριμένες συναρτήσεις αποτίμησης για κάθε λειτουργία που μπορεί να εμφανιστεί στον γράφο (η ίδια λειτουργία με διαφορετικούς τύπους ορισμάτων έχει διαφορετικές συναρτήσεις αποτίμησης).

◇ Η ενότητα λειτουργιών συστήματος βάσης δεδομένων. Δεδομένου ότι καμία βάση δεν έχει σχεδιαστεί για την αποθήκευση μεταδεδομένων RDF και ότι ο διερμηνέας σχεδιάστηκε για να μπορεί να χρησιμοποιηθεί πάνω από διαφορετικά τέτοια συστήματα, ορίσαμε ως χωριστή ενότητα ένα σύνολο από λειτουργίες που χρειάζεται να υποστηρίζει η βάση ώστε να μπορεί να χρησιμοποιηθεί ο διερμηνέας της RQL. Αυτό είναι και το μόνο τμήμα που πρέπει να επαναπρογραμματιστεί, προκειμένου να χρησιμοποιηθεί ο διερμηνέας της RQL πάνω από ένα διαφορετικό σύστημα διαχείρισης βάσεων δεδομένων (όχι απαραίτητα σχεσιακό).

Στη συνέχεια του κεφαλαίου θα παρουσιάσουμε αυτή τη σημασιολογική αναπαράσταση μέσα από μερικά χαρακτηριστικά παραδείγματα. Στη συνέχεια θα παρουσιάσουμε την δομή του διερμηνέα, θα αναλύσουμε τη διαδικασία αποτίμησης ερωτήσεων και θα κλείσουμε με την παρουσίαση του περιβάλλοντος προγραμματισμού εφαρμογών επερώτησης (Query API), που παρέχει τις βασικές λειτουργίες που απαιτούνται από ένα ΣΔΒΔ, ώστε να μπορέσει να χρησιμοποιηθεί ο διερμηνέας της RQL.

4.3 Σημασιολογική αναπαράσταση ερωτήσεων

Στην ενότητα αυτή θα δείξουμε μέσα από κάποια χαρακτηριστικά παραδείγματα τη σημασιολογική αναπαράσταση των ερωτήσεων της RQL και θα παρουσιάσουμε κάποιες βασικές έννοιες, που είναι απαραίτητες για την κατανόηση των δομών και αλγορίθμων που χρησιμοποιεί ο διερμηνέας.

4.3.1 Απλές ερωτήσεις

Έστω, π.χ., η ερώτηση (με βάση το πολιτιστικό παράδειγμα του σχήματος 3.1).
Ερώτηση 1: *Βρες τους πόρους που έχουν δημιουργήσει κάτι χρησιμοποιώντας κάποια τεχνική, τα δημιουργήματά τους και την τεχνική με την οποία δημιουργήθηκαν.*

```
select  X, Y, Z
from    {X}creates{Y}.using_technique{Z}
where   Z = "oil on canvas"
```

Η ερώτηση αυτή παριστάνεται από το κατευθυνόμενο, ακυκλικό γράφο του σχήματος 4.2.

Στο γράφο του σχήματος 4.2, οι κόμβοι αναπαριστούν λειτουργίες και οι ακμές εξαρτήσεις μεταξύ των λειτουργιών αυτών. Μια ακμή $A \rightarrow B$ δηλώνει ότι η λειτουργία A εξαρτάται από τη B, ή με άλλα λόγια ότι η λειτουργία A πρέπει να αποτιμηθεί μετά από τη B. Τέλος, το έντονο βέλος δείχνει την κεντρική (root) λειτουργία (αυτή που περιλαμβάνει όλες τις υπόλοιπες και επιστρέφει το τελικό αποτέλεσμα της ερώτησης.

Ο γράφος της ερώτησης περιλαμβάνει τρία είδη κόμβων, που εμφανίζονται με διαφορετικά σύμβολα:

- ◇ Οι εξάγωνοι κόμβοι υποδηλώνουν αλγεβρικούς τελεστές της μορφής Join-Select-Project (οι οποίοι ουσιαστικά ανταποκρίνονται στο βασικό φίλτρο της RQL). Στο παράδειγμά μας, περιλαμβάνει τρεις αναφορές: δύο σε λειτουργίες σάρωσης (join), μία στην boolean λειτουργία που περιέχει όλα τα κριτήρια επιλογής της ερώτησης (select) και τρεις στις λειτουργίες που έχουμε επιλέξει να προβληθούν στο τελικό αποτέλεσμα (projection).

◇ Οι τριγωνικοί κόμβοι αναπαριστούν λειτουργίες σάρωσης συλλογών. Η μοναδική ακμή που ξεκινάει από τέτοιους κόμβους υποδεικνύει τη συλλογή που πρέπει να σαρωθεί. Στην παραπάνω ερώτηση οι λειτουργίες αυτές ίσως δεν είναι προφανείς, λόγω της μη ύπαρξης μεταβλητών πάνω στις δύο αυτές συλλογές. Όπως όμως εξηγήσαμε στην ενότητα 3.3, η ερμηνεία μιας απλής έκφρασης μονοπατιού εισάγει μια μεταβλητή που παίρνει τιμές από την ερμηνεία της ιδιότητας. Στον γράφο του σχήματος 4.2 χρησιμοποιούμε τα ονόματα $p1$, $p2$ για τις αντίστοιχες λειτουργίες σάρωσης.

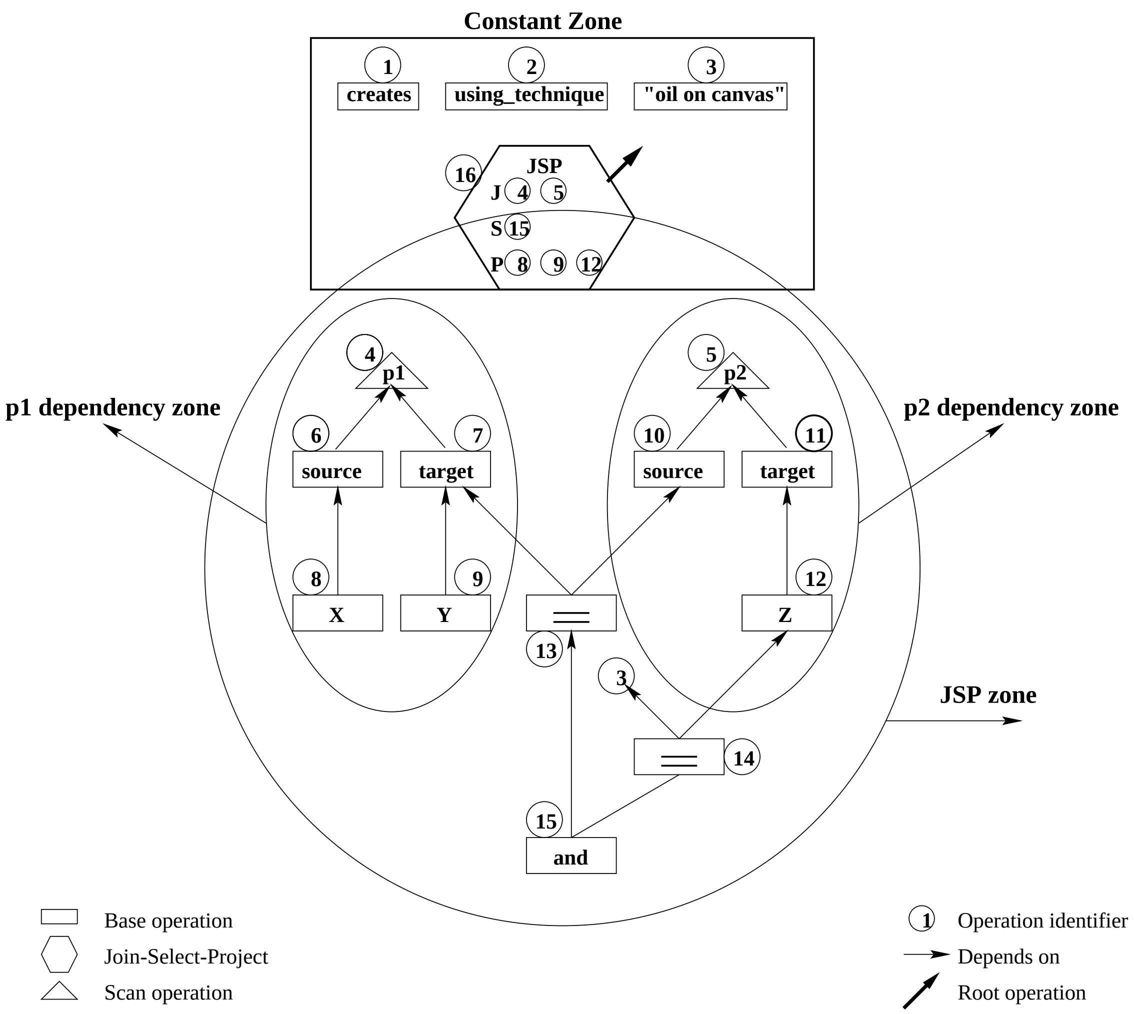
◇ Οι παραλληλόγραμμοι κόμβοι αναφέρονται σε οποιαδήποτε από τις υπόλοιπες λειτουργίες, που ονομάζονται και βασικές λειτουργίες. Οι κόμβοι αυτοί περιέχουν το είδος της λειτουργίας (ή το όνομα της μεταβλητής για λειτουργίες ανάθεσης ονόματος, όπως ο κόμβος 8), ενώ οι ακμές υποδεικνύουν τις παραμέτρους της κάθε λειτουργίας.

Ας δούμε τώρα τη σημασία των διαφορετικών ζωνών στις οποίες χωρίζεται ο γράφος του σχήματος:

◇ Το παραλληλόγραμμο στο πάνω μέρος του σχήματος περικλείει το σύνολο από τις *σταθερές λειτουργίες*. Οι λειτουργίες αυτές απαιτούν μια και μοναδική αποτίμηση - η οποία μάλιστα γίνεται κατά τη φάση κατασκευής του γράφου - καθώς δεν εξαρτώνται από καμία λειτουργία σάρωσης.

◇ Ο μεγάλος κύκλος - που περικλείει όλους τους υπόλοιπους - περιλαμβάνει όλες τις λειτουργίες που εξαρτώνται από τις λειτουργίες σάρωσης που περιέχονται στον αλγεβρικό τελεστή JSP. Μια ερώτηση μπορεί να περιλαμβάνει και εμφωλευμένους αλγεβρικούς τελεστές, οπότε ο γράφος γίνεται πιο περίπλοκος, όπως θα δούμε στη συνέχεια.

Αξίζει να σημειωθεί ότι υπάρχουν λειτουργίες (όπως οι 1, 2, 3) που δεν περικλείονται στον κύκλο αυτό, ενώ βρίσκονται στο ίδιο φίλτρο SFW της ερώτησης. Οι λειτουργίες αυτές ονομάζονται περικλειόμενες (included) σε αντίθεση με αυτές που περικλείονται από τον κύκλο, οι οποίες ονομάζονται εμφωλευμένες (nested).



Σχήμα 4.2 Η σημασιολογική αναπαράσταση της ερώτησης 1

Οι περικλειόμενες λειτουργίες είναι απαραίτητες για την αποτίμηση ενός αλγεβρικού τελεστή JSP αλλά μπορούν να αποτιμηθούν ανεξάρτητα, σε αντίθεση με τις εμφωλευμένες.

◇ Οι δύο ελλείψεις που περικλείονται από τον παραπάνω κύκλο περιέχουν τις λειτουργίες που εξαρτώνται μόνο από μία λειτουργία σάρωσης, την $p1$ ή $p2$ αντίστοιχα.

◇ Η τελευταία ζώνη (το τμήμα του εξωτερικού κύκλου που δεν περικλείεται από καμία από τις δύο ελλείψεις) περιλαμβάνει τις λειτουργίες που εξαρτώνται από περισσότερες από μία λειτουργίες σάρωσης. Ο διαχωρισμός των λειτουργιών αυτών είναι σημαντικός όταν ο αλγεβρικός τελεστής JSP περιλαμβάνει περισσότερες μεταβλητές σάρωσης, μεταξύ των οποίων υπάρχουν και εξαρτήσεις, ή η ερώτηση περιλαμβάνει εμφωλευμένους αλγεβρικούς τελεστές JSP - οπότε ο μεγάλος κύκλος μπορεί να περικλείεται από άλλους κύκλους ή ελλείψεις. Για λόγους απλότητας της παρουσίασης δε θα επεκταθούμε περισσότερο σε αυτή την περίπτωση.

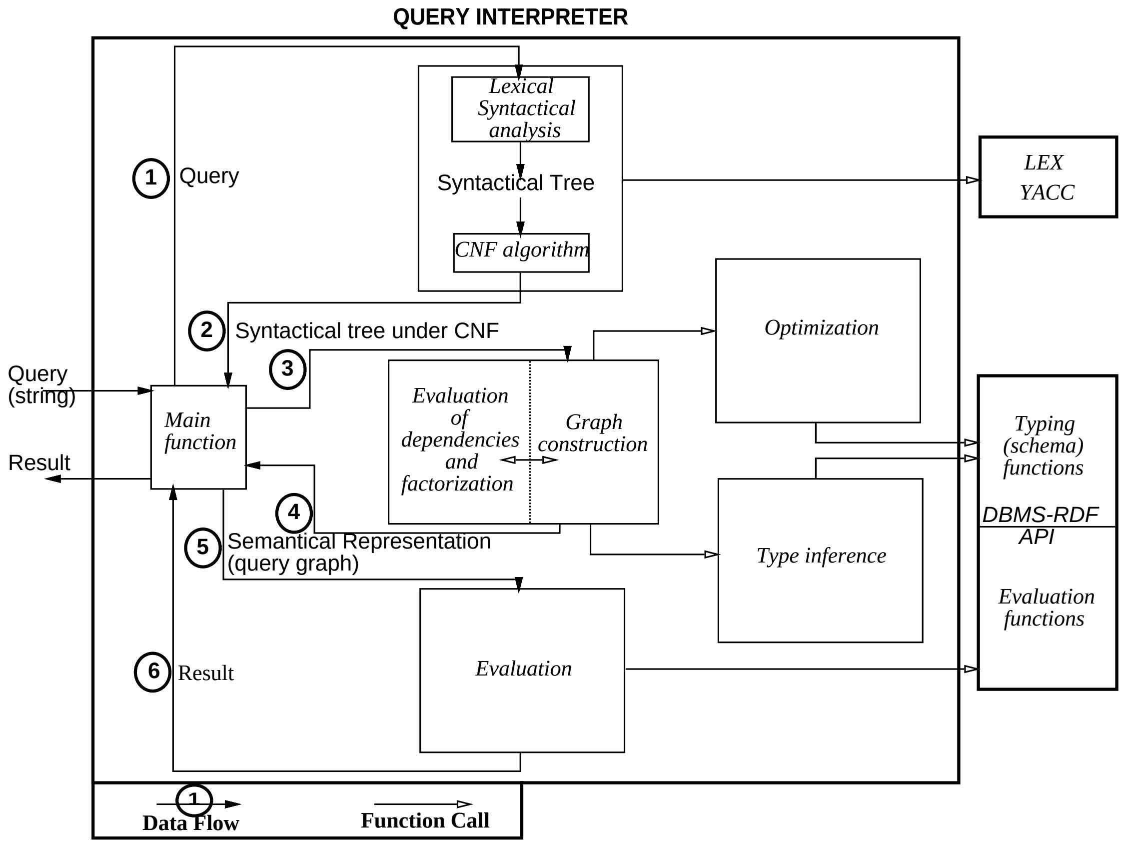
4.4 Αρχιτεκτονική του διερμηνέα ερωτήσεων

Η αρχιτεκτονική του διερμηνέα φαίνεται στο σχήμα 4.3. Μια επερώτηση εισάγεται στο διερμηνέα ως μια συμβολοσειρά. Εφόσον έχει εισαχθεί στο διερμηνέα, την χειρίζεται η κύρια ενότητα, που εκτελεί τις παρακάτω λειτουργίες από τις άλλες ενότητες:

◇ Η επερώτηση (ως συμβολοσειρά) στέλνεται στον συντακτικό αναλυτή (βέλος ροής δεδομένων (1)) που δημιουργεί ένα συντακτικό δένδρο και μετατρέπει την επερώτηση σε συζευκτική κανονική μορφή. Ο συντακτικός αναλυτής χρησιμοποιεί τα εργαλεία Lex και Yacc, για τη λεξικογραφική και τη συντακτική ανάλυση. Διαχειρίζεται το δικό του χώρο μνήμης, έτσι ώστε να εγγυάται οτι κάθε τμήμα του δέντρου που έχει κατασκευαστεί απελευθερώνεται αν υπάρξει συντακτικό λάθος (αλλιώς είναι αδύνατο να απελευθερώνεται ένα εν μέρει κατασκευασμένο δένδρο με ασύνδετους κλάδους). Αν δεν ανιχνευθεί συντακτικό λάθος, η ενότητα του συντακτικού αναλυτή επιστρέφει το συντακτικό δένδρο σε συζευκτική κανονική μορφή (βέλος ροής δεδομένων (2)), αλλιώς επιστρέφει έναν κώδικα λάθους και το σύμβολο στο οποίο το λάθος ανιχνέυθηκε.

◇ Το συντακτικό δένδρο δίνεται ως είσοδος στην ενότητα κατασκευής γράφων (βέλος ροής δεδομένων (3)). Αυτή η ενότητα ανακαλύπτει τις εξαρτήσεις μεταξύ των λειτουργιών της ερώτησης και παραγοντοποιεί τις κοινές εκφράσεις. Επίσης καλεί:

  ▷ συναρτήσεις της ενότητας ελέγχου τύπων για να αναθέσει τύπο και τις κατάλληλες συναρτήσεις αποτίμησης στους κόμβους του γράφου. Η ενότητα αυτή βασίζεται σε συναρτήσεις ελέγχου τύπων που παρέχει το API της βάσης, και επιτρέπουν - μεταξύ άλλων - ελέγχους για το αν ένα όνομα αντιστοιχεί είναι όνομα κλάσης ή όνομα ιδιότητας. Επίσης, ελέγχει την "κα-



Σχήμα 4.3 Η αρχιτεκτονική του διερμηνέα ερωτήσεων της RQL

ταλληλότητα" του τύπου των ορισμάτων (π.χ. ένας τελεστής σάρωσης μπορεί να εφαρμοστεί μόνο σε μια συλλογή). Αν οι έλεγχοι αυτοί επιτύχουν αποδίδεται στον κόμβο ο τύπος του καθώς και η συνάρτηση αποτίμησής του (αν δεν πρόκειται για iterator). Σε αντίθετη περίπτωση, επιστρέφεται κατάλληλος κωδικός λάθους.

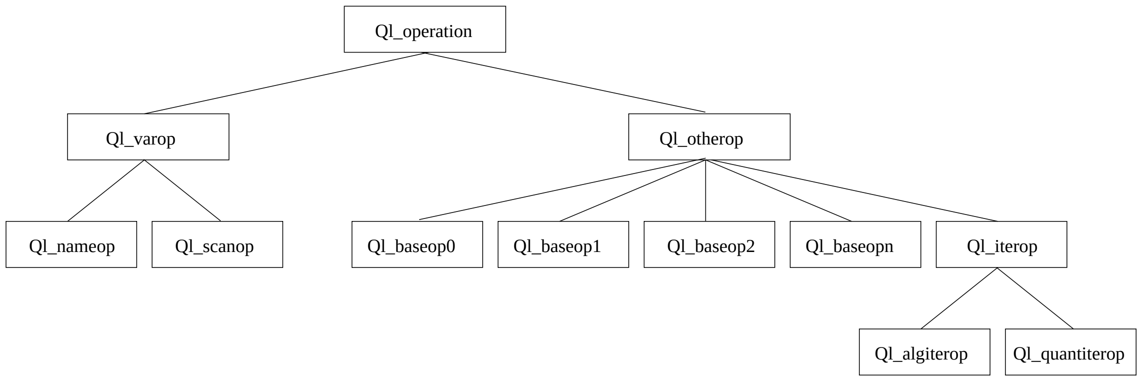
Αν δεν ανιχνευθεί λάθος σε τύπο, ο κατασκευαστής γράφων επιστρέφει την σημασιολογική αναπαράσταση της επερώτησης (βέλος ρόης δεδομένων (4)), αλλιώς απελευθερώνει το χώρο που είχε καταληφθεί κι επιστρέφει κώδικα λάθους.

◇ Αφού ο γράφος έχει κατασκευαστεί, η κύρια ενότητα αποστέλλει τη σημασιολογική αναπαράσταση στην ενότητα αποτίμησης (βέλος ροής δεδομένων (5)) η οποία, με τη σειρά της, καλεί τη συνάρτηση αποτίμησης της κεντρικής λειτουργίας. Η κλήση αυτή προκαλεί την εκτέλεση μιας σειράς από επικαλυπτόμενες συναρτήσεις που τελειώνει όταν μια επερώτηση αποτιμάται ή όταν υπάρχει λάθος κατά τη διάρκεια της εκτέλεσης. Έπειτα, η ενότητα αποτίμησης επιστρέφει το αποτέλεσμα της επερώτησης ή τον κώδικα λάθους στην κυρίως ενότητα (βέλος ροής δεδομένων (6)).

◇ Η κυρίως ενότητα απελευθερώνει τον χώρο που παραμένει κατειλλημένος κι επιστρέφει το αποτέλεσμα ή έναν κώδικα λάθους.

4.5 Οι κόμβοι της σημασιολογικής αναπαράστασης

Όπως προαναφέραμε, οι κόμβοι του γράφου μιας ερώτησης αναπαρστούν λειτουργίες. Οι διαφορετικές αυτές λειτουργίες μοντελοποιούνται στην υλοποίηση της RQL από την ιεραρχία κλάσεων που φαίνεται στο σχήμα 4.4.



Σχήμα 4.4 Η ιεραρχία λειτουργιών στον διερμηνέα της RQL

Στη συνέχεια αυτής της ενότητας θα δείξουμε πώς η σημασιολογική μας αναπαράσταση υλοποιείται με τη χρήση αυτών των κλάσεων.

4.5.1 Ql_operation

Η κλάση αυτή - η ρίζα της ιεραρχίας - αναπαριστά κάθε κόμβο του γράφου. Περιέχει πληροφορίες που είναι κοινές για όλες τις λειτουργίες, όπως τον τύπο της, τις ζώνες εξάρτησης και τους iterators στους οποίους περιέχεται (οι σταθερές λειτουργίες δεν περιέχονται σε καμία ζώνη εξάρτησης). Επίσης, διατηρεί την τιμή της - όπως προκύπτει σε κάθε φάση της αποτίμησης. Τέλος, περιέχει πληροφορία για το ποιές άλλες λειτουργίες εξαρτώνται από αυτή, δηλ. τις λεγόμενες *αντίστροφες ακμές* (inverse edges), που είναι χρήσιμες για την παραγοντοποίηση κοινών εκφράσεων: Όταν πρόκειται να δημιουργήσουμε μια λειτουργία $f(q_1, q_2, \ldots, q_n)$, μπορούμε να ελέγξουμε αν η λειτουργία αυτή υπάρχει ήδη με τις ίδιες παραμέτρους, ελέγχοντας τις αντίστροφες ακμές των λειτουργιών που έχουν ήδη δημιουργηθεί.

4.5.2 Ql_varop

Η κλάση αυτή αναπαριστά λειτουργίες σε μεταβλητές, όπως την ανάκτηση ή την ενημέρωση της τιμής τους. Έχει δύο σημαντικές υποκλάσεις, τις **Ql_scanop** και **Ql_nameop**.

4.5.3 Ql_scanop

Η κλάση αυτή αναπαριστά λειτουργίες σάρωσης (που αναπαριστούνται με τριγωνικούς κόμβους στη σημασιολογική αναπαράσταση). Οι λειτουργίες αυτές διατηρούν επιπλέον πληροφορία για τη συλλογή σάρωσης καθώς και πιθανή πληροφορία που προστίθεται στη φάση βελτιστοποίησης, όπως για παράδειγμα μία αναφορά σε κάποιο σύστημα δεικτοδότησης (index) που πρέπει να χρησιμοποιηθεί για τη σάρωση της συλλογής.

4.5.4 Ql_nameop

Η κλάση αυτή αναπαριστά λειτουργίες ανάθεσης ονόματος σε άλλες λειτουργίες. Τέτοια περίπτωση είναι, π.χ., η χρήση των μεταβλητών X και Y στο απλό παράδειγμα της προηγούμενης ενότητας, που απλά "ονοματίζουν" τις λειτουργίες λήψης των τιμών στα άκρα της ιδιότητας `creates`, που σαρώνεται από τη λειτουργία σάρωσης με όνομα $p1$. Ουσιαστικά αναπαριστά όλες τις περιπτώσεις μεταβλητών που δεν αντιστοιχούν σε λειτουργίες σάρωσης.

4.5.5 Ql_otherops

Η κλάση αυτή αναπαριστά όλες τις άλλες λειτουργίες που μπορεί να εμφανιστούν στον γράφο μιας ερώτησης. Οι λειτουργίες αυτές ταξινομούνται περαιτέρω στις κλάσεις που θα δούμε στη συνέχεια. Υπάρχουν όμως κάποια κοινά στοιχεία - που διατηρούνται στην κλάση **Ql_otherops**, όπως η ζώνη(-ες) εξάρτησης στην οποία πιθανώς ανήκει (δηλ. η λειτουργία σάρωσης από την οποία εξαρτάται). Η πληροφορία αυτή χρησιμοποιείται για να συμπεράνουμε σε κάθε φάση της αποτίμησης αν μια τέτοια λειτουργία πρέπει να αποτιμηθεί ξανά (κάτι που δεν είναι απαραίτητο αν η λειτουργία σάρωσης δεν έχει αλλάξει τιμή).

4.5.6 Ql_baseop0, Ql_baseop1, Ql_baseop2, Ql_baseopn

Αυτές οι υποκλάσεις αναπαριστούν αντίστοιχα λειτουργίες με καμία, μία, δύο ή n παραμέτρους. Ειδικά για την πρώτη, πρόκειται για απλές λειτουργίες - όπως αναγνωριστικά αντικειμένων, συμβολοσειρές ή ακέραιους αριθμούς - οι οποίες αποτιμούνται μόνο μια φορά στην αρχή της αποτίμησης της ερώτησης. Οι υπόλοιπες διατηρούν πληροφορία για τις παραμέτρους, το όνομα και τη συνάρτηση αποτίμησής τους και αποτιμούνται ξανά κάθε φορά που αυτές αλλάζουν τιμή.

4.5.7 Ql_iterop

Η κλάση αυτή χρησιμοποιείται για την αναπαράσταση iterators. Περικλέιει πληροφορία αναφορές στις λειτουργίες που περιέχει καθώς και στον iterator στον οποίο (πιθανώς) περικλείεται. Οι λειτουργίες τις οποίες περιέχει χωρίζονται σε iterators (εμφωλευμένους κύκλους), ονοματισμένες μεταβλητές (τύπου Ql_nameop) και βασικές λειτουργίες (εμφωλευμένα παραλληλόγραμμα). Η σημασιολογία του iterator δίνεται από τη συνάρτηση αποτίμησής του.

4.5.8 Ql_algiterop

Η κλάση αυτή αναπαριστά αλγεβρικούς iterators, όπως ο SFW. Περιλαμβάνουν επιπλέον πληροφορία για τις λειτουργίες σάρωσής τους (τριγωνικοί κόμβοι), τις λειτουργίες προβολής και επιλογής. Οι τελευταίες περιλαμβάνουν τα κριτήρια επιλογής που εκφράζονται ρητά στο τμήμα του `where` της ερώτησης, που προστίθενται στο τέλος της κατασκευής του γράφου. Επίσης, περιλαμβάνουν και τα κριτήρια επιλογής που σχετίζονται με τα μονοπάτια και τη συνένωσή τους (βλ. ενότητα 3.3) τα οποία προστίθενται κατά τη διάρκεια της κατασκευής του γράφου της ερώτησης.

4.5.9 Ql_quantiterop

Η κλάση αυτή αναπαριστά ποσοδείκτες (υπαρξιακούς - καθολικούς) και περιλαμβάνει πληροφορία για τη λειτουργία σάρωσης και τη boolean λειτουργία που περιέχουν.

4.6 Αποτίμηση των ερωτήσεων της RQL

Στην ενότητα αυτή θα εξηγήσουμε τη διαδικασία αποτίμησης μιας ερώτησης, με δεδομένη τη σημασιολογική της αναπαράσταση, από το διερμηνέα ερωτήσεων της RQL. Η αποτίμηση αυτή ουσιαστικά ξεκινά από την αποτίμηση της κεντρικής λειτουργίας του γράφου της ερώτησης. Στη συνέχεια, η αποτίμηση κάθε λειτουργίας γίνεται από την αντίστοιχη συνάρτηση αποτίμησης, που της ανατίθεται στη φάση του ελέγχου τύπων. Πιο συγκεχριμένα:

◇ **Ql_otherops:** Όπως είδαμε προηγουμένως, η κλάση αυτή - και όλες οι υποκλάσεις της - αναπαριστούν τις λειτουργίες που δεν είναι μεταβλητές. Οι λειτουργίες αυτές μπορεί να εξαρτώνται από μία ή περισσότερες μεταβλητές. Σε κάθε φάση της αποτίμησης ελέγχουμε - μέσω κατάλληλων συναρτήσεων και με βάση τις εξαρτήσεις αυτές - αν μια τέτοια λειτουργία πρέπει να αποτιμηθεί ξανά (αν δηλαδή κάποια από τις μεταβλητές από τις οποίες εξαρτάται έχει αλλάξει τιμή).

◇ **Ql_scanop:** Η λειτουργία αυτή δεν έχει - ουσιαστικά - συνάρτηση αποτίμησης ανάλογη με τις υπόλοιπες. Η κλήση των απαραίτητων συναρτήσεων για τη σάρωση συλλογών διαχειρίζεται από τις συναρτήσεις αποτίμησης των iterators.

◇ **Ql_nameop:** Οι λειτουργίες αυτές απλά επιστρέφουν την τιμή της λειτουργίας ορισμού τους κατά την αποτίμηση.

◇ **Ql_baseop0:** Η κλάση αυτή - όπως προαναφέραμε - αναπαριστά σταθερές λειτουργίες, η αποτίμηση των οποίων γίνεται μόνο μία φορά, κατά τον έλεγχο τύπων.

◇ **Υπόλοιπες λειτουργίες:** Κάθε άλλη λειτουργία συνοδεύεται από μια πολύ συγκεκριμένη συνάρτηση αποτίμησης. Όταν πρέπει να αποτιμηθεί ξανά, καλεί τη συνάρτηση αυτή στις παραμέτρους της, για να πάρει το νέο αποτέλεσμα.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η αποτίμηση των αλγεβρικών iterators, καθώς αυτοί - όπως προαναφέραμε - "συντονίζουν" τη διαδικασία αποτίμησης καλώντας τις κατάλληλες συναρτήσεις των λειτουργιών σάρωσης. Συγκεκριμένα θα αναφερθούμε στην αποτίμηση ενός φίλτρου sf/sfw.

Ένα τέτοιο φίλτρο περιέχει ένα κριτήριο ανά μεταβλητή σάρωσης συν ένα. Το επιπλέον αυτό κριτήριο περιέχει τις συνθήκες που αναφέρονται σε περισσότερες από μία μεταβλητές. Αποτιμάται μετά από κάθε αποτίμηση οποιασδήποτε μεταβλητής. Στη συνέχεια θα αναφερόμαστε σε αυτό ως "*κριτήριο 0*". Τα υπόλοιπα κριτήρια αντιστοιχούν σε συνθήκες που αναφέρονται σε μία μόνο μεταβλητή. Έτσι, το κριτήριο i αποτιμάται ξανά μετά από κάθε νέα αποτίμηση της μεταβλητής i (δηλαδή κάθε φορά που λαμβάνεται το επόμενο στοιχείο από την αντίστοιχη λειτουργία σάρωσης). Τα κριτήρια αυτά παίρνουν συνεχώς νέες τιμές μέχρι να ικανοποιηθεί η συνθήκη (pushed selections). Συνολικά, ο αλγόριθμος διαχείρισης των λειτουργιών σάρωσης - που βασίζεται σε φωλιασμένες ανακυκλώσεις για την αποτίμηση των συνδέσεων (joins) - έχει ως εξής:

1. Δημιουργία πίνακα για αποθήκευση του αποτελέσματος
2. "Άνοιγμα" των λειτουργιών σάρωσης
3. Όσο δεν υπάρχει λάθος στην αποτίμηση κάποιας λειτουργίας
   (a) Αν το κριτήριο 0 περιέχει κάποια συνθήκη και η αποτίμησή του επιστρέφει κωδικό λάθους: επιστροφή κωδικού λάθους
   (b) αλλιώς
      i. Αποτίμηση των λειτουργιών που περιέχονται στην προβολή της ερώτησης
      ii. Αν υπήρξε λάθος: επέστρεψε λάθος
      iii. Ενημέρωση αποτελέσματος με νέα πλειάδα
   (c) Επόμενη λειτουργία σάρωσης (θα αναλυθεί στη συνέχεια)
4. "Κλείσιμο" των λειτουργιών σάρωσης
5. Επιστροφή αποτελέσματος ή κωδικού λάθους

Οι λειτουργίες σάρωσης οργανώνονται σε μία εμφωλευμένη δομή, ανάλογα με τις εξαρτήσεις τους. Έτσι, αν μια μεταβλητή X εξαρτάται από την Y, ο βρόχος (loop) με τον οποίο παίρνουν όλα τα στοιχεία της συλλογής που αναφέρεται η X θα είναι εμφωλευμένος μέσα στον αντίστοιχο βρόχο της Y. Συνεπώς, για την επιλογή της επόμενης λειτουργίας σάρωσης ακολουθείται ο εξής αλγόριθμος:

1. Όσο η λειτουργία σάρωσης Y δεν έχει φτάσει στο τέλος της συλλογής της
   (a) Όσο η λειτουργία σάρωσης X δεν έχει φτάσει στο τέλος της συλλογής της
      i. πάρε το επόμενο στοιχείο της συλλογής που σαρώνει η X.
   (b) Πάρε το επόμενο στοιχείο της συλλογής που σαρώνει η Y.
2. επέστρεψε κωδικό τέλους

Για περισσότερες από δύο λειτουργίες ακολουθείται η ίδια λογική εμφωλευμένων ανακυκλώσεων (για λόγους οικονομίας χώρου και εξαιτίας της πολυπλοκότητάς του, δεν παραθέτουμε τέτοιο παράδειγμα, όμως η διαδικασία πρέπει να είναι προφανής)

4.6.1 Μετάφραση ερωτήσεων σε SQL3

Στην προηγούμενη ενότητα εξηγήσαμε τη λογική αποτίμησης εκφράσεων που περιλαμβάνουν εξαρτήσεις, συνθέτοντας συναρτήσεις αποτίμησης απλών εκφράσεων. Οι εκφράσεις αυτές μπορούν να αποτιμηθούν από αντίστοιχες ερωτήσεις SQL3. Μάλιστα, σε κάποιες περιπώσεις είναι προτιμότερο να συντεθούν τέτοιες απλές ερωτήσεις σε εμφωλευμένες ερωτήσεις SQL3, ώστε να εκμεταλλευτούμε και δυνατότητες βελτιστοποίησης του ΣΔΒΔ.

Ανεξάρτητα με το συγκεκριμένο αλγόριθμο αποτίμησης, μπορούμε σε κάθε περίπτωση να σχεδιάσουμε το πλάνο αποτίμησης μιας ερώτησης RQL, το οποίο μπορεί να χρησιμοποιηθεί είτε για αναδιάταξη ερωτήσεων (query rewriting) είτε για την αναγνώριση απλών ερωτήσεων SQL3 και τη μετάφραση της ερώτησης σε SQL3.

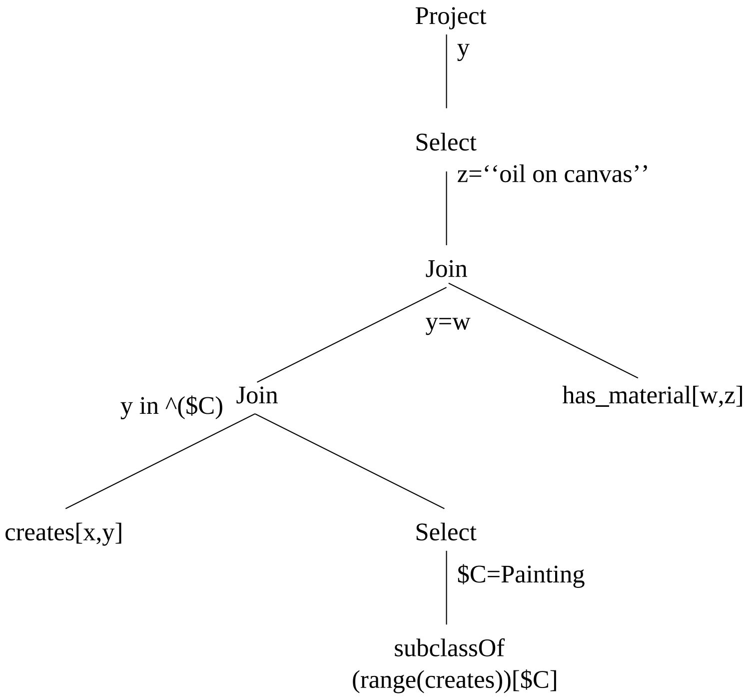
Για παράδειγμα, στο σχήμα 4.5 βλέπουμε το πλάνο αποτίμησης της παρακάτω ερώτησης.

Ερώτηση 2

```
select  y
```

`from` {x}creates{y:Painting}.has_material{z}

`where` z=“oil on canvas”



Σχήμα 4.5 Το πλάνο αποτίμησης της ερώτησης 2

Στο δέντρο του σχήματος, οι κόμβοι αναπαριστούν βασικούς τελεστές της γλώσσας ενώ τα φύλλα είναι ουσιαστικά οι στοιχειώδεις εκφράσεις από τις οποίες αποτελείται η ερώτησή μας. Ο συμβολισμός “[]” υποδηλώνει τον ορισμό μιας λειτουργίας σάρωσης πάνω στην αντίστοιχη συλλογή. Για παράδειγμα, το πιο αριστερό φύλλο υποδηλώνει μια λειτουργία σάρωσης (μεταβλητές ,) πάνω στην ερμηνεία της ιδιότητας creates.

Έτσι, στο σχήμα αυτό βλέπουμε ότι η συνένωση των μονοπατιών μεταφράζεται σε μία σύνδεση (join) μεταξύ των ερωτήσεων που αναπαριστούνται από τα αντίστοιχα υποδέντρα. Η αποτίμηση του δεξιού από αυτά - ένα μονοπάτι δεδομένων - είναι τετριμμένη. Για το αριστερό τμήμα - μικτό μονοπάτι - το πλάνο περιέχει άλλη μια σύνδεση, μεταξύ της ερμηνείας της ιδιότητας `creates` και της πληροφορίας σχήματος που προκύπτει ως αποτέλεσμα των συναρτήσεων που εφαρμόζονται στο πεδίο τιμών της ιδιότητας. Η συνθήκες της σύνδεσης αυτής - όπως εξηγήσαμε για την ερμηνεία των μικτών μονοπατιών στην ενότητα 3.3 - συσχετίζουν τις μεταβλητές δεδομένων με τις αντίστοιχες μεταβλητές σχήματος ($Yin\hat{}(\$C)$).

Με βάση τον αλγόριθμο των εμφωλευμένων ανακυκλώσεων που αναλύθηκε προηγουμένως, η αποτίμηση της ερώτησης αυτής γίνεται ως εξής:

◇ Οι κόμβοι-φύλλα αποτιμούνται από ερωτήσεις SQL3. Στο συγκεκριμένο παράδειγμα, οι αντίστοιχες ερωτήσεις είναι:

▷ creates[x,y]:
select source, target from creates*
(Το σύμβολο "*" χρησιμοποιείται για να συμπεριλάβει και τα περιεχόμενα των υποπινάκων, π.χ. paints και sculpts στο πολιτιστικό μας παράδειγμα).

▷ subclassOf(range(creates)):
select C from `subclassOf`(`range`(creates)) C where C = 'Painting'
Η συναρτήσεις `subclassOf`, `range` έχουν προγραμματιστεί ως προσθήκη στην SQL3, για τον υπολογισμό της μεταβατικής κλειστότητας της σχέσης `subClass` και της κλάσης ορισμού μιας ιδιότητας.

▷ has_material[w,z]:
select source, target from has_material[w,z]

◇ Στη συνέχεια, η σύνθεση αυτών των αποτελεσμάτων γίνεται εσωτερικά στο διερμηνέα της RQL, με χρήση εμφωλευμένων ανακυκλώσεων για την αποτίμηση των συνδέσεων. Έτσι, για παράδειγμα, το αριστερό υπόδεντρο αποτιμάται ως εξής:

Όσο η λειτουργία σάρωσης [$\$C$] δεν έχει φτάσει στο τέλος της συλλογής της
  πάρε το επόμενο στοιχείο της συλλογής που σαρώνει η [$\$C$]
  Αν $\$C2 = Painting$
    Όσο η λειτουργία σάρωσης [x, y] δεν έχει φτάσει στο τέλος της συλλογής της
      πάρε το επόμενο στοιχείο της συλλογής που σαρώνει η [x, y]
      Αν $y \in\hat{}(\$C)$
        Πρόσθεσε νέα πλειάδα στο αποτέλεσμα
Επέστρεψε κωδικό τέλους

Όπως βλέπουμε, αυτός ο τρόπος αποτίμησης περιλαμβάνει αρκετό όγκο υπολογισμών από την πλευρά του διερμηνέα, ειδικά για την αποτίμηση συνδέσεων. Προκειμένου να εκμεταλλευτούμε γνωστές τεχνικές βελτιστοποιήσεων τέτοιων υπολογισμών - που υποστηρίζονται από τα περισσότερα ΣΔΒΔ, θα ήταν προτιμότερο να συγκεντρώνουμε τέτοιους υπολογισμούς σε πιο περίπλοκες ερωτήσεις, η αποτίμηση των οποίων θα γίνεται από το ΣΔΒΔ αντί για το διερμηνέα της *RQL*. Έτσι, για παράδειγμα, η ερώτηση του σχήματος 4.5 θα μπορούσε να γραφεί ως ερώτηση SQL3 (που μπορεί να αποτιμηθεί εξ'ολοκλήρου από το ΣΔΒΔ) όπως φαίνεται στο σχήμα 4.6. Στην ερώτηση αυτή, το κατηγόρημα `instanceOf` έχει υλοποιηθεί επίσης ως συνάρτηση της SQL3, για να ελέγξει αν ένα αναγνωριστικό πόρου είναι άμεσο στιγμιότυπο μίας κλάσης (π.χ. $y \in\hat{}(\$C)$).

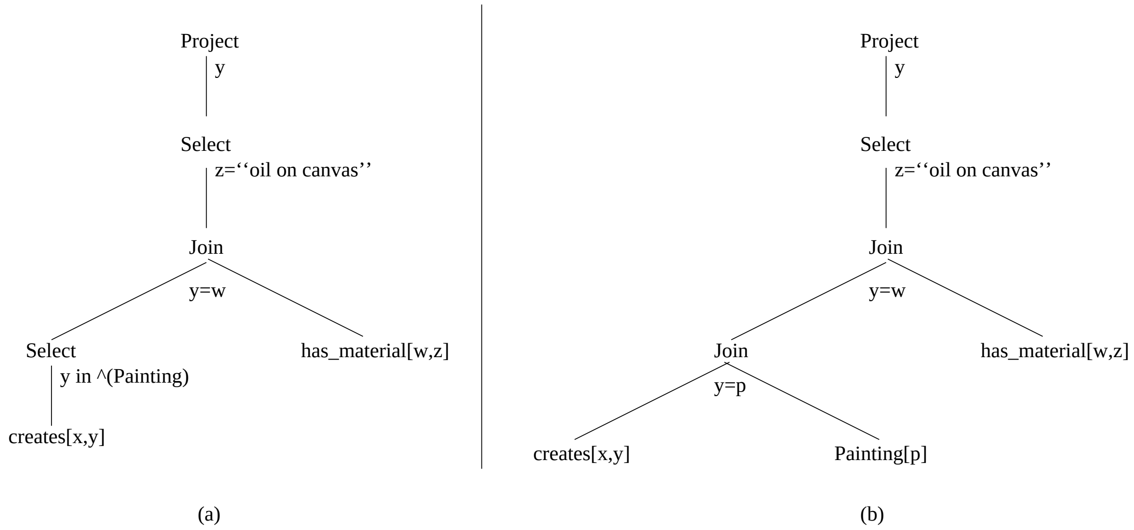
```
select X.target
from   subclassOf(range(creates)) C,
       creates* X, has_material* Y
where  C = 'Painting' and instanceOf(X.target, C)
       and X.target = Y.source and Y.target='oil on canvas'
```

Σχήμα 4.6 Μετάφραση ερώτησης RQL σε SQL3

Η παραπάνω μετάφραση είναι κάπως απλοϊκή. Για παράδειγμα, θα μπορούσαμε να αποφύγουμε την σάρωση του συνόλου `subclassOf(range(creates))`, δεδομένου ότι η ερώτηση απαιτεί την ισότητα με `Painting`. Έτσι προκύπτει το πλάνο αποτίμησης που φαίνεται στο σχήμα 4.7(a). Επίσης, αντί για το κατηγόρημα `instanceOf(X.target, Painting)`, που για κάθε ελέγχει αν κάθε τιμή προέλευσης του X είναι στιγμιότυπο της κλάσης αυτής, θα μπορούσαμε να κάνουμε μία σύνδεση μεταξύ του συνόλου των κόμβων προέλευσης του X και των άμεσων στιγμιοτύπων της κλάσης Painting, εκμεταλλευόμενοι έτσι και πιθανές δομές δεικτών - ευρετήρια - που μπορεί να έχουν κατασκευαστεί στη βάση. Το τελικό πλάνο αποτίμησης φαίνεται στο σχήμα 4.7(b).

Η αντίστοιχη ερώτηση SQL3 που προκύπτει από αυτό το πλάνο φαίνεται στο σχήμα 4.8.

Αξίζει να σημειωθεί ότι με τον τρόπο αυτό εκμεταλλευόμαστε γνωστές τεχνικές βελτιστοποίησης, όπως αναδιάταξη συνδέσεων (join reordering), προώθηση επιλογών σε χαμηλότερα "επίπεδα" (pushed selections) κλπ, που υποστηρίζει το ΣΔΒΔ, χωρίς μάλιστα να χρειάζονται σχετικοί μετασχηματισμοί στο πλάνο αποτίμησης του διερμη-



Σχήμα 4.7 Μετασχηματισμοί στο πλάνο αποτίμησης της ερώτησης 2

```
select X.target
from   creates* X, has_material* Y, Painting P
where  X.target = Y.source and X.target = P.uri and Y.target='oil on canvas'
```

Σχήμα 4.8 Μετάφραση βελτιστοποιημένης ερώτησης RQL σε SQL3

νέα της *RQL*: η παραγόμενη ερώτηση στέλνεται στο ΣΔΒΔ που κατασκευάζει το δικό του πλάνο αποτίμησης και εφαρμόζει σε αυτό τέτοιες βελτιστοποιήσεις.

4.7 Περιβάλλον Προγραμματισμού Εφαρμογών Επερώτησης ΣΔΒΔ

Όπως αναφέραμε και στην αρχή αυτού του κεφαλαίου, μία πολύ σημαντική ενότητα του διερμηνέα ερωτήσεων είναι το Περιβάλλον Προγραμματισμού Εφαρμογών Επερώτησης Συστήματος Διαχείρισης Βάσεων Δεδομένων (DBMS query API), το οποίο χωρίζεται σε συναρτήσεις ελέγχου τύπων και συναρτήσεις που είναι απαραίτητες για την αποτίμηση ερωτήσεων. Πράγματι, το οντοκεντρικό/σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων που χρησιμοποιήθηκε για την αποθήκευση σχημάτων και περιγραφών RDF "αγνοεί" τη σημασιολογία της αποθήκευσης (καθώς μπορεί να χειριστεί την αποθηκευμένη πληροφορία μόνο ως μια σχεσιακή δομή). Στη συνέχεια αυτής της ενότητας θα αναλύσουμε τις δύο αυτές κατηγορίες συναρτήσεων, τόσο ως προς τη χρησιμότητά τους για το διερμηνέα ερωτήσεων όσο και ως προς την υλοποίησή τους

- με τη χρήση ερωτήσεων SQL3 - πάνω από σύστημα διαχείρισης βάσεων δεδομένων.

4.7.1 Συναρτήσεις ελέγχου τύπων

4.7.1.1 isMetaclass, isClass, isProperty

Δεδομένου ότι οι μόνες μετακλάσεις που υπάρχουν είναι οι `Class`, `Property`, η συνάρτηση αυτή είναι τετριμμένη. Εξάλλου, οι συναρτήσεις `isClass`, `isProperty` ελέγχουν αν το όρισμά τους είναι το όνομα μιας κλάσης/ιδιότητας που έχει οριστεί σε κάποιο σχήμα RDF. Ουσιαστικά ισοδυναμούν με ένα έλεγχο για την ύπαρξη - ή μη - του ονόματος στη σχέση `Class/Property`. Οι συναρτήσεις αυτές μας παρέχουν όλη την επιθυμητή λειτουργικότητα για τον έλεγχο τύπων λειτουργιών σάρωσης (με μεταβλητές σχήματος για τις μετακλάσεις και δεδομένων για κλάσεις και ιδιότητες). Κατά τον έλεγχο τύπων, με βάση τη θέση ενός ονόματος σε μία ερώτηση, επιλέγουμε να καλέσουμε την κατάλληλη από τις παραπάνω συναρτήσεις. Έτσι, για παράδειγμα, στην ερώτηση:

```
select  X, Y
from    {X}creates{Y}
```

αρκεί να ελέγξουμε αν το όνομα `creates` αντιστοιχεί σε μία ιδιότητα. Αν όχι, τότε ο έλεγχος τύπων επιστρέφει λάθος. Ειδάλλως, θέτουμε τον τύπο της λειτουργίας αυτής σε "Property" και συνεχίζουμε τον έλεγχο τύπων της ερώτησης.

4.7.1.2 isBag, isSeq, isAlt

Ελέγχοντας την αντίστοιχη σχέση, η συνάρτηση αυτή απαντά αν το όρισμά της αντιστοιχεί στο όνομα της αντίστοιχης συλλογής RDF. Είναι απαραίτητες για τον έλεγχο τύπων σε λειτουργίες που επιτρέπονται μόνο σε συλλογές (πράξεις μεταξύ συλλογών, τελεστής `''[]''` σε λίστες κλπ.) καθώς και σε λειτουργίες σάρωσης (καθώς αυτές μπορούν να γίνουν μόνο σε συλλογές, ειδικές περιπτώσεις των οποίων είναι οι συλλογές στιγμιοτύπων κλάσεων και ιδιοτήτων).

4.7.1.3 isSubClassOf, isSubPropertyOf

Οι δύο αυτές συναρτήσεις ελέγχουν τη διάταξη δύο κλάσεων/ιδιοτήτων, ως προς τη σχέση υπαλληλίας. Η αποτίμησή τους ουσιαστικά συνίσταται στον υπολογισμό της πεπερασμένης κλειστότητας (transitive closure) της σχέσης subclass/subproperty, ξεκινώντας από μία από τις δύο κλάσεις/ιδιότητες. Ο υπολογισμός σταματάει αν βρεθεί η δεύτερη κλάση/ιδιότητα ή αφού τελειώσει ο υπολογισμός της πεπερασμένης κλειστότητας.

Για το υπολογισμό της πεπερασμένης κλειστότητας χρησιμοποιήθηκε ο αλγόριθμος $\delta - wavefront$ που περιγράφεται στο [QHK91]. Χονδρικά, ο αλγόριθμος αυτός ακολουθεί τα εξής βήματα:

1. $wave = (query\_constants)$;
2. $closure = \emptyset$;
3. $while(wave \neq \emptyset)$
4. $wave = \pi_{\gamma}\{wave(X) \bowtie_{1.1=2.1} A(X,Y)\}$;
5. $wave = wave - closure$;
6. $closure = wave \cup closure$;

Δεδομένου ότι - όπως φαίνεται και από την παραπάνω περιγραφή - ο υπολογισμός της πεπερασμένης κλειστότητας περιλαμβάνει πολλές ενημερώσεις πινάκων και πράξεις μεταξύ αυτών, επιλέξαμε να τον υλοποιήσουμε ως συνάρτηση της SQL3, χρησιμοποιώντας τη διεπαφή χαμηλού επιπέδου SPI της Postgres (ισοδύναμη με διαδικασιακή γλώσσα SQL - PL/SQL). Έτσι, τα βήματα του $\delta - wavefront$ υλοποιούνται από τις ερωτήσεις SQL:

1. while (not finished)
   (a) insert into tempwave (select id from wave)
   (b) delete from wave
   (c) insert into wave
   ```
   (select subid
    from tempwave, subclass
    where tempwave.id = subclass.superid and
      NOT EXISTS (select id from closure where result.id = subclass.subid))
   ```

   (d) if (select id from wave where id = classX) return true;

   (e) insert into closure select id from wave

2. return false;

Οι έλεγχοι αυτοί είναι απαραίτητοι για τον έλεγχο της συμβατότητας των τύπων των παραμέτρων δυαδικών (binary) λειτουργιών.

4.7.1.4 getDomainOf, getRangeOf

Οι συναρτήσεις αυτές δέχονται ως όρισμα ένα όνομα ιδιότητας και επιστρέφουν αντίστοιχα την κλάση ορισμού ή τιμών της. Είναι απαράιτητες για την ανάθεση τύπων σε λειτουργίες που αναφέρονται στις τιμές μιας ιδιότητας αλλά και για την αποτίμηση μεταβλητών σχήματος.

4.7.2 Συναρτήσεις αποτίμησης

Οι συναρτήσεις αυτές παρέχουν την απαραίτητη λειτουργικότητα για τη σάρωση συλλογών, καθώς και κάποιες επιπλέον λειτουργίες που απαιτούνται για την αποτίμηση.

4.7.2.1 scan_open, scan_close

Χρησιμοποιούνται για το άνοιγμα/κλείσιμο των συλλογών τις οποίες υποδηλώνει το όνομά τους. Επειδή σκοπός μας είναι να παίρνουμε ένα-ένα τα στοιχεία των συλλογών - προκειμένου να αποτιμήσουμε τις υπόλοιπες λειτουργίες της ερώτησης, όπως εξηγήθηκε σε προηγούμενη ενότητα - χρησιμοποιούμε τον μηχανισμό τον δεικτών (cursors) που παρέχει η SQL3. Εξάλλου, η συλλογή που πρέπει να σαρωθεί προκύπτει από το αποτέλεσμα μιας απλής ερώτησης SQL. Για παράδειγμα, το άνοιγμα μιας σάρωσης στα στιγμιότυπα της κλάσης `Painter` ισοδυναμεί με την ερώτηση:

```
declare  cursor1 cursor for
select * from Painter*
```

Στο παραπάνω παράδειγμα χρησιμοποιήσαμε το όνομα Painter* αντί για Painter. Με τον τρόπο αυτό αξιοποιούμε τη δυνατότητα που μας παρέχει η SQL3 να πάρουμε τα περιεχόμενα μίας σχέσης αλλά και όλων των υπο-σχέσεών της. Δεδομένου ότι στην *RQL* οι ερωτήσεις επεκτείνονται αυτόματα με βάση τη σχέση υπαλληλίας (σε κλάσεις και ιδιότητες), οργανώνουμε τις σχέσεις που αντιστοιχούν σε στιγμιότυπα κλάσεων/ιδιοτήτων σε μια τέτοια ιεραρχία υπο-σχέσεων και χρησιμοποιούμε αυτή τη λειτουργικότητα της SQL3 για τη σάρωσή τους.

Το κλείσιμο ενός τέτοιου δείκτη είναι τετριμέννο: `close` cursor1

4.7.2.2 scan_next

Η συνάρτηση αυτή παίρνει ως όρισμα μία λειτουργία σάρωσης και επιστρέφει το επόμενο στοιχείο της συλλογής στην οποία αυτή αναφέρεται. Ανάλογα με τον τύπο της λειτουργίας σάρωσης μπορούμε να διακρίνουμε τις εξής περιπτώσεις:

- ◇ **Στιγμιότυπα κλάσης:** Επιστρέφεται το αναγνωριστικό του επόμενου στιγμιοτύπου της κλάσης.
- ◇ **Στιγμιότυπα ιδιοτήτων:** Επιστρέφονται τα αναγνωριστικά των κόμβων που συνδέει το επόμενο στιγμιότυπο της ιδιότητας.
- ◇ **Μετακλάση Class:** Επιστρέφεται το όνομα της επόμενης κλάσης.
- ◇ **Μετακλάση Property:** Επιστρέφεται το όνομα της επόμενης ιδιότητας, καθώς και οι κλάσεις ορισμού και τιμών της.

Όταν φτάσουμε στο τέλος μιας συλλογής, επιστρέφεται κατάλληλος κωδικός ώστε η διαδικασία αποτίμησης με τις εμφωλευμένες ανακυκλώσεις να περάσει στην επόμενη φάση.

4.7.2.3 getSub/SuperClassesOf, getSub/SuperPropertiesOf

Οι συναρτήσεις αυτές, που χρησιμοποιούν τον αλγόριθμο υπολογισμού πεπερασμένης κλειστότητας που περιγράφηκε παραπάνω είναι απαραίτητες για την αποτίμηση των ομώνυμων λειτουργιών της γλώσσας (οι οποίες μπορούν είτε να χρησιμοποιούνται ρητά σε μία ερώτηση είτε να υποδηλώνονται έμμεσα, όπως στην περίπτωση της αποτίμησης μικτών μονοπατιών - βλέπε ενότητα 3.3).

4.7.2.4 isInstanceOf

Η συνάρτηση αυτή αντιστοιχεί στη συνθήκη `instanceOf`, που ελέγχει αν ένας πόρος είναι άμεσο στιγμιότυπο μιας κλάσης. Όπως και οι προηγούμενες λειτουργίες, μπορεί να χρησιμοποιηθεί ρητά σε μία ερώτηση αλλά εμφανίζεται και στην περίπτωση της αποτίμησης μικτών μονοπατιών (βλέπε ενότητα 3.3).

Συγκρίνοντας την RQL με άλλες γλώσσες επερώτησης

5.1 Γλώσσες επερώτησης για RDF

Μέχρι σήμερα έχουν προταθεί και κάποιες άλλες γλώσσες επερώτησης για μεταδεδομένα RDF. Χονδρικά, οι γλώσσες αυτές μπορούν να χωριστούν σε δύο κατηγορίες:

- ◇ αυτές που ακολουθούν μια σχεσιακή προσέγγιση ή ερμηνεύουν τα μεταδεδομένα RDF με βάση το μοντέλο της XML.
- ◇ αυτές που αντιμετωπίζουν το RDF ως γλώσσα αναπαράστασης γνώσης και χρησιμοποιούν τεχνικές αναπαράστασης γνώσης και λογισμού (reasoning) για την επερώτησή τους.

Στη συνέχεια θα παρουσιάσουμε δύο προτάσεις από κάθε κατηγορία, τις RDF Query Specification και RDFdb από την πρώτη και τις SiLRI και Metalog από τη δεύτερη, και θα υποδείξουμε τις αδυναμίες και τα μειονεκτήματά τους, τα οποία οδήγησαν στην ανάγκη δημιουργίας της RQL.

5.1.1 Προσέγγιση βάσεων δεδομένων

5.1.1.1 RDF Query Specification (IBM)

Αυτή η προσέγγιση - η οποία προτάθηκε από την IBM - βασίζεται στο πακέτο λογισμικού RDF for XML [R4J], το οποίο ουσιαστικά είναι ένα Περιβάλλον Προγραμματισμού Εφαρμογών (Application Programming Interface - API) για μεταδεδομένα RDF. Το περιβάλλον αυτό παρέχει εργαλεία για τη δημιουργία αντικειμένων RDF καθώς και διάφορες λειτουργίες χειρισμού τέτοιων αντικειμένων. Ως επέκταση του περιβάλλοντος αυτού δημιουργήθηκε η γλώσσα επερώτησης που περιγράφεται στο [MS98a]. Τα βασικά χαρακτηριστικά αυτής της γλώσσας είναι:

◇ Η χρήση συντακτικού XML, το οποίο επιτρέπει την ενσωμάτωση ερωτήσεων σε έγγραφα RDF. Ουσιαστικά πρόκειται για ένα σύνολο από στοιχεία XML που αντιστοιχούν στα συνηθισμένα τμήματα μιας ερώτησης SQL (select, from, where, group, order, ποσοδείκτες κλπ) και τους ανάλογους τελεστές (λογικούς, πράξεις μεταξύ συνόλων κλπ). Οι αντίστοιχες "παράμετροι" των διαφόρων τμημάτων και τελεστών εκφράζονται μέσω γνωρισμάτων των αντίστοιχων στοιχείων XML.

◇ Όλες οι λειτουργίες της γλώσσας γίνονται πάνω σε συλλογές RDF. Για παράδειγμα, στο τμήμα του `from` μιας ερώτησης συναντάμε γνωρίσματα όπως `eachResource="BagOfURIs"`.

◇ Η δυνατότητα ορισμού εκφράσεων μονοπατιών, που επιτρέπουν τη διάσχιση ενός γράφου RDF, μέσω επώνυμων ιδιοτήτων. Για το σχηματισμό τέτοιων εκφράσεων απαιτείται ο προσδιορισμός ενός αντικειμένου, ως αρχικό σημείο του μονοπατιού. Έτσι, για παράδειγμα, μπορούμε να σχηματίσουμε την ερώτηση (βασισμένοι στο πολιτιστικό παράδειγμα του σχήματος 3.1 και θεωρώντας ότι υπάρχει μια συλλογή `BagOfPainterURIs`, που περιέχει τα αναγνωριστικά των πόρων $ns2$`#picasso` και $ns2$`#rodin`), "βρες τα στυλ όλων των πόρων που έχουν ζωγραφίσει οι πόροι που περιέχονται στο `BagOfPainterURIs`".

```
<rdfq:rdfquery>
  <rdfq:From eachResource="BagOfPainterURIs">
    <rdfq:Select>
      <rdfq:Property path="paints/has_style"/>
    </rdfq:Select>
  </rdfq:From>
</rdfq:rdfquery>
```

Η γλώσσα αυτή λοιπόν παρέχει την απαιτούμενη λειτουργικότητα μιας γλώσσας επερώτησης, η οποία όμως αντιμετωπίζει τα μεταδεδομένα RDF σαν απλά έγγραφα

XML. Ουσιαστικά, πρόκειται για μια παραλλαγή της γλώσσας επερώτησης εγγράφων XML *XQL*, με την προσθήκη της έννοιας των συλλογών του RDF. Έτσι, ερμηνεύει τα μεταδεδομένα RDF χρησιμοποιώντας το μοντέλο αντικειμένων εγγράφου (Document Object Model - DOM) και όχι το μοντέλο του RDF. Επίσης, η γλώσσα αυτή δεν χρησιμοποιεί καθόλου την πληροφορία για τα μεταδεδομένα που περιγράφεται σε σχήματα. Έτσι, χάνει αρκετή από τη σημασιολογία των περιγραφών RDF και βέβαια δεν μπορεί να εκμεταλλευτεί την πληροφορία του σχήματος, τόσο για το σχηματισμό όσο και για τη βελτιστοποίηση της αποτίμησης κάποιων ερωτήσεων. Γι'αυτό και η πρόταση αυτή δεν έχει χρησιμοποιηθεί ιδιαίτερα για την επερώτηση μεταδεδομένων RDF σε πραγματικά συστήματα.

5.1.1.2 RDFdb

Η RDFdb [Guh00] είναι ένα σύστημα αποθήκευσης RDF περιγραφών, οι περιγραφές αποθηκεύονται στην βάση δεδομένων με την μορφή τριάδων. Έχει υλοποιηθεί στην γλώσσα προγραμματισμού C πάνω από την βάση δεδομένων Berkeley DB. Παρέχει μια διεπαφή χρήσης που μοιάζει πολύ με την SQL και επιτρέπει την προσθήκη, διαγραφή και επερώτηση των τριάδων. Τα δεδομένα μπορούν να αποθηκευτούν στην βάση είτε δίνοντας την διεύθυνση του αρχείου που περιέχει τις RDF/XML περιγραφές, είτε προσθέτοντας τριάδες χρησιμοποιώντας την διεπαφή χρήσης π.χ. η εντολή `insert into [database_name] (arc source target)` αποθηκεύει στην βάση την τριάδα (arc source target). Η σύνταξη που παρέχεται για τις επερωτήσεις έχει την μορφή `select [variable1, variable2, ... ... ] from database where [constraint1, onstraint2, ...]`. Για παράδειγμα μια επερώτηση είναι η εξής: `select ?x from db where (odp#title ?x 'National Gallery of Art')` η οποία επιστρέφει τους πόρους που έχουν την ιδιότητα odp#title με τιμή National Gallery of Art. Εκτός από την δυνατότητα φόρτωσης ενός αρχείου παρέχει και την δυνατότητα διαγραφής όλων των τριάδων που έχουν φορτωθεί στην βάση από κάποιο αρχείο. Η rdfDB είναι έχει δοκιμαστεί για μεγάλους όγκους δεδομένων και είναι επεκτάσιμη (scalable).

5.1.2 Προσέγγιση αναπαράστασης γνώσης

Η δεύτερη αυτή προσέγγιση - που φαίνεται κατ'αρχήν να ακολουθείται και από την ομάδα εργασίας του RDF στο W3C, καθώς κάποια μέλη της ομάδας αυτής συμμετέχουν σε αυτές τις προτάσεις - προέρχεται κυρίως από ερευνητές από το χώρο των συστημάτων αναπαράστασης γνώσης. Η προσέγγιση αυτή φαίνεται πιο κοντά στην αρχική ιδέα της δημιουργίας του RDF, την περιγραφή πληροφορίας με τρόπο αναγνώσιμο από τον άνθρωπο αλλά και τέτοιο ώστε οι υπολογιστές να μπορούν να "κατανοήσουν" τη σημασιολογία της. Κάτω από αυτό το πρίσμα μπορούμε να σκιαγραφήσουμε μερικές βασικές προδιαγραφές, τις οποίες πρέπει να πληρεί μια γλώσσα επερώτησης για μεταδεδομένα RDF ([GLMB98], [DBSA98]):

◇ Ένα σύστημα αποθήκευσης για μεταδεδομένα RDF θα πρέπει να κατανοεί, αναπαριστά και εκμεταλλεύεται την εκφραστική δύναμη των RDF και RDF Schema.

◇ Η γλώσσα επερώτησης θα πρέπει να "αναφέρεται" στο μοντέλο δεδομένων του RDF, αδιαφορώντας για το χρησιμοποιούμενο συντακτικό XML που είναι απλά ένα από τους εναλλακτικούς τρόπους αναπαράστασης του μοντέλου.

◇ Όσον αφορά τη λειτουργικότητα της γλώσσας, αυτή θα πρέπει να περιλαμβάνει ένα μεγάλο εύρος λειτουργιών, από απλές ερωτήσεις σε ζεύγη γνωρισμάτων-τιμών ως διάσχιση μονοπατιών και περίπλοκες ερωτήσεις σε στυλ Datalog [Bid91]. Επίσης, μπορούμε να φανταστούμε αρκετές περιπτώσεις όπου είναι δυνατή η εξαγωγή συμπερασμάτων ή η ανακάλυψη πληροφορίας που περιγράφεται έμμεσα:

  ▷ Διάσχιση ιεραρχιών υπαλληλίας για την ανακάλυψη σχέσεων υπαλληλίας μεταξύ κλάσεων ή ιδιοτήτων που δεν συνδέονται άμεσα μέσω της μεταβατικής (transitive) αυτής ιδιότητας.

  ▷ Συμπλήρωση αντίστροφων ιδιοτήτων: Αν, για παράδειγμα, γνωρίζουμε ότι ο ζωγράφος $ns2$`#picasso` έχει ζωγραφίσει (`paints`) τον πίνακα ζωγραφικής $ns2$`#guernica`, τότε μπορούμε να εξάγουμε το συμπέρασμα ότι ο πίνακας αυτός ζωγραφίστηκε από (painted_by) τον $ns2$`#picasso`.

  ▷ Αυτόματη επέκτασης ερώτησης (automatic query expansion) - σε αντιστοιχία με ανάλογες τεχνικές που χρησιμοποιούνται σε συστήματα ανάκτησης πληροφορίας που βασίζονται σε θησαυρούς [Fos97]. Η ίδια τεχνική μπορεί να χρησιμοποιηθεί για την πλοήγηση σε ιεραρχίες υπαλληλίας και την εκμετάλλευση/ανακάλυψη πληροφορίας που δεν εκφράζεται ρητά για τιμές που εμφανίζονται στα άκρα ιδιοτήτων.

Με βάση αυτές τις αρχικές ιδέες και προδιαγραφές προτάθηκαν δύο γλώσσες επερώτησης και εξαγωγής συμπερασμάτων για το RDF, η SiLRI [DBSA98] και η Metalog [MS98b].

5.1.2.1 SiLRI

Η προσέγγιση αυτή [DBSA98], [BDS98] χρησιμοποιεί το μοντέλο δεδομένων της Frame Logic [KL89] προσπαθώντας να επιτύχει τους παραπάνω στόχους. Με βάση την πρόταση αυτή τα μεταδεδομένα RDF αποθηκεύονται σε μια βάση γνώσης που υποστηρίζει το μοντέλο της F-logic, με βάση σχήματα F-logic. Ουσιαστικά δηλαδή απαιτεί τη μετάφραση των σχημάτων RDF που περιγράφουν μεταδεδομένα RDF σε αντίστοιχα σχήματα F-logic, όπως το παρακάτω:

◇ Δηλώσεις κλάσεων

```
Person:: Resource.
Artist:: Person.
Object:: Resource.
```

◇ Δηλώσεις γνωρισμάτων

```
Person[first_name=>> Literal;
       last_name=>> Literal;
       creates=>> Object].
Object[has_title=>> Literal].
```

Στον ορισμό των γνωρισμάτων, η χρήση του συμβόλου "=>>" υπονοεί ότι οι τιμές του γνωρίσματος είναι σύνολα αυτού του τύπου. Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε τη γλώσσα επερώτησης του μοντέλου της F-logic για να επερωτήσουμε τα μεταδεδομένα αυτά, όπως:

"Βρες όλους τους πόρους που έχουν δημιουργήσει `creates` κάτι"

```
FORALL X,Y <- X[creates->>Y].
```

Ανάλογα με τις δηλώσεις του σχήματος, η προσέγγιση αυτή επιτρέπει βασικές λειτουργίες εξαγωγής συμπερασμάτων από μεταδεδομένα RDF. Για παράδειγμα, η F-logic επιτρέπει την εξαγωγή συμπερασμάτων που βασίζονται σε σχέσεις υποκλάσης ή εκλέπτυνσης (refinement), π.χ. περιορίζοντας τις απαντήσεις μιας ερώτησης με βάση στοιχεία του σχήματος. Επίσης, μας επιτρέπει να εκφράσουμε περιορισμούς στις τιμές των ιδιοτήτων και συνεπώς να κάνουμε ελέγχους εγκυρότητας ή να συμπεράνουμε ελλειπή πληροφορία για πόρους. Για παράδειγμα, ξέρουμε από το παραπάνω σχήμα ότι:

```
Person:: Resource[creates=>>Object].
```

Έτσι, με βάση τη σημασιολογία της F-logic, μπορούμε να συμπεράνουμε ότι όλοι οι πόροι που έχουν δημιουργήσει (`creates`) κάτι είναι στιγμιότυπα της κλάσης `Person`, ακόμα κι αν αυτή η πληροφορία δεν εκφράζεται ρητά. Επίσης, στην F-logic μπορούμε να εκφράσουμε και ρητά κάποιους κανόνες εξαγωγής συμπερασμάτων, όπως:

```
FORALL X,Y
    X:Object[created_by->>Y] <- Y:Person[creates->>X]
```

Ο παραπάνω κανόνας σημαίνει ότι αν ξέρουμε ότι ένας πόρος Y είναι τύπου `Person` και έχει δημιουργήσει ένα πόρο Y, τότε ο Y είναι τύπου `Object` και έχει δημιουργηθεί από τον X. Στη συνέχεια, η πληροφορία αυτή μπορεί να χρησιμοποιηθεί ως δεδομένο (fact), όπως δηλαδή θα γινόταν αν αυτή η πληροφορία εκφραζόταν ρητά σε κάποια περιγραφή που υπάρχει στη βάση. Μια άλλη κατηγορία κανόνων εξαγωγής συμπερασμάτων είναι η αυτόματη επέκταση ερώτησης, όπως παρουσιάζεται και στο παράδειγμα αντιστοίχησης σχημάτων ταξινόμησης που παρουσιάζεται στο [BDS98].

Επίσης, η F-logic επιτρέπει τη σύνταξη αναδρομικών ερωτήσεων (που μπορούν να χρησιμοποιηθούν για τη διάσχιση γράφων σε οποιοδήποτε βάθος). Τέλος, η γλώσσα αυτή παρέχει κάποια λειτουργικότητα για επερώτηση σχημάτων. Για παράδειγμα, μας επιτρέπει να ζητήσουμε όλα τα ονόματα γνωρισμάτων που έχει η κλάση `Person`, τα οποία για κάποιο στιγμιότυπο της κλάσης έχουν την τιμή "Picasso":

```
FORALL X, P, T <- X:Person[P=>>T] AND X[P->>"Picasso"]
```

Η παραπάνω ερώτηση θα μας δώσει ως αποτέλεσμα την ιδιότητα `last_name`, με πεδίο τιμών `Literal`.

Παρολαυτά, η προσέγγιση αυτή έχει διάφορα προβλήματα. Κατ'αρχήν, πολλά από τα στοιχεία του RDF δεν μπορούν να εκφραστούν στην F-logic, όπως οι συλλογές RDF ή οι υποστασιοποιημένες προτάσεις. Το πιο σημαντικό μειονέκτημα όμως είναι ότι το μοντέλο της F-logic είναι διαφορετικό από αυτό των RDF και RDF Schema. Πράγματι, το μοντέλο αυτό είναι ένα οντοκεντρικό μοντέλο που δεν χειρίζεται τις ιδιότητες ως πολίτες πρώτης κατηγορίας αλλά ως γνωρίσματα κλάσεων. Έτσι, δεν μπορεί να αναπαραστήσει π.χ. σχέσεις υπαλληλίας μεταξύ ιδιοτήτων, ενώ δεν μας επιτρέπει να εκφράσουμε ερωτήσεις στις οποίες γνωρίζουμε μόνο ονόματα ιδιοτήτων (καθώς ακόμα και οι εκφράσεις μονοπατιών πρέπει να ξεκινούν από στιγμιότυπα μιας γνωστής κλάσης). Επίσης, οι ορισμοί κλάσεων ουσιαστικά δηλώνουν ένα τύπο για τα στιγμιότυπά του, που περιλαμβάνει τους ορισμούς των γνωρισμάτων. Αυτό δεν ισχύει στην περίπτωση του RDF, όπου οι ιδιοτητες ορίζονται ανεξάρτητα. Επίσης, οι τιμές των γνωρισμάτων έχουν ένα αυστηρά καθορισμένο τύπο. Έτσι, προφανώς, ο μόνος τροπος να αναπαραστήσουμε προαιρετικές και επαναλαμβανόμενες ιδιότητες είναι μέσω τιμών συνόλων, αναπαράσταση που δε διατηρεί τη σημασιολογία του μοντέλου του RDF.

Συνεπώς, το μοντέλο της F-logic είναι πολύ περιοριστικό για τους σκοπούς του RDF, καθώς δεν δίνει την ελευθερία για σύνθεση πληροφορίας από διάφορα σχήματα και ουσιαστικά προϋποθέτει την ύπαρξη ενός "κεντρικού", προκαθορισμένου σχήματος το οποίο ακολουθούν όλοι οι δημιουργοί περιγραφών. για το RDF Schema. Πράγματι, η γλώσσα αυτή προτάθηκε ως τμήμα του συστήματος Ontobroker [FDES98], [BFP98], που χρησιμοποιεί την F-logic ως εσωτερική αναπαράσταση, με σκοπό να επιτρέψει την αποθήκευση μεταδεδομένων RDF που αναμένεται να χρησιμοποιηθεί πολυ πιο ευρέως από τη γλώσσα περιγραφής μεταδεδομένων που προτάθηκε αρχικά από το Ontobroker. Έτσι, δεν έγινε καμμία προσπάθεια αναπαράστασης του μοντέλου του RDF με τα ειδικά χαρακτηριστικά του αλλά χρησιμοποιήθηκε ουσιαστικά ως συντακτικό για την αναπαράσταση μεταδεδομένων για το Ontobroker.

5.1.2.2 Metalog

H Metalog [MS98b] - που επίσης προτάθηκε από μέλη του W3C - προτείνει ουσιαστικά ένα σύνολο από επεκτάσεις στο μοντέλο του RDF, ώστε να μπορεί να αναπαραστήσει λογικά προγράμματα και κανόνες εξαγωγής συμπερασμάτων. Επίσης, η Metalog περιλαμβάνει και μια γλώσσα επερώτησης για τέτοιες λογικές εκφράσεις, που χρησιμοποιεί μάλιστα συντακτικό φυσικής γλώσσας. Η γλώσσα αυτή αποτελεί ουσιαστικά ένα είδος σχήματος για μεταδεδομένα RDF, το οποίο βασίζεται στον κατηγορηματικό λογισμό πρώτης τάξης, και στο οποίο επίσης μπορούν να εκφραστούν κανόνες εξαγωγής συμπερασμάτων. Ένα παράδειγμα τέτοιου σχήματος (τμήμα του παραδείγματος 3.1) και κανόνων εξαγωγής συμπερασμάτων που μπορούν να εκφραστούν είναι το εξής:

```
creates(X, Y) => in(X, Person)
creates(X, Y) => in(Y, Object)

creates(X,Y) => created_by(Y, X)
```

Στη συνέχεια, μπορεί να χρησιμοποιηθεί κάποια γλώσσα λογικού προγραμματισμού για την επερώτηση πληροφορίας με βάση τέτοια σχήματα. Για την υλοποίηση της Metalog προτείνεται η χρήση την Datalog [Bid91], παρέχοντας επίσης και ένα συντακτικό φυσικής γλώσσας - που μεταφράζεται σε Datalog - για τη διευκόλυνση του χρήστη. Με τη γλώσσα αυτή μπορούμε επίσης να εκφράσουμε ερωτήσεις που χρησιμοποιούν πληροφορία που εξάγεται από κανόνες, ζητώντας - π.χ. - τους πόρους που έχουν δημιουργηθεί από `created_by` τον $ns2$`#picasso`.

Η προσέγγιση αυτή ουσιαστικά δεν είναι μια γλώσσα επερώτησης για μεταδεδομένα RDF αλλά μια πρόταση επέκτασής του για να ανταποκριθεί στις απαιτήσεις συστημάτων αναπαράστασης γνώσης. Έτσι, δεν εκμεταλλεύεται την πληροφορία που περιέχεται σε σχήματα RDF. Επίσης, και αυτή η πρόταση - όπως και η προηγούμενη - δεν "ταιριάζει" ιδιαίτερα με την ημιδομημένη φύση των μεταδεδομένων RDF, αν και χειρίζεται τις ιδιότητες ως πολίτες πρώτης κατηγορίας, καθώς τα δομικά της στοιχεία είναι δυαδικές σχέσεις που αναπαριστούν ιδιότητες στο RDF.

5.2 Γλώσσες επερώτησης ημιδομημένων δεδομένων/εγγράφων XML

Οι γλώσσες επερώτησης ημιδομημένων δεδομένων και εγγράφων XML μπορούν επίσης να συγκριθούν με την *RQL*, καθώς έχουν μοιράζονται κάποιες ιδέες και αρχές. Πράγματι, οι γλώσσες επερώτησης ημιδομημένων όπως οι Lorel [AQM+97], UnQL [BDS95], StruQL [FFK+97] βασίζονται σε μοντέλα που δεν έχουν αυστηρό σύστημα τύπων (weakly-typed) για να μπορούν να αναπαραστήσουν ιδιαίτερα ή ασυνήθιστα χαρακτηριστικά (irregularities) των ημιδομημένων δεδομένων. Επίσης, οι γλώσσες αυτές χρησιμοποιούν γενικευμένες εκφράσεις μονοπατιών (με μεταβλητές σε κόμβους και ακμές) οι οποίες μπορούν να χρησιμοποιηθούν για την επερώτηση δεδομένων χωρίς τη γνώση (ή την ύπαρξη) σχήματος. Όμως, αν και τα μεταδεδομένα RDF μοιράζονται αρκετά στοιχεία με τα ημιδομημένα δεδομένα, το μοντέλο του RDF είναι πιο πλούσιο από τα μοντέλα ημιδομημένων, καθώς με την ύπαρξη σχημάτων περιλαμβάνει και ετικέττες για τους κόμβους (σε αντίθεση με ημιδομημένα μοντέλα όπως το OEM [CGH+94], [PGW95], [PGU96] στο οποίο βασίζεται η Lorel, που έχουν ετικέττες μόνο στις ακμές), όπως αναλύουμε και στην ενότητα 2.7.4. Επίσης, προφανώς, οι γλώσσες αυτές δεν αξιοποιούν την σημασιολογία των σχημάτων RDF για να διευκολύνουν το σχηματισμό ή τη βελτιστοποίηση της αποτίμησης ερωτήσεων καθώς και την πλοήγηση σε γράφους μεταδεδομένων. Πράγματι, ακόμα και στην περίπτωση των Dataguides [ABS99], η δομή των δεδομένων εξάγεται εκ των υστέρων, και δε χρησιμοποιείται προυπάρχουσα πληροφορία σχήματος, όπως αυτή που υπάρχει στην περίπτωση του RDF. Εξάλλου και το YAT [CCS00], [CDSS98], που βασίζεται σε ένα πρωτότυπο σύστημα τύπων που την αναπαράσταση και ανάκτηση πληροφορίας σε οποιοδήποτε επίπεδο (μοντέλου, σχήματος, δεδομένων), δεν μπορεί να αξιοποιήσει τη σημασιολογία που περιέχεται σε σχήματα RDF, υπό μορφή ιεραρχιών κλάσεων ή ιδιοτήτων.

Κάτι ανάλογο ισχύει και για τις γλώσσες επερώτησης εγγράφων XML, όπως οι XML-QL [DFF+99], XQL [Rob99] και XML-GL [CCD+99]. Πράγματι, όπως αναλύεται και στην ενότητα 2.7.4, αν και το RDF χρησιμοποιεί συντακτικό XML, τα δύο μοντέλα είναι διαφορετικά. Έτσι οι γλώσσες αυτές δεν είναι κατάλληλες για το RDF. Επίσης, όσον αφορά την επερώτηση σχημάτων, η XML-GL παρέχει τη δυνατότητα αναπαράστασης και επερώτησης Ορισμών Τύπων Εγγράφων (DTDs). Όμως, δεδομένου ότι τα σχήματα RDF δεν μπορούν να εκφραστούν ούτε σε DTDs (λόγω της ύπαρξης σχέσεων υπαλληλίας μεταξύ κλάσεων ή ιδιοτήτων) αλλά ούτε και σε σχήματα XML [TBMM00], [MM00] (καθώς οι σχέσεις επέκτασης και refinement που ορίζει μεταξύ κλάσεων - αλλά όχι και μεταξύ ιδιοτήτων - είναι λιγότερο εκφραστικές από τη σχέση υπαλληλίας του RDF Schema, όπως αναλύεται στο [KS]).

5.3 Γλώσσες επερώτησης πληροφορίας σχήματος

Τέλος, η RQL αντλεί ιδέες και από γλώσσες επερώτησης πληροφορίας σχημάτων που έχουν προταθεί για βάσεις δεδομένων, όπως η XSQL [KKS92], η SchemaSQL [LSS96] ή η POQL [CACS94], [CCM96], [ACC$^+$97]. Οι γλώσσες αυτές επιτρέπουν ερωτήσεις σχετικά με τη δομή των δεδομένων (π.χ. ονόματα πινάκων, γνωρισμάτων, κλάσεων κλπ.) σε σχεσιακά ή οντοκεντρικά συστήματα βάσεων δεδομένων. Έτσι, δεν μπορούν να χρησιμοποιηθούν για την επερώτηση σχημάτων RDF, καθώς όπως εξηγήσαμε προηγουμένως, το μοντέλο αυτό έχει χαρακτηριστικά που δεν μπορούν να εκφραστούν στα μοντέλα αυτά. Ουσιαστικά, οι ιδέες αυτών των προσεγγίσεων (μαζί με αυτές των γλωσσών επερώτησης για το διαδίκτυο, όπως οι WebSQL [MMM96], WebOQL [AM98], Araneus [MMA99]) αποτέλεσαν τη βάση για τη δημιουργία των συστημάτων διαχείρισης ημιδομημένων δεδομένων που παρουσιάστηκαν προηγουμένως. Μάλιστα, πολλές από τις ιδέες τους έχουν προσαρμοστεί στις ανάγκες και τις ιδιαιτερότητες του μοντέλου του RDF, για την επερώτηση σχημάτων με την RQL.

Επίλογος

Στην εργασία αυτή παρουσιάσαμε μια δηλωτική γλώσσα επερώτησης, την *RQL*, που είναι κατάλληλη για την επερώτηση μεταδεδομένων και σχημάτων RDF για πύλες κοινοτήτων διαδικτύου. Η γλώσσα μας, βασίζεται σε ένα μοντέλο που αναπαριστά τις ιδιαιτερότητες και τη σημασιολογία του μοντέλου του RDF, ενώ προσθέτει σε αυτό και ένα βασικό σύστημα τύπων, που είναι απαραίτητο για τον καλό ορισμό του αλλά και για τη βελτιστοποίηση της αποθήκευσης και της επερώτησης. Επίσης, η *RQL* είναι η πρώτη γλώσσα επερώτησης για RDF που διατηρεί τη σημασιολογία των RDF/RDFS (σε αντίθεση με άλλες προτάσεις που ουσιαστικά χρησιμοποιούν την RDF ως ένα συντακτικό για να εκφράσουν περιγραφές που υπακούουν κάποιο άλλο μοντέλο, π.χ. F-logic). Μάλιστα, η *RQL* προσφέρει τη λειτουργικότητα για ομοιόμορφη επερώτηση περιγραφών πόρων και σχημάτων RDF, με τρόπο φυσικό και διαφανή για τα μέλη μιας κοινότητας διαδικτύου.

Η *RQL* σχεδιάστηκε - όπως αναφέρθηκε προηγουμένως - για να ικανοποιήσει τις ανάγκες πυλών ιστών κοινοτήτων, όπως αυτές ορίστηκαν στο πλαίσιο του ευρωπαϊκού προγράμματος CWeb[1]. Μάλιστα, η εκφραστική δύναμη και η λειτουργικότητα που προσφέρει η *RQL* αξιολογήθηκαν και πρόκειται να χρησιμοποιηθούν σε δύο ακόμη ευρωπαϊκά προγράμματα, τα MESMUSES[2] και Ontoknowledge[3].

Προκειμένου να αξιολογήσουμε την αποδοτικότητα τόσο της αποθήκευσης όσο και του διερμηνέα ερωτήσεων της *RQL*, χρησιμοποιήσαμε ως βάση για πειράματα τα δεδομένα της πύλης Open Directory[4], που διατίθενται σε μορφή RDF/XML. Η πύλη αυτή περιλαμβάνει (ως σχήμα RDF) μια τεράστια ιεραρχία από 321,988 κατη-

[1] cweb.inria.fr
[2] cweb.inria.fr/private/mesmuses
[3] www.ontoknowledge.org
[4] www.dmoz.org

γορίες/θέματα (Topics), που απαιτούν περίπου 100MB για αποθήκευση. Ο αριθμός των πόρων που περιγράφονται ανέρχεται σε 2,123,382 δικτυακούς τόπους (το συνολικό μέγεθος των αρχείων περιγραφών είναι περίπου 700MB).

Με δεδομένο το μέγεθος αυτής της βάσης πειραμάτων, θα προσπαθήσουμε να αξιολογήσουμε την αποδοτικότητα του συστήματός μας. Επίσης, πρόκειται να εξετάσουμε την αποτελεσματικότητα διάφορων αλγεβρικών μεταφράσεων στις ερωτήσεις, όπως αυτές που προτάθηκαν στην ενότητα σχετικά με την αποτίμηση των ερωτήσεων από το διερμηνέα της RQL, αλλά και να πειραματιστούμε με διάφορες τεχνικές δεικτοδότησης, τόσο σε συλλογές σχήματος όσο και δεδομένων. Ιδιαίτερο ενδιαφέρον παρουσιάζει η περίπτωση ερωτήσεων που περιλαμβάνουν υπολογισμούς μεταβατικής κλειστότητας (transitive closure) σε ιεραρχίες κλάσεων ή ιδιοτήτων. Στην περίπτωση αυτή μπορούμε να χρησιμοποιήσουμε κατάλληλες - προϋπολογισμένες - δομές δεικτών, για να αποφύγουμε τέτοιους υπολογισμούς κατά τη διάρκεια της αποτίμησης μιας ερώτησης. Επίσης, σκοπεύουμε να εξετάσουμε και να συγκρίνουμε διάφορες ευρετικές μεθόδους για άλλους αλγεβρικούς μετασχηματισμούς ερωτήσεων (αναδιατάξεις συνδέσεων κλπ), πιθανώς βάσει κατάλληλων στατιστικών στοιχείων που θα συλλέξουμε, τόσο για ερωτήσεις στο σχήμα όσο και σε περιγραφές πόρων.

Μία άλλη πιθανή επέκταση της RQL θα μπορούσε να είναι η προσθήκη τελεστών αναδιάρθρωσης (restructuring) των αποτελεσμάτων των ερωτήσεων, όπως για παράδειγμα ομαδοποίηση αποτελεσμάτων βάσει των τιμών κάποιας μεταβλητής. Για παράδειγμα, είναι πιθανό σε μία ερώτηση να ζητήσουμε όλες τις ιδιότητες που μπορούν να εφαρμοστούν σε διάφορες κλάσεις, ομαδοποιημένες με βάση την κλάση αυτή.

Τέλος, ιδιαίτερο ενδιαφέρον παρουσιάζει το θέμα της ενημέρωσης (update) μιας βάσης περιγραφών πόρων. Στην παρούσα φάση, η RQL δεν υποστηρίζει ενημερώσεις, και η φόρτωση δεδομένων στη βάση γίνεται μαζικά (bulk loading). Ιδιαίτερα σημαντική, με δεδομένη την επεκτασιμότητα των σχημάτων RDF θα ήταν η υποστήριξη ενημερώσεων σε περίπτωση τροποποιήσεων.

A.0.1 Συναρτήσεις ελέγχου τύπων

A.0.1.1 isMetaclass, isClass, isProperty

Δεδομένου ότι οι μόνες μετακλάσεις που υπάρχουν είναι οι `Class`, `Property`, η συνάρτηση αυτή είναι τετριμμένη. Εξάλλου, οι συναρτήσεις `isClass`, `isProperty` ελέγχουν αν το όρισμά τους είναι το όνομα μιας κλάσης/ιδιότητας που έχει οριστεί σε κάποιο σχήμα RDF. Ουσιαστικά ισοδυναμούν με ένα έλεγχο για την ύπαρξη - ή μη - του ονόματος στη σχέση `Class/Property`. Οι συναρτήσεις αυτές μας παρέχουν όλη την επιθυμητή λειτουργικότητα για τον έλεγχο τύπων λειτουργιών σάρωσης (με μεταβλητές σχήματος για τις μετακλάσεις και δεδομένων για κλάσεις και ιδιότητες). Κατά τον έλεγχο τύπων, με βάση τη θέση ενός ονόματος σε μία ερώτηση, επιλέγουμε να καλέσουμε την κατάλληλη από τις παραπάνω συναρτήσεις. Έτσι, για παράδειγμα, στην ερώτηση:

`select` X, Y
`from` {X}creates{Y}

αρκεί να ελέγξουμε αν το όνομα `creates` αντιστοιχεί σε μία ιδιότητα. Αν όχι, τότε ο έλεγχος τύπων επιστρέφει λάθος. Ειδάλλως, θέτουμε τον τύπο της λειτουργίας αυτής σε "Property" και συνεχίζουμε τον έλεγχο τύπων της ερώτησης.

A.0.1.2 isBag, isSeq, isAlt

Ελέγχοντας την αντίστοιχη σχέση, η συνάρτηση αυτή απαντά αν το όρισμά της αντιστοιχεί στο όνομα της αντίστοιχης συλλογής RDF. Είναι απαραίτητες για τον έλεγχο τύπων σε λειτουργίες που επιτρέπονται μόνο σε συλλογές (πράξεις μεταξύ συλλογών, τελεστής `''[]''` σε λίστες κλπ.) καθώς και σε λειτουργίες σάρωσης (καθώς αυτές μπορούν να γίνουν μόνο σε συλλογές, ειδικές περιπτώσεις των οποίων είναι οι συλλογές στιγμιοτύπων κλάσεων και ιδιοτήτων).

subclassesof text *subclassesof(int4 x)

Description Computes the subclasses of class x. Returns The name of a table containing all subclasses of x (text is an internal Postgres type, containing a char * with the string value and an integer, holding the length of the string)

Parameters x: the integer id for the corresponding class (internal class name encoding) Algorithm Computes the transitive closure of the subclass relation on the subclass attribute, starting from the given class. Stores the result in a temporary relation (table) and returns the name of that table. The name of the table depends on the name of the class and so, if the required subclass tree has already been computed in the same transaction, the name of the table is returned without any computation.

superclassesof text *superclassesof(int4 x) Same as subclassesof, computing the superclasses of a given class x.

subpropertiesof text *subpropertiesof(int4 x) Same as subclassesof, computing the subproperies of a given property x (according to the subproperty table)

superpropertiesof text *superpropertiesof(int4 x); Same as subpropertiesof, computing the superproperties of a given property x.

issubclassof bool issubclassof(int4 x, int4 y) Description Checks if x is a subclass of Y

Returns true or false (postgres values for the type boolean)

Parameters x, y: the integer ids for the corresponding classes (internal class name encoding) Algorithm Starts from the supposed subclass (x) and computes the transitive closure according to the superclass relationship (bottom-up), using the $\delta - wavefront$ algorithm. Stops computing if y is found (returning true) or if the whole closure is computed and y is not found (returning false).

issuperclassof bool issuperclassof(int4 x, int4 y) Same as issubclassof, checking if x is a superclassof y. Equivalent with issubclassof(y, x)

issubpropertyof bool issubpropertyof(int4 x, int4 y) Same as issubclassof, for properties x, y (internal property name encoding). Also starts computing from the bottom of the hierarchy (the supposed subclass), looking for the supposed superclass in the transitive closure.

issuperpropertyof bool issuperpropertyof (int4 x, int4 y) Equivalent to issubpropertyof(y, x)

class RqlDatabase: public PgDatabase

Fields

PgTransaction *trans A handle to the transaction between the interpreter and the database server.

Constructor/destructor RqlDatabase(char *db)

Description Connects to the Postgres server, calling the constructor of its superclass (PgDatabase). Creates a new PgTransaction (trans).

Parameters char *db: the options needed for the constructor of PgDatabase. It is actually a string encoding for the list of parameters it accepts. For example: "dbName=odp_test, host=athena.ics.forth.gr, port=7777"

RqlDatabase();

Description Aborts the transaction and closes the connection with the Postgres server.

Methods

SubClassesOf char *SubClassesOf(int x) Description Computes the subclasses of class x and returns the name of the table containing the result. Actually sends an SQL query, using the aforementioned SPI function subclassesof: select subclassesof(x)

Parameters x: the integer id for the corresponding class (internal class name encoding)

Returns The name of a temporary table storing all the names of the subclasses of x (including x)

SubPropertiesOf char *SubPropertiesOf(int x) Description Same as SubClassesOf, for subproperties. Uses the SPI function subpropertiesof: select subpropertiesof(x)

isSubClassOf int isSubClassOf(int x, int y) Description Checks if x is a subclass of y. Actually sends the query: select issubclassof(x, y)

Returns TRUE if the query returns true or FALSE otherwise (internal interpreter defines)

isSubPropertyOf int isSubPropertyOf(int x, int y); Description Same as isSubClassOf for properties, using the query: select issubpropertyof(x, y) (see SPI functions above)

isInstanceOf int isInstanceOf(char *uri, int c) Description Checks if the resource, identified by uri, is a proper instance of the class with id c.

isClass int isClass(char *c) Description Checks if the string c is a name of a class

Returns The internal integer id of the corresponding class or 0, if c is not a class name

Algorithm Counts the results of the query: select id from class where lname='c'

isProperty int isProperty(char *p); Description Checks if the string p is a name of a property

Returns The internal integer id of the corresponding property or 0, if p is not a property name

Algorithm Counts the results of the query: select id from property where lname='p'

isBag int isBag(char *c) Description Checks if the string c is a name of a bag

isSeq int isSeq(char *c) Description Checks if the string c is a name of a sequence

getDomainOf int getDomainOf(int p) Description Returns the domain class of a property

Returns The internal id of the domain class or 0 if something went wrong

Parameters p: the internal id of a property.

getRangeOf int getRangeOf(int) Same as getDomainOf, for the range of a property

DB_scan_open PgCursor *DB_scan_open(Ql_res res) Description Opens a new scan on a collection.

Returns A PgCursor handle (Postgres class) to the corresponding cursor

Parameters Ql_res res: A structure, containing a union of a string value (for the case of the collections "Class" and "Property", and an integer value (the internal integer id of a class or a property)

Algorithm Sends an sql query as: declare mycursor cursor for select * from (t¡id¿*Class*Property)

DB_scan_next int DB_scan_next(PgCursor *c, Ql_scanop *s) Description Fetches the next tuple of the collection and moves the cursor forward, to proceed with the scan

Returns OK if everything was ok, or KO if the scan reached the end of the collection.

Parameters PgCursor *c: the cursor to be used Ql_scanop *s: the scan operation of the interpreter, referring to the cursor c.

Algorithm The result tuple fetched is stored (as a list of values) in the result field of this operation. It contains: 1 string value, if the collection is a class 2 string values, if the collection is a property (source, target pairs) 1 integer value, if the collection is the Metaclass "Class" 3 integer values, if the collection is the Metaclass "Property" (a property id and the ids of its domain and range classes)

DB_scan_close int DB_scan_close(PgCursor *c); Description Destroys a cursor. Used when a scan operation on a collection is finished (and the cursor is not useful anymore) (Note: the PgCursor destructor has a bug, and we can't destroy cursors...).

Returns OK if everything was OK, KO otherwise

Parameters c: The handle of the cursor to be destroyed

createResult int createResult(char *rname, int t) Description Creates a table in the database in order to store the results of a query.

Returns OK if everything was ok, KO otherwise

Parameters rname: the name of the table to be created (should be unique for everything to work properly) t: The list of types (kinds) of the variables in the result (unions?)

appendResult int appendResult(char *rname, char *tuple, int num) Description Appends a tuple in the result of a query.

Returns Ok if everything was ok, KO otherwise

Parameters rname: the name of the result table tuple: the tuple, encoded as a string (list of values) num: the number of attributes of the table (number of variables in the result)

Algorithm Stores tuples in a temporary buffer (in main memory), for efficiency. When the buffer is full, it flushes its contents in the table with name rname in the database.

destroyResult int destroyResult(char *rname) Drops a table. Used to drop a table - created in order to store the results of a query - is not needed anymore.

Returns Ok if everything was ok, KO otherwise

Parameters rname: the name of the table to be destroyed

getNameFromId char *getNameFromId(int id, int kind) Description Gets the name of a schema construct, using its internal id.

Returns The name (string) of the corresponding class or property (or literal etc ...)

Parameters: id: the internal id kind: the type of the construct (QL_CLASS or QL_PROPERTY) ;

Βιβλιογραφία

[ABS99] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[ACC+97] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, and J. Siméon. Querying Documents in Object Databases. *International Journal on Digital Libraries*, 1(1):5–18, April 1997.

[AF99] B. Amann and I. Fundulaki. Integrating Ontologies and Thesauri to Build RDF Schemas. In *ECDL-99: Research and Advanced Technologies for Digital Libraries*, Lecture Notes in Computer Science, pages 234–253, Paris, France, September 1999. Springer-Verlag.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[AK89] S. Abiteboul and P. Kanellakis. Object Identity as a Query Language Primitive. In *Proceedings of ACM SIGMOD Conf. on Management of Data*, pages 159–173, Portland, OR., June 1989.

[Ale00] S. Alexaki. Αποθήκευση μεταδεδομένων rdf για κοινότητες διαδικτύου, με τη χρήση σχεσιακών βάσεων δεδομένων. Master's thesis, Τμήμα Επιστήμης Υπολογιστών, Παν. Κρήτης, 2000.

[AM98] G.O. Arocena and A.O. Mendelzon. WebOQL: Restructuring Documents, Databases and Webs. In *Proceedings of ICDE '98*, Orlando, Florida, 1998.

[AQM+97] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The

Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.

[ASR94] B. Amann, M. Scholl, and A. Rizk. Querying typed hypertexts in multicard/O2. In *Proceedings of the ECHT'94 European Conference on Hypermedia Technologies*, Papers, pages 198–205, 1994.

[ASR95] B. Amann, M. Scholl, and A. Rizk. Schema-based authoring and querying of large hypertexts. *International Journal of Human-Computer Studies*, 43(3):281–299, 1995.

[AYJLS00] S. Amer-Yahia, H. Jagadish, L. Lakshmanan, and D. Srivastava. On bounding-schemas for ldap directories. In Carlo Zaniolo, Peter C. Lockemann, Marc H. Scholl, and Torsten Grust, editors, *Advances in Database Technology - EDBT 2000, 6th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings*, volume 1777 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2000.

[BDS95] P. Buneman, S.B. Davidson, and D. Suciu. Programming Constructs for Unstructured Data. In *Proceedings of International Workshop on Database Programming Languages*, Gubbio, Italy, 1995.

[BDS98] D. Brickley, S. Decker, and J. Saarela. Classification Schema Mapping: a simple demonstration using RDFIE and SiRPAC, 1998. Available at http://purl.org/net/rdf/papers/classmap.

[BFM98] T. Bray, C. Frankston, and A. Malhotra. Document content description for xml. W3C Note, July 1998. Available at http://www.w3.org/TR/-NOTE-dcd.

[BFP98] V.R. Benjamins, D. Fensel, and A. Gomez Perez. Knowledge management through ontologies. In *Proc. of the Second Int. Conf. on Practical Aspects of Knowledge Management (PAKM'98)*, Basel, Switzerland, October 1998.

[BG00] D. Brickley and R.V. Guha. Resource Description Framework (RDF) Schema Specification. W3C Candidate Recommendation, March 2000. Available at http://www.w3.org/TR/rdf-schema.

[BHL99] T. Bray, D. Hollander, and A. Layman. Namespaces in XML. W3C Recommendation, January 1999. Available at http://www.w3.org/-TR/REC-xml-names.

[Bid91] N. Bidoit. *Bases de Données Déductives (Présentation de Datalog).* Armand Colin, Paris, 1991.

[BLFIM98] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. IETF-RFC2396, August 1998.

[BPS98] T. Bray, J. Paoli, and C.M. Sperberg-McQueen. Extensible markup language (XML) 1.0. W3C Recommendation, February 1998. Available at http://www.w3.org/TR/REC-xml/.

[CACS94] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From Structured Documents to Novel Query Facilities. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 313–324, Minneapolis, Minnesota, May 1994.

[Car88] L. Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76(2/3):138–164, 1988.

[CB97] R.G.G. Cattell and D. Barry. *The Object Database Standard ODMG 2.0.* Morgan Kaufmann, 1997.

[CCD+99] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: a Graphical Language for Querying and Restructuring XML Documents. In *Proceedings of International World Wide Web Conference*, Toronto, Canada, 1999.

[CCM96] V. Christophides, S. Cluet, and G. Moerkotte. Evaluating Queries with Generalized Path Expressions. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 413–422, Montreal, Canada, June 1996.

[CCS00] V. Christophides, S. Cluet, and J. Siméon. On Wrapping Query Languages and Efficient XML Integration. In *Proceedings of ACM SIGMOD Conf. on Management of Data*, Dallas, TX., May 2000.

[CD99] J. Clark and S. DeRose. XML Path Language (XPath). W3C Recommendation, November 1999. Available at http://www.w3.org/TR/-xpath.

[CDSS98] S. Cluet, C. Delobel, J. Siméon, and K. Smaga. Your Mediators Need Data Conversion! In *Proceedings of ACM SIGMOD Conf. on Management of Data*, pages 177–188, Seattle, WA., June 1998.

[CGH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of heterogeneous information sources. In *IPSJ*, Tokyo, Japan, October 1994.

[DBSA98] S. Decker, D. Brickley, J. Saarela, and J. Angele. A query and inference service for rdf. In *W3C Query Languages Workshop*, Cambridge, Mass., 1998.

[DC98] M. Doerr and Nick Crofts. Electronic Communication on Diverse Data - The Role of an Object-Oriented CIDOC Reference Model. In *CIDOC'98 Conference*, Melbourne, Australia, October 1998.

[DC00] D. Florescu D. Chamberlin, J. Robie. Quilt: An xml query language for heterogeneous data sources. In *WebDB'2000*, pages 53–62, Dallas, US., May 2000.

[DFF+99] A. Deutsch, M.F. Fernandez, D. Florescu, A. Levy, and D. Suciu. A Query Language for XML. In *Proceedings of the 8th International World Wide Web Conference*, Toronto, 1999.

[DJM00] S. DeRose, R. Daniel Jr., and E. Maler. XML Pointer Language (XPointer). W3C Canditate Recommendation, July 2000. Available at http://www.w3.org/TR/xptr.

[DM99] L. Delcambre and D. Maier. Models for superimposed information. In P. P. Chen, D. W. Embley, J. Kouloumdjian, S. W. Liddle, and J. F. Roddick, editors, *Advances in Conceptual Modeling: ER '99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling, Paris, France, November 15-18, 1999, Proceedings*, volume 1727 of *Lecture Notes in Computer Science*, pages 264–280. Springer, 1999.

[DMOT00] S. DeRose, E. Maler, D. Orchard, and B. Trafford. XML Linking Language (XLink). W3C Candidate Recommendation, July 2000. Available at http://www.w3.org/TR/xlink.

[FA99] C. Finkelstein and P. Aiken. *Building Corporate Portals using XML*. McGraw-Hill, 1999.

[FDES98] Dieter Fensel, Stefan Decker, Michael Erdmann, and Rudi Studer. Ontobroker: Or How to Enable Intelligent Access to the WWW. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management*, Banff, Canada, April 1998.

[FFK+97] M.F. Fernandez, D. Florescu, J. Kang, A.Y. Levy, and D. Suciu. System Demonstration - Strudel: A Web-site Management System. In *Proceedings of ACM SIGMOD Conf. on Management of Data*, Tucson, AZ., May 1997. Exhibition Program.

[Fos97] D.J. Foskett. Theory of clumps. In K. Sparck Jones and P. Willett, editors, *Readings in Information Retrieval*, pages 111–134. Morgan Kaufmann, 1997.

[GLMB98] R. V. Guha, O. Lassila, E. Miller, and D. Brickley. Enabling inference. In *W3C Query Languages Workshop*, Cambridge, Mass., 1998.

[Guh96] R.V. Guha. Meta Content Framework, 1996. http://www.guha.com/-mcf/mcf_spec.html.

[Guh00] R.V. Guha. RDFdb, 2000. Available at http://web1.guha.com/rdfdb/.

[JLM+99] H. Jagadish, L. Lakshmanan, T. Milo, D. Srivastava, and D. Vista. Querying network directories. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 133–144. ACM Press, 1999.

[KKS92] M. Kifer, W. Kim, and Y. Sagiv. Querying object-oriented databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 393–402, 1992.

[KL89] M. Kifer and G. Lausen. F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. In *Proceedings of ACM SIGMOD Conf. on Management of Data*, volume 18, pages 134–146, Portland, Oregon, June 1989.

[KS] G.M. Kuper and J. Siméon. Subsumption for XML Types. To appear in the International Conference on Database Theory (ICDT 2001), January 2001, London, UK.

[LJM+98] A. Layman, E. Jung, E. Maler, H.S. Thompson, J. Paoli, J. Tigue, N.H. Mikula, and S. De Rose. XML-Data. W3C Note, January 1998. Available at http://www.w3.org/TR/1998/NOTE-XML-data.

[LLJ96] C. Lagoze, C.A. Lynch, and R. Daniel Jr. The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata. Technical Report 1593, Cornell University, June 1996. Available at http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR96-1593.

[LS99] O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, February 1999. Available at http://www.w3.org/TR/REC-rdf-syntax.

[LSS96] L.V.S. Lakshmanan, F. Sadri, and I.N. Subramanian. SchemaSQL - a language for interoperability in relational multi-database systems. In *Proceedings of International Conference on Very Large Databases (VLDB)*, pages 239–250, Bombay, India, September 1996.

[MD99] D. Maier and L. Delcambre. Superimposed information for the internet. In *ACM SIGMOD Workshop on The Web and Databases Philadelphia, Pennsylvania, June 3-4*, pages 1–9, 1999.

[Mil96] J. Miller. PICS Label Distribution Label Syntax and Communication Protocols, v1.1. W3C Recommendation, October 1996. Available at http://www.w3.org/PICS.

[MM00] M. Maloney and A. Malhotra. XML schema part 2: Datatypes. W3C Working Draft, April 2000. Available at http://www.w3.org/TR/-xmlschema-2/.

[MMA99] G. Mecca, P. Merialdo, and P. Atzeni. Araneus in the Era of XML. *Bulletin of IEEE Computer Society Technical Committee on Data Engineering*, 1999.

[MMM96] A.O. Mendelzon, G.A. Mihaila, and T. Milo. Querying the World Wide Web. In *Proceedings of the Parallel and Distributed Information Systems*, Orlando, Florida, 1996.

[Mom00] B. Momjian. *PostgreSQL: Introduction and Concepts*. Addison-Wesley, 2000.

[MR93] I.S. Mumick and K.A. Ross. Noodle: A Language for Declarative Querying in an Object-Oriented Database. In *Proceedings of International Conference on Deductive and Object-Oriented Databases (DOOD)*, pages 360–378, Phoenix, Arizona, December 1993.

[MS98a] A. Malhotra and N. Sundaresan. Rdf query specification. In *W3C Query Languages Workshop*, Cambridge, Mass., 1998.

[MS98b] M. Marchiori and J. Saarela. Query + metadata + logic = metalog. In *W3C Query Languages Workshop*, Cambridge, Mass., 1998.

[PGU96] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. MedMaker: A Mediation System Based on Declarative Specifications. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pages 132–141, New Orleans, LA., February 1996.

[PGW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pages 251–260, Taipei, Taiwan, March 1995.

[QHK91] G. Qadah, L. Henschen, and J. Kim. Efficient Algorithms for the Instantiated Transitive Closure Queries. *IEEE Transactions on Software Engineering*, 17(3):296–309, 1991.

[R4J] IBM's RDF for Java package. http://www.alphaworks.ibm.com/-formula/rdfxml/.

[RDF] Some proposed RDF APIs.
GINF: http://www-db.stanford.edu/~melnik/rdf/api.html,
RADIX: http://www.mailbase.ac.uk/lists/rdf-dev/1999-06/0002.html,
Netscape Communicator: http://lxr.mozilla.org/seamonkey/source/-rdf/base/idl/,
RDF for Java: http://www.alphaworks.ibm.com/formula/rdfxml/.

[Rob99] J. Robie. XQL (XML Query Language), 1999. http://metalab.unc.edu/xql/xql-proposal.html.

[SBM99a] M. Stonebraker, P. Brown, and D. Moore. *Object-Relational DBMSs: Tracking the next great wave*. Morgan-Kaufmann, San Francisco, 1999.

[SBM99b] M. Stonebraker, P. Brown, and D. Moore. *Object-Relational DBMSs: Tracking the next great wave*. Morgan-Kaufmann, San Francisco, 1999.

[SJGP90] M. Stonebraker, A. Jhingran, J. Goh, and S. Potamianos. On Rules, Procedures, Caching and Views in Database Systems. Technical Report UCB/ERL M90/36, University of California, Berkeley, 1990.

[TBMM00] H.S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML schema part 1: Structures. W3C Working Draft, April 2000. Available at http://www.w3.org/TR/xmlschema-1/.

[Wei96] S. Weibel. A proposed convention for embedding metadata in html. W3C Technical Report, June 1996. Available at http://www.w3.org/-Search/9605-Indexing-Workshop/ReportOutcomes/S6Group2.

[WMD95] S. Weibel, J. Miller, and R. Daniel. Dublin Core. In *OCLC/NCSA metadata workshop report*, 1995.