Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Πρότυπο Σύστημα Ομότιμων Κόμβων Βασισμένο σε Σχήματα RDF

Διπλωματικη Εργασια

της

ΕΙΡΗΝΗΣ Κ. ΣΠΥΡΟΠΟΥΛΟΥ

Επιβλέπων: Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εργαστηριο Συστηματων Βασεων Γνωσεων και Δεδομενων

Αθήνα, Οκτώβριος 2005

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων

Πρότυπο Σύστημα Ομότιμων Κόμβων Βασισμένο σε Σχήματα RDF

Διπλωματικη Εργασια

της

ΕΙΡΗΝΗΣ Κ. ΣΠΥΡΟΠΟΥΛΟΥ

Επιβλέπων: Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Οκτωβρίου 2005.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Τιμολέων Σελλής	Ιωάννης Βασιλείου	Νικόλαος Μήτρου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2005

(Υπογραφή)

.....

Ειρηνη Σπυροπουλου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Τιμολέοντα Σελλή για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων. Επίσης ευχαριστώ ιδιαίτερα τον Δρ. Θοδωρή Δαλαμάγκα για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Περίληψη

Ένα σύστημα ομότιμων κόμβων αποτελείται από ένα σύνολο αυτόνομων υπολογιστικών κόμβων στο Διαδίκτυο, οι οποίοι συνεργάζονται με σκοπό την ανταλλαγή δεδομένων. Στα συστήματα ομότιμων κόμβων που χρησιμοποιούνται ευρέως σήμερα, η αναζήτηση πληροφορίας γίνεται με χρήση λέξεων κλειδιών. Η ανάγκη για πιο εκφραστικές λειτουργίες, σε συνδυασμό με την ανάπτυξη του Σημασιολογικού Ιστού, οδήγησε στα συστήματα ομότιμων κόμβων βασισμένα σε σχήματα. Στα συστήματα αυτά κάθε κόμβος χρησιμοποιεί ένα σχήμα με βάση το οποίο οργανώνει τα τοπικά διαθέσιμα δεδομένα. Για να είναι δυνατή η αναζήτηση δεδομένων στα συστήματα αυτά υπάρχουν δύο τρόποι. Ο πρώτος είναι όλοι οι κόμβοι να χρησιμοποιούν το ίδιο σχήμα κάτι το οποίο δεν είναι ευέλικτο. Ο δεύτερος τρόπος δίνει την αυτονομία σε κάθε κόμβο να επιλέγει όποιο σχήμα θέλει και απαιτεί την ύπαρξη κανόνων αντιστοίχισης μεταξύ των σχημάτων για να μπορούν να αποτιμώνται οι ερωτήσεις. Αυτός ο τρόπος προσφέρει ευελιξία όμως δεν υποστηρίζει την αυτόματη δημιουργία και τη δυναμική ανανέωση των κανόνων, που είναι απαραίτητες για ένα σύστημα ομότιμων κόμβων.

Στόχος της διπλωματικής εργασίας είναι η ανάπτυξη ενός συστήματος ομότιμων κόμβων βασισμένο σε σχήματα το οποίο (α) θα επιτρέπει μια σχετική ευελιξία στην χρήση των σχημάτων και (β) θα δίνει την δυνατότητα μετασχηματισμού ερωτήσεων χωρίς την ανάγκη διατύπωσης κανόνων αντιστοίχισης μεταξύ σχημάτων, ξρησιμοποιώντας κόμβους με σχήματα RDF που αποτελούν υποσύνολα-όψεις ενός βασικού σχήματος (καθολικό σχήμα).

Λέξεις Κλειδιά

Σύστημα ομότιμων κόμβων, Σύστημα ομότιμων κόμβων βασισμένο σε σχήματα, Σημασιολογικός Ιστός, RDF/S, RQL, Jxta

Abstract

A peer-to-peer system is a set of autonomous computing nodes (the peers) which cooperate in order to exchange data. The peers in the peer-to-peer systems that are widely used today, rely on simple keyword selection in order to search for data. The need for richer facilities in exchanging data, as well as, the evolution of the Semantic Web, led to the evolution of the schema-based peer-to-peer systems. In those systems every node uses a schema to organize the local data. So the are two ways in order for data search to be feasible. The first but not so flexible way implies that every node uses the same schema. The second way gives every node the flexibility to choose a schema according with its needs, but on the same time requires the existence of mapping rules in order for queries to be replied. This way though, doesn't offer automatic creation and dynamic renewal of the mapping rules which would be essential for peer-to-peer systems.

This diploma thesis aims to the development of a schema-based peer-to-peer system that allows a certain flexibility for schema selection and on the same time enables query transformation without the use of mapping rules. The peers use RDF schemas that are subsets (views) of a big common schema called global schema.

Keywords

Peer-to-peer, Schema-based peer-to-peer, Semantic Web, RDF/S, RQL, Jxta

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Ο Παγκόσμιος Ιστός αποτελεί χώρο διακίνησης τεράστιου όγκου πληροφοριών. Ωστόσο, η συντριπτική πλειοψηφία των πληροφοριών του Ιστού, είναι προσανατολισμένη προς τον άνθρωπο-χρήστη και δεν είναι κατανοητή από τις εφαρμογές. Για να αξιοποιηθεί λοιπόν η διαθέσιμη πληροφορία και να γίνει πιο εύκολη η ανταλλαγή και η επεξεργασία της, ο Παγκόσμιος Ιστός εξελίσσεται στο Σημασιολογικό Ιστό.

Ο Σημασιολογικός Ιστός, είναι μια εξέλιξη του σημερινού Ιστού, μέσα στον οποίο δίνεται καλά ορισμένο νόημα στην πληροφορία που διακινείται, διευκολύνοντας τη συνεργασία μεταξύ υπολογιστή και ανθρώπου [3]. Πιο συγκεκριμένα δίνει τη δυνατότητα καλύτερης πρόσβασης σε μεγάλο όγκο πηγών πληροφορίας, καθώς και πιο αποτελεσματικής διακίνησης των πληροφοριών, χρησιμοποιώντας δεδομένα που τις περιγράφουν και ονομάζονται "μεταδεδομένα". Η καλύτερη γνώση της σημασίας, της χρήσης και της ποιότητας των πηγών διευκολύνει σημαντικά τη δυνατότητα πρόσβασης σε πηγές του Ιστού και την αυτόματη επεξεργασία του περιεχομένου που υπάρχει διαθέσιμο στο Διαδίκτυο βάσει του νοήματος και όχι μόνο της μορφής της πληροφορίας.

Ένα από τα πιο βασικά θέματα για την ανάπτυξη του Σημασιολογικού Ιστού είναι το να μπορούν οι υπολογιστές να ανταλλάσσουν δεδομένα μεταξύ εφαρμογών. Σε ένα ανοιχτό περιβάλλον όπως είναι ο Σημασιολογικός Ιστός χρειάζεται ένα ευέλικτο και δυναμικό μοντέλο ανταλλαγής δεδομένων όπως είναι τα συστήματα ομότιμων κόμβων (Peer-to-Peer systems).

Ένα σύστημα ομότιμων κόμβων αποτελείται από ένα σύνολο αυτόνομων υπολογιστικών κόμβων, οι οποίοι συνεργάζονται με σκοπό την ανταλλαγή δεδομένων. Τα συστήματα ομότιμων κόμβων που χρησιμοποιούνται ευρέως σήμερα κυρίως για την ανταλλαγή αρχείων μουσικής, έχουν πολύ μικρές δυνατότητες διαχείρισης δεδομένων. Η αναζήτηση πληροφορίας στα περισσότερα από αυτά γίνεται με χρήση λέξεων κλειδιών (keyword-based search).

Η ανάγκη για πιο εκφραστικές λειτουργίες, σε συνδυασμό με την ανάπτυξη του Σημασιολογικού Ιστού, οδήγησε στα συστήματα ομότιμων κόμβων που είναι βασισμένα σε σχήματα (schema-based peer-to-peer systems). Στα συστήματα αυτά κάθε κόμβος χρησιμοποιεί ένα σχήμα με βάση το οποίο οργανώνει τα τοπικά διαθέσιμα δεδομένα. Οι τεχνολογίες του Σημασιολογικού Ιστού δίνουν τη δυνατότητα οργάνωσης των δεδομένων μέσω σχημάτων που τα περιγράφουν.

Το πλαίσιο RDF είναι ένα τέτοιο εργαλείο αναπαράστασης μεταδεδομένων. Σε ένα RDF αρχείο ορίζονται δηλώσεις για αντικείμενα του Ιστού όπως σελίδες, συγγραφείς, προγράμματα κ.τ.λ. Μια επέκταση του πλαισίου RDF είναι το RDF Schema το οποίο παρέχει μηχανισμούς περιγραφής σχετικών αντικειμένων του Ιστού καθώς και των σχέσεων μεταξύ τους. Το RDF Schema βασίζεται σε κλάσεις και ιδιότητες έννοιες γνωστές από το χώρο των Αντικειμενοστρεφών συστημάτων. Η βασική διαφορά είναι ότι στο πλαίσιο RDF οι ιδιότητες ορίζονται ανεξάρτητα από τις κλάσεις.

Χρησιμοποιώντας λοιπόν τις τεχνολογίες του Σημασιολογικού Ιστού μπορούμε να δημιουργήσουμε συστήματα ομότιμων κόμβων με αυξημένη διαλειτουργικότητα τα οποία θα ανταλλάσσουν μεταξύ τους πληροφορία με νόημα και θα έχουν τη δυνατότητα διατύπωσης ερωτήσεων πιο εκφραστικών από αυτές που βασίζονται σε λέξεις κλειδιά.

1.1 Αντικείμενο της διπλωματικής

Το βασικό ζήτημα που προκύπτει για τα συστήματα ομότιμων κόμβων που είναι βασισμένα σε σχήματα, είναι πώς θα μπορούν οι κόμβοι να αναζητούν και να ανταλλάσσουν δεδομένα, διατηρώντας την αυτονομία τους. Δύο προσεγγίσεις έχουν προταθεί στην βιβλιογραφία:

1. Η πρώτη προσέγγιση απαιτεί να υπάρχει ένα κεντρικό σχήμα το οποίο θα χρησιμοποιούν όλοι οι κόμβοι [15]. Οι ερωτήσεις διατυπώνονται και αποτιμούνται με βάση το ίδιο σχήμα. Μια τέτοια λύση θα ήταν καλή για περιβάλλοντα με καθορισμένα όρια, όπως για παράδειγμα το τοπικό δίκτυο ενός οργανισμού. Όμως σε ένα ανοιχτό περιβάλλον όπως είναι ο Παγκόσμιος Ιστός χρειάζεται ένα πιο ευέλικτο μοντέλο που να επιτρέπει την χρήση πολλών σχημάτων.

2. Η δεύτερη προσέγγιση δίνει την αυτονομία σε κάθε κόμβο να επιλέγει όποιο σχήμα θέλει. Οι ερωτήσεις διατυπώνονται με βάση ένα σχήμα και αποτιμούνται με βάση άλλα σχήματα, μέσω μιας διαδικασίας μετασχηματισμού ερωτήσεων (query reformulation). Η διαδικασία αυτή απαιτεί την ύπαρξη κανόνων αντιστοίχισης (mapping rules) [8]. Όμως, σε ένα σύστημα ομότιμων κόμβων οι κόμβοι μπορούν να μπαίνουν και να βγαίνουν στο δίκτυο συνεχώς. Δεν είναι γνωστό επομένως εκ των προτέρων τα ζευγάρια των κόμβων μεταξύ των οποίων πρέπει να υπάρχουν κανόνες αντιστοίχισης. Επίσης, οι κανόνες αυτοί φτιάχονται χειρωνακτικά και είναι δύσκολη η συντήρησή τους.

Αντικείμενο της διπλωματικής είναι η ανάπτυξη ενός συστήματος ομότιμων κόμβων βασισμένο σε σχήματα το οποίο (α) θα επιτρέπει μια σχετική ευελιξία στην χρήση των σχημάτων και (β) θα δίνει την δυνατότητα μετασχηματισμού ερωτήσεων χωρίς την ανάγκη διατύπωσης κανόνων αντιστοίχισης μεταξύ σχημάτων. Το σύστημα δηλαδή βρίσκεται ανάμεσα στα δύο μοντέλα που περιγράφηκαν παραπάνω, από πλευράς ευελιξίας και δίνει τη δυνατότητα αυτόματου μετασχηματισμού ερωτήσεων. Χρησιμοποιεί κόμβους με σχήματα RDFS που αποτελούν υποσύνολα-όψεις (views) ενός βασικού σχήματος (καθολικό σχήμα).

Ένα παράδειγμα εφαρμογής του συστήματος αυτού θα ήταν η ανταλλαγή βιβλιογραφικών δεδομένων μεταξύ των ερευνητών. Κάθε ερευνητής θα συμμετείχε σε αυτό το σύστημα ομό-

τιμων κόμβων με ένα δικό του RDF σχήμα σύμφωνα με το οποίο θα οργάνωνε τις δημοσιεύσεις του και ταυτόχρονα θα μπορούσε να αναζητήσει ανάλογα δεδομένα από άλλους κόμβους. Σ΄ ένα τέτοιο σύστημα θα μπορούσαν να συμμετέχουν ως κόμβοι εκτός από μεμονωμένοι ερευνητές και εργαστήρια ή και συνέδρια.

1.2 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη σε επτά κεφάλαια: Στο Κεφάλαιο 2 δίνεται το θεωρητικό υπόβαθρο των βασικών τεχνολογιών που σχετίζονται με τη διπλωματική αυτή. Αρχικά περιγράφονται τα δίκτυα ομότιμων κόμβων, στη συνέχεια το πλαίσιο RDF και τέλος δίνεται μια μελέτη των γλωσσών ερωτήσεων για RDF. Στο Κεφάλαιο 3 αρχικά περιγράφονται οι σχετικές με το θέμα εργασίες και στη συνέχεια δίνεται ο στόχος της συγκεκριμένης εργασίας. Στο Κεφάλαιο 4 παρουσιάζεται η ανάλυση και η σχεδίαση του συστήματος, δηλαδή η περιγραφή των υποσυστημάτων και των εφαρμογών του. Η περιγραφή της υλοποίησης του συστήματος, με ανάλυση των βασικών αλγορίθμων καθώς και λεπτομέρειες σχετικά με τις πλατφόρμες και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν δίνεται στο Κεφάλαιο 5. Στο Κεφάλαιο 6 παρουσιάζεται ο έλεγχος καλής λειτουργίας του συστήματος με βάση ένα συγκεκριμένο σενάριο χρήσης. Τέλος στο Κεφάλαιο 7 δίνεται η συνεισφορά αυτής της διπλωματικής εργασίας, καθώς και μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζονται αναλυτικά οι τρεις βασικές τεχνολογίες που έχουν σχέση με την εργασία αυτή, δηλαδή τα συστήματα ομότιμων κόμβων, το πλαίσιο RDF και οι γλώσσες ερωτήσεων για RDF.

2.1 Συστήματα ομότιμων κόμβων

2.1.1 Τι είναι τα συστήματα ομότιμων κόμβων

Στα μεγάλα κατανεμημένα συστήματα όπως είναι ο Παγκόσμιος Ιστός, γίνονται εμφανή τα προβλήματα του παραδοσιακού μοντέλου πελάτη/εξυπηρετητή: Οι πηγές πληροφορίας βρίσκονται μαζεμένες σε λίγους κόμβους (εξυπηρετητές) στους οποίους συνδέονται πάρα πολλοί πελάτες [20]. Σε αυτούς πρέπει να παρέχεται αξιόπιστη πρόσβαση στους πόρους των εξυπηρετητών, με αποδεκτό χρόνο απόκρισης. 'Ολα αυτά σε συνδυασμό με το μικρό εύρος ζώνης των δικτύων δημιουργούν ένα σενάριο συμφόρησης. Αυτά τα προβλήματα έδωσαν κίνητρο στους επιστήμονες να σκεφτούν τεχνικές κατανομής της επεξεργασίας του φορτίου και του δικτυακού εύρους ζώνης σε όλους τους κόμβους που συμμετέχουν σε ένα κατανεμημένο πληροφοριακό σύστημα. Έτσι δημιουργήθηκαν τα συστήματα ομότιμων κόμβων (Peer-to-Peer Systems).

Σε ένα σύστημα ομότιμων κόμβων, κάθε κόμβος ενεργεί και σαν πελάτης και σαν εξυπηρετητής, παρέχοντας πρόσβαση σε κάποιους από τους πόρους του, παρέχοντας μέρος της επεξεργαστικής του ισχύος και/ή μέρος του μέρος του δίσκου του.

Οι αρχές που διέπουν τα συστήματα ομότιμων κόμβων είναι οι εξής:

- Η αρχή του μοιράσματος των πόρων: Όλα τα συστήματα ομότιμων κόμβων περιέχουν μία πτυχή της κοινής εκμετάλλευσης των πόρων, όπου οι πόροι μπορεί να είτε είναι φυσικοί πόροι, όπως χώρος στο δίσκο ή δικτυακό εύρος ζώνης, είτε και λογικοί, όπως υπηρεσίες ή διαφορετικοί τύποι γνώσης. Με βάση αυτή την αρχή μπορούν να δημιουργηθούν εφαρμογές οι οποίες δε θα μπορούσαν να ¨στηθούν' από ένα και μόνο κόμβο. Αυτό ήταν και το κίνητρο για τη δημιουργία του συστήματος ομότιμων κόμβων Napster.

- Η αρχή της αποκέντρωσης: Η αρχή αυτή είναι ένα άμεσο αποτέλεσμα του μοιράσματος των πόρων. Τα μέρη του συστήματος δεν διαχειρίζονται πλέον κεντρικά. Η αποκέντρωση είναι ιδιαιτέρως ενδιαφέρουσα για την αποφυγή αποτυχιών σε ένα σημείο ή συμφορήσεων απόδοσης στο σύστημα. Παραδείγματα πλήρως αποκεντρωμένων συστημάτων είναι το Gnutella και το Freenet.

- Η αρχή της αυτοοργάνωσης: Όταν ένα σύστημα ομότιμων κόμβων γίνει πλήρως αποκεντρωμένο, τότε δεν υπάρχει κάποιος κεντρικός κόμβος που να συντονίζει τις δραστηριότητές του, ούτε μια βάση δεδομένων για την αποθήκευση πληροφοριών για το σύστημα. Επομένως οι κόμβοι πρέπει να αυτοοργανόνονται βασισμένοι στην τοπική πληροφορία που έχουν και αλληλεπιδρώντας με τοπικά προσβάσιμους κόμβους (γείτονες).

2.1.2 Η εξέλιξη των συστημάτων ομότιμων κόμβων

Το πιο γνωστό σύστημα ομότιμων κόμβων, με διαφορά σε σχέση με τα υπόλοιπα, είναι το Napster [1]. Έγινε γνωστό σαν ένα σύστημα ανταλλαγής μουσικών αρχείων όμως προσέφερε και άλλες υπηρεσίες που άρεσαν στους χρήστες.

Από άποψη αρχιτεκτονικής το Napster είναι ένα πολύ απλό σύστημα ομότιμων κόμβων: Ο εξυπηρετητής του Napster κρατάει σε μια κεντρική βάση δεδομένων όλα τα μουσικά αρχεία που προσφέρονται από τους κόμβους που συμμετέχουν στο σύστημα. Οι πελάτες συνδέονται με τον εξυπηρετητή και του στέλνουν τη λίστα με τα αρχεία που προφέρουν. Οι νέοι χρήστες πρέπει πρώτα να δημιουργήσουν ένα λογαριασμό στον εξυπηρετητή του Napster γι΄ αυτό το σκοπό. Κάθε πελάτης μπορεί να στείλει αιτήσεις αναζήτησης στον εξυπηρετητή του Napster και να πάρει τη λίστα των πελατών που προσφέρουν το αρχείο που ζήτησε. Ο αιτών μπορεί να επιλέξει από αυτή τη λίστα ένα πελάτη από τον οποίο θα γίνει η μεταφορά του αρχείου. Συνεπώς το Napster δεν είναι ένα ¨καθαρόαιμο' σύστημα ομότιμων κόμβων αλλά ένας συνδυασμός συστήματος πελάτη/εξυπηρετητή και συστήματος ομότιμων κόμβων.

Το πρώτο πλήρως αποκεντρωμένο και αυτοοργανούμενο σύστημα ομότιμων κόμβων για ανταλλαγή αρχείων ήταν το Gnutella. Το Gnutella βασίζεται σε ένα πρωτόκολλο για κατανεμημένη αναζήτηση αρχείων. Ένας κόμβος για να συμμετέχει στο Gnutella πρέπει πρώτα να συνδεθεί με ένα γνωστό κόμβο του Gnutella. Εξειδικευμένοι εξυπηρετητές του Gnutella στέλνουν σε κάθε νεοεισερχόμενο κόμβο λίστες με κόμβους με τους οποίους μπορεί να συνδεθεί. Για την αναζήτηση ενός αρχείου ένας κόμβος στέλνει το μήνυμα στους γείτονές του. Κάθε μήνυμα περιέχει ένα πεδίο Time-to-Live που είναι ένας αριθμός που καθορίζει μετά από πόσους κόμβους το μήνυμα θα σταματήσει να διαδίδεται και μειώνεται κατά ένα σε κάθε κόμβο. Κάθε κόμβος ο οποίος λαμβάνει αυτό το μήνυμα, αν το πεδίο Time-To-Live του μηνύματος είναι μεγαλύτερο του 0, στέλνει το μήνυμα στους γείτονές του. Επίσης ελέγχει αν μπορεί να απαντήσει και αν ναι στέλνει την απάντηση στον κόμβο αποστολέα .

Το Gnutella από την πλευρά του χρήστη είναι ένα από και αποτελεσματικό πρωτόκολλο: Ο ρυθμός επιτυχίας στις ερωτήσεις αναζήτησης είναι αρκετά υψηλός, είναι ανεκτικό σε λάθη εξαιτίας της αποτυχίας κάποιων κόμβων και προσαρμόζεται εύκολα στο δυναμικό πληθυσμό των κόμβων. Όμως από την πλευρά του δικτύου καταναλώνει μεγάλο εύρος ζώνης.

Μια διαφορετική αρχιτεκτονική είναι αυτή του συστήματος Freenet. Το Freenet είναι ένα σύστημα ομότιμων κόμβων για τη δημοσίευση και την ανάκτηση αρχείων δεδομένων. Ο κύριος σκοπός του είναι να παρέχει μια υποδομή που προστατεύει την ανωνυμία των συγγραφέων και των αναγνωστών. Έχει σχεδιαστεί με τέτοιο τρόπο ώστε να είναι ανέφικτο να προσδιορίσει κανείς την προέλευση των αρχείων ή τον προορισμό των ερωτημάτων. Επίσης τα αρχεία όταν στέλνονται στο δίκτυο είναι κρυπτογραφημένα. Εκτός όμως από την προστασία της ανωνυμίας, το σύστημα Freenet υλοποιεί και μια άλλη ενδιαφέρουσα ιδέα: Ένα προσαρμοζόμενο σχέδιο δρομολόγησης για αποτελεσματικές αναζητήσεις στις φυσικές τοποθεσίες που είναι πιθανότερο να εμφανιστούν. Προκειμένου να βελτιωθεί η αποδοτικότητα της αναζήτησης, το Freenet πίνακες δρομολόγησης που ενημερώνονται δυναμικά καθώς συμβαίνουν ενημερώσεις και εισαγωγές νέων στοιχείων.

Όταν ένας κόμβος συνδέεται στο Freenet, πρέπει να γνωρίζει έναν ήδη υπάρχοντα κόμβο (όπως και στο Gnutella). Μέσω της αλληλεπίδρασης με το δίκτυο γεμίζει τον πίνακα δρομολόγησής του και έτσι η δομή του Freenet εξελίσσεται.

Τα συστήματα ομότιμων κόμβων που περιγράφηκαν στην ενότητα αυτή, διαφέρουν ως προς την αρχιτεκτονική και τις υπηρεσίες που προσφέρουν στους χρήστες. Υπάρχει και μια άλλη κατηγορία, που ονομάζονται συστήματα ομότιμων κόμβων βασισμένα σε σχήματα (schema-based peer-to-peer systems) και διαφέρουν με αυτά που περιγράφηκαν εδώ ως προς τον τρόπο αναζήτησης. Έχουν τη δυνατότητα διατύπωσης ερωτήσεων πιο εκφραστικών από αυτές που βασίζονται σε λέξεις κλειδιά. Σ' αυτή την κατηγορία συστημάτων ομότιμων κόμβων θα αναφερθούμε εκτενώς στο Κεφάλαιο 3.

2.2 Το πλαίσιο RDF

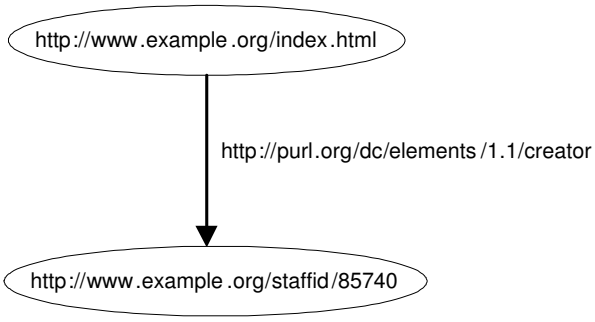
Το πλαίσιο RDF [12] είναι μια γλώσσα για την αναπαράσταση της πληροφορίας που περιγράφει τις πηγές πληροφοριών του παγκόσμιου ιστού. Προορίζεται για περιπτώσεις στις οποίες αυτή η πληροφορία χρειάζεται να μπορεί να επεξεργασθεί από διάφορες εφαρμογές και όχι μόνο απλά να παρουσιάζεται στους χρήστες. Το RDF παρέχει ένα κοινό μοντέλο για την αναπαράσταση αυτής της πληροφορίας έτσι ώστε να μπορεί να ανταλλάσσεται μεταξύ εφαρμογών, χωρίς να χάνει τη σημασία της. Το βασικότερο χαρακτηριστικό του μοντέλου RDF είναι η δυνατότητα να υπερθέτει περιγραφές για τις ίδιες πηγές σε εφαρμογές διαφορετικού περιεχομένου.

2.2.1 Το μοντέλο RDF

Το πλαίσιο RDF παρέχει μια γλώσσα αναπαράστασης μεταδεδομένων βασισμένη πάνω σε προσανατολισμένους γράφους στους οποίους οι κόμβοι αποτελούν τις πηγές (resources) και οι ακμές τις ιδιότητες (properties). Μια εντολή RDF αποτελείται από την τριπλέτα : υποκείμενο (subject), ιδιότητα (predicate) και αντικείμενο (object) που αποτελεί τιμή αυτής της ιδιότητας. Το μοντέλο RDF χρησιμοποιεί URIs σαν αναγνωριστικά για το υποκείμενο, την ιδιότητα και το αντικείμενο. Έστω για παράδειγμα η πρόταση στα Αγγλικά που περιγράφει την ημερομηνία δημιουργίας μιας σελίδας στο internet:

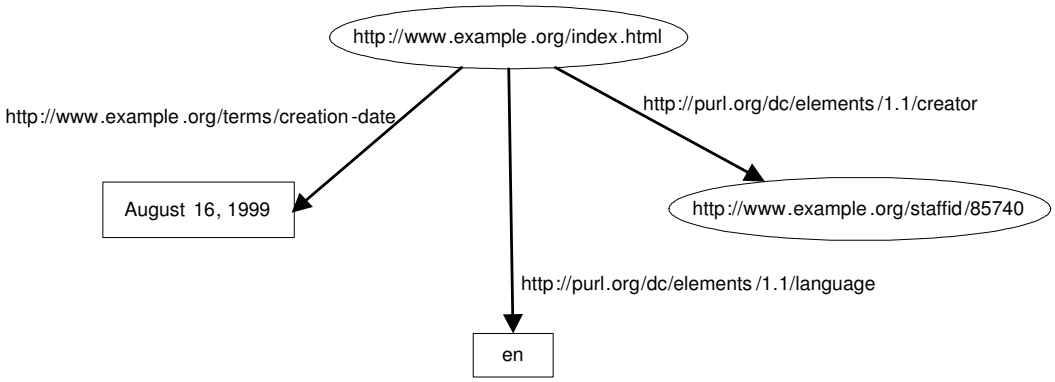
http://www.example.org/index.html has a creator whose value is John Smith

Το υποκείμενο για αυτή την πρόταση είναι το http://www.example.org/index.html, η ιδιότητα είναι το creator και το αντικείμενο John Smith. Στο Σχήμα 2.1 φαίνεται η αναπαράσταση αυτής της πρότασης με γράφο.



Σχήμα 2.1: Ο γράφος μιας απλής RDF έκφρασης.

Παρατηρούμε ότι χρησιμοποιείται ένας κόμβος για το υποκείμενο, ένας για το αντικείμενο και ένα τόξο για την ιδιότητα, κατευθυνόμενο από το υποκείμενο στο αντικείμενο. Στο παράδειγμα του Σχήματος 2.2, παρατηρούμε πώς αναπαριστάται ένα σύνολο εκφράσεων RDF και διαπιστώνουμε ότι τα αντικείμενα του RDF μπορούν να είναι είτε URIs είτε σταθερές τιμές που ονομάζονται κυριολεκτικά (λιτεραλς) και αναπαριστώνται από συμβολοσειρές προκειμένου να αντιπροσωπεύσουν ορισμένα είδη ιδιοτήτων. Τα κυριολεκτικά δεν μπορούν να χρησιμοποιηθούν ως υποκείμενα ή ως ιδιότητες σε RDF εκφράσεις.



Σχήμα 2.2: Πολλές Εκφράσεις για την ίδια πηγή.

Επειδή η αναπαράσταση με το γράφο δεν εξυπηρετεί πάντα, χρησιμοποιείται και ένας τρόπος γραφής των τριπλετών του γράφου. Σύμφωνα μ΄ αυτόν τον τρόπο το προηγούμενο παράδειγμα θα μπορούσε να γραφεί ως εξής:

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740>.
```

```
<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date> "August 16, 1999" .

<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/language> "en" .
```

Κάθε τριπλέτα αντιστοιχεί σε ένα βέλος στο γράφο, συμπεριλαμβανομένων και των κόμβων αρχής και τέλους τους οποίους συνδέει το βέλος.

Για περισσότερη ευκολία το RDF χρησιμοποιεί namespaces για τη σύντμηση των URIs. Έτσι έχοντας ορίσει τα namespaces:

```
ex:  http://www.example.org/
dc:http://purl.org/dc/elements/1.1/
exterms:http://www.example.org/terms/
exstaff:http://www.example.org/staffid/
```

Το προηγούμενο παράδειγμα μπορεί να γραφεί ως εξής:

```
ex:index.html dc:creator exstaff:85740 .
ex:index.html exterms:creation-date "August 16, 1999" .
ex:index.html dc:language "en" .
```

2.2.2 RDF/XML

Το πλαίσιο RDF παρέχει ένα συντακτικό σε μορφή XML(RDF/XML) για την καταγραφή και την ανταλλαγή των γράφων που περιγράψαμε στην προηγούμενη υποενότητα.

Σύμφωνα με το συντακτικό αυτό, το παράδειγμα που αναπαριστάται στο Σχήμα 2.2 γράφεται ως εξής:

```
1.<?xmlversion="1.0"?>
2.<rdf:RDFxmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.      xmlns:dc="http://purl.org/dc/elements/1.1/"
4.      xmlns:exterms="http://www.example.org/terms/">
5.   <rdf:Description rdf:about="http://www.example.org/index.html">
6.     <exterms:creation-date>August 16, 1999</exterms:creation-date>
7.     <dc:language>en</dc:language>
8.     <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
9.   </rdf:Description>
10.</rdf:RDF>
```

Η γραμμή 1 αποτελεί τη δήλωση της XML, που καθορίζει ότι το περιεχόμενο που ακολουθεί είναι XML. Η γραμμή 2 ξεκινά με ένα στοιχείο `<rdf:RDF`, που δηλώνει ότι το υπόλοιπο XML περιεχόμενο που ξεκινάει εδώ και τελειώνει με το `</rdf:RDF>` στη γραμμή 10, είναι

μια αναπαράσταση RDF. Οι γραμμές 3 και 4 ορίζουν τα namespaces exterms και dc. Τέλος οι γραμμές 5-9 αποτελούν το RDF/XML για τις εκφράσεις RDF που αναπαριστώνται στο γράφο.

Όπως στην Html το RDF/XML είναι μηχανή επεξεργάσιμη και χρησιμοποιώντας τα URIs μπορεί να συνδέσει κομμάτια πληροφοριών στον Ιστό. Εντούτοις αντίθετα από το συμβατικό υπερκείμενο hypertext, τα URIs μπορούν να αναφερθούν σε οποιοδήποτε ευπροσδιόριστη οντότητα, συμπεριλαμβανομένων και αυτών που δεν είναι άμεσα ανακτήσιμες από τον Ιστό, όπως για παράδειγμα επιχειρήσεις, άνθρωποι, γεγονότα ειδήσεων κτλ.

2.2.3 RDF Schema

Το πλαίσιο RDF παρέχει ένα τρόπο να εκφραστούν απλές εκφράσεις για πόρους, χρησιμοποιώντας ιδιότητες και τιμές. Παρόλα αυτά , οι κοινότητες χρηστών RDF χρειάζονται επίσης να καθορίσουν το λεξιλόγιο που αυτοί σκοπεύουν να χρησιμοποιήσουν σε εκείνες τις εκφράσεις και συγκεκριμένα, να δείξουν ότι περιγράφουν συγκεκριμένα είδη ή κατηγορίες πόρων και γι΄ αυτό θα χρησιμοποιήσουν συγκεκριμένες ιδιότητες στην περιγραφή εκείνων των πόρων. Το ίδιο το RDF δεν περέχει κανένα μέσο για τον προσδιορισμό κλάσεων και ιδιοτήτων. Αντίθετα διαθέτει μια γλώσσα ορισμού τύπων για την περιγραφή κλάσεων και ιδιοτήτων των κλάσεων αυτών που ονομάζεται RDFSchema.

Οι βασικοί τύποι που ορίζονται με τη βοήθεια του RDFSchema είναι rdfs:Class, rdfs:subClassOf rdf:Property και rdfs:subPropertyOf. Μια κλάση (rdfs:Class) αντιστοιχεί στη γενική έννοια ενός τύπου ή μιας κατηγορίας, όπως ακριβώς η ένοια της κλάσης στις αντικειμενοστεφείς γλώσσες προγραμματισμού. Στο RDFSchema ορίζεται επίσης και η έννοια της υποκλάσης (rdfs:SubClassOf) για την περιγραφή πιο ειδικών κατηγοριών μιας κατηγορίας.

Εκτός από την περιγραφή συγκεκριμένων κλάσεων πραγμάτων που θέλουν να περιγράψουν, οι κοινότητες χρηστών πρέπει επίσης να είναι σε θέση να περιγράψουν τις ιδιότητες αυτών των κλάσεων. Στο RDFSchema οι ιδιότητες περιγράφονται χρησιμοποιώντας την έκφραση rdf:Property. Επειδή όμως οι ιδιότητες και οι κλάσεις προορίζονται για να χρησιμοποιηθούν μαζί στα RDF δεδομένα παρέχεται και λεξιλόγιο για την υποστήριξη αυτής ακριβώς της λειτουργίας. Η έκφραση rdfs:domain χρησιμοποιείται για να δηλώσει την κλάση που αποτελεί πεδίο ορισμού μιας ιδιότητας ενώ η έκφραση rdfs:range για την κλάση που αποτελεί πεδίο τιμών της ιδιότητας. Το rdfs:range μπορεί επίσης να χρησιμοποιηθεί για να δείξει ότι η τιμή μιας ιδιότητας δίνεται από ένα κυριολεκτικό literal. Τέλος μια ιδιότητα μπορεί να είναι υποϊδιότητα μιας άλλης ιδιότητας και αυτό ορίζεται στο RDFSchema με την έκφραση rdfs:subPropertyOf. Τα rdfs:range και rdfs:domain που ισχύουν για μια ιδιότητα ισχύουν και για τις υποιδιότητές της.

Έστω ένα παράδειγμα το οποίο περιγράφει τα έργα τέχνης ενός μουσείου. Το κομμάτι της RDF περιγραφής σε RDF/XML που ορίζει τις κλάσεις, τις υποκλάσεις και τις ιδιότητες φαίνεται παρακάτω.

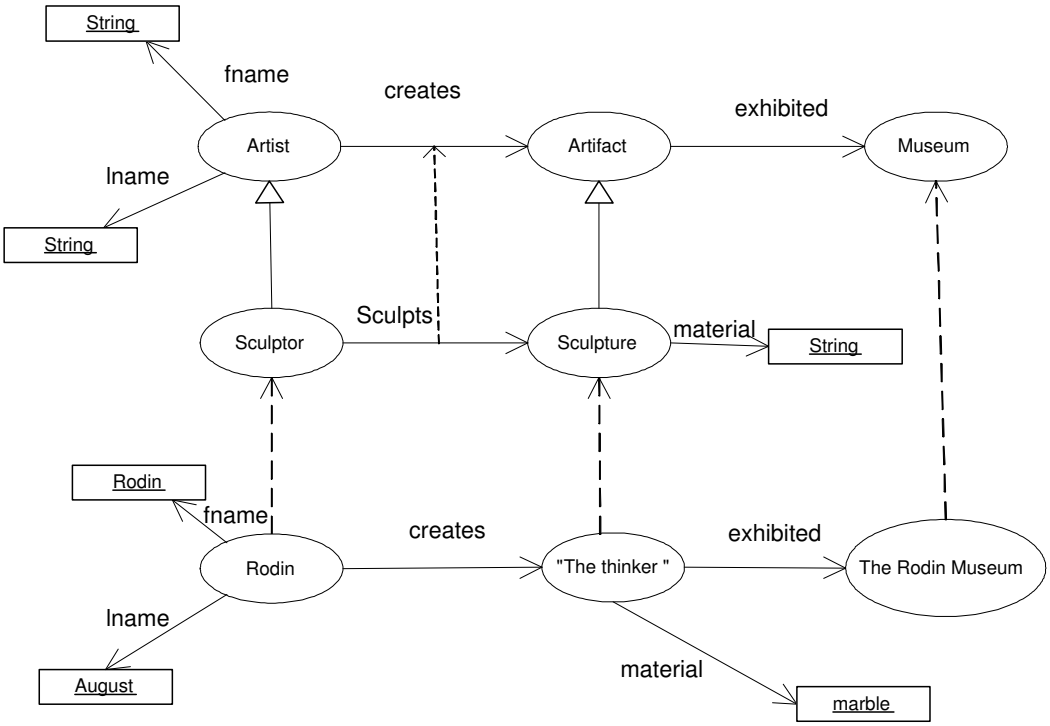
```
<rdfs:Class rdf:ID="Artist"/>
```

```
<rdfs:Class rdf:ID="Sculptor">
  <rdfs:subClassOf rdf:resource="#Artist"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Artifact"/>

<rdfs:Class rdf:ID="Sculpture">
  <rdfs:subClassOf rdf:resource="#Artifact"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Museum"/>

<rdf:Property rdf:ID="creates">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="#Artifact"/>
</rdf:Property>

<rdf:Property rdf:ID="sculpts">
  <rdfs:subPropertyOf rdf:resource="#creates"/>
</rdf:Property>

<rdf:Property rdf:ID="exhibited">
  <rdfs:domain rdf:resource="#Artifact"/>
  <rdfs:range rdf:resource="#Museum"/>
</rdf:Property>

<rdf:Property rdf:ID="fname">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource=http://www.w3.org/2001/XMLSchema#string>
</rdf:Property>

<rdf:Property rdf:ID="lname">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource=http://www.w3.org/2001/XMLSchema#string>
</rdf:Property>

<rdf:Property rdf:ID="material">
  <rdfs:domain rdf:resource="#Sculpture"/>
  <rdfs:range rdf:resource=http://www.w3.org/2001/XMLSchema#string>
</rdf:Property>
```

Στο παραπάνω XML κείμενο παρατηρούμε ότι όταν ορίζουμε μια κλάση ή ιδιότητα δηλώ-

νουμε το όνομά της με το rdf:ID. Όταν όμως θέλουμε να αναφερθούμε σ΄ αυτή χρησιμοποιούμε το rdf:resource. Επίσης όταν το πεδίο τιμών μιας ιδιότητας είναι κυριολεκτικό, ως τιμή στο rdfs:range βάζουμε τον τύπο του κυριολεκτικού.

Το RDFSchema όμως πρέχει και ένα τρόπο αναπαράστασης με γράφο, στον οποίο οι κλάσεις εμφανίζονται μέσα σε οβάλ σχήματα, τα κυριολεκτικά μέσα σε ορθογώνια και οι ιδιότητες αναπαριστώνται με ένα βέλος με φορά από την κλάση πεδίο ορισμού, στην κλάση πεδίο τιμών. Ο γράφος RDF του παραδείγματος που περιγράψαμε παραπάνω φαίνεται στο Σχήμα 2.3. Στο Σχήμα αυτό φαίνονται οι κλάσεις οι υποκλάσεις οι ιδιότητές τους.



Σχήμα 2.3: Γράφος για ένα απλό RDFSchema.

Στο Σχήμα 2.3 όμως, εκτός από τις κλάσεις υπάρχουν και τα στιγμιότυπά τους, δηλαδή τα δεδομένα. Για να περιγράψουμε τα στιγμιότυπα που φαίνονται στο σχήμα, θα γράφαμε σε RDF/XML:

```
<Sculptor rdf:about="#Rodin">
  <fname>Rodin</fname>
  <lname>August</lname>
  <creates>
    <Sculpture rdf:about="#The Thinker">
      <material>marble</material>
      <exhibited>
          <Museum rdf:about=#"The Rodin Museum" >
      </exhibited>
    </Sculpture>
```

```
  </creates>
</Sculptor>
```

Η δημιουργία του παραπάνω RDF/XML κειμένου γίνεται ως εξής: Ξεκινώντας από μια κλάση, γράφουμε τις ιδιότητες της και τα πεδία τιμών τους και συνεχίζουμε με τις κλάσεις - πεδία τιμών των ιδιοτήτων. Όταν μια ιδιότητα έχει σαν πεδίο τιμών ένα κυριολεκτικό, τότε απλά γράφουμε την τιμή του κυριολεκτικού αυτού. Επίσης όταν περιγράφουμε το στιγμιότυπο μιας κλάσης χρησιμοποιούμε το rdf:about για να ορίσουμε την πηγή (resource) του στιγμιοτύπου αυτού. Οι πηγές συνήθως είναι URIs, όμως εδώ για χάρη απλότητας του παραδείγματος τις έχουμε αναπαραστήσει με ένα απλό αλφαριθμητικό.

Στο σημείο αυτό, πρέπει να πούμε ότι παρόλο που το σύστημα τύπων του RDFSchema είναι παρόμοιο με αυτό των αντικειμενοστρεφών γλωσσών προγραμματισμού, υπάρχουν σημαντικές διαφορές σε αρκετά σημεία. Πρώτον το RDFSchema, αντί να περιγράφει τις κλάσεις σαν να έχουν κάποιες συγκεκριμένες ιδιότητες, περιγράφει τις ιδιότητες σαν να εφαρμόζονται σε συγκεκριμένες κλάσεις που αποτελούν πεδίο ορισμού ή πεδίο τιμών τους. Για παράδειγμα, αν είχαμε μια κλάση Book με ένα χαρακτηριστικό author και μια άλλη κλάση Software Module που επίσης είχε χαρακτηριστικό author, αυτά τα δύο χαρακτηριστικά θα θεωρούνταν διαφορετικά στις γλώσσες προγραμματισμού. Με άλλα λόγια το πεδίο δράσης ενός χαρακτηριστικού καθορίζεται μέσα από την κλάση που το ορίζει. Αντιθετα στο RDFSchema, οι ιδιότητες είναι ανεξάρτητες και έχουν καθολικό πεδίο δράσης, εκτός αν οριστούν να εφαρμόζονται σε συγκεκριμένες κλάσεις.

Επιπλέον οι ιδιότητες στο RDFSchema μπορούν να κληρονομηθούν σε αντίθεση με τον αντικειμενοστρεφή προγραμματισμό που κάτι τέτοιο δεν ισχύει.

Τέλος μια εξίσου σημαντική διαφορά είναι ότι στο RDFSchema τα στιγμιότυπα μιας ιδιότητας είναι προαιρετικά. Δηλαδή είναι δυνατό να υπάρχει στιγμιότυπο μιας κλάσης χωρις να υπάρχει στιγμιότυπο της ιδιότητας της οποίας η κλάση αποτελεί πεδίο ορισμού.

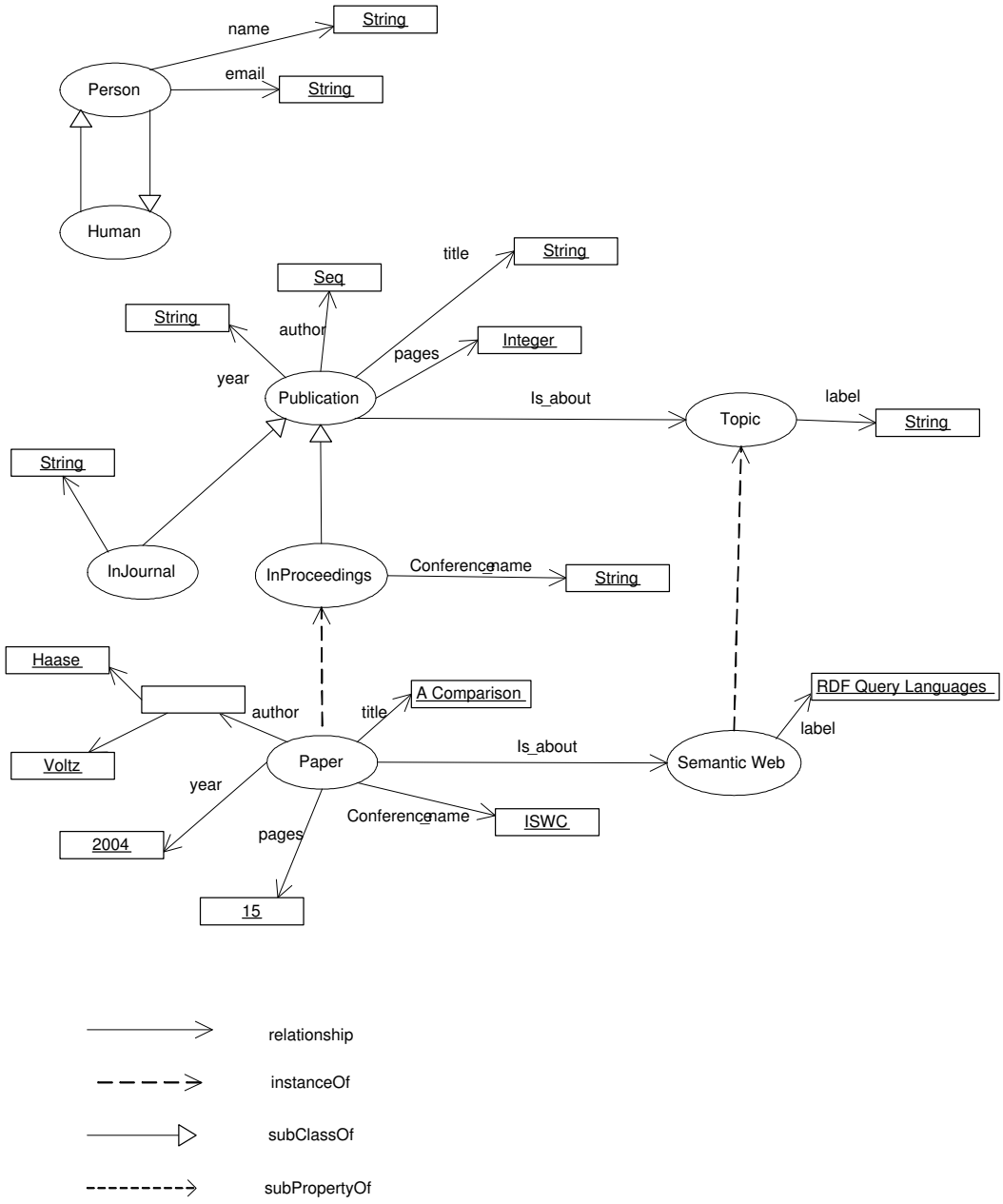
2.3 Γλώσσες ερωτήσεων για RDF

Στην παράγραφο αυτή εξετάζουμε έξι γλώσσες ερωτήσεων για το μοντέλο RDF, και παραθέτουμε παραδείγματα υλοποίησης συγκεκριμένων ερωτήσεων για καθεμία από αυτές. Τα παραδείγματα αυτά είναι ενδεικτικά και δεν καλύπτουν πλήρως τις δυνατότητες των γλωσσών.

Για τα παραδείγματα των ερωτήσεων θα χρησιμοποιήσουμε μια συγκεκριμένη RDF περιγραφή ο γράφος της οποίας φαίνεται στο Σχήμα 2.4. Στο σχήμα αυτό φαίνονται οι κλάσεις οι υποκλάσεις οι ιδιότητές τους και τα στιγμιότυπα αυτών.

Ο τύπος δεδομένων seq είναι ένας τύπος συλλογών ο οποίος περιέχει μια λίστα από διατεταγμένα στοιχεία. Στο συμβολισμό του στιγμιότυπου της κλάσης InProccedings έχουμε συμβολίσει το στιγμιότυπο του seq με ένα άδειο κουτί από το οποίο "κρέμονται" όλα τα στοιχεία.

Η σύγκριση των δυνατοτήτων των γλωσσών ερωτήσεων που έχουν προταθεί για να αξιοποιούν πληροφορίες που έχουν αποτυπωθεί με βάση το πλαίσιο RDF , θα γίνει εξετάζοντας τα ακόλουθα χαρακτηριστικά [7] :



Σχήμα 2.4: Βοηθητικός γράφος RDF για τη διατύπωση παραδειγμάτων ερωτήσεων.

- Εκφραστικότητα (Expressiveness) : Η εκφραστικότητα καθορίζει τη δυνατότητα των γλωσσών να μπορούν να εκφράζουν διάφορους τύπους ερωτήσεων. Τυπικά μια γλώσσα πρέπει τουλάχιστον να παρέχει τα μέσα που προσφέρει η σχεσιακή άλγεβρα. Συνήθως η εκφραστικότητα περιορίζεται από την ανάγκη αποδοτικής εκτέλεσης των ερωτήσεων και διατήρησης άλλων ιδιοτήτων όπως η ασφάλεια.

- Κλειστότητα (Closure) : Η κλειστότητα απαιτεί τα αποτελέσματα μιας λειτουργίας να είναι και πάλι στοιχεία του μοντέλου δεδομένων. Αυτό σημαίνει ότι αν μια γλώσσα ερωτήσεων λειτουργεί πάνω σε ένα μοντέλο δεδομένων με γράφους θα πρέπει τα

αποτελέσματα των ερωτήσεων να είναι και πάλι γράφοι.

- Αποτελεσματικότητα (Adequacy) : Μια γλώσσα ερωτήσεων θεωρείται αποτελεσματική αν χρησιμοποιεί όλα τα στοιχεία του μοντέλου δεδομένων. Αυτή η ιδιότητα συμπληρώνει την ιδιότητα της κλειστότητας.

- Ορθογωνιότητα (Orthogonality) : Η ορθογωνιότητα απαιτεί ότι όλες οι λειτουργίες μιας γλώσσας ερωτήσεων πρέπει να είναι ανεξάρτητες του περιεχομένου στο οποίο χρησιμοποιούνται.

- Ασφάλεια (Safety) : Μια γλώσσα ερωτήσεων θεωρείται ασφαλής αν κάθε ερώτηση η οποία είναι συντακτικά σωστή επιστρέφει ένα πεπερασμένο σύνολο από αποτελέσματα. Λειτουργίες οι οποίες μπορεί να προκαλέσουν έλλειψη ασφάλειας είναι η αναδρομή, η άρνηση και οι ενσωματωμένες συναρτήσεις.

2.3.1 RQL

Η RQL [14] είναι μία λειτουργική γλώσσα με τύπους, που στηρίζεται σ΄ ένα τυπικό μοντέλο με γράφους που είναι σύμφωνο με τις αρχές του μοντέλου RDF. Υποστηρίζει γενικευμένες εκφράσεις μονοπατιών (generalized path expressions) τόσο για τους κόμβους όσο και για τις ακμές του RDF γράφου. Η καινοτομία της RQL έγκειται στην ικανότητά της να κάνει ερωτήσεις τόσο σε RDF περιγραφές όσο και σε RDF σχήματα. Χρησιμοποιεί ένα συντακτικό που μοιάζει με αυτό της OQL.

Οι βασικές ερωτήσεις της RQL παρέχουν τη δυνατότητα πρόσβασης στις περιγραφές RDF και μπορούν να χρησιμοποιηθούν για τη δημιουργία απλών διαπροσωπειών για τη διάσχιση (browsing) των βάσεων που περιέχουν τις RDF περιγραφές.

Η RQL επεκτείνει τις γενικευμένες εκφράσεις μονοπατιών εισάγοντας μονοπάτια κληρονομικότητας κλάσεων, για τη διάσχιση (browsing) RDF σχημάτων. Έχει την ιδιότητα της ορθογωνιότητας αλλά όχι της κλειστότητας, αφού οι ερωτήσεις επιστρέφουν τιμές μεταβλητών αντί για γράφους. Επίσης η σημασιολογία (semantics) της RQL δεν είναι πλήρως συμβατή με τη σημασιολογία του RDF και γι΄ αυτό χρειάζονται κάποιοι περιορισμοί στα RDF μοντέλα για να μπορούν να γίνονται ερωτήσεις μέσω RQL.

Η RQL χρησιμοποιεί το φίλτρο Select-From-Where για να εξάγει το περιεχόμενο από RDF δεδομένα και σχήματα. Μετά το Select μπαίνουν οι μεταβλητές τις τιμές των οποίων θέλουμε να εμφανίσουμε και μετά το From μπαίνει η έκφραση μονοπατιού, η οποία είναι της μορφής {`Q`} `property` {`Y`} όπου Q και Y μεταβλητές κλάσεων. Μετά το where πάντα μπαίνει η έκφραση using namespace με την οποία ορίζουμε τα namespaces που χρησιμοποιούμε.

Οι ερωτήσεις που μπορούν να διατυπωθούν μέσω της RQL χωρίζονται σε ερωτήσεις σχήματος και ερωτήσεις δεδομένων. Οι ερωτήσεις σχήματος χρησιμεύουν για να εξάγουμε δομικές πληροφορίες του RDF σχήματος. Έστω για παράδειγμα η παρακάτω ερωτήση:

E1: Επέστρεψε την κλάση η οποία συνδέεται με την κλάση Publication με την ιδιότητα Is_about. Η ερώτηση αυτή υλοποιείται σε RQL ως εξής:

```
select $C from s:Publication.s:Is_about{$C}
using namespace
s = http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf
```

Το σύμβολο $ μπαίνει μπροστά από τη μεταβλητή για να δηλώσει ότι πρόκειται για μεταβλητή κλάσης. Η ερώτηση αυτή θα επέστρεφε την κλάση Topic. Εκτός όμως από κλάσεις που ικανοποιούν μια συγκεκριμένη έκφραση μονοπατιού, μπορούμε να αναζητούμε και ιδιότητες. Αυτό φαίνεται στο παρακάτω παράδειγμα:

E2: Επέστρεψε τις ιδιότητες που μπορούν να εφαρμοστούν στην κλάση InProceedings. Η ερώτηση αυτή διατυπώνεται ως εξής:

```
select @P from {s:InProceedings}@P
using namespace
s = http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf
```

Η ερώτηση αυτή λόγω κληρονομικότητας επιστρέφει τις ιδιότητες year, author, title, pages, Is_about, Conference_name.

Οι ερωτήσεις δεδομένων χρησιμεύουν για την ανάκτηση των στιγμιοτύπων των κλάσεων του σχήματος. Έστω για παράδειγμα η παρακάτω ερώτηση:

E3: Επέστρεψε τα ονόματα των συγγραφέων (author) της δημοσίευσης (publication) X. Η ερώτηση αυτή υλοποιείται σε RQL ως εξής :

```
select PersonName
from {X}s:author{Y}.rdfs:member{Z}.s:name{PersonName}
using namespace
s = http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf ,
rdfs = http://www.w3.org/2000/01/rdf-schema#
```

Επειδή το range της ιδιότητας author είναι τύπου seq χρησιμοποιούμε το rdfs:member για να διατρέξουμε τα στοιχεία του.

Η RQL υποστηρίζει τους βασικούς τελεστές συνόλων (ένωση, τομή, διαφορά) οι οποίοι εφαρμόζονται σε συλλογές του ίδιου τύπου. Έστω ότι θέλουμε να κάνουμε την εξής ερώτηση:

E4: Επέστρεψε τις επικεφαλίδες (label) όλων των θεμάτων (topic) και τους τίτλους (title) όλων των δημοσιεύσεων (publication). Η ερώτηση αυτή υλοποιείται σε RQL με χρήση του τελεστή ένωση (union) ως εξής :

```
(select TopicLabel
from s:Topic{X}.s:label{TopicLabel} )
union
( select PubTitle from s:Publication{X}.s:title{PubTitle} )
using namespace
```

```
s = http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf
```

Η παραπάνω ερώτηση θα επέστρεφε τη συλλογή (seq), RDF Query Languages, A Comparison. Έστω τέλος η ερώτηση:

Ε5: Επέστρεψε όλα τα στιγμιότυπα που είναι μέλη της κλάσης Publication. Η ερώτηση αυτή εκφράζεται σε RQL ως εξής:

```
select X from s:Publication{X}
using namespace
s = http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf
```

Στη συγκεκριμένη περίπτωση η ερώτηση αυτή θα επιστρέψει το Paper που είναι στιγμιότυπο της κλάσης InProccedings και άρα και της Publication.

Η RQL διαθέτει επίσης έτοιμες συναρτήσεις για κάποιες βασικές λειτουργίες. Μερικές από αυτές είναι: SuperClassOf και SubClassOf για την εύρεση της υπερκλάσης και της υποκλάσης αντίστοιχα μιας κλάσης, SuperPropertyOf και SubPropertyOf για την εύρεση της υπερ-ιδιότητας και της ύπο-ιδιότητας αντίστοιχα μιας κλάσης, domain και range για την εύρεση του πεδίου ορισμού και του πεδίου τιμών μιας ιδιότητας.

2.3.2 SeRQL

Η SeRQL [4] σχεδιάστηκε για να δώσει λύση σε πρακτικά προβλήματα των γλωσσών ερωτήσεων που ήδη υπήρχαν. Το συντακτικό της μοιάζει μ΄ αυτό της RQL, όμως έχουν γίνει διάφορες αλλαγές με σκοπό να γίνεται πιο εύκολα η συντακτική ανάλυση (parsing). Σε αντίθεση με την RQL η SeRQL είναι βασισμένη απευθείας στη θεωρία του μοντέλου RDF.

Η SeRQL υποστηρίζει γενικευμένες εκφράσεις μονοπατιών (generalized path expressions), λογικούς περιορισμούς, προαιρετικό ταίριασμα (optional matching), καθώς και δύο βασικά φίλτρα : select-from-where και construct-from-where. Το πρώτο επιστρέφει τις τιμές των μεταβλητών σε μορφή πίνακα ενώ το δεύτερο επιστρέφει τον υπογράφο που ταιριάζει. Έτσι οι construct-from-where ερωτήσεις πληρούν την κλειστότητα και την ορθογωνιότητα επιτρέποντας τη σύνθεση ερωτήσεων. Η SeRQL δεν είναι ασφαλής γιατί παρέχει ενσωματωμένες συναρτήσεις.

Μερικά χαρακτηριστικά γλωσσών ερωτήσεων λείπουν από την SeRQL. Τα πιο σημαντικά από αυτά είναι συναθροιστικές ερωτήσεις (minimum, maximum, average, count) και φώλιασμα ερωτήσεων.

Η SeRQL όπως και η RQL υποστηρίζει εκφράσεις μονοπατιών αλλά το συντακτικό της εμφανίζει κάποιες διαφορές σε σχέση με της RQL. Τα namespaces εμφανίζονται στη μορφή <namespace> και τα URIs στη μορφή <!URI>. Έτσι η ερώτηση Ε3 εκφράζεται σε SeRQL ως εξής :

```
select PersonName from {X}<s:author>{} <rdfs:member>{}<s:name>
{PersonName} using namespace
```

```
s = <!http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf>
```

Η SeRQL όμως υποστηρίζει και προαιρετικό ταίριασμα για τις εκφράσεις μονοπατιών. Επειδή το μοντέλο RDF είναι ημιδομημένο πρέπει οι γλώσσες ερωτήσεων να μπορούν να αντιμετωπίζουν περιπτώσεις όπου οι τιμές κάποιων μεταβλητών δεν υπάρχουν και αυτό γίνεται με το προαιρετικό ταίριασμα . Έστω για παράδειγμα η ερώτηση :

E6: Επέστρεψε το όνομα (name) και αν είναι γνωστό το e-mail των συγγραφέων (author) της δημοσίευσης (publication) X. Η ερώτηση αυτή υλοποιείται σε RQL ως εξής :

```
select PersonName, Email from {X}<s:author>{}<rdfs:member>{}<s:name>
{PersonName}; [<s:email> {Email}]
using namespace
s = <!http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf>
```

Η SeRQL λοιπόν υποστηρίζει το προαιρετικό ταίριασμα βάζοντας ανάμεσα σε [ ] τη μεταβλητή εκείνη της οποίας η τιμή δεν είναι απαραίτητο να βρεθεί.

Όπως είπαμε και παραπάνω η SeRQL υποστηρίζει και το φίλτρο construct-from-where το οποίο χρησιμοποιείται όπως και το select-from-where με τη διαφορά ότι επιστρέφει τα αποτελέσματα που ταιριάζουν με την έκφραση μονοπατιού σε μορφή γράφου. Έστω για παράδειγμα η ερώτηση :

E7: Επέστρεψε όλες τις κλάσεις με τις υποκλάσεις τους. Η ερώτηση αυτή υλοποιείται ως εξής :

```
Construct * from {Y}<rdfs:subClassOf>{Q}
using namespace
rdfs = <http://www.w3.org/2000/01/rdf-schema#>
```

Το αποτέλεσμα αυτής της ερώτησης θα είναι ένας γράφος που περιέχει όλες τις κλάσεις και υποκλάσεις του RDF σχήματος.

2.3.3 TRIPLE

Η TRIPLE [18] είναι μια διαστρωματωμένη γλώσσα κανόνων η οποία βασίζεται στη λογική Horn και δανείζεται κάποια βασικά χαρακτηριστικά της F-Logic. Οι τριπλέτες του RDF μοντέλου (υποκείμενο - ιδιότητα - αντικείμενο) αναπαριστώνται μέσω εκφράσεων της F-Logic οι οποίες μπορούν και να είναι φωλιασμένες. Επίσης η TRIPLE εκφράζει τόσο τους κανόνες όσο και τις ερωτήσεις με τον ίδιο τρόπο και υποστηρίζει και εκφράσεις μονοπατιών.

Η TRIPLE ονομάζεται διαστρωματωμένη γιατί υποστηρίζει δύο διαφορετικά στρώματα : Συντακτικές επεκτάσεις της λογικής Horn για την αναπαράσταση των πηγών και των εντολών RDF καθώς και στοιχεία για σημασιολογικές επεκτάσεις του RDF. Έτσι η TRIPLE δεν έχει συγκεκριμένη σημασιολογία για το μοντέλο RDF, αλλά η παραπάνω αρχιτεκτονική της

επιτρέπει να ορίζει εύκολα (σαν ένα σύνολο κανόνων) διάφορα αντικειμενοστρεφή και άλλα μοντέλα δεδομένων.

Η TRIPLE δεν έχει την ιδιότητα της κλειστότητας καθώς τα αποτελέσματα των ερωτήσεων έχουν τη μορφή πίνακα που περιέχει τιμές μεταβλητών. Επίσης δεν είναι ασφαλής καθώς επιτρέπει μη ασφαλείς κανόνες και δεν υποστηρίζει τύπους δεδομένων.

Μια τριπλέτα RDF γράφεται στην TRIPLE ως εξής υποκείμενο[ιδιότητα -> αντικείμενο] (subject[predicate->object]). Οι εκφράσεις της TRIPLE μπορεί να είναι κανόνες ή γεγονότα. Η TRIPLE πρέπει να ορίσει τη σημασιολογία του RDF σχήματος από την αρχή για να μπορέσει να κάνει ερωτήσεις πάνω σ΄ αυτό. Για να ορίσουμε ότι κάποιες εκφράσεις είναι αληθείς για ένα συγκεκριμένο μοντέλο γράφουμε @model{clauses}. Παρακάτω φαίνεται η υλοποίηση της ερώτησης E3, αφού πρώτα ορίζεται με αξιωματικό τρόπο η σημασιολογία του RDF σχήματος. Αρχικά ορίζονται τα namespaces:

```
rdfs :='http://www.w3.org/2000/01/rdf-schema#'.
rdf := 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
rdfs := 'http://www.w3.org/2000/01/rdf-schema#'.
default := 'http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf#'.
```

Στη συνέχεια ορίζεται η σημασιολογία του RDF σχήματος:

```
FORALL Mdl @rdfschema(Mdl) {
  FORALL O,P,V O[P->V] <- O[P->V]@Mdl.
  FORALL O,P,V O[P->V] <-
       EXISTS S (S[rdfs:subPropertyOf->P] AND O[S->V]).
  FORALL O,P,V O[rdfs:subClassOf->V] <-
       EXISTS W (O[rdfs:subClassOf->W] AND W[rdfs:subClassOf->V]).
  FORALL O,P,V O[rdfs:subPropertyOf->V] <-
       EXISTS W (O[rdfs:subPropertyOf->W] AND W[rdfs:subPropertyOf->V]).
  FORALL O,T O[rdf:type->T] <-
       EXISTS S (S[rdfs:subClassOf->T] AND O[rdf:type->S]). }
```

Τέλος διατυπώνεται η ερώτηση:

```
FORALL C,X,A,N <- (
    default:Paper[default:author->C] AND
    C[X->A] AND
    A[default:name->N]
)@rdfschema(default:ln).
```

Από τους κανόνες οι οποίοι βρίσκονται μέσα στο μπλοκ εντολών FORALL Mdl @rdfschema(Mdl) {...} προκύπτουν κάποια γεγονότα τα οποία έχουν ως αναγνωριστικό μοντέλου το rdfschema(Mdl). Η παράμετρος Mdl χρησιμοποιείται για το RDF Schema. Το μοντέλο rdfschema(Mdl) περιέχει όλες τις εντολές του μοντέλου Mdl και ότι προκύπτει επιπλέον από τους κανόνες. Ο κανόνας FORALL O,P,V O[P->V] <- O[P->V]@Mdl ορίζει

ότι κάθε τριπλέτα που περιέχεται στο μοντέλο Mdl είναι επίσης στοιχείο του μοντέλου rdf-schema(Mdl). Οι υπόλοιποι κανόνες ορίζουν κληρονομικότητα τιμών από υπό ιδιότητες σε υπέρ ιδιότητες καθώς και τις σημασιολογίες των subClassof, subPropertyof και type. Στη συνέχεια διατυπώνεται η ερώτηση.

Για την υλοποίηση της ένωσης (union) στην TRIPLE χρησιμοποιούμε ένα εικονικό predicate μεταξύ δύο μεταβλητών, το οποίο αληθεύει ταυτόχρονα για δύο διαφορετικά σύνολα δεδομένων. Έτσι το τελικό αποτέλεσμα είναι η ένωση των δύο συνόλων. Η ερώτηση E4 υλοποιείται ως εξής :

```
FORALL T,L T[default:result->L] <- (
    T[rdf:type->default:Topic] AND T[default:label->L]
)@rdfschema(default:ln).

FORALL T,L T[default:result->L] <- (
    T[rdf:type->default:Publication] AND T[default:title->L]
)@rdfschema(default:ln).

FORALL T,L <- (
    T[default:result->L]
)@rdfschema(default:ln).
```

2.3.4 RDQL

Η RDQL [11] σχεδιάστηκε για να εξάγει πληροφορία από τους γράφους RDF δεν μπορεί όμως να μεταφράσει πληροφορία RDF Schema. Το συντακτικό της μοιάζει με αυτό της SQL (Select-from-where) και υποστηρίζει συντομογραφίες για namespaces μέσω της εντολής υσινγ. Τα αποτελέσματα των ερωτήσεων έχουν τη μορφή πίνακα μεταβλητών και έτσι η RDQL δεν έχει την ιδιότητα της ορθογωνιότητας. Η RDQL είναι ασφαλής και υποστηρίζει κάποιους πρωταρχικούς τύπους δεδομένων.

Μια ερώτηση RDQL στο τμήμα μετά το ωηερε αποτελείται από μοτίβα (patterns) τριπλετών. Κάθε μοτίβο τριπλέτας αποτελείται από μεταβλητές και RDF τιμές. Επίσης μετά τους ορισμούς των μοτίβων μπορούν να εισαγωνται και περιορισμοί για τις μεταβλητές που συμμετέχουν σε αυτά. Οι μεταβλητές ορίζονται με ?ονομα_μεταβλητής και τα URI references <URIref>. Η ερώτηση E3 υλοποιείται σε RDQL ως εξής :

```
select ?n
where
(<file:///C:/papers/2004/vldb04/queries/sample.rdf#Paper>,
<s:author>, ?c),
    (?c, ?collection, ?a),
    (?a, <s:name>, ?n)
```

```
USING
    s FOR
    <http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf#>
```

Η RDQL δεν παρέχει τρόπο πρόσβασης στα στοιχεία ενός αντικειμένου που είναι τύπου seq. Επομένως για να μπορέσουμε να πάρουμε όλα τα μέλη του sequence που αποτελεί αντικείμενο της ιδιότητας author, χρησιμοποιούμε ένα ενδιάμεσο εικονικό predicate το οποίο συνδέει το αντικείμενο του αυτηορ με κάθε μέλος του. Οι ερωτήσεις E4 και E6 δεν μπορούν να υλοποιηθούν στην RDQL.

2.3.5 N3

Η Notation3(N3) [10] έχει συντακτικό που μοιάζει με κείμενο. Το μοντέλο δεδομένων της είναι σύμφωνο με αυτό του RDF. Επίσης έχει τη δυνατότητα να ορίζει κανόνες οι οποίοι χρησιμοποιούνται και για ερωτήσεις. Με την εντολή CWM δίνει τη δυνατότητα αυτόματης επιλογής των δεδομένων που παράγονται από τους κανόνες. Η N3 πληρεί τις ιδιότητες της ορθογωνιότητας και της κλειστότητας. Δεν παρέχει σημασιολογία για το μοντέλο RDF και γι΄ αυτό η σημασιολογία πρέπει να ορίζεται από συγκεκριμένους κανόνες.

Στην N3 τα namespaces ορίζονται ως εξής @prefix abbreviation: <URIref>. Το κενό namespace @prefix: <#> αποτελεί συντομογραφία για το δοσυμεντ στο οποίο γράφουμε. Η N3 λειτουργεί με κανόνες οι οποίοι βρίσκονται ανάμεσα σε { } και μπορεί να συνδέονται με συνεπαγωγή η οποία συμβολίζεται με =>. Ένας κανόνας μπορεί να αποτελείται από πολλές εκφράσεις οι οποίες διαχωρίζονται με . και για να είναι αληθής ένας κανόνας πρέπει να είναι αληθείς όλες οι εκφράσεις που τον αποτελούν. Οι μεταβλητές ορίζονται με ?ονομα_μεταβλητής. Το a στην N3 το βάζουμε όταν θέλουμε να δηλώσουμε τι τύπου είναι μια μεταβλητή. Έχει δηλαδή την ίδια λειτουργία με το rdf:type. Επίσης δύο εκφράσεις που αναφέρονται στο ίδιο υποκείμενο χωρίζονται με ;. Όταν υπάρχει η έκφραση :x :y σημαίνει ότι το y αντικαθίσταται από το x χωρίς να αλλάζει το αποτέλεσμα. Έτσι η ερώτηση E3 εκφράζεται στην N3 ως εξής:

```
@prefix :  <#> .
@prefix XML: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22rdfsyntaxns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sam:
<http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf#> .

{ ?y sam:author ?z.
?z ?p ?x.
?x a sam:Person; sam:name ?result.
} => { ?result a :QueryResult.}.
```

Και η N3 δεν παρέχει τρόπο πρόσβασης στα στοιχεία ενός αντικειμένου που είναι τύπου

seq. Επομένως για να μπορέσουμε να πάρουμε όλα τα μέλη του sequence που αποτελεί αντικείμενο της ιδιότητας author, χρησιμοποιούμε ένα ενδιάμεσο εικονικό predicate όπως και με την RDQL. Έστω η ερώτηση:

E8 : Μέτρησε τον αριθμό των συγγραφέων μιας δημοσίευσης. Η ερώτηση αυτή υλοποιείται σε N3 ως εξής:

```
@prefix :  <#>.
@prefix XML: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
.  @prefix sam:<http://www.aifb.uni-karlsruhe.de/WBS/pha/
rdf-query/sample.rdf#>.
@prefix math:  <http://www.w3.org/2000/10/swap/math#> .

{?y.sam:author math:memberCount ?result .} => {:Query :Result
?result}.
```

Για την υλοποίηση της μέτρησης χρησιμοποιήθηκε το predicate math:memberCount.

2.3.6 Versa

Η Versa [9] ορίζει εκφράσεις των οποίων δομικό στοιχείο είναι η λίστα από RDF πηγές. Υπάρχουν δύο ειδών εκφράσεις. Οι εκφράσεις που χρησιμοποιούνται για τη διάσχιση RDF γράφων και έχουν τη μορφή Λίστα - Λίστα -> Λογική έκφραση, και οι εκφράσεισ-φίλτρα που έχουν τη μορφή Λίστα ¸ - Λίστα -> Λογική έκφραση. Το παραπάνω συντακτικό της Versa επιτρέπει τη δημιουργία φωλιασμένων ερωτήσεων. Η λίστα (list) είναι ένα διατεταγμένο σύνολο από τύπους δεδομένων. Η λογική έκφραση μπορεί να είναι true ή false. Το * χρησιμοποιείται για να συμβολίσει το true.

Οι ερωτήσεις της Versa λειτουργούν στο μοντέλο δεδομένων RDF. Η Versa παρέχει και κάποια υποστήριξη για κανόνες δεν παρέχει όμως ενσωματωμένες συναρτήσεις, όψεις και διαχείριση δεδομένων. Όμως πληρεί τις ιδιότητες της ορθογωνιότητας και της ασφάλειας. Η ερώτηση E3 εκφράζεται στη Versa ως εξής :

```
QUERY=((@"../versa-sample.rdf#Paper" - s:author ->
*) - properties(.)  -> *) - s:name -> *
```

Η ερώτηση αυτή για να υλοποιηθεί χρησιμοποιεί φωλιασμένες εκφράσεις. Η πρώτη έκφραση γίνεται τρυε για όλους τους συγγραφείς (authors) του Paper, η δεύτερη για όλες τις ιδιότητες (προπερτιες) κάθε αυτηορ και η τελευταία για την ιδιότητα name κάθε συνόλου ιδιοτήτων. Η Versa υποστηρίζει και προαιρετικό ταίριασμα. Επομένως η ερώτηση E4 υλοποιείται ως εξής:

```
QUERY=distribute(((@"../versa-sample.rdf#Paper" - s:author -> *) -
properties(.)  -> *), ".- s:name -> *", ".- s:email -> *")
```

Ο τελεστής distribute δημιουργεί μια λίστα από λίστες. Μέσα σ΄ αυτές τις λίστες το e-mail μπορεί να υπάρχει ή να μην υπάρχει. Η ερώτηση E4 υλοποιείται στη Versa ως εξής:

```
QUERY=union((all() - s:title -> *), (all() - s:label -> *))
```

Η Versa διαθέτει ειδικό τελεστή union για την εκτέλεση της ένωσης. Ο τελεστής αυτός παίρνει δύο ορίσματα. Το ένα γίνεται τρυε για όλα τα υποκείμενα που έχουν ιδιότητα title και το δεύτερο για όλα τα υποκείμενα που έχουν ιδιότητα label.

Κεφάλαιο 3

Περιγραφή θέματος

Στο κεφάλαιο αυτό αρχικά γίνεται μια περιγραφή των συστημάτων ομότιμων κόμβων που είναι βασισμένα σε σχήματα (schema-based peer-to-peer systems). Στη συνέχεια περιγράφονται τρία βασικά συστήματα που ανήκουν σε αυτή την κατηγορία, καθώς και ένα σύστημα για τη διαχείριση RDF σχημάτων, και τέλος αναλύεται ο στόχος της παρούσας εργασίας.

3.1 Σχετικές εργασίες

Οι βάσεις δεδομένων εισήγαγαν ένα τρόπο αποθήκευσης και ανάκτησης των δεδομένων που βασιζόταν στο σχήμα [16]. Τα πρώτα συστήματα ομότιμων κόμβων που περιγράψαμε στην Υποενότητα 2.1.2 έδιναν μεγάλη σημασία στην αρχιτεκτονική του συστήματος και την δρομολόγηση των ερωτήσεων και λιγότερη στον τρόπο αναπαράστασης και τις δυνατότητες αναζήτησης. Η αναζήτηση σε αυτά τα συστήματα ομότιμων κόμβων γίνεται με βάση προκαθορισμένα χαρακτηριστικά - δείκτες, ή με προσπάθεια αντιστοίχισης μιας λέξης κλειδί.

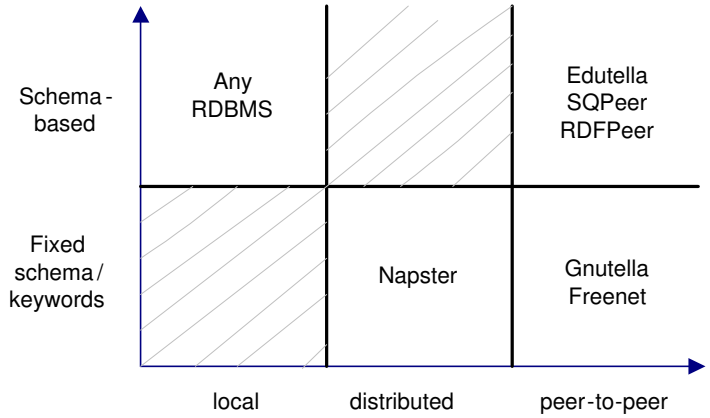
Η ανάγκη λοιπόν για πιο εκφραστικές λειτουργίες οδήγησε στα συστήματα ομότιμων κόμβων τα οποία είναι βασισμένα σε σχήματα (schema based peer-to-peer systems). Πρόκειται για ομότιμες υποδομές διαχείρισης δεδομένων που όμως διατηρούν όλα τα χαρακτηριστικά των συστημάτων ομότιμων κόμβων. Στο Σχήμα 3.1 φαίνεται μια τοποθέτηση των συστημάτων ομότιμων κόμβων που περιγράψαμε στην Υποενότητα 2.1.2 καθώς και αυτών που θα περιγράψουμε στην επόμενη υποενότητα, σε σχέση με τον τρόπο αναζήτησης και το είδος της κατανομής που τα χαρακτηρίζει.

Ένα βασικό και ενδιαφέρον ζήτημα στα συστήματα ομότιμων κόμβων που είναι βασισμένα σε σχήματα είναι το πώς θα μπορέσουν οι κόμβοι που διαθέτουν διαφορετικά σχήματα να συνεργαστούν μεταξύ τους διατηρώντας όμως την αυτονομία τους.

Τα βασικά δομικά στοιχεία που πρέπει να έχει ένα σύστημα ομότιμων κόμβων βασισμένο σε σχήματα είναι:

- Μια γλώσσα σχήματος για τον ορισμό και τη χρήση διάφορων σχημάτων που καθορίζουν το είδος των δεδομένων που είναι διαθέσιμα στο σύστημα ομότιμων κόμβων.

- Μια γλώσσα ερωτήσεων για την ανάκτηση των δεδομένων που βρίσκονται αποθηκευμέ-



Σχήμα 3.1: Αναζήτηση και κατανομή

να στο σύστημα ομότιμων κόμβων.

- Μια τοπολογία δικτύου συνδυασμένη με ένα κατάλληλο αλγόριθμο δρομολόγησης ερωτήσεων, που θα επιτρέπει αποδοτικές ερωτήσεις.

- Δυνατότητες ενσωμάτωσης ετερογενούς πληροφορίας που βρίσκεται αποθηκευμένη στο σύστημα ομότιμων κόμβων.

3.1.1 Edutella

Το σύστημα Edutella [17] δημιουργήθηκε για να χρησιμοποιηθεί για την ανταλλαγή εκπαιδευτικού υλικού μεταξύ των μελών των πανεπιστημίων. Αυτό είναι ένα πεδίο εφαρμογής, στο οποίο είναι απαραίτητο να υπάρχει μεγάλη εκφραστικότητα στις ερωτήσεις που θα βασίζεται σε πλούσια μεταδεδομένα.

Το Edutella χρησιμοποιεί το πλαίσιο RDF για την αναπαράσταση των μεταδεδομένων, καθώς προσφέρει τη δυνατότητα κατανεμημένης εκχώρησης μεταδεδομένων, (δηλαδή κάθε κόμβος μπορεί να έχει διαφορετικά μεταδεδομένα για την ίδια πηγή πληροφορίας) κάτι που το κάνει κατάλληλο για τη δημιουργία κατανεμημένων αποθηκών. Επίσης το Edutella βασίζεται στην τεχνολογία JXTA για την ανάπτυξη και τη διαχείριση του δικτύου ομότιμων κόμβων.

Η τοπολογία των κόμβων του Edutella ακολουθεί την τοπολογία των Super-Peer δικτύων. Σύμφωνα με την τοπολογία αυτή όλοι οι κόμβοι δεν είναι ίσοι. Υπάρχει ένα μικρό υποσύνολο των κόμβων, οι super κόμβοι, οι οποίοι αναλαμβάνουν τη δρομολόγηση των ερωτήσεων. Έτσι στα συστήματα ομότιμων κόμβων που χρησιμοποιούν αυτή την τοπολογία οι ερωτήσεις δρομολογούνται πρώτα στους super κόμβους και στη συνέχεια κατανέμονται στους κόμβους που είναι συνδεδεμένοι με αυτούς. Στο Edutella οι super κόμβοι διαθέτουν δείκτες δρομολόγησης, που γνωρίζουν τη σημασιολογική ετερογένεια των σχημάτων του συστήματος και έχουν επομένως πληροφορία σχήματος. Η κατανομή των κόμβων στο δίκτυο του Edutella ακολουθεί τον αλγόριθμο HyperCup.

Το σύστημα Edutella υλοποιεί έναν αριθμό υπηρεσιών, που έχουν σαν στόχο να κάνουν την κατανεμημένη φύση των μεμονομένων κόμβων να φαίνεται διάφανη. Κάθε κόμβος χαρακτηρίζεται από το σύνολο των υπηρεσιών που προσφέρει. Οι βασικές υπηρεσίες λοιπόν του Edutella είναι:

- Υπηρεσία ερωτημάτων (Query Service). Η υπηρεσία ερωτημάτων του Edutella είναι η πιο βασική υπηρεσία. Οι κόμβοι δηλώνουν τα ερωτήματα τα οποία μπορούν να ερωτηθούν από την υπηρεσία καθορίζοντας συγκεκριμένα σχήματα μεταδεδομένων που υποστηρίζουν, ή συγκεκριμένες ιδιότητες ή τιμές αυτών των ιδιοτήτων. Τα ερωτήματα στέλνονται μέσω του δικτύου του Edutella, στο υποσύνολο των κόμβων που έχουν δηλώσει ότι ενδιαφέρονται για το συγκεκριμένο ερώτημα. Οι απαντήσεις στέλνονται πίσω στον κόμβο που έκανε την ερώτηση.

- Υπηρεσία αναπαραγωγής (Replication Service). Αυτή η υπηρεσία συμπληρώνει την τοπική αποθήκευση των δεδομένων, δημιουργώντας αντίγραφα αυτών σε επιπρόσθετους κόμβους, με σκοπό την σταθερότητα και διαθεσιμότητα των δεδομένων, καθώς και την ισορροπία του φορτίου, ενώ εξασφαλίζει την ακεραιότητα και τη συνέπεια των δεδομένων.

- Υπηρεσία αντιστοιχίσεων (Mapping Service). Παρόλο που μπορεί μια ομάδα κόμβων να συμφωνήσει στη χρήση ενός κοινού σχήματος, σε κάποιες περιπτώσεις μπορεί να χρειάζονται επεκτάσεις. Η υπηρεσία αντιστοιχίσεων του Edutella, είναι ικανή να διαχειρίζεται αντιστοιχίσεις μεταξύ διαφορετικών σχημάτων και να χρησιμοποιεί αυτές τις αντιστοιχίσεις για να μεταφράζει ερωτήσεις από το ένα σχήμα στο άλλο. Η υπηρεσία αυτή παρέχει επίσης διαλειτουργικότητα μεταξύ RDF και XML αποθηκών.

- Υπηρεσία διαμεσολάβησης (Mediation Service). Η υπηρεσία αυτή μεσολαβεί ώστε να υπάρχει πρόσβαση ανάμεσα στις διάφορες υπηρεσίες. Ορίζει όψεις που συγχωνεύουν τα δεδομένα που προέρχονται από διαφορετικές πηγές μεταδεδομένων και απομακρύνουν την περιττή και μη συμβατή πληροφορία.

- Υπηρεσία επισήμανσης (Annotation Service). Η υπηρεσία αυτή είναι υπεύθυνη για την επισήμανση του υλικού που βρίσκεται αποθηκευμένο οπουδήποτε μέσα στο δίκτυο του Edutella.

Η υπηρεσία ερωτήσεων του Edutella χρησιμοποιεί το πρωτόκολλο ερωτήσεων RDF-QEL. Το πρωτόκολλο παρέχει μια γλώσσα για ανταλλαγή ερωτήσεων που λειτουργεί σαν μια συμβατική φόρμα ανταλλαγής ερωτήσεων, στην οποία μεταφράζονται οι τοπικές γλώσσες ερωτήσεων. Οι κόμβοι του Edutella συνδέονται στο σύστημα χρησιμοποιώντας μια αρχιτεκτονική περιτυλίγματος (wrapper-based), όπου το σύστημα - wrapper είναι υπεύθυνο, για τη μετάφραση των τοπικών γλωσσών ερωτήσεων στο σύνηθες μοντέλο δεδομένων του Edutella (ECDM). Οι δομές τριπλετών και κυρίως οι παραμετροποιημένες όψεις που χρησιμοποιούνται, είναι πολύ χρήσιμες για την παροχή υπηρεσιών μετατροπή και διαμεσολάβησης.

Η υπηρεσία διαμεσολάβησης χρησιμοποιεί μια προσέγγιση δύο επιπέδων. Υπάρχουν οι απλοί κόμβοι διαμεσολαβητές οι οποίοι κατανέμουν τα ερωτήματα στον κατάλληλο κόμβο, με τον περιορισμό ότι τα ερωτήματα μπορούν ν΄ απαντηθούν εντελώς από έναν Edutella κόμβο. Σε δεύτερο επίπεδο υπάρχουν, οι διαμεσολαβητές οι οποίοι είναι υπεύθυνη για την απάντηση τελικά του ερωτήματος.

3.1.2 SQPeer

Το σύστημα SQPeer [15] είναι ένα σύστημα για τη δρομολόγηση και την επεξεργασία ερωτήσεων σε γλώσσα RQL και όψεων RVL (γώσσα δημιουργίας όψεων για RDF), που παίρνει υπόψη του την κατανομή των δεδομένων στις βάσεις των κόμβων ενός συστήματος ομότιμων κόμβων.

Το SQPeer χρησιμοποιεί την τεχνολογία των Semantic Overlay Networks ή SONs. Τα SONs είναι ένας τρόπος ομαδοποίησης κόμβων που μοιράζονται το ίδιο σχήμα. Έτσι λοιπόν κόμβοι που χρησιμοποιούν ένα ή περισσότερα θέματα από το σχήμα μιας κοινότητας θεωρούνται ότι ανήκουν στο ίδιο SON. Αυτή η προσέγγιση διευκολύνει τη δρομολόγηση των ερωτημάτων, αφού κάθε κόμβος είναι εύκολο να ταυτοποιήσει τους παρόμοιους κόμβους αντί να διαδίδει τα ερωτήματα σε όλους τους κόμβους του δικτύου.

Τα ερωτήματα στο SQPeer δημιουργούνται από τους κόμβους σε γλώσσα RQL σύμφωνα με το RDF σχήμα της κοινότητας στην οποία ανήκουν. Αφού οι κόμβοι ενός SON μπορούν να χρησιμοποιήσουν ένα ή περισσότερα RDF σχήματα κοινοτήτων χρειάζεται ένας τρόπος ώστε να μπορούμε να βγάζουμε συμπέρασμα για την ομοιότητα των ερωτημάτων και των περιεχομένων των βάσεων των κόμβων. Το SQPeer βασίζεται στα πρότυπα ερωτημάτων (query patterns) για την εξαγωγή της πληροφορίας σχήματος που βρίσκεται μέσα σε ένα RQL ερώτημα. Τα πρότυπα ερωτημάτων εξάγονται από τις εκφράσεις μονοπατιών που εμφανίζονται σε ένα RQL ερώτημα.

Επειδή τα RDF σχήματα μιας κοινότητας μπορεί να έχουν συσχετισθεί με τα δεδομένα της βάσης του κόμβου, μπορεί όμως και όχι, το SQPeer χρησιμοποιεί την ιδέα των ενεργών σχημάτων για τη δημιουργία διαφημίσεων των βάσεων των κόμβων. Ένα ενεργό σχήμα είναι ένα υποσύνολο των RDF σχημάτων μιας κοινότητας, για το οποίο όλες οι κλάσεις έχουν συσχετισθεί με δεδομένα στη βάση του κόμβου. Τα ενεργά σχήματα διαδίδονται σε όλους τους κόμβους του συστήματος ομότιμων κόμβων.

Οι τεχνικές της εξαγωγής προτύπων ερωτημάτων και των ενεργών σχημάτων δίνουν τη δυνατότητα μετατροπής στα ερωτήματα που στέλνονται σε κόμβους με διαφορετικό σχήμα και επίσης μειώνουν το φορτίο των ερωτημάτων που επεξεργάζεται κάθε κόμβος, αφού φτάνουν σάυτόν μόνο σχετικά με τη βάση του ερωτήματα.

Το SQPeer χρησιμοποιεί έναν αλγόριθμο δρομολόγησης για τα ερωτήματα, ο οποίος παίρνει σαν είσοδο ένα πρότυπο ερωτήματος και το επεκτείνει με επισημάνσεις για το ποιος κόμβος μπορεί να το απαντήσει. Στη συνέχεια χρησιμοποιούνται τεχνικές που αποφαίνονται πιο κομμάτι της ερώτησης μπορεί να απαντηθεί από ένα ενεργό σχήμα και μετασχηματίζουν κατάλληλα το ερώτημα που στέλνεται σε ένα κόμβο.

Η επεξεργασία ερωτήσεων στο SQPeer είναι υπεύθυνη για την παραγωγή κατανεμημένων πλάνων ερωτήσεων, ανάλογα με το αποτέλεσμα του αλγόριθμου δρομολόγησης. Για την υλοποίηση αυτών των πλάνων και την ανταλλαγή δεδομένων μεταξύ των εμπλεκόμενων κόμβων το SQPeer βασίζεται σε κατάλληλα κανάλια επικοινωνίας.

Στο SQPeer υπάρχουν δύο πιθανοί τρόποι βελτιστοποίησης των κατανεμημένων πλάνων ερωτημάτων. Ο πρώτος βασίζεται σε ισότητα αλγεβρικών εκφράσεων (όπως κατανεμημένων συνενώσεων και ενώσεων) δίνοντας τη δυνατότητα να σπρώξουμε όσο το δυνατόν περισσότερη αποτίμηση ερωτήσεων στον ίδιο κόμβο και ο δεύτερος βασίζεται σε στατιστικές που γίνονται σε κάθε κόμβο και δίνουν τη δυνατότητα επιλογής μεταξύ διαφορετικών εκτελέσεων του ίδιου πλάνου.

Τέλος είναι αξιοσημείωτο το γεγονός ότι το SQPeer μπορεί να χρησιμοποιηθεί σε διάφορες αρχιτεκτονικές συστημάτων ομότιμων κόμβων, γιατί οι αλγόριθμοι δρομολόγησης και επεξεργασίας ερωτήσεων, δεν επηρεάζονται από το είδος της αρχιτεκτονικής.

3.1.3 RDFPeers

Το σύστημα RDFPeers [5] είναι ένα κατανεμημένο σύστημα, για τη δεικτοδότηση, αποθήκευση και ερώτηση, μεμονωμένων RDF τριπλετών και το οποίο δεν απαιτεί τον ορισμό ενός RDF σχήματος πριν την εισαγωγή των τριπλετών στο σύστημα.

Οι παροχείς της αποθήκευσης των RDF τριπλετών, αυτοοργανώνονται σ'ενα συνεργάσιμο και δομημένο δίκτυο ομότιμων κόμβων, που βασίζεται στην τυχαία επιλογή αναγνωριστικών κόμβων. Όταν μια τριπλέτα RDF εισάγεται στο σύστημα, αποθηκεύεται τρεις φορές, ανάλογα με το αποτέλεσμα μιας συνάρτησης κατακερματισμού που εφαρμόζεται στο υποκείμενο την ιδιότητα και το αντικείμενό της. Έτσι τα ερωτήματα μπορούν να δρομολογηθούν αποτελεσματκά σε αυτούς τους κόμβους που ξέρουμε ότι έχουν αποθηκευτεί οι τριπλέτες της ερώτησης.

Το δίκτυο στο οποίο αυτοοργανώνονται οι κόμβοι ονομάζεται MAAN (Multi-Attribute Addressable Network). Το σύστημα RDFPeers εκμεταλλεύεται το MAAN ως δικτυακή υποδομή και το επεκτείνει με εξειδικευμένες τεχνικές για αποθήκευση, ανάκτηση και εξισορρόπηση φορτίου. Έτσι λοιπόν κάθε κόμβος στο RDFPeers αποτελείται από πέντε συστατικά: τη δικτυακή υποδομή MAAN, το σύστημα που φορτώνει τις τριπλέτες RDF, το τοπικό σύστημα αποθήκευσης των τριπλετών, το εγγενές σύστημα αποτίμησης της ερώτησης και το σύστημα μετατροπής της RDQL στο εγγενές μοντέλο.

Το πρωτόκολλο του MAAN περιλαμβάνει τρεις κλάσεις μηνυμάτων για (α) συντήρηση της τοπολογίας, (β) αποθήκευση και (γ) αναζήτηση. (α) Η συντήρηση της τοπολογίας περιλαμβάνει μηνύματα που χρησιμοποιούνται για να διατηρούνται οι σωστές συνδέσεις με τους γείτονες και οι σωστοί πίνακες δρομολόγησης. (β) Το μήνυμα για την αποθήκευση εισάγει τις τριπλέτες στο σύστημα. (γ) Το μήνυμα για την αναζήτηση επισκέπτεται τους κόμβους όπου είναι γνωστό ότι οι τριπλέτες της ερώτησης είναι αποθηκευμένες, και επιστρέφει τις τριπλέτες που αντιστοιχούν, στον κόμβο που έκανε την ερώτηση.

Το σύστημα που φορτώνει τις τριπλέτες διαβάζει το αρχείο RDF το μετατρέπει σε RDF τριπλέτες και στη συνέχεια χρησιμοποιεί το μήνυμα αποθήκευσης για να αποθηκεύσει τις

τριπλέτες στο σύστημα. Όταν ένας κόμβος του RDFPeers λαμβάνει ένα μήνυμα αποθήκευσης, αποθηκεύει τις τριπλέτες σε μια τοπική σχεσιακή βάση δεδομένων. Το εγγενές σύστημα αποτίμησης ερωτήσεων, διαβάζει τις ερωτήσεις και χρησιμοποιεί το μήνυμα αναζήτησης για την απάντησή τους. Πάνω από το εγγενές σύστημα αποτίμησης ερωτήσεων μπορεί να υπάρξει οποιοδήποτε υψηλότερου επιπέδου σύστημα ερωτήσεων, όπως ο μετατροπέας της RDQL ερώτησης στο εγγενές μοντέλο.

3.1.4 RDFSculpt

Το RDFSculpt [13] είναι ένα σύστημα διαχείρισης RDF σχημάτων. Δεν αποτελεί σύστημα ομότιμων κόμβων όπως τα προηγούμενα που περιγράφηκαν, όμως οι λειτουργίες του αποδεικνύονται πολύ χρήσιμες για τα συστήματα ομότιμων κόμβων που είναι βασισμένα σε σχήμα.

Το RDFSculpt έχει τη δυνατότητα να επιτελεί τις εξής βασικές πράξεις (όπως αυτές ορίζονται από τη συνολοθεωρία) ανάμεσα σε δύο RDF σχήματα: επιλογή, ένωση, τομή και διαφορά. Όλα όμως τα σχήματα που συμμετέχουν στις πράξεις έχουν την ιδιότητα ότι έχουν προέλθει (πάλι χρησιμοποιώντας τις βασικές αυτές πράξεις) από ένα αρχικό καθολικό σχήμα και επομένως είναι συμβατά μεταξύ τους, δηλαδή αποτελούν υποσύνολα αυτού του σχήματος.

Το RDFSculpt υποστηρίζεται από μια γραφική διαπροσωπεία χρήστη, η οποία βοηθάει το χρήστη να επιτελέσει τις πράξεις αυτές, επιλέγοντας κλάσεις και κυριολεκτικά από κάθε σχήμα που συμμετέχει στην πράξη. Το τελικό σχήμα - αποτέλεσμα μιας πράξης έχει τη δυνατότητα αποθήκευσης τόσο ως αρχείο RDF/XML όσο και σε σχεσιακή βάση δεδομένων με τη βοήθεια του εργαλείου RSSDB. Επίσης υπάρχουν λειτουργίες προεπισκόπισης των σχημάτων και αλλαγής του καθολικού σχήματος. Το RDFSculpt χρησιμοποιεί τη γλώσσα RQL για να κάνει ερωτήσεις στα RDF σχήματα, οι οποίες βοηθούν στην κατασκευή του αποτελέσματος της πράξης.

3.1.5 Piazza

ο σύστημα Piazza [8], προσφέρει μια υποδομή διαχείρισης δεδομένων για το Σημασιολογικό Ιστό. Αποτελείται από πολλούς κόμβους καθένας από τους οποίους συμμετέχει στο σύστημα προσφέροντας δεδομένα ή παρέχοντας σχήμα ή και τα δύο. Η σημασιολογική αντιστοίχιση σ΄ αυτό το σύστημα γίνεται εφικτή με τη χρήση κανόνων αντιστοίχισης (mapping rules) μεταξύ ζευγαριών σχημάτων. Όταν μια ερώτηση διατυπώνεται στο σχήμα ενός κόμβου, τότε το σύστημα χρησιμοποιεί δεδομένα από όλους τους κόμβους που συνδέονται έμμεσα με τον κόμβο αυτό, χρησιμοποιώντας τους κανόνες αντιστοίχισης.

Συγκεκριμένα το σύστημα αυτό προσφέρει:

- Μια γλώσσα για τη διατύπωση των κανόνων αντιστοίχισης η οποία βασίζεται στην X-Query και μπορεί να δημιουργήσει κανόνες αντιστοίχισης μεταξύ κόμβων που περιέχουν είτε RDF, είτε XML δεδομένα.

- Έναν αλγόριθμο απάντησης ερωτήσεων, ο οποίος συνδυάζει αλυσιδωτά τους κανόνες αντιστοίχισης που ορίζονται μέσω της παραπάνω γλώσσας.

3.2 Στόχος

Τα συστήματα ομότιμων κόμβων που είναι βασισμένα σε σχήμα, έχουν πολύ εκφραστικές δυνατότητες αναζήτησης δεδομένων ενώ ταυτόχρονα παρέχουν ευελιξία στην αναπαράσταση των δεδομένων. Όμως στα συστήματα ομότιμων κόμβων, οι κόμβοι είναι από τη φύση τους αυτόνομοι, που στην προκειμένη περίπτωση σημαίνει ότι πρέπει να μπορούν να δημιουργούν και να χρησιμοποιούν ο καθένας το δικό του ανεξάρτητο σχήμα ανάλογα με τις ανάγκες του. Έτσι προκύπτει το ζήτημα του πώς θα επικοινωνούν αυτοί οι κόμβοι μεταξύ τους για την ανταλλαγή δεδομένων. Όμως όπως αναπτύχθηκε και στην παράγραφο 1.1 η λύση να έχει κάθε κόμβος την αυτονομία να επιλέγει όποιο σχήμα θέλει, απαιτεί την ύπαρξη κανόνων αντιστοίχισης ανάμεσα στα σχήματα ώστε να μπορούν να μετασχηματίζονται οι ερωτήσεις. Επειδή όμως πρόκειται για ένα δυναμικό σύστημα (οι κόμβοι μπορούν να εισέρχονται και να εξέρχονται συνεχώς) και η παραγωγή των κανόνων αντιστοίχισης δε γίνεται ακόμα με αυτόνομο τρόπο, η τελευταία αυτή λύση είναι δύσκολο να υλοποιηθεί.

Στόχος αυτής της εργασίας είναι η ανάπτυξη ενός συστήματος ομότιμων κόμβων βασισμένο σε σχήματα RDF, το οποίο θα δίνει μια σχετική ευελιξία στην επιλογή σχήματος και θα έχει τη δυνατότητα αυτόματου μετασχηματισμού ερωτήσεων χωρίς τη χρήση κανόνων αντιστοίχισης. Κάθε κόμβος αυτού του συστήματος θα χρησιμοποιεί το RDFSculpt για τη δημιουργία του σχήματος που θέλει. Στη συνέχεια, το σύστημα, εκμεταλλεύεται τη συμβατότητα μεταξύ των σχημάτων που προκύπτουν, για το μετασχηματισμό των ερωτήσεων. Συγκεκριμένα αναπτύσσονται τα παρακάτω:

- Η δικτυακή υποδομή του συστήματος ομότιμων κόμβων και ένα είδος πρωτοκόλλου με ειδικά μηνύματα για την επικοινωνία των κόμβων.
- Ένας ειδικός κόμβος ο οποίος έχει βοηθητικό ρόλο και ονομάζεται διαχειριστής του συστήματος.
- Η ενσωμάτωση του RDFSculpt για τη δημιουργία και την αποθήκευση του σχήματος κάθε κόμβου.
- Μια λειτουργία που βοηθάει το χρήστη να ενσωματώσει τα σχεσιακά δεδομένα του στο RDF σχήμα.
- Μια ευέλικτη διαπροσωπεία χρήστη για τη διατύπωση RQL ερωτήσεων.
- Μια λειτουργία ελέγχου της ικανοποιησιμότητας των ερωτήσεων και μετασχηματισμού τους.

Κεφάλαιο 4

Ανάλυση και σχεδίαση

Στο κεφάλαιο αυτό παρουσιάζεται η μελέτη που έγινε για την υλοποίηση του συστήματος. Αρχικά περιγράφεται η αρχιτεκτονική του συστήματος και γίνεται ο διαχωρισμός του στα επιμέρους υποσυστήματα, ενώ στη συνέχεια περιγράφονται οι εφαρμογές του συστήματος.

4.1 Ανάλυση - περιγραφή αρχιτεκτονικής

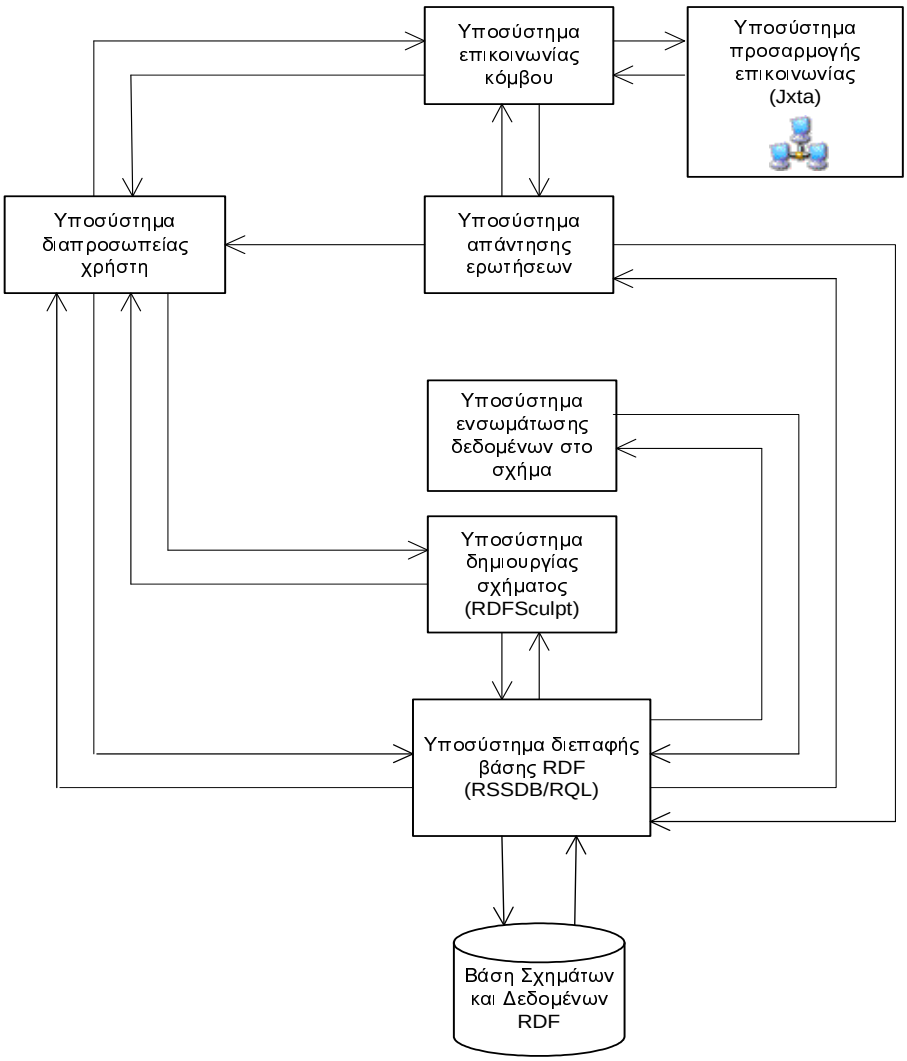
Στην ενότητα αυτή παρουσιάζεται η ανάλυση του συστήματος και ο χωρισμός του σε υποσυστήματα όσον αφορά την αρχιτεκτονική.

4.1.1 Διαχωρισμός υποσυστημάτων

Το σύστημα αποτελείται από τους απλούς κόμβους και ένα κόμβο διαχειριστή. Στο σημείο αυτό αναλύουμε το σύστημα ενός απλού κόμβου, το οποίο αποτελείται από τα εξής υποσυστήματα:

- Υποσύστημα δημιουργίας σχήματος.
- Υποσύστημα ενσωμάτωσης δεδομένων στο σχήμα.
- Υποσύστημα επικοινωνίας κόμβου.
- Υποσύστημα απάντησης ερωτήσεων αναζήτησης.
- Υποσύστημα διαπροσωπείας χρήστη.
- Υποσύστημα διεπαφής βάσης.
- Υποσύστημα προσαρμογής επικοινωνίας.

Το Σχήμα 4.1 απεικονίζει την αρχιτεκτονική ενός απλού κόμβου, όπου φαίνονται τα υποσυστήματα που αναφέραμε και η επικοινωνία μεταξύ τους. Κάθε κόμβος έχει τη δυνατότητα να δημιουργεί ένα καινούργιο σχήμα και στη συνέχεια να ενσωματώνει τα δεδομένα που έχει τοπικά αποθηκευμένα στο σχήμα αυτό. Μέσω της διαπροσωπείας χρήστη γίνεται η αντιστοίχιση των δεδομένων με τις κλάσεις του RDF σχήματος και στη συνέχεια, σχήμα και

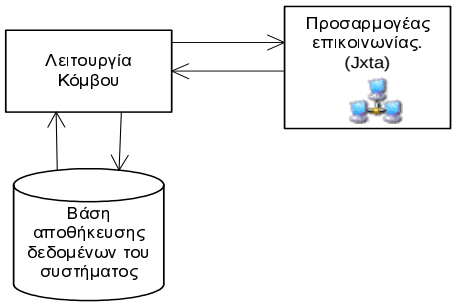


Σχήμα 4.1: Αριτεκτονική Απλού Κόμβου

δεδομένα αποθηκεύονται στην βάση σχημάτων και δεδομένων RDF. Επίσης ο κόμβος έχει τη δυνατότητα να στέλνει ερωτήματα καθώς και να απαντάει ερωτήματα άλλων κόμβων. Η κατασκευή των ερωτημάτων αναζήτησης γίνεται με τη βοήθεια της διαπροσωπείας χρήστη, με βάση κάποιο RDF σχήμα. Όταν ο κόμβος λάβει κάποιο ερώτημα, το προωθεί στο υποσύστημα απάντησης ερωτήσεων για να αποτιμηθεί. Το υποσύστημα επικοινωνίας κόμβου επικοινωνεί με τη διαπροσωπεία χρήστη και τον προσαρμογέα επικοινωνίας για την αποστολή και τη λήψη μηνυμάτων προς και από το δίκτυο ομότιμων κόμβων. Το υποσύστημα προσαρμογής επικοινωνίας είναι υπεύθυνο για τη δικτυακή επικοινωνία μεταξύ των κόμβων.

Το σύστημα του κόμβου διαχειριστή, του οποίου το διάγραμμα αρχιτεκτονικής φαίνεται στο Σχήμα 4.2, αποτελείται από τα εξής υποσυστήματα:

- Υποσύστημα επικοινωνίας κόμβου διαχειριστή.
- Υποσύστημα προσαρμογής επικοινωνίας.



Σχήμα 4.2: Αριτεκτονική Κόμβου Διαχειριστή

Ο κόμβος διαχειριστής είναι υπεύθυνος για να διατηρεί στη βάση του στοιχεία τοπολογίας του συστήματος καθώς και στοιχεία για τον κάθε κόμβο που είναι συνδεδεμένος στο σύστημα και να τα στέλνει σε όποιο κόμβο τα χρειαστεί μέσω του προσαρμογέα επικοινωνίας.

4.1.2 Περιγραφή υποσυστημάτων

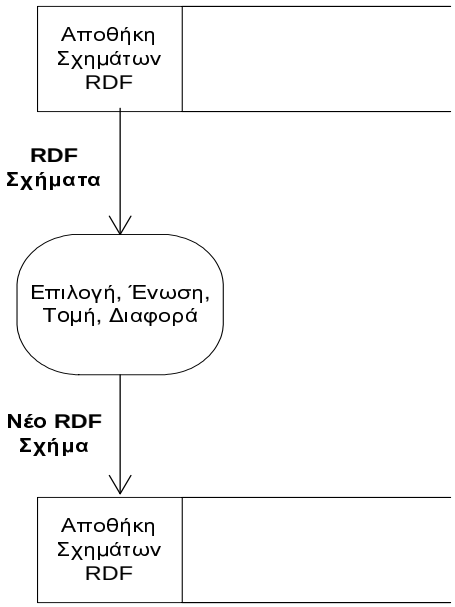
Παρακάτω δίνεται λεπτομερής περιγραφή για καθένα από τα συστήματα που αναφέραμε. Η περιγραφή αυτή γίνεται με βάση τα διαγράμματα ροής δεδομένων.

Υποσύστημα δημιουργίας σχήματος

Κάθε κόμβος συμμετέχει στο σύστημα με ένα σχήμα RDF. Το υποσύστημα λοιπόν αυτό βοηθάει το χρήστη να δημιουργήσει το σχήμα αυτό, κάτω από το οποίο θα αποθηκεύσει στη συνέχεια τα δεδομένα του. Όταν ένας κόμβος εισέρχεται στο σύστημα, επικοινωνεί με τον κόμβο διαχειριστή και αυτός του στέλνει το καθολικό σχήμα του συστήματος, το οποίο αποθηκεύει στην βάση σχημάτων και δεδομένων RDF. Ξεκινώντας λοιπόν από αυτό το σχήμα και χρησιμοποιώντας πράξεις που εφαρμόζονται στα σχήματα, ο χρήστης δημιουργεί διάφορα σχήματα μέχρι να καταλήξει τελικά στο σχήμα που θα χρησιμοποιήσει. Στο τέλος έχει τη δυνατότητα επισκόπησης του σχήματος που δημιούργησε. Το υποσύστημα λοιπόν αυτό, παρέχει τις λειτουργίες και το γραφικό περιβάλον που χρειάζονται για την εκτέλεση πράξεων (επιλογή, ένωση, τομή, διαφορά) πάνω σε RDF σχήματα. Ως υποσύστημα δημιουργίας σχήματος χρησιμοποιήθηκε το RDFSculpt[19] το οποίο περιγράφηκε στην ενότητα 3.1.4. Το διάγραμμα ροής δεδομένων του συστήματος αυτού φαίνεται στο Σχήμα 4.3.

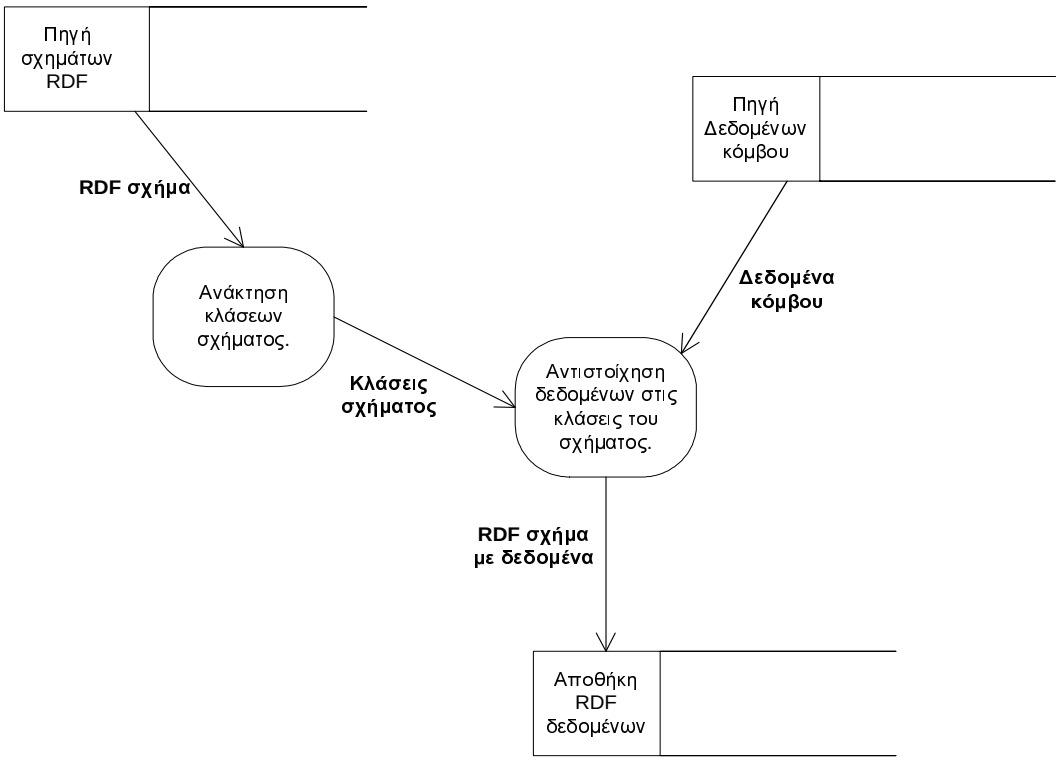
Υποσύστημα ενσωμάτωσης δεδομένων στο σχήμα

Το υποσύστημα αυτό είναι υπεύθυνο για την αποθήκευση των τοπικά αποθηκευμένων δεδομένων του κόμβου κάτω από το RDF σχήμα. Αρχικά γίνεται επιλογή, μέσω της διαπροσωπείας χρήστη, ενος RDF σχήματος, από αυτά που υπάρχουν στην βάση αποθήκευσης. Στη συνέχεια γίνεται ανάκτηση των κλάσεων του σχήματος και των τοπικά αποθηκευμένων δεδομένων του κόμβου και με τη βοήθεια της διαπροσωπείας χρήστη γίνεται αντιστοίχιση των δεδομένων σε συγκεκριμένες κλάσεις του σχήματος. Το αποτέλεσμα, δηλαδή το σχήμα



Σχήμα 4.3: Υποσύστημα δημιουργίας σχήματος

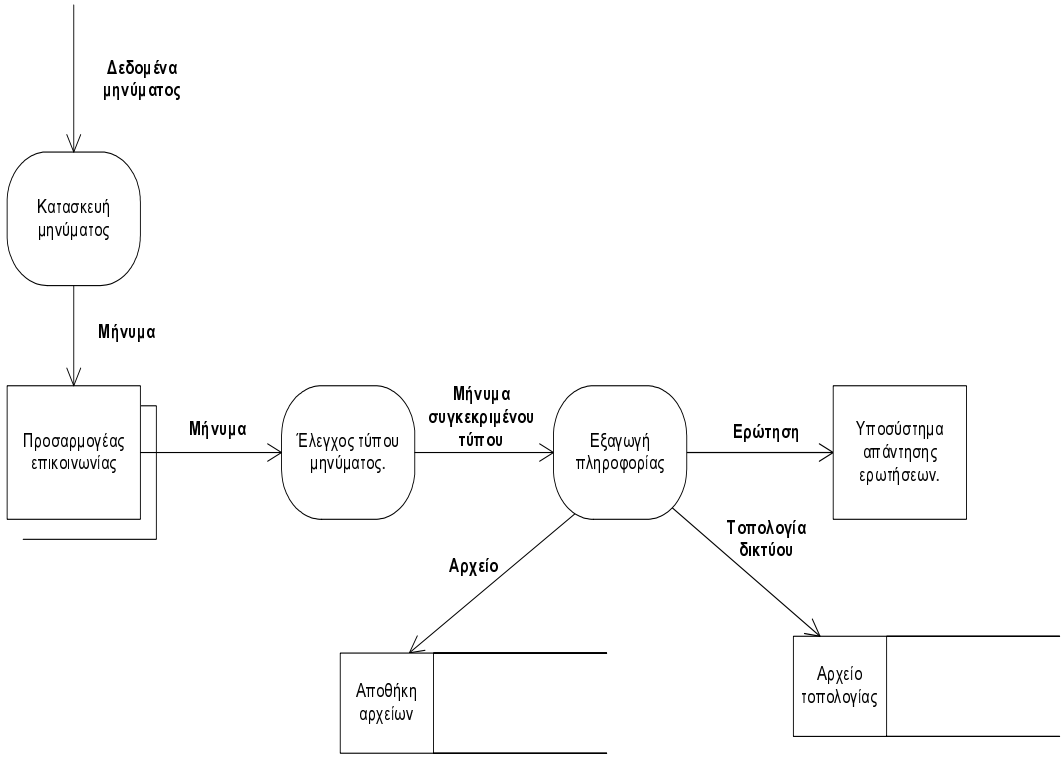
με τα αντίστοιχα δεδομένα, αποθηκεύεται στην βάση. Το διάγραμμα ροής δεδομένων του υποσυστήματος αυτού φαίνεται στο Σχήμα 4.4.



Σχήμα 4.4: Υποσύστημα ενσωμάτωσης δεδομένων

Υποσύστημα επικοινωνίας κόμβου

Το υποσύστημα αυτό είναι υπεύθυνο για τη σύνδεση του κόμβου στο δίκτυο ομότιμων κόμβων, καθώς και για την αποστολή και λήψη μηνυμάτων. Ανάλογα με το είδος των δεδομένων προς αποστολή (αρχείο, ερώτηση, γενικές πληροφορίες), κατασκευάζεται το κατάλληλο μήνυμα και αποστέλεται μέσω του προσαρμογέα επικοινωνίας. Επίσης το υποσύστημα αυτό διαχειρίζεται τα μηνύματα που λαμβάνει από το δίκτυο. Αν πρόκειται για μήνυμα που περιέχει αρχείο, το αρχείο αποθηκεύεται στην αποθήκη αρχείων. Αν πρόκειται για μήνυμα ερώτησης, η ερώτηση δρομολογείται στο υποσύστημα απάντησης ερώτησης. Τέλος αν πρόκειται για μήνυμα που περιέχει πληροφορία σχετικά με την τοπολογία του δικτύου ομότιμων κόμβων, η πληροφορία αυτή αποθηκεύεται στο αρχείο τοπολογίας. Το διάγραμμα ροής δεδομένων του συγκεκριμένου υποσυστήματος φαίνεται στο Σχήμα 4.5.

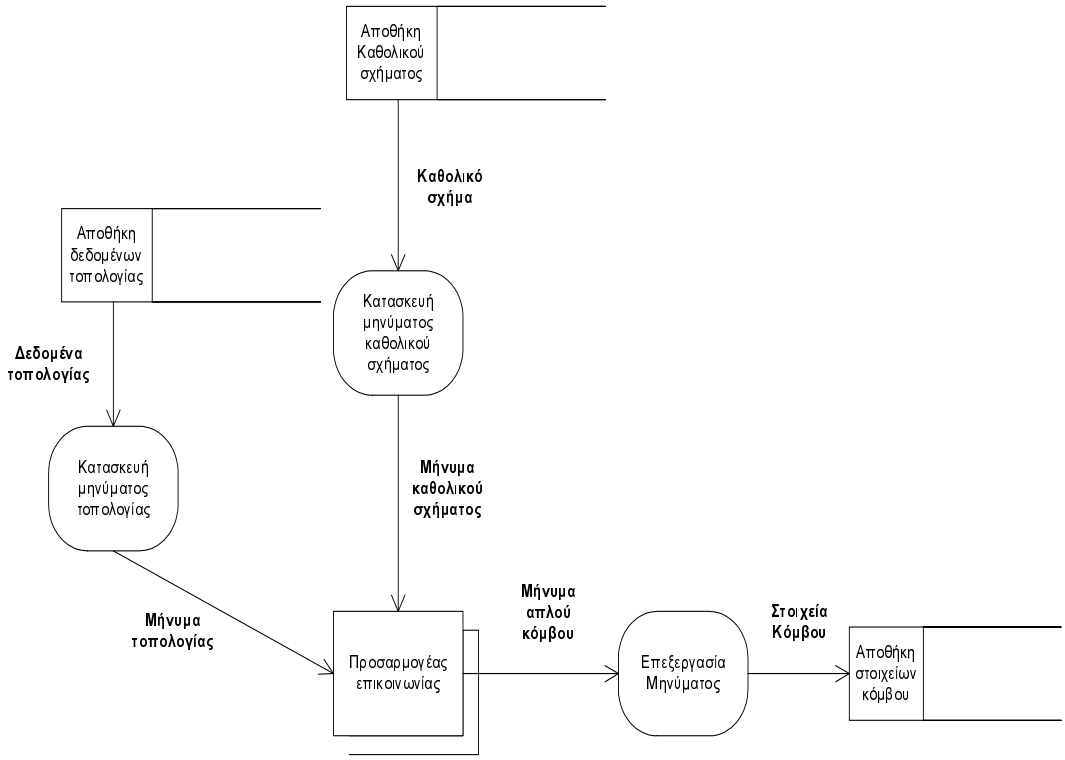


Σχήμα 4.5: Υποσύστημα επικοινωνίας κόμβου

Υποσύστημα επικοινωνίας του κόμβου διαχειριστή

Το υποσύστημα αυτό είναι υπεύθυνο για για τη διαχείριση και αποστολή μηνυμάτων, από και προς τους υπόλοιπους κόμβους. Όταν δέχεται ένα μήνυμα ενός απλού κόμβου μέσω του προσαρμογέα επικοινωνίας, το επεξεργάζεται και στη συνέχεια τα δεδομένα του μηνύματος, δηλαδή τα στοιχεία του απλού κόμβου αποθηκεύονται στην αποθήκη στοιχείων κόμβου. Επίσης για να στείλει ένα μήνυμα τοπολογίας ή καθολικού σχήματος αντλεί τα αντίστοιχα δεδομένα από τις αντίστοιχες αποθήκες, κατασκευάζει το μήνυμα και το στέλνει μέσω του

προσαρμογέα επικοινωνίας. Το διάγραμμα ροής δεδομένων του συγκεκριμένου υποσυστήματος φαίνεται στο Σχήμα 4.6.



Σχήμα 4.6: Υποσύστημα επικοινωνίας κόμβου διαχειριστή

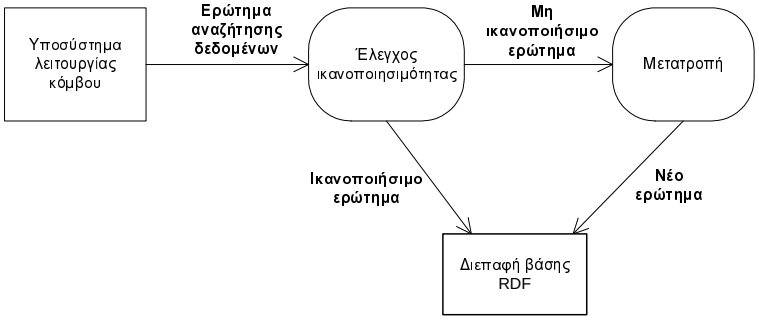
Υποσύστημα απάντησης ερωτήσεων αναζήτησης

Το υποσύστημα απάντησης ερωτήσεων είναι υπεύθυνο για την κατάλληλη μετατροπή των ερωτήσεων που φθάνουν σ΄ ένα κόμβο ώστε να μπορούν ν΄ απαντηθούν. Το υποσύστημα αυτό παίρνει σαν είσοδο το ερώτημα από το υποσύστημα επικοινωνίας κόμβου και ελέγχει κατά πόσο το ερώτημα αυτό είναι ικανοποιήσιμο με βάση το σχήμα αυτού του κόμβου. Αν δεν είναι το μετατρέπει έτσι ώστε να γίνει ικανοποιήσιμο και στη συνέχεια το προωθεί στη διεπαφή βάσης RDF για να αποτιμηθεί. Αν είναι ικανοποιήσιμο προωθείται απευθείας στη βάσης RDF. Το διάγραμμα του υποσυστήματος αυτού φαίνεται στο Σχήμα 4.7.

Υποσύστημα διαπροσωπείας χρήστη

Το υποσύστημα αυτό αποτελεί τη διεπαφή όλου του συστήματος με το χρήστη. Στην περίπτωση όπου ο χρήστης επιλέξει να δημιουργήσει ένα νέο σχήμα, το υποσύστημα αυτό επικοινωνεί με το υποσύστημα δημιουργίας σχήματος, βοηθώντας τον να επιλέξει τα αρχικά σχήματα, την πράξη που θα εφαρμόσει καθώς και τις κλάσεις από το κάθε σχήμα που θέλει να χρησιμοποιήσει για τη συγκεκριμένη πράξη.

Όταν ο χρήστης επιλέξει να εισάγει στιγμιότυπα (δεδομένα) σε συγκεκριμένο σχήμα, τότε αρχικά επιλέγει το σχήμα, στη συνέχεια χρησιμοποιεί μια φόρμα που τον βοηθά να ανακτήσει



Σχήμα 4.7: Υποσύστημα απάντησης ερωτήσεων

τα δεδομένα που έχει τοπικά αποθηκευμένα και στη συνέχεια το υποσύστημα τον βοηθά να αντιστοιχίσει αυτά τα δεδομένα σε συγκεκριμένες κλάσεις του σχήματος.

Τέλος μέσω της διαπροσωπείας χρήστη, ο χρήστης έχει τη δυνατότητα να κατασκευάσει και να στείλει ερωτήματα αναζήτησης δεδομένων από τους υπόλοιπους κόμβους. Αρχικά, επιλέγεται ένα RDF σχήμα και στη συνέχεια επιλέγονται οι κλάσεις των οποίων τα στιγμιότυπα αναζητούνται. Έτσι κατασκευάζεται το ερώτημα αναζήτησης δεδομένων και στη συνέχεια προωθείται στο υποσύστημα επικοινωνίας κόμβου.

Υποσύστημα διεπαφής βάσης

Το υποσύστημα διεπαφής βάσης είναι υπεύθυνο για την αποθήκευση των σχημάτων και των δεδομένων RDF, στη σχεσιακή βάση δεδομένων καθώς και για την αποτίμηση των ερωτημάτων.

Υποσύστημα προσαρμογής επικοινωνίας

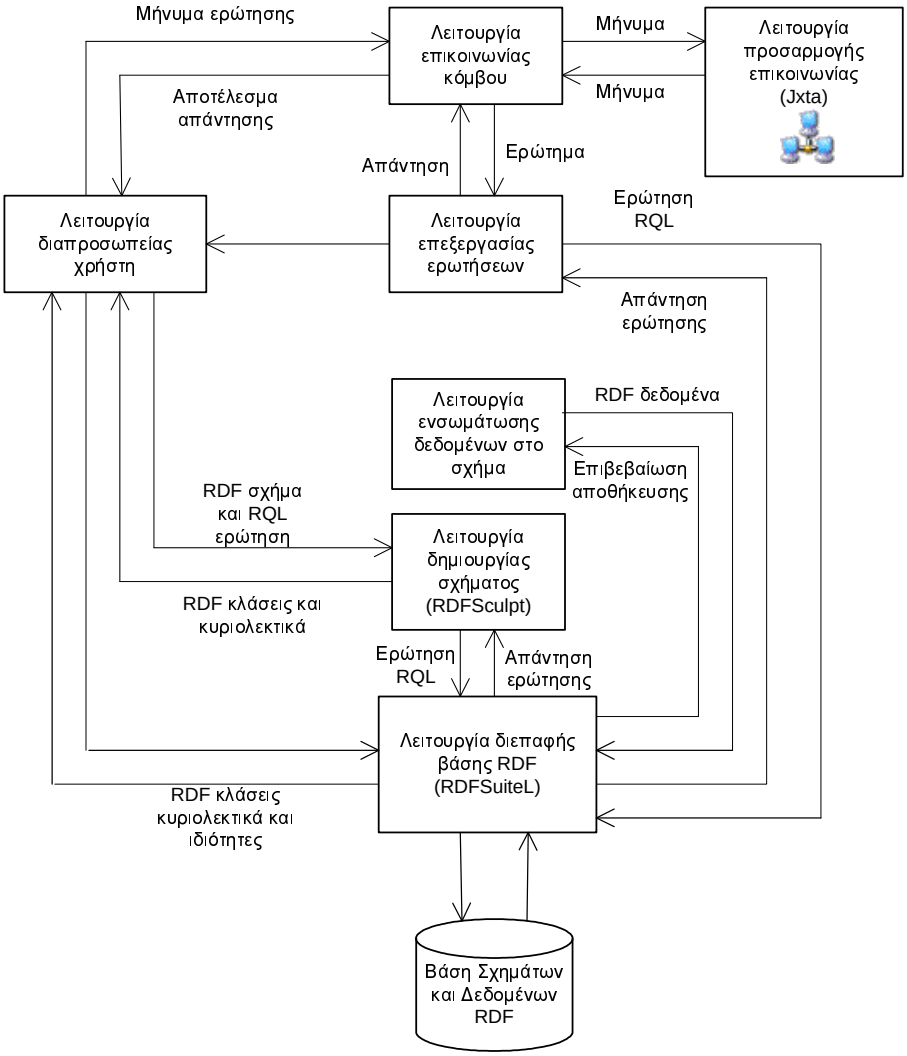
Το υποσύστημα προσαρμογής επικοινωνίας είναι υπεύθυνο για τη δικτυακή επικοινωνία μεταξύ των κόμβων, δηλαδή για την αποστολή και λήψη μηνυμάτων που στέλνουν οι κόμβοι μεταξύ τους.

4.2 Σχεδίαση

Στην ενότητα αυτή παρουσιάζεται η σχεδίαση του συστήματος, δηλαδή περιγράφονται οι λειτουργίες που αυτό εκτελεί. Οι λειτουργίες του συστήματος καθώς και τα δεδομένα που ανταλλάσουν μεταξύ τους φαίνονται στο συνολικό διάγραμμα του Σχήματος 4.8.

4.2.1 Λειτουργίες

Για τη λειτουργία διαπροσωπείας βάσης χρησιμοποιήθηκε το σύστημα RDFSuite. Για τη λειτουργία δημιουργίας σχήματος χρησιμοποιήθηκε το σύστημα RDFSculpt και για τη λειτουργία προσαρμογής επικοινωνίας το σύστημα Jxta. Οι λειτουργίες οι οποίες αναπτύχθηκαν στα πλαίσια αυτής της διπλωματικής είναι οι εξής:



Σχήμα 4.8: Συνολικό διάγραμμα λειτουργιών

- Λειτουργία επικοινωνίας απλού κόμβου.
- Λειτουργία επικοινωνίας κόμβου διαχειριστή.
- Λειτουργία επεξεργασίας ερωτήσεων.
- Λειτουργία ενσωμάτωσης δεδομένων.
- Λειτουργία διαπροσωπείας χρήστη.

Οι λειτουργίες αυτές περιγράφονται παρακάτω με τη χρήση διαγραμμάτων δραστηριοτήτων.

Λειτουργία επικοινωνίας απλού κόμβου

Όταν ένας κόμβος ξεκινάει να λειτουργεί, κάνει τις αρχικοποιήσεις που χρειάζονται ώστε να συνδεθεί στο δίκτυο ομότιμων κόμβων. Στη συνέχεια περιμένει μέχρι να ανιχνεύσει το δίκτυο και να συνδεθεί τελικά σε αυτό. Αφού συνδεθεί, στέλνει μήνυμα στον κόμβο διαχειριστή,

που περιλαμβάνει το όνομά του, τον αριθμό ταυτοποίησης του και το όνομα της RDF βάσης που θα χρησιμοποιήσει. Στη συνέχεια ξεκινάει η διαδικασία αναμονής για νέα μηνύματα, η οποία του επιτρέπει να δέχεται μηνύματα, ασύγχρονα, από τους υπόλοιπους κόμβους και να τα διαχειρίζεται κατάλληλα.

Όταν δεχθεί ένα μήνυμα ερώτησης, αφού απαντήσει την ερώτηση, εξετάζει αν η ερώτηση έχει φθάσει το μέγιστο αριθμό κόμβων από τους οποίους της επιτρέπεται ν΄ απαντηθεί. Αν όχι την προωθεί στους γειτονικούς του κόμβους. Αν ναι συνεχίζει να περιμένει για νέα μηνύματα. Όταν δεχθεί μήνυμα αρχείου, το αποθηκεύει. Το αρχείο αυτό μπορεί να είναι το αρχείο του καθολικού σχήματος από τον κόμβο διαχειριστή ή απάντηση σε κάποια ερώτηση. Σε περίπτωση που δεχθεί μήνυμα τοπολογίας από τον κόμβο διαχειριστή, αποθηκεύει τους γειτονικούς του κόμβους.

Κατά τη διαδικασία αναμονής για νέα μηνύματα, υπάρχει περίπτωση ο κόμβος να σταματήσει να λειτουργεί οπότε τερματίζει και η επικοινωνία του με το δίκτυο.

Το διάγραμμα καταστάσεων της λειτουργίας αυτής φαίνεται στο Σχήμα 4.9.

Λειτουργία επικοινωνίας κόμβου διαχειριστή

Όταν ο κόμβος ξεκινάει να λειτουργεί, κάνει τις αρχικοποιήσεις που χρειάζονται ώστε να συνδεθεί στο δίκτυο ομότιμων κόμβων. Επειδή ο κόμβος διαχειριστής είναι ο πρώτος που συνδέεται στο δίκτυο, είναι και αυτός ο οποίος δημιουργεί το δίκτυο. Αφού το δημιουργήσει, συνδέεται σε αυτό. Στη συνέχεια ξεκινάει η διαδικασία αναμονής για νέα μηνύματα, η οποία λειτουργεί με τον ίδιο τρόπο με την αντίστοιχη διαδικασία του απλού κόμβου.

Όταν ο κόμβος διαχειριστής δεχθεί μήνυμα από κάποιον απλό κόμβο, αποθηκεύει τα στοιχεία του και στη συνέχεια του στέλνει ένα μήνυμα με το καθολικό RDF σχήμα καθώς και ένα μήνυμα τοπολογίας που περιλαμβάνει του κόμβουσ-γείτονες, του κόμβου αυτού.

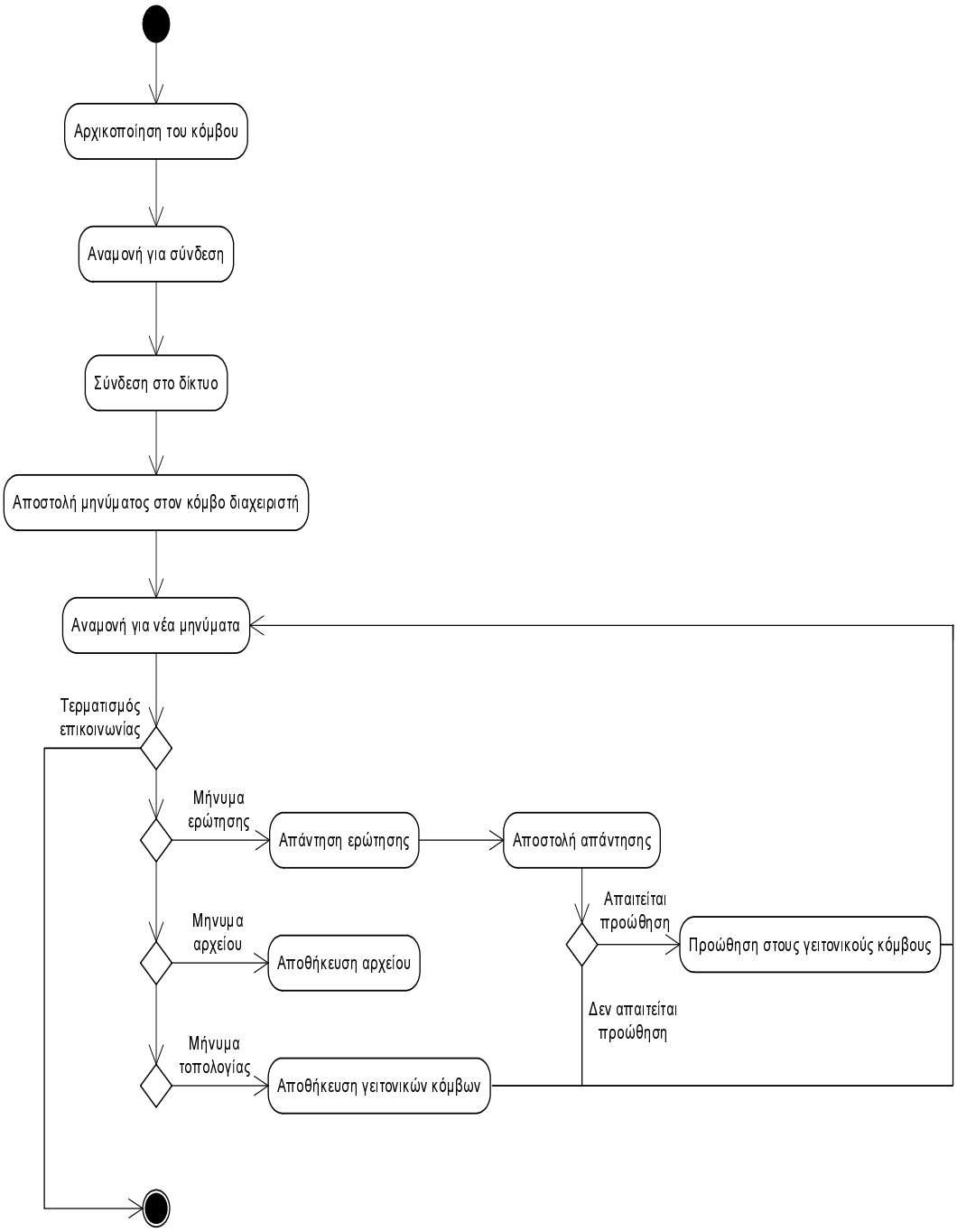
Κατά τη διαδικασία αναμονής για νέα μηνύματα, υπάρχει περίπτωση ο κόμβος να σταματήσει να λειτουργεί οπότε τερματίζει και η επικοινωνία του με το δίκτυο. Στην περίπτωση αυτή όμως δε χάνει τα σχτοιχεία που κρατάει αποθηκευμένα και έτσι όταν ξαναρχίσει να λειτουργεί, η λειτουργία του συστήματος συνεχίζεται κανονικά.

Το διάγραμμα καταστάσεων της λειτουργίας αυτής φαίνεται στο Σχήμα 4.10.

Λειτουργία επεξεργασίας ερωτήσεων

Η λειτουργία επεξεργασίας ερωτήσεων, ξεκινάει με την ανάγνωση των ρυθμίσεων του ερωτήματος. Στην περίπτωση που το ερώτημα έχει απαντηθεί από το μέγιστο αριθμό κόμβων η διαδικασία σταματάει. Αλλιώς εξετάζεται αν το ερώτημα είναι ικανοποιήσιμο. Αν είναι, τότε μετατρέπεται σε ερώτηση RQL και αποτιμάται. Αν δεν είναι ικανοποιήσιμο, ελέγχεται αν η ρύθμιση επίτρεψης μετατροπής είναι αληθής. Αν είναι αληθής, μετατρέπεται σε ικανοποιήσιμο ερώτημα και στη συνέχεια σε ερώτηση RQL και αποτιμάται. Αν δεν είναι αληθής η ρύθμιση επίτρεψης μετατροπής τότε το ερώτημα δεν αποτιμάται και σταματάει η διαδικασία.

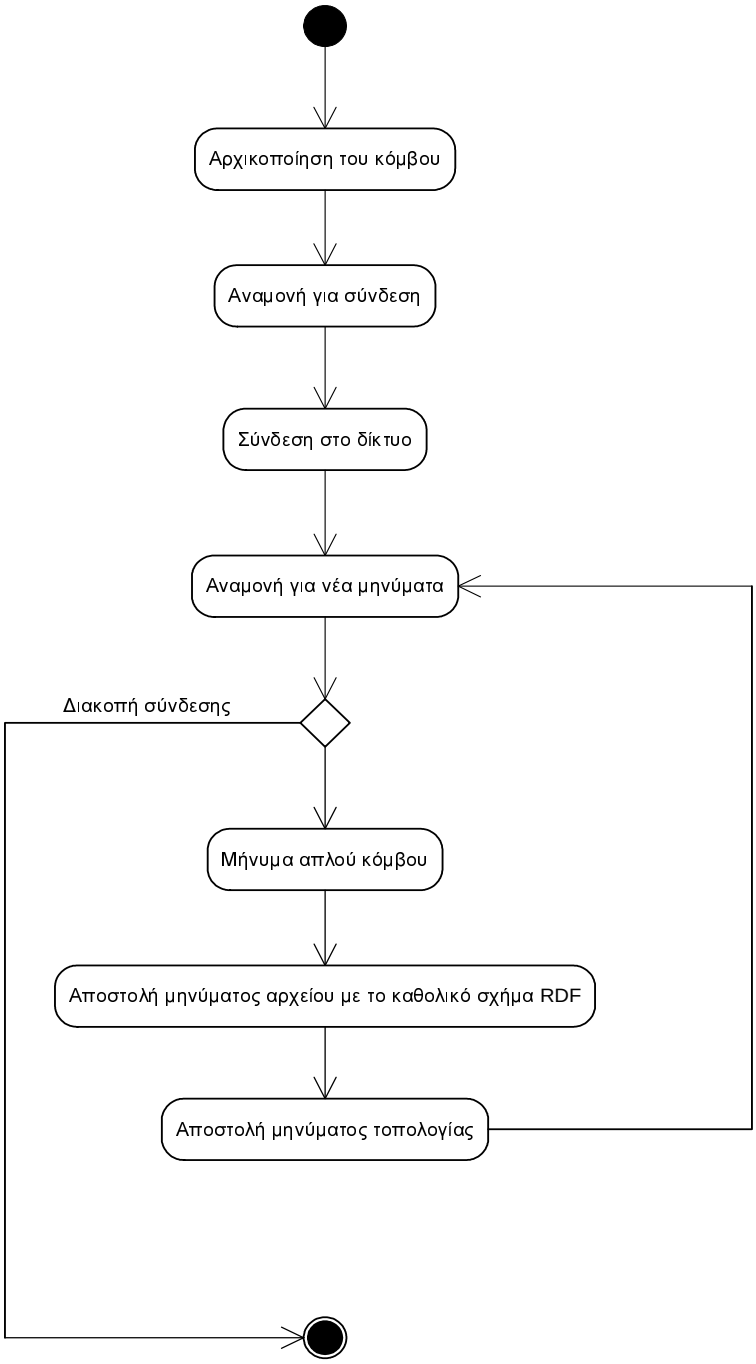
Το διάγραμμα καταστάσεων της λειτουργίας αυτής φαίνεται στο Σχήμα 4.11.



Σχήμα 4.9: Λειτουργία επικοινωνίας κόμβου

Λειτουργία ενσωμάτωσης δεδομένων

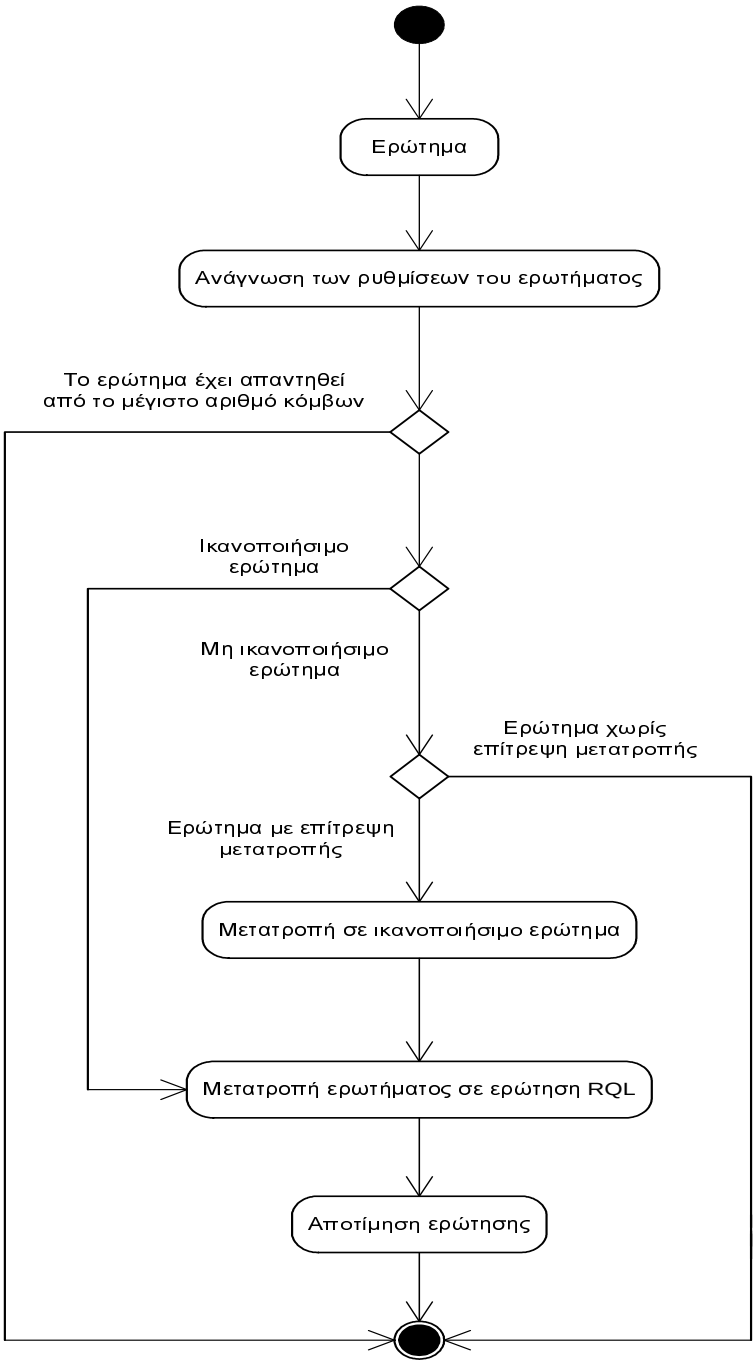
Η λειτουργία ενσωμάτωσης δεδομένων, αποτελεί τη διαδικασία κατά την οποία τα δεδομένα του κόμβου αποθηκεύονται κάτω από το RDF σχήμα. Υποθέτουμε ότι όλοι οι κόμβοι έχουν αποθηκευμένα τα δεδομένα τους τοπικά σε μία σχεσιακή βάση δεδομένων. Η διαδικασία εισαγωγής στο σχήμα γίνεται ως εξής. Αρχικά μέσω της διαπροσωπείας χρήστη, ο χρήστης διατυπώνει ένα SQL ερώτημα και ανακτά μέρος των δεδομένων. Στη συνέχεια και πάλι



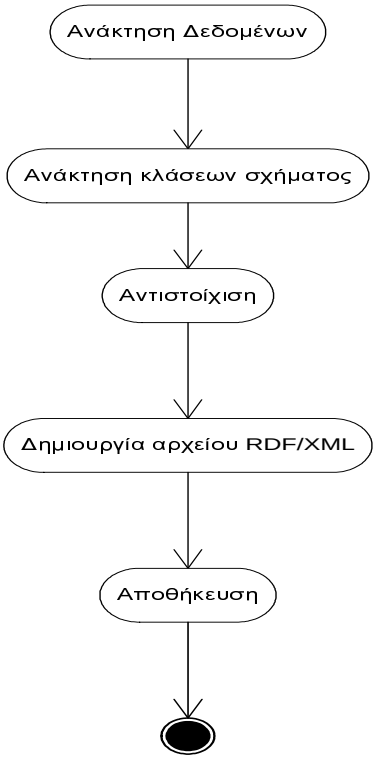
Σχήμα 4.10: Λειτουργία επικοινωνίας κόμβου διαχειριστή

μέσω της διαπροσωπείας χρήστη, αντιστοιχίζει κάθε χαρακτηριστικό του αποτελέσματος που προέκυψε από το SQL ερώτημα σε ένα χαρακτηριστικό μιας κλάσης του RDF σχήματος. Τέλος με βάση τις αντιστοιχίσεις που προκύπτουν δημιουργείται το κατάλληλο RDF αρχείο για την αποθήκευση των δεδομένων στην RDF βάση.

Το διάγραμμα καταστάσεων της λειτουργίας αυτής φαίνεται στο Σχήμα 4.12.



Σχήμα 4.11: Λειτουργία επεξεργασίας ερωτήσεων



Σχήμα 4.12: Λειτουργία ενσωμάτωσης δεδομένων

Κεφάλαιο 5

Υλοποίηση

Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση του συστήματος, με βάση τη μελέτη που παρουσιάστηκε στο προηγούμενο κεφάλαιο. Αρχικά παρουσιάζεται η πλατφόρμα και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν. Στη συνέχεια δίνονται οι λεπτομέρειες υλοποίησης για τους βασικούς αλγορίθμους του συστήματος καθώς και η δομή του κώδικα.

5.1 Λεπτομέρειες υλοποίησης

Στην ενότητα αυτή παρουσιάζονται οι βασικοί αλγόριθμοι που αναπτύχθηκαν καθώς και λεπτομέρειες σχετικά με την υλοποίηση της επικοινωνίας των κόμβων.

5.1.1 Αλγόριθμοι

Αλγόριθμος εισαγωγής δεδομένων

Όταν ένας κόμβος εισέρχεται για πρώτη φορά στο σύστημα, αρχικά δημιουργεί το σχήμα που θέλει χρησιμοποιώντας το RDFSculpt. Στη συνέχεια πρέπει να αποθηκεύει τα τοπικά δεδομένα, που υποθέτουμε ότι τα διατηρεί σε μια σχεσιακή βάση δεδομένων, κάτω από το RDF σχήμα. Αρχικά μέσω της διαπροσωπείας χρήστη, ο χρήστης διατυπώνει ένα SQL ερώτημα και ανακτά μέρος των δεδομένων. Στη συνέχεια και πάλι μέσω της διαπροσωπείας χρήστη, αντιστοιχίζει κάθε χαρακτηριστικό του αποτελέσματος που προέκυψε από το SQL ερώτημα σε ένα κυριολεκτικό μιας κλάσης του RDF σχήματος.

Ο αλγόριθμος εισαγωγής δεδομένων παίρνει σαν είσοδο τις αντιστοιχίσεις που προκύπτουν από την παραπάνω διαδικασία και δημιουργεί ένα RDF/XML αρχείο που περιέχει τα δεδομένα ως στιγμιότυπα των κλάσεων και είναι κατάλληλο για να αποθηκευτεί στην RDF βάση.

Αρχικά παρουσιάζεται ο αλγόριθμος σε ψευδοκώδικα και στη συνέχεια δίνεται ένα παράδειγμα εφαρμογής του. Για να γίνει δυνατή η περιγραφή του αλγορίθμου σημειώνεται ότι τα δεδομένα που έχουν ανακτηθεί από τη βάση αποτελούν μια όψη, με στήλες-χαρακτηριστικά και γραμμές−εγγραφές. Αφού κάθε χαρακτηριστικό αντιστοιχεί σε ένα κυριολεκτικό και η αντιστοιχία αυτή είναι ένα προς ένα, τα δεδομένα εισάγονται στο RDF/XML αρχείο ως τιμές κυριολεκτικών.

Είσοδος: Η όψη που προέκυψε από το SQL ερώτημα και το διάνυσμα, mapping, με τις αντιστοιχίσεις των κυριολεκτικών των κλάσεων του RDF σχήματος με τα χαρακτηριστικά των σχεσιακών δεομένων. Η μορφή του διανύσματος είναι η εξής:
[[[Κλάση1,Κυριολεκτικό1],Χαρακτηριστικό1],[[Κλάση2,Κυριολεκτικό2], Χαρακτηριστικό2],...]

Έξοδος: Αρχείο RDF/XML με τα δεδομένα.

Αλγόριθμος

```
Κατασκευή του διανύσματος groupedMapping.
Περιέχει ομαδοποιημένα τα στοιχεία του mapping που ανήκουν στην ίδια κλάση.
Το διάνυσμα groupedMapping έχει τη μορφή:
[[[[Κλάση1,Κυριολεκτικό1],Χαρακτηριστικό1],[[Κλάση1,Κυριολεκτικό2]
,Χαρακτηριστικό2],...],[[[Κλάση2,Κυριολεκτικό3],Χαρακτηριστικό3], [[Κλάση2,
Κυριολεκτικό4],Χαρακτηριστικό4],...]]

Για κάθε εγγραφή
Δημιούργησε αντίγραφο του groupedMapping, που ονομάζεται imapping
΄Οσο το imapping έχει στοιχεία
  Πάρε το πρώτο στοιχείο του διανύσματος έστω classMapping
     Βάλε την κλάση που ανήκει στο πρώτο στοιχείο, στο διάνυσμα
     classesToWrite
     ΄Οσο το διάνυσμα classesToWrite έχει στοιχεία
         Πάρε το στοιχείο-κλάση που βρίσκεται στην αρχή του διανύσματος
         Αν είμαστε στην πρώτη κλάση που γράφουμε στο αρχείο εξόδου
           Γράψε το opening tag της κλάσης στο αρχείο εξόδου
           (<Class rdf:about="#Item i">)
         Αλλιώς
           Από το διάνυσμα classProperties βρες την ιδιότητα
           της οποίας το πεδίο τιμών είναι η κλάση
           Γράψε το opening tag της ιδιότητας στο αρχείο εξόδου
           (<Property>)
           Γράψε το opening tag της κλάσης στο αρχείο εξόδου
           (<Class rdf:about="#Item i">)
         Για κάθε κυριολεκτικό του classMapping βρες το χαρακτηριστικό
         που του αντιστοιχεί, καθώς και την τιμή του χαρακτηριστικού για
         τη συγκεκριμένη εγγραφή.
         Γράψε στο αρχείο εξόδου τα κυριολεκτικά της κλάσης
         (<Literal>τιμή χαρακτηριστικού < /Literal>)
         Διέγραψε από το imapping το στοιχείο classMapping
```

```
Πρόσθεσε την κλάση στο διανυσμα writtenClasses
Διέγραψε την κλάση από το διάνυσμα classesToWrite
Βρες τις ιδιότητες που μορούν να εφαρμοστούν στην κλάση και από
αυτές αφαίρεσε αυτές που αποτελούν υπεριδιότητες των άμεσων
ιδιοτήτων της.
Για κάθε ιδιότητα
  Αν η κλάση-πεδίο τιμών της περιέχεται στο imapping
  Πρόσθεσέ τη στην αρχή του διανύσματος classesToWrite
Βάλε στο διάνυσμα classProperties το διάνυσμα:
[Κλάση,[Ιδιότητα1,ΠεδιοΤιμών1],[Ιδιότητα2,ΠεδίοΤιμών2],...]
Αν η κλάση δεν έχει ιδιότητες ή καμία από τις ιδιότητές της δεν
περιέχεται στο imapping
  Για κάθε στοιχείο-κλάση του writtenClasses
  (ξεκινώντας από το τελευταίο)
    Βρες τις ιδιότητες της κλάσης από το διάνυσμα classProperties
    Για κάθε ιδιότητα
      Αν η κλάση-πεδίο τιμών της περιέχεται στο διάνυσμα
      closedClasses
      Γράψε το closing tag της ιδιότητας στο αρχείο εξόδου
      (< /Property>)
      Πρόσθεσε την ιδιότητα στο closedProperties
    Αν όλες οι ιδιότητες της κλάσης περιέχονται στο
    closedProperties ή δεν έχει ιδιότητες
    Γράψε στο αρχείο εξόδου το closing tag της κλάσης
    Πρόσθεσε την κλάση στο closedClasses.
```

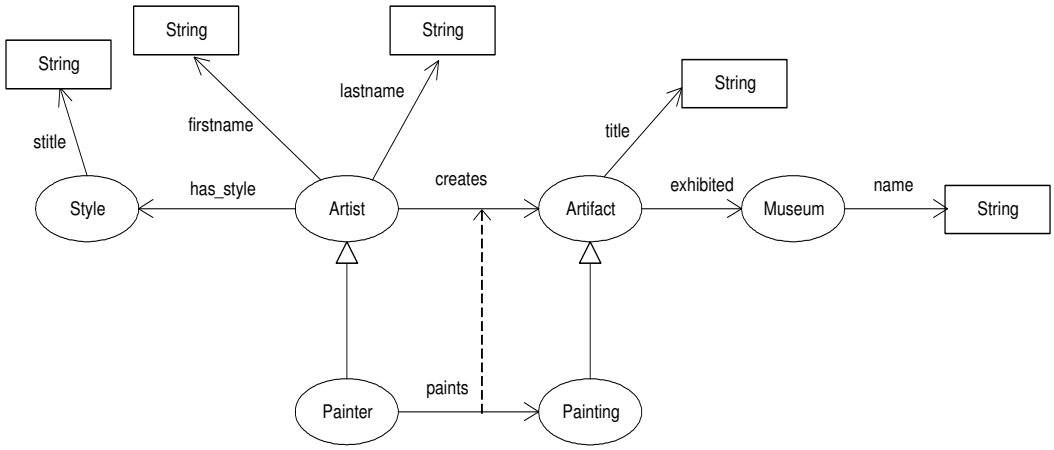
Παράδειγμα

Έστω ότι ο κόμβος έχει επιλέξει να συμμετέχει στο σύστημα με το RDF σχήμα που φαίνεται στο Σχήμα 5.1. Έστω επίσης ότι από το SQL ερώτημα που έχει κάνει στη σχεσιακη βάση, έχει προκύψει η όψη που φαίνεται στον Πίνακα 5.1. Για τις ανάγκες του παραδείγματος θεωρούμε ότι η όψη αυτή περιέχει μόνο μία εγγραφή.

Name	Surname	Style	Artifact	Museum
Pablo	Picasso	Cubist	Guernica	Reina Sofia

Πίνακας 5.1: Η όψη αποτέλεσμα του SQL ερωτήματος

Έστω επίσης ότι η αντιστοίχιση που έχει ορίσει ο χρήστης είναι η εξής (σε μορφή του διανύσματος mapping): [[[Painter,firstname],Name],[[Painter,lastname],Surname], [[Style,stitle], Style],[[Painting,title],Artifact],[[Museum,name],Museum]]



Σχήμα 5.1: Παράδειγμα RDF σχήματος κόμβου

Το διάνυσμα imapping θα έχει αρχικά τη μορφή: [[[[Painter,firstname],Name],[[Painter, lastname],Surname]],[[[Style,stitle],Style]],[[[Painting,title],Artifact]],[[[Museum,name], Museum]]]]

Για κάθε επανάληψη του αλγορίθμου παρουσιάζονται οι τιμές των βασικών διανυσμάτων καθώς και το περιεχόμενο του αρχείου εξόδου όπως διαμορφώνονται μετά το τέλος της επανάληψης.

Πρώτη Επανάληψη

imapping: [[[Style,stitle],Style],[[Painting,title],Artifact],[[Museum,name],Museum]]
writtenClasses: [Painter]
classesToWrite: [Painting,Style]
classProperties: [[Painter,[paints,Painting],[hasstyle,Style]]]

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:sref="file:./rdf/peerSchema.rdfs#">

<sref:Painter rdf:about="#Item0">
   <sref:firstname>Pablo</sref:firstname>
   <sref:lastname>Picasso</sref:lastname>
```

Δεύτερη Επανάληψη

imapping: [[[Style,stitle],Style],[[Museum,name],Museum]]
writtenClasses: [Painter,Painting]
classesToWrite: [Museum,Style]
classProperties: [[Painter,[paints,Painting],[hasstyle,Style]],[Painting,[exhibited,Museum]]]

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:sref="file:./rdf/peerSchema.rdfs#">

<sref:Painter rdf:about="#Item0">
   <sref:firstname>Pablo</sref:firstname>
   <sref:lastname>Picasso</sref:lastname>
   <sref:paints>
      <sref:Painting rdf:about="#Item1">
         <sref:title>Guernica</sref:title>
```

Τρίτη Επανάληψη

imapping: [[[Style,stitle],Style]]
writtenClasses: [Painter,Painting,Museum]
classesToWrite: [Style]
classProperties: [[Painter,[paints,Painting],[hasstyle,Style]],[Painting,[exhibited,Museum]]]

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:sref="file:./rdf/peerSchema.rdfs#">

<sref:Painter rdf:about="#Item0">
   <sref:firstname>Pablo</sref:firstname>
   <sref:lastname>Picasso</sref:lastname>
   <sref:paints>
      <sref:Painting rdf:about="#Item1">
         <sref:title>Guernica</sref:title>
```

```
         <sref:exhibited>
            <sref:Museum rdf:about="#Item2">
            <sref:name>Reina Sofia</sref:name>
            </sref:Museum>
         </sref:exhibited>
      </sref:Painting>
   </sref:paints>
```

Τέταρτη Επανάληψη

imapping: []
writtenClasses: [Painter,Painting,Museum,Style]
classesToWrite: []
classProperties: [[Painter,[paints,Painting],[hasstyle,Style]],[Painting,[exhibited,Museum]]]

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:sref="file:./rdf/peerSchema.rdfs#">

<sref:Painter rdf:about="#Item0">
   <sref:firstname>Pablo</sref:firstname>
   <sref:lastname>Picasso</sref:lastname>
   <sref:paints>
      <sref:Painting rdf:about="#Item1">
         <sref:title>Guernica</sref:title>
         <sref:exhibited>
            <sref:Museum rdf:about="#Item2">
            <sref:name>Reina Sofia</sref:name>
            </sref:Museum>
         </sref:exhibited>
      </sref:Painting>
   </sref:paints>
   <sref:hasstyle>
      <sref:Style rdf:about="#Item3">
         <sref:stitle>Cubist</sref:stitle>
      </sref:Style>
   </sref:hasstyle>
```

```
</sref:Painter>
</rdf:RDF>
```

Αλγόριθμος ελέγχου ικανοποιησιμότητας ερώτησης

Ο αλγόριθμος αυτός ελέγχει κάθε ερώτηση που φτάνει στον κόμβο και αποφαίνεται για το αν η ερώτηση αυτή είναι ικανοποιήσιμη. Δηλαδή ελέγχει αν μπορεί να απαντηθεί η συγκεκριμένη ερώτηση παίρνοντας υπόψη του το σχήμα του κόμβου.

Σημειώνεται εδώ ότι οι ερωτήσεις που στέλνουν και λαμβάνουν οι κόμβοι δεν είναι σε μορφή RQL, αλλά σε μια άλλη μορφή που έχουμε ορίσει εμείς για διευκόλυνση της υλοποίησης. Η μορφή αυτή είναι ως εξής:
Κκλάση1(Κυριολεκτικό1, Κυριολεκτικό2,...).ιδιότητα1.Κλάση2(Κυριολεκτικό3,...)...
Ο κόμβος αφού επεξεργαστεί την ερώτηση σε αυτή τη μορφή, τη μετατρέπει στη συνέχεια σε RQL για να αποτιμηθεί.

Είσοδος: Η ερώτηση στη μορφή που λαμβάνεται από το δίκτυο και το σχήμα του κόμβου.

Έξοδος: Απόφαση για την ικανοποιησιμότητα ή μη της ερώτησης.

Αλγόριθμος:

```
Βρες το σύνολο των κλάσεων που συμμετέχουν στην ερώτηση και πρόσθεσέ τες
στο διάνυσμα queryClasses
Βρες το σύνολο των κυριολεκτικών που συμμετέχουν στην ερώτηση και προσθεσέ
τα στο διανυσμα queryLiterals
Βρες όλες τις κλάσεις από το σχήμα του κόμβου και πρόσθεσέ τες στο διάνυσμα
rdfsClass
Για κάθε κλάση του rdfsClass
   Βρες τα κυριολεκτικά και πρόσθεσέ τα στο διάνυσμα rdfsLits
Αν το rdfsClass περιέχει όλα τα στοιχεία του queryClasses και το
rdfsLits όλα τα στοιχεία του queryLiterals
   Επέστρεψε ικανοποιήσιμη
Αλλιως
   Επέστρεψε μη ικανοποιήσιμη
```

Παράδειγμα

Έστω ο κόμβος 1 ο οποίος έχει το RDF σχήμα που φαίνεται στο Σχήμα 5.2 και ο κόμβος 2 ο οποίος έχει το σχήμα που φαίνεται στο Σχήμα 5.3. Έστω επίσης η ερώτηση:
Painter(firstname,lastname).paints.Painting(title)

Για την ερώτηση αυτή τα διανύσματα queryClasses και queryLiterals έχουν τις εξής τιμές:

queryClasses: [Painter,Painting]
queryLiterals: [firstname,lastname,title]

Για τον κόμβο 1 τα διανύσματα rdfsClass και rdfsLits έχουν τις εξής τιμές:

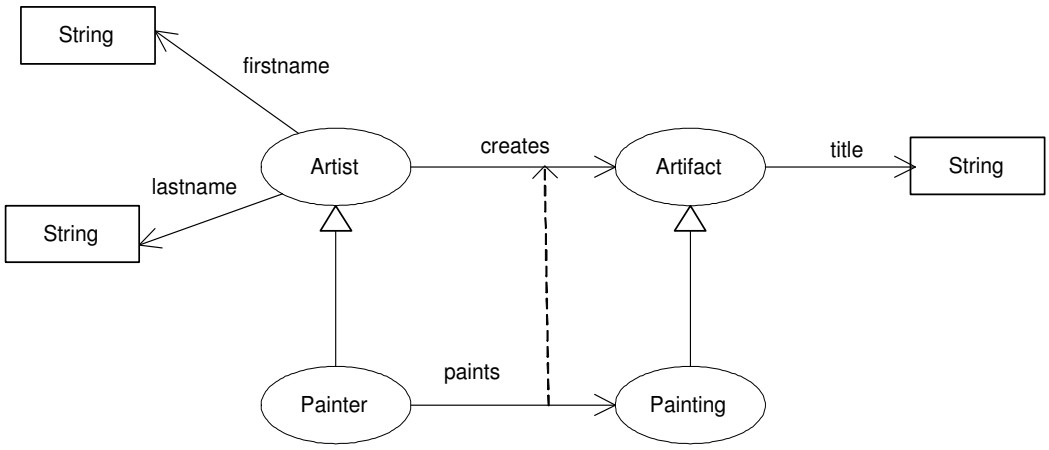
rdfsClass: [Artist,Artifact,Painter,Painting]
rdfsLits: [firstname,lastname,title]

Παρατηρούμε ότι το rdfsClass περιέχει όλα τα στοιχεία του queryClasses και το rdfsLits όλα τα στοιχεία του queryLiterals. Άρα η ερώτηση αυτή είναι ικανοποιήσιμη για τον κόμβο 1.

Για τον κόμβο 2 τα διανύσματα rdfsClass και rdfsLits έχουν τις εξής τιμές:

rdfsClass: [Artist,Artifact,Sculptor,Sculpture]
rdfsLits: [firstname,lastname,title]

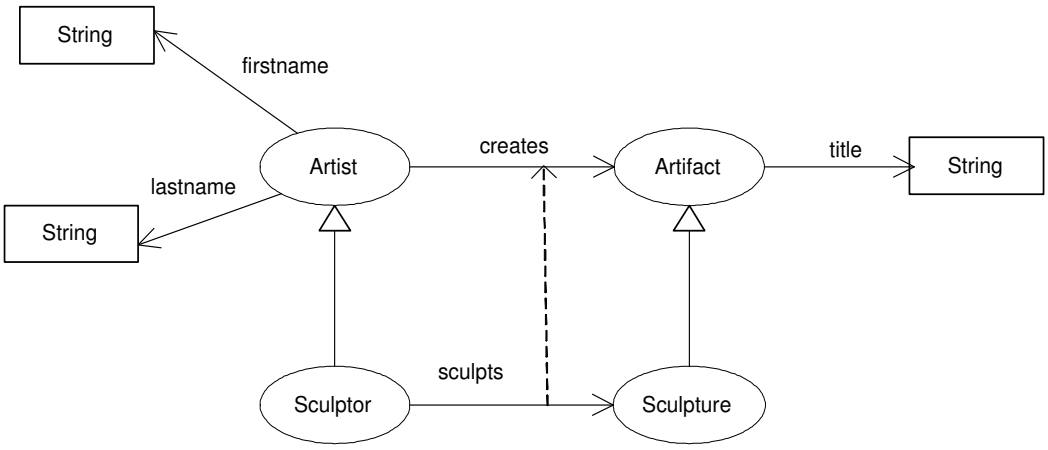
Παρατηρούμε ότι το rdfsClass δεν περιέχει όλα τα στοιχεία του queryClasses ενώ το rdfsLits περιέχει όλα τα στοιχεία του queryLiterals. Άρα η ερώτηση αυτή δεν είναι ικανοποιήσιμη για τον κόμβο 2.



Σχήμα 5.2: Σχήμα κόμβου 1

Αλγόριθμος μετατροπής ερώτησης

Ο αλγόριθμος αυτός μετατρέπει μια μη ικανοποιήσιμη ερώτηση σε ικανοποιήσιμη σύμφωνα με το σχήμα του κόμβου. Επίσης μετατρέπει τις ικανοποιήσιμες ερωτήσεις σε ερωτήσεις RQL για να μπορέσουν να αποτιμηθούν.



Σχήμα 5.3: Σχήμα κόμβου 2

Είσοδος: Το διάνυσμα queryFragments που περιέχει τα κομμάτια από τα οποία αποτελείται η ερώτηση και έχει τη μορφή: [[Κλάση1,[Κυριολεκτικό1,Κυριολεκτικό2..],[Ιδιότητα],[Κλάση2 ,[Κυριολεκτικό3...]],...]. Επίσης η επιλογή επίτρεψης μετασχηματισμού (EEM) και το καθολικό σχήμα.

Έξοδος: Μετασχηματισμένη ερώτηση.

Αλγόριθμος:

```
Αν η EEM είναι αληθής
   Μετέτρεψε την ερώτηση σε ερώτηση RQL
Αλλιώς
   Για το πρώτο στοιχείο του queryFragments (το οποίο αποτελεί κλάση)
   Βρες την κοντινότερη υπερκλάση από το καθολικό σχήμα που περιέχεται
   στο τοπικό σχήμα του κόμβου
   Αν υπάρχει τέτοια κλάση τότε
      Αντικατέστησε το πρώτο στοιχείο του queryFragments με την κλάση αυτή
      Βρες από το καθολικό σχήμα την τομή των άμεσων ιδιοτήτων της κλάσης
      αυτής με την ένωση του δεύτερου στοιχείου του queryFragments
      (που αποτελεί ιδιότητα),με τις υπεριδιότητες αυτού.
      Αντικατέστησε το δεύτερο στοιχείο του queryFragments με το αποτέλεσμα
      της τομής.

      Για κάθε στοιχείο του queryFragments ξεκινώντας από το τρίτο στοιχείο
         Αν πρόκειται για κλάση, βρες από το καθολικό σχήμα το πεδίο τιμών
         του προηγούμενου στοιχείου και την αντικατέστησέ τη στο διανυσμα
         queryFragments με αυτό.
         Αν πρόκειται για ιδιότητα,
```

```
        βρες από το καθολικό σχήμα την τομή των άμεσων ιδιοτήτων
        του προηγούμενου στοιχείου αυτής με την ένωση της ιδιότητας
        αυτής με τις υπεριδιότητές της.
        Αντικατέστησε την ιδιότητα στο διάνυσμα queryFragments
        με το παραπάνω αποτέλεσμα.

Αλλιώς δε γίνεται μετατροπή της ερώτησης.
```

Παράδειγμα

Έστω ότι στέλνουμε την ερώτηση Painter(firstname,lastname).paints.Painting(title) στον κόμβο 2 ο οποίος έχει το σχήμα που φαίνεται στο Σχήμα 5.3. Η ερώτηση αυτή όπως αποδείξαμε στο προηγούμενο παράδειγμα δεν είναι ικανοποιήσιμη για τον κόμβο 2.Έστω επίσης ότι το καθολικό σχήμα είναι αυτό που φαίνεται στο Σχήμα 5.4.

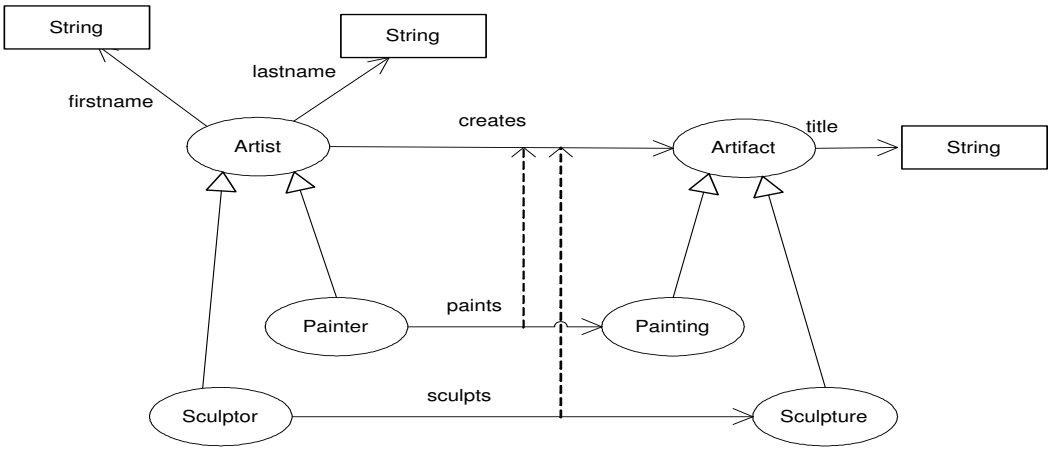
Το διάνυσμα queryFragments για αυτή την ερώτηση είναι το εξής:

queryFragments: [[Painter,[firstname,lastname]],[paints],[Painting,[title]]]

Σύμφωνα με τον αλγόριθμο το διανυσμα αυτό μετατρέπεται στο:

queryFragments: [[Artist,[firstname,lastname]],[creates],[Artifact,[title]]]

Έτσι τελικά η ερώτηση που απαντάει ο κόμβος 2 είναι η: Artist(firstname,lastname).creates. Artifact(title) η οποία είναι ικανοποιήσιμη.



Σχήμα 5.4: Καθολικό Σχήμα

5.1.2 Επικοινωνία κόμβων

Για τη δημιουργία του συστήματος ομότιμων κόμβων και την επικοινωνία μεταξύ των κόμβων χρησιμοποιήθηκε η πλατφόρμα Jxta η οποία περιγράφεται αναλυτικά στην Ενότητα 5.2.3. Για να επικοινωνήσουν οι κόμβοι χρησιμοποιούν ένα μηχανισμό του Jxta ο οποίος μοιάζει με αυτό τον sockets. Συγκεκριμένα όταν ένας κόμβος εισέρχεται στο σύστημα δημιουργεί ένα input pipe, όπως λέγεται, μέσω του οποίου περιμένει συνδέσεις από άλλους κόμβους. Τα μηνύματα που ανταλλάσσονται μέσω αυτών των συνδέσεων είναι XML αρχεία. Για τις ανάγκες του συγκεκριμένου συστήματος ορίσαμε τέσσερα είδη τέτοιων μηνυμάτων: μηνύματα πληροφορίας κόμβου, μηνύματα τοπολογίας, μηνύματα ερωτήσεων και μηνύματα αρχείων.

Μηνύματα πληροφορίας κόμβου

Τα μηνύματα αυτά χρησιμοποιούνται για την αποστολή των στοιχείων του κάθε κόμβου όταν εισέρχεται στο σύστημα, στον κόμβο διαχειριστή. Η μορφή ενός τέτοιου μηνύματος σε XML φαίνεται παρακάτω.

```
<?xml version="1.0"?>
<message>
   <Type>InfoMessage</Type>
   <PeeerInfo>...</PeerInfo>
</message>
```

Παρατηρούμε ότι το μήνυμα αποτελείται από τα εξής πεδία:

- Type: Το πεδίο αυτό δηλώνει τον τύπο του μηνύματος και υπάρχει σε όλα τα είδη μηνυμάτων. Στη συγκεκριμένη περίπτωση έχει τιμή InfoMessage.
- PeeerInfo: Το πεδίο αυτό περιέχει σε μορφή αλφαριθμητικού τις πληροφορίες του κόμβου, δηλαδή το όνομά του, τον αριθμό ταυτοποίησης του ID και το όνομα της βάσης στην οποία αποθηκεύει τις RDF περιγραφές.

Μηνύματα τοπολογίας

Τα μηνύματα αυτά στέλνονται από τον κόμβο διαχειριστή στους υπόλοιπους κόμβους για τους ενημερώσει για την τοπολογία του δικτύου δηλαδή για το ποιοι κόμβοι είναι γειτονικοί τους. Όταν ο κόμβος διαχειριστής λάβει μήνυμα πληροφορίας από κάποιον νεοεισερχόμενο κόμβο, τότε του στέλνει μήνυμα τοπολογίας. Η μορφή ενός τέτοιου μηνύματος σε XML φαίνεται παρακάτω.

```
<?xml version="1.0"?>
<message>
   <Type>TopologyMessage</Type>
   <Neighboors>...</Neighboors>
</message>
```

Παρατηρούμε ότι το μήνυμα αποτελείται από τα εξής πεδία:

- Type: Το πεδίο αυτό δηλώνει τον τύπο του μηνύματος και υπάρχει σε όλα τα είδη μηνυμάτων. Στη συγκεκριμένη περίπτωση έχει τιμή TopologyMessage.
- Neighboors: Το πεδίο αυτό περιέχει σε μορφή αλφαριθμητικού τα IDs όλων των κόμβων που αποτελούν γείτονές του.

Μηνύματα ερωτήσεων

Τα μηνύματα ερωτήσεων χρησιμοποιούνται για την αποστολή ερωτήσεων από τον ένα κόμβο στον άλλο. Ο κόμβος ο οποίος διατυπώνει την ερώτηση τη στέλνει στους γείτονές του και αυτοί στους δικούς τους γείτονες και έτσι η ερώτηση διαδίδεται μέχρι να επισκεφθεί το μέγιστο αριθμό κόμβων που έχει ορίσει ο χρήστης. Η μορφή ενός μηνύματος ερώτησης σε XML φαίνεται παρακάτω.

```
<?xml version="1.0"?>
<message>
   <Type>QueryMessage</Type>
   <Query>...</Query>
   <TransmitterID>...</TransmitterID>
   <TransmitterName>...</TransmitterName>
   <MaxHops>...</MaxHops>
   <Hop>...</Hop>
   <Transform>...</Transform>
</message>
```

Παρατηρούμε ότι το μήνυμα αποτελείται από τα εξής πεδία:

- Type: Το πεδίο αυτό δηλώνει τον τύπο του μηνύματος και έχει τιμή QueryMessage.
- TransmitterID: Το πεδίο αυτό περιέχει το ID του αρχικού κόμβου που διατύπωσε την ερώτηση.
- TransmitterName: Το πεδίο αυτό περιέχει το όνομα του αρχικού κόμβου που διατύπωσε την ερώτηση.
- MaxHops: Το πεδίο αυτό περιέχει το μέγιστο αριθμό κόμβων που μπορεί να επισκεφθεί η ερώτηση και έχει οριστεί από το χρήστη πριν την αποστολή της ερώτησης.
- Hop: Το πεδίο αυτό περιέχει τον αριθμό των κόμβων που έχει επισκεφθεί η ερώτηση μέχρι τώρα.
- Transform: Το πεδίο αυτό περιέχει το επίπεδο μετασχηματισμού που έχει ορίσει ο χρήστης για την αποστολή της ερώτησης. Έχει τιμή "Level 0" όταν η ερώτηση δεν έχει επίτρεψη μετασχηματισμού και "Level 1" όταν έχει.

Μηνύματα αρχείων

Τα μηνύματα αρχείων χρησιμοποιούνται για την αποστολή αρχείων μεταξύ των κόμβων. Τα αρχεία αυτά μπορεί να είναι αρχεία που περιέχουν την απάντηση μιας ερώτησης που στέλνει ένας κόμβος στον αρχικό κόμβο που έκανε την ερώτηση. Μπορεί επίσης να είναι αρχεία που περιέχουν την RDF περιγραφή του καθολικού σχήματος. Αυτά τα αρχεία τα στέλνει ο κόμβος διαχειριστής σε κάθε νεοεισερχόμενο κόμβο. Η μορφή ενός μηνύματος αρχείου σε XML φαίνεται παρακάτω.

```
<?xml version="1.0"?>
<message>
   <Type>FileMessage</Type>
   <TransmitterID>...</TransmitterID>
   <TransmitterName>...</TransmitterName>
   <RDFfile>...</RDFfile>
</message>
```

Παρατηρούμε ότι το μήνυμα αποτελείται από τα εξής πεδία:

- Type: Το πεδίο αυτό δηλώνει τον τύπο του μηνύματος και έχει τιμή FileMessage.
- TransmitterID: Το πεδίο αυτό περιέχει το ID του κόμβου αποστολέα.
- TransmitterName: Το πεδίο αυτό περιέχει το όνομα του κόμβου αποστολέα.
- RDFfile: Το πεδίο αυτό περιέχει το αρχείο που αποστέλεται σε stream από bytes.

5.2 Πλατφόρμες και προγραμματιστικά εργαλεία

5.2.1 Γλώσσα Προγραμματισμού

Το σύστημα υλοποιήθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού Java. Η Java επιλέχθηκε γιατί το σύστημα RDFSculpt που χρησιμοποιήθηκε ως σύστημα για τη δημιουργία σχήματος ήταν υλοποιημένο σε Java όπως επίσης και η πλατφόρμα δημιουργίας του δικτύου ομότιμων κόμβων και οι βιβλιοθήκες διαχείρισης RDF.

Για την ανάπτυξη του κώδικα σε Java χρησιμοποιήθηκε το περιβάλλον ανάπτυξης Eclipse.

5.2.2 Λειτουργικό σύστημα

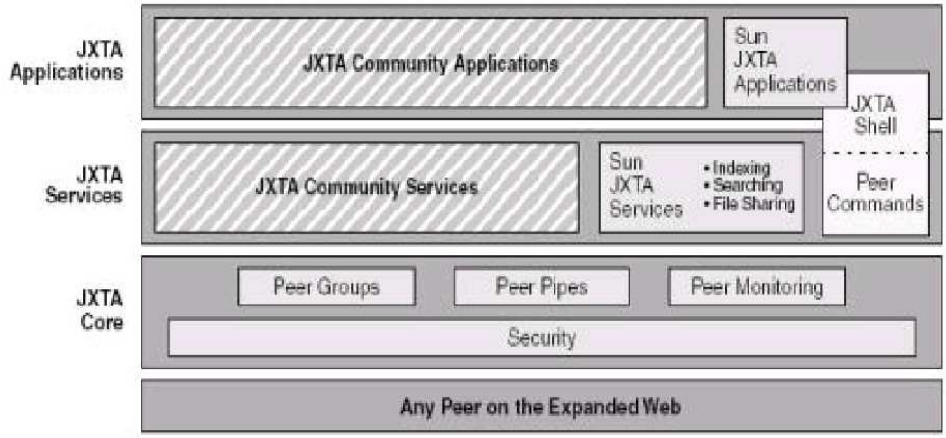
Το σύστημα αναπτύχθηκε σε λειτουργικό σύστημα Suse Linux. Η επιλογή αυτή βασίστηκε κυρίως στο ότι οι βιβλιοθήκες που χρησιμοποιήθηκαν για τη διαχείριση των RDF σχημάτων και δεδομένων έχουν υλοποιηθεί για να λειτουργούν μόνο σε περιβάλλον Linux.

5.2.3 Πλατφόρμα υλοποίησης δικτύου ομότιμων κόμβων

Για την υλοποίηση του δικτύου ομότιμων κόμβων, χρησιμοποιήθηκε η πλατφόρμα Jxta [6]. Το Jxta είναι ένα σύνολο ανοικτών, γενικευμένων πρωτοκόλλων ομότιμων κόμβων, που επιτρέπει σε όλες τις δικτυακές συσκευές να επικοινωνούν και να συνεργάζονται σαν ομότιμοι κόμβοι. Τα πρωτόκολλα αυτά είναι ανεξάρτητα της γλώσσας προγραμματισμού, της πλατφόρμας ανάπτυξης καθώς και των πρωτοκόλλων μεταφοράς.

Σκοπός της ανάπτυξης των παραπάνω πρωτοκόλλων είναι η προτυποποίηση του τρόπου με τον οποίο οι κόμβοι:

- Ανακαλύπτουν ο ένας τον άλλο.
- Αυτοοργανώνονται σε ομάδες κόμβων (peergroups)
- Διαφημίζουν και ανακαλύπτουν τις υπηρεσίες του δικτύου.
- Επικοινωνούν μεταξύ τους.
- Παρακολουθούν ο ένας τη λειτουργία του άλλου.



Σχήμα 5.5: Αρχιτεκτονική του Jxta

Η αρχιτεκτονική του JXTA η οποία φαίνεται στο Σχήμα 5.5 είναι οργανωμένη σε τρία επίπεδα. Το κατώτερο επίπεδο που ονομάζεται πυρήνας του Jxta (Jxta Core), περιλαμβάνει τα δομικά στοιχεία που χρειάζονται οι βασικοί μηχανισμοί των εφαρμογών ομότιμων κόμβων, όπως η δημιουργία κόμβων και ομάδων κόμβων (Peer Groups), ο μηχανισμός επικοινωνίας (PeerPipes) η αναζήτηση και ο έλεγχος άλλων κόμβων (Peer Monitoring) και οι βασικοί μηχανισμοί ασφάλειας (Security). Το επόμενο επίπεδο είναι το επίπεδο των υπηρεσιών (Jxta Services). Σε αυτό το επίπεδο περιλαμβάνονται υπηρεσίες που μπορεί να μην είναι εντελώς απαραίτητες για τη λειτουργία ενός συστήματος ομότιμων κόμβων, είναι όμως επιθυμητές. Παραδείγματα τέτοιων υπηρεσιών είναι υπηρεσίες δεικτοδότησης (indexing), αναζήτησης (searching) και ανταλλαγής αρχείων (file shearing). Το τελευταίο επίπεδο είναι το

επίπεδο των εφαρμογών (Jxta Applications). Σε αυτό το επίπεδο περιλαμβάνονται εφαρμογές όπως ανταλλαγή μηνυμάτων (instant messaging), ανταλλαγή αρχείων (file sharing) κ.α.

Το δίκτυο του Jxta αποτελείται από ένα σύνολο διασυνδεόμενων κόμβων (peers). Οι κόμβοι αυτοοργανώνονται σε ομάδες κόμβων (peer groups), που παρέχουν το ίδιο σύνολο υπηρεσιών.

Οι κόμβοι διαφημίζουν τις υπηρεσίες που προσφέρουν με XML documents που ονομάζονται διαφημίσεις (advertisements). Οι διαφημίσεις δίνουν τη δυνατότητα στους άλλους κόμβους του δικτύου να μάθουν πως να συνδεθούν και να αλληλεπιδράσουν με τις υπηρεσίες του κόμβου.

Στο Jxta χρησιμοποιούνται τέσσερα είδη κόμβων. Οι απλοί συνοριακοί κόμβοι minimal edge peers οι οποίοι μπορούν απλά να στέλνουν και να λαμβάνουν μηνύματα. Οι πλήρεις συνοριακοί κόμβοι full featured edge peers οι οποίοι στέλνουν και λαμβάνουν μηνύματα, αλλά μπορούν επίσης και να αποθηκεύουν τις διαφημίσεις. Οι Rendezvous κόμβοι οι οποίοι διαφοροποιούνται από τους υπόλοιπους στο ότι προωθούν στους κόμβους της ομάδας στην οποία ανήκουν, αιτήσεις ανακάλυψης υπηρεσιών discovery requests. Όταν ένας νέος κόμβος εισέρχεται στη ομάδα κόμβων, συνδέεται με έναν Rendezvous κόμβο και μαθαίνει τις υπηρεσίες των άλλων κόμβων της ομάδας. Τέλος υπάρχουν και οι Relay κόμβοι οι οποίοι διατηρούν πληροφορίες δρομολόγησης και δρομολογούν τα μηνύματα στους υπόλοιπου κόμβους.

Οι κόμβοι του Jxta χρησιμοποιούν σωληνώσεις (pipes) για την ανταλλαγή μηνυμάτων μεταξύ τους. Οι σωληνώσεις αποτελούν έναν ασύγχρονο μηχανισμό μεταφοράς μηνυμάτων. Τα μηνύματα είναι απλά XML documents τα οποία περιέχουν πληροφορίες δρομολόγησης και πιστοποίησης καθώς και το μήνυμα αυτό καθαυτό που θέλουν να ανταλλάξουν οι κόμβοι μεταξύ τους. Η επικοινωνία των κόμβων, βασίζεται επίσης στο μηχανισμό ονοματοδοσίας, όπου κάθε κόμβος χαρακτηρίζεται από ένα μοναδικό αριθμό ταυτοποίησης Peer ID.

5.2.4 Εργαλεία διαχείρισης RDF

Ως υποσύστημα διεπαφής της βάσης RDF χρησιμοποιήθηκε το σύστημα RDFSuite [2], το οποίο παρέχει τα κατάλληλα εργαλεία για την αποθήκευση RDF σχημάτων και δεδομένων, καθώς και για την αποτίμηση ερωτήσεων πάνω σ΄ αυτά.

Το RDFSuite αποτελείται από τρία επιμέρους υποσυστήματα:

- Ένα συντακτικό αναλυτή για RDF περιγραφές (Validating RDF Parser) ο οποίος αναλύει τις RDF περιγραφές και εξετάζει αν είναι συντακτικά ορθές σύμφωνα με τη σημασιολογία του μοντέλου RDF.

- Μια ειδική σχεσιακή βάση για την αποθήκευση RDF σχημάτων και δεδομένων (RDF Schema Specific DataBase). Η βάση αυτή είναι PostgreSQL.

- Ένα μεταφραστή για τη γλώσσα RQL (RQL intepreter) ο οποίος αρχικά αναλύει συντακτικά τις ερωτήσεις και στη συνέχεια τις αποτιμά με βάση τις RDF περιγραφές που υπάρχουν στη βάση. Επειδή όμως η βάση είναι σχεσιακή για να αποτιμήσει τις ερωτήσεις τις μετατρέπει πρώτα σε ερωτήσεις SQL3.

5.3 Περιγραφή κλάσεων

Στην ενότητα αυτή δίνεται μια σύντομη περιγραφή των κλάσεων, των πεδίων και των μεθόδων που τις απαρτίζουν.

5.3.1 public class FirstUi

Η κλάση αυτή κατασκευάζει την οθόνη εισαγωγής του χρήστη στο σύστημα.

Πεδία

- `private GridBagLayout blayout`
  Το layout για όλα τα Panel.
- `private GridBagConstraints con`
  Τα constraints για το layout.
- `private Icon arrowR`
  Εικονίδιο για το κουμπί Next.
- `private FileOutputStream outputFile`
  Αρχείο στο οποίο γράφονται τα ονόματα των βάσεων που εισάγει ο χρήστης.
- `private PrintStream out`
  Stream για τύπωμα στο αρχείο εξόδου.
- `JFrame entryframe`
  Το frame της οθόνης αυτής.

Μέθοδοι

- `public FirstUi()`
  Ο κατασκευαστής της κλάσης ο οποίος καλεί την createEntryFrame().
- `private void createEntryFrame()`
  Μέθοδος που κατασκευάζει το ενφραμε.
- `private void addComponent(Component component, Container container, int row, int column, int width, int height)`
  Μέθοδος που τοποθετεί το component πάνω στον container την επιθυμητή θέση.

5.3.2 public class ui

Η κλάση αυτή κατασκευάζει την αρχική οθόνη του συστήματος.

Πεδία

- `private String dbName`
  Το όνομα της σχεσιακής βάσης του κόμβου.

- `private String globalSchema`
  Το καθολικό σχήμα.

- `private GridBagLayout blayout`
  Το layout για όλα τα Panel.

- `private GridBagConstraints con`
  Τα constraints για το layout.

- `private JTextArea trace`
  Η TextArea στην οποία τυπώνονται τα μηνύματα των κόμβων.

Μέθοδοι

- `public ui()`
  Κατασκευαστής της κλάσης στον οποίο γίνεται η αρχικοποίηση του κόμβου.

- `private void createFrame()`
  Μέθοδος η οποία δημιουργεί την αρχική οθόνη.

5.3.3 public class ui1

Η κλάση αυτή κατασκευάζει όλες τις διαπροσωπείες του συστήματος εκτός από την αρχική.

Πεδία

- `private GridBagLayout blayout`
  Το layout για όλα τα Panel.

- `private GridBagConstraints con`
  Τα constraints για το layout.

- `private Icon arrowR`
  Εικονίδιο για αριστερό βελάκι.

- `private Icon arrowL`
  Εικονίδιο για δεξί βελάκι.

- `private Icon saveIcon`
  Εικονίδιο για σώσιμο στη βάση.

- `private Icon upArrow`
  Εικονίδιο για επάνω βελάκι.

- `private String selectedTable`
  Επιλεγμένος πίνακας της σχεσιακής βάσης.

- `private String query`
  String που περιέχει το SQL ερώτημα.

- `private String select`
  Το Select πεδίο του SQL ερωτήματος.

- `private String where`
  Το Where πεδίο του SQL ερωτήματος.

- `private String peerDBName`
  Το όνομα της RDF βάσης του κόμβου.

- `private String schemaSelected`
  Επιλεγμένο σχήμα για εισαγωγή δεδομένων.

- `private String querySchema`
  Το σχήμα με βάση το οποίο δημιουργείται η ερώτηση.

- `private String maxhops`
  Ο μέγιστος αριθμός κόμβων για απάντηση της ερώτησης.

- `private String transform`
  Η επιλογή για το επίπεδο μετασχηματισμού μιας ερώτησης.

- `private String dataDb`
  Η σχεσιακή βάση του κόμβου.

- `private Connect c`
  Η μεταβλητή σύνδεσης με τη βάση.

- `private PeerFunctions pf`
  Στιγμιότυπο της κλάσης PeerFunctions.

- `private RQLQuery rquery`
  Στιγμιότυπο της κλάσης RQLQuery.

- `private QueryFunctions qf`
  Στιγμιότυπο της κλάσης QueryFunctions.

- `private PrintStream out`
  Το αρχείο που περιέχει τα ονόματα των βάσεων.

- `private ResultSet rs`
  Το αποτέλεσμα του SQL ερωτήματος.

- `private Vector columnVector`
  Διάνυσμα με τα πεδία που συμμετείχαν στο ερώτημα SQL.
- `private Vector tablesVector`
  Διάνυσμα που περιέχει τους πίνακες της σχεσιακής βάσης.
- `private Vector columnsSelected`
  Τα επιλεγμένα πεδία.
- `private Vector selectedTables`
  Οι επιλεγμένοι πίνακες.
- `private Vector rdfsClasses`
  Οι κλάσεις του RDF σχήματος.
- `private Vector mappingVector`
  Το διάνυσμα με τις αντιστοιχίσεις.
- `private Vector literalVector`
  Το διάνυσμα με τα κυριολεκτικά.
- `private Vector peerVector`
  Το διάνυσμα που περιέχει τα ονόματα και τα IDs των κόμβων που απάντησαν.
- `private Vector peerNames`
  Το διάνυσμα που περιέχει τα ονόματα των κόμβων που απάντησαν.
- `private Vector classVector`
  Διάνυσμα με τις κλάσεις του σχήματος.
- `private Vector propertyVector`
  Διάνυσμα με τις ιδιότητες του σχήματος.
- `private Vector pathVector`
  Διάνυσμα με το μονοπάτι που δημιουργεί ο χρήστης όταν κατασκευάζει μια ερώτηση.

Μέθοδοι

- `public ui1(RQLQuery rquery,PeerFunctions pf)`
  Κατασκευαστής που αρχικοποιεί τα πεδία με τις τιμές των παραμέτρων.
- `public void createSelectSchemaFrame()`
  Μέθοδος που κατασκευάζει την οθόνη για την επιλογή σχήματος.
- `private void createSqlQueryFrame()`
  Μέθοδος που κατασκευάζει την οθόνη για την κατασκευή του SQL ερωτήματος.
- `private void createMappingFrame()`
  Μέθοδος που κατασκευάζει την οθόνη για τη δημιουργία των αντιστοιχίσεων.

- `public void createSelectSchemaFrame2()`
  Μέθοδος που κατασκευάζει την οθόνη για την επιλογή σχήματος.

- `public void createQueryFrame()`
  Μέθοδος που κατασκευάζει την οθόνη για την κατασκευή του RQL ερωτήματος.

- `private String printPathVector(Vector v)`
  Μέθοδος που χρησιμεύει στην προηγούμενη για την εμφάνιση του μονοπατιού που δημιουργεί ο χρήστης πάνω στο γράφο.

- `public void createSettingsFrame()`
  Μέθοδος που κατασκευάζει την οθόνη για την εισαγωγή των παραμέτρων της ερώτησης.

- `private void addComponent(Component component, Container container, int row, int column, int width, int height)`
  Μέθοδος που τοποθετεί το component πάνω στον container την επιθυμητή θέση.

5.3.4 public class InsertData

Η κλάση αυτή υλοποιεί τον αλγόριθμο εισαγωγής δεδομένων.

Πεδία

- `private FileOutputStream outputFile`
  Το αρχείο στο οποίο θα γραφτεί το RDF/XML.

- `private PrintStream out`
  Stream για τύπωμα στο αρχείο εξόδου.

- `private String rdfsname`
  Το όνομα του αρχείου που περιέχει το RDF σχήμα του κόμβου.

- `private String rdfname`
  Το όνομα του αρχείου RDF/XML.

- `private Vector mapping`
  Το διάνυσμα που περιέχει τις αντιστοιχίσεις χαρακτηριστικών-κυριολεκτικών κλάσεων.

- `private ResultSet rs`
  Το ResultSet που παράχθηκε από το SQL query.

- `private QueryFunctions d`
  Στιγμιότυπο της κλάσης QueryFunctions.

Μέθοδοι

- `public InsertData(Vector mapping,String rdfsname,String rdfname, ResultSet rs,QueryFunctions d)`
  Κατασκευαστής που αρχικοποιεί τα πεδία με τις τιμές των παραμέτρων.
- `public Vector groupByClass(Vector m)`
  Μέθοδος που ομαδοποιεί ανά κλάση το διάνυσμα m.
- `public void writeToFile()`
  Μέθοδος που κατασκευάζει το αρχείο RDF/XML.
- `String getMappingClass(Vector m)`
  Μέθοδος που επιστρέφει ένα συγκεκριμένο στοιχείο του m.
- `Vector getProperties(String mclass)`
  Μέθοδος που βρίσκει τις περισσότερο ειδικές ιδιότητες που εφαρμόζονται σε μια κλάση.

5.3.5 public class StoreDataToDB

Η κλάση αυτή αποθηκεύει το αρχείο RDF/XML που περιέχει τα δεδομένα στην RDF βάση.

Πεδία

- `private SpecReprDBStorage storage`
  Αντικείμενο της κλάσης SpecReprDBStorage του RSSDB API.
- `private String dbname`
  Το όνομα της βάσης.
- `private String filename`
  Το όνομα του RDF/XML αρχείου.

Μέθοδοι

- `public StoreDataToDB(String dbname, String filename)`
  Κατασκευαστής για για αρχικοποίηση της βάσης.
- `public void store()`
  Μέθοδος που αποθηκεύει το αρχείο.

5.3.6 public class QueryProcessing

Η κλάση αυτή περιέχει τις μεθόδους που αφορούν την επεξεργασία ερωτήσεων.

Πεδία

- `private RQLQuery rquery`
  Αντικείμενο της κλάσης RQLQuery του RDFSculpt η οποία κάνει της απαραίτητες αρχικοποιήσεις και παρέχει μια μέθοδο που αποτιμά το RQL ερώτημα.

- `private Vector queryClasses`
  Διάνυσμα με τις κλάσεις που συμμετέχουν στο path expression του ερωτήματος.
- `private Vector queryLiterals`
  Διάνυσμα με τα κυριολεκτικά που συμμετέχουν στο path expression του ερωτήματος.
- `private Vector queryProperties`
  Διάνυσμα με τις ιδιότητες που συμμετέχουν στο path expression του ερωτήματος.
- `private Vector queryFragments`
  Διάνυσμα που περιέχει τα τμήματα από τα οποία αποτελείται το path expression του ερωτήματος, στη μορφή [[Κλάση1,[Κυριολεκτικό1,Κυριολεκτικό2..],[Ιδιότητα],[Κλάση2 ,[]], ...].
- `private Vector rdfsClass`
  Διάνυσμα που περιέχει τις κλάσεις του RDF σχήματος του κόμβου.
- `private String globalSchema`
  Το καθολικό σχήμα.
- `private String peerSchema`
  Το σχήμα του κόμβου.
- `private QueryFunctions qf`
  Αντικείμενο της κλάσης QueryFunctions

Μέθοδοι

- `public QueryProcessing (RQLQuery rq)`
  Κατασκευαστής που αρχικοποιεί τα πεδία με τις τιμές των παραμέτρων.
- `public boolean checkSatisfiability(String query)`
  Μέθοδος που ελέγχει την ικανοποιησιμότητα της ερώτησης.
- `public String transform(String query,int level)`
  Συνάρτηση που μετατρέπει την μη ικανοποιήσιμη ερώτηση σε ικανοποιήσιμη.
- `public String constructQuery(Vector fragmentsVector)`
  Συνάρτηση που μετατρέπει το ερώτημα σε RQL ερώτηση.

5.3.7 public class PeerFunctions

Η κλάση αυτή κάνει τις αρχικοποιήσεις της πλατφόρμας Jxta που χρειάζονται και υλοποιεί τις συναρτήσεις για την επικοινωνία του κόμβου με τους υπόλοιπους κόμβους του δικτύου και τη διαχείριση των μηνυμάτων που δέχεται από αυτούς.

Πεδία

- `private static PeerGroup netpg`
  Η γενική ομάδα κόμβων NetPeerGroup του Jxta στην οποία ανήκουν όλοι οι κόμβοι.

- `private static PeerGroup rdfNet`
  Η ομάδα κόμβων του δικού μας συστήματος.

- `private static DiscoveryService disco`
  Η υπηρεσία αναζήτησης κόμβων του Jxta.

- `private static PipeService pipes`
  Η υπηρεσία σωληνώσεων του Jxta.

- `private static RendezVousService rdv`
  Η υπηρεσία κόμβου RendezVous του Jxta

- `private static PipeAdvertisement myAdv`
  Η διαφήμιση της σωλήνωσης με την οποία ο κόμβος δέχεται μηνύματα από τους υπόλοιπους κόμβους.

- `private static InputPipe myPipe`
  Η σωλήνωση με την οποία ο κόμβος δέχεται μηνύματα από τους υπόλοιπους κόμβους.

- `private Vector neighboors`
  Διάνυσμα που περιέχει τα IDs των γειτονικών κόμβων του κόμβου.

- `private static Vector peersReplied`
  Διάνυσμα που περιέχει τα ονόματα και τα IDs των κόμβων που απαντούν σε μια ερώτηση του συγκεκριμένου κόμβου.

- `private JTextArea trace`
  Το JTextArea για την εκτύπωση των μηνυμάτων που λαμβάνει και στέλνει ο κόμβος.

- `private static Vector rdfPeerAdvs`
  Διάνυσμα με τις διαφημίσεις των κόμβων του συστήματος.

- `static String groupURL`
  Το URL που ταυτοποιεί την ομάδα κόμβων του συστήματος.

- `private String myIdentity`
  String με το ID αυτού του κόμβου.

- `private static String dbName` Το όνομα της RDF βάσης.

- `private static RQLQuery rquery` Αντικείμενο της κλάσης RQLQuery του RDFSculpt η οποία κάνει της απαραίτητες αρχικοποιήσεις και παρέχει μια μέθοδο που αποτιμά το RQL ερώτημα.

- `private static String peerSchema`
  Το όνομα του σχήματος του κόμβου.

Μέθοδοι

- `public PeerFunctions(String dbNam, JTextArea trace)`
  Κατασκευαστής που αρχικοποιεί τα πεδία με τις τιμές των παραμέτρων και καλεί τη μέθοδο startJxta για να ενεργοποιηθεί ο κόμβος.

- `private void startJxta()`
  Η μέθοδος του Jxta η οποία ενεργοποιεί τον κόμβο και καλεί τις μεθόδους joinRdfNet και run.

- `public void joinRdfNet()`
  Μέθοδος με την οποία ο κόμβος ανακαλύπτει την ομάδα κόμβων του συστήματος RDFNet και γίνεται μέλος της.

- `public void run()`
  Μέθοδος με την οποία ο κόμβος δημιουργεί τη σωλήνωση για να δέχεται μηνύματα και τη διαφημίζει στους άλλους κόμβους για να επικοινωνούν μαζί του.

- `public void discoverRdfPeer(String peername)`
  Μέθοδος με την οποία ο κόμβος ανακαλύπτει έναν άλλο κόμβο που βρίσκεται στην ίδια ομάδα με αυτόν με βάση το ID του.

- `public OutputPipe connectToRdfPeer(String peername)`
  Μέθοδος η οποία δημιουργεί σωλήνωση εξόδου η οποία συνδέεται με τη σωλήνωση εισόδου ενός συγκεκριμένου κόμβου. Στην ουσία συνδέει τους δύο κόμβους για να στείλει ο ένας μήνυμα στον άλλο.

- `public void pipeMsgEvent(PipeMsgEvent ev)`
  Μέθοδος η οποία δέχεται ασύγχονα και διαχειρίζεται τα μηνύματα από τους άλλους κόμβους.

- `public void sendMsg(String messageString,String type,String peername)`
  Μέθοδος για την κατασκευή και αποστολή μηνύματος με τις πληροφορίες κόμβου στον κόμβο διαχειριστή.

- `public void sendFile(File file,String receiver,String transmitterName, String transmitterID)`
  Μέθοδος για την κατασκευή και αποστολή μηνύματος αρχείου.

- `public void sendQuery(String query,String transmitterID,String transmitterName,String receiver,String maxhops,String hop,String transform)`
  Μέθοδος για την κατασκευή και αποστολή μηνύματος ερώτησης.

- `public PeerGroupID mkGroupID()`
  Μέθοδος που δημιουργεί την ομάδα κόμβων.

- `public String getName()`
  Μέθοδος που επιστρέφει το όνομα του συγκεκριμένου κόμβου.

- `public String getID()`
  Μέθοδος που επιστρέφει το ID του συγκεκριμένου κόμβου.

- `public Vector getNeighboors()`
  Μέθοδος που επιστρέφει τους γείτονες του συγκεκριμένου κόμβου.

- `public Vector getPeersReplied()`
  Μέθοδος που επιστρέφει τους κόμβους που απάντησαν σε ένα ερώτημα.

- `public void clearPeerVector()`
  Μέθοδος που αδειάζει το διάνυσμα των κόμβων που απάντησαν σε μια ερώτηση.

5.3.8 public class QueryResult

Η κλάση αυτή μετατρέπει το αποτέλεσμα μιας RQL ερώτησης σε διάνυσμα για να είναι εύκολα επεξεργάσιμο.

Πεδία
Η κλάση αυτή δεν έχει πεδία.

Μέθοδοι

- `public QueryResult()`
  Ο κατασκευαστής της κλάσης.

- `public Vector getResult()`
  Η μέθοδος αυτή κάνει parse το αποτέλεσμα μιας RQL ερώτησης το οποίο είναι σε μορφή RDF/XML και το μετατρέπει σε διάνυσμα.

5.3.9 public class ReplyResult

Η κλάση αυτή μετατρέπει σε διάνυσμα την απάντηση που δέχεται ο κόμβος από κάποιον άλλο κόμβο, σε μια ερώτηση που είχε στείλει.

Πεδία
Η κλάση αυτή δεν έχει πεδία.

Μέθοδοι

- `public ReplyResult()`
  Ο κατασκευαστής της κλάσης.

- `getResult(String filename)`
  Η μέθοδος αυτή κάνει parse την απάντηση η οποία είναι σε μορφή RDF/XML και τη μετατρέπει σε διάνυσμα.

5.3.10 public class SPeerFunctions

Η κλάση αυτή κάνει τις αρχικοποιήσεις της πλατφόρμας Jxta που χρειάζονται και υλοποιεί τις συναρτήσεις για την επικοινωνία του κόμβου διαχειριστή με τους υπόλοιπους κόμβους του δικτύου και τη διαχείριση των μηνυμάτων που δέχεται από αυτούς.

Πεδία

- `private static PeerGroup netpg`
  Η γενική ομάδα κόμβων NetPeerGroup του Jxta στην οποία ανήκουν όλοι οι κόμβοι.

- `private static PeerGroup rdfNet`
  Η ομάδα κόμβων του δικού μας συστήματος.

- `private static DiscoveryService disco`
  Η υπηρεσία αναζήτησης κόμβων του Jxta.

- `private static PipeService pipes`
  Η υπηρεσία σωληνώσεων του Jxta.

- `private static RendezVousService rdv`
  Η υπηρεσία κόμβου RendezVous του Jxta

- `private static PipeAdvertisement myAdv`
  Η διαφήμιση της σωλήνωσης με την οποία ο κόμβος δέχεται μηνύματα από τους υπόλοιπους κόμβους.

- `private static InputPipe myPipe`
  Η σωλήνωση με την οποία ο κόμβος δέχεται μηνύματα από τους υπόλοιπους κόμβους.

- `private static Vector rdfPeerAdvs`
  Διάνυσμα με τις διαφημίσεις των κόμβων του συστήματος.

- `static String groupURL`
  Το URL που ταυτοποιεί την ομάδα κόμβων του συστήματος.

- `private String myIdentity`
  String με το ID αυτού του κόμβου.

Μέθοδοι

- `public SPeerFunctions()`
  Κατασκευαστής που δημιουργεί τη σύνδεση με τη βάση του κόμβου διαχειριστή.
- `public void joinRdfNet()`
  Μέθοδος με την οποία ο κόμβος ανακαλύπτει την ομάδα κόμβων του συστήματος RDFNet και γίνεται μέλος της.
- `public void setPeerPipe()`
  Μέθοδος με την οποία ο κόμβος δημιουργεί τη σωλήνωση για να δέχεται μηνύματα και τη διαφημίζει στους άλλους κόμβους για να επικοινωνούν μαζί του.
- `public void discoverRdfPeer(String peername)`
  Μέθοδος με την οποία ο κόμβος ανακαλύπτει έναν άλλο κόμβο που βρίσκεται στην ίδια ομάδα με αυτόν με βάση το ID του.
- `public OutputPipe connectToRdfPeer(String peername)`
  Μέθοδος η οποία δημιουργεί σωλήνωση εξόδου η οποία συνδέεται με τη σωλήνωση εισόδου ενός συγκεκριμένου κόμβου. Στην ουσία συνδέει τους δύο κόμβους για να στείλει ο ένας μήνυμα στον άλλο.
- `public void public void receiveMsg()`
  Μέθοδος η οποία δέχεται ασύγχονα και διαχειρίζεται τα μηνύματα από τους άλλους κόμβους.
- `public void sendMsg(String messageString,String type,String peername)`
  Μέθοδος για την κατασκευή και αποστολή μηνύματος με την τοπολογία του δικτύου.
- `public void sendFile(File file,String receiver,String transmitterName, String transmitterID)`
  Μέθοδος για την κατασκευή και αποστολή μηνύματος αρχείου.
- `public PeerGroupID mkGroupID()`
  Μέθοδος που δημιουργεί την ομάδα κόμβων.
- `public String getName()`
  Μέθοδος που επιστρέφει το όνομα του συγκεκριμένου κόμβου.
- `public String getID()`
  Μέθοδος που επιστρέφει το ID του συγκεκριμένου κόμβου.

5.3.11 public class SuperPeer

Η κλάση αυτή χρησιμοποιείται για την ενεργοποίηση του κόμβου διαχειριστή.

Πεδία

Η κλάση αυτή δεν έχει πεδία.

Μέθοδοι

- `private void startJxta()`
  Η μέθοδος του Jxta η οποία ενεργοποιεί τον κόμβο και καλεί τις μεθόδους joinRdfNet και setPeerPipe.

5.3.12 public class Connect

Κλάση η οποία χρησιμοποιείται για σύνδεση με τη βάση και εκτέλεση SQL ερωτημάτων.

Πεδία

- `private Connection conn`
  Η σύνδεση με τη βάση.
- `private ResultSet rs`
  Το αποτέλεσμα της ερώτησης.

Μέθοδοι

- `public Connect(String dbName)`
  Κατασκευαστής που υλοποιεί τις απαραίτητες αρχικοποιήσεις για τη σύνδεση με τη βάση.
- `ResultSet execute(String query)`
  Μέθοδος η οποία αποτιμά την ερώτηση και επιστρέφει το αποτέλεσμά της.

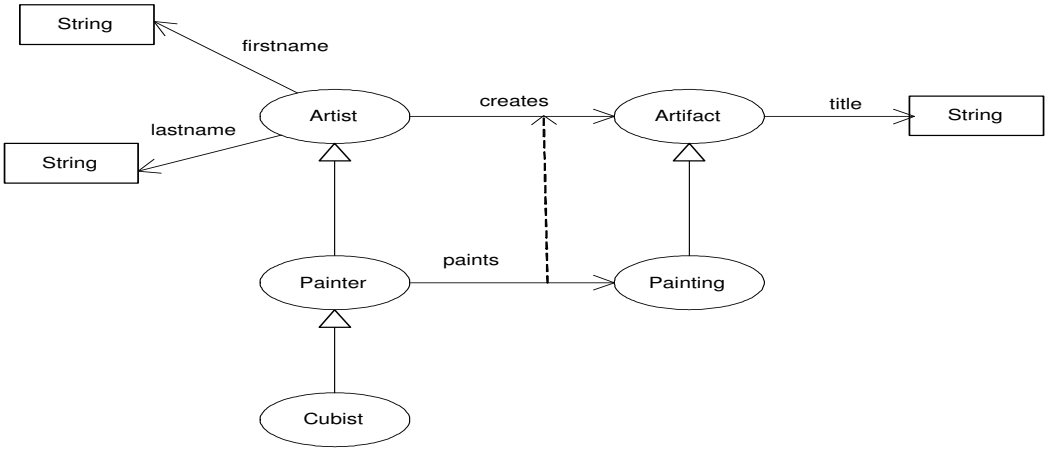
Κεφάλαιο 6

Έλεγχος

Στο κεφάλαιο αυτό γίνεται ο έλεγχος καλής λειτουργίας του συτσήματος.

6.1 Μεθοδολογία Ελέγχου

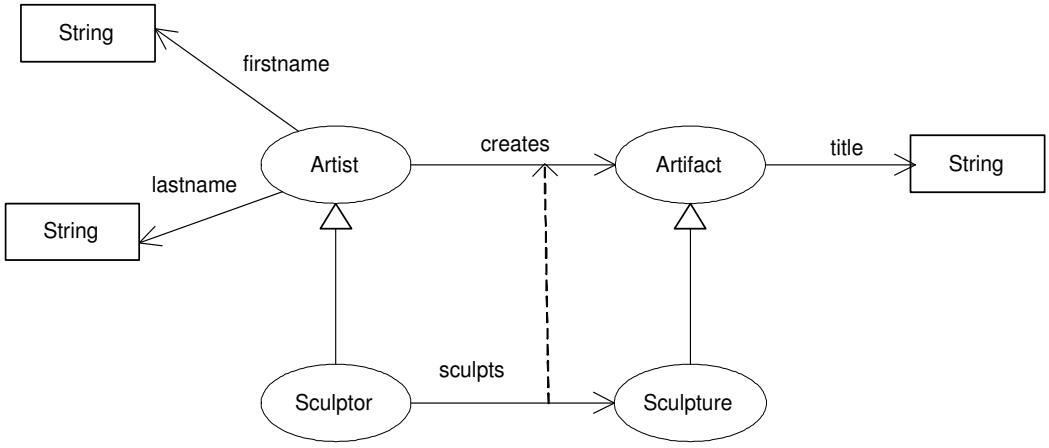
Ο έλεγχος του συστήματος αυτού πραγματοποιήθηκε με τη χρήση ενός σεναρίου λειτουργίας. Σύμφωνα με το σενάριο αυτό θεωρούμε ότι στο σύστημα υπάρχουν τρεις κόμβοι (peer1,peer2,peer3). Θεωρούμε επίσης ότι οι κόμβοι peer2 και peer3 έχουν ήδη σχήμα και δεδομένα. Το σχήμα του peer2 φαίνεται στο Σχήμα 6.1 και το σχήμα του κόμβου peer3 φαίνεται στο Σχήμα 6.2.



Σχήμα 6.1: Σχήμα κόμβου 2

Επίσης η τοπολογία του συστήματος έχει ως εξής: ο peer2 είναι γείτονας του peer1 και ο peer3 γείτονας του peer2.

Αρχικά λοιπόν θα δημιουργήσουμε σχήμα για τον κόμβο peer1 και στη συνέχεια θα εισάγουμε σε αυτό δεδομένα εξετάζοντας έτσι την καλή λειτουργία του υποσυστήματος δημιουργίας σχήματος και του υποσυστήματος εισαγωγής δεδομένων. Στη συνέχεια από τον κόμβο αυτό στέλνουμε ερωτήσεις στους υπόλοιπους για τον έλεγχο του υποσυστήματος απάντησης ερωτήσεων και επικοινωνίας κόμβων.



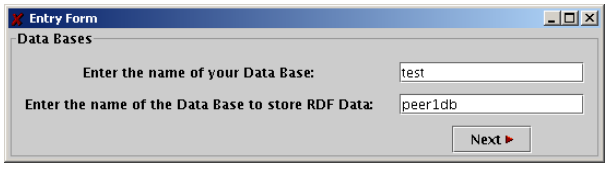
Σχήμα 6.2: Σχήμα κόμβου 3

6.2 Αναλυτική παρουσίαση ελέγχου

Στην ενότητα αυτή παρουσιάζουμε αναλυτικά τον έλεγχο του συστήματος σύμφωνα με το σενάριο που περιγράφηκε στην προηγούμενη ενότητα.

Όταν ξεκινάει να λειτουργεί ο κόμβος peer1 αρχικά εμφανίζεται η οθόνη που φαίνεται στο Σχήμα 6.3. Η οθόνη αυτή είναι η οθόνη εισαγωγής στο σύστημα στην οποία ο χρήστης εισάγει το όνομα της βάσης στην οποία είναι αποθηκευμένα τα δεδομένα του καθώς και το όνομα της βάσης στην οποία θα αποθηκεύει τα RDF δεδομένα.

Μόλις ο χρήστης πατήσει το κουμπί Next τότε εισάγεται στο σύστημα και του εμφανίζεται η οθόνη που φαίνεται στο Σχήμα 6.4. Στην οθόνη αυτή βλέπουμε το μενού το οποίο αποτελείται από τις επιλογές File, View και Search, καθώς και μια κονσόλα στην οποία φαίνεται ο κατάλογος των μηνυμάτων που λαμβάνει και στέλνει ο κόμβος. Όταν ο κόμβος εισέρχεται στο σύστημα στέλνει μήνυμα στον κόμβο διαχειριστή και αυτός του στέλνει μήνυμα αρχείου με το καθολικό σχήμα. Αυτή η επικοινωνία φαίνεται στην κονσόλα του κόμβου.

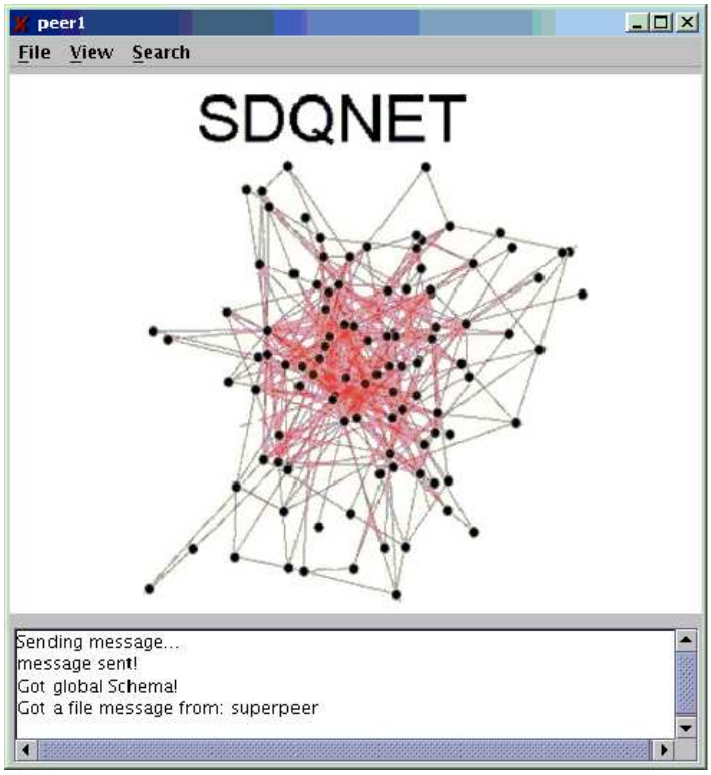


Σχήμα 6.3: Οθόνη εισαγωγής στο σύστημα

6.2.1 Δημιουργία Σχήματος

Για να δημιουργήσουμε ένα σχήμα πηγαίνουμε από το μενού στο File και μετά στο Create New Schema, όπως φαίνεται στο Σχήμα 6.5.

Στη συνέχεια εμφανίζεται η οθόνη που φαίνεται στο Σχήμα 6.6 η οποία μας λέει να επιλέξουμε τα σχήματα τα οποία θα χρησιμοποιήσουμε για να κάνουμε την πράξη. Επειδή



Σχήμα 6.4: Αρχική οθόνη

αρχικά ο κόμβος έχει μόνο το καθολικό σχήμα, αναγκαστικά επιλέγει αυτό και κάνει την πράξη της επιλογής.

Όταν πατήσουμε Next εμφανίζεται η οθόνη που φαίνεται στο Σχήμα 6.7, με την οποία δηλώνουμε το όνομα του νέου σχήματος.

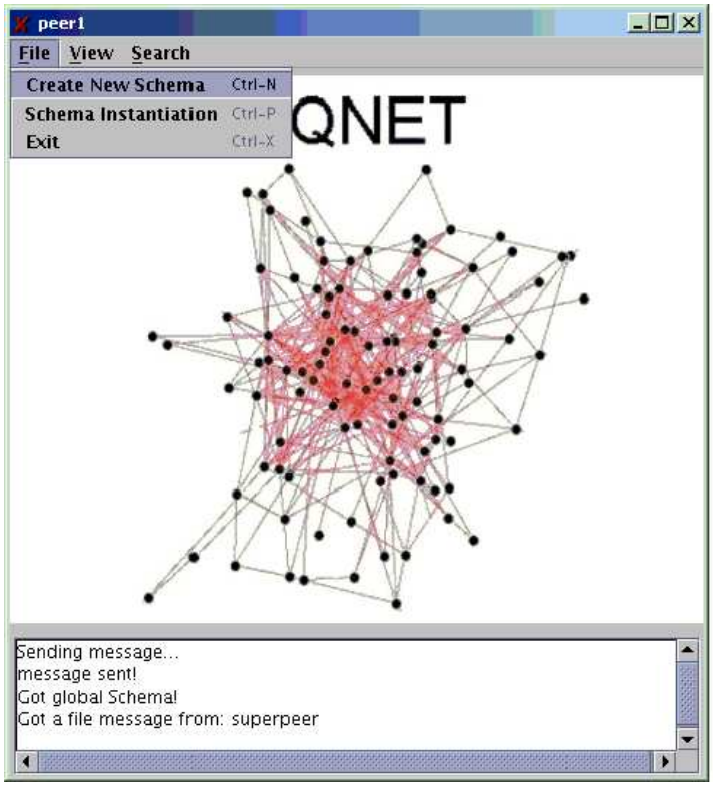
Όταν πατήσουμε OK η επόμενη οθόνη που εμφανίζεται είναι για να επιλέξουμε ποιες κλάσεις και ποια κυριολεκτικά θέλουμε να έχει το σχήμα. Η οθόνη αυτή φαίνεται στο Σχήμα 6.8. Στη δεξιά λίστα φαίνονται οι κλάσεις που έχουμε επιλέξει. Αφού τα επιλέξουμε πατάμε Save για το σώσιμο του σχήματος ως αρχείο RDF/XML και SaveToDB για το σώσιμο στη βάση. Έτσι ολοκληρώνεται η διαδικασία δημιουργίας σχήματος.

Τα σχήματα τα οποία δημιουργούμε, έχουμε τη δυνατότητα να τα δούμε μέσω της επιλογής View του μενού. Αν πατήσουμε View και μετά View the RDFS Graphs στη συνέχεια επιλέγουμε από την οθόνη που φαίνεται στο Σχήμα 6.9 το σχήμα του οποίου θέλουμε να δούμε το γράφο. Στη συνέχεια πατάμε View και το αποτέλεσμα φαίνεται στο Σχήμα 6.10. Αν πατήσουμε View the RDFS files, βλέπουμε το σχήμα σε μορφή RDF/XML όπως φαίνεται στο Σχήμα 6.11.

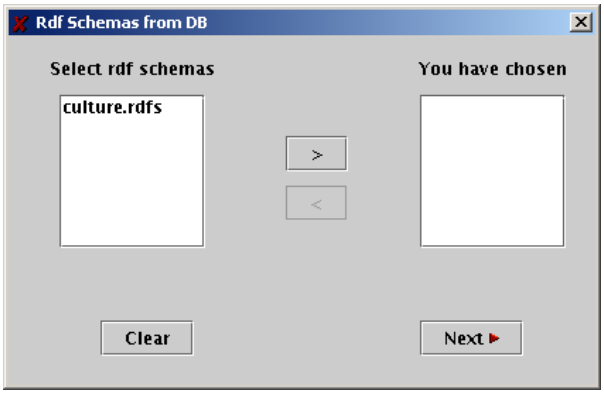
6.2.2 Εισαγωγή δεδομένων στο σχήμα

Για να εισάγουμε δεδομένα στο σχήμα, στο μενού της αρχικής οθόνης πατάμε File και μετά SchemaInstantiation όπως φαίνεται στο Σχήμα 6.12.

Αρχικά εμφανίζεται η οθόνη που φαίνεται στο Σχήμα 6.13 με την οποία ο χρήστης επιλέγει

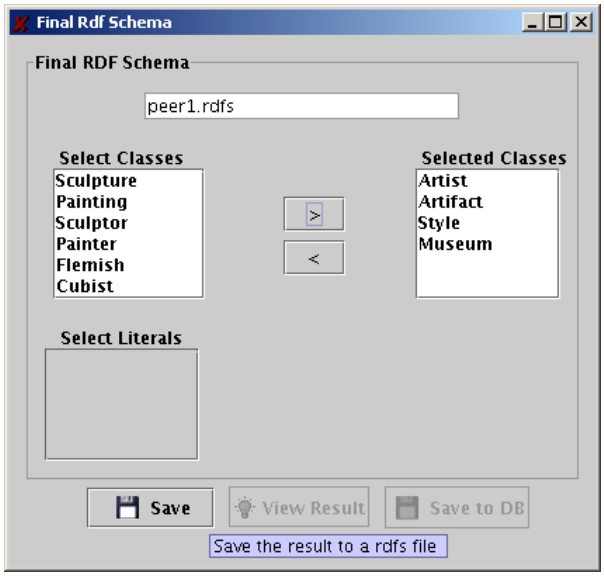


Σχήμα 6.5: Δημιουργία Σχήματος - Πρώτη οθόνη

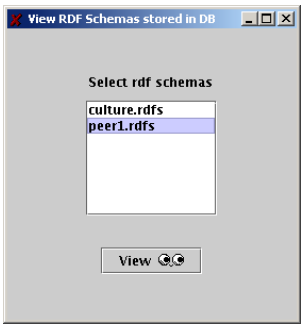


Σχήμα 6.6: Δημιουργία Σχήματος - Επιλογή σχημάτων για την πράξη

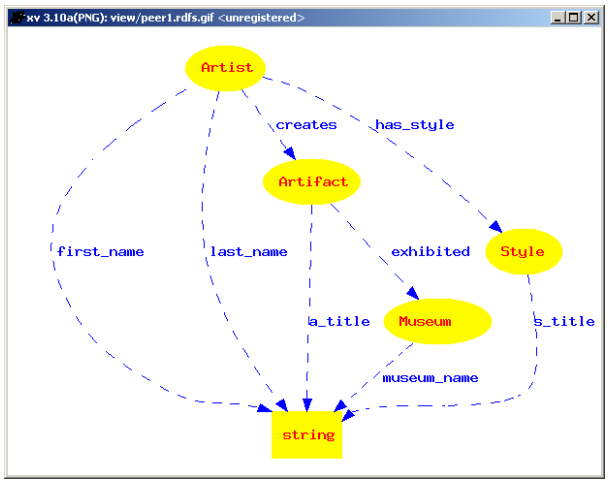
Σχήμα 6.7: Δημιουργία Σχήματος - Επιλογή ονόματος σχήματος



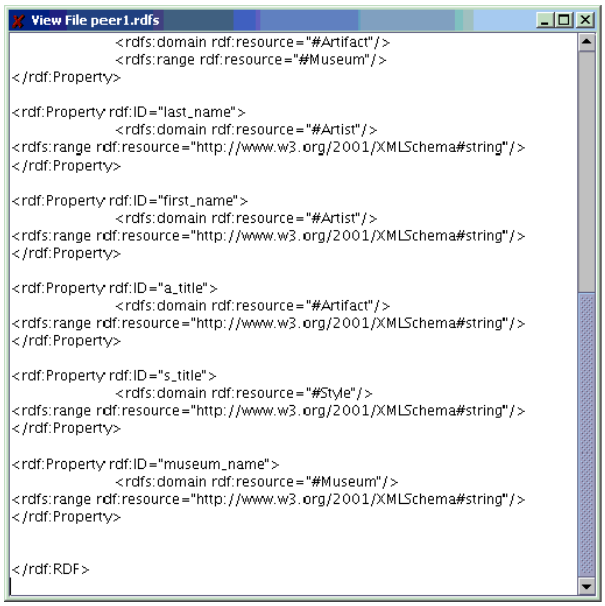
Σχήμα 6.8: Δημιουργία Σχήματος - Επιλογή κλάσεων σχήματος



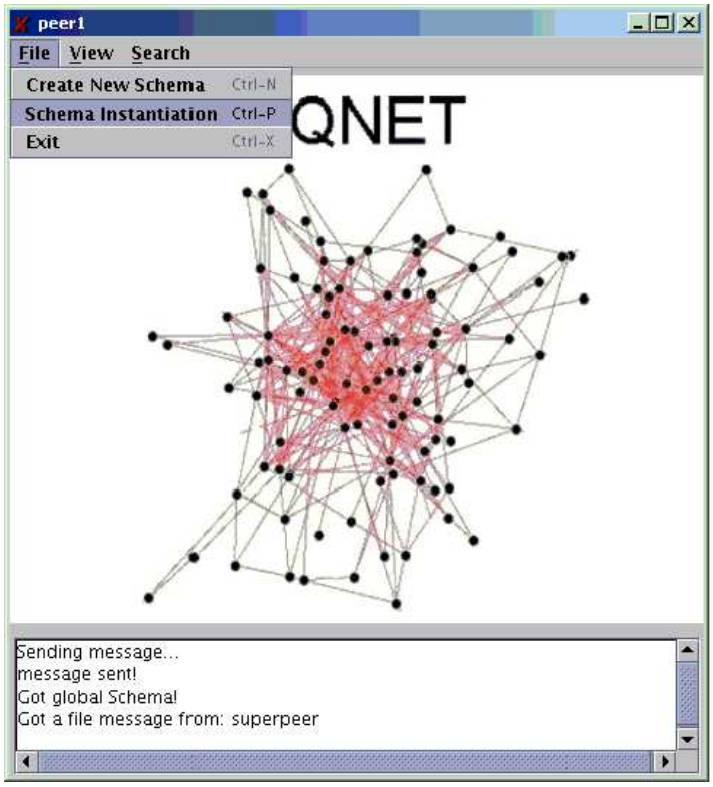
Σχήμα 6.9: Επισκόπιση σχήματος - Επιλογή σχήματος



Σχήμα 6.10: Επισκόπιση σχήματος - Γράφος



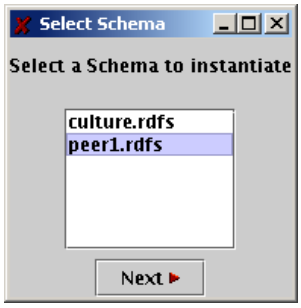
Σχήμα 6.11: Επισκόπιση σχήματος - RDF/XML



Σχήμα 6.12: Εισαγωγή δεδομένων-Αρχική οθόνη

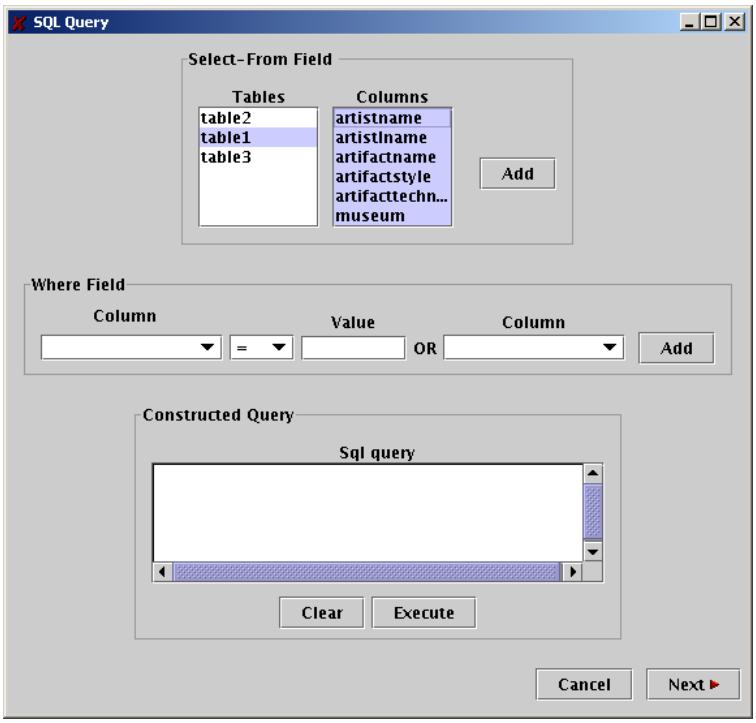
το σχήμα κάτω από το οποίο επιθυμεί να αποθηκεύσει τα δεδομένα.

Όταν ο χρήστης πατήσει Next τότε εμφανίζεται η οθόνη που φαίνεται στο Σχήμα 6.14. Μέσω της οθόνης αυτής ο χρήστης διατυπώνει ένα SQL ερώτημα για την ανάκτηση μέρους



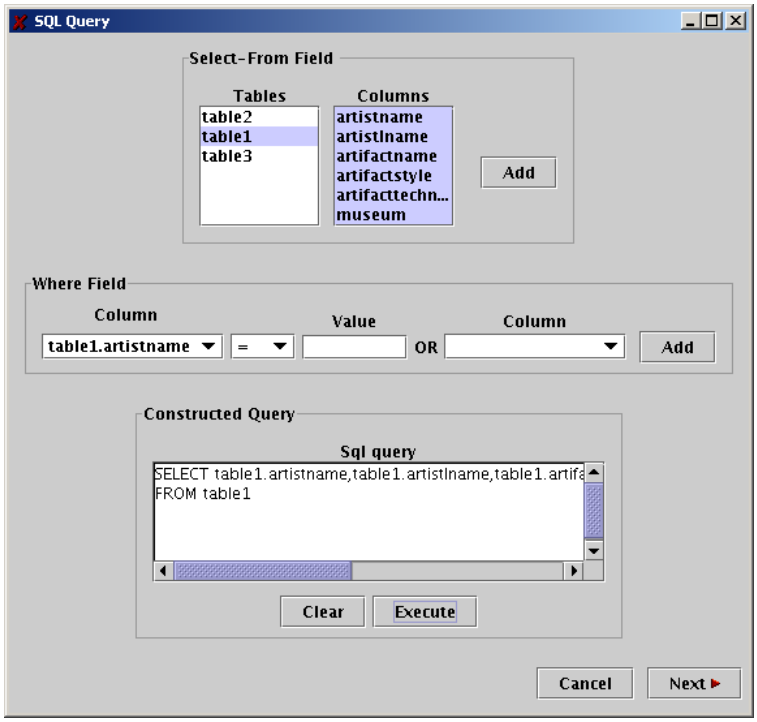
Σχήμα 6.13: Εισαγωγή δεδομένων-Οθόνη επιλογής σχήματος

των δεδομένων του, τα οποία θα αποθηκεύσει στη συνέχεια κάτω από το RDF σχήμα που έχει επιλέξει. Στο Select-From Field επιλέγει τους πίνακες που θέλει να συμμετέχουν στο ερώτημα και για κάθε πίνακα επιλέγει τα πεδία στα οποία θέλει να γίνει η προβολή. Στη συγκεκριμένη περίπτωση έχουμε επιλέξει όλα τα πεδία του table1. Όταν πατήσουμε Add, εμφανίζονται στο Where Field τα πεδία που έχουμε επιλέξει από κάθε πίνακα. Αυτό φαίνεται στο Σχήμα 6.15. Επίσης στο Constructed Query εμφανίζεται το ερώτημα έτσι όπως έχει δημιουργηθεί μέχρι στιγμής. Στο Where Field κατασκευάζουμε αν θέλουμε τους περιορισμούς του ερωτήματος. Στο συγκεκριμένο ερώτημα δεν έχουμε βάλει πεδίο Where. Όταν πατάμε Execute αποτιμάται το ερώτημα. Με το Clear σβήνεται το ερώτημα που έχουμε δημιουργήσει και μπορούμε να ξεκινήσουμε από την αρχή.



Σχήμα 6.14: Εισαγωγή δεδομένων-Οθόνη κατασκευής SQL ερωτήματος

Όταν πατήσουμε Next η επόμενη οθόνη που εμφανίζεται είναι αυτή που φαίνεται στο

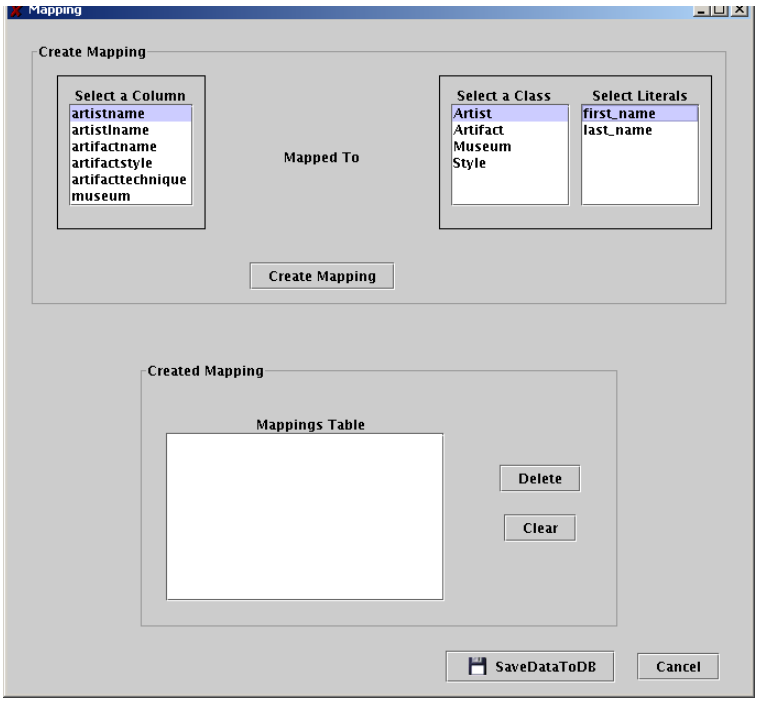


Σχήμα 6.15: Εισαγωγή δεδομένων-Οθόνη κατασκευής SQL ερωτήματος

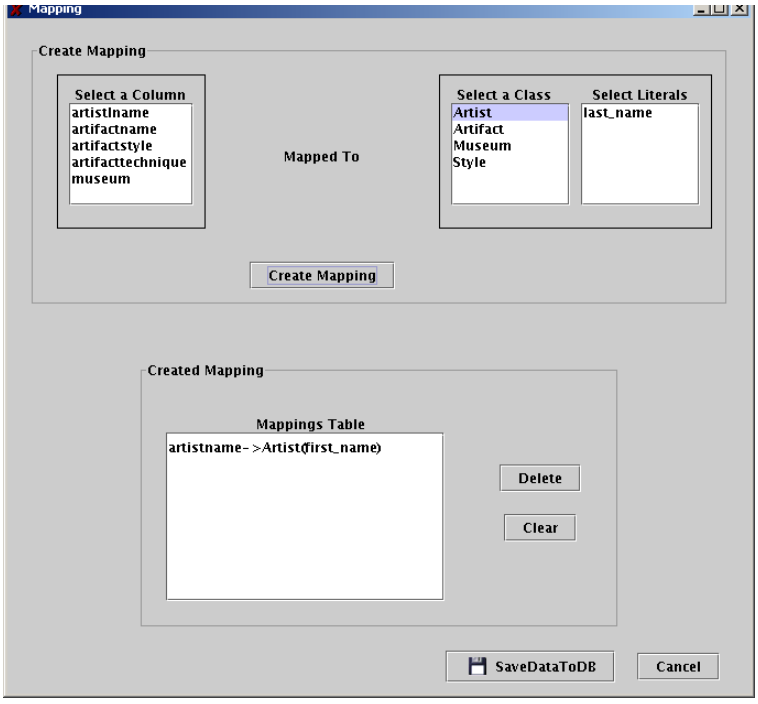
Σχήμα 6.16. Η οθόνη αυτή βοηθάει το χρήστη να κατασκευάσει αντιστοιχίσεις ανάμεσα στα πεδία που πήραν μέρος στο SQL ερώτημα και στα κυριολεκτικά των κλάσεων του σχήματος. Αριστερά φαίνονται τα πεδία και δεξιά οι κλάσεις με τα κυριολεκτικά τους. Αφού επιλέξουμε πεδίο και κυριολεκτικό κλάσης, πατάμε Create Mapping και τότε εμφανίζεται η αντιστοίχιση που μόλις κάναμε στο Constructed Mappings όπως φαίνεται στο Σχήμα 6.17. Συνεχίζοντας να κάνουμε αντιστοιχίσεις έχουμε το αποτέλεσμα που φαίνεται στο Σχήμα 6.18. Με το κουμπί Delete μπορούμε να σβήσουμε όποια αντιστοίχιση θέλουμε και με το κουμπί Clear να σβήσουμε όλες τις αντιστοιχίσεις και να ξαναρχίσουμε από την αρχή. Όταν πατήσουμε το κουμπί SaveDataToDB τα δεδομένα σώζονται στην RDF βάση, με βάση τις αντιστοιχίσεις που κάναμε και το σύστημα μας ενημερώνει ότι τα δεδομένα σώθηκαν με επιτυχία όπως φαίνεται στο Σχήμα 6.19.

6.2.3 Ερωτήσεις αναζήτησης δεδομένων

Η κατασκευή και αποστολή ερωτήσεων αναζήτησης δεδομένων, γίνεται μέσω της επιλογής Search του μενού. Αρχικά πατάμε την επιλογή Settings για να ορίσουμε τις παραμέτρους της ερώτησης. Η οθόνη που αντιστοιχεί σε αυτή την επιλογή φαίνεται στο Σχήμα 6.20. Το Maximum Number of Hops είναι ο μέγιστος αριθμός κόμβων στους οποίους θέλουμε να πάει η ερώτηση. Αρχικά επιλέγουμε 4. Το Level of Query Transformation είναι το επίπεδο του μετασχηματισμού της ερώτησης. Αν θέλουμε η ερώτηση να απαντηθεί ως έχει επιλέγουμε Level 0 ενώ αν θέλουμε να μετασχηματίζεται όπου είναι απαραίτητο, επιλέγουμε Level 1. Εδώ επιλέγουμε αρχικά Level 0 και πατάμε Apply για να αποθηκευτούν οι επιλογές.

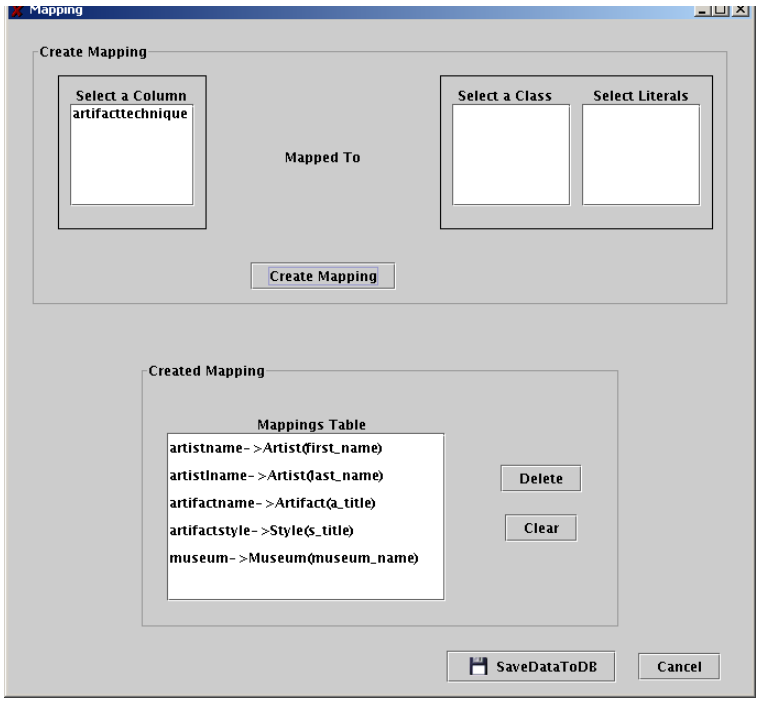


Σχήμα 6.16: Εισαγωγή δεδομένων - Οθόνη αντιστοιχίσεων 1

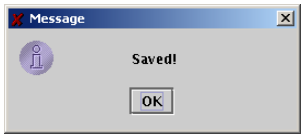


Σχήμα 6.17: Εισαγωγή δεδομένων - Οθόνη αντιστοιχίσεων 2

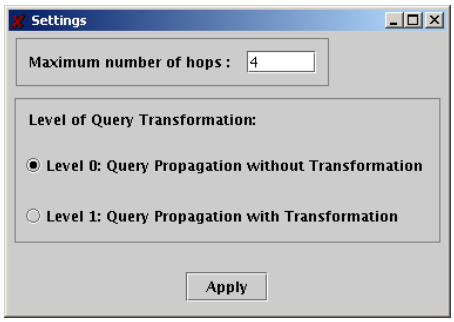
Στη συνέχεια επιλέγουμε New query από το μενού Search και εμφανίζεται η οθόνη που φαίνεται στο Σχήμα 6.21. Με την οθόνη αυτή επιλέγουμε το σχήμα με βάση το οποίο θέλουμε να κατασκευάσουμε τις ερωτήσεις. Αφού επιλέξουμε το σχήμα, πατάμε Next και εμφανίζεται



Σχήμα 6.18: Εισαγωγή δεδομένων - Οθόνη αντιστοιχίσεων 3

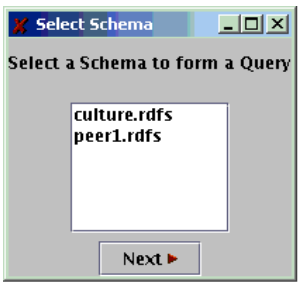


Σχήμα 6.19: Εισαγωγή δεδομένων-Οθόνη επιβεβαίωσης

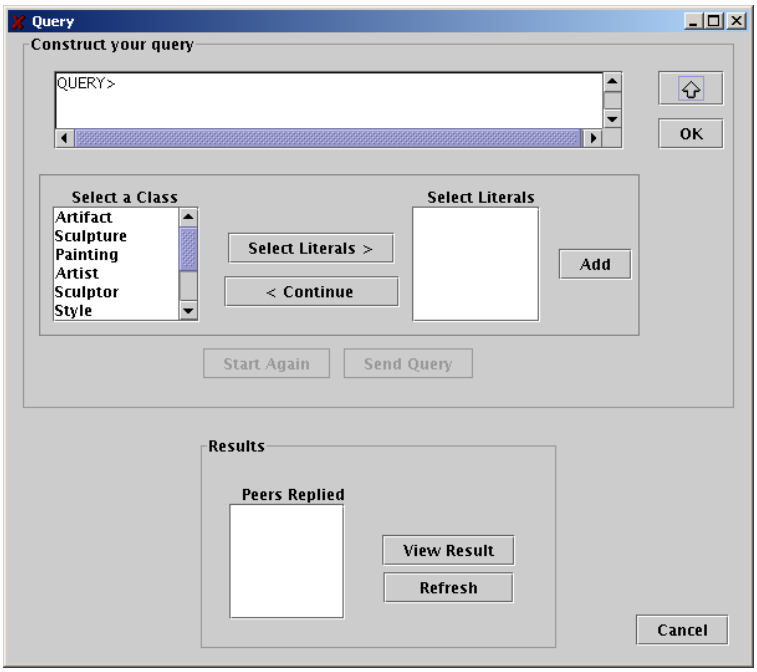


Σχήμα 6.20: Οθόνη Παραμέτρων ερώτησης

η οθόνη κατασκευής ερώτησης που φαίνεται στο Σχήμα 6.22. Με τη βοήθεια της οθόνης αυτής κατασκευάζουμε ένα ερώτημα επιλέγοντας μονοπάτι πάνω στον RDF γράφο. Από κάθε κλάση του μονοπατιού επιλέγουμε τα κυριολεκτικά στα οποία θέλουμε να γίνει η προβολή. Στην οθόνη που φαίνεται στο Σχήμα 6.23 έχουμε επιλέξει κλάση και μετά Select Literals για να εμφανιστούν τα κυριολεκτικά. Όταν πατήσουμε Add αρχίζει να εμφανίζεται από πάνω το μονοπάτι που κατασκευάζουμε όπως φαίνεται στο Σχήμα 6.24.



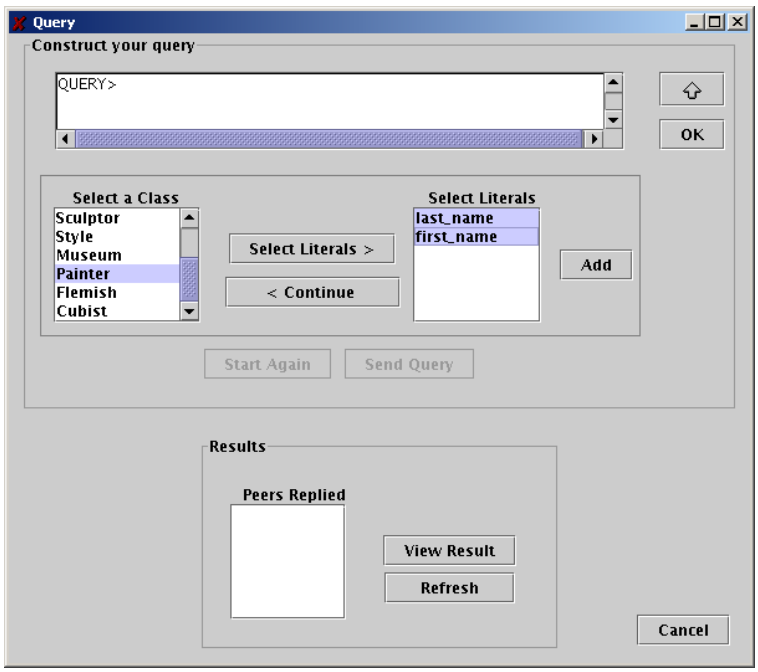
Σχήμα 6.21: Ερωτήσεις αναζήτησης – Οθόνη Επιλογής σχήματος



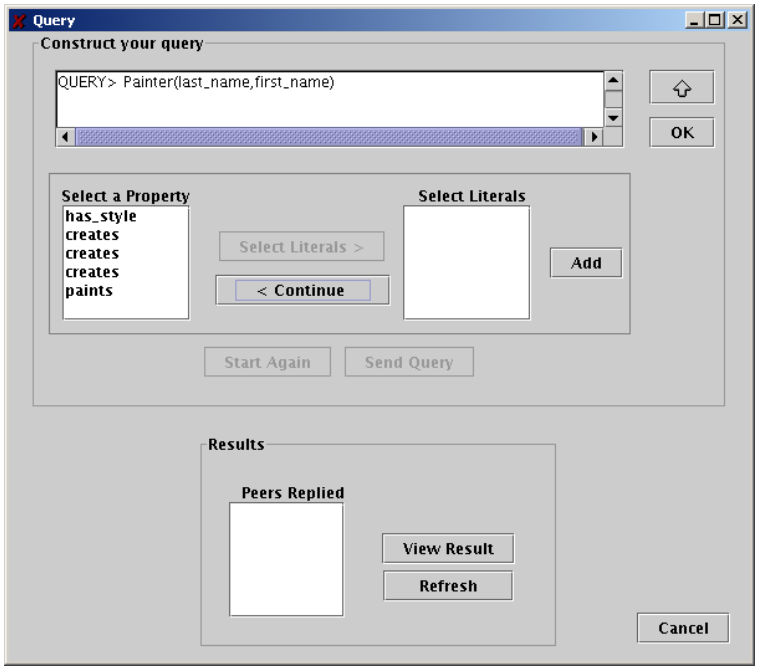
Σχήμα 6.22: Ερωτήσεις αναζήτησης – Οθόνη Κατασκευής ερώτησης 1

Για να συνεχίσουμε να κατασκευάζουμε το μονοπάτι πατάμε Continue και επιλέγουμε ιδιότητα, όπως φαίνεται στο Σχήμα 6.24. Στη συνέχεια πατώντας πάλι Continue ξαναεπιλέγουμε κλάση και έτσι κατασκευάζουμε το ερώτημα που θέλουμε. Κουμπί με το βέλος μας δίνει τη δυνατότητα να πηγαίνουμε ένα βήμα πίσω στο μονοπάτι που έχουμε κατασκευάσει. φού κατασκευάσουμε την ερώτηση πατάμε OK. Η αντίστοιχη οθόνη φαίνεται στο Σχήμα 6.25. Στην οθόνη αυτή φαίνεται και το ερώτημα που έχει κατασκευαστεί στη συγκεκριμένη περίπτωση.

Τώρα έχουμε τη δυνατότητα είτε να στείλουμε το ερώτημα στους γειτονικούς κόμβους (Send Query) είτε να αρχίσουμε την κατασκευή του ερωτήματος ξανά από την αρχή (Start Again). Αν πατήσουμε Send Query, το ερώτημα στέλνεται στους γειτονικούς κόμβους και περιμένουμε τις απαντήσεις τους. Στο πλαίσιο Peers Replied εμφανίζονται οι κόμβοι οι οποίοι απαντούν. Οι απαντήσεις που παίρνουμε για τη συγκεκριμένη ερώτηση φαίνονται στο Σχήμα 6.26.

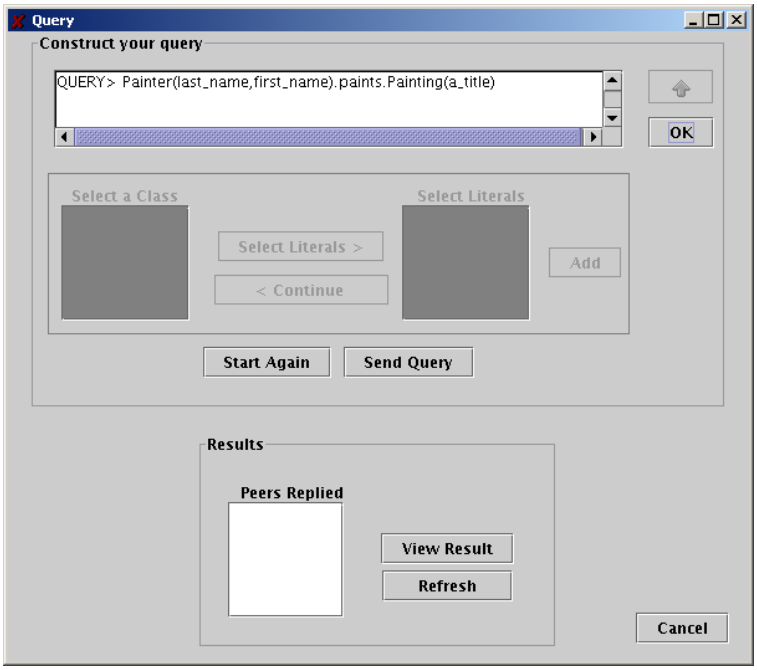


Σχήμα 6.23: Ερωτήσεις αναζήτησης – Οθόνη Κατασκευής ερώτησης 2

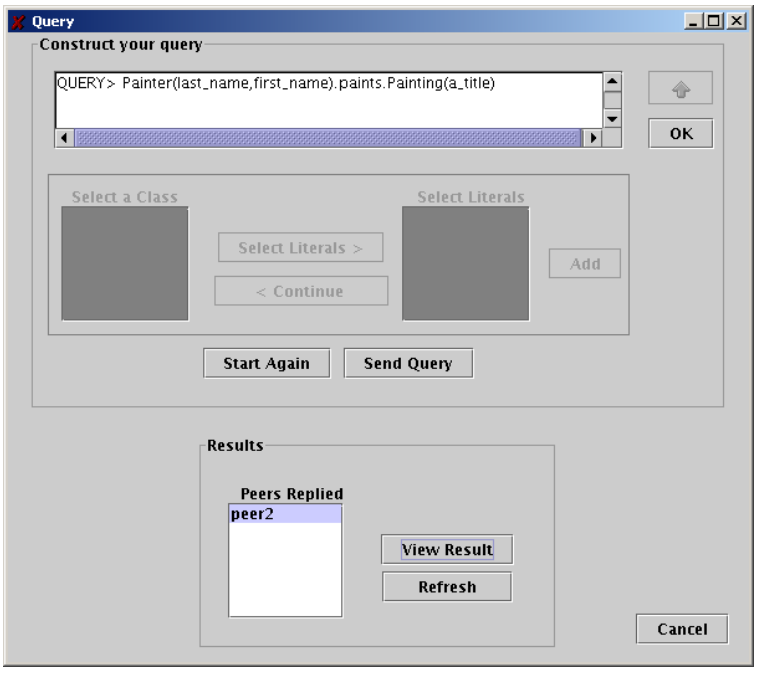


Σχήμα 6.24: Ερωτήσεις αναζήτησης – Οθόνη Κατασκευής ερώτησης 3

Όπως φαίνεται, στη συγκεκριμένη ερώτηση απαντάει μόνο ο peer2. Αυτό οφείλεται στο γεγονός ότι για τον peer3 η ερώτηση δεν είναι ικανοποιήσιμη, όπως φαίνεται και από το σχήμα του (Σχήμα 6.2, ενώ ταυτόχρονα στα Settings έχουμε επιλέξει διάδοση της ερώτησης χωρίς επίτρεψη μετασχηματισμού (Query Propagation without transformation). Αν επιλέξουμε



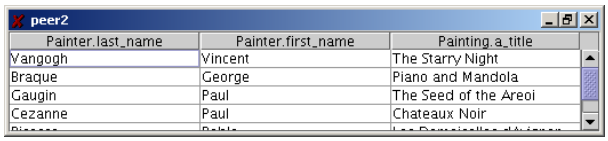
Σχήμα 6.25: Ερωτήσεις αναζήτησης – Οθόνη Κατασκευής ερώτησης 4



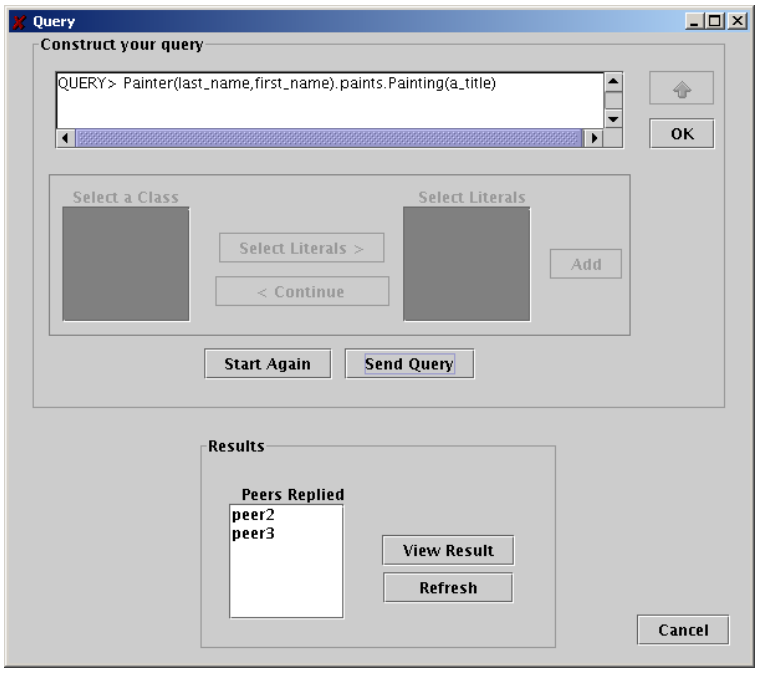
Σχήμα 6.26: Ερωτήσεις αναζήτησης – Οθόνη αποτελεσμάτων ερώτησης 1

τον peer2 και μετά πατήσουμε View μπορούμε να δούμε την απάντησή του σε μορφή πίνακα όπως φαίνεται στο Σχήμα 6.27.

Αν αλλάξουμε τα Settings και επιλέξουμε Query Propagation with Transformation τότε για την ίδια ερώτηση το αποτέλεσμα που παίρνουμε είναι αυτό που φαίνεται στο Σχήμα 6.28.



Σχήμα 6.27: Ερωτήσεις αναζήτησης – Απάντηση κόμβου



Σχήμα 6.28: Ερωτήσεις αναζήτησης – Οθόνη αποτελεσμάτων ερώτησης 2

Τέλος αξιοσημείωτο είναι ότι αν στα Settings επιλέξουμε Query Propagation with Transformation αλλά βάλουμε Maximum Number of Hops ίσο με ένα τότε θα πάρουμε και πάλι την απάντηση που φαίνεται στο Σχήμα 6.26.

Κεφάλαιο 7

Επίλογος

7.1 Συμπεράσματα

Τα συστήματα ομότιμων κόμβων, προκειμένου να υποστηρίζουν πιο εκφραστικές λειτουργίες αναπαράστασης και αναζήτησης δεδομένων, εξελίχθηκαν στα συστήματα ομότιμων κόμβων τα οποία βασίζονται στις τεχνολογίες του Σημασιολογικού Ιστού για την αναπαράσταση των δεδομένων μέσω σχημάτων που τα περιγράφουν (Schema-based peer-to-peer systems).

Στα συστήματα αυτά κάθε κόμβος χρησιμοποιεί ένα σχήμα για την αναπαράσταση των δεδομένων του. Όμως σε ένα σύστημα ομότιμων κόμβων, κάθε κόμβος έχει διαφορετικές απαιτήσεις αναπαράστασης δεδομένων. Επομένως πρέπει να υπάρχει ευελιξία στην επιλογή σχήματος. Τα συστήματα που έχουν προταθεί μέχρι τώρα και παρέχουν αυτή την ευελιξία, για να είναι δυνατή η αναζήτηση πληροφορίας, απαιτούν την ύπαρξη κανόνων αντιστοίχισης μεταξύ των σχημάτων με βάση τους οποίους να μετασχηματίζονται οι ερωτήσεις. Όμως δεν υποστηρίζεται ακόμα αυτόματη δημιουργία και δυναμική ανανέωση των κανόνων, που είναι απαραίτητα για τα συστήματα ομότιμων κόμβων.

Η συνεισφορά της παρούσας διπλωματικής εργασίας έχει δύο σκέλη. Το πρώτο αφορά τη δημιουργία ενός πλήρους συστήματος ομότιμων κόμβων βασισμένο σε σχήματα RDF το οποίο παρέχει: (α) την υποδομή για την επικοινωνία των κόμβων,(β) μηχανισμό δημιουργίας σχήματος, (γ) μηχανισμό ενσωμάτωσης σχεσιακών δεδομένων στο σχήμα με τη χρήση αντιστοιχίσεων που δημιουργεί ο χρήστης με τη βοήθεια ειδικής διαπροσωπείας, (δ) ευέλικτη διαπροσωπεία χρήστη για τη διατύπωση ερωτημάτων και (ε) μηχανισμό απάντησης και επεξεργασίας ερωτήσεων.

Το δεύτερο σκέλος αφορά το γεγονός ότι το συγκεκριμένο σύστημα προσφέρει μια σχετική ευελιξία ως προς την επιλογή του σχήματος από τον κάθε κόμβο, ενώ ταυτόχρονα δίνει τη δυνατότητα μετασχηματισμού ερωτήσεων χωρίς τη χρήση κανόνων αντιστοίχισης. Συγκεκριμένα, τα σχήματα των κόμβων αποτελούν υποσύνολα−όψεις(views) ενός βασικού σχήματος που ονομάζεται καθολικό σχήμα. Εκμεταλλευόμενοι λοιπόν το γεγονός ότι τα σχήματα αυτά είναι συμβατά μεταξύ τους, έχουμε τη δυνατότητα ελέγχου της ικανοποιησιμότητας μιας ερώτησης και μετατροπής της όπου χρειάζεται, χρησιμοποιώντας τόσο το σχήμα του κόμβου

όσο και το καθολικό σχήμα.

Συμπερασματικά το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής είναι ένα πλήρες σύστημα ομότιμων κόμβων βασισμένο σε σχήματα, το οποίο καθιστά δυνατή την αναζήτηση της πληροφορίας με ένα διαφορετικό τρόπο απ΄ ότι τα προϋπάρχοντα συστήματα.

7.2 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί περαιτέρω, τουλάχιστον ως προς τρεις κατευθύνσεις. Συγκεκριμένα, αναφέρονται τα ακόλουθα:

- Ενσωμάτωση διαδικασίας επιλογής σχήματος με βάση το οποίο ο κόμβος θα συμμετέχει στο σύστημα. Έτσι όπως έχει σχεδιαστεί το σύστημα, κάθε κόμβος έχει τη δυνατότητα να δημιουργήσει πολλά σχήματα και να αποθηκεύσει δεδομένα σε περισσότερα από ένα. Ως σχήμα του κόμβου (με βάση το οποίο απαντάει τις ερωτήσεις), θεωρείται το τελευταίο στο οποίο αποθήκευσε δεδομένα. Η δυνατότητα επιλογής θα του παρείχε περισσότερη ευελιξία.

- Δυνατότητα αντιστοίχισης δεδομένων τα οποία να μην είναι αποθηκευμένα σε βάση δεδομένων αλλά σε αρχεία. Η αποδέσμευση από τη βάση δεδομένων θα έκανε το σύστημα πιο εύκολο στην εγκατάσταση και τη χρήση.

- Αξιολόγηση του συστήματος ως προς τη συμπεριφορά του αν συμμετέχει σε αυτό μεγάλος αριθμός κόμβων (scalability testing) και αν χρησιμοποιηθεί ένα πολύ μεγάλο καθολικό σχήμα. Η αξιολόγηση αυτή αφορά την ταχύτητα με την οποία ένας κόμβος παίρνει απαντήσεις σε μια ερώτηση καθώς και την ποιότητα των απαντήσεων.

Βιβλιογραφία

[1] Karl Aberer και Manfred Hauswirth. An overview of peer-to-peer information systems. Στο *WDAS*, σελίδες 171–188, 2002.

[2] Sofia Alexaki, Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis και Karsten Tolle. The ics-forth rdfsuite: Managing voluminous rdf description bases. Στο *SemWeb*, 2001.

[3] Tim Berners-Lee, James Hendler και Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[4] Jeen Broekstra, Arjohn Kampman και Frankvan Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. Στο *Proceedings of the first International Semantic Web Conference (ISWC 2002)*Ian Horrocks και James Hendler, επιμελητές, αριθμός 2342 στο Lecture Notes in Computer Science, σελίδες 54–68, Sardinia, Italy, 2002. Springer Verlag, Heidelberg Germany.

[5] Min Cai και Martin Frank. Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. Στο *WWW '04: Proceedings of the 13th international conference on World Wide Web*, σελίδες 650–657, New York, NY, USA, 2004. ACM Press.

[6] Li Gong. Jxta: A network programming environment. *IEEE Internet Computing*, 5(3):88–95, 2001.

[7] Peter Haase, Jeen Broekstra, Andreas Eberhart και Raphael Volz. A comparison of rdf query languages. Στο *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004.*, 2004.

[8] Alon Y. Halevy, Zachary G. Ives, Peter Mork και Igor Tatarinov. Piazza: data management infrastructure for semantic web applications. Στο *WWW*, σελίδες 556–567, 2003.

[9] http://4suite.org/docs/ftssdocs/ Versa.doc?xslt=/4Suite/stdpage.xslt, χ.χ.

[10] http://www.w3.org/DesignIssues/Notation3.html, χ.χ.

[11] http://www.w3.org/Submission/2004/SUBM RDQL-20040109, χ.χ.

[12] http://www.w3.org/TR/rdf primer, χ.χ.

[13] Zoi Kaoudi, Theodore Dalamagas και Timos K. Sellis. Rdfsculpt: Managing rdf schemas under set-like semantics. Στο *ESWC*, σελίδες 123–137, 2005.

[14] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis και Michel Scholl. Rql: a declarative query language for rdf. Στο *WWW '02: Proceedings of the 11th international conference on World Wide Web*, σελίδες 592–603, New York, NY, USA, 2002. ACM Press.

[15] Giorgos Kokkinidis και Vassilis Christophides. Semantic query routing and processing in p2p database systems: The ics-forth sqpeer middleware. Στο *EDBT Workshops*, τόμος 3268 στο *Lecture Notes in Computer Science*, σελίδες 486–495. Springer, 2004.

[16] Wolfgang Nejdl, Wolf Siberski και Michael Sintek. Design issues and challenges for rdf- and schema-based peer-to-peer systems. *SIGMOD Rec.*, 32(3):41–46, 2003.

[17] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjorn Naeve, Mikael Nilsson, Matthias Palmer και Tore Risch. Edutella: a p2p networking infrastructure based on rdf. Στο *WWW '02: Proceedings of the 11th international conference on World Wide Web*, σελίδες 604–615, New York, NY, USA, 2002. ACM Press.

[18] Michael Sintek και Stefan Decker. Triple - a query, inference, and transformation language for the semantic web. Στο *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, σελίδες 364–378, London, UK, 2002. Springer-Verlag.

[19] Ζωή Καούδη. Πρότυπο Σύστημα Αποθήκευσης και Διαχείρισης σχημάτων RDFS. KDBS Lab, NTU Athens, 2004.

[20] Έλλη Ανδρουλάκη. Υλοποίηση Ενεργού Μηχανισμού σε σύστημα Ομότιμων Βάσεων. KDBS Lab, NTU Athens, 2005.