ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανταλλαγή Δεδομένων σε Αδόμητα Δίκτυα Ομότιμων Κόμβων με χρήση Οντολογιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Δ. ΚΑΡΑΝΑΣΟΥ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανταλλαγή Δεδομένων σε Αδόμητα Δίκτυα Ομότιμων Κόμβων με χρήση Οντολογιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Δ. ΚΑΡΑΝΑΣΟΥ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Οκτωβρίου 2008.

.....	.....	.....
Τιμολέων Σελλής	Ιωάννης Βασιλείου	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2008

.....

ΚΩΝΣΤΑΝΤΙΝΟΣ Δ. ΚΑΡΑΝΑΣΟΣ

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

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Τίμο Σελλή για την επίβλεψη της παρούσας διπλωματικής. Επίσης, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Δημήτρη Σκούτα για την καθοδήγηση και τη βοήθειά του στην εκπόνηση της εργασίας. Τέλος, θα ήθελα να πω ένα μεγάλο ευχαριστώ σε όλους εκείνους που ήταν δίπλα μου και με στήριξαν όλα αυτά τα χρόνια.

Περίληψη

Η παρούσα διπλωματική εργασία εξετάζει αδόμητα συστήματα ομότιμων κόμβων στα οποία οι κόμβοι ανταλλάσουν δομημένα δεδομένα. Κάθε κόμβος διαθέτει μια τοπική βάση δεδομένων και αντιστοιχίσεις μεταξύ του σχήματος της βάσης του και των σχημάτων των κόμβων με τους οποίους είναι απευθείας συνδεδεμένος. Οι κόμβοι θέτουν ερωτήματα στο δίκτυο ή επανεγγράφουν στο τοπικό τους σχήμα ερωτήματα που τους απευθύνονται. Όμως, οι τυπικοί αλγόριθμοι επανεγγραφής ερωτημάτων, που θεωρούν μόνο ερωτήματα που μπορούν να επανεγγραφούν πλήρως σε ένα τοπικό σχήμα, δεν είναι επαρκείς για ένα περιβάλλον όπου οι κόμβοι αναζητούν και ικανοποιούνται με πληροφορία που είναι σημασιολογικά σχετική, αλλά όχι απαραίτητα ταυτόσημη με τις απαιτήσεις τους. Ουσιώδες πρόβλημα σε τέτοια συστήματα είναι ότι οι κόμβοι δεν μπορούν να κρίνουν με ευκολία τη σημασιολογική ομοιότητα μεταξύ του δικού τους ενδιαφέροντος και της πληροφορίας των υπόλοιπων κόμβων. Επιπροσθέτως, οι κόμβοι δεν μπορούν να αξιολογήσουν τη σημασιολογική σχετικότητα των απαντήσεων που δέχονται στα ερωτήματα που θέτουν στο δίκτυο.

Σκοπός της διπλωματικής είναι η ανάπτυξη ενός συστήματος στο οποίο να υπάρχει η δυνατότητα προσδιορισμού του βαθμού ομοιότητας των σχημάτων των κόμβων, καθώς και των ερωτημάτων που προκύπτουν από τη διαδικασία επανεγγραφής σε σχέση με τα αρχικά. Στα πλαίσια αυτά, θεωρούμε την ύπαρξη μιας οντολογίας η οποία περιγράφει το πεδίο ενδιαφέροντος των κόμβων. Η οντολογία χρησιμοποιείται για να επισημειώνεται σημασιολογικά κάθε κόμβος, με αποτέλεσμα να περιγράφεται με σαφήνεια η πληροφορία που παρέχεται από αυτόν. Το μέτρο ομοιότητας στο οποίο βασίζεται το σύστημα υιοθετεί τις έννοιες της ευστοχίας και της ακρίβειας από το πεδίο της ανάκτησης πληροφοριών και λαμβάνει υπόψη τις σημασιολογικές επισημειώσεις των κόμβων, τις αντιστοιχίσεις των σχημάτων και το εκάστοτε ερώτημα. Υλοποιήθηκαν αλγόριθμοι που υπολογίζουν τη σημασιολογική ομοιότητα των κόμβων με βάση τα κριτήρια αυτά. Η υλοποίηση βασίστηκε σε εργαλεία ανοικτού κώδικα, όπως το Jena και το Pellet. Για την πειραματική αξιολόγηση του συστήματος χρησιμοποιήθηκαν δεδομένα από τη βάση ταινιών IMDb.

Λέξεις Κλειδιά: Δίκτυα Ομότιμων Κόμβων, Σημασιολογικός Ιστός, RDF, RDFS, OWL, επανεγγραφή ερωτημάτων, ευστοχία, ακρίβεια

Abstract

The present thesis examines unstructured peer-to-peer systems in which peers share structured data. Every peer owns a local database and mappings between the schema of its database and the ones of the peers that are directly linked to it. Peers express their queries and rewrite incoming queries on their local schema. However, the available rewriting algorithms, which consider only queries that can be completely rewritten to a target schema, are not enough for an environment where peers seek, and are satisfied with, information semantically similar, but not necessarily identical, to their requests. A major problem in such systems is that peers cannot easily judge the semantic relativeness of their interests to interests of other peers. Moreover, peers cannot evaluate the semantic relativeness of answers that they receive to the queries they pose to the network.

The purpose of the thesis is the development of a system which is capable of evaluating the semantic relativeness between peer schemas, as well as between queries and their rewritten versions on the peers. To this end, we assume the existence of a shared ontology describing the domain of interest of the peers. The ontology is used to semantically annotate each peer schema, making explicit the type of information provided by it. The semantic similarity measure on which the system is based, adopts the notions of recall and precision from the field of Information Retrieval and takes into consideration the semantic annotations of the peers, the pairwise mappings between the peers and the particular queries posed. Several algorithms that compute the semantic similarity of the peers were implemented with respect to these parameters. The implementation was based on open source tools like Jena and Pellet. Data from the movie database IMDb were used for the experimental evaluation of the system.

Keywords: peer-to-peer networks, Semantic Web, RDF, RDFS, OWL, query rewriting, recall, precision

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1

Εισαγωγή

Στο κεφάλαιο αυτό γίνεται αρχικά μια εισαγωγή στο γενικότερο πεδίο με το οποίο καταπιάνεται η παρούσα διπλωματική εργασία (Ενότητα 1.1). Στην Ενότητα 1.2 παρουσιάζουμε το αντικείμενο της εργασίας και τη συνεισφορά της. Τέλος, στην ενότητα 1.3 παρουσιάζεται επιγραμματικά το περιεχόμενο του κάθε κεφαλαίου της εργασίας.

1.1 Δίκτυα Ομότιμων Κόμβων και Σημασιολογικός Ιστός

Η συνεχώς αυξανόμενη και ευρέως διαδεδομένη διαθεσιμότητα δεδομένων από πηγές πληροφοριών του Διαδικτύου έχει δημιουργήσει μεγάλο ενδιαφέρον όσον αφορά στις δυνατότητες ανταλλαγής πληροφοριών. Στα πλαίσια αυτά δημιουργήθηκαν τα δίκτυα ομότιμων κόμβων, τα οποία αποτελούν συστήματα με δυνατότητα κατανεμημένων υπολογισμών, όπου οι κόμβοι ανταλλάσουν πληροφορίες και υπηρεσίες απευθείας με τους άλλους κόμβους του δικτύου. Τα δίκτυα ομότιμων κόμβων έχουν χρησιμοποιηθεί εκτενώς τα τελευταία χρόνια για το μαζικό διαμοιρασμό και ανταλλαγή αδόμητων δεδομένων, όπως είναι τα αρχεία.

Μια εξέλιξη των δικτύων ομότιμων κόμβων αποτελούν τα Συστήματα Διαχείρισης Δεδομένων Κόμβων (Peer Data Management Systems, PDMS). Τα PDMS παίζουν ηγετικό ρόλο στην ανταλλαγή δομημένων πληροφοριών. Αποτελούνται από αυτόνομες πηγές οι

οποίες αποθηκεύουν και διαχειρίζονται δομημένα δεδομένα με τοπικό τρόπο, αποκαλύπτοντας μέρος του τοπικού σχήματος δεδομένων στους υπόλοιπους κόμβους.

Τα "αγνά" συστήματα ομότιμων κόμβων (που δεν έχουν υπερ-κόμβους), θεωρείται ότι λειτουργούν χωρίς κάποιο καθολικό σχήμα. Χωρίς σχήμα αναφοράς, οι βάσεις δεδομένων εκφράζουν και απαντούν ερωτήματα βασισμένα στο τοπικό τους σχήμα. Πιο συγκεκριμένα, οι κόμβοι που είναι άμεσα συνδεδεμένοι, χρησιμοποιούν ένα τετριμμένο τρόπο ανταλλαγής και κατανόησης των δεδομένων των γειτόνων τους. Συνήθως αυτό πραγματοποιείται με χρήση αντιστοιχίσεων μεταξύ των σχημάτων των κόμβων. Χρησιμοποιώντας αυτές τις αντιστοιχίσεις και ένα κατάλληλο αλγόριθμο επανεγγραφής, δύο γειτονικοί κόμβοι μπορούν να προωθούν ερωτήματα μεταξύ τους.

Η φύση των δομημένων δεδομένων που είναι αποθηκευμένα στο δίκτυο επιβάλλει αυστηρές μεθόδους επερώτησης και επανεγγραφής ερωτημάτων. Εντούτοις, συχνά ο χρήστης επιζητά πληροφορίες που είναι σημασιολογικά σχετικές με το ερώτημα που θέτει, παρά πληροφορίες που ικανοποιούν πολύ συγκεκριμένους δομικούς περιορισμούς. Οι διαθέσιμοι αλγόριθμοι επανεγγραφής αναφέρονται σε ερωτήματα που μπορούν να επανεγγραφούν πλήρως με τη χρήση ενός συνόλου αντιστοιχίσεων. Η προσέγγιση αυτή δεν είναι αρκετή για ένα περιβάλλον όπου οι κόμβοι αναζητούν και ικανοποιούνται από πληροφορίες που είναι σημασιολογικά όμοιες αλλά όχι απαραίτητα πανομοιότυπες με τις απαιτήσεις τους.

Χαρακτηριστικό παράδειγμα αποτελούν οι εφαρμογές ομότιμων κόμβων για την ανταλλαγή αρχείων. Ένα άλλο πιο σύγχρονο παράδειγμα, στο οποίο η σημασιολογική ομοιότητα παίζει ένα σημαντικό ρόλο, είναι η δημιουργία των υπηρεσιών κοινωνικής δικτύωσης (social networks), οι οποίες μοιάζουν με τις ανθρώπινες κοινωνικές ομάδες. Οι υπηρεσίες αυτές σχηματίζουν εικονικές κοινότητες, με τον κάθε συμμετέχοντα να θέτει τα δικά του χαρακτηριστικά και ενδιαφέροντα. Σε ένα τέτοιο δίκτυο, η αναζήτηση πανομοιότυπης πληροφορίας ανάμεσα στους χρήστες δεν είναι ένα ρεαλιστικό σενάριο.

1.2 Αντικείμενο Διπλωματικής

Με βάση τα προβλήματα που υπάρχουν στα συστήματα ομότιμων κόμβων, τα οποία αναφέρθηκαν στην παραπάνω ενότητα, προκύπτει η ανάγκη ενασχόλησης με την έννοια της σημασιολογικής ομοιότητας τόσο των σχημάτων των κόμβων όσο και των ερωτημάτων που προκύπτουν από τη διαδικασία επανεγγραφής σε σχέση με τα αρχικά [SKS+07]. Αντικείμενο της παρούσας διπλωματικής είναι η δημιουργία ενός συστήματος το οποίο θα χρησιμοποιεί διάφορα σημασιολογικά κριτήρια για να καταστήσει αποδοτικότερη την επικοινωνία μεταξύ των κόμβων του δικτύου. Έτσι,

χρησιμοποιώντας αυτά τα κριτήρια, οι κόμβοι θα μπορούν να ανακαλύψουν άλλους ομότιμους οι οποίοι έχουν παρόμοια ενδιαφέροντα με αυτούς. Παράλληλα, το σύστημα θα μπορεί να αποφασίσει, για ένα συγκεκριμένο ερώτημα, ποιος κόμβος μπορεί να το επανεγγράψει καλύτερα. Τα σχήματα των κόμβων και οι επανεγγραφές ερωτημάτων θα ιεραρχούνται με βάση την σημασιολογική τους ομοιότητα ως προς ένα σχήμα αναφοράς ή ένα αρχικό ερώτημα αντίστοιχα.

Δεν είναι, όμως, εφικτό να αντιμετωπιστεί το πρόβλημα της σημασιολογικής ομοιότητας στο πλαίσιο των δομημένων δεδομένων χωρίς κάποια επιπλέον σημασιολογική πληροφορία. Τα σχήματα των βάσεων δεδομένων και οι ανάλογες αντιστοιχίσεις δεν είναι αρκετές για να παρέχουν ικανοποιητικά σημασιολογικά μεταδεδομένα ώστε να είναι εφικτή μια ποιοτική λύση στο πρόβλημα. Για να αντιμετωπιστεί αυτό το ζήτημα, στο σύστημα που θα αναπτυχθεί βασιζόμαστε στις τεχνολογίες του Σημασιολογικού Ιστού. Συγκεκριμένα, θεωρούμε ένα PDMS συνοδευόμενο από μια οντολογία πεδίου, η οποία εξυπηρετεί ως ένα κοινό λεξιλόγιο μεταξύ των κόμβων. Πρέπει να σημειωθεί ότι η οντολογία αυτή διατηρεί τη βασική αρχή των αγνών συστημάτων ομότιμων κόμβων περί απουσίας κεντρικού σχήματος. Έτσι, οι κόμβοι δε χρειάζεται να ακολουθούν κάποιο περιορισμό, εκτός από το να χρησιμοποιούν όρους της οντολογίας για να επισημειώνουν σημασιολογικά τα στοιχεία τους, ώστε να ορίζουν με σαφήνεια την πληροφορία που παρέχουν στο δίκτυο.

Τέλος, για τις ανάγκες του συστήματος, απαιτούνται ορισμένες μετρικές για την αξιολόγηση της σημασιολογικής ομοιότητας. Έτσι, θα υιοθετηθούν οι έννοιες της ευστοχίας (recall) και της ακρίβειας (precision) από το πεδίο της Ανάκτησης Πληροφοριών (Information Retrieval).

1.2.1 Συνεισφορά

Η βασική συνεισφορά της παρούσας εργασίας συνοψίζεται στα ακόλουθα σημεία.

- Καταπιανόμαστε με το πρόβλημα της σημασιολογικής ομοιότητας μεταξύ των σχημάτων και των ερωτημάτων σε ένα PDMS εφοδιασμένο με μία καθολική οντολογία.
- Χρησιμοποιούνται οι έννοιες της ευστοχίας και της ακρίβειας για την ποσοτική εκτίμηση της σημασιολογικής ομοιότητας

- Το σύστημα που υλοποιείται χρησιμοποιεί ένα μέτρο σημασιολογικής ομοιότητας το οποίο συνδυάζει τη σημασιολογική επισημείωση κάθε κόμβου, τις αντιστοιχίσεις μεταξύ τους, καθώς και το εκάστοτε ερώτημα.
- Γίνεται προσπάθεια εξομοίωσης πραγματικών συνθηκών λειτουργίας ενός PDMS, τόσο ως προς τα δεδομένα όσο και προς το μέγεθος του δικτύου, ώστε να αξιολογηθεί πειραματικά η αποδοτικότητα του συστήματος μας.

1.3 Οργάνωση κειμένου

Το υπόλοιπο της διπλωματικής οργανώνεται στα ακόλουθα κεφάλαια.

Στο κεφάλαιο 2 παρουσιάζονται κάποιες βασικές έννοιες που χρησιμοποιήθηκαν στην παρούσα εργασία και είναι απαραίτητες για την κατανόηση του συστήματος. Ανάμεσά τους είναι τα δίκτυα ομότιμων κόμβων, ο Σημασιολογικός Ιστός και η ανάκτηση πληροφοριών.

Το κεφάλαιο 3 αναφέρεται σε κάποιες προϋπάρχουσες εργασίες που σχετίζονται με την προσπάθεια συνδυασμού των τεχνολογιών του Σημασιολογικού Ιστού με τα δίκτυα ομότιμων κόμβων.

Το κεφάλαιο 4 αφορά στην ανάλυση των απαιτήσεων του συστήματος. Το σύστημα χωρίζεται σε υποσυστήματα και παρουσιάζονται αναλυτικά οι λειτουργίες του καθενός από αυτά.

Το κεφάλαιο 5 αναφέρεται στη σχεδίαση του συστήματος. Αρχικά, δίνεται μια γενική περιγραφή των πακέτων στα οποία οργανώνονται οι κλάσεις της εφαρμογής και στη συνέχεια περιγράφονται αναλυτικά οι λειτουργίες κάθε κλάσης.

Στο κεφάλαιο 6 παρουσιάζονται ορισμένες λεπτομέρειες για την υλοποίηση του συστήματος. Περιγράφονται οι πιο ενδιαφέροντες από τους αλγορίθμους που υλοποιήθηκαν κατά την ανάπτυξη του συστήματος και γίνεται αναφορά στις πλατφόρμες, τα προγραμματιστικά εργαλεία και τις βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση.

Το κεφάλαιο 7 αναφέρεται στην πειραματική αξιολόγηση του συστήματος. Περιγράφεται το σύνολο δεδομένων και η τοπολογία του δικτύου που χρησιμοποιήθηκε και παρουσιάζονται τα αποτελέσματα που προέκυψαν από την εκτέλεση των πειραμάτων.

Στο κεφάλαιο 8 γίνεται μια σύνοψη της εργασίας, παρατίθενται τα συμπεράσματα και προτείνονται ιδέες για μελλοντικές επεκτάσεις.

Στο κεφάλαιο 9 δίνεται η βιβλιογραφία που χρησιμοποιήθηκε.

2

Θεωρητικό υπόβαθρο

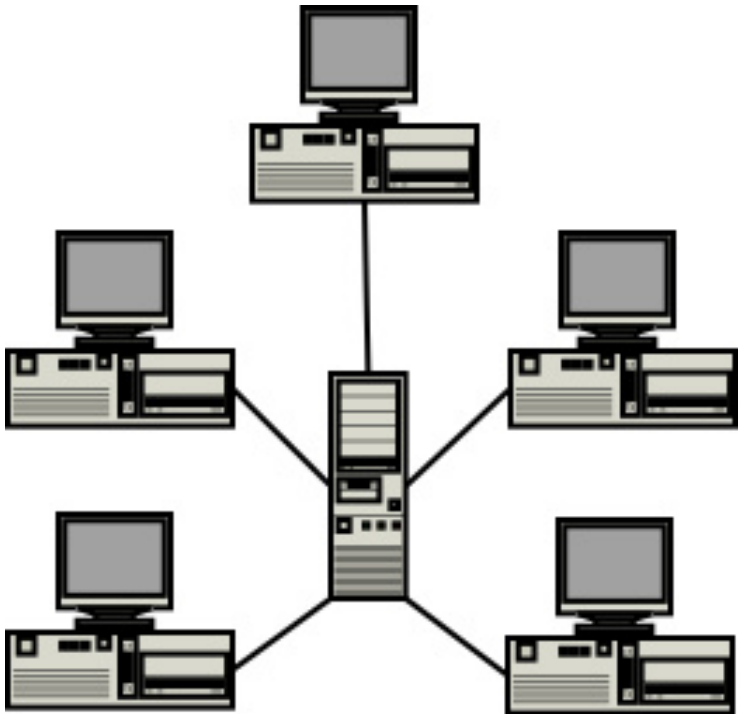
Στο κεφάλαιο αυτό παρουσιάζονται κάποιες βασικές έννοιες που χρησιμοποιήθηκαν στην παρούσα εργασία και είναι απαραίτητες για την κατανόηση του συστήματος. Στην ενότητα 2.1 περιγράφονται τα Ομότιμα Δίκτυα, στη 2.2 δίνεται μια παρουσίαση του Σημασιολογικού Ιστού, ενώ στη 2.3 αναπτύσσονται επιλεκτικά κάποιες έννοιες του τομέα της Ανάκτησης Πληροφοριών (Information Retrieval).

2.1 Δίκτυα Ομότιμων Κόμβων

Οι δύο γνωστότερες αρχιτεκτονικές για την επικοινωνία μεταξύ υπολογιστών είναι το μοντέλο του πελάτη-εξυπηρετητή (client-server architecture) [Ede94] [Sch96] και το μοντέλο ομότιμων κόμβων (peer-to-peer network). Αρχικά, θα γίνει μια σύντομη παρουσίαση του μοντέλου πελάτη-εξυπηρετητή και στη συνέχεια μια διεξοδική ανάλυση του μοντέλου ομότιμων κόμβων.

Το μοντέλο πελάτη-εξυπηρετητή διαχωρίζει τα συστήματα πελάτη από τα συστήματα εξυπηρετητή, τα οποία επικοινωνούν μέσω ενός δικτύου υπολογιστών. Μια εφαρμογή πελάτη-εξυπηρετητή είναι ένα κατανεμημένο σύστημα, το οποίο αποτελείται από το λογισμικό πελάτη, το οποίο εκκινεί μια σύνοδο (session) επικοινωνίας, και το λογισμικό εξυπηρετητή, που αναμένει για αιτήσεις από τον πελάτη. Κάθε στιγμιότυπο πελάτη στέλνει αιτήσεις δεδομένων σε έναν ή περισσότερους συνδεδεμένους σε αυτόν εξυπηρετητές, οι

οποίοι με τη σειρά τους δέχονται αυτές τις αιτήσεις, τις επεξεργάζονται και επιστρέφουν στον πελάτη τα δεδομένα που ζητήθηκαν. Χαρακτηριστικό παράδειγμα λογισμικού πελάτη είναι ο διαφυλλιστής Ιστού (web browser), με τον οποίο μπορούμε να έχουμε πρόσβαση σε οποιοδήποτε εξυπηρετητή Ιστού. Πολλά από τα βασικά πρωτόκολλα του Διαδικτύου (HTTP, SMTP, Telnet, DNS, …) χρησιμοποιούν επίσης αυτό το μοντέλο.



Εικόνα 2.1 Το μοντέλο πελάτη-εξυπηρετητή

Παρά τις ευκολίες που προσφέρει, το μοντέλο πελάτη-εξυπηρετητή υποφέρει από ορισμένα ουσιώδη προβλήματα. Η συμφόρηση κίνησης αποτελούσε πρόβλημα από την έναρξη λειτουργίας του μοντέλου. Όσο αυξάνεται ο αριθμός των πελατών, αυξάνεται αναπόφευκτα και ο αριθμός των ταυτόχρονων αιτήσεων από αυτούς και ο εξυπηρετητής μπορεί να υπερφορτωθεί σημαντικά. Επιπλέον, το μοντέλο στερείται της ευρωστίας που χαρακτηρίζει ένα δίκτυο ομότιμων. Αν ένας εξυπηρετητής τεθεί εκτός λειτουργίας, τα δεδομένα που προσφέρει παύουν να είναι διαθέσιμα στο δίκτυο και οι αιτήσεις των πελατών δεν μπορούν να εκπληρωθούν.

Όπως το μοντέλο του πελάτη-εξυπηρετητή ήρθε να ξεπεράσει τα προβλήματα των απολύτως κεντροποιημένων, μονολιθικών κεντρικών υπολογιστών (mainframe computers), έτσι και τα Δίκτυα Ομότιμων Κόμβων ήρθαν να δώσουν λύσεις στα βασικά προβλήματα του μοντέλου πελάτη-εξυπηρετητή.

2.1.1 Τι είναι τα Δίκτυα Ομότιμων Κόμβων

Τα δίκτυα ομότιμων κόμβων είναι μια κατηγορία δικτύων στην οποία κάθε υπολογιστής (κόμβος) έχει ίδια δικαιώματα και υποχρεώσεις. Δεν υπάρχει, δηλαδή, η έννοια του πελάτη και του εξυπηρετητή, καθώς κάθε κόμβος μπορεί να έχει και τους δύο ρόλους. Με τον τρόπο αυτό μεταφέρουμε την "ευφυΐα" του δικτύου στα άκρα του και όχι σε κάποιους συγκεκριμένους κόμβους. Ένας πιο τυπικός ορισμός θα έλεγε ότι τα δίκτυα ομότιμων είναι μια κλάση αρχιτεκτονικής δικτύου και εφαρμογών που χρησιμοποιούν κατανεμημένους πόρους με στόχο να πραγματοποιήσουν εργασίες δρομολόγησης και δικτύωσης με ένα αποκεντρωμένο και αυτό-οργανονούμενο τρόπο [EHK+06].

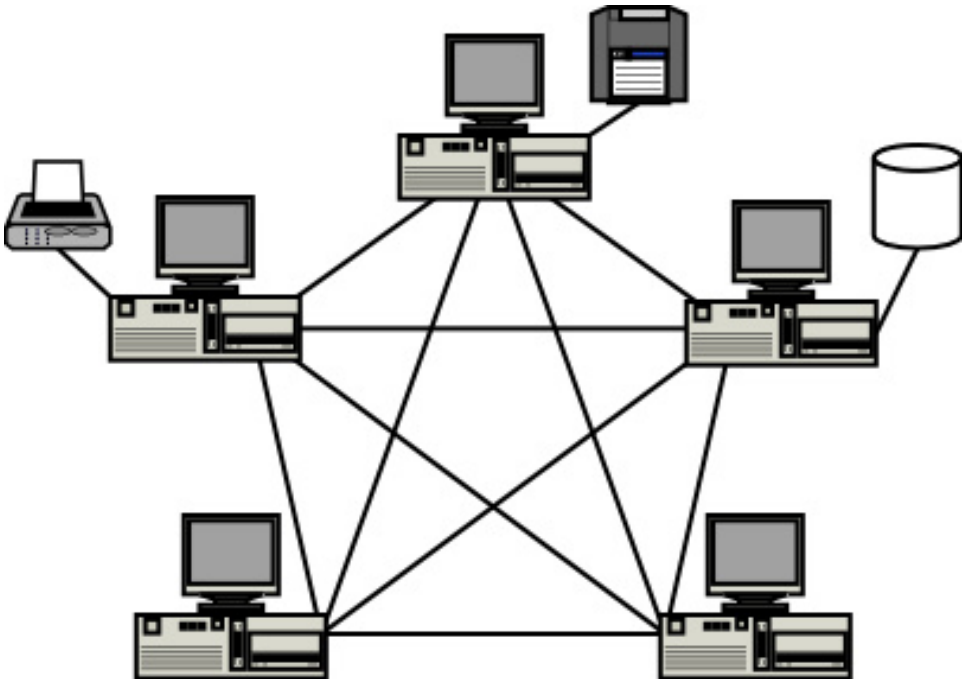
Τα δίκτυα αυτά είναι χρήσιμα για ποικίλους σκοπούς. Ο διαμοιρασμός αρχείων που περιέχουν ήχο, εικόνα, δεδομένα ή οτιδήποτε άλλο σε ψηφιακή μορφή είναι πολύ συνηθισμένος, ενώ και δεδομένα πραγματικού χρόνου, όπως η τηλεφωνική κίνηση, μπορούν να περαστούν χρησιμοποιώντας τεχνολογία ομότιμων.

Αρχές Λειτουργίας

Οι βασικές αρχές λειτουργίας των δικτύων ομότιμων κόμβων συνοψίζονται στα παρακάτω σημεία:

- Διαμοιρασμός πόρων και υπηρεσιών: σε ένα δίκτυο ομότιμων κάθε κόμβος ενεργεί τόσο ως πάροχος όσο και ως καταναλωτής υπηρεσιών και πόρων, όπως πληροφορίες, αρχεία, εύρος ζώνης, αποθηκευτικό χώρο και κύκλους επεξεργαστή. Με τη διανομή των πόρων μπορούν να εκτελεστούν εφαρμογές οι οποίες δεν θα μπορούσαν να οργανωθούν από έναν ενιαίο κόμβο. Αυτό ήταν η βασική κινητήρια ιδέα για ένα P2P σύστημα όπως το Napster [Nap08] ή το Seti@home [Set03].
- Αποκέντρωση: δεν υπάρχει κεντρική συντονιστική αρχή για την οργάνωση του δικτύου ή για τη χρήση και την επικοινωνία μεταξύ των κόμβων του δικτύου. Αυτό συνδέεται με το γεγονός ότι κανένας κόμβος δεν ασκεί κεντρικό έλεγχο πάνω στον άλλο. Δεδομένου αυτού, η επικοινωνία μεταξύ των κόμβων γίνεται απευθείας.

  Βέβαια, η πλήρης αποκέντρωση αφορά στα λεγόμενα "αγνά" (pure) δίκτυα ομότιμων, τα οποία αποτελούν και τα δίκτυα αναφοράς. Υπάρχουν, όμως, και τα υβριδικά δίκτυα, τα οποία θα αναλυθούν σε επόμενη ενότητα.
- Αυτονομία: κάθε κόμβος μπορεί αυτόνομα να αποφασίσει πότε και σε τι βαθμό θα κάνει τους πόρους του διαθέσιμους στο δίκτυο.



Εικόνα 2.2 Το μοντέλο ομότιμων κόμβων

Πλεονεκτήματα

Τα πλεονεκτήματα των δικτύων ομότιμων, σχετίζονται, όπως είναι φυσικό, με τις αρχές λειτουργίας τους.

Ένας σημαντικός στόχος στα συστήματα ομότιμων κόμβων είναι ότι όλοι οι κόμβοι προσφέρουν πόρους, συμπεριλαμβανομένου του εύρους ζώνης, του αποθηκευτικού χώρου και της υπολογιστικής δύναμης. Έτσι, καθώς νέοι κόμβοι προσθέτονται στο δίκτυο και οι απαιτήσεις στο σύστημα αυξάνονται, η συνολική χωρητικότητα του δικτύου επίσης αυξάνεται. Κάτι τέτοιο δε συμβαίνει στο μοντέλο πελάτη-εξυπηρετητή, όπου η είσοδος νέων χρηστών μπορεί να σημαίνει μειωμένη ταχύτητα μεταφοράς δεδομένων για όλους τους χρήστες.

Η κατανεμημένη φύση των δικτύων ομότιμων αυξάνει και την ευρωστία του συστήματος σε περίπτωση κάποιας αποτυχίας, μιας και διατηρούνται πολλαπλά αντίγραφα των δεδομένων στους διάφορους κόμβους. Στην περίπτωση δε των “αγνών” συστημάτων ομότιμων, οι χρήστες δε βασίζονται σε κάποιο κεντροποιημένο εξυπηρετητή ευρετηρίου κι έτσι δεν υπάρχει περίπτωση αποτυχίας στο σύστημα.

Επικαλύπτοντα Δίκτυα

Ένα επικαλύπτον δίκτυο [sol08] είναι ένα δίκτυο που είναι “χτισμένο” πάνω σε ένα άλλο. Οι κόμβοι στο δίκτυο αυτό μπορεί να θεωρηθεί ότι συνδέονται με εικονικές (virtual) ή λογικές (logical) συνδέσεις, καθεμιά από τις οποίες αντιστοιχεί σε ένα μονοπάτι, πιθανώς μέσω

πολλών φυσικών συνδέσεων στο υποκείμενο δίκτυο. Για παράδειγμα, οι dial-up συνδέσεις στο Διαδίκτυο αποτελούν ένα επικαλύπτον δίκτυο πάνω από το τηλεφωνικό δίκτυο.

Τα επικαλύπτοντα δίκτυα μπορούν να κατασκευαστούν με σκοπό να επιτρέπουν τη δρομολόγηση μηνυμάτων σε προορισμούς οι οποίοι δεν έχουν IP-διεύθυνση. Τα επικαλύπτοντα δίκτυα έχουν επίσης προταθεί σαν ένας τρόπος να βελτιωθεί η δρομολόγηση στο Διαδίκτυο, έτσι ώστε μέσω της ποιότητας των υπηρεσιών (QoS) να εγγυάται καλύτερη ποιότητα στα ρεύματα πολυμέσων (streaming media). Το πλεονέκτημα των δικτύων αυτών σε σχέση με άλλες τεχνολογίες που έχουν προταθεί είναι ότι δεν απαιτούνται αλλαγές στα ήδη υπάρχοντα δίκτυα.

Τέλος, να σημειωθεί ότι το επικαλύπτον δίκτυο δεν μπορεί να ελέγξει πώς δρομολογούνται τα πακέτα στο υποκείμενο δίκτυο, αλλά μπορεί να ελέγξει την ακολουθία των κόμβων του επικαλύπτοντος δικτύου που θα μεσολαβήσουν μέχρι το μήνυμα να φτάσει στον προορισμό του.

Τα δίκτυα ομότιμων χρησιμοποιούνται για να δημιουργήσουν επικαλύπτοντα δίκτυα (overlay networks) τα οποία προσφέρουν υπηρεσίες στο IP επίπεδο που βρίσκεται ένα επίπεδο πιο κάτω [Cro08].

2.1.2 Τύποι Δικτύων Ομότιμων Κόμβων και χαρακτηριστικά παραδείγματα

Υπάρχουν ποικίλες κατηγοριοποιήσεις [AS04] για τα δίκτυα ομότιμων, ανάλογες με το κριτήριο ταξινόμησης που χρησιμοποιούμε.

Κατηγοριοποίηση με βάση τις εφαρμογές

Όσον αφορά στις εφαρμογές για τις οποίες χρησιμοποιείται κάθε δίκτυο ομότιμων, προκύπτουν οι παρακάτω κατηγορίες:

- Επικοινωνία και συνεργασία: Η κατηγορία αυτή περιλαμβάνει συστήματα που παρέχουν την υποδομή για να διευκολύνουν την απευθείας, συνήθως πραγματικού χρόνου, επικοινωνία και συνεργασία μεταξύ των κόμβων. Περιλαμβάνει εφαρμογές συνομιλίας (chat), στιγμιαίων μηνυμάτων (instant messaging) και τηλεφωνίας μέσω διαδικτύου (VoIP), όπως οι εφαρμογές AOL, ICQ, MSN, Jabber και Skype.
- Κατανεμημένος υπολογισμός: Σε αυτή την κατηγορία περιλαμβάνονται συστήματα που εκμεταλλεύονται την υπολογιστική δύναμη (κύκλοι CPU) των κόμβων. Αυτό επιτυγχάνεται διασπώντας μια εργασία σε μικρότερα τμήματα και διανέμοντάς την σε διαφορετικούς κόμβους. Καθένας από τους κόμβους εκτελεί την εργασία που του δόθηκε και επιστρέφει τα αποτελέσματα. Παραδείγματα τέτοιων συστημάτων είναι τα Seti@home [Set03] και genome@home [Gen03].

- Υποστήριξη υπηρεσιών Διαδικτύου: Ένας αριθμός διαφορετικών εφαρμογών βασισμένων σε συστήματα ομότιμων έχουν χρησιμοποιηθεί για να υποστηρίξουν μια ποικιλία υπηρεσιών διαδικτύου, όπως συστήματα πολυδιανομής (multicast) ομότιμων και υποδομές ασφάλειας εφαρμογών που προσφέρουν προστασία από ιούς και επιθέσεις άρνησης υπηρεσίας (DoS attacks).
- Συστήματα Βάσεων Δεδομένων: Σημαντική προσπάθεια έχει γίνει για το σχεδιασμό κατανεμημένων βάσεων δεδομένων βασισμένων σε υποδομές ομότιμων κόμβων. Στο LRM (Τοπικό Σχεσιακό Μοντέλο) [BGK+02] το σύνολο των δεδομένων είναι αποθηκευμένα σε ένα δίκτυο ομότιμων που αποτελείται από ασυνεπείς τοπικές σχεσιακές βάσεις διασυνδεδεμένες με σύνολα κανόνων μετάφρασης και σημασιολογικών εξαρτήσεων μεταξύ τους. Το PIER [HHL+03] είναι μια κλιμακούμενη κατανεμημένη μηχανή ερωτημάτων χτισμένη πάνω από ένα δίκτυο ομότιμων, που επιτρέπει να αποτιμώνται σχεσιακά ερωτήματα δια μέσου χιλιάδων υπολογιστών. Άλλο χαρακτηριστικό παράδειγμα είναι το σύστημα Piazza, για το οποίο γίνεται αναλυτική παρουσίαση στο κεφάλαιο 3.
- Διανομή περιεχομένου: Τα περισσότερα από τα τρέχοντα δίκτυα ομότιμων υπάγονται σε αυτή την κατηγορία, η οποία περιλαμβάνει συστήματα και υποδομές σχεδιασμένα για την ανταλλαγή ψηφιακών μέσων και άλλων ειδών δεδομένων μεταξύ των χρηστών. Τα δίκτυα ομότιμων κόμβων διανομής περιεχομένου κυμαίνονται από απλές εφαρμογές άμεσης ανταλλαγής αρχείων μέχρι πιο περίπλοκα συστήματα που δημιουργούν ένα μέσο κατανεμημένης αποθήκευσης για ασφαλή και αποδοτική έκδοση, οργάνωση, αναζήτηση, ανανέωση και ανάκτηση δεδομένων. Μερικά από αυτά είναι τα: Napster, Gnutella, Kazaa, Freenet, Chord, Groove.

Κατηγοριοποίηση με βάση την κεντροποίηση του επικαλύπτοντος δικτύου

Παρόλο που στην πιο αγνή τους μορφή τα επικαλύπτοντα δίκτυα ομότιμων πρέπει να είναι πλήρως αποκεντρωμένα, πρακτικά αυτό δε συμβαίνει πάντα, για διάφορους λόγους, όπως είναι η οργάνωση και η απόδοση του συστήματος. Διακρίνουμε τρεις βασικές κατηγορίες που αφορούν στο βαθμό κεντροποίησης (centralization).

- Πλήρως αποκεντρωμένες (“αγνές”) αρχιτεκτονικές: Όλοι οι κόμβοι του δικτύου εκτελούν ακριβώς τις ίδιες εργασίες, δρώντας τόσο ως πελάτες όσο και ως εξυπηρετητές, χωρίς να υπάρχει κεντρικός συντονισμός των δραστηριοτήτων.
- Εν μέρει κεντροποιημένες αρχιτεκτονικές: Η βάση είναι η ίδια με τα πλήρως αποκεντρωμένα συστήματα. Εντούτοις, ορισμένοι κόμβοι παίζουν ένα σημαντικότερο ρόλο δρώντας ταυτόχρονα ως τοπικά κεντρικά ευρετήρια για τα αρχεία που διαμοιράζονται στους τοπικούς κόμβους. Ο τρόπος με τον οποίο αυτοί οι

υπερ-κόμβοι επιφορτίζονται με αυτό το ρόλο από το δίκτυο ποικίλει μεταξύ των διαφόρων συστημάτων. Είναι σημαντικό, όμως, να αναφερθεί ότι αυτοί οι κόμβοι δεν αποτελούν μοναδικά σημεία αποτυχίας για το σύστημα, αφού το δίκτυο ορίζει τους υπερ-κόμβους δυναμικά. Έτσι, αν κάποιος τεθεί εκτός λειτουργίας, αντικαθίσταται από κάποιον άλλο.

- Υβριδικές αποκεντρωμένες αρχιτεκτονικές: Σε αυτά τα συστήματα, υπάρχει ένας κεντρικός εξυπηρετητής που διευκολύνει την αλληλεπίδραση μεταξύ των κόμβων, διατηρώντας καταλόγους από μεταδεδομένα που περιγράφουν τα διαμοιραζόμενα αρχεία του συστήματος. Αν και η από-άκρο-σε-άκρο αλληλεπίδραση και ανταλλαγή αρχείων μπορεί να γίνει απευθείας ανάμεσα σε δύο κόμβους, ο κεντρικός εξυπηρετητής διευκολύνει τη δραστηριότητα εκτελώντας τις αναζητήσεις και αναγνωρίζοντας τους κόμβους στους οποίους είναι αποθηκευμένα τα απαιτούμενα αρχεία. Προφανώς, σε αυτά τα συστήματα υπάρχει σημείο μοναδικής αποτυχίας (ο εξυπηρετητής). Το γεγονός αυτό κάνει τα συστήματα αυτά μη κλιμακούμενα (ο εξυπηρετητής υπερφορτώνεται) και ευάλωτα σε τεχνικές βλάβες ή κακόβουλες επιθέσεις.

Κατηγοριοποίηση με βάση τη δομή του δικτύου

Όσον αφορά στη δομή, ενδιαφερόμαστε για το αν το επικαλύπτον δίκτυο δημιουργείται μη ντετερμινιστικά καθώς οι κόμβοι και το περιεχόμενο προστίθεται στο δίκτυο ή αν η δημιουργία του βασίζεται σε συγκεκριμένους κανόνες. Διακρίνουμε τρεις κατηγορίες.

- Αδόμητα δίκτυα: Η τοποθέτηση του περιεχομένου (των αρχείων) είναι εντελώς ασυσχέτιστη με το τοπολογία του επικαλύπτοντος δικτύου. Οι μηχανισμοί αναζήτησης σε ένα αδόμητο δίκτυο κυμαίνονται από εξαντλητικές (brute force) τεχνικές, όπως είναι η πλημμύρα του δικτύου με ερωτήματα που προωθούνται με ένα τρόπο πρώτα-σε-βάθος (depth-first) ή πρώτα-σε-πλάτος (breadth-first) μέχρις ότου εντοπιστούν τα επιθυμητά δεδομένα, μέχρι πιο πολύπλοκες και δεσμεύουσες πόρους τεχνικές, όπως η χρήση τυχαίων δρόμων ή ευρετηρίων δρομολόγησης. Οι μηχανισμοί αναζήτησης στα αδόμητα δίκτυα έχουν αναμενόμενες επιπλοκές, κυρίως όσον αφορά σε θέματα διαθεσιμότητας, κλιμάκωσης και μονιμότητας.

  Τα αδόμητα συστήματα είναι γενικά κατάλληλα για να δεχτούν προσωρινές ομάδες κόμβων. Μερικά αντιπροσωπευτικά παραδείγματα αδόμητων συστημάτων είναι τα Napster, Publius, Gnutella, Kazaa και Edutella.

- Δομημένα δίκτυα: Προέκυψαν κυρίως στην προσπάθεια να αντιμετωπιστούν τα θέματα κλιμάκωσης τα οποία αντιμετώπισαν τα αδόμητα συστήματα. Στα δομημένα δίκτυα, η τοπολογία του επικαλύπτοντος δικτύου είναι άμεσα ελεγχόμενη και τα

αρχεία (ή δείκτες σε αυτά) τοποθετούνται σε αυστηρά καθορισμένες θέσεις. Αυτά τα συστήματα παρέχουν μια αντιστοίχιση μεταξύ του περιεχομένου και της τοποθεσίας, στη μορφή ενός κατανεμημένου πίνακα δρομολόγησης, έτσι ώστε τα ερωτήματα να μπορούν να προωθηθούν αποτελεσματικά στον κόμβο με το επιθυμητό περιεχόμενο. Ένα μειονέκτημα των δομημένων συστημάτων είναι ότι είναι δύσκολο να διατηρείται η δομή που απαιτείται για αποτελεσματική δρομολόγηση μηνυμάτων στην περίπτωση ενός πολύ δυναμικού πληθυσμού κόμβων, όπου οι κόμβοι εισέρχονται και εξέρχονται από το δίκτυο σε ταχείς ρυθμούς. Τυπικά παραδείγματα δομημένων συστημάτων είναι τα Chord, CAN, PAST και Tapestry.

- Ελαφρώς (loosely) δομημένα δίκτυα: Είναι μια κατηγορία που βρίσκεται νοηματικά ανάμεσα στις δύο που ήδη αναφέρθηκαν. Αν και η τοποθεσία του περιεχομένου δεν είναι ακριβώς καθορισμένη, επηρεάζεται από στοιχεία δρομολόγησης. Τυπικό παράδειγμα είναι το Freenet.

Στον ακόλουθο πίνακα συνοψίζονται οι κατηγορίες που αναφέρθηκαν με παραδείγματα δικτύων ομότιμων κόμβων. Παρατηρούμε ότι όλα τα δομημένα συστήματα είναι πλήρως αποκεντρωμένα.

	Κεντροποίηση		
	Υβριδική	**Μερική**	**Καμία**
Αδόμητα	Napster, Publius	Kazaa, Mprpheus, Gnutella, Edutella	Gnutella, FreeHaven
Δομημένα			Chord, CAN, Tapestry, Pastry

Πίνακας 2.1 Βασική κατηγοριοποίηση των δικτύων ομοτίμων με παραδείγματα

Κατηγοριοποίηση με βάση τη μορφή των δεδομένων

Μια τελευταία κατηγοριοποίηση των δικτύων ομότιμων κόμβων μπορεί να γίνει με βάση τη μορφή των δεδομένων που είναι αποθηκευμένα στο δίκτυο. Διακρίνουμε δύο βασικές κατηγορίες δεδομένων.

- Αδόμητα δεδομένα (unstructured data): Πρόκειται για δεδομένα που είτε δεν έχουν κάποια δομή είτε έχουν δομή η οποία δεν είναι εύκολα χρησιμοποιήσιμη από μια εφαρμογή υπολογιστή. Για να μπορέσουν να αξιοποιηθούν από τον υπολογιστή απαιτείται ανθρώπινη παρέμβαση (π.χ. ετικέτες και λέξεις-κλειδιά σε κείμενα).
- Δομημένα δεδομένα (structured data): Πρόκειται για δεδομένα που ακολουθούν μια δομή δεδομένων. Τα δομημένα δεδομένα μπορούν να αξιοποιηθούν από τους

υπολογιστές με τεχνολογία που επιτρέπει την επερώτηση πάνω σε προκαθορισμένες δομές δεδομένων. Χαρακτηριστικό παράδειγμα δομημένων δεδομένων είναι αυτά που οργανώνονται σε βάσεις δεδομένων.

2.2 Σημασιολογικός Ιστός

2.2.1 Βασικά Χαρακτηριστικά

Ο Παγκόσμιος Ιστός έκανε διαθέσιμα πολύ μεγάλα ποσά πληροφορίας και αποτελεί μια πολύ επιτυχημένη ιστορία τόσο από την πλευρά της διαθέσιμης πληροφορίας όσο και από την πλευρά του αριθμού των ανθρώπων που τον χρησιμοποιούν. Αυτή η επιτυχία βασίζεται σε μεγάλο βαθμό στην απλότητά του, δίνοντας στους προγραμματιστές, στους παρόχους πληροφοριών και στους απλούς χρήστες εύκολη πρόσβαση σε νέο περιεχόμενο. Αυτή η απλότητα, όμως, δημιουργεί προβλήματα για την περαιτέρω ανάπτυξη και εξέλιξή του.

Ο Σημασιολογικός Ιστός [BKB+06] αποτελεί μια επέκταση του υπάρχοντος διαδικτύου. Αποτελεί τη μεγαλύτερη προσπάθεια αυτόματης ενοποίησης συστημάτων ώστε να συνεργάζονται διαλειτουργικά σε παγκόσμιο επίπεδο. Ο Tim – Berners-Lee που επινόησε τον Παγκόσμιο Ιστό το 1989, είχε το όραμα ενός ιστού δεδομένων αυτόματα επεξεργάσιμων από τις εφαρμογές, βάσει του νοήματος και όχι της μορφής της πληροφορίας [Ber01].

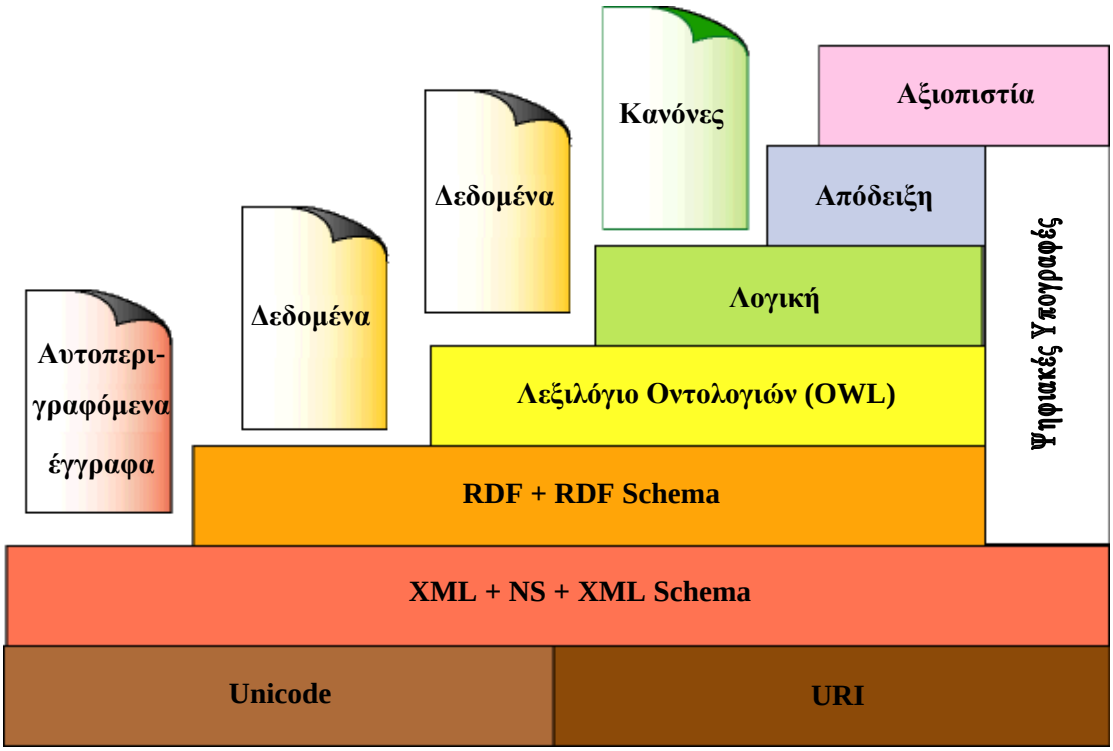
Έτσι, το κέντρο βάρους των περιεχομένων του διαδικτύου μετατοπίζεται συνεχώς από το ελεύθερο κείμενο που είναι πλήρως κατανοητό μόνο από τον άνθρωπο, προς την ημιδομημένη ή και πλήρως δομημένη πληροφορία, η οποία μπορεί να γίνει αυτόματα κατανοητή από διαδικτυακές εφαρμογές, όπως είναι οι διαδικτυακές υπηρεσίες (web services) και οι ευφυείς πράκτορες. Ο Σημασιολογικός Ιστός αποτελεί πρωτοβουλία της Κοινοπραξίας του Παγκόσμιου Ιστού (W3C) και η σχετική δραστηριότητα (Semantic Web Activity) έχει δημιουργηθεί με σκοπό το σχεδιασμό προδιαγραφών και την ανοιχτή ανάπτυξη της τεχνολογίας μέσω της συνεργασίας.

Στο Σημασιολογικό Ιστό η πληροφορία που παρουσιάζεται στο χρήστη ορίζεται με τέτοιο τρόπο ώστε να είναι κατανοητή όχι μόνο από τους ανθρώπους αλλά και από προγράμματα, ενισχύοντας έτσι τη διαλειτουργικότητα της επεξεργασίας των πληροφοριών μεταξύ των προγραμμάτων, αλλά και διευκολύνοντας τη λειτουργικότητα της χρήσης του διαδικτύου από τους ανθρώπους με τη βοήθεια των προγραμμάτων. Ο Σημασιολογικός Ιστός βασίζεται στην ιδέα της οργάνωσης και διασύνδεσης της πληροφορίας που υπάρχει στο διαδίκτυο, ώστε να

μπορεί να χρησιμοποιηθεί πιο αποτελεσματικά για την ανακάλυψη, αυτοματοποίηση, ομαδοποίηση και επαναχρησιμοποίησή της από διαφορετικές μεταξύ τους διαδικτυακές εφαρμογές.

Αρχιτεκτονική Σημασιολογικού Ιστού

Ο Σημασιολογικός Ιστός αποτελείται από στρώματα (layers). Η χρήση διαστρωματωμένης αρχιτεκτονικής, όπως είναι γνωστό, έχει ποικίλα πλεονεκτήματα. Κάθε στρώμα υλοποιεί μια λειτουργικότητα, χρησιμοποιώντας και επεκτείνοντας τη λειτουργικότητα και τις τεχνολογίες που παρέχονται από τα χαμηλότερα στρώμα. Έτσι λοιπόν στα χαμηλά επίπεδα υλοποιούνται λειτουργίες οι οποίες είναι πολύ κοντά στον Παγκόσμιο Ιστό και στις μηχανές, όπως είναι οι τεχνολογίες που ασχολούνται με τον καθορισμό και την αναγνώριση των πόρων (URIs), ενώ καθώς ανεβαίνουμε στην ιεραρχία των επιπέδων διατρέχουμε επίπεδα τα οποία υλοποιούν λειτουργικότητες αναπαράστασης γνώσης, πολύπλοκης συλλογιστικής και εμπιστοσύνης πλησιάζοντας στην ανθρώπινη γνώση και σκέψη [Sto06] [Ant04]. Τα επίπεδα της αρχιτεκτονικής του Σημασιολογικού Ιστού δίνονται σχηματικά στην Εικόνα 2.3.



Εικόνα 2.3 Η διαστρωματωμένη αρχιτεκτονική του Σημασιολογικού Ιστού

Οντολογίες

Οντολογία είναι η αυστηρά μαθηματική περιγραφή ενός πεδίου γνώσης και περιλαμβάνει ένα σύνολο από όρους και τις σημασιολογικές αντιστοιχίσεις μεταξύ τους. Οι έννοιες

περιλαμβάνουν κλάσεις αντικειμένων, δηλαδή έννοιες-πρότυπα σχετικές με αντικείμενα. Οι συσχετίσεις συνήθως αφορούν σε ιεραρχικές εξαρτήσεις μεταξύ των όρων. Άλλες πληροφορίες που μπορεί να υπάρχουν σε μια οντολογία είναι οι ιδιότητες των όρων, περιορισμοί γύρω από αυτές , σχέσεις ισοδυναμίας και διαχωρισμού, καθώς και σημασιολογικοί συσχετισμοί μεταξύ των όρων με τη χρήση της λογικής.

Οι οντολογίες είναι μια από τις ουσιαστικότερες τεχνολογίες του Σημασιολογικού Ιστού και γενικότερα της διαχείρισης δομημένης γνώσης στα πλαίσια των κατανεμημένων συστημάτων. Προσφέρουν μηχανικά-επεξεργάσιμη σημασιολογία των δεδομένων. Η πληροφορία μπορεί έτσι να γίνει κατανοητή από τους υπολογιστές και να βοηθήσει τους χρήστες να ψάξουν, να εξάγουν, να μεταφράσουν και να επεξεργαστούν τις πληροφορίες που είναι διαθέσιμες [LSH+08].

2.2.2 Οι γλώσσες του Σημασιολογικού Ιστού

Στη διαστρωματωμένη αρχιτεκτονική του Σημασιολογικού Ιστού που φαίνεται στην Εικόνα 2.3, το σημασιολογικό κομμάτι ενεργοποιείται από μια στοίβα εξελισσόμενων γλωσσών [DKD+05] [Sto07], οι οποίες διαθέτουν τα πλεονεκτήματα τόσο των φορμαλισμών της αναπαράστασης γνώσης (Knowledge Representation) όσο και των μεθόδων εννοιολογικής μοντελοποίησης των βάσεων δεδομένων. Το RDF (Resource Description Framework) [KC04] [MM04] προσφέρει ένα απλό μοντέλο αναφοράς με γράφους, το RDFS (RDF Schema) [BG04] προσφέρει ένα απλό λεξιλόγιο και αξιώματα για αντικειμενοστρεφή μοντελοποίηση και η OWL (Web Ontology Language) [MH04] προσφέρει επιπλέον λειτουργίες οντολογιών βασισμένες στη γνώση και αξιώματα. Παρακάτω δίνεται μια πιο αναλυτική παρουσίαση για καθεμιά από αυτές.

Το Πλαίσιο Περιγραφής Διαδικτυακών Πόρων RDF

Το RDF είναι μια γλώσσα για την αναπαράσταση πληροφοριών που αναφέρονται σε πόρους του Παγκόσμιου Ιστού. Προορίζεται κυρίως για την παρουσίαση μεταδεδομένων για τους πόρους αυτούς. Γενικεύοντας την έννοια του διαδικτυακού πόρου, το RDF μπορεί να χρησιμοποιηθεί για να αναπαραστήσει και πληροφορίες που μπορούν να υπάρξουν στο Διαδίκτυο, ακόμη κι αν δεν μπορούν άμεσα να ανακτηθούν από αυτό.

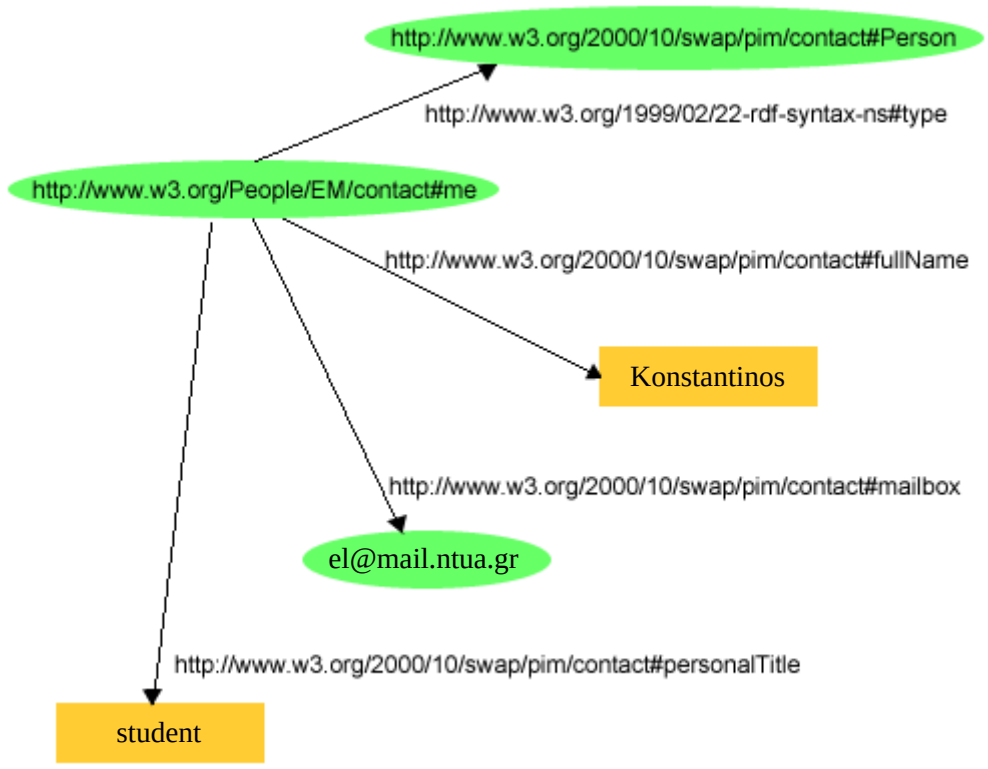
Το RDF προορίζεται για καταστάσεις όπου την πληροφορία πρέπει να την επεξεργαστεί μια εφαρμογή, παρά για να παρουσιαστεί σε έναν άνθρωπο. Παρέχει ένα πλαίσιο ώστε να εκφραστεί αυτή η πληροφορία και να μπορέσει να ανταλλαχθεί μεταξύ εφαρμογών χωρίς απώλεια σημασίας. Οι σχεδιαστές λογισμικού μπορούν να εκμεταλλευτούν συνήθεις RDF

parsers και εργαλεία επεξεργασίας. Έτσι, οι πληροφορίες μπορεί να είναι εκμεταλλεύσιμες και για εφαρμογές διαφορετικές από αυτές για τις οποίες αρχικά δημιουργήθηκαν.

Το RDF βασίζεται στην ιδέα ότι οι πόροι οι οποίοι πρέπει να περιγραφούν έχουν ιδιότητες οι οποίες έχουν συγκεκριμένοι τιμή. Μια RDF πρόταση αποτελείται από μια τριάδα που περιέχει ένα υποκείμενο (subject), ένα κατηγόρημα (predicate, καλείται και ιδιότητα) και ένα αντικείμενο (object). Υποκείμενο είναι ο πόρος που περιγράφουμε, κατηγόρημα η ιδιότητα που του αποδίδουμε και αντικείμενο η τιμή της ιδιότητας. Συντακτικά, οι τριάδες αυτές δηλώνονται από μια διατεταγμένη τριάδα “υποκείμενο κατηγόρημα αντικείμενο.”.

Χρησιμοποιούνται τα λεγόμενα αναγνωριστικά πόρων (URI) ώστε να αναφερόμαστε με μοναδικό τρόπο σε ένα διαδικτυακό πόρο.

Εκτός από την αναπαράσταση με τις τριάδες, το RDF χρησιμοποιεί και γράφους από κόμβους και ακμές για να αναπαραστήσει τις RDF προτάσεις. Έστω, για παράδειγμα, ότι υπάρχει ένα άτομο με URI “http://www.w3.org/People/EM/contact#me”, του οποίου το όνομα είναι Konstantinos, το e-mail του el@mail.ntua.gr και του οποίου ο τίτλος είναι student. Οι πληροφορίες αυτές μπορούν να αναπαρασταθούν με το γράφο στην Εικόνα 2.4:



Εικόνα 2.4 Ένας RDFγράφος που περιγράφει τον “Konstantinos”

Όπως φαίνεται και στο παραπάνω σχήμα, χρησιμοποιούμε κόμβους για να αναπαραστήσουμε το υποκείμενο και το αντικείμενο και ακμές για να τα κατηγορήματα. Επιπλέον,

παρατηρούμε ότι το αντικείμενο μιας πρότασης (οι τιμές των ιδιοτήτων) μπορεί να είναι είτε διαδικτυακοί πόροι (για τους οποίους δίνουμε το URI τους) είτε τιμές δεδομένων, όπως ακέραιοι, ημερομηνίες ή συμβολοσειρές. Η χρήση πόρων στη θέση των αντικειμένων μας δίνει τη δυνατότητα να δημιουργήσουμε αλυσίδες από τριάδες RDF όπου το αντικείμενο της μιας τριάδας εμφανίζεται ως υποκείμενο της επόμενης.

Μια τρίτη μορφή αναπαράστασης των RDF προτάσεων είναι το λεγόμενο RDF/XML, που χρησιμοποιείται για την καταγραφή και την ανταλλαγή τέτοιων γράφων:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

</rdf:RDF>
```

Εικόνα 2.5 RDF/XML περιγραφή του "Konstantinos"

Όπως και η HTML, το RDF/XML είναι επεξεργάσιμο από μηχανή και χρησιμοποιώντας URIs μπορεί να συνδέσει κομμάτια πληροφοριών στον Ιστό. Εντούτοις, αντίθετα από το συμβατικό υπερκείμενο, τα URIs μπορούν να αναφερθούν σε οποιοδήποτε ευπροσδιόριστο αντικείμενο, συμπεριλαμβανομένων και εκείνων που μπορεί να μην είναι άμεσα ανακτήσιμα από τον Ιστό (όπως το πρόσωπο Konstantinos).

Για λόγους συντομίας, συχνά οι διευθύνσεις αντικαθίστανται από συντομογραφίες κι έτσι δε χρειάζεται να γίνεται παράθεση ολόκληρης της διεύθυνσης κάθε φορά. Ορίζεται ένα πρόθεμα για τα αρχικά τμήματα κάθε διεύθυνσης, το οποίο ακολουθείται από ένα τοπικό όνομα που συμπληρώνει τη διεύθυνση.

Αν λοιπόν στο παραπάνω παράδειγμα αντικαταστήσουμε το “http://www.w3.org/2000/10/swap/pim/contact” με “co”, “http://www.w3.org/People/EM/contact” με “pe” και το “http://www.w3.org/1999/02/22-rdf-syntax-ns” με “rdf”, και χρησιμοποιήσουμε την αναπαράσταση των τριάδων (δίνεται αυτή η αναπαράσταση για λόγους πληρότητας), προκύπτει το παρακάτω:

```
<pe:me>        <co:fullName>          “Konstantinos” .
<pe:me>        <co:mailbox>           <el@mail.ntua.gr> .
<pe:me>        <co:personalTitle>     “student” .
<pe:me>        <rdf:type>             <co:Person> .
```

Εικόνα 2.6 Τριάδες για την περιγραφή του "Konstantinos" με χρήση συντομεύσεων

Το RDF περιλαμβάνει αρκετές ακόμη έννοιες, όπως η έννοια του κενού κόμβου, η δήλωση τύπων δεδομένων για κάποια λεκτικά (typed literals), στοιχεία για την περιγραφή συνόλων (bags), ακολουθιών (sequences), εναλλακτικών επιλογών (alternatives) και συλλογών (collections) και άλλα. Οι έννοιες αυτές ξεφεύγουν, όμως, από τα πλαίσια της συνοπτικής αυτής παρουσίασης και μπορούν να αναζητηθούν στη σχετική βιβλιογραφία.

Συμπερασματικά, το RDF παρέχει ένα τρόπο αναπαράστασης δηλώσεων εύκολα επεξεργάσιμο από προγράμματα στο διαδίκτυο, διατηρώντας παράλληλα την αναγνωσιμότητα των δηλώσεων και από ανθρώπους.

Η γλώσσα περιγραφής λεξιλογίου RDFS

Το RDF μας παρέχει τη δυνατότητα να δημιουργήσουμε απλές προτάσεις για τους πόρους τους οποίους θέλουμε να περιγράψουμε χρησιμοποιώντας ιδιότητες, τιμές και URIs για τον προσδιορισμό των συστατικών που συμμετέχουν σε μια πρόταση. Εντούτοις, δεν παρέχει τη δυνατότητα να ορίσουμε και να περιγράψουμε ένα επιπλέον λεξιλόγιο το οποίο πιθανόν να επιθυμούμε να χρησιμοποιήσουμε στις εφαρμογές μας για να ορίσουμε, για παράδειγμα, τις κλάσεις των πόρων και τις ιδιότητες που αναφέρονται στους πόσους αυτούς. Η γλώσσα η οποία παρέχει τη λειτουργικότητα αυτή είναι το RDF Schema (RDFS). Ουσιαστικά το RDFS παρέχει ένα επιπλέον λεξιλόγιο πάνω σε αυτό του RDF το οποίο περιλαμβάνει στοιχεία τα οποία προορίζονται στο να προσδώσουν την επιπρόσθετη αυτή λειτουργικότητα.

Οι πόροι στο λεξιλόγιο του RDFS έχουν URIs με πρόθεμα `http://www.w3.org/2000/01/rdf-schema#` από σύμβαση συνδέεται με τη συντόμευση “`rdfs:`”. Οι περιγραφές λεξιλογίου για το RDFS είναι αποδεκτοί RDF γράφοι κι έτσι μπορούν να αναγνωριστούν ακόμη κι από ένα πρόγραμμα που δεν υποστηρίζει το λεξιλόγιο του RDFS. Στην περίπτωση αυτή, απλά δεν μπορούμε να εκμεταλλευτούμε την επιπλέον σημασιολογία του RDFS.

Για τον ορισμό μιας κλάσης χρησιμοποιούμε τον πόρο “rdfs:Class”. Στο RDFS, κλάση είναι οποιοσδήποτε πόρος έχει ως τιμή της ιδιότητας “rdf:type” τον πόρο “rdfs:Class”. Έτσι, ο ορισμός της κλάση μηχανοκίνητο όχημα με URI ex:MotorVehicle δίνεται ως εξής:

```
ex:MotorVehicle   rdf:type   rdfs:Class .
```

Εικόνα 2.7 Ορισμός κλάσης στο RDFS

Ο ορισμός του exthings:companyCar ως αντικειμένου της παραπάνω κλάσης γίνεται ως εξής:

```
exthings:companyCar   rdf:type   ex:MotorVehicle .
```

Εικόνα 2.8 Ορισμός αντικειμένου στο RDFS

Έστω ότι αντίστοιχα ορίζουμε τις κλάσεις ex:Van και ex:Truck. Από τη στιγμή που έχουμε ορίσει τις κλάσεις της εφαρμογής μας, μπορούμε να ορίσουμε και σχέσεις υπαγωγής ανάμεσα σε αυτές. Για να ορίσουμε τις δύο νέες κλάσεις ως υποκλάσεις της αρχικής, γράφουμε:

```
ex:Van     rdfs:subClassOf   ex:MotorVehicle .
ex:Truck   rdfs:subClassOf   ex:MotorVehicle .
```

Εικόνα 2.9 Ορισμός υποκλάσεων στο RDFS

Η XML/RDF σύνταξη των παραπάνω ορισμών δίνεται παρακάτω:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">

<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

Εικόνα 2.10 RDF/XML ορισμός κλάσεων και αντικειμένων στο RDFS

Εκτός από την περιγραφή συγκεκριμένων κλάσεων, μπορούμε να περιγράψουμε συγκεκριμένες ιδιότητες που χαρακτηρίζουν αυτές τις κλάσεις. Στο RDFS, οι ιδιότητες περιγράφονται χρησιμοποιώντας την RDF κλάση “rdf:Property” και τις RDFS ιδιότητες “rdfs:domain”. “rdfs:range” και “rdfs:subPropertyOf”. Παρακάτω ορίζουμε μια ιδιότητα, καθώς και το πεδίο ορισμού και το σύνολο τιμών της.

```
ex:driver     rdf:type     rdf:Property .
ex:primaryDriver   rdf:type   rdf:Property .
ex:driver     rdfs:domain  ex:Person .
ex:driver     rdfs:range   ex:MotorVehicle .
```

Εικόνα 2.11 Δήλωση ιδιότητας, πεδίου ορισμού της και συνόλου τιμών της

Η ιδιότητα “rdfs:subPropertyOf” χρησιμοποιείται για να ορίσει ότι μια ιδιότητα είναι υπο-ιδιότητα μιας άλλης:

```
ex:primaryDriver   rdfs:subPropertyOf   ex: driver .
```

Εικόνα 2.12 Δήλωση υπο-ιδιότητας

Αφού η primaryDriver είναι υπο-ιδιότητα της driver, όλα τα ζεύγη που σχετίζονται με την primaryDriver, σχετίζονται και με την driver.

Η XML/RDF σύνταξη των παραπάνω ακολουθεί:

```
<rdf:Property rdf:ID="driver">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="primaryDriver">
  <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>
```

Εικόνα 2.13 RDF/XML ορισμός ιδιοτήτων στο RDFS

Θα μπορούσαμε να πούμε ότι το RDFS δημιουργεί ένα σύστημα τύπων για το RDF. Υπό μία έννοια αυτό το σύστημα τύπων μοιάζει με το σύστημα τύπων αντικειμενοστρεφών γλωσσών προγραμματισμού, όπως είναι η Java. Η ουσιώδης διαφορά των κλάσεων και των ιδιοτήτων του RDF από αυτές των γλωσσών προγραμματισμού είναι ότι οι δηλώσεις στο RDFS δεν αποτελούν μια απαραίτητη υποχρέωση για τα αντικείμενα. Προσθέτουν απλά μια επιπλέον σημασιολογική πληροφορία, η οποία μπορεί να χρησιμοποιηθεί ποικιλοτρόπως.

Η γλώσσα Οντολογιών Διαδικτύου OWL

Η OWL θεωρείται η λεξικογραφική επέκταση του RDF και είναι η επίσημη γλώσσα του W3C που χρησιμοποιείται για τη δημιουργία οντολογιών του Διαδικτύου. Αποτελεί μετεξέλιξη της γλώσσας DAML+OIL. Μια οντολογία στην OWL περιλαμβάνει την περιγραφή κλάσεων, τον ορισμό ιδιοτήτων και στιγμιότυπων της κάθε κλάσης. Παρέχει, όμως, και τη δυνατότητα εξαγωγής συμπερασμάτων, δηλαδή παραγωγής γεγονότων που δεν έχουν σαφώς οριστεί. Η OWL δίνει λύσεις σε πολλά προβλήματα ανεπάρκειας των RDF και RDFS.

Η OWL παρέχει τρεις αυξανόμενης εκφραστικότητας υπο-γλώσσες, καθεμιά από τις οποίες απευθύνεται σε συγκεκριμένη μερίδα χρηστών:

- OWL Lite: Προορίζεται για χρήστες που χρειάζονται πρωτίστως μία ιεραρχία ταξινόμησης και απλούς περιορισμούς. Για παράδειγμα, ενώ υποστηρίζει περιορισμούς πληθικότητας, επιτρέπει τιμές πληθικότητας μόνο 0 και 1. Η δημιουργία και ο χειρισμός μίας οντολογίας με αυτή τη γλώσσα είναι σαφώς πιο εύκολα, σε σχέση με τις άλλες δύο γλώσσες, αφού διαθέτει λιγότερο πολύπλοκο λεξιλόγιο και σύνταξη. Παράλληλα, όμως, προσφέρει μειωμένη εκφραστικότητα σε σχέση με τις άλλες δύο γλώσσες.
- OWL DL: Προορίζεται για χρήστες που χρειάζονται μέγιστη εκφραστικότητα, διατηρώντας παράλληλα υπολογιστική πληρότητα (όλα τα συμπεράσματα είναι υπολογίσιμα) και αποκρισιμότητα (όλοι οι υπολογισμοί τελειώνουν σε πεπερασμένο χρονικό διάστημα). Περιλαμβάνει όλα τα στοιχεία της γλώσσας OWL, τα οποία, όμως, μπορούν να χρησιμοποιηθούν κάτω από ορισμένους περιορισμούς (για παράδειγμα, μία κλάση μπορεί να είναι υποκλάση πολλών κλάσεων, όχι όμως στιγμιότυπο μίας άλλης κλάσης).
- OWL Full: Προορίζεται για χρήστες που χρειάζονται μέγιστη εκφραστικότητα και τη συντακτική ελευθερία της RDF, χωρίς να ενδιαφέρονται για την διατήρηση της υπολογιστικής πληρότητας και αποκρισιμότητας. Για παράδειγμα, μία κλάση μπορεί υπολογίζεται ως συλλογή στιγμιότυπων, αλλά ταυτόχρονα και η ίδια ως στιγμιότυπο. Η OWL Full επιτρέπει σε μία οντολογία να επεκτείνει τη σημασία προκαθορισμένου RDF ή OWL λεξιλογίου. Λόγω της μεγάλης εκφραστικότητας, η γλώσσα αυτή αφήνει λίγα περιθώρια για πλήρη συλλογιστική υποστήριξη.

Για τις τρεις αυτές υπογλώσσες της OWL ισχύουν οι παρακάτω προτάσεις:

- Κάθε νόμιμη OWL Lite οντολογία είναι και νόμιμη OWL DL οντολογία.
- Κάθε νόμιμη OWL DL οντολογία είναι και νόμιμη OWL Full οντολογία.
- Κάθε έγκυρο OWL Lite συμπέρασμα είναι και έγκυρο OWL DL συμπέρασμα.
- Κάθε έγκυρο OWL DL συμπέρασμα είναι και έγκυρο OWL Full συμπέρασμα.

Η επιλογή ανάμεσα στις τρεις εκδοχές της OWL καθορίζεται από τις απαιτήσεις που έχει ο χρήστης από τη γλώσσα. Η επιλογή ανάμεσα στην OWL Lite και στην OWL DL εξαρτάται από το πόσο αναγκαίες είναι οι πιο εκφραστικές δομές που χρησιμοποιεί η OWL DL. Η επιλογή ανάμεσα στην OWL DL και στην OWL Full εξαρτάται από την ανάγκη για χρήση των διευκολύνσεων μετα-μοντελοποίησης του RDFS (για παράδειγμα δημιουργία κλάσεων από κλάσεις, ενσωμάτωση ιδιοτήτων σε κλάσεις). Όταν χρησιμοποιείται η OWL Full

συγκρινόμενη με την OWL DL, η υποστήριξη συλλογιστικής είναι λιγότερο προβλέψιμη, δεδομένου ότι πλήρεις υλοποιήσεις για OWL Full δεν υπάρχουν αυτή τη στιγμή.

Η OWL Full μπορούμε να πούμε ότι είναι μία επέκταση της RDF, ενώ οι OWL Lite και OWL DL επεκτάσεις μίας περιορισμένης όψης της RDF. Κάθε OWL (Lite, DL, Full) έγγραφο είναι και RDF έγγραφο και κάθε RDF έγγραφο είναι και OWL Full έγγραφο, αλλά μόνο ορισμένα RDF έγγραφα μπορούν να είναι και OWL Lite OWL ή DL έγγραφα. Εξαιτίας αυτού, πρέπει να δείξουμε προσοχή όταν θέλουμε να μετατρέψουμε ένα RDF έγγραφο σε OWL.

Θα αναφερθούν κάποιες από τα βασικά στοιχεία της OWL. Τα ακόλουθα χαρακτηριστικά της OWL σχετίζονται με τις κλάσεις και τα στιγμιότυπά τους:

- Class: Οι κλάσεις μπορούν να οργανωθούν σε ιεραρχίες με τη χρήση του “subClassOf”. Κάθε άτομο στην OWL είναι αντικείμενο της κλάσης Thing, που είναι υπερκλάση όλων. Ορίζεται επίσης η κλάση Nothing, ως η κλάση που δεν περιέχει κανένα άτομο και είναι υποκλάση όλων των κλάσεων.
- Individual: τα άτομα είναι στιγμιότυπα των κλάσεων και ιδιότητες μπορούν να χρησιμοποιηθούν για να συνδέσουμε τα άτομα μεταξύ τους.
- intersectionOf: Ορίζει την τομή δύο κλάσεων (δεν επιτρέπεται στην OWL Lite).
- unionOf: Ορίζει την ένωση δύο κλάσεων (δεν επιτρέπεται στην OWL Lite).
- complementOf: Ορίζει τη συμπληρωματική κλάση (δεν επιτρέπεται στην OWL Lite).

Ακολουθούν ορισμένα χαρακτηριστικά της OWL Lite που σχετίζονται με ισότητα και ανισότητα:

- equivalentClass: Δύο κλάσεις μπορεί να δηλωθούν ως ισοδύναμες (έχουν τα ίδια στιγμιότυπα). Η ισοδυναμία χρησιμοποιείται για να δηλώσουμε συνώνυμες κλάσεις.
- equivalentProperty: Δύο ιδιότητες μπορεί να δηλωθούν ως ισοδύναμες (σχετίζουν ένα άτομο με το ίδιο σύνολο ατόμων). Η ισοδυναμία χρησιμοποιείται για να δηλώσουμε συνώνυμες ιδιότητες.
- sameAs: Δύο άτομα μπορεί να δηλωθούν ως ίδια. Έτσι μπορεί να δημιουργηθεί ένας αριθμός ονομάτων που αναφέρονται στο ίδιο άτομο.
- differentFrom: Ένα άτομο μπορεί να δηλωθεί ότι είναι διαφορετικό από όλα τα άλλα άτομα. Αυτό είναι χρήσιμο σε γλώσσες όπως η RDF και η OWL, όπου δε θεωρείται ότι ένα άτομο έχει μόνο ένα όνομα.
- AllDifferent: Δηλώνει ότι κάθε στιγμιότυπο σε μια ομάδα στιγμιότυπων είναι διαφορετικό από τα υπόλοιπα.

Όσον αφορά στις ιδιότητες, στην OWL υπάρχουν δύο ειδών: οι ιδιότητες τύπων δεδομένων (DatatypeProperty) και οι ιδιότητες αντικειμένων (ObjectProperty). Οι ιδιότητες τύπων δεδομένων είναι σχέσεις μεταξύ ατόμων κλάσεων και RDF λεκτικών και XML Schema τύπων δεδομένων, ενώ οι ιδιότητες αντικειμένων είναι σχέσεις μεταξύ ατόμων δύο κλάσεων. Ακολουθούν τα αναγνωριστικά της OWL Lite που σχετίζονται με ιδιότητες:

- inverseOf: Μια ιδιότητα μπορεί να οριστεί ότι είναι αντίστροφη μιας άλλης. Σε αυτήν την περίπτωση αν το στιγμιότυπο X συνδέεται μέσω μίας ιδιότητας με το στιγμιότυπο Y, τότε το Y συνδέεται με το X μέσω της αντίστροφης ιδιότητας.
- TransitiveProperty: Μια ιδιότητα μπορεί να οριστεί ως μεταβατική. Σε αυτήν την περίπτωση αν τα ζεύγη (X, Y) και (Y, Z) είναι στιγμιότυπα μίας μεταβατικής ιδιότητας, τότε και το ζεύγος (X, Z) είναι στιγμιότυπο της ίδιας ιδιότητας. Οι OWL Lite και OWL DL επιβάλλουν τον περιορισμό ότι οι μεταβατικές ιδιότητες (και οι υπεριδιότητες τους) δεν μπορούν να έχουν περιορισμό μέγιστης πληθικότητας 1. Σε διαφορετική περίπτωση δεν θα ίσχυε η αποκρισιμότητα της γλώσσας.
- SymmetricProperty: Μια ιδιότητα μπορεί να δηλωθεί ως συμμετρική. Σε αυτήν την περίπτωση αν το ζεύγος (X, Y) είναι στιγμιότυπο μίας συμμετρικής ιδιότητας, τότε και το ζεύγος (Y, X) είναι στιγμιότυπο της ίδιας ιδιότητας.
- FunctionalProperty: Στην περίπτωση αυτή μια ιδιότητα δεν έχει περισσότερες από μία τιμή για κάθε άτομο. Είναι ένας έμμεσος τρόπος για να ορίσουμε ότι η ελάχιστη πληθυκότητα της ιδιότητας είναι το 0 και η μέγιστη το 1.
- InverseFunctionalProperty: Η αντίστροφη αυτής της ιδιότητας είναι FunctionalProperty.

Η OWL Lite επιτρέπει να τίθενται περιορισμοί όσον αφορά στο πώς οι ιδιότητες μπορούν να χρησιμοποιηθούν από τα άτομα των κλάσεων. Οι περιορισμοί ανήκουν στην κλάση owl:Restriction και με το στοιχείο owl:onProperty καθορίζεται το σε ποια ιδιότητα εισάγεται ο περιορισμός. Ακολουθούν οι περιορισμοί. Οι δύο πρώτοι περιορισμοί οριοθετούν ποιες τιμές μπορούν να χρησιμοποιηθούν από τις ιδιότητες, ενώ οι δύο επόμενοι οριοθετούν πόσες τιμές μπορούν να πάρουν οι ιδιότητες.

- allValuesFrom: Ορίζει ότι το πεδίο τιμών της ιδιότητας θα αποτελείται από άτομα που ανήκουν μόνο σε μια δοσμένη κλάση. Ο περιορισμός αυτός δε δηλώνει ότι υπάρχει σίγουρα μια τιμή για αυτή την ιδιότητα.
- someValuesFrom: Δηλώνει ότι τουλάχιστον μια τιμή της ιδιότητας αυτής ανήκει σε μια δοσμένη κλάση.

- minCardinality: Παίρνει ως ορίσματα μια κλάση κι έναν ακέραιο n και δηλώνει ότι κάθε άτομο της κλάσης θα σχετίζεται με τουλάχιστον n άτομα μέσω της ιδιότητας αυτής. Στη ουσία, απαιτείται η ιδιότητα να έχει τουλάχιστον n τιμές για όλα τα άτομα της κλάσης. Στην OWL Lite οι μοναδικές τιμές του n που επιτρέπονται είναι το 0 και το 1. Το 0 σημαίνει ότι η ιδιότητα αυτή είναι προαιρετική για τα άτομα της κλάσης.
- maxCardinality: Όμοια με τη minCardinality με τη διαφορά ότι καθορίζεται ο μέγιστος και όχι ο ελάχιστος αριθμός τιμών. Επίσης το n είναι ή 0 ή 1.
- cardinality: Με τον περιορισμό αυτό καθορίζεται ο ακριβής αριθμός τιμών που πρέπει να έχουν τα άτομα της κλάσης ως προς αυτή την ιδιότητα. Στην ουσία είναι σαν να συνδυάζουμε maxCardinality n και ταυτόχρονα minCardinality n. Και εδώ ισχύει ο περιορισμός για το n.

Τόσο η OWL DL όσο και η OWL Full έχουν το ίδιο λεξιλόγιο, αν και η OWL DL υπόκειται σε κάποιους επιπλέον περιορισμούς. Συνοπτικά, η OWL DL απαιτεί διαχωρισμό τύπων (μια κλάση δεν μπορεί να είναι και άτομο ή ιδιότητα και αντίστοιχα, μια ιδιότητα δεν μπορεί να είναι άτομο ή κλάση). Ακόμη, η OWL DL απαιτεί οι ιδιότητες να είναι είτε ιδιότητες τύπων δεδομένων είτε ιδιότητες αντικειμένων. Μερικά χαρακτηριστικά της OWL DL και της OWL Full δίνονται παρακάτω:

- one of: Οι κλάσεις μπορούν να περιγραφούν ως απαρίθμηση των στιγμιότυπων από τα οποία αποτελούνται. Τα μέλη της κλάσης είναι ακριβώς τα απαριθμημένα στιγμιότυπα και μόνο αυτά.
- hasValue: Μία ιδιότητα ενδέχεται να απαιτείται να έχει ως τιμή ένα συγκεκριμένο στιγμιότυπο.
- disjointWith: κλάσεις μπορούν να δηλωθούν ότι είναι ξένες μεταξύ τους, δηλαδή ότι δεν έχουν ούτε ένα κοινό στοιχείο. Έτσι, αν κάποιο στιγμιότυπο ανήκει σε μία κλάση, αποκλείεται να ανήκει και σε μία ξένη προς αυτήν κλάση.
- Επίσης ορίζονται οι πληθικοί περιορισμοί με το n να παίρνει οποιαδήποτε τιμή. Υπάρχουν επίσης και τα αναγνωριστικά unionOf, complementOf και intersectionOf που αναλύθηκαν παραπάνω.
- complex classes: Σε πολλές δομές, η OWL Lite περιορίζει τη σύνταξη σε μοναδικά ονόματα κλάσεων. Η OWL Full επεκτείνει αυτόν τον περιορισμό στο να επιτρέπει σύνθετες περιγραφές κλάσεων, όπως απαριθμημένες κλάσεις, περιορισμούς ιδιοτήτων και συνδυασμούς λογικών εκφράσεων. Επίσης, η OWL Full επιτρέπει σε κλάσεις να χρησιμοποιούνται ως στιγμιότυπα.

2.3 Ανάκτηση Πληροφοριών

Το πεδίο της ανάκτησης πληροφοριών (Information Retrieval) [SKS02] έχει αναπτυχθεί παράλληλα με το πεδίο των βάσεων δεδομένων. Στο παραδοσιακό μοντέλο που χρησιμοποιείται στο πεδίο της ανάκτησης πληροφοριών, οι πληροφορίες οργανώνονται σε έγγραφα. Γενικότερα, ανάκτηση πληροφοριών ορίζεται η επιστήμη που ασχολείται με την αναζήτηση εγγράφων, πληροφοριών μέσα στα έγγραφα και μεταδεδομένων για τα έγγραφα, καθώς επίσης και με την αναζήτηση στις σχεσιακές βάσεις και τον Παγκόσμιο Ιστό. Τα δεδομένα που περιέχονται σε έγγραφα είναι χωρίς δομή, χωρίς κάποιο συσχετιζόμενο σχήμα. Η διαδικασία της ανάκτησης πληροφοριών αποτελείται από τον εντοπισμό σχετικών εγγράφων, σε σχέση με αυτό που εισάγει ο χρήστης, όπως λέξεις κλειδιά και παραδείγματα εγγράφων. Οι μηχανές αναζήτησης του διαδικτύου είναι το χαρακτηριστικότερο παράδειγμα εφαρμογών ανάκτησης πληροφοριών.

Όπως παρουσιάζεται αναλυτικά και στο [Sin01], υπάρχουν διάφορα μοντέλα και υλοποιήσεις της ανάκτησης πληροφοριών.

Δε θα γίνει περεταίρω ανάλυση του πεδίου αυτού, καθώς κάτι τέτοιο θα ήταν πέρα από τα πλαίσια της παρούσας εργασίας. Στην παρακάτω ενότητα εστιάζουμε στις μετρικές επίδοσης της ανάκτησης πληροφοριών, οι οποίες χρησιμοποιήθηκαν εκτενώς στην εργασία.

2.3.1 Μετρικές επίδοσης

Πολλές διαφορετικές μετρικές έχουν προταθεί για την αξιολόγηση της επίδοσης των συστημάτων ανάκτησης πληροφοριών. Μερικές από τις πιο ευρέως χρησιμοποιούμενες είναι η ευστοχία (recall), η ακρίβεια (precision) και η F-μετρική (F-measure).

Ευστοχία: μετρά ποιο ποσοστό των δεδομένων που είναι σχετικά με την αίτηση αναζήτησης ανακτήθηκαν επιτυχώς. Ο τύπος που υπολογίζει την ευστοχία είναι ο παρακάτω:

$$ευστοχία = \frac{|\{σχετικά\} \cap \{ανακτηθέντα\}|}{|\{σχετικά\}|}$$

Ακρίβεια: είναι το ποσοστό των δεδομένων που ανακτήθηκαν τα οποία είναι σχετικά με την αίτηση αναζήτησης. Ο τύπος που υπολογίζει την ακρίβεια είναι ο παρακάτω:

$$ακρίβεια = \frac{|\{σχετικά\} \cap \{ανακτηθέντα\}|}{|\{ανακτηθέντα\}|}$$

Η **F-μετρική** είναι ο αρμονικός σταθμισμένος μέσος της ευστοχίας και της ακρίβειας. Για οποιοδήποτε α μη αρνητικό πραγματικό αριθμό, η γενική φόρμουλα που υπολογίζει τη μετρική αυτή δίνεται παρακάτω:

$$F_\alpha = \frac{(1+\alpha) * ευστοχία * ακρίβεια}{\alpha * ακρίβεια + ευστοχία}$$

Είναι εμφανές ότι για τιμές του α μικρότερες της μονάδας δίνουμε μεγαλύτερη βαρύτητα στην ακρίβεια, ενώ για τιμές μεγαλύτερες της μονάδας δίνουμε μεγαλύτερη βαρύτητα στην ευστοχία.

3

Σχετικές Εργασίες

Εξαιτίας του ολοένα αυξανόμενου μεγέθους των διαθέσιμων πληροφοριών στα δίκτυα και της ανάγκης για σημασιολογικό προσδιορισμό τους, γίνεται μια προσπάθεια να συνδυαστούν τα Δίκτυα Ομότιμων Κόμβων με το Σημασιολογικό Ιστό. Το κεφάλαιο αυτό αναφέρεται σε κάποιες προϋπάρχουσες εργασίες που σχετίζονται με την προσπάθεια αυτή ([MMS+06], [HIM+03], [LIK06]). Σε καθεμιά από τις ενότητες που ακολουθούν παρουσιάζεται και μια εργασία. Στο τέλος κάθε ενότητας γίνεται μια σύγκριση της εκάστοτε εργασίας με την παρούσα.

3.1 SRI: Αξιοποίηση της Σημασιολογικής Πληροφορίας για Αποτελεσματική Δρομολόγηση Ερωτήσεων σε ένα PDMS

Μια εξέλιξη των δικτύων ομότιμων κόμβων είναι τα Συστήματα Διαχείρισης Δεδομένων Ομότιμων (Peer Data Management Systems - PDMS), όπου κάθε κόμβος περιέχει κι ένα σχήμα το οποίο αντιπροσωπεύει το πεδίο ενδιαφέροντος του κόμβου, καθώς και τις σημασιολογικές αντιστοιχίσεις του με τους υπόλοιπους κόμβους. Με τον τρόπο αυτό μπορούν να δημιουργηθούν μηχανισμοί οι οποίοι επιτρέπουν την αποτελεσματική άντληση πληροφοριών εκμεταλλευόμενοι τη διαθέσιμη σημασιολογική πληροφορία.

Έτσι, ένα PDMS είναι μια αποκεντρωμένη αρχιτεκτονική για ανταλλαγή πληροφοριών σε μεγάλου μεγέθους δίκτυα. Οι κόμβοι είναι αυτόνομοι ως προς τα δεδομένα που αποθηκεύουν τοπικά και ως προς τη σημασιολογική περιγραφή και την εννοιολογική οργάνωση που προσφέρουν για τα δεδομένα που επιθυμούν να μοιραστούν με τους άλλους κόμβους. Για να θέσουμε μια ερώτηση σε ένα κόμβο ενός PDMS, χρησιμοποιείται το σχήμα του κόμβου αυτού για το σχηματισμό της ερώτησης, αλλά η απάντηση μπορεί να έρθει από οποιοδήποτε άλλο κόμβο που είναι συνδεδεμένος μέσω ενός σημασιολογικού μονοπατιού με τον αρχικό κόμβο. Καθώς η ερώτηση μεταφέρεται από τον ένα κόμβο στον άλλο, μετασχηματίζεται με βάση τις σημασιολογικές αντιστοιχίσεις που υπάρχουν ανάμεσα στα σχήματα των κόμβων. Εντούτοις, εξαιτίας της ετερογένειας των σχημάτων των κόμβων, ο μετασχηματισμός μιας ερώτησης μπορεί να οδηγήσει σε σημασιολογική προσέγγιση κι έτσι τα επιστρεφόμενα δεδομένα μπορεί να μην ταιριάζουν απόλυτα στις προϋποθέσεις της ερώτησης. Για το σκοπό αυτό, μπορεί να προστεθεί μια βαθμολογία στις σημασιολογικές αντιστοιχίσεις, ως ένα μέτρο της σημασιολογικής συμβατότητας μεταξύ δύο σχημάτων. Δεδομένου ότι αυτή η βαθμολογία δείχνει τη σχετικότητα των δεδομένων ενός κόμβου με την ερώτηση που τίθεται, μπορεί να χρησιμοποιηθεί στη φάση της δρομολόγησης της ερώτησης, για την επιλογή του σημασιολογικού μονοπατιού το οποίο ικανοποιεί καλύτερα τις προϋποθέσεις της ερώτησης.

Σε αυτό το σημείο κρίνεται σημαντική η επεξήγηση της σημασίας της δρομολόγησης ερωτημάτων (της διαδικασίας επιλογής των πλέον υποσχόμενων κόμβων) για τα κατανεμημένα συστήματα. Κάτω από ένα PDMS υπάρχει ένα τεράστιο δίκτυο το οποίο πρέπει να διαχειριστεί τεράστιες ποσότητες δεδομένων. Κάθε σχετικός κόμβος προσθέτει νέες απαντήσεις σε ένα ερώτημα και διαφορετικά μονοπάτια προς τον ίδιο κόμβο μπορεί να οδηγήσουν σε διαφορετικές απαντήσεις. Έτσι, στα πλαίσια ενός σημασιολογικού δικτύου, ένα ερώτημα που τίθεται σε ένα κόμβο, θα πρέπει να προωθηθεί αρχικά σε εκείνους από τους άμεσους γείτονές του οι οποίοι προσφέρουν τα πιο σημασιολογικώς σχετικά αποτελέσματα και στη συνέχεια στους επόμενους κατάλληλους γείτονες αυτών κ.ο.κ.

Όταν ένα ερώτημα προωθείται σε ένα σημασιολογικό μονοπάτι, υφίσταται διαδοχικούς μετασχηματισμούς, κάτι το οποίο μπορεί να οδηγήσει σε μια σειρά σημασιολογικών προσεγγίσεων. Για το σκοπό αυτό προτείνεται να χρησιμοποιηθούν αυτές οι προσεγγίσεις ώστε να επιλεγεί η κατεύθυνση που είναι πιθανότερο να δώσει τα καλύτερα αποτελέσματα για ένα ερώτημα. Σε ορισμένες περιπτώσεις, όμως, οι σημασιολογικές αντιστοιχίσεις δεν είναι αρκετές για να αποφανθούμε για το βέλτιστο μονοπάτι. Ένα τέτοιο παράδειγμα είναι το να έχουν όλοι οι γειτονικοί κόμβοι την ίδια σημασιολογική βαθμολογία (mirrors), ώστε να μην μπορούμε να επιλέξουμε τη βέλτιστη κατεύθυνση. Έτσι, χρειαζόμαστε ένα είδος πληροφορίας για τη σχετικότητα όλου του σημασιολογικού μονοπατιού. Για το σκοπό αυτό προτείνεται ένας κατανεμημένος μηχανισμός ευρετηρίου, ο οποίος ονομάστηκε SRI. Το SRI

περιγράφει για κάθε έννοια του σχήματος ενός κόμβου, τις ικανότητες σημασιολογικής προσέγγισης κάθε υποδικτύου το οποίο είναι προσπελάσιμο από τους άμεσους γείτονες του κόμβου, κι έτσι δίνει μια ιδέα για τη σχετικότητα των δεδομένων που μπορούν να βρεθούν σε κάθε μονοπάτι. Η σημασιολογική γνώση που περιέχεται σε κάθε SRI συνοψίζεται στις διαθέσιμες κατευθύνσεις ώστε να διατηρείται το μέγεθος του σημασιολογικού ευρετηρίου ανάλογο με τον αριθμό των γειτόνων κάθε κόμβου.

Στην εργασία αυτή, μελετούνται τα SRIs για τη σημασιολογική δρομολόγηση ερωτημάτων και αξιολογείται η αποτελεσματικότητά τους. Πιο συγκεκριμένα:

- Η έννοια της σημασιολογικής αντιστοίχισης και των σημασιολογικών μονοπατιών επεκτείνεται με μια βαθμολογία, η οποία εκφράζει το βαθμό της αβεβαιότητας που προκύπτει ανάμεσα σε δύο σημασιολογικές έννοιες. Στα πλαίσια αυτά, έγινε χρήση της ασαφούς θεωρίας.
- Με βάση την προσανατολισμένη στην ασάφεια όψη των αντιστοιχίσεων, εισάγονται τα SRIs ως κατανεμημένα ευρετήρια για δρομολόγηση ερωτημάτων και δίνονται οι απαραίτητες λειτουργίες για τη δημιουργία και συντήρησή τους.
- Δίνεται μια αξιολόγηση της αποτελεσματικότητας των SRIs μέσω μιας σειράς πειραμάτων και τέλος δίνονται ιδέες για μελλοντικές επεκτάσεις.

Σημασιολογικές Αντιστοιχίσεις

Οι κόμβοι σε ένα σημασιολογικό δίκτυο είναι συνδεδεμένοι μεταξύ τους με χρήση σημασιολογικών αντιστοιχίσεων ανάμεσα στα σχήματά τους. Στα πλαίσια της εργασίας θεωρήθηκε το απλοποιημένο σενάριο όπου οι σημασιολογικές αντιστοιχίσεις είναι κατευθυνόμενες και ένα-προς-ένα.

Μια σημασιολογική αντιστοίχιση μπορεί να εδραιωθεί από ένα σχήμα-πηγή S_j προς ένα σχήμα-στόχο S_i και ορίζει πώς αναπαρίσταται το S_i με όρους του λεξιλογίου του S_j. Πιο συγκεκριμένα, αντιστοιχίζει κάθε έννοια του S_i σε μια αντίστοιχη έννοια στο S_j, δίνοντας ένα βαθμό της σημασιολογικής ομοιότητας των δύο εννοιών. Οποτεδήποτε ένας κόμβος εισέρχεται σε ένα PDMS, διαλέγει ένα μικρό αριθμό από γείτονες και υπολογίζει και αποθηκεύει τις κατάλληλες σημασιολογικές αντιστοιχίσεις.

Σημασιολογικά Μονοπάτια

Ένα σημασιολογικό μονοπάτι είναι μια αλυσίδα από σημασιολογικές αντιστοιχίσεις που ενώνουν ένα ζευγάρι κόμβων. Με το μετασχηματισμό ενός ερωτήματος μέσω των αντιστοιχίσεων που συνθέτουν ένα σημασιολογικό μονοπάτι, το PDMS μπορεί να προσπελάσει δεδομένα σε απομακρυσμένους κόμβους. Οι σημασιολογικές προσεγγίσεις που

προκύπτουν σε ένα σημασιολογικό μονοπάτι μπορούν να υπολογιστούν από τη σύνθεση των ασαφών σχέσεων των επιμέρους αντιστοιχίσεων.

Η συνάρτηση σύνθεσης πρέπει να λαμβάνει υπόψη της ότι όσο μεγαλώνει η αλυσίδα των αντιστοιχίσεων τόσο συσσωρεύονται σημασιολογικές προσεγγίσεις και η βαθμολογία του μονοπατιού μειώνεται. Μια από τις πολλές εναλλακτικές που έχουν προταθεί είναι το αλγεβρικό γινόμενο.

Γενικευμένες Σημασιολογικές Αντιστοιχίσεις

Η επεξεργασία ενός ερωτήματος ξεκινά στον κόμβο p_i στον οποίο τέθηκε το ερώτημα. Ο p_i μετασχηματίζει το ερώτημα με βάση τον επόμενο γειτονικό κόμβο p_j και το ερώτημα μεταφέρεται στο υποδίκτυο με ρίζα τον p_j. Για να μοντελοποιηθεί η σημασιολογική προσέγγιση του υποδικτύου του p_j όσον αφορά στο σχήμα του p_i, οι σημασιολογικές προσεγγίσεις που δίνονται από κάθε μονοπάτι του υποδικτύου του p_j συσσωρεύονται σε ένα μέγεθος που δείχνει τη συνολική σχετικότητα του υποδικτύου και αποτελεί τις γενικευμένες σημασιολογικές αντιστοιχίσεις.

Η συνάρτηση g που συνθέτει τις σημασιολογικές βαθμολογίες πρέπει να επιλεγεί σωστά ώστε κάθε βαθμολογία που προκύπτει για μια έννοια να είναι αντιπροσωπευτική της σημασιολογικής προσέγγισης που δίνεται από ένα κόμβο και το υποδίκτυό του. Υπάρχουν πολλές εναλλακτικές για την g, όπως οι συναρτήσεις min και max, κάποιος γενικευμένος μέσος ή κάποια συνάρτηση μέσου όρου με βάρη.

Ευρετήρια Σημασιολογικής Δρομολόγησης (SRI)

Οι βαθμοί συμμετοχής που δίνονται από τις γενικευμένες σημασιολογικές αντιστοιχίσεις μπορούν να χρησιμοποιηθούν για τη σωστή προώθηση του μετασχηματισμένου στον κόμβο p ερωτήματος προς την πιο υποσχόμενη σημασιολογικά κατεύθυνση στο δίκτυο. Για το σκοπό αυτό, κάθε κόμβος p διατηρεί ένα πίνακα ο οποίος περιέχει τους βαθμούς συμμετοχής ανάμεσα στο σχήμα του κόμβου και τα σχήματα των γειτόνων του. Ο πίνακας αυτός χρησιμοποιείται ως πίνακας δρομολόγησης και ονομάζεται Ευρετήριο Σημασιολογικής Δρομολόγησης (Semantic Routing Index - SRI). Αν ο p έχει n γείτονες, το SRI περιέχει n+1 εγγραφές, όπου η πρώτη αναφέρεται στη γνώση που υπάρχει στο σχήμα του ίδιου του p.

Ο χώρος που απαιτείται για την αποθήκευση του SRI είναι ανάλογος του αριθμού των γειτόνων ενός κόμβου και άρα αρκετά μικρός για να μπορούν τα SRIs να χρησιμοποιηθούν σε δίκτυα με μεγάλο αριθμό κόμβων.

Όταν ένας κόμβος p θέλει να προωθήσει ένα ερώτημα q, ανατρέχει στο SRI του για να βρει το γειτονικό κόμβο που είναι περισσότερο συσχετισμένος σημασιολογικά με αυτόν (που έχει τη μεγαλύτερη βαθμολογία για τις έννοιες που μας ενδιαφέρουν).

Διάδοση Σημασιολογικής Πληροφορίας στο Δίκτυο

Δεδομένου ότι τα SRIs συνοψίζουν τη σημασιολογική πληροφορία που υπάρχει στο δίκτυο, όταν το δίκτυο αλλάζει, αλλάζουν και τα SRIs. Αυτό μπορεί να συμβεί είτε κατά την ένταξη/αποσύνδεση κόμβων στο δίκτυο είτε από αλλαγές στα σχήματα των κόμβων

Η εξέλιξη των SRIs διευθετείται με έναν αυξητικό τρόπο. Αρχικά θεωρούμε έναν απομονωμένο κόμβο, του οποίου το SRI περιέχει μόνο μία σειρά [1,...,1]. Όταν ένας κόμβος συνδέεται με έναν άλλο, κάθε ένας εμπλουτίζει το SRI του με γραμμές σύμφωνα με μια αθροιστική συνάρτηση g.

Οι αποσυνδέσεις από το δίκτυο αντιμετωπίζονται με παρόμοιο τρόπο: οι γείτονες του κόμβου αυτού διαγράφουν από το SRI τους τη σειρά που αναφέρεται στον κόμβο και στη συνέχεια προωθούν την αλλαγή στους υπόλοιπους κόμβους ακολουθώντας τον παραπάνω τρόπο. Μια παρόμοια διαδικασία ακολουθείται όταν συμβεί μια αλλαγή στη σημασιολογική πληροφορία που διατηρείται σε ένα κόμβο. Όταν συμβούν πολλές αλλαγές υπάρχει η πρόκληση να ακολουθηθεί κατάλληλη στρατηγική ώστε να ελαχιστοποιηθεί η κίνηση που προκαλείται στο δίκτυο.

Το SRI στην πράξη - Πειράματα

Ορισμένα πειράματα πραγματοποιήθηκαν για να διαπιστωθεί η αποτελεσματικότητα των SRIs στη δρομολόγηση ερωτημάτων. Διαλέχτηκαν κόμβοι που να ανήκουν σε διαφορετικές σημασιολογικές κατηγορίες και δημιουργήθηκαν αντιστοιχίσεις ανάμεσα στα σχήματα που ανήκουν στις ίδιες σημασιολογικές κατηγορίες. Όσον αφορά στην τοπολογία αρχικά δοκιμάστηκαν απλές τοπολογίες δέντρου και στη συνέχεια πιο πολύπλοκες.

Στα πλαίσια των πειραμάτων τέθηκαν ερωτήσεις σε διάφορους κόμβους. Κατά την επεξεργασία τα ερωτήματα προωθούνται σε άλλους κόμβους μέχρι να ικανοποιηθεί μια συνθήκη τερματισμού. Υπάρχουν δύο εναλλακτικές: η πρώτη είναι η επεξεργασία των ερωτημάτων μέχρι να φτάσουμε ένα δεδομένο αριθμό κόμβων (μηνυμάτων) στους οποίους θα τεθεί το ερώτημα και η μέτρηση της ποιότητας των αποτελεσμάτων, και η δεύτερη είναι να δοθεί ένας βαθμός ικανοποίησης (η ικανοποίηση είναι ανάλογη της ποιότητας των αποτελεσμάτων) και να μετρηθεί ο αριθμός των μηνυμάτων που απαιτούνται για να φτάσουμε στο βαθμό αυτό. Η στρατηγική αναζήτησης που χρησιμοποιείται είναι η DFS (αναζήτηση κατά βάθος). Στα πλαίσια των πειραμάτων συγκρίθηκε ο μηχανισμός επιλογής

των γειτονικών κόμβων με χρήση των SRIs με το μηχανισμό όπου η επιλογή του επόμενου γείτονα βασίζεται στις σημασιολογικές αντιστοιχίσεις (SemMap) και με την τυχαία επιλογή του επόμενου γείτονα (Random).

Από τα αποτελέσματα των πειραμάτων διαπιστώθηκε ότι ο μηχανισμός SemMap είναι πολύ καλύτερος από τον Random, αλλά ο SRI απαιτεί ακόμη λιγότερα μηνύματα.

Στη συνέχεια μελετήθηκε η συμπεριφορά των στρατηγικών δρομολόγησης καθώς οι συνθήκες τερματισμού γίνονται πιο αυστηρές. Και πάλι ο Random ήταν με διαφορά ο χειρότερος. Η διαφορά μεταξύ των SemMap και SRI δεν είναι ιδιαίτερα μεγάλη όταν οι συνθήκες τερματισμού είναι σχετικά χαλαρές, αλλά αυξάνεται σημαντικά υπέρ του SRI όταν γίνονται αυστηρότερες.

Ακολούθως, έγιναν πειράματα για να διαπιστωθεί το ρόλο που παίζει ο ορίζοντας ανανέωσης (καθορίζοντας τον ορίζοντα ανανέωσης περιορίζουμε το τμήμα του δικτύου που συνοψίζει κάθε βαθμολογία στο SRI. Καθώς αυξάνεται ο ορίζοντας ανανέωσης, αυξάνεται και η ποιότητα των αποτελεσμάτων. Βέβαια, μετά από ένα σημείο η βελτίωση των αποτελεσμάτων είναι αμελητέα. Η τελευταία παρατήρηση είναι σημαντική για τον περιορισμό του κόστους συντήρησης των SRIs.

Τέλος εφαρμόστηκε ένας μηχανισμός κλαδέματος μονοπατιών (με βάση ένα κατώφλι) που είναι σχεδόν απίθανο να οδηγήσουν σε χρήσιμα αποτελέσματα κατά τη διαδικασία της δρομολόγησης ερωτημάτων. Ο μηχανισμός αυτός οδηγεί σε σημαντική μείωση των μηνυμάτων που απαιτούνται.

Σύγκριση με παρούσα εργασία

Όσον αφορά στην τοπολογία και την οργάνωση του δικτύου υπάρχουν αρκετές ομοιότητες με την παρούσα εργασία. Και οι δύο εργασίες αναφέρονται σε αδόμητα P2P – PDMS δίκτυα. Παράλληλα, τα δεδομένα είναι δομημένα και κάθε κόμβος έχει το τοπικό του σχήμα. Η διαφορά, βέβαια, εδώ είναι ότι στο σύστημά μας θεωρούμε την ύπαρξη μιας καθολικής οντολογίας, ενώ το SRI θεωρεί τοπικές σημασιολογικές αντιστοιχίσεις. Επίσης, και οι δύο εργασίες καταπιάνονται με την αποδοτική απάντηση ερωτήσεων. Η ουσιώδης διαφορά είναι ότι μελετούνται διαφορετικές πτυχές του ζητήματος. Στη δικιά μας εργασία, δίνεται ιδιαίτερη έμφαση στον αυτόματο υπολογισμό του βαθμού ομοιότητας μεταξύ των κόμβων. Κάθε κόμβος επικοινωνεί με τους γείτονές του και με βάση τα σχήματα των βάσεων, τις σημασιολογικές αντιστοιχίσεις, αλλά και τους περιορισμούς που θέτει κάθε ερώτημα, υπολογίζουν την καταλληλότητα του κάθε κόμβου να απαντήσει στο ερώτημα που έχει τεθεί. Ο υπολογισμός γίνεται κάθε φορά που υπάρχει ανάγκη κι έτσι δεν απαιτείται η διατήρηση κάποιας δομής σε κάθε κόμβο που να δείχνει το βαθμό ομοιότητάς του με τους γειτονικούς. Αντίθετα, στο SRI δεν προτείνεται κάποιος τρόπος υπολογισμό του βαθμού ομοιότητας

μεταξύ των σχέσεων των κόμβων (αντίθετα, θεωρούνται κάποιες δεδομένες τιμές εκ των προτέρων), ενώ οι βαθμοί αυτοί αποθηκεύονται τοπικά σε κάθε κόμβο (η τοπική αποθήκευση διατηρεί το αποκεντρωμένο πρότυπο που πρεσβεύουν τα δίκτυα ομότιμων κόμβων). Μεγαλύτερη έμφαση δίνεται σε θέματα δρομολόγησης των ερωτημάτων. Στα πλαίσια αυτά μελετούνται διάφοροι τρόποι για αποδοτικότερη δρομολόγηση των ερωτημάτων στο δίκτυο με τη χρήση των ευρετηρίων SRI.

3.2 Piazza: Υποδομή Διαχείρισης Δεδομένων για Εφαρμογές Σημασιολογικού Ιστού

Η ανάπτυξη οντολογιών και γλωσσών αναπαράστασης, στα πλαίσια του Σημασιολογικού Ιστού, δημιουργούν δύο σημαντικά προβλήματα. Το πρώτο πρόβλημα είναι το χάσμα που υπάρχει ανάμεσα στον κόσμο του RDF και τα περισσότερα σημερινά δεδομένα και εφαρμογές. Το RDF επικεντρώνεται στη δομή του πεδίου, αναπαριστώντας τα πάντα ως ένα σύνολο από κλάσεις και ιδιότητες και δημιουργώντας ένα γράφο συσχετίσεων. Αντίθετα, οι σημερινές πηγές δεδομένων που χρησιμοποιούν την XML συχνά εμφωλιάζουν πληροφορίες για τις συσχετιζόμενες οντότητες κατευθείαν στην περιγραφή των πιο σημαντικών αντικειμένων κι έτσι αφήνουν τον τύπο της σχέσης απροσδιόριστο. Είναι προφανώς επιθυμητό ο Σημασιολογικός Ιστός να προσφέρει διαλειτουργικότητα με τις ήδη υπάρχουσες πηγές και καταναλωτές. Για να δημιουργήσουμε εφαρμογές Σημασιολογικού Ιστού θα πρέπει όχι μόνο να μπορούμε να αντιστοιχίσουμε τις δομές πεδίου δύο πηγών, αλλά και τις δομές εγγράφου.

Το δεύτερο πρόβλημα σχετίζεται με τον αριθμό των οντολογιών που θα υπάρχουν στο Σημασιολογικό Ιστό και των σχημάτων που θα είναι υπεύθυνα για τη διαμεσολάβηση ανάμεσα στις οντολογίες. Πιστεύεται ότι δε θα υπάρξει μόνο μια οντολογία για κάθε τομέα, αλλά ότι θα υπάρχουν αρκετές, πιθανώς επικαλυπτόμενες, οντολογίες. Έτσι, για να αναπτυχθεί ένα τόσο μεγάλο σύστημα όπως ο Σημασιολογικός Ιστός, θα χρειαστούμε μια αρχιτεκτονική που να επιτρέπει τη δημιουργία ενός ιστού δεδομένων στον οποίο να μπορούν να εισάγονται συνεχώς νέες πηγές. Κάθε νέα πηγή θα αντιστοιχίζεται με όσες άλλες πηγές θεωρεί σημαντικές. Επιπλέον, θα χρειαστούμε αποδοτικούς αλγόριθμους που να επιτρέπουν να αντλούμε δεδομένα από μακρινούς αλλά σχετικούς κόμβους.

Η εργασία αυτή περιγράφει ένα σύστημα που παρέχει μια υποδομή για τη δημιουργία εφαρμογών Σημασιολογικού Ιστού και το οποίο αντιμετωπίζει τα προαναφερθέντα προβλήματα: το σύστημα Piazza. Μια εφαρμογή Piazza αποτελείται από πολλούς κόμβους, καθένας από τους οποίους παρέχει είτε δεδομένα είτε ένα σχήμα (ή μια οντολογία) είτε και τα δύο μαζί. Το σημασιολογικό κομμάτι του Piazza οφείλεται στις τοπικές αντιστοιχίσεις ανάμεσα σε μικρά σύνολα κόμβων. Η αρχιτεκτονική του Piazza υποστηρίζει τόσο τις τοπικές αντιστοιχίσεις ανάμεσα στις πηγές δεδομένων όσο και τη συνεργασία μεταξύ ενδιάμεσων οντολογιών. Στην ουσία πρόκειται για ένα PDMS. Η προσφορά της εργασίας συνοψίζεται στα παρακάτω σημεία:

- Προτείνεται μια γλώσσα για τη διαμεσολάβηση ανάμεσα στους κόμβους που επιτρέπει την αποτύπωση απλών δομών πεδίου, αλλά και σύνθετων δομών εγγράφου. Βασίζεται στην XQuery (γλώσσα επερώτησης για την XML). Δείχνεται επίσης, ότι η γλώσσα αυτή μπορεί να αντιστοιχίσει κόμβους με RDF δεδομένα σε κόμβους με XML δεδομένα.
- Περιγράφεται ένας αλγόριθμος για να απαντούνται ερωτήματα στο Piazza που συνδέουν σημασιολογικές αντιστοιχίσεις που έχουν προσδιοριστεί χρησιμοποιώντας την παραπάνω γλώσσα. Ιδιαίτερη σημασία έχει δοθεί στην υποστήριξη πηγών με XML δεδομένα, καθώς και στο γεγονός ότι οι αντιστοιχίσεις είναι κατευθυνόμενες.
- Περιγράφεται ένα υλοποιημένο σενάριο χρησιμοποιώντας το Piazza και δίνονται ορισμένες παρατηρήσεις από αυτή την εμπειρία.

Δεδομένα, Σχήματα και Ερωτήματα

Όπως είναι γνωστό, τα δεδομένα σε μορφή XML είναι πολύ διαδεδομένα σήμερα. Στην ιδανική περίπτωση, κάποιος θα μπορούσε να αντιστοιχίσει τα XML-δεδομένα στο Σημασιολογικό Ιστό και να θέσει ερωτήσεις πάνω σε αυτά χρησιμοποιώντας εργαλεία του Σημασιολογικού Ιστού, και ομοίως, να αντιστοιχίσει τα αποτελέσματα των ερωτήσεων πίσω σε μορφή XML, ώστε να μπορούν να χρησιμοποιηθούν από συμβατικές εφαρμογές.

Στο σημείο αυτό θα γίνει μια σύγκριση ανάμεσα στο RDF και την XML. Το RDF αναπαριστά ένα γράφο από διασυνδεδεμένα αντικείμενα, ιδιότητες και τιμές. Επιπλέον, το RDF δίνει μοναδικές σημασιολογικές ερμηνείες σε ορισμένα δεσμευμένα αντικείμενα και ιδιότητες και ονοματίζει άμεσα τις σχέσεις ανάμεσα σε αντικείμενα. Από την άλλη, η XML (δεδομένου ότι δε συνοδεύεται από κάποιο σχήμα), δε δίνει σημασιολογική ερμηνεία σε καμία ιδιότητα και χρησιμοποιεί την ιεραρχία ώστε να κωδικοποιεί έμμεσα τις λογικές σχέσεις. Η XML κωδικοποιεί τα δεδομένα με ένα τρόπο παρόμοιο με τις σχεσιακές βάσεις δεδομένων. Επιπλέον, το RDF και η XML χρησιμοποιούν διαφορετικούς φορμαλισμούς για

να εκφράσουν τα σχήματα. Η XML χρησιμοποιεί την XML Schema, η οποία βασίζεται στις αντικειμενοστρεφείς κλάσεις και τις βάσεις δεδομένων σχημάτων, ενώ το RDF χρησιμοποιεί γλώσσες όπως RDFS, DAML+OIL και OWL που προέρχονται από το χώρο της Αναπαράστασης Γνώσης (Knowledge Representation - KR) με χρήση οντολογιών. Είναι σημαντικό εδώ να σημειωθεί ότι ένα μέρος της λειτουργικότητας των KR περιγραφών και ορισμών εννοιών, μπορούν να αποτυπωθούν στην XML με χρήση προβολών (views).

Για τη μετατροπή ανάμεσα στην RDF και την XML, απαιτούνται διάφοροι τύποι σημασιολογικών αντιστοιχίσεων: ένα-προς-ένα αντιστοιχίσεις ανάμεσα στις έννοιες (που απαιτούν μια απλή μετονομασία), πιο σύνθετες ν-προς-μ αντιστοιχίσεις (οι οποίες μπορεί να απαιτούν λειτουργίες τύπου συνδέσμου), σύνθετες μετατροπές ορισμών εννοιών και τέλος ορισμένοι ορισμοί εννοιών που μπορεί να απαιτήσουν σημαντικές δυνατότητες συμπερασμού. Οι αντιστοιχίσεις που προτάθηκαν βασίζονται στο XQuery, αφού το τελευταίο προσφέρει σημαντικές δυνατότητες μετασχηματισμού, συνδέσμων και μετονομασίας.

Ανταλλαγή Δεδομένων και Διαμεσολάβηση, Επεξεργασία Ερωτημάτων

Τα ερωτήματα στο Piazza θέτονται στο πλαίσιο του σχήματος ενός συγκεκριμένου κόμβου, που αντιστοιχεί στην ορολογία που προτιμά ο χρήστης. Για την απάντηση στο ερώτημα χρησιμοποιείται όλη η σημασιολογικά σχετική πληροφορία του δικτύου. Για να χρησιμοποιήσουμε, όμως, δεδομένα άλλων κόμβων, οι διάφοροι κόμβοι θα πρέπει να έχουν μεταξύ τους σημασιολογικές αντιστοιχίσεις, οι οποίες μπορεί να είναι δύο τύπων: αντιστοιχίσεις μεσολάβησης, στις οποίες οι πηγές δεδομένων σχετίζονται μέσω ενός ενδιάμεσου σχήματος ή οντολογίας, και οι σημείου-προς-σημείο αντιστοιχίσεις, που περιγράφουν πώς μπορούν να μετασχηματιστούν τα δεδομένα ώστε να συμμορφώνονται με το σχήμα ενός άλλου κόμβου.

Ο φορμαλισμός για τη δημιουργία αντιστοιχίσεων εξαρτάται από το είδος των κόμβων που αντιστοιχούμε. Μπορεί να έχουμε είτε ζευγάρια OWL/RDF κόμβων είτε ζευγάρια XML/XML Schema κόμβων είτε αντιστοιχίσεις XML-σε-RDF.

Δοθέντος ενός συνόλου κόμβων, των σημασιολογικών αντιστοιχίσεων μεταξύ τους και ενός ερωτήματος, το βασικό πρόβλημα που προκύπτει είναι η επεξεργασία του ερωτήματος. Αυτό που μας απασχολεί στην εργασία αυτή είναι το πώς θα αποκτήσουμε σημασιολογικά σωστές απαντήσεις, καθώς και η αποδοτική επεξεργασία του ερωτήματος.

Αντιστοιχίσεις στο Piazza, Γλώσσα Αντιστοίχισης

Στην εργασία περιγράφεται η γλώσσα που χρησιμοποιείται για να περιγράψουμε τις αντιστοιχίσεις ανάμεσα στους κόμβους του δικτύου Piazza. Επικεντρωνόμαστε σε κόμβους

που έχουν δεδομένα σε μορφή XML και το σχήμα κάθε κόμβου περιγράφεται σε XML Schema. Υπάρχει σαφής διαχωρισμός ανάμεσα στο πεδίο των δεδομένων που περιγράφεται από το σχήμα και στο στιγμιότυπο των δεδομένων που αποθηκεύονται. Έτσι, οι αντιστοιχίσεις παίζουν δύο ρόλους: την περιγραφή αποθήκευσης που καθορίζει τα δεδομένα που αποθηκεύονται στον κόμβο και την αντιστοίχιση σχημάτων που περιγράφει πώς η ορολογία και η δομή ενός κόμβου αντιστοιχίζεται με τα αντίστοιχα χαρακτηριστικά ενός άλλου.

Ο σκοπός του Piazza είναι να χρησιμοποιεί τις αντιστοιχίσεις για να μετασχηματίζει ερωτήματα κι έτσι να τα απαντά. Μια αντιστοίχιση ανάμεσα σε δύο κόμβους μπορεί να είναι είτε συμπερίληψης (inclusion) είτε ισότητας. Το όραμα του Piazza είναι η γενίκευση της ενσωμάτωσης δεδομένων. Επιχειρείται να μεταφερθούν οι τεχνικές LAV (local-as-view) και GAV (global-as-view) της ενσωμάτωσης δεδομένων στον κόσμο της XML, όπως επίσης και οι σειριοποιήσεις του RDF. Μια αντιστοίχιση καθορίζεται υπό το πρίσμα του στόχου και άρα το σχήμα του στόχου καθορίζεται με ένα τρόπο που μοιάζει με το GAV. Από την άλλη, υποστηρίζεται η κατάσταση όπου το σχήμα της πηγής είναι μια προβολή του στόχου και επιπλέον υποστηρίζονται τα properties της πηγής (αναφέρθηκαν παραπάνω). Τα δύο τελευταία είναι χαρακτηριστικά του LAV.

Η γλώσσα αντιστοίχισης δανείζεται στοιχεία από το XQuery, αλλά είναι πιο εύχρηστη στο συμπερασμό και μπορεί να εκφραστεί σε τμηματική μορφή. Οι αντιστοιχίσεις στη γλώσσα αυτή είναι ένας ή περισσότεροι ορισμοί αντιστοιχίσεων και είναι κατευθυνόμενοι από την πηγή στο στόχο. Κάθε ορισμός αντιστοίχισης ξεκινά με ένα XML “καλούπι” (template), που αντιστοιχεί ορισμένα μονοπάτια ή υποδέντρα ενός στιγμιότυπου του σχήματος του στόχου.

Αλγόριθμος Απάντησης Ερωτημάτων

Δοθέντος ενός δικτύου Piazza από κόμβους με XML δεδομένα, ενός συνόλου σημασιολογικών αντιστοιχίσεων ανάμεσα στους κόμβους και ενός ερωτήματος στο σχήμα ενός από τους κόμβους, δίνεται ένας αλγόριθμος που παράγει όλες τις βέβαιες απαντήσεις που παρέχει το σύστημα.

Δοθέντος ενός ερωτήματος Q σε ένα κόμβο P, χρησιμοποιούμε τις περιγραφές αποθήκευσης του P για να μετασχηματίσουμε το Q στο Q’ πάνω στα δεδομένα που είναι αποθηκευμένα στον P, και στη συνέχεια βρίσκουμε όλους τους σημασιολογικούς γείτονες του P και μετασχηματίζουμε το Q σε Q’’ ώστε να μπορεί να απαντηθεί από τους γείτονες. Η τεχνική αυτή εφαρμόζεται επαναληπτικά και τελικά συνοψίζουμε όλα τα αποτελέσματα.

Προτείνεται μία γραφική αναπαράσταση των ερωτημάτων και των αντιστοιχίσεων. Κάθε ερώτημα χωρίζεται σε μπλοκ και σε κάθε μπλοκ αντιστοιχίζονται κάποιες δομές που είναι απαραίτητες για την εκτέλεση του αλγορίθμου.

Παρουσιάζεται αναλυτικά ο αλγόριθμος επανεγγραφής, τα δύο βασικά βήματα του οποίου είναι η αντιστοίχιση προτύπων και η διαχείριση των επιστρεφόμενων μεταβλητών και κατηγορημάτων.

Το πρώτο βήμα λαμβάνει υπόψη τα δεντρικά πρότυπα στο ερώτημα Q και βρίσκει τα αντίστοιχα πρότυπα στο σχήμα του στόχου.Να σημειωθεί ότι ένα πρότυπο t' στο στόχο μπορεί να επιβάλλει επιπλέον συνθήκες στο αρχικό πρότυπο t του ερωτήματος και ότι μπορεί να υπάρχουν περισσότερα του ενός πρότυπα στο στόχο που να αντιστοιχούν σε ένα πρότυπο του ερωτήματος. Τέλος, αν δε βρεθούν αντιστοιχίσεις, το αποτέλεσμα της επανεγγραφής θα είναι κενό

Στο δεύτερο βήμα ο αλγόριθμος εξασφαλίζει ότι οι μεταβλητές που απαιτούνται από το ερώτημα πράγματι μπορούν να επιστραφούν και ότι έχουν εφαρμοστεί όλα τα κατηγορήματα του ερωτήματος. Η παρουσία της XML έναντι του σχεσιακού μοντέλου προσθέτει κάποιες επιπλέον λεπτομέρειες. Ένα πιθανό πρόβλημα δημιουργείται όταν το πρώτο βήμα δεν επιστρέφει ακριβώς τις μεταβλητές που ζητά το ερώτημα. Σημειώνεται ότι υπάρχουν περιπτώσεις που μια αντιστοίχιση είναι χρήσιμη ακόμη κι αν δεν επιστρέφει όλες τις τιμές εξόδου ή δεν ικανοποιεί όλα τα κατηγορήματα. Επιπλέον, ορισμένες φορές μπορούμε να δούμε ότι μια αντιστοίχιση δεν παράγει κανένα αποτέλεσμα που να ικανοποιεί το ερώτημα, κάνοντας χρήση ορισμένων ιδιοτήτων.

Μια Εφαρμογή Piazza

Για να ελεγχθεί το σύστημα Piazza δημιουργήθηκε μια εφαρμογή, η DB Research, που αποτελείται από ένα δίκτυο 15 κόμβων (που προσφέρουν είτε δεδομένα είτε σχήματα ή και τα δύο) σε συνδυασμό με μια μηχανή επανεγγραφής και μια μηχανή αποτίμησης ερωτημάτων, και τέθηκαν διάφορα ερωτήματα στους κόμβους.

Ο χρόνος μετασχηματισμού αποδείχτηκε ικανοποιητικά μικρός. Παράλληλα, το Piazza προσφέρει τοπικότητα στη διαχείριση των εννοιών (όχι καθολικές οντολογίες) , ενώ υπάρχει δυνατότητα επιλογής ανάμεσα στο αν τα ερωτήματα θα αποτιμώνται με αυστηρό τρόπο ή αν θα απαντούνται με την αρχή της βέλτιστης προσπάθειας (best effort).

Πρόκληση αποτελούν οι βελτιστοποιήσεις για τη μείωση των απαιτούμενων μετασχηματισμών, καθώς και η διαχείριση των δικτύων αντιστοιχίσεων, ώστε να αξιοποιούνται τα επιπλέον μονοπάτια ανάμεσα στους κόμβους για να μειωθεί η απώλεια πληροφορίας που μπορεί να προκληθεί κατά τις αντιστοιχίσεις.

Σύγκριση με παρούσα εργασία

Και στις δύο εργασίες θεωρούμε αδόμητο δίκτυο ομότιμων. Στο σύστημα Piazza, όμως, τα δεδομένα είναι οργανωμένα σε ποικίλες μορφές (XML, RDF/OWL), σε αντίθεση με το δικό μας σύστημα, όπου χρησιμοποιούνται σχεσιακές βάσεις δεδομένων. Το Piazza εστιάζει στη θεωρητική μελέτη και την κατασκευή αλγορίθμων για τη διαλειτουργικότητα μεταξύ πιθανώς διαφορετικών μοντέλων δεδομένων (π.χ. XML, RDF) σε δίκτυα ομότιμων. Ωστόσο, δε μελετώνται θέματα που αναφέρονται σε κριτήρια σημασιολογικής ομοιότητας των κόμβων του δικτύου ή επιλογής των κατάλληλων κόμβων για την προώθηση των ερωτημάτων, τα οποία βρίσκονται στο επίκεντρο της δικής μας εργασίας.

3.3 Αποτίμηση Συνδετικών Ερωτημάτων Προτύπων Τριάδων σε Μεγάλα Δομημένα Επικαλυπτόμενα Δίκτυα

Ένα από τα σημαντικότερα ανοικτά προβλήματα στην περιοχή που μελετάται είναι η αποτίμηση ερωτημάτων εκφρασμένων σε γλώσσες ερωτημάτων του Σημασιολογικού Ιστού πάνω σε δίκτυα ομοτίμων.

Στην εργασία αυτή μελετάται το πρόβλημα της αποτίμησης συνδετικών ερωτημάτων που αποτελούνται από πρότυπα τριάδων πάνω σε RDF δεδομένα αποθηκευμένα σε κατανεμημένους πίνακες κατακερματισμού (Distributed Hash Tables - DHT). Οι DHTs παίζουν σημαντικό ρόλο για τα P2P δίκτυα, δίνουν τη δυνατότητα να αναπτυχθούν κλιμακούμενες, εύρωστες και ανεκτικές σε λάθη κατανεμημένες εφαρμογές και έχουν πρόσφατα χρησιμοποιηθεί για την κατανεμημένη αποθήκευση και ανάκτηση διαφόρων ειδών πληροφορίας (σχεσιακή, κειμένου, RDF). Από την άλλη, οι συνδυασμοί προτύπων τριάδων είναι κομβικά στοιχεία για αρκετές γλώσσες ερωτημάτων για RDF. Η συνεισφορά της εργασίας συνοψίζεται στα παρακάτω σημεία:

- Παρουσιάζονται δύο πρωτότυποι αλγόριθμοι για την αποτίμηση συνδετικών RDF ερωτημάτων αποτελούμενων από τριάδες πάνω στο DHT τύπου Chord [SMK+01]. Επεκτείνοντας αυτές τις κλάσεις ερωτημάτων σε πλήρη συνδετικά ερωτήματα είναι σημαντικό ζήτημα για την πλήρη λειτουργικότητα των υπαρχουσών RDF γλωσσών ερωτημάτων.
- Παρουσιάζεται μια πειραματική αξιολόγηση των δύο αλγορίθμων, δίνοντας βαρύτητα στην ποσότητα των δεδομένων που αποθηκεύονται στο δίκτυο, στην

κατανομή του φορτίου και στην προκύπτουσα κίνηση στο δίκτυο. Οι αλγόριθμοι έχουν σχεδιαστεί ώστε να συμμετέχουν στην αποτίμηση του ερωτήματος όσο το δυνατόν περισσότεροι κόμβοι γίνεται, ελαχιστοποιώντας τα δεδομένα που αποθηκεύονται στο δίκτυο, καθώς και την ποσότητα της κίνησης που δημιουργούν. Στην προσπάθεια επίτευξης αυτών των στόχων συζητούνται τα tradeoffs που πρέπει να γίνουν.

Μοντέλο Συστήματος και Μοντέλο Δεδομένων

Θεωρούμε ένα επικαλυπτόμενο δίκτυο όπου όλοι οι κόμβοι είναι ίσοι (ίδιο λογισμικό, δικαιώματα και υποχρεώσεις). Κάθε κόμβος n και κάθε δεδομένο i έχει ένα μοναδικό κλειδί (key(n) και key(i)). Οι κόμβοι είναι οργανωμένοι σύμφωνα με το πρωτόκολλο του Chord (και θεωρούμε ότι έχουν συγχρονισμένα ρολόγια. Το Chord χρησιμοποιεί συναρτήσεις κατακερματισμού (SHA-1 ή MD5) για να αντιστοιχεί κλειδιά σε αναγνωριστικά. Αν Hash η συνάρτηση και id το αναγνωριστικό, ισχύει $id(n) = Hash(key(n))$ και $id(i) = Hash(key(i))$. Χρησιμοποιείται το API του Chord, στο οποίο έχουν προστεθεί ορισμένες δυνατότητες πολλαπλής διανομής (multicast).

Μοντέλο Δεδομένων

Κάθε κόμβος περιγράφει σε RDF τους πόρους που θέλει να κάνει διαθέσιμους στο δίκτυο, δημιουργώντας και εισάγοντας μετα-δεδομένα σε μορφή RDF τριάδων, και θέτει ερωτήματα για πληροφορίες που θέλει να λάβει.Κάθε κόμβος χρησιμοποιεί κάποια από τα διαθέσιμα σχήματα και δεν υποστηρίζονται αντιστοιχίσεις σχημάτων.

Χρησιμοποιείται η έννοια της RDF τριάδας $(subject, predicate, object)$ ή απλά (s, p, o) για να περιγράψει το πεδίο της εφαρμογής. Θεωρούμε ένα άπειρο, αριθμήσιμο σύνολο D από URIs και RDF κυριολεκτικά (literals). Τα subject και predicate είναι URIs από το D, ενώ το object είναι ένα URI ή ένα literal από το D.

Στην RDQL, ένα πρότυπο τριάδας είναι μια έκφραση της μορφής (s, p, o), όπου τα s και p είναι URIs ή μεταβλητές και το o είναι URI, μεταβλητή ή literal. Ένα συνδετικό ερώτημα q είναι της μορφής $?x_1,...,?x_n : (s_1, p_1, o_1) \wedge (s_2, p_2, o_2) \wedge ... \wedge (s_n, p_n, o_n)$, όπου οι $?x_1,...,?x_n$ είναι μεταβλητές (καλούνται και μεταβλητές απάντησης), κάθε (s_i, p_i, o_i) είναι ένα πρότυπο τριάδας και κάθε μεταβλητή $?x_i$ εμφανίζεται σε τουλάχιστον ένα πρότυπο τριάδας. Μια μεταβλητή αρχίζει πάντα με το '?'. Ένα ερώτημα καλείται ατομικό αν περιέχει μόνο ένα σύνδεσμο.

Όσον αφορά στην αποτίμηση, έστω V ένα πεπερασμένο σύνολο από μεταβλητές. Μια αποτίμηση v πάνω στο V είναι μια πλήρης συνάρτηση v από το V στο D.

Επιπλέον, στην εργασία έχουν χρησιμοποιηθεί έννοιες από τη θεωρία των σχεσιακών βάσεων δεδομένων. Μια RDF βάση δεδομένων είναι ένα σύνολο από τριάδες. Έστω DB μια τέτοια βάση και q ένα συνδετικό ερώτημα της μορφής $q_1 \wedge ... \wedge q_n$. Η απάντηση στο q από την DB είναι όλες οι n-τριάδες $(v(?x_1),...,v(?x_n))$, όπου v είναι μια αποτίμηση πάνω στο σύνολο των μεταβλητών του q και $v(q_i) \in DB,\ i = 1,...,n$. Τέλος, στους ακόλουθους αλγόριθμους, κάθε ερώτημα q έχει ένα μοναδικό κλειδί, το key(q).

Ο αλγόριθμος QC

Το βασικό χαρακτηριστικό του είναι ότι το ερώτημα αποτιμάται από μια αλυσίδα κόμβων. Τα ενδιάμεσα αποτελέσματα ρέουν δια μέσου των κόμβων της αλυσίδας και τελικά ο τελευταίος κόμβος της αλυσίδας στέλνει το αποτέλεσμα πίσω στον κόμβο που έθεσε την ερώτηση. Αρχικά παρουσιάζεται ο τρόπος που οι τριάδες αποθηκεύονται και έπειτα, πώς αποτιμάται ένα ερώτημα.

Όσον αφορά στην αποθήκευση των τριάδων, έστω ένας κόμβος x που θέλει να κάνει ένα πόρο διαθέσιμο στο δίκτυο. Ο x κάνει μια περιγραφή d που χαρακτηρίζει τον πόρο και την δημοσιοποιεί. Επειδή δεν επιθυμούμε μια κεντροποιημένη λύση, χωρίζουμε το d σε τριάδες και το διασκορπίζουμε στο δίκτυο. Διαχειριζόμαστε κάθε τριάδα ξεχωριστά και την καταχωρούμε σε τρεις κόμβους. Κάθε κόμβος που λαμβάνει μια τριάδα t, την αποθηκεύει στον τοπικό πίνακα τριάδων TT. Στη συνέχεια θα μεταχειριστούμε τον TT σαν μία σχεσιακή τριπλή σχέση.

Όσον αφορά στην αποτίμηση ενός ερωτήματος, έστω ένας κόμβος x που κάνει ένα συνδετικό ερώτημα q που αποτελείται από πρότυπα τριάδων $q_1,\ldots,q_k$. Κάθε τριάδα είναι πιθανό να αποτιμηθεί από διαφορετικό κόμβο και το σύνολο των κόμβων αυτών αποτελεί την αλυσίδα του ερωτήματος για το q.

Ο κόμβος x καθορίζει τον κόμβο n που θα αποτιμήσει την τριάδα q_1 χρησιμοποιώντας μία από τις σταθερές που εμφανίζονται στο q_1. Ο n αποτιμά την κατάλληλη τριάδα χρησιμοποιώντας το τοπικό TT. Αν η αποτίμηση οδηγήσει σε μια μη-κενή σχέση, επιλέγεται ο επόμενος κόμβος που θα συνεχίσει την αποτίμηση των υπόλοιπων τριάδεων. Σε περίπτωση κενής σχέσης, η αποτίμηση σταματά και μια κενή απάντηση στέλνεται στον κόμβο x. Μόλις αποτιμηθεί και η τελευταία τριάδα, στέλνεται απάντηση στον x.

Θα πρέπει να σημειωθεί εδώ ότι η σειρά με την οποία τα διάφορα πρότυπα τριάδων του q αποτιμούνται είναι σημαντική και επηρεάζει το φορτίο επεξεργασίας του ερωτήματος και άλλους πόρους.

Ο αλγόριθμος SBV

Ο αλγόριθμος SBV επεκτείνει τις ιδέες του QC για να πετύχει καλύτερη διανομή του φορτίου επεξεργασίας. Εκμεταλλευόμενος τις τιμές των αντιστοιχιζόμενων τριάδων που βρίσκονται στην πορεία της επεξεργασίας, διανέμει την ευθύνη αποτίμησης του ερωτήματος σε περισσότερους κόμβους, δημιουργώντας πολλαπλές αλυσίδες (ο QC δημιουργεί μόνο μία).

Όσον αφορά στην καταχώρηση μιας νέας τριάδας,, έστω μια τριάδα $t = (s, p, o)$. Το t θα αποθηκευτεί στους κόμβους στους οποίους αντιστοιχούν τα αναγνωριστικά Hash(s), Hash(p), Hash(o), Hash(s+p), Hash(s+o), Hash(p+o) και Hash(s+p+o). Αυτά τα ρεπλίκα θα χρησιμοποιηθούν για καλύτερη κατανομή φορτίου.

Όσον αφορά στη αποτίμηση ενός ερωτήματος, όπως και στον QC, ο κόμβος που θέτει ένα ερώτημα q της μορφής $q_1 \wedge ... \wedge q_n$ στέλνει το q σε ένα κόμβο r_1 ο οποίος μπορεί να αποτιμήσει την πρώτη τριάδα q_1. Στη συνέχεια, το πλάνο αποτίμησης του ερωτήματος δημιουργείται δυναμικά εκμεταλλεύοντας τις τιμές των αντιστοιχιζόμενων τριάδων που οι κόμβοι βρίσκουν σε κάθε βήμα ώστε να μειώσουν το φορτίο επεξεργασίας του ερωτήματος. Καθώς αναμένεται να έχουμε πολλαπλές αντιστοιχιζόμενες τιμές για τις μεταβλητές του q_1, θα έχουμε και πολλαπλούς επόμενους κόμβους όπου τα νέα ερωτήματα θα συνεχίσουν την αποτίμησή τους. Έτσι, πολλαπλές αλυσίδες αναλαμβάνουν την ευθύνη της αποτίμησης του q. Οι κόμβοι στα φύλλα αυτών των αλυσίδων θα στείλουν απαντήσεις πίσω στον κόμβο που έθεσε το q.

Για να καθορίσουμε τον κόμβο που θα αποτιμήσει μια τριάδα, χρησιμοποιούμε τα σταθερά μέρη των τριάδων, όπως και στον QC. Η διαφορά είναι ότι αν υπάρχουν πολλαπλές σταθερές σε μια τριάδα, χρησιμοποιούμε ένα συνδυασμό τους για να υπολογίσουμε το αναγνωριστικό που θα καθορίσει τον επόμενο κόμβο. Έτσι βελτιώνεται η κατανομή του φορτίου επεξεργασίας.

Βελτιστοποίηση της κίνησης του δικτύου

Ένας νέος πίνακας δρομολόγησης με όνομα IPC (IP Cache) μπορεί να χρησιμοποιηθεί με τους παραπάνω αλγόριθμους για να μειώσει σημαντικά την κίνηση στο δίκτυο. Και στους δύο αλγορίθμους, η αποτίμηση ενός ερωτήματος περνάει από μια σειρά κόμβων. Μπορούμε να εκμεταλλευτούμε το γεγονός ότι παρόμοια ερωτήματα θα ακολουθήσουν διαδρομές με κοινούς κόμβους. Έστω ένας κόμβος x_j ο οποίος συμμετέχει στην αποτίμηση ενός ερωτήματος q και χρειάζεται να στείλει ένα μήνυμα σε ένα επόμενο κόμβο x_{j+1}, ο οποίος απέχει $O(\log N)$ βήματα. Αφού γίνει πρώτη φορά η αποστολή του μηνύματος, ο x_j μπορεί να κρατήσει την IP διεύθυνση του x_{j+1}. Μελλοντικά ο x_j θα μπορεί να στέλνει μηνύματα στον x_{j+1} σε μόλις ένα βήμα. Αυτό μειώνει σημαντικά την κίνηση του δικτύου, ενώ ταυτόχρονα

μειώνεται και το φορτίο δρομολόγησης από τους κόμβους του δικτύου. Το κόστος συντήρησης του IPC είναι τοπικό.

Πειράματα

Για την αξιολόγηση των αλγορίθμων υλοποιήθηκε ένας εξομοιωτής του Chord στην Java, πάνω στον οποίο αναπτύχθηκαν οι αλγόριθμοι. Δημιουργήθηκαν τριάδες και ερωτήματα RDF θεωρώντας ένα RDFS σχήμα. Οι μετρικές που μελετήθηκαν είναι η ποσότητα της κίνησης που δημιουργείται στο δίκτυο και η κατανομή του φορτίου επεξεργασίας και αποθήκευσης ανάμεσα στους κόμβους του δικτύου. Τα συμπεράσματα που προέκυψαν είναι τα παρακάτω:

- Ο QC προκαλεί λιγότερη κίνηση από τον SBV
- Με τη χρήση του IPC, καθώς το δίκτυο εκπαιδεύεται μειώνεται όλο και περισσότερο η κίνηση που προκαλείται.
- Ο QC μπορεί να αξιοποιήσει καλύτερα το IPC
- Το κόστος συντήρησης του IPC σε μέγεθος είναι πολύ μεγαλύτερο στον SBV
- Και οι δύο αλγόριθμοι δημιουργούν το ίδιο φορτίο επεξεργασίας στο δίκτυο, αλλά ο SBV πετυχαίνει να κατανέμει το φορτίο σε πολύ μεγαλύτερο αριθμό κόμβων.
- Το συνολικό φορτίο αποθήκευσης στον QC είναι μικρότερο απ΄ ότι στον SBV
- Το υψηλότερο φορτίο αποθήκευσης είναι ένα τίμημα που πρέπει να πληρώσουμε αν επιθυμούμε την καλύτερη κατανομή του φορτίου επεξεργασίας που προσφέρει ο SBV.

Σύγκριση με παρούσα εργασία

Σε αντίθεση με το αδόμητο δίκτυο που θεωρούμε για το σύστημά μας, εδώ χρησιμοποιούνται DHT δίκτυα, που είναι μια από τις δημοφιλέστερες μορφές δομημένων δικτύων ομότιμων κόμβων. Στα DHT δίκτυα μπορούμε να ξέρουμε πού να αναζητήσουμε τα δεδομένα, μιας και η κατανομή τους γίνεται με βάση μια συνάρτηση κατακερματισμού. Επιπλέον, στη δικιά μας εργασία τα δεδομένα είναι οργανωμένα σε σχεσιακές βάσεις δεδομένων, ενώ εδώ τα δεδομένα είναι σε RDF μορφή.

Με αφορμή την εργασία αυτή γνωρίσαμε τρία πολύ σημαντικά στοιχεία που αφορούν στην αποδοτικότητα ενός δικτύου ομότιμων: τη σωστή κατανομή του φόρτου επεξεργασίας των ερωτημάτων, την ελαχιστοποίηση της κίνησης στο δίκτυο και την ικανότητα κλιμάκωσης του συστήματος όταν ο αριθμός των δεδομένων αυξάνει σημαντικά.

4

Ανάλυση Απαιτήσεων Συστήματος

Το κεφάλαιο αυτό αφορά στην ανάλυση των απαιτήσεων του συστήματος. Για να γίνει ευκολότερη και πιο αποτελεσματική η ανάλυση του συστήματος, το χωρίζουμε σε υποσυστήματα, τα οποία παρουσιάζουμε στην ενότητα 4.1. Στην ενότητα 4.2 δίνεται η αναλυτική περιγραφή των λειτουργιών που επιτελεί κάθε υποσύστημα.

4.1 Αρχιτεκτονική

Τα επιμέρους τμήματα – υποσυστήματα στα οποία θα μπορούσε να χωριστεί το παρόν σύστημα είναι τα ακόλουθα.

- Υποσύστημα Δημιουργίας Τοπολογίας Δικτύου και Αρχικοποίησης Κόμβων: Το υποσύστημα αυτό είναι υπεύθυνο για τη δημιουργία της τοπολογίας του επικαλύπτοντος δικτύου, καθώς και για την αρχικοποίηση των κόμβων του δικτύου (π.χ. φόρτωση του τοπικού σχήματος βάσης δεδομένων ή της τοπικής οντολογίας)

- Υποσύστημα Παραμετροποίησης Συστήματος: Το υποσύστημα αυτό δίνει τη δυνατότητα στο χρήστη να καθορίσει διάφορες από τις παραμέτρους του συστήματος. Ενδεικτικά αναφέρουμε τον αριθμό των κόμβων του συστήματος και την τιμή της παραμέτρου a στον τύπο που έχει δοθεί για τον υπολογισμό της F-μετρικής (βλ. Ενότητα 2.3).

- Υποσύστημα Υπολογισμού Recall (ευστοχίας) και Precision (ακρίβειας): Το υποσύστημα αυτό υλοποιεί όλους εκείνους τους αλγόριθμους που χρησιμοποιούνται για τον υπολογισμό του recall και του precision μεταξύ κλάσεων, ιδιοτήτων, περιορισμών.

- Υποσύστημα Σύγκρισης Σχημάτων Κόμβων: Το υποσύστημα αυτό υπολογίζει το βαθμό ομοιότητας μεταξύ των σχημάτων δύο κόμβων, υπολογίζοντας το recall και το precision μεταξύ των λιστών των κλάσεων στις οποίες αντιστοιχίζεται το σχήμα της βάσης κάθε κόμβου.

- Υποσύστημα Επανεγγραφής, Σύγκρισης και Απάντησης Ερωτημάτων: Με το συγκεκριμένο υποσύστημα, ένα ερώτημα που έρχεται σε ένα κόμβο από ένα γειτονικό του μπορεί αρχικά να επανεγγραφεί. Στη συνέχεια μπορεί να υπολογιστεί η ομοιότητα των δύο κόμβων (του αποστολέα και του παραλήπτη) λαμβάνοντας υπόψη και τους περιορισμούς που θέτει το δεδομένο ερώτημα και τέλος να απαντηθεί το ερώτημα, αν το θελήσει ο αποστολέας.

- Υποσύστημα Κατανομής Συνόλου Δεδομένων: Το υποσύστημα αυτό είναι υπεύθυνο για την κατανομή του συνόλου των αρχικών διαθέσιμων δεδομένων στο δίκτυο στους κόμβους του συστήματος, κάνοντας χρήση οριζόντιας και κατακόρυφης κατάτμησης.

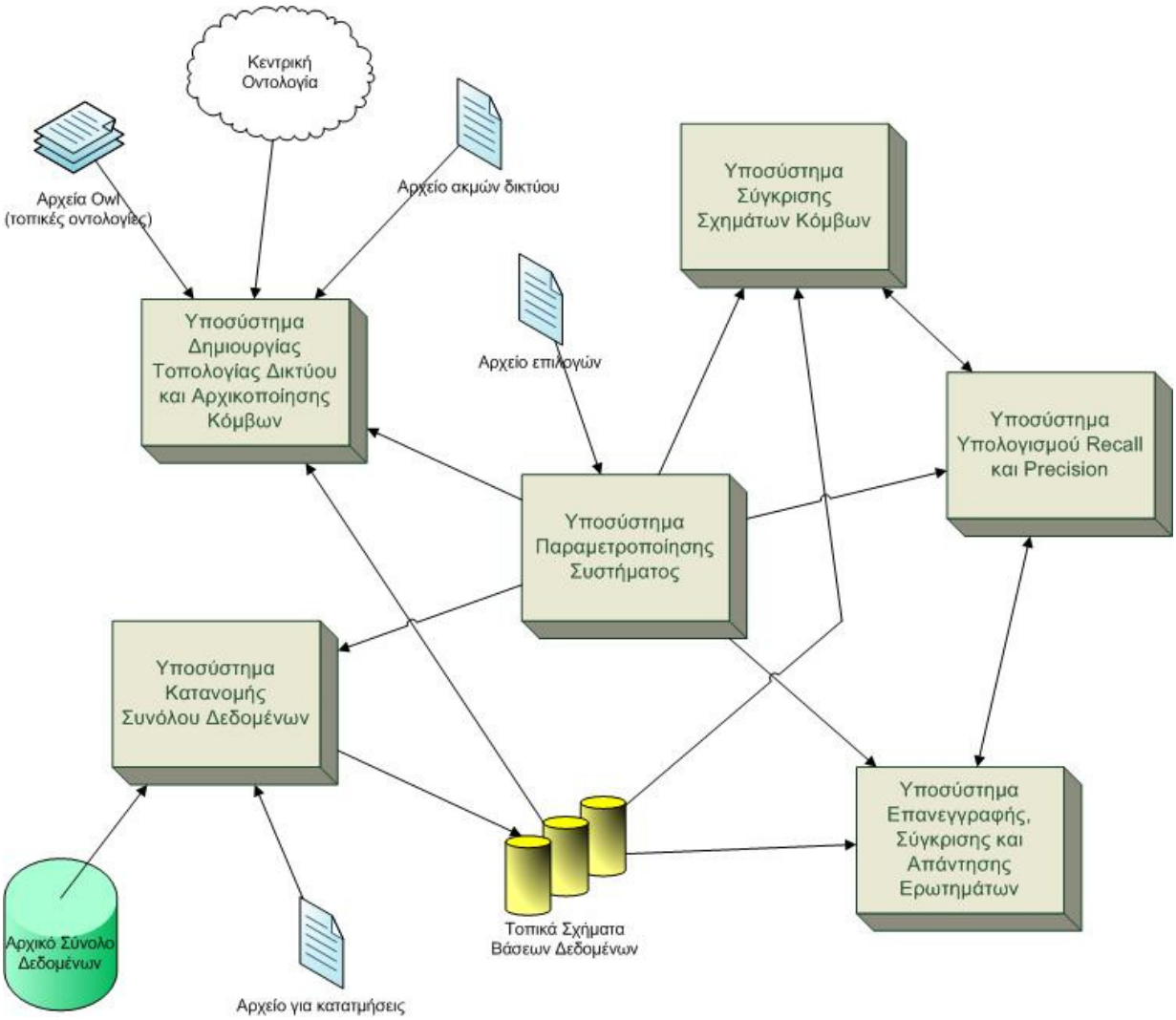
Στην Εικόνα 4.1 δίνεται ένα γενικό σχήμα στο οποίο εικονίζονται όλα τα υποσυστήματα του συστήματος, καθώς και οι συσχετίσεις μεταξύ τους.

4.2 Περιγραφή Λειτουργιών

Στην ενότητα αυτή περιγράφονται αναλυτικότερα οι λειτουργίες που επιτελεί καθένα από τα υποσυστήματα του συστήματος. Σε κάθε περίπτωση δίνεται κι ένα Διάγραμμα Ροής Δεδομένων που παρουσιάζει σχηματικά τις λειτουργίες αυτές.

4.2.1 Υποσύστημα Δημιουργίας Τοπολογίας Δικτύου και Αρχικοποίησης Κόμβων

Το υποσύστημα αυτό επιτελεί δύο βασικές λειτουργίες. Η πρώτη είναι η δημιουργία της τοπολογίας του δικτύου. Για να δημιουργήσουμε μια τυχαία τοπολογία ενός επικαλύπτοντος



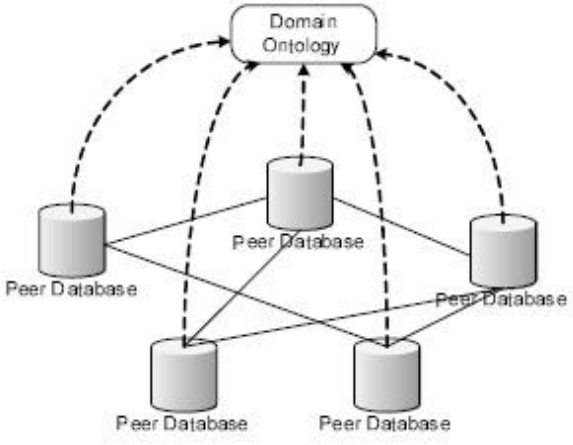
Εικόνα 4.1 Αρχιτεκτονική του συστήματος

δικτύου αποτελούμενου από σημαντικό αριθμό κόμβων, χρησιμοποιήθηκε μια γεννήτρια γράφων. Περισσότερες πληροφορίες για την εφαρμογή αυτή δίνονται στην Ενότητα 6.2 όπου παρουσιάζονται και τα υπόλοιπα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν. Η γεννήτρια αυτή παράγει ως

έξοδο ένα αρχείο κειμένου σε κάθε γραμμή του οποίου αναφέρεται κι από μια ακμή του γράφου (δύο αριθμοί που δείχνουν τα αναγνωριστικά των δύο κόμβων που συνδέονται με την κάθε ακμή). Το παρόν υποσύστημα χρησιμοποιεί το αρχείο αυτό για να δημιουργήσει έναν πίνακα κατακερματισμού ο οποίος περιέχει για κάθε κόμβο του δικτύου μια λίστα με τους γειτονικούς κόμβους.

Η δεύτερη λειτουργία που επιτελεί το υποσύστημα είναι η αρχικοποίηση των κόμβων του δικτύου. Προτού, όμως, αναφερθούμε στη λειτουργία αυτή, κρίνεται σκόπιμη η παρουσίαση της δομής του δικτύου μας και κυρίως η δομή ενός τυπικού κόμβου του δικτύου.

Το σύστημά μας χρησιμοποιεί μια κεντρική οντολογία ώστε να επισημειώνονται σημασιολογικά οι κόμβοι του δικτύου, περιγράφοντας το είδος της πληροφορίας που κάθε κόμβος κάνει διαθέσιμη στους υπόλοιπους κόμβους. Μια αφηρημένη σχεδίαση του συστήματος δίνεται στην Εικόνα 4.2. Η σημασιολογική επισημείωση πραγματοποιείται δηλώνοντας αντιστοιχίσεις ανάμεσα σε όρους του τοπικού σχεσιακού σχήματος και σε όρους της οντολογίας. Η ύπαρξη μιας καθολικής οντολογίας θα μπορούσε να θεωρηθεί ως μειονέκτημα του συστήματος, καθώς είναι δύσκολο σε ένα πραγματικό σύστημα οι σημασιολογικές επισημειώσεις να αναφέρονται σε μία μοναδική οντολογία. Εντούτοις, το πρόβλημα αυτό είναι εύκολο να αντιμετωπιστεί θεωρώντας αντιστοιχίσεις μεταξύ των οντολογιών. Κάτι τέτοιο, όμως, ξεφεύγει από τους σκοπούς της παρούσας εργασίας και χωρίς βλάβη της γενικότητας θεωρούμε μία μόνο οντολογία.



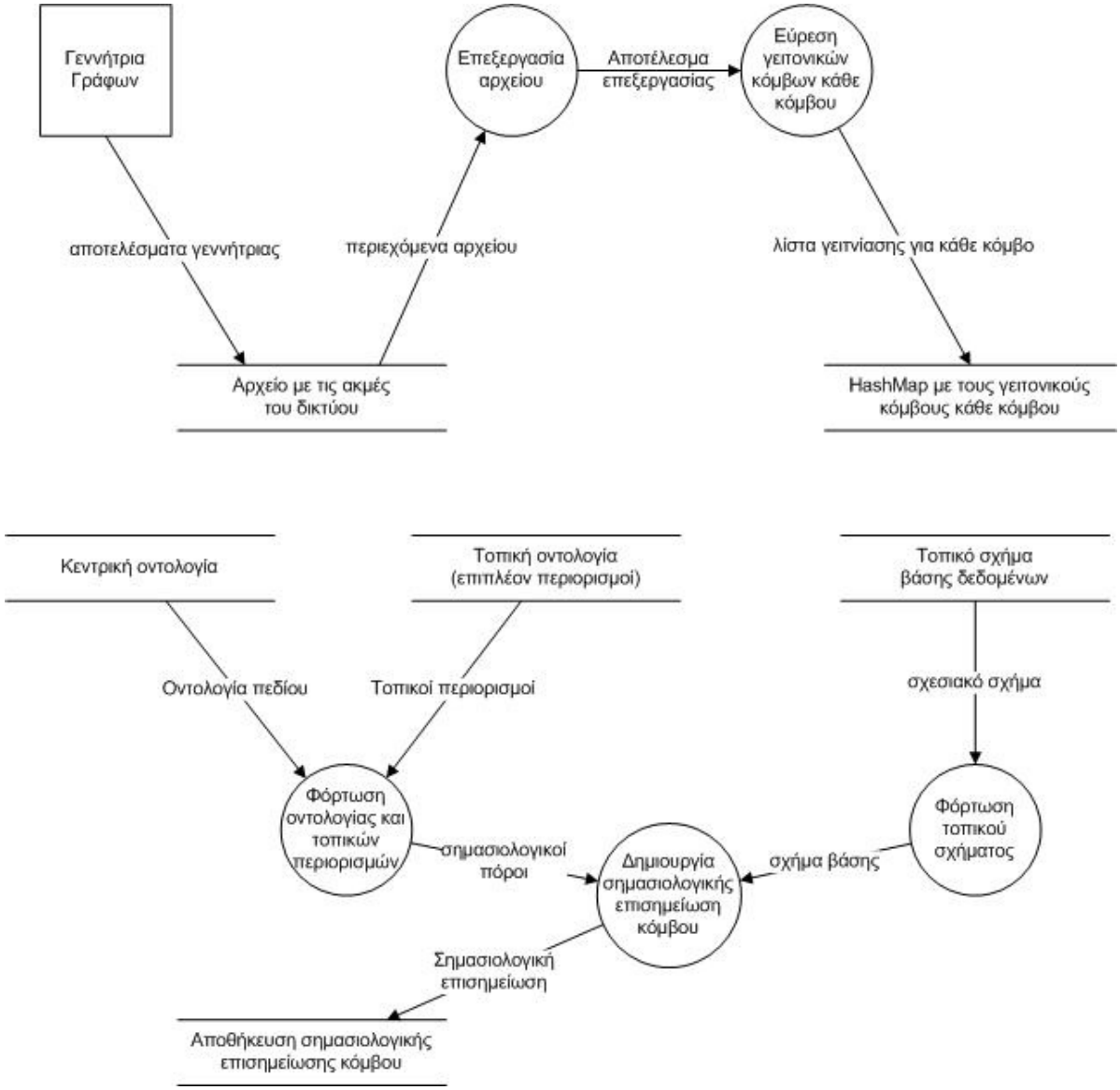
Εικόνα 4.2 Δομή του δικτύου των σημασιολογικών κόμβων

Κάθε σημασιολογικός κόμβος (έτσι αποκαλούμε τους κόμβους του δικτύου, δεδομένου ότι περιέχουν και σημασιολογική πληροφορία) είναι στην ουσία μια τριάδα $\mathcal{P} = (\mathcal{R}, \mathcal{O}, \mathcal{A})$, όπου $\mathcal{R}$ είναι το τοπικό σχήμα της βάσης του κόμβου, $\mathcal{O}$ η κεντρική οντολογία και $\mathcal{A}$ η σημασιολογική αντιστοίχιση του κόμβου. Το $\mathcal{A}$ περιέχει το σύνολο των αντιστοιχίσεων A για τις σχέσεις του $\mathcal{R}$. Κάθε A από ένα σύνολο της μορφής (R, $\mathcal{C}$) και ένα σύνολο από ζεύγη της μορφής (R.t, C_i.P), $C_i \in \mathcal{C}$. Έτσι, μια σχέση R είναι σημασιολογικά επισημειωμένη από ένα σύνολο κλάσεων $\mathcal{C}$. Κάθε $C_i \in \mathcal{C}$ είναι μια κλάση της οντολογίας, πιθανώς επαυξημένη με επιπλέον περιορισμούς ώστε να προσδιορίζεται με σαφήνεια η σημασιολογία της υποκείμενης σχέσης, δηλαδή $C_i \equiv C_i' \pi_k R_k$, $C_i' \in \mathcal{O}$. Τα χαρακτηριστικά (attributes) R.t

επισημειώνονται μέσω των ιδιοτήτων (properties) των ίδιων συνόλων κλάσεων, δηλαδή (R.t, C_i.P), $C_i \in C$.

Κατά την αρχικοποίησή του, κάθε κόμβος χρησιμοποιώντας το τοπικό σχήμα της βάσης του, την κεντρική οντολογία του, καθώς και τους επιπλέον περιορισμούς που θέτει, επισημειώνει σημασιολογικά το περιεχόμενό του.

Το διάγραμμα ροής δεδομένων του υποσυστήματος δίνεται στην Εικόνα 4.3.



Εικόνα 4.3 Διάγραμμα Ροής Δεδομένων Υποσυστήματος Δημιουργίας Τοπολογίας Δικτύου και Αρχικοποίησης Κόμβων

Σενάριο Λειτουργίας

Για να γίνουν ευκολότερα κατανοητά τα ανωτέρω, θα παρουσιαστεί ένα απλό σενάριο στο οποίο θεωρούμε δύο κόμβους, τους $\mathcal{P}_1$ και $\mathcal{P}_2$, οι οποίοι θέλουν να ανταλλάξουν δεδομένα για

μουσικές μπάντες. Στον ακόλουθο πίνακα δίνονται ορισμένοι συμβολισμοί που θα χρησιμοποιηθούν κατά τη διάρκεια του κεφαλαίου:

Κατηγόρημα της OWL	**Συμβολισμός**	**Περιγραφή**
owl:Class	C	Κλάση
owl:ObjectProperty	P	Ιδιότητα Αντικειμένου
owl:DatatypeProperty	P	Ιδιότητα Τύπων Δεδομένων
owl:equivalentClass	$C_1 \equiv C_2$	Ισοδυναμία Κλάσεων
rdfs:subClassOf	$C_1 \sqsubseteq C_2$	Υπαγωγή Κλάσεων
owl:equivalentProperty	$P_1 \equiv P_2$	Ισοδυναμία Ιδιοτήτων
rdfs:subPropertyOf	$P_1 \sqsubseteq P_2$	Υπαγωγή Ιδιοτήτων
owl:Thing	$\top$	Η Κλάση με όλα τα Άτομα
owl:Nothing	$\bot$	Η Κλάση με κανένα Άτομο
owl:DataRange	d	Τύπος Δεδομένων
rdfs:domain	domain(P)	Το πεδίο ορισμού της ιδιότητας
rdfs:range	range(P)	Το σύνολο τιμών της ιδιότητας
owl:allValuesFrom	∀P.C	Περιορισμός σε Ιδιότητα Αντικειμένου
owl:allValuesFrom	∀P.d	Περιορισμός σε Ιδιότητα Τύπου
owl:minCardinality	≥ n P	Περιορισμός Ελάχιστης Πληθυκότητας
owl:maxCardinality	≤ n P	Περιορισμός Μέγιστης Πληθυκότητας

Πίνακας 4.1 Συμβολισμοί που χρησιμοποιήθηκαν

Θεωρούμε ότι το σχήμα της βάσης του $\mathcal{P}_1$ είναι το:

$\mathcal{P}_1$: bands(name, members, year)

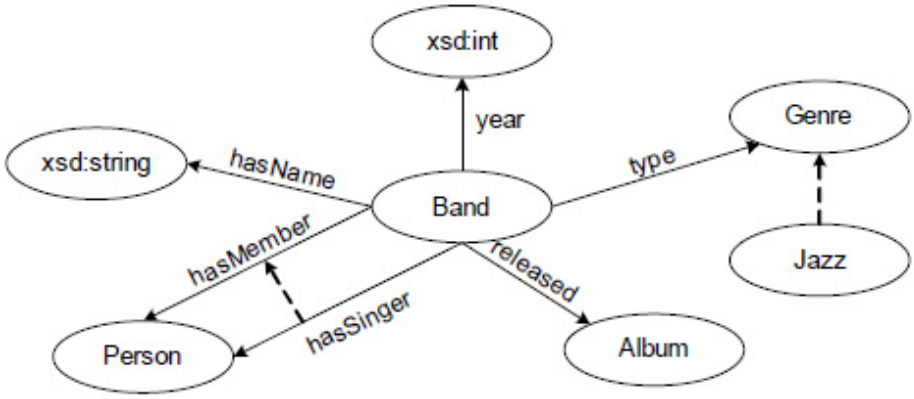
και το σχήμα του $\mathcal{P}_2$ είναι το :

$\mathcal{P}_2$: bands(name, singer, year)

Οι αντιστοιχίσεις μεταξύ των σχημάτων των δύο κόμβων είναι:

M_{P_1, P_2} : bands(name, members, year) :- bands(name, singer, year)

Μια οντολογία για τον τομέα της μουσικής χρησιμοποιείται για να περιγράψει σημασιολογικά τα περιεχόμενα των κόμβων. Η οντολογία δίνεται σχηματικά στην Εικόνα 4.4. Οι κόμβοι αναπαριστούν κλάσεις ή τύπους δεδομένων, ενώ οι ακμές αναπαριστούν ιδιότητες. Οι διακεκομμένες γραμμές δηλώνουν σχέση υπαγωγής.



Εικόνα 4.4 Οντολογία για τον τομέα της μουσικής

Θεωρούμε ότι ο $\mathcal{P}_1$ περιλαμβάνει πληροφορίες για μπάντες που έχουν τουλάχιστον ένα άλμπουμ, ενώ ο $\mathcal{P}_2$, όντας πιο συγκεκριμένος, αποθηκεύει πληροφορίες για μπάντες που παίζουν Jazz μουσική, δημιουργήθηκαν πριν το 2000 και έχουν εκδώσει τουλάχιστον τρία άλμπουμ. Οι πληροφορίες αυτές μπορούν να δηλωθούν με σαφήνεια, επισημειώνοντας τις σχέσεις και τα χαρακτηριστικά (attributes) του τοπικού σχήματος κάνοντας χρήση της κεντρικής οντολογίας, όπως φαίνεται στον Πίνακας 4.2.

Στοιχείο Σχήματος	**Στοιχείο Οντολογίας**
bands.name	hasName
bands.members	hasMember
bands.year	year
bands	P1_Band

Κόμβος P_1

Στοιχείο Σχήματος	**Στοιχείο Οντολογίας**
bands.name	hasName
bands.singer	hasSinger
bands.year	year
bands	P2_Band

Κόμβος P_2

Πίνακας 4.2 Σημασιολογική επισημείωση των κόμβων

Οι P1_Band και P2_Band είναι δύο νέες κλάσεις, οι οποίες προέκυψαν από την κλάση Band, προσθέτοντας τους κατάλληλους για τον κάθε κόμβο περιορισμούς. Ο ακριβής ορισμός των κλάσεων αυτών είναι ο ακόλουθος:

$$P1\_Band \equiv Band \sqcap \geq 1\ released$$

$$P2\_Band \equiv Band \sqcap \forall type.Jazz \sqcap \geq 3\ released \sqcap \forall year.(\leq 2000)$$

Με βάση τα παραπάνω είναι εμφανές ότι οι πληροφορίες του $\mathcal{P}_2$ είναι εν μέρει μόνο επαρκείς στον $\mathcal{P}_1$, ενώ αντίθετα ο $\mathcal{P}_1$ περιέχει πολλές πληροφορίες που είναι άσχετες με τα ενδιαφέροντα του $\mathcal{P}_2$. Εκμεταλλευόμαστε, λοιπόν, αυτή τη σημασιολογική πληροφορία, ώστε

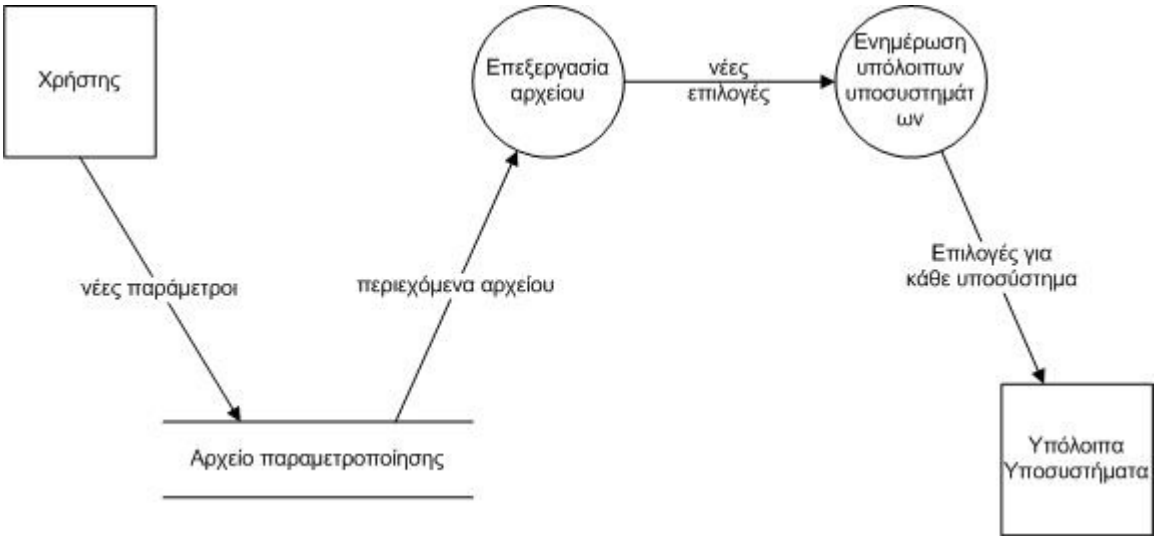
να υπολογίσουμε την σχέση αυτή των δύο κόμβων κι έτσι να επωφεληθούμε στην ανταλλαγή δεδομένων μεταξύ τους.

4.2.2 Υποσύστημα Παραμετροποίησης Συστήματος

Το υποσύστημα αυτό κάνει χρήση ενός αρχείο στο οποίο ο χρήστης του συστήματος μπορεί να αλλάξει ορισμένες επιλογές για τη λειτουργία της εφαρμογής. Αφού επεξεργαστεί τις επιλογές του χρήστη, ενημερώνει τα υπόλοιπα υποσυστήματα, ώστε να εναρμονίσουν τις λειτουργίες τους με αυτό.

Ο χρήστης έχει αρχικά τη δυνατότητα να αλλάξει επιλογές που σχετίζονται με τις βάσεις δεδομένων και την κεντρική οντολογία, όπως είναι η διεύθυνση των βάσεων, το όνομα οδηγού για την προσπέλασή τους μέσω της εφαρμογής, το όνομα της οντολογίας και άλλα. Παράλληλα υπάρχουν επιλογές για την παραμετροποίηση του υπολογισμού του recall και του precision. Χαρακτηριστικό παράδειγμα είναι οι επιλογές που δίνουν μεγαλύτερο εύρος στις τιμές του recall και του precision, καθώς και η παράμετρος που καθορίζει αν στην F-μετρική θα δοθεί περισσότερο βάρος στο recall ή στο precision (παράμετρος α στη σχέση F_α). Τέλος, υπάρχουν επιλογές για τον αριθμό των κόμβων στο σύστημα, καθώς και για τον αριθμό των γειτονικών κόμβων που επιλέγεται να αποτιμήσουν ένα ερώτημα που τίθεται.

Στην Εικόνα 4.5 δίνεται το διάγραμμα ροής δεδομένων του υποσυστήματος.



Εικόνα 4.5 Διάγραμμα Ροής Δεδομένων Υποσυστήματος Παραμετροποίησης Συστήματος

4.2.3 Υποσύστημα Υπολογισμού Recall και Precision

Το υποσύστημα αυτό είναι υπεύθυνο για τις λειτουργίες που έχουν σχέση με τον υπολογισμό του recall και του precision. Όπως έχει ήδη αναφερθεί, το recall (ευστοχία) και το precision (ακρίβεια) είναι δυο έννοιες δανεισμένες από το πεδίο της Ανάκτησης Πληροφοριών (Ενότητα 3.2), οι οποίες χρησιμοποιούνται στην παρούσα εργασία με σκοπό τον υπολογισμό τόσο της ομοιότητας μεταξύ των σχημάτων δύο κόμβων όσο και της ομοιότητας των κόμβων σε σχέση με ένα συγκεκριμένο ερώτημα που τίθεται.

Το παρόν υποσύστημα υπολογίζει το recall και το precision ανάμεσα σε δύο σημασιολογικούς πόρους και χρησιμοποιείται από τα υποσυστήματα υπολογισμού της ομοιότητας των κόμβων που θα παρουσιαστούν παρακάτω. Όταν αναφερόμαστε σε σημασιολογικούς πόρους, συμπεριλαμβάνουμε κλάσεις της οντολογίας, ιδιότητες των κλάσεων, περιορισμούς, λίστες από κλάσεις και λίστες από ιδιότητες. Γενικά, για τον υπολογισμό του recall και του precision βασιζόμαστε στη σημασιολογική πληροφορία που υπάρχει στην κεντρική οντολογία και κυρίως στην ιεραρχία των κλάσεων, την ιεραρχία των ιδιοτήτων και τους περιορισμούς πάνω στις ιδιότητες των κλάσεων. Ακολουθεί ο τρόπος με τον οποίο υπολογίζονται οι δύο μετρικές για τους διάφορους πόρους.

Στην περίπτωση κλάσεων οι οποίες δεν περιγράφονται από καμία ιδιότητα, το recall και το precision μπορούν να υπολογιστούν από την αναλογία των κοινών προγόνων τους. Αν το A(C) συμβολίζει το σύνολο των υπερκλάσεων μιας κλάσης C, τότε:

$$recall(C_1, C_2) = \frac{|A(C_1) \cap A(C_2)|}{|A(C_2)|}$$

$$precision(C_1, C_2) = \frac{|A(C_1) \cap A(C_2)|}{|A(C_1)|}$$

Με όμοιο τρόπο, το recall και το precision ανάμεσα σε δύο ιδιότητες, p_1 και p_2 μπορεί να υπολογιστεί από την αναλογία των κοινών υπεριδιοτήτων, όπως προκύπτουν από την ιεραρχία ιδιοτήτων στην κεντρική οντολογία:

$$recall(p_1, p_2) = \frac{|A(p_1) \cap A(p_2)|}{|A(p_2)|}$$

$$precision(p_1, p_2) = \frac{|A(p_1) \cap A(p_2)|}{|A(p_1)|}$$

Όσον αφορά στους περιορισμούς που ορίζονται πάνω στις ιδιότητες μιας κλάσης, μπορούμε να διακρίνουμε τρεις περιπτώσεις:

(α) περιορισμοί τιμών σε ιδιότητες αντικειμένου (για παράδειγμα $\forall type.Jazz$)

(β) περιορισμοί τιμών πάνω σε ιδιότητες τύπων δεδομένων (για παράδειγμα $\forall year.(\leq 2000)$)

(γ) περιορισμοί πληθυκότητας (για παράδειγμα $\geq 3 released$)

Στην (α) περίπτωση χρησιμοποιούμε το recall και το precision μεταξύ των δύο κλάσεων στις οποίες περιορίζονται οι τιμές των ιδιοτήτων. Στις άλλες δύο περιπτώσεις βγάζουμε σχέσεις υπαγωγής, όπως αυτές που δίνονται στον Πίνακας 4.3.Πίνακας 4.1

condition	**recall(R_1, R_2)**	**precision(R_1, R_2)**
R1 ≡ R2	1	1
R1 ⊑ R2	1	0.5
R1 ⊒ R2	0.5	1
¬(R1 ⊓ R2 ⊑ ⊥)	0.5	0.5
R1 ⊓ R2 ⊑ ⊥	0	0

Πίνακας 4.3 Διαφορετικές περιπτώσεις υπαγωγής ανάμεσα σε δύο περιορισμούς R1 και R2

Στην περίπτωση (β) υπάρχουν διάφορες παραλλαγές ανάλογα με τον τύπο των δεδομένων στον οποίο αναφέρονται οι ιδιότητες. Ο προσδιορισμός του recall και του precision στις περιπτώσεις αυτές και η βελτίωση της ήδη υπάρχουσας τεχνικής παρουσίασε έντονο αλγοριθμικό ενδιαφέρον, το οποίο αναλύεται στην Ενότητα 6.2, όπου περιγράφονται και οι υπόλοιποι αλγόριθμοι που υλοποιήθηκαν.

Αν στους τύπους που υπολογίζουν τις μετρικές για τις ιδιότητες λάβουμε υπόψη και τους περιορισμούς που επιβάλλονται στις ιδιότητες, οι τύποι γίνονται:

$$recall(p_1, p_2) = \frac{|A(p_1) \cap A(p_2)|}{|A(p_2)|} \cdot \prod_{R(p_2)} recall(R_i'(p_1), R_i(p_2))$$

$$precision(p_1, p_2) = \frac{|A(p_1) \cap A(p_2)|}{|A(p_1)|} \cdot \prod_{R(p_1)} recall(R_i(p_1), R_i'(p_2))$$

όπου R(p) είναι το σύνολο των περιορισμών πάνω στην ιδιότητα p και το (R'(p_1), R(p_2)) δηλώνουν ένα ζεύγος αντιστοιχιζόμενων περιορισμών (περιορισμών ίδιου τύπου). Το γινόμενο που προστέθηκε στους υπολογισμούς του recall και του precision για τις ιδιότητες χρησιμοποιήθηκε σαν μια φθίνουσα συνάρτηση που δείχνει διαισθητικά ότι κάθε παράγοντας συμβάλλει αρνητικά στο τελικό αποτέλεσμα, αφού εισάγει άλλη μια σημασιολογική προσέγγιση. Αν δεν υπάρχουν αντιστοιχιζόμενοι περιορισμοί σε μια από τις ιδιότητες που συγκρίνουμε, σε περίπτωση περιορισμού τιμής σε ιδιότητα αντικειμένου, το σύνολο τιμών που καθορίζεται στον ένα περιορισμό συγκρίνεται με το εξ ορισμού εύρος ζώνης της άλλης ιδιότητας. Στις άλλες δύο περιπτώσεις, η τιμή του recall και του precision τίθεται σε μια προκαθορισμένη τιμή.

Με βάση τα παραπάνω, δοθέντων δύο κλάσεων C_1 και C_2, το recall και το precision υπολογίζονται προσθέτοντας τα αποτελέσματα για κάθε ιδιότητά τους και κανονικοποιώντας ως προς τον αριθμό των ιδιοτήτων:

$$recall(C_1, C_2) = \frac{\sum_{P(C_2)} recall(p'(C_1), p(C_2))}{|P(C_2)|}$$

$$precision(C_1, C_2) = \frac{\sum_{P(C_1)} precision(p(C_1), p'(C_2))}{|P(C_1)|}$$

όπου P(C) το σύνολο των ιδιοτήτων της κλάσης C και ($p(C_1)$, $p'(C_2)$) ένα ζεύγος αντιστοιχιζόμενων ιδιοτήτων στις κλάσεις που συγκρίνουμε.

Η σύγκριση μπορεί να επεκταθεί στη σύγκριση συνόλων κλάσεων, συγκρίνοντας κάθε κλάση στο ένα σύνολο με την αντίστοιχη ιδιότητα του άλλου συνόλου, δηλαδή την ιδιότητα που μεγιστοποιεί το recall ή το precision αντίστοιχα και στη συνέχεια κανονικοποιώντας ως προς την πληθυκότητα των συνόλων. Έτσι, για δύο σύνολα κλάσεων, C_1 και C_2 ισχύει:

$$recall(C_1, C_2) = \frac{\sum_{C_i \in C_1} \max_{C_j \in C_2} recall(C_i, C_j)}{|C_1|}$$

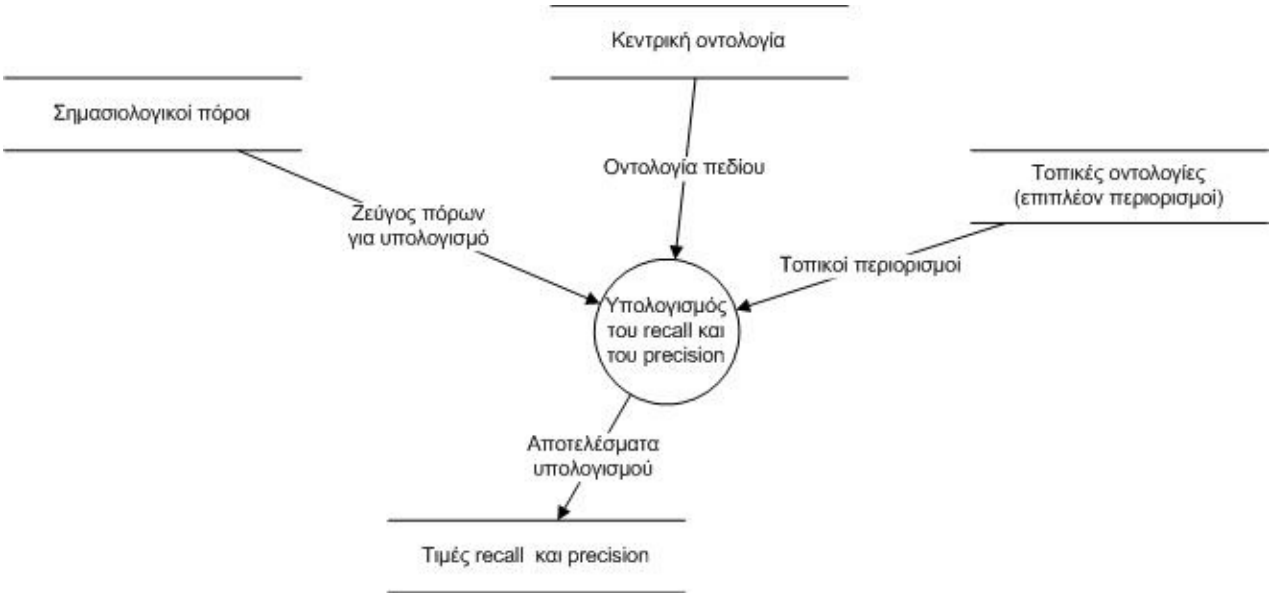
$$precision(C_1, C_2) = \frac{\sum_{C_j \in C_2} \max_{C_i \in C_1} precision(C_i, C_j)}{|C_2|}$$

Το διάγραμμα ροής δεδομένων του υποσυστήματος δίνεται στην Εικόνα 4.6.

4.2.4 Υποσύστημα Σύγκρισης Σχημάτων Κόμβων

Το υποσύστημα αυτό είναι υπεύθυνο για τον υπολογισμό της σημασιολογικής ομοιότητας ανάμεσα στις σημασιολογικές πληροφορίες δύο κόμβων. Για τον υπολογισμό της ομοιότητας χρησιμοποιείται το recall και το precision.

Η σημασιολογία του κάθε κόμβου δίνεται από την (σημασιολογική) επισημείωση του, η οποία αποτελείται από ένα σύνολο κλάσεων. Έτσι, η εκτίμηση της σημασιολογικής ομοιότητας ανάμεσα σε δύο κόμβους P_1 και P_2 ισοδυναμεί με την εκτίμηση των αντίστοιχων συνόλων κλάσεων C_{P_1} και C_{P_2}. Έστω ότι αρχικά θεωρούμε ότι κάθε κόμβος επισημειώνεται με μία μόνο κλάση. Έτσι, για να υιοθετήσουμε τις έννοιες του recall και του



Εικόνα 4.6 Διάγραμμα Ροής Δεδομένων Υποσυστήματος Υπολογισμού Recall και Precision

precision στο δικό μας περιεχόμενο, θεωρούμε ως σχετικά αντικείμενα (περισσότερες πληροφορίες για την ορολογία που χρησιμοποιείται στην Ενότητα 2.3) τα στιγμιότυπα της κλάσης στη σημασιολογική επισημείωση του κόμβου που ζητάει (requesting peer) και ως ανακτηθέντα τα στιγμιότυπα της κλάσης στην επισημείωση του κόμβου που προσφέρει (provider peer). Έτσι ισχύει:

$$recall\left(C_{P_1}, C_{P_2}\right) = \frac{\left|\left\{x \mid x \in \left(C_{p_1} \Pi C_{P_2}\right)\right\}\right|}{\left|\left\{x \mid x \in C_{P_1}\right\}\right|}$$

$$precision\left(C_{P_1}, C_{P_2}\right) = \frac{\left|\left\{x \mid x \in \left(C_{p_1} \Pi C_{P_2}\right)\right\}\right|}{\left|\left\{x \mid x \in C_{P_2}\right\}\right|}$$

Οι παραπάνω ορισμοί έχουν τις ακόλουθες ιδιότητες:

- Όταν δύο κλάσεις είναι ισοδύναμες, δηλαδή $C_{P_1} \equiv C_{P_2} \equiv C_{P_1} \Pi C_{P_2}$, τότε recall = 1 και precision = 1, που σημαίνει ότι τα περιεχόμενα των δύο κόμβων αναφέρονται σε ίδιου τύπου οντότητες.
- Όταν $C_{P_1} \sqsubseteq C_{P_2}$, τότε $C_{P_1} \Pi C_{P_2} \equiv C_{P_1}$ και άρα recall = 1 και precision < 1, που σημαίνει ότι ο P_2 διαθέτει πληροφορίες που δεν ενδιαφέρουν τον P_1.
- Όταν $C_{P_1} \sqsupseteq C_{P_2}$, τότε $C_{P_1} \Pi C_{P_2} \equiv C_{P_2}$ και άρα recall < 1 και precision = 1, που σημαίνει ότι ο P_2 μπορεί μόνο εν μέρει να καλύψει την ανάγκη του P_1 σε πληροφορία.

- Όταν $\neg(C_{P_1} \sqcap C_{P_2} \sqsubseteq \bot)$, τότε είναι recall < 1 και precision < 1, που σημαίνει ότι μόνο μέρος της πληροφορίας του ενός κόμβου είναι ενδιαφέρουσα για τον άλλο.
- Τέλος, όταν $C_{P_1} \sqcap C_{P_2} \sqsubseteq \bot$, τότε είναι recall = 0 και precision = 0, που σημαίνει ότι η σύνδεση των δύο αυτών κόμβων δεν έχει καμία σημασία, μιας και δεν έχουν κανένα κοινό τύπο πληροφορίας.

Το αρνητικό με τον παραπάνω τρόπο υπολογισμού του recall και του precision είναι ότι απαιτείται γνώση για τα στιγμιότυπα της κάθε κλάσης (τα περιεχόμενά της). Αντί αυτού του τρόπου, χρησιμοποιούμε τη σημασιολογική πληροφορία που προκύπτει από την οντολογία, όπως περιγράφηκε στο Υποσύστημα Υπολογισμού του Recall και του Precision. Το αξιοσημείωτο είναι ότι η μέθοδος αυτή που εφαρμόστηκε διατηρεί τις ιδιότητες που μόλις παρουσιάστηκαν για το recall και το precision.

Με βάση τα παραπάνω, διαπιστώνουμε ότι για να υπολογίσουμε τη σημασιολογική ομοιότητα δύο κόμβων χρησιμοποιούμε το Υποσύστημα Υπολογισμού του Recall και του Precision, το οποίο τροφοδοτούμε με τα σύνολα των κλάσεων επισημείωσης.

Το διάγραμμα ροής δεδομένων του υποσυστήματος δίνεται στην Εικόνα 4.7.

Εικόνα 4.7 Διάγραμμα Ροής Δεδομένων Υποσυστήματος Σύγκρισης Σχημάτων Κόμβων

Σενάριο Λειτουργίας (συνέχεια)

Θα εφαρμόσουμε τους ανωτέρω υπολογισμούς για να υπολογίσουμε τη σημασιολογική ομοιότητα ανάμεσα στα σχήματα των δύο κόμβων P_1 και P_2 που παρουσιάστηκαν στην

Ενότητα 4.2.1. Το σχήμα κάθε κόμβου αποτελείται από μία μόνο σχέση, ενώ έχουν δοθεί οι σημασιολογικές περιγραφές τους. Χρησιμοποιούμε τις σχέσεις που υπολογίζουν το recall και το precision ανάμεσα σε δύο κλάσεις. Ο P_2 περιέχει τρεις περιορισμούς οι οποίοι είναι πιο περιοριστικοί από τους αντίστοιχους του P_1 (στους δύο απ' τους τρεις δεν υπάρχει καν αντίστοιχος περιορισμός στον P_1). Για την ακρίβεια, υπάγονται και οι τρεις στην περίπτωση R1 ⊒ R2. Στην περίπτωση αυτή το recall είναι 0.5 και το precision 1. Δεδομένου ότι η κλάση Band της οντολογίας έχει έξι ιδιότητες, από τις οποίες οι τρεις δεν έχουν κανένα περιορισμό και οι άλλες τρεις είναι αυτές που μόλις αναφέρθηκαν το αποτέλεσμα του υπολογισμού είναι το ακόλουθο:

$$recall(P_1, P_2) = \frac{1+1+1+0.5+0.5+0.5}{6} = 0.75 \text{ και}$$

$$precision(P_1, P_2) = \frac{1+1+1+1+1+1}{6} = 1.$$

Κρίνοντας το αποτέλεσμα ποιοτικά, θα μπορούσαμε να πούμε ότι δεδομένου ότι ισχύει $C_{P_1} \sqsupseteq C_{P_2}$, αφού ο P_1 επιβάλλει λιγότερους περιορισμούς από τον P_2, θα πρέπει recall < 1 και precision = 1, κάτι το οποίο επιβεβαιώθηκε από τους υπολογισμούς μας.

4.2.5 Υποσύστημα Επανεγγραφής, Σύγκρισης και Απάντησης Ερωτημάτων

Μια βασική λειτουργία του υποσυστήματος αυτού είναι η επανεγγραφή των ερωτημάτων. Όταν ένα ερώτημα Q_o προωθείται από τον κόμβο P_1 στον P_2, επανεγγράφεται σύμφωνα με τις αντιστοιχίσεις στο ερώτημα Q_r.

Παρόλα αυτά, κατά τη διαδικασία της επανεγγραφής μπορεί κάποια χαρακτηριστικά να μην επανεγγραφούν, ενώ κάποιες συνθήκες μπορεί να χαθούν ή να προστεθούν, εξαιτίας της φύσης των συγκεκριμένων αντιστοιχίσεων. Ως αποτέλεσμα, η ανακτηθείσα πληροφορία μπορεί να μην εναρμονίζεται πλήρως στην αρχική αίτηση. Έτσι, στην ανάλυση μας για την ομοιότητα των κόμβων πρέπει να συμπεριλάβουμε τους εξής δύο επιπλέον παράγοντες:

- το ποσοστό των χαρακτηριστικών που επανεγγράφηκαν και πόσο ακριβής ήταν η επανεγγραφή τους.
- τις συνθήκες που καθορίζονται τόσο στην αρχική όσο και στην επανεγγραμμένη έκδοση του ερωτήματος.

Οι συνθήκες που υπάρχουν στο ερώτημα, περιορίζουν περαιτέρω την πληροφορία που ζητείται ή παρέχεται από τον κόμβο. Έτσι, αυτές οι συνθήκες πρέπει να ληφθούν υπόψη, μαζί με τους περιορισμούς που ήδη υπάρχουν στις κλάσεις που περιγράφουν σημασιολογικά το

σχήμα του κόμβου. Για το σκοπό αυτό, κάθε συνθήκη στο ερώτημα μεταφράζεται σε ένα αντίστοιχο περιορισμό τιμής, ο οποίος προστίθεται στην κατάλληλη κλάση στην επισημείωση του κόμβου. Η διαδικασία που ακολουθείται αποτελείται από τα ακόλουθα βήματα:

1. Το ερώτημα Q_o τίθεται στον κόμβο P_1.
2. Βασιζόμενος στις σημασιολογικές του επισημειώσεις, ο P_1 επιλέγει το σύνολο των κλάσεων C_{Q_o}, το οποίο περιέχει τις κλάσεις που επισημειώνουν τις σχέσεις που εμφανίζονται στο Q_o.
3. Κάθε κλάση στο C_{Q_o} ενισχύεται με επιπλέον περιορισμούς τιμών στις ιδιότητές του, σύμφωνα με τις συνθήκες που καθορίζονται στο Q_o και προκύπτει το σύνολο κλάσεων $C_{Q_o,e}$.
4. Το Q_o στέλνεται στον κόμβο P_2, όπου επανεγγράφεται σύμφωνα με τις αντιστοιχίσεις και προκύπτει το ερώτημα Q_r.
5. Τα βήματα 2 και 3 επαναλαμβάνονται για το Q_r και προκύπτει το σύνολο των κλάσεων $C_{Q_r,e}$.

Στη συνέχεια, το πρώτο βήμα για να αποτιμηθεί η ποιότητα της επανεγγραφής είναι να υπολογιστεί το recall και το precision ανάμεσα στα σύνολα των κλάσεων $C_{Q_o,e}$ και $C_{Q_r,e}$. Αυτό πραγματοποιείται από το Υποσύστημα Υπολογισμού του Recall και του Precision.

Ο δεύτερος παράγοντας που πρέπει να ληφθεί υπόψη είναι η επανεγγραφή των χαρακτηριστικών που εμφανίζονται στο SELECT τμήμα του ερωτήματος. Αν ένα χαρακτηριστικό t επανεγγραφεί σε ένα χαρακτηριστικό t', τότε η ποιότητα της επανεγγραφής μετράται από την τιμή του recall και του precision μεταξύ των αντίστοιχων ιδιοτήτων p_t και p_t' που επισημειώνουν τα χαρακτηριστικά αυτά. Αυτό υπολογίζεται και πάλι από το Υποσύστημα Υπολογισμού του Recall και του Precision. Στη συνέχεια, υπολογίζεται το άθροισμα για όλα τα χαρακτηριστικά του ερωτήματος και κανονικοποιείται στον αριθμό των χαρακτηριστικών του ερωτήματος. Αν ένα χαρακτηριστικό αποτύχει να επανεγγραφεί, η τιμή του recall για αυτό είναι μηδενική, καθώς η αντίστοιχη πληροφορία δεν μπορεί να ανακτηθεί. Η τιμή του precision δεν επηρεάζεται, αφού δεν προκύπτει κάποιος πλεονασμός στα αποτελέσματα.

Τα αποτελέσματα από τους δύο παράγοντες πολλαπλασιάζονται, καθώς και οι δύο συμβάλλουν αρνητικά στην ποιότητα της πραγματοποιούμενης επανεγγραφής. Τα παραπάνω συνοψίζονται στους ακόλουθους τύπους:

$$recall\left(C_{Q_o,e}, C_{Q_r,e}, Q_o, Q_r\right) = \frac{\sum recall\left(p_t, p_t'\right)}{\left|SELECT\left(Q_o\right)\right|} \cdot recall\left(C_{Q_o,e}, C_{Q_r,e}\right)$$

$$precision\left(C_{Q_o,e}, C_{Q_r,e}, Q_o, Q_r\right) = \frac{\sum precision\left(p_t, p_t'\right)}{\left|SELECT\left(Q_r\right)\right|} \cdot precision\left(C_{Q_o,e}, C_{Q_r,e}\right)$$

Μια τελευταία λειτουργία του υποσυστήματος είναι η απάντηση του ερωτήματος από τον P_2. Αν ο P_1 αποφασίσει ότι η πληροφορία που είναι διαθέσιμη από τον P_2 για το δεδομένο ερώτημα του είναι χρήσιμη, τον ειδοποιεί να απαντήσει το ερώτημα.

Το διάγραμμα ροής δεδομένων του υποσυστήματος δίνεται στην Εικόνα 4.8.

Σενάριο Λειτουργίας (συνέχεια)

Θα χρησιμοποιήσουμε και πάλι τους κόμβους P_1 και P_2 που παρουσιάστηκαν στην Ενότητα 4.2.1, για να δείξουμε δύο χαρακτηριστικές περιπτώσεις, ως συνέχεια του παραδείγματος που δόθηκε στην Ενότητα 4.2.4.

Περίπτωση 1. Έστω ότι ο P_1 ενδιαφέρεται για τις μπάντες που σχηματίστηκαν τη δεκαετία του ’80 κι έτσι θέτει το ερώτημα:

Q_o: SELECT name, members, year FROM bands

WHERE year >= 1980 AND year <=1990

Βασιζόμενοι στο ερώτημα αυτό και τις συνθήκες του, ο σημασιολογικός ορισμός του P_1 γίνεται:

$C_{Q_o,e}$: P1_Band ≡ Band ⊓ ≥1 released ⊓ ∀year.([1980, 1990])

Στη συνέχεια το ερώτημα προωθείται στον κόμβο P_2, όπου επανεγγράφεται στο ακόλουθο ερώτημα:

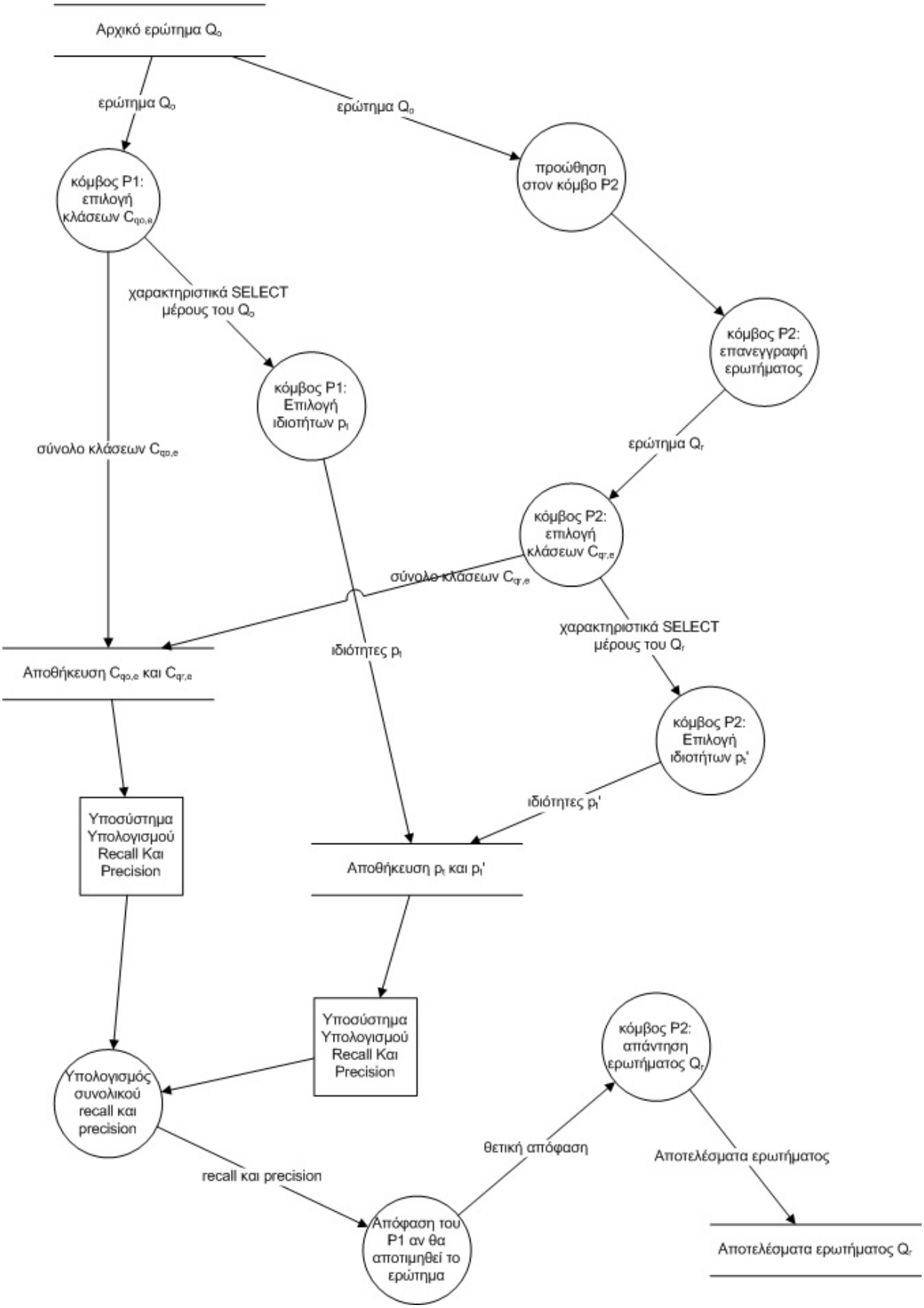
Q_r: SELECT name, singer, year FROM bands

WHERE year >= 1980 AND year <=1990

Να σημειωθεί ότι το χαρακτηριστικό members, το οποίο αντιστοιχίζεται με την ιδιότητα hasMember, έχει επανεγγραφεί ως singer, που αντιστοιχίζεται με την ιδιότητα hasSinger, ενώ ισχύει hasSinger ⊑ hasMember.

Ο σημασιολογικός ορισμός του P_2 γίνεται:

$C_{Q_r,e}$: P2_Band ≡ Band ⊓ ∀type.Jazz ⊓ ≥3 released ⊓ ∀year.([1980, 1990])



Εικόνα 4.8 Διάγραμμα Ροής Δεδομένων Υποσυστήματος Επανεγγραφής, Σύγκρισης και Απάντησης Ερωτημάτων

Επειδή ο περιορισμός στην ιδιότητα year που επιβάλλεται από τις συνθήκες του ερωτήματος είναι πιο περιοριστικός από τον ήδη υπάρχοντα περιορισμό στο year (∀year.(≤2000)), υπερκαλύπτει τον παλιό περιορισμό.

Εφαρμόζοντας τις σχέσεις υπολογισμού που παρουσιάστηκαν παραπάνω για το παρόν υποσύστημα, προκύπτει:

$$recall(P_1, P_2, Q_o, Q_r) = \frac{1+1+0.5}{3} \cdot \frac{1+1+1+1+0.5+0.5}{6} = 0.7 \text{ και}$$

$$precision(P_1, P_2, Q_o, Q_r) = 1.$$

Παρατηρούμε ότι η τιμή του recall επηρεάστηκε αρνητικά από την επανεγγραφή του members σε singer, και θετικά από την ύπαρξη της συνθήκης του ερωτήματος στο χαρακτηριστικό year, η οποία αλληλεπιδρά με τον αντίστοιχο περιορισμό στη σημασιολογική επισημείωση του P_2.

Περίπτωση 2. Έστω ότι το χαρακτηριστικό year δεν υπήρχε στο σχήμα του P_2 ή ότι δεν υπήρχε η αντιστοίχιση μεταξύ του χαρακτηριστικών year του P_1 και του year του P_2. Τότε, η επανεγγραμμένη έκδοση του ερωτήματος θα ήταν:

Q_r: SELECT name, singer FROM bands

Καθώς δεν υπάρχουν συνθήκες στο Q_r, ο σημασιολογικός ορισμός του P_2 δεν επηρεάζεται. Έτσι οι υπολογισμοί δίνουν:

$$recall(P_1, P_2, Q_o, Q_r) = \frac{1+0.5+0}{3} \cdot \frac{1+1+1+1+0.5+0.5}{6} = 0.42 \text{ και}$$

$$precision(P_1, P_2, Q_o, Q_r) = 1 \cdot \frac{1+1+1+1+1+0.5}{6} = 0.92.$$

Παρατηρούμε ότι η αδυναμία επανεγγραφής του χαρακτηριστικού year μειώνει σημαντικά την τιμή του recall. Παράλληλα, εξαιτίας αυτής της αδυναμίας, τα bands που επιστρέφονται από το ερώτημα έχουν τον περιορισμό year≤2000 και όχι το year∈[1980,1990], το οποίο επηρεάζει αρνητικά την τιμή του precision.

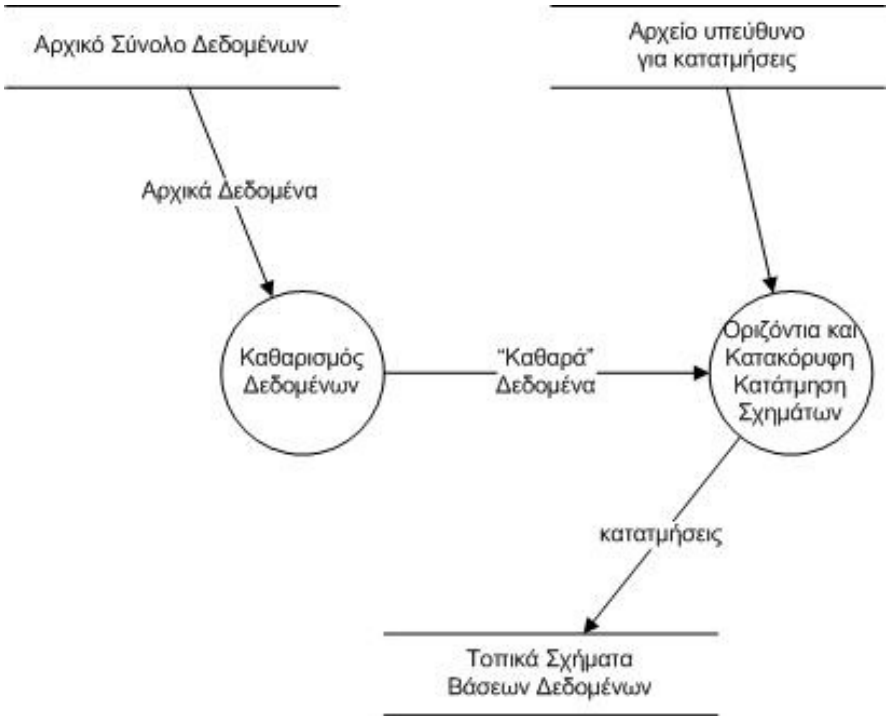
4.2.6 Υποσύστημα Κατανομής Συνόλου Δεδομένων

Ένα από τα βασικότερα στοιχεία ενός δικτύου ομότιμων κόμβων είναι η κατανομή των δεδομένων στο δίκτυο. Στην εργασία μας, θεωρούμε την ύπαρξη ενός αδόμητου επικαλύπτοντος δικτύου, που σημαίνει ότι τα δεδομένα είναι κατανεμημένα με τυχαίο τρόπο

στους κόμβους και όχι με βάση κάποια συνάρτηση κατακερματισμού, όπως συμβαίνει στα δομημένα δίκτυα. Το παρόν υποσύστημα είναι υπεύθυνο για την κατανομή ενός αρχικού συνόλου δεδομένων στους κόμβους του δικτύου.

Είσοδος του συστήματος είναι μια σχεσιακή βάση, η οποία περιέχει συγκεντρωμένη στις σχέσεις της το σύνολο της πληροφορίας του δικτύου. Με βάση ένα αρχείο το οποίο περιέχει την πιθανότητα ενός σχήματος της αρχικής βάσης να συμπεριληφθεί στο τοπικό σχήμα του κάθε κόμβου. Χρησιμοποιώντας το αρχείο αυτό, πραγματοποιούμε τόσες τυχαίες κατατμήσεις στη βάση όσοι είναι και οι κόμβοι του δικτύου. Καθεμιά από αυτές τις κατατμήσεις θα αποτελέσει το τοπικό σχήμα του ενός κόμβου. Έτσι οι κόμβοι αποκτούν ένα τυχαίο τμήμα της συνολικής πληροφορίας του δικτύου. Οι κατατμήσεις είναι και οριζόντιες και κατακόρυφες, ενώ υπάρχει προφανώς επικάλυψη μεταξύ τους, αφού, όπως συμβαίνει και σε ένα πραγματικό δίκτυο ομότιμων, έχουμε επανάληψη της πληροφορίας στο δίκτυο. Θα πρέπει τέλος να σημειωθεί ότι είναι πολύ πιθανό το αρχικό σύνολο δεδομένων να πρέπει να περάσει από ένα στάδιο καθαρισμού, ώστε τα δεδομένα να βρεθούν σε μορφή άμεσα εκμεταλλεύσιμη από το δίκτυο.

Το διάγραμμα ροής δεδομένων του υποσυστήματος δίνεται στην Εικόνα 4.9.



Εικόνα 4.9 Διάγραμμα Ροής Δεδομένων Υποσυστήματος Κατανομής Συνόλου Δεδομένων

5

Σχεδίαση Συστήματος

Το κεφάλαιο αυτό αναφέρεται στη σχεδίαση του συστήματος. Αρχικά (Ενότητα 5.1), δίνεται μια γενική περιγραφή της σχεδίασης, παρουσιάζοντας τα πακέτα από τα οποία αποτελείται η εφαρμογή, ενώ στη συνέχεια (Ενότητα 5.2) περιγράφονται αναλυτικότερα οι κλάσεις από τις οποίες αποτελούνται τα πακέτα, δίνοντας συνοπτικά τη βασική λειτουργία που επιτελεί κάθε κλάση, καθώς και τις μεθόδους από τις οποίες απαρτίζεται

5.1 Αρχιτεκτονική

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η αντικειμενοστρεφής γλώσσα προγραμματισμού Java. Ο κώδικας της εφαρμογής οργανώθηκε σε πακέτα τα οποία αποτελούνται από έναν αριθμό κλάσεων. Στην ενότητα αυτή γίνεται μια σύντομη περιγραφή των πακέτων της εφαρμογής.

- `package thesis.recallprecision`: Στο πακέτο αυτό περιλαμβάνονται οι κλάσεις
    - `RecallAndPrecision`
    - `ResourceRecallPrecision`
    - `ClassRecallPrecision`
    - `ClassListRecallPrecision`

- PropertyRecallPrecision
- RestrictionRecallPrecision
- RestrictionListRecPrec
- RecallPrecisionAndResultSet

Οι παραπάνω κλάσεις είναι υπεύθυνες για την υλοποίηση των αλγορίθμων που σχετίζονται με τον υπολογισμό του recall και του precision ανάμεσα σε διάφορους σημασιολογικούς πόρους (κλάσεις, ιδιότητες, περιορισμούς, λίστες κλάσεων, λίστες περιορισμών). Οι αλγόριθμοι αυτοί θα μελετηθούν αναλυτικά στην Ενότητα 6.1.

- `package thesis.semanticpeer:` Στο πακέτο αυτό περιλαμβάνονται οι κλάσεις
  - SemanticPeer
  - SemAnnotation
  - OverlayNetwork
  - TestSemanticPeerWithSchemas
  - TestSemanticPeerWithQuery

Οι παραπάνω κλάσεις είναι υπεύθυνες για τη μοντελοποίηση ενός σημασιολογικού κόμβου του δικτύου (αρχικοποίησή του και διάφορες λειτουργίες του), τη δημιουργία της τοπολογίας του επικαλύπτοντος δικτύου και την εξομοίωση των δύο βασικών λειτουργιών του κόμβου.

- `package thesis.query:` Στο πακέτο αυτό περιλαμβάνονται οι κλάσεις
  - RenderSQLQuery
  - RewritingResult
  - SemanticSQLQuery

Οι παραπάνω κλάσεις σχετίζονται με τις διάφορες λειτουργίες που επιτελούνται σε ένα ερώτημα που υποβάλλεται σε ένα κόμβο του δικτύου.

- `package thesis.mySQLSchema:` Στο πακέτο αυτό περιλαμβάνονται οι κλάσεις
  - DeleteViews
  - DisplayResultSet
  - ExecuteQueryInCommandLine
  - ExtractSchema

Οι κλάσεις του πακέτου αυτού είναι υπεύθυνες για διάφορες λειτουργίες που επιτελούνται στα τοπικά σχήματα των βάσεων δεδομένων των κόμβων.

- `package thesis.jmdb:` Στο πακέτο αυτό περιλαμβάνονται οι κλάσεις
    - `MakePartitions`
    - `PrepareJMDBTest`
    - `SemanticPeerForJMDB`
    - `TestSemanticPeerWithSchemas`
    - `TestSemanticPeerWithQuery`

  Οι ανωτέρω κλάσεις σχετίζονται με το βασικό σύνολο δεδομένων που χρησιμοποιήθηκε για τη μελέτη της λειτουργίας του συστήματος σε συνθήκες που πλησιάζουν πραγματικές συνθήκες λειτουργίας ενός δικτύου ομότιμων κόμβων. Κρίνεται σκόπιμη η αναφορά τους εδώ, μιας και μπορούν να χρησιμοποιηθούν είτε απευθείας είτε με ελάχιστες αλλαγές για τη διαχείριση οποιουδήποτε συνόλου δεδομένων.

- `package thesis:` Στο πακέτο αυτό περιλαμβάνονται οι κλάσεις
    - `RecPrecOptions`

  Η παραπάνω κλάση είναι υπεύθυνη για την παραμετροποίηση των διαφόρων επιλογών του συστήματος.

- `package thesis.oldtests` και `package thesis.finaltests:` Τα πακέτα αυτά περιλαμβάνουν κλάσεις οι οποίες χρησιμοποιήθηκαν κατά την ανάπτυξη του συστήματος για τον έλεγχο των επιμέρους λειτουργιών του. Το είδος του ελέγχου το οποίο εξυπηρέτησαν είναι ο λεγόμενος έλεγχος μαύρου κουτιού (black box testing), κατά τον οποίο δίνουμε σε ένα πρόγραμμα διάφορες κατάλληλα επιλεγμένες εισόδους και ελέγχουμε αν η έξοδος είναι η αναμενόμενη. Λόγω του σκοπού των κλάσεων αυτών, δεν κρίνεται σκόπιμη η περαιτέρω ανάλυση των πακέτων αυτών. Για το λόγο αυτό παραλείπονται και από τις περισσότερες αναλύσεις της σχεδίασης της εφαρμογής που ακολουθούν.

5.2 Περιγραφή Κλάσεων

Στην ενότητα αυτή περιγράφονται συνοπτικά οι λειτουργίες, τα πεδία και οι μέθοδοι των κλάσεων. Να σημειωθεί ότι δε γίνεται αναφορά στους κατασκευαστές (constructors) των

κλάσεων ούτε στις μεθόδους που χρησιμοποιούνται τυπικά στον αντικειμενοστρεφή προγραμματισμό για την προσπέλαση και αλλαγή των πεδίων των κλάσεων (γνωστών και ως getters και setters).

5.2.1 recallprecision.RecallAndPrecision

Η κλάση αυτή αναπαριστά ένα ζεύγος τιμών recall και precision και περιέχει ορισμένες μεθόδους για το χειρισμό τους

Πεδία

- `private double recall`

  Δεκαδικός αριθμός που αναπαριστά την τιμή του recall.
- `private double precision`

  Δεκαδικός αριθμός που αναπαριστά την τιμή του precision.

Μέθοδοι

- `public double getFMeasure()`

  Χρησιμοποιεί τις τιμές του recall και του precision για να υπολογίσει την F-μετρική (σταθμισμένος αρμονικός μέσος του recall και του precision)

5.2.2 recallprecision.ResourceRecallPrecision

Η κλάση αυτή αποτελεί την υπερκλάση ενός συνόλου κλάσεων που χρησιμοποιούνται για τον υπολογισμό του recall και του precision μεταξύ διαφόρων σημασιολογικών πόρων. Επειδή δεν δημιουργούμε άμεσα αντικείμενα από την κλάση αυτή, είναι δηλωμένη ως αφηρημένη (abstract class).

Πεδία

- `public static final int NOT_AVAILABLE = -1`

  Μια σταθερά που χρησιμοποιείται κατά σύμβαση όταν θέλουμε να δείξουμε ότι η τιμή του recall και του precision δεν είναι διαθέσιμες. Για το λόγο αυτό έχει μια μη επιτρεπτή τιμή (-1), αφού το recall και το precision παίρνουν τιμές από το 0 ως το 1.

- `protected double recall`

  Δεκαδικός αριθμός που αναπαριστά την τιμή του recall.

- `protected double precision`

  Δεκαδικός αριθμός που αναπαριστά την τιμή του precision.

Μέθοδοι

- `public void makeRecPrecNotAvailable()`

  Δίνει την τιμή -1 στο recall και το precision (για να δηλώσει ότι δεν είναι διαθέσιμα).

5.2.3 recallprecision.ClassRecallPrecision

Πρόκειται για την υπεύθυνη κλάση για τον υπολογισμό του recall και του precision ανάμεσα σε δύο κλάσεις. Είναι υποκλάση της κλάσης ResourceRecallPrecision (Ενότητα 5.2.2).

Πεδία

- `public static final int ONLY_RECALL = 1`
  `public static final int ONLY_PRECISION = 2`
  `public static final int RECALL_AND_PRECISION = 3`

  Οι τρεις αυτές σταθερές χρησιμοποιούνται για να δηλώσουν αν επιθυμούμε να αποτιμήσουμε μόνο την τιμή του recall (*`ONLY_RECALL`*), μόνο την τιμή του precision (*`ONLY_PRECISION`*) ή και των δύο (*`RECALL_AND_PRECISION`*).
- `private int recallOrPrecision`

  Ακέραιος που παίρνει την τιμή μιας από τις τρεις σταθερές που περιγράφηκαν παραπάνω.
- `private OntClass ontClass1`

  Η πρώτη από τις κλάσεις προς σύγκριση.
- `private OntClass ontClass2`

  Η δεύτερη από τις κλάσεις προς σύγκριση.

Μέθοδοι

- `public void calcClassRecPrec()`

  Υπολογίζει το recall και το precision για δύο οποιεσδήποτε κλάσεις, οι οποίες είναι αποθηκευμένες στα πεδία `ontClass1` και `ontClass2`. Είναι η γενική μέθοδος για τον υπολογισμό των μετρικών, η οποία καλεί με τη σειρά της μια άλλη κατάλληλη μέθοδο, ανάλογα με το είδος των κλάσεων που συγκρίνονται.

- `public void calcRawClassRecPrec()`

  Υπολογίζει το recall και το precision για δύο κλάσεις οι οποίες δεν περιγράφονται από καμία ιδιότητα.

- `public void calcClassPropsRecPrec`
  `(LinkedList<OntProperty> listProps1,`
  `LinkedList<OntProperty> listProps2)`

  Υπολογίζει το recall και το precision για δύο κλάσεις οι οποίες περιγράφονται από ορισμένες ιδιότητες, Οι ιδιότητες αυτές δίνονται σε μορφή συνδεδεμένης λίστας για καθεμιά από τις κλάσεις.

- `public void calcPropsRecall(LinkedList<OntProperty> listProps1,`
  `LinkedList<OntProperty> listProps2)`

  Υπολογίζει το recall για δύο κλάσεις οι οποίες περιγράφονται από ορισμένες ιδιότητες. Οι ιδιότητες αυτές δίνονται σε μορφή συνδεδεμένης λίστας για καθεμιά από τις κλάσεις.

- `public void calcPropsPrecision`
  `(LinkedList<OntProperty> listProps1,`
  `LinkedList<OntProperty> listProps2)`

  Υπολογίζει το precision για δύο κλάσεις οι οποίες περιγράφονται από ορισμένες ιδιότητες. Οι ιδιότητες αυτές δίνονται σε μορφή συνδεδεμένης λίστας για καθεμιά από τις κλάσεις.

- `public double calcClassSinglePropsRecall`
  `(LinkedList<OntProperty> listProps1,`
  `OntProperty ontProp2)`

  Η μέθοδος αυτή δέχεται ως παραμέτρους μια ιδιότητα και μια λίστα από ιδιότητες. Υπολογίζει το recall ανάμεσα στην ιδιότητα και καθεμιά από τις ιδιότητες της λίστας και επιστρέφει τη μέγιστη τιμή του recall.

- `public double calcClassSinglePropsPrecision`
  `(OntProperty ontProp1, LinkedList<OntProperty> listProps2)`

  Η μέθοδος αυτή δέχεται ως παραμέτρους μια ιδιότητα και μια λίστα από ιδιότητες. Υπολογίζει το precision ανάμεσα στην ιδιότητα και καθεμιά από τις ιδιότητες της λίστας και επιστρέφει τη μέγιστη τιμή του precision.

- `private void showSuperClasses(String className,`
  `HashSet<OntClass> superClasses)`

Παίρνει το όνομα μιας κλάσης και τη λίστα με τις υπερκλάσεις της, και τις τυπώνει στην οθόνη (στο standard output γενικότερα).

- `private void showRecallAndPrecision()`

  Εμφανίζει στην οθόνη την τιμή του recall, του precision ή και των δύο.

5.2.4 recallprecision.ClassListRecallPrecision

Πρόκειται για την υπεύθυνη κλάση για τον υπολογισμό του recall και του precision ανάμεσα σε δύο λίστες κλάσεων. Είναι υποκλάση της κλάσης ResourceRecallPrecision (Ενότητα 5.2.2).

Πεδία

- `private LinkedList<OntClass> classListP1`

  Η πρώτη από τις λίστες κλάσεων προς σύγκριση.

- `private LinkedList<OntClass> classListP2`

  Η δεύτερη από τις λίστες κλάσεων προς σύγκριση.

Μέθοδοι

- `public void calcClassListRecPrec()`

  Υπολογίζει το recall και το precision ανάμεσα στις δύο λίστες κλάσεων που είναι αποθηκευμένες στα πεδία `classListP1` και `classListP2`.

- `private void calcClassListRecall()`

  Υπολογίζει το recall ανάμεσα στις δύο λίστες κλάσεων.

- `private void calcClassListPrecision()`

  Υπολογίζει το precision ανάμεσα στις δύο λίστες κλάσεων.

- `private double calcSingleClassListRecall(OntClass tempClass1)`

  Η μέθοδος αυτή δέχεται ως παράμετρο μια κλάση. Υπολογίζει το recall ανάμεσα στην κλάση αυτή και καθεμιά από τις κλάσεις της λίστας `classListP2` και επιστρέφει τη μεγαλύτερη τιμή του recall. Η κλάση της λίστας που δίνει το μεγαλύτερο recall είναι η αντίστοιχη κλάση σε αυτήν που η μέθοδος δέχεται ως παράμετρο.

- `private double calcSingleClassListPrecision`
  `(OntClass tempClass2)`

Η μέθοδος αυτή δέχεται ως παράμετρο μια κλάση. Υπολογίζει το precision ανάμεσα στην κλάση αυτή και καθεμιά από τις κλάσεις της λίστας `classListP1` και επιστρέφει τη μεγαλύτερη τιμή του precision. Η κλάση της λίστας που δίνει το μεγαλύτερο precision είναι η αντίστοιχη κλάση σε αυτήν που η μέθοδος δέχεται ως παράμετρο.

5.2.5 recallprecision.PropertyRecallPrecision

Είναι η υπεύθυνη κλάση για τον υπολογισμό του recall και του precision ανάμεσα σε δύο ιδιότητες. Είναι υποκλάση της κλάσης ResourceRecallPrecision (Ενότητα 5.2.2).

Πεδία

- `public static final int ONLY_RECALL = 1`
  `public static final int ONLY_PRECISION = 2`
  `public static final int RECALL_AND_PRECISION = 3`
  `private int recallOrPrecision`
  Επιτελούν την ίδια λειτουργία με τα πεδία της κλάσης `ClassRecallPrecision` (Ενότητα 5.2.3).
- `private OntProperty ontProp1`
  Η πρώτη από τις ιδιότητες προς σύγκριση.
- `private OntProperty ontProp2`
  Η δεύτερη από τις ιδιότητες προς σύγκριση.

Μέθοδοι

- `public void calcPropertyRecPrec(boolean withRestrictions)`
  Υπολογίζει το recall και το precision ανάμεσα στις δύο ιδιότητες που είναι αποθηκευμένες στα πεδία `ontProp1` και `ontProp2`. Δέχεται μια boolean μεταβλητή ως παράμετρο, η οποία καθορίζει αν θα ληφθούν υπόψη στον υπολογισμό οι περιορισμοί που είναι ορισμένοι πάνω στις ιδιότητες που συγκρίνονται.
- `public void calcPropertyRecPrec()`
  Καλεί την `calcPropertyRecPrec` με όρισμα true, δηλαδή υπολογίζει το recall και το precision λαμβάνοντας υπόψη και τους περιορισμούς.
- `private void showSuperProperties(String propName,`
  `HashSet<OntProperty> superProps)`

Παίρνει το όνομα μιας ιδιότητας και τη λίστα με τις υπερκλάσεις της, και τις τυπώνει στην οθόνη.

- `private void updateRecPrecWithRestrictions (ExtendedIterator iterRestr1, ExtendedIterator iterRestr2)`

  Η μέθοδος αυτή δέχεται ως παράμετρο μια κλάση. Υπολογίζει το recall ανάμεσα στην κλάση αυτή και καθεμιά από τις κλάσεις της λίστας `classListP2` και επιστρέφει τη μεγαλύτερη τιμή του recall. Η κλάση της λίστας που δίνει το μεγαλύτερο recall είναι η αντίστοιχη κλάση σε αυτήν που η μέθοδος δέχεται ως παράμετρο.

5.2.6 recallprecision.RestrictionRecallPrecision

Είναι η υπεύθυνη κλάση για τον υπολογισμό του recall και του precision ανάμεσα σε δύο περιορισμούς. Είναι υποκλάση της κλάσης ResourceRecallPrecision (Ενότητα 5.2.2). Περιέχει διάφορες μεταβλητές (σταθερές και μη) οι οποίες έχουν καθαρά αλγοριθμικό ενδιαφέρον και δεν αναλύονται περαιτέρω εδώ.

Πεδία

- `private Restriction restriction1`

  Ο πρώτος από τους δύο περιορισμούς που θα συγκριθούν.

- `private Restriction restriction2`

  Ο δεύτερος από τους δύο περιορισμούς που θα συγκριθούν.

- `private OntProperty onProperty1`

  Η ιδιότητα πάνω στην οποία έχει τεθεί ο περιορισμός `restriction1`.

- `private OntProperty onProperty2`

  Η ιδιότητα πάνω στην οποία έχει τεθεί ο περιορισμός `restriction2`.

Μέθοδοι

- `private void initializeParameters()`

  Προσπελαύνει το αρχείο παραμετροποίησης του συστήματος και αρχικοποιεί ορισμένα πεδία που είναι απαραίτητα για τον υπολογισμό του recall και του precision.

- `public void calcRestrictionRecPrec()`

  Υπολογίζει το recall και το precision ανάμεσα σε δύο περιορισμούς.

- `private void calcValueObjRestrRecPrec()`

  Υπολογίζει το recall και το precision ανάμεσα σε δύο περιορισμούς τιμών σε ιδιότητες αντικειμένων.

- `private void calcValueDatRestrRecPrec()`

  Υπολογίζει το recall και το precision ανάμεσα σε δύο περιορισμούς τιμών σε ιδιότητες τύπων δεδομένων.

- `private void calcCardinRestrRecPrec()`

  Υπολογίζει το recall και το precision ανάμεσα σε δύο περιορισμούς πληθυκότητας.

- `private boolean isNullNotObjProperty()`

  Επιστρέφει true αν ο ένας από τους δύο περιορισμούς είναι κενός (null) και ο άλλος δεν είναι περιορισμός τιμής σε ιδιότητα αντικειμένου.

- `private boolean isSomeCardinalityRestriction`
  `(Restriction restriction)`

  Επιστρέφει true αν ο περιορισμός που δέχεται ως παράμετρο είναι περιορισμός πληθυκότητας.

- `private boolean isSameCardinalityRestriction()`

  Επιστρέφει true αν και οι δύο περιορισμοί είναι περιορισμοί πληθικότητας ίδιου τύπου ή αν ο ένας είναι περιορισμός μέγιστης πληθικότητας και ο άλλος ελάχιστης.

- `public static void addIndivsFromEnumClass`
  `(EnumeratedClass enumClass, HashSet<Individual> indivSet)`

  Δέχεται ως παραμέτρους μια κλάση απαρίθμησης (enumerated class) και ένα σύνολο ατόμων, στο οποίο εισάγει τα άτομα που ανήκουν στην κλάση απαρίθμησης.

- `public double[] renderHasValueLiterals(Literal literal1,`
  `Literal literal2, boolean withRange)`

  Δέχεται τις λεκτικές τιμές δύο περιορισμών τιμής (hasValue restrictions) και καθορίζει τη σχέση μεταξύ των περιορισμών (υπαγωγής, ισότητας, επικάλυψης).

- `public static int[] renderStrLiteral(String strLiteral)`

  Δέχεται ένα αλφαριθμητικό της μορφής "min int", "max int" ή "[int, int]" και επιστρέφει έναν πίνακα με δύο ακεραίους που είναι τα δύο άκρα του αντίστοιχου διαστήματος.

- `private void setRecPrecFromState(int subsumState)`

  Δέχεται ένα αλφαριθμητικό που αντιστοιχεί στη σχέση υπαγωγής μεταξύ των περιορισμών και επιστρέφει την κατάλληλη τιμή του recall και του precision.

- `public static EnumeratedClass isEnumClass(OntClass ontClass)`

  Δέχεται μια κλάση ως παράμετρο και ελέγχει αν είναι κλάση απαρίθμησης.

- `private double calcNumDiff(int num1, int num2)`

  Δέχεται ως παραμέτρους δύο αριθμούς και επιστρέφει έναν αριθμό από το 0 ως το 1 που δείχνει πόσο κοντά είναι δύο αριθμοί.

- `private void updateRecPrecWithRange(double factor)`

  Ανανεώνει τις τιμές των recall και precision δίνοντας μεγαλύτερη ποικιλία στις τιμές τους (και όχι μόνο τις κβαντισμένες τιμές 0, 0.5 και 1).

- `private void makeRecPrecAvailable()`

  Δίνει αρχικές τιμές στο recall και το precision, θέτοντάς τα ίσα με 1.

5.2.7 recallprecision.RestrictionListRecPrec

Πρόκειται για την υπεύθυνη κλάση για τον υπολογισμό του recall και του precision ανάμεσα σε δύο λίστες περιορισμών. Είναι υποκλάση της κλάσης ResourceRecallPrecision (Ενότητα 5.2.2).

Πεδία

- `private LinkedList<Restriction> restrictionListP1`

  Η πρώτη από τις δύο λίστες των περιορισμών που θα συγκριθούν.

- `private LinkedList<Restriction> restrictionListP2`

  Η δεύτερη από τις δύο λίστες των περιορισμών που θα συγκριθούν.

- `private OntProperty onProperty1`

  Η ιδιότητα πάνω στην οποία έχει τεθεί η λίστα των περιορισμών `restrictionListP1`.

- `private OntProperty onProperty2`

  Η ιδιότητα πάνω στην οποία έχει τεθεί η λίστα των περιορισμών `restrictionListP2`.

Μέθοδοι

- `public double calcRecallRestrFactor()`

  Υπολογίζει το γινόμενο των τιμών του recall ανάμεσα στους περιορισμούς της `onProperty2` και των αντίστοιχων περιορισμών της `onProperty1`.

- `public double calcPrecisionRestrFactor()`

  Υπολογίζει το γινόμενο των τιμών του precision ανάμεσα στους περιορισμούς της `onProperty1` και των αντίστοιχων περιορισμών της `onProperty2`.

- `private double calcSingleRecallRestr(Restriction restr2)`

  Παίρνει ένα περιορισμό και υπολογίζει το recall ανάμεσα σε αυτό τον περιορισμό και κάθε ένα από τους περιορισμούς της `onProperty2`. Επιστρέφει τη μεγαλύτερη τιμή του recall.

- `private double calcSinglePrecisionRestr(Restriction restr1)`

  Παίρνει ένα περιορισμό και υπολογίζει το precision ανάμεσα σε αυτό τον περιορισμό και κάθε ένα από τους περιορισμούς της `onProperty1`. Επιστρέφει τη μεγαλύτερη τιμή του precision.

5.2.8 recallprecision.RecallPrecisionAndResultSet

Αποτελεί επέκταση της κλάσης RecallAndPrecision (Ενότητα 5.2.1), περιέχοντας ένα επιπλέον πεδίο.

Πεδία

- `private ResultSet queryResultSet`

  Περιέχει το σύνολο των αποτελεσμάτων ενός ερωτήματος.

5.2.9 semanticpeer.SemanticPeer

Πρόκειται για την κλάση που μοντελοποιεί ένα σημασιολογικό κόμβο του συστήματος. Είναι πιθανώς η σημαντικότερη κλάσης όλης της εφαρμογής. Διατηρεί όλες τις ουσιώδεις πληροφορίες και δεδομένα που πρέπει να διαθέτει ένας κόμβος ώστε να είναι δυνατή η επισημείωσή του στην κεντρική οντολογία και να μπορεί να συγκριθεί σημασιολογικά με τους άλλους κόμβους του δικτύου.

Πεδία

- `private String peerID`

  Το όνομα – αναγνωριστικό του κόμβου.

- `private String dbSchemaName`

  Το όνομα του σχήματος της τοπικής βάσης δεδομένων του κόμβου.

- `private String[] dbSchema`

  Πίνακας που περιέχει τα ονόματα των ιδιοτήτων (attributes, τις αποκαλούμε “χαρακτηριστικά” για να τις ξεχωρίζουμε από τις σημασιολογικές ιδιότητες - properties) του τοπικού σχήματος της βάσης.

- `private OntModel domainOntModel`

  Το μοντέλο που αναπαριστά την κεντρική οντολογία.

- `private OntModel localOntModel`

  Η οντολογία επαυξημένη με τους τοπικούς περιορισμούς του κόμβου.

- `private String domainOntologyURL`

  Το URL της κεντρικής οντολογίας.

- `private String localOntologyURL`

  Το URL της τοπικής οντολογίας.

- `private HashMap< String, SemAnnotation > annotList`

  Η λίστα που περιέχει τις σημασιολογικές επισημειώσεις για κάθε σχέση του σχήματος της βάσης.

- `private HashMap< String, HashMap< String, String > > dbMapList`

  Μια λίστα που περιέχει τις αντιστοιχίσεις ανάμεσα στις σχέσεις της βάσης του κόμβου και τις σχέσεις των γειτονικών κόμβων του.

- `private LinkedList<OntClass> peerAnnotClassList`

  Λίστα που περιέχει το σύνολο των κλάσεων που επισημειώνουν σημασιολογικά το σχήμα του κόμβου.

- `private LinkedList<OntClass> neighbourAnnotClassList`

  Λίστα που περιέχει το σύνολο των κλάσεων που επισημειώνουν σημασιολογικά το σχήμα ενός από τους γείτονες του κόμβου.

- `private HashMap<String, RecallAndPrecision> neighboursRecPrecMap`

  Πίνακας κατακερματισμού που αποθηκεύει το recall και το precision για κάθε γείτονα του κόμβου.

- `private LinkedList<String> neighboursSortedList`

  Λίστα που περιέχει τα αναγνωριστικά των γειτόνων ταξινομημένα με βάση την F-μετρική κάθε γείτονα.

- `private String neighbourPeerID`

  Το αναγνωριστικό ενός γείτονα του κόμβου.

- `private String initialQuery`

  Το αρχικό ερώτημα Q_o που τίθεται στο δίκτυο από τον κόμβο.

- `private String rewrittenQuery`

  Το ερώτημα Q_r που προέκυψε ύστερα από την επανεγγραφή ενός ερωτήματος που εισήλθε στον κόμβο.

- `private OntProperty[] peerAnnotPropertyArr`

  Πίνακας που περιέχει τις ιδιότητες που επισημειώνουν τα χαρακτηριστικά που εμφανίζονται στο SELECT-τμήμα του ερωτήματος Q_o.

- `private OntProperty[] neighbourAnnotPropertyArr`

  Πίνακας που περιέχει τις ιδιότητες που επισημειώνουν τα χαρακτηριστικά που εμφανίζονται στο SELECT-τμήμα του επανεγγραμμένου ερωτήματος Q_r.

- `private HashMap<String, ResultSet> neighbourResultSetHash`

  Ένας πίνακας κατακερματισμού που περιέχει τα αποτελέσματα που προέκυψαν από την απάντηση των ερωτημάτων από ορισμένους κόμβους.

Μέθοδοι

- ```
  public void addInAnnotList(String tableName,
        SemAnnotation newAnnotation)
  ```

  Μέθοδος που προσθέτει μια νέα σημασιολογική επισημείωση στη λίστα των σημασιολογικών επισημειώσεων. Δέχεται ως παραμέτρους τη σημασιολογική αυτή επισημείωση και το όνομα της σχέσης που θα επισημειωθεί.

- ```
  public void addPeerMappings(String peerId,
        HashMap< String, String > peerMappings)
  ```

  Προσθέτει τις αντιστοιχίσεις των χαρακτηριστικών των σχέσεων ανάμεσα στον κόμβο και έναν από τους γείτονές του.

- `private void createDBSchema()`

  Επικοινωνεί με τη βάση δεδομένων για να ανακτήσει πληροφορίες για το τοπικό σχήμα.

- `private void loadOntologies()`

  Φορτώνει το μοντέλο της κεντρικής και της τοπικής οντολογίας.

- ```
  public RecallAndPrecision comparePeerSchemas
        (String neighbourID,
        LinkedList<OntClass> neighbAnnotClassList)
  ```
  ```

Ο κόμβος συγκρίνει το σχήμα του με το σχήμα ενός γειτονικού κόμβου, υπολογίζοντας το recall και το precision ανάμεσα στις κλάσεις που επισημειώνουν κάθε σχήμα.

- `public void compareAndSortNeighbourSchemas`
  `(SemanticPeerForJMDB[] semPeersArr,`
  `HashMap<String, LinkedList<String>> connectedPeers)`

  Δέχεται ένα πίνακα με κόμβους και μια αναπαράσταση του γράφου του επικαλύπτοντος δικτύου και υπολογίζει το recall και το precision ανάμεσα στον κόμβο και τους γείτονες του, τους οποίου και ταξινομεί με βάση την επιστρεφόμενη F-μετρική.

- `public RecallAndPrecision comparePeersWithQuery`
  `(String neighbourID, String initQuery,`
  `LinkedList<OntClass> neighbAnnotClassList,`
  `OntProperty[] neighAnnotPropertyArr)`

  Ένα ερώτημα τίθεται στον κόμβο από ένα γείτονά του, ο κόμβος επανεγγράφει το ερώτημα και συγκρίνει τους δύο κόμβους λαμβάνοντας υπόψη το συγκεκριμένο ερώτημα.

- `public void compareAndSortNeighboursWithQuery`
  `(String initialQuery, SemanticPeerForJMDB[] semPeersArr,`
  `HashMap<String, LinkedList<String>> connectedPeers)`

  Παίρνει ένα ερώτημα, έναν πίνακα από κόμβους και μια αναπαράσταση του γράφου του επικαλύπτοντος δικτύου και υπολογίζει το recall και το precision ανάμεσα στον κόμβο και τους γειτονικούς του, τους οποίους ταξινομεί με βάση την F-μετρική.

- `public ResultSet answerRewrittenQuery(String neighbourID,`
  `String initQuery,`
  `LinkedList<OntClass> neighbAnnotClassList,`
  `OntProperty[] neighAnnotPropertyArr)`

  Ένα ερώτημα τίθεται στον κόμβο από ένα γείτονά του, ο κόμβος το επανεγγράφει και επιστρέφει τα αποτελέσματα της αποτίμησης του ερωτήματος πίσω στο γείτονα.

- `public void preparePeerAnnotClassList()`

  Συμπληρώνει τη λίστα που περιέχει τις σημασιολογικές επισημειώσεις των κλάσεων με τις κλάσεις που επισημειώνουν τις σχέσεις του σχήματος της βάσης.

- `public boolean prepareInitialQuery(String posedQuery)`

Ανανεώνει τα `peerAnnotClassList` και `peerAnnotPropertyArr` για το συγκεκριμένο ερώτημα που δίνεται ως παράμετρος.

- `private boolean prepareAndRewriteQuery(String posedQuery, String peerIDFrom)`

  Επανεγγράφει το ερώτημα που έρχεται από ένα γείτονα και στη συνέχεια το αναλύει για να ανανεώσει τα `peerAnnotClassList` και `peerAnnotPropertyArr`.

- `private void prepareQuery(SemanticSQLQuery semanQuery, String[] tempAttrsArr)`

  Δέχεται ένα ερώτημα και υπολογίζει τις κλάσεις και ιδιότητες που επισημειώνουν τις σχέσεις και τα χαρακτηριστικά αντίστοιχα που εμφανίζονται στο ερώτημα.

- `public void sortNeighbours (HashMap<String,RecallAndPrecision>neighboursRecPrecHash)`

  Παίρνει ένα πίνακα κατακερματισμού που περιέχει τα αναγνωριστικά των γειτόνων και το recall και το precision τους και δημιουργεί μια λίστα που περιέχει τα αναγνωριστικά των κόμβων ταξινομημένα με βάση την F-μετρική κάθε κόμβου.

- `public static String getAttributeName(String attributeCName)`

  Παίρνει ένα αλφαριθμητικό της μορφής “σχέση.χαρακτηριστικό” και επιστρέφει το “χαρακτηριστικό”.

- `public static String getTableName(String attributeCName)`

  Παίρνει ένα αλφαριθμητικό της μορφής “σχέση.χαρακτηριστικό” και επιστρέφει τη “σχέση”.

5.2.10 semanticpeer.SemAnnotation

Η κλάση αυτή μοντελοποιεί τη σημασιολογική επισημείωση για μια σχέση της βάσης δεδομένων ενός κόμβου.

Πεδία

- `private LinkedList<String> ontClassSet`

  Λίστα που περιλαμβάνει το σύνολο των κλάσεων που χρησιμοποιούνται για τη σημασιολογική επισημείωση της σχέσης της βάσης.

- `private HashMap<String, String> attPropMappings`

  Πίνακας κατακερματισμού με ζεύγη της μορφής (R.t, C_i.p), όπου R η σχέση, R.t ένα χαρακτηριστικό της σχέσης, C_i μια από τις κλάσεις που χρησιμοποιούνται για τη

σημασιολογική επισημείωση και C_i.p η ιδιότητα της C_i στην οποία αντιστοιχίζεται το R.t.

Μέθοδοι

- `public boolean addOntClass(String oClass)`

  Προσθέτει μια κλάση στη λίστα `ontClassSet`.

- `public void addMapping(String attributeName, String propertyName)`

  Προσθέτει ένα ζεύγος της μορφής (R.t, C_i.p) στο `attPropMappings`.

5.2.11 semanticpeer.OverlayNetwork

Περιέχει μια μέθοδο υπεύθυνη για τη μοντελοποίηση της τοπολογίας του επικαλύπτοντος δικτύου.

Μέθοδοι

- `public static HashMap<String, LinkedList<String>> loadGraphEdgesFromFile(String fileName)`

  Διαβάζει ένα αρχείο το οποίο περιέχει τις ακμές του δικτύου που προκύπτει από μια τυχαία γεννήτρια γράφων και δημιουργεί ένα πίνακα κατακερματισμού, ο οποίος περιέχει για κάθε κόμβο μια λίστα με όλους τους κόμβους που είναι συνδεδεμένοι με αυτόν.

5.2.12 semanticpeer.TestSemanticPeerWithSchemas

Η κλάση αυτή προσομοιώνει την απλή περίπτωση λειτουργίας του δικτύου, κατά την οποία επιλέγεται ένας αρχικός κόμβος και ένας γείτονας του και υπολογίζεται η ομοιότητα μεταξύ των σχημάτων τους.

Πεδία

- `private static final String PEER1ID = "P1"`

  Αλφαριθμητική σταθερά που δηλώνει το αναγνωριστικό του αρχικού κόμβου.

- `private static final String PEER2ID = "P2"`

  Αλφαριθμητική σταθερά που δηλώνει το αναγνωριστικό του γειτονικού κόμβου με τον οποίο θα συγκριθεί σημασιολογικά ο αρχικός κόμβος.

Μέθοδοι

- `public static void main(String[] args)`

  Κατά τη μέθοδο αυτή, αρχικά γίνεται η σημασιολογική επισημείωση των δύο προς σύγκριση κόμβων και στη συνέχεια συγκρίνονται μεταξύ τους. Τα αποτελέσματα, καθώς και ο χρόνος διάρκειας, της σύγκρισης τυπώνονται στην οθόνη.

5.2.13 semanticpeer.TestSemanticPeerWithQuery

Η κλάση αυτή προσομοιώνει την απλή περίπτωση λειτουργίας του δικτύου, κατά την οποία επιλέγεται ένας αρχικός κόμβος, ένας γείτονας του και ένα αρχικό ερώτημα και υπολογίζεται η ομοιότητα μεταξύ των σχημάτων τους αναφορικά λαμβάνοντας υπόψη και το ερώτημα.

Πεδία

- `private static final String PEER1ID = "P1"`
  `private static final String PEER2ID = "P2"`

  Τα αναγνωριστικά των κόμβων που θα συγκριθούν με βάση το ερώτημα.
- `private static final String QUERY`

  Το αρχικό ερώτημα που τίθεται στον P1.

Μέθοδοι

- `public static void main(String[] args)`

  Κατά τη μέθοδο αυτή, το ερώτημα επανεγγράφεται από τον κόμβο P2 και στη συνέχεια γίνεται η σημασιολογική επισημείωση των δύο κόμβων επαυξημένη με τις συνθήκες του κάθε ερωτήματος. Τελικά οι κόμβοι συγκρίνονται μεταξύ τους με βάση το ερώτημα. Τα αποτελέσματα, καθώς και ο χρόνος διάρκειας, της σύγκρισης τυπώνονται στην οθόνη.

5.2.14 query.RenderSQLQuery

Η κλάση αυτή περιέχει στατικές μεθόδους για την επεξεργασία ενός ερωτήματος. Οι περισσότερες από τις μεθόδους της παίρνουν για όρισμα μια παράμετρο (`boolean print`) η οποία καθορίζει αν θα τυπώνονται ή όχι διαγνωστικά μηνύματα κατά τη διάρκεια της μεθόδου.

Μέθοδοι

- `public static String[] calcPartsArr(String query, boolean print)`

  Παίρνει ένα ερώτημα και το χωρίζει στα τρία μέρη του: SELECT-τμήμα, FROM-τμήμα, WHERE-τμήμα., τα οποία επιστρέφει σε ένα πίνακα.

- `public static String[] calcAttributesArr(String selectPart, boolean print)`

  Παίρνει το SELECT-τμήμα ενός ερωτήματος και επιστρέφει ένα πίνακα με τα χαρακτηριστικά στα οποία αναφέρεται.

- `public static String[] calcRestrictionsArr(String wherePart, boolean print)`

  Παίρνει το WHERE-τμήμα ενός ερωτήματος και επιστρέφει ένα πίνακα με τις συνθήκες τις οποίες αυτό επιβάλλει.

- `public static String[] calcRelationsArr(String fromPart, boolean print)`

  Παίρνει το FROM-τμήμα ενός ερωτήματος και επιστρέφει ένα πίνακα με τις σχέσεις στις οποίες αναφέρεται.

- `public static RewritingResult rewriteQuery(String initialQuery, HashMap< String, String > dbMappings)`

  Δέχεται ως παράμετρο ένα ερώτημα, το επανεγγράφει με βάση τις δοσμένες αντιστοιχίσεις και επιστρέφει την επανεγγραμμένη έκδοση του ερωτήματος.

- `public static String getPropFromAtt(String attName, String[] relationsArr, HashMap< String, SemAnnotation > annotList)`

  Παίρνει ένα χαρακτηριστικό ενός σχήματος και επιστρέφει την ιδιότητα της οντολογίας με την οποία έχει επισημειωθεί, χρησιμοποιώντας τη λίστα με τις σημασιολογικές επισημειώσεις.

- `private static void printArr(String[] array, String messageToUser)`

  Τυπώνει τα περιεχόμενα ενός πίνακα που δέχεται ως παράμετρο, μαζί με ένα σχετικό μήνυμα.

5.2.15 query.RewritingResult

Η κλάση αυτή αναπαριστά το αποτέλεσμα της επανεγγαφής ενός ερωτήματος.

Πεδία

- `private String rewrittenQuery`

  Η επανεγγραμμένη εκδοχή του ερωτήματος.
- `private String[] rewrittenAttrArr`

  Ένας πίνακας από αλφαριθμητικά που περιέχει τα χαρακτηριστικά που εμφανίζονται στο SELECT-τμήμα του επανεγγραμμένου ερωτήματος. Μια κενή (null) τιμή δείχνει ότι ένα χαρακτηριστικό στο αρχικό ερώτημα δεν είχε αντίστοιχο χαρακτηριστικό στο νέο κόμβο κι έτσι απέτυχε να επανεγγραφεί.

5.2.16 query.SemanticSQLQuery

Αυτή η κλάση είναι υπεύθυνη για τα ερωτήματα που τίθενται σε ένα κόμβο.

Πεδία

- `private String[] partsArr`

  Ένας πίνακας από αλφαριθμητικά που περιέχει τα τρία τμήματα ενός ερωτήματος.
- `private String[] attributesArr`

  Ένας πίνακας από αλφαριθμητικά που περιέχει τα χαρακτηριστικά του SELECT-τμήματος του ερωτήματος.
- `private String[] relationsArr`

  Ένας πίνακας από αλφαριθμητικά που περιέχει τις σχέσεις του FROM-τμήματος του ερωτήματος.
- `private String[] restrictionsArr`

  Ένας πίνακας από αλφαριθμητικά που περιέχει τις συνθήκες του WHERE-τμήματος του ερωτήματος.
- `private LinkedList<String> annotClassURIList`

  Μια συνδεδεμένη λίστα που περιλαμβάνει όλα τα URI των κλάσεων με τις οποίες είναι επισημειωμένες οι σχέσεις.
- `private LinkedList<OntClass> annotClassList`

  Μια συνδεδεμένη λίστα που περιλαμβάνει όλες τις κλάσεις με τις οποίες είναι επισημειωμένες οι σχέσεις.

Μέθοδοι

- `private void calcAnnotClassURIs`
  `(HashMap< String, SemAnnotation > annotationList)`

  Υπολογίζει τη λίστα που περιέχει όλα τα URI των κλάσεων με τις οποίες είναι επισημειωμένες σημασιολογικά οι σχέσεις που εμφανίζονται στο FROM-τμήμα του ερωτήματος, κάνοντας χρήση της λίστας με τις επισημειώσεις.

- `private void calcAnnotClasses(OntModel localModel)`

  Χρησιμοποιεί τη λίστα `annotClassURIList` για να δημιουργήσει μια λίστα με τις κλάσεις που έχουν τα συγκεκριμένα URI. Οι κλάσεις αυτές προέρχονται από την τοπική οντολογία η οποία είναι επαυξημένη με τους περιορισμούς που θέτει το ερώτημα

- `private void calcRestrictions(OntModel localModel,`
  `HashMap< String, SemAnnotation > annotationList)`

  Δημιουργεί ένα νέο μοντέλο οντολογίας βασισμένο στην τοπική οντολογία του κόμβου, προσθέτοντας τους επιπλέον περιορισμούς που επιβάλλουν οι συνθήκες του WHERE-τμήματος του ερωτήματος.

- `private void renderStringToRestriction(String strRestriction,`
  `OntModel localOntModel,`
  `HashMap< String, SemAnnotation > annotationList)`

  Παίρνει μια αλφαριθμητική αναπαράσταση μιας συνθήκης, η οποία προέρχεται από το WHERE-τμήμα του ερωτήματος και δημιουργεί ένα νέο περιορισμό, τον οποίο προσθέτει στην τοπική οντολογία.

- `private void createLiteralHasValueRestr(String attributeName,`
  `String stringValue, RDFDatatype literalDatatype,`
  `OntModel localOntModel,`
  `HashMap<String, SemAnnotation> annotationList)`

  Δημιουργεί ένα περιορισμό τιμής με τιμή ένα λεκτικό. Δέχεται ως παραμέτρους το όνομα της ιδιότητας πάνω στην οποία ορίζεται ο περιορισμός, την τιμή του περιορισμού, τον τύπο δεδομένων του, καθώς και την τοπική οντολογία στην οποία θα προστεθεί και τη λίστα με τις επισημειώσεις.

- `private void createIndividualHasValueRestr`
  `(String attributeName, String indivName,`
  `OntModel localOntModel,`
  `HashMap<String, SemAnnotation> annotationList)`

Δημιουργεί ένα περιορισμό τιμής με τιμή ένα άτομο. Δέχεται ως παραμέτρους το όνομα της ιδιότητας πάνω στην οποία ορίζεται ο περιορισμός, το όνομα του ατόμου, καθώς και την τοπική οντολογία στην οποία θα προστεθεί και τη λίστα με τις επισημειώσεις.

- `private void createAllValuesFromRestr(String attributeName,`
  `LinkedList<String> indivsNameList,`
  `OntModel localOntModel,`
  `HashMap<String, SemAnnotation> annotationList)`

  Δημιουργεί ένα ολικό περιορισμό τιμών (allValues restriction) με τιμή μια κλάση απαρίθμησης. Δέχεται ως παραμέτρους το όνομα της ιδιότητας πάνω στην οποία ορίζεται ο περιορισμός, μια λίστα με τα άτομα της κλάσης απαρίθμησης, καθώς και την τοπική οντολογία στην οποία θα προστεθεί και τη λίστα με τις επισημειώσεις

5.2.17 mySQLSchema.ExtractSchema

Πρόκειται για μια βοηθητική κλάση η οποία είναι υπεύθυνη για την εξαγωγή του σχήματος μιας βάσης δεδομένων.

Μέθοδοι

- `public static void main(String[] args)`

  Η μέθοδος αυτή συνδέεται με την επιθυμητή βάση δεδομένων και προσπελάζοντας τα μεταδεδομένα που διατηρεί η σύνδεση, εξάγει τις σχέσεις της, καθώς και τα χαρακτηριστικά τα οποία περιέχουν. Είναι ένας απλός τρόπος να μάθουμε το σχήμα μιας άγνωστης βάσης. Το θετικό είναι ότι ο τρόπος αυτός είναι ανεξάρτητος του συστήματος διαχείρισης δεδομένων που φιλοξενεί τη βάση μας κι έτσι μπορεί να χρησιμοποιηθεί για οποιαδήποτε βάση.

5.2.18 jmdb.MakePartitions

Η υπεύθυνη κλάση για την κατάτμηση του αρχικού συνόλου δεδομένων. Επειδή για τα πειράματα χρησιμοποιήθηκε η βάση δεδομένων του IMDB (βλέπε Κεφάλαιο 7 για Πειραματική Αξιολόγηση), ορισμένα στοιχεία της κλάσης είναι σχεδιασμένα ειδικά για αυτή τη βάση, αλλά μπορούν με μικρές αλλαγές να υποστηρίξουν την κατάτμηση οποιασδήποτε βάσης.

Πεδία

- `private static int NUMBER_OF_PEERS`

  Ο αριθμός των κόμβων του δικτύου.

- `private static final String[] DATABASE_TABLES`

  Πίνακας αλφαριθμητικών που περιέχει τα ονόματα των σχέσεων της βάσης δεδομένων.

- `private String[] relationsArr`

  Ένας πίνακας από αλφαριθμητικά που περιέχει τις σχέσεις του FROM-τμήματος του ερωτήματος.

- `private static OntModel jmdbDomainOntModel`

  Το μοντέλο που αναπαριστά την κεντρική οντολογία του συστήματος.

- `private static HashMap<String, Double> tableProbs`

  Πίνακας κατακερματισμού που διατηρεί την πιθανότητα κάθε σχέσης της βάσης δεδομένων να συμπεριληφθεί σε μια κατάτμηση.

- διάφοροι πίνακες που περιέχουν ποικίλες συνθήκες, οι οποίες θα χρησιμεύσουν στην οριζόντια κατάτμηση των σχέσεων.

Μέθοδοι

- `public static void main(String[] args)`

  Είναι η κύρια μέθοδος της κλάσης η οποία πραγματοποιεί την κατάτμηση του αρχικού συνόλου δεδομένων σε όσα τμήματα καθορίζει το πεδίο που δηλώνει τον αριθμό των κόμβων στο δίκτυο.

- ```
  private static void createPeerSchemaAndOntology(int peerId,
        OntModel peerOntModel, LinkedList<String> peerTablesList,
        LinkedList<String> peerRestrList, Connection conn,
        Properties schemasProps)
  ```

  Εξετάζει καθεμιά από τις σχέσεις του αρχικού συνόλου δεδομένων και καθορίζει ποιες από αυτές θα δοθούν στον κάθε κόμβο (κατακόρυφη κατάτμηση), ποια τμήματά τους (οριζόντια κατάτμηση), δημιουργεί το τοπικό σχήμα και προσθέτει τους κατάλληλους περιορισμούς στην τοπική οντολογία.

- ```
  private static void createIndividualsRestriction(int peerId,
        String tableName, String[] rangeArr,
        OntModel peerOntModel, LinkedList<String> peerRestrList,
        Connection conn)
  ```

Δημιουργεί ένα περιορισμό με άτομα ως τιμές. Αν υπάρχει μόνο ένα άτομο, δημιουργείται ένας περιορισμός τιμής, ενώ αν υπάρχουν πολλά δημιουργείται ολικός περιορισμός τιμών από μια κλάση απαρίθμησης. Σε κάθε περίπτωση δημιουργείται και μια όψη στη βάση δεδομένων.

- `private static void createAllValuesFromClass(int peerId, String tableName, String[] rangeArr, OntModel peerOntModel, LinkedList<String> peerRestrList, Connection conn)`

  Δημιουργεί έναν ολικό περιορισμό τιμών με μια κλάση ως τιμή. Μια όψη στη βάση δημιουργείται επίσης.

- `private static void createIntervalHasValue(int peerId, String tableName, int[] rangeArr, OntModel peerOntModel, LinkedList<String> peerRestrList, Connection conn)`

  Δημιουργεί ένα περιορισμό τιμής με ένα διάστημα αριθμών ως τιμή. Μια όψη στη βάση δημιουργείται επίσης.

- `private static void createView(int peerID, Connection conn, String tableName, String selectPart, String fromPart, String wherePart)`

  Δημιουργεί μια όψη στη βάση δεδομένων με βάση τα δοσμένα χαρακτηριστικά.

- `public static LinkedList<Integer> chooseNRandomInts(int maxNum, int numNo)`

  Επιστρέφει `numNo` (παράμετρος της μεθόδου) ψευδοτυχαίους αριθμούς ανάμεσα στο 0 και το `maxNum` (παράμετρος) ταξινομημένους κατά αύξουσα σειρά. Κάθε αριθμός είναι διαφορετικός από όλους τους υπόλοιπους.

- `private static void loadTableProbabilities (HashMap<String, Double> probsMap)`

  Φορτώνει σε ένα πίνακα κατακερματισμού από το κατάλληλο αρχείο τις πιθανότητες κάθε σχέσης να εμφανιστεί σε κάποιο.

5.2.19 jmdb.TestSemanticPeerWithSchemas

Η κλάση αυτή προσομοιώνει την περίπτωση λειτουργίας του δικτύου, κατά την οποία θεωρούμε αυθαίρετο αριθμό κόμβων, επιλέγουμε έναν από αυτούς ως αρχικό κόμβο και συγκρίνουμε το σχήμα του με τα σχήματα των γειτονικών του κόμβων.

Πεδία

- `private static int NUMBER_OF_PEERS`

  Ο αριθμός των κόμβων του επικαλύπτοντος δικτύου.

- `private static int PEERID_TO_COMPARE`

  Δηλώνει το αναγνωριστικό του αρχικού κόμβου.

- `private static final String edgesFile`

  Το όνομα του αρχείου που περιέχει τις ακμές του επικαλύπτοντος δικτύου.

Μέθοδοι

- `public static void main(String[] args)`

  Κατά τη μέθοδο αυτή, αρχικά γίνεται η αρχικοποίηση των κόμβων του συστήματος, φορτώνονται οι αντιστοιχίσεις μεταξύ των σχημάτων και πραγματοποιείται η σημασιολογική τους επισημείωση. Στη συνέχεια, φορτώνεται από το κατάλληλο αρχείο η τοπολογία του δικτύου και ο αρχικός κόμβος συγκρίνεται με τους γείτονες του. Τελικά επιστρέφεται μια λίστα με τους γείτονες του αρχικού κόμβου ταξινομημένη με βάση την ομοιότητά τους με τον κόμβο, καθώς και ο χρόνος διάρκειας της επεξεργασίας.

5.2.20 jmdb.TestSemanticPeerWithQuery

Η κλάση αυτή προσομοιώνει την περίπτωση λειτουργίας του δικτύου, κατά την οποία θεωρούμε αυθαίρετο αριθμό κόμβων, επιλέγουμε έναν από αυτούς ως αρχικό κόμβο, καθορίζουμε ένα ερώτημα το οποίο θέτει ο κόμβος αυτός και τον συγκρίνουμε με τους γειτονικούς του κόμβους, λαμβάνοντας υπόψη το δεδομένο ερώτημα. Στους “καλύτερους” από τους γείτονες ζητείται και η αποτίμηση του ερωτήματος.

Πεδία

- `private static int NUMBER_OF_PEERS`

  Ο αριθμός των κόμβων του επικαλύπτοντος δικτύου.

- `private static int PEERID_TO_BEGIN_QUERY`

  Δηλώνει το αναγνωριστικό του αρχικού κόμβου που θα θέσει το ερώτημα.

- `private static final String edgesFile`

  Το όνομα του αρχείου που περιέχει τις ακμές του επικαλύπτοντος δικτύου.

- `private static final String QUERY`

Το ερώτημα που τίθεται από τον αρχικό κόμβο.

- `private static int PEERS_TO_QUERY`

  Ο μέγιστος αριθμός των κόμβων που τελικά θα αποτιμήσουν το ερώτημα.

Μέθοδοι

- `public static void main(String[] args)`

  Κατά τη μέθοδο αυτή, αρχικά γίνεται η αρχικοποίηση των κόμβων του συστήματος, φορτώνονται οι αντιστοιχίσεις μεταξύ των σχημάτων και πραγματοποιείται η σημασιολογική τους επισημείωση. Στη συνέχεια, φορτώνεται από το κατάλληλο αρχείο η τοπολογία του δικτύου και ο αρχικός κόμβος θέτει το ερώτημα στο δίκτυο. Αυτό προωθείται στους γείτονες του, με τους οποίους τελικά συγκρίνεται ο αρχικός κόμβος, δεδομένου του δοσμένου ερωτήματος. Οι γειτονικοί κόμβοι ταξινομούνται με βάση το βαθμό ομοιότητας και τελικά επιλέγονται οι πρώτοι `PEERS_TO_QUERY` για να αποτιμήσουν το ερώτημα και να επιστρέψουν τα αποτελέσματα πίσω στον αρχικό κόμβο. Και εδώ τυπώνεται η διάρκεια της διαδικασίας για να έχουμε ένα σχετικό μέτρο της επίδοσης του συστήματος.

5.2.21 RecPrecOptions

Η κλάση αυτή είναι υπεύθυνη για τη φόρτωση όλων των παραμέτρων που βρίσκονται στο αρχείο παραμετροποίησης του συστήματος.

Πεδία

- `private static final String OPTIONS_FILE_NAME`

  Αλφαριθμητικό που δηλώνει το όνομα του αρχείου παραμετροποίησης.

- `private static final String OPTIONS_FILE_PATH`

  Αλφαριθμητικό που δηλώνει το μονοπάτι προς το αρχείο παραμετροποίησης.

- διάφορες στατικές μεταβλητές οι οποίες βρίσκονται σε ένα-προς-ένα αντιστοιχία με τις παραμέτρους στο αρχείο παραμετροποίησης και παίρνουν τιμές από αυτές.

Μέθοδοι

- `private static void load()`

  Προσπελαύνει το αρχείο παραμετροποίησης και φορτώνει τις παραμέτρους, δίνοντας τιμές στα αντίστοιχα στατικά πεδία της κλάσης.

- `private static boolean getBooleanProperty`

```
(Properties properties, String property)
```

Διαβάζει μια ιδιότητα που έχει boolean τιμή από το αρχείο παραμετροποίησης και επιστρέφει τον αντίστοιχο boolean.

- `private static int getIntProperty(Properties properties, String property)`

  Διαβάζει μια ιδιότητα που έχει ακέραια τιμή από το αρχείο παραμετροποίησης και επιστρέφει τον αντίστοιχο ακέραιο.

- `private static double getDoubleProperty(Properties properties, String property)`

  Διαβάζει μια ιδιότητα που έχει δεκαδική τιμή από το αρχείο παραμετροποίησης και επιστρέφει τον αντίστοιχο δεκαδικό.

5.3 Κωδικοποίηση Αρχείων

Στην ενότητα αυτή παρουσιάζονται τα αρχεία που χρησιμοποιούνται από το σύστημα τόσο για είσοδο όσο και για έξοδο πληροφοριών κατά τη λειτουργία του.

5.3.1 recprec.properties

Το αρχείο αυτό περιέχει διάφορες επιλογές υπεύθυνες για την ακριβή λειτουργία του συστήματος.

Ο χρήστης έχει τη δυνατότητα να αλλάξει τις διάφορες παραμέτρους που περιέχονται σε αυτό. Έτσι, μπορεί να αλλάξει το URL της κεντρικής οντολογίας, ν α καθορίσει διάφορες μεταβλητές υπεύθυνες για τον υπολογισμό του recall και του precision, να δηλώσει τον αριθμό των κόμβων που θα υπάρχουν στο δίκτυο και πολλά άλλα.

Κάθε γραμμή του αρχείου αναφέρεται και σε μια παράμετρο με τη μορφή:

`όνομα_παραμέτρου = τιμή_παραμέτρου`

Η κωδικοποίηση αυτή της πληροφορίας δεν είναι τυχαία και χρησιμοποιήθηκε και σε άλλα αρχεία, καθώς η γλώσσα προγραμματισμού Java που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος παρέχει μια πρότυπη κλάση (Properties) και ορισμένες μεθόδους για την άμεση διαχείριση αρχείων που έχουν την ανωτέρω μορφή.

5.3.2 tableProbabilities

Το αρχείο αυτό είναι υπεύθυνο για τις κατακόρυφες κατατμήσεις του αρχικού συνόλου δεδομένων. Η κωδικοποίησή του είναι ίδια με του παραπάνω αρχείο:

```
όνομα_σχέσης = πιθανότητα_συμμετοχης_στην_κατάτμηση
```

5.3.3 Edges.txt

Το αρχείο αυτό αποτελεί την έξοδο της γεννήτριας γράφων που χρησιμοποιήθηκε για τη δημιουργία τυχαίων τοπολογιών δικτύου, η οποία θα αναλυθεί στην Ενότητα 6.1.

Θεωρούμε ότι οι κόμβοι του δικτύου έχουν αναγνωριστικά από το 1 εώς το n, όπου n ο αριθμός των κόμβων του δικτύου. Σε κάθε γραμμή του αρχείου δηλώνεται κι από μία ακμή του δικτύου δίνοντας δύο αριθμούς που αποτελούν τα αναγνωριστικά των κόμβων που συνδέει η ακμή.

5.3.4 localPeerSchemas

Το αρχείο αυτό δημιουργείται κατά τη διαδικασία της κατανομής των δεδομένων στο δίκτυο. Περιέχει τόσες γραμμές όσοι και οι κόμβοι του συστήματος και σε κάθε γραμμή του δίνονται τα ονόματα των σχέσεων του αρχικού συνόλου δεδομένων που τελικά θα συμπεριληφθούν στο τοπικό σχήμα του κάθε κόμβου. Η γραμμή που αναφέρεται στον κόμβο με αναγνωριστικό 8 θα μπορούσε να είναι η ακόλουθη:

```
8=actors, movies2actors, colorinfo, countries, language, ratings, movies
```

5.3.5 Αρχεία owl

Το σύστημα περιέχει ένα αρχείο owl το οποίο κωδικοποιεί την κεντρική οντολογία του δικτύου και κατασκευάζεται συνήθως από έναν επεξεργαστή οντολογιών (ontology editor). Επιπλέον, υπάρχει κι ένα αρχείο για κάθε κόμβο του δικτύου το οποίο κωδικοποιεί την τοπική οντολογία (τοπικοί περιορισμοί κάθε κόμβου) και παράγεται είτε από έναν επεξεργαστή οντολογιών είτε μέσω της γλώσσας προγραμματισμού Java.

5.3.6 output.txt

Το αρχείο αυτό χρησιμοποιείται για να ανακατευθύνουμε κατά βούληση την έξοδο της εφαρμογής από την οθόνη.

6

Υλοποίηση

Στο κεφάλαιο αυτό παρουσιάζονται ορισμένες λεπτομέρειες για την υλοποίηση του συστήματος, του οποίου η ανάλυση απαιτήσεων και η σχεδίαση έχουν δοθεί στα Κεφάλαια 4 και 5 αντίστοιχα. Στην ενότητα 6.1 περιγράφονται οι πιο ενδιαφέροντες από τους αλγορίθμους που υλοποιήθηκαν κατά την ανάπτυξη του συστήματος. Στην ενότητα 6.2 γίνεται αναφορά στις πλατφόρμες, τα προγραμματιστικά εργαλεία και τις βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση.

6.1 Αλγόριθμοι

Στην ενότητα αυτή δίνονται σε μορφή ψευδοκώδικα οι σημαντικότεροι από τους αλγορίθμους που υλοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.

6.1.1 Αλγόριθμος υπολογισμού recall και precision μεταξύ κλάσεων

Ο αλγόριθμος αυτός δέχεται ως είσοδο δύο κλάσεις της οντολογίας (`κλάση-1` και `κλάση-2`) και υπολογίζει το recall και το precision ανάμεσα στις δύο αυτές κλάσεις. Ακολουθεί ο αντίστοιχος ψευδοκώδικας.

```
RecallPrecisionΚλασεων(κλάση-1, κλάση-2) {
    Αν η κλάση-1 είναι ξένη (disjoint) με την κλάση-2
        Επέστρεψε recall = 0.0 και precision = 0.0.
    Αλλιώς {
        Τοποθέτησε στη λίστα-1 τις ιδιότητες της κλάση-1.
        Τοποθέτησε στη λίστα-2 τις ιδιότητες της κλάση-2.
        Αν (η λίστα-1 είναι κενή) ή (η λίστα-2 είναι κενή)
            RecallPrecisionΚαθαρωνΚλασεων(κλάση-1, κλάση-2)
        Αλλιώς
            RecallPrecisionΚλασεωνΜεΙδιοτητες(κλάση-1, κλάση-2, λίστα-1,
                λίστα-2)
        Επέστρεψε το recall και το precision
    }
}

RecallPrecisionΚαθαρωνΚλασεων(κλάση-1, κλάση-2) {
    Τοποθέτησε στη λίσταΥπερκλάσεων-1 τις υπερκλάσεις της κλάση-1.
    Τοποθέτησε στη λίσταΥπερκλάσεων-2 τις υπερκλάσεις της κλάση-2.
    Τοποθέτησε στη λίσταΤομήςΥπερκλάσεων την τομή της
        λίσταΥπερκλάσεων-1 και της λίσταΥπερκλάσεων-1.
    Τοποθέτησε στο μήκος-1 το μήκος της λίσταΥπερκλάσεων-1.
    Τοποθέτησε στο μήκος-2 το μήκος της λίσταΥπερκλάσεων-2.
    Τοποθέτησε στο μήκοςΤομής το μήκος της λίσταΤομήςΥπερκλάσεων.
    Επέστρεψε recall = μήκοςΤομής / μήκος-2 και precision =
        μήκοςΤομής / μήκος-1.
}

RecallPrecisionΚλασεωνΜεΙδιοτητες(κλάση-1, κλάση-2, λίστα-1, λίστα-2) {
    συνολικόRecall = 0.0
    Για κάθε ιδιότητα-Β της λίστα-2 {
        συνολικόRecall += (μέγιστη τιμή recall ανάμεσα στην ιδιότητα-Β
            και καθεμιά από τις ιδιότητες της λίστα-1)
    }
    Επέστρεψε recall = συνολικόRecall / (μήκος της λίστα-2).
```

```
    συνολικόPrecision = 0.0
    Για κάθε ιδιότητα-A της λίστα-1 {
        συνολικόPrecision += (μέγιστη τιμή precision ανάμεσα στην
            ιδιότητα-A και καθεμιά από τις ιδιότητες της λίστα-2)
    }
    Επέστρεψε precision = συνολικόPrecision / (μήκος της λίστα-1).
}
```

6.1.2 *Αλγόριθμος υπολογισμού recall και precision μεταξύ ιδιοτήτων*

Ο αλγόριθμος αυτός δέχεται ως είσοδο δύο ιδιότητες της οντολογίας (*ιδιότητα-1* και *ιδιότητα-2*) και υπολογίζει το recall και το precision ανάμεσα στις δύο αυτές ιδιότητες. Ο αλγόριθμος δέχεται ακόμη, ως παράμετρο, μια boolean μεταβλητή (*μεΠεριορισμούς*), η οποία καθορίζει αν θα συμπεριληφθούν στους υπολογισμούς οι περιορισμοί που έχουν τεθεί σε καθεμιά από τις ιδιότητες. Ακολουθεί ο αντίστοιχος ψευδοκώδικας.

```
RECALLPRECISIONΙΔΙΟΤΗΤΩΝ(ιδιότητα-1, ιδιότητα-2, μεΠεριορισμούς) {
    Αν (η ιδιότητα-1 είναι κενή) ή (η ιδιότητα-2 είναι κενή)
        Επέστρεψε recall = 0.0 και precision = 0.0.
    Αλλιώς {
        Τοποθέτησε στη λίσταΥπεριδιότητων-1 τις υπεριδιότητες της
            ιδιότητα-1.
        Τοποθέτησε στη λίσταΥπεριδιότητων-2 τις υπεριδιότητες της
            ιδιότητα-2.
        Τοποθέτησε στη λίσταΤομήςΥπεριδιότητων την τομή της
            λίσταΥπεριδιότητων -1 και της λίσταΥπεριδιότητων -1.
        Τοποθέτησε στο μήκος-1 το μήκος της λίσταΥπεριδιότητων -1.
        Τοποθέτησε στο μήκος-2 το μήκος της λίσταΥπεριδιότητων -2.
        Τοποθέτησε στο μήκοςΤομής το μήκος της λίσταΤομήςΥπεριδιότητων.
        recall = μήκοςΤομής / μήκος-2
        precision = μήκοςΤομής / μήκος-1.
        Αν (μεΠεριορισμούς == αληθές) και (recall*precision != 0.0) {
            Τοποθέτησε στη λίσταΠεριορισμών-1 τους περιορισμούς της
                ιδιότητα-1.
```

```
            Τοποθέτησε στη λίσταΠεριορισμών-2 τους περιορισμούς της
                ιδιότητα-2.
            Αν (η λίσταΠεριορισμών-1 δεν είναι κενή) και
                (η λίσταΠεριορισμών-2 δεν είναι κενή) {
                recall *= RecallΛιστώνΠεριορισμών(λίσταΠεριορισμών-1,
                    λίσταΠεριορισμών-2)
                precision *= PrecisionΛιστώνΠεριορισμών(λίσταΠεριορισμών-1,
                    λίσταΠεριορισμών-2)
            }
        }
        Επέστρεψε το recall και το precision.
    }
}
```

6.1.3 Αλγόριθμος υπολογισμού recall και precision μεταξύ περιορισμών

Ο αλγόριθμος αυτός δέχεται ως είσοδο δύο περιορισμούς της οντολογίας (*περιορισμός-1* και *περιορισμός-2*), καθώς και τις αντίστοιχες ιδιότητες πάνω στις οποίες επιβάλλονται (*ιδιότητα-1* και *ιδιότητα-2*) και υπολογίζει το recall και το precision ανάμεσα στις δύο αυτές κλάσεις. Ακολουθεί ο αντίστοιχος ψευδοκώδικας.

```
RecallPrecisionΠεριορισμών(περιορισμός -1, περιορισμός -2,
    ιδιότητα -1, ιδιότητα -2) {
    Αν ο ένας από τους περιορισμούς είναι κενός και ο άλλος είναι
        περιορισμός πληθυκότητας ή ορίζεται πάνω σε ιδιότητα τύπων
        δεδομένων {
        Αν ο περιορισμός-1 είναι κενός {
            Δώσε την προκαθορισμένη τιμή στο recall.
            precision = 1.0.
        }
        Αλλιώς αν ο περιορισμός-2 είναι κενός {
            recall = 1.0.
            Δώσε την προκαθορισμένη τιμή στο precision.
        }
    }
```

```
    Αλλιώς αν και οι δύο περιορισμοί είναι πληθυκότητας{
        RecallPrecisionΠεριορισμωνΠληθυκοτητας(περιορισμός -1,
            περιορισμός -2)
    }
    Αλλιώς αν και οι δύο είναι περιορισμοί σε ιδιότητες αντικειμένων{
        RecallPrecisionΠεριορισμωνΑντικειμενων(περιορισμός -1,
            περιορισμός -2)
    }
    Αλλιώς αν και οι δύο είναι περιορισμοί σε ιδιότητες τύπων
        δεδομένων {
        RecallPrecisionΠεριορισμωνΤυπωνΔεδομενων(περιορισμός -1,
            περιορισμός -2)
    }
    Αλλιώς {
        recall = -1.0 και precision = -1.0
    }
}

RecallPrecisionΠεριορισμωνΠληθυκοτητας(περιορισμός -1, περιορισμός -2) {
    Τοποθέτησε στο διάστημα-1 το διάστημα αριθμών της πληθυκότητας
        του περιορισμός-1.
    Τοποθέτησε στο διάστημα-2 το διάστημα αριθμών της πληθυκότητας
        του περιορισμός-2.
    Αν το διάστημα-1 είναι ισοδύναμο με το διάστημα-2 {
        recall = 1.0 και precision = 1.0
    }
    Αλλιώς αν το διάστημα-1 είναι υπερσύνολο του διάστημα-2 {
        recall = 0.5 και precision = 1.0
    }
    Αλλιώς αν το διάστημα-1 είναι υποσύνολο του διάστημα-2 {
        recall = 1.0 και precision = 0.5
    }
    Αλλιώς αν υπάρχει επικάλυψη ανάμεσα στο διάστημα-1 και το
        διάστημα-2 {
        recall = 0.5 και precision = 0.5
```

```
    }
    Αλλιώς αν το διάστημα-1 είναι ξένο με το διάστημα-2 {
        recall = 0.0 και precision = 0.0
    }
}

RecallPrecisionΠεριορισμώνΑντικειμένων (περιορισμός -1, περιορισμός -2,
    ιδιότητα -1, ιδιότητα -2) {
    Τοποθέτησε στην κλάση-1 την κλάση στην οποία περιορίζονται οι
        τιμές της ιδιότητα-1.
    Τοποθέτησε στην κλάση-2 την κλάση στην οποία περιορίζονται οι
        τιμές της ιδιότητα-2.
    Αν οι κλάση-1 και κλάση-2 δεν είναι κλάσεις απαρίθμησης {
        RecallPrecisionΚλάσεων(κλάση-1, κλάση-2)
    }
    Αλλιώς αν οι κλάση-1 και κλάση-2 είναι κλάσεις απαρίθμησης {
        Αν οι κλάση-1 και κλάση-2 περιέχουν μόνο ένα άτομο {
            Αν τα άτομα είναι ταυτόσημα
                Υποβάθμισε το recall και το precision κατά ένα
                    παράγοντα-1.
            Αλλιώς αν τα άτομα έχουν το ίδιο όνομα
                Υποβάθμισε το recall και το precision κατά ένα
                    παράγοντα-2.
            Αλλιώς {
                Τοποθέτησε στην κλάσηΑτόμου-1 και στην κλάσηΑτόμου-2
                    τις κλάσεις στις οποίες ανήκουν τα άτομα της
                    κλάση-1 και κλάση-2.
                RecallPrecisionΚλάσεων(κλάσηΑτόμου-1, κλάσηΑτόμου-2)
            }
        }
    }
    Αλλιώς {
        recall = -1.0 και precision = -1.0
    }
}
```

```
RecallPrecisionΠεριορισμώνΤύπωνΔεδομένων(περιορισμός -1, περιορισμός -2) {
    Αν και οι δύο περιορισμοί είναι περιορισμοί ακριβούς τιμής και
        έχουν ως τιμές λεκτικά {
        Τοποθέτησε στο λεκτικό-1 την τιμή του περιορισμος-1.
        Τοποθέτησε στο λεκτικό-2 την τιμή του περιορισμος-2.
        Σύγκρινε το λεκτικό-1 με το λεκτικό-2.
        Αν ο περιορισμός-1 είναι ισοδύναμος με τον περιορισμός-2 {
            recall = 1.0 και precision = 1.0
        }
        Αλλιώς αν ο περιορισμός-1 είναι υπερσύνολο του περιορισμός-2 {
            recall = 0.5 και precision = 1.0
        }
        Αλλιώς αν ο περιορισμός-1 είναι υποσύνολο του περιορισμός-2 {
            recall = 1.0 και precision = 0.5
        }
        Αλλιώς αν υπάρχει επικάλυψη ανάμεσα στον περιορισμός-1 και τον
            περιορισμός-2 {
            recall = 0.5 και precision = 0.5
        }
        Αλλιώς αν ο περιορισμός-1 είναι ξένος με τον περιορισμός-2 {
            recall = 0.0 και precision = 0.0
        }
    }
    Αλλιώς {
        recall = -1.0 και precision = -1.0
    }
}
```

6.1.4 Αλγόριθμος υπολογισμού recall και precision μεταξύ λιστών κλάσεων

Ο αλγόριθμος αυτός δέχεται ως είσοδο δύο λίστες κλάσεων (*λίσταΚλάσεων-1* και *λίσταΚλάσεων-2*) και υπολογίζει το recall και το precision ανάμεσα στις δύο αυτές λίστες. Ακολουθεί ο αντίστοιχος ψευδοκώδικας.

```
RecallPrecisionΛιστώνΚλασεων(λίσταΚλάσεων-1, λίσταΚλάσεων-2) {
```

```
    συνολικόRecall = 0.0
    Για κάθε κλάση-Α της λίσταΚλάσεων-1 {
        συνολικόRecall += (μέγιστη τιμή recall ανάμεσα στην κλάση-Α
            και καθεμιά από τις κλάσεις της λίσταΚλάσεων-2)
    }
    Επέστρεψε recall = συνολικόRecall / (μήκος της λίσταΚλάσεων-1).

    συνολικόPrecision = 0.0
    Για κάθε κλάση-Β της λίσταΚλάσεων-2 {
        συνολικόPrecision += (μέγιστη τιμή precision ανάμεσα στην
            κλάση-Β και καθεμιά από τις ιδιότητες της λίσταΚλάσεων-1)
    }
    Επέστρεψε precision = συνολικόPrecision / (μήκος
        της λίσταΚλάσεων-2).
}
```

6.1.5 *Αλγόριθμος υπολογισμού recall και precision μεταξύ λιστών περιορισμών*

Ο αλγόριθμος αυτός δέχεται ως είσοδο δύο λίστες περιορισμών (*λίσταΠεριορισμών-1* και *λίσταΠεριορισμών-2*) και υπολογίζει το recall και το precision ανάμεσα στις δύο αυτές λίστες. Ακολουθούν δύο συναρτήσεις σε μορφή ψευδοκώδικα. Στην πρώτη υπολογίζεται το recall και στη δεύτερη το precision.

```
RecallΛιστώνΠεριορισμων(λίσταΠεριορισμών-1, λίσταΠεριορισμών-2) {
    recall = 1.0
    Για κάθε περιορισμό-Β της λίσταΠεριορισμών-2 {
        recall *= (μέγιστη τιμή recall ανάμεσα στον περιορισμό-Β και
             κάθε έναν από τους περιορισμούς της λίσταΠεριορισμών-1)
    }
    Επέστρεψε το recall.
}

PrecisionΛιστώνΠεριορισμων(λίσταΠεριορισμών-1, λίσταΠεριορισμών-2) {
    precision = 1.0
    Για κάθε περιορισμό-Α της λίσταΠεριορισμών-1 {
```

```
        precision *= (μέγιστη τιμή precision ανάμεσα στον περιορισμό-Α
            και κάθε έναν από τους περιορισμούς της
            λίσταΠεριορισμών-1)
    }
    Επέστρεψε το precision.
}
```

6.1.6 *Αλγόριθμος υπολογισμού σημασιολογικής ομοιότητας μεταξύ σχημάτων κόμβων*

Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί τις κλάσεις που επισημειώνουν σημασιολογικά τις σχέσεις του τοπικού σχήματος του κόμβου σε συνδυασμό με τις αντίστοιχες κλάσεις ενός γειτονικού κόμβου. Σκοπός είναι η εκτίμηση της σημασιολογικής ομοιότητας μεταξύ των σχημάτων των κόμβων, υπολογίζοντας τις τιμές του recall και του precision ανάμεσα στα δύο αυτά σύνολο των κόμβων. Ακολουθεί ο ψευδοκώδικας του αλγορίθμου.

```
ΣΥΓΚΡΙΣΗΣΧΗΜΑΤΩΝΚΟΜΒΩΝ(λίσταΚλάσεωνΚόμβου, λίσταΚλάσεωνΓείτονα) {
    Αν (η λίσταΚλάσεωνΚόμβου δεν είναι κενή) και
        (η λίσταΚλάσεωνΓείτονα δεν είναι κενή) {
        RECALLPRECISIONΛΙΣΤΩΝΚΛΑΣΕΩΝ(λίσταΚλάσεωνΚόμβου,
            λίσταΚλάσεωνΓείτονα)
    }
    Αλλιώς
       Επέστρεψε recall = 0.0 και precision = 0.0.
}
```

6.1.7 *Αλγόριθμος υπολογισμού σημασιολογικής ομοιότητας μεταξύ κόμβων σε σχέση με κάποιο ερώτημα*

Ο αλγόριθμος αυτός υλοποιεί το σενάριο σημασιολογικής σύγκρισης δύο κόμβων με βάση ένα συγκεκριμένο ερώτημα. Θεωρούμε ότι ένας κόμβος δέχεται ένα αρχικό ερώτημα που τίθεται από ένα γειτονικό του κόμβο. Παράλληλα, ο κόμβος αυτός του στέλνει τη λίστα των κλάσεων που επισημειώνουν τις σχέσεις που εμφανίζονται στο FROM-τμήμα του αρχικού ερωτήματος και τον πίνακα των ιδιοτήτων που επισημειώνουν τα χαρακτηριστικά που

εμφανίζονται στο SELECT-τμήμα. Οι κλάσεις είναι επαυξημένες με τις επιπλέον συνθήκες που θέτει το αρχικό ερώτημα. Ο κόμβος, χρησιμοποιώντας και τις δικές του σημασιολογικές επισημειώσεις πραγματοποιεί τη σημασιολογική σύγκριση.

```
ΣΥΓΚΡΙΣΗΚΟΜΒΩΝΜΕΕΡΩΤΗΜΑ(αρχικόΕρώτημα, λίσταΚλάσεωνΚόμβου,
    λίσταΚλάσεωνΓείτονα, πίνακαςΙδιοτήτωνΚόμβου,
    πίνακαςΙδιοτήτωνΓείτονα) {
    Επανέγγραψε το αρχικόΕρώτημα.
    Με βάση το επανεγγραμμένο ερώτημα, δημιούργησε τη
       λίσταΚλάσεωνΚόμβου και τον πίνακαςΙδιοτήτωνΚόμβου.
    Κάλεσε την RECALLPRECISIONΛΙΣΤΩΝΚΛΑΣΕΩΝ(λίσταΚλάσεωνΚόμβου,
        λίσταΚλάσεωνΓείτονα) και δώσε τιμές στο recallΚλάσεων και
        το precisionΚλάσεων.
    recallΙδιοτήτων = 0.0
    precisionΙδιοτήτων = 0.0
    επανεγγραμμένεςΙδιότητες = 0
    Για κάθε ιδιότητα-Α στον πίνακαςΙδιοτήτωνΓείτονα {
      Αν υπάρχει αντίστοιχη ιδιότητα-Β στον πίνακαςΙδιοτήτωνΚόμβου {
           Κάλεσε την RECALLPRECISIONΙΔΙΟΤΗΤΩΝ(ιδιότητα-Α, ιδιότητα-Β)
              και πρόσθεσε τις τιμές του recall και precision στα
              recallΙδιοτήτων και precisionΙδιοτήτων.
           επανεγγραμμένεςΙδιότητες ++
      }
    }
    recallΙδιοτήτων /= (μέγεθος πίνακαςΙδιοτήτωνΓείτονα).
    precisionΙδιοτήτων /= επανεγγραμμένεςΙδιότητες.
    recall = recallΚλάσεων * recallΙδιοτήτων.
    precision = precisionΚλάσεων * precisionΙδιοτήτων.
    Επέστρεψε τις τιμές του recall και του precision.
}
```

6.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Στην ενότητα αυτή περιγράφονται τα χαρακτηριστικά της συγκεκριμένης υλοποίησης, όπως η πλατφόρμα ανάπτυξης και εκτέλεσης, τα προγραμματιστικά εργαλεία και οι βιβλιοθήκες που χρησιμοποιήθηκαν.

6.2.1 Πλατφόρμα ανάπτυξης

Το σύστημα υλοποιήθηκε με χρήση της γλώσσας προγραμματισμού Java. Για τη μεταγλώττιση και εκτέλεση του κώδικα χρησιμοποιήθηκε το Java Development Kit της Sun Microsystems και συγκεκριμένα η έκδοση 1.6.0_07. Για την αποδοτικότερη παραγωγή του κώδικα χρησιμοποιήθηκε η προγραμματιστική πλατφόρμα Eclipse (έκδοση 3.4).

6.2.2 Jena

Το Jena [jen08] είναι ένα προγραμματιστικό πλαίσιο σε Java για τη δημιουργία εφαρμογών Σημασιολογικού Ιστού. Παρέχει ένα προγραμματιστικό περιβάλλον για RDF, RDFS και OWL. Επιπλέον, υποστηρίζει την SPARQL και περιλαμβάνει μια μηχανή συμπερασμού βασισμένη σε κανόνες. Το Jena είναι ανοικτού κώδικα και έχει δημιουργηθεί κυρίως από το HP Labs Semantic Web Program. Το πλαίσιο του Jena περιλαμβάνει:

- ένα API (προγραμματιστική διεπαφή) για RDF
- ανάγνωση και εγγραφή RDF σε διάφορες μορφές (χρησιμοποιήθηκε η RDF/XML)
- ένα API για OWL
- αποθήκευση στη μνήμη ή σε άλλη μόνιμη θέση (σε βάση δεδομένων για παράδειγμα)
- μια μηχανή ερωτημάτων SPARQL (δε χρησιμοποιήθηκε)

Το Jena χρησιμοποιήθηκε εκτενώς στην εργασία για να δημιουργηθούν μοντέλα οντολογιών, που περιείχαν κλάσεις, ιδιότητες και περιορισμούς, μέσω των οποίων επισημειώνονται σημασιολογικά οι κόμβοι του δικτύου και στη συνέχεια είναι εφικτή η σημασιολογική σύγκριση της πληροφορίας που παρέχουν στο δίκτυο.

6.2.3 Μηχανή Συλλογισμού

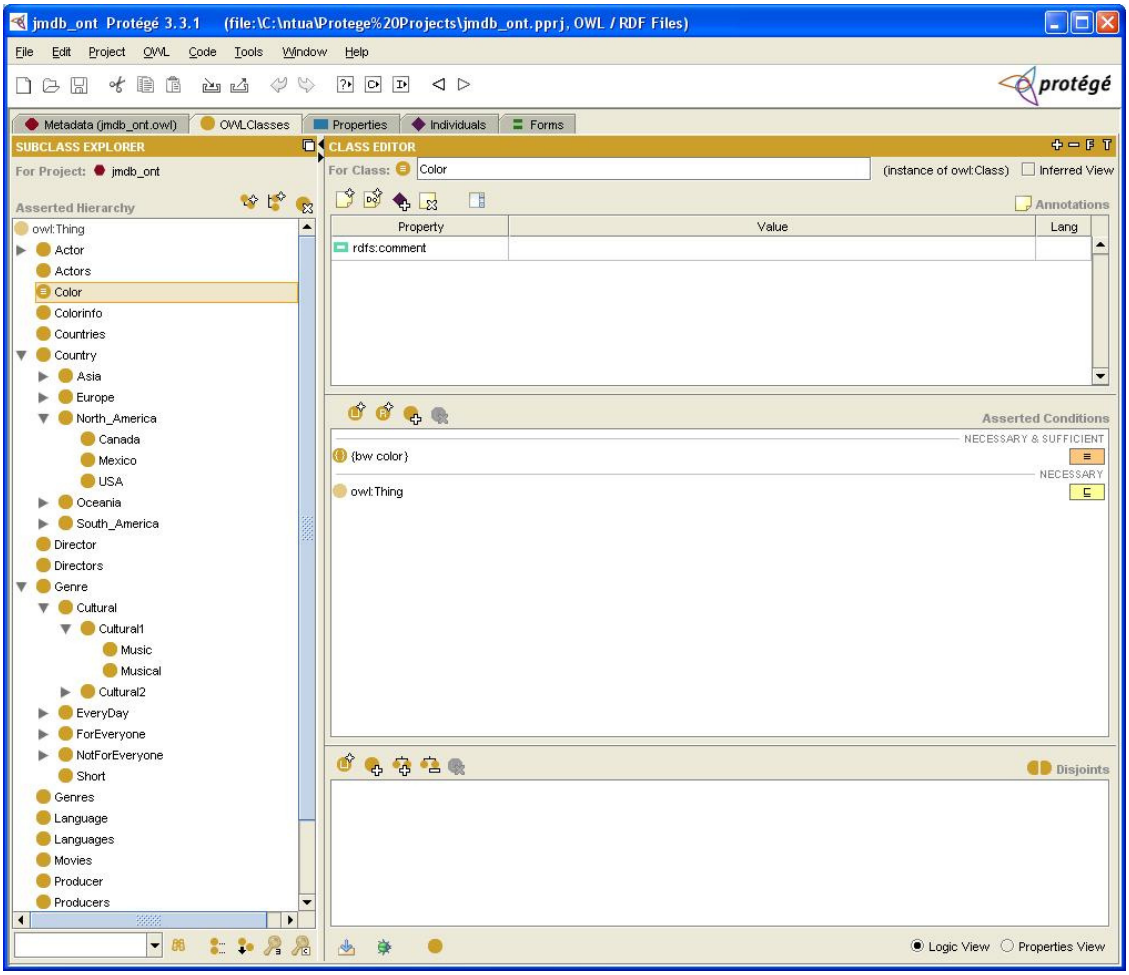
Η μηχανή συλλογισμού (reasoner) είναι μια εφαρμογή η οποία είναι ικανή να εξάγει λογικά συμπεράσματα από κάποια γεγονότα και αξιώματα που της δίνονται.

Για τις ανάγκες του συστήματός μας (συμπερασμός στις οντολογίες), χρησιμοποιήθηκε ο Pellet [pel08]. Ο Pellet είναι μια ανοικτού κώδικα μηχανή συλλογισμού για OWL DL σε Java. Βασίζεται στους αλγορίθμους ταμπλό για τις Περιγραφικές Λογικές και υποστηρίζει την πλήρη εκφραστικότητα της OWL DL (υποστηρίζει όλα τα χαρακτηριστικά της OWL 1.1 εκτός από τους ν-αδικούς τύπους δεδομένων).

Ο Pellet χρησιμοποιήθηκε απευθείας μέσω της κατάλληλης διεπαφής που διαθέτει το Jena.

6.2.4 Επεξεργαστής Οντολογιών

Για τη δημιουργία των κεντρικών οντολογιών που χρησιμοποιήθηκαν κατά τη λειτουργία του συστήματός μας, χρησιμοποιήθηκε το Protégé [pro08]. Το Protégé είναι ένας ανοικτού κώδικα επεξεργαστής οντολογιών και ένα πλαίσιο βάσης γνώσης. Υπάρχουν δύο διαθέσιμοι τρόποι μοντελοποίησης οντολογιών: μέσω του Protégé-Frames και μέσω του Protégé-OWL. Στην παρούσα εργασία χρησιμοποιήθηκε ο δεύτερος. Τέλος, να σημειωθεί ότι το Protégé παρέχει διάφορους τρόπους αποθήκευσης μιας οντολογίας. Χρησιμοποιήσαμε τη μορφή RDF/XML για να είναι συμβατή και με το Jena.



Εικόνα 6.1 Επεξεργασία οντολογίας μέσω του Protege

6.2.5 Σύστημα Διαχείρισης Βάσεων Δεδομένων

Για τις σχεσιακές βάσεις δεδομένων που απαιτούνται για τη λειτουργία του συστήματός μας (αρχικό σύνολο δεδομένων, τοπικά σχήματα κόμβων), χρησιμοποιήθηκε η MySQL[mys08]. Η MySQL είναι το δημοφιλέστερο σύστημα διαχείρισης βάσεων δεδομένων ανοικτού κώδικα. Πέρα από το εργαλείο γραμμής εντολών που περιλαμβάνει, χρησιμοποιήθηκαν και διάφορα άλλα εργαλεία, όπως το GUI Tools (γραφικό περιβάλλον διαχείρισης των βάσεων) και το MySQL Wokbench.

7

Έλεγχος και Πειραματική Αξιολόγηση

Στο κεφάλαιο αυτό παρουσιάζεται η διαδικασία ελέγχου και πειραματικής αξιολόγησης του συστήματος. Στην Ενότητα 7.1 περιγράφονται οι μεθοδολογίες ελέγχου που χρησιμοποιήθηκαν. Στην Ενότητα 7.2 γίνεται αναφορά στο σύνολο των δεδομένων που χρησιμοποιήθηκε για την πειραματική αξιολόγηση του συστήματος, ενώ στην Ενότητα 7.3 περιγράφεται η δημιουργία της τοπολογίας του δικτύου. Στην Ενότητα 7.4 παρουσιάζεται η διαδικασία των πειραμάτων και τα αποτελέσματα που εξήχθησαν από τα πειράματα αυτά.

7.1 Μεθοδολογίες Ελέγχου

Ο έλεγχος σωστής λειτουργίας του συστήματος πραγματοποιήθηκε σε πολλαπλά επίπεδα.

Κατά τη διαδικασία ανάπτυξης του συστήματος, χρησιμοποιήθηκε ο έλεγχος υποσυστημάτων (unit testing), με βάση τον οποίο κάθε υποσύστημα ελέγχεται για τη λειτουργία του ανεξάρτητα από τα υπόλοιπα.

Καθώς δημιουργούνταν νέα υποσυστήματα, τα οποία χρησιμοποιούσαν τα παλιότερα κατά την εκτέλεσή τους, πραγματοποιήθηκε έλεγχος ενσωμάτωσης (integration testing), ώστε να βεβαιωθούμε ότι τα υποσυστήματα συνεργάζονται ικανοποιητικά μεταξύ τους.

Μόλις ολοκληρώθηκε η ανάπτυξη του συστήματος, το σύστημα ελέγχθηκε εξ ολοκλήρου για να διαπιστωθεί ότι ικανοποιεί τις απαιτήσεις λειτουργίας που είχαν τεθεί κατά τη σχεδίασή του.

Για τη διεξαγωγή των παραπάνω ελέγχων χρησιμοποιήθηκε τόσο η μεθοδολογία μαύρου κουτιού (black box testing) όσο και η μεθοδολογία άσπρου κουτιού (white box testing). Σύμφωνα με τον έλεγχο μαύρου κουτιού, το σύστημα εξετάζεται σα να ήταν ένα μαύρο κουτί, δηλαδή χωρίς να ξέρουμε την εσωτερική του υλοποίηση. Έτσι, δίνονται κάποιες τιμές ως είσοδοι και ελέγχεται αν η έξοδος είναι η επιθυμητή. Λόγω, όμως, της πολυπλοκότητας των αλγορίθμων, η χρήση των ελέγχων άσπρου κουτιού κρίθηκε απαραίτητη. Κατά τη μεθοδολογία αυτή, λαμβάνεται υπόψη η εσωτερική δομή του κώδικα που υλοποιεί το σύστημα. Βασικότερο μέλημά μας ήταν να κληθούν τουλάχιστον μία φορά όλες οι εντολές του συστήματος, δηλαδή να “καλυφθεί” όλος ο κώδικας. Κάτι τέτοιο ήταν σχεδόν αδύνατο να πραγματοποιηθεί με τη μεθοδολογία του μαύρου κουτιού.

7.2 Επιλογή και επεξεργασία συνόλου δεδομένων

Ένα από τα βασικότερα στοιχεία ενός δικτύου ομότιμων κόμβων είναι τα δεδομένα που είναι κατανεμημένα στο δίκτυο.

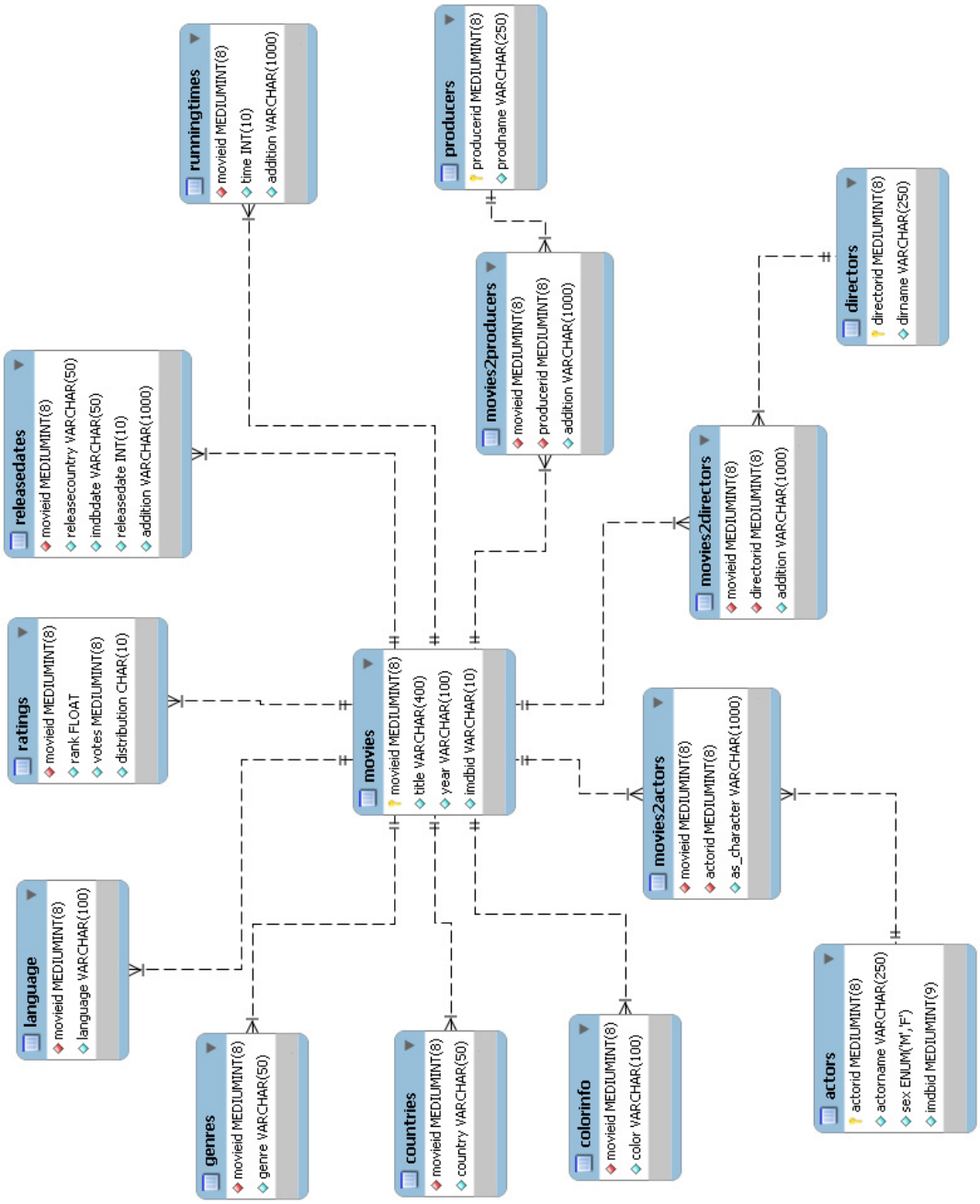
Για τις ανάγκες των πειραμάτων μας ήταν απαραίτητο ένα αρχικό σύνολο δεδομένων το οποίο θα περιείχε ικανοποιητικό όγκο δεδομένων και θα διαμοιραζόταν με τυχαίο τρόπο στους κόμβους του συστήματος. Για το σκοπό αυτό, χρησιμοποιήθηκε η Διεθνής Βάση Δεδομένων Ταινιών (IMDb) [IMD08], τα δεδομένα της οποίας διατίθενται στους χρήστες σε διάφορες μορφές. Στην παρούσα εργασία χρησιμοποιήθηκε η μορφή των αρχείων απλού κειμένου. Για να εισαχθούν τα δεδομένα από τα αρχεία σε μια βάση δεδομένων MySQL, έγινε χρήση της εφαρμογής JMDB [JMD08], η οποία αυτοματοποιεί τη διαδικασία αυτή.

Ενδεικτικό του μεγέθους της βάσης δεδομένων που δημιουργήθηκε είναι το γεγονός ότι περιέχει, μεταξύ άλλων, 1.232.719 ταινίες και 1.495.699 ηθοποιούς.

Επεξεργασία δεδομένων

Για να μπορέσουν να χρησιμοποιηθούν αποδοτικά τα δεδομένα από το σύστημά μας, έπρεπε να περάσουν ένα στάδιο επεξεργασίας. Αρχικά, διαγράφηκαν ορισμένοι από τους πίνακες της βάσης δεδομένων, μιας και ο αρχικό τους αριθμός ήταν μεγάλος και περιείχαν πληροφορίες

που δεν ήταν απαραίτητες. Το σχεσιακό διάγραμμα της βάσης που τελικά χρησιμοποιήθηκε απεικονίζεται στην Εικόνα 7.2.



Εικόνα 7.1 Σχεσιακό διάγραμμα της βάσης δεδομένων

Στη συνέχεια, η βάση δεδομένων έπρεπε να καθαριστεί ώστε να είναι αξιοποιήσιμη από την εφαρμογή μας. Χρησιμοποιήθηκαν διάφορα SQL-ερωτήματα για το σκοπό αυτό. Ακολουθούν μερικά από αυτά.

- ```
  update colorinfo
  set color = case
                  when color like 'Black And White%' then 'bw'
  ```
  ```

```
            when color like 'Color%' then 'color'
            else color
        end;
```

Το ερώτημα αυτό χρησιμοποιήθηκε ώστε η ιδιότητα που δείχνει το χρώμα μιας ταινίας να παίρνει μόνο τις τιμές “bw” (για ασπρόμαυρες) και “color” (για έγχρωμες ταινίες).

- ```
  update runningtimes
  set time = substring(time, 2)
      where time not regexp "^[0-9]"
  ```

  Το ερώτημα αυτό διαγράφει μη αριθμητικούς χαρακτήρες που μπορεί να εμφανίζονται στην αρχή της ιδιότητας που δείχνει τη διάρκεια μιας ταινίας. Στη συνέχεια, οι τιμές της ιδιότητας αυτής μετατράπηκαν σε ακεραίους ώστε να μπορούν να συμμετέχουν σε αριθμητικές πράξεις.

- ```
  update releasedates
      set releasedate = year(str_to_date(releasedate))
  ```

  Το παραπάνω ερώτημα μετατρέπει την ημερομηνία της ιδιότητας που δείχνει την ημερομηνία κυκλοφορίας της ταινίας σε απλή χρονολογία.

Κατανομή των δεδομένων

Αφού ολοκληρώθηκε η επεξεργασία των δεδομένων, χρησιμοποιήθηκε μια λειτουργία της εφαρμογής μας με την οποία οι σχέσεις της βάσης υφίστανται οριζόντια και κατακόρυφη κατάτμηση. Έτσι, δημιουργούνται τόσες κατατμήσεις όσοι και οι κόμβοι του δικτύου και στη συνέχεια κάθε κόμβος παίρνει από μία κατάτμηση. Οι κατατμήσεις προφανώς επικαλύπτονται μεταξύ τους, αφού, σε ένα δίκτυο ομότιμων κόμβων, τα ίδια δεδομένα είναι συχνά διαθέσιμα σε περισσότερους από ένα κόμβους.

7.3 Γεννήτρια γράφων

Για να μπορέσουν να εκτελεστούν τα πειράματα σε ένα περιβάλλον που είναι κοντά σε πραγματικά συστήματα, έπρεπε, εκτός από την επιλογή του κατάλληλου συνόλου δεδομένων,

να δημιουργηθούν τυχαίες τοπολογίες δικτύων με ικανοποιητικό αριθμό κόμβων. Για το σκοπό αυτό χρησιμοποιήθηκε η γεννήτρια γράφων Barabasi [Dre02].

Η γεννήτρια γράφων Barabasi είναι μια γεννήτρια γράφων που βασίζεται στο μοντέλο του Barabasi, το οποίο είναι σχεδιασμένο για να δημιουργεί γράφους που διέπονται από πολυωνυμικές κατανομές (power laws) συσχετισμένες με το βαθμό διακλάδωσης (outdegree) των κόμβων. Βασίζεται στη δημοσίευση [BA99].

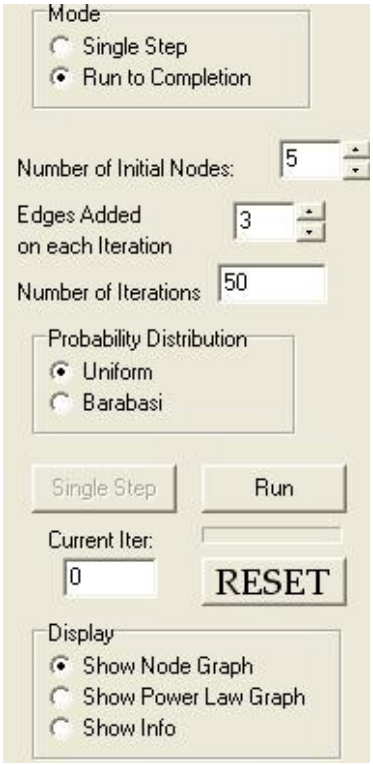
Μέθοδος γένεσης γράφων

Ένας μικρός αριθμός – έστω N – αρχικών κόμβων δημιουργείται. Στη συνέχεια, σε κάθε επανάληψη της γεννήτριας ένας νέος κόμβος προστίθεται και E ακμές δημιουργούνται από το νέο κόμβο στους ήδη υπάρχοντες. Αυτό συνεχίζεται έως ότου επιτευχθεί το επιθυμητό μέγεθος του γράφου. Όσον αφορά στο πώς συνάπτονται οι ακμές στους κόμβους, μια κατανομή πιθανότητας χρησιμοποιείται. Στο [BA99] η κατανομή πιθανότητας που χρησιμοποιήθηκε βασιζόταν στη συνδεσιμότητα των κόμβων. Έστω C_i ο βαθμός του κόμβου i. Τότε, η κατανομή πιθανότητας μπορεί να δοθεί από τη σχέση $P(i) = \frac{C_i + 1}{\sum_i (C_i + 1)}$. Στην παρούσα γεννήτρια γράφων, ο δημιουργός της προσέθεσε άλλη μία κατανομή πιθανότητας, την ομοιόμορφη κατανομή. Στην ομοιόμορφη κατανομή, σε κάθε επανάληψη κάθε κόμβος έχει τις ίδιες πιθανότητες να επιλεγεί.

Μόλις ο γράφος δημιουργηθεί , θέλουμε να δούμε το νόμο δύναμης που σχετίζεται με την τάξη (rank) του κάθε γράφου. Για κάθε βαθμό κορυφής υπολογίζουμε τον αριθμό των κόμβων με αυτό το βαθμό. Έτσι έχουμε ένα σύνολο ζευγών (βαθμός, συχνότητα) και μπορεί να δημιουργηθεί μια γραφική παράσταση με αυτά τα σημεία. Αυτό αποκαλείται τάξη του γράφου.

Στην πραγματικότητα, σχεδιάζουμε την παράσταση αυτή σε λογαριθμική κλίμακα και τα ζεύγη είναι πλέον τα (log(βαθμός), log(συχνότητα)). Θεωρούμε ότι η τάξη ακολουθεί νόμο δύναμης και είναι μια συνάρτηση της μορφής $f(x) = ax^k$, όπου α και k σταθερές. Λογαριθμώντας και τα δύο μέλη της σχέσης προκύπτει $\log(f(x)) = \log a + k \log x$. Ορίζουμε τις νέες μεταβλητές $Y = \log(f(x))$ και $X = \log x$ κι έτσι προκύπτει η σχέση $Y = \log a + kX$, που είναι η εξίσωση μιας γραμμής. Σημαντικά μεγέθη για το αποτέλεσμα είναι η κλίση k της γραμμής και η συσχέτιση που εκφράζει πόσο πιστά ακολουθούν τη γραμμή τα διάφορα σημεία.

Στην Εικόνα 7.2 δίνεται η οθόνη με τις επιλογές.



Εικόνα 7.2 Η οθόνη επιλογών της γεννήτριας γράφων

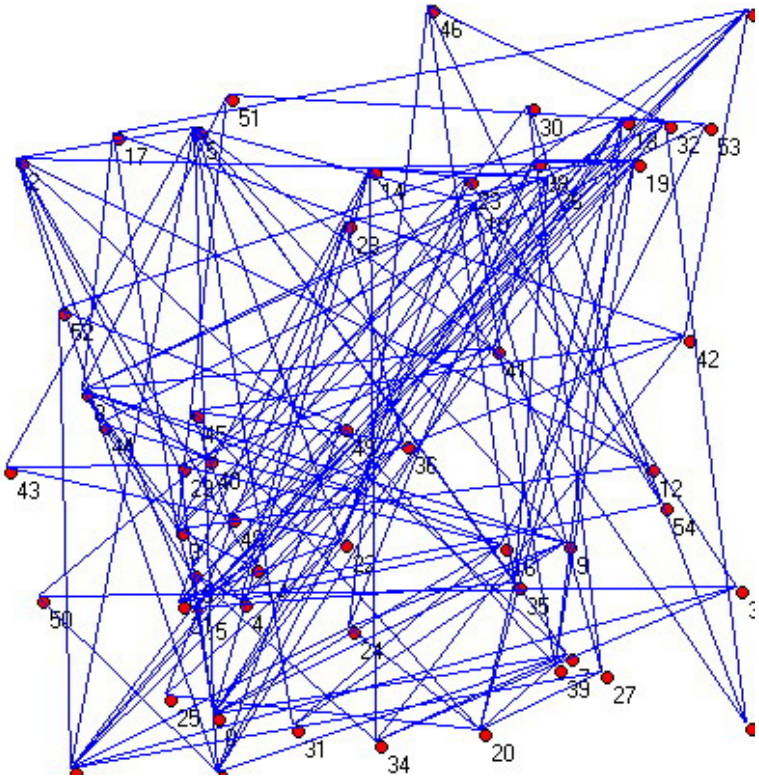
Ο χρήστης έχει τη δυνατότητα να επιλέξει ανάμεσα σε δύο τρόπους λειτουργίας: το βηματικό (single step) και τον απευθείας (run to completion). Για τις ανάγκες τις παρούσας εργασίας χρησιμοποιήθηκε ο δεύτερος. Ο πρώτος είναι χρήσιμος για να δει κανείς τα αποτελέσματα σε κάθε επανάληψη και να καταλάβει τον αλγόριθμο εκτέλεσης.

Επιπλέον, επιλέγει τον αριθμό των αρχικών κόμβων, τον αριθμό των ακμών που θα εισάγονται σε κάθε επανάληψη και τον αριθμό των επαναλήψεων. Πατώντας το κουμπί “Run” με τις παραπάνω ρυθμίσεις, προκύπτει ένας γράφος με 55 κόμβους (5 αρχικοί + ένας σε κάθε επανάληψη). Επίσης, υπάρχει η δυνατότητα επιλογής της κατανομής πιθανότητας ανάμεσα στην ομοιόμορφη και αυτή που προτάθηκε από τον Barabasi στη δημοσίευση.

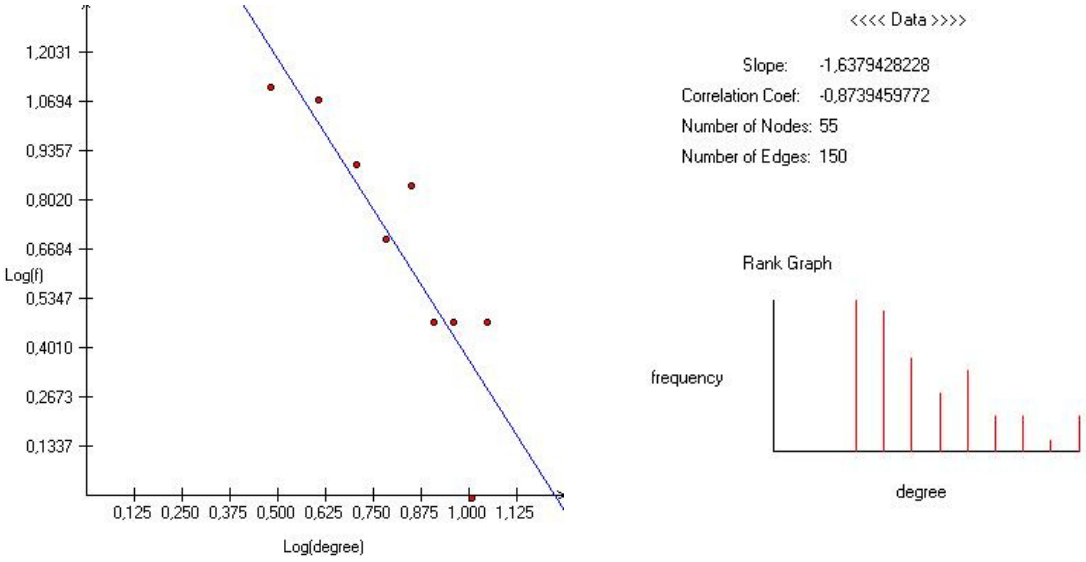
Αφού δημιουργηθεί ο γράφος μπορούμε να δούμε στην οθόνη είτε την τοπολογία του γράφου είτε τη γραφική παράσταση του νόμου δύναμης είτε διάφορες στατιστικές πληροφορίες για το γράφο. Στην Εικόνα 7.3 και την Εικόνα 7.4 φαίνονται οι τρεις αυτές έξοδοι για το γράφο που δημιουργήθηκε με τις παραπάνω επιλογές.

Τέλος, θα πρέπει να σημειωθεί ότι το πρόγραμμα δίνει τη δυνατότητα να εξάγουμε σε ένα txt αρχείο την τοπολογία του γράφου. Το αρχείο αυτό περιέχει τόσες γραμμές όσες είναι οι ακμές του γράφου και σε κάθε γραμμή δίνονται οι δύο αριθμοί των κόμβων που συνδέονται από την εκάστοτε ακμή. Για παράδειγμα, η γραμμή “1 3” δηλώνει ότι ο κόμβος 1 συνδέεται με τον κόμβο 3. Το χαρακτηριστικό αυτό είναι που χρησιμοποιήθηκε κυρίως στην εργασία.

Το πρόγραμμα παίρνει ως είσοδο το αρχείο αυτό κι έτσι αναγνωρίζει την τοπολογία του δικτύου.



Εικόνα 7.3 Τοπολογία του γράφου



Εικόνα 7.4 Γραφική παράσταση νόμου δύναμης και χαρακτηριστικά γράφου

7.4 Διεξαγωγή Πειραμάτων και Αποτελέσματα

Κατά την πειραματική αξιολόγηση του συστήματος εκτιμήθηκε η αποδοτικότητα της σημασιολογικής σύγκρισης μεταξύ των σχημάτων των κόμβων, αλλά και της σύγκρισης των κόμβων αναφορικά με ένα συγκεκριμένο ερώτημα.

Σε κάθε σενάριο λειτουργίας δημιουργούμε ένα δίκτυο ομότιμων κόμβων, διαλέγουμε έναν κόμβο τυχαία και τον συγκρίνουμε με τους γειτονικούς του (είτε μόνο τα σχήματά τους είτε με βάση κάποιο ερώτημα).

Για να υπάρχει ποικιλία συνθηκών, διάφορες παράμετροι μεταβάλλονταν μεταξύ των πειραμάτων. Οι βασικότερες από αυτές τις παραμέτρους είναι ο αριθμός των κόμβων του δικτύου (ο οποίος έμμεσα καθορίζει και τον αριθμό των γειτόνων του κάθε κόμβου), η τοπολογία του δικτύου (χρησιμοποιούνται διάφορες τυχαίες τοπολογίες) και οι κατατμήσεις (γίνεται χρήση ποικίλων κατατμήσεων).

Ένα βασικό ζήτημα είναι ο ποσοτικός καθορισμός της ποιότητας των γειτονικών κόμβων ώστε να αξιολογήσουμε αν το σύστημά μας επιλέγει πράγματι τους καλύτερους κόμβους. Στην περίπτωση της σύγκρισης των σχημάτων των κόμβων, υπολογίζουμε τον αριθμό των ταινιών κάθε γείτονα που ικανοποιούν τους περιορισμούς που θέτει ο αρχικός κόμβος (ο κόμβος που επιλέξαμε για να τον συγκρίνουμε με τους γύρω του) προς το συνολικό αριθμό των ταινιών του γειτονικού κόμβου. Στην περίπτωση που η σύγκριση πραγματοποιείται με βάση ένα συγκεκριμένο ερώτημα, υπολογίζουμε τον αριθμό των ταινιών του γειτονικού κόμβου που ικανοποιούν τους περιορισμούς που θέτει ο αρχικός κόμβος και τους περιορισμούς που επιβάλλουν οι συνθήκες του εκάστοτε ερωτήματος προς τον αριθμό των ταινιών του γείτονα.

Για να εκτιμηθεί η μέθοδος εύρεσης των βέλτιστων γειτόνων που προτείνεται από το σύστημά μας, έπρεπε κάθε φορά να συγκριθεί με κάποιες άλλες μεθόδους. Οι επιπλέον μέθοδοι που χρησιμοποιήθηκαν στη διαδικασία των πειραμάτων είναι η μέθοδος της τυχαίας επιλογής και η μέθοδος της επιλογής με βάση τις αντιστοιχίσεις των σχημάτων. Η μέθοδος της τυχαίας επιλογής, επιλέγει με τυχαίο τρόπο έναν ή περισσότερους από τους γείτονες. Η μέθοδος των αντιστοιχίσεων εκμεταλλεύεται τις αντιστοιχίσεις των σχέσεων για την ταξινόμηση των γειτόνων, χωρίς, όμως, να εκμεταλλεύεται τη επιπλέον σημασιολογική πληροφορία που προσφέρεται από την οντολογία.

Ύστερα από την εκτέλεση σημαντικού πλήθους πειραμάτων, τα αποτελέσματα ήταν ιδιαίτερα ενθαρρυντικά. Χαρακτηριστικά, αναφέρουμε ότι στην ταξινόμηση των γειτόνων με τη χρήση του recall και του precision, ο βέλτιστος (ο πιο σχετικός) κόμβος περιέχεται πάντα στους 20% πρώτους κόμβους που επιστρέφονται. Η απόδοση της μεθόδου τυχαίας επιλογής είναι

ιδιαίτερα μικρότερη, κάτι το οποίο ήταν αναμενόμενο. Το θετικό είναι ότι και η απόδοση της μεθόδου των αντιστοιχίσεων είναι αισθητά χαμηλότερη από τη μέθοδο που χρησιμοποιεί το σύστημά μας. Επίσης, σημαντικό είναι το γεγονός ότι το σύστημά μας προσφέρει κλιμάκωση, αφού η απόδοση του δεν πέφτει καθώς ο αριθμός των γειτόνων αυξάνεται. Αντίθετα, γίνεται ακόμη αποδοτικότερο από τις υπόλοιπες μεθόδους.

8

Επίλογος

Με το κεφάλαιο αυτό ολοκληρώνεται η παρούσα διπλωματική εργασία. Στην Ενότητα 8.1 γίνεται μια σύνοψη της εργασίας και παρουσιάζονται κάποια βασικά συμπεράσματα. Στην ενότητα 8.2 περιγράφονται ορισμένες ενέργειες που θα μπορούσαν μελλοντικά να επεκτείνουν την εργασία αυτή.

8.1 Σύνοψη και Συμπεράσματα

Σκοπός της εργασίας ήταν να αντιμετωπίσουμε τα σημασιολογικά προβλήματα που προκύπτουν στα συστήματα ομότιμων κόμβων που αποτελούνται από κόμβους, οι οποίοι ανταλλάσουν δομημένα δεδομένα σε ένα αδόμητο δίκτυο κάνοντας χρήση των αντιστοιχίσεων μεταξύ των σχημάτων των τοπικών βάσεων δεδομένων. Σε αυτά τα συστήματα, οι κόμβοι ζητούν πληροφορίες θέτοντας ερωτήματα πάνω στο τοπικό σχήμα της βάσης τους. Τα ερωτήματα επανεγγράφονται στα σχήματα των βάσεων των γειτονικών κόμβων, κάνοντας χρήση των διαθέσιμων αντιστοιχίσεων μεταξύ των σχημάτων των κόμβων. Οι διαθέσιμοι αλγόριθμοι επανεγγραφής αναφέρονται σε ερωτήματα που μπορούν να επανεγγραφούν πλήρως με τη χρήση ενός συνόλου αντιστοιχίσεων. Η προσέγγιση αυτή δεν είναι αρκετή για ένα περιβάλλον όπου οι κόμβοι αναζητούν και ικανοποιούνται από πληροφορίες που είναι σημασιολογικά όμοιες αλλά όχι ταυτόσημες με τις απαιτήσεις τους.

Για να αντιμετωπιστούν τα προβλήματα αυτά, δημιουργήσαμε ένα σύστημα το οποίο παρέχει τη δυνατότητα προσδιορισμού του βαθμού ομοιότητας των σχημάτων των κόμβων, καθώς και των ερωτημάτων που προκύπτουν από τη διαδικασία επανεγγραφής σε σχέση με τα αρχικά. Στα πλαίσια αυτά, χρησιμοποιήθηκε μια οντολογία η οποία περιγράφει το πεδίο ενδιαφέροντος των κόμβων. Η οντολογία χρησιμοποιήθηκε για να επισημειώνεται σημασιολογικά κάθε κόμβος, με αποτέλεσμα να περιγράφεται με σαφήνεια η πληροφορία που παρέχεται από αυτόν. Το μέτρο ομοιότητας στο οποίο βασίστηκε το σύστημα υιοθετεί τις έννοιες της ευστοχίας και της ακρίβειας από το πεδίο της ανάκτησης πληροφοριών και λαμβάνει υπόψη τις σημασιολογικές επισημειώσεις των κόμβων, τις αντιστοιχίσεις των σχημάτων και το εκάστοτε ερώτημα.

Για την εκτίμηση της αποδοτικότητας του συστήματος έγινε προσπάθεια να δημιουργηθούν συνθήκες λειτουργίας εφάμιλλες των συνθηκών των πραγματικών δικτύων ομότιμων κόμβων. Έτσι δημιουργήθηκαν τοπολογίες δικτύων με σημαντικό αριθμό κόμβων και χρησιμοποιήθηκε ένα μεγάλο σύνολο δεδομένων το οποίο διαμοιράστηκε σε όλο το δίκτυο. Στη συνέχεια πραγματοποιήθηκε πλήθος πειραμάτων. Τα αποτελέσματα του συστήματός μας, συγκρινόμενα με τις μεθόδους εύρεσης κατάλληλων κόμβων βασισμένες στην τυχαία επιλογή ή στη χρήση μόνο των αντιστοιχίσεων των σχημάτων, είναι ιδιαίτερα ενθαρρυντικά.

8.2 Μελλοντικές επεκτάσεις

Τα θετικά αποτελέσματα που προέκυψαν από την εκτέλεση των πειραμάτων, δίνουν έναυσμα για περαιτέρω εργασίες με σκοπό τη βελτίωση και επέκταση του συστήματος.

Στο υπάρχον σύστημα, για τον υπολογισμό των μετρικών σημασιολογικής ομοιότητας των κόμβων, χρησιμοποιούνται μόνο πληροφορίες που δίνονται από την οντολογία και τις σημασιολογικές επισημειώσεις των σχημάτων των κόμβων. Οι πληροφορίες αυτές θα μπορούσαν να συνδυαστούν με στατιστικά δεδομένα που σχετίζονται με τα περιεχόμενα των τοπικών σχημάτων, ώστε να έχουμε πιο ακριβή υπολογισμό της σημασιολογικής ομοιότητας.

Μια ακόμη πρόκληση αποτελεί το να εφαρμοστούν πιο προχωρημένοι μηχανισμοί δρομολόγησης των ερωτημάτων στο δίκτυο. Για παράδειγμα, κάθε κόμβος θα μπορούσε να διαθέτει εκτιμήσεις της σημασιολογικής ομοιότητας ολόκληρων μονοπατιών του δικτύου και όχι μόνο των γειτονικών του κόμβων.

Επιπλέον, το σύστημα θα μπορούσε να επεκταθεί για να υποστηρίξει περισσότερες της μίας κεντρικές οντολογίες, χρησιμοποιώντας αντιστοιχίσεις μεταξύ τους. Αρκετές λύσεις έχουν

ήδη προταθεί για την αντιστοίχιση οντολογιών, καθώς αποτελεί ουσιώδες ζήτημα του Σημασιολογικού Ιστού.

Σημαντικό για την εκτίμηση της αποτελεσματικότητας του συστήματος θα ήταν το να γίνουν επιπλέον πειράματα με νέα σύνολα δεδομένων και τοπολογίες δικτύων.

Τέλος, για τη χρηστικότητα του συστήματος, θα ήταν θετικό να δημιουργηθεί ένα γραφικό περιβάλλον φιλικό προς το χρήστη.

9

Βιβλιογραφία

[SKS+07] Dimitrios Skoutas, Verena Kantere, Alkis Simitsis and Timos Sellis. A Semantic Similarity Measure for Data Sharing in P2P Databases. SWDB, 2007

[Ede94] Edelstein, Herb. "Unraveling Client/Server Architecture." *DBMS 7,* 5 (May 1994): 34(7).

[Sch96] Schussel, George. *Client/Server Past, Present, and Future* [online]. Available WWW <URL: http://news.dci.com/geos/dbsejava.htm> (1995).

[Pet08] Mike Peterson, Basic Network Types, http://www.pcc-services.com/articles/network_types.html.

[SKS02] Abraham Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts, The McGraw-Hill Companies Inc. , chapter 22, 2002.

[Sin01] Singhal, Amit (2001), "Modern Information Retrieval: A Brief Overview", *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 24(4): 35–43.

[Cro08] Jon Crowcroft, presentation with title “Peer-peer and Application-level Networking”.

[AS04] Stephanos Androutsellis-Theotokis And Diomidis Spinellis, A Survey of Peer-to-Peer Content Distribution Technologies, ACM Computing Surveys, Vol. 36, No. 4, December 2004, pp. 335–371.

[LCP+04] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma and Steven Lim, A Survey and Comparison of Peer-to-Peer Overlay Network Schemes, IEEE Communications Survey And Tutorial, MARCH 2004.

[Web08] webopedia.com, All about Peer-To-Peer (P2P) , http://www.webopedia.com/DidYouKnow/Internet/2005/peer_to_peer.asp.

[EHK+06] J. Eberspächer, Q. Hofstätter, G. Kunzmann, and S. Zöls, EUNICE 2006 - Tutorial: Peer-to-Peer Networking.

[sol08] P2P World, http://www.solyrich.com.

[Set03] SetiAtHome 2003, http://setiathome.ssl.berkeley.edu.

[Nap08] Napster, http://free.napster.com/.

[Gen03] GenomeAtHome 2003, http://genomeathome.stanford.edu/.

[BGK+02] Bernstein, P.,Giunchiglia, F.,Kementsietsidis, A.,Mylopoulos, J.,Serafini, L., And Zaihrayeu, I. 2002. Data management for peer-to-peer computing: A vision. In Proceedings of theWorkshop on theWeb and Databases (WebDB'02).

[HHL+03] Huebsch, R., Hellerstein, J., Lanham, N., And Thau Loo, B. 2003. Querying the internet with pier. In Proceedings of the 29th VLDB Conference. Berlin, Germany.

[Gnu03] Gnutella 2003, http://gnutella.wego.com.

[MMS+06] F. Mandreoli, R. Martoglia, S. Sassatelli, W. Penzo. SRI: Exploiting Semantic Information for Effective Query Routing in a PDMS. WIDM, 2006.

[HIM+03] A. Halevy, Z. Ives, P. Mork, I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. WWW, 2003.

[LIK06] E. Liarou, S. Idreos, M. Koubarakis. Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks. ISWC, 2006.

[SMK+01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, SIGCOMM 2001.

[BKB+06] I. Vlahavas, P. Kefalas, F. Kokkoras, I. Sakellariou, Artificial Inteligence, 3rd edition, V. Giourdas Publications, 2006.

[Ber01] Tim Berners-Lee, "The Semantic Web", Scientific American, 01-05-2001.

[DKD+05] Li Ding, Pranam Kolari, Zhongli Ding, Sasikanth Avancha, Tim Finin and Anupam Joshi, Using Ontologies in the Semantic Web: A Survey, 2005.

[Sto07] G. Stoilos, Γλώσσες Αναπαράστασης Γνώσης στο Σημασιολογικό Ιστό.

[Ant04] Grigoris Antoniou, A Semantic Web Primer, 2004.

[LSH+08] H. Lausen, M. Stollberg, R. L. Hernández, Y. Ding, S.-K. Han and D. Fensel, Semantic Web Portals – State of the Art Survey

[KC04] Klyne Graham and Carroll JeremyJ, Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/2004/ , REC-rdf-concepts-20040210/; 2004

[MM04] Frank Manola, Eric Miller, RDF Primer, http://www.w3.org/TR/2004/ , REC-rdf-primer-20040210/; 2004

[BG04] Brickley Dan and Guha RV, RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/2004/ , REC-rdf-schema-20040210/; 2004

[MH04] Deborah L. McGuinness and Frank van Harmelen, OWL Web Ontology Language Overview, http://www.w3.org/TR/2004/ , REC-owl-features-20040210/; 2004

[jen08] Jena, A Semantic Web Framework for Java, http://jena.sourceforge.net/

[pel08] Pellet, The open-source OWL DL reasoner, http://pellet.owldl.com/

[pro08] Protégé, The Protégé ontology editor and knowledge acquisition system, http://protege.stanford.edu/

[mys08] MySQL, http://www.mysql.com/

[Dre02] Barabasi Graph Generator, Derek Dreier, Department of Computer Science, University of California Riverside, Version 1.4, July 7, 2002, http://www.cs.ucr.edu/~ddreier/barabasi.html

[BA99] Albert-Lazlo Barabasi and Reka Albert, Emergence of Scaling in Random Networks, Science Vol. 286 October 15, 1999

[IMD08] International Movie Database, http://www.imdb.com/

[JMD08] Java Movie Database, http://www.jmdb.de/