ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση Μηχανισμού Ερωταποκρίσεων για Δίκτυο Ομότιμων Βάσεων Δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΕΩΡΓΙΟΥ Ι. ΟΡΦΑΝΟΥΔΑΚΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ
ΚΑΙ ΔΕΔΟΜΕΝΩΝ

Υλοποίηση Μηχανισμού Ερωταποκρίσεων για Δίκτυο Ομότιμων Βάσεων Δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΕΩΡΓΙΟΥ Ι. ΟΡΦΑΝΟΥΔΑΚΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Μαρτίου 2007

.....	.....	.....
Τιμολέων Σελλής Καθηγητής Ε.Μ.Π.	Νεκτάριος Κοζύρης Επίκ. Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Επίκ. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ
ΚΑΙ ΔΕΔΟΜΕΝΩΝ

.....

ΓΕΩΡΓΙΟΣ Ι. ΟΡΦΑΝΟΥΔΑΚΗΣ

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

Ευχαριστίες

Πρώτα από όλους θα ήθελα να ευχαριστήσω τον κ. Τίμο Σελλή για το ενδιαφέρον που μου έδειξε και για το ότι με την αγάπη του για τους φοιτητές και την επιστήμη του αποτέλεσε για εμένα και για πολλούς άλλους πρότυπο ανθρώπου και καθηγητή. Επίσης θα ήθελα να ευχαριστήσω την υποψήφια διδάκτωρ και συνεπιβλέπουσα της εργασίας μου, Βηρένα Καντερέ, για τη βοήθεια, την εμπιστοσύνη και την υπομονή της σε όλο αυτό το διάστημα. Τέλος ευχαριστώ πολύ την οικογένειά μου, τους συγγενείς και τους κοντινούς ανθρώπους που με στήριξαν σε όλη τη διάρκεια των σπουδών μου και συνετέλεσαν στο να φτάσω σε αυτή την ευχάριστη στιγμή.

Περίληψη

Η διπλωματική εργασία ανήκει στον τομέα των Δικτύων Ομότιμων Βάσεων Δεδομένων (Peer-to-Peer / P2P Database Systems). Μία P2P Βάση Δεδομένων αποτελείται από κόμβους καθένας από τους οποίους διαθέτει μία ιδιωτική βάση δεδομένων. Το σχήμα (schema) της βάσης αυτής είναι εν γένει μοναδικό για κάθε κόμβο. Τα ερωτήματα που θέτει κάθε κόμβος στη βάση του διαδίδονται και στο δίκτυο με σκοπό να απαντηθούν από άλλους κόμβους και να επιστρέψουν αποτελέσματα. Λόγω όμως της ανομοιογένειας στα σχήματα των ΒΔ, απαιτείται η μετάφραση των ερωτημάτων κάθε κόμβου τα οποία βέβαια είναι γραμμένα στο δικό του σχήμα, προς τα σχήματα των άλλων κόμβων. Η μετάφραση επιτυγχάνεται με τη χρήση αντιστοιχήσεων (mappings) ανάμεσα στις ΒΔ γειτονικών κόμβων του δικτύου. Οι αντιστοιχήσεις αυτές δεν είναι απλές ισοδυναμίες ιδιοτήτων αλλά ορίζονται με όψεις (views) πάνω στα σχήματα των γειτονικών κόμβων.

Σκοπός της εργασίας είναι η ανάπτυξη ενός μηχανισμού που θα εκτελεί αποδοτικά τη μετάφραση των ερωτημάτων μίας P2P Βάσης Δεδομένων, χρησιμοποιώντας δύο διαφορετικούς τύπους αντιστοιχήσεων. Ο μηχανισμός αυτός θα λειτουργεί σε δύο στάδια.

Αρχικά θα κάνει μία προεπεξεργασία των τιθέμενων ερωτημάτων, με σκοπό την εύρεση των καλύτερων υποερωτημάτων τους που μπορούν να μεταφραστούν από τις διαθέσιμες αντιστοιχήσεις. Η διαδικασία αυτή είναι απαραίτητη για να μπορεί ο μηχανισμός να ενσωματωθεί σε μία πραγματική P2P ΒΔ, όπου οι αντιστοιχήσεις συνήθως δεν επαρκούν για πλήρη μετάφραση.

Αφού βρεθούν τα ζητούμενα υποερωτήματα, ο μηχανισμός θα χρησιμοποιεί αλγορίθμους μετάφρασης και θα τα μεταφράζει προς τους αντίστοιχους κόμβους. Οι αλγόριθμοι που θα ενσωματωθούν στον μηχανισμό είναι δύο, ένας για κάθε χρησιμοποιούμενο τύπο αντιστοιχήσεων.

Η σχεδίαση και υλοποίηση τόσο της διαδικασίας της προεπεξεργασίας όσο και των αλγορίθμων μετάφρασης θα γίνει για ερωτήματα και αντιστοιχήσεις σε γλώσσα SQL.

Λέξεις Κλειδιά: Δίκτυα Ομότιμων Βάσεων Δεδομένων, P2P Βάσεις Δεδομένων, Μετάφραση ερωτημάτων, Αντιστοιχήσεις, Αλγόριθμοι μετάφρασης .

Abstract

The present thesis refers to Peer-to-Peer Database Systems. Each peer of a P2P Database system owns a private database. The schema of this database is usually unique for each peer. The queries of each peer are also propagated to other peers of the network, in order to return more results. The problem is that because of the difference among the peers' database schemas, the queries need to be translated before being sent to other peers. The query translation is obtained by using mappings between the schemas of neighboring peers. These mappings are not just correspondences between attributes, but views over the peers' schemas.

The purpose of this thesis is to develop a mechanism that translates the queries of a P2P Database system, by using two types of mappings. The mechanism will work in two stages.

Initially, it will execute a query preprocessing, in order to discover the best subquery of each original query that can be translated by the available mappings. This procedure is essential, because the mappings of a real P2P database system are rarely sufficient for the complete rewriting of the queries.

Since the proper subqeries are found, the mechanism will use query rewriting algorithms to translate them to the schemas of the relevant peers. Two algorithms will be embedded to the mechanism, one for each type of used mappings.

The designing and the implementation of both the preprocessing and the rewriting algorithms, will be based on the SQL representation of queries and mappings.

Keywords: Peer-to-Peer Database systems, Query translation, Mappings, Query rewriting algorithms .

Πίνακας περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Αλγορίθμων

1

Εισαγωγή

Η εξάπλωση του Παγκόσμιου Ιστού και γενικά των δικτύων Η/Υ είχε ως αποτέλεσμα να παρουσιάζονται συχνά προβλήματα συμφόρησης. Τα προβλήματα αυτά πηγάζουν από το γεγονός ότι ένα μεγάλο πλήθος πελατών (clients) λαμβάνει πληροφορίες από έναν μικρό αριθμό εξυπηρετητών (servers). Για τη λύση των προβλημάτων αυτών προτάθηκε η αντικατάσταση της αρχιτεκτονικής client – server από μία περισσότερο κατανεμημένη αρχιτεκτονική, εκείνη των Δικτύων Ομότιμων Κόμβων (Peer-to-Peer networks). Σε ένα καθαρό P2P δίκτυο όλοι οι κόμβοι είναι ισότιμοι. Κάθε κόμβος δρα άλλοτε σαν πελάτης και άλλοτε σαν εξυπηρετητής, με αποτέλεσμα η διάδοση των πληροφοριών να γίνεται ομοιόμορφα κατά μήκος του δικτύου και όχι συγκεντρωτικά από ολιγάριθμους κόμβους.

Τα P2P δίκτυα έχουν σημαντικές εφαρμογές, μία από τις οποίες αφορά τις κατανεμημένες Βάσεις Δεδομένων. Μία P2P Βάση Δεδομένων είναι ένα P2P δίκτυο στο οποίο κάθε κόμβος διαθέτει μία προσωπική βάση δεδομένων, και χρησιμοποιεί το δίκτυο για να ανταλλάσσει ερωτήματα και δεδομένα με τους υπόλοιπους κόμβους. Ωστόσο, μία P2P Βάση Δεδομένων διαφέρει σημαντικά από μία κατανεμημένη βάση δεδομένων για τον εξής λόγο. Μία κατανεμημένη ΒΔ αποτελείται από ένα σύνολο κόμβων καθένας από τους οποίους έχει ένα τμήμα μίας αρχικής ΒΔ . Η κατάτμηση της αρχικής ΒΔ έχει γίνει με τρόπο ώστε κάθε κόμβος να μπορεί να ανακτά αποδοτικά από τους υπολοίπους τα δεδομένα που επιθυμεί. Αντίθετα, σε μία P2P ΒΔ δεν υπάρχει κοινή αρχική ΒΔ. Κάθε κόμβος που εισέρχεται στο δίκτυο έχει μία ιδιωτική ΒΔ που μπορεί να σχετίζεται μόνο εννοιολογικά με των υπολοίπων, αλλά όχι σε επίπεδο σχεδιασμού.

Συνεπώς προκύπτει το πρόβλημα της συννενόησης των κόμβων όσον αφορά την απάντηση των ερωτημάτων. Εφόσον κάθε κόμβος έχει μία διαφορετική ΒΔ, πρέπει να βρεθεί ένας τρόπος τα ερωτήματα κάθε κόμβου πριν σταλούν στο δίκτυο να μεταφράζονται προς τα σχήματα (schemas) των ΒΔ των γειτονικών κόμβων. Η μετάφραση των ερωτημάτων είναι εφικτή χάρις την ύπαρξη αντιστοιχήσεων (mappings) που συσχετίζουν ανά δύο τα σχήματα γειτονικών κόμβων.

1.1 Αντικείμενο της διπλωματικής

Η παρούσα διπλωματική εργασία έχει στόχο τη σχεδίαση και υλοποίηση ενός μηχανισμού μετάφρασης ερωτημάτων με χρήση αντιστοιχήσεων, που να μπορεί να λειτουργήσει αποδοτικά στο πλαίσιο μίας P2P Βάσης Δεδομένων. Ο μηχανισμός θα δουλεύει σε δύο στάδια:

Σε πρώτη φάση θα εκτελεί μία προεπεξεργασία των τιθέμενων ερωτημάτων, με σκοπό την εύρεση των καλύτερων υποερωτημάτων που μπορούν να μεταφραστούν από τις διαθέσιμες αντιστοιχήσεις. Η φάση αυτή είναι απαραίτητη για να μπορεί ο μηχανισμός να ενσωματωθεί σε μία πραγματική P2P ΒΔ, όπου οι αντιστοιχήσεις ανάμεσα στα σχήματα των κόμβων είναι γενικά ελλιπείς. Οι ελλιπείς αντιστοιχήσεις οδηγούν σε αποτυχία τη μετάφραση των ερωτημάτων, καθιστώντας έτσι αδύνατη τη λειτουργία της P2P ΒΔ. Η διαδικασία της προεπεξεργασίας έχει στόχο την αποκοπή από τα αρχικά ερωτήματα των τμημάτων που δεν μπορούν να μεταφραστούν, καθιστώντας έτσι τη μετάφρασή τους εφικτή, και στη συνέχεια τη μέτρηση της ποιότητας των υποερωτημάτων που προκύπτουν με βάση συγκεκριμένα κριτήρια.

Η δεύτερη φάση του μηχανισμού θα είναι εκείνη της μετάφρασης των (υπο)ερωτημάτων. Για τη μετάφραση θα χρησιμοποιηθούν δύο διαφορετικοί αλγόριθμοι μετάφρασης ερωτημάτων με χρήση αντιστοιχήσεων, καθένας από τους οποίους εργάζεται με έναν άλλο τύπο αντιστοιχήσεων. Η ενσωμάτωση δύο αλγορίθμων στο στάδιο της μετάφρασης αυξάνει το εύρος των ερωτημάτων που μπορούν να μεταφραστούν, άρα και την απόδοση του συνολικού μηχανισμού.

Η εργασία λοιπόν περιλαμβάνει τη σχεδίαση και υλοποίηση των δύο τμημάτων της διαδικασίας της προεπεξεργασίας (μείωση των αρχικών ερωτημάτων και αξιολόγηση των αξιολόγηση των υποερωτημάτων που προκύπτουν), καθώς και των δύο αλγορίθμων μετάφρασης. Αναλυτικότερα περιγράφεται ο στόχος της εργασίας στην παράγραφο 2.3.4 .

1.2 Οργάνωση του τόμου

Η διπλωματική εργασία είναι οργανωμένη σε επτά κεφάλαια.

Το παρόν κεφάλαιο αποτελεί μία γενική εισαγωγή στο αντικείμενο και στους στόχους της εργασίας.

Στο δεύτερο κεφάλαιο δίνεται το θεωρητικό υπόβαθρο στο οποίο βασίζεται η υλοποίηση της εργασίας. Το κεφάλαιο ξεκινά με μία γενική εισαγωγή στα δίκτυα ομοτίμων κόμβων. Έπειτα εξειδικεύεται στις P2P Βάσεις Δεδομένων και συνεχίζει καλύπτοντας μία σειρά θεμάτων που σχετίζονται με τις διαδικασίες μετάφρασης ερωτημάτων στις P2P ΒΔ. Τέλος, παρουσιάζει το σύστημα GrouPeer, τμήμα του οποίου θα κατασκευαστεί στην εργασία αυτή.

Το τρίτο είναι το κεφάλαιο στο οποίο γίνεται η ανάλυση και η σχεδίαση των μηχανισμών που αναπτύχθηκαν στο πλαίσιο της εργασίας. Οι μηχανισμοί αυτοί είναι αλγόριθμοι που χρησιμοποιούνται για την αποδοτική μετάφραση των ερωτημάτων που ανταλλάσσονται ανάμεσα στους κόμβους μίας P2P Βάσης Δεδομένων. Στο κεφάλαιο περιγράφονται εκτενώς οι σχεδιαστικές ιδέες και πρωτοβουλίες με βάση τις οποίες έγινε η υλοποίηση των αλγορίθμων αυτών.

Στο τέταρτο κεφάλαιο παρουσιάζεται η ακριβής υλοποίηση των αλγορίθμων, δηλαδή εξηγείται η σημασία και η λειτουργία αντίστοιχα των πεδίων και των συναρτήσεων των κλάσεων που δημιουργήθηκαν. Επίσης συσχετίζεται κάθε τμήμα της υλοποίησης με το αντίστοιχο τμήμα της σχεδίασης του προηγούμενου κεφαλαίου.

Στο πέμπτο κεφάλαιο γίνεται ο έλεγχος όσων κατασκευάστηκαν στην εργασία, μέσα από παραδείγματα εκτέλεσης που καλύπτουν ένα μεγάλο μέρος από όσα θα κληθούν οι αλγόριθμοι να αντιμετωπίσουν.

Το έκτο κεφάλαιο αποτελεί τον επίλογο της εργασίας. Στο κεφάλαιο αυτό γίνεται μία σύνοψη όσων επιτεύχθηκαν μέσα από αυτήν, και προτείνονται κάποιες σημαντικές επεκτάσεις της.

Τέλος στο έβδομο κεφάλαιο δίνεται η βιβλιογραφία που χρησιμοποιήθηκε στην εκπόνηση της εργασίας.

2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό γίνεται μία περιγραφή των βασικών εννοιών γύρω από τις οποίες κινείται η παρούσα εργασία. Η περιγραφή είναι αρχικά γενική, στην πορεία εξειδικεύεται στα τμήματα της θεωρίας που συνδέονται στενότερα με το αντικείμενο της εργασίας, και καταλήγει στους ακριβείς της στόχους.

2.1 Δίκτυα Ομοτίμων Κόμβων

Ένα Peer to Peer (P2P) δίκτυο, στα ελληνικά δίκτυο ομοτίμων κόμβων, είναι ένα δίκτυο Η/Υ αποτελούμενο από κόμβους που έχουν ίσες δυνατότητες και δικαιώματα. Σε ένα P2P δίκτυο δεν υπάρχουν κόμβοι – πελάτες και κόμβοι – εξυπηρετητές όπως στις γνωστές client – server αρχιτεκτονικές δικτύων αλλά όλοι οι κόμβοι αναλαμβάνουν και τους δύο ρόλους. Η λειτουργία ενός P2P δικτύου στηρίζεται στην υπολογιστική δύναμη και στη δυνατότητες αποθήκευσης πληροφορίας στους κόμβους του. Η ανταλλαγή πληροφοριών γίνεται μέσω απευθείας συνδέσεων από κόμβο σε κόμβο με τρόπο ώστε, ο κάθε κόμβος του δικτύου να έχει άλλοτε τον ρόλο του πελάτη και άλλοτε τον ρόλο του εξυπηρετητή. Η απουσία κόμβων – εξυπηρετητών έχει ως συνέπεια τα P2P δίκτυα να είναι πλήρως αποκεντρωμένα, γεγονός που αποτρέπει φαινόμενα συμφόρησης που εμφανίζονται συχνά στους κόμβους αυτούς. Επίσης το μοίρασμα της υπολογιστικής δύναμης, του αποθηκευτικού χώρου και του εύρους ζώνης σε ένα P2P δίκτυο, το καθιστά περισσότερο λειτουργικό και αποδοτικό. Τέλος ένα

σημαντικότατο πλεονέκτημα των P2P δικτύων είναι η δυνατότητά τους να επεκτείνονται και να οργανώνονται αυτόματα, χωρίς τη μεσολάβηση ενός κόμβου – πατέρα που διατηρεί όλους τους κόμβους (- παιδιά) του δικτύου κάτω από την εποπτεία του. Αξίζει κανείς να σκεφθεί τη σημασία που αποκτούν οι παραπάνω ιδιότητες των P2P δικτύων δεδομένης της ραγδαίας εξάπλωσης του παγκόσμιου ιστού και της αυξανόμενης χρήσης των Η/Υ και των διαδικτυακών υπηρεσιών.

Παρακάτω θα αναφέρουμε τις κυριότερες εφαρμογές των P2P δικτύων:

2.1.1 File Sharing

Μέσω ενός P2P δικτύου επιτυγχάνεται εύκολα η ανταλλαγή αρχείων μεταξύ των κόμβων που συμμετέχουν σε αυτό. Το πιο γνωστό σύστημα ανταλλαγής αρχείων ήταν το Napster, που χρησιμοποιούνταν για την ανταλλαγή αρχείων μουσικής. Το Napster, όπως και άλλα δίκτυα (OpenNAP, IRC) δεν είναι «καθαρά» P2P διότι χρησιμοποιούν κεντρικούς εξυπηρετητές για ορισμένες εργασίες. Καθαρά P2P δίκτυα που χρησιμοποιούνταν για ανταλλαγή αρχείων ήταν το Gnutella και το Freenet.

2.1.2 Distributed Computing

Ένα P2P δίκτυο μπορεί να χρησιμοποιηθεί για να εκμεταλλευθεί τους πόρους που διαθέτουν οι κόμβοι που το αποτελούν. Ο ελεύθερος χρόνος της CPU και ο ελεύθερος χώρος στο δίσκο ενός κόμβου διατίθενται μέσω του P2P δικτύου για την επεξεργασία ενός μεγάλου προβλήματος που σπάει σε κατάλληλα μέρη και λύνεται συνεργατικά από το δίκτυο. Η ιδέα αυτή λέγεται και GRID computing και χρησιμοποιείται για την επίλυση προβλημάτων που απαιτούν υπολογιστική ισχύ πολλών Η/Υ. Παράδειγμα τέτοιου προβλήματος είναι η προσομοίωση της εξάπλωσης του καρκίνου στον οργανισμό που γίνεται στις ΗΠΑ για ερευνητικούς σκοπούς. Στο P2P δίκτυο που δημιουργήθηκε για τον σκοπό αυτό συμμετέχουν εθελοντικά 2.000.000 χρήστες προσφέροντας τους πόρους των Η/Υ τους.

2.1.3 Instant Messaging

Τα P2P δίκτυα παρέχουν την υποδομή για συνομιλία πραγματικού χρόνου ανάμεσα στους χρήστες τους. Παραδείγματα τέτοιων δικτύων είναι το MSN Messenger και το AOL Instant Messenger. Πολλές εταιρίες επίσης έχουν υιοθετήσει το instant messaging μέσω P2P δικτύου σαν κύριο τρόπο επικοινωνίας και συνεργασίας των εργαζομένων τους.

2.1.4 Data Sharing

Ένα P2P δίκτυο στο οποίο ο κάθε κόμβος διαθέτει μία σχεσιακή Βάση Δεδομένων μπορεί να χρησιμοποιηθεί για ανταλλαγή δεδομένων ανάμεσα στους κόμβους. Οι ΒΔ των κόμβων περιέχουν δεδομένα για ένα συγκεκριμένο αντικείμενο και τα ανταλλάσσουν χρησιμοποιώντας το δίκτυο. Τα ερωτήματα που θέτουν οι κόμβοι στις βάσεις τους, διαδίδονται και στο δίκτυο και επιστρέφουν αποτελέσματα και από άλλους κόμβους. Σε αυτήν τη λειτουργία των P2P δικτύων αναφέρεται η παρούσα εργασία, γι αυτό και θα την αναλύσουμε περισσότερο στην παράγραφο που ακολουθεί.

2.2 P2P Βάσεις Δεδομένων

Όπως προαναφέραμε, ένα P2P data sharing system είναι ένα P2P δίκτυο στο οποίο κάθε κόμβος διαθέτει μία Βάση Δεδομένων, και τα ερωτήματα που θέτει σε αυτήν τίθενται επίσης προς απάντηση και στις Βάσεις Δεδομένων άλλων κόμβων του δικτύου. Κάθε κόμβος είναι ανεξάρτητος και αυτό έχει ως συνέπεια το σχήμα της ΒΔ του (local schema) να είναι διαφορετικό από των άλλων, παρότι όλες οι ΒΔ αναφέρονται γενικά στο ίδιο αντικείμενο. Συνεπώς, πρέπει να βρεθεί κάποιος τρόπος ώστε οι κόμβοι να μπορούν να «συνεννοούνται» μεταξύ τους, δηλαδή τα ερωτήματα ενός κόμβου – τα οποία τίθενται πάντα με βάση το δικό του σχήμα – να μπορεί να τα απαντήσει και κάποιος άλλος κόμβος, που θα έχει διαφορετικό σχήμα. Απαιτείται δηλαδή να γίνεται μία μετάφραση των ερωτημάτων από το σχήμα ενός κόμβου που θέτει ένα ερώτημα στο δίκτυο, προς το σχήμα των άλλων κόμβων του δικτύου, προκειμένου να μπορούν να το απαντήσουν.

Ένας τρόπος για να επιτευχθεί αυτή η μετάφραση είναι αυτός που χρησιμοποιούν τα data integration συστήματα. Τα συστήματα αυτά χρησιμοποιούν ένα ενδιάμεσο σχήμα (mediated schema) μέσω του οποίου επικοινωνούν οι κόμβοι. Δηλαδή, όταν ένας κόμβος αποστέλλει ένα ερώτημα στο δίκτυο, τότε αυτό μεταφράζεται αρχικά στο ενδιάμεσο σχήμα, και έπειτα από το ενδιάμεσο σχήμα στο σχήμα των άλλων κόμβων που θα το απαντήσουν. Η μετάφραση από σχήμα σε σχήμα γίνεται με τη χρήση αντιστοιχήσεων (mappings) που αντιστοιχίζουν στοιχεία (ονόματα πινάκων και γνωρισμάτων) του ενός σχήματος με στοιχεία του άλλου. Σε ενότητα που ακολουθεί, θα γίνει μία πιο λεπτομερής αναφορά στη δομή και στα είδη των αντιστοιχήσεων που χρησιμοποιούνται. Προκειμένου να λειτουργήσει ένα data integration σύστημα, πρέπει κάθε κόμβος με την εισαγωγή του στο δίκτυο να έρχεται σε επαφή με έναν κεντρικό κόμβο (στον οποίο είναι αποθηκευμένο το ενδιάμεσο σχήμα) και να δημιουργεί τις αντιστοιχήσεις με το σχήμα αυτό, ώστε να μπορεί να θέσει και να απαντήσει ερωτήματα. Η ύπαρξη όμως ενός τέτοιου κεντρικού κόμβου με τόσο καθοριστικό ρόλο για τη λειτουργία

του συστήματος, καταργεί την αρχή της αποκέντρωσης που πρέπει να έχουν τα P2P συστήματα.

Ένας δεύτερος τρόπος «μετάφρασης» των ερωτημάτων είναι με τη χρήση αντιστοιχήσεων απευθείας ανάμεσα στα σχήματα των κόμβων, χωρίς τη χρήση ενδιάμεσου σχήματος. Με αυτόν τον τρόπο δουλεύουν τα P2P data sharing συστήματα. Κάθε κόμβος του δικτύου διαθέτει αντιστοιχήσεις με τα σχήματα των γειτόνων του, εκείνοι με τα σχήματα των γειτόνων τους, κοκ. Έτσι όταν ένας κόμβος θέτει ένα ερώτημα, αυτό μεταφράζεται και αποστέλλεται στους γείτονές του, έπειτα εκείνοι το μεταφράζουν και στο αποστέλλουν στους δικούς τους γείτονες, κοκ. Αντίθετα με τα data integration συστήματα, στα P2P data sharing συστήματα δεν υπάρχει κεντρικός κόμβος. Κάθε νέος κόμβος που επιθυμεί να εισέλθει στο δίκτυο, δημιουργεί με ημιαυτόματο τρόπο αντιστοιχήσεις με κόμβους του δικτύου, οι οποίοι συνήθως επιλέγονται τυχαία, και μεταφράζει τα ερωτήματα με βάση αυτές. Στο σημείο αυτό προκύπτει ένα πρόβλημα για τα P2P data sharing συστήματα, το οποίο θα αναλύσουμε στην παράγραφο 2.3 .

2.2.1 Η γλώσσα Datalog

Στο σημείο αυτό, θα κάνουμε μία σύντομη αναφορά στον συμβολισμό της Datalog για την αναπαράσταση συζευκτικών ερωτημάτων (conjunctive queries). Τα συζευκτικά ερωτήματα της Datalog είναι κατάλληλα για την έκφραση select – project – join ερωτημάτων (SPJ queries). Επειδή στην παρούσα εργασία θα ασχοληθούμε μόνο SPJ ερωτήματα και αντιστοιχήσεις, όταν λέμε ότι ένα ερώτημα/αντιστοίχηση είναι σε συζευκτική μορφή (CNF), θα θεωρούμε ότι είναι γραμμένο/η σε Datalog.

Ένα ερώτημα, για την ακρίβεια μία όψη στη γλώσσα της Datalog έχει τη μορφή :

$M(X) :- T_1(X_1), T_2(X_2), \dots , T_n(X_n)$.

Το αριστερό τμήμα του ερωτήματος λέγεται κεφαλή (head) του ερωτήματος ενώ το δεξιό λέγεται σώμα (body). Κάθε στοιχείο (atom) του ερωτήματος λέγεται και κατηγόρημα (predicate) και αποτελείται από ένα όνομα πίνακα και από ένα σύνολο ιδιοτήτων. Το όνομα του κατηγορήματος της κεφαλής είναι το όνομα της όψης. Θα δούμε παρακάτω ότι οι αντιστοιχήσεις που χρησιμοποιούμε σε P2P ΒΔ είναι τέτοιες όψεις, γι αυτό και το όνομα της κεφαλής θα το συμβολίσουμε με M (Mapping). Τα ονόματα των κατηγορημάτων του σώματος είναι τα ονόματα των πινάκων που συμμετέχουν σε μία αντιστοίχηση. Τα συμβολίζουμε με $T_1, T_2, \dots , T_n$. Αντίστοιχα, το X είναι το σύνολο των ιδιοτήτων της αντιστοίχησης, και τα $X_1, X_2, \dots , X_n$ τα σύνολα ιδιοτήτων των πινάκων. Σημαντικό στο συμβολισμό της Datalog είναι το γεγονός ότι τα ονόματα των ιδιοτήτων κάθε πίνακα θεωρούνται γνωστά και δεν εμφανίζονται στο Datalog ερώτημα. Οι ιδιότητες ενός πίνακα

διακρίνονται μεταξύ τους από τη θέση στην οποία βρίσκονται μέσα στο κατηγόρημα που αντιστοιχεί στον πίνακα. Η τιμή κάθε ιδιότητας αναπαρίσταται από μία μεταβλητή, ή μία σταθερά, εάν θέλουμε αυτή να έχει μία συγκεκριμένη τιμή. Με τον τρόπο αυτό, οι σύνδεσμοι (joins) ανάμεσα στους πίνακες εκφράζονται σαν πολλαπλές εμφανίσεις της ίδιας μεταβλητής στις θέσεις των αντίστοιχων ιδιοτήτων. Οι ιδιότητες της αντιστοίχησης παίρνουν τιμές από μεταβλητές που εμφανίζονται σαν ιδιότητες πινάκων στο σώμα του ερωτήματος. Έτσι, για να έχει η αντιστοίχηση νόημα πρέπει το σύνολο X να είναι υποσύνολο (όχι γνήσιο) της ένωσης των συνόλων $X_1, X_2, \dots, X_n$. Εκτός από του παραπάνω τύπου κατηγορήματα, το σώμα ενός ερωτήματος μπορεί να περιέχει κατηγορήματα με αριθμητικές συγκρίσεις. Τα κατηγορήματα αυτά έχουν τη μορφή : $x_i <$ const, όπου στη θέση του '<' μπορεί να υπάρχει οποιοσδήποτε τελεστής σύγκρισης, και στη θέση του 'const' μία αριθμητική ή άλλου τύπου σταθερά, ανάλογα με τον τύπο της μεταβλητής x_i. Απαιτούμε βέβαια η x_i να ανήκει σε κάποιο(α) από τα $X_1, X_2, \dots, X_n$. Τα κατηγορήματα κάθε τύπου του σώματος ενός ερωτήματος Datalog λέγονται subgoals. Τα κατηγορήματα των αριθμητικών συγκρίσεων λέγονται comparison subgoals.

2.2.2 Αντιστοιχήσεις (Mappings)

Στην παράγραφο αυτή θα αναλύσουμε τις αντιστοιχήσεις που χρησιμοποιούνται για τη μετάφραση ερωτημάτων σε P2P data sharing συστήματα. Για να γίνει η ανάλυση περισσότερο κατανοητή, θα εισάγουμε ένα παράδειγμα P2P δικτύου με δύο κόμβους. Οι κόμβοι αυτοί θα είναι δύο νοσοκομεία με τις αντίστοιχες σχεσιακές Βάσεις Δεδομένων τους.

Κόμβος 1

K1_Ιατρός (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

K1_Νοσοκόμος (κωδ_Νοσοκόμου, κωδ_Κλινικής, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

K1_Υπάλληλος (κωδ_Υπαλλήλου, κωδ_Κλινικής, ειδικότητα, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

K1_Κλινική (κωδ_Κλινικής, όνομα, κτίριο, όροφος, αριθμός_κλινών)

K1_Υπάγεται (κωδ_Ιατρού, κωδ_Κλινικής)

K1_Διευθύνει (κωδ_Ιατρού, κωδ_Κλινικής)

K1_Νοσηλευόμενος (κωδ_Νοσηλευομένου, κωδ_Ιατρού, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κόμβος 2

Κ2_Ιατρός (κωδικός_Ιατρού, κωδικός_Κλινικής, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ2_Εργαζόμενος (κωδικός_Εργαζομένου, θέση, ειδικότητα, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ2_Κλινική (κωδικός_Κλινικής, κωδικός_Διευθυντή, όνομα, κτίριο, όροφος, αριθμός κλινών)

Κ2_Εργαζόμενος_Υπάγεται (κωδικός_Εργαζομένου, κωδικός_Κλινικής)

Κ2_Ασθενής (κωδικός_Ασθενούς, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ2_Επιβλέπει (κωδικός_Ιατρού, κωδικός_Ασθενούς)

Ας πάρουμε επίσης ένα SQL ερώτημα στην ΒΔ του Κόμβου 1, και ας υποθέσουμε ότι αυτό είναι το σώμα μίας όψης με όνομα 'Ιατρός_Κλινική':

select Κ1_Ιατρός.όνομα, Κ1_Ιατρός.επώνυμο, Κ1_Κλινική.όνομα

from Κ1_Ιατρός, Κ1_Κλινική, Κ1_Υπάγεται

where Κ1_Ιατρός.κωδ_Ιατρού = Κ1_Υπάγεται.κωδ_Ιατρού **and**
Κ1_Κλινική.κωδ_Κλινικής = Κ1_Υπάγεται.κωδ_Κλινικής **and**
Κ1_Κλινική.κτίριο = 'Α' **and**
Κ1_Κλινική.όροφος > 2

Το ερώτημα αυτό σε συμβολισμό Datalog θα γραφεί ως εξής :

Ιατρός_Κλινική(α, β, γ) :- Κ1_Ιατρός(χ, -, α, β, -, -), Κ1_Κλινική(ψ, γ, 'Α', δ, -),
Κ1_Υπάγεται(χ, ψ), δ > 2

Ορίζουμε τώρα τους τρεις τύπους αντιστοιχήσεων που χρησιμοποιούνται στα P2P συστήματα Βάσεων Δεδομένων. Οι ονομασίες τους είναι οι εξής:

- GAV (Global As View)
- LAV (Local As View)
- GLAV (Global & Local As View)

και είναι δανεισμένες από τα data integration συστήματα, όπου Local είναι το σχήμα του κόμβου που στέλνει ή λαμβάνει ένα ερώτημα και Global είναι το ενδιάμεσο (mediated) σχήμα του συστήματος. Στα P2P data sharing συστήματα, Local είναι το σχήμα του κόμβου

που θέτει ένα ερώτημα και Global το σχήμα του κόμβου που το απαντά. Ας συμβολίσουμε με 'Req_Peer' (Requesting Peer) τον κόμβο που θέτει το ερώτημα και με 'Ans_Peer' (Answering Peer) τον κόμβο που το απαντά. Τότε, οι παραπάνω τύποι αντιστοιχήσεων εκφράζονται σε Datalog ως εξής :

2.2.2.1 GAV αντιστοιχήσεις

GAV είναι οι αντιστοιχήσεις του Req_Peer οι οποίες έχουν στην κεφαλή τους έναν πίνακα από το σχήμα του κόμβου αυτού, και στο σώμα τους πίνακες από το σχήμα του Ans_Peer. Δηλαδή είναι της μορφής :

$R_i(X)$:- $A_1(X_1)$, $A_2(X_2)$, … , $A_n(X_n)$

, όπου R_i ένας πίνακας του Req_Peer και A_1, A_2, … , A_n πίνακες του Ans_Peer.

Οι ιδιότητες του πίνακα της κεφαλής αντιστοιχίζονται σε ιδιότητες των πινάκων του σώματος. Έτσι αν έχουμε μία GAV αντιστοίχηση για κάθε πίνακα R_i του Req_Peer, η οποία αντιστοιχίζει όλες τις ιδιότητες του πίνακα με ιδιότητες των πινάκων του Ans_Peer, μπορούμε εύκολα, με τρόπο που θα εξηγήσουμε σε επόμενη ενότητα, να μεταφράσουμε ένα ερώτημα που τίθεται στο σχήμα του Req_Peer, στο σχήμα του Ans_Peer. Για το λόγο αυτό, οι GAV είναι οι απλούστερες και πιο εύχρηστες αντιστοιχήσεις. Παράδειγμα GAV αντιστοίχησης για τον Κόμβο1 προς τον Κόμβο2 είναι η εξής :

K1_Νοσηλευόμενος(α, β, γ, δ, ε, ζ) :- K2_Ασθενής(α, -, γ, δ, ε, ζ), K2_Επιβλέπει(α, β)

2.2.2.2 LAV αντιστοιχήσεις

LAV τώρα, είναι οι αντιστοιχήσεις του Req_Peer οι οποίες έχουν στην κεφαλή τους έναν πίνακα από το σχήμα του κόμβου Ans_Peer, και στο σώμα τους πίνακες από το σχήμα του Req_Peer. Είναι δηλαδή της μορφής :

$A_i(X)$:- $R_1(X_1)$, $R_2(X_2)$, … , $R_n(X_n)$

, όπου A_i ένας πίνακας του Ans_Peer και R_1, R_2, … , R_n πίνακες του Req_Peer.

Και πάλι οι ιδιότητες του πίνακα της κεφαλής αντιστοιχίζονται σε ιδιότητες των πινάκων του σώματος, όμως η μετάφραση δεν είναι τόσο εύκολη όσο στην προηγούμενη περίπτωση. Η μόνη περίπτωση που η μετάφραση είναι προφανής, είναι εκείνη που το ερώτημα που τίθεται από τον Req_Peer συμπίπτει με μία από τις LAV αντιστοιχήσεις του. Τότε το μεταφρασμένο ερώτημα θα είναι απλά μία επιλογή ιδιοτήτων από έναν πίνακα του Ans_Peer (εκείνου που βρίσκεται στην κεφαλή της συγκεκριμένης LAV αντιστοίχησης). Για τη μετάφραση με LAV αντιστοιχήσεις, χρησιμοποιούνται αλγόριθμοι για τους οποίους θα μιλήσουμε στην επόμενη

ενότητα. Έχουμε να παρατηρήσουμε ότι οι GAV αντιστοιχήσεις ενός κόμβου προς ένα γείτονά του μπορούν να χρησιμοποιηθούν σαν LAV αντιστοιχήσεις του γείτονα προς αυτόν, και το αντίστροφο, δηλαδή οι LAV αντιστοιχήσεις του κόμβου σαν GAV του γείτονα. Παράδειγμα LAV αντιστοίχησης του Κόμβου1 προς τον Κόμβο2 είναι η εξής :

Κ2_Κλινική(α, β, γ, δ, ε, ζ) :- Κ1_Κλινική(α, γ, δ, ε, ζ), Κ1_Διευθύνει(β, α)

2.2.2.3 GLAV αντιστοιχήσεις

GLAV τέλος, λέγονται οι αντιστοιχήσεις που στην κεφαλή τους δεν περιέχουν έναν αλλά περισσότερους πίνακες του Req_Peer και στο σώμα τους επίσης περισσότερους από έναν πίνακες του Ans_Peer. Συνεπώς οι αντιστοιχήσεις αυτού του τύπου σε συμβολισμό Datalog έχουν τη μορφή :

$R_1(X_1), R_2(X_2), \dots , R_n(X_n)$:- $A_1(Y_1), A_2(Y_2), \dots , A_m(Y_m)$

Η μετάφραση ερωτημάτων με χρήση GLAV αντιστοιχήσεων απαιτεί προχωρημένες τεχνικές και δεν θα μας απασχολήσει στην παρούσα εργασία.

2.2.3 Αλγόριθμοι μετάφρασης ερωτημάτων με χρήση αντιστοιχήσεων

2.2.3.1 Query Containment

Σημαντικό χαρακτηριστικό των αλγορίθμων που χρησιμοποιούνται για τη μετάφραση ερωτημάτων με κάθε τύπου αντιστοιχήσεις, είναι ότι σχεδιάζονται έτσι ώστε τα μεταφρασμένα ερωτήματα που αποδίδουν να είναι maximally contained στα αρχικά. Αυτό σημαίνει ότι τα μεταφρασμένα ερωτήματα δεν απαιτείται να είναι καθ' όλα ισοδύναμα με τα αρχικά, αλλά να περιέχονται νοηματικά σε αυτά και να είναι τα καλύτερα δυνατά. Ένα μεταφρασμένο ερώτημα Q' επιτρέπεται να εισάγει περισσότερους περιορισμούς στις ιδιότητες που περιλαμβάνει σε σχέση με το αντίστοιχο αρχικό ερώτημα Q, όταν αυτό είναι αναγκαίο για τη μετάφραση. Δεν μπορεί όμως να αφαιρεί περιορισμούς, ή να εισάγει νέες ιδιότητες που δεν εμφανίζονται στο Q, γιατί τότε το Q' δεν περιέχεται στο Q. Η απαίτηση αυτή εξασφαλίζει ότι ένα μεταφρασμένο ερώτημα όταν εκτελεστεί δεν θα επιστρέψει λάθος αποτελέσματα, δηλαδή εγγραφές που δεν υπακούουν στους περιορισμούς του αρχικού ερωτήματος. Η εισαγωγή νέων περιορισμών επιτρέπεται επειδή μπορεί να μειώνει την εκφραστικότητα του μεταφρασμένου ερωτήματος, άρα και τα αποτελέσματα που αυτό θα επιστρέψει, όμως δεν καταργεί την ορθότητά τους.

2.2.3.2 GAV

Η μετάφραση ερωτημάτων για την περίπτωση των GAV αντιστοιχήσεων, αν θεωρήσουμε ότι τόσο το ερώτημα όσο και οι αντιστοιχήσεις που διαθέτουμε προς τους γειτονικούς κόμβους βρίσκονται σε συζευκτική μορφή, γίνεται με τον τρόπο που περιγράφεται στο [HIM+03], με την κατασκευή ενός rule – goal δέντρου. Το δέντρο έχει ρίζα το αρχικό ερώτημα και παιδιά της ρίζας τους subgoals του ερωτήματος. Για να μεταφράζεται το ερώτημα πρέπει να υπάρχει τουλάχιστον μία GAV αντιστοίχηση για κάθε subgoal. Αν χρησιμοποιήσουμε την αντιστοίχηση αυτή, οι subgoals του αρχικού ερωτήματος μεταφράζονται ο καθένας σε έναν ή περισσότερους subgoals που ανήκουν στο σχήμα του γειτονικού κόμβου. Συνεπώς, προκύπτει ένα ερώτημα σε συζευκτική μορφή που μπορεί να απαντηθεί από τον κόμβο αυτόν.

2.2.3.3 LAV

Στην παράγραφο αυτή θα περιγράψουμε δύο αλγορίθμους που μεταφράζουν ερωτήματα με χρήση LAV αντιστοιχήσεων [Hal01]. Θεωρούμε ότι τόσο το ερώτημα όσο και οι LAV αντιστοιχήσεις που διαθέτουμε βρίσκονται σε CNF.

1. Ο αλγόριθμος Bucket

Ο αλγόριθμος Bucket παίρνει τους subgoals του αρχικού ερωτήματος (που θα το συμβολίσουμε με Q) που δεν περιέχουν τελεστές σύγκρισης και δημιουργεί έναν bucket για καθέναν από αυτούς. Σε κάθε bucket τοποθετεί ορισμένες από τις LAV αντιστοιχήσεις που στο σώμα τους περιέχουν τον πίνακα του αντίστοιχου subgoal. Αν συμβολίσουμε με g τον subgoal του ερωτήματος (στον οποίο ανήκει ο bucket), και g_i τους subgoals μίας LAV αντιστοίχησης (που θα τη συμβολίσουμε με V), αυτή τοποθετείται στον bucket όταν :

Α. Υπάρχει μία αντιστοίχηση μεταβλητών (ενοποιητής) θ του g με μεταβλητές του g_i τέτοια ώστε οι δύο subgoals να ενοποιούνται, δηλαδή να ισχύει $\theta(g) = \theta(g_i)$.

Β. Δεδομένης μίας τέτοιας αντιστοίχησης θ, βρίσκουμε την $\theta_{h(V)}$, η οποία αντιστοιχίζει μόνο τις μεταβλητές που υπάρχουν στον g_i αλλά και στην κεφαλή της V, με αντίστοιχες στον g. Εφαρμόζουμε την $\theta_{h(V)}$ στο Q και στην V και ελέγχουμε τις ακόλουθες δύο συνθήκες. Πρώτον, ότι οι αριθμητικοί περιορισμοί στο $\theta_{h(V)}(Q)$ και στο $\theta_{h(V)}(V)$ μπορούν για κάποια πεδία τιμών των μεταβλητών τους να ικανοποιούνται ταυτόχρονα. Και δεύτερον ότι για κάθε μεταβλητή x η οποία βρίσκεται στον g αλλά και στην κεφαλή του Q, η αντίστοιχή της με βάση την ενοποίηση με θ, βρίσκεται επίσης στην κεφαλή της V.

Αν ισχύουν οι παραπάνω συνθήκες, προσθέτουμε στον bucket του g το κατηγόρημα θ(head(V)). Ακολουθώντας την παραπάνω διαδικασία για όλους τους subgoals του Q,

κατασκευάζουμε τα buckets για καθέναν από αυτούς. Σημειώνουμε ότι ένας subgoal g του Q μπορεί με διαφορετικά θ να ενοποιείται με περισσότερους από έναν subgoals μίας αντιστοίχησης V. Στην περίπτωση αυτή, ο bucket του g ίσως περιέχει την κεφαλή της V παραπάνω από μία φορές, τα θ(head(V)) όμως θα διαφέρουν.

Στη συνέχεια ο αλγόριθμος σχηματίζει υποερωτήματα σε συζευκτική μορφή παίρνοντας όλους τους συνδυασμούς που προκύπτουν αν διαλέξουμε ένα στοιχείο από κάθε bucket. Για να σχηματιστεί ένα τέτοιο υποερώτημα πρέπει να πάρουμε στοιχεία από όλα τα buckets. Αν κάποιο bucket είναι άδειο, αυτό σημαίνει ότι ο αντίστοιχος subgoal του Q δεν μπορεί να μεταφραστεί με καμία LAV αντιστοίχηση, οπότε η μετάφραση του Q αποτυγχάνει. Κάθε υποερώτημα αντιπροσωπεύει και έναν διαφορετικό τρόπο μετάφρασης του Q με χρήση κάποιων από τις LAV αντιστοιχήσεις, και πιθανόν επιστρέφει άλλες πλειάδες σαν αποτέλεσμα. Το τελικό μεταφρασμένο ερώτημα το λαμβάνουμε σαν ένωση των παραπάνω υποερωτημάτων (ώστε να επιτύχουμε τα περισσότερα δυνατά αποτελέσματα) και αν χρειαστεί με την προσθήκη αριθμητικών περιορισμών που περιέχονται στο Q και δεν έχουν μεταφραστεί μέσα από τις αντιστοιχήσεις.

2. Ο αλγόριθμος MiniCon

Ο MiniCon είναι ο αλγόριθμος που υλοποιήσαμε στην παρούσα διπλωματική εργασία, γι αυτό και σε επόμενα κεφάλαια θα αναλυθεί διεξοδικά τόσο ως προς τη λειτουργία όσο και ως προς τη σχεδίαση και υλοποίησή του. Στο σημείο αυτό περιγράφουμε την ιδέα της λειτουργίας του και τον συγκρίνουμε με τον αλγόριθμο Bucket.

Ο MiniCon αρχίζει όπως και ο bucket, βρίσκοντας για κάθε subgoal g του ερωτήματος Q, subgoals των LAV αντιστοιχήσεων που μπορούν να ενοποιηθούν με αυτόν. Όμως, κάθε φορά που θα βρεθεί για κάποιον g ένας τέτοιος g_i από μία αντιστοίχηση V, ο αλγόριθμος δεν συνεχίζει όπως ο bucket αλλά εξετάζει τις μεταβλητές του g μέσα στο ερώτημα Q. Ακολουθώντας τις μεταβλητές με το ίδιο όνομα, δηλαδή τους συνδέσμους (joins) του ερωτήματος, ο MiniCon βρίσκει το ελάχιστο σύνολο των επιπλέον subgoals του Q που πρέπει επίσης να ενοποιούνται με κάποιον subgoal της V, δεδομένου ότι ο g ενοποιείται με τον g_i . Αν για το σύνολο αυτό υπάρχουν όντως οι ζητούμενοι g_i's, ο αλγόριθμος κατασκευάζει για το σύνολο των subgoals του Q που καλύπτονται από την V ένα «επεκτεταμένο» bucket, το οποίο λέγεται MiniCon Descriptor (MCD). Σε κάθε MCD αποθηκεύονται όλα τα στοιχεία που θα χρησιμεύσουν για τη δημιουργία του μεταφρασμένου ερωτήματος. (Η δομή των MCDs θα αναλυθεί σε επόμενο κεφάλαιο). Τέλος, τα MCDs συνδυάζονται με κατάλληλο αλγόριθμο και παράγουν το τελικό ερώτημα.

Το πλεονέκτημα του MiniCon είναι ότι στο τελικό του βήμα χρειάζεται να κάνει πολύ λίγους συνδυασμούς ανάμεσα στα MCDs για να παράγει το μεταφρασμένο ερώτημα. Συνεπώς, ο

χρόνος που απαιτεί η επίπονη εργασία κατασκευής των MCDs αντισταθμίζεται από το πλεονέκτημα αυτό. Πειράματα έχουν δείξει ότι ο MiniCon είναι σημαντικά αποδοτικότερος από τον bucket αλγόριθμο, καθώς και ότι λειτουργεί ικανοποιητικά για μεγάλους αριθμούς διαθέσιμων αντιστοιχήσεων συγκεκριμένων αλλά συνηθισμένων μορφών.

2.3 GrouPeer

Η εισαγωγή νέων κόμβων σε P2P data sharing συστήματα όπως προαναφέραμε γίνεται με τυχαίο τρόπο όσον αφορά την επιλογή των γειτόνων κάθε νέου κόμβου. Σε ένα P2P δίκτυο όμως ο κάθε κόμβος έχει ένα σχήμα το οποίο μπορεί να διαφέρει σημαντικά από το σχήμα άλλων κόμβων. Αντίθετα, μέσα στο δίκτυο σίγουρα υπάρχουν κόμβοι των οποίων τα σχήματα έχουν σημαντικές ομοιότητες με το σχήμα της ΒΔ του νέου κόμβου. Η ομοιότητα όμως των σχημάτων δύο κόμβων επηρεάζει άμεσα την ποιότητα των αντιστοιχήσεων που μπορούν να δημιουργηθούν μεταξύ τους και κατ' επέκταση την ακρίβεια της μετάφρασης των ερωτημάτων ανάμεσά τους. Κατανοούμε λοιπόν πόσο σημαντικό είναι κάθε κόμβος που εισέρχεται στο σύστημα να δημιουργεί αντιστοιχήσεις με όσο το δυνατόν ομοιότερους σε αυτόν κόμβους. Η απαίτηση αυτή αποκτά ακόμη μεγαλύτερη σημασία αν λάβουμε υπόψη και τον τρόπο που απαντώνται τα ερωτήματα στα P2P data sharing συστήματα.

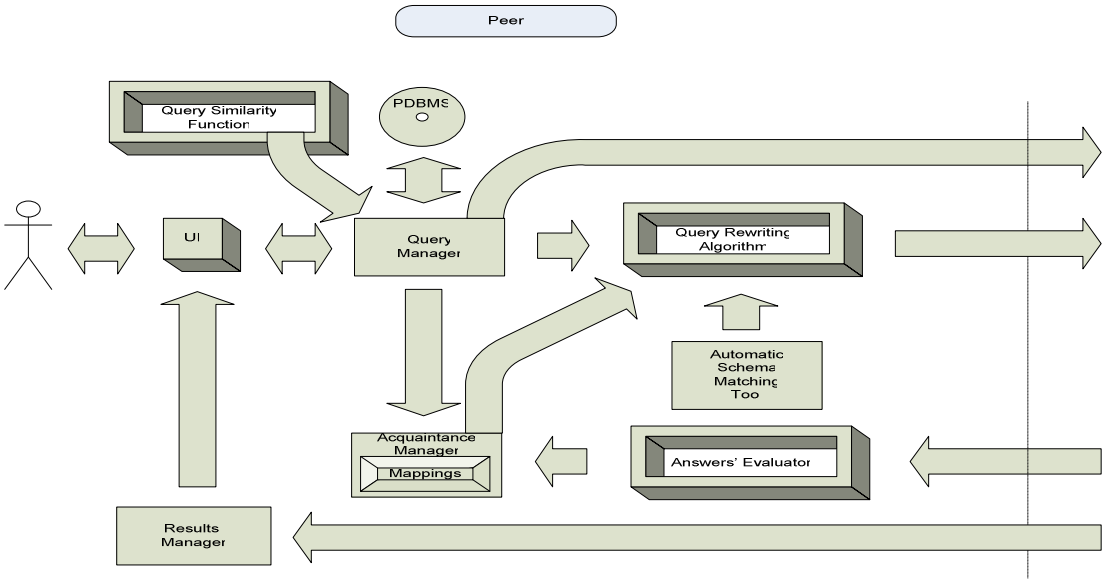
Αναφέραμε παραπάνω ότι η απάντηση των ερωτημάτων γίνεται με διαδοχικές μεταφράσεις και διαδόσεις των ερωτημάτων στο δίκτυο. Η μεταφράσεις γίνονται με βάση τις αντιστοιχήσεις κάθε κόμβου. Εάν αυτές δεν είναι ποιοτικές, τότε κατά τη διάδοσή τους στο δίκτυο τα ερωτήματα γρήγορα απαξιώνονται διότι ορισμένα από τα γνωρίσματά τους χάνονται (διότι οι αντιστοιχήσεις κάποιου κόμβου δεν μπορούν να τα μεταγράψουν) ή δεν μεταφράζονται σωστά (λόγω κακής ποιότητας αντιστοιχήσεων). Συνεπώς, ακόμη και αν μέσα στο δίκτυο υπάρχουν κόμβοι που θα μπορούσαν να απαντήσουν επαρκώς το αρχικό ερώτημα αν διέθεταν τις κατάλληλες αντιστοιχήσεις με τον κόμβο – αποστολέα, δεν θα φανούν χρήσιμοι διότι το ερώτημα θα φτάσει σε αυτούς έχοντας λιγότερα και αλλαγμένα γνωρίσματα.

Το GrouPeer είναι ένας μηχανισμός λειτουργίας P2P data sharing συστημάτων που έχει στόχο να ξεπεράσει τα παραπάνω προβλήματα, διατηρώντας όμως τις αρχές της αποκέντρωσης και αυτοοργάνωσης των P2P δικτύων. Ο μηχανισμός χρησιμοποιεί τα ερωτήματα που διακινούνται στο δίκτυο για να ανακαλύψει και να ομαδοποιήσει κόμβους με παρόμοια σχήματα ΒΔ. Ανάμεσα στους κόμβους αυτούς δημιουργούνται αντιστοιχήσεις που συνδέουν τα κοινά τους στοιχεία και τους καθιστούν πλέον γείτονες. Η ομαδοποίηση

(clustering) επιτυγχάνεται με τη χρήση κλασσικών μεθόδων μεταγραφής ερωτημάτων σε βάσεις δεδομένων, καθώς και με την χρησιμοποίηση εργαλείων που ανακαλύπτουν ομοιότητες ανάμεσα σε σχήματα ΒΔ (automatic schema matching tools).

2.3.1 Λειτουργία του GrouPeer

Στην παράγραφο αυτή θα περιγράψουμε με μεγαλύτερη λεπτομέρεια το μηχανισμό του GrouPeer. Στην περιγραφή θα βοηθήσει το Σχήμα 2.1, στο οποίο φαίνεται η δομή ενός κόμβου στο GrouPeer.



Σχήμα 2.1 : Δομή κόμβου του GrouPeer

Όπως προαναφέραμε, η απάντηση των ερωτημάτων στα P2P data sharing συστήματα στηρίζεται στην μετάφραση και διάδοσή τους από κόμβο σε κόμβο. Οι μηχανισμοί μετάφρασης που χρησιμοποιεί το GrouPeer βασίζονται στους γνωστούς αλγορίθμους για μετάφραση ερωτημάτων με χρήση όψεων (answering queries using views), οι οποίες στην περίπτωσή μας είναι οι αντιστοιχήσεις ανάμεσα στους κόμβους. Οι αλγόριθμοι αυτοί έχουν κατασκευασθεί για να μεταφράζουν ερωτήματα μόνο όταν αυτά μπορούν να μεταφραστούν πλήρως με βάση τις διαθέσιμες όψεις. Το πρόβλημα στα P2P δίκτυα είναι ότι οι αντιστοιχήσεις ανάμεσα στους κόμβους σπάνια είναι πλήρεις, λόγω της ποικιλομορφίας και της ανομοιογένειας που τους χαρακτηρίζει. Προκειμένου λοιπόν τα ερωτήματα που δεν μπορούν να μεταφραστούν πλήρως να μην μεταφράζονται καθόλου, ενισχύουμε τους αλγορίθμους μετάφρασης ώστε να μεταφράζουν όποιο μέρος του ερωτήματος είναι δυνατόν να μεταφραστεί και να παραλείπουν το υπόλοιπο. Έτσι το ερώτημα που τελικά απαντάται

από τους γειτονικούς ενός κόμβου είναι συχνά μειωμένο σε σχέση με το αρχικό. Αυτό δεν αποτελεί πρόβλημα για τα P2P data sharing συστήματα, αφού σε αυτά, όπως και στα P2P file sharing συστήματα, ενδιαφέρουν και οι απαντήσεις που δεν απαντούν ακριβώς το αρχικό ερώτημα, αλλά το προσεγγίζουν σημασιολογικά. Αν με Q_{orig} συμβολίσουμε το αρχικό ερώτημα, τότε εφαρμόζοντάς του μία προεπεξεργασία, δηλαδή αποκόπτοντας από αυτό τα κομμάτια που δεν μπορούν να μεταφραστούν με τις αντιστοιχήσεις που διαθέτουμε, προκύπτει το ερώτημα Q'_{orig}. Στη συνέχεια το Q'_{orig} με χρήση των κλασσικών αλγορίθμων μετάφρασης, μετατρέπεται στο Q'_{rewr} , το οποίο και απαντάται από τον εκάστοτε γειτονικό κόμβο.

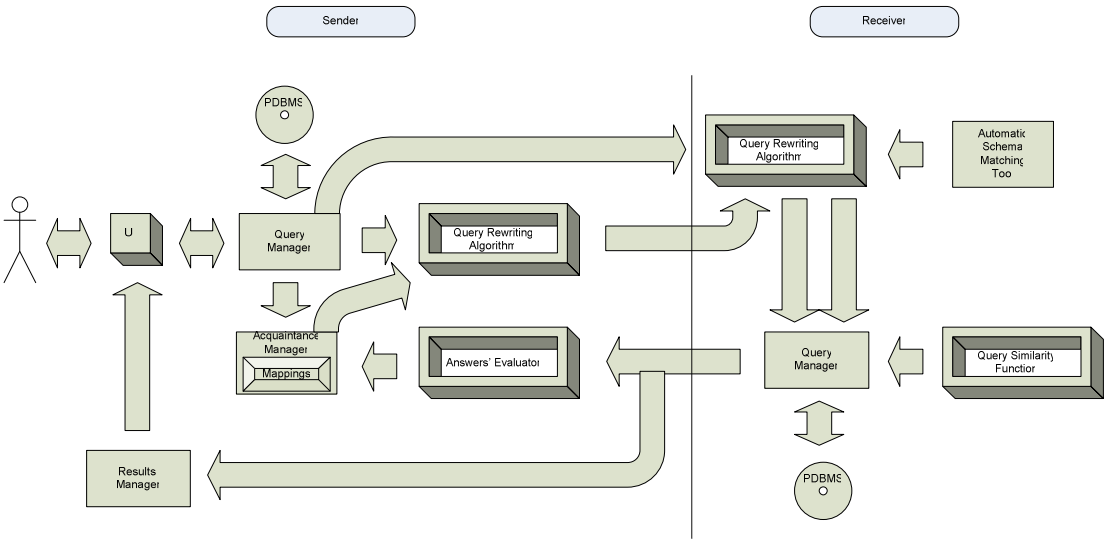
Στο GrouPeer, για να ανακαλυφθούν κόμβοι με κοινά ενδιαφέροντα μέσα στο δίκτυο, εκτός από τη μεταφρασμένη έκδοση κάθε ερωτήματος, αποστέλλεται στον γειτονικό κόμβο και το ίδιο το αρχικό ερώτημα (όπως αυτό τέθηκε από τον πρώτο κόμβο). Στο σημείο αυτό χρησιμοποιείται το εργαλείο για automatic schema matching που υπάρχει στον γειτονικό κόμβο. Το εργαλείο αυτό παίρνει σαν είσοδο το σχήμα του κόμβου αυτού, καθώς και το αρχικό ερώτημα ή/και το σχήμα του αρχικού κόμβου, και κατασκευάζει δυνατές αντιστοιχήσεις ανάμεσα στα στοιχεία τους. Αυτό το πετυχαίνει με τη χρήση λεξικών, τα οποία περιέχουν σε ομάδες λέξεις συνώνυμες ή με κοντινή σημασία. Έτσι όταν στα δύο σχήματα υπάρχουν ονόματα πινάκων ή γνωρισμάτων που βρίσκονται στην ίδια ομάδα του λεξικού, δημιουργείται μία αντιστοιχία ανάμεσα σε αυτά. Χρησιμοποιώντας τώρα τις αντιστοιχήσεις αυτές, το αρχικό ερώτημα μεταφράζεται με τον γνωστό τρόπο και απαντάται από τον κόμβο. Η μέθοδος αυτή δίνει τη δυνατότητα στους κόμβους να απαντούν ερωτήματα χωρίς να επηρεάζονται από την απαξίωση που αυτά έχουν υποστεί λόγω της διάδοσής τους στο δίκτυο. Έτσι, ένας κόμβος που βρίσκεται βαθιά στο δίκτυο σε σχέση με έναν άλλο που έθεσε ένα αρχικό ερώτημα, και ο οποίος έχει σχήμα που ταιριάζει με αυτού, μπορεί μέσα από τις αυτόματα δημιουργούμενες αντιστοιχήσεις να απαντήσει με επιτυχία το ερώτημα αυτό. Γιατί εφόσον τα σχήματα των δύο κόμβων ταιριάζουν, το automatic schema matching tool θα ανακαλύψει τις ομοιότητές τους και θα κατασκευάσει ικανοποιητικές αντιστοιχήσεις. Σε αντίθετη περίπτωση, χωρίς τη χρήση του εργαλείου, ο απομακρυσμένος κόμβος θα αδυνατούσε να απαντήσει το ερώτημα, γιατί θα έφτανε σε αυτόν κατά πολύ αλλαγμένο και κομμένο. Με τον τρόπο αυτό, οι κόμβοι του δικτύου ανακαλύπτουν άλλους με παρόμοια ενδιαφέροντα και σχήματα και σταδιακά δημιουργούν νέες αντιστοιχήσεις μαζί τους.

Συνδυάζοντας τώρα τους δύο παραπάνω τρόπους μετάφρασης και απάντησης ερωτημάτων από το δίκτυο, δηλαδή τη μετάφραση με τη βοήθεια των υπαρχόντων αντιστοιχήσεων και τη μετάφραση με τη βοήθεια των αυτόματα δημιουργούμενων αντιστοιχήσεων, ο μηχανισμός συνεχίζει ως εξής. Χρησιμοποιώντας μία ειδική συνάρτηση η οποία μετρά την ομοιότητα του αρχικού ερωτήματος με το μεταφρασμένο (με τον πρώτο ή τον δεύτερο τρόπο) ο κόμβος

επιλέγει το μεταφρασμένο ερώτημα που προσεγγίζει περισσότερο το αρχικό και το απαντά. (Ο τρόπος κατασκευής της συνάρτησης μέτρησης ομοιότητας θα συζητηθεί στην παράγραφο 2.3.3). Η απάντηση αποστέλλεται στον αρχικό κόμβο ο οποίος έθεσε το ερώτημα. Ο κόμβος αυτός αξιολογεί την απάντηση και αποθηκεύει το αποτέλεσμα της αξιολόγησης. Αν τα αποτελέσματα των αξιολογήσεων είναι καλά για έναν ικανό αριθμό ερωτημάτων, τότε οι κόμβοι γίνονται γείτονες και χρησιμοποιούν τις αντιστοιχήσεις που έχουν έως τη στιγμή αυτή δημιουργηθεί. Οι αντιστοιχήσεις αυτές θα είναι σίγουρα ποιοτικές διότι χρησιμοποιήθηκαν για μετάφραση ερωτημάτων που έδωσαν ικανοποιητικές απαντήσεις. Επίσης συμβαίνει και το αντίστροφο. Εάν δηλαδή ένας κόμβος λαμβάνει από έναν γειτονικό του απαντήσεις που δεν αξιολογούνται ως καλές, τότε οι κόμβοι αυτοί παύουν να είναι γείτονες.

Με τον τρόπο αυτό, επιτυγχάνεται η σταδιακή ομαδοποίηση (clustering) των κόμβων σε ομάδες κόμβων που έχουν ομοιότητες στο σχήμα της ΒΔ, και διαθέτουν καλές αντιστοιχήσεις μεταξύ τους. Ο κάθε κόμβος που συνδέεται στο δίκτυο, ενώ αρχικά συνδέεται με τυχαίους κόμβους, σταδιακά βρίσκει την ομάδα που του ταιριάζει και προσαρτάται σε αυτήν. Τελικά, το δίκτυο διαμορφώνεται έτσι ώστε να γίνεται αποδοτικά και ποιοτικά η απάντηση των ερωτημάτων.

Στο Σχήμα 2.2 που ακολουθεί απεικονίζεται ο μηχανισμός λειτουργίας του GrouPeer μέσα από τη διαδικασία επικοινωνίας δύο κόμβων του P2P δικτύου.



Σχήμα 2.2 : Λειτουργία του GrouPeer

2.3.2 Προεπεξεργασία ερωτημάτων

Όπως αναλύσαμε στην προηγούμενη παράγραφο, προκειμένου να επιτύχουμε την επιτύχουμε τη μετάφραση ερωτημάτων με βάση ορισμένες αντιστοιχήσεις ανάμεσα σε σχήματα δύο κόμβων, απαιτείται η προεπεξεργασία των ερωτημάτων. Και αυτό διότι οι αλγόριθμοι που χρησιμοποιούμε μπορούν να κάνουν μόνο πλήρη μετάφραση ερωτημάτων, δηλαδή δίνουν αποτέλεσμα μόνο για ερωτήματα που μπορούν να μεταφραστούν ολόκληρα με βάση δεδομένες αντιστοιχήσεις. Η προεπεξεργασία σκοπεύει στο να αποκόψει από το αρχικό ερώτημα τα κομμάτια εκείνα τα οποία δεν μπορούν να μεταφραστούν και δημιουργούν το πρόβλημα, και να δώσει σαν είσοδο στους αλγορίθμους το νέο, μειωμένο ερώτημα προς μετάφραση. Παράλληλα, δίνει μία εκτίμηση για το πόσο καλό είναι ένα μειωμένο ερώτημα σε σχέση με το αρχικό. Το πώς ακριβώς βρίσκουμε και αποκόπτουμε τα ανεπιθύμητα κομμάτια των ερωτημάτων, θα αναλυθεί σε επόμενο κεφάλαιο. Εδώ θα συζητήσουμε τη λογική με την οποία υπολογίζεται η τιμή της συνάρτησης που μετρά την ποιότητα του μειωμένου ερωτήματος. Η ιδέα είναι ότι ένα ερώτημα που προκύπτει από την προεπεξεργασία είναι τόσο καλύτερο όσο μεγαλύτερη είναι η ομοιότητά του με το αρχικό.

2.3.3 Μέτρηση ομοιότητας ερωτημάτων

Στη βιβλιογραφία, η ομοιότητα δύο ερωτημάτων μετράται με βάση τα αποτελέσματα τα οποία αυτά επιστρέφουν. Ωστόσο, ο ορισμός αυτός της ομοιότητας δεν είναι χρήσιμος για P2P περιβάλλοντα διότι το αρχικό και το μεταφρασμένο ερώτημα είναι γραμμένα και απαντώνται σε διαφορετικά σχήματα ΒΔ. Επιπλέον, σκοπός μας είναι να μετρήσουμε την ομοιότητα δύο μεταφρασμένων ερωτημάτων (με βάση υπάρχουσες και με βάση αυτόματα δημιουργούμενες αντιστοιχήσεις) με ένα αρχικό ερώτημα προκειμένου να επιλέξουμε ποιο από τα δύο να απαντήσουμε. Συνεπώς, πρέπει να στηριχθούμε στην ανάλυση των ίδιων των ερωτημάτων και όχι των απαντήσεων που δίνουν για να μετρήσουμε την ομοιότητα με το αρχικό ερώτημα. Εξετάζουμε το πόσο θα διαφοροποιήσει το μεταφρασμένο ερώτημα η αποκοπή – λόγω έλλειψης κατάλληλων αντιστοιχήσεων – κάθε στοιχείου του αρχικού ερωτήματος. Επίσης εξετάζουμε την επιρροή που θα έχει η προσθήκη στο μεταφρασμένο ερώτημα επιπλέον στοιχείων, τα οποία εμφανίζονται στις αντιστοιχήσεις :

Απουσία 'select' ιδιοτήτων

Η αποκοπή ιδιοτήτων από το select κομμάτι του αρχικού ερωτήματος μειώνει σημαντικά την ομοιότητα των ερωτημάτων, αφού τα πεδία που επιστρέφει το μεταφρασμένο ερώτημα είναι λιγότερα από εκείνα που ζητά το αρχικό.

Απουσία ιδιοτήτων – κλειδιών

Τα κλειδιά προσδιορίζουν μοναδικά τις τιμές των άλλων ιδιοτήτων και συνεπώς η απουσία τους μειώνει δραστικά την ομοιότητα των ερωτημάτων. Η επίδραση αυτή είναι η ίδια σε όποιο σημείο του αρχικού ερωτήματος και αν βρίσκονται.

Απουσία συνδέσμων σε ιδιότητες – κλειδιά

Εδώ διακρίνουμε δύο περιπτώσεις. Η πρώτη είναι εκείνη όπου ο σύνδεσμος υπάρχει «κρυμμένος» μέσα στις χρησιμοποιούμενες αντιστοιχήσεις, οπότε στην ουσία δεν απουσιάζει από το μεταφρασμένο ερώτημα. Αν αυτό ισχύει, τότε το γεγονός ότι δεν εμφανίζεται στο μεταφρασμένο ερώτημα δεν επιδρά αρνητικά στην ομοιότητα των ερωτημάτων. Η δεύτερη περίπτωση είναι εκείνη όπου αυτό δε συμβαίνει, οπότε ο σύνδεσμος χάνεται. Τα αποτελέσματα που θα επιστραφούν θα είναι κατά πολύ περισσότερα από τα αναμενόμενα. Δηλαδή, η απουσία ενός τέτοιου συνδέσμου είναι κατά πολύ καθοριστικότερη από την απουσία ενός αριθμητικού περιορισμού και έχει σημαντική αρνητική επίδραση στην ομοιότητα των ερωτημάτων.

Απουσία αριθμητικών περιορισμών

Η απουσία ενός αριθμητικού περιορισμού από το μεταφρασμένο ερώτημα έχει ως αποτέλεσμα την επιστροφή ενός υπερσυνόλου των αναμενόμενων αποτελεσμάτων. Τα επιπλέον αποτελέσματα είναι εκείνα που δεν υπακούουν στον περιορισμό. Ωστόσο, δεν αναμένουμε αυτά να είναι κατά το πλήθος πολλά. Συνεπώς θεωρούμε ότι η απουσία ενός τέτοιου περιορισμού επιδρά αρνητικά αλλά όχι καθοριστικά στην ομοιότητα των ερωτημάτων.

Επιπλέον σύνδεσμοι σε ιδιότητες - κλειδιά

Για τους συνδέσμους που συνδέουν ιδιότητες που είναι κλειδιά πινάκων, διακρίνουμε δύο περιπτώσεις. Στη μία περίπτωση οι σύνδεσμοι είναι απαραίτητο να εμφανίζονται στο μεταφρασμένο ερώτημα, ενώ στην άλλη είναι περιττό. Η πρώτη περίπτωση είναι εκείνη όπου στο ‘select’ κομμάτι του μεταφρασμένου ερωτήματος εμφανίζονται ιδιότητες και από τους δύο πίνακες που συνδέονται από τον σύνδεσμο. Στην περίπτωση αυτή ο σύνδεσμος είναι απαραίτητος, γιατί αν έλειπε, μέσω των ‘select’ ιδιοτήτων θα παίρναμε λάθος αποτελέσματα που θα προέκυπταν από καρτεσιανό γινόμενο των δύο σχέσεων. Συνεπώς στην περίπτωση αυτή δεν έχουμε αρνητική επίδραση του συνδέσμου στην ομοιότητα των ερωτημάτων. Στην αντίθετη περίπτωση, όπου μία ή καμία από τις δύο σχέσεις που συνδέονται δεν προσφέρει

‘select’ ιδιότητες, ο σύνδεσμος είναι άχρηστος και απλά περιορίζει τα αποτελέσματα, οπότε επιδρά αρνητικά στην ομοιότητα των ερωτημάτων.

Επιπλέον σύνδεσμοι σε ιδιότητες που δεν είναι κλειδιά

Οι επιπλέον σύνδεσμοι αυτοί εν γένει επιφέρουν μία μείωση στο πλήθος των αποτελεσμάτων που επιστρέφει το μεταφρασμένο ερώτημα, συνεπώς θεωρούμε ότι επηρεάζουν αρνητικά την ομοιότητα των ερωτημάτων.

Επιπλέον αριθμητικοί περιορισμοί

Οι αριθμητικοί περιορισμοί (value constraints) επίσης μειώνουν το πλήθος των αποτελεσμάτων που επιστρέφει το μεταφρασμένο ερώτημα. Πρέπει όμως να υπάρχει κάποιος λόγος για την ύπαρξή τους μέσα στις αντιστοιχήσεις που τους συνήθως τους καθιστά χρήσιμους και ρεαλιστικούς. Γι αυτό και θεωρούμε ότι οι περιορισμοί αυτοί γενικά μειώνουν την ομοιότητα των ερωτημάτων αλλά όχι σε δραστικό βαθμό.

Συγκεντρώνοντας τα αποτελέσματα της παραπάνω ανάλυσης, κατατάσσουμε τα κριτήρια μέτρησης της ομοιότητας ερωτημάτων κατά φθίνουσα σειρά σημαντικότητας :

1. Οι ιδιότητες - κλειδιά, σε οποιαδήποτε θέση του ερωτήματος και αν βρίσκονται, μεταφράζονται.
2. Οι ‘select’ ιδιότητες μεταφράζονται.
3. Οι ιδιότητες των συνδέσμων μεταφράζονται.
4. Οι αριθμητικά περιορισμένες ιδιότητες μεταφράζονται.
5. Δεν εισάγονται επιπλέον αριθμητικοί περιορισμοί ούτε και σύνδεσμοι σε ιδιότητες που δεν είναι κλειδιά.

2.3.4 Στόχος

Η παρούσα διπλωματική εργασία περιλαμβάνει την υλοποίηση του τμήματος του GrouPeer που αφορά την προεπεξεργασία και μετάφραση των ερωτημάτων με χρήση GAV και LAV αντιστοιχήσεων. Υλοποιούμε έναν αλγόριθμο μετάφρασης ερωτημάτων με βάση GAV αντιστοιχήσεις, καθώς και τον αλγόριθμο MiniCon για τη μετάφραση με LAV αντιστοιχήσεις. Η σχεδίαση και υλοποίηση του MiniCon αποτελεί το κυριότερο και μεγαλύτερο κομμάτι της εργασίας. Επίσης υλοποιούμε διαδικασίες προεπεξεργασίας ερωτημάτων, δηλαδή αλγορίθμους οι οποίοι αποκόπτουν από τα αρχικά ερωτήματα τα τμήματα που δεν μπορούν να μεταφραστούν με τις διαθέσιμες αντιστοιχήσεις και εκτιμούν

την ποιότητα των (εν γένει ελαττωμένων) ερωτημάτων που προκύπτουν. Οι διαδικασίες αυτές είναι απαραίτητες για την αποτελεσματική μετάφραση ερωτημάτων σε P2P περιβάλλοντα.

Πιο συγκεκριμένα, στόχος της εργασίας είναι να δημιουργήσει το υποσύστημα του GrouPeer που με βάση ένα δεδομένο ερώτημα και τις διαθέσιμες αντιστοιχήσεις κάθε κόμβου :

1. Θα τρέχει τη διαδικασία της προεπεξεργασίας ξεχωριστά για GAV και LAV αντιστοιχήσεις, και θα παράγει για κάθε τύπο αντιστοιχήσεων το καλύτερο ελαττωμένο ερώτημα που μπορεί να απαντηθεί με βάση αυτές.
2. Θα αξιολογεί τα δύο ελαττωμένα ερωτήματα που προκύπτουν μετρώντας την ομοιότητά τους με το αρχικό, και θα αποφασίζει ποιό από αυτά να απαντήσει.
3. Ανάλογα με το ερώτημα που επιλέχθηκε, θα εκτελεί τον αλγόριθμο μετάφρασης με GAV ή με LAV αντιστοιχήσεις (δηλ. τον MiniCon) και θα μεταφράζει το ελαττωμένο ερώτημα.

3

Ανάλυση και Σχεδίαση

Στο κεφάλαιο αυτό θα περιγράψουμε αναλυτικά τη σχεδίαση των αλγορίθμων που υλοποιήσαμε. Ξεκινάμε από την περιγραφή σχεδίασης των αλγορίθμων μετάφρασης για GAV και LAV αντιστοιχήσεις και με βάση αυτήν περιγράφουμε τη σχεδίαση της προεπεξεργασίας των ερωτημάτων.

Μία γενική γραμμή που ακολουθήσαμε σε όλη την εργασία είναι η εξής. Θεωρήσαμε ότι τόσο τα ερωτήματα όσο και οι αντιστοιχήσεις που διαθέτουμε είναι SPJ (Select – Project – Join) ερωτήματα γραμμένα σε SQL. Χρησιμοποιήσαμε τη γλώσσα SQL παρά το γεγονός ότι τα papers στα οποία βασίστηκε η εργασία χρησιμοποιούν ερωτήματα σε συζευκτική μορφή. Η επιλογή αυτή έγινε για να μπορέσει το (υπο)σύστημα που κατασκευάστηκε να ενσωματωθεί ευκολότερα σε πραγματικά περιβάλλοντα Βάσεων Δεδομένων, τα οποία χρησιμοποιούν την SQL. Η επιλογή της SQL είχε πέρα από το παραπάνω, ορισμένα πλεονεκτήματα και μειονεκτήματα. Σημαντικό πλεονέκτημα όσον αφορά την υλοποίηση ήταν ότι βρήκαμε και χρησιμοποιήσαμε έναν SQL parser που λέγεται ZQL, ο οποίος βοήθησε στην επεξεργασία των ερωτημάτων και αντιστοιχήσεων, και συνεισέφερε στη μετατροπή τους σε συζευκτική μορφή όταν ήταν απαραίτητο. Ωστόσο, λόγω του συνδυασμού της με την SQL, η συζευκτική μορφή που χρησιμοποιήσαμε στην εργασία (στο κομμάτι του MiniCon και της προεπεξεργασίας) δεν είναι η γνωστή μορφή της Datalog (παρ. 2.2.1), αλλά διαφέρει σε σημεία που θα αναλύσουμε παρακάτω. Ας ονομάσουμε την αλλαγμένη αυτή μορφή SQL_CNF (SQL – Conjunctive Normal Form). Η χρήση όμως της SQL και κατ' επέκταση της SQL_CNF, δημιούργησε μία σημαντική δυσκολία. Χρειάστηκε για το κομμάτι της

μετάφρασης με GAV, αλλά κυρίως για το κομμάτι του MiniCon, ο επανασχεδιασμός των αλγορίθμων για να λειτουργούν με ερωτήματα και αντιστοιχήσεις σε SQL ή SQL_CNF. Χρειάστηκε δηλαδή να γίνει μία «μετάφραση» του κάθε βήματος των αλγορίθμων από τη λειτουργία με CNF στη λειτουργία με SQL ή SQL_CNF. Η ανάλυση που ακολουθεί, θα γίνει βέβαια για τη λειτουργία των αλγορίθμων με βάση τη μορφή που επιλέξαμε. Ωστόσο, θα γίνεται αναφορά στο αντίστοιχο τμήμα του paper στο οποίο βασιστήκαμε και στον τρόπο που αυτό «μεταφράζεται», ώστε να αποδεικνύεται η ορθότητα του σχεδιασμού μας.

3.1 SQL_CNF

Η μορφή SQL_CNF έχει δημιουργηθεί συνδυάζοντας τη CNF (Datalog) και την SQL μορφή των ερωτημάτων. Την κατασκευάσαμε προκειμένου να εφαρμόσουμε αλγορίθμους που επεξεργάζονται ερωτήματα που είναι γραμμένα σε CNF, σε αντίστοιχα που είναι σε SQL. Οι αλγόριθμοι αυτοί εργάζονται με τα ερωτήματα χωρισμένα σε subgoals. Συνεπώς, για τη μετατροπή ενός ερωτήματος από SQL σε SQL_CNF, αρχικά το χωρίζουμε σε subgoals παίρνοντας τους πίνακες που εμφανίζονται στο from κομμάτι του ερωτήματος και προσθέτοντας τις ιδιότητες καθενός από αυτούς που εμφανίζονται στο ερώτημα. Η διαφορά είναι ότι στην SQL_CNF μορφή δεν χρησιμοποιούμε μεταβλητές αλλά τα πραγματικά ονόματα των ιδιοτήτων για κάθε πίνακα - subgoal. Με τον τρόπο αυτό προκύπτει το σώμα του SQL_CNF ερωτήματος. Έπειτα κατασκευάζουμε την κεφαλή του κατ' αναλογία με την κεφαλή του αντίστοιχου CNF ερωτήματος, εισάγοντας σε αυτήν τα (πραγματικά) ονόματα των ιδιοτήτων που εμφανίζονται στο select κομμάτι του SQL ερωτήματος. Στη συνέχεια, αποθηκεύουμε τους συνδέσμους του ερωτήματος. Οι σύνδεσμοι στην CNF μορφή παριστάνονται με πολλαπλές εμφανίσεις της ίδιας μεταβλητής ανάμεσα στους subgoals. Όπως προείπαμε όμως, στην SQL_CNF μορφή δε χρησιμοποιούμε μεταβλητές αλλά τα πραγματικά ονόματα των ιδιοτήτων κάθε subgoal. Γι αυτό, αποθηκεύουμε τους συνδέσμους εκτός του ερωτήματος σε μορφή συνόλων εξισωμένων ιδιοτήτων. Τα σύνολα αυτά τα ονομάζουμε Join Sets. Έτσι, κάθε join set περιέχει τα πραγματικά ονόματα των ιδιοτήτων που συνδέονται με την ίδια μεταβλητή στο αντίστοιχο ερώτημα σε CNF. Τέλος, αποθηκεύουμε τους αριθμητικούς περιορισμούς του SQL ερωτήματος επίσης εκτός του ερωτήματος, όπως ακριβώς είναι στο SQL ερώτημα. Αν λάβουμε υπόψη και τα ήδη δημιουργημένα join sets παρατηρούμε ότι αν συνδυάσουμε έναν αριθμητικό περιορισμό με το/τα αντίστοιχο/α join set/s των ιδιοτήτων που περιέχει, η αναπαράστασή μας είναι ισοδύναμη και με την CNF αναπαράσταση των συγκρίσεων και των σταθερών.

Στον πίνακα που ακολουθεί παρουσιάζουμε τις αντιστοιχίες και τις ομοιότητες και διαφορές ανάμεσα στις τρεις μορφές ερωτημάτων: SQL, CNF (Datalog) και SQL_CNF.

SQL	CNF	SQL_CNF
Ονόματα ιδιοτήτων	Μεταβλητές	Ονόματα ιδιοτήτων
Select κομμάτι	Κεφαλή	Κεφαλή
From κομμάτι	Ονόματα των subgoals. Οι subgoals διατηρούν το όνομα του πραγματικού πίνακα σε περίπτωση μετονομασμένων πινάκων.	Ονόματα των subgoals, όπως εμφανίζονται στο from κομμάτι του αντίστοιχου SQL ερωτήματος.
Σύνδεσμοι με '='	Σύνδεσμοι με πολλαπλές εμφανίσεις της ίδιας μεταβλητής	Σύνδεσμοι με δημιουργία join sets
Συγκρίσεις στο where κομμάτι του ερωτήματος	Συγκρίσεις σε comparison subgoals εντός του ερωτήματος	Συγκρίσεις σε comparison subgoals εκτός του ερωτήματος
Σταθερές στο where κομμάτι του ερωτήματος	Σταθερές μέσα στους subgoals του ερωτήματος	Σταθερές σε subgoals τους οποίους θα ονομάζουμε constant subgoals, εκτός του ερωτήματος

Πίνακας 3.1 : Αντιστοιχίες μεταξύ SQL, CNF και SQL_CNF

Δίνουμε εδώ ένα παράδειγμα μετάφρασης ερωτήματος σε SQL_CNF, επεκτείνοντας το παράδειγμα εκείνο που είχαμε εισαγάγει στην ενότητα 2.2.1 για την Datalog (CNF) μορφή των ερωτημάτων.

SQL:

select Κ1_Ιατρός.όνομα, Κ1_Ιατρός.επώνυμο, Κ1_Κλινική.όνομα

from Κ1_Ιατρός, Κ1_Κλινική, Κ1_Υπάγεται

where Κ1_Ιατρός.κωδ_Ιατρού = Κ1_Υπάγεται.κωδ_Ιατρού **and**

Κ1_Κλινική.κωδ_Κλινικής = Κ1_Υπάγεται.κωδ_Κλινικής **and**

Κ1_Κλινική.κτίριο = 'Α' **and**

Κ1_Κλινική.όροφος > 2

Datalog:

Ιατρός_Κλινική(α, β, γ) :- Κ1_Ιατρός(χ, -, α, β, -, -), Κ1_Κλινική(ψ, γ, 'Α', δ, -),

Κ1_Υπάγεται(χ, ψ), δ > 2

SQL_CNF:

Ιατρός_Κλινική(Κ1_Ιατρός.όνομα, Κ1_Ιατρός.επώνυμο, Κ1_Κλινική.όνομα) :-

Κ1_Ιατρός(κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο),

Κ1_Κλινική(κωδ_Κλινικής, όνομα, κτίριο, όροφος, αριθμός_κλινών),

Κ1_Υπάγεται(κωδ_Ιατρού, κωδ_Κλινικής)

- Join Sets:

  { Κ1_Ιατρός.κωδ_Ιατρού, Κ1_Υπάγεται.κωδ_Ιατρού}

  { Κ1_Κλινική.κωδ_Κλινικής, Κ1_Υπάγεται.κωδ_Κλινικής}

- Comparison Predicates:

  Κ1_Κλινική.όροφος > 2

- Constant Predicates:

  Κ1_Κλινική.κτίριο = 'Α'

Η μορφή SQL_CNF δεν χρησιμοποιείται στη σχεδίαση και υλοποίηση του αλγορίθμου για μετάφραση ερωτημάτων με GAV αντιστοιχήσεις, αλλά στα δύο άλλα μέρη της εργασίας, δηλαδή στον αλγόριθμο για μετάφραση με LAV αντιστοιχήσεις και στην προεπεξεργασία των ερωτημάτων.

3.2 Αλγόριθμος μετάφρασης με αντιστοιχήσεις τύπου GAV

Ο αλγόριθμος αυτός εργάζεται με το ερώτημα και τις GAV αντιστοιχήσεις γραμμένα σε SQL. Ο αλγόριθμος του PDMS [HIM+03] εργάζεται με ερωτήματα και αντιστοιχήσεις σε CNF, και χρησιμοποιεί ένα rule – goal tree του οποίου οι κόμβοι είναι subgoals και οι ακμές είναι κανόνες αντιστοιχήσεων. Κάθε subgoal από το σχήμα του αρχικού κόμβου αναλύεται με βάση τις διαθέσιμες γι αυτόν αντιστοιχήσεις σε subgoals από το σχήμα του κόμβου – γείτονα. Για να επιτευχθεί η μετάφραση, οι subgoals αυτοί απλά καταλαμβάνουν τη θέση που είχε στο ερώτημα ο αρχικός. Αν για κάποιον subgoal δεν υπάρχουν αντιστοιχήσεις, τότε αυτός παραλείπεται από τη μετάφραση. Αν πάλι υπάρχουν ελλιπείς αντιστοιχήσεις, αυτές εισάγονται στο μεταφρασμένο ερώτημα και προκύπτει έτσι η καλύτερη δυνατή μετάφραση.

Τον αλγόριθμο αυτόν, τον μετατρέψαμε σε έναν αλγόριθμο που χρησιμοποιεί την SQL μορφή. Με τον πίνακα που ακολουθεί θα εξηγήσουμε βήμα προς βήμα πώς περάσαμε από τον αλγόριθμο που δουλεύει με ερωτήματα και αντιστοιχήσεις σε CNF σε αυτόν που δουλεύει με SQL, αποδεικνύοντας έτσι την ορθότητα του αλγόριθμου που υλοποιήσαμε.

Αλγόριθμος για CNF	Αλγόριθμος για SQL
Εργάζεται με κάθε subgoal του αρχικού ερωτήματος ξεχωριστά. Μεταφράζει τις ιδιότητες που στη θέση τους υπάρχει κάποια μεταβλητή της Datalog. Αντικαθιστά κάθε subgoal με το σώμα της ανάλογης αντιστοίχησης ρυθμίζοντας κατάλληλα τις μεταβλητές που αυτή περιέχει. Με τον τρόπο αυτό μεταφράζει πλήρως τον κάθε subgoal.	Εργάζεται με κάθε ιδιότητα του αρχικού (SQL) ερωτήματος ξεχωριστά, άρα ελέγχει όλες τις ιδιότητες που εμφανίζονται στο ερώτημα. Αντικαθιστά κάθε ιδιότητα με την αντίστοιχή της στο select κομμάτι της ανάλογης αντιστοίχησης. Έπειτα προσθέτει στο μεταφρασμένο ερώτημα το from και το where κομμάτι της αντιστοίχησης (αν δεν έχουν προστεθεί ήδη από τη μετάφραση μίας ιδιότητας που την χρησιμοποιεί). Αυτό γίνεται όμως για όλες τις ιδιότητες κάθε πίνακα, οπότε ο αντίστοιχος subgoal μεταφράζεται πλήρως με χρήση και των τριών κομματιών (select – from – where) της ανάλογης αντιστοίχησης.

Αν για κάποιον subgoal δεν υπάρχει η ανάλογη αντιστοίχηση, τότε αυτός παραλείπεται. Επίσης, δεν μεταφράζονται οι ιδιότητες ενός subgoal για τις οποίες δεν υπάρχει μεταβλητή στην κεφαλή της ανάλογης αντιστοίχησης. Στην περίπτωση αυτή, καταστρέφονται και οι σύνδεσμοι ή οι αριθμητικές συνθήκες στις οποίες οι ιδιότητες αυτές συμμετείχαν στο αρχικό ερώτημα.	Αν για μία ιδιότητα δεν υπάρχει αντίστοιχη στο select κομμάτι της ανάλογης αντιστοίχησης, τότε αυτή παραλείπεται. Εάν αυτή συμμετέχει σε ένα σύνδεσμο ή σε μία αριθμητική συνθήκη, τότε αυτός/ή παραλείπεται ολόκληρη.
Εάν ένας subgoal εμφανίζεται στο αρχικό ερώτημα δύο ή περισσότερες φορές με το ίδιο όνομα πίνακα, το μεταφρασμένο ερώτημα περιέχει ισάριθμες φορές το σώμα της ανάλογης αντιστοίχησης.	Εάν ένας πίνακας εμφανίζεται (μετονομασμένος) δύο ή περισσότερες φορές σε ένα (SQL) ερώτημα, τότε πριν μεταφράσουμε τις ιδιότητες που ανήκουν στους μετονομασμένους πίνακες, βρίσκουμε την αντιστοίχηση που αναλογεί στον πραγματικό πίνακα και μετονομάζουμε όλους τους πίνακες σε αυτήν προσθέτοντας στο όνομά τους τη συμβολοσειρά «_*» , όπου * το όνομα του μετονομασμένου πίνακα στον οποίο ανήκει η ιδιότητα που θέλουμε να μεταφράσουμε. Προκύπτει έτσι για τον μετονομασμένο πίνακα αυτόν μία νέα αντιστοίχηση που θα την ονομάσουμε mapping_Ren.

Πίνακας 3.2 : Αντιστοιχίες μεταξύ αλγορίθμων μετάφρασης με GAV αντιστοιχήσεις

Ο ψευδοκώδικας του νέου αλγορίθμου είναι ο εξής :

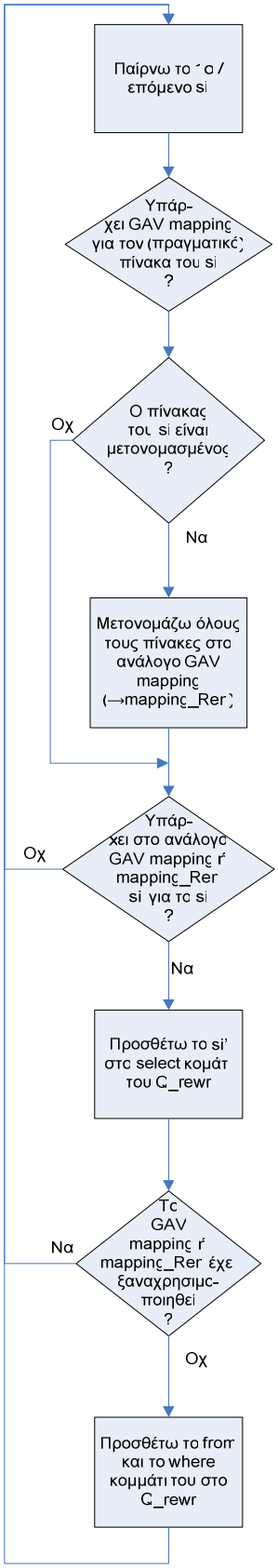
```
procedure GAV_Translation (Q, GAV_set)
  For each attribute T.f (Table.field) of Q's select part
    If there is a GAV mapping M for Real(T) then
      If T = Real(T)
        If T.f appears as alias in the select part of M then
```

```
          add the correspondent select item to Q_rewr's select part,
          If M has never been used before then
            add M's from and where part to Q_rewr
        else
          continue
      else /* T is renamed */
        Create M_Ren from M
        If T.f appears as alias in the select part of M_Ren then
          add the correspondent select item to Q_rewr's select part,
          If M_Ren has never been used before then
            add M_Ren's from and where part to Q_rewr
        else
          continue
    else
      continue
For each join or condition of Q's where part
  If both its sides T1.f1, T1.f2 can be translated
    translate T1.f1 and T1.f2 exactly as above
      (without adding them to Q_rewr's select part), and
    add the translated join or condition to Q_rewr's where part
```

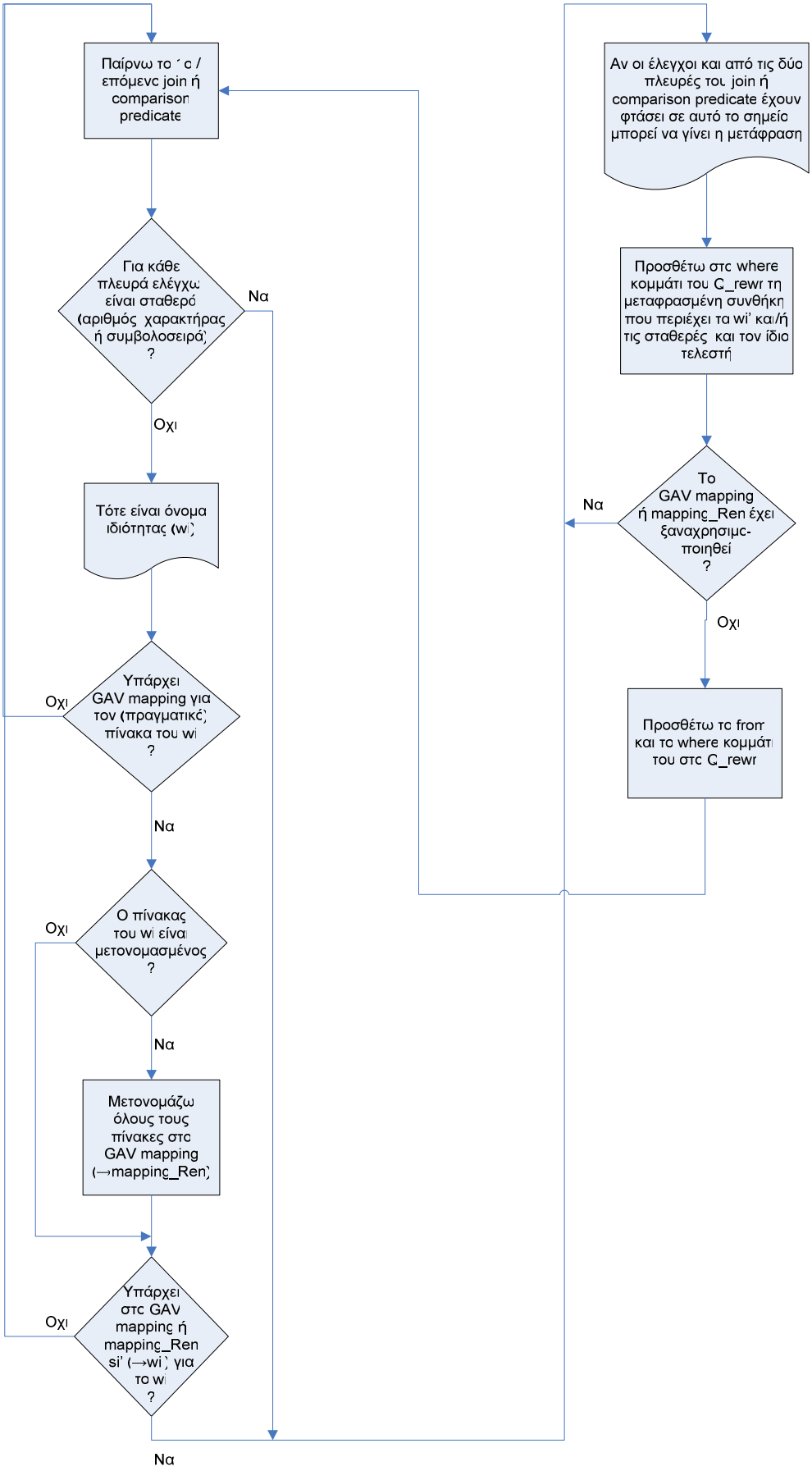
Αλγόριθμος 3.1 : Μετάφραση ερωτήματος με αντιστοιχήσεις τύπου GAV

Στα Σχήματα 3.1 και 3.2 φαίνεται το διάγραμμα ροής του αλγορίθμου. Το πρώτο από αυτά περιγράφει τη μετάφραση του select κομματιού του αρχικού ερωτήματος ενώ το δεύτερο τη μετάφραση του where κομματιού. Συμβολίζουμε με :

- si : Select Item, δηλαδή ιδιότητα που εμφανίζεται στο select κομμάτι του ερωτήματος.
- wi : Where Item, δηλαδή ιδιότητα που εμφανίζεται στο where κομμάτι του ερωτήματος.
- si' : Select Item στην αντιστοίχηση, αντίστοιχο του si στο ερώτημα.
- wi' : Select Item στην αντιστοίχηση, αντίστοιχο του wi στο ερώτημα.
- Q_rewr (Query Rewritten) : Το μεταφρασμένο ερώτημα.
- mapping_Ren (mapping Renamed) : Η αντιστοίχηση με μετονομασμένους τους πίνακές της. Χρησιμοποιείται για τη μετάφραση κάθε ιδιότητας του ερωτήματος που ανήκει σε ένα μετονομασμένο πίνακα.



Σχήμα 3.1 : Αλγόριθμος μετάφρασης με GAV αντιστοιχήσεις – select κομμάτι



Σχήμα 3.2 : Αλγόριθμος μετάφρασης με GAV αντιστοιχήσεις – where κομμάτι

3.3 Αλγόριθμος MiniCon

3.3.1 Βασική δομή του MiniCon

Όπως και για τον αλγόριθμο για μετάφραση με GAV αντιστοιχήσεις, θα ξεκινήσουμε την ανάλυση παρουσιάζοντας την αρχική ιδέα του αλγορίθμου MiniCon για επεξεργασία ερωτημάτων σε CNF, όπως αυτή δίδεται στο [PH01].

Αρχικά, δίνουμε κάποιους ορισμούς και συμβολισμούς που θα χρησιμοποιηθούν στην ανάλυση του αλγορίθμου :

- Με Q θα συμβολίζουμε το αρχικό ερώτημα σε CNF και με V_i τις δεδομένες όψεις – αντιστοιχήσεις.
- Με Vars(Q) θα συμβολίζουμε τις μεταβλητές του ερωτήματος Q και με Vars(V) τις μεταβλητές της αντιστοίχησης V.
- Ονομάζουμε διακεκριμένες (distinguished) τις μεταβλητές ενός ερωτήματος (ή μίας αντιστοίχησης), τις μεταβλητές που βρίσκονται στην κεφαλή του. Τις υπόλοιπες τις ονομάζουμε υπαρξιακές (existential).
- Δεδομένης μίας αντιστοιχίας T από τις Vars(Q) στις Vars(V), θα λέμε ότι ένας subgoal g_V της V 'καλύπτει' έναν subgoal g του Q, εάν $T(g) = g_V$.
- Ορίζουμε ως ομομορφισμό στην κεφαλή (head homomorphism) μίας αντιστοίχησης V, μία αντιστοιχία που την ονομάζουμε H_c, από τις Vars(V) στις Vars(V) η οποία αντιστοιχίζει τις υπαρξιακές μεταβλητές στον εαυτό τους, αλλά μπορεί να εξισώνει κάποιες από τις διακεκριμένες μεταβλητές.
- Ονομάζουμε F_c μία μερική αντιστοιχία από τις Vars(Q) στο H_c(Vars(V)). Η αντιστοιχία αυτή πρέπει να είναι ένα προς ένα.
- Ορίζουμε ένα MCD (MiniCon Desciptor) για ένα ερώτημα Q και μία αντιστοίχηση V, ως μία πλειάδα που έχει τη μορφή: { H_c, $V(Y)_c$, F_c, G_c }, όπου $V(Y)_c$ είναι η αντιστοίχηση που προκύπτει αν εφαρμόσουμε τον ομομορφισμό H_c στη V, και G_c ένα σύνολο από subgoals του Q, οι οποίοι καλύπτονται από κάποιον subgoal στο H_c(V) χρησιμοποιώντας την αντιστοιχία μεταβλητών F_c. Σημειώνουμε ότι το G_c δεν περιέχει απαραίτητα όλους τους subgoals του Q με αυτήν την ιδιότητα. Παρατηρούμε επίσης, ότι το $V(Y)_c$ ορίζεται μοναδικά από την αντιστοίχηση V και τον ομομορφισμό H_c. Το συμπεριλαμβάνουμε όμως εντός των MCDs για να διευκολύνουμε την ανάλυση. Τέλος, τονίζουμε ότι η κατασκευή ενός MCD, γίνεται μόνο για τον ελάχιστο (σε πλήθος εξισωμένων μεταβλητών) δυνατό ομομορφισμό που δίνει ένα μη κενό σύνολο G_c.

Το πρώτο βήμα του αλγορίθμου είναι η κατασκευή των MCDs, δεδομένου του ερωτήματος και των διαθέσιμων LAV αντιστοιχήσεων. Η κατασκευή των MCDs στηρίζεται στην ακόλουθη ιδιότητα:

Ιδιότητα 1:

Αν το C είναι ένα MCD για το ερώτημα Q και την αντιστοίχηση V, τότε το C μπορεί να χρησιμοποιηθεί σε μία μετάφραση του Q χωρίς επαναλαμβανόμενους (περιττούς) subgoals αν ισχύουν οι ακόλουθες συνθήκες:

1. Για κάθε διακεκριμένη μεταβλητή x του Q, η οποία βρίσκεται στην περιοχή της F_c (δηλαδή αντιστοιχίζεται μέσω της αντιστοιχίας μεταβλητών F_c σε κάποια μεταβλητή $F_c(x)$ στο $H_c(V)$), πρέπει και η $F_c(x)$ να είναι διακεκριμένη στο $H_c(V)$.
2. Για κάθε υπαρξιακή μεταβλητή $F_c(x)$ στο $H_c(V)$, αντίστοιχη μίας μεταβλητή x στο Q, πρέπει για κάθε subgoal g του Q που την περιέχει να ισχύουν οι συνθήκες: (1) Όλες οι μεταβλητές του g να είναι στην περιοχή της F_c, και (2) Ο subgoal που προκύπτει αν στον g αντικαταστήσουμε όλες τις μεταβλητές με τις αντίστοιχές τους μέσω της F_c, ο subgoal που προκύπτει να εμφανίζεται στο $H_c(V)$, δηλαδή ο $F_c(g)$ να ανήκει στο $H_c(V)$.

Η Ιδιότητα 1.1 (όμοια υπάρχει και στον αλγόριθμο Bucket που παρουσιάσαμε στην παράγραφο 2.2.3.2), αναφέρεται στις μεταβλητές – ιδιότητες που βρίσκονται στην κεφαλή του ερωτήματος Q. Αυτές είναι οι ιδιότητες για τις οποίες το ερώτημα Q πρέπει να επιστρέψει αποτελέσματα. Συνεπώς, οι αντίστοιχές τους μέσω κάποιας αντιστοιχίας F_c θα πρέπει επίσης να βρίσκονται στην κεφαλή του μεταφρασμένου ερωτήματος Q_rewr. Όμως αφού βρίσκονται στην κεφαλή του Q_rewr, πρέπει επίσης να βρίσκονται και στο σώμα του. Αλλά το σώμα του Q_rewr αποτελείται μόνο από subgoals που είναι κεφαλές αντιστοιχήσεων (και ίσως επιπλέον κάποιους comparison subgoals). Άρα, οι ιδιότητες αυτές πρέπει να εμφανίζονται στην κεφαλή των αντιστοιχήσεων που μεταφράζουν τους subgoals του Q.

Η Ιδιότητα 1.2 τώρα, εξασφαλίζει ότι οι σύνδεσμοι που εμφανίζονται ανάμεσα σε κάποιους subgoals στο ερώτημα Q, θα μπορέσουν να μεταφραστούν από τις χρησιμοποιούμενες αντιστοιχήσεις. Όπως είναι γνωστό, ένας σύνδεσμος στην CNF μορφή αναπαρίσταται από πολλαπλές εμφανίσεις της ίδιας μεταβλητής – ας τη συμβολίσουμε με j – στα σημεία που αντιστοιχούν στις συνδεδεμένες ιδιότητες. Για να μπορεί να μεταφραστεί ένας σύνδεσμος του ερωτήματος Q πρέπει για κάθε subgoal g_j που συμμετέχει σε αυτό, να ισχύει το εξής:

Αν (1) στην αντιστοίχηση V που μεταφράζει το g_j η μεταβλητή που αντιστοιχεί στην j είναι διακεκριμένη, τότε ο σύνδεσμος γι' αυτόν τον subgoal μπορεί να αναδημιουργηθεί αργότερα χάρις τη μεταβλητή αυτή. Αν όμως η εν λόγω μεταβλητή είναι υπαρξιακή στο V (2), τότε ο

σύνδεσμος δεν μπορεί να αναδημιουργηθεί, άρα η μόνη περίπτωση να μπορεί να μεταφραστεί είναι αν υπάρχει ήδη μέσα στο V. Αυτό σημαίνει ότι το V πρέπει να περιέχει και όλους τους άλλους g_j's που συμμετέχουν στον σύνδεσμο και να έχει μία κοινή μεταβλητή στις αντίστοιχες θέσεις – ιδιότητες. Εφόσον όμως η μεταβλητή αυτή είναι κοινή, δεν μπορεί για κάποιον g_j να είναι διακεκριμένη, είναι για όλα τα g_j υπαρξιακή.

Καταλήγουμε έτσι ότι ένας σύνδεσμος του Q μπορεί να μεταφραστεί σε δύο περιπτώσεις. Η πρώτη είναι όταν όλοι οι subgoals που μετέχουν σε αυτό μεταφράζονται από αντιστοιχήσεις (εν γένει διαφορετικές) στις οποίες η μεταβλητή που αντιστοιχεί στη μεταβλητή j του Q, είναι διακεκριμένη. Η δεύτερη είναι όταν όλοι οι subgoals μπορούν να μεταφραστούν από την ίδια αντιστοίχηση, στην οποία υπάρχει ήδη ο ζητούμενος σύνδεσμος. Η μεταβλητή που εκτελεί τον σύνδεσμο μέσα στην αντιστοίχηση μπορεί να είναι υπαρξιακή ή διακεκριμένη (αν είναι διακεκριμένη εμπίπτουμε στην πρώτη περίπτωση).

Η ιδιότητα 1.2 λοιπόν εξασφαλίζει ότι αν σε μία αντιστοίχηση V, περιέχεται ένας subgoal που αντιστοιχεί σε έναν g_j του Q, ο οποίος στη θέση της μεταβλητής j (x) έχει μία υπαρξιακή μεταβλητή ($F_c(x)$), τότε το V περιέχει και όλους τους άλλους subgoals g_j. Συνεπώς, το V μπορεί να χρησιμοποιηθεί για μετάφραση του συνδέσμου ανάμεσα στους g_j.

Με βάση τα παραπάνω, σχεδιάζεται η συνάρτηση που κατασκευάζει τα MCDs. Την παραθέτουμε εδώ όπως εμφανίζεται στο paper [PH01] για τον MiniCon:

```
procedure formMCDs(Q,V_set) /* Original */
  C_set = Ø
  For each subgoal g in Q
    For each mapping V in V_set
      For each subgoal u in V
        Let H be the least restrictive head Homomorphism on V
          s.t. there exists a mapping F, s.t. F(g) = H(u)
        If F, H exist then add to C_set any new MCD C
          that can be constructed where:
            (a) F_c/H_c is an extension of F/H
            (b) Gc is the minimal subset of subgoals of Q s.t.
                  F_c, H_c, G_c satisfy Property 1
            (c) It is not possible to extend F and H to F_c' and H_c'
                  s.t. (b) is satisfied and G_c', as defined in (b),
                  is a subset of G_c
  Return C_set.
```

Αλγόριθμος 3.2 (α) : Διαδικασία formMCDs για CNF – Αρχική μορφή

Τον Αλγόριθμο 3.2 (α) τον επανασχεδιάσαμε ως εξής:

```
procedure formMCDs(Q,V_set) /* CNF */
  C_set = Ø
  For each subgoal g in Q
    For each mapping V in V_set
      For each subgoal u in V that has the same table name as g
        Let H be the least restrictive head Homomorphism on V
          s.t. there exists a variable mapping F, s.t. F(g) = H(u)
        Let G = {g}
        Let (MCD) C = [H, H(V), F, G]
        Repeat
          If (C satisfies Property 1.1) then
            If (C satisfies Property 1.2.1) then
              If (C satisfies Property 1.2.2) then
                add C to C_set, break the loop
              else
                extend F to F_c and G to G_c, let C = [H_c, H_c(V), F_c, G_c]
            else
              extend H to H_c, let C = [H_c, H_c(V), F_c, G_c]
          else
            C is rejected
        Until (No extension could be made
               in order to satisfy Property1)
  Return C_set.
```

Αλγόριθμος 3.2 (β) : Διαδικασία formMCDs για CNF – Επανασχεδιασμένη

Θα δείξουμε ότι υπάρχει τρόπος να γνωρίζουμε τι τύπου επέκταση (στο H_c ή στο F_c και το G_c) πρέπει να κάνουμε προκειμένου να ικανοποιείται (πιθανόν) η *Ιδιότητα1*. Επίσης, η επέκταση (κάθε είδους) γίνεται σταδιακά, ώστε να προκύψουν τα ελάχιστα H_c, ή F_c και G_c αντίστοιχα. Αυτός είναι και ο λόγος που καθιστά τους Αλγορίθμους 3.2 (α) και 3.2 (β) ισοδύναμους. Γιατί ο συγκεκριμένος και ελάχιστος τρόπος με τον οποίο κατασκευάζουμε τα MCDs με τον δεύτερο αλγόριθμο, συνεπάγεται ότι αυτά ικανοποιούν και τις συνθήκες (b), (c) του πρώτου αλγορίθμου.

Στο σημείο αυτό θα αναλύσουμε τον τρόπο με τον οποίο επιλέγουμε τον τύπο της επέκτασης που πρέπει να εκτελέσουμε σε κάθε βήμα του αλγορίθμου, για συγκεκριμένα g και u. Η

επιλογή βασίζεται στις απαιτήσεις της *Ιδιότητας1*, και αποτελεί σημείο – κλειδί για τη σωστή κατασκευή των MCDs. Ο αλγόριθμος λοιπόν λειτουργεί ως εξής:

Για κάθε subgoal g του Q, ελέγχει σε κάθε View V αν υπάρχει subgoal u που να έχει όνομα πίνακα όμοιο με του g. Έστω ότι βρίσκουμε έναν τέτοιο u. Ζητάμε τώρα τον ελάχιστο ομομορφισμό H, τ.ω. να υπάρχει μία αντιστοιχία μεταβλητών F, από τον g στον u. Για να υπάρχει όμως τέτοια αντιστοιχία, η οποία αντιστοιχίζει κάθε μεταβλητή (όχι κάθε ιδιότητα) του g με μία μόνο μεταβλητή του u, πρέπει οι μεταβλητές που εμφανίζονται σε περισσότερες από μία θέσεις εντός του g, να εμφανίζονται και στις αντίστοιχες θέσεις στον u. Δηλαδή απαιτούμε εάν υπάρχει ένας σύνδεσμος ανάμεσα σε ιδιότητες του g, αυτός να εμφανίζεται και στον u. Ο ελάχιστος H που πετυχαίνει το ζητούμενο είναι μοναδικός, και εξισώνει (δηλ. εξομοιώνει) τις μεταβλητές των ιδιοτήτων που είναι όμοιες στον g, και μόνο αυτές. Αφού εφαρμόσουμε τον H στο V, μπορούμε να πάρουμε και την αντιστοιχία F, ανάμεσα στις μεταβλητές του g και του H(u). Τώρα, προσθέτουμε στο σύνολο G (που περιέχει τους subgoals του Q που περιλαμβάνει το αντίστοιχο MCD) και κατασκευάζουμε το αρχικό MCD C = [H, H(V), F, G] που περιέχει το ελάχιστο σύνολο από subgoals του Q που ίσως ικανοποιούν την *Ιδιότητα1*. Ελέγχουμε τώρα εάν η *Ιδιότητα1* για το C ικανοποιείται. Εάν ναι, τότε το προσθέτουμε στο σύνολο C_{set} που περιέχει όλα τα αποδεκτά MCDs. Εάν όχι, καλούμαστε να αποφασίσουμε τι τύπου επέκταση θα κάνουμε στο H ή στο F και στο G, προκειμένου το νέο C που θα προκύψει να την ικανοποιεί. Ο τύπος και το εύρος της ελάχιστης επέκτασης (εάν υπάρχει), που απαιτείται για να ικανοποιείται η *Ιδιότητα1*, προσδιορίζεται από τους εξής ελέγχους, που αντιστοιχούν στις υποϊδιότητές της:

1.1 : Ελέγχουμε εάν ικανοποιείται η Ιδιότητα 1.1, δηλαδή εάν για κάθε μεταβλητή x του Q που είναι διακεκριμένη στο Q, και ανήκει στην περιοχή της αντιστοιχίας F_c, είναι και η μεταβλητή $F_c(x)$ διακεκριμένη στο $H_c(V)$. Εάν η ιδιότητα ικανοποιείται, τότε συνεχίζουμε με τον έλεγχο για την 1.2. Εάν όμως όχι, τότε το MCD C απορρίπτεται. Αυτό συμβαίνει διότι δεν υπάρχει τρόπος, χρησιμοποιώντας επεκτάσεις των F_c και H_c, να μετατρέψουμε τη μεταβλητή που προκαλεί το πρόβλημα από υπαρξιακή σε διακεκριμένη μέσα στην $H_c(V)$.

1.2 : Αρχικά ελέγχουμε εάν ικανοποιείται η Ιδιότητα 1.2.1. Διατρέχουμε μία προς μία τις μεταβλητές του $H_c(V)$ που ανήκουν στην περιοχή (των εικόνων των μεταβλητών) του F_c. Για καθεμία από αυτές που είναι υπαρξιακή στο $H_c(V)$, βρίσκουμε τους subgoals του Q που περιέχουν την αντίστοιχή της. Για καθέναν από αυτούς τους subgoals, απαιτούμε όλες οι μεταβλητές του να ανήκουν στην περιοχή του F_c. Ας υποθέσουμε λοιπόν ότι κατά τον έλεγχο της Ιδιότητας 1.2.1, βρίσκουμε έναν subgoal του Q, του οποίου δεν ανήκουν όλες οι μεταβλητές στο F_c. Η μόνη επέκταση που μπορούμε να κάνουμε για να λυθεί το πρόβλημα, είναι να επεκτείνουμε την αντιστοιχία μεταβλητών F_c. Η ελάχιστη επέκταση

είναι βέβαια το να προσθέσουμε σε αυτήν τις μεταβλητές του Q που ανήκουν στον συγκεκριμένο subgoal αλλά λείπουν από το F_c. Παρατηρούμε λοιπόν ότι ο τύπος και το εύρος της απαιτούμενης επέκτασης για τη συγκεκριμένη περίπτωση ορίζονται μονοσήμαντα. Παράλληλα με την επέκταση του F_c, επεκτείνουμε επίσης μονοσήμαντα το Gc, προσθέτοντας σε αυτό τον subgoal του Q ο οποίος πλέον δεν δημιουργεί πρόβλημα.

Αφού ελέγξουμε για μία συγκεκριμένη αντιστοίχηση V εάν ικανοποιείται η Ιδιότητα 1.2.1, και κάνουμε την επέκταση του F_c και του G_c που αυτή ορίζει, ελέγχουμε εάν ικανοποιείται η Ιδιότητα 1.2.2 . Ελέγχουμε δηλαδή, εάν όλοι οι subgoals του Q που εμφανίζονται στο G_c, εμφανίζονται επίσης, μέσω των F_c και H_c, και στο $H_c(V)$. Η απαίτηση αυτή δεν αναφέρεται μόνο στα ονόματα των subgoals του G_c, αλλά και στους συνδέσμους που εμφανίζονται σε αυτούς. Συνεπώς, κατά τον έλεγχο της Ιδιότητας 1.2.2 μπορούν να εμφανιστούν τρεις περιπτώσεις. Η πρώτη είναι όλα τα ονόματα και οι σύνδεσμοι των subgoals του G_c να εμφανίζονται και στο $H_c(V)$, οπότε η ιδιότητα ικανοποιείται. Η δεύτερη είναι κάποιος subgoal να μην υπάρχει στο $H_c(V)$. Στην περίπτωση αυτή, η ιδιότητα δεν μπορεί να ικανοποιηθεί αφού δεν έχουμε δυνατότητα να αλλάζουμε το εσωτερικό των αντιστοιχήσεων προσθέτοντας νέους subgoals σε αυτά. Η τρίτη περίπτωση είναι κάποιος σύνδεσμος που υπάρχει ανάμεσα στους subgoals του G_c στο Q, να μην εμφανίζεται στους subgoals του $H_c(V)$. Όπως και παραπάνω, δεν μπορούμε να δημιουργήσουμε αυτόν τον σύνδεσμο τροποποιώντας εσωτερικά την αντιστοίχηση V. Ο μόνος τρόπος να μπορέσουμε να τον εμφανίσουμε είναι μέσω ενός νέου ομομορφισμού στο $H_c(V)$, δηλαδή μέσω της επέκτασης του H_c. Η μέθοδος αυτή μπορεί να εφαρμοστεί μόνο εάν ο σύνδεσμος που απουσιάζει συνδέει (ή ακριβέστερα, αν υπήρχε θα συνέδεε) διακεκριμένες μεταβλητές στην $H_c(V)$, οπότε ένας ομομορφισμός που μπορεί και τις εξισώνει, τον αναδημιουργεί. Και σε αυτήν την περίπτωση, το είδος και το εύρος της επέκτασης που απαιτείται καθορίζεται με μοναδικό τρόπο. Στην περίπτωση που μία από τις δύο μεταβλητές είναι υπαρξιακή στο $H_c(V)$, ο σύνδεσμος δεν μπορεί να δημιουργηθεί, και η Ιδιότητα 1.2.2 δεν ικανοποιείται.

Περιγράψαμε έως τώρα το πώς λειτουργεί ο αλγόριθμος κατασκευής MCDs, όπως εμείς τον επανασχεδιάσαμε. Ωστόσο, ο αλγόριθμος αυτός βασίζεται στην CNF μορφή των ερωτημάτων και των αντιστοιχήσεων. Ο αλγόριθμος της διεργασίας κατασκευής των MCDs που τελικά υλοποιήσαμε, δουλεύει με ερωτήματα και αντιστοιχήσεις σε SQL_CNF μορφή. Στον πίνακα 3.2 που ακολουθεί, φαίνονται ορισμένες βασικές διαφορές μεταξύ των δύο αλγορίθμων. Σημειώνουμε ότι το πρώτο βήμα κατά την εκτέλεση του MiniCon όπως τον υλοποιήσαμε, είναι η μετατροπή των ερωτημάτων και των αντιστοιχήσεων σε SQL_CNF.

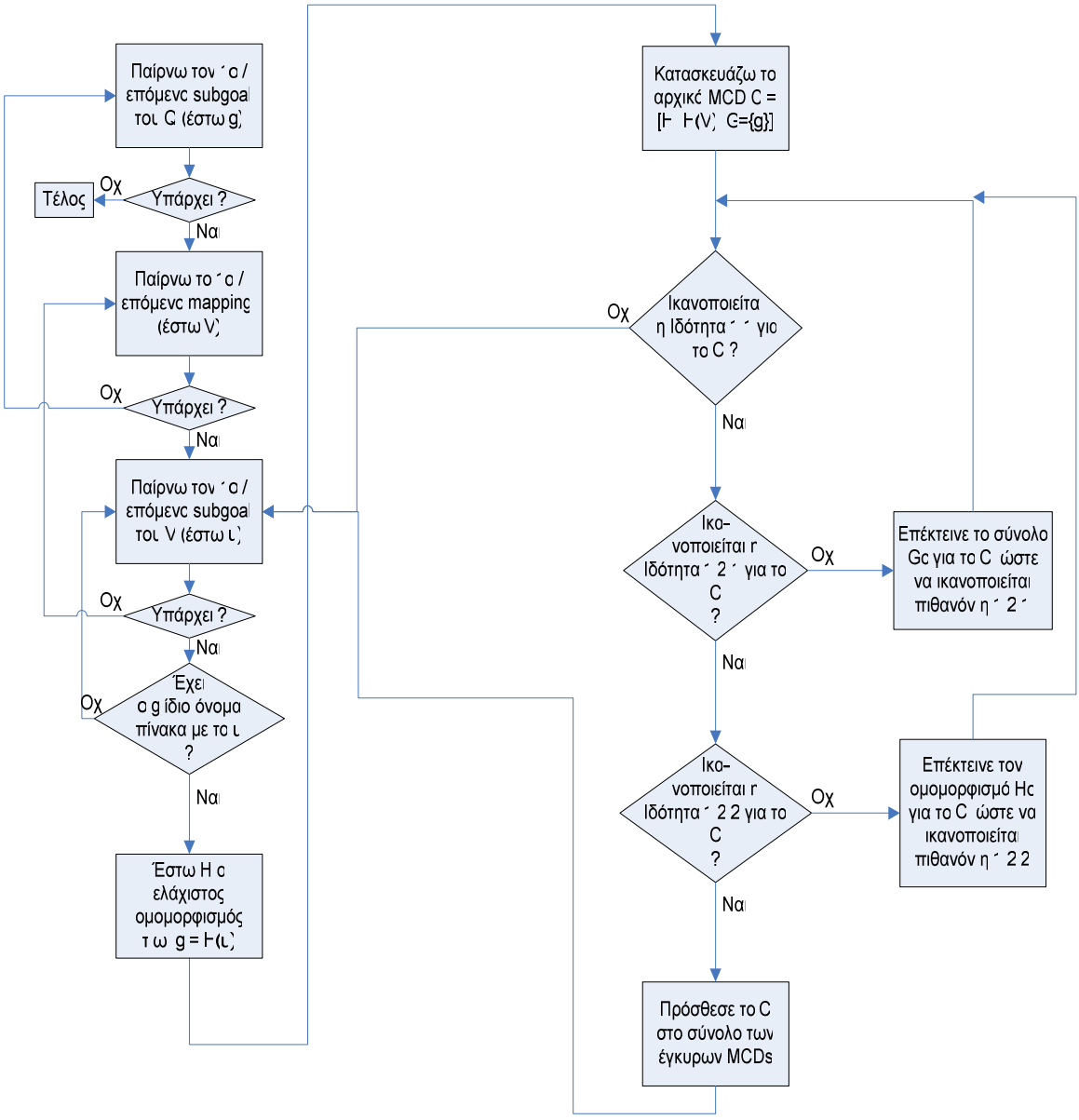
Αλγόριθμος formMCDs για CNF	Αλγόριθμος formMCDs για SQL_CNF
Η αναπαράσταση CNF χρησιμοποιεί μεταβλητές στη θέση των ιδιοτήτων των subgoals. Επειδή τα ονόματα των μεταβλητών είναι αυθαίρετα, ο αλγόριθμος έχει ανάγκη από αντιστοιχίες μεταβλητών ανάμεσα σε subgoals του ερωτήματος και subgoals των αντιστοιχήσεων.	Η SQL_CNF διατηρεί τα ονόματα των ιδιοτήτων, οπότε όταν δύο subgoals έχουν ίδιο όνομα πίνακα και ίδια ονόματα ιδιοτήτων, η αντιστοιχία μεταξύ των ιδιοτήτων τους είναι η προφανής.
Ένας ομομορφισμός σε κάποια αντιστοίχηση αναπαρίσταται σαν εξομοίωση μεταβλητών της κεφαλής της.	Ένας ομομορφισμός σε κάποια αντιστοίχηση επιτυγχάνεται με την προσθήκη ενός επιπλέον συνδέσμου ανάμεσα σε ιδιότητες της κεφαλής της.
Για κάθε ζεύγος subgoals g και u με το ίδιο όνομα πίνακα, ο αλγόριθμος ελέγχει εάν υπάρχει ομομορφισμός H και αντιστοιχία μεταβλητών F τ.ω. F(g) = H(u).	Για κάθε ζεύγος subgoals g και u με το ίδιο όνομα πίνακα, ο αλγόριθμος ελέγχει εάν οι σύνδεσμοι που εμφανίζονται εντός του g εμφανίζονται επίσης εντός του u. Δεν ασχολείται με αντιστοιχίες μεταβλητών διότι χρησιμοποιεί πραγματικά ονόματα ιδιοτήτων.
Ένα MCD έχει τη μορφή C = [H, H(V), F, G]	Ένα MCD έχει τη μορφή C = [H, H(V), G]

Πίνακας 3.3 : Αντιστοιχίες των διαδικασιών formMCDs για CNF και για SQL_CNF

Ακολουθούν ο ψευδικώδικας και το διάγραμμα ροής του αλγορίθμου.

```
Procedure formMCDs(Q,V_set) /* SQL_CNF */
  C_set = Ø
  For each subgoal g in Q
    For each mapping V in V_set
      For each subgoal u in V that has the same table name as g
        Let H be the least restrictive head Homomorphism on V
          s.t. g = H(u) (all joins that appear in g must appear in u)
        Let G = {g}
        Let (MCD) C = [H, H(V), G]
        Repeat
          If (C satisfies Property 1.1) then
            If (C satisfies Property 1.2.1) then
              If (C satisfies Property 1.2.2) then
                add C to C_set, break the loop
              else
                extend F to F_c and G to G_c, let C = [H_c, H_c(V), G_c]
            else
              extend H to H_c, let C = [H_c, H_c(V), G_c]
          else
            C is rejected
        Until (No extension could be made
               in order to satisfy Property1)
  Return C_set.
```

Αλγόριθμος 3.2 (γ) : Διαδικασία formMCDs για SQL_CNF



Σχήμα 3.3 : Διαδικασία formMCDs για SQL_CNF

Συνεχίζουμε με την ανάλυση της δεύτερης κύριας διεργασίας διαδικασίας του MiniCon, που ευθύνεται για την συνένωση των MCDs για την παραγωγή του τελικού ερωτήματος. Η λειτουργία της διεργασίας αυτής βασίζεται στην ακόλουθη ιδιότητα :

Ιδιότητα 2:

Δεδομένου ενός ερωτήματος Q, ένα σύνολο από όψεις (αντιστοιχήσεις) Vset, και ένα σύνολο από MCDs Cset, οι μόνοι συνδυασμοί MCDs που δίνουν σαν αποτέλεσμα ένα μεταφρασμένο ερώτημα που δεν περιέχει επαναλήψεις, έχουν τη μορφή $C_1,\ldots,C_k$, με:

1. $Gc_1 \cup Gc_2 \ldots \cup Gc_k = subgoals(Q)$, και
2. Για κάθε $i \neq j$, $Gc_i \cap Gc_j = \emptyset$

Η απόδειξη της ιδιότητας απορρέει από την απόδειξη της ορθότητας του MiniCon που μπορεί να αναζητηθεί στο [PH01].

Η διεργασία που συνενώνει τα ήδη δημιουργημένα MCDs με τρόπο ώστε να προκύπτουν μεταφρασμένα ερωτήματα που δεν περιέχουν επαναλήψεις, λέγεται combineMCDs. Ο ψευδοκώδικάς της στο [PH01] είναι ο αλγόριθμος 3.3 (α) που ακολουθεί.

Σημειώνουμε ότι μία αντιστοιχία μεταβλητών F_c ενός MCD C, μπορεί να αντιστοιχίζει ένα σύνολο από μεταβλητές $x_1, \ldots, x_n$ του ερωτήματος Q στην ίδια μεταβλητή y του $H_c(V)$. Για κάθε σύνολο τέτοιων μεταβλητών ορίζουμε μία μεταβλητή που το αντιπροσωπεύει. Η μεταβλητή αυτή μπορεί να επιλεχθεί τυχαία ανάμεσα στις $x_1, \ldots, x_n$, όμως αν κάποια από αυτές είναι διακεκριμένη, αυτή προτιμάται. Έτσι για κάθε μεταβλητή του Q ορίζουμε ως $EC_{Fc}(x)$ τη μεταβλητή που αντιπροσωπεύει το σύνολο στο οποίο ανήκει η x. Αν η x δεν εμφανίζεται στο Q, ορίζουμε το $EC_{Fc}(x)$ να είναι η ίδια η x.

```
procedure combineMCDs(Cset) /* Original */
  /* Cset are MCDs formed by the formMCDs algorithm */
  /* Each MCD has the form [Hc, V(Y), Fc, Gc, ECc] */
  Given a set of MCDs C1,… ,Cn ,we define EC on Vars(Q) as follows:
  If for i≠j, ECFi(x)≠ ECFj(x), define EC(x) to be one of them
    arbitrarily but consistently across all y for which
    ECFi(y) = ECFi(x).
  Let Answer = Ø.
  For every subset C1,… ,Cn of Cset s.t.
    Gc1 U Gc2 … U Gcn = subgoals(Q) , and
    for every i≠j, Gci ∩ Gcj = Ø
      Define a variable mapping Ψi on the Yi's as follows:
      If there exists a variable x in Q s.t Fi(x) = y
        Ψi(y) = x
      else
        Ψi(y) is a fresh copy of y
      Create the conjunctive rewriting:
        Q'(EC(X)) :- Vc1(EC(Ψ1(Yc1))),… ,(EC(Ψn(Ycn)))
      Add Q' to Answer.
  Return Answer.
```

Αλγόριθμος 3.3 (α) : Διαδικασία combineMCDs για CNF – Αρχική μορφή

Παραθέτουμε τώρα στον πίνακα 3.4 τις παρατηρήσεις με βάση τις οποίες μετατρέψαμε τον αλγόριθμο 3.3 (α) σε εκείνον που υλοποιήσαμε, ο οποίος δουλεύει με την SQL_CNF.

Αλγόριθμος combineMCDs για CNF	Αλγόριθμος combineMCDs για SQL_CNF
Στο πρώτο βήμα ο αλγόριθμος θέτει κοινά ονόματα μεταβλητών ανάμεσα στα MCDs που έχουν δημιουργηθεί. Ο τρόπος δημιουργίας υποσυνόλων από MCDs που να ικανοποιούν την *Ιδιότητα 2* είναι κοινός και στους δύο αλγορίθμους.	Ο αλγόριθμος δεν χρησιμοποιεί μεταβλητές. Τα ονόματα των αντίστοιχων ιδιοτήτων είναι τα ίδια για όλα τα MCDs.
Ο αλγόριθμος, με τις αντιστοιχίες μεταβλητών Ψ_i, δημιουργεί στο τελικό ερώτημα τους συνδέσμους που υπήρχαν στο Q.	Ο αλγόριθμος ελέγχει έναν προς έναν τους συνδέσμους του αρχικού ερωτήματος και δημιουργεί όσους από αυτούς μπορούν να δημιουργηθούν, και στο τελικό ερώτημα. Όσοι δεν μπορούν να δημιουργηθούν στο τελικό ερώτημα είναι «κρυμμένοι» μέσα στις αντιστοιχήσεις. Αυτό εξασφαλίζεται από τον τρόπο κατασκευής των MCDs.
Ο αλγόριθμος προσθέτει στο *Answer* το υποερώτημα που προκύπτει από κάθε υποσύνολο MCDs που ικανοποιεί την *Ιδιότητα 2*. Στο *Answer* παίρνουμε τελικά το μεταφρασμένο ερώτημα.	Το μεταφρασμένο ερώτημα λαμβάνεται σαν ένωση (union) των υποερωτημάτων που προκύπτουν από κάθε υποσύνολο MCDs που ικανοποιεί την *Ιδιότητα 2*.

Πίνακας 3.4 : Αντιστοιχίες των διαδικασιών combineMCDs για CNF και για SQL_CNF

Για να υλοποιήσουμε την παραπάνω διεργασία σχεδιάσαμε έναν αναδρομικό συνδυαστικό αλγόριθμο που βρίσκει τα υποσύνολα των MCDs που ικανοποιούν της ιδιότητες 2.1 και 2.2. Ο αλγόριθμος δεν εκτελεί εξαντλητική αναζήτηση σε όλα τα πιθανά υποσύνολα από MCDs ελέγχοντας για καθένα από αυτά εάν ικανοποιούνται οι ιδιότητες. Εκμεταλλεύεται εξ' αρχής το γεγονός ότι τα υποσύνολα MCDs που θα επιστρέψει πρέπει να καλύπτουν όλους της subgoals του Q και τον καθένα από μία μόνο φορά, και βρίσκει τα ζητούμενα υποσύνολα πολύ αποδοτικά.

Σημειώνουμε της ότι κατά τη «μετάφραση» του αλγορίθμου για να λειτουργεί με SQL_CNF, εμφανίζεται το εξής πρόβλημα: Κάθε εμφάνιση της της αντιστοίχησης στο τελικό ερώτημα, εισάγει λανθασμένα ένα επιπλέον καρτεσιανό γινόμενο στο ερώτημα.

Η πολλαπλή εμφάνιση της αντιστοίχησης V στο τελικό ερώτημα προκαλείται από το γεγονός ότι στο υποσύνολο από MCDs που χρησιμοποιήθηκε για τη μετάφραση υπάρχουν πολλά MCDs που ανήκουν στη V, τα οποία καλύπτουν διαφορετικούς subgoals του ερωτήματος. Δύο τέτοια MCDs είναι δυνατόν να εμφανίζονται σε ένα υποσύνολο για τον εξής λόγο. Στη διαδικασία formMCDs παρατηρούμε ότι αναζητούμε τον ελάχιστο ομομορφισμό ώστε να εξομοιώνεται της subgoal του ερωτήματος και της της αντιστοίχησης V. Εάν υπάρχει τέτοιος, δημιουργείται το αντίστοιχο MCD. Αυτό της δεν αποκλείει ότι το ίδιο μπορεί να συμβεί και για έναν άλλον subgoal του ερωτήματος, με αποτέλεσμα να δημιουργηθούν δύο (ή και περισσότερα) MCDs για τη V.

Το πρόβλημα δεν εμφανίζεται στον αρχικό αλγόριθμο διότι μέσα από τα Ψ_i 's εξισώνονται οι αντίστοιχες ιδιότητες των V, οπότε δεν προκύπτει το καρτεσιανό γινόμενο. Για να λύσουμε το πρόβλημα εισάγουμε μία συνάρτηση που συγχωνεύει με κατάλληλο τρόπο της πολλαπλές εμφανίσεις του V σε μία.

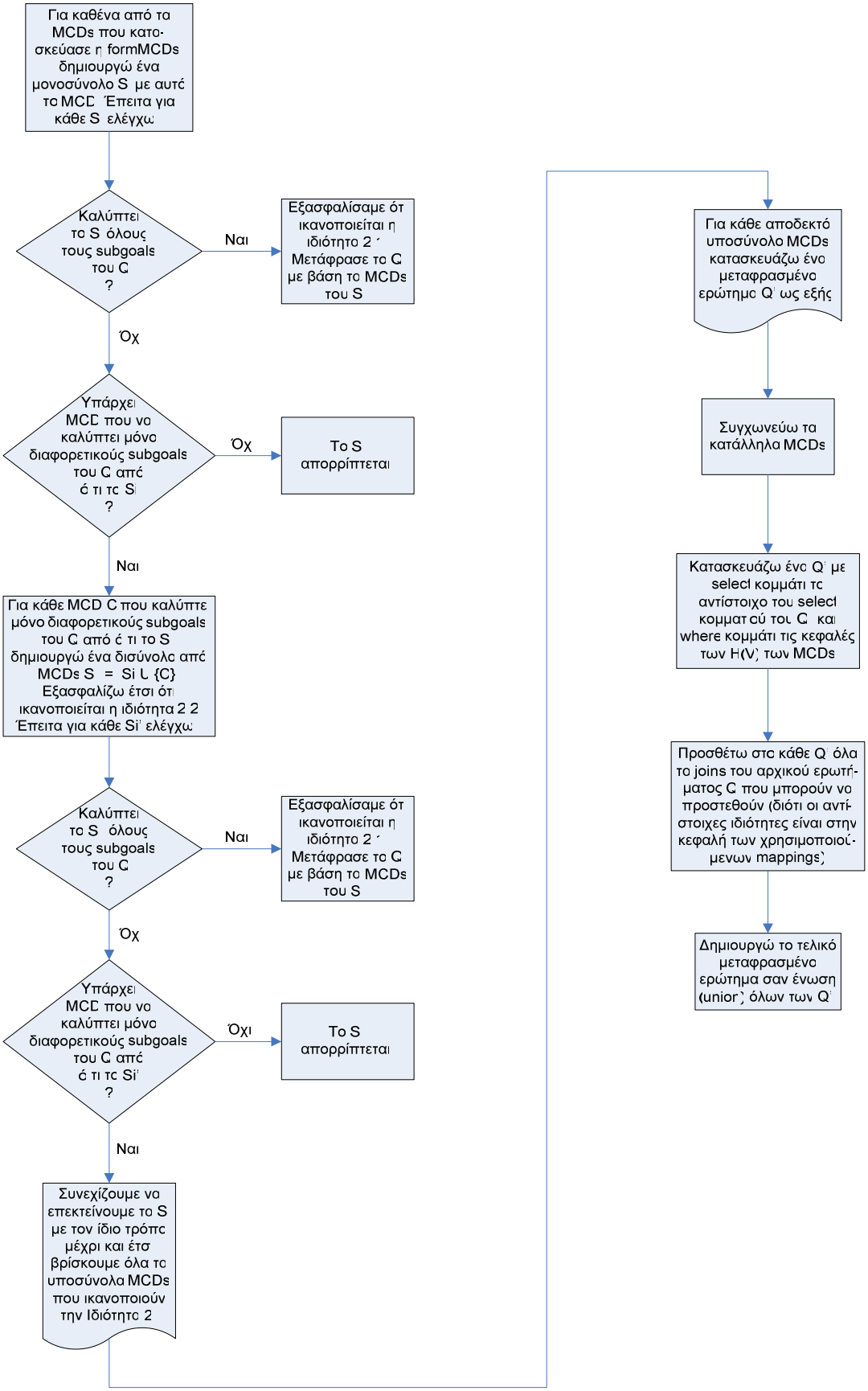
Στον Αλγόριθμο 3.3 (β) και στο Σχήμα 3.4 φαίνονται ο ψευδοκώδικας και το διάγραμμα ροής του αλγορίθμου combineMCDs για την SQL_CNF.

Οι διαδικασίες formMCDs και combineMCDs που αναλύσαμε έως τώρα μπορούν να μεταφράσουν μόνο ερωτήματα που δεν περιέχουν σταθερές και συγκρίσεις. Της επόμενες ενότητες θα περιγραφούν οι επεκτάσεις που έγιναν ώστε να μπορεί ο MiniCon να χειρίζεται και της περιπτώσεις τέτοιων ερωτημάτων. Της, θα εκτεθεί αναλυτικά ο τρόπος με τον οποίον αντιμετωπίσαμε τα ερωτήματα που περιέχουν self – joins. Τα ερωτήματα αυτά θεωρητικά δεν έχουν καμία διαφορά από εκείνα που δεν περιέχουν self – joins, στην υλοποίηση της προκαλούν πολύ σημαντικές δυσκολίες της οποίες θα μελετήσουμε και θα λύσουμε στη συνέχεια.

```
procedure combineMCDs(Q, C_set) /* SQL_CNF */
  /* C_set are MCDs formed by the formMCDs algorithm */
  /* Each MCD has the form [H_c, H_c(V), G_c] */
  Let Answer = "".
  For every subset C_1,… ,C_n of C_set s.t.
    Gc_1 U Gc_2 … U Gc_n = subgoals(Q) , and
    for every i≠j, Gc_i ∩ Gc_j = Ø
      If subset C_i = {MCD_1, MCD_2,…,MCD_n}
      Unify the appropriate MCDs
      Create a query Q'
        whose head is the correspondent of the head of Q and
        its body consists of the heads of H_1(V_1), H_2(V_2),…, H_n(V_n)
      For each Join Set js in Q create the greatest possible
        Join Set js' in Q', s.t. js' is a subset of js
    Add Q' to Answer.
  Return Answer.
```

Αλγόριθμος 3.3 (β) : Διαδικασία combineMCDs για SQL_CNF

Στο διάγραμμα ροής που ακολουθεί, περιγράφεται και ο τρόπος εύρεσης των υποσυνόλων των MCDs που ικανοποιούν την Ιδιότητα 2.



Σχήμα 3.4 : Διαδικασία combineMCDs για SQL_CNF

3.3.2 Ερωτήματα με σταθερές

Για να χειριστούμε και την περίπτωση ερωτημάτων με σταθερές, κάνουμε της παρακάτω αλλαγές της Ιδιότητες 1 και 2 και κατ' επέκταση της αλγορίθμους που της χρησιμοποιούν. Της αλλαγές αυτές της γράφουμε απευθείας της λειτουργούν για την SQL_CNF. Η ισοδυναμία της με εκείνες που αναγράφονται στο [PH01] συνάγεται εύκολα.

1. Τροποποιούμε την *Ιδιότητα 1.1* ως εξής: Αν μία ιδιότητα είναι διακεκριμένη στο ερώτημα Q, τότε στο Hc(V) πρέπει ή να είναι διακεκριμένη (της πριν), ή να είναι σταθερά. Το να είναι σταθερά σημαίνει ότι στην SQL_CNF μορφή του ερωτήματος υπάρχει της subgoal που εξισώνει την ιδιότητα αυτή με μία σταθερά του αντίστοιχου τύπου.
2. Επεκτείνουμε την *Ιδιότητα 1* προσθέτοντας την *Ιδιότητα 1.3* που είναι η εξής: Αν a είναι μία σταθερά στο ερώτημα Q, δηλαδή αν υπάρχει ιδιότητα x και subgoal "x = a" στο Q, τότε στο $H_c(V)$ πρέπει είτε η ιδιότητα x να είναι διακεκριμένη, είτε να ισούται με τη σταθερά a (δηλαδή να υπάρχει subgoal "x = a" και στο $H_c(V)$).
3. Επεκτείνουμε την *Ιδιότητα 2* θέτοντας έναν επιπλέον περιορισμό: Δύο MCDs C_1 και C_2 τ.π. στα Gc_1 και Gc_2 εμφανίζεται η ιδιότητα x ή κάποια άλλη ιδιότητα y από το join set της x, τότε αυτά μπορούν να συνδυαστούν της εξής περιπτώσεις: (1) Αν και τα δύο εξισώνουν τη x / y με την ίδια σταθερά, ή (2) αν το ένα εξισώνει τη x με κάποια σταθερά και στο άλλο η x / y είναι διακεκριμένη.
4. Σε καθεμία από της παραπάνω περιπτώσεις, όταν μία ιδιότητα x του ερωτήματος Q εμφανίζεται στο $H_c(V)$ που την καλύπτει εξισωμένη με κάποια σταθερά (δηλαδή υπάρχει στο $H_c(V)$ της subgoal "x = a", με a σταθερά), τότε στο μεταφρασμένο ερώτημα Q', εξισώνουμε τη x με την ίδια σταθερά (δηλαδή προσθέτουμε και στο Q' τον subgoal "x = a").

Της παραπάνω αλλαγές / ιδιότητες θα αναφερόμαστε με τα ονόματα Const1 – Const4.

3.3.3 Ερωτήματα με συγκρίσεις

Ο MiniCon της έχει παρουσιασθεί έως τώρα, μπορεί να χρησιμοποιηθεί για τη μετάφραση ερωτημάτων που δεν περιέχουν συγκρίσεις χρησιμοποιώντας αντιστοιχήσεις που μπορούν να περιέχουν συγκρίσεις. Με της επεκτάσεις που ακολουθούν, ο MiniCon θα μπορεί να δίνει maximally contained μεταφράσεις για της περισσότερες περιπτώσεις ερωτημάτων και αντιστοιχήσεων που έχουν συγκρίσεις. Υπάρχουν ωστόσο ορισμένα παραδείγματα ερωτημάτων για τα οποία ο αλγόριθμος δεν θα επιστρέψει αποτέλεσμα, παρότι υπάρχει δυνατή μετάφραση (βλ. [PH01]). Αποδεικνύεται της ότι για ερωτήματα που έχουν μόνο semi – interval περιορισμούς, δηλαδή της οι συγκρίσεις της είναι της μορφής $x < c$ ή $x \leq c$ (ή της

της μορφής x > c ή x ≥ c), ο της MiniCon δουλεύει και επιστρέφει ένα maximally contained ερώτημα σαν αποτέλεσμα.

Ορίζουμε ως I(Q) το σύνολο των comparison subgoals του Q. Της, δεδομένου της συνόλου μεταβλητών X στο Q, ορίζουμε ως $I_x(Q)$ το υποσύνολο των comparison subgoals του Q που περιέχουν: (1) Μόνο μεταβλητές από το X ή σταθερές, και (2) τουλάχιστον μία υπαρξιακή μεταβλητή του Q. Αν υποθέσουμε ότι οι subgoals της οποίους ανήκουν οι μεταβλητές του X καλύπτονται από ένα MCD C, τότε το $I_x(Q)$ δηλώνει το σύνολο των comparison subgoals του Q που πρέπει της να καλύπτονται από το C.

Κάνουμε τώρα της ακόλουθες αλλαγές στον MiniCon για να μπορεί να χειρίζεται ερωτήματα με συγκρίσεις :

1. Επεκτείνουμε την Ιδιότητα 1.2.1 ως εξής: Αν η μεταβλητή (ή η ιδιότητα) x ανήκει της Vars(Q) και x ανήκει στο F_c, τότε αν η $F_c(x)$ είναι υπαρξιακή μεταβλητή στο $H_c(V)$ και η y εμφανίζεται στον ίδιο comparison subgoal με τη x στο Q, πρέπει και η y να ανήκει στο F_c.
2. Επεκτείνουμε την Ιδιότητα 1.2.2 ως εξής: Αν το X είναι σύνολο μεταβλητών (ή ιδιοτήτων) που ανήκουν στο F_c, τότε πρέπει $I(H_c(V)) \models F_c(I_x(Q))$, δηλαδή πρέπει τα comparison subgoals του $H_c(V)$ να συνεπάγονται λογικά τα comparison subgoals του Q που ανήκουν στο $I_x(Q)$. Παρατηρούμε εδώ ότι επειδή το σύνολο X πρέπει από τον ορισμό του να περιέχει τουλάχιστον μία υπαρξιακή ιδιότητα του Q, οι μόνοι comparison subgoals του Q που δεν απαιτούμε να συμπεραίνονται λογικά από της comparison subgoals του $H_c(V)$, είναι εκείνοι που οι ιδιότητές της είναι διακεκριμένες στο $H_c(V)$. Ο λόγος είναι ότι οι subgoals αυτοί μπορούν να προστεθούν στο τελικό ερώτημα μετά τη σύνθεση των MCDs, της ακριβώς προστίθενται στο τελικό ερώτημα οι σύνδεσμοι του αρχικού ερωτήματος που συνέδεαν διακεκριμένες ιδιότητες των $H_c(V)$. Αυτό είναι εφικτό γιατί οι ιδιότητες της οποίες αναφέρονται θα βρίσκονται στην κεφαλή των αντιστοιχήσεων που χρησιμοποιήθηκαν στη μετάφραση.
3. Η επόμενη αλλαγή που κάνουμε είναι ότι απορρίπτουμε τα MCDs που θέτουν αριθμητικούς περιορισμούς σε ιδιότητες, ασύμβατους με εκείνους που έχουν οι ιδιότητες αυτές στο αρχικό ερώτημα. Για παράδειγμα, αν στο αρχικό ερώτημα Q η ιδιότητα X.x του πίνακα X εμφανίζεται σε έναν comparison subgoal “X.x > 10” ενώ στο MCD που καλύπτει τον πίνακα X υπάρχει ο comparison subgoal “X.x < 10”, τότε το MCD αυτό πρέπει να απορριφθεί. Εάν η ιδιότητα X.x είναι υπαρξιακή, η απόρριψη του MCD εξασφαλίζεται από την προηγούμενη αλλαγή (2), αφού η ο περιορισμός “X.x < 10” δεν συνεπάγεται ότι “X.x > 10”. Αυτό που προσθέτουμε εδώ είναι ότι για της ιδιότητες του Q που είναι διακεκριμένες στο $H_c(V)$, και

εμφανίζονται σε comparison subgoals τόσο στο Q όσο και στο $H_c(V)$, πρέπει να ελέγξουμε ότι οι δύο αυτοί comparison subgoals δεν είναι ασύμβατοι. Ωστόσο, δεν είναι απαραίτητο ο comparison subgoal του $H_c(V)$ να συνεπάγεται εκείνον του Q της πριν.

4. Η τελευταία αλλαγή που κάνουμε είναι αυτή που περιγράψαμε παραπάνω, ότι δηλαδή προσθέτουμε στο τελικό ερώτημα όσους comparison subgoals δεν έχουν μεταφραστεί μέσω των αντιστοιχήσεων διότι οι ιδιότητες που περιείχαν ήταν διακεκριμένες της αντιστοιχήσεις που χρησιμοποιήθηκαν για τη μετάφραση.

Της παραπάνω αλλαγές / ιδιότητες θα αναφερόμαστε με τα ονόματα Comp1 – Comp4.

3.3.4 Ερωτήματα με self – joins

Υπενθυμίζουμε ότι ένα self – join είναι της σύνδεσμος που εμφανίζεται ανάμεσα σε δύο «αντίγραφα» του ίδιου πίνακα μέσα σε ένα ερώτημα. Στην CNF, τα δύο αντίγραφα έχουν το ίδιο όνομα, ενώ αντίθετα στην SQL και στην SQL_CNF μορφή, πρέπει τα ονόματά της να διαφέρουν. Γι αυτό και ο πίνακας της περιπτώσεις αυτές μετονομάζεται με δύο διαφορετικά ονόματα.

Η αντιμετώπιση των ερωτημάτων που περιέχουν self – joins ήταν ένα πρόβλημα που αύξησε σημαντικά την πολυπλοκότητα της σχεδίασης και της υλοποίησης του αλγορίθμου MiniCon. Παρά το γεγονός ότι τα ερωτήματα αυτά θεωρητικά δεν διαφέρουν από εκείνα που δεν περιέχουν self – joins, στην πράξη χρήζουν διαφορετικής αντιμετώπισης. Θα δείξουμε γιατί αυτό συμβαίνει μέσα από τα παραδείγματα που ακολουθούν.

Χρησιμοποιούμε το παράδειγμα του P2P δικτύου με δύο κόμβους που είχαμε εισαγάγει στην παράγραφο 2.2.2, και το επεκτείνουμε για να μπορούμε να θέσουμε ερωτήματα με self – joins που να έχουν πραγματικό νόημα.

Προσθέτουμε στον Κόμβο 1 τον πίνακα:

Κ1_Αντικαθιστά (κωδ_Ιατρού, κωδ _Αντικαταστάτη)

που χρησιμοποιείται στην περίπτωση που της ιατρός δεν μπορεί για κάποιο λόγο να έρθει στο νοσοκομείο. Τότε, με βάση τον κωδικό του (κωδ_Ιατρού), βρίσκεται από τον πίνακα της ιατρός (με κωδικό κωδ_Αντικαταστάτη) για να τον αντικαταστήσει.

Έναν αντίστοιχο πίνακα προσθέτουμε και στον Κόμβο 2 με τη μορφή:

Κ2_Αμ.Αντικαθιστά (κωδικός_Ιατρού, όνομα_Ιατρού, επώνυμο_Ιατρού,
όνομα_Αντικαταστάτη, επώνυμο_Αντικαταστάτη)

όπου το "Αμ.Αντικαθιστά" σημαίνει Αμοιβαία Αντικαθιστά.

Προσθέτουμε επίσης τους πίνακες:

Κ1_Νοσοκόμος_Αντικαθιστά, **Κ2_Εργαζόμενος_Αντικαθιστά** και **Κ2_Επείγοντα**

που θα χρησιμεύσουν στο κεφάλαιο 5.

Τα σχήματα των δύο κόμβων έχουν τώρα ως εξής:

Κόμβος 1

Κ1_Ιατρός (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ1_Νοσοκόμος (κωδ_Νοσοκόμου, κωδ_Κλινικής, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ1_Υπάλληλος (κωδ_Υπαλλήλου, κωδ_Κλινικής, ειδικότητα, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ1_Κλινική (κωδ_Κλινικής, όνομα, κτίριο, όροφος, αριθμός_κλινών)

Κ1_Υπάγεται (κωδ_Ιατρού, κωδ_Κλινικής)

Κ1_Διευθύνει (κωδ_Ιατρού, κωδ_Κλινικής)

Κ1_Νοσηλευόμενος (κωδ_Νοσηλευομένου, κωδ_Ιατρού, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ1_Αντικαθιστά (κωδ_Ιατρού, κωδ_Αντικαταστάτη)

Κ1_Νοσοκόμος_Αντικαθιστά (κωδ_Νοσοκόμου, κωδ_Αντικαταστάτη)

Κόμβος 2

Κ2_Ιατρός (κωδικός_Ιατρού, κωδικός_Κλινικής, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ2_Εργαζόμενος (κωδικός_Εργαζομένου, θέση, ειδικότητα, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ2_Κλινική (κωδικός_Κλινικής, κωδικός_Διευθυντή, όνομα, κτίριο, όροφος, αριθμός κλινών)

Κ2_Εργαζόμενος_Υπάγεται (κωδικός_Εργαζομένου, κωδικός_Κλινικής)

Κ2_Ασθενής (κωδικός_Ασθενούς, όνομα, επώνυμο, διεύθυνση, τηλέφωνο)

Κ2_Επιβλέπει (κωδικός_Ιατρού, κωδικός_Ασθενούς)

Κ2_Αμ.Αντικαθιστά (κωδικός_Ιατρού, όνομα_Ιατρού, επώνυμο_Ιατρού, όνομα_Αντικαταστάτη, επώνυμο_Αντικαταστάτη)

Κ2_Εργαζόμενος_Αντικαθιστά (κωδικός_Εργαζομένου, κωδικός_Αντικαταστάτη)

Κ2_Επείγοντα (κωδικός_Ιατρείου, όνομα, κτίριο, όροφος)

Προσθέτουμε ακόμη στον Κόμβο 1 την εξής LAV αντιστοίχηση, γραμμένη σε SQL_CNF (Η κεφαλή της αντιστοίχησης περιέχει και τα ονόματα των ιδιοτήτων του πίνακα K2_Αμ.Αντικαθιστά της οποίες αντιστοιχούν οι ιδιότητες του σώματός της) :

LAV_K2_Αμ.Αντικαθιστά (K1_Ιατρός(1).κωδ_Ιατρού → κωδ_Ιατρού,
K1_Ιατρός(1).όνομα_Ιατρού → όνομα_Ιατρού,
K1_Ιατρός(1).επώνυμο_Ιατρού → επώνυμο_Ιατρού,
K1_Ιατρός(2).όνομα_Ιατρού → όνομα_Αντικαταστάτη,
K1_Ιατρός(2).επώνυμο_Ιατρού → επώνυμο_Αντικαταστάτη) :-

K1_Αντικαθιστά(1) (κωδ_Ιατρού, κωδ _Αντικαταστάτη),

K1_Αντικαθιστά(2) (κωδ_Ιατρού, κωδ _Αντικαταστάτη)

K1_Ιατρός(1) (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο),

K1_Ιατρός(2) (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο),

- Join Sets:

{ K1_Αντικαθιστά(1).κωδ_Ιατρού,
K1_Αντικαθιστά(2).κωδ_Αντικαταστάτη,
K1_Ιατρός(1).κωδ_Ιατρού },

{ K1_Αντικαθιστά(1).κωδ_Αντικαταστάτη,
K1_Αντικαθιστά(2).κωδ_Ιατρού,
K1_Ιατρός(2).κωδ_Ιατρού }

Comparison Predicates: -

Constant Predicates: -

Η παραπάνω LAV αντιστοίχηση συσχετίζει τον πίνακα "**K1_Αντικαθιστά**" του Κόμβου 1 με τον πίνακα "**K2_Αμ.Αντικαθιστά**" του Κόμβου 2. Παρατηρούμε ότι περιέχει πολλαπλές εμφανίσεις των πινάκων " **K1_Ιατρός** " και " **K1_Αντικαθιστά** ". Για το λόγο αυτό οι πίνακες έχουν μετονομαστεί και έχουν λάβει την αρίθμηση (1) και (2). Της περιέχει δύο self – joins στον πίνακα "**K1_Αντικαθιστά**".

Έστω τώρα ότι ο Κόμβος 1 θέτει της μετάφραση το εξής ερώτημα Q (σε SQL) :

select K1_Ιατρός(1).επώνυμο, K1_Ιατρός(2).επώνυμο
from K1_Αντικαθιστά(1), K1_Αντικαθιστά(2), K1_Ιατρός(1), K1_Ιατρός(2),
where K1_Αντικαθιστά(1).κωδ_Ιατρού = K1_Αντικαθιστά(2).κωδ_Αντικαταστάτη **and**
K1_Αντικαθιστά(2).κωδ_Ιατρού = K1_Αντικαθιστά(1).κωδ_Αντικαταστάτη **and**
K1_Ιατρός(1).κωδ_Ιατρού = K1_Αντικαθιστά(2).κωδ_Ιατρού **and**
K1_Ιατρός(2).κωδ_Ιατρού = K1_Αντικαθιστά(1).κωδ_Ιατρού **and**
K1_Ιατρός(1).επώνυμο > K1_Ιατρός(2).επώνυμο

θέλοντας να δει τα επώνυμα των ιατρών που ανά δύο αντικαθιστούν όταν είναι ανάγκη ο της τον άλλον. Η συνθήκη σύγκρισης στο τέλος του ερωτήματος θεωρούμε ότι γίνεται λεξικογραφικά και εξασφαλίζει ότι τα ζητούμενα ζεύγη δεν θα εμφανιστούν από δύο φορές το καθένα λόγω συμμετρίας του ερωτήματος, αλλά μόνο μία φορά.

Για να απαντηθεί το ερώτημα αυτό και από τον Κόμβο 2 πρέπει να μεταφραστεί με χρήση των διαθέσιμων αντιστοιχήσεων. Έστω ότι χρησιμοποιούμε LAV αντιστοιχήσεις. Τότε, θα έλεγε κανείς ότι η μετάφραση είναι προφανής διότι η LAV αντιστοίχηση που εισαγάγαμε παραπάνω μεταφράζει το ερώτημα ως εξής:

select LAV_K2_Αμ.Αντικαθιστά.επώνυμο_Ιατρού,
LAV_K2_ Αμ.Αντικαθιστά.επώνυμο_Αντικαταστάτη
from LAV_K2_ Αμ.Αντικαθιστά
where LAV_K2_Αμ.Αντικαθιστά.επώνυμο_Ιατρού >
LAV_K2_Αμ.Αντικαθιστά.επώνυμο_Αντικαταστάτη

Και όντως η παραπάνω μετάφραση είναι σωστή, της ο MiniCon της τον έχουμε περιγράψει έως τώρα δεν μπορεί να την ανακαλύψει για το λόγο που θα εξηγήσουμε αμέσως παρακάτω.

Γράφουμε το ερώτημα Q του Κόμβου 1 σε SQL_CNF και σημειώνουμε με πλάγια γραφή της διαφορές που έχει με τη LAV αντιστοίχηση " LAV_K2_Αμ.Αντικαθιστά ".

- Ερώτημα:

Q (*K1_Ιατρός(1).επώνυμο, K1_Ιατρός(2).επώνυμο*) :-

K1_Αντικαθιστά(1) (κωδ_Ιατρού, κωδ _Αντικαταστάτη),

K1_Αντικαθιστά(2) (κωδ_Ιατρού, κωδ _Αντικαταστάτη)

K1_Ιατρός(1) (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο),

K1_Ιατρός(2) (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο),

- Join Sets:

{ K1_Αντικαθιστά(1).κωδ_Ιατρού,
K1_Αντικαθιστά(2).κωδ_Αντικαταστάτη,
K1_Ιατρός(2).κωδ_Ιατρού },

{ K1_Αντικαθιστά(1).κωδ_Αντικαταστάτη,
K1_Αντικαθιστά(2).κωδ_Ιατρού,
K1_Ιατρός(1).κωδ_Ιατρού }

Comparison Predicates:

K1_Ιατρός(1).επώνυμο > K1_Ιατρός(2).επώνυμο

Constant Predicates: -

Οι διαφορές που παρατηρούμε στο κομμάτι της επιλογής και της σύγκρισης δεν αποτελούν πρόβλημα. Πρόβλημα της είναι οι διαφορές στα join sets του ερωτήματος. Διότι ο αλγόριθμος κατά τη δημιουργία των MCDs και συγκεκριμένα κατά τον έλεγχο της Ιδιότητας 1.2.2 θα αναζητήσει της συνδέσμους του ερωτήματος Q μέσα στην αντιστοίχηση. Και αν κάποιος σύνδεσμος λείπει από την αντιστοίχηση, θα απαιτήσει οι ιδιότητες της οποίες της συνδέει να είναι διακεκριμένες σε αυτή. Εάν αυτό δε συμβαίνει, τότε η αντιστοίχηση δεν μπορεί να χρησιμοποιηθεί. Στην περίπτωση αυτή εμπίπτει και το παράδειγμά της. Διότι το πρώτο join set του ερωτήματος δεν εμφανίζεται αυτούσιο στην αντιστοίχηση και οι ιδιότητα K1_Ιατρός(2).κωδ_Ιατρού που προκαλεί το πρόβλημα είναι υπαρξιακή. Ομοίως και στο

δεύτερο join set του ερωτήματος, οι ιδιότητες Κ1_Αντικαθιστά(1).κωδ_Αντικαταστάτη και Κ1_Αντικαθιστά(2).κωδ_Ιατρού είναι υπαρξιακές. Άρα τα join sets του Q δεν μπορούν να μεταφραστούν και ο αλγόριθμος δεν επιστρέφει αποτέλεσμα.

Το παραπάνω πρόβλημα προκύπτει διότι οι πίνακες Κ1_Αντικαθιστά(1) και Κ1_Αντικαθιστά(2) αντιμετωπίζονται από τον αλγόριθμο σαν δύο τελείως διαφορετικοί πίνακες. Αυτό της στην πραγματικότητα δεν ισχύει. Οι Κ1_Αντικαθιστά(1) και Κ1_Αντικαθιστά(2) είναι αντίγραφα του ίδιου πίνακα, γι αυτό και μπορούμε να τα εναλλάξουμε μέσα στο ερώτημα Q και να πάρουμε ένα ισοδύναμο ερώτημα Q' που θα είναι το εξής (σε SQL και SQL_CNF):

select Κ1_Ιατρός(1).επώνυμο, Κ1_Ιατρός(2).επώνυμο

from Κ1_Αντικαθιστά(2), Κ1_Αντικαθιστά(1), Κ1_Ιατρός(1), Κ1_Ιατρός(2),

where Κ1_Αντικαθιστά(2).κωδ_Ιατρού = Κ1_Αντικαθιστά(1).κωδ_Αντικαταστάτη **and**
Κ1_Αντικαθιστά(1).κωδ_Ιατρού = Κ1_Αντικαθιστά(2).κωδ_Αντικαταστάτη **and**
Κ1_Ιατρός(1).κωδ_Ιατρού = Κ1_Αντικαθιστά(1).κωδ_Ιατρού **and**
Κ1_Ιατρός(2).κωδ_Ιατρού = Κ1_Αντικαθιστά(2).κωδ_Ιατρού **and**
Κ1_Ιατρός(1).επώνυμο > Κ1_Ιατρός(2).επώνυμο

- Ερώτημα:

Q' (Κ1_Ιατρός(1).επώνυμο, Κ1_Ιατρός(2).επώνυμο) :-

Κ1_Αντικαθιστά(2) (κωδ_Ιατρού, κωδ _Αντικαταστάτη),

Κ1_Αντικαθιστά(1) (κωδ_Ιατρού, κωδ _Αντικαταστάτη)

Κ1_Ιατρός(1) (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο),

Κ1_Ιατρός(2) (κωδ_Ιατρού, βαθμός, όνομα, επώνυμο, διεύθυνση, τηλέφωνο),

- Join Sets:

{ Κ1_Αντικαθιστά(2).κωδ_Ιατρού,
Κ1_Αντικαθιστά(1).κωδ_Αντικαταστάτη,
Κ1_Ιατρός(2).κωδ_Ιατρού },

{ Κ1_Αντικαθιστά(2).κωδ_Αντικαταστάτη,
Κ1_Αντικαθιστά(1).κωδ_Ιατρού,
Κ1_Ιατρός(1).κωδ_Ιατρού }

Comparison Predicates:

K1_Ιατρός(1).επώνυμο > K1_Ιατρός(2).επώνυμο

Constant Predicates: -

Παρατηρούμε τώρα ότι τα join sets του Q' είναι ακριβώς όμοια με εκείνα της αντιστοίχησης "LAV_K2_Αμ.Αντικαθιστά", άρα το Q' μπορεί να μεταφραστεί επιτυχώς από τον MiniCon και να δώσει τη μετάφραση που περιμέναμε ότι θα έδινε και το Q.

Αυτό που επιτύχαμε με την παραπάνω διαδικασία μπορούμε να το επιτύχουμε και χωρίς να αλλάξουμε το αρχικό της ερώτημα Q, αλλά αντιστοιχώντας τον πίνακα K1_Αντικαθιστά(1) του Q στον πίνακα K1_Αντικαθιστά(2) της αντιστοίχησης και τον πίνακα K1_Αντικαθιστά(2) του Q στον πίνακα K1_Αντικαθιστά(1) της αντιστοίχησης. Δηλαδή κάθε σύνδεσμο του πίνακα K1_Αντικαθιστά(1) στο Q να τον αναζητάμε στον πίνακα K1_Αντικαθιστά(2) στην αντιστοίχηση και αντίστοιχα για τον K1_Αντικαθιστά(2). Αυτό είναι επιτρεπτό αφού οι δύο πίνακες είναι αντίγραφα του ίδιου πίνακα και τα ονόματά της έχουν επιλεχθεί τυχαία μέσα στο ερώτημα. Θα μπορούσε εξ' αρχής να έχουν ο της το όνομα του άλλου οπότε να μην είχε εμφανιστεί καθόλου το πρόβλημα.

Προκύπτει τώρα το εξής ερώτημα: Πώς δεδομένου της ερωτήματος Q που περιέχει self – joins και της LAV αντιστοίχησης V, θα βρούμε αν υπάρχει αντιστοιχία πινάκων του Q και της V τέτοια ώστε η V να μπορεί να χρησιμοποιηθεί για τη μετάφραση του Q; Και αν υπάρχει τέτοια αντιστοιχία ποια είναι;

Κατασκευάσαμε και ενσωματώσαμε στον MiniCon έναν αλγόριθμο που λύνει το παραπάνω πρόβλημα στη γενική του περίπτωση.

3.3.5 Αλγόριθμος εύρεσης αντιστοιχιών πινάκων

Θεωρούμε ερώτημα Q περιέχει στο σώμα του της πίνακες Q_1, Q_2, … Q_n και LAV αντιστοίχηση V που έχει της πίνακες V_1, V_2, … V_k. Της ενδιαφέρει να ανακαλύψουμε, αν υπάρχει, μία αντιστοιχία πινάκων του Q με πίνακες της V, ώστε η V να μπορεί να συμμετάσχει σε ένα MCD για τη μετάφραση του Q. Για να το επιτύχουμε αυτό, βρίσκουμε της της δυνατές αντιστοιχίες πινάκων του Q και της V, και τρέχουμε την formMCDs του MiniCon για καθεμία από αυτές. Στην τετριμμένη περίπτωση που κανένα από τα Q, V δεν περιέχει πολλαπλές εμφανίσεις πινάκων (δηλαδή κανένας πίνακας δεν εμφανίζεται σε δύο ή περισσότερα αντίγραφα), η αντιστοιχία των πινάκων είναι προφανής γιατί κάθε πίνακας του

Q απλά αντιστοιχεί στον πίνακα της V που έχει το ίδιο όνομα. Συνεπώς, για να δούμε αν η V μπορεί να χρησιμοποιηθεί στη μετάφραση, αρκεί να ελέγξουμε μόνο μία αντιστοιχία. Εάν της ή το Q ή η V ή και τα δύο περιέχουν πολλαπλές εμφανίσεις της ή περισσότερων πινάκων, τότε βρίσκουμε της της δυνατές αντιστοιχίσεις πινάκων του Q με πίνακες του V με τον εξής τρόπο :

1. Χωρίζουμε της πίνακες του Q σε ομάδες (Groups). Κάθε ομάδα περιέχει μόνο αντίγραφα του ίδιου πίνακα. Άρα οι ομάδες που θα δημιουργηθούν είναι σε πλήθος όσες και οι διαφορετικοί πίνακες που συμμετέχουν στο Q. Το ίδιο κάνουμε και για τη V. Για παράδειγμα, αν το Q και η V είναι τα παρακάτω (οι δείκτες δηλώνουν αντίγραφα του ίδιου πίνακα) :

   $Q(\ldots)$:- $A_1(\ldots)$, $A_2(\ldots)$, $A_3(\ldots)$, $B_1(\ldots)$, $B_2(\ldots)$, $C_1(\ldots)$

   και

   $V(\ldots)$:- $A_1(\ldots)$, $A_2(\ldots)$, $B_1(\ldots)$, $B_2(\ldots)$, $C_1(\ldots)$, $C_2(\ldots)$

   , τότε οι ομάδες πινάκων που θα δημιουργηθούν θα είναι:

   Q: $Q\_group_A = \{ A_1, A_2, A_3 \}$, $Q\_group_B = \{ B_1, B_2 \}$, $Q\_group_C = \{ C_1 \}$

   και

   V: $V\_group_A = \{ A_1, A_2 \}$, $V\_group_B = \{ B_1, B_2 \}$, $V\_group_C = \{ C_1, C_2 \}$.

   Σημείωση

   Στο παράδειγμά της τα Q, V έχουν της της πίνακες (A, B, C), ενώ αυτό δεν ισχύει στη γενική περίπτωση. Παρατηρούμε της ότι αν το Q έχει έναν πίνακα που δεν εμφανίζεται στη V, τότε της απλά δεν αντιστοιχίζεται πουθενά, άρα δεν επηρεάζει τη διαδικασία αντιστοίχησης. Το ίδιο ισχύει και αν η V έχει έναν πίνακα που δεν εμφανίζεται στο Q.

2. Για κάθε ομάδα πινάκων του Q βρίσκουμε την αντίστοιχή της στη V, αν υπάρχει. Εάν η αντίστοιχη ομάδα της V περιέχει λιγότερους πίνακες από εκείνη του Q συμπληρώνουμε την ομάδα αυτή με αριθμημένους “dummy” πίνακες, έτσι ώστε οι δύο ομάδες να έχουν τελικά ίσο αριθμό πινάκων. Ακολουθώντας αυτή τη διαδικασία, οι ομάδες πινάκων της V στο παραπάνω παράδειγμα μεταβάλλονται ως εξής:

   V: $V\_group_A = \{ A_1, A_2, A_{dummy\_1} \}$, $V\_group_B = \{ B_1, B_2 \}$, $V\_group_C = \{ C_1, C_2 \}$.

   Όταν της πίνακας του Q αντιστοιχεί σε έναν dummy πίνακα της V, αυτό θα σημαίνει ότι μένει εκτός από την αντιστοιχία πινάκων.

3. Εργαζόμαστε τώρα με κάθε ομάδα πινάκων του Q χωριστά. Παίρνουμε την αντίστοιχή της στη V, η οποία πλέον έχει ίσο αριθμό στοιχείων, και βρίσκουμε της της δυνατές αντιστοιχίες μεταξύ των πινάκων της με τον εξής απλό τρόπο. Για τον πρώτο πίνακα της ομάδας του Q, λέμε ότι μπορεί να αντιστοιχεί σε καθέναν από της

πίνακες της ομάδας της V. Θεωρούμε λοιπόν της ισάριθμες περιπτώσεις και για καθεμία από αυτές επαναλαμβάνουμε τη διαδικασία για τον δεύτερο πίνακα της ομάδας του Q, κοκ μέχρι η ομάδα να μην έχει της πίνακες. Δημιουργούμε με τον τρόπο αυτό ένα σύνολο αντιστοιχήσεων των πινάκων των δύο ομάδων. Της, της από αυτές της αντιστοιχήσεις δεν είναι εφικτές διότι αντιστοιχούν δύο ή περισσότερους πίνακες του Q στον ίδιο πίνακα της V. Γι αυτό, το τελευταίο βήμα του αλγορίθμου είναι η διαγραφή όσων από της παραχθείσες αντιστοιχήσεις έχουν αυτήν την ιδιότητα. Ο τρόπος της εύρεσης των αποδεκτών αντιστοιχήσεων μπορεί να μην είναι αρκετά αποδοτικός, της απλουστεύει την υλοποίηση του αλγορίθμου. Για το παράδειγμά της, οι αποδεκτές αντιστοιχήσεις για της πίνακες της ομάδας A (Q_group_A), που είναι οι A_1, A_2 και A_3 ,είναι οι εξής :

$\{A_1, A_2, A_3\} \rightarrow \{A_1, A_2, A_{d\_1}\}$,

$\{A_1, A_2, A_3\} \rightarrow \{A_1, A_{d\_1}, A_2\}$,

$\{A_1, A_2, A_3\} \rightarrow \{A_2, A_1, A_{d\_1}\}$,

$\{A_1, A_2, A_3\} \rightarrow \{A_2, A_{d\_1}, A_1\}$,

$\{A_1, A_2, A_3\} \rightarrow \{A_{d\_1}, A_1, A_2\}$,

$\{A_1, A_2, A_3\} \rightarrow \{A_{d\_1}, A_2, A_1\}$.

Αντίστοιχα, για της πίνακες των ομάδων Q_group_B και Q_group_C, οι αποδεκτές αντιστοιχήσεις θα είναι :

$\{B_1, B_2\} \rightarrow \{B_1, B_2\}$,

$\{B_1, B_2\} \rightarrow \{B_2, B_1\}$

και

$\{C_1\} \rightarrow \{C_1\}$,

$\{C_1\} \rightarrow \{C_2\}$.

4. Το τελευταίο βήμα του αλγορίθμου είναι ο συνδυασμός των ανά ομάδα πινάκων αντιστοιχήσεων που βρέθηκαν στο προηγούμενο βήμα, προκειμένου να παραχθούν οι τελικές αντιστοιχήσεις καθεμία από της οποίες θα αναφέρεται συνολικά της πίνακες του ερωτήματος Q. Ο τρόπος που αυτές παράγονται είναι προφανής. Αρκεί να πάρουμε όλους της δυνατούς συνδυασμούς από αντιστοιχίες πινάκων που προκύπτουν εάν διαλέξουμε από μία παραχθείσα αντιστοίχηση για κάθε ομάδα πινάκων του Q. Έτσι, για το παράδειγμά της οι τελικές αντιστοιχήσεις που είναι και τα αποτελέσματα του αλγορίθμου είναι εκείνες που φαίνονται στον πίνακα 3.5. Στην πρώτη στήλη του πίνακα φαίνονται οι πίνακες του ερωτήματος Q και στη δεύτερη οι πίνακες της αντιστοίχησης V :

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, A_2, -, B_1, B_2, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, -, A_2, B_1, B_2, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, A_1, -, B_1, B_2, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, -, A_1, B_1, B_2, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_1, A_2, B_1, B_2, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_2, A_1, B_1, B_2, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, A_2, -, B_1, B_2, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, -, A_2, B_1, B_2, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, A_1, -, B_1, B_2, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, -, A_1, B_1, B_2, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_1, A_2, B_1, B_2, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_2, A_1, B_1, B_2, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, A_2, -, B_2, B_1, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, -, A_2, B_2, B_1, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, A_1, -, B_2, B_1, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, -, A_1, B_2, B_1, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_1, A_2, B_2, B_1, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_2, A_1, B_2, B_1, C_1\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, A_2, -, B_2, B_1, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_1, -, A_2, B_2, B_1, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, A_1, -, B_2, B_1, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{A_2, -, A_1, B_2, B_1, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_1, A_2, B_2, B_1, C_2\}$,

$\{A_1, A_2, A_3, B_1, B_2, C_1\} \rightarrow \{-, A_2, A_1, B_2, B_1, C_2\}$.

Πίνακας 3.5 : Αποτελέσματα αλγορίθμου εύρεσης αντιστοιχιών πινάκων

Για καθεμία από τις παραπάνω αντιστοιχίες πινάκων του ερωτήματος Q με τη LAV αντιστοίχηση, V τρέχουμε τον αλγόριθμο MiniCon έως ότου βρούμε μία από αυτές που οδηγεί στην παραγωγή ενός MCD για τη V. Τότε αποθηκεύουμε στο νέο MCD την αντιστοίχηση πινάκων που χρησιμοποιήσαμε και δεν ελέγχουμε για τις υπόλοιπες. Αν δεν βρεθεί καμία αντιστοίχηση που να δίνει αυτό το αποτέλεσμα, τότε η V δεν μπορεί να χρησιμεύσει στη μετάφραση.

3.3.6 Τελική σχεδίαση του αλγορίθμου MiniCon

Λαμβάνοντας υπόψη την αρχική, απλουστευμένη μορφή του MiniCon (παρ. 3.3.1) και όσα αναφέρθηκαν στις προηγούμενες παραγράφους όσον αφορά την αντιμετώπιση ερωτημάτων με σταθερές, συγκρίσεις και self – joins, παρουσιάζουμε την τελική μορφή του αλγορίθμου μέσα από τους ψευδοκώδικες και τα διαγράμματα ροής που ακολουθούν.

Σημειώνουμε εδώ ότι επειδή οι *Ιδιότητες 1* και *2* έχουν επεκταθεί λόγω των αλλαγών που έγιναν για τον χειρισμό ερωτημάτων με σταθερές και συγκρίσεις, θα αναφερόμαστε σε αυτές ως *Επ. Ιδιότητα 1* και *2* (*Ext. Property 1, 2)* αντίστοιχα.

Όσον αφορά τις Ιδιότητες με βάση τις οποίες λειτουργεί η formMCDs, υπενθυμίζουμε ότι η Ιδιότητα 1.1 τροποποιήθηκε από την Const1, η Ιδιότητα 1.2.1 από την Comp1 και η Ιδιότητα 1.2.2 από την Comp2 και την Comp3. Επίσης, λόγω της Const2 προστέθηκε στην *Ιδιότητα 1* η Ιδιότητα 1.3.

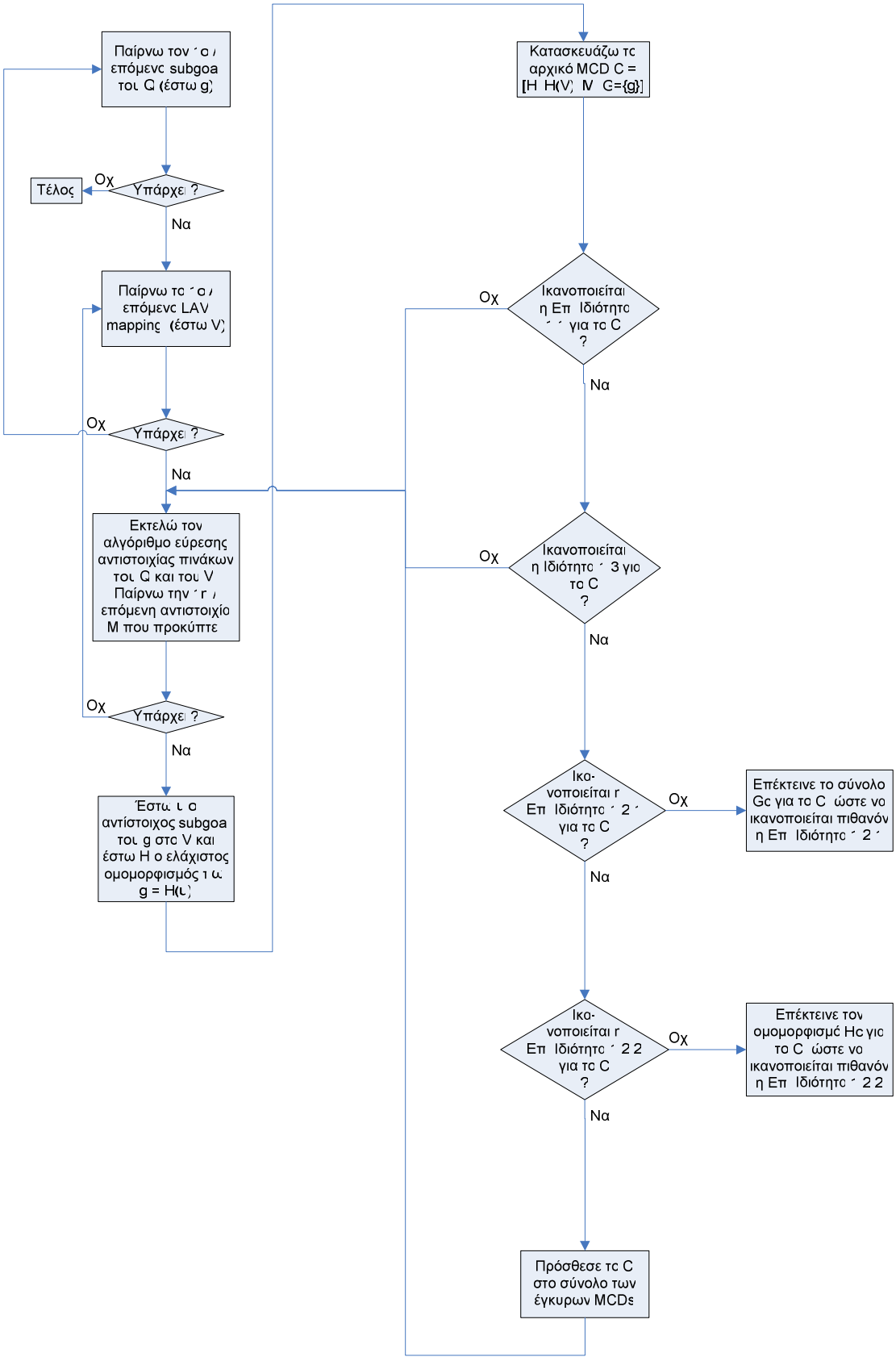
Αντίστοιχα, για τις Ιδιότητες που χρησιμοποιεί η combineMCDs, η Ιδιότητα 2 τροποποιήθηκε από την Const3. Επίσης, επισημαίνουμε ότι από τη διαδικασία συγχώνευσης των MCDs στην combineMCDs εξαιρείται η περίπτωση που οι subgoals που καλύπτονται από μία αντιστοίχηση V αντιστοιχούν σε αντίγραφα του ίδιου πραγματικού πίνακα. Τότε τα MCDs που τους καλύπτουν δεν είναι δυνατόν να συγχωνευθούν. Τέλος, η συγγραφή του τελικού ερωτήματος πρέπει τώρα να υπακούει και στις αλλαγές Const4 και Comp4.

Διαδικασία formMCDs – Ψευδοκώδικας

```
procedure formMCDs(Q,Vset) /* Final */
  Cset = Ø
  For each subgoal g in Q
    For each mapping V in Vset
      For each table mapping (correspondence) M between Q and V
        If u is the correspondent of g in V
          Let H be the least restrictive head
            Homomorphism on V s.t. g = H(u) (all joins that
            appear in g, must also appear in u)
          Let G = {g}
          Let (MCD) C = [H, H(V), M, G]
          Repeat
            If (C satisfies Ext.Property 1.1) then
              If (C satisfies Property 1.3) then
                If (C satisfies Ext.Property 1.2.1) then
                  If (C satisfies Ext.Property 1.2.2) then
                    add C to Cset, break the loop
                  else
                    extend G to Gc, let C = [Hc, Hc(V), Mc, Gc]
                else
                  extend H to Hc, let C = [Hc, Hc(V), Mc, Gc]
              else
                C is rejected
            else
              C is rejected
          Until (No extension could be made
                 in order to satisfy Ext.Property1)
  Return Cset.
```

Αλγόριθμος 3.4 : Διαδικασία formMCDs – Τελική σχεδίαση

Διαδικασία formMCDs - Διάγραμμα Ροής



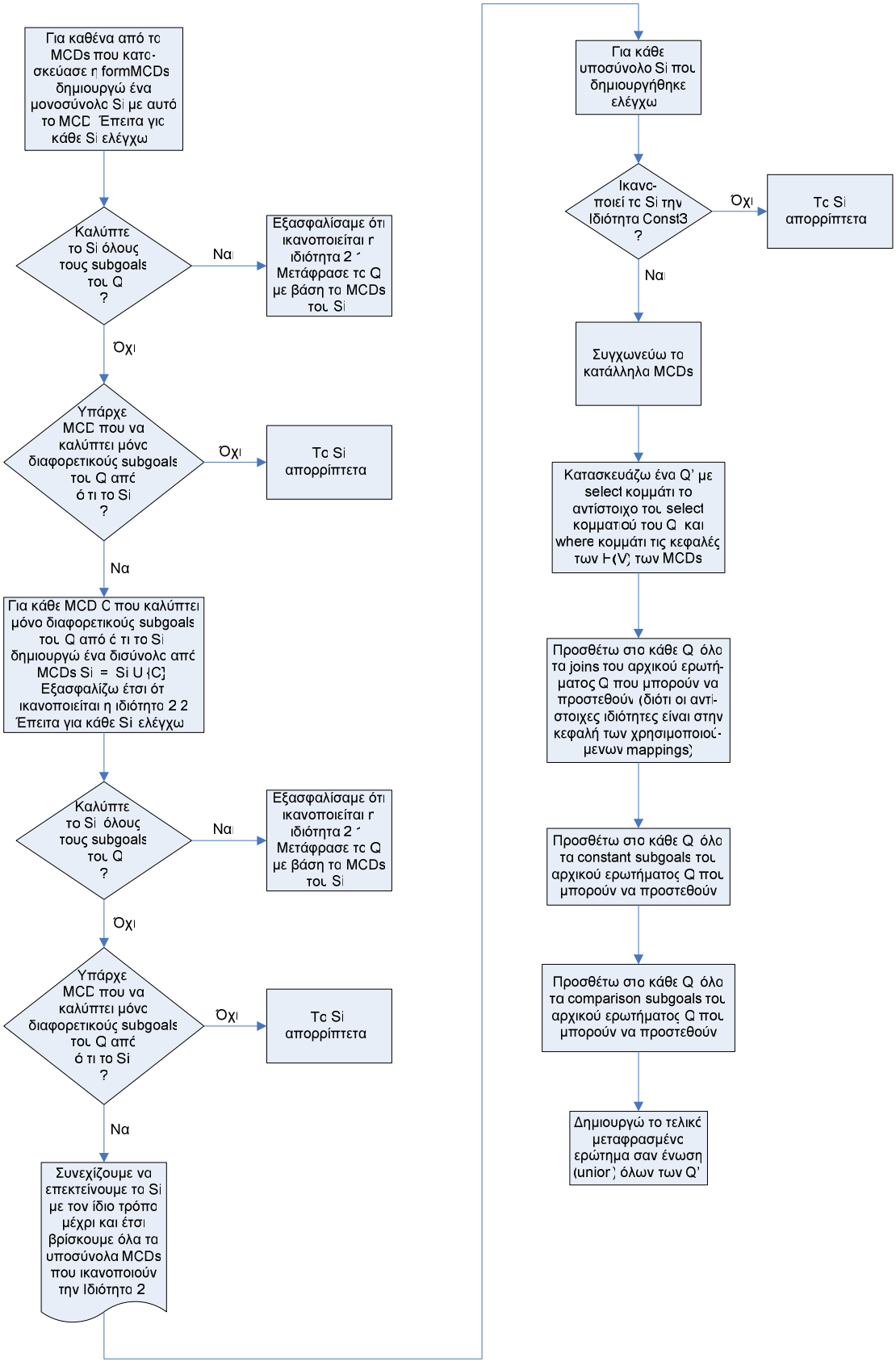
Σχήμα 3.5 : Διαδικασία formMCDs – Τελική σχεδίαση

Διαδικασία combineMCDs – Ψευδοκώδικας

```
procedure combineMCDs(Q, C_set) /* Final */
  /* C_set are MCDs formed by the formMCDs algorithm */
  /* Each MCD has the form [H_c, H_c(V), M_c, G_c] */
  Let Answer = "".
  For every subset C_1,… ,C_n of C_set s.t.
    Gc_1 U Gc_2 … U Gc_n = subgoals(Q), and
    for every i≠j, Gc_i ∩ Gc_j = Ø, and
    for every i≠j, Property Const3 is satisfied
      If subset C_i = {MCD_1, MCD_2,…,MCD_n}
      Unify the appropriate MCDs
      Create a query Q'
        whose head is the correspondent of the head of Q and
        its body consists of the heads of H_1(V_1), H_2(V_2),…, H_n(V_n)
      For each Join Set js in Q create the greatest possible
        Join Set js' in Q', s.t. js' is a subset of js
      For each constant subgoal in Q create it in
        Q' if possible
      For each comparison subgoal in Q create it in
        Q' if possible
    Add Q' to Answer.
  Return Answer.
```

Αλγόριθμος 3.5 : Διαδικασία combineMCDs – Τελική σχεδίαση

Διαδικασία combineMCDs – Διάγραμμα Ροής



Σχήμα 3.6 : Διαδικασία combineMCDs – Τελική σχεδίαση

3.4 Προεπεξεργασία ερωτημάτων

Το θέμα της προεπεξεργασίας των ερωτημάτων εξετάστηκε θεωρητικά σε προηγούμενο κεφάλαιο, κυρίως όσον αφορά τη μέτρηση ποιότητας των επεξεργασμένων (μειωμένων) ερωτημάτων. Στην παράγραφο αυτή θα αναλυθεί και η διαδικασία μείωσης των ερωτημάτων που δεν μπορούν να απαντηθούν με τις διαθέσιμες GAV ή LAV αντιστοιχήσεις. Εξετάζουμε ξεχωριστά τις περιπτώσεις χρήσης GAV και LAV αντιστοιχήσεις και όχι μαζί. Κι αυτό διότι οι αλγόριθμοι μετάφρασης που διαθέτουμε δεν εργάζονται με GAV και LAV αντιστοιχήσεις συγχρόνως, αλλά είναι διαφορετικοί για κάθε είδος. (Θα ήταν πολύ ενδιαφέρουσα και χρήσιμη η σχεδίαση ενός αλγορίθμου που να εκμεταλλεύεται συγχρόνως όλες τις διαθέσιμες αντιστοιχήσεις για τη μετάφραση ενός ερωτήματος). Επίσης, η διαδικασία μείωσης ενός ερωτήματος ώστε το μειωμένο ερώτημα να είναι το καλύτερο δυνατό που να μπορεί να απαντηθεί, είναι διαφορετική για χρήση GAV και για χρήση LAV αντιστοιχήσεων. Μάλιστα, για την περίπτωση των LAV αντιστοιχήσεων, η διαδικασία αυτή είναι ιδιαίτερα πολύπλοκη γιατί σχετίζεται με τον ίδιο τον αλγόριθμο μετάφρασης. Ωστόσο, είναι σημαντικό το γεγονός ότι παρά το ότι η διαδικασία μείωσης των ερωτημάτων διαφέρει για κάθε είδος αντιστοιχήσεων, τα μειωμένα ερωτήματα που προκύπτουν είναι συγκρίσιμα ως προς την ποιότητά τους. Έτσι, για κάθε είδος αντιστοιχήσεων, μειώνουμε το αρχικό ερώτημα και μετράμε την ομοιότητα του μειωμένου ερωτήματος με το αρχικό. Έπειτα το σύστημα συγκρίνει τις τιμές που προκύπτουν και απαντά το καλύτερο από τα δύο ερωτήματα.

Η διαδικασία μέτρησης ποιότητας των μειωμένων ερωτημάτων βασίστηκε στο [KS].

Στις παραγράφους που ακολουθούν θα αναφερόμαστε στο αρχικό μας ερώτημα ως Q, και στο μειωμένο ερώτημα που μπορεί να απαντηθεί με τις διαθέσιμες GAV ή LAV αντιστοιχήσεις ως Q^-.

3.4.1 GAV αντιστοιχήσεις

3.4.1.1 Εκτίμηση ποιότητας μειωμένου ερωτήματος

Όταν χρησιμοποιούμε GAV αντιστοιχήσεις είναι εύκολο να γνωρίζουμε για κάθε ιδιότητα που εμφανίζεται στο αρχικό ερώτημα εάν μπορεί να μεταφραστεί. Απλά ελέγχουμε εάν η ιδιότητα αυτή εμφανίζεται στην κεφαλή κάποιου από τις διαθέσιμες GAV αντιστοιχήσεις. Κάθε ιδιότητα που δεν μπορεί να μεταφραστεί μειώνει την ομοιότητα του Q με το Q^-. Το μέγεθος της μείωσης καθορίζεται από τον ρόλο της ιδιότητας μέσα στο αρχικό ερώτημα. Με βάση την ανάλυση που κάναμε σε προηγούμενο κεφάλαιο, αναθέτουμε σε κάθε ιδιότητα του

Q ένα βάρος w που εξαρτάται από τη θέση – ρόλο της μέσα στο ερώτημα. Το βάρος αυτό είναι κατά φθίνουσα σειρά το εξής :

1. w_k, αν η ιδιότητα είναι κλειδί (key) κάποιου πίνακα
2. w_s, αν είναι στο select κομμάτι του Q
3. w_j, αν συμμετέχει σε κάποιον σύνδεσμο στο Q
4. w_c, αν συμμετέχει σε κάποιο constant ή comparison subgoal του Q.

Αν μία ιδιότητα έχει περισσότερους από έναν από τους παραπάνω ρόλους μέσα στο Q, τότε το βάρος της θεωρούμε ότι είναι το μεγαλύτερο από τα αντίστοιχα βάρη.

Η διαδικασία μέτρησης της ομοιότητας του Q με το Q^- ακολουθεί τα εξής βήματα :

1. Για κάθε subgoal S του Q βρίσκουμε την ανάλογη GAV αντιστοίχηση, έστω M. Έπειτα για κάθε ιδιότητα του S που εμφανίζεται στο Q, βρίσκουμε το βάρος της μέσα στο ερώτημα και υπολογίζουμε τη συνάρτηση Mas(S,M) που ορίζεται ως το άθροισμα των βαρών όσων από τις ιδιότητες του S δεν μπορούν να μεταφραστούν από τη M. Για GAV αντιστοιχήσεις, αυτό σημαίνει ότι οι ιδιότητες αυτές δεν βρίσκονται στην κεφαλή της M.

   Δηλαδή :

   $\text{Mas(S,M)} = \sum_{\forall i \in S} w_i \cdot a_i$ όπου $a_i = 0$ αν η ιδιότητα *i* μεταφράζεται από τη M, αλλιώς 1.

   Επίσης υπολογίζουμε το άθροισμα των βαρών όλων των ιδιοτήτων του S που εμφανίζονται στο Q, ανεξάρτητα από το αν μπορούν να μεταφραστούν ή όχι.

2. Υπολογίζουμε το άθροισμα των βαρών w_{add} όλων των επιπλέον συνδέσμων και των αριθμητικών περιορισμών που εμφανίζονται στο where κομμάτι των χρησιμοποιούμενων αντιστοιχήσεων. Τα βάρη αυτά είναι διαφορετικά από τα w_k, w_s, w_j, w_c και μικρότερα σε τιμή.
3. Υπολογίζουμε τη συνάρτηση $\text{Sim}(Q,M_{set})$ που μετρά την ομοιότητα του ερωτήματος Q με το ερώτημα Q^- θα προκύψει αν αποκόψουμε από το Q όποια του τμήματα δεν μπορούν να μεταφραστούν χρησιμοποιώντας τις αντιστοιχήσεις του συνόλου M_{set}. Η $\text{Sim}(Q, M_{set})$ ορίζεται από την εξής σχέση :

$$Sim(Q, M_{set}) = 1 - \frac{\sum_{\forall S} Mas(S, M) + \sum_{\forall M} w_{add}}{\sum_{\forall S} w_i}$$

Δίνουμε εδώ τον ψευδοκώδικα για τη μέτρηση της ποιότητας του μειωμένου ερωτήματος.

```
procedure calculateSimForGAV(Q, GAV mappings)
  For each subgoal S of Q
    Find the corresponding GAV mapping M and add it to M_set
    Calculate Mas(S,M)
    Calculate the sum of the w_i's for all properties of S
  Calculate Sim(Q,M_set)
  Return Sim(Q,M_set)
```

Αλγόριθμος 3.6 : Υπολογισμός συνάρτησης Sim για τις GAV αντιστοιχήσεις

3.4.1.2 Εύρεση μειωμένου ερωτήματος Q^-

Η μείωση του ερωτήματος Q επιτυγχάνεται διατρέχοντας το ερώτημα και αποκόπτοντας από αυτό κάθε ιδιότητα που δεν εμφανίζεται στην κεφαλή της ανάλογης GAV αντιστοίχησης. Επίσης αν έχουμε έναν σύνδεσμο ή έναν αριθμητικό περιορισμό με δύο ιδιότητες εκ των οποίων έστω και η μία δεν μπορεί να μεταφραστεί, τότε αυτό αποκόπτεται.

Ο ψευδοκώδικας με βάση τον οποίο βρίσκουμε το Q^- είναι ο εξής :

```
procedure cutQuery(Q, M_set)
  For each select item si of Q
    If (si cannot be rewritten using M_set)
      Cut si from the select part
  For each item wi in the where part of Q
    If (wi cannot be rewritten)
      Cut the whole join/arithmetic constraint in which wi appears
  For each table T in the from part of Q
    If (T has no attributes in the cut version of Q)
      Cut T from the from part
  Return Q as Q-
```

Αλγόριθμος 3.7 : Εύρεση μειωμένου ερωτήματος

3.4.2 LAV αντιστοιχήσεις

Η προεπεξεργασία του ερωτήματος στην περίπτωση που χρησιμοποιούνται LAV αντιστοιχήσεις θα μπορούσε να ακολουθήσει τα ίδια βήματα με εκείνη για GAV αντιστοιχήσεις εάν υπήρχε τρόπος να γνωρίζουμε ποια κομμάτια του αρχικού ερωτήματος μπορούν να μεταφραστούν και ποια όχι. Στην περίπτωση των GAV αντιστοιχήσεων, όπως προαναφέραμε, απαντάμε το ερώτημα αυτό ελέγχοντας απλά ποιες ιδιότητες βρίσκονται στις κεφαλές τους. Όταν όμως χρησιμοποιούμε LAV αντιστοιχήσεις, το πρόβλημα είναι πολύ πιο σύνθετο διότι μία LAV αντιστοίχηση μπορεί να μεταφράζει και ιδιότητες που δεν βρίσκονται στην κεφαλή της. Στην πραγματικότητα, η διαδικασία της προεπεξεργασίας πρέπει αρχικά να εκτελεί τους ίδιους ελέγχους με τον αντίστοιχο αλγόριθμο μετάφρασης, τον MiniCon, για να αποφαίνεται αν το ερώτημα μπορεί να μεταφραστεί ως έχει, και επιπλέον, αν δεν μπορεί να μεταφραστεί, να εντοπίζει και να αποκόπτει από αυτό τα κομμάτια που προκαλούν το πρόβλημα. Παράλληλα, πρέπει να εξασφαλίζει ότι τα κομμάτια του ερωτήματος που κόβονται είναι τα ελάχιστα δυνατά, ή καλύτερα είναι εκείνα που θα αποδώσουν το καλύτερο ποιοτικά μειωμένο ερώτημα Q^-.

Στην παρούσα εργασία δεν υλοποιήθηκε η παραπάνω διαδικασία προεπεξεργασίας για LAV αντιστοιχήσεις αλλά μία απλούστερη η οποία όμως είναι πιθανόν να αποκόψει από αρχικό ερώτημα κομμάτια που στην πραγματικότητα μπορούν να μεταφραστούν. Η διαδικασία αυτή βασίζεται στην παρατήρηση ότι ο αλγόριθμος MiniCon μεταφράζει κάθε subgoal του ερωτήματος χρησιμοποιώντας ένα μόνο MCD, δηλαδή μία μόνο LAV αντιστοίχηση. Θεωρώντας τώρα απλουστευμένα, κατ' αντιστοιχία με την περίπτωση των GAV αντιστοιχήσεων, ότι κάθε LAV αντιστοίχηση μπορεί να μεταφράσει μόνο τις ιδιότητες που εμφανίζονται στο select κομμάτι της, υπολογίζει για κάθε subgoal του ερωτήματος την τιμή της συνάρτησης Mas όλων των LAV αντιστοιχήσεων. Έπειτα επιλέγει για κάθε subgoal εκείνη την αντιστοίχηση που αποδίδει τη μικρότερη τιμή της Mas και θεωρεί ότι αυτή μεταφράζει καλύτερα τον αντίστοιχο subgoal. Στη συνέχεια αποκόπτει από το ερώτημα τις ιδιότητες των subgoals που δεν μεταφράζονται από τις επιλεχθείσες αντιστοιχήσεις, δημιουργώντας έτσι το μειωμένο ερώτημα Q^-. Με βάση τα παραπάνω γίνεται και η εκτίμηση της ποιότητας του μειωμένου ερωτήματος, αφού βασίζεται στο ποια κομμάτια του αρχικού ερωτήματος αποκόπτονται.

3.4.2.1 Εκτίμηση ποιότητας μειωμένου ερωτήματος

Με βάση τα παραπάνω, η διαδικασία μέτρησης της ομοιότητας του Q με το Q^- για την περίπτωση των LAV αντιστοιχήσεων ακολουθεί αντίστοιχα βήματα με την περίπτωση των GAV αντιστοιχήσεων, με τις εξής διαφορές :

- Για κάθε subgoal S του ερωτήματος, η LAV αντιστοίχηση που θα το μεταφράσει δεν είναι γνωστή εξ' αρχής, αλλά επιλέγεται συγκρίνοντας τις τιμές Mas(S, M) για όλες τις διαθέσιμες LAV αντιστοιχήσεις, και επιλέγοντας εκείνη που δίνει τη μικρότερη τιμή.
- Ο υπολογισμός της συνάρτησης Sim δεν λαμβάνει υπόψη τους συνδέσμους και τα κατηγορήματα με συγκρίσεις ή σταθερές στο where κομμάτι των LAV όπως κάνει στην περίπτωση των GAV αντιστοιχήσεων. Διότι δεν μπορούμε να γνωρίζουμε αν αυτά είναι πρόσθετα σε σχέση με το ερώτημα αν δεν εκτελέσουμε ελέγχους του αλγορίθμου MiniCon. Συνεπώς, για τις LAV αντιστοιχήσεις, η Sim υπολογίζεται από τη σχέση:

$$Sim(Q, M_{set}) = 1 - \frac{\sum_{\forall S} Mas(S, M)}{\sum_{\forall S} w_i}$$

Δίνουμε εδώ τον ψευδοκώδικα για τη μέτρηση της ποιότητας του μειωμένου ερωτήματος για την περίπτωση των LAV αντιστοιχήσεων.

```
procedure calculateSimForLAV(Q, LAV mappings)
  For each subgoal S of Q
    Find the LAV mapping with the minimum Mas(S,M) and add it to M_set
    Calculate the sum of the w_i's for all properties of S
  Calculate Sim(Q,M_set)
  Return Sim(Q,M_set)
```

Αλγόριθμος 3.8 : Υπολογισμός συνάρτησης Sim για τις LAV αντιστοιχήσεις

3.4.2.2 Εύρεση μειωμένου ερωτήματος Q^-

Η μείωση του ερωτήματος Q σε Q^- γίνεται και πάλι αντίστοιχα με την περίπτωση των GAV αντιστοιχήσεων. Η συνάρτηση που εκτελεί τη μείωση είναι και πάλι η cutQuery της οποίας ο ψευδοκώδικας δόθηκε παραπάνω (Αλγόριθμος 3.7).

4

Υλοποίηση

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Η εργασία υλοποιήθηκε στην γλώσσα προγραμματισμού JAVA, χρησιμοποιώντας το προγραμματιστικό περιβάλλον Eclipse. Για την υλοποίηση χρησιμοποιήθηκε PC με CPU Intel Pentium M στα 1.86 GHz και μνήμη 1Gb, με λειτουργικό σύστημα Microsoft Windows XP Professional.

4.2 Λεπτομέρειες υλοποίησης

Η υλοποίηση χωρίζεται σε τρία πακέτα (packages) κλάσεων, ένα για κάθε κύριο τμήμα της εργασίας. Τα πακέτα αυτά είναι τα : GAV, MiniCon και Preprocessing. Θα περιγράψουμε αναλυτικά τις κλάσεις των πακέτων αυτών. Η περιγραφή των μεθόδων κάθε κλάσης θα γίνεται έτσι ώστε να μην υπάρχει επικάλυψη με όσα αναφέρθηκαν στο κεφάλαιο της Ανάλυσης και Σχεδίασης. Επίσης, για περισσότερες λεπτομέρειες μπορεί κανείς να ανατρέξει στα σχόλια του κώδικα.

Όπως προείπαμε, οι αλγόριθμοι μετάφρασης και προεπεξεργασίας υλοποιήθηκαν για να τρέχουν με ερωτήματα και αντιστοιχήσεις σε SQL. Για το parsing της SQL χρησιμοποιήσαμε έναν SQL parser για JAVA που λέγεται ZQL και διατίθεται δωρεάν στη διεύθυνση www.experlog.com/gibello/zql/ . Εδώ θα περιγράψουμε συνοπτικά και όσες από τις κλάσεις

της ZQL χρησιμοποιήσαμε για την υλοποίηση. Για περισσότερες λεπτομέρειες μπορεί κανείς να ανατρέξει στην παραπάνω διεύθυνση.

4.2.1 Package ZQL

4.2.1.1 class ZAliasedName

Σε αντικείμενα της κλάσης αυτής αποθηκεύεται κάθε τμήμα ενός ερωτήματος που έχει τη γενική μορφή Table.Column (as) Alias. Alias είναι το ψευδώνυμο (νέο όνομα) που λαμβάνει μία μετονομασμένη ιδιότητα ή πίνακας.

4.2.1.2 class ZSelectItem

Η κλάση αυτή κληρονομεί από την ZAliasedName. Στα αντικείμενά της αποθηκεύονται οι ιδιότητες του select κομματιού ενός ερωτήματος.

4.2.1.3 class ZFromItem

Η κλάση αυτή επίσης κληρονομεί από την ZAliasedName. Στα αντικείμενά της αποθηκεύονται οι πίνακες του from κομματιού ενός ερωτήματος.

4.2.1.4 interface ZExp

Κοινό interface για τις κλάσεις ZExpression, ZQuery και ZConstant.

4.2.1.5 class ZExpression

Στην κλάση αυτή αποθηκεύεται το where κομμάτι του ερωτήματος. Τα αντικείμενα της κλάσης δημιουργούνται χρησιμοποιώντας τον αναδρομικό της ορισμό:
ZExpression = ZExp op ZExp, όπου op ο τελεστής της έκφρασης.

4.2.1.6 class ZQuery

Στην κλάση αυτή αποθηκεύεται ένα πλήρες ερώτημα. Το select κομμάτι του ερωτήματος αποθηκεύεται σαν ένα διάνυσμα (Vector) από ZSelectItems, το from κομμάτι σαν διάνυσμα από ZFromItems, ενώ το where κομμάτι σαν μία ZExp.

4.2.1.7 class ZConstant

Στην κλάση αυτή αποθηκεύονται οι αριθμητικές και αλφαριθμητικές σταθερές που μπορούν να εμφανιστούν σε ένα ερώτημα. Επίσης σαν ZConstants αποθηκεύονται τα ονόματα των ιδιοτήτων όταν τα χειριζόμαστε σαν ZExps, δηλ. στο where κομμάτι του ερωτήματος.

4.2.1.8 interface ZStatement

Η κλάση ZQuery υλοποιεί αυτό το interface.

4.2.1.9 class ZqlParser

Ένα αντικείμενο της κλάσης αυτής χρησιμοποιείται κάθε φορά που θέλουμε να κάνουμε το parsing ενός SQL ερωτήματος. Το ερώτημα δίνεται στον ZqlParser σαν αντικείμενο της κλάσης java.io.InputStream. Η έξοδος λαμβάνεται σαν ένα ZStatement το οποίο γίνεται cast σε ZQuery. Το ZQuery αυτό περιέχει τα στοιχεία του ερωτήματος οργανωμένα στις κλάσεις που περιγράψαμε παραπάνω.

4.2.2 Package GAV

4.2.2.1 public class GAV_Translation

Η κλάση αυτή είναι εκείνη που εκτελεί όλη τη διαδικασία της μετάφρασης με χρήση GAV αντιστοιχήσεων.

Πεδία :

- `private static String Q_file;`
  Το όνομα του αρχείου που περιέχει το SQL ερώτημα Q που πρόκειται να μεταφραστεί.

- `private static String sender;`
  Το όνομα του κόμβου που αποστέλλει το ερώτημα Q. Το όνομα αυτό χρησιμοποιείται για να βρεθεί ο φάκελος με το σχήμα και τις αντιστοιχήσεις του κόμβου.

- `private static String receiver;`
  Το όνομα του κόμβου προς τον οποίο πρέπει να μεταφραστεί το ερώτημα Q. Το όνομα αυτό χρησιμοποιείται για να βρεθούν τις αντίστοιχες GAV αντιστοιχήσεις του κόμβου sender προς τον κόμβο αυτόν. Οι αντιστοιχήσεις περιέχονται σε .txt αρχεία και είναι γραμμένα σε SQL. Οι GAV αντιστοιχήσεις του sender προς τον receiver βρίσκονται στον φάκελο …/sender/To_receiver/GAV/ , όπου οι τελείες είναι το path όπου βρίσκεται η κλάση GAV_Translation και όπου sender και receiver τα ονόματα των αντίστοιχων κόμβων.

- `private static Vector allTableNames;`
  Διάνυσμα από Strings που περιέχει τα ονόματα των πινάκων που εμφανίζονται στο ερώτημα Q.

- `private static Vector renamedTableNames;`
  Διάνυσμα από Strings που περιέχει τα ονόματα των πινάκων που είναι μετονομασίες πραγματικών πινάκων του κόμβου sender.

- `private static Vector renamedTables;`
  Διάνυσμα από αντικείμενα της κλάσης TableAlias που περιέχει τους μετονομασμένους πίνακες και τα πραγματικά ονόματά τους.

- `private static Vector newFrom;`
  Διάνυσμα από αντικείμενα της κλάσης Zql.ZFromItem στο οποίο θα αποθηκευτεί το from κομμάτι του μεταφρασμένου ερωτήματος.

- `private static Vector mappingsUsed;`
  Διάνυσμα από Strings στο οποίο αποθηκεύονται τα ονόματα των αρχείων των αντιστοιχήσεων που έχουν χρησιμοποιηθεί μέχρι κάποιο στάδιο της μετάφρασης. Το κάθε όνομα αρχείου όμως, επεκτείνεται όπως και το αντίστοιχο mapping_Ren, όταν η αντιστοίχηση που περιέχει χρησιμοποιείται για τη μετάφραση ιδιοτήτων κάποιου μετονομασμένου πίνακα (βλ. παρ. 3.2).

Μέθοδοι :

- `public static void main(String[] args)`
  Η κύρια διαδικασία της μετάφρασης με GAV αντιστοιχήσεις. Στο σώμα της γίνεται η μετάφραση του select κομματιού του ερωτήματος Q. Επίσης, προστίθενται στο from κομμάτι του Q' (στο διάνυσμα newFrom) οι πίνακες που εμφανίζονται στο select κομμάτι του Q', ενώ στο where κομμάτι του προστίθενται τα where κομμάτια των αντιστοιχήσεων που χρησιμοποιήθηκαν. Εδώ γίνεται η χρήση του διανύσματος mappingsUsed, ώστε να εξασφαλίζεται ότι το from και το where κομμάτι μίας αντιστοίχησης δεν θα προστεθεί στο Q' παραπάνω από μία φορά.

- `private static ZExpression newCondition(ZExpression oldCond);`
  Η διαδικασία αυτή μεταφράζει το where κομμάτι του Q. Η μετάφραση γίνεται σε δύο στάδια. Αρχικά μεταφράζονται οι ιδιότητες που εμφανίζονται στις συνθήκες στο Q στις αντίστοιχές τους με βάση τις GAV αντιστοιχήσεις, οπότε παράγονται και προστίθενται αντίστοιχες συνθήκες στο Q'. Στη συνέχεια προστίθενται στο Q' και from και τα where κομμάτια των αντιστοιχήσεων που χρησιμοποιήθηκαν για τη μετάφραση των παραπάνω ιδιοτήτων. Και εδώ γίνεται χρήση του διανύσματος mappingsUsed.

- `private static ZQuery renameAllTablesInMapping(ZQuery mapping, String s);`
  Η διαδικασία αυτή χρησιμεύει στην περίπτωση που το Q περιέχει μετονομασμένους πίνακες. Τότε, πρέπει οι μεταφρασμένοι πίνακες να είναι επίσης μετονομασμένοι ώστε να μπορούν και στο Q' να διακρίνονται μεταξύ τους τα αντίγραφα του ίδιου πίνακα. Αν παρατηρήσουμε καλύτερα, θα δούμε ότι για να γίνει σωστά η μετάφραση μίας ιδιότητας που ανήκει σε ένα μετονομασμένο πίνακα του Q, πρέπει πρώτα να μετονομαστούν όλα τα στοιχεία της GAV αντιστοίχησης V που την μεταφράζει (βλ. παρ. 3.2). Αυτή την εργασία εκτελεί η *renameAllTablesInMapping* και επιστρέφει τη GAV αντιστοίχηση V' σαν ένα νέο ZQuery που έχει όλους τους πίνακες που

εμφανίζονται σε αυτό μετονομασμένους. Η μετονομασία γίνεται προσθέτοντας στο τέλος κάθε ονόματος πίνακα του V ένα underscore ('_') και το όνομα του μετονομασμένου πίνακα του Q. Αντίστοιχα μετονομάζεται και η GAV αντιστοίχηση σε mapping_Ren.

- `private static void updateNewFrom(Vector fv);`
  Η διαδικασία αυτή ενημερώνει το from κομμάτι του Q' (διάνυσμα newFrom) κάθε φορά που χρησιμοποιείται μία GAV αντιστοίχηση. Δέχεται σαν παράμετρο το from κομμάτι αυτής της αντιστοίχησης και προσθέτει στο from κομμάτι του Q' όσους από τους πίνακες του πρώτου δεν εμφανίζονται ήδη σε αυτό.

- `private static String originalNameOf(String tableName);`
  Παίρνει παράμετρο ένα όνομα πίνακα του Q και ελέγχει εάν είναι μετονομασία άλλου πίνακα. Εάν ναι, επιστρέφει το όνομα του πίνακα αυτού, αλλιώς επιστρέφει το όνομα που έλαβε σαν παράμετρο.

- `private static ZSelectItem noAlias(ZSelectItem s);`
  Παίρνει παράμετρο ένα ZSelectItem του Q και επιστρέφει το αντίστοιχο ZSelectItem που δεν έχει το ψευδώνυμο (alias) του αρχικού.

4.2.2.2 public class TableAlias

Η κλάση αυτή χρησιμεύει για την αποθήκευση των μετονομασμένων πινάκων ενός ερωτήματος.

Πεδία :

- `String originalName;`
  Το πραγματικό όνομα του πίνακα, δηλαδή εκείνο που υπάρχει στο σχήμα του αντίστοιχου κόμβου.

- `String aliasedName;`
  Το νέο όνομα του πίνακα.

Μέθοδοι :

- `public TableAlias(ZFromItem zfi)`
  Constructor της κλάσης.

- `public TableAlias(String on, String an)`
  Constructor της κλάσης.

- `public TableAlias copy()`
  Αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public String getOriginalName()`
  Επιστρέφει το originalName.

- `public String getAliasedName()`
  Επιστρέφει το aliasedName.

4.2.3 Package MiniCon

4.2.3.1 public class MiniMain

Η κλάση αυτή είναι η κύρια (εκτελέσιμη) κλάση του αλγορίθμου μετάφρασης με χρήση LAV αντιστοιχήσεις. Από την κλάση αυτή καλούνται οι κύριες συναρτήσεις άλλων κλάσεων που κατά σειρά εκτελούν: parsing του ερωτήματος και των διαθέσιμων LAV αντιστοιχήσεων (SQLToConj), δημιουργία των MCDs (FormMCDs), συνδυασμό των κατάλληλων MCDs (CombineMCDs) και κατασκευή του τελικού ερωτήματος (FinalSQL).

Πεδία :

- `private static String Q_file;`
  Το όνομα του αρχείου που περιέχει το SQL ερώτημα Q που πρόκειται να μεταφραστεί.

- `private static String sender;`
  Το όνομα του κόμβου που αποστέλλει το ερώτημα Q. Το όνομα αυτό χρησιμοποιείται για να βρεθεί ο φάκελος με το σχήμα και τις αντιστοιχήσεις του κόμβου.
- `private static String receiver;`
  Το όνομα του κόμβου προς τον οποίο πρέπει να μεταφραστεί το ερώτημα Q. Το όνομα αυτό χρησιμοποιείται για να βρεθούν οι αντίστοιχες LAV αντιστοιχήσεις του κόμβου sender προς τον κόμβο αυτόν. Οι αντιστοιχήσεις αυτές, όπως και οι GAV, είναι SQL ερωτήματα τα οποία περιέχονται σε .txt αρχεία που βρίσκονται στον φάκελο …/sender/To_receiver/LAV/ , όπου οι τελείες είναι το path του πακέτου MiniCon και όπου sender και receiver τα ονόματα των αντίστοιχων κόμβων.

Μέθοδοι :

- `public static void main(String[] args)`
  Καλεί τις διαδικασίες που αναφέραμε παραπάνω, καθώς και τις συναρτήσεις που ακολουθούν. Στη main ορίζεται και η μεταβλητή wholeSchema, στην οποία θα αποθηκευτεί το σχήμα της ΒΔ του κόμβου sender σαν αντικείμενο της κλάσης Query.

- `private static Query schemaOf(String sender)`
  Ανοίγει το αρχείο …/sender/sender_schema.txt και διαβάζει το σχήμα της ΒΔ του κόμβου sender. Το σχήμα αυτό είναι γραμμένο μέσα στο αρχείο σε μορφή SQL ερωτήματος. Στο select κομμάτι του ερωτήματος βρίσκονται όλες οι ιδιότητες των πινάκων του σχήματος του sender, ενώ στο from κομμάτι όλοι οι πίνακες του sender. Επειδή το σώμα των LAV αντιστοιχήσεων είναι γραμμένο στο σχήμα του sender, η γνώση του σχήματός του βοηθά στην αναγνώριση λαθών στο σώμα τους.

- `private static Vector allAvailableLAVMappings(String sender, String receiver, Query wholeSchema)`
  Η συνάρτηση αυτή επιστρέφει ένα διάνυσμα που περιέχει όλες τις διαθέσιμες LAV αντιστοιχήσεις του κόμβου sender σαν αντικείμενα της κλάσης Query, δηλαδή σε SQL_CNF μορφή.

- `public static Vector removeRedundantMCDs(Vector MCDs)`
  Η συνάρτηση αυτή παίρνει παράμετρο τα MCDs που έχουν δημιουργηθεί από την formMCDs. Ο αλγόριθμος της formMCDs με τον τρόπο που είναι σχεδιασμένος,

δημιουργεί κάθε MCD τόσες φορές όσοι είναι οι subgoals του Q που αυτό καλύπτει. Αυτό δεν είναι λάθος, αλλά αντίθετα αποτελεί επαλήθευση ότι ο αλγόριθμος λειτουργεί σωστά. Αυτός είναι και ο λόγος που δεν ενσωματώθηκε στην formMCDs έλεγχος που να αποτρέπει την δημιουργία πολλών ίδιων MCDs. Η συνάρτηση αυτή έχει το ρόλο να μειώνει το σύνολο των MCDs αφαιρώντας τις επαναλήψεις, αφότου όμως αυτά έχουν δημιουργηθεί και έχουμε ελέγξει ότι βρέθηκαν σωστά.

- `public static Vector setNames(Vector MCDs)`
  Η συνάρτηση αυτή αριθμεί τα MCDs που απέμειναν μετά την κλήση της removeRedundantMCDs.

- `public static void printMCDs(Vector MCDs)`
  Η συνάρτηση αυτή τυπώνει για κάθε MCD το όνομα (MCD#...), το όνομα της LAV αντιστοίχησης V, τους subgoals του Q που καλύπτει και την αντιστοιχία πινάκων που χρησιμοποιεί.

4.2.3.2 public class SQLToConj

Η κλάση αυτή αναλαμβάνει τη μετατροπή ενός SQL ερωτήματος σε ερώτημα SQL_CNF. Το νέο ερώτημα αποθηκεύεται σε ένα αντικείμενο της κλάσης Query.

Πεδία :

- `Vector selectPart;`
  Διάνυσμα στο οποίο αποθηκεύεται το select κομμάτι του αρχικού (SQL) ερωτήματος με τη μορφή ZSelectItems.

- `Vector fromPart;`
  Διάνυσμα στο οποίο αποθηκεύεται το from κομμάτι του αρχικού ερωτήματος με τη μορφή ZFromItems.

- `ZExpression wherePart;`
  ZExpression στην οποία αποθηκεύεται το where κομμάτι του αρχικού ερωτήματος.

Μέθοδοι :

- `public SQLToConj(FileInputStream fis)`
  Constructor της κλάσης SQLToConj. Παίρνει παράμετρο ένα File Input Stream που περιέχει ένα SQL ερώτημα, εκτελεί parsing χρησιμοποιώντας την Zql και αποθηκεύει τα κομμάτια του ερωτήματος στα αντίστοιχα πεδία της κλάσης SQLToConj.

- `public Query transform(String name, Query wholeSchema)`
  Η κύρια διαδικασία της κλάσης, που αναλύει το αρχικό ερώτημα στα στοιχεία του (select ιδιότητες, πίνακες, σύνδεσμοι, constant και comparison subgoals), τα αποθηκεύει στις αντίστοιχες κλάσεις και τελικά επιστρέφει το ερώτημα σαν αντικείμενο της κλάσης Query. Παράλληλα ελέγχει εάν στο ερώτημα εμφανίζονται πίνακες ή ιδιότητες που δεν υπάρχουν στο σχήμα του κόμβου στον οποίο αυτό αναφέρεται.

- `public String originalNameOf(String table)`
  Βρίσκει το πραγματικό όνομα ενός (πιθανώς μετονομασμένου) πίνακα που εμφανίζεται στο ερώτημα.

- `private boolean isRenamed(String tableName)`
  Παίρνει ένα όνομα πίνακα και ελέγχει εάν είναι μετονομασία άλλου πίνακα. Αν ναι επιστρέφει true, αλλιώς false.

- `public static ZSelectItem noAlias(ZSelectItem s)`
  Παίρνει ένα ZSelectItem του αρχικού ερωτήματος και επιστρέφει το αντίστοιχο ZSelectItem χωρίς το ψευδώνυμο (alias) του πρώτου.

4.2.3.3 public class FormMCDs

Η κλάση αυτή ευθύνεται για τη δημιουργία των MCDs. Ο αλγόριθμος της formMCDs υλοποιήθηκε με τον τρόπο που περιγράψαμε στο κεφάλαιο της Ανάλυσης και Σχεδίασης.

Πεδία :

- `Query main;`
  Το ερώτημα Q αφότου έχει μετατραπεί σε αντικείμενο της κλάσης Query από την SQLToConj.

- `Vector views;`
  Διάνυσμα από αντικείμενα της κλάσης Query στα οποία έχουν αποθηκευτεί οι διαθέσιμες LAV αντιστοιχήσεις.

Μέθοδοι :

- `public FormMCDs(Query main, Vector views)`
  Constructor της κλάσης FormMCDs.

- `public Vector execute()`
  Στην κλάση αυτή ενσωματώνονται τα εξωτερικά for loops της διαδικασίας formMCDs για την επιλογή των subgoals g του Q και u του V, καθώς και οι κλήσεις των συναρτήσεων που ελέγχουν αν ικανοποιούνται οι απαιτούμενες ιδιότητες από τα MCDs.

- `public static Query Hc_of_V(Query view, Homo h)`
  Η συνάρτηση αυτή εφαρμόζει έναν ομομορφισμό σε μία αντιστοίχηση V και επιστρέφει την $H_c(V)$.

- `private Homo checkHomoExistance (Query main, Query view, Subgoal viewSub, Subgoal mainSub, Mapping map)`
  Η συνάρτηση αυτή ελέγχει εάν για τους subgoals g (mainSub) και u (viewSub) υπάρχει ομομορφισμός H τ.ω. g = H(u). Εάν υπάρχει τέτοιος ομομορφισμός, η συνάρτηση τον βρίσκει και τον επιστρέφει σαν αντικείμενο της κλάσης Homo.

- `private MCD checkProperty1(Query main, Query view, Query newView, Vector mainSubsOfGc, Subgoal viewSub, Mapping map)`
  Από τη συνάρτηση αυτή ξεκινά ο έλεγχος των Ιδιοτήτων 1.1, 1.2 και 1.3, που πρέπει να ικανοποιεί ένα MCD. Αναφερόμαστε εδώ στις επεκτεταμένες Ιδιότητες, όπως περιγράφηκαν στο κεφάλαιο Ανάλυσης και Σχεδίασης. Η συνάρτηση δέχεται ως παραμέτρους την $H_c(V)$ (newView) και το G_c (mainSubsOfGc) ενός αρχικού MCD

και ελέγχει αν αυτό ικανοποιεί τις ζητούμενες ιδιότητες. Αν όχι, τότε εφαρμόζει τις κατάλληλες επεκτάσεις του $H_c(V)$ ή του G_c που απαιτούνται για την ικανοποίησή τους. Αν τελικά καταλήξει σε ένα αποδεκτό MCD, το επιστρέφει σαν αποτέλεσμα, αλλιώς επιστρέφει null.

- `private boolean checkC1(Query main, Query view, Vector mainSubsOfGc, Subgoal viewSub, Mapping map)`

  Η συνάρτηση αυτή ελέγχει αν ικανοποιείται η Επ. Ιδιότητα 1.1 από το σύνολο G_c (mainSubsOfGc) ενός MCD. Επιστρέφει true αν ναι, false αλλιώς.

- `private Extension checkC2(Query main, Query newView, Vector mainSubsOfGc, Subgoal viewSub, Mapping map)`

  Η συνάρτηση αυτή καλεί τις συναρτήσεις που ακολουθούν και ελέγχει αν ικανοποιείται η Επ. Ιδιότητα 1.2 από ένα MCD. Τα στοιχεία του MCD που χρειάζονται για τον έλεγχο είναι όπως και παραπάνω η $H_c(V)$ (newView) και το G_c (mainSubsOfGc) του MCD. Σε κάθε περίπτωση η συνάρτηση επιστρέφει ένα αντικείμενο της κλάσης Extension που περιγράφει αν χρειάζεται επέκταση του H_c ή του G_c, αν είναι δυνατή, και αν ναι ποια ακριβώς είναι. Την Extension αυτή την λαμβάνει σαν αποτέλεσμα η συνάρτηση checkProperty1 η οποία καλεί την checkC2 και την χειρίζεται ανάλογα με το είδος της.

- `private static Extension checkC2_1(Query main, Subgoal g, Vector Fc)`

  Η συνάρτηση ελέγχει αν ικανοποιείται η Ιδιότητα 1.2.1. Επιστρέφει ένα αντικείμενο της κλάσης Extension που μπορεί να δηλώνει ότι χρειάζεται επέκταση του G_c του MCD. Το MCD προσδιορίζεται από τις ιδιότητες του Q που περιέχονται στο διάνυσμα F_c .

- `private Extension checkC2_2(Query main, Query newView, Vector mainSubsOfGc, Subgoal g, Mapping map)`

  Η συνάρτηση ελέγχει αν ικανοποιείται η Ιδιότητα 1.2.2. Επιστρέφει ένα αντικείμενο της κλάσης Extension που μπορεί να δηλώνει ότι χρειάζεται επέκταση του H_c του MCD. Το MCD προσδιορίζεται όπως παραπάνω, από την $H_c(V)$ (newView) και το G_c (mainSubsOfGc).

- `private Extension checkComparisons1(Query main, Query newView, Vector Fc, Mapping map)`

  Η συνάρτηση αυτή ελέγχει αν ικανοποιείται η ιδιότητα Comp1 (βλ. κεφ. Ανάλυση και Σχεδίαση), η οποία είναι επέκταση της Ιδιότητας 1.2.1. Επιστρέφει ένα αντικείμενο Extension ανάλογο με εκείνο που επιστρέφει η checkC2_1.

- `private boolean checkComparisons2(Query main, Query newView, Vector Fc, Mapping map)`

  Η συνάρτηση αυτή ελέγχει αν ικανοποιούνται οι ιδιότητες Comp2 και Comp3, οι οποίες είναι επεκτάσεις της Ιδιότητας 1.2.2. Επιστρέφει true ή false ανάλογα.

- `private boolean checkC3(Query main, Query newView, Vector mainSubsOfGc, Mapping map)`

  Η συνάρτηση αυτή ελέγχει αν ικανοποιείται η Ιδιότητα 1.3 που ταυτίζεται με την Const2. Επιστρέφει true ή false ανάλογα.

- `private Vector allCoveredSubgoals(Query main, MCD m)`

  Η συνάρτηση αυτή διορθώνει ένα πρόβλημα που μπορεί να εμφανιστεί σε ορισμένες περιπτώσεις. Αυτό είναι ότι ένα MCD καλύπτει στην πραγματικότητα περισσότερους subgoals από εκείνους που ανακαλύπτει ο αλγόριθμος. Η συνάρτηση παίρνει παράμετρο το κάθε MCD που δημιουργείται και προσθέτει σε αυτό όλους τους subgoals του ερωτήματος που αυτό καλύπτει.

- `public static Vector Fc(Query main, Vector mainSubsOfGc, Mapping map)`

  Η συνάρτηση αυτή παίρνει σαν παράμετρο το σύνολο G_c ενός MCD και προσθέτει στο διάνυσμα που επιστρέφει, όλες τις ιδιότητες που εμφανίζονται στους subgoals του. Επίσης προσθέτει κάθε ιδιότητα του Q που ανήκει στο ίδιο join set με κάποια από τις προηγούμενες.

- `public static Vector Fc_of_x(Query main, ZSelectItem zsi, Query view, Mapping map)`

  Η συνάρτηση αυτή παίρνει παράμετρο ένα ZSelectItem (zsi) του Q και βρίσκει σε τι αντιστοιχεί στην αντιστοίχηση V (view). Επειδή δεν εργαζόμαστε με μεταβλητές αλλά με ιδιότητες, το zsi δεν αντιστοιχεί σε μία μόνο ιδιότητα της V. Το ζητούμενο σύνολο ιδιοτήτων βρίσκεται ακολουθώντας την ακόλουθη διαδικασία. Παίρνουμε το

join set του zsi στο Q. Για κάθε ιδιότητά του, βρίσκουμε την αντίστοιχή της στη V (view), καθώς και το join set της στη V. Τέλος, προσθέτουμε στο διάνυσμα που επιστρέφεται ($F_c(x)$) τις ιδιότητες από όλα αυτά τα join sets της V.

- `private static boolean sameSubAttrs(Subgoal Vsub, Subgoal Qsub)`
  Η συνάρτηση αυτή χρησιμοποιείται για να ελέγξει ότι δύο subgoals, ένας του Q και ένας της V που έχουν το ίδιο όνομα πίνακα, έχουν όντως και τις ίδιες ιδιότητες. Αν βρει ότι υπάρχει διαφοροποίηση, επιστρέφεται το κατάλληλο μήνυμα λάθους.

- `public static String simpleName(String tableName)`
  Στην υλοποίησή μας, όταν ένα ερώτημα ή αντιστοίχηση περιέχει μετονομασμένους πίνακες, τότε αυτοί, ανεξάρτητα με το πώς είναι αρχικά μετονομασμένοι, παίρνουν τα ονόματα X#1, X#2 … , όπου X το όνομα του πραγματικού πίνακα. Ανάλογα ονομάζονται και οι αντίστοιχοι σε αυτούς subgoals. Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα πίνακα (π.χ A#2) και επιστρέφει το όνομα του πραγματικού πίνακα (A).

- `public static boolean areSameTables(String t1, String t2)`
  Η συνάρτηση αυτή επιστρέφει true αν τα ονόματα των πινάκων στις παραμέτρους της ανήκουν στον ίδιο πραγματικό πίνακα. Π.χ. επιστρέφει true στις περιπτώσεις που οι παράμετροι είναι (A#1, A#3) ή (A, A#1), και false όταν είναι (B, A) ή (B, A#1).

- `public static String mappingOf (String table, Mapping map)`
  Η συνάρτηση επιστρέφει τον πίνακα της αντιστοίχησης V στον οποίον αντιστοιχεί ο πίνακας table του ερωτήματος Q με βάση την αντιστοιχία πινάκων map.

- `public static String mappingOfViewTable (String table, Mapping map)`
  Η συνάρτηση επιστρέφει τον πίνακα του Q στον οποίον αντιστοιχεί ο πίνακας table της αντιστοίχησης V με βάση την αντιστοιχία πινάκων map. Είναι η αντίστροφη της προηγούμενης.

- `public static ZSelectItem mappingOf(ZSelectItem zsi, Mapping map)`
  Η συνάρτηση επιστρέφει το ZSelectItem της V στο οποίο αντιστοιχεί το zsi του Q με βάση την αντιστοιχία πινάκων map.

- `public Vector mappingOfFc(Vector Fc, Mapping map)`
  Η συνάρτηση αυτή χρησιμοποιεί την παραπάνω και επιστρέφει ένα διάνυσμα από Strings που αποτελείται από τις αντίστοιχες (με βάση την αντιστοιχία πινάκων map) των ιδιοτήτων του Q που περιέχονται στο F_c.

- `public static Vector copyStringVector(Vector v)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα διάνυσμα από Strings, το αντιγράφει και επιστρέφει ένα όμοιό του που είναι πλέον ανεξάρτητο από το αρχικό.

4.2.3.4 public class Join

Στην κλάση αυτή αποθηκεύονται τα Join Sets.

Πεδία :

- `Vector JElements;`
  Διάνυσμα από Strings που περιέχει τις ιδιότητες που συμμετέχουν το join set.

Μέθοδοι :

- `public Join()`
  Constructor της κλάσης.

- `public Join (ZSelectItem left, ZSelectItem right)`
  Constructor της κλάσης που προσθέτει στο διάνυσμα JElements τις ιδιότητες left και right σε μορφή Strings.

- `public Join copy()`
  Συνάρτηση που αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

4.2.3.5 public class Homo

Στην κλάση αυτή αποθηκεύονται οι ομομορφισμοί (Homomorphisms).

Πεδία :

- `Vector newJoins;`
  Διάνυσμα από αντικείμενα της κλάσης Join που περιέχει τους συνδέσμους που εισάγει ο ομομορφισμός ανάμεσα σε διακεκριμένες ιδιότητες μίας αντιστοίχησης.

- `String type;`
  Το πεδίο αυτό χρησιμοποιείται για να μπορεί ένα αντικείμενο Homo να επιστραφεί σαν αποτέλεσμα από συναρτήσεις που ελέγχουν για την ικανοποίηση ιδιοτήτων που σχετίζονται με ομομορφισμούς. Ο τύπος ενός αντικειμένου Homo μπορεί να είναι ένας από τους παρακάτω :
  - "OK" : Δεν απαιτείται εφαρμογή επιπλέον ομομορφισμού για την ικανοποίηση της ιδιότητας. Η ιδιότητα ήδη ικανοποιείται.
  - "NoHomo" : Δεν υπάρχει ομομορφισμός που να οδηγεί στην ικανοποίηση της ιδιότητας. Η ιδιότητα δεν μπορεί να ικανοποιηθεί.
  - "Extension" : Βρέθηκε ομομορφισμός που οδηγεί στην ικανοποίηση της ιδιότητας.

Μέθοδοι :

- `public Homo()`
  Constructor της κλάσης που δημιουργεί ένα αντικείμενο τύπου OK.

- `public Homo(int i)`
  Constructor της κλάσης που δημιουργεί ένα αντικείμενο τύπου NoHomo.

- `public Homo(Join j)`
  Constructor της κλάσης που δημιουργεί ένα αντικείμενο τύπου Extension. Ο νέος σύνδεσμος που θα εισάγει ο ομομορφισμός δίνεται σαν παράμετρος.

- `public Homo copy()`
  Η συνάρτηση αυτή αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public Vector getNewJoins()`
  Επιστρέφει το πεδίο newJoins.

- `public String getType()`
  Επιστρέφει το πεδίο type.

- `public void setType(String type)`
  Θέτει στο πεδίο type την τιμή της παραμέτρου.

- `public void addJoin(Join j)`
  Προσθέτει στο πεδίο newJoins ένα επιπλέον αντικείμενο της κλάσης Join που πρέπει να συμπεριληφθεί στον ομομορφισμό.

- `public void printNewJoins()`
  Τυπώνει τους νέους συνδέσμους που εισάγει ο ομομορφισμός.

4.2.3.6 public class Constant

Στην κλάση αυτή αποθηκεύονται οι constant subgoals, δηλαδή εκείνοι που έχουν τη μορφή "ιδιότητα = σταθερά". Η σταθερά μπορεί να είναι αριθμητική ή αλφαριθμητική. Οι αλφαριθμητικές σταθερές για να αναγνωριστούν από την Zql πρέπει να μπουν σε μονά εισαγωγικά ('…').

Πεδία :

- `String attribute;`
  Η ιδιότητα του constant subgoal.

- `String constant;`
  Η σταθερά του constant subgoal.

Μέθοδοι :

- `public Constant(ZSelectItem zsi, String c)`
  Constructor της κλάσης. Η ιδιότητα δίνεται σαν ZSelectItem.

- `public Constant(String si, String c)`
  Constructor της κλάσης. Η ιδιότητα δίνεται σαν String.

- `public Constant copy()`
  Η συνάρτηση αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public String getAttr()`
  Επιστρέφει το πεδίο attribute.

- `public String getConstant()`
  Επιστρέφει το πεδίο constant.

- `public void print()`
  Τυπώνει τον constant subgoal.

4.2.3.7 public class Comparison

Στην κλάση αυτή αποθηκεύονται οι comparison subgoals, δηλαδή εκείνοι που έχουν τη μορφή "ιδιότητα op σταθερά" ή "ιδιότητα op ιδιότητα", όπου ο τελεστής op μπορεί να είναι ένας από τους: <, >, ≥, ≤. Η σταθερά στους comparison subgoals μπορεί να είναι μόνο αριθμητική. Επισημαίνουμε ότι για να απλοποιήσουμε τους ελέγχους για τις συγκρίσεις, κατά τη μετατροπή των ερωτημάτων σε SQL_CNF από την κλάση SQLToConj, όποτε βρίσκουμε έναν comparison subgoal της μορφής "σταθερά op ιδιότητα" τον μετατρέπουμε στον αντίστοιχο της μορφής "ιδιότητα op σταθερά" αντιστρέφοντας τον τελεστή του.

Πεδία :

- `private ZSelectItem leftOp;`
  Η αριστερή ιδιότητα της σύγκρισης.

- `private ZSelectItem rightOp;`
  Η δεξιά ιδιότητα της σύγκρισης, στις συγκρίσεις της μορφής "ιδιότητα op ιδιότητα".

- `private String constant;`
  Η σταθερά της σύγκρισης, στις συγκρίσεις της μορφής “ιδιότητα op σταθερά”.

- `private String operator;`
  Ο τελεστής της σύγκρισης.

- `private int type;`
  Ο τύπος του comparison subgoal. Ισούται με :
  - 0 αν το Comparison αντικείμενο δεν έχει αρχικοποιηθεί ακόμη.
  - 1 αν είναι της μορφής “ιδιότητα op σταθερά”.
  - 2 αν είναι της μορφής “ιδιότητα op ιδιότητα ”.

- `private Vector operands;`
  Διάνυσμα από Strings που περιέχει τις ιδιότητες που συμμετέχουν στη σύγκριση.

Μέθοδοι :

- `public Comparison()`
  Constructor της κλάσης για αντικείμενο τύπου 0.

- `public Comparison(ZSelectItem leftOp, String operator, String constant)`
  Constructor της κλάσης για αντικείμενο τύπου 1.

- `public Comparison(ZSelectItem leftOp, String operator, ZSelectItem rightOp)`
  Constructor της κλάσης για αντικείμενο τύπου 2.

- `public Comparison copy()`
  Αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public ZSelectItem getLeftOp()`
  Επιστρέφει το πεδίο leftOp.

- `public ZSelectItem getRightOp()`
  Επιστρέφει το πεδίο rightOp.

- `public String getConstant()`
  Επιστρέφει το πεδίο constant.

- `public String getOperator()`
  Επιστρέφει το πεδίο operator.

- `public int getType()`
  Επιστρέφει το πεδίο type.

- `public Vector getOperands()`
  Επιστρέφει το πεδίο operands.

- `public boolean covers(Comparison comp)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα άλλο αντικείμενο της κλάσης Comparison και ελέγχει εάν η συνθήκη σύγκρισης που αυτό περιέχει καλύπτεται από εκείνη του τρέχοντος αντικειμένου. Π.χ. αν η παράμετρος είναι η σύγκριση "x ≥ 1" και το τρέχον αντικείμενο είναι η σύγκριση "x > 1" ή η "x > 2", η συνάρτηση επιστρέφει true. Αλλιώς, αν το τρέχον αντικείμενο είναι η σύγκριση "x > 0" ή η "x < 3" επιστρέφει false.

- `public static String invertOp(String op)`
  Παίρνει έναν τελεστή σαν παράμετρο και επιστρέφει τον αντίστροφό του.

- `public void print()`
  Τυπώνει τη σύγκριση που περιέχει το τρέχον αντικείμενο.

4.2.3.8 public class Alias

Η κλάση αυτή χρησιμεύει για την αποθήκευση των ιδιοτήτων στο select κομμάτι μίας αντιστοίχησης και των ψευδωνύμων (aliases) τους σε ένα αντικείμενο. Τα aliases παίζουν

πολύ σημαντικό ρόλο στον τρόπο αναπαράστασης των αντιστοιχήσεων σε SQL, διότι εκφράζουν την αντιστοιχία των ιδιοτήτων ανάμεσα σε πίνακες δύο κόμβων. Για παράδειγμα, ας υποθέσουμε ότι στο select κομμάτι της LAV αντιστοίχησης του πίνακα Peer1_A υπάρχει το ζεύγος select ιδιότητας – ψευδωνύμου: Peer2_B.b as a. Αυτό θα σημαίνει ότι η ιδιότητα Peer2_B.b του πίνακα B του κόμβου 2 αντιστοιχεί με την ιδιότητα Peer1_A.a του πίνακα A του κόμβου 1.

Πεδία :

- `private String selectItem;`
  Η ιδιότητα στο select κομμάτι.

- `private String alias;`
  Το ψευδώνυμο που αντιστοιχεί σε αυτήν.

Μέθοδοι :

- `public Alias(String selectItem, String alias)`
  Constructor της κλάσης.

- `public Alias copy()`
  Συνάρτηση που αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public String getSelectItem()`
  Επιστρέφει το πεδίο selectItem.

- `public String getAlias()`
  Επιστρέφει το πεδίο alias.

- `public String siAsAlias(String tableName)`
  Τυπώνει το περιεχόμενο του τρέχοντος αντικειμένου στη μορφή “selectItem as alias”, όπου selectItem και alias τα αντίστοιχα πεδία.

4.2.3.9 public class TableAlias

Η κλάση αυτή χρησιμοποιείται για τον χειρισμό ερωτημάτων και αντιστοιχήσεων που περιέχουν μετονομασμένους πίνακες. Διαφέρει ελαφρά από τη συνώνυμη κλάση του πακέτου GAV γιατί σε κάθε αντικείμενό της αποθηκεύεται και το νέο όνομα που αναθέτει η εκτέλεση της SQLToConj σε κάθε μετονομασμένο πίνακα. Το νέο αυτό όνομα είναι της μορφής X#1, X#2 … , όπου X είναι το πραγματικό όνομα του μετονομασμένου πίνακα. Η αλλαγή αυτή γίνεται για να φαίνεται από το όνομα και μόνο ενός μετονομασμένου πίνακα ποιού πραγματικού πίνακα είναι μετονομασία.

Πεδία :

- `String originalName;`
  Το πραγματικό όνομα του πίνακα, δηλαδή εκείνο που υπάρχει στο σχήμα του αντίστοιχου κόμβου.

- `String aliasedName;`
  Το νέο όνομα που του έχει ανατεθεί στο SQL ερώτημα.

- `String newName;`
  Το νέο όνομα της μορφής X#.., που του ανατίθεται από την SQLToConj. Όπου X είναι το πραγματικό όνομα του πίνακα.

Μέθοδοι :

- `public TableAlias(ZFromItem zfi, int i)`
  Constructor της κλάσης.

- `public TableAlias(String on, String an, String nn)`
  Constructor της κλάσης.

- `public TableAlias copy()`
  Αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public String getOriginalName()`
  Επιστρέφει το πεδίο originalName.

- `public String getAliasedName()`
  Επιστρέφει το πεδίο aliasedName.

- `public String getNewName()`
  Επιστρέφει το πεδίο newName.

4.2.3.10 public class Subgoal

Σε αντικείμενα της κλάσης αυτής αποθηκεύονται μετά την εκτέλεση της SQLToConj οι subgoals του ερωτήματος και των αντιστοιχήσεων.

Πεδία :

- `private String table;`
  Ο πίνακας του subgoal.

- `private Vector attrs;`
  Διάνυσμα από Strings που περιέχει τις ιδιότητες του subgoal, δηλαδή τις ιδιότητες του αντίστοιχου πίνακα.

- `private Vector aliases;`
  Διάνυσμα από αντικείμενα της κλάσης Alias, στα οποία αποθηκεύονται οι ιδιότητες του subgoal που διαθέτουν ψευδώνυμα (aliases). Κάθε ιδιότητα αποθηκεύεται μαζί με το ψευδώνυμό της. Οι ιδιότητες αυτές είναι οι διακεκριμένες μεταβλητές του subgoal, στην περίπτωση που αυτός ανήκει σε μία αντιστοίχηση. Αλλιώς, επειδή στο ερώτημα Q δεν εμφανίζονται ψευδώνυμα στο select κομμάτι, το διάνυσμα aliases για κάθε subgoal του είναι κενό.

- `private String originalTable;`
  Το όνομα του πραγματικού πίνακα του subgoal. Ταυτίζεται με το πεδίο table αν ο πίνακας δεν είναι μετονομασμένος.

Μέθοδοι :

- `public Subgoal(String table)`
  Constructor της κλάσης.

- `public Subgoal copy()`
  Αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public String getTable()`
  Επιστρέφει το πεδίο table.

- `public Vector getAttrs()`
  Επιστρέφει το πεδίο attrs.

- `public Vector getAliases()`
  Επιστρέφει το πεδίο aliases.

- `public String getOriginalTable()`
  Επιστρέφει το πεδίο originalTable.

- `public void setTable(String t)`
  Θέτει στο πεδίο table την τιμή της παραμέτρου t.

- `public void setOriginalTable(String s)`
  Θέτει στο πεδίο originalTable την τιμή της παραμέτρου s.

- `public void addAttr(String s)`
  Προσθέτει στον subgoal την ιδιότητα που δέχεται σαν είσοδο.

- `public void addAlias(Alias a)`
  Προσθέτει ένα αντικείμενο Alias στο πεδίο aliases.

- `public boolean hasAttr(String attr)`
  Επιστρέφει true αν η ιδιότητα attr που δίνεται σαν παράμετρος εμφανίζεται ανάμεσα στις ιδιότητες του subgoal.

- `public void print()`
  Τυπώνει τον subgoal στη μορφή: table(x_1, x_2, …, x_k), όπου table ο πίνακας στο πεδίο table και x_1, x_2, …, x_k οι ιδιότητες του διανύσματος attrs.

4.2.3.11 public class Query

Σε αντικείμενα της κλάσης αυτής αποθηκεύονται μετά την εκτέλεση της SQLToConj το ερώτημα και οι αντιστοιχήσεις. Κάθε ερώτημα ή αντιστοίχηση βρίσκεται σε SQL_CNF μορφή και συντίθεται από τους subgoals (κλάση Subgoal), τους συνδέσμους (κλάση Join), τους constant subgoals (κλάση Constant) και τους comparison subgoals (κλάση Comparison).

Πεδία :

- `private Subgoal head;`
  Η κεφαλή του ερωτήματος / αντιστοίχησης, δηλαδή ένας subgoal που περιέχει τις ιδιότητες που εμφανίζονται στο select κομμάτι του. Το όνομα του subgoal αυτού είναι “main” αν πρόκειται για το ερώτημα Q, ενώ αν πρόκειται για αντιστοίχηση είναι το όνομα του πίνακα στον οποίο ανήκει η αντιστοίχηση.

- `private Vector subgoals;`
  Διάνυσμα από αντικείμενα της κλάσης Subgoal, στο οποίο αποθηκεύονται οι subgoals του ερωτήματος / αντιστοίχησης.

- `private Vector joins;`
  Διάνυσμα από αντικείμενα της κλάσης Join, στο οποίο αποθηκεύονται οι σύνδεσμοι του ερωτήματος / αντιστοίχησης.

- `private Vector constants;`
  Διάνυσμα από αντικείμενα της κλάσης Constant, στο οποίο αποθηκεύονται οι constant subgoals του ερωτήματος / αντιστοίχησης.

- `private Vector comparisons;`
  Διάνυσμα από αντικείμενα της κλάσης Comparison, στο οποίο αποθηκεύονται οι comparison subgoals του ερωτήματος / αντιστοίχησης.

- `private Vector homos;`
  Διάνυσμα από αντικείμενα της κλάσης Homo, στο οποίο αποθηκεύονται οι ομομορφισμοί που έχουν ήδη εφαρμοστεί στη συγκεκριμένη αντιστοίχηση. Αν πρόκειται για το ερώτημα, το διάνυσμα αυτό είναι κενό.

- `private Vector tableAliases;`
  Διάνυσμα από αντικείμενα της κλάσης TableAlias, στο οποίο αποθηκεύονται οι μετονομασμένοι πίνακες του ερωτήματος / αντιστοίχησης.

- `private int MCDnumber;`
  Πεδίο που χρησιμοποιείται για τη δημιουργία του τελικού μεταφρασμένου ερωτήματος από την κλάση FinalSQL.

Μέθοδοι :

- `public Query(Subgoal head)`
  Constructor της κλάσης.

- `public Query copy()`
  Συνάρτηση που αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public Subgoal getHead ()`
  Επιστρέφει το πεδίο head.

- `public Vector getSubgoals()`
  Επιστρέφει το πεδίο subgoals.

- `public Vector getJoins()`
  Επιστρέφει το πεδίο joins.

- `public Vector getConstants()`
  Επιστρέφει το πεδίο constants.

- `public Vector getComparisons()`
  Επιστρέφει το πεδίο comparisons.

- `public Vector getHomos()`
  Επιστρέφει το πεδίο Homos.

- `public int getMCDnumber()`
  Επιστρέφει το πεδίο MCDnumber.

- `public void setMCDnumber(int i)`
  Θέτει στο πεδίο MCDnumber την τιμή της παραμέτρου i.

- `public boolean hasZSI(ZSelectItem zsi)`
  Επιστρέφει true αν στο ερώτημα / αντιστοίχηση εμφανίζεται η ιδιότητα που δίνεται στην παράμετρο zsi, αλλιώς επιστρέφει false.

- `public void addAttrToSubgoal(ZSelectItem zsi)`
  Προσθέτει την ιδιότητα που δίνεται στην παράμετρο zsi στον αντίστοιχο subgoal του ερωτήματος / αντιστοίχησης, δηλαδή σε εκείνον που αντιστοιχεί στον πίνακα της ιδιότητας. Αν ο subgoal αυτός δεν υπάρχει, τότε τον δημιουργεί.

- `public void addWholeTableSub(Query wholeSchema, ZFromItem zfi)`
  Προσθέτει στο ερώτημα έναν νέο subgoal που αντιστοιχεί στον πίνακα που δίνεται στην παράμετρο zfi μαζί με όλες του τις ιδιότητες. Αν ο πίνακας αυτός είναι μετονομασμένος, του αναθέτει έναν αύξοντα αριθμό και τον μετονομάζει (ξανά) σε X#i, όπου X το όνομα του πραγματικού πίνακα και i ο αύξων αριθμός. Έπειτα τυπώνει ένα μήνυμα που ενημερώνει το χρήστη για τη μετονομασία αυτή. Τέλος, αποθηκεύει τον πίνακα σε ένα αντικείμενο TableAlias που εισάγεται στο πεδίο (διάνυσμα) tableAliases.

- `public int whereInSubgoals(String table)`
  Παίρνει σαν παράμετρο ένα όνομα πίνακα και επιστρέφει έναν ακέραιο που δηλώνει τη θέση του αντίστοιχου subgoal στο πεδίο (διάνυσμα) subgoals.

- `public Subgoal getSubgoal(String table)`
  Παίρνει σαν παράμετρο ένα όνομα πίνακα και επιστρέφει το αντίστοιχο αντικείμενο Subgoal.

- `public Vector getSubgoalTables()`
  Επιστρέφει ένα διάνυσμα από Strings που περιέχει όλα τα ονόματα των πινάκων που εμφανίζονται στο ερώτημα / αντιστοίχηση.

- `public Vector getAllAttrs()`
  Επιστρέφει ένα διάνυσμα από Strings που περιέχει όλα τα ονόματα των ιδιοτήτων που εμφανίζονται στο ερώτημα / αντιστοίχηση.

- `public Vector getExistentialAttrs()`
  Επιστρέφει ένα διάνυσμα από Strings που περιέχει όλα τα ονόματα των υπαρξιακών ιδιοτήτων που εμφανίζονται στο ερώτημα / αντιστοίχηση.

- `public Vector getDistinguishedAttrs()`
  Επιστρέφει ένα διάνυσμα από Strings που περιέχει όλα τα ονόματα των διακεκριμένων ιδιοτήτων που εμφανίζονται στο ερώτημα / αντιστοίχηση.

- `public int whereInJoins(ZSelectItem zsi)`
  Παίρνει σαν παράμετρο μία ιδιότητα και επιστρέφει έναν ακέραιο που δηλώνει τη θέση του Join αντικειμένου στο πεδίο (διάνυσμα) joins, στου οποίου το join set εμφανίζεται η ιδιότητα αυτή. Επιστρέφει -1 αν η ιδιότητα δεν βρεθεί σε κανένα join set.

- `public Vector getJoinSetOf(ZSelectItem zsi)`
  Παίρνει σαν παράμετρο μία ιδιότητα και επιστρέφει το join set της σαν διάνυσμα από Strings.

- `public boolean hasJoin(ZSelectItem zsi1, ZSelectItem zsi2)`
  Επιστρέφει true αν στο ερώτημα / αντιστοίχηση οι δύο ιδιότητες που δίνονται σαν παράμετροι έχουν ένα σύνδεσμο μεταξύ τους, δηλαδή αν ανήκουν στο ίδιο join set.

- `public void storeJoin(ZSelectItem leftOp, ZSelectItem rightOp)`
  Δημιουργεί έναν σύνδεσμο ανάμεσα στις δύο ιδιότητες που δίνονται σαν παράμετροι, δηλαδή τις εισάγει στο ίδιο join set του ερωτήματος / αντιστοίχησης.

- `public void storeHomo(ZSelectItem leftOp, ZSelectItem rightOp)`
  Αποθηκεύει στο πεδίο (διάνυσμα) homos τον ομομορφισμό που εξισώνει τις δύο ιδιότητες που δίνονται σαν παράμετροι. Επίσης εφαρμόζει τον ομομορφισμό αυτόν στο τρέχον αντικείμενο (αντιστοίχηση) καλώντας τη συνάρτηση storeJoin με τις ίδιες παραμέτρους.

- `public void addHomos(Vector homos)`
  Η συνάρτηση αυτή εκτελεί την ίδια λειτουργία με την storeHomo, αλλά για ένα διάνυσμα από ομομορφισμούς. Καλείται μόνο στην κλάση FinalSQL, από τη συνάρτηση removeRedundantViews.

- `public void storeConstant(Constant c)`
  Αποθηκεύει στο πεδίο (διάνυσμα) constants την παράμετρο c.

- `public void setAllConstants()`
  Η συνάρτηση αυτή εκτελείται αφού έχει τελειώσει η δημιουργία όλου του ερωτήματος από την SQLToConj. Παίρνει κάθε αντικείμενο Constant που υπάρχει στο πεδίο (διάνυσμα) constants, βρίσκει το join set της ιδιότητάς του, και εισάγει στο πεδίο constants ένα νέο αντικείμενο Constant για κάθε ιδιότητα του join set. Η σταθερά κάθε νέου αντικειμένου είναι ίδια με εκείνη του αρχικού. Αυτό είναι σωστό διότι εφόσον μία ιδιότητα από ένα join set ισούται με κάποια σταθερά, και όλες οι υπόλοιπες ιδιότητές του ισούνται επίσης με την ίδια σταθερά.

- `public Vector getConstantAttrs()`
  Επιστρέφει ένα διάνυσμα από Strings που περιέχει όλες τις ιδιότητες που εμφανίζονται στα αντικείμενα Constant του πεδίου (διανύσματος) constants, όπως αυτό έχει διαμορφωθεί μετά την κλήση της συνάρτησης setAllConstants.

- `public String getConstantOf(String attr)`
  Επιστρέφει τη σταθερά με την οποία ισούται η ιδιότητα attr που δίνεται σαν παράμετρος. Αν δεν υπάρχει τέτοια σταθερά επιστρέφει null.

- `public void storeComparison(Comparison comp)`
  Αποθηκεύει στο πεδίο (διάνυσμα) comparisons την παράμετρο comp.

- `public Vector getComparisonAttrs()`
  Επιστρέφει ένα διάνυσμα από Strings που περιέχει όλες τις ιδιότητες που εμφανίζονται στα αντικείμενα Comparison του πεδίου (διανύσματος) comparisons.

- `public Vector getComparisonsOf(String x, int type)`
  Επιστρέφει ένα διάνυσμα από αντικείμενα της κλάσης Comparison που περιέχει όλους τους comparison subgoals τύπου type στους οποίους εμφανίζεται η ιδιότητα x ή κάποια άλλη ιδιότητα από το join set της x.

- `public Vector Ix(Vector x)`
  Επιστρέφει όλα τα αντικείμενα της κλάσης Comparison, στα οποία εμφανίζονται μόνο ιδιότητες από το διάνυσμα x, που δίνεται σαν παράμετρος. Κατά την κλήση της συνάρτησης αυτής δίνουμε στη θέση της παραμέτρου x το διάνυσμα των ιδιοτήτων F_c και για τους comparison subgoals που επιστρέφονται ελέγχουμε αν ικανοποιείται η ιδιότητα Const2.

- `public boolean entailsComparison(Comparison comp)`
  Επιστρέφει true αν από κάποιον από τους comparison subgoals του τρέχοντος αντικειμένου (αντιστοίχησης) συνάγεται λογικά ο comparison subgoal που δίνεται στην παράμετρο comp.

- `public String getOriginalName(String aliasedName)`
  Επιστρέφει το πραγματικό όνομα του πίνακα που δίνεται σαν παράμετρος.

- `public String getNewName(String aliasedName)`
  Επιστρέφει το νέο όνομα (newName) του πίνακα που δίνεται σαν παράμετρος.

- `public void print()`
  Τυπώνει την κεφαλή και τους subgoals του ερωτήματος / αντιστοίχησης.

- `public void printJoins()`
  Τυπώνει τα join sets του ερωτήματος / αντιστοίχησης.

- `public void printConstants()`
  Τυπώνει τους constant subgoals του ερωτήματος / αντιστοίχησης.

- `public void printComparisons()`
  Τυπώνει τους comparison subgoals του ερωτήματος / αντιστοίχησης.

4.2.3.12 public class Extension

Σε αντικείμενα της κλάσης αυτής αποθηκεύονται οι επεκτάσεις των ομομορφισμών και των συνόλων καλυπτόμενων subgoals που εκτελεί η FormMCDs για την κατασκευή των MCDs.

Πεδία :

- `private Subgoal newSubgoal;`
  Ο νέος subgoal με τον οποίον πρέπει να επεκταθεί το G_c ενός MCD.

- `private Homo newHomo;`
  Ο νέος ομομορφισμός με τον οποίον πρέπει να επεκταθεί το H_c ενός MCD.

- `private int type;`
  Ο τύπος της επέκτασης που μπορεί να πάρει τις τιμές :
  -1, αν δεν υπάρχει επέκταση που να δημιουργεί ένα αποδεκτό MCD.
  0, αν δεν απαιτείται κάποια περαιτέρω επέκταση.
  1, αν απαιτείται επέκταση του συνόλου G_c.
  2, αν απαιτείται επέκταση του ομομορφισμού H_c.

Μέθοδοι :

- `public Extension(int i)`
  Constructor αντικειμένου τύπου -1.

- `public Extension()`
  Constructor αντικειμένου τύπου 0.

- `public Extension(Subgoal newSubgoal)`
  Constructor αντικειμένου τύπου 1. Ο subgoal με τον οποίον πρέπει να επεκταθεί το G_c δίνεται σαν παράμετρος.

- `public Extension(Homo newHomo)`
  Constructor αντικειμένου τύπου 2. Ο ομομορφισμός με τον οποίον πρέπει να επεκταθεί ο H_c δίνεται σαν παράμετρος.

- `public Subgoal getNewSubgoal()`
  Επιστρέφει το πεδίο newSubgoal.

- `public Homo getNewHomo()`
  Επιστρέφει το πεδίο newHomo.

- `public int getType()`
  Επιστρέφει το πεδίο type.

4.2.3.13 public class MCD

Σε αντικείμενα της κλάσης αυτής αποθηκεύονται τα MCDs που παράγει η διαδικασία formMCDs.

Πεδία :

- `private Query view;`
  Η αντιστοίχηση V στην οποία ανήκει το MCD.

- `private Query viewAfterHomo;`
  Το $H_c(V)$, δηλαδή η αντιστοίχηση αφού της έχουν εφαρμοστεί οι ομομορφισμοί που βρέθηκαν από τη διαδικασία formMCDs.

- `private Vector tablesCovered;`
  Διάνυσμα από Strings που περιέχει τα ονόματα των subgoals που καλύπτονται από το MCD.

- `private Mapping map;`
  Η αντιστοιχία πινάκων ερωτήματος – αντιστοίχησης για την οποία βρέθηκε αυτό το MCD.

- `private String name;`
  Το όνομα του MCD. Έχει τη μορφή MCD#... για τα MCDs που παράγονται από την formMCDs, ενώ έχει τη μορφή newMCD#... για τα unified MCDs που παράγονται από τη συνάρτηση removeRedundantMCDs της κλάσης MiniMain.

Μέθοδοι :

- `public MCD()`
  Constructor της κλάσης.

- `public MCD(Query view, Query viewAfterHomo, Vector subsCovered, Mapping map)`
  Constructor της κλάσης.

- `public MCD copy()`
  Αντιγράφει το τρέχον MCD και επιστρέφει ένα όμοιό του.

- `public Query getView()`
  Επιστρέφει το πεδίο view.

- `public Query getViewAfterHomo()`
  Επιστρέφει το πεδίο viewAfterHomo.

- `public Vector getTablesCovered()`
  Επιστρέφει το πεδίο tablesCovered.

- `public String getName()`
  Επιστρέφει το πεδίο name.

- `public int getIndex()`
  Επιστρέφει τον αύξοντα αριθμό του MCD που δηλώνεται στο όνομά του.

- `public void setName(String s)`
  Θέτει στο πεδίο name την τιμή της παραμέτρου.

- `public void setViewAfterHomo(Query q)`
  Θέτει στο πεδίο viewAfterHomo την τιμή της παραμέτρου.

- `public void print()`
  Τυπώνει το όνομα του MCD, το όνομα της αντιστοίχησης στο οποίο ανήκει, τους subgoals του ερωτήματος που καλύπτει και την αντιστοιχία πινάκων που χρησιμοποιεί.

4.2.3.14 public class Mapping

Στην κλάση αυτή αποθηκεύονται οι αντιστοιχίες ανάμεσα στους πίνακες του ερωτήματος και στους πίνακες κάποιας αντιστοίχησης. Δεν αποθηκεύονται σε αυτήν οι GAV και LAV αντιστοιχήσεις, αυτές αποθηκεύονται στην κλάση Query. Σημειώνουμε εδώ ότι στην εξήγηση των πεδίων και των μεθόδων της κλάσης, όπου αναφερόμαστε σε πίνακες της αντιστοίχησης συμπεριλαμβάνουμε και τους dummy πίνακες που εισάγει ο αλγόριθμος εύρεσης των αντιστοιχιών. Για τη χρησιμότητα των αντιστοιχιών πινάκων βλ. παραγράφους 3.3.4 και 3.3.5.

Πεδία :

- `private Vector mappedMainTables;`
  Διάνυσμα από Strings που περιλαμβάνει τους πίνακες του ερωτήματος που μετέχουν στην αντιστοίχηση.

- `private Vector mappedViewTables;`
  Διάνυσμα από Strings που περιλαμβάνει τους πίνακες της αντιστοίχησης που μετέχουν στην αντιστοίχηση.

- `private Vector mappings;`
  Κάθε στοιχείο του διανύσματος αυτού είναι ένα διάνυσμα που περιέχει δύο Strings. Το πρώτο είναι το όνομα ενός πίνακα στο ερώτημα και το δεύτερο το όνομα ενός πίνακα στην αντιστοίχηση. Ο πίνακας του ερωτήματος αντιστοιχεί σε εκείνον της αντιστοίχησης. Κάθε τέτοιο διάνυσμα περιγράφει μία επιμέρους αντιστοιχία πινάκων, δηλαδή ένα τμήμα της συνολικής αντιστοιχίας πινάκων ανάμεσα στο ερώτημα και στην αντιστοίχηση. Όλα τα διθέσια αυτά διανύσματα εισάγονται στο διάνυσμα mappings και συνθέτουν τη συνολική αντιστοιχία που περιλαμβάνει όλους τους πίνακες του ερωτήματος (μαζί με τους αντίστοιχούς τους) που μπορούν να αντιστοιχιστούν σε κάποιον πίνακα της αντιστοίχησης.

Μέθοδοι :

- `public Mapping()`
  Constructor της κλάσης.

- `public Mapping copy()`
  Αντιγράφει το τρέχον αντικείμενο Mapping και επιστρέφει ένα όμοιό του.

- `public void addMapping(String mainTable, String viewTable)`
  Προσθέτει στο τρέχον αντικείμενο Mapping μία επιμέρους αντιστοιχία ανάμεσα στον πίνακα mainTable του ερωτήματος και στον πίνακα viewTable της αντιστοίχησης.

- `public Vector getMappedMainTables()`
  Επιστρέφει το πεδίο mappedMainTables.

- `public Vector getMappedViewTables()`
  Επιστρέφει το πεδίο mappedViewTables.

- `public Vector getMappings()`
  Επιστρέφει το πεδίο mappings.

- `public String getMappingOfTable(String mainTable)`
  Επιστρέφει τον πίνακα της αντιστοίχησης στον οποίον αντιστοιχεί ο πίνακας mainTable του ερωτήματος, με βάση την τρέχουσα αντιστοιχία πινάκων.

- `public String getMappingOfViewTable(String viewTable)`
  Επιστρέφει τον πίνακα του ερωτήματος στον οποίον αντιστοιχεί ο πίνακας viewTable της αντιστοίχησης, με βάση την τρέχουσα αντιστοιχία πινάκων.

- `public boolean isSameWith(Mapping map)`
  Επιστρέφει true αν το τρέχον αντικείμενο Mapping είναι όμοιο με εκείνο που δίνεται στην παράμετρο map.

- `public void print()`
  Τυπώνει την τρέχουσα αντιστοιχία πινάκων.

4.2.3.15 public class MappingAlgorithm

Από την κλάση αυτή εκτελείται ο αλγόριθμος εύρεσης των πιθανών αντιστοιχιών ανάμεσα στους πίνακες του ερωτήματος και στους πίνακες μίας LAV αντιστοίχησης. Η ανάλυση της λειτουργίας του αλγορίθμου μπορεί να αναζητηθεί στο κεφάλαιο της Ανάλυσης και Σχεδίασης, στην παράγραφο 3.3.5.

Πεδία :

- `Vector mainTables;`
  Διάνυσμα από Strings που περιέχει τα ονόματα των πινάκων του ερωτήματος.

- `Vector viewTables;`

  Διάνυσμα από Strings που περιέχει τα ονόματα των πινάκων της αντιστοίχησης. Κατά την εκτέλεση του αλγορίθμου, στο διάνυσμα προστίθενται οι απαιτούμενοι dummy πίνακες.

Μέθοδοι :

- `public MappingAlgorithm(Query main, Query view)`

  Constructor της κλάσης.

- `private Vector groupTables(Vector tables)`

  Η συνάρτηση αυτή παίρνει σαν παράμετρο ένα διάνυσμα από ονόματα πινάκων. Τα ονόματα αυτά είναι ονόματα αντίστοιχων subgoals ενός ερωτήματος ή αντιστοίχησης που έχει μετατραπεί σε SQL_CNF από την κλάση SQLToConj. Γι αυτό και έχουν τη μορφή X ή X#i, όπου X το όνομα ενός πραγματικού πίνακα. Η παρούσα συνάρτηση ομαδοποιεί τους πίνακες που δέχεται σαν είσοδο με βάση το όνομα αυτού του πίνακα. Για κάθε ομάδα πινάκων δημιουργεί ένα αντικείμενο της κλάσης Group, το οποίο και εισάγει στο διάνυσμα που επιστρέφει. Συνεπώς, η συνάρτηση επιστρέφει μία ομαδοποίηση των πινάκων που δέχεται σαν είσοδο.

- `private Group findCorrespondant(Group mainGroup, Vector viewGroups)`

  Η συνάρτηση αυτή παίρνει σαν παραμέτρους μία ομάδα πινάκων A του ερωτήματος, και ένα διάνυσμα από αντικείμενα της κλάσης Group, που περιέχει όλες τις ομάδες πινάκων για μία συγκεκριμένη αντιστοίχηση. Ψάχνει ανάμεσα στις ομάδες αυτές και επιστρέφει εκείνη που αντιστοιχεί στην A, δηλαδή εκείνη που αναφέρεται στον ίδιο πραγματικό πίνακα με την A.

- `private void setWantedSizes(Vector mainGroups, Vector viewGroups)`

  Η συνάρτηση αυτή εκτελείται αφού σχηματιστούν όλες οι ομάδες (groups) πινάκων για το ερώτημα και για μία ορισμένη αντιστοίχηση. Οι ομάδες αυτές είναι αποθηκευμένες αντίστοιχα στα διανύσματα mainGroups και viewGroups. Η συνάρτηση setWantedSizes για κάθε ομάδα A, πινάκων από το πρώτο διάνυσμα τρέχει την findCorrespondant και βρίσκει την αντίστοιχή της, έστω B, στο δεύτερο διάνυσμα. Μετρά πόσα μέλη η ομάδα B περιέχει, και θέτει το αποτέλεσμα στο πεδίο wantedSize της ομάδας A. Για τη χρήση του πεδίου wantedSize βλ. στο αντίστοιχο πεδίο της κλάσης Group.

- `public Vector findAllPossibleMappings()`

  Αυτή είναι η κύρια συνάρτηση της κλάσης, από την οποία επιστρέφονται οι αντιστοιχίες – αποτελέσματα σε ένα διάνυσμα από αντικείμενα της κλάσης Mapping. Στο σώμα της συνάρτησης αρχικά καλούνται οι παραπάνω συναρτήσεις προκειμένου να προστεθούν οι κατάλληλοι dummy πίνακες σε κάθε ομάδα πινάκων. Στη συνέχεια, με τη βοήθεια των συναρτήσεων που ακολουθούν, εκτελείται ο αλγόριθμος αντιστοίχησης όπως περιγράψαμε στο κεφάλαιο Ανάλυσης και Σχεδίασης.

- `private Vector correspondantMappings(Vector mainTables, Vector viewTables)`

  Η συνάρτηση αυτή παίρνει παραμέτρους δύο διανύσματα από Strings. Στο πρώτο διάνυσμα περιέχονται τα ονόματα των πινάκων του ερωτήματος. Στο δεύτερο περιέχονται τα ονόματα των πινάκων της αντιστοίχησης για την οποία αναζητούμε την αντιστοιχία πινάκων, καθώς και τα ονόματα των dummy πινάκων που προστέθηκαν από την findAllPossibleMappings. Λειτουργεί αναδρομικά και επιστρέφει ένα διάνυσμα από αντικείμενα της κλάσης Mapping στο οποίο περιέχονται όλες οι δυνατές και μη αντιστοιχίες πινάκων ανάμεσα στο ερώτημα και την αντιστοίχηση. Οι μη δυνατές αντιστοιχίες είναι εκείνες που αντιστοιχούν δύο ή περισσότερους πίνακες του ερωτήματος στον ίδιο πίνακα της αντιστοίχησης.

- `private Vector correspondantTables(String mainTable, Vector viewTables)`

  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα πίνακα του ερωτήματος (mainTable) και ένα διάνυσμα από Strings που περιέχει τα ονόματα των πινάκων της αντιστοίχησης (viewTables). Επιστρέφει ένα διάνυσμα από Strings που περιέχει τα ονόματα των πινάκων της αντιστοίχησης στους οποίους μπορεί να αντιστοιχιστεί ο mainTable. Οι πίνακες αυτοί είναι εκείνοι που έχουν το ίδιο πραγματικό όνομα πίνακα με τον mainTable.

- `private Vector combineMappings(String firstMainTable, Vector ct, Vector cm)`

  Η συνάρτηση αυτή χρησιμοποιείται στην κατασκευή των αντιστοιχιών πινάκων ως εξής. Παίρνει τρεις παραμέτρους. Η πρώτη από αυτές είναι ένα όνομα πίνακα του ερωτήματος και η δεύτερη το διάνυσμα που επιστρέφεται από την correspondantTables για τον πίνακα αυτόν. Με βάση αυτές τις παραμέτρους κατασκευάζεται ένα τμήμα της αντιστοιχίας πινάκων, που αφορά τον συγκεκριμένο πίνακα. Το τμήμα αυτό προσαρτάται στα αντίστοιχα τμήματα για τους υπόλοιπους

πίνακες του ερωτήματος. Τα τμήματα αυτά δίνονται στην τρίτη παράμετρο της συνάρτησης, που είναι ένα διάνυσμα από αντικείμενα Mapping. Η κλήση της συνάρτησης γίνεται στο σώμα της correspondantMappings ως εξής:

combineMappings(firstMainTable,correspondantTables(firstMainTable, viewTables), correspondantMappings(restMainTables, viewTables)).

- `private boolean containsTwice(Mapping map, String viewTable)`
  Η συνάρτηση αυτή παίρνει παράμετρο μία αντιστοιχία πινάκων (map) και ένα όνομα πίνακα της αντιστοίχησης που μας ενδιαφέρει (viewTable) και ελέγχει αν υπάρχουν δύο ή περισσότεροι πίνακες του ερωτήματος που αντιστοιχούν στον πίνακα αυτόν. Επιστρέφει true αν ναι, αλλιώς false.

- `private boolean mappingOK(Mapping map, Vector viewTables)`
  Η συνάρτηση αυτή τρέχει την παραπάνω για όλους τους πίνακες της αντιστοίχησης και επιστρέφει true αν η containsTwice δεν επιστρέφει true για κανέναν από αυτούς. Τότε η αντιστοιχία map είναι αποδεκτή (βλ. κεφ. Ανάλυση και Σχεδίαση).

4.2.3.16 public class Group

Σε αντικείμενα της κλάσης αυτής αποθηκεύονται οι ομάδες πινάκων που χρησιμοποιεί ο αλγόριθμος εύρεσης των αντιστοιχιών πινάκων.

Πεδία :

- `private int size;`
  Το πλήθος των πινάκων που περιέχονται στην ομάδα.

- `private Vector tables;`
  Διάνυσμα από Strings όπου περιέχονται τα ονόματα των πινάκων της ομάδας.

- `private int wantedSize;`
  Το πεδίο αυτό χρησιμεύει μόνο για τις ομάδες πινάκων του ερωτήματος και όχι των αντιστοιχήσεων. Λαμβάνει τιμή κατά την εκτέλεση της συνάρτησης setWantedSizes της κλάσης MappingAlgorithm. Η τιμή αυτή ισούται με το πλήθος των πινάκων που

περιέχονται στην αντίστοιχη ομάδα πινάκων της αντιστοίχησης για την οποία εκτελείται ο αλγόριθμος. Η αντίστοιχη μίας ομάδας είναι εκείνη που αναφέρεται στον ίδιο πραγματικό πίνακα. Για την ομάδα πινάκων X του ερωτήματος, η διαφορά size – wantedSize δηλώνει τον αριθμό των dummy πινάκων της μορφής X#dummy_i που πρέπει να προστεθούν στο σύνολο των πινάκων της αντιστοίχησης πριν την εκτέλεση των συνδυαστικών συναρτήσεων της κλάσης MappingAlgorithm.

Μέθοδοι :

- `public Group()`
  Constructor της κλάσης.

- `public Group copy()`
  Αντιγράφει το τρέχον αντικείμενο και επιστρέφει ένα όμοιό του.

- `public int getSize()`
  Επιστρέφει το πεδίο size.

- `public Vector getTables()`
  Επιστρέφει το πεδίο tables.

- `public int getWantedSize()`
  Επιστρέφει το πεδίο wantedSize.

- `public void setWantedSize(int k)`
  Θέτει στο πεδίο wantedSize την τιμή της παραμέτρου.

- `public void addTable(String table)`
  Προσθέτει στο πεδίο tables τον πίνακα table και αυξάνει το πεδίο size κατά 1.

- `public void removeTable(String table)`
  Αφαιρεί από το πεδίο tables τον πίνακα table και μειώνει το πεδίο size κατά 1.

- `public void print()`
  Τυπώνει τους πίνακες της ομάδας και το πεδίο size.

- `public void printWantedSize()`
  Τυπώνει τους πίνακες της ομάδας, το πεδίο size και το πεδίο wantedSize.

4.2.3.17 public class CombineMCDs

Από την κλάση αυτή εκτελείται η διαδικασία combineMCDs του MiniCon, χωρίς όμως το τελικό βήμα της κατασκευής του τελικού ερωτήματος. Συγκεκριμένα, από την κλάση CombineMCDs επιστρέφεται ένα σύνολο από υποσύνολα των MCDs που ικανοποιούν τις Ιδιότητες 2.1 και 2.2.

Πεδία :

- `private Vector MCDs;`
  Διάνυσμα από αντικείμενα της κλάσης MCD, στο οποίο περιέχονται τα MCDs που παρήγαγε η διαδικασία formMCDs.

- `private int numOfMainSubs;`
  Ο αριθμός των subgoals του ερωτήματος. Δεν περιλαμβάνονται οι constant και οι comparison subgoals.

- `private Vector subsets;`
  Διάνυσμα από διανύσματα Strings όπου θα αποθηκευτούν τα ζητούμενα υποσύνολα από MCDs.

Μέθοδοι :

- `public CombineMCDs(Vector MCDs, int numOfMainSubs)`
  Constructor της κλάσης.

- `public Vector execute()`
  Συνάρτηση που καλεί τις αναδρομικές συναρτήσεις που εκτελούν τον αλγόριθμο και επιστρέφει το διάνυσμα subsets.

- `private void recursiveFS(Vector tablesCovered, Vector MCDsUsed)`
  Η αναδρομική συνάρτηση (recursive Find Subsets) δημιουργεί τα ζητούμενα υποσύνολα από MCDs. Παίρνει σαν παραμέτρους δύο διανύσματα από Strings. Στο πρώτο από αυτά (tablesCovered) είναι αποθηκευμένα τα ονόματα των πινάκων που καλύπτονται από τον συνδυασμό των MCDs τα ονόματα των οποίων βρίσκονται στο δεύτερο διάνυσμα (MCDsUsed). Η συνάρτηση καλείται αρχικά με κενά διανύσματα στις παραμέτρους. Σε κάθε επανάληψη χρησιμοποιεί τη συνάρτηση emptyIntesection και βρίσκει όλα τα MCDs που δεν καλύπτουν κανέναν από τους subgoals που βρίσκονται στο διάνυσμα tablesCovered. Για καθένα από αυτά καλεί ξανά τον εαυτό της έχοντας όμως προσθέσει στο διάνυσμα tablesCovered τους subgoals που καλύπτει το νέο MCD και στο διάνυσμα MCDsUsed το όνομά του. Όταν σε κάποια επανάληψη η συνάρτηση emptyIntesection δεν επιστρέψει κανένα MCD, τότε γίνεται έλεγχος εάν στο διάνυσμα tablesCovered, όπως έχει διαμορφωθεί μέχρι τότε, περιλαμβάνονται τα ονόματα όλων των subgoals του ερωτήματος. Αν ναι, τότε το αντίστοιχο σύνολο MCDsUsed είναι αποδεκτό γιατί ικανοποιεί τις ιδιότητες 2.1 και 2.2, εάν όχι το σύνολο απορρίπτεται.

- `private Vector emptyIntersection(Vector tablesCovered)`
  Η συνάρτηση αυτή χρησιμοποιείται όπως περιγράψαμε παραπάνω από την recursiveFS. Παίρνει παράμετρο ένα διάνυσμα από Strings με ονόματα πινάκων και επιστρέφει ένα διάνυσμα από Strings με τα ονόματα των MCDs που δεν καλύπτουν κανέναν από αυτούς τους πίνακες. Η συνάρτηση εξασφαλίζει ότι τα υποσύνολα subsets θα ικανοποιούν την ιδιότητα 2.2.

- `private void printSubsets()`
  Η συνάρτηση αυτή τυπώνει τα αποδεκτά υποσύνολα των MCDs.

4.2.3.18 public class ConstantMCDs

Η κλάση αυτή χρησιμοποιείται για τον δεύτερο έλεγχο που επιβάλλει η διαδικασία CombineMCDs όπως περιγράφεται στο κεφ. Ανάλυση και Σχεδίαση. Ο έλεγχος αυτός αναφέρεται στην ικανοποιησιμότητα της ιδιότητας Const3 από τα υποσύνολα των MCDs. Η κλάση ConstantMCDs παίρνει τα υποσύνολα MCDs που επιστρέφει η CombineMCDs και απορρίπτει εκείνα που δεν ικανοποιούν την ιδιότητα αυτή.

Πεδία :

- `private Vector MCDsubsets;`
  Διάνυσμα από διανύσματα Strings στα οποία περιέχονται τα υποσύνολα των MCDs που έχει επιστρέψει η συνάρτηση execute της κλάσης CombineMCDs.

- `private Vector MCDs;`
  Διάνυσμα από αντικείμενα της κλάσης MCD, στο οποίο περιέχονται τα MCDs που έχει κατασκευάσει η FormMCDs.

- `private Query main;`
  Το αρχικό ερώτημα σαν αντικείμενο της κλάσης Query.

Μέθοδοι :

- `public ConstantMCDs(Vector MCDsubsets, Vector MCDs, Query main)`
  Constructor της κλάσης.

- `public Vector execute()`
  Η συνάρτηση αυτή ελέγχει την ιδιότητα Const3 για κάθε υποσύνολο από MCDs.

- `private static MCD MCDThatCovers(String table, Vector subset, Vector MCDs)`
  Η συνάρτηση αυτή παίρνει παραμέτρους ένα όνομα πίνακα (table), ένα διάνυσμα από Strings που περιέχει τα ονόματα των MCDs ενός υποσυνόλου MCDs (subset), καθώς και ένα διάνυσμα με όλα τα δημιουργηθέντα αντικείμενα MCD. Από αυτά επιστρέφει το MCD που καλύπτει τον πίνακα table στο υποσύνολο subset

4.2.3.19 public class FinalSQL

Στην κλάση αυτή εκτελείται το τελευταίο μέρος της διαδικασίας combineMCDs, δηλαδή η συγγραφή του μεταφρασμένου ερωτήματος. Οι κλάσεις CombineMCDs και ConstantMCDs έχουν ανακαλύψει τα υποσύνολα από MCDs που θα χρησιμοποιηθούν στη μετάφραση. Στην FinalSQL εκτελούνται τα εξής βήματα για κάθε τέτοιο υποσύνολο :

1. Συγχωνεύονται τα κατάλληλα MCDs του υποσυνόλου.
2. Μεταφράζεται το select κομμάτι του ερωτήματος.
3. Προστίθενται σαν σύνδεσμοι στο where κομμάτι του Q' οι ομομορφισμοί των H(V) των MCDs του υποσυνόλου.
4. Μεταφράζονται οι σύνδεσμοι του ερωτήματος.
5. Μεταφράζονται οι constant subgoals του ερωτήματος.
6. Μεταφράζονται οι comparison subgoals του ερωτήματος.

Σημειώνουμε για τα βήματα 4, 5 και 6 ότι αν κάποιος σύνδεσμος ή constant / comparison subgoal δεν μπορεί να μεταφραστεί, τότε περιέχεται στο σώμα κάποιας αντιστοίχησης οπότε δεν χάνεται. Αυτό εξασφαλίζεται από τις ιδιότητες στις οποίες υπακούουν τα MCDs. Το ίδιο ισχύει για το βήμα 2 για τις ιδιότητες του select κομματιού που αντιστοιχούν σε σταθερές.

Πεδία :

- `private Vector MCDs;`
  Διάνυσμα από αντικείμενα της κλάσης MCD, στο οποίο περιέχονται τα MCDs που κατασκεύασε η FormMCDs.

- `private Vector MCDsubsets;`
  Διάνυσμα από διανύσματα Strings στα οποία περιέχονται τα υποσύνολα των MCDs που έχει επιστρέψει η CombineMCDs και έχει ελέγξει η ConstantMCDs.

- `private Query main;`
  Το αρχικό ερώτημα σαν αντικείμενο της κλάσης Query.

Μέθοδοι :

- `public FinalSQL(Vector MCDs, Vector MCDsubsets, Query main)`
  Constructor της κλάσης.

- `public void writeSQL()`
  Από τη συνάρτηση αυτή εκτελείται η συγγραφή του τελικού ερωτήματος με τα βήματα που περιγράψαμε παραπάνω.

- `private static Vector removeRedundantViews(Vector subset, Vector MCDs)`

  Η συνάρτηση αυτή εκτελεί τη συγχώνευση των κατάλληλων MCDs του υποσυνόλου subset. Συγχωνεύει τα πολλαπλά MCDs της ίδιας αντιστοίχησης V σε ένα από αυτά, προσθέτοντας σε αυτό τους ομομορφισμούς και των άλλων. Από τη συγχώνευση εξαιρείται η περίπτωση που από τα MCDs καλύπτονται αντίγραφα του ίδιου πίνακα.

- `private static String findAlias(String si, Vector newSubsetMCDs)`

  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας (si) και ένα διάνυσμα από αντικείμενα MCD (newSubsetMCDs). Βρίσκει την αντιστοίχηση που καλύπτει την ιδιότητα στο υποσύνολο αυτό των MCDs και επιστρέφει το ψευδώνυμο της ιδιότητας σε αυτήν.

- `private static String findAlias(String si, Query view)`

  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας (si) και μία αντιστοίχηση (view) και βρίσκει το ψευδώνυμο της ιδιότητας στην αντιστοίχηση.

- `private static String findConstant(String si, Vector newSubsetMCDs)`

  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας (si) και ένα διάνυσμα από αντικείμενα MCD (newSubsetMCDs). Βρίσκει την αντιστοίχηση που καλύπτει την ιδιότητα στο υποσύνολο αυτό των MCDs και ελέγχει αν σε αυτό υπάρχει constant subgoal της μορφής “x = c”, όπου x η ιδιότητα si ή κάποια άλλη ιδιότητα από το join set της si. Αν ναι επιστρέφει τη σταθερά c.

- `private static MCD MCDThatCovers(String table, Vector newSubsetMCDs)`

  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα πίνακα (table) και ένα διάνυσμα από αντικείμενα MCD (newSubsetMCDs), και επιστρέφει το MCD που καλύπτει τον πίνακα στο υποσύνολο αυτό των MCDs.

4.2.4 Package Preprocessing

Στο πακέτο αυτό περιέχονται οι κλάσεις που εκτελούν την προεπεξεργασία των ερωτημάτων. Οι κλάσεις του πακέτου PreProcessing είναι οι εξής : PreMain, PreSubgoal, PreQuery, PreSQLToConj, PreJoin, PreConstant, PreComparison, PreAlias, PreTableAlias. Από αυτές, η PreMain είναι η μόνη κλάση που χρησιμοποιείται μόνο για την προεπεξεργασία. Οι PreSubgoal και PreQuery είναι οι κλάσεις Subgoal και Query αντίστοιχα του πακέτου MiniCon με ορισμένες προσθήκες, ενώ οι υπόλοιπες κλάσεις είναι όμοιες με τις αντίστοιχες του πακέτου MiniCon. Παρακάτω θα αναλύσουμε την κλάση PreMain και τις πρόσθετες λειτουργίες των κλάσεων PreSubgoal και PreQuery.

4.2.4.1 public class PreMain

Στην κλάση αυτή εκτελούνται όλες οι διαδικασίες της προεπεξεργασίας ερωτημάτων. Συγκεκριμένα, γίνεται η εκτίμηση της ποιότητας των μειωμένων ερωτημάτων μέσω της συνάρτησης Sim και η συγγραφή τους μέσω της συνάρτησης cutQuery που αποκόπτει από το αρχικό ερώτημα τα μέρη που δεν μπορούν να μεταφραστούν.

Πεδία :

- `private static String Q_file;`
  Το όνομα του αρχείου που περιέχει το αρχικό ερώτημα Q.

- `private static String sender;`
  Το όνομα του κόμβου που αποστέλλει το ερώτημα.

- `private static String receiver;`
  Το όνομα του κόμβου που λαμβάνει το ερώτημα.

- ```
  public static final int KEY_WEIGHT;
  public static final int SELECT_WEIGHT;
  public static final int JOIN_WEIGHT;
  public static final int CONSTANT_WEIGHT;
  public static final int COMPARISON_WEIGHT;
  public static final int ADDITIONAL_JOINONKEYS_WEIGHT;
  public static final int ADDITIONAL_JOINONNONKEYS_WEIGHT;
  ```
  ```

```
public static final int ADDITIONAL_CONSTANT_WEIGHT;
public static final int ADDITIONAL_COMPARISON_WEIGHT;
```

Τα βάρη w_k, w_s, w_j, w_{const}, w_{comp} και τα αντίστοιχα w_{add} που χρησιμοποιούνται στη μέτρηση της ποιότητας των μειωμένων ερωτημάτων.

Μέθοδοι :

- `public static void main(String[] args)`
  Από τη συνάρτηση main καλούνται οι συναρτήσεις που εκτελούν κατά σειρά τις εξής λειτουργίες. Διαβάζουν από τα αντίστοιχα αρχεία τις ιδιότητες – κλειδιά των κόμβων sender και receiver και μετατρέπουν το ερώτημα Q και τις GAV και LAV αντιστοιχήσεις σε SQL_CNF. Στη συνέχεια υπολογίζουν για GAV και LAV αντιστοιχήσεις χωριστά την τιμή της Mas κάθε subgoal του Q και έπειτα την Sim για το Q. Τέλος, αποκόπτουν από το Q τα κατάλληλα κομμάτια και παράγουν το Q^- για GAV και LAV αντιστοιχήσεις.

- `private static PreQuery schemaOf(String peerName)`
  Η συνάρτηση παίρνει σαν παράμετρο ένα όνομα κόμβου και επιστρέφει ένα αντικείμενο της κλάσης PreQuery στο οποίο αποθηκεύεται το σχήμα του κόμβου, δηλαδή οι πίνακές του με τις αντίστοιχες ιδιότητες.

- `private static Vector keysOf(String peerName)`
  Η συνάρτηση αυτή παίρνει σαν παράμετρο ένα όνομα κόμβου και επιστρέφει ένα διάνυσμα από Strings που περιέχει τις ιδιότητες – κλειδιά του κόμβου αυτού.

- `private static Vector allAvailableGAVMappings(String sender, String receiver, PreQuery wholeSchemaOfReceiver)`
  Η συνάρτηση αυτή χρησιμοποιεί την κλάση PreSQLToConj και μετατρέπει σε SQL_CNF τις GAV αντιστοιχήσεις του sender προς τον receiver. Επιστρέφει ένα διάνυσμα από αντικείμενα PreQuery που περιέχουν το καθένα από μία GAV αντιστοίχηση.

- `private static Vector allAvailableLAVMappings(String sender, String receiver, PreQuery wholeSchemaOfSender)`
  Η αντίστοιχη συνάρτηση με την παραπάνω για LAV αντιστοιχήσεις.

- `private static PreQuery calculateMasForGAV(PreQuery main, Vector GAVMappings, Vector keysOfSender)`

  Όπως θα δούμε στην ανάλυση της κλάσης PreSubgoal, σε κάθε subgoal του ερωτήματος αποθηκεύεται μία τιμή Mas. Η συνάρτηση calculateMasForGAV παίρνει τις απαραίτητες παραμέτρους και επιστρέφει ένα αντικείμενο – ερώτημα PreQuery που είναι όμοιο με το αρχικό με τη μόνη διαφορά ότι για κάθε subgoal του έχει υπολογιστεί και αποθηκευτεί η τιμή Mas με βάση τις διαθέσιμες GAV αντιστοιχήσεις.

- `private static PreQuery calculateMasForLAV(PreQuery main, Vector LAVMappings, Vector keysOfSender)`

  Συνάρτηση αντίστοιχη με την παραπάνω για LAV αντιστοιχήσεις.

- `private static float calculateSimForGAV(PreQuery mainWithGAVMas, Vector keysOfReceiver)`

  Η συνάρτηση αυτή παίρνει παράμετρο το ερώτημα με τις τιμές της Mas που επιστρέφει η calculateMasForGAV (mainWithGAVMas) και υπολογίζει την τιμή Sim για το ερώτημα που θα προκύψει με αν κόψουμε το αρχικό με βάση τις διαθέσιμες GAV αντιστοιχήσεις. Ο υπολογισμός της Sim περιγράφεται στο κεφάλαιο Ανάλυσης και Σχεδίασης.

- `private static float calculateSimForLAV(PreQuery mainWithLAVMas, Vector keysOfReceiver)`

  Η αντίστοιχη συνάρτηση της παραπάνω για τα LAV αντιστοιχήσεις.

- `private static PreQuery cutQuery(PreQuery main, Vector remainingAttrs)`

  Η συνάρτηση αυτή εκτελείται για να δώσει το μειωμένο ερώτημα Q^- για GAV και για LAV αντιστοιχήσεις. Παίρνει παράμετρο το αρχικό ερώτημα Q (main) και ένα διάνυσμα από Strings (remainingAttrs). Στο διάνυσμα αυτό έχουν αποθηκευτεί από την calculateMasForGAV / calculateMasForLAV αντίστοιχα, οι ιδιότητες του ερωτήματος Q που μπορούν να μεταφραστούν με τη βοήθεια των GAV / LAV αντιστοιχήσεων. Οι υπόλοιπες ιδιότητες αποκόπτονται κατάλληλα από το Q και έτσι προκύπτει το Q^-.

- `private static PreQuery findGAVMapping(String subTable, Vector GAVMappings)`
  Η συνάρτηση αυτή είναι βοηθητική για την calculateMasForGAV. Παίρνει παραμέτρους ένα όνομα πίνακα (subTable) και ένα διάνυσμα από αντικείμενα PreQuery (GAVMappings) που περιέχει τις διαθέσιμες GAV αντιστοιχήσεις σε SQL_CNF. Επιστρέφει εκείνη από αυτές, αν υπάρχει, που αντιστοιχεί στον πίνακα subTable.

- `public static Vector copyStringVector(Vector v)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα διάνυσμα από Strings και επιστρέφει ένα όμοιό του.

4.2.4.2 public class PreSubgoal

Η κλάση στην οποία αποθηκεύονται οι subgoals των ερωτημάτων και των αντιστοιχήσεων που μετέχουν στην προεπεξεργασία. Είναι όμοια με την κλάση Subgoal του MiniCon, επεκτεταμένη με τα παρακάτω πεδία και τις αντίστοιχες μεθόδους που χρησιμεύουν για την ανάγνωση και την ανάθεση τιμών σε αυτά.

Πεδία :

- `private int Mas;`
  Η τιμή της συνάρτησης Mas για τον συγκεκριμένο subgoal.

- `private int weightSum;`
  Το άθροισμα των βαρών των ιδιοτήτων του subgoal. Χρησιμεύει για τον υπολογισμό της συνάρτησης Sim.

- `private PreQuery GAVMapping;`
  Η GAV αντιστοίχηση που μεταφράζει τον subgoal.

- `private PreQuery LAVMapping;`
  Η LAV αντιστοίχηση που μεταφράζει τον subgoal.

Μέθοδοι :

- `public int getMas() / public void setMas(int i)`
  Συναρτήσεις που διαβάζουν / αναθέτουν τιμή στο πεδίο Mas.

- `public int getWeightSum() / public void setWeightSum(int i)`
  Συναρτήσεις που διαβάζουν / αναθέτουν τιμή στο πεδίο weightSum.

- `public PreQuery getGAVMapping() / public void setGAVMapping(PreQuery gav)`
  Συναρτήσεις που διαβάζουν / αναθέτουν τιμή στο πεδίο GAVMapping.

- `public PreQuery getLAVMapping() / public void setLAVMapping(PreQuery lav)`
  Συναρτήσεις που διαβάζουν / αναθέτουν τιμή στο πεδίο LAVMapping.

4.2.4.3 public class PreQuery

Στην κλάση αυτή αποθηκεύονται τα ερωτήματα Q και Q' καθώς και οι GAV και LAV αντιστοιχήσεις σε SQL_CNF μορφή. Είναι όμοια με την κλάση Query του MiniCon, επεκτεταμένη με τα εξής πεδία και μεθόδους.

Πεδία :

- `private Vector remainingAttrs;`
  Το πεδίο αυτό χρησιμοποιείται μόνο στο αντικείμενο PreQuery που αντιστοιχεί στο ερώτημα Q. Είναι ένα διάνυσμα από Strings το οποίο περιέχει τα ονόματα των ιδιοτήτων του ερωτήματος Q που μπορούν να μεταφραστούν από τις διαθέσιμες GAV / LAV αντιστοιχήσεις. Οι ιδιότητες αυτές εισάγονται στο διάνυσμα από τις συναρτήσεις calculateMasForGAV / calculateMasForLAV.

Μέθοδοι :

- `public Vector getRemainingAttrs() / public void addRemainingAttr(String attr)`
  Συναρτήσεις που διαβάζουν / προσθέτουν ιδιότητες στο πεδίο remainingAttrs.

- `public boolean appearsInSelectPart(String attr)`
  Η συνάρτηση αυτή, όπως και εκείνες που ακολουθούν, χρησιμοποιούνται για την εκτίμηση της ποιότητας του μειωμένου ερωτήματος, βρίσκοντας τη θέση που κατέχει μία ιδιότητα στο αρχικό ερώτημα. Η συγκεκριμένη συνάρτηση παίρνει παράμετρο ένα όνομα ιδιότητας και επιστρέφει true αν αυτή εμφανίζεται στο select κομμάτι του ερωτήματος / αντιστοίχησης.

- `public boolean appearsAsAlias(String attr)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας και επιστρέφει true αν αυτή εμφανίζεται σαν ψευδώνυμο (alias) στο select κομμάτι του ερωτήματος.

- `public boolean appearsInJoins(String attr)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας και επιστρέφει true αν αυτή εμφανίζεται σε κάποιον σύνδεσμο του ερωτήματος.

- `public boolean appearsInConstants(String attr)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας και επιστρέφει true αν αυτή εμφανίζεται σε κάποιο constant subgoal του ερωτήματος.

- `public boolean appearsInComparisons(String attr)`
  Αντίστοιχη της παραπάνω για comparison subgoals.

- `public boolean appearsInWherePart(String attr)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας και επιστρέφει true αν αυτή εμφανίζεται στο where κομμάτι του ερωτήματος.

- `public int weightInQuery(String attr)`
  Η συνάρτηση αυτή παίρνει παράμετρο ένα όνομα ιδιότητας και χρησιμοποιώντας τις παραπάνω συναρτήσεις και τα βάρη w που ορίζονται στα αντίστοιχα πεδία της κλάσης PreMain, επιστρέφει το βάρος της ιδιότητας στο ερώτημα.

- `public void removeAttribute(String attr)`
  Η συνάρτηση αυτή χρησιμοποιείται στη μείωση του ερωτήματος, δηλαδή στην κατασκευή του Q^-. Αποκόπτει από το ερώτημα τα κομμάτια στα οποία εμφανίζεται η ιδιότητα attr.

- `public void printSQL()`
  Η συνάρτηση αυτή τυπώνει το παρόν ερώτημα σε SQL μορφή.

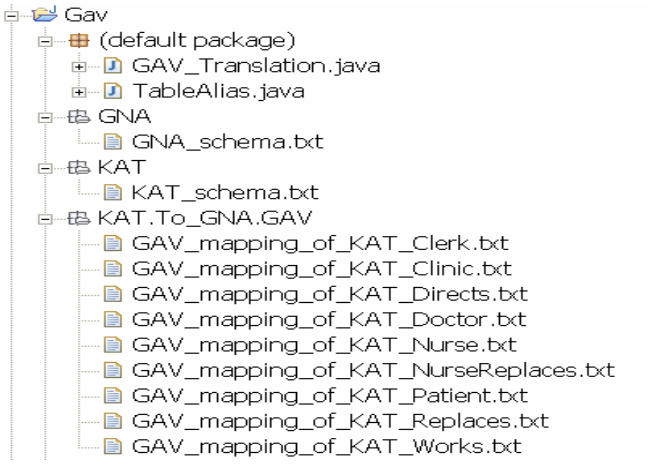
4.3 Εγκατάσταση

4.3.1 Package GAV

Αντιγράφουμε τα αρχεία του πακέτου GAV σε έναν κοινό φάκελο, έστω "GAV_Folder". Στη συνέχεια, δημιουργούμε υποφακέλους στον GAV_Folder που περιέχουν τα σχήματα και τις GAV αντιστοιχήσεις των κόμβων που θα εισάγουμε. Οι φάκελοι αυτοί έχουν τα ονόματα των αντίστοιχων κόμβων, έστω "Peer1", "Peer2"… "PeerK". Στον φάκελο PeerN τώρα, και αντίστοιχα σε καθέναν από "Peer1", "Peer2"… "PeerK", δημιουργούμε ένα αρχείο που περιέχει το σχήμα του κόμβου. Το αρχείο αυτό έχει όνομα "PeerN_schema.txt" και περιέχει ένα SQL ερώτημα που στο select κομμάτι του εμφανίζονται όλες οι ιδιότητες των πινάκων του PeerN και στο from κομμάτι του όλοι οι πίνακες του PeerN. Επίσης, στον φάκελο PeerN δημιουργούμε φακέλους με ονόματα "To_Peer1", "To_Peer2", … , έναν για κάθε κόμβο εκτός του PeerN. Στους φακέλους αυτούς θα αποθηκευτούν οι αντιστοιχήσεις του PeerN με τους υπόλοιπους κόμβους. Σε κάθε φάκελο της μορφής "To_PeerM", δημιουργούμε έναν υποφάκελο με όνομα "GAV", στον οποίο εισάγουμε τις GAV αντιστοιχήσεις του PeerN προς τον PeerM. Για κάθε πίνακα T του PeerN, δημιουργούμε ένα αρχείο με όνομα "GAV_mapping_of_T", και γράφουμε σε αυτό την αντιστοίχηση του πίνακα T, σαν ερώτημα SQL. Σημειώνουμε ότι επειδή η αντιστοίχηση είναι τύπου GAV, το ερώτημα αυτό θα είναι γραμμένο πάνω στο σχήμα του κόμβου PeerM. Για να δηλώσουμε την αντιστοιχία των ιδιοτήτων του πίνακα T – που ανήκει στον PeerN – με τις ιδιότητες στο select κομμάτι του ερωτήματος, χρησιμοποιούμε για τις δεύτερες ψευδώνυμα (aliases), τα οποία δεν πρέπει με κανένα τρόπο να παραλείψουμε. Κάθε ψευδώνυμο είναι όνομα κάποιας ιδιότητας του T. Τέλος δημιουργούμε ένα αρχείο κειμένου (π.χ. Query.txt) στον GAV_Folder, και εισάγουμε σε αυτό το SQL ερώτημα που θέλουμε να μεταφραστεί. Σημειώνουμε εδώ ότι ο ZQL parser που χρησιμοποιείται για την ανάγνωση των ερωτημάτων και των αντιστοιχήσεων απαιτεί κάθε SQL ερώτημα να τελειώνει με ένα ελληνικό ερωτηματικό (;). Επίσης, αν ένα ερώτημα περιέχει strings, αυτά πρέπει να τίθενται σε μονά (‘’) και όχι διπλά εισαγωγικά.

Για το παράδειγμα που χρησιμοποιούμε στο κεφάλαιο 5, μετά την εγκατάσταση η διάρθρωση των φακέλων και των αρχείων θα πρέπει να είναι εκείνη που φαίνεται στο σχήμα 4.1 που δανειστήκαμε από το περιβάλλον eclipse. Σημειώνουμε ότι με την τελεία (.) δηλώνονται οι

υποφάκελοι. Επίσης σημειώνουμε ότι δεν έχουμε δημιουργήσει αντιστοιχήσεις για τον κόμβο GNA, διότι θεωρούμε στα παραδείγματά μας τον χρησιμοποιήσαμε μόνο σαν δέκτη ερωτημάτων και όχι σαν αποστολέα.

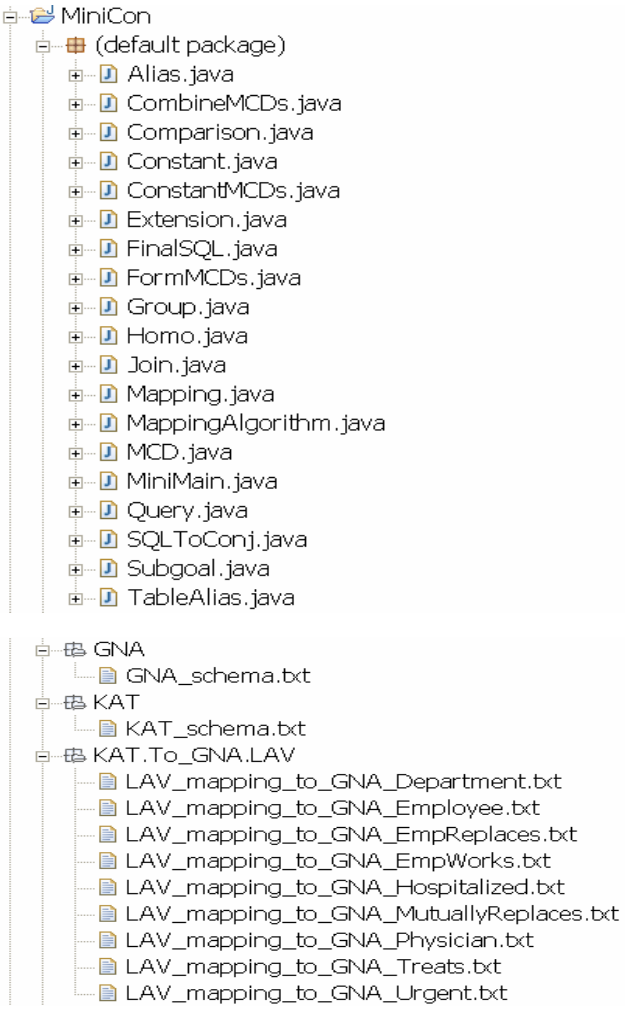


Σχήμα 4.1 : Διάρθρωση πακέτου GAV

4.3.2 Package Minicon

Η διαδικασία είναι πανομοιότυπη με εκείνη της εγκατάστασης του πακέτου GAV, με τη διαφορά ότι σε κάθε φάκελο της μορφής "To_PeerM", δημιουργούμε έναν υποφάκελο με όνομα "LAV" αντί για "GAV", στον οποίο εισάγουμε τις LAV αντιστοιχήσεις του PeerN προς τον PeerM. Για κάθε πίνακα T' του κόμβου PeerM, δημιουργούμε ένα αρχείο με όνομα "LAV_mapping_to_T'" και εισάγουμε σε αυτό την αντιστοίχηση σαν SQL ερώτημα. Το ερώτημα τώρα είναι γραμμένο πάνω στο σχήμα του κόμβου PeerN, ενώ ο πίνακας T' ανήκει στον PeerM. Για να δηλώσουμε την αντιστοιχία των ιδιοτήτων του T' με τις ιδιότητες στο select κομμάτι του ερωτήματος, χρησιμοποιούμε και πάλι ψευδώνυμα, που τώρα παίρνουν τα ονόματά τους από τις ιδιότητες του T'.

Η διάρθρωση των αρχείων φαίνεται για το πακέτο Minicon, φαίνεται στο σχήμα 4.2.

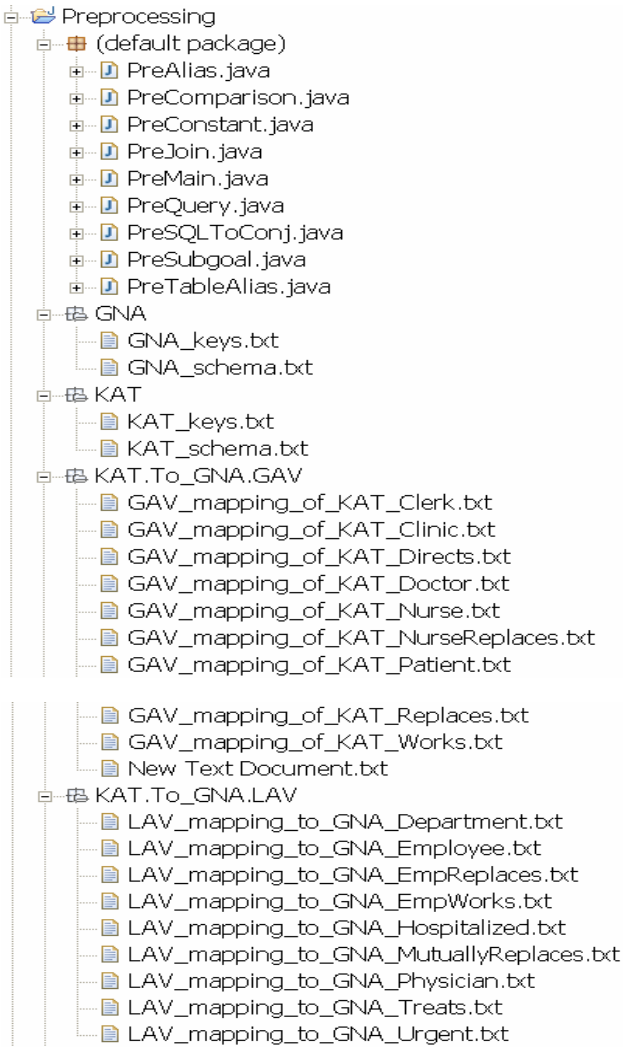


Σχήμα 4.2 : Διάρθρωση πακέτου MiniCon

4.3.3 Package Preprocessing

Για την προεπεξεργασία των ερωτημάτων χρησιμοποιούνται και οι GAV και οι LAV αντιστοιχήσεις των κόμβων. Συνεπώς, αφού αντιγράψουμε τα αρχεία του κώδικα της προεπεξεργασίας σε έναν κοινό φάκελο, δημιουργούμε με τον γνωστό τρόπο τους φακέλους και τα αρχεία τόσο για τις GAV όσο και για τις LAV αντιστοιχήσεις. Επιπλέον, μέσα στον φάκελο που αντιστοιχεί σε κάθε κόμβο, δημιουργούμε ένα αρχείο που περιέχει τις ιδιότητες – κλειδιά του σχήματος του κόμβου. Το αρχείο αυτό έχει όνομα “PeerN_keys.txt”, όπου PeerN το όνομα του κόμβου και περιέχει ένα SQL ερώτημα που στο select κομμάτι του εμφανίζονται οι ιδιότητες αυτές.

Η διάρθρωση των αρχείων φαίνεται για το πακέτο Preprocessing, φαίνεται στο σχήμα 4.3.



Σχήμα 4.3 : Διάρθρωση πακέτου Preprocessing

4.4 Εκτέλεση

Όπως περιγράψαμε στην παράγραφο 4.2, στην υλοποίηση χρησιμοποιήθηκε ένας SQL parser υλοποιημένος σε JAVA, που λέγεται ZQL. Οι κλάσεις του ZQL βρίσκονται στο αρχείο ZQL.jar, το οποίο πρέπει να γίνει import κατά το building της κάθε εφαρμογής. Στο περιβάλλον eclipse, αυτό επιτυγχάνεται κάνοντας δεξί κλικ στο project (φάκελο) που έχουμε δημιουργήσει για την κάθε εφαρμογή, επιλέγοντας "Build Path" → "Add external archives", και βρίσκοντας το αρχείο ZQL.jar .

Στη συνέχεια, μπορούμε να εκτελέσουμε την κάθε εφαρμογή δίνοντας τρεις παραμέτρους χωρισμένες με κενό, που είναι οι εξής και για τις τρεις εφαρμογές :

1. Το όνομα του αρχείου, π.χ. "Query.txt", στο οποίο έχουμε γράψει το SQL ερώτημα που θα τεθεί προς μετάφραση ή προεπεξεργασία. Το ερώτημα πρέπει βέβαια να είναι γραμμένο πάνω στο σχήμα του κόμβου που το αποστέλλει.

2. Το όνομα του κόμβου που αποστέλλει το ερώτημα, π.χ. “ΚΑΤ”.
3. Το όνομα του κόμβου προς τον οποίο πρέπει να μεταφραστεί το ερώτημα, π.χ. “GNA”.

Στο περιβάλλον eclipse, οι παράμετροι δίνονται από το μενού “Run”, επιλέγοντας “Run” → “Run” → “Arguments”, για την αντίστοιχη εκτελέσιμη κλάση.

5

Έλεγχος

5.1 Μεθοδολογία Ελέγχου

Ο έλεγχος της υλοποίησης θα γίνει μέσα από την εκτέλεση παραδειγμάτων που καλύπτουν ένα ευρύ φάσμα από διαφορετικούς τύπους και συνδυασμούς ερωτημάτων και αντιστοιχήσεων. Για τον έλεγχο εισάγουμε δύο κόμβους – νοσοκομεία, που αντιστοιχούν στους Κόμβους 1 και 2 της παραγράφου 3.3.4. Ο Κόμβος 1 θα είναι το νοσοκομείο ΚΑΤ και ο Κόμβος 2 το νοσοκομείο ΓΝΑ. Παρακάτω ορίζουμε ξανά τα σχήματά τους χρησιμοποιώντας αγγλικά ονόματα και δημιουργούμε GAV και LAV αντιστοιχήσεις μεταξύ τους.

Νοσοκομείο ΚΑΤ (Κόμβος 1)

KAT_Doctor (doctorID, rank, name, surname, address, phoneNum)

KAT_Nurse (nurseID, clinicID, name, surname, address, phoneNum)

KAT_Clerk (clerkID, clinicID, speciality, name, surname, address, phoneNum)

KAT_Clinic (clinicID, name, buildingName, floor, bedsNum)

KAT_Works (doctorID, clinicID)

KAT_Directs (doctorID, clinicID)

KAT_Patient (patientID, doctorID, name, surname, address, phoneNum)

KAT_Replaces (doctorID, substituteID)

KAT_NurseReplaces (nurseID, substituteID)

Νοσοκομείο ΓΝΑ (Κόμβος 2)

GNA_Physician (physicianID, departmentID, grade, name, surname, address, phoneNum)

GNA_Employee (employeeID, position, speciality, name, surname, address, phoneNum)

GNA_Department (departmentID, directorID, name, building, floor, beds)

GNA_EmpWorks (employeeID, departmentID)

GNA_Hospitalized (hospitalizedID, name, surname, address, phoneNum)

GNA_Treats (physicianID, hospitalizedID)

GNA_MutuallyReplaces (physicianID, physicianName, physicianSurname, substituteName, substituteSurname)

GNA_EmpReplaces (employeeID, substituteID)

GNA_Urgent (infirmaryID, name, building, floor)

Θα θεωρήσουμε ότι τα ερωτήματα αποστέλλονται μόνο από τον κόμβο ΚΑΤ, γι' αυτό και θα κατασκευάσουμε αντιστοιχήσεις μόνο για τον κόμβο αυτό. Οι αντιστοιχήσεις αυτές σε μορφή ερωτημάτων SQL, είναι οι εξής :

GAV αντιστοιχήσεις

KAT_Doctor

```
select GNA_Physician.physicianID as doctorID,
      GNA_Physician.grade as rank,
      GNA_Physician.name as name,
      GNA_Physician.surname as surname,
      GNA_Physician.address as address,
      GNA_Physician.phoneNum as phoneNum
from GNA_Physician;
```

KAT_Nurse

```
select GNA_Employee.employeeID as nurseID,
      GNA_EmpWorks.departmentID as clinicID,
      GNA_Employee.name as name,
      GNA_Employee.surname as surname,
      GNA_Employee.address as address,
      GNA_Employee.phoneNum as phoneNum
from GNA_Employee, GNA_EmpWorks
where GNA_Employee.position = 'Nurse' and
      GNA_EmpWorks.employeeID = GNA_Employee.employeeID;
```

KAT_Clerk

```
select GNA_Employee.employeeID as clerkID,
      GNA_EmpWorks.departmentID as clinicID,
      GNA_Employee.speciality as speciality,
      GNA_Employee.name as name,
      GNA_Employee.surname as surname,
      GNA_Employee.address as address,
      GNA_Employee.phoneNum as phoneNum
from GNA_Employee, GNA_EmpWorks
where GNA_Employee.position = 'Clerk' and
      GNA_EmpWorks.employeeID = GNA_Employee.employeeID;
```

KAT_Clinic

```
select GNA_Department.departmentID as clinicID,
      GNA_Department.name as name,
      GNA_Department.building as buildingName,
      GNA_Department.floor as floor,
      GNA_Department.beds as bedsNum
from GNA_Department;
```

KAT_Works

Δεν υπάρχει GAV αντιστοίχηση γι' αυτόν τον πίνακα.

KAT_Directs

```
select GNA_Department.directorID as doctorID,
      GNA_Department.departmentID as clinicID
from GNA_Department;
```

KAT_Patient

```
select GNA_Hospitalized.hospitalizedID as patientID,
      GNA_Treats.physicianID as doctorID,
      GNA_Hospitalized.name as name,
      GNA_Hospitalized.surname as surname,
      GNA_Hospitalized.address as address,
      GNA_Hospitalized.phoneNum as phoneNum
from GNA_Hospitalized, GNA_Treats
where GNA_Hospitalized.hospitalizedID = GNA_Treats.hospitalizedID;
```

KAT_Replaces

Δεν υπάρχει GAV αντιστοίχηση γι' αυτόν τον πίνακα.

KAT_NurseReplaces

```
select GNA_EmpReplaces.employeeID as nurseID,
      GNA_EmpReplaces.substituteID as substituteID
from GNA_EmpReplaces, GNA_Employee
where GNA_EmpReplaces.employeeID = GNA_Employee.employeeID and
      GNA_Employee.position = 'Nurse';
```

LAV αντιστοιχήσεις

GNA_Physician

```
select KAT_Doctor.doctorID as physicianID,
      KAT_Works.clinicID as departmentID,
      KAT_Doctor.rank as grade,
      KAT_Doctor.name as name,
      KAT_Doctor.surname as surname,
      KAT_Doctor.address as address,
      KAT_Doctor.phoneNum as phoneNum
```

```
from KAT_Doctor, KAT_Works
where KAT_Doctor.doctorID = KAT_Works.doctorID;
```

GNA_Employee

```
select KAT_Nurse.nurseID as employeeID,
      KAT_Nurse.name as name,
      KAT_Nurse.surname as surname,
      KAT_Nurse.address as address,
      KAT_Nurse.phoneNum as phoneNum
from KAT_Nurse;
```

(Η αντιστοίχηση αυτή είναι ελλιπής διότι αντιστοιχίζει τον πίνακα με όλους τους εργαζομένους του νοσοκομείου ΓΝΑ με τον πίνακα μόνο των νοσοκόμων του ΚΑΤ, παραλείποντας τους υπαλλήλους (clerks))

GNA_Department

```
select KAT_Clinic.clinicID as departmentID,
      KAT_Directs.doctorID as directorID,
      KAT_Clinic.name as name,
      KAT_Clinic.buildingName as building,
      KAT_Clinic.floor as floor,
      KAT_Clinic.bedsNum as beds
from KAT_Clinic, KAT_Directs
where KAT_Clinic.clinicID = KAT_Directs.clinicID;
```

GNA_EmpWorks

```
select KAT_Nurse.nurseID as employeeID,
      KAT_Nurse.clinicID as departmentID
from KAT_Nurse;
```

(Η αντιστοίχηση αυτή είναι ελλιπής διότι αντιστοιχίζει τις ιδιότητες του πίνακα GNA_EmpWorks που αναφέρεται σε όλους τους εργαζομένους του νοσοκομείου ΓΝΑ, νοσοκόμους και υπαλλήλους, σε ιδιότητες του πίνακα KAT_Nurse που αναφέρεται μόνο σε νοσοκόμους.)

GNA_Hospitalized

```
select KAT_Patient.patientID as hospitalizedID,
      KAT_Patient.name as name,
      KAT_Patient.surname as surname,
      KAT_Patient.address as address,
      KAT_Patient.phoneNum as phoneNum
from KAT_Patient;
```

GNA_Treats

```
select KAT_Patient.doctorID as physicianID,
      KAT_Patient.patientID as hospitalizedID
from KAT_Patient;
```

GNA_MutuallyReplaces

```
select D1.doctorID as physicianID,
      D1.name as physicianName,
      D1.surname as physicianSurname,
      D2.name as substituteName,
      D2.surname as substituteSurname
from  KAT_Replaces R1, KAT_Replaces R2,
      KAT_Doctor D1, KAT_Doctor D2
where R1.doctorID = R2.substituteID and
      R2.doctorID = R1.substituteID and
      R1.doctorID = D1.doctorID and
      R2.doctorID = D2.doctorID;
```

GNA_EmpReplaces

```
select KAT_NurseReplaces.nurseID as employeeID,
      KAT_NurseReplaces.substituteID as substituteID
from KAT_NurseReplaces;
```

(Η αντιστοίχηση αυτή είναι ελλιπής διότι αντιστοιχίζει τις ιδιότητες του πίνακα GNA_EmpReplaces που αναφέρεται σε όλους τους εργαζομένους του νοσοκομείου ΓΝΑ, νοσοκόμους και υπαλλήλους, σε ιδιότητες του πίνακα KAT_NurseReplaces που αναφέρεται μόνο σε νοσοκόμους. Ωστόσο, είναι η καλύτερη που μπορεί να δημιουργηθεί)

GNA_Urgent

Για τον πίνακα αυτόν θα θεωρήσουμε αρχικά ότι δεν υπάρχει LAV αντιστοίχηση. Στη συνέχεια θα εισάγουμε διάφορες LAV αντιστοιχήσεις για να επιδείξουμε τις δυνατότητες του αλγορίθμου Minicon.

Σημειώνουμε ότι κάθε LAV αντιστοίχηση θα αντιστοιχίζει ιδιότητες του πίνακα αυτού με ιδιότητες του πίνακα KAT_Clinic ο οποίος καλύπτεται ήδη χάρις τη LAV αντιστοίχηση του πίνακα GNA_Department. Συνεπώς, η ύπαρξη ή όχι LAV αντιστοίχησης για τον πίνακα GNA_Urgent δεν θα επηρεάσει το κομμάτι της προεπεξεργασίας.

5.2 Αναλυτική παρουσίαση έλεγχου

5.2.1 Έλεγχος πακέτου GAV

5.2.1.1 Παράδειγμα GAV.1

Εμφάνιση τηλεφωνικού καταλόγου των ιατρών.

Αρχικό ερώτημα:

```
select KAT_Doctor.name,
      KAT_Doctor.surname,
      KAT_Doctor.phoneNum
from KAT_Doctor;
```

Μεταφρασμένο ερώτημα:

```
select GNA_Physician.name,
      GNA_Physician.surname,
      GNA_Physician.phoneNum
from GNA_Physician
```

5.2.1.2 Παράδειγμα GAV.2

Εμφάνιση των μεγάλων κλινικών του κτιρίου Α.

Το ερώτημα περιέχει σταθερές και συγκρίσεις.

Αρχικό ερώτημα:

```
select KAT_Clinic.name,
      KAT_Clinic.buildingName,
      KAT_Clinic.floor
from KAT_Clinic
where KAT_Clinic.buildingName = 'A' and
      KAT_Clinic.bedsNum > 80;
```

Μεταφρασμένο ερώτημα:

```
select GNA_Department.name,
      GNA_Department.building,
      GNA_Department.floor
from GNA_Department
where ((GNA_Department.building = 'A') AND
      (GNA_Department.beds > 80))
```

5.2.1.3 Παράδειγμα GAV.3

Εμφάνιση των κωδικών και των ονοματεπωνύμων των ασθενών που τους έχει αναλάβει ο ιατρός με κωδικό 1.

Το ερώτημα περιέχει σταθερές και μεταφράζεται με αντιστοιχήσεις που εισάγουν επιπλέον συνδέσμους στο where κομμάτι.

Αρχικό ερώτημα:

```
select KAT_Patient.patientID,
      KAT_Patient.name,
      KAT_Patient.surname
from KAT_Patient
where KAT_Patient.doctorID = 1;
```

Μεταφρασμένο ερώτημα:

```
select GNA_Hospitalized.hospitalizedID,
      GNA_Hospitalized.name,
      GNA_Hospitalized.surname
```

```
from GNA_Hospitalized, GNA_Treats
where ((GNA_Hospitalized.hospitalizedID = GNA_Treats.hospitalizedID)
      AND (GNA_Treats.physicianID = 1))
```

5.2.1.4 Παράδειγμα GAV.4

Εμφάνιση των ονοματεπωνύμων των ασθενών που τους επιβλέπει ο ιατρός Παύλος Ιωάννου.

Αρχικό ερώτημα:

```
select KAT_Patient.name,
      KAT_Patient.surname
from KAT_Doctor, KAT_Patient
where KAT_Doctor.doctorID = KAT_Patient.doctorID and
      KAT_Doctor.name = 'Pavlos' and
      KAT_Doctor.surname = 'Ioannou';
```

Μεταφρασμένο ερώτημα:

```
select GNA_Hospitalized.name,
      GNA_Hospitalized.surname
from GNA_Hospitalized, GNA_Treats, GNA_Physician
where ((GNA_Hospitalized.hospitalizedID = GNA_Treats.hospitalizedID)
      AND ((GNA_Physician.physicianID = GNA_Treats.physicianID)
      AND (GNA_Physician.name = 'Pavlos')
      AND (GNA_Physician.surname = 'Ioannou')))
```

5.2.1.5 Παράδειγμα GAV.5

Εμφάνιση των ονοματεπωνύμων των λογιστών κάθε κλινικής.

Η μετάφραση εισάγει στο ερώτημα τόσο επιπλέον συνδέσμους όσο και επιπλέον σταθερές.

Αρχικό ερώτημα:

```
select KAT_Clinic.name,
      KAT_Clerk.name,
      KAT_Clerk.surname
```

```
from KAT_Clinic, KAT_Clerk
where KAT_Clinic.clinicID = KAT_Clerk.clinicID and
      KAT_Clerk.speciality = 'Bookkeeper';
```

Μεταφρασμένο ερώτημα:

```
select GNA_Department.name,
      GNA_Employee.name,
      GNA_Employee.surname
from GNA_Department, GNA_Employee, GNA_EmpWorks
where (((GNA_Employee.position = 'Clerk') AND
      (GNA_EmpWorks.employeeID = GNA_Employee.employeeID)) AND
      ((GNA_Department.departmentID = GNA_EmpWorks.departmentID) AND
      (GNA_Employee.speciality = 'Bookkeeper')))
```

5.2.1.6 Παράδειγμα GAV.6

Εμφάνιση των κωδικών των νοσοκόμων που αλληλοαντικαθιστούνται.

Το ερώτημα περιέχει self – joins τα οποία αναπαράγονται μετονομάζοντας και τους πίνακες του μεταφρασμένου ερωτήματος.

Αρχικό ερώτημα:

```
select NR1.nurseID,
      NR2.nurseID
from KAT_NurseReplaces NR1, KAT_NurseReplaces NR2
where NR1.nurseID = NR2.substituteID and
      NR2.nurseID = NR1.substituteID;
```

Μεταφρασμένο ερώτημα:

```
select GNA_EmpReplaces_NR1.employeeID,
      GNA_EmpReplaces_NR2.employeeID
from GNA_EmpReplaces GNA_EmpReplaces_NR1,
      GNA_Employee GNA_Employee_NR1,
      GNA_EmpReplaces GNA_EmpReplaces_NR2,
      GNA_Employee GNA_Employee_NR2
where ((((GNA_EmpReplaces_NR1.employeeID =
```

```
                    GNA_Employee_NR1.employeeID) AND
      (GNA_Employee_NR1.position = 'Nurse')) AND
      ((GNA_EmpReplaces_NR2.employeeID =
                    GNA_Employee_NR2.employeeID) AND
      (GNA_Employee_NR2.position = 'Nurse'))) AND
      ((GNA_EmpReplaces_NR1.employeeID =
                    GNA_EmpReplaces_NR2.substituteID) AND
      (GNA_EmpReplaces_NR2.employeeID =
                    GNA_EmpReplaces_NR1.substituteID)))
```

5.2.1.7 Παράδειγμα GAV.7

Εμφάνιση των επωνύμων των νοσοκόμων που αλληλοαντικαθιστούνται.

Αρχικό ερώτημα:

```
select N1.surname,
      N2.surname
from KAT_NurseReplaces NR1, KAT_NurseReplaces NR2,
      KAT_Nurse N1, KAT_Nurse N2
where NR1.nurseID = NR2.substituteID and
      NR2.nurseID = NR1.substituteID and
      NR1.nurseID = N1.nurseID and
      NR2.nurseID = N2.nurseID;
```

Μεταφρασμένο ερώτημα:

```
select GNA_Employee_N1.surname,
      GNA_Employee_N2.surname
from GNA_Employee GNA_Employee_N1,
      GNA_EmpWorks GNA_EmpWorks_N1,
      GNA_Employee GNA_Employee_N2,
      GNA_EmpWorks GNA_EmpWorks_N2,
      GNA_EmpReplaces GNA_EmpReplaces_NR1,
      GNA_Employee GNA_Employee_NR1,
      GNA_EmpReplaces GNA_EmpReplaces_NR2,
      GNA_Employee GNA_Employee_NR2
```

```
where ((((GNA_Employee_N1.position = 'Nurse')
      AND (GNA_EmpWorks_N1.employeeID = GNA_Employee_N1.employeeID))
      AND ((GNA_Employee_N2.position = 'Nurse')
      AND (GNA_EmpWorks_N2.employeeID = GNA_Employee_N2.employeeID)))
      AND ((((GNA_EmpReplaces_NR1.employeeID =
                        GNA_EmpReplaces_NR2.substituteID)
      AND ((GNA_EmpReplaces_NR1.employeeID =
                        GNA_Employee_NR1.employeeID)
      AND (GNA_Employee_NR1.position = 'Nurse')))
      AND ((GNA_EmpReplaces_NR2.employeeID =
                        GNA_Employee_NR2.employeeID)
      AND (GNA_Employee_NR2.position = 'Nurse')))
      AND (GNA_EmpReplaces_NR2.employeeID =
                        GNA_EmpReplaces_NR1.substituteID)))
```

5.2.2 Έλεγχος πακέτου MiniCon

5.2.2.1 Παράδειγμα MiniCon.1

Εμφάνιση των στοιχείων των ιατρών της ορθοπεδικής κλινικής.

Αρχικό ερώτημα:

```
select KAT_Doctor.name,
      KAT_Doctor.surname,
      KAT_Doctor.rank,
      KAT_Doctor.address,
      KAT_Doctor.phoneNum
from KAT_Doctor, KAT_Works, KAT_Clinic
where KAT_Doctor.doctorID = KAT_Works.doctorID and
      KAT_Works.clinicID = KAT_Clinic.clinicID and
      KAT_Clinic.name = 'Orthopaedic';
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Physician#0.name,
       GNA_Physician#0.surname,
       GNA_Physician#0.grade,
       GNA_Physician#0.address,
       GNA_Physician#0.phoneNum
FROM GNA_Physician as GNA_Physician#0,
       GNA_Department as GNA_Department#1
WHERE GNA_Physician#0.departmentID = GNA_Department#1.departmentID
       AND GNA_Department#1.name = 'Orthopaedic'
```

5.2.2.2 Παράδειγμα MiniCon.2

Εμφάνιση των ονομάτων των κλινικών και των διευθυντών τους.

Το ερώτημα μεταφράζεται σαν ένωση (union) ερωτημάτων διότι οι ιδιότητες του πίνακα KAT_Doctor μεταφράζονται μέσω LAV αντιστοιχήσεων τόσο σε ιδιότητες του πίνακα GNA_Physician, όσο και του πίνακα GNA_MutuallyReplaces.

Αρχικό ερώτημα:

```
select KAT_Clinic.name,
       KAT_Doctor.name,
       KAT_Doctor.surname
from KAT_Clinic, KAT_Directs, KAT_Doctor
where KAT_Clinic.clinicID = KAT_Directs.clinicID and
       KAT_Directs.doctorID = KAT_Doctor.doctorID;
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Department#0.name,
       GNA_Physician#1.name,
       GNA_Physician#1.surname
FROM GNA_Department as GNA_Department#0,
       GNA_Physician as GNA_Physician#1
WHERE GNA_Department#0.directorID = GNA_Physician#1.physicianID

UNION
```

```
SELECT GNA_Department#0.name,
      GNA_MutuallyReplaces#1.physicianName,
      GNA_MutuallyReplaces#1.physicianSurname
FROM GNA_Department as GNA_Department#0,
      GNA_MutuallyReplaces as GNA_MutuallyReplaces#1
WHERE GNA_Department#0.directorID =
                              GNA_MutuallyReplaces#1.physicianID
```

5.2.2.3 Παράδειγμα MiniCon.3

Εμφάνιση των κωδικών των νοσοκόμων που αλληλοαντικαθιστούνται.

Το ερώτημα περιέχει self – joins που αναδημιουργούνται χάρις τον αλγόριθμο εύρεσης αντιστοιχίας πινάκων. Συγκεκριμένα, ο πίνακας KAT_NurseReplaces στη LAV αντιστοίχηση του πίνακα GNA_EmpReplaces, αντιστοιχίζεται την πρώτη φορά με τον πίνακα NR1 του ερωτήματος, και τη δεύτερη με τον πίνακα NR2, δημιουργώντας δύο διαφορετικά MCDs.

Αρχικό ερώτημα:

```
select NR1.nurseID,
      NR2.nurseID
from KAT_NurseReplaces NR1, KAT_NurseReplaces NR2
where NR1.nurseID = NR2.substituteID and
      NR2.nurseID = NR1.substituteID;
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_EmpReplaces#0.employeeID,
      GNA_EmpReplaces#1.employeeID
FROM GNA_EmpReplaces as GNA_EmpReplaces#0,
      GNA_EmpReplaces as GNA_EmpReplaces#1
WHERE GNA_EmpReplaces#0.employeeID = GNA_EmpReplaces#1.substituteID
  AND GNA_EmpReplaces#1.employeeID = GNA_EmpReplaces#0.substituteID
```

5.2.2.4 Παράδειγμα MiniCon.4

Εμφάνιση των ονοματεπωνύμων των νοσοκόμων που αλληλοαντικαθιστούνται.

Αρχικό ερώτημα:

```
select N1.name,
       N1.surname,
       N2.name,
       N2.surname
from KAT_NurseReplaces NR1, KAT_NurseReplaces NR2,
       KAT_Nurse N1, KAT_Nurse N2
where NR1.nurseID = NR2.substituteID and
       NR2.nurseID = NR1.substituteID and
       N1.nurseID = NR1.nurseID and
       N2.nurseID = NR2.nurseID;
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Employee#2.name,
       GNA_Employee#2.surname,
       GNA_Employee#3.name,
       GNA_Employee#3.surname
FROM GNA_Employee as GNA_Employee#2,
       GNA_Employee as GNA_Employee#3,
       GNA_EmpReplaces as GNA_EmpReplaces#0,
       GNA_EmpReplaces as GNA_EmpReplaces#1
WHERE GNA_EmpReplaces#0.employeeID =
                              GNA_EmpReplaces#1.substituteID AND
       GNA_EmpReplaces#1.substituteID = GNA_Employee#2.employeeID AND
       GNA_EmpReplaces#1.employeeID =
                              GNA_EmpReplaces#0.substituteID AND
       GNA_EmpReplaces#0.substituteID = GNA_Employee#3.employeeID
```

5.2.2.5 Παράδειγμα MiniCon.5

Εμφάνιση των ονοματεπωνύμων των ιατρών που αλληλοαντικαθιστούνται.

Το ερώτημα αυτό χρησιμοποιήθηκε σαν παράδειγμα στην παράγραφο 3.3.4 για την επεξήγηση της αναγκαιότητας του αλγορίθμου εύρεσης αντιστοιχιών πινάκων. Περιέχει self – joins τα οποία όμως δεν εμφανίζονται στο μεταφρασμένο ερώτημα διότι περιέχονται στη LAV αντιστοίχηση του πίνακα GNA_MutuallyReplaces. Επίσης το μεταφρασμένο ερώτημα είναι ένωση δύο ερωτημάτων διότι οι ιδιότητες doctorID, name και surname του πίνακα KAT_Doctor μπορούν να μεταφραστούν από τη LAV αντιστοίχηση τόσο του πίνακα GNA_Physician, όσο και του πίνακα GNA_MutuallyReplaces.

Αρχικό ερώτημα:

```
select D1.doctorID,
      D1.name,
      D1.surname,
      D2.name,
      D2.surname
from KAT_Replaces R1, KAT_Replaces R2,
      KAT_Doctor D1, KAT_Doctor D2
where R1.doctorID = R2.substituteID and
      R2.doctorID = R1.substituteID and
      R1.doctorID = D1.doctorID and
      R2.doctorID = D2.doctorID;
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Physician#1.physicianID,
      GNA_Physician#1.name,
      GNA_Physician#1.surname,
      GNA_MutuallyReplaces#0.substituteName,
      GNA_MutuallyReplaces#0.substituteSurname
FROM GNA_Physician as GNA_Physician#1,
      GNA_MutuallyReplaces as GNA_MutuallyReplaces#0
WHERE GNA_MutuallyReplaces#0.physicianID =
                              GNA_Physician#1.physicianID

UNION
```

```
SELECT GNA_MutuallyReplaces#1.physicianID,
       GNA_MutuallyReplaces#1.physicianName,
       GNA_MutuallyReplaces#1.physicianSurname,
       GNA_MutuallyReplaces#0.substituteName,
       GNA_MutuallyReplaces#0.substituteSurname
FROM GNA_MutuallyReplaces as GNA_MutuallyReplaces#1,
       GNA_MutuallyReplaces as GNA_MutuallyReplaces#0
WHERE GNA_MutuallyReplaces#0.physicianID =
                                GNA_MutuallyReplaces#1.physicianID
```

Θα παρουσιάσουμε τώρα μερικά παραδείγματα που δείχνουν την επιρροή της εισαγωγής σταθερών ή/και συγκρίσεων στη μετάφραση των ερωτημάτων. Θα χρησιμοποιήσουμε τη LAV αντιστοίχηση του πίνακα GNA_Urgent, που έως τώρα την είχαμε αφήσει κενή.

5.2.2.6 Παράδειγμα MiniCon.6

LAV αντιστοίχηση του GNA_Urgent (1):

```
select KAT_Clinic.clinicID as infirmaryID,
       KAT_Clinic.name as name,
       KAT_Clinic.floor as floor
from KAT_Clinic
where KAT_Clinic.buildingName = 'A';
```

(Τα ιατρεία των επειγόντων περιστατικών είναι εκείνα του κτιρίου Α)

Εμφάνιση των ιατρείων – κλινικών του κτιρίου Α, του 2^{ου} ορόφου.

Αρχικό ερώτημα:

```
select KAT_Clinic.name,
       KAT_Clinic.floor
from KAT_Clinic
where KAT_Clinic.buildingName = 'A' and
       KAT_Clinic.floor = 2;
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Department#0.name,
      GNA_Department#0.floor
FROM GNA_Department as GNA_Department#0
WHERE GNA_Department#0.building = 'A' AND
      GNA_Department#0.floor = 2

UNION

SELECT GNA_Urgent#0.name,
      GNA_Urgent#0.floor
FROM GNA_Urgent as GNA_Urgent#0
WHERE GNA_Urgent#0.floor = 2
```

5.2.2.7 Παράδειγμα MiniCon.7

Εμφάνιση των ιατρείων – κλινικών του κτιρίου Β.

Παρατηρούμε ότι το ερώτημα τώρα δεν μπορεί να μεταφραστεί από την αντιστοίχηση του πίνακα GNA_Urgent, λόγω διαφοράς στην τιμή της σταθεράς.

Αρχικό ερώτημα:

```
select KAT_Clinic.name,
      KAT_Clinic.floor
from KAT_Clinic
where KAT_Clinic.buildingName = 'B';
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Department#0.name,
      GNA_Department#0.floor
FROM GNA_Department as GNA_Department#0
WHERE GNA_Department#0.building = 'B' AND
      GNA_Department#0.floor = 2
```

5.2.2.8 Παράδειγμα MiniCon.8

LAV αντιστοίχηση του GNA_Urgent (2):

```
select KAT_Clinic.clinicID as infirmaryID,
      KAT_Clinic.name as name,
      KAT_Clinic.buildingName as building
from KAT_Clinic
where KAT_Clinic.floor <= 2;
```

(Τα ιατρεία των επειγόντων περιστατικών βρίσκονται στους δύο πρώτους ορόφους κάθε κτιρίου)

5.2.2.9 Παράδειγμα MiniCon.9

Εμφάνιση των ιατρείων – κλινικών των τεσσάρων πρώτων ορόφων κάθε κτιρίου.

Το ερώτημα μεταφράζεται και από τη LAV αντιστοίχηση του GNA_Urgent, διότι η συνθήκη KAT_Clinic.floor ≤ 2 είναι ισχυρότερη από την KAT_Clinic.floor ≤ 4 που περιέχεται στο ερώτημα. Συνεπώς, το ερώτημα δεν μεταφράζεται πλήρως αλλά ωστόσο μεταφράζεται σωστά με βάση όσα αναφέρθηκαν στην παράγραφο 2.2.3 για query containment.

Αρχικό ερώτημα:

```
select KAT_Clinic.name,
      KAT_Clinic.buildingName
from KAT_Clinic
where KAT_Clinic.floor <= 4;
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Department#0.name,
      GNA_Department#0.building
FROM GNA_Department as GNA_Department#0
WHERE GNA_Department#0.floor <= 4

UNION

SELECT GNA_Urgent#0.name
FROM GNA_Urgent as GNA_Urgent#0
```

5.2.2.10 Παράδειγμα MiniCon.10

Εμφάνιση των ιατρείων – κλινικών του 2ου έως 5ου ορόφου κάθε κτιρίου.

Το ερώτημα δεν μεταφράζεται από τη LAV αντιστοίχηση του GNA_Urgent, διότι η συνθήκη KAT_Clinic.floor ≥ 2 του ερωτήματος δεν μπορεί να μεταφραστεί.

Αρχικό ερώτημα:

```
select KAT_Clinic.name,
      KAT_Clinic.buildingName
from KAT_Clinic
where KAT_Clinic.floor >= 2 and
      KAT_Clinic.floor <= 5;
```

Μεταφρασμένο ερώτημα:

```
SELECT GNA_Department#0.name,
      GNA_Department#0.building
FROM GNA_Department as GNA_Department#0
WHERE GNA_Department#0.floor >= 2 AND
      GNA_Department#0.floor <= 5
```

5.2.3 Προεπεξεργασία

Σε κάθε παράδειγμα εκτέλεσης της προεπεξεργασίας ερωτημάτων, θα δείχνουμε το αρχικό ερώτημα και την έξοδο του προγράμματος που αποτελείται από τα εξής:

- Το ερώτημα ξαναγραμμένο μετά από την επεξεργασία του από την κλάση PreSQLToConj η οποία αλλάζει (ξανά) τα ονόματα των μετονομασμένων πινάκων.
- Το ερώτημα που μπορεί να μεταφραστεί από τις διαθέσιμες GAV αντιστοιχήσεις, καθώς και την τιμή της συνάρτησης Sim για αυτό.
- Το ερώτημα που μπορεί να μεταφραστεί από τις διαθέσιμες LAV αντιστοιχήσεις, και την τιμή της συνάρτησης Sim για αυτό.

5.2.3.1 Παράδειγμα Preprocessing.1

Εμφάνιση των ονοματεπωνύμων των διευθυντών και των ονομάτων των κλινικών που διευθύνουν.

Το ερώτημα αυτό μπορεί να μεταφραστεί πλήρως και από τους δύο τύπους αντιστοιχήσεων.

Αρχικό ερώτημα:

```
select KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
from KAT_Doctor, KAT_Clinic, KAT_Directs
where KAT_Directs.doctorID = KAT_Doctor.doctorID and
      KAT_Directs.clinicID = KAT_Clinic.clinicID;
```

Έξοδος:

```
Initial Query
--------------
SELECT KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
FROM KAT_Doctor, KAT_Clinic, KAT_Directs
WHERE KAT_Directs.doctorID = KAT_Doctor.doctorID AND
      KAT_Directs.clinicID = KAT_Clinic.clinicID

GAV - Sim = 1.0
----
SELECT KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
FROM KAT_Doctor, KAT_Clinic, KAT_Directs
WHERE KAT_Directs.doctorID = KAT_Doctor.doctorID AND
      KAT_Directs.clinicID = KAT_Clinic.clinicID

LAV - Sim = 1.0
----
SELECT KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
FROM KAT_Doctor, KAT_Clinic, KAT_Directs
WHERE KAT_Directs.doctorID = KAT_Doctor.doctorID AND
      KAT_Directs.clinicID = KAT_Clinic.clinicID
```

5.2.3.2 Παράδειγμα Preprocessing.2

Εμφάνιση του ονοματεπωνύμου κάθε νοσοκόμου και εκείνου ή εκείνων που τον αντικαθιστούν.

Το ερώτημα μεταφράζεται πλήρως τόσο χρησιμοποιώντας GAV όσο και χρησιμοποιώντας LAV αντιστοιχήσεις. Ωστόσο, η τιμή της συνάρτησης Sim στην πρώτη περίπτωση είναι μικρότερη λόγω της ύπαρξης επιπλέον σταθερών στα where κομμάτια των χρησιμοποιούμενων GAV αντιστοιχήσεων.

Αρχικό ερώτημα:

```
select N1.name, N1.surname, N2.name, N2.surname
from KAT_NurseReplaces, KAT_Nurse N1, KAT_Nurse N2
where KAT_NurseReplaces.nurseID = N1.nurseID and
      KAT_NurseReplaces.substituteID = N2.nurseID;
```

Έξοδος:

```
Initial Query
--------------
SELECT KAT_Nurse#1.name,
      KAT_Nurse#1.surname,
      KAT_Nurse#2.name,
      KAT_Nurse#2.surname
FROM KAT_NurseReplaces, KAT_Nurse KAT_Nurse#1, KAT_Nurse KAT_Nurse#2
WHERE KAT_NurseReplaces.nurseID = KAT_Nurse#1.nurseID AND
      KAT_NurseReplaces.substituteID = KAT_Nurse#2.nurseID

GAV - Sim = 0.9444444
----
SELECT KAT_Nurse#1.name,
      KAT_Nurse#1.surname,
      KAT_Nurse#2.name,
      KAT_Nurse#2.surname
FROM KAT_NurseReplaces, KAT_Nurse KAT_Nurse#1, KAT_Nurse KAT_Nurse#2
WHERE KAT_NurseReplaces.nurseID = KAT_Nurse#1.nurseID AND
      KAT_NurseReplaces.substituteID = KAT_Nurse#2.nurseID
```

```
LAV - Sim = 1.0
----
SELECT KAT_Nurse#1.name,
      KAT_Nurse#1.surname,
      KAT_Nurse#2.name,
      KAT_Nurse#2.surname
FROM KAT_NurseReplaces, KAT_Nurse KAT_Nurse#1, KAT_Nurse KAT_Nurse#2
WHERE KAT_NurseReplaces.nurseID = KAT_Nurse#1.nurseID AND
      KAT_NurseReplaces.substituteID = KAT_Nurse#2.nurseID
```

5.2.3.3 Παράδειγμα Preprocessing.3

Εμφάνιση των ονοματεπωνύμων όλων των λογιστών.

Το ερώτημα δεν μπορεί να μεταφραστεί από τις διαθέσιμες LAV αντιστοιχήσεις, οπότε η Sim γι' αυτές έχει τιμή 0. Επίσης η τιμή της Sim για τις GAV αντιστοιχήσεις δεν είναι 1 λόγω ύπαρξης επιπλέον σταθερών στη GAV αντιστοίχηση του πίνακα KAT_Clerk.

Αρχικό ερώτημα:

```
SELECT KAT_Clerk.name, KAT_Clerk.surname
FROM KAT_Clerk
WHERE KAT_Clerk.speciality = 'Bookkeeper';
```

Έξοδος:

```
Initial Query
--------------
SELECT KAT_Clerk.name, KAT_Clerk.surname
FROM KAT_Clerk
WHERE KAT_Clerk.speciality = 'Bookkeeper'

GAV - Sim = 0.9
----
SELECT KAT_Clerk.name, KAT_Clerk.surname
FROM KAT_Clerk
WHERE KAT_Clerk.speciality = 'Bookkeeper'
```

```
LAV - Sim = 0.0
----
The Query cannot be translated
```

Εμφάνιση του ονοματεπωνύμου κάθε ιατρού και του ονόματος της κλινικής όπου δουλεύει.

Το ερώτημα αυτό μπορεί να μεταφραστεί πλήρως από τις διαθέσιμες LAV αντιστοιχήσεις, αλλά ελλιπώς από τις GAV. Αυτό συμβαίνει επειδή ο πίνακας KAT_Works δεν έχει GAV αντιστοίχηση.

Αρχικό ερώτημα:

```
select KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
from KAT_Doctor, KAT_Clinic, KAT_Works
where KAT_Doctor.doctorID = KAT_Works.doctorID and
      KAT_Works.clinicID = KAT_Clinic.clinicID;
```

Έξοδος:

```
Initial Query
--------------
SELECT KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
FROM KAT_Doctor, KAT_Clinic, KAT_Works
WHERE KAT_Doctor.doctorID = KAT_Works.doctorID AND
      KAT_Works.clinicID = KAT_Clinic.clinicID

GAV - Sim = 0.6875
----
SELECT KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
FROM KAT_Doctor, KAT_Clinic, KAT_Works

LAV - Sim = 1.0
----
SELECT KAT_Doctor.name, KAT_Doctor.surname, KAT_Clinic.name
FROM KAT_Doctor, KAT_Clinic, KAT_Works
WHERE KAT_Doctor.doctorID = KAT_Works.doctorID AND
      KAT_Works.clinicID = KAT_Clinic.clinicID
```

5.2.3.4 Παράδειγμα Preprocessing.4

Εμφάνιση του ονοματεπωνύμου κάθε ιατρού και το όνομα της κλινικής όπου δουλεύει.

Το ερώτημα αυτό μεταφράζεται πλήρως με GAV αντιστοιχήσεις και ελλιπώς με LAV αντιστοιχήσεις. Η τιμή της Sim για τις GAV αντιστοιχήσεις δεν είναι 1 λόγω ύπαρξης επιπλέον σταθερών στη GAV αντιστοίχηση του πίνακα KAT_Nurse. Επίσης, στην περίπτωση των LAV αντιστοιχήσεων, ο σύνδεσμος που περιέχει το ερώτημα αποκόπτεται διότι για τη μετάφραση του subgoal KAT_Nurse χρησιμοποιείται μία μόνο LAV αντιστοίχηση, εκείνη του πίνακα GNA_Employee, η οποία δεν μπορεί να μεταφράσει την ιδιότητα KAT_Nurse.clinicID .

Αρχικό ερώτημα:

```
select KAT_Nurse.name, KAT_Nurse.surname, KAT_Clinic.name
from KAT_Nurse, KAT_Clinic
where KAT_Nurse.clinicID = KAT_Clinic.clinicID;
```

Έξοδος:

```
Initial Query
--------------
SELECT KAT_Nurse.name, KAT_Nurse.surname, KAT_Clinic.name
FROM KAT_Nurse, KAT_Clinic
WHERE KAT_Nurse.clinicID = KAT_Clinic.clinicID

GAV - Sim = 0.95
----
SELECT KAT_Nurse.name, KAT_Nurse.surname, KAT_Clinic.name
FROM KAT_Nurse, KAT_Clinic
WHERE KAT_Nurse.clinicID = KAT_Clinic.clinicID

LAV - Sim = 0.85
----
SELECT KAT_Nurse.name, KAT_Nurse.surname, KAT_Clinic.name
FROM KAT_Nurse, KAT_Clinic
```

5.2.3.5 Παράδειγμα Preprocessing.5

Εμφάνιση των 4^{ων} πρώτων πεδίων του πίνακα KAT_Patient.

Το ερώτημα αυτό χρησιμοποιείται στην παράγραφο 6.2.2 για να δείξουμε πώς μπορεί να βελτιωθεί ο αλγόριθμος MiniCon που υλοποιήθηκε για τη μετάφραση ερωτημάτων με βάση LAV αντιστοιχήσεις. Στο ερώτημα, το πεδίο KAT_Patient.doctorID μπορεί να μεταφραστεί με χρήση της LAV αντιστοίχησης του πίνακα GNA_Treats, όμως δεν μεταφράζεται διότι ο αλγόριθμος επιλέγει μόνο μία LAV αντιστοίχηση για να μεταφράσει κάθε subgoal του ερωτήματος. Για τον subgoal KAT_Patient επιλέγεται η αντιστοίχηση του πίνακα GNA_Hospitalized, οπότε το πεδίο KAT_Patient.doctorID που δε μπορεί να μεταφραστεί από αυτή αποκόπτεται.

Αρχικό ερώτημα:

```
select KAT_Patient.patientID, KAT_Patient.doctorID, KAT_Patient.name,
KAT_Patient.surname
from KAT_Patient;
```

Έξοδος:

```
Initial Query
--------------
SELECT KAT_Patient.patientID,
      KAT_Patient.doctorID,
      KAT_Patient.name,
      KAT_Patient.surname
FROM KAT_Patient

GAV - Sim = 1.0
----
SELECT KAT_Patient.patientID,
      KAT_Patient.doctorID,
      KAT_Patient.name,
      KAT_Patient.surname
FROM KAT_Patient
```

```
LAV - Sim = 0.7222222
----
SELECT KAT_Patient.patientID,
      KAT_Patient.name,
      KAT_Patient.surname
FROM KAT_Patient
```

6

Επίλογος

6.1 Σύνοψη και συμπεράσματα

Στα πλαίσια της εργασίας σχεδιάστηκε και υλοποιήθηκε ένας μηχανισμός μετάφρασης ερωτημάτων για P2P Βάσεις Δεδομένων. Ο μηχανισμός περιελάμβανε ένα στάδιο προεπεξεργασίας ερωτημάτων, απαραίτητο για τη μετάφραση ερωτημάτων σε P2P περιβάλλοντα, και ένα στάδιο μετάφρασης με χρήση δύο τύπων αντιστοιχήσεων (mappings), GAV και LAV (βλ. παρ. 2.2.2).

Για το κομμάτι της προεπεξεργασίας ερωτημάτων σχεδιάστηκε μία διαδικασία μείωσης ερωτημάτων, η οποία αποκόπτει από τα αρχικά ερωτήματα τα τμήματα που δεν μπορούν να μεταφραστούν με βάση τις διαθέσιμες GAV / LAV αντιστοιχήσεις. Επίσης μελετήθηκε η επίδραση που έχει η αποκοπή συγκεκριμένων τμημάτων από ένα ερώτημα. Με βάση τη μελέτη αυτή υλοποιήθηκε μία συνάρτηση που εκτιμά την ποιότητα των μειωμένων ερωτημάτων, δηλαδή την ομοιότητά τους με τα αντίστοιχα αρχικά ερωτήματα. Ανάλογα με την τιμή της συνάρτησης αυτής για GAV / LAV αντιστοιχήσεις, επιλέγεται και ο αλγόριθμος μετάφρασης που θα χρησιμοποιηθεί στη συνέχεια για να μεταφράσει το αντίστοιχο μειωμένο ερώτημα.

Για τη μετάφραση των ερωτημάτων σχεδιάστηκαν και υλοποιήθηκαν δύο αλγόριθμοι. Ο πρώτος εργάζεται με GAV αντιστοιχήσεις και είναι σχετικά απλός, ενώ ο δεύτερος, που λέγεται MiniCon, εργάζεται με LAV αντιστοιχήσεις και είναι αρκετά πολύπλοκος. Οι δύο

αλγόριθμοι, αρχικά σχεδιασμένοι για τη γλώσσα Datalog, αναλύθηκαν διεξοδικά και επανασχεδιάστηκαν για να λειτουργούν με ερωτήματα και αντιστοιχήσεις σε SQL, με σκοπό να μπορούν να χρησιμοποιηθούν σε μία πραγματική P2P Βάση Δεδομένων.

Η σχεδίαση και υλοποίηση του MiniCon αποτέλεσε το μεγαλύτερο μέρος της εργασίας. Στη διαδικασία αυτή σημαντικό ρόλο έπαιξε μία νέα αναπαράσταση όψεων, η SQL_CNF (παρ. 3.1), στην οποία βασίζεται ολόκληρος ο αλγόριθμος αυτός, αλλά και τα δυσκολότερα τμήματα της προεπεξεργασίας.

Ολόκληρος ο μηχανισμός ελέγχθηκε από μία σειρά παραδειγμάτων εκτέλεσης που καλύπτουν μεγάλο εύρος περιπτώσεων και βαθμών πολυπλοκότητας. Τα αποτελέσματα ήταν πολύ ικανοποιητικά, γεγονός που οδηγεί στο συμπέρασμα ότι θα μπορέσει να λειτουργήσει ικανοποιητικά στο πλαίσιο μίας πραγματικής P2P Βάσης Δεδομένων.

6.2 Μελλοντικές επεκτάσεις

Στην ενότητα αυτή θα παρουσιάσουμε ορισμένες ιδέες που μπορούν να υλοποιηθούν για την επέκταση και τη βελτίωση όσων κατασκευάστηκαν στην παρούσα εργασία.

6.2.1 Εύρεση μειωμένου ερωτήματος για μετάφραση με LAV αντιστοιχήσεις

Στην παράγραφο 3.4.2 εκθέσαμε το πρόβλημα που παρουσιάζεται κατά την προσπάθεια εύρεσης του καλύτερου μειωμένου ερωτήματος που να μπορεί να απαντηθεί με βάση ορισμένες αντιστοιχήσεις τύπου LAV. Στην παρούσα εργασία το πρόβλημα αυτό λύθηκε με απλουστευμένο τρόπο που περιγράφεται στην ίδια παράγραφο. Όμως, μπορεί να αναζητηθεί μία καλύτερη λύση του προβλήματος, η οποία κατά πάσα πιθανότητα θα συνδέεται στενά με τον αλγόριθμο MiniCon. Συκεκριμένα, μία διαδικασία η οποία θα ελέγχει σταδιακά τις ιδιότητες που πρέπει να ικανοποιούν τα MCDs του αλγορίθμου (βλ. παρ. 3.3) και θα οπισθοδρομεί κάθε φορά που μία ιδιότητα δεν μπορεί να ικανοποιηθεί, αποκόπτοντας παράλληλα το κομμάτι του ερωτήματος που δημιουργεί το πρόβλημα, μπορεί τελικά να ανακαλύψει ένα ποιοτικό μειωμένο ερώτημα. Ο σχεδιασμός μίας τέτοιας διαδικασίας δεν είναι απλός, ιδιαίτερα εάν τεθεί και ο περιορισμός της εύρεσης του καλύτερου δυνατού μειωμένου ερωτήματος με βάση κάποια κριτήρια όπως αυτά που παρουσιάστηκαν στην παράγραφο 2.3.3. Μπορεί όμως μετά από κατάλληλη μελέτη να βρεθεί μία ικανοποιητική λύση.

6.2.2 Βελτίωση του αλγορίθμου MiniCon

Ας ανατρέξουμε στο παράδειγμα Preprocessing.5, στην παράγραφο 5.2.3.5. Στο παράδειγμα αυτό παρουσιάζεται το παρακάτω γενικό πρόβλημα .

Έστω ότι σε ένα P2P δίκτυο έχουμε δύο κόμβους P1 και P2 και έστω ότι ο P2 έχει έναν πίνακα T (a, b, c) , στον οποίο η ιδιότητα a είναι το κλειδί. Έστω επίσης ότι ο P1 έχει δύο LAV αντιστοιχήσεις προς τον P2, που είναι οι εξής (σε Datalog) :

L1 (a, b) :- T (a, b) και L2 (a, c) :- T (a, c).

Με βάση μόνο αυτές τις δύο αντιστοιχήσεις, ο αλγόριθμος MiniCon δεν μπορεί να μεταφράσει κανένα ερώτημα που περιέχει όλες τις ιδιότητες του πίνακα T. Διότι ο subgoal που αντιστοιχεί στον T θα πρέπει να καλύπτεται από ένα μόνο MCD, δηλαδή από μία μόνο αντιστοίχηση. Όμως, καμία από τις L1, L2 δεν μπορεί να καλύψει όλες τις ιδιότητες του T. (Στο παράδειγμα Preprocessing.5, το ρόλο του πίνακα T έχει ο πίνακας KAT_Patient, ενώ το ρόλο των αντιστοιχήσεων L1, L2 έχουν οι LAV αντιστοιχήσεις των πινάκων GNA_Treats και GNA_Hospitalized).

Ας πάρουμε το παρακάτω ερώτημα που περιέχει όλες τις ιδιότητες του T:

select T.a, T.b, T.c

from T;

Όπως είπαμε παραπάνω, ο MiniCon δεν μπορεί να μεταφράσει το ερώτημα αυτό με βάση μόνο τις L1 και L2. Ωστόσο, με βάση το ότι η ιδιότητα T.a είναι κλειδί του T, το ερώτημα θα μπορούσε να μεταφραστεί ως εξής:

select L1.a, L1.b, L2.c

from L1, L2

where L1.a = L2.a;

Η βελτίωση που προτείνεται για τον MiniCon, είναι το να λαμβάνει σαν είσοδο εκτός από τις LAV αντιστοιχήσεις προς έναν κόμβο, και τα κλειδιά του κόμβου αυτού, και να τροποποιηθεί ώστε να μπορεί να μεταφράζει και τις παραπάνω περιπτώσεις ερωτημάτων.

Οι αλλαγές που θα πρέπει να γίνουν στον MiniCon θα είναι σημαντικές, διότι τα MCDs θα πρέπει να επεκταθούν σε "multiMCDs", καθένα από τα οποία θα ανήκει όχι πλέον σε μία αλλά πιθανόν σε παραπάνω αντιστοιχήσεις (που έχουν συνδεθεί μεταξύ τους με βάση ένα κλειδί). Συνεπώς, τόσο η διαδικασία formMCDs όσο και η combineMCDs θα πρέπει να τροποποιηθούν κατάλληλα για να χειρίζονται αυτόν τον νέο τύπο MCDs.

Ωστόσο, το πλεονέκτημα που αποκτά ο αλγόριθμος μετά τις τροποποιήσεις αυτές είναι πολύ σημαντικό για τον εξής λόγο. Αν υποθέσουμε ότι σε ένα P2P δίκτυο έχουμε δύο κόμβους, P1 και P2, που οι Βάσεις Δεδομένων τους είναι πολύ κοντά σημασιολογικά, όπως οι ΒΔ των

νοσοκομείων που χρησιμοποιήσαμε στο κεφάλαιο 5. Τότε κάθε πίνακας T του P1 (και αντίστοιχα του P2), εμπίπτει με μεγάλη πιθανότητα σε μία από τις παρακάτω τρεις κατηγορίες. Ή ο T «σπάει» σε δύο ή παραπάνω πίνακες στον P2, ή υπάρχει όμοιος του T στον P2, ή ο T είναι ένα «κομμμάτι» ενός πίνακα T' του P2, που μαζί με άλλους πίνακες του P1 δημιουργούν τον T'. Αν κατασκευάσουμε LAV αντιστοιχήσεις για τους P1 και P2, θα δούμε ότι ο MiniCon αποτυγχάνει να μεταφράσει ερωτήματα πολλά ερωτήματα με πίνακες που ανήκουν στην πρώτη κατηγορία, και ο λόγος είναι ακριβώς αυτός που περιγράψαμε στην αρχή της παραγράφου. Όμως οι πίνακες της κατηγορίας αυτής είναι ένα πολύ σημαντικό ποσοστό επί των συνολικών. Αν υποθέσουμε για παράδειγμα ότι οι κόμβοι P1 και P2 έχουν κατά 10% επί των συνολικών όμοιους ακριβώς πίνακες (κατηγορία 2), και κατά 20% πίνακες που δεν ανήκουν σε καμία από τις τρεις κατηγορίες, τότε κατά μέσο όρο έχουν 35% πίνακες που ανήκουν στην πρώτη κατηγορία.

Συνεπώς, η βελτίωση που προτείνεται θα αυξήσει κατά ένα πολύ σημαντικό ποσοστό το εύρος των ερωτημάτων που ο MiniCon μπορεί να μεταφράσει.

6.2.3 Αλγόριθμος μετάφρασης με συνδυασμό GAV και LAV αντιστοιχήσεων

Ένα προκλητικό πρόβλημα στον τομέα της μετάφρασης ερωτημάτων με χρήση αντιστοιχήσεων είναι η σχεδίαση ενός αλγορίθμου που να μπορεί να εκμεταλλεύεται συγχρόνως τις διαθέσιμες GAV και LAV αντιστοιχήσεις ενός κόμβου για τη μετάφραση των ερωτημάτων του.

6.2.4 Ολοκλήρωση του GrouPeer

Το σύστημα GrouPeer περιγράφηκε αναλυτικά στην παράγραφο 2.3. Όπως φαίνεται και στο Σχήμα 2.1 της παραγράφου αυτής, ένας κόμβος στο GrouPeer αποτελείται από τα εξής στοιχεία:

- User Interface
- PDBMS
- Query Manager
- Query Rewriting Algorithms
- Query Similarity Function
- Automatic Schema Matching Tool
- Answers' Evaluator

- Acquaintance Manager
- Results' Manager

Στην παρούσα εργασία υλοποιήσαμε τους αλγορίθμους μετάφρασης ερωτημάτων και τη συνάρτηση μέτρησης ομοιότητας (ποιότητας) ερωτημάτων. Απομένει λοιπόν σαν μελλοντική εργασία η υλοποίηση των υπολοίπων τμημάτων του κάθε κόμβου, και στη συνέχεια η ενσωμάτωση ενός αριθμού κόμβων σε ένα P2P δίκτυο και η προσομοίωση της λειτουργίας του μηχανισμού του GrouPeer.

7

Βιβλιογραφία

[Hal01] A. Y. Halevy: “Answering queries using views: A survey”, In: The VLDB Journal: 270 – 294 (2001)

[KST+05] V. Kantere, T. Sellis, D. Tsoumakos, N. Roussopoulos: “GrouPeer: Dynamic Clustering of P2P Databases”. NTUA (National Technical University of Athens) & University of Maryland

[KS] V. Kantere, T. Sellis: “Reusing classical query rewriting in P2P Databases”. NTUA

[HIS+] A. Y. Halevy, Z. G. Ives, D. Suciu, I. Tatarinov: “Schema Mediation in Peer Data Management Systems”. University of Washington

[TIM+] I. Tatarinov, Z. Ives, J. Madhavan: “The Piazza Peer Data Management Project”. University of Washington, University of Pennsylvania

[HIM+03] A. Y. Halevy, Z. Ives, J. Madhavan: “The Piazza Peer Data Management System”. University of Washington, University of Pennsylvania

[PH01] R. Pottinger, A. Y. Halevy: “MiniCon: A scalable algorithm for answering queries using views”, In: The VLDB Journal (2001) 10: 182 – 198

[Len02] M. Lenzerini: “Data Integration: A Theoretical Perspective”, In: ACM PODS 2002

[Σπυ05] Ε. Σπυροπούλου: "Πρότυπο Σύστημα Ομότιμων Κόμβων Βασισμένο σε Σχήματα RDF". Διπλ. εργασία Ειρήνης Κ. Σπυροπούλου, ΕΜΠ 2005