ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΖΩΓΡΑΦΟΥ 157 73, ΑΘΗΝΑ

ΕΒΓΔ-ΔΙΠΛ-2002-06 30 Ιανουαρίου 2003

ΥΛΟΠΟΙΗΣΗ ΓΛΩΣΣΑΣ ΕΠΕΡΩΤΗΣΕΩΝ ΣΕ ΠΟΛΥΔΙΑΣΤΑΤΑ ΗΜΙΔΟΜΗΜΕΝΑ ΔΕΔΟΜΕΝΑ (MQL)

(ΤΟΜΟΣ Ι)

ΠΡΙΣΤΟΥΡΗΣ ΚΩΣΤΗΣ
ΥΦΑΝΤΗΣ ΑΝΤΩΝΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Τίμος Σελλής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Συστημάτων Βάσεων και Γνώσεων Δεδομένων (ΕΒΓΔ) του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Ευχαριστούμε θερμά τον καθηγητή κ. Τ. Σελλή και τον υποψήφιο διδάκτορα και επιβλέποντα της διπλωματικής κ. Γιάννη Σταύρακα για τις πολύτιμες και ζωτικής σημασίας συμβουλές και υποδείξεις τους κατά την εκπόνηση της διπλωματικής αυτής εργασίας, καθώς και όλα τα μέλη του εργαστηρίου για την προθυμία τους για βοήθεια και τεχνική υποστήριξη.

Ιανουάριος 2003

ΠΡΙΣΤΟΥΡΗΣ ΚΩΣΤΗΣ
ΥΦΑΝΤΗΣ ΑΝΤΩΝΗΣ

Περίληψη

Αντικείμενο της διπλωματικής αυτής είναι η υλοποίηση ενός συστήματος επερωτήσεων πάνω σε πολυδιάστατα ημιδομημένα δεδομένα μοντελοποιημένα σύμφωνα με το πρότυπο MOEM (Multidimensional Object Exchange Model). Η γλώσσα που υποστηρίζεται από το σύστημα είναι η MQL, η οποία έχει σχεδιαστεί ειδικά για το MOEM μοντέλο. Τα πολυδιάστατα ημιδομημένα δεδομένα αποτελούν μία επέκταση στα ημιδομημένα δεδομένα στην οποία εισάγεται η έννοια των πολλαπλών όψεων (στο περιεχόμενο ή και στη δομή) που μπορεί να λάβει μια πληροφοριακή οντότητα ανάλογα με τις συνθήκες (συμφραζόμενα, context) κάτω από τις οποίες αυτή έχει υπόσταση. Η παρούσα υλοποίηση βασίστηκε στο σύστημα διαχείρισης Lore, το οποίο παρέχει μεταξύ άλλων τη γλώσσα Lorel για επερωτήσεις σε ημιδομημένα δεδομένα που αναπαρίστανται με OEM γράφους. Για αυτό το σκοπό σχεδιάστηκε ένα μοντέλο μεταφοράς MOEM γράφων σε OEM και αντίστοιχης μετάφρασης MQL ερωτήσεων σε Lorel. Ως βάση για την εφαρμογή που αναπτύχθηκε για την επίδειξη του συστήματος αυτού χρησιμοποιήθηκε το πρόγραμμα επεξεργασίας MOEM γράφων MSSDesigner.

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ: πολυδιάστατα, ημιδομημένα δεδομένα, MOEM, MQL, context, γλώσσα επερωτήσεων

Summary

The object of this Diploma Thesis is the implementation of a query system for multidimensional semistructured data according to the Multidimensional Object Exchange Model (MOEM). The query language supported by the system is MQL, which is specifically designed for the MOEM model. Multidimensional semistructured data is an extension of semistructured data, which introduces the concept of multiple facets (of content and/or structure) of an information entity that hold under different contexts. The current implementation was based on the Lore management system, which provides the Lorel query language for OEM modeled semistructured data. To this end a schema was designed, which transforms MOEM graphs into OEM graphs and translates MQL queries into Lorel. The application developed for the demonstration of this system was based on the MOEM graph editing program, MSSDesigner.

KEYWORDS: multidimensional, semistructured data, MOEM, MQL, context, query language

Πίνακας Περιεχομένων

1

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Με την ανάπτυξη του διαδικτύου και την ολοένα και πιο συχνή χρήση του, ο όγκος των εγγράφων και γενικότερα των δεδομένων που βρίσκονται και διακινούνται σε αυτό αυξάνεται ολοένα και περισσότερο. Πέρα από αυτό οι επιχειρηματικές δραστηριότητες και οι υπηρεσίες που προσφέρονται μέσω του διαδικτύου απαιτούν αξιοπιστία και προτυποποίηση του τρόπου διακίνησης της πληροφορίας, καθώς και της μορφής της.

Προκειμένου να ικανοποιηθούν αυτές οι νέες απαιτήσεις, επιβάλλεται η λύση ορισμένων προβλημάτων που προκύπτουν, όπως το γεγονός ότι η μορφή με την οποία υπάρχουν τα δεδομένα δεν είναι μία αλλά πολλές, αλλά και το γεγονός ότι τα δεδομένα βρίσκονται διασπαρμένα σε διαφορετικές τοποθεσίες στο διαδίκτυο. Με άλλα λόγια καλούμαστε να διαχειριστούμε διαφορετικές μορφές αποθήκευσης και οργάνωσης δεδομένων που όμως βρίσκονται σε ένα κοινό δίκτυο υπολογιστών.

Προς αυτό το στόχο κρίνεται αναγκαία η εύρεση ενός τρόπου ενοποίησης αυτών των ετερογενών πηγών πληροφορίας μέσω ενός μοντέλου διαχείρισης που μπορεί να είναι αρκετά ευέλικτο, ώστε να μπορεί να χειρίζεται δεδομένα διαφορετικής φύσεως, μορφής και οργάνωσης με το ίδιο ομοιογενή τρόπο, που θα μπορεί να προσδίδει δομή και οργάνωση σε δεδομένα ετερογενούς και μη αυστηρά καθορισμένης φύσης.

Τα τελευταία χρόνια έχει αναπτυχθεί έντονη δραστηριότητα σε ένα νέο τομέα των βάσεων δεδομένων που ασχολείται με τα ημιδομημένα δεδομένα, τα δεδομένα αυτά δηλαδή που δεν ακολουθούν κάποιο σαφώς καθορισμένο σχήμα, είναι ανομοιογενή και αυτοπεριγραφόμενα. Κατά καιρούς έχουν προταθεί διάφορα μοντέλα διαχείρισης αυτού του είδους δεδομένων, το καθένα από τα οποία αντιμετωπίζει το ζήτημα από ποικίλες σκοπιές.

Η παρούσα διπλωματική εργασία ασχολείται με την υλοποίηση της γλώσσας επερωτήσεων ενός τέτοιου μοντέλου περιγραφής ημιδομημένων δεδομένων, το οποίο ονομάζεται Multidimensional Object Exchange Model (MOEM). Το συγκεκριμένο μοντέλο αποτελεί επέκταση του αρκετά διαδεδομένου μοντέλου OEM. Η κατεύθυνση που καλούμαστε να ακολουθήσουμε είναι αυτή της υλοποίησης της γλώσσας MQL για αυτό το μοντέλο, αξιοποιώντας τις δυνατότητες επερωτήσεων σε ημιδομημένα δεδομένα OEM που παρέχονται από την ειδικά σχεδιασμένη για αυτό γλώσσα επερωτήσεων Lorel.

1.2 Οργάνωση του τόμου

Στο κεφάλαιο που ακολουθεί περιγράφουμε αναλυτικά τις έννοιες με τις οποίες θα ασχοληθούμε σε αυτή την εργασία. Συγκεκριμένα, περιγράφεται αρχικά το μοντέλο OEM για ημιδομημένα δεδομένα, καθώς και η γλώσσα επερωτήσεων Lorel. Στη συνέχεια, παρουσιάζεται συνοπτικά η επέκταση του μοντέλου αυτού στο MOEM, όπου εισάγεται η έννοια των πολυδιάστατων ημιδομημένων δεδομένων, ενώ δίνεται και μια περιγραφή της γλώσσας MQL, η οποία αξιοποιεί τις δυνατότητες του μοντέλου αυτού. Τέλος, προσδιορίζονται οι στόχοι τους οποίους καλείται να πετύχει η εργασία.

Στο 3^{ο} κεφάλαιο προχωρούμε στα βασικά σημεία της διπλωματικής εργασίας. Παρουσιάζεται ο σχεδιασμός του συστήματος με το οποίο θα υλοποιήσουμε την γλώσσα MQL με τη βοήθεια OEM γράφων και επερωτήσεων Lorel πάνω σε αυτούς. Ιδιαίτερη ανάλυση γίνεται όσον αφορά στη μεταφορά της πληροφορίας MOEM γράφων σε OEM, καθώς και στον αλγόριθμο μετάφρασης ερωτήσεων που καλύπτουν ένα βασικό υποσύνολο της MQL σε Lorel.

Το 4^{ο} κεφάλαιο αφορά στην υλοποίηση στην οποία καταλήξαμε για ένα σύστημα επερωτήσεων με βάση το σύστημα Lore. Αναφερόμαστε στις βασικές δομές δεδομένων που χρησιμοποιήσαμε και στην υποστήριξη των λειτουργιών που απαιτούνται από το σύστημα. Αναλύονται οι αλλαγές που εισάγαμε στην εφαρμογή MSSDesigner προκειμένου να την αξιοποιήσουμε ως το περιβάλλον εργασίας του συστήματος, ενώ παράλληλα σημειώνονται τα προβλήματα που παρουσιάστηκαν κατά την χρησιμοποίηση του συστήματος Lore ως βάση για το σύστημα επερωτήσεων.

Στο 5ο κεφάλαιο του τόμου παρουσιάζεται το σύστημα όπως διαμορφώθηκε τελικά, προκειμένου να δοθεί μια συνοπτική περιγραφή του τρόπου λειτουργίας του. Έτσι, περιγράφουμε τα βασικά εργαλεία με τη βοήθεια των οποίων μπορεί κανείς να υποβάλλει ερωτήσεις MQL σε έναν γράφο MOEM. Επιπλέον, παρουσιάζουμε τα αποτελέσματα δοκιμών που έγιναν στην εφαρμογή προκειμένου να ελεγχθεί η ορθή λειτουργία της όταν αυτή καλείται να χειριστεί γράφους και ερωτήσεις σε αυτούς, οι οποίοι περιέχουν ιδιαιτερότητες που τους καθιστούν ενδιαφέροντα παραδείγματα για δοκιμή.

Τέλος, στο 6ο κεφάλαιο συνοψίζουμε τα αποτελέσματα της εργασίας και αναφερόμαστε σε τρόπους με τους οποίους θα μπορούσε κανείς να αξιοποιήσει αυτά για περαιτέρω ανάπτυξη της υλοποίησης της MQL, αλλά και γενικότερα ενός συστήματος διαχείρισης πολυδιάστατων ημιδομημένων δεδομένων. Στο τελευταίο τμήμα της εργασίας περιλαμβάνεται πίνακας με τη βιβλιογραφία που μας βοήθησε στην ανάπτυξη και τη σύνταξη της διπλωματικής.

2

Περιγραφή θέματος

2.1 Ημιδομημένα δεδομένα

2.1.1 Το μοντέλο OEM

Το Object Exchange Model (OEM) αποτελεί την πρώτη προσπάθεια που έγινε στο χώρο των βάσεων δεδομένων για τον ορισμό ενός μοντέλου ενοποίησης ετερογενών και χαλαρά δομημένων μορφών δεδομένων που υπάρχουν κυρίως στο internet και γενικά σε δίκτυα υπολογιστών. Το συγκεκριμένο μοντέλο υπήρξε από τότε το πρότυπο για την γραφική αναπαράσταση της δομής και των τύπων των ημιδομημένων δεδομένων γενικά.

Το μοντέλο OEM στηρίζεται στη χρήση αντικειμένων και ακμών. Τα αντικείμενα είναι μια τριάδα πληροφορίας: ένας μοναδικός αριθμός (oid), ένας τύπος (type) και μια τιμή (value). Ο μοναδικός αριθμός είναι η ταυτότητα του αντικειμένου, ο τύπος του είναι είτε σύνθετος (complex), είτε ατομικός (atomic). Όταν ο τύπος του είναι complex σημαίνει ότι έχει ως τιμή ένα σύνολο από άλλα αντικείμενα (oid), που σημαίνει ότι υπάρχουν ακμές με κατεύθυνση προς τα άλλα αντικείμενα και που τον συνδέουν με αυτά. Όταν ο τύπος του είναι ατομικός αυτό σημαίνει πως η τιμή του είναι κάποιο δεδομένο κάποιου τύπου (ακέραιος (integer), χαρακτήρας (char), σύνολο χαρακτήρων (string), μια jpeg εικόνα κτλ.).

Οι ακμές συνδέουν κάθε complex αντικείμενο με άλλα αντικείμενα και έχουν και μια ετικέτα (label) που είναι ένα σύνολο χαρακτήρων (string) που επεξηγεί το νόημα του κόμβου στον οποίο καταλήγουν. Ουσιαστικά δηλαδή το συγκεκριμένο μοντέλο είναι ένας γράφος που έχει μια μοναδική ρίζα από την οποία ξεκινούν διάφορα αντικείμενα που συνδέονται μεταξύ τους χωρίς κάποιους περιορισμούς. Τα φύλλα αυτού το γράφου είναι ατομικά αντικείμενα. Έχουμε να κάνουμε δηλαδή με κατευθυνόμενους γράφους με ακμές που έχουν κάποια εννοιολογική σημασία.

Αυτό το μοντέλο γραφικής αναπαράστασης είναι ισοδύναμο με το συντακτικό που έχει οριστεί για τον προσδιορισμό των ημιδομημένων δεδομένων που αποτελείται από εκφράσεις SSD:

```
<ssd-expressions> ::=   <value> | oid<value> | oid
<value>           ::=   atomic_value| <complexvalue>
<complexvalue>    ::=   {label : <ssd-expr>,…,label : <ssd-expr>}
```

Η πιο διαδεδομένη μορφή ημιδομημένων δεδομένων που συναντάται σήμερα και που μπορούμε να πούμε πως η χρήση τους διαδίδεται και εξαπλώνεται ολοένα και περισσότερο είναι αυτά που υπάρχουν με την μορφή XML αρχείων.

2.1.2 Γλώσσες επερωτήσεων για ημιδομημένα δεδομένα

Για να είναι κάποιο μοντέλο περιγραφής και οργάνωσης δεδομένων χρήσιμο σε επίπεδο υλοποίησης και πλήρες στις δυνατότητες που δίνει σε αυτούς που το χρησιμοποιούν θα πρέπει να υποστηρίζεται από κάποιο εύκολο τρόπο διερεύνησης και παρουσίασης της πληροφορίας που περιλαμβάνει, καθώς και από κάποιο τρόπο αλλαγής της δομής του και ανανέωσης της πληροφορίας που περιέχει. Από την ήδη ανεπτυγμένη θεωρία στο χώρο των βάσεων δεδομένων και του σχεσιακού μοντέλου, είναι γνωστό πως τέτοιες δυνατότητες παρέχονται από κάποια γλώσσα επερωτήσεων που ορίζεται σε σχέση με το μοντέλο οργάνωσης και περιγραφής των δεδομένων.

Καταυτόν τον τρόπο και μια και στην ουσία το OEM είναι ένα σχήμα που θέλει να ορίσει μια νέα μορφή βάσεων δεδομένων, έχει οριστεί για αυτό και μια γλώσσα επερωτήσεων ημιδομημένων δεδομένων που ονομάζεται Lorel[ABS00]. Πέρα από τη συγκεκριμένη γλώσσα, έχουν γίνει και άλλες προσπάθειες ορισμού τέτοιων γλωσσών για ημιδομημένα δεδομένα. Κατά την μελέτη του ορισμού τους, ορίστηκαν κάποιες απαιτήσεις που ισχύουν γενικά για όλες τις γλώσσες επερωτήσεων που έχουν ορισθεί ή πρόκειται να ορισθούν για ημιδομημένα δεδομένα. Αυτές είναι οι εξής:

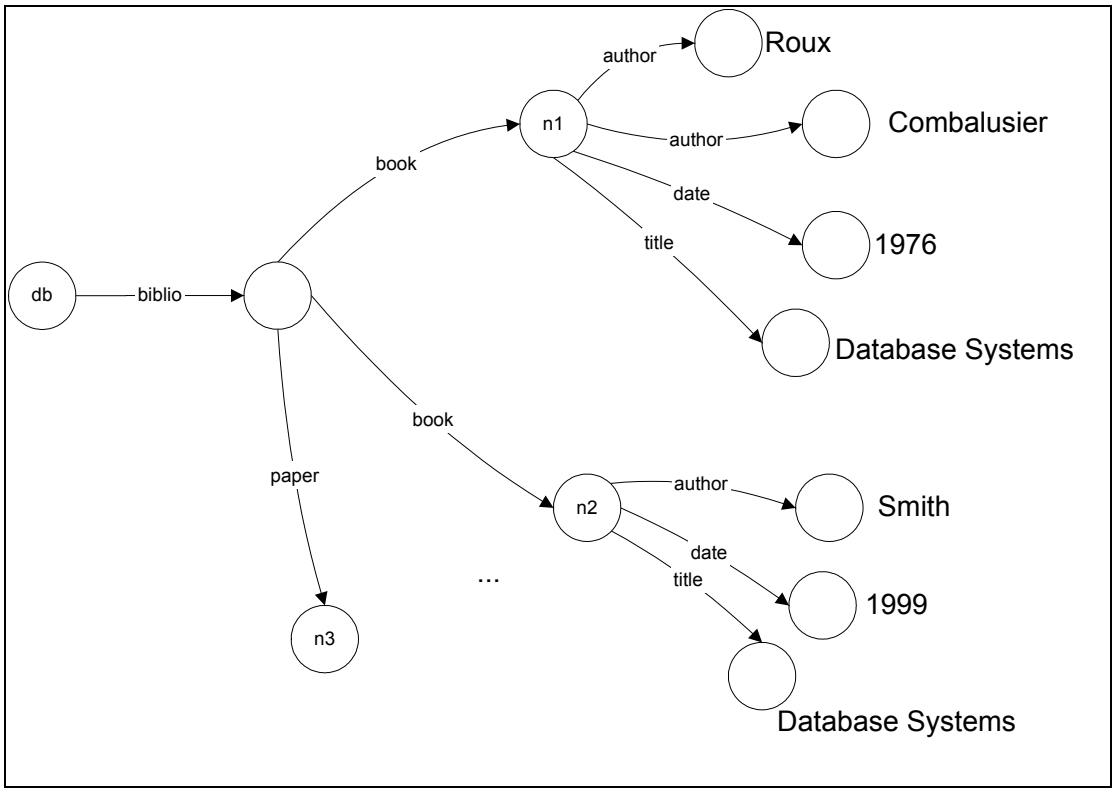
- Εκφραστική δύναμη[ABS00]: Λόγω της μορφής των δεδομένων και του γεγονότος της ασάφειας στον τρόπο οργάνωσης τους, η γλώσσα θα πρέπει να έχει μεγάλη εκφραστική ικανότητα. Θα μπορούσαμε να πούμε πως εάν περιγράφαμε μια σχεσιακή βάση με έναν OEM γράφο, τότε αυτή η γλώσσα θα ήταν πλήρης εάν μπορούσε να κάνει ό,τι ακριβώς κάνει η SQL. Παρ' όλα αυτά αυτό δεν είναι σωστό διότι στα ημιδομημένα δεδομένα δεν υπάρχει σαφώς καθορισμένη έννοια πληρότητας, και συνεπώς θα έπρεπε η συγκεκριμένη γλώσσα να έχει περισσότερες εκφραστικές ικανότητες απ' ότι η SQL για να μπορεί να ανταποκριθεί στην απρόβλεπτη δομή ενός OEM γράφου.

- Σημασιολογία[ABS00]: Λόγω του ότι αυτό το μοντέλο στηρίζεται πολύ για την περιγραφή των δεδομένων στο συντακτικό, μια αυστηρά καθορισμένη σημασιολογία βοηθά πολύ στον τρόπο περιγραφής των δεδομένων.

- Ικανότητα στοιχειοθέτησης[ABS00]: Αυτό γενικά σημαίνει ότι μια ερώτηση μπορεί να υποβληθεί στο αποτέλεσμα μιας άλλης. Αυτό συνεπάγεται ότι το αποτέλεσμα της ερώτησης ακολουθεί τη σημασιολογία του μοντέλου περιγραφής των δεδομένων.

- Δομικό μοντέλο[ABS00]: Παρόλο που τα ημιδομημένα δεδομένα δεν ακολουθούν κάποιο συγκεκριμένο μοντέλο, οι γλώσσες θα πρέπει να είναι σε θέση να «συνειδητοποιούν» κάποιο δομικό μοντέλο, αν αυτό υπάρχει στα δεδομένα, ούτως ώστε να μπορούν να το εκμεταλλευτούν για βελτιστοποίηση και άλλες λειτουργίες.
- Προγραμματιστική χρησιμοποίηση[ABS00]: Με βάση τα σημερινά δεδομένα οι περισσότερες SQL ερωτήσεις παράγονται από κάποιο πρόγραμμα και όχι από ανθρώπους. Συνεπώς για να συμβαίνει αυτό και για άλλες γλώσσες επερωτήσεων χρειάζεται ο ορισμός αρχικά μιας απλής αλλά εκφραστικά αρκετά δυνατής βασικής γλώσσας. Αυτό μπορεί βέβαια να είναι εις βάρος της φιλικότητάς της προς το χρήστη, αλλά είναι σαφώς πιο χρήσιμο.

Όλες οι γλώσσες επερωτήσεων για ημιδομημένα που έχουν οριστεί έχουν την ικανότητα να διατρέχουν διάφορα μονοπάτια και να φτάνουν σε οποιοδήποτε βάθος στο γράφο των δεδομένων. Γι' αυτό το λόγο όλες αυτές οι γλώσσες ορίζουν κάποια μορφή εκφράσεων μονοπατιών (path expressions). Μια έκφραση μονοπατιού στους OEM γράφους που ορίσαμε παραπάνω είναι η ακολουθία κάποιων ακμών του γράφου, που οδηγούν σε κάποιους κόμβους. Έστω για παράδειγμα ότι έχουμε τον παρακάτω OEM γράφο, ο οποίος ορίζει μια βάση δεδομένων με βιβλία:



Σχήμα 1

Κάποια απλά path expressions είναι το biblio.book και το biblio.book.title. Το πρώτο φτάνει στους κόμβους n1 και n2 και συνεπώς εάν το θεωρούσαμε μια απλή ερώτηση θα τους επέστρεφε ως αποτέλεσμα. Το δεύτερο path expression φτάνει στους ατομικούς κόμβους με τιμές Smith, Roux και Combalusier και αντίστοιχα τους επιστρέφει ως αποτέλεσμα.

Ο ορισμός μιας έκφρασης μονοπατιού $l_1,l_2,\ldots,l_n$ είναι το σύνολο των κόμβων v_n για τα οποία υπάρχουν οι ακμές (r,l_n,v_n), (r,l_n,v_n),… ,(r,l_n,v_n), όπου r είναι η ρίζα του γράφου. Το path expression είναι η βασική δομική μονάδα όλων των γλωσσών επερωτήσεων ημιδομημένων δεδομένων. Με βάση αυτά και με τη χρησιμοποίηση ειδικών συμβόλων και κανονικών εκφράσεων διανθίζονται αυτές οι γλώσσες και αποκτούν και άλλες ικανότητες όπως επιλογής διαφορετικών μονοπατιών από την ίδια έκφραση ή την παρεμβολή κάποιου μονοπατιού ανάμεσα σε κάποιο άλλο που μας ενδιαφέρει.

Τα παραπάνω όμως δεν φτάνουν για να ορίσουν μια πλήρη γλώσσα επερωτήσεων. Αυτή θα πρέπει να υποστηρίζει και λειτουργίες πάνω στους κόμβους (διαγραφή, αλλαγή τύπου, προσθήκη κτλ.) καθώς και πρόσθετες πράξεις πάνω στα δεδομένα όπως π.χ. η ένωση. Αυτές οι λειτουργίες παρέχονται από τις υπόλοιπες λειτουργίες που είναι κοινές για όλες τις γλώσσες επερωτήσεων. Τα path expressions όμως είναι η κύρια δομική μονάδα των γλωσσών που αφορούν τα ημιδομημένα δεδομένα, αφού αυτά είναι που ουσιαστικά ορίζουν τη λειτουργίας τους.

2.1.3 Η γλώσσα Lorel

Η γλώσσα Lorel, σαν γλώσσα που εφαρμόζεται πάνω σε OEM γράφους, είναι λογικό να έχει ως κύρια δομική μονάδα τις εκφράσεις μονοπατιών. Παρακάτω παρατίθεται σύντομη περιγραφή των κυριότερων στοιχείων του συντακτικού της Lorel[ABS00], τα οποία αποτελούν τον πυρήνα της αφού μέσω αυτών γίνονται οι κυριότερες λειτουργίες της πάνω στα δεδομένα.

Το βασικό συντακτικό μιας Lorel ερώτησης είναι το εξής:

```
select      label: X
from        l1,l2,…,ln X
```

Με αυτήν την ερώτηση παίρνουμε τα αντικείμενα στα οποία οδηγεί το μονοπάτι $l_1,l_2,\ldots,l_n$ και δημιουργούμε ένα νέο γράφο ο οποίος ξεκινάει από ένα κόμβο ρίζα ο οποίος οδηγεί μέσω ακμών label στα αντικείμενα αυτά.

Εάν θέλουμε να επιβάλουμε κάποιο περιορισμό στο αντικείμενο X, τότε αυτό στη Lorel γίνεται ως εξής:

```
select      label:X
from        l1,l2,…,ln X
where       X = "value"
```

ή:

```
select      label:X
from        l1,l2,…,ln X
where       "value" in X
```

Σε γενικές γραμμές οι ερωτήσεις της Lorel λειτουργούν ως εξής: Αρχικά γίνεται επεξεργασία της φράσης from, μετά της φράσης where και τέλος της select. Έτσι αρχικά διαλέγονται όλα τα αντικείμενα του γράφου που ικανοποιούν τα path expressions του from clause και δημιουργούνται σύνολα αυτών που κάθε ένα αντιστοιχεί σε μια μεταβλητή που ορίζεται στην from. Στη συνέχεια αυτά τα σύνολα φιλτράρονται έτσι ώστε να ικανοποιούν τις απαιτήσεις της where φράσης. Τέλος φτιάχνουμε την ανάλογη SSD έκφραση η οποία περιγράφει το αποτέλεσμα. Για κάθε μεταβλητή που υπάρχει στην select δημιουργείται ένα νέο αντικείμενο το οποίο δείχνει στους κόμβους που είναι έγκυροι μετά την επεξεργασία της ερώτησης. Εάν στην select υπάρχουν δύο οι περισσότερες μεταβλητές (έστω n), τότε στο αποτέλεσμα θα δημιουργηθούν n+1 κόμβοι. Ο παραπάνω κόμβος θα είναι η ρίζα του γράφου της ερώτησης οποίος θα δείχνει στους n νέους κόμβους οι οποίοι με τη σειρά τους θα δείχνουν στους κόμβους του αρχικού γράφου που αντιστοιχούν στις n μεταβλητές του select.

Η Lorel υποστηρίζει και φώλιασμα άλλης ερώτησης μέσα στη select φράση. Αυτός είναι ένας άλλος τρόπος για τη δημιουργία πολλών καινούριων κόμβων στη select. Κάποια τέτοια παραδείγματα είναι τα εξής:

```
select      row: (select      author: Y
                  from        X.author Y)
from        biblio.book X
```

και:

```
select      row: (select      row:{author: Y, title: T}
                  from        X.author Y,
                              X.title T)
from        biblio.book X
where       "Roux" in X.author
```

Η πρώτη ερώτηση θα επιστρέψει τα ονόματα των συγγραφέων των βιβλίων του γράφου, ενώ η δεύτερη θα έχει ως αποτέλεσμα τους συγγραφείς και τους τίτλους των βιβλίων των οποίων ο ένας συγγραφέας είναι ο Roux.

Ένα πολύ σημαντικό χαρακτηριστικό της Lorel έχει να κάνει με τον τρόπο που χρησιμοποιεί τους τελεστές ισότητας και ανισότητας. Έστω ότι έχουμε μια ερώτηση που στη φράση where περιέχει τα εξής:

```
X.author = "Smith"
```

Εδώ έχουμε την σύγκριση ενός συνόλου κόμβων και ενός συνόλου χαρακτήρων, δηλαδή τελείως διαφορετικών τύπων δεδομένων. Αυτό που ουσιαστικά σημαίνει η παραπάνω ισότητα είναι το εξής:

```
exists Y in X.author: Y = "author"
```

Δηλαδή δεν είναι η ισότητα μεταξύ ενός συνόλου κόμβων αλλά η επιλογή ενός συνόλου του οποίου κάθε στοιχείο (κόμβος) έχει τη συγκεκριμένη τιμή. Με το ίδιο σκεπτικό στη Lorel ο χρήστης δεν χρειάζεται να ασχοληθεί με τον τύπο των δεδομένων που συγκρίνει μεταξύ τους. Όταν δηλαδή σχηματίζει μια ανισότητα ή μια ισότητα π.χ. X.year > 1998, η Lorel θα επιστρέψει τους κόμβους που έχουν τιμή μεγαλύτερη του 1998, είτε αυτή η τιμή είναι ένας ακέραιος αριθμός ή απλά ένα σύνολο χαρακτήρων.

Τέλος, ένα ακόμα σημαντικό στοιχείο της Lorel είναι ότι μια ερώτηση μπορεί να περιέχει εκτός από μεταβλητές κόμβων, (σαν αυτές που παρουσιάστηκαν στα παραπάνω παραδείγματα), και μεταβλητές που αναφέρονται σε ετικέτες ακμών ή και μονοπατιών. Με αυτόν τον τρόπο υπάρχει δυνατότητα επιστροφής των labels που οδηγούν σε κάποιο αντικείμενο καθώς και η δυνατότητα πράξεων πάνω στα labels με αποτέλεσμα την επιλογή συνόλων κόμβων και αποτελεσμάτων που δεν γίνονται με απλά path expressions.

2.2 Πολυδιάστατα ημιδομημένα δεδομένα

Τα πολυδιάστατα δεδομένα είναι ένα μοντέλο θεώρησης δεδομένων, με τη βοήθεια του οποίου είναι δυνατή η οργάνωση και διαχείριση ημιδομημένων δεδομένων με έναν εμπλουτισμένο τρόπο. Τα πολυδιάστατα ημιδομημένα δεδομένα είναι βασικά και αυτά ημιδομημένα δεδομένα, που παρουσιάζουν όμως διαφορετικές όψεις (facets), δηλαδή διαφορετικό περιεχόμενο, κάτω από διαφορετικά περιβάλλοντα που αυτά μπορούν να έχουν υπόσταση.

Τα περιβάλλοντα αυτά ονομάζονται κόσμοι (worlds), και προκειμένου να οριστούν χρησιμοποιούμε ένα σύνολο παραμέτρων, οι οποίες ονομάζονται διαστάσεις (dimensions). Καταυτόν τον τρόπο, ένας κόσμος ορίζεται μέσω της απόδοσης σε κάθε διάσταση μιας τιμής. Το ζεύγος διάσταση-τιμή αποτελεί ουσιαστικά την περιγραφή μιας συνθήκης που ικανοποιείται στα πλαίσια αυτού του κόσμου.

Για παράδειγμα, θα μπορούσαμε να περιγράψουμε έναν κόσμο ορίζοντας την τιμή των διαστάσεων: γλώσσα, λεπτομέρεια, μορφή αρχείου. Έτσι, αν δώσουμε τις αντίστοιχες τιμές ελληνικά, υψηλή, PDF, θα έχουμε μια έκφραση της μορφής:

```
[lang=gr, detail=high, format=pdf]
```

Η έκφραση αυτή ονομάζεται στα πλαίσια των πολυδιάστατων ημιδομημένων δεδομένων "context specifier". Ένας context specifier μπορεί να ορίζει όχι μόνο μεμονωμένους κόσμους, αλλά και σύνολα αυτών, όπως στο εξής παράδειγμα:

```
[lang=gr, detail=high]
```

όπου αναφερόμαστε στους κόσμους όπου η γλώσσα είναι ελληνικά, η λεπτομέρεια είναι υψηλή και το format μπορεί να είναι οτιδήποτε.

Με αυτό τον τρόπο είναι δυνατό να έχουμε διαφορετικά στιγμιότυπα της ίδιας πληροφορίας, με το καθένα από αυτά να ισχύει κάτω από ένα διαφορετικό σύνολο κόσμων. Μια τέτοια πληροφοριακή οντότητα που περικλείει έναν αριθμό από διαφορετικά στιγμιότυπα (όψεις) λέγεται Πολυδιάστατη Οντότητα (Multidimensional Entity). Εάν οι όψεις e_1, e_2, .. e_n μιας πολυδιάστατης οντότητας ισχύουν κάτω από ένα κόσμο w (ή κάτω από κάθε κόσμο που ορίζει ένας context specifier c), τότε λέμε ότι η e αποτιμάται σε e_1, e_2, .. e_n κάτω από τον w (ή κάτω από τον c).

2.2.1 Το μοντέλο MOEM

Το μοντέλο MOEM είναι μια επέκταση του OEM [SG01], η οποία διατηρεί την ευελιξία του OEM, αλλά παράλληλα εισάγει σε αυτό την έννοια των κόσμων κάτω από τους οποίους ισχύουν τα δεδομένα που είναι αποθηκευμένα στο γράφο.

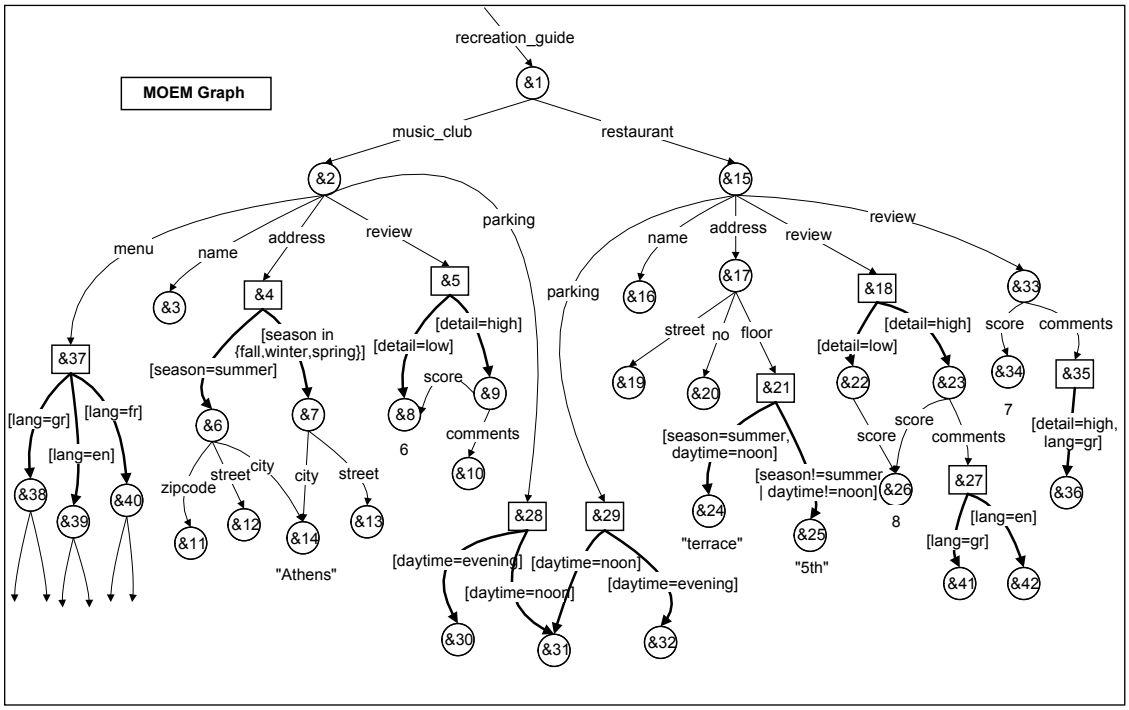
Δύο είναι τα βασικά στοιχεία που υλοποιούν αυτή την επέκταση:

- **Πολυδιάστατοι Κόμβοι**: Ένας πολυδιάστατος κόμβος (multidimensional node) αναπαριστά μία πολυδιάστατη οντότητα και χρησιμοποιείται για την ομαδοποίηση κόμβων που παριστάνουν διαφορετικές όψεις της οντότητας αυτής. Στο συγκεκριμένο μοντέλο δεδομένων που περιγράφουμε, οι πολυδιάστατοι κόμβοι έχουν ορθογώνιο σχήμα για να τους διακρίνουμε από τους συμβατικούς κυκλικούς κόμβους.
- **Context Ακμές**: Οι context ακμές είναι κατευθυνόμενες ακμές που συνδέουν έναν πολυδιάστατο κόμβο με τις εναλλακτικές του μορφές. Το όνομα μιας context ακμής που δείχνει στη όψη r, είναι ένας context specifier που ορίζει το σύνολο κόσμων κάτω από τους οποίους η r έχει υπόσταση. Οι context ακμές σχεδιάζονται σαν παχιές ή διπλές γραμμές, για να διακρίνονται από τις συμβατικές ακμές.

Το νέο μοντέλο που περιγράφουμε ονομάζεται *Multidimensional Object Exchange Model* (*MOEM*). Στο MOEM οι συμβατικοί κυκλικοί κόμβοι ονομάζονται context κόμβοι και αναπαριστούν εναλλακτικές όψεις σχετιζόμενες με κάποιο context. Οι συμβατικές ακμές του OEM (λεπτές γραμμές) ονομάζονται entity ακμές (οντοτήτων).

Όπως και στο OEM, όλοι οι MOEM κόμβοι θεωρούνται αντικείμενα και χαρακτηρίζονται από ένα μοναδικό αναγνωριστικό (object identifier). Στη συνέχεια, στο πλαίσιο του MOEM, οι όροι κόμβος και αντικείμενο θα χρησιμοποιούνται εναλλακτικά. Όσον αφορά στα context αντικείμενα, ισχύουν όσα έχουμε περιγράψει για τους OEM γράφους.

Στο Σχήμα 2 έχουμε ένα παράδειγμα ενός οδηγού ψυχαγωγίας σε μορφή MOEM γράφου. Για λόγους απλότητας ο γράφος δεν έχει αναπτυχθεί πλήρως και μερικοί από τους atomic κόμβους δεν έχουν τιμές.



Σχήμα 2

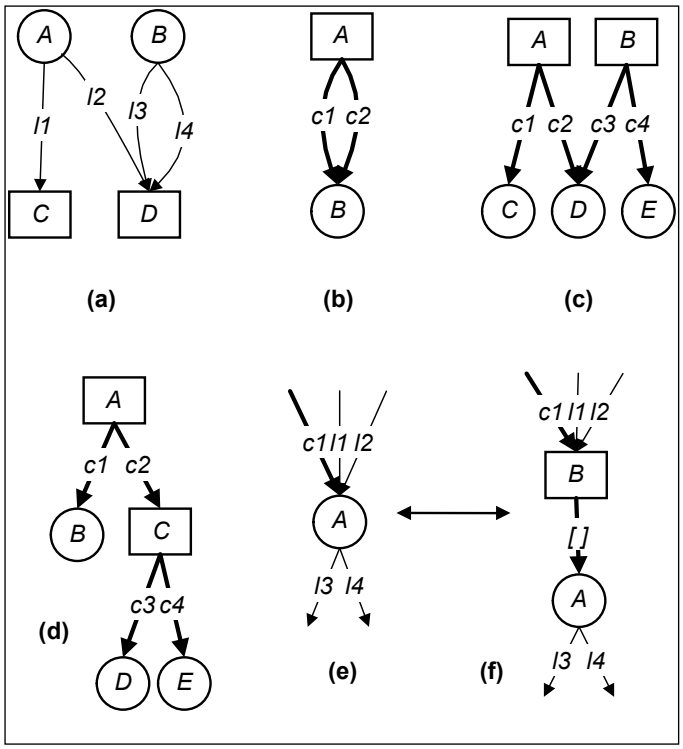
Για το παράδειγμα του σχήματος 2, οι διαστάσεις και τα πεδία τιμών τους είναι τα ακόλουθα:

- season με τιμές {summer, fall, winter, spring}
- daytime με τιμές {noon, evening}
- detail με τιμές {high, low} και
- lang με τιμές {en, fr, gr}

Το restaurant με αναγνωριστικό &15 κανονικά βρίσκεται στον πέμπτο όροφο, αλλά τα μεσημέρια του καλοκαιριού βρίσκεται στη βεράντα. Για αυτό το λόγο ο κόμβος με αναγνωριστικό &21 είναι ένα πολυδιάστατο αντικείμενο του οποίου η (atomic) τιμή εξαρτάται από τις διαστάσεις summer και daytime. Εκτός από διαφορετικές τιμές, τα context αντικείμενα μπορούν να έχουν και διαφορετική δομή, όπως στην περίπτωση των κόμβων &6 και &7 που είναι εναλλακτικές μορφές του πολυδιάστατου αντικείμενου address με αναγνωριστικό &4.

Μία context ακμή δεν μπορεί να ξεκινά από έναν context κόμβο και μία απλή ακμή δεν μπορεί να ξεκινά από έναν πολυδιάστατο κόμβο. Αυτοί οι δύο είναι όλοι οι περιορισμοί στη μορφολογία του MOEM γράφου.

Στο σχήμα 3 φαίνονται μερικές όχι και τόσο απλές, αλλά νόμιμες MOEM δομές. Στο παράδειγμα (b) περισσότερες από μια context ακμές συνδέουν έναν πολυδιάστατο κόμβο με έναν άλλο context κόμβο. Ο πολυδιάστατος κόμβος A αποτιμάται στον B κάτω από την ένωση των κόσμων που ορίζονται από τους c_1, c_2. Έτσι οι δύο context ακμές μπορούν να αντικατασταθούν με μία με context specifier που εκφράζει την τομή των συνόλων που εκφράζουν οι c_1, c_2. Το παράδειγμα (c) δείχνει έναν context κόμβο D που αποτελεί κομμάτι δύο πολυδιάστατων κόμβων A και B. Αυτή είναι η περίπτωση του κόμβου &31 στο παράδειγμα του σχήματος 2. Μια πολυδιάστατη οντότητα μπορεί να είναι μέρος μιας άλλης πολυδιάστατης οντότητας , όπως φαίνεται στο παράδειγμα (d). Η δομή του παραδείγματος (e) είναι ισοδύναμη με αυτή του (f): από την οπτική των l_1, l_2 ο context κόμβος A δε σχετίζεται μέσω μιας context ακμής, και έτσι θεωρείται η μοναδική όψη της πολυδιάστατης οντότητας που αναπαρίσταται με B και έχει νόημα κάτω από οποιοδήποτε κόσμο.



Σχήμα 3

Εύκολα καταλαβαίνουμε ότι το OEM είναι μια ειδική περίπτωση πολυδιάστατου γράφου δεδομένων, όπου δεν υπάρχουν πολυδιάστατοι κόμβοι και context ακμές.

Στο [SG01] παραθέτονται τυπικοί ορισμοί για όλες τις προηγούμενες έννοιες. Στην ίδια εργασία ορίζονται και αλγόριθμοι με τη βοήθεια των οποίων μπορούμε να λάβουμε τον OEM γράφο κάθε κόμβος του οποίου αποτελεί όψη ενός MOEM γράφο κάτω από έναν συγκεκριμένο κόσμο (reduction).

Όπως έχει αναπτυχθεί στο [ABS00] ο θεμελιώδης λίθος του συντακτικού για ημιδομημένα δεδομένα είναι η *ssd-expression*. Στο [SG01] έχει οριστεί η *mssd-expression* επεκτείνοντας το συντακτικό για SSD ώστε να συμπεριλάβει context specifier. Η γραμματική για mssd-expressions δίνεται παρακάτω σε μορφή *Extended Backus-Naur Form* (EBNF), όπου όσα σύμβολα αρχίζουν με κεφαλαίο μπορούν να οριστούν με μια κανονική έκφραση.

```
mssd-expr          ::=   val | Oid value | Oid
value              ::=   Atomicvalue | "{" comlexvalue "}"
                         | "{" multidimvalue "}"
complexvalue       ::=   Label ":" mssd-expr ("," complexvalue)?
multidimvalue      ::=   contspec ":" mssd-expr ("," multidimvalue)?
```

Είναι προφανές ότι η *multidimvalue* αντιστοιχεί στον πολυδιάστατο κόμβο του MOEM, ενώ οι *atomicvalue* και *complexvalue* αντιστοιχούν σε context κόμβους. Σημειώστε ότι οι *multidimvalues* μπορούν να έχουν κάποιο αναγνωριστικό, όπως ακριβώς και complex ή atomic τιμές. Οι συμβάσεις που αφορούν τη σύνταξη των ετικετών, των atomic τιμών και των αναγνωριστικών αντικειμένων όπως και οι απαιτήσεις για συνέπεια (consistency) των ssd-expressions [ABS00] ισχύουν και για τις mssd-expressions.
Ένας context specifier είναι της μορφής:

```
contspec           ::=   "[" contspecclause ("|" contspecclause)* "]"
contspecclause     ::=   "" | "-" | dimlist
dimlist            ::=   dimspec ("," dimspec)*
dimspec            ::=   dimname (atomicop Dimvalue
                                  | setop "{" setdimvalue "}" )
atomicop           ::=   "=" | "!="
setop              ::=   "in" | "not in"
setdimvalue        ::=   Dimvalue ("," Dimvalue)*
```

Σαν παράδειγμα παραθέτουμε την ακόλουθη mssd-expression που περιγράφει το αντικείμενο music_club με αναγνωριστικό &2 στο σχήμα 2.

```
&2 {
      menu: &37(
            [lang=gr]: &38 "Greek menu",
            [lang = en]: &39 "English menu",
            [lang = fr]: &40 "French menu"
      ),
      name: &3,
      address: &4 (
            [season=summer]: &6 {
                  zipcode : &11,
                  street  : &12,
                  city    : &14 "Athens"
            },
            [season in {fall,winter,spring}] : &7 {
                  city : &14,
                  street : &13
            }
      ),
      review: &5 (
            [detail=low]: &8 6,
            [detail=high]: &9 {
                  score: &8,
                  comments:&10
            }
      ),
      parking: &28 (
            [daytime=evening]: &30,
            [daytime=noon]   : &31
      )
}
```

2.2.2 Η γλώσσα MQL [Sta02]

Όπως και στην περίπτωση του OEM μοντέλου για το οποίο ορίστηκε η Lorel, έτσι και στην περίπτωση των πολυδιάστατων ημιδομημένων δεδομένων έχει αναπτυχθεί σε θεωρητικό επίπεδο η ανάλογη γλώσσα επερωτήσεων για MOEM γράφους η οποία ονομάστηκε MQL (Multidimensional Query Language). Να επισημάνουμε ότι για την παρακάτω ανάλυση θεωρούμε ότι αναφερόμαστε σε MOEM γράφους που βρίσκονται σε κανονική μορφή. Βασική έννοια και αυτής της γλώσσας για ημιδομημένα δεδομένα είναι οι εκφράσεις μονοπατιού που περιλαμβάνουν contexts (context path expressions).

Η επιπρόσθετη ιδιότητά τους στην MQL μορφή σε σχέση με το απλό OEM μοντέλο είναι η εισαγωγή των context specifier qualifiers, και συνεπώς έχουμε την εισαγωγή της έννοιας του context path expression, τα οποία έχουν τη μορφή:

$[c_1]l_1, [c_2]l_2, \ldots [c_n]l_n$

Οι εκφράσεις μέσα στις αγκύλες είναι οι context specifier qualifiers. Η λειτουργία τους είναι να καθορίζουν κάτω από ποιους κόσμους ισχύουν οι entity ακμές που καθορίζονται από το (label) που τους ακολουθούν.

Ένα context path expression γυρνάει ως αποτέλεσμα αποκλειστικά context κόμβους και όχι multidimensional κόμβους. Ένας context κόμβος q συμπίπτει με ένα context path expression όταν:

1. υπάρχει ένα μονοπάτι στο γράφο $l_1.A_1…A_j.l_2.B_1…B_j,l_3,…,l_n.Z_1…Z_m$ το οποίο καταλήγει στο κόμβο q, όπου A_i, B_i, ... Z_i είναι ακμές context με i >= 0. Με άλλα λόγια υπάρχει ένα μονοπάτι στον γράφο το οποίο καταλήγει στον κόμβο q στο οποίο μεταξύ δύο entity ακμών μπορεί να υπάρχει ένας οποιοσδήποτε αριθμός context ακμών. Αυτό μπορεί να γραφτεί και ως εξής: $l_1._*.l_2._*,l_3,…,l_n._*$, όπου η έκφραση _* αντιστοιχεί μόνο σε ακμές context τύπου. Εδώ παρατηρούμαι ότι το συγκεκριμένο path καταλήγει πάντα σε _*, αφού διαφορετικά το μονοπάτι θα κατέληγε σε πολυδιάστατο κόμβο.
2. το inherited context των $l_1.A_1…A_j$ πρέπει να είναι υπερσύνολο του $[c_1]$, το inherited context των $l_2.A_2…A_m$ πρέπει να είναι υπερσύνολο του $[c_2]$ κοκ. Το υπερσύνολο ορίζεται σε επίπεδο κόσμων που ορίζονται από κάθε context specifier qualifier. Το context α είναι υπερσύνολο του context β όταν η τομή των είναι ίση με το β.

Με βάση τα παραπάνω ορίζεται το inherited context ενός μονοπατιού, το οποίο είναι το μέγιστο σύνολο κόσμων το οποίο είναι υποσύνολο του inherited context όλων των ακμών (entity ή context) του συγκεκριμένου μονοπατιού. Το inherited context του path δηλαδή ορίζει το σύνολο των κόσμων για τους οποίους υφίσταται το συγκεκριμένο μονοπάτι.

Εάν κάποιος context specifier qualifier ενός label του path παραλείπεται, τότε εννοείται ότι για το συγκεκριμένο label ισχύει ο context specifier qualifier του προηγούμενου, εκτός και εάν παραλείπεται αυτός του πρώτου label, οπότε για αυτό ισχύει το universal context specifier [], το οποίο περιλαμβάνει όλους τους κόσμους. Πέρα από αυτό ορίζεται και empty context [-], το οποίο είναι το υποσύνολο όλων των contexts, το οποίο σημαίνει πως δεν υπάρχουν περιορισμοί στα contexts. Παραδείγματα τέτοιων path expression είναι τα εξής:

$[c_1]l_1,l_2,…l_n = [c_1]l_1,[c_1]l_2,…[c_1]l_n$

$l_1,l_2,…[c_n]l_n = []l_1,[]l_2,…[c_n]l_n$

Έστω ότι έχουμε το εξής path:

```
[x]a.b.c
```

Ένας κόμβος που αντιστοιχεί σε αυτό το path είναι αντιστοιχεί και στο μονοπάτι a.b.c όλων των συμβατικών OEM γράφων που προκύπτουν από partial reduction για τους κόσμους που ορίζει το [x]. Σε περίπτωση που το [x] ορίζει διαφορετικά σύνολα κόσμων έχουμε και πάνω από ένα σύνολα από partially reduced OEM γράφους. Με λίγα λόγια το [x]a.b.c επιστρέφει ως αποτέλεσμα τους κόμβους που είναι προσβάσιμοι μέσω του a.b.c κάτω από όλους τους κόσμους του ενός από τα διαφορετικά σύνολα κόσμων που ορίζονται από το [x]. Αυτό δεν είναι το ίδιο με το να υπολογίσουμε το αποτέλεσμα του a.b.c στο γράφο που προκύπτει από το partial reduction του MOEM γράφου στο [x]. Αυτό θα έκοβε από τον MOEM γράφο ότι δεν ισχύει στο [x], ενώ τα context path expressions θέλουν να καταλήγουν σε κόμβους που ισχύουν για κάποιο υπερσύνολο του [x].

Έστω ότι έχουμε την εξής έκφραση:

```
[x]a.b.[y]c
```

Η έννοια αυτού του path μπορεί να εξηγηθεί εάν θεωρήσουμε την εξής έκφραση με τη χρήση μιας μεταβλητής P:

```
[x]a.b P,
[y]P.c
```

Η μεταβλητή P αντιστοιχεί σε κόμβους που είναι αποτέλεσμα του πρώτου context path expression. Συνεπώς το y στη δεύτερη έκφραση επιδρά πάνω στο c και όχι στο P, αφού το τελευταίο αντιστοιχεί σε κόμβους και όχι σε ακμές. Συνεπώς η δεύτερη έκφραση θεωρεί ως ρίζα κάθε κόμβο του συνόλου P και επιστρέφει τους κόμβους που είναι προσβάσιμοι μέσω τις ακμής c για όλους τους κόσμους που ορίζει το [y], όπως εξηγείται και παραπάνω.

Όπως αναφέρθηκε και παραπάνω τα context paths επιστρέφουν μόνο context κόμβους και δεν δίνουν τη δυνατότητα περιήγησης στο γράφο μέσω των context ακμών αυτού. Για την ύπαρξη τέτοιας δυνατότητας έχουν οριστεί οι παρακάτω τελεστές:

<*context path expression***>**
<*context path expression***>:[***explicit_context***]**

Ο πρώτος τελεστής είναι αγκύλες που περικλείουν ένα context path expression και το οποίο επιστρέφει ως αποτέλεσμα ένα σύνολο από multidimensional κόμβους. Η συγκεκριμένη έκφραση δηλαδή είναι της μορφής l_1._*.l_2._*,l_3,…,l_n, όπως ορίστηκε παραπάνω με τη διαφορά ότι τώρα καταλήγει σε entity edge. Έτσι, στην περίπτωση που ο γράφος είναι κανονικής μορφής το αποτέλεσμα είναι πάντα multidimensional κόμβος.

Η δεύτερη έκφραση κάνει ότι και η πρώτη με τη διαφορά ότι στη συνέχεια καταλήγει στους context κόμβους που οδηγούν ακμές με explicit context υπερσύνολο του σετ των κόσμων που ορίζεται από το [*explicit_context*]. Αυτό πάλι συμβαίνει μόνο αν ο MOEM γράφος είναι κανονικοποιημένος.

Μετά τη παραπάνω ανάλυση στα βασικά στοιχεία της MQL γλώσσας που είναι ορισμένα συγκεκριμένα για τους MOEM γράφους, μπορούμε να περιγράψουμε γενικά το συντακτικό της γλώσσας.

Μια MQL ερώτηση απαρτίζεται από τις παρακάτω φράσεις (clause):

1. **select**:

Όπως και στη Lorel, η λειτουργία της είναι η παρουσίαση και η μορφοποίηση των αποτελεσμάτων. Πέρα από αυτά όμως ορίζοντας κάποιο context specifier qualifier μπορούμε να κάνουμε reduction των αποτελεσμάτων σε κάποιες διαστάσεις. Για παράδειγμα:

```
select      [detail=high, lang= en] recreation_guide: X
from        recreation quide X
```

2. **context**:

Στην πρόταση context ορίζονται μεταβλητές που αντιστοιχούν σε κάποιο context με σκοπό να χρησιμοποιηθούν στη select. Οι τρόποι που ορίζονται η παραπάνω μεταβλητές είναι τρεις: Είτε με απευθείας ανάθεση κάποιου context clause, είτε σε σχέση με κάποια μεταβλητή που έχει οριστεί στο from, είτε χρησιμοποιώντας ειδικές αθροιστικές συναρτήσεις. Π.χ. :

```
select      all_worlds: Y
context     [Y] as union([X])
from        [X] recreation quide A
```

3. **from**:

Στη συγκεκριμένη φράση ορίζονται νέα μεταβλητές που αντιστοιχούν είτε σε κόμβους στους οποίους καταλήγει κάποιο context path expression είτε σε κάποια ακμή, η οποία είναι μέρος κάποιου path expression. Στην τελευταία περίπτωση οι μεταβλητές ονομάζονται path variables.

Πρέπει να επισημανθεί ότι οι context specifier qualifiers κάθε μονοπατιού δεν επιδρούν πάνω στις μεταβλητές που αντιστοιχούν σε κόμβους λόγω του ότι το context γενικότερα ορίζεται πάνω σε ακμές. Ένα παράδειγμα ενός from clause είναι το εξής:

```
select floor:[X,Y]
from  recreation_quide.restaurant X,
      [season = winter]X.adress.floor Y,
      [season = summer, daytime = noon]X.address.floor Z
```

Εάν στις αλλαγές γραμμών δεν οριστεί κάποιος qualifier τότε δεν ισχύει αυτός της προηγούμενης γραμμής αλλά ισχύει ο universal. Στο παραπάνω παράδειγμα θα μπορούσαμε να ορίσουμε και μεταβλητές πάνω στο μονοπάτι (path variables) ως εξής:

```
select &P:X
from  recreation_quide.restaurant X,
      [season = winter]X.address@P Y
```

όπου P είναι η εν λόγω μεταβλητή, και ακόμα και context μεταβλητές που θα περιγραφούν παρακάτω.

4. **where**:

Όπως και στη Lorel, με τη συγκεκριμένη πρόταση εισάγουμε περιορισμούς και ορίζουμε διάφορες πράξεις πάνω στους κόμβους που επιστρέφονται από τη from clause.

5. **within**:

Το within ορίζει περιορισμούς στις μεταβλητές τύπου context που έχουν οριστεί στο from. Οι περιορισμοί αυτοί εισάγονται ορίζοντας πράξεις (τομή, ένωση, διαφορά, κτλ.) πάνω σε αυτές. Ουσιαστικά το within λειτουργεί τα context variables όπως λειτουργεί το where για τις μεταβλητές κόμβων. Ας πάρουμε το παράδειγμα:

```
select      row: {context:X, menu: Y}
from        recreation_guide.music_club.[X]menu Y,
within      [X]*[lang in {gr, en}] !=[-]
```

Στο παραπάνω παράδειγμα η ερώτηση βρίσκει όλα τα μενού του music_club που είναι είτε στα αγγλικά είτε στα ελληνικά. Εάν αντί να χρησιμοποιούσαμε το within βάζαμε απευθείας το [lang in {gr, en}] στη θέση του X τότε το αποτέλεσμα δεν θα ήταν το ίδιο αλλά θα ήταν ίσο με όλα τα μενού που είναι γραμμένα και στις δύο γλώσσες.

2.3 Στόχοι διπλωματικής

Στα πλαίσια της εργασίας αυτής καλούμαστε να υλοποιήσουμε ένα σύστημα διαχείρισης βάσης δεδομένων που χρησιμοποιείται για την αποθήκευση και επεξεργασία πολυδιάστατων ημιδομημένων δεδομένων, σύμφωνα με το μοντέλο MOEM.

Η προσπάθειά μας επικεντρώνεται στην υποστήριξη από αυτό το σύστημα της γλώσσας MQL, της οποίας η υλοποίηση θα επιτευχθεί μέσω της γλώσσας Lorel. Συγκεκριμένα θα πρέπει να αναπτύξουμε ένα σχήμα, με τη βοήθεια του οποίου θα μεταφράζεται μια ερώτηση MQL πάνω σε έναν γράφο MOEM σε μία ισοδύναμη ερώτηση Lorel σε έναν γράφο OEM. Ο γράφος OEM θα πρέπει να έχει τέτοια μορφή, ώστε να είναι δυνατό να αποθηκεύεται σε αυτόν το σύνολο της πληροφορίας που παριστάνεται από τον αντίστοιχο MOEM γράφο.

Το σύστημα επερωτήσεων που θα προκύψει από την εργασία θα ενσωματωθεί στο σύστημα αναπαράστασης και διαχείρισης MOEM γράφων MSSDesigner, έτσι ώστε το συγκεκριμένο εργαλείο να έχει τη δυνατότητα υποβολής επερωτήσεων στις πολυδιάστατες βάσεις δεδομένων και παρουσίαση των αποτελεσμάτων των ερωτήσεων με γραφική μορφή.

3

Ανάλυση και σχεδίαση

3.1 Περιγραφή αρχιτεκτονικής

Όταν μια γλώσσα επερωτήσεων είναι πλήρως ορισμένη σε θεωρητικό επίπεδο και πρέπει να γίνει μια υλοποίησή της, αρχικά ο πρώτος στόχος είναι να υλοποιηθεί ο πυρήνας της , όπως αυτός έχει οριστεί στην εισαγωγή, δηλαδή οι βασικές της συντακτικές μονάδες οι οποίες επιτρέπουν την περιήγηση στα δεδομένα και την επιστροφή των αποτελεσμάτων. Πέρα από αυτό, σε αυτό το πρωταρχικό στάδιο της υλοποίησης, δεν είναι σημαντικό να αναπτυχθεί ένα αυτόνομο σύστημα με σημαντικές επιδόσεις αλλά ένα σύστημα που να επιδεικνύει τις δυνατότητες της γλώσσας.

Συνεπώς και στην περίπτωση της MQL το σύστημα θα αναπτυχθεί δίνοντας βάση στην υλοποίηση της γλώσσας ανεξάρτητα της αυτονομίας και της επίδοσης του. Γι’ αυτό το λόγο αποφασίστηκε το συγκεκριμένο σύστημα να βασίζεται πάνω σε ένα ήδη ανεπτυγμένο και αξιόπιστο σύστημα διαχείρισης δεδομένων. Κατ’ αυτό τον τρόπο θα είμαστε σε θέση να ασχοληθούμε σχετικά γρήγορα και σε βάθος με την ίδια την MQL έχοντας ήδη βρει λύση στο πρόβλημα της ανάπτυξης ενός συστημικού υπόβαθρου που να εκτελεί τις απαραίτητες λειτουργίες χαμηλού επιπέδου.

Το πρόβλημα όμως που προκύπτει από τη συγκεκριμένη προσέγγιση είναι το γεγονός ότι το συγκεκριμένο μοντέλο δεδομένων που χρησιμοποιούμε θα πρέπει να μεταφραστεί πλήρως σε κάποιο άλλο μοντέλο δεδομένων που χρησιμοποιείται από το σύστημα που θα χρησιμοποιήσουμε. Το ίδιο βέβαια ισχύει και με τη γλώσσα που θέλουμε να χρησιμοποιήσουμε αφού το βασικό σύστημα που θα χρησιμοποιηθεί θα στηρίζεται σε κάποια άλλη γλώσσα επερωτήσεων.

Γι΄ αυτό το λόγο αρχικά θα αναπτυχθεί ένα μοντέλο OEM γράφου που θα εμπεριέχει όλη την πληροφορία ενός MOEM γράφου, καθώς και ένας αλγόριθμος μετατροπής ενός MOEM γράφου σε αυτό το OEM μοντέλο και αντίστροφα. Σε δεύτερο στάδιο θα αναπτυχθεί ένας αλγόριθμος μετατροπής μιας οποιασδήποτε MQL ερώτησης σε κάποια ισοδύναμη Lorel που να επιστρέφει και γενικά να έχει πρόσβαση στην ανάλογη πληροφορία του OEM μοντέλου.

Αυτός βέβαια είναι ο πυρήνας της υλοποίησης αφού πέρα από αυτές τις διαδικασίες θα πρέπει να αναπτυχθούν διάφορα υποσυστήματα μετατροπής του MOEM γράφου σε κανονική μορφή, κωδικοποίηση του context κάθε γράφου σε κάποια μορφή που να αποθηκεύεται εύκολα στον OEM γράφο, μετατροπής των δεδομένων που επιστρέφει το Lore σε κατάλληλη μορφή κτλ. Ένα σημαντικό στάδιο είναι και η ενσωμάτωση του όλου συστήματος σε κάποιο γενικότερο σύστημα διαχείρισης MOEM γράφων, ούτως ώστε να μπορέσουμε να χρησιμοποιήσουμε τους ήδη υπάρχοντες μηχανισμούς παρουσίασης και αποθήκευσης MOEM γράφων.

3.2 Περιγραφή λειτουργιών

3.2.1 Μεταφορά MOEM σε OEM

Στα προηγούμενα αναφέραμε την αναγκαιότητα της μεταφοράς της πληροφορίας ενός MOEM γράφου στη μορφή που μπορεί να διαχειριστεί η Lorel, δηλαδή αυτή ενός OEM γράφου. Όπως είναι πια γνωστό, ένας MOEM γράφος, σε αντίθεση με έναν OEM, περιέχει δύο τύπους ακμών (context και entity), καθώς και δύο τύπους κόμβων (multidimensional και context) Η επίλυση του προβλήματος της μεταφοράς θα επικεντρωθεί στην αναπαράσταση αυτών των ιδιαιτεροτήτων των MOEM γράφους σε όρους OEM.

Μπορούμε να επιχειρήσουμε μια συνοπτική περιγραφή της διαδικασίας μετατροπής, λέγοντας καταρχήν ότι και οι δύο τύποι MOEM κόμβων θα αντιστοιχούν στο μοντελοποιημένο OEM γράφο σε κοινούς OEM κόμβους. Όσον αφορά στους δύο τύπους MOEM ακμών, η μεταφορά τους στον OEM γράφο θα πρέπει να γίνεται με τέτοιο τρόπο, ώστε παράλληλα με τη σύνδεση των κόμβων μεταξύ τους με ακμές, όπως και στον MOEM, να αποθηκεύεται στον OEM γράφο και η πληροφορία όχι μόνο για το explicit context των context ακμών, αλλά και για το inherited context που χαρακτηρίζει και τους δύο τύπους ακμών.

Η αποθήκευση του inherited context, το οποίο αποτελεί πληροφορία που ορίζεται από τη μορφή και το περιεχόμενο του MOEM γράφου, αλλά δεν είναι άμεσα προσβάσιμη από αυτόν, είναι απαραίτητη για τη μετέπειτα μετάφραση των path expression της MQL σε ισοδύναμα path της Lorel, έτσι ώστε να μπορεί να γίνεται παράλληλα το qualification των ακμών που εμφανίζονται στο εκάστοτε path.

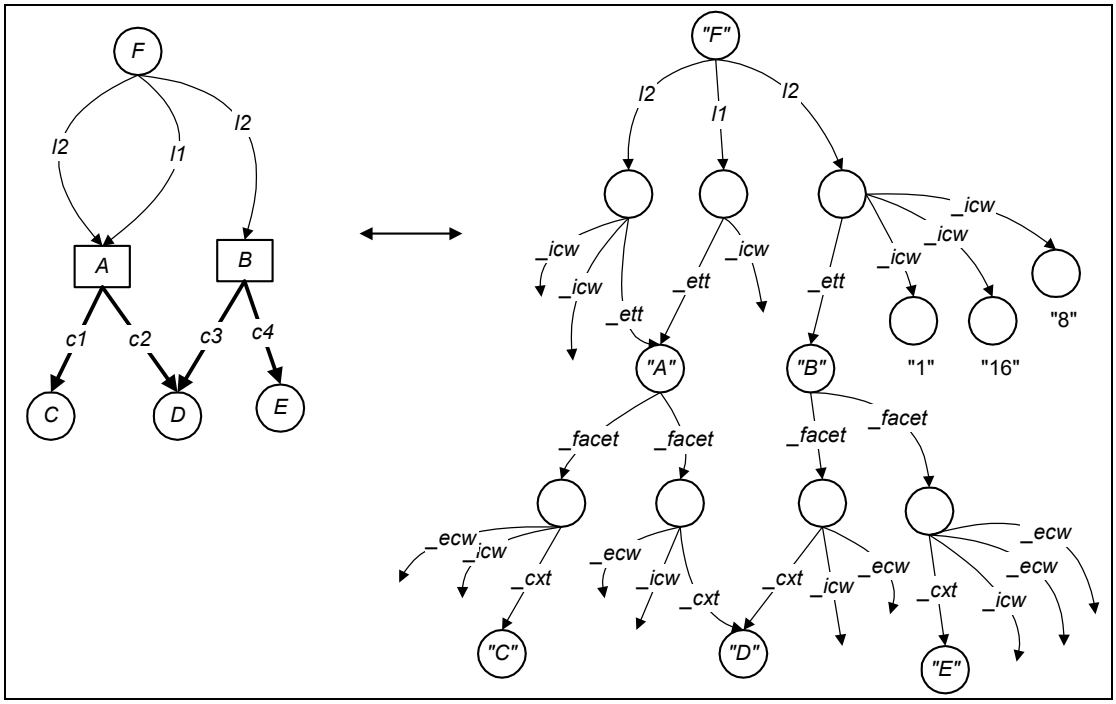
Προκειμένου λοιπόν να επιτευχθεί η αποθήκευση των context στις MOEM ακμές, καθεμιά από αυτές θα αντικαθίσταται στον ισοδύναμο OEM από έναν απλό κόμβο, στον οποίο θα καταλήγει μια ακμή από τον κόμβο-αφετηρία της MOEM ακμής, και από τον οποίο θα ξεκινά μια ακμή προς τον κόμβο-προορισμό της ίδιας ακμής. Ο διαχωρισμός μεταξύ entity και context ακμών μπορεί να γίνει με ειδική ονομασία των ακμών στον OEM γράφο. Οι ενδιάμεσοι κόμβοι που εμφανίζονται σε κάθε ακμή του αρχικού MOEM γράφου θα βοηθήσουν στη μεταφορά της πληροφορίας του context (explicit ή inherited) που συνοδεύει κάθε τέτοια ακμή.

Προκειμένου να παραστήσουμε τα σύνολα κόσμων που περιγράφει ένας context specifier, κατά τη μετατροπή του MOEM γράφου σε OEM θα υπολογιστούν αρχικά οι ακριβείς εκφράσεις κάθε κόσμου που ανήκει στο σύνολο αυτό. Στη συνέχεια θα πρέπει να αντιστοιχηθεί μοναδικά ένας κωδικός αριθμός για κάθε τέτοιο κόσμο. Ο σκοπός των παραπάνω είναι να δοθεί η δυνατότητα να συνδέονται μέσω ακμών οι ενδιάμεσοι κόμβοι που εμφανίζονται στην OEM αναπαράσταση κάθε ακμής του αρχικού MOEM γράφου με ειδικούς κόμβους, η τιμές των οποίων αντιστοιχούν σε κόσμους που ανήκουν στο context specifier που χαρακτηρίζει την εν λόγω MOEM ακμή. Με αυτόν τον τρόπο επιτυγχάνεται από τη μια πλευρά μια πλήρης μεταφορά της πληροφορίας των explicit ή και inherited context στον παραγόμενο OEM γράφο, ενώ από την άλλη αυτή η πληροφορία παρουσιάζεται στο γράφο με την απλούστερη μορφή, κατάλληλη για τις μετέπειτα συγκρίσεις ή και πράξεις συνόλων που θα απαιτούνται από τη διαδικασία αποτίμησης των διαφόρων path που θα δίνονται από το χρήστη σε MQL ώστε να μεταγλωττιστούν σε Lorel.

Θα πρέπει συνεπώς να προηγηθεί της διαδικασίας μεταφοράς MOEM σε OEM η απαρίθμηση και κωδικοποίηση των κόσμων που ανήκουν στο οικουμενικό context του εκάστοτε δεδομένου MOEM γράφου. Ως οικουμενικό, μπορούμε προφανώς να θεωρήσουμε το context του οποίου οι διαστάσεις και τα αντίστοιχα πεδία τιμών θα προκύπτουν από μια διάσχιση ολόκληρου του γράφου κατά την οποία θα λαμβάνονται όλες οι εκφράσεις που εμφανίζονται στα explicit context των ακμών του.

Οι πιο πάνω διαδικασίες θα αναλυθούν περισσότερο όταν θα περιγράψουμε αναλυτικά την συγκεκριμένη υλοποίηση που συνοδεύει την παρούσα εργασία. Για την ώρα, και για όσα ακολουθήσουν, θα θεωρήσουμε ότι οι διάφορες εκφράσεις context που μπορούμε να συναντήσουμε σε έναν MOEM γράφο θα βρίσκονται στη μορφή [w in $\{w_1, w_2, w_3\}$], όπου w_n είναι οι κωδικοί των κόσμων που συμπεριλαμβάνονται στο αντίστοιχο context.

Συνοψίζοντας τα παραπάνω, και πριν παρουσιάσουμε την περιγραφή ενός αλγορίθμου που θα αναλαμβάνει το μετασχηματισμό ενός MOEM γράφου σε OEM και αντίστροφα, παραθέτουμε το παράδειγμα ενός τέτοιου μετασχηματισμού:



Σχήμα 4

Μια αρχική παρατήρηση που πρέπει να κάνουμε είναι ότι χρησιμοποιούμε ακμές με label που αρχίζει με το χαρακτήρα _ (underscore) ως ειδικές ακμές για την οργάνωση της επιπλέον πληροφορίας που μεταφέρει ο MOEM γράφος. Θα πρέπει ως εκ τούτου να περιορίσουμε το σύνολο των δυνατών label που μπορεί να έχει μια ακμή στον αρχικό MOEM γράφο.

Με οδηγό το παραπάνω σχεδιάγραμμα, μπορούμε να προχωρήσουμε στην αναλυτική παρουσίαση αλγορίθμων με τη βοήθεια των οποίων μπορούμε να έχουμε μια επιτυχή μεταφορά ενός MOEM γράφου σε μορφή OEM, καθώς και το αντίστροφο.

Αλγόριθμος μετατροπής γράφου MOEM στον ειδικό OEM

Θεωρούμε γράφο MOEM που παριστάνεται στο σύστημά μας έστω με τη βοήθεια δύο πινάκων, του V και του E. Ο πρώτος περιλαμβάνει όλους τους κόμβους του γράφου, multidimensional, context και atomic, συνοδευόμενους από την αντίστοιχη τιμή τους. Για τους κόμβους που δεν είναι atomic αυτή η τιμή έστω ότι αντιστοιχεί στον τύπο τους, δηλαδή έστω ότι τιμή 'M' αναφέρεται σε multidimensional, ενώ τιμή 'C' σε complex. Ο δεύτερος πίνακας E περιέχει τις ακμές του γράφου σε μορφή διανυσμάτων (p, l, q), όπου l είτε η ετικέτα της αντίστοιχης entity ακμής, είτε η έκφραση του explicit context μιας context ακμής, p και q οι κόμβοι προέλευσης και κατάληξης της ακμής. Ο πίνακας E θα πρέπει να περιέχει επίσης για κάθε τέτοιο διάνυσμα και την πληροφορία για το inherited context της αντίστοιχης ακμής που παριστάνει, που θεωρούμε ότι έχει υπολογιστεί σε κάποιο προηγούμενο βήμα. Προφανώς η ακριβής μορφή της δομής δεδομένων στην οποία θα αποθηκεύεται ο MOEM γράφος είναι δυνατόν να διαφοροποιείται ανάλογα με τις απαιτήσεις της υλοποίησης. Θεωρούμε, τέλος, ότι ο γράφος έχει ήδη περάσει από το στάδιο της επαλήθευσης της εγκυρότητάς του, οπότε δεν αναμένεται να παρουσιαστεί κάποιο λάθος στην πιο κάτω διαδικασία.

Με βάση αυτή τη γενικευμένη απεικόνιση του γράφου MOEM, μπορούμε να κατασκευάσουμε τον ισοδύναμο OEM, χρησιμοποιώντας τον ίδιο τρόπο αναπαράστασης με αυτόν που περιγράψαμε για το MOEM[1]. Η διαδικασία μεταφοράς του MOEM σε μορφή OEM θα είναι η εξής:

- **Βήμα 1^{ο}**: Αρχικοποίηση του OEM γράφου: Αντιγραφή στον V πίνακα του OEM τα περιεχόμενα του V πίνακα του MOEM και αρχικοποίηση του κενού πίνακα E.
- **Βήμα 2^{ο}**: Με βάση το σύνολο των κόσμων που ανήκουν στο universal context του MOEM γράφου, πρόσθεσε στον OEM για κάθε κόσμο έναν κόμβο με τιμή τον αντίστοιχο κωδικό του
- **Βήμα 3^{ο}**: Για κάθε context ακμή h = (p, c, q) που εμφανίζεται στον πίνακα E του MOEM γράφου πρόσθεσε στον OEM έναν κόμβο x. Στη συνέχεια πρόσθεσε στον OEM ακμές h_1 = (p, _facet, x) και h_2 = (x, _cxt, q), και για κάθε κόσμο που περιλαμβάνεται στο c πρόσθεσε επιπλέον μία ακμή (x, _ecw, x_1), όπου x_1 ο κόμβος με τιμή τον κωδικό του αντίστοιχου κόσμου. Τέλος, για κάθε κόσμο w που περιέχεται στο inherited context της ακμής h, πρόσθεσε ακόμα μία ακμή (x, _icw, x_2) στον OEM, όπου x_2 κόμβος του οποίου η τιμή αντιστοιχεί στον w
- **Βήμα 4^{ο}**: Για κάθε entity ακμή h = (p, l, q) πρόσθεσε στον OEM κόμβο x, καθώς επίσης και τις ακμές h_1 = (p, l, x) και h_2 = (x, _ett, q). Επιπλέον, για κάθε κόσμο w που περιλαμβάνεται στο inherited context της h πρόσθεσε στον πίνακα E του OEM ακμή (x, _icw, x_1), με x_1 ο κόμβος με τιμή w

Αλγόριθμος μετατροπής ειδικού γράφου OEM στον αντίστοιχο MOEM

Για την αντίστροφη διαδικασία, θεωρούμε γράφο OEM που παριστάνεται από τους πίνακες της προηγούμενης παραγράφου. Θεωρούμε επίσης ότι γνωρίζουμε το universal context του γράφου που μας δίνεται, δηλαδή γνωρίζουμε τις διαστάσεις που εμπλέκονται στην έκφραση των context, τα πεδία τιμών τους καθώς και την κωδικοποίηση που έχουμε θεωρήσει για τους κόσμους που το συγκροτούν.

Αξίζει πάντως να τονιστεί ότι, σε αντίθεση με την αντίστροφη διαδικασία που παρουσιάστηκε πριν, θα φροντίσουμε τώρα να ελέγχουμε παράλληλα με την μετατροπή του γράφου OEM που θεωρούμε σαν είσοδο και την εγκυρότητά του, δηλαδή θα πρέπει να ανιχνεύονται τυχόν λάθη στη δομή του που τον καθιστούν μη έγκυρο γράφο αναπαράστασης MOEM γράφου σε μορφή OEM.

- **Βήμα 1^{ο}**: Αρχικοποίηση των πινάκων του γράφου MOEM: Ξεκίνησε από κενούς πίνακες
- **Βήμα 2^{ο}:** Για κάθε ακμή (p, _facet, q) στον πίνακα E του OEM γράφου, αντίγραψε στον MOEM γράφο τον κόμβο p και θέσε την τιμή του 'M' (αποτελεί δηλαδή multidimensional κόμβο), αν ο p δεν υπάρχει ήδη σε αυτόν

[1] Υπενθυμίζουμε ότι ένας OEM γράφος μπορεί να θεωρηθεί MOEM που δεν έχει κανένα πολυδιάστατο κόμβο, και ως εκ τούτου καμία entity ακμή.

- **Βήμα 3ο:** Για κάθε ακμή (p, l, q) του OEM γράφου με l τέτοιο ώστε να μην αρχίζει με τον χαρακτήρα _, αντίγραψε στον MOEM γράφο τον κόμβο p και θέσε την τιμή του 'C' (complex context κόμβος), αν δεν υπάρχει ήδη σε αυτόν
- **Βήμα 4ο:** Για κάθε φύλλο p στον OEM γράφο αντίγραψε το p στον MOEM γράφο.
- **Βήμα 5ο:** Για κάθε context κόμβο στον MOEM γράφο, βρες για τον αντίστοιχο κόμβο του OEM γράφου τις ακμές l που αναχωρούν από αυτόν, καθώς και τους κόμβους ο όπου αυτές καταλήγουν. Στη συνέχεια βρες τον κόμβο q όπου καταλήγει μία μόνο ακμή _ett από κάθε κόμβο ο. Αν για κάποια ακμή (p, l, o) το l αρχίζει με τον χαρακτήρα _, ή αν δεν βρεις μοναδική ακμή (o, _ett, q), επέστρεψε λάθος. Αλλιώς, για κάθε αλληλουχία ακμών (p, l, o), (o, _ett, q), πρόσθεσε μία ακμή (p, l, q) στον MOEM γράφο
- **Βήμα 6ο:** Για κάθε multidimensional κόμβο στον MOEM γράφο, βρες για τον αντίστοιχο κόμβο του OEM γράφου τις ακμές _facet που αναχωρούν από αυτόν, καθώς και τους κόμβους ο όπου αυτές καταλήγουν. Στη συνέχεια βρες τον κόμβο q όπου καταλήγει μία μόνο ακμή _cxt από κάθε κόμβο ο. Βρες επίσης όλες τις ακμές της μορφής (o, _ecw, w), και κατασκεύασε έναν context specifier c, ο οποίος θα αντιπροσωπεύει το σύνολο των κόσμων που προκύπτει από τους αντίστοιχους κωδικούς που είναι αποθηκευμένοι ως τιμές κάθε κόμβου w. Αν από το p ξεκινά ακμή με label διαφορετικό του _facet, ή αν δεν βρεις μοναδική ακμή (o, _cxt, q), ή αν ξεκινούν ακμές από τον ο που έχουν label διαφορετικό των _ecw, _icw, ή αν για κάποιο w η τιμή του δεν είναι κωδικός κόσμου που ανήκει στο universal context του γράφου, επέστρεψε λάθος. Αλλιώς, για κάθε αλληλουχία ακμών (p, _facet, o), (o, _cxt, q), πρόσθεσε μία ακμή (p, c, q) στον MOEM γράφο

3.2.2 Κανονικοποίηση

Προκειμένου να συνεχίσουμε στο σχεδιασμό μιας διαδικασίας που θα μπορεί να χειρίζεται γράφους της παραπάνω μορφής με σκοπό την υποβολή και εκτέλεση ερωτήσεων MQL πάνω στους MOEM γράφους που αυτοί αναπαριστούν, πρέπει προτού προβούμε στη μεταφορά ενός γράφου σε OEM γράφο, να λάβουμε πρώτα την κανονική μορφή του.

Η κανονική μορφή ενός MOEM γράφου ορίζεται ως ο μοναδικός MOEM γράφος που περιέχει την ίδια ακριβώς πληροφορία με την αρχική του μορφή, αλλά χαρακτηρίζεται από έναν επιπλέον περιορισμό στη μορφή του, δηλαδή το γεγονός ότι σε αυτόν κάθε entity ακμή μπορεί να καταλήγει μόνο σε πολυδιάστατο κόμβο, ενώ αντίστοιχα κάθε context ακμή δείχνει μόνο σε context κόμβους. Για τη μορφή αυτή και για τους γράφους που μας ενδιαφέρουν μπορούμε επίσης να καθορίσουμε η ρίζα του να είναι πολυδιάστατος κόμβος, περιορισμός που συμφωνεί με τη σύμβαση ύπαρξης entity ακμής από το πουθενά που δείχνει στη ρίζα ενός MOEM γράφου.

Συνέπεια αυτού του περιορισμού είναι το γεγονός ότι για οποιοδήποτε γράφο και καθώς διασχίζουμε τους κόμβους του, ένας context κόμβος θα ακολουθείται πάντα από έναν πολυδιάστατο, ο οποίος με τη σειρά του θα ακολουθείται από έναν context κόμβο, κ.ο.κ. Η αλληλουχία αυτή θα ισχύει σε όλο το γράφο μέχρι να καταλήξει σε έναν ατομικό κόμβο, ή να μπει σε κυκλική διαδρομή.

Η εκ των προτέρων γνώση της αλληλουχίας του τύπου των ακμών, άρα και των κόμβων σε έναν MOEM γράφο μας δίνει ένα πολύ σημαντικό πλεονέκτημα προκειμένου να αναπτύξουμε το σχέδιο μετάφρασης MQL σε Lorel. Συγκεκριμένα, βοηθά αφενός στην κατασκευή του ισοδύναμου Lorel path για ένα δεδομένο path της MQL, αφετέρου δίνει λύση στο πρόβλημα μετάφρασης εκφράσεων των οποίων η σημασιολογική ανάλυση απαιτεί διαφορετική αντιμετώπιση ανάλογα με τον τύπο του κόμβου ή της ακμής που συμμετέχει σε αυτή. Στην επόμενη ενότητα θα έχουμε τη δυνατότητα να επεκταθούμε στη χρησιμότητα αλλά και αναγκαιότητα αυτής της μετατροπής.

Παρακάτω ακολουθεί ο αλγόριθμος με τον οποίο θα παράγουμε την κανονική μορφή του MOEM γράφου που μας δίνεται κάθε φορά. Σε αυτόν τον κανονικοποιημένο γράφο θα εφαρμόζουμε τον προηγούμενο μετασχηματισμό σε ισοδύναμο OEM, προκειμένου να υποβάλλουμε ερωτήσεις.

Αλγόριθμος κανονικοποίησης MOEM γράφου

- **Βήμα 1ο:** Αφαίρεσε από το γράφο όλες τις context ακμές οι οποίες καταλήγουν στον κόμβο από όπου ξεκινούν
- **Βήμα 2ο:** Για κάθε entity ακμή που καταλήγει σε context κόμβο, επανατοποθέτησε την ακμή ώστε να δείχνει σε έναν νέο πολυδιάστατο κόμβο και πρόσθεσε μία context ακμή με explicit context το universal context η οποία ξεκινά από τον νέο πολυδιάστατο κόμβο και δείχνει στον context κόμβο που έδειχνε αρχικά η entity ακμή.
- **Βήμα 3ο:** Για κάθε context ακμή (p, c, q), όπου q είναι πολυδιάστατος κόμβος, κάνε τα εξής:
    - Για κάθε (context εξορισμού) ακμή (q, c_i, r_i) που ξεκινά από τον κόμβο q, και εφόσον $r_i \neq p$ (προς αποφυγή κυκλικής αναφοράς στον κόμβο p), πρόσθεσε μία context ακμή (p, s_i, r_i), όπου s_i η context τομή μεταξύ c και c_i.
    - Αφού τελειώσει η παραπάνω διαδικασία, αφαίρεσε την context ακμή (p, c, q) και, αν δεν δείχνει καμία άλλη ακμή στον πολυδιάστατο κόμβο q, αφαίρεσέ τον επίσης, μαζί φυσικά με τις ακμές που αναχωρούν από αυτόν[2].

[2] Προφανώς, με το τέλος της διαδικασίας αυτής, όλοι οι πολυδιάστατοι κόμβοι στους οποίους δεν καταλήγει μία entity ακμή θα αφαιρεθούν από το γράφο

Ο παραπάνω μετασχηματισμός διατηρεί στο ακέραιο τη χρήσιμη πληροφορία του γράφου, ενώ σαν χρήσιμη παρενέργειά του προκύπτει η απάλειψη από τον γράφο προβληματικών σχηματισμών (ανωμαλίες) που δεν επιδρούν στην πληροφορία που είναι αποθηκευμένη σε αυτόν. Αυτοί είναι:

- ✓ Πολυδιάστατοι κόμβοι στους οποίους δε δείχνουν entity ακμές
- ✓ Context ακμές που καταλήγουν στον κόμβο από όπου ξεκινούν

Κατά την επιστροφή στον χρήστη των αποτελεσμάτων της ερώτησης, χρήσιμο είναι να απλοποιούμε το γράφο – απάντηση, ο οποίος βρίσκεται σε κανονική μορφή, απαλείφοντας από τον γράφο πολυδιάστατους κόμβους από όπου φεύγει μία μόνο ακμή με universal explicit context, ανακατευθύνοντας παράλληλα τις ακμές που φτάνουν σε αυτόν στον context κόμβο όπου φτάνει η μοναδική αυτή context ακμή.

3.2.3 Μετάφραση MQL ερωτήσεων στη Lorel

Με βάση το μοντέλο αναπαράστασης MOEM μέσω OEM που περιγράφηκε στα προηγούμενα, θα παρουσιάσουμε τώρα ένα σχήμα μετάφρασης μιας MQL ερώτησης στη Lorel ισοδύναμη που δίνει τα επιθυμητά αποτελέσματα στη μορφή OEM γράφου.

Μετάφραση απλών path expression

Θα θεωρήσουμε αρχικά ότι συναντούμε στο from clause ένα path expression της μορφής:

```
from [c1]a.b.c.[c2]d.e Q
```

Θα ήταν καλό από την αρχή να μετασχηματίσουμε την παραπάνω έκφραση ως εξής:

```
from  [c1]a X
      X.[c1]b Y,
      Y.[c1]c Z,
      Z.[c2]d W,
      W.[c2]e Q
```

δηλαδή να χωρίσουμε το path σε τμήματα που αποτελούνται μόνο από ένα label, οπότε αρκεί στη συνέχεια να μελετήσουμε τη μετάφραση path expression της μορφής

```
from  X.[c]l Y
```

όπου X είναι context node. Για το μετασχηματισμό αυτών των path μας ενδιαφέρει:

- ✓ να σχηματίσουμε από το path αυτό τρία Lorel path που να αποτιμούνται το ένα στον κόμβο (ή στους κόμβους) που δείχνει η ακμή l από τον κόμβο X στο OEM μοντέλο που θεωρήσαμε (στο εξής θα αναφερόμαστε στον κόμβο αυτό ως «κόμβο-entity edge»), το δεύτερο στον ή στους κόμβους που δείχνει η ακμή _cxt από τους multidimensional ισοδύναμους κόμβους («κόμβους-context edge») και το τρίτο στους context κόμβους που ικανοποιούν τις απαιτήσεις του path, χωρίς να λαμβάνουμε υπόψη τα ic. Οι πρώτες δύο κατηγορίες κόμβων μας παρέχουν πρόσβαση στα ec, ic των αντίστοιχων context και entity ακμών του MOEM.

- ✓ να ελέγξουμε αν ο context κόμβος που ακολουθεί και ενώνεται με τον κόμβο-context edge μέσω της ακμής _facet, πληροί τις προϋποθέσεις ώστε να αποτελεί επιστροφή του αρχικού MQL path. Πρέπει δηλαδή να γίνει σύγκριση του c με τα ic των κόμβων-entity edge και των κόμβων-context edge, ώστε να διαπιστώσουμε αν τα context στα οποία αντιστοιχούν είναι υπερσύνολα ή όχι του c. Ο έλεγχος αυτός θα γίνει σε ένα where clause που θα συνοδεύει το αντίστοιχο from clause, όπως θα γίνει κατανοητό από τα παρακάτω παραδείγματα.

Σύμφωνα λοιπόν με τα παραπάνω, θα έχουμε μια μετατροπή της μορφής:

```
from  X.l entityEdge, (πρώτο path)
      entityEdge._ett._facet contextEdge, (δεύτερο)
      contextEdge._cxt Y (τρίτο)
where entityEdge._icw{W1} = 'w1'
      and contextEdge._icw{W1} = 'w1'
      and entityEdge._icw{W2} = 'w2'
      and contextEdge._icw{W2} = 'w2' and ...
```

Στα παραπάνω w_n είναι οι κωδικοί των κόσμων που ανήκουν στο σύνολο που παράγεται από το [c]. Σημειώνουμε επίσης την ανάγκη ορισμού των ονομάτων των κόμβων Wn στους οποίους η Lorel θα αντιστοιχεί κάθε path που εμφανίζεται στο where clause, ώστε η αποτίμηση να γίνεται για διαφορετικούς κόμβους στον OEM γράφο.

Επιστρέφοντας στο αρχικό path expression, στην ερώτηση Lorel που τελικά θα παραχθεί με τον παραπάνω τρόπο, θα πρέπει να γίνεται μια παράθεση όλων των path που θα σχηματίζονται από την ερώτηση MQL στο from clause, ενώ οι εκφράσεις που θα παράγονται στο where clause θα προστίθενται σ' αυτό, αν υπάρχει ήδη στην ερώτηση MQL, αλλιώς θα πρέπει να προσθέσουμε ένα where clause ειδικά για την ερώτηση σε Lorel.

Παίρνουμε για παράδειγμα το εξής query σχετικά με τον γράφο recreation_guide:

```
select      restaurant: {name: X.name, winter_floor: Y}
from        recreation_guide.restaurant X,
            [season=winter]X.address.floor Y,
            [season=summer,daytime=noon]X.address.floor Z
where       Z="terrace"
```

Σε πρώτη φάση, θα πρέπει τα context specifier που εμφανίζονται στο παραπάνω query να μετατραπούν στη μορφή [w in{…}] όπου αναφερθήκαμε προηγουμένως. Στη συνέχεια θα απλοποιηθούν τα path expression σε απλά, όπως περιγράψαμε. Θέτοντας λοιπόν:

```
[c1]=[-], [c2]=[season=winter], [c3]=[season=summer, daytime=noon]
```

θα έχουμε:

```
select      restaurant: {name: X.name, winter_floor: Y}
from        [c1]recreation_guide X1,
            X1.[c1]restaurant X,
            X.[c2]address Y1,
            Y1.[c2]floor Y,
            X.[c3]address Z1,
            Z1.[c3]floor Z
where       Z=”terrace”
```

Στο παραπάνω query έχουμε path της μορφής για την οποία μιλήσαμε πιο πάνω, οπότε μπορούμε να λάβουμε το τελικό Lorel query, που θα είναι ως εξής:

```
select      restaurant: {name: X.name, winter_floor: Y}
from        recreation_guide._ett._facet._cxt X1,

            X1.restaurant XeE,
            XeE._ett._facet XcE,
            XcE._cxt X,

            X.address Y1eE,
            Y1eE._ett._facet Y1cE,
            Y1cE._cxt Y1,

            Y1.floor YeE,
            YeE._ett._facet YcE,
            YcE._cxt Y,

            X.address Z1eE,
            Z1eE._ett._facet Z1cE,
            Z1cE._cxt Z1,

            Z1.floor ZeE,
            ZeE._ett._facet ZcE,
            ZcE._cxt Z
where       Z = ”terrace”
            and έλεγχος XeE
            and έλεγχος XcE
            and έλεγχος Y1eE
            and έλεγχος Y1cE
            and έλεγχος YeE
            and έλεγχος YcE
            and έλεγχος Z1eE
            and έλεγχος Z1cE
            and έλεγχος ZeE
            and έλεγχος ZcE
```

Στα παραπάνω συμβολίζουμε ως *έλεγχος X* τις εκφράσεις που θα κάνουν έλεγχο για το αν το inherited context που είναι αποθηκευμένο στον κόμβο *X* είναι υπερσύνολο του δεδομένου context specifier που αντιστοιχεί στην ακμή της οποίας ο *X* αποτελεί τον ενδιάμεσο κόμβο-entity edge ή κόμβο-context edge. Όπως δόθηκε προηγουμένως, μια τέτοια έκφραση θα είναι της μορφής:

```
X._icw{W1} = 'w1'
and X._icw{W2} = 'w2'
and X._icw{W3} = 'w3'
and X._icw{W4} = 'w4' and ...
```

Είναι προφανές ότι αν σε κάποιο από τα label που συγκροτούν ένα path χρησιμοποιήσουμε το wildcard % που συναντάμε και στην Lorel, η αποτίμηση της έκφρασης θα τελεστεί χωρίς κανένα πρόβλημα αν ακολουθήσουμε την παραπάνω διαδικασία. Πρόβλημα όμως παρουσιάζεται στην υποστήριξη της έκφρασης #, η οποία είναι ισοδύναμη με την κανονική έκφραση (.%)*, όπως εξάλλου και στην υποστήριξη γενικότερα των regular expression σε ένα path. Με την παρούσα μεθοδολογία, κάτι τέτοιο είναι αδύνατο, γιατί δεν μπορούμε να δεσμεύουμε μεταβλητές για τους ενδιάμεσους κόμβους που συμμετέχουν σε ένα path που ορίζεται με αυτόν τον τρόπο. Συνεπώς, δεν μπορούμε να έχουμε στο where clause πρόσβαση στους ενδιάμεσους κόμβους-entity edge ή –context edge, οι οποίοι είναι απαραίτητοι προκειμένου να επαληθεύσουμε αν ικανοποιούνται οι context specifier του path expression.

Υποστήριξη πολυδιάστατων κόμβων σε path expression

Στα προηγούμενα αναφερθήκαμε σε path expression που έχουν αφετηρία context κόμβους και αποτιμώνται επίσης σε context κόμβους. Θα ασχοληθούμε τώρα με εκφράσεις που έχουν σχέση με multidimensional κόμβους, χρησιμοποιώντας ένα εμπλουτισμένο σύνολο στοιχείων που συγκροτούν ένα path.

Για να ικανοποιήσουμε την ανάγκη να μπορεί ένα path expression να αποτιμάται σε multidimensional κόμβο, θεωρήσαμε τους τελεστές < και >, οι οποίοι περικλείουν ένα συνηθισμένο path expression, οπότε θα έχουμε μια έκφραση της μορφής

```
<[c1]l1.l2.[c3]l3>
```

Το αποτέλεσμα μιας τέτοιας έκφρασης θα είναι το σύνολο των κόμβων στους οποίους καταλήγει η τελευταία ακμή του path, l3, και οι οποίοι φυσικά ικανοποιούν τα context specifier του path. Με τον τρόπο αυτό προφανώς δίνεται η δυνατότητα σε ένα context path expression να αποτιμηθεί και σε multidimensional κόμβους.

Προφανώς δεν μπορούμε γενικά να προβλέψουμε τι τύπου κόμβοι θα είναι το αποτέλεσμα μιας έκφρασης όπως η παραπάνω. Το γεγονός αυτό δημιουργεί ένα πρόβλημα όσον αφορά στην υλοποίηση των διάφορων path expression που έχουν σαν αφετηρία κόμβους που προκύπτουν από τέτοιες εκφράσεις. Η θεώρηση όμως ότι οι γράφοι που χειριζόμαστε στο σύστημά μας βρίσκονται στην κανονική μορφή, μας εξασφαλίζει ότι, κατά τον χειρισμό των MQL ερωτήσεων στο σύστημά μας, κόμβοι που προκύπτουν από αυτές τις εκφράσεις θα είναι πάντα multidimensional. Συνεπώς, έχουμε τη δυνατότητα να γνωρίζουμε τον τύπο κόμβου που δεσμεύει κάθε μεταβλητή που ορίζεται σε ένα MQL query και στη συνέχεια να χρησιμοποιήσουμε τις κατάλληλες εκφράσεις για την υλοποίηση των path expression στα οποία αυτή μπορεί να συμμετέχει.

Με την εισαγωγή της παραπάνω έκφρασης εμφανίζεται η ανάγκη για εκφράσεις που θα μπορούν να ελέγχουν πώς θα συνεχίζεται η διάσχιση ενός MOEM γράφου πέρα από τους multidimensional κόμβους που τυχόν εμφανίζονται σε ένα path expression της παραπάνω μορφής. Αυτές θα είναι οι:

- `X[ic]`
- `X:[ec]`

Η πρώτη έκφραση μας δίνει τους context κόμβους όπου δείχνουν εκείνες οι context ακμές του X, το inherited context των οποίων είναι υπερσύνολο του [ic].[3] Η δεύτερη έχει παρόμοιο αποτέλεσμα, μόνο που κάνει τον ίδιο έλεγχο για το explicit context κάθε ακμής. Σε περίπτωση που το X παριστάνει context κόμβο, η λειτουργία των παραπάνω τελεστών προφανώς αγνοείται και επιστρέφεται ο ίδιος κόμβος.

Έστω τώρα έκφραση της μορφής

```
from  X.[c]l Y
```

όπου X multidimensional κόμβος. Με βάση όσα έχουμε πει ως τώρα, δεν είναι σαφές πώς θα χειριστεί το σύστημα την παραπάνω έκφραση, δηλαδή πώς από τον πολυδιάστατο κόμβο θα διασχιστεί ο γράφος μέχρι έναν context κόμβο – παιδί του, ώστε από αυτόν να συνεχιστεί η αποτίμηση του path expression. Για να ξεπεράσουμε το πρόβλημα, θεωρήσαμε ότι η παραπάνω έκφραση είναι ισοδύναμη με την εξής:

```
from  X:[].[c]l Y
```

[3] Για μη κανονικοποιημένους γράφους, προβλέπεται ότι στην περίπτωση που άμεσο παιδί του multidimensional κόμβου X είναι επίσης ένας multidimensional κόμβος, συνεχίζεται με τον ίδιο τρόπο η διάσχιση του γράφου, μέχρι να φτάσει στον πρώτο context κόμβο. Θεωρώντας έναν τέτοιο γράφο, η παρούσα προσέγγιση θα δώσει το ίδιο ακριβώς αποτέλεσμα αν εφαρμοστεί στην κανονική μορφή του, λόγω του τρόπου που αυτή παράγεται

Η παραπάνω θεώρηση απορρέει από τη σκέψη ότι, προκειμένου να χρησιμοποιήσουμε έναν πολυδιάστατο κόμβο X στη θέση ενός context κόμβου, θα πρέπει να θεωρήσουμε ότι ο X ισοδυναμεί με τους context κόμβους – παιδιά του, οι οποίοι έχουν νόημα σε όλους τους κόσμους όπου έχει νόημα και ο X, ή πιο αυστηρά[4], που θα εμφανίζονται σε όλους τους δυνατούς reduced MOEM γράφους που περιέχουν τον κόμβο X. Ο ακριβής υπολογισμός των ισοδύναμων context κόμβων απαιτεί τον υπολογισμό του inherited context του X καθώς και όλων των context κόμβων όπου καταλήγει η ακμή l.

Χάριν ευκολίας της υλοποίησης, θεωρούμε ως ισοδύναμους αυτούς τους κόμβους για τους οποίους ισχύει ότι η context ακμή που τους συνδέει με τον X έχει explicit context το universal. Αυτή η θεώρηση θα μας δώσει κόμβους που είναι ισοδύναμοι του X σε κάθε περίπτωση, ίσως όμως οδηγήσει σε εξαίρεση κάποιων κόμβων που είναι ισοδύναμοι, αλλά η context ακμή που τους συνδέει δεν έχει universal explicit context. Αυτό όμως συμβαίνει εξαιρετικά σπάνια, οπότε εκτιμούμε ότι μια αρκετά καλή προσέγγιση του ζητήματος.

Για την μετάφραση της προηγούμενης έκφρασης χρειάζεται να γνωρίζουμε αν ο X είναι context ή multidimensional κόμβος. Επειδή ο γράφος που θα χρησιμοποιήσουμε για την υλοποίηση προκύπτει με κανονικοποίηση του αρχικού, είναι δυνατόν το σύστημα να γνωρίζει σε ποιο τύπο κόμβου αναφέρεται κάθε μεταβλητή. Η υλοποίηση συνεπώς θα πρέπει να χρησιμοποιεί τον κατάλληλο μετασχηματισμό ανάλογα με την περίπτωση.

Μπορούμε τώρα να ασχοληθούμε με την υλοποίηση των παραπάνω. Αρχικά απλοποιούμε αναδρομικά κάθε path της μορφής

```
<[c1]l1.l2.[c3]l3>
```

στο ισοδύναμό του:

```
[c1]l1.l2.<[c3]l3>
```

Οπότε αρκεί να μετασχηματίσουμε σε Lorel εκφράσεις της μορφής

```
from  X.<[c]l> Y
```

Εφαρμόζοντας την ίδια λογική με πιο πάνω, η παραπάνω έκφραση με X context κόμβο γίνεται

```
from  X.l entityEdge,
      entityEdge._ett Y
where έλεγχος entityEdge
```

[4] Δεν αρκεί να απαιτήσουμε το inherited context του παιδιού να είναι υπερσύνολο του ih του X, γιατί υπάρχει περίπτωση αυτό να συμβαίνει, αλλά να μην είναι δυνατή η ισοδυναμία μεταξύ των δύο. Συγκεκριμένα, αν μια context ακμή από άλλον multidimensional κόμβο καταλήγει στο ίδιο context κόμβο, τότε μπορεί να ισχύει το παραπάνω, ακόμα και αν η ακμή από τον X δεν έχει explicit context το []. Τότε όμως ο context κόμβος δεν θα εμφανίζεται σε όλους τους δυνατούς reduced MOEM γράφους που περιέχουν τον X και όχι τον άλλο πολυδιάστατο κόμβο.

Όσον αφορά τώρα στην υλοποίηση μιας έκφρασης της μορφής “from X[ic] Y”, αυτή μπορεί να γίνει ως εξής:

```
from  X._facet contextEdge,
      contextEdge._cxt Y
where έλεγχος contextEdge
```

Ο μετασχηματισμός μιας έκφρασης της μορφής “from X:[ec] Y” θα γίνεται κατά παρόμοιο τρόπο:

```
from  X._facet contextEdge,
      contextEdge._cxt Y
where έλεγχος contextEdge
```

Αν το X είναι context κόμβος, θα αγνοήσουμε τους τελεστές [ic] ή :[ec] μετά από το X και θα συνεχίσουμε κανονικά την αποτίμηση του path expression.

Πρέπει τέλος να προσθέσουμε ότι σε κάθε περίπτωση όπου έχουμε path expression που έχει σαν αφετηρία κόμβο X για τον οποίο γνωρίζουμε από την κανονικότητα του γράφου ότι είναι multidimensional, θα πρέπει να θεωρήσουμε έναν μετασχηματισμό της MQL ερώτησης που υλοποιεί την αποτίμηση του X στους ισοδύναμους context κόμβους τους, όπως αναλύθηκε πιο πάνω. Θα πρέπει λοιπόν να αντικατασταθεί μια έκφραση της μορφής

```
from  X.[c1]l1 X1
```

με την εξής:

```
from  X:[] castedX,
      castedX.[c1]l1 X1
```

Υποστήριξη path variable και του within clause

Συνεχίζουμε την επέκταση της γενικής μορφής ενός context path expression εισάγοντας μία ειδική κατηγορία μεταβλητών, τις context variable. Η MQL υποστηρίζει τη δυνατότητα να δεσμεύεται το inherited context[5] ενός path expression ή τμήματός του σε οριζόμενες από το χρήστη μεταβλητές, οι οποίες εμφανίζονται σε ένα expression μεταξύ δύο αγκύλων (bracket), για παράδειγμα μία μεταβλητή [P]. Αυτές είναι δυνατόν να χρησιμοποιηθούν, αφού οριστούν στο from clause, σε ένα clause ειδικά σχεδιασμένο για την MQL, το within clause.

Η παρουσία του within clause σε ένα MQL query έχει παρόμοιο ρόλο όσον αφορά στα context variable, με το ρόλο που έχει το where clause όσον αφορά στα object variables, συνιστά δηλαδή ένα είδος «φίλτρου» που θέτει κάποιες συνθήκες που πρέπει αυτά να ικανοποιούν, προκειμένου να επιστραφεί το αποτέλεσμα ενός context path expression που περιέχει context variable.

[5] Path Inherited Context: Η τομή των inherited context όλων των ακμών που συμμετέχουν στο σχηματισμό ενός path.

Ας θεωρήσουμε το παρακάτω παράδειγμα:

```
select      Y
from        recreation_guide.music_club.[X]review.score Y
within      [X]*[w in {1, 3, 4, 5}] <= [w in {1, 3}]
```

Στο παραπάνω παράδειγμα ορίζεται η context μεταβλητή X, η οποία αποτιμάται στο inherited context του path "review.score", και στη συνέχεια ελέγχεται αν η τομή της με το [w in {1, 3, 4, 5}] είναι υποσύνολο του [w in {1, 3}]. Το query θα επιστρέψει τελικά αυτά τα Y, για τα οποία το αντίστοιχο path inherited context ικανοποιεί την πιο πάνω συνθήκη.

Προκειμένου να υλοποιηθεί η μετάφραση μιας τέτοιας ερώτησης, θα πρέπει να βρούμε εκφράσεις σε γλώσσα Lorel, οι οποίες θα μπορούν να εκτελέσουν τις διάφορες πράξεις καθώς και συγκρίσεις μεταξύ των συνόλων context που εμφανίζονται σε ένα within clause. Η Lorel διαθέτει τελεστές που εκτελούν αυτές τις λειτουργίες μεταξύ συνόλων που αποτελούνται από OEM κόμβους. Χρειάζεται συνεπώς να εκφράσουμε τα διάφορα σύνολα που συμμετέχουν στις within εκφράσεις σε σύνολα κόμβων του OEM γράφου μας.

Μια τέτοια έκφραση για το path inherited context που αντιστοιχεί σε μια context μεταβλητή μπορεί να προκύψει από μια σχετικά απλή διαδικασία: Θεωρούμε ότι κάθε context μεταβλητή παριστάνεται στο σύστημά μας συνοδευόμενη από τις ενδιάμεσες μεταβλητές που προκύπτουν από διαδικασία που περιγράφηκε πιο πάνω, οι οποίες αντιστοιχούν σε context edge ή entity edge κόμβους, και οι οποίες περιλαμβάνονται στο path το inherited context του οποίου θέλουμε να εκφράσουμε.

Επανερχόμενοι στο προηγούμενο παράδειγμα, θα έχουμε δηλαδή μια έκφραση του τύπου (έστω MusicClub η μεταβλητή όπου αποτιμάται το MQL path "[]recreation_guide.[]music_club"):

```
from        MusicClub.review Y1eE,
            Y1eE._ett._facet Y1cE,
            Y2cE._cxt Y1,
            Y1.score YeE,
            YeE._ett._facet YcE,
            YcE._cxt Y
```

Η context μεταβλητή [X] θα συσχετίζεται συνεπώς με τις μεταβλητές Y1eE, Y1cE, YeE και YcE. Μπορούμε πια να εκφράσουμε το περιεχόμενο της [X], δηλαδή το path inherited context του "review.score" ως εξής:

```
Xpic[6] =     (     Y1eE._icw intersect Y1cE._icw
                    intersect YeE._icw intersect YcE._icw
              )
```

Με παρόμοιο τρόπο μπορούμε προφανώς να εκφράσουμε το context που δεσμεύεται σε οποιαδήποτε μεταβλητή[7].

[6] Δίνουμε όνομα στην έκφραση αυτή, για να μην την επαναλάβουμε πιο κάτω.

Σε μια συνθήκη του within clause θα συναντήσουμε όμως εκτός από context μεταβλητές, και context specifier που θα ορίζονται απευθείας από τον χρήστη, όπως φαίνονται και στο παράδειγμα. Επειδή κατευθυνόμαστε προς μια λύση που θα χρησιμοποιεί τις πράξεις σε σύνολα που υποστηρίζει η Lorel, θα πρέπει να βρούμε μια έκφραση της Lorel όπως και πιο πάνω, στην οποία θα περιέχονται οι αντίστοιχοι κόμβοι του OEM γράφου. Ας θεωρήσουμε την έκφραση %._icw, δηλαδή το σύνολο των κόσμων που ανήκουν στο inherited context της context ακμής που δείχνει στη ρίζα του γράφου. Αυτό είναι προφανώς το universal context του γράφου, όποια μορφή κι αν έχει ακολούθως. Με τη βοήθεια της έκφρασης αυτής μπορούμε να εκφράσουμε ένα σύνολο *W* που θα περιέχει όλους τους κόσμους που ανήκουν στο εκάστοτε context specifier.

Θεωρώντας για παράδειγμα την έκφραση [w in {1, 3, 4, 5}] του παραδείγματος θα έχουμε την εξής έκφραση:

```
W1 =  (     select      W
            from        %._icw W
            where       W = '1' or W = '3'
                        or W = '4' or W = '5'
      )
```

Και για την έκφραση [w in {1, 3}] την εξής:

```
W2 =  (     select      W
            from        %._icw W
            where       W = '1' or W = '3'
      )
```

Έχουμε λοιπόν ως τώρα καταφέρει να εκφράσουμε με τη βοήθεια της Lorel όλα τα σύνολα που θα υπεισέρχονται σε μια συνθήκη του within clause. Με παρόμοιο τρόπο μπορούμε να εκτελέσουμε πράξεις μεταξύ τους, όπως αυτές θα καθορίζονται από την εκάστοτε συνθήκη.

Για το παράδειγμα που αναφέραμε στην αρχή, μπορούμε να εκφράσουμε τα σύνολα για το αριστερό και δεξιό μέλος της συνθήκης ως εξής (αντικαθιστούμε γενικά τα ονόματα με πλάγια γράμματα με τις εκφράσεις που ορίσαμε πιο πάνω):

```
left  = ( Xpic intersect W1),
right =   W2
```

Με τη βοήθεια των τελεστών της Lorel "union" και "except" μπορούμε εκτός της τομής (MQL τελεστής *) να εκφράσουμε την ένωση (MQL τελεστής +) και την αφαίρεση (MQL τελεστής -) συνόλων. Είναι συνεπώς προφανές ότι με αυτόν τον τρόπο είναι εξαιρετικά εύκολο να μεταφερθεί μια οποιαδήποτε παράσταση πράξεων μεταξύ συνόλων σε Lorel μορφή.

---

[7] Μπορούμε να χρησιμοποιήσουμε μεταβλητές που ελέγχουν το explicit context μιας (μόνο) ακμής, όπως στην έκφραση X:[P], όπου η μεταβλητή P δεσμεύεται στο explicit context της ακμής που ακολουθεί το X. Σε αυτήν την περίπτωση προφανώς αντί για _icw ακμές θα έχουμε _ecw.

Αυτό που μένει είναι να εκφράσουμε τους τελεστές σύγκρισης μεταξύ συνόλων, δηλαδή τις σχέσεις υποσυνόλου και υπερσυνόλου, ισότητας και διαφοράς, ή ακόμα και γνησίου υποσυνόλου και υπερσυνόλου. Μπορούμε να εκφράσουμε μέσω Lorel έναν έλεγχο υποσύνόλου της μορφής *[c_left]* <= *[c_right]* όπως παρακάτω:

```
for all Vleft in ( left ):
      exists Vright in ( right ):
            Vleft = Vright
```

όπου *left* και *right* είναι εκφράσεις που προέκυψαν όπως προηγουμένως από τις αντίστοιχες εκφράσεις *[c_left]* και *[c_right]* που στη συγκεκριμένη περίπτωση παριστάνουν γενικά πράξεις μεταξύ context μεταβλητών και specifier.

Για το παράδειγμα με το οποίο ασχολούμαστε θα προέκυπτε τελικά ένα Lorel query που στο where clause του θα εμφανιζόταν, συζευγμένη με τυχόν άλλες συνθήκες η έκφραση:

```
for all V1 in ( Xpic intersect W1):
      exists V2 in ( W2 ):
            V1 = V2
```

Χρησιμοποιώντας τα συμβολικά ονόματα *left* και *right* για τα δύο μέλη μιας έκφρασης σύγκρισης συνόλων, μπορούμε να γράψουμε αντίστοιχες εκφράσεις και για τους υπόλοιπους τελεστές σύγκρισης. Μάλιστα αρκεί που αναφερθήκαμε μόνο στον έλεγχο υποσυνόλου, αφού από αυτόν και από τους λογικούς τελεστές "not" ($\neg$) και "and" ($\wedge$) μπορούμε να εκφράσουμε όλους τους υπόλοιπους τελεστές, αφού ισχύουν τα εξής:

1. $A \supseteq B \Leftrightarrow B \subseteq A$
2. $A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$
3. $A \neq B \Leftrightarrow \neg(A \subseteq B \wedge B \subseteq A)$
4. $A \subset B \Leftrightarrow A \subseteq B \wedge (\neg B \subseteq A)$
5. $A \supset B \Leftrightarrow B \subseteq A \wedge (\neg A \subseteq B)$

Ειδική μέριμνα θα πρέπει να προβλέψουμε για τον χειρισμό εκφράσεων που περιλαμβάνουν σαν context specifier την έκφραση του empty context [-] ή μια έκφραση που παράγει το empty context, οι οποίες θα πρέπει να εκφραστούν λίγο διαφορετικά. Περιοριζόμαστε και πάλι στην έκφραση της συνθήκης υποσυνόλου, από την οποία προκύπτουν και οι υπόλοιπες, οπότε έχουμε, για μια έκφραση της μορφής *[c_left]* <= *[c_right],* τις εξής περιπτώσεις:

1. αν το *[c_left]* είναι κενό, η συνθήκη είναι πάντα αληθής, αφού το κενό σύνολο είναι υποσύνολο κάθε συνόλου, ακόμα και του κενού.
2. αν το *[c_right]* είναι κενό, η συνθήκη είναι αληθής μόνο αν και το *[c_left]* είναι κενό. Θα πρέπει συνεπώς να μεταφέρουμε στο Lorel ερώτημα την έκφραση "not exists(*right*)" αντί της κανονικής έκφρασης

Για να γίνουν τα προηγούμενα λίγο πιο ξεκάθαρα, ας θεωρήσουμε ένα δεύτερο παράδειγμα μιας έκφρασης στο within clause:

```
[P]*[w in {1, 4, 5, 6}] != [-]
```

Η παραπάνω έκφραση, σύμφωνα με τα προηγούμενα, θα είναι ισοδύναμη με την εξής:

```
not (     [P]*[w in {1, 4, 5, 6}] <= [-]
      and [-] <= [P]*[w in {1, 4, 5, 6}]
    )
```

Θέτοντας *P* την έκφραση σε Lorel του path inherited context που εκφράζει η μεταβλητή [P], θα έχουμε τελικά την ακόλουθη Lorel έκφραση:

```
not (     not exists( P intersect
                      ( select W from %._icw
                        where W = '1' or W = '4'
                              or W = '5' or W = '6'
                      )
                    )
    )
      and true
    )
```

Αν απλοποιήσουμε την προηγούμενη έκφραση προκύπτει η εξής:

```
exists( P intersect
          ( select W from %._icw
            where W = '1' or W = '4'
                  or W = '5' or W = '6'
          )
      )
```

Η παραπάνω έκφραση αποτελεί και τον πιο άμεσο και κομψό τρόπο να εκφραστεί η αρχική συνθήκη. Η πολυπλοκότητα στην τελική έκφραση που προκύπτει είναι το τίμημα της προσέγγισης στο ζήτημα της μετάφρασης με τρόπο που μπορεί σχετικά άμεσα να υλοποιηθεί σε κάποιο πρόγραμμα.

Ανάλογα με το πόσο σημαντικό για μια υλοποίηση είναι οι εκφράσεις Lorel που προκύπτουν να είναι όσο το δυνατόν απλούστερες μπορεί να ακολουθηθεί μια διαφορετική πολιτική επεξεργασίας τους, η οποία πιθανόν να μην χρησιμοποιεί μόνο την έκφραση για το υποσύνολο, όπως περιγράψαμε πιο πάνω, να χειρίζεται με ειδικό τρόπο και άλλους τελεστές σύγκρισης συνόλων, ούτως ώστε να προκύπτουν κομψότερες εκφράσεις.

4

Υλοποίηση

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Προκειμένου να καταλήξουμε όσον αφορά στο σύστημα επερωτήσεων μέσω του οποίου θα ήταν καλύτερο να υλοποιηθεί η γλώσσα MQL, έγινε αρχικά μια διερεύνηση των δυνατοτήτων γνωστών συστημάτων. Ένα από τα συστήματα που μελετήθηκαν για αυτό το σκοπό ήταν και το AceDb. Το σύστημα αυτό παρέχει ένα μοντέλο οργάνωσης δεδομένων που προσεγγίζει στη μορφή του τις SSD εκφράσεις. Χρησιμοποιείται ευρύτατα για την οργάνωση και αποθήκευση δεδομένων που έχουν να κάνουν με βιολογική έρευνα και πιο συγκεκριμένα για αποθήκευση πληροφοριών για τα γονίδια διαφόρων οργανισμών. Επίσης διαθέτει και μια γλώσσα επερωτήσεων που στηρίζεται σε εκφράσεις μονοπατιών. Το συγκεκριμένο σύστημα είναι πολύ σταθερό και αξιόπιστο αφού χρησιμοποιείται εδώ και χρόνια σε επαγγελματικό επίπεδο.

Παρά τις ομοιότητες με το OEM μοντέλο όμως, που αποτελεί βάση για το MOEM, το AceDB χαρακτηρίζεται από μια σειρά από ιδιαιτερότητες που δυσχεραίνουν τη μεταφορά του μοντέλου MOEM σε αυτό, αλλά και τη μετάφραση MQL ερωτήσεων στη γλώσσα που αυτό υποστηρίζει. Για αυτό το λόγο προτιμήσαμε για την υλοποίηση το σύστημα Lore, που υποστηρίζει τόσο το OEM μοντέλο, όσο και τη Lorel, στα οποία η μεταφορά του MOEM και της MQL αντίστοιχα είναι πολύ πιο άμεση. Αν και το AceDB είναι πολύ πιο σταθερό και αξιόπιστο από το Lore, η υλοποίησή του κρίθηκε αρκετά ικανοποιητική για τους σκοπούς που αυτό θα χρησιμοποιηθεί.

4.1.1 Το σύστημα Lore

Το σύστημα Lore αναπτύχθηκε σε ερευνητικό επίπεδο στο πανεπιστήμιο του Stanford ως το πρώτο αυτόνομο και πλήρες σύστημα διαχείρισης βάσεων των οποίων τα δεδομένα είναι ημιδομημένα. Το συγκεκριμένο σύστημα δημιουργεί βάσεις δεδομένων από αρχεία με SSD εκφράσεις (αρχεία OEM), καθώς και από αρχεία XML. Περιλαμβάνει ένα πλήρες σύστημα διαχείρισης των συγκεκριμένων βάσεων, διάφορες εφαρμογές για τη συντήρηση και την ανανέωση αυτών καθώς και ένα υποσύστημα υποβολής επερωτήσεων. Η γλώσσα επερωτήσεων που χρησιμοποιεί το υποσύστημα αυτό είναι η Lorel, τόσο για τη σύνταξη απλών ερωτήσεων που επιστρέφουν κάποια δεδομένα όσο και για τη σύνταξη ερωτήσεων δημιουργίας, διαγραφής κόμβων και ακμών, αλλαγής των δεδομένων κ.ο.κ.

Πέρα από το ίδιο το σύστημα έχει αναπτυχθεί και μια προγραμματιστική διαπροσωπεία της εφαρμογής (API), έτσι ώστε να υπάρχει η δυνατότητα ανάπτυξης εφαρμογών που συνδέονται σε κάποια βάση, έχουν πρόσβαση στα δεδομένα και μπορούν και να υποβάλλουν ερωτήσεις. Το συγκεκριμένο API έχει υλοποιηθεί σε C++ σε περιβάλλον Linux, όπου εξάλλου έχει υλοποιηθεί και ολόκληρο το σύστημα Lore.

Μέσω του συγκεκριμένου API μπορεί να γίνει ο έλεγχος δεδομένων που υπάρχουν στο σύστημα είτε εκφρασμένα σε SSD expressions είτε σε XML μορφή. Η δεύτερη περίπτωση δεν αφορά άμεσα την παρούσα εργασία και γι' αυτό δεν θα αναπτυχθεί περαιτέρω. Για την περίπτωση τώρα των βάσεων που περιέχουν δεδομένα σε μορφή SSD εκφράσεων, οι κυριότερες δυνατότητες που παρέχονται και τις οποίες αξιοποιήσαμε κατά την υλοποίηση είναι οι εξής:

- *connect():* Σύνδεση του προγράμματος με κάποια βάση του Lore.
- *disconnect():* Αποσύνδεση του προγράμματος από κάποια βάση του Lore.
- *submit ():* Υποβολή Lorel ερώτησης στη συγκεκριμένη βάση που έχουμε συνδεθεί.
- *LoreOem:* Αντικείμενο που υπάρχει στη μνήμη και περιέχει έναν OEM γράφο. Αυτός μπορεί να είναι ο αρχικός αλλά μπορεί και να είναι και η απάντηση σε κάποια ερώτηση που έχουμε υποβάλλει. Επίσης κάθε τέτοιο αντικείμενο μπορεί να είναι complex ή atomic, ανάλογα με το εάν έχει παιδιά ή απλά μια ατομική τιμή.
- *LoreOid ():* Επιστρέφει το αναγνωριστικό ενός κόμβου του OEM γράφου.
- *Value ():* Συνάρτηση που επιστρέφει το τιμή κάποιου ατομικού κόμβου του LoreOem αντικειμένου.
- *LoreIterator*(): Αντικείμενο που δημιουργεί μια διάταξη με τα παιδιά κάποιου complex LoreOem αντικειμένου.
- *Next*(): Μέθοδος της κλάσης LoreIterator, η οποία όποτε καλείται επιστρέφει ένα LoreOem αντικείμενο με ρίζα το επόμενο παιδί του complex αντικειμένου με βάση το οποίο έχουμε δημιουργήσει τον συγκεκριμένο iterator.
- *Label*(): Επιστρέφεται η ετικέτα κάποιας συγκεκριμένης ακμής.

- *LabelKey*(): Συνάρτηση που δίνει πρόσβαση στον μοναδικό αριθμό το οποίο δίνει το σύστημα σε κάθε ακμή.
- *MatchedPath*: Επιπρόσθετο αντικείμενο στη δομή που επιστρέφεται από την υποβολή μιας ερώτησης το οποίο για κάθε κόμβο της απάντησης προσδιορίζει μέσω των id των ακμών το μονοπάτι του αρχικού γράφου το οποίο από τη ρίζα καταλήγει στον συγκεκριμένο κόμβο.

Κάτι που δεν υπάρχει στο συγκεκριμένο API και θα ήταν αρκετά χρήσιμο θα ήταν η δυνατότητα διαχείρισης των βάσεων προγραμματιστικά. Έτσι η δημιουργία, η επισκευή και η καταστροφή των βάσεων έγινε υποχρεωτικά με την κλήση εξωτερικών προγραμμάτων που υπάρχουν στο Lore (dbcreate, dbrepair, dbdestroy).

4.1.2 Το περιβάλλον εργασίας MSSDesigner

Ένα ήδη ανεπτυγμένο σύστημα διαχείρισης MOEM γράφων είναι ο MSSDesigner. Η συγκεκριμένη εφαρμογή έχει τη δυνατότητα δημιουργίας νέων γράφων και πλήρους καθορισμού όλων των παραμέτρων που έχουν αναπτυχθεί παραπάνω και οι οποίες ορίζουν πλήρως τόσο όλους τους τύπους ακμών αλλά και κόμβων που ορίζονται από το μοντέλο των πολυδιάστατων ημιδομημένων δεδομένων.

Περαιτέρω, πάνω σε κάποιο ήδη ορισμένο γράφο μπορούν να γίνουν οι διάφορες λειτουργίες reduction σε OEM γράφους αλλά και υπολογισμού του inherited context αυτού και ελέγχου της ορθότητάς του. Τέλος, εκτός από άλλες λειτουργίες που δεν είναι απαραίτητες στα πλαίσια της παρούσας εργασίας η συγκεκριμένη εφαρμογή μπορεί να γίνει φόρτωμα ενός MOEM γράφου από αρχείο που περιγράφει ένα γράφο μέσω MSSD εκφράσεων, καθώς και παραγωγή αρχείου με MSSD εκφράσεις από κάποιο MOEM γράφο.

Με βάση αυτή την εφαρμογή, έχουμε τη δυνατότητα να την επεκτείνουμε ώστε να συμπεριληφθεί σε αυτή το σύστημα επερωτήσεων σε γλώσσα MQL, το οποίο θα αξιοποιεί το Lore. Το σκεπτικό είναι να δημιουργηθεί μια ειδική διαπροσωπεία και ένα ειδικό μενού στο MSSDesigner, έτσι ώστε να μπορεί να αρχικοποιηθεί το υποσύστημα επερωτήσεων, να υποβάλλονται οι διάφορες ερωτήσεις σε MQL και να παρουσιάζονται τα αποτελέσματα με γραφικό τρόπο χρησιμοποιώντας τους ήδη υπάρχοντες μηχανισμούς της εφαρμογής.

4.2 Βασικές δομές και αρχεία

Πριν αναλύσουμε τον τρόπο με τον οποίο υλοποιήθηκαν στο υπό ανάπτυξη σύστημα οι διάφορες λειτουργίες που αυτό επιτελεί, είναι χρήσιμο να αναφερθούμε συνοπτικά στις δομές δεδομένων, με την βοήθεια των οποίων παριστάνουμε και διαχειριζόμαστε σε αυτό MOEM γράφους. Επιπλέον, στα επόμενα θα αναφερθούμε και στα αρχεία αποθήκευσης γράφων που θα χρησιμοποιήσουμε κατά την υλοποίηση.

4.2.1 Δομές αποθήκευσης γράφων

Για την παράσταση των γράφων χρησιμοποιήσαμε την κλάση MOEMGraph στο πακέτο moemgraph, η οποία έχει ήδη αναπτυχθεί για το πρόγραμμα MSSD Designer. Στα πλαίσια αυτής κάθε γράφος παριστάνεται με τη βοήθεια των πινάκων m_vEdges και m_vNodes, στους οποίους αποθηκεύονται αντίστοιχα οι ακμές και οι κόμβοι που τον συγκροτούν. Αντικείμενα των κλάσεων Edge και Node αποτελούν αναπαραστάσεις των ακμών και των κόμβων αντίστοιχα. Η κλάση αυτή έχει τη δυνατότητα να αναπαριστά, εκτός από MOEM γράφους, και OEM, αφού όπως έχουμε ήδη πει ένας OEM γράφος μπορεί να θεωρηθεί ως MOEM χωρίς πολυδιάστατους κόμβους, και συνεπώς χωρίς context ακμές.

Για τις ανάγκες της εφαρμογής μας υλοποιήθηκαν επιπλέον μέθοδοι στις παραπάνω κλάσεις, οι οποίες έχουν κυρίως βοηθητικό ρόλο για την ευκολότερη υλοποίηση των λειτουργιών μετατροπής των γράφων, στις οποίες αναφερθήκαμε στο 3^{ο} κεφάλαιο και στις οποίες θα αναφερθούμε αναλυτικότερα στα επόμενα.

4.2.2 Μορφές αρχείων που παράγονται

Στο σημείο αυτό πρέπει να αναφερθούμε σε δύο μορφές αρχείων περιγραφής γράφων που υλοποιήσαμε στα πλαίσια της εργασίας και με τις οποίες επιτυγχάνεται η μεταφορά των γράφων OEM που παριστάνουν MOEM από την εφαρμογή στο σύστημα του Lore, καθώς και η επιστροφή των αποτελεσμάτων της ερώτησης από το Lore στην εφαρμογή μας.

Η υποστήριξη εισαγωγής και εξαγωγής αυτών των αρχείων στο MSSD Designer υλοποιείται με τη βοήθεια μεθόδων της κλάσης MOEMFile του πακέτου moemfile. Ο πρώτος τύπος αρχείων που υποστηρίζεται για την ανάγκη της εφαρμογής είναι τα αρχεία *.oem, τα οποία υποστηρίζονται από το σύστημα Lore, το οποίο καθορίζει και τη μορφή τους. Ο δεύτερος τύπος είναι ένα XML αρχείο που έχει σχεδιαστεί ειδικά για την αναπαράσταση όχι μόνο OEM, αλλά και MOEM γράφων. Χρησιμοποιούμε για αυτά την επέκταση mox.

Αρχεία .oem

Τα αρχεία αυτά υποστηρίζονται από το Lore προκειμένου να φορτώνεται μαζικά στο σύστημα του Lore ένας OEM γράφος (bulk loading). Στην εφαρμογή μας με τη βοήθεια της μεθόδου saveAsSsd() έχουμε τη δυνατότητα δημιουργίας αυτών των αρχείων, μόνο για OEM γράφους όμως. Τα αρχεία αυτά έχουν τη μορφή μιας SSD expression, στην οποία αναφερθήκαμε στην Εισαγωγή και η οποία έχει ομοιότητες με την MSSD expression.

Αρχεία .mox

Τα αρχεία αυτά παράγονται από την πλευρά του server του συστήματος κατά την παραλαβή των αποτελεσμάτων μιας ερώτησης που υποβάλλεται σε αυτό. Η εφαρμογή του client έχει τη δυνατότητα όχι μόνο να διαβάζει αυτά τα αρχεία και να παρουσιάζει τον γράφο που περιγράφουν στο γραφικό περιβάλλον, αλλά, για λόγους πληρότητας, και να τα παράγει από γράφο που βρίσκεται στη μνήμη. Επιλέξαμε τη μορφή XML γιατί είναι η πλέον διαδεδομένη μορφή αρχείων για περιγραφή ημιδομημένων δεδομένων.

Κάθε κόμβος του MOEM γράφου παριστάνεται με τη βοήθεια tag, του οποίου το όνομα είναι το όνομα μιας ακμής (της πρώτης καθώς διασχίζουμε με κάποιο τρόπο το γράφο) που καταλήγει στον κόμβο. Για να περιγράψουμε γράφους στους κόμβους των οποίων καταλήγουν περισσότερες της μίας ακμές, υποστηρίζουμε τη δυνατότητα να δίνεται παράλληλα και ένας ακέραιος που τον περιγράφει μοναδικά τον κόμβο. Αυτό το αναγνωριστικό συμπεριλαμβάνεται στα attributes του αντίστοιχου tag, και το όνομα του attribute που το αποθηκεύει έχει οριστεί ως "id". Μια ακμή που καταλήγει σε κόμβο ο οποίος έχει ήδη καταγραφεί στο αρχείο είναι ένα empty tag με όνομα το όνομα της ακμής και attribute το "ref", του οποίου η τιμή είναι το αναγνωριστικό του αντίστοιχου κόμβου.

Η τιμή κάθε τέτοιου tag είναι είτε μια συμβολοσειρά, η οποία αντιστοιχεί στην τιμή του κόμβου, αν αυτός είναι atomic, είτε μια σειρά από φωλιασμένα tag, αν ο κόμβος είναι complex ή πολυδιάστατος. Για ευκολία στην ανάγνωση του αρχείου, μπορούμε προαιρετικά να καθορίσουμε τον τύπο του κόμβου τον οποίο περιγράφει ένα tag, με τη βοήθεια του attribute "type", το οποίο μπορεί να πάρει μία από τις τιμές "atomic", "complex" και "multidim".

Ειδικά για την αναπαράσταση context ακμών, οι οποίες δεν έχουν όνομα, αλλά χαρακτηρίζονται από τον context specifier που περιγράφει το explicit context τους, ορίσαμε ένα επιπλέον attribute με όνομα "specifier", του οποίου η τιμή είναι το explicit context. Τα tag που περιέχουν αυτό το attribute μπορούν να έχουν θεωρητικά οποιοδήποτε όνομα, αφού αυτό ούτως ή άλλως αγνοείται, το όνομα πάντως που χρησιμοποιούμε για ευκολία στην ανάγνωση είναι το "context"[8].

[8] Το όνομα αυτό δεν θεωρείται δεσμευμένη λέξη για τη μορφή του αρχείου αυτού. Μια entity ακμή μπορεί να ονομάζεται "context", φτάνει να μην έχει το "specifier" attribute.

Για να συνοψίσουμε τα παραπάνω, δίνουμε το παρακάτω αρχείο MOX, το οποίο περιγράφει το αντικείμενο music_club του σχήματος 2:

```
<music_club type='complex' id='2'>
  <menu type='multidim' id='37'>
    <context specifier='[lang=gr]' type='atomic' id='38'>
      Greek menu
    </context>
    <context specifier='[lang=en]' type='atomic' id='39'>
      English menu
    </context>
    <context specifier='[lang=fr]' type='atomic' id='40'>
      French menu
    </context>
  </menu>
  <name type='atomic' id='3'>Some name</name>
  <address type='multidim' id='4'>
    <context specifier='[season=summer]' type='complex' id='6'>
      <zipcode type='atomic' id='11'>Some code</zipcode>
      <street type='atomic' id='12'>Some street</street>
      <city type='atomic' id='14'>Athens</city>
    </context>
    <context specifier='[season in {fall,winter,spring}]' id='7'>
      <city ref='14' />
      <street type='atomic' id='13'>Some street</street>
    </context>
  </address>
  <review type='multidim' id='5'>
    <context specifier='[detail=low]' type='atomic' id='8'>
      6
    </context>
    <context specifier='[detail=high]' type='complex' id='9'>
      <score type='atomic' id='8'></score>
      <comments type='atomic' id='10'></comments>
    </context>
  </review>
  <parking type='multidim' id='28'>
    <context specifier='[daytime=evening]' type='atomic' id='30'>
      Parking1
    </context>
    <context specifier='[daytime=noon]' type='atomic' id='31'>
      Parking2
    </context>
  </parking>
</music_club>
```

4.3 Υλοποίηση λειτουργιών

4.3.1 Κωδικοποίηση κόσμων

Έχουμε ήδη αναφερθεί στην ανάγκη αντιστοίχισης κάθε κόσμου που εμφανίζεται στο universal context ενός MOEM γράφου με ένα ακέραιο που τον περιγράφει αμφιμονοσήμαντα. Η υλοποίηση των λειτουργιών κωδικοποίησης ενός κόσμου με τη βοήθεια ακεραίου και εξαγωγής ενός context specifier που περιγράφει ένα κόσμο με βάση τον κωδικό αυτό επιτυγχάνεται μέσω μεθόδων της κλάσης WorldCoding που συμπεριλήφθηκε στο πακέτο moemgraph. Κάθε MOEM γράφος πρέπει να συνοδεύεται από ένα αντικείμενο της κλάσης αυτής στην οποία αποθηκεύεται το universal context του. Αυτό γίνεται με την προσθήκη του πεδίου wCoding στην κλάση MOEMGraph, το οποίο δείχνει σε αυτό το WorldCoding αντικείμενο.

Η κλάση WorldCoding αποθηκεύει στο δισδιάστατο πίνακα vDim τα ονόματα όλων των διαστάσεων που εμφανίζονται στο γράφο, συνοδευόμενα με όλες τις δυνατές τιμές που αυτές μπορούν να πάρουν. Με κατάλληλο κατασκευαστή της κλάσης, ο οποίος δέχεται ως παράμετρο το αντικείμενο MOEMGraph που περιγράφει το γράφο, διαβάζονται όλες οι context ακμές του γράφου προκειμένου να συμπληρωθεί ο vDim με τις κατάλληλες πληροφορίες. Εκτός αυτού, η κλάση WorldCoding παρέχει μεθόδους που επιτρέπουν την επεξεργασία του vDim από το χρήστη της.

Αφού έχει σχηματιστεί σωστά ο πίνακας vDim, μέθοδοι της κλάσης αναλαμβάνουν να τις λειτουργίες κωδικοποίησης και αποκωδικοποίησης. Δεδομένου ενός context specifier που παριστάνεται από ένα αντικείμενο της κλάσης ContSpec, η μέθοδος calcWorlds() επιστρέφει ένα Vector που περιέχει τους ακεραίους που αντιστοιχούν στους κόσμους που περιλαμβάνονται στον specifier. Αντίστροφα, η μέθοδος makeContSpec() επιστρέφει ένα αντικείμενο ContSpec που περιγράφει το σύνολο των κόσμων που περνάμε στη μέθοδο ως παράμετρο με τη μορφή ενός Vector.

Αξίζει στο σημείο αυτό να σημειώσουμε ότι ο context specifier που προκύπτει είναι κατά το δυνατόν απλοποιημένος, χρησιμοποιώντας τόσο μεθόδους που προϋπήρχαν στις κλάσεις που αναλαμβάνουν την αναπαράσταση context specifier, όσο και μεθόδους που δημιουργήθηκαν κατά την επέκταση του MSSD Designer, οι οποίες έχουν το πλεονέκτημα ότι λαμβάνουν υπόψη τους και το universal context του αντίστοιχου γράφου.

4.3.2 Μετατροπή MOEM σε OEM

Οι λειτουργίες μετατροπής MOEM σε OEM και αντίστροφα υλοποιούνται από μεθόδους που, για λόγους οργάνωσης, περιλαμβάνονται στην κλάση GraphOperations του πακέτου moemgraph. Οι αλγόριθμοι που επιτυγχάνουν τις λειτουργίες αυτές έχουν ήδη περιγραφεί αναλυτικά στην ενότητα 3.2.1, σελ. 22. Οι μέθοδοι που τους υλοποιούν είναι οι genOEM() και getMOEMfromOEM(), από τις οποίες η δεύτερη παράλληλα με την αντίστοιχη μετατροπή ελέγχει τον δεδομένο OEM γράφο αν βρίσκεται σε μορφή που μπορεί να παράγει MOEM και σε περίπτωση λάθους επιστρέφει εξαίρεση της κλάσης InvalidMOEMasOEMException στην καλούσα διαδικασία.

4.3.3 Ο parser της MQL

Το πακέτο mqlparser περιέχει τις κλάσεις που απαιτούνται για τη μετάφραση των MQL ερωτήσεων που θα υποβάλλει ο χρήστης σε Lorel ερωτήσεις που θα εκτελούνται από το Lore. Όπως φαίνεται και από την αντίστοιχη ενότητα στο κεφάλαιο σχεδιασμού του συστήματος, πολύ σημαντικός είναι ο τρόπος χειρισμού των context path expression της MQL. Για αυτό το λόγο δημιουργήσαμε τρεις κλάσεις: την Variable (για παράσταση όλων των τύπων μεταβλητών), την PathComp (τμήματα ενός path) και την PathExpression, με τη βοήθεια των οποίων παριστάνουμε ένα path με μια δομή δεδομένων, την οποία μπορούμε να επεξεργαστούμε κατάλληλα και να μεταφράσουμε στην αντίστοιχη Lorel έκφραση, όπως αυτή έχει ήδη περιγραφεί. Στα επόμενα θα περιγράψουμε τις κλάσεις αυτές καθώς και τον τρόπο που το σύστημα τις αξιοποιεί προκειμένου να υλοποιήσει τη μετάφραση.

Η κλάση Variable

Αρχίζουμε με την περιγραφή της κλάσης αυτής, αφού χρησιμοποιείται από όλες τις υπόλοιπες. Ο ρόλος της είναι να συγκεντρώνονται στα πεδία της διάφορες πληροφορίες που έχουν σχέση με τη μεταβλητή που παριστάνεται από αυτή.
Συγκεκριμένα, εκτός του ονόματος της μεταβλητής, ένα αντικείμενο της κλάσης αυτής παρέχει τα εξής για τη μεταβλητή αυτή:

1) το πεδίο nType χαρακτηρίζει κάθε μεταβλητή ως:
   i) OBJ_VAR: η μεταβλητή εμφανίζεται είτε στην αρχική MQL ερώτηση, είτε σε κάποιο ενδιάμεσο στάδιο επεξεργασίας αυτής, είναι δε object variable.
   ii) CXT_VAR: ομοίως, μόνο που είναι context variable
   iii) LOREL_VAR: η μεταβλητή είναι μεν object variable, παράγεται όμως κατά το στάδιο μετάφρασης της MQL ερώτησης σε Lorel και εμφανίζεται μόνο στην τελική Lorel ερώτηση που παράγεται.
2) Σε περίπτωση που η μεταβλητή είναι τύπου LOREL_VAR, τα πεδία csContext (τύπου ContSpec) και cxtVar (τύπου Variable, είναι πάντα για αυτή nType = CXT_VAR) παρέχουν τον context specifier ή την context μεταβλητή αντίστοιχα που συνδέονται με αυτή τη μεταβλητή κατά τρόπο που θα περιγραφεί πιο κάτω.

Για να υλοποιήσουμε τον πίνακα συμβόλων, με τη βοήθεια του οποίου θα γίνει το parsing, ορίσαμε την επόμενη κλάση, η οποία διαχειρίζεται αντικείμενα της κλάσης Variable.

Η κλάση QNameSpace

Όπως θα δούμε πιο αναλυτικά και παρακάτω, καθώς αρχίζει η μετάφραση ενός MQL ερωτήματος, το πρώτο πράγμα που γίνεται είναι να δημιουργείται ένα αντικείμενο αυτής της κλάσης.

Στην κλάση αυτή αποθηκεύονται στο Vector vNames οι μεταβλητές που έχουν δηλωθεί από τον χρήστη στο ερώτημα, αλλά και οι βοηθητικές μεταβλητές που παράγονται από τη διαδικασία της μετάφρασης, ούτως ώστε να μπορεί να λαμβάνει τα αντικείμενα Variable που τις περιγράφουν δίνοντας σαν παράμετρο το όνομά τους. Αυτή ακριβώς τη λειτουργία αναλαμβάνει η μέθοδος getVar(), η οποία επιπρόσθετα σε περίπτωση που της ζητηθεί μεταβλητή που δεν υπάρχει στο vNames, την προσθέτει προσωρινά στο Vector vUnresolvedNames. Όταν στη συνέχεια του parsing βρεθεί η δήλωσή μιας μεταβλητής που εμφανίζεται σε αυτό το Vector, αφαιρείται πρώτα από αυτό πριν προστεθεί στο vNames.

Η προσθήκη μιας μεταβλητής στον πίνακα των ονομάτων γίνεται με τη μέθοδο addVar(). Σε περίπτωση δεύτερης δήλωσης μιας μεταβλητής, η addVar επιστρέφει null, ενώ ο parser επιστρέφει κατάλληλο exception.

Η κλάση QNameSpace παρέχει επιπλέον τις μεθόδους hasUnresolved() και getUnresolved(), με τις οποίες στο κατάλληλο στάδιο της μετάφρασης γίνεται έλεγχος για μεταβλητές που χρησιμοποιούνται στην ερώτηση αλλά δεν έχουν δηλωθεί.

Η κλάση PathComp

Με την κλάση αυτή παριστάνουμε στο σύστημά μας ένα path component. Όπως αυτή ορίζεται από τη γλώσσα, η γενική μορφή του είναι η εξής:

`(`*`context_label`*`)(`**`<`**`)`*`label`*`(`**`>`**`)((`**`:`**`)(`*`context_multidim`*`))(`**`{`**`*as_variable*`**`}`**`)`[9]

Για την παράσταση στη δομή του PathComp της παραπάνω γενικής μορφής χρησιμοποιούμε τα εξής πεδία σε αυτήν:

1) Για το *label* του τμήματος:
   i) το strLabel, τύπου String
   ii) το bMultidim, τύπου boolean, το οποίο είναι αληθές αν παριστάνεται πολυδιάστατος κόμβος
2) Για το *context_label*:
   i) το cxtLabel, τύπου ContSpec, αν είναι context specifier
   ii) το cxtVarLabel, τύπου Variable, αν είναι context μεταβλητή
3) Για το *context_multidim*:
   i) το cxtMultidim, τύπου ContSpec, αν είναι context specifier
   ii) το cxtVarMultidim, τύπου Variable, αν είναι context μεταβλητή

[9] Στο εξής με **παχιά** γράμματα σημειώνουμε σύμβολα που εμφανίζονται ως έχουν στην έκφραση, ενώ με *πλάγια* εκφράσεις που ορίζονται από τη γλώσσα. Οι παρενθέσεις χρησιμοποιούνται όπως και στις κανονικές εκφράσεις (regular expressions).

iii) τα πεδία τύπου boolean bInherited και bExplicit που καθορίζουν αν το context_multidim αναφέρεται στο inherited ή explicit context της ακμής, αντίστοιχα

4) Για το *as_variable* το πεδίο τύπου Variable asVar

Επειδή η κλάση χρησιμοποιείται και για την παράσταση του αρχικού τμήματος κάθε path expression, του οποίου το *label* αναφέρεται σε μεταβλητή και όχι σε label ακμής, στα πεδία συμπεριλαμβάνεται και το πεδίο τύπου Variable startVar, στο οποίο αποθηκεύεται η μεταβλητή αυτή.

Η βασική λειτουργία που επιτελείται με αυτήν την κλάση είναι το parsing των path component κατά τη δημιουργία ενός αντικειμένου της. Τυχόν λάθος στη σύνταξή του από τον χρήστη επιστρέφει MQLParseException, το οποίο διαχειρίζεται κατάλληλα ο parser. Αντικείμενα της κλάσης αυτής συναποτελούν αντικείμενα της κλάσης PathExpression, η περιγραφή της οποίας ακολουθεί.

Η κλάση PathExpression

Με την κλάση αυτή υλοποιείται το parsing των path expression σε ένα MQL ερώτημα. Η δομή της περιλαμβάνει ένα Vector το οποίο περιέχει με τη σειρά όλα τα τμήματα που σχηματίζουν το εκάστοτε path. Εκτός αυτού, περιλαμβάνονται άλλα δύο πεδία τύπου Variable στα οποία αποθηκεύεται τόσο η μεταβλητή με την οποία αρχίζει το path (και είναι ίδια με τη startVar στο πρώτο PathComp του Vector), όσο και η μεταβλητή στην οποία αποτιμάται το path (και είναι ίδια με τη asVar του τελευταίου PathComp). Η δημιουργία ενός αντικειμένου της κλάσης αυτής γίνεται είτε με παράμετρο ένα String, στο οποίο γίνεται parsing, είτε με παραμέτρους τα PathComp που την αποτελούν.

Όλες οι παραπάνω κλάσεις διαθέτουν μεθόδους οι οποίες αναλαμβάνουν να επιτελέσουν λειτουργίες και μετασχηματισμούς πάνω σε αυτές, οι οποίες αξιοποιούνται κατάλληλα από το μεταφραστή της MQL σε Lorel, ο οποίος υλοποιείται με την κλάση MQLParser.

Η κλάση MQLParser

Στην κλάση αυτή περιλαμβάνονται όλες οι μέθοδοι που χρειάζονται για τη μετάφραση μιας MQL ερώτησης σε ισοδύναμη Lorel. Αντικείμενα της κλάσης αυτής δημιουργούνται από το σύστημα μία φορά για έναν συγκεκριμένο γράφο και στη συνέχεια μπορεί να κληθεί η μέθοδος parse(String), η οποία δέχεται σαν παράμετρο το αρχικό MQL query και επιστρέφει τη μετάφρασή του σε Lorel.

Ένα αντικείμενο της κλάσης MQLParser περιέχει την πληροφορία για το όνομα της βάσης στην οποία υποβάλλεται η κάθε ερώτηση, το οποίο είναι απαραίτητο για τον έλεγχο ορθότητας της ερώτησης, καθώς και για την υλοποίηση της μετάφρασης. Επιπλέον, από το MOEM γράφο στον οποίο υποβάλλεται η ερώτηση λαμβάνεται και το πεδίο wCoding που είναι αντικείμενο της κλάσης WorldCoding, το οποίο περιγράφει το universal context του γράφου και παρέχει μεθόδους για την κωδικοποίηση, όπου αυτό είναι αναγκαίο, των context specifier σε σύνολα κόσμων.

Ο πιο σημαντικός τύπος ερώτησης που μπορεί να τεθεί στο σύστημα ως γνωστόν είναι οι ερωτήσεις τύπου select…from…where…within. Στην υλοποίησή μας υποστηρίζονται και άλλες μορφές ερωτήσεων, παρόμοιες με της Lorel, καθώς και φωλιασμένες ερωτήσεις στο select clause. Ο parser είναι έτσι σχεδιασμένος, ώστε να τελεί παράλληλα τη λεκτική και συντακτική ανάλυση του δεδομένου ερωτήματος με τη μετάφρασή του. Η διαδικασία μετάφρασης του ερωτήματος ακολουθεί τα παρακάτω βήματα:

1. Αρχικά με τη βοήθεια της μεθόδου scanClauses() ανιχνεύονται στο ερώτημα τα substring του τα οποία αποτελούν περιεχόμενο κάθε πιθανού clause. Η μέθοδος φροντίζει να χειρίζεται κατάλληλα την πιθανότητα nested queries, και επιστρέφει σε έναν πίνακα τα αντίστοιχα string, για τα clause που υπάρχουν στην ερώτηση.

2. Στη συνέχεια καλείται η μέθοδος from_expr(), με παράμετρο το περιεχόμενο του from clause, το οποίο είναι υποχρεωτικό να εμφανίζεται στο ερώτημα. Η μέθοδος αναλαμβάνει να διαχωρίσει, αν υπάρχουν περισσότερα του ενός, τα path που εμφανίζονται στο clause και να κατασκευάζει τα αντίστοιχα αντικείμενα της κλάσης PathExpression. Καθώς το πρόγραμμα αναγνωρίζει δηλώσεις ή και αναφορές σε object ή context μεταβλητές, καλεί τις ανάλογες μεθόδους της κλάσης QNameSpace, addVar() ή getVar() αντίστοιχα, προκειμένου να ενημερωθεί ο πίνακας συμβόλων, αλλά και να ελεγχθεί αν οι αναφερόμενες μεταβλητές ορίζονται στο clause. Αφού τελειώσει η διαδικασία χωρίς λάθη, επιστρέφεται Vector με τα PathExpression που δημιουργήθηκαν σε αυτή.

3. Το σύνολο αυτό ελέγχεται για μεταβλητές, το path με το οποίο ορίζονται εμφανίζεται ίδιο συντακτικά ως πρόθεμα σε άλλα path. Σε αυτήν την περίπτωση τα PathExpression ενημερώνονται, αντικαθιστώντας το path – πρόθεμα με τη μεταβλητή αυτή. Όλα αυτά γίνονται με κλήση της μεθόδου mergePaths με παράμετρο το σύνολο των PathExpression που λάβαμε από την αμέσως προηγούμενη διαδικασία.

4. Η επόμενη διαδικασία συνίσταται στην απλοποίηση των MQL path expression σε εκφράσεις που αποτελούνται από την αρχική μεταβλητή και ένα μόνο path component, όπως αυτή παρουσιάστηκε αναλυτικά σε προηγούμενο κεφάλαιο. Η απλοποίηση αυτή γίνεται στο επίπεδο της δομής της κλάσης PathExpression και όχι με παραγωγή MQL κώδικα που στη συνέχεια περνάει ξανά από τον parser[10]. Η απλοποίηση γίνεται για κάθε path ξεχωριστά, με κλήση της μεθόδου getSimplePaths() της κλάσης PathExpression. Οι νέες μεταβλητές που παράγονται καθώς απλοποιείται κάθε path expression προστίθενται και αυτές στο τρέχον αντικείμενο της κλάσης QNameSpace για το συγκεκριμένο query. Αφού τελειώσει και αυτή η διαδικασία καταλήγουμε στην τελική μορφή του MQL ερωτήματος αμέσως πριν τη μετάφρασή του σε Lorel, έχοντας ενδιάμεσα συγκεντρώσει όλες τις απαιτούμενες πληροφορίες για αυτή.

[10] Είναι πάντως εύκολο να επιστρέφεται σε μορφή MQL έκφρασης κάθε ενδιάμεσο στάδιο της μετάφρασής της σε Lorel.

5. Προκειμένου να ολοκληρωθεί η μετάφραση αυτή κατασκευάζουμε αρχικά το from clause της Lorel ερώτησης με τη βοήθεια της μεθόδου from_clause(). Όπως περιγράφηκε και προηγούμενα, στο clause αυτό θα έχουμε για κάθε απλό path τρία Lorel path expression που το συγκροτούν, κατά το σχηματισμό των οποίων ορίζουμε δύο (ή ένα αναλόγως) αντικείμενα τύπου Variable, για τα οποία το πεδίο nType παίρνει τιμή LOREL_VAR, προκειμένου να τις διαχωρίσουμε από τις υπόλοιπες, MQL μεταβλητές. Οι μεταβλητές αυτές έχουν επιπλέον πληροφορίες για το context έναντι του οποίου θα πρέπει να συγκριθεί το inherited ή explicit ανάλογα context που είναι αποθηκευμένο στο γράφο στη μορφή απογόνων των OEM κόμβων στους οποίους αποτιμούνται οι μεταβλητές αυτές. Η πληροφορία αυτή είναι είτε ένας context specifier που έχει δηλωθεί ρητά στο MQL ερώτημα, είτε η κατάλληλη context μεταβλητή που συσχετίζεται με κάθε path component.

6. Η κατασκευή του where clause της Lorel ερώτησης γίνεται σε δύο στάδια. Στο παρόν βήμα για κάθε μεταβλητή τύπου LOREL_VAR που ορίστηκε προηγουμένως προστίθεται στο clause μια σειρά εκφράσεων με τις οποίες υλοποιείται ο έλεγχος για το αν το context που ορίζεται από τον specifier του path component αποτελεί ή όχι υποσύνολο του inherited (ή explicit) context της ακμής που αντιστοιχεί στη μεταβλητή. Η μορφή αυτών των εκφράσεων έχει ήδη παρουσιαστεί αναλυτικά.

7. Κατά το δεύτερο στάδιο γίνεται το parsing του within clause, εφόσον φυσικά αυτό υπάρχει. Η προσέγγισή του είναι αρκετά πιο απλή, αφού μετά το parsing αρκεί να αντικαταστήσουμε κάθε context μεταβλητή που εμφανίζεται σε αυτό με την αντίστοιχη έκφραση *set_query*, όπως έχουμε περιγράψει. Προκειμένου να υπολογίσουμε την έκφραση αυτή, διατρέχουμε για κάθε τέτοια μεταβλητή το πίνακα συμβόλων και συγκεντρώνουμε όλες τις LOREL_VAR που έχουν οριστεί στο βήμα 5 και για τις οποίες είχε καθοριστεί κάποιο από τα πεδία cxtVarLabel, cxtVarMultidim να δείχνουν στην αντίστοιχη context μεταβλητή. Όσον αφορά στην έκφραση που έχουμε προβλέψει για τους context specifier που εμφανίζονται στο within clause, χρησιμοποιούμε τη μέθοδο getCodesQuery(), η οποία παρέχεται από την κλάση WorldCoding, που αντικείμενό της είναι το πεδίο wCoding. Η μέθοδος αυτή δέχεται σαν παράμετρο ένα ContSpec και επιστρέφει το String της αντίστοιχης έκφρασης, όπως αυτή έχει ήδη περιγραφεί.

8. Στο τελικό βήμα συγκεντρώνονται οι εκφράσεις που παράχθηκαν από τα προηγούμενα και κατασκευάζεται η τελική Lorel ερώτηση που θα υποβληθεί στον ισοδύναμο OEM γράφο.

Προκειμένου να υποστηρίξουμε δυνατότητα nested queries στο select clause, αμέσως πριν το τελικό βήμα καλείται η μέθοδος select_expr(), η οποία αναλαμβάνει να κάνει parsing και στο select clause, ανιχνεύοντας τα διάφορα αντικείμενα που πρέπει να επιστρέφονται από την ερώτηση και τα οποία χωρίζονται με κόμμα. Αυτά μπορεί να είναι είτε μεταβλητές αντικειμένων που έχουν οριστεί στο from clause, οπότε γίνεται ένας έλεγχος αν έχουν καταχωρηθεί στον πίνακα συμβόλων, είτε φωλιασμένα select queries.

Στη δεύτερη περίπτωση καλείται ξανά από τη αρχή η διαδικασία με την οποία γίνεται η μετάφραση ενός select ερωτήματος, μόνο που ο πίνακας συμβόλων αρχικοποιείται κατάλληλα, ώστε να περιλαμβάνει τα ονόματα των μεταβλητών που εμφανίστηκαν στο from clause του παρόντος query, ώστε να θεωρηθούν γνωστά κατά τη διαδικασία της μετάφρασης του νέου query.

4.3.4 Πρόσβαση στο σύστημα Lore

Το γεγονός ότι το σύστημα Lore έχει αναπτυχθεί σε περιβάλλον Linux θέτει εν γένει έναν σημαντικό περιορισμό στη μεταφερσιμότητα του συστήματος επερωτήσεων και σε αυτή του MSSDesigner. Χωρίς αυτόν τον περιορισμό και λόγω της χρήσης της Java Virtual Machine, η συγκεκριμένη εφαρμογή θα μπορούσε να χρησιμοποιηθεί σε οποιοδήποτε σύστημα μπορεί να τρέξει η JVM.

Για να περιοριστεί αυτό το πρόβλημα αποφασίστηκε να αναπτυχθεί ανεξάρτητα από τον MSSDesigner ένας δικτυακός διακομιστής (Web Server) ο οποίος τρέχει σε περιβάλλον Linux και χειρίζεται αποκλειστικά το κομμάτι του υποσυστήματος ερωτήσεων που έχει να κάνει με τη χρήση του συστήματος Lore. Η συγκεκριμένη μέθοδος αντιμετώπισης του προβλήματος για ολοκληρωμένη λειτουργία του συστήματος επερωτήσεων προϋποθέτει την πρόσβαση του MSSDesigner στο διαδίκτυο αλλά το πλεονέκτημα είναι φυσικά το γεγονός ότι η εφαρμογή μπορεί έτσι να εκμεταλλευτεί τη μεταφερσιμότητα που παρέχει η Java.

Άμεση συνέπεια αυτού του σχεδιασμού είναι η ανάγκη ανάπτυξης ενός μηχανισμού επικοινωνίας μεταξύ του MSSDesigner και του συγκεκριμένου διακομιστή. Αυτός ο μηχανισμός αποτελεί για την εφαρμογή ενός συγκεκριμένου υποσυστήματος το οποίο δέχεται δεδομένα κάποιας συγκεκριμένης μορφής, και ανάλογα με αυτά στέλνει στον διακομιστή τις κατάλληλες εντολές σε μορφή HTTP requests.

Πιο συγκεκριμένα τώρα, στα πλαίσια της εφαρμογής ορίστηκε κάποιο σύνολο εντολών η οποίες αποθηκεύονται με κάποια προκαθορισμένη δομή στο σύστημα. Κάθε τέτοια δομή αντιστοιχεί σε κάποια συγκεκριμένη HTTP request η οποία με τη σειρά της δίνει μια συγκεκριμένη εντολή ζητώντας κάποιο συγκεκριμένο αρχείο από τον server, και περιμένει να λάβει μια συγκεκριμένη απάντηση καθώς και τα κατάλληλα δεδομένα ανάλογα με την περίπτωση.

Οι τρεις βασικές λειτουργίες που γίνονται στον server είναι:

- Η δημιουργία μιας Lore βάσης με βάση κάποιο αρχείο με SSD εκφράσεις.
- Η υποβολή ερωτήσεων στη συγκεκριμένη βάση και δημιουργία αρχείου .mox με τα αποτελέσματα.
- Η καταστροφή της βάσης μετά το τέλος χρήσης αυτής.

Σε συμφωνία με τις παραπάνω λειτουργίες οι εντολές που στέλνει το υποσύστημα επικοινωνίας του MSSDesigner είναι οι εξής:

- Αίτηση δημιουργίας βάσης (αίτηση για το αρχείο DbCreate.html) η οποία περιέχει ως παράμετρο το αρχείο με την πληροφορία για τη δημιουργία αυτής, όπως αυτό έχει δημιουργηθεί με βάση τον ανάλογο MOEM γράφο.
- Αίτηση για υποβολή ερώτησης στέλνοντας κάποιο request για το αρχείο που θα περιλαμβάνει την απάντηση το οποίο όπως έχει αναφερθεί και παραπάνω είναι τύπου .mox (LoreResult.mox), θέτοντας ως παράμετρο την αντίστοιχη Lorel ερώτηση.
- Αίτηση για καταστροφή της βάσης ζητώντας το αρχείο DbDestroy.html.

Μια τυπική περίπτωση επικοινωνίας μεταξύ του MSSDesigner και του server είναι η εξής: Μετά τη δημιουργία του αρχείου με ssd-expressions που περιέχει την πληροφορία του ισοδύναμου OEM γράφου, δημιουργείται μια εντολή δημιουργίας βάσης η οποία περιέχει το συγκεκριμένο αρχείο και στέλνεται στον διακομιστή. Όταν ο τελευταίος λάβει την αίτηση και την αναγνωρίσει δημιουργεί την βάση και στέλνει το αρχείο DbCreate.html καθώς και κάποιο κωδικό επιτυχούς δημιουργίας της βάσης. Έτσι ο MSSDesigner παρέχει πλέον τη δυνατότητα υποβολής MQL ερωτήσεων. Με την υποβολή μιας τέτοιας ερώτησης και αφού δημιουργηθεί η ισοδύναμη Lorel, αυτή στέλνεται στον server. Εκεί γίνεται η εκτέλεσή της και στη συνέχεια το .mox αρχείο που δημιουργείται με βάση την απάντηση από το Lore συμπεριλαμβάνεται στην απάντηση που περιμένει ο MSSDesigner. Τέλος, με το κλείσιμο του υποσυστήματος επερωτήσεων στέλνεται εντολή καταστροφής της βάσης μέσω της αίτησης DbDestroy.html, και ο server επιστρέφει τον ανάλογο κωδικό.

Ο server έχει υλοποιηθεί με χρήση της γλώσσας Java, και μας δόθηκε έτοιμος για την ολοκλήρωση της συγκεκριμένης εργασίας. Αυτό που προστέθηκε για την εξυπηρέτηση των αναγκών της συγκεκριμένης εφαρμογής όπως αυτή περιγράφηκε παραπάνω είναι κάποιος κώδικας για την εκτέλεση συγκεκριμένων λειτουργιών όταν ζητούνται συγκεκριμένα html αρχεία. Έτσι ουσιαστικά προστέθηκε μια κλάση που ονομάζεται *LoreUse*, και η οποία περιέχει τρεις μεθόδους κάθε μία από τις οποίες καλεί ένα εξωτερικό πρόγραμμα με τις κατάλληλες παραμέτρους. Αυτές οι μέθοδοι είναι οι εξής:

- DbLoreCreation(): (Δημιουργία Lore βάσης) Δέχεται ως παράμετρο το αρχείο με την πληροφορία για την δημιουργία της βάσης και ένα όνομα για τη βάση αυτή. Καλεί το εξωτερικό πρόγραμμα του Lore το οποίο ονομάζεται DbCreate. Το όνομα της βάσης που δημιουργείται για κάθε γράφο είναι το ίδιο (Loredb).
- DbLoreDestruction(): Δέχεται ως παράμετρο το όνομα μιας βάσης και την καταστρέφει. Καλεί το εξωτερικό πρόγραμμα του Lore “DbDestroy”.
- LoreQuerySubmit():Υποβολή μιας ερώτησης και δημιουργία του LoreResult.mox. Καλεί το πρόγραμμα QuerySub, το οποίο δέχεται ως παράμετρο το όνομα της βάσης στην οποία θα υποβληθεί η ερώτηση καθώς και την προς υποβολή Lorel ερώτηση.

Πέρα από αυτή την κλάση έχουν προστεθεί στα κατάλληλα σημεία του κώδικα του server κάποιες συναρτήσεις που κάνουν τη δουλειά της αναγνώρισης των συγκεκριμένων αιτήσεων, της κλήσεις των ανάλογων μεθόδων από την κλάση *LoreUse* καθώς και του απαραίτητου parsing των παραμέτρων για την αποθήκευση του αρχείου με τις ssd εκφράσεις τοπικά και των ερωτήσεων σε κάποια μεταβλητή

Στη πλευρά του MSSDesigner τώρα για την υλοποίηση της επικοινωνίας έχουν προστεθεί δύο κλάσεις: Η *CommObject* και η *MOEMClient*. Η πρώτη είναι μια απλή κλάση που δημιουργεί αντικείμενα που περιέχουν τρία πεδία για την αποθήκευση μιας αίτησης και των παραμέτρων αυτής. Η δεύτερη είναι η βασική κλάση που δέχεται αντικείμενα *CommObject* και δημιουργεί έγκυρες html αιτήσεις με τις σωστές παραμέτρους.

4.3.5 Επεξεργασία αποτελεσμάτων

Όταν ο server λαμβάνει κάποια Lorel ερώτηση από τον MSSDesigner την υποβάλλει στο σύστημα Lore περνώντας την ως παράμετρο στο πρόγραμμα QuerySub. Το συγκεκριμένο πρόγραμμα έχει αναπτυχθεί με χρήση της C++ σε περιβάλλον Linux και κάνει χρήση του API του Lore. Η βασική λειτουργία που εκτελεί είναι η υποβολή της ερώτησης στην τοπική βάση του συστήματος, η οποία αντιστοιχεί στον αρχικό MOEM γράφο με βάση τον οποίο έχει αρχικοποιηθεί το υποσύστημα των ερωτήσεων του MSSDesigner, και η αποθήκευση της απάντησης που επιστρέφει το Lore σύστημα σε μορφή .mox αρχείου.

Όπως έχει περιγραφεί σε προηγούμενη παράγραφο, το Lore επιστρέφει την απάντηση σε κάποια ερώτηση που του έχει υποβληθεί μέσω μιας δομής στη μνήμη τύπου LoreOem. Το QuerySub, χρησιμοποιώντας τις μεθόδους που παρέχει το Lore API, διασχίζει κατά βάθος τη δομή της απάντησης. Σε κάθε κόμβο που συναντάει προσδίδει ένα αριθμό που τον αναγνωρίζει μοναδικά και κρατάει σε κάποιο πίνακα τα νούμερα των κόμβων που έχει επισκεφθεί. Έτσι σε περίπτωση που συναντήσει κάποιο κόμβο που έχει ήδη καταγράψει στο αρχείο, σταματάει την αναδρομική διάσχιση του γράφου σε αυτόν τον κόμβο και μεταφέρει στο αρχείο την ακμή που καταλήγει σε αυτόν, ορίζοντας τον κόμβο με τη βοήθεια του αναγνωριστικού του. Με αυτόν τον τρόπο επιτυγχάνεται και η σωστή αντιμετώπιση των κύκλων.

Η απάντηση που επιστρέφεται από το σύστημα Lore, είναι όπως γνωρίζουμε ένας γράφος από τη ρίζα του οποίου ξεκινούν ακμές που καταλήγουν σε καθέναν από τους κόμβους που ανήκουν στο σύνολο της απάντησης που αντιστοιχεί στο ερώτημα που έχουμε υποβάλλει. Το πρόβλημα είναι ότι τα ονόματα των ακμών αυτών δεν αντιστοιχούν στα ονόματα των ακμών που καταλήγουν στους αντίστοιχους κόμβους που εμφανίζονται στον αρχικό MOEM γράφο, λόγω του σχεδιασμού του OEM μοντέλου. Για να αντιμετωπιστεί αυτό, λαμβάνουμε τα κατάλληλα ονόματα αξιοποιώντας τη πληροφορία που μας δίνεται μέσω του κόμβου MATCHEDPATH, όπου αποθηκεύονται από το σύστημα Lore τα path που αντιστοιχούν σε κάθε κόμβο-απάντηση.

Γενικά η απάντηση που επιστρέφεται είναι σύμφωνη με το μοντέλο OEM γράφου που έχουμε ορίσει. Εξαίρεση αποτελεί το τμήμα του γράφου μεταξύ της ρίζας και των κόμβων-απαντήσεων. Για αυτό το λόγο παρεμβάλλονται από το πρόγραμμα ενδιάμεσοι κόμβοι και κατάλληλες ακμές στο τμήμα αυτό που καθιστούν τον τελικό γράφο έγκυρο.

4.4 Υλοποίηση διαπροσωπείας

Για την ενσωμάτωση του συστήματος υποβολής επερωτήσεων στην εφαρμογή MSSDesigner πρέπει να γίνουν κάποιες επεκτάσεις της διαπροσωπείας χρήστη (GUI) για να δίνεται η δυνατότητα πλήρους ελέγχου του υποσυστήματος, εύκολης υποβολής ερωτήσεων, παρουσίασης και μελέτης των ενδιάμεσων βημάτων μέχρι την παραγωγή του τελικού αποτελέσματος της ερώτησης.

Αρχικά λοιπόν θα προστεθεί ένα νέο μενού στα ήδη υπάρχοντα με την ονομασία «Query». Σκοπός αυτού είναι να παρέχονται οι εξής λειτουργίες που αφορούν το σύστημα υποβολής MQL ερωτήσεων:

- **Αρχικοποίηση του συστήματος:** Μέσω της επιλογής "Initialize DB" γίνονται τα ανάλογα βήματα προετοιμασίας πριν την ενεργοποίηση της δυνατότητας υποβολής των επερωτήσεων πάνω σε κάποιον επιλεγμένο MOEM γράφο. Πιο συγκεκριμένα αρχικά γίνεται απενεργοποίηση κάποιου τυχόντος ενεργοποιημένου συστήματος ερωτήσεων σε σχέση με κάποιον άλλο γράφο. Στη συνέχεια ο επιλεγμένος MOEM γράφος αρχικά κανονικοποιείται, στη συνέχεια παράγεται ο αντίστοιχος OEM γράφος και στέλνεται εντολή στον server να κάνει την παραγωγή της αντίστοιχης βάσης στο Lore σύστημα, στην οποία θα υποβληθούν αργότερα οι αντίστοιχες Lorel ερωτήσεις. Παράλληλα γίνεται οι κωδικοποίηση των ανάλογων κόσμων κάτω από τους οποίους ισχύουν τα δεδομένα του MOEM γράφου. Αφού έχουν ολοκληρωθεί επιτυχώς οι παραπάνω διαδικασίες ενεργοποιούνται και οι υπόλοιπες επιλογές του συγκεκριμένου μενού καθώς και δύο κουμπιά δίπλα στα ήδη υπάρχοντα κουμπιά του MSSDesigner.
- **Παρουσίαση του κανονικοποιημένου γράφου:** Μέσω της επιλογής "Show C.F." παρουσιάζεται στην επιφάνεια εργασίας ένα παράθυρο με τον αρχικό MOEM γράφο σε κανονική μορφή.
- **Παρουσίαση του ανάλογου OEM γράφου:** Μέσω της επιλογής "Show Equivalent OEM" παρουσιάζεται στην επιφάνεια εργασίας ένα παράθυρο με τον ισοδύναμο του αρχικού OEM γράφο.
- **Παρουσίαση της κωδικοποίησης των κόσμων:** Με την επιλογή "Show Context Coding" ο χρήστης μπορεί να δει την αναλογία κόσμων και κωδικών μέσω ενός παραθύρου στην επιφάνεια εργασίας. Αυτό γίνεται και μέσω του δεύτερου κατά σειρά παρουσίασης κουμπιού που έχει προστεθεί στο περιβάλλον.

- **Άνοιγμα παραθύρου υποβολής ερωτήσεων:** Επιλογή “MQL Query Window”. Ανοίγει το παράθυρο υποβολής ερωτήσεων, το οποίο περιγράφεται αναλυτικά στη συνέχεια. Η ίδια λειτουργία γίνεται και από το πρώτο από τα δύο κουμπιά που έχουν προστεθεί στο περιβάλλον.
- **Παρουσίαση αποτελέσματος MQL ερώτησης:** Μέσω της επιλογής “Show Query Result” παρουσιάζεται το αποτέλεσμα της τελευταίας MQL ερώτησης που έχει υποβληθεί, αφού στην ουσία διαβάζει τα αποτελέσματα από το τοπικό αρχείο στο οποίο αποθηκεύεται το αποτέλεσμα που επιστρέφει ο Lore server.
- **Απενεργοποίηση συστήματος υποβολής επερωτήσεων:** Με την επιλογή “Close DB” διαγράφονται από τη μνήμη όλα τα αποτελέσματα των ενδιάμεσων σταδίων που περιγράφηκαν στην αρχικοποίηση του συστήματος και τα οποία καλούνται μέσω των υπολοίπων επιλογών του μενού. Επίσης δίνεται εντολή στον Lore server να γίνει διαγραφή της Lore βάσης που δημιουργήθηκε στην αρχή. Πέρα από αυτά απενεργοποιούνται όλες οι επιλογές του μενού εκτός από αυτή της αρχικοποίησης του συστήματος.

Εκτός του συγκεκριμένου μενού “Query”, στο ήδη υπάρχον μενού “Operations” προστέθηκε μια ακόμη επιλογή με το όνομα “Canonical Form”, η οποία παρουσιάζει στην επιφάνεια εργασίας την κανονική μορφή κάποιου επιλεγμένου MOEM γράφου ανεξάρτητα από τις υπόλοιπες λειτουργίες του υποσυστήματος επερωτήσεων της εφαρμογής.

Για την υποβολή των MQL ερωτήσεων στο σύστημα όπως περιγράφηκε και παραπάνω θα είναι διαθέσιμο στο χρήστη ένα παράθυρο που επιτελεί αυτή αλλά και άλλες λειτουργίες που έχουν να κάνουν με την υποβολή των ερωτήσεων στο σύστημα.

Κατ’ αρχάς το συγκεκριμένο παράθυρο θα χωρίζεται σε τρεις περιοχές: Στην πρώτη περιοχή από πάνω προς τα κάτω ο χρήστης μπορεί να γράψει μια οποιαδήποτε MQL ερώτηση. Στην δεύτερη περιοχή υπάρχουν κουμπιά που κάνουν κάποιες λειτουργίες και στην τρίτη περιοχή εμφανίζεται η αντίστοιχη Lorel ερώτηση η οποία υποβάλλεται στο Lore σύστημα μέσω του διαθέσιμου server. Για πιο λεπτομερή μελέτη της αντίστοιχης Lorel ερώτησης θα είναι διαθέσιμη η λειτουργία της εμφάνισης στην επιφάνεια εργασίας ενός ξεχωριστού παραθύρου που την περιέχει.

Οι λειτουργίες που πρέπει να είναι διαθέσιμες μέσω των επιλογών της δεύτερης επιφάνειας εργασίας είναι οι εξής:

- ✓ Υποβολή ερώτησης στον Lore server (κουμπί “Submit”): Με την συγκεκριμένη επιλογή η MQL ερώτηση αφού ελεγχθεί για λάθη και μεταφραστεί στην αντίστοιχη Lorel ερώτηση υποβάλλεται με την ανάλογη εντολή στον Lore server, ο οποίος παράγει την απάντηση και την στέλνει στον MSSDesigner ο οποίος την αποθηκεύει τοπικά σε αρχείο xml μορφής.

- ✓ Παρουσίαση αποτελέσματος ερώτησης (κουμπί "Show Result"): Σε αυτή την περίπτωση το συγκεκριμένο κουμπί είναι ενεργοποιημένο μόνο στην περίπτωση που ο Lore server έχει επιστρέψει έγκυρο αποτέλεσμα και με την επιλογή του παράγεται ο ανάλογος γράφος-απάντηση σε MOEM μορφή.
- ✓ Παρουσίαση αντίστοιχης Lorel ερώτησης (κουμπί "Lorel Equivalent")
- ✓ Παρουσίαση ισοδύναμης Lorel ερώτησης στον ανάλογο χώρο του παραθύρου χωρίς την υποβολή αυτής στον Lorel server.
- ✓ Φόρτωμα MQL ερώτησης από αρχείο (κουμπί "Load"): Δίνεται η δυνατότητα άμεσου φορτώματος στο σύστημα μιας MQL ερώτησης από αρχείο txt. Σε κάθε τέτοιο αρχείο πρέπει να υπάρχει μια μόνο ερώτηση που τερματίζεται με τον ειδικό χαρακτήρα ';'.
- ✓ Καθαρισμός πεδίων παρουσίασης ερωτήσεων (κουμπί "Clear"): Μέσω αυτής της λειτουργίας καθαρίζονται οι περιοχές του παραθύρου όπου γράφεται η MQL ερώτηση και παρουσιάζεται η ισοδύναμη Lorel ερώτηση.

Η υλοποίηση των πιο πάνω επιπρόσθετων λειτουργιών στα πλαίσια του MSSDesigner στηρίχθηκε βασικά στην επέκταση της ήδη υπάρχουσας δομής κλάσεων με τις οποίες υλοποιείται η βασική διαπροσωπεία χρήστη αλλά και στη δημιουργία κάποιον καινούριων κλάσεων για την υλοποίηση καινούριων χαρακτηριστικών του GUI.

Πιο συγκεκριμένα η κύρια κλάση όπου υλοποιούνται όλα τα μενού και τα κουμπιά της βασικής διαπροσωπείας της εφαρμογής είναι η *Mainframe*. Εκεί λοιπόν προστέθηκαν οι ανάλογες εντολές και οι ανάλογες ρουτίνες για την υλοποίηση του βασικού μενού "Query" καθώς και η προσθήκη της επιλογής "Canonical Form" στο μενού "Operations" και των δύο νέων κουμπιών.

Πέρα των απλών προσθηκών νέων αντικειμένων σε συναρτήσεων στην *Mainframe* έγιναν κάποιες αλλαγές στον τρόπο κατασκευής και αντιμετώπισης των εσωτερικών παραθύρων της εφαρμογής: Στη βασική εφαρμογή τα εσωτερικά παράθυρα δημιουργούνται από την κλάση *InternalGraphFrame*, η οποία είναι απόγονος της βασικής *JInternalFrame* του Swing. Μέσω αυτής της κλάσης δημιουργούνται τα παράθυρα σχεδιασμού των γράφων στην εφαρμογή, η οποία ελέγχει, διαχειρίζεται και γενικά αναγνωρίζει μόνο αντικείμενα τύπου *InternalGraphFrame* στην επιφάνεια εργασίας της. Έτσι δεν υπήρχε η δυνατότητα δημιουργίας άλλων τύπων εσωτερικών παραθύρων που να έχουν π.χ. επιφάνειες επεξεργασίας κειμένου που χρειάζονται στην περίπτωση του υποσυστήματος επερωτήσεων. Για την επέκταση λοιπόν των τύπων των εσωτερικών παραθύρων της εφαρμογής χωρίς να επηρεαστεί το υπάρχων σύστημα ελέγχου και αναγνώρισης αυτών έγιναν τα εξής:

- Κλάση *MssDInternalFrame:* δημιουργήθηκε η συγκεκριμένη κλάση η οποία είναι απόγονος της *JInternalFrame* και το μόνο χαρακτηριστικό που προσθέτει σε αυτή είναι μια μεταβλητή boolean με όνομα *isGraph* που υποδεικνύει εάν το παράθυρο που δημιουργείται είναι γράφος ή όχι.

- Όλα τα εσωτερικά παράθυρα της εφαρμογής κληρονομούν αυτή την κλάση. Τα παράθυρα τύπου *InternalGraphFrame* θέτουν την μεταβλητή ελέγχου *isGraph* αληθής ενώ όλα τα εσωτερικά παράθυρα έχουν την συγκεκριμένη μεταβλητή ψευδής.

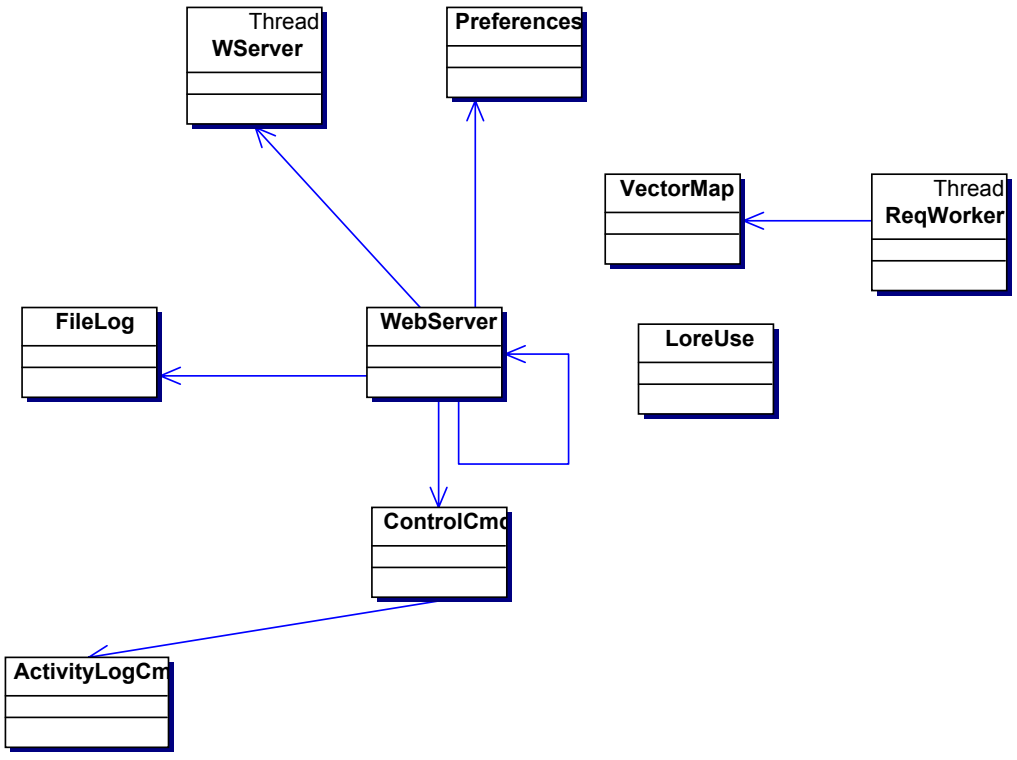
Με αυτόν τον τρόπο η εφαρμογή μπορεί να ελέγχει όλα τα εσωτερικά παράθυρα που δημιουργούνται αλλά συγχρόνως να μπορεί να ξεχωρίζει ποια από αυτά περιέχουν γράφους και να κάνουν τις ειδικές λειτουργίες και υπολογισμούς πάνω στους γράφους.

Πέρα από αυτά τώρα για την υποστήριξη δημιουργίας του ειδικού παραθύρου υποβολής επερωτήσεων (MQL Query) δημιουργήθηκε η κλάση *QueryConsole* η οποία είναι απόγονος της *MssDInternalFrame*.

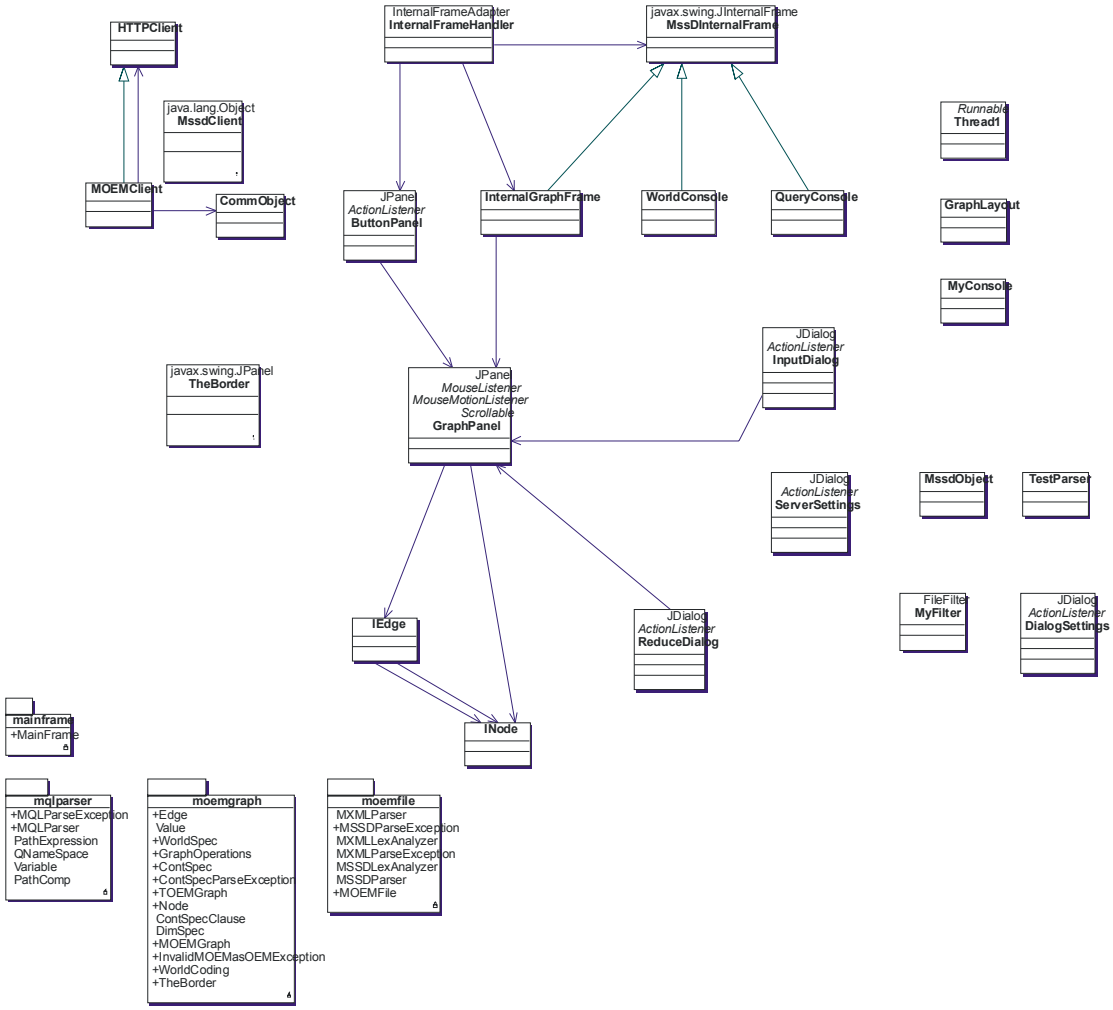
Τέλος, για την δημιουργία παραθύρων που περιέχουν μόνο κείμενο και τα οποία στην εφαρμογή χρησιμοποιούνται για την ανεξάρτητη παρουσίαση της κωδικοποίησης των κόσμων και των ισοδύναμων Lorel ερωτήσεων δημιουργήθηκε η κλάση *WorldConsole*.

4.5 Διαγράμματα Αρχιτεκτονικής Συστήματος

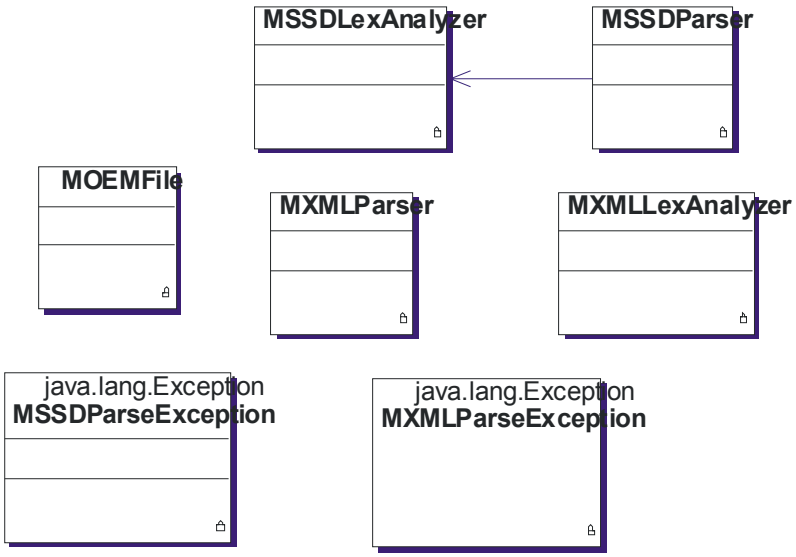
Στη συνέχεια παρουσιάζεται το διάγραμμα κλάσσεων του συνολικά του συστήματος που υλοποιήθηκε, τόσο στη μεριά του client όσο και στη μεριά του server.



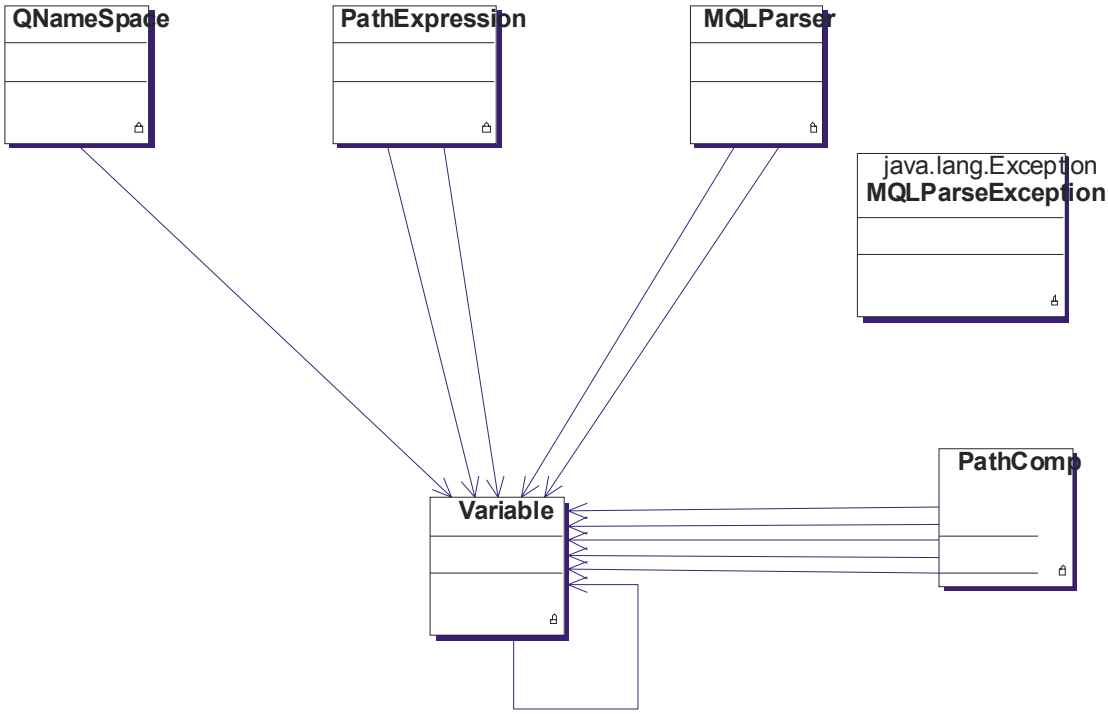
Διάγραμμα κλάσεων του MQLServer



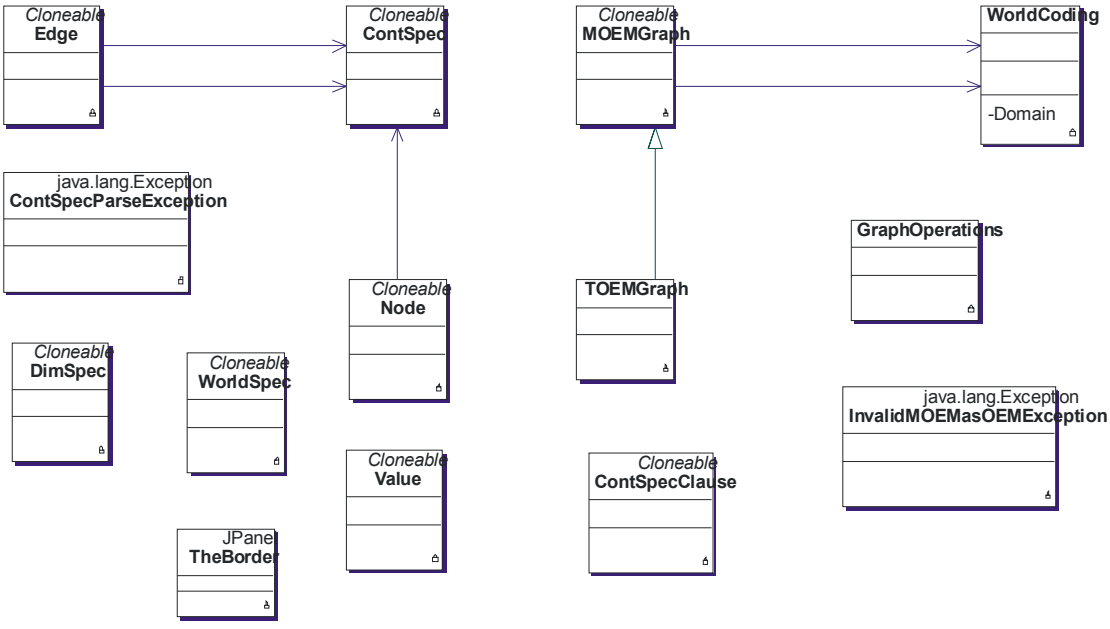
Διάγραμμα κλάσεων του MQLClient 1(πακέτο mssd)



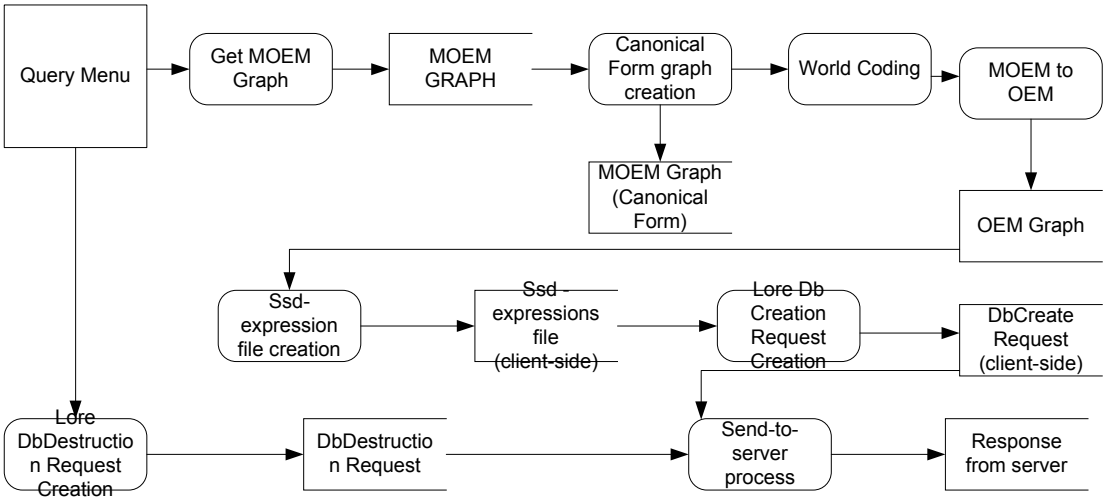
Διάγραμμα κλάσεων του MQLClient 2(πακέτο moemfile)



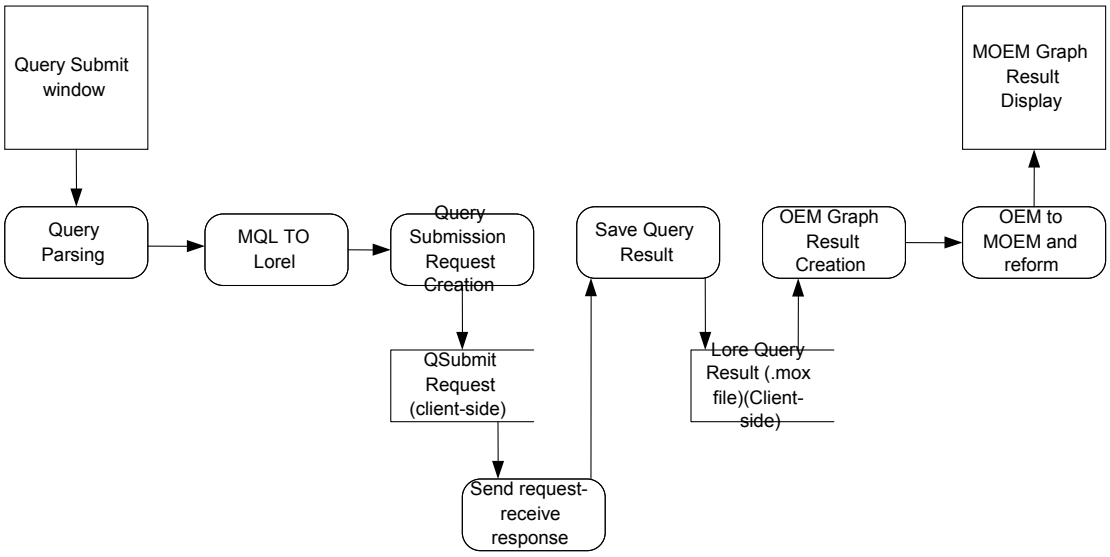
Διάγραμμα κλάσεων του MQLClient 3 (πακέτο MQLParser)



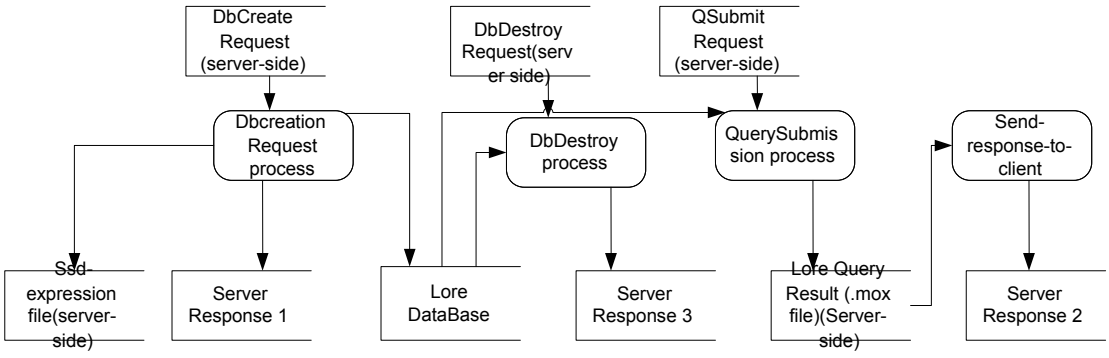
Διάγραμμα κλάσεων του MQLClient 4 (πακέτο moemgraph)



Διάγραμμα διαδικασιών στον MQLClient 1



Διάγραμμα διαδικασιών στον MQLClient 2



Διάγραμμα διαδικασιών στον MQLServer

4.6 Παρατηρήσεις

Το σύστημα επερωτήσεων που υλοποιήθηκε όπως περιγράψαμε παρουσιάζει ορισμένες ελλείψεις όσον αφορά στο σύνολο των διάφορων ερωτήσεων MQL τις οποίες απαντά με επιτυχία. Οι ελλείψεις αυτές, πέρα από τμήματα της MQL που παραλείψαμε από το σχεδιασμό ακόμα του συστήματος μετάφρασης, οφείλονται σε μεγάλο βαθμό και στα προβλήματα που παρουσιάζει η υλοποίηση της Lorel από το σύστημα Lore. Το γεγονός αυτό έχει ως αποτέλεσμα την αδυναμία από το σύστημα να χειριστεί ορισμένες μορφές ερωτήσεων MQL, οι οποίες υποστηρίζονται θεωρητικά από το σχήμα μετάφρασης που παρουσιάσαμε στο 3^{ο} κεφάλαιο αυτού του τόμου.

Συγκεκριμένα εντοπίσαμε κατά την ανάπτυξη της εφαρμογής τρία σημαντικά προβλήματα του Lore:

- Υπάρχει ένα όριο που έχει θέσει το Lore όσον αφορά στο μέγεθος των ερωτήσεων που μπορεί να απαντήσει. Επειδή αρκετές φορές οι ισοδύναμες Lorel ερωτήσεις που προκύπτουν από τη μετάφραση μιας MQL ερώτησης είναι αρκετά μεγάλες, το όριο αυτό ξεπερνιέται συχνά με αποτέλεσμα ορισμένες ερωτήσεις να μην εκτελούνται.
- Δεν υποστηρίζεται η δεσμευμένη λέξη “not” κατά την σύνταξη λογικών εκφράσεων στα πλαίσια ενός where clause. Αυτό σημαίνει ότι δεν μπορούμε να χρησιμοποιήσουμε την άρνηση ούτε και στα where και within clause της MQL, αφού λόγω του τρόπου μετάφρασής της σε Lorel, η λέξη αυτή αναγνωρίζεται από την MQL και περνά ως είναι στην ισοδύναμη Lorel ερώτηση.

- Παρουσιάζονται γενικά προβλήματα κατά τη χρησιμοποίηση των εκφράσεων “for all”, “exists” και “in”, οι οποίες εμφανίζονται στο where clause της Lorel και με τη βοήθεια των οποίων εκφράζουμε συγκρίσεις μεταξύ συνόλων. Το σύστημα μετάφρασης βασίζεται σε αυτές τις εκφράσεις της Lorel προκειμένου να μεταφέρει σε αυτήν το within clause της MQL. Οι λόγοι για τους οποίους παρουσιάζει το Lore αυτά τα προβλήματα είναι πολύ δύσκολο να εντοπιστούν με ακρίβεια, οπότε δεν μπορέσαμε να αποκλείσουμε από το σύνολο των ερωτήσεων MQL που υποστηρίξαμε τις ερωτήσεις εκείνες που θα έχουν πρόβλημα εκτέλεσης από το Lore.

5

Έλεγχος

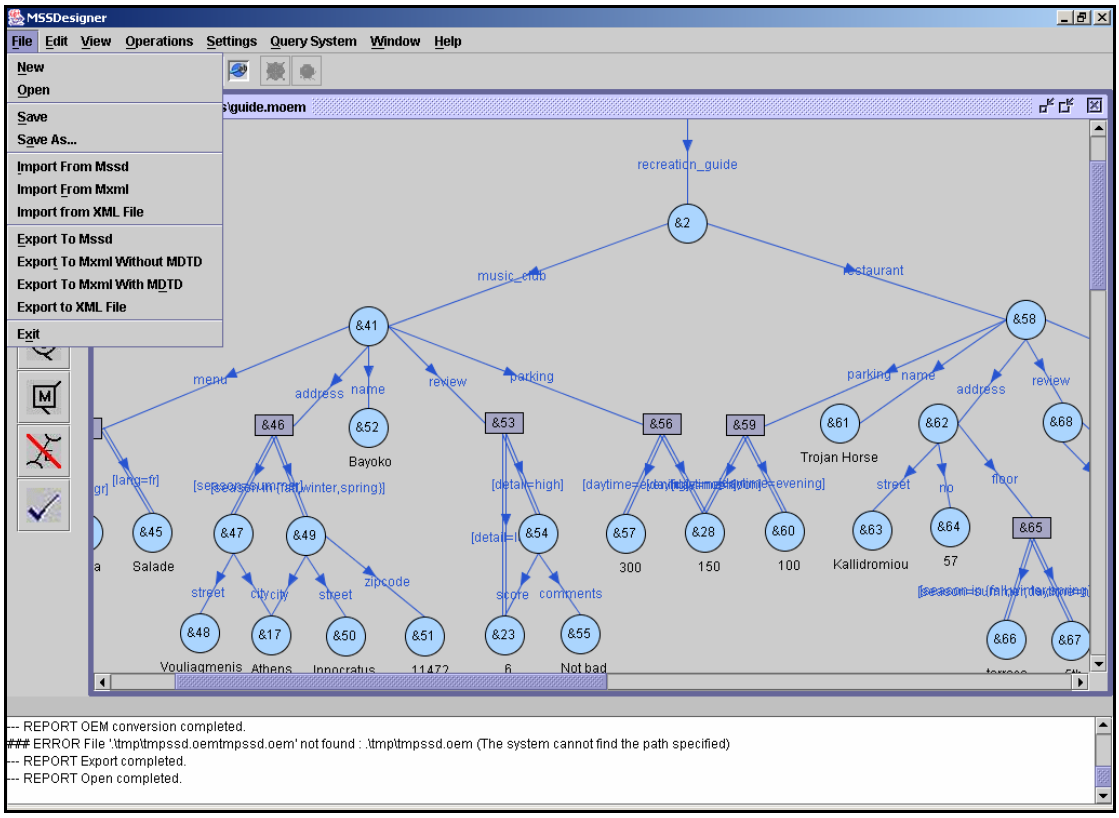
Στο κεφάλαιο αυτό θα παρουσιάσουμε μερικά παραδείγματα χρήσης του τελικού προγράμματος, με σκοπό να επιδείξουμε τα αποτελέσματα των διαφόρων δοκιμαστικών εφαρμογών που έγιναν με σκοπό τον έλεγχο της σωστής λειτουργίας του. Οι γράφοι που χρησιμοποιήσαμε στα παραδείγματα που θα ακολουθήσουν, καθώς και οι ερωτήσεις που υποβάλλαμε δοκιμαστικά στον καθένα, χαρακτηρίζονται από ιδιαιτερότητες που καθιστούν τις δοκιμές που πραγματοποιήθηκαν μη τετριμμένες περιπτώσεις χρήσεις της εφαρμογής, οι οποίες θα μπορούσαν να αποκαλύψουν αδυναμίες στο σχεδιασμό ή την υλοποίηση της.

Η εφαρμογή επίδειξης της εργασίας αυτής αποτελεί επέκταση της εφαρμογής MSSDesigner η οποία υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας [ΔΖ01]. Στην περιγραφή που θα ακολουθήσει αναφερόμαστε αρχικά στις βασικές λειτουργίες της εφαρμογής στην οποία προσθέσαμε το υποσύστημα επερωτήσεων, καθώς και στο υποσύστημα χρησιμοποίησης του MSSDServer ο οποίος πρέπει να είναι εγκατεστημένος σε κάποιο μηχάνημα με Linux ή Unix λειτουργικό και να είναι προσβάσιμος από το δίκτυο. Στη συνέχεια περιγράφουμε τις λειτουργίες της εφαρμογής που υλοποιήθηκαν στα πλαίσια της παρούσας διπλωματικής.

Πρέπει να διευκρινιστεί ότι παρέχονται δύο εκδόσεις της εφαρμογής. Η stand-alone έκδοση πρέπει να τρέχει σε λειτουργικό Linux/Unix αλλά δεν χρειάζεται δίκτυο, ενώ η client-server έκδοση τρέχει σε οποιοδήποτε λειτουργικό σύστημα αλλά πρέπει να τρέχει σε κάποιο Linux/Unix μηχάνημα του προσβάσιμου δικτύου η εφαρμογή MSSDServer.

5.1 Γενική περιγραφή

Παραθέτουμε τη βασική οθόνη που περιγράφει τη διαπροσωπεία με το χρήστη για να αποτελέσει οδηγό αναφοράς για το υπόλοιπο αυτού του κεφαλαίου.



Σχήμα 1: Το interface της εφαρμογής

Περιγραφή Menu

Το μενού της εφαρμογής αποτελείται από 7 υπομενού που το καθένα περιέχει ένα σετ λειτουργιών. Αυτά είναι από αριστερά προς τα δεξιά: File, Edit, View, Operations, Settings, Window και Help.

Το μενού *File* περιέχει τις ακόλουθες λειτουργίες :

- New
- Open
- Save
- SaveAs
- Import
- Export
- Exit

Επιλέγοντας *New* ο χρήστης δημιουργεί ένα νέο panel πάνω στο οποίο μπορεί να σχεδιάσει έναν γράφο. Με *Open* επιλέγει να φορτώσει στο περιβάλλον εργασίας ένα γράφο που έχει αποθηκευτεί σε αρχείο. Την αποθήκευση σε αρχείο την πραγματοποιεί με τις λειτουργίες *Save* και *SaveAs*. Το ζευγάρι των λειτουργιών *Import, Export* φορτώνει και αποτυπώνει ένα γράφο σε αρχείο με μορφή μιας *mssd-expression*. Σε μεταγενέστερα στάδια έχουν προστεθεί και άλλες λειτουργίες *import-export* για διάφορες μορφές αρχείων. Πιο συγκεκριμένα υπάρχει δυνατότητα δημιουργίας από και αποθήκευσης σε αρχεία και τύπου MXML καθώς και απλής εξαγωγής της πληροφορίας ενός MOEM γράφου σε αρχείο XML. Τέλος με *Exit* κλείνει η εφαρμογή.

Το μενού *Edit* αποτελείται από τις ακόλουθες λειτουργίες :

- Add Context Node
- Update Context Node
- Add MultiDim Node
- Add Edge
- Remove Edge

Η λειτουργία *Add Context Node* δημιουργεί έναν νέο απλό κόμβο (atomic ή complex), ενώ η *Add MultiDim Node* δημιουργεί έναν πολυδιάστατο κόμβο. Με την *Update Context Node* ενημερώνεται η τιμή ενός κόμβου και οι δύο τελευταίες λειτουργίες *Add Edge* και *Remove Edge* προσθέτουν και αφαιρούν μια ακμή αντίστοιχα.

Το μενού *View* περιέχει δύο λειτουργίες:

- Inherited Context
- Labels/Explicit Context

Η πρώτη εμφανίζει τα Inherited Context των ακμών και των κόμβων πάνω στο γράφο, ενώ η δεύτερη εμφανίζει είτε τις ετικέτες (για απλές ακμές) ή τα Explicit Context (για context ακμές).

Το μενού *Operations* περιλαμβάνει τέσσερις λειτουργίες με τις οποίες επεμβαίνει ο χρήστης πάνω στο γράφο:

- Check Graph Connectivity
- Calculate Inherited Context
- Check Graph Validity
- Reduce…
  - To OEM
  - Partial MOEM

Η πρώτη λειτουργία εκτελεί έναν έλεγχο συνεκτικότητας στο γράφο, απομακρύνοντας τους μη προσβάσιμους κόμβους. Η δεύτερη υπολογίζει τα Inherited Context των ακμών και των κόμβων του γράφου, σαρώνοντάς τον, ξεκινώντας από τη ρίζα. Η λειτουργία *Check Graph Validity* βρίσκει τις άκυρες ακμές και τις μαρκάρει με άλλο χρώμα. Τέλος η λειτουργία *Reduce* χωρίζεται σε δύο διαφορετικές λειτουργίες : η πρώτη απλοποιεί το γράφο σε έναν OEM γράφο κάτω από κάποιο κόσμο, ενώ η δεύτερη απλοποιεί το γράφο σε έναν MOEM γράφο κάτω από έναν context specifier που δεν περιγράφει απαραίτητα έναν κόσμο.

Το μενού *Settings* διαθέτει την επιλογή *Window Size,* μέσω της οποίας καθορίζεται το μήκος και το πλάτος του παραθύρου που περιέχει το panel στο οποίο σχεδιάζουμε το γράφο.

Το μενού *Window* διευκολύνει την περιήγηση του χρήστη ανάμεσα στα παράθυρα με τους γράφους που έχει ανοίξει.

Τέλος το μενού *Help* με την επιλογή *About* δίνει κάποιες πληροφορίες σχετικά με την έκδοση της εφαρμογής.

Περιγραφή Toolbar

Η Toolbar της εφαρμογής αποτελείται από επτά κουμπιά που υλοποιούν κάποιες από τις βασικότερες λειτουργίες. Οι λειτουργίες αυτές μπορούν να εκτελεστούν και από το μενού της εφαρμογής, απλά η toolbar προσφέρει έναν εύχρηστο τρόπο χρήσης του προγράμματος. Ακολουθούν περιγραφές της εργασίας που επιτελεί το κάθε κουμπί, αφού παρουσιάσουμε στο επόμενο σχήμα τη μορφή της εν λόγω toolbar:

Σχήμα 2: Η toolbar της εφαρμογής

Η πρώτη τριάδα κουμπιών αντιστοιχεί σε λειτουργίες του μενού *File*. Το πρώτο κουμπί είναι το αντίστοιχο του *New*, το δεύτερο αντιστοιχεί στο *Open* και το τρίτο στο *Save*. Ακολουθούν δύο λειτουργίες του μενού *Operations* και συγκεκριμένα ο έλεγχος συνεκτικότητας και η διαδικασία απλοποίησης. Η πρώτη αντιστοιχεί στο κουμπί με το tick και η δεύτερη στο κουμπί με την υδρόγειο. Το επόμενο ζευγάρι κουμπιών αντιστοιχεί στις δύο λειτουργίες του μενού *View* και για την ακρίβεια το πρώτο κουμπί εμφανίζει το Inherited Context των στοιχείων του γράφου (ακμές και κόμβοι), ενώ το δεύτερο το Explicit Context ή τις ετικέτες των ακμών, ανάλογα με το είδος της ακμής. Τέλος, τα τελευταία δύο κουμπιά είναι μεταγενέστερα από τα προηγούμενα και έχουν να κάνουν με λειτουργίες του συστήματος επερωτήσεων, οι οποίες θα διευκρινιστούν παρακάτω. Το πρώτο ανοίγει το παράθυρο υποβολής MQL επερωτήσεων και το δεύτερο το παράθυρο παρουσίασης της κωδικοποίησης κόσμων.

Περιγραφή του Panel Κουμπιών

Πρόκειται για ένα σύνολο επτά κουμπιών τοποθετημένων πάνω σε ένα panel που βρίσκεται στο αριστερό μέρος της οθόνης. Η κύρια λειτουργία του panel έχει να κάνει με το σχεδιασμό του εκάστοτε γράφου, δηλαδή με τη δημιουργία κόμβων, ακμών, κτλ. Ουσιαστικά αντιστοιχεί στις λειτουργίες του μενού *Edit*, με εξαίρεση το πρώτο και το τελευταίο κουμπί.

Σχήμα 3: Το panel των κουμπιών

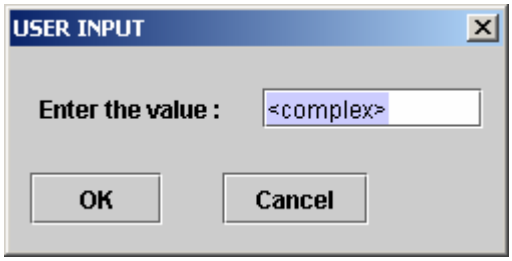
Αναλυτικότερα, το πρώτο κουμπί είναι αυτό που είναι πατημένο από default και όταν είναι πατημένο επιτρέπει το μαρκάρισμα ενός κόμβου και τη μετακίνησή του με τη βοήθεια του ποντικιού (drag) ή εναλλακτικά το μαρκάρισμα ενός συνόλου κόμβων και τη μετακίνηση αυτών με τον ίδιο τρόπο. Το δεύτερο κουμπί όταν είναι πατημένο επιτρέπει τη σχεδίαση ακμών από έναν κόμβο προς έναν άλλο πατώντας με το ποντίκι πάνω στον πρώτο και έχοντας το κουμπί πατημένο ώσπου να φτάσει στο δεύτερο κόμβο όπου και ελευθερώνεται. Τα δύο πρώτα κουμπιά, σε αντίθεση με τα υπόλοιπα, λειτουργούν σαν toggle buttons, δηλ. όταν πατιέται το ένα ελευθερώνεται το άλλο. Σημειώνουμε εδώ ότι όταν ο χρήστης πατήσει το δεύτερο κουμπί, μπορεί να προσθέτει με το ποντίκι ακμές κατά βούληση, με τον τρόπο που περιγράψαμε.

Τα υπόλοιπα πέντε κουμπιά διαχωρίζονται από τα δύο πρώτα επειδή οι λειτουργίες που επιτελούν συμβαίνουν μόνο όταν πατηθεί το αντίστοιχο πλήκτρο. Για παράδειγμα, για να προσθέσει ο χρήστης τρεις πολυδιάστατους κόμβους, θα πρέπει να πατήσει το κουμπί με το γράμμα M τρεις φορές. Έτσι το κουμπί με το γράμμα C δημιουργεί έναν context κόμβο, το επόμενο ενημερώνει την τιμή του κόμβου που έχει επιλεχθεί, το κουμπί με το M φτιάχνει έναν πολυδιάστατο κόμβο και το τέταρτο κουμπί απομακρύνει (διαγράφει) μια επιλεγμένη ακμή. Το τελευταίο κουμπί ελέγχει τη συνεκτικότητα του γράφου και απομακρύνει τους κόμβους που δεν είναι προσβάσιμοι από τη ρίζα.

5.2 Επεξεργασία γράφων

Προσθήκη ενός context κόμβου

Μπορεί να γίνει με δύο τρόπους: από το μενού πηγαίνοντας στο "Edit" και μετά "Add Context Node" ή από το panel των κουμπιών πατώντας το τρίτο κουμπί. Θα εμφανιστεί ένας διάλογος σαν αυτόν που φαίνεται στο Σχήμα 4, που θα επιτρέπει στο χρήστη να εισάγει μια τιμή για τον κόμβο, αν πρόκειται να χρησιμοποιηθεί σαν atomic. Αν ο κόμβος προορίζεται για complex, αρκεί ο χρήστης να πατήσει OK.



Σχήμα 4: Ο διάλογος προσθήκης νέου context κόμβου

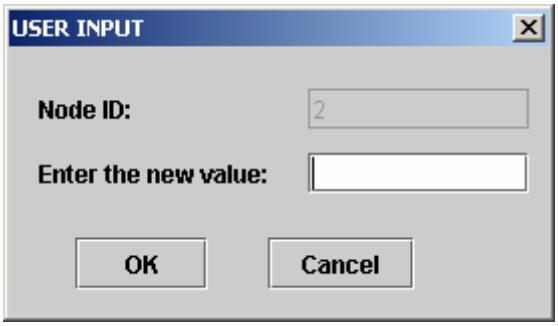
Προσθήκη ενός πολυδιάστατου κόμβου

Πατώντας απλά το κουμπί με το γράμμα M του panel αριστερά στην οθόνη ή από το μενού με την επιλογή "Add MultiDim Node" του μενού "Edit". Δε χρειάζεται καμιά επιπλέον εργασία από το χρήστη και ο κόμβος εμφανίζεται αμέσως στην οθόνη.

Ενημέρωση ενός context κόμβου

Και εδώ είναι δυνατή η λειτουργία αυτή με δύο τρόπους: από το μενού "Edit" με την επιλογή "Update Context Node" ή από το panel των κουμπιών με το τέταρτο κουμπί που έχει έναν κόμβο και ένα κόκκινο βέλος που δείχνει μέσα σε αυτόν. Με την επιλογή ενημέρωσης κόμβου εμφανίζεται ένας διάλογος με δύο πεδία: το αναγνωριστικό (id) του κόμβου και την τιμή που θα εισαχθεί (Σχήμα 5). Το πεδίο με το id μπορεί να είναι ενεργοποιημένο, οπότε προτρέπει το χρήστη να καθορίσει τον κόμβο στον οποίο αναφέρεται ή μπορεί να είναι απενεργοποιημένο. Η δεύτερη περίπτωση μπορεί να συμβεί εάν ο χρήστης έχει επιλέξει με το ποντίκι έναν κόμβο (μαρκάροντάς τον) πριν επιλέξει την ενημέρωση κόμβου. Σε αυτή την περίπτωση το πεδίο με το id είναι απενεργοποιημένο και αναγράφεται το αναγνωριστικό του επιλεγμένου κόμβου.

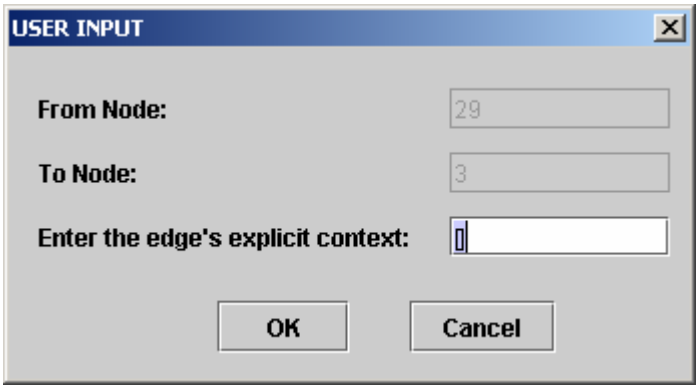
Αυτή ακριβώς η περίπτωση απεικονίζεται στο Σχήμα 5.



Σχήμα 5: Ο διάλογος ενημέρωσης της τιμής ενός κόμβου

Προσθήκη ακμής

Για να προστεθεί μία νέα ακμή πρέπει αρχικά να καθοριστούν οι κόμβοι εκκίνησης και προορισμού. Αυτό μπορεί να γίνει με δύο τρόπους. Από το μενού επιλέγοντας "Edit" και μετά "Add Edge" θα εμφανιστεί ένας διάλογος (Σχήμα 6) με τρία πεδία όπου ο χρήστης καλείται να δηλώσει τους δύο κόμβους καθώς και την ετικέτα ή το explicit context της ακμής, ανάλογα με το αν πρόκειται για απλή ή context ακμή. Εναλλακτικά πατώντας το δεύτερο κουμπί του panel και στη συνέχεια τραβώντας μια γραμμή από τον κόμβο εκκίνησης στον κόμβο προορισμού. Σε αυτή την περίπτωση ο χρήστης καλείται να συμπληρώσει μόνο το πεδίο που αφορά την ετικέτα/explicit context της ακμής. Σημειώνεται εδώ ότι το είδος της ακμής (απλή ή context) καθορίζεται αυτόματα από το πρόγραμμα, ανάλογα με το είδος του κόμβου εκκίνησης. Αν πρόκειται για πολυδιάστατο, η ακμή θα είναι context, διαφορετικά θα είναι απλή.

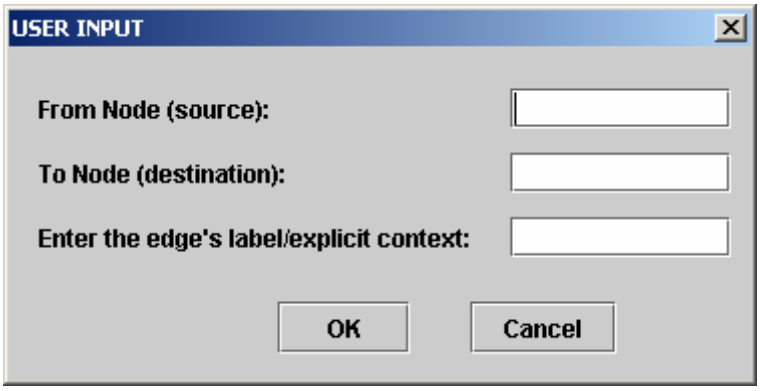


Σχήμα 6: Ο διάλογος δημιουργίας μιας ακμής

Διαγραφή ακμής

Η διαγραφή μιας ακμής λειτουργεί περίπου σαν την προσθήκη μιας νέας ακμής. Και εδώ ο χρήστης μπορεί να επιλέξει μια ακμή και στη συνέχεια να πατήσει το κουμπί του panel που διαγράφει ακμές (έκτο από πάνω προς τα κάτω) ή να χρησιμοποιήσει το μενού "Edit" και την επιλογή "Remove Edge", οπότε εμφανίζεται ο διάλογος του σχήματος 7 με τα πεδία που περιγράψαμε και παραπάνω.

Για ευκολία και πληρότητα παρατίθεται και το ακόλουθο σχήμα:

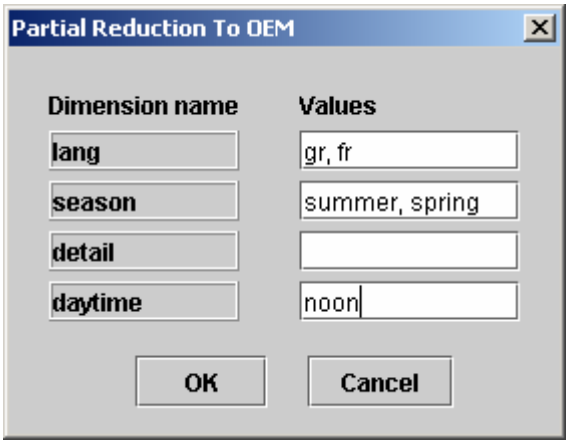


Σχήμα 7: Ο διάλογος διαγραφής μιας ακμής

5.3 Επιπλέον λειτουργίες

Απλοποίηση ενός Γράφου

Κατά την απλοποίηση ενός γράφου (επιλογή "Reduce To…" του μενού "Operations" ή από το αντίστοιχο κουμπί της toolbar) εκτελείται μια διαδικασία εύρεσης όλων των διαστάσεων κάτω από τις οποίες έχει νόημα ο γράφος. Στη συνέχεια εμφανίζεται ένας διάλογος σαν αυτόν του σχήματος 8 στην οθόνη, όπου ο χρήστης καλείται να συμπληρώσει πλάι σε κάθε διάσταση την τιμή ή τις τιμές για τις οποίες επιθυμεί να πάρει το στιγμιότυπο.



Σχήμα 8: Ο διάλογος του Partial Reduce

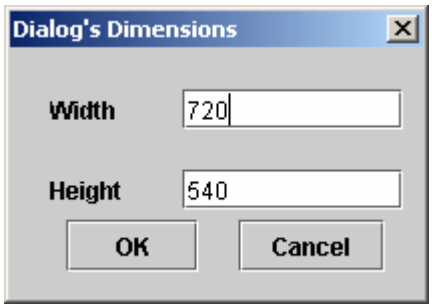
Σημειώνεται ότι μια τιμή για μια διάσταση μπορεί:

- να μείνει κενή, δηλ. να μην καθοριστεί τιμή για αυτήν (περίπτωση Partial MOEM)
- να συμπληρωθεί με μία τιμή (αν συμβεί για όλες τις διαστάσεις έχουμε την περίπτωση Reduce To OEM, αφού έτσι καθορίζεται ένας κόσμος)
- να συμπληρωθεί με ένα πλήθος τιμών χωρισμένων με κόμματα. (και εδώ περίπτωση Partial MOEM)

Στο Σχήμα 8 απεικονίζονται οι διαστάσεις ενός MOEM γράφου: lang, season, detail, daytime. Προσέξτε ότι ο χρήστης έχει συμπληρώσει στο πεδίο lang δύο γλώσσες: gr, fr (ελληνικά, γαλλικά), όπως και στο season (summer, spring), έχει αφήσει το πεδίο detail κενό και έχει εισάγει στο daytime μία μόνο τιμή: noon.

Ρύθμιση του μεγέθους του panel

Με την επιλογή του μενού "Settings", "Window Size" ο χρήστης μπορεί να καθορίσει τις διαστάσεις κάθε νέου panel που δημιουργεί. Συγκεκριμένα εμφανίζεται στην οθόνη ο διάλογος του σχήματος 9 με προεπιλεγμένες τις τιμές που ισχύουν εκείνη τη στιγμή για το μέγεθος του panel. Στην περίπτωση που περιγράφουμε στο σχήμα, το πλάτος είναι 720 και το ύψος 420. Σημειώνεται ότι πολύ μεγάλες τιμές πλάτους ή ύψους (π.χ. μεγαλύτερες από την ανάλυση της οθόνης) δε λαμβάνονται υπόψη και τίθενται αυτόματα στις μέγιστες επιτρεπτές τιμές, όπως αυτές έχουν οριστεί από την εφαρμογή.

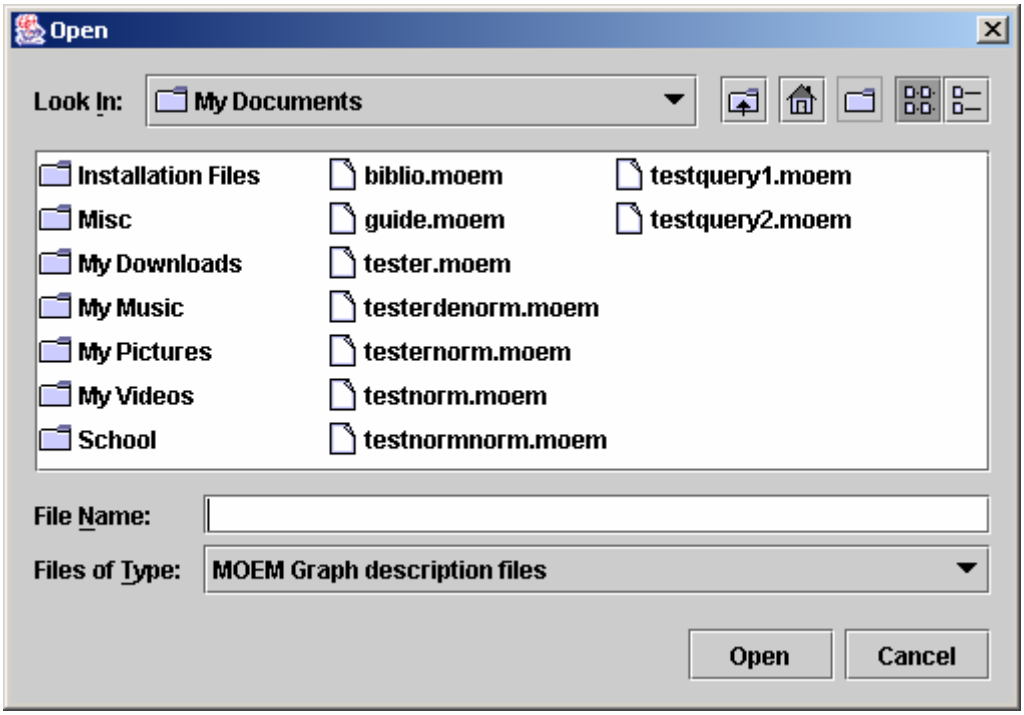


Σχήμα 9 : Ο διάλογος των διαστάσεων του panel

Open/Save

Οι λειτουργίες αυτές έχουν να κάνουν με το άνοιγμα και το σώσιμο ενός γράφου σε αρχείο με τέτοια μορφή ώστε να κρατούνται οι ακριβείς θέσεις των στοιχείων του γράφου. Μπορούν να κληθούν τόσο από το μενού "File", όσο και από την toolbar (δεύτερο και τρίτο κουμπί).

Παραθέτουμε ενδεικτικά το επόμενο σχήμα που παρουσιάζει το διάλογο στην περίπτωση open:

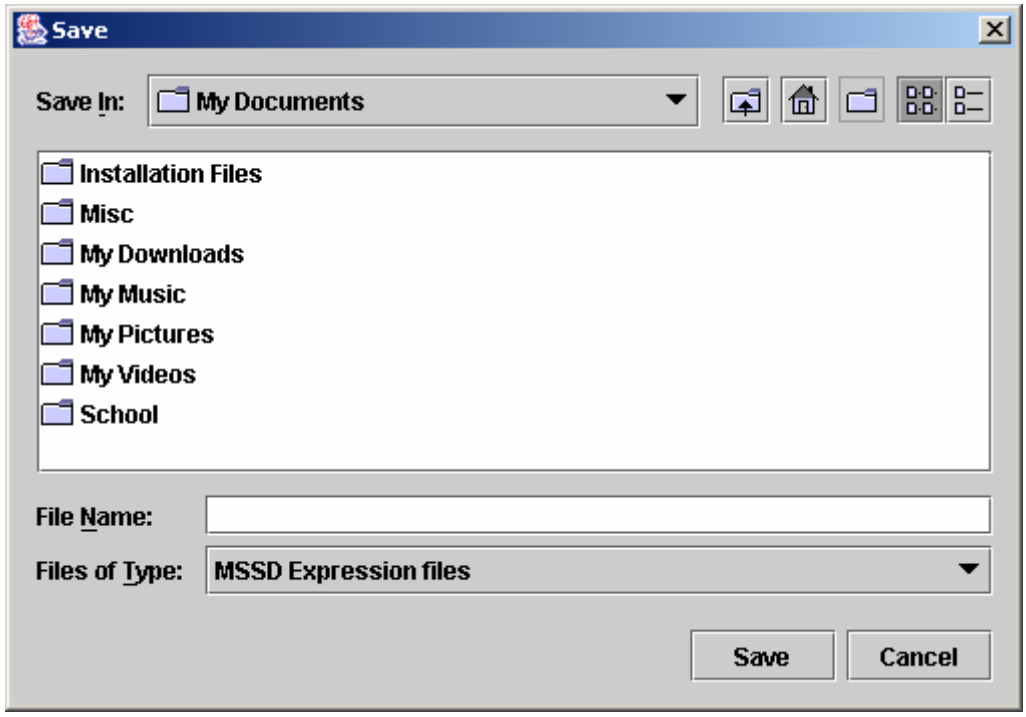


Σχήμα 10 : Ο διάλογος του Open

Import/Export

Και σε αυτήν την περίπτωση έχουμε άνοιγμα αρχείου και αποτύπωση σε αρχείο, αλλά σε μορφή *mssd-expression*. Μπορούν να κληθούν μόνο από το μενού "File", "Import" ή "Export" και εμφανίζουν έναν διάλογο όμοιο με αυτό της περίπτωσης open/save.

Ενδεικτικά παρουσιάζουμε τη μορφή ενός τέτοιου διαλόγου στην περίπτωση του Export.

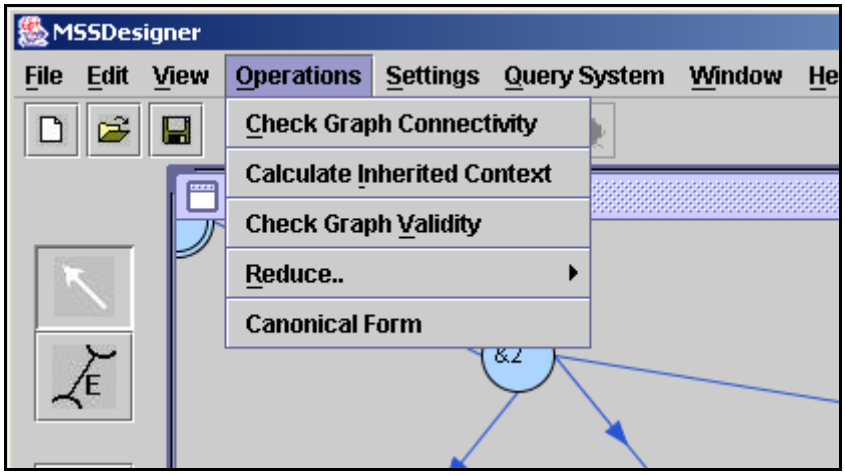


Σχήμα 11 : Ο διάλογος του Export

5.4 Νέες λειτουργίες – το σύστημα επερωτήσεων

Μετά το άνοιγμα της εφαρμογής, προκειμένου να χρησιμοποιήσουμε το σύστημα επερωτήσεων χρειάζεται να φορτώσουμε στην εφαρμογή έναν MOEM γράφο. Αυτό μπορεί να γίνει είτε με δημιουργία από την αρχή ενός νέου γράφου χρησιμοποιώντας τα εργαλεία κατασκευής γράφου που παρέχονται, είτε με άνοιγμα ενός αρχείου. Τα αρχεία που μπορούν να διαβαστούν από την εφαρμογή είναι τύπου MOEM, MSSD ή MOX και έχουν ονόματα με αντίστοιχες επεκτάσεις.

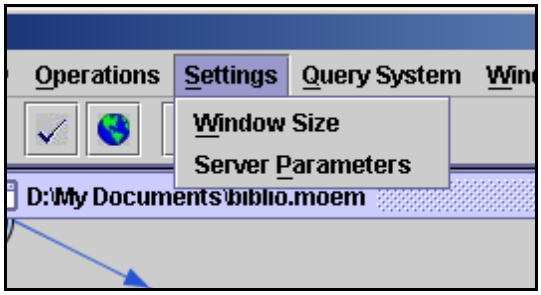
Το μενού *Operations* της αρχικής εφαρμογής επεκτάθηκε με μία επιπλέον λειτουργία, αυτή της κατασκευής της κανονικοποιημένης μορφής του ενεργού γράφου (*Canonical Form*). Η μορφή του μενού αυτού είναι πλέον η εξής:



Σχήμα 13: Το μενού Operations

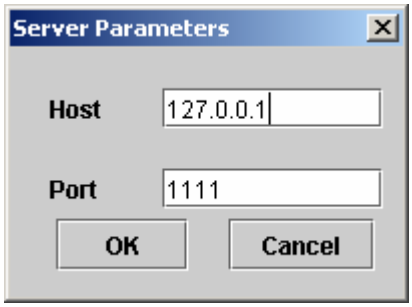
Στην περίπτωση της client-server έκδοσης στο μενού *Settings*, εκτός από την επιλογή ρύθμισης του μεγέθους των παραθύρων της επιφάνειας εργασίας προστέθηκε και η λειτουργία προσδιορισμού των παραμέτρων του MSSDServer, η οποία είναι απαραίτητη για την ορθή λειτουργία του υποσυστήματος ερωτήσεων.

Η μορφή του μενού αυτού είναι η εξής:



Σχήμα 14: Το μενού Settings

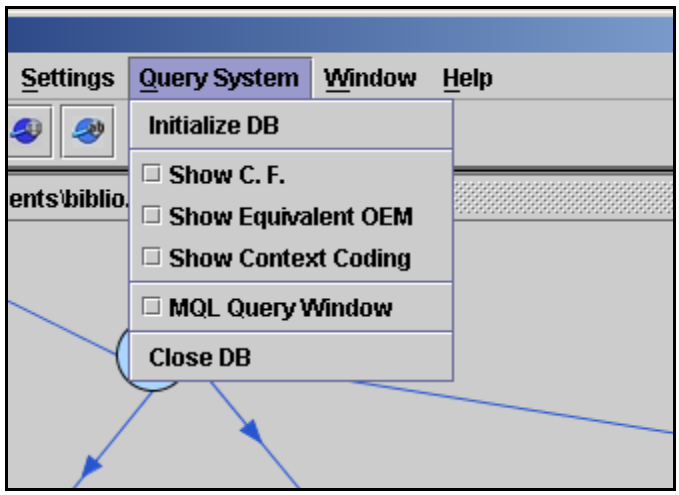
Το παράθυρο προσδιορισμού των παραμέτρων του MSSDServer παρουσιάζεται στην παρακάτω εικόνα:



Σχήμα 15: Server Parameters

Στο συγκεκριμένο παράθυρο προσδιορίζεται το όνομα (ή IP διεύθυνση) του host στο οποίο τρέχει ο MSSDServer καθώς και η θύρα του μηχανήματος στην οποία ο διακομιστής αναμένει για κλήσεις.

Η υποστήριξη επερωτήσεων στον ενεργό γράφο που είναι φορτωμένος στην εφαρμογή παρέχεται από την εφαρμογή μέσω του νέου μενού με όνομα *Query System*.

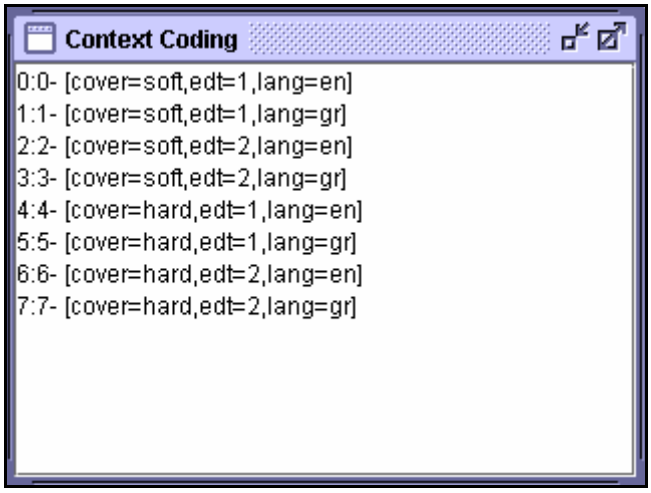


Σχήμα 16: Το μενού Query

Αρχικά η μόνη δυνατή επιλογή από το παραπάνω μενού είναι η πρώτη, *Initialize DB*, η οποία αναλαμβάνει να αρχικοποιήσει το σύστημα επερωτήσεων. Συγκεκριμένα, επιλέγοντάς την η εφαρμογή ελέγχει τη συνεκτικότητα του ενεργού γράφου και αφαιρεί αυτόματα τυχόν κόμβους που είναι απροσπέλαστοι από τη ρίζα του και στη συνέχεια υπολογίζει το inherited context του. Έπειτα, αφού παράγει την κανονική μορφή του, τον μεταφέρει στην αντίστοιχη μοντελοποιημένη μορφή OEM. Εφόσον δεν υπάρξει απρόβλεπτο πρόβλημα στις πιο πάνω διαδικασίες, ενεργοποιούνται και οι υπόλοιπες επιλογές του μενού *Query*.

Η επιλογές *Show C.F.* και *Show Equivalent OEM* φέρνουν στο προσκήνιο της εφαρμογής αντίστοιχα τον κανονικοποιημένο γράφο και την ισοδύναμη OEM μορφή του. Τα παράθυρα των γράφων αυτών είναι ήδη φορτωμένα από την εφαρμογή, βρίσκονται όμως τοποθετημένα πίσω από τον αρχικό γράφο.

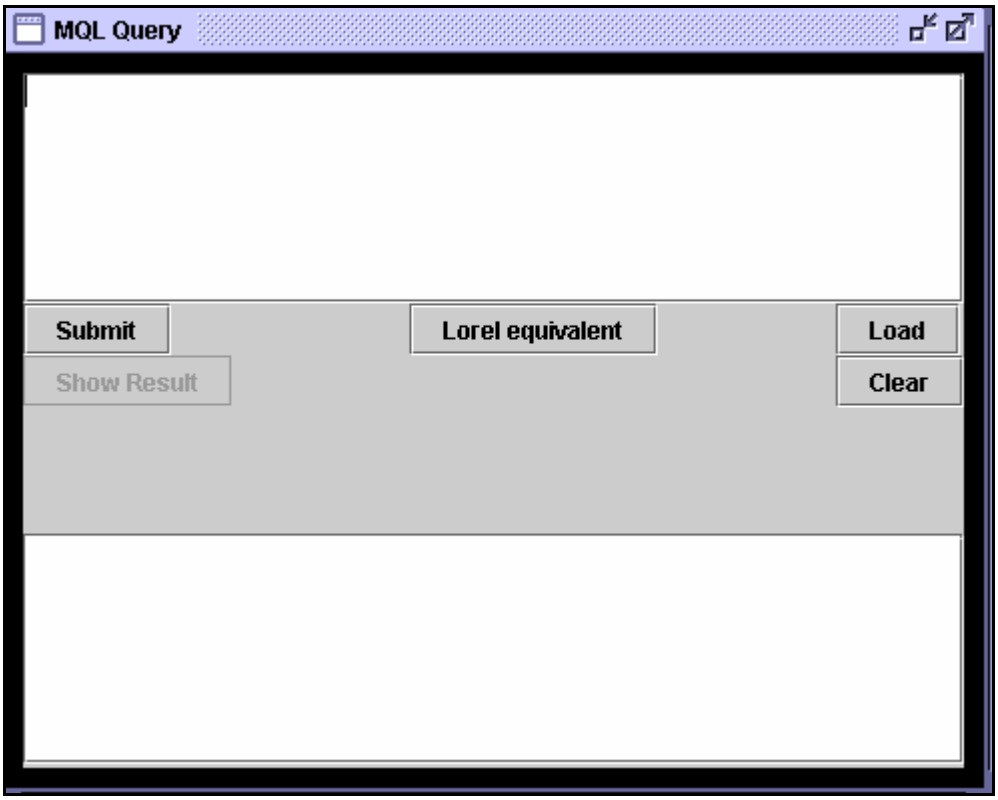
Η επιλογή *Show Context Coding* ανοίγει νέο παράθυρο όπου παρουσιάζεται η κωδικοποίηση κάθε κόσμου που ανήκει στο οικουμενικό context του γράφου. Η μορφή αυτού του παραθύρου είναι η εξής:



Σχήμα 17: Το παράθυρο κωδικοποίησης κόσμων

Με την επιλογή *Close DB* τελούνται μια σειρά από ενέργειες προκειμένου απενεργοποιηθεί το σύστημα επερωτήσεων. Σε αυτές περιλαμβάνονται η αποσύνδεση της εφαρμογής από τον server και το κλείσιμο όλων των παραθύρων που ανοίχτηκαν κατά την υποβολή ερωτήσεων, εκτός από τον αρχικό γράφο και από τους γράφους που παρουσιάζουν αποτελέσματα των ερωτήσεων που υποβλήθηκαν στο σύστημα.

Η επιλογή *MQL Query Window* ανοίγει ένα εσωτερικό παράθυρο μέσω του οποίου είναι δυνατή η υποβολή ερωτήσεων στο σύστημα. Η μορφή του είναι η εξής:

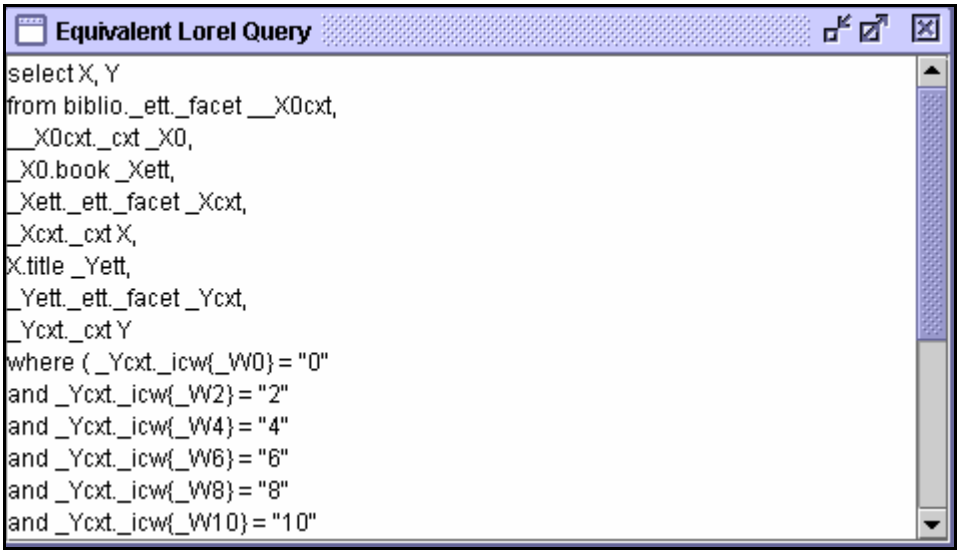


Σχήμα 18: Το παράθυρο υποβολής ερωτήσεων

Στο πάνω πεδίο κειμένου ο χρήστης καλείται να πληκτρολογήσει την ερώτηση MQL την οποία θέλει να υποβάλλει. Αντί να πληκτρολογήσει την ερώτηση, δίνεται η δυνατότητα να την φορτώσει από ένα αρχείο κειμένου επιλέγοντας το πλήκτρο "Load". Και στις δύο περιπτώσεις απαιτείται η ερώτηση να τελειώνει με ερωτηματικό (semicolon, ;). Εδώ πρέπει να διευκρινιστεί ότι προς το παρόν παρέχεται η δυνατότητα αποθήκευσης μόνο μίας ερώτησης ανά αρχείο, αφού και το παράθυρο υποβολής ερωτήσεων μπορεί αντίστοιχα να επεξεργαστεί μόνο μία ερώτηση κάθε φορά.

Η υποβολή της ερώτησης γίνεται με το πλήκτρο "Submit". Η εφαρμογή μεταφράζει την ερώτηση MQL στην αντίστοιχη Lorel και στη συνέχεια επικοινωνεί με τον server προκειμένου να λάβει την απάντηση σε αυτή. Είναι δυνατόν να γίνει απλώς η μετάφραση, χωρίς να σταλθεί η ερώτηση στο server. Αυτό γίνεται με τη βοήθεια του πλήκτρου "Lorel equivalent". Και στις δύο περιπτώσεις η ισοδύναμη Lorel ερώτηση προβάλλεται στο κάτω πεδίο.

Σε περίπτωση που η ερώτηση Lorel που προκύπτει είναι μεγάλη, με απλό κλικ στο κάτω πεδίο ο χρήστης μπορεί να την προβάλλει σε ένα νέο παράθυρο, όπως στο παρακάτω παράδειγμα:



Σχήμα 19: Παράθυρο προβολής MQL ερώτησης σε Lorel.

Εφόσον η υποβολή της ερώτησης είναι επιτυχής, ενεργοποιείται στο παράθυρο υποβολής το κουμπί "Show Result". Με επιλογή αυτού η εφαρμογή ανοίγει ένα νέο παράθυρο με το γράφο – απάντηση στην αντίστοιχη ερώτηση.

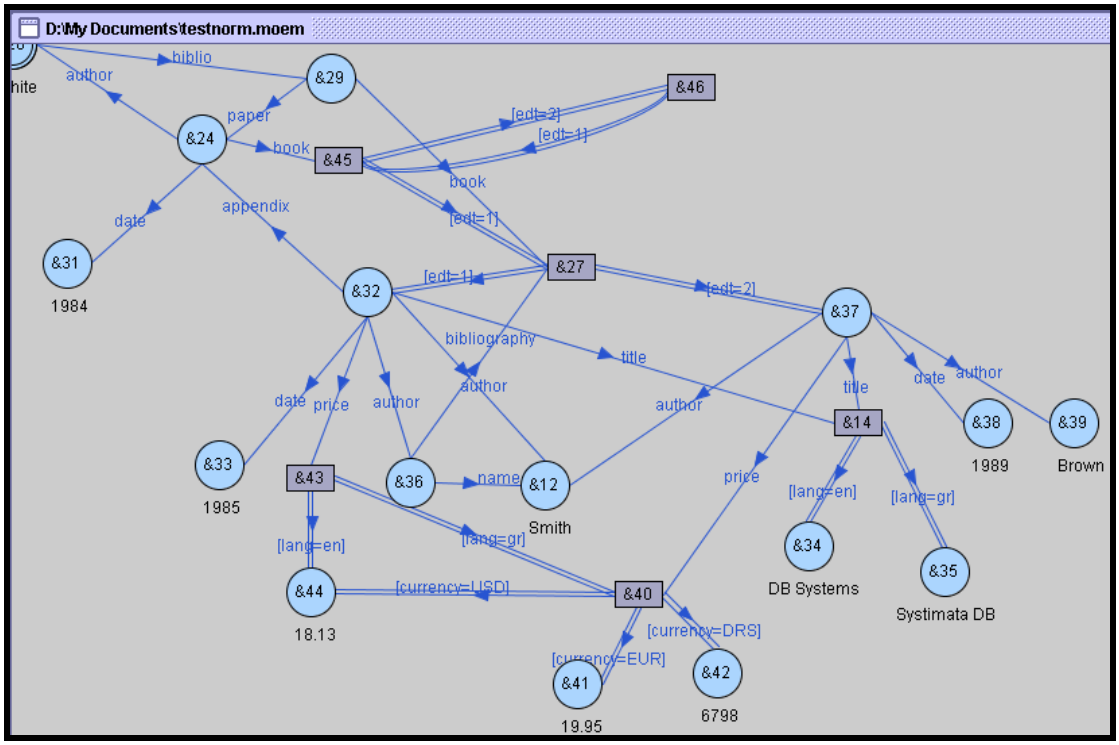
Ο χρήστης μπορεί να χειριστεί την απάντηση όπως οποιονδήποτε MOEM γράφο, δηλαδή μπορεί να την σώσει σε αρχείο, να την αλλάξει, να εφαρμόσει τους διαθέσιμους από την εφαρμογή μετασχηματισμούς, ακόμα και να υποβάλλει εκ νέου ερωτήσεις σε αυτήν, αφού φυσικά πρώτα κλείσει το σύστημα επερωτήσεων για τον αρχικό γράφο.

5.5 Έλεγχος αλγορίθμων μετασχηματισμών

Όπως περιγράφηκε πιο πάνω, στα πλαίσια της μεταφοράς ενός MOEM γράφου στο σύστημα Lore προς υποβολή ερωτήσεων σε αυτόν, τελείται μια σειρά μετασχηματισμών πάνω στο γράφο αυτό. Θα παρουσιάσουμε εδώ το αποτέλεσμα των μετασχηματισμών αυτών σε γράφους όπου εμφανίζονται διάφορες ιδιαιτερότητες που οι αλγόριθμοι μετασχηματισμού χρειάζεται να χειριστούν κατάλληλα.

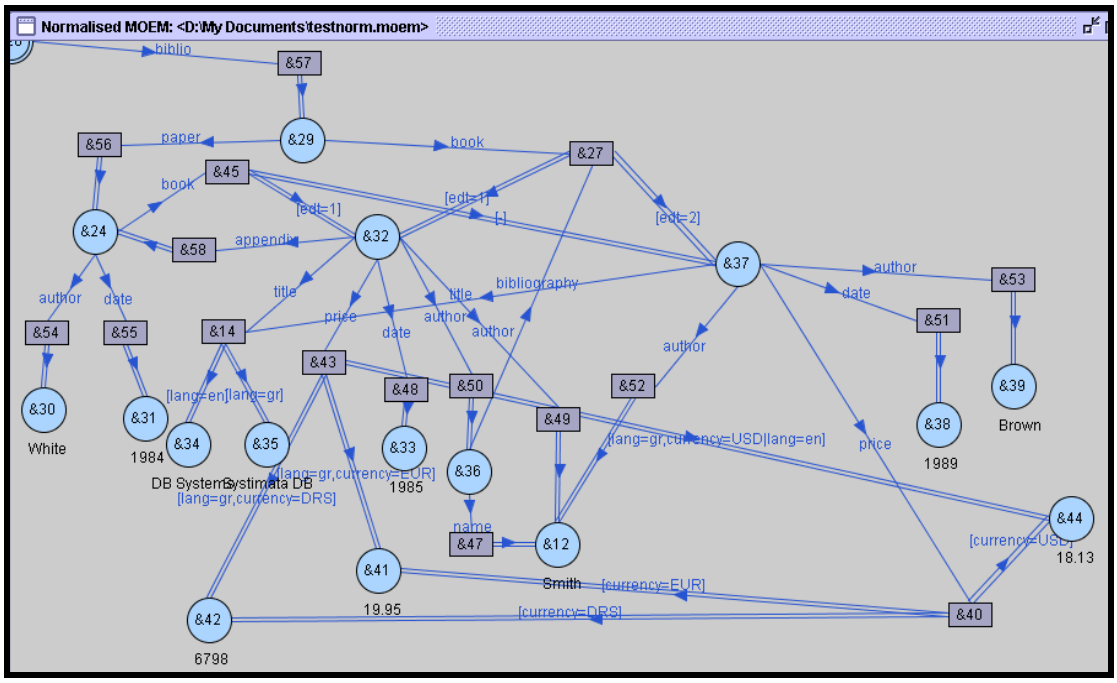
5.5.1 Κανονικοποίηση γράφου με ανωμαλίες

Ο επόμενος γράφος παρουσιάζει συνηθισμένους σχηματισμούς που εξαλείφονται κατά τη διαδικασία της κανονικοποίησης, όπως context ακμές που καταλήγουν σε πολυδιάστατους κόμβους. Επιπλέον, περιλαμβάνει και κύκλους, οπότε η κανονικοποίησή του έχει ιδιαίτερο ενδιαφέρον:



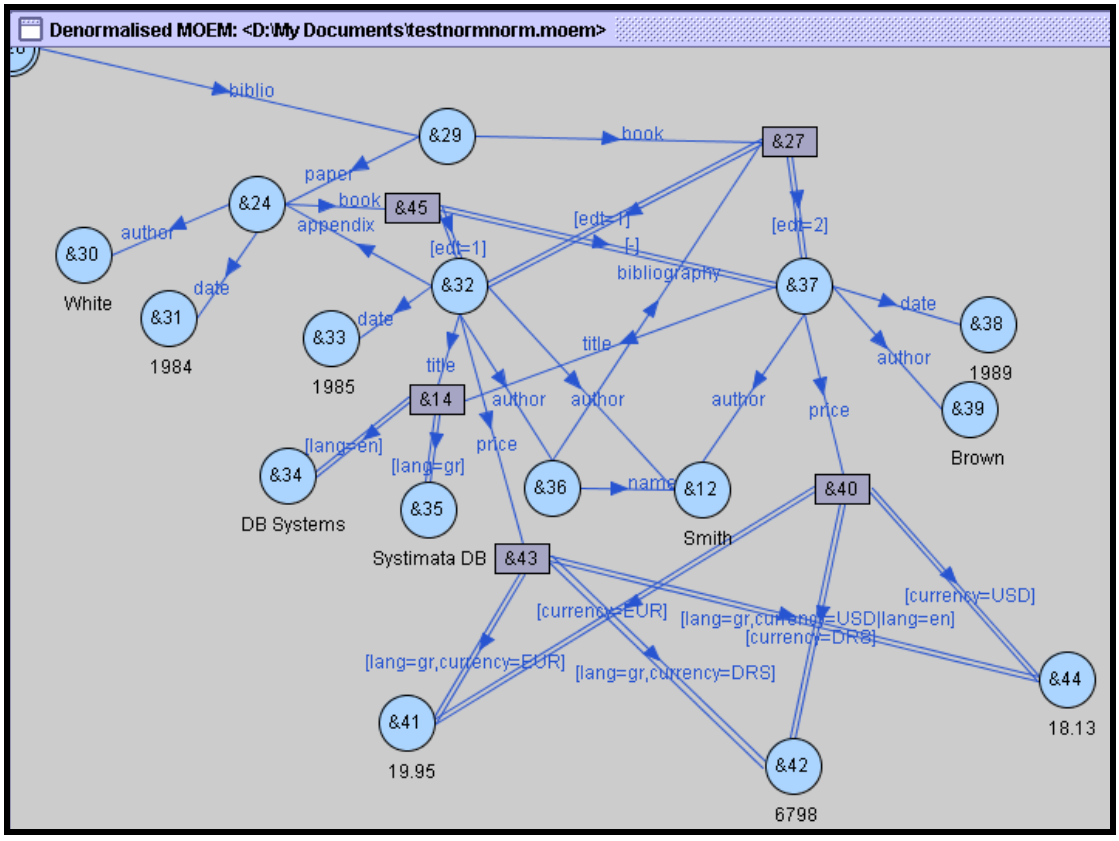
Σχήμα 20: MOEM γράφος με κύκλους

Η κανονική μορφή του προέκυψε από την επιλογή της εντολής “Operations→Canonical Form” και είναι η εξής:



Σχήμα 21: Κανονικοποιημένη μορφή γράφου με κύκλους

Αξίζει να παρουσιάσουμε την απλοποιημένη μορφή του τελευταίου γράφου, η οποία περιμένουμε να μοιάζει με την αρχική του μορφή, απαλλαγμένη από διάφορες ανωμαλίες.

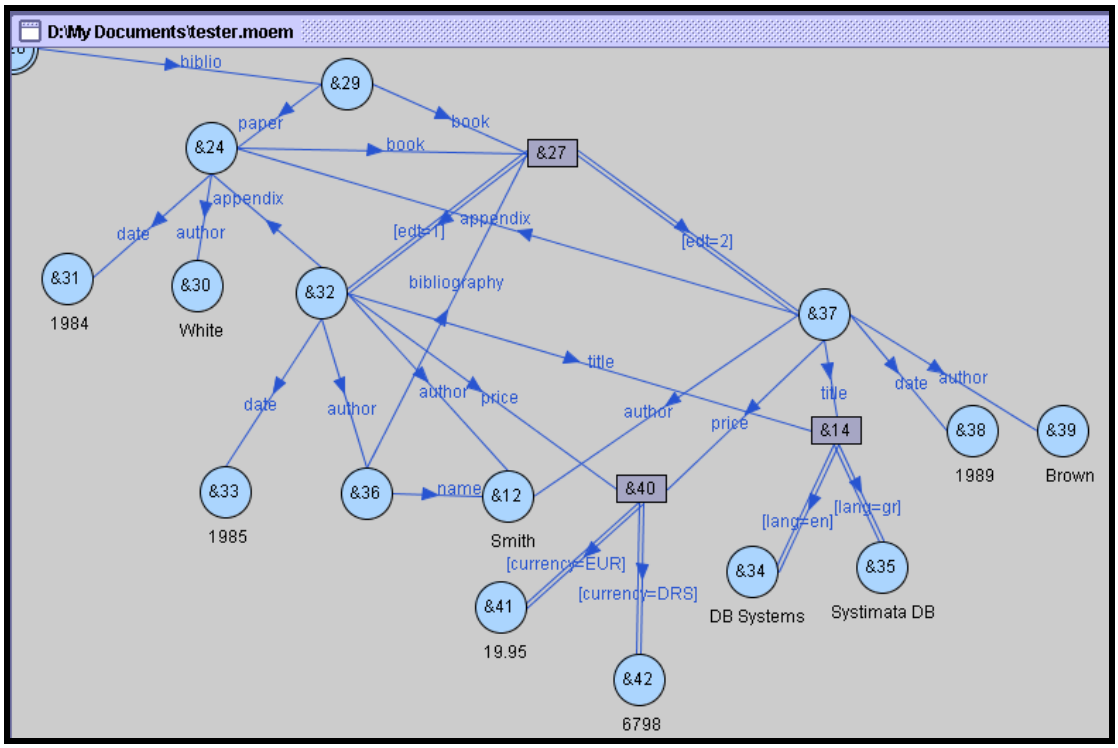


Σχήμα 22: Απλοποιημένη μορφή γράφου με κύκλους

Συγκρίνοντας λοιπόν τον παραπάνω γράφο με τον αρχικό γράφο, λείπει από αυτόν ο πολυδιάστατος κόμβος με αναγνωριστικό «46», επειδή δεν καταλήγει σε αυτόν καμία entity ακμή, οπότε δεν προσφέρει τίποτε παραπάνω στην πληροφορία του γράφου.

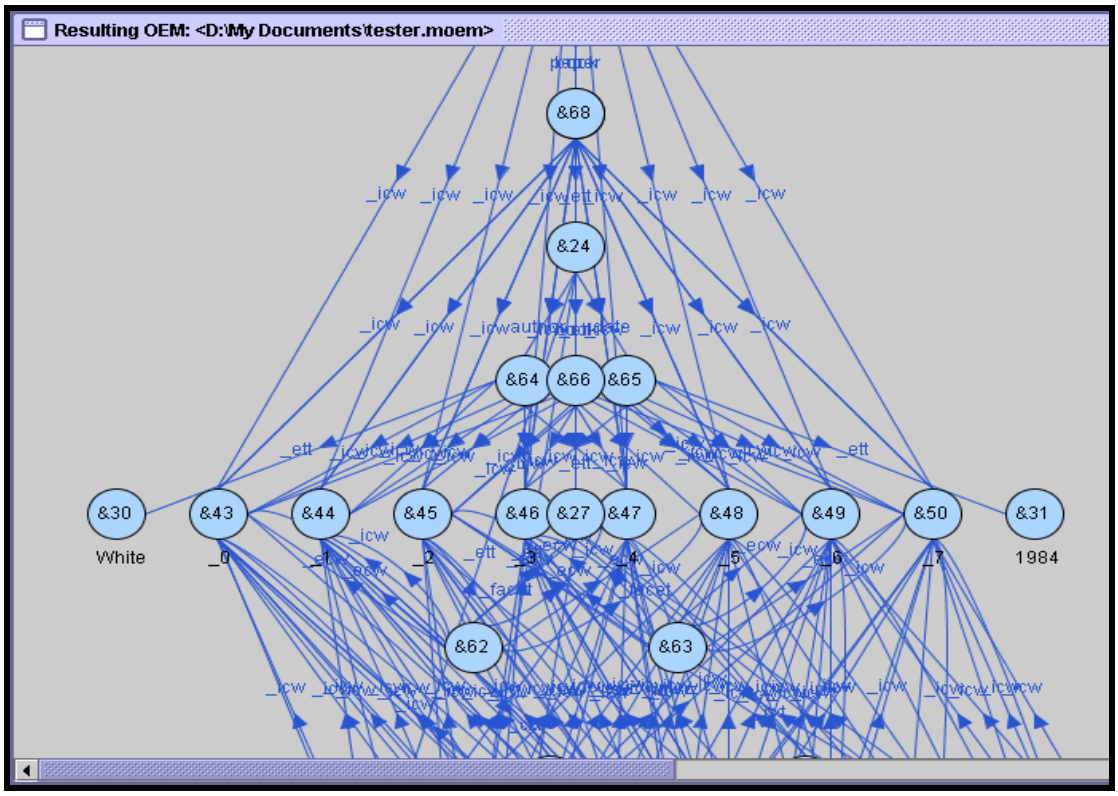
5.5.2 Μετατροπή γράφου με κύκλους σε OEM

Θεωρούμε τον εξής γράφο, που δεν βρίσκεται σε κανονική μορφή, προκειμένου να δοκιμάσουμε τους μετασχηματισμούς που ακολουθούν και σε μη κανονικοποιημένους γράφους:



Σχήμα 23: Κανονικοποιημένος γράφος με κύκλους

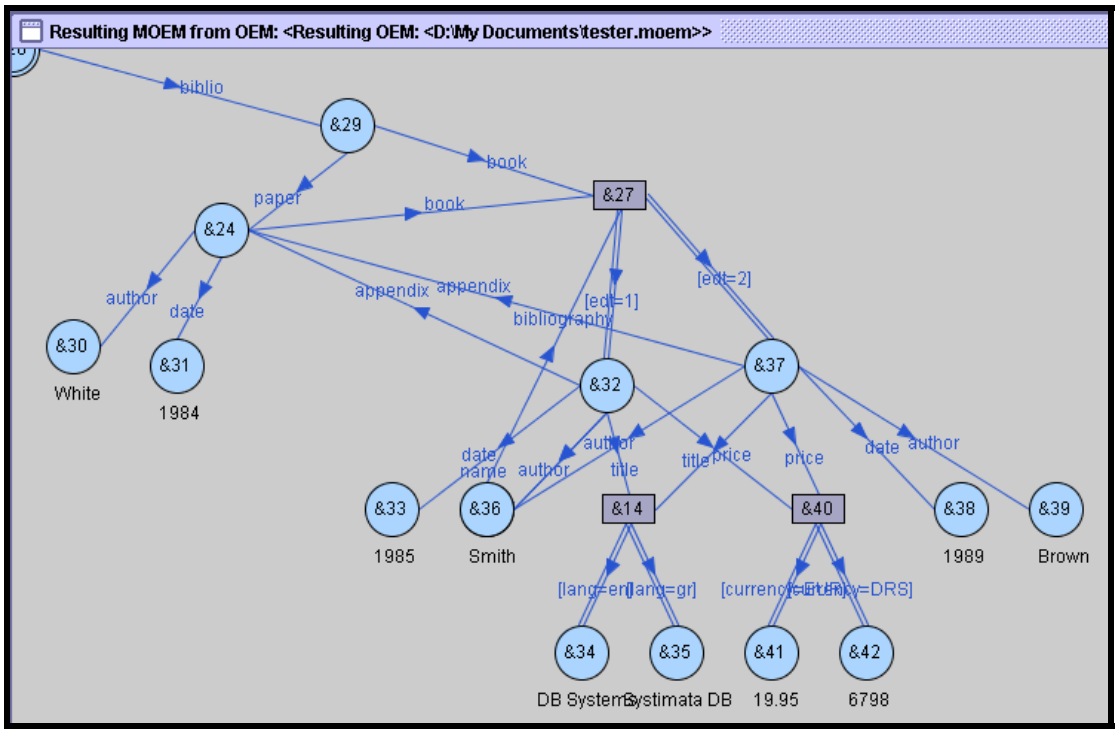
Με εφαρμογή του υλοποιημένου αλγορίθμου μετατροπής του στο μοντέλο OEM, παράγεται γράφος, τμήμα του οποίου ακολουθεί:



Σχήμα 24: Αντίστοιχος OEM γράφος με κύκλους

Λόγω του μεγέθους του παραπάνω γράφου, δεν είναι δυνατόν να παρουσιαστεί ολόκληρος. Προκειμένου να βγάλουμε συμπεράσματα για την ορθότητα της μετατροπής, δοκιμάσαμε τον αντίστροφο μετασχηματισμό, περιμένοντας να μας δώσει τον αρχικό γράφο.

Το αποτέλεσμα της αντίστροφης διαδικασίας είναι ο εξής γράφος:



Σχήμα 25: Αποτέλεσμα αντίστροφης μετατροπής από OEM σε MOEM

Συγκρίνοντας τον παραπάνω γράφο με τον αρχικό, είναι προφανές ότι οι μετατροπές έγιναν επιτυχημένα, με σωστό χειρισμό των κύκλων.

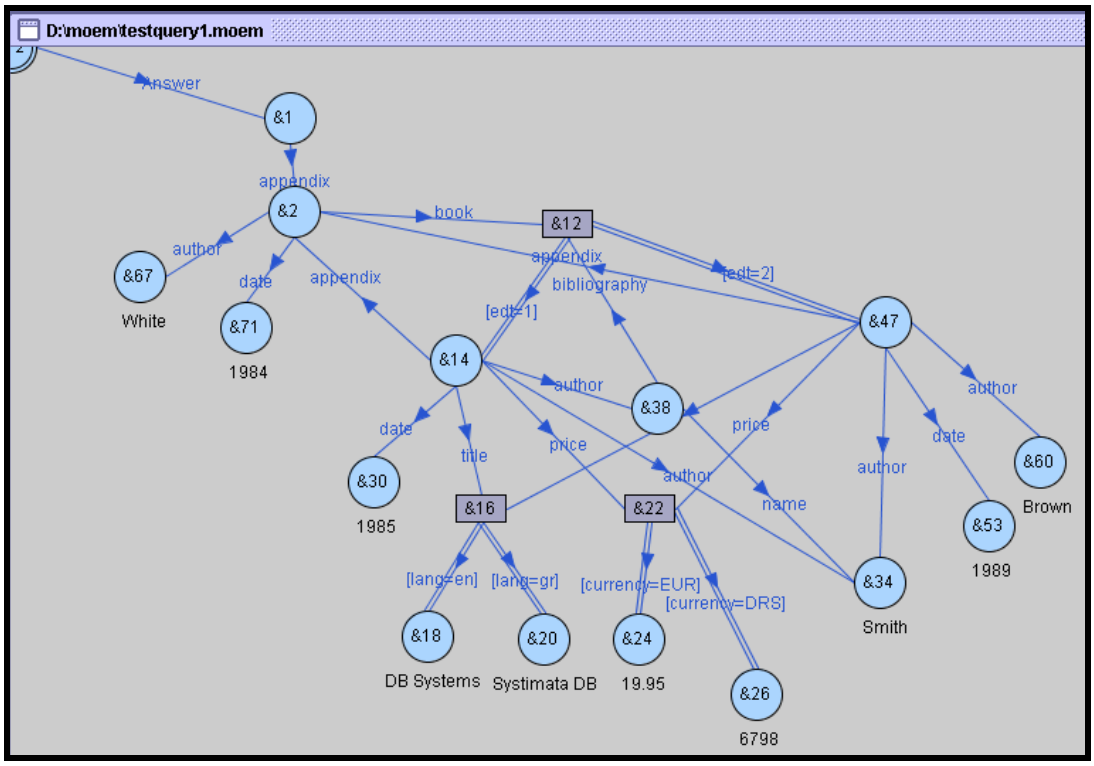
5.6 Έλεγχος συστήματος ερωτήσεων

Ολοκληρώνουμε τον έλεγχο καλής λειτουργίας της εφαρμογής υποβάλλοντας δύο ερωτήσεις στον γράφο tester.moem που χρησιμοποιήσαμε και προηγουμένως. Με τις ερωτήσεις αυτές ελέγχουμε τη σωστή διαχείριση των κύκλων όταν αυτοί εμφανίζονται στην απάντηση μιας MQL ερώτησης.

Η πρώτη ερώτηση που υποβάλαμε στον γράφο είναι η ακόλουθη:

```
select X
from  biblio.book.appendix X;
```

Το σύστημα παρουσίασε τον παρακάτω γράφο ως απάντηση στο προηγούμενο ερώτημα:



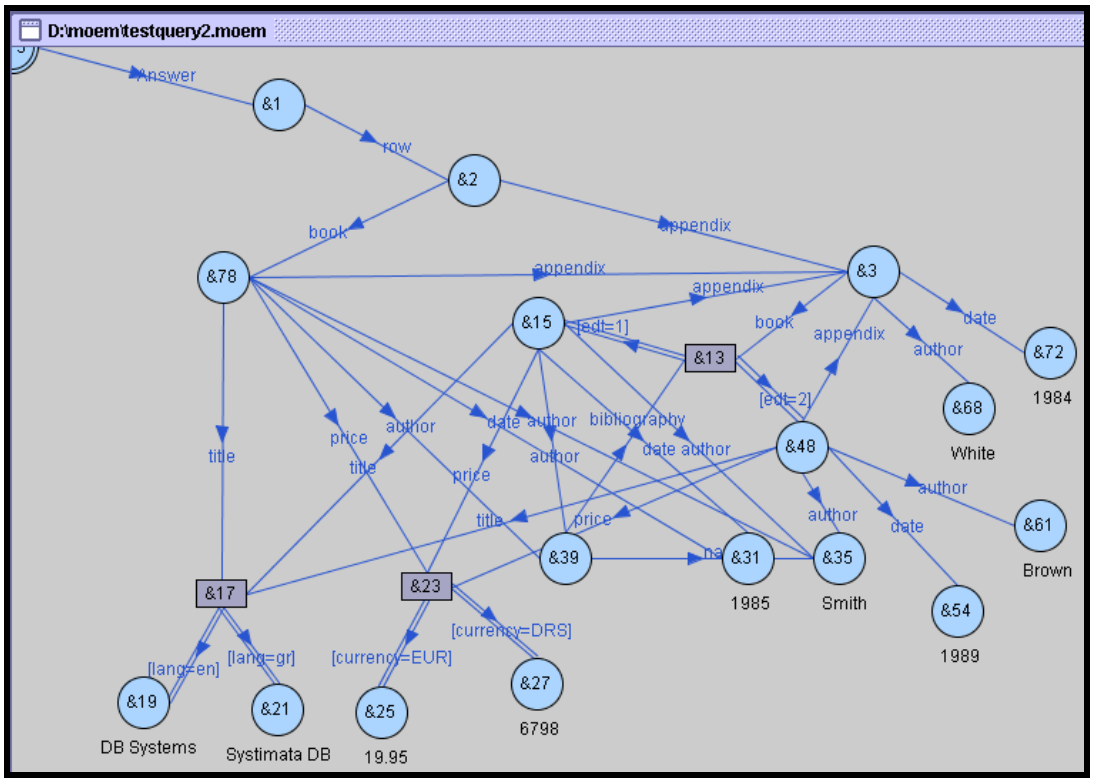
Σχήμα 26: Αποτέλεσμα MQL ερώτησης σε κυκλικό γράφο

Παρατηρούμε το σωστό χειρισμό των κύκλων, καθώς ακμές που ξεκινούν από απογόνους της απάντησης (κόμβος &2) και που καταλήγουν σε αυτήν σχεδιάστηκαν με το σωστό τρόπο.

Η επόμενη ερώτηση επιστρέφει ζεύγη αντικειμένων, μεταξύ των οποίων υπάρχει κάποια σχέση:

```
select X , Y
from  [edt=2]biblio.book.appendix X,
      X.[edt=1]book Y
```

Με την υποβολή της στο σύστημα, λάβαμε την εξής απάντηση:



Σχήμα 27: Αποτέλεσμα MQL ερώτησης δύο αντικειμένων

Το αποτέλεσμα, όπως περιμέναμε είναι λίγο πιο πολύπλοκο, πάντως φαίνεται καθαρά η σωστή παρουσίαση ακμών μεταξύ αντικειμένων όπου αποτιμούνται τα path που εμφανίζονται στην ερώτηση.

Σημειώνουμε ακόμα το γεγονός ότι οι κόμβοι &78 και &15, αν και παρουσιάζονται ως διαφορετικοί στην απάντηση, στην πραγματικότητα αναφέρονται και οι δύο στον κόμβο &32 του γράφου tester.moem. Ο συγκεκριμένος κόμβος, καθώς και οι ακμές που αναφέρονται σε αυτόν, εμφανίζεται στο γράφο της απάντησης δύο φορές, προκειμένου να είναι ευκρινές το γεγονός ότι αποτελεί αντικείμενο στο οποίο κατά την υποβολή της ερώτησης δεσμεύτηκε κάποια από τις μεταβλητές που συμμετέχουν στο select clause. Στην περίπτωση που εμφανιζόταν στην απάντηση μόνο ο κόμβος &15, δεν θα υπήρχε η ακμή “book” από τον κόμβο &2, με τη βοήθεια της οποίας δείχνουμε ότι ο εν λόγω κόμβος προήλθε σαν αποτέλεσμα της έκφρασης “Y” στο select clause της ερώτησης.

6

Επίλογος

6.1 Σύνοψη και συμπεράσματα

Με την παρούσα διπλωματική εργασία πετύχαμε την υλοποίηση ενός συστήματος επερωτήσεων που χρησιμοποιεί τη γλώσσα MQL, η οποία έχει σχεδιαστεί για πολυδιάστατα ημιδομημένα δεδομένα που παριστάνονται με τη βοήθεια ενός MOEM γράφου. Η συγκεκριμένη υλοποίηση παράγει OEM γράφους, που περιέχουν ισοδύναμη πληροφορία με τους αντίστοιχους MOEM, με τη βοήθεια ενός μοντέλου μετατροπής MOEM σε OEM που ορίστηκε στα πλαίσια της εργασίας. Με αυτόν τον τρόπο καθίσταται δυνατή η εκτέλεση ερωτήσεων σε γλώσσα MQL μέσω της χρησιμοποίησης του συστήματος διαχείρισης ημιδομημένων δεδομένων Lore, στα πλαίσια του οποίου υποστηρίζεται το μοντέλο OEM και η γλώσσα Lorel. Προκειμένου να επιτευχθεί αυτό, αναπτύχθηκε μία μέθοδος μετάφρασης ερωτήσεων από MQL σε Lorel.

Το συγκεκριμένο σύστημα επερωτήσεων ενσωματώθηκε στην ήδη υπάρχουσα εφαρμογή διαχείρισης MOEM γράφων MSSDesigner, η οποία έχει αναπτυχθεί στα πλαίσια του [ΔΖ01]. Για λόγους μεταφερσιμότητας το σύστημα υποβολής των ερωτήσεων και της επεξεργασίας των αντίστοιχων αποτελεσμάτων υλοποιήθηκε ως ένας δικτυακός διακομιστής που δέχεται από την κύρια εφαρμογή την Lorel ερώτηση καθώς και τον γράφο στον οποίο υποβάλλεται αυτή. Το αποτέλεσμα κάθε ερώτησης επιστρέφεται στον MSSDesigner προκειμένου να παρουσιαστεί στον χρήστη.

Η εργασία αυτή αποτελεί σημαντική πρακτική εφαρμογή μιας μελέτης που βρίσκεται ακόμα σε ερευνητικό στάδιο και συνεπώς τα συμπεράσματά της μπορούν να φανούν χρήσιμα κατά την αξιολόγηση του θεωρητικού υποβάθρου στο οποίο βασίζεται.

Η παρούσα υλοποίηση ενός βασικού υποσυνόλου του συντακτικού της MQL και η πρακτική εφαρμογή αυτής, καταδεικνύει την εκφραστική της πληρότητα καθώς και τη συμβατότητα της γλώσσας αυτής με τα θεωρητικά πρότυπα που έχουν οριστεί γενικά για τις γλώσσες επερωτήσεων σε ημιδομημένα δεδομένα.

Ένα πλήρες σύστημα διαχείρισης δεδομένων δεν νοείται χωρίς να παρέχει τη δυνατότητα υποβολής ερωτήσεων σε αυτά. Η εργασία αυτή μπορεί να αποτελέσει έναν οδηγό για την ανάπτυξη του υποσυστήματος που θα υλοποιήσει αυτή τη δυνατότητα στα πλαίσια ενός συστήματος που διαχειρίζεται πολυδιάστατα ημιδομημένα δεδομένα.

6.2 Μελλοντικές επεκτάσεις

Ένα βασικό τμήμα της MQL το οποίο έμεινε εκτός της υλοποίησης που αναπτύχθηκε, είναι οι δυνατότητες που παρέχει για την κατασκευή των αποτελεσμάτων μιας ερώτησης στα πλαίσια του select clause. Με τη βοήθεια κατάλληλων συντακτικών συμβάσεων, είναι δυνατόν να λάβουμε σαν αποτέλεσμα έναν γράφο που είναι περιορισμένος σε έναν μόνο κόσμο ή σε ένα σύνολο κόσμων. Προβλέπεται επίσης από την MQL η δυνατότητα επιστροφής όχι δεδομένων του γράφου, αλλά συνόλων κόσμων κάτω από τα οποία έχουν νόημα κάποια από αυτά. Τα παραπάνω περιγράφονται σε μια ερώτηση MQL με τη βοήθεια του context clause.

Για να υποστηριχθούν οι δυνατότητες αυτές, πέρα από την κατάλληλη επεξεργασία τέτοιου είδους ερωτήσεων από το υποσύστημα μετάφρασής τους, είναι αναγκαία και μία αναθεώρηση του μοντέλου OEM που χρησιμοποιούμε για την αναπαράσταση του MOEM γράφου στο Lore. Συγκεκριμένα, θα έπρεπε να εισάγουμε κάποιες επιπλέον συμβάσεις στο μοντέλο, με τις οποίες θα μπορούμε να περιγράφουμε και σύνολα κόσμων.

Με δεδομένο το αναθεωρημένο αυτό μοντέλο, μπορούμε να το επεκτείνουμε περαιτέρω συμπεριλαμβάνοντας και την αντιστοίχιση των κωδικών που περιγράφουν κόσμους με τους context specifier που ορίζουν αυτούς. Καταυτόν τον τρόπο επιτυγχάνεται η συγκέντρωση όλης της πληροφορίας του MOEM γράφου στο σύστημα του Lore.

Αξίζει να παρατηρήσουμε ότι χωρίς την τελευταία αυτή προσθήκη θα ήταν αδύνατο να υποστηριχτούν με βάση την παρούσα εφαρμογή ερωτήσεις ενημέρωσης του γράφου, μέσω εισαγωγών, διαγραφών και αλλαγών. Αυτό συμβαίνει διότι με κάθε τέτοια ερώτηση χρειάζεται να επαναϋπολογιστεί το inherited context ενός αρκετά μεγάλου αριθμού των ακμών του γράφου και συνεπώς είναι απαραίτητη η γνώση του universal context του γράφου, δηλαδή όλων των δυνατών διαστάσεων καθώς και των αντίστοιχων πεδίων τιμών τους. Με αυτόν τον τρόπο θα είναι επιπλέον δυνατή η επέκταση ή η συρρίκνωση του εν λόγω οικουμενικού συνόλου κόσμων.

7

Βιβλιογραφία

[SG01] Yannis Stavrakas, Manolis Gergatsoulis, *Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web,* Athens 2001

[AQM+96] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, Janet L. Wiener, *The Lorel Query Language for Semistructured Data,* Stanford University 1996

[GCC+] Roy Goldman, Sudarshan Chawathe, Arturo Crespo, Jason McHugh, *A Standard Textual Interchange Format for the Object Exchange Model (OEM),* Stanford University

[MAG+97] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, Jennifer Widom, *Lore: A Database Management System for Semistructured Data,* Stanford University 1997 (http://www-db.stanford.edu/lore)

[ΔΖ01] Δουλκερίδης Χρήστος, Ζαφείρης Βασίλειος, *Σχεδιασμός και υλοποίηση πλατφόρμας μοντελοποίησης πολυδιάστατων ημιδομημένων δεδομένων,* Διπλωματική Εργασία, Εργαστήριο Συστημάτων Βάσεων Γνώσεων & Δεδομένων, Ε.Μ.Π., Αθήνα 2001

[ABS00] Serge Abiteboul, Peter Buneman, Dan Suciu,
Data on the Web, From Relations to Semistructured Data and XML, Morgan Kaufmann Publishers San Francisco, California 2000

[Sta02] Yannis Stavrakas, *Multidimensional Query Language*, Technical Report, Knowledge and Database Systems Laboratory, National Technical University, Athens 2002