Αποθήκευση και διαχείριση XML με χρήση Σχεσιακών Συστημάτων Βάσεων Δεδομένων για εφαρμογές του Παγκόσμιου Ιστού

Φουστέρης Νικόλαος

Διδακτορική Διατριβή

Τμήματος Αρχειονομίας, Βιβλιοθηκονομίας και Μουσειολογίας
Ιόνιο Πανεπιστήμιο

Κέρκυρα, Νοέμβριος 2016

Περίληψη

Με βάση τα σημερινά δεδομένα, η γλώσσα XML (eXtended Markup Language) αποτελεί ένα από τα πιο ευρέως διαδεδομένα πρότυπα αναπαράστασης δεδομένων στον παγκόσμιο ιστό. Ένα μεγάλο ποσοστό εφαρμογών του διαδικτύου διαχειρίζεται πληροφορίες ημιδομημένης φύσεως, ενώ διάφορες παραλλαγές των πληροφοριών αυτών μπορούν να παρατηρηθούν, ανάλογα με τις επικρατούσες συνθήκες. Η *Πολυδιάστατη XML* (Multidimensional XML ή MXML) αποτελεί μία επέκταση της συμβατικής XML, η οποία είναι κατάλληλη για την αναπαράσταση, με έναν συμπαγή τρόπο, των διαφόρων εκφάνσεων των υπό επεξεργασία δεδομένων, σύμφωνα με τις διαφορετικές τιμές ή δομή που αποκτούν τα δεδομένα αυτά κάτω από διαφορετικά *ερμηνευτικά περιβάλλοντα* (contexts). Τα ερμηνευτικά αυτά περιβάλλοντα προσδιορίζονται αντιστοιχώντας τις κατάλληλες τιμές σε έναν ορισμένο αριθμό διαστάσεων. Η παρούσα διδακτορική διατριβή ασχολείται με τη μελέτη του προβλήματος της αποθήκευσης, της ενημέρωσης και την υποβολής ερωτημάτων προς MXML δεδομένα, με τη χρήση σχεσιακών βάσεων δεδομένων.

Εξαιτίας της ήδη υπάρχουσας εμπειρίας στον χώρο των *Συστημάτων Διαχείρισης Σχεσιακών Βάσεων Δεδομένων* (RDBMS), το πρόβλημα της αποθήκευσης και εκτέλεσης ερωτημάτων για XML δεδομένα που είναι αποθηκευμένα σε σχεσιακές βάσεις δεδομένων αποτελεί ένα θέμα που έχει εκτενώς ερευνηθεί, ενώ πολλές σχετικές τεχνικές έχουν αναπτυχθεί. Με βάση παλιότερες ερευνητικές εργασίες που σχετίζονται με την αποθήκευση και υποβολή ερωτημάτων σε XML δεδομένα που είναι αποθηκευμένα σε σχεσιακά σχήματα, παρουσιάζουμε τα ερευνητικά μας αποτελέσματα σε ότι αφορά την αποθήκευση, την πλοήγηση, την υποβολή ερωτημάτων και την ενημέρωση MXML κειμένων με τη χρήση σχεσιακών βάσεων δεδομένων, λαμβάνοντας υπόψη τα επιπρόσθετα χαρακτηριστικά της MXML σε σχέση με την απλή XML. Τα επιπρόσθετα αυτά χαρακτηριστικά προκύπτουν από την ενσωμάτωση του ερμηνευτικού περιβάλλοντος στα MXML κείμενα.

Μελετώντας το πρόβλημα της *αποθήκευσης MXML δεδομένων* (MXML Storage), παρουσιάζουμε τεχνικές και εναλλακτικά σχεσιακά σχήματα για την αποθήκευση MXML κειμένων σε σχεσιακές βάσεις δεδομένων. Βασικά χαρακτηριστικά των προτεινόμενων τεχνικών είναι η δυνατότητα α) της αναδόμησης του αρχικού MXML κειμένου από την σχεσιακή του αναπαράσταση και β) της μετατροπής MXML ερωτημάτων που περιλαμβάνουν πληροφορίες σχετικές με ερμηνευτικά περιβάλλοντα σε SQL ερωτήματα. Επιπλέον, για όλες τις προτεινόμενες τεχνι-

κές αποθήκευσης, ο τρόπος αναπαράστασης του ερμηνευτικού περιβάλλοντος στο σχεσιακό σχήμα αποτελεί ένα πολύ σημαντικό θέμα, καθώς η αναπαράσταση αυτή μπορεί να διευκολύνει τη διαχείριση και την πλοήγηση εντός των MXML δεδομένων.

Σε ότι αφορά την υποβολή MXML ερωτημάτων, παρουσιάζουμε μία επέκταση της XPath, η οποία ονομάζεται *πολυδιάστατη XPath* (Multidimensional XPath ή MXPath). Η επέκταση αυτή είναι κατάλληλη για την διάσχιση MXML δεδομένων, αλλά και την κατασκευή ερωτημάτων συμπεριλαμβανομένου των πληροφοριών που αφορούν ερμηνευτικά περιβάλλοντα. Επιπλέον, ορίζουμε τη σύνταξη της MXPath, παραθέτουμε σχετικά παραδείγματα επιδεικνύοντας τη λειτουργία της και δίνουμε τη σημασιολογική της ερμηνεία.

Επιπρόσθετα, αναπτύσσουμε τεχνικές που αφορούν την ενημέρωση των MXML κειμένων και μελετάμε το θέμα αυτό σε δύο επίπεδα: α) σε γραφικό επίπεδο, δηλαδή ανεξαρτήτως υλοποίησης και β) σε επίπεδο σχεσιακής βάσης δεδομένων. Πιο συγκεκριμένα ορίζουμε έξι βασικές πράξεις ενημερώσεως, οι οποίες μπορούν να επιτύχουν οποιαδήποτε πιθανή μεταβολή. Ο ορισμός των πράξεων αυτών γίνεται αρχικά σε επίπεδο ανεξάρτητο υλοποίησης, ενώ εξηγούνται οι επιπτώσεις της εφαρμογής τους στα MXML κείμενα μέσω σχετικών παραδειγμάτων. Ακόμα, παραθέτουμε αλγόριθμους που υλοποιούν τις παραπάνω πράξεις με τη χρήση της SQL, βάσει συγκεκριμένου σχεσιακού σχήματος αποθήκευσης των MXML δεδομένων.

Τέλος, παρουσιάζουμε έναν αλγόριθμο που μετατρέπει MXML ερωτήματα σε SQL ερωτήματα, τα οποία εκτελούνται πάνω στα MXML δεδομένα που είναι αποθηκευμένα σε σχεσιακό σχήμα βασισμένο στα μονοπάτια του *MXML δένδρου* (MXML tree). Επιπλέον, περιγράφουμε το σύστημα που αναπτύχθηκε (MXML Tools) προκειμένου να υλοποιηθεί ο σχετικός αλγόριθμος μετατροπής, παρέχοντας τις απαραίτητες τεχνικές λεπτομέρειες που αφορούν την υποδομή του συγκεκριμένου συστήματος.

Abstract

In nowadays, XML (eXtended Markup Language) is one of the most popular standard for representing data on the WEB. Although, in a wide spectrum of internet applications it is often required to manipulate information of semistructured nature, which may present variations according to different circumstances. Multidimensional XML (MXML) is an extension of XML suitable for representing, in a compact way, data that assume different facets, having different value and structure under different contexts, which are determined by assigning values to a number of dimensions. This doctoral thesis studies the problem of storing, updateing and querying MXML data using relational databases.

Because of the great experience that already exists on the field of Relational Database Managment Systems (RDBMS), the problem of storing (XML Storage) and quering XML data using relational databases has been considered a lot and many related techniques have been developed. Following previous work on storing and quering XML in relational databases, we present the results of our research on storing (MXML Storage), navigating, querying and updating MXML documents using relational databases, taking into account the additional features of MXML compared to conventional XML. Those features stem from the incorporation of context into MXML.

Investigating the problem of MXML storage, we present techniques and alternative relational schemas for storing MXML documents in relational databases, based on storage techniques previously proposed for conventional XML documents. Essential characteristics of the proposed techniques are the capabilities a) to reconstruct the original MXML document from its relational representation and b) to express MXML context-aware queries in SQL. Also, for all the proposed storage techniques, the way that the context is represented in the relational schema is a very important issue, as such a representation may facilitate the management and the navigation through the stored MXML data.

As regards MXML quering, we introduce an extension of XPath called Multidimensional XPath (MXPath), which is suitable for navigating in MXML documents, and allows for context-aware querying. We present the syntax of MXPath, we provide examples demonstrating its use and investigate its semantics.

Additionally, we develop techniques for updating MXML documents and we investigate this problem in two levels: a) at the graph level, i.e. in an implementa-

tion independent way, and b) at the relational storage level. More specificaly, we introduce six basic update operations, which are capable of any possible change. We specify those operations in an implementation independent way, and explain their effect in the document through examples. Also, we give algorithms that implement those operations using SQL on a specific storage method that employs relational tables for keeping MXML.

Finally, we present an algorithm for converting context-aware MXML queries to SQL queries for execution over the MXML data, which are stored in a path-based relational database. Also, we describe the system we developed (MXML Tools) in order to implement the above conversion algorithm and we provide all the necessary technical information concerning the infrastructure of the system.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Μανόλη Γεργατσούλη, του οποίου η καθοδήγηση και οι συμβουλές με βοήθησαν σημαντικά σε όλη τη διάρκεια της ερευνητικής μου εργασίας. Η γνώση, η εμπειρία και η κριτική του ικανότητα αποτέλεσε πολλές φορές για εμένα έναν πολύτιμο σύμμαχο στην όλη προσπάθειά μου.

Επιπλέον, θα ήθελα να ευχαριστήσω τους καθηγητές που συμμετείχαν στην τριμελή επιτροπή Χρήστο Παπαθεοδώρου και Μηχαήλ Βαζιργιάννη, οι οποίοι με τις παρατηρήσεις και τις συμβουλές τους με βοήθησαν να προχωρήσω και να ολοκληρώσω την ερευνητική μου εργασία.

Ακόμα, θα ήθελα να αναφερθώ στους συναδέλφους μου Λίνα Μπουντούρη, Ειρήνη Λούρδη και Παντελή Λιλή, οι οποίοι υπήρξαν σημαντικοί συνεργάτες στο ξεκίνημα της διατριβής μου, στα πλαίσια της ομάδας συνεργασίας που είχε τότε συσταθεί, υπό τον συντονισμό των κυρίων Γεργατσούλη και Παπαθεοδώρου, καθώς και στην εξαιρετική συνεργασία που είχα κατά καιρούς με τον ερευνητή του Ινστιτούτου Πληροφοριακών Συστημάτων Γιάννη Σταύρακα.

Επιπρόσθετα, αξίζει να σημειωθεί, ότι καθ'όλη τη διάρκεια των μεταπτυχιακών μου σπουδών, πολύτιμη υπήρξε η οικονομική αρωγή του Ιδρύματος Κρατικών Υποτροφιών (Ι.Κ.Υ.).

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, η οποία όλα αυτά τα χρόνια με στήριζε και με βοηθούσε, δίνοντας μου το κουράγιο και την δύναμη να συνεχίζω και να προσπερνώ κάθε εμπόδιο και δυσκολία.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Στο πρώτο κεφάλαιο της διδακτορικής διατριβής παρουσιάζεται το πρόβλημα της αποθήκευσης XML δεδομένων σε σχεσιακές βάσεις δεδομένων, καθώς και η τρέχουσα πραγματικότητα γύρω από το ζήτημα αυτό. Το πρόβλημα αυτό έχει απασχολήσει για αρκετά χρόνια την επιστημονική κοινότητα, αφού αφορά την αξιοποίηση της ήδη υπάρχουσας εμπειρίας γύρω από τα *σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων* (RDBMS) για την διαχείριση XML δεδομένων, τα οποία χρησιμοποιούν οι σύγχρονες εφαρμογές του *διαδικτύου* (internet).

Δεδομένης της υπάρχουσας ερευνητικής δουλειάς πάνω στο πρόβλημα της αποθήκευσης XML δεδομένων, η παρούσα διατριβή διευρύνει την εν λόγο έρευνα και εισάγει ως νέο αντικείμενο αυτής μία επέκταση της XML, την *Πολυδιάστατη XML* (Multidimensional XML ή MXML). Έτσι, ακολουθεί ο ορισμός ενός νέου προβλήματος, το οποίο αν και περιέχει τα ίδια ερωτήματα που ανακύπτουν κατά την αποθήκευση της απλής XML σε ένα σχεσιακό σχήμα, περιλαμβάνει επιπλέον όλα τα επιπρόσθετα δεδομένα τα οποία χαρακτηρίζουν την MXML. Η αντιμετώπιση αυτού του νέου προβλήματος αποτελεί και στόχο αυτής της διδακτορικής διατριβής.

1.1 Τρέχουσα Πραγματικότητα

Η γλώσσα XML (Extensible Markup Language) [GM03, CON07a] αποτελεί ένα πρότυπο αναπαράστασης δεδομένων, το οποίο δεσπόζει στον *παγκόσμιο ιστό* (WEB) και χρησιμοποιείται ευρύτατα από εφαρμογές του διαδικτύου για την πε-

ριγραφή δεδομένων με έναν ιεραρχικά δομημένο τρόπο. Αρκετά εμπορικά προϊόντα υποστηρίζουν εγγενώς το πρότυπο της γλώσσας XML (native XML Database Servers) [GMW99, NDM+01] ενώ και κάποια άλλα παραδοσιακά συστήματα βάσεων δεδομένων περιλαμβάνουν επιπρόσθετες συμβατές με την XML λειτουργίες (Oracle, DB2, SQL Server) [Ben03]. Από την άλλη πλευρά, υπάρχει μία μεγάλη ποσότητα δεδομένων και πληροφοριών, οι οποίες είναι αποθηκευμένες σε κοινά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων, τα οποία δεν είναι συμβατά με το πρότυπο της XML. Ανακύπτει λοιπόν το ερώτημα σχετικά με το πως θα ήταν δυνατή η *ολοκλήρωση* (intergration) των δύο αυτών κόσμων (XML και RDBMS) και πως οι XML εφαρμογές θα μπορούσαν να χρησιμοποιήσουν τα δεδομένα, που είναι αποθηκευμένα σε προϋπάρχουσες σχεσιακές βάσεις δεδομένων, αξιοποιώντας τη γνώση και την εμπειρία πάνω σε τέτοιου είδους πλατφόρμες.

Γενικά, το πρόβλημα της ολοκλήρωσης μεταξύ XML δεδομένων και των δεδομένων σε σχεσιακές βάσεις δεδομένων, σύμφωνα με την υπάρχουσα βιβλιογραφία, εκφράζεται με δύο διαφορετικές προσεγγίσεις σε ότι αφορά τις XML εφαρμογές:

1. *Αποθήκευση XML δεδομένων*: Η προσέγγιση αυτή περιλαμβάνει τεχνικές όπου τα XML δεδομένα αποθηκεύονται σε σχεσιακά σχήματα. Έτσι, οι XML εφαρμογές μπορούν να εκμεταλλευτούν τα πλεονεκτήματα της RDBMS τεχνολογίας.

2. *Έκδοση XML δεδομένων*: Σύμφωνα με την προσέγγιση αυτή αναζητούνται τρόποι ώστε οι XML εφαρμογές να θέτουν ερωτήματα προς XML δεδομένα, τα οποία προκύπτουν από δεδομένα που είναι ήδη αποθηκευμένα σε σχεσιακά σχήματα.

1.1.1 Αποθήκευση XML δεδομένων σε σχεσιακές βάσεις δεδομένων

Εισαγωγή

Το πρόβλημα της *αποθήκευσης XML δεδομένων* (XML Storage) σε σχεσιακές βάσεις δεδομένων έχει αποτελέσει τα τελευταία χρόνια αντικείμενο μελέτης πολλών ερευνητών. Ο αντικειμενικός σκοπός αυτών των ερευνών είναι η ανακάλυψη

τεχνικών, οι οποίες επιτρέπουν την αποθήκευση XML δεδομένων σε σχεσιακές βάσεις δεδομένων, ούτως ώστε να είναι δυνατή στη συνέχεια η υποβολή σχετικών *ερωτημάτων* (queries) προς τα αποθηκευμένα δεδομένα. Σύμφωνα με αυτή την προσέγγιση, ένα σχεσιακό σχήμα επιλέγεται για την αποθήκευση των XML δεδομένων ενώ ακολουθεί η αποθήκευση των δεδομένων αυτών σε ένα σύστημα διαχείρισης σχεσιακής βάσης δεδομένων. Τα διάφορα ερωτήματα προς τα αποθηκευμένα δεδομένα, υποβάλλονται αρχικά ως XML ερωτήματα από τις εφαρμογές των χρηστών και στη συνέχεια μετασχηματίζονται σε αντίστοιχα SQL ερωτήματα. Τα αποτελέσματα που προκύπτουν από την εκτέλεση των SQL ερωτημάτων μετασχηματίζονται ξανά σε XML δεδομένα και επιστρέφονται πίσω στην εφαρμογή του χρήστη.

Οι διάφορες τεχνικές που χρησιμοποιούνται για την επίτευξη της αποθήκευσης XML δεδομένων διαχωρίζονται σε δύο βασικές κατηγορίες, ανάλογα με την ύπαρξη ή μη ύπαρξη *XML σχήματος* (XML schema) για τα XML δεδομένα:

1. *Τεχνικές αποθήκευσης XML δεδομένων βάσει XML σχήματος*: Είναι τεχνικές σύμφωνα με τις οποίες ο προσδιορισμός του σχεσιακού σχήματος για την αποθήκευση των XML δεδομένων γίνεται χρησιμοποιώντας κάποιο XML σχήμα

2. *Τεχνικές αποθήκευσης XML δεδομένων ανεξαρτήτως XML σχήματος*: Είναι τεχνικές σύμφωνα με τις οποίες δεν χρειάζεται κάποιο XML σχήμα για τον προσδιορισμό του σχεσιακού σχήματος στο οποίο θα αποθηκευθούν τα XML δεδομένα.

Οι δύο αυτές προσεγγίσεις, περιγράφονται στα παρακάτω υποκεφάλαια.

Αποθήκευση XML δεδομένων βάσει XML σχήματος

Στις τεχνικές *αποθήκευση XML δεδομένων βάσει κάποιου XML σχήματος* (Schema Based XML Storage) [RFHR03, TVB+02, DAYF04, TDCZ02, KP02, SSK+01, BFRS02, AYS02, STZ+99, LC00, ML02], αυτό που προσπαθούμε να επιτύχουμε είναι η εύρεση ενός σχεσιακού σχήματος για την αποθήκευση XML κειμένων, λαμβάνοντας υπόψη το XML σχήμα στο οποίο τα κείμενα αυτά υπακούουν. Για το λόγο αυτό και δεδομένου ενός XML σχήματος (ή DTD), καλούμαστε να επιλέξουμε το κατάλληλο σχεσιακό σχήμα μαζί με τη σχετική *απεικόνιση* (mapping)

της XML προς το σχεσιακό σχήμα. Επιπλέον, χρειαζόμαστε να προσδιορίσουμε τρόπους μετατροπής των XML ερωτημάτων σε SQL ερωτήματα, με βάση την προαναφερθείσα απεικόνιση.

Στο [TDCZ02] παρουσιάζεται μία προσέγγιση αποθήκευσης XML δεδομένων βάσει XML σχήματος, η οποία ονομάζεται "Relational DTD approach". Σύμφωνα με την προσέγγιση αυτή, το DTD που αντιστοιχεί σε ένα XML έγγραφο μεταφράζεται σε γράφο, ο οποίος απεικονίζει το γεγονός ότι κάποιο *στοιχείο* (element) του XML δένδρου μπορεί να εμφανιστεί μία ή περισσότερες φορές κάτω από τον αντίστοιχο κόμβο γονέα. Στη συνέχεια, έναν ξεχωριστός πίνακας στο σχεσιακό σχήμα χρησιμοποιείται για να αποθηκεύσει την *περιορισμένη βάσει συνόλου σχέση* (set-containment relationship) μεταξύ ενός στοιχείου (κόμβου γονέα) του XML δένδρου και του συνόλου των στοιχείων παιδιών του που έχουν την ίδια *ετικέτα* (tag). Σε κάθε *έγγραφη* (tuple) ενός σχεσιακού πίνακα, η οποία αποθηκεύει ένα XML στοιχείο, αντιστοιχίζεται ένας *μοναδικός κωδικός αριθμός* (ID), ενώ περιλαμβάνεται ένα πεδίο *μοναδικού κωδικού αριθμού γονέα* (parentlD) προκειμένου να προσδιοριστεί ο γονέας του στοιχείου αυτού. Κάθε στοιχείο που μπορεί να εμφανίζεται μόνο μία φορά κάτω από τον γονέα του, *ενσωματώνεται* (inlined) σαν μία στήλη του πίνακα ο οποίος αναπαριστά τον γονέα του στοιχείου αυτού. Εάν ένας DTD γράφος περιέχει έναν κύκλο, τότε για τη διάσπαση του κύκλου αυτού χρησιμοποιείται ένας ξεχωριστός πίνακας.

Στο [LC00], παρουσιάζεται ένας *αλγόριθμος ενσωμάτωσης σημασιολογικών περιορισμών* (constraint preserving inlining algorithm ή CPI), ο οποίος κατά την μετατροπή ενός DTD σε σχεσιακό σχήμα, μπορεί να καταγράφει τους *σημασιολογικούς περιορισμούς* (semantic constrains) που προκύπτουν από το DTD αυτό. Στη συνέχεια οι περιορισμοί που έχουν καταγραφεί αποδίδονται στο προκύπτον σχεσιακό σχήμα. Σύμφωνα με την σχετική έρευνα, για την μετατροπή ενός DTD σε σχεσιακό σχήμα επιλέχθηκε ένας *υβριδικός αλγόριθμος ενσωμάτωσης* (hybrid inlining algorithm), ο οποίος περιγράφεται παρακάτω στο [STZ+99], ώστε να γίνει αντιληπτό το πως οι σημασιολογικοί περιορισμοί μπορούν να παραχθούν.

Το σύστημα X2Rmap [KP02] παρέχει μία *ενδιάμεση δομή απεικόνισης* (mapping structure middleware) μεταξύ της XML και των σχεσιακών δεδομένων, μέσω της οποίας επιτυγχάνεται καλύτερη απόδοση κατά την εκτέλεση των XML ερωτημάτων, αναλύοντας τα στατιστικά στοιχεία που προκύπτουν μέσω της εκτέλεσης των ερωτημάτων αυτών και μεταβάλλοντας κατάλληλα το σχεσιακό σχήμα,

ακόμα και στην περίπτωση που αυτό έχει ήδη καθοριστεί. Σύμφωνα με την προσέγγιση αυτή τα XML δεδομένα τμηματοποιούνται και αποθηκεύονται σε ένα σχεσιακό σχήμα, ωστόσο δεδομένου των πληροφοριών απεικόνισης που έχουν αποθηκευθεί, το XML σχήμα και το σχεσιακό σχήμα ανεξαρτητοποιούνται. Με αυτόν τον τρόπο, το σχεσιακό σχήμα μπορεί ελεύθερα να τροποποιηθεί, αφού οι πληροφορίες απεικόνισης διατηρούνται (σε μορφή X2RMap δενδροειδούς πληροφορίας σε XML αρχείο) και κάθε XML ερώτημα μπορεί να *επαναδιατυπωθεί* (rewritten) ως αντίστοιχη SQL έκφραση, χρησιμοποιώντας την απεικόνιση αυτή (το σύστημα αποθηκεύει τις πληροφορίες του τροποποιημένου σχήματος στη δομή απεικόνισης). Η δομή απεικόνισης του X2RMap εκφράζεται με έναν εκτεταμένο DTD γράφο, ο οποίος διαχωρίζεται από τον DTD γράφο του XML κειμένου, ενώ υλοποιείται με μία δενδροειδή δομή δεδομένων. Αυτή η δομή δεδομένων περιλαμβάνει τις βασικές πληροφορίες απεικόνισης μεταξύ των πινάκων (στηλών πίνακα) και κάθε κόμβου του XML δένδρου.

Σύμφωνα με το [STZ+99], το DTD ενός XML κειμένου επεξεργάζεται κατά τέτοιο τρόπο, ώστε να προκύψει το επιθυμητό σχεσιακό σχήμα. Τότε, τα XML κείμενα μεταφέρονται σε σχεσιακούς πίνακες κάποιου εμπορικού συστήματος διαχείρισης βάσεων δεδομένων (στην προκειμένη περίπτωση, του IBM DB2). Τελικά, XML ερωτήματα (ορισμένα σε μία γλώσσα παρόμοια με την XML-QL [DFF+98] ή τη Lorel [AQM+97]) προς τα XML δεδομένα μετατρέπονται σε SQL ερωτήματα που απευθύνονται στα σχεσιακά δεδομένα, ενώ τα αποτελέσματα μετατρέπονται ξανά σε XML. Σε ότι αφορά το ζήτημα της αποθήκευσης XML δεδομένων, τα κύρια θέματα που αντιμετωπίζει η προσέγγιση αυτή σχετίζονται (α) με την πολυπλοκότητα του ορισμού των DTD στοιχείων, (β) με τα προβλήματα που δημιουργούνται μεταξύ της φύσεως των σχεσιακών σχημάτων που αποτελείται από δύο επίπεδα (πίνακες και γνωρίσματα πινάκων) και της μη σαφώς καθορισμένης *ενθυλάκωσης* (nesting) των XML DTD σχημάτων και (γ) την διαχείριση των *γνωρισμάτων συνόλου τιμών* (set-valued attributes) και της *αναδρομής* (recursion) που μπορεί να παρουσιαστεί σε ένα DTD. Θεωρείται λοιπόν ότι οι λεπτομέρειες ενός DTD μπορούν να απλοποιηθούν, διατηρώντας τη δυνατότητα παραγωγής ενός σχεσιακού σχήματος, το οποίο να μπορεί να δέχεται ερωτήματα και να αποθηκεύει XML δεδομένα σύμφωνα με το συγκεκριμένο DTD. Για αυτή την απλοποίηση, χρησιμοποιείται ένα σύνολο από μετατροπές, οι οποίες δεν υπονομεύουν την αποδοτικότητα των ερωτημάτων που απευθύνονται σε κείμενα τα οποία αντιστοι-

χούν στο αρχικό DTD. Για την μετατροπή ενός XML DTD σε σχεσιακό σχήμα, προτείνονται διάφορες τεχνικές:

1. *Η Βασική τεχνική ενσωμάτωσης (basic inlining)*: Η τεχνική αυτή υλοποιείται με την ενσωμάτωση όσο το δυνατόν περισσοτέρων απογόνων ενός XML στοιχείου σε έναν σχεσιακό πίνακα. Ωστόσο, η βασική προσέγγιση δημιουργεί σχεσιακούς πίνακες για κάθε στοιχείο, αφού ένα XML κείμενο μπορεί να έχει ως ρίζα κάθε στοιχείο σε ένα DTD. Σε αυτή την τεχνική, η δομή του DTD εκφράζεται σαν γράφος (DTD Graph), όπου το φαινόμενο της αναδρομής απεικονίζεται σαν κύκλος. Από τον γράφο αυτόν, άλλοι γράφοι μπορούν να κατασκευαστούν (γράφοι στοιχείων ή Element Graphs) με βάση συγκεκριμένα στοιχεία του DTD γράφου. Δεδομένου ενός γράφου στοιχείου, μπορούν να δημιουργηθούν οι αντίστοιχοι σχεσιακοί πίνακες, μέσω κατάλληλου αλγορίθμου.

2. *Η Διαμοιραζόμενη τεχνική ενσωμάτωσης (Shared inlining)*: Σύμφωνα με την τεχνική αυτή, εξασφαλίζεται ότι ένας κόμβος στοιχείου (element node) ενός DTD απεικονίζεται σε έναν ακριβώς σχεσιακό πίνακα. Η βασική ιδέα που ακολουθεί η τεχνική αυτή είναι ο εντοπισμός των κόμβων στοιχείων που αναπαριστώνται σε περισσότερους από έναν πίνακες κατά την Βασική τεχνική, προκειμένου οι κόμβοι αυτοί να απεικονιστούν σε έναν ξεχωριστό πίνακα του σχεσιακού σχήματος.

3. *Η Υβριδική τεχνική ενσωμάτωσης (Hybrid inlining)*: Αποτελεί έναν συνδυασμό της Βασικής και της Διαμοιραζόμενης τεχνικής ενσωμάτωσης.

Το LegoDB [BFRS02] είναι ένας μηχανισμός απεικόνισης της XML προς τα σχεσιακά δεδομένα, ο οποίος βασίζεται τόσο στο XML σχήμα όσο και σε διάφορα κόστη ή βάρη (cost based) τα οποία χρησιμοποιούνται σε στατιστικές μετρήσεις. Ο παραπάνω μηχανισμός εξερευνά μία περιοχή από πιθανές απεικονίσεις και επιλέγει την πιο αποδοτική από αυτές για μία συγκεκριμένη εφαρμογή. Τα δεδομένα εισόδου του μηχανισμού αποτελούνται από το XML σχήμα των XML δεδομένων, κάποια XML στατιστικά δεδομένα και έναν XML *φόρτο ερωτημάτων* (query workload). Για την λειτουργία του, το LegoDB χρησιμοποιεί τα ονομαζόμενα *φυσικά XML σχήματα* (physical XML schemas ή p-schemas), τα οποία είναι XML σχήματα που έχουν επεκταθεί με την προσθήκη στατιστικών στοιχείων σχετικά

με τα υφιστάμενα XML δεδομένα. Μετά την κατασκευή κάποιας απεικόνισης για ένα p-schema, ένας φόρτος ερωτημάτων (σύνολο από ερωτήματα σε κάθε ένα από τα οποία να αντιστοιχεί ένας δείκτης βάρους, ο οποίος αντιπροσωπεύει, σε σχέση με τα υπόλοιπα ερωτήματα, τη σημαντικότητα του κάθε ερωτήματος για κάποια εφαρμογή) εκτελείται. Η ίδια διαδικασία επαναλαμβάνεται, μετά από κάθε μετατροπή του p-schema, για αρκετές φορές, μέχρι η πλέον αποδοτική απεικόνιση να επιλεγεί από το σύστημα.

Στο [RFHR03], παρουσιάζεται στο σύστημα FlexMap. Το FlexMap αποτελεί έναν μηχανισμό παραγωγής απεικονίσεων της XML προς τα σχεσιακά δεδομένα, με τη χρήση ενός συνόλου από *μετασχηματισμούς σχήματος* (schema transformations). Ορίζοντας ένα σύνολο από μετασχηματισμούς σχήματος, οδηγούμαστε σε διαφορετικές διευθετήσεις του σχεσιακού σχήματος. Επιπλέον, με βάση κάποιον XML φόρτο ερωτημάτων, εκτιμώνται τα ποιοτικά χαρακτηριστικά του σχεσιακού αυτού σχήματος. Αρχικά, χρησιμοποιείται ένα *δένδρο σχήματος* (schema tree) για την αναπαράσταση του XML σχήματος, ενώ με βάση το σχήμα αυτό εξάγεται το σχεσιακό σχήμα (mapping procedure). Στη συνέχεια, μετασχηματισμοί μπορούν να εφαρμοστούν στο δένδρο σχήματος ώστε να προκύψουν νέα σχεσιακά σχήματα. Για κάθε σχεσιακό σχήμα, ένας αλγόριθμος αναζήτησης παίρνει σαν δεδομένα εισόδου έναν φόρτο ερωτημάτων και το αρχικό σχήμα, ενώ μετά από επεξεργασία επιλέγεται το καταλληλότερο και χαμηλότερου κόστους σχήμα. Ένας φόρτος ερωτημάτων αποτελείται από ένα σύνολο ερωτημάτων με συγκεκριμένο βάρος (σε μία κλίμακα από το 0 έως το 1) για το κάθε ένα. Αυτά τα βάρη, αντικατοπτρίζουν τη σημασία του κάθε ερωτήματος για μία συγκεκριμένη εφαρμογή.

Σύμφωνα τέλος με τις αναφορές [BFRS02, STZ+99], οι πληροφορίες σχετικά με το σχήμα ενός κειμένου μπορούν να χρησιμοποιηθούν για τον σχεδιασμό του καλύτερου δυνατού σχήματος απεικόνισης, με βάση και τη απόδοση κατά την εκτέλεση των XML ερωτημάτων. Από την άλλη πλευρά, η σωστή μετατροπή των XML ερωτημάτων και ενημερώσεων σε SQL εντολές είναι ένα πολύ σημαντικό θέμα. Για τον λόγο αυτό, στα [BFM05, BFM04] ορίζονται τα αποκαλούμενα *άνευ απωλειών* (lossless) και *προς επικύρωση* (validating) σχήματα απεικόνισης, τα οποία επιτυγχάνουν την ορθότητα της παραπάνω μετατροπής.

Αποθήκευση XML δεδομένων ανεξαρτήτως XML σχήματος

Στις τεχνικές *αποθήκευση XML δεδομένων ανεξαρτήτως XML σχήματος* (Schema Oblivious XML Storage) [TVB+02, DAYF04, Pan02, TDCZ02, HLM03, YASU01, Kud02, SSK+01, AYS02, BFM04, FK99, DFS99, SKWW00], επιχειρείται η εύρεση ενός σχεσιακού σχήματος για την αποθήκευση XML κειμένων, ανεξάρτητα από την ύπαρξη ή όχι κάποιου σχήματος για τα XML δεδομένα. Για τον λόγο αυτό, σε ότι αφορά τον σχεδιασμό του σχεσιακού σχήματος, επιλέγεται ένα γενικό σχεσιακό σχήμα για τα XML δεδομένα, ενώ θα πρέπει να ληφθεί υπόψη και το πρόβλημα της μετατροπής των XML ερωτημάτων σε SQL ερωτήματα μέσω κατάλληλων αλγορίθμων μετατροπής.

Ως εναλλακτική προσέγγιση των τεχνικών αποθήκευση XML δεδομένων ανεξαρτήτως XML σχήματος, υπάρχουν μερικές υλοποιήσεις [Cor] που χρησιμοποιούν τα λεγόμενα Character Large Objects (CLOBs), για την αποθήκευση των XML κειμένων σε σχεσιακές βάσεις δεδομένων (opaque approach). Σύμφωνα με αυτή την προσέγγιση, ένας πίνακας του σχεσιακού σχήματος (document table) περιέχει ως εγγραφή έναν μοναδικό προσδιοριστή ενός κειμένου (document ID) μαζί με μία CLOB στήλη στην οποία αποθηκεύεται ολόκληρο το XML κείμενο (xml file) [Kud02].

Στο [TDCZ02], παρουσιάζονται δύο τεχνικές αποθήκευσης XML δεδομένων ανεξαρτήτως XML σχήματος . Η πρώτη ονομάζεται ”Edge Approach”. Με βάση αυτή την τεχνική, ο κατευθυνόμενος γράφος ενός XML αρχείου αποθηκεύεται σε έναν και μοναδικό πίνακα (edge table). Σε κάθε κόμβος του κατευθυνόμενου γράφου αντιστοιχίζεται ένας μοναδικός κωδικός αριθμός (id). Κάθε εγγραφή στον Edge table αντιστοιχεί σε μία *ακμή* (edge) του κατευθυνόμενου γράφου και περιέχει τους μοναδικούς κωδικούς αριθμούς των δύο κόμβων που συνδέονται μεταξύ τους με αυτή την ακμή, την *ετικέτα* (tag) του κόμβου όπου καταλήγει η ακμή και έναν αύξοντα αριθμό (ordinal number) που χρησιμοποιείται για την κωδικοποίηση της σειράς που κατέχει ένας κόμβος παιδί μεταξύ των αδερφικών του κόμβων. Όταν ένα στοιχείο έχει μονάχα ένα παιδί και είναι τύπου κειμένου (text), το κείμενο ενσωματώνεται στον κόμβο γονέα (inlined). Η δεύτερη προσέγγιση ονομάζεται ”Attribute Approach”. Η προσέγγιση αυτή είναι μία οριζόντια τμηματοποίηση της Edge approach με βάση το πεδίο της ετικέτας. Εγγραφές με διαφορετικά ονόματα αποθηκεύονται σε διαφορετικούς πίνακες στο σχεσιακό σχήμα.

Η γλώσσα επερωτήσεων ”XML-SQL” όπως αναφέρεται στο [Pan02], παρέχει

μία μέθοδο σύμφωνα με την οποία ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων χρησιμοποιείται για την εκτέλεση ερωτημάτων σε ημιδομημένα δεδομένα. Η μέθοδος χρησιμοποιεί έναν συνδυασμό από σχεσιακές και ημιδομημένες τεχνικές για την επεξεργασία των XML κειμένων. Αρχικά, ένα XML κείμενο *αναλύεται* (parsing) και απεικονίζεται με τη μορφή ενός edge table σε μία σχεσιακή βάση δεδομένων. Στον edge table, κάθε κόμβος (element, attribute, text) χαρακτηρίζεται από έναν μοναδικό αύξοντα αριθμό (id). Στη συνέχεια, ένα σύνολο από πίνακες που ονομάζονται *πίνακες μονοπατιών* (path tables) δημιουργούνται με βάση τα δεδομένα που αποτελούν τα αποτελέσματα εκτέλεσης σχετικού ερωτήματος (το ερώτημα αυτό αναφέρονται ως ”answer query”), ενώ σχηματίζεται ένα SQL ερώτημα (ο SQL ορισμός του ”answer query”) αναφορικά με τους πίνακες μονοπατιών. Μετά την εκτέλεση του ερωτήματος αυτού, ένας νέος πίνακες που ονομάζεται ”answer table” παράγεται και αποτελεί το πρώτο τμήμα ενός ”XML-SQL” ερωτήματος. Το δεύτερο τμήμα του ”XML-SQL” ερωτήματος περιλαμβάνει τον ”XML-SQL κανόνα” (XML-SQL rule) που θέτει τα κριτήρια σύμφωνα με τα οποία κατασκευάζεται το XML αποτέλεσμα του ερωτήματος, με βάση τα δεδομένα που περιέχει ο ”answer table”.

Σύμφωνα με την προσέγγιση που αναφέρεται στο [DFS99], ένας *αλγόριθμος εξόρυξης δεδομένων* (data-mining algorithm) για ημιδομημένα δεδομένα εφαρμόζεται στα (εκφρασμένα με έναν γράφο) XML δεδομένα, ενώ τα αποτελέσματα της φάσης κατά την εξόρυξη δεδομένων χρησιμεύουν στην παραγωγή ενός σχεσιακού σχήματος και της σχετικής απεικόνισης που προκύπτει βάσει της STORED (Semistructured TO Relational Data) γλώσσας διατύπωσης ερωτημάτων. Αφού το σχεσιακό σχήμα δημιουργηθεί, η STORED απεικόνιση μεταφράζει κάθε στιγμιότυπο των ημιδομημένων δεδομένων στη μορφή του σχήματος αυτού. Η απεικόνιση είναι πάντοτε άνευ απωλειών, δηλαδή τμήματα των ημιδομημένων δεδομένων που δεν αντιστοιχούν στο σχεσιακό σχήμα, αποθηκεύονται σε έναν *γράφο υπερχείλισης* (overflow graph). Οι STORED απεικονίσεις εκφράζονται σαν ερωτήματα (στην STORED γλώσσα), τα οποία παράγονται μέσω του αλγορίθμου εξόρυξης δεδομένων, ενώ τα αποτελέσματα που προκύπτουν από την εκτέλεση των ερωτημάτων αυτών αποθηκεύονται στο σχεσιακό σχήμα.

Στο σύστημα XRel [YASU01], τα *μονοπάτια* (paths) αποτελούν τη δομική μονάδα βάσει της οποίας μπορεί να γίνει η αποδόμηση ενός XML δένδρου. Για κάθε κόμβο των XML δεδομένων, εκτός της ρίζας, οι πληροφορίες για τα μονοπάτια

(από τη ρίζα μέχρι τον εκάστοτε κόμβο) αποθηκεύονται στους σχεσιακούς πίνακες, σύμφωνα με τον τύπο του κόμβου. Κατά το XRel, τα XML κείμενα που εκφράζονται με την μορφή *εκφράσεων μονοπατιών* (path expressions) αποθηκεύονται σε προκαθορισμένα και σταθερά σχεσιακά σχήματα, χωρίς να χρειάζεται η χρήση των αντίστοιχων DTDs. Επιπλέον, για την επεξεργασία των XML ερωτημάτων, παρουσιάζεται ένας αλγόριθμος μετατροπής ενός βασικού συνόλου από XPath εκφράσεις σε SQL ερωτήματα.

Με βάση το [FK99], θεωρείται ότι ένα XML κείμενο μπορεί να αναπαρασταθείς σαν ένας διατεταγμένος και σημασμένος (labeled) κατευθυνόμενος γράφος. Κάθε XML στοιχείο απεικονίζεται στον γράφο ως κόμβος, ο οποίος έχει σημανθεί με έναν μοναδικό αριθμό (id). Σύμφωνα με αυτή την ιδέα, έχουν προταθεί διάφορες προσεγγίσεις ως προς την αποθήκευση XML δεδομένων σε σχεσιακές βάσεις δεδομένων, χωρίς να χρειάζεται η χρήση XML σχήματος. Κατά παρόμοιο τρόπο με το [TDCZ02] που αναφέρθηκε παραπάνω, μία πρώτη προσέγγιση αναφέρεται ως "Edge approach". Κατά την προσέγγιση αυτή, όλες οι ακμές του γράφου που αναπαριστά ένα XML κείμενο, αποθηκεύονται σε έναν και μοναδικό πίνακα. Αυτή τη φορά ωστόσο, οι εγγραφές του Edge table περιέχουν τα ids των στοιχείων προέλευσης (source) και προορισμού (target) της κάθε ακμής του γράφου, το όνομα (label) της ακμής, έναν *σηματοδότη* (flag) που σηματοδοτεί το πότε η ακμή αντιπροσωπεύει έναν εσωτερικό κόμβο ή οδηγεί σε τιμή (δηλ. φύλλο ή leaf) και έναν αύξοντα αριθμό (ordinal number) καθώς οι ακμές είναι διατεταγμένες. Στη συνέχεια αναφέρεται η *Δυαδική προσέγγιση* (Binary approach), σύμφωνα με την οποία όλες οι ακμές με το ίδιο όνομα αποθηκεύονται σε έναν ξεχωριστό πίνακα (οριζόντια κατάτμηση του Edge table της πρώτης προσέγγισης, χρησιμοποιώντας το όνομα της ακμής σαν χαρακτηριστικό κατάτμησης). Τέλος παρουσιάζεται η ονομαζόμενη *Καθολική προσέγγιση* (Universal approach), με βάση την οποία παράγεται ένας μοναδικός πίνακας για την αποθήκευση όλων των ακμών (ο πίνακας αυτός αποτελεί το αποτέλεσμα ενός outer join των δυαδικών πινάκων). Ακόμα, προτείνονται δύο εναλλακτικοί τρόποι που αφορούν την απεικόνιση των τιμών του XML κειμένου (α) αποθηκεύοντας τις τιμές σε ξεχωριστούς πίνακες (Value tables) ή (β) αποθηκεύοντας τις τιμές μαζί με τις ακμές (inlining).

Στο σχεσιακό μοντέλο αποθήκευσης που αναφέρεται στο [Kud02], χρησιμοποιεί έναν κατάλληλα διαμορφωμένο και *δομικά προσανατολισμένο* (structure-oriented) αλγόριθμο μαζί με μία δομή γράφου, η οποία βασίζεται στο XML-QL

μοντέλο. Η δομή του γράφου αναπαρίσταται από έναν edge table που έχει εμπλουτιστεί με κάποιες πληροφορίες προκειμένου να υπάρχει διαφοροποίηση μεταξύ των κόμβων προορισμού (target) μίας ακμής, οι οποίοι μπορεί να είναι στοιχεία (elements), γνωρίσματα (attributes) ή τιμές (content). Τα φύλλα (οι κόμβοι που δεν αποτελούν αφετηρία κάποιας ακμής) περιλαμβάνονται επίσης στον edge table και οι τιμές τους αποθηκεύονται σε έναν πίνακα φύλλων (leaf table). Αντίστοιχα, οι τιμές των γνωρισμάτων διαχωρίζονται και τοποθετούνται σε έναν πίνακα γνωρισμάτων (attribute table). Έτσι, το σχεσιακό σχήμα αποτελείται τελικά από τους παρακάτω πίνακες:

Edge (sourceid, targetid, leafid, attrid, docid, type, name, depth)

Leaf (id, value)

Attr (id, value)

Doc (id, value)

Η γλώσσα επερωτήσεων που χρησιμοποιείται βασίζεται στην XML-QL, ενώ κάθε ερώτημα χρειάζεται να μεταφραστεί στην αντίστοιχη SQL εντολή προκειμένου να εκτελεστεί από το εκάστοτε RDBMS.

Προσεγγίσεις που καλύπτουν τόσο την ύπαρξη όσο και την απουσία XML σχήματος

Σε μερικές περιπτώσεις, οι τεχνικές οι οποίες χρησιμοποιούνται για την γεφύρωση του XML προτύπου με τα συστήματα σχεσιακών βάσεων δεδομένων μπορούν να εφαρμόζονται ανεξάρτητα από την ύπαρξη ή όχι των πληροφοριών που προκύπτουν από κάποιο XML σχήμα. Αυτές οι προσεγγίσεις περιλαμβάνουν τόσο τις απεικονίσεις όσο και την υποβολή ερωτημάτων προς XML δεδομένα που είναι αποθηκευμένα σε ένα σχεσιακό σχήμα.

Στο [AYS02], οι MXM και IMXM παρουσιάζονται σαν ένα σχήμα απεικόνισης και ένα ενδιάμεσο περιβάλλον αλληλεπίδρασης (interface) αποτελούμενο από βιβλιοθήκες διαδικασιών (API) αντίστοιχα, για να ορίσουν και να υποβάλουν ερωτήματα σε απεικονίσεις XML δεδομένων προς σχεσιακά δεδομένα. Μία απεικόνιση εκφράζεται ως ένα *στιγμιότυπο* (instance) της MXM. Η MXM είναι μία γλώσσα απεικονίσεων, βασισμένη στην XML, που μπορεί να εκφράσει τα περισσότερα από τις υπάρχουσες απεικονίσεις της XML προς σχεσιακά δεδομένα. Με την MXM, μετά από την απαιτούμενη επεξεργασία, μπορούν να εκφραστούν απεικονίσεις για κείμενα για τα οποία δεν υπάρχουν πληροφορίες για το XML σχήμα

που ακολουθούν ή για κείμενα που ακολουθούν κάποιο DTD ή XML σχήμα. Η αποθήκευση των απεικονίσεων που προκύπτουν γίνεται σε μία *αποθήκη απεικονίσεων* (mapping repository). Το IMXM είναι ένα περιβάλλον το οποίο επιτρέπει την υποβολή ερωτημάτων (querying) σε πληροφορίες οι οποίες περιέχονται σε μία MXM απεικόνιση. Το περιβάλλον αυτό έχει σχεδιαστεί σαν μία βιβλιοθήκη από διαδικασίες (functions σε γλώσσα C με τη χρήση ODBC ή Java με τη χρήση JDBC), η οποία μπορεί εύκολα να χρησιμοποιηθεί από οποιαδήποτε εφαρμογή που χρειάζεται να αποκτήσει πρόσβαση στη σχετική απεικόνιση. Ακόμα, το MXM είναι επεκτάσιμο και μπορεί να ενσωματώσει νέες απεικονίσεις. Η δημιουργία του σχεσιακού σχήματος, η μετατροπή των XQuery ερωτημάτων σε SQL και η αλληλεπίδραση των XML εφαρμογών με τη σχεσιακή βάση δεδομένων γίνονται μέσω του IMXM περιβάλλοντος.

Το ShreX αποτελεί ένα ελεύθερα διαθέσιμο σύστημα *αποδόμησης* (shredding), *φόρτωσης* (loading) και υποβολής ερωτημάτων σε XML κείμενα με τη χρήση σχεσιακών βάσεων δεδομένων [DAYF04]. Στο ShreX, μία απεικόνιση μπορεί να προσδιοριστεί με την προσθήκη *σημάνσεων* (annotations) στο XML σχήμα, οι οποίες υποδεικνύουν πως τα στοιχεία και τα γνωρίσματα πρέπει να αποθηκευτούν σε μία σχεσιακή βάση. Συνδυάζοντας διαφορετικές τέτοιες σημάνσεις, μπορεί να εκφραστεί μία μεγάλη ποικιλία από απεικονίσεις, συμπεριλαμβανομένου αυτών που προκύπτουν από όλες τις στρατηγικές απεικόνισης που προτείνονται στην σχετική βιβλιογραφία, όπως και από στρατηγικές που υποστηρίζονται από κατασκευαστές βάσεων δεδομένων. Έτσι, το ShreX παρέχει γενικές (mapping-independent [schema based ή schema-less]) διαδικασίες για την διαδικασία της αποδόμησης XML κειμένων και την μετατροπή ερωτημάτων, κάτι που είναι δυνατόν λόγο μίας βιβλιοθήκης διαδικασιών (API), η οποία παρέχει πρόσβαση στις πληροφορίες απεικόνισης. Στο σύστημα Shrex, περιλαμβάνεται ένας *επεξεργαστής σημάνσεων* (annotation processor), ο οποίος αναλύει ένα εμπλουτισμένο με σημάνσεις XML σχήμα, ελέγχει την ορθότητα της απεικόνισης και δημιουργεί το αντίστοιχο σχεσιακό σχήμα. Σημειώνεται ότι οι πληροφορίες απεικόνισης παραμένουν διαρκώς στην αποθήκη απεικονίσεων. Τέλος, το τμήμα του συστήματος που αποδομεί τα κείμενα (document shredder) δέχεται ως είσοδο ένα κείμενο και χρησιμοποιεί την βιβλιοθήκη διαδικασιών για την πρόσβαση των πληροφοριών στην αποθήκη απεικονίσεων, προκειμένου να δημιουργήσει τις *εγγραφές* (tuples) και να συμπληρώσει τους πίνακες στο σχεσιακό σχήμα. Η αποθήκη απεικονίσεων

είναι επίσης προσβάσιμο και κατά την μετατροπή των ερωτημάτων, οπότε και η σχετική διαδικασία (query translator) παράγει SQL ερωτήματα από XML ερωτήματα.

XML δεικτοδότηση και διάταξη

Ένα πολύ σημαντικό ζήτημα που αφορά την αναπαράσταση των προς αποθήκευση XML δεδομένων ενός XML κείμενου σε μία σχεσιακή βάση, είναι το ζήτημα της *δεικτοδότησης* (indexing) και της *διάταξης* (ordering) των XML κόμβων. Από δομικής άποψης, είναι γνωστό, ότι η κύρια διαφορά μεταξύ των XML και των σχεσιακών δεδομένων αφορά το γεγονός ότι τα XML δεδομένα είναι δομημένα σε πολλά επίπεδα, αντίθετα με τα σχεσιακά δεδομένα που παρουσιάζουν μία σχετικά μονοδιάστατη (flat) δομή. Ανακύπτει λοιπόν το ερώτημα, πως είναι δυνατόν να υποστηριχθεί με αποτελεσματικό τρόπο η διάταξη των XML δεδομένων από το σχεσιακό μοντέλο, δεδομένου ότι στα σχεσιακά σχήματα δεν ακολουθείται κάποια παρόμοια διάταξη;

Σύμφωνα με το [TVB+02], τα διατεταγμένα δεδομένα ενός XML μοντέλου μπορούν να υποστηριχθούν από μία σχεσιακή βάση δεδομένων. Αυτό επιτυγχάνεται κωδικοποιώντας την διάταξη σαν μία τιμή στα αποθηκευμένα δεδομένα. Έτσι, σύμφωνα με την δενδροειδή δομή των XML κειμένων προτείνονται τρεις *άνευ απωλειών μέθοδοι κωδικοποίησης* (lossless encoding methods) για την αναπαράσταση της XML διάταξης στο σχεσιακό μοντέλο:

1. *Καθολική διάταξη (Global order)*: Σε κάθε κόμβο του XML κειμένου, αντιστοιχεί ένας αριθμός που αντιπροσωπεύει την *απόλυτη θέση* (absolute position) του κόμβου αυτού.

2. *Τοπική διάταξη (Local order)*: Σε κάθε κόμβο του XML κειμένου, αντιστοιχεί ένας αριθμός που αντιπροσωπεύει την *σχετική θέση* (relative position) του κόμβου αυτού μεταξύ των *αδερφικών* του *κόμβων* (siblings).

3. *Dewey διάταξη*: Σε κάθε κόμβο του XML κειμένου, αντιστοιχεί ένα *διάνυσμα* (vector) που αντιπροσωπεύει το μονοπάτι από την ρίζα του κειμένου μέχρι τον κόμβο αυτόν.

Στο [LM01], παρουσιάζεται ένα σύστημα για την δεικτοδότηση και την αποθήκευση XML δεδομένων βασισμένο σε ένα σχήμα δεικτοδότησης για *στοιχεία*

(elements) και *γνωρίσματα* (attributes) XML κειμένων. Το σχήμα αυτό οποίο βοηθάει στην αποδοτικότερη αναζήτηση δεδομένων με βάση κάποια τιμή ή τα δομικά χαρακτηριστικά της αναζητούμενης πληροφορίας. Για τον λόγο αυτό, το παραπάνω σχήμα δεικτοδότησης καθορίζει άμεσα την σχέση προγόνου-απογόνου μεταξύ των στοιχείων μέσα στην ιεραρχία των XML δεδομένων.

Ακόμα, το σύστημα XISS/R [HLM03], είναι βασισμένο στο σχήμα δεικτοδότησης του XISS (XML Indexing and Storage System), το οποίο εμπεριέχει την ιεραρχική δομή των XML δεδομένων και παρέχει τη δυνατότητα της αποθήκευσης δεδομένων και της επεξεργασίας ερωτημάτων ανεξάρτητα από την επιμέρους δομή των δεδομένων. Το σύστημα αυτό αποτελεί ένα XML σύστημα δεικτοδότησης και αποθήκευσης, το οποίο μπορεί να λειτουργήσει στην κορυφή κάποιου εμπορικού συστήματος σχεσιακών βάσεων δεδομένων. Αναλύοντας περαιτέρω το σύστημα XISS-R, θα μπορούσαμε να αναφέρουμε ότι περιλαμβάνει τα παρακάτω βασικά στοιχεία:

1. Μία απεικόνιση των XML δεδομένων προς το σχεσιακό σχήμα
2. Μία XPath *μηχανή αναζήτησης* (query engine)
3. Ένα κατάλληλα διαμορφωμένο για τους χρήστες ενδιάμεσο περιβάλλον αλληλεπίδρασης που είναι προσβάσιμο μέσω του παγκόσμιου ιστού (WEB-based)

Η απεικόνιση των XML δεδομένων προς το σχεσιακό σχήμα υλοποιείται με τη χρήση του ονομαζόμενου *εκτεταμένου προ-διατεταγμένου σχήματος αριθμοδότησης* (extended preorder numbering schema). Αυτό το σχήμα αριθμοδότησης παρέχει μία μέθοδο σύμφωνα με την οποία τα δενδροειδώς δομημένα δεδομένα κωδικοποιούνται σε ζεύγη ακεραίων αριθμών, ανεξάρτητα από τα περιεχόμενα των δεδομένων αυτών. Υπάρχουν δύο σχεσιακά σχήματα που χρησιμοποιούν καλύτερα αυτό το σχήμα αριθμοδότησης. Σχήμα Α: Το XISS/R διαχωρίζει τους κόμβους σε τρεις κατηγορίες , *στοιχεία* (elements), *γνωρίσματα* (attributes) και *κόμβους κειμένου* (text nodes). Στη συνέχεια οι κόμβοι αυτοί καταχωρούνται αντίστοιχα στους πίνακες Document Table, Element Table, Attribute Table και Text Table του σχεσιακού σχήματος. Σχήμα Β: Οι κόμβοι διαχωρίζονται σε διαφορετικούς πίνακες ανάλογα με το είδος τους, όμοια με το Σχήμα Α (στοιχεία, γνωρίσματα ή κόμβοι κειμένου). Επιπρόσθετα, γίνεται ένας οριζόντιος διαχωρισμός των στοιχείων

και των γνωρισμάτων με βάση το όνομα της ετικέτας τους (tag-name). Έτσι, ένας element ή attribute table δημιουργείται για κάθε μοναδικό όνομα.

Από την άλλη πλευρά, η μηχανή αναζήτησης της Xpath, δέχεται XPath ερωτήματα και παράγει SQL ερωτήματα προκειμένου αυτά να σταλούν προς εκτέλεση στο σύστημα της σχεσιακής βάσης δεδομένων. Τα αποτελέσματα της εκτέλεσης μορφοποιούνται και αποστέλλονται από την μηχανή αναζήτησης στον χρήστη μέσω του ενδιάμεσου περιβάλλοντος αλληλεπίδρασης.

Τέλος, το ενδιάμεσο περιβάλλον αλληλεπίδρασης επιτρέπει στους χρήστες να θέτουν στο XISS/R σύστημα Xpath ερωτήματα και να λάβουν τα αποτελέσματα της εκτέλεσής τους μέσω του διαδικτύου.

Τεχνικές αποθήκευσης βασισμένες σε μονοπάτια

Μία σημαντική βελτίωση των τεχνικών αποθήκευσης XML δεδομένων σε σχεσιακές βάσεις δεδομένων, προκειμένου να επιτευχθεί ταχύτερη εκτέλεση XML ερωτημάτων προς τα δεδομένα αυτά αποτέλεσε η δημιουργία σχεσιακών σχημάτων που περιελάμβαναν τη δυνατότητα αποθήκευσης μονοπατιών (path based τεχνικές). Σύμφωνα με τις τεχνικές αυτές, τα πιθανά μονοπάτια του XML δένδρου που αναπαριστά τα XML δεδομένα αποθηκεύονται σε έναν ξεχωριστό πίνακα της σχεσιακής βάσης. Όπως αναφέρεται στο [YASU01], πληροφορίες για κάθε μονοπάτι από την ρίζα του του XML δένδρου μέχρι κάποιον κόμβο στοιχείου ή γνωρίσματος, μπορούν να αναπαρασταθούν με τη μορφή εκφράσεων μονοπατιού (path expressions), ενώ αποθηκεύονται σε έναν ξεχωριστό πίνακα του σχεσιακού σχήματος. Με βάση το [PBM04], το προτεινόμενο σύστημα αποθηκεύει μόνο τα μονοπάτια και τις *τιμές* (values) των *φύλλων* (leaf nodes) προκειμένου να περιορίσει τα θ-joins κατά την εκτέλεση σχετικών ερωτημάτων που βασίζονται σε XPath εκφράσεις.

XML πρότυπα δένδρου

Σύμφωνα με τη βιβλιογραφία [HD13, CJLP03, JLST01], ένα *πρότυπο δένδρου* (tree pattern ή tree pattern query (TPQ)) μπορεί να μοντελοποιήσει ένα ερώτημα χρήστη προς κάποιο XML δένδρο. Πιο συγκεκριμένα, ένα πρότυπο δένδρου είναι μία γραφική αναπαράσταση ενός XML ερωτήματος, που παρέχει έναν εύκολο τρόπο για τον προσδιορισμό των αποτελεσμάτων ενός ερωτήματος. Η διαδικα-

σία εκτέλεσης ενός XML ερωτήματος που εκφράζεται με την μορφή ενός πρότυπου δένδρου ονομάζεται *ταυτοποίηση πρότυπου δένδρου* (tree pattern matching). Η διαδικασία της ταυτοποίησης πρότυπου δένδρου απεικονίζει (mapping) ένα πρότυπο δένδρου πάνω στη δομή και το περιεχόμενο ενός XML δένδρου, ενώ το αποτέλεσμα της απεικόνισης αυτής αποκαλείται *παριστάμενο δένδρο* (witness tree).

Διατήρηση πληροφορίας (Information Preservation)

Ως γνωστό, σε ένα σύστημα αποθήκευσης XML βάσει απεικονίσεων, όπου ένα XML κείμενο αναπαριστάται ως δεδομένα κάποιας σχεσιακής βάσης, τα διάφορα XML ερωτήματα (αλλά και ενημερώσεις) μετατρέπονται στα αντίστοιχα SQL ερωτήματα τα οποία απευθύνονται στη βάση δεδομένων.

Όπως αναφέρθηκε και παραπάνω, σύμφωνα με τα [BFM05] και [BFM04] ορίζονται τα άνευ απωλειών σχήματα απεικόνισης αλλά και τα *προς επαλήθευση σχήματα απεικόνισης* (validating mapping schemas). Πιο συγκεκριμένα, μία απεικόνιση πρέπει να χαρακτηρίζεται ως ”άνεφ απωλειών”, πράγμα που σημαίνει ότι το XML κείμενο θα πρέπει να είναι πλήρως ανακτήσιμο από το σχεσιακό σχήμα στο οποίο είναι αποθηκευμένο. Επιπλέον, κατά τη διεξαγωγή ενημερώσεων, θα πρέπει να είναι δυνατή η επαλήθευση ότι μία τέτοιου είδους πράξη οδηγεί σε ένα ορθά εκπεφρασμένο κείμενο, προτού η πράξη αυτή εφαρμοστεί στη σχεσιακή βάση δεδομένων. Μόνο οι ενημερώσεις που έχουν σαν αποτελέσματα έγκυρα κείμενα θα πρέπει να επιτρέπονται. Η έννοια της απεικόνισης της XML προς το σχεσιακό σχήμα ορίζεται ως ένα σύνολο τριών στοιχείων (s,p,S), όπου S είναι ένα σχεσιακό σχήμα, s είναι μία διαδικασία απεικόνισης που αντιστοιχεί στοιχεία σχεσιακών βάσεων σε στοιχεία XML κειμένων και p είναι μία διαδικασία *έκδοσης* (publishing) που αντιστοιχεί, αντιστρόφως με την s, στοιχεία XML κειμένων σε στοιχεία σχεσιακών βάσεων. Ορίζοντας κλάσεις σχημάτων απεικόνισης, με βάση γλώσσες που χρησιμοποιούνται για τον προσδιορισμό των s, S και p, παρουσιάζεται η XDS κλάση (βασισμένη στην XQuery γλώσσα, σε συνδυασμό με ένα βασικό πλαίσιο έκδοσης δεομένων όπως είναι το SilkRoute που περιγράφεται παρακάτω). Με την XDS μπορούν να αναπαρασταθούν όλα τα γνωστά σχήματα απεικόνισης (Edge mapping κτλ). Ακόμα, το Edge++ παρουσιάζεται σαν ένα ”άνεφ απωλειών” και ”προς επελήθευση” σχήμα απεικονισης με τη χρήση της XDS.

1.1.2 Έκδοση XML δεδομένων από σχεσιακές βάσεις δεδομένων

Εισαγωγή

Το πρόβλημα της *έκδοσης XML δεδομένων* από σχεσιακές βάσεις (XML Publishing), συνίσταται στην προσπάθεια να μεταχειριστούμε τα υπάρχοντα σχεσιακά δεδομένα ως XML δεδομένα. Αυτό μπορεί να επιτευχθεί ορίζοντας μία *XML όψη* (XML view) των σχεσιακών δεδομένων [FKS+02a, SSB+00, FKS+02b, SKS+01]. Οι XML όψεις είναι ιδεατές ενδιάμεσες δομές (virtual middlewares) πάνω στις οποίες μπορούν να υποβληθούν XML ερωτήματα, ενώ δημιουργούνται με βάση την σχεσιακή βάση δεδομένων, η οποία αποτελεί το σχετικό υπόβαθρο. Οι εν λόγω όψεις μπορούν επίσης να εκφράζουν ένα μόνο κομμάτι της σχεσιακής βάσης, σύμφωνα με τις ανάγκες μίας XML εφαρμογής [FKS+02a]. Τα XML ερωτήματα που παράγονται από τις XML εφαρμογές, ώστε να εκτελεστούν πάνω σε XML όψεις, μετατρέπονται σε αντίστοιχα SQL ερωτήματα και τα αποτελέσματα της εκτέλεσής τους επιστρέφονται στην XML εφαρμογή, αφού πρώτα μετασχηματιστούν ξανά σε XML μορφή. Για την κατασκευή XML ερωτημάτων μπορούν να χρησιμοποιηθούν διάφορες γλώσσες επερωτήσεων (Query Languages) όπως η Xpath ή η XQuery [CON99a, CON01].

Σε ότι αφορά τον ορισμό μίας XML όψης, υπάρχουν δύο βασικές προσεγγίσεις. Θεωρώντας ότι ένα XML κείμενο εκφράζει ένα *καθολικό σχήμα* (Global schema) (όπου τίθενται τα ερωτήματα των χρηστών) και το υφιστάμενο σχεσιακό σχήμα ένα *τοπικό σχήμα* (Local schema) (κάτω από το καθολικό σχήμα), οι παρακάτω προσεγγίσεις μπορούν να χρησιμοποιηθούν:

- **Global as View (GAV) προσέγγιση**

  Στην GAV προσέγγιση, η XML όψη (καθολικό σχήμα) ορίζεται ως μία όψη πάνω από το σχεσιακό σχήμα (τοπικό σχήμα). Αυτή η προσέγγιση χρησιμοποιείται από συστήματα όπως το SilkRoute [FKS+02a] και το Xperanto [SKS+01]. Στο SilkRoute μία XML όψη δημιουργείται σαν ένας συνδυασμός από SQL ερωτήματα. Κάθε σύνολο από SQL ερωτήματα που μπορούν αν ορίσουν μία XML όψη ονομάζεται ”plant” και η επιλογή του πιο αποδοτικού plant αποτελεί το πρόβλημα της μεγιστοποίησης της απόδοσης για το σύστημα. Το Xperanto, εκτός των κλασικών σχεσιακών σχημάτων, υπο-

στηρίζει και αντικειμενοστραφείς σχεσιακές δομές (object-relational structures), ενώ δίνει τη δυνατότητα στον χρήστη να εκδίδει αντικειμενοστρεφή σχεσιακά δεδομένα σε μορφή XML, χωρίς ο ίδιος να χρειάζεται να ασχοληθεί με το σχεσιακό σχήμα της βάσης δεδομένων ή την σύνταξη SQL ερωτημάτων.

Στο [CKS+00] εξηγείται ότι ο σκοπός του έργου Xperanto ως ενδιάμεση πλατφόρμα είναι η δυνατότητα να παρέχει ''ερωτήσημες'' (query-able) XML όψεις (προκαθορισμένες όψεις) πάνω από μία υφιστάμενη σχεσιακή βάση δεδομένων. Οι χρήστες μπορούν τότε να θέτουν τα ερωτήματά τους και να αναδομούν τα XML δεδομένα χρησιμοποιώντας μία XML γλώσσα επερωτήσεων, χωρίς να χρειάζεται να ασχοληθούν με τους σχεσιακούς πίνακες και την γλώσσα SQL. Το σύστημα Xperanto μεταφράζει τα XML ερωτήματα σε SQL ερωτήματα, λαμβάνει τα δομημένα αποτελέσματα των ερωτημάτων, τα μετατρέπει σε XML και τα επιστρέφει υπό μορφή XML κειμένων στους χρήστες και στις εφαρμογές του συστήματος.

Η αρχιτεκτονική επεξεργασίας των ερωτημάτων όπως αυτή περιγράφεται στο [SKS+01], υλοποιείται υπό την έννοια της ενδιάμεσης πλατφόρμας του συστήματος Xperanto. Αρχικά, το Xperanto δημιουργεί αυτόματα μία *προκαθορισμένη XML όψη* (default XML view), η οποία είναι μία χαμηλού επιπέδου XML όψη της υφιστάμενης σχεσιακής βάσης. Οι χρήστες μπορούν τότε να ορίζουν τις δικές τους όψεις πάνω από αυτή την προκαθορισμένη όψη χρησιμοποιώντας την γλώσσα XQuery, η οποία περιέχει Xpath εκφράσεις. Το κυριότερο πλεονέκτημα της προσέγγισης αυτής είναι ότι μία συμβατική γλώσσα επερωτήσεων χρησιμοποιείται για την υποβολή ερωτημάτων αλλά και την κατασκευή των όψεων. Αυτό έρχεται σε αντίθεση με άλλες προσεγγίσεις (πχ. SilkRoute), όπου αποκλειστικά κάποια προκαθορισμένη γλώσσα χρησιμοποιείται για τον καθορισμό κάποιας όψης της υφιστάμενης σχεσιακής βάσης δεδομένων.

- **Local as View (LAV) προσέγγιση**

  Κατά την LAV προσέγγιση, το τοπικό σχήμα εκφράζεται ως όψεις του καθολικού σχήματος. Στο σύστημα Agora [MFK01] περιγράφεται ένα γενικό, ιδεατό σχεσιακό σχήμα που μοντελοποιεί με έναν γενικό τρόπο τη δομή ενός XML κειμένου. Το τοπικό σχεσιακό σχήμα ορίζετε τότε ως όψεις πάνω από

το γενικό, ιδεατό σχήμα.

- **Global-Local as View (GLAV) προσέγγιση**

  Σε ορισμένες περιπτώσεις, τόσο GAV όσο και LAV σχήματα, είναι δυνατόν να συνδυαστούν, δίνοντας ως αποτέλεσμα κάποια υβριδικά συστήματα [DT03].

Υποβολή ερωτημάτων σε XML όψεις

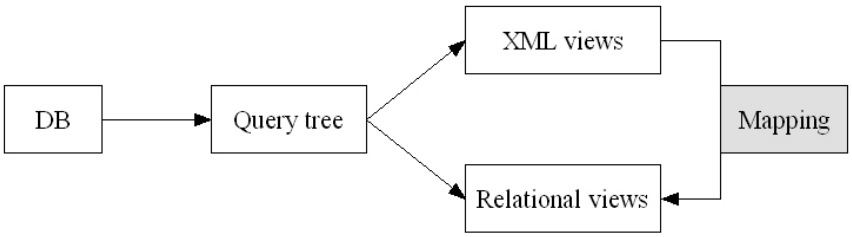
Μέχρι τώρα, η ερευνητική περιοχή που αφορά την υποβολή ερωτημάτων και την εξαγωγή πληροφοριών από δεδομένα που είναι αποθηκευμένα σε σχεσιακές βάσεις δεδομένων, με τη χρήση XML όψεων [FKS+02a, SSB+00, SKS+01, CKN03], αποτελεί ένα πεδίο που έχει μελετηθεί εκτενώς.

Ενδεικτικά, σύμφωνα με την τεχνική που παρουσιάζεται στο [SKS+01], όταν εισάγεται ένα XQuery ερώτημα, αρχικά αναλύεται (parsing) και μετατρέπεται σε μία εσωτερική αναπαράσταση ερωτημάτων, η οποία ονομάζεται ”XML Query Graph Model” (XQGM). Το XQGM αποτελείται από ένα σύνολο (αναφορικά με τη σχεσιακή βάση) *τελεστών* (operators) και (αναφορικά με την XML) *διαδικασιών* (functions). Στην συνέχεια το ερώτημα βελτιστοποιείται (optimization) και αναδιαμορφώνεται σύμφωνα με XML όψεις σε μορφή XQGM οι οποίες υπολογίζονται με βάση το σχεσιακό σχήμα, ενώ το τροποποιημένο XQGM επεξεργάζεται από το ονομαζόμενο ”Computation Pushdown” *υποσύστημα* (module), το οποίο διαχωρίζει το XQGM σε δύο κομμάτια. Το πρώτο κομμάτι περιλαμβάνει όλα εκείνα τα δεδομένα που απαιτούνται προκειμένου να προωθηθούν με τη μορφή SQL στο σύστημα διαχείρισης της σχεσιακής βάσης δεδομένων, ενώ το δεύτερο κομμάτι αποτελείται από μία δομή (tagger graph structure), την οποία χρησιμοποιεί το λεγόμενο ”Tagger Runtime” υποσύστημα για την κατασκευή του XML αποτελέσματος από τα αποτελέσματα του SQL ερωτήματος.

Ενημέρωση των XML όψεων

Πέρα από το πρόβλημα της υποβολής ερωτημάτων σε XML όψεις που έχουν δημιουργηθεί με βάση κάποιο σχεσιακό σχήμα, ενδιαφέρον παρουσιάζει και η μελέτη του προβλήματος της *ενημέρωσης* (updating) των XML όψεων.

Σύμφωνα με την προσέγγιση, η οποία αναλύεται στο [BDH04], μπορούμε να δούμε τον τρόπο με τον οποίο οι XML όψεις εκφραζόμενες μέσω των *δένδρων επερωτήσεως* (query trees) μπορούν να απεικονιστούν σε ένα σύνολο από αντίστοιχες όψεις του σχεσιακού μοντέλου. Τότε, οι ενημερώσεις προς XML όψεις μπορούν επίσης να απεικονιστούν σε ενημερώσεις στις αντίστοιχες σχεσιακές όψεις. Η στρατηγική που ακολουθείται με βάση αυτή την τεχνική περιλαμβάνει την παραγωγή μίας XML όψης των σχεσιακών δεδομένων, εκπεφρασμένης με τη μορφή των δένδρων επερωτήσεως (συμπεριλαμβανομένων αναφορών προς τη σχεσιακή βάση δεδομένων). Στη συνέχεια, χρησιμοποιώντας γλώσσες επερωτήσεων, κατασκευάζονται ερωτήματα ενημέρωσης (update queries) που απευθύνονται στην XML όψη. Με βάση την XML όψη παράγονται σχεσιακές όψεις (σε SQL) και τα ερωτήματα πάνω στην XML όψη αντιστοιχίζονται σε ένα σύνολο από πράξης ενημέρωσης πάνω στις υφιστάμενες σχεσιακές όψεις. Έτσι, είναι πλέον εφικτή η χρήση οποιασδήποτε υπάρχουσας τεχνικής ενημέρωσης μέσω σχεσιακών όψεων, προκειμένου να μεταφραστούν οι αντίστοιχες ενημερώσεις στις κατάλληλες πράξεις ενημέρωσης προς τη σχεσιακή βάση δεδομένων (Σχήμα 1.1).



Σχήμα 1.1: Ενημέρωση XML όψεων

Για τον ίδιο σκοπό, στο [BDH03], η γλώσσα UXQuery (XQuery με επιπρόσθετες εκφράσεις) χρησιμοποιείται για την κατασκευή των XML όψεων.

Το [WMR03] εστιάζει σε ένα πρόβλημα, το οποίο αποκαλείται ”round-trip XML view update problem”. Στην περίπτωση αυτή, δοθέντος ενός XML σχήματος και ενός XML κειμένου, δημιουργείται μία σχεσιακή βάση δεδομένων, με τη χρήση ενός συγκεκριμένου αλγόριθμου ”φόρτωσης” (ακολουθώντας τεχνικές όπως οι inlining, edge ή universal και χρησιμοποιώντας μία κατάλληλη απεικόνιση που ενσωματώνει σχετικούς σημασιολογικούς περιορισμούς). Ένα XML ερώτημα

(με τη μορφή XQuery έκφρασης) προς την σχεσιακή βάση το οποίο αποκαλείται *ερώτημα εξαγωγής* (extraction query) , δημιουργεί μία XML όψη με περιεχόμενο παρόμοιο με το XML κείμενο που χρησιμοποιήθηκε σαν δεδομένο εισόδου στο αλγόριθμο φόρτωσης. Η όψη αυτή ονομάζεται ”twin-view”. Το πρόβλημα που αφορά την ενημέρωση της βάσης δεδομένων μέσω αυτής της twin-view είναι γνωστό σαν το round-trip XML view update πρόβλημα.

1.1.3 Αποθήκευση XML δεδομένων με χρήση της έκδοσης XML δεδομένων

Με βάση το [SSK+01], κάθε τεχνική δημιουργίας ενός σχεσιακού σχήματος περιλαμβάνει το δικό της *μηχανισμό επεξεργασίας ερωτημάτων* (query processor) για τη μετατροπή των XML ερωτημάτων σε SQL ερωτήματα. Στην προτεινόμενη τεχνική, μία από τις διάφορες τεχνικές δημιουργίας σχεσιακού σχήματος (βάσει XML σχήματος ή ανεξαρτήτως XML σχήματος) θα μπορούσε να χρησιμοποιηθεί για την αυτόματη κατασκευή σχεσιακών πινάκων και την αποθήκευση XML κειμένων. Τα XML κείμενα αποδομούνται (shredding) και αποθηκεύονται σαν εγγραφές σε αυτούς τους πίνακες. Επιπρόσθετα και σύμφωνα με τον κατάλληλο αλγόριθμο, δημιουργείται μία δομή (σε XQuery), σε σχέση με τους σχεσιακούς πίνακες, που ονομάζεται *XML όψη επανασύστασης* (reconstruction XML view). Η δομή αυτή (ιδεατά) αναδομεί τα αποθηκευμένα XML κείμενα από τις αποδομημένες εγγραφές. Όπως αναφέρεται στη σχετική μελέτη, η XML όψη επανασύστασης είναι μία δομή που μας επιτρέπει να μεταχειριζόμαστε τα XML κείμενα σαν να ήταν μία XML όψη των σχεσιακών δεδομένων. Αποτέλεσμα αυτού είναι το γεγονός ότι ένα ερώτημα προς τα XML κείμενα που είναι αποθηκευμένα σε σχεσιακές βάσεις, μπορούν να επεξεργαστούν ως ερωτήματα προς την αντίστοιχη XML όψη επανασύστασης. Αυτά με τη σειρά τους μπορούν να διαχειρίζονται αποτελεσματικά από έναν μηχανισμός επεξεργασίας ερωτημάτων (όπως ο μηχανισμός του Xperanto που χρησιμοποιεί προκαθορισμένες XML όψεις), ο οποίος θα χρησιμοποιείται για την επεξεργασία ερωτημάτων προς XML όψεις σχεσιακών δεδομένων [CFI+00, CCS00, FTS00]. Έτσι, ένας μοναδικός μηχανισμός επεξεργασίας ερωτημάτων είναι αρκετός για να παρέχει γενικά τη δυνατότητα υποβολής XML ερωτημάτων προς XML κείμενα, ανεξάρτητα από την τεχνική δημιουργίας του σχεσιακού σχήματος. Επιπλέον, αναφέρεται ότι για κάθε τεχνική που βασίζεται

στο XML σχήμα μία διαφορετική XML όψη επανασύστασης απαιτείται για κάθε DTD. Από την άλλη πλευρά, για τις τεχνικές που είναι ανεξάρτητες από τις πληροφορίες που παρέχουν τα XML σχήματα, μπορεί να χρησιμοποιηθεί η ίδια XML όψη επανασύστασης για κάθε XML κείμενο.

1.1.4 Επιπλέον ερευνητικές περιοχές

Παρόλο που έχει γίνει σημαντική έρευνα γύρω από τα θέματα της αποθήκευσης και της έκδοσης XML δεδομένων, υπάρχουν διάφορα ερευνητικά πεδία τα οποία δεν έχουν ερευνηθεί εκτενώς. Ένα από τα πεδία αυτά, περιλαμβάνει το πρόβλημα της ενημέρωσης των XML δεδομένων. Όπως αναφέρθηκε και παραπάνω, στο [BDH04] παρουσιάζεται το πως είναι δυνατόν ενημερώσεις που πραγματοποιούνται πάνω σε κάποια XML view να απεικονίζονται σε ενημερώσεις πάνω σε αντίστοιχες σχεσιακές όψεις. Επιπλέον, στο [TIHW01] προτείνεται ένα σύνολο από βασικές *πράξεις ενημερώσεως* (update operations), τόσο για διατεταγμένα (ordered) όσο και για μη διατεταγμένα (unordered) XML δεδομένα, με τη χρήση επεκτάσεων της γλώσσας υποβολής XML ερωτημάτων XQuery, προκειμένου να διευκολυνθεί η δημιουργία των πράξεων αυτών.

Σε ότι αφορά τα αναδρομικά (recursive) XML σχήματα, στο [KCKN04] παρουσιάζεται ένας γενικός αλγόριθμος, ο οποίος μετατρέπει ερωτήματα με βάση εκφράσεις μονοπατιών σε SQL για περιπτώσεις όπου παρατηρείται αναδρομή στο XML σχήμα και τα ερωτήματα. Ο αλγόριθμος αυτός μετατρέπει τα XML ερωτήματα σε SQL, όταν τα XML κείμενα έχουν αποθηκευτεί σε κάποια σχεσιακή βάση δεδομένων με τη βοήθεια μίας τεχνικής βάσει XML σχήματος. Παρόλα αυτά δεν υπάρχει δημοσιευμένη κάποια μελέτη που να προτείνει έναν αλγόριθμος μετατροπής XML ερωτημάτων σε SQL ερωτήματα για την περίπτωση που αυτός θα πρέπει να διαχειριστεί αναδρομικά XML σχήματα.

Ακόμα, στο [KKN03] αναφέρεται ότι στην περίπτωση της αποθήκευσης XML δεδομένων ανεξαρτήτως XML σχήματος, η προσοχή των ερευνητών εστιάζεται γενικά σε ερωτήματα με βάση εκφράσεις μονοπατιών, ενώ σε περιπτώσεις προσεγγίσεων που η αποθήκευση των XML δεδομένων γίνεται με βάση το XML σχήμα, δεν υπάρχει δημοσιευμένος κάποιος αλγόριθμος μετατροπής ερωτημάτων (οι μόνες γνωστές προσεγγίσεις είναι μέσω κάποιας μετάπτωσης σε κάποιο σενάριο έκδοσης XML δεδομένων).

1.1.5 XML επεκτάσεις

Τα προηγούμενα χρόνια, διάφορες προσπάθειες για την επέκταση της XML έχουν πραγματοποιηθεί. Αρκετές ερευνητικές εργασίες έχουν ασχοληθεί με την αποθήκευση XML δεδομένων που περιλαμβάνουν χρονικές επεκτάσεις (temporal XML) [WZZ06, AYU00, AYU01]. Στο [LJ88, LM97] μπορούμε να δούμε πως είναι δυνατή η επέκταση της σχεσιακής άλγεβρας προκειμένου να υποστηρίζει ερωτήματα που εκτελούνται πάνω σε χρονικά δεδομένα. Επιπλέον, στο [HvK10] εξηγείται πως είναι δυνατή η υποβολή ερωτημάτων σε XML δεδομένα που εμπεριέχουν επεκτάσεις με βάση τις πιθανότητες (probabilistic XML) χρησιμοποιώντας κάποια ”uncertain” σχεσιακή βάση σαν υπόβαθρο. Με βάση το [NJ02], προτείνεται ένα μοντέλο για την διαχείριση πιθανολογικών (probabilistic) δεδομένων, που αναπαριστώνται μέσω της XML.

Επιπλέον των παραπάνω, προέκυψε η ανάγκη αναπαράστασης XML δεδομένων, ανάλογα με τις συνθήκες οι οποίες ορίζονται από το εκάστοτε περιβάλλον, όπου αναπτύσσονται και εκτελούνται οι διάφορες εφαρμογές του διαδικτύου. Αντίθετα με τα παραδοσιακά περιβάλλοντα σχεσιακών βάσεων δεδομένων όπου το είδος και οι ανάγκες των χρηστών είναι μία σχετικά αναμενόμενη και γνωστή πληροφορία, οι χρήστες του διαδικτύου μπορεί να παρουσιάζουν διαφορετικές ανάγκες και να λειτουργούν κάτω από πολύ διαφορετικά περιβάλλοντα, με αποτέλεσμα να υπάρχει η ανάγκη πρόσβασης σε διαφορετικού είδους πληροφορία για κάθε έναν από αυτούς. Παρόμοια είναι τα προβλήματα κατά την ολοκλήρωση (intergration) των XML δεδομένων που προέρχονται από πολλές διαφορετικές πηγές. Και σε αυτή την περίπτωση τα δεδομένα αυτά, αν και μπορεί να έχουν την ίδια φύση, είναι δυνατόν να έχουν διαφορετική μορφή ή δομή ανάλογα με το περιβάλλον της πηγής.

Σύμφωνα με τις παραπάνω ανάγκες και προβλήματα, πραγματοποιήθηκε μία ακόμα σημαντική απόπειρα επέκτασης της XML με σκοπό τη δυνατότητα αναπαράστασης της πληροφορίας με διαφορετικό τρόπο, ανάλογα με τις επικρατούσες συνθήκες κάτω από τις οποίες υφίσταται η πληροφορία αυτή. Η επέκταση αυτή ονομάστηκε *Πολυδιάστατη XML* (Multidimensional XML ή MXML) [SG02] και αποτελεί έναν τρόπο αναπαράστασης δεδομένων που παρουσιάζουν διαφορετικά χαρακτηριστικά κάτω από διαφορετικές συνθήκες. Οι συνθήκες που μπορεί να διαφοροποιούνται σύμφωνα με έναν συγκεκριμένο αριθμό παραγόντων αποτελούν το τρέχον περιβάλλον κάτω από το οποίο τα XML δεδομένα αποκτούν μία

συγκεκριμένη μορφή.

1.2 Ορισμός του προβλήματος της διατριβής

Το ζήτημα της αποθήκευσης XML δεδομένων σε σχεσιακές βάσεις δεδομένων παρουσιάζει εξίσου σημαντικό ενδιαφέρον και για τις περιπτώσεις επεκτάσεων της XML. Η Πολυδιάστατη XML, όπως προαναφέραμε, είναι μία από τις επεκτάσεις αυτές. Η MXML αποτελεί έναν τρόπο αναπαράστασης πολλαπλών XML τεκμηρίων σε ένα μονάχα κείμενο. Η κεντρική έννοια που χρησιμοποιείται για την επίτευξη του σκοπού αυτού είναι η έννοια του *"ερμηνευτικού περιβάλλοντος"* (context). Ένα *ερμηνευτικό περιβάλλον* προσδιορίζει εκείνες τις συνθήκες κάτω από τις οποίες τα διάφορα στοιχεία του MXML κειμένου έχουν νόημα και εκφράζεται με την βοήθεια σημασιολογικών προσδιοριστών, οι οποίοι ονομάζονται *"προσδιοριστές περιβάλλοντος"* (context specifiers). Αυτοί οι *"προσδιοριστές περιβάλλοντος"* χρησιμοποιούνται για να ορίσουν ένα σύνολο από *"κόσμους"* (worlds). Κάθε *"κόσμος"* αποτελεί ένα περιβάλλον μέσα στο οποίο τα δεδομένα αποκτούν ένα συγκεκριμένο νόημα και ορίζεται προσδίδοντας από μία τιμή σε ένα σύνολο από *"διαστάσεις"* (dimensions). Έτσι, ανάλογα με το *"ερμηνευτικό περιβάλλον"* που διαμορφώνεται κάθε φορά, μπορούμε να έχουμε διαφορετικές *"εκφάνσεις"* (facets) των διαφόρων XML *στοιχείων* (elements) ή *γνωρισμάτων* (attributes), άρα και ένα διαφορετικό ως προς το περιεχόμενο και την δομή του XML κείμενο.

Κατά παρόμοιο τρόπο με την αποθήκευση XML δεδομένων, το πρόβλημα το οποίο ανακύπτει αφορά την ανάπτυξη αντίστοιχων τεχνικών αποθήκευσης MXML κειμένων σε σχεσιακές βάσεις δεδομένων. Αυτή τη φορά ωστόσο απαιτείται να ληφθούν υπόψη όλα εκείνα τα επιπρόσθετα στοιχεία που εισάγει η MXML. Επιπλέον, πέραν των μεθόδων αποθήκευσης, χρειάζεται να μελετηθούν τρόποι διαχείρισης, ενημέρωσης (update) αλλά και ανάκτησης (querying) των αποθηκευμένων δεδομένων με έναν διαφανή προς τον χρήστη τρόπο.

1.3 Συνεισφορά Διατριβής

Αρχικά, η παρούσα διδακτορική διατριβή προτείνει τεχνικές αποθήκευσης των MXML δεδομένων σε σχεσιακές βάσεις δεδομένων [FGS07a, FGS, FGS12, FGS13]. Οι προτεινόμενες αυτές τεχνικές είναι εμπνευσμένες από την κατηγορία των τεχνικών αποθήκευσης XML δεδομένων ανεξάρτητα από την ύπαρξη XML σχήματος, συνεπώς για την αποθήκευση των MXML κειμένων δεν απαιτείται η ύπαρξη κάποιου αντίστοιχου σχήματος. Με βάση τις ιδιαιτερότητες της MXML, τα σχεσιακά σχήματα που χρησιμοποιούνται από τις παραπάνω τεχνικές, πρέπει να περιέχουν την κατάλληλη δομή για την αποθήκευση των επιπρόσθετων πληροφοριών που αφορούν το ερμηνευτικό περιβάλλον.

Επιπλέον, για την σχηματική αναπαράσταση των MXML δεδομένων, προτείνεται ένα γραφικό μοντέλο αναπαράστασης. Το μοντέλο αυτό ονομάζεται MXML γράφος (MXML graph) και αναπαριστά τα δεδομένα σε μία δενδροειδή δομή [FGS07a]. Με τον τρόπο αυτό, τα διάφορα δομικά στοιχεία της MXML πληροφορίας απεικονίζονται με τη μορφή κόμβων στο MXML δένδρο, ενώ οι ακμές του δένδρου αυτού προσδιορίζουν τις σχέσεις μεταξύ των κόμβων αλλά και τα περιβάλλοντα κάτω από τα οποία οι κόμβοι αυτοί υφίστανται σε συνάρτηση με τους αντίστοιχους προσδιοριστές περιβάλλοντος.

Το επόμενο στάδιο αυτού της αποθήκευσης της MXML, αφορά την υποβολή ερωτημάτων από την πλευρά των εφαρμογών του χρήστη, προς τα MXML δεδομένα που είναι αποθηκευμένα στη σχεσιακή βάση. Τα ερωτήματα που παράγονται από τις διάφορες εφαρμογές του παγκόσμιου ιστού βρίσκονται αρχικά σε XML μορφή. Στη συνέχεια, και με διαφανείς προς τον χρήστη διαδικασίες, τα XML ερωτήματα μετατρέπονται σε SQL ερωτήματα, τα οποία υποβάλλονται στη σχεσιακή βάση δεδομένων. Ακολουθεί τέλος η αντίστροφη διαδικασία, σύμφωνα με την οποία τα SQL αποτελέσματα μετατρέπονται ξανά σε XML δεδομένα και επιστρέφονται στην εφαρμογή του χρήστη. Σχετικά με την παραπάνω διαδικασία, η διδακτορική διατριβή παρουσιάζει αρχικά μία γλώσσα αναπαράστασης MXML ερωτημάτων. Η γλώσσα αυτή είναι μία προέκταση της XPath και ονομάζεται *Πολυδιάστατη XPath* (Multidimensional XPath ή MXPath) [FGS, FSG08]. Επιπλέον, προτείνεται ένας αλγόριθμος μετατροπής MXML ερωτημάτων σε SQL ερωτήματα [FGS12].

Πέρα από το πρόβλημα της υποβολής ερωτημάτων στα αποθηκευμένα MXML

δεδομένα, ανακύπτει και το ζήτημα της ενημέρωσης (update) των δεδομένων αυτών. Σχετικά με το πρόβλημα αυτό, η διδακτορική διατριβή προτείνει μεθόδους ενημέρωσης, με βάση το γραφικό μοντέλο αναπαράστασης των MXML δεδομένων. Πιο συγκεκριμένα, εξετάζονται οι λειτουργίες της διαγραφής (delete) υποδένδρου, της προσθήκης (insert) υποδένδρου, της ενημέρωσης (update) ετικέτας (label) κόμβου, της ενημέρωσης ερμηνευτικού περιβάλλοντος, της ενημέρωσης τιμής (value) και της αντικατάστασης (replace) υποδένδρου, ενώ προτείνονται και σχετικοί αλγόριθμοι υλοποίησης των λειτουργιών αυτών [FGS07b, FGS, FGS08].

Στα πλαίσια της βελτιστοποίησης των μεθόδων αποθήκευσης, η διδακτορική διατριβή διεξάγει περαιτέρω μελέτη σε ότι αφορά την αναπαράσταση και αποθήκευση του ερμηνευτικού περιβάλλοντος στις σχεσιακές βάσεις δεδομένων. Η μελέτη προς αυτήν την κατεύθυνση αποτέλεσε σημαντικό παράγοντα για την εξεύρεση λύσης στο πρόβλημα της μετατροπής MXML ερωτημάτων, τα οποία εκφράζονται με τη βοήθεια της MXPath, σε SQL ερωτήματα προς τη σχεσιακή βάση δεδομένων, μέσω του προαναφερθέντος αλγορίθμου μετατροπής. Η κατάλληλη αναπαράσταση του ερμηνευτικού περιβάλλοντος στο σχεσιακό σχήμα, είναι αυτή που επιτρέπει την μετατροπή των MXPath συνθηκών, που βασίζονται σε προσδιοριστές περιβάλλοντος, σε αντίστοιχες SQL συνθήκες [FGS11, FGS13].

Τέλος, προκειμένου να διεξαχθούν κατάλληλες δοκιμές για την διαπίστωση της ορθότητας των προτεινόμενων μεθόδων αποθήκευσης και τεχνικών μετατροπής MXML ερωτημάτων σε SQL ερωτήματα, υλοποιήθηκε σχετικό σύστημα υλοποίησης.

Κεφάλαιο 2

Πολυδιάστατη XML

2.1 Εισαγωγή

Η *Πολυδιάστατη XML* (Multidimensional XML ή MXML) [GSK01a, GSK+01b] αποτελεί μία επέκταση της συμβατικής XML, κατάλληλη για την αναπαράσταση δεδομένων που παρουσιάζουν διαφορετικές εκδοχές κάτω από διαφορετικά *ερμηνευτικά περιβάλλοντα* (contexts). Επιπλέον, η MXML επεκτείνει το συντακτικό της XML, επιτρέποντας σε *προσδιοριστές περιβάλλοντος* (context specifiers) να χαρακτηρίζουν στοιχεία και γνωρίσματα (προσδίδοντας σε αυτά διαφορετική σημασία) και να καθορίζουν τα ερμηνευτικά περιβάλλοντα (ή κόσμους) κάτω από τα οποία τα συστατικά μέρη ενός εγγράφου έχουν υπόσταση.

Στην MXML, τα ερμηνευτικά περιβάλλοντα προσδιορίζονται αποδίδοντας τιμές σε μία ή περισσότερες *διαστάσεις* (dimensions). Για το λόγο αυτό, η MXML είναι κατάλληλη για την αναπαράσταση δεδομένων, τα οποία υπό διαφορετικά ερμηνευτικά περιβάλλοντα, μπορούν να εκφραστούν ως διαφορετικές *εκφάνσεις* (facets), οι οποίες με τη σειρά τους διαφοροποιούνται τόσο ως προς τις τιμές όσο και ως προς τη δομή των δεδομένων αυτών. Εάν εναλλακτικά δημιουργούσαμε ένα διαφορετικό XML κείμενο για κάθε πιθανό συνδυασμό των τιμών των παραπάνω διαστάσεων, θα έπρεπε τα ίδια δεδομένα να επαναλαμβάνονται πολλές φορές μέσα στα κείμενα αυτά.

Ακόμα, σε ότι αφορά την γραφική αναπαράσταση των MXML κειμένων, αυτή γίνεται με την χρήση δένδρων τα οποία αποκαλούνται *MXML δένδρα* (MXML trees) και αποτελούνται από κατάλληλου είδους κόμβους και ακμές.

2.2 Η σύνταξη της Πολυδιάστατης XML

Όπως αναφέραμε και παραπάνω, στην MXML, τα δεδομένα αποδίδονται ως διαφορετικές εκφάνσεις, έχοντας διαφορετικές τιμές ή δομή κάτω από διαφορετικά *ερμηνευτικά περιβάλλοντα* (contexts), σε συνάρτηση με έναν αριθμό από *διαστάσεις* (dimensions), οι οποίες μπορούν να επηρεάσουν τόσο τα στοιχεία όσο και τα γνωρίσματα των δεδομένων αυτών [GSK01a, GSK+01b]. Για την MXML η έννοια του *κόσμου* (world) είναι θεμελιώδης. Ένας κόσμος αποτελεί το περιβάλλον μέσα στο οποίο τα δεδομένα αποκτούν συγκεκριμένο νόημα, ενώ προσδιορίζεται αποδίδοντας σε κάθε διάσταση κάποια τιμή από το σύνολο τιμών της εκάστοτε διάστασης. Επιπλέον, στην MXML χρησιμοποιούνται σημασιολογικοί προσδιοριστές, οι οποίοι ονομάζονται *προσδιοριστές περιβάλλοντος* (context specifiers) και επιτρέπουν τον προσδιορισμό συνόλων από κόσμους θέτοντας τις κατάλληλες περιοριστικές συνθήκες σχετικά με τις τιμές τις οποίες μπορούν να πάρουν οι αντίστοιχες διαστάσεις. Τα στοιχεία ή τα γνωρίσματα που αποδίδονται ως διαφορετικές εκφάνσεις υπό διαφορετικά ερμηνευτικά περιβάλλοντα ονομάζονται *πολυδιάστατα στοιχεία/γνωρίσματα* (multidimensional elements/attributes). Κάθε πολυδιάστατο στοιχείο ή γνώρισμα περιλαμβάνει μία ή περισσότερες εκφάνσεις οι οποίες αποκαλούνται *στοιχεία/γνωρίσματα περιβάλλοντος* (context elements/attributes) και συνοδεύονται από τον αντίστοιχο προσδιοριστή περιβάλλοντος που προσδιορίζει το σύνολο των κόσμων κάτω από το οποίο η κάθε έκφανση αποτελεί την ισχύουσα έκφανση του στοιχείου/γνωρίσματος αυτού. Η σύνταξη των πολυδιάστατων στοιχείων σε ένα MXML κείμενο παρουσιάζεται παρακάτω:

```
<@element_name  attribute_specification>
  [context_specifier_1]
    <element_name attribute_specification_1>
       element_content_1
    </element_name>
  [/]
  . . .
  [context_specifier_N]
    <element_name attribute_specification_N>
       element_content_N
    </element_name>
  [/]
</@element_name>
```

Για την δήλωση ενός πολυδιάστατου γνωρίσματος, η ακόλουθη σύνταξη χρησιμοποιείται:

```
attribute_name =
   [context_specifier_1] attribute_value_1 [/] ...
   [context_specifier_n] attribute_value_n [/]
```

Με βάση τα παραπάνω, παρουσιάζονται δύο σχετικά παραδείγματα που απεικονίζουν την σύνταξη των MXML κειμένων.

Παράδειγμα 2.1 *Το παράδειγμα MXML κειμένου που ακολουθεί αναπαριστά ένα συγκεκριμένο μοντέλο αυτοκινήτου, του οποίου τα χαρακτηριστικά ποικίλουν ανάλογα με την αγορά στην οποία απευθύνεται. Δύο διαστάσεις χρησιμοποιούνται σε αυτό το MXML κείμενο. Η διάσταση* `factory` *η οποία παίρνει τιμές από το σύνολο τιμών* {`Japan, Italy`}*, και η διάσταση* `market` *η οποία παίρνει τιμές από το σύνολο τιμών* {`USA, Europe`}.

```
<car type=[factory=Japan]"sport"[/]
          [factory=Italy]"family"[/]>
  <@designer>
     [factory=Japan]
        <designer>groupo Bertone</designer>
     [/]
     [factory=Italy,market=Europe]
        <designer>Pedro Seelig</designer>
     [/]
     [factory=Italy,market=USA]
        <designer>Rollo Dixon</designer>
     [/]
  </@designer>
  <@engine>
     [factory=Japan]
        <engine>
           <capacity>1.8lt</capacity>
           <@power>
             [market=Europe]
                <power>180hp</power>
             [/]
             [market=USA]
                <power>200hp</power>
```

```
            [/]
         </@power>
      </engine>
   [/]
   [factory=Italy]
      <engine>
         <capacity>1.6lt</capacity>
         <@power>
           [market=Europe]
              <power>120hp</power>
           [/]
           [market=USA]
              <power>140hp</power>
           [/]
         </@power>
      </engine>
   [/]
</@engine>
<@performance>
   [factory=Japan]
      <performance>
         <top_speed>250km/h</top_speed>
         <@acceleration>
           [market=Europe]
              <acceleration>0-100 in 6sec
                 </acceleration>
           [/]
           [market=USA]
              <acceleration>0-100 in 5sec
                 </acceleration>
           [/]
         </@acceleration>
      </performance>
   [/]
   [factory=Italy]
      <performance>
         <acceleration>0-100 in 5sec
            <acceleration/>
         <@top_speed>
           [market=Europe]
              <top_speed>200km/h</top_speed>
```

```
            [/]
            [market=USA]
               <top_speed>210km/h</top_speed>
            [/]
          </@top_speed>
        </performance>
     [/]
  </@performance>
</car>
```

Παρατηρήστε τα πολυδιάστατα στοιχεία (για παράδειγμα το στοιχείο `price`*) των οποίων το όνομα αναπαριστάται με το σύμβολο* `@` *την στιγμή που τα αντίστοιχα στοιχεία περιβάλλοντος έχουν το ίδιο όνομα στοιχείου χωρίς όμως το σύμβολο* `@`*.*

Παράδειγμα 2.2 *Το MXML κείμενο που παρουσιάζεται παρακάτω αναπαριστά ένα βιβλίο κάποιου βιβλιοπωλείου. Δύο διαστάσεις χρησιμοποιούνται σε αυτή την περίπτωση. Η διάσταση* `edition` *η οποία παίρνει τιμές από το σύνολο τιμών* {`greek, english`}*, και η διάσταση* `customer_type` *η οποία παίρνει τιμές από το σύνολο τιμών* {`student, library, teacher`}*.*

```
<book isbn=[edition=english]"0-13-110362-8"[/]
           [edition=greek]"0-13-110370-9"[/]>
  <title>The C programming language</title>
  <authors>
     <author>Brian W. Kernighan</author>
     <author>Dennis M. Ritchie</author>
  </authors>
  <@publisher>
     [edition = english] <publisher>Prentice Hall</publisher>[/]
     [edition = greek] <publisher>Klidarithmos</publisher>[/]
  </@publisher>
  <@translator>
     [edition = greek] <translator>Thomas Moraitis</translator>[/]
  </@translator>
  <@price>
     [edition=english]<price>15</price>[/]
    [edition=greek,customer_type in {student, teacher}]<price>9</price>[/]
     [edition=greek,customer_type=library]<price>12</price>[/]
  </@price>
  <@cover>
```

```
    [edition=english]<cover><material>leather</material></cover>[/]
    [edition=greek]
       <cover>
          <material>paper</material >
          <@picture>
            [customer_type=student]<picture>student.bmp</picture>[/]
            [customer_type=library]<picture>library.bmp</picture>[/]
          </@picture>
        </cover>
     [/]
  </@cover>
</book>
```

Ένα MXML κείμενο μπορεί να θεωρηθεί σαν μία συμπαγής αναπαράσταση ενός συνόλου από συμβατικά XML κείμενα, κάθε ένα εκ των οποίων υφίσταται υπό έναν συγκεκριμένο κόσμο. Η διαδικασία εξαγωγής XML κειμένων από ένα MXML κείμενο κάτω από την ισχύ συγκεκριμένων κόσμων ονομάζεται *αναγωγή* (reduction) [GSK01a].

2.3 Χαρακτηριστικά των MXML κειμένων

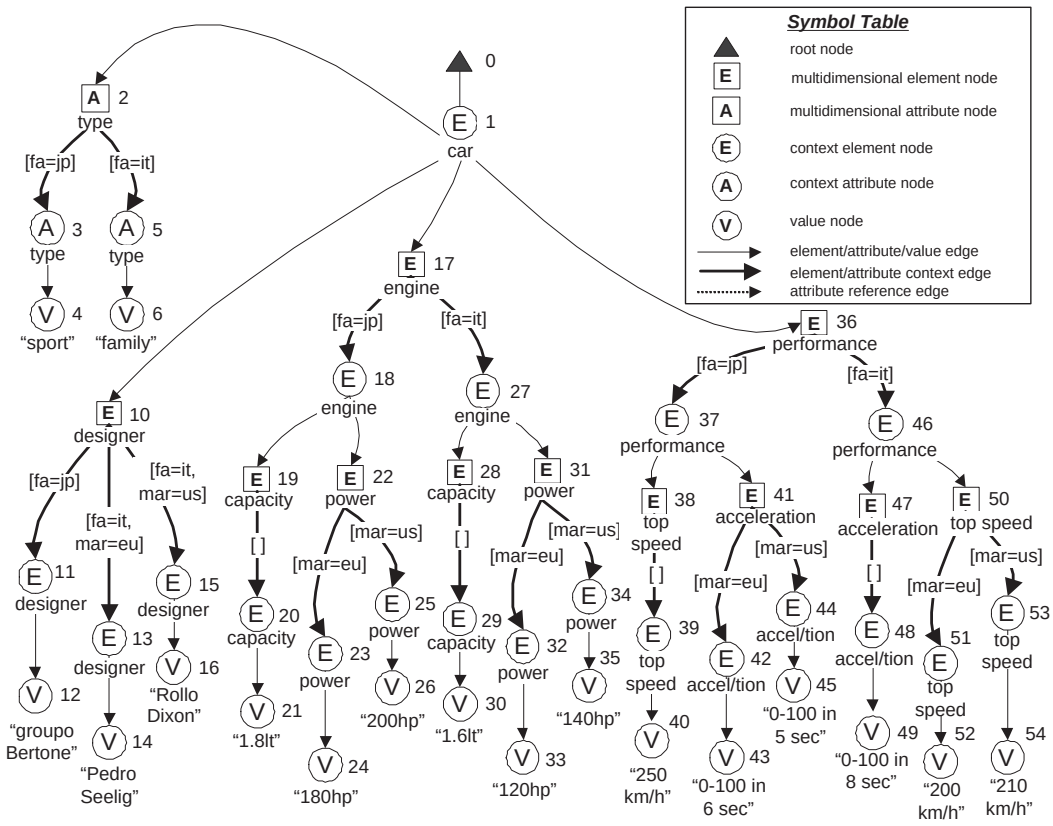
2.3.1 Γραφικό μοντέλο αναπαράστασης της MXML

Σε αυτή την ενότητα παρουσιάζεται ένα γραφικό μοντέλο αναπαράστασης για την MXML το οποίο ονομάζεται *MXML δένδρο* (MXML tree). Το προτεινόμενο μοντέλο περιλαμβάνει ένα δένδρο από κόμβους, ενώ κάθε κόμβος του δένδρου αυτού προσδιορίζεται από έναν μοναδικό αριθμό (node id). Στο MXML δένδρο εκτός από τον κόμβο που αποτελεί την ρίζα του δένδρου (root node) υπάρχουν και τα παρακάτω είδη κόμβων: *πολυδιάστατοι κόμβοι στοιχείων* (multidimensional element nodes), *κόμβοι στοιχείων περιβάλλοντος* (context element nodes), *πολυδιάστατοι κόμβοι γνωρισμάτων* (multidimensional attribute nodes), *κόμβοι γνωρισμάτων περιβάλλοντος* (context attribute nodes) και *κόμβοι τιμών* (value nodes). Οι κόμβοι στοιχείων περιβάλλοντος, οι κόμβοι γνωρισμάτων περιβάλλοντος και οι κόμβοι τιμών αντιστοιχούν στους κόμβους στοιχείων, γνωρισμάτων και τιμών ενός συμβατικού XML γράφου. Κάθε πολυδιάστατος κόμβος στοιχείου ή κόμβος στοιχείου περιβάλλοντος χαρακτηρίζεται από το αντίστοιχο όνομα στοιχείου,

ενώ κάθε πολυδιάστατος κόμβος γνωρίσματος ή κόμβος γνωρίσματος περιβάλλοντος από το αντίστοιχο όνομα γνωρίσματος. Όπως ισχύει και στη συμβατική XML, οι κόμβοι τιμών είναι *κόμβοι φύλλα* (leaf nodes) και περιέχουν τις σχετικές τιμές των στοιχείων/γνωρισμάτων. Οι διάφορες εκφάνσεις (κόμβοι στοιχείων/γνωρισμάτων περιβάλλοντος) ενός πολυδιάστατου κόμβου είναι συνδεδεμένες με τον κόμβο αυτόν με ακμές που προσδιορίζονται με βάση τους σχετικούς προσδιοριστές περιβάλλοντος, οι οποίοι υποδηλώνουν τις συνθήκες κάτω από τις οποίες η κάθε έκφανση βρίσκεται σε ισχύ. Οι ακμές αυτές ονομάζονται *ακμές στοιχείων/γνωρισμάτων περιβάλλοντος* (element/attribute context edges) αντίστοιχα. Τα στοιχεία/γνωρίσματα περιβάλλοντος συνδέοντα με τους κόμβους στοιχείων/γνωρισμάτων ή τιμών που αποτελούν παιδιά τους, μέσω ακμών που ονομάζονται *ακμές στοιχείων/γνωρισμάτων* ή *τιμών* (element/attribute/value edges) αντίστοιχα. Τέλος, τα γνωρίσματα περιβάλλοντος τύπου IDREF(S) συνδέονται με τους κόμβους στοιχείων στους οποίους οδηγούν μέσω ακμών που ονομάζονται *ακμές αναφοράς γνωρισμάτων* (attribute reference edges).

Παράδειγμα 2.3 *Στο Σχήμα 2.1, μπορούμε να δούμε την αναπαράσταση του MXML κειμένου του Παραδείγματος 2.1 σαν MXML δένδρο. Σημειώνουμε ότι στην περίπτωση που υπάρχουν κόμβοι στοιχείων/γνωρισμάτων περιβάλλοντος εκπεφρασμένοι μόνο με μία έκφανση στο MXML κείμενο, προστίθενται επιπρόσθετοι πολυδιάστατοι κόμβοι με το ίδιο όνομα (πχ. κόμβοι 7 και 10) σαν πατρικοί κόμβοι αυτών των κόμβων περιβάλλοντος για να εξασφαλιστεί ότι οι τύποι των ακμών εναλλάσσονται με συνέπεια σε κάθε διαδρομή του δένδρου. Αυτό δεν επηρεάζει τις πληροφορίες που περιέχει το MXML κείμενο, αλλά διευκολύνει τη διάσχιση του δένδρου και τον σχηματισμό σχετικών ερωτημάτων. Για οικονομία χώρου, στο Σχήμα 2.1 χρησιμοποιούνται προφανείς συντομεύσεις για τα ονόματα των διαστάσεων και των τιμών που εμφανίζονται στο MXML κείμενο.*

Παράδειγμα 2.4 *Στο Σχήμα 2.2, βλέπουμε αντίστοιχα την γραφική αναπαράσταση και του MXML κειμένου του Παραδείγματος 2.2.*

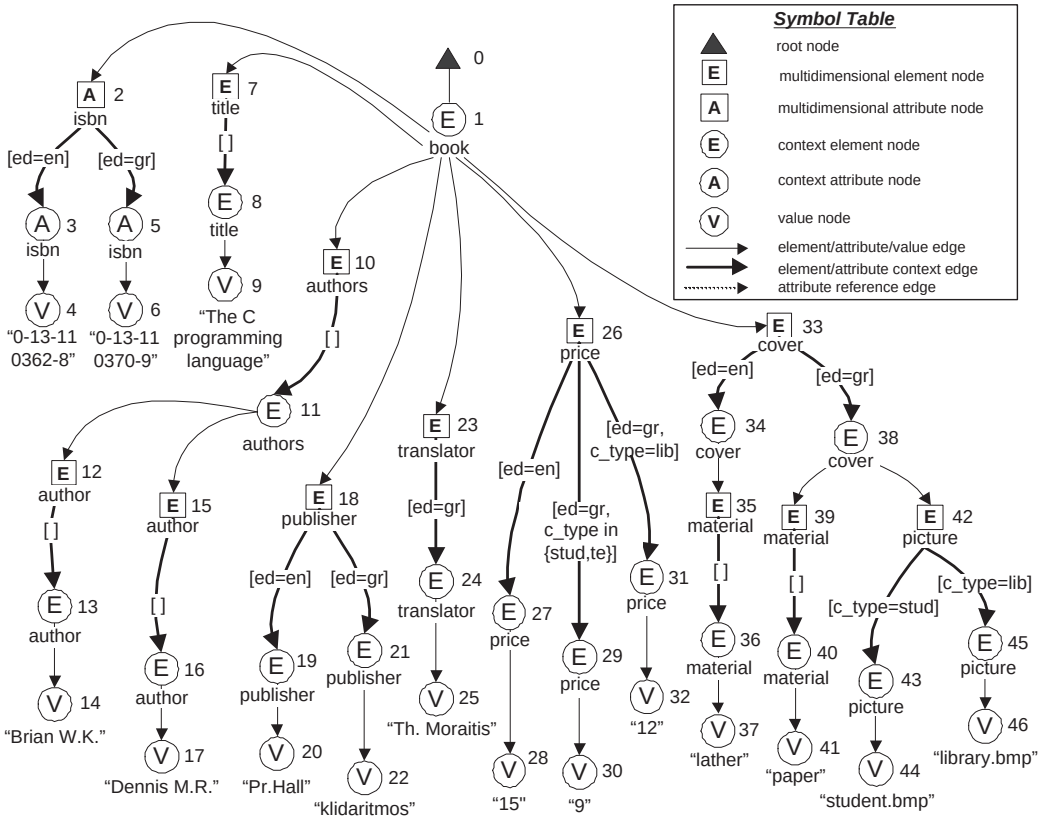


Σχήμα 2.1: Γραφική αναπαράσταση της MXML (MXML tree)

2.3.2 Οι Προσδιοριστές Περιβάλλοντος και τα χαρακτηριστικά τους στην MXML

Η κεντρική έννοια σχετικά με την MXML είναι η έννοια του *κόσμου* (world) και ορίζεται όπως φαίνεται παρακάτω:

Ορισμός 2.3.1. Έστω $\mathcal{S}$ ένα σύνολο από *ονόματα διαστάσεων* και για κάθε $d \in \mathcal{S}$, έστω $\mathcal{D}_d$, με $\mathcal{D}_d \neq \emptyset$, να είναι το *σύνολο τιμών* του d. Ένας *κόσμος* w είναι ένα σύνολο ζευγαριών (d, u), όπου $d \in \mathcal{S}$ και $u \in \mathcal{D}_d$ έτσι ώστε για κάθε όνομα διάστασης στο $\mathcal{S}$ υπάρχει μονάχα ένα στοιχείο στον κόσμο w. Το σύνολο όλων των πιθανών κόσμων συμβολίζεται με $\mathcal{U}$.



Σχήμα 2.2: Γραφική αναπαράσταση της MXML (MXML tree)

Στην MXML, οι προσδιοριστές περιβάλλοντος που χαρακτηρίζουν τις ακμές περιβάλλοντος, αποδίδουν το *ρητό περιβάλλον* (explicit context ή ec) των κόμβων στους οποίους οι ακμές αυτές οδηγούν. Το ρητό περιβάλλον όλων των άλλων κόμβων είναι εξ ορισμού το *καθολικό περιβάλλον* (universal context) το οποίο συμβολίζεται με $[\,]$ και αναπαριστά το σύνολο όλων των πιθανών κόσμων $\mathcal{U}$. Ένα ρητό περιβάλλον μπορεί να θεωρηθεί ως το πραγματικό ερμηνευτικό περιβάλλον μέσα στα όρια ενός μονάχα πολυδιάστατου στοιχείου/γνωρίσματος. Όταν τα στοιχεία και τα γνωρίσματα συνδυάζονται προκειμένου να σχηματίσουν ένα MXML κείμενο, τα ρητά τους περιβάλλοντα δεν προσδιορίζουν μονάχα τους κόσμους υπό τους οποίους τα στοιχεία/γνωρίσματα αυτά έχουν ισχύ, αφού όταν ένα στοιχείο/γνώρισμα e_2 είναι τμήμα ενός άλλου στοιχείου/γνωρίσματος e_1, τότε το e_2 έχει νόημα μόνο κάτω από τους κόσμους υπό τους οποίους και το e_1 έχει νόημα.

Αυτό εκλαμβάνεται ως σαν το ερμηνευτικό περιβάλλον του e_1 να κληρονομείται στο e_2. Ένα ερμηνευτικό περιβάλλον που μεταδίδεται με αυτόν τον τρόπο συνδυάζεται με το ρητό περιβάλλον ενός κόμβου για να παράγει το *κληρονομούμενο περιβάλλον* (inherited context ή ic) του κόμβου αυτού. Τυπικά, το κληρονομούμενο περιβάλλον $ic(q)$ ενός κόμβου q ορίζεται ως $ic(q) = ic(p) \cap^c ec(q)$, όπου $ic(p)$ είναι το κληρονομούμενο περιβάλλον του πατρικού του κόμβου p και $\cap^c$ είναι η πράξη της *τομής ερμηνευτικού περιβάλλοντος* (context intersection) που ορίζεται στο [SG02], η οποία συνδυάζοντας δύο προσδιοριστές περιβάλλοντος υπολογίζει έναν νέο που αναπαριστά την τομή (intersection) των κόσμων που ορίζονται από τους αρχικούς προσδιοριστές. Ο υπολογισμός του κληρονομούμενου περιβάλλοντος ξεκινά από την ρίζα του MXML δένδρου. Εξ ορισμού, το κληρονομούμενο περιβάλλον της ρίζας είναι το καθολικό περιβάλλον $[\,]$. Επιπλέον, τα ερμηνευτικά περιβάλλοντα δεν κληρονομούνται μέσω attribute reference ακμών, ενώ όπως και στην συμβατική XML, οι κόμβοι φύλλα του MXML δένδρου πρέπει να είναι κόμβοι τιμών (value nodes).

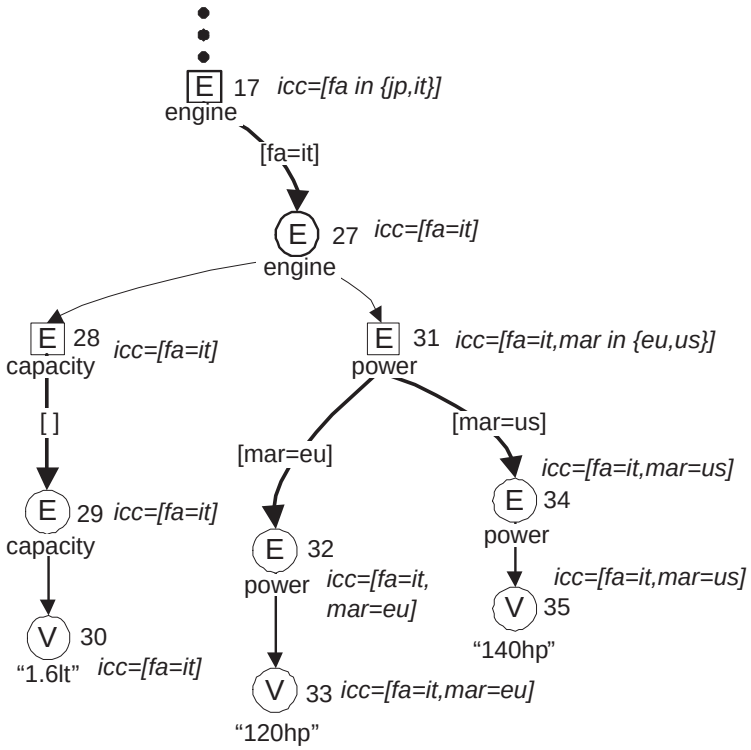
Η *κληρονομούμενη κάλυψη περιβάλλοντος* (inherited context coverage ή icc) ενός κόμβου, θέτει περαιτέρω περιορισμούς στο κληρονομούμενο περιβάλλον του κόμβου αυτού, έτσι ώστε να περιλαμβάνει μόνο τους κόσμους υπό τους οποίους ο κόμβος αυτός μπορεί να έχει πρόσβαση σε κάποιον κόμβο τιμής. Η κληρονομούμενη κάλυψη περιβάλλοντος αποδίδει το πραγματικό ερμηνευτικό περιβάλλον ενός κόμβου στα πλαίσια ενός κειμένου, λαμβάνοντας υπόψη τα ερμηνευτικά περιβάλλοντα τόσο των πατρικών όσο και των κόμβων παιδιών. Το $icc(n)$ ενός κόμβου n ορίζεται ως εξής:

Εάν n είναι ένας κόμβος τιμής τότε $icc(n) = ic(n)$; αλλιώς εάν n είναι κόμβος φύλλο αλλά όχι κόμβος τιμής τότε $icc(n) = [-]$; αλλιώς $icc(n) = icc(n_1) \cup^c icc(n_2) \cup^c ... \cup^c icc(n_k)$, όπου $n_1, \ldots, n_k$ είναι οι κόμβοι παιδιά του κόμβου n.

Το $[-]$ έχει την έννοια του *κενού προσδιοριστή περιβάλλοντος* (empty context specifier) που αντιπροσωπεύει το καινό σύνολο κόσμων. Το σύμβολο $\cup^c$ αναπαριστά την πράξη της *ένωσης ερμηνευτικού περιβάλλοντος* (context union) που ορίζεται στο [SG02]. Με βάση την πράξη αυτή δύο προσδιοριστές περιβάλλοντος συνδιάζονται και υπολογίζεται ένας νέος, ο οποίος αντιπροσωπεύει την ένωση (union) των κόσμων που προσδιορίζονται από τους αρχικούς προσδιοριστές.

Παράδειγμα 2.5 *Το Σχήμα 2.3 απεικονίζει ένα τμήμα του MXML δένδρου του σχήματος 2.1 στο οποίο έχουν σημειωθεί οι κληρονομούμενες καλύψεις περιβάλλοντος*

(icc) του κάθε κόμβου.



Σχήμα 2.3: Representing the Inherited Context Coverage

2.4 Συμπεράσματα - Παρατηρήσεις

Μέσα στο περιβάλλον του διαδικτύου και εξαιτίας της μεγάλης αύξησης της χωρητικότητας των δικτυακών επικοινωνιών ένας τεράστιος όγκος δεδομένων διακινείται καθημερινά την ίδια στιγμή που τα δεδομένα αυτά μεταβάλλονται διαρκώς τόσο ως προς τη δομή, τη σημαντικότητα αλλά και τη μορφή τους. Το πρότυπο της XML, αποτελεί σε σχέση με τις σχεσιακές βάσεις δεδομένων, ένα ιεραρχικό πρότυπο, το οποίο εξαιτίας της ευελιξία του, έχει επικρατήσει σε ότι αφορά την αναπαράσταση των πληροφοριών που διαχειρίζονται πολλές από τις εφαρμογές του παγκόσμιου ιστού. Ωστόσο, το πρότυπο αυτό αδυνατεί να απεικονίσει με έναν

ενιαίο τρόπο τις διαφορετικές εκδόσεις των απεικονιζόμενων πληροφοριών, κάτω από μεταβαλλόμενες συνθήκες.

Η MXML, ως μία επέκταση της συμβατικής XML, έχει σαν σκοπό να επιτύχει την ταυτόχρονη απεικόνιση πολλών διαφορετικών εκφάνσεων ή στιγμιοτύπων των δεδομένων που αναπαριστά ένα απλό XML κείμενο, με έναν ολοκληρωμένο (ολοκλήρωση δεδομένων) και συμπαγή τρόπο. Εισάγοντας σαν παράμετρο διαμόρφωσης ενός περιβάλλοντος, κάτω από το οποίο τα XML δεδομένα μεταβάλλονται τόσο ως προς τη δομή τους, όσο και ως προς το περιεχόμενό τους, την έννοια του ερμηνευτικού περιβάλλοντος, η MXML επιτυγχάνει την αποφυγή της επανάληψης των δεδομένων που παρατηρείται σε ένα σύνολο από συναφή XML κείμενα, ενώ παρέχει, σε αντίθεση με τη συμβατική XML, τη δυνατότητα ορισμού διαφορετικών συνθηκών κάτω από τις οποίες τα δεδομένα αποκτούν διαφορετικό κάθε φορά νόημα.

Κεφάλαιο 3

Αποθήκευση της Πολυδιάστατης XML σε σχεσιακές βάσεις δεδομένων

Σε αυτό το κεφάλαιο παρουσιάζονται τεχνικές που μπορούν να χρησιμοποιηθούν για την αποθήκευση MXML δεδομένων σε σχεσιακές βάσεις δεδομένων (MXML Storage). Οι τεχνικές αυτές βασίζονται σε μεθόδους που έχουν χρησιμοποιηθεί στο παρελθόν για την αποθήκευση των συμβατικών XML κειμένων σε σχεσιακά σχήματα, περιλαμβάνουν ωστόσο τις κατάλληλες επεκτάσεις και βελτιώσεις προκειμένου να καλύψουν τις ανάγκες της MXML.

3.1 Βασική προσέγγιση

Η βασική προσέγγιση αποθήκευσης MXML δεδομένων σε σχεσιακές βάσεις δεδομένων αποτελεί μία προφανή λύση στο πρόβλημα του MXML Storage. Σύμφωνα με την προσέγγιση αυτή, οι MXML κόμβοι αποθηκεύονται στο σύνολό τους χωρίς να προηγηθεί κάποια ομαδοποίηση.

3.1.1 Σχεσιακό σχήμα αναπαράστασης των MXML κόμβων

Η *βασική προσέγγιση* (naive approach) αποθήκευσης MXML δεδομένων σε σχεσιακές βάσεις δεδομένων χρησιμοποιεί έναν μονάχα πίνακα για την αποθή-

κευση όλων των δεδομένων[1] που περιέχονται σε ένα MXML κείμενο. Ο πίνακας αυτός, που αποκαλείται *πίνακας κόμβων* (Node Table), περιέχει όλες τις απαραίτητες πληροφορίες για την κατασκευή του MXML δένδρου, ενώ κάθε γραμμή του πίνακα αναπαριστά έναν MXML κόμβο. Τα πεδία που χαρακτηρίζουν τον πίνακα κόμβων έχουν ως εξής: `node_id` αποθηκεύει έναν μοναδικό κωδικό αριθμό που ταυτοποιεί τον κάθε κόμβο, `parent_id` αποθηκεύει το μοναδικό κωδικό αριθμό που ταυτοποιεί τον πατρικό κόμβο, `ordinal` αποθηκεύει έναν αριθμό ο οποίος υποδηλώνει την διάταξη των κόμβων μεταξύ των αδερφικών τους κόμβων (siblings), `tag` αποθηκεύει την ετικέτα (tag) του κόμβου ή το κενό (συμβολιζόμενο ως "-") εάν είναι κόμβος τιμής (value node), `value` αποθηκεύει την τιμή του κόμβου εάν πρόκειται για κόμβο τιμής ή το κενό (NULL) σε οποιαδήποτε άλλη περίπτωση, `type` αποθηκεύει έναν κωδικό που υποδηλώνει το είδος του κόμβου (CE για στοιχείο περιβάλλοντος, CA για γνώρισμα περιβάλλοντος, ME για πολυδιάστατο στοιχείο, MA για πολυδιάστατο γνώρισμα και VN για κόμβο τιμής) και `explicit_context` για την αποθήκευση του ρητού περιβάλλοντος του κόμβου (σαν αλφαριθμητικό). Να σημειωθεί ότι στην περίπτωση αυτή το ρητό περιβάλλον διατηρείται για λόγους πληρότητας και δεν εξυπηρετεί σκοπούς ανάκτησης των αποθηκευμένων δεδομένων. Στην συνέχεια του κεφαλαίου αυτού θα δούμε πως γίνεται στο σχεσιακό σχήμα η κωδικοποίηση της αντιστοίχισης των κόμβων με τους κόσμους υπό τους οποίους οι κόμβοι αυτοί βρίσκονται σε ισχύ.

Παράδειγμα 3.1 *Το Σχήμα. 3.1 δείχνει πως το MXML δένδρο του Σχήματος 2.2 αποθηκεύεται στον πίνακα κόμβων. Μερικοί από τους κόμβους έχουν παραληφθεί και αντικατασταθεί από αποσιωπητικά "....", χάριν συντομίας.*

3.1.2 Σχεσιακό σχήμα αναπαράστασης του ερμηνευτικού περιβάλλοντος

Στα πλαίσια της βασικής προσέγγισης, για την αποθήκευση της πληροφορίας που αφορά το ερμηνευτικό περιβάλλον, χρησιμοποιούνται τρεις επιπρόσθετοι πίνακες στο σχεσιακό σχήμα, όπως φαίνεται και στο Σχήμα 3.2. Ο πρώτος πίνακας ονομάζεται *πίνακας πιθανών κόσμων* (Possible Worlds Table) και αντιστοιχίζει

[1] Ο σχετικός τρόπος αναπαράστασης της πληροφορίας που αφορά το ερμηνευτικό περιβάλλον για τα MXML δεδομένα παρουσιάζεται παρακάτω.

Node Table						
node_id	parent_id	ordinal	tag	value	type	explicit_context
1	0	1	book	-	CE	-
2	1	1	isbn	-	MA	-
3	2	1	isbn	-	CA	[ed=en]
4	3	1	-	0-13-110362-8	VN	-
5	2	2	isbn	-	CA	[ed=gr]
6	5	1	-	0-13-110370-9	VN	-
7	1	2	title	-	ME	-
8	7	1	title	-	CE	[ ]
9	8	1	-	The C progr. lang.	VN	-
....	....	....	....	....	....	....
43	42	1	picture	-	CE	[c_type=stud]
....	....	....	....	....	....	....

Σχήμα 3.1: Αποθήκευση του MXML δένδρου του Σχήματος 2.2 σε έναν πίνακα κόμβων (Node Table).

έναν μοναδικό αριθμό (πεδίο `word_id`) σε κάθε πιθανό συνδυασμό των τιμών των διαστάσεων. Κάθε διάσταση του MXML κειμένου καταλαμβάνει ένα ξεχωριστό πεδίο στο σχήμα του πίνακα αυτού. Ο δεύτερος πίνακας ονομάζεται *πίνακας ρητού περιβάλλοντος*(Explicit Context Table) και διατηρεί την αντιστοιχία του κάθε κόμβου με τους κόσμους που εκφράζει το ρητό περιβάλλον του κόμβου αυτού. Τέλος, ο τρίτος πίνακας ονομάζεται *πίνακας κληρονομούμενης κάλυψης* (Inherited Coverage Table) και κατά όμοιο τρόπο διατηρεί την αντιστοιχία του κάθε κόμβου με τους κόσμους που εκφράζει την κληρονομούμενη κάλυψη περιβάλλοντος του κόμβου αυτού.

Παράδειγμα 3.2 *Το Σχήμα. 3.2, παρουσιάζει τμήματα των πινάκων Possible Worlds Table, Explicit Context Table, και Inherited Coverage Table που προκύπτουν κωδικοποιώντας και αποθηκεύοντας την σχετική με το ερμηνευτικό περιβάλλον πληροφορία, η οποία εξάγεται από το MXML δένδρο του Σχήματος 2.2.*

Για παράδειγμα, η κληρονομούμενη κάλυψη περιβάλλοντος του κόμβου με `node_id=6` *περιλαμβάνει τους κόσμους:*

Possible Worlds Table		
world_id	edition	customer_type
1	gr	stud
2	gr	lib
3	gr	te
4	en	stud
5	en	lib
6	en	te

Explicit Context Table	
node_id	world_id
1	1
1	2
1	3
1	4
1	5
1	6
....	....
5	1
5	2
5	3
6	1
6	2
6	3
6	4
6	5
6	6
....	....

Inherited Coverage Table	
node_id	world_id
1	1
1	2
1	3
1	4
1	5
1	6
....	....
5	1
5	2
5	3
6	1
6	2
6	3
....	....

Σχήμα 3.2: Αντιστοίχηση MXML κόμβων σε κόσμους

$w_1 = \{$`(edition, greek)`, `(customer_type, student)`$\}$,
$w_2 = \{$`(edition, greek)`, `(customer_type, library)`$\}$ *και*
$w_3 = \{$`(edition, greek)`, `(customer_type, teacher)`$\}$

Αυτό κωδικοποιείται στον πίνακα κληρονομούμενης κάλυψης σαν τρεις γραμμές με `node_id=6` *και τα world ids 1, 2 και 3. Αντίστοιχα, στον πίνακα ρητού περιβάλλοντος ο ίδιος κόμβος αντιστοιχεί σε όλους τους πιθανούς κόσμους (ids 1, 2, 3, 4, 5 και 6).*

3.2 Προσέγγιση βάσει είδους

Μία βελτιωμένη προσέγγιση σε ότι αφορά την αποθήκευση MXML δεδομένων σε σχεσιακές βάσεις δεδομένων αποτελεί η προσέγγιση που βασίζεται στο είδος των MXML κόμβων και ονομάζεται *προσέγγιση βάσει είδους* (Type Approach). Σύμφωνα με την προσέγγιση αυτή, οι MXML κόμβοι διαχωρίζονται σε ομάδες ανάλογα με το είδος τους, ενώ η κάθε ομάδα κόμβων αποθηκεύεται σε διαφορετικό πίνακα στη σχεσιακή βάση δεδομένων. Οι πίνακες αυτοί παίρνουν το όνομά τους από το είδος των κόμβων τους οποίους αποθηκεύουν. Έτσι ο *πίνακας πολυδιάστατων στοιχείων*(Multidimensional Element ή ME Table) αποθηκεύει πολυδιάστατους κόμβους στοιχείων, ο *πίνακας στοιχείων περιβάλλοντος* (Context Element ή CE Table) αποθηκεύει κόμβους στοιχείων περιβάλλοντος, ο *πίνακας πολυδιάστατων γνωρισμάτων*(Multidimensional Attribute ή MA Table) αποθηκεύει πολυδιάστατους κόμβους γνωρισμάτων, ο *πίνακας γνωρισμάτων περιβάλλοντος* (Context Attribute ή CA Table) αποθηκεύει κόμβους γνωρισμάτων περιβάλλοντος και ο *πίνακας τιμών* (Value Table) αποθηκεύει κόμβους τιμών.

Στα παρακάτω σχήματα, μπορούμε να δούμε τους πίνακες που περιλαμβάνει η προσέγγιση βάσει του είδους των MXML κόμβων του MXML δένδρου του Σχήματος 2.2. Πιο συγκεκριμένα, το Σχήμα 3.3 παρουσιάζει τους πίνακες που περιέχουν τους πολυδιάστατους κόμβους, το Σχήμα 3.4 τους πίνακες που περιέχουν τους κόμβους περιβάλλοντος και το Σχήμα 3.5 τον πίνακα με τους κόμβους τιμών.

ME Table			
node_id	parent_id	ordinal	tag
7	1	2	title
10	1	3	authors
....	....	....	....

MA Table			
node_id	parent_id	ordinal	tag
2	1	1	isbn

Σχήμα 3.3: Προσέγγιση βάσει είδους: Οι πίνακες των πολυδιάστατων κόμβων

Κάθε γραμμή των εν λόγο πινάκων αντιστοιχεί σε έναν κόμβο του MXML δένδρου. Τα γνωρίσματα στους πίνακες αυτούς έχουν την ίδια σημασία όπως τα

CE Table				
node_id	parent_id	ordinal	tag	explicit_ context
1	0	1	book	-
8	7	1	title	[ ]
....	....	....	....	....
19	18	1	publisher	[ed=en]
21	18	2	publisher	[ed=gr]
....	....	....	....	....

CA Table				
node_d	parent_id	ordinal	tag	explicit_context
3	2	1	isbn	[ed=en]
5	2	2	isbn	[ed=gr]

Σχήμα 3.4: Προσέγγιση βάσει είδους: Οι πίνακες των κόμβων περιβάλλοντος

Value Table		
node_id	parent_id	value
4	3	0-13-110362-8
6	5	0-13-110362-9
9	8	The C programming language
....	....	....

Σχήμα 3.5: Προσέγγιση βάσει είδους: Ο πίνακας των κόμβων τιμών

αντίστοιχα γνωρίσματα στον πίνακα κόμβων. Χρησιμοποιώντας όμως αυτή την προσέγγιση αποφεύγουμε μερικά από τα προβλήματα της βασικής προσέγγισης. Πιο συγκεκριμένα, εξαλείφονται *κενές τιμές* (NULL values) και μη σχετικά γνωρίσματα, ενώ την ίδια στιγμή μειώνουμε το μέγεθος των πινάκων που εμπλέκονται σε joins κατά την πλοήγηση στο MXML δένδρο και την υποβολή σχετικών ερωτημάτων (querying).

3.3 Προσέγγιση βάσει μονοπατιών

Σε αυτή την ενότητα παρουσιάζεται μία τεχνική σύμφωνα με την οποία η αποθήκευση ενός MXML κειμένου σε μία σχεσιακή βάση γίνεται με βάση τα μονοπάτια (paths) του MXML δένδρου. Η τεχνική αυτή αποτελεί τη λεγόμενη *προσέγγιση βάσει μονοπατιών* (path based approach). Γενικά, σε ένα σχεσιακό σχήμα που προκύπτει με βάση τα μονοπάτια [YASU01], οι κόμβοι ενός XML κειμένου αποθηκεύονται σε διαφορετικούς πίνακες της σχεσιακή βάσης δεδομένων, ανάλογα με το είδος τους (στοιχεία, γνωρίσματα ή κόμβοι τιμών). Επιπλέον, υπάρχει ένας πίνακας που ονομάζεται *πίνακας μονοπατιών* (Path Table) και περιέχει όλα τα πιθανά μονοπάτια του XML δένδρου. Ακόμα, κάθε μονοπάτι προσδιορίζεται μονοσήμαντα από έναν μοναδικό αριθμό ταυτοποίησης (path identification number ή path-id), μέσω του οποίου τα στοιχεία του XML κειμένου μπορούν να συσχετίζονται με το εκάστοτε μονοπάτι.

Για την MXML, χρησιμοποιούνται, παρόμοια με την προσέγγιση βάσει είδους, τρεις πίνακες στο σχεσιακό σχήμα για την αποθήκευση των διαφόρων ειδών MXML κόμβων. Έτσι έχουμε έναν *πίνακα στοιχείων* (Element Table) για την αποθήκευση των κόμβων στοιχείων, έναν *πίνακα γνωρισμάτων* (Attribute Table) για την αποθήκευση των κόμβων γνωρισμάτων και έναν *πίνακα τιμών* (Value Table) για την αποθήκευση των κόμβων τιμών. Κάθε ένας από αυτούς τους πίνακες περιέχει σαν πεδία τον αριθμό ταυτοποίησης του κάθε κόμβου (node id) με βάση το Dewey σχήμα δεικτοδότησης και έναν μοναδικό αριθμό ταυτοποίησης μονοπατιού (path-id), που προσδιορίζει το μονοπάτι στον πίνακα μονοπατιών το οποίο οδηγεί στον κόμβο αυτόν. Για τον πίνακα τιμών, η τιμή του κάθε κόμβου αποθηκεύεται επιπρόσθετα στον πίνακα. Σε ότι αφορά τον πίνακα μονοπατιών, αυτός περιέχει όλα τα πιθανά μονοπάτια του MXML κειμένου ξεκινώντας από τον κόμβο της ρίζας και αντιστοιχώντας σε κάθε μονοπάτι τον μοναδικό αριθμό ταυτοποίησης του μονοπατιού. Όπως φαίνεται στο παρακάτω παράδειγμα, τα μονοπάτια που αποθηκεύονται στον πίνακα μονοπατιών και αποκαλούνται *απλές εκφράσεις μονοπατιών* (simple path expressions) είναι εκφράσεις μονοπατιών με δύο επιπλέον χαρακτηριστικά. Το πρώτο χαρακτηριστικό είναι ο συμβολισμός ”$->$” που χρησιμοποιείται κατά των σχηματισμό MXML ερωτημάτων (βλέπε Ενότητα 4.3) για την υπόδειξη των πολυδιάστατων κόμβων και το δεύτερο είναι ο χαρακτήρας “#”, ο οποίος προστίθεται πριν από κάθε “/” άξονα για τη διευκόλυνση της μετατροπής

MXML ερωτημάτων σε SQL ερωτήματα.

Επιπλέον υπάρχουν δύο ακόμα πίνακες στο σχεσιακό σχήμα, ο *πίνακας ρητού περιβάλλοντος* (explicit context ή EC table) και ο *πίνακας κληρονομούμενης κάλυψης* (inherited context coverage ή ICC table), που χρησιμοποιούνται για την αποθήκευση πληροφοριών ερμηνευτικού περιβάλλοντος των MXML κόμβων (ρητό περιβάλλον ή κληρονομούμενη κάλυψη περιβάλλοντος αντίστοιχα) με μία μορφή *δυαδικών διανυσμάτων κόσμων* (binary-based world vectors), όπως περιγράφεται στην ενότητα 3.3.2.

Παράδειγμα 3.3 *Στα παρακάτω σχήματα, απεικονίζονται τμήματα των πινάκων που περιλαμβάνονται στο σχεσιακό σχήμα με βάση τα μονοπάτια, για την αποθήκευση του MXML κειμένου που αναπαριστάται στο MXML δένδρο του Σχήματος 2.2. Πιο συγκεκριμένα, στο Σχήμα 3.6 φαίνονται οι πίνακες που αποθηκεύουν τα διάφορα είδη κόμβων, στο Σχήμα 3.7 ο πίνακας που περιέχει όλα τα πιθανά μονοπάτια και στο Σχήμα 3.8 οι πίνακες που αποθηκεύουν τις πληροφορίες σχετικά με τα ερμηνευτικά περιβάλλοντα των MXML κόμβων.*

Element Table	
node_id	path_id
1	1
1.1	2
1.1.2	5
1.1.2.1	6
....	....

Attribute Table	
node_id	path_id
1.1.1	3
1.1.1.1	4
1.1.1.2	4

Value Table		
node_id	path_id	value
1.1.1.1.1	4	0-13-110362-8
1.1.1.2.1	4	0-13-110370-9
1.1.2.1.1	6	The C programming language
....	....	....

Σχήμα 3.6: Σχεσιακό σχήμα βάσει μονοπατιών: Πίνακες κόμβων

Σημειώνουμε την αναπαράσταση του ερμηνευτικού περιβάλλοντος για κάθε κόμβο του MXML κειμένου στον πίνακες ρητού περιβάλλοντος και κληρονομούμενης κά-

Path Table	
path_id	path
1	#/->book
2	#/book
3	#/book#/->@isbn
4	#/book#/@isbn
5	#/book#/->title
6	#/book#/title
....	....

Σχήμα 3.7: Σχεσιακό σχήμα βάσει μονοπατιών: Πίνακας μονοπατιών

EC_Table	
node_id	world vector
....	....
1.1.7.2	000111
1.1.7.2.1	111111
1.1.7.2.1.1	111111
1.1.7.2.2	111111
1.1.7.2.2.1	100100
....	....

ICC_Table	
node_id	world vector
....	....
1.1.7.2	000111
1.1.7.2.1	000111
1.1.7.2.1.1	000111
1.1.7.2.2	000101
1.1.7.2.2.1	000100
....	....

Σχήμα 3.8: Σχεσιακό σχήμα βάσει μονοπατιών - Πίνακες ερμηνευτικού περιβάλλοντος

λυψης με την χρήση των δυαδικών διανυσμάτων (Σχήμα 3.8). Επιπλέον, μπορούμε να δούμε ότι τα μονοπάτια προς τα διάφορα στοιχεία περιβάλλοντος ή πολυδιάστατα στοιχεία ή γνωρίσματα, αποθηκεύονται στον πίνακα μονοπατιών μαζί με την προσθήκη του ειδικού χαρακτήρα "`#`" πριν από κάθε "`/`" άξονα (Σχήμα 3.7). Αυτή είναι μία τεχνική που χρησιμοποιείται και στο XRel [YASU01], προκειμένου να επιλυθεί το πρόβλημα της μετατροπής XML ερωτημάτων που περιέχουν "`//`" άξονες σε SQL ερωτήματα, χρησιμοποιώντας κατάλληλες SQL συνθήκες (για παράδειγμα, το ερώτημα που περιγράφεται από την έκφραση "`/book//price`" θα μπορούσε να αποδοθεί από το αλφαριθμητικό "`#/book#%/price`" σε μία LIKE έκφραση ενός SQL ερωτήματος, προκειμένου να επιστρέψει σαν αποτέλεσμα του ερωτήματος τους

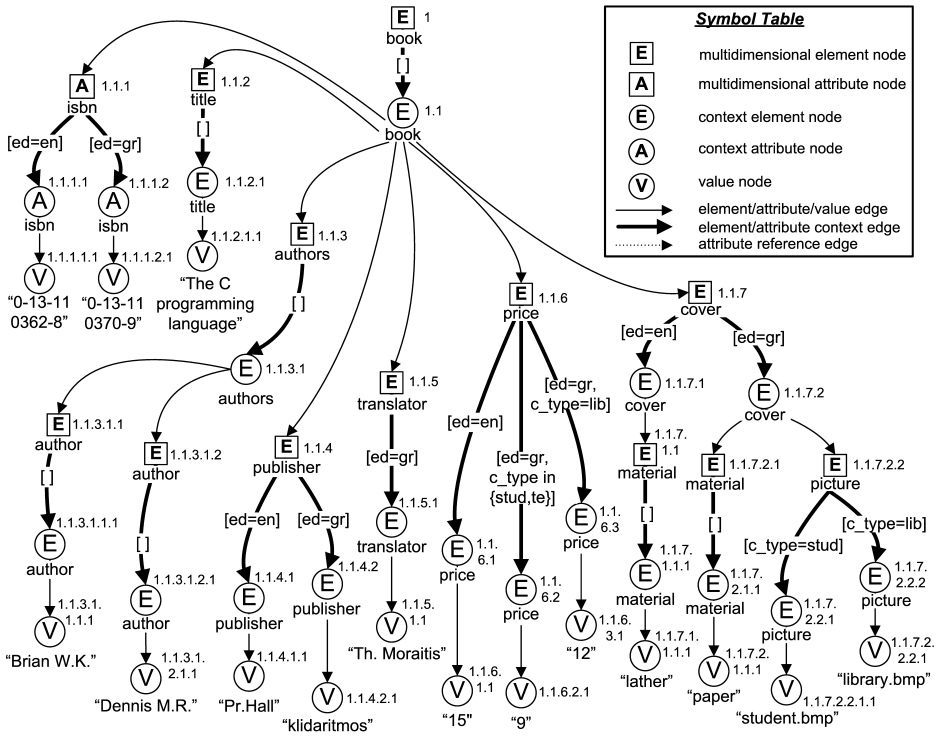
κόμβους "`price`" που είναι απόγονοι του κόμβου "`book`", με βάση τα μονοπάτια που έχουν αντιστοιχηθεί στους κόμβους αυτούς).

3.3.1 Τεχνικές αρίθμησης και διάταξης

Σε συνδυασμό με την προσέγγιση βάσει μονοπατιών, η τεχνική που χρησιμοποιείται για τη δεικτοδότηση (indexing) των κόμβων του MXML δένδρου ακολουθεί τα πρότυπα του Dewey σχήματος δεικτοδότησης [TVB+02]. Γενικά, το σχήμα αυτό αντιστοιχίζει σε κάθε κόμβο έναν μοναδικό αριθμό ταυτοποίησης (identification number), ο οποίος διαμορφώνεται με τη χρήση τελειών (dotted format), με βάση τη θέση του κάθε κόμβου στην ιεραρχία (ιεραρχικό επίπεδο και θέση μεταξύ των αδερφικών κόμβων) του MXML δένδρου. Έτσι, εάν θεωρήσουμε ότι $N_1, N_2, \ldots, N_d$ είναι μία ακολουθία από κόμβους που περιέχονται σε κάποιο μονοπάτι του MXML δένδρου, τέτοια ώστε ο N_1 να είναι η ρίζα του δένδρου και ο N_{i-1} να είναι ο πατρικός κόμβος του κόμβου N_i, μπορούμε να ορίσουμε τον Dewey *δείκτη* (index) του κόμβου N_d αποδιδόμενο σαν τον διαμορφωμένο με την χρήση τελειών αριθμό ταυτοποίησης $s_{N_1}.s_{N_2}.s_{N_3}\ldots s_{N_d}$, όπου s_{N_i} αποτελεί τον αριθμό θέσης του κόμβου N_i μεταξύ των αδερφικών του κόμβων.

Παράδειγμα 3.4 *Στο Σχήμα 3.9, μπορούμε να δούμε την αναπαράσταση του MXML κειμένου του Παραδείγματος 2.2 σαν MXML δέντρο του οποίου οι κόμβοι απαριθμούνται με βάση το Dewey σχήμα δεικτοδότησης. Παρατηρήστε τη διαμόρφωση με τη χρήση τελειών η οποία ακολουθείται, με τον αριθμό των τελειών να αντιπροσωπεύει το ιεραρχικό επίπεδο (level) του κάθε κόμβου και τον αριθμό μετά από κάθε τελεία να παριστάνει την θέση του κόμβου αυτού μεταξύ των αδελφικών του κόμβων. Για παράδειγμα, ο κόμβος* `1.1` *είναι κόμβος του πρώτου επιπέδου (ξεκινώντας από το επίπεδο 0 της ρίζας του δένδρου) και το πρώτο παιδί του κόμβου 1.*

Στα πλαίσια της προσέγγισης βάσει των μονοπατιών και προκειμένου να επιτευχθεί η αποθήκευση της πληροφορίας που αφορά τα ερμηνευτικά περιβάλλοντα στο σχεσιακό σχήμα με έναν πιο εύκολο και ευέλικτο τρόπο, αναπτύχθηκε μία τεχνική διάταξης (ordering) σύμφωνα με την οποία επιτυγχάνεται η διάταξη των πιθανών κόσμων ενός MXML κειμένου. Να σημειώσουμε, ότι η τεχνική αυτή διευκολύνει σημαντικά τον σχηματισμό ερωτημάτων με βάση τα ερμηνευτικά περιβάλλοντα (context-aware queries).

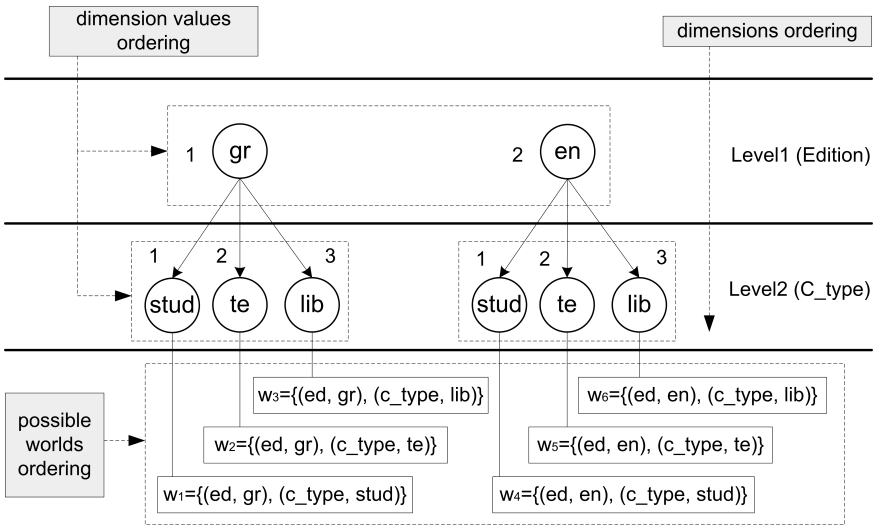


Σχήμα 3.9: Γραφική αναπαράσταση του MXML δένδρου με τη χρήση της δεικτοδότησης Dewey

Η βασική ιδέα η οποία περιλαμβάνεται στην παραπάνω τεχνική διάταξης, είναι η επίτευξη μίας *καθολικής διάταξης* (total ordering) όλων των πιθανών κόσμων, με βάση α) την καθολική διάταξη των διαστάσεων και β) την καθολική διάταξη των στοιχείων που περιλαμβάνονται στο σύνολο των πιθανών τιμών της κάθε διάστασης. Έτσι, για k διαστάσεις με κάθε διάσταση i να μπορεί να πάρει m_i πιθανές τιμές, μπορούμε να έχουμε $n = m_1 * m_2 * \ldots * m_k$ πιθανούς διατεταγμένους κόσμους. Σε κάθε έναν από τους κόσμους αυτούς αντιστοιχίζεται ένας μοναδικός ακέραιος αριθμός μεταξύ του 1 και του n.

Παράδειγμα 3.5 *Στο Σχήμα. 3.10, παρουσιάζεται το πως είναι δυνατόν να γίνει η διάταξη όλων των πιθανών κόσμων, με βάση τις διαστάσεις και τις τιμές των διαστάσεων του παραδείγματος 2.2.*

Προκειμένου να είναι πιο κατανοητή η διάταξη που επιτυγχάνεται, χρησιμοποιούμε ένα δάσος (forest) από δέντρα. Όπως φαίνεται, κάθε διάσταση του MXML



Σχήμα 3.10: Διάταξη των πιθανών κόσμων

κειμένου αντιστοιχεί σε ένα επίπεδο (level) του δάσους. Η διάταξη αυτών των επιπέδων αναπαριστά τη διάταξη των διαστάσεων. Ακόμα, για κάθε επίπεδο μπορούμε να δούμε την διάταξη όλων των πιθανών τιμών της σχετικής διάστασης, κάτω από κάθε κόμβο του προηγούμενου επιπέδου. Κάθε πιθανός κόσμος μπορεί να παραχθεί διασχίζοντας ένα μονοπάτι από έναν κόμβο-ρίζα του δάσους προς έναν κόμβο-φύλλο του αντίστοιχου δένδρου. Τελικά, η διάταξη των φύλλων του δάσους αντιπροσωπεύουν την καθολική διάταξη όλων των πιθανών κόσμων, αντιστοιχίζοντας έναν μοναδικό ακέραιο αριθμό σε κάθε κόσμο ($w_1, w_2, \ldots, w_6$).

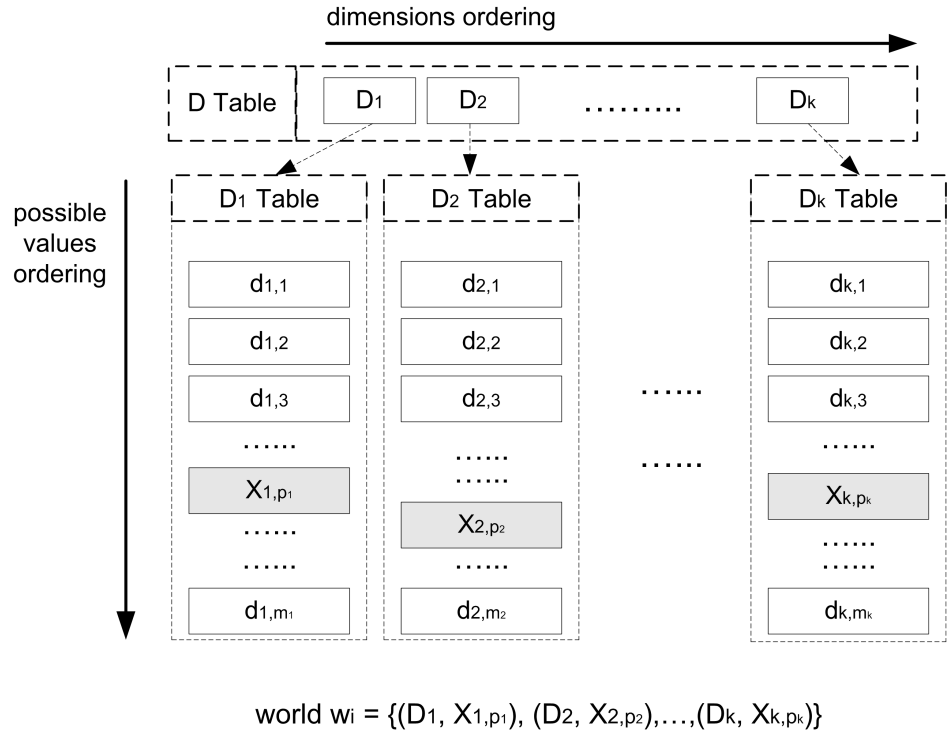
3.3.2 Αναπαράσταση ερμηνευτικού περιβάλλοντος βάσει διάταξης

Θεωρώντας ότι όλοι οι πιθανοί κόσμοι ενός MXML κειμένου βρίσκονται σε καθολική διάταξη, μπορούμε να ορίσουμε ένα διάνυσμα από *δυαδικά ψηφία* (bits) το οποίο ονομάζεται *διάνυσμα κόσμων* (world vector). Η τεχνική αυτή αποκαλείται *αναπαράσταση ερμηνευτικού περιβάλλοντος βάσει διάταξης* (ordered-based context representation).

Ορισμός 3.3.1. Έστω μία καθολική διάταξη κόσμων $W = (w_1, w_2, \ldots, w_n)$, όπου n είναι ο αριθμός των πιθανών κόσμων, ορίζουμε ως $V(c) = (a_1, a_2, \ldots, a_n)$ το

διάνυσμα κόσμων ενός προσδιοριστή περιβάλλοντος c, όπου a_i με $i = 1, 2, \ldots, n$, είναι ένα δυαδικό ψηφίο που περιέχει 1 εάν ο κόσμος w_i βρίσκεται μεταξύ των κόσμων που αναπαριστώνται από τον c ή 0 εάν ο κόσμος w_i δεν περιλαμβάνεται μεταξύ των κόσμων που αναπαριστά ο c.

Στο σχήμα 3.11 μπορούμε να δούμε πως γενικά μπορούμε να αποθηκεύσουμε πληροφορίες σχετικές με τις διαστάσεις σε μία σχεσιακή βάση δεδομένων. Ένα πίνακας (Table D) χρησιμοποιείται για την αποθήκευση διατεταγμένων διαστάσεων και ένας ξεχωριστός πίνακας D_i όπου $i = 1, 2, \ldots, k$ χρησιμοποιείται για την αποθήκευση των διατεταγμένων τιμών $d_{i,j}$ με $j = 1, 2, \ldots, m_i$ και m_i να είναι ο αριθμός των διαφορετικών τιμών της διάστασης D_i.



Σχήμα 3.11: Αναπαράσταση ερμηνευτικού περιβάλλοντος βάσει διάταξης στο σχεσιακό σχήμα

D Table	
dimension_id	dimention_name
1	edition
2	customer_type

D_1 Table	
value_id	value
1	greek
2	english

D_2 Table	
value_id	value
1	student
2	teacher
3	library

Σχήμα 3.12: Πίνακες διάταξης διαστάσεων

Εντοπισμός της θέσης ενός κόσμου μέσα σε ένα διάνυσμα κόσμων

Ένα πρόβλημα το οποίο προκύπτει όταν χρησιμοποιούμε την τεχνική αναπαράστασης των κόσμων βάσει διάταξης, είναι το πρόβλημα του καθορισμού της θέσης κάποιου συγκεκριμένου κόσμου, μέσα σε ένα διάνυσμα κόσμων.

Όπως φαίνεται στο Σχήμα 3.11, θεωρώντας ότι ένας προσδιοριστής περιβάλλοντος περιέχει έναν κόσμο w_i, και ότι κάθε αριθμός $p_1, p_2, \ldots, p_k$ αναπαριστά την θέσης κάποιας τιμής μεταξύ των διατεταγμένων τιμών της κάθε διάστασης $D_1, D_2, \ldots, D_k$ αντίστοιχα, μπορούμε να προσδιορίσουμε τη θέση του δυαδικού ψηφίου i που αντιστοιχεί σε αυτόν τον κόσμο μέσα στο διάνυσμα κόσμων του προσδιοριστή περιβάλλοντος, χρησιμοποιώντας τον παρακάτω τύπο:

$$i = p_k + \sum_{j=2}^{k}[(p_{j-1} - 1) * (\prod_{w=j}^{k} m_w)]$$

Παράδειγμα 3.6 *Το Σχήμα 3.13, απεικονίζει τμήματα των πινάκων ρητού περιβάλλοντος και κληρονομούμενης κάλυψης, οι οποίοι προέκυψαν από την κωδικοποίηση της πληροφορίας ερμηνευτικού περιβάλλοντος που υπάρχει στο MXML δένδρο του σχήματος 3.9. Οι πληροφορίες που αφορούν τη διάταξη των πιθανών κόσμων διατηρούνται στους πίνακες του Σχήματος 3.12, όπου ο πίνακας D χρησιμοποιείται για την αποθήκευση των δύο διατεταγμένων διαστάσεων ($edition$ και $customer\_type$) του σχετικού παραδείγματος και οι πίνακες D_1, D_2 για την αποθήκευση των διατεταγμένων τιμών των διαστάσεων αυτών.*

Για παράδειγμα, το ρητό περιβάλλον του κόμβου με `node_id=1.1.1.1` *περιλαμβάνει τους κόσμους:*

$w'_1 = \{$`(edition, english)`, `(customer_type, student)`$\}$,
$w'_2 = \{$`(edition, english)`, `(customer_type`, `teacher)`$\}$ *και*

Inherited Coverage Table	
node_id	world_vector
1.1	111111
1.1.1	111111
1.1.1.1	000111
1.1.1.1.1	000111
1.1.1.2	111000
1.1.1.2.1	111000
....	....

Explicit Context Table	
node_id	world_vector
1.1	111111
1.1.1	111111
1.1.1.1	000111
....	....
1.1.6.3	001000
....	....
1.1.7.2.2.1	100100
....	....

Σχήμα 3.13: Πίνακες ερμηνευτικού περιβάλλοντος

$w'_3 = \{$`(edition, english)`, `(customer_type, library)`$\}$

Σύμφωνα με τη διάταξη του σχήματος 3.10, οι θέσεις των δυαδικών ψηφίων αυτών των κόσμων στο σχετικό διάνυσμα κόσμων είναι οι θέσεις 4, 5 και 6 αντίστοιχα. Αυτό έχει σαν αποτέλεσμα, ότι ο προσδιοριστής ρητού περιβάλλοντος του κόμβου κωδικοποιείται στον πίνακα ρητού περιβάλλοντος σαν μία γραμμή με `node_id=1.1.1.1` *και διάνυσμα κόσμων* `000111`.

Εντοπισμός του κόσμου που αντιστοιχεί σε ένα δυαδικό ψηφίο του διανύσματος κόσμων

Το αντίστροφο πρόβλημα της ανεύρεσης της θέσης ενός κόσμου μέσα σε ένα διάνυσμα κόσμων είναι το πρόβλημα κατά το οποίο καλούμαστε να εντοπίσουμε ποιος κόσμος αντιστοιχεί στη θέση ενός δυαδικού ψηφίου i ενός διανύσματος κόσμων. Προκειμένου να επιτευχθεί αυτό, μπορούμε να χρησιμοποιήσουμε τον ακόλουθο αλγόριθμο που εκφράζεται σαν διαδικασία (function) προγράμματος, χρησιμοποιώντας τη σημασιολογία του Σχήματος 3.11.

Function Convert_bit-position_i_of_world_vector_to_world(*i*)
Input: Η θέση του δυαδικού ψηφίου *i* σε ένα διάνυσμα κόσμων.
Output: Ένας κόσμος $(p_1, p_2, ..., p_k)$
begin
- **If** (i>1) **then**
  - k'=1

- i'=i-1
- **While** (k'<k) **do**
  - $a_{k'}$=i' DIV ($m_{k'+1}$*$m_{k'+2}$*...*m_k)
  - $u_{k'}$=i' MOD ($m_{k'+1}$*$m_{k'+2}$*...*m_k)
  - **If** ($a_{k'}$=0) **then**
    - $p_{k'}$=1
  - **else**
    - $p_{k'}$=$a_{k'}$+1
  - **end if**
  - k'=k'+1
  - i'=$u_{k'}$
- **end while**
- **If** (i'=0) **then**
  - p_k=1
- **else**
  - p_k=i_k+1
- **end if**

- else

- p_1=p_2=...=p_k=1

- end if

end

Ο παραπάνω αλγόριθμος, έχει σαν είσοδο (input) την θέση i ενός κόσμου σε ένα διάνυσα κόσμων. Το αποτέλεσμα του αλγορίθμου (output) είναι μία ακολουθία αριθμών $(p_1, p_2, \ldots, p_k)$. Κάθε αριθμός p_i παριστάνει την θέση μίας τιμής ανάμεσα στις διατεταγμένες τιμές της διάστασης D_i. Χρησιμοποιώντας αυτή τη θέση, εξάγουμε την τιμή X_{i,p_i} της διάστασης D_i από τον αντίστοιχο πίνακα D_i του Σχήματος 3.11. Το σύνολο των ζευγαριών (D_i, X_{i,p_i}) αναπαριστούν τον κόσμο που αποτελεί το αποτέλεσμα του αλγορίθμου.

Στο Παράδειγμα 3.6, σε ότι αφορά τον κόμβο με `node_id=1.1.1.1`, το αποτέλεσμα του παραπάνω αλγορίθμου για κάθε θέση δυαδικού ψηφίου που περιέχει την τιμή 1 στο διάνυσμα κόσμων (000111) του κόμβου, περιλαμβάνει την ακολουθία (2,1), (2,2) και (2,3), η οποία αναπαριστά τους κόσμους w'_1, w'_2 και w'_3 αντίστοιχα.

3.3.3 Πράξεις - Συγκρίσεις βάσει διάταξης για ερμηνευτικά περιβάλλοντα

Κατά τη διαδικασία μετατροπής MXML ερωτημάτων σε SQL ερωτήματα, είναι απαραίτητο να υπάρχει η δυνατότητα μεταφοράς τον διαφόρων συνθηκών που τίθενται στα MXML ερωτήματα στις αντίστοιχες συνθήκες που εκφράζονται στα SQL ερωτήματα. Προκειμένου να επιτευχθεί αυτό, χρειάζεται μία διαδικασία με βάση την οποία θα είναι δυνατή η σύγκριση συνόλων από κόσμους που εκφράζονται με τη βοήθεια των προσδιοριστών περιβάλλοντος. Σε αυτή την ενότητα ορίζουμε τον τρόπο, σύμφωνα με τον οποίο μπορούμε να εκτελούμε πράξεις (operations) και συγκρίσεις (comparison) μεταξύ συνόλων από κόσμους που εκφράζονται από τους αντίστοιχους προσδιοριστών περιβάλλοντος οι οποίοι έχουν κωδικοποιηθεί σύμφωνα με την τεχνική αναπαράστασης ερμηνευτικού περιβάλλοντος βάσει διάταξης.

Αρχικά, δείχνουμε πως η τομή (intersection) και η ένωση (union) μεταξύ των προσδιοριστών περιβάλλοντος εκτελείται σε επίπεδο διανυσμάτων κόσμων.

Λήμμα 3.3.1. Θεωρούμε c_1, c_2 δύο προσδιοριστές περιβάλλοντος και b_1, b_2 τα διανύσματα κόσμων των c_1, c_2 αντίστοιχα. Τότε το διάνυσμα κόσμων b_3 της τομής των ερμηνευτικών περιβαλλόντων (context intersection) $c_1 \cap^c c_2$ προκύπτει εφαρμόζοντας την πράξη AND[2] στα αντίστοιχα δυαδικά ψηφία (bits) των b_1 και b_2. Αντίστοιχα, το διάνυσμα κόσμων b_4 της ένωσης των ερμηνευτικών περιβαλλόντων (context union) $c_1 \cup^c c_2$ προκύπτει εφαρμόζοντας την πράξη OR[3] στα αντίστοιχα δυαδικά ψηφία (bits) των b_1 και b_2.

Παράδειγμα 3.7 *Έστω οι προσδιοριστές περιβάλλοντος:*
$c_1 = [edition = english]$, *και*
$c_2 = [edition = english, customer\_type = student]$.
Όπως δείξαμε στο παράδειγμα 3.6 το διάνυσμα κόσμων του προσδιοριστή περιβάλλοντος c_1 *είναι* $V(c_1)$*=000111. Κατά όμοιο τρόπο, προκύπτει ότι* $V(c_2)$*=000100. Τότε έχουμε:*
$b_3 = V(c_1 \cap^c c_2) = 000111\ AND_b\ 000100 = 000100$

[2]Για την δυαδική πράξη AND θα χρησιμοποιούμε την συντομογραφία AND_b.
[3]Για την δυαδική πράξη OR θα χρησιμοποιούμε την συντομογραφία OR_b.

και
$b_4 = V(c_1 \cup^c c_2) = 000111\ OR_b\ 000100 = 000111$

Είναι επίσης δυνατή η σύγκριση δύο προσδιοριστών περιβάλλοντος χρησιμοποιώντας τα διανύσματα κόσμων αυτών. Αυτό είναι κάτι πολύ χρήσιμο όταν προσπαθούμε να μετασχηματίσουμε MXML ερωτήματα που περιέχουν σχετικά συνθήκες σε SQL ερωτήματα προς μία σχεσιακή βάση δεδομένων. Αυτές οι συνθήκες εμπεριέχουν συγκρίσεις μεταξύ των προσδιοριστών περιβάλλοντος που είναι αποθηκευμένοι μαζί με το MXML κείμενο μέσα στο σχεσιακό σχήμα και των προσδιοριστών περιβάλλοντος οι οποίοι χρησιμοποιούνται στα MXML ερωτήματα. Παρόμοια με τα AND_b και OR_b, στο Λήμμα 3.3.3 χρησιμοποιούμε τη συντόμευση XOR_b για την δυαδική πράξη XOR.

Λήμμα 3.3.2. Έστω οι προσδιοριστές περιβάλλοντος c_1, c_2 και b_1, b_2 τα διανύσματα κόσμων των c_1, c_2 αντίστοιχα. Τότε

1. c_1 = c_2 εάν και μόνο εάν b_1 = b_2, εναλλακτικά c_1 = c_2 εάν και μόνο εάν (b_1 XOR_b b_2) = 0
2. $c_1 \neq c_2$ εάν και μόνο εάν NOT(b_1 = b_2)
3. $c_1 \geq c_2$ εάν και μόνο εάν (b_1 AND_b b_2) = b_2
4. $c_1 > c_2$ εάν και μόνο εάν ((b_1 AND_b b_2) = b_2) και ($b_1 \neq b_2$).

Παράδειγμα 3.8 *Έστω οι προσδιοριστές περιβάλλοντος:*
$c_1 = [edition = english]$ *και*
$c_2 = [edition = english, customer\_type = student]$.
Υπολογίζοντας τα διανύσματα κόσμων αυτών των δύο προσδιοριστών περιβάλλοντος, έχουμε $V(c_1)$*= 000111=*b_1 *και* $V(c_2)$*=000100=*b_2*. Τότε η έκφραση* $c_1 \geq c_2$ *είναι , αφού* (b_1 AND_b b_2) = (000111 AND_b 000100) = 000100 = b_2 *(Βλέπε περίπτωση 3 του Λήμματος 3.3.3).*

3.4 Συμπεράσματα - Παρατηρήσεις

Όπως και στην περίπτωση της συμβατικής XML, έτσι και σε ότι αφορά την MXML, το πρόβλημα της αποθήκευσης MXML δεδομένων σε σχεσιακές βάσεις, αποτελεί ένα πολύ ενδιαφέρον πεδίο έρευνας. Μέσω της δυνατότητας αυτής, επιτυγχάνεται η ολοκλήρωση του MXML με το σχεσιακό σχήμα, ενώ δίνεται η δυνατότητα εκμετάλλευσης όλης της υπάρχουσας υποδομής που παρέχουν τα συστήματα σχεσιακών βάσεων δεδομένων.

Σε ότι αφορά τη σύγκριση των παραπάνω μεθόδων αποθήκευσης MXML κειμένων σε σχεσιακές βάσεις δεδομένων, μπορούμε να παρατηρήσουμε ότι αν και η βασική προσέγγιση είναι απλούστερη στην εφαρμογή της, ωστόσο παρουσιάζει αρκετά μειονεκτήματα εξαιτίας της χρήσης ενός και μοναδικού πίνακα στο σχεσιακό σχήμα. Καθώς τα διάφορα είδη των κόμβων αποθηκεύονται σε έναν πίνακα, πολλές κενές (NULL) τιμές παρουσιάζονται στα πεδία `explicit_context`, `tag` και `value`. Αυτές οι τιμές αποφεύγονται κατά την αποθήκευση των κόμβων σε διαφορετικούς πίνακες με βάση το είδος τους, πράγμα που εφαρμόζεται κατά την προσέγγιση βάσει είδους.

Επιπλέον, κατά τη βασική προσέγγιση, ερωτήματα προς MXML δεδομένα εμπλέκουν έναν μεγάλο αριθμό από self-joins του πίνακα κόμβων (Node Table). Αυτό επιβαρύνει πολύ τον χρόνο εκτέλεσης των ερωτημάτων αυτών αφού ο πίνακας κόμβων είναι αρκετά μεγάλος περιέχοντας ολόκληρο το MXML δένδρο. Διασπώντας λοιπόν, κατά την προσέγγιση βάσει είδους, τον πίνακα κόμβων, επιτυγχάνουμε την μείωση του μεγέθους των πινάκων που εμπλέκονται στα αντίστοιχα joins και την βελτίωση του χρόνου εκτέλεσης των ερωτημάτων.

Σχετικά με την αναπαράσταση του ερμηνευτικού περιβάλλοντος, η βασική προσέγγιση ξανά υστερεί σε θέματα απόδοσης των MXML ερωτημάτων, αφού και σε αυτή την περίπτωση το μέγεθος των σχετικών πινάκων είναι αρκετά μεγάλο. Ωστόσο, με βάση την τεχνική αναπαράστασης ερμηνευτικού περιβάλλοντος βάσει διάταξης, προτείνεται ένα σχήμα το οποίο μειώνει σημαντικά το μέγεθος των πινάκων, αρά και τον αριθμό των joins στα ερωτήματα που περιλαμβάνουν συνθήκες σύμφωνα με τα ερμηνευτικά περιβάλλοντα.

Συμπερασματικά, και με βάση τα παραπάνω, είναι φανερό ότι η προσέγγιση που περιλαμβάνει την αναπαράσταση ερμηνευτικού περιβάλλοντος βάσει διάταξης αποτελεί μία ολοκληρωμένη και αποδοτική λύση σε ότι αφορά το πρόβλημα

της αποθήκευσης MXML δεδομένων σε σχεσιακές βάσεις. Ιδιαιτέρως η τεχνική της αναπαράστασης του ερμηνευτικού περιβάλλοντος μέσω των διανυσμάτων κόσμων καθιστά το εν λόγο μοντέλο ιδιαίτερα ευέλικτο, κυρίως σε ότι αφορά την δυνατότητα εκτέλεσης πράξεων μεταξύ των συνόλων των κόσμων που αντιπροσωπεύουν οι προσδιοριστές περιβάλλοντος ενός MXML δένδρου. Με τον τρόπο αυτό, διευκολύνεται σημαντικά η διαδικασία μετατροπής MXML ερωτημάτων σε SQL ερωτήματα που απευθύνονται προς το σχεσιακό σχήμα, αφού οι περιορισμοί που τίθενται με βάση το context της MXML απεικονίζονται άμεσα σε αντίστοιχες συνθήκες στην γλώσσα SQL.

Κεφάλαιο 4

Επέκταση της XPath

4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται μία επέκταση της γλώσσας επερωτήσεων XPath, η οποία ονομάζεται *Πολυδιάστατη MXPath* (Multidimensional XPath ή MXPath) για την πλοήγηση (navigation) και την υποβολή ερωτημάτων (querying) σε MXML κείμενα. Η MXPath χρησιμοποιεί ερμηνευτικά περιβάλλοντα με σκοπό τον προσδιορισμό μονοπατιών πλοήγησης στα MXML δένδρα. Παρακάτω, παρουσιάζεται η σύνταξη της MXPath και επιδεικνύεται η χρήση της μέσω ενός αριθμού παραδειγμάτων. Σύμφωνα με τα παραδείγματα αυτά, υποβάλλονται σχετικά ερωτήματα σε MXML δεδομένα. Επιπλέον, διερευνούμε την σχέση της MXPath με την συμβατική XPath, δεδομένου ότι με την απλή XML εκφράζονται διαφορετικά στιγμιότυπα των MXML κειμένων.

Μέχρι τώρα, αρκετές γλώσσες υποβολής XML ερωτημάτων έχουν προταθεί [AQM+97, BC00, GB04, PCL02]. Η XPath [CON07c] παρέχει τη δυνατότητα πλοήγησης μέσα σε ένα XML κείμενο και για τον λόγο αυτό, αποτελεί το βασικό συστατικό σε πολλές γλώσσες υποβολής ερωτημάτων [CON99b, CON07b, YASU01]. Από την άλλη πλευρά, υπάρχουν κάποιες ερευνητικές εργασίες [AYU00, Dyr01, ZD02] που προτείνουν επεκτάσεις της XPath, αν και οι εργασίες αυτές περιλαμβάνουν κυρίως επεκτάσεις που σχετίζονται με τον παράγοντα του χρόνου (temporal επεκτάσεις). Πιο συγκεκριμένα στο [ZD02] περιγράφεται μία επέκταση της XPath που περιλαμβάνει τον λεγόμενο *έγκυρο χρόνο* (valid time), τον χρόνο δηλαδή κατά τον οποίο κάποιο γεγονός λαμβάνει χώρα ή μία οντότητα

αποκτά θετική τιμή αληθείας (true) εντός του πραγματικού κόσμου. Ακόμα, στο [AYU00] προτείνεται ένα λογικό μοντέλο δεδομένων για την αναπαράσταση ιστορικών εκδοχών XML εγγράφων. Το προτεινόμενο αυτό μοντέλο επεκτείνει την XPath ώστε να δύναται να απεικονίσει ιστορικές αλλαγές σε XML κείμενα. Τέλος, στο [Dyr01] παρουσιάζεται μία επέκταση της XPath με βάση τον χρόνο συνδιαλλαγής (transaction-time XPath ή TTPATH), δηλαδή τον χρόνο αλληλεπίδρασης με το σύστημα και όχι με τον πραγματικό κόσμο. Έτσι, η TTXPath γλώσσα προσδίδει στην XPath έναν επιπλέον άξονα (transaction-time axis), ώστε να μπορεί να εκφράζει ερωτήματα τα οποία έχουν πρόσβαση σε προγενέστερες ή μεταγενέστερες καταστάσεις ενός XML κειμένου, χρησιμοποιώντας δομές που επιτρέπουν την διατήρηση και τη σύγκριση με βάση τον χρόνο.

4.2 XPath

Η XPath (XML Path Language) [CON07c] είναι μία γλώσσα που έχει προταθεί από το W3C με σκοπό τον εντοπισμό τμημάτων ενός XML κειμένου. Η XPath βασίζεται στη δενδροειδή αναπαράσταση των XML δεδομένων και παρέχει τη δυνατότητα πλοήγησης ανάμεσα στα στοιχεία (elements) και τα γνωρίσματα (attributes), μέσα σε ένα XML δένδρο, επιλέγοντας κόμβους με βάση τον καθορισμό διαφόρων κριτηρίων. Το βασικό δομικό στοιχείο της XPath είναι η *XPath έκφραση* (XPath expression), η οποία μπορεί να επιστρέφει είτε ένα σύνολο κόμβων (node-set), είτε ένα αλφαριθμητικό (string), είτε μία δυαδική τιμή (boolean), ή έναν αριθμό. Το πιο κοινό είδος XPath έκφρασης είναι η *έκφραση μονοπατιού* (path expression ή location path expression). Μία έκφραση μονοπατιού εκφράζεται με μία ακολουθία από *βήματα* (τα ονομαζόμενα *location steps*) προκειμένου να μεταβούμε από έναν XML κόμβο (τον τρέχοντα κόμβο περιβάλλοντος) σε έναν άλλον κόμβο ή σύνολο κόμβων. Να σημειώσουμε ότι ο όρος ”κόμβος περιβάλλοντος” (context node) που χρησιμοποιείται στην XPath αναφέρεται στον εκάστοτε κόμβο του XML δένδρου, ο οποίος θεωρείται ο τρέχον κόμβος για την εκτέλεση της XPath έκφρασης, σε αντίθεση με την σημασία του όρου ”context” όταν χρησιμοποιείται στην MXML και την MXPath. Τα βήματα μία εκφράσεως μονοπατιού χωρίζονται με ”/” χαρακτήρες. Χρησιμοποιώντας τις εκφράσεις μονοπατιού, συγκεκριμένοι κόμβοι μπορεί να επιλέγονται σε ένα XML κείμενο, όταν ακολουθούνται καθορισμένα βήματα. Ένα βήμα εκφράσεως μονοπατιού αποτελείται από

τρία τμήματα:

1. έναν *άξονα* (axis), ο οποίος προσδιορίζει τη σχέση μεταξύ των επιλεγμένων κόμβων και του τρέχοντος κόμβου, στο XML δένδρο
2. έναν *έλεγχο κόμβου* (node-test), ο οποίος προσδιορίζει έναν κόμβο μέσω ενός άξονα, και
3. μηδέν ή περισσότερα *κατηγορήματα* (predicates), τα οποία θέτουν περισσότερα κριτήρια επιλογής πάνω στο επιστρεφόμενο σύνολο κόμβων.

Η σύνταξη ενός βήματος εκφράσεως μονοπατιού είναι η ακόλουθη:

```
axisname::nodetest[predicate_1]...[predicate_n]
```

Το "/" μπροστά από μία έκφραση μονοπατιού συμβολίζει τη ρίζα ενός κειμένου, ενώ γενικά η XPath μπορεί να έχει είτε *εκτεταμένη* (expanded) ή *συνεπτυγμένη* (abbreviated) σύνταξη.

Παράδειγμα 4.1 *Ας εξετάσουμε την ακόλουθη έκφραση μονοπατιού, η οποία είναι γραμμένη με βάση την εκτεταμένη σύνταξη:*

`child::A/descendant-or-self::node()/child::B`
`/child::*[position()=1]`

Αυτή η έκφραση επιλέγει το πρώτο στοιχείο (που προσδιορίζεται από το κατηγόρημα "`[position() = 1]`"), ανεξάρτητα από το όνομά του (όπως προσδιορίζεται από το "`*`"), το οποίο είναι στοιχείο παιδί ("`child`" axis) ενός `B` στοιχείου το οποίο με τη σειρά του είναι είτε παιδί, είτε παιδί κάποιου απογόνου (όπως προσδιορίζει το "`descendant-or-self::node()`") ενός στοιχείου `A` το οποίο είναι παιδί του τρέχοντος κόμβου περιβάλλοντος (καθώς η έκφραση δεν αρχίζει με "/"). Μπορούμε να παρατηρήσουμε στη παραπάνω "ανεπτυγμένη σύνταξη", ότι σε κάθε βήμα της XPath έκφρασης, η ακμή (πχ. "`child`" ή "`descendant-or-self`") προσδιορίζεται άμεσα, και ακολουθείται από "`::`" και τη συνθήκη κόμβου, όπως η "A" ή η "`node()`". Η παραπάνω XPath έκφραση μπορεί να γραφτεί με την συνεπτυγμένη της μορφή ως εξής:

`A//B/*[1]`

Σε αυτή τη μορφή, η ακμή "`child`" παραλείπεται, η έκφραση "`[position()=1]`" συνοψίζεται σε "`[1]`" και η έκφραση "`/descendant-or-self::node()/`" αντικαθίσταται από "`//`".

4.3 Πολυδιάστατη XPath (MXPath)

Στην ενότητα αυτή παρουσιάζουμε την *Πολυδιάστατη XPath* (Multidimensional XPath ή MXPath) σαν μία επέκταση της XPath, η οποία χρησιμοποιείται για την πλοήγηση μέσα στο MXML δένδρο. Πέρα από τον κλασικό τρόπο λειτουργίας της XPath, η MXPath χρησιμοποιεί επιπρόσθετα τις έννοιες της κληρονομούμενης κάλυψης περιβάλλοντος και του ρητού περιβάλλοντος της MXML, προκειμένου να επιλέγει κόμβους μέσα από το MXML κείμενο. Κατά όμοιο τρόπο με την XPath, η MXPath χρησιμοποιεί και αυτή *εκφράσεις μονοπατιού* (path expressions) σαν μία ακολουθία από βήματα για να φτάσει από έναν κόμβο σε κάποιον άλλον, ή σε ένα σύνολο κόμβων.

Σε μία MXPath έκφραση [Appendix A΄], τα κριτήρια επιλογής που αφορούν το ρητό περιβάλλον εκφράζονται μέσω των *περιοριστών ρητού περιβάλλοντος* (explicit context qualifiers). Από την άλλη πλευρά, τα κριτήρια επιλογής που αφορούν την κληρονομούμενη κάλυψη περιβάλλοντος εκφράζονται μέσω του *περιοριστή κληρονομούμενης κάλυψης περιβάλλοντος* (inherited context coverage qualifier), ο οποίος τοποθετείται στην αρχή της MXPath έκφρασης.

4.3.1 Σύνταξη της MXPath

Μία *MXPath έκφραση* (MXPath expression) περιέχει έναν *περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος* (inherited context coverage qualifier ή icc qualifier), ο οποίος ακολουθείται από το *σώμα της MXPath έκφρασης* (MXPath expression body). Ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος τοποθετείται στην αρχή της έκφρασης και φιλτράρει του κόμβους που αποτελούν τα αποτελέσματα αυτής, σύμφωνα με την κληρονομούμενη κάλυψη περιβάλλοντος που τους χαρακτηρίζει. Η σύνταξη μίας MXPath έκφρασης φαίνεται παρακάτω:

```
[inherted_context_coverage_qualifier],MXPath_expression_body
```

Μία MXPath έκφραση, μπορεί να επιστρέφει είτε πολυδιάστατους κόμβους είτε κόμβους περιβάλλοντος. Παρακάτω, αναλύουμε κάθε τμήμα μίας MXPath έκφρασης ξεχωριστά.

Περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος

Η σύνταξη του περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος είναι η ακόλουθη:

```
icc() comparison_op context_specifier_expression
```

όπου `comparison_op` είναι ένας από τους *τελεστές σύγκρισης* (comparison operators) =, !=, <, >, <=, ή >=.

Είναι προφανές ότι για την κληρονομούμενη κάλυψη περιβάλλοντος των κόμβων στο μονοπάτι $r, n_1, \ldots, n_k$, από την ρίζα r του MXML δένδρου έως τον κόμβο n_k, ισχύει ότι $icc(n_k) \subseteq icc(n_{k-1}) \subseteq \ldots \subseteq icc(r)$. Έτσι το $icc(n_k)$ αντιπροσωπεύει τους κόσμους κάτω από τους οποίους το συνολικό μονοπάτι υφίσταται. Η διαδικασία `icc()` επιστρέφει το icc του τρέχοντος κόμβου, και συνεπώς, του υπό επεξεργασία τρέχοντος μονοπατιού του MXML κειμένου. Αυτό το icc στη συνέχεια συγκρίνεται έναντι του προσδιοριστή περιβάλλοντος (`context_specifier_expression`), σύμφωνα με τον τελεστή σύγκρισης (`comparison_op`). Ο τελεστής "=" ελέγχει ως προς την ισότητα, ο "<" ελέγχει ως προς την ύπαρξη γνήσιου υποσυνόλου, ">" ελέγχει ως προς την ύπαρξη γνήσιου υπερσυνόλου, κτλ. Σημειώνουμε ότι στην πραγματικότητα, τα σύνολα των κόσμων τα οποία αναπαριστώνται από τα ερμηνευτικά περιβάλλοντα είναι αυτά τα οποία συγκρίνονται. Στην περίπτωση που η σύγκριση επιστρέψει μη αληθή τιμή (false), το τρέχον μονοπάτι απορρίπτεται και δεν εξετάζεται περαιτέρω. Εάν ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος έχει παραληφθεί σε μία MXPath έκφραση, η *προκαθορισμένη τιμή* (default value) που υπονοείται είναι η: `icc()>= "-"`. Αυτός ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος επιστρέφει πάντα, ως αποτέλεσμα σύγκρισης, αληθή τιμή (true).

Σώμα MXPath έκφρασης

Το σώμα μίας MXPath έκφρασης αντιστοιχεί σε αυτό μίας κοινής XPath έκφρασης. Όπως στην XPath, έτσι και στην MXPath υπάρχουν δύο είδη σωμάτων έκφρασης, τα *απόλυτα* (absolute) και τα *σχετικά* (relative). Ένα απόλυτο σώμα μίας MXPath έκφρασης είναι στην ουσία ένα σχετικό, στο οποίο προηγείται το σύμβολο "/" που αναπαριστά τη ρίζα του MXML δένδρου. Ένα `MXPath_expression_body` αποτελείται από ένα ή περισσότερα *βήματα* (MXPath steps) τα οποία διαχωρίζονται μεταξύ τους με "/". Έτσι, η σύνταξη ενός σχετικού `MXPath_expression_body` είναι της μορφής:

```
MXPath_step_1/MXPath_step_2/.../MXPath_step_n
```

MXPath βήματα

Υπάρχουν δύο ειδών *MXPath βημάτων* (MXPath steps), τα *MXPath βήματα περιβάλλοντος* (Context MXPath steps) τα οποία επιστρέφουν κόμβους περιβάλλοντος, και *πολυδιάστατα MXPath βήματα* (Multidimensional MXPath steps) τα οποία επιστρέφουν πολυδιάστατους κόμβους.

Η σύνταξη ενός MXPath βήματος περιβάλλοντος είναι η εξής:

```
axis::node_test[pred_1][pred_2]...[pred_n]
```

Αντίστοιχα, η σύνταξη ενός πολυδιάστατου MXPath βήματος φαίνεται παρακάτω:

```
axis->node_test[pred_1][pred_2]...[pred_n]
```

Να σημειωθεί, ότι και οι δύο τύποι των MXPath βημάτων περιέχουν μία *ακμή* (axis), ένα *έλεγχο κόμβου* (node test) και μηδέν ή περισσότερα *κατηγορήματα* (predicates). Η μοναδική διαφορά έγκειται στο γεγονός ότι σε ένα MXPath βήμα περιβάλλοντος η ακμή ακολουθείται από το σύμβολο "::" το οποίο υποδηλώνει ότι το βήμα αντιστοιχεί σε κόμβο περιβάλλοντος, ενώ σε ένα πολυδιάστατο MXPath

βήμα η ακμή ακολουθείται από το σύμβολο "->" το οποίο υποδηλώνει πως το βήμα αντιστοιχεί σε πολυδιάστατο κόμβο.

MXPath κατηγορήματα

Στην MXPath, ένα *κατηγόρημα* (predicate) αποτελείται από μία έκφραση η οποία ονομάζεται *MXPath έκφραση κατηγορήματος* (MXPath predicate expression), που εσωκλείεται σε αγκύλες. Ένα κατηγόρημα επιτυγχάνει το φιλτράρισμα μίας ακολουθία αποτελεσμάτων, διατηρώντας κάποια από αυτά και απορρίπτοντας κάποια άλλα. Πολλαπλά κατηγορήματα μπορούν να χρησιμοποιηθούν σε μία MXPath έκφραση. Στην περίπτωση πολλαπλών παρακείμενων κατηγορημάτων, τα κατηγορήματα αυτά εφαρμόζονται από τα αριστερά προς τα δεξιά, ενώ τα αποτελέσματα της εφαρμογής του κάθε κατηγορήματος αποτελούν τα εισαγωγικά δεδομένα για το επόμενο προς εφαρμογή κατηγόρημα. Για κάθε στοιχείο της ακολουθίας των εισαγωγικών δεδομένων, η έκφραση κατηγορήματος εφαρμόζεται, και μία τιμή (τιμή αληθείας) που υποδηλώνει το κατά πόσο η έκφραση αληθεύει επιστρέφεται. Τα στοιχεία εκείνα για τα οποία η τιμή αληθείας ενός κατηγορήματος είναι θετική (true) διατηρούνται, ενώ εκείνα για τα οποία η εφαρμογή ενός κατηγορήματος παράγει μη αληθή τιμή (false) απορρίπτονται.

Οι τελεστές που χρησιμοποιούνται στα MXPath κατηγορήματα (λογικοί τελεστές, τελεστές σύγκρισης κτλ.) είναι οι ίδιοι που χρησιμοποιούνται και στη συμβατική XPath. Ωστόσο, τα MXPath κατηγορήματα μπορεί επίσης να περιέχουν σώματα MXPath εκφράσεων με τον ίδιο τρόπο όπως οι XPath εκφράσεις χρησιμοποιούνται στα κατηγορήματα της απλής XPath. Εκτός από αυτές τις συντακτικές δομές, και οι περιοριστές ρητού περιβάλλοντος (ec qualifiers) χρησιμοποιούνται στα MXPath κατηγορήματα. Ένας περιοριστής ρητού περιβάλλοντος μπορεί να εφαρμοστεί για κάθε βήμα μίας MXPath έκφρασης και να φιλτράρει τους κόμβους που προκύπτουν από το αντίστοιχο βήμα, σύμφωνα με το ρητό περιβάλλον των κόμβων αυτών. Οι περιοριστές ρητού περιβάλλοντος έχουν την παρακάτω μορφή:

```
ec() comparison_op context_specifier_expression
```

Η διαδικασία `ec()` επιστρέφει το ρητό περιβάλλον του τρέχοντος κόμβου. Ση-

μειώνεται ότι τα κατηγορήματα που αντιστοιχούν σε κάποιο MXPath βήμα περιβάλλοντος εφαρμόζονται στους κόμβους περιβάλλοντος που επιστρέφονται από τον υπολογισμό του συγκεκριμένου βήματος. Κατά τον ίδιο τρόπο, εάν ένα MXPath βήμα είναι πολυδιάστατο MXPath βήμα, τα κατηγορήματα εφαρμόζονται στους προκύπτοντες από το βήμα πολυδιάστατους κόμβους.

4.3.2 MXPath παραδείγματα

Σε αυτή την ενότητα παρουσιάζουμε μερικά MXPath παραδείγματα και εξηγείται ο τρόπος με τον οποίον εκτελούνται με βάση την γραφική αναπαράσταση του Σχήματος 2.1.

Παράδειγμα 4.2 *Ανάκτηση των κόμβων περιβάλλοντος με βάση την κληρονομούμενη κάλυψη περιβάλλοντος που τους χαρακτηρίζει.*

Ερώτημα*: Ποια είναι η επιτάχυνση ενός αυτοκινήτου που παράγεται στην Ιαπωνία (Japan) και πωλείται στις Ηνωμένες Πολιτείες της Αμερικής (USA);*

MXPath*:* `[icc()="factory=Japan,market=USA"],`
`/child::car/child::performance/child::acceleration`

Στο παράδειγμα αυτό, `[icc()="factory=Japan,market=USA"]` *είναι ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος της MXPath έκφρασης. Η έκφραση που ακολουθεί είναι το σώμα της MXPath έκφρασης το οποίο, για αυτό το παράδειγμα, συντακτικά είναι όμοιο με αυτό μιας συμβατικής XPath έκφρασης. Ωστόσο, υπάρχει διαφορά, αφού στην MXPath ένα βήμα περιγράφει μονοπάτια που περιλαμβάνουν δύο MXML κόμβους: έναν πολυδιάστατο κόμβο ακολουθούμενο από τον αντίστοιχο κόμβο περιβάλλοντος. Υπενθυμίζουμε ότι στην MXML ένα στοιχείο περιβάλλοντος έχει το ίδιο όνομα στοιχείου όπως τα αντίστοιχο πολυδιάστατο στοιχείο.*

Ας εξετάσουμε για παράδειγμα το βήμα `child::performance` *της παραπάνω MXPath έκφρασης, υποθέτοντας ότι ο τρέχον (current) κόμβος με βάση τον οποίο θα γίνει η εκτέλεση του βήματος αυτού είναι ο κόμβος 1 (έχει προκύψει σαν αποτέλεσμα της εκτέλεση του προηγούμενου βήματος). Αυτό το βήμα, ψάχνει για κόμβους που είναι παιδιά του κόμβου 1, περνάει από τον πολυδιάστατο κόμβο με όνομα "*`performance`*" και με ID 36 και επιστρέφει τους κόμβους περιβάλλοντος με όνομα "*`performance`*" και με IDs 37 και 46. Ομοίως, το MXPath βήμα* `child::acceleration`

επιστρέφει κατά την εκτέλεσή του τους κόμβους περιβάλλοντος με όνομα "`acceleration`*". Πιο συγκεκριμένα, όταν το βήμα εκτελείται θεωρώντας σαν τρέχον κόμβο τον κόμβο με ID 37, λαμβάνονται ως αποτέλεσμα οι κόμβοι με IDs 42 και 44. Όταν το ίδιο βήμα εκτελείται θεωρώντας σαν τρέχον κόμβο τον κόμβο με ID 46, έχουμε ως αποτέλεσμα τον κόμβο με ID 48. Στη συνέχεια, ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος εφαρμόζεται στο αποτέλεσμα. Καθώς η κληρονομούμενη κάλυψη περιβάλλοντος των αποτελεσμάτων πρέπει να είναι ίσο με το ερμηνευτικό περιβάλλον* `[factory=Japan,market=USA]`*, οι κόμβοι με IDs 42 και 48 αποκλείονται και το τελικό αποτέλεσμα περιλαμβάνει τον κόμβο με ID 44.*

Παρακάτω παραθέτουμε ενδεικτικά και την συνεπτυγμένη μορφή της παραπάνω MXPath έκφρασης:

```
[icc()="factory=Japan,market=USA"],
    /car/performance/acceleration
```

Παράδειγμα 4.3 *Ανάκτηση πολυδιάστατων κόμβων με βάση την κληρονομούμενη κάλυψη περιβάλλοντος που τους χαρακτηρίζει.*

Query*: Ποια είναι η ιπποδύναμη (power) των αυτοκινήτων που παράγονται στην Ιταλία (Italy) και πωλούνται είτε στις Ηνωμένες Πολιτείες της Αμερικής (USA) είτε στην Ευρώπη (Europe);*

MXPath*:*

```
[icc()="factory=Italy, market in {USA, Europe}"],
     /child::car/child::engine/child−>power
```

Στην περίπτωση αυτή, ακολουθούμε τους ίδιου κανόνες πλοήγησης όπως στο Παράδειγμα 4.2, όμως τώρα το MXPath βήμα `child−>power` *είναι πολυδιάστατο με αποτέλεσμα να επιστρέφει τους πολυδιάστατους κόμβους με όνομα "*`power`*" και IDs 22 και 31. Στη συνέχεια, λόγω της εφαρμογής του περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος* `[icc()="factory=Italy, market in {USA, Europe}"]`*, ο κόμβος 22 αποκλείεται και λαμβάνουμε ως τελικό αποτέλεσμα τον κόμβο 31.*

Παράδειγμα 4.4 *Ανάκτηση κόμβων περιβάλλοντος με βάση το ρητό περιβάλλον που τους χαρακτηρίζει.*

Query*: Ποια είναι η επιτάχυνση των αυτοκινήτων που πωλούνται στην Ευρώπη (Europe);*

***MXPath**:*

```
[icc()>="-"], /child::car/child::performance
  /child::acceleration[ec()>="market=Europe"]
```

Στο παράδειγμα αυτό, ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος προσδιορίζει ότι η κληρονομούμενης κάλυψης περιβάλλοντος των αποτελεσμάτων πρέπει να είναι, ως προς το ερμηνευτικό περιβάλλον, υπερσύνολο του κενού ερμηνευτικού περιβάλλοντος `[-]`. Όμως, εφόσον κάτι τέτοιο είναι πάντοτε αληθές, αυτός ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος μπορεί να παραληφθεί. Το κατηγόρημα `[ec()>="market=Europe"]` είναι ένας περιοριστής ρητού περιβάλλοντος που υποδηλώνει ότι το σύνολο των κόσμων που εκφράζεται από το ρητό περιβάλλον των κόμβων περιβάλλοντος με όνομα "`acceleration`" πρέπει να είναι υπερσύνολο του συνόλου των κόσμων που εκφράζονται από τον προσδιοριστή περιβάλλοντος `[market=Europe]`. Έτσι, εάν και το αποτέλεσμα του υπολογισμού του βήματος `child::acceleration` περιλαμβάνει τους κόμβους 42, 44 και 48, εντούτοις το τελικό αποτέλεσμα μετά την εφαρμογή του αντίστοιχου κατηγορήματος, περιέχει μόνο τους κόμβους περιβάλλοντος 42 και 48.

Παράδειγμα 4.5 *Ανάκτηση πολυδιάστατων κόμβων με βάση το ρητό περιβάλλον που τους χαρακτηρίζει.*

***Query**: Ποια είναι η ιπποδύναμη ενός αυτοκινήτου που παράγεται στην Ιαπωνία (Japan);*

***MXPath**:*

```
/child::car/child::engine[ec()="factory=Japan"]
 /child− >power
```

Όπως και στο Παράδειγμα 4.3, αυτό το ερώτημα επιστρέφει πολυδιάστατους κόμβους με όνομα "`power`". Εδώ, εξαιτίας του περιοριστής ρητού περιβάλλοντος στο βήμα `child::engine[ec()="factory=Japan"]`, το βήμα αυτό επιστρέφει τον κόμβο περιβάλλοντος με ID 18. Έτσι, το τελικό αποτέλεσμα περιλαμβάνει τον κόμβο 22.

Παράδειγμα 4.6 *Ανάκτηση κόμβων τιμής.*

***Query**: Από τα αυτοκίνητα που παράγονται στην Ιταλία (Italy), ποια είναι η μέγιστη ταχύτητα (top speed) των αυτοκινήτων που πωλούνται είτε στην Ευρώπη (Europe) είτε στις Ηνωμένες Πολιτείες της Αμερικής (USA);*

***MXPath**: [icc()<="market in {Europe,USA}"],*
/child::car/child::performance
[ec()="factory=Italy"]/child::top_speed

Στο παράδειγμα αυτό χρησιμοποιούνται και οι δύο τύποι των περιοριστών. Σύμφωνα με τους κανόνες πλοήγησης που περιγράψαμε παραπάνω, λαμβάνουμε ως αποτέλεσμα του ερωτήματος τους κόμβους με IDs 51 και 53.

Παράδειγμα 4.7 *Χρήση μίας MXPath έκφρασης ως κατηγόρημα.*

***Query**: Ποια είναι η μέγιστη ταχύτητα των αυτοκινήτων που είναι διαθέσιμα στην αγορά (market) των Ηνωμένων Πολιτειών της Αμερικής (USA) και των οποίων η επιτάχυνση (acceleration) είναι "0-100 in 5 sec";*

***MXPath**: [icc()>="-"],*
/child::car/child::performance[child::acceleration
[ec()<="market=USA"] ="0-100 in 5 sec"]
/child::top_speed

Στην περίπτωση αυτή, το σώμα μίας ολόκληρης MXPath έκφρασης περιέχεται στο κατηγόρημα του δεύτερου MXPath βήματος. Το κατηγόρημα αυτό φαίνεται παρακάτω:

[child::acceleration[ec()<="market=USA"]="0-100 in 5 sec"]

και είναι υπεύθυνο να διατηρήσει τους κόμβους (εάν υπάρχουν τέτοιοι κόμβοι) με όνομα στοιχείου `performance`, οι οποίοι έχουν ως παιδί κάποιον κόμβο στοιχείο με όνομα `acceleration`, του οποίου η τιμή για την αγορά (market) των Ηνωμένων Πολιτειών της Αμερικής (USA) είναι "0-100 in 5 sec". Όπως μπορούμε να δούμε, η εκτέλεση του παραπάνω ερωτήματος επιστρέφει τον κόμβο με ID 39.

4.4 Σημασιολογική ερμηνεία της MXPath

Σε αυτή την ενότητα, γίνεται αναφορά σε θέματα σημασιολογίας της MXPath. Πιο συγκεκριμένα, με βάση την έννοια της *αναγωγής* (reduction) για τα MXML κείμενα, όπως αναφέρεται στο [GSK01a], παρουσιάζεται το πως η σημασιολογία (μέρος αυτής) της MXPath μπορεί να εκφραστεί σύμφωνα με την σημασιολογία της XPath.

4.4.1 Αναγωγή της MXML στην XML

Αναγωγή (reduction) είναι μία διαδικασία σύμφωνα με την οποία, δοθέντος ενός κόσμου w και ενός MXML κειμένου (MXML δένδρου) G, παράγεται ένα συμβατικό XML κείμενο (XML δένδρο) $G' = \mathcal{R}(w, G)$ το οποίο είναι το ισχύον *στιγμιότυπο* (instance) του G κάτω από τον κόσμο w.

Το νόημα πίσω από την έννοια της αναγωγής είναι ότι, δοθέντος ενός κόσμου w, μπορούμε να εξειδικεύσουμε ένα MXML κείμενο G εξαλείφοντας τα τμήματα εκείνα του κειμένου (υποδένδρα) τα οποία δεν έχουν ισχύ κάτω από τον κόσμο w (με άλλα λόγια, εξαλείφοντας τα τμήματα εκείνα του κειμένου για τα οποία ο κόσμος w δεν ανήκει στο σύνολο των κόσμων που προσδιορίζεται από την αντίστοιχη κληρονομούμενη κάλυψη περιβάλλοντος των τμημάτων αυτών), αποκτώντας με αυτόν τον τρόπο ένα συμβατικό XML κείμενο G' το οποίο έχει ισχύ κάτω από τον κόσμο w. Σύμφωνα λοιπόν με τα παραπάνω, η διαδικασία της αναγωγής αποτελείται από τα παρακάτω βήματα:

1. Εξάλειψη όλων των υποδένδρων του G για τα οποία ισχύει ότι ο κόσμος w δεν ανήκει στους κόσμους που ορίζονται από την κληρονομούμενη κάλυψη περιβάλλοντος των ριζών των υποδένδρων αυτών.

2. Εξάλειψη κάθε ακμής στοιχείου περιβάλλοντος (αντίστοιχα ακμής γνωρίσματος περιβάλλοντος) (p, C, q) του δένδρου G_1, που προκύπτει από το G του βήματος 1, ως εξής: Έστω (s, p) η ακμή στοιχείου (αντίστοιχα ακμή γνωρίσματος) που οδηγεί στον κόμβο p. Τότε:

   - Προστίθεται μία νέα ακμή στοιχείου (αντίστοιχα ακμή γνωρίσματος) (s, q), και

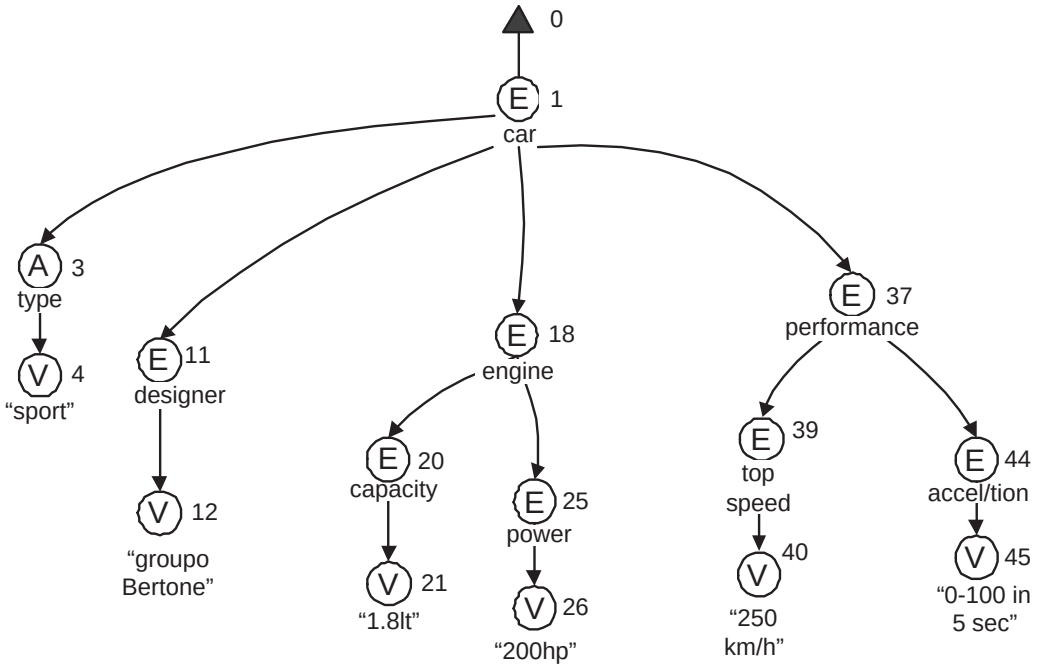
   - αφαιρούνται οι ακμές (p, C, q) και (s, p), καθώς και ο κόμβος p.

Το XML κείμενο (XML δένδρο) G', που προκύπτει μετά την εκτέλεση του βήματος 2, είναι το αποτέλεσμα της διαδικασίας την αναγωγής που εφαρμόζεται στο MXML κείμενο G σε συνάρτηση με τον κόσμο w.

Παράδειγμα 4.8 *Ας θεωρήσουμε το MXML κείμενο του παραδείγματος 2.1 και τον κόσμο* $w = \{(factory, Japan), (market, USA)\}$. *Εφαρμόζοντας την διαδικασία της αναγωγής σε αυτό το MXML κείμενο, προκύπτει το παρακάτω XML κείμενο:*

```
<car type="sport">
    <designer>groupo Bertone</designer>
    <engine>
       <capacity>1.8lt</capacity>
       <power>200hp</power>
    </engine>
    <performance>
       <top_speed>250km/h</top_speed>
       <acceleration>0-100 in 5sec</acceleration>
    </performance>
</car>
```

Το XML δένδρο του κειμένου φαίνεται στο σχήμα 4.1.



Σχήμα 4.1: Προκύπτον XML δένδρο από την εφαρμογή της αναγωγής στο MXML δένδρο του σχήματος 2.1.

Να σημειωθεί ότι εφαρμόζοντας τη διαδικασία της αναγωγής, κάθε MXML κείμενο μπορεί να αναχθεί σε ένα αριθμό από XML κείμενα, με κάθε ένα από τα

οποία να ισχύει κάτω από έναν συγκεκριμένο κόσμο. Μπορούμε έτσι να θεωρήσουμε ένα MXML κείμενο σαν μία συνεπτυγμένη αναπαράσταση ενός συνόλου από XML κείμενα. Ωστόσο, ένα MXML κείμενο είναι κάτι περισσότερο από απλό σύνολο XML κειμένων, αφού εμπεριέχει τον συσχετισμό των διαφόρων τμημάτων των κειμένων αυτών (όντας οι διαφορετικές εκδόσεις του ίδιου αντικειμένου).

4.4.2 Σημασιολογική σχέση μεταξύ MXPath και XPath

Σε αυτή την ενότητα διερευνάται η σχέση μεταξύ της MXPath και της XPath, ενώ υποδεικνύεται ο τρόπος σύμφωνα με τον οποίο μπορεί να επιτευχθεί μία σημασιολογική συσχέτιση μεταξύ της MXPath και XPath. Το παράδειγμα 4.9 δίνει μία αίσθηση σχετικά με το περιεχόμενο της υπόλοιπης ενότητας:

Παράδειγμα 4.9 *Στο παράδειγμα 4.2, είδαμε το παρακάτω MXPath ερώτημα:*

```
[icc()="factory=Japan,market=USA"],
/child::car/child::performance/child::acceleration
```

και δείξαμε ότι το αποτέλεσμα της εκτέλεσής του είναι ο κόμβος με ID 44. Σημειώνουμε ότι οι κόμβοι οι οποίοι προκύπτουν από την εκτέλεση του σώματος της MXPath έκφρασης του ερωτήματος (κόμβοι με IDs 42, 44 και 46) φιλτράρονται με τη χρήση του περιοριστή κληρονομούμενης κάλυψης περιβάλλοντος με σκοπό να αποκτηθεί το τελικό αποτέλεσμα. Παρατηρώντας τον περιοριστή κληρονομούμενης κάλυψης περιβάλλοντος, βλέπουμε ότι οι κόμβοι οι οποίοι παραμένουν στο τελικό αποτέλεσμα είναι οι κόμβοι των οποίων η κληρονομούμενη κάλυψης περιβάλλοντος αποτελείται από τον κόσμο $w = \{$`(factory,Japan), (market,USA)`$\}$*. Επιπλέον, παρατηρούμε ότι στο παράδειγμα 4.8, εφαρμόσαμε την διαδικασία της αναγωγής στο MXML κείμενο σε συνδυασμό με τον ίδιο κόσμο* w *και πήραμε ως αποτέλεσμα το XML κείμενο του οποίου η γραφική αναπαράσταση φαίνεται στο σχήμα 4.1). Είναι ενδιαφέρουσα η παρατήρηση ότι μεταχειριζόμενοι το σώμα μίας MXPath έκφρασης κατά τον ίδιο τρόπο όπως σε μία συμβατική XPath έκφραση* [1] *και εφαρμόζοντας αυτή την έκφραση στο XML δένδρο του σχήματος 4.1, παίρνουμε το ίδιο αποτέλεσμα (δηλαδή τον κόμβο 44) όπως και με το αρχικό MXPath ερώτημα.*

[1] Στο σημείο αυτό το σώμα της MXPath έκφρασης είναι μία συμβατική XPath έκφραση. Ωστόσο, αυτό δεν αληθεύει σε όλες τις περιπτώσεις.

Με βάση το παραπάνω παράδειγμα, ένα ενδιαφέρον ερώτημα που προκύπτει είναι το παρακάτω:

Μπορούμε να χρησιμοποιήσουμε συμβατικά XML κείμενα που έχουν προκύψει εφαρμόζοντας τη διαδικασία της αναγωγής σε ένα MXML κείενο G μαζί με ένα ερώτημα της συμβατικής XPath που έχει προέλθει από το αρχικό MXPath ερώτημα E, προκειμένου να ορίσουμε το νόημα της εκτέλεσης του ερωτήματος E πάνω στο κείμενο G;

Το ακόλουθο Λήμμα 4.4.2 απαντά στο ερώτημα αυτό για μία υποκατηγορία MXPath ερωτημάτων. Πιο συγκεκριμένα, το Λήμμα 4.4.2 καθορίζει μία σημασιολογική σχέση για μία υποκατηγορία MXPath ερωτημάτων, η οποία περιλαμβάνει ερωτήματα των οποίων το σώμα μίας MXPath έκφρασης δεν περιέχει ούτε περιοριστές ρητού περιβάλλοντος ούτε πολυδιάστατα MXpath βήματα.

Στο Λήμμα 4.4.2 χρησιμοποιούμε τους ακόλουθους συμβολισμούς: Έστω E μία MXPath έκφραση (αντίστοιχα XPath έκφραση) και G ένα MXML κείμενο (αντίστοιχα XML κείμενο). Τότε με $E(G)$ αναπαριστούμε το σύνολο των IDs των κόμβων που επιστρέφει η εκτέλεση του E πάνω στο G. Με $\mathcal{U}_G$ συμβολίζεται το σύνολο όλων των πιθανών κόσμων που σχετίζονται με το G. Επιπλέον, με $\mathcal{W}(C)$ συμβολίζεται το σύνολο των κόσμων που ορίζονται από τον προσδιοριστή περιβάλλοντος C. Τέλος, υπενθυμίζουμε ότι με $\mathcal{R}(w, G)$ αναπαριστούμε το XML κείμενο που παράγεται κατά την εφαρμογή της διαδικασίας της αναγωγής πάνω στο G σε συνδυασμό με έναν κόσμο $w \in \mathcal{U}_G$.

Λήμμα 4.4.1. Έστω G ένα MXML κείμενο και $E = (I, P)$ ένα MXPath ερώτημα, όπου I είναι ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος και P το σώμα της MXPath έκφρασης του ερωτήματος. Υποθέτουμε ότι το P αποτελείται μόνο από MXPath βήματα περιβάλλοντος και δεν περιέχει περιοριστές ρητού περιβάλλοντος [2]. Υποθέτουμε επίσης ότι το C είναι ένας προσδιοριστής περιβάλλοντος τέτοιος ώστε $C \neq$ `"-"`. Τότε ισχύουν τα παρακάτω:

- Εάν I είναι της μορφής $[icc() >$ `"-"`$]$, τότε:
  $E(G) = \bigcup_{G' \in S_{\mathcal{U}}}(P(G'))$,
  όπου $S_{\mathcal{U}} = \{\mathcal{R}(w, G) | w \in \mathcal{U}_G\}$.

[2]Στην περίπτωση αυτή το P είναι μία (συμβατική) XPath έκφραση.

- Εάν I είναι της μορφής $[icc() >= C]$, όπου C είναι ένας προσδιοριστής περιβάλλοντος, τότε:

  $E(G) = \bigcap_{G' \in S_C}(P(G'))$,

  όπου $S_C = \{\mathcal{R}(w, G) | w \in \mathcal{W}(C)\}$.

- Εάν I είναι της μορφής $[icc() = C]$, όπου C είναι ένας προσδιοριστής περιβάλλοντος, τότε:

  $E(G) = \bigcap_{G' \in S_C}(P(G')) - \bigcup_{G'' \in S_{\overline{C}}}(P(G''))$,

  όπου $S_C = \{\mathcal{R}(w, G) | w \in \mathcal{W}(C)\}$ και

  $S_{\overline{C}} = \{\mathcal{R}(w, G) | w \in \mathcal{U}_G - \mathcal{W}(C)\}$.

- Εάν I είναι της μορφής $[icc() <= C]$, όπου C είναι ένας προσδιοριστής περιβάλλοντος, τότε:

  $E(G) = \bigcup_{G' \in S_C}(P(G')) - \bigcup_{G'' \in S_{\overline{C}}}(P(G''))$,

  όπου $S_C = \{\mathcal{R}(w, G) | w \in \mathcal{W}(C)\}$ και

  $S_{\overline{C}} = \{\mathcal{R}(w, G) | w \in \mathcal{U}_G - \mathcal{W}(C)\}$.

- Εάν I είναι της μορφής $[icc()! = C]$, όπου C είναι ένας προσδιοριστής περιβάλλοντος, τότε:

  $E(G) = \bigcup_{G' \in S_{\mathcal{U}}}(P(G')) -$

  $(\bigcap_{G'' \in S_C}(P(G'')) - \bigcup_{G''' \in S_{\overline{C}}}(P(G''')))$,

  όπου $S_{\mathcal{U}} = \{\mathcal{R}(w, G) | w \in \mathcal{U}_G\}$ and $S_C = \{\mathcal{R}(w, G) | w \in \mathcal{W}(C)\}$ και $S_{\overline{C}} = \{\mathcal{R}(w, G) | w \in \mathcal{U}_G - \mathcal{W}(C)\}$.

- Εάν I είναι της μορφής $[icc() > C]$, όπου C είναι ένας προσδιοριστής περιβάλλοντος, τότε:

  $E(G) = (\bigcap_{G' \in S_C}(P(G'))) \cap (\bigcup_{G'' \in S_{\overline{C}}}(P(G'')))$,

  όπου $S_C = \{\mathcal{R}(w, G) | w \in \mathcal{W}(C)\}$ και

  $S_{\overline{C}} = \{\mathcal{R}(w, G) | w \in \mathcal{U}_G - \mathcal{W}(C)\}$.

- Εάν I είναι της μορφής $[icc() < C]$, όπου C είναι ένας προσδιοριστής περιβάλλοντος, τότε:
  $E(G) = ((\bigcup_{G' \in S_C}(P(G')) - \bigcup_{G'' \in S_{\overline{C}}}(P(G'')))-$
  $(\bigcap_{G' \in S_C}(P(G')) - \bigcup_{G'' \in S_{\overline{C}}}(P(G'')))$.
  όπου $S_C = \{\mathcal{R}(w, G) | w \in \mathcal{W}(C)\}$ και
  $S_{\overline{C}} = \{\mathcal{R}(w, G) | w \in \mathcal{U}_G - \mathcal{W}(C)\}$.

Το Λήμμα 4.4.2 υπονοεί ότι μπορούμε να πάρουμε σαν αποτελέσματα τους ίδιους κόμβους, εκτελώντας τόσο XPath ερωτήματα σε XML δεδομένα όσο και αντίστοιχα MXPath ερωτήματα σε MXML δεδομένα. Να σημειωθεί ωστόσο, ότι αν και οι κόμβοι αυτοί, και στις δύο περιπτώσεις, είναι όμοιοι (έχουν τα ίδια IDs), αναπαριστούν διαφορετικά υποδένδρα ανάλογα με το XML κείμενο από το οποίο προέρχονται. Είναι φανερό ότι όταν ένα MXML ερώτημα, εκτελούμενο πάνω σε ένα MXML έγγραφο G, επιστρέφει έναν κόμβο περιβάλλοντος N, τότε τα στιγμιότυπα του ίδιου κόμβου N που επιστρέφονται κατά την εφαρμογή του σώματος της MXPath έκφρασης στις εκφάνσεις του G, όπως καθορίζει το παραπάνω λήμμα, έχουν ρίζα στα XML δένδρα που αποκτώνται κατά την εκτέλεση της διαδικασίας της αναγωγής στα πολυδιάστατα υποδένδρα του G που έχουν ρίζα στον κόμβο N. Με αυτή την έννοια, τόσο το MXML κείμενο όσο και η MXPath έκφραση μπορούν να θεωρηθούν σαν μία συμπαγής (compact) αναπαράσταση μίας ομάδας από XML κείμενα και μίας ομάδας από τα αντίστοιχα XPath ερωτήματα πάνω στα κείμενα αυτά. Επιπλέον, προκειμένου να απαντηθούν κάποια από αυτά τα XPath ερωτήματα, ορισμένα XML κείμενα θα πρέπει να εξεταστούν παράλληλα.

Σύμφωνα με το Λήμμα 4.4.2, ένα MXPath ερώτημα είναι αρκετό ως προς την εκφραστικότητά του, προκειμένου να προσδιορίσει κόμβους οι οποίοι αντιστοιχούν στον συνδυασμό των αποτελεσμάτων πολλών XPath εκδοχών που εφαρμόζονται σε σχετικά αποτελέσματα της MXML διαδικασίας αναγωγής. Ωστόσο, η MXPath περιλαμβάνει περισσότερα πράγματα από μία περισσότερο συμπαγή αναπαράσταση ερωτημάτων σε σχέση με την XPath. Μερικά χαρακτηριστικά της MXPath είναι εγγενώς πολυδιάστατα και μπορούν να ερμηνευθούν μόνο σε συνδυασμό με την MXML. Αυτά τα χαρακτηριστικά βασίζονται στο γεγονός ότι η

MXML ομαδοποιεί τις εκφάνσεις μίας οντότητας (κόμβοι περιβάλλοντος) κάτω από τον ίδιο πολυδιάστατο κόμβο. Χρησιμοποιώντας πολυδιάστατα MXPath βήματα (βήματα που περιέχουν το σύμβολο `->`) μπορούμε να εντοπίσουμε πολυδιάστατους κόμβους ενώ με τη χρήση των περιοριστών ρητού περιβάλλοντος μπορούμε να πλοηγηθούμε ανάμεσα στις εκφάνσεις τους. Ωστόσο, αυτού του είδους τα ερωτήματα είναι δύσκολο να εκφραστούν σε συνάρτηση με τα XML κείμενα που επιστρέφονται με τη διαδικασία της αναγωγής πάνω στο έγγραφο G.

4.5 Διαχείριση MXML κειμένων με τη χρήση της MXPath

Ο ορισμός της MXPath αποτελεί το πρώτο βήμα προς την κατεύθυνση της απόκτησης των κατάλληλων εργαλείων για την διαχείριση της MXML. Η αξιοποίηση της MXPath μπορεί να πραγματοποιηθεί σε τρεις βασικές κατευθύνσεις:

Α. Η MXPath είναι απαραίτητη για την ενημέρωση της MXML. Στο Κεφάλαιο 5 ([FGS07b]) ορίζεται ένας αριθμός από βασικές πράξεις ενημερώσεως της MXML. Κάθε μία από τις πράξεις αυτές χρησιμοποιεί MXPath εκφράσεις ως εισερχόμενα δεδομένα, προκειμένου να προσδιορίσει τα τμήματα του MXML δένδρου, τα οποία μεταβάλλονται μέσω της αντίστοιχης πράξης.

Β. Η MXPath μπορεί να αποτελέσει βασικό δομικό στοιχείο μιας γλώσσας επερωτήσεων για την MXML. Η XQuery [CON07b] αποτελεί μία γλώσσα διαχείρισης και υποβολή ερωτημάτων σε XML κείμενα. Ως γνωστόν, η XQuery χρησιμοποιεί XPath εκφράσεις για την πλοήγηση μεταξύ των στοιχείων σε ένα XML κείμενο. Ωστόσο, χρησιμοποιώντας επιπρόσθετα στοιχεία, όπως για παράδειγμα μεταβλητές, joins και δόμηση των αποτελεσμάτων, η γλώσσα αυτή εμπεριέχει λειτουργικά περισσότερες δυνατότητες. Τόσο η XQuery όσο και η XPath μοιράζονται το ίδιο μοντέλο δεδομένων, ενώ υποστηρίζουν τις ίδιες λειτουργίες και τελεστές. Μία επέκταση της XQuery (Πολυδιάστατη XQuery ή MXQuery) θα μπορούσε να εκφράσει πολυδιάστατα ερωτήματα (με βάση το ερμηνευτικό περιβάλλον) για MXML δεδομένα, χρησιμοποιώντας MXPath εκφράσεις.

Γ. Η MXPath μπορεί να χρησιμοποιηθεί για την αναμόρφωση της MXML. Η *Extensible Stylesheet Language* (XSLT) [CON99b], η οποία είναι μία γλώσσα αναμόρφωσης των XML κειμένων, χρησιμοποιεί την XPath για την επιλογή στοιχείων. Κατά τη διαδικασία της αναμόρφωσης, η XSLT χρησιμοποιεί τις XPath εκφράσεις για τον προσδιορισμό τμημάτων του XML κειμένου, τα οποία θα πρέπει να ταιριάζουν με ένα ή περισσότερα προκαθορισμένα *πρότυπα* (patterns). Όταν υπάρχει ταύτιση των παραπάνω τμημάτων, η XSLT αναμορφώνει τα τμήματα αυτά του XML κειμένου και παράγει το αναμορφωμένο κείμενο ως αποτέλεσμα. Οι XPath εκφράσεις χρησιμοποιούνται στην XSLT για διάφορους σκοπούς συμπεριλαμβανομένου (α) την επιλογή κόμβων προς επεξεργασία, (β) τον καθορισμό των συνθηκών για τον προσδιορισμό των διαφορετικών τρόπων επεξεργασίας ενός κόμβου, ή (γ) της παραγωγή κειμένου προκειμένου να εισαχθεί στο προκύπτον από την επεξεργασία δέντρο. Η XSLT θα μπορούσε να επεκταθεί μέσω της MXPath (προκειμένου να γίνει Πολυδιάστατη XSLT ή MXSLT) ώστε να έχει τη δυνατότητα να εκφράσει λειτουργίες αναμόρφωσης των MXML κειμένων.

4.6 Συμπεράσματα - Παρατηρήσεις

Όπως ισχύει για την XPath αναφορικά με την την απλή XML, έτσι και η MXPath αποτελεί ένα βασικό δομικό στοιχείο για τον σχηματισμό εκφράσεων που προσδιορίζουν τη θέση ενός MXML κόμβου σε ένα MXML δένδρο. Με τον τρόπο αυτό, δίνει τη δυνατότητα σχηματισμού MXML ερωτημάτων, ενώ (όπως περιγράφεται παρακάτω) είναι δυνατός ο προσδιορισμός σημείων μέσα στο MXML κείμενο όπου θα μπορούσαν να πραγματοποιηθούν πράξεις ενημερώσεως.

Τα πλεονεκτήματα της MXPath σε σχέση με την XPath, απορρέουν από τα βασικά χαρακτηριστικά της MXML. Έτσι, αν και ένα MXPath ερώτημα θα μπορούσε να εκφραστεί σαν ένα σύνολο από επιμέρους XPath ερωτήματα, εντούτοις κάτι τέτοιο θα απαιτούσε την ομαδοποίηση και συνδυασμό των αποτελεσμάτων (απλά XML κείμενα) προκειμένου τα αποτελέσματα αυτά να εκφραστούν με τον τρόπο, όπως θα τα εξέφραζε απευθείας ένα MXML κείμενο. Συνεπώς, εκτός από τη δυνατότητα να λαμβάνουμε MXML αποτελέσματα κατά την εκτέλεση των MXPath ερωτημάτων, η MXPath μας επιτρέπει την πλοήγηση στα επιπλέον χαρακτηριστικά της MXML (επιστροφή αποτελεσμάτων που περιλαμβάνουν πολυδιάστατους κόμβους κτλ.), πράγμα το οποίο δεν παρέχεται από την απλή XPath.

Κεφάλαιο 5

Ενημέρωση της Πολυδιάστατης XML

5.1 Εισαγωγή

Σε αντίθεση με την απλή XML, οι διαδικασίες που σχετίζονται με την *ενημέρωση* (updating) της MXML θα πρέπει να λαμβάνουν υπόψη τα επιπρόσθετα χαρακτηριστικά που προκύπτουν από την έννοια του ερμηνευτικού περιβάλλοντος μέσα στα MXML κείμενα. Με βάση αυτά τα χαρακτηριστικά, μελετήθηκε το πρόβλημα της ενημέρωσης της MXML σε δύο επίπεδα:

α) Σε επίπεδο MXML γραφικού μοντέλου αναπαράστασης (MXML δένδρο), δηλαδή με τρόπο ανεξάρτητο υλοποίησης.

β) Σε επίπεδο σχεσιακού σχήματος αποθήκευσης.

Σε αυτό το κεφάλαιο παρουσιάζονται έξι βασικές *πράξεις ενημέρωσης* (update operations), οι οποίες αντιπροσωπεύουν κάθε πιθανή μεταβολή των MXML δεδομένων. Οι πράξεις αυτές προσδιορίζονται με έναν τρόπο ανεξάρτητο υλοποίησης, ενώ εξηγείται η επίπτωσή τους σε κάποιο κείμενο μέσω σχετικών παραδειγμάτων. Επιπλέον, έχουν αναπτυχθεί κατάλληλοι αλγόριθμοι που μπορούν να χρησιμοποιηθούν για την υλοποίηση των παραπάνω πράξεων με τη χρήση της SQL και σύμφωνα με κάποια συγκεκριμένη μέθοδο αποθήκευσης της MXML σε μία σχεσιακή βάση.

5.2 MXML Πράξεις Ενημερώσεως - Γραφικό Μοντέλο Αναπαράστασης

Στην ενότητα αυτή ορίζουμε, σε επίπεδο MXML γραφικού μοντέλου αναπαράστασης, ένα σύνολο από βασικές πράξεις ενημερώσεως (*delete*, *insert*, *update_label*, *update_context*, *update_value* και *replace*). Οι πράξεις αυτές χρησιμοποιούνται για την μεταβολή MXML κειμένων (MXML δένδρων). Αυτές οι πράξεις μπορούν να συνδυαστούν προκειμένου να προκαλέσουν οποιαδήποτε τροποποίηση σε κάποιο κείμενο, ενώ όταν εφαρμόζονται σε ένα MXML κείμενο που βρίσκεται σε *κανονική μορφή* (well-formed ή canonical form) [Sta03] το κείμενο το οποίο παράγεται είναι και αυτό σε κανονική μορφή.

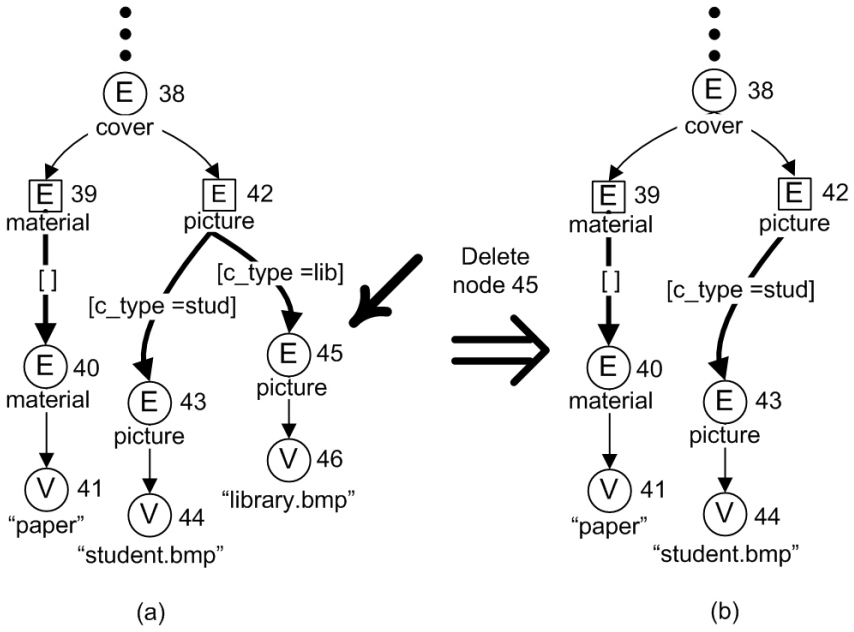
5.2.1 Διαγραφή υποδένδρων (delete)

Η πράξη *delete(P)* χρησιμοποιείται για τη διαγραφή υποδένδρων που έχουν ρίζες κόμβους (περιβάλλοντος ή πολυδιάστατους) οι οποίοι προσδιορίζονται από την MXPath έκφραση P. Παρακάτω φαίνεται ο αλγόριθμος βάσει του οποίου ορίζεται η εν λόγω πράξη:

*Αλγόριθμος **delete(P)**:*
*Είσοδος:**P**: Μία MXPath έκφραση (MXPath expression).*

1. Έστω N_P το σύνολο των κόμβων που επιστρέφεται κατά την εκτέλεση της P.
*2. **Για κάθε** $n \in N_P$ **επανέλαβε***
- Έστω p ο κόμβος γονέας του κόμβου n.
- Διέγραψε το υποδένδρο που έχει ρίζα τον n καθώς και την ακμή που οδηγεί στον n.
- Υπολόγισε ξανά τα $iccs$ κάθε κόμβου που βρίσκεται στο μονοπάτι που αρχίζει από τον κόμβο p και καταλήγει στη ρίζα του δένδρου.
- Διέγραψε το υποδένδρο που έχει ρίζα κάθε κόμβο m για τον οποίο ισχύει $icc(m) = [-]$ καθώς και τις ακμές που οδηγούν στον m.
Τέλος επανάληψης

Τέλος



Σχήμα 5.1: Διαγραφή υποδένδρου

Παράδειγμα 5.1 *Το δένδρο του σχήματος 5.1(b) προκύπτει από το δένδρο του σχήματος 5.1(a) (το οποίο είναι τμήμα του δένδρου που απεικονίζεται στο σχήμα 2.2), με την διαγραφή του υποδένδρου το οποίο έχει ρίζα τον κόμβο 45. Επαναπροσδιορίζοντας τις κληρονομούμενες καλύψεις περιβάλλοντος όλων των κόμβων από τον κόμβο 42 μέχρι την ρίζα του δένδρου, έχουμε:* `icc(42)=[ed=gr,c_type=stud]`, `icc(38)=[ed=gr]`, `icc(33)=[ed in {gr,en}]` *και* `icc(1)=[]`. *Σημειώνουμε ότι μία MXPath έκφραση P που θα μπορούσε να επιστρέψει τον κόμβο 45 κατά την εκτέλεσή της είναι η:*

```
/book/cover[ec()=''ed=gr'']/picture[ec()=''c_type=lib'']
```

Το τελευταίο τμήμα `[ec()= ''c_type=lib'']` *είναι ένα κατηγόρημα που αφορά τα ερμηνευτικά περιβάλλοντα των κόμβων περιβάλλοντος με όνομα* `picture`. *Σύμφωνα με το κατηγόρημα αυτό, οι κόμβοι που αποτελούν τα αποτελέσματα την MXPath έκφρασης θα πρέπει να έχουν ρητό περιβάλλον ίσο με* `[c_type=lib]`. *Έτσι τελικά επιστρέφονται οι* `''library''` `picture` *κόμβοι για τις ελληνικές εκδόσεις (*`greek edition`*). Εάν το αποτέλεσμα θέλαμε να είναι οι* `''library''`

`picture` κόμβοι για κάθε πιθανή έκδοση (`edition`), θα παραλείπαμε το κατηγόρημα `[ec()= ''ed=gr'']`, υπονοώντας το κατηγόρημα `[ec()>= ''-'']` στη θέση του. Σε αυτή την περίπτωση, παίρνουμε σαν αποτέλεσμα τους κόμβους 43 και 45. Διαγράφοντας και τους δύο κόμβους, η κληρονομούμενη κάλυψη περιβάλλοντος του κόμβου 42 γίνεται [-] και γιαυτό ο κόμβος 42 επίσης διαγράφεται.

5.2.2 Εισαγωγή υποδένδρων (insert)

Η εισαγωγή ενός δένδρου (κανονικοποιημένου) T σε συγκεκριμένα σημεία στο MXML δένδρο, τα οποία προσδιορίζονται από μία MXPath έκφραση P, γίνεται μέσω της πράξης *insert(P, T)*, όπως φαίνεται παρακάτω:

***insert(P, T)**:*
*Είσοδος:**P**: Μία MXPath έκφραση.*
__T__: Ένα κανονικοποιημένο MXML δένδρο.

1. Έστω N_P το σύνολο των κόμβων που επιστρέφεται από την εκτέλεση της έκφρασης P.
*2. **Για κάθε** $n \in N_P$ **επανέλαβε***
__Εάν__ n και $root(T)$ είναι πολυδιάστατοι κόμβοι
__και__ $label(n) = label(root(T))$ __τότε__
- Έστω C_n το σύνολο των προσδιοριστών περιβάλλοντος όλων των ακμών περιβάλλοντος που ξεκινούν από τον κόμβο n.
- Έστω C_T το σύνολο των προσδιοριστών περιβάλλοντος όλων των ακμών περιβάλλοντος που ξεκινούν από την ρίζα του δένδρου T.
- __Εάν__ $C_n \cup C_T$ είναι ένα context deterministic σύνολο από προσδιοριστές περιβάλλοντος __τότε__
- Συνέδεσε το T στον κόμβο n (ενοποιώντας τον n με την ρίζα του T).
- Επαναπροσδιόρισε τις κληρονομούμενες καλύψεις περιβάλλοντος όλων των προγόνων και απογόνων του κόμβου n στο προκύπτον δένδρο.
__Τέλος εάν__
__Αλλιώς εάν__ n και $root(T)$ είναι κόμβοι περιβάλλοντος
__και__ $label(n) = label(root(T))$ __τότε__

- Συνέδεσε το T στον κόμβο n (ενοποιώντας τον n με την ρίζα του T).

- Επαναπροσδιόρισε τις κληρονομούμενες καλύψεις περιβάλλοντος του κόμβου n και όλων των προγόνων και απογόνων του στο προκύπτον δένδρο.

Τέλος εάν

3. Διέγραψε τα υποδένδρα που έχουν ρίζα κάθε κόμβο m, για τον οποίο ισχύει $icc(m) = [-]$, όπως επίσης και τις ακμές που οδηγούν σε κάθε m.

Τέλος επανάληψης

Τέλος

Να σημειώσουμε στο σημείο αυτό ότι:

α) Η πράξη της εισαγωγής μπορεί να εφαρμοστεί μόνο σε κόμβους του ίδιου είδους και *ετικέτας* (label) με την ρίζα του υποδένδρου T.

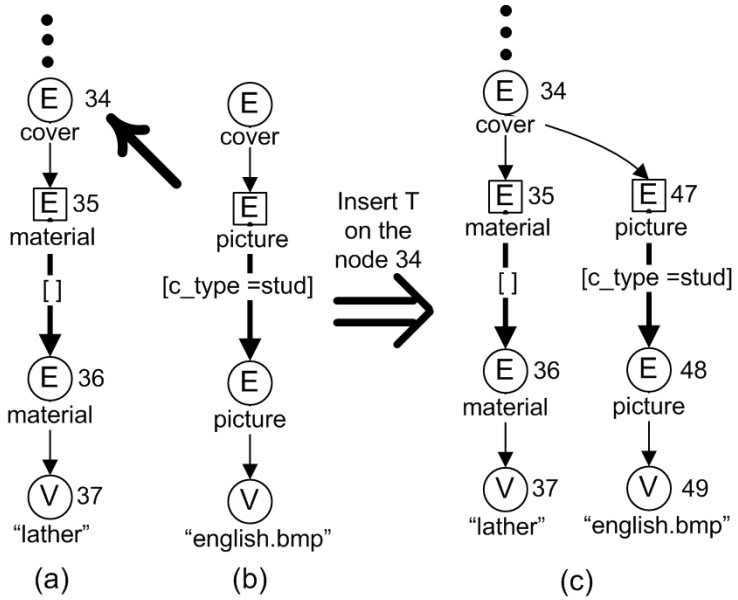
β) Εάν η ρίζα του T είναι πολυδιάστατος κόμβος, ο αλγόριθμος εξασφαλίζει ότι το δένδρο που προκύπτει είναι *ακέραιο ως προς το ερμηνευτικό περιβάλλον* (context deterministic).

γ) Εάν η έκφραση P επιστρέφει περισσότερους από έναν κόμβους, η εισαγωγή του T εφαρμόζεται για κάθε κόμβο ξεχωριστά. Η σειρά εφαρμογής δεν έχει σημασία.

Παράδειγμα 5.2 *Το δένδρο του σχήματος 5.2(c) προκύπτει από τη εισαγωγή του δένδρου T, το οποίο φαίνεται στο σχήμα 5.2(b), στη θέση του κόμβου 34 του δένδρου που φαίνεται στο σχήμα 5.2(a) (το οποίο είναι τμήμα του δένδρου που απεικονίζεται στο σχήμα 2.2).*

Με τον επαναπροσδιορισμό των κληρονομούμενων καλύψεων περιβάλλοντος των κόμβων έχουμε: `icc(49)= icc(48)= icc(47) =[ed=en,c_type=stud]` *και* `icc(34)=[ed=en]`*. Σημειώνουμε ότι η P είναι η MXPath έκφραση:*

```
/book/cover[ec()=''ed=en'']
```



Σχήμα 5.2: Εισαγωγή υποδένδρου

5.2.3 Ενημέρωση κόμβων

Η ενημέρωση (updating) κόμβων ενός MXML δένδρου μπορεί να πραγματοποιηθεί με τρεις τρόπους. Είτε ενημερώνοντας την ετικέτα (label) (εάν ο κόμβος είναι πολυδιάστατος), είτε ενημερώνοντας το ερμηνευτικό περιβάλλον, είτε ενημερώνοντας την τιμή του κόμβου αυτού. Σε ότι αφορά την ενημέρωση του ερμηνευτικό περιβάλλον ενός κόμβου, θα πρέπει να εξετάζονται οι απαραίτητες συνθήκες προκειμένου να διαφυλάσσεται η ακεραιότητα σε ότι αφορά το ερμηνευτικό περιβάλλον (context determinism) του δένδρου που θα προκύψει μετά την ενημέρωση.

Ενημέρωση ετικέτας (updating label)

Η πράξη *update_label(P, L)* χρησιμοποιείται για την ενημέρωση της ετικέτας ενός ή περισσοτέρων κόμβων ενός MXML δένδρου και αλγοριθμικά ορίζεται ως εξής:

update_label(P, L):

*Είσοδος:**P**: Μία MXPath έκφραση.*
__L__: Μία ετικέτα κόμβου (node label).

1. Έστω N_P το σύνολο των κόμβων που που επιστρέφεται από την εκτέλεση της έκφρασης P.
*2. **Για κάθε** πολυδιάστατο κόμβο $n \in N_P$ **επανέλαβε***
- Αντικατέστησε την ετικέτα του κόμβου n με την ετικέτα L.
- Αντικατέστησε με την ετικέτα L την ετικέτα του κάθε παιδιού του κόμβου n, που είναι κόμβος περιβάλλοντος.
Τέλος επανάληψης

Τέλος

Να σημειώσουμε ότι η πράξη *update_label* μπορεί να εφαρμοστεί μόνο σε πολυδιάστατους κόμβους. Αυτό δεν αποτελεί περιορισμό όταν το MXML δένδρο βρίσκεται σε κανονική μορφή. Εξάλλου, με αυτόν τον τρόπο αποφεύγονται καταστάσεις ασυνέπειας κατά τις οποίες ένας κόμβος στοιχείου περιβάλλοντος που είναι παιδί μπορεί να έχει διαφορετική ετικέτα από τον πατρικό του πολυδιάστατο κόμβο στοιχείου.

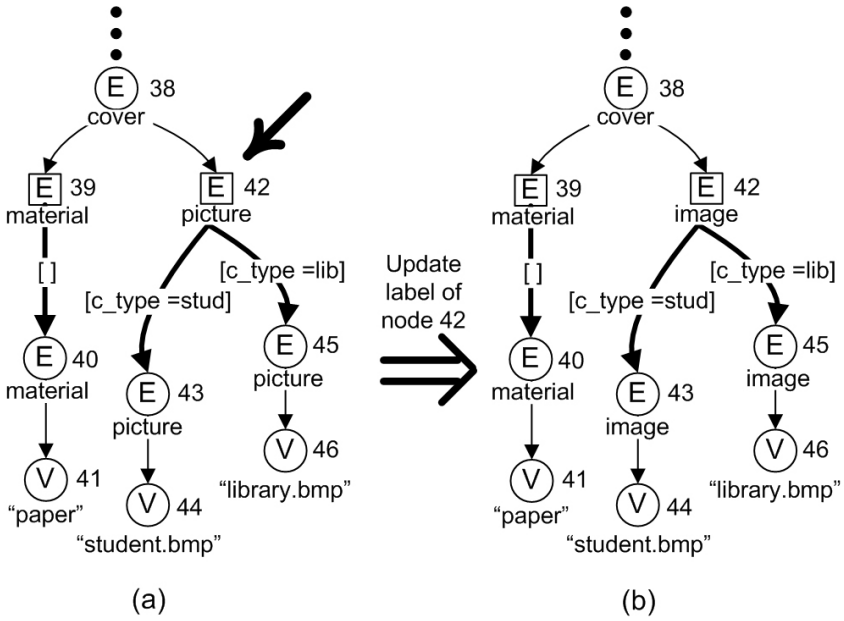
Παράδειγμα 5.3 *Στο σχήμα 5.3 μπορούμε να δούμε πως η πράξη update_label εφαρμόζεται στον κόμβο 42 και αλλάζει την ετικέτα* `picture` *του κόμβου (και των κόμβων περιβάλλοντος 43 και 45 που είναι παιδιά αυτού), με μία νέα ετικέτα* `image`*. Η παράμετρος P είναι η έκφραση:*

```
/child::book/child::cover[ec()=''ed=gr'']/child->picture
```

ή συνεπτυγμένα:

```
/book/cover[ec() = ''ed=gr'']/->picture
```

η οποία έχει σαν αποτέλεσμα τον πολυδιάστατο κόμβο 42. Στην περίπτωση που επιθυμούσαμε να αλλάξουμε τις ετικέτες όλων των `picture` *κόμβων, ανεξάρτητα από την τιμή της διάστασης* `edition`*, θα έπρεπε να παραλείψουμε το κατηγόρημα* `[ec() = ''ed=gr'']`*.*



Σχήμα 5.3: Ενημέρωση ετικέτας κόμβου

Ενημέρωση ερμηνευτικού περιβάλλοντος (updating context)

Η πράξη *update_context(P, E)* χρησιμοποιείται για την ενημέρωση του ρητού περιβάλλοντος των κόμβων που προκύπτουν από την εκτέλεση της MXPath έκφρασης P. Η μεταβλητή E αντιπροσωπεύει μία *έκφραση περιβάλλοντος* (context expression), η οποία προσδιορίζει μέσω των πράξεων μεταξύ των ερμηνευτικών περιβαλλόντων (πχ. context union, context intersection κτλ.), το πως θα παραχθούν τα νέα ρητά περιβάλλοντα. Ακολουθεί ο αλγοριθμικός ορισμός της εν λόγω πράξης:

***update_context(P, E)**:*
*Είσοδος:**P**: Μία MXPath έκφραση.*
***E**: Μία έκφραση περιβάλλοντος (context expression).*

1. Έστω N_P το σύνολο των κόμβων που επιστρέφεται από την εκτέλεση της έκφρασης P.
2. Απέρριψε όλους τους μη κόμβους περιβάλλοντος από το N_P. Έστω M ένα

τμήμα (segment)του N_P τέτοιο ώστε κάθε μέλος του τμήματος αυτού να αποτελείται από όλους τους κόμβους στο N_P οι οποίοι είναι αδέρφια.
3. ***Για κάθε*** *$m \in M$* ***επανέλαβε***
- Συνέλεξε σε ένα σύνολο S_m όλα τα ρητά περιβάλλοντα των κόμβων που βρίσκονται στο m και στο S τα ρητά περιβάλλοντα όλων των αδερφικών κόμβων των κόμβων του m, οι οποίοι δεν βρίσκονται στο m.
- Έστω S_m^E το σύνολο των προσδιοριστών περιβάλλοντος που προκύπτουν από την εφαρμογή της E στα στοιχεία που περιέχει το S_m.
- ***Εάν*** *$S_m^E \cup S$ είναι ένα context deterministic σύνολο από προσδιοριστές περιβάλλοντος* ***τότε***
- Αντικατέστησε το ρητό περιβάλλον κάθε κόμβου στο m με το αντίστοιχο στοιχείο του S_m^E.
- Επαναπροσδιόρισε τις κληρονομούμενες καλύψεις περιβάλλοντος όλων των κόμβων του m και όλων των προγόνων και απογόνων τους στο προκύπτον δένδρο.
Τέλος εάν
4. Διέγραψε κάθε υποδένδρο με ρίζα σε κάθε κόμβο n για τον οποίο ισχύει $icc(n) = [-]$ καθώς και τις ακμές που οδηγούν σε κάθε n.
Τέλος επανάληψης

Τέλος

Η πράξη *update_context* μπορεί να εφαρμοστεί μόνο σε κόμβους στοιχείων/γνωρισμάτων περιβάλλοντος, καθώς οι κόμβοι αυτοί διαθέτουν κάποιο (ορισμένο από τον χρήστη) ρητό περιβάλλον.

Σε ότι αφορά την έκφραση περιβάλλοντος E, η σύνταξή της επιτρέπει τον προσδιορισμό ενός αριθμού πράξεων μεταξύ των ερμηνευτικών περιβαλλόντων. Η σύνταξη των εκφράσεων περιβάλλοντος είναι η ακόλουθη:

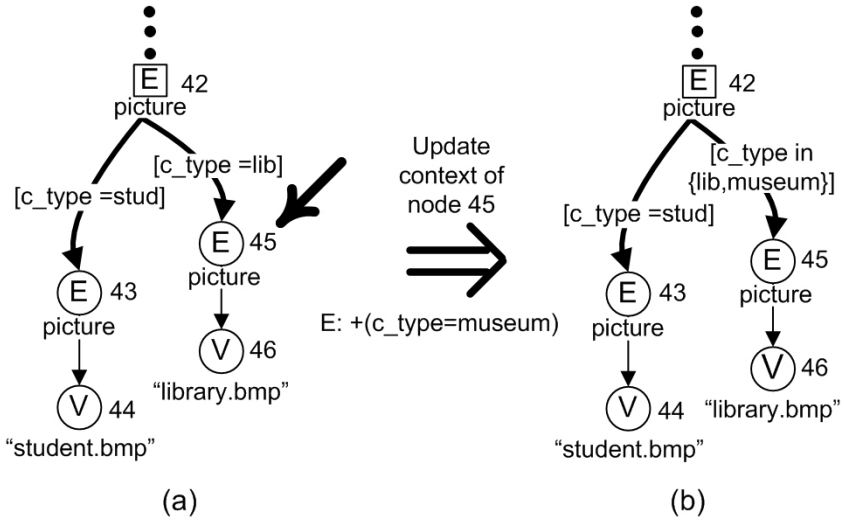
```
Context_Operation(Context_Modifier)
```

Η *πράξη ερμηνευτικού περιβάλλοντος* (`Context_Operation`) μπορεί να είναι είτε "new" είτε κάποιο σύμβολο εκ των "+", "-", και "*" αντιπροσωπεύοντας την πράξη της *ένωσης ερμηνευτικού περιβάλλοντος* (context union) ($\cup^c$), την πράξη της

διαφοράς ερμηνευτικού περιβάλλοντος (context difference) ($-^c$) και την πράξη της *τομής ερμηνευτικού περιβάλλοντος* (context intersection) ($\cap^c$) αντίστοιχα. Ο *μετατροπέας περιβάλλοντος* (`Context_Modifier`) είναι ένας προσδιοριστής περιβάλλοντος. Το νέο ρητό περιβάλλον που προκύπτει εφαρμόζοντας μία έκφραση περιβάλλοντος *E = Context_Operation(Context_Modifier)* σε ένα ρητό περιβάλλον `Explicit_Context` είναι είτε ο `Context_Modifier` εάν η πράξη ερμηνευτικού περιβάλλοντος έχει τιμή "new", είτε το αποτέλεσμα της εκτέλεσης της έκφρασης:

```
Explicit_Context Context_Operation Context_Modifier
```

Οι πράξεις ερμηνευτικού περιβάλλοντος *context union* ($\cup^c$), *context difference* ($-^c$) και *context intersection* ($\cap^c$) έχουν οριστεί στα [SG02] and [Sta03].



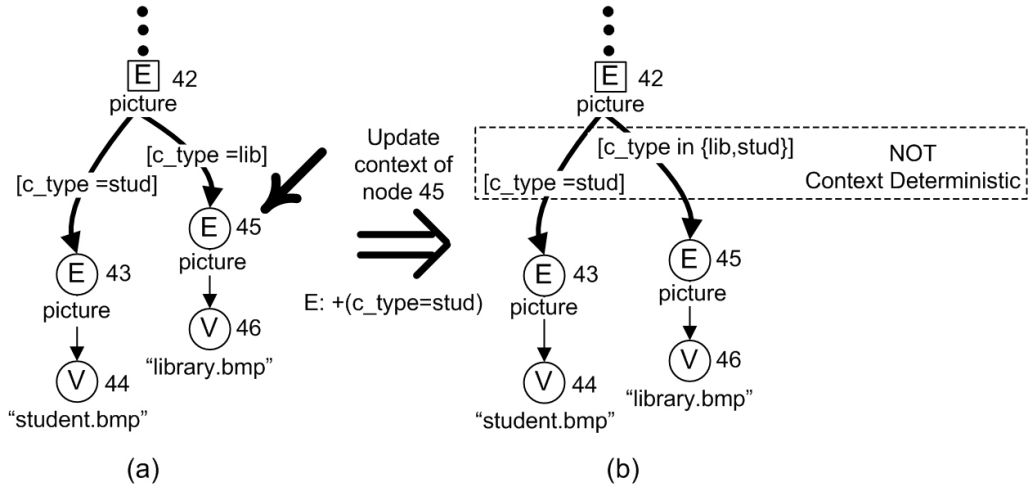
Σχήμα 5.4: Ενημέρωση του context ενός κόμβου

Παράδειγμα 5.4 *Το δένδρο του σχήματος 5.4(b) προκύπτει ενημερώνοντας το ερμηνευτικό περιβάλλον του κόμβου 45 στο δένδρο του σχήματος 5.4(a) (το οποίο είναι τμήμα του δένδρου του σχήματος 2.2). Η σχετική MXPath έκφραση P είναι η:*

```
/child::book/child::
cover[ec()=''ed=gr'']/child::picture[ec()=''c_type=lib'']
```

Η έκφραση περιβάλλοντος E = `[+(c_type=museum)]` *προσθέτει την τιμή "*`museum`*" στις τιμές της διάστασης "*`c_type`*" για να σχηματίσει το νέο ρητό περιβάλλον του κόμβου 45 το οποίο γίνεται* `[c_type in {lib,museum}]`*. Υπολογίζοντας ξανά τις κληρονομούμενες καλύψεις περιβάλλοντος έχουμε* `icc(46)=icc(45)=[c_type in {lib,museum}, ed=gr]` *και* `icc(42)=[ed=gr,c_type in {lib,stud,museum}]`*.*

Να επισημάνουμε ότι εάν η εκτέλεση της P *επέστρεφε δύο ή περισσότερους κόμβους περιβάλλοντος οι οποίοι θα ήταν όλοι παιδιά του ίδιου πολυδιάστατου κόμβου, αυτοί οι κόμβοι θα ενημερώνοντας ταυτόχρονα, διότι το τελικό δένδρο θα έπρεπε να παραμείνει ακέραιο σε ότι αφορά το ερμηνευτικό περιβάλλον.*



Σχήμα 5.5: Μη επιτρεπτή ενημέρωση του context ενός κόμβου

Παράδειγμα 5.5 *(Συνεχίζοντας από το παράδειγμα 5.4) Παρατηρούμε ότι εάν η έκφραση* E *που χρησιμοποιείται στο παράδειγμα 5.4 ήταν της μορφής* `E = [+(c_type =stud)]`*, η πράξη της ενημέρωσης δεν θα επιτρεπόταν, αφού αυτή θα παραβίαζε την συνθήκη κατά την οποία το MXML δένδρο θα πρέπει να παραμένει ακέραιο ως προς το ερμηνευτικό περιβάλλον. Αυτό φαίνεται στο σχήμα 5.5.*

Είναι φανερό ότι εάν το *Context_Operation* της πράξης *update_context* είναι είτε "*" (context intersection) είτε "-" (context difference), τότε το σύνολο των

προσδιοριστών περιβάλλοντος που προκύπτουν από την ενημέρωση αυτή είναι πάντοτε ακέραιο ως προς το ερμηνευτικό περιβάλλον. Σε αυτή την περίπτωση, ο αλγόριθμος που παρουσιάστηκε παραπάνω, μπορεί να βελτιωθεί με την αφαίρεση του ελέγχου σύμφωνα με τον οποίο εξετάζεται εάν το σύνολο $S_m^E \cup S$ παρουσιάζει ακεραιότητα ερμηνευτικού περιβάλλοντος. Ωστόσο, αυτό δεν ισχύει στην περίπτωση κατά την οποία η πράξη ερμηνευτικού περιβάλλοντος (`Context_Operation`) είναι είτε "+" (context union) είτε "new". Τότε, ο εν λόγω έλεγχος δε μπορεί να παραληφθεί.

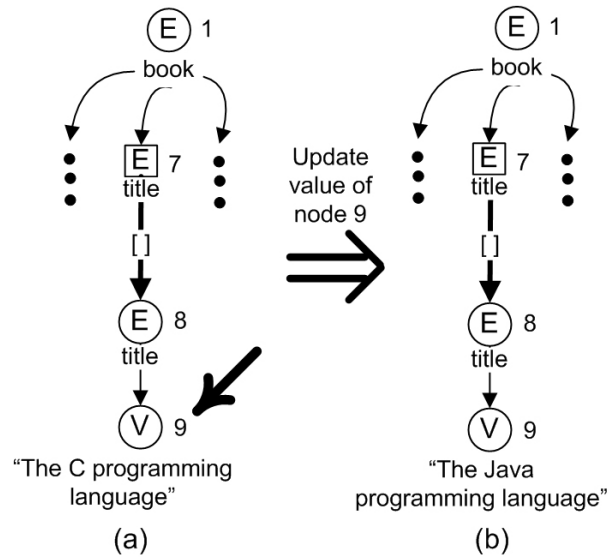
Ενημέρωση τιμών (updating values)

Για την ενημέρωση της τιμής κάποιου κόμβου, χρησιμοποιούμε την πράξη *update_value(P, C)*. Η πράξη αυτή αντικαθιστά την τιμή ενός ή περισσοτέρων τερματικών κόμβων ή *φύλλων* (leaf nodes), που προσδιορίζεται από την MXPath έκφραση P, με μία νέα τιμή που προκύπτει από την εκτέλεση της *έκφρασης τιμής* (value expression) C. Ο ορισμός της πράξης φαίνεται παρακάτω:

***update_value(P, C)**:*
*Είσοδος:**P**: Μία MXPath έκφραση.*
***C**: Μία έκφραση τιμής (value expression).*

1. Έστω N_P το σύνολο των κόμβων που επιστρέφεται από την εκτέλεση της έκφρασης P.
*2. **Για κάθε** $N \in N_P$ **επανέλαβε***
*- **Εάν** N είναι ένας κόμβος φύλλο **τότε***
- Αντικατέστησε την τιμή του N με το αποτέλεσμα της εκτέλεσης της έκφρασης C.
Τέλος εάν
Τέλος επανάληψης

Τέλος



Σχήμα 5.6: Ενημέρωση της τιμής κόμβου

Παράδειγμα 5.6 *Στο σχήμα 5.6(a) βλέπουμε ένα τμήμα του MXML δένδρου του σχήματος 2.2. Στο σχήμα 5.6(b) βλέπουμε το δένδρο το οποίο προκύπτει από το δένδρο 5.6(a) μετά την ενημέρωση της τιμής του κόμβου με id=9. Σύμφωνα με αυτή την ενημέρωση, η παλιά τιμή ("The C programming language") του κόμβου 9 έχει αντικατασταθεί από την τιμή "The Java programming language" αλλάζοντας το αλφαριθμητικό "C" σε "Java" (C expression). Η MXPath έκφραση P είναι η:*

```
/book/title
```

Αυτή η MXPath έκφραση επιστρέφει στην πραγματικότητα τον κόμβο με id=8, ωστόσο η update_value function αντιλαμβάνεται ότι πρέπει να χρησιμοποιήσει τον αντίστοιχο τερματικό κόμβο με id=9.

5.2.4 Αντικατάσταση υποδένδρων (replace)

Η πράξη *replace(P, T)* αντικαθιστά τα υποδένδρα του MXML δένδρου T, που έχουν σαν ρίζες τους κόμβους (περιβάλλοντος ή πολυδιάστατους) οι οποίοι επιστρέφονται κατά την εκτέλεση της MXPath έκφρασης P. Οι ρίζες των προς αν-

τικατάσταση υποδένδρων πρέπει να ταιριάζουν ως προς τον τύπο και την ετικέτα με την ρίζα του υποδένδρου T.

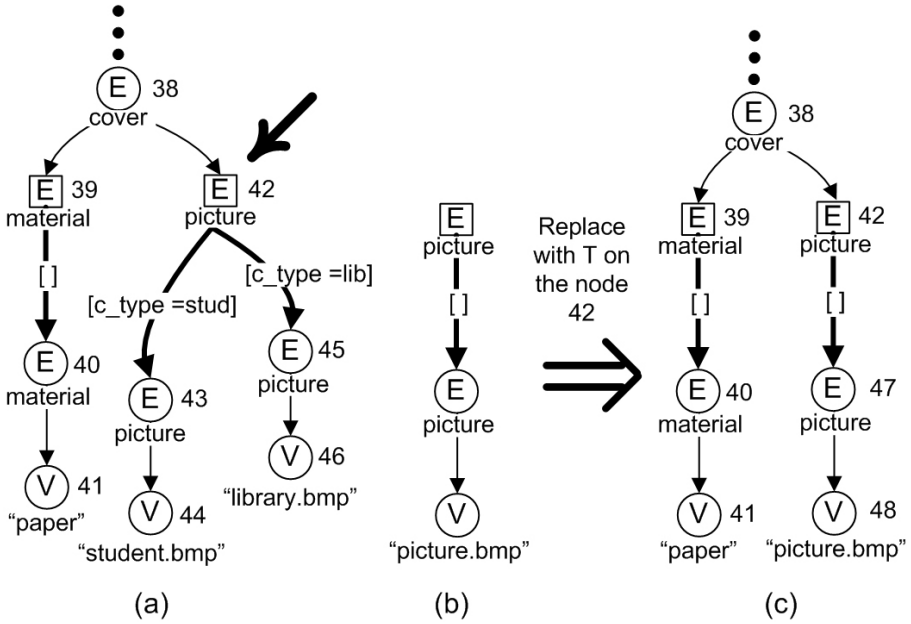
replace(P, T):
*Είσοδος:**P**: Μία MXPath έκφραση.*
__T__: Ένα κανονικοποιημένο (well-formed) MXML υποδένδρο.

1. Έστω N_P το σύνολο των κόμβων που επιστρέφεται από την εκτέλεση της έκφρασης P.
*2. **Για κάθε** $n \in N_P$ **επανέλαβε***
*- **Εάν** n και $root(T)$ είναι κόμβοι του ίδιου τύπου (περιβάλλοντος ή πολυδιάστατοι) **και** $label(n) = label(root(T))$ **τότε***
- Αντικατέστησε το υποδένδρο με ρίζα τον κόμβο n με το υποδένδρο T.
- Επαναπροσδιόρισε τις κληρονομούμενες καλύψεις περιβάλλοντος όλων των κόμβων n και όλων των προγόνων και απογόνων τους στο προκύπτον δένδρο.
Τέλος εάν
3. Διέγραψε το υποδένδρο που εχει ρίζα κάθε κόμβο m για τον οποίο ισχύει $icc(m) = [-]$ καθώς και τις ακμές που οδηγούν στους κόμβους m.
Τέλος επανάληψης

Τέλος

Παράδειγμα 5.7 *Στο Σχήμα 5.7 φαίνεται το δένδρο που προκύπτει (Σχήμα 5.7(c)) από την αντικατάσταση του υποδένδρου του Σχήματος 5.7(a) που έχει ρίζα τον κόμβο 42 από το δένδρο που φαίνεται στο σχήμα 5.7(b). Το υποδένδρο του Σχήματος 5.7(a) αποτελεί τμήμα του MXML δένδρου του Σχήματος 2.2).*

Από τον εκ νέου υπολογισμό των κληρονομούμενων καλύψεων περιβάλλοντος έχουμε `icc(48)= icc(47)= icc(42)= icc(38)= [ed=gr]`.



Σχήμα 5.7: Αντικατάσταση υποδένδρου

5.3 MXML Πράξεις Ενημερώσεως - Σχεσιακό Μοντέλο Αναπαράστασης

Η υλοποίηση MXML πράξεων ενημερώσεως, όπως περιγράφηκαν στην ενότητα 5.2, σε επίπεδο σχεσιακού σχήματος αποθήκευσης είναι το αντικείμενο της ενότητας αυτής. Πιο συγκεκριμένα, θα δείξουμε πως οι πράξεις *delete(P)* και *update_label(P,L)* μπορούν να υλοποιηθούν με αλγόριθμους των οποίων ο ψευδοκώδικα περιλαμβάνει SQL κώδικα, σύμφωνα με τη βασική προσέγγιση αποθήκευσης MXML δεδομένων. Κατά παρόμοιο τρόπο είναι δυνατή η υλοποίηση και των υπολοίπων πράξεων ενημέρωσης.

5.3.1 Διαγραφή υποδένδρων

Στο σημείο αυτό παρουσιάζεται ένας αλγόριθμος που υλοποιεί τη διαγραφή υποδένδρων σε επίπεδο αποθηκευμένων δεδομένων σε σχεσιακή βάση. Ο αλγόριθμος περιλαμβάνει τη διαδικασία *Delete(P)*, μέσω της οποίας καλούνται οι

υπο-διαδικασίες *Delete_subtree(N)*, και *Update_context_path(E)*, όπως φαίνεται παρακάτω:

Delete(P)
*Είσοδος:**P**: Μία MXPath έκφραση.*

1. Έστω N_P το σύνολο των κόμβων που επιστρέφεται από την εκτέλεση της έκφρασης P.
*2. **Για κάθε** $n \in N_P$ **επανέλαβε***
- Έστω p το node id που επιστρέφεται από την έκφραση ”SELECT parent_id FROM Node_Table WHERE node_id = n”
- Delete_subtree(n)
- Update_context_path(p)
Τέλος επανάληψης
3. Έστω D το σύνολο αποτελεσμάτων της έκφρασης ”SELECT node_id FROM Node_Table WHERE node_id NOT IN (SELECT DISTINCT node_id FROM Inherited_Coverage_Table)”
*4. **Για κάθε** $m \in D$ **επανέλαβε***
- Delete_subtree(m)
Τέλος επανάληψης

Τέλος

Η διαδικασία *Delete_subtree(N)* είναι υπεύθυνη για τη διαγραφή όλων των εγγραφών του *Node_table*, του *Explicit_Context_Table* και του *Inherited_Coverage_Table* που αντιστοιχούν σε όλους τους κόμβους των υποδένδρων που θα πρέπει να διαγραφούν. Παρακάτω φαίνεται η αλγοριθμική υλοποίηση της εν λόγω διαδικασίας:

Delete_subtree(N)
*Είσοδος:**N**: Ένα node id.*

1. - Έστω N_C το σύνολο αποτελεσμάτων που επιστρέφεται από την έκφραση

"SELECT node_id FROM Node_Table WHERE parent_id = N"
2. ***Για κάθε*** $c \in N_C$ ***επανέλαβε***
- Delete_subtree(c)
Τέλος επανάληψης
3. DELETE FROM Node_Table WHERE node_id = N
4. DELETE FROM Explicit_Context_Table WHERE node_id = N
5. DELETE FROM Inherited_Coverage_Table WHERE node_id = N

Τέλος

Η διαδικασία *Update_context_path(E)* είναι υπεύθυνη για τον υπολογισμό των νέων κληρονομούμενων καλύψεων περιβάλλοντος των κόμβων που ανήκουν στο μονοπάτι από τον γονέα του κόμβου που έχει διαγραφεί μέχρι την ρίζα του δένδρου. Η υλοποίηση της διαδικασίας αυτής φαίνεται παρακάτω:

Update_context_path(E)
*Είσοδος:**N**: Ένα node id.*

1. - Έστω $i = E$
2. ***Όσο*** $i \neq 0$ ***επανέλαβε***
- DELETE FROM Inherited_Coverage_Table WHERE node_id = i
- Έστω N_C *το σύνολο αποτελεσμάτων της έκφρασης*
"SELECT node_id FROM Node_Table WHERE parent_id = i"
- Έστω W *το σύνολο αποτελεσμάτων της έκφρασης*
"SELECT world_id FROM Inherited_Coverage_Table WHERE node_id in N_C*"*
- Για κάθε $w \in W$ ***επανέλαβε***
- INSERT INTO Inherited_Coverage_Table VALUES (i, w)
Τέλος επανάληψης
- Έστω i = "SELECT parent_id FROM Node_Table WHERE node_id = i"
Τέλος επανάληψης

Τέλος

5.3.2 Ενημέρωση ετικετών

Ο αλγόριθμος για την υλοποίηση της ενημέρωσης ετικετών των MXML κόμβων *update_label(P, L)*, έχει ως εξής:

Update_label(P,L)
*Είσοδος:**P**: Μία MXPath έκφραση.*

1. Έστω N_P το σύνολο των κόμβων που επιστρέφεται από την εκτέλεση της έκφρασης P.
*2. **Για κάθε** $n \in N_P$ **επανέλαβε***
- Έστω t η τιμή που επιστρέφεται από την έκφραση
"SELECT type FROM Node_Table WHERE node_id=n"
*- **Εάν** $(t = `ME' t = `MA')$ **τότε***
UPDATE Node_Table SET tag = L
WHERE (node_id = n) OR (parent_id = n)
Τέλος εάν
Τέλος επανάληψης

Τέλος

5.4 Συμπεράσματα - Παρατηρήσεις

Στο κεφάλαιο αυτό παρουσιάστηκαν αλγόριθμοι που υλοποιούν πράξεις ενημέρωσης των MXML κειμένων που είναι αποθηκευμένα σε ένα σχεσιακό σχήμα. Οι ενημερώσεις αυτές μπορούν να οριστούν τόσο σε επίπεδο γραφικού μοντέλου αναπαράστασης, όσο και σε επίπεδο σχεσιακού μοντέλου.

Οι αλγόριθμοι που βασίζονται στο γραφικό μοντέλο αναπαράστασης, δεν εξαρτώνται από το σχεσιακό σχήμα όπου είναι αποθηκευμένα τα MXML δεδομένα. Από την άλλη πλευρά, οι ενδεικτικοί αλγόριθμοι διαγραφής υποδένδρου και ενημέρωσης ετικετών που παρουσιάστηκαν, εμπεριέχουν SQL κώδικα, ο οποίος σχετίζεται άμεσα με το σχεσιακό σχήμα το οποίο χρησιμοποιείται. Σύμφωνα με τα παραπάνω, ένα πεδίο μελλοντικής έρευνας που ανακύπτει, περιλαμβάνει την εύρεση

διαδικασιών μέσω των οποίων, οι διάφορες πράξεις ενημέρωσης που απευθύνονται στο γραφικό μοντέλο αναπαράστασης θα μπορούν να εξειδικεύονται σύμφωνα με το εκάστοτε σχεσιακό σχήμα, παράγοντας σαν αποτέλεσμα τον κατάλληλα προσαρμοσμένο προς το σχεσιακό σχήμα SQL κώδικα, που θα περιέχεται στους αντίστοιχους αλγορίθμους επιπέδου σχεσιακού μοντέλου. Τέτοιου είδους διαδικασίες περιγράφονται εν μέρη και στο επόμενο κεφάλαιο, όπου παρουσιάζεται ο τρόπος σύμφωνα με τον οποίο MXPath εκφράσεις μετατρέπονται σε SQL ερωτήματα, δεδομένου ότι η MXPath θα μπορούσε να χρησιμοποιείται ως μέσω για τον προσδιορισμό των σημείων μέσα στο MXML δένδρο, όπου μπορούν να πραγματοποιηθούν ενημερώσεις.

Κεφάλαιο 6

Μετατροπή MXML ερωτημάτων σε SQL ερωτήματα

6.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται μία μεθοδολογία σχετικά με τον τρόπο επεξεργασίας MXML ερωτημάτων, προκειμένου αυτά να μπορέσουν να μετατραπούν σε SQL ερωτήματα προς MXML δεδομένα που είναι αποθηκευμένα σε κάποιο σχεσιακό σχήμα. Ακόμα, παρουσιάζεται ένας αλγόριθμος μετατροπής MXML ερωτημάτων σε SQL ερωτήματα, με τη χρήση σχετικού παραδείγματος.

Προκειμένου να αναπαραστήσουμε ερωτήματα με βάση το ερμηνευτικό περιβάλλον πάνω σε MXML δεδομένα, χρησιμοποιείται η γλώσσα MXPath, ενώ για την γραφική αναπαράσταση των ερωτημάτων αυτών χρησιμοποιούνται *πολυδιάστατα πρότυπα δένδρου* (Multidimensional tree patterns). Τα πρότυπα αυτά απεικονίζουν τους περιορισμούς και τις συνθήκες που πρέπει να πληρούν τα παραγόμενα αποτελέσματα ενός MXML ερωτήματος. Για την αποθήκευση των MXML κειμένων στη σχεσιακή βάση δεδομένων χρησιμοποιείται η τεχνική αποθήκευσης βάσει μονοπατιών, σε συνδυασμό με το Dewey σχήμα δεικτοδότησης. Επιπλέον, η αναπαράσταση του ερμηνευτικού περιβάλλοντος στο σχεσιακό σχήμα γίνεται με τη χρήση των διανυσμάτων κόσμων πράγμα που διευκολύνει εξαιρετικά τη διαδικασία της εν λόγο μετατροπής, αφού η δυνατότητα διεξαγωγής πράξεων και συγκρίσεων μεταξύ των διανυσμάτων κόσμων, σε συνδυασμό και με την παραπάνω περιγραφόμενη υποδομή, προσφέρει καλύτερη απόδοση κατά την παραγωγή

SQL ερωτημάτων.

6.2 Η MXPath και τα πρότυπα δένδρου

Αναφορικά με την MXML, χρειάζεται να επεκτείνουμε τον ορισμό των *προτύπων δένδρου* (tree patterns) προκειμένου να καλυφθούν όλα τα επιπρόσθετα χαρακτηριστικά (ερμηνευτικά περιβάλλοντα, διαφορετικοί τύποι κόμβων) ενός MXML δένδρου, αλλά και να μπορούν να απεικονιστούν οι συνθήκες που ενσωματώνει ένα MXML ερώτημα. Σε αυτή την ενότητα, παρουσιάζουμε τα *πολυδιάστατα πρότυπα δένδρου* (Multidimensional tree patterns ή MTPs), ορίζουμε τη σημασιολογία τους αναφορικά με το *ταίριασμα προτύπων* (pattern match) και δείχνουμε πως μπορούμε να αναπαραστήσουμε MXPath εκφράσεις ως MTPs. Έτσι, δεδομένου ενός MXPath ερωτήματος που περιέχει ακμές περιβάλλοντος (σύμβολο "/") ή πολυδιάστατες ακμές (σύμβολο "$->$") που οδηγούν σε κόμβους παιδιά, ακμές που οδηγούν σε κόμβους *απογόνους* (descendents) (σύμβολο "//"), κατηγορήματα, περιοριστές περιβάλλοντος, ονόματα *μπαλαντέρ* (wildcards) (σύμβολο "*") και κόμβους τιμής, είναι δυνατόν να κατασκευάσουμε MTPs.

Ορισμός 6.2.1. Ένα *πολυδιάστατο πρότυπο δένδρου* (multidimensional tree pattern ή MTP) είναι ένα ζεύγος *P=(T,F)*, όπου T είναι ένα δένδρο και F ένας περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος, τέτοιο ώστε: (i) Εάν $N = N_C \cup N_M \cup N_V$ είναι το σύνολο των κόμβων του *T*, όπου N_C το σύνολο των κόμβων περιβάλλοντος, N_M το σύνολο των πολυδιάστατων κόμβων και N_V το σύνολο των κόμβων τιμής, N_T=$\{C, M, V\}$ είναι το σύνολο όλων των δυνατών τύπων ενός κόμβου (*C* για κόμβο περιβάλλοντος, *M* για πολυδιάστατο κόμβο και V για κόμβο τιμής), L το σύνολο όλων των δυνατών ονομάτων ενός κόμβου περιβάλλοντος ή πολυδιάστατου κόμβου, V το σύνολο όλων των δυνατών τιμών ενός κόμβου τιμής, p το σύνολο των κατηγορημάτων (predicates), p_C το σύνολο των κατηγορημάτων που περιέχουν περιοριστές ρητού περιβάλλοντος και p_M ένα σύνολο κατηγορημάτων τέτοιο ώστε $p_M = p - p_C$, τότε κάθε κόμβος $n \in N$ έχει έναν προσδιοριστή $n_T \in N_T$, κάθε κόμβος $n_{CM} \in (N_C \cup N_M)$ έχει ένα όνομα (ετικέτα) $l \in (L \cup \{*\})$, όπου το σύμβολο "*" αναπαρηστά όλα τα πιθανά ονόματα, κάθε κόμβος $n_C \in N_C$ έχει προαιρετικά ένα ή περισσότερα κατηγορήματα $p' \in p$, κάθε κόμβος $n_M \in N_M$ έχει προαιρετικά ένα ή περισσότερα κατηγορή-

ματα $p'_M \in p_M$ και κάθε κόμβος $n_V \in N_V$ έχει μία τιμή $v' \in V$. (ii) Κάθε ακμή του T έχει έναν συγκεκριμένο τύπο, ο οποίος αναπαριστάται γραφικά με λεπτή γραμμή όταν πρόκειται για ακμή περιβάλλοντος, με παχιά γραμμή όταν πρόκειται για πολυδιάστατη ακμή, με διπλή γραμμή όταν πρόκειται για ακμή που οδηγεί σε απογόνους (πλην των κόμβων τιμής) και με διακεκομμένη γραμμή όταν πρόκειται για ακμή που οδηγεί σε κόμβο τιμής.

Ορισμός 6.2.2. Έστω ένα πολυδιάστατο πρότυπο δένδρου *P=(T,F)*. Ορίζουμε ως *κόμβο διακλάδωσης* (branching node) του *P*, κάθε κόμβο του *T* του οποίου ο αριθμός των παιδιών είναι μεγαλύτερος του 1, ως *ακμή διακλάδωσης* (branching edge) κάθε ακμή που έχει σαν αφετηρία έναν κόμβο διακλάδωσης και ως *ρίζα διακλάδωσης* (root of branch) κάθε κόμβο στον οποίο οδηγεί μία ακμή διακλάδωσης.

Ορισμός 6.2.3. Έστω ένα πολυδιάστατο πρότυπο δένδρου *P=(T,F)*. Ορίζουμε ως *διακλάδωση* (branch) του *P*, κάθε υποδένδρο του *T* του οποίου η ρίζα αποτελεί ρίζα διακλάδωσης.

Δεδομένου ενός MXPath ερωτήματος, είναι δυνατή η απεικόνιση του ερωτήματος αυτού με την μορφή ενός πολυδιάστατου προτύπου δένδρου.

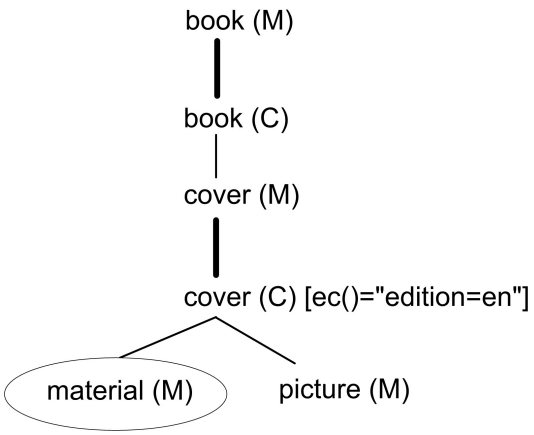
Ορισμός 6.2.4. Έστω ένα MXPath ερώτημα *Q*. Ορίζουμε ως *P=(T,F)* το πολυδιάστατο πρότυπο δένδρου που προκύπτει από το *Q*, τηρουμένων των ακόλουθων συνθηκών: (i) Το *F* είναι ο περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος του *Q*. (ii) Ακολουθώντας τα βήματα του *Q* με κατεύθυνση από αριστερά προς τα δεξιά, (α) για κάθε βήμα περιβάλλοντος προστίθενται δύο κόμβοι K_1 και K_2 στο δένδρο *T* ως εξής: (α1) Ο έλεγχος κόμβου του βήματος του *Q* τίθεται ως *l* (ετικέτα) στους κόμβους K_1 και K_2. (α2) Ο τύπος n_T του κόμβου K_1 χαρακτηρίζεται ως πολυδιάστατος και η ακμή με την οποία συνδέεται ο κόμβος K_1 με τον προηγούμενο κόμβο, εφόσον ο K_1 δεν είναι η ρίζα του *T*, χαρακτηρίζονται ως ακμή περιβάλλοντος. (α3) Ο τύπος n_T του κόμβου K_2 χαρακτηρίζεται ως περιβάλλοντος και η ακμή με την οποία συνδέεται ο κόμβος K_1 με τον K_2 χαρακτηρίζεται ως πολυδιάστατη ακμή. (α4) Κάθε κατηγόρημα που αφορά το βήμα του *Q* και αποτελεί περιοριστή ρητού περιβάλλοντος προστίθεται ως κατηγόρημα στον κόμβο K_2. (α5) Κάθε κατηγόρημα που αφορά το βήμα του *Q* και αποτελεί νέο MXPath ερώτημα προστίθεται ως διακλάδωση κάτω από τον κόμβο K_2. (β) για κάθε πολυδιάστατο βήμα προστίθεται ένας κόμβος K στο δένδρο *T* ως εξής: (β1) Ο έλεγχος

κόμβου του βήματος του Q τίθεται ως l στον κόμβο K. (β2) Ο τύπος n_T του κόμβου K χαρακτηρίζεται ως πολυδιάστατος και η ακμή με την οποία συνδέεται ο κόμβος K με τον προηγούμενο, εφόσον ο κόμβος K δεν είναι η ρίζα του T, χαρακτηρίζονται ως ακμή περιβάλλοντος.

Με βάση τον παραπάνω ορισμό, προκύπτει κατά αντίστροφο τρόπο και ο ορισμός ενός MXPath ερωτήματος από ένα πολυδιάστατο πρότυπο δένδρου. Συνεπώς, η διαδικασία της εκτέλεσης ενός MXML ερωτήματος με τη μορφή ενός πολυδιάστατου προτύπου δένδρου (ταίριασμα προτύπων) πάνω σε κάποιο MXML κείμενο, ταυτίζεται με την εκτέλεση του αντίστοιχου ερωτήματος σε μορφή MXPath πάνω στο κείμενο αυτό.

Ορισμός 6.2.5. Έστω ένα πολυδιάστατο πρότυπο δένδρου *P=(T,F)*. Κάθε μονοπάτι του δένδρου *T*, το οποίο έχει ως αφετηρία τη ρίζα και κατάληξη τους κόμβους που αποτελούν τα αποτελέσματα εκτέλεσης του ερωτήματος που προκύπτει από το *P*, ονομάζεται *μονοπάτι επιλογής* (selection path) του *P*, ενώ το φύλλο κάθε τέτοιου μονοπατιού απεικονίζεται γραφικά με την ετικέτα του ή την τιμή του (εάν είναι κόμβος τιμής) μέσα σε κύκλο.

Παράδειγμα 6.1 *Έστω το MXPath ερώτημα:*
`/book/cover[ec()="edition=en"][->picture]/->material`.
Στο σχήμα 6.1, φαίνεται το πολυδιάστατο πρότυπο δένδρου που αναπαριστά το εν



Σχήμα 6.1: Πολυδιάστατο Πρότυπο Δένδρου (MTP)

λόγω ερώτημα. Όπως μπορούμε να δούμε, οι κόμβοι που έχουν χαρακτηριστεί με "M" προσδιορίζουν το γεγονός ότι οι κόμβοι αυτοί θα πρέπει να είναι πολυδιάστατοι, ενώ οι παχιές ακμές υποδηλώνουν ότι πρέπει να οδηγούν σε κόμβους περιβάλλοντος (σημειώνονται με "C"). Ακόμα, μπορούμε να δούμε τον περιοριστή ρητού περιβάλλοντος "`[ec()="edition=en"]`*" που εκφράζει τον περιορισμό σύμφωνα με τον οποίο, η συνθήκη που τίθεται σαν κατηγόρημα του ερωτήματος πάνω στον κόμβο περιβάλλοντος "*`cover`*", θα πρέπει να επαληθεύεται. Το δεύτερο κατηγόρημα που περιλαμβάνει το παραπάνω ερώτημα προσδιορίζει ότι κάτω από τον κόμβο περιβάλλοντος "*`cover`*" θα πρέπει να υπάρχει πολυδιάστατος κόμβος με ετικέτα "*`picture`*". Το κατηγόρημα αυτό εκφράζεται στο πολυδιάστατο πρότυπο δένδρου σαν διακλάδωση. Τέλος, μπορούμε να δούμε τους πολυδιάστατους κόμβους "*`material`*" που παράγονται σαν αποτελέσματα (στο τέλος του μονοπατιού επιλογής) του ερωτήματος να απεικονίζονται με έναν κύκλο, ο οποίος περιέχει την ετικέτα των κόμβων αυτών.*

6.3 Μετατροπή MXML ερωτημάτων σε SQL ερωτήματα

Για τη διαδικασία της μετατροπής MXML ερωτημάτων σε SQL ερωτήματα, ένας κατάλληλος αλγόριθμός χρησιμοποιείται. Σύμφωνα με τον αλγόριθμο αυτόν, μία MXPath έκφραση διαχωρίζεται σε μικρότερα υπο-μονοπάτια ή *τμήματα μονοπατιών* (segments), με βάση τους *κανόνες τμηματοποίησης μονοπατιών* (segmentation rules) που αναφέρονται παρακάτω. Βασισμένοι στα παραγόμενα τμήματα μονοπατιών του αρχικού ερωτήματος, επιλέγονται τα κατάλληλα *στιγμιότυπα πινάκων* της σχεσιακής βάσης (table instances) προκειμένου αυτά να χρησιμοποιηθούν στο FROM τμήμα του τελικού SQL ερωτήματος. Επιπλέον, τα κατηγορήματα που περιέχουν περιοριστές περιβάλλοντος, επεξεργάζονται σύμφωνα με το σχήμα αναπαράστασης του ερμηνευτικού περιβάλλοντος βάσει διάταξης, ώστε να συμπεριληφθούν ως *συνθήκες ερωτημάτων* (query conditions) στο WHERE τμήμα του SQL ερωτήματος. Τέλος, θα πρέπει να επισημάνουμε ότι ο αλγόριθμος μετατροπής είναι *αναδρομικός* (recursive), προκειμένου να καλύψει περιπτώσεις MXPath ερωτημάτων τα οποία περιέχουν διακλαδώσεις (branches).

6.3.1 Τμηματοποίηση μονοπατιών της MXPath

Έστω ότι αναπαριστούμε το MXPath ερώτημα Q σαν ένα πολυδιάστατο πρότυπο δένδρου (MTP) P. Μπορούμε τότε να διαιρέσουμε το P σε επιμέρους υπομονοπάτια (sub-paths ή segments) σύμφωνα με τους *κανόνες τμηματοποίησης μονοπατιών* (segmentation rules) που περιγράφονται παρακάτω. Κάθε κανόνας περιλαμβάνει ένα *σημείο τμηματοποίησης μονοπατιού* (segmentation point), σύμφωνα με το οποίο γίνεται η τμηματοποίηση του P:

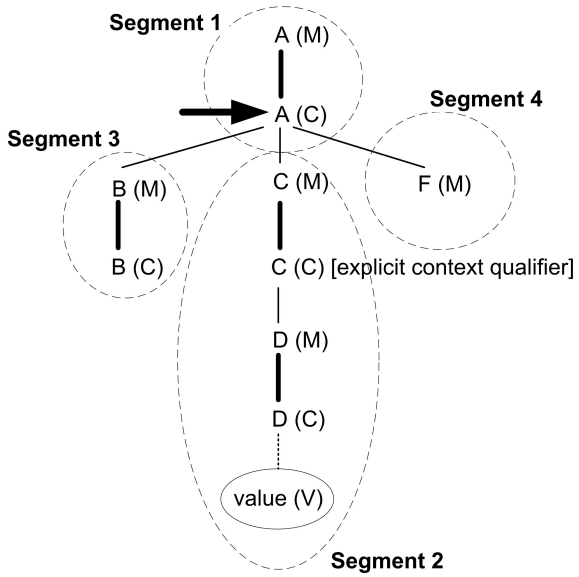
Κανόνας τμηματοποίησης μονοπατιών "Branch Segmentation Rule"

Έστω ένα πολυδιάστατο πρότυπο δένδρου $P = (T, F)$. Ο κανόνας τμηματοποίησης $Branch\ Segmentation\ Rule$ διαιρεί το T σε υποδένδρα σε κάθε σημείο τμηματοποίησης μονοπατιού S_b που αποτελεί κόμβο διακλάδωσης (branching node) του T. Τότε, το πρώτο υποδένδρο περιέχει το μονοπάτι από την ρίζα του T μέχρι και το σημείο τμηματοποίησης μονοπατιού S_b, ένα δεύτερο περιέχει το υπόλοιπο μονοπάτι (διακλάδωση) εντός του μονοπατιού επιλογής και όλα τα υπόλοιπα υποδένδρα περιλαμβάνουν τις υπόλοιπες, εκτός μονοπατιού επιλογής, διακλαδώσεις με κόμβο διακλάδωσης τον S_b.

Παράδειγμα 6.2 *Έστω το MXPath ερώτημα:*
`/A[B][->F]/C[explicit context qualifier]/D`.
Στο σχήμα 6.2, φαίνεται το δένδρο T του MTP που αναπαριστά το παραπάνω ερώτημα. Σε διακεκομμένους κύκλους φαίνονται τα τέσσερα τμήματα που προκύπτουν κατά την εφαρμογή του κανόνα τμηματοποίησης Branch Segmentation Rule, ενώ το βέλος δείχνει το σχετικό σημείο τμηματοποίησης.

Κανόνας τμηματοποίησης μονοπατιών "Context Segmentation Rule"

Έστω ένα πολυδιάστατο πρότυπο δένδρου $P = (T, F)$. Ο κανόνας τμηματοποίησης $Context\ Segmentation\ Rule$ διαιρεί το T σε δύο υποδένδρα σε κάθε σημείο τμηματοποίησης μονοπατιού S_c που αποτελεί κόμβο στον οποίο αντιστοιχεί κατηγόρημα που περιέχει έναν περιοριστή περιβάλλοντος. Το ένα υποδένδρο περιέχει το μονοπάτι από την ρίζα του T μέχρι και το σημείο τμηματοποίησης μονοπατιού S_c, ενώ το δεύτερο είναι το υποδένδρο κάτω από το S_c.



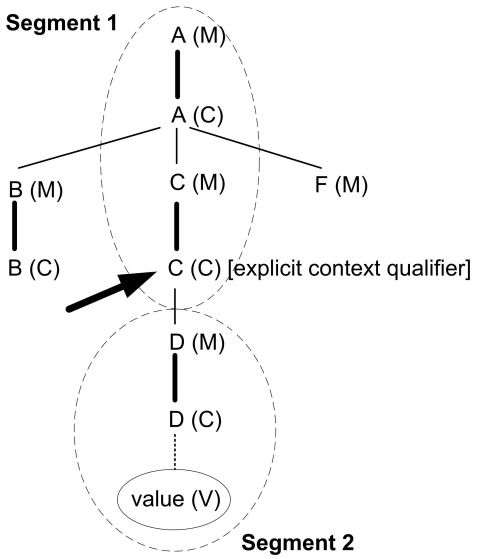
Σχήμα 6.2: Branch Segmentation Rule

Παράδειγμα 6.3 *Έστω το MXPath ερώτημα:*
`/A[B][− >F]/C[explicit context qualifier]/D`.
Στο σχήμα 6.3, φαίνεται το δένδρο T του MTP που αναπαριστά το παραπάνω ερώτημα. Σε διακεκομμένους κύκλους φαίνονται τα δύο τμήματα που προκύπτουν κατά την εφαρμογή του κανόνα τμηματοποίησης Context Segmentation Rule, ενώ το βέλος δείχνει το σχετικό σημείο τμηματοποίησης.

Κανόνας τμηματοποίησης μονοπατιών "* Segmentation Rule"

Έστω ένα πολυδιάστατο πρότυπο δένδρου $P = (T, F)$. Ο κανόνας τμηματοποίησης $* \; Segmentation \; Rule$ διαιρεί το T σε δύο υποδένδρα σε κάθε σημείο τμηματοποίησης μονοπατιού S_w που αποτελεί πατρικό κόμβο ενός κόμβου με ετικέτα "*" (συμβόλου μπαλαντέρ), χωρίς ο ίδιος ωστόσο να είναι κόμβος με ετικέτα "*". Το πρώτο υποδένδρο περιέχει το μονοπάτι από την ρίζα του T μέχρι και το σημείο τμηματοποίησης μονοπατιού S_w, ενώ το δεύτερο είναι το υποδένδρο κάτω από το S_w.

Παράδειγμα 6.4 *Έστω το MXPath ερώτημα:*



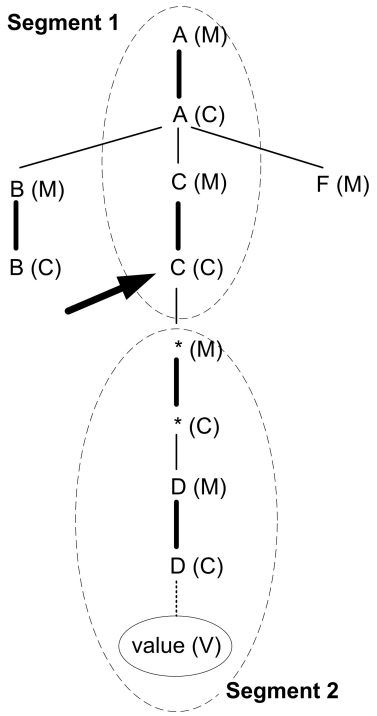
Σχήμα 6.3: Context Segmentation Rule

/A[B][− >F]/C//D.*
*Στο σχήμα 6.4, φαίνεται το δένδρο T του MTP που αναπαριστά το παραπάνω ερώτημα. Σε διακεκομμένους κύκλους φαίνονται τα δύο τμήματα που προκύπτουν κατά την εφαρμογή του κανόνα τμηματοποίησης * Segmentation Rule, ενώ το βέλος δείχνει το σχετικό σημείο τμηματοποίησης.*

Κανόνας τμηματοποίησης μονοπατιών "Value Segmentation Rule"

Έστω ένα πολυδιάστατο πρότυπο δένδρου $P = (T, F)$. Ο κανόνας τμηματοποίησης $Value\ Segmentation\ Rule$ διαιρεί το T σε δύο υποδένδρα σε κάθε σημείο τμηματοποίησης μονοπατιού S_v που αποτελεί πατρικό κόμβο ενός κόμβου τιμής. Το πρώτο υποδένδρο περιέχει το μονοπάτι από την ρίζα του T μέχρι και το σημείο τμηματοποίησης μονοπατιού S_v, ενώ το δεύτερο είναι ο ίδιος ο κόμβος τιμής.

Παράδειγμα 6.5 *Έστω το MXPath ερώτημα:*
/A[B][− >F]/C[explicit context qualifier]/D.
Στο σχήμα 6.5, φαίνεται το δένδρο T του MTP που αναπαριστά το παραπάνω ερώτημα. Σε διακεκομμένους κύκλους φαίνονται τα δύο τμήματα που προκύπτουν κατά



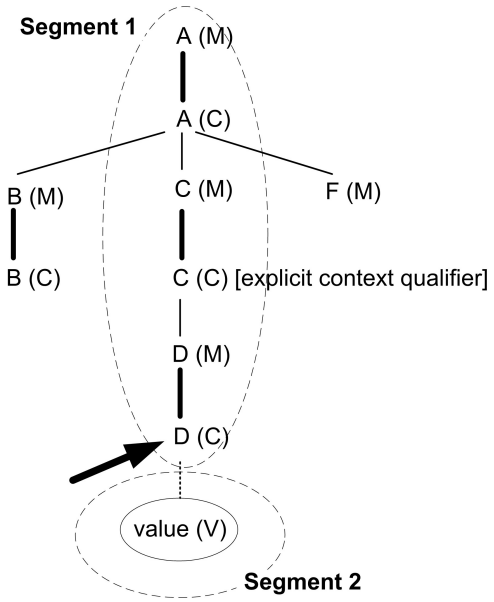
Σχήμα 6.4: * Segmentation Rule

την εφαρμογή του κανόνα τμηματοποίησης Value Segmentation Rule, ενώ το βέλος δείχνει το σχετικό σημείο τμηματοποίησης.

Να σημειωθεί ότι κατά τη διαδικασία της μετατροπή ερωτημάτων, όλοι οι παραπάνω κανόνες τμηματοποίησης μονοπατιών εφαρμόζονται αρχικά στο συνολικό MTP ενός ερωτήματος. Μετά από αυτό, οι κανόνες εφαρμόζονται αναδρομικά για κάθε διακλάδωση (branch) του αρχικού MTP πλην του μονοπατιού επιλογής, θεωρώντας ότι κάθε τέτοια διακλάδωση αποτελεί ένα νέο προς επεξεργασία MTP.

6.3.2 Αλγόριθμος μετατροπής

Σε αυτό το σημείο, περιγράφεται ο αλγόριθμος που χρησιμοποιείται για τη μετατροπή ενός MXPath ερωτήματος πάνω σε MXML δεδομένα σε SQL ερώτημα πάνω στο αντίστοιχο σχεσιακό σχήμα.



Σχήμα 6.5: Value Segmentation Rule

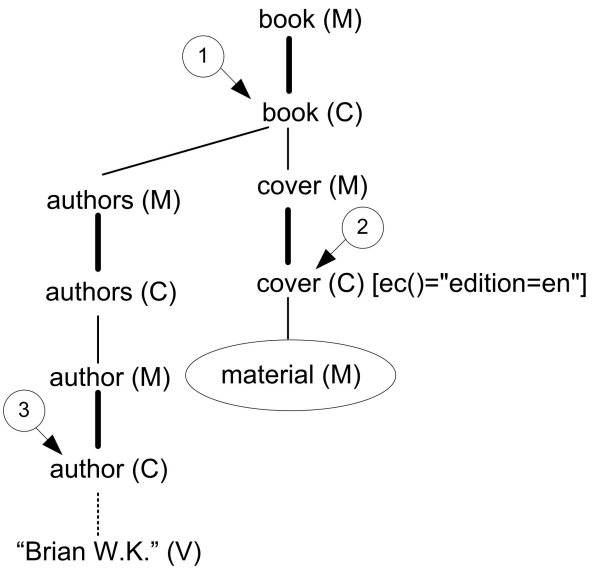
Για την καλύτερη κατανόηση του αλγορίθμου παραθέτουμε το ακόλουθο παράδειγμα, όπου δείχνουμε το πως ένα MXPath ερώτημα προς το MXML κείμενο του Σχήματος 3.9 μπορεί να μετατραπεί σε SQL ερώτημα προς το σχεσιακό σχήμα των Σχημάτων 3.6, 3.7 και 3.6, σύμφωνα με τον αλγόριθμο μετατροπής.

Παράδειγμα 6.6 *Ας θεωρήσουμε το παρακάτω MXPath ερώτημα.*
MXPath*:*

```
/book[authors[author="Brian W.K."]]/cover[ec()="ed=en"]
/->material
```

Στο σχήμα 6.6, φαίνεται το πολυδιάστατο πρότυπο δένδρου που εκφράζει το ερώτημα του παραδείγματος 6.6.

Αρχικά, ο αλγόριθμος τμηματοποιεί το υπό επεξεργασία MXPath ερώτημα του παραδείγματος 6.6 που εκφράζεται από το αντίστοιχο πολυδιάστατο πρότυπο δένδρου σε τμήματα μονοπατιών (*sub_path_1*, *sub_path_2*, ..., *sub_path_n*, *VALUE*), όπως φαίνεται στο παράδειγμα 6.7, σύμφωνα με τους κανόνες τμηματοποίησης μονοπατιών που αναφέρθηκαν στην ενότητα 6.3.1 και με τη σειρά που προκύπτει



Σχήμα 6.6: Πολυδιάστατο Πρότυπο Δένδρου (MTP)

κατά τη διάσχιση αρχικά του μονοπατιού επιλογής ξεκινώντας από τη ρίζα του MTP και στη συνέχεια της κάθε διακλάδωσης.

Παράδειγμα 6.7 *Εφαρμόζοντας τους κανόνες τμηματοποίησης μονοπατιών για το παραπάνω ερώτημα, διαιρούμε το πολυδιάστατο πρότυπο δένδρου του σχήματος 6.6 σε τρία σημεία τμηματοποίησης μονοπατιών, τα οποία υποδεικνύονται από τα χαρακτηρισμένα με τις ετικέτες ”1”, ”2” και ”3” βέλη (Branch Segmentation, Context Segmentation και Value Segmentation αντίστοιχα). Έτσι, παράγονται τα παρακάτω τμήματα:*

sub_path_1: /book
sub_path_2: cover
sub_path_3: ->material
sub_path_4: authors/author
VALUE: ”Brian W.K.”

Παρατηρήστε τα τμήματα sub_path_4: $authors/author$ *και VALUE:”*$BrianW.K.$*”, που προέρχονται από την τμηματοποίηση μίας διακλάδωσης (branch).*

Τα υπομονοπάτια (*sub_path_1*, *sub_path_2* ...) που αναφέρθηκαν στο παράδειγμα 6.7 χρησιμοποιούνται από τον αλγόριθμο για την κατασκευή των αντίστοι-

χων απλών εκφράσεων μονοπατιού (simple path expressions). Όπως φαίνεται στο παράδειγμα 6.8, αυτές οι εκφράσεις δημιουργούνται κατά την ενοποίηση κάθε επόμενου υπομονοπατιού με το προηγούμενό του. Επιπλέον, ένα *πρόθεμα* (prefix) *Q_pref* αναπαριστά το μονοπάτι από τη ρίζα του δένδρου έως τη ρίζα της υπό επεξεργασία διακλάδωσης, ενώ προστίθενται και οι κατάλληλοι "#" χαρακτήρες. Όταν ο αλγόριθμος καλείται για πρώτη φορά, η μεταβλητή *Q_pref* περιέχει αρχικά την τιμή NULL, καθώς την στιγμή εκείνη η υπό επεξεργασία διακλάδωση είναι ολόκληρο το δένδρο.

Παράδειγμα 6.8 *Χρησιμοποιώντας τα υπομονοπάτια του παραδείγματος 6.7 κατασκευάζονται οι αντίστοιχες απλές εκφράσεις μονοπατιού που φαίνονται παρακάτω:*

Path_expr_1': #/book
Path_expr_2': #/book#/cover
Path_expr_3': #/book#/cover#/->material
Path_expr_4': #/book#/authors#/author

Όταν οι απλές εκφράσεις μονοπατιού παραχθούν, ξεκινάει η κατασκευή του SQL ερωτήματος. Κατά τη δημιουργία του FROM τμήματος, ο αλγόριθμος χρησιμοποιεί ένα στιγμιότυπο πίνακα (table instance) του Element Table ή Attribute Table και ένα στιγμιότυπο πίνακα του Path Table για κάθε τμήμα μονοπατιού, εκτός από την περίπτωση όπου υπάρχει κάποιο "*" σύμβολο μπαλαντέρ. Σε αυτή την περίπτωση, ένα επιπλέον στιγμιότυπο πίνακα του Element Table ή Attribute Table απαιτείται για την αναπαράσταση κάθε πιθανού κόμβου που μπορεί να απεικονίζει το σύμβολο "*". Επιπλέον, ανάλογα με την ύπαρξη περιοριστών περιβάλλοντος ή κόμβων τιμής, σχετικά στιγμιότυπα πινάκων των ICC Table, EC Table ή Value Table μπορούν να χρησιμοποιηθούν.

Σχετικά με το ερώτημα του παραδείγματος 6.6, παρουσιάζουμε στο παράδειγμα 6.9 το τμήμα του SQL ερωτήματος που περιλαμβάνει την FROM ενότητα.

Παράδειγμα 6.9 *Δημιουργία την FROM ενότητα του SQL ερωτήματος:*

```
SELECT a_3.node_id FROM Element_Table a_1, Path_Table p_1,
Element_Table a_2, Path_Table p_2, EC_Table ec_2, Element_Table
a_3, Path_Table p_3 ...
```

Στην WHERE ενότητα του SQL ερωτήματος, οι κατάλληλες συνθήκες προστίθενται. Για κάθε έκφραση μονοπατιού, υπάρχει μία συνθήκη που ελέγχει την ύπαρξη ενός κόμβου στον Path Table, χρησιμοποιώντας την έκφραση LIKE. Εάν υπάρχει κάποιο κατηγόρημα ρητού περιβάλλοντος μαζί με μία έκφραση μονοπατιού, ο αλγόριθμος προσθέτει μία συνθήκη που συγκρίνει το σύνολο των κόσμων που εκφράζεται από τον προσδιοριστή περιβάλλοντος του κατηγορήματος, με τα διανύσματα κόσμων που είναι αποθηκευμένα στον EC Table, σύμφωνα με τον τελεστή σύγκρισης του κατηγορήματος αυτού. Εάν υπάρχουν μία ή περισσότερες διακλαδώσεις, η συνθήκη που προσθέτει ο αλγόριθμος στην WHERE ενότητα, περιλαμβάνει την εκ νέου εκτέλεση του αλγορίθμου για κάθε διακλάδωση.

Στο παράδειγμα 6.10, βλέπουμε το SQL ερώτημα που παράγεται κατά την εκτέλεση του αλγορίθμου μετατροπής, συμπεριλαμβανομένου της ενότητας WHERE.

Παράδειγμα 6.10 *Κατασκευή της ενότητας WHERE του SQL ερωτήματος:*

```
SELECT a_3.node_id
FROM Element_Table a_1, Path_Table p_1, Element_Table a_2, Path_Table p_2,
    EC_Table ec_2, Element_Table a_3, Path_Table p_3
WHERE p_1.path Like '#/book' and
    a_1.path_id = p_1.path_id and
    a_1.node_id = ANY (
```

Evaluate_branch()

```
) and
    p_2.path Like '#/book#/cover' and
    a_2.path_id = p_2.path_id and
    a_2.node_id = ec_2.node_id and
    ec_2.world_vector = 56 and
    a_2.node_id Like CONCAT(a_1.node_id, '%') and
    p_3.path Like '#/book#/cover#/->material' and
    a_3.path_id = p_3.path_id and
    a_3.node_id Like CONCAT(a_2.node_id, '%')
```

Παρατηρήστε την κλήσης της διαδικασίας (function call) `Evaluate_branch()` *που αντιπροσωπεύει την εκτέλεση του αλγορίθμου για τη διακλάδωση του σχήματος 6.6. Μετά την εκτέλεση, ο παραγόμενος SQL κώδικας τοποθετείται στο σημείο όπου πραγματοποιήθηκε η κλήση της διαδικασίας.*

Σε αυτό το σημείο θα πρέπει να επισημάνουμε ότι για κάθε δύο διαδοχικά στιγμιότυπα του πίνακα Element Table, ο αλγόριθμος προσθέτει μία SQL συνθήκη βασισμένη στο Dewey σχήμα δεικτοδότησης προκειμένου να εξασφαλιστεί η σχέση προγόνου-απογόνου μεταξύ αυτών των στιγμιοτύπων. Αυτή η συνθήκη εκφράζεται μέσω του χαρακτήρα ”%” σε μία SQL LIKE συνθήκη, εμπλέκοντας στην ουσία τους μοναδικούς κωδικούς αριθμούς (ids) των δύο στιγμιοτύπων.

Στην περίπτωση που υπάρχει ”*” σύμβολο μπαλαντέρ σε ένα υπομονοπάτι, ο χαρακτήρας ”%” χρησιμοποιείται στην SQL LIKE συνθήκη προκειμένου να εκφράσει το σύμβολο ”*”. Όπως προαναφέραμε, στην περίπτωση αυτή, χρησιμοποιείται ένα επιπλέον ενδιάμεσο στιγμιότυπο πίνακα, ενώ ελέγχεται η διαφορά μεταξύ των επιπέδων (levels) όπου βρίσκονται το στιγμιότυπο αυτό και το τρέχον στιγμιότυπο μέσα στο δένδρο, χρησιμοποιώντας τους (με βάση το Dewey σχήμα δεικτοδότησης, μοναδικούς κωδικούς αριθμούς των δύο στιγμιοτύπων.

Στις περιπτώσεις όπου μία έκφραση μονοπατιού περιέχει κάποια τιμή (value), η αντίστοιχη συνθήκη που χρησιμοποιείται στο SQL ερώτημα περιλαμβάνει ένα στιγμιότυπο του Value Table. Από την άλλη πλευρά, όταν ένα κατηγόρημα κληρονομούμενης κάλυψης περιβάλλοντος υπάρχει στο υπό επεξεργασία ερώτημα, ο αλγόριθμος λειτουργεί με τον ίδιο τρόπο όπως και στην περίπτωση των κατηγορημάτων ρητού περιβάλλοντος, μόνο που αυτή τη φορά χρησιμοποιείται ο ICC Table μέσω του αντίστοιχου στιγμιότυπου πίνακα.

Επιστρέφοντας στο παράδειγμα 6.10, ο αλγόριθμος εκτελείται μέσω της διαδικασίας `Evaluate_branch()` με σκοπό την επεξεργασία της διακλάδωσης που παρουσιάζει το ερώτημα. Γενικά, κατά την εκτέλεση του αλγορίθμου για κάθε διακλάδωση, λαμβάνεται υπόψη ένας αριθμός που συμβολίζει το βάθος (*Level* variable) του σημείου έναρξης της κάθε διακλάδωσης. Ο αριθμός αυτός χρησιμοποιείται με την SQL έκφραση ”SELECT SUBSTRING_INDEX”, η οποία επιστρέφει ένα τμήμα αλφαριθμητικού (substring) από τον Dewey δείκτη πριν από *Level* επαναλήψεις της διαχωριστικής τελείας. Με αυτόν τον τρόπο εξασφαλίζεται, μέσω του Dewey σχήματος δεικτοδότησης, ότι η υπό επεξεργασία διακλάδωση είναι κάτω από τον σωστό κόμβο.

Μετά την εκτέλεση του αλγορίθμου μετατροπής για τη διακλάδωση, παράγεται ο SQL κώδικα που φαίνεται στο παράδειγμα 6.11.

Παράδειγμα 6.11 *Αποτέλεσμα εκτέλεσης του αλγόριθμου μετατροπής για τη διακλάδωση:*

```
SELECT SUBSTRING_INDEX(a_1.node_id, '.', 2)
FROM Element_Table a_1, Path_Table p_1, Value_Table v
WHERE p_1.path Like '#/book#/authors#/author' and
    a_1.path_id = p_1.path_id and
    v.path_id = p_1.path_id and
    v.value = 'Brian W.K.' and
    v.node_id Like CONCAT(a_1.node_id,"%")
```

Εισάγοντας τον SQL κώδικα που παράγεται κατά την εκτέλεση του αλγόριθμου μετατροπής για τη διακλάδωση (Παράδειγμα 6.11) μέσα στον κώδικα του παραδείγματος 6.10, στο σημείο όπου πραγματοποιείται η κλήση της διαδικασίας *Evaluate_branch()*, έχουμε το τελικό SQL ερώτημα που παρουσιάζεται στο παράδειγμα 6.12.

Παράδειγμα 6.12 *Το SQL ερώτημα στο σύνολό του:*

```
SELECT a_3.node_id
FROM Element_Table a_1, Path_Table p_1, Element_Table a_2, Path_Table p_2,
    EC_Table ec_2, Element_Table a_3, Path_Table p_3
WHERE p_1.path Like '#/book' and
    a_1.path_id = p_1.path_id and
    a_1.node_id = ANY (
    SELECT SUBSTRING_INDEX(a_1.node_id, '.', 2)
    FROM Element_Table a_1, Path_Table p_1, Value_Table v
    WHERE p_1.path Like '#/book#/authors#/author' and
        a_1.path_id = p_1.path_id and
        v.path_id = p_1.path_id and
        v.value = 'Brian W.K.' and
        v.node_id Like CONCAT(a_1.node_id,"%")
    ) and
    p_2.path Like '#/book#/cover' and
    a_2.path_id = p_2.path_id and
    a_2.node_id = ec_2.node_id and
    ec_2.world_vector = 56 and
    a_2.node_id Like CONCAT(a_1.node_id, '%') and
    p_3.path Like '#/book#/cover#/->material' and
    a_3.path_id = p_3.path_id and
    a_3.node_id Like CONCAT(a_2.node_id, '%')
```

Το αποτέλεσμα του παραπάνω SQL ερωτήματος, είναι ο πολυδιάστατος κόμβος με `node_id` *"1.1.7.1.1".*

6.4 Συμπεράσματα - Παρατηρήσεις

Στο κεφάλαιο αυτό περιγράψαμε έναν αλγόριθμο μετατροπής MXML ερωτημάτων εξεφρασμένων σε MXPath σε SQL ερωτήματα. Χρησιμοποιώντας ως τεχνική αποθήκευσης των MXML δεδομένων στη σχεσιακή βάση την τεχνική με βάση τα μονοπάτια, χρησιμοποιήθηκαν μέθοδοι κατάτμησης του πολυδιάστατου πρότυπου δένδρου (το οποίο αναπαριστά ένα MXML ερώτημα), προκειμένου να προκύψουν τα σχετικά συγκρίσιμα μεγέθη (υπομονοπάτια) για την κατασκευή των συνθηκών του SQL ερωτήματος.

Επιπλέον, όπως έχουμε προαναφέρει, πολύ σημαντικός είναι ρόλο της τεχνικής αναπαράστασης του ερμηνευτικού περιβάλλοντος με βάση τη διάταξη σε ότι αφορά την κατασκευή του WHERE τμήματος του SQL ερωτήματος, Η τεχνική αυτή διευκολύνει σημαντικά τις πράξεις μεταξύ των συνόλων που εκφράζει κάθε προσδιοριστής περιβάλλοντος, ο οποίος περιέχεται σε μία συνθήκη ενός περιοριστή περιβάλλοντος.

Ακόμα, να σημειωθεί ότι ο παραπάνω αλγόριθμος είναι αναδρομικός και εκτελεί την ίδια διαδικασία μετατροπής για κάθε διακλάδωση (όπως αυτή έχει οριστεί) ξεχωριστά. Με αυτόν τον τρόπο, ο αλγόριθμος παραμένει λειτουργικός, ανεξάρτητα από τον αριθμό των διακλαδώσεων που περιέχονται σε κάποιο ερώτημα. Τέλος, να επισημάνουμε ότι ο αλγόριθμος μετατροπής υποστηρίζει MXPath εκφράσεις που περιέχουν τα σύμβολα "/", "//", "*", "->" αλλά και κατηγορήματα κληρονομούμενης κάλυψης περιβάλλοντος, κατηγορήματα ρητού περιβάλλοντος, διακλαδώσεις, στοιχεία, γνωρίσματα και κατηγορήματα που περιλαμβάνουν κόμβους τιμών.

Κεφάλαιο 7

Υλοποίηση MXML αποθήκευσης και υποβολής ερωτημάτων

7.1 Εισαγωγή

Προκειμένου να υλοποιήσουμε την προτεινόμενη τεχνική αποθήκευσης MXML δεδομένων με βάση τα μονοπάτια, αναπτύξαμε ένα σύστημα το οποίο ονομάστηκε ”MXML Storage Tools”. Το σύστημα αυτό δύναται να αποθηκεύει MXML κείμενα σε σχεσιακές βάσεις δεδομένων με βάση την παραπάνω τεχνική και χρησιμοποιώντας το αντίστοιχο σχεσιακό σχήμα. Επιπλέον, παρέχει τη δυνατότητα εκτέλεσης MXPath ερωτημάτων πάνω στη σχεσιακή βάση δεδομένων, αφού προηγηθεί η μετατροπή των ερωτημάτων αυτών σε SQL ερωτήματα.

7.2 Σκοπός της υλοποίησης

Ο βασικός σκοπός της υλοποίησης του συστήματος αποθήκευσης MXML δεδομένων σε σχεσιακή βάση είναι να μας δώσει τη δυνατότητα διεξαγωγής δοκιμών σε ότι αφορά τις προτεινόμενες τεχνικές αποθήκευσης. Πιο συγκεκριμένα, χρησιμοποιώντας το σχετικό *γραφικό περιβάλλον αλληλεπίδρασης με τον χρήστη* (Graphical User Interface ή GUI) που βασίζεται στο WEB μπορούμε να αποθηκεύσουμε εύκολα πραγματικά MXML δεδομένα σε μία σχεσιακή βάση δεδομένων της οποίας το σχήμα υπαγορεύεται από την προτεινόμενη τεχνική αποθήκευσης

με βάση τα μονοπάτια (path based).

Εκτός από τη δυνατότητα αποθήκευσης, το σύστημα παρέχει τη δυνατότητα υποβολής ερωτημάτων που εκφράζονται με βάση τη γλώσσα MXPath. Τα ερωτήματα αυτά υποβάλλονται με έναν εξίσου απλό τρόπο μέσω της ειδικής φόρμας που παρέχει και πάλι το γραφικό περιβάλλον αλληλεπίδρασης με τον χρήστη. Το σύστημα αναλαμβάνει την εκτέλεση των ερωτημάτων και επιστρέφει τους κόμβους του MXML δένδρου που αποτελούν τα αποτελέσματα της εκτέλεσης των ερωτημάτων αυτών. Για την εκτέλεση των MXPath ερωτημάτων, το σύστημα υλοποίησης ενσωματώνει τον αλγόριθμο μετατροπής των MXPath ερωτημάτων σε SQL ερωτήματα, όπως έχουμε περιγράψει στην Ενότητα 6.3. Συνεπώς, και σε αυτή την περίπτωση, ο σκοπός του υλοποιημένου συστήματος είναι να μας επιτρέπει την διεξαγωγή δοκιμαστικών μετατροπών από MXPath σε SQL ερωτήματα, προκειμένου να διαπιστώσουμε την ορθή λειτουργία του σχετικού αλγορίθμου μετατροπής. Για τον σκοπό αυτό, το σύστημα υλοποίησης, εκτός των τελικών αποτελεσμάτων που παράγει κατά την εκτέλεση ενός ερωτήματος, παρέχει και ορισμένες πληροφορίες που απεικονίζουν τη σταδιακή και κατά τμήματα μετατροπή της MXPath σε SQL σύμφωνα με τους κανόνες τμηματοποίησης τους οποίους ακολουθεί ο αλγόριθμος μετατροπής.

7.3 Αρχιτεκτονική της υλοποίησης

Για τη δημιουργία του συστήματος υλοποίησης "MXML Storage Tools" χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++ για την *συντακτική ανάλυση* (parsing) των MXML δεδομένων και η γλώσσα προγραμματισμού PHP τόσο για τη δημιουργία του *ενδιάμεσου περιβάλλοντος αλληλεπίδρασης με τον χρήστη* (interface) μέσω WEB, όσο και για την υλοποίηση του αλγορίθμου μετατροπής των MXPath ερωτημάτων σε SQL κώδικα. Επιπλέον, η σχεσιακή βάση δεδομένων υλοποιήθηκε σε περιβάλλον MySQL ενώ το όλο σύστημα εκτελείται κάτω από λειτουργικό σύστημα Linux.

7.3.1 Δομικές μονάδες - Λειτουργίες

Οι δομικές μονάδες που αποτελούν την προτεινόμενη υλοποίηση, περιλαμβάνουν τρία βασικά στοιχεία. Το πρώτο αφορά τον MXML *συντακτικό αναλυτή*

(parser), το δεύτερο το ενδιάμεσο περιβάλλον αλληλεπίδρασης με τον χρήστη και το τρίτο το σύστημα διαχείρισης της σχεσιακής βάσης δεδομένων.

Όπως φαίνεται και στο Σχήμα 7.1, η διαδικασία της ανάλυσης των MXML δεδομένων εκκινείται κατά την κλήση του MXML αναλυτή από το ενδιάμεσο περιβάλλον αλληλεπίδρασης με τον χρήστη και αφού ο χρήστης του συστήματος εισάγει τα απαραίτητα δεδομένα εισόδου. Τα δεδομένα αυτά περιλαμβάνουν τόσο το MXML κείμενο υπό μορφή αρχείου κειμένου με επέκταση ”mxml” όσο και τις πληροφορίες που περιλαμβάνουν όλες τις πιθανές τιμές των διαστάσεων του κειμένου αυτού υπό μορφή αρχείου κειμένου με συγκεκριμένη σύνταξη και επέκταση ”dim”. Στο Παράδειγμα 7.1 φαίνεται η μορφή ενός αρχείου με επέκταση ”dim”. Επιπλέον, να σημειωθεί ότι κατά την ανάλυση των MXML δεδομένων, ο MXML αναλυτής εκτελεί και συντακτικό έλεγχο.

Παράδειγμα 7.1 *Το κείμενο που παρουσιάζεται παρακάτω, αποτελεί το περιεχόμενο ενός αρχείου με επέκταση ”dim”. Πιο συγκεκριμένα περιλαμβάνει τις πιθανές τιμές των διαστάσεων του MXML κειμένου του Παραδείγματος 2.2.*

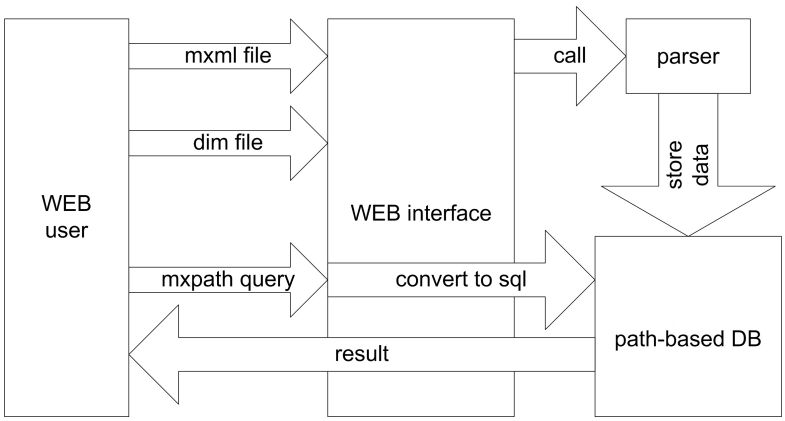
```
ed:gr,en
c_type:stud,te,lib
```

Παρατηρήστε ότι κάθε γραμμή του αρχείου αυτού περιέχει στην αρχή το όνομα της κάθε διάστασης, ενώ μετά τον χαρακτήρα ”:” ακολουθούν όλες οι πιθανές τιμές της εν λόγο διάστασης οι οποίες διαχωρίζονται μεταξύ τους με κόμμα. Να σημειώσουμε, ότι η διάταξη των τιμών των διαστάσεων που ακολουθείται στο αρχείο αυτό είναι και η διάταξη που ακολουθείται κατά την εφαρμογή της τεχνικής αναπαράστασης ερμηνευτικού περιβάλλοντος με βάση τη διάταξη (διανύσματα κόσμων).

Μετά την εκτέλεση της ανάλυσης των MXML δεδομένων, παράγεται το *αναλυτικό δένδρο* (parse tree), το οποίο διασχίζεται προκειμένου να τοποθετηθούν τα διάφορα τμήματα του MXML δένδρου στη σχεσιακή βάση δεδομένων της οποίας το σχήμα βασίζεται στα μονοπάτια. Η σχεσιακή βάση δεδομένων υλοποιείται σε περιβάλλον συστήματος διαχείρισης βάσεων δεδομένων MySQL.

Κατά την υποβολή ενός MXML ερωτήματος που εκφράζεται στη γλώσσα MXPath, μέσω του ενδιάμεσου περιβάλλοντος αλληλεπίδρασης του συστήματος

με τον χρήστη, πραγματοποιείται, σύμφωνα με τον αλγόριθμο μετατροπής, η μετατροπή του ερωτήματος αυτού σε SQL ερώτημα. Ακολουθεί η εκτέλεση του SQL ερωτήματος από το περιβάλλον της σχεσιακής βάσης δεδομένων και η επιστροφή των αποτελεσμάτων στον χρήστη και πάλι μέσω του ενδιάμεσου περιβάλλοντος αλληλεπίδρασης.



Σχήμα 7.1: Δομικές μονάδες και λειτουργίες συστήματος υλοποίησης

7.3.2 Περιορισμοί

Δεδομένου ότι το προτεινόμενο σύστημα υλοποίησης, αποτελεί μία εφαρμογή για την διεξαγωγή δοκιμών, στην παρούσα φάση ανάπτυξής της, υπόκειται σε ορισμένους περιορισμούς.

Πιο συγκεκριμένα, τα MXML κείμενα που εισάγονται από τον χρήστη (αρχεία με επέκταση ”mxml”) θα πρέπει να βρίσκονται σε κανονική μορφή, ενώ τα κείμενα αυτά θα πρέπει να ακολουθούν την παρακάτω γραμματική (EBNF), προκειμένου να περάσουν επιτυχώς από τον συντακτικό έλεγχο:

```
[ 1] S ::= (#x20 | #x9 | #xD | #xA)+
[ 2] Name ::= (Letter | '_' | ':') (NameChar)*
[ 3] NameChar ::= (Letter | Digit | '.' | '_' | '-' | ':')
[ 4] Letter ::= [a-z] | [A-Z]
[ 5] Digit ::= [0..9]
[ 6] M_Element ::= X_Element | MDElelement
```

```
[ 7] MDElement ::= SMDtag (Context_element)+ EMDtag
[ 8] SMDtag ::= '<@' Name (S Attribute)* S? '>'
[ 9] EMDtag ::= '</@' Name S? '>'
[10] Context_element ::= SContextSpec X_Element EContextSpec
[11] SContextSpec ::= '[' DimSpec (',' DimSpec)* ']'
[12] EContextSpec ::= '[/]'
[13] DimSpec ::= 'default' | (Name ValExpr)
[14] ValExpr ::= ( (Eq|Ne) Name) | ((In|NIn) SetExpr)
[15] In ::= 'in'
[16] NIn ::= 'not in'
[17] SetExpr ::= '{' Name (',' Name S?)* '}'
[18] X_Element ::= PrimeSTag ( (EmptySTag) | (TerinalSTag Content ETag)
[19] PrimeSTag ::= '<' Name (S Attribute)*
[20] EmptySTag ::= '/>'
[21] TerminalSTag ::= '>'
[22] ETag ::= '</' Name S? '>'
[23] Attribute ::= Name Eq ( (AttValue) | (SContextSpec AttValue EContextSpec)+)
[24] Eq ::= S? '=' S?
[25] Ne ::= S? '!=' S?
[26] Content ::= CharData? (M_Element CharData?)*
[27] CharData ::= anyChar - [ ^<& ]
[28] AttValue::= ('"' [^<&"] '"') | ("'" [^<&'] "'")
```

Σε ότι αφορά την διαδικασία της μετατροπής των MXPath ερωτημάτων σε SQL ερωτήματα, θα πρέπει να αναφερθεί ότι κατά την εισαγωγή των MXPath ερωτημάτων στο σύστημα, δεν έχει υλοποιηθεί η δυνατότητα αναφοράς συντακτικών λαθών στην MXPath έκφραση (τα MXPath ερωτήματα θεωρούνται ότι είναι συντακτικά ορθά). Ωστόσο, κάτι τέτοιο θα μπορούσε να αποτελέσει μία μεταγενέστερη προσθήκη στο σύστημα, αφού θα ήταν δυνατόν να υλοποιηθεί ως ξεχωριστή λειτουργική μονάδα του συστήματος.

7.4 Αποτελέσματα - Παρατηρήσεις

Συμπερασματικά, στο σημείο αυτό θα πρέπει να σημειωθεί ότι η διεξαγωγή πειραματικών μετρήσεων δεν αποτέλεσε το βασικό μέλημα του παραπάνω συστήματος υλοποίησης, αφού κάτι τέτοιο προϋποθέτει την κατασκευή κατάλληλης γεννήτριας παραγωγής μεγάλων ποσοτήτων από MXML δεδομένα, προκειμένου

να υπάρξουν αξιόπιστα αποτελέσματα σχετικά με το πως ο όγκος των δεδομένων επηρεάζει τόσο την αποθήκευση όσο και τον χρόνο εκτέλεσης ερωτημάτων. Η κατασκευή μίας τέτοιας γεννήτριας αποτελεί ένα ξεχωριστό και αρκετά πολύπλοκο πρόβλημα, αφού εξαιτίας της ύπαρξης του ερμηνευτικού περιβάλλοντος στα MXML δεδομένα, ανακύπτει το ζήτημα της διατήρησης της ακεραιότητας ερμηνευτικού περιβάλλοντος για τα παραγόμενα από την γεννήτρια MXML δεδομένα.

Παρά το γεγονός ότι δεν περιλαμβάνεται η δυνατότητα παραγωγής κατάλληλων δεδομένων προς διεξαγωγή πειραματικών μετρήσεων στο παραπάνω σύστημα υλοποίησης, δίνεται ωστόσο η δυνατότητα, κατά την εισαγωγή ενός προς αποθήκευση στην σχεσιακή βάση MXML κειμένου, αποθήκευσης ενός μεγαλύτερου (επαυξημένου) MXML κειμένου. Το κείμενο αυτό προκύπτει με τον κατά μήκος πολλαπλασιασμό του αρχικού MXML δένδρου (επανάληψη του ίδιου υποδένδρου που βρίσκεται κάτω από την ρίζα ενός MXML δένδρου, διατηρώντας ως ρίζα του τελικού επαυξημένου δένδρου την ρίζα του αρχικού δένδρου). Με τον τρόπο αυτό, μας παρέχεται η δυνατότητα διεξαγωγής στοιχειωδών δοκιμών σε ότι αφορά την μέτρηση της αποδοτικότητας του συστήματος (με αρκετά διακριτές τιμές κατά την μέτρηση του χρόνου) σε συνάρτηση με ορισμένους παράγοντες. Για παράδειγμα, διαπιστώθηκε ότι ο χρόνος εκτέλεσης των MXPath ερωτημάτων δεν μεταβάλλεται σημαντικά τόσο κατά την αύξηση του βάθους των μονοπατιών, όσο και κατά την αύξηση του αριθμού των διακλαδώσεων (branches) που περιέχονται στα ερωτήματα αυτά.

Παράρτημα Α΄

MXPath EBNF

```
[ 1] MXPath    ::=    ("["ICQualifier"]")? (,PathExpr)?
[ 2] PathExpr    ::=    ("/" RelativePathExpr?) | RelativePathExpr
[ 3] RelativePathExpr    ::=    StepExpr (("/" |  "//") StepExpr)*
[ 4] StepExpr   ::=   (ForwardStep) (Predicates)* | (MForwardStep) (MPredicates)*
[ 5] ForwardStep    ::=    (ForwardAxis NodeTest) |  AbbreviatedForwardStep
[ 6] AbbreviatedForwardStep   ::=  ("@" NameTest) | NameTest
[ 7] MForwardStep   ::=   (MForwardAxis NodeTest) |  MAbbreviatedForwardStep
[ 8] MAbbreviatedForwardStep   ::=  ("->" NameTest) | ("@->" NameTest)
[ 9] ForwardAxis    ::=    <"child" "::">
                             |  <"descendant" "::">
                             |  <"attribute" "::">
[10] MForwardAxis    ::=    <"child" "->">
                             |  <"descendant" "->">
                             |  <"attribute" "->">
[11] NameTest    ::=    QName |  Wildcard
[12] Wildcard    ::=    "*"
[13] Predicates    ::=    ("["ECQualifier"]")* | ("["RelativePathExpr"]")*
[14] MPredicates    ::=    ("["RelativePathExpr"]")*
[15] ComparisonExpr    ::=    GeneralComp
[16] ECQualifier ::= "ec()" ComparisonExpr CSpecifier
[17] ICQualifier ::= "icc()" ComparisonExpr CSpecifier
[18] GeneralComp    ::=    "=" |  "!=" |  "<" |  "<=" |  ">" |  ">="
[19] CSpecifier := """CSpecifierType (,CSpecifierType)*"""
[20] CSpecifierType := DimName "=" DimValue | DimName "in" "{"DimValue (,DimValue)*"}"
```

Παράρτημα Β΄

Γλωσσάρι

annotation	σήμανση
API	βιβλιοθήκη διαδικασιών
abbreviated syntax	συνεπτυγμένη σύνταξη
absolute expression	απόλυτη έκφραση
absolute position	απόλυτη θέση
annotation processor	επεξεργαστής σημάνσεων
attribute	γνώρισμα
attribute context edge	ακμή γνωρίσματος περιβάλλοντος
attribute edge	ακμή γνωρίσματος
attribute nodes	κόμβοι γνωρισμάτων
attribute reference edge	ακμή αναφοράς γνωρίσματος
attribute table	πίνακας γνωρισμάτων
axis	άξονας
binary approach	δυαδική προσέγγιση
bit	δυαδικό ψηφίο
branching edge	ακμή διακλάδωσης
branching node	κόμβος διακλάδωσης
branching root	ρύζα διακλάδωσης
children nodes	κόμβοι παιδιά
comparison operator	τελεστής σύγκρισης
context	ερμηνευτικό περιβάλλον
context attribute	γνώρισμα περιβάλλοντος
context attribute node	κόμβος γνωρίσματος περιβάλλοντος
context attribute ή CA table	πίνακας γνωρισμάτων περιβάλλοντος
context coverage	κάλυψη περιβάλλοντος
context determinism	ακεραιότητα ερμηνευτικού περιβάλλοντος
context difference	διαφορά ερμηνευτικού περιβάλλοντος
context edge	ακμή περιβάλλοντος
context element	στοιχείο περιβάλλοντος
context element node	κόμβος στοιχείου περιβάλλοντος
context element ή CE table	πίνακας στοιχείων περιβάλλοντος
context expression	έκφραση περιβάλλοντος
context intersection	τομή ερμηνευτικού περιβάλλοντος
context modifier	μετατροπέας περιβάλλοντος
context MXPath step	MXPath βήμα περιβάλλοντος
context node	κόμβος περιβάλλοντος
context operation	πράξη ερμηνευτικού περιβάλλοντος
context qualifier	περιοριστής περιβάλλοντος
context specifier	προσδιοριστή περιβάλλοντος
context union	ένωση ερμηνευτικού περιβάλλοντος
CPI (constraint preserving inlining algorithm)	αλγόριθμος ενσωμάτωσης σημασιολογικών περιορισμών
current node	τρέχον κόμβος

data-mining algorithm	αλγόριθμος εξόρυξης δεδομένων
default value	προκαθορισμένη τιμή
default XML view	προκαθορισμένη XML όψη
descendent	απόγονος
dimension	διάσταση
edge	ακμή
element	στοιχείο
element context edge	ακμή στοιχείου περιβάλλοντος
element edge	ακμή στοιχείου
element table	πίνακα στοιχείων
empty context specifier	κενός προσδιοριστής περιβάλλοντος
expanded syntax	εκτεταμένη σύνταξη
explicit context qualifier	περιοριστής ρητού περιβάλλοντος
explicit context table	πίνακας ρητού περιβάλλοντος
explicit context ή ec	ρητό περιβάλλον
explicit context ή EC table	πίνακας ρητού περιβάλλοντος
extended preorder numbering schema	εκτεταμένο προ-διατεταγμένο σχήμα αριθμοδότησης
extraction query	ερώτημα εξαγωγής
facet	έκφανση
flag	σηματοδότης
forest	δάσος δένδρων
function	διαδικασία
function call	κλήση διαδικασίας
global order	καθολική διάταξη
global schema	καθολικό σχήμα
graphical user interface ή GUI	γραφικό περιβάλλον αλληλεπίδρασης με τον χρήστη
hybrid inlining algorithm	υβριδικός αλγόριθμος ενσωμάτωσης
identification number (ID)	μοναδικός κωδικός αριθμός
index	δείκτης
indexing	δεικτοδότηση
information preservation	διατήρηση της πληροφορίας
inherited context coverage qualifier	περιοριστής κληρονομούμενης κάλυψης περιβάλλοντος
inherited context coverage ή ICC	κληρονομούμενη κάλυψη περιβάλλοντος
inherited context coverage ή ICC table	πίνακας κληρονομούμενης κάλυψης
inherited context ή IC	κληρονομούμενο περιβάλλον
inherited coverage ή IC table	πίνακας κληρονομούμενης κάλυψης
inlining	ενσωμάτωση
input	δεδομένα εισόδου
instance	στιγμιότυπο
interface	ενδιάμεσο περιβάλλον αλληλεπίδρασης με τον χρήστη
intergration	ολοκλήρωση
internet	διαδίκτυο
label	ετικέτα
leaf ή leaf node	φύλλο
level	ιεραρχικό επίπεδο
loading	φόρτωση
local order	τοπική διάταξη
local schema	τοπικό σχήμα
location step	βήμα εκφράσεως μονοπατιού
lossless encoding method	άνευ απωλειών μέθοδος κωδικοποίησης
lossless mapping schema	άνευ απωλειών σχήμα απεικόνισης
mapping	απεικόνιση
mapping repository	αποθήκη απεικονίσεων
mapping structure middleware	ενδιάμεση δομή απεικόνισης
mapping structure middleware	ενδιάμεση δομή απεικόνισης
middleware	ενδιάμεση δομή ή πλατφόρμα
module	υποσύστημα
multidimensional attribute	πολυδιάστατο γνώρισμα
multidimensional attribute node	πολυδιάστατος κόμβος γνωρίσματος
multidimensional attribute ή MA table	πίνακας πολυδιάστατων γνωρισμάτων
multidimensional element	πολυδιάστατο στοιχείο
multidimensional element node	πολυδιάστατος κόμβος στοιχείου
multidimensional element ή ME table	πίνακας πολυδιάστατων στοιχείων
multidimensional MXPath step	πολυδιάστατο MXPath βήμα
multidimensional tree pattern ή MTP	πολυδιάστατο πρότυπο δένδρου
multidimensional XML (MXML)	πολυδιάστατη XML

multidimensional XPath (MXPath)	πολυδιάστατη MXPath
MXML graph	MXML γράφος
MXML storage	αποθήκευση MXML δεδομένων
MXML tree	MXML δένδρο
MXPath expression	MXPath έκφραση
MXPath expression body	σώμα MXPath έκφρασης
MXPath predicate expression	MXPath έκφραση κατηγορήματος
MXPath step	MXPath βήμα
naive approach	βασική προσέγγιση
navigation	πλοήγηση
nesting	ενθυλάκωση
node	κόμβος
node table	πίνακας κόμβων
node test	έλεγχος κόμβου
NULL value	κενή τιμή
numbering	αριθμοδότηση
operator	τελεστής
optimization	βελτιστοποίηση
order based approach	προσέγγιση βάσει διάταξης
ordered-based context representation	αναπαράσταση ερμηνευτικού περιβάλλοντος βάσει διάταξης
ordering	διάταξη
overflow graph	γράφος υπερχείλισης
parent nodes	πατρικοί κόμβοι
parentID	μοναδικός κωδικός αριθμός γονέα
parse tree	αναλυτικό δένδρο
parser	συντακτικός αναλυτής
parsing	συντακτική ανάλυση
path based approach	προσέγγιση βάσει μονοπατιών
path expression ή location path expression	έκφραση μονοπατιού
path table	πίνακας μονοπατιών
pattern	πρότυπο
pattern match	ταίριασμα προτύπων
physical XML schema (or p-schema)	φυσικό XML σχήμα
possible worlds table	πίνακας πιθανών κόσμων
predicate	κατηγόρημα
prefix	πρόθεμα
probabilistic XML	πιθανολογική επέκταση της XML
query	ερώτημα
query conditions	συνθήκες ερωτημάτων
query engine	μηχανή αναζήτησης
query language	γλώσσα επερωτήσεων ή γλώσσα υποβολής ερωτημάτων
query processor	μηχανισμός επεξεργασίας ερωτημάτων
query tree	δένδρο επερωτήσεως
query workload	φόρτος ερωτημάτων
RDBMS	σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων
reconstruction XML view	XML όψη επανασύστασης
recursion	αναδρομή
reduction	αναγωγή
relational schema	σχεσιακό σχήμα
relative expression	σχετική έκφραση
relative position	σχετική θέση
rewriting	επαναδιατύπωση
root node	κόμβος ρίζας
root of branch	ρίζα διακλάδωσης
row	εγγραφή πίνακα
schema based XML storage	αποθήκευση XML δεδομένων βάσει XML σχήματος
schema oblivious XML storage	αποθήκευση XML δεδομένων ανεξαρτήτως XML σχήματος
schema transformation	μετασχηματισμός σχήματος
schema tree	δένδρο σχήματος
segment	τμήμα μονοπατιού
segmentation point	σημείο τμηματοποίησης μονοπατιού
segmentation rule	κανόνας τμηματοποίησης μονοπατιού
selection path	μονοπάτι επιλογής
semantic constrains	σημασιολογικοί περιορισμοί
semantic constrains	σημασιολογικοί περιορισμοί
set-containment relationship	περιορισμένη βάσει συνόλου σχέση

set-valued attribute	γνώρισμα συνόλου τιμών
shredding	αποδόμηση
sibling nodes	αδερφικοί κόμβοι
siblings	αδερφικοί κόμβοι
simple path expression	απλή έκφραση μονοπατιού
structure-oriented algorithm	δομικά προσανατολισμένος αλγόριθμος
sub-path	υπομονοπάτι
table instance	στιγμιότυπο πίνακα
tag	ετικέτα
temporal XML	χρονική επέκταση της XML
text node	κόμβος κειμένου
total ordering	καθολική διάταξη
transaction time	χρόνος συνδιαλλαγής
tree pattern	πρότυπο δένδρου
tree pattern matching	ταυτοποίηση πρότυπου δένδρου
tuple	εγγραφή
type approach	προσέγγιση βάσει είδους
universal approach	καθολική προσέγγιση
universal context	καθολικό περιβάλλον
update operations	πράξεις ενημέρωσης
updating	ενημέρωση
valid time	έγκυρος χρόνος
validating mapping schema	προς επικύρωση σχήμα απεικόνισης
validating mapping schema	προς επελήθευση σχήμα απεικόνισης
value	τιμή
value edge	ακμή τιμής
value expression	έκφρασης τιμής
value node	κόμβος τιμής
value table	πίνακας τιμών
vector	διάνυσμα
WEB	παγκόσμιος ιστός
well form (or canonical form)	κανονική μορφή
wildcard	μπαλαντέρ
witness tree	παριστάμενο δένδρο
world	κόσμος
world ordering	καθολική διάταξη κόσμων
world vector	διάνυσμα κόσμων
XML publishing	έκδοση XML δεδομένων
XML schema	XML σχήμα
XML storage	αποθήκευση XML δεδομένων
XML view	XML όψη
XPath expression	XPath έκφραση

Πίνακας Β΄.1: Γλωσσάρι απόδοσης αγγλικών όρων στα ελληνικά.

Βιβλιογραφία

[AQM+97] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[AYS02] S. Amer-Yahia and D. Srivastava. A mapping schema and interface for XML stores. In *Fourth ACM CIKM International Workshop on Web Information and Data Management (WIDM 2002), SAIC Headquaters, LcLean, Virginia, USA, November 8, 2002*, pages 23–30. ACM, 2002.

[AYU00] T. Amagasa, M. Yoshikawa, and S. Uemura. A Data Model for Temporal XML Documents. In *Database and Expert Systems Applications, 11th International Conference, DEXA 2000, London, UK, September 4-8, 2000, Proceedings*, volume 1873 of *Lecture Notes in Computer Science*, pages 334–344. Springer, 2000.

[AYU01] T. Amagasa, M. Yoshikawa, and S. Uemura. Realizing Temporal XML Repositories using Temporal Relational Databases. In *Proceedings of the Third International Symposium on Cooperative Database Systems and Applications, Beijing, China*, pages 63–68, 2001.

[BC00] Angela Bonifati and Stefano Ceri. Comparative Analysis of Five XML Query Languages. *SIGMOD Record*, 29(1):68–79, 2000.

[BDH03] V. P. Braganholo, S. B. Davidson, and C. A. Heuser. UXQuery: Building Updatable XML Views over Relational Databases. In *XVIII*

Simpósio Brasileiro de Bancos de Dados, 6-8 de Outubro, Manaus, Amazonas, Brasil, Anais/Proceedings, pages 26–40. UFAM, 2003.

[BDH04] V. P. Braganholo, S. B. Davidson, and C. A. Heuser. From XML View Updates to Relational View Updates: old solutions to a new problem. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 276–287. Morgan Kaufmann, 2004.

[Ben03] B. Benz. Relational XML - MS SQL Server, DB2 and Oracle. December 2003.

[BFM04] D. Barbosa, J. Freire, and A. O. Mendelzon. Information Preservation in XML-to-Relational Mappings. In *Database and XML Technologies, Second International XML Database Symposium, XSym 2004, Toronto, Canada, August 29-30, 2004, Proceedings*, Lecture Notes in Computer Science, pages 66–81. Springer, 2004.

[BFM05] D. Barbosa, J. Freire, and A. O. Mendelzon. Designing Information-Preserving Mapping Schemes for XML. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 109–120. ACM, 2005.

[BFRS02] P. Bohannon, J. Freire, P. Roy, and J. Simeon. From XML Schema to Relations: A Cost-Based Approach to XML Storage. In *Proceedings of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA*, pages 64–75. IEEE Computer Society, 2002.

[CCS00] V. Christophides, S. Cluet, and J. Simeon. On Wrapping Query Languages and Efficient XML Integration. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 141–152. ACM, 2000.

[CFI+00] M. J. Carey, D. Florescu, Z. G. Ives, Y. Lu, J. Shanmugasundaram, E. J. Shekita, and S. N. Subramanian. XPERANTO: Publishing

Object-Relational Data as XML. In *Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Adam's Mark Hotel, Dallas, Texas, USA*, pages 105–110, 2000.

[CJLP03] Zhimin Chen, H. V. Jagadish, Laks V. S. Lakshmanan, and Stelios Paparizos. From tree patterns to generalized tree patterns: On efficient evaluation of xquery. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 237–248. VLDB Endowment, 2003.

[CKN03] S. Chaudhuri, R. Kaushik, and J. F. Naughton. On Relational Support for XML Publishing: Beyond Sorting and Tagging. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, pages 611–622. ACM, 2003.

[CKS+00] M. J. Carey, J. Kiernan, J. Shanmugasundaram, E. J. Shekita, and S. N. Subramanian. XPERANTO: Middleware for Publishing Object-Relational Data as XML Documents. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 646–648. Morgan Kaufmann, 2000.

[CON99a] WORLD WIDE WEB CONSORTIUM. XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath, November 1999.

[CON99b] WORLD WIDE WEB CONSORTIUM. XSL Transformations (XSLT). http://www.w3.org/TR/xslt, November 1999.

[CON01] WORLD WIDE WEB CONSORTIUM. XQuery: A Query Language for XML. http://www.w3.org/TR/2001/WD-xquery-20010215/, February 2001.

[CON07a] WORLD WIDE WEB CONSORTIUM. Extensible Markup Language (XML). http://www.w3.org/XML, May 2007.

[CON07b] WORLD WIDE WEB CONSORTIUM. Extensible Markup Language (XML). http://www.w3.org/TR/xquery/, January 2007.

[CON07c] WORLD WIDE WEB CONSORTIUM. XML Path Language (XPath) 2.0. http://www.w3.org/TR/xpath20/, January 2007.

[Cor] Oracle Corp. XML Developers Kit. http://www.oracle.com/xml.

[DAYF04] F. Du, S. Amer-Yahia, and J. Freire. ShreX: Managing XML Documents in Relational Databases. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 1297–1300. Morgan Kaufmann, 2004.

[DFF+98] A. Deutsch, M. F. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. XML-QL. In *QL '98, The Query Languages Workshop, Boston, Massachussets, USA*, 1998.

[DFS99] A. Deutsch, M. F. Fernandez, and D. Suciu. Storing Semistructured Data with STORED. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 431–442. ACM Press, 1999.

[DT03] A. Deutsch and V. Tannen. MARS: A System for Publishing XML from Mixed and Redundant Storage. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 201–212. Morgan Kaufmann, 2003.

[Dyr01] Curtis E. Dyreson. Observing Transaction-Time Semantics with TTXPath. In *Proceedings of the Second International Conference on Web Information Systems Engineering (WISE2001)*, pages 193–202, 2001.

[FGS] Nikolaos Fousteris, Manolis Gergatsoulis, and Yannis Stavrakas. Storing-updating and querying Multidimensional XML documents using relational databases. In *IADIS International Conference WWW/INTERNET 2007, Vila Real, Portugal, 2007, Proceedings, Volume I.*

[FGS07a] N. Fousteris, M. Gergatsoulis, and Y. Stavrakas. Storing Multidimensional XML documents in Relational Databases. In *Database and Expert Systems Applications, 18th International Conference on Database and Expert Systems Applications, DEXA 2007 Regensburg, Germany 3-7 September 2007*, pages 23–33. Springer, 2007.

[FGS07b] Nikolaos Fousteris, Manolis Gergatsoulis, and Yannis Stavrakas. Updating Multidimensional XML Documents. In *iiWAS'2007 - The Ninth International Conference on Information Integration and Web-based Applications Services, 3-5 December 2007, Jakarta, Indonesia*, pages 257–266, 2007.

[FGS08] Nikolaos Fousteris, Manolis Gergatsoulis, and Yannis Stavrakas. Updating multidimensional XML documents. volume 4, pages 142–164, 2008.

[FGS11] Nikolaos Fousteris, Manolis Gergatsoulis, and Yannis Stavrakas. MXML Storage and the Problem of Manipulation of Context. In *First Workshop on Digital Information Management, 30-31 March, 2011, Corfu, Greece*, pages 45–60, 2011.

[FGS12] Nikolaos Fousteris, Manolis Gergatsoulis, and Yannis Stavrakas. Path-based MXML Storage and Querying. In *Second Workshop on Digital Information Management, 25-26 April, 2012, Corfu, Greece*, pages 51–65, 2012.

[FGS13] Nikolaos Fousteris, Manolis Gergatsoulis, and Yannis Stavrakas. MXML Path-Based Storage and Ordered-Based Context Manipulation. In *Database and Expert Systems Applications - 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings, Part II*, pages 203–212, 2013.

[FK99] D. Florescu and D. Kossmann. Storing and Querying XML Data using an RDMBS. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 22(3):27–34, 1999.

[FKS+02a] M. F. Fernandez, Y. Kadiyska, D. Suciu, A. Morishima, and W. Chiew Tan. SilkRoute: A framework for publishing relational data in XML. *ACM Transactions on Database Systems*, 27(4):438–493, 2002.

[FKS+02b] J. E. Funderburk, G. Kiernan, J. Shanmugasundaram, E. J. Shekita, and C. Wei. XTABLES: Bridging relational technology and XML. *IBM Systems Journal*, 41(4):616–641, 2002.

[FSG08] Nikolaos Fousteris, Yannis Stavrakas, and Manolis Gergatsoulis. Multidimensional XPath. In *iiWAS'2008 - The Tenth International Conference on Information Integration and Web-based Applications Services, 24-26 November 2008, Linz, Austria*, pages 162–169, 2008.

[FTS00] M. F. Fernandez, W. Chiew Tan, and D. Suciu. SilkRoute: trading between relations and XML. *Computer Networks*, 33(1-6):723–745, 2000.

[GB04] Sven Groppe and Stefan Bottcher. Query Reformulation for the XML standards XPath, XQuery and XSLT. In *Berliner XML Tage 2004, 11.-13. October 2004 in Berlin*, pages 53–64, 2004.

[GM03] F. Berzal Galiano and N. Marin. Review of Data on the Web: from relational to semistructured data and XML by Serge Abiteboul, Peter Buneman, and Dan Suciu. Morgan Kaufmann 1999. *SIGMOD Record*, 32(4):109–110, 2003.

[GMW99] R. Goldman, J. McHugh, and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In *WebDB (Informal Proceedings)*, pages 25–30, 1999.

[GSK01a] M. Gergatsoulis, Y. Stavrakas, and D. Karteris. Incorporating Dimensions in XML and DTD. In *Database and Expert Systems Applications, 12th International Conference, DEXA 2001 Munich, Germany, September 3-5, 2001, Proceedings*, volume 2113 of *Lecture Notes in Computer Science*, pages 646–656. Springer, 2001.

[GSK+01b] M. Gergatsoulis, Y. Stavrakas, D. Karteris, A. Mouzaki, and D. Sterpis. A Web-Based System for Handling Multidimensional Information through MXML. In *Advances in Databases and Information Systems, 5th East European Conference, ADBIS 2001, Vilnius, Lithuania, September 25-28, 2001, Proceedings*, volume 2151 of *Lecture Notes in Computer Science*, pages 352–365. Springer, 2001.

[HD13] Marouane Hachicha and Jerome Darmont. A Survey of XML Tree Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):29–46, 2013.

[HLM03] P. J. Harding, Q. Li, and B. Moon. XISS/R: XML Indexing and Storage System using RDBMS. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 1073–1076. Morgan Kaufmann, 2003.

[HvK10] Emiel S. Hollander and Maurice van Keulen. Storing and querying probabilistic xml using a probabilistic relational dbms. In *MUD*, pages 35–49, 2010.

[JLST01] H. V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava, and Keith Thompson. Tax: A tree algebra for xml. In *In Proc. DBPL Conf*, pages 149–164, 2001.

[KCKN04] R. Krishnamurthy, V. T. Chakaravarthy, R. Kaushik, and J. F. Naughton. Recursive XML Schemas, Recursive XML Queries, and Relational Storage: XML-to-SQL Query Translation. In *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA*, pages 42–53. IEEE Computer Society, 2004.

[KKN03] R. Krishnamurthy, R. Kaushik, and J. F. Naughton. XML-SQL Query Translation Literature: The State of the Art and Open Problems. In *Database and XML Technologies, First International XML Database Symposium, XSym 2003, Berlin, Germany, September 8, 2003, Proceedings*, Lecture Notes in Computer Science, pages 1–18. Springer, 2003.

[KP02] J. Kim and S. Park. Flexible Modification of Relational Schema by X2RMap in Storing XML into Relations. In *EurAsia-ICT 2002: Information and Communication Technology, First EurAsian Conference, Shiraz, Iran, October 29-31, 2002, Proceedings*, Lecture Notes in Computer Science, pages 330–338. Springer, 2002.

[Kud02] T. Kudrass. Management of XML Documents Without Schema in Relational Database Systems. *Information and Software Technology*, 44(4):269–275, 2002.

[LC00] D. Lee and W. W. Chu. Constraints-Preserving Transformation from XML Document Type Definition to Relational Schema. In *Conceptual Modeling - ER 2000, 19th International Conference on Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000, Proceedings*, Lecture Notes in Computer Science, pages 323–338. Springer, 2000.

[LJ88] Nikos A. Lorentzos and Roger G. Johnson. Extending relational algebra to manipulate temporal data. *Inf. Syst.*, 13(3):289–296, 1988.

[LM97] Nikos A. Lorentzos and Yannis G. Mitsopoulos. Sql extension for interval data. *IEEE Trans. Knowl. Data Eng.*, 9(3):480–499, 1997.

[LM01] Quanzhong Li and Bongki Moon. Indexing and querying xml data for regular path expressions. In *VLDB*, pages 361–370, 2001.

[MFK01] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML Queries on Heterogeneous Data Sources. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 241–250. Morgan Kaufmann, 2001.

[ML02] M. Mani and D. Lee. XML to Relational Conversion Using Theory of Regular Tree Grammars. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb. Revised Papers*, Lecture Notes in Computer Science, pages 81–103. Springer, 2002.

[NDM+01] J. F. Naughton, D. J. DeWitt, D. Maier, A. Aboulnaga, J. Chen, L. Galanis, J. Kang, R. Krishnamurthy, Q. Luo, N. Prakash, R. Ramamurthy, J. Shanmugasundaram, F. Tian, K. Tufte, S. Viglas, Y. Wang, C. Zhang, B. Jackson, A. Gupta, and R. Chen. The Niagara Internet Query System. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(2):27–33, 2001.

[NJ02] Andrew Nierman and H. V. Jagadish. ProTDB: Probabilistic Data in XML. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases*, pages 646–657, 2002.

[Pan02] T. Pankowski. XML-SQL: An XML Query Language Based on SQL and Path Tables. In *Proceedings of EDBT Workshops*, Lecture Notes in Computer Science, pages 184–209. Springer, 2002.

[PBM04] Sandeep Prakash, Sourav S. Bhowmick, and Sanjay Kumar Madria. Sucxent: An efficient path-based approach to store and query xml documents. In *DEXA*, pages 285–295, 2004.

[PCL02] Juan Manuel Pérez, María José Aramburu Cabo, and Rafael Berlanga Llavori. XRL: A XML-Based Query Language for Advanced Services in Digital Libraries. In *Database and Expert Systems Applications, 13th International Conference, DEXA 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings*, pages 300–309, 2002.

[RFHR03] M. Ramanath, J. Freire, J. R. Haritsa, and P. Roy. Searching for Efficient XML-to-Relational Mappings. In *First International XML Database Symposium, XSym 2003, Berlin, Germany, September 8, 2003, Proceedings*, Lecture Notes in Computer Science, pages 19–36. Springer, 2003.

[SG02] Yannis Stavrakas and Manolis Gergatsoulis. Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In *Advanced Information Systems Engineering, 14th International Conference, CAiSE 2002, Toronto, Canada, May 27-31, 2002, Proceedings*, volume 2348, pages 183–199, 2002.

[SKS+01] J. Shanmugasundaram, J. Kiernan, E. J. Shekita, C. Fan, and J. E. Funderburk. Querying XML Views of Relational Data. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 261–270. Morgan Kaufmann, 2001.

[SKWW00] A. Schmidt, M. L. Kersten, M. Windhouwer, and F. Waas. Efficient Relational Storage and Retrieval of XML Documents. In *WebDB (Informal Proceedings)*, pages 47–52, 2000.

[SSB+00] J. Shanmugasundaram, E. J. Shekita, R. Barr, M. J. Carey, B. G. Lindsay, H. Pirahesh, and B. Reinwald. Efficiently Publishing Relational Data as XML Documents. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 65–76. Morgan Kaufmann, 2000.

[SSK+01] J. Shanmugasundaram, E. J. Shekita, J. Kiernan, R. Krishnamurthy, S. Viglas, J. F. Naughton, and I. Tatarinov. A General Technique for Querying XML Documents using a Relational Database System. *SIGMOD Record*, 30(3):20–26, 2001.

[Sta03] Y. Stavrakas. *Multidimensional Semistructured Data: Representing and Querying Context-Dependent Multifaceted Information on theWeb*. PhD thesis, Department of Electrical and Computer Engineering, National Technical University of Athens, Greece, June 2003.

[STZ+99] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 302–314. Morgan Kaufmann, 1999.

[TDCZ02] F. Tian, D. J. DeWitt, J. Chen, and C. Zhang. The Design and Performance Evaluation of Alternative XML Storage Strategies. *SIGMOD Record*, 31(1):5–10, 2002.

[TIHW01] I. Tatarinov, Z. G. Ives, A. Y. Halevy, and D. S. Weld. Updating XML. In *SIGMOD Conference Proceedings*, pages 413–424, 2001.

[TVB+02] I. Tatarinov, S. Viglas, K. S. Beyer, J. Shanmugasundaram, E. J. Shekita, and C. Zhang. Storing and querying ordered XML using a relational database system. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*, pages 204–215. ACM, 2002.

[WMR03] L. Wang, M. Mulchandani, and E. A. Rundensteiner. Updating XQuery Views Published over Relational Data: A Roundtrip Case Study. In *Xsym*, pages 223–237, 2003.

[WZZ06] F. Wang, X. Zhou, and C. Zaniolo. Using XML to Build Efficient Transaction-Time Temporal Database Systems on Relational Databases. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 131. IEEE Computer Society, 2006.

[YASU01] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura. XRel: a path-based approach to storage and retrieval of XML documents using relational databases. *ACM Transactions on Internet Technology*, 1(1):110–141, 2001.

[ZD02] Shuohao Zhang and Curtis E. Dyreson. Adding Valid Time to XPath. In *Databases in Networked Information Systems, Second International Workshop, DNIS 2002, Aizu, Japan, December 16-18, 2002, Proceedings*, pages 29–42, 2002.