



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

**Γλώσσα Ερωτήσεων
για Δεδομένα Δενδρικής Δομής**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΠΟΥΡΟΥ ΠΑΝΑΓΙΩΤΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2003



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

Γλώσσα Ερωτήσεων για Δεδομένα Δενδρικής Δομής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΠΟΥΡΟΥ ΠΑΝΑΓΙΩΤΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3^η Νοεμβρίου 2003.

(Υπογραφή)

.....
Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

(Υπογραφή)

.....
Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

(Υπογραφή)

.....
Νεκτάριος Κοζύρης
Επίκουρος καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2003

(Υπογραφή)

.....

ΠΑΝΑΓΙΩΤΗΣ Γ. ΜΠΟΥΡΟΣ

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

© 2003 – All rights reserved

Ευχαριστίες

Η διπλωματική εργασία αυτή εκπονήθηκε στο Εργαστήριο Συστημάτων και Βάσεων Γνώσεων και Δεδομένων του Εθνικού Μετσόβιου Πολυτεχνείου. Αποτελέσει μια πολύ καλή και ενδιαφέρουσα επαφή και ασχολία με το χώρο των συστημάτων διαχείρισης δεδομένων και του προγραμματισμού. Στο σημείο αυτό, θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Τίμο Σελλή για την καθοδήγηση και τη βοήθεια που μου προσέφερε με τις συμβουλές και τις παρατηρήσεις του. Ακόμα, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Θεόδωρη Δαλαμάγκα για τη στήριξη και το χρόνο που αφιέρωσε για τα προβλήματα και τα θέματα που ανέκυψαν σε όλη τη διάρκεια της εκπόνησης. Η άψογη συνεργασία μας βοήθησε στο να ξεπεραστούν όλες οι δυσκολίες ώστε να παρουσιαστεί μία ολοκληρωμένη εργασία.

Περίληψη

Η γλώσσα XML αποτελεί το νέο πρότυπο ανταλλαγής δεδομένων στο διαδίκτυο. Επιτρέπει την καταγραφή και οργάνωση της πληροφορίας τόσο ως προς το περιεχόμενό της όσο και ως προς δομή της. Το μοντέλο δεδομένων της XML εμφανίζεται στη γενική περίπτωση με τη μορφή γράφου, ωστόσο υπό συγκεκριμένες προϋποθέσεις έχει τη μορφή δέντρου. Πολλοί κόμβοι του παγκόσμιου ιστού, π.χ. ηλεκτρονικά καταστήματα, πύλες πληροφόρησης, κ.λ.π., προσφέρουν πληροφορίες στο χρήστη οργανωμένες σε θεματικές δενδρικές ιεραρχίες χρησιμοποιώντας το πρότυπο της XML. Στόχος της εργασίας αυτής είναι ο ορισμός και η ανάπτυξη της γλώσσας ερωτήσεων TreeSQuerL για δενδρικές μορφές XML δεδομένων. Η προτεινόμενη γλώσσα TreeSQuerL περιέχει τελεστές αντίστοιχους με αυτούς που ορίζονται στη σχεσιακή άλγεβρα με τη διαφορά ότι δεν εφαρμόζονται σε σχέσεις, αλλά σε δέντρα. Η αποτίμηση των ερωτήσεων γίνεται σε ιεραρχίες και το αποτέλεσμά τους είναι ένα σύνολο μονοπατιών τα οποία καταλήγουν σε σύνολα εγγραφών από δεδομένα. Στις ερωτήσεις της γλώσσας TreeSQuerL μπορούν επίσης να χρησιμοποιηθούν φίλτρα σχεσιακής επιλογής και προβολής για τις εγγραφές των δεδομένων.

Λέξεις Κλειδιά:

Δέντρα, ιεραρχίες, μονοπάτια, αντικείμενα, σχέσεις, επιλογή, προβολή, ένωση, τομή, διαφορά, γινόμενο.

Abstract

The XML language is becoming the new standard for exchanging data through internet. It records and organizes information based on its content and on its structure. In general, the XML data model is a graph, but under certain conditions becomes a tree. Many nodes in the web information space, i.e. e-shops, information gates, portals etc., provide information to users, organized in thematic hierarchies using XML standard. This thesis defines and develops the query language TreeSQuerL for tree-structure XML data. The suggesting language TreeSQuerL contains operators similar to those defined in relational algebra, with the difference that TreeSQuerL uses trees and not relations. The queries are evaluated over the hierarchy and their result is a set of paths to sets of data records. With TreeSQuerL's queries, filters of relational selection and projection can be used on these data records.

Keywords:

Trees, hierarchies, paths, items, relations, selection, project, product, union, intersection, difference.

Πίνακας περιεχομένων

1	Εισαγωγή.....	1
1.1	Αντικείμενο της διπλωματικής.....	2
1.2	Οργάνωση του τόμου.....	3
2	Περιγραφή Θέματος.....	5
2.1	Σχετικές εργασίες.....	5
2.1.1	Γλώσσα ερωτήσεων XPath.....	5
2.1.2	Γλώσσα ερωτήσεων XQuery.....	7
2.1.3	Άλγεβρα TAX.....	9
2.2	Στόχος.....	13
3	Θεωρητική Μελέτη.....	15
3.1	Δικτυακός κατάλογος – Tree Structure Relation.....	15
3.2	Rp – Ri μοντέλο.....	18
3.2.1	Περιγραφή μοντέλου.....	18
3.2.2	Κατασκευή – Μετάβαση από σχήμα TSR σε μοντέλο Rp - Ri.....	18
3.3	Σύγκριση μονοπατιών.....	20
3.3.1	Ισότητα – Ανισότητα.....	20
3.3.2	Αυστηρό υποσύνολο.....	20
3.3.3	Χαλαρό υποσύνολο.....	21
3.4	Ερωτήσεις – Πράξεις.....	21
3.4.1	Βασικοί τελεστές.....	21
3.4.1.1	Επιλογή (σ).....	21
3.4.1.2	Προβολή (π).....	22
3.4.1.3	Καρτεσιανό Γινόμενο (X).....	23
3.4.1.4	Ένωση (U).....	24
3.4.1.5	Τομή (\cap).....	26
3.4.1.6	Διαφορά ($-$).....	26
3.4.2	Σύνθετοι τελεστές.....	26

3.4.2.1	Σύνδεση (⊗).....	26
4	Ανάλυση και σχεδίαση	29
4.1	Ανάλυση – Περιγραφή Αρχιτεκτονικής	29
4.1.1	Διαχωρισμός υποσυστημάτων	29
4.1.2	Περιγραφή υποσυστημάτων.....	30
4.1.2.1	Υποσύστημα ανάγνωσης σχημάτων TSR	30
4.1.2.2	Υποσύστημα αποθήκευσης σχημάτων TSR.....	31
4.1.2.3	Υποσύστημα ερωτήσεων – πράξεων.....	31
4.1.2.4	Υποσύστημα διαπροσωπείας χρήστη	32
4.1.2.5	Υποσύστημα απεικόνισης σχημάτων TSR.....	33
4.2	Σχεδίαση του συστήματος	34
4.2.1	Εφαρμογές.....	34
4.2.1.1	Διαχειριστής XML αρχείου – σχήματος	35
4.2.1.2	Διαχειριστής βάσης δεδομένων.....	37
4.2.1.3	Εφαρμογή εκτέλεσης – αποτίμησης ερωτήσεων γλώσσας TreeSQuerL	38
4.2.1.4	Μεταγλωττιστής (Interpreter)	39
4.2.1.5	Γραφικό περιβάλλον παρουσίασης αποτελεσμάτων TSR σχημάτων	40
5	Υλοποίηση	43
5.1	Πλατφόρμες και προγραμματιστικά εργαλεία.....	43
5.2	Λεπτομέρειες υλοποίησης	44
5.2.1	Βάση δεδομένων.....	44
5.2.2	Μεταγλωττιστής.....	45
5.2.2.1	Γραμματική γλώσσας TreeSQuerL	45
5.2.2.2	Συντακτική ανάλυση – Χειρισμός εντολών	48
5.2.3	Περιγραφή κλάσεων	48
5.2.3.1	public class Attribute.....	48
5.2.3.2	public class Path.....	49
5.2.3.3	public class OR	50
5.2.3.4	public class Item.....	51
5.2.3.5	public class TSR.....	52
5.2.3.6	public class Condition	54
5.2.3.7	public class TSRfunctions.....	55
5.2.3.8	public class XMLschema	56
5.2.3.9	public class DBschema.....	57

5.2.3.10	public class TSRxmlDialog	58
5.2.3.11	public class TSRdbDialog.....	60
5.2.3.12	public class TSRviewer.....	61
5.2.3.13	public class HelpViewer	62
5.2.4	Αλγόριθμοι	63
5.2.4.1	Ανάγνωση TSR σχήματος από τη βάση δεδομένων.....	65
5.2.4.2	Ανάγνωση TSR σχήματος από XML αρχείο.....	66
5.2.4.3	Αποθήκευση TSR σχήματος στη βάση δεδομένων	66
5.2.4.4	Αποθήκευση TSR σχήματος σε XML αρχείο	67
5.2.4.5	Επιλογή – Select.....	67
5.2.4.6	Προβολή – Project	71
5.2.4.7	Καρτεσιανό γινόμενο – Cartesian Product	72
5.2.4.8	Ένωση – Union	74
5.2.4.9	Τομή – Intersection	75
5.2.4.10	Διαφορά – Difference	76
5.2.4.11	Αυστηρό υποσύνολο.....	77
5.2.4.12	Χαλαρό υποσύνολο.....	77
6	Έλεγχος.....	79
6.1	Μεθοδολογία Ελέγχου.....	79
6.2	Αναλυτική παρουσίαση έλεγχου	79
7	Επίλογος	85
7.1	Σύνοψη και συμπεράσματα	85
7.2	Μελλοντικές επεκτάσεις.....	86
8	Βιβλιογραφία	87

1

Εισαγωγή

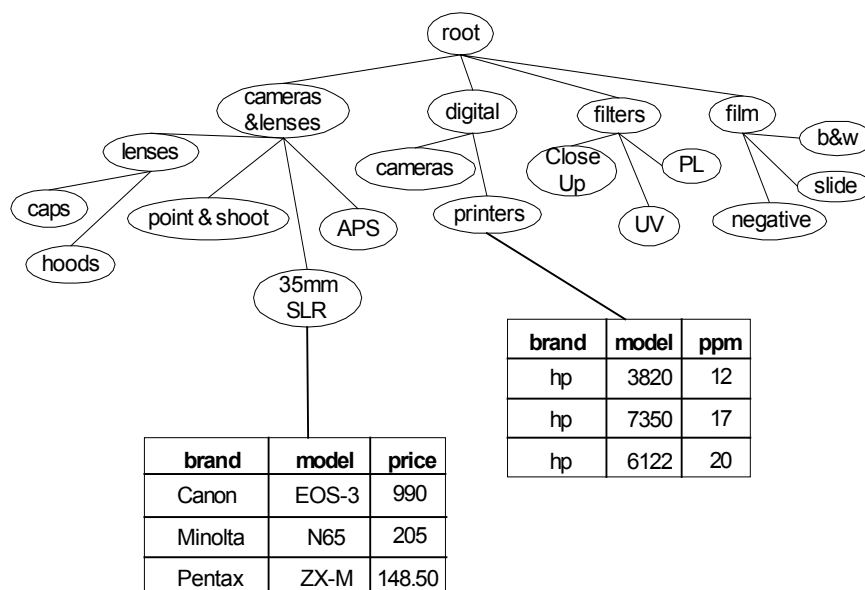
Το διαδίκτυο (internet) προσφέρει ένα ενιαίο και απλό τρόπο ανταλλαγής και μετάδοσης πληροφοριών. Η διαχείριση της πληροφορίας που μεταφέρεται μέσω του διαδικτύου προϋποθέτει τρεις παράγοντες: τα δεδομένα, τη δομή με την οποία οργανώνονται και ένα μηχανισμό για την ανάκτησή τους. Όσον αφορά τη δομή, τα δεδομένα που μεταφέρονται είναι οργανωμένα σε ιεραρχίες και σχήματα που περιγράφονται με τον όρο ημιδομημένα μοντέλα οργάνωσης της πληροφορίας. Το τρίτο στοιχείο περιγράφει τις γλώσσες ερωτήσεων που αποτιμώνται πάνω στα δεδομένα ή και στη δομή της πληροφορίας, αποτελώντας τα εργαλεία επεξεργασίας της πληροφορίας.

Η χρήση του μοντέλου της γλώσσας XML (www.w3.org/xml) παρέχει ένα πρότυπο για τη μετάδοση της πληροφορίας, λαμβάνοντας υπόψη τους παραπάνω πρώτους δύο παράγοντες. Το μοντέλο αυτό είναι στη γενική του μορφή ένας γράφος που σύμφωνα με κατάλληλες προϋποθέσεις μπορεί να πάρει τη μορφή δέντρου. Πρόκειται για ένα μοντέλο που κατορθώνει να παντρεύει την χαλαρότητα των απαιτήσεων των ημιδομημένων δεδομένων με τους αυστηρούς περιορισμούς των καλά δομημένων συστημάτων βάσεων δεδομένων. Η XML προσφέρει ένα πρότυπο επικοινωνίας και ανταλλαγής πληροφοριών μεταξύ ετερογενών κόμβων πληροφοριών του διαδικτύου και αποτελεί πλέον standard στο διαδίκτυο.

1.1 Αντικείμενο της διπλωματικής

Το αντικείμενο της διπλωματικής εργασίας αυτής είναι ο τρίτος παράγοντας του χώρου της διαχείρισης της πληροφορίας στο διαδίκτυο και συγκεκριμένα ένας νέος μηχανισμός ανάκτησης δεδομένων με τη γλώσσα ερωτήσεων TreeSQuerL¹. Η TreeSQuerL αποτελείται από σύνολο ερωτήσεων ανάλογων της σχεσιακής άλγεβρας, οι οποίες όμως αποτιμώνται σε δενδρικές δομές δεδομένων, δίνοντας ως αποτέλεσμα ένα σύνολο μονοπατιών που οδηγούν σε εγγραφές δεδομένων.

Ας φανταστούμε ένα δικτυακό τόπο με μεγάλο όγκο δεδομένων συγκεκριμένου αντικειμένου ή τομέα. Ένας τέτοιος τόπος ονομάζεται *κάθετος δικτυακός κατάλογος*. Χρησιμοποιώντας το μοντέλο της XML, ο τεράστιος όγκος πληροφοριών του καταλόγου οργανώνεται με συγκεκριμένη ιεραρχία, προσθέτοντας ταυτόχρονα το στοιχείο της δομικής πληροφορίας που δεν υπάρχει στις κλασικές βάσεις δεδομένων. Στο Σχ. 1.1 παρουσιάζεται η ιεραρχία για τον κάθετο κατάλογο παρουσίασης φωτογραφικού υλικού adorama (www.adorama.com).



- Σχ. 1.1 -

Στα φύλλα της ιεραρχίας αυτής βρίσκονται τοποθετημένες ομάδες εγγραφών με τα χαρακτηριστικά τους για κάθε αντίστοιχο προϊόν με τη μορφή των κλασικών σχέσεων.

Χρησιμοποιώντας γλώσσες ερωτήσεων ημιδομημένων δεδομένων (π.χ. XQuery, XPath) στην παραπάνω ιεραρχία μπορούν να απαντηθούν ερωτήματα όπως:

¹ Η λέξη TreeSQuerL προέρχεται από τις λέξεις Tree Structure Query Language και βασίζεται στη έκφραση tree squirrel, δηλαδή σκίουρος του δέντρου.

- Φέρτε όλα τα UV φίλτρα που βρίσκονται κάτω ακριβώς από τον κόμβο *filter* της ρίζας.
- Βρες τους ψηφιακούς εκτυπωτές, κάτω από τη ρίζα σε οποιοδήποτε βάθος, με περισσότερες από 15 σελίδες ανά λεπτό.

Με τη γλώσσα TreeSQuerL μπορούν να εκφραστούν ερωτήματα πάνω στα μονοπάτια της ιεραρχίας αλλά και ερωτήσεις σχεσιακής επιλογής και προβολής στις εγγραφές. Επίσης δίνεται η δυνατότητα συνδυασμού δύο ιεραρχιών με τις πράξεις της ένωσης, της τομής, της διαφοράς, αλλά και του καρτεσιανού γινομένου, παράγοντας έτσι μια καινούρια ιεραρχία με τις επιθυμητές εγγραφές προϊόντων και τα επιθυμητά μονοπάτια που καταλήγουν σ' αυτές.

Έτσι με χρήση της TreeSQuerL γίνονται ερωτήματα όπως:

- (Από ένα σχήμα) Φέρτε όλες τις εγγραφές των φακών με προϋπόθεση ότι τα μονοπάτια που οδηγούν σ' αυτούς είναι τμήματα του μονοπατιού *root/lenses/caps*, καθώς και τα μονοπάτια αυτά.
- (Από πολλά σχήματα) Φέρτε τις εγγραφές των εκτυπωτών με την προϋπόθεση ότι τα μονοπάτια που οδηγούν σ' αυτούς περιέχουν το μονοπάτι */digital/printers*, καθώς και τα μονοπάτια αυτά.

που όμως δεν μπορούν να εκφραστούν σε υπάρχουσες γλώσσες ημιδομημένων δεδομένων.

1.2 Οργάνωση του τόμου

Η εργασία αυτή χωρίζεται σε 8 κεφάλαια. Στο 2^ο κεφάλαιο γίνεται μια σύντομη παρουσίαση εργασιών με συναφές θέμα. Στο 3^ο κεφάλαιο παρουσιάζεται η θεωρητική μελέτη του θέματος της εργασίας. Δίνονται οι αντίστοιχοι ορισμοί των βασικών εννοιών, παρουσιάζονται τα μοντέλα που χρησιμοποιούνται και ορίζονται οι πράξεις της γλώσσας TreeSQuerL. Στη συνέχεια, το 4^ο κεφάλαιο ασχολείται με την αρχιτεκτονική του συστήματος που υλοποιεί τη γλώσσα TreeSQuerL. Δίνονται τα ανεξάρτητα υποσυστήματα που το αποτελούν και οι λειτουργίες τους, καθώς και οι εφαρμογές που υλοποιούν το καθένα απ' αυτά. Οι λεπτομέρειες υλοποίησης του συστήματος παρουσιάζονται στο 5^ο κεφάλαιο μαζί με τα προγραμματιστικά εργαλεία και τις τεχνολογίες που χρησιμοποιήθηκαν. Παράλληλα παρατίθενται τμήματα του κώδικα του συστήματος και οι βασικοί του αλγόριθμοι. Το 6^ο κεφάλαιο βοηθάει στην καλύτερη κατανόηση του συστήματος που υλοποιεί τη γλώσσα TreeSQuerL, δίνοντας κατάλληλα παραδείγματα εκτέλεσης. Στο 7^ο κεφάλαιο επιχειρείται μια σύνοψη της εργασίας και παρουσιάζονται μελλοντικές επεκτάσεις και θέματα γύρω από τη γλώσσα TreeSQuerL. Τέλος, το 8^ο κεφάλαιο παρουσιάζει τη σχετική βιβλιογραφία.

2

Περιγραφή Θέματος

Στο κεφάλαιο αυτό παρουσιάζεται το θεωρητικό υπόβαθρο της διπλωματικής. Αναφέρονται ερευνητικές εργασίες και συναφείς τεχνολογίες. Τέλος παρουσιάζεται η προσφορά της παρούσας μελέτης σε σχέση με αυτές τις εργασίες.

2.1 Σχετικές εργασίες

Στην ενότητα αυτή περιγράφονται περιληπτικά τα θέματα που καλύπτουν οι εργασίες:

- Γλώσσα ερωτήσεων XPath
- Γλώσσα ερωτήσεων XQuery
- Άλγεβρα TAX

2.1.1 Γλώσσα ερωτήσεων XPath

Η γλώσσα XPath [XPATH] δημιουργήθηκε για να προσφέρει συντακτικό και σημασιολογία στις λειτουργίες που ορίζουν οι XSLT [XSLT] μετασχηματισμοί και οι XPointers [XPTR]. Βασική λειτουργία της είναι η αναγνώριση συγκεκριμένων τμημάτων XML εγγράφων. Εντοπίζονται κόμβοι με βάση τη θέση τους στο έγγραφο, τον τύπο, το περιεχόμενο ή οποιαδήποτε άλλο κριτήριο. Επίσης, προσφέρεται η δυνατότητα αναπαράστασης και διαχείρισης αλφαριθμητικών γραμματοσειρών αλλά και αριθμητικών με αποτέλεσμα να είναι εφικτή η εφαρμογή και η αποτίμηση απλών μαθηματικών εκφράσεων μέσω των ερωτήσεων.

Η σύνταξη της XPath ορίζει το XML έγγραφο ως ένα δέντρο με τα εξής δομικά στοιχεία:

- ρίζα (root node)
- κόμβος – αντικείμενο (element node)
- κόμβος κειμένου (text node)
- κόμβος – ιδιότητα (attribute node)
- κόμβος – σχόλιο (comment node)
- κόμβος επεξεργασίας (processing – instruction node)
- κόμβος ονομασίας (namespace node)

Η βασικότερη έκφραση της γλώσσας XPath είναι αυτή του μονοπατιού.

Ορισμός 2.1. *Μονοπάτι (location path) είναι ένα σύνολο από έναν ή περισσότερους κόμβους – δομικά στοιχεία του XML εγγράφου.*

Οι κόμβοι του μονοπατιού ξεχωρίζονται από το χαρακτήρα / (slash) που χρησιμοποιείται και για το συμβολισμό της ρίζας. Στην περίπτωση που ένας κόμβος του μονοπατιού είναι κόμβος – ιδιότητα τότε χρησιμοποιείται το πρόθεμα @.

Έτσι έστω για παράδειγμα το portal.xml έγγραφο που ορίζει τα περιεχόμενα ενός δικτυακού τόπου που προσφέρει πληροφορίες και πουλάει φωτογραφικό υλικό:

```
<portal>
  <cameras>
    <digital>
      <cam brand="Canon" model="A60" price="550"></cam>
    </digital>
    <SLR>
      <cam brand="Canon" model="EOS-3" price="980"></cam>
    </SLR>
  </cameras>
</portal>
```

Ένα μονοπάτι προσπέλασης όλων των ψηφιακών μηχανών είναι:

```
/cameras/digital
```

Ενώ για να προσπελαστεί η ιδιότητα τιμή κάθε SLR μηχανής, το μονοπάτι είναι:

```
/cameras/SLR/cam/@price
```

Ωστόσο επειδή τις περισσότερες φορές αυτό που χρειάζεται για την επεξεργασία ενός XML εγγράφου είναι η επιλογή των κόμβων εκείνων που ικανοποιούν συγκεκριμένες συνθήκες, η XPath ορίζει προθέματα σε κάθε κόμβο που περιλαμβάνεται στο μονοπάτι.

Ορισμός 2.2. *Πρόθεμα (Predicate) είναι μια Boolean έκφραση που αποτιμάται σε ένα κόμβο.*

Παράδειγμα:

Στο σχήμα του δικτυακού τόπου που περιγράφεται στο έγγραφο portal.xml του προηγούμενου παραδείγματος, για να επιλεγθούν οι SLR κάμερες μάρκας Canon με τιμή μεγαλύτερη από 100 Ευρώ συντάσσεται η XPath έκφραση

```
/cameras/SLR/cam[@brand = 'Canon' and @price >= 100]
```

2.1.2 Γλώσσα ερωτήσεων XQuery

Η XQuery [XQUERY] είναι μια συναρτησιακή γλώσσα που δημιουργήθηκε από το W3C (www.w3c.org) ως γλώσσα ερωτήσεων XML δεδομένων. Αποτελεί υπερσύνολο της γλώσσας XPath που παρουσιάστηκε στην προηγούμενη παράγραφο και κατά συνέπεια χρησιμοποιεί και επεκτείνει τη λειτουργικότητα αυτής.

Το μοντέλο δεδομένων της XQuery στηρίζεται στο αντίστοιχο μοντέλο της XPath για το XML έγγραφο. Ως βασικές έννοιές του ορίζονται ο *κόμβος*, το *αντικείμενο*, η *ατομική τιμή* και η *ακολουθία*. Ο κόμβος (node) ορίζεται αντίστοιχα με την έννοια του κόμβου στη γλώσσα XPath, ενώ η ατομική τιμή (atomic value) μπορεί να είναι ένας από τους βασικούς τύπους δεδομένων της γλώσσας XQuery: συμβολοακολουθία (string), ακέραιος (integer), δεκαδικός (decimal), ημερομηνία (date).

Ορισμός 2.3. *Αντικείμενο (item) ονομάζεται ένας κόμβος ή μία ατομική τιμή.*

Ορισμός 2.4. *Ακολουθία (sequence) είναι μια ταξινομημένη συλλογή από ένα ή περισσότερα αντικείμενα.*

Κατά αντιστοιχία με τις εκφράσεις της XPath ορίζονται οι εκφράσεις της γλώσσας XQuery καθώς και η έννοια του *προθέματος*.

Ορισμός 2.5. *Πρόθεμα είναι μία έκφραση που περιέχεται σε [] και χρησιμοποιείται στο φιλτράρισμα ακολουθιών τιμών.*

Με βάση τις παραπάνω βασικές έννοιες της γλώσσας το μοντέλο δεδομένων βασίζεται σε μία ετερογενή ακολουθία κόμβων ή ατομικών τιμών. Στα Σχ. 2.1 και 2.2 βλέπουμε δύο μοντέλα δεδομένων από τα items.xml και bids.xml αρχεία αντίστοιχα. Οι κόμβοι D, E, A και T αναπαριστούν το έγγραφο, αντικείμενο, ιδιότητα και κόμβο – κείμενο αντίστοιχα.

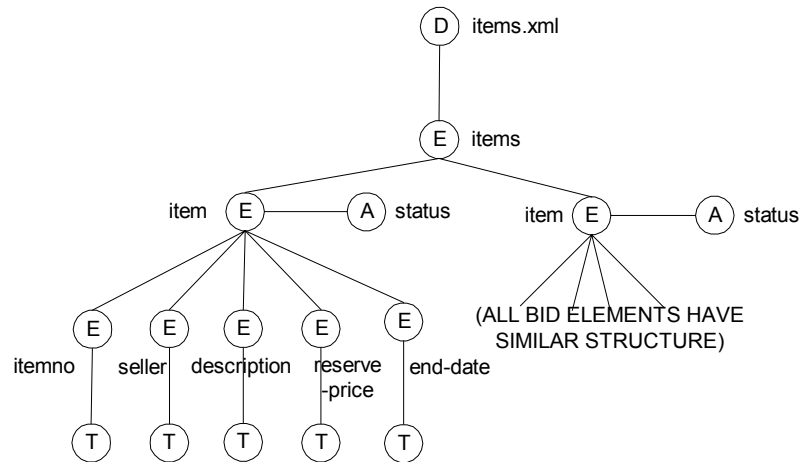
Ένα απλό ερώτημα σε XQuery θα μπορούσε να επιλέγει από το έγγραφο items.xml όλα τα αντικείμενα που πωλούνται από ένα άτομο με όνομα Smith:

```
document("items.xml")/*/*item[seller = "Smith"]/description
```

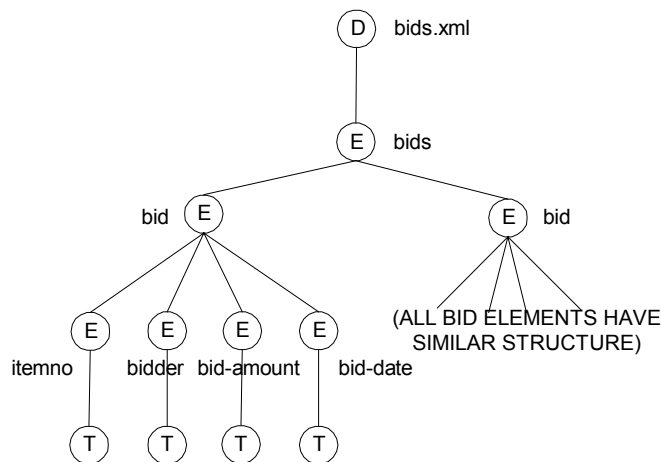
Η XQuery εκτός από τη δυνατότητα επιλογής κόμβων και αντικειμένων από ένα μοντέλο δεδομένων, προσφέρει και τη δυνατότητα κατασκευής νέων κόμβων. Για παράδειγμα με το παρακάτω ερώτημα κατασκευάζεται ένα νέο αντικείμενο με όνομα highbid που έχει τη μεγαλύτερη προσφορά. Οι τιμές της ιδιότητας status και των κόμβων – κειμένου itemno και bid-amount δεν είναι σταθερές, αλλά προκύπτουν από το αρχικό μοντέλο του Σχ. 2.2

μέσω των μεταβλητών $\$s$, $\$i$ που αντιστοιχούν στην ιδιότητα `status` και στο αντικείμενο `item` αντίστοιχα.

```
<highbid status = "{$s}">
  <itemno> {$i} </itemno>
  <bid-amount>
    {max($bids[itemno = $i]/bid-amount)}
  </bid-amount>
</highbid>
```



- Σχ. 2.1 (Από κείμενο [Cha02]) -



- Σχ. 2.2 (Από κείμενο [Cha02]) -

Βασική λειτουργία που προσφέρει η γλώσσα XQuery είναι οι εκφράσεις FLWR, που περιέχουν επαναληπτικό βρόχο (For), έκφραση ανάθεσης (Let), έκφραση συνθήκης (Where) και επιστροφής – κατασκευής αποτελέσματος (Return). Έστω για παράδειγμα ότι θέλουμε για κάθε αντικείμενο του Σχ. 2.1 που περιέχει πάνω από δέκα προσφορές, που παρουσιάζονται στο Σχ. 2.2, να κατασκευάσουμε ένα νέο αντικείμενο με όνομα `popular-item` που περιέχει τον αριθμό του (`itemno`), την περιγραφή (`description`) και το πλήθος των προσφορών. Συντάσσεται η παρακάτω ερώτηση:

```

for $i in document("items.xml")/*/*item
let $b : = document("bids.xml")
    /*/*bid[itemno = $i/itemno]
where count($b) > 10
return
<popular-item>
  (
    $i/itemno
    $i/description
    <bid-count>( count ($b) )</bid-count>
  )
</popular-item>

```

Στο παραπάνω ερώτημα ορίζουμε ως μεταβλητή για κάθε αντικείμενο του `items.xml` τη `$i` και μεταβλητή κάθε προσφοράς από το `bids.xml` το `$b`. Έτσι οι εκφράσεις `$i/itemno` και `$i/description` δίνουν τον αριθμό και την περιγραφή κάθε αντικειμένου. Η συνάρτηση `count` μετράει το πλήθος των προσφορών `$b` κάθε αντικειμένου. Τέλος, η έκφραση `where` εξασφαλίζει ότι το αντικείμενο `popular-item` κατασκευάζεται μόνο όταν οι προσφορές για το `$i` είναι τουλάχιστον 10.

2.1.3 Άλγεβρα TAX

Η TAX [JLS+01] είναι μία άλγεβρα για το χειρισμό και την επεξεργασία XML δεδομένων που είναι οργανωμένα σε δάση και δενδρικές μορφές δεδομένων. Είναι μια προσπάθεια επέκτασης της σχεσιακής άλγεβρας ώστε να διαχειρίζεται δέντρα και όχι σχέσεις. Επίσης η TAX είναι σχεσιακά πλήρης και μπορεί να εκφράσει ερωτήσεις των βασικών γλωσσών ερωτήσεων XML δεδομένων, συμπεριλαμβανομένου και της XQuery που περιγράφεται στην ενότητα 2.1.2. Δεδομένου ότι η άλγεβρα TAX ορίζεται σε αντιστοιχία με τη σχεσιακή άλγεβρα, όπως στο σχεσιακό μοντέλο βασική έννοια αποτελεί η πλειάδα, σ' αυτήν υπάρχει το δέντρο δεδομένων.

Ορισμός 2.6. *Δέντρο δεδομένων (data tree) είναι ένα ταξινομημένο δέντρο με βασικά συστατικά μία ρίζα και ονομασμένους κόμβους, που καθένας τους έχει δεδομένα σε μορφή ιδιοτήτων (attributes).*

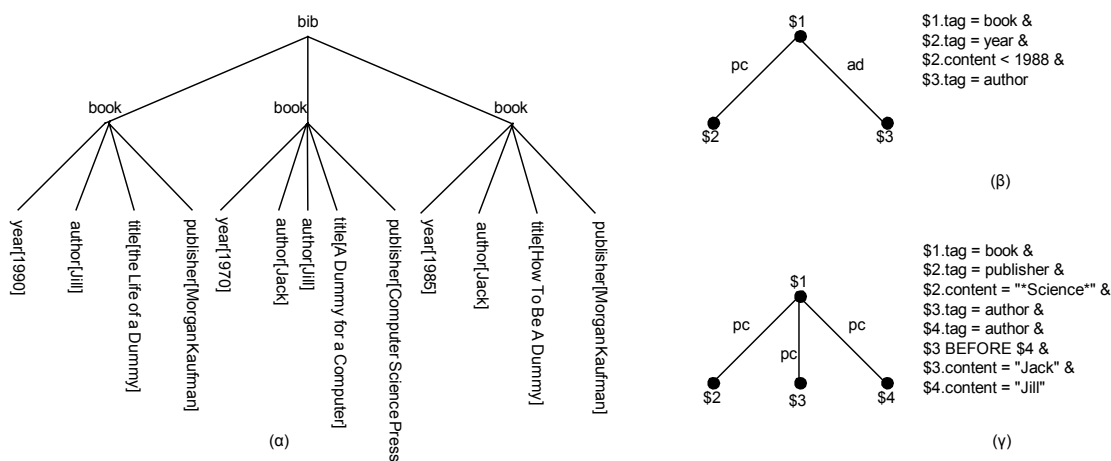
Κατά αντιστοιχία με τη σχεσιακή άλγεβρα ορίζεται η έννοια της συλλογής (collection) δέντρων δεδομένων αντίστοιχα με την έννοια της σχέσης. Έτσι, όπως η σχεσιακή βάση δεδομένων είναι ένα σύνολο σχέσεων, μία XML βάση δεδομένων είναι ένα σύνολο συλλογών δέντρων δεδομένων. Τέλος κατ' αντιστοιχία με τους σχεσιακούς τελεστές, οι τελεστές της TAX δεν αποτιμώνται σε σχέσεις, αλλά σε συλλογές δέντρων δεδομένων.

Ορισμός 2.7. *Δέντρο – πρότυπο (Pattern Tree) ορίζεται ως ένα ζευγάρι $P = (T, F)$, όπου $T = (V, E)$ είναι ένα δέντρο με ονομασμένους κόμβους V και ακμές E , τέτοιο ώστε:*

- Κάθε κόμβος του V έχει ένα μοναδικό ακέραιο ως αναγνωριστικό του.

- Κάθε ακμή ονομάζεται είτε *pc* (πατέρας – παιδί) είτε *ad* (πρόγονος – απόγονος).
- Η *F* είναι μία φόρμουλα, όπως συνδυασμός προθεμάτων στους κόμβους.

Στο Σχ. 2.3 (α) βλέπουμε ένα παράδειγμα XML βάσης δεδομένων και στο (β) ένα πρότυπο που βρίσκει τα βιβλία που εκδόθηκαν πριν το 1988 και έχουν ένα τουλάχιστον συγγραφέα. Στο Σχ. 2.3 (γ) παρουσιάζεται το δέντρο – πρότυπο που δείχνει τα βιβλία που εκδόθηκαν από εκδοτικό οίκο με όνομα που περιέχει το ‘Science’.



- Σχ. 2.3 (Από κείμενο [JLS+01]) -

Ορισμός 2.8. Αν C είναι μια συλλογή δέντρων δεδομένων, $P = (T, F)$ είναι ένα δέντρο – πρότυπο και $h : P \rightarrow C$ είναι μία απεικόνιση του δέντρου – προτύπου P στη συλλογή C τότε δέντρο – μάρτυρας (*Witness Tree*) $h^c(P)$ ορίζεται μια συσχέτιση με το P ως εξής:

- Κάθε κόμβος n της C υπάρχει στο δέντρο – μάρτυρας αν $n = h(u)$ για κάποιο κόμβο u του δέντρου – προτύπου P .
- Για κάθε ζευγάρι κόμβων n, m του δέντρου – μάρτυρα, όταν ο m είναι ο κοντινότερος πρόγονος του n στη C μεταξύ των κόμβων που υπάρχουν στο δέντρο – μάρτυρας, τότε το δέντρο – μάρτυρας περιέχει την ακμή (m, n) .
- Το δέντρο – μάρτυρας διατηρεί τη σειρά των κόμβων που υπάρχει στη συλλογή δέντρων δεδομένων C .

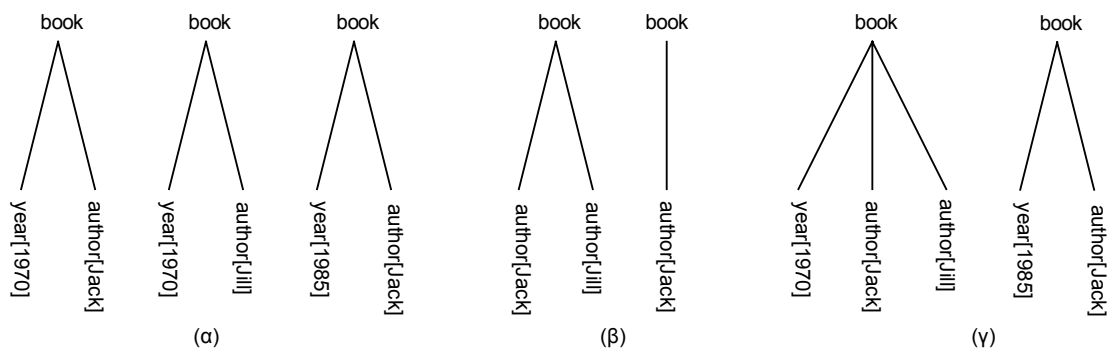
Για παράδειγμα στο Σχ. 2.4 (α), (β), (γ) φαίνονται δέντρα – μάρτυρες που μπορούν να προκύψουν από το δέντρο του Σχ. 2.3 (α).

Έχοντας ορίσει τις βασικές έννοιες του δέντρου – προτύπου και του δέντρου – μάρτυρα ορίζονται οι βασικοί τελεστές της επιλογής, προβολής και καρτεσιανού γινομένου.

Η επιλογή $\sigma_{P,SL}(C)$ δέχεται ως είσοδο μία συλλογή C , ένα δέντρο – πρότυπο P και μία λίστα παραμέτρων SL και επιστρέφει μία νέα συλλογή δέντρων δεδομένων σύμφωνα με τα παρακάτω:

- Ένας κόμβος u της συλλογής C ανήκει στο αποτέλεσμα αν u απεικονίζεται μέσω της $h : P \rightarrow C$ σ' ένα κόμβο του P ή u είναι απόγονος ενός κόμβου v της C , ο οποίος απεικονίζεται μέσω της h σ' ένα κόμβο w του P , που το αναγνωριστικό του περιέχεται στη λίστα SL .
- Αν δύο κόμβοι u, v περιέχονται στο αποτέλεσμα και u είναι ο κοντινότερος πρόγονος του v στη συλλογή C , τότε το αποτέλεσμα περιέχει και την ακμή (u, v) .
- Η σειρά των κόμβων στη συλλογή εισόδου C διατηρείται και στο αποτέλεσμα.

Στο Σχ. 2.4 (α) φαίνεται η επιλογή στη συλλογή του Σχ. 2.3 (α) με δέντρο – πρότυπο το Σχ. 2.3 (β) και λίστα παραμέτρων κενή.



- Σχ. 2.4 (Από κείμενο [JLS+01]) -

Η προβολή $\pi_{P,SL}(C)$ δέχεται ως είσοδο μία συλλογή δέντρων δεδομένων C , ένα δέντρο πρότυπο P και μία λίστα παραμέτρων SL και παράγει μία νέα συλλογή σύμφωνα με τα εξής:

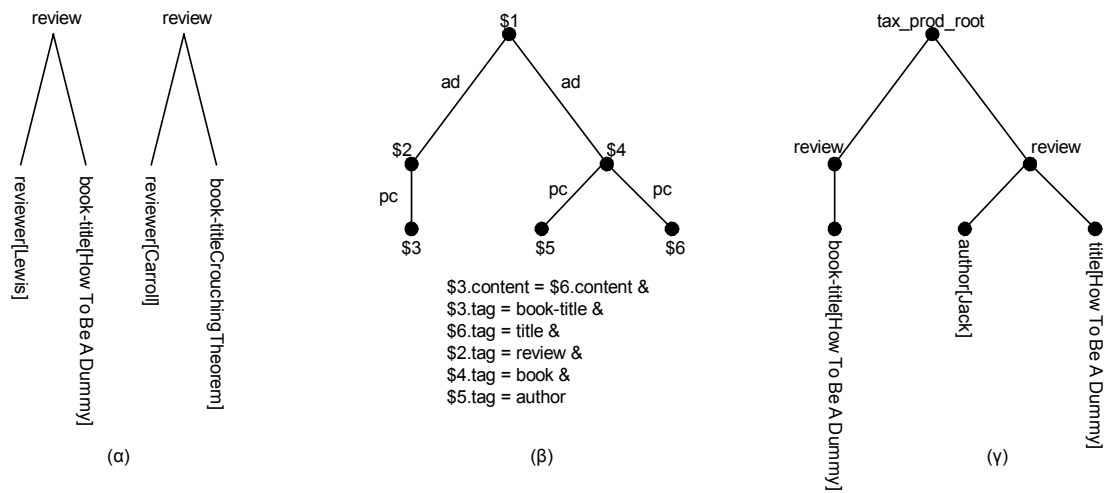
- Ένας κόμβος u της C ανήκει στο αποτέλεσμα αν υπάρχει απεικόνιση $h : P \rightarrow C$ τέτοια ώστε u απεικονίζεται σ' ένα κόμβο του P του οποίου το αναγνωριστικό περιέχεται στη SL ή u είναι απόγονος ενός κόμβου v της C , ο οποίος απεικονίζεται σ' ένα κόμβο w του P , που έχει αναγνωριστικό που περιέχεται στη SL .
- Όταν δύο κόμβοι u, v ανήκουν στο αποτέλεσμα και u είναι ο κοντινότερος πρόγονος του v στη συλλογή εισόδου C , τότε το αποτέλεσμα περιέχει και την ακμή (u, v) .
- Η σειρά των κόμβων στη συλλογή εισόδου C διατηρείται και στο αποτέλεσμα.

Στο Σχ. 2.4 (γ) φαίνεται η επιλογή στη συλλογή του Σχ. 2.3 (α) με δέντρο – πρότυπο το Σχ. 2.3 (β) και λίστα παραμέτρων $\{\$1, \$2, \$3\}$.

Το καρτεσιανό γινόμενο $C \times D$ δέχεται ως εισόδους δύο συλλογές δέντρων δεδομένων C και D και παράγει συνδυάζοντας ζευγάρια δέντρων από το C στο D μία νέα συλλογή σύμφωνα με τα παρακάτω:

- Κάθε ζευγάρι δέντρων δεδομένων $T_i \in C$ και $T_j \in D$ το $C \times D$ περιέχει ένα δέντρο, του οποίου η ρίζα είναι ένας νέος κόμβος με όνομα `tax_product_root` και χωρίς ιδιότητες. Το αριστερό παιδί της ρίζας είναι η ρίζα του T_i και το δεξί του T_j .
- Για κάθε κόμβο στο αριστερό και στο δεξιό υποδέντρο της νέας ρίζας οι ιδιότητες είναι όμοιες μ' αυτές που υπήρχαν στις εισόδους C και D .

Για παράδειγμα το γινόμενο της συλλογής του Σχ. 2.5 (α) μ' αυτήν του Σχ. 2.3 (α) και με επιλογή στο αποτέλεσμα με βάση το δέντρο – πρότυπο του Σχ. 2.5 (β) δίνει τη συλλογή του Σχ. 2.5 (γ).

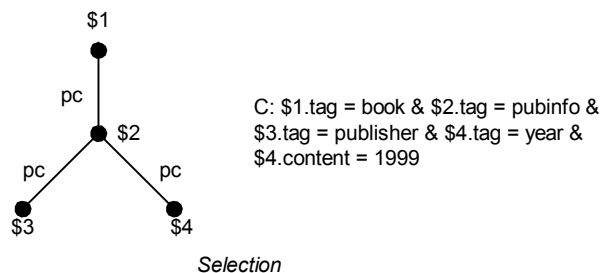


- Σχ. 2.5 (Από κείμενο [JLS+01]) -

Τέλος σύμφωνα με τον ορισμό της TAX είναι δυνατόν να εκφραστούν ερωτήματα που έχουν συνταχθεί σε γλώσσες ερωτήσεων XML δεδομένων, όπως η XQuery. Για παράδειγμα το παρακάτω ερώτημα σε XQuery επιλέγει από ένα XML σχήμα τα βιβλία που εκδόθηκαν το 1999 ανά συγγραφέα.

```
FOR $pub IN DISTINCT //publisher
LET $bk := //book[pubinfo/publisher=$pub AND pubinfo/year="1999"]
RETURN $pub
```

Χρησιμοποιώντας την άλγεβρα TAX και την πράξη της επιλογής μπορούμε να εκτελέσουμε το παραπάνω ερώτημα με το δέντρο – πρότυπο και τη λίστα παραμέτρων του Σχ. 2.6.



- Σχ. 2.6 (Από κείμενο [JLS+01]) -

2.2 Στόχος

Οι εργασίες που παρουσιάστηκαν στην προηγούμενη ενότητα προσφέρουν ένα τρόπο ανάκτησης και επεξεργασίας δεδομένων από XML έγγραφα. Ας φανταστούμε ένα προϊόν που παρουσιάζεται σε διάφορους δικτυακούς καταλόγους. Η προσπέλαση αυτού του προϊόντος μπορεί να γίνεται με διαφορετικούς τρόπους στους καταλόγους αυτούς. Θα θέλαμε να μπορούμε να εκμεταλλευτούμε τη διαφορετικότητα αυτή.

Στόχος της διπλωματικής εργασίας αυτής είναι να συμπεριλάβει και να επεκτείνει τη λειτουργικότητα επιλογής και επεξεργασίας δεδομένων οργανωμένων σε XML δενδρικές μορφές. Με τη γλώσσα TreeSQuerL μπορούν να συνδυαστούν οι διαφορετικοί τρόποι προσπέλασης σε ένα αντικείμενο – προϊόν και να κατασκευαστεί μία καινούρια ιεραρχία, σύνοψη των αρχικών. Με το σύνολο των ερωτήσεων επιλογής, προβολής, καρτεσιανού γινομένου, ένωσης, τομής και διαφοράς που προτείνει η παρούσα διπλωματική, μπορούν να επιλεγούν και να επεξεργαστούν τα δεδομένα και οι τρόποι προσπέλασης (μονοπάτια) ενός προϊόντος από πολλούς δικτυακούς καταλόγους και να παραχθεί μια καινούρια οργάνωση προϊόντων.

3

Θεωρητική Μελέτη

Στο κεφάλαιο αυτό παρουσιάζουμε τους ορισμούς των βασικών δομών της γλώσσας TreeSQuerL, ενώ ταυτόχρονα εισάγεται το μοντέλο Rp-Ri που χρησιμοποιείται στην αναπαράσταση των δομών. Τέλος ορίζεται το σύνολο των πράξεων της άλγεβρας της γλώσσας με τους βασικούς τελεστές: επιλογή, προβολή, καρτεσιανό γινόμενο, ένωση, τομή και διαφορά, καθώς και το σύνθετο τελεστή της σύνδεσης.

3.1 Δικτυακός κατάλογος – Tree Structure Relation

Τα περιεχόμενα ενός δικτυακού καταλόγου οργανώνονται σε μια *ιεραρχία* με μορφή δέντρου που ονομάζεται *σχήμα καταλόγου* (catalog schema). Το σχήμα ενός καταλόγου αποτελείται από ονομασμένους κόμβους που αντιστοιχούν σε θεματικές κατηγορίες και φύλλα που περιέχουν εγγραφές των περιεχόμενων αντικειμένων. Ακολουθεί ο ορισμός του σχήματος καταλόγου:

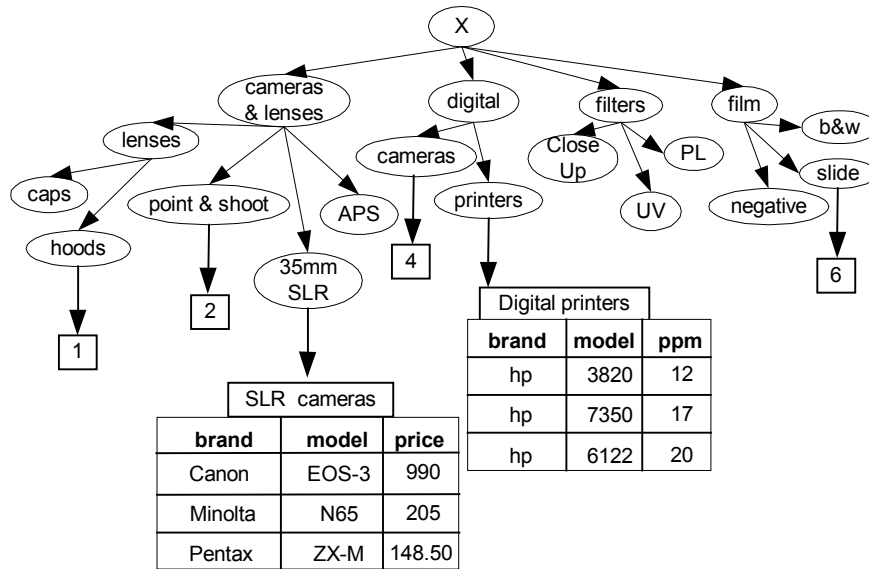
Ορισμός 3.1. Έστω τα παρακάτω δομικά στοιχεία:

- ρίζα (root) \otimes
- κόμβος (property node) \circ
- κόμβος – αντικείμενο (resource node – item) \square . Κάθε τέτοιο αντικείμενο έχει μια σειρά ιδιοτήτων (attributes).

Σχήμα καταλόγου (CS) είναι ένα δέντρο με ρίζα \otimes , κόμβους \circ και φύλλα \square .

$CS := Tree(root, V, R)$, όπου V το σύνολο των κόμβων και R το σύνολο των κόμβων – αντικειμένων.

Στο Σχ. 3.1 φαίνεται το σχήμα του δικτυακού καταλόγου adorama (www. adorama.com) που περιέχει φωτογραφικό εξοπλισμό. Για παράδειγμα, το αντικείμενο SLR cameras περιέχει εγγραφές για κάμερες SLR 35mm με ιδιότητες brand, model και price για το μάρκα, μοντέλο και την τιμή αντίστοιχα.



- Σχ. 3.1 -

Σύμφωνα με τη θεωρία γράφων και δέντρων εισάγουμε την έννοια του μονοπατιού για το σχήμα καταλόγου.

Ορισμός 3.2. Ορίζουμε μονοπάτι του σχήματος καταλόγου μια ακολουθία κόμβων v_0 (ρίζα), v_1, v_2, \dots, v_{k-1} με $k \in \mathbb{N}$ που οδηγούν μέσω ακμών στον κόμβο – αντικείμενο v_k .

Για παράδειγμα στο Σχ. 3.1 για να προσπελάσουμε το αντικείμενο Digital printers με τις εγγραφές των ψηφιακών εκτυπωτών ακολουθούμε το μονοπάτι: /digital/printers

Στη συνέχεια, εισάγουμε την έννοια της δομικής σχέσης TSR (Tree Structure Relation), με την οποία περιγράφουμε και συνδυάζουμε σχήματα καταλόγων.

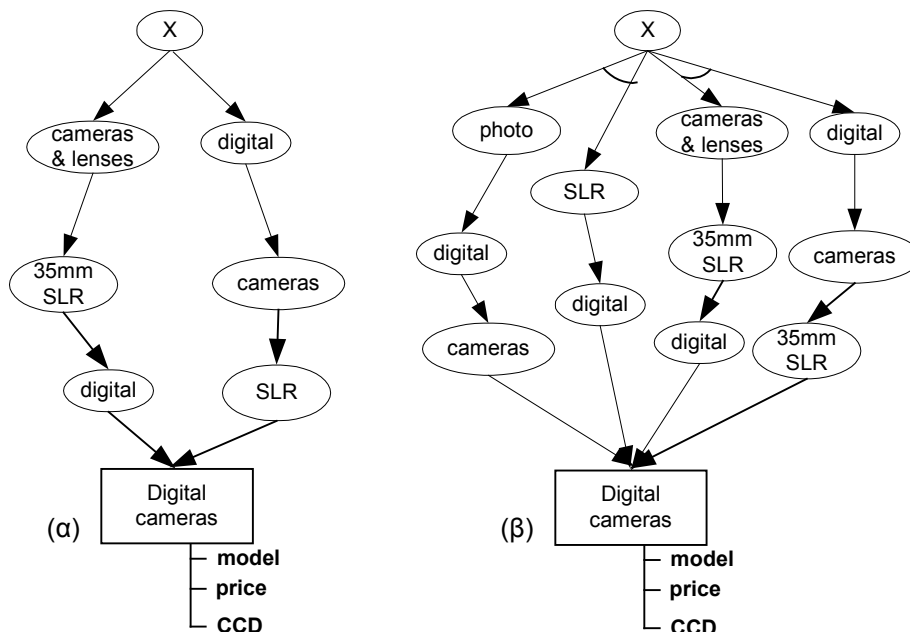
Ορισμός 3.3 . Έστω $S = \{CS_1, CS_2, \dots, CS_n\}$ σύνολο σχημάτων καταλόγων και ένας κόμβος – αντικείμενο που υπάρχει σε όλα τα σχήματα. Ένα σχήμα TSR είναι ένας AND – OR γράφος που αποτελείται από μία ρίζα \otimes , τον κοινό κόμβο – αντικείμενο και συνδυασμούς των μονοπατιών από τα σχήματα του S που καταλήγουν σ' αυτόν.

Σε ένα σχήμα TSR:

1. OR ονομάζονται τα μονοπάτια που αντιστοιχούν στην ύπαρξη περισσότερων του ενός δρόμων προσπέλασης του κόμβου – αντικείμενου
2. Τα AND μονοπάτια ορίζουν μια σύνθετη έννοια κατά αντιστοιχία μ' αυτή που προκύπτει από την εφαρμογή καρτεσιανού γινομένου σε σχέσεις στη σχεσιακή άλγεβρα.
3. OR συνιστώσα ενός σχήματος TSR ονομάζεται ο συνδυασμός των γειτονικών AND και των ανεξάρτητων OR μονοπατιών.

Το Σχ. 3.2 (α) αποτελεί μια εκφυλισμένη μορφή του AND – OR γράφου μιας TSR με δύο OR μονοπάτια που δείχνουν τους διαφορετικούς τρόπους προσπέλασης εγγραφών φωτογραφικών μηχανών. Στην περίπτωση αυτή τα δύο ανεξάρτητα OR μονοπάτια σχηματίζουν τις δύο OR συνιστώσες του σχήματος.

Στο Σχ. 3.2 (β) φαίνεται μια TSR που περιγράφει τα μονοπάτια που οδηγούν σε εγγραφές ψηφιακών φωτογραφικών μηχανών. Ένα τέτοιο σχήμα προέρχεται από συνδυασμό μονοπατιών πολλών σχημάτων καταλόγων που αναφέρονται στο ίδιο κόμβο – αντικείμενο. Τα γειτονικά AND μονοπάτια /photo/digital/cameras και /SLR/digital ορίζουν την πρώτη συνιστώσα της TSR, ενώ τα /cameras&lenses/35mmSLR/digital και /digital/cameras/35mmSLR ορίζουν τη δεύτερη OR συνιστώσα. Ενώνοντας δύο μονοπάτια με ένα τόξο δηλώνουμε ότι πρόκειται για AND μονοπάτια. Το Σχ.3.2 (β) θα γίνει πιο σαφές όταν θα οριστούν οι τελεστές που εφαρμόζονται πάνω σε σχήματα TSR.



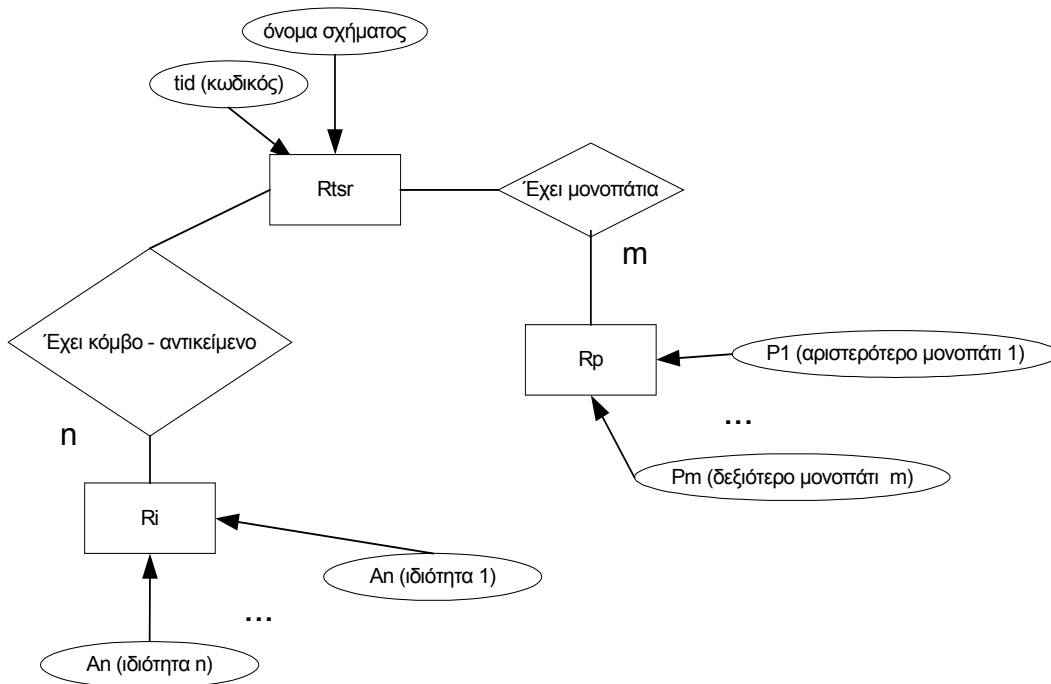
- Σχ. 3.2 -

3.2 $R_p - R_i$ μοντέλο

3.2.1 Περιγραφή μοντέλου

Το μοντέλο $R_p - R_i$ αναπαριστά ένα σχήμα TSR με σχέσεις σχεσιακής άλγεβρας. Για κάθε λοιπόν κόμβο – αντικείμενο ορίζουμε τη σχέση R_i που περιέχει τις ιδιότητες και τις εγγραφές του και επιπλέον τη σχέση R_p με τα μονοπάτια που οδηγούν σ' αυτόν. Τέλος ορίζουμε και μια σχέση R_{tsr} που περιέχει πληροφορίες του TSR σχήματος.

Στο Σχ. 3.3 παρουσιάζεται το διάγραμμα οντοτήτων – συσχετίσεων του μοντέλου $R_p - R_i$. Παρατηρούμε ότι για κάθε TSR φυλάσσεται το όνομα και το σχήμα καταλόγου απ' όπου προήρθε. Η οντότητα R_p έχει γνωρίσματα τις μεταβλητές μονοπατιών (*path variables*) που αντιστοιχούν στα AND μονοπάτια της TSR και στην οντότητα R_i υπάρχουν γνωρίσματα που αντιστοιχούν στις ιδιότητες που έχει ο κόμβος – αντικείμενο της TSR.



- Σχ. 3.3 -

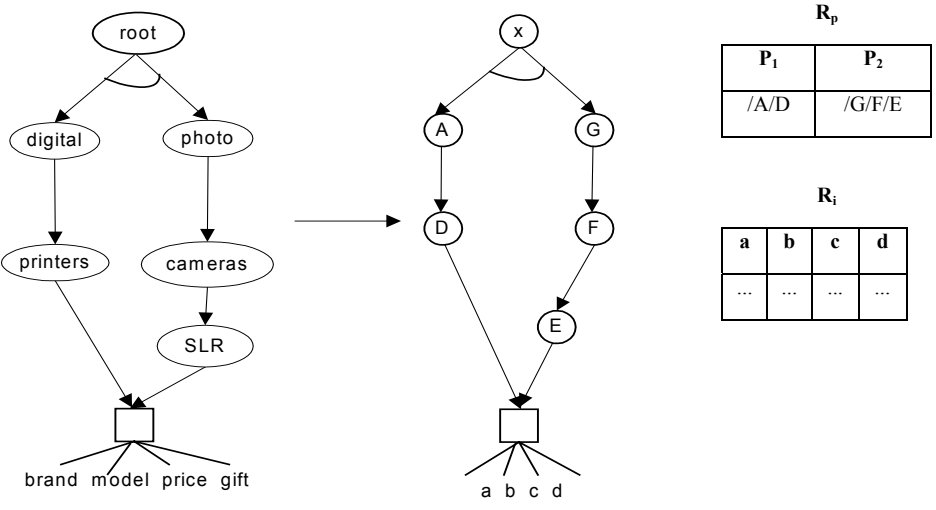
3.2.2 Κατασκευή – Μετάβαση από σχήμα TSR σε μοντέλο $R_p - R_i$

Για την κατασκευή του $R_p - R_i$ μοντέλου μιας TSR πρέπει να εντοπιστούν οι OR συνιστώσες του σχήματος, καθεμιά από τις οποίες προσθέτει μία πλειάδα στην R_p σχέση. Η OR συνιστώσα με τα περισσότερα AND μονοπάτια καθορίζει τον αριθμό των μεταβλητών

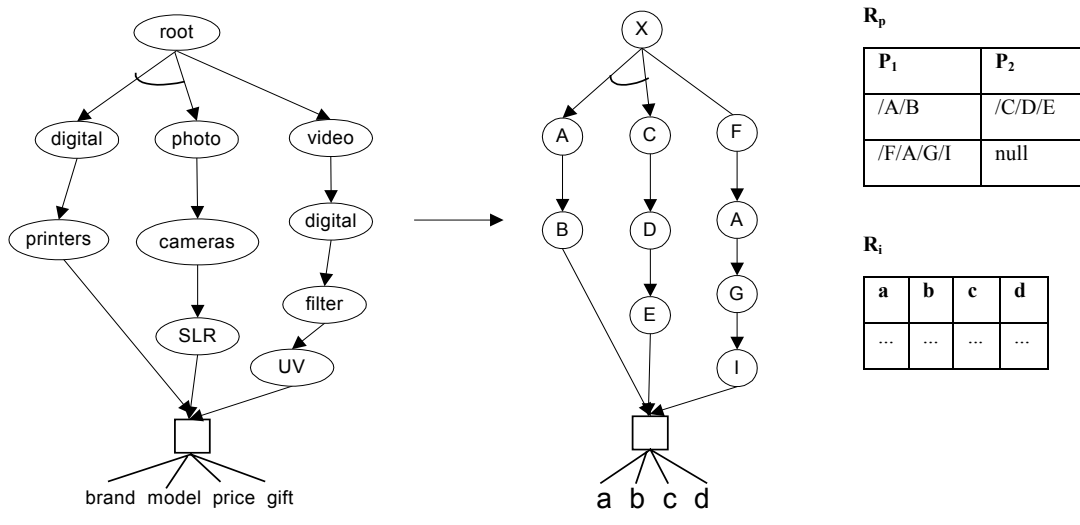
μονοπατιών P_i που υπάρχουν. Οι αρίθμηση των μεταβλητών μονοπατιών μιας OR συνιστώσας γίνεται ξεκινώντας από το αριστερό μονοπάτι προς το δεξιό. Τέλος για κάθε OR συνιστώσα τοποθετούνται στη σχέση R_p στην κατάλληλη μεταβλητή μονοπατιών τα AND μονοπάτια. Η R_i σχέση κατασκευάζεται απλά χρησιμοποιώντας τις ιδιότητες του κόμβου – αντικειμένου της TSR.

Για παράδειγμα στα Σχ. 3.4 και Σχ. 3.5 παρουσιάζονται οι σχέσεις R_p και R_i για δύο TSR σύμφωνα με το μοντέλο $R_p - R_i$. Για απλούστευση, οι εγγραφές των κόμβων – αντικειμένων δεν εισάγονται στις σχέσεις R_i και τα ονόματα των κόμβων έχουν αντικατασταθεί με αναγνωριστικά (A, B, C...).

Στο Σχ. 3.4 τα μονοπάτια /A/D και /G/F/E αποτελούν τα AND μονοπάτια που ορίζουν τη μοναδική συνιστώσα του σχήματος. Τα AND αυτά μονοπάτια καθορίζονται από τις δύο μεταβλητές μονοπατιών P_1 και P_2 , όπου P_1 είναι το αριστερότερο από τα δύο.



- Σχ. 3.4 -



- Σχ. 3.5 -

Στο Σχ. 3.5 υπάρχουν δύο OR συνιστώσες στο σχήμα της TSR και κατά συνέπεια δημιουργούνται δύο πλειάδες στη σχέση Rp. Η πρώτη συνιστώσα αποτελείται από τα AND μονοπάτια /A/B και /C/D/E, ενώ η δεύτερη από το /F/A/G/I. Επειδή η δεύτερη συνιστώσα έχει μόνο ένα AND μονοπάτι η πλειάδα της Rp σχέση έχει στην αντίστοιχη μεταβλητή μονοπατιού null.

3.3 Σύγκριση μονοπατιών

Στη συνέχεια ορίζουμε ένα σύνολο τελεστών που καθορίζουν τη σύγκριση μεταξύ δύο μονοπατιών μιας TSR σχέσης.

3.3.1 Ισότητα – Ανισότητα

Ορισμός 3.7. Για δύο μονοπάτια P1 και P2 ισχύει $P1 = P2$ ανν έχουν τους ίδιους ακριβώς κόμβους με την ίδια σειρά

Παράδειγμα:

$$/cameras/SLR = /cameras/SLR$$

Ορισμός 3.8. Για δύο μονοπάτια P1 και P2 ισχύει $P1 \neq P2$ ανν οι κόμβοι τους δεν είναι ίδιοι ή δεν είναι με την ίδια σειρά προηγούμενου – επόμενου.

Παράδειγμα:

$$/SLR/cameras/digital \neq /digital/cameras/SLR$$

3.3.2 Αυστηρό υποσύνολο

Ορισμός 3.9. Για δύο μονοπάτια P1 και P2 ισχύει $P1 \subseteq P2$ ανν όλοι οι κόμβοι του πρώτου υπάρχουν στο δεύτερο και με την ίδια ακριβώς σειρά προηγούμενου – επόμενου, χωρίς να παρεμβάλλονται άλλοι.

Μια τέτοια σχέση μεταξύ των μονοπατιών σημαίνει εναλλακτικά ότι το P1 προσφέρει μια πιο γενική κατηγοριοποίηση από το P2, δηλαδή το P2 είναι πιο εξειδικευμένο.

Παράδειγμα:

$$/cameras/SLR \subseteq /digital/cameras/SLR$$

Αντίθετα το μονοπάτι /digital/SLR δεν είναι αυστηρό υποσύνολο του /digital/cameras/SRL.

3.3.3 Χαλαρό υποσύνολο

Ορισμός 3.10. Για δύο μονοπάτια $P1$ και $P2$ ισχύει $P1 \subset P2$ ανν όλοι οι κόμβοι του πρώτου περιέχονται στο δεύτερο με την ίδια σειρά προηγούμενου – επόμενου, ανεξάρτητα αν παρεμβάλλονται και άλλοι κόμβοι.

Μια τέτοια σχέση μεταξύ των μονοπατιών σημαίνει εναλλακτικά ότι το $P1$ προσφέρει μια πιο γενική κατηγοριοποίηση από το $P2$, δηλαδή το $P2$ είναι πιο εξειδικευμένο.

Παράδειγμα:

$/digital/SLR \subset /digital/cameras/SLR$

3.4 Ερωτήσεις – Πράξεις

Στην παράγραφο αυτή ορίζεται το σύνολο των πράξεων που μπορούν να εφαρμοστούν σε TSR σχέσεις. Το σύνολο των τελεστών αυτών πρέπει να ικανοποιεί την απαίτηση της κλειστότητας, δηλαδή σε κάθε πράξη είσοδος να είναι ένα ή περισσότερα TSR σχήματα και το αποτέλεσμα πρέπει να είναι κι αυτό TSR σχήμα.

3.4.1 Βασικοί τελεστές

Αρχικά ορίζονται οι βασικοί τελεστές που αποτελούν τον πυρήνα της γλώσσας.

3.4.1.1 Επιλογή (σ)

Με την επιλογή επιλέγονται τα μονοπάτια που οδηγούν στον κόμβο – αντικείμενο της TSR και τα δεδομένα των εγγραφών που ικανοποιούν τις συνθηκών μονοπατιών και ιδιοτήτων αντίστοιχα. Ακολουθεί η σύνταξη της πράξης:

Σύνταξη σ

$\sigma \langle \text{Συνθήκη ιδιοτήτων} \rangle \langle \text{Συνθήκη μονοπατιών} \rangle (\text{TSR})$

Συνθήκη ιδιοτήτων: Ένωση ή τομή Boolean συνθηκών που αναφέρονται στα attributes των κόμβων – αντικειμένων με τη μορφή ιδιότητα **τελεστής** τιμή ή ιδιότητα **τελεστής** ιδιότητα, όπου **τελεστής** $\in \{=, \neq, >, <, \geq, \leq\}$.

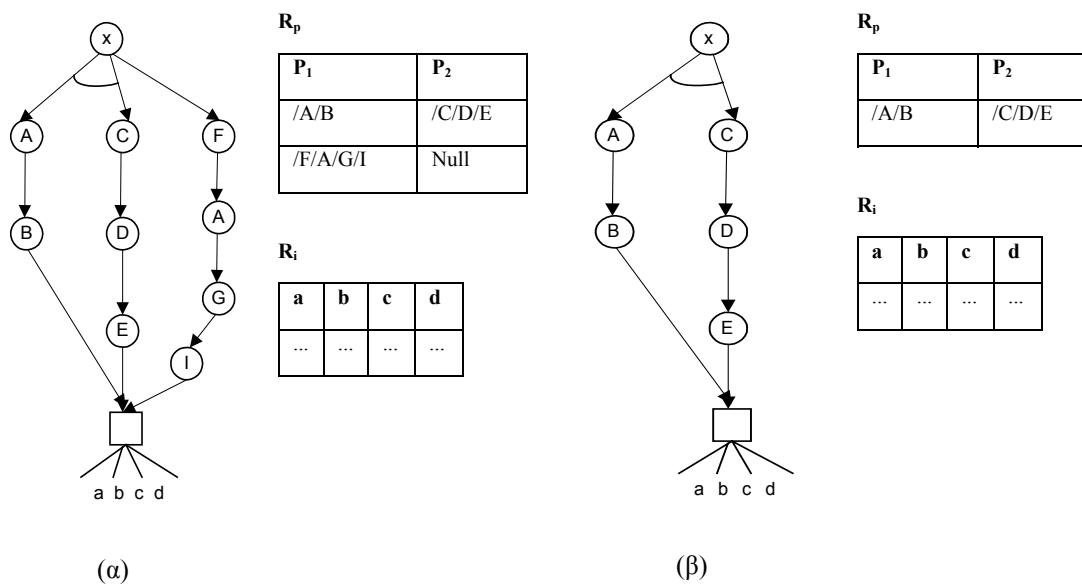
Συνθήκη μονοπατιών: Ένωση ή τομή Boolean συνθηκών που αναφέρονται στα μονοπάτια των κόμβων – αντικειμένων μέσω της σχέσης R_p με τη μορφή μεταβλητή μονοπατιών **τελεστής** τιμή ή μεταβλητή μονοπατιών **τελεστής** μεταβλητή μονοπατιών, όπου **τελεστής** $\in \{=, \neq, \supset, \subset, \ni, \subseteq\}$.

Παράδειγμα

Θεωρούμε τη TSR R_1 του Σχ. 3.6 (α). Θέλουμε με χρήση της επιλογής να κατασκευάσουμε μια νέα TSR τις οποίας τα αριστερότερα μονοπάτια κάθε OR συνιστώσας να είναι αυστηρά υπερσύνολα του $/A/B$, ενώ ο κόμβος – αντικείμενο να έχει εγγραφές που ικανοποιούν τη συνθήκη ιδιοτήτων $a = c$. Η ερώτηση που εκτελούμε είναι:

$$\sigma_{\langle a = c \rangle \langle P1 \rangle} \supseteq /A/B \langle R_1 \rangle$$

(αν θέλαμε τα δεξιότερα μονοπάτια να ικανοποιούν τη συνθήκη θα βάζαμε P_2 αντί για P_1) και το αποτέλεσμα βρίσκεται στο Σχ. 3.6 (β) (σ.σ. για συντομία δεν εισάγονται στους R_i οι εγγραφές).



- Σχ. 3.6 -

3.4.1.2 Προβολή (π)

Χρησιμοποιώντας την προβολή καθορίζονται ποια AND μονοπάτια σε κάθε OR συνιστώσα της TSR και ποιες ιδιότητες του κόμβου – αντικειμένου θα υπάρχουν στο αποτέλεσμα. Ακολουθεί η σύνταξη της πράξης:

Σύνταξη π

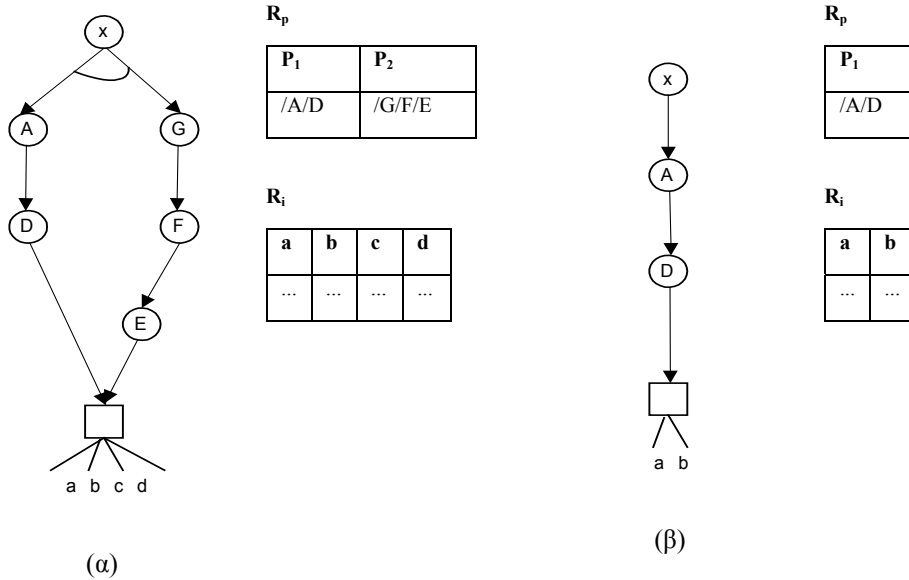
$$\pi_{\langle \text{Λίστα ιδιοτήτων} \rangle \langle \text{Λίστα μεταβλητών μονοπατιών} \rangle} \langle \text{TSR} \rangle$$

Παράδειγμα

Θεωρούμε τη TSR R_1 του Σχ. 3.7 (α). Θέλουμε με χρήση της προβολής να κατασκευάσουμε μια νέα TSR που να έχει μόνο τα μονοπάτια με μεταβλητή μονοπατιού P_1 από τα αρχικά, ενώ παράλληλα ο κόμβος – αντικείμενό της να έχει εγγραφές με ιδιότητες μόνο τις a και c . Η ερώτηση που εκτελούμε είναι:

$$\pi_{\langle a,b \rangle \langle P_1 \rangle} (R_1)$$

και το αποτέλεσμα βρίσκεται στο Σχ. 3.7 (β).



- Σχ. 3.7 -

3.4.1.3 Καρτεσιανό Γινόμενο (X)

Το καρτεσιανό γινόμενο εφαρμόζεται πάνω σε δύο TSR . Παράγεται μια νέα TSR της οποίας τα μονοπάτια προκύπτουν από συνδυασμούς των OR συνιστωσών των αρχικών σχέσεων και ο νέος κόμβος – αντικείμενο έχει ιδιότητες το συνδυασμό των ιδιοτήτων των εισόδων. Ακολουθεί η σύνταξη της πράξης:

Σύνταξη X

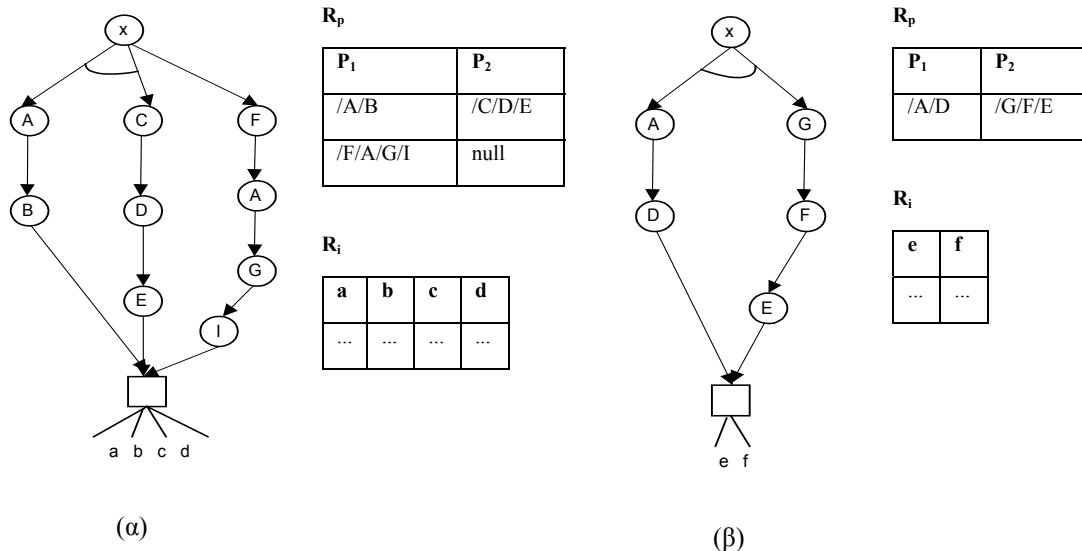
$$(TSR) X (TSR)$$

Παράδειγμα

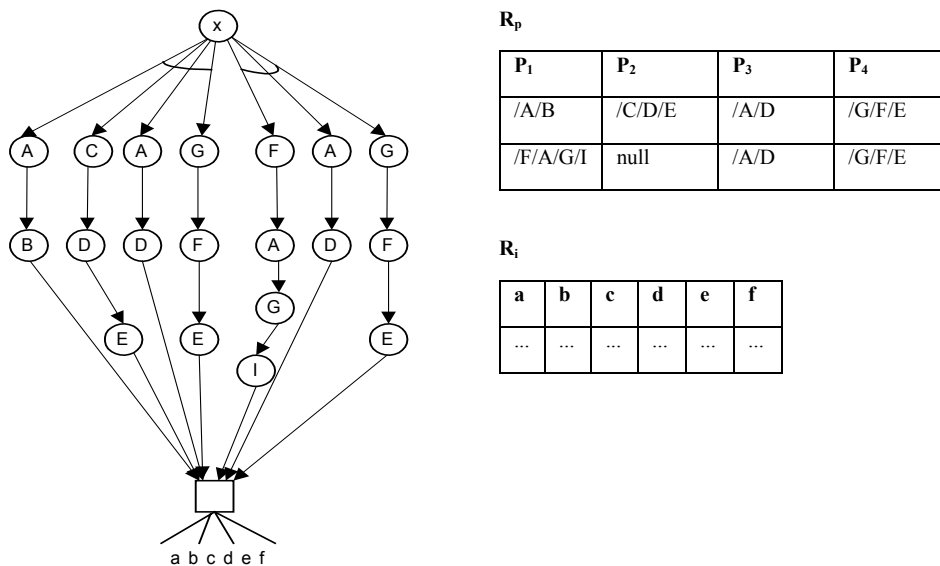
Θεωρούμε τα σχήματα TSR R_1 και R_2 των Σχ. 3.8 (α) και Σχ. 3.8 (β). Κατασκευάζουμε μια νέα TSR με συνδυασμό των αρχικών χρησιμοποιώντας το καρτεσιανό γινόμενο. Ο συνδυασμός δημιουργεί νέες OR συνιστώσες συνδυάζοντας αυτές της πρώτης TSR με αυτής της δεύτερης. Παράλληλα στον κόμβο – αντικείμενο συνδυάζονται οι εγγραφές των αρχικών σχημάτων. Η ερώτηση που εκτελούμε είναι:

$$(R_1)X(R_2)$$

και το αποτέλεσμα βρίσκεται στο Σχ. 3.9.



- Σχ. 3.8 -



- Σχ. 3.9 -

3.4.1.4 Ένωση (U)

Ένωση εφαρμόζεται σε δύο TSR με αποτέλεσμα μια νέα TSR με σύνολο μονοπατιών που προκύπτει από την ένωση των OR συνιστωσών των αρχικών TSR και κόμβο – αντικείμενο με εγγραφές που υπάρχουν είτε στη μία είσοδο είτε στην άλλη. Ακολουθεί η σύνταξη της πράξης:

Σύνταξη

$$(TSR) U (TSR)$$

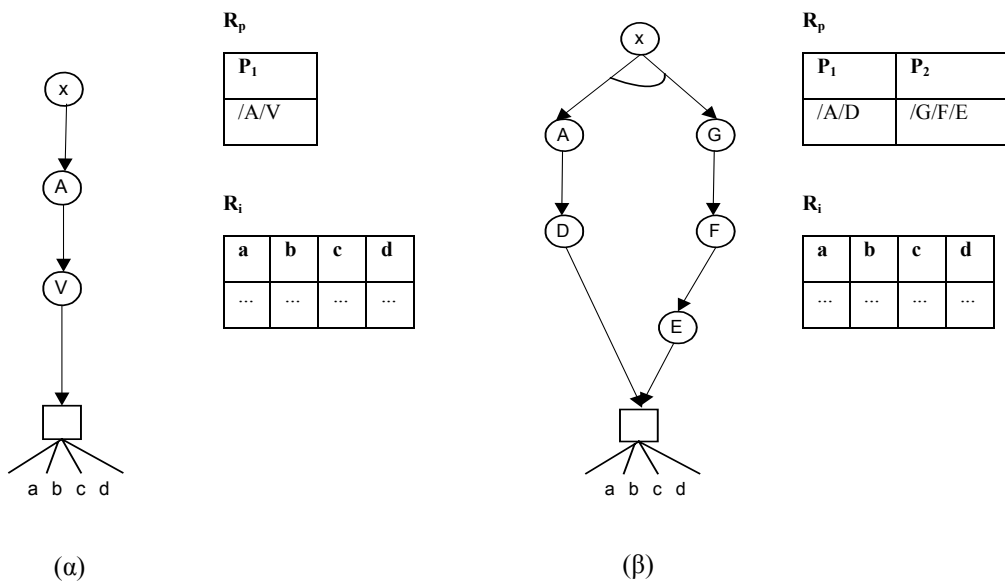
Για να θεωρηθούν οι TSR έγκυρες ως προς την πράξη της ένωσης πρέπει οι κόμβοι - αντικείμενα τους έχουν την ίδια δομή, δηλαδή τις ίδιες ιδιότητες.

Παράδειγμα

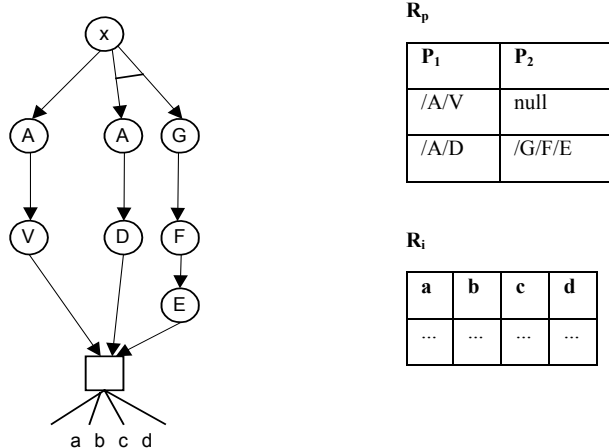
Θεωρούμε τα σχήματα R_1 και R_2 TSR των Σχ. 3.10 (α) και Σχ. 3.10 (β). Κατασκευάζουμε μια νέα TSR που αποτελεί την ένωση των αρχικών. Η νέα TSR προκύπτει από την πρώτη προσθέτοντας ως καινούριες OR συνιστώσες τις συνιστώσες της δεύτερης εισόδου. Η ερώτηση που εκτελούμε είναι:

$$(R_1)U(R_2)$$

και το αποτέλεσμα βρίσκεται στο Σχ. 3.11.



- Σχ. 3.10 -



- Σχ. 3.11 -

3.4.1.5 Τομή (\cap)

Η τομή ορίζεται ανάλογα με τον ένωση με τη διαφορά ότι στο αποτέλεσμα υπάρχουν μόνο οι κοινές εγγραφές των κόμβων – αντικειμένων των αρχικών TSR. Ακολουθεί η σύνταξη της πράξης:

Σύνταξη

$$(TSR) \cap (TSR)$$

Για να είναι οι σχέσεις έγκυρες ως προς την πράξη της τομής πρέπει οι κόμβοι - αντικείμενα τους να έχουν την ίδια δομή, δηλαδή τις ίδιες ιδιότητες.

3.4.1.6 Διαφορά ($-$)

Ομοίως με ένωση και τομή ορίζεται και η διαφορά, μόνο που το τελικό αποτέλεσμα περιέχει τις εγγραφές που υπάρχουν στο πρώτο κόμβο – αντικείμενο και δεν υπάρχουν στο δεύτερο. Ακολουθεί η σύνταξη της πράξης:

Σύνταξη

$$(TSR) - (TSR)$$

Για να είναι οι σχέσεις έγκυρες ως προς την πράξη της διαφοράς πρέπει οι κόμβοι - αντικείμενα τους να έχουν την ίδια δομή, δηλαδή τις ίδιες ιδιότητες.

3.4.2 Σύνθετοι τελεστές

Χρησιμοποιώντας τις βασικές πράξεις της άλγεβρας της γλώσσας TreeSQuerL μπορούν να οριστούν σύνθετοι τελεστές.

3.4.2.1 Σύνδεση (\bowtie)

Η πράξη της σύνδεσης εφαρμόζεται σε δύο TSR και δίνει αποτέλεσμα ένα καινούριο σχήμα επιλέγοντας από τους συνδυασμούς των OR συνιστωσών των αρχικών σχημάτων αυτούς που τα AND μονοπάτια ικανοποιούν συγκεκριμένες συνθήκες. Ακόμα στο νέο σχήμα για τον κόμβο – αντικείμενο επιλέγονται από τις εγγραφές των συνδυασμών των αρχικών TSR αυτές που ικανοποιούν τις δοσμένες συνθήκες. Ακολουθεί η σύνταξη της πράξης:

Σύνταξη

$$(TSR) \bowtie_{\langle \text{Συνθήκη ιδιοτήτων} \rangle \langle \text{Συνθήκη μονοπατιών} \rangle} (TSR)$$

Συνθήκη ιδιοτήτων: Ένωση ή τομή Boolean συνθηκών που αναφέρονται στα attributes των κόμβων – αντικειμένων με τη μορφή ιδιότητα **τελεστής** τιμή ή ιδιότητα **τελεστής** ιδιότητα, όπου **τελεστής** $\in \{=, \neq, >, <, \geq, \leq\}$.

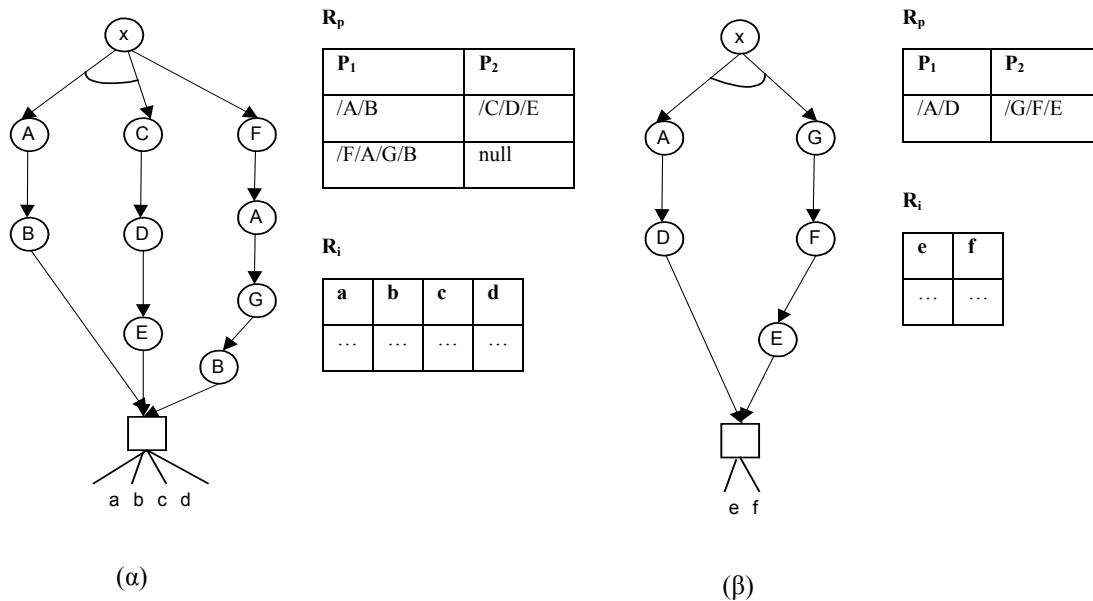
Συνθήκη μονοπατιών: Ένωση ή τομή Boolean συνθηκών που αναφέρονται στα μονοπάτια των κόμβων – αντικειμένων μέσω της σχέσης R_p με τη μορφή μεταβλητή μονοπατιών **τελεστής** τιμή ή μεταβλητή μονοπατιών **τελεστής** μεταβλητή μονοπατιών, όπου **τελεστής** $\in \{=, \neq, \supset, \subset, \ni, \subseteq\}$.

Παράδειγμα

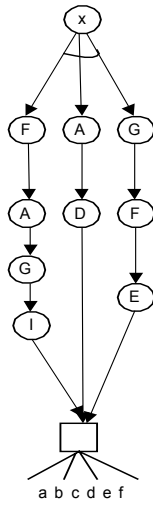
Θεωρούμε τα σχήματα R_1 και R_2 TSR των Σχ. 3.12 (α) και Σχ. 3.12 (β). Θέλουμε να συνδέσουμε τα δύο σχήματα κρατώντας από τους συνδυασμούς των μονοπατιών αυτούς των οποίων η μεταβλητή μονοπατιού P_1 είναι χαλαρό υπερσύνολο του $/A/B$ και η P_2 είναι διαφορετική από $/C/D/E$. Ενώ από τους συνδυασμούς των εγγραφών θέλουμε να κρατήσουμε αυτούς που ικανοποιούν τις συνθήκες ιδιοτήτων $a = f$ ή $a = c$. Η ερώτηση που εκτελούμε είναι:

$$(R_1) \bowtie_{\langle a=f \mid a=c \rangle \langle P_1 \supset /A/B, P_2 \neq /C/D/E \rangle} (R_2)$$

και το αποτέλεσμα βρίσκεται στο Σχ. 3.13.



- Σχ. 3.12 -



R_p

P_1	P_2	P_3	P_4
/F/A/G/I	null	/A/D	/G/F/E

R_i

a	b	c	d	e	f
...

- Σχ. 3.13 -

4

Ανάλυση και σχεδίαση

Στο κεφάλαιο αυτό παρουσιάζεται η μελέτη του συστήματος που υλοποιεί τη γλώσσα TreeSQuerL. Αρχικά γίνεται ο διαχωρισμός του συστήματος σε επιμέρους υποσυστήματα και παρουσιάζονται οι λειτουργικές απαιτήσεις του καθενός. Τέλος παρουσιάζονται αναλυτικά οι εφαρμογές που υλοποιούν το σύστημα.

4.1 Ανάλυση – Περιγραφή Αρχιτεκτονικής

Στην ενότητα αυτή περιγράφεται η αρχιτεκτονική του συστήματος, ξεκινώντας με την απαρίθμηση των επιμέρους ανεξάρτητων μονάδων του.

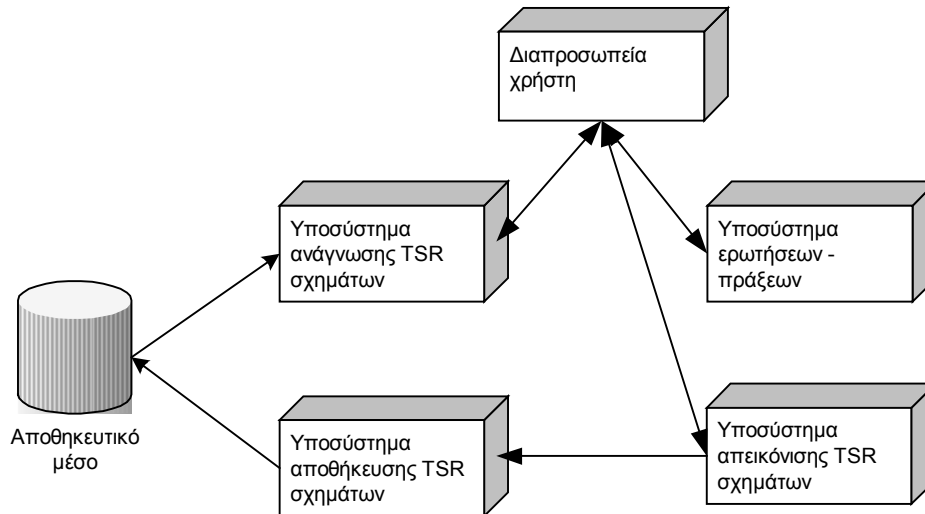
4.1.1 Διαχωρισμός υποσυστημάτων

Το σύστημα απαρτίζεται από πέντε υποσυστήματα:

1. Υποσύστημα ανάγνωσης σχημάτων TSR
2. Υποσύστημα αποθήκευσης σχημάτων TSR
3. Υποσύστημα ερωτήσεων – πράξεων
4. Υποσύστημα διαπροσωπείας χρήστη
5. Υποσύστημα απεικόνισης σχημάτων TSR

Το Σχ. 4.1 παρουσιάζει την αρχιτεκτονική του συστήματος. Μέσα από τη *διαπροσωπεία χρήση (interface)* διατυπώνονται ερωτήσεις σε σχήματα TSR που βρίσκονται στο

αποθηκευτικό μέσο του συστήματος. Για να εκτελεστεί κάθε πράξη, χρησιμοποιείται αρχικά το υποσύστημα ανάγνωσης σχημάτων TSR και στη συνέχεια το υποσύστημα ερωτήσεων – πράξεων της γλώσσας TreeSQuerL. Το υποσύστημα απεικόνισης σχημάτων είναι υπεύθυνο για την παρουσίαση των αποτελεσμάτων μιας πράξης. Σε περίπτωση που χρειαστεί να αποθηκευτεί ένα σχήμα TSR τότε χρησιμοποιείται το υποσύστημα αποθήκευσης TSR.



- Σχ. 4.1 -

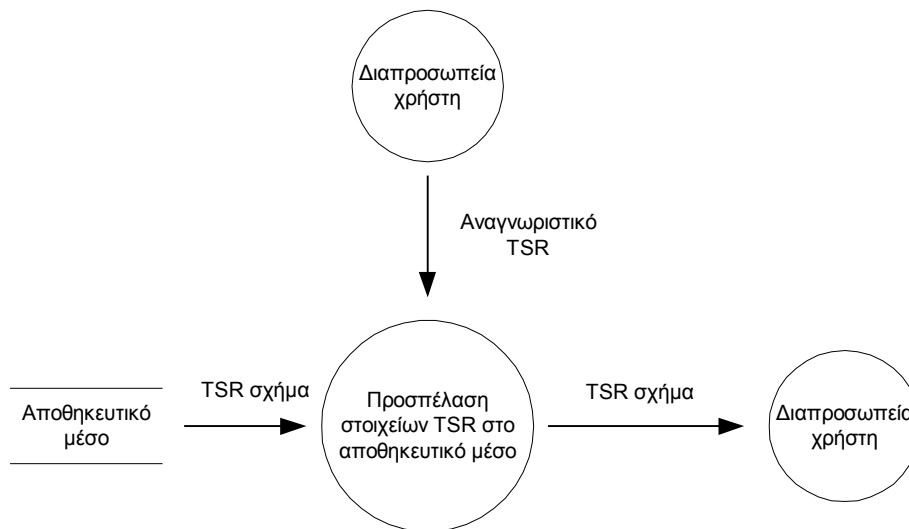
4.1.2 Περιγραφή υποσυστημάτων

Στη συνέχεια παρουσιάζονται οι προδιαγραφές του κάθε υποσυστήματος και οι λειτουργίες που πρέπει να επιτελεί.

4.1.2.1 Υποσύστημα ανάγνωσης σχημάτων TSR

Τα TSR σχήματα που επεξεργάζεται η γλώσσα TreeSQuerL βρίσκονται αποθηκευμένα στο αποθηκευτικό μέσο που προσφέρει το σύστημα. Η εκτέλεση μιας ερώτησης του υποσυστήματος ερωτήσεων – πράξεων προϋποθέτει τη φόρτωση των σχημάτων που ορίζονται στη σύνταξή της. Αυτή είναι και η βασική λειτουργία του υποσυστήματος ανάγνωσης σχημάτων TSR.

Στο Σχ. 4.2 φαίνεται το σχετικό διάγραμμα ροής δεδομένων. Από το υποσύστημα της διαπροσωπείας χρήστη ζητείται η φόρτωση ενός σχήματος TSR. Στη συνέχεια το υποσύστημα ανάγνωσης εντοπίζει την εγγραφή της TSR στο αποθηκευτικό μέσο και προσπελαύνει το σύνολο μονοπατιών και τον κόμβο – αντικείμενο για να δημιουργήσει το TSR σχήμα το οποίο θα χρησιμοποιηθεί για την εκτέλεση της ερώτησης.



- Σχ. 4.2 -

4.1.2.2 Υποσύστημα αποθήκευσης σχημάτων TSR

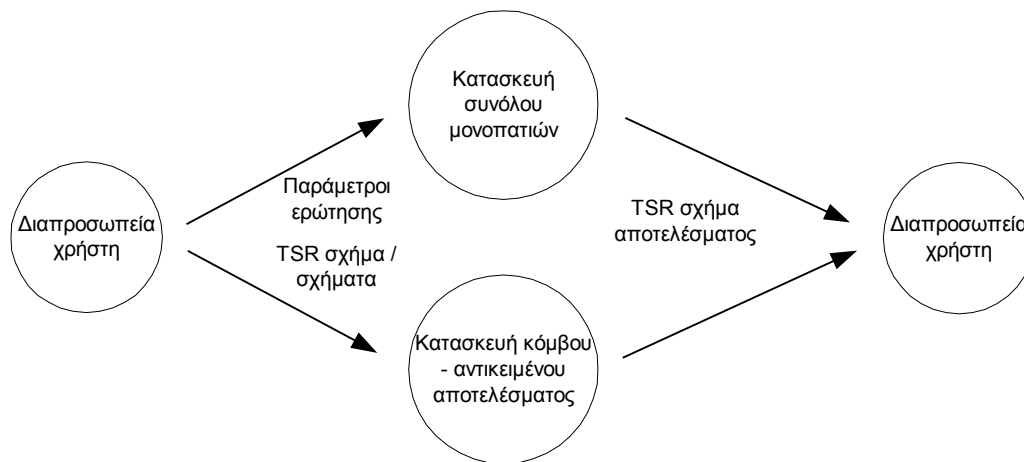
Το υποσύστημα αυτό είναι υπεύθυνο για τη αποθήκευση των TSR. Στο Σχ. 4.3 βλέπουμε πως όταν από το υποσύστημα απεικόνισης σχημάτων ζητείται η αποθήκευση μιας TSR, τότε με κατάλληλο τρόπο καταγράφονται στο αποθηκευτικό μέσο οι ιδιότητες και οι εγγραφές του κόμβου – αντικειμένου που περιέχονται στο TSR σχήμα, καθώς και το σύνολο των μονοπατιών που οδηγούν στον κόμβο αυτό.



- Σχ. 4.3 -

4.1.2.3 Υποσύστημα ερωτήσεων – πράξεων

Το υποσύστημα αυτό αποτελεί την καρδιά του συστήματος, καθώς υλοποιεί τους βασικούς τελεστές της γλώσσας TreeSQuerL. Από το υποσύστημα διαπροσωπείας χρήστη συλλέγονται όλες οι παράμετροι που χρειάζονται για την εκτέλεση της ερώτησης στα TSR σχήματα που έχουν φορτωθεί μέσω του υποσυστήματος ανάγνωσης σχημάτων TSR. Το αποτέλεσμα κάθε ερώτησης επιστρέφει στη διαπροσωπεία χρήστη. Στο Σχ. 4.4 φαίνεται το διάγραμμα ροής του υποσυστήματος ερωτήσεων – πράξεων.



- Σχ. 4.4 -

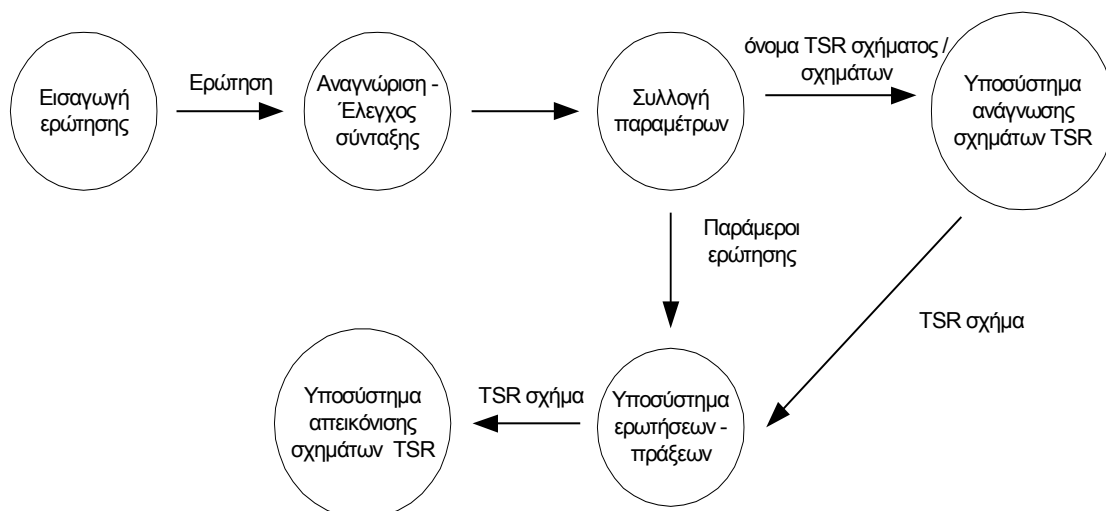
4.1.2.4 Υποσύστημα διαπροσωπείας χρήστη

Το υποσύστημα διαπροσωπείας χρήστη μαζί με το υποσύστημα της απεικόνισης σχημάτων TSR αποτελούν τη διεπαφή ολόκληρου του συστήματος και αναλαμβάνουν την αλληλεπίδρασή του με το χρήστη. Βασική λειτουργία που επιτελεί αυτό το κομμάτι της διεπαφής είναι να προσφέρει μια γραμματική για τη σύνταξη ερωτήσεων της γλώσσας TreeSQuerL σε TSR σχήματα. Λαμβάνοντας υπόψη τη λειτουργία αυτή, οι προδιαγραφές του υποσυστήματος διαπροσωπείας χρήστη είναι οι εξής:

1. Απαιτείται ένας εύχρηστος και σαφώς ορισμένος τρόπος σύνταξης των ερωτήσεων, ώστε να μπορεί ο χρήστης να διατυπώνει ερωτήσεις σύντομα και απλά.
2. Εξίσου σημαντικό είναι σε περίπτωση λάθους ο χρήστης να λαμβάνει επεξηγηματικά μηνύματα ώστε να μπορεί να διορθώσει και να επαναλάβει την ερώτηση εύκολα.
3. Τέλος, απαραίτητη προϋπόθεση που θέτει η διαπροσωπεία στο χρήστη είναι να γνωρίζει το σχήμα της TSR όταν θέλει να εκτελέσει μια ερώτηση. Βέβαια οι δύο πρώτες προδιαγραφές και ειδικά η δεύτερη εξασφαλίζουν ότι ο χρήστης σε περίπτωση λάθους θα εντοπίσει τι πρέπει να αλλάξει εύκολα.

Στο Σχ. 4.5 βλέπουμε τον τρόπο επικοινωνίας και συνεργασίας του υποσυστήματος με τα άλλα υποσυστήματα, καθώς και τη διαδικασία εκτέλεσης της βασικής του λειτουργίας. Ο χρήστης αρχικά εισάγει την ερώτηση με τη σύνταξη που ορίζει το υποσύστημα διαπροσωπείας χρήστη. Έπειτα η ερώτηση αναγνωρίζεται και ελέγχεται για την ορθότητα της σύνταξής της. Στη συνέχεια ακολουθεί μια διαδικασία συλλογής των παραμέτρων που χρειάζονται για την αποτίμησή της. Το πλήθος και το είδος των παραμέτρων εξαρτάται από την ίδια την πράξη, αλλά σίγουρα ανάμεσα τους είναι TSR σχήματα τα οποία θα προσπελαστούν μέσω του υποσυστήματος ανάγνωσης σχημάτων TSR. Από το υποσύστημα ανάγνωσης τα σχήματα TSR

περνούν στο υποσύστημα ερωτήσεων – πράξεων για την εκτέλεση της ερώτησης. Το αποτέλεσμα στη συνέχεια απεικονίζεται μέσω του υποσυστήματος απεικόνισης σχημάτων TSR.



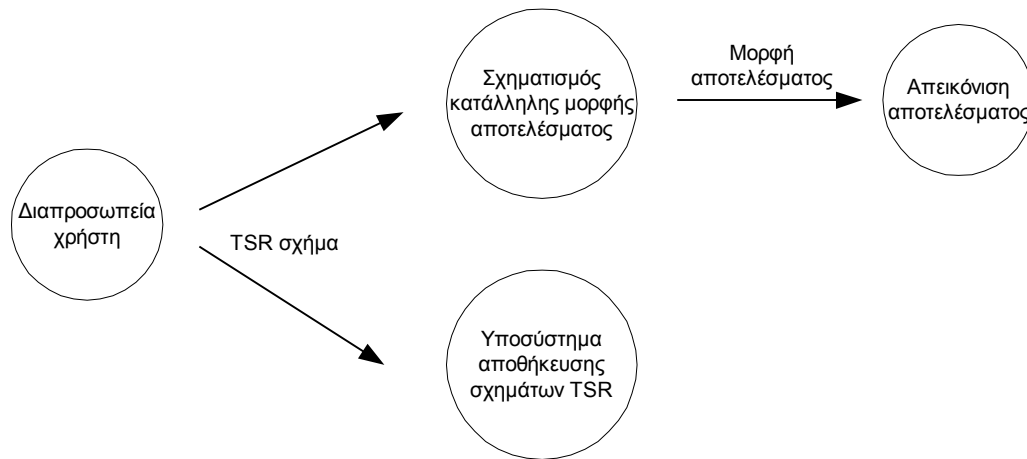
- Σχ. 4.5 -

4.1.2.5 Υποσύστημα απεικόνισης σχημάτων TSR

Βασική λειτουργία του υποσυστήματος αυτού είναι η απεικόνιση των TSR σχημάτων που παράγονται από την εκτέλεση ερώτησης του υποσυστήματος ερωτήσεων – πράξεων. Με δεδομένη αυτή τη λειτουργία υπάρχουν οι ακόλουθες προδιαγραφές:

1. Φιλικότητα προς το χρήστη.
2. Ευκολία ως προς τη χρήση.
3. Απεικόνιση των σχημάτων TSR με τέτοιο τρόπο ώστε να είναι εμφανή τόσο ο κόμβος – αντικείμενο με τις ιδιότητες και τις εγγραφές του όσο και το σύνολο μονοπατιών που οδηγούν σ' αυτό.
4. Δυνατότητα αποθήκευσης του TSR σχήματος που απεικονίζεται.

Στο Σχ. 4.6 διακρίνεται η επικοινωνία του υποσυστήματος απεικόνισης αποτελεσμάτων με άλλα υποσυστήματα. Μετά την εκτέλεση της πράξης, το αποτέλεσμα περνάει στο υποσύστημα απεικόνισης σχημάτων TSR, όπου σχηματίζεται μια δενδρική μορφή. Η μορφή αυτή απεικονίζεται με γραφικό τρόπο στο χρήστη. Το υποσύστημα απεικόνισης σχημάτων TSR επικοινωνεί και με το υποσύστημα αποθήκευσης, δίνοντας τη δυνατότητα στο χρήστη να αποθηκεύσει το αποτέλεσμα ως ένα καινούριο TSR σχήμα.



- Σχ. 4.6 -

4.2 Σχεδίαση του συστήματος

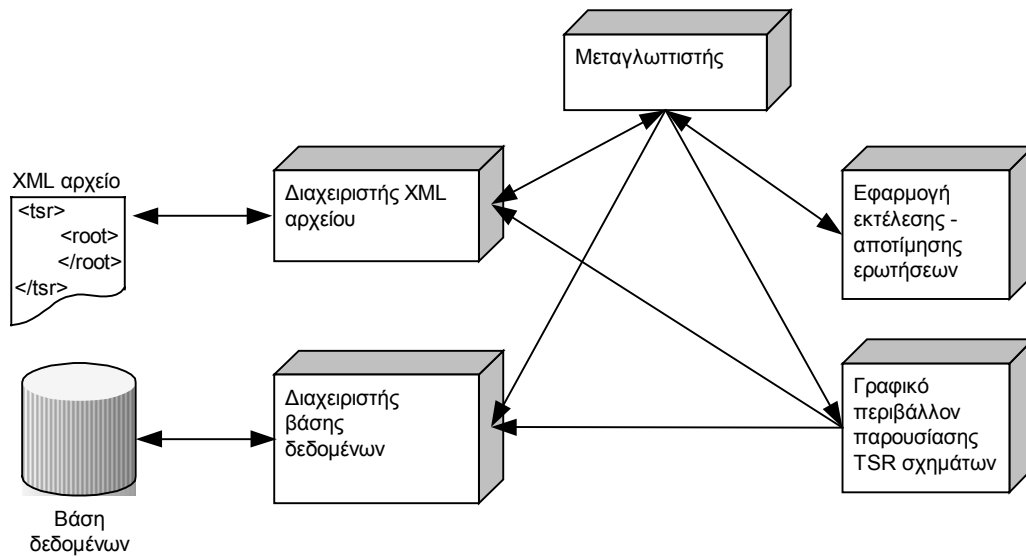
Στην ενότητα αυτή παρουσιάζονται οι εφαρμογές που επιτελούν τις βασικές λειτουργίες του συστήματος.

4.2.1 Εφαρμογές

Οι εφαρμογές του συστήματος είναι οι εξής:

1. Διαχειριστής XML αρχείου – σχήματος
2. Διαχειριστής βάσης δεδομένων
3. Εκτέλεσης – αποτίμησης ερωτήσεων γλώσσας TreeSQuerL
4. Μεταγλωττιστής (Interpreter)
5. Γραφικό περιβάλλον παρουσίασης αποτελεσμάτων TSR σχημάτων

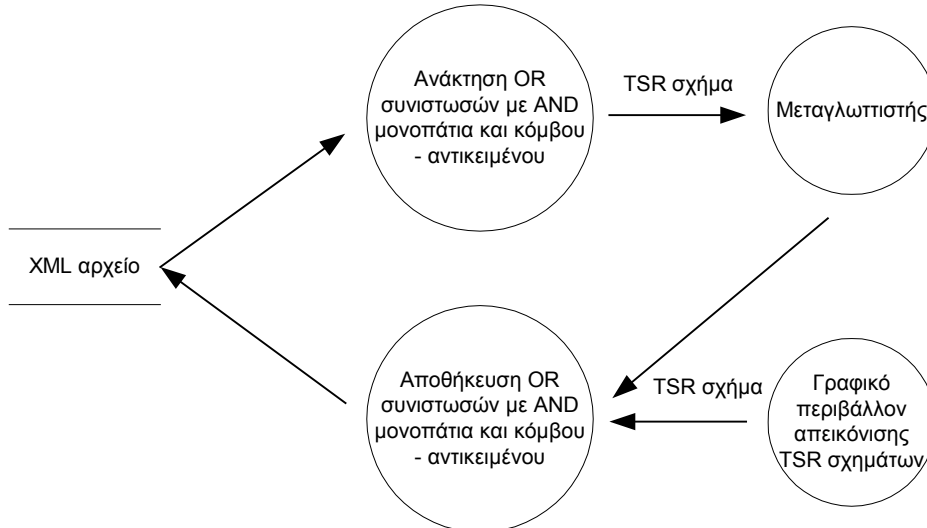
Στο Σχ. 4.7 βλέπουμε τον τρόπο επικοινωνίας των εφαρμογών μεταξύ τους για την εκτέλεση των λειτουργιών τους.



- Σχ. 4.7 -

4.2.1.1 Διαχειριστής XML αρχείου – σχήματος

Η εφαρμογή αυτή είναι υπεύθυνη για την αποθήκευση και τη φόρτωση ενός TSR σχήματος σε και από XML αρχείο αντίστοιχα. Κατά συνέπεια υλοποιεί μέρος του υποσυστήματος ανάγνωσης και του υποσυστήματος αποθήκευσης σχημάτων TSR. Στο Σχ. 4.8 φαίνεται η διαδικασία σχηματισμού ή αποθήκευσης ενός TSR σχήματος με διαχείριση του XML αρχείου.

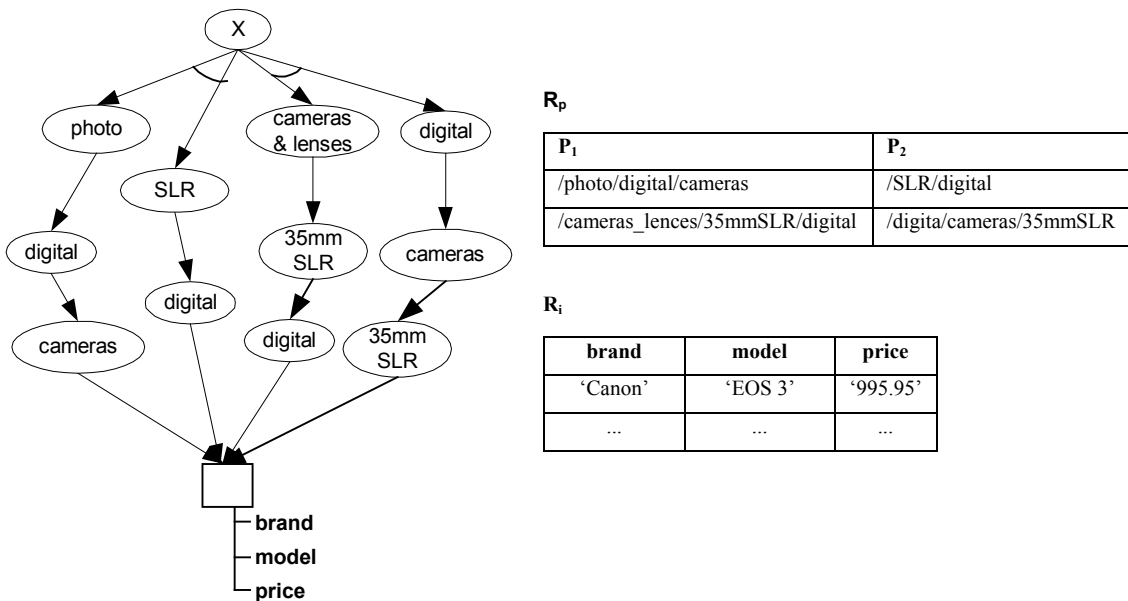


- Σχ. 4.8 -

Κάθε XML αρχείο κρατάει πληροφορίες για το όνομα της TSR, των σύνολο των μονοπατιών της και τον κόμβο – αντικείμενο. Συγκεκριμένα, περιγράφεται το $R_p - R_i$ μοντέλο (Βλ. κεφάλαιο 3, παράγραφο 3.2) του σχήματος παραθέτοντας τις OR συνιστώσες και προσθέτοντας σε κάθε μία τα AND μονοπάτια που περιέχει. Κάθε OR συνιστώσα περικλείεται σε <or> tag, ενώ κάθε AND μονοπάτι σε <and> tag. Για τον κόμβο αντικείμενο παρατίθενται

το όνομα και ο τύπος των ιδιοτήτων του καθώς και οι εγγραφές των περιεχομένων του. Ολόκληρος ο κόμβος – αντικείμενο περικλείεται σε ένα <item> tag και οι ιδιότητες και οι εγγραφές του σε <attribute> και <tuple> tags αντίστοιχα.

Για παράδειγμα, το Σχ. 4.9 παρουσιάζει τον τρόπο που οργανώνεται το σχήμα της TSR R₁ σε XML αρχείο. Τα γειτονικά AND μονοπάτια /photo/digital/cameras και /SLR/digital ορίζουν την πρώτη OR συνιστώσα του σχήματος, ενώ τα /cameras_lences/35mmSLR/digital και /digita/cameras/35mmSLR ορίζουν τη δεύτερη. Όσον αφορά τον κόμβο – αντικείμενο, υπάρχουν οι ιδιότητες brand, model, price για τη μάρκα, το μοντέλο και την τιμή αντίστοιχα.



- Σχ. 4.9 -

Με βάση το παραπάνω TSR σχήμα το XML αρχείο θα πάρει την εξής μορφή:

```
<root>
  <tsr name="r1">
    <or>
      <and>/photo/digital/cameras</and>
      <and>/SLR/digital</and>
    </or>
    <or>
      <and>/cameras_lences/35mmSLR/digital</and>
      <and>/digita/cameras/35mmSLR</and>
    </or>
    <item>
      <attribute name="brand" type="..."/>
      <attribute name="model" type="..."/>
      <attribute name="price" type="..."/>
      <tuple> 'Canon', 'EOS 3', '995.95' </tuple>
      <tuple> ... </tuple>
    </item>
  </tsr>
</root>
```

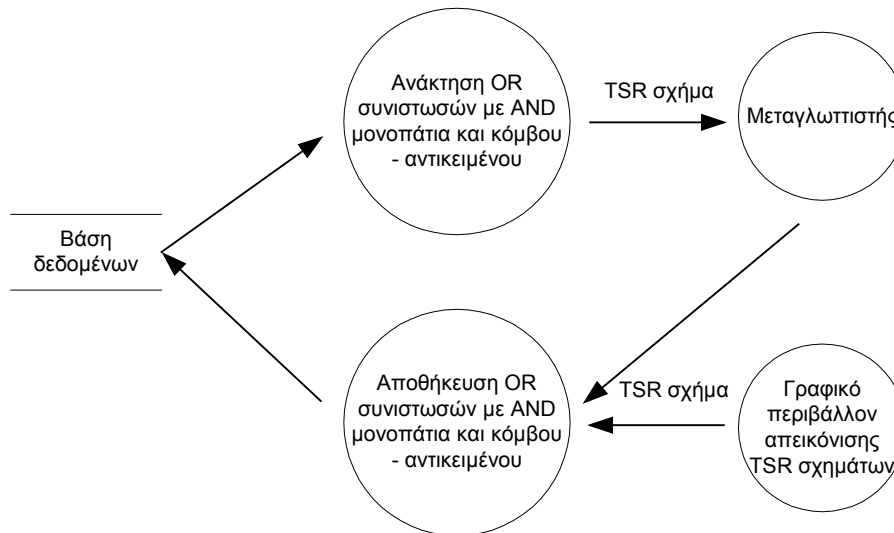
```

    </item>
  </tsr>
</root>

```

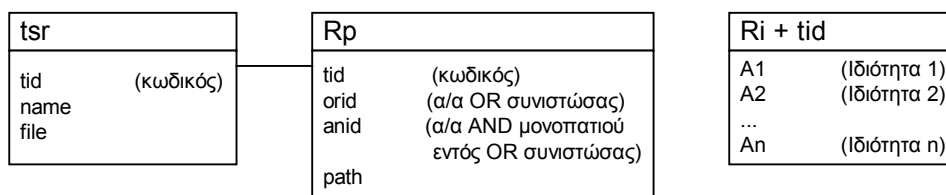
4.2.1.2 Διαχειριστής βάσης δεδομένων

Η εφαρμογή αυτή αποθηκεύει / προσπελαύνει TSR σχήματα στη / από τη βάση δεδομένων του συστήματος. Στο Σχ. 4.10 φαίνεται η διαδικασία σχηματισμού ή αποθήκευσης ενός TSR σχήματος με διαχείριση της βάσης δεδομένων.



- Σχ. 4.10 -

Κάθε TSR σχήμα που αποθηκεύεται στη βάση δεδομένων, οργανώνεται ακολουθώντας το Rp-Ri μοντέλο. Σύμφωνα μ' αυτό το μοντέλο, για κάθε TSR σχήμα κατασκευάζεται ο πίνακας *tsr* που είναι αντίστοιχος της οντότητας *Rtsr* όπου φυλάσσονται τα στοιχεία: *όνομα* και *XML αρχείο (σχήμα)* της TSR. Για την οντότητα *Ri* ορίζεται ένας νέος πίνακας με όνομα *Ri* συν τον κωδικό *tid* του TSR σχήματος από την αντίστοιχη εγγραφή στον *tsr* πίνακα. Τέλος τα μονοπάτια όλων των TSR σχημάτων αποθηκεύονται στον πίνακα *Rp*. Στον πίνακα αυτό κάθε πλειάδα αντιστοιχεί σε μονοπάτι κάποιας TSR με κωδικό *tid*, το οποίο ανήκει στην *orid* OR συνιστώσα του σχήματός της και είναι το *andid* αριστερότερο μονοπάτι εντός της συνιστώσας. Στο Σχ. 4.11 φαίνονται οι παραπάνω πίνακες.



- Σχ. 4.11 -

Για παράδειγμα για το TSR σχήμα R_1 του Σχ. 4.9 οι πίνακες της βάσης δεδομένων παρουσιάζονται στο Σχ. 4.12.

tid	name	File
1	r1	portal.xml

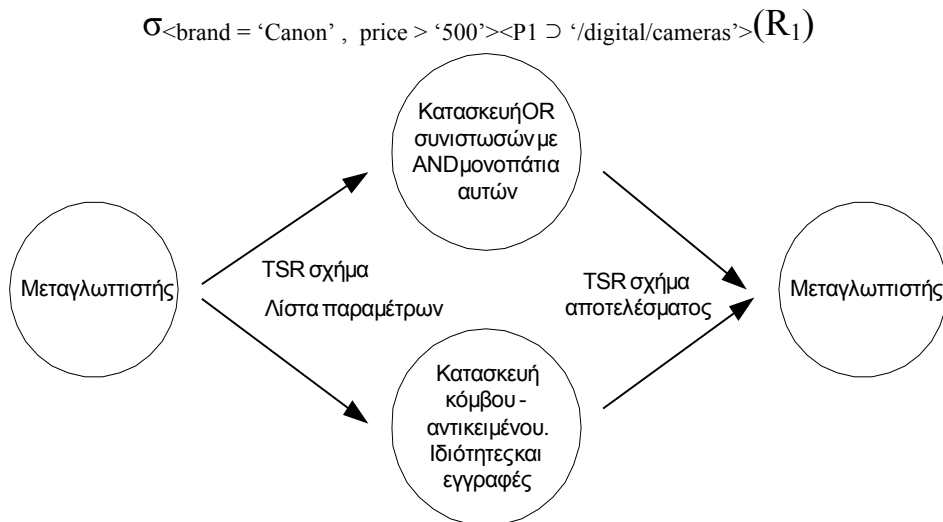
tid	orid	andid	path
1	1	1	/photo/digital/cameras
1	1	2	/SLR/digital
1	2	1	/cameras_lences/35mmSLR/digital
1	2	2	/digita/cameras/35mmSLR

brand	model	price
'Canon'	'EOS 3'	'995.95'

- Σχ. 4.12 -

4.2.1.3 Εφαρμογή εκτέλεσης – αποτίμησης ερωτήσεων γλώσσας TreeSQuerL

Η εφαρμογή αυτή αναλαμβάνει την υλοποίηση των βασικών τελεστών της γλώσσας TreeSQuerL, δηλαδή της επιλογής (σ), της προβολής (π), του καρτεσιανού γινομένου (X), της ένωσης (U), της τομής (\cap) και της διαφοράς ($-$). Κάθε φορά που πρόκειται να εκτελεστεί μια ερώτηση από τις παραπάνω, η εφαρμογή δέχεται ως είσοδο ένα ή δύο TSR σχήματα μαζί με τις λίστες παραμέτρων που χρειάζονται και επιστρέφει το αποτέλεσμα της πράξης ως ένα νέο TSR σχήμα. Για παράδειγμα στην ερώτηση «Βρες από την TSR R_1 του Σχ. 4.9 τις κάμερες μάρκας 'Canon' με τιμή μεγαλύτερη από 500 Ευρώ, στις οποίες οδηγούν αριστερότερα μονοπάτια που είναι χαλαρά υπερσύνολα του μονοπατιού /digital/cameras» αντιστοιχεί η παρακάτω ερώτηση της γλώσσας TreeSQuerL:



- Σχ. 4.13 -

Στο Σχ. 4.13 φαίνεται η διαδικασία αποτίμησης μιας ερώτησης που δέχεται ως είσοδο ένα TSR σχήμα και μια λίστα παραμέτρων. Στην περίπτωση που η ερώτηση εφαρμόζεται σε δύο σχήματα η είσοδος είναι δύο TSR. Ανάλογα με τη φύση της ερώτησης η εφαρμογή κατασκευάζει το νέο σχήμα δημιουργώντας τις OR συνιστώσες του με τα AND μονοπάτια που περιέχουν αυτές και τον κόμβο – αντικείμενο με τις ιδιότητες και τις εγγραφές του.

4.2.1.4 Μεταγλωττιστής (Interpreter)

Πρόκειται για την εφαρμογή που υλοποιεί το *υποσύστημα διαπροσωπείας χρήστη*. Προσφέρει στο χρήστη μια γραμματική σύνταξης ερωτήσεων επιλογής, προβολής, καρτεσιανού γινομένου, ένωσης, τομής και διαφοράς σε σχήματα TSR μέσω των αντίστοιχων εντολών. Χρησιμοποιώντας ένα συντακτικό αναλυτή (parser) αναγνωρίζει και ελέγχει την ορθότητα της σύνταξης της εντολής της αντίστοιχης πράξης της γλώσσας TreeSQuerL. Στη συνέχεια συλλέγει τις κατάλληλες παραμέτρους που χρειάζονται για την αποτίμηση της ερώτησης που γίνεται από την αντίστοιχη εφαρμογή και παράλληλα φορτώνει τα TSR σχήματα που χρειάζονται για την εκτέλεση της πράξης. Υπάρχει επίσης η δυνατότητα το αποτέλεσμα της πράξης να αποθηκευτεί μέσω της εφαρμογής διαχείρισης του XML αρχείου.

Ένα παράδειγμα σύνταξης ερώτησης μέσω του μεταγλωττιστή είναι το:

$s(\text{brand} = \text{'Canon'} , \text{price} > \text{'500'}) (p1 \text{ LH } \text{'/digital/cameras'}) (\text{portal.xml}\#r1)$

που αντιστοιχεί στην πράξη της TreeSQuerL γλώσσας για το TSR σχήμα του Σχ. 4.9:

$\sigma_{\langle \text{brand} = \text{'Canon'} , \text{price} > \text{'500'} \rangle} \langle P1 \supset \text{'/digital/cameras'} \rangle (R_1)$

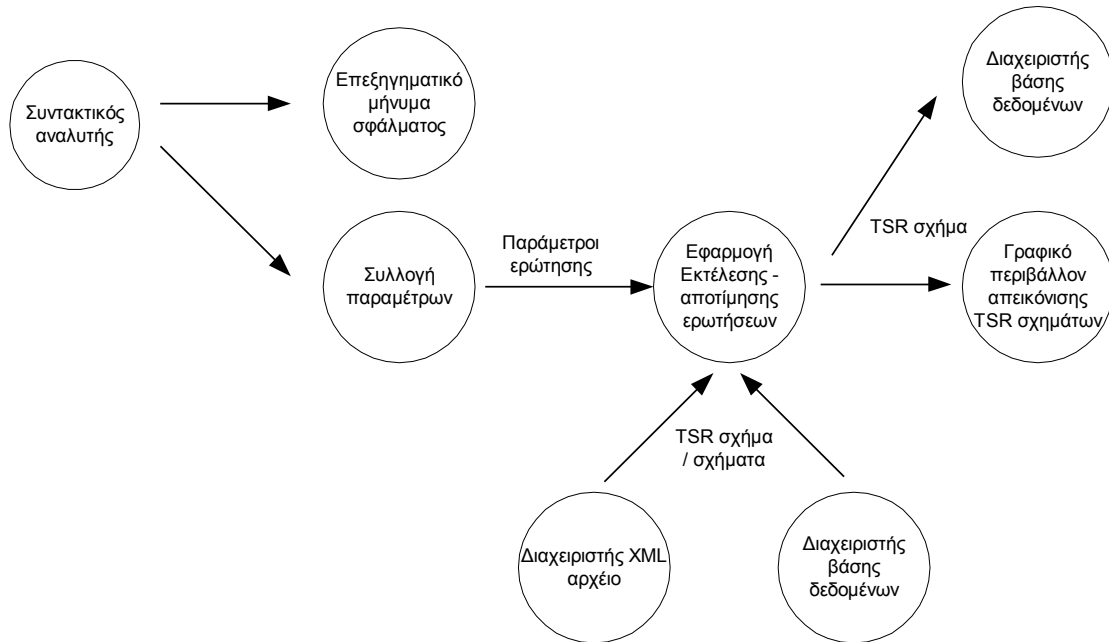
όπου s σημαίνει πράξη επιλογής (σ), το $P1$ συμβολίζει τη μεταβλητή μονοπατιού $p1$ και LH είναι ο συμβολισμός για τον τελεστή σύγκρισης μονοπατιών του χαλαρού υπερσυνόλου.

Σύμφωνα με τις προδιαγραφές του *υποσυστήματος διαπροσωπείας χρήστη*, ο μεταγλωττιστής εμφανίζει επεξηγηματικά μηνύματα σφάλματος στη σύνταξη μιας ερώτησης, καθώς και ένα εγχειρίδιο όπου φαίνεται η σύνταξη κάθε εντολής του μαζί με ένα χαρακτηριστικό παράδειγμα. Ακόμα υπάρχει μια πρόσθετη εντολή – λειτουργία που προσφέρει ο μεταγλωττιστής με την οποία μπορεί να μεταφερθεί ένα TSR σχήμα από το αποθηκευτικό μέσο XML αρχείο στη βάση δεδομένων. Τέλος υπάρχει η δυνατότητα δημιουργίας αρχείου ερωτήσεων (query file) με ένα ‘σενάριο’ ερωτήσεων που θέλουμε να εκτελεστεί σειριακά.

Το αρχείο ερωτήσεων έχει προέκταση .qry και η σύνταξή του είναι ανάλογη της σύνταξης των ερωτήσεων της γλώσσας TreeSQuerL. Στην αρχή κάθε αρχείου ερωτήσεων υπάρχει μία γραμμή με τον τίτλο του αρχείου -- Query file και στη συνέχεια εισάγονται οι εντολές του μεταγλωττιστή που θέλει ο χρήστης να εκτελέσει. Ένα παράδειγμα σύνταξης αρχείου ερωτήσεων με μια εντολή επιλογής είναι το παρακάτω:

```
-- Query file
s(brand = 'Canon' , price > '500')
  (p1 LH '/digital/cameras') (portal.xml#r1)
```

Στο Σχ. 4.14 παρουσιάζεται το διάγραμμα λειτουργίας τις εφαρμογής του μεταγλωττιστή τις περιγράφηκε.

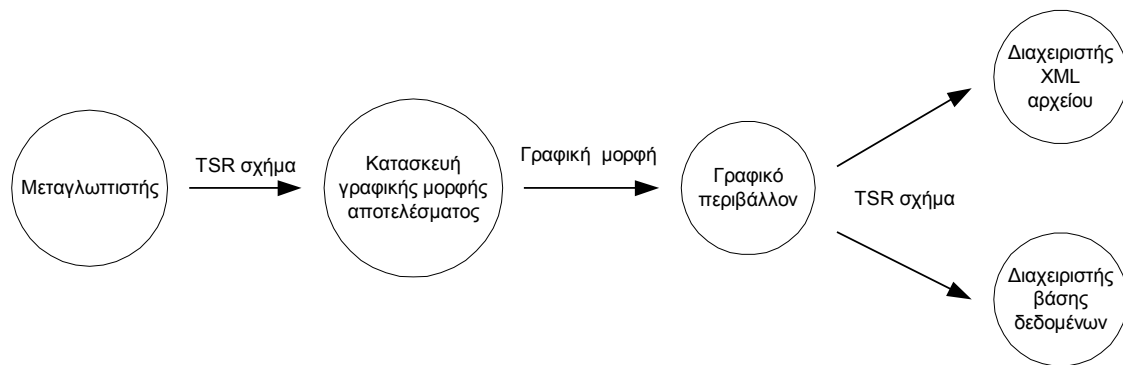


- Σχ. 4.14 -

4.2.1.5 Γραφικό περιβάλλον παρουσίασης αποτελεσμάτων TSR σχημάτων

Μ' αυτή την εφαρμογή υλοποιείται το υποσύστημα απεικόνισης σχημάτων TSR και ολοκληρώνεται η διεπαφή του συστήματος με το χρήστη. Από ένα TSR σχήμα κατασκευάζεται μια γραφική δενδρική απεικόνιση, ανάλογη του XML αρχείου που χρησιμοποιείται ως αποθηκευτικό μέσο, των OR συνιστωσών και των AND μονοπατιών τους και του κόμβου – αντικείμενου με τις ιδιότητες και τις εγγραφές του. Το γραφικό περιβάλλον προσφέρει τη δυνατότητα αποθήκευσης του TSR σχήματος που παρουσιάζεται είτε σε XML αρχείο είτε στη βάση δεδομένων. Το Σχ. 4.15 παρουσιάζει τις παραπάνω λειτουργίες της εφαρμογής.

Στο Σχ. 4.16 παρουσιάζεται το γραφικό περιβάλλον που απεικονίζει τα TSR σχήματα. Η μορφή είναι όμοια με το σχήμα του XML αρχείου που χρησιμοποιείται ως αποθηκευτικό μέσο. Έτσι χρησιμοποιούνται <or> tags για τις OR συνιστώσες, <and> για τα AND μονοπάτια, <item> tag για τον κόμβο – αντικείμενο και <attribute> και <tuple> tags για τις ιδιότητες και τις εγγραφές του αντίστοιχα. Το όνομα της TSR είναι DigitalCameras και είναι αποθηκευμένη στο αρχείο RitzCameras.xml. Διακρίνονται επίσης οι επιλογές για αποθήκευση σε XML αρχείο ή στη βάση δεδομένων.



- Σχ. 4.15 -

TSR Viewer [RitzCameras.xml#DigitalCameras]

```

<tsr name="DigitalCameras">
  <or>
    <and>/cameras</and>
  </or>
  <item>
    <attribute name="brand" type="text"/>
    <attribute name="model" type="text"/>
    <attribute name="CCD" type="float"/>
    <attribute name="opticalZoom" type="float"/>
    <attribute name="digitalZoom" type="float"/>
    <attribute name="LCD" type="float"/>
    <attribute name="weight" type="float"/>
    <attribute name="price" type="float"/>
    <tuple>'Canon', 'Powershot A60', '2', '0', '2.5', '1.5', '215', '229.99'</tuple>
    <tuple>'Canon', 'Powershot A70', '3.2', '3', '3.2', '1.5', '215', '299.98'</tuple>
    <tuple>'Casio', 'EX Z4', '4', '3', '4', '2', '194', '399.99'</tuple>
    <tuple>'Fujifilm', 'FinePix 2650 xD', '2', '3', '2.5', '1.5', '200', '169.99'</tuple>
    <tuple>'Fujifilm', 'FinePix A 205', '2', '3', '2.5', '1.5', '200', '199.99'</tuple>
    <tuple>'Nikon', 'Coolpix 2100', '2', '3', '4', '1.5', '150', '249.99'</tuple>
    <tuple>'Nikon', 'CoolPix 4300', '4', '3', '4', '1.5', '215', '399.99'</tuple>
    <tuple>'Panasonic', 'DMC LC20S', '2.1', '3', '2', '1.5', '100', '249.99'</tuple>
  </item>
</tsr>
  
```

- Σχ. 4.16 -

5

Υλοποίηση

Το αντικείμενο του κεφαλαίου αυτού είναι η περιγραφή της υλοποίησης του συστήματος και των εφαρμογών που το αποτελούν. Αρχικά, αναφέρονται με λίγα λόγια τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Στη συνέχεια, παρατίθενται οι λεπτομέρειες υλοποίησης των εφαρμογών με σύντομη περιγραφή του μεταγλωττιστή και αναφορά των κλάσεων που έχουν γραφτεί. Τελειώνοντας, παρουσιάζονται οι αλγόριθμοι που υλοποιούν τις εφαρμογές διαχείρισης των αποθηκευτικών μέσων, τις πράξεις της γλώσσας TreeSQL και τους τελεστές σύγκρισης μονοπατιών.

5.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Για την υλοποίηση του συστήματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java. Ο προγραμματισμός έγινε στο περιβάλλον ανάπτυξης της Borland JBuilder. Παράλληλα, χρησιμοποιήθηκαν JDBC οδηγί πρόσβασης σε βάση δεδομένων και το πακέτο dom4j για τη διαχείριση XML αρχείων. Ειδικά για την υλοποίηση της εφαρμογής του μεταγλωττιστή χρησιμοποιήθηκε η γεννήτρια συντακτικών αναλυτών JavaCC. Για τη βάση δεδομένων του συστήματος χρησιμοποιήθηκε η `mysql 4.0`.

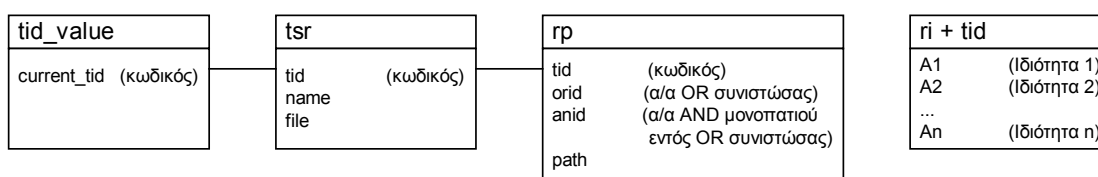
5.2 Λεπτομέρειες υλοποίησης

Στην ενότητα αυτή παρουσιάζονται οι λεπτομέρειες υλοποίησης του συστήματος ξεκινώντας από τη βάση δεδομένων και το σχήμα της. Έπειτα, περιγράφεται ο τρόπος δημιουργίας του μεταγλωττιστή, η σύνταξη και έλεγχος των ερωτήσεων που μπορούν να διατυπωθούν. Στη συνέχεια, γίνεται μια σύντομη περιγραφή των κλάσεων που έχουν συνταχθεί για την υλοποίηση των εφαρμογών του συστήματος. Τελειώνοντας, παρουσιάζονται οι βασικοί αλγόριθμοι του συστήματος που υλοποιούν τις ερωτήσεις της γλώσσας TreeSQuerL.

5.2.1 Βάση δεδομένων

Ως αποθηκευτικό μέσο των TSR σχημάτων, χρησιμοποιείται η βάση δεδομένων *dbTSR* στο σύστημα της *mysql*. Σύμφωνα με την παρουσίαση που έγινε στην περιγραφή της εφαρμογής διαχείρισης της βάσης δεδομένων (Βλ. Κεφάλαιο 4, παράγραφος 4.1.2.1), ακολουθώντας το *Rp – Ri* μοντέλο, για κάθε σχήμα φυλάσσονται το όνομά του, το XML αρχείο απ' όπου προέρχεται, το σύνολο των μονοπατιών και οι ιδιότητες και οι εγγραφές του κόμβου – αντικειμένου στους πίνακες *tsr*, *rp* και *ri*. Επειδή κάθε εγγραφή TSR σχήματος στη βάση πρέπει να χαρακτηρίζεται από ένα κωδικό *tid*, υπάρχει ένας βοηθητικός πίνακας *tid_value* όπου φυλάσσεται ο κωδικός του τελευταίου σχήματος που αποθηκεύτηκε. Έτσι κάθε φορά που επιχειρείται η εισαγωγή ενός TSR σχήματος στη βάση δίνεται σ' αυτή ένας κωδικός που προκύπτει με αναβάθμιση της εγγραφής του πίνακα *tid_value*.

Στο Σχ. 5.1 φαίνονται όλοι οι πίνακες της *dbTSR* βάσης δεδομένων. Ακολουθεί ο SQL κώδικας που δημιουργεί τους πίνακες εκτός από τον *ri* που δημιουργείται για κάθε σχήμα ξεχωριστά κάθε φορά.



- Σχ. 5.1 -

```
DROP DATABASE IF EXISTS dbTSR;

CREATE DATABASE dbTSR;
USE dbTSR;

CREATE TABLE tid_value (
    current      integer      NOT NULL );
CREATE TABLE tsr (
    tid          integer      NOT NULL,
    name         text         NOT NULL,
    file         text         NOT NULL,
```

```

        PRIMARY KEY(tid));
CREATE TABLE rp (
    tid          integer    NOT NULL,
    orid         integer    NOT NULL,
    andid        integer    NOT NULL,
    path         text       NOT NULL );

```

5.2.2 Μεταγλωττιστής

Η εφαρμογή του μεταγλωττιστή υλοποιήθηκε με χρήση της γεννήτριας συντακτικών αναλυτών (parsers) JavaCC (Java Compiler Compiler). Με χρήση του JavaCC ορίζεται η γραμματική μιας γλώσσας και αυτόματα παράγονται οι Java κλάσεις που υλοποιούν το συντακτικό αναλυτή που αναγνωρίζει τη γλώσσα αυτή. Ο ορισμός της γραμματικής και η διαχείριση της γλώσσας γίνεται στο πηγαίο αρχείο προέκτασης jj, στο οποίο γράφονται εκφράσεις της γλώσσας προγραμματισμού Java, αλλά και κανονικές εκφράσεις (regular expressions) και ειδικές εκφράσεις που ορίζει το εργαλείο JavaCC.

5.2.2.1 Γραμματική γλώσσας TreeSQuerL

Η δημιουργία του συντακτικού αναλυτή ξεκινά από τον ορισμό της γλώσσας που αναλύει. Έτσι, πριν από κάθε πράξη της γλώσσας TreeSQuerL ορίζονται οι αντίστοιχες εκφράσεις και ταυτόχρονα ορίζονται και οι λέξεις (tokens) που διαχειρίζεται η γλώσσα.

Οι λέξεις της γλώσσας παρουσιάζονται στις παρακάτω εκφράσεις στο σύνολο TOKEN. Οι εκφράσεις που περιέχονται στο σύνολο SKIP είναι αυτές που αγνοεί ο αναλυτής κατά τη συντακτική ανάλυση της έκφρασης που εισήγαγε ο χρήστης.

TOKEN :

{

Οι τελεστές της γλώσσας TreeSQuerL

< SELECT:	"s" "S" >
< PROJECT:	"p" "P" >
< CARTECIAN_PRODUCT:	"x" "X" >
< UNION:	"u" "U" >
< INTERSECTION:	"i" "I" >
< DIFFERENCE:	"d" "D" >

Εντολές για μελλοντική χρήση

< STRUCTURE_UNION:	"su" "SU" >
< STRUCTURE_INTERSECTION:	"si" "SI" >
< STRUCTURE_DIFFERENCE:	"sd" "SD" >

Επιπλέον εντολές

< LOAD:	"load" "LOAD" >
< SAVE2DB:	"s2db" "S2DB" >

Οι τελεστές σύγκρισης των εγγραφών των ιδιοτήτων και των μονοπατιών

```
| < ATTRIBUTE_OPERATOR:  "<" | ">" | "=" | "!=" | ">=" | "<=" >
| < PATH_OPERATOR:      "EQ" | "SS" | "SH" | "LS" | "LH" | "NE"
|                       | "eq" | "ss" | "sh" | "ls" | "lh" |
|                       | "ne" >

| < EXIT:                "exit" | "EXIT" >
| < HELP:                "help" | "HELP" >
```

Η έκφραση <AS> προστίθεται στο τέλος κάθε εντολής τελεστή της γλώσσας TreeSQuerL και σημαίνει αποθήκευση του αποτελέσματος ως νέο σχήμα σε XML αρχείο.

```
| < AS:                  "as" | "AS" >
```

Έκφραση για το όνομα του XML αρχείου

```
| < FILE:                (<INDENTIFIER> | ".." | "C:" | "c:")
|                       ("/" <INDENTIFIER>)* "." <XML> >
| < XML:                 "xml" | "XML" >
```

Η έκφραση <INDENTIFIER> αντιστοιχεί σε αλφαριθμητική λέξη όπως είναι η μεταβλητή ιδιοτήτων και το όνομα ενός TSR σχήματος. Η <PATH_VAR> είναι η έκφραση για τη μεταβλητή μονοπατιών.

```
| < INDENTIFIER:         <LETTER> ( <LETTER> | <DIGIT> )* >
| < PATH_VAR:           "$" ( ["1"-"9"] )+ >

| < LETTER:             ["a"-"z"] | ["A"-"Z"] >
| < DIGIT:              ["0"-"9"] >

| < EOL:                "\n" >
```

Η έκφραση <VALUE> αντιστοιχεί στην τιμή που μπορεί να πάρει μεταβλητή ιδιοτήτων του κόμβου αντικειμένου, ενώ η <PATH> στη μορφή που έχει το μονοπάτι ως τιμή της μεταβλητής μονοπατιών

```
| < VALUE:              "'" ( (<LETTER> | <DIGIT>)+
|                       ( "." (<LETTER> | <DIGIT>)+ )? ) "'" >
| < PATH:               ("/" (<LETTER> | <DIGIT>)+ )+ >
| }
```

SKIP :

```
{
|   " "
|   "\r"
|   "\t"
| }
```

Επιπρόσθετα ορίζονται οι εκφράσεις για τις λίστες συνθηκών ιδιοτήτων και μονοπατιών καθώς και οι λίστες ιδιοτήτων και μεταβλητών μονοπατιών.

ιδιότητα τελεστής ιδιότητα ή ιδιότητα τελεστής τιμή

```
< ATTRIBUTE_CONDITION:      (<INDENTIFIER> <ATTRIBUTE_OPERATOR>
                             (<INDENTIFIER> | <VALUE>)) ?
                             ((", " | "|") <INDENTIFIER>
                             <ATTRIBUTE_OPERATOR>
                             (<INDENTIFIER> | <VALUE>)) * >
```

μεταβλητή μονοπατιών τελεστής μεταβλητή μονοπατιών ή μεταβλητή μονοπατιών τελεστής τιμή (μονοπάτι)

```
< PATH_CONDITION:          (<PATH_VAR> <PATH_OPERATOR>
                             (<PATH_VAR> | <PATH>)) ?
                             ((", " | "|") <PATH_VAR>
                             <PATH_OPERATOR>
                             (<PATH_VAR > | <PATH>)) * >

< ATTRIBUTE_LIST:         <INDENTIFIER> (", " <INDENTIFIER>)* >
< PATH_LIST:              <PATH_VAR> (", " <PATH_VAR>)* >
< TSR:                    <INDENTIFIER> "#" <FILE> >
```

Με βάση τις λέξεις της γλώσσας μπορούμε να ορίσουμε τις εκφράσεις των ερωτήσεων – πράξεων της TreeSQL. Μέσα σε [] βρίσκονται προαιρετικά τμήματα των εκφράσεων και στην προκειμένη περίπτωση δηλώνεται η δυνατότητα αποθήκευσης στο XML αρχείο <FILE> με όνομα <INDENTIFIER>.

1. Επιλογή

```
<SELECT> "(" <ATTRIBUTE_CONDITION> ")" (" <PATH_CONDITION> ") ("
<TSR>") [" <AS> "("<TSR>")"]
```

2. Προβολή

```
<PROJECT> "(" <ATTRIBUTE_LIST> ")" (" <PATH_LIST> ") (" <TSR> ") ["
<AS> "("<TSR>")"]
```

3. Καρτεσιανό γινόμενο

```
<CARTECIAN_PRODUCT> "(" <TSR> ")" (" <TSR> ") [" <AS> "("<TSR>")"]
```

4. Ένωση

```
<UNION> "(" <TSR> ")" (" <TSR> ") [" <AS> "("<TSR>")"]
```

5. Τομή

```
<INTERSECTION> "(" <TSR> ")" (" <TSR> ") [" <AS> "("<TSR>")"]
```

6. Διαφορά

```
<DIFFERENCE> "(" <TSR> ")" (" <TSR> ") [" <AS> "(" <TSR> ")" ] "
```

Επίσης ορίζονται οι εντολές `s2db`, `load` και `help`, για το φόρτωμα TSR σχημάτων από XML αρχείο στη βάση δεδομένων, την εκτέλεση αρχείου ερωτημάτων (query file) και εμφάνιση του εγχειριδίου χρήσης αντίστοιχα.

7. s2db

```
<S2DB> "(" <TSR> ")" "
```

8. load

```
<LOAD> "(" <IDENTIFIER> ".qry) "
```

9. help

```
<HELP>
```

5.2.2.2 Συντακτική ανάλυση – Χειρισμός εντολών

Ο πυρήνας του συντακτικού αναλυτή είναι η συνάρτηση `parseExpression`, η οποία αναλαμβάνει να αναγνωρίσει την εντολή που εισήγαγε ο χρήστης και να καλέσει την αντίστοιχη συνάρτηση για μια από τις λειτουργίες που προσφέρονται. Η διαδικασία αυτή βρίσκεται σε ένα ατέρμονο βρόχο και τελειώνει μόνο με τη εντολή `exit`. Κάθε εντολή του μεταγλωττιστή υλοποιείται από μία συνάρτηση, στο σώμα της οποίας καλούνται συναρτήσεις που με περαιτέρω ανάλυση συλλέγουν τις παραμέτρους της εντολής. Στη συνέχεια καλούνται συναρτήσεις άλλων εφαρμογών όπως του διαχειριστή XML αρχείου και της βάσης δεδομένων και στην περίπτωση των εντολών που αντιστοιχούν στις ερωτήσεις της γλώσσας `TreeSQL`, οι συναρτήσεις της εφαρμογής ερωτήσεων – πράξεων.

5.2.3 Περιγραφή κλάσεων

Στην ενότητα αυτή παρατίθενται οι κλάσεις που υλοποιούν τις εφαρμογές του συστήματος. Για κάθε κλάση παρουσιάζεται το UML διάγραμμα της με σύντομη αναφορά στα πεδία και τις μεθόδους της. Μια σημαντική παρατήρηση για τα διαγράμματα κλάσεων είναι ότι το σύμβολο `+` ορίζει τον προσδιοριστή πρόσβασης (access modifier) `public`, ενώ το `-` τον προσδιοριστή `private`.

5.2.3.1 *public class Attribute*

Η κλάση *Attribute* υλοποιεί μία ιδιότητα του κόμβου – αντικειμένου.

Πεδίο

- `public String name`

Το όνομα της ιδιότητας.

- `Public String type`

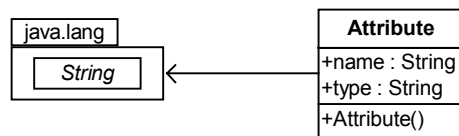
Ο τύπος της ιδιότητας. Μπορεί να είναι `text`, δηλαδή κείμενο ή `integer`, `float` για αριθμητικό.

Μέθοδοι

- `public Attribute(String name, String type)`

Κατασκευαστής αντικειμένων της κλάσης. Οι παράμετροι `name` και `type` αρχικοποιούν τα αντίστοιχα πεδία.

Στο Σχ. 5.2 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.2 -

5.2.3.2 *public class Path*

Χρησιμοποιώντας την κλάση *Path* ορίζεται το AND μονοπάτι που οδηγεί σε κάποιο κόμβο – αντικείμενο.

Πεδίο

- `private String image`

Απεικόνιση του μονοπατιού `image` σε μορφή *String*.

Μέθοδοι

- `public boolean equal(Path path)`

Τελεστής σύγκρισης μονοπατιών ισότητα.

- `public boolean looseHyperset(Path path)`

Τελεστής σύγκρισης μονοπατιών χαλαρό υπερσύνολο.

- `public boolean looseSubset(Path path)`

Τελεστής σύγκρισης μονοπατιών χαλαρό υποσύνολο.

- `public boolean strictHyperset(Path path)`
Τελεστής σύγκρισης μονοπατιών αυστηρό υπερσύνολο.

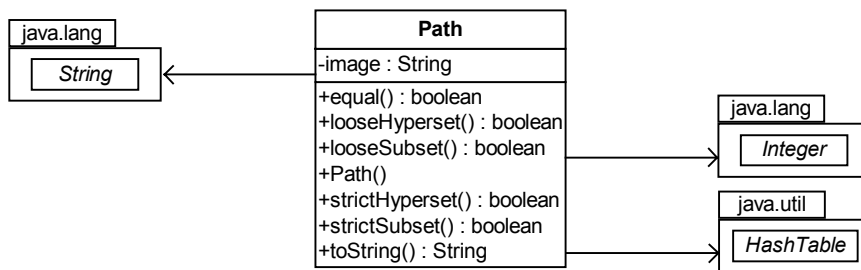
- `public boolean strictSubset(Path path)`
Τελεστής σύγκρισης μονοπατιών αυστηρό υποσύνολο.

- `public String toString()`
Προσπέλαση String απεικόνισης του μονοπατιού.

- `public Path(String image)`

Κατασκευαστής αντικειμένων της κλάσης. Η παράμετρος `image` αρχικοποιεί το πεδίο `image` της κλάσης.

Στο Σχ. 5.3 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.3 -

5.2.3.3 *public class OR*

Η κλάση *OR* υλοποιεί την OR συνιστώσα κάθε σχήματος TSR.

Πεδία

- `private Vector vAND`

Διάνυσμα που περιέχει τα AND μονοπάτια που περιέχονται στη συνιστώσα. Αποτελείται από αντικείμενα της κλάσης *Path*.

Μέθοδοι

- `public void addAND(Path path)`

Προσθήκη του AND μονοπατιού `path` στην OR συνιστώσα.

- `public int countANDs()`

Καταμέτρηση περιεχόμενων AND μονοπατιών.

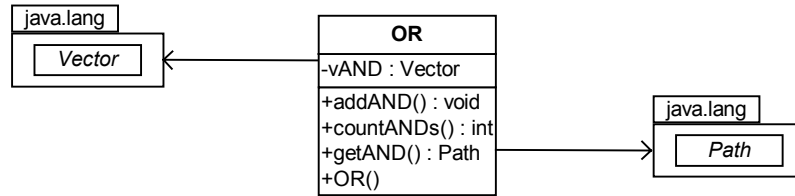
- `public Path getAND(int index)`

Προσπέλαση του `index` αριστερότερου AND μονοπατιού της συνιστώσας.

- `public OR()`

Κατασκευαστής αντικειμένων της κλάσης.

Στο Σχ. 5.4 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.4 -

5.2.3.4 *public class Item*

Χρησιμοποιώντας την κλάση *Item* ορίζεται ο κόμβος – αντικείμενο μιας TSR.

Πεδία

- `private Vector vAttributes`

Διάνυσμα που περιέχει τις ιδιότητες του κόμβου – αντικείμενου. Αποτελείται από αντικείμενα της κλάσης *Attribute*.

- `private Vector vTuples`

Διάνυσμα που περιέχει τις εγγραφές του κόμβου – αντικείμενου.

Μέθοδοι

- `public void addAttributes(Attribute attr)`

Προσθήκη της ιδιότητας `attr` στον κόμβο αντικείμενο.

- `public void addTuple(String tuple)`

Προσθήκη της εγγραφής `tuple` στον κόμβο αντικείμενο.

- `public int countAttributes()`

Καταμέτρηση των ιδιοτήτων.

- `public int countTuples()`

Καταμέτρηση των εγγραφών.

- `public boolean existAttribute(Attribute attr)`

Αναζήτηση της ιδιότητας `attr` στον κόμβο – αντικείμενο.

- `public boolean existAttributeByName(String attrName)`

Αναζήτηση της ιδιότητας με όνομα `attrName` στον κόμβο – αντικείμενο.

- `public boolean existTuple(String tuple)`

Αναζήτηση της εγγραφής `tuple` στο κόμβο – αντικείμενο.

- `public Attribute getAttribute(int index)`

Προσπέλαση της `index` ιδιότητας.

- `public int getAttributeIndexByName(String attrName)`

Εύρεση του αύξοντα αριθμού της ιδιότητας με όνομα `attrName` στον κόμβο – αντικείμενο.

- `public String getTuple(int index)`

Προσπέλαση της `index` εγγραφής.

- `public Item()`

Κατασκευαστής αντικειμένων της κλάσης.

- `public boolean sameAttributes(TSR other)`

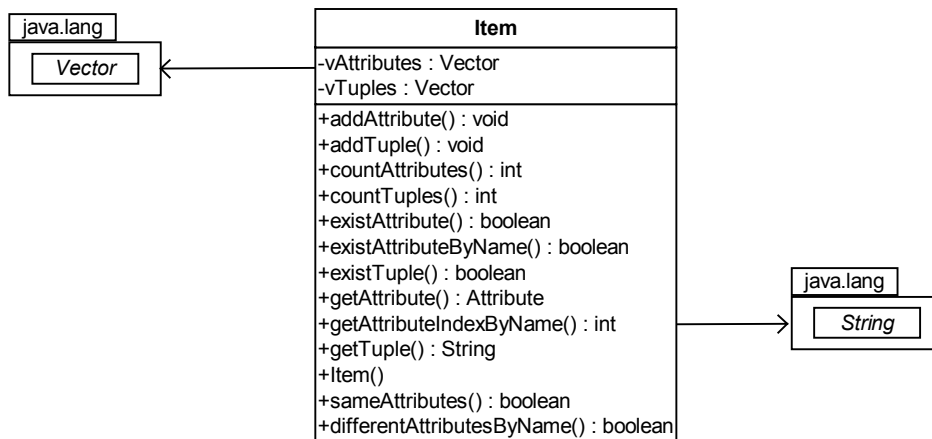
Σύγκριση των ιδιοτήτων του αντικειμένου μ' αυτές του αντικειμένου της `other` TSR.

Στην περίπτωση που οι ιδιότητες είναι ακριβώς ίδιες επιστρέφεται `true`.

- `public boolean differentAttributesByName(TSR other)`

Σύγκριση των ονομάτων των ιδιοτήτων του αντικειμένου μ' αυτές του αντικειμένου της `other` TSR. Στην περίπτωση που οι ιδιότητες είναι όλες διαφορετικές μεταξύ τους επιστρέφεται `true`.

Στο Σχ. 5.5 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.5 -

5.2.3.5 *public class* TSR

Με την κλάση *TSR* ορίζεται ένα σήμα *TSR*.

Πεδία

- `public String file`

Το όνομα του XML αρχείου απ' όπου προέρχεται το TSR σχήμα.

- `public Item item`

Ο κόμβος – αντικείμενο.

- `public String name`

Το όνομα του TSR σχήματος.

- `private Vector vOR`

Διάνυσμα που περιέχει τις OR συνιστώσες του TSR σχήματος. Αποτελείται από αντικείμενα της κλάσης *OR*.

Μέθοδοι

- `public void addOR(OR or)`

Προσθήκη της OR *or* συνιστώσας στο TSR σχήμα.

- `public int countORs()`

Καταμέτρηση των OR συνιστωσών του TSR σχήματος.

- `public boolean existANDindex(int index)`

Αναζήτηση του *index* αριστερότερου AND μονοπατιού. Με τη μέθοδο αυτή διαπιστώνεται αν το *index* αριστερότερο μονοπάτι υπάρχει στη OR συνιστώσα με τα περισσότερα μονοπάτια, άρα και σε ολόκληρο το TSR σχήμα.

- `public OR getMaxOR()`

Προσπέλαση της μεγαλύτερης OR συνιστώσας του TSR σχήματος, δηλαδή αυτής με τα περισσότερα AND μονοπάτια.

- `public OR getOR(int index)`

Προσπέλαση της *index* OR συνιστώσας του TSR σχήματος.

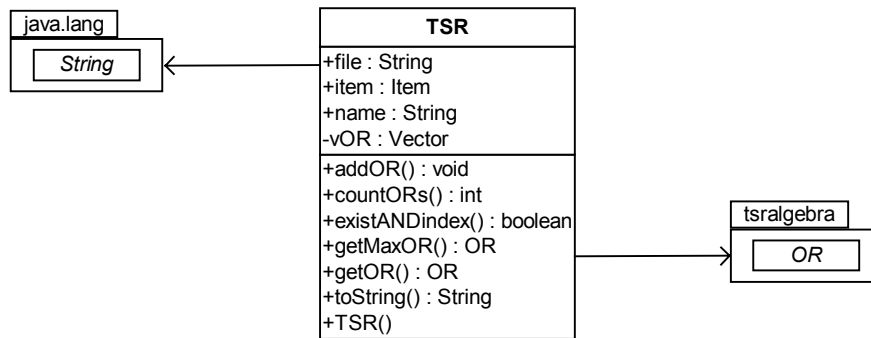
- `public String toString()`

Απεικόνιση του TSR σχήματος σύμφωνα με το σχήμα του XML αρχείου που χρησιμοποιείται ως αποθηκευτικό μέσο.

- `public TSR()`

Κατασκευαστής αντικειμένων της κλάσης.

Στο Σχ. 5.6 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.6 -

5.2.3.6 *public class Condition*

Στη σύνταξη των ερωτήσεων επιλογής δίνεται μια σειρά από συνθήκες ιδιοτήτων και μονοπατιών που αποτιμώνται πάνω στις εγγραφές και τα μονοπάτια του κόμβου – αντικείμενου αντίστοιχα. Η κλάση που ορίζει τέτοιες συνθήκες είναι η *Condition*. Η μορφή των συνθηκών μπορεί να είναι μεταβλητή τελεστής μεταβλητή ή μεταβλητή τελεστής τιμή, όπου μεταβλητή είναι είτε μεταβλητή ιδιότητας είτε μεταβλητή μονοπατιών.

Πεδία

- `public String operator`

Ο τελεστής της συνθήκης, που είναι τελεστής σύγκρισης ιδιοτήτων ή μονοπατιών.

- `public String previousBitwise`

Το πεδίο αυτό παίρνει τις τιμές 'AND' ή 'OR' ανάλογα με το πώς συνδέεται η συνθήκη με την αμέσως προηγούμενη.

- `public String term1`

Πρώτος όρος συνθήκης. Είναι πάντα μια μεταβλητή.

- `public String term2`

Δεύτερος όρος συνθήκης. Είναι ανάλογα με τον τύπο της συνθήκης είτε μεταβλητή είτε τιμή.

- `public boolean type`

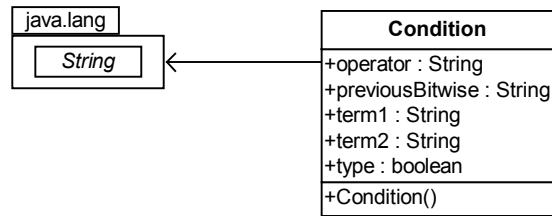
Ο τύπος της συνθήκης. Είναι `true` όταν πρόκειται για σύγκριση μεταβλητών και `false` όταν συγκρίνεται μεταβλητή με τιμή.

Μέθοδοι

- `public Condition(String term1, String op, String term2, boolean type, String previousBitwise)`

Κατασκευαστής αντικειμένων της κλάσης. Οι παράμετροι `term1`, `term2`, `type`, `previousBitwise` αρχικοποιούν τα αντίστοιχα πεδία.

Στο Σχ. 5.7 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.7 -

5.2.3.7 *public class TSRfunctions*

Η κλάση αυτή υλοποιεί τις ερωτήσεις – πράξεις της γλώσσας TreeSQuerL πάνω σε σχήματα – αντικείμενα της κλάσης *TSR*.

Πεδία

Μέθοδοι

- `public TSR select(Vector vAttributeCondition, Vector vPathCondition, TSR tsr)`

Η πράξη της επιλογής με λίστα συνθηκών ιδιοτήτων `vAttributeCondition` και μονοπατιών `vPathCondition` στο σχήμα `tsr`.

- `public TSR project(Vector vAttributeVariable, Vector vPathVariable, TSR tsr)`

Η πράξη της προβολής με λίστα ιδιοτήτων `vAttributeVariable` και μεταβλητών μονοπατιών `vPathVariable` στο σχήμα `tsr`.

- `public TSR cartecianProduct(TSR tsr1, TSR tsr2)`

Καρτεσιανό γινόμενο των σχημάτων `tsr1` και `tsr2`.

- `public TSR union(TSR tsr1, TSR tsr2)`

Ένωση των σχημάτων `tsr1` και `tsr2`.

- `public TSR intersection(TSR tsr1, TSR tsr2)`

Τομή των σχημάτων `tsr1` και `tsr2`.

- `public TSR difference(TSR tsr1, TSR tsr2)`

Διαφορά των σχημάτων `tsr1` και `tsr2`.

- `private boolean existAttributes(Vector vAttributeVariable, TSR tsr)`

Η συνάρτηση αυτή χρησιμοποιείται για τον έλεγχο ορθότητας παραμέτρων μιας πράξης. Ελέγχει αν το σχήμα `tsr` περιέχει τις ιδιότητες `vAttributeVariable`.

- `private boolean existPaths(Vector vPathVariable, TSR tsr)`

Η συνάρτηση αυτή χρησιμοποιείται για τον έλεγχο ορθότητας παραμέτρων μιας πράξης. Ελέγχει αν το σχήμα `tsr` περιέχει τις μεταβλητές μονοπατιών `vPathVariable`.

- `private boolean evaluate(String term1, String operator, String term2)`

Η συνάρτηση αυτή αποτιμά τη έκφραση `term1 operator term2`.

- `private boolean isNumber(String term)`

Η συνάρτηση αυτή καθορίζει αν ο όρος `term` είναι αριθμητικό ή αλφαριθμητικό.

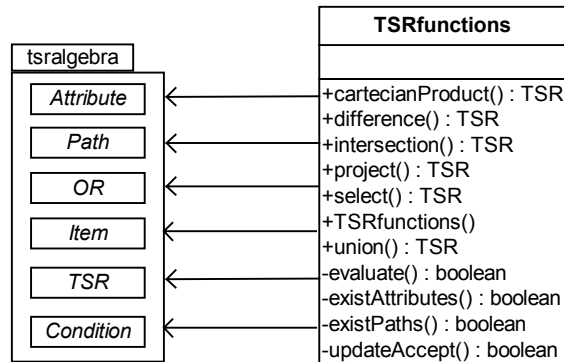
- `private boolean updateAccept(boolean accept, String previousBitwise, boolean term)`

Η συνάρτηση αυτή χρησιμοποιείται στον ολικό έλεγχο ισχύος της λίστας συνθηκών.

- `public TSRfunctions()`

Κατασκευαστής αντικειμένων της κλάσης.

Στο Σχ. 5.8 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.8 -

5.2.3.8 *public class XMLschema*

Στην κλάση αυτή υλοποιείται η εφαρμογή διαχείρισης XML αρχείου.

Πεδία

- `private SAXReader parser`

Ο `parser` που θα προσπελάσει τα στοιχεία του XML αρχείου.

- private Document source

Το αντικείμενο της κλάσης Document που δημιουργείται από το XML αρχείο που θα διαχειριστεί.

Μέθοδοι

- public void addTSR(TSR tsr)

Προσθήκη του σχήματος tsr στο XML αρχείο.

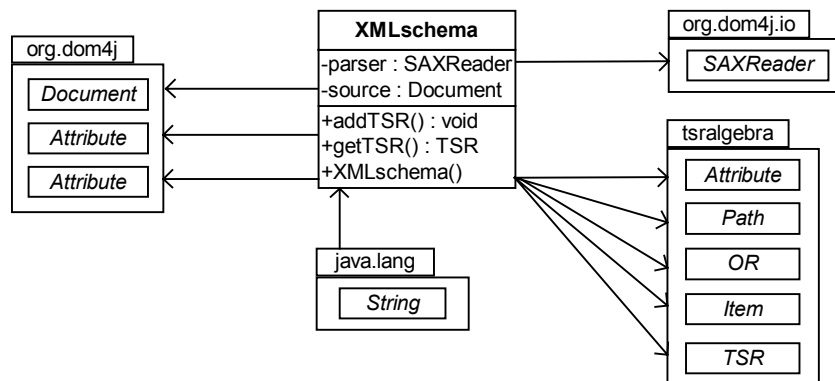
- public TSR getTSR(String XMLfile, String TSRname)

Προσπέλαση TSRname TSR σχήματος στο XML αρχείο XMLfile.

- public XMLschema()

Κατασκευαστής αντικειμένων της κλάσης.

Στο Σχ. 5.9 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.9 -

5.2.3.9 public class DBschema

Στην κλάση αυτή υλοποιείται η εφαρμογή διαχείρισης της βάσης δεδομένων.

Πεδία

- private Connection pConn

Η σύνδεση με τη βάση δεδομένων που χρειάζεται για την προσπέλασή της.

Μέθοδοι

- public void addTSR(TSR tsr)

Προσθήκη του σχήματος tsr στη βάση δεδομένων.

- public DBschema()

Κατασκευαστής αντικειμένων της κλάσης.

- `public boolean existTSR(String strTSRFilename, String strTSRName)`

Αναζήτηση του TSR σχήματος με χαρακτηριστικά `strTSRFilename` και `strTSRName`.

- `public TSR getTSR(String strTSRFilename, String strTSRName)`

Προσπέλαση TSR σχήματος στη βάση με χαρακτηριστικά `strTSRFilename` και `strTSRName`.

- `public void execQuery(String strQuery)`

Εκτέλεση του ερωτήματος `strQuery` στη βάση δεδομένων.

- `public int getNextTid()`

Εύρεση του επόμενου κωδικού που μπορεί να δοθεί σε μια νέα καταχώρηση TSR σχήματος στη βάση δεδομένων.

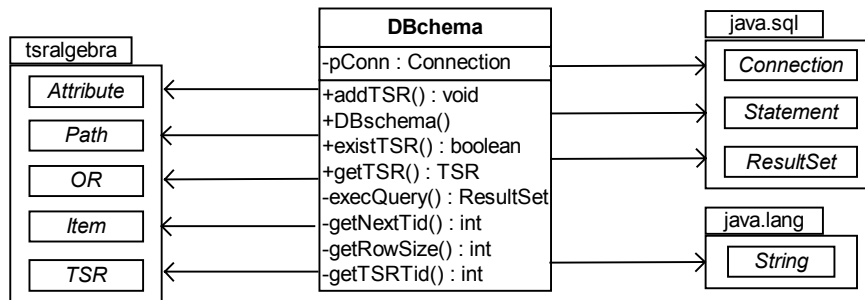
- `public int getRowSize(ResultSet rs)`

Εύρεση του μεγέθους μιας γραμμής του `ResultSet rs`.

- `public int getTSRTid(String strTSRFilename, String strTSRName)`

Εύρεση του κωδικού του TSR σχήματος με χαρακτηριστικά `strTSRFilename` και `strTSRName`.

Στο Σχ. 5.10 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.10 -

5.2.3.10 *public class TSRxmlDialog*

Η κλάση αυτή ορίζει ένα βοηθητικό παράθυρο στην εφαρμογή παρουσίασης TSR σχημάτων. Αναλαμβάνει τη συλλογή στοιχείων για την αποθήκευση του σχήματος σε XML αρχείο.

Πεδία

- `public boolean onOK`

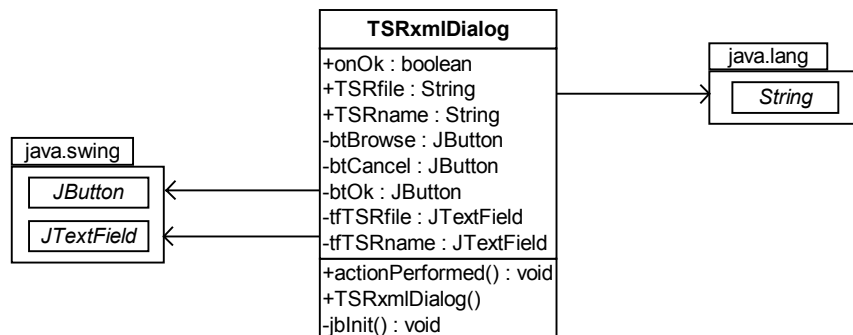
Καθορίζει αν ο χρήστης ζήτησε την εισαγωγή του σχήματος στο XML αρχείο.

- `public String TSRfile`
Το όνομα του XML αρχείου όπου θα αποθηκευτεί το TSR σχήμα.
- `public String TSRname`
Το όνομα του TSR σχήματος που θα αποθηκευτεί.
- `private JButton btBrowse`
Το κουμπί Browse.
- `private JButton btCancel`
Το κουμπί Cancel.
- `private JButton btOk`
Το κουμπί Ok.
- `private JTextField tfTSRfile`
Εδώ αναγράφεται το όνομα του XML αρχείου όπου θα αποθηκευτεί το TSR σχήμα.
- `private JTextField tfTSRname`
Εδώ εισάγεται το όνομα του TSR σχήματος που θα αποθηκευτεί.

Μέθοδοι

- `public void actionPerformed(ActionEvent e)`
Χειριστής των ενεργειών του χρήστη.
- `public TSRxmlDialog()`
Κατασκευαστής αντικειμένων της κλάσης.
- `private void jbInit()`
Η μέθοδος αυτή αρχικοποιεί το παράθυρο.

Στο Σχ. 5.11 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.11 -

5.2.3.11 *public class TSRdbDialog*

Η κλάση αυτή ορίζει ένα βοηθητικό παράθυρο στην εφαρμογή παρουσίασης TSR σχημάτων. Αναλαμβάνει τη συλλογή στοιχείων για την αποθήκευση του σχήματος στη βάση δεδομένων.

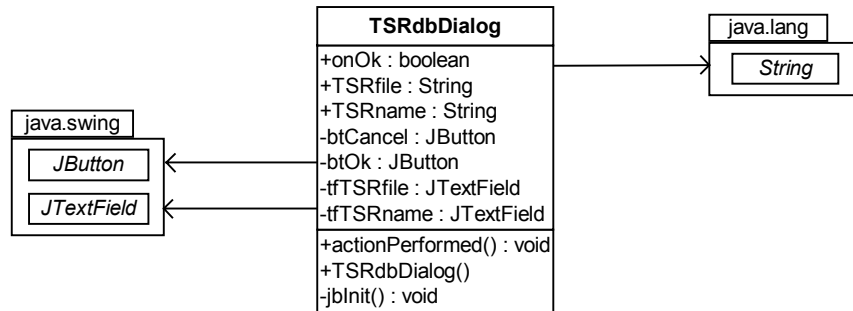
Πεδία

- `public boolean onOK`
Καθορίζει αν ο χρήστης ζήτησε την εισαγωγή του σχήματος στη βάση δεδομένων.
- `public String TSRfile`
Το όνομα του XML αρχείου απ' όπου προέρχεται το TSR σχήμα.
- `public String TSRname`
Το όνομα του TSR σχήματος που θα αποθηκευτεί.
- `private JButton btCancel`
Το κουμπί Cancel.
- `private JButton btOk`
Το κουμπί Ok.
- `private JTextField tfTSRfile`
Εδώ εισάγεται το όνομα του XML αρχείου απ' όπου προέρχεται το TSR σχήμα.
- `private JTextField tfTSRname`
Εδώ εισάγεται το όνομα του TSR σχήματος που θα αποθηκευτεί.

Μέθοδοι

- `public void actionPerformed(ActionEvent e)`
Χειριστής των ενεργειών του χρήστη.
- `public TSRdbDialog()`
Κατασκευαστής αντικειμένων της κλάσης.
- `private void jbInit()`
Η μέθοδος αυτή αρχικοποιεί το παράθυρο.

Στο Σχ. 5.12 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.12 -

5.2.3.12 public class TSRviewer

Η κλάση αυτή υλοποιεί το αρχικό παράθυρο της εφαρμογής παρουσίασης σχημάτων TSR.

Πεδία

- private JButton btCopy

Το κουμπί Copy.

- private JButton btExit

Το κουμπί Exit.

- private JButton btSave2DB

Το κουμπί Save to Database.

- private JButton btSave2XML

Το κουμπί Save to XML αρχείο.

- private JTextArea editor

Εδώ απεικονίζεται η μορφή του TSR σχήματος.

- private JToolBar jToolBar

Η Toolbar με τα κουμπιά λειτουργιών.

- private Connection pConn

Η σύνδεση με τη βάση δεδομένων που χρειάζεται για την προσπέλασή της.

- private JScrollPane scroller

- private TSR tsr

Το TSR σχήμα που απεικονίζεται.

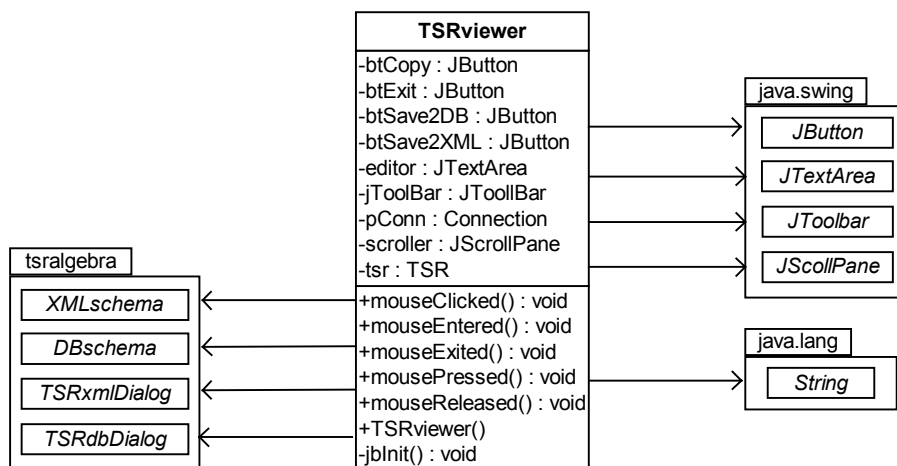
Μέθοδοι

- public void mouseClicked(MouseEvent event)

Χειριστής των ενεργειών του ποντικιού.

- `public void mouseEntered(MouseEvent event)`
Χειριστής των ενεργειών του ποντικιού.
- `public void mouseExited(MouseEvent event)`
Χειριστής των ενεργειών του ποντικιού.
- `public void mousePressed(MouseEvent event)`
Χειριστής των ενεργειών του ποντικιού.
- `public void mouseReleased(MouseEvent event)`
Χειριστής των ενεργειών του ποντικιού.
- `public TSRviewer()`
Κατασκευαστής αντικειμένων της κλάσης.
- `private void jbInit()`
Η μέθοδος αυτή αρχικοποιεί το παράθυρο.

Στο Σχ. 5.13 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.13 -

5.2.3.13 *public class HelpViewer*

Η κλάση αυτή υλοποιεί το παράθυρο με τον εγχειρίδιο χρήσης της εφαρμογής του μεταγλωττιστή. Πρόκειται για απεικόνιση HTML εγγράφων που περιέχουν τη σύνταξη κάθε εντολής μαζί με ένα παράδειγμα.

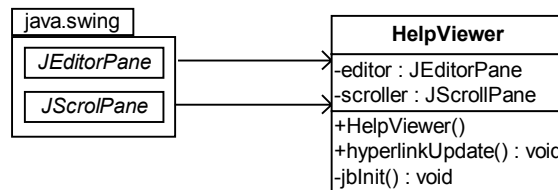
Πεδία

- `private JEditorPane editor`
- `private JScrollPane scroller`

Μέθοδοι

- `public HelpViewer()`
Κατασκευαστής αντικειμένων της κλάσης.
- `public void hyperlinkUpdate(HyperlinkEvent event)`
Πλοήγηση links HTML σελίδων.
- `private void jbInit()`
Η μέθοδος αυτή αρχικοποιεί το παράθυρο.

Στο Σχ. 5.14 φαίνεται το UML διάγραμμα της κλάσης.



- Σχ. 5.14 -

5.2.4 Αλγόριθμοι

Στην παράγραφο αυτή παρουσιάζονται οι κυριότεροι αλγόριθμοι του συστήματος. Ανάμεσά τους αυτοί της διαχείρισης των αποθηκευτικών μέσων, των ερωτήσεων – πράξεων της γλώσσας TreeSQuerL και αυτοί του αυστηρού και του χαλαρού υποσυνόλου από τη σύγκριση μονοπατιών.

Για την καλύτερη κατανόηση των αλγόριθμων παρουσιάζεται η διαδικασία που επιτελούν μέσα από παραδείγματα που εφαρμόζονται στα παρακάτω TSR σχήματα:

```
---BH.xml
<tsr name="DigitalCameras">
  <or>
    <and>/digital/cameras</and>
  </or>
  <item>
    <attribute name="brand" type="text"/>
    <attribute name="model" type="text"/>
    <attribute name="CCD" type="float"/>
    <attribute name="price" type="float"/>
    <attribute name="otherPrice" type="float"/>
    <tuple>'Canon', 'Powershot A60', '2', '229', '230'</tuple>
    <tuple>'Hewlett Packard', 'Photosmart 635', '2.1', '216',
      '230'</tuple>
    <tuple>'Kodak', 'EasyShare CX6230 Zoom', '2', '205',
      '199.95'</tuple>
    <tuple>'Konica', 'Revio C2', '1.2', '199.95',
```

```

        '199.95'</tuple>
    <tuple>'Minolta', 'DiMAGE E 223', '2', '199',
        '199'</tuple>
    <tuple>'Minolta', 'DiMAGE X20', '2', '256',
        '249.95'</tuple>
    <tuple>'Nikon', 'Coolpix 2100', '2', '240.45',
        '249.95'</tuple>
    <tuple>'Olympus', 'Camedia D 390', '2', '180.77',
        '179.95'</tuple>
</item>
</tsr>

```

---BH.xml

```

<tsr name="SLRcameras">
    <or>
        <and>/photo/35mmSystems/SLRcameras</and>
    </or>
    <item>
        <attribute name="sbrand" type="text"/>
        <attribute name="smodel" type="text"/>
        <attribute name="sprice" type="float"/>
        <tuple>'Canon', 'EOS 3', '849.95'</tuple>
        <tuple>'Canon', 'EOS Rebel Ti', '254.95'</tuple>
        <tuple>'Minolta', 'Maxxum 3 QD', '384.95'</tuple>
        <tuple>'Minolta', 'Maxxum 4 QD', '164.95'</tuple>
        <tuple>'Minolta', 'Maxxum 9 QD', '1099.95'</tuple>
        <tuple>'Minolta', 'X 370s', '149.95'</tuple>
        <tuple>'Nikon', 'N65', '199.95'</tuple>
        <tuple>'Nikon', 'F80', '349.95'</tuple>
        <tuple>'Pentax', 'ZX M', '149.95'</tuple>
        <tuple>'Sigma', 'SA 7', '249.95'</tuple>
    </item>
</tsr>

```

---RitzCameras.xml

```

<tsr name="SLRcameras">
    <or>
        <and>/SLRcameras</and>
    </or>
    <item>
        <attribute name="sbrand" type="text"/>
        <attribute name="smodel" type="text"/>
        <attribute name="sprice" type="float"/>
        <tuple>'Canon', 'EOS 3', '999.95'</tuple>
        <tuple>'Canon', 'EOS Rebel Ti', '264.95'</tuple>
        <tuple>'Canon', 'EOS Elan 7', '424.95'</tuple>
        <tuple>'Minolta', 'Maxxum 3 QD', '139.95'</tuple>
        <tuple>'Minolta', 'Maxxum 4 QD', '194.95'</tuple>
    </item>
</tsr>

```

```

        <tuple>'Minolta', 'Maxxum 7 QD', '599.99'</tuple>
        <tuple>'Nikon', 'N65', '199.95'</tuple>
        <tuple>'Pentax', 'ZX M', '149.95'</tuple>
    </item>
</tsr>

```

5.2.4.1 Ανάγνωση TSR σχήματος από τη βάση δεδομένων

Μία από τις δύο βασικές λειτουργίες της εφαρμογής διαχείρισης βάσης δεδομένων είναι η προσπέλαση TSR σχημάτων και η ανάγνωση τους. Ο αλγόριθμος ανάγνωσης σχημάτων από τη βάση στηρίζεται στην υποβολή τριών βασικών SQL ερωτήσεων στο σύστημα διαχείρισης της MySQL. Το όνομα του σχήματος και το όνομα του αρχείου απ' όπου προέρχεται είναι τα γνωστά στοιχεία, τα οποία χρησιμοποιούνται για την προσπέλαση των στοιχείων του σχήματος. Απ' αυτά θα προέρθει ο κωδικός `tid` που έχει η εγγραφή του σχήματος στη βάση, στον πίνακα `tsr`. Ο κόμβος – αντικείμενο του σχήματος βρίσκεται από τον αντίστοιχο πίνακα `ri`, ενώ το σύνολο των μονοπατιών από τον πίνακα `rp`. Για παράδειγμα έστω ότι θέλουμε να προσπελάσουμε το σχήμα `DigitalCameras` που προέρχεται από το αρχείο `BH.xml`. Αρχικά, βρίσκουμε τον κωδικό `tid` της εγγραφής στη βάση υποβάλλοντας το ερώτημα:

```
SELECT tid FROM tsr WHERE name = 'DigitalCameras' AND file = 'BH.xml';
```

Έστω ότι ο κωδικός `tid` του σχήματος είναι 4. Έχοντας αυτήν την πληροφορία μπορούμε να ζητήσουμε τα υπόλοιπα στοιχεία του σχήματος. Ξεκινάμε από τις ιδιότητες, οι οποίες είναι τα πεδία του πίνακα `ri+tid`, δηλαδή του `ri4`. Με το ερώτημα:

```
DESCRIBE ri4;
```

παίρνουμε τις ιδιότητες του TSR σχήματος, τις οποίες εισάγουμε στο `vAttributes` του κόμβου – αντικειμένου `resultTSR.item` του αποτελέσματος. Για να συλλέξουμε τις εγγραφές ζητάμε από τον πίνακα `ri4` να μας φέρει όλες τις πλειάδες του, υποβάλλοντας το ερώτημα:

```
SELECT * FROM ri4;
```

Κάθε πλειάδα του αποτελέσματος της ερώτησης τοποθετείται στο `vTuples` του `resultTSR.item`. Στη συνέχεια, για να προσπελάσουμε το σύνολο μονοπατιών του σχήματος συντάσσουμε την ερώτηση:

```
SELECT orid, andid, path FROM rp WHERE tid = 4;
```

Για κάθε μία από τις τιμές – μονοπάτια του αποτελέσματος συγκρίνουμε την τιμή του `orid` με την αντίστοιχη τιμή για το προηγούμενο μονοπάτι και συμπεράνουμε αν ανήκουν στην ίδια OR συνιστώσα ή όχι. Αν ανήκουν, προσθέτουμε το μονοπάτι στο `vAND` της `resultOR`, αλλιώς προσθέτουμε τη `resultOR` στο διάνυσμα `vOR` με τις συνιστώσες του αποτελέσματος και κατασκευάζουμε μια καινούρια `resultOR`.

5.2.4.2 Ανάγνωση TSR σχήματος από XML αρχείο

Η ανάγνωση TSR σχήματος από XML αρχείο στηρίζεται την ανάλυση (parsing) XML εγγράφου από το DOM αναλυτή (parser) του πακέτου dom4j. Ο αλγόριθμος δέχεται ως είσοδο το όνομα του σχήματος που θέλουμε να προσπελάσουμε και το όνομα του XML αρχείου που θ' αναλυθεί. Εφόσον το XML αρχείο υπάρχει, χρησιμοποιώντας κατά βάθος αναζήτηση κατασκευάζουμε τα μονοπάτια (tag <and>) που εισάγονται στο vAND κάθε OR συνιστώσας και στη συνέχεια εισάγεται κάθε OR συνιστώσα (tag <or>) στο vOR του σχήματος. Έπειτα εντοπίζονται οι ιδιότητες του σχήματος (tag <attribute>) και εισάγονται στο vAttributes του κόμβου – αντικείμενου του αποτελέσματος. Τέλος, με παρόμοιο τρόπο κάθε εγγραφή (tag <tuple>) εισάγεται στο vTuples του κόμβου – αντικείμενου του αποτελέσματος.

5.2.4.3 Αποθήκευση TSR σχήματος στη βάση δεδομένων

Η αποθήκευση XML σχημάτων στη βάση δεδομένων είναι λειτουργία της εφαρμογής διαχείρισης της βάσης δεδομένων. Ο αλγόριθμος αποθήκευσης σχημάτων στη βάση στηρίζεται στην υποβολή SQL ερωτήσεων στο σύστημα διαχείρισης της MySQL και δέχεται ως είσοδο ένα αντικείμενο `tsr` της κλάσης `TSR`, το οποίο θα αποθηκευτεί στη βάση. Έστω για παράδειγμα ότι θέλουμε να αποθηκεύσουμε το σχήμα `DigitalCameras` που προέρχεται από το αρχείο `BH.xml`. Αρχικά, θα πρέπει να ελεγχθεί αν το σχήμα `BH.xml#DigitalCameras` υπάρχει στη βάση δεδομένων. Αυτή η πληροφορία θα προέλθει από τον πίνακα `tsr`, που περιέχει στοιχεία για όλα τα αποθηκευμένα σχήματα, με σύνταξη του παρακάτω SQL ερωτήματος:

```
SELECT count(tid) FROM tsr WHERE file = 'BH.xml' AND name = 'DigitalCameras';
```

Αν ο αριθμός το `tid` καταχωρήσεων στη βάση είναι διάφορος του μηδενός τότε ένα σχήμα με ίδιο όνομα και ίδιο αρχείο είναι ήδη αποθηκευμένο, οπότε δεν μπορεί να συνεχιστεί η διαδικασία. Έστω ότι κάτι τέτοιο δε συμβαίνει στο συγκεκριμένο παράδειγμα. Έπειτα μέσα από τον πίνακα `tid_value` βρίσκουμε την επόμενη τιμή του κωδικού `tid` για το `tsr`, αυξάνοντας κατά ένα την τελευταία. Για να βρούμε την τελευταία τιμή υποβάλλουμε το ερώτημα:

```
SELECT current FROM tid_value;
```

Έστω ότι 6 είναι ο κωδικός του προηγούμενου σχήματος που αποθηκεύτηκε. Οπότε ο καινούριος κωδικός είναι 7.

```
UPDATE tid_value SET current = 7;
```

Στη συνέχεια εισάγουμε τα στοιχεία του νέου σχήματος στον πίνακα `tsr`:

```
INSERT INTO tsr VALUES (7, 'DigitalCameras', 'BH.xml');
```


Για την κατασκευή και το γέμισμα του πίνακα `ri7` που αποθηκεύει τον κόμβο – αντικείμενο του `tsr` υποβάλλουμε ένα ερώτημα κατασκευής πίνακα χρησιμοποιώντας τις ιδιότητες του `vAttributes` του `tsr.item` και τόσα ερωτήματα εισαγωγής πλειάδων όσα είναι οι εγγραφές του `vTuples` του `tsr.item`. Οπότε, για την κατασκευή του `ri7` έχουμε το:

```
CREATE TABLE ri7
  (brand text NOT NULL,
   model text NOT NULL,
   CCD float NOT NULL,
   price float NOT NULL,
   otherPrice NOT NULL);
```

και για παράδειγμα για την εγγραφή 'Canon', 'Powershot A60', '2', '229', '230' το:

```
INSERT INTO ri7 VALUES ('Canon', 'Powershot A60', '2', '229', '230');
```

Τέλος για το σύνολο των μονοπατιών στον πίνακα `rp`, για κάθε AND μονοπάτι που περιέχεται στο `vAND` κάθε OR συνιστώσας του `vOR` του `tsr` υποβάλλεται ένα ερώτημα εισαγωγής πλειάδας στον `rp`. Έτσι, στο συγκεκριμένο παράδειγμα έχουμε το ερώτημα:

```
INSERT INTO rp VALUES (7, 1, 1, '/digital/cameras');
```

5.2.4.4 Αποθήκευση TSR σχήματος σε XML αρχείο

Για την αποθήκευση TSR σχημάτων σε XML αρχείο δε χρησιμοποιήθηκε κάποιος αναλυτής XML εγγράφου όπως DOM ή SAX, αλλά τα `InputStreams` της Java και η `string` απεικόνιση που προσφέρει η μέθοδος `toString()` της κλάσης `TSR`. Ο αλγόριθμος δέχεται ως είσοδο ένα αντικείμενο `tsr` της κλάσης `TSR`. Διακρίνονται δύο περιπτώσεις. Στη πρώτη περίπτωση το XML αρχείο που προσδιορίζει το `tsr.file` υπάρχει, οπότε πρέπει να εισαχθεί η `string` απεικόνιση του `tsr` πριν από το tag `</root>`. Έτσι, διαβάζονται αρχικά τα περιεχόμενα του αρχείου και αποθηκεύονται σε ένα αντικείμενο `String` και στη συνέχεια αντικαθίσταται το `string </root>` με την απεικόνιση `tsr.toString()` συν το `string </root>`. Στη δεύτερη περίπτωση το αρχείο δεν υπάρχει, οπότε δημιουργείται από το σύστημα και προστίθενται η επικεφαλίδα `<?xml version="1.0" ?>`, το tag `<root>` της ρίζας, η `string` απεικόνιση `tsr.toString()` και το tag `</root>` κλείσιμο της ρίζας.

5.2.4.5 Επιλογή – Select

Η πράξη της επιλογής εφαρμόζεται σε ένα `TSR` σχήμα. Σύμφωνα με τον ορισμό της γλώσσας `TreeSQL`, χρειάζονται μία λίστα με συνθήκες ιδιοτήτων και μία δεύτερη λίστα με συνθήκες μονοπατιών του σχήματος. Ο αλγόριθμος στηρίζεται στην επιλογή των εγγραφών που ικανοποιούν τις δοσμένες συνθήκες ιδιοτήτων και των `OR` συνιστωσών των οποίων τα `AND`

μονοπάτια ικανοποιούν τις αντίστοιχες δοσμένες συνθήκες μονοπατιών. Έστω ότι συντάσσεται η ερώτηση:

```
s(CCD > '1.5', price > otherPrice) ($1 EQ '/digital/cameras')  
(BH.xml#DigitalCameras)
```

Αρχικά η εφαρμογή διαχείρισης XML αρχείων ή βάσης δεδομένων προσπελαύνει το σχήμα BH.xml#DigitalCameras και κατασκευάζει ένα αντικείμενο `tsr` της κλάσης `TSR`. Σύμφωνα με την ερώτηση κατασκευάζονται οι λίστες παραμέτρων `vAttributeCondition` και `vPathCondition` που περιέχουν αντικείμενα κλάσης `Condition`. Για παράδειγμα, για τη συνθήκη `price > otherPrice` το αντικείμενο της `Condition` έχει την εξής μορφή:

```
term1 = "price"  
term2 = "otherPrice"  
type = true  
operator = ">"  
previousBitwise = "AND"
```

Αν δεν υπήρχαν καθόλου συνθήκες ιδιοτήτων, τότε ο κόμβος – αντικείμενο `resultTSR` θα ήταν αντιγραφή του αρχικού `tsr.item`. Αντίστοιχα, αν δεν υπήρχαν καθόλου συνθήκες μονοπατιών, το `vOR` του `resultTSR` θα ήταν το ίδιο με το αρχικό `vOR` του `tsr`. Στη συνέχεια, ελέγχεται κατά πόσο ο πρώτος όρος κάθε συνθήκης είναι ιδιότητα του `vAttributes` του αρχικού `tsr.item`. Στην περίπτωση που η συνθήκη είναι `true`, δηλαδή ιδιότητα τελεστής ιδιότητα, εξετάζεται αν και ο δεύτερος όρος είναι ιδιότητα του `vAttributes`. Στο συγκεκριμένο παράδειγμα όλες οι ιδιότητες `price`, `otherPrice` και `CCD` περιέχονται στο `vAttributes` του `tsr.item`. Έτσι, προστίθενται οι ιδιότητες του `tsr` στο σχήμα του αποτελέσματος `resultTSR` και συγκεκριμένα στο διάνυσμα `vAttributes`. Οπότε το `vAttributes` θα έχει τη μορφή:

```
[name = "brand" type = "text" ,  
 name = "brand" type = "text" ,  
 name = "CCD" type = "float" ,  
 name = "price" type = "float" ,  
 name = "otherPrice" type = "float"]
```

Για κάθε εγγραφή του αρχικού σχήματος επιλέγουμε αυτές που ικανοποιούν τις δοσμένες συνθήκες `CCD > '1.5'` και `price > otherPrice`. Για να γίνει η επιλογή αυτή ελέγχουμε αν η εγγραφή ικανοποιεί κάθε μία συνθήκη ξεχωριστά και μετά, σύμφωνα με τη μορφή της λίστας συνθηκών `vAttributeCondition`, αν τις ικανοποιεί και ταυτόχρονα. Έτσι για παράδειγμα ελέγχουμε την εγγραφή: 'Nikon', 'Coolpix 2100', '2', '240.45', '249.95'. Χωρίζουμε την εγγραφή σε τιμές με βάση το χαρακτήρα κόμμα (,) και ξεκινάμε απ' τη συνθήκη `CCD > '1.5'`. Εξετάζουμε αν η αντίστοιχη τιμή της ιδιότητας στην εγγραφή, δηλαδή η '2', ικανοποιεί τη συνθήκη `CCD > '1.5'`. Δεν την ικανοποιεί. Ανάλογα οι τιμές '240.45' και '249.45' για τις ιδιότητες `price` και `otherPrice` αντίστοιχα δεν

ικανοποιούν τη $price > otherPrice$. Συνεπώς η εγγραφή δεν ικανοποιεί ταυτόχρονα τις συνθήκες, οπότε και δε θα κρατηθεί. Αντίθετα, η 'Minolta', 'DiMAGE X20', '2', '256', '249.95' εισάγεται στο $vTuples$ του $resultTSR.item$. Με ανάλογο τρόπο κατασκευάζουμε τις OR συνιστώσες του vOR του $resultTSR$. Αρχικά, αν η λίστα $vPathCondition$ ήταν άδεια τότε όλες οι συνιστώσες από το vOR του tsr θα αντιγράφονταν στο vOR του $resultOR$. Κάτι τέτοιο δεν ισχύει στο συγκεκριμένο παράδειγμα, οπότε ελέγχουμε αν υπάρχει στο αρχικό σχήμα μεταβλητή μονοπατιών \$1, σύμφωνα με τη συνθήκη. Αυτό σημαίνει ότι στη μεγαλύτερη OR συνιστώσα του tsr , δηλαδή αυτή που περιέχει τα περισσότερα AND μονοπάτια στο $vAND$ της, υπάρχει αριστερότερο μονοπάτι, πράγμα που ισχύει. Στην προκειμένη περίπτωση το μοναδικό μονοπάτι ικανοποιεί τη συνθήκη $/digital/cameras$ οπότε κατασκευάζεται μία νέα OR συνιστώσα που εισάγεται στο vOR του αποτελέσματος $resultTSR$. Ακολουθεί η επίσημη διατύπωση του αλγόριθμου.

Είσοδος: TSR σχήμα tsr , λίστα συνθηκών ιδιοτήτων $vAttributeCondition$, λίστα συνθηκών μονοπατιών $vPathCondition$.

Έξοδος: TSR σχήμα $resultTSR$.

Αλγόριθμος:

Αν η $vAttributeCondition$ είναι άδεια τότε
 ο κόμβος - αντικείμενο $resultTSR.item$ είναι ο κόμβος -
 αντικείμενο $tsr.item$.

αλλιώς

Για κάθε ιδιότητα του $vAttributes$ του $tsr.item$
 Προσθήκη ιδιότητας στο $vAttributes$ του $resultTSR.item$.
 Για κάθε συνθήκη (class Condition) της $vAttributeCondition$
 Αν πρώτος όρος της συνθήκης $term1$ δεν είναι ιδιότητα του
 $vAttributes$ του $tsr.item$ τότε
 Επέστρεψε κενό σχήμα.
 Αν η συνθήκη είναι type αληθής, δηλαδή ιδιότητα
 τελεστής ιδιότητα τότε
 Αν δεύτερος όρος της συνθήκης $term2$ δεν είναι
 ιδιότητα του $vAttributes$ του $tsr.item$ τότε
 Επέστρεψε κενό σχήμα.

Για κάθε εγγραφή του $vTuples$ του $tsr.item$
 Τεμαχισμός της εγγραφής σε τιμές.
 Για κάθε συνθήκη (class Condition) της
 $vAttributeCondition$
 Αν η συνθήκη είναι type ψευδής, δηλαδή ιδιότητα
 τελεστής τιμή τότε
 Δες αν η αντίστοιχη με την ιδιότητα τιμή
 ικανοποιεί τη συνθήκη.
 Δες αν η εγγραφή ικανοποιεί την ακολουθία των
 μέχρι αυτήν συνθηκών. *

αλλιώς αν η συνθήκη είναι type αληθής, δηλαδή ιδιότητα
 τελεστής ιδιότητα τότε

Δες αν η αντίστοιχη με τις ιδιότητες τιμές ικανοποιεί τη συνθήκη.

Δες αν η εγγραφή ικανοποιεί την ακολουθία των μέχρι αυτήν συνθηκών. *

Αν η εγγραφή ικανοποιεί την ακολουθία συνθηκών ιδιοτήτων τότε Προσθήκη εγγραφής στο vTuples του resultTSR.item.

Αν η vPathCondition είναι άδεια τότε

Για κάθε OR συνιστώσα του vOR του tsr
Προσθήκη συνιστώσας στο vOR του resultTSR.

αλλιώς

Για κάθε συνθήκη (class Condition) της vPathCondition

Αν πρώτος όρος της συνθήκης term1 δεν είναι μεταβλητή μονοπατιών του tsr τότε

Επέστρεψε κενό σχήμα.

Αν η συνθήκη είναι type αληθής, δηλαδή μεταβλητή μονοπατιού τελεστής μεταβλητή μονοπατιού τότε

Αν πρώτος όρος της συνθήκης term1 δεν είναι μεταβλητή μονοπατιών του tsr τότε

Επέστρεψε κενό σχήμα.

Για κάθε OR συνιστώσα του vOR του tsr

Για κάθε AND μονοπάτι του vAND της OR συνιστώσας

Για κάθε συνθήκη (class Condition) της vPathCondition

Αν η συνθήκη είναι type ψευδής, δηλαδή μεταβλητή μονοπατιού τελεστής τιμή τότε

Δες αν το μονοπάτι ικανοποιεί τη συνθήκη.

Δες αν το μονοπάτι ικανοποιεί την ακολουθία των μέχρι αυτήν συνθηκών. *

αλλιώς αν η συνθήκη είναι μεταβλητή μονοπατιών τελεστής μεταβλητή μονοπατιών τότε

Δες αν το μονοπάτι ικανοποιεί τη συνθήκη.

Δες αν το μονοπάτι ικανοποιεί την ακολουθία των μέχρι αυτήν συνθηκών. *

Αν το μονοπάτι ικανοποιεί την ακολουθία συνθηκών μονοπατιών τότε

Προσθήκη του μονοπατιού στο vAND της συνιστώσας resultOR.

Αν η συνιστώσα resultOR δεν είναι άδεια τότε

Προσθήκη της συνιστώσας ResultOR στο vOR του resultTSR.

Δημιουργία νέας OR συνιστώσας resultOR.

Αν δεν υπάρχουν OR συνιστώσες στο vOR του resultTSR ή

αν δεν υπάρχουν εγγραφές στο vTuples του resultTSR.item τότε

Επέστρεψε κενό σχήμα.

Επέστρεψε το resultTSR.

* Η διαδικασία ελέγχου αν μία εγγραφή ή ένα μονοπάτι ικανοποιεί την ακολουθία συνθηκών πραγματοποιείται ως εξής: Η λίστα συνθηκών αποτελεί μια Boolean έκφραση. Το σύστημα ελέγχει αν η τιμή (τιμές) της εγγραφής

ή το μονοπάτι ικανοποιούν μία μία τις συνθήκες και αποτιμά με βάση την πληροφορία αυτή ολόκληρη την έκφραση της λίστας συνθηκών.

5.2.4.6 Προβολή – Project

Η πράξη της προβολής εφαρμόζεται σε ένα TSR σχήμα. Σύμφωνα με τον ορισμό της γλώσσας TreeSQuerL, χρειάζονται μία λίστα με ιδιότητες και μία δεύτερη λίστα με μεταβλητές μονοπατιών του σχήματος. Ο αλγόριθμος στηρίζεται στην επιλογή από το αρχικό σχήμα όσον αφορά τον κόμβο – αντικείμενο, των ιδιοτήτων της λίστας ιδιοτήτων και των αντίστοιχων τιμών των εγγραφών. Ενώ για το σύνολο των μονοπατιών, από κάθε OR συνιστώσα του αρχικού σχήματος επιλέγονται τα μονοπάτια που αντιστοιχούν στις μεταβλητές της λίστας μεταβλητών μονοπατιών. Για παράδειγμα, έστω ότι συντάσσεται η ερώτηση:

```
p(price, CCD) ($1) (BH.xml#DigitalCameras)
```

Αρχικά, η εφαρμογή διαχείρισης XML αρχείων ή βάσης δεδομένων προσπελαύνει το σχήμα BH.xml#DigitalCameras και κατασκευάζει ένα αντικείμενο `tsr` της κλάσης TSR. Από τη σύνταξη της ερώτησης κατασκευάζονται οι δύο λίστες – παράμετροι, `vAttributeVariable` και `vPathVariable`, με την εξής μορφή:

```
['price', 'CCD'] και [$1]
```

Αν οι ιδιότητες της `vAttributeVariable` και οι μεταβλητές μονοπατιών της `vPathVariable` δεν υπήρχαν στο σχήμα `tsr`, τότε οι λίστες παράμετροι δεν θα περιείχαν έγκυρα δεδομένων και η διαδικασία θα τελείωνε με επιστροφή κενού σχήματος. Αν η λίστα `vAttributeVariable` ήταν κενή τότε ο κόμβος – αντικείμενο του αποτελέσματος `resultTSR` θα ήταν αντιγραφή του `tsr.item`. Χρησιμοποιώντας τις ιδιότητες του `vAttributeVariable`, κατασκευάζουμε ένα διάνυσμα με τη θέση τους [3, 2] στο `vAttributes` του `tsr`, με το όνομα `attrIndexContainedIntoVAttributeVariable`, και στη συνέχεια για κάθε εγγραφή του `tsr.item` κάνουμε το εξής: φτιάχνουμε ένα πίνακα με τιμές που προκύπτουν από τεμαχισμό της εγγραφής με βάση το χαρακτήρα κόμμα (,) και κρατάμε μόνο τη δεύτερη και την τρίτη τιμή, όπως δείχνει το διάνυσμα `attrIndexContainedIntoVAttributeVariable`. Για παράδειγμα από την εγγραφή 'Nikon', 'Coolpix 2100', '2', '240.45', '249.95' κρατάμε μόνο τις '249.95' και '2' και κατασκευάζουμε τη νέα εγγραφή '249.95', '2' που εισάγουμε στον κόμβο – αντικείμενο `resultTSR.item` του αποτελέσματος. Με παρόμοια διαδικασία θα εισαχθούν και τα κατάλληλα μονοπάτια του `tsr` στο `resultTSR`. Αν η `vPathVariable` ήταν άδεια, θα εισάγονταν όλα τα μονοπάτια του αρχικού σχήματος στο αποτέλεσμα. Στην προκειμένη περίπτωση, για κάθε OR συνιστώσα του `vOR` του `tsr` κρατάμε (σύμφωνα με τα περιεχόμενα της `vPathVariable` [\$1]) το μηδενικό στοιχείο του διανύσματος `vAND` της συνιστώσας, δηλαδή το πρώτο AND μονοπάτι (διότι η αρίθμηση στο διάνυσμα ξεκινά από το 0). Έτσι, εισάγουμε το μονοπάτι `/digital/cameras` σε ένα αντικείμενο `resultOR` της κλάσης OR

που εφόσον δεν είναι άδαιο εισάγεται στο vOR διάλυμα του αποτελέσματος resultTSR. Αν υπήρχαν και άλλες συνιστώσες στο vOR του tsr η διαδικασία επιλογής του μονοπατιού θα επαναλαμβανόταν.

Ακολουθεί η επίσημη διατύπωση του αλγόριθμου.

Είσοδος: TSR σχήμα tsr, λίστα ιδιοτήτων vAttributeVariable, λίστα μεταβλητών μονοπατιών vPathVariable.

Έξοδος: TSR σχήμα resultTSR.

Αλγόριθμος:

Αν οι ιδιότητες της vAttributeVariable περιέχονται στο tsr.item ή
αν οι μεταβλητές μονοπατιών της vPathVariable περιέχονται στο tsr
τότε

Επέστρεψε κενό σχήμα.

Αν η vAttributeVariable είναι άδεια τότε

ο κόμβος - αντικείμενο resultTSR.item είναι ο κόμβος -
αντικείμενο tsr.item.

αλλιώς

Για κάθε ιδιότητα της vAttributeVariable

Προσθήκη ιδιότητας στο vAttributes του resultTSR.item.

Βρες τον αύξοντα αριθμό της στο vAttributes του tsr.item.

Για κάθε εγγραφή του vTuples του tsr.item

Τεμαχισμός της εγγραφής με βάση το χαρακτήρα ','.

Κράτα μόνο τις τιμές - τεμάχια που αντιστοιχούν στους
αύξοντες αριθμούς στο vAttributes των ιδιοτήτων της
vAttributeVariable.

Προσθήκη εγγραφής στο vTuples του resultTSR.item.

Αν η vPathVariable είναι άδεια τότε

Για κάθε OR συνιστώσα του vOR του tsr

Προσθήκη συνιστώσας στο vOR του resultTSR.

αλλιώς

Για κάθε OR συνιστώσα του vOR του tsr

Για κάθε μεταβλητή μονοπατιού της vPathVariable

Προσθήκη στη συνιστώσα resultOR του
αντίστοιχου, με τη μεταβλητή μονοπατιών,
μονοπατιού του vAND της συνιστώσας του tsr.

Αν η συνιστώσα resultOR δεν είναι άδεια τότε

Προσθήκη της συνιστώσας ResultOR στο vOR του
resultTSR.

Δημιουργία νέας OR συνιστώσας resultOR.

Επέστρεψε το resultTSR.

5.2.4.7 Καρτεσιανό γινόμενο – Cartesian Product

Η πράξη του καρτεσιανού γινομένου εκτελείται σε δύο TSR σχήματα. Για να είναι έγκυρα τα σχήματα αυτά πρέπει να περιέχουν διαφορετικές ως προς το όνομα ιδιότητες. Αυτή είναι μια

απαίτηση που δεν προέρχεται από τον ορισμό της ερώτησης της γλώσσας TreeSQuerL, αλλά από την υλοποίηση του συστήματος. Για παράδειγμα, έστω ότι συντάσσεται η ερώτηση:

```
x(BH.xml#DigitalCameras) (BH.xml#SLRcameras)
```

Αρχικά, η εφαρμογή διαχείρισης XML αρχείων ή βάσης δεδομένων προσπελαίνει τα σχήματα BH.xml#DigitalCameras, BH.xml#SLRcameras και κατασκευάζει τα tsr1, tsr2 αντίστοιχα. Ο αλγόριθμος ελέγχει αν τα σχήματα tsr1 και tsr2 έχουν ίδιες ως προς το όνομα ιδιότητες. Αν είχαν, η πράξη δε θα μπορούσε να γίνει. Στο παράδειγμα οι ιδιότητες του tsr1: [sbrand, smodel, sprice] και του tsr2: [brand, model, CCD, price] έχουν διαφορετικά ονόματα οπότε η πράξη μπορεί να γίνει. Έτσι προσθέτουμε στον κόμβο – αντικείμενο του αποτελέσματος resultTSR.item όλες τις ιδιότητες των tsr1.item και tsr2.item. Το vAttributes του resultTSR.item γίνεται:

```
[name = "brand" type = "text" ,  
name = "model" type = "text" ,  
name = "CCD" type = "float" ,  
name = "price" type = "float" ,  
name = "otherPrice" type = "float",  
name = "sbrand" type = "text" ,  
name = "smodel" type = "text" ,  
name = "sprice" type = "float"]
```

Στη συνέχεια, παίρνουμε κάθε εγγραφή του tsr1.item και την ενώνουμε με κάθε μία του tsr2.item κατασκευάζοντας καινούριες εγγραφές, συνδυασμούς των αρχικών, τις οποίες τις προσθέτουμε στο resultTSR.item. Για παράδειγμα, η εγγραφή 'Canon', 'Powershot A60', '2', '229', '230' από το πρώτο σχήμα και η 'Minolta', 'Maxxum 3 QD', '384.95' από το δεύτερο δημιουργούν την 'Canon', 'Powershot A60', '2', '229', '230', 'Minolta', 'Maxxum 3 QD', '384.95'. Για το σύνολο των μονοπατιών του αποτελέσματος ακολουθείται η εξής διαδικασία: κατασκευάζουμε καινούρια συνιστώσα, παίρνοντας όλα τα μονοπάτια κάθε OR συνιστώσας του vOR του tsr1 με όλα τα μονοπάτια του vOR του tsr2. Δηλαδή στο συγκεκριμένο παράδειγμα, κατασκευάζουμε την resultOR συνιστώσα με vAND διάνυσμα που περιέχει τα /digital/cameras και /photo/35mmSystems/SLRcameras. Στη συνέχεια προσθέτουμε τη συνιστώσα resultOR στο vOR του resultTSR.

Ακολουθεί η επίσημη διατύπωση του αλγόριθμου.

Είσοδος: Δύο TSR σχήματα tsr1 και tsr2.

Έξοδος: TSR σχήμα resultTSR.

Αλγόριθμος:

Αν οι tsr1.item και tsr2.item έχουν ίδιες ιδιότητες ιότι
Επέστρεψε κενό σχήμα.

Για κάθε ιδιότητα του vAttributes του tsr1.item

Προσθήκη ιδιότητας στο `vAttributes` του `resultTSR.item`.
Για κάθε ιδιότητα του `vAttributes` του `tsr2.item`
 Προσθήκη ιδιότητας στο `vAttributes` του `resultTSR.item`.
Για κάθε εγγραφή του `vTuples` του `tsr1.item`
Για κάθε εγγραφή του `vTuples` του `tsr2.item`
 Ένωση της εγγραφής του `tsr1` με την εγγραφή του `tsr2`.
 Προσθήκη της ένωσης ως εγγραφή στο `vTuples` του `resultTSR.item`.
Για κάθε OR συνιστώσα του `vOR` του `tsr1`
Για κάθε OR συνιστώσα του `vOR` του `tsr2`
Για κάθε AND μονοπάτι του `vAND` της OR συνιστώσας του `tsr1`
 Προσθήκη του μονοπατιού στο `vAND` της συνιστώσας `resultOR`.
Για κάθε AND μονοπάτι του `vAND` της OR συνιστώσας του `tsr2`
 Προσθήκη του μονοπατιού στο `vAND` της συνιστώσας `resultOR`.
 Προσθήκη της συνιστώσας `resultOR` στο `vOR` του `resultTSR`.
 Δημιουργία νέας OR συνιστώσας `resultOR`.
Επέστρεψε το `resultTSR`.

5.2.4.8 Ένωση – Union

Η πράξη της ένωσης εκτελείται σε δύο TSR σχήματα. Σύμφωνα με τον ορισμό του τελεστή στη γλώσσα TreeQuerL τα σχήματα που εισάγονται πρέπει να έχουν ίδιες κατά όνομα και τύπο ιδιότητες. Έτσι η διαδικασία ξεκινά με έλεγχο εγκυρότητας των εισόδων ως προς την πράξη της ένωσης. Εφόσον τα TSR σχήματα είναι έγκυρα, ο αλγόριθμος κατασκευάζει τον κόμβο αντικείμενο του αποτελέσματος και το σύνολο των μονοπατιών που οδηγούν σ' αυτό. Για παράδειγμα, έστω ότι συντάσσεται η ερώτηση:

```
u (BH.xml#SLRcameras) (RitzCameras.xml#SLRcameras)
```

Αρχικά, η εφαρμογή διαχείρισης XML αρχείων ή βάσης δεδομένων προσπελαύνει τα σχήματα `BH.xml#DigitalCameras`, `RitzCameras.xml#SLRcameras` και κατασκευάζει τα `tsr1`, `tsr2` αντίστοιχα. Ο αλγόριθμος ελέγχει αν τα σχήματα `tsr1` και `tsr2` έχουν ίδιες ιδιότητες τόσο ως προς το όνομα όσο κι ως προς τον τύπο. Στο παράδειγμα, κάτι τέτοιο ισχύει και έτσι μπορεί να γίνει η πράξη, διαφορετικά η διαδικασία θα τελείωνε με επιστροφή κενού σχήματος. Στη συνέχεια, κατασκευάζεται ο κόμβος – αντικείμενο του αποτελέσματος `resultTSR.item`. Προστίθενται στο `vAttributes` του `resultTSR` οι ιδιότητες του `tsr1` και έτσι το αποτέλεσμα έχει τις εξής ιδιότητες:

```
[name = "sbrand" type = "text" ,
 name = "smodel" type = "text" ,
 name = "sprice" type = "float"]
```

Για εισαγωγή των εγγραφών στον κόμβο – αντικείμενο του αποτελέσματος, σύμφωνα με τον ορισμό της πράξης, εισάγονται στο `vTuples` του `resultTSR.item` όλες οι εγγραφές από τα αντίστοιχα `vTuples` διανύσματα των `tsr1.item` και `tsr2.item`. Έπειτα, κατασκευάζεται το `vOR` διάνυσμα του `resultTSR`, που περιέχει τις OR συνιστώσες του πρώτου `tsr1` και του

δεύτερου `tsr2` σχήματος. Έτσι το τελικό σχήμα έχει σύνολο OR συνιστώσων `vOR` με τα εξής περιεχόμενα:

```
or1 = [/photo/35mmSystems/SLRcameras]
or2 = [/SLRcameras]
[or1, or2]
```

δηλαδή:

```
<or>
  <and>/photo/35mmSystems/SLRcameras</and>
</or>
<or>
  <and>/SLRcameras</and>
</or>
```

Ακολουθεί η επίσημη διατύπωση του αλγόριθμου.

Είσοδος: Δύο TSR σχήματα `tsr1` και `tsr2`.

Έξοδος: TSR σχήμα `resultTSR`.

Αλγόριθμος:

Αν οι `tsr1.item` και `tsr2.item` δεν έχουν ίδιες ιδιότητες ιότι
Επέστρεψε κενό σχήμα.

Για κάθε ιδιότητα του `vAttributes` του `tsr1.item`
Προσθήκη ιδιότητας στο `vAttributes` του `resultTSR.item`.

Για κάθε εγγραφή του `vTuples` του `tsr1.item`
Προσθήκη εγγραφής στο `vTuples` του `resultTSR.item`.

Για κάθε εγγραφή του `vTuples` του `tsr2.item`
Προσθήκη εγγραφής στο `vTuples` του `resultTSR.item`.

Για κάθε OR συνιστώσα του `vOR` του `tsr1`
Προσθήκη της συνιστώσας στο `vOR` του `resultTSR`.

Για κάθε OR συνιστώσα του `vOR` του `tsr2`
Προσθήκη της συνιστώσας στο `vOR` του `resultTSR`.

Επέστρεψε το `resultTSR`.

5.2.4.9 Τομή – Intersection

Η πράξη της τομής εκτελείται σε δύο TSR σχήματα. Σύμφωνα με τον ορισμό του τελεστή στη γλώσσα `TreeSQuerL`, τα σχήματα που εισάγονται πρέπει να έχουν ίδιες κατά όνομα και τύπο ιδιότητες. Έτσι η διαδικασία ξεκινά με έλεγχο εγκυρότητας των εισόδων ως προς την πράξη της τομής. Εφόσον τα TSR σχήματα είναι έγκυρα, ο αλγόριθμος κατασκευάζει τον κόμβο – αντικείμενο του αποτελέσματος και το σύνολο των μονοπατιών που οδηγούν σ' αυτό. Για παράδειγμα, έστω ότι συντάσσεται η ερώτηση:

```
i (BH.xml#SLRcameras) (RitzCameras.xml#SLRcameras)
```

Η διαδικασία που ακολουθείται είναι παρόμοια με αυτήν στην πράξη της ένωσης. Με τον ίδιο τρόπο εισάγονται οι κοινές ιδιότητες των σχημάτων. Ενώ, για την εισαγωγή των εγγραφών γίνεται αναζήτηση κάθε εγγραφής του `vTuples` του `tsr1.item` στο `vTuples` του `tsr2.item`, για να επιλεγούν αυτές που υπάρχουν και στα δύο. Για το σύνολο των

μονοπατιών η διαδικασία είναι ακριβώς ίδια με αυτήν στην πράξη της ένωσης. Ακολουθεί η επίσημη διατύπωση του αλγόριθμου.

Είσοδος: Δύο TSR σχήματα `tsr1` και `tsr2`.

Έξοδος: TSR σχήμα `resultTSR`.

Αλγόριθμος:

Αν οι `tsr1.item` και `tsr2.item` δεν έχουν ίδιες ιδιότητες τότε
 Επέστρεψε κενό σχήμα.
Για κάθε ιδιότητα του `vAttributes` του `tsr1.item`
 Προσθήκη ιδιότητας στο `vAttributes` του `resultTSR.item`.
Για κάθε εγγραφή του `vTuples` του `tsr1.item`
 Αν η εγγραφή υπάρχει στο `vTuples` του `tsr2.item` τότε
 προσθήκη εγγραφής στο `vTuples` του `resultTSR.item`.
Για κάθε OR συνιστώσα του `vOR` του `tsr1`
 Προσθήκη της συνιστώσας στο `vOR` του `resultTSR`.
Για κάθε OR συνιστώσα του `vOR` του `tsr2`
 Προσθήκη της συνιστώσας στο `vOR` του `resultTSR`.
Επέστρεψε το `resultTSR`.

5.2.4.10 Διαφορά – Difference

Η πράξη της τομής εκτελείται σε δύο TSR σχήματα. Σύμφωνα με τον ορισμό του τελεστή στη γλώσσα `TreeQuerL`, τα σχήματα που εισάγονται πρέπει να έχουν ίδιες κατά όνομα και τύπο ιδιότητες. Έτσι η διαδικασία ξεκινά με έλεγχο εγκυρότητας των εισόδων ως προς την πράξη της διαφοράς. Εφόσον τα TSR σχήματα είναι έγκυρα, ο αλγόριθμος κατασκευάζει τον κόμβο – αντικείμενο του αποτελέσματος και το σύνολο των μονοπατιών που οδηγούν σ’ αυτό. Για παράδειγμα, έστω ότι συντάσσεται η ερώτηση:

`d(BH.xml#SLRcameras) (RitzCameras.xml#SLRcameras)`

Η διαδικασία που ακολουθείται είναι παρόμοια με αυτήν στην πράξη της ένωσης. Με τον ίδιο τρόπο εισάγονται οι κοινές ιδιότητες των σχημάτων στον κόμβο – αντικείμενο του αποτελέσματος. Ενώ, για την εισαγωγή των εγγραφών γίνεται αναζήτηση κάθε εγγραφής του `vTuples` του `tsr1.item` στο `vTuples` του `tsr2.item`, για να επιλεγούν αυτές που υπάρχουν στο πρώτο και δεν υπάρχουν στο δεύτερο. Για το σύνολο των μονοπατιών η διαδικασία είναι ακριβώς ίδια με αυτήν στην πράξη της ένωσης. Ακολουθεί η επίσημη διατύπωση του αλγόριθμου.

Είσοδος: Δύο TSR σχήματα `tsr1` και `tsr2`.

Έξοδος: TSR σχήμα `resultTSR`.

Αλγόριθμος:

Αν οι `tsr1.item` και `tsr2.item` δεν έχουν ίδιες ιδιότητες τότε
 Επέστρεψε κενό σχήμα.
Για κάθε ιδιότητα του `vAttributes` του `tsr1.item`
 Προσθήκη ιδιότητας στο `vAttributes` του `resultTSR.item`.
Για κάθε εγγραφή του `vTuples` του `tsr1.item`
 Αν η εγγραφή δεν υπάρχει στο `vTuples` του `tsr2.item` τότε

προσθήκη εγγραφής στο `vTuples` του `resultTSR.item`.
Για κάθε OR συνιστώσα του `vOR` του `tsr1`
 Προσθήκη της συνιστώσας στο `vOR` του `resultTSR`.
Για κάθε OR συνιστώσα του `vOR` του `tsr2`
 Προσθήκη της συνιστώσας στο `vOR` του `resultTSR`.
Επέστρεψε το `resultTSR`.

5.2.4.11 Αυστηρό υποσύνολο

Ο αλγόριθμος σύγκρισης μονοπατιών αυστηρού υποσυνόλου στηρίζεται στην αναζήτηση των κόμβων του πρώτου μονοπατιού στο δεύτερο. Στόχος είναι να εντοπιστούν όλοι οι κόμβοι του πρώτου μονοπατιού στο δεύτερο, αλλά με την ίδια σειρά προηγούμενου – επόμενου (*predecessor – successor*) και χωρίς να παρεμβάλλονται άλλοι ενδιάμεσα. Για παράδειγμα, έστω ότι θέλουμε να συγκρίνουμε τα μονοπάτια `/cameras/digital` και `/photo/cameras/digital/canon`. Εισάγουμε τους κόμβους του δεύτερου μονοπατιού σε ένα πίνακα κατακερματισμού. Έτσι, ο κόμβος `photo` μπαίνει στη θέση 1, ο `cameras` στη 2 κ.τ.λ. Στη συνέχεια ελέγχουμε αν ο κόμβος `cameras` του πρώτου μονοπατιού υπάρχει στον πίνακα κατακερματισμού. Αν δεν υπάρχει η διαδικασία τελειώνει. Παρόμοια ελέγχουμε και τον κόμβο `digital` του πρώτου μονοπατιού. Ο κόμβος `digital` βρίσκεται στη θέση 2. Η διαφορά θέσης των `digital` και `cameras` στον πίνακα κατακερματισμού είναι 1, οπότε οι κόμβοι είναι συνεχόμενοι τόσο στο πρώτο όσο και στο δεύτερο μονοπάτι, άρα το `/cameras/digital` είναι αυστηρό υποσύνολο `/photo/cameras/digital/canon`. Ακολουθεί η επίσημη διατύπωση του αλγόριθμου:

Είσοδος: Δύο μονοπάτια `path1` και `path2`.

Έξοδος: Αληθής αν ισχύει $path1 \subseteq path2$, και ψευδής αν δεν ισχύει.

Αλγόριθμος:

Για κάθε κόμβο του `path2`
 προσθήκη στον πίνακα κατακερματισμού `ht`
Για κάθε κόμβο του `path1`
αν ο κόμβος `vi` δεν υπάρχει στον `ht` τότε
επέστρεψε ψευδής.
αλλιώς
αν η διαφορά της θέσης του `vi` στον `ht` με τη θέση του προηγούμενου `vi-1` είναι διάφορη του 1 τότε
επέστρεψε ψευδής.
Επέστρεψε αληθής.

5.2.4.12 Χαλαρό υποσύνολο

Ο αλγόριθμος σύγκρισης μονοπατιών χαλαρού υποσυνόλου στηρίζεται στην αναζήτηση των κόμβων του πρώτου μονοπατιού στο δεύτερο. Στόχος είναι να εντοπιστούν όλοι οι κόμβοι του πρώτου μονοπατιού στο δεύτερο, αλλά με την ίδια σειρά προηγούμενου – επόμενου (*predecessor – successor*), ανεξάρτητα αν παρεμβάλλονται άλλοι ανάμεσα τους. Για

παράδειγμα, έστω ότι θέλουμε να συγκρίνουμε τα μονοπάτια `/cameras/canon` και `cameras/photo/digital/canon`. Εισάγουμε τους κόμβους του δεύτερου μονοπατιού σε ένα πίνακα κατακερματισμού. Έτσι, ο κόμβος `cameras` μπαίνει στη θέση 1, ο `photo` στη 2 κ.τ.λ. Στη συνέχεια ελέγχουμε αν ο κόμβος `cameras` του πρώτου μονοπατιού υπάρχει στον πίνακα κατακερματισμού. Αν δεν υπάρχει η διαδικασία θα τελειώνει. Παρόμοια ελέγχουμε και τον κόμβο `canon` του πρώτου μονοπατιού. Ο κόμβος `canon` βρίσκεται στη θέση 4. Η διαφορά θέσης των `canon` και `cameras` στον πίνακα κατακερματισμού είναι μεγαλύτερη του 0, οπότε οι κόμβοι έχουν την ίδια σειρά προηγούμενου – επόμενου τόσο στο πρώτο όσο και στο δεύτερο μονοπάτι, άρα το `/cameras/canon` είναι χαλαρό υποσύνολο του `cameras/photo/digital/canon`. Ακολουθεί η επίσημη διατύπωση του αλγόριθμου:

Είσοδος: Δύο μονοπάτια `path1` και `path2`.

Έξοδος: Αληθής αν ισχύει $path1 \subset path2$, και ψευδής αν δεν ισχύει.

Αλγόριθμος:

Για κάθε κόμβο του `path2`
 προσθήκη στον πίνακα κατακερματισμού `ht`
Για κάθε κόμβο του `path1`
 αν ο κόμβος `vi` δεν υπάρχει στον `ht` τότε
 επέστρεψε ψευδής.
 αλλιώς
 αν η διαφορά της θέσης του `vi` στον `ht` με τη θέση του
 προηγούμενου `vi-1` είναι μικρότερη του 0 τότε
 επέστρεψε ψευδής.
Επέστρεψε αληθής.

6

Έλεγχος

Η υλοποίηση του συστήματος ολοκληρώνεται με το λεπτομερή έλεγχο.

6.1 Μεθοδολογία Ελέγχου

Ο έλεγχος του συστήματος πραγματοποιήθηκε με χρήση σεναρίων λειτουργίας. Με βάση τα χαρακτηριστικά του συστήματος, συντάσσονται σενάρια ελέγχου που χρησιμοποιούν όλες τις λειτουργίες που προσφέρει το σύστημα. Στο τέλος, κρίνεται η ισχύς των αποτελεσμάτων, η συμπεριφορά και η απόδοση του. Ωστόσο, δεν είναι εύκολο να βρεθούν όλα τα δυνατά σενάρια που να καλύπτουν όλες τις περιπτώσεις ελέγχου του συστήματος. Παρόλα αυτά, καλύπτουν όλο το φάσμα των δυνατών τρόπων χρήσης του συστήματος.

6.2 Αναλυτική παρουσίαση έλεγχου

Στην ενότητα αυτή περιγράφεται ένα σενάριο χρήσης του συστήματος με σύνταξη μιας ερώτησης επιλογής της γλώσσας TreeSQuerL στο σχήμα καταλόγου BH (www.bhphotovideo.com) που περιέχει φωτογραφικό εξοπλισμό. Θεωρούμε το TSR σχήμα SLRcameras που περιέχει πληροφορίες για SLR φωτογραφικές μηχανές. Το σχήμα είναι αποθηκευμένο στο BH.xml αρχείο με τη παρακάτω μορφή:

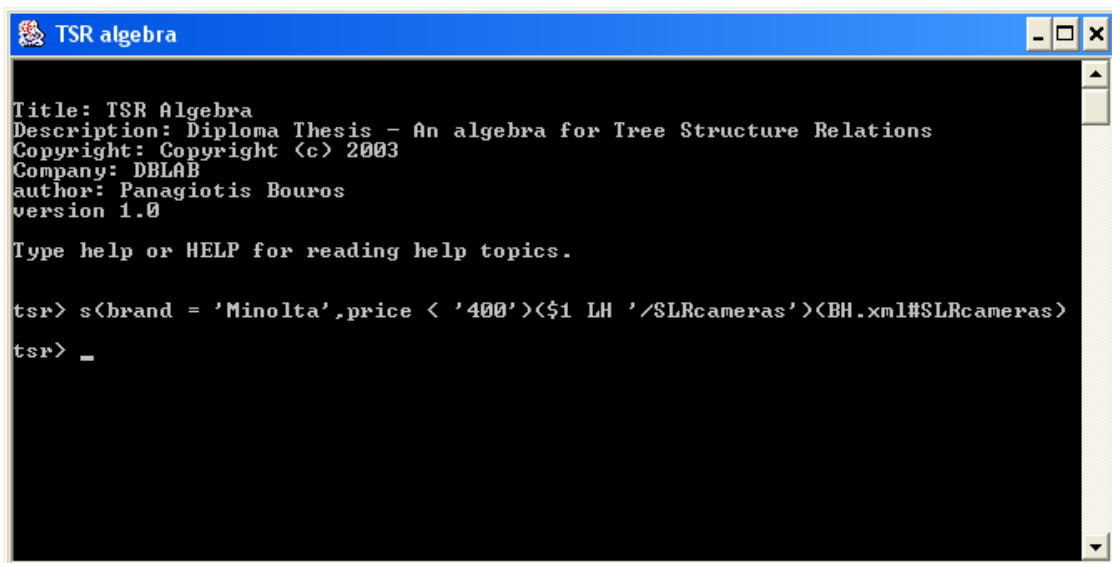
```
<tsr name="SLRcameras">  
  <or>  
    <and>/photo/35mmSystems/SLRcameras</and>
```

```

</or>
<item>
  <attribute name="brand" type="text" />
  <attribute name="model" type="text" />
  <attribute name="price" type="float" />
  <tuple>'Canon', 'EOS 3', '849.95'</tuple>
  <tuple>'Canon', 'EOS Rebel Ti', '254.95'</tuple>
  <tuple>'Minolta', 'Maxxum 3 QD', '384.95'</tuple>
  <tuple>'Minolta', 'Maxxum 4 QD', '164.95'</tuple>
  <tuple>'Minolta', 'Maxxum 9 QD', '1099.95'</tuple>
  <tuple>'Minolta', 'X 370s', '149.95'</tuple>
  <tuple>'Nikon', 'N65', '199.95'</tuple>
  <tuple>'Nikon', 'F80', '349.95'</tuple>
  <tuple>'Pentax', 'ZX M', '149.95'</tuple>
  <tuple>'Sigma', 'SA 7', '249.95'</tuple>
</item>
</tsr>

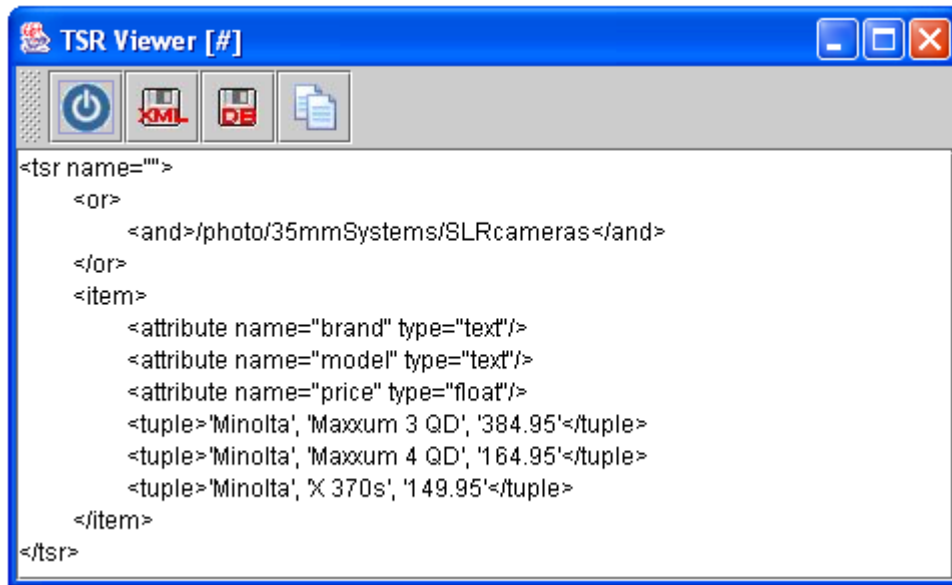
```

Χρησιμοποιώντας την εφαρμογή του μεταγλωττιστή στο Σχ. 6.1, συντάσσουμε την ερώτηση επιλογής μηχανών Minolta με τιμή μικρότερη των 400 Ευρώ στις οποίες οδηγεί αριστερότερο μονοπάτι, χαλαρό υπερσύνολο του /SLRcameras:



- Σχ. 6.1 -

Το αποτέλεσμα παρουσιάζεται με το γραφικό περιβάλλον παρουσίασης σχημάτων TSR στο Σχ. 6.2. Παρατηρούμε ότι το σχήμα – αποτέλεσμα της πράξης έχει τη μορφή με την οποία αποθηκεύεται σε XML αρχείο. Το μονοπάτι του σχήματος είναι /photo/35mmSystems/SLRcameras που είναι υπερσύνολο του /SLRcameras. Ο κόμβος – αντικείμενο περιέχει όλες τις ιδιότητες του αρχικού σχήματος BH.xml#SLRcameras και τις εγγραφές που ικανοποιούν ταυτόχρονα τις συνθήκες brand = 'Minolta' και price < '400'.



- Σχ. 6.2 -

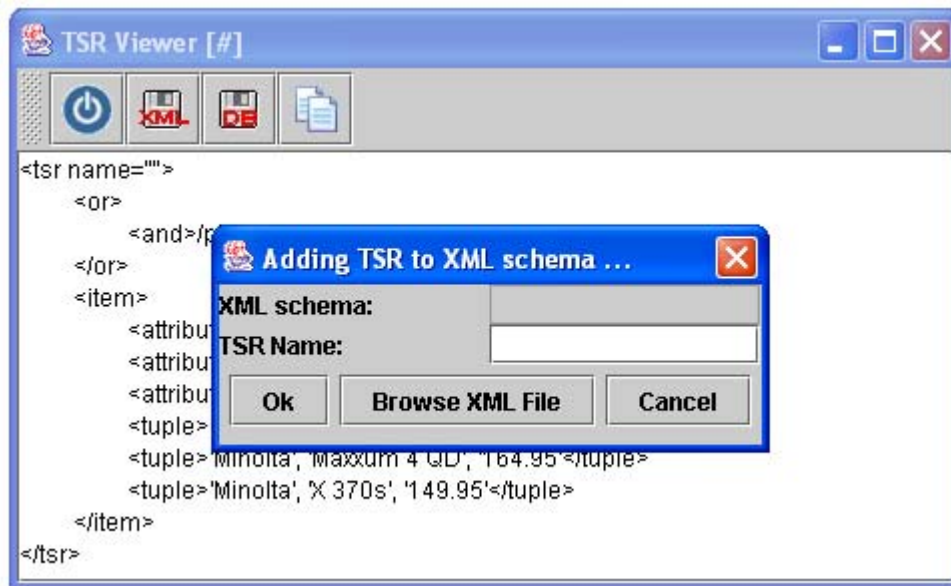
Η εφαρμογή προσφέρει τη δυνατότητα αποθήκευσης του αποτελέσματος σε XML αρχείο και στη βάση δεδομένων του συστήματος. Επιλέγοντας την αποθήκευση σε XML αρχείο, η εφαρμογή ζητάει το όνομα του XML αρχείου όπου θα αποθηκευτεί το σχήμα και το όνομά του. Στο Σχ. 6.3 φαίνεται το παράθυρο εισαγωγής των στοιχείων του TSR σχήματος που θα αποθηκευτεί. Η εισαγωγή του ονόματος του XML αρχείου γίνεται μέσω ενός παραθύρου επιλογής αρχείου που φαίνεται στο Σχ. 6.4. Μ' αυτόν τον τρόπο ο χρήστης μπορεί να αποθηκεύσει το σχήμα είτε σε υπάρχον XML αρχείο, προσθέτοντάς το στο τέλος, είτε σε ένα καινούριο που θα δημιουργηθεί απ' το σύστημα. Έστω ότι επιλέγουμε να το αποθηκεύσουμε σ' ένα νέο αρχείο με όνομα `photorama.xml` με όνομα σχήματος SLR. Ανοίγοντας το αρχείο αυτό διαπιστώνουμε ότι περιέχει το σχήμα που βλέπουμε στο Σχ. 6.2 με όνομα SLR. Συγκεκριμένα:

```

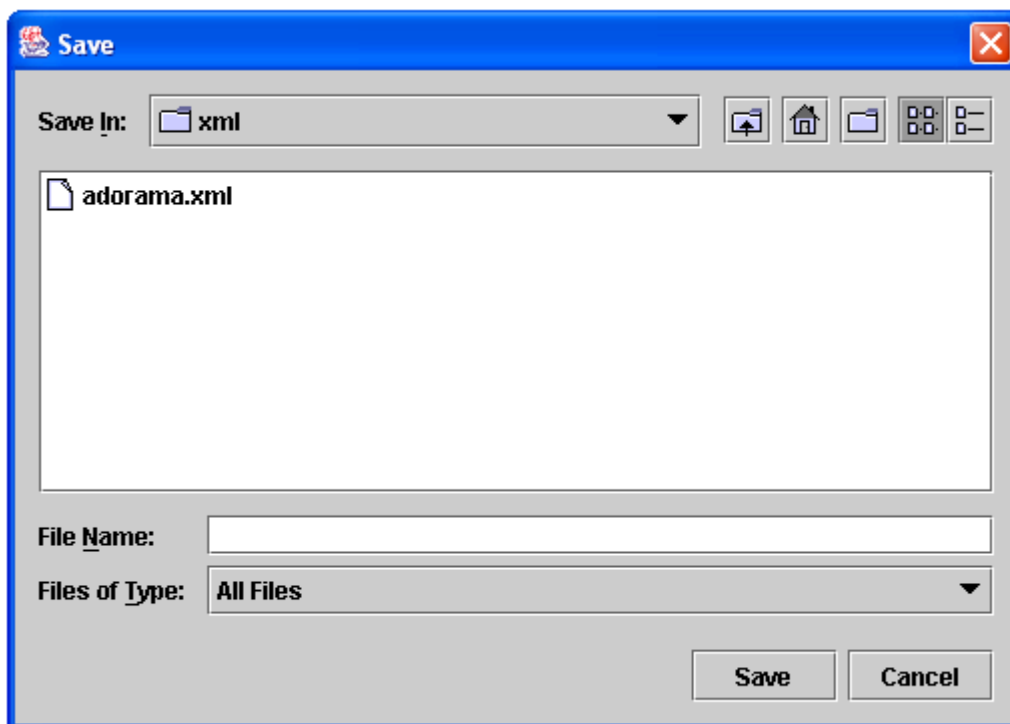
<?xml version="1.0" ?>
<root>
<tsr name="SLR">
  <or>
    <and>/photo/35mmSystems/SLRcameras</and>
  </or>
  <item>
    <attribute name="brand" type="text"/>
    <attribute name="model" type="text"/>
    <attribute name="price" type="float"/>
    <tuple>'Minolta', 'Maxxum 3 QD', '384.95'</tuple>
    <tuple>'Minolta', 'Maxxum 4 QD', '164.95'</tuple>
    <tuple>'Minolta', 'X 370s', '149.95'</tuple>
  </item>
</tsr>

```

</root>

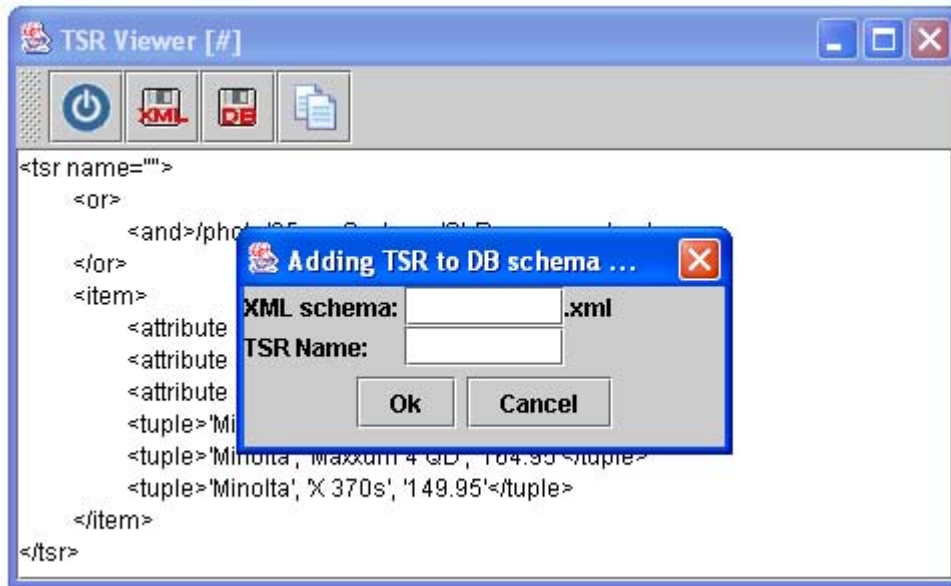


- Σχ. 6.3 -



- Σχ. 6.4 -

Σύμφωνα με τις επιλογές της εφαρμογής παρουσίασης σχημάτων TSR μπορούμε να αποθηκεύσουμε το αποτέλεσμα της πράξης στη βάση δεδομένων του συστήματος. Και σ' αυτή την περίπτωση, όπως φαίνεται στο Σχ 6.5, ζητείται το όνομα του TSR και το όνομα του XML σχήματος (αρχείου) απ' όπου προέρχεται.



- Σχ. 6.5 -

Εναλλακτικά αν θέλαμε να φορτώσουμε το σχήμα στη βάση δεδομένων μπορούμε να χρησιμοποιήσουμε την εντολή `s2db` του μεταγλωττιστή, που μεταφέρει το σχήμα από XML αρχείο στη βάση δεδομένων. Η εντολή θα είχε τη μορφή:

```
s2db (photorama.xml#SLR)
```

Επίσης ένας άλλος τρόπος να αποθηκευτεί το αποτέλεσμα μιας πράξης κατευθείαν σε XML αρχείο χωρίς τη χρήση της εφαρμογής απεικόνισης TSR σχημάτων, είναι με χρήση της επιλογής `AS` στο τέλος της πράξης. Με σύνταξη της εντολής:

```
s (brand = 'Minolta' , price < '400') ($1 LH '/SLRcameras')
  (BH.xml#SLRcameras) as (photoram.xml#SLR)
```

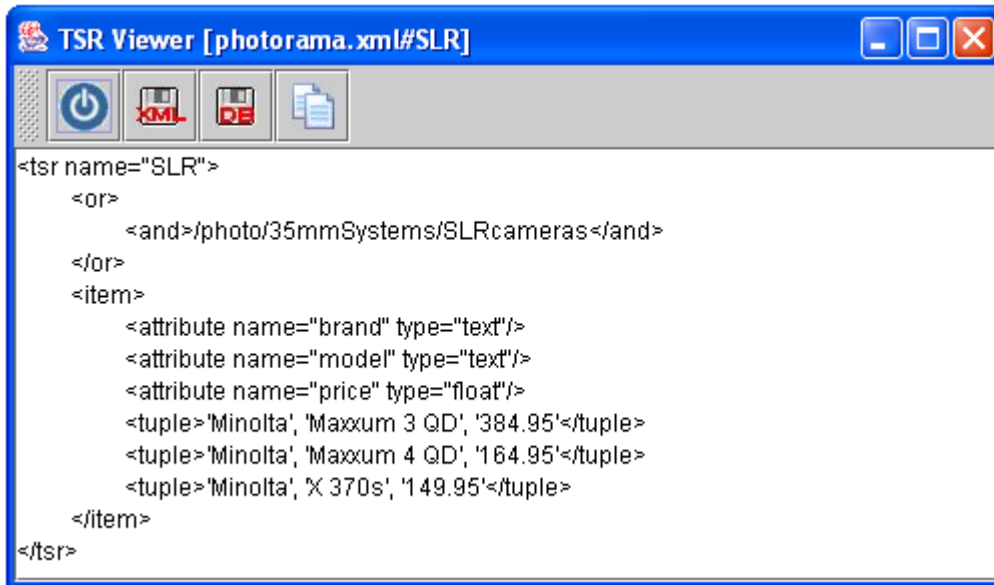
το αποτέλεσμα της ερώτησης της επιλογής αποθηκεύεται στο αρχείο `photorama.xml` με όνομα `SLR`. Σ' αυτήν περίπτωση το γραφικό περιβάλλον παρουσίασης αποτελεσμάτων απεικονίζει ονομασμένο το σχήμα αποτέλεσμα, όπως φαίνεται στο Σχ. 6.6.

Τέλος, η εφαρμογή του μεταγλωττιστή προσφέρει και τη δυνατότητα εκτέλεσης αρχείου ερωτήσεων. Έτσι η ερώτηση που θέλουμε να συντάξουμε θα μπορούσε να βρίσκεται στο αρχείο `photorama.qry` ως εξής:

```
--Query
s (brand = 'Minolta' , price < '400') ($1 LH '/SLRcameras')
  (BH.xml#SLRcameras)
```

και στη γραμμή εντολών του μεταγλωττιστή συντάσσεται η εντολή:

```
load (photorama.qry)
```



The image shows a screenshot of a software application window titled "TSR Viewer [photorama.xml#SLR]". The window has a blue title bar and standard Windows window controls (minimize, maximize, close). Below the title bar is a toolbar with four icons: a power button, an XML file icon, a save icon, and a print icon. The main content area of the window displays the following XML code:

```
<tsr name="SLR">
  <or>
    <and>/photo/35mmSystems/SLRcameras</and>
  </or>
  <item>
    <attribute name="brand" type="text"/>
    <attribute name="model" type="text"/>
    <attribute name="price" type="float"/>
    <tuple>'Minolta', 'Maxxum 3 QD', '384.95'</tuple>
    <tuple>'Minolta', 'Maxxum 4 QD', '164.95'</tuple>
    <tuple>'Minolta', 'X 370s', '149.95'</tuple>
  </item>
</tsr>
```

- Σχ. 6.6 -

7

Επίλογος

Με το κεφάλαιο αυτό ολοκληρώνεται η παρούσα διπλωματική. Παρουσιάζονται τα βασικά συμπεράσματά της και αναφέρονται μελλοντικές της επεκτάσεις.

7.1 Σύνοψη και συμπεράσματα

Η εξάπλωση και η καθιέρωση του XML προτύπου ως standard στη επικοινωνία και την ανταλλαγή πληροφοριών μεταξύ ετερογενών κόμβων πληροφοριών του διαδικτύου καθιστά σημαντικότερη τη διαχείριση του μεγάλου όγκου πληροφορίας που οργανώνεται ιεραρχικά σ' αυτούς του κόμβους. Σκοπός της διπλωματικής εργασίας αυτής ήταν ο ορισμός και η υλοποίηση μιας γλώσσας ερωτήσεων σε ιεραρχίες στις οποίες οργανώνεται το πλήθος αυτό των πληροφοριών.

Αρχικά παρουσιάστηκαν ερευνητικές εργασίες και τεχνολογίες συναφείς με το θέμα της διπλωματικής. Συγκεκριμένα, συζητήθηκαν οι γλώσσες XPath και XQuery και η άλγεβρα TAX και τονίστηκε η απαίτηση επέκτασης της λειτουργικότητας που προσφέρεται απ' αυτές.

Στη συνέχεια, παρουσιάστηκε η θεωρητική μελέτη του ζητήματος διαχείρισης πληροφορίας στους δικτυακούς καταλόγους. Ορίστηκαν οι βασικές δομές δεδομένων της γλώσσας, καθώς και το Rp-Ri μοντέλο αναπαράστασης των δομών αυτών. Παράλληλα, ορίστηκε το σύνολο των πράξεων της άλγεβρας της γλώσσας που εφαρμόζονται στις παραπάνω δομές. Οι βασικοί τελεστές που παρουσιάστηκαν είναι η επιλογή, η προβολή, το

καρτεσιανό γινόμενο, η ένωση, η τομή και η διαφορά, ενώ ορίστηκε και ο σύνθετος τελεστής της σύνδεσης.

Μετά τον ορισμό της γλώσσας TreeSQuerL παρουσιάστηκε η ανάλυση, η σχεδίαση και τέλος η υλοποίηση σε γλώσσα προγραμματισμού Java του συστήματος. Έγινε ένας διαχωρισμός του συστήματος σε επιμέρους υποσυστήματα και έπειτα συζητήθηκαν οι εφαρμογές που τα αποτελούν. Επίσης, αναλύθηκαν λεπτομερώς οι αλγόριθμοι του συστήματος που υλοποιούν τους βασικούς τελεστές της γλώσσας, τη σύγκριση μονοπατιών και τη διαχείριση των αποθηκευτικών μέσων. Τέλος η ολοκλήρωση του συστήματος πέρασε από το σημαντικό στάδιο ελέγχου λειτουργίας με σενάρια χρήσης του. Ένα απ' αυτά τα σενάρια λειτουργίας παρουσιάστηκε αναλυτικά.

Εν κατακλείδι, η διπλωματική εργασία αυτή κατάφερε να συμπεριλάβει και να επεκτείνει τη λειτουργικότητα επιλογής και επεξεργασίας δεδομένων οργανωμένων σε XML δενδρικές μορφές. Με δεδομένο ότι στο διαδίκτυο υπάρχουν πολλοί τρόποι για να προσπελάσει κανείς ένα αντικείμενο, το σύνολο των τελεστών της η γλώσσα TreeSQuerL προσφέρει τη δυνατότητα να επιλεγθούν και να επεξεργασθούν τα δεδομένα του αντικειμένου και οι διαφορετικοί τρόποι προσπέλασής (μονοπάτια) του, καθώς και να κατασκευαστεί μία καινούρια ιεραρχική οργάνωσή του.

7.2 Μελλοντικές επεκτάσεις

Η προσπάθεια ορισμού και υλοποίησης της γλώσσας TreeSQuerL που παρουσιάστηκε σ' αυτήν τη διπλωματική εργασία αποτελεί μια καλή βάση για να συνεχιστεί και να επεκταθεί η μελέτη της διαχείρισης του μεγάλου όγκου πληροφορίας που αποθηκεύεται και μεταφέρεται στο διαδίκτυο με τη χρήση του XML προτύπου. Υπό το πρίσμα αυτού του πλαισίου υπάρχουν μια σειρά από ιδέες και προτάσεις για την επέκταση και αναβάθμιση της μελέτης:

- Προσθήκη τελεστών εισαγωγής και διαγραφής δεδομένων στα TSR σχήματα, όπως για παράδειγμα εισαγωγή OR συνιστωσών ή AND μονοπατιών και εισαγωγή ιδιοτήτων ή εγγραφών.
- Ορισμός και ενσωμάτωση επιπλέον τελεστών διαχείρισης των δενδρικών ιεραρχικών δομών των δικτυακών καταλόγων στη γλώσσα TreeSQuerL.
- Διαφοροποίηση της σύνταξης και επέκταση της υλοποίησης των πράξεων ώστε να καταργηθεί ο περιορισμός που εισάγεται στο καρτεσιανό γινόμενο για σχήματα διαφορετικών ως προς όνομα ιδιοτήτων.
- Υλοποίηση της πράξης της σύνδεσης στο σύστημα.
- Δημιουργία συστήματος αυτόματης δημιουργίας TSR σχημάτων από σχήματα δικτυακών καταλόγων.

8

Βιβλιογραφία

- [XPATH] XML Path Language (XPath) Version 1.0, W3C Recommendation 1999, see <http://www.w3.org/TR/xpath>
- [XSLT] XSL Transformation (XSLT) Version 1.0, W3C Recommendation 1999, see <http://www.w3.org/TR/xslt>
- [XPTR] XML Pointers (XPointers) Version 1.0, W3C Recommendation 2001, see <http://www.w3.org/TR/xptr>
- [XQUERY] XQuery 1.0: An XML Query Language, W3C Working Draft 2002, see <http://www.w3.org/TR/xquery>
- [JLS+01] H. V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava, Keith Thomson, TAX: A Tree Algebra for XML, in Proceedings of DBPL'01, 2001
- [AQW+97] Serge Abiteboul, Dallon Quass, Jason McHugh, Jennifer Widom, Janet L. Wiener, The Lorel query language for semistructured data, International Journal on Digital Libraries 1997
- [Cha02] D. Chamberlin. XQuery: An XML query language. IBM Systems Journal, VOL 41, NO 4, 2002
- [ABS02] Serge Abiteboul, Peter Buneman, Dan Suciu, Data on the Web: from Relations to Semistructured Data and XML, Morgan Kaufmann, 2002

- [HM02] Elliotte Rusty Harold, W. Scott Means, XML in a Nutshell, 2nd Edition, O' Reilly, 2002
- [McL01] Brett McLaughlin, Java & XML: Solutions to Real-World Problems, 2nd Edition, O' Reilly, 2001
- [Man00] Ιωάννης Μανωλόπουλος, Δομές Δεδομένων, Art Of Text, 2000
- [ENA01] R. Elmasri, S. B. Navathe, Θεμελιώδεις Αρχές Βάσεων Δεδομένων 3^η Έκδοση, Δίαυλος, 2001
- [Zac98] Ευστάθιος Ζάχος, Αλγόριθμοι και Πολυπλοκότητα, Σημειώσεις μαθήματος, 1998
- [PS02] Νικόλαος Σ. Παπασπύρου, Εμμανουήλ Σ. Σκορδαλάκης, Μεταγλωττιστές, Συμμετρία, 2002
- [JAVACC] Java Compiler Compiler (JavaCC): The Java Parser Generator, see <https://javacc.dev.java.net>
- [JWORLD] JavaWorld, Build your own languages with JavaCC, December 2000, see http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-cooltools_p.html
- [Som03] Ian Sommerville, Software Engineering 6th Edition, Addison Wesley, 2003