ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Κατασκευή ενός Πολυμορφικού Συστήματος Διαχείρισης Βάσεων Δεδομένων: Σχεδιασμός και Υλοποίηση Υποσυστήματος μετατροπής ερωτημάτων σε PL\SQL και εκτέλεσής τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΦΙΛΙΠΠΟΣ-ΔΗΜ. ΚΑΛΑΜΙΔΑΣ

Επιβλέπων : Τίμος Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2010

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Κατασκευή ενός Πολυμορφικού Συστήματος Διαχείρισης Βάσεων Δεδομένων: Σχεδιασμός και Υλοποίηση Υποσυστήματος μετατροπής ερωτημάτων σε PL\SQL και εκτέλεσής τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΦΙΛΙΠΠΟΥ-ΔΗΜ. ΚΑΛΑΜΙΔΑ

Επιβλέπων : Τίμος Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22η Μαρτίου 2010.

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

Αθήνα, Μάρτιος 2010

.....

ΦΙΛΙΠΠΟΣ-ΔΗΜ. ΚΑΛΑΜΙΔΑΣ

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

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Συστημάτων Βάσεων και Γνώσεων Δεδομένων (ΕΒΓΔ) του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Υπεύθυνος καθηγητής είναι ο κ. Τίμος Σελλής τον οποίο ευχαριστώ για την εμπιστοσύνη και το ενδιαφέρον που μου έδειξε καθ' όλη τη διάρκεια εκπόνησης της παρούσας εργασίας. Θα ήθελα επίσης να ευχαριστήσω, τον συν-επιβλέποντα της διπλωματικής και υποψήφιο διδάκτορα του ΕΒΓΔ κ. Ιωάννη Ρούσο με τον οποίο είχα την τιμή να συνεργαστώ. Δε θα μπορούσα να μην αναφέρω την καθοριστική συμβολή του στην επιτυχή ολοκλήρωση της εργασίας καθώς με βοήθησε σε όποια δυσκολία κλήθηκα να αντιμετωπίσω και αφιέρωσε σημαντικό χρόνο και γνώση στην προσπάθεια μου αυτή. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την ουσιαστική συμπαράσταση τους στην μέχρι τώρα πορεία μου και ειδικότερα τον πατέρα μου που από μικρή ηλικία με ώθησε προς την κατεύθυνση της επιστήμης.

Περίληψη

Ο στόχος της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η δημιουργία του πρώτου ερευνητικού Συστήματος Διαχείρισης Βάσεων Δεδομένων, το οποίο ενσωματώνει το context στη διαχείριση της πληροφορίας. Συγκεκριμένα η διπλωματική αυτή ασχολείται με την υλοποίηση του υποσυστήματος μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων το οποίο προσφέρει αφενός τη δυνατότητα δημιουργίας context-aware βάσεων δεδομένων και αφετέρου αυτή της εκτέλεσης των ερωτημάτων.

Για να γίνει αυτό πραγματικότητα υλοποιήθηκαν τα παρακάτω υποσυστήματα: το Υποσύστημα Επικοινωνίας με το σχεσιακό ΣΔΒΔ. που είναι υπεύθυνο για την επικοινωνία με το σχεσιακό σύστημα ΣΔΒΔ (POSTGRES), το Υπόσύστημα του Καταλόγου που διαχειρίζεται μετα-data σχετικά με το context αλλά και τα context-aware δεδομένα, το Υποσύστημα Διαχείρισης Δεδομένων που είναι υπεύθυνο για τις εισαγωγές/διαγραφές/ενημερώσεις των δεδομένων στο σύστημα, το Υποσύστημα Αναπαράστασης των ερωτημάτων το οποίο μοντελοποιεί το ερώτημα και το φέρνει στην κατάλληλη προς την περεταίρω επεξεργασία του μορφή και το Υποσύστημα μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων το οποίο είναι υπεύθυνο για την τελική μετατροπή των ερωτημάτων, την εκτέλεση τους και την επιστροφή των αποτελεσμάτων.

Σημαντικό κομμάτι της διπλωματικής ήταν ο σχεδιασμός σε επίπεδο συστήματος (system software engineering) και σε επίπεδο διαχείρισης βάσεων δεδομένων (database management). Τέλος αξίζει να αναφερθεί ότι κατά τη διάρκεια της διπλωματικής υλοποιήθηκαν δύο διαφορετικές υλοποιήσεις σχετικά με την αποθήκευση των δεδομένων στο σχεσιακό ΣΔΒΔ ενώ εφαρμόστηκαν και δύο τεχνικές σχετικά με την εκτέλεση των ερωτημάτων. Επίσης πραγματοποιήθηκαν πειράματα πάνω σε μεγάλο όγκο δεδομένων τα οποία οδήγησαν στην εξαγωγή χρήσιμων συμπερασμάτων όσον αφορά την απόδοση του συστήματος αλλά το overhead που εισάγουν οι σχετικές με το context-aware λειτουργίες στο κλασικό σχεσιακό ΣΔΒΔ.

Λέξεις Κλειδιά: context, context-aware συστήματα, context-aware ΣΔΒΔ, polymorphs, morphs, μετατροπή ερωτημάτων

Abstract

The aim of this Diploma Thesis is the design and implementation of the first research relational Data Base Management System (DBMS), which integrates the context as first class citizen in the information's management. Particularly, this diploma thesis includes the design and implementation of the Subsystems of “Query Rewriting (in PL\SQL) and Execution . The implemented system offers the possibility to create context-aware databases and to execute queries.

This system is composed of the following sybsystems: the DataBase Layer, which is responsible for the communication of the system with the relational DBMS (POSTGRES), the Catalogue, which manages the metadata that refer to the context and to the context-aware data, the sybsystem of Data, which is responsible for the inserts/deletes/updates of the data in the system, the sybsystem of the query representation which is responsible for the query's suitable form which is required in the next level of the process and the sybsystem of the Query Rewriting (in PL\SQL) and Execution that finally rewrites the query, executes it and returns the result.

An important part of this diploma thesis was both the design in the system level (system software engineering) as in the database management one. Finally it is worth to mention that during this diploma thesis two different implementations have been made in the relational DBMS and also two different methods as far as the rewriting technique is concerned. Furthemore, extensive experiments have been made in a context-aware database over numerous data, according to which we came to conclusions as far as the performance of the system is concerned as well as the overhead added to the relational DBMS.

Keywords: context, context-aware systems, context-aware DBMS, polymorph, morphs, query rewriting

Πίνακας περιεχομένων

1

Εισαγωγή

Στο κεφάλαιο αυτό, αναλύουμε το αντικείμενο της διπλωματικής εργασίας στην ενότητα 1.1 ενώ στην 1.2 γίνεται μια επισκόπηση της οργάνωσης του τόμου.

1.1 Αντικείμενο της διπλωματικής

Στη σύγχρονη εποχή, η ραγδαία ανάπτυξη των νέων τεχνολογιών και ειδικότερα του διαδικτύου, στο οποίο πλέον προσφέρεται με χαρακτηριστική ευκολία κάθε είδους πληροφορία, έχει οδηγήσει στη δημιουργία μεγάλου πλήθους ετερογενών δεδομένων. Τα δεδομένα αυτά διαφοροποιούνται ανάλογα με εξωτερικούς παράγοντες, δηλαδή παράγοντες που περιγράφουν την κατάσταση τους, το περιβάλλον τους, τις αλληλεπιδράσεις τους κλπ και οι οποίοι δεν περιλαμβάνονται στα δεδομένα αυτά καθαυτά.

Από την άλλη πλευρά, οι χρήστες, λόγω πάλι της ραγδαίας ανάπτυξης των νέων τεχνολογιών, έχουν όλο και μεγαλύτερες απαιτήσεις από τις εφαρμογές που χρησιμοποιούν. Αξιώνουν εξατομικευμένα αποτελέσματα στις ερωτήσεις τους, αποτελέσματα που θα είναι φιλτραρισμένα ανάλογα με την εκάστοτε κατάσταση τους και το περιβάλλον τους, χωρίς όμως να υπεισέρχονται στη διαδικασία ορισμού ή απόδοσης τιμών στις παραμέτρους αυτές. Για παράδειγμα, ένας χρήστης κινητής τηλεφωνίας που έχει πρόσβαση σε κάποια υπηρεσία που παρέχεται από τον πάροχο του και τον συμβουλεύει σχετικά με προτεινόμενα εστιατόρια λαμβάνει τα αποτελέσματα του ερωτήματος του σε μορφή κειμένου αν χρησιμοποιεί κάποιο κινητό τερματικό σχετικά παλαιότερης τεχνολογίας, ενώ αναμένει να δει και φωτογραφίες ή ακόμα και βίντεο από τα προτεινόμενα μέρη στην περίπτωση που χρησιμοποιεί κάποιο high-end κινητό τερματικό νέας γενιάς. Η απαίτηση του αυτή δεν εδράζεται στο γεγονός ότι εκείνος όρισε σαφώς τον τρόπο παρουσίασης των δεδομένων που επιθυμεί, αλλά στο ότι η εφαρμογή θα πρέπει να επιτελέσει τους κατάλληλους ελέγχους και να στείλει τα αποτελέσματα στην πλέον κατάλληλη μορφή.

Τις εξωτερικές αυτές παραμέτρους οι οποίες χρησιμοποιούνται για το χαρακτηρισμό της κατάστασης στην οποία βρίσκεται μια οντότητα- όπου ο όρος οντότητα δεν περιορίζεται μόνο στα φυσικά πρόσωπα- οι ερευνητές τις περιγράφουν χρησιμοποιώντας τον όρο context.

Αν παρατηρήσει κανείς τις σύγχρονες τάσεις της βιομηχανίας αλλά και της έρευνας, θα διαπιστώσει μια στροφή προς την παροχή context aware υπηρεσιών σε πολλά διαφορετικά επίπεδα. Για να γίνει περισσότερο αντιληπτή η έννοια του context και των παραμέτρων που περιλαμβάνει, θα επεκτείνουμε το προαναφερθέν παράδειγμα με σκοπό να αναδείξουμε το ότι η πληροφορία αλλάζει ανάλογα με το context του χρήστη. Έστω ότι ο χρήστης που αναφέραμε προηγούμενα θέτει ένα ερώτημα στην υπηρεσία σχετικά με την εύρεση κάποιου εστιατορίου. Τότε, εκτός από την παράμετρο που αναφέραμε σχετικά με το κινητό τερματικό του και τον τρόπο απεικόνισης των αποτελεσμάτων, τα αποτελέσματα διαφοροποιούνται ανάλογα και με άλλες παραμέτρους. Καταρχάς το μεταφορικό μέσον που χρησιμοποιεί καθορίζει την μέγιστη απόσταση την οποία μπορεί να διανύσει, ενώ με παρόμοιο τρόπο επιδρούν και οι καιρικές συνθήκες. Αν μετακινείται με αυτοκίνητο μια απόσταση της τάξεως των 5 χλμ δεν θα ήταν απαγορευτική, ενώ αν βρέχει και μετακινείται με μηχανή το σύστημα δε θα είναι επιθυμητό να του προτείνει κάποιο εστιατόριο στην απόσταση των 5 χλμ. Αντίστοιχα, οι γαστρονομικές του προτιμήσεις θα μπορούσαν να περιορίσουν τα αποτελέσματα. Αν είναι χορτοφάγος μια παραδοσιακή ταβέρνα δεν θα ήταν η προτιμότερη επιλογή. Επίσης η περίσταση του γεύματος είναι παράγοντας. Αν λ.χ. πρόκειται για επαγγελματικό δείπνο πρέπει να επιλεγεί ο κατάλληλος χώρος. Και όπως είναι προφανές, η λίστα των πιθανών παραμέτρων context δεν περιορίζεται στα παραπάνω και θα μπορούσε να συμπεριλαμβάνει την ώρα, την ημερομηνία ή το ιστορικό του χρήστη.

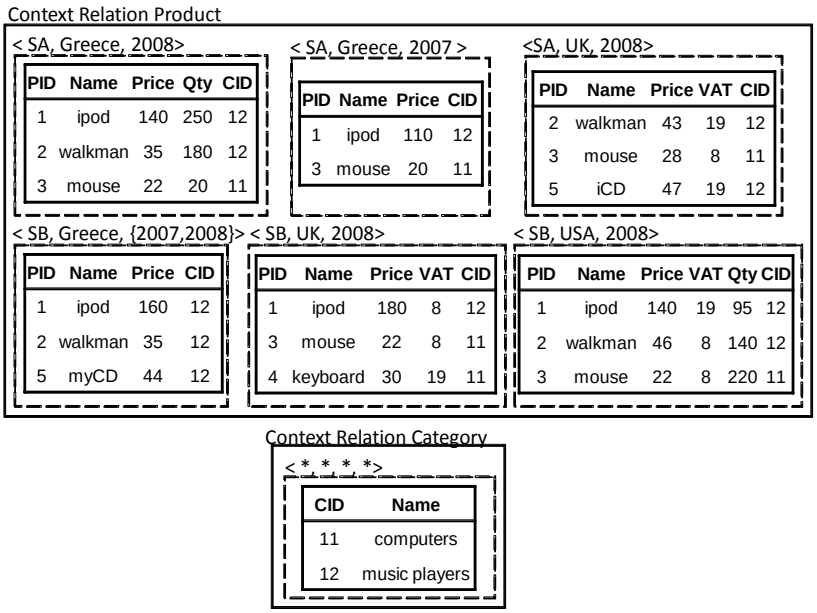
Συνοψίζοντας όσα προαναφέρθηκαν καθίσταται αναγκαία η ύπαρξη context-aware συστημάτων, δηλαδή συστημάτων που παίρνοντας ως παράμετρο μεταβλητές context παράγουν εξατομικευμένα αποτελέσματα για κάθε περίσταση. Ωστόσο μέχρι στιγμής, η διαχείριση αυτών των πληροφοριών του context παραμένει στα στενά όρια των εφαρμογών. Στο προαναφερθέν παράδειγμα, στο αίτημα του χρήστη για ένα κατάλογο με εστιατόρια, η εφαρμογή θέτει ένα ερώτημα στη βάση δεδομένων. Ως απάντηση η βάση στέλνει μια λίστα αποτελεσμάτων και στη συνέχεια η εφαρμογή είναι αποκλειστικά υπεύθυνη για την τελική επιλογή των εστιατορίων που θα σταλούν στο χρήστη. Οπότε, στο ενδεχόμενο δύο χρήστες να θέσουν το ίδιο ερώτημα στην υπηρεσία που προαναφέραμε, η βάση δεδομένων θα δώσει τα ακριβώς ίδια αποτελέσματα στην εφαρμογή η οποία είναι υπεύθυνη να φιλτράρει τα αποτελέσματα και ανάλογα με το context του καθενός θα αποκριθεί κατάλληλα. Ωστόσο μια συνολικότερη προσέγγιση του ζητήματος θα ήταν επιθυμητή. Θα ήταν προτιμότερο ανάλογα με τις παραμέτρους context του κάθε χρήστη να διαφοροποιείται η επιστρεφόμενη πληροφορία και το σχήμα της βάσης από το οποίο θα εξαχθεί το τελικό αποτέλεσμα.

Παρ’ όλα αυτά μέχρι στιγμής, αν και έχει γίνει αρκετή έρευνα γύρω από τη συλλογή και τη μοντελοποίηση του context δεν έχει γίνει κάποια προσπάθεια ενσωμάτωσης του σε κάποιο σύστημα διαχείρισης βάσεων δεδομένων. Αυτός είναι και ο στόχος της παρούσας

διπλωματικής εργασίας. Δηλαδή ο σχεδιασμός και η υλοποίηση του υποσυστήματος επεξεργασίας ερωτημάτων ενός context-aware Συστήματος Διαχείρισης Βάσεων Δεδομένων. Στα πλαίσια του συστήματος αυτού επεκτείνεται το σχεσιακό μοντέλο ώστε να υποστηρίζει νέες λειτουργίες σχετικές με το context. Στην κατεύθυνση αυτή, όπως αναφέρεται στο technical report ("A Model for Context Aware Relational Databases", I.Roussos, T.Sellis) που προτείνεται το συγκεκριμένο μοντέλο, ήταν απαραίτητη η δημιουργία νέων τελεστών που επεκτείνουν την κλασική σχεσιακή άλγεβρα, και για την υποστήριξη αυτών μια νέα γλώσσα που επεκτείνει την SQL και ονομάζεται E(extended)-SQL.

Στο σύστημα αυτό, το σχήμα της βάσης από το οποίο προκύπτει το αποτέλεσμα κάθε ερωτήματος, διαφοροποιείται ανάλογα με το context του χρήστη. Για την υποστήριξη των νέων αυτών σχέσεων με πολλαπλά σχήματα ορίζονται καταρχάς σχήματα context τα οποία περιλαμβάνουν τις παραμέτρους context που θα χρησιμοποιηθούν. Στη συνέχεια ορίζονται εκτεταμένες σχέσεις που ονομάζονται Polymorphs και οι οποίες αποτελούνται από ένα σύνολο από τις κλασικές σχέσεις (ονομάζονται morphs στο σύστημα), με τη σημείωση ότι κάθε morph ορίζεται για ένα στιγμιότυπο του σχήματος context. Αυτό έχει ως αποτέλεσμα, με βάση το εκάστοτε context, η απάντηση για το ίδιο ερώτημα να διαφοροποιείται ανάλογα με τις μεταβολές στο context τόσο όσον αφορά το πλήθος και τις τιμές των εγγραφών που επιστρέφονται, όσο και αναφορικά με το schema του τελικού αποτελέσματος.

Για παράδειγμα, έστω μια πολυεθνική εταιρεία που πουλάει προϊόντα σε διάφορες χώρες. Ένας χρήστης επισκέπτεται το site της εταιρείας αυτής και ανάλογα με το context του, βλέπει τα διαθέσιμα σε αυτόν προϊόντα, τις αντίστοιχες τιμές κλπ. Το σχήμα της βάσης από την οποία θα προκύψει το τελικό αποτέλεσμα αλλάζει σε ενδεχόμενο ερώτημα του σχετικά με τα διαθέσιμα σε αυτόν προϊόντα ανάλογα με τη χώρα προέλευσής του ή την ημερομηνία που τον ενδιαφέρει. Όμως, μπορεί να εκτελεί και ερωτήματα τα οποία να εμφανίζουν για κάποιο προϊόν τη τιμή του σε όλες τις χώρες που πωλείται, οπότε να κάνει σύγκριση των τιμών αυτών. Για τα δεδομένα μιας τέτοιας περίπτωσης δίνεται στο παρακάτω σχήμα το στιγμιότυπο των polymorphs product και category μιας βάσης δεδομένων. Το σχήμα context που χρησιμοποιήθηκε είναι το <supplier, country, year>. Δηλαδή ο ίδιος προμηθευτής μπορεί πουλάει διαφορετικά προϊόντα με διαφορετικά χαρακτηριστικά (τιμή, φόρους κλπ), ανάλογα με τη χώρα και το έτος. Στο σημείο αυτό πρέπει να σημειώσουμε ότι ανάλογα με το context, υπάρχει η δυνατότητα να αλλάζει και το σχήμα του κάθε morph. Για παράδειγμα μπορεί σε μια χώρα να υπάρχει ένας ειδικός φόρος κατανάλωσης ο οποίος να μην υπάρχει σε άλλες. Τότε στο morph που ορίζεται για το στιγμιότυπο context που περιλαμβάνει τη χώρα αυτή υπάρχει το αντίστοιχο attribute για το φόρο ενώ δεν υπάρχει στα υπόλοιπα morphs.


Context Relation Product
< SA, Greece, 2008>
PID Name Price Qty CID
1 ipod 140 250 12
2 walkman 35 180 12
3 mouse 22 20 11
< SA, Greece, 2007 >
PID Name Price CID
1 ipod 110 12
3 mouse 20 11
<SA, UK, 2008>
PID Name Price VAT CID
2 walkman 43 19 12
3 mouse 28 8 11
5 iCD 47 19 12
< SB, Greece, {2007,2008}>
PID Name Price CID
1 ipod 160 12
2 walkman 35 12
5 myCD 44 12
< SB, UK, 2008>
PID Name Price VAT CID
1 ipod 180 8 12
3 mouse 22 8 11
4 keyboard 30 19 11
< SB, USA, 2008>
PID Name Price VAT Qty CID
1 ipod 140 19 95 12
2 walkman 46 8 140 12
3 mouse 22 8 220 11

Context Relation Category
< * * * *>
CID Name
11 computers
12 music players

σχετικές με το θέμα εργασίες. Τέλος περιγράφεται το μοντέλο στο οποίο βασίστηκε η παρούσα διπλωματική.

Στο **τρίτο κεφάλαιο** γίνεται εκτενής αναφορά στην αρχιτεκτονική του συνολικού συστήματος αλλά και των επιμέρους υποσυστημάτων του.

Στο **τέταρτο κεφάλαιο** γίνεται η περιγραφή της υλοποίησης του συστήματος. Αρχικά τεκμηριώνεται η επιλογή της πλατφόρμας και των προγραμματιστικών εργαλείων που επιλέχθηκαν για την ανάπτυξη ενώ στη συνέχεια περιγράφονται οι κλάσεις του κώδικα του υποσυστήματος.

Το **πέμπτο κεφάλαιο** πραγματεύεται τον έλεγχο του συστήματος. Παρουσιάζεται αρχικά η μεθοδολογία του ελέγχου και κατόπιν παρατίθενται αναλυτικά τα αποτελέσματα του ελέγχου τα οποία αρχικά εφαρμόστηκαν σε μια μικρή πολυμορφική βάση δεδομένων. Στη συνέχεια παρατίθενται πειράματα με τα αποτελέσματα τους πάνω από μεγάλο όγκο δεδομένων ενώ στο τέλος εξάγονται συμπεράσματα σχετικά με την απόδοση του συστήματος.

Στο **έκτο κεφάλαιο**, που αποτελεί τον επίλογο της διπλωματικής, γίνεται επισκόπηση της εργασίας και παρουσιάζονται ορισμένες ιδέες, που αφορούν βελτιώσεις και μελλοντικές επεκτάσεις του συστήματος.

Στο **έβδομο κεφάλαιο**, τέλος, δίνεται η βιβλιογραφία και γενικότερα οι πηγές από τις οποίες αντλήθηκαν οι απαραίτητες πληροφορίες για τη συγγραφή της διπλωματικής.

2

Περιγραφή θέματος

Στο κεφάλαιο αυτό παρουσιάζουμε εκτενέστερα το αντικείμενο της διπλωματικής και την έρευνα γύρω από αυτό. Αρχικά γίνεται μια εισαγωγή στις έννοιες του context και του context-aware και στον τρόπο μοντελοποίησης τους. Στη συνέχεια παρουσιάζονται οι σχετικές με το χώρο εργασίες και τελικά το μοντέλο στο οποίο βασίστηκε η παρούσα διπλωματική εργασία.

2.1Εισαγωγή στις έννοιες του context και του context-awareness

Για να καταλάβουμε καλύτερα πώς μπορούμε να χρησιμοποιήσουμε το context ώστε να διευκολύνουμε τη δημιουργία context-aware εφαρμογών είναι απαραίτητο να κατανοήσουμε πλήρως από τι αποτελείται μια context-aware εφαρμογή καθώς και τί ακριβώς αντιπροσωπεύει το context. Και αυτό διότι, η κατανόηση της έννοιας του context θα δώσει στους σχεδιαστές εφαρμογών τη δυνατότητα να επιλέγουν το είδος του context που θα περιλαμβάνουν οι εφαρμογές τους και η κατανόηση της έννοιας του context-aware θα τους βοηθά να αποφασίζουν ποιες context-aware συμπεριφορές να ενσωματώνουν.

Πριν περιγράψουμε τις παραπάνω έννοιες θα πρέπει αρχικά να απαντήσουμε στο ερώτημα σχετικά με τον τρόπο με τον οποίο θα παρέχεται το context στους υπολογιστές ώστε από εκεί και πέρα να συνεχίσουμε με την επεξεργασία του. Μια προσέγγιση θα ήταν αυτό να εκφράζεται ρητά από τους χρήστες των εφαρμογών, οι οποίοι θα έπρεπε με κάποιον τρόπο να εκφράζουν όλες τις πληροφορίες σχετικά με κάποια κατάσταση. Ωστόσο κάτι τέτοιο φαντάζει δύσκολο για τον απλό χρήστη και συν τοις άλλοις η όλη ιδέα του context-aware υπολογιστικού συστήματος στηρίζεται στη διευκόλυνση της αλληλεπίδρασης ανθρώπου-μηχανής. Επιπροσθέτως, πολύ συχνό θα ήταν το φαινόμενο ένας χρήστης να μην γνωρίζει ακριβώς ποια πληροφορία είναι σχετική με την κατάσταση που έχει κληθεί να περιγράψει. Σαν συνέπεια των παραπάνω, στην προσέγγιση που θα χρησιμοποιήσουμε σχετικά με τις

context-aware εφαρμογές θα θεωρήσουμε ότι όλη η πληροφορία διατίθεται στο περιβάλλον εργασίας του υπολογιστή με αυτοματοποιημένα μέσα και από εκείνο το σημείο και έπειτα ο σχεδιαστής της εφαρμογής καλείται να αποφασίσει ποια πληροφορία είναι σχετική και ποια όχι καθώς και τον τρόπο με τον οποίο θα την αξιοποιήσει.

2.1.1Context

Γεγονός είναι ότι ο καθένας, λιγότερο ή περισσότερο, έχει κάποια αντίληψη σχετικά με την έννοια του context. Ωστόσο μια θολή και ακαθόριστη γενική εικόνα δεν είναι αρκετή. Πρέπει να γίνει σαφές, τι ακριβώς αντιπροσωπεύει το context, καθώς μόνο έτσι θα μπορέσει να γίνει αποτελεσματική χρήση του. Στη συνέχεια, θα παραθέσουμε αρχικά τις προσπάθειες ορισμού της έννοιας αυτής και στη συνέχεια θα δώσουμε την άποψη που έχει επικρατήσει περισσότερο στις μέρες μας.

Αρχικά λοιπόν, η πρώτη προσπάθεια περιγραφής του context μιας κατάστασης, περιελάμβανε τις έννοιες της τοποθεσίας, των ανθρώπων και των αντικειμένων που βρίσκονταν κοντά καθώς και τις αλλαγές σε αυτά. Αργότερα, στην ίδια λογική, προστέθηκαν στα παραπάνω οι έννοιες της ώρας, της εποχής, της θερμοκρασίας κλπ. Μια άλλη προσέγγιση περιέγραφε το context ως την ψυχολογική κατάσταση του χρήστη, την εστίασης της προσοχής του, την τοποθεσία, τον προσανατολισμό του, τη μέρα και την ώρα, τους ανθρώπους και τα αντικείμενα κοντά του. Άλλες ερμηνείες παρουσίασαν το context ως το περιβάλλον του χρήστη το οποίο γνωρίζει ο υπολογιστής, άλλες ως την κατάσταση του χρήστη και άλλες περιελάμβαναν ολόκληρο το περιβάλλον του χρήστη και όριζαν ως context τις διαφορετικές πτυχές της τρέχουσας κατάστασης κάθε φορά. Κάποιοι άλλοι εστίασαν περισσότερο στην εφαρμογή παρά στον χρήστη και περιέγραψαν το context ως την κατάσταση των οντοτήτων πέριξ της εφαρμογής ή ακόμα και τον τρόπο τοποθέτησής της ενώ άλλοι ισχυρίστηκαν ότι στο context ανήκει το υποσύνολο των φυσικών και εννοιολογικών καταστάσεων ενδιαφέροντος για μια ιδιαίτερη οντότητα.

Από τα παραπάνω γίνονται σαφή δύο πράγματα. Καταρχήν, στην περίπτωση που χρησιμοποιήσουμε κάποιον από τους παραπάνω ορισμούς και κληθούμε να αποφασίσουμε αν κάποιο είδος πληροφορίας ανήκει ή όχι στο τρέχον context, αυτό δεν είναι δυνατό να γίνει με απόλυτο και σαφή τρόπο. Κατά δεύτερον, οι περισσότεροι από τους παραπάνω ορισμούς περιλαμβάνουν το πνεύμα το οποίο θα επιθυμούσαμε να περιλαμβάνει ο ορισμός του context. Δηλαδή, αν συνοψίσουμε τα παραπάνω μπορούμε να ισχυριστούμε ότι στο context περιλαμβάνεται η έννοια του διαρκώς μεταβαλλόμενου περιβάλλοντος, όπου στο περιβάλλον αυτό περιλαμβάνονται:

- το περιβάλλον του υπολογιστή, δηλαδή οι επεξεργαστές, οι συσκευές I/O, απεικόνισης, η δικτύωση, η συνδεσιμότητα κλπ
- το περιβάλλον του χρήστη, δηλαδή η τοποθεσία που βρίσκεται, τα αντικείμενα και οι άνθρωποι γύρω του, η κατάσταση του
- το φυσικό περιβάλλον, δηλαδή ο θόρυβος, η θερμοκρασία, ο φωτισμός κλπ

Όμως πρέπει να παρατηρήσουμε ότι όλοι οι παραπάνω ορισμοί είναι ελλιπείς. Και αυτό συμβαίνει διότι είναι πολύ συγκεκριμένοι. Αντιθέτως η έννοια του context είναι περισσότερο αφηρημένη καθώς εν ολίγοις περιλαμβάνει τα πάντα σχετικά με μια κατάσταση και τους χρήστες που την απαρτίζουν. Δεν μπορούμε να περιορίσουμε τις πτυχές που θα θεωρούμε σημαντικές διότι αυτές είναι φυσικό να αλλάζουν ανάλογα με τις συνθήκες . Συνεπώς καταλήγουμε στον παρακάτω ορισμό:

Ως context ορίζεται κάθε πληροφορία που μπορεί να χρησιμοποιηθεί για την περιγραφή της κατάστασης μια οντότητας, όπου οντότητα θεωρούνται τα άτομα, οι τοποθεσίες και τα αντικείμενα που θεωρούνται σχετικά με την αλληλεπίδραση ανθρώπου-εφαρμογής, του ανθρώπου και της εφαρμογής συμπεριλαμβανομένων.[DA00]

Όπως γίνεται εύκολα αντιληπτό, ο παραπάνω ορισμός διευκολύνει το σχεδιαστή της εφαρμογής να απαριθμήσει τους παράγοντες του context για ένα συγκεκριμένο σενάριο εφαρμογής. Αν ένα κομμάτι πληροφορίας χρησιμοποιείται για να περιγράψει την κατάσταση ενός συμμετέχοντα στην αλληλεπίδραση τότε ανήκει στο context. Αν όχι, δεν ανήκει. Εδώ για λόγους σαφήνειας θα πρέπει να αναφέρουμε ότι με τον παραπάνω ορισμό δε γίνεται διάκριση ανάμεσα στην πληροφορία που δηλώνεται ρητά και σε εκείνη που προκύπτει από κάποια περιγραφή (υπονοούμενη πληροφορία). Δηλαδή, για παράδειγμα, είτε η ταυτότητα ενός χρήστη προκύπτει ρητά από την εισαγωγή των διαπιστευτηρίων του σε μια φόρμα είτε υπονοείται από την οπτική επαφή του με κάποιον άλλο χρήστη, η ταυτότητα αυτή παραμένει κομμάτι του context.

2.1.1.1Κατηγοριοποίηση του context

Η κατηγοριοποίηση του context θα επιτρέψει στους σχεδιαστές εφαρμογών να ανακαλύψουν τα πιο πιθανά είδη context που θα είναι χρήσιμα για τις εφαρμογές τους. Επιπλέον, θα τους βοηθήσει να δομήσουν το ήδη υπάρχον context που χρησιμοποιούν.

Στην προσπάθεια για την κατηγοριοποίηση αυτή, παρατηρούμε ότι στην πράξη ορισμένα είδη context είναι σημαντικότερα από άλλα. Αυτά είναι: η τοποθεσία , το προφίλ του χρήστη, η δραστηριότητα και ο χρόνος τα οποία και αποτελούν τους πρωταρχικούς τύπους context για το χαρακτηρισμό της κατάστασης μιας οντότητας. Ο λόγος που επιλέχθηκαν αυτά τα είδη είναι ότι όχι μόνο απαντούν στα ερωτήματα ποιος, τί, πότε και πού, που χαρακτηρίζουν

πλήρως μια κατάσταση αλλά επίσης αποτελούν δείκτες προς άλλες πηγές πληροφορίας context. Για να γίνει αυτό περισσότερο κατανοητό ας υποθέσουμε ότι έχουμε ως πληροφορία την ταυτότητα ενός ατόμου. Με βάση αυτήν, μπορούμε να αποκτήσουμε περισσότερες πληροφορίες σχετικές με αυτό το άτομο όπως το τηλέφωνο του, τη διεύθυνση του, τη λίστα των φίλων του κλπ. Αυτές τις πληροφορίες, δηλαδή αυτού του τύπου το context, που προκύπτει χρησιμοποιώντας ως δείκτες τους πρωταρχικούς τύπους, θα το ονομάζουμε δευτερεύον.

Σύμφωνα με την παραπάνω κατηγοριοποίηση έχουμε συνεπώς ένα σύστημα δύο επιπέδων το οποίο διευκολύνει τόσο τη χρήση όσο και την οργάνωση του context. Στο $1^{ο}$ ανήκουν οι πρωταρχικοί τύποι που αναφέρθηκαν παραπάνω και στο $2^{ο}$ όλοι οι υπόλοιποι οι οποίοι μοιράζονται ένα κοινό χαρακτηριστικό: μπορούν να εντοπιστούν με βάση κάποιον πρωτεύοντα τύπο context διότι αποτελούν ιδιότητες της οντότητας που χαρακτηρίζεται από το πρωταρχικό context. Έτσι τις περισσότερες φορές θα αρκεί ένας πρωταρχικός τύπος για την εξαγωγή κάποιων δευτερευόντων. Υπάρχει βέβαια και η περίπτωση όπου θα απαιτείται συνδυασμός των πρωταρχικών τύπων για την εύρεση των δευτερευόντων. Για παράδειγμα, για την πρόγνωση του καιρού, απαιτούνται πληροφορίες σχετικά τόσο με την τοποθεσία όσο και με το χρονικό σημείο της πρόβλεψης.

2.1.1.2 Μοντέλα Context

Από τη στιγμή που ορίσαμε την έννοια του context το επόμενο βήμα είναι η περιγραφή των μοντέλων του. Η περιγραφή αυτή βασίστηκε στο [SP04]. Με αυτά γίνεται μια προσπάθεια ενιαίας μοντελοποίησης του που τελικά θα έχει ως αποτέλεσμα τη διευκόλυνση στο διαμοιρασμό του ούτως ώστε σε τελικό στάδιο διαφορετικές εφαρμογές να μπορούν να λειτουργούν κάτω από ένα κοινό πλαίσιο.

<u>Θεμελιώδη χαρακτηριστικά μοντέλων context</u>

Στην κατεύθυνση αυτή και πριν περιγράψουμε τα σημαντικότερα μοντέλα που έχουν προταθεί πρέπει να αναφέρουμε ορισμένα χαρακτηριστικά που θεωρούνται θεμελιώδους σημασίας για κάθε προσέγγιση μοντελοποίησης του context. Αυτά είναι τα παρακάτω:

Η κατανεμημένη σύνθεση (distributed composition): η σύνθεση και η διαχείριση κάθε μοντέλου context αλλάζουν δυναμικά ανάλογα με το χρόνο, την τοπολογία και τις πηγές του δικτύου και αυτό είναι κάτι που πρέπει να υποστηρίζεται από το μοντέλο του context

Η μερική επικύρωση (partial validation): η δυνατότητα της μερικής επικύρωσης της context γνώσης με βάση κάποιο πρότυπο είναι άκρως επιθυμητή καθώς είναι πιθανόν να μην υπάρχει κανένα σημείο του χώρου ή του χώρου όπου θα υπάρχει συσσωρευμένη η γνώση αυτή. Αυτό

είναι αποτέλεσμα της κατανεμημένης σύνθεσης. Το χαρακτηριστικό αυτό είναι ιδιαίτερα σημαντικό αν αναλογιστούμε τις αλληλοσυσχετίσεις ανάμεσα στο context που κάνουν κάθε προσπάθεια μοντελοποίησης επιρρεπή σε λάθη

Ο πλούτος και η ποιότητα της πληροφορίας (richness and quality of information): η ποιότητα της πληροφορίας που συλλέγεται από τους αισθητήρες όπως είναι φυσικό ποικίλει καθώς εξαρτάται από διάφορους εξωγενείς εκτός των άλλων παράγοντες. Επίσης η χρήση διαφορετικών ειδών αισθητήρων έχει ως αποτέλεσμα τη συλλογή ποικίλης πληροφορίας και συνεπώς τον πλούτο της συγκεντρωμένης πληροφορίας. Συνεπώς ένα κατάλληλο μοντέλο για τη μοντελοποίηση του context θα πρέπει να προνοεί και για τα δύο αυτά χαρακτηριστικά

Η ατέλεια και ασάφεια (incompleteness and ambiguity): το σύνολο της διαθέσιμης πληροφορίας σχετικά με τις οντότητες μια κατάστασης σε μια τυχαία στιγμή είναι πιθανόν να είναι ατελές και ασαφές, ειδικά αν αναλογιστούμε τον τρόπο συλλογής της από τα δίκτυα αισθητήρων. Προφανώς κάθε μοντέλο context θα πρέπει να παρέχει κάποια μέριμνα σχετικά με το ζήτημα αυτό

Το επίπεδο φορμαλισμού (level of formality): η περιγραφή γεγονότων σχετικών με το context αλλά και των αλληλοσυσχετίσεων με σαφή και ανιχνεύσιμο τρόπο αποτελεί μια πρόκληση στην οποία πρέπει να ανταπεξέλθουν τα μοντέλα του context. Είναι άκρως επιθυμητό κάθε μέρος που συμμετέχει σε μια αλληλεπίδραση να μεταφράζει με τον ίδιο τρόπο τα δεδομένα προς ανταλλαγή (shared understanding)

Η εφαρμοσιμότητα σε υπάρχοντα περιβάλλοντα (applicability to existing environments): κοιτάζοντας από την σκοπιά της εφαρμογής των μοντέλων είναι σημαντικό αυτά να είναι εφαρμόσιμα μέσα στην υποδομή των ευρέως διαδεδομένων υπολογιστικών συστημάτων

Προσεγγίσεις μοντέλων context

Παρακάτω δίνονται τα κυριότερα μοντέλα για την μοντελοποίηση του context ακολουθούμενα από μια σύντομη περιγραφή σε κάθε περίπτωση:

Το Μοντέλο τιμής-κλειδιού (key-value models) αποτελεί την απλούστερη περίπτωση δόμησης των δεδομένων για την μοντελοποίηση της πληροφορίας του context. Χρησιμοποιούνται ζευγάρια τιμής-κλειδιού και η μοντελοποίηση γίνεται παρέχοντας την τιμή της πληροφορίας context σε μια εφαρμογή ως μεταβλητή του περιβάλλοντος της. Όπως γίνεται εύκολα αντιληπτό τα ζευγάρια τιμής-κλειδιού είναι εύκολα να διαχειρισθούν αλλά δεν παρέχουν δυνατότητες για περισσότερο περίπλοκη δόμηση και συνεπώς ούτε για τη χρησιμοποίηση αποδοτικών αλγορίθμων ανάκτησης του context.

Τα Μοντέλα σχεδίου σήμανσης (markup scheme models) έχουν ως κοινό χαρακτηριστικό την ιεραρχική δόμηση των δεδομένων με χρήση ετικετών με ιδιότητες και περιεχόμενο το οποίο

συνήθως ορίζεται αναδρομικά από άλλες ετικέτες επισήμανσης. Τυπικό παράδειγμα αυτού του είδους της μοντελοποίησης είναι τα προφίλ. Συνήθως στηρίζονται στην SGML (Standard Generic Markup Language) ενώ άλλα μοντέλα ορίζονται ως επεκτάσεις των προτύπων CC/PP (Composite Capabilities / Preferences Profile) και UAProf(User Agent Profile) τα οποία παίρνουν τις μεθόδους έκφρασης από το RDF σχήμα και τη σειριακή διάταξη της XML. Στη συνέχεια προσπαθούν να επεκτείνουν τα στατικά αυτά προφίλ ώστε να υποστηρίζεται η δυναμική και η πολυπλοκότητα της πληροφορίας του context. Παράδειγμα αυτή της προσέγγισης είναι τα CSCP (Comprehensive Structured Context Profiles) τα οποία σε αντίθεση με τα προηγούμενα δεν ορίζουν κάποια σταθερή ιεραρχία αλλά αντίθετα υποστηρίζουν η ευλυγισία του RDF σχήματος όπου τα ονόματα των ιδιοτήτων μεταφράζονται με γνώμονα το context σύμφωνα με τη θέση τους στη δομή του προφίλ. Ένα άλλο παράδειγμα, παρόμοιο με τα CSCP είναι η επέκταση των προτύπων CC/PP (CC/PP Context Extension). Σύμφωνα με αυτά επεκτείνεται το λεξιλόγιο των CC/PP και UAProf και προστίθενται δένδρα ιδιοτήτων σχετιζόμενα με κάποια πτυχές του context (π.χ τοποθεσία, χαρακτηριστικά δικτύου κλπ). Τέλος μια διαφορετική προσέγγιση στη μοντελοποίηση αυτής της κατηγορίας που δεν στρέφεται προς τα CC/PP είναι η PPDL (Pervasive Profile Description Language). Αυτή η γλώσσα που βασίζεται στην XML έχει ως κύριο χαρακτηριστικό τη δυνατότητα που προσφέρει σχετικά με το να λαμβάνεται υπ ‘όψιν η πληροφορία context και οι εξαρτήσεις όταν ορίζονται σχήματα αλληλεπίδρασης.

Στα Γραφικά μοντέλα (Graphical Models) δύο είναι οι κύριες τάσεις μοντελοποίησης. Η πρώτη αφορά τη γλώσσα UML και η δεύτερη μια επέκταση του ORM (Object-Role Modeling). Όσον αφορά τη γλώσσα UML γνωρίζουμε ότι είναι ένα ευρέως διαδεδομένο εργαλείο μοντελοποίησης με ισχυρά γραφικά χαρακτηριστικά λόγω των UML διαγραμμάτων. Λόγω της γενικής χρήσης της δομής της είναι κατάλληλη και για τη μοντελοποίηση και του context. Όσον αφορά την επέκταση του ORM (Henricksen et al) γνωρίζουμε ότι η βασική έννοια γύρω από την οποία περιστρέφεται το ORM είναι το γεγονός και συνεπώς η μοντελοποίηση περικλείει την αναγνώριση των τύπων των γεγονότων και το ρόλο που οι οντότητες διαδραματίζουν σε αυτά. Στην επέκταση του ORM προστέθηκε η δυνατότητα κατηγοριοποίησης των τύπων των γεγονότων είτε ως στατικά είτε ως δυναμικά. Η τελευταία αυτή κατηγορία (τα δυναμικά γεγονότα) διαχωρίζονται ακόμα περισσότερο ανάλογα με τη πηγή των γεγονότων σε τύπους προφίλ, εκείνους που προέρχονται από κάποια αίσθηση και στους παραγόμενους τύπους. Επίσης προστέθηκε μια χρονική συνιστώσα του context για την περιγραφή των ιστορικών τύπων γεγονότων. Επιπρόσθετα, στην προτεινόμενη επέκταση προστέθηκαν και εξαρτήσεις ανάμεσα στα γεγονότα (σχέση dependsOn). Τέλος αξίζει να αναφέρουμε ότι από τα μοντέλα αυτού του είδους μπορεί να προκύψει σχετικά εύκολα ένα E-R διάγραμμα, που είναι πολύ χρήσιμο ως

εργαλείο δόμησης μιας σχεσιακής βάσης δεδομένων σε ένα πληροφοριακό σύστημα με αρχιτεκτονική βασιζόμενη στο context.

Τα Αντικειμενοστραφή μοντέλα (Object Oriented Models) έχουν ως κοινό χαρακτηριστικό την πρόθεση να εκμεταλλευτούν τα χαρακτηριστικά της αντικειμενοστραφούς προσέγγισης (ενθυλάκωση, επαναχρησιμοποίηση) για να αντιμετωπίσουν τα προβλήματα που προκύπτουν από το δυναμικό χαρακτήρα του context. Έτσι, οι λεπτομέρειες της επεξεργασίας του context ενθυλακώνονται στο επίπεδο των αντικειμένων (objects) και συνεπώς δεν είναι ορατά στα υπόλοιπα μέρη της εφαρμογής. Η πρόσβαση στις πληροφορίες context δε, γίνεται μόνο μέσω των αυστηρά ορισμένων διαπροσωπειών (interfaces). Αντιπροσωπευτικό παράδειγμα της προσέγγισης αυτής είναι τα cues που αναπτύχθηκαν στα πλαίσια του TEA project. Λειτουργούν ως συναρτήσεις που δέχονται ως είσοδο μια τιμή από έναν αισθητήρα και παρέχουν μια έξοδο από κάποιο προκαθορισμένο σύνολο. Το context μοντελοποιείται ως ένα αφαιρετικό επίπεδο στην κορυφή των cues και έτσι αυτά είναι αντικείμενο τα οποία κρύβουν τις ιδιότητες τους και παρέχουν τις πληροφορίες context μέσω των ορισμένων διαπροσωπειών. Άλλα παραδείγματα αυτής της προσέγγισης είναι το Active Object Model (Guide Project) και η προσέγγιση των Bouzy-Cazenave. Και στις δύο περιπτώσεις επιλέχθηκαν τα αντικειμενοστραφή μοντέλα λόγων των πλεονεκτημάτων που προσφέρουν, της ενθυλάκωσης στην 1η περίπτωση και της κληρονομικότητας και της επαναχρησιμοποίησης στη 2η.

Στα Λογικά μοντέλα (Logic Based Models), όπως άλλωστε είναι αναμενόμενο, για τον ορισμό του context χρησιμοποιούνται γεγονότα, εκφράσεις και κανόνες ενώ κοινό χαρακτηριστικό τους αποτελεί το υψηλό επίπεδο φορμαλισμού. Μια πρώτη προσπάθεια χρήσης αυτής της προσέγγισης δημοσιεύτηκε το 1993 ως “Formalizing Context” (McArthy) στην οποία το context παρουσιάστηκε ως αφηρημένες μαθηματικές οντότητες με ιδιότητες χρήσιμες στην τεχνητή νοημοσύνη. Οι κανόνες που συσχέτιζαν την αλήθεια σε ένα context σε σχέση με εκείνη σε ένα άλλο αποτέλεσαν σημαντικό κομμάτι της προσέγγισης αυτής στην οποία τη βασική σχέση ήταν η ist(c,p) που ισχυριζόταν ότι η πρόταση p ήταν αληθής στο context c ενώ υποστηριζόταν και η κληρονομικότητα. Μια δεύτερη προσέγγιση στην κατεύθυνση αυτή (Giunchiglia) γνωστή και ως “Multicontext Systems” στράφηκε περισσότερο στη συλλογιστική του context (reasoning) ενώ μια τρίτη γνωστή και ως “Extended Situation Theory” (Akman, Surav) επέκτεινε τη θεωρία της κατάστασης (situation theory) ώστε να μοντελοποιηθεί το context ως τύπους κατάστασης που ήταν αντικείμενα πρώτης τάξης της θεωρίας αυτής. Τέλος άλλες προσπάθειες στην προσέγγιση αυτή είναι γνωστές ως “Sensed Context Model” (Grat,Salber) και “Multimedia System” (Bacon).

Τα Μοντέλα με χρήση οντολογιών (Ontology Based Models) χρησιμοποιούν οντολογίες οι οποίες είναι ένα πολλά υποσχόμενο εργαλείο για την περιγραφή εννοιών και

αλληλοσυσχετίσεων. Ειδικότερα μπορούν να περιγράψουν έννοιες και πληροφορίες της καθημερινότητας και να τις παρουσιάσουν σε δομές δεδομένων κατάλληλες για τη χρήση από τους υπολογιστές. Μια πρώτη προσέγγιση στα μοντέλα αυτά έγινε από τους Otzturk και Aamodt οι οποίες αναλύοντας ψυχολογικές μελέτες σχετικά με τη διαφορά των εννοιών της ανάκλησης (recall) και της αναγνώρισης (recognition) σχετικά με αρκετά ζητήματα, σε συνδυασμό με πληροφορίες context κατέληξαν στην αναγκαιότητα κανονικοποίησης και συνδυασμού της γνώσης από διαφορετικούς τομείς. Έτσι πρότειναν ένα τρόπο μοντελοποίησης του context με χρήση οντολογιών. Μια δεύτερη προσέγγιση στην ίδια κατεύθυνση, γνωστή και ως μοντέλο ASC (Aspect-Scale-ContextInformation) χρησιμοποίησε κριτές (reasoners) οντολογιών για την αποτίμηση της γνώσης. Υλοποιήθηκε με χρήση επιλεγμένων γλωσσών αναπαράστασης οντολογιών με αποτέλεσμα τη δημιουργία του πυρήνα της γλώσσας CoOL που χρησιμοποιείται για την υποστήριξη context-aware υπηρεσιών σε κατανεμημένα πλαίσια εργασίας σε ποικίλες εφαρμογές. Ακόμα η μοντελοποίηση του context γνωστή και ως "CANON" είχε ως αποτέλεσμα τη δημιουργία μιας υπερ-οντολογίας που συγκέντρωνε γενικά χαρακτηριστικά των βασικών context οντοτήτων καθώς και μια κατηγοριοποίηση και συλλογή των εξειδικευμένων οντοτήτων και των χαρακτηριστικών τους σε κάθε τομέα. Τέλος μια τελευταία προσέγγιση στα μοντέλα αυτά είναι το σύστημα "CoBra" το οποίο παρέχει ένα σύνολο οντολογικών εννοιών για το χαρακτηρισμό των οντοτήτων (άτομα, μέρη κλπ) και χρησιμοποιεί μια αρχιτεκτονική ως broker-centric. Σύμφωνα με την αρχιτεκτονική αυτή μείζονος σημασίας είναι ένας πράκτορας ονόματι context-broker (μεσίτης) ο ρόλος του οποίου είναι να διατηρήσει ένα συνεπές μοντέλο context το οποίο να μπορεί να διαμοιραστεί σε όλες τις υπολογιστικές οντότητες του χώρου αλλά και να επιβάλλει τις προκαθορισμένες από το χρήστη πολιτικές σχετικά με την ιδιωτικότητα.

2.1.2 Context-Aware Computing

Από τη στιγμή που ορίσαμε την έννοια του context και αναλύσαμε τις κατηγορίες στις οποίες χωρίζεται αλλά και τα μοντέλα στα οποία μπορεί να αναπαρασταθεί το επόμενο στάδιο είναι ο τρόπος με τον οποίο θα μπορέσουμε να το χρησιμοποιήσουμε. Όμως για να συμβεί αυτό θα πρέπει να ορίσουμε την έννοια του context-aware,να περιγράψουμε τα χαρακτηριστικά των context-aware εφαρμογών καθώς και τις τεχνικές διαχείρισης των context-aware δεδομένων.

Ο πρώτος ορισμός των context-aware εφαρμογών επέκτεινε την έννοια της πληροφόρησης και περιελάμβανε την έννοια της προσαρμοστικότητας μιας εφαρμογής ανάλογα με το εισερχόμενο context. Σε αυτήν την κατεύθυνση το context-aware σύστημα έγινε συνώνυμο με όρους όπως: προσαρμόσιμο, αντιδραστικό, αποκριτικό, ευρισκόμενο, context-ευαίσθητο και κατευθυνθέν στο περιβάλλον. Εν συντομία οι παραπάνω ορισμοί, κατατάσσονται σε δύο

κατηγορίες: σε αυτούς που χρησιμοποιούν το context και σε αυτούς που προσαρμόζονται σε αυτό.

Όσον αφορά την πρώτη κατηγορία, δηλαδή τις εφαρμογές που χρησιμοποιούν το context, μια πρώτη προσέγγιση ήταν ότι context-aware ήταν η ικανότητα των υπολογιστικών συστημάτων να ανιχνεύουν, να διαισθάνονται, να μεταφράζουν και να αντιδρούν στις πτυχές του περιβάλλοντος του χρήστη και της εφαρμογής (Hull et al, Pascoe et al). Στη συνέχεια η έννοια περιορίστηκε στη διαπροσωπεία ανθρώπου-υπολογιστή (Dey) και προστέθηκε σε αυτήν η ιδέα της προσαρμοστικότητας η οποία οδηγούσε ένα σύστημα στην αυτοματοποίησή του ανάλογα με το εισερχόμενο context του χρήστη. Στην ίδια κατεύθυνση στον ορισμό προστέθηκε η έννοια της μέγιστης ευλυγισίας μιας εφαρμογής σε σχέση με το εισερχόμενο σε πραγματικό χρόνο context (Salber et al).

Όσον αφορά τη δεύτερη κατηγορία, δηλαδή τις εφαρμογές που προσαρμόζονται στο context, οι περισσότεροι ερευνητές ορίζουν ως context-aware εφαρμογές, τις εφαρμογές εκείνες που αλλάζουν δυναμικά ή προσαρμόζουν τη συμπεριφορά τους βασισμένες στο context του χρήστη και της εφαρμογής. Πιο συγκεκριμένα μια πρώτη προσέγγιση ορίζει τις context-aware εφαρμογές ως εκείνες που παρακολουθούν αισθητήρες του περιβάλλοντος και επιτρέπουν στο χρήστη να επιλέξει από μια γκάμα φυσικών και λογικών context ανάλογα με τα τρέχοντα ενδιαφέροντά του ή τις δραστηριότητές του (Ryan). Μια δεύτερη τις ορίζει ως τις εφαρμογές που- αντίθετα με την προηγούμενη προσέγγιση- αυτόματα παρέχουν πληροφορίες ή/και λαμβάνουν δράση ανάλογα με το τρέχον context του χρήστη (Brown). Τέλος άλλοι βρίσκονται ανάμεσα στις δύο παραπάνω προσεγγίσεις και τις ορίζουν ως τις εφαρμογές εκείνες που παρακολουθούν το περιβάλλον και προσαρμόζονται αυτόματα αλλά σύμφωνα με προ-ορισμένες κατευθυντήριες γραμμές είτε του σχεδιαστή είτε του χρήστη (Fickas et al).

Ωστόσο, με μια πιο προσεκτική ματιά καταλήγουμε στο συμπέρασμα ότι εφόσον επιθυμούμε την εξαγωγή ενός ενιαίου ορισμού για τις context-aware εφαρμογές, καμία από τις δύο προαναφερθείσες κατηγορίες δεν μας καλύπτει. Και αυτό διότι και οι δύο προηγούμενοι ορισμοί είναι τόσο εξειδικευμένοι ώστε σε περίπτωση χρησιμοποίησης τους αποκλείονται από το χαρακτηρισμό ως context-aware εφαρμογές, που εκ των πραγμάτων ανήκουν στην κατηγορία αυτή. Για παράδειγμα έστω ότι έχουμε μια εφαρμογή που απλά παρουσιάζει το context του περιβάλλοντος ενός χρήστη, χωρίς να προσαρμόζεται σε αυτό δηλαδή να αλλάζει τη συμπεριφορά της. Τότε σύμφωνα με την δεύτερη κατηγορία ορισμών αυτή η εφαρμογή δεν ανήκει στις context-aware κάτι προφανώς μη επιθυμητό. Κατά συνέπεια προκύπτει η ανάγκη ενός γενικότερου ορισμού που θα ενοποιεί κατά κάποιον τρόπο τους δύο προαναφερθέντες. Αυτός στον οποίο έχουμε καταλήξει κατά κύριο λόγο είναι ο εξής:

Ένα σύστημα χαρακτηρίζεται ως context-aware αν χρησιμοποιεί το context για να παρέχει σχετικές πληροφορίες ή/και υπηρεσίες στο χρήστη, όπου η σχετικότητα εξαρτάται από τη δραστηριότητα του χρήστη .[DA00]

2.1.2.1 Κατηγοριοποίηση των context-aware εφαρμογών

Σε μια προσπάθεια να ορίσουμε περαιτέρω το πεδίο των context-aware εφαρμογών θα παρουσιάσουμε τις προσπάθειες κατηγοριοποίησης των χαρακτηριστικών τους και στο τέλος θα παρουσιάσουμε μια πρόταση που συνδυάζει τις ομοιότητες τους αλλά λαμβάνει υπ' όψιν και τις κύριες διαφορές τους. Δύο ήταν οι κύριες τάσεις στον τομέα αυτό όπως περιγράφονται στο [DA00].

Η πρώτη έχει 2 ορθογώνιες διαστάσεις: η πρώτη είναι αν ο σκοπός είναι η άντληση πληροφοριών ή η εκτέλεση εντολών και η δεύτερη αν το έργο εκτελείται αυτόματα ή χειροκίνητα. Έτσι καταλήγουμε σε 4 συνδυασμούς, δηλαδή σε 4 χαρακτηριστικά των εφαρμογών:

- εφαρμογές που αντλούν πληροφορίες για το χρήστη με βάση το διαθέσιμο context χειροκίνητα χαρακτηρίζονται ως κοντινής επιλογής (proximate selection applications)
- εφαρμογές που αντλούν πληροφορίες για το χρήση με βάση το διαθέσιμο context αυτόματα χαρακτηρίζονται ως αυτόματης επαναρύθμισης (automatic contextual reconfiguration)
- εφαρμογές που εκτελούν εντολές για το χρήστη με βάση το διαθέσιμο context χειροκίνητα χαρακτηρίζονται ως εντεταλμένες με βάση το context(contextual command applications)
- εφαρμογές που εκτελούν εντολές για το χρήστη με βάση το διαθέσιμο context αυτόματα χαρακτηρίζονται ως σκανδαλισμένες από το context (context-triggered actions)

Η δεύτερη τάση προτείνει μια ταξινόμηση που αποσκοπεί στην αναγνώριση των ενδότερων χαρακτηριστικών των εφαρμογών της κατηγορίας context-aware. Είναι γεγονός ότι ανάμεσα στην προσέγγιση αυτή και στην προηγούμενη υπάρχει μια αισθητή επικάλυψη, ωστόσο υπάρχουν και ουσιώδεις διαφορές. Στην προσέγγιση αυτή το πρώτο χαρακτηριστικό προς εξέταση είναι η «αίσθηση του context» (contextual sensing) δηλαδή η ικανότητα για ανίχνευση και παρουσίαση πληροφορίας context στο χρήστη αυξάνοντας το σύστημα αισθητήρων του. Αυτό το χαρακτηριστικό είναι παρόμοιο με το προαναφερθέν χαρακτηριστικό της κοντινής επιλογής (proximate selection) με τη διαφορά ότι στην

περίπτωση αυτή ο χρήστης δεν καλείται απαραίτητα να επιλέξει ένα από τα αντικείμενα context για περισσότερες πληροφορίες. Το επόμενο χαρακτηριστικό της ταξινόμησης αυτής είναι η προσαρμοστικότητα με βάση το context (context adaptation) δηλαδή η ικανότητα μιας εφαρμογής να εκτελείται ή να διαφοροποιείται αυτόματα με βάση το τρέχον context. Αυτό το χαρακτηριστικό ταυτίζεται με εκείνο της προηγούμενης ταξινόμησης σχετικά με τις εφαρμογές που είναι σκανδαλισμένες από το context(context-triggered actions). Το τρίτο χαρακτηριστικό είναι η ανακάλυψη πόρων context (context resource discovery) που επιτρέπει στις εφαρμογές να εντοπίζουν και να εκμεταλλεύονται πηγές και πληροφορίες σχετικές με το context του χρήστη που μας οδηγεί στο χαρακτηριστικό της αυτόματης επαναρύθμισης με βάση το context (automatic contextual reconfiguration) της προηγούμενης ταξινόμησης. Τέλος το τέταρτο χαρακτηριστικό είναι η επαύξηση του context (contextual augmentation) δηλαδή η ικανότητα συσχέτισης ψηφιακού υλικού με το context του χρήστη.

Κάνοντας μια κριτική των δύο προαναφερθεισών προσεγγίσεων ως ομοιότητες μπορούμε να πούμε ότι καταγράφουν α)την ικανότητα εκμετάλλευσης πόρων σχετικών με το context του χρήστη, β) την ικανότητα εκτέλεσης μιας εντολής αυτόματα μα βάση το τρέχον context και γ) την ικανότητα απεικόνισης σχετικής πληροφορίας στο χρήστη. Όσον αφορά τις διαφορές τους μπορούμε να επισημάνουμε ότι η δεύτερη προσέγγιση για την απεικόνιση σχετικής πληροφορίας στο χρήστη περιλαμβάνει και την εμφάνιση του context και όχι μόνο πληροφορίας με βάση την οποία θα πρέπει να γίνει κάποια επιλογή ενώ επίσης περιλαμβάνει μια κατηγορία που δεν υπάρχει στην $1^{η}$ σχετικά με την επαύξηση του context (context augmentation). Επιπροσθέτως, η $1^{η}$ ταξινόμηση υποστηρίζει την κατηγορία των εφαρμογών που εκτελούν εντολές για το χρήστη με βάση το διαθέσιμο context χειροκίνητα (contextual command applications) σε αντίθεση με τη $2^{η}$.

Με βάση τα παραπάνω καταλήγουμε σε μια προτεινόμενη κατηγοριοποίηση (Dey,Abowd) των χαρακτηριστικών των context-aware εφαρμογών η οποία λαμβάνει υπ' όψιν της τις ομοιότητες αλλά και τις 3 κυριότερες διαφορές που αναφέρθηκαν παραπάνω. Οι κατηγορίες εδώ είναι οι εξής:

- παρουσίαση της πληροφορίας και των υπηρεσιών στο χρήστη
- αυτόματη εκτέλεση μιας υπηρεσίας
- τοποθέτηση ταμπελών(tagging) στο context για περαιτέρω χρήση στο μέλλον

Όπως γίνεται αντιληπτό η 1η κατηγορία, η παρουσίαση, περιλαμβάνει τα χαρακτηριστικά proximate selection(κοντινής επιλογής) και contextual commands(εφαρμογές προσταζόμενες με βάση το context) της 1ης προσέγγισης ενώ έχει προστεθεί η έννοια της παρουσίασης του context της 2ης. Η αυτόματη εκτέλεση είναι παρόμοια με τις context-triggered actions (σκανδαλισμένες από το context) της 1ης και την contextual adaptation (προσαρμοστικότητα

με βάση το context) της 2ης. Τέλος η τοποθέτηση ταμπελών είναι ίδια με την contextual augmentation (επαύξηση του context) της 2ης.

Στο σημείο πρέπει να παρατηρήσουμε δύο πράγματα. Καταρχήν δεν υπάρχει διαφοροποίηση στις κατηγορίες ανάμεσα σε πληροφορίες και υπηρεσίες. Ο λόγος για τον οποίο συνέβη αυτό είναι ο εξής: στις περισσότερες περιπτώσεις είναι δύσκολο να γίνει ξεκάθαρος διαχωρισμός ανάμεσα στην παρουσίαση μιας πληροφορίας και στην παρουσίαση μιας υπηρεσίας. Και αυτό διότι το σκεπτικό του χρήστη μπορεί να αλλάζει από περίπτωση σε περίπτωση και συνεπώς να μεταβάλλεται και το είδος της παρουσίασης. Για παράδειγμα η παρουσίαση της λίστας των εκτυπωτών που δίνεται στο χρήστη μπορεί να είναι είτε μια πληροφορία, αν ο χρήστης θέλει απλώς να δει ποιους εκτυπωτές έχει διαθέσιμους, είτε μια υπηρεσία αν ο χρήστης επιλέξει έναν εκτυπωτή για να εκτυπώσει.

Το δεύτερο σημείο το οποίο πρέπει να παρατηρήσουμε είναι ότι έχει αφαιρεθεί η έννοια της ανακάλυψης-εκμετάλλευσης των πόρων context που υπήρχε και στις δυο προσεγγίσεις (automatic contextual reconfiguration, contextual resource discovery). Αυτό συνέβη διότι το χαρακτηριστικό αυτό δεν θεωρείται ότι ανήκει σε κάποια ξεχωριστή κατηγορία αλλά εντάσσεται στις 2 πρώτες που δίνονται παραπάνω (παρουσίαση,εκτέλεση). Διότι η ανακάλυψη πόρων είναι η ικανότητα εντοπισμού νέων υπηρεσιών με βάση το context και αυτή η ικανότητα δεν διαφέρει σε τίποτα από την επιλογή υπηρεσιών με βάση το context.

2.2 Σχετικές εργασίες για τα context-aware συστήματα

Στην ενότητα αυτή περιγράφονται εν συντομία τα μοντέλα που έχουν προταθεί από την ερευνητική κοινότητα σχετικά με τη διαχείριση των δεδομένων σε context-aware συστήματα.

2.2.1 Ένα μοντέλο που βασίζεται σε πολυδιάστατα ημιδομημένα δεδομένα (multidimensions semistructured data)

Το συγκεκριμένο μοντέλο([Sta03] εισάγει την έννοια του context για ημιδομημένα δεδομένα. Τα πολυδιάστατα δεδομένα είναι ένα μοντέλο θεώρησης δεδομένων, με τη βοήθεια του οποίου είναι δυνατή η οργάνωση και διαχείριση ημιδομημένων δεδομένων με έναν εμπλουτισμένο τρόπο. Τα πολυδιάστατα ημιδομημένα δεδομένα αποτελούν ημιδομημένα δεδομένα με τη διαφορά ότι παρουσιάζουν διαφορετικές όψεις κάτω από διαφορετικά περιβάλλοντα στα οποία μπορεί να βρεθούν.

Τα περιβάλλοντα αυτά ονομάζονται κόσμοι (worlds) και για τον ορισμό τους χρησιμοποιούνται ένα σύνολο παραμέτρων που ονομάζονται διαστάσεις (dimensions). Έτσι ένας κόσμος ορίζεται μέσω της απόδοσης τιμών στις διαστάσεις του. Το ζεύγος διάσταση-

τιμή αποτελεί ουσιαστικά την περιγραφή μιας συνθήκης που ικανοποιείται στα πλαίσια του κόσμου αυτού.

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

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

Η έκφραση αυτή ονομάζεται "context specifier" και μπορεί να ορίζει όχι μόνο μεμονωμένους κόσμους αλλά και σύνολα αυτών. Δηλαδή με το παράδειγμα `[lang=gr, format=pdf]` αναφερόμαστε στους κόσμους όπου η γλώσσα είναι τα ελληνικά, η μορφή του αρχείου είναι pdf και η λεπτομέρεια μπορεί να είναι οτιδήποτε.

Με αυτό τον τρόπο είναι δυνατό να έχουμε διαφορετικά στιγμιότυπα της ίδιας πληροφορίας καθένα εκ των οποίων ισχύει κάτω από ένα διαφορετικό σύνολο κόσμων. Μια τέτοια πληροφοριακή οντότητα που περικλείει έναν αριθμό από στιγμιότυπα (όψεις) λέγεται Πολυδιάστατη Οντότητα. Εάν οι όψεις e1,e2,... μιας πολυδιάστατης οντότητας ισχύουν κάτω από ένα κόσμο w τότε λέμε ότι η e αποτιμάται σε e1,e2,... κάτω από τον w.

Το μοντέλο MOEM

Το μοντέλο MOEM είναι μια επέκταση του OEM [SG01] η οποία διατηρεί την ευελιξία του OEM αλλά παράλληλα εισάγει σε αυτό την έννοια των κόσμων κάτω από τους οποίους ισχύουν τα δεδομένα που είναι αποθηκευμένα στο γραφο. Δύο είναι τα βασικά στοιχεία που υλοποιούν την επέκταση αυτή:

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

Το νέο μοντέλο που περιγράφουμε ονομάζεται Multidimensional Object Exchange Model (MOEM). Στο MOEM οι συμβατικοί κυκλικοί κόμβοι ονομάζονται context κόμβοι και

αναπαριστούν εναλλακτικές όψεις σχετιζόμενες με κάποιο context. Οι συμβατικές ακμές του OEM ονομάζονται entity ακμές. Όπως και στο OEM, στο MOEM όλοι οι κόμβοι θεωρούνται αντικείμενα και χαρακτηρίζονται από ένα μοναδικό αναγνωριστικό (object identifier). Όσον αφορά τα context αντικείμενα ισχύουν όσα αναφέρθηκαν για τους OEM γράφους.

2.2.2 Ένα context-aware μοντέλο διαχείρισης δεδομένων για την Περιβάλλουσα Νοημοσύνη (Ambient Intelligence)

Στο μοντέλο αυτό, που παρουσιάζεται στο [FAJ04] προτείνεται ένα σύστημα για context-aware data management, το οποίο αποτελεί ουσιαστικά ένα middleware το οποίο δεν ενσωματώνεται στο ΣΔΒΔ. Αρχικά προτείνονται οι στρατηγικές χειρισμού για το context, στη συνέχεια γίνεται αναφορά στην υποδομή για τη διαχείριση του και τέλος αναφέρεται η διαδικασία επεξεργασίας των ερωτημάτων.

Πιο συγκεκριμένα, όσον αφορά τις στρατηγικές διαχείρισης του context έχουμε τις ακόλουθες:

- Context ως οι επί-τόπου συνθήκες του ερωτήματος (Context as Present On-the-Spot Query Condition): Τα πλήρως δυναμικά περιβάλλοντα παρακινούν τους χρήστες να θέτουν ερωτήματα ανά πάσα στιγμή και ανεξάρτητα από την τοποθεσία στην οποία βρίσκονται. Τέτοια ερωτήματα συνήθως περιλαμβάνουν το τρέχον context όπως ο χρόνος, ο τόπος, η κίνηση κλπ ως σημείο αναφοράς του ερωτήματος.
- Context ως εργαλείο ανάκλησης με βάση τη παρελθοντική κατάσταση κατά την οποία τέθηκε ένα ερώτημα (Context as Present On-the-Spot Query Condition): Είναι σύνηθες για κάποιο χρήστη να του είναι ευκολότερο να θυμηθεί το context υπό το οποίο είχε πρόσβαση σε κάποια δεδομένα παρά αναλυτικά τα δεδομένα αυτά κάθε αυτά. Για παράδειγμα ένας χρήστης μπορεί να δυσκολεύεται να θυμηθεί τον τίτλο της εφημερίδας που διάβασε, αλλά να θυμάται το context υπό το οποίο έκανε την πράξη αυτή, δηλαδή τους ανθρώπους που ήταν γύρω του, το μέρος στο οποίο διάβασε την εφημερίδα κλπ. Μάλιστα, αυτή η παρατήρηση, ότι δηλαδή το context μπορεί να χρησιμεύσει ως εργαλείο ανάκλησης πληροφοριών έχει και την "ψυχολογική" σκοπιά του, καθώς ερευνητές της ψυχολογίας έχουν αναπτύξει σχετικές θεωρίες που αφορούν τη μνήμη.
- Context ως περιοριστική συνθήκη στα ερωτήματα (Context as Query Constraint): Στην περίπτωση αυτή διακρίνουμε τις εξής υποπεριπτώσεις:
    1. Κατανόηση της πραγματικής πρόθεσης του χρήστη (Understanding user's real query intention): Όταν ο χρήστης θέτει ένα ερώτημα έχει κάποιο απώτερο σκοπό στο μυαλό του, ο οποίος πρέπει να λαμβάνεται υπ' όψιν

για την απάντηση του ερωτήματός του. Π.χ ο χρήστης στο ερώτημα του σχετικά με τα εστιατόρια μιας περιοχής στα οποία μπορεί να πάει με φίλους να γευματίσει, έχει νόημα να λάβει ως απάντηση μόνο εκείνα που είναι ανοικτά.

2. Εξατομίκευση των δεδομένων του χρήστη (Personalizing user's data request): Η χρησιμότητα των δεδομένων συχνά υπόκειται στο context. Για παράδειγμα στο ερώτημα κάποιου για την εύρεση της διαδρομής προς κάποιο προορισμό, δεν θα ήταν επιθυμητή μια απάντηση που θα περιλαμβάνει μια διαδρομή εν μέσω σκοτεινού δάσους αν η ώρα είναι 12 το βράδυ, ούτε εν μέσω των πιο πολυσύχναστων οδικών αρτηριών αν η ώρα είναι 12 το μεσημέρι.
3. Συντονισμός του περιεχομένου στο επίπεδο του ερωτήματος (Tuning the level of query content): Το επιθυμητό επίπεδο αφαίρεσης των προς απάντηση δεδομένων εξαρτάται από το context. Δηλαδή, ένας χρήστης που έχει κοντά του μια μεγάλη οθόνη θα επιθυμούσε ως απάντηση μια HD εικόνα, ενώ κάποιος με ένα κινητό παλιάς τεχνολογίας θα επιθυμούσε απλό κείμενο.

- Context ως κριτήριο για την καταμέτρηση των αποτελεσμάτων του ερωτήματος (Context as Criteria for Query Result Measurement): Αν λάβουμε υπ' όψιν μας τους περιορισμούς που μπορούν να τεθούν από ορισμένες μικρές ή παλαιάς τεχνολογίας συσκευές θα ήταν επιθυμητό τα αποτελέσματα του ερωτήματος του χρήστη να είναι με τέτοιο τρόπο ταξινομημένα ώστε να έρχονται πρώτα εκείνα που τον ενδιαφέρουν και να χάσει εκείνα που ενδεχομένως να τον ενδιαφέρουν λιγότερο. Π.χ στην ερώτηση του χρήστη σχετικά με εστιατόρια, θα ήταν επιθυμητό στα αποτελέσματα να σταλούν πρώτα εκείνα που σερβίρουν το φαγητό της προτίμησης του και στη συνέχεια τα υπόλοιπα.
- Context ως οδηγός στην μεταβίβαση του αποτελέσματος του ερωτήματος (Context as Guide to Query Result Delivery): Η μορφή του αποτελέσματος θα πρέπει επίσης να προσαρμόζεται στο context του χρήστη. Δηλαδή, αν ο χρήστης οδηγεί να ήταν βολικό να λάβει την απάντηση σε μορφή ομιλίας, ενώ αν βρίσκεται σε μια συζήτηση σε κλειστό χώρο, σε μορφή δόνησης στο κινητό του.

Όσον αφορά την υποδομή για τη διαχείριση του context το σύστημα περιλαμβάνει ένα Δημόσιο και έναν Ιδιωτικό Διαχειριστή Δεδομένων. Ο Δημόσιος μπορεί να είναι ένας οποιοσδήποτε συμβατικός διαχειριστής μιας βάσης δεδομένων ο οποίος υποστηρίζει το χρήστη στην πρόσβαση σε δημόσιες βάσεις δεδομένων εφόσον έχει τα κατάλληλα δικαιώματα. Σημαντικότερο ρόλο όμως στο σύστημα επιτελεί ο «Ιδιωτικός Διαχειριστής

Δεδομένων» ο οποίος λειτουργεί ως ένας προσωπικός βοηθός που έχει ως ρόλο να ικανοποιεί τις πληροφοριακές ανάγκες του χρήστη. Τα βασικά συστατικά του είναι τα ακόλουθα:

- Ο διαχειριστής context που ορίζει και συντηρεί το χώρο του context.
- Ο διαχειριστής προφίλ που είναι υπεύθυνος για τη διαχείριση και παροχή πληροφορίας σχετικά με το προφίλ του χρήστη
- Ο διαχειριστής ατζέντας που διαχειρίζεται την καθημερινή ατζέντα του χρήστη.
  Ο διαχειριστής ημερολογίου που διαχειρίζεται και αποθηκεύει το ιστορικό των προσπελάσεων στη βάση δεδομένων του χρήστη.
- Η υπηρεσία επικοινωνίας βάσης δεδομένων που συνδέει τον Ιδιωτικό Διαχειριστή Δεδομένων του χρήστη με τους εξωτερικούς δημόσιους διαχειριστές δεδομένων.
- Η πολυσχηματική διεπαφή σε εξωτερικές υπηρεσίες που προσφέρει είσοδο και έξοδο και προσαρμόζεται στο context.

Η context-aware μονάδα συντονισμού των ερωτημάτων που αναλαμβάνει την εκτέλεση των ερωτημάτων με βάση κάποια στρατηγική.

Όσον αφορά τη διαδικασία επεξεργασίας των ερωτημάτων στο σύστημα αυτό, αυτή χωρίζεται σε τρία στάδια. Το πρώτο αφορά την προ-επεξεργασία του ερωτήματος, το δεύτερο την εκτέλεση του και το τρίτο τη μετα-επεξεργασία του. Όσα ερωτήματα δεν εκτελούνται σε κάποιο δημόσιο διαχειριστή δεδομένων εκτελούνται στον Ιδιωτικό μέσω του context-aware query coordinator ενώ μετά την εκτέλεση τους, στη φάση της μετα-επεξεργασίας τους το αποτέλεσμα αναδιαμορφώνεται με βάση το context του χρήστη.

2.2.3 Ένα context-aware σύστημα βάσεων δεδομένων που βασίζεται στις προτιμήσεις (Preferences)

Το μοντέλο που αναφέρεται στην ενότητα αυτή περιγράφεται στο [SPV05]. Όπως αναφέρθηκε στα προηγούμενα, ένα context-aware σύστημα είναι ένα σύστημα το οποίο χρησιμοποιεί το context για να παρέξει σχετικές υπηρεσίες και πληροφορίες στους χρήστες του. Μέχρι σήμερα, η διαχείριση της πληροφορίας που εξαρτάται από το context γίνεται από κάθε εφαρμογή ξεχωριστά καθώς η έννοια του δεν έχει ενσωματωθεί σε κάποιο σύστημα διαχείρισης βάσεων δεδομένων. Στην κατεύθυνση αυτή, θα περιγραφεί ένα context-aware σύστημα βάσεων δεδομένων που βασίζεται στην έννοια των preferences (προτιμήσεων) και το οποίο είναι σε θέση να απαντά context-aware ερωτήματα, δηλαδή ερωτήματα η απάντηση των οποίων εξαρτάται κάθε φορά από το τρέχον context τη στιγμή της υποβολής τους. Στην κατεύθυνση περιγραφής του παραπάνω συστήματος θα πρέπει αρχικά να μοντελοποιήσουμε τις έννοιες του context και των preferences των χρηστών.

Λογικό Μοντέλο

Στα πλαίσια αυτής της μοντελοποίησης αρχικά θα περιγραφούν οι θεμελιώδεις έννοιες που αφορούν το μοντέλο του context που επιλέχθηκε και δεν είναι άλλο από το λογικό. Σύμφωνα με το μοντέλο αυτό λοιπόν έχουμε τις παρακάτω έννοιες:

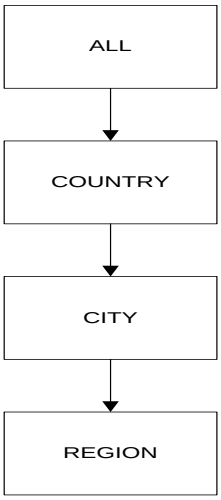
Πεδία ορισμού (domains): Το πεδίο ορισμού είναι μετρήσιμο σύνολο τιμών για το οποίο ορίζονται οι ιδιότητες των σχέσεων. Περιλαμβάνει μια ειδική τιμή NULL η οποία αντιστοιχεί στην έλλειψη γνώσης για την τιμή της συγκεκριμένης ιδιότητας.

Ιδιότητες και Σχέσεις (Attributes and Relations): Κάθε ιδιότητα Ai χαρακτηρίζεται από ένα όνομα (αριθμήσιμο σύνολο) και ένα πεδίο ορισμού (dom(Ai)). Ένα σχήμα σχέσης είναι ένα πεπερασμένο σύνολο από ιδιότητες και ένα αποτύπωμα (instance) της σχέσης αυτής είναι το υποσύνολο του καρτεσιανού γινομένου όλων των domains των ιδιοτήτων του σχήματος της σχέσης.

Παράμετροι context (Context Parameters): Το context μοντελοποιείται μέσω ενός αριθμήσιμου συνόλου ιδιοτήτων ειδικού-σκοπού (special purpose) που καλούνται παράμετροι context. Αυτές ιδιότητες διαχωρίζονται σε δύο κατηγορίες: α)στις στατικές οι οποίες παίρνουν μια τιμή από το domain τους και β) στις δυναμικές οι οποίες παίρνουν την τιμή τους από την εφαρμογή μιας συνάρτησης (φυσικά πάλι από το domain τους). Ως περιβάλλον context (Cx) ορίζουμε ένα σύνολο από παραμέτρους context {c1,c2,…,cn}.

Κατάσταση context (Context State): Κατάσταση context ορίζεται μια ανάθεση τιμών στις παραμέτρους context. Έστω μια χρονική στιγμή t. Τότε η κατάσταση context τη χρονική στιγμή αυτή είναι CSX(t) = {c1(t), c2(t), . . . cn(t)}, όπου ci(t) είναι η τιμή της παραμέτρου context ci τη χρονική στιγμή t.

Ιεραρχίες Ιδιοτήτων (Hierarchies for Attributes): Είναι πιθανό μια ιδιότητα να συμμετέχει σε μια ιεραρχία επιπέδων, δηλαδή να μπορεί να παρατηρηθεί από διαφορετικά επίπεδα λεπτομέρειας. Στην περίπτωση αυτή ορίζουμε μια ιεραρχία ιδιοτήτων ως μια δομή επιπέδων στην οποία το υψηλότερο επίπεδο είναι πάντα το επίπεδο ALL ώστε να μπορεί να γίνει ομαδοποίηση όλων των τιμών των επιμέρους επιπέδων σε μια μόνο τιμή. Στο κατώτατο όριο της ιεραρχίας υπάρχει το αναλυτικότερο επίπεδο περιγραφής της παραμέτρου. Η συσχέτιση μεταξύ των επιπέδων γίνεται με τη χρήση της συνάρτησης ancL1L2 η οποία αναθέτει μια τιμή του domain L2 σε μια τιμή του domain L1. Για παράδειγμα ancRegionCity(Acropolis)=Athens δηλαδή ανατέθηκε στην πόλη της Αθήνας η περιοχή της Ακρόπολης (στην ιεραρχία των επιπέδων). Παρακάτω δίνεται ενδεικτικά η μορφή μιας ιεραρχίας επιπέδων.



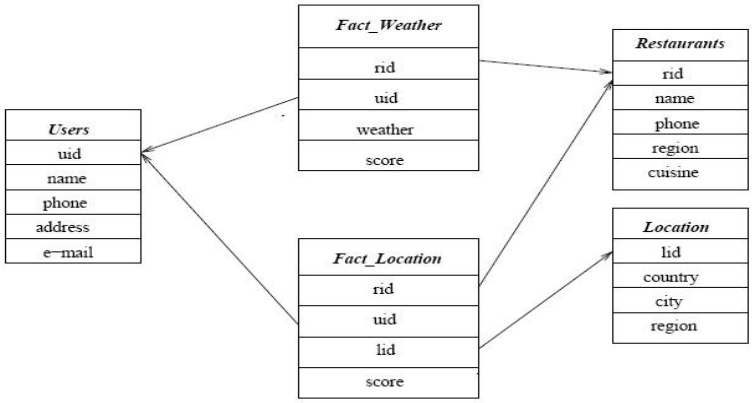
Περί το context preferences

Στο σημείο αυτό θα περιγραφούν συνοπτικά οι contextual preferences.Αρχικά θα πρέπει να τις χωρίσουμε σε δύο κατηγορίες:

- Βασικές(Basic) preferences: Οι βασικές preferences περιγράφονται από μια παράμετρο context, ένα σύνολο από μη-context παραμέτρους και ένα βαθμό ενδιαφέροντος ο οποίος ορίζεται ως ένας πραγματικός αριθμός μεταξύ 0 και 1. Η τιμή 1 εκφράζει ακραίο ενδιαφέρον ενώ η τιμή 0 καθόλου ενδιαφέρον για τη συγκεκριμένη preference. Συνεπώς μια βασική preference θα είναι της μορφής: preferencebasici (ci,Ak+1, . . . ,An) = interest scorei
  όπου ci είναι η παράμετρος context και Ai οι μη-context παράμετροι
- Συναθροιστικές (Aggregate) Preferences: Οι συναθροιστικές preferences προκύπτουν από κάποιο συνδυασμό βασικών και εκφράζονται από ένα σύνολο παραμέτρων context, ένα σύνολο μη-context παραμέτρων και ένα βαθμό ενδιαφέροντος. Η μορφή τους είναι η εξής:
  preference (c1, . . . ck,Ak+1, . . . ,An) = interest score
  Ο βαθμός ενδιαφέροντος προκύπτει από την εφαρμογή κάποιας συνάρτησης. Στα πλαίσια του μοντέλου που περιγράφεται, θα υποτεθεί ότι η συνάρτηση αυτή υπολογίζει το σταθμισμένο μέσο όρο των βαθμών ενδιαφέροντος των βασικών preferences από τις οποίες αποτελείται η συναθροιστική. Δηλαδή, ο βαθμός ενδιαφέροντος θα είναι της μορφής:
  interest score = w1 × interest score1 + . . . + wk × interest scorek

Αντιμετώπιση ερωτημάτων

Η παράγραφος αυτή ασχολείται με την ταξινόμηση των ερωτημάτων που μπορούν να τεθούν στο σύστημα που περιγράφεται. Αρχικά η απλούστερη περίπτωση περιλαμβάνει ένα ερώτημα για τον υπολογισμό του οποίου δεν είναι απαραίτητος ο υπολογισμός μιας συναθροιστικής preference. Στην περίπτωση αυτή, οι χρήστες δηλώνουν ρητά ότι δεν ενδιαφέρονται για συνδυασμούς παραμέτρων context και κατά συνέπεια τα ερωτήματα αυτά αντιμετωπίζονται ως απλά ερωτήματα μιας βάσης δεδομένων στα οποία η μοναδική παράμετρος context που ενδεχομένως να προσδιορίζεται χρησιμοποιείται ως μια ιδιότητα όπως όλες οι άλλες. Για παράδειγμα έστω ότι έχουμε το παρακάτω σχήμα βάσης:



και τίθεται το ερώτημα:
Look for Mary's most preferable restaurants near Acropolis, independently of the status of weather.

Τότε αυτό μεταφράζεται σε γλώσσα sql στο:

```
SELECT R.name, FL.score
FROM Users U, Restaurants R, Fact Location FL,
Location L
WHERE U.uid = FL.uid AND R.rid = FL.rid
AND L.lid = FL.lid AND U.name =0Mary0 AND
L.location ='Acropolis'
ORDER BY FL.score DESC;
```

Ενδιαφέρον παρουσιάζει η περίπτωση όπου κατά την οποία η επεξεργασία ενός ερωτήματος οδηγεί στον υπολογισμό του βαθμού μιας συναθροιστικής preference. Για παράδειγμα και πάλι με βάση το παραπάνω σχήμα βάσης έστω ότι έχουμε το ερώτημα:

Look for Mary's most preferable restaurants (in the current context).

Τότε η εκτέλεση του ερωτήματος αυτού οδηγεί στην εκτέλεση των παρακάτω υποερωτημάτων (ως context παράμετροι θεωρούνται οι: location και weather οι οποίες έχουν τις τιμές Acropolis και sunny αντίστοιχα):

```
SELECT R.name, FL.score
FROM Users U, Restaurants R, Fact Location FL,
Location L
WHERE U.name =0Mary0 AND U.uid = FL.uid
AND R.rid = FL.rid AND L.lid = FL.lid AND
current location ='Acropolis';

SELECT R.name, FW.score
FROM Users U, Restaurants R, Fact Weather FW
WHERE U.name =0Mary0 AND U.uid = FW.uid AND
R.rid = FW.rid AND current weather ='sunny';
```

2.2.4 Chameleon: Ένα context-aware μοντέλο για ΣΔΒΔ)

Το μοντέλο που αναφέρεται στην ενότητα αυτή περιγράφεται στο [EAM09] και αποτελεί ένα σύστημα διαχείρισης βάσεων δεδομένων που ενσωματώνει την έννοια του context-aware, ονόματι Chameleon. Το σύστημα αυτό περιορίζει την ανάγκη χρησιμοποίησης χωρικών, χρονικών ή άλλων βάσεων δεδομένων καθώς ενσωματώνει τις έννοιες αυτές ως context σε αντίθεση με τα περισσότερα context-aware συστήματα τα οποία έχουν στηθεί στην κορυφή μιας υποδομής που τους παρέχει δεδομένα. Το σύστημα αυτό, υποστηρίζει πολλαπλά contexts καθώς και preferences των χρηστών. Ακόμα παρέχει μια γενικής χρήσης διαπροσωπεία για τον ορισμό και την επεξεργασία των πληροφοριών context οι οποίες ταξινομούνται με βάση τις ιδιότητές τους.

Ταξινόμηση του context

Δύο είναι οι κύριες οντότητες που λαμβάνουν μέρος στο σύστημα αυτό, ο εκδότης του ερωτήματος (χρήστης) και τα δεδομένα τα οποία αφορά το ερώτημα που θέτει. Με βάση τις οντότητες αυτές το context ταξινομείται σε δυο κατηγορίες:

- Context του αντικειμένου (Object Context): είναι το context των δεδομένων που υπόκεινται στο ερώτημα και το οποίο μπορεί να περιλαμβάνει περισσότερες από μια ιδιότητες των δεδομένων αυτών.
- Context του χρήστη (User Context): αναφέρεται στο context του χρήστη. Μπορεί να είναι η τοποθεσία, η ταυτότητα, οι προτιμήσεις ή οτιδήποτε άλλο μπορεί να περιγράψει κάποια πληροφορία σχετική με το χρήστη. Χωρίζεται επιπλέον σε 3 διαστάσεις:
- Πρόσημο του Context (Context Sign): Το context του χρήστη μπορεί να είναι θετικό ή αρνητικό. Το θετικό ορίζει από τί αντιπροσωπεύει το context και το αρνητικό το αντίθετο (τί δεν αντιπροσωπεύει).
- Σχέση του Context (Contextual Relation): Αφορά τη σχέση ανάμεσα στα δεδομένα του context, δηλαδή το βαθμό συνάφειας μεταξύ των δεδομένων. Οι τιμές της

διάστασης αυτής είναι: σχέση ισοδυναμίας, σχέση ολικής διάταξης και σχέση μερικής διάταξης.

1. Κατάταξη σε λίστα των δεδομένων (Listing of Data): Αφορά τον τρόπο με τον οποίο τα δεδομένα θα πρέπει να κατατάσσονται σε λίστα. Οι τιμές της διάστασης αυτής όσον αφορά τα δεδομένα είναι: απαγορεύεται να μην είναι καταχωρημένα, επιτρέπεται να μην είναι καταχωρημένα.
2. Ως παράδειγμα του context του χρήστη ας θεωρήσουμε την πληροφορία ότι ο χρήστης επιθυμεί κατά τον ίδιο βαθμό να παρακολουθήσει έναν αγώνα μπάσκετ και έναν ποδοσφαίρου. Το context αυτό ορίζεται ως ένα σημείο στον τρισδιάστατο χώρο που ορίστηκε παραπάνω για το οποίο γνωρίζουμε ότι οι διαστάσεις του έχουν τις παρακάτω τιμές: (θετικό, σχέση ισοδυναμίας, απαγορεύεται να μην είναι καταχωρημένα).

Κατασκευές SQL για την ενσωμάτωση της ιδιότητας του context-aware

Παρακάτω δίνονται οι διάφορες SQL κατασκευές που απαιτούνται για την ενεργοποίηση της ιδιότητας του context-aware σε ένα σύστημα διαχείρισης βάσεων δεδομένων, και τις οποίες παρέχει το σύστημα Chameleon.

- Δημιουργία αντικειμένων context (Creating Object Contexts): η εντολή αυτή χρησιμοποιείται για τον ορισμό του context ενός αντικειμένου όταν το context αυτό δεν ανήκει σε κάποια από τις ιδιότητες της σχέσης του αντικειμένου. Η σύνταξη της εντολής είναι της μορφής:

```
CREATE OBJECT CONTEXT contextname (
{ col spec | table constraint } [, . . . ]
 ,table binding
);
```

Όπου:

1. η κατασκευή το col spec αναφέρεται στην προδιαγραφή του αντικειμένου (όνομα, τύπος κλπ),
2. η κατασκευή table constraint αναφέρεται σε κάθε είδους περιορισμό που μπορεί να τεθεί από το context (περιορισμοί ελέγχου)
3. και η κατασκευή table_binding συνδέει το αντικείμενο με το περιεχόμενό του. Η σύνταξη της κατασκευής αυτής είναι η παρακάτω:

```
BINDING KEY ([col name [, . . . ]])
REFERENCES ref table [( ref col [, . . . ] )] WITH bool_expr
```

Πρέπει να σημειώσουμε ότι αν και μοιάζει με τον ορισμό του foreign key, το Binding key δεν έχει αναφορά σε κάποιο πρωτεύον κλειδί ενώ απόφαση για δέσιμο ενός context με κάποιο αντικείμενο δε στηρίζεται απαραίτητα στην ισότητα με μια τιμή του πίνακα στον οποίο δείχνει η αναφορά καθώς με τη χρήση του with, η λογική έκφραση που το ακολουθεί λειτουργεί ως σύνδεσμος.

- Δημιουργία context χρήστη (Creating User Contexts): Με παρόμοιο τρόπο όπως το context των αντικειμένων, έτσι και εκείνο το χρήστη υλοποιείται σε μια σχέση. Για κάθε πίνακα που επηρεάζεται από το context του χρήστη, ένα κλειδί δεσίματος χρησιμοποιείται για να δείξει πως το context αντανακλά στον πίνακα. Παρακάτω δίνεται η σύνταξη της εντολής SQL:

```
CREATE [context sign] CONTEXT contextname (
fcol spec | table constraintg [, . . . ]
, table binding [, . . . ]
[, substituting key [, . . . ]]
) [
AS contextual relation clause]
[WITH UNLISTED unlisted status];
context sign: positive | negative
contextual relation: equivalence | partial order
| total order [USING ordering func]
unlisted status: excluded | included
```

- Κατασκευή γενικής αντικατάστασης (Global substitution Construct): Ορισμένες ιδιότητες είναι φυσικό να χρειάζονται τροποποίηση όσον αφορά την παρουσίασή τους εφόσον ενσωματωθεί η έννοια του context-aware. Για παράδειγμα, εφόσον ο χρήστης βρίσκεται στην Αμερική, είναι λογικό να επιθυμεί όλες οι πληροφορίες σχετικά με την παρουσίαση της ώρας ,τα ραντεβού του, η λίστα υποχρεώσεών του κλπ να κινούνται γύρω από εκείνη τη ζώνη ώρας. Αυτή η μετατροπή, ονομάζεται γενική αντικατάσταση στο σύστημα Chameleon από τη στιγμή που αφορά όλους τους πίνακες για το τρέχον context, και οι τιμές αλλάζουν σε όλους. Το κλειδί αντικατάστασης (substituting key) ορίζει την εντολή για τη διαδικασία αυτή η οποία ορίζεται κατά τον ορισμό του context του χρήστη. Παρακάτω δίνεται η σύνταξη της εντολής:

  SUBSTITUTE table name(col name) BY expression;

- Θέτοντας ενεργά context (Setting Active Contexts): Ο χρήστης είναι πιθανόν να έχει ορίσει διαφορετικά context, τα οποία δεν του είναι απαραίτητα όλα την ίδια στιγμή. Για το λόγο αυτό του παρέχεται η δυνατότητα ορισμού των ενεργών του contexts

δηλαδή εκείνων που θα λαμβάνονται υπ' όψιν για το χρήστη αυτό. Παρακάτω δίνεται η σύνταξη της εντολής αυτής:

```
SET ACTIVE CONTEXT [FOR USER user name]
AS context name [, . . . ];
f [WITH RANKING ORDER context name [, . . . ] ]
| [WITH RANKING EXPRESSION expression
| [WITH SKYLINE OF expression fMAX|MINg [, . . . ] ] g;
```

Αυτό που πρέπει να επισημανθεί για την παραπάνω εντολή είναι ότι οδηγεί στη σύνταξη σύνθετων context από απλά. Αυτό διότι, αν περισσότερα από ένα contexts επηρεάζουν την διάταξη κάποιων δεδομένων τότε παρέχονται 3 μηχανισμοί επίλυσης αυτής της σύγκρουσης, που οδηγούν στη σύνθετη διάταξή τους:

- Ορίζοντας τη σειρά με την οποία τα διαφορετικά context θα επενεργούν στη διάταξη των αντικειμένων
- Ορίζοντας μια έκφραση κατάταξης που θα χρησιμοποιηθεί στην περίπτωση χρήσης ενός αλγορίθμου κατάταξης
- Ζητώντας το περίγραμμα των αντικειμένων όταν η τάξη διαφορετικών context είναι αντίστροφα

Κριτική του συστήματος Chameleon

Το σύστημα αυτό επιτυγχάνει να παρέχει στο χρήστη τις δυνατότητες εξειδικευμένων συστημάτων (spatial-temporal-Hippocratic databases) για κάποιο συγκεκριμένο context αποφεύγοντας ταυτόχρονα να θέσει περιορισμούς που θέτουν τα συστήματα αυτά. Από τη στιγμή που έννοιες όπως ο χρόνος και ο χώρος μοντελοποιούνται ως context παύει να υπάρχει η ανάγκη δημιουργίας των εξειδικευμένων συστημάτων βάσεων δεδομένων που αναφέρονται παραπάνω, και που σε κάθε περίπτωση κάθε άλλο παρά εύκολο είναι. Επιπρόσθετα, το σύστημα υποστηρίζει και συνδυάζει πολλαπλά contexts σε ένα σύστημα, κάτι το οποίο δεν υποστηρίζεται αλλού ενώ τέλος επιτρέπει και τη δημιουργία σύνθετης μορφής context. (από απλής)

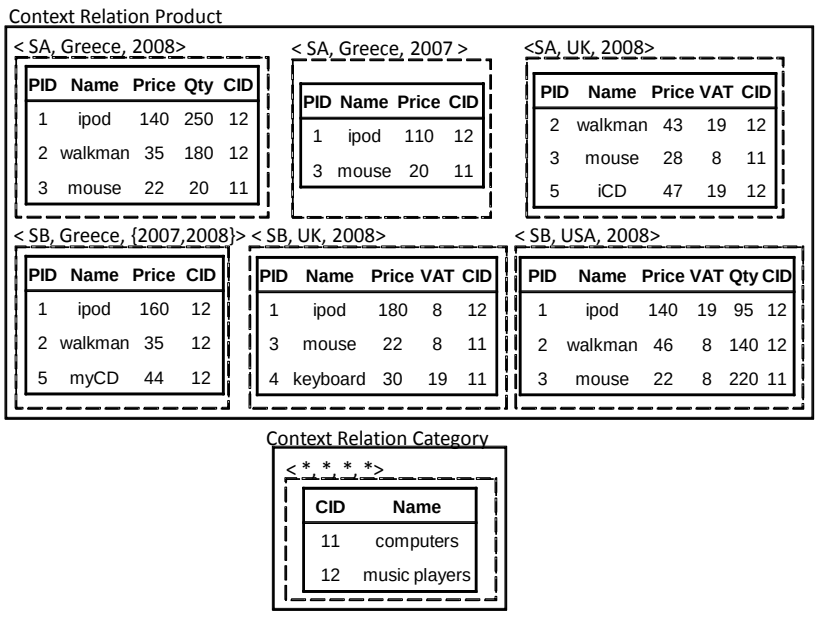
2.3 Ένα μοντέλο για context-aware σχεσιακές βάσεις δεδομένων

Τα μοντέλα που προαναφέρθηκαν στην υποενότητα 2.2 επιχειρούν μια προσέγγιση του θέματος χωρίς ωστόσο να ενσωματώνουν το context στο επίπεδο των δεδομένων. Πιο συγκεκριμένα το μοντέλο της υποενότητας 2.3.1 προτείνει μια προσέγγιση για ημιδομημένα δεδομένα, ενώ εκείνα των 2.3.2 και 2.3.3 μοντελοποιούν το context ως προς τις προτιμήσεις

του χρήστη. Ακόμα, το 2.3.4 ουσιαστικά δρα ως ένα επίπεδο επεξεργασίας του context ανάμεσα στην εφαρμογή και τη βάση δεδομένων. Γίνεται σαφές λοιπόν ότι στα παραπάνω μοντέλα το context δεν ορίζεται ως μια πρωτεύουσα οντότητα (first class citizen) στα συστήματα αυτά. Αυτό το κενό έρχεται να καλύψει το μοντέλο [RS09] που περιγράφεται στην ενότητα αυτή και το οποίο αφορά την ενσωμάτωση των εννοιών του context και του context-aware σε ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων. Στο μοντέλο αυτό άλλωστε –το οποίο παρουσιάζει context aware βάσεις δεδομένων- βασίστηκε και η παρούσα διπλωματική εργασία.

Στο [RS09] καταρχάς προτείνεται μια high-level κατηγοριοποίηση του context η οποία λόγω ακριβώς της υπόστασης του context δεν είναι εξαντλητική. Έτσι αυτό χωρίζεται στο context που αφορά το χρήστη (πχ ηλικία, εθνικότητα, βιομετρικά χαρακτηριστικά), σε αυτό που αφορά το περιβάλλον (πχ τοποθεσία, χρόνος) και σε αυτό που είναι σχετικό με το σύστημα (πχ. δυνατότητες δικτύου, υπολογιστική ισχύ κλπ). Παρακάτω δίνονται μερικοί ορισμοί οι οποίοι είναι απαραίτητοι για την περιγραφή του μοντέλου:

- Το **Context Schema C_s** είναι το σύνολο των attributes που αποτελούν στο σχήμα αυτό. Και από εδώ και στο εξής θα χρησιμοποιείται στο μορφή <attribute1, attribute2,…>. Για παράδειγμα ως context schema μπορούμε να ορίσουμε το <year,location>.
- Το **Context Schema instance** (ή αλλιώς context instance) αποτελεί μια ανάθεση τιμών για ένα συγκεκριμένο Context Schema. Για παράδειγμα για το Context Schema <year,location> οι αναθέσεις τιμών <2008,Greece> και <2005,UK> αποτελούν context instances.
- Ο **Context Specifier C_p** αποτελεί ένα σύνολο από context instances. Για παράδειγμα τα context instances για το παραπάνω context schema, <2008,Greece> και <2005,UK> μπορούν να αποθηκευτούν σε ένα Context Specifier.
- Το **Morph** αποτελεί μια σχέση όπως τη γνωρίζουμε στα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων με τη διαφορά ότι κάθε morph ορίζεται για ένα συγκεκριμένο context instance.
- Το **Polymorph** αποτελεί ένα σύνολο από Morphs. Για κάθε Polymorph ορίζεται το Context Schema με βάση το οποίο ορίζεται και κατά συνέπεια κάθε morph που ανήκει στο polymorph αυτό ορίζεται με βάση κάποιο context instance του context schema αυτού. Για παράδειγμα έστω το Context Schema Product το οποίο διατηρεί πληροφορίες για τα προϊόντα που πωλούνται από μια επιχείρηση. Τότε αν υποθέσουμε ότι έχει οριστεί το context schema <year, location> και σε αυτό έχουν αποδοθεί οι τιμές <2008,Greece> και <2005,UK> μπορούν να οριστούν δύο morphs, ένα για κάθε context instance. Αν τώρα υποθέσουμε ότι στις δύο αυτές χώρες έχουν


Context Relation Product
< SA, Greece, 2008>
PID Name Price Qty CID
1 ipod 140 250 12
2 walkman 35 180 12
3 mouse 22 20 11
< SA, Greece, 2007 >
PID Name Price CID
1 ipod 110 12
3 mouse 20 11
<SA, UK, 2008>
PID Name Price VAT CID
2 walkman 43 19 12
3 mouse 28 8 11
5 iCD 47 19 12
< SB, Greece, {2007,2008}>
PID Name Price CID
1 ipod 160 12
2 walkman 35 12
5 myCD 44 12
< SB, UK, 2008>
PID Name Price VAT CID
1 ipod 180 8 12
3 mouse 22 8 11
4 keyboard 30 19 11
< SB, USA, 2008>
PID Name Price VAT Qty CID
1 ipod 140 19 95 12
2 walkman 46 8 140 12
3 mouse 22 8 220 11
Context Relation Category
< *, *, *, *>
CID Name
11 computers
12 music players

Στο ενδεχόμενο ερωτήματος του χρήστη το σχήμα της βάσης από την οποία θα προκύψει το τελικό αποτέλεσμα αλλάζει. Έτσι σε ενδεχόμενο ερώτημα χρήστη σχετικά με τα διαθέσιμα σε αυτόν προϊόντα ανάλογα με τη χώρα προέλευσής του και την ημερομηνία που τον ενδιαφέρει το ερώτημα θα εκτελεστεί πάνω από το κατάλληλο morph κάθε φορά το οποίο θα έχει οριστεί με βάση το συγκεκριμένο context instance. Διαφορετικά αν θέσει κάποιο ερώτημα με σκοπό την προβολή για κάποιο προϊόν τη τιμής του σε όλες τις χώρες που πωλείται, ώστε για παράδειγμα να κάνει σύγκριση των τιμών αυτών, τότε το σύστημα θα εκτελέσει το ερώτημα αυτό πάνω από όλα τα morphs. Γίνεται συνεπώς προφανές ότι ανάλογα με το οριζόμενο κάθε φορά context, το σχήμα της βάσης πάνω από την οποία θα εκτελεστεί το εκάστοτε ερώτημα μεταβάλλεται.

Όπως ήδη αναφέρθηκε, για την υποστήριξη των νέων αυτών λειτουργιών επεκτάθηκε η σχεσιακή άλγεβρα τόσο με την πρόσθεση νέων τελεστών όσο και με την τροποποίηση των υπαρχόντων. Στο σημείο αυτό να σημειώσουμε ότι ένας τελεστής μπορεί να είναι είτε unary είτε binary. Στην περίπτωση που είναι unary, δέχεται ως είσοδο ένα polymorph και παράγει ένα άλλο ως έξοδο. Διαφορετικά, αν είναι binary δέχεται δύο polymorphs ως είσοδο και παράγει ένα ως έξοδο, ωστόσο πρέπει να τονιστεί ότι για να έχει νόημα η πράξη που θα επιτελέσει ο τελεστής αυτός πρέπει τα δύο polymorphs που δίνονται ως είσοδος να έχουν οριστεί με βάση το ίδιο context schema.

Παρακάτω δίνονται οι σημαντικότεροι εξ' αυτών:

Project: Ο τελεστής αυτός πραγματοποιεί ένα κλασικό project πάνω από κάθε morph του polymorph για το οποίο εκτελείται το ερώτημα. Στο ενδεχόμενο να ζητείται η προβολή κάποιου attribute που δεν υπάρχει σε κάποιο morph, τότε το morph αυτό αυτόματα αποκλείεται από το παραγόμενο αποτέλεσμα.

Select: Ο τελεστής αυτός πραγματοποιεί ένα κλασικό select πάνω από κάθε morph του polymorph για το οποίο εκτελείται το ερώτημα και το οποίο περιλαμβάνει όλα τα απαιτούμενα από το ερώτημα attributes. Αν ζητείται να γίνει select σε κάποιο attribute που δεν υπάρχει σε κάποιο morph, τότε το morph αυτό αυτόματα αποκλείεται από το παραγόμενο αποτέλεσμα.

Cartesian Product: Ο τελεστής αυτός πραγματοποιεί ένα κλασικό cartesian product για κάθε ζευγάρι morphs που ανήκουν στα polymorphs που έχουν δοθεί ως είσοδος, τα οποία ορίζονται με βάση το ίδιο context instance. Στην περίπτωση που κάποιο morph ενός εκ των δύο polymorphs έχει οριστεί με βάση κάποιο context instance για το οποίο δεν έχει αντίστοιχα οριστεί κάποιο άλλο morph το οποίο να ανήκει στο άλλο polymorph, τότε αυτόματα το πρώτο δεν λαμβάνει μέρος στο τελικό αποτέλεσμα.

Union, Intersection, Difference: Για την ένωση, κάθε ζευγάρι morphs που έχει οριστεί με το ίδιο context instance και το οποίο είναι union-compatible (κλασικός ορισμός στο σχεσιακό

μοντέλο) λαμβάνει μέρος στο τελικό αποτέλεσμα. Εδώ να σημειωθεί ότι στην περίπτωση που κάποιο morph ενός εκ των δύο polymorphs έχει οριστεί με βάση κάποιο context instance για το οποίο δεν έχει αντίστοιχα οριστεί κάποιο άλλο morph το οποίο να ανήκει στο άλλο polymorph, αυτό λαμβάνει κανονικά μέρος στο τελικό αποτέλεσμα απλά δεν γίνεται πράξη ένωσης και μεταφέρεται αυτούσιο. Για την τομή, γίνεται η πράξη ανάμεσα σε morphs ορισμένα με βάση το ίδιο context instance και αποκλείονται όσα έχουν context instance για τα οποία δεν έχουν οριστεί morphs και για τα δύο polymorphs (αφού γίνεται τομή με το κενό). Για τη διαφορά, στο τελικό αποτέλεσμα περιλαμβάνονται όσα morphs προκύπτουν από τη διαφορά κάθε φορά των morphs που είναι ορισμένα με βάση το ίδιο context instance συν αυτά που υπάρχουν στο polymorph που βρίσκεται στο αριστερό μέρος της πράξης και τα οποία έχουν οριστεί με βάση context instances για τα οποία δεν υπάρχουν αντίστοιχα ορισμένα morphs στο δεύτερο polymorph.

Select Schema: Ο τελεστής αυτός πραγματοποιεί ένα κλασικό select πάνω από τα δεδομένα του context. Το τελικό αποτέλεσμα περιλαμβάνει εκείνα τα morphs τα οποία έχουν οριστεί με βάση τα context instances που ικανοποιούνται στο select.

Map: Ο τελεστής αυτός είναι υπεύθυνος για την τροποποίηση του context schema ενός polymorph. Μπορεί τόσο να προσθέσει όσο και να αφαιρέσει attributes από το context schema με βάση το οποίο ορίζεται το polymorph. Η πρόσθεση δεν έχει καμία πολυπλοκότητα, ενώ αντίθετα για την αφαίρεση ο τελεστής είναι υπεύθυνος για να κάνει merge όσα morphs πλέον ορίζονται με βάση το ίδιο context instance.

3

Ανάλυση και Σχεδίαση

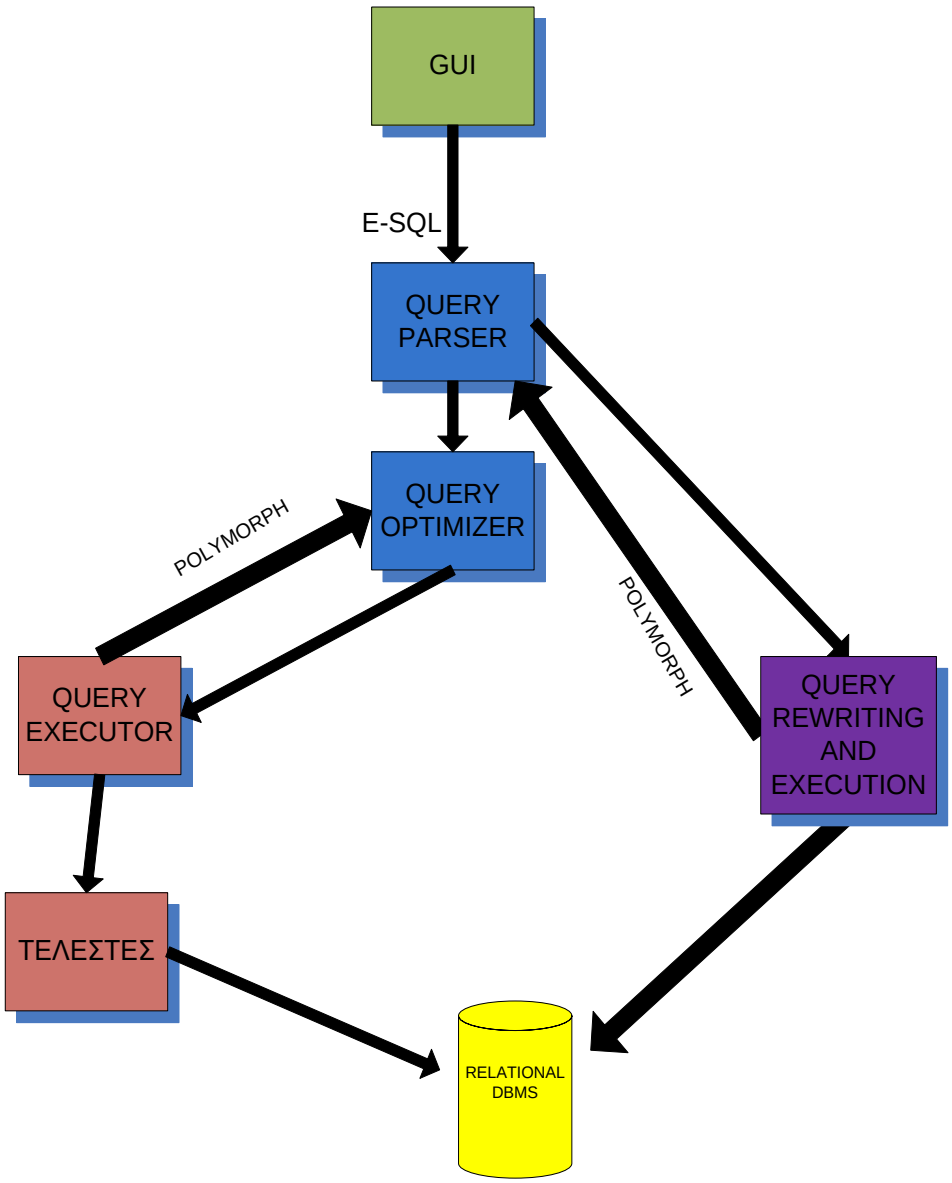
Στο κεφάλαιο αυτό περιγράφεται η γενική ανάλυση και σχεδίαση τόσο του συνολικού συστήματος που υλοποιεί το μοντέλο που περιγράφεται στο κεφάλαιο 2 όσο και του υποσυστήματος μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων. Πιο συγκεκριμένα στην ενότητα 1 δίνεται μια εποπτική εικόνα του συνολικού συστήματος και των επιμέρους υποσυστημάτων που το αποτελούν. Στην ενότητα 2- στην οποία δίνεται και περισσότερη έμφαση μιας και αποτελεί το αντικείμενο της εργασίας αυτής- περιγράφεται αναλυτικά το υποσύστημα μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων μαζί με όλα τα υποσυστήματα του υλοποιήθηκαν. Στην ενότητα 3 αναφέρονται οι διαφορετικές υλοποιήσεις που πραγματοποιήθηκαν όσον αφορά την αποθήκευση των δεδομένων στο σχεσιακό ΣΔΒΔ που χρησιμοποιήθηκε ενώ τέλος στην ενότητα 4 περιγράφονται οι λεπτομέρειες υλοποίησης των επιμέρους υποσυστημάτων.

3.1 Γενική Αρχιτεκτονική του Συστήματος

Όπως γίνεται αντιληπτό παρατηρώντας το διάγραμμα με τη συνολική αρχιτεκτονική (σχήμα 3.1), το σύστημα αποτελείται από τέσσερα επίπεδα. Στο ανώτερο επίπεδο βρίσκεται το υποσύστημα διεπαφής του χρήστη με το σύστημα, ενώ ακολουθεί το υποσύστημα ανάλυσης και εκτέλεσης των ερωτημάτων. Στο επόμενο επίπεδο υπάρχουν δύο ανεξάρτητα υποσυστήματα υπεύθυνα για την εκτέλεση των ερωτημάτων. Το υποσύστημα αποθήκευσης και εκτέλεσης τελεστών και το υποσύστημα μετατροπής των ερωτημάτων σε PL\SQL και εκτέλεσης τους. Τέλος στο κατώτερο επίπεδο βρίσκεται στο σχεσιακό ΣΔΒΔ που χρησιμοποιείται ως physical layer. Η ροή του συστήματος είναι «από πάνω προς τα κάτω» όταν τίθεται ένα ερώτημα και «από κάτω προς τα πάνω» όταν έχει ολοκληρωθεί η εκτέλεση και στέλνονται τα αποτελέσματα.

Θέτοντας ένα ερώτημα μέσω του υποσυστήματος διεπαφής χρήστη, αυτό περνάει από διάφορα στάδια επεξεργασίας μέχρι την τελική απάντησή του. Αρχικά προωθείται στο υποσύστημα ανάλυσης και εκτέλεσης των ερωτημάτων όπου ελέγχεται συντακτικά και

σημασιολογικά από τον Query Parser, ενώ στη συνέχεια ο Query Optimizer είναι υπεύθυνος για τη δημιουργία του βέλτιστου πλάνου εκτέλεσης. Από το υποσύστημα αυτό, υπάρχουν δύο ανεξάρτητα πορείες για την εκτέλεση του. Στην πρώτη το βέλτιστο πλάνο εκτέλεσης προωθείται στο υποσύστημα αποθήκευσης και εκτέλεσης τελεστών όπου εκτελείται μέσω της εκτέλεσης κάθε τελεστή που αυτό περιλαμβάνει. Έτσι ολοκληρώνεται η εκτέλεση και στέλνεται το αποτέλεσμα στη μορφή ενός polymorph στο γραφικό περιβάλλον το οποίο είναι υπεύθυνο για την παρουσίαση του στο χρήστη.



Σχήμα 3.1 Γενική αρχιτεκτονική context-aware συστήματος

Η παρούσα εργασία ασχολείται με το δεύτερο υποσύστημα εκτέλεσης του ερωτήματος. Από τη στιγμή που έχει ολοκληρωθεί ο συντακτικός και σημασιολογικός έλεγχος από τον Query

Parser το ερώτημα προωθείται στο υποσύστημα μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων, το οποίο από το σημείο αυτό και μετά αναλαμβάνει την όλη διαδικασία εκτέλεσης του. Έχοντας ως είσοδο ένα ερώτημα στη γλώσσα E-SQL, το υποσύστημα «γεννάει» πολλαπλά σχεσιακά ερωτήματα τα οποία εκτελεί στο περιβάλλον του ΣΔΒΔ Postgres. Ακολούθως, επιστρέφει το αποτέλεσμα στη μορφή ενός polymorph στο γραφικό περιβάλλον το οποίο είναι πάλι υπεύθυνο για την παρουσίαση του στο χρήστη.

Στο σημείο αυτό, εφόσον περιγράφηκε η γενική αρχιτεκτονική του συνολικού συστήματος μπορούμε να προχωρήσουμε σε εκείνη του υποσυστήματος μετατροπής των ερωτημάτων σε PL\SQL και εκτέλεσης τους.

3.2 Αρχιτεκτονική υποσυστήματος μετατροπής των ερωτημάτων σε PL\SQL και εκτέλεσης τους

Εξετάζοντας το διάγραμμα που δίνεται στο σχήμα 3.2 «από κάτω προς τα πάνω» παρατηρούμε τα εξής υποσυστήματα:

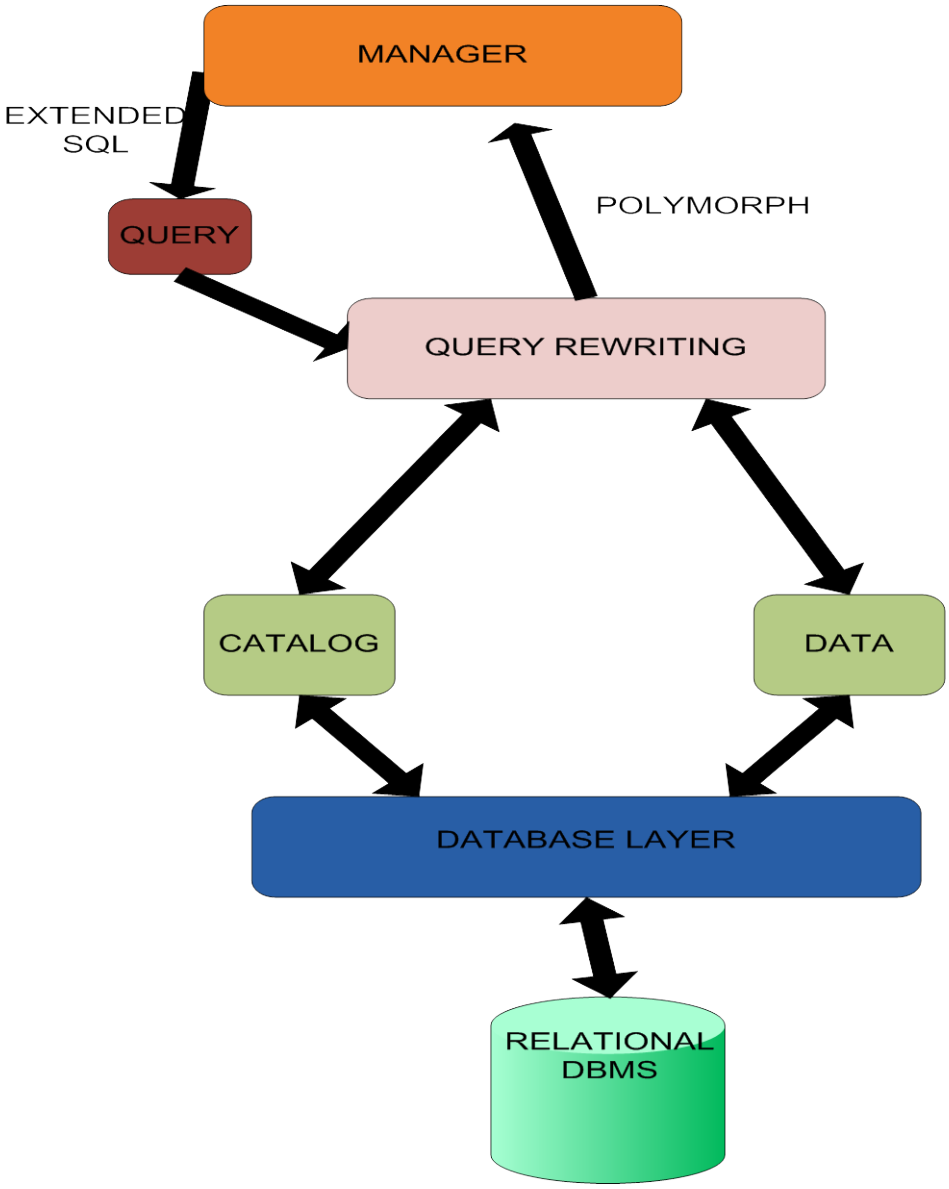
- Το **υποσύστημα επικοινωνίας με το σχεσιακό ΣΔΒΔ** που αποτελεί ένα επίπεδο επικοινωνίας με το ΣΔΒΔ.
- Το **υποσύστημα του καταλόγου** που είναι υπεύθυνο για το χειρισμό των meta-δεδομένων τόσο του context όσο και των morphs/polymorphs.
- Το **υποσύστημα διαχείρισης δεδομένων** που είναι υπεύθυνο για το χειρισμό των δεδομένων τόσο του context όσο και των morphs/polymorphs.
- Το **υποσύστημα αναπαράστασης ερωτημάτων** που αναπαριστά το ερώτημα σε βολική για την επεξεργασία του μορφή.
- Το **υποσύστημα μετατροπής των ερωτημάτων σε PL\SQL και εκτέλεσής τους** που μετατρέπει τελικά το ερώτημα σε πολλαπλά σχεσιακά τα οποία και εκτελεί ενώ τελικά είναι αυτό που επιστρέφει το αποτέλεσμα.

Εξετάζοντας την πορεία εκτέλεσης ενός ερωτήματος, αρχικά αυτό προωθείται στο υποσύστημα μετά τον συντακτικό και σημασιολογικό έλεγχο από τον Query Parser. Στη συνέχεια το υποσύστημα αναπαράστασης των ερωτημάτων το μετατρέπει στην επιθυμητή μορφή για την επεξεργασία του και το προωθεί στο υποσύστημα μετατροπής των ερωτημάτων σε PL\SQL και εκτέλεσής τους. Εκεί δημιουργούνται συνδυασμοί morphs των αντίστοιχων polymorphs που λαμβάνουν μέρος στο ερώτημα. Κάθε τέτοιος συνδυασμός περιλαμβάνει ένα morph από κάθε polymorph, με τον περιορισμό ότι τα morphs του κάθε συνδυασμού έχουν οριστεί με βάση το ίδιο context instance. Έτσι τελικά το σύστημα «γεννάει» σχεσιακά ερωτήματα τα οποία εκτελεί για όλους τους συνδυασμούς. Η εκτέλεση

γίνεται πάνω από τους πίνακες που είναι αποθηκευμένα τα δεδομένα των morphs που ανήκουν στον εκάστοτε συνδυασμό.

Για την υλοποίηση καθενός από τα παραπάνω συστήματα χρησιμοποιήθηκε το αντικειμενοστραφές μοντέλο. Κάθε ένα αποτελεί μια αυτόνομη μονάδα (συλλογή αντικειμένων) χωρίς ο τρόπος λειτουργίας του να είναι ορατός στα υπόλοιπα. Χρησιμοποιείται ως «μαύρο κουτί» και η τεχνική αυτή επιτρέπει την τροποποίηση ή αντικατάσταση του χωρίς να επηρεαστούν οι λειτουργίες του υπόλοιπου υποσυστήματος, κάτι που ήταν στόχος καθ' όλη τη διάρκεια της υλοποίησης. Και αυτό διότι με την τεχνική αυτή, μελλοντικές επεκτάσεις μπορούν να υλοποιηθούν κάνοντας ελάχιστες τροποποιήσεις στο όλο σύστημα. Επιπλέον, με τον τρόπο αυτό επιτυγχάνεται ο αποδοτικότερος έλεγχος του συνολικού υποσυστήματος αφού κάθε μονάδα από τη στιγμή που είναι αυτόνομη, μπορεί να ελεγχθεί ανεξάρτητα από τις υπόλοιπες.

Μια άλλη σημαντική παράμετρος του όλου υποσυστήματος είναι ότι στο σύστημα διαχείρισης των δεδομένων υλοποιήθηκαν δύο τρόποι για την αποθήκευση των δεδομένων των morphs/polymorphs (υλοποιήσεις A,B). Σκοπός της επιλογής αυτής ήταν η συγκριτική μελέτη της απόδοσης του συστήματος σε κάθε περίπτωση ώστε αφενός να καταλήξουμε στην αποδοτικότερη και αφετέρου να αποκτήσουμε μια περισσότερη ολοκληρωμένη εικόνα σχετικά με το overhead που εισάγεται από το υποσύστημα.



Σχήμα 3.2 Αρχιτεκτονική υποσυστήματος μετατροπής των ερωτημάτων σε PL\SQL και εκτέλεσής τους

3.3 Τρόποι αποθήκευσης των polymorphs στο context-aware ΣΔΒΔ

Στην υποενότητα αυτή θα αναφερθούμε στις δύο υλοποιήσεις που επιλέξαμε σχετικά με τον τρόπο αποθήκευσης των polymorphs στο context-aware ΣΔΒΔ. Πριν αναλύσουμε τους δύο αυτούς τρόπους αποθήκευσης κρίνεται σκόπιμο να αναφερθούμε στους λόγους οι οποίοι μας οδήγησαν στην κατεύθυνση αυτή αλλά και τους λόγους για τους οποίους δεν ακολουθήσαμε παρόμοιο χειρισμό στην αποθήκευση των υπόλοιπων πληροφοριών στο σύστημα, τα οποία

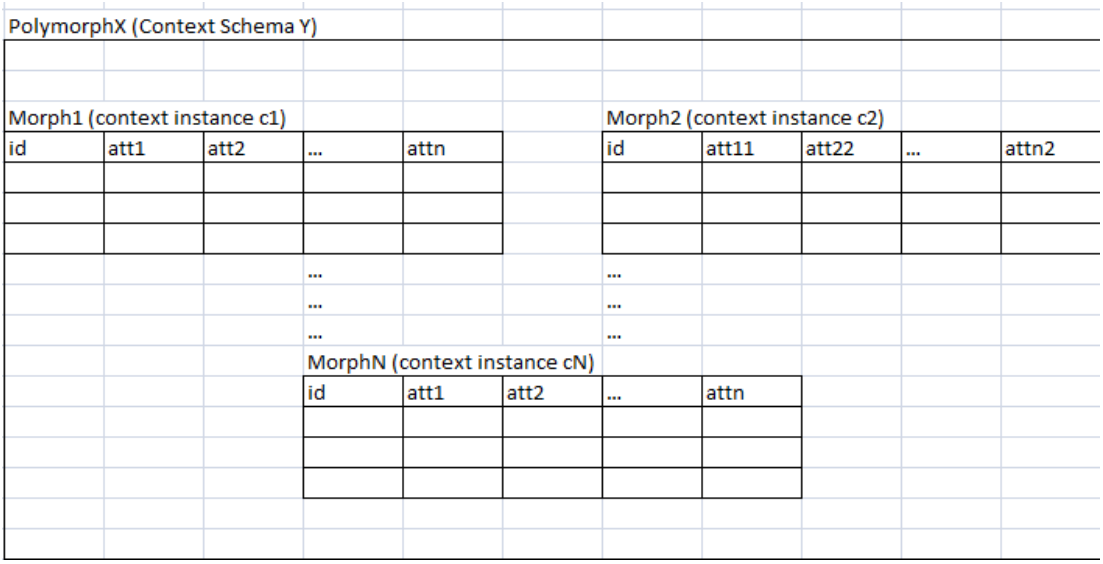
όπως αναφέρονται και σε προηγούμενη υποενότητα, αποτελούν τα metadata σχετικά με το context, δηλαδή τα context instances.

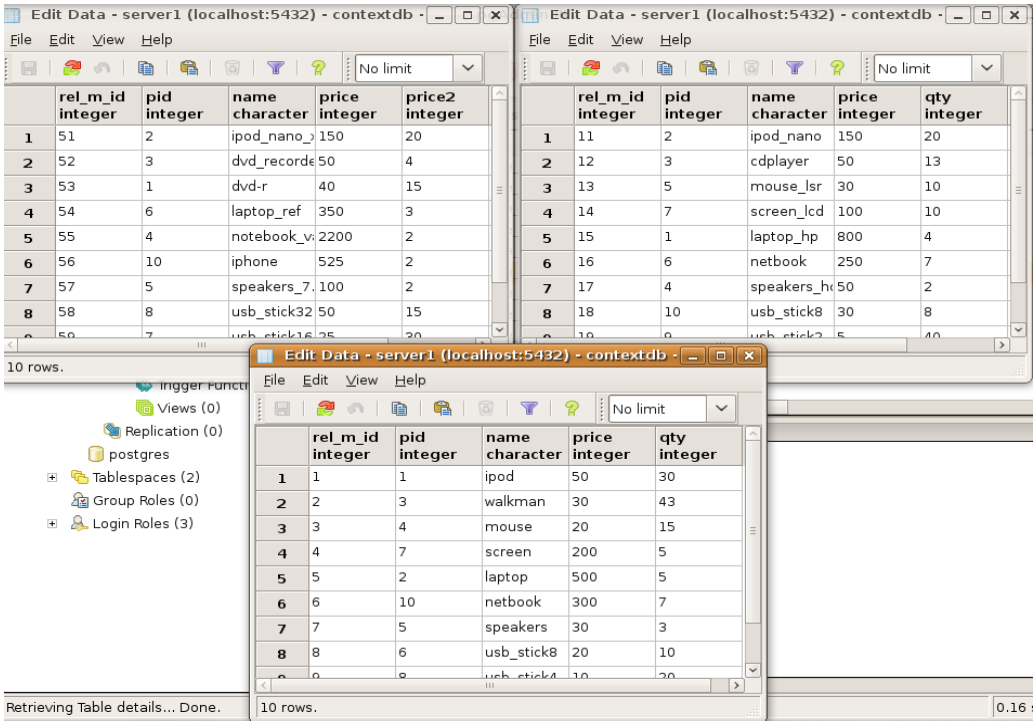
Αν λάβουμε υπ' όψιν μας τη διαδικασία σχετικά με την μετατροπή και τελικά την εκτέλεση των ερωτημάτων και ειδικότερα το κομμάτι που αφορά την εκτέλεση των συνθηκών σχετικά με το context, τότε διαπιστώνουμε ότι το στάδιο αυτό της επεξεργασίας του ερωτήματος είναι λογικό να καταναλώνει λίγο χρόνο. Και αυτό διότι καταρχάς υπάρχει η πιθανότητα να μην ορίζεται καμιά συνθήκη για ένα ή περισσότερα polymorph που λαμβάνουν μέρος στο ερώτημα οπότε και να μην απαιτείται να εκτελεστεί τίποτα. Ακόμα όμως και να ορίζεται κάποια συνθήκη δεν θεωρείται αναμενόμενο να έχουν οριστεί τέτοιες που θα περιλαμβάνουν πολλά attributes ως παραμέτρους. Και αυτό διότι οι πίνακες που αποθηκεύονται τα context instances δεν αναμένεται να περιέχουν πολλές στήλες στη γενική περίπτωση καθώς το context του χρήστη μπορεί να περιγραφεί -και να μοντελοποιηθεί ακολούθως- σε ικανοποιητικό βαθμό χωρίς να απαιτείται η χρήση πολλών παραμέτρων. Επίσης αυτό έχει σαν συνέπεια να μην έχει η νόημα η ύπαρξη πολυάριθμων context schemas αφού ούτε πολλά attributes υπάρχουν αλλά ούτε και πολλά context schemas χρησιμοποιούνται στην πράξη από τη στιγμή που τελικά για κάθε ερώτημα και για να είναι δυνατόν να γίνουν joins μεταξύ των polymorphs θα πρέπει αυτά να έχουν οριστεί με βάση το ίδιο. Αν σε όλα τα παραπάνω προσθέσουμε ότι ο αριθμός των context instances για να παρασταθεί επαρκώς το context είναι επίσης μικρός και ακόμα ότι δεν χρησιμοποιούνται join conditions ανάμεσα σε context schemas τότε εύλογα καταλήγουμε στο συμπέρασμα ότι διαφορετικές υλοποιήσεις στον τρόπο αποθήκευσης των context schemas θα επέφεραν μικρές διαφοροποιήσεις στους χρόνους εκτέλεσης των ερωτημάτων οι οποίοι όπως θα αναφέρουμε αργότερα σε επόμενο κεφάλαιο, για το στάδιο εκτέλεσης των συνθηκών context είναι εξαιρετικά μικροί. Στην ακραία περίπτωση που αυτό καταστεί αναγκαίο, ο πίνακας αυτός μπορεί άλλωστε να σηκωθεί στη μνήμη.

Σε αντίθεση με τα παραπάνω τα δεδομένα των morphs είναι πολυάριθμα όπως και τα attributes αυτών. Επίσης πάνω σε αυτά τα δεδομένα εφαρμόζουμε διαρκώς πολλαπλά joins. Είναι λογικό διαφορετικές υλοποιήσεις στον τρόπο αποθήκευσης τους να μεταβάλλουν το συνολικό χρόνο απόκρισης του συστήματος αφού όπως θα φανεί και σε επόμενο κεφάλαιο όπου θα παρουσιαστούν τα αποτελέσματα, πάνω σε αυτά καταναλώνεται ο περισσότερος χρόνος της εκτέλεσης κάθε ερωτήματος. Συνεπώς όχι μόνο έχει νόημα η σύγκριση μεταξύ διαφορετικών υλοποιήσεων στο κομμάτι αυτό αλλά μπορεί να οδηγήσει και σε χρήσιμα συμπεράσματα.

Υλοποίηση Α

Στην υλοποίηση αυτή κάθε morph αποθηκεύεται σε έναν ξεχωριστό σχεσιακό πίνακα στη βάση δεδομένων με όνομα cat_mor_X όπου X είναι το αναγνωριστικό του morph αυτού. Ως στήλες του πίνακα αυτού ορίζονται τα attributes τα οποία έχουν οριστεί για το morph αυτό και στις γραμμές βρίσκονται οι tuples των δεδομένων του. Αυτό έχει ως συνέπεια όπου χρειάζεται για παράδειγμα να γίνει κάποια πράξη προβολής των δεδομένων των morphs ενός polymorph το σύστημα να κάνει προσπέλαση σε τόσους πίνακες όσα τα morphs που ανήκουν στο polymorph αυτό, κάτι που προκαλεί καθυστέρηση στο σύστημα. Αντίθετα όπου χρειάζεται να γίνουν πράξεις πολλαπλών γινομένων επειδή οι πίνακες πάνω στους οποίους θα γίνουν οι πράξεις αυτές είναι μικροί αυτές γίνονται γρήγορα. Παρακάτω δίνεται σχηματικά η αναπαράσταση της υλοποίησης αυτής καθώς και ένα screenshot από πίνακες στους οποίους αποθηκεύονται τα δεδομένα τριών morph ενός polymorph.





Σχήμα 3.3 Παράδειγμα αποθήκευσης δεδομένων για την υλοποίηση Α

Υλοποίηση Β

Στην υλοποίηση αυτή δημιουργούμε έναν πίνακα για κάθε polymorph με το όνομα pol_cat_mor_table_X όπου X είναι το αναγνωριστικό του polymorph. Ως στήλες του πίνακα αυτού χρησιμοποιούνται όλα τα attributes που έχουν οριστεί για όλα τα morphs του polymorph συν άλλη μια στήλη που ορίζει σε ποιο morph ανήκει η συγκεκριμένη tuple. Προφανώς για attributes που εμφανίζονται παραπάνω από μία φορά στα morphs χρησιμοποιείται μια στήλη στον πίνακα. Κάθε tuple δεδομένων καθενός morph καταλαμβάνει μια tuple στον πίνακα αυτό με τη σημείωση ότι τα πεδία των attributes τα οποία δεν ορίζονται για το morph αυτό έχουν την τιμή NULL. Γενικά ο πίνακας αυτός είναι αρκετά αραιός, ιδιαίτερα σε περιπτώσεις όπου ανάμεσα στα morphs υπάρχουν λίγα κοινά attributes ενώ είναι και αρκετά μεγάλος σε πλάτος. Ωστόσο παρέχει την ευκολία ότι μπορεί με προσπέλαση σε έναν μόνο πίνακα να έχει διαθέσιμα όλα τα δεδομένα ενός polymorph. Χαρακτηριστικό παράδειγμα αποτελούν απλά ερωτήματα που αφορούν κάποια πράξη προβολής των δεδομένων των morphs ενός polymorph. Αντίθετα σε ερωτήματα με πολλαπλά joins ανάμεσα σε polymorphs αναμένεται να υπάρχει καθυστέρηση σε σχέση με την προηγούμενη υλοποίηση διότι οι πίνακες που συμμετέχουν στις πράξεις αυτές είναι μεγάλοι και είναι λίγες οι σχετικές με το κάθε ερώτημα tuples. Παρακάτω δίνεται σχηματικά η αναπαράσταση της υλοποίησης αυτής καθώς και ένα screenshot από ένα πίνακα στον οποίο αποθηκεύονται τα δεδομένα ενός morph.

Polymorph1 (context schema c1)

id	morph_id	att1	att2	...	attn

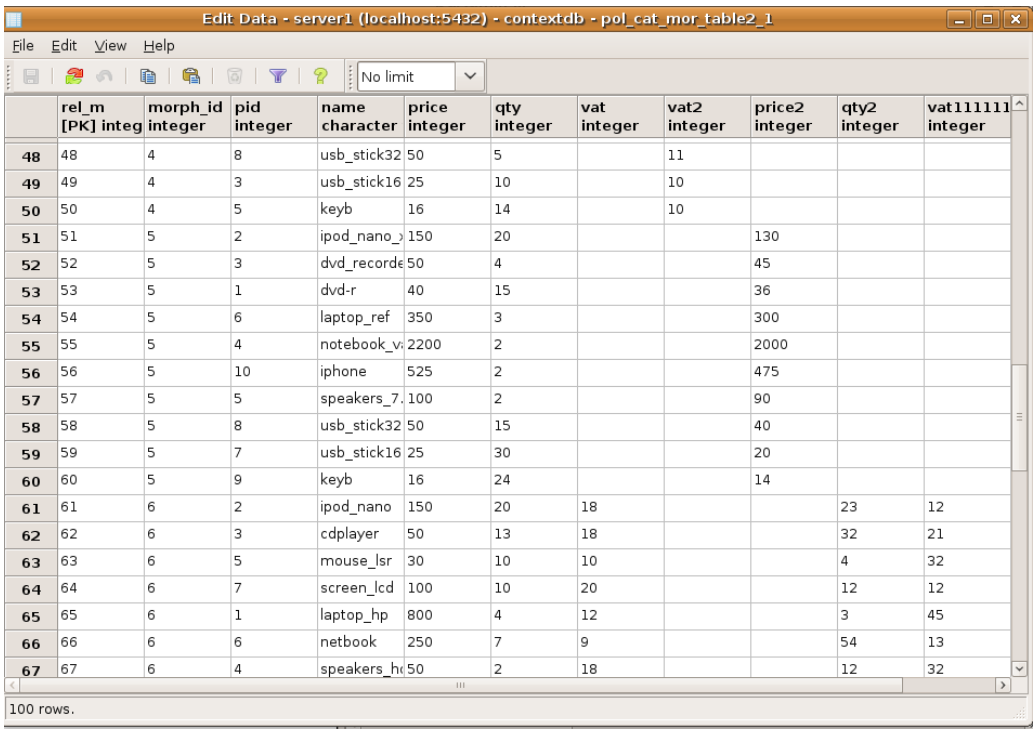
Polymorph2 (context schema c2)

id	morph_id	att11	att22	...	attn2

...
...
...

PolymorphN (context schema cN)

id	morph_id	att1	att2	...	attn



	rel_m [PK] integ	morph_id integer	pid integer	name character	price integer	qty integer	vat integer	vat2 integer	price2 integer	qty2 integer	vat111111 integer
48	48	4	8	usb_stick32	50	5		11			
49	49	4	3	usb_stick16	25	10		10			
50	50	4	5	keyb	16	14		10			
51	51	5	2	ipod_nano_	150	20			130		
52	52	5	3	dvd_recorde	50	4			45		
53	53	5	1	dvd-r	40	15			36		
54	54	5	6	laptop_ref	350	3			300		
55	55	5	4	notebook_v	2200	2			2000		
56	56	5	10	iphone	525	2			475		
57	57	5	5	speakers_7.	100	2			90		
58	58	5	8	usb_stick32	50	15			40		
59	59	5	7	usb_stick16	25	30			20		
60	60	5	9	keyb	16	24			14		
61	61	6	2	ipod_nano	150	20	18			23	12
62	62	6	3	cdplayer	50	13	18			32	21
63	63	6	5	mouse_lsr	30	10	10			4	32
64	64	6	7	screen_lcd	100	10	20			12	12
65	65	6	1	laptop_hp	800	4	12			3	45
66	66	6	6	netbook	250	7	9			54	13
67	67	6	4	speakers_h	50	2	18			12	32

Σχήμα 3.4 Παράδειγμα αποθήκευσης δεδομένων για την υλοποίηση Β

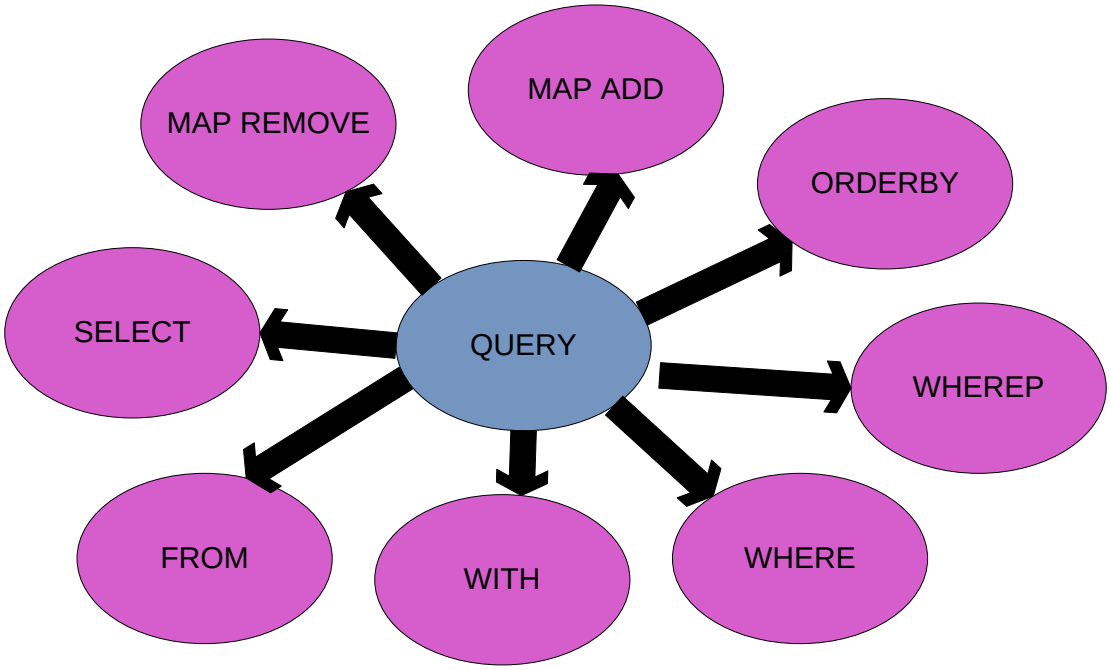
3.4 Λεπτομέρειες υποσυστημάτων

Στην ενότητα αυτή παρατίθενται οι λεπτομέρειες σχεδίασης και υλοποίησης για τα επιμέρους υποσυστήματα που αποτελούν το υποσύστημα μετατροπής ερωτημάτων σε PL\SQL και εκτέλεσης τους.

3.4.1 Υποσύστημα αναπαράστασης, μετατροπής και εκτέλεσης ερωτήματος

3.4.1.1 Υποσύστημα αναπαράστασης ερωτήματος

Όπως φαίνεται και από το συνολικό διάγραμμα του μοντέλου (σχήμα 3.1) που προτείνεται στο μοντέλο που περιγράφηκε στην ενότητα 2.4, ακριβώς πριν το υποσύστημα που περιγράφεται στην παρούσα διπλωματική εργασία προηγείται εκτός των άλλων μια διαδικασία επεξεργασίας του ερωτήματος. Αρχικά ο χρήστης όταν θέτει κάποιο ερώτημα χρησιμοποιεί την γλώσσα ESQL (Extended-Sql) και όπως είναι προφανές αυτή η μορφή δεν είναι η καταλληλότερη για τη μετέπειτα επεξεργασία. Συνεπώς, θα μπορούσαμε να ισχυριστούμε ότι το υποσύστημα αναπαράστασης των ερωτημάτων λειτουργεί ως ένα interface το οποίο δέχεται στην κατάλληλη μορφή το ερώτημα που θέτει ο χρήστης και το προωθεί στο επόμενο και τελικό στάδιο της μετατροπής του. Στη συνέχεια ακολουθεί ένα διάγραμμα των συστατικών μερών του υποσυστήματος αυτού καθώς και η περιγραφή των σημαντικότερων εξ' αυτών.



Σχήμα 3.5 Αναπαράσταση μοντελοποίησης ερωτήματος στα επιμέρους αντικείμενα

Όπως φαίνεται και από το παραπάνω διάγραμμα κάθε ερώτημα έχει χωριστεί στα επιμέρους συστατικά που το αποτελούν. Αυτό αρχικά έγινε ώστε να διευκολύνεται η επεξεργασία κάθε μιας συνθήκης αλλά συνάμα διότι δεν αποτελούν όλες οι συνθήκες απαραίτητη πληροφορία για την ύπαρξη κάθε ερωτήματος. Για παράδειγμα είναι πιθανό ο χρήστης να επιθυμεί να θέσει κάποιο ερώτημα το οποίο δε θα περιλαμβάνει τη συνθήκη WITH. Το σύστημα, θα

πρέπει να είναι σε θέση να απαντήσει στο ερώτημα αυτό λαμβάνοντας υπ' όψιν ότι θα προχωρήσει κανονικά στην προβλεπόμενη επεξεργασία και μετατροπή του δοθέντος ερωτήματος με τη σημείωση ότι η συνθήκη αυτή είναι κενή και το στοιχείο αυτό θα πρέπει να μπορεί να οριστεί με σαφή τρόπο από το χρήστη. Συνεπώς η ανάγκη διάσπασης του ερωτήματος σε μικρότερες και νοηματικά ανεξάρτητες συνθήκες καθίσταται αρκετά προφανής. Εν ολίγοις το σύστημα είναι σε θέση να εκτελέσει από απλά ερωτήματα της μορφής «SELECT X FROM Y» μέχρι πολύπλοκα ερωτήματα με πολυάριθμες συνθήκες.

Από το διάγραμμα παρατηρούμε ότι το αντικείμενο ερώτημα χωρίζεται στα επιμέρους αντικείμενα SELECT, FROM, WITH, WHERE, WHEREP, ORDERBY, MAP ADD, MAP REMOVE τα οποία με τη σειρά τους χωρίζονται σε λίστες μικρότερων αντικειμένων τα οποία ενθυλακώνουν και όλες τις απαιτούμενες μεθόδους για τη λειτουργία τους. Στο σημείο αυτό και πριν παραθέσουμε το παράδειγμα ενός ερωτήματος για να γίνει περισσότερο εμφανές το τι ακριβώς αντιπροσωπεύει η κάθε συνθήκη κρίνεται σκόπιμο να αναφερθούμε στη συνθήκη WHEREP. Για τη συνθήκη αυτή, καταρχάς πρέπει να τονίσουμε ότι δεν υπάρχει ούτε στην κλασική SQL αλλά ούτε και στην E-SQL που είναι η γλώσσα του συνολικού μοντέλου. Με άλλα λόγια αποτελεί μια «κατασκευασμένη» συνθήκη ο ορισμός της οποίας όμως κρίνεται απαραίτητος για τη σωστή λειτουργία του συστήματός μας. Και αυτό διότι, πρέπει να είμαστε σε θέση να διαχωρίζουμε τις συνθήκες WHERE της μορφής <attribute><symbol><value> από τις συνθήκες <attribute><symbol><attribute> διότι οι τελευταίες που είναι γνωστές και ως join conditions θα οδηγήσουν σε πράξεις γινομένων και όπως γνωρίζουμε οι πράξεις αυτές καταναλώνουν τον περισσότερο χρόνο για την εκτέλεση του κάθε ερωτήματος και εμφανίζουν τη μεγαλύτερη πολυπλοκότητα. Αν λάβουμε δε υπ' όψιν μας, ότι ο παράγοντας χρόνος, και ειδικότερα η στο μέτρο του δυνατού ταχύτερη απόκριση του συστήματος είναι παράγοντας κεφαλαιώδους σημασίας για την παρούσα εργασία, τότε εύλογα καταλήγουμε στο ότι οι συνθήκες αυτές πρέπει να διαχειρίζονται με ειδικό τρόπο και κατ' επέκταση να αναπαρίστανται με ειδικό τρόπο. Επιπροσθέτως θα πρέπει να αναφερθούμε και στο αντικείμενο MAP και σε τί αυτό αντιπροσωπεύει. Εφόσον ο χρήστης λοιπόν ορίσει μια συνθήκη MAP τότε δηλώνει ότι επιθυμεί να προσθέσει (ADD) ή να αφαιρέσει (REMOVE) ένα ή περισσότερα attributes από το context schema του polymorph που επιστρέφεται ως αποτέλεσμα. Αναλυτικότερα σχετικά το πώς επιτυγχάνεται αυτό θα αναφερθούμε στη συνέχεια. Ακόμα, πρέπει να αναφερθούμε στο αντικείμενο WITH το οποίο αντιπροσωπεύει τις συνθήκες που θα πρέπει να ικανοποιούν τα polymorphs του ερωτήματος σχετικά με το context.

Ακολουθεί η παράθεση ενός ερωτήματος στη γλώσσα E-SQL ώστε να γίνει περισσότερο ευδιάκριτο πως αναπαρίσταται ένα ερώτημα, ποια είναι τα αντικείμενα στα οποία το έχουμε χωρίσει και τι περιλαμβάνει η κάθε μια από αυτές τις συνθήκες (αντικείμενα). Έστω μια

πολυμορφική βάση δεδομένων η οποία περιλαμβάνει τα polymorphs PRODUCT, STORE και INVENTORY όπου το polymorph PRODUCT περιλαμβάνει τα δεδομένα σχετικά με τα προϊόντα που πωλούνται όπως για παράδειγμα η τιμή τους, ο Φ.Π.Α και το διαθέσιμο απόθεμα, το polymorph STORE περιλαμβάνει τα δεδομένα σχετικά με τα καταστήματα όπως για παράδειγμα τη διεύθυνση τους, το νομό στον οποίο υπάγονται και τον αριθμό των υπαλλήλων που απασχολούν και το polymorph INVENTORY περιλαμβάνει τα δεδομένα σχετικά με την αποθήκη της εταιρείας δηλαδή πληροφορίες σχετικά με το ποια προϊόντα μπορούν να προωθηθούν σε ποια καταστήματα, σε ποια τιμή και πόσο είναι το διαθέσιμο απόθεμα. Έστω λοιπόν ότι θέλουμε να θέσουμε ένα ερώτημα το οποίο να αφορά το context <GREECE, year>2004 > το οποίο θα επιστρέφει την τιμή στην οποία πωλούνται τα προϊόντα, τον αριθμό των εργατών που εργάζονται στα καταστήματα και το διαθέσιμο απόθεμα στην αποθήκη ταξινομημένα με βάση το απόθεμα όπου η τιμή πώλησης θα είναι μεγαλύτερη από 50, το διαθέσιμο απόθεμα στο κατάστημα θα είναι μεγαλύτερο από 10, ο αριθμός των υπαλλήλων του καταστήματος θα είναι μικρότερος του 15, η τιμή στην αποθήκη θα είναι μικρότερη από 100 και το απόθεμα της αποθήκης θα είναι μεγαλύτερο από 20 ενώ στο αποτέλεσμα θέλουμε να αφαιρέσουμε το attribute year από το context schema του επιστρεφόμενου ως αποτέλεσμα polymorph

Το προαναφερθέν ερώτημα τίθεται στην παρακάτω μορφή στην E-SQL:

```
SELECT      PRODUCT.price, STORE.workers, INVENTORY.qty
FROM        PRODUCT, STORE, INVENTORY
WITH        PRODUCT::location =’GREECE’ AND PRODUCT::year>2004 AND
            STORE::location =’GREECE’ AND STORE:: year >2004  AND
            INVENTORY::location=’GREECE’ AND INVENTORY:: year >2004
WHERE       PRODUCT.price>50 AND PRODUCT.qty>10 AND STORE.workers<15
            AND INVENTORY.price<100 AND INVENTORY.qty>20 AND
            PRODUCT.pid=INVENTORY.pid AND STORE.sid=INVENTORY.sid
ORDER BY    INVENTORY.qty
MAP REMOVE  year
```

Από το παραπάνω είναι προφανής η σημασιολογία των αντικειμένων SELECT, FROM, WITH, ORDERBY και MAP ενώ όπως έχει αναλυθεί το αντικείμενο WHERE χωρίζεται σε δυο αντικείμενα ανάλογα με το αν η συνθήκη θα οδηγήσει σε πράξη γινομένου ή όχι. Παρακάτω δίνεται το περιεχόμενο των αντικειμένων:

```
SELECT->      (PRODUCT,price),(STORE,workers),(INVENTORY,qty)
FROM->        (PRODUCT,STORE,INVENTORY)
WITH->        (PRODUCT, location ='GREECE', year >2004 ),(STORE,
              location ='GREECE', year >2004), (INVENTORY, location
              ='GREECE', year >2004)
WHERE->       (PRODUCT, price>50, qty>10),(STORE,
              workers<15),(INVENTORY, price<100, qty>20)
WHEREP->      (PRODUCT.pid,=,INVENTORY.pid),
              (STORE.sid,=,INVENTORY.sid)
ORDERBY->     (INVENTORY,qty)
MAP REMOVE-> (year)
```

Από τη στιγμή λοιπόν που έχει μοντελοποιηθεί το ερώτημα σε μια μορφή η οποία είναι αναγνωρίσιμη από το σύστημα μπορούμε να προχωρήσουμε στο επόμενο κομμάτι του συστήματος όπου θα μετατρέψουμε και τελικά θα εκτελέσουμε το ερώτημα. Στο σημείο όμως αυτό θα πρέπει να αναφέρουμε ότι το υποσύστημα αυτό που μόλις περιγράψαμε εκτός από τη μοντελοποίηση του ερωτήματος προσφέρει επιπρόσθετες υπηρεσίες και δεδομένα στο όλο σύστημα. Για παράδειγμα έστω ότι ο χρήστης έχει ήδη εισάγει ένα ερώτημα το οποίο όμως ακόμα δεν έχει εκτελέσει. Στην περίπτωση που επιθυμεί να κάνει οποιαδήποτε αλλαγή, το σύστημα του παρέχει αυτή τη δυνατότητα και μάλιστα αν προσπαθήσει να εισάγει την ίδια συνθήκη δύο φορές το σύστημα τον αποτρέπει. Επίσης, σε ανώτερο επίπεδο μας είναι απαραίτητο να γνωρίζουμε τη λίστα με όλα τα attributes που χρησιμοποιούνται σε όλες τις συνθήκες του ερωτήματος για κάθε polymorph και αυτή είναι μια πληροφορία που δεν παρέχεται έτοιμη στο σύστημα αλλά τη δημιουργεί αυτό. Με άλλα λόγια το υποσύστημα αυτό εκτός από τη μοντελοποίηση του ερωτήματος επιτελεί και κάποιες μικρές –πλην όμως σημαντικές- διαδικασίες parsing που απαιτούνται από το σύστημα σε κατώτερο επίπεδο.

3.4.1.2 Υποσύστημα μετατροπής και εκτέλεσης ερωτημάτων

Το υποσύστημα μετατροπής και εκτέλεσης ερωτημάτων, δοθέντος ενός ερωτήματος στη γλώσσα E-SQL, είναι υπεύθυνο για τη μετατροπή του, την εκτέλεσή του και τελικά την επιστροφή του αποτελέσματος. Όπως είναι λογικό, για το λόγο ότι κάθε polymorph περιλαμβάνει συνήθως αρκετά morphs ένα ερώτημα στη γλώσσα E-SQL, θα μεταφράζεται σε πολλαπλά σχεσιακά ερωτήματα SQL τα οποία θα εκτελούνται ανάμεσα σε morphs των

polymorphs αυτών τα οποία έχουν οριστεί με το ίδιο context instance του ίδιου context schema.

Θέλοντας να περιγράψουμε τη διαδικασία σε γενικές γραμμές και πριν αναφερθούμε αναλυτικότερα σε κάθε επιμέρους κομμάτι της μπορούμε να πούμε ότι δοθέντος ενός ερωτήματος αρχικά εκτελούμε τη συνθήκη WITH ώστε να εξαλείψουμε τα morphs που δε την ικανοποιούν. Στη συνέχεια για τα εναπομείναντα morphs ελέγχουμε αν διαθέτουν όλα τα attributes που απαιτούνται από το ερώτημα με σκοπό να τα μειώσουμε ακόμα περισσότερο ενώ ακολούθως έχουμε εφαρμόσει δύο τεχνικές και για τις δύο υλοποιήσεις. Η πρώτη δημιουργεί προσωρινούς πίνακες με όσα morph έχουν απομείνει και στη συνέχεια εκτελεί τα ερωτήματα πάνω στους πίνακες αυτούς ενώ η δεύτερη εκτελεί απευθείας τα ερωτήματα στους αρχικούς πίνακες. Τελικά η εκτέλεση καθενός από αυτά τα σχεσιακά ερωτήματα που προκύπτουν από τη μετατροπή του αρχικού οδηγεί στη δημιουργία ενός morph και όλα αυτά τα morphs μαζί δημιουργούν ένα polymorph το οποίο αποτελεί και το αποτέλεσμα της όλης επεξεργασίας. Εφόσον έχουν τεθεί από το χρήστη και συνθήκες MAP τότε αυτές εκτελούνται επί του τελικού polymorph και στη συνέχεια αυτό επιστρέφεται ως αποτέλεσμα.

Ακολουθεί αναλυτικότερη περιγραφή της παραπάνω διαδικασίας.

Εκτέλεση WITH,FROM

Στο πρώτο αυτό στάδιο μετατροπής του ερωτήματος εκτελούνται οι συνθήκες που ορίζονται στο αντικείμενο WITH με τελικό στόχο να μείνουν προς επεξεργασία μόνο τα morphs εκείνα από κάθε polymorph τα οποία ικανοποιούν τους περιορισμούς που τίθενται από το ερώτημα και αφορούν το context.

Αρχικά παίρνοντας τις λίστες των polymorphs όπως αυτές δίνονται στα αντικείμενα FROM και WITH διατηρούμε όλα τα morphs των polymorphs εκείνων τα οποία υπάρχουν στη λίστα του FROM και όχι σε εκείνη του WITH. Πρακτικά αυτό σημαίνει ότι για τα polymorphs αυτά δεν έχουν οριστεί περιορισμοί που να αφορούν το context και συνεπώς όλα τα morphs τους θα συνεχίσουν στο επόμενο στάδιο της μετατροπής.

Τα υπόλοιπα polymorphs, δηλαδή αυτά για τα οποία έχουν οριστεί συνθήκες που να αφορούν το context υπόκεινται στην παρακάτω επεξεργασία. Ελέγχεται αν το context schema τους έχει ως attributes αυτά για τα οποία έχουν οριστεί συνθήκες από το χρήστη και ακολούθως αν το συγκεκριμένο context instance για το οποίο έχει οριστεί κάθε morph ικανοποιεί τους περιορισμούς αυτούς. Ο πρώτος έλεγχος γίνεται στους πίνακες του καταλόγου και ο δεύτερος στον πίνακα που είναι αποθηκευμένα τα δεδομένα του context. Όσα morphs λοιπόν έχουν context instances τα οποία ικανοποιούν τους περιορισμούς αυτούς περνούν το στάδιο αυτό, διαφορετικά αφαιρούνται από τη διαδικασία επεξεργασίας. Έτσι τελικά καταλήγουμε να έχουμε για κάθε polymorph μια λίστα από morphs από την οποία αφού ολοκληρωθούν και τα

επόμενα στάδια της επεξεργασίας του ερωτήματος θα προκύψουν τελικά τα morphs εκείνα πάνω στα οποία θα εκτελεστεί το ερώτημα.

Αφαίρεση morphs ανάλογα με τα ζητούμενα από το ερώτημα attributes

Στο στάδιο αυτό έχουμε ως δεδομένα για κάθε polymorph μια λίστα από τα morphs του τα οποία είτε έχουν περάσει με επιτυχία τους ελέγχους σχετικά με το context τους είτε για τα οποία δεν έχουν οριστεί περιορισμοί στη συνθήκη WITH. Στο σημείο αυτό έχουμε επίσης ως δεδομένα από το υποσύστημα μοντελοποίησης και αναπαράστασης του ερωτήματος συγκεντρωτικά για κάθε polymorph ποια είναι τα attributes εκείνα τα οποία το αφορούν και αναφέρονται στο ερώτημα που τίθεται. Είναι προφανές ότι εφόσον ο χρήστης ορίζει κάποια συνθήκη σχετικά με κάποιο attribute (where clause) ή επιθυμεί την εμφάνιση του attribute αυτού ανάμεσα στο αποτέλεσμα (select clause), είναι απαραίτητο τα morphs πάνω στα οποία τελικά θα εκτελεστεί το ερώτημα να περιλαμβάνουν το attribute αυτό.

Εδώ θα πρέπει να θυμίσουμε ότι το σχήμα για κάθε κάποιο morph ορίζεται εντελώς ανεξάρτητα από εκείνο ενός άλλου ακόμα εντός του ίδιου polymorph οπότε ο παραπάνω έλεγχος είναι αναγκαίος. Για παράδειγμα μπορεί στην Ελλάδα να υπάρχει ένας ειδικός φόρος κατανάλωσης ενώ στη Γαλλία να μην υπάρχει καν. Συνεπώς αν ο χρήστης ορίσει σε κάποιο σημείο του ερωτήματος κάποια συνθήκη σχετικά με το φόρο αυτόν τότε αυτόματα όσα morphs αναφέρονται στη Γαλλία θα αφαιρεθούν από την επεξεργασία καθώς δε θα περιλαμβάνουν το attribute αυτό.

Οπότε στο στάδιο αυτό γίνεται ο παραπάνω έλεγχος ο οποίος πραγματοποιείται στους πίνακες του καταλόγου που αφορούν τα morphs. Στο τέλος του ελέγχου αυτού επιστρέφεται εκ νέου για κάθε polymorph μια λίστα από τα morphs του τα οποία περιλαμβάνουν όλα τα απαραίτητα attributes. Και από τη στιγμή που ο έλεγχος γίνεται μόνο στα morphs εκείνα τα οποία ολοκλήρωσαν επιτυχώς τον προηγούμενο έλεγχο σχετικά με το context (είτε το context τους ικανοποιεί τις συνθήκες είτε δεν έχουν οριστεί συνθήκες) τότε οι λίστες που προκύπτουν από το στάδιο αυτό περιλαμβάνουν morphs τα οποία έχουν περάσει με επιτυχία και τα δύο προαναφερθέντα στάδια.

Συνέχεια στην επεξεργασία και εκτέλεση ερωτημάτων με δύο τεχνικές

Ολοκληρώνοντας τη διαδικασία που αναφέρθηκε παραπάνω εφαρμόζουμε δύο τεχνικές σχετικά με την από εδώ και πέρα μετατροπή του ερωτήματος οι οποίες μεταβάλλουν σε κάποιο βαθμό και την εκτέλεσή του οπότε και θα τις εξετάσουμε ξεχωριστά. Στο σημείο αυτό πρέπει να αναφέρουμε ότι και οι δύο τεχνικές αυτές ξεκινούν από το ίδιο σημείο δηλαδή παίρνουν την ίδια είσοδο και δίνουν και τα αποτελέσματα τους στην ίδια μορφή. Επίσης ο

διαχωρισμός αυτός έγινε και για τις δύο υλοποιήσεις -όσον αφορά τον τρόπο αποθήκευσης των δεδομένων- που έχουν αναφερθεί σε προηγούμενη υποενότητα και στόχος μας είναι να μελετήσουμε την απόδοση του συστήματος ανάλογα με τη δημιουργία ή μη προσωρινών πινάκων που αποθηκεύουν τα δεδομένα των morphs που είναι απαραίτητα για την εκτέλεση των ερωτημάτων.

Τεχνική 1: Με τη δημιουργία ενδιάμεσων-προσωρινών πινάκων

Η πρώτη προσέγγιση στηρίζεται στο ότι δεν υπάρχει λόγος να εκτελέσουμε τελικά τα πολλαπλά σχεσιακά ερωτήματα που θα προκύψουν ανάμεσα στα morphs των polymorphs πάνω από τους αρχικούς πίνακες στους οποίους είναι αποθηκευμένα τα δεδομένα τους διότι τα attributes που τελικά χρειαζόμαστε είναι λίγα τις περισσότερες φορές σε σχέση με όσα ορίζονται για κάθε morph. Συνεπώς θα ήταν χρησιμότερο να δημιουργήσουμε ενδιάμεσους-προσωρινούς πίνακες στους οποίους θα αποθηκεύονται μόνο τα attributes εκείνα που μας ενδιαφέρουν και πάνω από τους οποίους θα είναι πιθανώς ταχύτερο να τρέξουμε τελικά τα ερωτήματα. Στην κατεύθυνση αυτή για κάθε morph εκτελούμε πάνω στον αρχικό πίνακα των δεδομένων του ένα ερώτημα το οποίο διατηρεί μόνο τις tuples εκείνες οι οποίες ικανοποιούν τους περιορισμούς που έχουν τεθεί στη συνθήκη WHERE για το polymorph αυτό από το χρήστη, ενώ κρατάμε και μόνο τις στήλες με τα attributes που μας ενδιαφέρουν. Με τα δεδομένα αυτά και για κάθε morph, δημιουργούμε ενδιάμεσους πίνακες οι οποίοι είναι προσωρινοί στη βάση δεδομένων –δηλαδή μετά την ολοκλήρωση της μετατροπής και εκτέλεσης του ερωτήματος θα σβηστούν- πάνω από τους οποίους θα τρέξουμε στη συνέχεια τα ερωτήματα που θα προκύψουν.

Με βάση τα παραπάνω στο σημείο αυτό έχουμε για κάθε polymorph μια λίστα με τα morphs του τα οποία θα πρέπει να λάβουμε υπ’ όψιν μας τελικά στην εκτέλεση του ερωτήματος και για κάθε morph από αυτά έχουμε τα δεδομένα του που μας αφορούν σε κάποιο προσωρινό πίνακα. Αυτό που απομένει είναι να επεξεργαστούμε τις συνθήκες WHEREP και στη συνέχεια να δημιουργήσουμε ένα μηχανισμό που θα «γεννάει» ερωτήματα ανάμεσα στα morphs των polymorphs που ορίζονται στο ερώτημα.

Αρχικά και επειδή δεν είναι σίγουρο ότι ο χρήστης έχει ορίσει ορθά όλες τις συνθήκες του, γίνεται ένας έλεγχος για κάθε συνθήκη που ορίζεται στο αντικείμενο WHEREP ώστε τα polymorphs που λαμβάνουν μέρος στη συνθήκη να έχουν οριστεί με βάση το ίδιο context schema. Αυτό το λάθος από το χρήστη είναι δυνατόν να συμβεί αν αναλογιστούμε για παράδειγμα ότι μπορεί να θέσει ένα ερώτημα που να αφορά τα polymorphs X,Y,Z τα οποία να ορίζονται με βάση τα context schemas 1,1,2 αντίστοιχα και στο ερώτημα αυτό στη συνθήκη WITH να μην συμπεριλάβει κάποιο έλεγχο για το context του polymorph Z. Τότε σύμφωνα με όσα έχουν αναφερθεί παραπάνω όλα τα morphs του Z θα περάσουν από το

πρώτο στάδιο οπότε κάποια εξ' αυτών θα βρεθούν στο σημείο που είμαστε αυτή τη στιγμή ένα βήμα πριν την εκτέλεση πράξεων γινομένων ανάμεσα στα X και Z ή/και στα Y και Z τα οποία όμως δεν έχουν ορισθεί με βάση το ίδιο context schema και συνεπώς οι πράξεις αυτές δεν έχουν νόημα.

Οπότε μετά τον παραπάνω έλεγχο τελικά έχουν πλέον απομείνει μόνο οι συνθήκες WHEREP ανάμεσα σε polymorphs με βάση το ίδιο context schema και ακολούθως το σύστημα δημιουργεί N-άδες (όπου N ο αριθμός των polymorphs) ανάμεσα στα morphs των polymorphs αυτών όπου κάθε N-άδα περιλαμβάνει morphs που έχουν οριστεί με το ίδιο context instance. Τελικά πάνω στις N-άδες αυτές θα εκτελεστούν τα ερωτήματα που θα προκύψουν.

Για κάθε N-άδα εκτελείται ένα ερώτημα για το οποίο η συνθήκες SELECT και ORDERBY του αρχικού ερωτήματος τοποθετούνται αυτούσιες, στο FROM τη θέση των polymorphs παίρνουν οι προσωρινοί πίνακες στους οποίους είναι αποθηκευμένα τα δεδομένα κάθε morph της N-άδας και στη θέση του WHERE τοποθετούνται οι σχετικές συνθήκες που έχουν απομείνει από το αντικείμενο WHEREP. Αν στα παραπάνω προσθέσουμε ότι για κάθε morph έχουν ήδη εκτελεστεί οι συνθήκες στο WITH που το αφορούν αλλά και οι WHERE όταν δημιουργείτο ο προσωρινός πίνακας που βρίσκονται τα δεδομένα του που μας αφορούν τότε αντιλαμβανόμαστε ότι δεν έχει παραληφθεί κάποιο κομμάτι του αρχικού ερωτήματος.

Τεχνική 2: Χωρίς τη δημιουργία ενδιάμεσων-προσωρινών πινάκων

Στην ακριβώς προηγούμενη παράγραφο αναλύθηκε η διαδικασία επεξεργασίας και μετατροπής του ερωτήματος με τη δημιουργία ενδιάμεσων-προσωρινών πινάκων. Η δεύτερη προσέγγιση που ακολουθήσαμε στηρίζεται στην υπόθεση ότι οι πίνακες των morphs δεν είναι συνήθως και τόσο μεγάλοι όσον αφορά το πλάτος τους (δηλαδή το πλήθος των attributes) οπότε ίσως χάνουμε περισσότερο χρόνο με τη δημιουργία των ενδιάμεσων-προσωρινών πινάκων από το χρόνο που τελικά κερδίζουμε με το να τρέχουμε τα τελικά ερωτήματα πάνω σε μικρούς πίνακες που έχουν μόνο στήλες που μας ενδιαφέρουν. Σε αυτό πρέπει να προσθέσουμε και το γεγονός ότι τρέχοντας τα ερωτήματα στους αρχικούς πίνακες εκμεταλλευόμαστε την ύπαρξη indexes, partitioning κλπ κάτι που χάνουμε αν τα εκτελέσουμε πάνω από τους προσωρινούς.

Οπότε παραλείπουμε το σημείο εκείνο που αναφέρεται παραπάνω και αφορά τη δημιουργία των νέων πινάκων και τρέχουμε τα ερωτήματα πάνω στους ήδη υπάρχοντες και μεγαλύτερους πίνακες. Η όλη διαδικασία είναι ακριβώς η ίδια με την προαναφερθείσα με τις εξής διαφορές: καταρχάς κατά τη δημιουργία των τελικών ερωτημάτων στη συνθήκη FROM δεν βάζουμε πλέον τους προσωρινούς πίνακες αλλά τους αρχικούς πίνακες με τα δεδομένα. Και κατά δεύτερον έχουμε παραλείψει να εκτελέσουμε τις συνθήκες WHERE οι οποίες στην

προηγούμενη προσέγγιση αποτελούν μέρος της δημιουργίας των προσωρινών πινάκων. Έτσι στην τεχνική αυτή, απλά στη συνθήκη WHERE των τελικών ερωτημάτων προσθέτουμε και το αντικείμενο WHERE του αρχικού ερωτήματος του χρήστη ώστε να είναι το ερώτημα ολοκληρωμένο.

Εκτέλεση MAP, ολοκλήρωση εκτέλεσης των ερωτημάτων-επιστροφή αποτελεσμάτων

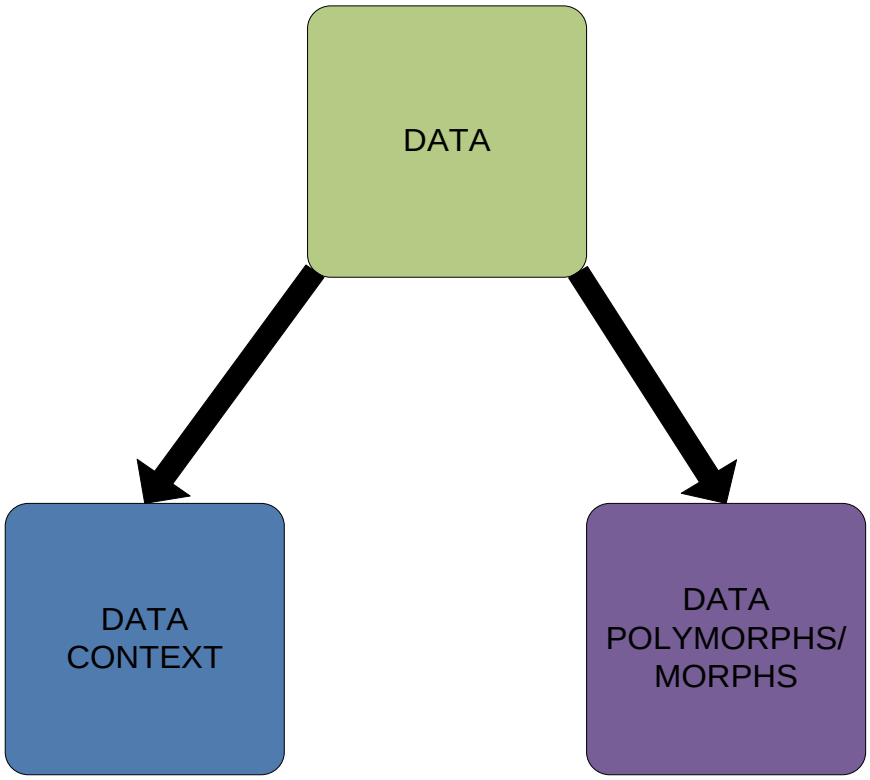
Σύμφωνα με όσα προαναφέρθηκαν είμαστε πλέον σε θέση, δοθέντος ενός ερωτήματος που αφορά κάποια N polymorphs να δημιουργήσουμε πολλαπλά σχεσιακά ερωτήματα και να τα εκτελέσουμε ανάμεσα σε N-άδες morphs που ανήκουν σε αυτά. Αφού το πραγματοποιήσουμε αυτό από κάθε ερώτημα δημιουργούμε έναν πίνακα στη βάση δεδομένων με τα αποτελέσματά του. Όμως κάθε τέτοιος πίνακας αφού έχει προκύψει από την εκτέλεση του αρχικού ερωτήματος για μια N-άδα morphs τα οποία έχουν οριστεί με το ίδιο context instance έστω X, αποτελεί τον πίνακα στον οποίο είναι αποθηκευμένα τα δεδομένα ενός morph που είναι ορισμένο για το context instance X. Όλα μαζί τα αποτελέσματα αποτελούν συνεπώς morphs ενός polymorph ορισμένο με βάση το context schema με το οποίο έχουν οριστεί τα polymorph στα οποία εκτελέστηκε αρχικά το ερώτημα. Στην περίπτωση που δεν έχουν οριστεί συνθήκες MAP το polymorph αυτό λοιπόν είναι και αυτό που τελικά επιστρέφεται ως αποτέλεσμα από την εκτέλεση του αρχικού ερωτήματος.

Σε διαφορετική περίπτωση εφόσον έχει οριστεί κάποια συνθήκη MAP ADD τότε απλά προσθέτουμε το δοθέν attribute στον πίνακα cat_context_attr του καταλόγου αλλά και στον πίνακα που κρατούνται τα δεδομένα για το context schema που εξετάζουμε και ακολούθως επιστρέφεται το polymorph ως αποτέλεσμα.

Στην περίπτωση όμως που έχει οριστεί κάποια συνθήκη MAP REMOVE το να αφαιρέσουμε το attribute από τους δύο αυτούς πίνακες δεν αρκεί. Και αυτό διότι μπορεί πλέον να έχουν δημιουργηθεί δύο η περισσότερες tuples με τα ίδια δεδομένα context οπότε τα morphs που ορίζονται με βάση αυτά θα έχουν τελικά τα ίδια context instances. Οπότε στην περίπτωση αυτή γίνεται έλεγχος για να διαπιστωθεί αν αυτό συνέβη από την αφαίρεση του/των attribute(s) και στη συνέχεια όσα morphs ορίζονται με βάση το ίδιο context instance ενώνονται σε ένα και τελικά επιστρέφεται το polymorph ως αποτέλεσμα.

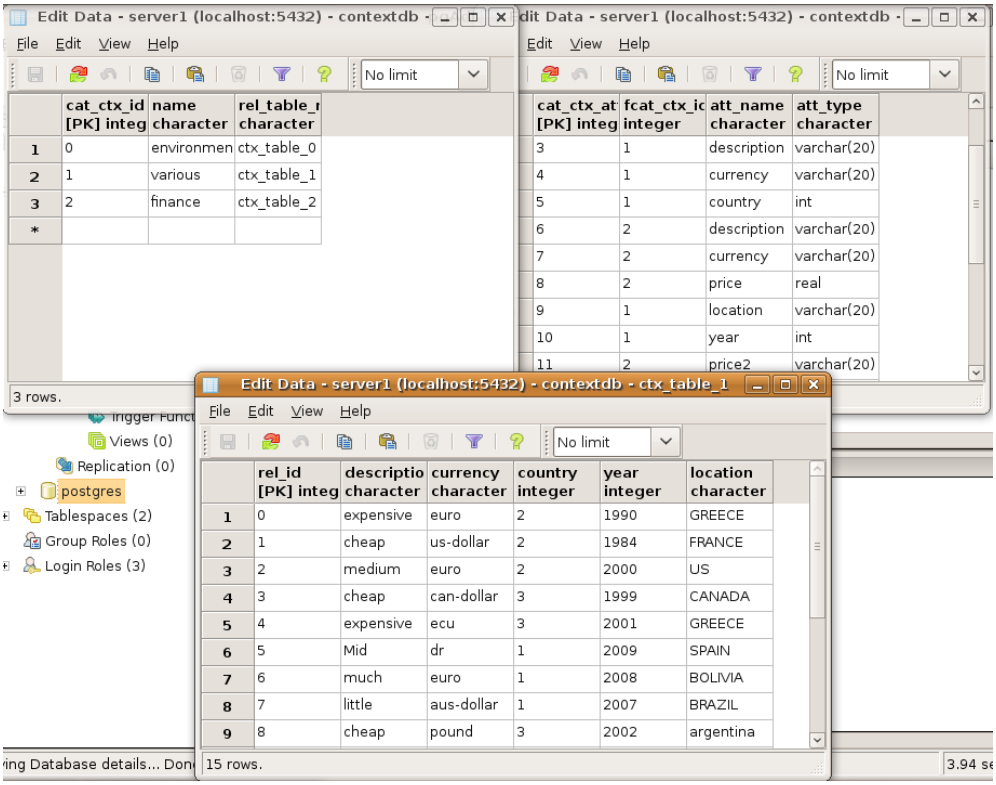
3.4.2 Υποσύστημα Διαχείρισης Δεδομένων

Το υποσύστημα διαχείρισης δεδομένων είναι υπεύθυνο για την εισαγωγή-διαγραφή-ενημέρωση (insert-delete-update) των πληροφοριών που περιλαμβάνονται στους πίνακες που ανήκουν στη βάση δεδομένων. Τα δεδομένα αυτά αφορούν είτε πληροφορίες σχετικά με το context είτε πληροφορίες σχετικά με polymorphs και morphs.



Σχήμα 3.6 Γενική λειτουργία υποσυστήματος διαχείρισης δεδομένων

Όσον αφορά τις πληροφορίες για το context, και για την πληρότητα των όσων αναφέρθηκαν παραπάνω, αρχικά πρέπει να αναφέρουμε ότι με κατάλληλη εντολή και εφόσον έχει ολοκληρωθεί επιτυχώς ο ορισμός κάποιου context schema το σύστημα δημιουργεί έναν πίνακα με όνομα ctx_table_XX όπου XX είναι το id του συγκεκριμένου context schema ενώ ταυτόχρονα ενημερώνει το attribute rel_table_talbe της tuple του πίνακα cat_context με το όνομα του νέου αυτού πίνακα. Ο πίνακας αυτός έχει ως στήλες τα attributes που έχουν ορισθεί στον πίνακα cat_context_attr για το context schema αυτό και μια επιπλέον πρώτη στήλη με ένα id το οποίο και αποτελεί το αναγνωριστικό των context instances του context schema αυτού. Στη συνέχεια δίνεται ένα screenshot όπου δείχνει την πράξη τα δεδομένα των ενός πίνακα για κάποιο context schema καθώς και τα δεδομένα των πινάκων του καταλόγου από τα οποία αυτός δημιουργήθηκε.

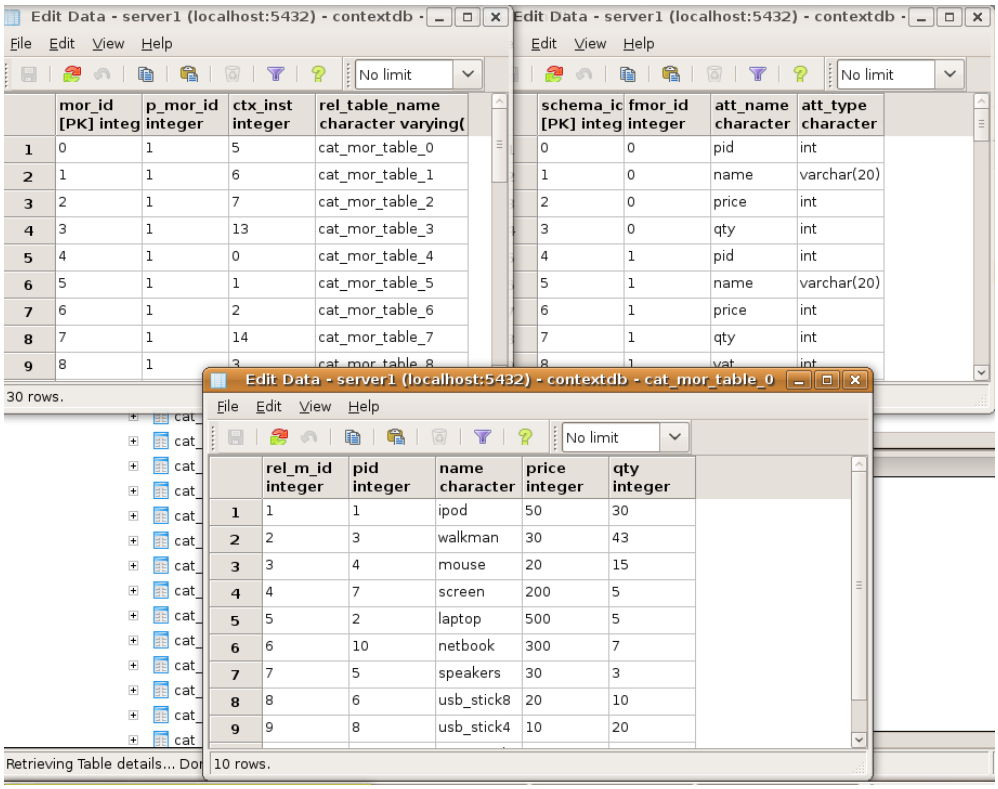


Σχήμα 3.7 Παράδειγμα περιεχομένου των πινάκων του καταλόγου και ενός πίνακα με δεδομένα context.

Όσον αφορά τη διαγραφή ενός context_instance αυτή γίνεται ελεύθερα με κατάλληλη εντολή με τη σημείωση ότι ενημερώνονται κατάλληλα οι πίνακες του καταλόγου που αφορούν τα morphs. Δηλαδή εφόσον ο χρήστης αποφασίσει τη διαγραφή ενός context instance τότε μπορεί ελεύθερα να το κάνει, όμως ταυτόχρονα σβήνεται το πεδίο ctx_inst του πίνακα cat_mor_table για όποια εγγραφή είχε οριστεί στο id αυτό. Σχετικά με τις ενημερώσεις ο χρήστης μπορεί ελεύθερα να τις πραγματοποιήσει, με τη σημείωση όμως ότι δεν μπορεί τελικά να δημιουργήσει μια tuple που να έχει ίδιες τιμές για όλα τα attributes του context schema με κάποια άλλη εντός του ίδιου schema. Τέλος όσον αφορά τις εισαγωγές, πάλι επιτρέπονται ελεύθερα με τον ίδιο περιορισμό όσον αφορά τη μοναδικότητα των context instances εντός του ίδιου context schema.

Η διαδικασία για τη διαχείριση των δεδομένων που αφορούν polymorphs και morphs είναι παρόμοια με αυτήν που αναφέρεται παραπάνω. Αρχικά και εφόσον ολοκληρωθεί με επιτυχία ο ορισμός ενός polymorph, των morph τα οποία το αποτελούν αλλά και των attributes που ανήκουν σε κάθε morph, το σύστημα δημιουργεί έναν πίνακα στο οποίο θα αποθηκεύονται από το σημείο αυτό και έπειτα τα δεδομένα που θα εισάγει ο χρήστης και ενημερώνει κατάλληλα το πεδίο rel_table_name του πίνακα cat_mor_table. Στην υλοποίηση Α ο πίνακας αυτός ονομάζεται cat_mor_table_XX όπου XX είναι το id του morph για το οποίο δημιουργήθηκε και για κάθε morph είναι ξεχωριστός, ενώ στην υλοποίηση Β το όνομα του

είναι pol_cat_mor_table_XX όπου XX είναι το id του polymorph για το οποίο δημιουργήθηκε και είναι ενιαίος για όλα τα morphs που ανήκουν σε κάποιο polymorph. Δηλαδή στην υλοποίηση Α υλοποιούμε πίνακες δεδομένων ανά morph ενώ στην υλοποίηση Β ανά polymorph. Στη συνέχεια δίνεται ένα screenshot όπου δείχνει την πράξη τα δεδομένα ενός πίνακα για κάποιο morph καθώς και τα δεδομένα των πινάκων του καταλόγου από τα οποία αυτός δημιουργήθηκε.



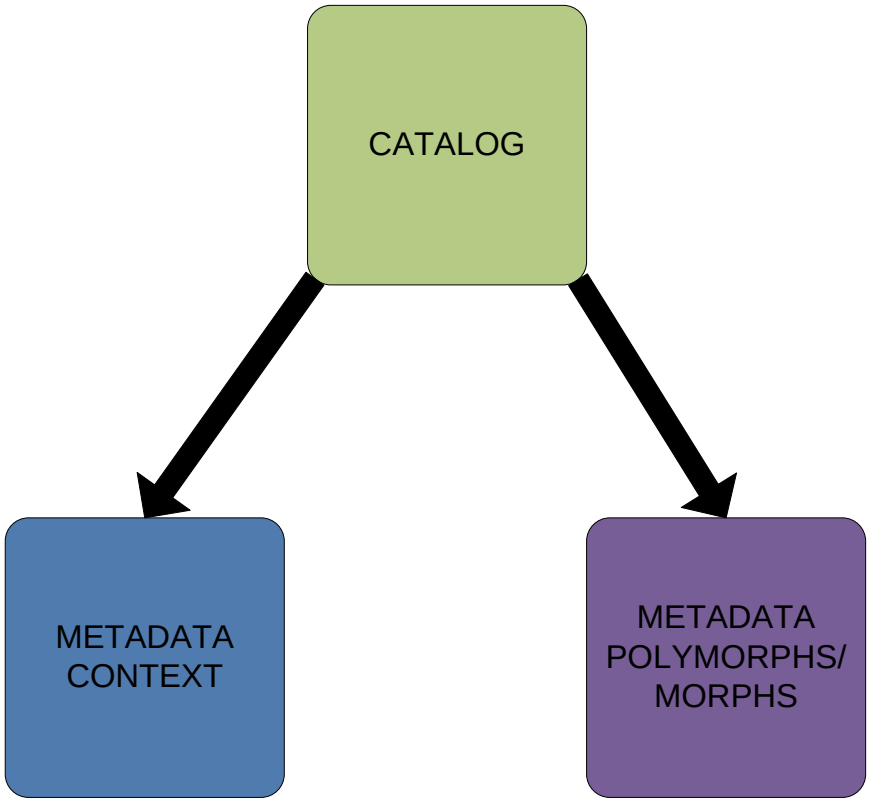
Σχήμα 3.8 Παράδειγμα περιεχομένου των πινάκων του καταλόγου και ενός πίνακα με δεδομένα ενός morph.

Όσον αφορά τη διαγραφή μιας tuple κάποιου morph, αυτή επιτρέπεται ελεύθερα στο χρήστη, όπως αντίστοιχα και οι εισαγωγές πάλι όμως με τον περιορισμό που έχει ήδη αναφερθεί σχετικά με τη μοναδικότητα των εγγραφών εντός του ίδιου πίνακα, ενώ το ίδιο ισχύει και για τις ενημερώσεις.

3.4.3 Υποσύστημα Καταλόγου

Ο κατάλογος μιας βάσης δεδομένων αποτελείται από metadata στα οποία αποθηκεύονται οι ορισμοί όλων των αντικειμένων της βάσης. Για το λόγο ότι στα αντικείμενα αυτά συμπεριλαμβάνονται τόσο οι πίνακες, οι προβολές επί αυτών, τα επιτρεπτά όρια ανάθεσης τιμών σε μεταβλητές όσο και τα ευρετήρια, οι πληροφορίες σχετικά με τους χρήστες κλπ

γίνεται αντιληπτός ο κομβικός ρόλος του υποσυστήματος αυτού, κάτι που άλλωστε επιβεβαιώνεται από το γεγονός ότι χρησιμοποιείται κατά κόρον στα σύγχρονα Συστήματα Διαχείρισης Βάσεων Δεδομένων. Κατ' επέκταση, για το σύστημα που περιγράφεται στην παρούσα διπλωματική εργασία είναι αρκετά προφανής η ανάγκη δημιουργίας μιας τέτοιας μονάδας λόγω των πολύπλοκων σχέσεων που εισάγει η έννοια του context. Στην κατεύθυνση αυτή υλοποιήσαμε τη συγκεκριμένη μονάδα η οποία διαχειρίζεται τόσο τα metadata που αφορούν το context όσο και εκείνα που αφορούν τα polymorphs.



Σχήμα 3.9 Γενική λειτουργία υποσυστήματος καταλόγου

Πριν προχωρήσουμε στην ανάλυση του τρόπου με τον οποίο γίνεται η διαχείριση των παραπάνω metadata πρέπει να αναφέρουμε ότι παρά το γεγονός ότι η αποθήκευση των polymorphs υλοποιήθηκε με διαφορετικούς τρόπους (υλοποιήσεις A,B) το υποσύστημα του καταλόγου που αφορά τη διαχείριση της πληροφορίας του context παρέμεινε αμετάβλητο-ενώ όπως είναι λογικό μεταβλήθηκε εκείνο που αφορά τη διαχείριση των metadata των polymorphs. Παρακάτω αναλύεται ο τρόπος υλοποίησης και λειτουργίας των δύο υποσυστημάτων του Καταλόγου.

Context

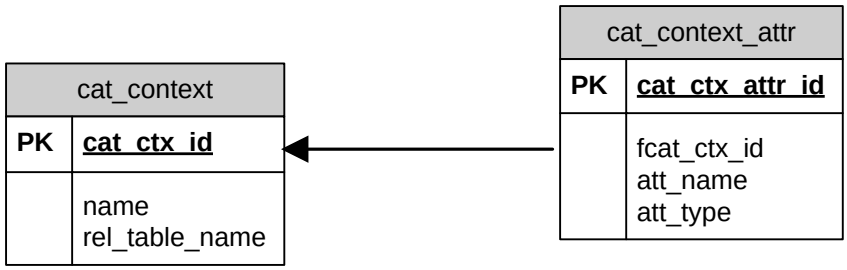
Για τη διαχείριση των metadata που αφορούν το context δημιουργήθηκαν δύο πίνακες, ο cat_context με σχήμα:

(cat_ctx_id integer (PK), name character varying(30), rel_table_name character varying(30)

και ο cat_context_attr με σχήμα:

(cat_ctx_att_id integer (PK),fcat_ctx_id integer(FK), att_name character varying(20), att_type character varying(20))

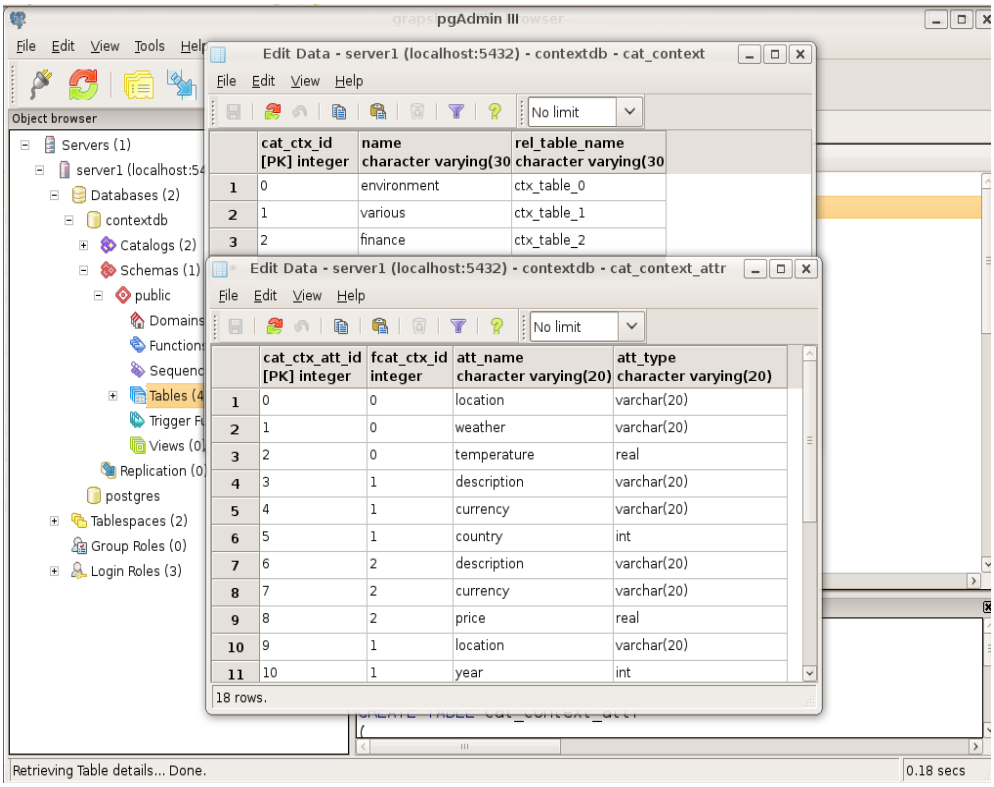
για τους οποίους ακολούθως δίνεται το σχεσιακό διάγραμμα:



Όπως γίνεται εύκολα αντιληπτό ο πίνακας cat_context αποθηκεύει πληροφορίες σχετικά με τα context schemas και πιο συγκεκριμένα το id (cat_ctx_id) τους (που αποτελεί μοναδικό χαρακτηριστικό-πρωτεύον κλειδί), το όνομα (name) τους και τον πίνακα (rel_table_name) της βάσης στον οποίο βρίσκονται τα instances του context schema αυτού ενώ μια tuple του δεύτερου πίνακα αντιπροσωπεύει το id (cat_ctx_attr_id) ενός attribute που ανήκει σε κάποιο context schema (fcat_ctx_id- εξωτερικό κλειδί) ενώ για αυτό έχει οριστεί όνομα (att_name) και τύπος (att_type).

Όσον αφορά τη διαδικασία που ακολουθείται για τη δημιουργία ενός νέου context schema αυτή έχει ως εξής: αρχικά καλείται μια μέθοδος η οποία εισάγει στον πίνακα cat_context μια tuple της μορφής (X,NULL,NULL) όπου X είναι ο αμέσως επόμενος διαθέσιμος ακέραιος (δηλαδή αν έχουν ήδη ορισθεί τα context schemas με id τα 0,1,2,3 τότε X=4) ο οποίος και θα αποτελεί το id του context schema αυτού και επιστρέφεται από το σύστημα μετά την επιτυχή εισαγωγή της προαναφερθείσας tuple. Στη δίνεται η δυνατότητα να οριστεί ένα όνομα για το context schema αυτό, το οποίο όμως δεν είναι υποχρεωτικό ενώ ακολούθως και η δυνατότητα να οριστούν τα attributes τα οποία θα περιλαμβάνονται στο context schema αυτό. Η διαδικασία εδώ είναι παρόμοια με τη δημιουργία ενός νέου context schema καθώς με την επιλογή να προστεθεί ένα νέο attribute στο context schema το σύστημα επιχειρεί να εισάγει στον πίνακα cat_context_attr μια tuple της μορφής (Y,X,temp,varchar(20)) όπου Y είναι το id του attribute (προκύπτει ομοίως με παραπάνω από τον πρώτο διαθέσιμο ακέραιο), X είναι το

id του context schema στο οποίο ανήκει το attribute αυτό και εισάγεται αυτόματα από το σύστημα, ενώ αρχικά το όνομα του attribute και ο τύπος του αρχικά καθορίζονται ως "temp" και "varchar(20)" αντίστοιχα. Στο σημείο αυτό θα πρέπει να παρατηρήσουμε αρχικά ότι ο λόγος που το id του context schema εισάγεται αυτόματα από το σύστημα και όχι μετά από κάποια ενέργεια του χρήστη είναι ότι κάθε context schema (και γενικότερα κάθε αντικείμενο είτε αυτό είναι context schema είτε morph, polymoprh, query κλπ όπως θα δούμε στη συνέχεια) αποτελεί ένα αυτόνομο αντικείμενο το οποίο έχει πρόσβαση στο συνολικό σύστημα αλλά μόνο για να διαχειρισθεί πληροφορίες που το αφορούν. Δηλαδή ναι μεν το context schema έχει πρόσβαση στους πίνακες του καταλόγου αλλά μπορεί να μεταβάλλει/διαγράψει/εισάγει μόνο εγγραφές που αφορούν το δικό του id. Και παρά το γεγονός ότι ενδεχομένως μια διαφορετική προσέγγιση, για παράδειγμα η δημιουργία ενός "context schema manager" ο οποίος θα ήταν υπεύθυνος για το χειρισμό όλων των πινάκων και όλων των tuples, να ήταν γρηγορότερη, επελέγη η αντικειμενοστραφής προσέγγιση όπως έχει ήδη αναφερθεί. Μια δεύτερη παρατήρηση που θα πρέπει να κάνουμε στο σημείο αυτό αφορά την αρχικοποίηση του ονόματος και τύπου του attribute. Η επιλογή αυτή- αν και ήσσονος σημασίας από σχεδιαστική άποψη, μιας και δεν προσθέτει καμιά πολυπλοκότητα στην υλοποίηση- έγινε αρχικά διότι ο τύπος varchar θεωρήθηκε ο πιο συνήθης και επίσης ώστε να δίνεται στο χρήστη η δυνατότητα να ορίσει εκ παραδρομής ένα παραπάνω attribute από αυτά που ήθελε και να μην εγερθεί κάποια εξαίρεση από την "αμέλειά" του αυτή. Η διαδικασία θα γίνει περισσότερο αντιληπτή σε επόμενη υποενότητα που αφορά το υποσύστημα που διαχειρίζεται τα δεδομένα (Data Manager). Παρακάτω δίνεται ένα screenshot των πινάκων που αναφέρονται στην ενότητα αυτή καθώς και των περιεχομένων τους.



Σχήμα 3.10 Περιεχόμενα πινάκων cat_context και cat_context_attr

Polymorphs/Morphs

Για τη διαχείριση των metadata που αφορούν τα polymorphs και τα morphs δημιουργήθηκαν τρεις πίνακες για τους οποίους πρέπει αρχικά να τονίσουμε ότι είναι οι ίδιοι ανεξάρτητα από τη διαφορετική υλοποίηση της αποθήκευσης των polymorphs. Όπως αναφέρθηκε και σε προηγούμενη υποενότητα ο διαφορετικός τρόπος αποθήκευσης των polymorphs μας οδήγησε στο να χειριζόμαστε με διαφορετικό τρόπο τα metadata που τα αφορούν, όμως, αυτό δεν σημαίνει απαραίτητα ότι άλλαξε στο ελάχιστο η δομή των πινάκων που χρησιμοποιούνται στο συγκεκριμένο μέρος του καταλόγου όσο ότι άλλαξε ο τρόπος διαχείρισης της πληροφορίας που αυτοί περιλαμβάνουν. Κατά συνέπεια όσον αφορά τους πίνακες που δημιουργήθηκαν στο υποσύστημα αυτό, αρχικά έχουμε τον πίνακα pol_cat_mor_table με σχήμα:

(pol_mor_id integer (PK), pol_name character varying(20), ctx_sch_id integer)

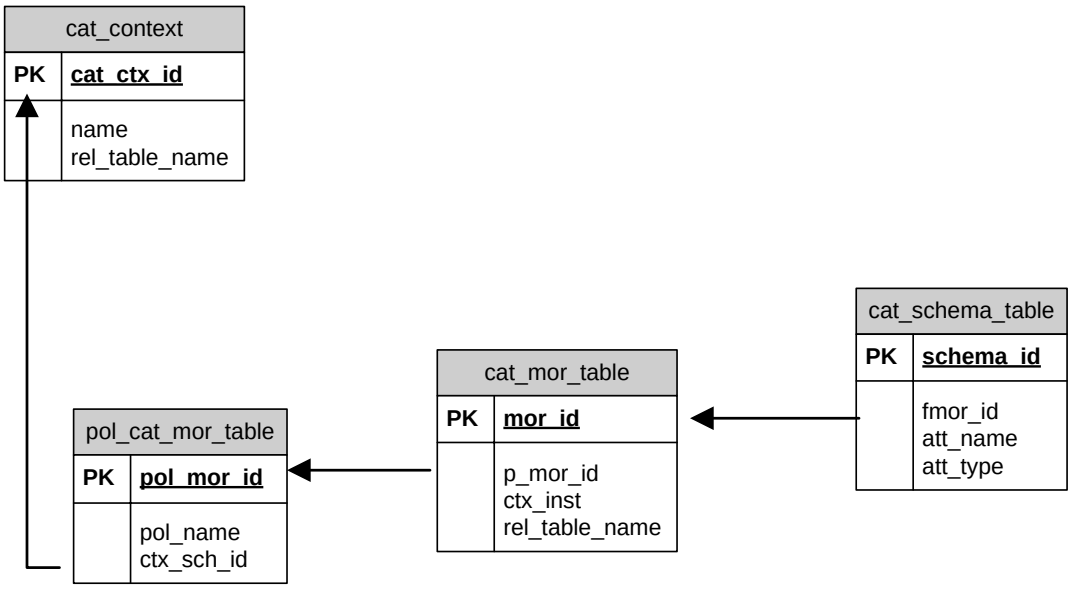
τον πίνακα cat_mor_table με σχήμα:

(mor_id integer (PK), p_mor_id integer(FK), ctx_inst integer, rel_table_name character varying(30))

και τον πίνακα cat_schema_table με σχήμα:

(schema_id integer (PK), fmor_id integer(FK), att_name character varying(20), att_type character varying(20))

για τους οποίους ακολούθως δίνεται το σχεσιακό διάγραμμα:



Όπως γίνεται εύκολα αντιληπτό ο πίνακας pol_cat_mor_table περιέχει τα polymorphs, ο πίνακας cat_mor_table τα morphs καθένα από τα οποία ανήκει σε κάποιο polymorph και ο πίνακας cat_schema_table περιέχει τα attributes καθένα εκ των οποίων ανήκει σε κάποιο morph. Πιο αναλυτικά ο πίνακας pol_cat_mor_table περιλαμβάνει για κάθε polymorph το id του (pol_mor_id), το όνομά του (pol_name) και το context schema με βάση το οποίο ορίζεται (ctx_sch_id). Ο πίνακας cat_mor_table περιλαμβάνει για κάθε morph, το id του (mor_id), το polymorph στο οποίο ανήκει (p_mor_id), το context instance του context schema με βάση το οποίο ορίζεται (ctx_inst) και το όνομα του πίνακα στον οποίο βρίσκονται αποθηκευμένα τα δεδομένα του (rel_table_name).

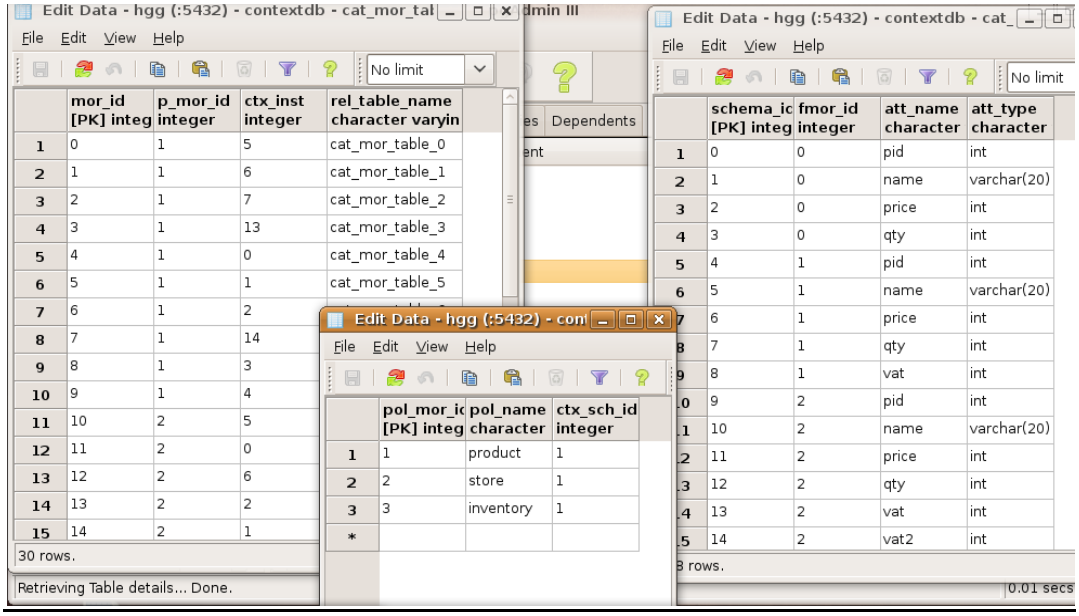
Η διαδικασία που ακολουθείται κατά τη δημιουργία ενός νέου polymorph είναι η εξής: με κατάλληλη εντολή εισάγεται στον πίνακα pol_cat_mor_table μια tuple της μορφής (X,NULL, NULL) όπου X κατ' αντιστοιχίαν με τη δημιουργία ενός νέου context schema, είναι ο πρώτος διαθέσιμος ακέραιος ο οποίος από το σημείο αυτό και έπειτα αποτελεί το μοναδικό αναγνωριστικό του polymorph αυτού και επιστρέφεται από το σύστημα. Στη συνέχεια τίθεται για το polymorph αυτό ένα όνομα και το context schema με βάση το οποίο ορίζεται. Αυτό δεν του επιβάλλεται από το σύστημα, ωστόσο είναι απαραίτητο για τα στάδια της επεξεργασίας που θα ακολουθήσουν και αποτελούν το κομμάτι της μετατροπής των ερωτημάτων καθώς στο σημείο εκείνο ο χρήστης που θέτει ερωτήματα, όπως είναι λογικό δε γνωρίζει id- αφού αυτά αποτελούν τον τρόπο για το διαχωρισμό των polymorphs εντός του συστήματος- παρά μόνο ονόματα. Ακόμα, και ο ορισμός του context schema βρίσκεται στην ευχέρεια του χρήστη και δεν είναι απαραίτητος στο σημείο αυτό, πλην όμως εφόσον τεθεί κάποιο ερώτημα

που αφορά κάποιο polymorph για το οποίο δεν έχει οριστεί το context schema με το οποίο ορίζεται είναι εύλογο το polymorph αυτό τελικά να μη ληφθεί υπ' όψιν κατά τη διάρκεια επεξεργασίας και μετατροπής του ερωτήματος αυτού.

Στη συνέχεια και αφού έχει ολοκληρωθεί επιτυχώς η δημιουργία του polymorph δίνεται η δυνατότητα να δημιουργηθούν ένα πλήθος από morphs τα οποία θα το αποτελούν. Κατά συνέπεια, στον πίνακα cat_mor_table με τη δημιουργία ενός morph εισάγεται μια tuple της μορφής (Y,X,NULL,NULL) όπου Y είναι ο πρώτος διαθέσιμος ακέραιος ο οποίος από το σημείο αυτό και έπειτα αποτελεί το μοναδικό αναγνωριστικό του morph αυτού και επιστρέφεται από το σύστημα και X είναι το id του polymorph στο οποίο ανήκει το morph και εισάγεται αυτόματα. Στη συνέχεια θα πρέπει να οριστεί το context instance του context schema με το οποίο έχει οριστεί το polymorph στο οποίο ανήκει με τη σημείωση ότι καταρχάς το instance αυτό θα πρέπει να υπάρχει και κατά δεύτερον να μην χρησιμοποιείται από κάποιο άλλο morph του ίδιου polymorph. Το πεδίο rel_table_name συμπληρώνεται αυτόματα από το σύστημα εφόσον έχει ολοκληρωθεί επιτυχώς ο ορισμός του morph όμως αυτό είναι κάτι που θα αναφερθεί αναλυτικά σε επόμενη υποενότητα που θα αφορά το υποσύστημα διαχείρισης των δεδομένων.

Ακολούθως και αφού έχει δημιουργηθεί επιτυχώς κάποιο morph, ορίζονται τα attributes εκείνα που θα περιλαμβάνονται στο morph αυτό. Η διαδικασία είναι παρόμοια με όσα έχουν προαναφερθεί και αφορά την εισαγωγή στον πίνακα cat_schema_table μιας tuple της μορφής (Z,Y,temp,varchar(20)) όπου Z είναι ο πρώτος διαθέσιμος ακέραιος ο οποίος από το σημείο αυτό και έπειτα αποτελεί το μοναδικό αναγνωριστικό του attribute αυτού και επιστρέφεται από το σύστημα, Y είναι το id του morph στο οποίο ανήκει ενώ όσον αφορά το όνομα και τον τύπο του επιλέχθηκαν οι αρχικές τιμές temp και varchar(20) οι οποίες προφανώς μπορούν να αλλάξουν κατά βούληση του χρήστη.

Στη συνέχεια δίνεται ένα screenshot των τριών προαναφερθέντων πινάκων και των περιεχομένων τους.



Σχήμα 3.11 Περιεχόμενα πινάκων cat_mor_table, cat_schema_table, pol_cat_mor_table

3.4.4 Υποσύστημα Επικοινωνίας με το Σχεσιακό ΣΔΒΔ

Το κομμάτι αυτό του συστήματος είναι υπεύθυνο για την επικοινωνία με τη Βάση Δεδομένων (POSTGRES), την οποία χρησιμοποιούμε ως physical layer. Είναι σε θέση να ανοίγει και να κλείνει συνδέσεις (αντίστοιχα transactions), να θέτει ερωτήματα και να δέχεται τα αποτελέσματα. Χρησιμοποιείται κατά κόρον από τα επιμέρους υποσυστήματα που βρίσκονται σε υψηλότερο επίπεδο στο διάγραμμα που δίνεται παραπάνω (σχήμα 3.1) χωρίς φυσικά ο τρόπος λειτουργίας του να είναι γνωστός σε αυτά. Αυτό, όπως και κάθε άλλο κομμάτι του συστήματος που επιτελεί κάποιες λειτουργίες χρησιμοποιείται ως «μαύρο κουτί» από τα υπόλοιπα.

4

Υλοποίηση

Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση του συνολικού συστήματος το οποίο πραγματεύεται η παρούσα διπλωματική εργασία μαζί με όλα τα υποσυστήματα που το απαρτίζουν. Αρχικά στην παράγραφο 4.1 γίνεται αναφορά στην πλατφόρμα και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν ενώ στη συνέχεια στην παράγραφο 4.2 παρουσιάζονται αναλυτικά οι λεπτομέρειες υλοποίησης των σημαντικότερων υποσυστημάτων.

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Το υποσύστημα έχει αναπτυχθεί με τη γλώσσα προγραμματισμού ANSI C++ [Str97] και τρέχει σε πλατφόρμα Linux (πυρήνας 2.6.25) και συγκεκριμένα σε διανομή Ubuntu 8.04 [UBU09]. Ο database server που τρέχει στο κατώτερο επίπεδο του συστήματος είναι PostgreSql 8.3.8. Η επιλογή τόσο της γλώσσας προγραμματισμού και της γενικότερης πλατφόρμας όσο και του λειτουργικού συστήματος έγινε καταρχάς με κριτήριο την ταχύτερη απόδοση του συστήματος αλλά και την στο μέτρο του δυνατού απλούστερη υλοποίηση ενώ σημαντικό ρόλο έπαιξε και το γεγονός ότι όλα τα εργαλεία που χρησιμοποιήθηκαν ανήκουν στην κατηγορία του ελεύθερου λογισμικού.

Συνολικά το υποσύστημα εκτείνεται σε περισσότερες από 20.000 γραμμές κώδικα για κάθε υλοποίηση, συμπεριλαμβανομένων των κλάσεων που χρησιμοποιήθηκαν για τις δοκιμές και τους ελέγχους. Συνολικά δηλαδή για τις δύο υλοποιήσεις χρειάστηκαν περισσότερες από 40.000 γραμμές κώδικα.

Για την ανάπτυξη χρησιμοποιήθηκε ο μεταγλωττιστής g++ (4.3.2) της GNU ενώ ως περιβάλλον ανάπτυξης χρησιμοποιήθηκε το Code::Blocks. Όσον αφορά την επικοινωνία με τον server της PostgreSql αυτή κατέστη δυνατή με τη χρήση της βιβλιοθήκης libpqxx 3.0.2 ενώ τέλος, έχει γίνει εκτενής χρήση της στάνταρτ βιβλιοθήκης προτύπων της C++ (standard template library STL) [STL94] σε όλο τον κώδικα της μονάδας.

4.2 Λεπτομέρειες υλοποίησης

Στην ενότητα αυτή παρουσιάζονται οι λεπτομέρειες υλοποίησης των επιμέρους υποσυστημάτων που απαρτίζουν το σύστημα μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων το οποίο αποτελεί το αντικείμενο της παρούσας διπλωματικής εργασίας. Συνεπώς αναλύονται οι λεπτομέρειες υλοποίησης των υποσυστημάτων του Επιπέδου Διεπαφής με το ΣΔΒΔ, του Καταλόγου, του χειρισμού των δεδομένων, της αναπαράστασης των ερωτημάτων και τελικά της μετατροπής και εκτέλεσης τους. Στην πρώτη παράγραφο αναφέρεται η ιεραρχία εξαιρέσεων που δημιουργήθηκε με στόχο την αποτελεσματική διαχείριση των λαθών που προκύπτουν κατά τη λειτουργία του συστήματος ενώ στη συνέχεια εξετάζοντας το σύστημα από κάτω προς το πάνω περιγράφονται τα επιμέρους υποσυστήματα που προαναφέρθηκαν.

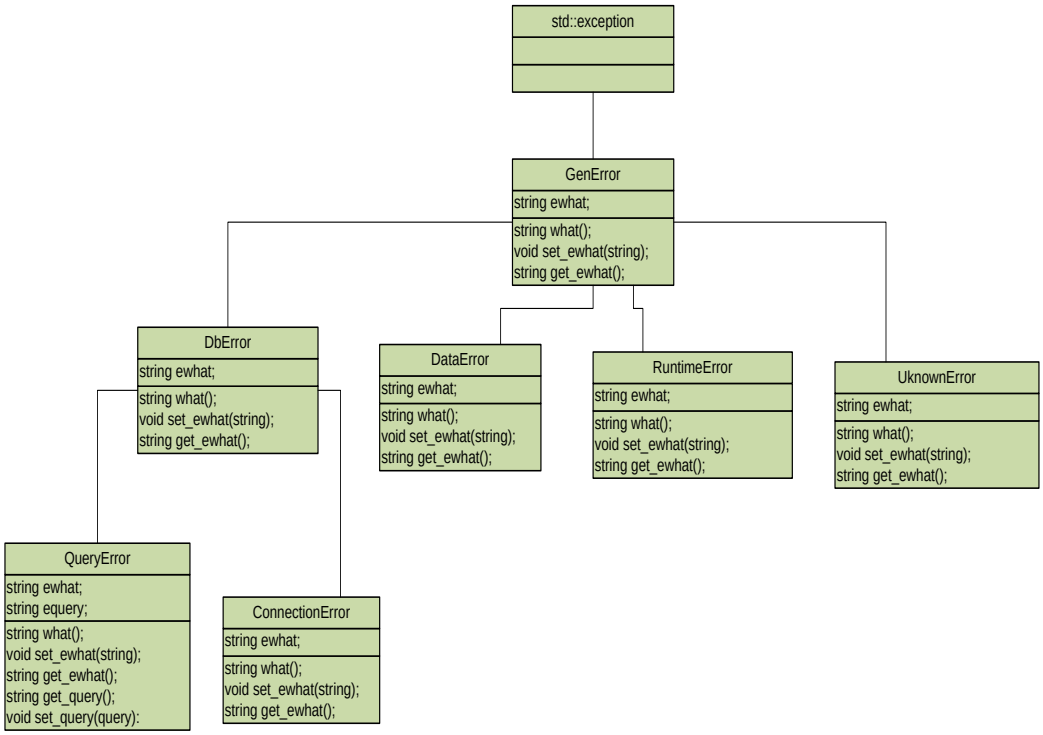
Μεγάλο μέρος της ενότητας αποτελείται από διαγράμματα κλάσεων, κάποια εκ των οποίων επαναλαμβάνονται όταν κρίνεται απαραίτητο να περιγραφεί η σχέση ανάμεσα σε δύο κλάσεις. Πέρα από τα διαγράμματα κλάσεων, παρουσιάζονται τα attributes κάθε κλάσης αλλά και οι σημαντικότερες μέθοδοι που δημιουργήθηκαν ακολουθούμενες από μια σύντομη περιγραφή της λειτουργίας τους. Στο σημείο αυτό πρέπει να σημειώσουμε ότι έχει γίνει εκτεταμένη χρήση βοηθητικών κλάσεων οι οποίες δεν κρίνεται σκόπιμο να παρουσιαστούν όλες αναλυτικά καθώς ο ρόλος τους είναι να παρέχουν κάποια προγραμματιστική ευελιξία και δε βοηθούν ιδιαίτερα στην κατανόηση του τρόπου υλοποίησης του συστήματος. Ακόμα για κάθε κλάση έχει υλοποιηθεί μια κλάση ελέγχου και ούτε οι κλάσεις αυτές απεικονίζονται στα παρακάτω διαγράμματα καθώς εξυπηρετούν τον έλεγχο του κάθε υποσυστήματος και δεν αποτελούν κομμάτι της λειτουργίας του.

4.2.1 Ιεραρχία Εξαιρέσεων

Στα πλαίσια του αποτελεσματικού χειρισμού των λαθών που προκύπτουν κατά τη λειτουργία του συστήματος μετατροπής και εκτέλεσης των ερωτημάτων υλοποιήθηκε μια ιεραρχία εξαιρέσεων. Η βασική κλάση που βρίσκεται και στο ανώτερο σημείο της ιεραρχίας είναι η `std::exception` η οποία και βρίσκεται στις βιβλιοθήκες της C++ ενώ οι υπόλοιπες κλάσεις κληρονομούν από αυτήν. Κάθε κλάση που πετάει κάποια εξαίρεση ορίζει και το αντίστοιχο μήνυμα λάθους που θα εμφανιστεί. Στο σχήμα 4.1 δίνεται το διάγραμμα της ιεραρχίας των κλάσεων ενώ στη συνέχεια αναφέρονται τα attributes αλλά και οι βασικές μέθοδοι των παραγόμενων υποκλάσεων.

Τα βασικά attributes που χρησιμοποιούνται στις κλάσεις είναι τα ακόλουθα:

- `string ewhat`: Η μεταβλητή αυτή περιέχει το ακριβές μήνυμα λάθους που προκύπτει από την αντίστοιχη κλάση που πετάει την εξαίρεση. Δεν πρόκειται για το ίδιο μήνυμα με αυτό της what() που προέρχεται από την std::exception.
- `string equery`: Εφόσον η εξαίρεση προκύψει μετά από κάποιο ερώτημα που έθεσε ο χρήστης, τότε εκτός από το χειρισμό της, αποθηκεύεται και το ερώτημα που την προκάλεσε.



Σχήμα 4.1 Διάγραμμα ιεραρχίας εξαιρέσεων

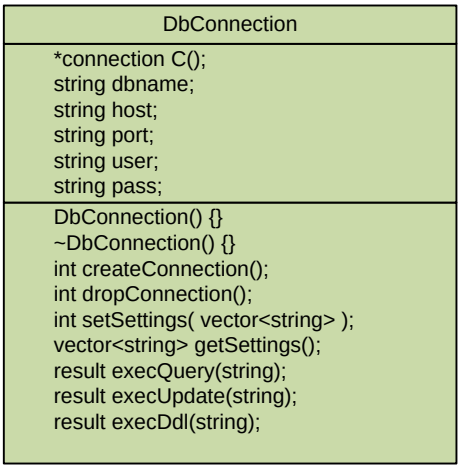
Οι βασικές μέθοδοι που υλοποιούνται στις κλάσεις είναι οι ακόλουθες:

- `string what()`: Προέρχεται από την std::exception και περιέχει το μήνυμα λάθους
- `string get_ewhat()`: Επιστρέφει το μήνυμα λάθους που τέθηκε από την αντίστοιχη κλάση που πέταξε την εξαίρεση.
- `void set_ewhat(string)`: Αποθηκεύεται στη μεταβλητή ewhat το ακριβές μήνυμα λάθους από την κλάση που πέταξε την εξαίρεση
- `string get_query()`: Σε περιπτώσεις εξαιρέσεων που προέκυψαν λόγω κάποιουερωτήματος, η μέθοδος αυτή επιστρέφει το ερώτημα αυτό.

- `void set_query(query):` Αποθηκεύεται το ερώτημα που προκάλεσε την εξαίρεση από την κλάση από την οποία προήλθε.

4.2.2 Υποσύστημα Επικοινωνίας με το Σχεσιακό ΣΔΒΔ

Η κλάση που υλοποιεί το υποσύστημα αυτό είναι η `DbConnection` το διάγραμμα της οποίας δίνεται στο παρακάτω σχήμα (4.2).



Σχήμα 4.2 Διάγραμμα κλάσης DbConnection

Τα attributes της κλάσης αυτής είναι τα ακόλουθα:

- connection* conn: Δείκτης σε ένα αντικείμενο τύπου connection που ορίζεται στη βιβλιοθήκη libpqxx. Μέσω του αντικειμένου αυτού επιτυγχάνεται η σύνδεση με τη βάση δεδομένων.
- string dbname: Το όνομα της προς σύνδεσης βάσης δεδομένων.
- string host: Το όνομα του host του server της βάσης δεδομένων.
- string port: Η θύρα στην οποία θα γίνει η σύνδεση στο server της βάσης δεδομένων.
- string user: Το όνομα του χρήστη για την σύνδεση.
- string pass: Ο κωδικός του χρήστη.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `int createConnection():` Επιστρέφει έναν ακέραιο ανάλογα με τον αν η σύνδεση με τη βάση δεδομένων ήταν ή όχι επιτυχής.

- `int dropConnection()`: Επιστρέφει έναν ακέραιο ανάλογα με τον αν η αποσύνδεση από τη βάση δεδομένων ήταν ή όχι επιτυχής.
- `result execQuery(string)`: Επιστρέφει ένα αντικείμενο τύπου result που ορίζεται στη βιβλιοθήκη libpqxx με το αποτέλεσμα του ερωτήματος που έδωσε ο χρήστης ως παράμετρο.
- `result execUpdate(string)`: Επιστρέφει ένα αντικείμενο τύπου result με το αποτέλεσμα του ερωτήματος που έδωσε ο χρήστης ως παράμετρο και αφορά την τροποποίηση κάποιων εγγραφών στη βάση δεδομένων.
- `result execDdl(string)`: Επιστρέφει ένα αντικείμενο τύπου result με το αποτέλεσμα του ερωτήματος που έδωσε ο χρήστης ως παράμετρο και αφορά κάποια DDL ή DML εντολή.

4.2.3 Υποσύστημα του Καταλόγου

Οι κλάσεις που υλοποιούν το υποσύστημα του Καταλόγου είναι οι: `ContextSchema` και `Context` που διαχειρίζονται τα metadata σχετικά με το context και οι `Morphs` και `Poly_morph` που διαχειρίζονται τα δεδομένα σχετικά με τα morphs και τα polymorphs αντίστοιχα. Τα διαγράμματα των κλάσεων αυτών δίνονται παρακάτω στο σχήμα 4.3.

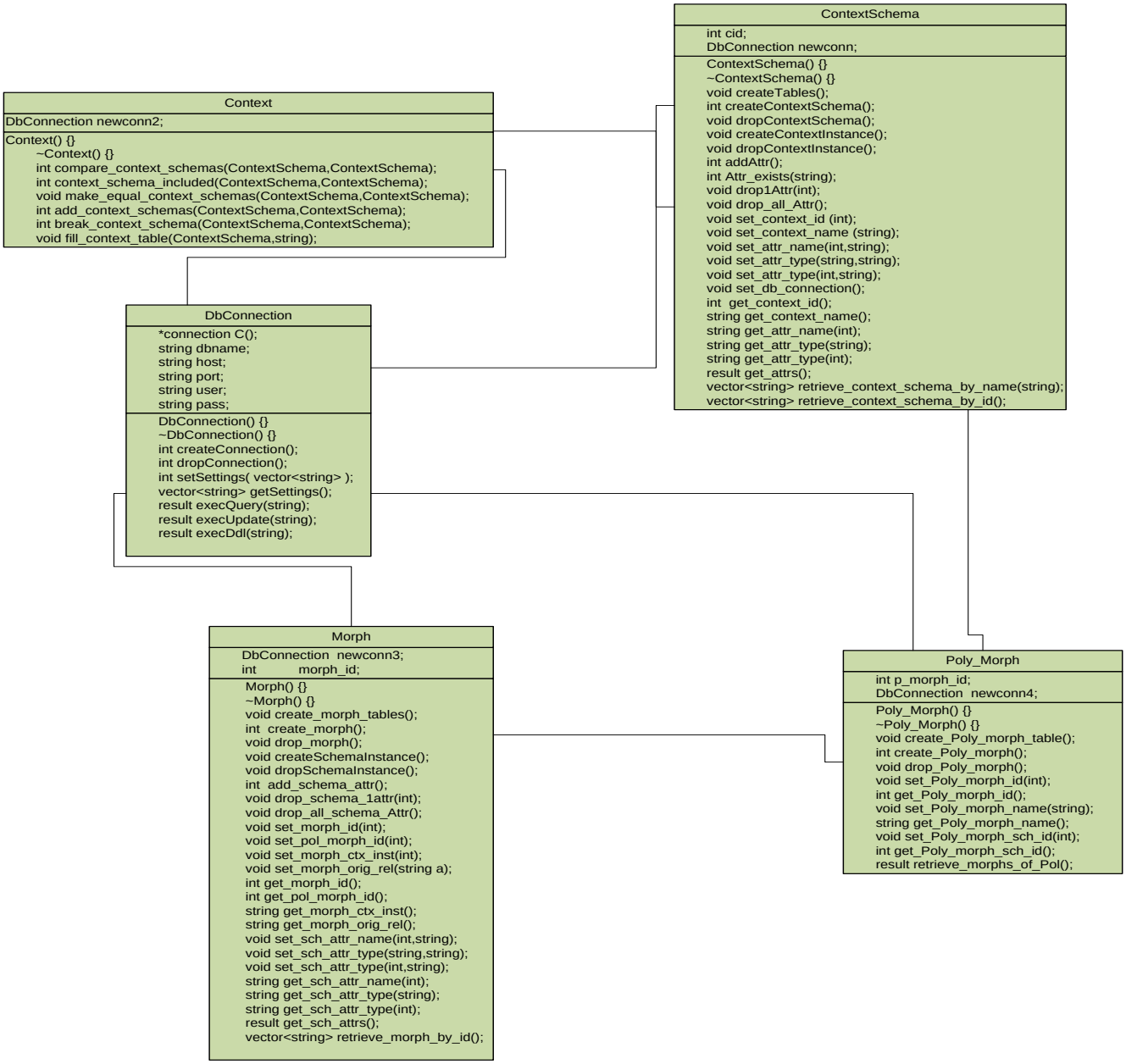
Τα attributes της κλάσης `ContextSchema` είναι τα ακόλουθα:

- `int cid`: Αποτελεί το μοναδικό χαρακτηριστικό κάθε αντικειμένου τύπου `ContextSchema` που δημιουργείται
- `DbConnection newconn2`: Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `void createTables()`: Ελέγχει αν έχουν δημιουργηθεί οι πίνακες στη βάση δεδομένων που αφορούν το χειρισμό των metadata του Context και αν δεν έχουν δημιουργηθεί τους δημιουργεί.
- `int createContextSchema()`: Δημιουργεί ένα νέο ContextSchema στον κατάλογο και επιστρέφει το id του.
- `void dropContextSchema()`: Διαγράφει το συγκεκριμένο ContextSchema από τον κατάλογο.

- `void createContextInstance()`: Ελέγχει το ContextSchema αν έχει οριστεί πλήρως και εφόσον αυτό συμβαίνει δημιουργεί έναν πίνακα στη βάση δεδομένων στον οποίο σε επόμενο στάδιο θα είναι δυνατόν να αποθηκεύονται τα context instances.
- `void dropContextInstance()`: Διαγράφει (εφόσον υπάρχει) τον πίνακα της βάσης δεδομένων στον οποίο αποθηκεύονται τα δεδομένα του συγκεκριμένου ContextSchema
- `int addAttr()`: Προσθέτει στο ContextSchema ένα attribute και εφόσον επιτύχει επιστρέφει το id του.
- `void drop1Attr(int)`: Διαγράφει από το ContextSchema το attribute του οποίου το id δίνεται ως παράμετρο.
- `void drop_all_Attr()`: Διαγράφει όλα τα attributes για το συγκεκριμένο ContextSchema
- `vector<string> retrieve_context_schema_by_id()` : Επιστρέφει σε μορφή vector<string> όλα τα attributes του συγκεκριμένου ContextSchema.



Σχήμα 4.3 Διάγραμμα κλάσεων καταλόγου

Το μοναδικό attribute της κλάσης `ContextSchema` είναι το ακόλουθο:

- `DbConnection newconn2`: Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `int compare_context_schemas(ContextSchema,ContextSchema):` Ελέγχει αν τα context schemas που δίνονται ως παράμετροι είναι ίσα δηλαδή περιέχουν τα ίδια attributes.
- `int context_schema_included(ContextSchema,ContextSchema):` Για τα δύο context schemas που δίνονται ως παράμετρο, ελέγχει αν τα attributes του δεύτερου αποτελούν υποσύνολο εκείνων του πρώτου.
- `void make_equal_context_schemas(ContextSchema,ContextSchema):` Παίρνοντας ως παραμέτρους δύο context schemas εκ των οποίων το δεύτερο περιέχεται στο πρώτο, προσθέτει στο δεύτερο όσα attributes του λείπουν ώστε να γίνει ίσο με το πρώτο.
- `int add_context_schemas(ContextSchema,ContextSchema):` Παίρνοντας ως παραμέτρους δύο context schemas προσθέτει το δεύτερο στο πρώτο.
- `int break_context_schema(ContextSchema,ContextSchema):` Παίρνοντας ως παραμέτρους δύο context schemas αφαιρεί το δεύτερο από το πρώτο.

Τα attributes της κλάσης `Morphs` είναι τα ακόλουθα:

- `int morph_id`: Αποτελεί το id του συγκεκριμένου morph στον κατάλογο.
- `DbConnection newconn3`: Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `void create_morph_tables():` Ελέγχει αν έχουν δημιουργηθεί οι πίνακες στη βάση δεδομένων που αφορούν το χειρισμό των metadata των morphs και αν δεν έχουν δημιουργηθεί τους δημιουργεί.
- `int create_morph():` Δημιουργεί ένα νέο morph στον κατάλογο και επιστρέφει το id του.
- `void drop_morph():` Διαγράφει το συγκεκριμένο morph από τον κατάλογο.
- `void createSchemaInstance():` Ελέγχει το morph αν έχει οριστεί πλήρως και εφόσον αυτό συμβαίνει δημιουργεί έναν πίνακα στη βάση δεδομένων στον οποίο σε επόμενο στάδιο θα είναι δυνατόν να αποθηκεύονται τα δεδομένα του.
- `void dropSchemaInstance():`Διαγράφει (εφόσον υπάρχει) τον πίνακα της βάσης δεδομένων στον οποίο αποθηκεύονται τα δεδομένα του συγκεκριμένου morph
- `int add_schema_attr():`Προσθέτει στο morph ένα attribute και εφόσον επιτύχει επιστρέφει το id του.

- `void drop_schema_1attr(int)`: Διαγράφει από το morph το attribute του οποίου το id δίνεται ως παράμετρο.
- `void drop_all_schema_Attr()`: Διαγράφει όλα τα attributes για το συγκεκριμένο morph
- `vector<string> retrieve_morph_by_id()`:Επιστρέφει σε μορφή vector<string> όλα τα attributes του συγκεκριμένου morph.

Τα attributes της κλάσης `Poly_morph` είναι τα ακόλουθα:

- `int p_morph_id`: Αποτελεί το id του συγκεκριμένου polymorph στον κατάλογο.
- `DbConnection newconn4`: Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `void create_Poly_morph_table()`:Ελέγχει αν έχουν δημιουργηθεί οι πίνακες στη βάση δεδομένων που αφορούν το χειρισμό των metadata των polymorphs και αν δεν έχουν δημιουργηθεί τους δημιουργεί.
- `int create_Poly_morph()`: Δημιουργεί ένα νέο polymorph στον κατάλογο και επιστρέφει το id του.
- `void drop_Poly_morph()`: Διαγράφει το συγκεκριμένο polymorph από τον κατάλογο.
- `result retrieve_morphs_of_Pol()`: Επιστρέφει σε μορφή αντικειμένου result τα morphs που ανήκουν στο συγκεκριμένο polymorph.

4.2.4 Υποσύστημα Διαχείρισης Δεδομένων

Οι κλάσεις που υλοποιούν το υποσύστημα διαχείρισης των δεδομένων είναι οι εξής: η ContextData που διαχειρίζεται τα δεδομένα σχετικά με το context και η MorphsData που διαχειρίζεται τα δεδομένα σχετικά με το περιεχόμενο των morphs. Παρακάτω δίνεται το διάγραμμα των κλάσεων αυτών στο σχήμα 4.4.

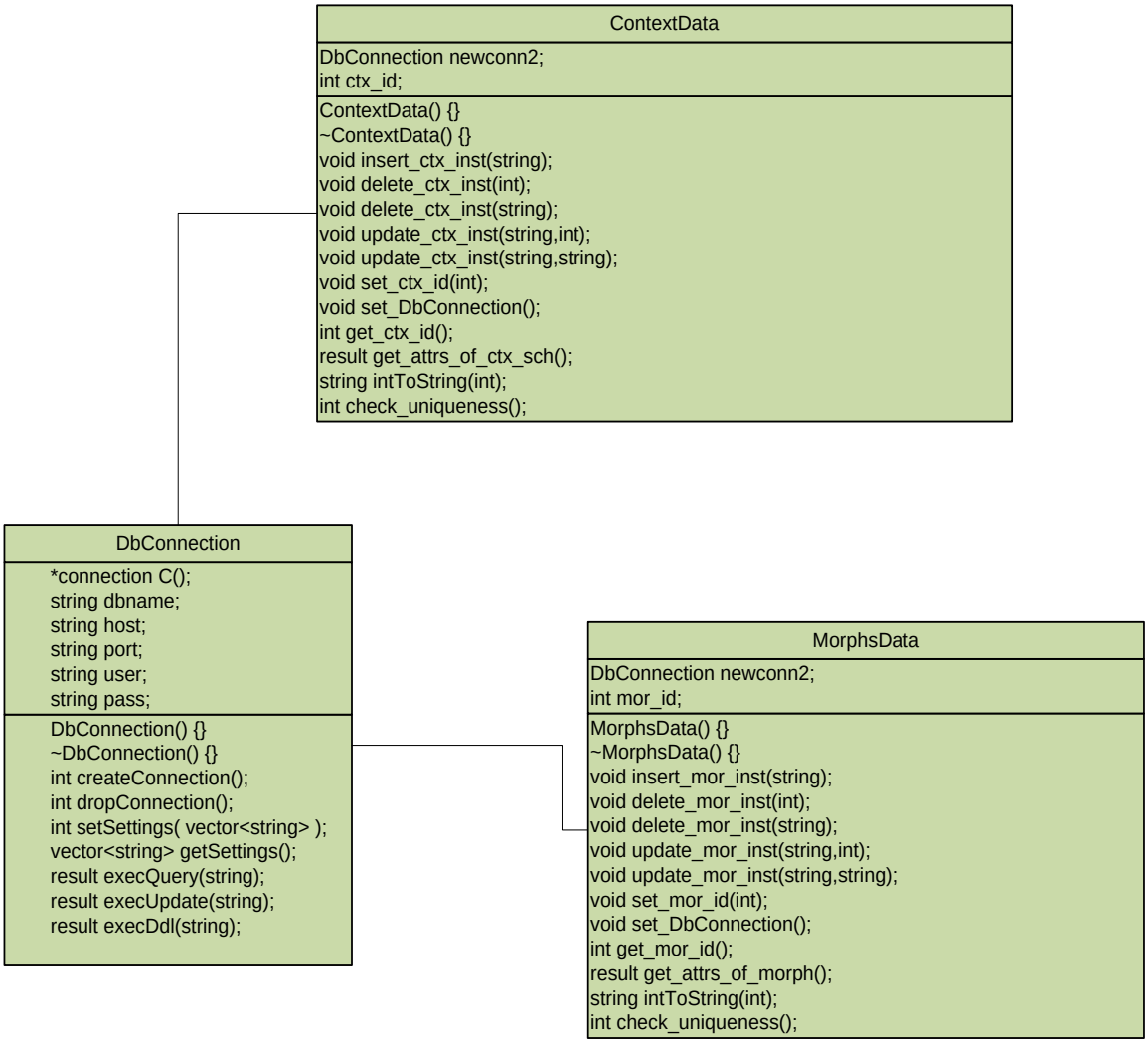
Τα attributes της κλάσης ContextData είναι τα ακόλουθα:

- `int ctx_id`: Αποτελεί το μοναδικό χαρακτηριστικό κάθε αντικειμένου τύπου `ContextData` που δημιουργείται

- `DbConnection newconn2:` Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `void insert_ctx_inst(string):` Εκτελεί τις εντολές για την εισαγωγή νέων context instances
- `void delete_ctx_inst(int):` Διαγράφει υπάρχοντα context instances.
- `void update_ctx_inst(string,int):` Ενημερώνει υπάρχοντα context instances ανάλογα με τις παραμέτρους που δίνει ο χρήστης.
- `result get_attrs_of_ctx_sch():` Επιστρέφει τα attributes του πίνακα ο οποίος έχει δημιουργηθεί για την αποθήκευση των context instances.
- `int check_uniqueness():` Ελέγχει τη μοναδικότητα των context instances για κάθε πίνακα context. Η μέθοδος αυτή εκτελείται κάθε φορά που δημιουργείται ή τροποποιείται κάποιο context instance.



Σχήμα 4.4 Διάγραμμα κλάσεων υποσυστήματος διαχείρισης δεδομένων

Τα attributes της κλάσης MorphsData είναι τα ακόλουθα:

- `int mor_id`: Αποτελεί το μοναδικό χαρακτηριστικό κάθε αντικειμένου τύπου MorphsData που δημιουργείται
- `DbConnection newconn2`: Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `void insert_mor_inst(string)`: Εκτελεί τις εντολές για την εισαγωγή νέων δεδομένων.
- `void delete_mor_inst(int)`: Διαγράφει υπάρχοντα δεδομένα.

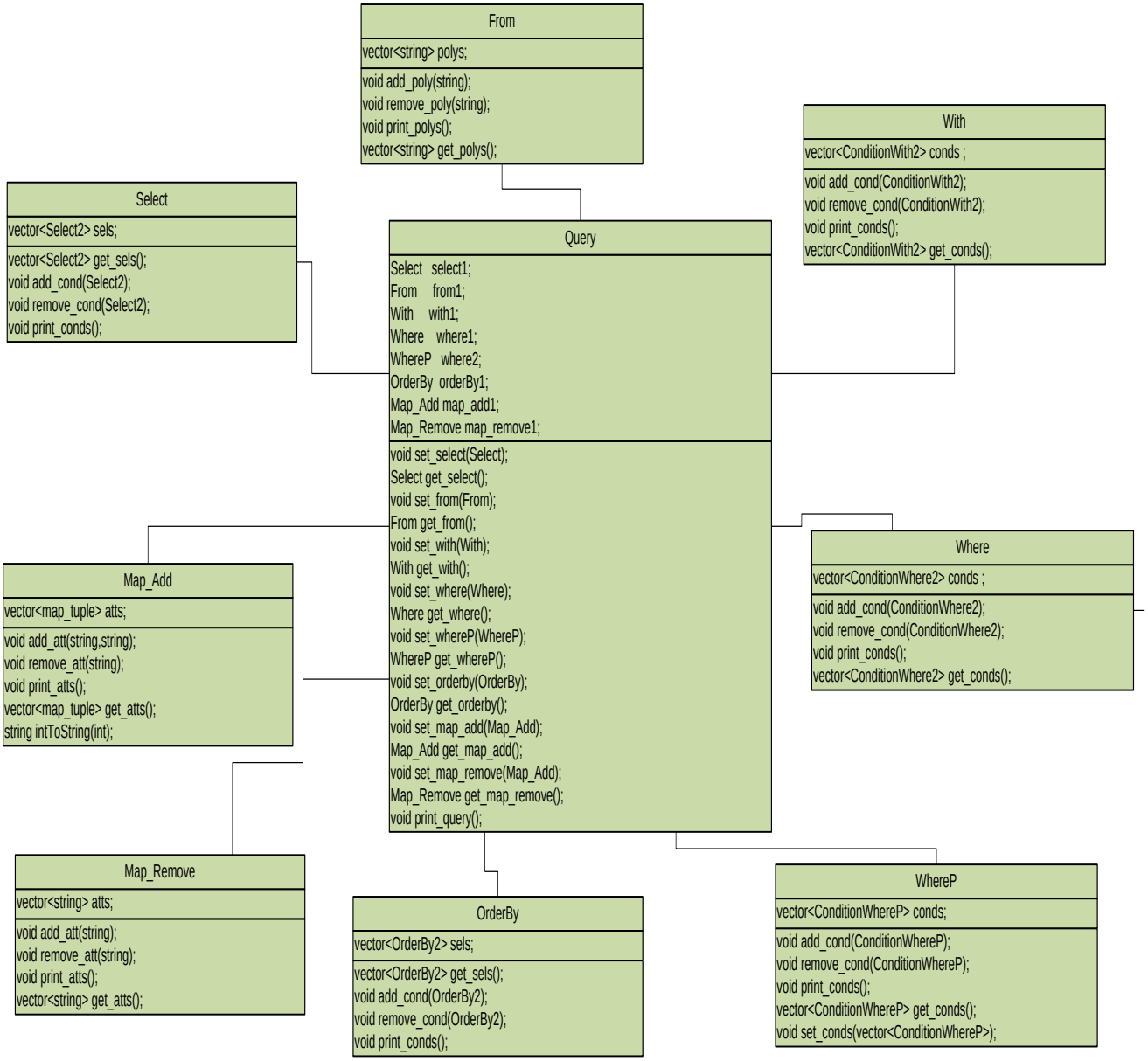
- `void update_mor_inst(string,int):` Ενημερώνει υπάρχοντα δεδομένα ανάλογα με τις παραμέτρους που δίνει ο χρήστης.
- `result get_attrs_of_morph():` Επιστρέφει τα attributes του πίνακα ο οποίος έχει δημιουργηθεί για την αποθήκευση των δεδομένων.
- `int check_uniqueness():` Ελέγχει τη μοναδικότητα των εγγραφών για κάθε πίνακα που αποθηκεύει δεδομένα κάποιου morph. Η μέθοδος αυτή εκτελείται κάθε φορά που δημιουργείται ή τροποποιείται κάποια εγγραφή που αφορά τα δεδομένα ενός morph.

4.2.5 Υποσύστημα αναπαράστασης ερωτημάτων

Η κλάση που υλοποιεί το υποσύστημα αναπαράστασης των ερωτημάτων είναι η Query. Στην κλάση αυτή ορίζονται αντικείμενα τύπου Select, From, With, Where, WhereP, OrderBy, Map_ADD, Map_REMOVE τα περισσότερα εκ των οποίων με τη σειρά τους για να οριστούν χρησιμοποιούν άλλες κλάσεις. Παρακάτω δίνεται το διάγραμμα των κλάσεων αυτών συγκεντρωτικά στο σχήμα 4.5 ενώ στη συνέχεια για όσα αντικείμενα χρησιμοποιήθηκαν επιπρόσθετες βοηθητικές κλάσεις, κάθε αντικείμενο από αυτά περιγράφεται ξεχωριστά δίνοντας το αντίστοιχο διάγραμμα κλάσεων που χρησιμοποιήθηκαν για τον ορισμό του.

Τα attributes της κλάσης Query κάθε ένα εκ των οποίων αντιπροσωπεύει μια συνθήκη του αρχικού ερωτήματος είναι τα ακόλουθα:

- `Select  select1`
- `From  from1`
- `With  with1`
- `Where  where1`
- `WhereP  where2`
- `OrderBy  orderBy1`
- `Map_Add  map_add1`
- `Map_Remove map_remove1`



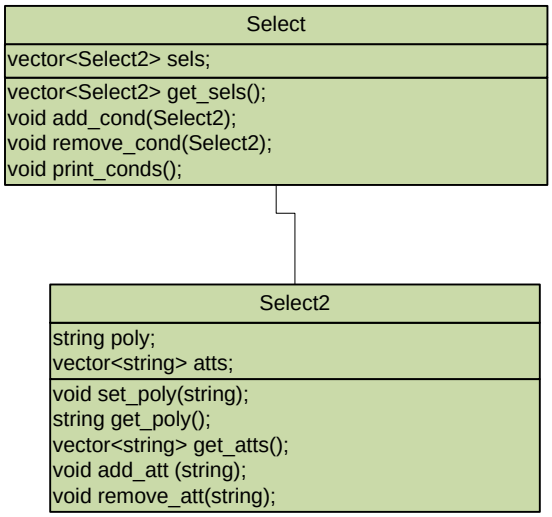
Σχήμα 4.5 Διάγραμμα κλάσεων υποσυστήματος αναπαράστασης ερωτημάτων

Όσον αφορά τις βασικές μεθόδους που υλοποιούνται στην κλάση αυτή, επειδή οι περισσότερες αφορούν τον ορισμό των επιμέρους παραμέτρων που ορίζονται στο αντικείμενο αυτό κρίνεται σκόπιμο να αναφέρουμε μόνο τη μέθοδο:

- `void print_query()`: Η μέθοδος αυτή τυπώνει το ερώτημα στη μορφή που το χρειάζεται το υποσύστημα μετατροπής και εκτέλεσης των ερωτημάτων για να προχωρήσει στην επεξεργασία του.

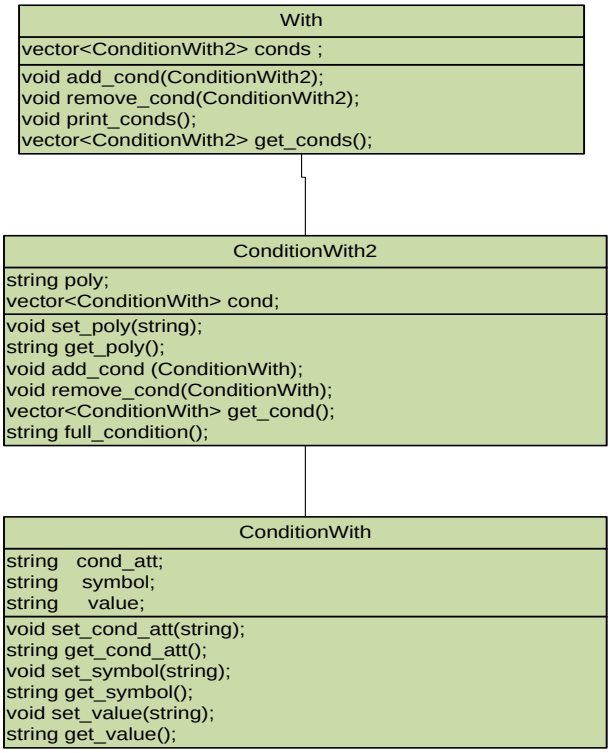
Για να περιγράψουμε αναλυτικότερα πως προέκυψαν τα αντικείμενα που χρησιμοποιούνται ως παράμετροι σε ένα αντικείμενο τύπου Query θα αναφερθούμε στις επιμέρους κλάσεις που χρησιμοποιήθηκαν για την περιγραφή κάθε αντικειμένου ξεχωριστά.

- Για το αντικείμενο Select χρησιμοποιήθηκε η κλάση Select2. Παρακάτω δίνεται το αντίστοιχο διάγραμμα κλάσεων (σχήμα 4.6).



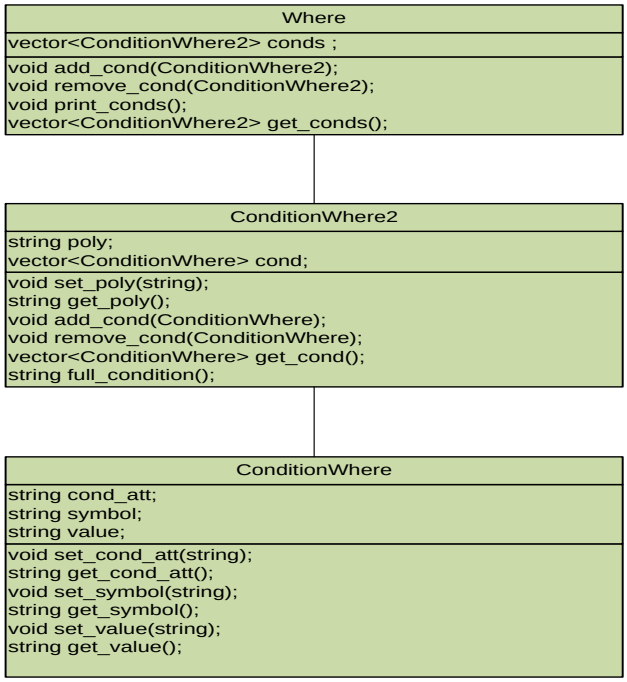
Σχήμα 4.6 Διάγραμμα κλάσεων Select,Select2

- Για το αντικείμενο From δε χρειάστηκε η χρήση κάποιας επιπλέον κλάσης από τη στιγμή που το αντικείμενο αυτό μοντελοποιείται ως μια λίστα από ονόματα polymorphs.
- Για το αντικείμενο With χρησιμοποιήθηκαν οι κλάσεις ConditionWith και ConditionWith2. Παρακάτω δίνεται το αντίστοιχο διάγραμμα κλάσεων στο σχήμα 4.7



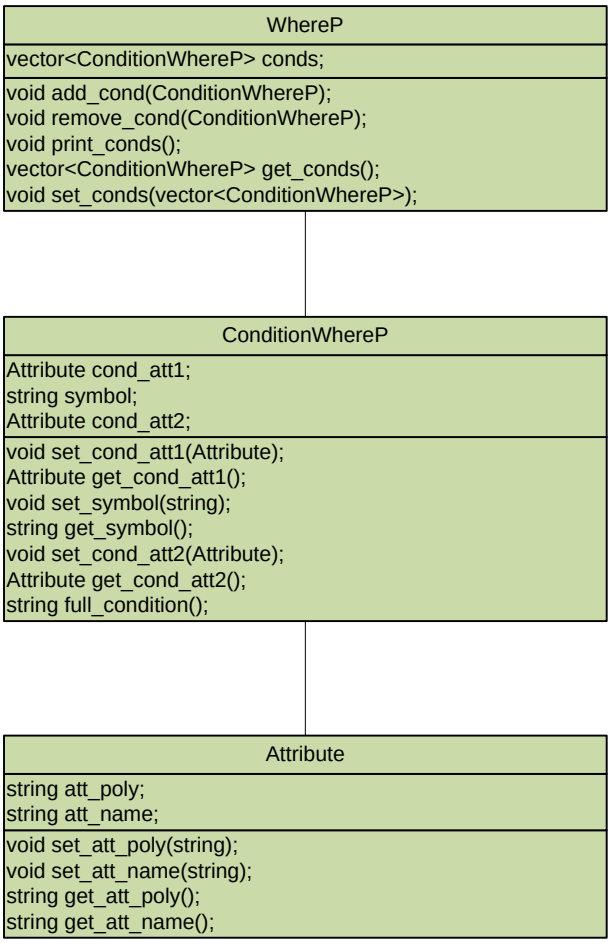
Σχήμα 4.7 Διάγραμμα κλάσεων ConditionWith, ConditionWith2,With

- Για το αντικείμενο Where χρησιμοποιήθηκαν οι κλάσεις ConditionWhere και ConditionWhere2. Παρακάτω δίνεται το αντίστοιχο διάγραμμα κλάσεων (σχήμα 4.8).



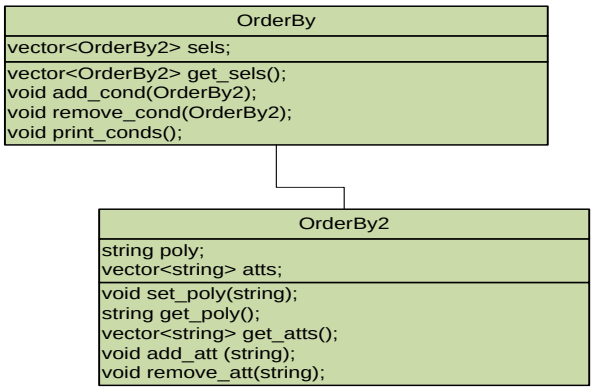
Σχήμα 4.8 Διάγραμμα κλάσεων ConditionWhere, ConditionWhere2,Where

- Για το αντικείμενο WhereP χρησιμοποιήθηκε η κλάση ConditionWhereP. Παρακάτω δίνεται το αντίστοιχο διάγραμμα κλάσεων (σχήμα 4.9).



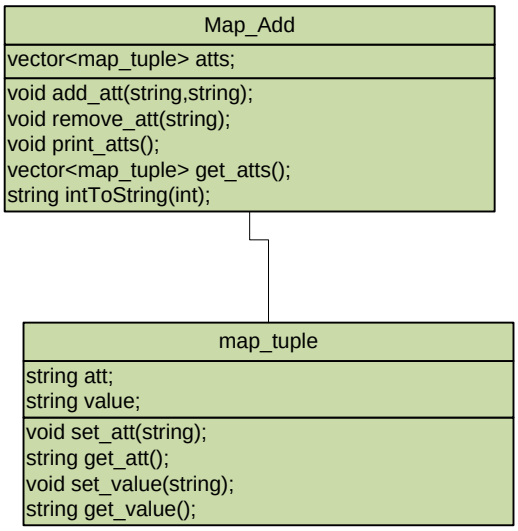
Σχήμα 4.9 Διάγραμμα κλάσεων Attribute, ConditionWhereP,WhereP

Για το αντικείμενο OrderBy χρησιμοποιήθηκε η κλάση OrderBy2. Παρακάτω δίνεται το αντίστοιχο διάγραμμα κλάσεων (σχήμα 4.10).



Σχήμα 4.10 Διάγραμμα κλάσεων OrderBy2, OrderBy

- Για το αντικείμενο Map_Add χρησιμοποιήθηκε η κλάση map_tuple. Παρακάτω δίνεται το αντίστοιχο διάγραμμα κλάσεων (σχήμα 4.11).

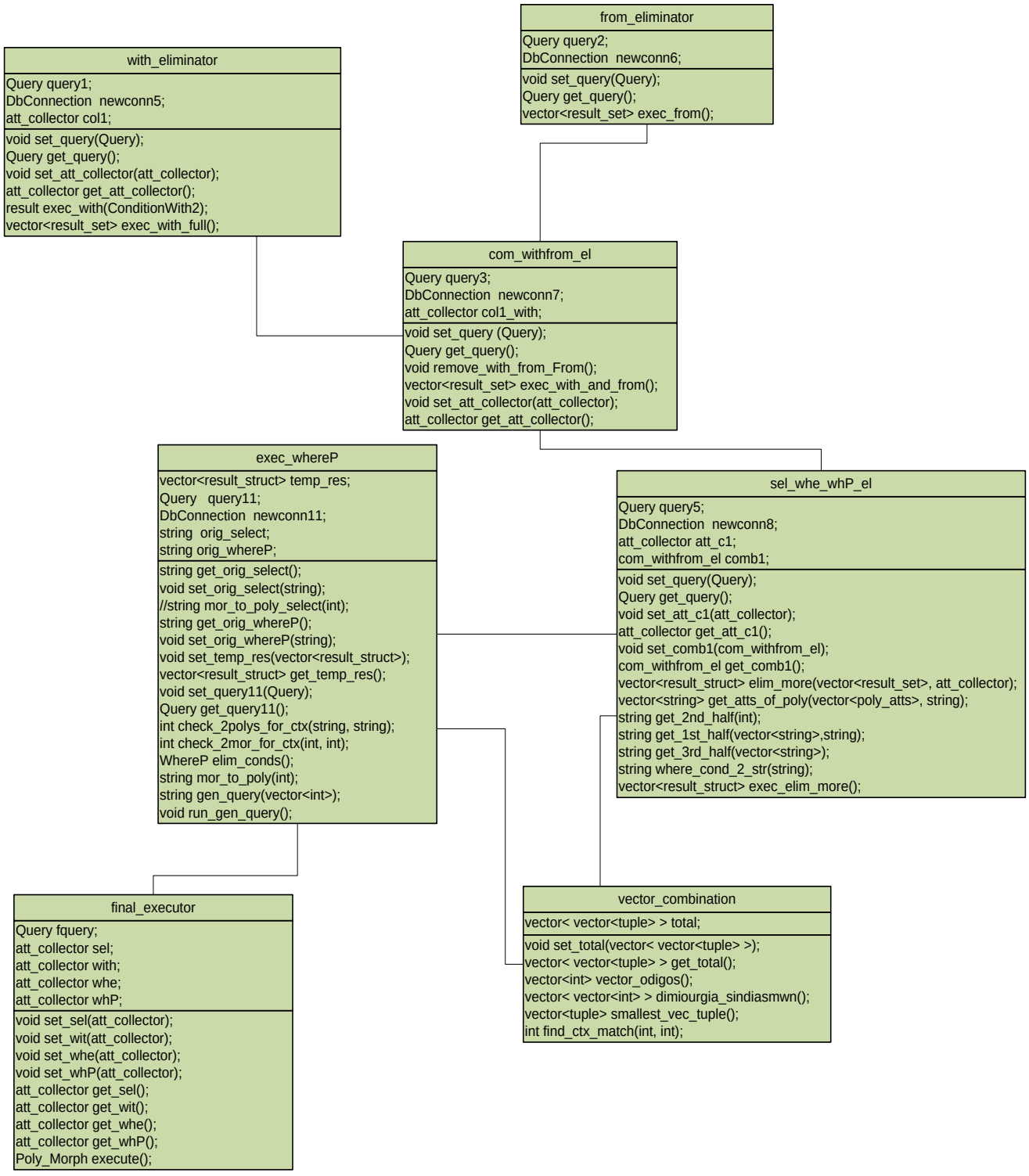


Σχήμα 4.11 Διάγραμμα κλάσεων map_tuple, Map_Add

- Για το αντικείμενο Map_Remove δε χρειάστηκε η χρήση κάποιας επιπλέον κλάσης από τη στιγμή που το αντικείμενο αυτό μοντελοποιείται ως μια λίστα από ονόματα attributes.

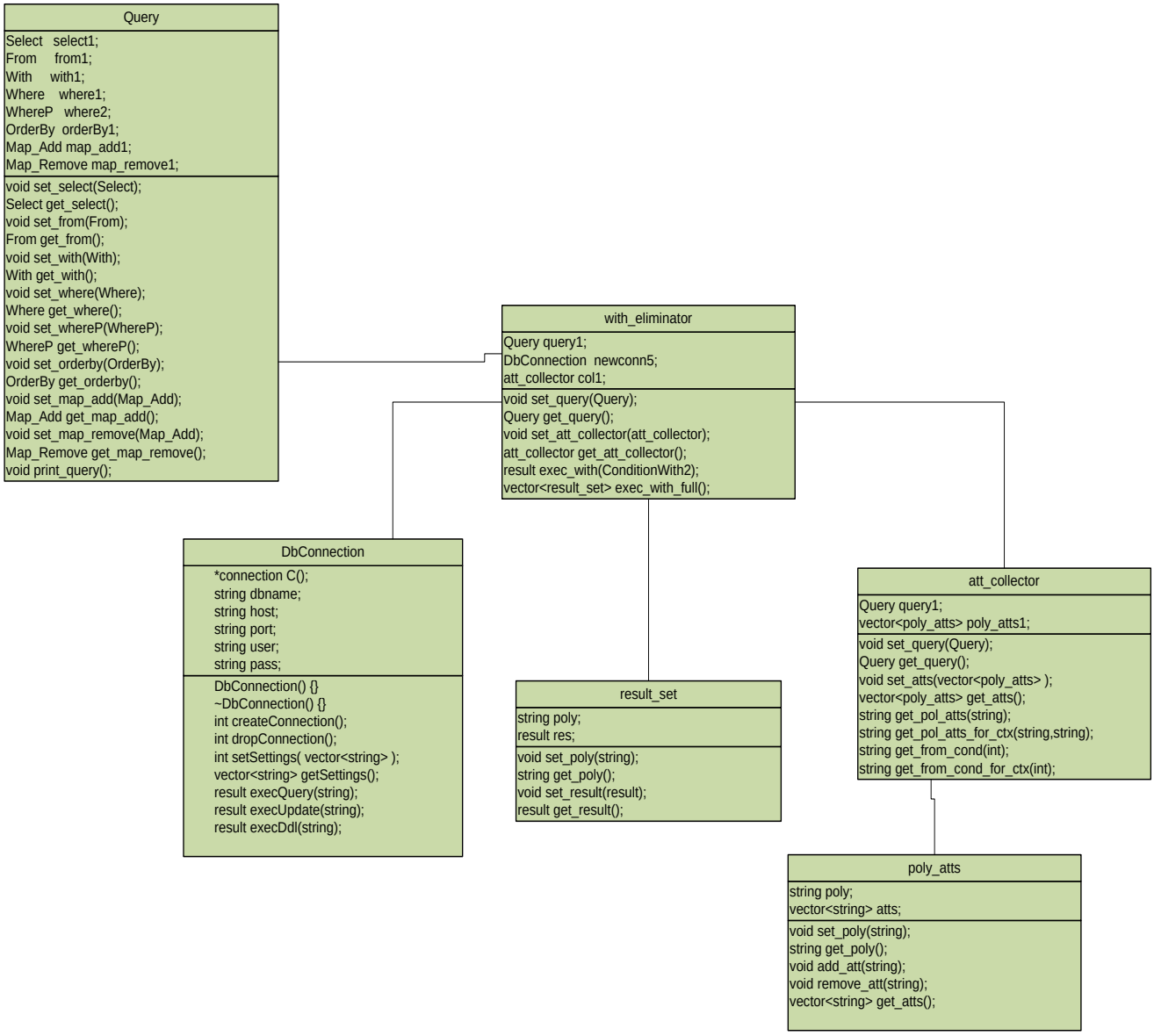
4.2.6 Υποσύστημα μετατροπής ερωτημάτων σε PL\SQL και εκτέλεσής τους

Οι κλάσεις που υλοποιούν το υποσύστημα αυτό είναι οι: With_eliminator, From_eliminator, com_withfrom_el, sel_whe_whP_el, vector_combination, exec_query και final_executor το συγκεντρωτικό διάγραμμα κλάσεων των οποίων δίνεται παρακάτω. Στη συνέχεια για κάθε κλάση δίνεται το διάγραμμά της (σχήμα 4.12), αναλύονται τα attributes που περιέχει καθώς και οι βασικότερες μέθοδοι που αυτή περιλαμβάνει.



Σχήμα 4.12 Συνολικό διάγραμμα κλάσεων υποσυστήματος μετατροπής σεPL\SQL και εκτέλεσης των ερωτημάτων

Για την κλάση `with_eliminator` έχουμε το ακόλουθο διάγραμμα (σχήμα 4.13):



Σχήμα 4.13 Διάγραμμα κλάσης with_eliminator

Τα attributes της κλάσης αυτής είναι τα ακόλουθα:

- `Query query1:` Αντικείμενο τύπου Query στο οποίο είναι αποθηκευμένο το ερώτημα του χρήστη στη μορφή που απαιτείται για την περαιτέρω επεξεργασία του.
- `DbConnection newconn5:` Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.
- `att_collector col1:` Αποτελεί αντικείμενο της βοηθητικής κλάσης att_collector και στην ουσία πρόκειται για συλλέκτη attributes. Για κάθε context schema για το

οποίο ορίζεται κάποια συνθήκη context είναι απαραίτητο να ελέγξουμε ότι αυτό περιλαμβάνει όλα τα attributes που αναφέρονται στο ερώτημα και η παράμετρος αυτή αντιπροσωπεύει τα attributes αυτά.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `result exec_with(ConditionWith2):` Παίρνοντας ως παράμετρο μια συνθήκη για context επιστρέφει ως αποτέλεσμα ένα αντικείμενο τύπου result που περιλαμβάνει όλα τα morphs-που ανήκουν στο polymorph για το οποίο τέθηκε η συνθήκη- που την ικανοποιούν.
- `vector<result_set> exec_with_full():` Εκτελώντας διαδοχικά την παραπάνω μέθοδο για όλες τις συνθήκες που όρισε ο χρήστης και για όλα τα polymorphs η μέθοδος αυτή επιστρέφει ένα vector<result_set> όπου το result_set είναι το αντικείμενο που προκύπτει από μια βοηθητική κλάση που δημιουργήθηκε με σκοπό την αποθήκευση των ενδιάμεσων αποτελεσμάτων. Το μέγεθος του vector είναι όσα και τα polymorphs για τα οποία έχουν οριστεί συνθήκες context και κάθε αντικείμενο τύπου result_set περιλαμβάνει το όνομα ενός polymorph και τα morphs του που πέρασαν με επιτυχία τους ελέγχους σχετικά με το context.

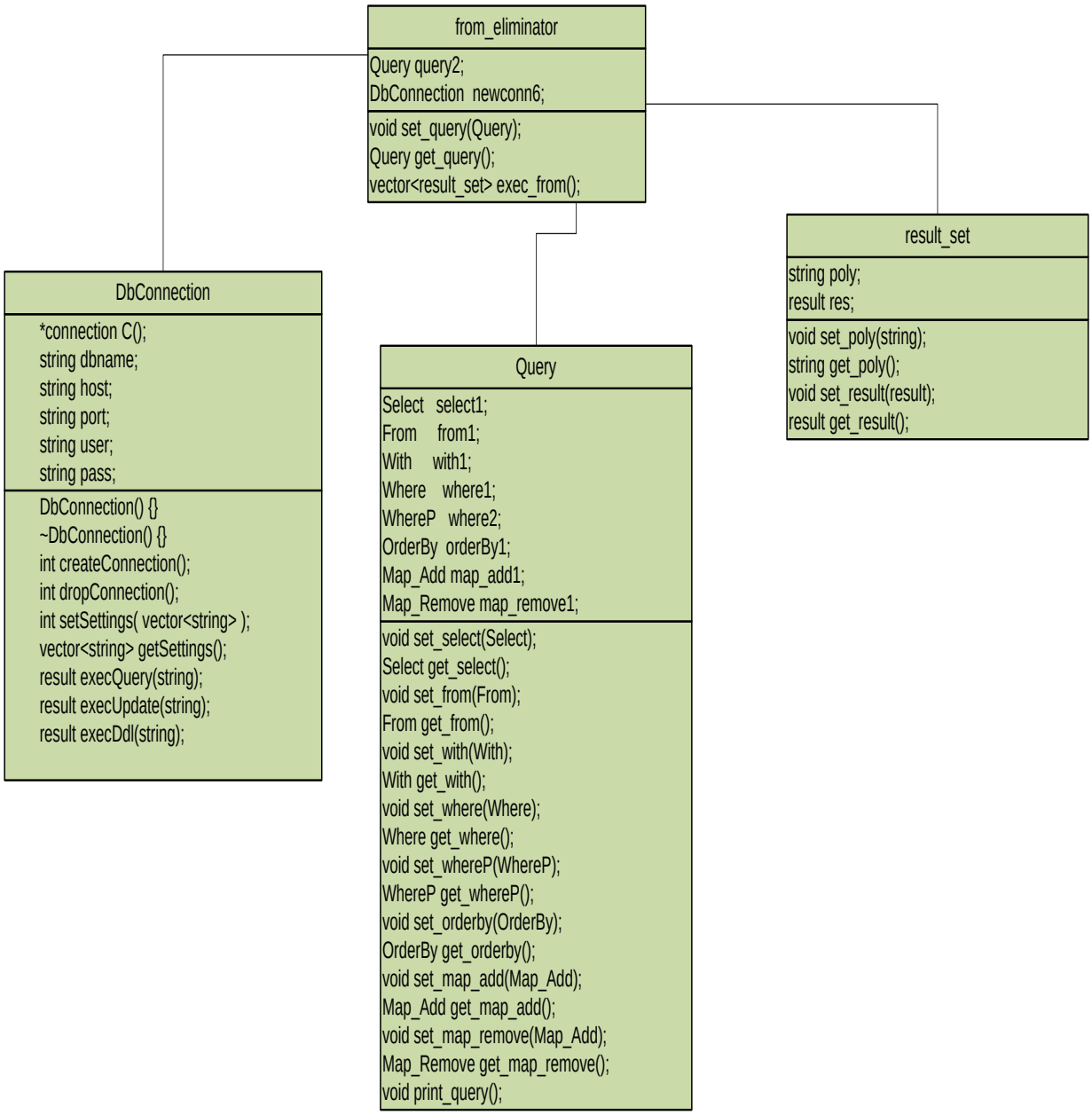
Για την κλάση `from_eliminator` το ακόλουθο διάγραμμα που δίνεται στο σχήμα 4.14.

Τα attributes της κλάσης αυτής είναι τα ακόλουθα:

- `Query query2:` Αντικείμενο τύπου Query στο οποίο είναι αποθηκευμένο το ερώτημα του χρήστη στη μορφή που απαιτείται για την περαιτέρω επεξεργασία του.
- `DbConnection newconn6:` Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

Η βασική μέθοδος που υλοποιείται στην κλάση αυτή είναι η ακόλουθη:

- `vector<result_set> exec_from():` Επιστρέφει ένα vector<result_set> με μέγεθος ίσα με τον αριθμό των polymorphs τα οποία λαμβάνουν μέρος στο ερώτημα αλλά για τα οποία δεν έχουν οριστεί συνθήκες context. Κάθε αντικείμενο τύπου result_set του vector αυτού περιλαμβάνει το όνομα ενός polymorph καθώς και όλα τα morphs αυτού.



Σχήμα 4.14 Διάγραμμα κλάσης from_eliminator

Για την κλάση `com_with_from_el` έχουμε το ακόλουθο διάγραμμα (σχήμα 4.15):

Σχήμα 4.15 Διάγραμμα κλάσης com_withfrom_el

Τα attributes της κλάσης αυτής είναι τα ακόλουθα:

- `Query query3`: Αντικείμενο τύπου Query στο οποίο είναι αποθηκευμένο το ερώτημα του χρήστη στη μορφή που απαιτείται για την περαιτέρω επεξεργασία του.
- `DbConnection newconn7`: Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.

- `att_collector col1_with:` Αντικείμενο της βοηθητικής κλάσης att_collector που αντιπροσωπεύει τα attributes που αναφέρονται στο ερώτημα και αφορούν το context.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `void remove_with_from_From():` Εφόσον έχουν οριστεί συνθήκες σχετικά με το context για κάποιο polymorph επιθυμούμε να εκτελέσουμε τις μεθόδους του αντικειμένου with_eliminator, διαφορετικά εκείνες του from_eliminator. Η μέθοδος αυτή επιτελεί αυτόν ακριβώς το διαχωρισμό.
- `vector<result_set> exec_with_and_from():` Εκτελούνται συγκεντρωτικά για όλα τα polymorphs οι μέθοδοι των with_eliminator και from_eliminator και επιστρέφεται ένα vector<result_set> μεγέθους όσα και τα polymorphs του ερωτήματος. Για κάθε polymorph έχουν απομείνει τελικά είτε μόνο τα morphs εκείνα που πέρασαν με επιτυχία τους ελέγχους (εφόσον για αυτά έχουν οριστεί συνθήκες context) είτε όλα τα morphs (εφόσον για αυτά δεν έχουν οριστεί συνθήκες context).

Για την κλάση `sel_whe_whP_el` έχουμε το διάγραμμα στο σχήμα 4.16:

Τα attributes της κλάσης αυτής είναι τα ακόλουθα:

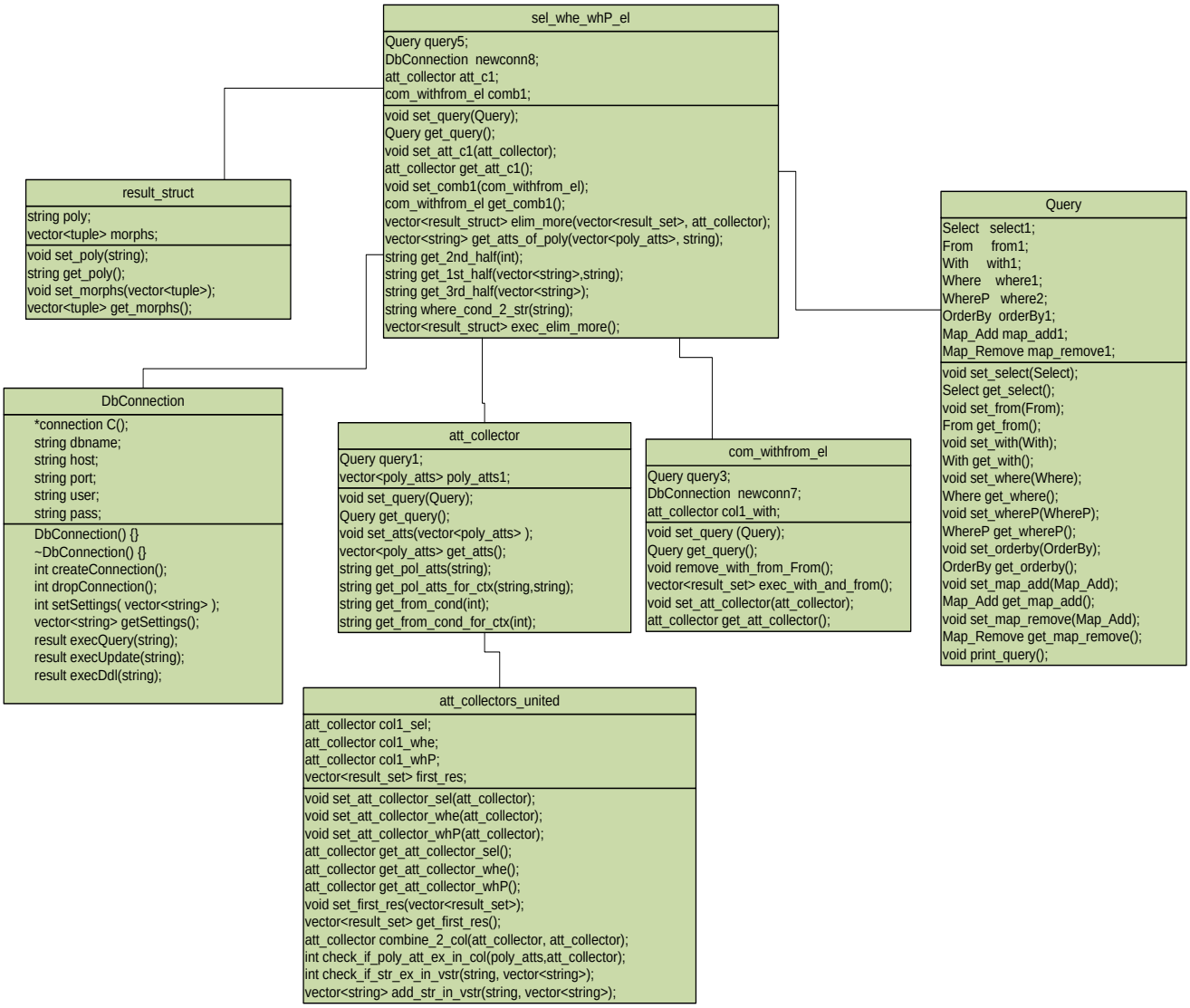
- `Query query5:` Αντικείμενο τύπου Query στο οποίο είναι αποθηκευμένο το ερώτημα του χρήστη στη μορφή που απαιτείται για την περαιτέρω επεξεργασία του.
- `DbConnection newconn8:` Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.
- `att_collector att_c1:` Αντικείμενο της βοηθητικής κλάσης att_collector που αντιπροσωπεύει τα attributes που αναφέρονται στο ερώτημα σε κάποιο εκ των αντικειμένων Select, Where ή WhereP και είναι απαραίτητο να ορίζονται για κάθε morph του polymorph για το οποίο ζητούνται.
- `com_withfrom_el comb1`: Αντικείμενο που εκτελεί τις συνθήκες σχετικά με το context.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `vector<result_struct> elim_more(vector<result_set>,att_collector):`

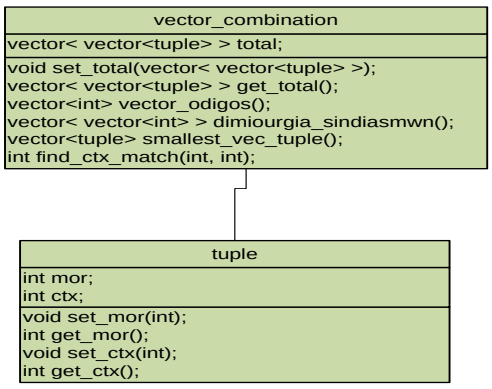
Η μέθοδος αυτή ελέγχει αν για κάθε morph έχουν οριστεί όλα τα attributes που ζητούνται στο ερώτημα, για το polymorph στο οποίο ανήκει. Στην περίπτωση που χρησιμοποιούμε ενδιάμεσους πίνακες, εδώ είναι το σημείο όπου αυτοί δημιουργούνται (είτε ένας για κάθε morph εφόσον βρισκόμαστε στην υλοποίηση A είτε ένας για κάθε polymorph εφόσον βρισκόμαστε στην υλοποίηση B). Τελικά επιστρέφεται ένα vector<result_struct> όπου το result_struct αποτελεί αντικείμενο μιας βοηθητικής κλάσης που έχει δημιουργηθεί. Κάθε αντικείμενο τύπου result_struct περιλαμβάνει ένα polymorph και τα morphs του τα οποία έχουν περάσει με επιτυχία όλους τους προηγούμενους ελέγχους.

- `vector<result_struct> exec_elim_more()`: Στη μέθοδο αυτή αρχικά το αντικείμενο comb1 τύπου com_withfrom_el εκτελεί τους ελέγχους του ερωτήματος σχετικά με το context. Πάνω στο αποτέλεσμα της εκτέλεσης αυτής, η μέθοδος exec_elim_more() στη συνέχεια εκτελεί την παραπάνω μέθοδο
  (elim_more(vector<result_set>, att_collector)).



Σχήμα 4.16 Διάγραμμα κλάσης sel_whe_whP_el

Για την κλάση `vector_combination` έχουμε το ακόλουθο διάγραμμα (σχήμα 4.17):



Σχήμα 4.17 Διάγραμμα κλάσης vector_combination

Το μοναδικό attribute της κλάσης αυτής είναι το ακόλουθο:

- `vector< vector<tuple> > total`: Σε κάθε αντικείμενο τύπου tuple που έχει προκύψει από μια βοηθητική κλάση που έχει δημιουργηθεί, αποθηκεύονται το id ενός morph και το context instance αυτού. Σε κάθε vector<tuple> αποθηκεύονται οι παραπάνω πληροφορίες για όλα τα morphs ενός polymorph και τελικά στο αντικείμενο total αποθηκεύονται τα παραπάνω για όλα τα polymorphs.

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

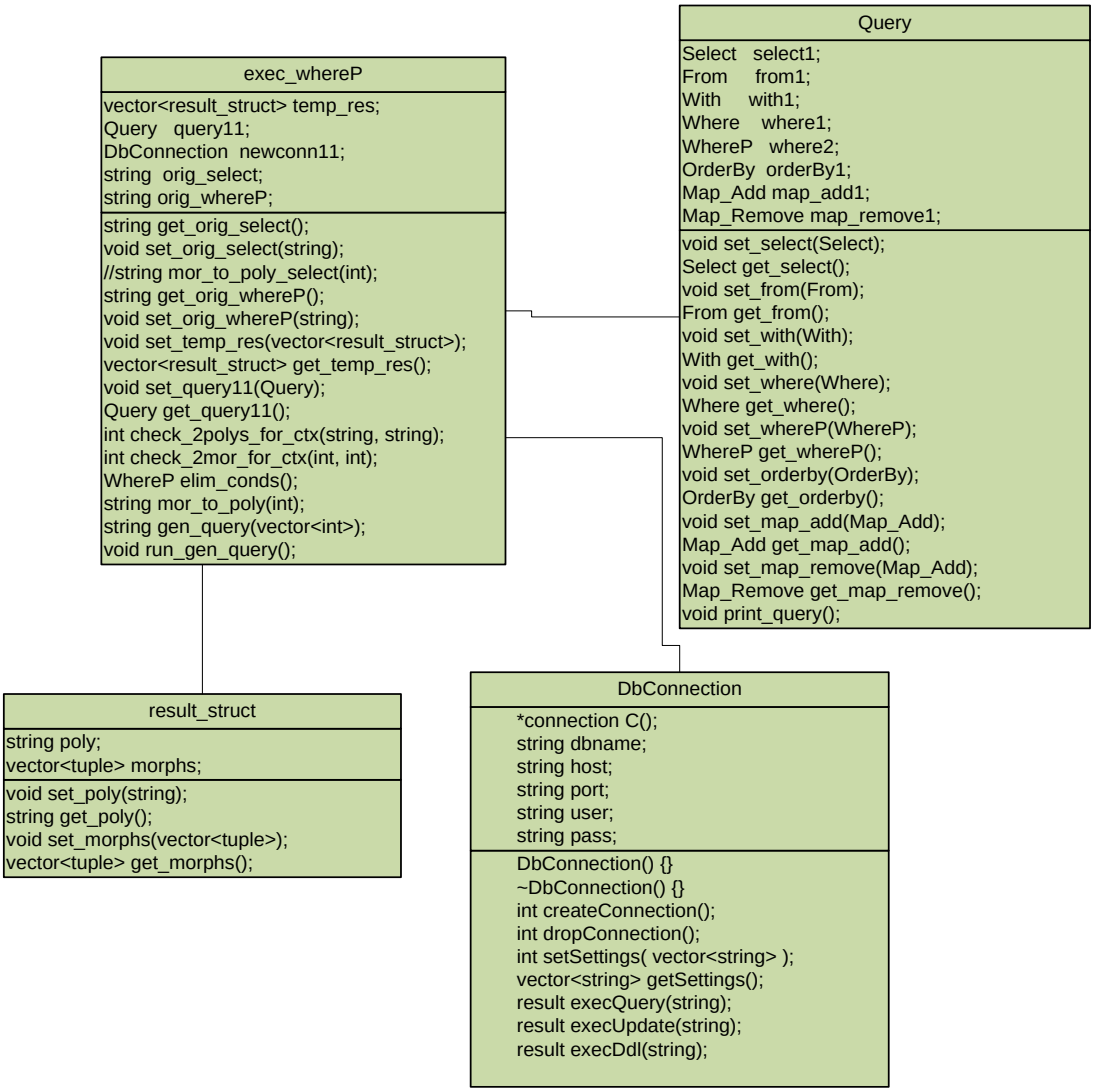
- `vector<int> vector_odigos()`: Η μέθοδος αυτή δημιουργεί ένα vector<int> με μέγεθος ίσο με τον αριθμό των polymorphs εκ των οποίων θα προκύψει ο ζητούμενος συνδυασμός από morphs πάνω στο οποίο θα γραφτεί τελικά ο συνδυασμός αυτός.
- `vector<tuple> smallest_vec_tuple()`: Σε κάθε επανάληψη για τη δημιουργία των συνδυασμών προτιμάται η αναζήτηση των morphs να ξεκινάει από τα morphs του polymorph εκείνου τα οποία είναι τα λιγότερα καθώς επειδή πρέπει σε κάθε συνδυασμό να έχουν χρησιμοποιηθεί ένα morph από κάθε polymorph αν n ο αριθμός των morphs του polymorph εκείνου που έχει τα λιγότερα, τότε το πολύ n συνδυασμοί είναι δυνατόν να προκύψουν.
- `vector< vector<int> > dimiourgia_sindiasmwn()`: Η μέθοδος αυτή επιστρέφει τους συνδυασμούς από τα morphs πάνω στους οποίους θα εκτελεστούν τα τελικά ερωτήματα.

Για την κλάση `exec_whereP` έχουμε το ακόλουθο διάγραμμα (σχήμα 4.18):

Τα attributes της κλάσης αυτής είναι τα ακόλουθα:

- `vector<result_struct> temp_res`: Vector αντικειμένων τύπου result_struct στα οποία βρίσκονται τα polymorphs με όσα morphs τους έχουν απομείνει από την προηγούμενη επεξεργασία.
- `Query query11`: Αντικείμενο τύπου Query στο οποίο είναι αποθηκευμένο το ερώτημα του χρήστη στη μορφή που απαιτείται για την περαιτέρω επεξεργασία του.
- `DbConnection newconn11`: Αποτελεί αντικείμενο μέσω του οποίου γίνεται η επικοινωνία με τη βάση δεδομένων.
- `string orig_select`: String που περιλαμβάνει τη συνθήκη select όπως ορίστηκε από το χρήστη στο ερώτημα που έθεσε.

- `string orig_whereP:` String που περιλαμβάνει τη συνθήκη whereP όπως ορίστηκε από το χρήστη στο ερώτημα που έθεσε.



Σχήμα 4.18 Διάγραμμα κλάσης exec_whereP

Οι βασικές μέθοδοι που υλοποιούνται στην κλάση αυτή είναι οι ακόλουθες:

- `WhereP elim_conds():` Ελέγχει αν στο αντικείμενο whereP υπάρχουν συνθήκες ανάμεσα σε polymorphs τα οποία δεν έχουν οριστεί με βάση το ίδιο context schema και εφόσον τέτοιες υπάρχουν τις αφαιρεί. Τελικά επιστρέφει ένα αντικείμενο τύπου whereP (με το οποίο αντικαθιστά και το αρχικά whereP του ερωτήματος) το οποίο περιλαμβάνει μόνο συνθήκες ανάμεσα σε polymorphs που έχουν οριστεί με βάση το ίδιο context schema.

- `string gen_query(vector<int>):` Ως παράμετρος στη μέθοδο αυτή δίνει ένα vector από int. Για κάθε τέτοιο vector η μέθοδος δημιουργεί τις συνθήκες Select, From και OrderBy των τελικών ερωτημάτων που θα εκτελεστούν.
- `void run_gen_query():` Η μέθοδος αυτή αρχικά χρησιμοποιεί ένα αντικείμενο τύπου vector_combination το οποίο όπως αναφέρθηκε παραπάνω δημιουργεί τους συνδυασμούς ανάμεσα στα morphs των polymorphs πάνω από τα οποία θα εκτελεστούν τα τελικά ερωτήματα. Στη συνέχεια για κάθε τέτοιο συνδυασμό προσθέτει στο αποτέλεσμα της μεθόδου gen_query(vector<int>) τις συνθήκες where και τελικά εκτελεί το ερώτημα. Τελικά επιστρέφει ένα polymorph ως αποτέλεσμα.

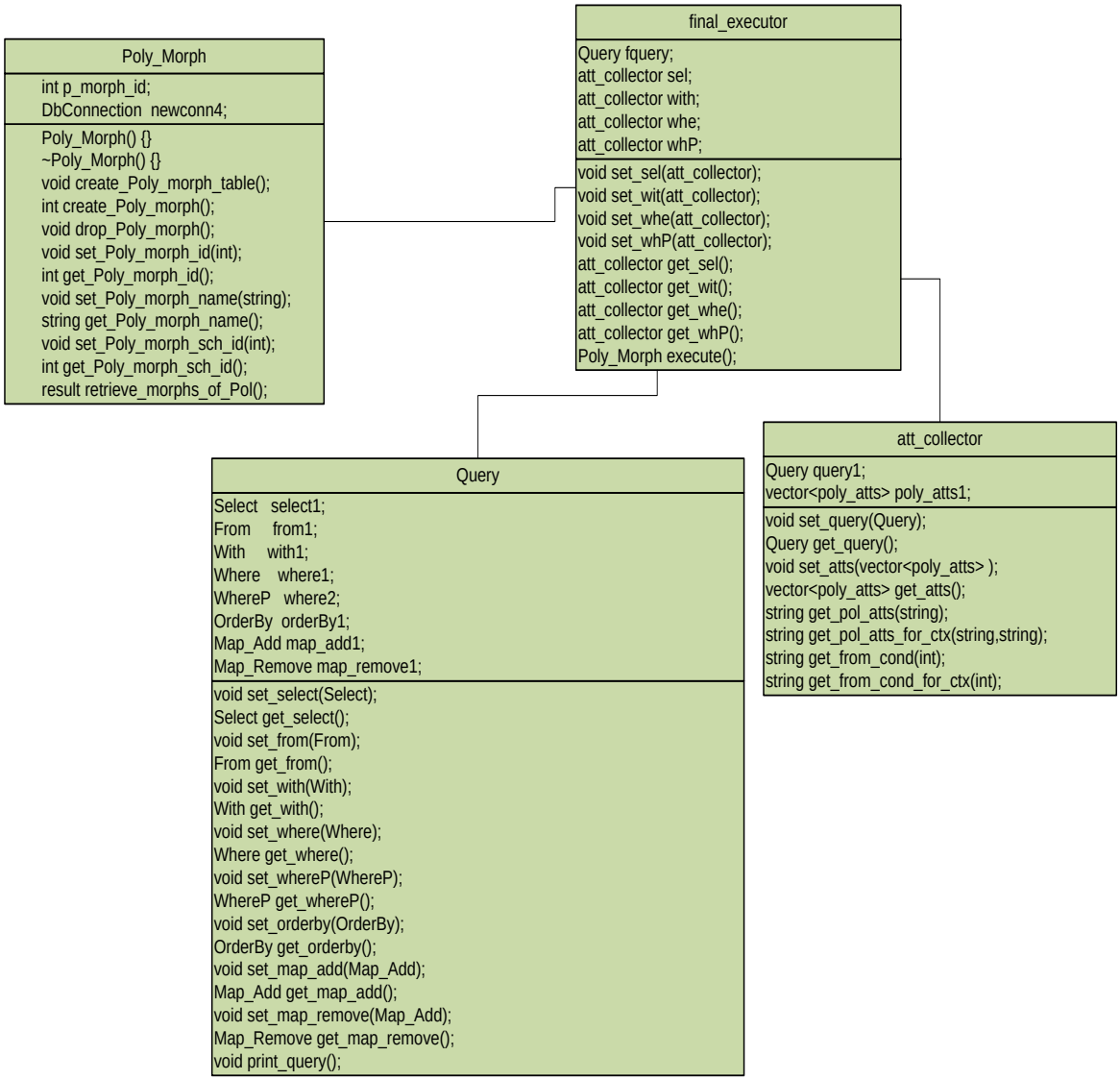
Για την κλάση `final_executor` έχουμε το διάγραμμα του σχήματος 4.19:

Τα attributes της κλάσης αυτής είναι τα ακόλουθα:

- `Query fquery:` Το ερώτημα που έθεσε ο χρήστης στη μορφή κατά την οποία μπορεί να επεξεργαστεί από το σύστημα
- `att_collector sel:` Τα attributes της συνθήκης Select.
- `att_collector wit`: Τα attributes της συνθήκης With.
- `att_collector whe:` Τα attributes της συνθήκης Where.
- `att_collector whP:` Τα attributes της συνθήκης WhereP.

Όσον αφορά τις βασικές μεθόδους που υλοποιούνται στην κλάση αυτή, επειδή οι περισσότερες αφορούν τον ορισμό των επιμέρους παραμέτρων που ορίζονται στο αντικείμενο αυτό κρίνεται σκόπιμο να αναφέρουμε μόνο τη μέθοδο:

- `Poly_Morph execute():` Η μέθοδος αυτή δημιουργώντας αντικείμενα όλων των προαναφερθεισών κλάσεων του υποσυστήματος αυτού, εκτελεί όλα τα στάδια της επεξεργασίας του ερωτήματος και τελικά επιστρέφει το αποτέλεσμα.



Σχήμα 4.19 Διάγραμμα κλάσης final_executor

5

Έλεγχος

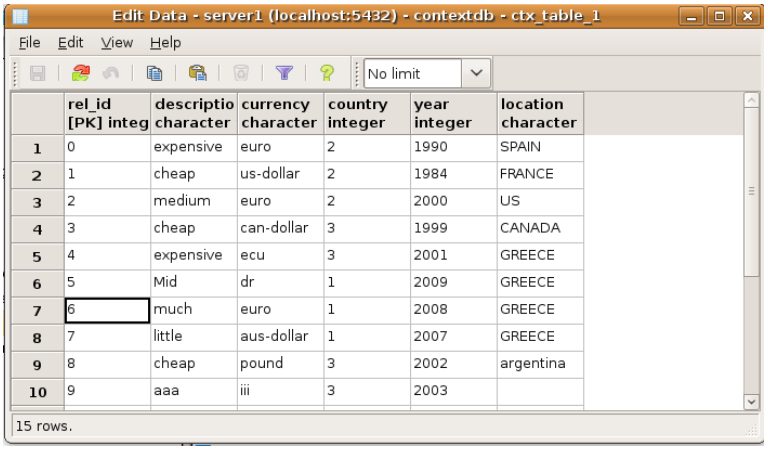
Στο κεφάλαιο αυτό παρουσιάζεται ο έλεγχος του συνολικού συστήματος. Στην ενότητα 5.1 παρουσιάζεται η μεθοδολογία που χρησιμοποιήθηκε για τον έλεγχο του υποσυστήματος μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων σε συνδυασμό με ένα παράδειγμα το οποίο μπορεί να μην είναι ιδιαίτερα ρεαλιστικό όσον αφορά την ποσότητα των δεδομένων πλην όμως συμβάλει στο να αποδείξουμε ότι το σύστημα αποκρίνεται όπως αναμενόταν. Στη συνέχεια, στην ενότητα 5.2 παρουσιάζεται ένα αντιπροσωπευτικό και αρκούντως ρεαλιστικό παράδειγμα το οποίο σκοπό έχει αφενός να δείξει ότι το σύστημα αποκρίνεται ανεξάρτητα από την ποσότητα των δεδομένων που είναι αποθηκευμένα σε αυτό και αφετέρου να αναδείξει την ταχύτητα απόκρισης με βάση την οποία θα εξαχθούν συμπεράσματα σχετικά με το overhead που δημιουργείται από τις πρόσθετες λειτουργίες που εισάγει το Context Aware σύστημα που έχει υλοποιηθεί πάνω από ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων στο σύστημα αυτό. Τελικά, στην ενότητα 5.3 συνοψίζονται τα συμπεράσματα από την λειτουργία και απόδοση του συστήματος συγκριτικά για τις δύο υλοποιήσεις που δημιουργήθηκαν (A και B) αλλά και για καθεμιά από τις δύο προσεγγίσεις (χρήση ή μη ενδιάμεσων-προσωρινών πινάκων) που χρησιμοποιήθηκαν σε κάθε υλοποίηση.

5.1 Έλεγχος ορθότητας

Κατά την υλοποίηση του συστήματος, η δημιουργία κάθε κλάσης συνοδευόταν από μια κλάση ελέγχου η οποία είχε σκοπό τον έλεγχο της σωστής λειτουργίας της πρώτης. Η κλάση αυτή του ελέγχου περιελάμβανε τόσο συγκεκριμένα σενάρια για τα οποία γνωρίζαμε εκ των προτέρων τα αναμενόμενα αποτελέσματα τους οπότε μετά την επιτυχή ολοκλήρωση τους γινόταν και η σχετική επαλήθευση όσο και παθολογικές περιπτώσεις χρήσης ώστε να αναδειχθούν οι εξαιρέσεις ή τα σφάλματα στον κώδικα και πώς τα διαχειριζόμαστε. Εκτελώντας τελικά κάποια από αυτά τα σενάρια που ολοκληρώνονται με επιτυχία

δημιουργήθηκε ένα πρώτο ολοκληρωμένο παράδειγμα το οποίο και θα παρουσιαστεί στην ενότητα αυτή.

Το παράδειγμα αυτό αφορά μια ηλεκτρονική αποθήκη οι οποία προμηθεύει με διάφορα προϊόντα ένα σύνολο καταστημάτων. Το εκάστοτε σχήμα της βάσης αλλάζει ανάλογα με το έτος το οποίο είναι προς εξέτασιν αλλά και τη χώρα που μας ενδιαφέρει. Είναι συνεπώς προφανής η ανάγκη ορισμού ενός context schema που θα περιλαμβάνει αυτές τις παραμέτρους. Άρα το context schema που θα χρησιμοποιήσουμε είναι το <YEAR, LOCATION> για το οποίο για τις ανάγκες των πειραμάτων δημιουργήθηκαν διάφορα context instances εκ των οποίων θα χρησιμοποιηθούν δέκα. Παρακάτω δίνεται σε screenshot ο πίνακας στο οποίο είναι αποθηκευμένα τα δεδομένα αυτά.



	rel_id [PK] integ	descriptio character	currency character	country integer	year integer	location character
1	0	expensive	euro	2	1990	SPAIN
2	1	cheap	us-dollar	2	1984	FRANCE
3	2	medium	euro	2	2000	US
4	3	cheap	can-dollar	3	1999	CANADA
5	4	expensive	ecu	3	2001	GREECE
6	5	Mid	dr	1	2009	GREECE
7	6	much	euro	1	2008	GREECE
8	7	little	aus-dollar	1	2007	GREECE
9	8	cheap	pound	3	2002	argentina
10	9	aaa	iii	3	2003	

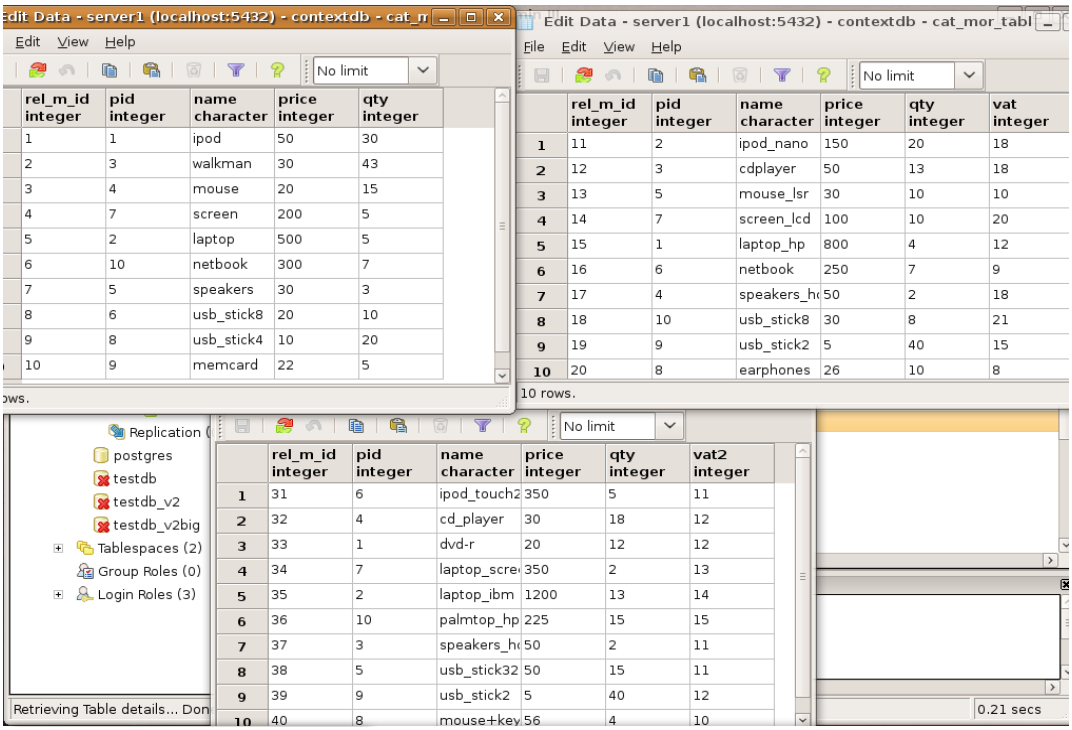
15 rows.

Στη συνέχεια δημιουργήθηκαν τρία polymorphs, τα οποία είναι τα εξής: PRODUCT, STORE και INVENTORY. Το PRODUCT αντιπροσωπεύει τα προϊόντα τα οποία είναι προς πώληση το STORE τα καταστήματα τα οποία πωλούν τα προϊόντα και το INVENTORY διατηρεί πληροφορίες σχετικά με ποιο προϊόν προωθείται σε ποιο κατάστημα. Κάθε ένα από αυτά τα polymorphs αποτελείται από 10 morphs (άρα χρειάζονται 10 context instances) τα attributes των οποίων διαφέρουν στη γενική περίπτωση, παρ' όλα αυτά κάποια διατηρούνται κοινά. Για παράδειγμα σε μια χώρα μπορεί να υπάρχει ένας ειδικός φόρος για ένα προϊόν ο οποίος να μην υπάρχει σε μια άλλη. Κατά συνέπεια στο morph το οποίο αντιπροσωπεύει το product για το context της πρώτης χώρας υπάρχει το αντίστοιχο attribute ενώ για τη δεύτερη όχι. Ωστόσο όπως αναφέρθηκε παραπάνω υπάρχουν κάποια κοινά attributes ανάμεσα στα morphs κάθε polymorph και αυτό συμβαίνει ώστε να μην καταλήγουμε να εκτελούμε στο τέλος ένα σχεσιακό ερώτημα επειδή τελικά είναι μόνο ένα το morph εκείνο που έχει ορισμένα όλα τα attributes που ζητούνται από το ερώτημα του χρήστη. Παρακάτω δίνονται για κάθε polymorph τα κοινά attributes των morphs του και τι αυτά αντιπροσωπεύουν.

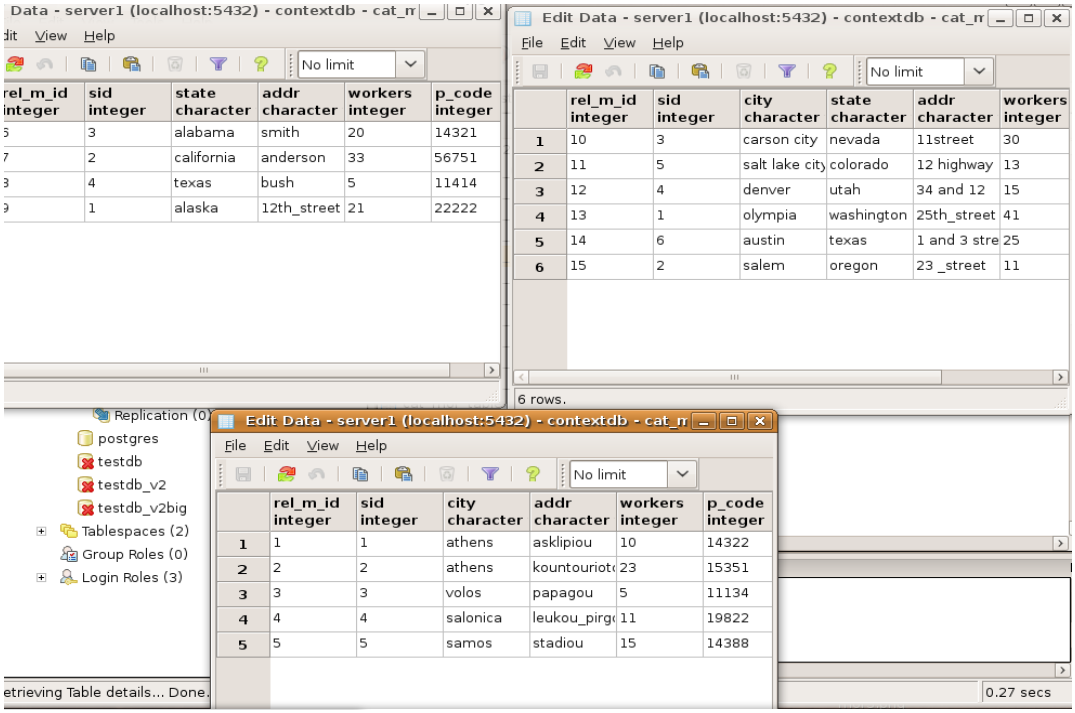
Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθούμε στον όγκο των δεδομένων που είναι αποθηκευμένα σε κάθε morph ώστε να είναι δυνατή μια καλύτερη εκτίμηση σχετικά με τους αναμενόμενους χρόνους των αποτελεσμάτων. Συνεπώς για κάθε morph του PRODUCT έχουμε κατά μέσο όρο 10 tuples για κάθε morph του STORE 5 tuples και για κάθε morph του INVENTORY 15 tuples. Στη συνέχεια και πριν προχωρήσουμε στην εκτέλεση των ερωτημάτων παρουσιάζονται για κάθε προαναφερθέν polymorph μερικά morphs με τα δεδομένα τους σε screenshot τόσο για την υλοποίηση A όσο και για τη B.

Υλοποίηση Α

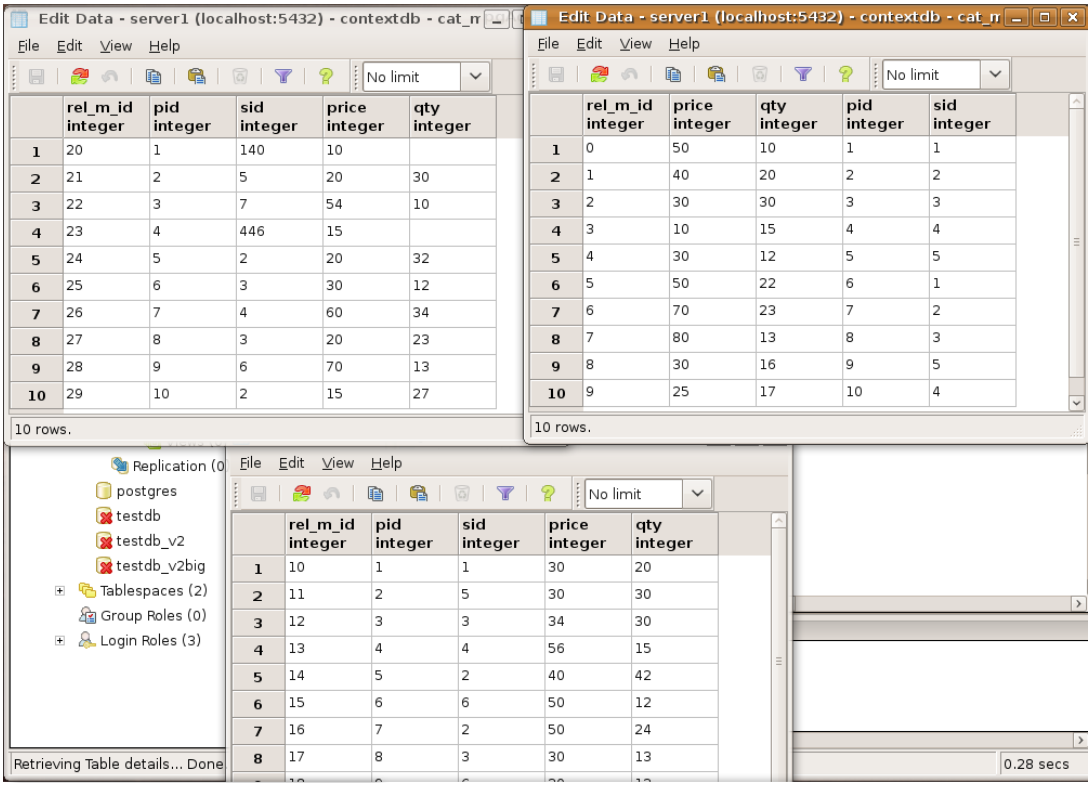
PRODUCT:



STORE:

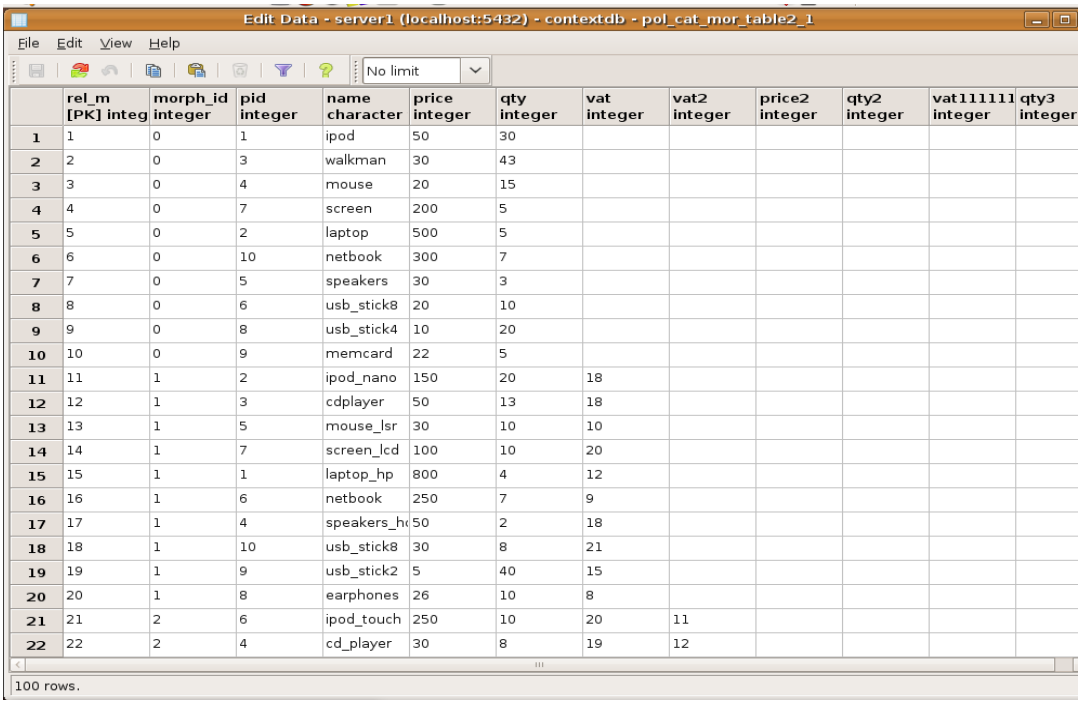


INVENTORY:



Υλοποίηση Β

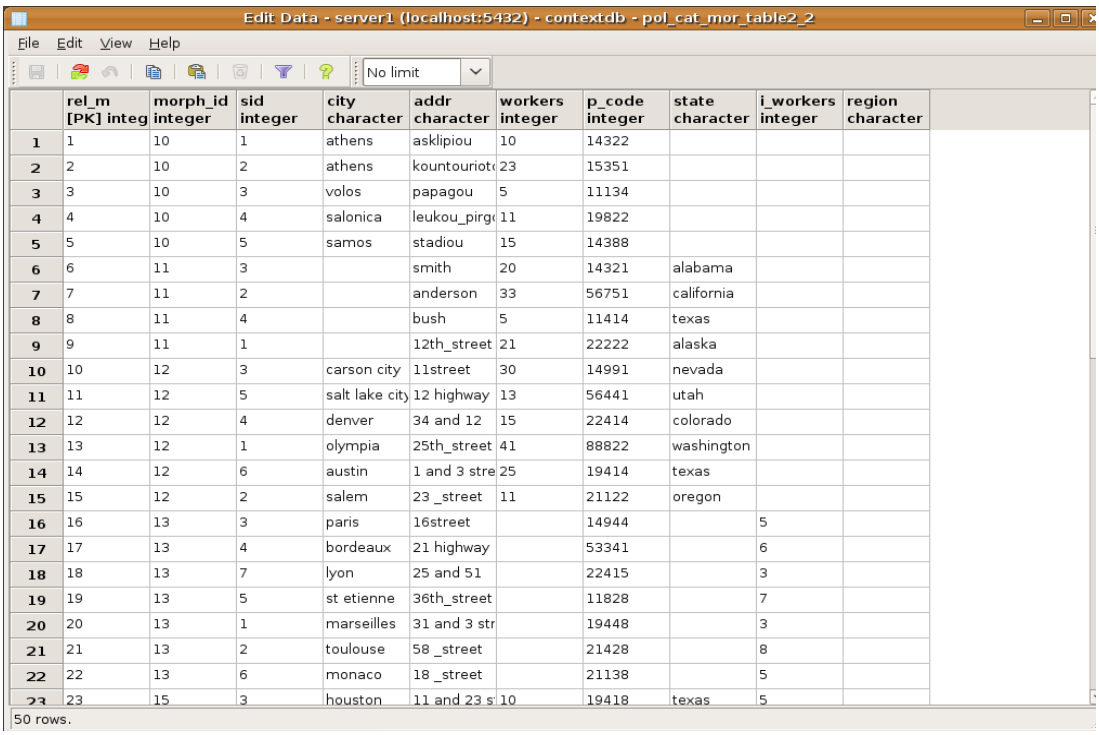
PRODUCT:



	rel_m [PK] integ	morph_id integer	pid integer	name character	price integer	qty integer	vat integer	vat2 integer	price2 integer	qty2 integer	vat111111 integer	qty3 integer
1	1	0	1	ipod	50	30						
2	2	0	3	walkman	30	43						
3	3	0	4	mouse	20	15						
4	4	0	7	screen	200	5						
5	5	0	2	laptop	500	5						
6	6	0	10	netbook	300	7						
7	7	0	5	speakers	30	3						
8	8	0	6	usb_stick8	20	10						
9	9	0	8	usb_stick4	10	20						
10	10	0	9	memcard	22	5						
11	11	1	2	ipod_nano	150	20	18					
12	12	1	3	cdplayer	50	13	18					
13	13	1	5	mouse_lsr	30	10	10					
14	14	1	7	screen_lcd	100	10	20					
15	15	1	1	laptop_hp	800	4	12					
16	16	1	6	netbook	250	7	9					
17	17	1	4	speakers_h	50	2	18					
18	18	1	10	usb_stick8	30	8	21					
19	19	1	9	usb_stick2	5	40	15					
20	20	1	8	earphones	26	10	8					
21	21	2	6	ipod_touch	250	10	20	11				
22	22	2	4	cd_player	30	8	19	12				

100 rows.

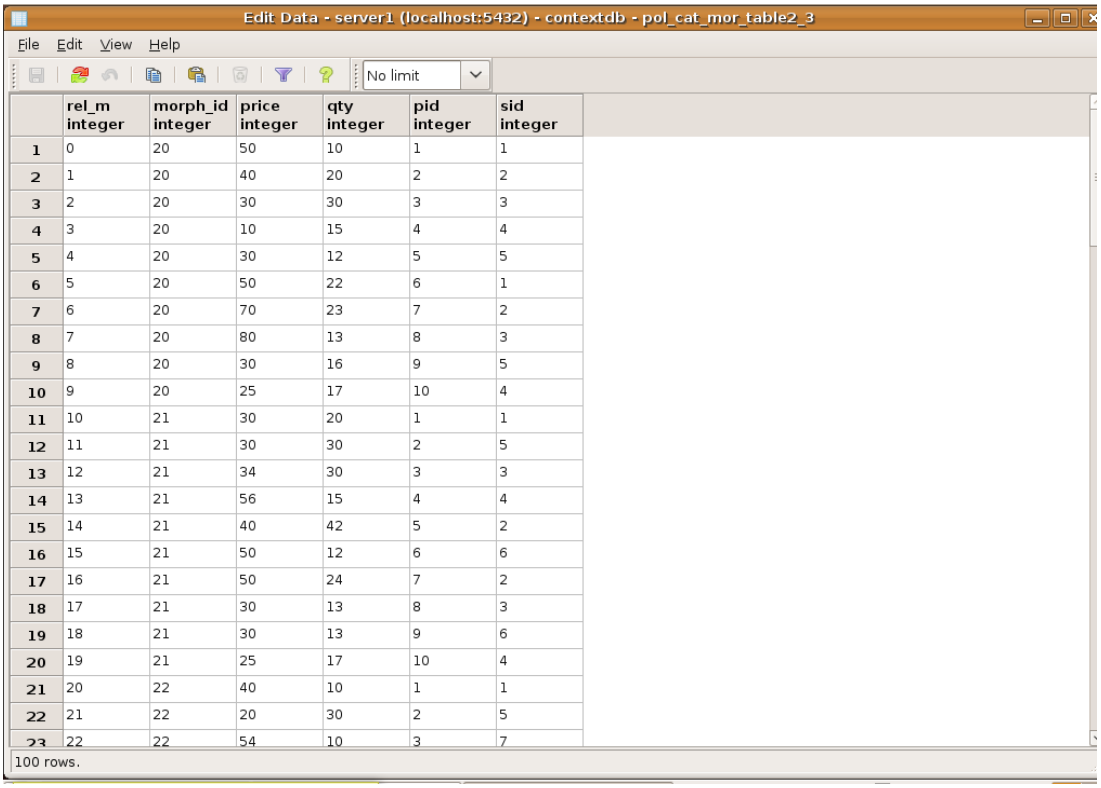
STORE:



	rel_m [PK] integ	morph_id integer	sid integer	city character	addr character	workers integer	p_code integer	state character	i_workers integer	region character
1	1	10	1	athens	asklipiou	10	14322			
2	2	10	2	athens	kountouriot	23	15351			
3	3	10	3	volos	papagou	5	11134			
4	4	10	4	salonica	leukou_pirg	11	19822			
5	5	10	5	samos	stadiou	15	14388			
6	6	11	3		smith	20	14321	alabama		
7	7	11	2		anderson	33	56751	california		
8	8	11	4		bush	5	11414	texas		
9	9	11	1		12th_street	21	22222	alaska		
10	10	12	3	carson city	11street	30	14991	nevada		
11	11	12	5	salt lake city	12 highway	13	56441	utah		
12	12	12	4	denver	34 and 12	15	22414	colorado		
13	13	12	1	olympia	25th_street	41	88822	washington		
14	14	12	6	austin	1 and 3 stre	25	19414	texas		
15	15	12	2	salem	23 _street	11	21122	oregon		
16	16	13	3	paris	16street		14944		5	
17	17	13	4	bordeaux	21 highway		53341		6	
18	18	13	7	lyon	25 and 51		22415		3	
19	19	13	5	st etienne	36th_street		11828		7	
20	20	13	1	marseilles	31 and 3 str		19448		3	
21	21	13	2	toulouse	58 _street		21428		8	
22	22	13	6	monaco	18 _street		21138		5	
23	23	15	3	houston	11 and 23 s	10	19418	texas	5	

50 rows.

INVENTORY:



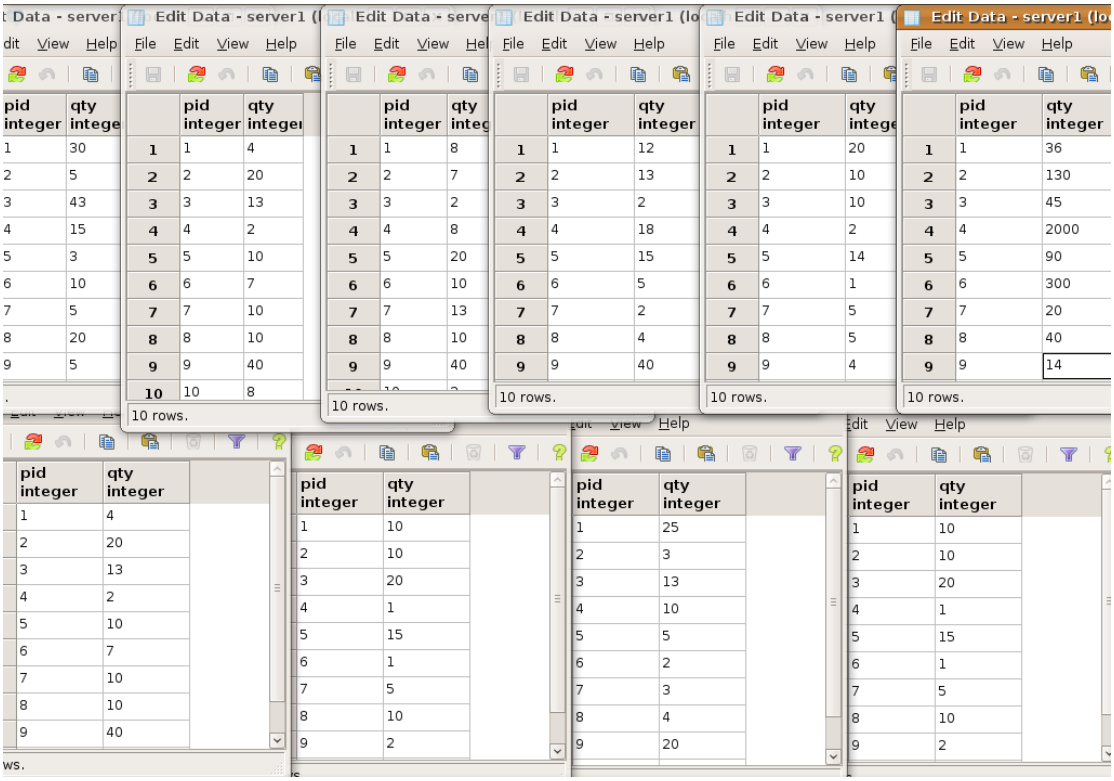
	rel_m integer	morph_id integer	price integer	qty integer	pid integer	sid integer
1	0	20	50	10	1	1
2	1	20	40	20	2	2
3	2	20	30	30	3	3
4	3	20	10	15	4	4
5	4	20	30	12	5	5
6	5	20	50	22	6	1
7	6	20	70	23	7	2
8	7	20	80	13	8	3
9	8	20	30	16	9	5
10	9	20	25	17	10	4
11	10	21	30	20	1	1
12	11	21	30	30	2	5
13	12	21	34	30	3	3
14	13	21	56	15	4	4
15	14	21	40	42	5	2
16	15	21	50	12	6	6
17	16	21	50	24	7	2
18	17	21	30	13	8	3
19	18	21	30	13	9	6
20	19	21	25	17	10	4
21	20	22	40	10	1	1
22	21	22	20	30	2	5
23	22	22	54	10	3	7

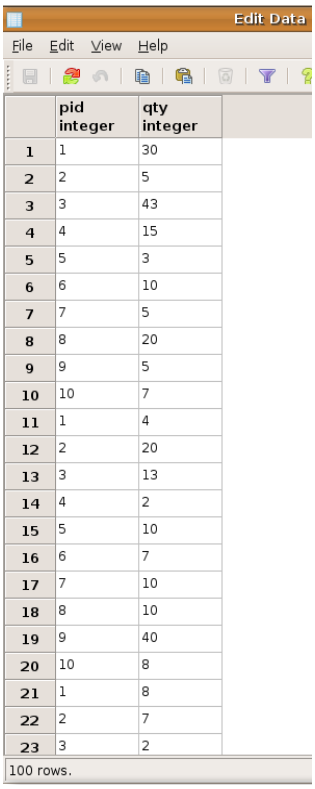
5.1.1 Ερώτημα 1

Το πρώτο ερώτημα που εκτελέστηκε για το σύστημα που περιγράφεται στην ενότητα αυτή δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT  product.pid, product.qty
FROM    product
```

Ουσιαστικά ζητάει την προβολή των pid και qty για όλα τα προϊόντα. Με βάση το ερώτημα αυτό, πάνω από κάθε morph του polymorph PRODUCT που έχει ορισμένα τα attributes pid και qty εκτελείται το σχεσιακό ερώτημα που προβάλλει τα attributes pid και qty. Για τα δεδομένα του συστήματος μας όμως, κάθε morph του PRODUCT περιλαμβάνει τα δύο ζητούμενα attributes οπότε και συμμετέχει στο τελικό αποτέλεσμα εφόσον δεν ορίζεται συνθήκη σχετική με το context που θα οδηγούσε σε περικοπή κάποιων morphs.. Παρακάτω δίνεται ένα screenshot με τα αποτελέσματα και για τις 2 υλοποιήσεις.





Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις έχουμε τα ακόλουθα αποτελέσματα:

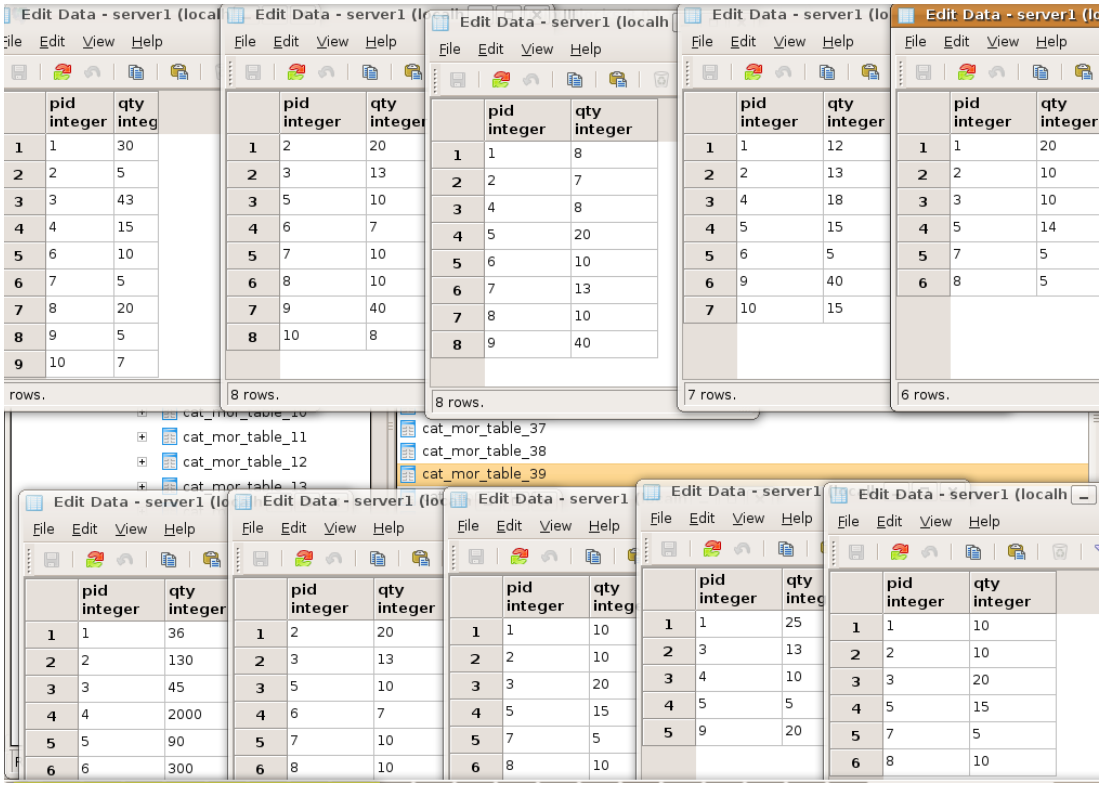
	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β	
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες
Εκτέλεση WITH	0,013	0,019	0,019	0,038
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	0,128	0,073	0,072	0,042
Τελική εκτέλεση +WHEREP	0,035	0,038	0,077	0,062
Συνολικά	0,177	0,138	0,169	0,144

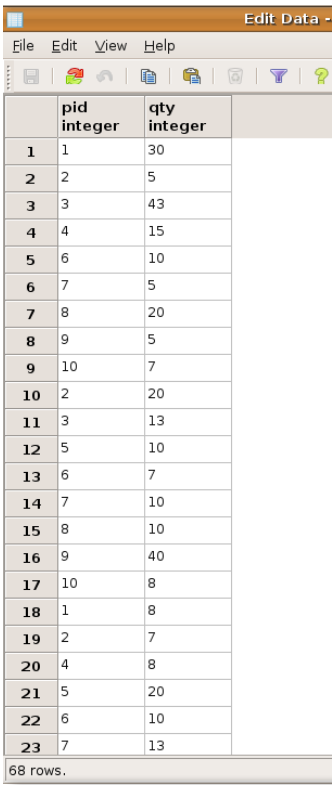
5.1.2 Ερώτημα 2

Το δεύτερο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT product.pid, product.qty
FROM   product
WHERE  product.qty>4
```

Ουσιαστικά ζητάει την προβολή των pid και qty για όλα τα προϊόντα των οποίων το qty είναι μεγαλύτερο του 10. Με βάση το ερώτημα αυτό, πάνω από κάθε morph του polymorph PRODUCT που έχει ορισμένα τα attributes pid και qty εκτελείται το σχεσιακό ερώτημα που προβάλλει τα attributes pid και qty εφόσον η ποσότητα του προϊόντος είναι μεγαλύτερη του 10. Για τα δεδομένα του συστήματος μας όμως, κάθε morph του PRODUCT περιλαμβάνει τα δύο ζητούμενα attributes οπότε και συμμετέχει στο τελικό αποτέλεσμα εφόσον δεν ορίζεται ούτε συνθήκη σχετική με το context που θα οδηγούσε σε περικοπή κάποιων morphs. Παρακάτω δίνεται ένα screenshot με τα αποτελέσματα και για τις 2 υλοποιήσεις.





Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις έχουμε τα ακόλουθα αποτελέσματα:

	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β	
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες
Εκτέλεση WITH	0,025	0,041	0,012	0,013
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	0,137	0,042	0,109	0,033
Τελική εκτέλεση +WHEREP	0,082	0,070	0,072	0,087
Συνολικά	0,245	0,154	0,195	0,134

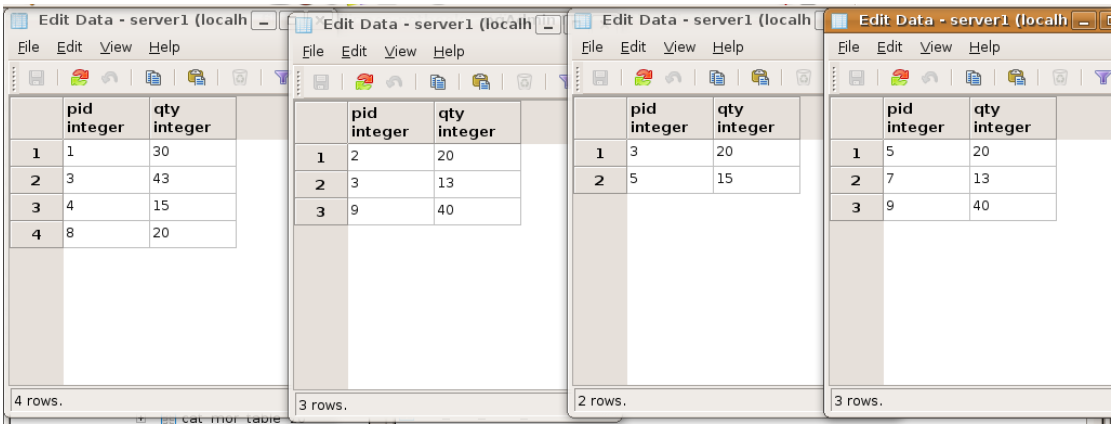
5.1.3 Ερώτημα 3

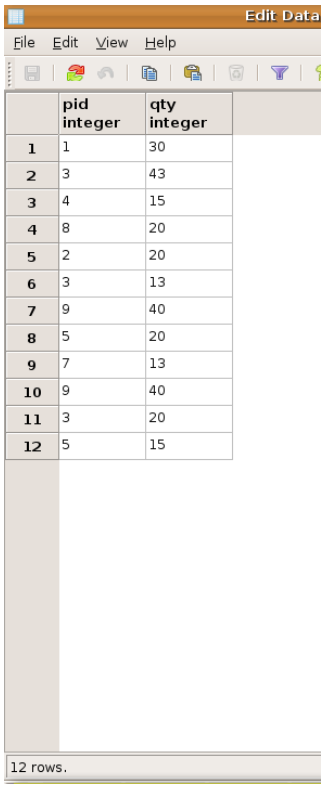
Το τρίτο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT product.pid, product.qty, product.price2
FROM   product
WITH   product::year>2005 AND product::country='GR'
WHERE  product.qty>10
```

Ουσιαστικά ζητάει την προβολή των pid και qty και price2 για όλα τα προϊόντα των οποίων το qty είναι μεγαλύτερο του 10 και για τα οποία το context instance τους αφορά την Ελλάδα για όλα τα έτη μετά το 2005. Με βάση το ερώτημα αυτό, πάνω από κάθε morph του polymorph PRODUCT που έχει ορισμένα τα attributes pid ,qty,price2 και για το οποίο ικανοποιούνται οι συνθήκες σχετικά με το context που αναφέρουν ότι το έτος θα πρέπει να είναι μεγαλύτερο του 2005 και η χώρα η Ελλάδα εκτελείται το σχεσιακό ερώτημα που προβάλλει τα attributes pid ,qty και price2 εφόσον η ποσότητα του προϊόντος είναι μεγαλύτερη του 10. Η εισαγωγή των συνθηκών σχετικά με το context έχει ως αποτέλεσμα να κοπούν από την επεξεργασία τα morphs εκείνα που ορίζονται για διαφορετικό context ενώ το attribute price2 καθότι δεν υπάρχει σε όλα τα morphs έχει ως αποτέλεσμα την περεταίρω μείωση των τελευταίων από το τελικό αποτέλεσμα.

Παρακάτω δίνεται ένα screenshot με τα αποτελέσματα και για τις 2 υλοποιήσεις.





Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις έχουμε τα ακόλουθα αποτελέσματα:

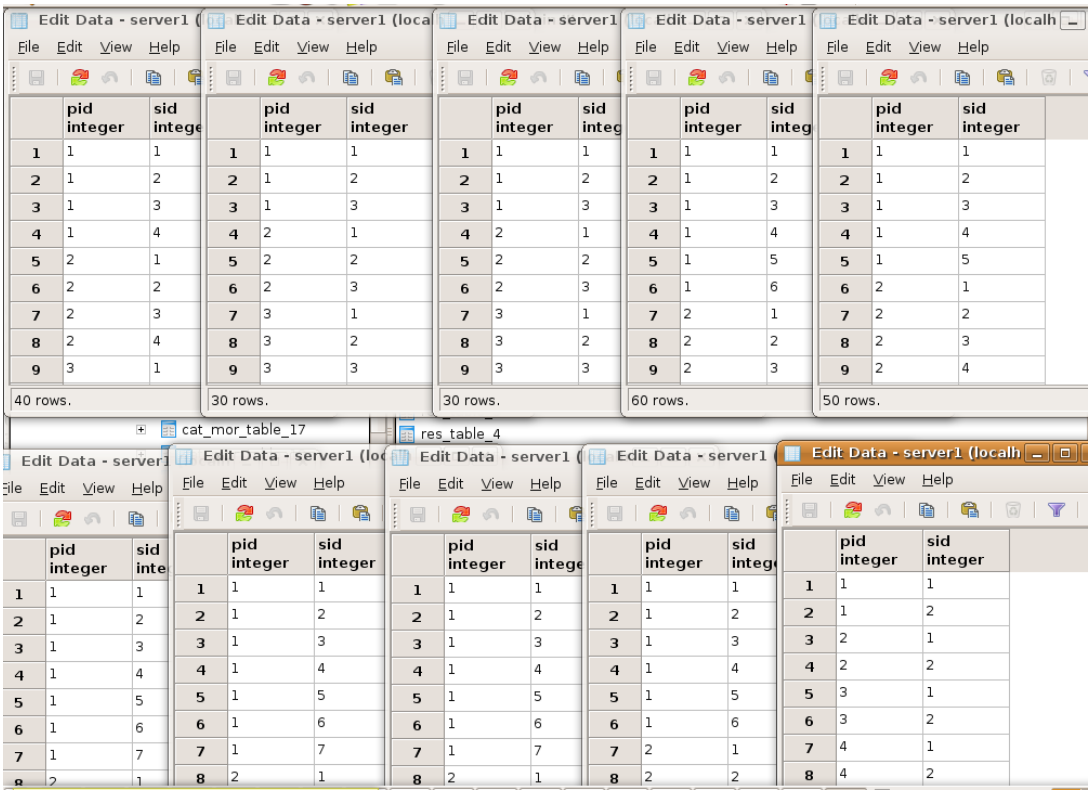
	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β	
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες
Εκτέλεση WITH	0,023	0,020	0,044	0,019
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	0,085	0,028	0,050	0,020
Τελική εκτέλεση +WHEREP	0,052	0,028	0,041	0,032
Συνολικά	0,160	0,077	0,137	0,073

5.1.4 Ερώτημα 4

Το τέταρτο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT product.pid, store.sid
FROM   product, store
```

Ουσιαστικά ζητάει την προβολή των pid,sid για όλα τα προϊόντα και όλα τα καταστήματα αντίστοιχα, σε εγγραφές όπου το κάθε pid συνδυάζεται με κάθε sid και αντίστροφα (πράξη γινομένου). Με βάση το ερώτημα αυτό, πάνω από κάθε morph του polymorph PRODUCT που έχει ορισμένο το attribute pid και πάνω από κάθε morph του polymorph STORE που έχει ορισμένο το attribute sid πραγματοποιείται μια πράξη γινομένου. Με δεδομένο ότι όλα τα morphs των PRODUCT, STORE έχουν τα attributes pid,sid αντίστοιχα το σύστημα δημιουργεί δυάδες morphs- ένα από το PRODUCT και ένα από το STORE- τα οποία ορίζονται με βάση το ίδιο context instance και στη συνέχεια για κάθε τέτοια δυάδα εκτελεί την προαναφερθείσα πράξη γινομένου. Παρακάτω δίνεται ένα screenshot με τα αποτελέσματα και για τις 2 υλοποιήσεις.



Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις έχουμε τα ακόλουθα αποτελέσματα:

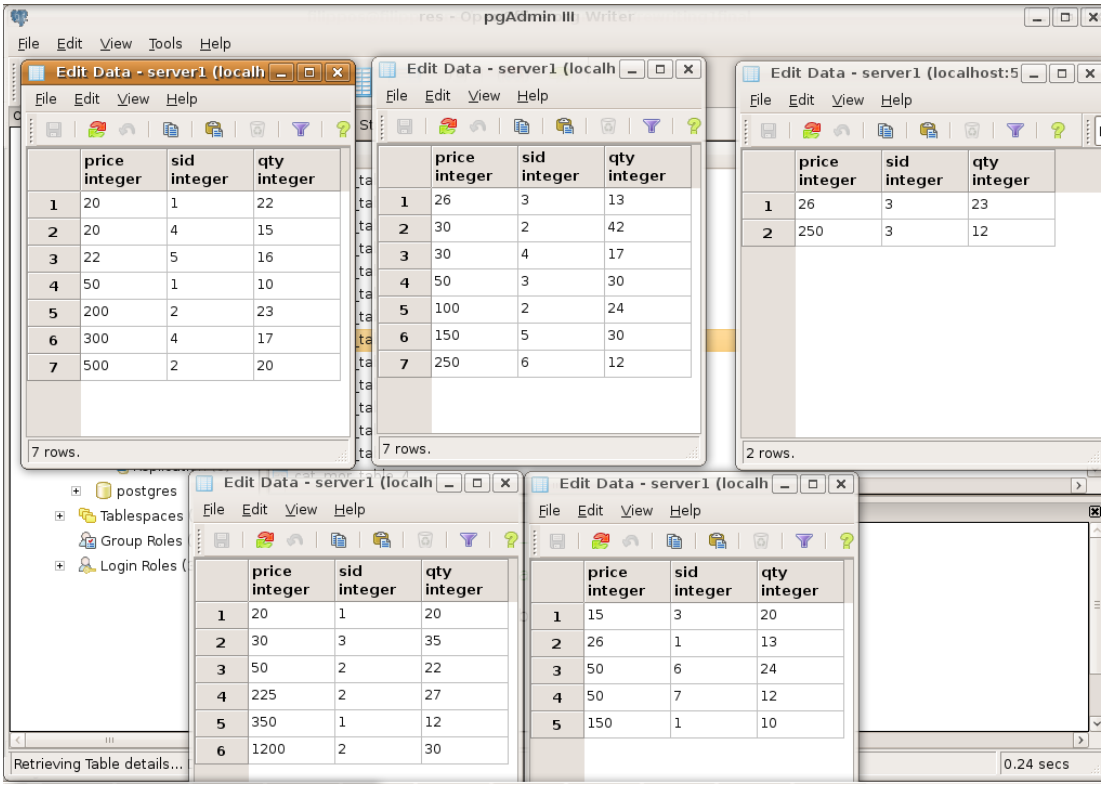
	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β	
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες
Εκτέλεση WITH	0,045	0,018	0,015	0,017
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	0,184	0,060	0,118	0,072
Τελική εκτέλεση +WHEREP	0,099	0,071	0,115	0,119
Συνολικά	0,329	0,149	0,249	0,208

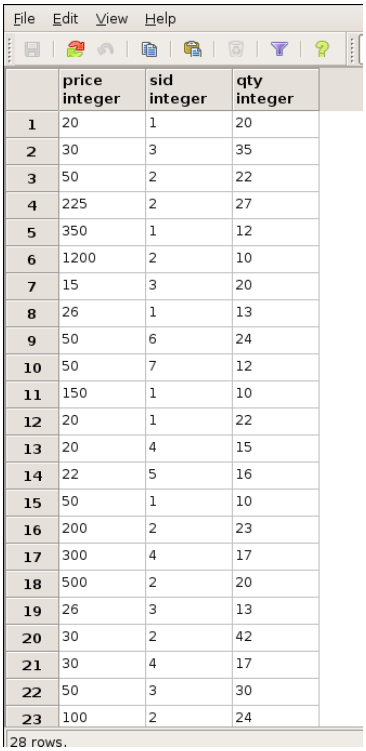
5.1.5 Ερώτημα 5

Το πέμπτο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT  product.price, store.sid, inventory.qty
FROM    product, store, inventory
WITH    product::year>2005 AND store::country='GR'
WHERE   product.price>10 AND product.qty>4 AND store.workers>5 AND
        inventory.price<150 AND inventory.qty<60 AND
        inventory.pid=product.pid
        AND inventory.sid=store.sid
```

Ουσιαστικά ζητάει την προβολή των price,sid,qty σχετικά με το ποια προϊόντα πωλούνται σε ποια καταστήματα όπου η ποσότητα του προϊόντος είναι μεγαλύτερη από 4, η τιμή του προϊόντος είναι μεγαλύτερη από 10, ο αριθμός των εργαζομένων στο κατάστημα είναι μεγαλύτερος από 5, η τιμή προμήθειας είναι μικρότερη από 150 και η ποσότητα στην αποθήκη είναι μικρότερη από 60. Επίσης το ερώτημα αυτό ορίζεται για όλα τα έτη μετά το 2005 και για χώρα την Ελλάδα. Παρακάτω δίνεται ένα screenshot με τα αποτελέσματα και για τις 2 υλοποιήσεις.


pgAdmin III
File Edit View Tools Help
Edit Data - server1 (localh
File Edit View Help
price integer | sid integer | qty integer
1 20 1 22
2 20 4 15
3 22 5 16
4 50 1 10
5 200 2 23
6 300 4 17
7 500 2 20
7 rows.
Edit Data - server1 (localh
File Edit View Help
price integer | sid integer | qty integer
1 26 3 13
2 30 2 42
3 30 4 17
4 50 3 30
5 100 2 24
6 150 5 30
7 250 6 12
7 rows.
Edit Data - server1 (localhost:5
File Edit View Help
price integer | sid integer | qty integer
1 26 3 23
2 250 3 12
2 rows.
postgres
Tablespaces
Group Roles
Login Roles
Edit Data - server1 (localh
File Edit View Help
price integer | sid integer | qty integer
1 20 1 20
2 30 3 35
3 50 2 22
4 225 2 27
5 350 1 12
6 1200 2 30
Edit Data - server1 (localh
File Edit View Help
price integer | sid integer | qty integer
1 15 3 20
2 26 1 13
3 50 6 24
4 50 7 12
5 150 1 10
Retrieving Table details...
0.24 secs


File Edit View Help
price integer | sid integer | qty integer
1 20 1 20
2 30 3 35
3 50 2 22
4 225 2 27
5 350 1 12
6 1200 2 10
7 15 3 20
8 26 1 13
9 50 6 24
10 50 7 12
11 150 1 10
12 20 1 22
13 20 4 15
14 22 5 16
15 50 1 10
16 200 2 23
17 300 4 17
18 500 2 20
19 26 3 13
20 30 2 42
21 30 4 17
22 50 3 30
23 100 2 24
28 rows.

Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις έχουμε τα ακόλουθα αποτελέσματα:

	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β	
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες
Εκτέλεση WITH	0,037	0,033	0,036	0,033
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	0,282	0,127	0,179	0,122
Τελική εκτέλεση +WHEREP	0,051	0,080	0,084	0,121
Συνολικά	0,372	0,242	0,300	0,278

5.2 Ρεαλιστικό παράδειγμα με μεγάλο όγκο δεδομένων

Στο σημείο αυτό, και αφού ολοκληρώθηκε με επιτυχία ο έλεγχος του συστήματος, είμαστε σε θέση να εκτελέσουμε ερωτήματα πάνω σε ένα μεγάλο όγκο δεδομένων ώστε να μετρήσουμε την πραγματική απόδοση τους συστήματος και το πραγματικό overhead που εισάγεται. Διότι όπως ήταν εμφανές στην προηγούμενη ενότητα οι χρόνοι εκτέλεσης ήταν παραπλήσιοι και δεν μας επέτρεπαν την εξαγωγή συνολικών συμπερασμάτων για τη λειτουργία του συστήματος. Παρ' όλα αυτά είναι γεγονός ότι ακόμα και σε αυτούς τους μικρούς χρόνους καταφέραμε να διαπιστώσουμε αυτά που περιμέναμε θεωρητικά. Για παράδειγμα, διαπιστώσαμε ότι η διαδικασία δημιουργίας προσωρινών πινάκων τελικά δεν επιδρά θετικά στην απόδοση. Ωστόσο στην ενότητα αυτή, τα συμπεράσματα θα μπορούν να εξαχθούν πέραν αμφιβολίας καθώς οι διαφορές στους χρόνους (λόγω του τεράστιου όγκου δεδομένων) θα είναι αρκετά σημαντικές.

Στην κατεύθυνση αυτή και διατηρώντας σε γενικές γραμμές το σενάριο της προηγούμενης ενότητας σχετικά με την ηλεκτρονική αποθήκη, στην προσπάθεια να μεγαλώσει ο όγκος των δεδομένων χρησιμοποιήσαμε ως Context Schema το <YEAR,WEEK,COUNTRY,USERROLE> για το οποίο το attribute YEAR παρέμεινε σταθερό στο 2010 (πιθανώς σε μελλοντικά πειράματα να μεταβάλλεται και αυτό), το attribute WEEK μεταβάλλεται από 0 μέχρι 51, το attribute COUNTRY παίρνει τιμές για 10 χώρες ενώ το attribute USERROLE παίρνει τις τιμές 'wholesale' και 'retail'. Όπως γίνεται αντιληπτό αν τα συνδυάσουμε πρόκειται για 1 x 52 x 10 x 2=1040 διαφορετικά context instances.

Όσον αφορά τα Polymorphs που χρησιμοποιήσαμε αυτά διατηρήθηκαν ως PRODUCT, STORE και INVENTORY, και εφόσον έχουμε διαθέσιμα 1040 context instances δημιουργήσαμε 1040 morphs σε κάθε ένα από αυτά. Παρακάτω δίνονται αναλυτικά τα attributes των morph για κάθε polymorph.

PRODUCT

Το γενικό σχήμα των morphs του polymorph PRODUCT αποτελείται από τα attributes:

- Pid: είναι ο κωδικός του προϊόντος και πρόκειται για ακέραιο μεγέθους 8 bytes
- Name: είναι το όνομα του προϊόντος και πρόκειται για string μεγέθους 10 bytes
- Descr: είναι η περιγραφή του προϊόντος και πρόκειται για string μεγέθους 81 bytes

Σε αυτά προστίθεται το παρακάτω attribute το οποίο όμως δεν ορίζεται για όλα τα morphs:

- Vat: είναι ο φόρος του προϊόντος και πρόκειται για ακέραιο μεγέθους 2 bytes

STORE

Το γενικό σχήμα των morphs του polymorph STORE αποτελείται από τα attributes:

- Sid: είναι ο κωδικός του καταστήματος και πρόκειται για ακέραιο μεγέθους 8 bytes
- Name: είναι το όνομα του καταστήματος και πρόκειται για string μεγέθους 10 bytes
- Type: είναι ο τύπος του καταστήματος και πρόκειται για string μεγέθους 2 bytes. Οι τιμές που μπορεί να πάρει είναι ‘es’ (e-shop) και ‘rs’ (regular shop)
- Descr: είναι η περιγραφή του καταστήματος και πρόκειται για string μεγέθους 80 bytes

INVENTORY

Το γενικό σχήμα των morphs του polymorph INVENTORY αποτελείται από τα attributes:

- Pid: είναι ο κωδικός του προϊόντος που πωλείται σε κάποιο κατάστημα και πρόκειται για ακέραιο μεγέθους 8 bytes
- Sid: είναι ο κωδικός του καταστήματος που πουλάει το συγκεκριμένο προϊόν και πρόκειται για ακέραιο μεγέθους 8 bytes
- Price: είναι η τιμή στην οποία πωλείται το προϊόν και πρόκειται για ακέραιο μεγέθους 4 bytes

Διαδικασία δημιουργίας της βάσης

Εφόσον έχουμε αναφέρει το Context Schema αλλά και τα Polymorphs που θα χρησιμοποιηθούν στο παράδειγμα, μπορούμε να περιγράψουμε τη διαδικασία που ακολουθήθηκε.

Αρχικά ορίσαμε μια μεταβλητή με τιμή 40% ώστε να επιλεγούν οι 4 χώρες από τις 10 του Context Schema. Για τα context instances αυτών των χωρών, τα morphs του PRODUCT έχουν ορισμένο το attribute vat. Οπότε εκτελώντας μια επαναληπτική διαδικασία δημιουργήθηκαν 1040 morphs με 10.000 προϊόντα για κάθε ένα από αυτά, για τα οποία το pid ορίστηκε ως μια μεταβλητή που αυξανόταν σε κάθε επανάληψη ενώ τα attributes name και descr προέκυψαν από τυχαία strings αναλόγου μεγέθους κάθε φορά. Το attribute vat όπου υπήρχε προέκυψε από μια τυχαία μεταβλητή με εύρος τιμών από 5 έως 20. Έτσι αποθηκεύτηκαν τα δεδομένα του PRODUCT είτε σε 1040 διαφορετικούς πίνακες (υλοποίηση A) είτε σε έναν πίνακα (υλοποίηση B).

Στη συνέχεια όσον αφορά τη δημιουργία των morphs του STORE, δημιουργήθηκαν 1040 με 1000 καταστήματα για κάθε ένα από αυτά για τα οποία το sid ορίστηκε ως μια μεταβλητή που αυξανόταν σε κάθε επανάληψη ενώ τα attributes name και descr προέκυψαν από τυχαία strings αναλόγου μεγέθους κάθε φορά. Το attribute type με πιθανότητα 75% ορίστηκε ως ‘es’. Στις εναπομείνασες περιπτώσεις (25%) τέθηκε ως ‘rs’. Έτσι αποθηκεύτηκαν τα δεδομένα του STORE είτε σε 1040 διαφορετικούς πίνακες (υλοποίηση A) είτε σε έναν πίνακα (υλοποίηση B).

Ακολούθως δημιουργήθηκαν 1040 morphs για το polymorph INVENTORY. Κάθε ένα από τα οποία θα χρησιμοποιηθεί για την εκτέλεση των join conditions, δηλαδή θα περιέχει πληροφορίες για το ποιο προϊόν πωλείται σε πιο κατάστημα. Έτσι ορίστηκε μια παράμετρος που με τυχαίο τρόπο επιλέγει για κάθε προϊόν σε κάθε morph σε πόσα καταστήματα πωλείται ώστε να προστεθούν οι αντίστοιχες tuples στον πίνακα και η τιμή της παραμέτρου αυτής κυμαίνεται από 1 μέχρι 5. Ανάλογα με την τιμή αυτή, έστω ν, επιλέγονται ν τυχαίες τιμές από τα sid του STORE οπότε κατά μέσο όρο υπάρχουν 30.000 tuples σε κάθε morph. Έτσι σε κάθε tuple αποθηκεύεται το pid του προϊόντος, το κατάστημα στο οποίο πωλείται (sid) και η τιμή του η οποία ορίστηκε με πιθανότητα 50% να είναι από 1 έως 50 και με το υπόλοιπο 50% από 50 και μέχρι 300. Ανάλογα με το που κυμαίνεται η τιμή αυτή (0-50 ή 51-300) μια άλλη συνάρτηση την παράγει με τυχαίο τρόπο. Έτσι αποθηκεύτηκαν τα δεδομένα του STORE είτε σε 1040 διαφορετικούς πίνακες (υλοποίηση A) είτε σε έναν πίνακα (υλοποίηση B).

Από τη στιγμή που έχει ολοκληρωθεί επιτυχώς η δημιουργία της βάσης δεδομένων και πριν εκτελέσουμε τα ερωτήματα κρίνεται σκόπιμο να αναφέρουμε το μέγεθος της.

PRODUCT

1 tuple-> 8+10+81+2*(0.4)=99,8 Bytes.

1 morph->10.000 tuples->0.951 MB

1 polymorph->1040 morphs-> 990 MB

STORE

1 tuple-> 8+10+80+2=100 Bytes.

1 morph->1.000 tuples->0.0953 MB

1 polymorph->1040 morphs-> 99 MB

INVENTORY

1 tuple-> 8+8+4=20 Bytes.

1 morph->30.000 tuples->0.572 MB

1 polymorph->1040 morphs-> 595 MB

Άρα προσεγγιστικά το μέγεθος της βάσης αγγίζει το 1,7 GB. Παρακάτω παρατίθενται τα ερωτήματα που εκτελέστηκαν για τα οποία να σημειώσουμε ότι εκτελέστηκαν σε netbook με επεξεργαστή N270 atom 1.6 GHZ, RAM 1 GB DDR2 και HD 120 GB (5400 rpm). Τέλος να σημειώσουμε ότι συγκρίνουμε τον αντίστοιχο χρόνο εκτέλεσης κάθε υλοποίησης για κάθε τεχνική με τον μέσο όρο της απευθείας εκτέλεσης των σχεσιακών ερωτημάτων στο ΣΔΒΔ της POSTGRES προσομοιώνοντας κατά κάποιο τρόπο τη διαδικασία εκτέλεσης των ερωτημάτων «με το χέρι», με απώτερο σκοπό τη μέτρηση του overhead που εισάγεται από το σύστημα.

5.2.1 Ερώτημα 1

Το πρώτο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT product.pid, inventory.price, product.vat
FROM   product, store, inventory
WITH   product::userrole='retail' AND product::country='UK'  AND
       product::week<5 AND  store::userrole='retail' AND
       store::country='UK' AND  store::week<5 AND
       inventory::userrole='retail' AND inventory::country='UK'  AND
       inventory::week<5
WHERE  inventory.price>50 AND store.type='rs' AND
       inventory.pid=product.pid AND inventory.sid=store.sid
```

Το ερώτημα αυτό ζητάει την προβολή των pid, price και vat για όσα προϊόντα πωλούνται σε τιμή μεγαλύτερη των 50 και σε καταστήματα που είναι e-shops. Επίσης μας ενδιαφέρουν μόνο τα αποτελέσματα που αφορούν καταστήματα που είναι στη χώρα UK και πουλούν σε λιανική ενώ τα αποτελέσματα περιορίζονται περεταίρω στις πρώτες 5 εβδομάδες.

Πιο αναλυτικά, τελικά από το ερώτημα αυτό καταλήγουν να εκτελούνται 5 σχεσιακά ερωτήματα αφού ο περιορισμός της χώρας ικανοποιείται σε 104 morphs, ο επιπρόσθετος περιορισμός σχετικά με τον τρόπο διάθεσης των προϊόντων μειώνει αυτά κατά το ήμισυ ενώ από τα εναπομείναντα 52 morphs μόνο τα 5 ικανοποιούν και τη συνθήκη σχετικά με τις εβδομάδες. Αν λάβουμε υπ' όψιν μας ότι

- Ο αριθμός των προϊόντων είναι 10.000
- Το 50% έχουν τιμή πάνω από 50
- Κάθε προϊόν υπάρχει σε 1 έως 3 καταστήματα
- το 25% των καταστημάτων είναι regular shops

τότε προσεγγιστικά το πλήθος των εγγραφών των αποτελεσμάτων είναι 10.000*0.5*(1+5)/2*0.25=3750

Το μέγεθος της tuple είναι: 8+4+2=14 Bytes. Άρα το συνολικό μέγεθος του αποτελέσματος είναι 0,25 MB.

Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις αλλά και την εκτέλεση των ερωτημάτων χωρίς τη χρήση του context-aware συστήματος έχουμε τα ακόλουθα αποτελέσματα:

	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β		Εκτέλεση σχεσιακών ερωτημάτων χωρίς τη χρήση του context-aware RDBMS (υλοποίηση Α,Β)
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	
Εκτέλεση WITH	0,416	0,294	0,096	0,070	-
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	1,314	0,520	87,23	0,52	-
Τελική εκτέλεση +WHEREP	0,774	0,755	1,236	410,2	-
Συνολικά	2,505	1,57	88,565	410,799	1,020(A) 362(B)

5.2.1 Ερώτημα 2

Το δεύτερο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT product.pid, inventory.price, product.vat
FROM   product, store, inventory
WITH   product::userrole='retail' AND product::country='UK'
       AND  store::userrole='retail' AND store::country='UK'
       AND  inventory::userrole='retail' AND inventory::country='UK'
WHERE  inventory.price>50 AND store.type='es' AND
       inventory.pid=product.pid AND inventory.sid=store.sid
```

Το ερώτημα αυτό αποτελεί μια παραλλαγή του ερωτήματος 1 καθώς αφαιρείται από αυτό ο περιορισμός σχετικά με τις 5 πρώτες εβδομάδες. Αυτό έχει ως συνέπεια να αυξηθεί ο τελικός αριθμός των morphs που ικανοποιούν τις συνθήκες context από 5 σε 52. Έτσι τελικά από το ερώτημα αυτό καταλήγουν να εκτελούνται 52 σχεσιακά ερωτήματα αφού ο περιορισμός της

χώρας ικανοποιείται σε 104 morphs και ο επιπρόσθετος περιορισμός σχετικά με τον τρόπο διάθεσης των προϊόντων μειώνει αυτά κατά το ήμισυ. Αν λάβουμε υπ' όψιν μας ότι

- Ο αριθμός των προϊόντων είναι 10.000
- Το 50% έχουν τιμή πάνω από 50
- Κάθε προϊόν υπάρχει σε 1 έως 3 καταστήματα
- το 75% των καταστημάτων είναι e-shops

τότε προσεγγιστικά το πλήθος των εγγραφών των αποτελεσμάτων είναι 10.000*0.5*(1+5)/2*0.75=11250

Το μέγεθος της tuple είναι: 8+4+2=14 Bytes. Άρα το συνολικό μέγεθος του αποτελέσματος είναι 7,81 MB.

Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις αλλά και την εκτέλεση των ερωτημάτων χωρίς τη χρήση του context-aware συστήματος έχουμε τα ακόλουθα αποτελέσματα:

	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β		Εκτέλεση σχεσιακών ερωτημάτων χωρίς τη χρήση του context-aware RDBMS
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	
Εκτέλεση WITH	0,068	0,073	0,161	0,084	-
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	14,706	4,663	411,67	4,857	-
Τελική εκτέλεση +WHEREP	16,346	14,566	70,42	4355	-
Συνολικά	31,121	19,303	482,26	4359,94	10,608(A) 3764,8(B)

5.2.3 Ερώτημα 3

Το τρίτο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT product.pid, inventory.price, product.vat
FROM   product, store, inventory
WITH   product::userrole='retail' AND product::country='UK'  AND
       product::week=23 AND  store::userrole='retail' AND
       store::country='UK' AND store::week=23 AND
       inventory::userrole='retail' AND inventory::country='UK' AND
       inventory::week=23
WHERE  inventory.price>50 AND store.type='es' AND
       inventory.pid=product.pid AND inventory.sid=store.sid
```

Το ερώτημα αυτό αποτελεί μια παραλλαγή του ερωτήματος 1 καθώς προσθέτει σε αυτό έναν επιπλέον περιορισμό όσον αφορά το context αφού τα αποτελέσματα περιορίζονται σε μια μόνο εβδομάδα. Η προσθήκη αυτή μειώνει σημαντικά τον αριθμό των τελικών ερωτημάτων που θα εκτελεστούν καθώς μόνο ένα morph από κάθε polymorph θα ικανοποιεί την εν λόγω συνθήκη. Ακόμα αποτελεί και ένα περισσότερο ρεαλιστικό παράδειγμα προς εκτέλεση καθώς εφόσον ο χρήστης επιλέξει να εκτελέσει κάποιο context-aware ερώτημα, το πιο πιθανό είναι να ορίσει επακριβώς τις συνθήκες context που θα το αφορούν παρά να το εκτελέσει για κάποιο εύρος αυτών. Και αυτός ακριβώς είναι και ο σκοπός της εκτέλεσης του ερωτήματος αυτού, να εκτελεστεί δηλαδή τελικά ένα SQL ερώτημα ώστε να μετρηθεί η απόδοση του συστήματος και να συγκριθεί με την εκτέλεση με το χέρι απευθείας στο σχεσιακό ΣΔΒΔ.

Πιο αναλυτικά ο περιορισμός της χώρας ικανοποιείται σε 104 morphs, ο επιπρόσθετος περιορισμός σχετικά με τον τρόπο διάθεσης των προϊόντων μειώνει αυτά κατά το ήμισυ ενώ από τα εναπομείναντα 52 morphs μόνο το 1 ικανοποιεί και τη συνθήκη σχετικά με τις εβδομάδες. Αν λάβουμε υπ' όψιν μας ότι

- Ο αριθμός των προϊόντων είναι 10.000
- Το 50% έχουν τιμή πάνω από 50
- Κάθε προϊόν υπάρχει σε 1 έως 3 καταστήματα
- το 75% των καταστημάτων είναι e-shops

τότε προσεγγιστικά το πλήθος των εγγραφών των αποτελεσμάτων είναι 10.000*0.5*(1+5)/2*0.75=11250

Το μέγεθος της tuple είναι: 8+4+2=14 Bytes. Άρα το συνολικό μέγεθος του αποτελέσματος είναι 0,15 MB.

Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις αλλά και την εκτέλεση του ερωτήματος χωρίς τη χρήση του context-aware συστήματος έχουμε τα ακόλουθα αποτελέσματα:

	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β		Εκτέλεση σχεσιακών ερωτημάτων χωρίς τη χρήση του context-aware RDBMS
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	
Εκτέλεση WITH	0,095	0,066	0,070	0,073	-
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	0,295	0,144	73,080	0,172	-
Τελική εκτέλεση +WHEREP	0,288	0,243	0,322	81,236	-
Συνολικά	0,680	0,453	73,473	81,482	0,204(A) 72,4(B)

5.2.4 Ερώτημα 4

Το τέταρτο ερώτημα που εκτελέστηκε δίνεται στη παρακάτω μορφή στη γλώσσα extended-sql:

```
SELECT product.pid, inventory.price, product.vat
FROM   product, store, inventory
WITH   product::userrole='retail' AND  store::userrole='retail'
       AND  inventory::userrole='retail'
WHERE  inventory.price>50 AND store.type='es' AND
       inventory.pid=product.pid AND inventory.sid=store.sid
```

Το ερώτημα αυτό αποτελεί μια παραλλαγή του ερωτήματος 1 καθώς από αυτό αφαιρείται ο

περιορισμός σχετικά με τη χώρα. Η αλλαγή αυτή επιδρά σημαντικά στον τελικό αριθμό των ερωτημάτων που θα εκτελεστούν.

Πιο αναλυτικά, τελικά από το ερώτημα αυτό καταλήγουν να εκτελούνται 208 σχεσιακά ερωτήματα αφού ο περιορισμός σχετικά με τον τρόπο διάθεσης των προϊόντων μειώνει κατά το ήμισυ τα αρχικά 1040 morphs ενώ από τα εναπομείναντα 520 morphs μόνο για το 40% αυτών ορίζεται το attribute vat, οπότε τελικά μένουν 208. Το ερώτημα αυτό θα μπορούσε να χρησιμεύσει μόνο σε κάποιου data analysis, διαφορετικά δεν υπάρχει νόημα στην εκτέλεση του. Αν λάβουμε υπ' όψιν μας ότι

- Ο αριθμός των προϊόντων είναι 10.000
- Το 50% έχουν τιμή πάνω από 50
- Κάθε προϊόν υπάρχει σε 1 έως 3 καταστήματα
- το 75% των καταστημάτων είναι e-shops

τότε προσεγγιστικά το πλήθος των εγγραφών των αποτελεσμάτων είναι 10.000*0.5*(1+5)/2*0.75=11.250

Το μέγεθος της tuple είναι: 8+4+2=14 Bytes. Άρα το συνολικό μέγεθος του αποτελέσματος είναι 31,2 MB.

Όσον αφορά τις διαφοροποιήσεις στο χρόνο απόκρισης ανάμεσα στις δύο υλοποιήσεις αλλά και την εκτέλεση του ερωτήματος χωρίς τη χρήση του context-aware συστήματος έχουμε τα ακόλουθα αποτελέσματα:

	ΥΛΟΠΟΙΗΣΗ Α		ΥΛΟΠΟΙΗΣΗ Β		Εκτέλεση σχεσιακών ερωτημάτων χωρίς τη χρήση του context-aware RDBMS
	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	Με ενδιάμεσους πίνακες	Χωρίς Ενδιάμεσους πίνακες	
Εκτέλεση WITH	0,398	0,257	0,163	0.126	-
Έλεγχος attributes + WHERE (+προσωρινοί πίνακες)	204,019	45,22	2704,28	42.12	-
Τελική εκτέλεση +WHEREP	98,685	81,57	1762,39	14593	-
Συνολικά	303,103	127,053	4466,83	14635.2	42,43(A) 15059,2(B)

5.3 Συμπεράσματα

Από τη στιγμή που ολοκληρώθηκαν τα πειράματα μπορούμε να συνοψίσουμε τα συμπεράσματα που προέκυψαν από την εκτέλεση τους.

Αρχικά από τα πειράματα πάνω στα λίγα δεδομένα μπορούμε να πούμε ότι για ερωτήματα πάνω από μικρούς πίνακες όσον αφορά τον όγκο των δεδομένων τους, η υλοποίηση Β φαίνεται να είναι ελαφρώς ταχύτερη αν και οι χρόνοι είναι τόσοι μικροί που οι διαφορές δεν επιτρέπουν την εξαγωγή ασφαλών συμπερασμάτων. Ακόμα, η τεχνική της δημιουργίας προσωρινών πινάκων πάνω από τους οποίους εκτελούνται τελικά τα ερωτήματα δεν φαίνεται να επιδρά θετικά σε καμία από τις δύο υλοποιήσεις αφού ο χρόνος που απαιτείται για τη δημιουργία των πινάκων αυτών είναι μεγαλύτερος συγκριτικά με εκείνον που «χάνεται» αν το ερώτημα εκτελεστεί πάνω από έναν λίγο μεγαλύτερο πίνακα (τον αρχικό).

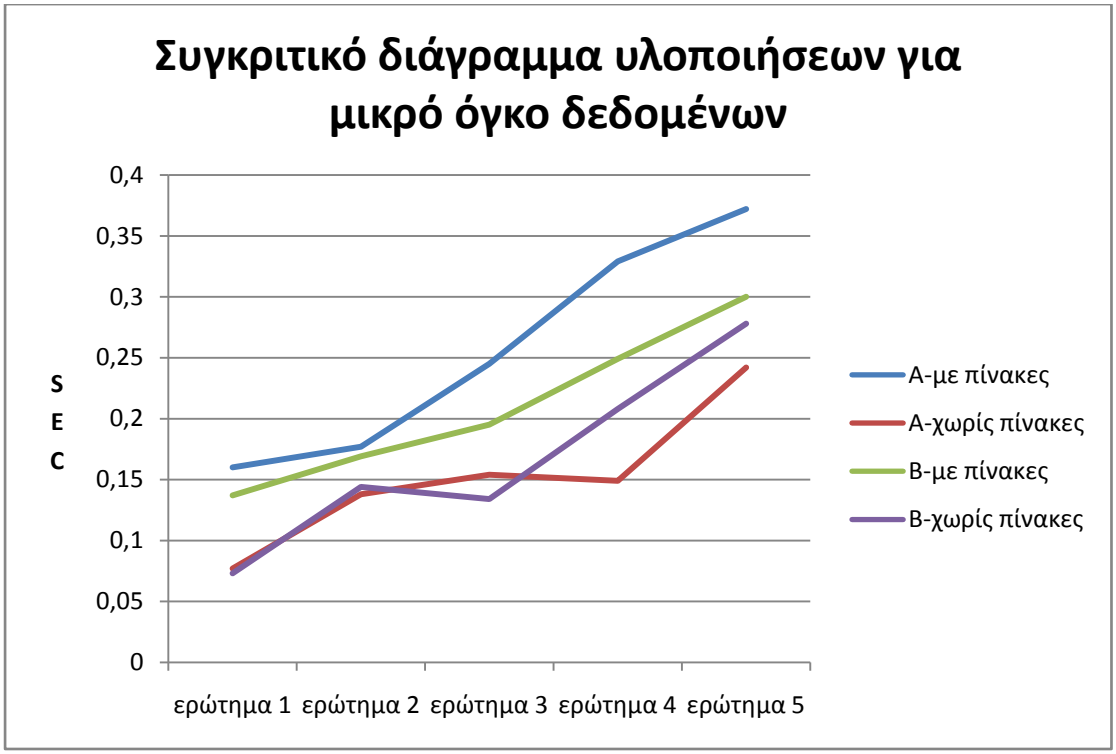
Θεωρητικά, αν μπορούσαμε να προβλέψουμε την απόδοση του συστήματος θα μπορούσαμε να ισχυριστούμε ότι για μικρούς πίνακες η υλοποίηση Β είναι λογικό να υπερτερεί της Α ενώ όσο μεγαλώνει ο όγκος των δεδομένων τόσο η υλοποίηση Α θα αρχίζει να ξεπερνάει σε απόδοση τη Β.

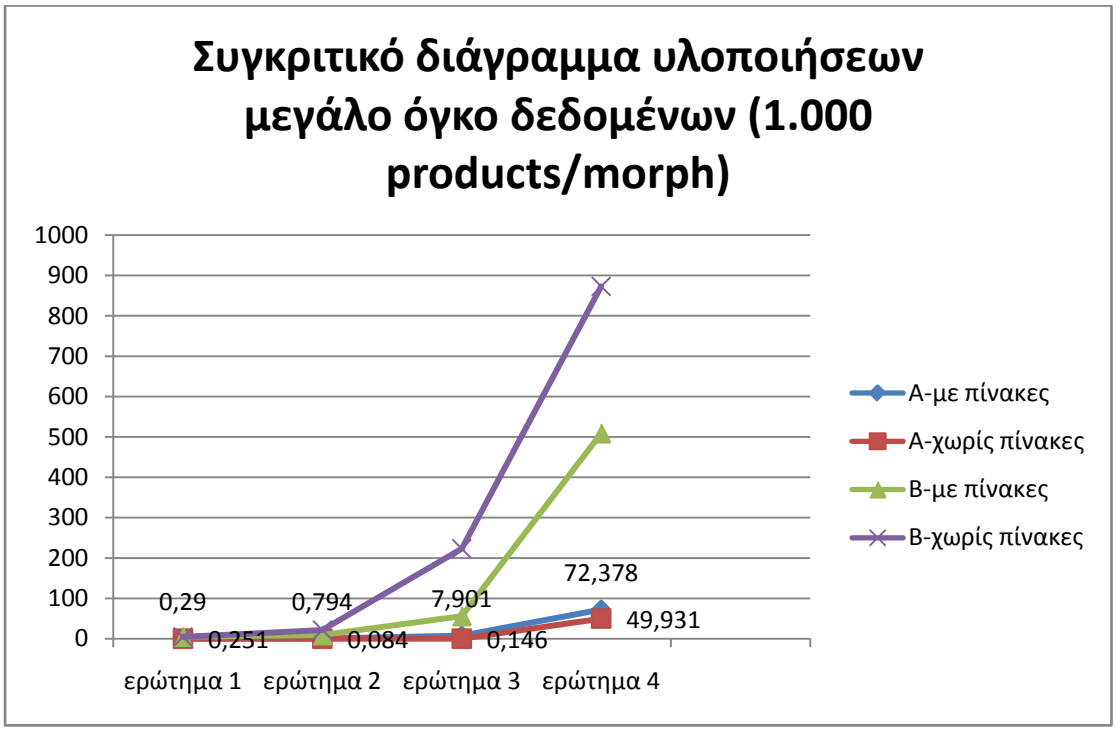
Στο δεύτερο μέρος των πειραμάτων παρατηρούμε ακριβώς αυτό. Ο όγκος των εγγραφών είναι τόσο μεγάλος (το polymorph inventory έχει 31.200.000 tuples, το μέγεθος του πίνακα είναι ~600MB ενώ εκείνου του product είναι ~1 GB) ώστε η διαφορά στην απόδοση της υλοποίησης Α από τη Β είναι αρκετά σημαντική. Αυτή είναι τόσο μεγάλη καθώς λόγω του μεγέθους τους, οι πίνακες δεν είναι δυνατόν να σηκωθούν στη μνήμη και έτσι όλα τα joins γίνονται πάνω από το δίσκο.

Επίσης παρατηρούμε ότι στην περίπτωση αυτή έχει νόημα η δημιουργία των ενδιάμεσων-προσωρινών πινάκων καθώς με τον τρόπο αυτό μειώνεται αρκετά ο πολύ μεγάλος αρχικός πίνακας με τα δεδομένα με αποτέλεσμα τα ερωτήματα να εκτελούνται πάνω σε μικρότερους και άρα ταχύτερα.

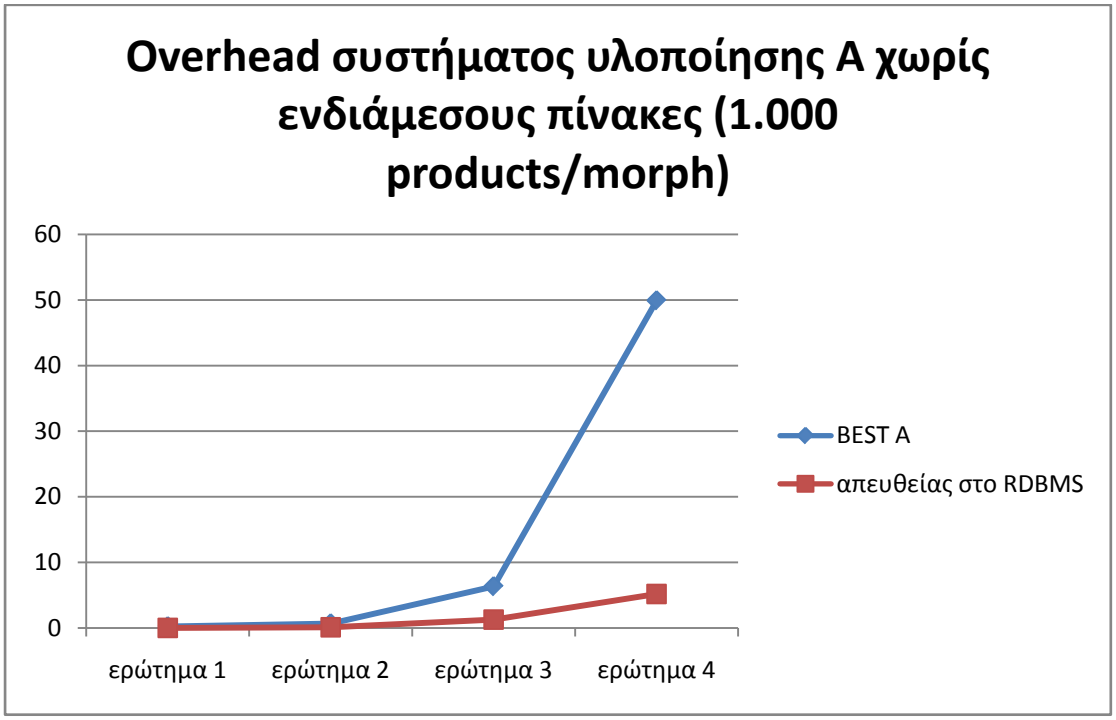
Στο δεύτερο μέρος των πειραμάτων γίνεται και μια σύγκριση ανάμεσα στην εκτέλεση του ερωτήματος από το υποσύστημα μετατροπής και εκτέλεσης των ερωτημάτων σε σχέση με την απευθείας εκτέλεση στο ΣΔΒΔ των σχεσιακών ερωτημάτων που προκύπτουν. Από τη σύγκριση αυτή παρατηρούμε ότι για την υλοποίηση Α το overhead του συστήματος ουσιαστικά αφορά το κομμάτι στο οποίο ελέγχεται για κάθε morph αν περιλαμβάνει τα ζητούμενα attributes. Όσον αφορά την υλοποίηση Β δεν εισάγεται κάποια σημαντική καθυστέρηση από το σύστημα, απλά λόγω του μεγάλου όγκου των δεδομένων ο τρόπος αποθήκευσης αυτός δεν ενδείκνυται.

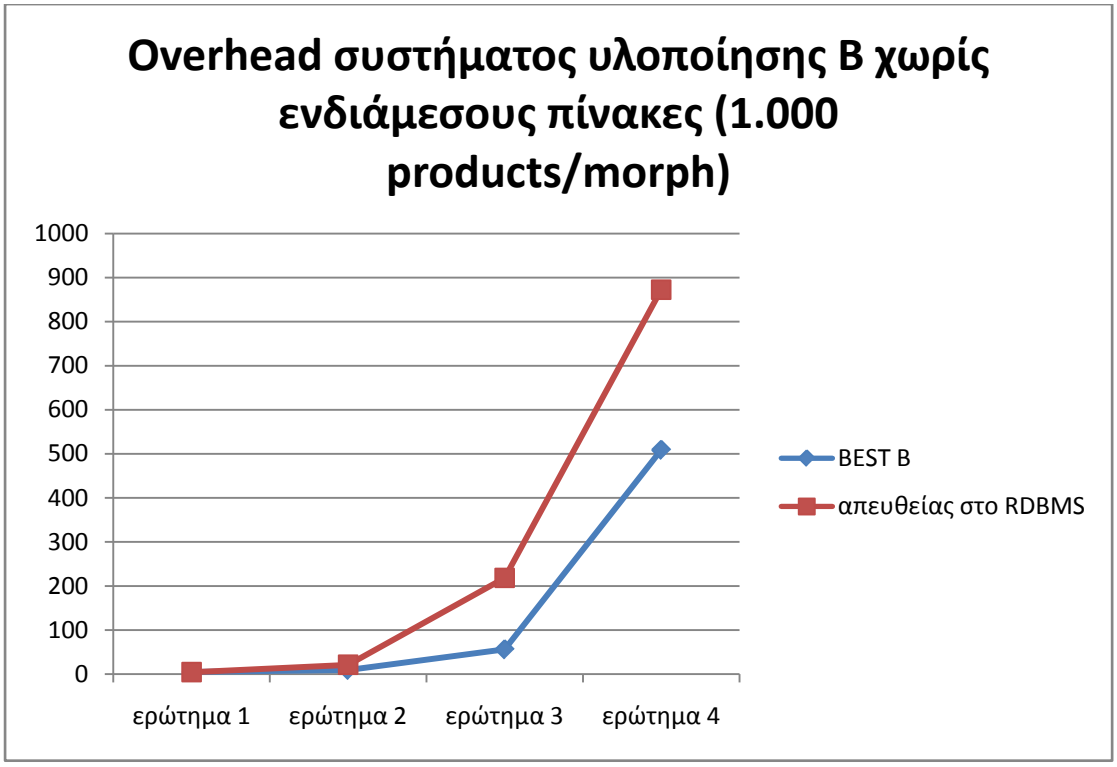
Συνοψίζοντας, καταλήγουμε στο συμπέρασμα ότι η υλοποίηση Α χωρίς ενδιάμεσους πίνακες είναι η πλέον αποδοτική. Αν παρατηρήσουμε τους χρόνους εκτέλεσης των ερωτημάτων σε αυτή, θα δούμε ότι ουσιαστικά το σύστημα εισάγει μια καθυστέρηση περίπου της τάξης του 10% σε σχέση με την απευθείας εκτέλεση των ερωτημάτων στο RDBMS στο τελευταίο στάδιο δηλαδή εκεί όπου τελικά εκτελούνται τα ερωτήματα και επιστρέφεται το αποτέλεσμα. Στο πρώτο στάδιο, εκεί δηλαδή όπου εκτελούνται οι συνθήκες σχετικά με το context ο χρόνος εκτέλεσης είναι αμελητέος (~0.1% του συνολικού χρόνου εκτέλεσης του ερωτήματος) ενώ στο δεύτερο στάδιο, δηλαδή εκεί που γίνεται ο έλεγχος στον κατάλογο για τα attributes του κάθε morph, έχουμε συνολικό χρόνο ανάλογο με τον αριθμό των morphs αφού για κάθε morph εκτελείται και ένα ερώτημα. Ο έλεγχος αυτός όμως γίνεται πάνω στους πίνακες του καταλόγου τους οποίους λόγω του μικρού τους μεγέθους μπορούμε να σηκώσουμε στη μνήμη και συνεπώς να εξαλείψουμε σε σημαντικό βαθμό την καθυστέρηση αυτή. Έτσι τελικά με αυτήν την υλοποίηση δεν εισάγεται σημαντική καθυστέρηση στο RDMS ενώ εισάγονται οι νέες λειτουργίες σχετικά με το context. Παρακάτω δίνονται διαγράμματα που απεικονίζουν τα συμπεράσματα που αναφέρθηκαν παραπάνω και σε αυτά τα ερωτήματα έχουν τοποθετηθεί σε αύξουσα σειρά ανάλογα με το πόσα σχεσιακά ερωτήματα «γεννώνται» σε κάθε περίπτωση.

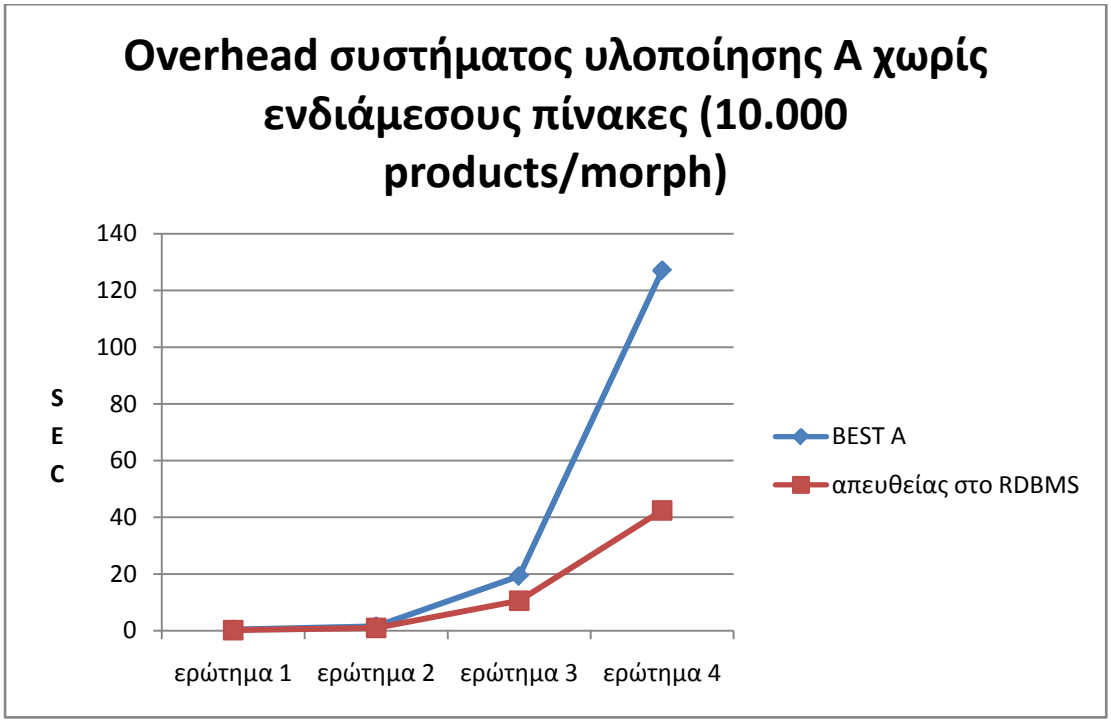

Συγκριτικό διάγραμμα υλοποιήσεων για μικρό όγκο δεδομένων
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
SEC
ερώτημα 1
ερώτημα 2
ερώτημα 3
ερώτημα 4
ερώτημα 5
A-με πίνακες
A-χωρίς πίνακες
B-με πίνακες
B-χωρίς πίνακες

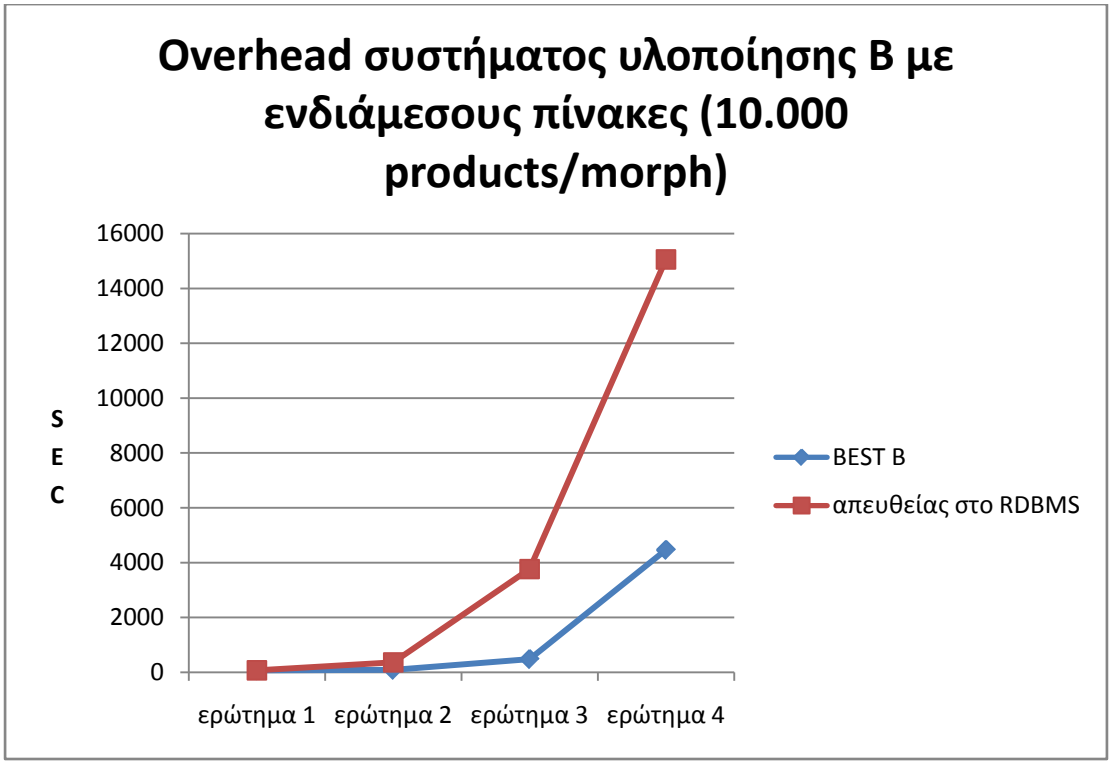

Συγκριτικό διάγραμμα υλοποιήσεων μεγάλο όγκο δεδομένων (1.000 products/morph)
1000
900
800
700
600
500
400
300
200
100
0
0,29
0,794
7,901
72,378
0,251
0,084
0,146
49,931
ερώτημα 1
ερώτημα 2
ερώτημα 3
ερώτημα 4
A-με πίνακες
A-χωρίς πίνακες
B-με πίνακες
B-χωρίς πίνακες


Συγκριτικό διάγραμμα υλοποιήσεων μεγάλο όγκο δεδομένων (10.000 products/morph)
16000
14000
12000
10000
8000
6000
4000
2000
0
SEC
0,68
2,505
31,121
303,103
0,453
1,57
19,303
127,053
ερώτημα 1
ερώτημα 2
ερώτημα 3
ερώτημα 4
Α-με πίνακες
Α-χωρίς πίνακες
Β-με πίνακες
Β-χωρίς πίνακες


Overhead συστήματος υλοποίησης Α χωρίς ενδιάμεσους πίνακες (1.000 products/morph)
60
50
40
30
20
10
0
ερώτημα 1
ερώτημα 2
ερώτημα 3
ερώτημα 4
BEST A
απευθείας στο RDBMS


Overhead συστήματος υλοποίησης B χωρίς ενδιάμεσους πίνακες (1.000 products/morph)
1000
900
800
700
600
500
400
300
200
100
0
ερώτημα 1
ερώτημα 2
ερώτημα 3
ερώτημα 4
BEST B
απευθείας στο RDBMS


Overhead συστήματος υλοποίησης A χωρίς ενδιάμεσους πίνακες (10.000 products/morph)
SEC
140
120
100
80
60
40
20
0
ερώτημα 1
ερώτημα 2
ερώτημα 3
ερώτημα 4
BEST A
απευθείας στο RDBMS


Overhead συστήματος υλοποίησης B με ενδιάμεσους πίνακες (10.000 products/morph)
SEC
16000
14000
12000
10000
8000
6000
4000
2000
0
ερώτημα 1
ερώτημα 2
ερώτημα 3
ερώτημα 4
BEST B
απευθείας στο RDBMS

6

Επίλογος

Στην ενότητα αυτή αρχικά συνοψίζουμε τα αποτελέσματα τις διπλωματικής εργασίας (ενότητα 6.1) και στη συνέχεια (ενότητα 6.2) παραθέτουμε τις πιθανές μελλοντικές επεκτάσεις, που έχει νόημα να υλοποιηθούν.

6.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία υλοποιήθηκε το υποσύστημα μετατροπής σε PL\SQL και εκτέλεσης των ερωτημάτων. Το υποσύστημα αυτό, μαζί με εκείνα της διεπαφής με το χρήστη, της ανάλυσης και εκτέλεσης ερωτημάτων και της αποθήκευσης και εκτέλεσης των τελεστών θα αποτελέσουν τη δημιουργία του πρώτου ερευνητικού σχεσιακού context-aware ΣΔΒΔ.

Στα πλαίσια του συστήματος που υλοποιήσαμε, υλοποιήθηκαν επιμέρους υποσυστήματα απαραίτητα για τη λειτουργία του συνολικού. Αυτά είναι τα εξής:

- Το **υποσύστημα επικοινωνίας με το σχεσιακό ΣΔΒΔ** που αποτελεί ένα επίπεδο επικοινωνίας με το ΣΔΒΔ.
- Το **υποσύστημα του καταλόγου** που είναι υπεύθυνο για το χειρισμό των meta-δεδομένων τόσο του context όσο και των morphs/polymorphs.
- Το **υποσύστημα διαχείρισης δεδομένων** που είναι υπεύθυνο για το χειρισμό των δεδομένων τόσο του context όσο και των morphs/polymorphs.
- Το **υποσύστημα αναπαράστασης ερωτημάτων** που αναπαριστά το ερώτημα σε βολική για την επεξεργασία του μορφή.
- Το **υποσύστημα μετατροπής των ερωτημάτων σε PL\SQL και εκτέλεσής τους** που μετατρέπει τελικά το ερώτημα σε πολλαπλά σχεσιακά ερωτήματα, τα εκτελεί, συγχωνεύει τα αποτελέσματα σε ένα τελικό polymorph και το επιστρέφει.

Με την ολοκλήρωση της υλοποίησης των συστημάτων αυτών είναι πλέον εφικτή η εκτέλεση ενός context-aware ερωτήματος μέσω της εκτέλεσης πολλαπλών σχεσιακών. Τα

αποτελέσματα των ερωτημάτων αυτών πάνω στους πίνακες του σχεσιακού ΣΔΒΔ αποτελούν τα morphs που ανήκουν στο polymorph που επιστρέφεται ως αποτέλεσμα.

Ως πειράματα, εκτελέστηκε εκτενής αριθμός ερωτημάτων. Αρχικά για τον έλεγχο ορθότητας του υποσυστήματος εκτελέστηκαν ερωτήματα σε μια πολυμορφική βάση δεδομένων με μικρό όγκο εγγραφών ώστε να μπορεί να γίνει η επαλήθευση των αποτελεσμάτων με αυτά που αναμέναμε θεωρητικά, ενώ στη συνέχεια εκτελέστηκαν ερωτήματα σε μια πολυμορφική βάση δεδομένων μεγέθους 1.7GB ώστε να μετρηθεί καταρχάς η απόδοση του συστήματος πάνω από πολυπληθή δεδομένα ώστε να καταλήξουμε στην πιο αποδοτική υλοποίηση (υλοποιήσεις A,B) αλλά και στην τεχνική (με-χωρίς ενδιάμεσους-προσωρινούς πίνακες) για την οποία αυτή επιτυγχάνεται. Κατά δεύτερον, στόχος μας ήταν να μετρήσουμε το overhead που εισάγεται από το υποσύστημα σε σχέση με την απευθείας εκτέλεση στο ΣΔΒΔ. Το συμπέρασμα από την εκτέλεση των τελευταίων αυτών πειραμάτων είναι ότι η τεχνική σύμφωνα με την οποία εκτελούμε τα αρχικά ερωτήματα πάνω από τους αρχικούς πίνακες δεδομένων είναι σίγουρα αποδοτικότερη εκείνης που δημιουργεί ενδιάμεσους πίνακες, όπως επίσης και ότι η υλοποίηση Α είναι αποδοτικότερη της Β όσο αυξάνονται τα δεδομένα. Σε εξαιρετικά ακραίες περιπτώσεις όπως για παράδειγμα αν έχουμε τεράστιο αριθμό από attributes για κάθε morph και λίγες tuples υπάρχει πιθανότητα να αλλάξουν τα παραπάνω δεδομένα ωστόσο κάποιο τέτοιο σενάριο καταρχάς φαντάζει μη ρεαλιστικό και είναι προφανές ότι στη γενική περίπτωση ισχύουν τα όσα προαναφέραμε.

6.2 Μελλοντικές επεκτάσεις

Μια πρώτη μελλοντική επέκταση αφορά μια τρίτη υλοποίηση (Γ) σχετικά με τον τρόπο αποθήκευσης των δεδομένων των morphs/polymorphs. Σύμφωνα με την υλοποίηση αυτή, που αποτελεί μια πιο "κάθετη" υλοποίηση, δημιουργείται στη βάση ένας πίνακας για κάθε attribute στον οποίο αποθηκεύονται τα δεδομένα για κάθε morph. Κάθε τέτοιος πίνακας έχει τρεις στήλες, από τις οποίες η πρώτη αποτελεί ένα id για την tuple, η δεύτερη αναφέρει σε ποιο morph ανήκει η συγκεκριμένη εγγραφή και η τρίτη αποτελεί την τιμή του attribute για το συγκεκριμένο morph στο οποίο ανήκει. Θα ήταν ενδιαφέρον να διενεργηθούν εκτεταμένα πειράματα για τον έλεγχο απόδοσης της υλοποίησης αυτής συγκριτικά με τις δύο που έχουν ήδη υλοποιηθεί.

Μια δεύτερη μελλοντική επέκταση, η οποία ανήκει στην κατηγορία εκείνων με στόχο τη βελτιστοποίηση του συστήματος αφορά την ενέργεια να ανέβει ο κατάλογος στη μνήμη. Όπως έχουμε παρατηρήσει από τα εκτεταμένα πειράματα του κεφαλαίου 5, σημαντικό ποσοστό του συνολικού χρόνου εκτέλεσης δαπανάται στον έλεγχο για κάθε morph σχετικά με το αν περιλαμβάνει όλα τα attributes που απαιτούνται για το polymorph στο οποίο ανήκει.

Οπότε εφόσον ανέβει ο πίνακας αυτός στη μνήμη -κάτι που είναι εφικτό λόγω του μικρού του μεγέθους- το συνολικό ποσό χρόνο που δαπανάται σε αυτό το στάδιο θα μειωθεί δραματικά.

Μια τρίτη μελλοντική επέκταση θα μπορούσε να είναι η επιλογή ενός διαφορετικού τρόπου αποθήκευσης των δεδομένων. Στην παρούσα εργασία επιλέχθηκε ως physical layer η POSTGRES ωστόσο λόγω της αρχιτεκτονικής του συστήματος κάτι τέτοιο δεν είναι δεσμευτικό και μπορεί να μεταβληθεί, είτε με κάποιο άλλο σύστημα είτε για παράδειγμα με απευθείας αποθήκευση στο δίσκο.

Τέλος, η συνένωση όλων των υποσυστημάτων που μαζί θα δημιουργήσουν το πρώτο ολοκληρωμένο context-aware ΣΔΒΔ αποτελεί μια μελλοντική επέκταση η οποία θα δώσει τη δυνατότητα στο χρήστη να χρησιμοποιήσει το σύστημα σε όλη του την έκταση, από τη δημιουργία σχέσεων με πολλαπλά σχήματα μέχρι την εκτέλεση ερωτημάτων. Εφόσον το σύστημα ολοκληρωθεί θα πρέπει να διενεργηθούν εκ νέου εκτεταμένα πειράματα ώστε να μετρηθεί συνολικά η απόδοση του συστήματος και να βγουν οριστικά συμπεράσματα για την απόδοσή του αλλά και για το overhead που εισάγει σε σχέση με τα κλασικά σχεσιακά ΣΔΒΔ.

7

Βιβλιογραφία

[DA00] Anind K. Dey and Gregory D. Abowd,
Towards a Better Understanding of Context and Context-Awareness,
Georgia Institute of Technology, Atlanta, GA

[EAM09] Hicham G. Elmongui, Walid G. Aref, Mohamed F. Mokbel,
Chameleon: Context-Awareness inside DBMSs,
Purdue University and University of Minnesota

[FAJ04] Ling Feng, Peter M.G. Apers and Willem Jonker,
Towards Context-Aware Data Management for Ambient Intelligence,
Dept. of Computer Science, University of Twente

[GPP] g++ compiler (gcc version 4.3.2)

[RS09] Ioannis N. Roussos and Timos Sellis,
A Model for Context Aware Relational Databases,
School of Electrical and Computer Engineering, National Technical University of Athens,
Institute for the Management of Information Systems (R.C. “Athena”) , 2009

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

[SPV05] Kostas Stefanidis, Evaggelia Pitoura and Panos Vassiliadis,
A Context-Aware Preference Database System,
Department of Computer Science, University of Ioannina, Greece, 2009

[SPV07] Kostas Stefanidis, Evaggelia Pitoura and Panos Vassiliadis,
Adding Context to Preferences
Department of Computer Science, University of Ioannina, Greece, 2009

[SP04] Thomas Strang and Claudia Linnhoff-Popien,
A Context Modeling Survey,
Institute for Communications and Navigation, Wessling/Oberpfaffenhofen,
Institute for Informatics, Munich, Germany

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

[Sta03] Yannis Stavrakas,
Multidimensional Semistructured Data,
Representing and Querying Context-Dependent
Multifaceted Information on the Web, PhD thesis,
Knowledge and Database Systems Laboratory,
National Technical University, June 2003

[STL94] Standard Template Library Programmer‟s Guide
http://www.sgi.com/tech/stl/, Hewlett-Packard Company 1994

[UBU09] The Ubuntu linux distribution 8.04
http://www.ubuntu.com/

[W3C07] W3C. Composite Capabilities / Preferences Profile (CC/PP),
http://www.w3.org/Mobile/CCPP