ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Υλοποίηση μιας Context Aware Σχεσιακής Βάσης Δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

ΜΑΡΚΟΥ Γ. ΔΑΣΚΑΛΑΚΗ
ΒΑΣΙΛΗ Χ. ΦΛΩΡΟΥ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάιος 2006

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Υλοποίηση μιας Context Aware Σχεσιακής Βάσης Δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

ΜΑΡΚΟΥ Γ. ΔΑΣΚΑΛΑΚΗ
ΒΑΣΙΛΗ Χ. ΦΛΩΡΟΥ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Μαΐου 2006.

.....	.....	.....
Τιμολέων Σελλής	Νεκτάριος Κοζύρης	Νίκος Παπασπύρου
Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Μάιος 2006

.....

.....

ΜΑΡΚΟΣ Γ. ΔΑΣΚΑΛΑΚΗΣ

ΒΑΣΙΛΗΣ Χ. ΦΛΩΡΟΣ

Διπλωματούχοι Ηλεκτρολόγοι Μηχανικοί και Μηχανικοί Υπολογιστών Ε.Μ.Π.

Περίληψη

Γενικά οι άνθρωποι είναι αρκετά επιτυχείς στην ανάπτυξη ιδεών ο ένας στον άλλον και στην κατάλληλη αντίδραση. Αυτό οφείλεται σε πολλούς παράγοντες όπως η αφθονία της γλώσσας που μοιράζονται, η κοινή κατανόηση για το πώς λειτουργεί ο κόσμος, και μια υπονοούμενη κατανόηση των καθημερινών καταστάσεων. Όταν οι άνθρωποι μιλούν με ανθρώπους, είναι σε θέση να χρησιμοποιήσουν τις υπονοούμενες περιστασιακές πληροφορίες, το επονομαζόμενο context, για να αυξήσουν το συνομιλητικό εύρος ζώνης. Δυστυχώς, αυτή η δυνατότητα να αναπτυχθούν οι ιδέες δεν μεταφέρεται όταν οι άνθρωποι αλληλεπιδρούν με υπολογιστές. Στον παραδοσιακό διαλογικό υπολογισμό, οι χρήστες έχουν έναν φτωχό μηχανισμό για να παρέχουν είσοδο στους υπολογιστές, συνεπώς οι υπολογιστές δεν είναι ικανοί αυτήν την περίοδο να εκμεταλλευθούν πλήρως το context του διαλόγου ανθρώπου-υπολογιστή.

Ένας γενικός ορισμός του context θα μπορούσε να είναι οποιαδήποτε πληροφορία που μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση μιας οντότητας. Μια οντότητα είναι ένα πρόσωπο, μια θέση, ή ένα αντικείμενο που θεωρείται σχετικό με την αλληλεπίδραση μεταξύ ενός χρήστη και μιας εφαρμογής, συμπεριλαμβανομένων του χρήστη και της εφαρμογής. Επομένως με τη βελτίωση της πρόσβασης του υπολογιστή στο context, αυξάνουμε την αφθονία της επικοινωνίας στην αλληλεπίδραση ανθρώπου-υπολογιστή και τον καθιστάμε πιθανό να παράγει περισσότερο χρήσιμες υπολογιστικές υπηρεσίες.

Με βάση τα παραπάνω γίνεται αντιληπτό ότι η χρήση του context είναι ιδιαίτερα σημαντική στις διαδραστικές εφαρμογές και ειδικά για τις εφαρμογές όπου το περιβάλλον του χρήστη αλλάζει γρήγορα, όπως συμβαίνει στο φορητό και πανταχού παρόντα υπολογισμό. Αυτή η εργασία εστιάζει στη σχέση του context με τα συστήματα βάσεων δεδομένων, περιγράφοντας τον τρόπο με τον οποίο είναι δυνατό να επιτευχθεί αυτή η σύζευξη, και στη συνέχεια τη λειτουργικότητα που έχει να προσφέρει.

Λέξεις Κλειδιά: context, context aware συστήματα βάσεων δεδομένων

Abstract

Generally the persons are enough successful in the growth of ideas the one in the other and in the suitable reaction. This is owed in a lot of factors as the abundance of language that they are shared, the common comprehension for how function the world, and an implied comprehension of daily situations. When the persons speak with persons, they are in place to use the implied casual information, named context, in order to increase the conversational breadth of area. Unfortunately, this possibility to develop the ideas isn't transported when the persons react with computers. In the traditional dialogic calculation, the users have a poor mechanism in order to provide entry in the computers; consequently the computers are not capable this period to exploit completely the context dialogue of person-computer.

A general definition of context could be any information that can be used in order to characterize the situation of an entity. An entity is a person, a place, or an object that is considered relative with the interaction between a user and an application, included the user and the application. Consequently with the improvement of access of computer in the context, we increase the abundance of communication in the interaction of person-computer and to produce more useful calculating services.

It becomes perceptible that the use of context is particularly important in the interact applications and specifically for the applications where the environment of the user changes fast, as it happens in the portable and everywhere present calculation. This work focuses in the relation of the context with the databases systems, describing the way with which it is possible to achieve this coupling, and then the functionalism that it has to offer.

Keywords: context, context aware database systems

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Οι πληροφορίες σήμερα προσεγγίζονται και χρησιμοποιούνται σε ένα παγκόσμιο περιβάλλον, όπου οι υπονοούμενες υποθέσεις για τα δεδομένα γίνονται όλο και λιγότερο εμφανείς. Η ερμηνεία και διαχείριση δεδομένων σύμφωνα με το context είναι ένα θέμα που δεν έχει εξερευνηθεί στην πλήρη δυνατότητά του στα διανεμημένα και ετερογενή περιβάλλοντα, όπως ο Ιστός. Η ικανότητα να προσαρμόζεται η πληροφορία σε χαρακτηριστικά του προσδοκώμενου πελάτη αποτελεί προς το παρόν κομμάτι της λογικής της εφαρμογής.

Αυτό που οραματιζόμαστε είναι το να γίνει η ικανότητα προσαρμογής κομμάτι της ίδιας της πληροφορίας και να την διαχειρίζονται με ενιαίο και ευέλικτο τρόπο τα συστήματα βάσεων δεδομένων. Η διαχείριση του context πρέπει να πραγματοποιηθεί στο επίπεδο των συστημάτων βάσεων δεδομένων με έναν ομοιόμορφο τρόπο και συνεπώς το context πρέπει να αντιμετωπιστεί ως πρώτης τάξεως πολίτης στα μοντέλα δεδομένων και τις γλώσσες διατύπωσης επερωτήσεων.

Το αντικείμενο της εργασίας αναφέρεται στη σχεδίαση και στην υλοποίηση μιας Context Aware Σχεσιακής Βάσης Δεδομένων, όπου με τον όρο context αναφερόμαστε στις διαφορετικές συνθήκες κάτω από τις οποίες αποκτά υπόσταση η πληροφορία. Ειδικότερα, οι σχεσιακές βάσεις δεδομένων (R-DBs) δεν μπορούν πλέον να θεωρηθούν ως απομονωμένες πηγές δεδομένων όπου οι χρήστες είναι εξοικειωμένοι με τις υπονοούμενες υποθέσεις απαραίτητες για την ερμηνεία των δεδομένων που ανακτούν.

Σήμερα οι R-DBs χρησιμοποιούνται ευρέως στις εφαρμογές Ιστού για τη δυναμική κατασκευή ιστοσελίδων που είναι παγκόσμια διαθέσιμες. Χρησιμοποιούνται επίσης ως back-ends στα συστήματα όπου οι context πληροφορίες παρέχονται συνεχώς από αισθητήρες. Είναι επομένως ενδιαφέρον να εξεταστεί το πώς η έννοια του context μπορεί να ενσωματωθεί σε ένα R-DBMS.

Για τον σκοπό αυτό μελετάμε μια προτεινόμενη υποδομή, το Context Relational μοντέλο (CRM), που μπορεί να ειδωθεί ως ένα εκτεταμένο σχεσιακό μοντέλο ικανό να υποστηρίξει το context και στο οποίο παρουσιάζεται ένα σύνολο βασικών λειτουργιών που επεκτείνουν τη σχεσιακή άλγεβρα. Η μελέτη αποσκοπεί στην εύρεση των κατάλληλων μεθόδων και αλγορίθμων για την μετατροπή των εννοιών και ειδικά των context λειτουργιών του CRM στις αντίστοιχές τους σχεσιακές. Το ενδιαφέρον είναι στραμμένο στο υπολογιστικό κόστος, την ποιότητα της λύσης που δίνεται, την ευκολία στην υλοποίηση και την δυνατότητα επέκτασης των αλγορίθμων για την επίλυση του προβλήματος.

1.2 Οργάνωση του τόμου

Το υπόλοιπο μέρος αυτής της εργασίας έχει την ακόλουθη δομή :

Στο δεύτερο κεφάλαιο παρουσιάζεται αναλυτικά το θεωρητικό υπόβαθρο πάνω στο οποίο στηρίζεται η διπλωματική μας. Περιγράφεται η εισαγωγή των εννοιών του context και του Context-Aware Relational Model. Επιπλέον εξηγούνται τα λεπτά σημεία και δίνονται οι ορισμοί των απαραίτητων εννοιών για την κατανόηση του μοντέλου που περιγράφεται.

Στο τρίτο κεφάλαιο περιγράφεται η διαδικασία ανάλυσης και σχεδίασης του προγράμματος. Αναφέρονται διαδικασίες όπως ο καθορισμός του προβλήματος και τα βήματα που ακολουθήθηκαν κατά την ανάλυση του προβλήματος σε μικρότερα υπό - προβλήματα. Τελειώνει με τη σχεδίαση της λύσης και παραθέτονται τα τμήματα στα οποία έχει σπάσει η σχεδίαση.

Στο τέταρτο κεφάλαιο παρουσιάζεται το τμήμα της υλοποίησης, που στηρίζεται στην ανάλυση και το σχεδιασμό που έχουν προηγηθεί. Παρουσιάζονται αναφορικά οι κυριότερες κλάσεις και περιγράφονται οι στοιχειώδεις λειτουργίες τους. Ακόμη αναλύονται τα σημαντικότερα επιμέρους θέματα υλοποίησης που αντιμετωπίστηκαν δίνοντας έτσι μια πληρέστερη εικόνα της εφαρμογής.

Στο πέμπτο κεφάλαιο αναφέρεται η μεθοδολογία του ελέγχου της εφαρμογής που πραγματοποιήθηκε κατά το στάδιο της υλοποίησης. Επίσης επιδεικνύεται η λειτουργία της εφαρμογής μέσω μιας κινητήριας υπόθεσης.

Στο έκτο κεφάλαιο που αποτελεί τον επίλογο, πραγματοποιείται σύνοψη της εργασίας και παρουσιάζονται τα συμπεράσματα και προτείνονται πιθανές μελλοντικές επεκτάσεις της διπλωματικής.

Στο έβδομο κεφάλαιο παρατίθεται η βιβλιογραφία που προσέφερε πολύτιμη βοήθεια στην οποία στηρίχθηκε η ανάπτυξη της εργασίας.

Το παράρτημα Α αποτελεί την αναλυτική περιγραφή της εφαρμογής. Είναι η τεκμηρίωση των κλάσεων της εφαρμογής, όπου μπορεί να ανατρέξει κανείς για να δει τη δομή του interface των κλάσεων.

2

Περιγραφή Θέματος

Στο κεφάλαιο αυτό παρουσιάζουμε εκτενέστερα το αντικείμενο της διπλωματικής. Η βασική έννοια που πραγματεύεται αυτή η εργασία είναι το context και συγκεκριμένα διερευνάτε το πρόβλημα της ενσωμάτωσής του στα σχεσιακά συστήματα βάσεων δεδομένων, μέσω της απόπειρας υλοποίησης μιας ενήμερης του context σχεσιακής βάσης δεδομένων. Οι έννοιες αυτές, θα επεξηγηθούν στη συνέχεια με περισσότερη λεπτομέρεια. Αρχικά γίνεται μία εισαγωγή στην έννοια του context, στη συνέχεια γίνεται αναλυτική περιγραφή του εκτεταμένου σχεσιακού μοντέλου και των λειτουργιών του, το οποίο είναι απαραίτητο για να καταστεί εφικτή η ενσωμάτωση, και τέλος γίνεται μία σύντομη περιγραφή των στόχων της εργασίας.

Η παρούσα διπλωματική στηρίχθηκε στις εργασίες που παρουσίασαν οι Σταύρακας Γιάννης και Ρούσσος Γιάννης με τίτλους Towards a Context-Aware Relational Model [RSP05] και Context Relational Model (CRM) []. Από εκεί και πέρα πηγή άντλησης χρήσιμων πληροφοριών αποτέλεσαν όλες οι αναφορές που υπάρχουν στο κεφάλαιο 7 : Βιβλιογραφία, για την ολοκλήρωση της εργασίας μας.

2.1 Σχετικές εργασίες

Η έννοια του context εισήχθη αρχικά στην πρόταση του Frege μιας αρχής του contextuality [Fre84]. Από τότε, το context έχει κληθεί να συμπεριληφθεί σε ποικίλα προβλήματα σε διαφορετικές περιοχές,. Το context έχει χρησιμοποιηθεί σε διαφορετικούς τομείς της

πληροφορικής ως εργαλείο για συλλογισμό με πεποιθήσεις απόψεων και υποβάθρων, και ως μηχανισμός αφαίρεσης για την αντιμετώπιση της πολυπλοκότητας, της ετερογένειας, και της μερικής γνώσης. Ένα επίσημο πλαίσιο για συλλογισμό σε ένα υποσύνολο μιας παγκόσμιας βάσης γνώσεων μπορεί να βρεθεί στο [GG01], ενώ παραδείγματα για το πώς μπορεί να χρησιμοποιηθεί το context για το χωρισμό μιας βάσης πληροφοριών σε εύχρηστα τεμάχια από σχετικά αντικείμενα μπορούν να βρεθούν στα [MMP95,TACS98].

Στα συστήματα πληροφοριών Ιστού το context συχνά καθορίζεται δίνοντας τιμές σε διάφορες καθορισμένες από το χρήστη μεταβλητές που καλούνται διαστάσεις ή χαρακτηριστικά. Με την ανάθεση τιμών σε τέτοιες μεταβλητές, οι χρήστες μπορούν στην πραγματικότητα να περιορίσουν τις πληροφορίες που είναι σχετικές με αυτούς διευκρινίζοντας ρητά τον τρόπο που ερμηνεύουν τα δεδομένα. Στα [Bro98,Yil97], εισάγεται η Intensional HTML για να εξετάσει το πρόβλημα της κατασκευής και της διατήρησης ιστοσελίδων που πρέπει να υπάρξουν σε πολλές εκδόσεις, παραδείγματος χάριν ανάλογα με τη γλώσσα, την πείρα του χρήστη, το επίπεδο συνδρομής, την ανάλυση της οθόνης.

Στα πολυδιάστατα ημιδομημένα δεδομένα [Sta03,SG02], ή MSSD, εφαρμόζονται παρόμοιες αρχές για να αντιπροσωπεύσουν οντότητες ημιδομημένων πληροφοριών που υποθέτουν διαφορετικές απόψεις, με ποικίλο περιεχόμενο (τιμή ή/και δομή), κάτω από διαφορετικά contexts. Αυτή η προσέγγιση θεωρεί το context ως ένα σύνολο περιορισμών, οι οποίοι συνδυάζονται μέσω των context λειτουργιών για να διευκρινίσουν τα περιβάλλοντα κάτω από τα οποία οι πληροφορίες λαμβάνουν μια σαφή έννοια. Ένα τέτοιο περιβάλλον καλείται κόσμος, ενώ οι περιορισμοί εκφράζονται με την ανάθεση τιμών στις μεταβλητές διαστάσεων. Η εργασία στο [SGDZ04] επιδεικνύει πώς το context μπορεί να χρησιμοποιηθεί για να αντιπροσωπεύσει και να επερωτήσει τον έγκυρο χρόνο των δεδομένων και τα ιστορικά των ημιδομημένων βάσεων δεδομένων.

Αυτή η αντίληψη για το context έχει χρησιμοποιηθεί επίσης στο OMSwe, μια Web-publishing πλατφόρμα που περιγράφεται στα [NP03b,NP0α]. Το OMSwe είναι βασισμένο σε ένα Object DBMS, το οποίο έχει επεκταθεί για να υποστηρίξει ένα εύκαμπτο, ανεξάρτητο του πεδίου ορισμού μοντέλο για την παράδοση πληροφοριών όπου το context διαδραματίζει έναν κεντρικό ρόλο.

Οι νέες mobile-aware εφαρμογές είναι πιο αποτελεσματικές και προσαρμοστικές στις ανάγκες πληροφοριών των χρηστών χωρίς να καταναλώνουν πάρα πολύ την προσοχή των χρηστών, εκμεταλλευόμενες τα δυναμικά χαρακτηριστικά του περιβάλλοντος, όπως η θέση του χρήστη, ο χρόνος, οι τριγύρω άνθρωποι, κ.λ.π. Μια λεπτομερής έρευνα για την εκμετάλλευση του context-aware υπολογισμού στα κινητά περιβάλλοντα παρουσιάζεται στο [CK00]. Στο [DVV03], οι συντάκτες προτείνουν μια context-aware υπηρεσία ανακάλυψης, περιγράφοντας ένα βασισμένο στο context ευρετήριο για να υποστηρίξουν την αποδοτική ανάκτηση υπηρεσιών Ιστού ενώ στο [VVV+03] παρουσιάζεται μια πλατφόρμα για τη

διανομή των context εξαρτώμενων δεδομένων και υπηρεσιών για κινητές πηγές, αποκαλούμενες MobiShare.

2.1.1 Το context ως ερμηνευτικό περιβάλλον

Ενώ οι περισσότεροι άνθρωποι καταλαβαίνουν σιωπηλά τι είναι το context, βρίσκουν δύσκολο να το διευκρινίσουν. Αυτό οφείλεται στο γεγονός πως οι περισσότεροι ερευνητές έχουν μια γενική ιδέα για το τι είναι και χρησιμοποιούν αυτή τη γενική ιδέα για να τους καθοδηγήσει στη χρήση του στις εργασίες τους. Γι' αυτό υπάρχουν διάφοροι καθορισμοί για το context, πολλοί από τους οποίους έχουν γίνει απαριθμώντας παραδείγματα ή απλά επιλέγοντας συνώνυμά του. Έτσι σε κάποιους από αυτούς το context αναφέρεται ως η θέση, οι ταυτότητες των κοντινών ανθρώπων και των αντικειμένων, και οι αλλαγές σε αυτά, ενώ σε άλλους αναφέρεται ως περιβάλλον ή κατάσταση. Το γεγονός αυτό δεν περιέχει αναγκαστικά κάτι το επιλήψιμο, αντιθέτως μαρτυρά ότι το context έχει κληθεί να συμπεριληφθεί σε ποικίλα προβλήματα σε διαφορετικά πεδία.

Οι προαναφερθείσες καταστάσεις καταδεικνύουν την ανάγκη καταρχήν για έναν τρόπο αναπαράστασης και διαχείρισης των πληροφοριακών οντοτήτων που εκδηλώνουν διαφορετικές παραλλαγές των οποίων το περιεχόμενο μπορεί να ποικίλει στη δομή και τιμή. Οι παραλλαγές μιας οντότητας πληροφοριών καλούνται εκφάνσεις της οντότητας. Σε αυτές τις καταστάσεις, το context μπορεί να χρησιμοποιηθεί ως ένας μηχανισμός άποψης που λαμβάνει υπόψη την υπονοούμενη γνώση υποβάθρου.

Σε ότι αφορά τη διαχείριση της πολυμορφίας των δεδομένων στο σημερινό παγκοσμιοποιημένο περιβάλλον είναι ιδιαίτερα σημαντικός ο ρόλος του ερμηνευτικού περιβάλλοντος. Γενικά το ερμηνευτικό περιβάλλον αντιμετωπίζεται σαν ένα σύνολο περιορισμών, οι οποίοι καλούνται διαστάσεις [dimensions], που συνδυάζονται για να ορίσουν πλαίσια κάτω από τα οποία η πληροφορία αποκτά συγκεκριμένο νόημα. Ένα τέτοιο πλαίσιο λέγεται κόσμος, ενώ οι περιορισμοί εκφράζονται με την ανάθεση τιμών στις μεταβλητές των διαστάσεων. Τα δεδομένα θα πρέπει να μπορούν να προσαρμόζονται σε διαφορετικά ερμηνευτικά περιβάλλοντα και οι πληροφοριακές οντότητες μπορούν με αυτόν τον τρόπο να εκδηλώνουν διαφορετικές εκφάνσεις, ανάλογα με το ερμηνευτικό περιβάλλον.

Στην προσέγγισή μας χρησιμοποιούμε το context για να αντιπροσωπεύει το περιβάλλον στο πλαίσιο του οποίου τα δεδομένα λαμβάνουν μια ουσία. Το σύνολο των παραμέτρων που χρησιμοποιούνται για να διευκρινίσουν το context καλούνται διαστάσεις. Ένας context specifier είναι ένα συντακτικό κατασκεύασμα που χρησιμοποιείται για να κάνει κατάλληλα κομμάτια από δεδομένα και να διευκρινίσει τα σύνολα των κόσμων (ή contexts) κάτω από τα οποία αυτά τα κομμάτια κρατούνται. Κατ' αυτό τον τρόπο, είναι δυνατό να υπάρξουν συγχρόνως παραλλαγές της ίδιας οντότητας πληροφοριών, καθεμία δεσμευμένη κάτω ένα διαφορετικό σύνολο κόσμων, ή ισοδύναμα κάτω από ένα διαφορετικό context.

2.2 Στόχος

Η φύση του Παγκόσμιου Ιστού έθεσε μια σειρά από νέα προβλήματα [BBC+98] στην ερευνητική κοινότητα των βάσεων δεδομένων. Ένα από αυτά είναι ότι, ενώ στις παραδοσιακές βάσεις δεδομένων και στα αντίστοιχα πληροφοριακά συστήματα το πλήθος των χρηστών είναι λίγο πολύ γνωστό και το υπόβαθρό τους είναι σε μεγάλο βαθμό ομογενές, οι χρήστες του Παγκόσμιου Ιστού δεν έχουν κοινό υπόβαθρο ούτε όμοιες θεωρήσεις όταν επεξεργάζονται και ερμηνεύουν δεδομένα. Αυτοί οι χρήστες μπορεί να έχουν διαφορετικές οπτικές των ίδιων πληροφοριακών οντοτήτων, κάτι που θα πρέπει να ληφθεί υπόψη από τα μοντέλα δεδομένων και τις γλώσσες επερωτήσεων για τον Παγκόσμιο Ιστό. Ένα συναφές ζήτημα είναι ότι συχνά οι παροχείς πληροφορίας πρέπει να διαχειριστούν διαφορετικές παραλλαγές των ίδιων ουσιαστικά δεδομένων, που απευθύνονται σε διαφορετικές ομάδες καταναλωτών. Παρόμοια προβλήματα εμφανίζονται κατά την ενοποίηση πληροφορίας από διάφορες πηγές [GPQ+97], όπου η ίδια εννοιολογική οντότητα μπορεί να εμφανίζει διαφορετική δομή, ή να περιέχει αλληλοσυγκρουόμενα δεδομένα.

Τα προβλήματα αυτά καταδεικνύουν την ανάγκη για έναν τρόπο αναπαράστασης και διαχείρισης πληροφοριακών οντοτήτων που εκδηλώνουν διαφορετικές παραλλαγές. Σαν ένα απλό παράδειγμα θεωρήστε ένα προϊόν (αυτοκίνητο, laptop computer, κτλ.) οι προδιαγραφές του οποίου αλλάζουν ανάλογα με την χώρα που εξάγεται. Ή μια ιστοσελίδα που πρόκειται να εμφανιστεί σε συσκευές με διαφορετικές δυνατότητες, όπως κινητά τηλέφωνα, PDAs, και προσωπικούς υπολογιστές. Ένα άλλο παράδειγμα είναι μια αναφορά που πρέπει να αναπαρασταθεί σε διάφορους βαθμούς λεπτομέρειας και σε διάφορες γλώσσες.

Μία λύση θα ήταν η δημιουργία μιας διαφορετικής αναφοράς για κάθε δυνατό συνδυασμό παραλλαγών. Βεβαίως μια τέτοια προσέγγιση δεν είναι πρακτική, αφού συνεπάγεται υπερβολική επανάληψη πληροφορίας. Ακόμη πιο σημαντικό, οι διαφορετικές παραλλαγές δεν συνδέονται μεταξύ τους σαν μέρη της ίδιας οντότητας. Αυτό το γεγονός δεν επιτρέπει την διατύπωση ενδιαφερόντων επερωτήσεων που ενέχουν εκφάνσεις πληροφοριακών οντοτήτων, όπως για παράδειγμα «δώσε μου σε μικρή λεπτομέρεια τις αναφορές των οποίων οι εκφάνσεις μεγάλης λεπτομέρειας ικανοποιούν την δεδομένη συνθήκη».

Αντιλαμβανόμαστε από τα προηγούμενα ότι πρέπει αφενός ο προμηθευτής πληροφοριών να διευκρινίζει το context κάτω από το οποίο γίνονται σχετικές οι πληροφορίες και αφετέρου οι χρήστες πληροφοριών πρέπει να μπορούν να διευκρινίζουν το τρέχον context τους όταν απαιτούν δεδομένα προκειμένου να δείξουν το τμήμα που είναι σχετικό με τη συγκεκριμένη κατάστασή τους.

Στα συστήματα πληροφοριών Ιστού το context συχνά καθορίζεται δίνοντας τιμές σε διάφορες καθορισμένες από το χρήστη μεταβλητές που καλούνται διαστάσεις ή χαρακτηριστικά. Με την ανάθεση τιμών σε τέτοιες μεταβλητές, οι χρήστες μπορούν στην πραγματικότητα να

περιορίσουν τις πληροφορίες που είναι σχετικές με αυτούς διευκρινίζοντας ρητά τον τρόπο που ερμηνεύουν τα δεδομένα. Όμως, η ικανότητα να προσαρμόζεται η πληροφορία σε χαρακτηριστικά του προσδοκώμενου πελάτη αποτελεί προς το παρόν κομμάτι της λογικής της εφαρμογής, και πρέπει να υλοποιείται εξ αρχής σε κάθε διαφορετική περίπτωση. Σε αυτή την προσέγγιση το context διαχειρίζεται από τη λογική της εφαρμογής σε χωριστό επίπεδο από τη διαχείριση των δεδομένων οδηγώντας σε μια αρχιτεκτονική άκαμπτη και σκληρή να διατηρηθεί, ενώ ad-hoc λύσεις πρέπει να αναπτυχθούν από την αρχή για κάθε νέα υπόθεση.

Σε μια άλλη προσέγγιση, το context διαχειρίζεται σε έναν μεσολαβητή, σε ένα ενδιάμεσο στρώμα μεταξύ της πηγής(ών) δεδομένων και της εφαρμογής(ών). Σε αυτήν την προσέγγιση ο μεσολαβητής περιέχει context μεταδεδομένα που "συνδέονται" με τα μέρη πληροφοριών που ανακοινώνουν στις πηγές. Οι επερωτήσεις απευθύνονται στο μεσολαβητή, ο οποίος χρησιμοποιεί τα context μεταδεδομένα για να πλοηγηθεί στα σχετικά δεδομένα.

Αυτό που οραματιζόμαστε είναι το να γίνει η ικανότητα προσαρμογής κομμάτι της ίδιας της πληροφορίας, και να την διαχειρίζονται με ενιαίο και ευέλικτο τρόπο τα συστήματα βάσεων δεδομένων. Κάτι τέτοιο θα ελαφρύνει το βάρος της ανάπτυξης ad-hoc λύσεων· αντί για αυτό, οι εφαρμογές θα θέτουν απλές επερωτήσεις, οι οποίες θα είναι σε θέση να αντιλαμβάνονται την εξάρτηση των διαφόρων εκφάνσεων της πληροφορίας.

Σκοπός αυτής της διπλωματικής, είναι να υλοποιήσουμε μια context aware σχεσιακή βάση δεδομένων, όπου το context θα αντιμετωπίζεται ως «first class citizen». Η εφαρμογή αυτή θα αποτελεί ένα εργαλείο front-end για σχεδιασμό και επερώτηση context-aware σχεσιακών βάσεων..

3

Ανάλυση και σχεδίαση

3.1 Εισαγωγή

Σκοπός αυτής της διπλωματικής όπως αναφέρθηκε, είναι να υλοποιήσουμε ένα σύστημα, μέσω της επέκτασης του σχεσιακού υπολογισμού και της άλγεβρας, που να δείχνει πως το context μπορεί να ενσωματωθεί στις σχεσιακές βάσεις δεδομένων. Πρόκειται για την απόπειρα ομοιόμορφου χειρισμού της πληροφορίας και του context που την χαρακτηρίζει. Ο στόχος είναι η αντιμετώπιση του context ως «first class citizen», όπου ένα αντικείμενο πρώτης τάξης είναι αυτό που έχει μια ταυτότητα ανεξάρτητη οποιουδήποτε άλλου αντικειμένου. Η ταυτότητα επιτρέπει στο αντικείμενο να παραμείνει όταν αλλάζουν οι ιδιότητές του, και σε άλλα αντικείμενα να απαιτήσουν σχέσεις με αυτό. Όλα αυτά συμβαίνουν με φόντο το σχεσιακό μοντέλο, γεγονός που υπαγορεύει την επέκταση του σχεσιακού λογισμού και της άλγεβρας προκειμένου να είναι λειτουργικά εφικτή αυτή η ενσωμάτωση, με τρόπο αδιαφανή προς τον χρήστη. Με το τελευταίο εννοούμε ότι δεν θα επιβαρύνεται ο χρήστης με την πολυπλοκότητα των μεθόδων και αλγορίθμων της μετατροπής των πράξεων πάνω στο CR μοντέλο, σε πράξεις πάνω στο σχεσιακό. Όλες αυτές οι έννοιες εξηγούνται με λεπτομέρεια και περιγράφονται αναλυτικά στη συνέχεια.

Η πρόκληση που αντιμετωπίζεται είναι το να γίνει η ικανότητα προσαρμογής κομμάτι της ίδιας της πληροφορίας, και να την διαχειρίζονται με ενιαίο και ευέλικτο τρόπο τα συστήματα βάσεων δεδομένων, αντιλαμβανόμενα την εξάρτηση των διαφόρων εκφάνσεων της πληροφορίας.

3.1.1 Αναλυτική περιγραφή του ερμηνευτικού περιβάλλοντος

Εναρκτήριο ερέθισμα αυτής της εργασίας υπήρξε καταρχήν η ανάγκη για έναν τρόπο αναπαράστασης και διαχείρισης των πληροφοριακών οντοτήτων που εκδηλώνουν διαφορετικές παραλλαγές των οποίων το περιεχόμενο μπορεί να ποικίλει στη δομή και τιμή. Οι παραλλαγές μιας οντότητας πληροφοριών καλούνται εκφάνσεις της οντότητας. Σε αυτές τις καταστάσεις, το context μπορεί να χρησιμοποιηθεί ως ένας μηχανισμός άποψης που λαμβάνει υπόψη την υπονοούμενη γνώση υποβάθρου.

Σε ότι αφορά τη διαχείριση της πολυμορφίας των δεδομένων στο σημερινό παγκοσμιοποιημένο περιβάλλον είναι ιδιαίτερα σημαντικός ο ρόλος του ερμηνευτικού περιβάλλοντος. Γενικά το ερμηνευτικό περιβάλλον αντιμετωπίζεται σαν ένα σύνολο περιορισμών, οι οποίοι καλούνται διαστάσεις [dimensions], που συνδυάζονται για να ορίσουν πλαίσια κάτω από τα οποία η πληροφορία αποκτά συγκεκριμένο νόημα. Ένα τέτοιο πλαίσιο λέγεται κόσμος, ενώ οι περιορισμοί εκφράζονται με την ανάθεση τιμών στις μεταβλητές των διαστάσεων. Τα δεδομένα θα πρέπει να μπορούν να προσαρμόζονται σε διαφορετικά ερμηνευτικά περιβάλλοντα και οι πληροφοριακές οντότητες μπορούν με αυτόν τον τρόπο να εκδηλώνουν διαφορετικές εκφάνσεις, ανάλογα με το ερμηνευτικό περιβάλλον.

Στην προσέγγισή μας χρησιμοποιούμε το context για να αντιπροσωπεύει το περιβάλλον στο πλαίσιο του οποίου τα δεδομένα λαμβάνουν μια ουσία. Το σύνολο των παραμέτρων που χρησιμοποιούνται για να διευκρινίσουν το context καλούνται διαστάσεις. Ένας context specifier είναι ένα συντακτικό κατασκεύασμα που χρησιμοποιείται για να κάνει κατάλληλα κομμάτια από δεδομένα και να διευκρινίσει τα σύνολα των κόσμων (ή contexts) κάτω από τα οποία αυτά τα κομμάτια κρατούνται. Κατ' αυτό τον τρόπο, είναι δυνατό να υπάρξουν συγχρόνως παραλλαγές της ίδιας οντότητας πληροφοριών, καθεμία κρατημένη κάτω ένα διαφορετικό σύνολο κόσμων, ή ισοδύναμα κάτω από ένα διαφορετικό context.

Ορισμός 1. Έστω ότι το D είναι ένα μη κενό σύνολο από ονόματα διαστάσεων και για κάθε $d \in D$, έστω ότι Vd είναι το πεδίο ορισμού του d, με $Vd \neq Ø$. Ένας κόσμος w με αναφορά στο D είναι ένα σύνολο από ζευγάρια (d,v), όπου $d \in D$ και $v \in Vd$, έτσι ώστε για κάθε $d \in D$ ακριβώς ένα (d,v) ανήκει στον w .

Προκειμένου να καταστούν κατανοητές οι παραπάνω έννοιες κρίνεται σκόπιμο σε αυτό το σημείο να περιγραφεί μέσω ενός παραδείγματος η εφαρμογή τους.

3.1.1.1 Παράδειγμα χρήσης του ερμηνευτικού περιβάλλοντος

Θεωρείστε μια ιστοσελίδα σχετική με ψηφιακές φωτογραφικές μηχανές. Οι πληροφορίες που παρέχονται για κάθε φωτογραφική μηχανή περιλαμβάνουν το εμπορικό σήμα, το μοντέλο της φωτογραφικής μηχανής, μια εικόνα, το μέγεθος σε megapixels και την τιμή. Οι πελάτες συνδέονται με αυτή την ιστοσελίδα χρησιμοποιώντας ποικίλες συσκευές που κυμαίνονται σε υπολογιστή γραφείου (PC), PDA και κινητό τηλέφωνο. Επιπλέον, μπορούν να επιλέξουν τη μέθοδο πληρωμής μεταξύ πιστωτικής κάρτας και μετρητών.

Τα δεδομένα για μια ψηφιακή φωτογραφική μηχανή που επιστρέφονται στους πελάτες εξαρτώνται από τη συσκευή πρόσβασης και τη μέθοδο πληρωμής. Δηλαδή ένας πελάτης χρησιμοποιώντας ένα PDA λαμβάνει μια εικόνα μικρότερης ανάλυσης από όταν χρησιμοποιεί έναν υπολογιστή γραφείου. Όταν χρησιμοποιείται ένα κινητό τηλέφωνο δεν υπάρχει καμιά εικόνα και παρέχονται μόνο πληροφορίες κειμένου. Επιπλέον, η τιμή μιας ψηφιακής φωτογραφικής μηχανής ποικίλλει σύμφωνα με τη μέθοδο πληρωμής. Προφανώς, η ψηφιακή φωτογραφική μηχανή είναι μια οντότητα πληροφοριών της οποίας το περιεχόμενο και η δομή σε αυτό το ιδιαίτερο παράδειγμα ποικίλλει σύμφωνα με τη συσκευή πρόσβασης και τη μέθοδο πληρωμής.

Σε όρους βάσεων δεδομένων υπάρχει μια κύρια σχέση dcamera, με τις ιδιότητες: Brand, Model, MPix, Photo, Price (βλ. πίνακα 2). Τα αιτήματα των πελατών εκφράζονται ως επερωτήσεις και τα δεδομένα που επιστρέφονται αντιστοιχούν στην αξιολόγηση αυτών των επερωτήσεων πέρα από αυτήν την σχέση. Το context της σχέσης του παραδείγματος εκφράζεται μέσω των διαστάσεων device που κυμαίνεται σε {PC, PDA, CELL} και payment που κυμαίνεται σε {Credit Card, Cash}

Το σχήμα της σχέσης dcamera και τα δεδομένα για μια ιδιαίτερη οντότητα μπορούν να διαφέρουν μεταξύ διαφορετικών κόσμων. Αυτό οφείλεται στο γεγονός ότι μια ιδιότητα μπορεί να μην υπάρχει σε μερικούς κόσμους και ότι η ίδια ιδιότητα μπορεί να έχει διαφορετικές τιμές κάτω από διαφορετικούς κόσμους. Ο πίνακας 1 παρουσιάζει όλους τους πιθανούς κόσμους, ενώ ο πίνακας 2 παρουσιάζει τους κόσμους κάτω από τους οποίους κάθε ιδιότητα καθορίζεται.

Κόσμος	**Device**	**Payment**
w1	PC	Credit Card
w2	PDA	Credit Card
w3	CELL	Credit Card
w4	PC	Cash

w5	PDA	Cash
w6	CELL	Cash

Πίνακας 3.1. Παρουσιάζονται όλοι οι πιθανοί κόσμοι του παραδείγματος

Ιδιότητες	**Κόσμοι**
Brand	ορισμένη σε όλους τους κόσμους
Model	ορισμένη σε όλους τους κόσμους
MPix	ορισμένη σε όλους τους κόσμους
Photo	ορισμένη σε όλους τους κόσμους με συσκευή πρόσβασης PC ή PDA
Price	ορισμένη σε όλους τους κόσμους, αλλά η τιμή της μπορεί να αλλάζει

Πίνακας 3.2 Παρουσιάζονται οι κόσμοι κάτω από τους οποίους ορίζεται η κάθε ιδιότητα

Στη συνέχεια παρουσιάζουμε σύνολα από κόσμους που αντιπροσωπεύονται από τους context specifiers, οι οποίοι μπορούν να θεωρηθούν ως περιορισμοί στις τιμές των διαστάσεων.

Παράδειγμα 1. Μερικοί context specifiers μπορεί να είναι οι ακόλουθοι:

(α) [device=PC]

(β) [device=PDA, payment in {credit card, cash}]

Στο παράδειγμα 1, ο context specifier (α) αντιπροσωπεύει τους κόσμους για τους οποίους η διάσταση συσκευή έχει την τιμή PC, ενώ ο (β) αντιπροσωπεύει τους κόσμους για τους οποίους η συσκευή είναι PDA και η πληρωμή είναι είτε πιστωτική κάρτα είτε μετρητά. Παρατηρήστε ότι, σύμφωνα με τον Ορισμό 1, για ένα σύνολο ζευγαριών (διάστασης, τιμής) για την αντιπροσώπευση ενός κόσμου με αναφορά σε ένα σύνολο από διαστάσεις D, πρέπει να περιέχεται ακριβώς ένα ζευγάρι για κάθε διάσταση στο D. Επομένως, εάν D = { device, payment } με Vdevice = { PC, PDA, CELL } και Vpayment = { creditcard, cash }, τότε το { (device, PC), (payment,cash) } είναι ένας από τους έξι πιθανούς κόσμους με αναφορά στο D. Δεν είναι απαραίτητο για έναν context specifier να περιέχει τιμές για κάθε διάσταση στο D. Η παράλειψη μιας διάστασης υπονοεί ότι η τιμή της μπορεί να κυμανθεί σε ολόκληρο το πεδίο ορισμού της διάστασης. Ο context specifier [ ] είναι ένα καθολικό context και αντιπροσωπεύει το σύνολο όλων των πιθανών κόσμων, ενώ ο context specifier [-] είναι ένα κενό context και αντιπροσωπεύει το κενό σύνολο κόσμων.

3.1.2 Ενσωμάτωση του ερμηνευτικού περιβάλλοντος στο επίπεδο των πηγών πληροφοριών

Αναφέρθηκε σε προηγούμενη παράγραφο ότι ο καθορισμός και ο χειρισμός του context συνηθίζεται να γίνεται σε επίπεδο εφαρμογής. Αυτό έχει μια σειρά μειονεκτημάτων: 1. πλεονασμός, καθώς η λογική και οι διαφορετικές βασικές λειτουργίες πρέπει να εφαρμοστούν ξανά σε κάθε εφαρμογή, 2. εφαρμογές που συνδέονται αυστηρά με τις συγκεκριμένες αποφάσεις σχεδίου και τη φύση κάθε εφαρμογής και 3. αυξημένη συντήρηση.

Σε μια σειρά εργασιών [SG02, Sta03] έχει υποστηριχθεί ότι η διαχείριση του context πρέπει να πραγματοποιηθεί στο επίπεδο των συστημάτων βάσεων δεδομένων με έναν ομοιόμορφο τρόπο και ότι συνεπώς το context πρέπει να αντιμετωπιστεί ως πρώτης τάξεως πολίτης στα μοντέλα δεδομένων και τις γλώσσες διατύπωσης επερωτήσεων. Ειδικότερα, οι σχεσιακές βάσεις δεδομένων (R-DBs) δεν μπορούν πλέον να θεωρηθούν ως απομονωμένες πηγές δεδομένων μέσα σε ένα τμήμα μιας επιχείρησης, όπου οι χρήστες είναι εξοικειωμένοι με τις υπονοούμενες υποθέσεις απαραίτητες για την ερμηνεία των δεδομένων που ανακτούν. Σήμερα οι R-DBs χρησιμοποιούνται ευρέως στις εφαρμογές Ιστού για τη δυναμική κατασκευή ιστοσελίδων που είναι παγκόσμια διαθέσιμες. Χρησιμοποιούνται επίσης ως back-ends στα συστήματα όπου οι context πληροφορίες παρέχονται συνεχώς από αισθητήρες. Είναι επομένως ενδιαφέρον να εξεταστεί το πώς η έννοια του context μπορεί να ενσωματωθεί σε ένα R-DBMS.

Στο [Sta03], έχει δειχθεί ότι η άμεση ενσωμάτωση του context στα συστήματα διαχείρισης βάσεων δεδομένων θα είχε τα ακόλουθα θετικά αποτελέσματα: α) διαχείριση των δεδομένων σύμφωνα με το πλαίσιο ερμηνείας, β) δυνατότητα να τεθούν cross-world επερωτήσεις που δεν έχουν αντίστοιχο στα context-unaware συστήματα, γ) εξατομίκευση κατά έναν εύκαμπτο και ομοιόμορφο τρόπο και δ) άμεση υποστήριξη για τη διαχείριση δεδομένων και των σχημάτων ιστορικών.

Προκειμένου να καταστεί εφικτή η ενσωμάτωση του context στα συστήματα διαχείρισης βάσεων δεδομένων και να αντιμετωπίζεται ως πρώτης τάξεως πολίτης στο επίπεδο των μοντέλων βάσεων δεδομένων και των γλωσσών διατύπωσης επερωτήσεων, έχει προταθεί στο [RSP05] μια υποδομή, το Context Relational μοντέλο (CR model). Δεδομένου ότι μας απασχολεί το σχεσιακό μοντέλο βάσεων δεδομένων, που έχει ένα ισχυρό επίσημο υπόβαθρο, το CRM μπορεί να ειδωθεί ως ένα εκτεταμένο σχεσιακό μοντέλο ικανό επίσης να υποστηρίξει το context. Το προτεινόμενο μοντέλο είναι σε θέση να προσαρμόσει τις οντότητες πληροφοριών που προκηρύσσουν διαφορετικές εκφάνσεις, των οποίων το περιεχόμενο μπορεί να ποικίλει στη δομή και τιμή. Κάθε έκφανση μιας τέτοιας πολύπλευρης οντότητας πληροφοριών συνδέεται με ένα context,

δηλώνοντας τους όρους κάτω από τους οποίους αυτή η όψη το κρατά. Στη συνέχεια ακολουθεί η περιγραφή του CRM και παρουσιάζεται επίσης ένα σύνολο βασικών λειτουργιών που επεκτείνουν τη σχεσιακή άλγεβρα ούτως ώστε να ληφθεί υπόψη το context.

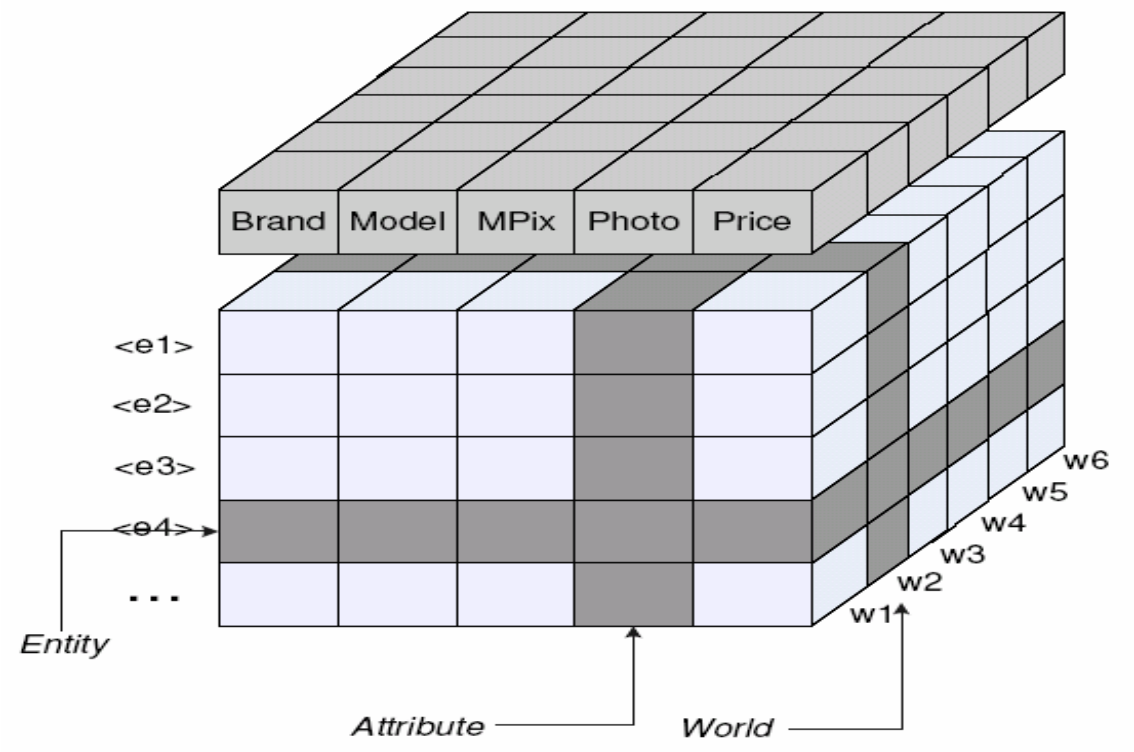
3.2 Περιγραφή του Context Relational μοντέλου (CRM)

3.2.1 Έκφανση μιας οντότητας – Facet

Η βασική έννοια του μοντέλου είναι η οντότητα πολλών εκφάνσεων. Μια οντότητα πολλών εκφάνσεων είναι μια οντότητα πληροφοριών που υποθέτει διαφορετικές εκφάνσεις όπως καθορίζονται κάτω από διαφορετικούς κόσμους. Στο υπόλοιπο του εγγράφου, όταν γράφουμε οντότητα εννοούμε μια οντότητα πολλών εκφάνσεων. Μια έκφανση fi,j είναι η παραλλαγή μιας οντότητας ei, που καθορίζεται κάτω από έναν συγκεκριμένο κόσμο wj.

3.2.2 Context σχέση

Ένα σύνολο από οντότητες ομαδοποιείται για να διαμορφώσει μια context σχέση. Για μια context-σχέση R, καθορίζονται μια σειρά από ιδιότητες σε κάθε κόσμο. Τα σύνολα των ιδιοτήτων που καθορίζονται για την R σε δύο διαφορετικούς κόσμους μπορούν (i) να είναι τα ίδια, (ii) να έχουν ένα κοινό υποσύνολο ή (iii) να μην έχουν καμιά κοινή ιδιότητα. Επιπλέον, η ίδια ιδιότητα Ai μιας μόνο οντότητας μπορεί να έχει διαφορετικές τιμές σε διαφορετικούς κόσμους. Αναφερόμαστε στην τιμή μιας ιδιότητας Ai στον κόσμο wj ως Ai{wj}.



Εικόνα 3.1. Η Context Σχέση για το παράδειγμα των ψηφιακών φωτογραφικών μηχανών

Στο σχήμα 1 παρουσιάζουμε τη context σχέση dcamera του παραδείγματος 2. Προκειμένου να παρουσιαστούν τα βασικά μέρη του μοντέλου μας, δίνουμε έμφαση στην οντότητα e4, στην ιδιότητα Photo και στον κόσμο w2.

Οι οντότητες που ανήκουν στη context σχέση dcamera έχουν τιμές για τους έξι κόσμους του πίνακα 1. Όπως δηλώνεται στον πίνακα 2, ιδιότητα Photo δεν καθορίζεται για τους κόσμους w3 και w6 (device = CELL PHONE), αποθηκεύει μια φωτογραφία υψηλής ανάλυσης για τους κόσμους w1 και w4 (device = PC) και μια χαμηλής ανάλυσης για τους κόσμους w2 και w5 (device = PDA). Επίσης η τιμή της ιδιότητας Price εξαρτάται από τη διάσταση payment. Παραδείγματος χάριν, η οντότητα e4 έχει Price { w1 } = 154 και Price { w4 } = 142.

3.2.3 Η context σχέση ως κύβος

Μια context-σχέση μπορεί να ειδωθεί ως ένας κύβος. Κάθε οντότητα της σχέσης είναι μια οριζόντια φέτα που εκτείνεται σε όλους τους κόσμους. Κάθε φέτα οντότητας έχει | a | * | w | κελιά, όπου | a | είναι ο συνολικός αριθμός των ιδιοτήτων της οντότητας και | w | είναι ο συνολικός αριθμός των κόσμων στο σύστημα. Ένα κελί { i, j } σε αυτήν την φέτα αντιπροσωπεύει την ιδιότητα Ai { wj }. Στο σχήμα 1, δίνουμε έμφαση στην οντότητα e4, η οποία είναι μια δισδιάστατη 5*6 οριζόντια φέτα, δεδομένου ότι η context-σχέση dcamera έχει πέντε ιδιότητες και καθορίζεται για έξι κόσμους. Η ιδιότητα Model είναι η δεύτερη ιδιότητα της

context-σχέσης dcamera και το { Cell Phone, Credit Card } είναι ο τρίτος κόσμος, έτσι το κύτταρο { 2,3 } αντιπροσωπεύει την ιδιότητα Model { w3 } για την οντότητα e4.

Μια ιδιότητα είναι μια δισδιάστατη |e| * |w| κάθετη φέτα, όπου |e| είναι ο συνολικός αριθμός των οντοτήτων στην εξαρτώμενη από το context σχέση. Ένα κελί {i,j} στη φέτα που αντιστοιχεί στην ιδιότητα ak αντιπροσωπεύει την τιμή της ak για την οντότητα ei στον κόσμο wj. Αναλογικά, ένας κόσμος είναι μια δισδιάστατη |e| * |a| κάθετη φέτα. Κάθε άποψη μιας οντότητας ei αντιπροσωπεύεται από τη τομή της φέτας οντότητας για την ei και της φέτας κόσμου για τον αντίστοιχο κόσμο.

Η μοντελοποίηση των οντοτήτων και των ιδιοτήτων ως δισδιάστατες φέτες μας επιτρέπει να εκθέσουμε αυτές τις σχέσεις και να ερμηνεύσουμε επερωτήσεις πέρα από διαφορετικούς κόσμους με έναν πιο σημαντικό τρόπο. Παραδείγματος χάριν, στο σχήμα 1, δίνουμε έμφαση στην ιδιότητα Photo. Η απάντηση σε μια επερώτηση που ζητά τις φωτογραφίες της οντότητας e4 αντιπροσωπεύεται από τη τομή της κάθετης φέτας ιδιότητας για την ιδιότητα Photo και της οριζόντιας φέτας οντότητας για την e4. Η απάντηση σε μια επερώτηση που ζητά όλες τις φωτογραφίες στον κόσμο w2 αντιπροσωπεύεται από τη τομή των δύο κάθετων φετών για την ιδιότητα Photo και τον κόσμο w2. Τέλος, η τομή των τριών φετών αντιστοιχεί στην τιμή της ιδιότητας Photo για την οντότητα e4 στον κόσμο w2.

3.2.4 Σύγκριση του CRM με το σχεσιακό μοντέλο

Σε αυτή την παράγραφο αναφέρονται συνοπτικά οι ποιοτικές διαφορές του CR μοντέλου σε σχέση με το σχεσιακό μοντέλο. Στο CR μοντέλο, μια φέτα κόσμου wi είναι μια κλασσική σχέση όπως στο σχεσιακό μοντέλο. Οι πλειάδες αυτής της σχέσης θα αντιστοιχούν στις εκφάνσεις των οντοτήτων για αυτόν τον κόσμο. Εντούτοις, εάν αποσυνθέσουμε μια context-σχέση σε μια σειρά από σχέσεις, η σύνδεση μεταξύ των απόψεων που αποτελούνται μια μόνο οντότητα πληροφοριών χάνεται. Αυτή η σύνδεση χρησιμοποιείται στο CR μοντέλο για να διατυπωθούν cross-world επερωτήσεις.

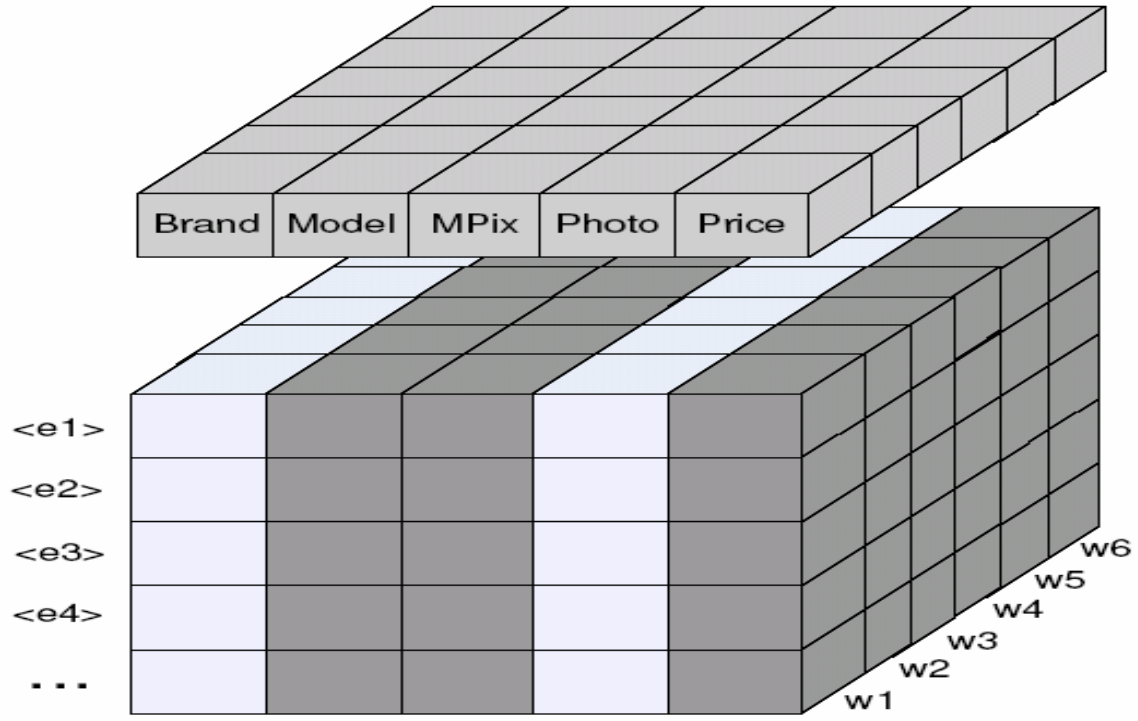
Κάποιος μπορεί να υποστηρίξει ότι αντί μιας context-σχέσης που καθορίζεται σε k κόσμους θα μπορούσαμε να έχουμε ένα σύνολο από k σχέσεις (όπως καθορίζεται στο σχεσιακό μοντέλο). Μια άλλη προσέγγιση θα ήταν να δημιουργηθεί μια μόνο denormalized σχέση με τόσες ιδιότητες όσο το άθροισμα του αριθμού των ιδιοτήτων που καθορίζονται σε κάθε κόσμο. Σε αυτήν την εργασία υποστηρίζουμε ότι καμία από αυτές τις προσεγγίσεις δεν θα βοηθούσε την επίλυση του προβλήματος που μελετάμε. Αυτό το ζήτημα αποτελεί το κομβικό σημείο της εργασίας και αναλύεται στις επόμενες παραγράφους.

3.3 Περιγραφή Λειτουργιών του CRM

Σε αυτή τη ενότητα παρουσιάζονται οι βασικές λειτουργίες (προβολή, επιλογή, καρτεσιανό γινόμενο, ένωση και λειτουργίες συνόλων) της σχεσιακής άλγεβρας που επεκτείνεται για να ενσωματώσει το context. Για να είναι διακριτές από τις κλασσικές λειτουργίες, χρησιμοποιείται ο όρος context μπροστά από αυτές, όπως για παράδειγμα context-προβολή, ενώ ισχύουν τα σχεσιακά σύμβολα, π.χ. το π για την context-προβολή.

3.3.1 Context-Προβολή (Context-Project)

Δοσμένης μιας εισαγόμενης context-σχέσης, ο στόχος είναι να παράγουμε μόνο τις ιδιότητες που διευκρινίζονται στη συνθήκη του τελεστή της context-προβολής. Το αποτέλεσμα είναι μια νέα context-σχέση που έχει μόνο τις κάθετες φέτες που αντιστοιχούν στις προβαλλόμενες ιδιότητες. Η προκύπτουσα context σχέση θα περιλάβει ακόμα τις φέτες για όλους τους πιθανούς κόσμους, εντούτοις περιέχει μόνο τις πραγματικές τιμές για το προβαλλόμενο μέρος (τονισμένα κελιά στα σχήματα αυτής της παραγράφου).

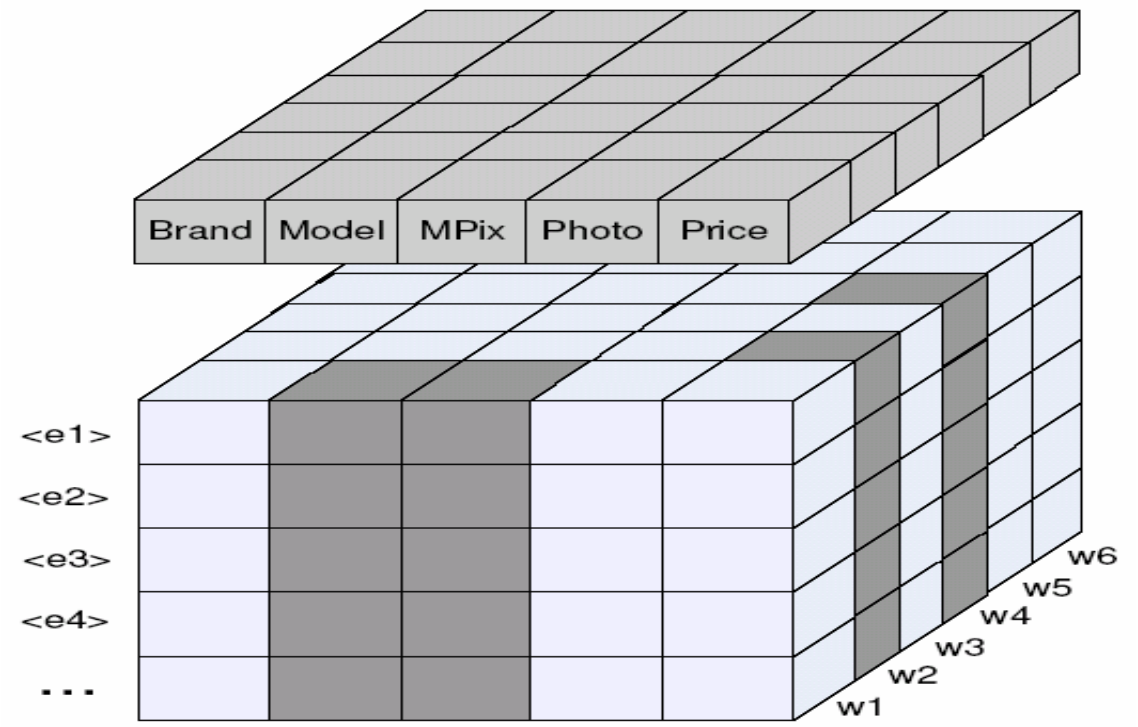


Εικόνα 3.2. Context-Προβολή για τις ιδιότητες Model, MPix και Price σε όλους τους κόσμους

Για παράδειγμα, στο σχήμα 2 βλέπουμε το αποτέλεσμα της προβολής των ιδιοτήτων Model, MPix και Price:

πModel[], MPix[], Price[]dcamera

Σημειώστε ότι ένας context specifier ακολουθεί την κάθε ιδιότητα. Η διευκρίνιση του context των προβαλλόμενων ιδιοτήτων είναι κρίσιμη όταν ενδιαφερόμαστε για τις τιμές των ιδιοτήτων μόνο για μερικούς κόσμους. Για κάθε ιδιότητα, ο context specifier δείχνει το σύνολο των κόσμων που θα προβληθεί. Εάν μια ιδιότητα Ai δεν καθορίζεται για έναν κόσμο wj στην context-σχέση εισαγωγής, τότε δεν εμφανίζεται στην προκύπτουσα context-σχέση, ακόμα κι αν ο wj εμφανίζεται στον context specifier της Ai.



Εικόνα 3.3 Επεξηγείται η ακόλουθη επερώτηση, που προβάλλει τις ιδιότητες Model και MPix για τον κόσμο w1 και την ιδιότητα Price για τους κόσμους w2 και w4.

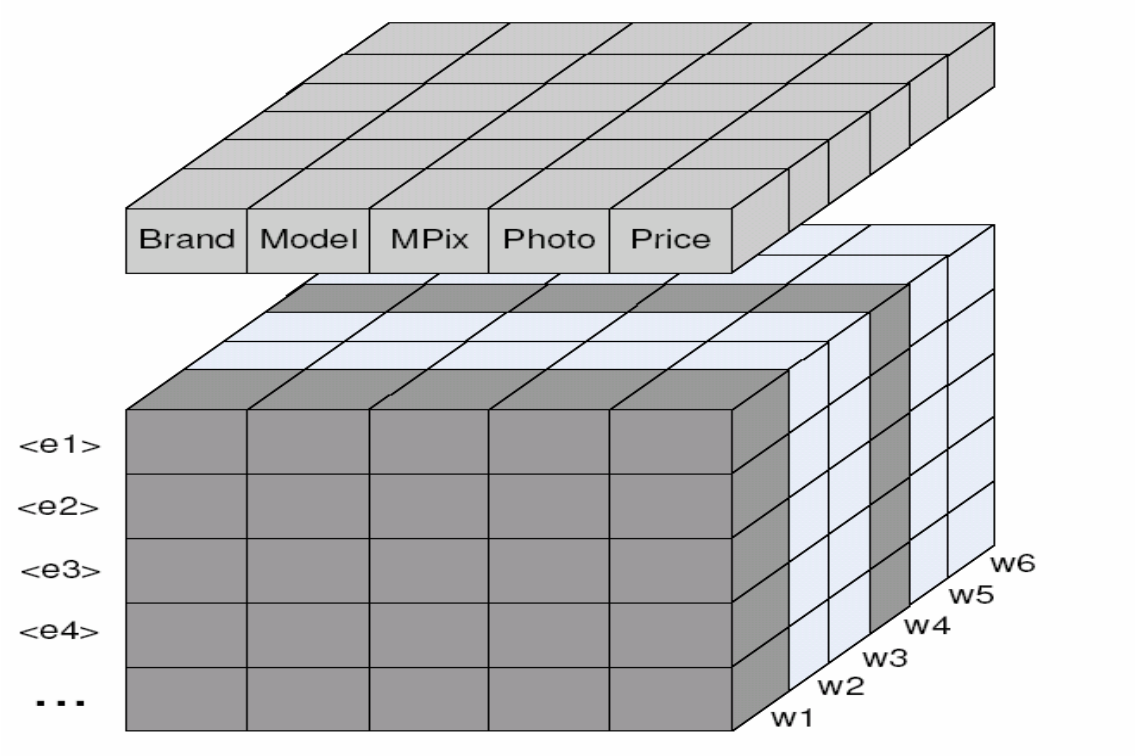
πModel[Device=PC,Payment=CreditCard], MPix[Device=PC,Payment=CreditCard], Price[Device=PDA,Payment in {CreditCard,Cash}]dcamera

3.3.2 Προβολή Κόσμου (World Project)

Αυτή η λειτουργία διατηρεί μόνο εκείνες τις απόψεις των οντοτήτων που κρατούν κάτω από διευκρινισμένους κόσμους. Για παράδειγμα, για έναν πελάτη που χρησιμοποιεί PC οι σχετικοί κόσμοι είναι οι w1 και w4. Στο σχήμα 3 παρουσιάζουμε το αποτέλεσμα της προβολής των

κόσμων w1 και w4 για την context-σχέση dcamera. Χρησιμοποιούμε το γράμμα κ για να αντιπροσωπεύσουμε τον τελεστή της προβολής-κόσμου:

κ[w1,w4]dcamera

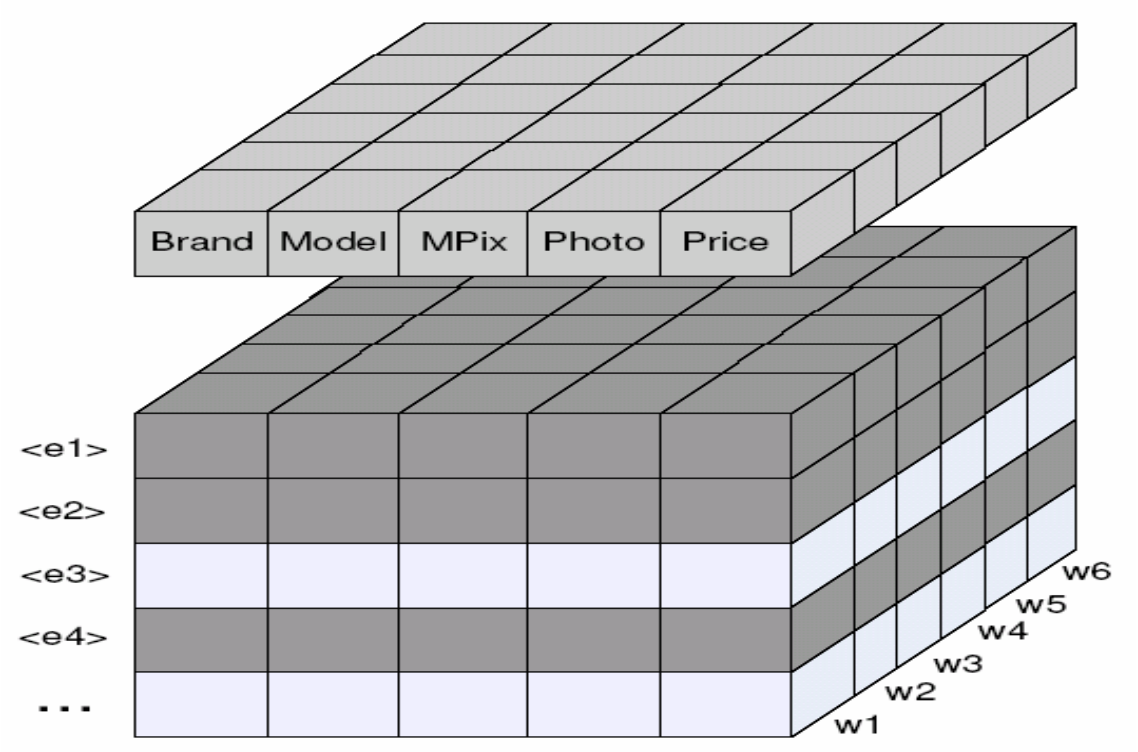


Εικόνα 3.4. Προβολή κόσμων

3.3.3 Context-Επιλογή Οντότητας (Entity Context-Select)

Η λειτουργία context-επιλογή οντότητας χρησιμοποιεί το context προκειμένου να εκφράσει τις συνθήκες που περιλαμβάνουν τις τιμές ιδιοτήτων κάτω από διαφορετικούς κόσμους. Οι ακόλουθες επερωτήσεις επεξηγούν αυτό το σημείο. Σημειώστε ότι χρησιμοποιούμε το σύμβολο σentity για να δείξουμε την context-επιλογή οντότητας. Το σχήμα 4 (α) τονίζει ψηφιακές φωτογραφικές μηχανές που δημιουργούνται από την Kodak, με περισσότερα από τρία megapixels.

$$\sigma^{entity}_{(BRAND[]='Kodak'\_AND\_M\_PIX[]>3)}dcamera$$



Εικόνα 3.5 Context-Επιλογή Οντότητας

Μια απλούστερη περίπτωση όπου τα κριτήρια επιλογής είναι περιορισμένα στις τιμές στους συγκεκριμένους κόσμους εμφανίζεται στο ακόλουθο παράδειγμα επερώτησης, η οποία επιλέγει μόνο τις οντότητες όπου η τιμή είναι λιγότερη από 250 όταν ο πελάτης πληρώνει σε μετρητά:

$$\sigma^{entity}_{(BRAND[]='Kodak'\_AND\_\Pr ice[Device=PC,Payment=Cash]<250)} dcamera$$

Σημειώστε ότι μια ιδιότητα μπορεί να εμφανιστεί περισσότερο από μία φορά στην ίδια συνθήκη με διαφορετικό context specifier. Η ακόλουθη επερώτηση είναι μια cross-world επερώτηση που συγκρίνει τις τιμές των ίδιων ιδιοτήτων σε διαφορετικούς κόσμους, ζητώντας τις φωτογραφικές μηχανές που έχουν χαμηλότερη τιμή εάν ο πελάτης πληρώνει σε μετρητά από όταν χρησιμοποιεί μια πιστωτική κάρτα:

$$\sigma^{entity}_{(\Pr ice[Device=PC,Payment=Cash]<\Pr ice[Device=PC,Payment=CreditCard])} dcamera$$

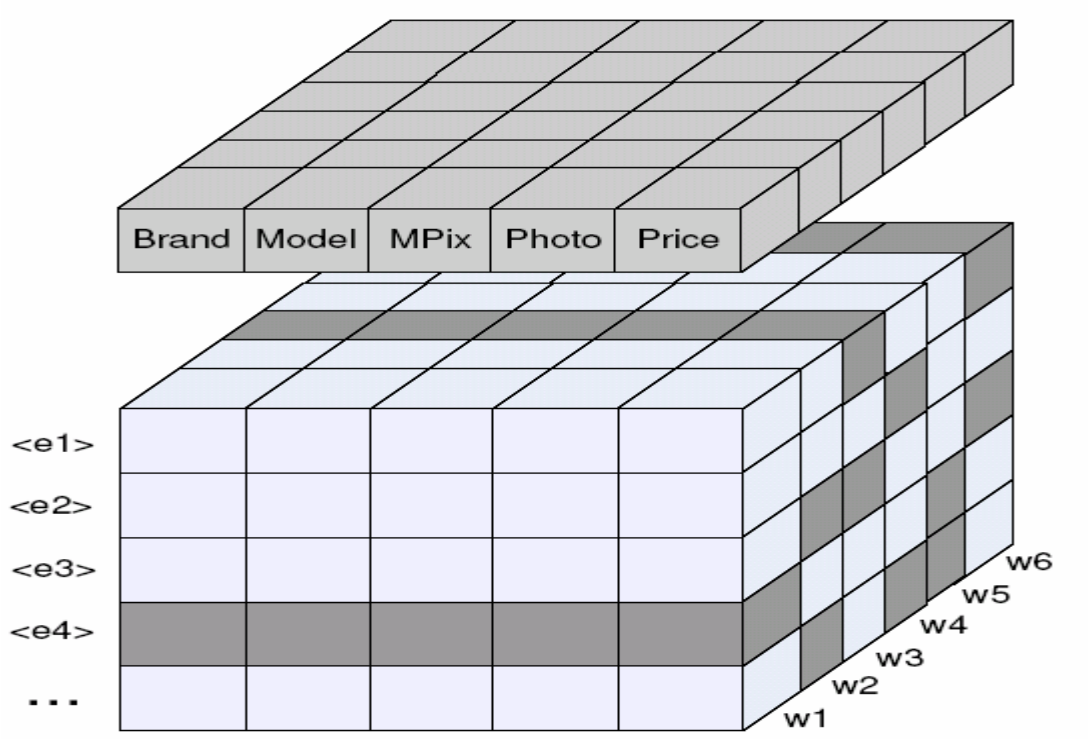
Στις προηγούμενες επερωτήσεις κάθε ιδιότητα στην συνθήκη της επιλογής αξιολογείται στο σύνολο κόσμων που υποδεικνύεται από τον αντίστοιχο context specifier. Σε περίπτωση που δεν υπάρχει κανένας context specifier για μια ιδιότητα, αξιολογούμε την συνθήκη σε αληθινή εάν υπάρχει τουλάχιστον ένας κόσμος όπου η συνθήκη κρατά. Για παράδειγμα, η ακόλουθη επερώτηση θέτει αιτήματα για τις οντότητες όπου τουλάχιστον μια άποψη έχει την Price λιγότερο από 250 Euro:

$$\sigma^{entity}_{(BRAND[]='Kodak'\_AND\_Price<250)} dcamera$$

Εάν τροποποιήσουμε την παραπάνω επερώτηση με την προσθήκη του καθολικού context specifier στην ιδιότητα Price, τότε οι επιστρεφόμενες οντότητες θα είναι εκείνες που έχουν την τιμή λιγότερο από 250 Euro σε όλους τους κόσμους.

3.3.4 Context-Επιλογή Έκφανσης (Facet Context-Select)

Οι χρήστες πρέπει να είναι σε θέση να θέσουν επερωτήσεις που περιλαμβάνουν επιλογή των απόψεων ή, με άλλα λόγια, να επιλέξουν μέρη μιας οντότητας. Αυτό μπορεί να γίνει χρησιμοποιώντας τον context-επιλογή τελεστή, που αναφέρεται ως σfacet. Ο σfacet τελεστής επιλέγει μόνο τις απόψεις μιας οντότητας που ικανοποιούν την συνθήκη, αντί ολόκληρης της οντότητας όπως στον σentity τελεστή.

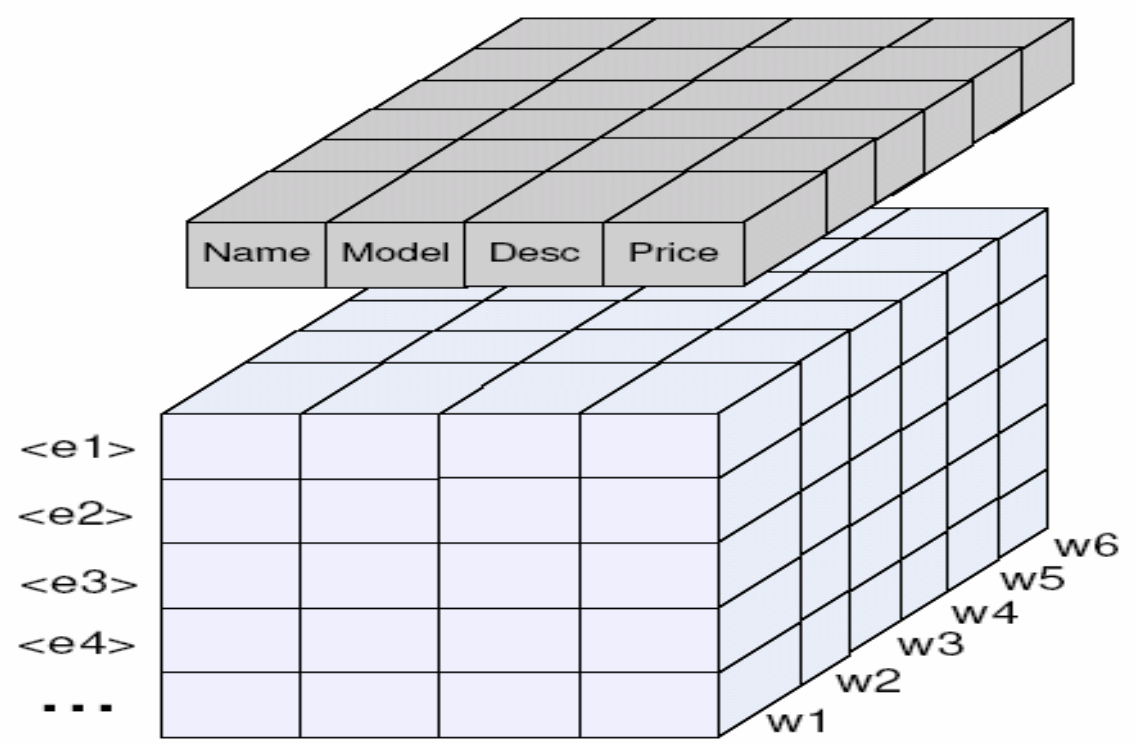


Εικόνα 3.6 Context- Επιλογή Έκφανσης

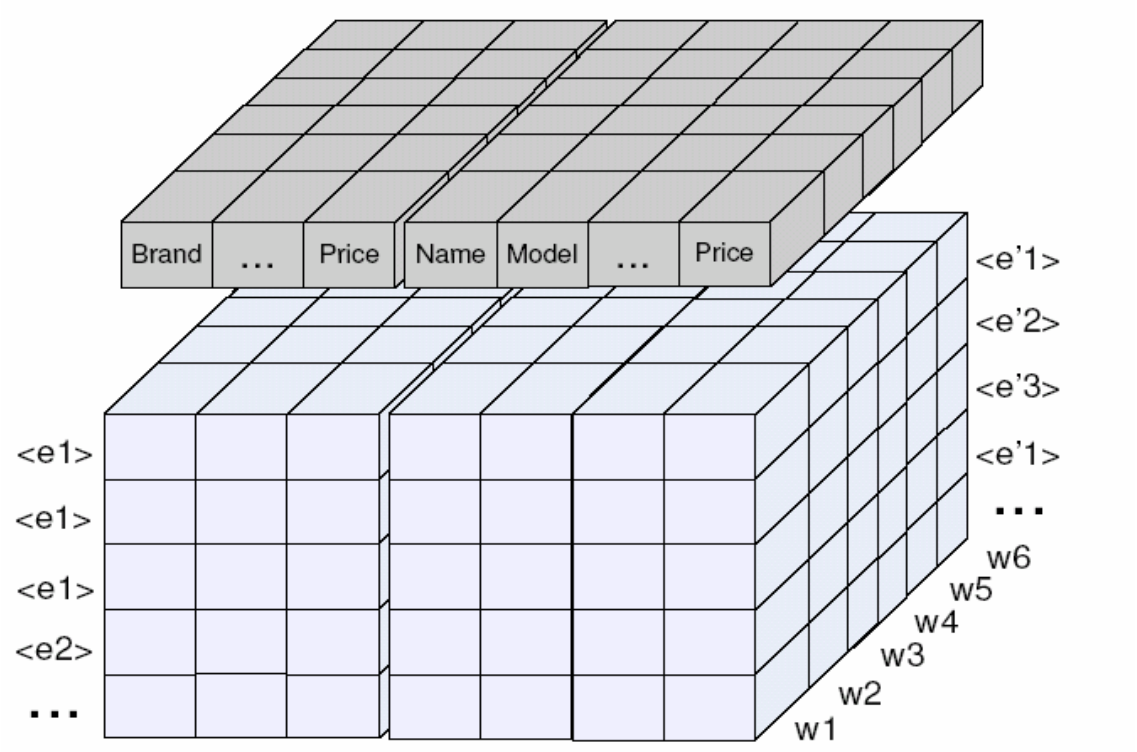
Στο σχήμα παρουσιάζουμε το αποτέλεσμα της επιλογής των απόψεων με την Price λιγότερο από 500 euro. Σημειώστε ότι η λειτουργία επιλογής άποψης είναι μια βασική λειτουργία στην άλγεβρά μας δεδομένου ότι δεν μπορεί να κατασκευαστεί από οποιοδήποτε συνδυασμό των άλλων λειτουργιών.

3.3.5 Context Καρτεσιανό Γινόμενο (Context Cartesian Product)

Με έναν τρόπο παρόμοιο με το σχεσιακό μοντέλο, το context καρτεσιανό γινόμενο δημιουργεί νέες context-σχέσεις από τις υπάρχουσες. Για τις ανάγκες του παραδείγματός μας το σχήμα 7 παρουσιάζει μια νέα context-σχέση που ονομάζεται accessories. Οι ιδιότητες που καθορίζονται για τα εξαρτήματα είναι το όνομα του εξαρτήματος, το μοντέλο της αντίστοιχης ψηφιακής φωτογραφικής μηχανής που ταιριάζει το εξάρτημα, μια περιγραφή του εξαρτήματος και τη τιμή του. Η ιδιότητα model χρησιμεύει ως ένα ξένο κλειδί στην context-σχέση dcamera.



Εικόνα 3.7 Η context-σχέση accessories



Εικόνα 3.8 Καρτεσιανό γινόμενο

Το καρτεσιανό γινόμενο των context-σχέσεων dcamera και accessories παρουσιάζεται στο σχήμα 8. Οι ιδιότητες της προκύπτουσας context-σχέσης είναι οι ιδιότητες των dcamera και accessories. Οι οντότητες της νέας σχέσης κατασκευάζονται συνδυάζοντας κάθε οντότητας της dcamera με όλες τις οντότητες της accessories.

3.3.6 Context-Ένωση (Context-Join)

Μια context-ένωση ορίζεται ως η σύνθεση του context καρτεσιανού γινομένου και των λειτουργιών context-επιλογής οντότητας, όπως στο σχεσιακό μοντέλο. Ως ένα παράδειγμα που επεξηγεί την context-ένωση, θεωρείστε μια επερώτηση που ζητά τις ψηφιακές φωτογραφικές μηχανές Kodak με φτηνό φακό 200mm (π.χ. όπου accessories.Price < 0.2 * dcamera.Price). Εκτός από την context-ένωση, που δεν προσθέτει καμία νέα περιπλοκή στην άλγεβρά μας, υπάρχει μια ανάγκη για μια πιο περιοριστική λειτουργία ένωσης, που ονομάζεται context φυσική ένωση. Αυτή η λειτουργία είναι απαραίτητη για την επιλογή οντοτήτων όπου (dcamera.Model = accessories.Model) για όλους τους κόσμους.

3.3.7 Λειτουργίες συνόλων (Set operations)

Οι λειτουργίες συνόλων όπως η ένωση, η τομή, η διαφορά και η διαίρεση καθορίζονται μόνο για τις context-σχέσεις που είναι συμβατές ένωσης. Προκειμένου για δύο context σχέσεις να είναι

συμβατές ένωσης, πρέπει να έχουν ακριβώς τις ίδιες ιδιότητες που καθορίζονται κάτω από κάθε κόσμο. Οι σημασιολογία είναι η ίδια όπως στην θεωρία συνόλων, αλλά με οντότητες ως βασικά στοιχεία. Για παράδειγμα, το αποτέλεσμα της ένωσης δύο συμβατών context-σχέσεων είναι μια νέα context-σχέση που έχει ως οντότητες τις οντότητες που ανήκουν σε οποιεσδήποτε από τις σχέσεις. Γενικά, οι λειτουργίες συνόλων καθορίζονται με έναν τρόπο παρόμοιο με το σχεσιακό μοντέλο.

3.4 Ανάλυση χειρισμού του ερμηνευτικού περιβάλλοντος

3.4.1 Συνήθεις περιπτώσεις αντιμετώπισης του context

Το σχεσιακό μοντέλο δεν μεταχειρίζεται το context και τις οντότητες ως πολίτες πρώτης κατηγορίας. Κατά συνέπεια, ο καθορισμός και ο χειρισμός των πληροφοριών που υπάρχει σε πολλαπλούς κόσμους γίνονται σε επίπεδο εφαρμογής. Αυτό έχει μια σειρά μειονεκτημάτων:

1. Πλεονασμός, καθώς η λογική και οι διαφορετικές βασικές λειτουργίες πρέπει να εφαρμοστούν ξανά σε κάθε εφαρμογή.
2. Εφαρμογές που συνδέονται αυστηρά με τις συγκεκριμένες αποφάσεις σχεδίου και τη φύση κάθε εφαρμογής.
3. Αυξημένη συντήρηση.

Υπάρχουν δύο άλλες προσεγγίσεις για την αντιμετώπιση του context. Στην πρώτη προσέγγιση το context διαχειρίζεται από τη λογική της εφαρμογής σε χωριστό επίπεδο από τη διαχείριση των δεδομένων οδηγώντας σε μια αρχιτεκτονική άκαμπτη και σκληρή να διατηρηθεί, ενώ ad-hoc λύσεις πρέπει να αναπτυχθούν από την αρχή για κάθε νέα υπόθεση.

Στη δεύτερη προσέγγιση, το context διαχειρίζεται σε έναν μεσολαβητή, σε ένα ενδιάμεσο στρώμα μεταξύ της πηγής(ών) δεδομένων και της εφαρμογής(ών). Σε αυτήν την προσέγγιση ο μεσολαβητής περιέχει context μεταδεδομένα που «συνδέονται» με τα μέρη πληροφοριών που ανακοινώνουν στις πηγές. Οι επερωτήσεις απευθύνονται στο μεσολαβητή, ο οποίος χρησιμοποιεί τα context μεταδεδομένα για να πλοηγηθεί στα σχετικά δεδομένα.

Ωστόσο όπως ήδη έχει αναφερθεί η διαχείριση του context στο επίπεδο των πηγών πληροφοριών (δηλ. συστήματα βάσεων δεδομένων) έχει μια σειρά πλεονεκτημάτων και αποτελεί πρόκληση να εξεταστεί το πώς είναι δυνατό να υλοποιηθεί.:

3.4.2 Διαφοροποιήσεις του CRM ως προς το σχεσιακό μοντέλο

Σε αυτή την παράγραφο συζητάμε τις διαφορές του CR μοντέλου σχετικά με το σχεσιακό μοντέλο. Στο CR μοντέλο, μια φέτα κόσμου wi είναι μια κλασσική σχέση όπως στο σχεσιακό μοντέλο. Οι πλειάδες αυτής της σχέσης θα αντιστοιχούν στις απόψεις των οντοτήτων για αυτόν τον κόσμο. Εντούτοις, εάν αποσυνθέσουμε μια context-σχέση σε μια σειρά από σχέσεις, η σύνδεση μεταξύ των απόψεων που αποτελούνται μια μόνο οντότητα πληροφοριών χάνεται. Αυτή η σύνδεση χρησιμοποιείται στο CR μοντέλο για να διατυπωθούν cross-world επερωτήσεις.

Κάποιος μπορεί να υποστηρίξει ότι αντί μιας context-σχέσης που καθορίζεται σε k κόσμους θα μπορούσαμε να έχουμε ένα σύνολο από k σχέσεις (όπως καθορίζεται στο σχεσιακό μοντέλο). Μια άλλη προσέγγιση θα ήταν να δημιουργηθεί μια μόνο denormalized σχέση με τόσες ιδιότητες όσο το άθροισμα του αριθμού των ιδιοτήτων που καθορίζονται σε κάθε κόσμο. Σε αυτήν την εργασία υποστηρίζουμε ότι καμία από αυτές τις προσεγγίσεις δεν θα βοηθούσε την επίλυση του προβλήματος που μελετάμε. Στις ακόλουθες παραγράφους διαμορφώνουμε περισσότερο αυτό το σημείο.

Η κύρια δυσχέρεια του σχεσιακού μοντέλου είναι ότι δεν υπάρχει κανένας τρόπος να καθοριστεί η στενή σχέση μεταξύ των ίδιων ιδιοτήτων κάτω από διαφορετικούς κόσμους. Ανεξάρτητα από το εάν μια context-σχέση καθορίζεται από k διαφορετικές σχέσεις ή από μια μόνο denormalized σχέση, το γεγονός ότι δύο ιδιότητες αντιπροσωπεύουν την ίδια ιδιότητα μιας οντότητας πληροφοριών κάτω από διαφορετικούς κόσμους δεν μπορεί να μοντελοποιηθεί.

Επίσης, στην περίπτωση της μοντελοποίησης των context-σχέσεων μέσω ενός σχήματος με k διαφορετικές σχέσεις, δεν μπορούμε να αγνοήσουμε το πρόβλημα που προκύπτει από τόσες πολλές σχέσεις. Ακόμη και οι απλές επερωτήσεις, όπως αυτή που ζητά μια συγκεκριμένη ιδιότητα Ai, αντιστοιχεί σε μια k-way ένωση όλων των σχέσεων. Σε περιβάλλοντα με πολλούς κόσμους το γενικό κόστος θα μπορούσε να ήταν τεράστιο. Το ίδιο επιχείρημα ισχύει για λειτουργίες αναπροσαρμογών. Για παράδειγμα, η εισαγωγή μιας νέας οντότητας μεταφράζεται σε k εισαγωγές σε όλες τις σχέσεις του σχήματος.

Στην περίπτωση μιας μόνο denormalized σχέσης αποφεύγεται το κόστος της ένωσης, αλλά δεν υπάρχει κανένας τρόπος να διευκρινιστούν οι ιδιότητες που ανήκουν σε έναν συγκεκριμένο κόσμο και ρωτούν intra-world ή cross-world επερωτήσεις. Επιπλέον, η διατήρηση σχέσεων με εκατοντάδες ιδιότητες προσθέτει μεγάλη πολυπλοκότητα στο στάδιο του σχεδιασμού ή της αλλαγής του σχήματος που αντιπροσωπεύει μια context aware βάση δεδομένων.

3.4.3 Το Context ως first class citizen σε μια σχεσιακή βάση δεδομένων

Οι σχεσιακές βάσεις δεδομένων είναι εξαιρετικές για την καταχώρηση δομημένων δεδομένων που προσαρμόζονται σε ένα καλά ορισμένο σχεσιακό σχήμα βάσεων δεδομένων. Δεν είναι τόσο καλές στην καταχώρηση πληροφοριών που δεν προσαρμόζονται σε ένα τέτοιο σχήμα. Δεδομένου ότι οι απαιτήσεις των χρηστών αλλάζουν αναπόφευκτα, αυτό σημαίνει δαπανηρές αναβαθμίσεις των βάσεων δεδομένων. Για να αποφύγει κανείς πάρα πολλές τέτοιες αναβαθμίσεις, σε μια μεγαλύτερη βάση δεδομένων συχνά βλέπει παροχές που επιτρέπουν στους χρήστες να προσθέσουν σχετικές πληροφορίες που δεν ταιριάζουν ωραία στο σχήμα βάσεων δεδομένων. Η επέκταση χαμηλότερου επιπέδου είναι ένα ελεύθερο πεδίο κειμένου σε ένα αρχείο που επιτρέπει στους χρήστες να προσθέσουν κειμενικά δεδομένα. Μια άλλη κοινή επέκταση είναι η εισαγωγή των καθορισμένων από το χρήστη καταλόγων τιμών.

Στην περίπτωση μας επεξεργαζόμαστε οντότητες πληροφοριών που προκηρύσσουν διαφορετικές εκφάνσεις, των οποίων το περιεχόμενο μπορεί να ποικίλει στη δομή και τιμή. Κάθε έκφανση μιας τέτοιας πολύπλευρης οντότητας πληροφοριών συνδέεται με ένα context, δηλώνοντας τους όρους κάτω από τους οποίους αυτή η όψη το κρατά. Προφανώς αυτό απέχει πολύ από το σχεσιακό σχήμα και αποτελεί ζητούμενο το πώς μπορεί να επεκταθεί το σχεσιακό μοντέλο για να συμφωνήσει με το context, να το θεωρεί δηλαδή ως first class citizen.

Γενικά στη διαμόρφωση βάσεων δεδομένων, ένα αντικείμενο πρώτης τάξης είναι αυτό που έχει μια ταυτότητα ανεξάρτητη οποιουδήποτε άλλου αντικειμένου. Η ταυτότητα επιτρέπει στο αντικείμενο να παραμείνει όταν αλλάζουν οι ιδιότητές του, και σε άλλα αντικείμενα να απαιτήσουν σχέσεις με το αντικείμενο.

Κατά γενικό κανόνα, τα αντικείμενα πρώτης τάξης αντιπροσωπεύουν πράγματα παρά σχέσεις. Για παράδειγμα, οι αντιπροσωπεύσεις ενός ανθρώπου και μιας επιχείρησης από τη βάση δεδομένων είναι το καθένα αντικείμενο πρώτης τάξης. Εντούτοις δεν είναι, το γεγονός ότι το πρόσωπο είναι ένας υπάλληλος εκείνης της επιχείρησης, ούτε τα δεδομένα για εκείνη την σχέση, π.χ. πληροφορίες για το μισθό που η επιχείρηση πληρώνει στον υπάλληλό της.

Τυπικά, μια σχεσιακή βάση δεδομένων θα περιλάβει διάφορους πίνακες, κάθε ένας από τους οποίους περιέχει γραμμές που αντιπροσωπεύουν αντικείμενα πρώτης τάξης ενός δεδομένου τύπου (π.χ. ένας πίνακας από ανθρώπους, ένας πίνακας των επιχειρήσεων). Περιέχει επίσης άλλους πίνακες που αντιπροσωπεύουν τις σχέσεις μεταξύ αυτών των αντικειμένων πρώτης τάξης. Σε έναν πίνακα που αντιπροσωπεύει τα αντικείμενα πρώτης τάξης, μια στήλη του πίνακα περιέχει χαρακτηριστικά έναν διαφορετικό ακέραιο αριθμό που ανατίθεται σε κάθε σειρά (αποτελεσματικά, σε κάθε αντικείμενο) ως μοναδικό προσδιοριστικό: δηλαδή, μοναδικό για τα

αντικείμενα αυτού του τύπου τα αντικείμενα των διαφορετικών τύπων, που αντιπροσωπεύονται στους διαφορετικούς πίνακες, μπορούν συμπτωματικά να έχουν το ίδιο προσδιοριστικό, αλλά η σύμπτωση είναι χωρίς νόημα.

Σε μια σχεσιακή βάση δεδομένων, ένας πίνακας που αντιπροσωπεύει μια σχέση μεταξύ δύο ή περισσότερων αντικειμένων πρώτης τάξης (ή δεδομένα για εκείνη την σχέση) συνήθως δεν θα έχει ειδικά προσδιοριστικά για τις σειρές του. Άντ' αυτού αυτές οι σειρές θα προσδιοριστούν από μια διατεταγμένη εγγραφή που αποτελείται από τα μοναδικά προσδιοριστικά των αντικειμένων πρώτης τάξης που περιλαμβάνονται στη σχέση.

Η κύρια δυσχέρεια του σχεσιακού μοντέλου είναι ότι δεν υπάρχει κανένας τρόπος να καθοριστεί η στενή σχέση μεταξύ των ίδιων ιδιοτήτων κάτω από διαφορετικούς κόσμους. Ανεξάρτητα από το εάν μια context-σχέση καθορίζεται από k διαφορετικές σχέσεις ή από μια μόνο denormalized σχέση, το γεγονός ότι δύο ιδιότητες αντιπροσωπεύουν την ίδια ιδιότητα μιας οντότητας πληροφοριών κάτω από διαφορετικούς κόσμους δεν μπορεί να μοντελοποιηθεί.

Επίσης, στην περίπτωση της μοντελοποίησης των context-σχέσεων μέσω ενός σχήματος με k διαφορετικές σχέσεις, δεν μπορούμε να αγνοήσουμε το πρόβλημα που προκύπτει από τόσες πολλές σχέσεις. Ακόμη και οι απλές επερωτήσεις, όπως αυτή που ζητά μια συγκεκριμένη ιδιότητα Ai, αντιστοιχεί σε μια k-way ένωση όλων των σχέσεων. Σε περιβάλλοντα με πολλούς κόσμους το γενικό κόστος θα μπορούσε να ήταν τεράστιο. Το ίδιο επιχείρημα ισχύει για λειτουργίες αναπροσαρμογών. Για παράδειγμα, η εισαγωγή μιας νέας οντότητας μεταφράζεται σε k εισαγωγές σε όλες τις σχέσεις του σχήματος.

Στην περίπτωση μιας μόνο denormalized σχέσης αποφεύγεται το κόστος της ένωσης, αλλά δεν υπάρχει κανένας τρόπος να διευκρινιστούν οι ιδιότητες που ανήκουν σε έναν συγκεκριμένο κόσμο και ρωτούν intra-world ή cross-world επερωτήσεις. Επιπλέον, η διατήρηση σχέσεων με εκατοντάδες ιδιότητες προσθέτει μεγάλη πολυπλοκότητα στο στάδιο του σχεδιασμού ή της αλλαγής του σχήματος που αντιπροσωπεύει μια context aware βάση δεδομένων.

Το ενδιαφέρον μέρος αυτής της προσέγγισης είναι ο ομοιόμορφος χειρισμός της πληροφορίας και του context που την χαρακτηρίζει και ότι το context αντιμετωπίζεται ως πρώτης τάξεως πολίτης από τα ΣΔΒΔ. Πρωτίστως ο στόχος είναι να επιτραπεί η διαχείριση του context να λαμβάνει χώρα στο επίπεδο των συστημάτων βάσεων δεδομένων. Το προτεινόμενο πλαίσιο πρέπει επίσης να μοντελοποιήσει ετερογενή περιβάλλοντα όπου οι χρήστες μπορούν να λάβουν διαφορετικές απαντήσεις για την ίδια επερώτηση ανάλογα με το context. Αυτό είναι μια συνέπεια της οπτικής για το μοντέλο, όπου μια ιδιότητα μπορεί να μην υπάρξει σε μερικούς κόσμους ή η ίδια ιδιότητα μπορεί να έχει διαφορετικές τιμές κάτω από διαφορετικούς κόσμους. Το προκλητικό μέρος αυτής της εργασίας αποτελέστηκε στο πώς να επεκτείνει τις κλασσικές

λειτουργίες της σχεσιακής άλγεβρας για να περιλάβει το context και να επιτρέψει στους χρήστες να θέσουν cross world επερωτήσεις.

Το CRM μπορεί να ειδωθεί ως ένα εκτεταμένο σχεσιακό μοντέλο ικανό να υποστηρίξει το context. Το προτεινόμενο μοντέλο είναι σε θέση να προσαρμόσει τις οντότητες πληροφοριών που προκηρύσσουν διαφορετικές όψεις, των οποίων το περιεχόμενο μπορεί να ποικίλει στη δομή και τιμή. Κάθε όψη μιας τέτοιας πολύπλευρης οντότητας πληροφοριών συνδέεται με ένα context, δηλώνοντας τους όρους κάτω από τους οποίους αυτή η όψη το κρατά.

3.4.4 Αρχικές προσεγγίσεις

3.4.4.1 Αντιστοίχηση κάθε γνωρίσματος στο CRM στο σύνολο των συνδυασμών

Ως μια πρώτη απόπειρα αναπαράστασης εκλαμβάνεται εκείνη όπου κάθε γνώρισμα στο CRM αντιστοιχίζεται στο σύνολο των συνδυασμών του γνωρίσματος σε κάθε κόσμο στον οποίο υπάρχει, ενώ ταυτόχρονα υπάρχει και ένας ειδικός πίνακας μεταπληροφορίας που λειτουργεί βοηθητικά, αποθηκεύοντας πληροφορίες, όπως ποιοι κόσμοι έχουν οριστεί, ποια attributes υπάρχουν κάτω από αυτούς τους κόσμους και ποια όχι κλπ.

Παράδειγμα : Έστω πως οι δυνατοί κόσμοι προκύπτουν από το συνδυασμό των τιμών των διαστάσεων currency και language. Οι τιμές του currency είναι {euro, dollar} και του language {Greek, English}. Τότε οι δυνατοί κόσμοι είναι :

w1 : (euro, Greek)

w2 : (euro, English)

w3 : (dollar, Greek)

w4 : (dollar, English)

Οπότε, αν έχουμε ένα γνώρισμα price, αυτό θα αντιστοιχισθεί στα γνωρίσματα :

price_w1, price_w2, price_w3, price_w4.

Στην περίπτωση που για κάποιο κόσμο αυτό το γνώρισμα δεν υπάρχει, τότε παραλείπουμε αυτό τον κόσμο στη δημιουργία αντιστοίχισης.

Τέλος, σαν αναγνωριστικό (πρωτεύον κλειδί) της multifacet οντότητας προσθέτουμε ένα πεδίο id που διαχειρίζεται εσωτερικά το σύστημα.

Με αυτή τη μέθοδο, μια οντότητα product η οποία έχει τα attributes : name, description, price θα αναπαρασταθεί ώς

id	name_w1	name_w2	name_w3	name_w4	description_w1	description_w2	description_w3

description_w4	price_w1	price_w2	price_w3	price_w4

Αυτή η μέθοδος παρέχει τη μεγαλύτερη δυνατή ευελιξία στην περίπτωση των cross-world queries. Με βάση αυτή την αναπαράσταση, η κάθε query βασισμένη στο context-aware μοντέλο, θα μετασχηματίζεται έτσι ώστε να δίνονται τα αναμενόμενα αποτελέσματα, χωρίς ο χρήστης να αντιλαμβάνεται σε κανένα σημείο τον τρόπο με τον οποίο υλοποιείται φυσικά η σχεδίαση.

3.4.4.2 Context σχέση

Ας υποθέσουμε μια οντότητα e, η οποία ορίζεται κάτω από ένα context C.

• Οι διάφοροι κόσμοι του C καθορίζονται από τις διαστάσεις D1,...,Dn

• Η οντότητα e χαρακτηρίζεται από τα γνωρίσματα A1, A2, ..,Ak

• Η οντότητα e έχει διαφορετικές τιμές ή και γνωρίσματα σε διαφορετικούς κόσμους του C.

Αναπαριστούμε αυτή την οντότητα χρησιμοποιώντας Context-Relation Rc

Για να μοντελοποιήσουμε αυτή την πολλαπλών όψεων οντότητα χρησιμοποιώντας το σχεσιακό μοντέλο χρησιμοποιούμε μια μέθοδο παρόμοια με (to temporal model's vertical decomposition). Οι κόσμοι για κάθε context C ορίζονται για ένα σχήμα βάσης χρησιμοποώντας έναν ειδικό πίνακα Ctx. Για παράδειγμα (υποθέτοντας ότι οι διαστάσεις D1,...,Dn είναι ακέραιοι) :

word	***D1***	***D2***	...	***Dn***
w1	1	1		1
w2	1	1		2
w3	1	2		1
...	...	...		...

Τα γνωρίσματα που ορίζονται για κάθε context-σχέση για κάθε κόσμο αποθηκεύονται σε έναν ειδικό πίνακα Attr_Ctx:

world	***Relation***	***Attribute***
w1	R	A1
w1	R	A4
w2	R	A2
...	...	...

Για παράδειγμα, το σχήμα μιας context-σχέσης R για τον κόσμο w1 είναι :{R#,A1,A4}. Διαφορετικές πολλαπλών όψεων οντότητες μιας context-σχέσης αναγνωρίζονται χρησιμοποιώντας ένα γνώρισμα – κλειδί (πχ R#).

Η σχέση Attr_Ctx χρησιμοποιείται από το σύστημα για να παρακολουθεί τη σχέση R σε διαφορετικούς κόσμους. Επίσης, στην περίπτωση εισαγωγών, η Attr_Ctx χρησιμοποιείται για να εξασφαλίσει ότι δεν θα εισαχθούν δεδομένα σε ένα γνώρισμα Ai μιας context-σχέσης Rc σε έναν κόσμο όπου το Ai δεν ορίζεται. Για παράδειγμα, μια ενημέρωση για το γνώρισμα Ai της σχέσης Rc στον κόσμο wj επιτρέπεται μόνο αν το παρακάτω ερώτημα επιστρέψει μονάδα.

SELECT Count(*)
FROM Attr_Ctx
WHERE word = wj AND Relation = Rc AND Attribute = Ai

Για κάθε context-σχέση Rc ορίζονται :

Ο πίνακας Rc_Ctx : η πολυπρόσωπη οντότητα ei ορίζεται στον κόσμο wj αν υπάρχει το tuple {Ri,wj} στη σχέση Rc_Ctx.

Η σχέση αυτή χρειάζεται για να μπορούμε να έχουμε οντότητες οι οποίες περιέχουν δεδομένα μόνο για κάποιους κόσμους (πχ για ένα προϊόν το οποίο εισάγεται μόνο σε ΗΠΑ και Καναδά).. Για παράδειγμα, η οντότητα e1 ορίζεται μόνο για τους κόσμους {w1,w2,w4).

R#	**word**
R1	w1
R1	w2
R1	w4
R2	w1
...	...

Για κάθε attribute Ai της context σχέσης Rc, ορίζεται μια σχέση Rc_Ai_Ctx που έχει για μέλη της όλες τις Rc_Ai_Ctx σχέσεις της.

R#	**word**	**Attribute**	**value**
R1	w1	a1	1
R1	w2	a1	5
R1	w4	a1	8

... ...

...

Με αυτή τη μέθοδο για κάθε context σχέση Rc ορίζεται το σύνολο R_A_C των ανωτέρω πινάκων. Κάποιες ιδέες για βελτίωση είναι αντί να υπάρχει σρην Rc_Ai_Ctx σχέση η στήλη Ai, για να απαλοιφθεί η αντιστοίχιση Attribute σε αριθμούς, θα υπάρχουν δυο στήλες, η attribute (με πεδία το όνομα του κάθε attribute που ορίζεται στον πίνακα) και η value, με την τιμή αυτού του attribute στο συγκεκριμένο κόσμο. Επίσης στις στήλες R# των δυο τελευταίων πινάκων δεν χρειάζεται το πρόθεμα R, αρκεί να εισαχθεί ένας integer σαν κωδικός, κάτι που γενικά θα απλοποιήσει τη διαδικασία.

3.4.5 Αναπαράσταση του Context Aware Σχεσιακού Μοντέλου

Σε αυτή την ενότητα παρουσιάζεται η σχεδίαση για την αναπαράσταση Context Aware Σχεσιακού Μοντέλου. Σε αυτή την αναπαράσταση θεωρούνται οι εξής συμβάσεις :
Όλες οι διαστάσεις της βάσης δεδομένων που σχεδιάζεται, καθώς και οι τιμές τους ορίζονται μια φορά, κατά τη δημιουργία της βάσης.
Οι κόσμοι της βάσης, ορίζονται αυτόματα σαν συνδυασμοί των διαστάσεων που έχουν οριστεί.
Καθορίζεται για κάθε attribute σε ποιους κόσμους ορίζεται, κατά τη δημιουργία των tables,.
Στον πίνακα **cameras** αποθηκεύονται οι δυνατές τιμές κάθε διάστασης.

cameras	
dimension	***value***

Ο πίνακας universe περιέχει όλους τους δυνατούς συνδυασμούς ανάμεσα στις τιμές των διαστάσεων που έχουν οριστεί και αποδίδει σε κάθε συνδυασμό ένα αναγνωριστικό (ακέραιος αριθμός).
Συνολικά παράγονται dimension_one.size * dimension_two.size * ... κόσμοι

universe		
world_key	dim_one_value	dim_two_value

Ο πίνακας schema αντιστοιχεί σε όλα τα attributes όλων των tables, τους κόσμους στους οποίους αυτό το attribute υπάρχει. Στο πρόγραμμα, η default επιλογή κατά τον ορισμό των attribute είναι αυτό να ορίζεται για κάθε κόσμο.

schema		
table	world	attribute

Στον πίνακα data αποθηκεύονται τελικά τα δεδομένα της βάσης.

Το πεδίο id, αποτελεί αναγνωριστικό μιας οντότητας.

Τα attributes της ίδιας οντότητας μπορούν να παίρνουν ίδιες ή διαφορετικές τιμές ανάλογα με τον κόσμο στον οποίο βρισκόμαστε. Αν στο πεδίο value υπάρχει η τιμή “null”, τότε θεωρείται πως έχει δοθεί από το χρήστη.

Αν ένα attribute δεν υπάρχει στον πίνακα για κάποιο κόσμο, (δηλαδή δεν υπάρχει γραμμή που να αντιστοιχεί στον συνδυασμό κόσμου-attribute), τότε θεωρείται πως δεν ορίζεται αυτό το attribute σε αυτό το κόσμο (not exists).

data					
row	table	world	attribute	id	value

Μετά από αυτή τη συνοπτική περιγραφή του μοντέλου ακολουθεί η ανάλυση των λειτουργιών που υποστηρίζει.

3.5 Ανάλυση των λειτουργιών του Context Aware Σχεσιακού Μοντέλου

3.5.1 Δομικές λειτουργίες

3.5.1.1 Ορισμός μιας Context Relational βάσης Δεδομένων

Για να δημιουργηθεί μια νέα βάση δεδομένων σύμφωνα με το μοντέλο, η διαδικασία είναι η εξής :

Δημιουργείται ένα σύνολο από Διαστάσεις. Οι διαστάσεις αντιπροσωπεύονται από στιγμιότυπα της κλάσης model.Dimension. Ο ορισμός τους περιλαμβάνει ένα όνομα για την κάθε διάσταση καθώς και ένα σύνολο τιμών.

Παράγεται το σύνολο των κόσμων που ορίζεται με βάση αυτές τις διαστάσεις και τις τιμές τους, παίρνοντας όλους τους δυνατούς συνδυασμούς που προκύπτουν αν επιλεχθεί κάθε φορά διαφορετική τιμή για κάθε διάσταση. Συνολικα προκύπτουν (n1)(n2)...(nk) κόσμοι, όπου k το πλήθος των διαστάσεων και ni το πλήθος των τιμών κάθε διάστασης.

Δημιουργείται ένα στιγμιότυπο της κλάσης model.Database. Ο ορισμός της περιλαμβάνει ένα όνομα για τη νέα βάση, το σύνολο των διαστάσεων και το σύνολο των κόσμων. Με αυτά τα δεδομένα, δημιουργείται μια νέα βάση δεδομένων στην postgreSQL.

Στη συνέχεια δημιουργείται ο πίνακας ctx και στον οποίο τελικά εισάγονται οι κόσμοι που έχουν αποθηκευτεί στην model.Database.

3.5.1.2 Ορισμός μιας νέας context Relational σχέσης

Δημιουργείται ένα σύνολο από Γνωρίσματα. Τα γνωρίσματα αντιπροσωπεύονται από στιγμιότυπα της κλάσης model.Attribute. Ο ορισμός κάθε γνωρίσματος περιλαμβάνει ένα όνομα που θα το χαρακτηρίζει και την επιλογή ενός συνόλου κόσμων στους οποίους το γνώρισμα αυτό θα ορίζεται.

Δημιουργείται ένα στιγμιότυπο της κλάσης model.Cube. Ο ορισμός της περιλαμβάνει ένα όνομα για το νέο κύβο καθώς και ένα μη-μηδενικό σύνολο από γνωρίσματα.

Δημιουργείται στη βάση δεδομένων τον πίνακα Attr_Ctx, εκτός και αν ήδη υπάρχει. Στον πίνακα αυτό προστίθενται για κάθε c-r σχέση , τα γνωρίσματα A που τον απαρτίζουν και οι κόσμοι όπου αυτά ορίζονται, με συνδυασμούς του τύπου (relation, attribute, world). Σε αυτό τον πίνακα γίνεται αναφορά για να διαπιστωθεί σε ποιους κόσμους ορίζεται κάθε attribute.

Δημιουργείται ο πίνακας Rc_Ctx, όπου Rc το όνομα της c-r σχέσης, στον οποίο αποθηκεύονται τα κλειδιά της κάθε οντότητας που υπάρχει στην c-r σχέση και οι κόσμοι στους οποίους ορίζεται. Έτσι είναι δυνατός ο έλεγχος για το σε ποιους κόσμους ορίζεται μια συγκεκριμένη entity.

Δημιουργείται τέλος ο πίνακας Rc_A_Ctx, στον οποίο αποθηκεύονται συνδυασμοί (entity, world, attribute, value) όπου γίνεται αναφορά προκειμένου να αποθηκευτεί η τιμή που λαμβάνει κάθε γνώρισμα μιας οντότητας σε ένα συγκεκριμένο κόσμο.

3.5.1.3 Επεξεργασία μιας context Relational σχέσης

Η λειτουργία της επεξεργασία μιας c-r σχέσης δίνει τη δυνατότητα να επεξεργαστεί μια υπάρχουσα σχέση, αλλάζοντας το σύνολο των γνωρισμάτων της και τα χαρακτηριστικά του κάθε γνωρίσματος δηλαδή το πεδίο ορισμού των κόσμων του. Για να πραγματοποιηθούν οι αλλαγές, η κλάση control.CubeHandler επεξεργάζεται τον πίνακα Attr_Ctx. Στον πίνακα αυτό προσθέτονται τα γνωρίσματα και οι κόσμοι όπου αυτά ορίζονται, με συνδυασμούς του τύπου (relation, attribute, world).

3.5.1.4 Διαγραφή μιας context Relational σχέσης

Η λειτουργία της διαγραφής μιας c-r σχέσης δίνει τη δυνατότητα να πραγματοποιηθεί διαγραφή μιας υπάρχουσας σχέση. Η διαδικασία που ακολουθείται για να γίνει αυτό είναι η εξής.

Διαγράφονται από τον πίνακα Attr_Ctx όλοι οι συνδυασμοί (relation, attribute, world) που αντιστοιχούν σε αυτό τον πίνακα, καθώς και οι πίνακες Rc_Ctx και Rc_A_Ctx που αντιστοιχούν σε αυτή τη σχέση.

3.5.1.5 Προσθήκη νέας οντότητας

Για να εισαχθεί μια νέα οντότητα σε μια c-r σχέση, η διαδικασία που ακολουθείται είναι η εξής:

Κατ'αρχάς δημιουργείται αυτόματα ένα αναγνωριστικό για τη νέα οντότητα, από την κλάση control.CubeHandler και προβάλλεται στην κλάση view.UseCube. Στη συνέχεια ορίζονται οι κόσμοι στους οποίους ορίζεται η νέα οντότητα και τέλος ορίζονται τιμές για τα γνωρίσματα των κόσμων στους οποίους ορίζεται η οντότητα. Αν δεν προστεθούν τιμές σε κάποια γνωρίσματα, τότε αυτά παίρνουν την default τιμή τους. Στους κόσμους όπου αυτά τα γνωρίσματα δεν ορίζονται, δεν επιτρέπεται από την εφαρμογή η προσθήκη τιμών. Από την κλάση control.CubeHandler παράγεται ένα στιγμιότυπο της κλάσης model.Entity και προστίθεται σε αυτό το αναγνωριστικό που είχε παραχθεί προηγουμένως. Προστίθεται επίσης το συνολο των

κόσμων όπου ορίζεται η οντότητα καθώς και ένα HashMap που περιλαμβάνει τις τιμές κάθε γνωρίσματος, σε κάθε κόσμο.

Η διαδικασία καταλήγει με την προσθήκη στον πίνακα R_Ctx των κόσμων όπου ορίζεται η οντότητα αυτή, την προσθήκη στον πίνακα Rc_A_Ctx των τιμών για τα γνωρίσματα που ορίζονται, την ανανέωση της κλάσης model.cube που αναπαριστά τη c-r σχέση στην οποία ανήκει η οντότητα και την ανανέωση της προβολής της σχέσης στο UI.

3.5.1.6 Αλλαγή υπάρχουσας οντότητας

Για να γίνει επεξεργασία μιας οντότητας σε μια c-r σχέση, η διαδικασία που ακολουθείται είναι η εξής:

Κατ' αρχάς ορίζονται οι κόσμοι στους οποίους θα ορίζεται πλέον η οντότητα και το νέο σύνολο τιμών για τα γνωρίσματα της οντότητας. Εάν πρόκειται να διατηρηθούν οι τιμές που υπήρχαν στην προηγούμενη κατάσταση της οντότητας, απλά δεν αλλάζουν. Επίσης, σε οποιοδήποτε σημείο της διαδικασίας, πριν την αποθήκευση των αλλαγών, είναι δυνατή η ακύρωση της διαδικασίας και η επαναφορά της οντότητας στην αρχική της κατάσταση. Για τη διεκπεραίωση του update, η κλάση control.CubeHandler συγκρίνει το νέο σύνολο κόσμων με το υπάρχον και προβαίνει στις εξής ενέργειες.

Διατηρεί τις τιμές των γνωρισμάτων για τους κόσμους που δεν αλλάζουν

Προσθέτει τις τιμές των γνωρισμάτων που ορίζονται στους νέους κόσμους

Διαγράφει όσους κόσμους πλέον δεν χρησιμοποιούνται (ταυτόχρονα διαγράφονται και οι τιμέςτους).

Στη συνέχεια παράγει ένα στιγμιότυπο της κλάσης model.Entity, προσθέτει το αναγνωριστικό του (το οποίο είναι αυτό της αρχικής οντότητας), προσθέτει το συνολο των κόσμων όπου ορίζεται και ένα HashMap που περιλαμβάνει τις τιμές κάθε γνωρίσματος, σε κάθε κόσμο.

Κατά τη διαδικασία αποθήκευσης της οντότητας, γίνεται έλεγχος αν υπάρχει ήδη οντότητα με αυτο το αναγνωριστικό και αφού προφανώς υπάρχει, καλεί τις συναρτήσεις για update.

Η διαδικασία καταλήγει ως εξής:

Προσθέτονται στον πίνακα R_Ctx οι νέοι κόσμοι όπου ορίζεται η οντότητα αυτή

Διαγράφονται από τον πίνακα R_Ctx οι κόσμοι όπου δεν ορίζεται πλέον.

Ανανεώνονται στον πίνακα Rc_A_Ctx οι τιμές για τα γνωρίσματα που ορίζονταν και πριν

Διαγράφονται στον πίνακα Rc_A_Ctx οι τιμές που αντιστοιχούσαν σε κόσμους που καταργήθηκαν

Προσθέτονται στον πίνακα Rc_A_Ctx τιμές για τους κόσμους που δημιουργήθηκαν

Ανανεώνεται το στιγμιότυπο της model.cube

3.5.1.7 Διαγραφή υπάρχουσας οντότητας

Για να διαγραφεί μια οντότητα σε μια c-r σχέση, αρχικά ανακτάται το αναγνωριστικό της οντότητας. Στη συνέχεια η κλάση model.CubeHandler εκτελεί τις εξής λειτουργίες.

Διαγράφει από τον πίνακα R_Ctx τις πλειάδες που αφορούν την οντότητα που πρόκειται να διαγραφούν.

Διαγράφει στον πίνακα Rc_A_Ctx τις πλειάδες που αφορούν την οντότητα που πρόκειται να διαγραφεί.

Ανανεώνει το στιγμιότυπο της model.cube

3.5.2 Λειτουργίες πράξεων

3.5.2.1 Υλοποίηση της c-r πράξης context project

Μια πράξη τύπου "attribute project", επιστρέφει από μια c-r σχέση τα γνωρίσματα εκείνα τα οποία περιλαμβάνονται στον ορισμό του ερωτήματος. Η πραγματοποίηση ενός τέτοιου ερωτήματος γίνεται ως εξής.

Φορτώνεται στην κλάση control.Project το στιγμιότυπο της κλάσης model.Cube η οποία αντιπροσωπεύει τον κύβο στον οποίο εφαρμόζεται το ερώτημα. Η κλάση model.Cube έχει ένα διάνυσμα το οποίο περιέχει τα γνωρίσματα του κύβου καθώς και ένα διάνυσμα το οποίο περιέχει στιγμιότυπα της κλάσης model.Entity. Η κλάση Entity περιέχει μια αντιστοίχιση γνωρισμάτων-τιμών. Η πράξη της προβολής υλοποιείται ουσιαστικά δημιουργώντας ένα νέο στιγμιότυπο κύβου όπου έχουν αφαιρεθεί από τα διανύσματα αυτά τα γνωρίσματα τα οποία δεν υπάρχουν στο νέο σύνολο. Η διαδικασία ακολουθεί την εξής μεθοδολογία.

Παρέχονται σαν είσοδοι τα εξής :

Το όνομα της c-r σχέσης πάνω στην οποία θα γίνει η προβολή

Το σύνολο των γνωρισμάτων που θα περιλαμβάνει η προβολή

Όνομα για την προσωρινή c-r σχέση.

Τα δεδομένα αυτά δίνονται σαν είσοδος στη συνάρτηση attributeProject της κλάσης control.Project, η οποία προβαίνει στην εξής επεξεργασία.

Ελέγχει αν όλα τα γνωρίσματα που ορίζονται στην προβολή υπάρχουν και στον ορισμό c-r σχέσης

Επεξεργάζεται τα γνωρίσματα της c-r σχέσης και αφαιρεί όσα δεν ανήκουν στο σύνολο της προβολής

Επεξεργάζεται τις οντότητες της c-r. Ελέγχεται αν όλα τα γνωρίσματα που ορίζονται στην προβολή υπάρχουν και στον ορισμό της c-r σχέσης και αφαιρούνται όσα γνωρίσματα δεν ανήκουν στο προβαλλόμενο σύνολο

Τέλος, δημιουργεί μια c-r σχέση στην οποία προστίθενται οι οντότητες που προέκυψαν στο προηγούμενο βήμα, τα γνωρίσματα που περιλαμβάνονται στην πράξη προβολής και ένα όνομα για τον προσωρινό κύβο.

3.5.2.2 Υλοποίηση της c-r πράξης facet select

Μια πράξη τύπου Facet Select, επιστρέφει το σύνολο των facets κάθε οντότητας που πληρούν κάποιες συνθήκες. Είναι σημαντικό να σημειωθεί πως η πράξη Facet Select αποτιμάται κάθε φορά στο σύνολο των οντοτήτων μιας c-r σχέσης και όχι επιλεκτικά σε κάποιες από αυτές.

Οι συνθήκες με βάση τις οποίες γίνεται η αποτίμηση, μπορούν να έχουν δυο μορφές :

Μορφή 1η :

<attribute - operation – vaue>

Στην περίπτωση αυτή συγκρίνεται ένα attribute με κάποια προκαθορισμένη τιμή

Μορφή 2η :

<attribute - operation - attribute – world>

Στην περίπτωση αυτή συγκρίνεται ένα attribute με την τιμή που έχει το ίδιο ή κάποια άλλο attribute (της ίδιας οντότητας) σε κάποιον από τους κόσμους όπου ορίζεται η οντότητα. Ένα παράδειγμα θα είχαμε αν ζητούνταν τα facets ενός προϊόντος των οποίων η τιμή είναι μεγαλύτερη από την τιμή που το προϊόν αυτό είχε σε κάποια συγκεκριμένη ημερομηνία.

Ένα παράδειγμα όπου τα δυο attribute είναι διαφορετικά είναι η περίπτωση όπου ζητάμε όλα τα facets ενός προϊόντος, όπου ο χρόνος(διάστημα) παραγωγής είναι μικρότερος από το χρόνο διάθεσης (σε κάποιο κόσμο) (οι κόσμοι θα μπορούσε απλώς να είναι διαφορετικές χώρες και εποχές).

Η λειτουργία υλοποιείται με την εξής μεθοδολογία.

Παρέχονται σαν είσοδοι :

Το όνομα της c-r σχέσης πάνω στον οποία θα γίνει η προβολή.

Οι συνθήκες που πρόκειται να επαληθευτούν.

Οι συνθήκες μπορούν να είναι περισσότερες από μια και θεωρούνται ότι συνδέονται με το λογικό "AND". Άλλες λογικές πράξεις δεν υλοποιούνται από την εφαρμογή.

Αν το πεδίο value είναι αριθμός, οι τελεστές <> επιστρέφουν λογικό αποτέλεσμα κατά τις συγκρίσεις παρόλο που το πεδίο είναι δηλωμένο σαν συμβολοακολουθία (από πλευράς υλοποίησης στον server της postgre). Είναι όμως ευθύνη του χρήστη να φροντίσει ώστε οι τελεστές να μπορούν να αποτιμηθούν στα δεδομένα που έχει δώσει.

Το όνομα της προσωρινής c-r σχέσης στην οποία θα αποθηκευτεί το αποτέλεσμα Με βάση αυτά τα δεδομένα, η κλάση control.FacetSelect προβαίνει στις εξής ενέργειες:

Παράγει έναν ψευδοκώδικα σε C-R SQL, ο οποίος αντιπροσωπεύει την πράξη (συνάρτηση makequery()).

Υλοποιεί την πράξη.

Προϋπόθεση είναι η c-r σχέση πάνω στην οποία θα γίνει η πράξη να είναι αποθηκευμένη στον server, γιατί είναι αναγκαία η χρήση sql κώδικα.

Για ένα ερώτημα τύπου 1, <attribute - operation – value>, το ερώτημα που δημιουργείται είναι της μορφής :

query : select world from Ctx-Attr where entity_id = <desired entity> and attribute = <attribute_name> and value <operation> <value>

Για ένα ερώτημα τύπου 2, <attribute - operation - attribute – world>, το ερώτημα που δημιουργείται είναι της μορφής :

query : select world from Ctx-Attr where entity_id = <desired entity> and attribute = <attribute_name> and value <operation> (select value from Ctx-Attr where entity_id = <same entity> and attribute = <another or the same attribute> and world = <value>)

Στη συνέχεια, εφόσον έχουμε περισσότερες από μια συνθήκες, τα ερωτήματα ενώνονται σε ένα, με τη χρήση του τελεστή INTERSECT. Έτσι δημιουργείται ένα ερώτημα της μορφής query INTERSECT query INTERSECT ...

Το κάθε query μας επιστρέφει το σύνολο των κόσμων μιας οντότητας όπου επαληθεύεται μια συγκεκριμένη συνθήκη. Στη συνέχεια, μέσω του τελεστή INTERSECT, επιστρέφεται η τομή των συνόλων που προκύπτουν για κάθε συνθήκη.

Η πράξη αυτή, του συνολικού δηλαδή query, εφαρμόζεται ξεχωριστά για κάθε οντότητα της c-r σχέσης και τελικά η σχέση που προκύπτει περιλαμβάνει τις οντότητες εκείνες που επιστρέφουν μη κενό σύνολο, οριζόμενες μόνο στους κόσμους του συνόλου που επιστρέφει το ερώτημα.

3.5.2.3 Υλοποίηση της c-r πράξης entity select

Μια πράξη τύπου Entity Select, επιστρέφει το σύνολο των entities που πληρούν κάποιες συνθήκες και είναι το αντίστοιχο της σχεσιακής πράξης της. Οι συνθήκες με βάση τις οποίες γίνεται η αποτίμηση, μπορούν να έχουν δυο μορφές :

Μορφή 1η :

<attribute -worlds- operation – vaue>

Στην περίπτωση αυτή συγκρίνεται η τιμή που λαμβάνει ένα attribute σε κάποιο σύνολο κόσμων με κάποια προκαθορισμένη τιμή. Το σύνολο αυτό των κόσμων διακρίνεται σε τρεις βασικές κατηγορίες:

<όλοι οι κόσμοι> που σε μορφή κώδικα c-r sql αντιστοιχεί στην πράξη σentity (<attribute>[]<operation><value>) σχέση και υποδηλώνει πως η συνθήκη πρέπει να επαληθεύεται αληθώς στο σύνολο των κόσμων όπου ορίζεται το attribute.

<συγκεκριμένοι κόσμοι> που σε μορφή κώδικα c-r sql αντιστοιχεί στην πράξη σentity (<attribute>[w1,w5]<operation><value>) σχέση και δηλώνει πως η συνθήκη πρέπει να επαληθεύεται σε αυτούς συγκεκριμένα τους κόσμους.

<οποιοσδήποτε κόσμος> που σε μορφή κώδικα c-r sql αντιστοιχεί στην πράξη

σentity (<attribute><operation><value>) σχέση και δηλώνει πως η συνθήκη αρκεί να επαληθεύεται σε οποιονδήποτε από τους κόσμους όπου ορίζεται το γνώρισμα.

Το σύμβολο [] είναι ο Universal Context Specifier, που σημαίνει ότι γίνεται αναφορά σε όλους τους κόσμους στους οποίους ορίζεται το attribute.

Μορφή 2η :

<attribute – world - operation - attribute – world>

Στην περίπτωση αυτή συγκρίνεται ένα attribute με την τιμή που έχει το ίδιο ή κάποια άλλο attribute (της ίδιας οντότητας) σε κάποιον από τους κόσμους όπου ορίζεται η οντότητα. Για το πρώτο attribute ισχύουν τα όσα γράφονται στην προηγούμενη ακριβώς παράγραφο, ενώ για το δεύτερο, τα όσα γράφονται στη 2η μορφή του Facet Select.

Η λειτουργία υλοποιείται με την εξής μεθοδολογία.

Παρέχονται σαν είσοδοι :

Το όνομα της c-r σχέσης πάνω στην οποία θα γίνει η προβολή Οι συνθήκες που πρόκειται να επαληθεφτούν. Οι συνθήκες μπορούν να είναι περισσότερες από μια και θεωρείται ότι συνδέονται με το λογικό “AND”. Άλλες λογικές πράξεις δεν υλοποιούνται από την εφαρμογή.

Αν το πεδίο value είναι αριθμός, οι τελεστές <> επιστρέφουν λογικό αποτέλεσμα κατά τις συγκρίσεις παρόλο που το πεδίο είναι δηλωμένο σαν συμβολοακολουθία (από πλευράς υλοποίησης στον server της postgre). Είναι όμως ευθύνη του χρήστη να φροντίσει ώστε οι τελεστές να μπορούν να αποτιμηθούν στα δεδομένα που έχει δώσει.

Το όνομα της προσωρινής c-r σχέσης στην οποία θα αποθηκευτεί το αποτέλεσμα

Με βάση αυτά τα δεδομένα, η κλάση control.EntitySelect προβαίνει στις εξής ενέργειες

Παράγει έναν ψευδοκώδικα σε C-R SQL, ο οποίος αντιπροσωπεύει την πράξη (συνάρτηση getSQL()).

Υλοποιεί την πράξη.

Προϋπόθεση είναι η c-r σχέση πάνω στην οποία θα γίνει η πράξη να είναι αποθηκευμένη στον server, γιατί είναι αναγκαία η χρήση sql κώδικα. Το πρώτο στάδιο της επεξεργασίας, καθορίζει σε ποια κατηγορία ανήκει η κάθε συνθήκη του ερωτήματος από πλευράς κόσμων και σύμφωνα με τα όσα αναφέρθηκαν παραπάνω.

Περίπτωση 1: <όλοι οι κόσμοι>

Αρκεί να συγκριθούν οι κόσμοι στους οποίους ορίζεται μια οντότητα, με τους κόσμους στους οποίους το attribute της entity πληρεί τη συνθήκη και τα set να είναι ισοδύναμα. Για να ελεγθούν αν τα δυο σύνολα είναι ισοδύναμα, τα αφαιρούνται μεταξύ τους και αν το αποτέλεσμα είναι το κενό σύνολο είναι ίσα. Επίσης, αν ένα σύνολο είναι κενό, η απαρίθμηση των στοιχείων του επιστρέφει 0. Η σύγκριση γίνεται με το εξής ερώτημα :

Μορφή 1η:

select count(world) from Rc_Ai_Ctx where world in

((select world from Rc_Ctx where entity_id = entity_id)

EXCEPT

(select world from Rc_Ai_Ctx where attribute = 'attributeName' and entity_id = entity_id

and value <operation><value>))

Μορφή 2η:

select count(world) from Rc_Ai_Ctx where world in

((select world from Rc_Ctx where entity_id = entity_id)

EXCEPT

(select world from Rc_Ai_Ctx where attribute = 'attributeName' and entity_id = entity_id

and value <operation> (select value from Rc_Ai_Ctx where entity_id = entity_id and

attribute = attribute and world = world))

Τα ερωτήματα αυτά αποτιμώνται ξεχωριστά για κάθε οντότητα της c-r σχέσης. Όποια οντότητα ικανοποιεί τις συνθήκες προστίθεται στο διάνυσμα που περιέχει το αποτέλεσμα.

Περίπτωση 2: <συγκεκριμένοι κόσμοι>

Για κάθε κόσμο στο οποίο επιθυμείται να επαληθευτεί η συνθήκη, δημιουργείται ένα ερώτημα ως εξής :

Μορφή 1η:

select count(entity_id) from Rc_Ai_Ctx where entity_id = 'id' and attribute = 'attributeName'

and value <operation><value> and world ='<world>'

Μορφή 2η:

select count(entity_id) from Rc_Ai_Ctx where entity_id = 'id' and attribute = 'attributeName'

and value <operation>

(select value from Rc_Ai_Ctx where entity_id = entity_id and attribute = attribute and

world = world)

and world ='<world>'

Αν το αποτέλεσμα είναι θετικό, τότε θεωρείται πως για αυτόν το κόσμο η αποτίμηση επαληθεύεται και ελέγχεται ο επόμενος. Αν για όλους τους κόσμους της οντότητας στους οποίους θέλουμε να ισχύει η συνθήκη επιστραφεί θετικό αποτέλεσμα, τότε θεωρείται πως η οντότητα συνολικά επαληθεύει τη συνθήκη και συμπεριλαμβάνεται στο αποτέλεσμα. Η διαδικασία αυτή εκτελείται για κάθε οντότητα της c-r σχέσης στην οποία εκτελείται η πράξη.

Περίπτωση 3: <οποιοσδήποτε κόσμος>

Για την περίπτωση αυτή αρκεί να επιστρέψουν θετικό αποτέλεσμα τα ερωτήματα που παρουσιάζονται παρακάτω. Σε αυτή την περίπτωση, το αποτέλεσμα ισούται με το πλήθος των κόσμων στους οποίους επαληθεύεται η συνθήκη. Η διαδικασία αυτή εκτελείται για κάθε οντότητα της c-r σχέσης στην οποία γίνεται η πράξη.

Μορφή 1η:

select count(world) from Rc_Ai_Ctx where attribute = 'attributeName' and entity_id =

entity_id and value <operation><value>

Μορφή 2η:

select count(world) from Rc_Ai_Ctx

where attribute = 'attributeName' and entity_id = entity_id and value <operation>

(select value from Rc_Ai_Ctx where entity_id = entity_id and attribute = attribute and

world = world

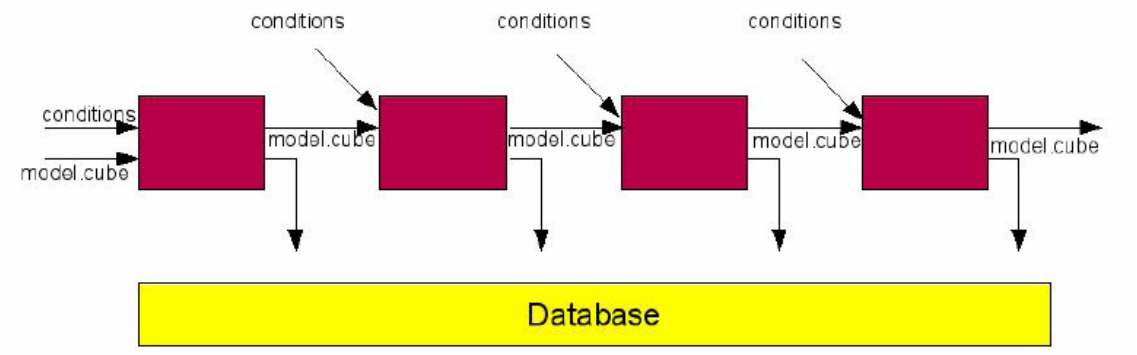
Μια πράξη entity select μπορεί να περιέχει συνθήκες διάφορων τύπων. Το τελικό αποτέλεσμα, προκύπτει από την τομή όλων των επιμέρους αποτελεσμάτων που προκύπτουν για κάθε συνθήκη.

3.5.2.4 Υλοποίηση της c-r πράξης του καρτεσιανού γινόμενου

Η πράξη του καρτεσιανού γινόμενου στην c-r άλγεβρα είναι αντίστοιχη του καρτεσιανού γινόμενου της σχεσιακής άλγεβρας. Αρκεί να σημειωθεί πως ο συνδυασμός των σχέσεων γίνεται ανάμεσα στα tupples που ανήκουν στον ίδιο κόσμο. Για λόγους απλότητας μπορεί κανείς να θεωρήσει πως κάθε c-r σχέση αποσυντίθεται σε ένα σύνολο σχέσεων του σχεσιακού μοντέλου, μια για κάθε κόσμο. Στη συνέχεια παίρνονται τα καρτεσιανά γινόμενα ανά δυο των σχέσεων που αντιστοιχούν στον ίδιο κόσμο και τέλος επανενώνονται τα επιμέρους γινόμενα για να εξαχθεί στο τελικό αποτέλεσμα για το c-r καρτεσιανό γινόμενο.

3.5.2.5 Υλοποίηση σύνθετων ερωτημάτων

Η εφαρμογή δίνει τη δυνατότητα να συνταχθεί ένα πλήθος από πράξεις όπως αυτές που αναφέρθηκαν προηγουμένως, με σχετική ευκολία. Κάθε μια από τις παραπάνω πράξεις υλοποιείται από μια κλάση η οποία σαν είσοδο χρειάζεται την c-r σχέση πάνω στην οποία θα γίνει η πράξη και ένα σύνολο από συνθήκες. Το αποτέλεσμα είναι μια νέα c-r σχέση, όπως φαίνεται και στο παρακάτω διάγραμμα. (μια c-r σχέση αντιπροσωπεύεται από ένα στιγμιότυπο της κλάσης model.Cube.



Εικόνα 3.9 Αλληλουχία context πράξεων

4

Υλοποίηση

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

4.1.1 Γλώσσα Προγραμματισμού Java

Σαν γλώσσα προγραμματισμού επιλέχθηκε η Java, στην έκδοση 1.5. Τα βασικά πλεονεκτήματα που ώθησαν στην επιλογή της είναι οι εξής :

• Platform Independent. Η εφαρμογή μπορεί να λειτουργήσει εύκολα σε οποιοδήποτε λειτουργικό σύστημα.

• Η java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, παρέχοντας τη δυνατότητα για ιδιαίτερα καθαρές υλοποιήσεις.

• Έχει ένα πλήθος χρήσιμων libraries οι οποίες διακρίνονται και για την συνεχή τους συντήρηση.

Η εφαρμογή έχει δοκιμαστεί με JVM 1.4 και 1.5.

Αξίζει να σημειωθεί πως όσον αφορά την άδεια χρήσης της java υπάρχουν διάφορα ζητήματα σχετικά με το κατά πόσον είναι μια γλώσσα ανοιχτού λογισμικού και για αυτό έχουν αναπτυχθεί διάφορες προσπάθειες για να φτιαχτεί μια ελεύθερη υλοποίηση του java api. Μια από τις σημαντικότερες είναι το Classpath Project.

4.1.2 Σύστημα Διαχείρισης Βάσεων Δεδομένων PostgreSQL

Η PostgreSQL είναι Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων ανοιχτού λογισμικού. Φημίζεται για την σταθερότητα και την προστασία των δεδομένων, ενώ παρόλο που είναι ένα έργο το οποίο βασίζεται στην ανάπτυξη από μια μεγάλη κοινότητα θεωρείται ένα σύστημα παραγωγής και έχει δυνατότητες που λίγα ακόμα συστήματα έχουν. Λειτουργεί σε όλα τα κύρια λειτουργικά συστήματα, συμπεριλαμβάνοντας το GNU/Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), και τα Windows. Η PostgreSQL ανταποκρίνεται πλήρως στις προδιαγραφές ACID, έχει πλήρη υποστήριξη για εξωτερικά κλειδιά, joins, views και stored procedures (σε διάφορες γλώσσες). Περιλαμβάνει τους περισσότερους SQL92 και SQL99 data types, συμπεριλαμβάνοντας INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, και TIMESTAMP. Επίσης υποστηρίζει την αποθήκευση μεγάλων δυαδικών αντικειμένων, συμπεριλαμβάνοντας εικόνες, ήχους ή video. Έχει native programming interfaces για γλώσσες όπως οι C/C++, Java, Perl, Python, Ruby, Tcl, ODBC.
Για την εφαρμογή χρησιμοποιήθηκε η έκδοση 8.0 , ενώ έχει δοκιμαστεί και με την 7.3.

4.1.3 Λειτουργικό Σύστημα GNU/Linux

Για την ανάπτυξη της εφαρμογής αποφασίστηκε να χρησιμοποιηθεί το λειτουργικό σύστημα GNU/Linux. Επιλέξαμε αρχικά τη διανομή Debian και στη συνέχεια τη διανομή Ubuntu, η οποία βασίζεται στη Debian. Ο κύριος λόγος που χρησιμοποιήθηκε το συγκεκριμένο λειτουργικό σύστημα είναι ότι αποτελεί ελεύθερο λογισμικό . Από πλευράς δυνατοτήτων επίσης, το λειτουργικό σύστημα GNU/Linux αποτελεί ένα πλήρες σύνολο, ενώ αξίζει να σημειωθεί πως λειτουργεί σε ένα τεράστιο εύρος επεξεργαστών.
Η εφαρμογή έχει δοκιμαστεί τόσο σε GNU/Linux όσο και σε Windows.

4.1.4 Περιβάλλον Προγραμματισμού Eclipse

Σαν περιβάλλον προγραμματισμού επιλέχθηκε το Eclipse. Το περιβάλλον αυτό αναπτύσσεται από την IBM, η οποία όμως έχει δημοσιοποιήσει τον κώδικα του. Αν και δεν χρησιμοποιείται για μεγάλο χρονικό διάστημα (συγκρινόμενο με άλλα περιβάλλοντα) παρέχει ένα ιδιαίτερα εύχρηστο user interface και ένα ιδιαίτερα αποτελεσματικό σύστημα επεκτάσεων με τη χρήση plugins. Μέσω αυτό γίνεται απλή η προσθήκη δυνατοτήτων που χρειαζόμασταν για την ανάπτυξη της εφαρμογής μας, όπως ο Visual Editor για τη σχεδίαση του γραφικού περιβάλλοντος και το Junit για τη διενέργεια των test.

4.2 Λεπτομέρειες υλοποίησης

4.2.1 Μεθοδολογία Ανάπτυξης

4.2.1.1 1ο στάδιο : καθορισμός απαιτήσεων

Το πρώτο βήμα για την υλοποίηση του προγραμματιστικού τμήματος της διπλωματικής, ήταν ο καθορισμός των απαιτήσεων για την εφαρμογή που θα προέκυπτε. Κάποιες από τις προδιαγραφές τέθηκαν κατά την ανάθεση της διπλωματικής και άλλες καθορίστηκαν από την ομάδα ανάπτυξης. Το σύνολο των απαιτήσεων είναι οι εξής :

• Ο στόχος της εφαρμογής είναι η επίδειξη του C-R σχεσιακού μοντέλου και όχι η λειτουργία του σε πραγματικές συνθήκες εργασίας. Με βάση αυτό δεν υπάρχει ανάγκη για βελτιστοποίηση των επιδόσεων. Ο στόχος μας είναι η υλοποίηση όσο γίνεται περισσοτέρων λειτουργιών με τον απλούστερο τρόπο. Το κόστος σε υπολογιστική ισχύ, χρόνο και χώρο είναι ένας παράγοντας ο οποίος αν και λαμβάνεται υπ' όψιν δεν έχει καθοριστική σημασία. (Παρόλα αυτά, σε αρκετά σημεία έγινε κατά την τελική φάση ανάπτυξης βελτίωση των αλγόριρθμων που χρησιμοποιούνταν).

• Η εφαρμογή θέλουμε να έχει τη δυνατότητα να επεκταθεί και να βελτιωθεί με τον απλούστερο τρόπο. Επιδιώκουμε ένα σχεδιαστικό μοντέλο το οποίο θα είναι το δυνατόν καθαρότερο και κατανοητό. Για το λόγο αυτό τα βασικά τμήματα της εφαρμογής πρέπει να είναι κατά το δυνατόν ανεξάρτητα και να υπάρχει η δυνατότητα να χρησιμοποιηθούν σε μεταγενέστερες υλοποιήσεις.

• Από πλευρά χρηστών κάνουμε τις εξής παραδοχές :

1. Τα συστήματα διαχείρισης βάσεων δεδομένων απευθύνονται κατά κανόνα σε ένα κοινό που έχει ένα σχετικά αναβαθμισμένο επίπεδο όσον αφορά τη χρήση Η/Υ, κάτι που συνεπάγεται συνήθως μικρή έμφαση στο user interface.

2. Ταυτόχρονα όμως, η ευκολία χρήσης μειώνει το χρόνο εκμάθησης της εφαρμογής δίνοντας τη δυνατότητα στο χρήστη να ασχοληθεί με την ουσία γρηγορότερα και ευκολότερα. Έτσι, ενισχύονται σημαντικά οι πιθανότητες να ασχοληθεί κανείς με το μοντέλο που υλοποιούμε και να ενδιαφερθεί για την περαιτέρω ανάπτυξή του.

3. Ο χρήστης της εφαρμογής δεν αναμένει ένα πρόγραμμα με πλήρεις λειτουργικές δυνατότητες αλλά ένα πρόγραμμα που θα του παρουσιάσει τις δυνατότητες του C-R σχεσιακού μοντέλου.

• Το user interface θα βασίζεται στη γραφική αλληλεπίδραση, δίνοντας συγκεκριμένες μόνο επιλογές για το τι μπορεί να κάνει ένας χρήστης. Δεν θα υπάρχει η δυνατότητα να δοθούν εντολές με κώδικα σε μια εκτεταμένη εκδοχή της sql (c-r sql) και στη δυνατότητα να γίνει

επεξεργασία μέσω ενός parser. Ένας από τους λόγους αυτής της προδιαγραφής είναι και το γεγονός ότι κατά τα αρχικά στάδια ανάπτυξης της εφαρμογής δεν είχε ακόμα θεμελιωθεί η C-R SQL . Θέλουμε όμως να υπάρχει η δυνατότητα να ενσωματωθεί ένας parser στην εφαρμογή με εύκολο τρόπο.

• Το σχήμα της C-R σχεσιακής βάσης που παράγεται θα είναι στατικό. Δηλαδή δεν θα δίνεται η δυνατότητα να γίνουν αλλαγές στο σχήμα μετά την αρχική δημιουργία και αποθήκευσή του. Η επιλογή αυτή έγινε για λόγους απλοποίησης της εφαρμογής και μείωσης του χρόνου ανάπτυξης κατά την εκπόνηση των αρχικών προδιαγραφών για την ανάπτυξη του προγράμματος.

4.2.1.2 2ο στάδιο : πρόχειρη σχεδίαση

• Διαγραμματική αναπαράσταση των λειτουργιών που θέλουμε να υλοποιήσουμε (πχ τι δεδομένα θα χρειαστούμε για να φτιάξουμε μια νέα βάση, τι αλληλεπίδραση θα χρειαστεί με το χρήστη).

• Βασική δομή κλάσεων

• Βασική σχεδίαση του user interface αρκετές.

4.2.1.3 3ο στάδιο : ανάπτυξη βασικών στοιχείων της εφαρμογής και test cases

Με βάση το παραπάνω, αναπτύξαμε ένα αρχικό πρόγραμμα το οποίο ήταν ικανό να δημιουργεί μια νέα context-retional βάση δεδομένων και να προσθέτει σε αυτή c-r σχέσεις. Για την ανάπτυξη αυτού του τμήματος, γράφτηκαν και οι αντίστοιχες test cases, έτσι ώστε να γίνουν οι απαραίτητες δοκιμές, βελτιώσεις αλλά και να είναι απλούστερη η ανάπτυξη του κυρίως προγράμματος.

Το gui είναι υποτυπώδες σε αυτό το στάδιο. Ορισμός και υλοποίηση ενός πρώτου user interface για εκείνα τα στοιχεία της εφαρμογής τα οποία απατούν αλληλεπίδραση με τον χρήστη. Το πρώτο αυτό interface, δεν ήταν το τελικό, αλλά παρείχε μια πρώτη εικόνα όσων είχαμε ορίσεις στις προδιαγραφές, επιτρέποντάς μας να κάνουμε έναν πρώτο έλεγχο και επανεξέταση της διαδικασίας που είχαμε ορίσει.

Βασικές κλάσεις και συναρτήσεις που αναπτύχθηκαν κατά τη διάρκει αυτού το σταδίου ήταν οι : (Database , Cubel, Attribute, Dimension, Worlds, CreateDatabase, CreateCube, Postgredataaccess, Generaldataaccess)

4.2.1.4 4ο στάδιο : τελική επανασχεδίαση της εφαρμογής

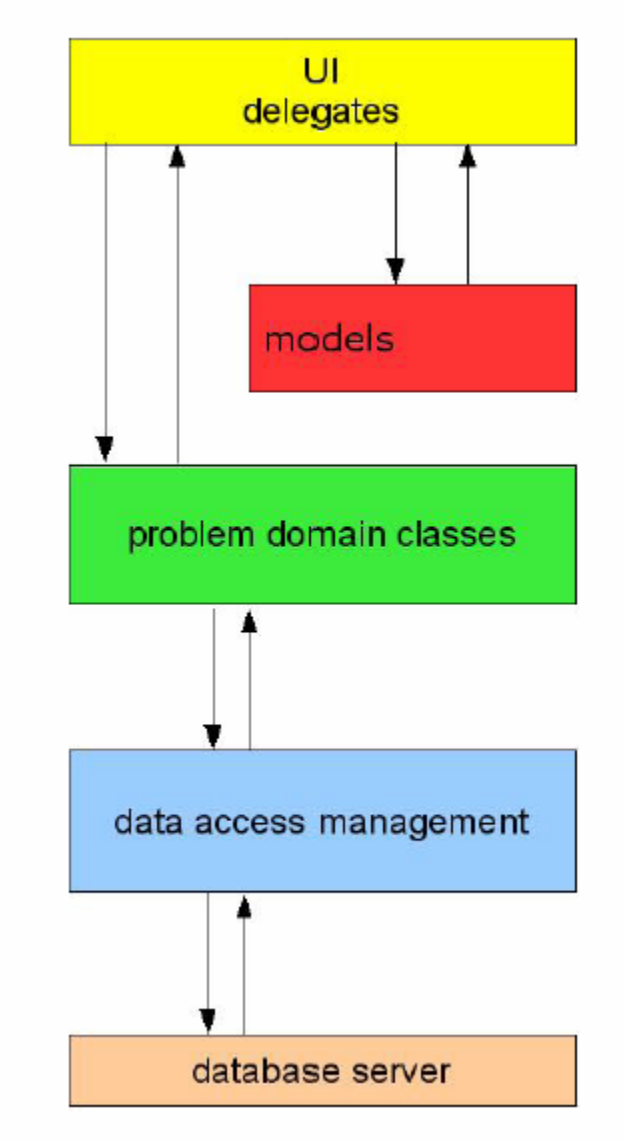
• Σχεδιασμός και τεκμηρίωση της ιεραρχίας κλάσεων.

• Σχεδιασμός και τεκμηρίωση της κάθε κλάσης και interface

Γενικα, οι κλάσεις μας χωρίζονται σε τρεις βασικές κατηγορίες :

1. User-interface classes / κλάσεις User-interface : Οι κλάσεις που ευθύνονται για το User interface, περιλαμβάνοντας παράθυρα, διαλόγους κλπ
2. Problem domain classes / κλάσης προβλήματος: Κλάσεις οι οποίες αναπαριστούν αντικείμενα τα οποία είχαμε ορίσει κατά τον ορισμο του προβλήματος.
3. Data management classes / κλάσεις χειρισμού πληροφορίας : Κλάσεις που χρησιμοποιούνται στο χειρισμό αντικειμένων ή δεδομένων.

4.3 Η αρχιτεκτονική της εφαρμογής Cubrik C - R ADMIN



Εικόνα 4.1. Αρχιτεκτονική της εφαρμογής

Η εφαρμογή είναι δομημένη γύρω από ένα σύνολο προδιαγραφών που θέσαμε και οι οποίες είναι οι εξής :

• Η επικοινωνία με τον server του συστήματος βάσεων δεδομένων να είναι όσο γίνεται ανεξάρτητη από την υπόλοιπη υλοποίηση αλλά και από το συγκεκριμένο ΣΔΒΔ που χρησιμοποιούμε. Με τον τρόπο αυτό, διατηρούμε ευελιξία στον τρόπο με τον οποίο το C-R SQL μοντέλο μεταφράζεται σε πράξεις στο σχεσιακό μοντέλο.

• Να μπορούμε οποιαδήποτε στιγμή και με ευκολία να παρέμβουμε στον τρόπο με τον οποίο υλοποιούνται οι βασικές λειτουργίες του c-r μοντέλου χωρίς να δημιουργηθεί πρόβλημα στο υπόλοιπο σύστημα.

• Η κάθε πράξη που υλοποιούμε να χρειάζεται συγκεκριμένα δεδομένα και να έχει συγκεκριμένη έξοδο, ενώ ταυτόχρονα να υπάρχει μια ενιαία μέθοδος για όλες τις πράξεις.

• Ο τρόπος διασύνδεσης με το χρήστη να είναι ανεξάρτητος της υπόλοιπης υλοποίησης. Έτσι αν αποφασίσουμε να χρησιμοποιήσουμε ένα διαφορετικό περιβάλλον επικοινωνίας, ή ακόμα να μετατρέψουμε την εφαρμογή μας σε έναν CR-SQL server αυτό να μπορεί να υλοποιηθεί με τη μεγαλύτερη δυνατή ευκολία.

Προκειμένου να υλοποιηθούν αυτές οι απαιτήσεις, η εφαρμογή σχεδιάστηκε με την εξής ιεραρχία κλάσεων.

4.3.1 User Interface Classes

Οι κλάσεις αυτές οι οποίες βρίσκονται στο package view, αναλαμβάνουν τη διασύνδεση με το χρήστη. Επίσης, περιέχουν και ένα σύνολο συναρτήσεων οι οποίες αναλαμβάνουν να κάνουν ελέγξουν και να διαμορφώσουν σε έναν αρχικό βαθμό τα δεδομένα που εισάγει ο χρήστης προτού καλέσουν τις κλάσεις ελέγχου.

AttributeProjectPanel	User Interface για την πράξη attribute project
AttributeProjectPanel1	Βοηθητική κλάση που καλείται από την κλάση CombinedOperations για την προσθήκη μιας πράξης attribute project στο σύνολο πράξεων
CombinedOperations	User Interface για την υλοποίηση ενός συνόλου διαδοχικών πράξεων
ContextProjectPanel	User Interface για την πράξη context project
ContextProjectPanel1	Βοηθητική κλάση που καλείται από την κλάση CombinedOperations για την προσθήκη μιας πράξης context project στο σύνολο πράξεων
CreateDatabase	User Interface για τη δημιουργία μιας C-R σχεσιακής βάσης

	δεδομένων
DropAllTempCubes	User Interface για τη διαγραφή του συνόλου των προσωρινών CR σχέσεων που έχουν αποθηκευτεί σε μια βάση δεδομένων
DropCube	User Interface για τη διαγραφή μιας C-R σχέσης από μια βάση δεδομένων
EditCube	User Interface για την αλλαγή της δομής μιας C-R σχέσης
EditCubes	User Interface που μας επιτρέπει την επεξεργασία μιας C-R σχέσης. Μέσω αυτού του interface μπορούμε να εισάγουμε, να επεξεργαστούμε και να διαγράψουμε οντότητες.
JoinPanel	User Interface για την πράξη cartesian product
JoinPanel1	Βοηθητική κλάση που καλείται από την κλάση CombinedOperations για την προσθήκη μιας πράξης cartesian product στο σύνολο πράξεων
LabelWithSeparator	κλάση που χρησιμοποιείται από το gui
MainPanelNew	Αρχική κλάση της εφαρμογής. Περιέχει το αρχικό user interface.
myTree	Κλάση που χρησιμοποιείται από το gui.
NewCube	User Interface για τη δημιουργία μιας νέας C-R σχέσης.
SelectEntityPanel	User Interface για την πράξη entity select
SelectEntityPanel1	Βοηθητική κλάση που καλείται από την κλάση CombinedOperations για την προσθήκη μιας πράξης entity select στο σύνολο πράξεων
SelectFacetPanel	User Interface για την πράξη facet select
SelectFacetPanel1	Βοηθητική κλάση που καλείται από την κλάση CombinedOperations για την προσθήκη μιας πράξης facet select στο σύνολο πράξεων
ShowCube	Βοηθητική κλάση για το gui που συστηματοποιεί τον τρόπο προβολής των c-r σχέσεων.
TreeHandler	Κλάση που χρησιμοποιείται από το gui.
UseCube	User Interface για την επεξεργασίας μιας νέας C-R σχέσης. Μέσω αυτού μπορούμε να εισάγουμε, διαγράψουμε και επεξεργαστούμε οντότητες.

WelcomePanel1	Κλάση που χρησιμοποιείται από το gui.για την εισαγωγή των στοιχείων του χρήση.

4.3.2 Model classes

Οι κλάσεις αυτές βρίσκονται στο package model και αναπαριστούν οντότητες που έχουμε ορίσει κατά το σχεδιασμό και αφορούν είτε τη λογική της εφαρμογής είτε κάποια χρηστικά μοντέλα.

Attribute	Αναπαριστά ένα Attribute.
Cube	Αναπαριστά μια C-R σχέση.
Database	Αναπαριστά μια C-R βάση δεδομένων.
Dimension	Αναπαριστά μια διάσταση.
Entity	Αναπαριστά μια C-R οντότητα.
SelectEntityModel	Αναπαριστά μια πράξη entity select.
SelectFacetModel	Αναπαριστά μια πράξη facet select.
TableModel	
TableModel1	
TableModelEntitiesPerWorlds	
TableModelForAttributes	
TableModelForCubes	
TableModelForCubesNew	
TableModelForWorlds	
Worlds	Αναπαριστά έναν κόσμο.

4.3.3 Problem Domain classes

Οι κλάσεις αυτές βρίσκονται στο package control και υλοποιούν τους αλγόριθμους επίλυσης του προβλήματος.

ControlCenter	Κλάση η οποία περιέχει κυρίως συναρτήσεις ελέγχου της σύνδεσης και υποστηρικτικές στη συνολική λειτουργία της εφαρμογής
Join	Κλάση η οποία χειρίζεται την πράξη του cartesian product

Project	Κλάση η οποία χειρίζεται τις πράξεις προβολής
SelectEntity	Κλάση η οποία χειρίζεται την πράξη του select entity
SelectFacet	Κλάση η οποία χειρίζεται την πράξη του select facet

4.3.4 Data Access Management Classes

Οι κλάσεις αυτές αναλαμβάνουν την επικοινωνία της εφαρμογής με το σύστημα διαχείρισης βάσεων δεδομένων.

GeneralDataAccess	(interface) : Χρησιμοποιείται σαν abstraction layer για την πρόσβαση στο σύστημα διαχείρισης βάσεων δεδομένων.
PostgreDataAccess	Κλάση που περιέχει την υλοποίηση όλων των λειτουργιών που απαιτούν άμεση πρόσβαση στη βάση δεδομένων.

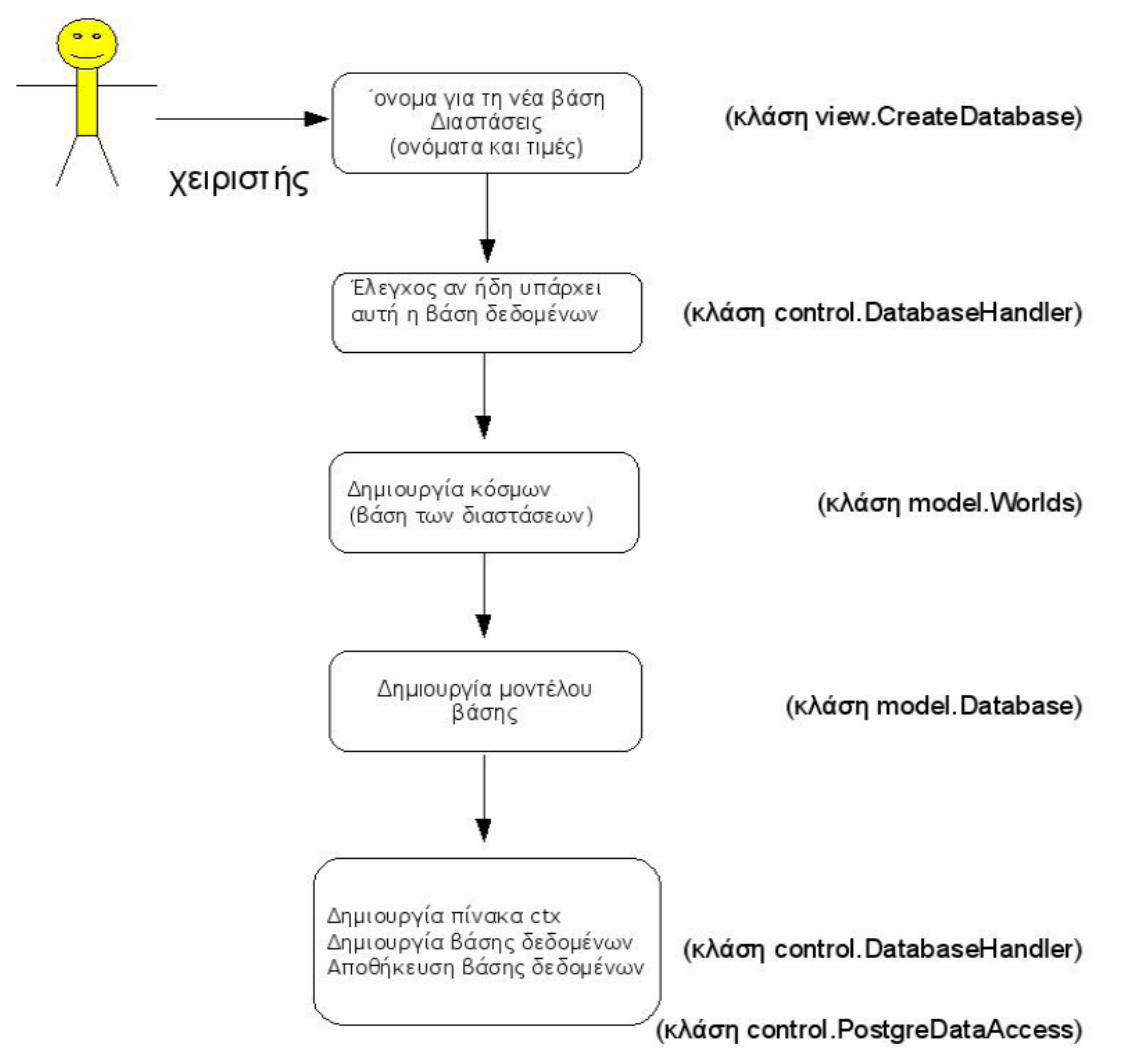
4.3.5 Test Framework Classes

Οι κλάσεις αυτές είναι τμήμα του testing framework που χρησιμοποιήθηκε κατά την ανάπτυξη της εφαρμογής προκειμένου να ελέγχεται διαρκώς η ορθότητα των παραγόμενων αλγορίθμων.

ControlCenterTestCase	Αυτή η κλάση χρησιμοποιήθηκε για τον έλεγχο πολλών συναρτήσεων της εφαρμογής, πολλές από τις οποίες μεταφέρθηκαν στη συνέχεια σε άλλες κλάσεις.
PostgreDataAccessTestCase	Αυτή η κλάση είναι υπεύθυνη κυρίως για την ορθότητα του παραγόμενου sql κώδικα από τη συνάρτηση PostgreDataAccess

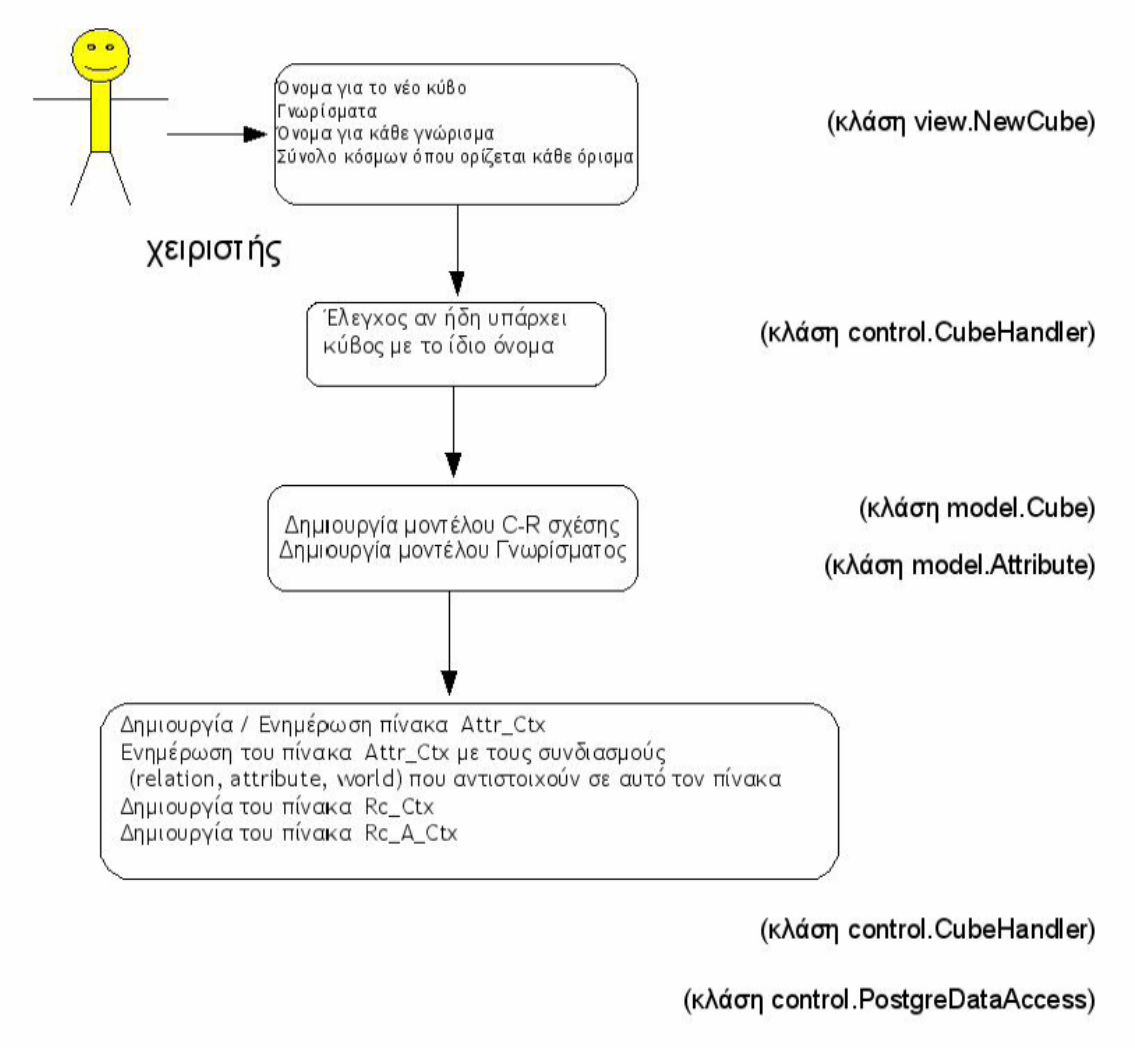
4.4 Σχηματική αναπαράσταση των λειτουργιών

4.4.1 Νέα βάση δεδομένων



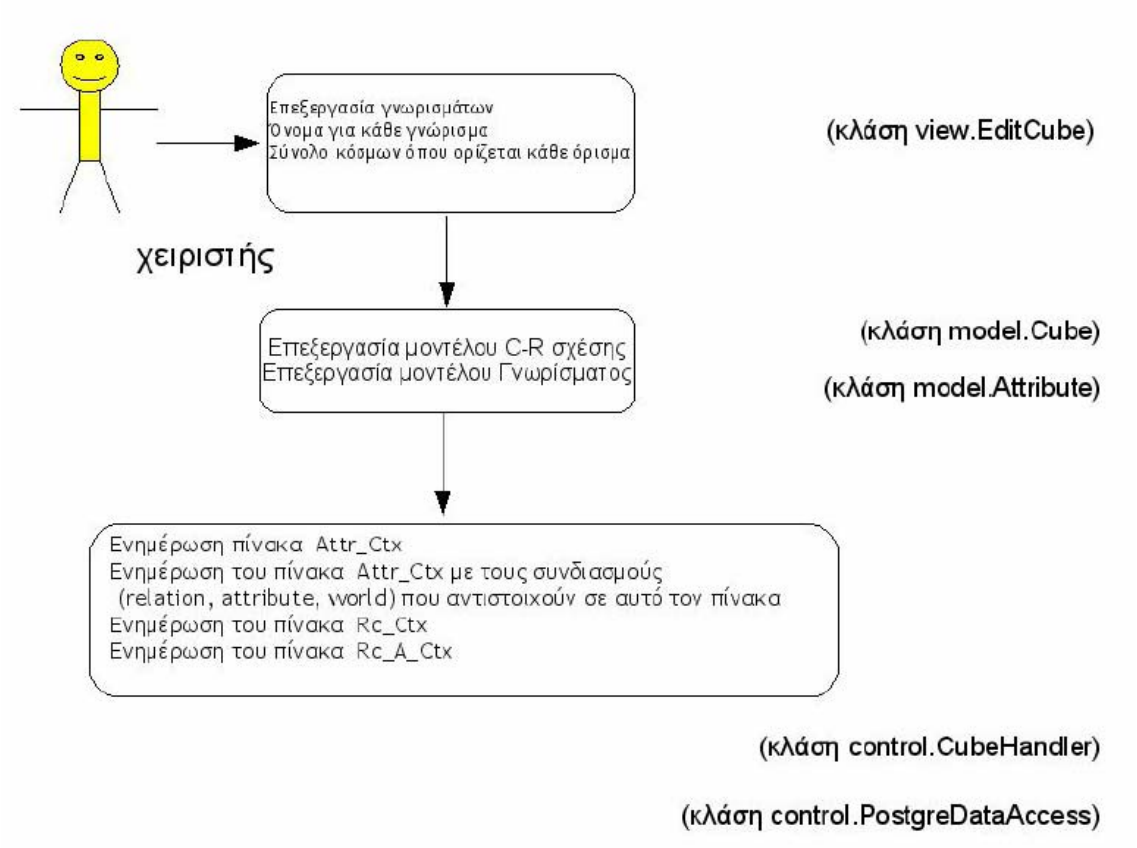
Εικόνα 4.2. Ορισμός μιας Context Relational βάσης Δεδομένων

4.4.2 Νέα C – R σχέση



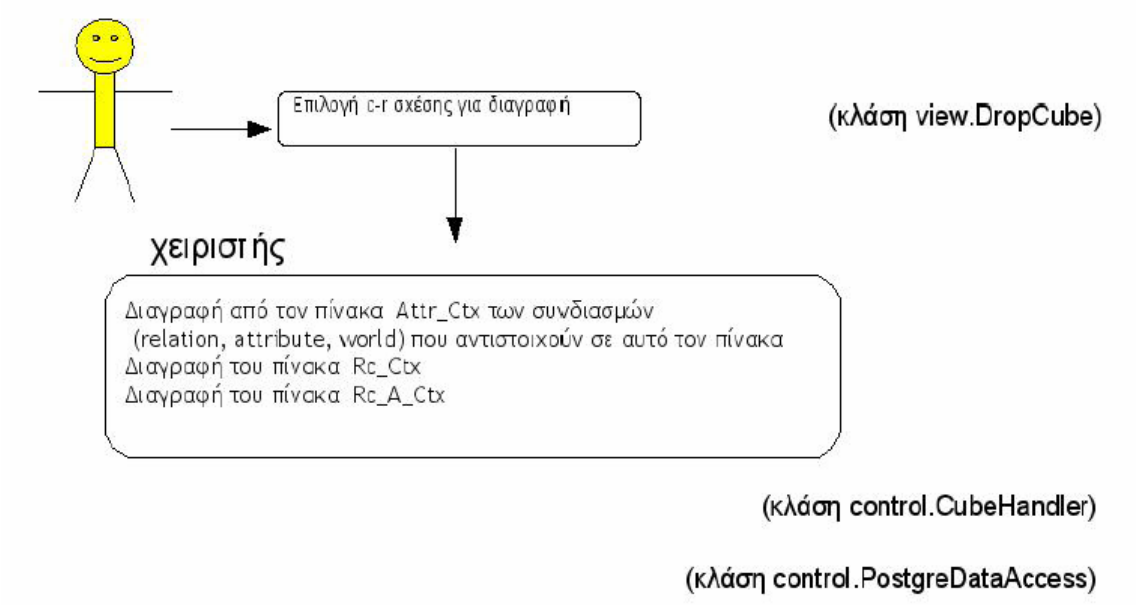
Εικόνα 4.3. Ορισμός μιας Context Relational σχέσης

4.4.3 Επεξεργασία Context Relational σχέσης



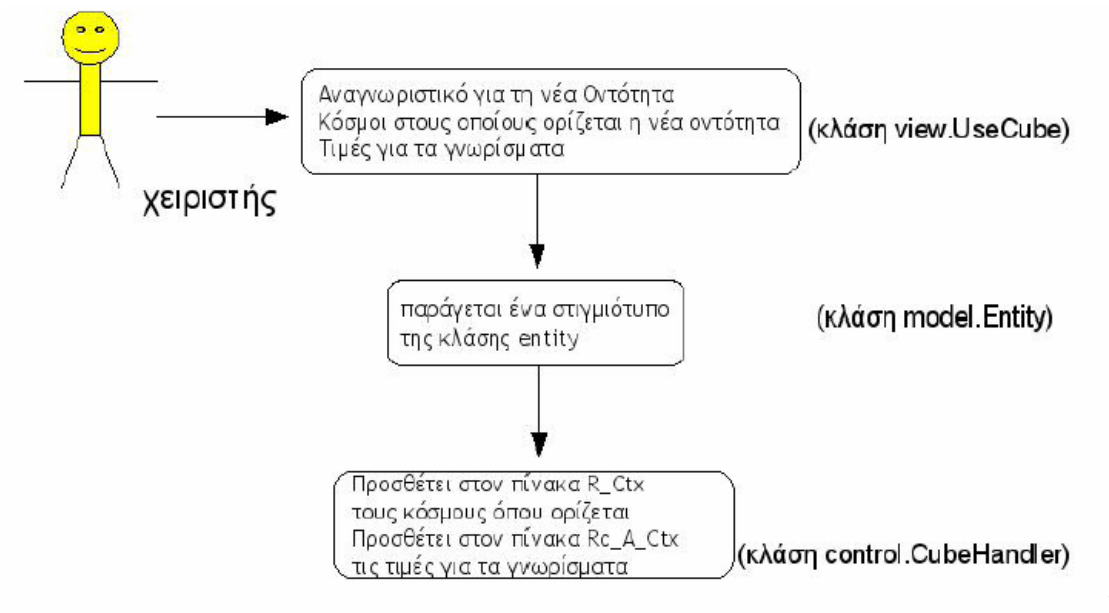
Εικόνα 4.4. Επεξεργασία μιας Context Relational σχέσης

4.4.4 Διαγραφή Context Relational σχέσης



Εικόνα 4.5. Διαγραφή μιας context Relational σχέσης

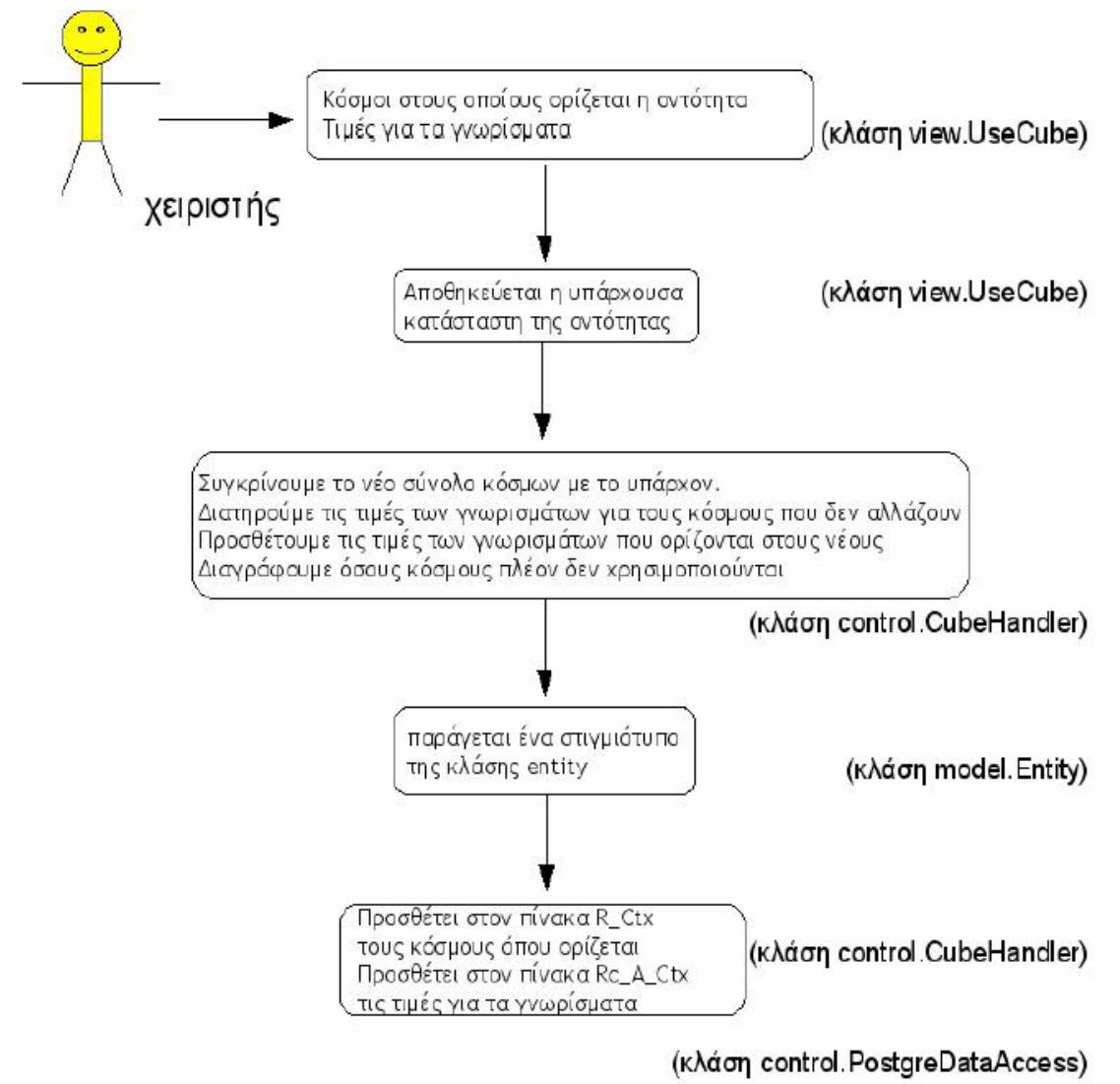
4.4.5 Προσθήκη οντότητας



Ει

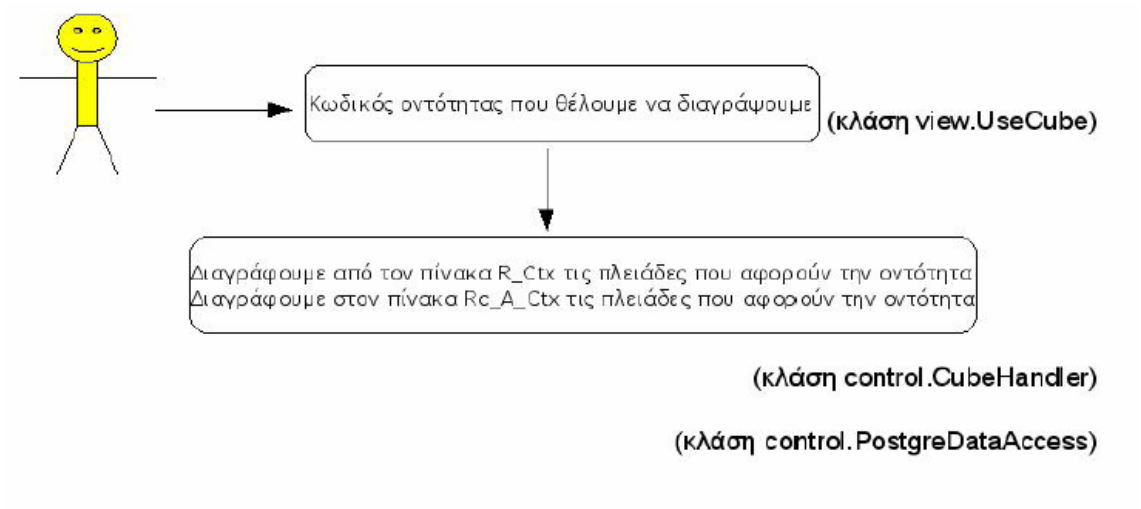
κόνα 4.6. Προσθήκη νέας οντότητας

4.4.6 Επεξεργασία οντότητα



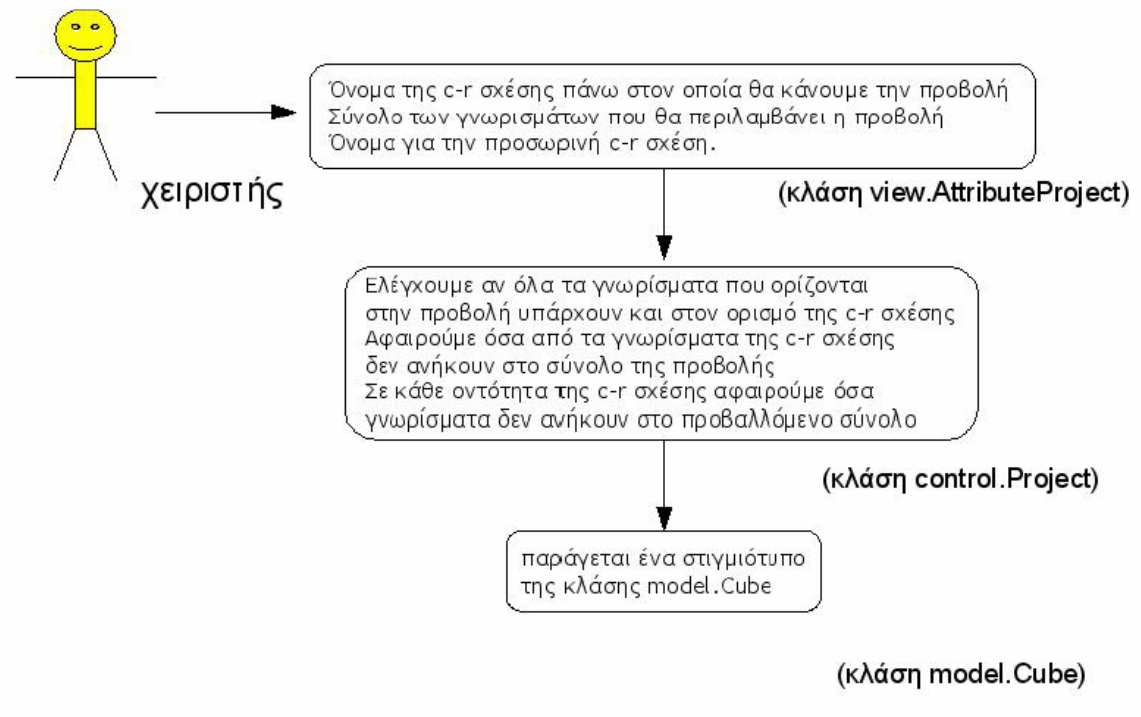
Εικόνα 4.7. Επεξεργασία οντότητας

4.4.7 Διαγραφή οντότητας



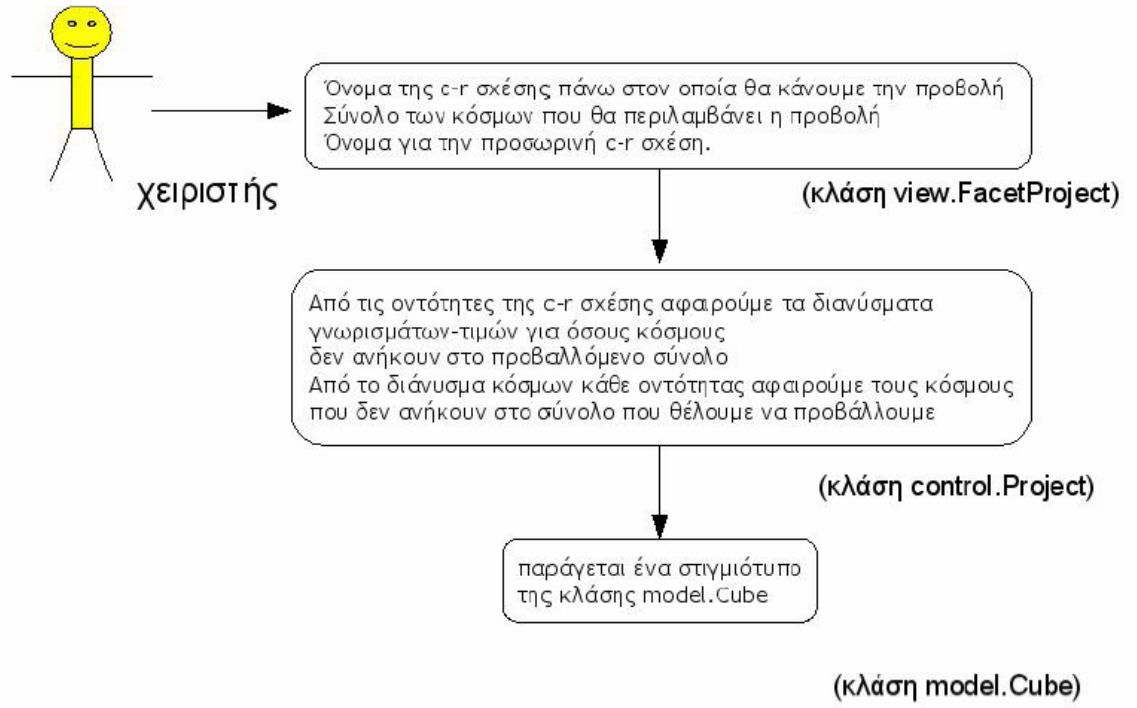
Εικόνα 4.7. Διαγραφή οντότητας

4.4.8 C-R πράξη "attribute project"



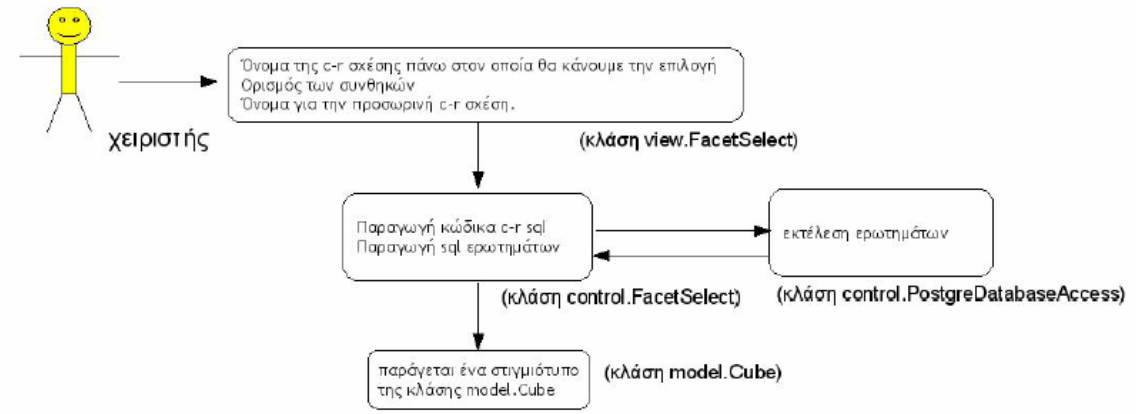
Εικόνα 4.8. Υλοποίηση της C-R πράξης attribute project

4.4.9 C-R πράξη "context project"



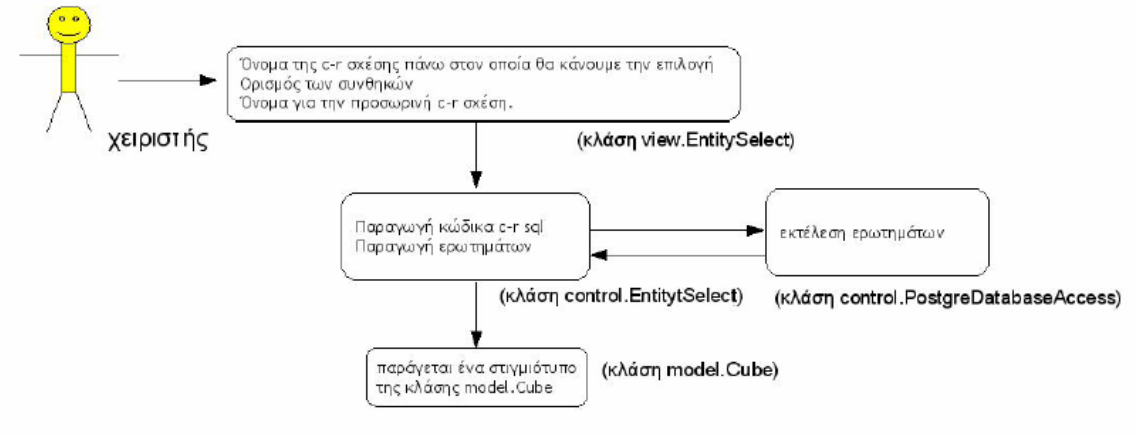
Εικόνα 4.9. Υλοποίηση της C-R πράξης context project

4.4.10 C-R πράξη "facet select"



Εικόνα 4.10. Υλοποίηση της C-R πράξης facet select

4.4.11 C-R πράξη "entity select"



Εικόνα 4.11. Υλοποίηση της C-R πράξης entity select

5

Έλεγχος

Για την ανάπτυξη και τον έλεγχο της εφαρμογής χρησιμοποιήθηκαν unit tests υλοποιημένα με το JUnit.

5.1 Μεθοδολογία Ελέγχου

5.1.1 Γενικά για τη χρήση unit tests.

Τα unit tests είναι μια μέθοδος με βάση την οποία για κάθε μονάδα ενός προγράμματος που έχει μια συγκεκριμένη λειτουργικότητα (στη δική μας περίπτωση για κάθε κλάση) αναπτύσσεται ένα σύνολο αυτόματων test τα οποία ελέγχουν κατά πόσο η συγκεκριμένη μονάδα πράγματι έχει την επιθυμητή λειτουργία. Η βασική ιδέα είναι ότι για μια συγκεκριμένη είσοδο πρέπει η λειτουργική μονάδα να επιστρέφει μια συγκεκριμένη έξοδο. Γράφοντας λοιπόν ένα πρόγραμμα το οποίο καλεί τη μονάδα που θέλουμε να ελέγξουμε παρέχοντας μια είσοδο και που στη συνέχεια ελέγχει την έξοδο ως προς ένα προκαθορισμένο αποτέλεσμα, μπορούμε να ελέγξουμε τη συμπεριφορά της λειτουργικής μονάδας.
Η χρήση unit tests, παρόλο που φαίνεται χρονοβόρα, παρέχει μια σειρά ιδιαίτερα σημαντικών πλεονεκτημάτων.

1. Εφόσον φτιάξουμε πρώτα τα test και στη συνέχεια αναπτύξουμε το πρόγραμμά μας, διαθέτουμε έναν πολύ καλό τρόπο να ελέγξουμε πως έχουμε τελειώσει τη συγγραφή του. Αρκεί το πρόγραμμα να επιστρέφει θετικό αποτέλεσμα σε όλες τις δοκιμές. Η συγγραφή των απαιτήσεων στην αρχή της ανάπτυξης, παρέχει έναν αρκετά πιο αξιόπιστο τρόπο ελέγχου από το να δοκιμάζουμε με εμπειρικό τρόπο το σύνολο της εφαρμογής και να προσπαθούμε να δούμε αν κάθε τμήμα της επιστρέφει τα σωστά αποτελέσματα.
2. Η ύπαρξη των test, απλοποιεί ιδιαίτερα τη διαδικασία του refactoring. Μπορούμε να κάνουμε οποιεσδήποτε αλλαγές σε οποιοδήποτε τμήμα του προγράμματος και να έχουμε έναν άμεσο τρόπο να ελέγξουμε το σύνολο της εφαρμογής για να επιβεβαιώσουμε πως δεν δημιουργείται κανένα πρόβλημα. Μετά από κάθε αλλαγή τρέχουμε το σύνολο των test και εφόσον έχουμε θετικό αποτέλεσμα σε όλα οι αλλαγές προστίθενται στον κώδικα, διαφορετικά επιστρέφουμε στην προηγούμενη σταθερή έκδοση.
3. Η παραπάνω ιδιότητα δίνει αυξημένη δυνατότητα για συγγραφή του κώδικα από ομάδες, αφού είναι πολύ εύκολο να ελεγχθεί η δυνατότητα ενσωμάτωσης των αλλαγών που προτείνει κάθε μέλος.
4. Επίσης, παρέχεται αυξημένη ευελιξία στην προσαρμογή της εφαρμογής στην μεταβολή των προδιαγραφών που μπορεί να προκύψει. Αιτήσεις για τέτοιες αλλαγές είναι ιδιαίτερα συχνές από τη στιγμή που μια εφαρμογή αρχίζει να χρησιμοποιείται και η βασική εναλλακτική σε περίπτωση που είναι δύσκολο να γίνει refactoring είναι η εξαντλητική σχεδίαση πριν την ανάπτυξη, κάτι που δεν είναι πάντα εύκολο να γίνει.

Ένα ιδιαίτερο χαρακτηριστικό των unit tests που συναντάται και στην εφαρμογή μας είναι πως υπάρχει μια δυσκολία να ενσωματωθούν σε εφαρμογές βάσεων δεδομένων.

5.1.2 Γενικά για το JUnit.

Παρόλο που τα unit tests μπορούν να συγγραφούν εξ ολοκλήρου εκ του μηδενός, η χρήση ενός framework μπορεί να συμβάλλει αρκετά στη μείωση του χρόνου ανάπτυξης και στην ευκολία χρήσης. Το JUnit είναι ένα framework για unit testing σε Java το οποίο διακρίνεται για την ευκολία στη χρήση του. Μπορεί να ενσωματωθεί στην ανάπτυξη της εφαρμογής με διάφορους τρόπους, πχ σε συνδιασμό με το ant. Υπάρχει επίσης σε μορφή plugin για το περιβάλλον προγραμματισμού eclipse, το οποίο και χρησιμοποιήσαμε.

Για τη δημιουργία της εφαρμογής μας, χρησιμοποιήσαμε test cases για δυο κυρίως ομάδες κλάσεων της εφαρμογής μας. Για τις κλάσεις που επικοινωνούν με τον database server, ώστε να ελέγξουμε αν οι συναρτήσεις παράγουν τον επιθυμητό sql κώδικα με βάση τις εισόδους και για τις κλάσεις που υλοποιούν τη βασική application logic.

Τέλος, αν και είναι εφικτό, δεν κρίθηκε αναγκαίο να χρησιμοποιηθούν test cases για τον έλεγχο του user interface.

5.2 Αναλυτική παρουσίαση έλεγχου

Σε αυτή την ενότητα, με τη βοήθεια ενός εκτεταμένου παραδείγματος, παρουσιάζεται η λειτουργία της εφαρμογής συνοδευόμενη από τις απαραίτητες επεξηγήσεις για τη λειτουργία τόσο του μοντέλου όσο και της εφαρμογής. Η παρουσίαση εμπλουτίστηκε από τα κατάλληλα screen-dumps. Το παρόν δύναται να χρησιμεύσει και σαν στοιχειώδης οδηγός χρήσης της εφαρμογής.

5.2.1 Επεξήγηση Παραδείγματος

Θεωρείστε μια ιστοσελίδα σχετική με ψηφιακές φωτογραφικές μηχανές. Οι πληροφορίες που παρέχονται για κάθε φωτογραφική μηχανή περιλαμβάνουν το εμπορικό σήμα, το μοντέλο της φωτογραφικής μηχανής, μια εικόνα, το μέγεθος σε megapixels και την τιμή. Οι πελάτες συνδέονται με αυτή την ιστοσελίδα χρησιμοποιώντας ποικίλες συσκευές που κυμαίνονται σε υπολογιστή γραφείου (PC), PDA και κινητό τηλέφωνο. Επιπλέον, μπορούν να επιλέξουν τη μέθοδο πληρωμής μεταξύ πιστωτικής κάρτας και μετρητών.

Τα δεδομένα για μια ψηφιακή φωτογραφική μηχανή που επιστρέφονται στους πελάτες εξαρτώνται από τη συσκευή πρόσβασης και τη μέθοδο πληρωμής. Δηλαδή ένας πελάτης χρησιμοποιώντας ένα PDA λαμβάνει μια εικόνα μικρότερης ανάλυσης από όταν χρησιμοποιεί έναν υπολογιστή γραφείου. Όταν χρησιμοποιείται ένα κινητό τηλέφωνο δεν υπάρχει καμιά εικόνα και παρέχονται μόνο πληροφορίες κειμένου. Επιπλέον, η τιμή μιας ψηφιακής φωτογραφικής μηχανής ποικίλλει σύμφωνα με τη μέθοδο πληρωμής. Προφανώς, η ψηφιακή φωτογραφική μηχανή είναι μια οντότητα πληροφοριών της οποίας το περιεχόμενο και η δομή σε αυτό το ιδιαίτερο παράδειγμα ποικίλλει σύμφωνα με τη συσκευή πρόσβασης και τη μέθοδο πληρωμής.

Σε όρους βάσεων δεδομένων υπάρχει μια κύρια σχέση cameras, με τις ιδιότητες: Brand, Model, MPix, Photo, Price (βλ. πίνακα 2). Τα αιτήματα των πελατών εκφράζονται ως επερωτήσεις και τα δεδομένα που επιστρέφονται αντιστοιχούν στην αξιολόγηση αυτών των επερωτήσεων πάνω σε αυτή την σχέση. Το context της σχέσης του παραδείγματος εκφράζεται μέσω των διαστάσεων device που κυμαίνεται σε {PC, PDA, CELL} και payment που κυμαίνεται σε {Credit Card, Cash}

Το σχήμα της σχέσης cameras και τα δεδομένα για μια ιδιαίτερη οντότητα μπορούν να διαφέρουν μεταξύ διαφορετικών κόσμων. Αυτό οφείλεται στο γεγονός ότι μια ιδιότητα μπορεί να μην υπάρχει σε μερικούς κόσμους και ότι η ίδια ιδιότητα μπορεί να έχει διαφορετικές τιμές κάτω από διαφορετικούς κόσμους. Ο πίνακας 5.1 παρουσιάζει όλους τους πιθανούς κόσμους, ενώ ο πίνακας 5.2 παρουσιάζει τους κόσμους κάτω από τους οποίους κάθε ιδιότητα καθορίζεται.

Κόσμος	**Device**	**Payment**
w0	cell	Cash
w1	cell	Credit Card
w2	pc	Cash
w3	pc	Credit Card
w4	pda	Cash
w5	pda	Credit Card

Πίνακας 1. Παρουσιάζονται όλοι οι πιθανοί κόσμοι

Ιδιότητες	**Κόσμοι**
brand	ορισμένη σε όλους τους κόσμους
model	ορισμένη σε όλους τους κόσμους
mpix	ορισμένη σε όλους τους κόσμους
photo	ορισμένη στους κόσμους με συσκευή πρόσβασης PC ή PDA
price	ορισμένη σε όλους τους κόσμους, αλλά η τιμή της μπορεί να αλλάζει

Πίνακας 2 Παρουσιάζονται οι κόσμοι κάτω από τους οποίους ορίζεται η κάθε ιδιότητα για τη σχέση cameras

Η βάση περιέχει μία επιπλέον σχέση την memories για την οποία ισχύουν οι ίδιες υποθέσεις. Ο αντίστοιχος πίνακάς της είναι:

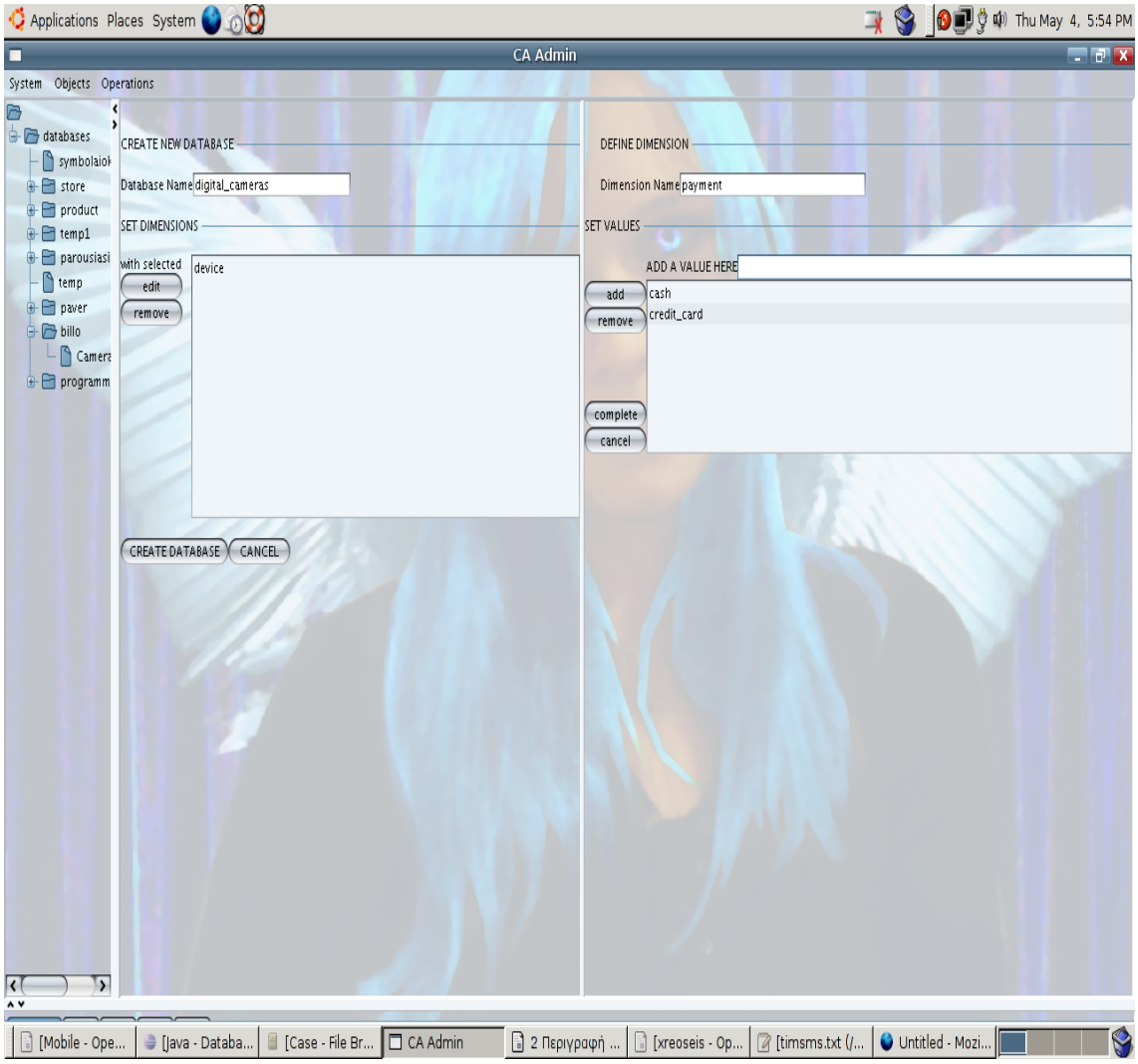
Ιδιότητες	**Κόσμοι**
brand	ορισμένη σε όλους τους κόσμους
type	ορισμένη σε όλους τους κόσμους
mb	ορισμένη σε όλους τους κόσμους
price	ορισμένη σε όλους τους κόσμους, αλλά η τιμή της μπορεί να αλλάζει

Πίνακας 5.3 Παρουσιάζονται οι κόσμοι κάτω από τους οποίους ορίζεται η κάθε ιδιότητα για τη σχέση memories

5.2.2 Δημιουργία της CR βάσης και των CR σχέσεών της

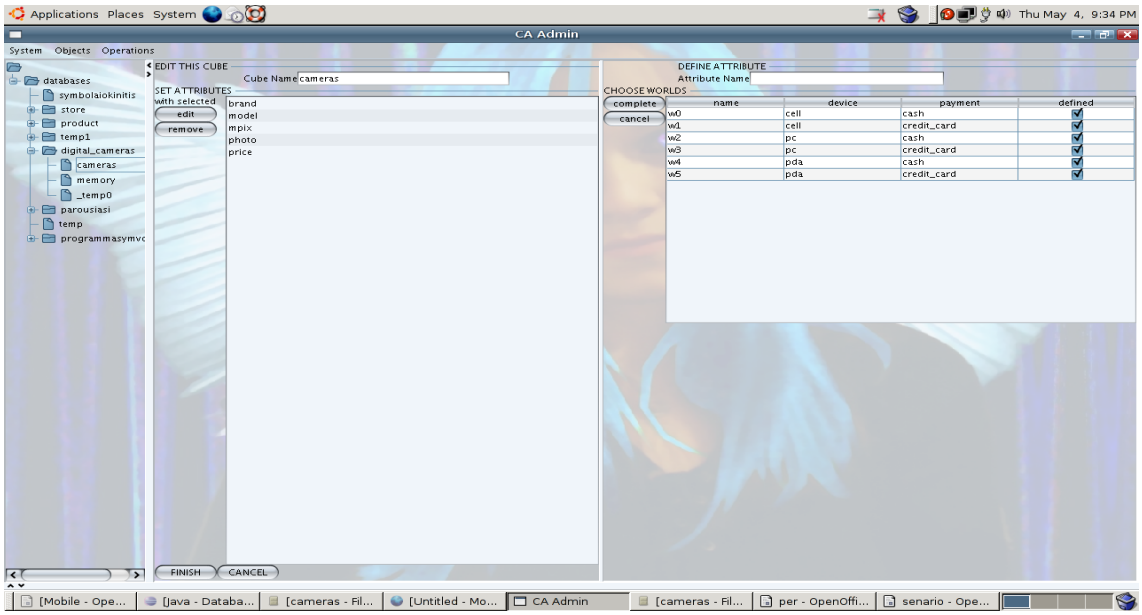
5.2.2.1 Create new db

Στο μενού Objects επιλέγουμε την επιλογή Create Database (ή εναλλακτικά με δεξί κλικ πάνω στο φάκελο databases του δέντρου στα αριστερά). Ανοίγει το παράθυρο όπου εισάγεται αρχικά το όνομα της βάσης στο πεδίο Database Name.
Στο μενού Objects επιλέγουμε το Create Database (ή εναλλακτικά με δεξί κλικ πάνω στο φάκελο databases του δέντρου στα αριστερά) και ανοίγει το παράθυρο όπου εισάγεται το όνομα της βάσης και ορίζονται οι διαστάσεις (όνομα, τιμές) στο δεξί τμήμα του παραθύρου. Μόλις είμαστε έτοιμοι κάνουμε κλικ στο CREATE DATABASE.



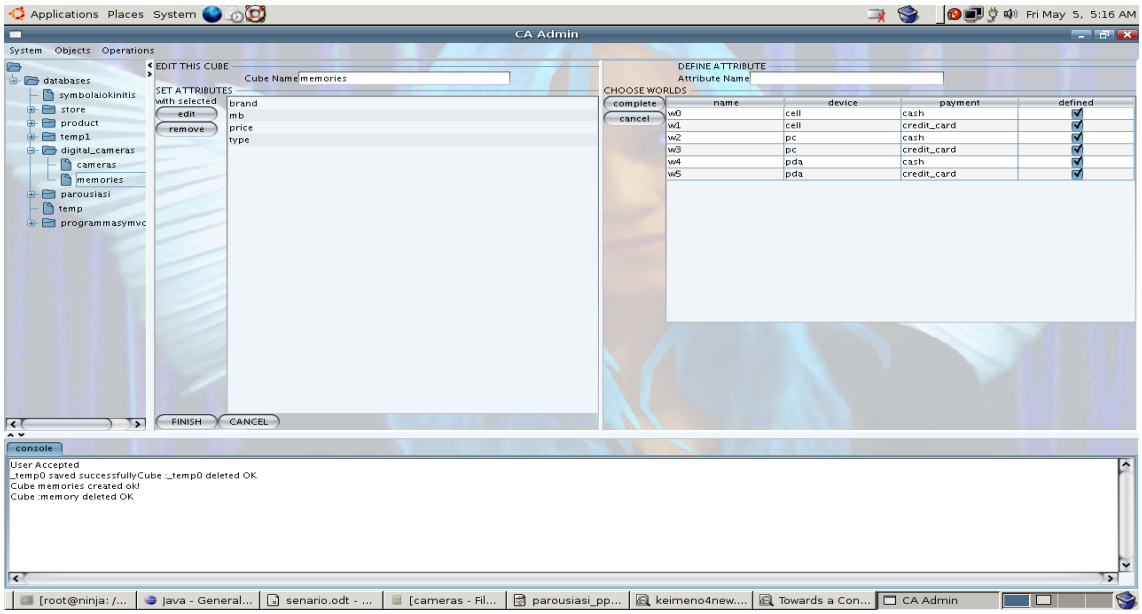
Εικόνα 5.1 Στιγμιότυπο από το στάδιο δημιουργίας νέας βάσης

5.2.2.2 Create new cube



Εικόνα 5.2 Στιγμιότυπο από το στάδιο δημιουργίας νέας σχέσης

Το επόμενο βήμα είναι να δημιουργήσουμε τις Context Σχέσεις της βάσης μας.. Επιλέγουμε το φάκελο με τη βάσης, και από το μενού Objects την επιλογή New Cube. Ανοίγει το παράθυρο, στο οποίο ορίζουμε την Context σχέση cameras δηλώνοντας το όνομά της, καθορίζοντας τις ιδιότητες που περιέχει και σε ποιους κόσμους ορίζεται η καθεμιά. Το ίδιο κάνουμε και για την memories. Παρέχεται η δυνατότητα αφού οριστεί μια σχέση, να επεξεργαστεί αργότερα η δομή της επιλέγοντας από το μενού Objects την επιλογή Edit Cube.

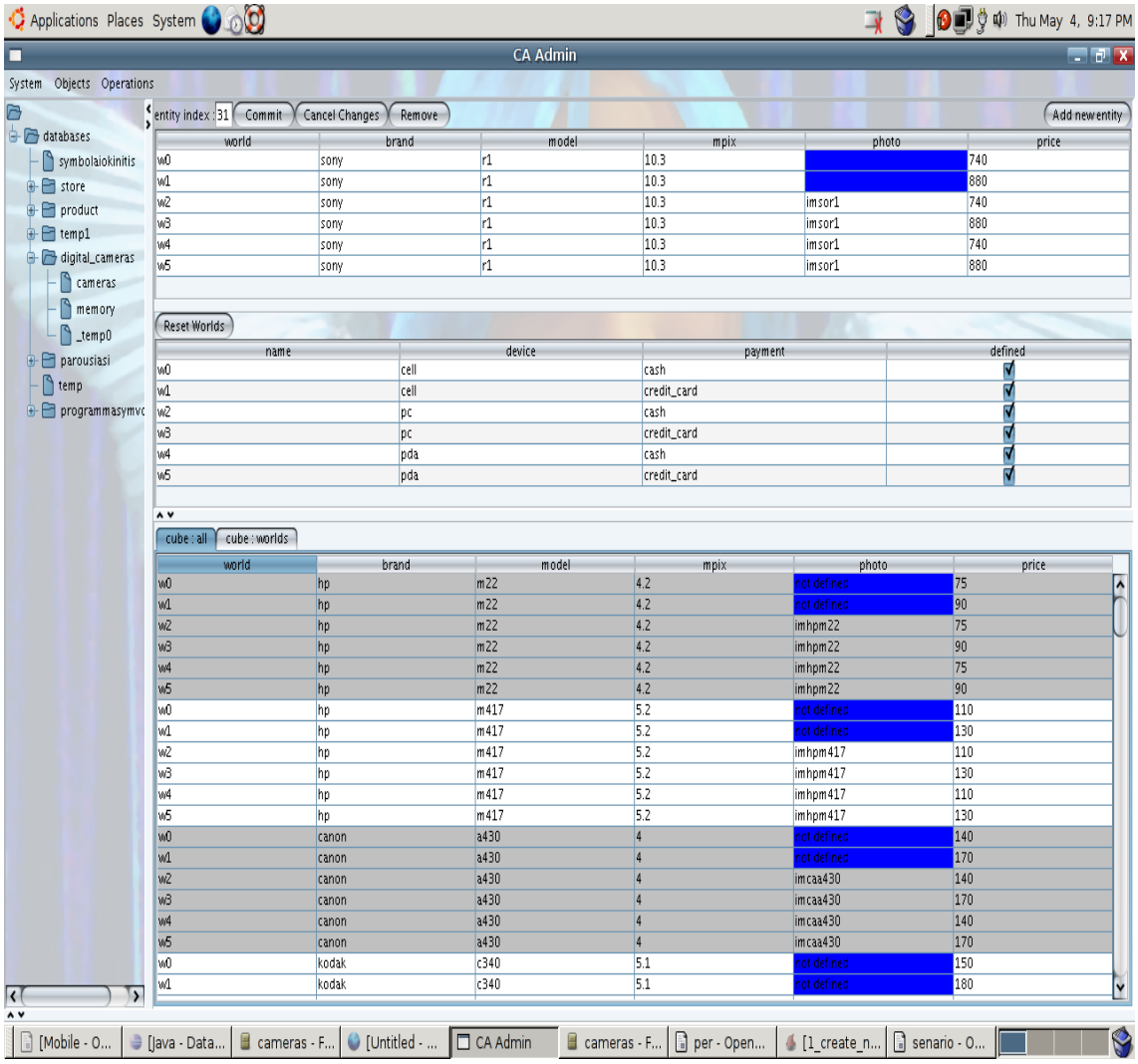


Εικόνα 5.3 Στιγμιότυπο από το στάδιο δημιουργίας της σχέσης memories

5.2.3 Data entry

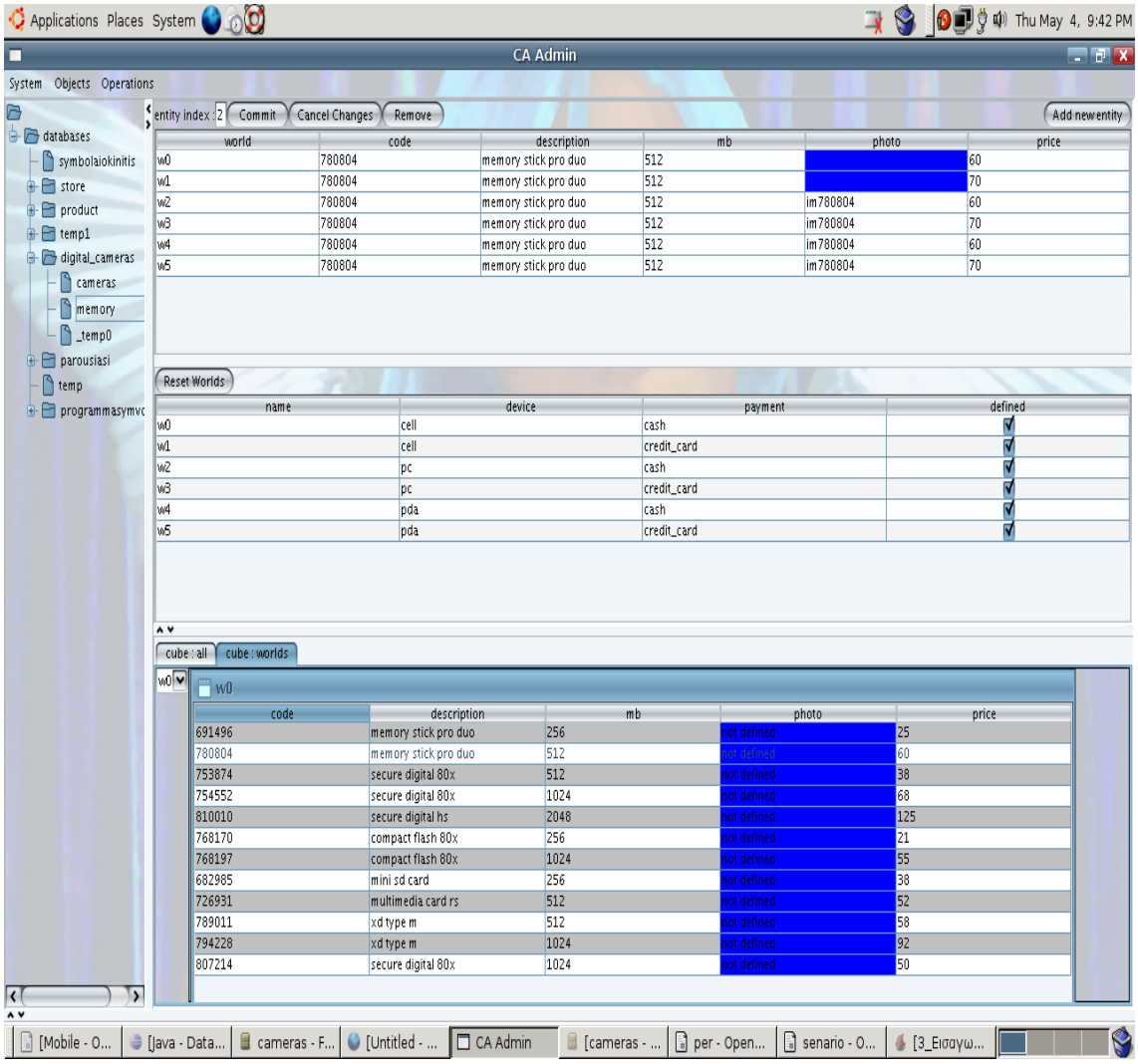
Εμφανίζουμε τα περιεχόμενα του φακέλου της βάσης μας, που είναι οι αποθηκευμένες Context Σχέσεις. Έχοντας επιλεγμένη αυτή που επιθυμούμε, από το μενού Object επιλέγουμε use. Εμφανίζεται το παράθυρο που ακολουθεί στο οποίο εισάγουμε τα δεδομένα μιας οντότητας στα κατάλληλα σημεία σύμφωνα με την ιδιότητα και τον κόσμο (στο πάνω κομμάτι του παραθύρου). Στο μεσαίο κομμάτι μπορούμε να καθορίσουμε σε ποιους κόσμους υφίσταται μια συγκεκριμένη οντότητα.

Στο κάτω κομμάτι εμφανίζονται τα περιεχόμενα της σχέσης. Επιλέγοντας μια οντότητα, εμφανίζονται τα στοιχεία της στο πάνω τμήμα απ' όπου μπορούμε να τα ενημερώσουμε. Τα κελιά που είναι καλυμένα με μπλε χρώμα έχουν την έννοια ότι δεν ορίζεται η ιδιότητα στο(υς) δεδομένο(υς) κόσμο(υς).



Εικόνα 5.4 Στιγμιότυπο από το στάδιο της καταχώρησης τιμών

Τέλος στο κάτω κομμάτι υπάρχει η δυνατότητα να παρουσιαστούν με διαφορετικό τρόπο τα περιεχόμενα της σχέσης επιλέγοντας το tab cube: worlds, αντί του προεπιλεγμένου cube: all. Η διαφορά έγκειται, όπως φαίνεται στο σχήμα που ακολουθεί και που αφορά πλέον τη σχέση memory, ότι δεν παρουσιάζονται όλες οι δυνατές εκφάνσεις των οντοτήτων, άλλα μία συγκεκριμένη για κάθε οντότητα σύμφωνα με τον κόσμο που επιλέγουμε.



Εικόνα 5.5 Στιγμιότυπο εναλλακτικής εμφάνισης των τιμών μιας σχέσης

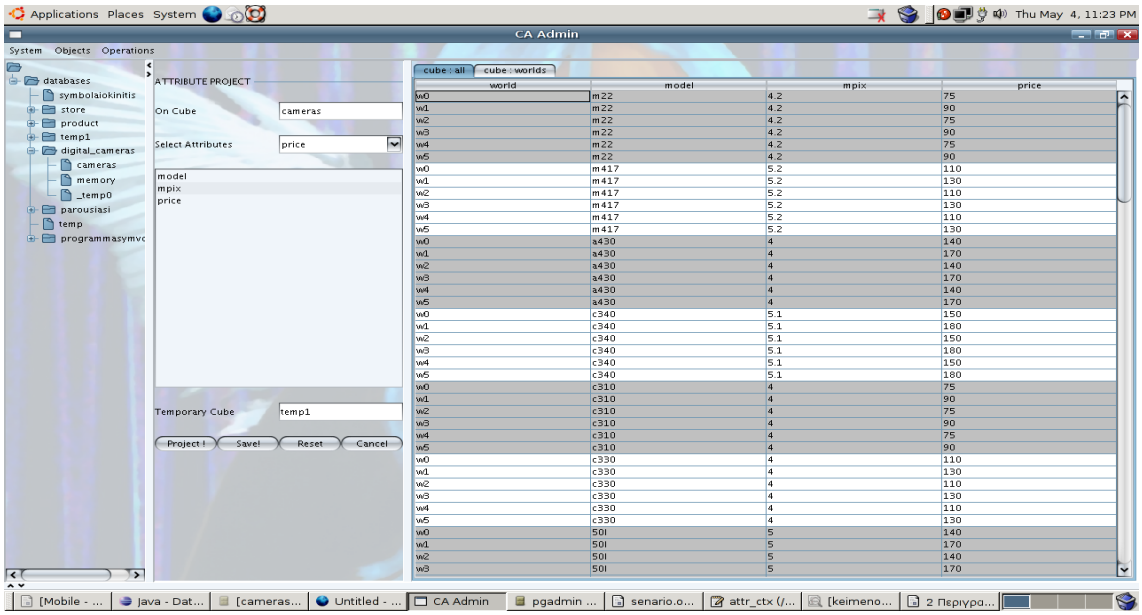
Η δυνατότητα προβολής με τους δύο εναλλακτικούς τρόπους είναι διαθέσιμη ανά πάσα στιγμή στον χρήστη. Είτε κατά το στάδιο εισαγωγής – ενημέρωσης μιας σχέσης, είτε αφού λάβει στην οθόνη του αναφορές που έχουν σχέση με επερωτήσεις που έχουν υποβληθεί.

5.2.4 Εκτέλεση λειτουργιών στις CR σχέσεις

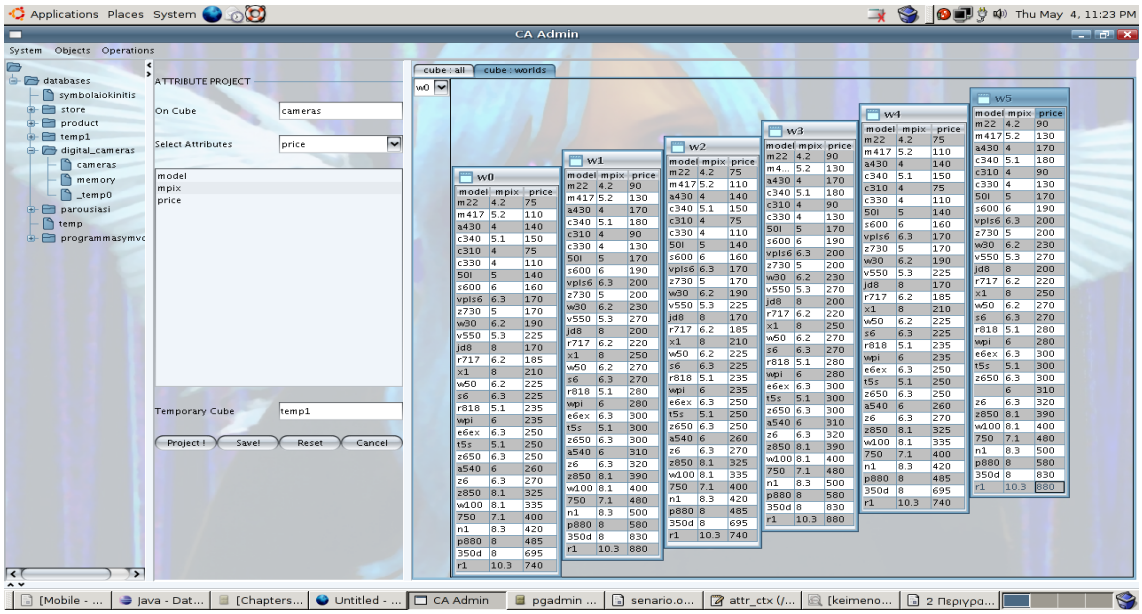
5.2.4.1 Attribute Project

Δοσμένης μιας εισαχθείσας context-σχέσης, ο στόχος είναι να παράγουμε μόνο τις ιδιότητες που διευκρινίζονται στη συνθήκη του τελεστή της attribute-προβολής. Το αποτέλεσμα είναι μια νέα context-σχέση που έχει μόνο τις κάθετες φέτες που αντιστοιχούν στις προβαλλόμενες ιδιότητες.

Στο μενού Operations επιλέγουμε Attribute Project και στο παράθυρο που ανοίγει επιλέγουμε τη context σχέση που αναφερόμαστε και τις ιδιότητες που μας ενδιαφέρουν. Στα σχήματα που ακολουθούν βλέπουμε το αποτέλεσμα της προβολής των ιδιοτήτων Model, MPix και Price: πModel[], MPix[], Price[] cameras



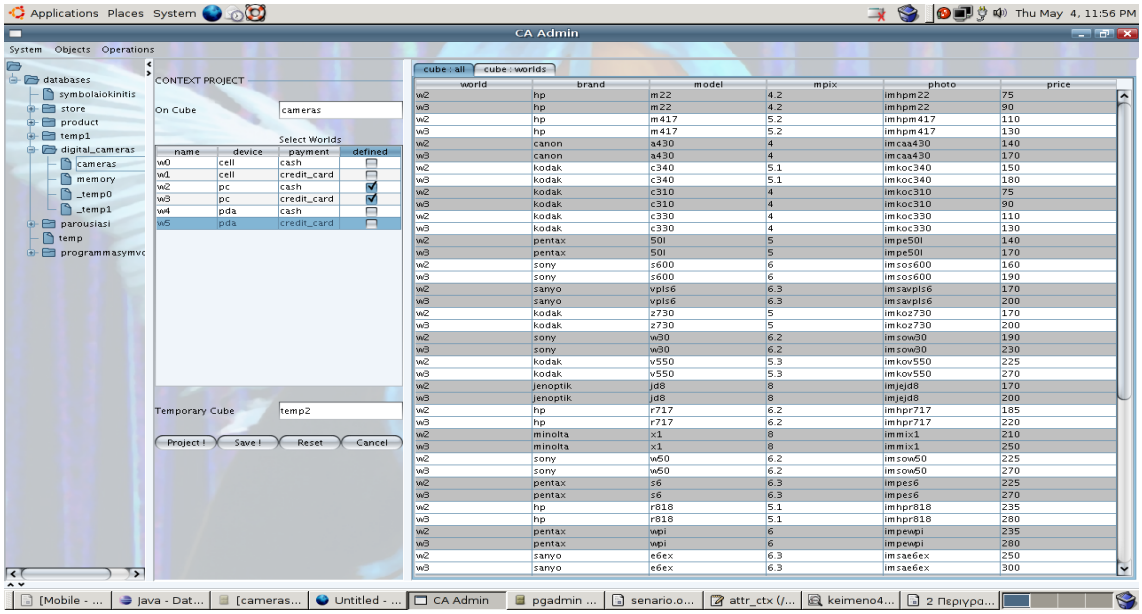
Εικόνα 5.6 Προβολή των ιδιοτήτων Model, MPix και Price



Εικόνα 5.6 Προβολή των ιδιοτήτων Model, MPix και Price, ειδομένα ανά κόσμο.

5.2.4.2 *Context Project*

Αυτή η λειτουργία διατηρεί μόνο εκείνες τις εκφάνσεις των οντοτήτων που κρατούνται κάτω από διευκρινισμένους κόσμους. Για παράδειγμα, για έναν πελάτη που χρησιμοποιεί PC οι σχετικοί κόσμοι είναι οι w2 και w3. Στα παρακάτω σχήματα παρουσιάζουμε το αποτέλεσμα της context προβολής των κόσμων w2 και w3 για την context-σχέση cameras.



Εικόνα 5.7 Προβολή των ιδιοτήτων Model, MPix και Price, ειδομένα ανά κόσμο.

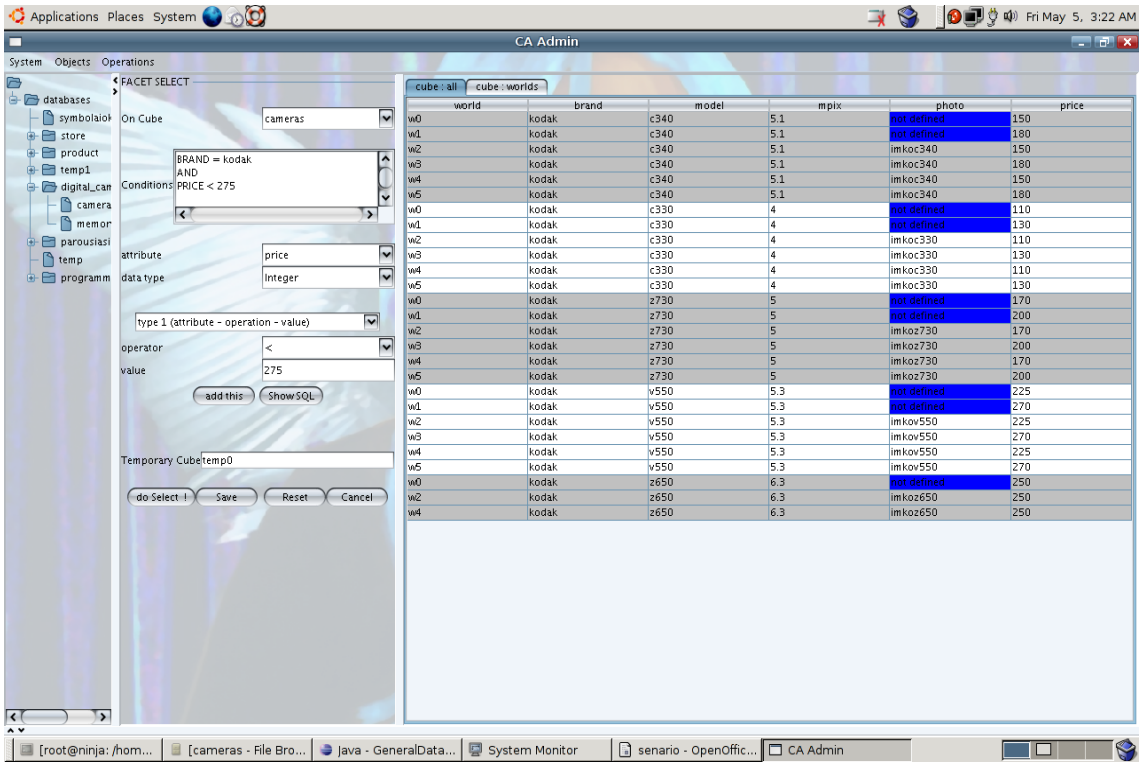
Χρησιμοποιούμε το γράμμα κ για να αντιπροσωπεύσουμε τον τελεστή της προβολής-κόσμου:

$$\kappa_{[device = pc]} cameras$$

5.2.4.3 *Select Facet*

Οι χρήστες είναι σε θέση να θέσουν επερωτήσεις που περιλαμβάνουν επιλογή των εκφάνσεων ή, με άλλα λόγια, να επιλέξουν μέρη μιας οντότητας. Αυτό μπορεί να γίνει χρησιμοποιώντας τον select facet τελεστή, που αναφέρεται ως σfacet.

$$\sigma^{facet}_{brand\ []= 'kodak'\_AND\_price\ \ 275}\ cameras$$



Εικόνα 5.8 Επιλογή εκφάνσεων με brand = kodak και με price λιγότερο από 275 euro.

Επιπλέον είναι δυνατή η διατύπωση cross-world επερωτήσεων που συγκρίνουν τις τιμές των ίδιων ιδιοτήτων σε διαφορετικούς κόσμους. Όπως για παράδειγμα αυτή που ζητά τις φωτογραφικές μηχανές που έχουν χαμηλότερη τιμή εάν ο πελάτης πληρώνει σε μετρητά από όταν χρησιμοποιεί μια πιστωτική κάρτα (κάτι που ισχύει de facto στο δεδομένο παράδειγμα γι' αυτό και δεν παρουσιάζουμε το αποτέλεσμα :

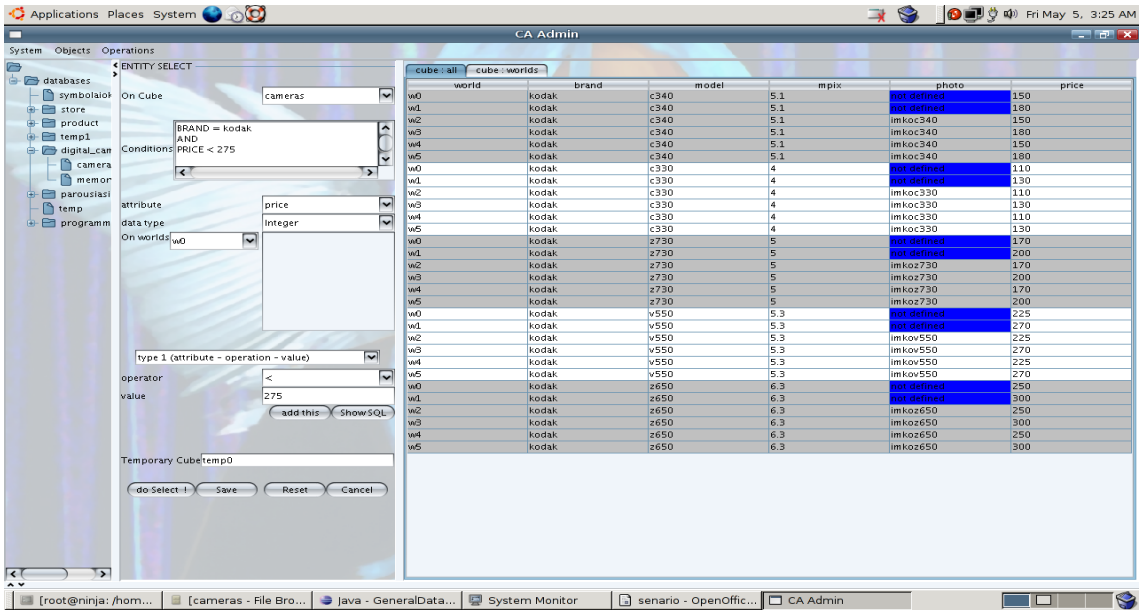
$$\sigma^{facet}_{\text{pr}\,ice[payment = cash] \quad \text{price}[payment = creditcard]}\ cameras$$

5.2.4.4 Select Entity

Κάθε ιδιότητα στην συνθήκη της context επιλογής οντότητας αξιολογείται στο σύνολο των κόσμων που υποδεικνύεται από τον αντίστοιχο context specifier.

Σε περίπτωση που δεν υπάρχει context specifier για μια ιδιότητα, αξιολογούμε την συνθήκη σε αληθινή εάν υπάρχει τουλάχιστον μια έκφανση που επαληθεύει τη συνθήκη.

$$\sigma^{entity}_{brand = 'kodak'\_AND\_\text{pr}\,ice \quad 275}\ cameras$$

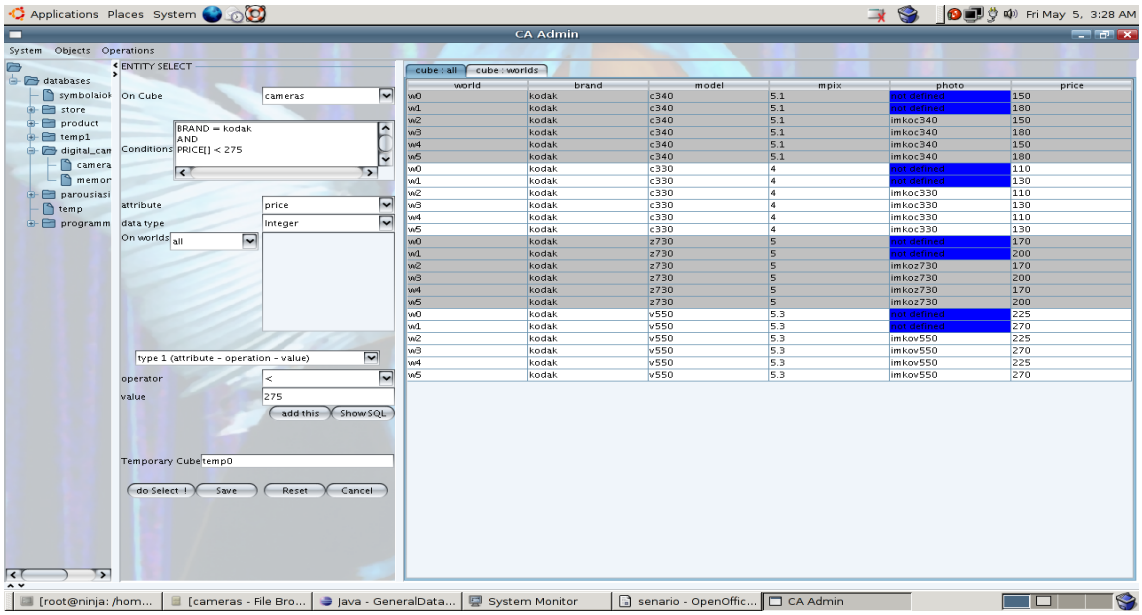


Εικόνα 5.9 Επιλογή οντοτήτων με brand = kodak και με price λιγότερο από 275 euro.

Σε περίπτωση που υπάρχει context specifier, αν είναι ο καθολικός τότε η συνθήκη πρέπει να επαληθεύεται σε όλους τους κόσμους, ενώ αν προσδιορίζει καθορισμένους κόσμους, η συνθήκη πρέπει να επαληθεύεται σε σύγκριση με αυτούς. Οι εκφράσεις που ακολουθούν επιστρέφουν το ίδιο αποτέλεσμα.

$$\sigma^{entity}_{brand = 'kodak'\_AND\_price\ []\ \ 275}\ cameras$$

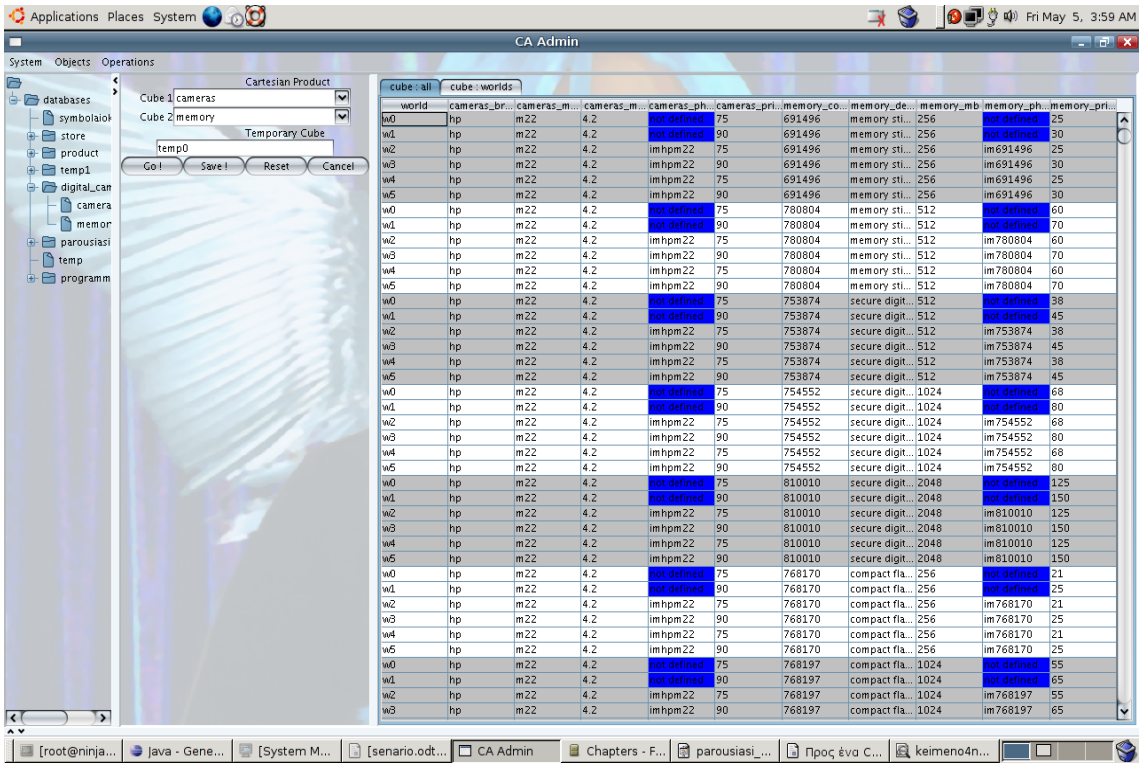
$$\sigma^{entity}_{brand = 'kodak'\_AND\_price\ [device=cell,payment=credit\_card]275}\ cameras$$



Εικόνα 5.10 Επιλογή οντοτήτων με χρήση context specifiers.

5.2.4.5 *Cartesian Product*

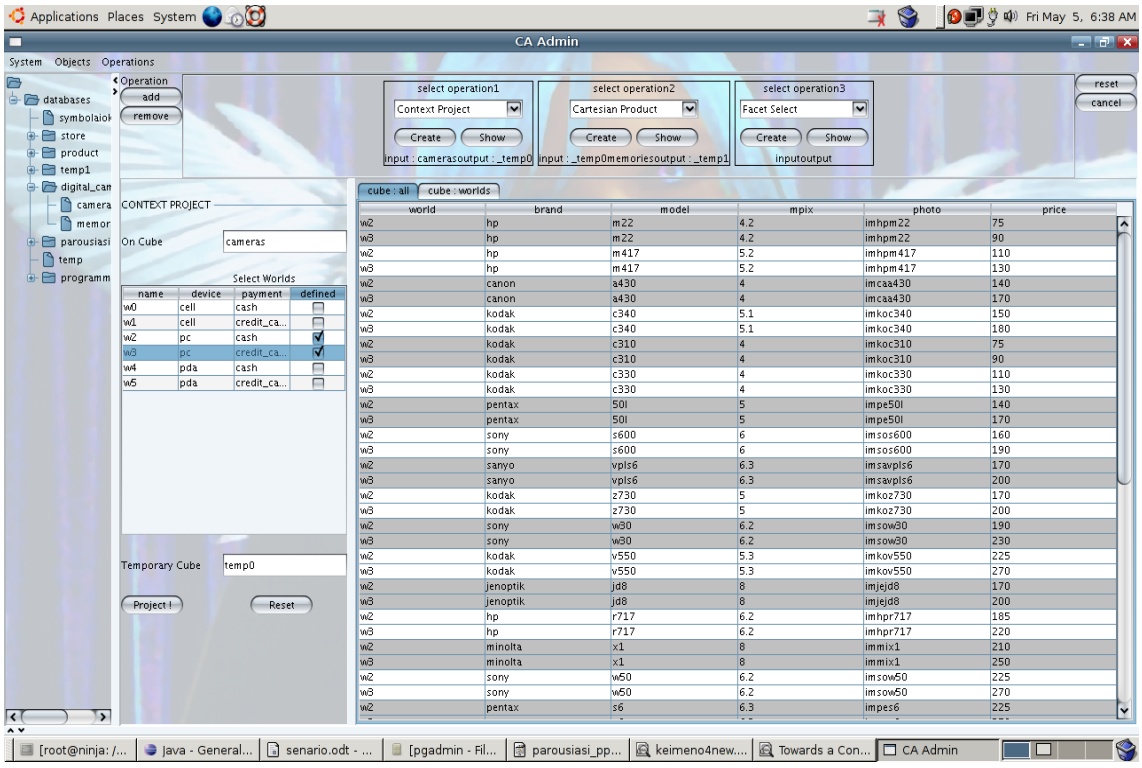
Η πράξη του καρτεσιανού γινόμενου στη c-r άλγεβρα είναι αντίστοιχη του καρτεσιανού γινόμενου της σχεσιακής άλγεβρας. Αρκεί να σημειωθεί πως ο συνδυασμός των σχέσεων γίνεται ανάμεσα στα tupples που ανήκουν στον ίδιο κόσμο. Για λόγους απλότητας μπορεί κανείς να θεωρήσει πως κάθε c-r σχέση αποσυντίθεται σε ένα σύνολο σχέσεων του σχεσιακού μοντέλου, μια για κάθε κόσμο. Στη συνέχεια παίρνουμε τα καρτεσιανά γινόμενα ανά δυο των σχέσεων που αντιστοιχούν στον ίδιο κόσμο και τέλος επανενώνουμε τα επιμέρους γινόμενα για να καταλήξουμε στο τελικό αποτέλεσμα για το c-r καρτεσιανό γινόμενο.



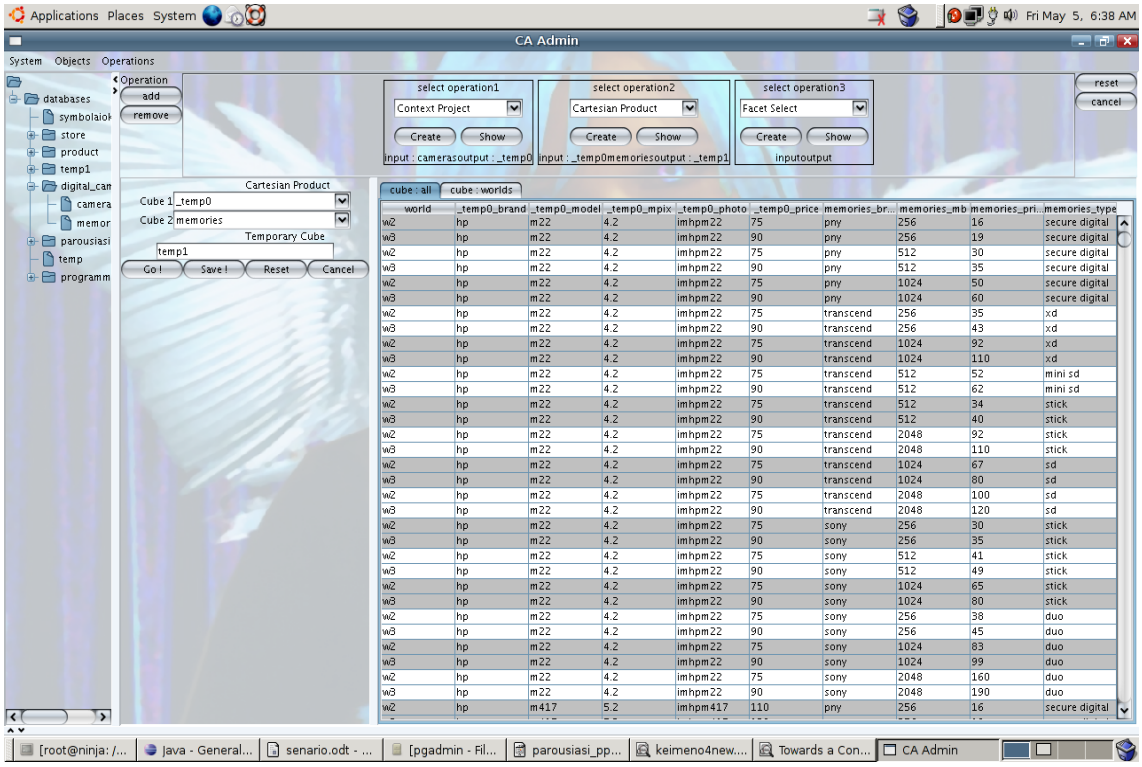
Εικόνα 5.11 Καρτεσιανό γινόμενο μεταξύ των context σχέσεων cameras και memories

5.2.4.6 *Combined Operations*

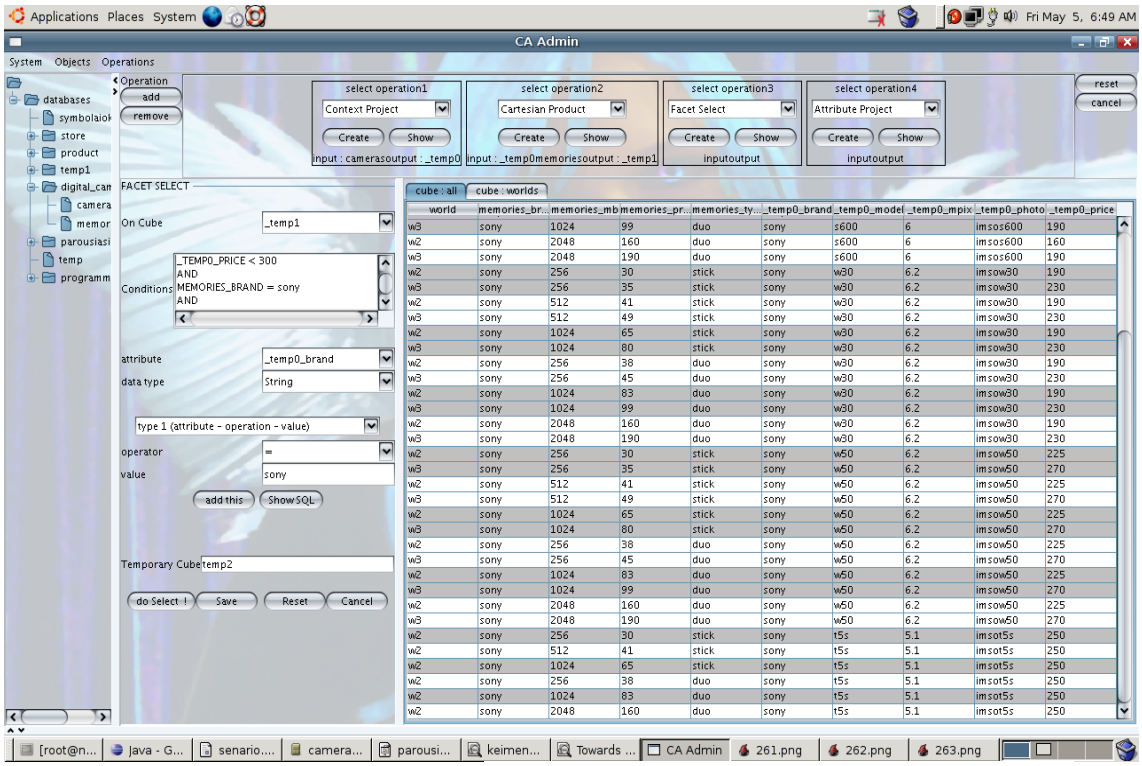
Το ακόλουθο παράδειγμα συγκεντρώνει πολλές από τις λειτουργίες που παρουσιάζονται σε αυτή την παράγραφο. Θεωρείστε έναν πελάτη που χρησιμοποιεί PC και θέλει να αγοράσει μια φωτογραφική μηχανή Sony που κοστίζει λιγότερο από 300 Euro. Θέλει επίσης να αγοράσει μνήμη για αυτήν την φωτογραφική μηχανή, έτσι για όλες τις επιλεγμένες φωτογραφικές μηχανές ζητά έπειτα να δει όλες τις διαθέσιμες εγγυημένες μνήμες (επίσης sony).



Εικόνα 5.12 Ctx-R temp0 ← $\kappa_{[device = pc]}cameras$



Εικόνα 5.13 Ctx-R temp1 ← **context cartesian product(temp0,memories)**



Εικόνα 5.14 Result ← $\sigma^{facet}_{(cameras.brand=memories.brand\ \_AND\_\ cameras.price<300)}temp1$

6

Επίλογος

Η παρούσα διπλωματική εργασία περιέγραψε τη σχεδίαση και την υλοποίηση μιας Context Aware Σχεσιακής Βάσης Δεδομένων, όπου με τον όρο context αναφερόμαστε στις διαφορετικές συνθήκες κάτω από τις οποίες αποκτά υπόσταση η πληροφορία. Περιέλαβε την ανάπτυξη μιας εφαρμογής που χρησιμοποιεί το πιο πάνω σύστημα για να επιδείξει τη χρησιμότητά του. Περιέγραψε το Context Relational Model, το οποίο μετασχηματίστηκε σε σχεσιακούς όρους.

Συγκεκριμένα υλοποιήθηκε ένα σύστημα, μέσω της επέκτασης του σχεσιακού υπολογισμού και της άλγεβρας, που δείχνει πως το context μπορεί να ενσωματωθεί στις σχεσιακές βάσεις δεδομένων. Επρόκειτο για την απόπειρα ομοιόμορφου χειρισμού της πληροφορίας και του context που την χαρακτηρίζει, επιτυγχάνοντας τον βασικό στόχο που ήταν η αντιμετώπιση του context ως «first class citizen».

Επιπλέον υπήρξε και το κομμάτι της διπλωματικής που αφορούσε την ανάπτυξη της εφαρμογής που επιδεικνύει τη λειτουργία του CRM. Χρησιμοποιήθηκε το θεωρητικό μοντέλο που αναπτύχθηκε στο πρώτο κομμάτι για να δημιουργηθεί μια πλατφόρμα με την οποία εκτελείται οποιαδήποτε πράξη σε αυτό.

6.1 Σύνοψη και συμπεράσματα

Υλοποιήθηκε λοιπόν μια εφαρμογή που παρέχει τη δυνατότητα δημιουργίας μιας context-aware βάσης δεδομένων, καθορισμού του context που υποστηρίζει, της δημιουργίας και της επεξεργασίας των σχέσεων που περιλαμβάνει, και της διατύπωσης όλου του φάσματος επερωτήσεων σε αυτή. Με αυτή τη πλατφόρμα επιδεικνύεται η χρηστικότητα του μοντέλου και η εκμετάλλευση των πλεονεκτημάτων που περιέχει η context πληροφορία, στην ιδιαίτερη περίπτωσή όπου το context εντοπίζεται στο επίπεδο των βάσεων δεδομένων.

Η διαχείριση λοιπόν του context στο επίπεδο των πηγών πληροφοριών (δηλ. συστήματα βάσεων δεδομένων) έχει τα ακόλουθα πλεονεκτήματα :

1. Σχεδιασμό περιεκτικών δεδομένων: το context και η σχέση του με τα δεδομένα λαμβάνονται υπόψη στο στάδιο του σχεδιασμού σχήματος. Εκτός από φυσικότερο, η ενσωμάτωση των context όρων στο σχήμα DB επιτρέπει επίσης τον έλεγχο συνέπειας των εισαγωγών και τις αναπροσαρμογές των εξαρτώμενων από το context δεδομένων.
2. Ευρύτερο φάσμα των πιθανών επερωτήσεων: τα μοντέλα δεδομένων και οι γλώσσες επερωτήσεων που ενσωματώνουν άμεσα το context μπορούν να είναι πιο εκφραστικά, οδηγώντας σε νέα είδη επερωτήσεων (cross-world επερωτήσεις [Sta03]), όπως «παρουσιάστε τις εικόνες των φωτογραφικών μηχανών που είναι ακριβότερες κατά την πληρωμή σε μετρητά από την πληρωμή χρησιμοποιώντας την πιστωτική κάρτα», ή επερωτήσεις διαχείρισης δεδομένων όπως «κάτω από ποια contexts δεν υπάρχει καμιά εικόνα».
3. Αποδοτική πρόσβαση: με την ώθηση της διαχείρισης του context στο επίπεδο των πηγών πληροφοριών, τα συστήματα βάσεων δεδομένων έχουν πλήρη έλεγχο και μπορούν να χρησιμοποιήσουν το context-aware σχήμα πληροφορίας τους όπως και ενδεχομένως ειδικά σχεδιασμένες μεθόδους προσπέλασης για να βελτιώσουν την αποδοτικότητα.

6.2 Μελλοντικές επεκτάσεις

Ίσως το σημαντικότερο συμπέρασμα όλων να αποτελούν οι προκλήσεις για ενασχόληση με μια σειρά θεμάτων που αποτέλεσαν το περιεχόμενο της εργασίας, όπως και η εκμετάλλευση των δυνατοτήτων που προσφέρει.

Σε ότι αφορά τη σχεδίαση, υπάρχει ένα ευρύ πεδίο που αφορά την εξέταση διάφορων τρόπων με τους οποίους θα μπορούσε να μεταφραστεί εναλλακτικά το Context-Aware σχεσιακό μοντέλο σε σχεσιακό. Θα ήταν ενδιαφέρουσα μια σύγκριση ανάμεσα σε διαφορετικές μεθοδολογίες, αξιολογώντας διάφορες πτυχές, όπως η απλότητα σχεδίασης, η πολυπλοκότητα της μετατροπής

των ερωτημάτων από το C-A μοντέλο στο σχεσιακό, το κόστος (προσεγγιστικά) σε χώρο και χρόνο που θα κοστίζει κάθε υλοποίηση, το κατά πόσον μπορεί να υλοποιηθεί το σύνολο των πράξεων.

Βασικός στόχος της εφαρμογής ήταν η επίδειξη του C-R σχεσιακού μοντέλου και η υλοποίηση όσο γίνεται περισσοτέρων λειτουργιών με τον απλούστερο τρόπο και όχι η λειτουργία του σε πραγματικές συνθήκες εργασίας. Με βάση αυτό υπάρχει ανάγκη για βελτιστοποίηση των επιδόσεων, που αφορούν το κόστος σε υπολογιστική ισχύ, χρόνο και χώρο. Η εφαρμογή έχει τη δυνατότητα να επεκταθεί και να βελτιωθεί με τον απλούστερο τρόπο. Για αυτόν ακριβώς τον λόγο επιλέχθηκε ένα σχεδιαστικό μοντέλο το οποίο ήταν όσο το δυνατόν καθαρότερο και κατανοητό. Τα βασικά τμήματα της εφαρμογής είναι σε μεγάλο βαθμό ανεξάρτητα και υπάρχει η δυνατότητα να χρησιμοποιηθούν σε μεταγενέστερες υλοποιήσεις.

Από χρηστική σκοπιά η εφαρμογή μπορεί να αποτελέσει εργαλείο για τη θεωρητική μελέτη των λεγόμενων διακοσμικών επερωτήσεων (cross world queries), οι οποίες προσδίδουν μια εντελώς νέα προοπτική που αφορά την άντληση πληροφοριών που βρίσκονται αγκιστρωμένα στο context. Πρόκειται δηλαδή για την αξιοποίηση πλέον της context πληροφορίας.

Ιδιαίτερα ενδιαφέρον αντικείμενο αποτελεί η χρησιμοποίηση της εφαρμογής στην διαχείριση του ιστορικού ενός σχήματος. Σε μια τέτοια προσέγγιση η εφαρμογή θα πρέπει να επιτρέπει να διαγραφούν και να προστεθούν columns στους πίνακες. Ο χρόνος θα θεωρείται σαν context και θα δηλώνεται το χρονικό διάστημα κάτω από το οποίο υφίσταται κάθε column. Στην συνέχεια θα μπορεί να δοθεί κάποια χρονική στιγμή και να ειδωθεί η βάση όπως ήταν δομικά την στιγμή εκείνη.

7

Βιβλιογραφία

[Bro98] G. Brown. Ihtml 2: Design and implementation. In Proceedings of the 11th International Symposium on Languages for Intensional Programming, pages 1-13, 1998.

[CK00] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College Computer Science, 2000.

[DVV03] C. Doulkeridis, E. Valavanis, and M. Vazirgiannis. Towards a context-aware service directory. In Proceedings of the 4th VLDB Workshop on Technologies for E-services (TES'03), 2003.

[Fre84] G. Frege. Die grundlagen der arithmetik. Koebner, Breslau, 1884.

[GG01] Ch. Ghidini and F. Giunchiglia. Local model semantics, or contextual reasoning = locality + compatibility. Artificial Intelligence, 127:221-259, 2001.

[GSK+01] M. Gergatsoulis, Y. Stavrakas, D. Karteris, A. Mouzaki, and D. Sterpis. A web-based system for handling multidimensional information through mxml. In Advances in Databases and Information Systems (ADBIS 01), pages 352-365, 2001.

[MMP95] J. Mylopoulos and R. Motschnig-Pitrik. Partitioning information bases with

contexts. In the 3rd International Conference on Cooperative Information Systems (CoopIS'95), pages 44-55, 1995.

[NP03a] M. C. Norrie and A. Palinginis. Empowering databases for context-dependent information delivery. In CAiSE'03 Workshop on Ubiquitous Mobile Information and Col-laboration Systems (UMICS'03), 2003.

[NP03b] M. C. Norrie and A. Palinginis. From state to structure: an xml web publishing frame-work. In the 15th International Conference on Advanced Information Systems Engineering (CAiSE'03), 2003.

[RSP] Y. Roussos, Y. Stavrakas V. Pavlaki. Towards a Context-Aware Relational Model. Department of Electrical and Computer Engineering, National Technical University of Athens (NTUA), 2005

[SG02] Y. Stavrakas and M. Gergatsoulis. Multidimensional semistructured data : representing context-dependent information on the web. In Advanced Information Systems Engineering, 14th International Conference (CAiSE02), pages 183-199, 2002.

[SGDZ04] Y. Stavrakas, M. Gergatsoulis, Ch. Doulkeridis, and V. Zafeiris. Representing and querying histories of semistructured databases using multidimensional oem. Information Systems, 29:461-482, 2004.

[Sta03] Y. Stavrakas. Multidimensional semistructured data: representing and querying context-dependent multifacet information on the web. PhD thesis, Department of Electrical and Computer Engineering, National Technical University of Athens, June 2003.

[TACS98] M. Theodorakis, A. Analyti, P. Constantopoulos, and N. Spyratos. Context in information bases. In the 3rd International Conference on Cooperative Information Systems (CoopIS'98), 1998.

[VVV+03] E. Valavanis, C. Ververidis, M. Vazirgiannis, G. Polyzos, and K. Norvag. Mobishare: Sharing context-dependent data and services from mobile sources. In Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI2003), 2003.

[Yil97] T. Yildirim. Intensional html. Master's thesis, Department of Computer Science, Univerity of Victoria, 1997.

ΠΑΡΑΡΤΗΜΑ Α

Τεκμηρίωση κλάσεων

Package view

Class Summary	
AttributeProjectPanel	
AttributeProjectPanel1	
CombinedOperations	
ContextProjectPanel	
ContextProjectPanel1	
CreateDatabase	
DropAllTempCubes	
DropCube	
EditCube	
EditCubes	
JoinPanel	
JoinPanel1	
LabelWithSeparator	Copyright: Copyright (c) 2004 Company: GE Energy Services
MainPanelNew	
myTree	
NewCube	
Operation	
SelectEntityPanel	
SelectEntityPanel1	
SelectFacetPanel	
SelectFacetPanel1	
ShowCube	
TreeHandler	
UseCube	
WelcomePanel	
WelcomePanel1	

Package model

Class Summary	
Attribute	
Cube	
Database	
Dimension	
Entity	
SelectEntityModel	
SelectFacetModel	
TableModel	
TableModel1	
TableModelEntitiesPerWorlds	
TableModelForAttributes	
TableModelForCubes	
TableModelForCubesNew	
TableModelForWorldsWorlds	

Package control

Interface Summary	
GeneralDataAccess	

Class Summary	
ControlCenter	
ControlCenterTestCase	
Join	
Postgre_database	
PostgreDataAccess	
PostgreDataAccessTestCase	
Project	
SelectEntity	
SelectFacet	

Class ControlCenter

Constructor Detail

ControlCenter

public **ControlCenter**()

Method Detail

userId

public int **userId**(java.lang.String user_,
java.lang.String password_)

setUser

public void **setUser**(java.lang.String user_,
java.lang.String password_,
Database db)

databaseConnection

public java.sql.Connection **databaseConnection**(java.lang.String DatabaseName)

ConnectionToDatabase

public java.util.Vector **ConnectionToDatabase**(java.lang.String DatabaseName)

Parameters:

DatabaseName -

Returns:

Νομίζω πως τελικά το σετάκι αυτών των δυο συναρτήσεων, είναι ότι καλύτερο και ίσως θα πρέπει να το υιοθετήσω παντού. Η συνάρτηση ConnectionToDatabase(String DatabaseName) μου επιστρέφει μια σύνδεση σε μια βάση, αλλά ταυτόχρονα και το στιγμιότυπο αυτης της βάσης. Με αυτό το τρόπο, έχω πάντα τη δυνατότητα να καλέσω συναρτήσεις αυτής της βάσης, όπως πχ την disconnect. Το πρώτο στοιχείο του διανύσματος που επιστρέφεται περιέχει το στιγμιότυπο της βάσης (δηλαδή μια μεταβλητή τύπου PostgreDataAccess και το δεύτερο περιέχει ένα αντικείμενο σύνδεσης δηλαδή Connection.

Disconnect

public void **Disconnect**(java.sql.Connection con,
PostgreDataAccess database)

establishConnectionClass ControlCenter

public void **establishConnection**(java.lang.String DatabaseName)

getDatabaseCubeNames

public java.util.Vector **getDatabaseCubeNames**(java.lang.String DatabaseName)

Με τις παρακάτω συναρτήσεις θα προσδιορίσουμε πόσοι κύβοι υπάρχουν σε μια βάση δεδομένων, έτσι ώστε να τους τροφοδοτήσουμε στο panel PopulateCube για να μπορούμε να διαλέξουμε. Το σύνολο των tables Αποθηκεύεται σε ένα διάνυσμα (τα ονόματά τους μόνο)

increaseKey

public java.lang.String **increaseKey**(java.lang.String cubeName,
java.lang.String lastKey_)

createNewKey

public java.lang.String **createNewKey**(java.lang.String cubeName)

checkIfEntityExists

public boolean **checkIfEntityExists**(int id,
java.lang.String cubeName,
java.lang.String databaseName)

saveEntity

public int **saveEntity**(Cube cube,
Entity entity,
Database databaseModel)

Η συνάρτηση saveEntity, αναλαμβάνει όλη τη διαδικασία αποθήκευσης μιας νέας οντότητας σε μια σχέση. Πρώτα προσθέτει στον πίνακα Rc_Ctx τους κόσμους στους οποίους ορίζεται η entity και στη συνέχεια προσθέτει στον πίνακα Rc_Ai_Ctx τις τιμές για κάθε Attribute σε κάθε κόσμο.

loadEntity

public Entity **loadEntity**(java.lang.String cubeName,
int id,java.util.Vector connectionData_)

Η συνάρτηση loadEntity δημιουργεί ένα αντικείμενο entity αναζητώντας τα στοιχεία του στη βάση, έχοντας σαν δεδομένο το id του. Καταρχήν αναζητά στον πίνακα Rc_Ctx για να δημιουργήσει μια λίστα με τους επιτρεπτούς κόσμους για αυτό το entity. Στη συνέχεια αναζητά στον πίνακα Rc_Ai_Ctx τα δεδομένα σε όσους κόσμους έχουν προστεθεί.

isCubePopulated

public int **isCubePopulated**(java.lang.String cubeName,java.lang.String databaseName)

Η συνάρτηση isCubePopulated Επιστρέφει 1 αν ο κύβος περιέχει στοιχεία (entities), διαφορετικά επιστρέφει 0.

saveCube

public int **saveCube**(Cube cube,
Database databaseModel)

Σε αντίθεση με τη createCube η οποία κατασκευάζει μόνο τον κύβο στη βάση δεδομένων αλλά χωρίς να προσθέτει entities, αυτή η συνάρτηση και τον κατασκευάζει και αποθηκεύει τα entities που βρίσκονται μέσα στο στιγμιότυπο της κλάσης Cube

removeCube

public int **removeCube**(Database databaseModel,
java.lang.String cubeName)

Η συνάρτηση removeCube διαγράφει εντελώς έναν κύβο, ως εξής 1.Έλεγχος αν υπάρχει αυτός ο κύβος στη βάση (μοντέλο) 2.Διαγραφή των πινάκων Rc_A_ctx και Rc_ctx που αντιστοιχούν στον πίνακα με τη συνάρτηση database.removeCube 3.Διαγραφή του πίνακα Rc_ctx που αντιστοιχεί στον πίνακα (Και τα δυο με μια εντολή, στη συνάρτηση database.removeCube). 4.Διαγραφή του κύβου και του ονόματός του από την κλάση Database.

Parameters:

databaseName -

cubeName -

Returns:

getCADatabases

public java.util.Vector **getCADatabases**()

Με την παρακάτω συνάρτηση βρίσκω τις CA βάσεις δεδομένων

createNodes

public javax.swing.tree.DefaultMutableTreeNode **createNodes**()

Η παρακάτω συνάρτηση θα μου επιστρέφει ένα node δέντρου για να τον περάσω στο MainPanelNew, όπου οι categories και τα leafs θα υπολογίζονται εδώ.

getDataFromEntities

public java.util.HashMap **getDataFromEntities**(Database database,

Cube cube)

Αναλαμβάνει να επιστρέψει ένα hashMap κατάλληλο για να προβάλλουμε έναν κύβο με τη μορφή «ένας κόσμος άνα table»

Returns:

removeEntity

public void **removeEntity**(Entity entity,

Cube cube,

java.lang.String databaseName)

Η συνάρτηση αυτή διαγράφει μια οντότητα

CompareVectors

private java.util.Vector **CompareVectors**(java.util.Vector firstV,

java.util.Vector secondV)

Η συνάρτηση CompareVectors επιστρέφει την τομή 2 διανυσμάτων που περιέχουν string

Parameters:

firstV -

secondV -

Returns:

substractVectors

private java.util.Vector **substractVectors**(java.util.Vector afairetis,
java.util.Vector afaireteos)

Η SupstractVectors συνάρτηση αφαιρεί ένα διάνυσμα που περιέχει string από ένα άλλο και επιτρέφει το αποτέλεσμα σε μορφή διανύσματος

Parameters:

afairetis -

afaireteos -

Returns:

Class PostgreDataAccess

Constructor Detail

PostgreDataAccess

public **PostgreDataAccess**()

Method Detail

PostgreDataAccess

public void **PostgreDataAccess**()

connect

public void **connect**(java.lang.String databaseName)

Specified by:

connect in interface GeneralDataAccess

connect

public int **connect**(java.lang.String databaseName,
java.lang.String user_,
java.lang.String password_)

Specified by:

connect in interface GeneralDataAccess

getConnection

public java.sql.Connection **getConnection**(java.lang.String databaseName)

Specified by:

getConnection in interface GeneralDataAccess

getConnection

public java.sql.Connection **getConnection**(java.lang.String databaseName,
java.lang.String user_,
java.lang.String password_)

Specified by:

getConnection in interface GeneralDataAccess

disconnect

public int **disconnect**()

Specified by:

disconnect in interface GeneralDataAccess

disconnect

public int **disconnect**(java.sql.Connection con)

Specified by:

disconnect in interface GeneralDataAccess

createDatabase

public int **createDatabase**(java.lang.String databaseName)

Δημιουργεί στον Server μια βάση δεδομένων το όνομα που δίνουμε σαν όρισμα

Specified by:

createDatabase in interface GeneralDataAccess

createWorlds

public int **createWorlds**(java.lang.String query)

Κατασκευάζει τον πινακα ctx, ο οποίος περιέχει τους κόσμους που έχουμε ορίσει για μια βάση δεδομένων. Το sql query που πρέπει να εκτελεστεί δίνεται σαν όρισμα εισόδου και παράγεται από την κλάση ControlCenter

Specified by:

createWorlds in interface GeneralDataAccess

populateWorlds

public int **populateWorlds**(java.lang.String query)

Προσθέτει στον πίνακα ctx, όλους τους κόσμους που ορίζονται σε μια δοσμένη βάση, και τις τιμές για το κάθε dimension από τις οποίες αποτελούνται

Specified by:

populateWorlds in interface GeneralDataAccess

retrieveWorlds

public Worlds **retrieveWorlds**()

Specified by:

retrieveWorlds in interface GeneralDataAccess

retrieveWorlds

public Worlds **retrieveWorlds**(java.sql.Connection con)

Specified by:

retrieveWorlds in interface GeneralDataAccess

retrieveWorldsNames

public java.util.Vector **retrieveWorldsNames**(java.sql.Connection con)

Specified by:

retrieveWorldsNames in interface GeneralDataAccess

GetDimensions

public java.util.Vector **GetDimensions**()

Specified by:

GetDimensions in interface GeneralDataAccess

GetDimensions

public java.util.Vector **GetDimensions**(java.sql.Connection c)

Specified by:

GetDimensions in interface GeneralDataAccess

createAttr_Ctx

public int **createAttr_Ctx**()

Specified by:

createAttr_Ctx in interface GeneralDataAccess

createAttr_Ctx

public int **createAttr_Ctx**(java.sql.Connection con)

Specified by:

createAttr_Ctx in interface GeneralDataAccess

createR_Ctx

public int **createR_Ctx**(java.sql.Connection con,
java.lang.String cubeName)

Δημιουργεί τον πίνακα r_ctx στη βάση δεδομένων, ο οποίος θα χρησιμοποιηθεί για να αποθηκευτούν συνδιασμοί οντοτήτων – κόσμων

Specified by:

createR_Ctx in interface GeneralDataAccess

createRc_Ai_Ctx

public int **createRc_Ai_Ctx**(java.sql.Connection con,
java.lang.String cubeName)

Δημιουργεί τον πίνακα Rc_A_Ctx στη βάση στον οποίο θα αποθηκευτούν στη συνέχεια συνδυασμοί οντότητες - κόσμοι - γνωρίσματα- τιμές

Specified by:

createRc_Ai_Ctx in interface GeneralDataAccess

removeCube

public int **removeCube**(java.sql.Connection con,
java.lang.String cubeName)

Διαγράφει έναν κύβο από τη βάση δεδομένων, αφαιρώντας τους αντίστοιχους πίνακες rc_a_ctx και rc_ctx, καθώς και όσες πλειάδες αναφέρονται σε αυτόν από τον πίνακα attr_ctx

Specified by:

removeCube in interface GeneralDataAccess

ifExists

public int **ifExists**(java.lang.String tableName)

Specified by:

ifExists in interface GeneralDataAccess

ifExists

public int **ifExists**(java.lang.String tableName,
java.sql.Connection con)

Specified by:

ifExists in interface GeneralDataAccess

populateAttr_Ctx

public int **populateAttr_Ctx**(java.lang.String query)

Specified by:

populateAttr_Ctx in interface GeneralDataAccess

populateAttr_Ctx

public int **populateAttr_Ctx**(java.lang.String query,
java.sql.Connection con)

Specified by:

populateAttr_Ctx in interface GeneralDataAccess

populateR_Ctx

public int **populateR_Ctx**(java.lang.String query,
java.sql.Connection con)

Specified by:

populateR_Ctx in interface GeneralDataAccess

removeFromR_Ctx

public int **removeFromR_Ctx**(java.lang.String cubeName,
int entityId,
java.lang.String worldName,
java.sql.Connection con)

Η συνάρτηση removeFromR_Ctx διαγράφει μια πλειάδα του πίνακα r_ctx, υλοποιώντας την αφάιρεση ενός κόσμου από το πεδίο ορισμού μιας οντότητας.

Specified by:

removeFromR_Ctx in interface GeneralDataAccess

Parameters:

cubeName -

entityId -

worldName -

con -

Returns:

insertIntoR_Ctx

public int **insertIntoR_Ctx**(java.lang.String cubeName,
int entityId,
java.lang.String worldName,
java.sql.Connection con)

Specified by:

insertIntoR_Ctx in interface GeneralDataAccess

Parameters:

cubeName -

entityId -

worldName -

con -

Returns:

Εισάγει μια πλειάδα στον πίνακα r_ctx, υλοποιώντας την εισαγωγή ενός κόσμου στη βάση δεδομένων

populateRc_Ai_Ctx

public int **populateRc_Ai_Ctx**(java.lang.String query,
java.sql.Connection con)

Προσθέτει συνδιασμούς οντότητας-κόσμου-γνωρίσματος-τιμής σε έναν κύβο

Specified by:

populateRc_Ai_Ctx in interface GeneralDataAccess

insertIntoRc_Ai_Ctx

public int **insertIntoRc_Ai_Ctx**(java.lang.String cubeName,
int entityId,
java.lang.String worldName,
java.lang.String attrName,
java.lang.String value,
java.sql.Connection con)

Specified by:

insertIntoRc_Ai_Ctx in interface GeneralDataAccess

removeFromRc_Ai_Ctx

public int **removeFromRc_Ai_Ctx**(java.lang.String cubeName,
int entityId,
java.lang.String worldName,
java.sql.Connection con)

Αφαιρεί ένα κόσμο που δίνεται σαν είσοδος από το σύνολο στο οποίο ορίζεται μια οντότητα ενός κύβου

Specified by:

removeFromRc_Ai_Ctx in interface GeneralDataAccess

Parameters:

cubeName -

entityId -

worldName -

con -

Returns:

updateRc_Ai_Ctx

public int **updateRc_Ai_Ctx**(java.lang.String cubeName,

int entityId,
java.lang.String worldName,
java.lang.String attrName,
java.lang.String value,
java.sql.Connection con)

Specified by:

updateRc_Ai_Ctx in interface GeneralDataAccess

checkIfAttributeExists

public int **checkIfAttributeExists**(java.sql.Connection con,
java.lang.String query)

Ελέγχει αν ένα attribute ήδη υπάρχει στη βάση

Specified by:

checkIfAttributeExists in interface GeneralDataAccess

checkWhereAttributeExists

public java.util.HashMap **checkWhereAttributeExists**(java.sql.Connection con,
Cube cube,
java.lang.String AttributeName)

Specified by:

checkWhereAttributeExists in interface GeneralDataAccess

checkWhereAttributeExists1

public java.util.HashMap **checkWhereAttributeExists1**(java.sql.Connection con,
java.lang.String cubeName,
java.lang.String AttributeName)

Specified by:

checkWhereAttributeExists1 in interface GeneralDataAccess

removeAttributeFromCube

public int **removeAttributeFromCube**(java.sql.Connection con,
Cube cube,
java.lang.String attributeName)

Specified by:

removeAttributeFromCube in interface GeneralDataAccess

addAttributeToCube

public int **addAttributeToCube**(java.sql.Connection con,
Cube cube,
Attribute attr)

Προσθέτει ένα Attribute σε έναν κύβο

Specified by:

addAttributeToCube in interface GeneralDataAccess

getDatabaseCubes

public java.util.Vector **getDatabaseCubes**()

Specified by:

getDatabaseCubes in interface GeneralDataAccess

getDatabaseCubes

public java.util.Vector **getDatabaseCubes**(java.sql.Connection c)

Specified by:

getDatabaseCubes in interface GeneralDataAccess

getCubeAttributes

public java.util.Vector **getCubeAttributes**(java.sql.Connection con,
java.lang.String cubeName)

Specified by:

getCubeAttributes in interface GeneralDataAccess

checkIfRelationExists

public int **checkIfRelationExists**(java.lang.String cubeName,
java.sql.Connection con)

Specified by:

checkIfRelationExists in interface GeneralDataAccess

checkIfFirstEntity

public int **checkIfFirstEntity**(java.lang.String cubeName,
java.sql.Connection con)

Specified by:

checkIfFirstEntity in interface GeneralDataAccess

getLastEntityKey

public int **getLastEntityKey**(java.lang.String cubeName,
java.sql.Connection con)

Specified by:

getLastEntityKey in interface GeneralDataAccess

ifEntityExists

public boolean **ifEntityExists**(int id,
java.lang.String cubeName,
java.sql.Connection con)

Specified by:

ifEntityExists in interface GeneralDataAccess

getDatabases

public java.util.Vector **getDatabases**(java.sql.Connection con)

Specified by:

getDatabases in interface GeneralDataAccess

getValidWorldsForEntity

public java.util.Vector **getValidWorldsForEntity**(java.lang.String cubeName,
int id,
java.sql.Connection con)

Specified by:

getValidWorldsForEntity in interface GeneralDataAccess

getAttributesAndValuesForWorld

public java.util.HashMap **getAttributesAndValuesForWorld**(java.lang.String cubeName,
java.lang.String worldName,
int id,
java.sql.Connection con)

Specified by:

getAttributesAndValuesForWorld in interface GeneralDataAccess

getHowManyEntitiesInCube

public int **getHowManyEntitiesInCube**(java.lang.String cubeName,
java.sql.Connection con)

Specified by:

getHowManyEntitiesInCube in interface GeneralDataAccess

FacetSelect

public java.util.Vector **FacetSelect**(java.lang.String query,
java.sql.Connection con)

Η συνάρτηση FacetSelect καλείται από την κλάση control.Select εκτελεί ένα προκαθορισμένο query που δέχεται από την control.Select και επιστρέφει ένα διάνυσμα. Αν η εκτέλεση είναι επιτυχής, το διάνυσμα περιέχει ένα μήνυμα επιτυχίας "ok" και ένα entity - χωρίς όμως αυτό το entity να έχει id ! Διαφορετικά, επιστρέφει ένα μήνυμα λάθους "oups" καθώς και το μήνυμα λάθους που επιστρέφει το σύστημα διαχείρισης βάσεων δεδομένων

Parameters:

query -

con -

Returns:

EntitySelectFirstType

public java.util.Vector **EntitySelectFirstType**(java.lang.String query,
java.sql.Connection con)

Specified by:

EntitySelectFirstType in interface GeneralDataAccess

Class Join

Constructor Detail

Join

public **Join**(Cube cube1,
Cube cube2)

Method Detail

Join

public Cube **Join**(Cube cube1,
Cube cube2,
java.lang.String newCubeName)

joinAttributes

private java.util.Vector **joinAttributes**()

Η συνάρτηση joinAttributes επιστρέφει το σύνολο των attributes που θα ορίζονται στον κύβο ο οποίος θα περιέχει το καρτεσιανό γινόμενο των 2 πινάκων.

joinAttributeNames

private java.util.Vector **joinAttributeNames**()

Η συνάρτηση joinAttributeNames επιστρέφει τα ονόματα του συνόλου των attributes που θα ορίζονται οστον κύβο ο οποίος θα περιέχει το καρτεσιανό γινόμενο των 2 πινάκων.

renameCubeAttributes

private java.util.Vector **renameCubeAttributes**(Cube cube)

Η συνάρτηση renameCubeAttributes επιστρέφει ένα διάνυσμα το οποίο περιέχει τα νέα ονόματα των attributes μόνο. Τα attributes μετονομάζονται, προσθέτοντας σαν πρόθεμα το όνομα του κύβου από τον οποίο προέρχονται.

Parameters:

cube -

Returns:

rebuildCubeAttributes

private java.util.Vector **rebuildCubeAttributes**(Cube cube)

Η συνάρτηση rebuildCubeAttributes αλλάζει το όνομα σε κάθε model.Attribute που περιέχει ο κύβος, με την ίδια αρχή που το κάνει και η συνάρτηση renameCubeAttributes, προσθέτοντας σαν πρόθεμα το όνομα του κύβου από τον οποίο προέρχονται.

Parameters:

cube -

Returns:

rebuildEntitiesAttributes

private java.util.Vector **rebuildEntitiesAttributes**(Cube cube)

Η συνάρτηση rebuildEntitiesAttributes αναλαμβάνει να επιστρέψει ένα διάνυσμα από entities, το οποίο είναι ίδιο με αυτό του κύβου που δίνεται σαν είσοδος, μόνο που στο HashMap data έχουμε αλλάξει όλα τα attributes με τα νέα ονόματα που θα πρέπει να έχουν

Parameters:

cube -

Returns:

joinEntities

private java.util.Vector **joinEntities**(Cube cube1,
Cube cube2)

CompareVectors

private java.util.Vector **CompareVectors**(java.util.Vector firstV,
java.util.Vector secondV)

Η συνάρτηση CompareVectors επιστρέφει την τομή 2 διανυσμάτων που περιέχουν string

Parameters:

firstV -

secondV -

Returns:

getJoinAttributes

public java.util.Vector **getJoinAttributes**()

getJoinAttributeNames

public java.util.Vector **getJoinAttributeNames**()

getJoinEntities

public java.util.Vector **getJoinEntities**()

getJoinCube

public Cube **getJoinCube**(java.lang.String cubeName)

Class Project

Constructor Detail

Project

public **Project**()

Method Detail

attributeProject

public Cube **attributeProject**(Cube cube,java.util.
Vector attributes,
java.lang.String nameForNewCube,
Database databaseModel,
ControlCenter cc)

Η συνάρτηση AttributeProject, υλοποιεί το Attribute - Project Λαμβάνοντας σαν όρισμα έναν κύβο και ένα σύνολο γνωρισμάτων, επιστρέφει ένα νέο κύβο που έχει τα γνωρίσματα τα οποία δώσαμε σαν όρισμα. Η υλοποίηση γίνεται ως εξής : Ελέγχουμε κατ'αρχήν αν τα γνωρίσματα που δίνονται υπάρχουν στον δοσμένο κύβο Φορτώνουμε όλες τις οντότητες αυτού του κύβου και κρατάμε όσα γνωρίσματα χρειαζόμαστε. Δημιουργούμε ένα νέο κύβο. Φορτώνουμε σε αυτόν τις οντότητες που έχουμε δημιουργήσει. Πρέπει να σημειωθεί, πως οι πράξεις που κάνουμε, δεν εφαρμόζονται απευθείας στη βάση δεδομένων με τη χρήση queries. Αυτό γίνεται, έτσι ώστε να έχουμε μεγαλύτερη ευελιξία στη δημιουργία σύνθετων πράξεων.

Parameters:

cube -
attributes -
databaseModel -
cc -

Returns:

checkAttributes

public boolean **checkAttributes**(Cube cube,java.util.
Vector attributes)

Συνάρτηση βοηθητική για το AttributeProject. Ελέγχει αν τα attributes που δίνονται υπάρχουν πράγματι στον κύβο

Parameters:

cube -
attributes -

Returns:

setAttributes

public Entity **setAttributes**(Entity entity,
java.util.Vector attributes)

Συνάρτηση βοηθητική για το AttributeProject. Ανακατασκευάζει τα entities του αρχικού κύβου με βάση τα attribute που θέλουμε να υπάρχουν μετά το project.

Parameters:

entity -

attributes -

Returns:

worldProject

public Cube **worldProject**(Cube cube,
java.util.Vector worldNames,
java.lang.String nameForNewCube,
Database databaseModel,
ControlCenter cc)

getObsoleteWorlds

private java.util.Vector **getObsoleteWorlds**(java.util.Vector worldNames,
java.util.Vector definedWorlds)

Η συνάρτηση getObsoleteWorlds παίρνει σαν όρισμα ένα διάνυσμα με τους κόσμους που θα παραμείνουν και επιστρέφει ένα διάνυσμα με τους κόσμους που πρέπει να αφαιρεθούν από το σύνολο των κόσμων που ορίζονται για να παραμείνουν οι κόσμοι που μας δίνονται.

Parameters:

worldNames -

Returns:

Class SelectEntity

Constructor Detail

SelectEntity

public **SelectEntity**(Cube cube,
java.util.Vector operations,
java.lang.String databaseName,
java.lang.String user,
java.lang.String pass)

Αυτός είναι ο constructor για το Select - Entity

Parameters:

entity -

operations -

cubeName -

pda -

Method Detail

getEntities

public java.util.Vector **getEntities**()

getSQL

public java.util.Vector **getSQL**(Cube cube,
java.util.Vector operations,
java.lang.String databaseName)

getFirstTypeEntities

public java.util.Vector **getFirstTypeEntities**(SelectEntityModel sem,
Cube cube,
java.sql.Connection con,
PostgreDataAccess database)

getSecondTypeEntities

public java.util.Vector **getSecondTypeEntities**(SelectEntityModel sem,
Cube cube,
java.sql.Connection con,
PostgreDataAccess database)

getThirdTypeEntities

public java.util.Vector **getThirdTypeEntities**(SelectEntityModel sem,
Cube cube,
java.sql.Connection con,
PostgreDataAccess database)

makeFirstTypeQuery

private java.lang.String **makeFirstTypeQuery**(SelectEntityModel sem,

int entityIdValue_)

Returns:

makeFirstTypeQuery : αναζήτηση αν για όλους τους κόσμους το συγκεκριμένο attribute για ένα συγκεκριμένο κάθε φορά entity (που ορίζεται από το entity_id) επαληθεύει τη συνθήκη. (περίπτωση 1)

makeSecondTypeQuery

private java.lang.String **makeSecondTypeQuery**(SelectEntityModel sem,

int entityIdValue_)

Parameters:

sem -

entityIdValue_ - makeSecondTypeQuery : αναζήτηση αν υπάρχει έστω και ένας κόσμος όπου το attribute για ένα συγκεκριμένο entity επαληθεύει τη συνθήκη.

(περίπτωση 2) Η συνθήκη ισχύει εφόσον το ερώτημα επιστρέψει ακέραιο μεγαλύτερο του μηδενός. Η συνάρτηση αυτή πρέπει να καλεστεί για όλα τα entities προκειμένου να βρούμε το σύνολο αυτών που επαληθεύουν τη συνθήκη.

Returns:

makeThirdTypeQuery

private java.lang.String **makeThirdTypeQuery**(SelectEntityModel sem,

java.util.Vector worlds)

Parameters:

sem -

entityIdValue_ - για κάθε attribute το οποίο εμφανίζεται στη συνθήκη, ελέγχω σε ποια entities ισχύει η πράξη που γίνεται. Η τομή των entities, είναι το επιστρέψιμο σύνολο. Τελικά, το sql ερώτημα που δημιουργούμε εδώ, επιστρέφει κατ'ευθείαν το σύνολο των entities τα οποία επαληθεύουν τη συνθήκη.

Returns:

CompareVectors

public java.util.Vector **CompareVectors**(java.util.Vector firstV,

java.util.Vector secondV)

Η συνάρτηση CompareVectors επιστρέφει την τομή 2 διανυσμάτων που περιέχουν integers

Parameters:

firstV -

secondV -

Returns:

ConnectionToDatabase

private java.util.Vector **ConnectionToDatabase**(java.lang.String DatabaseName)

addMessage

public java.util.Vector **addMessage**(java.lang.String message,
java.util.Vector messageVector)

getMessages

public java.util.Vector **getMessages**()

Class SelectFacet

Constructor Detail

SelectFacet

public **SelectFacet**(Entity entity,
java.util.Vector operations,
java.lang.String cubeName,
java.lang.String databaseName,
java.lang.String user,
java.lang.String pass)

Αυτός είναι ο constructor για το Select - Facet

Parameters:

entity -

operations -

cubeName -

pda -

Method Detail

makequery

private java.lang.String **makequery**()

getFacet

public java.util.Vector **getFacet**()

Στέλνω το query στην postgre και παίρνω σαν αποτέλεσμα ένα νέο entity

Returns:

ConnectionToDatabase

private java.util.Vector **ConnectionToDatabase**(java.lang.String DatabaseName)

addMessage

public java.util.Vector **addMessage**(java.lang.String message,
java.util.Vector messageVector)

getMessages

public java.util.Vector **getMessages**()

Class Attribute

Constructor Detail

Attribute

public **Attribute**()

Method Detail

DefineInDimension

public void **DefineInDimension**(java.lang.String world)

UnDefineInDimension

public void **UnDefineInDimension**(java.lang.String world)

getAttributeName

public java.lang.String **getAttributeName**()

Returns:

Returns the attributeName.

setAttributeName

public void **setAttributeName**(java.lang.String attributeName)

Parameters:

attributeName - The attributeName to set.

getWhereIsDefined

public java.util.HashMap **getWhereIsDefined**()

Returns:

Returns the whereIsDefined.

getWhereIsDefinedVector

public java.util.Vector **getWhereIsDefinedVector**()

setWhereIsDefined

public void setWhereIsDefined(java.util.HashMap whereIsDefined)

Parameters:

whereIsDefined - The whereIsDefined to set.

isAllowNull

public boolean **isAllowNull**()

Returns:

Returns the allowNull.

setAllowNull

public void **setAllowNull**(boolean allowNull)

Parameters:

allowNull - The allowNull to set.

Class Cube

Constructor Detail

Cube

public **Cube**()

Method Detail

addAttribute

public void **addAttribute**(Attribute attribute)

getAttributeAtIndex

public Attribute **getAttributeAtIndex**(int i)

getAttributeByName

public Attribute **getAttributeByName**(java.lang.String attrName)

getCubeName

public java.lang.String **getCubeName**()

Returns:

Returns the cubeName.

setCubeName

public void **setCubeName**(java.lang.String cubeName)

Parameters:

cubeName - The cubeName to set.

getAttributes

public java.util.Vector **getAttributes**()

Returns:

Returns the attributes.

setAttributes

public void **setAttributes**(java.util.Vector attributes)

Parameters:

attributes - The attributes to set.

getAttributeNames

public java.util.Vector **getAttributeNames**()

getEntities

public java.util.Vector **getEntities**()

setEntities

public void **setEntities**(java.util.Vector entities)

addEntities

public void **addEntities**(Entity entity)

getEntityIds

public java.util.Vector **getEntityIds**()

getEntityById

public Entity **getEntityById**(int id)

Class Database

Constructor Detail

Database

public **Database**()

Method Detail

Database

public void **Database**()

getDatabaseName

public java.lang.String **getDatabaseName**()

Returns:

Returns the databaseName.

setDatabaseName

public void **setDatabaseName**(java.lang.String databaseName)

Parameters:

databaseName - The databaseName to set.

getEncoding

public java.lang.String **getEncoding**()

Returns:

Returns the encoding.

setEncoding

public void **setEncoding**(java.lang.String encoding)

Parameters:

encoding - The encoding to set.

getNumberOfDimensions

public int **getNumberOfDimensions**()

Returns:

Returns the numberOfDimensions.

setNumberOfDimensions

public void **setNumberOfDimensions**(int numberOfDimensions)

Parameters:

numberOfDimensions - The numberOfDimensions to set.

getDimensions

public java.util.Vector **getDimensions**()

Returns:

Returns the dimensions.

setDimensions

public void **setDimensions**(java.util.Vector dimensions)

Parameters:

dimensions - The dimensions to set.

addDimension

public void **addDimension**(Dimension dimension)

getWorlds

public Worlds **getWorlds**()

setWorlds

public void **setWorlds**(Worlds worlds)

getCubes

public java.util.Vector **getCubes**()

setCubes

public void **setCubes**(java.util.Vector cubes)

addCube

public void **addCube**(Cube cube)

getCubeByName

public Cube **getCubeByName**(java.lang.String cubeName)

removeCubeByName

public void **removeCubeByName**(java.lang.String cubeName)

getCubeNames

public java.util.Vector **getCubeNames**()

setCubeNames

public void **setCubeNames**(java.util.Vector cubeNames)

addCubeNames

public void **addCubeNames**(java.lang.String cubeName)

getWorldNames

public java.util.Vector **getWorldNames**()

setWorldNames

public void **setWorldNames**(java.util.Vector worldNames)

getTempCubeName

public java.lang.String **getTempCubeName**()

removeTempCubes

public void **removeTempCubes**()

getTempCubeNames

public java.util.Vector **getTempCubeNames**()

getTempCubes

public java.util.Vector **getTempCubes**()

setTempCubes

public void **setTempCubes**(java.util.Vector tempCubes)

addTempCube

public void **addTempCube**(java.lang.String cubeName)

removeTempCube

public java.lang.String **removeTempCube**(java.lang.String cubeName)

Class Dimension

Constructor Detail

Dimension

public **Dimension**()

Method Detail

getDimensionName

public java.lang.String **getDimensionName**()

Returns:

Returns the dimensionName.

setDimensionName

public void **setDimensionName**(java.lang.String dimensionName)

Parameters:

dimensionName - The dimensionName to set.

getValues

public java.util.Vector **getValues**()

setValue

public void **setValue**(java.util.Vector value)

Parameters:

value - The value to set.

addValue

public void **addValue**(java.lang.String value)

getValue

public java.lang.String **getValue**(int i)

Class Entity

Constructor Detail

Entity

public **Entity**()

Το πεδίο data είναι ένα hasmap που εκφράζει μια αντιστοίχιση ανάμεσα σε ονόματα κόσμων και σε hashmaps. Αυτά τα hashmaps με τη σειρά τους, αντιστοιχίζουν γνωρίσματα με τιμές (attributes-values). Το πεδίο validWorlds είναι ένα διάνυσμα το οποίο περιέχει τους κόσμους στους οποίους ορίζεται αυτή η οντότητα. Το πεδίο id είναι το αναγνωριστικό αυτής της οντότητας.

Method Detail

getData

public java.util.HashMap **getData**()

setData

public void **setData**(java.util.HashMap data)

addData

public void **addData**(java.lang.String worldName,
java.util.HashMap AllValuesForaWorld)

removeData

public void **removeData**(java.lang.String worldName)

getValidWorlds

public java.util.Vector **getValidWorlds**()

setValidWorlds

public void **setValidWorlds**(java.util.Vector validWorlds)

getId

public int **getId**()

setId

public void **setId**(int id)

Class Worlds

Constructor Detail

Worlds

public **Worlds**(java.util.Vector dimensions_)

Method Detail

createWorlds

public void **createWorlds**()

Αυτή η συνάρτηση κατασκευάζει το σύνολο των εφικτών κόσμων, παίρνοντας σαν όρισμα τις διαστάσεις που έχω ορίσει στη βάση δεδομένων.

multiply_sizes

private int **multiply_sizes**(int startDimension,
int endDimension)

numberOfWorlds

private int **numberOfWorlds**()

getWorlds

public java.util.Vector **getWorlds**()

Returns:

Returns the worlds.

setWorlds

public void **setWorlds**(java.util.Vector worlds)

Parameters:

worlds - The worlds to set.

howManyWorlds

public int **howManyWorlds**()

getWorldNameAtIndex

public java.lang.String **getWorldNameAtIndex**(int i)

getDimensions

public java.util.Vector **getDimensions**()

setDimensions

public void **setDimensions**(java.util.Vector dimensions)

getWorldNames

public java.util.Vector **getWorldNames**()

Class SelectEntityModel

Constructor Detail

SelectEntityModel

public **SelectEntityModel**(java.lang.String attribute1,
java.lang.String attribute1_class,
java.util.Vector attribute1Worlds,
java.util.Vector attribute1DefinedWorlds,
java.lang.String operator,
java.lang.String value)

SelectEntityModel

public **SelectEntityModel**(java.lang.String attribute1,
java.lang.String attribute1_class,
java.util.Vector attribute1Worlds,
java.util.Vector attribute1DefinedWorlds,
java.lang.String operator,
java.lang.String attribute2,
java.lang.String world)

Method Detail

getAttribute1

public java.lang.String **getAttribute1**()

getAttribute2

public java.lang.String **getAttribute2**()

getOperator

public java.lang.String **getOperator**()

getType

public java.lang.String **getType**()

getValue

public java.lang.String **getValue**()

getWorld

public java.lang.String **getWorld**()

getAttribute1_class

public java.lang.String **getAttribute1_class**()

makeCRSQL

public java.lang.String **makeCRSQL**()

Η συνάρτηση makeCRSQL, με βάση τα δεδομένα που ορίζουμε στον constructor της κλάσης, επιστρέφει την condition σε μορφή CR-SQLa

Returns:

getAttribute1Worlds

public java.util.Vector **getAttribute1Worlds**()

Class SelectFacetModel

Constructor Detail

SelectFacetModel

public **SelectFacetModel**(java.lang.String attribute1,
java.lang.String attribute1_class,
java.lang.String operator,
java.lang.String value)

SelectFacetModel

public **SelectFacetModel**(java.lang.String attribute1,
java.lang.String attribute1_class,
java.lang.String operator,
java.lang.String attribute2,
java.lang.String world)

Method Detail

getAttribute1

public java.lang.String **getAttribute1**()

getAttribute2

public java.lang.String **getAttribute2**()

getOperator

public java.lang.String **getOperator**()

getType

public java.lang.String **getType**()

getValue

public java.lang.String **getValue**()

getWorld

public java.lang.String **getWorld**()

getAttribute1_class

public java.lang.String **getAttribute1_class**()

makeCRSQL

public java.lang.String **makeCRSQL**()

Η συνάρτηση makeCRSQL, με βάση τα δεδομένα που ορίζουμε στον constructor της κλάσης, επιστρέφει την condition σε μορφή CR-SQLa

Returns: