ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Γραφικό Περιβάλλον Χρήστη για Εξατομικευμένο Σύστημα Διαχείρισης Βάσεων Δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΜΑΡΟΥΔΑ ΕΜΜΑΝΟΥΗΛ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2012

Περίληψη

Στα πλαίσια της γενικότερης ανάπτυξης του κοινωνικού διαδικτύου, του ηλεκτρονικού εμπορίου αλλά και της ψυχαγωγίας μέσω του διαδικτύου, προκύπτουν πολυάριθμα συστήματα τα οποία μπορούν να αξιοποιήσουν τις προτιμήσεις των χρηστών τους για να τους προσφέρουν μια καλύτερη εμπειρία. Στο επίπεδο της βάσης δεδομένων όμως, η εκφραστικότητα του απλού σχεσιακού μοντέλου και οι αυστηροί περιορισμοί που επιβάλλονται κατά τη διατύπωση των ερωτημάτων που το χρησιμοποιούν, δεν είναι επαρκείς για να ενσωματωθούν οι –πολλές φορές αβέβαιες και ποικίλης βαρύτητας- προτιμήσεις των χρηστών.

Το σχεσιακό μοντέλο λοιπόν χρειάζεται να επεκταθεί για να συμπεριλάβει την έννοια της προτίμησης. Μια προσέγγιση για αυτή την επέκταση αποτελεί το σύστημα PrefDB, το οποίο τρέχει πάνω σε μια οποιαδήποτε σχεσιακή βάση δεδομένων και επεκτείνει τις λειτουργίες της, ώστε να μπορούν να οριστούν σε αυτή προτιμήσεις και να εκτελεστούν ερωτήματα που τις χρησιμοποιούν. Τα αποτελέσματα που επιστρέφονται έχουν δύο επιπλέον παραμέτρους: Τη βαθμολογία του καθενός με βάση τις προτιμήσεις που χρησιμοποιήθηκαν και την εμπιστοσύνη , που είναι ένα μέτρο της βεβαιότητας πως αυτή η βαθμολογία ανταποκρίνεται όντως στις προτιμήσεις του χρήστη.

Ο σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μιας διαπροσωπείας χρήστη με τη μορφή διαδικτυακής εφαρμογής για τη διαχείριση συστημάτων που ενσωματώνουν το PrefDB. Η διαπροσωπεία αυτή υλοποιήθηκε με στόχο να είναι χρήσιμη για διαχειριστές τέτοιων συστημάτων,καθώς και για προγραμματιστές που φτιάχνουν εφαρμογές που στηρίζονται σε τέτοια συστήματα. Οι βασικές δυνατότητες της διαπροσωπείας είναι: Προσθαφαίρεση και επεξεργασία προτιμήσεων και προφίλ (ομάδες προτιμήσεων), αποθήκευση και φόρτωση ερωτημάτων με προτιμήσεις, πλοήγηση στο σχήμα των βάσεων, εκτέλεση ερωτημάτων και διαχείριση των αποτελεσμάτων.

Στη διαδικάσία του σχεδιασμού, ξεκινώντας με πρότυπο τις εφαρμογές που παρέχουν τα ήδη υπάρχοντα συστήματα διαχείρισης βάσεων δεδομένων,καθορίστηκαν οι προδιαγραφές για το σύστημα. Στη συνέχεια, έγινε ανάπτυξη στο ευρέως διαδεδομένο περιβάλλον HTML/CSS/Javascript του πυρήνα της εφαρμογής, ενώ στη συνέχεια εμπλουτίστηκε με τη χρήση διαφόρων εργαλείων, ώστε να ικανοποιούνται όλες οι προδιαγραφές.

Λέξεις Κλειδιά: Βάσεις Δεδομένων, Προτιμήσεις, Εξατομίκευση

Abstract

Considering the evolution of the social web, e-commerce and entertainment through the internet, a significant number of systems have emerged that can utilize user preferences to enhance the user experience. However, expressing preferences using the relational model is not practical or even possible, due to the strict restrictions that it applies to search criteria used in queries, which directly contrasts the uncertainty and variety in significance that user preferences can have.

It is therefore apparent that the relational model needs to be expanded in order to be able to express preferences. One of the approaches towards this direction is PrefDB, a system that can be installed on any relational database and expand its functionality to include defining preferences and using them in queries. The results of a query enhanced with preferences include two additional columns, the score of each result, a measure of how good a fit each particular result is according to the preferences used and the confidence of each result, a measure of how certain it is that the preferences used and assorted scores actually reflect what the user likes or dislikes.

The scope of this diploma thesis was to develop a user interface, in the form of a web-based application, in order to make managing applications that implement PrefDB easier. The interface was geared towards administrators of said databases, as well as developers utilizing them. The basic functions of the interface are: Creating and editing preferences and preference groups (profiles), executing queries, browsing results and browsing the database schema.

The design process used the already existing database management systems for relational databases as a starting point, continuing with the specification of the functional and non-functional requirements for the system. Implementation of said requirements was the next step, using HTML/CSS/Javascript as the core of the application and using a wide spectrum of tools and frameworks to enhance the look and feel, as well as the functionality of the system.

Λέξεις Κλειδιά: Databases, Preferences, Personalization

Πίνακας περιεχομένων

1 Εισαγωγή

1.1 Εξατομικευμένα συστήματα βάσεων δεδομένων

Στα πλαίσια της εξέλιξης του social web αλλά και των online υπηρεσιών πώλησης, συναντώνται όλο και πιο συχνά περιπτώσεις όπου οι προτιμήσεις του χρήστη έχουν σημασία για να καθοριστεί η σελίδα που πρέπει να του παρουσιαστεί. Η εξατομίκευση ερωτημάτων σε βάσεις, οι προτάσεις για αγορά ή για συνδρομή σε κάποια υπηρεσία, οι κοινωνικές εφαρμογές, καθώς και διάφορα άλλα πεδία χρησιμοποιούν αλγόριθμους για να αξιοποιήσουν τις προτιμήσεις των χρηστών τους, τα δεδομένα που έχουν συλλέξει για αυτούς. Πολλές από τις μέχρι στιγμής προσεγγίσεις για την ενσωμάτωση των προτιμήσεων των χρηστών στα ανωτέρω συστήματα τις αντιμετωπίζουν σε επίπεδο επεξεργασίας των δεδομένων που επιστρέφονται από τη βάση ή με τη βοήθεια εξειδικευμένων τελεστών. Μια τέτοια προσέγγιση επιβάλει πολλούς περιορισμούς στο συνδυασμό ερωτημάτων και προτιμήσεων και επιπλέον ένα σύστημα που αναπτύσσεται για κάποια δεδομένα δε λειτουργεί για άλλα.

Θα ήταν χρήσιμο να δημιουργηθεί μια επέκταση του σχεσιακού μοντέλου η οποία να δίνει τη δυνατότητα να λαμβάνονται υπόψιν οι προτιμήσεις των χρηστών ανεξάρτητα από τη δομή της βάσης, με μια νέα σχεσιακή άλγεβρα. Με τη χρήση νέων τελεστών, είναι δυνατόν να αξιοποιηθούν σε ερωτήματα οι προτιμήσεις του χρήστη,και να του δοθούν αντίστοιχα αποτελέσματα ανεξάρτητα από τον τύπο ή τη μορφή των δεδομένων που περιέχονται στη βάση και χωρίς να επιβαρύνεται η εφαρμογή με επεξεργασία των δεδομένων που επιστρέφονται από το ερώτημα.

Μια τέτοια επέκταση είναι το PrefDB. Επεκτείνει τα συνήθη σχεσιακά ερωτήματα με τον τελεστή προτίμησης και την έννοια της βαθμολογίας και τις εμπιστοσύνης για τα αποτελέσματα. Προσθέτοντας στα ερωτήματα προτιμήσεις οι οποίες εξαρτώνται από τα attributes των πινάκων της βάσης και παίρνοντας ως είσοδο μια σειρά παραμέτρων που

καθορίζουν πώς θα χρησιμοποιηθούν οι προτιμήσεις αυτές για την εξαγωγή αποτελεσμάτων, το PrefDB μπορεί να τρέξει πάνω σε μια οποιαδήποτε σχεσιακή βάση δεδομένων Με αυτό τον τρόπο, τα δεδομένα που τελικά επιστρέφει η βάση είναι αυτά που ανταποκρίνονται στο ερώτημα που έγινε με βάση τις προτιμήσεις που χρησιμοποιήθηκαν, χωρίς να απαιτείται περαιτέρω επεξεργασία.

Για να αξιοποιηθεί σωστά το PrefDB, είναι χρήσιμο να υπάρχει και ένα σύστημα διαχείρισης βάσεων δεδομένων με προτιμήσεις. Αντικείμενο της παρούσας διπλωματικής είναι η ανάπτυξη ενός τέτοιου συστήματος διαχείρισης, το οποίο θα διευκολύνει τόσο τους διαχειριστές των βάσεων δεδομένων που ενσωματώνουν το PrefDB, όσο και τους developers των οποίων οι εφαρμογές χρησιμοποιούν τέτοιες βάσεις.

1.2 Αντικείμενο Διπλωματικής

Στόχος λοιπόν για τη διπλωματική ήταν η ανάπτυξη μιας εφαρμογής, χρηστικής και λειτουργικής, η οποία θα βοηθούσε στην εκτέλεση ερωτημάτων και τη διαχείριση των προτιμήσεων, προφίλ και παραμέτρων ερωτημάτων που απαιτείται να γίνεται σε μια βάση που ενσωματώνει το PrefDB. Για την ανάπτυξη μιας τέτοιας εφαρμογής, φιλικής προς το χρήστη και ικανής να διευκολύνει τις προαναφερθείσες λειτουργίες, έπρεπε να μελετηθεί εκτενώς το θεωρητικό υπόβαθρο σχετικά με το PrefDB και τα σχεσιακά ερωτήματα με ενσωματωμένες προτιμήσεις χρήστη. Στη συνέχεια, και με οδηγό τα υπάρχοντα συστήματα διαχείρισης βάσεων δεδομένων , έπρεπε να σχεδιαστούν τα σενάρια χρήσης και το διάγραμμα κλάσεων του συστήματος. Τελευταίο στάδιο ήταν η ανάπτυξη της εφαρμογής, ώστε να ανταποκρίνεται στους στόχους που είχαν τεθεί.

1.3 Οργάνωση κειμένου

Στο κεφάλαιο 2 παρουσιάζεται το θεωρητικό υπόβαθρο γενικά για την εξατομίκευση ερωτημάτων σε βάσεις δεδομένων, τις διαφορετικές προσεγγίσεις και αλγορίθμους που χρησιμοποιούνται. Στο κεφάλαιο 3 παρουσιάζεται το μοντέλο PrefDB, αυτόνομο και σε αντιπαράθεση με άλλα μοντέλα για επέκταση του σχεσιακού μοντέλου με προτιμήσεις. Στο κεφάλαιο 4, παρατίθενται τα σενάρια χρήσης του συστήματος και το μοντέλο οντοτήτων-συσχετίσεων (E-R). Στο κεφάλαιο 5, τα θέματα σχεδίασης συστήματος, οι κλάσεις και οι μέθοδοί τους. Στο κεφάλαιο 6, τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος. Στο κεφάλαιο 7 γίνεται έλεγχος, δηλαδή εκτελούνται τα σενάρια χρήσης στο

σύστημα. Στο κεφάλαιο 8 γίνεται μια σύντομη σύνοψη. Τέλος, το κεφάλαιο 9 περιέχει τη σχετική βιβλιογραφία.

2 Θεωρητικό Υπόβαθρο

2.1 Η ανάγκη για εξατομίκευση ερωτημάτων

Η επέκταση του Παγκόσμιου Ιστού στις μέρες μας έχει δώσει εντυπωσιακές δυνατότητες στους χρήστες, παρέχοντάς τους πρόσβαση σε πληθώρα πληροφοριών, αλλά παράλληλα έχει δημιουργήσει και νέες ανάγκες, μια από τις οποίες είναι η ταξινόμηση και η αξιολόγηση των πληροφοριών αυτών. Κάθε χρήστης του Παγκόσμιου Ιστού αναζητά πληροφορίες με διαφορετικά κριτήρια και είναι εξαιρετικά σημαντικό να έχει τη δυνατότητα να βρίσκει άμεσα τις πιο χρήσιμες πληροφορίες.

Ο κάθε χρήστης ελέγχει και αξιολογεί τα αποτελέσματα που παίρνει όταν κάνει μια αναζήτηση με βάση το ποιος ήταν ο αρχικός στόχος του, όχι ο τρόπος που τον δήλωσε. Έτσι, μια αναζήτηση στην ιδανική περίπτωση θα πρέπει να λειτουργεί όχι μόνο κάνοντας επεξεργασία στο ερώτημα του χρήστη, ή τις λέξεις κλειδιά που χρησιμοποίησε, αλλά και με βάση πληροφορίες που έχει ήδη συλλέξει για τον χρήστη. Για παράδειγμα, ένας χρήστης που κάνει αναζήτηση για τη λέξη “magic”, θα πάρει αποτελέσματα σχετικά με το γνωστό παιχνίδι με κάρτες(Magic: The Gathering) , διάφορα τραγούδια, την ομάδα του NBA,τη μαγεία, τα ταχυδακτυλουργικά κόλπα και άλλα θέματα. Βλέποντας όλες αυτές τις πληροφορίες, ο χρήστης πρέπει να κάνει ο ίδιος το φιλτράρισμα για να βρει αυτές που τον ενδιαφέρουν περισσότερο. Αν όμως η μηχανή αναζήτησης μπορεί να χρησιμοποιήσει την πληροφορία πως ο χρήστης πριν από αυτό έκανε αναζήτηση για τις λέξεις “bulls”, “hawks”, “heat” και “nuggets”, ή έχει δοθεί στο χρήστη η επιλογή να δηλώσει τις δραστηριότητες που τον ενδιαφέρουν για τον ελεύθερο χρόνο του, τότε υπάρχει η δυνατότητα να συμπεράνει πως ψάχνει για ομάδες του NBA και να επιστρέψει πρώτα τα σχετικά αποτελέσματα, που είναι πιο πιθανό να ανταποκρίνονται στο σκοπό του χρήστη, και στη συνέχεια όλα τα υπόλοιπα, δηλαδή να φιλτράρει τα αποτελέσματα με βάση τη θεματική περιοχή. Επιπρόσθετα, το

σύστημα θα μπορούσε να δίνει στο χρήστη και αποτελέσματα τα οποία δεν έχουν άμεση σχέση με τη λέξη που αναζήτησε, αλλά με τη θεματική περιοχή του NBA, η οποία φαίνεται να τον ενδιαφέρει, δηλαδή να του κάνει συστάσεις.

Είναι προφανές λοιπόν πως ο κάθε χρήστης όταν εκτελεί ένα ερώτημα σε κάποια βάση δεδομένων πολλές φορές αναμένει διαφορετικά αποτελέσματα ανάλογα με διάφορους παράγοντες, δηλαδή τις προτιμήσεις του, οπότε τα αποτελέσματα που θα του επιστρέψει το σύστημα πρέπει να προσαρμόζονται στις ανάγκες του.

2.2 Δημιουργία προφίλ χρήστη

Όπως αναφέρθηκε, με βάση κάποια δεδομένα που έχει διαθέσιμα για έναν χρήστη, ένα σύστημα έχει τη δυνατότητα να κάνει εξατομίκευση αποτελεσμάτων. Προκύπτουν όμως άμεσα 2 σημαντικά ερωτήματα: Ποιες είναι οι πληροφορίες που έχουν σημασία για να γίνει η εξατομίκευση και πώς το σύστημα θα αποκτήσει αυτές τις πληροφορίες;

Μια πρώτη κατηγορία τέτοιων πληροφοριών, οι οποίες μάλιστα συλλέγονται με την εγγραφή του χρήστη στα περισσότερα συστήματα είναι τα δημογραφικά χαρακτηριστικά του. Η γλώσσα που μιλάει, η ηλικία του, ο τόπος διαμονής του, η εργασία του είναι παράγοντες που μπορεί να επηρεάσουν την παρουσίαση της πληροφορίας στο χρήστη. Παραδείγματος χάριν, αν ένας χρήστης δηλώσει κατά την εγκατάσταση ενός προγράμματος πως η χώρα διαμονής του είναι η Ελλάδα, τότε μπορεί να συνεχίσει αυτόματα η εγκατάσταση στα ελληνικά. Αντίστοιχα, αν ένας χρήστης δηλώσει σε μια ιστοσελίδα όπως το Youtube[1] πως είναι άνω των 18, τότε θα έχει πρόσβαση σε επιπλέον περιεχόμενο, που απαγορεύεται να προσπελαστεί απο ανηλίκους.

Μια ακόμα κατηγορία είναι οι συνθήκες κάτω από τις οποίες ο χρήστης χρησιμοποιεί το σύστημα. Το Google Maps[2] παραδείγματος χάριν μπορεί να εντοπίσει τη θέση του χρήστη, εφόσον μπαίνει από κάποιο κινητό τηλέφωνο, και να τοποθετήσει εκεί την αρχική θέση. Επιπλέον, μπορεί να δείξει διαφορετικές πληροφορίες στον χρήστη που μπαίνει από το κινητό τηλέφωνο σε σχέση με κάποιον που χρησιμοποιεί ηλεκτρονικό υπολογιστή, ανάλογα με το μέγεθος της οθόνης του καθενός.

Τρίτη κατηγορία, και αυτή που έχει άμεση σχέση με το αντικέμενο της διπλωματικής είναι οι προτιμήσεις του χρήστη,οι οποίες μπορεί να αλλάζουν είτε τα δεδομένα που θα

[1] www.youtube.com

[2] maps.google.com

του παρουσιάσει το σύστημα, είτε τη σειρά με την οποία θα του τα παρουσιάσει. Το σύστημα έχει τη δυνατότητα να αξιοποιεί κάποια μέτρα σύγκρισης ανάμεσα στα αποτελέσματα, ώστε να επιστρέφει αυτά που είναι πιθανό να αντιμετωπιστούν θετικά από το χρήστη. Χαρακτηριστικό παράδειγμα είναι οι προτάσεις που κάνει το Amazon[3] στους χρήστες του, με βάση τις σελίδες που έχουν επισκεφτεί και τις αγορές που έχουν κάνει.

Για να δημιουργηθεί το προφίλ του χρήστη όμως, θα πρέπει το σύστημα να συλλέξει και να φιλτράρει τις κατάλληλες πληροφορίες. Αυτό μπορεί να γίνει με διάφορες μεθόδους, ξεκινώντας από την πιο άμεση, την εισαγωγή των δεδομένων από τον ίδιο τον χρήστη. Για αυτή τη μέθοδο, το σύστημα παραπέμπει το χρήστη να εισάγει τις πληροφορίες που χρειάζονται, άλλες προαιρετικές και άλλες υποχρεωτικές, οπότε αυτός μπορεί να δώσει από τον ελάχιστο όγκο πληροφοριών μέχρι και όλες όσες ζητούνται και να πάρει ανάλογα αποτελέσματα στην εξατομίκευση της εξόδου που θα του δώσει το σύστημα. Εναλλακτικά, το σύστημα για να διευκολύνει το χρήστη μπορεί να συνδέσει το προφίλ του χρήστη με αυτό που έχει σε κάποιο άλλο σύστημα και να αντλήσει από εκείνο τις πληροφορίες που χρειάζεται, παραδείγματος χάριν με το Facebook Connect[4], το Google Plus[5].Τέλος, το σύστημα μπορεί να ζητήσει μια σειρά απο αρχικές προτιμήσεις δεδομένων για το χρήστη, και στη συνέχεια για κάθε αποτέλεσμα που επιστρέφει να δέχεται μια βαθμολογία , την οποία θα τη χρησιμοποιεί για να βελτιώνει τα επόμενα αποτελέσματα (active learning, χρησιμοποιείται στο last.fm[6], έναν διαδικτυακό προσωποποιημένο ραδιοφωνικό σταθμό)

Επόμενη μέθοδος είναι η έμμεση εξαγωγή πληροφοριών για τις προτιμήσεις του χρήστη (implicit feedback). Για να το επιτύχει αυτό ένα σύστημα μπορεί να αξιοποιήσει το ιστορικό του χρήστη (όπως το amazon), τη βαθμολογία που δίνει σε διάφορα αποτελέσματα που του παρουσιάζει, κοινά χαρακτηριστικά που εντοπίζονται μεταξύ αυτού και άλλων χρηστών (collaborative filtering, last.fm), τη συχνότητα επισκέψεων, η ακόμα και το χρόνο που ο χρήστης επεξεργάζεται τα δοθέντα αποτελέσματα.

[3] www.amazon.com

[4] www.facebook.com

[5] plus.google.com

[6] www.last.fm

2.3 Αναπαράσταση προτιμήσεων

Οι προτιμήσεις ενός χρήστη, εξ' ορισμού, δεν είναι δυνατόν να εκφράζονται με απόλυτο τρόπο, καθώς σε αυτή την περίπτωση θα εκφυλίζονταν σε αυστηρούς περιορισμούς. Αποτελούν ένα μέτρο σύγκρισης, και ο κύριος άξονας γύρω από τον οποίο κινούνται τα διαφορετικά συστήματα αναπαράστασης είναι το αν η σύγκριση αυτή θα είναι ποιοτική ή ποσοτική. Στα ακόλουθα θα εξεταστούν αυτές οι δύο προσεγγίσεις σε θεωρητικό επίπεδο, ενώ θα δοθεί και ένας μαθηματικός φορμαλισμός με θεωρία συνόλων για να τις περιγράψει (Ο φορμαλισμός από το [SKP11]). Με R(A1,…,An) θα συμβολίζεται το σχεσιακό σχήμα μιας βάσης δεδομένων, όπου A1,..,An τα διαφορετικά attributes και dom(A1),…,dom(An) τα πεδία ορισμού τους. Έστω A = {A1,…,An} το σύνολο attributes του R , dom(A) = dom(A1)x…xdom(An) το σύνολο τιμών του R και r μια πλειάδα του R. Για κάθε υποσύνολο B του A θα ορίζεται t[B] η προβολή της πλειάδας t στο σύνολο χαρακτηριστικών B. Τέλος, το P θα συμβολίζει μια προτίμηση.

2.3.1 Ποιοτική προσέγγιση

Οι ποιοτικές προσεγγίσεις ([Cho03]) εκφράζουν την κάθε προτίμηση ως σύγκριση μεταξύ δύο ή περισσότερων επιλογών π.χ. «Μου αρέσουν τα παιχνίδια ρόλων πιο πολύ από τα παιχνίδια δράσης». Η εφαρμογή στις πλειάδες γίνεται τότε συγκρίνοντας τις αντίστοιχες τιμές και δίνοντας προτεραιότητα στις πλειάδες που έχουν attributes τα οποία έχει δηλώσει ως προτιμητέα σε σχέση με άλλα. Αν ο χρήστης που δήλωσε την προηγούμενη προτίμηση κάνει ερώτηση σε μια βάση με βιντεοπαιχνίδια, το Final Fantasy (παιχνίδι ρόλων) θα εμφανιστεί ψηλότερα από το Tomb Raider (παιχνίδι δράσης). Οι επαγόμενες προτιμήσεις από μια τέτοια ποιοτική προσέγγιση είναι συγκεκριμένες, αφού αν ο ίδιος χρήστης δηλώσει επιπλέον «Μου αρέσουν τα παιχνίδια δράσης πιο πολύ από τα αθλητικά», τότε επάγεται η προτίμηση «Μου αρέσουν τα παιχνίδια ρόλων περισσότερο από τα αθλητικά». Βέβαια, η αποτίμηση όλων των επαγόμενων προτιμήσεων, αλλά και η αναζήτηση των σχετικών με το δοθέν ερώτημα δεν είναι εύκολη, καθώς η εκφραστικότητα μιας τέτοιας προσέγγισης είναι απεριόριστη. Μάλιστα, το πρόβλημα της πλήρους επαγόμενης γνώσης χρησιμοποιώντας κανόνες λογικής είναι NP-complete πρόβλημα [YOK04]. Βελτιώσεις στην πολυπλοκότητα μπορεί να γίνουν περιορίζοντας την εκφραστικότητα των προτιμήσεων, ενώ εναλλακτικά μπορεί να γίνει προσέγγιση του προβλήματος με μη ντετερμινιστικό τρόπο.

Για να οριστεί ο φορμαλισμός για αυτή την προσέγγιση χρησιμοποιείται η έννοια της δυαδικής σχέσης πάνω στο σύνολο S, όπου κάθε τέτοια σχέση B εκφράζει ένα υποσύνολο του γινομένου SxS και για κάθε αϵS, bϵS, θα ισχύει πως αBb αν και μόνο αν το (α,β)ϵB. Επεκτείνοντας τα προηγούμενα, η P μπορεί να δοθεί ως μια δυαδική σχέση >P, η οποία αποτελεί υποσύνολο του dom(A) x dom(A), δηλαδή μπορεί να συγκρίνει οποιεσδήποτε τιμές οποιωνδήποτε χαρακτηριστικών. Αν ri >P rj ,με τα ri,rj να μην παραβιάζουν τους κανόνες συνέπειας της R, τότε το ri είναι προτιμότερο από το rj με βάση την P, ή επικρατεί του rj ως προς την P. Μια τέτοια δυαδική σχέση μπορεί να χαρακτηριστεί με βάση ορισμένες ιδιότητες που έχει:

Α) Είναι κατάταξη αν είναι ανακλαστική και μεταβατική (δηλαδή r >P r και t > P r και r >P s ➔ t >P s). Αν ακόμα είναι αντισυμμετρική (δηλαδή r >P t και t >P r ➔ r == t), τότε είναι μερική κατάταξη.

Β) Είναι αυστηρή μερική κατάταξη αν είναι μη ανακλαστική, μη συμμετρική (δηλαδή r >P t συνεπάγεται πως δεν ισχύει t >P r) και μεταβατική.

Γ) Είναι ολική κατάταξη αν είναι αυστηρή μερική κατάταξη και επιπλέον οποιαδήποτε δύο r και t μπορούν να συγκριθούν με βάση αυτή (δηλαδή ισχύει r >P t ή t >P r ή t == r για δύο οποιαδήποτε έγκυρα r,t).

Δ) Είναι ασθενής κατάταξη αν είναι αυστηρή μερική κατάταξη και επιπλέον είναι αρνητικά μεταβατική (δηλαδή αν το ¬ (r >P t) και ¬ (t >P s) τότε ¬ (r >P s)).

Η προτίμηση ορισμένη με αυτόν τον τρόπο είναι δυνατό να αναπαρασταθεί με έναν κατευθυνόμενο γράφο, όπου θα υπάρχει ακμή από το r στο t αν και μόνο αν r >P t. Επιπλέον, αν η >P είναι μεταβατική, τότε θα χρησιμοποιούμε το μεταβατικό κλείσιμο του γράφου, στο οποίο διαγράφουμε τις περιττές ακμές, δηλαδή αν r >P t και t >P s, τότε δεν υπάρχει ακμή από το r προς το s, αν και r >P s. Αν η προτίμηση αποτελεί και μερική κατάταξη, τότε ο αντίστοιχος γράφος λέγεται διάγραμμα Hasse. Αν δεν κρίνεται σκόπιμο να καταγραφούν όλες οι πιθανές σχέσεις μεταξύ πλειάδων μια προς μια, τότε μπορεί η προτίμηση να οριστεί με βάση μια φόρμουλα PF, η οποία θα είναι αληθής για τα r,t αν και μόνο αν r >P t. Η φόρμουλα αυτή προσδιορίζει την προτίμηση ως εσωτερική ή εξωτερική. Μια εσωτερική προτίμηση αποτιμάται χρησιμοποιώντας μόνο τα χαρακτηριστικά των συγκρινόμενων πλειάδων, ενώ μια εξωτερική απαιτεί τη χρήση επιπλέον πληροφοριών. Αν π.χ. μια προτίμηση μας δίνει την πληροφορία πως ο χρήστης προτιμά τα βιβλία της Agatha Christie από του Conan Doyle, τότε για να συγκριθούν δύο βιβλία αρκεί να ελέγχθει το πεδίο του συγγραφέα. Αν όμως ο χρήστης πει πως προτιμά βιβλία από συγγραφείς που έχουν γράψει περισσότερα βιβλία, τότε θα πρέπει να ανακτηθούν από τη βάση όλες οι πλειάδες που έχουν τις συγκεκριμένες τιμές στο πεδίο συγγραφέα, και να

καταμετρηθούν. Μια εξωτερική προτίμηση δεν είναι ανεξάρτητη από την κατάσταση της βάσης δεδομένων, καθώς μια εισαγωγή ενός νέου βιβλίου μπορεί να αλλάξει τη σειρά προτίμησης στο δοθέν παράδειγμα. Είναι αξιοσημείωτο το γεγονός πως οι κανονικές προτιμήσεις μπορούν να εκφραστούν σε DNF (disjunction normal form, κανονική μορφή διαζεύξεων), δηλαδή ως μια διάζευξη συζεύξεων πάνω στα χαρακτηριστικά των δύο συγκρινόμενων πλειάδων.

2.3.2 Ποσοτική προσέγγιση

Αντίθετα με τα προηγούμενα, οι ποσοτικές προσεγγίσεις ([AW00], [KI04])

εκφράζουν την προτίμηση ως ένα μέτρο αρέσκειας ή δυσαρέσκειας για κάποιες επιλογές π.χ. «Δε μου αρέσουν τα παιχνίδια στρατηγικής» «Απεχθάνομαι τα αθλητικά παιχνίδια». Αυτές οι δύο προτιμήσεις μας δίνουν μια ποσοτική εκτίμηση για το πόσο προτιμά ο χρήστης την επιλογή, με βάση κάποιο χαρακτηριστικό αυτής της επιλογής, δηλαδή ένα συγκεκριμένο attribute (ή περισσότερα από ένα). Η δεύτερη πρόταση εκφράζει λεκτικά μια σαφώς πιο έντονη δυσαρέσκεια, η οποία μπορεί να μεταφραστεί σε έναν αριθμό. Σε αυτές τις προσεγγίσεις, για κάθε πλειάδα που μπορεί να δοθεί ως αποτέλεσμα στο ερώτημα του χρήστη, με βάση όλες τις προτιμήσεις που αναφέρονται σε κάποιο από τα χαρακτηριστικά της πλειάδας, γίνεται μια συνάθροιση των διαφορετικών μέτρων προτίμησης, ώστε να προκύψει μια τελική βαθμολογία για την πλειάδα, η οποία και χρησιμοποιείται ως απόλυτο μέτρο για την ταξινόμηση των αποτελεσμάτων. Υπάρχουν προτιμήσεις οι οποίες είναι αδύνατον να εκφραστούν σωστά με αυτή τη μέθοδο, (αναφορικά αυτές που δεν αποτελούν κατάταξη) όμως καθιστά συχνά εύκολη τη σύγκριση αντικειμένων που έχουν ένα κοινό χαρακτηριστικό και τα υπόλοιπα διαφορετικά. Αν για παράδειγμα ένας χρήστης θέλει να διαλέξει ανάμεσα σε διαφορετικούς τρόπους διασκέδασης για ένα βράδυ, τότε μπορεί με βάση τις προτιμήσεις του να βαθμολογηθούν όλοι οι διαφορετικοί χώροι διασκέδασης, εστιατόρια, κινηματογράφοι, νυχτερινά κέντρα κτλ, χωρίς ο χρήστης να χρειαστεί να έχει δηλώσει ειδικές προτιμήσεις που να τα συγκρίνουν μεταξύ τους.

Για να οριστεί μια προτίμηση με ποσοτικό τρόπο χρησιμοποιείται η έννοια της συνάρτησης βαθμολόγησης f_p , dom(A) → R, η οποία δίνει σε κάθε πλειάδα μια τιμή με βάση την προτίμηση p. Θετικές τιμές ορίζεται να δείχνουν αρέσκεια, αρνητικές δυσαρέσκεια, ενώ μηδενικές αδιαφορία ή αδυναμία εφαρμογής της προτίμησης. Με βάση αυτή την αναπαράσταση, κάθε προτίμηση P μπορεί να εκφραστεί με ένα ζευγάρι (συνθήκηp,βαθμολογίαp), όπου η συνθήκηp καθορίζει τις πλειάδες στις οποίες αναφέρεται η προτίμηση με ένα Conjunctive Normal Form (CNF) , στο οποίο δίνονται τιμές για τα χαρακτηριστικά που πρέπει να έχει η πλειάδα ώστε να επηρεάζεται. Τα

ποιοτικά χαρακτηριστικά έχουν τελεστές = και ≠, ενώ τα ποσοτικά (δηλαδή αυτά που επιδέχονται σύγκρισης) έχουν επιπλέον και τους τελεστές σύγκρισης <,>,≤,≥. Η συνθήκηp μπορεί να είναι για παράδειγμα συνθήκηp1 = (Χρονολογία > 2000) ^ (Διάρκεια < 190) ^ (Σκηνοθέτης = «Κούεντιν Ταραντίνο»). Η βαθμολογία είναι ένας αριθμός από κάποιο προκαθορισμένο εύρος. Για να επεκταθεί αυτός ο τρόπος έκφρασης ώστε να μπορούν να εκφραστούν οι εξωτερικές προτιμήσεις που εξετάστηκαν νωρίτερα με αυτόν, θα πρέπει η συνθήκηp να μπορεί να είναι όχι απλά μια CNF, αλλά μια οποιαδήποτε επιλογή πάνω στη βάση, με τη χρήση joins.

Με αυτή τη μέθοδο κάθε πλειάδα μπορεί να έχει πολλές διαφορετικές βαθμολογίες, μια για κάθε προτίμηση της οποίας τη συνθήκη ικανοποιεί. Η τελική κατάταξη προκύπτει συναθροίζοντας αυτές τις βαθμολογίες. Ο τρόπος με τον οποίο θα γίνει αυτό στη γενική του μορφή οδηγεί σε ένα ακόμα NP-complete πρόβλημα, αυτό της συνολικής κατάταξης m αντικειμένων που υπάρχουν σε n ταξινομημένες λίστες, όπου η καθεμία έχει προκύψει με βάση διαφορετικά κριτήρια. Για να δοθεί μια πιο απλή λύση, χρησιμοποιούνται διάφορες μέθοδοι συνάθροισης, αξιοποιώντας επιπλέον πληροφορίες. Παραδείγματος χάριν, σε κάθε προτίμηση μπορεί να δοθεί ένας συντελεστής εμπιστοσύνης (C), ο οποίος θα χρησιμοποιείται ως βάρος για τη συνάθροιση των διαφορετικών βαθμολογιών. Έτσι, αν σε μια πλειάδα δίνεται βαθμολογία 0.7 από μια προτίμηση με εμπιστοσύνη 1 και βαθμολογία 0.9 από μια προτίμηση με εμπιστοσύνη 0.8, τότε η τελική βαθμολογία θα είναι (1*0.7 + 0.9*0.8)/2 =0.71. Εναλλακτικά , μπορεί να δοθεί μια κατάταξη για τις προτιμήσεις, όπου μόνο η $1^{η}$ προτίμηση που αναφέρεται στην πλειάδα θα έχει σημασία. Μια τρίτη προσέγγιση θα ήταν να συγκρίνονται οι πλειάδες με βάση την κάθε προτίμηση και να βαθμολογούνται υψηλότερα αυτές που υπερτερούν με βάση περισσότερες προτιμήσεις.

2.3.3 Σύγκριση

Συγκρίνοντας τις δύο προσεγγίσεις, το σημαντικότερο ίσως πλεονέκτημα των ποιοτικών προσεγγίσεων είναι ότι οι ποσοτικές δεν επιτρέπουν την έκφραση όλων των ειδών προτιμήσεων. Παραδείγματος χάριν, η προτίμηση: «Από το ρεπερτόριο κάθε τραγουδιστή μου αρέσουν τα πιο γρήγορά του τραγούδια», εκφράζεται ως (song1.singer = song2.singer) ^ (song1.bpm > song2.bpm) σε μια ποιοτική προσέγγιση. Σε μια ποσοτική, το γεγονός ότι για να συγκριθούν τα 2 τραγούδια πρέπει να έχουν τον ίδιο τραγουδιστή, καθιστά αδύνατο το να δοθεί μια βαθμολογία σε κάθε τραγούδι και ταυτόχρονα να ικανοποιείται η πρώτη συνθήκη για τη σύγκρισή τους. Γενικεύοντας, είναι προφανές πως καμία προτίμηση που δεν

εκφράζει μια ασθενή κατάταξη σε αριθμήσιμο πλήθος πλειάδων δεν είναι δυνατόν να αποδοθεί σωστά από μια συνάρτηση βαθμολόγησης [FIS99]. Η ίδια συνθήκη, δηλαδή η προτίμηση να εκφράζει ασθενή κατάταξη, είναι και ικανή ώστε να υπάρχει μια ισοδύναμη συνάρτηση βαθμολόγησης. Αν είναι θεμιτό η συνάρτηση βαθμολόγησης να είναι επαγόμενη αλλά όχι ισοδύναμη, τότε μπορεί η συνθήκη να χαλαρώσει και να αρκεί η προτίμηση να εκφράζεται ακυκλικά με μια ποιοτική προσέγγιση, δηλαδή να μην υπάρχουν t1…tm τέτοια ώστε t1>t2>…>tm>t1 για πεπερασμένο m. Επιπρόσθετα, οι χρήστες συνήθως μπορούν πιο εύκολα να συγκρίνουν δύο χαρακτηριστικά μεταξύ τους, παρά να δώσουν μια ακριβή βαθμολογία για το καθένα.

Στον αντίποδα, οι ποσοτικές προσεγγίσεις έχουν δύο διαφορετικά πλεονεκτήματα. Το πρώτο είναι η ευκολία στην αποτίμηση των βαθμολογιών με χρήση των μεθόδων που αναφέρθηκαν, οι οποίες έχουν πολύ μικρό κόστος υλοποίησης και δίνουν ικανοποιητικά αποτελέσματα. Το δεύτερο είναι πως δε δίνουν απλά μια εποπτική εικόνα του ποια επιλογή είναι προτιμότερη αλλά και κατά πόσο είναι προτιμότερη. Έτσι είναι εφικτό να τεθεί μια ελάχιστη βαθμολογία και τα αποτελέσματα που βαθμολογούνται χαμηλότερα να μην επιστρέφονται στο χρήστη, ακόμα και αν δεν βρεθούν άλλα καλύτερα.

Δεν υπάρχει λοιπόν ένας σαφώς καλύτερος τρόπος προσέγγισης, αλλά η κάθε εφαρμογή που θα αξιοποιήσει ένα τέτοιο σύστημα πρέπει να εξετάζει ποια από τα πλεονεκτήματα και μειονεκτήματα που παρατέθηκαν την επηρεάζουν περισσότερο.

2.4 Επιλογή σχετικών προτιμήσεων

Από όλες τις διαθέσιμες προτιμήσεις που υπάρχουν σε ένα σύστημα με βάση το προφίλ χρήστη, θα πρέπει πολλές φορές να μη χρησιμοποιούνται όλες, αλλά μόνο συγκεκριμένες, οι οποίες ανταποκρίνονται σε κάποιες συνθήκες, εσωτερικές ή εξωτερικές σε σχέση με τη βάση δεδομένων. Δημιουργείται με βάση αυτές τις συνθήκες ένα περιβάλλον χρήσης, μέσα στο οποίο πρέπει να αποφασιστεί ποιες προτιμήσεις θα χρησιμοποιηθούν σε ένα ερώτημα.

Παράδειγμα εσωτερικής παραμέτρου για ένα σύστημα που προτείνει ταινίες στο χρήστη θα μπορούσε να είναι ο αριθμός των βαθμολογιών που έχουμε για μια ταινία ώστε να ληφθεί ή όχι αυτή η βαθμολογία υπόψιν ως κριτήριο. Δηλαδή, ο χρήστης μπορεί να δηλώσει «Προτιμώ τις ταινίες με βαθμολογία μεγαλύτερη του 8, μόνο αν αυτή έχει προκύψει ως αποτέλεσμα τουλάχιστον 100 διαφορετικών βαθμολογήσεων». Βέβαια, πολλές φορές τέτοιου είδους εσωτερικές παράμετροι μπορούν να εκφραστούν και ως μέρος των προτιμήσεων, όμως αυτό δεν είναι πάντα εφικτό, ανάλογα και με την

εκφραστικότητα της αναπαράστασης που χρησιμοποιείται για τις προτιμήσεις. Εξάλλου, κάποιες φορές είναι προτιμότερο να ορίζεται μια εσωτερική παράμετρος σε χαρακτηριστικό, παρά να ενσωματώνεται σε όλες τις προτιμήσεις. Η προηγούμενη παράμετρος περιβάλλοντος θα μπορούσε να είναι πιο γενική: «Οι προτιμήσεις που έχουν να κάνουν με βαθμολογία θα λαμβάνονται υπόψιν μόνο στις ταινίες για τις οποίες υπάρχουν τουλάχιστον 100 βαθμολογίες».

Αντίθετα, μια εξωτερική παράμετρος σχετίζεται με το ευρύτερο περιβάλλον του χρήστη (context). Μπορεί μια προτίμηση να εξαρτάται από τον χρόνο «Μου αρέσει να βλέπω ταινίες τρόμου μετά τα μεσάνυχτα», τον τόπο «Θέλω να πάω στο πλησιέστερο ιταλικό εστιατόριο», τη διάθεση «Όταν είμαι χαρούμενος μου αρέσει να βλέπω μιούζικαλ» και διάφορους άλλους παράγοντες που δε σχετίζονται με το σχήμα τις βάσης δεδομένων, αλλά πρέπει να θεωρηθούν ως ανεξάρτητη είσοδος για το σύστημα, ώστε να αποφασιστεί αν οι προτιμήσεις που εξαρτώνται από τέτοιους παράγοντες θα χρησιμοποιηθούν ή όχι για την αποτίμηση του ερωτήματος.

2.5 Ενσωμάτωση προτιμήσεων σε ερωτήματα βάσεων δεδομένων

Εξετάστηκαν μέχρι στιγμής γενικές περιπτώσεις στις οποίες ένα σύστημα «μαύρο κουτί» είναι αυτό που επεξεργάζεται το προφίλ του χρήστη και απαντά στα ερωτήματά του με βάση αυτό. Εστιάζοντας στην περίπτωση των βάσεων δεδομένων, είναι δυνατόν να εκφραστεί πιο λεπτομερώς η αξία της προσέγγισης εξατομίκευσης ερωτημάτων χρησιμοποιώντας προτιμήσεις. Σε μια βάση δεδομένων, κάθε ερώτημα θέτει κάποιους αυστηρούς περιορισμούς με βάση τους οποίους επιστρέφονται συγκεκριμένες, αμετάβλητες πλειάδες στο χρήστη.

Εν γένει προκύπτουν δύο πιθανώς παθολογικά σενάρια [SKP11]: α) Τι γίνεται όταν αυτοί οι περιορισμοί είναι τόσο αυστηροί ώστε να μην επιστρέφεται κανένα αποτέλεσμα; Είναι θεμιτό να δίνεται ως απάντηση το κενό σύνολο στο χρήστη; β) Αντίθετα, αν οι περιορισμοί είναι υπερβολικά χαλαροί, έχει ο χρήστης τη δυνατότητα να ξεδιαλέξει τα αποτελέσματα που του είναι πιο χρήσιμα, ή έχουμε την περίπτωση των υπερβολικά πολλών απαντήσεων;

Η ενσωμάτωση προτιμήσεων στα ερωτήματα προς τη βάση δεδομένων μπορεί να μας επιτρέψει να παρακάμψουμε και τα 2 αυτά σενάρια. Στην πρώτη περίπτωση, αρκεί να

χαλαρώσουν οι περιορισμοί που τίθενται από το ερώτημα , μετατρέποντας κάποιους απ' αυτούς σε προτιμήσεις, ώστε ναι μεν να λαμβάνονται υπόψιν, αλλά, αν δε μπορούν να ικανοποιηθούν όλοι ταυτόχρονα, τότε ένας προς έναν (με μη τετριμμένη σειρά) να παραγκωνίζονται ώστε να βρεθούν πλειάδες που είναι κοντά στο να απαντούν το αρχικό ερώτημα. Ανάλογα με το πόσο σημαντικοί είναι οι διάφοροι περιορισμοί, αυτή η διαδικασία θα πρέπει να σταματάει όταν είτε έχουμε ικανοποιητικό αριθμό αποτελεσμάτων, ή κρίνεται μη ωφέλιμο να χαλαρώσουν περαιτέρω οι περιορισμοί.. Στη δεύτερη περίπτωση, είναι δυνατόν να προστεθούν περιορισμοί στο ερώτημα, έτσι ώστε να επιστραφούν στο χρήστη λιγότερες πλειάδες, ή να του επιστρέφει αυτές που είναι πιο πιθανό να ανταποκρίνονται στις προτιμήσεις του. Οι περιορισμοί αυτοί θα πρέπει να μπαίνουν με τέτοιο τρόπο ώστε να μην αλλοιώνεται το αρχικό ερώτημα του χρήστη τόσο ώστε να χαθούν δυνάμει χρήσιμες πληροφορίες.

2.6 Αλγόριθμοι αποτίμησης ερωτημάτων προτιμήσεων

Σημαντικό είναι να προσδιορίστεί ο τρόπος με τον οποίο θα γίνεται η τελική κατάταξη των αποτελεσμάτων αφού γίνει η επιμέρους ταξινόμησή τους με βάση τις διάφορες προτιμήσεις. Δύο διαφορετικές κλάσεις αλγορίθμων που αντιμετωπίζουν αυτό το πρόβλημα είναι οι top-k αλγόριθμοι και οι αλγόριθμοι για ερωτήματα κορυφογραμμής. Οι πρώτοι χρησιμοποιούν μια συνάρτηση συνάθροισης για να επιστρέψουν στο χρήστη τα αποτελέσματα με την υψηλότερη βαθμολογία, ενώ οι δεύτεροι αποκλείουν τα μη βέλτιστα αποτελέσματα έτσι ώστε όταν ένα αποτέλεσμα Α δεν επιστρέφεται, υπάρχει τουλάχιστον ένα αποτέλεσμα Β που επιστρέφεται και είναι καλύτερο από το Α ως προς όλες τις προτιμήσεις.

2.6.1 Ερωτήματα Top-k

Στα ερωτήματα top-k δίνονται ως δεδομένο m ταξινομημένες λίστες για n αντικείμενα και το ζητούμενο είναι να επιλεγούν ανάμεσα σε αυτά τα k καλύτερα. Ο προφανής τρόπος για να γίνει αυτό, δηλαδή να υπολογιστεί η τελική βαθμολογία για καθένα από αυτά με χρήση συναρτήσεων συνάθροισης και να γίνει η τελική κατάταξη είναι μεν σωστός, αλλά σίγουρα όχι αποδοτικός, Θεωρώντας πως η προσπέλαση στις ταξινομημένες λίστες μπορεί να γίνει είτε σειριακά ώστε να ακολουθείται η κατάταξη είτε με τυχαία προσπέλαση ώστε να είναι δυνατόν να

ανακτηθεί για οποιοδήποτε στοιχείο η βαθμολογία του από οποιαδήποτε λίστα, προτείνονται οι εξής αλγόριθμοι [BW07]:

- Ο αλγόριθμος FA προσπελαύνει σειριακά αντικείμενα έως ότου να βρει k αντικείμενα που να έχουν εμφανιστεί σε όλες τις λίστες και έπειτα υπολογίζει την τελική βαθμολογία για αυτά και όλα τα άλλα που έχουν εμφανιστεί μέχρι εκείνη τη στιγμή και εξάγει τα k καλύτερα από αυτά. Ο αλγόριθμος αυτός λειτουργεί διότι αν κάποιο αντικείμενο δεν έχει βρεθεί σε καμία από τις λίστες μέχρι αυτό το σημείο, τότε σίγουρα δεν είναι στα top-k για κάποια από αυτές και άρα δε μπορεί να είναι στα συνολικά top-k.
- Βελτιώνοντας αυτό τον αλγόριθμο, προκύπτει ο αλγόριθμος TA (threshold algorithm), ο οποίος κάνει σειριακή προσπέλαση στις λίστες και, για κάθε νέο αντικείμενο που συναντά υπολογίζει με τυχαίες προσπελάσεις τη βαθμολογία του. Αν αυτή η βαθμολογία είναι στις k υψηλότερες μέχρι αυτό το σημείο, τότε διατηρείται το αντικείμενο και οι βαθμολογίες του. Από τα m αντικείμενα που προσπελαύνει σε κάθε βήμα ο αλγόριθμος δημιουργεί ένα κατώφλι, το οποίο προκύπτει δίνοντας ως ορίσματα στη συνάρτηση συνάθροισης τις τιμές s_1^L, … , s_m^L που συνάντησε στην κάθε λίστα για αυτά τα αντικείμενα Αν τώρα το αντικείμενο με τη μικρότερη βαθμολογία στο top-k έχει βαθμολογία κάτω από αυτό το κατώφλι, ο αλγόριθμος συνεχίζει, αλλιώς έχει τελειώσει και έχουν βρεθεί τα top-k αποτελέσματα. Ο αλγόριθμος αυτός στηρίζεται στο ότι, αφού οι λίστες είναι ταξινομημένες, το κατώφλι είναι η βέλτιστη τιμή που μπορεί να έχει ένα αντικείμενο που δεν έχει εξεταστεί ακόμα. Αν αυτή η βαθμολογία επιτευχθεί ή ξεπεραστεί από όλα τα top-k αντικείμενα που προκύπτουν σε μια επανάληψη του αλγορίθμου, τότε δεν υπάρχει λόγος να εξεταστούν τα υπόλοιπα.
- Αν κάποια από τις λίστες δεν επιτρέπει σειριακή ταξινομημένη πρόσβαση, τότε ο προηγούμενος αλγόριθμος μπορεί να λειτουργήσει, όμως θα πρέπει για αυτές τις λίστες να χρησιμοποιείται η μέγιστη δυνατή τιμή για τον υπολογισμό του κατωφλίου.
- Αν αντίθετα δεν υπάρχει δυνατότητα τυχαίας πρόσβασης , τότε ο TA μετατρέπεται στον NRA, όπου για κάθε νέο αντικείμενο υπολογίζεται ένα κατώτατο όριο (με τιμή 0 όπου υπάρχει άγνωστη

τιμή) και ένα ανώτατο (με τιμή ίση με την τιμή που είχε στην τρέχουσα επανάληψη η λίστα, δηλαδή τη μέγιστη δυνατή). Διατηρούνται τα k αντικείμενα με τα μεγαλύτερα κατώτατα όρια, και ο αλγόριθμος τερματίζει όταν όλα αντικείμενα εκτός από τα top-k, έχουν ανώτατα όρια μικρότερα ή ίσα από το μικρότερο κατώτατο όριο στο top-k.

- Για να εφαρμοστεί ο TA σε έναν βελτιστοποιητή ερωτημάτων για μια βάση δεδομένων, υπάρχουν ορισμένα επιπλέον προβλήματα. Οι αρχικές λίστες δεν υπάρχουν διαθέσιμες στη βάση , οπότε θα πρέπει να περιοριστούν οι επιμέρους συναρτήσεις βαθμολόγησης, έτσι ώστε να μπορούν να αποδοθούν αποδοτικά με τη χρήση δεικτών. Για παράδειγμα, η συνάρτηση 3*c1, όπου c1 κάποιο χαρακτηριστικό που περιλαμβάνεται στο ερώτημα, μπορεί να αποδοθεί με ένα δείκτη που έχει τα στοιχεία ταξινομημένα ως προς c1. Η συνάρτηση (0.5 – c2) ^ 2 μπορεί να αποδοθεί με 2 δείκτες, έναν που θα ξεκινάει από τα στοιχεία με c2 = 0.5 και θα «ανεβαίνει» ως προς τις τιμές του c2, και έναν που θα «κατεβαίνει» αντίστοιχα, οπότε θα επιστρέφεται κάθε φορά η τιμή του δείκτη που είναι πιο κοντά στο 0.5. Αυτό βέβαια δεν είναι εφικτό για οποιαδήποτε συνάρτηση, οπότε πρέπει να ελέγχονται κατάλληλα οι συναρτήσεις βαθμολόγησης που θα χρησιμοποιούνται. Η τυχαία προσπέλαση που απαιτείται στη συνέχεια για να πάρουμε τις υπόλοιπες τιμές για την πλειάδα που θα επιστραφεί , γίνεται με το πολύ μια πρόσβαση στον κύριο δείκτη (ή και καμία αν ο δείκτης που χρησιμοποιούμε περιλαμβάνει ήδη τα ζητούμενα χαρακτηριστικά). Αξιοποιώντας αυτές τις μεθόδους και αφού επιλεγούν οι κατάλληλοι δείκτες, ο αλγόριθμος εφαρμόζεται κατά τα γνωστά.

2.6.2 Ερωτήματα κορυφογραμμής

Στα ερωτήματα κορυφογραμμής, πρέπει να γίνει η επιλογή αυτών των στοιχείων για τα οποία δεν υπάρχουν στοιχεία που να κυριαρχούν πάνω τους για όλες τις προτιμήσεις, όπου κυριαρχία ως προς μια προτίμηση σημαίνει πως η συγκεκριμένη προτίμηση αποδίδει υψηλότερη βαθμολογία στην πλειάδα που κυριαρχεί. Για παράδειγμα , έστω πως ένας χρήστης προτιμά τα ακριβά και γρήγορα αυτοκίνητα, και έχει τα Α,Β,Γ ως επιλογές. Αν το Α είναι πιο γρήγορο και πιο ακριβό απο τα Β

και Γ, το Β πιο γρήγορο από το Γ και το Γ πιο ακριβό από το Β. Σε αυτή την περίπτωση το Α είναι το μόνο στοιχείο της κορυφογραμμής, διότι αν μπει στην κορυφογραμμή το Β τότε υπάρχει εκτός το Γ, το οποίο υπερτερεί του Β σε ένα από τα κριτήρια, ενώ αν μπει το Γ, υπάρχει εκτός το Β, το οποίο υπερτερεί του Γ σε ένα από τα κριτήρια. Το να γίνουν nested loops για την αποτίμηση αυτών των ερωτημάτων, συγκρίνοντας σε ζεύγη τα στοιχεία σε σχέση με την καθεμία από τις προτιμήσεις είναι ο προφανής τρόπος και έχει αποδοτικότητα O(n^2), κάτι που δεν είναι επιθυμητό. Θα εξεταστούν και πάλι ορισμένοι δημοφιλείς αλγόριθμοι που προσπαθούν να το βελτιώσουν [BKS01], CGG+03].

- Ο αλγόριθμος Block Nested Loops (BNL) [BKS01] ξεκινάει με έναν κενό ορίζοντα. Για κάθε στοιχείο που συναντά ως είσοδο, υπάρχουν τρεις περιπτώσεις. Αν δεν υπερτερεί ούτε υστερεί σε σχέση με τα ήδη υπάρχοντα, τότε εισάγεται στον ορίζοντα. Αν υπερτερεί σε σχέση με κάποια από τα υπάρχοντα στοιχεία, τότε αυτά απορρίπτονται και το νέο τα αντικαθιστά στον ορίζοντα. Αν υστερεί σε σχέση με κάποιο από τα υπάρχοντα, τότε απορρίπτεται. Στην καλύτερη περίπτωση γίνεται μια σύγκριση για κάθε στοιχείο και η πολυπλοκότητα είναι O(n), ενώ στη χειρότερη δεν υπάρχει βελτίωση σε σχέση με το απλό nested loops.
- Ο Sort-Filter-Skyline (SFS) [CGG+03] αποτελεί βελτίωση του προηγούμενου αλγόριθμου, όπου γίνεται πρώτα μια τοπολογική ταξινόμηση των στοιχείων σε σχέση με τη ζητούμενη κορυφογραμμή. Με αυτό τον τρόπο, όταν ένα στοιχείο καταλήγει στον ορίζοντα σε μια επανάληψη, τότε δε μπορεί να εκτοπιστεί από κάποιο σε επόμενη επανάληψη, οπότε ανήκει αυτόματα στην κορυφογραμμή. Με μια κατάλληλη συνάρτηση ταξινόμησης, αυτός ο αλγόριθμος μπορεί να δώσει καλύτερα αποτελέσματα, ενώ σίγουρα επιτρέπει την εύρεση της κορυφογραμμής με τον ελάχιστο αριθμό περασμάτων. Επιπλέον, μπορεί να επιστρέφει σταδιακά αποτελέσματα κατά τη διάρκεια της εκτέλεσής του, καθώς, όπως αναφέρθηκε, αν ένα στοιχείο καταλήξει στον ορίζοντα είναι αδύνατον να εκτοπιστεί.
- Ο αλγόριθμος διαίρει και βασίλευε (DC) [BKS01], στην απλή του περίπτωση, επιχειρεί να κάνει την επίλυση του προβλήματος αποδοτική για μεγάλες βάσεις δεδομένων. Βρίσκεται μια «μέση τιμή» για κάποια από τις προτιμήσεις, και το σύνολο των στοιχείων χωρίζεται σε δύο υποσύνολα μέσω της σύγκρισης με αυτή την τιμή, ένα καλύτερο και ένα χειρότερο. Αναδρομικά υπολογίζεται η κορυφογραμμή για τα υποσύνολα, μέχρι να έχουμε τέτοιο αριθμό στοιχείων σε κάθε σύνολο ώστε το πρόβλημα να είναι

τετριμμένο. Από κατασκευής, στη σύνθεση των υποσυνόλων, δεν υπάρχει περίπτωση κάποιο στοιχείο από το «χειρότερο» υποσύνολο να κυριαρχεί σε κάποιο από το «καλύτερο». Εντοπίζονται λοιπόν τα στοιχεία του «καλύτερου» που κυριαρχούν σε στοιχεία του «χειρότερου» και τα τελευταία απορρίπτονται. Η πολυπλοκότητα εδώ είναι ίση με $O(n * (\log n)^{(d-2)}) + O(n * \log n)$, όπου d το πλήθος των προτιμήσεων που θα πρέπει να εξεταστούν. Η βελτίωση σε σχέση με τον Block Nested Loops υφίσταται μόνο στις κακές περιπτώσεις, ενώ στις καλές ο απλός διαίρει και βασίλευε είναι χειρότερος.

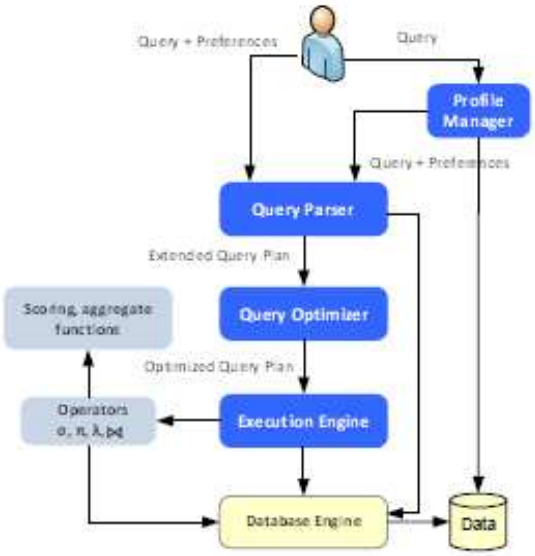
- Ο αλγόριθμος Branch and Bound [BBS] δέχεται ([PTFS03]) ως είσοδο ένα R-tree και, ξεκινώντας αρχικά με μια κενή κορυφογραμμή και ένα σωρό με τα στοιχεία της ρίζας, ταξινομημένα με βάση την ελάχιστη απόσταση από κάποιο μηδενικό σημείο-διάνυσμα, επεκτείνει σε κάθε βήμα μόνο αυτούς τους κόμβους στους οποίους δεν κυριαρχεί κανένα από τα στοιχεία της κορυφογραμμής, ενώ απορρίπτει τα παιδιά τους στα οποία κυριαρχεί κάποιο στοιχείο της κορυφογραμμής. Αν φτάσει σε φύλλο, τότε αυτό εισάγεται στην κορυφογραμμή.

3 Σχετικές Εργασίες

Ο χώρος των βάσεων δεδομένων με χρήση προτιμήσεων αποτελεί ένα σχετικά νέο πεδίο στην επιστήμη των βάσεων δεδομένων, με πολλά πεδία εφαρμογής, με τα πιο σημαντικά ίσως να είναι οι μηχανές αναζήτησης και τα συστήματα προτάσεων. Διάφορα συστήματα έχουν αναπτυχθεί, υλοποιώντας το καθένα κάποιο διαφορετικό από τα μοντέλα που εξετάστηκαν για την αναπαράσταση των προτιμήσεων και την αποτίμηση των ερωτημάτων. Θα εξεταστεί αρχικά το μοντέλο που χρησιμοποιείται στην αναπτυσσόμενη εφαρμογή, και στη συνέχεια ορισμένα άλλα ενδεικτικά μοντέλα και εφαρμογές, με κύριο άξονα τα σημεία στα οποία διαφοροποιούνται οι υλοποιήσεις μεταξύ τους.

3.1 Εξατομικευμένο σχεσιακό μοντέλο με προτιμήσεις PrefDB

Το μοντέλο που υλοποιείται στα πλαίσια αυτής της εργασίας ονομάζεται PrefDB και έχει ως κύριο στόχο του την ενσωμάτωση των ερωτημάτων με προτιμήσεις στη βάση δεδομένων, τη μοντελοποίηση ορισμένων παραμέτρων που δεν είχαν υλοποιηθεί σε άλλα παρόμοια συστήματα και τη βελτιστοποίηση της εκτέλεσης αυτών των ερωτημάτων. Ακολουθεί ποσοτική προσέγγιση για την κατάταξη των αποτελεσμάτων, δηλαδή με βάση το ερώτημα και τις προτιμήσεις, δίνει μια βαθμολογία σε κάθε πλειάδα και έτσι επιστρέφει τις ζητούμενες καλύτερες με βάση αυτή τη βαθμολογία. Ως μια γενική εικόνα του πώς εκτελούνται τα ερωτήματα με βάση το PrefDB, παρατίθεται το ακόλουθο σχήμα:



Ο χρήστης δίνει μέσω ενός interface το ερώτημα και τις προτιμήσεις που θέλει να χρησιμοποιηθούν σε αυτό, τα οποία περνάνε σε έναν query parser που ενσωματώνει στο ερώτημα τις προτιμήσεις και δημιουργεί το αρχικό πλάνο για την εκτέλεση του ερωτήματος. Αυτό στη συνέχεια βελτιώνεται με τη βοήθεια του query optimizer και του plan evaluator. Τέλος , για κάθε βήμα του τελικού πλάνου, χρησιμοποιείται ο αντίστοιχος τελεστής (και αν είναι απαραίτητο οι επιλεγμένες συναρτήσεις βαθμολόγησης, σύγκρισης και συνάθροισης), ώστε να επιστραφούν απευθείας τα δεδομένα που αντιστοιχούν καλύτερα στις προτιμήσεις του χρήστη. Παρουσιάζεται στη συνέχεια το μοντέλο PrefDB με βάση την εργασία [AK12].

3.1.1 Ορισμοί

Πριν εξεταστούν οι λεπτομέρειες της υλοποίησης του PrefDB, παρατίθενται οι ορισμοί και συμβολισμοί που χρησιμοποιούνται για τη θεωρητική ανάλυση του μοντέλου.

Έστω R = {A1,A2,…,An} ένα σχεσιακό σχήμα και r μια πλειάδα αυτού του σχήματος. Θα συμβολίζεται με r.Aj η τιμή του r για την ιδιότητα Aj. Επίσης, το πεδίο τιμών του Aj θα συμβολίζεται με dom(Aj).

Ως συνάρτηση αξιολόγησης ορίζεται μια συνάρτηση f: dom(A1)xdom(A2)x…xdom(Ak)→[0,1] . Παραδείγματος χάριν, η συνάρτηση fr(x1) = 0.1* x1 δέχεται τιμές από το 1 έως το 10 (τη βαθμολογία για κάποια ταινία), και επιστρέφει μια τιμή στο διάστημα [0,1]. Γενικά τα ορίσματα των συναρτήσεων αξιολόγησης είναι attributes του σχεσιακού σχήματος.

Ως ατομική προτίμηση ορίζεται μια διατεταγμένη τριάδα p[R] = (r,S,C) , όπου το S είναι είτε μια τιμή στο [0,1] ή null. Αν η τιμή του ανήκει στο δοθέν διάστημα τότε είναι η βαθμολογία

που δίνεται στην πλειάδα R από την προτίμηση p[R], ενώ αν είναι null, τότε η προτίμηση αυτή δεν αντιστοιχεί στην πλειάδα r καθόλου, δεν τη βαθμολογεί. Το C είναι ένα ένας θετικός πραγματικός αριθμός που εκφράζει την εμπιστοσύνη που δίνει η προτίμηση p[R] για τη συγκεκριμένη βαθμολογία, και είναι ένα μέτρο του πόσο σίγουρο είναι το σύστημα πως αυτή η βαθμολογία ανταποκρίνεται στην πραγματικότητα, με βάση τα όσα ξέρει για τον χρήστη. Η προτίμηση λοιπόν για κάθε πλειάδα r δίνει μια βαθμολογία S με εμπιστοσύνη C. Ως γενική προτίμηση, δηλαδή προτίμηση η οποία περιλαμβάνει ένα σύνολο από πλειάδες στις οποίες εφαρμόζεται, θα ορίζεται η p[R] = (σ,f,C), όπου το σ είναι μια συνθήκη επιλογής σε κάποια attributes , που επιστρέφει τις πλειάδες στις οποίες έχει εφαρμογή η προτίμηση p[R], το f μια συνάρτηση βαθμολόγησης από το σύνολο των attributes που επιλέγει η σ στο [0,1]U⊥ και το C η εμπιστοσύνη όπως και πριν. Μια πλειάδα r1 προτιμάται από μια άλλη r2 ή επικρατεί πάνω της όταν το S1 > S2. Το C είναι ένα μέτρο της βεβαιότητας με την οποία το σύστημα καταλήγει σε αυτό το συμπέρασμα.

Με βάση αυτό τον ορισμό γίνεται και η επέκταση της έννοιας του σχεσιακού σχήματος R. Σχεσιακό σχήμα με προτιμήσεις, Rp θα λέγεται ένα σχεσιακό σχήμα εμπλουτισμένο με τη βαθμολογία S και την εμπιστοσύνη C, με αρχικές τιμές ⊥ και 0 αντίστοιχα. Δηλαδή, Rp = {(r,S,C)|r□R, S = ⊥, C = 0}. Στο εξής, για την Rp θα χρησιμοποιείται το σύμβολο R. Η βαθμολόγηση μιας πλειάδας μπορεί να γίνει με αποτίμηση των προτιμήσεων του χρήστη. Αν τώρα υπάρχουν περισσότερες από μια προτιμήσεις που βαθμολογούν την ίδια πλειάδα, προκύπτουν διαφορετικά ζεύγη βαθμολογίας-εμπιστοσύνης για αυτή. Ο ποσοτικός χαρακτήρας του PrefDB απαιτεί να μπορεί να γίνει μια συνολική ταξινόμηση, άρα είναι απαραίτητη η συνάθροιση των αποτελεσμάτων. Η συνάρτηση συνάθροισης ορίζεται ως F: (S,C)*(S,C) → (S,C), παίρνει ως όρισμα δηλαδή δύο ζεύγη βαθμολογία-εμπιστοσύνη και τα συναθροίζει σε ένα νέο. Οι συναρτήσεις F που επιλέγονται, θα πρέπει να είναι προσεταιριστικές, καθώς δεν είναι θεμιτό η σειρά της συνάθροισης των επιμέρως αποτελεσμάτων να επηρρεάζει το τελικό. Παραδείγματα τέτοιων συναθροιστικών συναρτήσεων είναι το maximum, το άθροισμα κ.α.

3.1.2 Τελεστές

σ_{φ} (R) → Τελεστής επιλογής: Όπως και στο απλό σχεσιακό μοντέλο, επιλέγει τις πλειάδες που ικανοποιούν τη συνθήκη φ, μόνο που αυτή μπορεί να περιλαμβάνει δύο επιπλέον κριτήρια, τη βαθμολογία και την εμπιστοσύνη.

$\pi_{A1,...,Ak}$ (R) → Τελεστής προβολής: Όπως στο απλό σχεσιακό μοντέλο, κρατάει τα επιλεγμένα attributes από τις πλειάδες στις οποίες εφαρμόζεται. Επιπλέον διατηρεί τη βαθμολογία και την εμπιστοσύνη.

Ri $\cap_F$ Rj → Τελεστής τομής: Όπως στο απλό σχεσιακό μοντέλο, επιστρέφει τις πλειάδες που ανήκουν και στις δύο σχέσεις. Για να υπολογιστεί η βαθμολογία και η εμπιστοσύνη στις πλειάδες αυτές, χρησιμοποιείται η συνάρτηση συνάθροισης F.

Ri U_F Rj → Τελεστής ένωσης: Όπως στο απλό σχεσιακό μοντέλο, επιστρέφει τις πλειάδες που ανήκουν σε τουλάχιστον μια από τις δύο σχέσεις. Για τις πλειάδες που υπάρχουν και στις δύο σχέσεις, εφαρμόζεται η συνάρτηση συνάθροισης F για να υπολογιστούν η βαθμολογία και η εμπιστοσύνη τους.

Ri – Rj → Τελεστής διαφοράς: Περιλαμβάνει τις πλειάδες που ανήκουν στο Ri, αλλά όχι στο Rj.

Ri $\overset{?}{\cup}_{\varphi,F}$ Rj → Τελεστής εσωτερικής συνένωσης: Προκύπτουν οι πλειάδες όπως θα προέκυπταν σε μια κανονική συνένωση με βάση τη συνθήκη φ, ενώ οι βαθμολογία και η εμπιστοσύνη για καθεμιά τους προκύπτουν με τη συνάρτηση συνάθροισης F.

$\lambda_{p,F}$ (R) → Τελεστής προτίμησης. Νέος τελεστής. Όταν εφαρμόζεται στο R, τότε για τις πλειάδες που περιλαμβάνονται στη συνθήκη επιλογής της προτίμησης p, υπολογίζονται νέες τιμές για τη βαθμολογία και την εμπιστοσύνη. Οι υπάρχουσες τιμές συναθροίζονται μέσω της F με τιμές που αποδίδει η προτίμηση p. Τις πλειάδες που δεν περιλαμβάνονται στη συνθήκη επιλογής της προτίμησης p, ο τελεστής τις αφήνει ανέπαφες. Αν για παράδειγμα έχουμε την πρτίμηση p(age>25, 0.7,0.3) και μια πλειάδα r με (name, age, score, conf) = (John, 24, 0.9, 0.4) , τότε $\lambda_{p,F}$ (r) = r

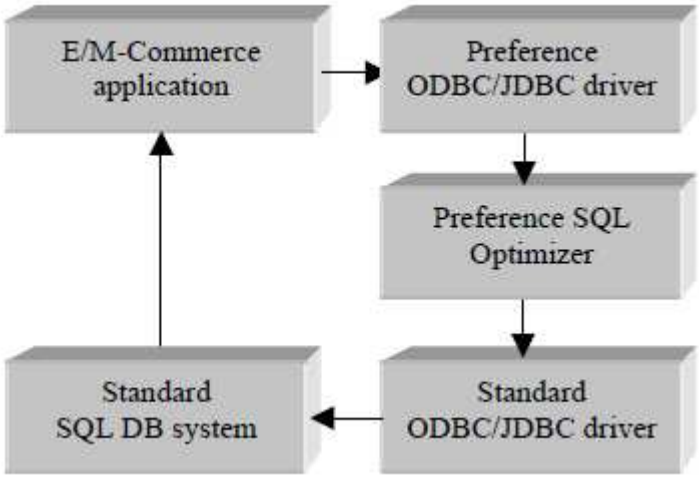
3.1.3 Ερωτήματα με προτιμήσεις στο PrefDB

Χρησιμοποιώντας τον τελεστή προτίμησης, καθώς και τους νέους ορισμούς για τους υπόλοιπους τελεστές, μπορούν τώρα να γραφτούν ερωτήματα με προτιμήσεις. Τα ερωτήματα αυτά επιστρέφουν τις πλειάδες που ανταποκρίνονται στις αυστηρές συνθήκες του ερωτήματος, μαζί με τη βαθμολογία και την εμπιστοσύνη που προκύπτουν από την εφαρμογή του τελεστή προτίμησης. Η ταξινόμηση αυτών των αποτελεσμάτων μπορεί να γίνει με πολλούς διαφορετικούς τρόπους, πέραν αυτών που θα ήταν διαθέσιμοι σε ένα απλό σχεσιακό σύστημα: Ανάλογα με τη βαθμολογία, ανάλογα με την εμπιστοσύνη, ανάλογα με το πόσες προτιμήσεις ικανοποιούσαν.

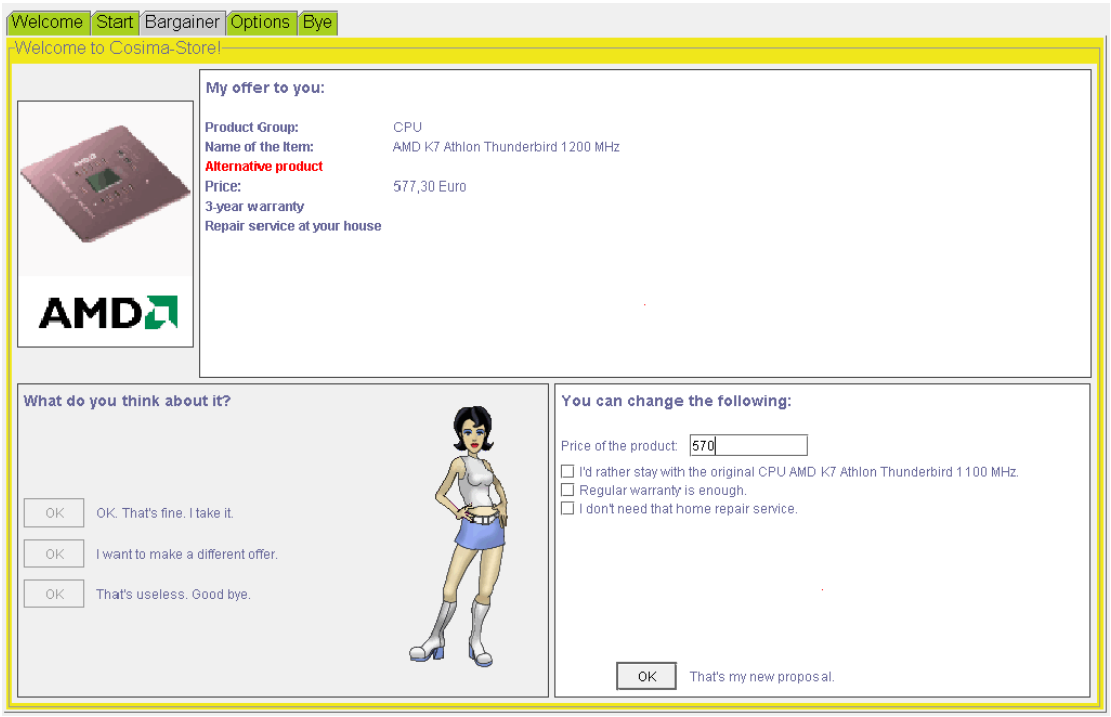
Επιπλέον των τελεστών , μπορούν να χρησιμοποιηθούν και παράμετροι για την εκτέλεση του ερωτήματος, ώστε να προσδιοριστούν περαιτέρω τα επιθυμητά αποτελέσματα. Οι παράμετροι αυτοί είναι: Ελάχιστη εμπιστοσύνη(ανάμεσα σε 0 και το πλήθος των προτιμήσεων) , ελάχιστη βαθμολογία(από 0 έως 1), αλγόριθμος ταξινόμησης, μέθοδος επιλογής προτιμήσεων, τρόπος ταξινόμησης αποτελεσμάτων(με βάση τη βαθμολογία, με βάση την εμπιστοσύνη) .

3.2 Σύγκριση με άλλα συστήματα

3.2.1 Preference SQL



Το σύστημα preference SQL [KK02] έχει ως στόχο την ενσωμάτωση των προτιμήσεων στα σχεσιακά ερωτήματα, με τρόπο παρόμοιο με αυτόν του PrefDB. Χρησιμοποιεί τους τελεστές "PREFERRING","BUT ONLY" και "CASCADE" για να επεκτείνει το σχεσιακό μοντέλο, και να δηλώσει ερωτήματα με προτιμήσεις και περιορισμούς. Οι πρώτες μπορούν να ταξινομηθούν με σειρά βαρύτητας. Από τα αποτελέσματα που προκύπτουν από τις υποχρεωτικές επιλογές στο ερώτημα, το σύστημα εξάγει τα κατά Pareto καλύτερα, δηλαδή την κορυφογραμμή με βάση τις προτιμήσεις (PREFERRING), χωρίς να παραβιάζονται όμως οι περιορισμοί (BUT ONLY). Στη συνέχεια το σύστημα παρουσιάζει στο χρήστη τα επικρατέστερα αυτά αποτελέσματα. Χαρακτηριστικό παράδειγμα χρήσης του Preference SQL είναι το COSIMA, ένα online σύστημα αγορών, στο οποίο η εικονική πωλήτρια

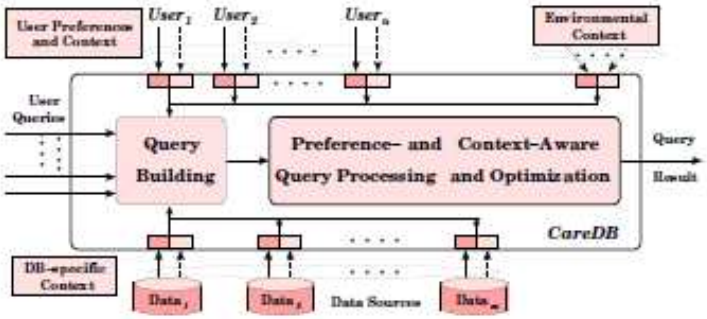


χρησιμοποιεί τεχνικές ψυχολογίας για να πείσει τον πελάτη πως τα αποτελέσματα που δίνονται από το Preference SQL ανταποκρίνονται καλύτερα στις ανάγκες του. Ως επιχειρήματα, χρησιμοποιεί τα χαρακτηριστικά του προϊόντος τα οποία το κάνουν να υπερέχει από τα υπόλοιπα εκτός κορυφογραμμής, δίνοντας την εντύπωση ενός έξυπνου πωλητή.

Επιλέγοντας έναν συγκεκριμένο αλγόριθμο για την εξαγωγή των βέλτιστων αποτελεσμάτων , το Preference SQL επιτρέπει την υλοποίηση τέτοιων έξυπνων συστημάτων. Σε κάθε περίπτωση, το σύστημα γνωρίζει ποιο ήταν το χαρακτηριστικό μιας πλειάδας που την έκανε να επικρατήσει απέναντι σε κάποια άλλη και να μείνει στην κορυφογραμμή. Το PrefDB , αντίθετα, προσφέρει μεγαλύτερη ευελιξία με τη δυνατότητα επιλογής αλγορίθμων, όμως αυτό δε διευκολύνει την αξιοποίηση των στοιχείων στα οποία υπερτερεί ένα αποτέλεσμα απέναντι στα άλλα κατά την παρουσίασή τους, καθώς η πληροφορία αυτή εξαρτάται από τον επιλεγμένο αλγόριθμο. Τα δύο συστήματα επιλέγουν επίσης διαφορετικό τρόπο για να περιορίσουν τη χαλαρότητα του ερωτήματος ως προς τις προτιμήσεις του χρήστη. Το preference SQL χρησιμοποιεί τον τελεστή BUT ONLY, ο οποίος θέτει αυστηρούς περιορισμούς για το εύρος τιμών γύρω από την επιθυμητή, ενώ το PrefDB υλοποιεί την εμπιστοσύνη για το αποτέλεσμα, ένα μέτρο για το πόσο σίγουρο μπορεί να είναι το σύστημα ότι το αποτέλεσμα ανταποκρίνεται όντως στις προτιμήσεις του χρήστη. Το PrefDB αντιλαμβάνεται πόσο ασφαλείς είναι οι επαγωγές που κάνει στο σύνολό τους, οπότε μπορεί να περιορίσει την απόκλιση των αποτελεσμάτων από το επιθυμητό, αλλά όχι να το κάνει σε συγκεκριμένα χαρακτηριστικά όπως το Preference SQL, ενώ λοιπόν το PrefDB υπολογίζει με αμιγώς ποσοτική προσέγγιση τα καλύτερα αποτελέσματα, το PreferenceSQL χρησιμοποιεί

και κάποια κριτήρια που παραπέμπουν σε ποιοτική προσέγγιση. Τέλος , το PreferenceSQL τρέχει μετατρέποντας τα ερωτήματα με προτιμήσεις σε απλά σχεσιακά, ενώ το PrefDB αλλάζει τον τρόπο με τον οποίο γίνεται η εκτέλεση του ερωτήματος στη βάση.

3.2.2 Care DB



Το σύστημα CareDB [LKM10] έχει αρκετές ομοιότητες με το PrefDB, αλλά και αρκετές διαφορές, στον τρόπο με τον οποίο διαχερίζεται τις πληροφορίες. Η κεντρική ιδέα είναι πως η κάθε προτίμηση δίνει βαθμολογία στις πλειάδες που έχουν κάποια συγκεκριμένα χαρακτηριστικά με βάση μια συνάρτηση βαθμολόγησης, στη συνέχεια γίνεται συνάθροιση και επιστρέφονται τα καλύτερα αποτελέσματα για κάθε ερώτημα. Η σημαντική διαφορά του CareDB είναι η δυνατότητα να περαστεί ως είσοδος σε αυτό ένα περιβάλλον εκτέλεσης. Το περιβάλλον αυτό χωρίζει τις παραμέτρους του σε τρεις κατηγορίες, καθεμία από τις οποίες μπορεί να έχει στατικά στοιχεία που μεταβάλλονται σπάνια και δυναμικά, που μεταβάλλονται συχνά. Οι κατηγορίες αυτές είναι: πληροφορίες για το χρήστη, πληροφορίες για τη βάση και πληροφορίες για την τοποθεσία. Για το χρήστη μπορεί να υπάρχουν στατικές πληροφορίες όπως ηλικία, επάγγελμα, εισόδημα και δυναμικές όπως ακριβής τοποθεσία, διάθεση. Για μια βάση με ταινίες μπορεί να υπάρχουν στατικές πληροφορίες όπως τίτλοι, χρονολογίες, ή δυναμικές όπως διαθεσιμότητα σε video club. Για το περιβάλλον μπορεί να υπάρχουν στατικές πληροφορίες όπως οι συγκοινωνίες της περιοχής, ή δυναμικές όπως η κίνηση τη συγκεκριμένη ώρα. Η εμπιστοσύνη δεν υλοποιείται και σε αυτό το σύστημα, αλλά ενσωματώνεται ένα διαφορετικό υποσύστημα, το οποίο διαχειρίζεται αβέβαια δεδομένα και κατ'επέκταση μπορεί να θεωρήσει την εμπιστοσύνη ως βαθμό αβεβαιότητας.

Η υλοποίηση των διαφορετικών αλγορίθμων εκτέλεσης γίνεται με το σύστημα FlexPref, το οποίο παίρνει ως plugins τους διαφορετικούς αλγόριθμους και επιτρέπει να χρησιμοποιηθεί ο επιλεγμένος ανεξάρτητα από τα δεδομένα και τις προτιμήσεις, με τρόπο παρόμοιο με αυτό που γίνεται και στο PrefDB.

4 Ανάλυση Απαιτήσεων Συστήματος

Εδώ αρχικά παρουσιάζονται οι απαιτήσεις του συστήματος, λειτουργικές και μη, ενώ στη συνέχεια γίνεται ανάλυση των προδιαγραφών του συστήματος, με τον πλήρη καθορισμό των περιπτώσεων χρήσης. Έπειτα παρατίθεται το μοντέλο οντοτήτων συσχετίσεων που σχεδιάστηκε για την ικανοποίηση αυτών των απαιτήσεων.

4.1 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις του συστήματος ήταν οι πρώτες που καθορίστηκαν επακριβώς και πραγματεύονται τις επιλογές που έχουν να κάνουν με την παρουσίαση και χρηστικότητα του συστήματος. Σημαντικότερη ίσως από αυτές τις επιλογές ήταν η ανάπτυξη του συστήματος ως διαδικτυακή εφαρμογή, με δομή server-client. Αυτό δίνει ένα εξαιρετικά σημαντικό πλεονέκτημα στο σύστημα, πως ο χρήστης δε χρειάζεται να το εγκαταστήσει σε κάποια συσκευή, αλλά τρέχει αυτόνομα πάνω σε όλους τους διαδεδομένους browsers. Το κόστος για αυτή την επιλογή είναι πως η εφαρμογή απαιτεί σύνδεση στο διαδίκτυο και επιπλέον τρέχει με ταχύτητα ελαφρά χαμηλότερη, αν και εφάμιλλη αντίστοιχων desktop εφαρμογών. Σημαντική παράμετρος για να επιτευχθεί αυτό ήταν το ότι η μεταφορά των δεδομένων από τον server γίνεται εξ'ολοκλήρου on demand, όταν είναι απαραίτητο να γίνει κλήση στον server για να ολοκληρωθεί κάποια διαδικασία. Μετά την πρώτη φορά που έρχονται τα δεδομένα στον client, αποθηκεύονται στο δικό του data model και ανανεώνονται με εσωτερικούς μηχανισμούς. Η όψη και η αίσθηση της εφαρμογής έπρεπε να είναι κατά το δυνατόν πιο κοντά στα ήδη υπάρχοντα συστήματα, όπως το pgAdmin[7], έτσι ώστε οι χρήστες να έχουν άμεση εξοικείωση, χωρίς να χρειάζεται να περάσουν κάποια καμπύλη εκμάθησης

[7] http://www.pgadmin.org/

ειδικά για τη συγκεκριμένη εφαρμογή. Για να επιτευχθούν αυτοί οι στόχοι, χρησιμοποιήθηκαν διάφορα προγραμματιστικά εργαλεία και πλατφόρμες. Η ανάπτυξη του συστήματος έγινε στο Eclipse[8] ώστε να είναι εύκολη η εποπτεία σε server και client, καθώς και οι δοκιμές του συστήματος. Η εφαρμογή γράφτηκε σε HTML/CSS/Javascript, χρησιμοποιώντας jQuery[9], Direct Web Remoting (DWR)[10], knockout.js[11], jQuery Datatables[12] και άλλα εργαλεία .

4.2 Λειτουργικές Απαιτήσεις

Οι διαχειριστές του συστήματος έπρεπε μέσω της εφαρμογής να έχουν τη δυνατότητα να εκτελούν τις εξής λειτουργίες :

- Σύνδεση στο σύστημα
- Πλοήγηση στο σχήμα του κάθε διαθέσιμου server, με όλες τις βάσεις, τα προφίλ, τους πίνακες και τις προτιμήσεις του.
- Επιλογή Βάσης Δεδομένων στην οποία θα εργαστεί ο χρήστης
- Δημιουργία προτιμήσεων
- Τροποποίηση/Διαγραφή προτιμήσεων
- Δημιουργία προφίλ
- Μετονομασία προφίλ
- Αποθήκευση ερωτημάτων, με τις παραμέτρους και τις επιλεγμένες προτιμήσεις τους.
- Φόρτωση ερωτημάτων που έχουν αποθηκευτεί.
- Εκτέλεση ερωτημάτων
- Πλοήγηση στα αποτελέσματα
- Εκτέλεση εντολών σε κονσόλα

[8] http://www.eclipse.org/

[9] www.jQuery.com

[10] http://directwebremoting.org/dwr/index.html

[11] http://knockoutjs.com/

[12] http://datatables.net/

Με βάση αυτές τις προδιαγραφές σχεδιάστηκαν τα σενάρια χρήσης που ακολουθούν, τα οποία καλύπτουν επακριβώς τις λειτουργίες που πρέπει να εκτελεί το σύστημα , καθώς και όλα τα διαφορετικά σενάρια εκτέλεσης.

4.3 Περιγραφή Λειτουργιών

Στη συνέχεια θα παρουσιαστούν σενάρια χρήσης του συστήματος που περιλαμβάνουν τις πιο σημαντικές λειτουργίες του συστήματος, καθώς και τους δράστες που συμμετέχουν σε αυτές. Κάθε τέτοιο σενάριο θα συνοδεύεται από το αντίστοιχο UML διάγραμμα, ώστε να είναι ξεκάθαρος ο ρόλος των υποσυστημάτων και η αλληλεπίδραση μεταξύ τους. Στα σενάρια αυτά θα θεωρηθεί δεδομένη η συνεχής και σωστή λειτουργία του server στον οποίο τρέχει η εφαρμογή.

Οι δράστες του συστήματος εν συντομία είναι οι εξής:

Χρήστης – Ενεργό εξωτερικό σύστημα, συνδέεται με το σύστημα και δίνει την εντολή για την εκτέλεση των περισσότερων λειτουργιών.

Κεντρικός Διακομιστής – Παθητικό εξωτερικό σύστημα, χρησιμοποιείται μόνο κατά τη σύνδεση του χρήστη στο σύστημα, για την επιβεβαίωση των στοιχείων του.

Διακομιστής – Παθητικό εξωτερικό σύστημα, γίνεται σύνδεση με αυτόν για να τραβήξει το σύστημα τις διαθέσιμες βάσεις δεδομένων.

Βάση Δεδομένων – Παθητικό εξωτερικό σύστημα, το σύστημα παίρνει δεδομένα από τη βάση και γράφει πίσω άλλα δεδομένα.

.

4.3.1 PrefDB01 – Σύνδεση του χρήστη στο σύστημα

Περιγραφή: Κατά την είσοδό του στη σελίδα, ο χρήστης θα πρέπει να δίνει τα στοιχεία του, όνομα χρήστη και κωδικό, ώστε να συνδεθεί με το σύστημα. Σε περίπτωση εισαγωγής λάθος στοιχείων, θα ενημερώνεται με κατάλληλο διαγνωστικό μήνυμα.

Δράστες: Χρήστης, Κεντρικός Διακομιστής

Προϋποθέσεις: -

Συνθήκη Εκκίνησης: Κάποιος χρήστης επισκέπτεται την αρχική σελίδα.

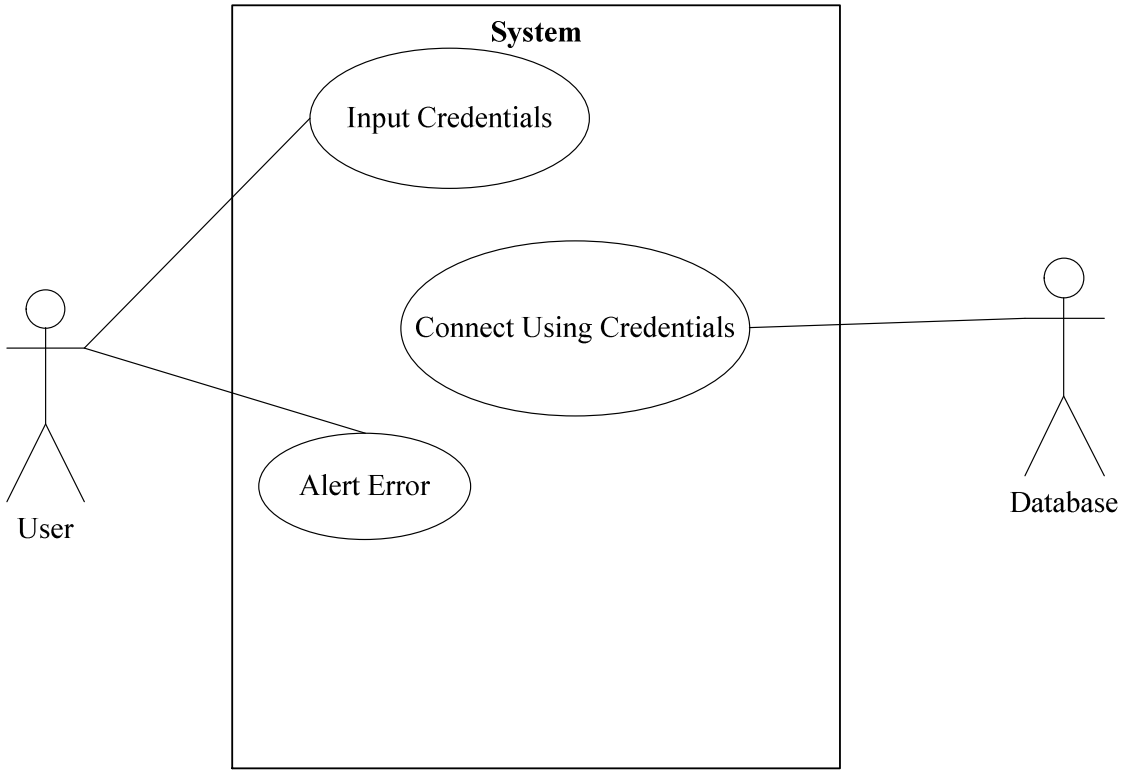
Κείμενο Σεναρίου:

1. Εμφανίζεται μια φόρμα για την εισαγωγή ονόματος χρήστη και κωδικού.
2. Ο χρήστης εισάγει τα στοιχεία του και επιλέγει «Login»
3. Το παράθυρο κλείνει και ο χρήστης βρίσκεται στην κεντρική οθόνη.

Εναλλακτικές Πορείες:

3a . Ο χρήστης δίνει λάθος στοιχεία στο βήμα 2.

1. Ο χρήστης ενημερώνεται για τον τύπο λάθους.
2. Εμφανίζεται με popup μήνυμα λάθους και ο χρήστης επαναλαμβάνει το βήμα 2.



4.3.2 PrefDB02 – Πλοήγηση στις βάσεις δεδομένων κατά προφίλ

Περιγραφή: Ο χρήστης πρέπει να έχει τη δυνατότητα να βλέπει για τους διαθέσιμους διακομιστές τις βάσεις δεδομένων, τα προφίλ και τις προτιμήσεις που υπάρχουν στον καθένα.

Δράστες: Χρήστης, Διακομιστές, Βάσεις δεδομένων

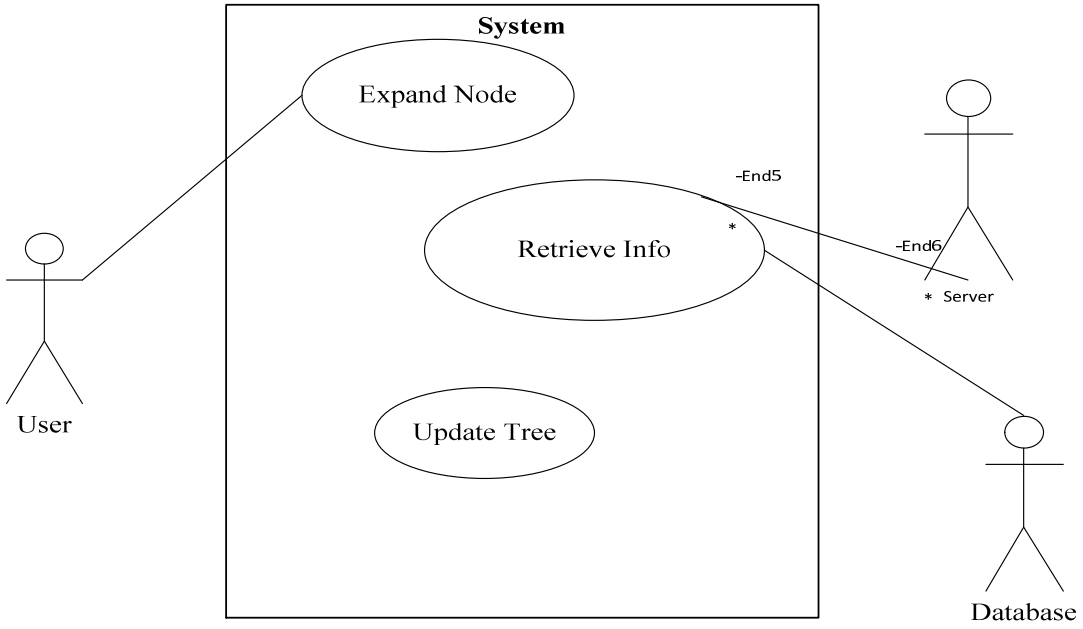
Προϋποθέσεις: Χρήστης συνδεδεμένος.

Συνθήκη Εκκίνησης: -

Κείμενο Σεναρίου:

1. Στο παράθυρο πλοήγησης, εμφανίζονται οι διακομιστές ως ρίζες δέντρου
2. Ο χρήστης έχει τη δυνατότητα να επεκτείνει τους κόμβους των διακομιστών, με τις διαθέσιμες βάσεις δεδομένων να μπαίνουν στο επόμενο επίπεδο.
3. Η κάθε βάση δεδομένων επεκτείνεται όταν την επιλέγει ο χρήστης, τα προφίλ που ανήκουν σε αυτή εμφανίζονται στο επόμενο επίπεδο.
4. Το κάθε προφίλ επεκτείνεται όταν το επιλέγει ο χρήστης, οι προτιμήσεις που ανήκουν σε αυτό εμφανίζονται στο επόμενο επίπεδο.

Εναλλακτικές Πορείες:



4.3.3 PrefDB03 – Πλοήγηση στις βάσεις δεδομένων κατά πίνακα

Περιγραφή: Ο χρήστης πρέπει να έχει τη δυνατότητα να βλέπει για τους διαθέσιμους διακομιστές τις βάσεις δεδομένων, τους πίνακες, τα χαρακτηριστικα, τα προφίλ και τις προτιμήσεις που υπάρχουν στον καθένα.

Δράστες: Χρήστης, Διακομιστές, Βάσεις δεδομένων

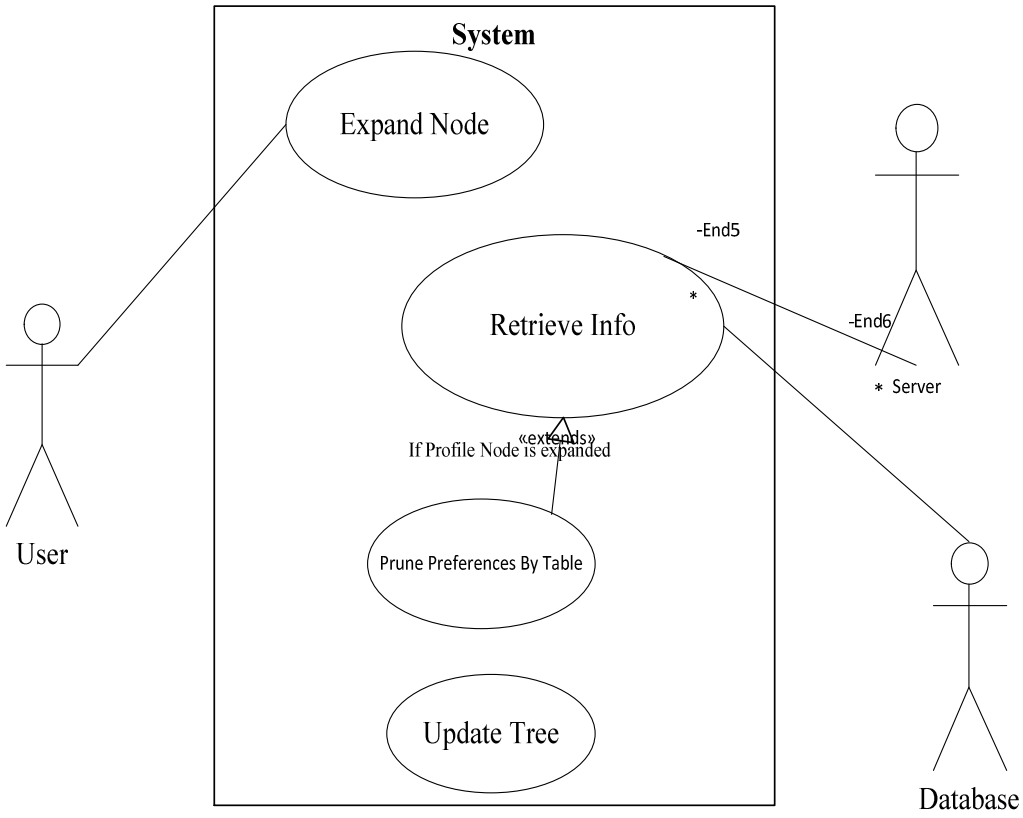
Προϋποθέσεις: Χρήστης συνδεδεμένος

Συνθήκη εκκίνησης: -

Κείμενο Σεναρίου:

1. Στο παράθυρο πλοήγησης, εμφανίζονται οι διακομιστές ως ρίζες δέντρου
2. Ο χρήστης έχει τη δυνατότητα να επεκτείνει τους κόμβους των διακομιστών, με τις διαθέσιμες βάσεις δεδομένων να μπαίνουν στο επόμενο επίπεδο.
3. Η κάθε βάση δεδομένων επεκτείνεται όταν την επιλέγει ο χρήστης, οι πίνακες που ανήκουν σε αυτή εμφανίζονται στο επόμενο επίπεδο.
4. Ο κάθε πίνακας επεκτείνεται όταν τον επιλέγει ο χρήστης, τα χαρακτηριστικά του και τα προφίλ μπαίνουν στο επόμενο επίπεδο.
5. Το κάθε προφίλ επεκτείνεται όταν το επιλέγει ο χρήστης, οι προτιμήσεις που ανήκουν σε αυτό και αναφέρονται στον πίνακα του οποίου είναι κόμβοι παιδιά εμφανίζονται στο επόμενο επίπεδο.

Εναλλακτικές Πορείες:



4.3.4 PrefDB04-Επιλογή διακομιστή και βάσης.

Περιγραφή: Ο χρήστης πρέπει να επιλέξει το διακομιστή και τη βάση στην οποία θέλει να εισάγει προτιμήσεις, προφίλ και να εκτελεί ερωτήματα.

Δράστες: Χρήστης, Διακομιστής

Προϋποθέσεις: Χρήστης συνδεδεμένος

Συνθήκη Εκκίνησης: Ο χρήστης επιλέγει "Select Database".

Κείμενο Σεναρίου:

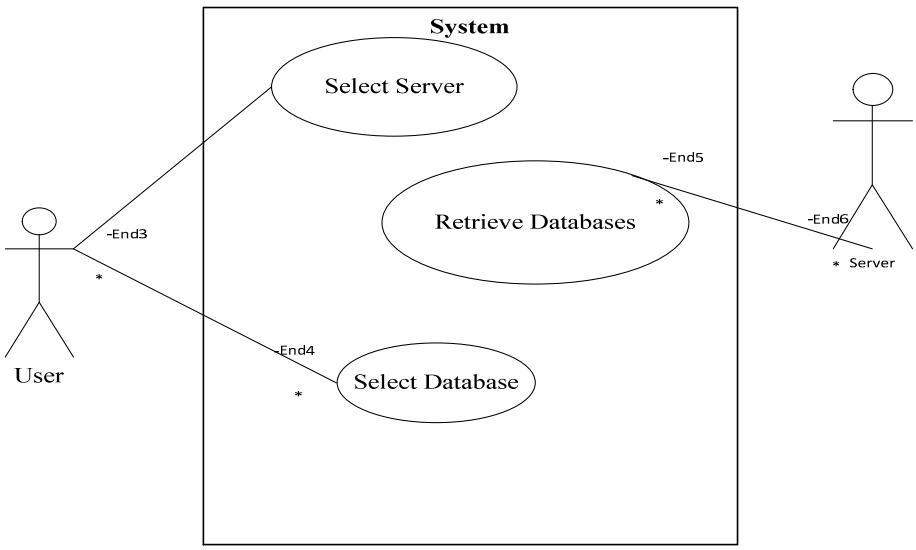
1. Εμφανίζεται το παράθυρο επιλογής.
2. Ο χρήστης επιλέγει διακομιστή και στη συνέχεια βάση.
3. Ο χρήστης επιλέγει "Select"

4. Το παράθυρο κλείνει και ανανεώνεται η επιλεγμένη βάση.

Εναλλακτικές Πορείες:

2a. Ο χρήστης επιλέγει ακύρωση.

1. Το παράθυρο επιλογής κλείνει και ο χρήστης επιστρέφει στην αρχική σελίδα.



4.3.5 PrefDB05-Εισαγωγή νέας προτίμησης/συνάρτησης

Περιγραφή: Ο χρήστης μπορεί να εισάγει νέες προτιμήσεις στο σύστημα. Η εισαγωγή προτίμησης γίνεται με την επιλογή μιας σειράς παραμέτρων. Ανάμεσα σε αυτές είναι και η συνάρτηση βαθμολόγησης. Αν ο χρήστης θέλει να ορίσει μια νέα συνάρτηση βαθμολόγησης, το κάνει παράλληλα με τον ορισμό μιας προτίμησης που θα τη χρησιμοποιεί.

Δράστες: Χρήστης, Διακομιστής, Βάση Δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος, βάση επιλεγμένη.

Συνθήκη Εκκίνησης: Ο χρήστης επιλέγει “Create Preference”.

Κείμενο Σεναρίου:

1. Αρχικοποιείται η φόρμα εισαγωγής προτίμησης.
2. Ο χρήστης εισάγει τις απαραίτητες παραμέτρους στη φόρμα εισαγωγής προτίμησης και επιλέγει submit.

3. Γίνεται έλεγχος των στοιχείων που εισάγει ο χρήστης.
4. Ο διακομιστής κάνει την εισαγωγή της νέας προτίμησης.
5. Το παράθυρο εισαγωγής προτίμησης κλείνει.

Εναλλακτικές Πορείες:

2a. Ο χρήστης επιλέγει ακύρωση.

1. Η φόρμα εξαφανίζεται και ο χρήστης επιστρέφει στην αρχική σελίδα.

3a. Ο χρήστης κάνει λάθος σε κάποιο πεδίο.

1. Εμφανίζεται διαγνωστικό μήνυμα με popup.
2. Ο χρήστης επιστρέφει στο βήμα 2.

3b. Ο χρήστης έχει επιλέξει νέα συνάρτηση

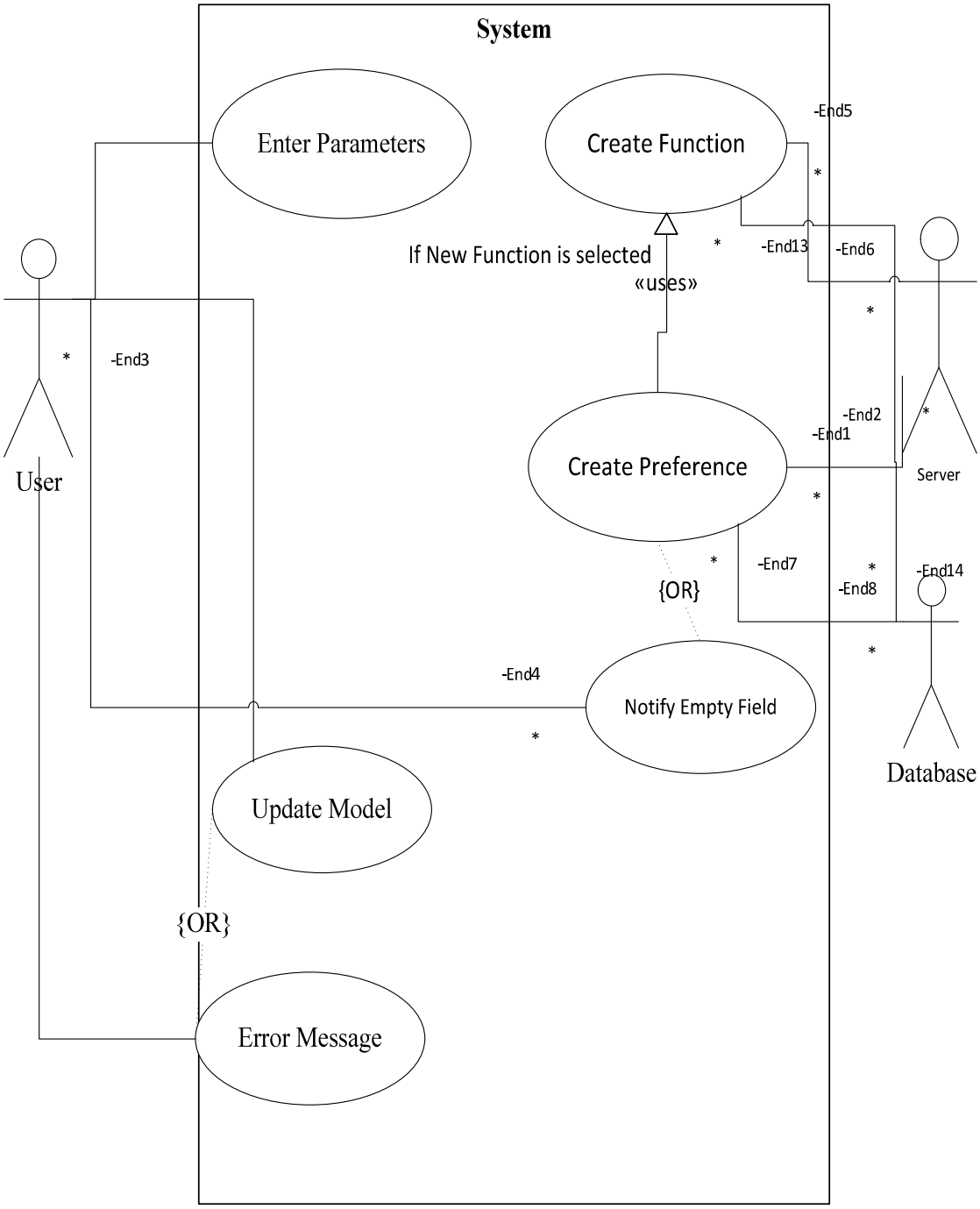
1. Γίνεται εισαγωγή της νέας συνάρτησης.

    1. Αν το όνομα που έδωσε ο χρήστης υπάρχει ήδη ,η εισαγωγή αποτυγχάνει, εμφανίζεται διαγνωστικό μήνυμα με popup και ο χρήστης επιστρέφει στο βήμα 2.

2. Ο χρήστης επιστρέφει στο βήμα 2.

4a. Ο χρήστης δίνει συνδυασμό προτίμησης-προφίλ που υπάρχει ήδη.

1. Εμφανίζεται διαγνωστικό μήνυμα με popup .
2. Ο χρήστης επιστρέφει στο βήμα 2.


System
Enter Parameters
Create Function
-End5
*
If New Function is selected
«uses»
*
-End13
-End6
*
-End3
*
-End2
-End1
Create Preference
*
Server
User
*
-End7
*
-End14
{OR}
-End8
*
-End4
Notify Empty Field
*
Database
Update Model
{OR}
Error Message

4.3.6 PrefDB06 – Τροποποίηση/Διαγραφή υπάρχουσας προτίμησης

Περιγραφή: Ο χρήστης μπορεί να τροποποιεί τις προτιμήσεις που υπάρχουν ήδη στο σύστημα.

Δράστες: Χρήστης, Διακομιστής, Βάση Δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος

Συνθήκη Εκκίνησης: Ο χρήστης κάνει δεξί κλικ σε μια προτίμηση στο παράθυρο πλοήγησης.

Κείμενο Σεναρίου:

1. Η φόρμα εισαγωγής προτίμησης παίρνει τις τιμές που αντιστοιχούν στην επιλεγμένη προτίμηση.
2. Ο χρήστης αλλάζει τις επιθυμητές παραμέτρους στη φόρμα εισαγωγής προτίμησης και επιλέγει submit.
3. Γίνεται έλεγχος των στοιχείων που εισάγει ο χρήστης.
4. Ο διακομιστής κάνει την τροποποίηση της προτίμησης.
5. Το παράθυρο εισαγωγής προτίμησης κλείνει.

Εναλλακτικές Πορείες:

2a. Ο χρήστης επιλέγει ακύρωση.

1. Η φόρμα εξαφανίζεται και ο χρήστης επιστρέφει στην αρχική σελίδα. Η προτίμηση δεν τροποποιείται.

2b. Ο χρήστης επιλέγει διαγραφή.

1. Εμφανίζεται popup για επιβεβαίωση στο χρήστη.

1i. Ο χρήστης επιλέγει ναι.

1ii. Η φόρμα εξαφανίζεται, ο χρήστης επιστρέφει στην αρχική σελίδα. Η προτίμηση διαγράφεται.

2i. Ο χρήστης επιλέγει όχι

2ii. Το popup επιβεβαίωσης κλείνει και ο χρήστης επιστρέφει στο βήμα 2.

3a. Ο χρήστης κάνει λάθος σε κάποιο πεδίο.

1. Εμφανίζεται διαγνωστικό μήνυμα με popup.
2. Ο χρήστης επιστρέφει στο βήμα 2.

3b. Ο χρήστης έχει επιλέξει νέα συνάρτηση

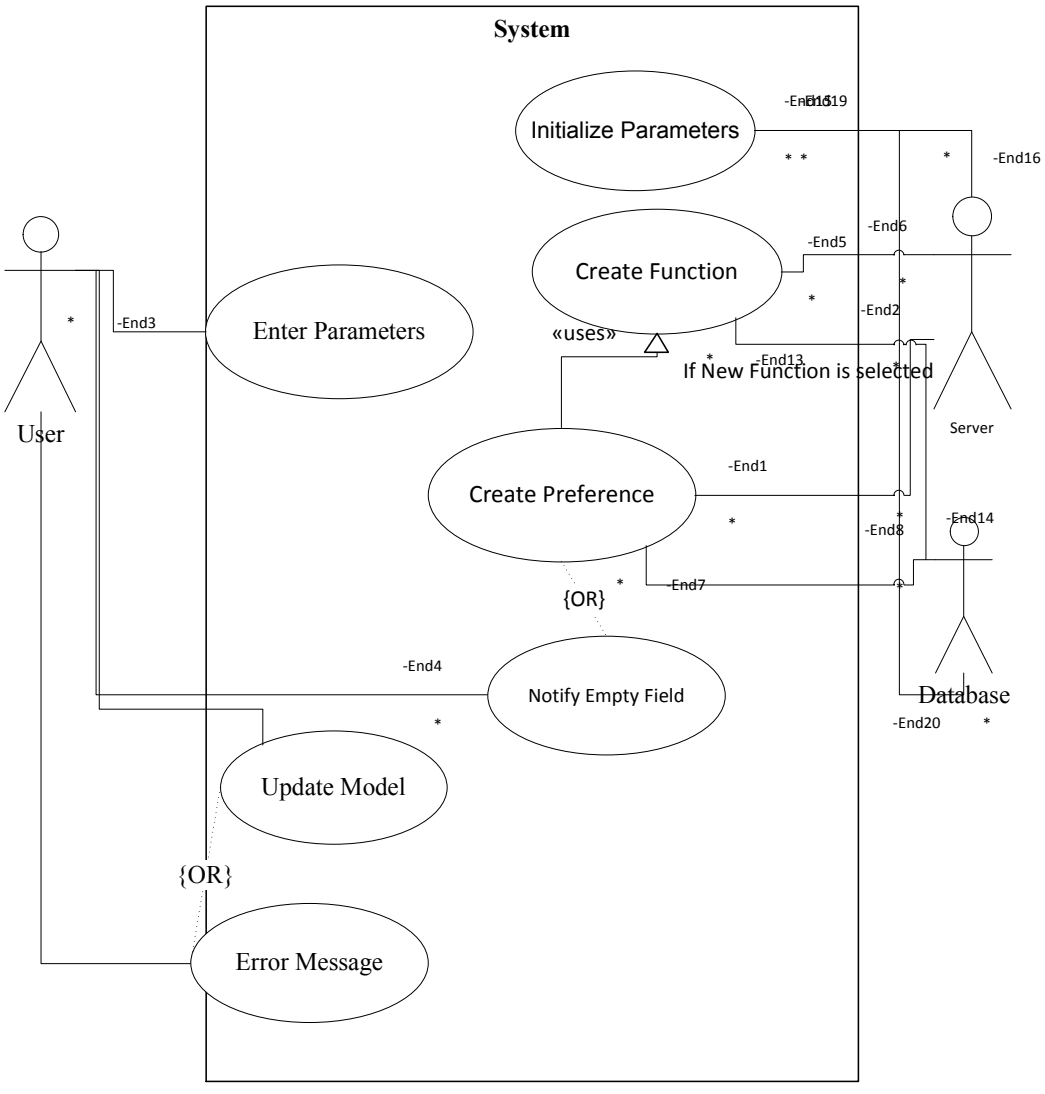
1. Γίνεται εισαγωγή της νέας συνάρτησης.

1. Αν το όνομα που έδωσε ο χρήστης υπάρχει ήδη ,η εισαγωγή αποτυγχάνει, εμφανίζεται διαγνωστικό μήνυμα με popup και ο χρήστης επιστρέφει στο βήμα 2.

2. Ο χρήστης επιστρέφει στο βήμα 2.

4a. Ο χρήστης δίνει συνδυασμό προτίμησης-προφίλ που υπάρχει ήδη.

1. Εμφανίζεται διαγνωστικό μήνυμα με popup .
2. Ο χρήστης επιστρέφει στο βήμα 2.



4.3.7 PrefDB07 – Δημιουργία προφίλ

Περιγραφή: Ο χρήστης έχει τη δυνατότητα να δημιουργεί προφίλ, δηλαδή ομάδες προτιμήσεων.

Δράστες: Χρήστης, Διακομιστής, Βάση δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος, βάση επιλεγμένη.

Συνθήκη Εκκίνησης: Ο χρήστης επιλέγει "Create Profile".

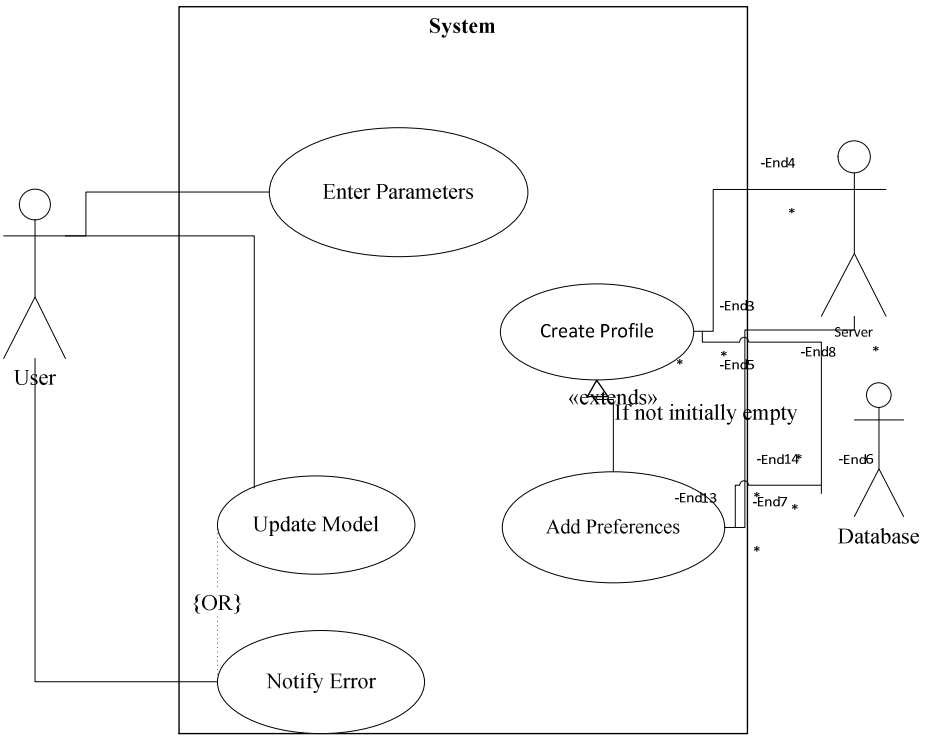
Κείμενο Σεναρίου:

1. Ο χρήστης εισάγει τις απαραίτητες παραμέτρους στη φόρμα δημιουργίας προφίλ.
2. Ο διακομιστής κάνει την εισαγωγή του νέου προφίλ.
3. Το παράθυρο εισαγωγής προφίλ κλείνει.

Εναλλακτικές Πορείες:

3a. Ο χρήστης δίνει συνδυασμό βάσης-ονόματος προφίλ που υπάρχει ήδη.

1. Εμφανίζεται διαγνωστικό μήνυμα πάνω στη φόρμα δημιουργίας προφίλ.
2. Ο χρήστης επιστρέφει στο βήμα 1.



4.3.8 PrefDB08 – Τροποποίηση προφίλ

Περιγραφή: Ο χρήστης έχει τη δυνατότητα να αλλάζει όνομα στα ήδη υπάρχοντα προφίλ.

Δράστες: Χρήστης, Διακομιστής, Βάση δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος

Συνθήκη Εκκίνησης: Ο χρήστης κάνει δεξί κλικ σε ένα προφίλ στο παράθυρο πλοήγησης.

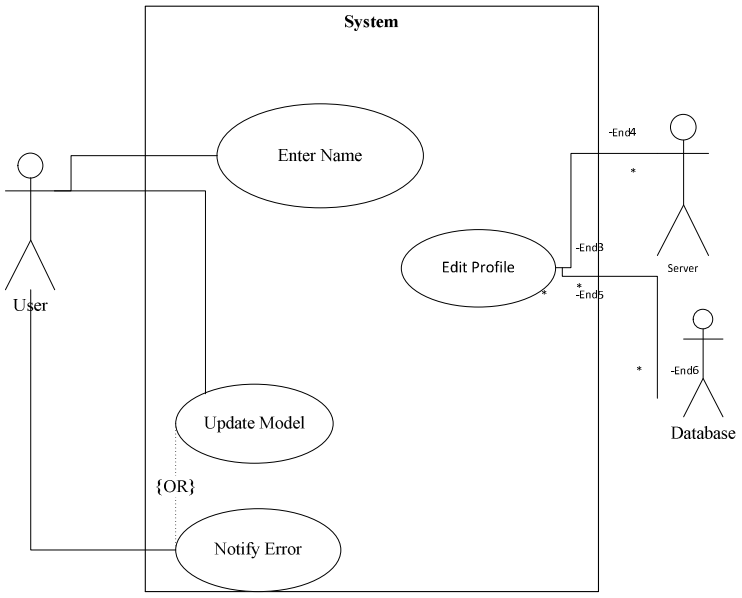
Κείμενο Σεναρίου:

1. Εμφανίζεται το παράθυρο τροποποίησης προφίλ
2. Ο χρήστης εισάγει το νέο όνομα για το προφίλ.
3. Ο χρήστης επιλέγει υποβολή.
4. Ο διακομιστής κάνει την τροποποίηση του προφίλ.
5. Το παράθυρο τροποποίησης προτίμησης κλείνει.

Εναλλακτικές Πορείες:

3a. Ο χρήστης δίνει συνδυασμό βάσης-ονόματος προφίλ που υπάρχει ήδη.

1. Εμφανίζεται διαγνωστικό μήνυμα πάνω στη φόρμα δημιουργίας προφίλ.
2. Ο χρήστης επιστρέφει στο βήμα 1.



4.3.9 PrefDB09 - Αποθήκευση παραμέτρων ερωτήματος

Περιγραφή: Ο χρήστης έχει τη δυνατότητα να αποθηκεύει στη βάση ερωτήματα, με το κείμενό τους, τις παραμέτρους εκτέλεσης, καθώς και τα προφίλ και τις προτιμήσεις που χρησιμοποιούν.

Δράστες: Χρήστης, Διακομιστής, Βάση δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος, Σύνδεση με βάση

Συνθήκη Εκκίνησης: Ο χρήστης επιλέγει "Save Query Configuration".

Κείμενο Σεναρίου:

1. Εμφανίζεται το παράθυρο αποθήκευσης παραμέτρων ερωτήματος.
2. Ο χρήστης εισάγει το όνομα με το οποίο επιθυμεί να αποθηκευθεί το ερώτημα.
3. Ο χρήστης επιλέγει "Submit".
4. Το ερώτημα αποθηκεύεται στη βάση.

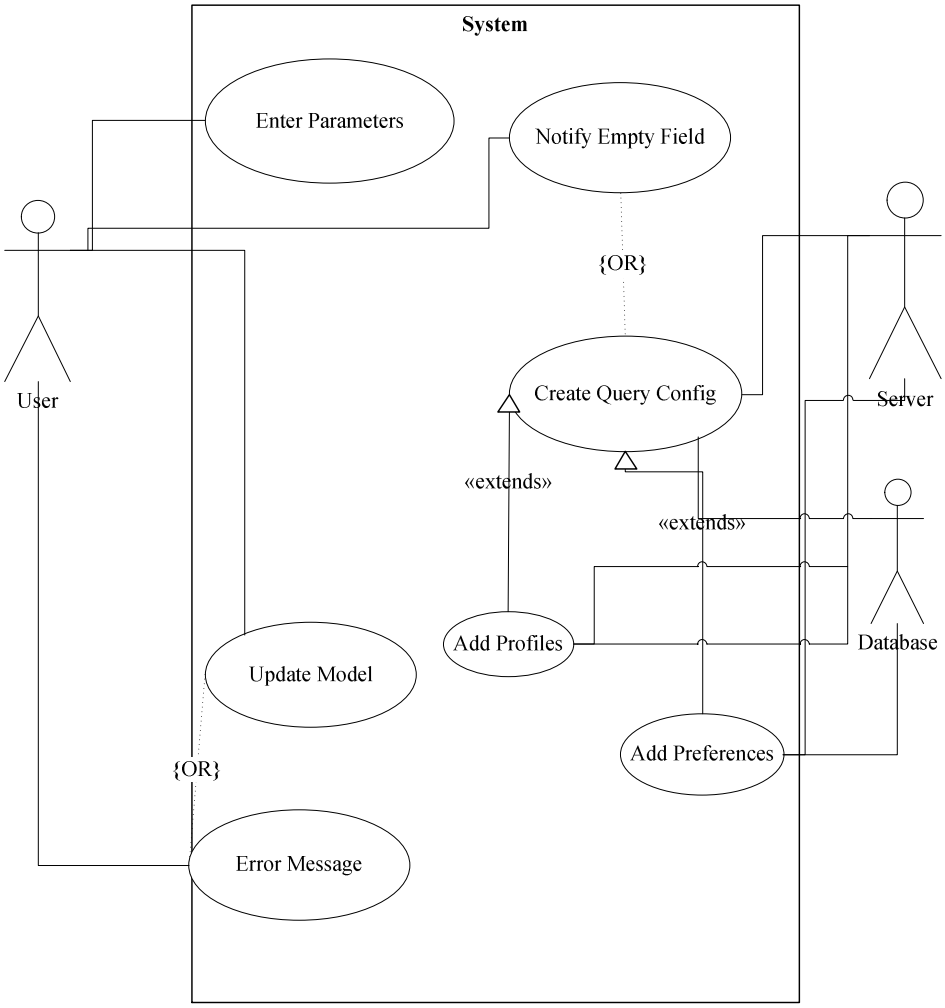
Εναλλακτικές Πορείες:

2a. Ο χρήστης επιλέγει ακύρωση.

1. Η φόρμα εξαφανίζεται και ο χρήστης επιστρέφει στην αρχική σελίδα. Το ερώτημα δεν αποθηκεύεται.

3a. Ο χρήστης κάνει λάθος σε κάποιο πεδίο.

1. Εμφανίζεται διαγνωστικό μήνυμα με popup.
2. Ο χρήστης επιστρέφει στο βήμα 2.



4.3.10 PrefDB10 – Φόρτωση παραμέτρων ερωτήματος

Περιγραφή: Ο χρήστης έχει τη δυνατότητα να ανακτά τα ερωτήματα που έχει αποθηκεύσει στη βάση.

Δράστες: Χρήστης, Διακομιστής, Βάση δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος, Σύνδεση με βάση

Συνθήκη Εκκίνησης: Ο χρήστης επιλέγει “Load Query Configuration”.

Κείμενο Σεναρίου:

1. Εμφανίζεται το παράθυρο φόρτωσης παραμέτρων ερωτήματος.

2. Ο χρήστης επιλέγει το όνομα του συνόλου παραμέτρων που θέλει να φορτώσει.
3. Ο χρήστης επιλέγει “Submit”.
4. Γίνεται φόρτωση των παραμέτρων, καθώς και όλων των προτιμήσεων και προφίλ που χρησιμοποιούνται.
5. Ο χρήστης επιστρέφει στην αρχική σελίδα όπου απεικονίζονται οι νέες παράμετροι.

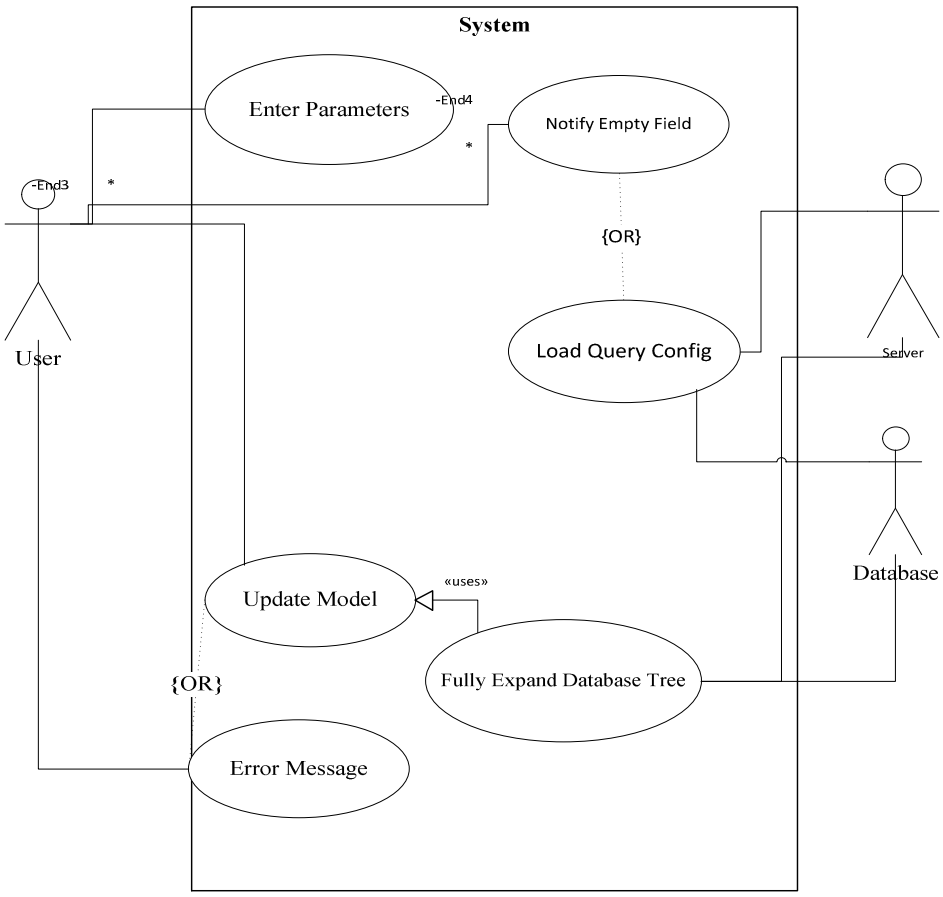
Εναλλακτικές Πορείες:

2a. Ο χρήστης επιλέγει ακύρωση.

1. Η φόρμα εξαφανίζεται και ο χρήστης επιστρέφει στην αρχική σελίδα. Δε γίνεται κάποια αλλαγή.

3a. Ο χρήστης κάνει λάθος σε κάποιο πεδίο.

1. Εμφανίζεται διαγνωστικό μήνυμα με popup.
2. Ο χρήστης επιστρέφει στο βήμα 2.



4.3.11 PrefDB11 – Εκτέλεση ερωτήματος

Περιγραφή: Ο χρήστης έχει τη δυνατότητα να εκτελεί ερωτήματα με προτιμήσεις στη βάση δεδομένων.

Δράστες: Χρήστης, Διακομιστής, Βάση δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος, Σύνδεση με βάση

Συνθήκη Εκκίνησης: -

Κείμενο Σεναρίου:

1. Στην αρχική σελίδα ο χρήστης εισάγει τις παραμέτρους για το ερώτημα και το query του σε SQL.
2. Ο χρήστης με πλοήγηση στο διακομιστή επιλέγει τα προφίλ και τις προτιμήσεις που θέλει να χρησιμοποιηθούν στο ερώτημα.
3. Ο χρήστης επιλέγει "Execute".
4. Εμφανίζεται η τελική μορφή του ερωτήματος στο χρήστη.
5. Ο διακομιστής εκτελεί το ερώτημα.
6. Εμφανίζονται τα αποτελέσματα.

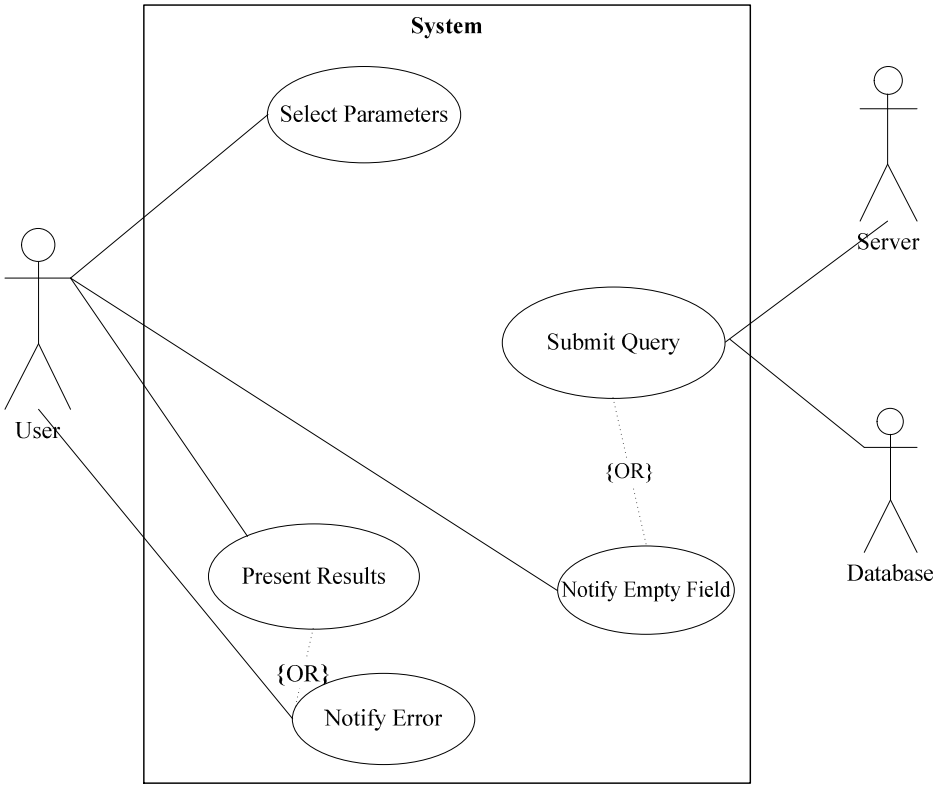
Εναλλακτικές Πορείες:

2a. Ο χρήστης εκτελεί ξανά ένα ερώτημα με διαφορετικές παραμέτρους.

1. Ο χρήστης ενημερώνεται για το αν θα πρέπει το ερώτημα να εκτελεστεί από την αρχή ή απλά να ανανεωθούν τα αποτελέσματα.
2. Ο χρήστης επιστρέφει στο βήμα 2.

3a. Ο χρήστης κάνει λάθος σε κάποιο πεδίο.

1. Εμφανίζεται διαγνωστικό μήνυμα με popup.
2. Ο χρήστης επιστρέφει στο βήμα 2.



4.3.12 PrefDB12 – Πλοήγηση στα αποτελέσματα

Περιγραφή: Ο χρήστης έχει τη δυνατότητα να εκτελεί ερωτήματα στη βάση δεδομένων.

Δράστες: Χρήστης

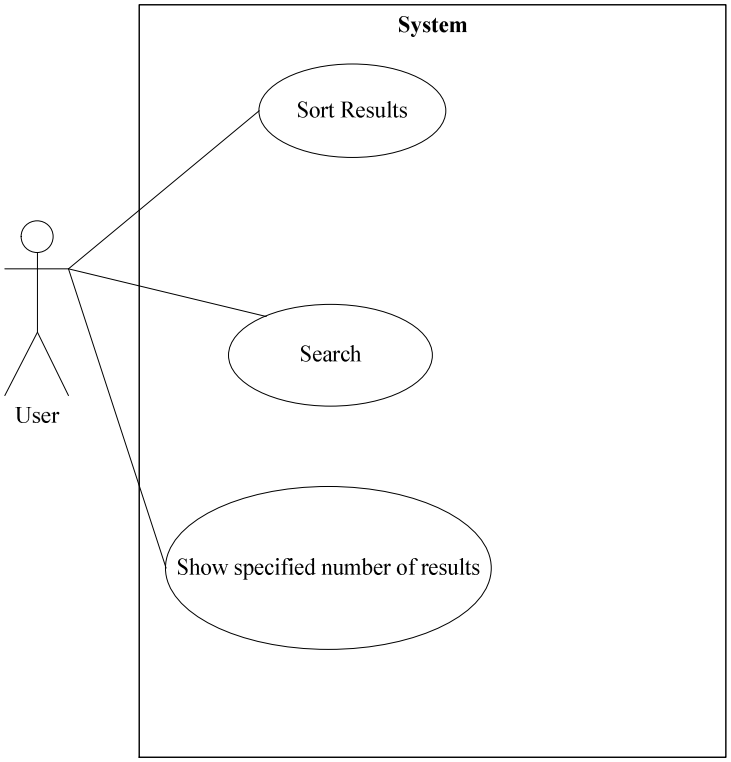
Προϋποθέσεις: Χρήστης συνδεδεμένος , Σύνδεση με βάση, Επιτυχής εκτέλεση ερωτήματος.

Συνθήκη εκκίνησης: -

Κείμενο σεναρίου:

1. Στην αρχική σελίδα εμφανίζονται τα αποτελέσματα σε μορφή πίνακα.
2. Ο χρήστης έχει τη δυνατότητα να επιλέξει να βλέπει λιγότερα αποτελέσματα, να τα ταξινομεί με βάση τα διάφορα πεδία και να κάνει αναζήτηση.

Εναλλακτικές Πορείες:



4.3.13 PrefDB13 – Εκτέλεση εντολών σε κονσόλα

Περιγραφή: Ο χρήστης έχει τη δυνατότητα να εκτελεί μέσω κονσόλας οποιαδήποτε εντολή προς τη βάση δεδομένων.

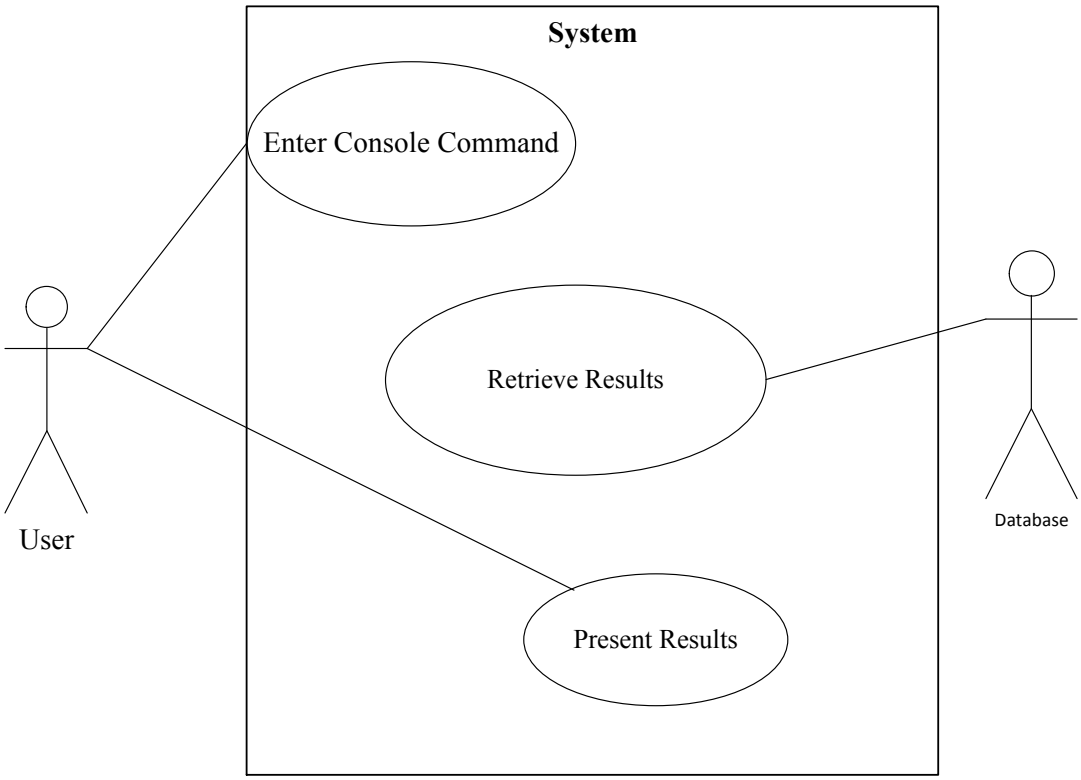
Δράστες: Χρήστης, Βάση δεδομένων

Προϋποθέσεις: Χρήστης συνδεδεμένος, Σύνδεση με βάση.

Συνθήκη Εκκίνησης: Ο χρήστης επιλέγει «Submit» στο κομμάτι input της κονσόλας.

Κείμενο Σεναρίου:

1. Τα αποτελέσματα εμφανίζονται στο κομμάτι output της κονσόλας



4.4 Μοντέλο Δεδομένων

Θα παρουσιαστεί στη συνέχει το μοντέλο δεδομένων, στην πλευρά του client και στην πλευρά του server. Τα δύο αυτά μοντέλα είναι διαφορετικά, καθώς στο server υπάρχει μια βάση δεδομένων για την οποία εξυπηρετεί σχεσιακό μοντέλο, ενώ στον client που υπάρχει ο browser και μια αντικειμενοστραφής γλώσσα, θα χρησιμοποιηθεί ένα class diagram για την περιγραφή.

4.4.1 Server Side

μοντέλο οντοτήτων συσχετίσεων σε σχεσιακό μοντέλο πήραμε την ακόλουθη δομή πινάκων:

Prefdb_users – **username**, password, role, name

Prefdb_functions – **id**,name,arg_type,definition

Prefdb_preferences–**id**,name, db_table , condition , scoring_type , score , scoring_attribute , confidence ,scoring_function(prefdb_functions.name)

Prefdb_profiles – **id**, name, type, owner,user_referring

Prefdb_preferences_profiles – pref_id(prefdb_preferences.id), profile_id (profiles.id)

Prefdb_query_config – **id**,filtering_type,num_results , score_threshold , confidence_threshold , owner (prefdb_users.username), datetime, algorithm, preference_selection_type, query, name

Prefdb_query_details – **query_id,** operation_order, operation_type, time_spent, non_pref_input_size, pref_input_size, non_pref_output_size, pref_output_size

Prefdb_query_log – **query_id,** query, filtering_type, num_results, socre_threshold, confidence_threshold, algorithm, preference_selection_type, owner(prefdb_users.username), datetime, results, query_log, non_pref_time, pref_time, index_time, opt_time, total_time, non_pref_input_size, pref_input_size, total_input_size, non_pref_output_size, pref_output_size, total_output_size

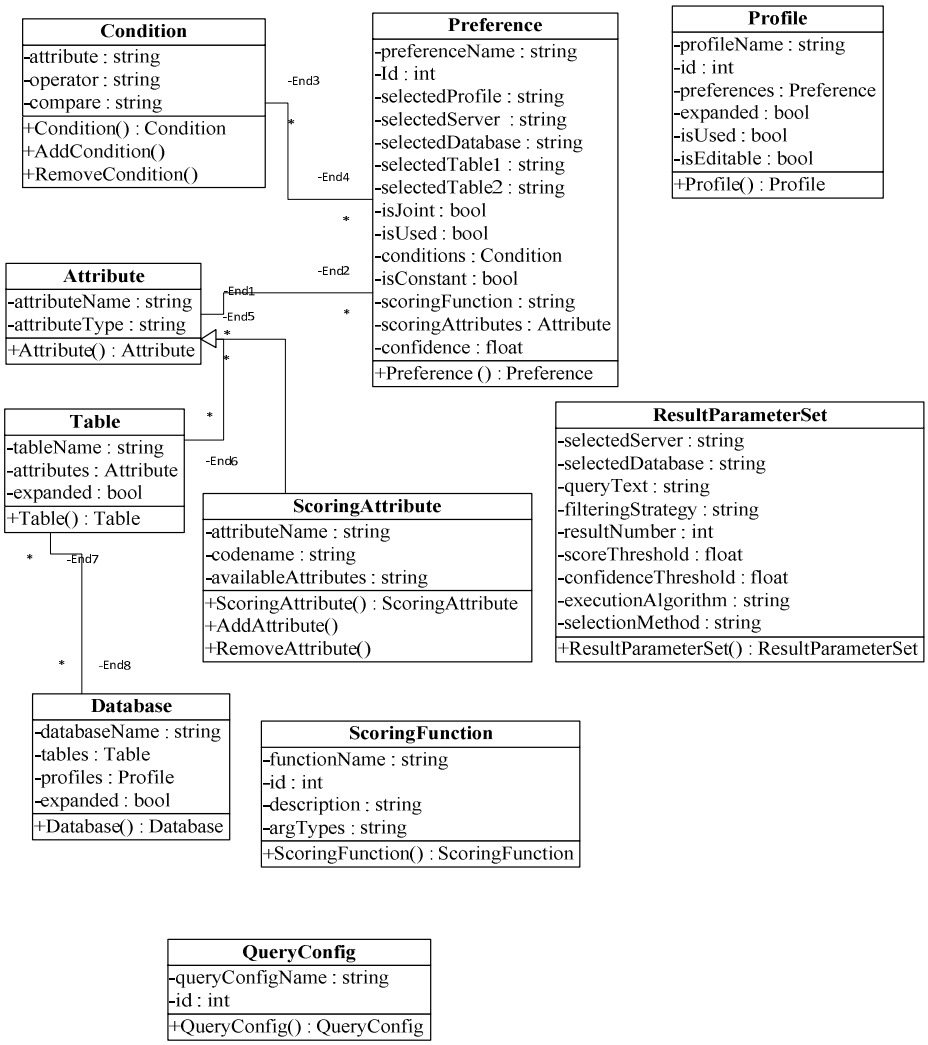
Prefdb_query_preferences – **query_id** (prefdb_query_config.query_id), **pref_id** (prefdb_preferences.id)

Prefdb_query_profiles – **query_id** (prefdb_query_config.query_id), **prof_id** (prefdb_profiles.id)

Prefdb_query_ratings – **query_id, result_id,** rating, owner(prefdb_users.username), datetime

Οι βασικές σχέσεις ανταποκρίνονται μια προς μια στις βασικές οντότητες που διαχειρίζεται το σύστημα ,ενώ οι επόμενες δίνουν τις επιπλέον πληροφορίες που απαιτούνται για τη συσχέτιση μεταξύ τους, χρησιμοποιώντας foreign keys.

4.4.2 Client Side



Στην client side χρησιμοποιείται αυτή η δομή κλάσεων , η οποία επιτρέπει τη σωστή οργάνωση των δεδομένων μετά την ανάκτησή τους από το server.

5 Σχεδίαση Συστήματος

Στη συνέχεια θα γίνει ανάλυση της σχεδίασης του συστήματος, με βάση τις πλατφόρμες στις οποίες αυτό υλοποιήθηκε.

5.1 Αρχιτεκτονική

Το σύστημα, ως διαδικτυακή εφαρμογή, αποτελείται από δύο βασικά υποσυστήματα. Το client side και το server side. Η ανάπτυξη του client side έγινε ακολουθώντας το πρότυπο Model-View-Viewmodel, κατά το οποίο, το σύστημα χωρίζεται σε τρία κομμάτια: Το Model, όπου υπάρχει το μεγαλύτερο κομμάτι της εσωτερικής λογικής, οι κλάσεις και οι μέθοδοί τους , το View που αποτελεί την απεικόνιση των γραφικών στοιχείων στο χρήστη, το σχεδιασμό και την τοποθέτησή τους, και το ViewModel το οποίο παρουσιάζει με κατάλληλο τρόπο τα δεδομένα του Model ώστε να μπορούν να απεικονιστούν στο View. Το client side επικοινωνεί με το server side μέσω ενός API το οποίο περιλαμβάνει κατάλληλες μεθόδους ώστε να παρέχονται στο client side όλες οι απαιτούμενες πληροφορίες. Θα γίνει εδώ μια σύντομη περιγραφή των κλάσεων ή των στοιχείων του κάθε τμήματος του client side, καθώς και αναφορά των συναρτήσεων του server side.

A) Model

1. Preference

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει μια προτίμηση.

2. Profile

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει ένα προφίλ.

3. Attribute

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει ένα χαρακτηριστικό σε πίνακα βάσης δεδομένων.

4. Table

Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει έναν πίνακα σε βάση δεδομένων.

5. Database

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει μια βάση δεδομένων.

6. Scoring Function

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει και μια συνάρτηση βαθμολόγησης.

7. Scoring Attribute

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει ένα χαρακτηριστικό που παίρνει ως όρισμα η συνάρτηση βαθμολόγησης.

8. Condition

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει μια συνθήκη με βάση την οποία επιλέγονται οι πλειάδες που επηρεάζονται από μια προτίμηση.

9. ResultParameterSet

   Το κάθε στοιχείο αυτής της κλάσης αντιπροσωπεύει ένα σύνολο παραμέτρων που απαιτούνται για να γίνει η εκτέλεση ενός ερωτήματος.

10. QueryConfig

    Το κάθε στοιχείο αυτής της κλάσης περιέχει τα στοιχεία που χρειάζονται για να ανακαλεστούν οι ρυθμίσεις εκτέλεσης ερωτήματος για ένα αποθηκευμένο ερώτημα.

B) View

1. MkTree

   Μετατρέπει μια HTML λίστα σε δενδρική δομή, δίνοντας τη δυνατότητα για πλοήγηση στο δέντρο με κουμπιά για επέκταση και συρρίκνωση των κόμβων του.

2. Dropdown Menu

   Μετατρέπει μια HTML λίστα σε δομή μενού, όπου το πρώτο επίπεδο είναι ένα οριζόντιο μενού, ενώ το $2^{ο}$ ένα dropdown για καθεμία από τις επιλογές του $1^{ου}$.

3. Jquery Window

   Παίρνει ένα HTML div και το εμφανίζει ως popup window , προσθέτοντάς κουμπιά ελέγχου.

4. Jquery Datatable

Δημιουργεί έναν πίνακα με δεδομένα από HTML, δίνοντας δυνατότητα για πλοήγηση, ταξινόμηση, επίδειξη περισσότερων ή λιγότερων αποτελεσμάτων, προσθαφαίρεση σειρών.

C) ViewModel

1. Preference ViewModel

   Το ViewModel για την ενεργή προτίμηση. Ανανεώνεται όταν ο χρήστης επιλέξει να τροποποιήσει κάποια προτίμηση ή να δημιουργήσει νέα.

2. Serverdata ViewModel

   Το ViewModel για τη δενδρική δομή του server. Κάθε φορά που γίνεται κλήση προς τον server, ανανεώνεται. Είναι συνδεδεμένο με ένα jQuery template στο view ώστε να γίνεται αυτόματα η ανανέωση της ιεραρχικής λίστας που μετατρέπεται σε δέντρο.

3. dbSelectViewModel

   Το ViewModel για τον επιλεγμένο server και βάση δεδομένων.

4. ResultParametersViewModel

   Το ViewModel που αποθηκεύει τις ρυθμίσεις του τελευταίου query που έτρεξε, καθώς και αυτές που έχει επιλεγμένες ο χρήστης.

5. ResultsViewModel

   Το ViewModel που αποθηκεύει τα αποτελέσματα των queries και τα μετατρέπει σε μορφή κατανοητή για το Datatable.

6. QueryconfigViewModel

   Το ViewModel που αποθηκεύει πληροφορίες για το ενεργό σύνολο ρυθμίσεων, χρησιμοποιείται όταν δημιουργείται νέο σύνολο ρυθμίσεων ή όταν φορτώνεται υπάρχον.

7. ProfileViewModel

   Το ViewModel που αποθηκεύει πληροφορίες για το ενεργό προφίλ, χρησιμοποιείται όταν δημιουργείται νέο προφίλ ή όταν τροποποιείται υπάρχον.

5.2 Περιγραφή Κλάσεων

Θα ακολουθήσει αναλυτικότερη περιγραφή των κλάσεων, των ιδιοτήτων τους καθώς και των πιο σημαντικών μεθόδων τους. Επίσης θα παρατεθεί το documentation για το API με το οποίο γίνεται η επικοινωνία με το server side. Πρέπει να σημειωθεί εδώ πως στο view, η συμπεριφορά των κλάσεων εξαρτάται εν μέρει και από τα css, ορισμένες δηλαδή από τις ενέργειες που συμβαίνουν δεν είναι προφανείς μόνο από τη ροή του κώδικα. Οι συναρτήσεις του Model που χρήζουν εξήγησης αλλά δεν υπάγονται σε κάποια κλάση θα παρατεθούν στο κομμάτι του Model, υπό το γενικό "Main". Στο κομμάτι του View θα αναφερθούν οι συναρτήσεις που χρησιμοποιούνται από καθένα από αυτά τα frameworks. Στο ViewModel, σημειώνεται πως κάθε ViewModel έχει μόνο ένα αντικείμενο της κλάσης του.

A)Model

1. Preference

   Properties:

   preferenceName – string, το όνομα της προτίμησης.

   Id - int, ο κωδικός της προτίμησης.

   selectedProfile – το προφίλ στο οποίο υπάγεται η προτίμηση.

   selectedServer – string,ο server στον οποίο υπάγεται η προτίμηση.

   selectedDatabase – string,η βάση στην οποία υπάγεται η προτίμηση.

   selectedTable1 – string,ο πρώτος πίνακας που χρησιμοποιεί η προτίμηση.

   selectedTable2 – string,ο δεύτερος πίνακας που χρησιμοποιεί η προτίμηση.

   isJoint – string, αν η προτίμηση είναι σύνθετη, δηλαδή χρησιμοποιεί δύο πίνακες έχει την τιμή "Joint" , αλλιώς "Simple".

   isUsed – Boolean, true αν η προτίμηση έχει επιλεγεί από το χρήστη.

   conditions – πίνακας από conditions, οι συνθήκες που καθορίζουν σε ποιες πλειάδες είναι εφαρμόσιμη η προτίμηση.

   isConstant – string, αν η προτίμηση αποδίδει σταθερό score στις πλειάδες στις οποίες εφαρμόζεται έχει την τιμή "Constant" , ενώ αν το κάνει με βάση κάποια συνάρτηση βαθμολόγησης "Variable".

   scoringFunction – string, το όνομα της συνάρτησης βαθμολόγησης που χρησιμοποιεί η προτίμηση (undefined αν δεν υπάρχει).

   scoringAttributes – πίνακας από scoringAttributes, περιέχει αυτά που χρησιμοποιεί για τη βαθμολόγηση η συνάρτηση βαθμολόγησης.

   confidence - float που δείχνει την εμπιστοσύνη.

Methods:

Preference (serverName,databaseName,profileName,id,name,table, condition ,scoringType,score,scoringFunction,scoringAttribute,confidence) – Constructor για την προτίμηση.

2. Profile

   Properties:

   profileName – string, το όνομα του προφίλ.

   Id – int , ο κωδικός του προφίλ.

   preferences – πίνακας από preferences , οι προτιμήσεις που ανήκουν σε αυτό το προφίλ.

   isUsed – Boolean, true αν το προφίλ έχει επιλεγεί από το χρήστη.

   expanded – Boolean, true αν ο κόμβος που αντιστοιχεί στο προφίλ σε ένα από τα δέντρα πλοήγησης έχει επεκταθεί τουλάχιστον μια φορά.

   isEditable – Boolean, true αν το προφίλ και οι προτιμήσεις του μπορούν να αλλαχθούν από τους χρήστες του συστήματος.

   Methods

   Profile(id,name,type) – Constructor για το προφίλ.

   Profile.isUsed.subscribe(newValue) – Συνάρτηση που καλείται όταν αλλάξει η τιμή του isUsed για το προφίλ. Κάνει την τιμή του isUsed σε όλες τις προτιμήσεις που ανήκουν σε αυτό το προφίλ ίση με newValue.

3. Attribute

   Properties:

   attributeName – string, το όνομα του χαρακτηριστικού.

   attributeType – string, ο τύπος που έχει το χαρακτηριστικό στη βάση δεδομένων.

   Methods:

   Attribute(attrname,attrtype) – Constructor για το χαρακτηριστικό

4. Table

   Properties:

   tableName – string, το όνομα του πίνακα.

   attributes – πίνακας από attributes που ανήκουν σε αυτό τον πίνακα.

   expanded - Boolean, true αν ο κόμβος που αντιστοιχεί στον πίνακα στο δέντρο πλοήγησης κατά πίνακες έχει επεκταθεί τουλάχιστον μια φορά.

Methods:

Table(tabname) – Constructor για τον πίνακα.

5. Database

   Properties:

   databaseName – string, το όνομα της βάσης.

   tables – πίνακας από πίνακες που ανήκουν σε αυτή τη βάση.

   profiles – πίνακας από προφίλ που ανήκουν σε αυτή τη βάση.

   expanded – Boolean, true αν ο κόμβος που αντιστοιχεί στη βάση σε ένα από τα δέντρα πλοήγησης έχει επεκταθεί τουλάχιστον μια φορά.

6. Scoring Function

   Properties:

   functionName – string, το όνομα της συνάρτησης.

   Id – int , ο κωδικός της συνάρτησης.

   Description – string, ο ορισμός της συνάρτησης.

   argTypes – πίνακας από strings, οι τύποι των ορισμάτων της συνάρτησης.

   Methods:

   ScoringFunction(id,name,args,definition) – Constructor για τη συνάρτηση.

7. Scoring Attribute

   Properties:

   attributeName – string, το όνομα του χαρακτηριστικού.

   codename – string, ο κωδικός του μέσα στη συνάρτηση, στη μορφή $i.

   availableAttributes – πίνακας από strings, αποθηκεύει τα χαρακτηριστικά από τα οποία μπορεί να διαλέξει ο χρήστης.

   Methods:

   ScoringAttribute(attr,codename) – Constructor για το χαρακτηριστικό.

   addAttribute() – Προσθέτει ένα νέο χαρακτηριστικό στον πίνακα στον οποίο ανήκει το χαρακτηριστικό που την καλεί.

   RemoveAttribute() – Αφαιρεί το χαρακτηριστικό που την καλεί από τον πίνακα στον οποίο ανήκει.

8. Condition

   Properties:

attribute – string, το χαρακτηριστικό στο οποίο εφαρμόζεται η συνθήκη.

operator – enum availableOperator, ο τελεστής με τον οποίο γίνεται η σύγκριση.

compare – string, η τιμή με την οποία γίνεται η σύγκριση.

Methods:

Condition(attr,operator,comp) – Constructor για τη συνθήκη.

addCondition() – Προσθέτει μια νέα συνθήκη στον πίνακα στον οποίο ανήκει η συνθήκη που την καλεί.

removeCondition() – Αφαιρεί τη συνθήκη που την καλεί από τον πίνακα στον οποίο ανήκει.

9. ResultParameterSet

   Properties:

   selectedServer – string, ο διακομιστής που έχει επιλεγεί

   selectedDatabase – string, η βάση που έχει επιλεγεί

   queryText – string, το κείμενο του ερωτήματος.

   filteringStrategy – enum filteringStrategies, καθορίζει τη στρατηγική επιλογής αποτελεσμάτων.

   resultNumber – int, ο αριθμός των αποτελεσμάτων που θα επιστραφούν.

   scoreThreshold – float, το ελάχιστο score που πρέπει να έχουν τα αποτελέσματα.

   confidenceThreshold – float, το ελάχιστο confidence που πρέπει να έχουν τα αποτελέσματα

   executionAlgorithm - enum executionAlgorithms, ο αλγόριθμος με τον οποίο θα γίνει η εκτέλεση του ερωτήματος.

   selectionMethod – enum selectionMethods, καθορίζει τον τρόπο με τον οποίο θα γίνει η επιλογή των προτιμήσεων για το ερώτημα.

   Methods:

   resultParameterSet(curQuery) – Constructor

10. QueryConfig

    Properties:

    queryConfigName – string, το όνομα του συνόλου ρυθμίσεων.

    id – int, ο κωδικός του συνόλου ρυθμίσεων.

Methods:

QueryConfig(name, id) - Constructor για το σύνολο ρυθμίσεων.

11. Main

Functions:

expandServerTreeNode(node) – Επεκτείνει έναν κόμβο κάποιου από τα δύο δέντρα πλοήγησης. Ελέγχει αν ο κόμβος έχει επεκταθεί στο παρελθόν και, αν όχι, καλεί την αντίστοιχη συνάρτηση της διεπαφής με τον διακομιστή και φέρνει τα απαραίτητα δεδομένα, τα οποία περνάνε ως ορίσματα στις επόμενες συναρτήσεις.

insertDatabasesToServer(str,serverName,executionMode)

insertProfilesToDatabase(str,serverName,databaseName,executionMode)

insertTablesToDatabase(str,serverName,databaseName,executionMode)

insertAttributesToTable(str,serverName,databaseName,tableName, executionMode)

insertPreferencesToProfile(str,serverName,databaseName,profileName, executionMode)

Σε αυτή την οικογένεια συναρτήσεων, που καλλούνται ως callback για συναρτήσεις του API, το str είναι τα δεδομένα που γυρνάνε από το server. Με κατάλληλες κλήσεις των constructors τα δεδομένα αυτά μετατρέπονται στη μορφή που τα περιμένει το ViewModel και περνάνε σε αυτό ώστε να ανανεωθεί το view των δέντρων πλοήγησης. Ανάλογα με την παράμετρο executionMode, γίνονται αρχικοποιήσεις ή ανανεώνονται στοιχεία του ViewModel πέραν αυτού που αντιστοιχεί στα δέντρα πλοήγησης.

updateScoringFunctions(str)

updateConditionAttributes1(attributeTable)

updateConditionAttributes2(attributeTable)

updateConditionTables1(tableTable)

updateConditionProfiles(profileTable)

updateAvailableDatabases(databaseTable)

updateResults(str)

updateQueryConfigs(str)

updateQueryConfigParameters(str,serverName,databaseName,queryconfigName)

Αυτή η οικογένεια συναρτήσεων καλείται όταν πρέπει να γίνει αλλαγή στους πίνακες που χρησιμοποιούνται από το ViewModel, είτε λόγω κάποιας κλήσης στο server, είτε λόγω αλλαγών στις επιλογές του χρήστη. Οι updateConditionAttributes1, updateConditionAttributes2 και updateScoringFunctions είναι οι μόνες συναρτήσεις στις οποίες υπάρχει εξάρτηση δεδομένων ανάμεσα σε Model και ViewModel, καθώς υλοποιούν κλειδώματα τα οποία τις αναγκάζουν αν κληθούν εν μέσω επικοινωνίας με το server να περιμένουν έως ότου ολοκληρωθεί και στη συνέχεια να κάνουν τις αλλαγές.

updateUsedProfiles(str,serverName,databaseName,queryId)

updateUsedPreferences(str,serverName,databaseName)

updateUsedPreferencesRecur(serverName,databaseName,profiles,i)

Όταν φορτώνεται ένα νέο Query Configuration και πρέπει να επεκταθεί όλο το δέντρο πλοήγησης για τη βάση στην οποία έγινε το load, αυτές οι συναρτήσεις καλούνται με τέτοιο τρόπο ώστε να μην υπάρχει σύγκρουση λόγω ταυτόχρονων κλήσεων στο server.

insertPrefsToProfile(id,serverName,databaseName,profileName,isEmpty)

insertPreferencesToQueryConfig(queryId,serverName,databaseName)

Καλούνται όταν δημιουργηθεί προφίλ ή σύνολο ρυθμίσεων αντίστοιχα για να αποθηκεύσουν σε αυτά μια προς μια τις προτιμήσεις που ήταν επιλεγμένες και θα έχουν αρχικά.

newPreferenceResult(str)

editPreferenceResult(str)

newFunctionResult(str)

newProfileResult(str)

editProfileResult(str)

Καλούνται μετά τη δημιουργία ή τροποποίηση του αντίστοιχου στοιχείου και ανανεώνουν τους πίνακες που χρειάζεται στο model ,ώστε να μη χρειαστεί να γίνει νέα κλήση στο server αν αυτή έχει γίνει ήδη προηγούμενα.

executeQuery()

Καλείται όταν εκτελείται ένα ερώτημα, και με βάση το κείμενο, τις παραμέτρους και τα επιλεγμένα προφίλ και προτιμήσεις, συνθέτει το τελικό κείμενο του ερωτήματος, ενώ στη συνέχεια περνάει στο viewModel τα αποτελέσματα.

B) View

1. MkTree

   convertTrees()

   Η βασική συνάρτηση που εντοπίζει τις λίστες του mkTree και τις μετατρέπει σε δέντρα με βάση τα css classes που χρησιμοποιεί. Καλείται κάθε φορά που αλλάζει κάτι σε ένα από τα δέντρα πλοήγησης

2. Dropdown Menu

   Δε χρησιμοποιεί καμία συνάρτηση, ρυθμίζεται εξολοκλήρου μέσω css.

3. Jquery Window

   createPreference()

   editPreference(event)

   createProfile()

   editProfile(event)

   saveQueryConfig()

   loadQueryConfig()

   loginUser()

   Σε όλες αυτές τις συναρτήσεις ρυθμίζονται οι αρχικές παράμετροι για την εκτέλεση των αντίστοιχων ενεργειών και παρουσιάζεται το παράθυρο για αυτές τις ενέργειες στο χρήστη, με τα αντίστοιχα πλήκτρα λειτουργιών. Οι συναρτήσεις edit έχουν επιπλέον μια παράμετρο event, η οποία τους επιτρέπει να βρουν τον κόμβο από τον οποίο έγινε η κλήση της συνάρτησης.

4. Jquery Datatable

   Ο πίνακας σχηματίζεται με βάση τις ακόλουθες ιδιότητες του βασικού αντικειμένου πίνακα , το οποίο δημιουργείται μόνο στην αρχική φόρτωση της σελίδας:

   aaData – τα δεδομένα που εμφανίζονται στον πίνακα

   bDestroy – επιτρέπει την αφαίρεση γραμμών κατά την ανανέωση του πίνακα.

   sTitle – ο τίτλος για κάθε στήλη

   mDataProp – δίνει τη δυνατότητα να διαβαστούν δεδομένα από JSON

C) ViewModel

1. Preference ViewModel

   Properties:

   preferenceName – string, το όνομα της προτίμησης.

   Id - int, ο κωδικός της προτίμησης.

   selectedProfile – το προφίλ στο οποίο υπάγεται η προτίμηση.

   selectedServer – string,ο server στον οποίο υπάγεται η προτίμηση.

   selectedDatabase – string,η βάση στην οποία υπάγεται η προτίμηση.

   selectedTable1 – string,ο πρώτος πίνακας που χρησιμοποιεί η προτίμηση.

   conditionAttributes1 – πίνακας από attributes, περιέχει τα attributes που αντιστοιχούν στο selectedTable1.

   selectedTable2 – string,ο δεύτερος πίνακας που χρησιμοποιεί η προτίμηση.

   conditionAttributes2 – πίνακας από attributes, περιέχει τα attributes που αντιστοιχούν στο selectedTable1

   intAttributes – πίνακας από τα attributes των conditionAttributes1 και conditionAttributes2 που έχουν type που μπορεί να μπει σε ακέραια μεταβλητή.

   floatAttributes – πίνακας από τα attributes των conditionAttributes1 και conditionAttributes2 που έχουν type που μπορεί να μπει σε πραγματική μεταβλητή.

   charAttributes – πίνακας από τα attributes των conditionAttributes1 και conditionAttributes2 που έχουν type που μπορεί να μπει σε μεταβλητή κειμένου.

   isJoint – string, αν η προτίμηση είναι σύνθετη, δηλαδή χρησιμοποιεί δύο πίνακες έχει την τιμή “Joint” , αλλιώς “Simple”.

   isUsed – Boolean, true αν η προτίμηση έχει επιλεγεί από το χρήστη.

conditions – πίνακας από conditions , οι συνθήκες που καθορίζουν σε ποιες πλειάδες είναι εφαρμόσιμη η προτίμηση.

isConstant – string, αν η προτίμηση αποδίδει σταθερό score στις πλειάδες στις οποίες εφαρμόζεται έχει την τιμή “Constant” , ενώ αν το κάνει με βάση κάποια συνάρτηση βαθμολόγησης “Variable”.

scoringFunction – string, το όνομα της συνάρτησης βαθμολόγησης που χρησιμοποιεί η προτίμηση (undefined αν δεν υπάρχει).

scoringAttributes – πίνακας από scoringAttributes,περιέχει αυτά που χρησιμοποιεί για τη βαθμολόγηση η συνάρτηση βαθμολόγησης.

confidence - float που δείχνει την εμπιστοσύνη.

newFunction – Boolean, true αν ο χρήστης έχει ορίσει νέα συνάρτηση βαθμολόγησης για την προτίμηση που δημιουργεί ή τροποποιεί.

newFunctionName – string, το κείμενο της νέας συνάρτησης, εφόσον αυτή υπάρχει.

newFunctionText – string, ο ορισμός της νέας συνάρτησης, εφόσον αυτή υπάρχει.

errorMessage – string, το πιο πρόσφατο μήνυμα λάθους, αν υπήρξε.

dummyScoringAttributes – πίνακας από scoringAttributes, χρησιμοποιείται προσωρινά έως ότου να ολοκληρωθούν οι κλήσεις προς τον server που επιστρέφουν τα κανονικά scoringAttributes. Χρησιμοποιείται διότι στο ViewModel, το κανονικό scoringAttributes είναι υποχρεωτικό να παίρνει τιμές από τους πίνακες availableAttributes, οι οποίοι γεμίζουν με αυτές τις κλήσεις.

dummyScoringFunction – ScoringFunction. Όπως και παραπάνω για τα scoringAttributes. Αποθηκεύεται προσωρινά η συνάρτηση βαθμολόγησης, έως ότου ολοκληρωθούν οι κλήσεις προς τον server για να μπορεί να επιλεγεί από τον πίνακα scoringFunctions.

dummyConditions – πίνακας από conditions. Καθώς οι συνθήκες χρησιμοποιούν Attributes, αποθηκεύονται προσωρινά έως ότου είναι έτοιμοι οι αντίστοιχοι πίνακες.

setDummy – Boolean, true όταν υπάρχουν dummies που περιμένουν για να αντιγραφούν στα conditions,scoringAttributes και scoringFunction.

scoringFunctionDefinition – String , κενό αν η scoringFunction είναι undefined, αλλιώς παίρνει την τιμή του scoringFunction.description

constantValueFixed – επιστρέφει την τιμή του constantValue με τα δεκαδικά ψηφία που χρειάζονται.

confidenceFixed – επιστρέφει την τιμή του confidence με τα δεκαδικά ψηφία που χρειάζονται.

Methods:

preferenceViewModel.selectedDatabase.subscribe(newDatabase) –

Όταν αλλάξει η επιλεγμένη βάση, φέρνει τους αντίστοιχους πίνακες, προφίλ και συναρτήσεις βαθμολόγησης.

preferenceViewModel.scoringFunction.subscribe(newScoringFunction) –

Όταν αλλάξει η συνάρτηση βαθμολόγησης, βάζει τον κατάλληλο αριθμό ScoringAttributes στον πίνακα scoringAttributes, και αλλάζει τα availableAttributes του καθενός ώστε να ταιριάζουν με αυτά που παίρνει η συνάρτηση ως όρισμα σε αυτή τη θέση.

preferenceViewModel.selectedTable1.subscribe(newTable1)

preferenceViewModel.selectedTable2.subscribe(newTable2)

Όαν αλλάζουν οι επιλεγμένοι πίνακες, ανανεώνουν τους πίνακες για την επιλογή χαρακτηριστικών, (conditionAttributes1, conditionAttributes2, intAttributes, floatAttributes, charAttributes) αφού πρώτα ενεργοποιήσουν τα κλειδώματα αν είναι αναγκαίο να γίνει κλήση στο server.

2. Serverdata ViewModel

   Properties:

   Servers – πίνακας από servers που αποθηκεύει το μοντέλο για τα δένδρα πλοήγησης

3. dbSelectViewModel

   Properties:

   availableServers – πίνακας από strings, οι servers από τους οποίους μπορεί να επιλέξει ο χρήστης.

   availableDatabases – πίνακας από strings, οι βάσεις από τις οποίες μπορεί να επιλέξει ο χρήστης, ανάλογα με τον server που έχει επιλεγεί.

   selectedServer – String, ο server με τον οποίο είναι συνδεδεμένος ο χρήστης.

   selectedDatabase – String, η βάση με την οποία είναι συνδεδεμένος ο χρήστης.

tempServer – String, ο server που έχει επιλέξει ο χρήστης, πριν επιβεβαιώσει την αλλαγή.

tempDatabase – string, η βάση που έχει επιλέξει ο χρήστης, πριν επιβεβαιώσει την αλλαγή.

4. ResultParametersViewModel

   Properties:

   resultParameters – ResultParameterSet, αποθηκεύει τις παραμέτρους του τελευταίου ερωτήματος που έτρεξε.

   tempResultParameters – ResultParameterSet, αποθηκεύει τις παραμέτρους που έχει επιλέξει ο χρήστης πριν πατήσει execute.

5. ResultsViewModel

   Properties:

   results - Πίνακας από Strings, αποθηκεύει τα αποτελέσματα.

   resultHeaders – Πίνακας από Strings, αποθηκεύει τις στήλες των αποτελεσμάτων.

6. QueryconfigViewModel

   Properties:

   selectedServer – String, ο server στον οποίο ανήκει το σύνολο ρυθμίσεων.

   selectedDatabase – String, η βάση στην οποία ανήκει το σύνολο ρυθμίσεων.

   queryConfigs – Πίνακας από QueryConfig, τα αποθηκευμένα σύνολα ρυθμίσεων στη βάση.

   queryConfig – QueryConfig, το επιλεγμένο σύνολο ρυθμίσεων.

   selectedProfiles – Πίνακας από strings, τα ονόματα των προφίλ που ανήκουν στο επιλεγμένο σύνολο ρυθμίσεων.

   selectedPreferences – Πίνακας από strings, τα ονόματα των προτιμήσεων που ανήκουν στο επιλεγμένο σύνολο ρυθμίσεων.

   errorMessage – string, το πιο πρόσφατο μήνυμα λάθους, αν υπήρξε.

   Methods:

   selectedDatabase.subscribe(function(newDatabase) – Όταν αλλάζει η επιλεγμένη συνάρτηση φέρνει τα αντίστοιχα σύνολα ρυθμίσεων.

7. ProfileViewModel

   Properties:

selectedServer – String, ο server στον οποίο ανήκει το προφίλ.

selectedDatabase – String, η βάση στην οποία ανήκει το προφίλ

profileName – String, το όνομα του προφίλ.

isEmpty – String, "Empty" αν στη δημιουργία προφίλ ο χρήστης έχει επιλέξει να μην έχει αρχικές προτιμήσεις.

errorMessage – string, το πιο πρόσφατο μήνυμα λάθους, αν υπήρξε.

D) Server side API

1. authenticateUser(String username, String password)

   Λειτουργία: Επαληθεύει τα στοιχεία του χρήστη.

   Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

2. getUserInfo(String hostname, String dbname, String username)

   Λειτουργία: -

   Επιστρέφει: Επιστρέφει το ρόλο του χρήστη.

3. executeQuery(String hostname, String dbname, String query, int filteringType, int numResults, double scoreThresh, double confThresh, String alg, int prefSelectionType, int optimizationOption, int indexThreshold, int numOfExps, boolean inputFromFile)

   Λειτουργία: Εκτελεί ένα ερώτημα προς τη βάση.

   Επιστρέφει: JSON string με τα αποτελέσματα σε περίπτωση επιτυχίας, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία

   Επεξήγηση παραμέτρων:

   query – Ερώτημα με προτιμήσεις, της μορφής

   "SELECT m.title, m.prod_year, r.rating, r.votes, g.genre_id \n\r" +

   "FROM MOVIES m, RATINGS r, GENRES g \n\r" +

   "WHERE m.movieid = r.movieid AND m.movieid = g.movieid AND m.prod_year > 1980 AND r.votes > 20 AND g.genre = 'Comedy' \n\r" +

   "PREFERENCE \n\r" +

   "TABLE MOVIES m \n\r" +

   "CONDITION m.prod_year > 2005.01 \n\r" +

   "SCORING_FUNCTION f_2 \n\r" +

"SCORING_ATTR m.prod_year \n\r" +

"CONF 1 \n\r" +

"PREFERENCE \n\r" +

"TABLE RATINGS r \n\r" +

"CONDITION r.rating > 6.01 \n\r" +

"SCORING_FUNCTION f_1 \n\r" +

"SCORING_ATTR r.rating \n\r" +

"CONF 1 \n\r" +

"PREFERENCE \n\r" +

"TABLE GENRES g \n\r" +

"CONDITION g.genre = 'Comedy' \n\r" +

"SCORING_FUNCTION f_3 \n\r" +

"SCORING_ATTR g.genre \n\r" +

"CONF 1 \n\r"

hostname - e.g. casablanca.dblab.ece.ntua.gr

dbname - e.g. jmdb

filteringType - 1 -> top-k scores, 2 -> top-k confidence, 3 -> skyline

alg - SQL/FP/BU/GBU/PETTA/RMA-1/…/RMA-6/

preferenceSelectionType - 1: only selected, 2 -> all relevant, 3 -> most confident

optimizationOption - 0 -> preferences directly on relations, 1-> preferences at the end, 2 -> cost-based optimization (preferred option: 2)

indexThreshold - integer >=0

numOfExps - 1

inputFromFile - false

4. createScoringFunction(String hostname, String dbname, String name, String argType, String definition)

   Λειτουργία: Δημιουργεί μια νέα συνάρτηση βαθμολόγησης.

   Επιστρέφει: Το id της νέας συνάρτησης για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

   Επεξήγηση Παραμέτρων:

argType: e.g. int, string
definition: the body of the scoring function in pg/sql syntax (everything between begin, end)

5. createProfile(String hostname, String dbname, String profileName, String owner)

   Λειτουργία: Δημιουργεί ένα νέο προφίλ.

   Επιστρέφει: Το id του νέου προφίλ για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

6. editProfile(String hostname, String dbname, int profileId, String profileName)

   Λειτουργία: Αλλάζει το όνομα ενός προφίλ.

   Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

7. removeProfile (String hostname, String dbname, int profileId)

   Λειτουργία: Διαγράφει ένα προφίλ.

   Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

8. createPreference(String hostname, String dbname, String name, String dbTable, String condition, int scoringType, double score, String scoringFunction, String scoringAttribute, double conf, int profileId)

   Λειτουργία: Δημιουργεί μια νέα προτίμηση.

   Επιστρέφει: Το id της νέας προτίμησης για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

   Επεξήγηση παραμέτρων:

   scoringType: 1 -> constant, 2 -> variable

   scoringFunction: function name, use ‘constant0 – constant1.0’ for constant

9. editPreference(String hostname, String dbname, String name, String dbTable, String condition, int scoringType, double score, String scoringFunction, String scoringAttribute, double conf, int prefId)

   Λειτουργία: Τροποποιεί μια προτίμηση.

   Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

10. removePreference(String hostname, String dbname, int prefId)

    Λειτουργία: Διαγράφει μια προτίμηση.

    Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

11. addPreferenceToProfile(String hostname, String dbname, int prefId, int profileId)

    Λειτουργία: Προσθέτει μια ήδη υπάρχουσα προτίμηση σε ένα προφίλ διαφορετικό από αυτό στο οποίο είναι ήδη.

    Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

12. removePreferenceFromProfile(String hostname, String dbname, int prefId, int profileId)

    Λειτουργία: Αφαιρεί μια προτίμηση από ένα προφίλ.

    Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

13. getScoringFunctions(String hostname, String dbname)

    Λειτουργία: -

    Επιστρέφει: JSON string με τις συναρτήσεις βαθμολόγησης της βάσης.

14. getPreferencesByProfile(String hostname, String dbname, int profileId)

    Λειτουργία: -

    Επιστρέφει: JSON string με τις προτιμήσεις που αντιστοιχούν στο προφίλ.

15. getProfilesByOwner(String hostname, String dbname, String owner)

    Λειτουργία: -

    Επιστρέφει: JSON string με τα προφίλ της βάσης που έχει φτιάξει ο συγκεκριμένος χρήστης.

16. getDatabases(String hostname)

    Λειτουργία: -

    Επιστρέφει: JSON string με τις βάσεις δεδομένων του server.

17. getTables(String hostname, String dbname)

    Λειτουργία: -

    Επιστρέφει: JSON string με τους πίνακες της βάσης.

18. getAttributes(String hostname, String dbname, String table)

    Λειτουργία: -

Επιστρέφει: JSON string με τα attributes του πίνακα.

19. getQueryConfigs(String hostname, String dbname, String owner)

    Λειτουργία: -

    Επιστρέφει: JSON string με τα αποθηκευμένα ερωτήματα που έχει κρατήσει ο συγκεκριμένος χρήστης.

20. saveQueryConfig(String hostname, String dbname, String queryAlias, String queryString, int filteringType, int numResults, double scoreThresh, double confThresh, String alg, int prefSelectionType, String owner)

    Λειτουργία: Αποθηκεύει ένα ερώτημα στη βάση.

    Επιστρέφει: Το id του νέου ερωτήματος για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

21. saveQueryConfigProfiles(String hostname, String dbname, int queryId, int profileId)

    Λειτουργία: Προσθέτει ένα προφίλ σε ένα ερώτημα που είναι αποθηκευμένο στη βάση.

    Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

22. saveQueryConfigPreferences(String hostname, String dbname, int queryId, int preferenceId)

    Λειτουργία: Προσθέτει μια προτίμηση σε ένα ερώτημα που είναι αποθηκευμένο στη βάση.

    Επιστρέφει: Θετικό αριθμό για επιτυχία, μήνυμα της μορφής «ERROR: *error message*» για αποτυχία.

23. loadQueryConfig(String hostname, String dbname, int queryId)

    Λειτουργία: -

    Επιστρέφει: JSON string με τις παραμέτρους ενός ερωτήματος αποθηκευμένου στη βάση.

24. getProfilesByQueryConfig(String hostname, String dbname, int queryId)

    Λειτουργία: -

    Επιστρέφει: JSON string με τα προφίλ ενός ερωτήματος αποθηκευμένου στη βάση.

25. getProfilesByTable(String hostname, String dbname, String tablename, String username)

Λειτουργία: -

Επιστρέφει: JSON string με τα προφίλ τα οποία έχουν προτιμήσεις που αναφέρονται στον δοθέντα πίνακα.

26. getPreferencesByQueryConfig(String hostname, String dbname, int queryId)

    Λειτουργία: -

    Επιστρέφει: JSON string με τις προτιμήσεις ενός ερωτήματος αποθηκευμένου στη βάση.

27. consoleCommand(String hostname, String dbname, String username, String query)

    Λειτουργία: Θέτει ένα ερώτημα στη βάση.

    Επιστρέφει: Τα αποτελέσματα του ερωτήματος που τέθηκε στη βάση.

6 Υλοποίηση

Στο ακόλουθο κεφάλαιο θα συζητηθούν πιο αναλυτικά οι επιλογές που έγιναν για την υλοποίηση του συστήματος, τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν για να επιτευχθεί το ζητούμενο αποτέλεσμα και οι απαιτήσεις για τη λειτουργία του συστήματος.

6.1 Απαιτήσεις συστήματος

Καταρχάς, πρέπει να σημειωθεί πως η εφαρμογή είναι διαδικτυακή, κατά συνέπεια δεν απαιτείται η εγκατάσταση κάποιου αυτόνομου προγράμματος στον υπολογιστή του χρήστη, αλλά τρέχει πάνω στον browser του. Η ανάπτυξη του συστήματος αλλά και οι περισσότερες δοκιμές έγιναν με τον Mozilla Firefox (9+). Το σύστημα είναι επιπλέον συμβατό με Microsoft Internet Explorer (8+), και Google Chrome. Το μεγαλύτερο κομμάτι του cross-browser compatibility το αναλαμβάνουν τα εργαλεία που προαναφέρθηκαν, χωρίς να είναι hard-coded διαφορετική συμπεριφορά για τις ιδιομορφίες του κάθε browser. Ο HTML κώδικας της εφαρμογής είναι γραμμένος με βάση τα standards της HTML 5.0.

Για το deployment της εφαρμογής σε local server απαιτείται java 1.6, για λόγους συμβατότητας με τον postgresql driver, καθώς και το server-side API , που είναι jar γραμμένο σε JRE 6. Το deployment έγινε με Apache Tomcat 6 και η βάση που χρησιμοποιήθηκε είναι η PostgreSQL 9.1. Το DWR χρησιμοποιεί τεχνολογία servlets και τρέχει μέσα σε έναν servlet container στον tomcat.

6.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Θα δούμε εδώ τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν στο σύστημα, καθώς και γιατί τα επιλέξαμε. Τα βασικά κομμάτια είναι τα εξής: Η αρχική σελίδα – JSP, τα CSS stylesheets που καθορίζουν την εμφάνιση, τα javascript functions που την ανανεώνουν δυναμικά, ο server πάνω στον οποίο τρέχουν οι Java classes και τα servlets που χρησιμοποιούνται από το σύστημα και τέλος οι βάσεις δεδομένων με τις οποίες αλληλεπιδρά ο χρήστης.

A) JSP

Η σελίδα στην οποία γίνεται όλη η αλληλεπίδραση με το χρήστη αποτελείται από ένα JSP. Το JSP τρέχει τμήματα κώδικα Java για να κάνει κλήσεις προς τον server μέσα από τη σελίδα.

B) CSS

Για να εμφανίζεται σωστά στο χρήστη το JSP είναι απαραίτητα stylesheets τα οποία καθορίζουν τη θέση των αντικειμένων πάνω στη σελίδα. Μάλιστα, χρησιμοποιώντας το σύστημα κλάσεων και κληρονομικότητας που παρέχει το CSS, υλοποιείται και κομμάτι της λειτουργικότητας της σελίδας. Είναι σημαντικό να προκύπτει ένα καθαρό layout, βολικό για το χρήστη, με τα στοιχεία της σελίδας τοποθετημένα ώστε να ελαχιστοποιείται ο χρόνος που θα απαιτείται για τη μετάβαση από το ένα βήμα κάποιας λειτουργίας στο επόμενο.

C) JavaScript

Πίσω από το JSP τρέχουν τα javascript functions τα οποία υλοποιούν το μεγαλύτερο κομμάτι της λειτουργικότητας της σελίδας. Τα γεγονότα που προκύπτουν από την πλοήγηση του χρήστη στη σελίδα προκαλούν δυναμικά αλλαγές στα στοιχεία της σελίδας,εμφάνιση ή απόκρυψη στοιχείων της και αποστολή αιτημάτων προς τον server.

D) Direct Web Remoting

Ένα πολύ χρήσιμο προγραμματιστικό εργαλείο που αξιοποιήθηκε στο σύστημα είναι το Direct Web Remoting (DWR). Αποτελείται από ένα javascript library και ένα servlet που τρέχει πάνω στον server και δίνει τη δυνατότητα για επικοινωνία javascript functions και Java classes, κάτι εξαιρετικά χρήσιμο για τη λειτουργία του συστήματος. Το library κάνει διαφανή για το σύστημα την κλήση των μεθόδων AJAX -το οποίο χρησιμοποιεί. Οι μέθοδοι των java classes μπορούν να καλεστούν μέσα από τα javascript functions άμεσα, με συντακτικό παρόμοιο με αυτό της Java, δίνοντας ως επιπλέον παράμετρο ένα αντικείμενο javascript με ιδιότητες όπως

timeout και callback (javascript function η οποία παίρνει ως παράμετρο την τιμή που επιστρέφει η java method). Ακόμα, το DWR δίνει τη δυνατότητα στο σύστημα να πιάνει τα Java exceptions μέσα στον javascript κώδικα, επιτρέποντας να γίνεται διαχείρισή τους στην πλευρά του χρήστη και να επιστρέφονται κατάλληλα διαγνωστικά, χωρίς να εμπλέκεται ο server σε αυτή τη διαδικασία.

E) knockout.js

Το knockout.js έχει επιλεγεί για την υλοποίηση του μοντέλου Model-View-ViewModel. Τα 2 σημαντικότερα εργαλεία που προσφέρει είναι τα observable variables και τα dependentObservables. Τα observable variables ορίζονται στο ViewModel , γίνονται bound σε στοιχεία του View και επεξεργάζονται κυρίως στο Model. Η προκαθορισμένη συμπεριφορά τους είναι πως όταν αλλάζει η τιμή τους, είτε από κάποια άμεση ενέργεια του χρήστη είτε έμμεσα λόγω αλλαγών στο σύστημα, ενημερώνουν τα στοιχεία με τα οποία είναι bound, τα οποία παίρνουν νέα τιμή. Συχνά αυτή η συμπεριφορά δεν είναι επαρκής, οπότε ορίζεται μια νέα συνάρτηση-ιδιότητα του observable variable, η subscribe, η οποία παίρνει ως όρισμα τη νέα τιμή του variable και υποδεικνύει ποια πρέπει να είναι η συμπεριφορά του. Το σημαντικό προτέρημα των observable variables είναι πως μπορούν να γίνουν bound σε σχεδόν οποιοδήποτε στοιχείο του view και να το ανανεώνουν αυτόματα. Για να γίνει αυτό, χρησιμοποιείται ένα νέο HTML attribute, το data-bind, στο οποίο δηλώνονται τα bindings. Πιο συχνά χρησιμοποιούνται τα εξής:

- Value: δένει την τιμή του observable variable με το value HTML attribute του στοιχείου
- Text: δένει την τιμή του observable variable με το attribute του στοιχείου που καθορίζει το κείμενο του.
- Options: δένει την τιμή του observable variable με τις επιλογές σε κάποιο στοιχείο επιλογής.
- optionsText: καθορίζει το κείμενο που θα εμφανίζεται για κάθε επιλογή. Το κείμενο αυτό πρέπει να είναι property του αντίστοιχου option object. Αν παραλειφθεί, τότε εμφανίζεται το ίδιο το option.
- optionsCaption: καθορίζει το αρχικό κείμενο που θα εμφανίζεται πριν ο χρήστης κάνει κάποια επιλογή.
- Event: δένει κάποιο event με μια συνάρτηση.
- Enable/disable: καθορίζουν μια συνθήκη με βάση την οποία το στοιχείο ενεργοποιείται ή απενεργοποιείται αντίστοιχα.

- selectedOptions: δένει τις επιλεγμένες τιμές ενός select με μια observable variable.
- Checked: δένει μια επιλογή που μπορεί να είναι checked (radio-button, checkbox) με μια boolean observable variable.
- Foreach: χρησιμοποιείται μαζί με το jQueryTemplate για περιπτώσεις όπου το view πρέπει να επαναλάβει το γραφικό στοιχείο για κάθε στοιχείο της observable variable.
- Template: δένει κάποιο template με το στοιχείο.
- Visible: καθορίζει μια συνθήκη με βάση την οποία το στοιχείο εμφανίζεται ή εξαφανίζεται.

Τα dependentObservables ορίζονται επίσης στο viewModel και μπορούν να χρησιμοποιηθούν ως observable variables, με τη διαφορά πως η τιμή τους δεν επηρεάζεται από τις ενέργειες που γίνονται άμεσα, αλλά υπολογίζεται με βάση άλλα observable variables. Έτσι, αν παραδείγματος χάριν έχουμε τα observable variables firstName και lastName που έχουν δεθεί με δύο στοιχεία input στο View, μπορεί ένα στοιχείο text στο view να δεθεί με τη dependentObservable fullName = firstName + " " + lastName, η οποία θα ανανεώνεται αυτόματα όταν γίνεται αλλαγή στο firstName ή στο lastName.

F) jQuery Templates

Χρησιμοποιήθηκε το template plugin του jQuery για τις περιπτώσεις όπου έπρεπε να υπάρχουν προκατασκευασμένα στοιχεία τα οποία επαναχρησιμοποιούνται. Στις περισσότερες περιπτώσεις , αυτά ήταν πίνακες για κάθε στοιχείο των οποίων έπρεπε να παρουσιάζεται ένα ίδιο γραφικό στοιχείο. Αξιοποιήθηκε μόνο η βασική λειτουργικότητα των Templates, καθώς επικοινωνούσαν με το knockout.js, το οποίο αυτοματοποιούσε την εφαρμογή τους.

G) jQuery Datatables

Plugin του jQuery, το οποίο επιτρέπει την εύκολη δημιουργία διαδραστικών πινάκων, με χρήσιμα χαρακτηριστικά όπως ταξινόμηση, αναζήτηση και περιορισμό αποτελεσμάτων.

H) Server

Εδώ τρέχουν οι java classes των οποίων οι μέθοδοι καλούνται από τα javascript functions, καθώς και το servlet το οποίο αναλαμβάνει τη μεταφορά δεδομένων μεταξύ javascript functions και java classes. Επίσης, με τον server γίνεται το deployment της εφαρμογής στο δίκτυο. Χρησιμοποιήθηκε ο Apche Tomcat 6 και το JDK 6.

I) Database - PostgreSQL

Η βάση δεδομένων δεν είναι αυστηρά κομμάτι του συστήματος, καθώς το σύστημα μπορεί να συνδεθεί με οποιαδήποτε βάση, όμως το σύστημα μας προσθέτει νέους πίνακες στις βάσεις με τις οποίες συνδέεται, ώστε να υποστηρίζουν τις λειτουργίες του σχεσιακού μοντέλου με προτιμήσεις. Η τωρινή υλοποίηση υποστηρίζει σύνδεση με βάσεις PostgreSQL 9.1.

J) Πλατφόρμα Ανάπτυξης - Eclipse

Η ανάπτυξη έγινε χρησιμοποιώντας την πλατφόρμα του Eclipse, η οποία διαθέτει συντακτικό έλεγχο και αυτόματη συμπλήρωση για όλες τις γλώσσες που χρησιμοποιήθηκαν, συνεργάζεται αυτόματα με τον Tomcat για το deployment της εφαρμογής, με το JDK για να κάνει αυτόματα compile τις java classes και διευκολύνει τη διαδικασία οργάνωσης του project.

7 Έλεγχος

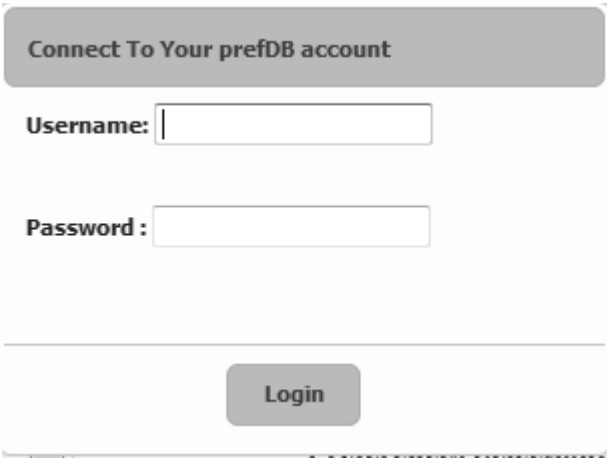
Στη συνέχεια θα παρουσιαστεί το σύστημα στην τελική του μορφή, σε συμφωνία με τις λειτουργικές και μη λειτουργικές απαιτήσεις.

7.1 Μεθοδολογία ελέγχου

Ο έλεγχος της εφαρμογής έγινε με την εκτέλεση ενός σεναρίου το οποίο περιλαμβάνει ένα προς ένα όλα τα σενάρια χρήσης που υπήρχαν στις λειτουργικές απαιτήσεις. Συγκεκριμένα ο χρήστης αρχικά συνδέεται στο σύστημα, στη συνέχεια κάνει πλοήγηση στο σχήμα του διαθέσιμου server, με όλες τις βάσεις, τα προφίλ, τους πίνακες και τις προτιμήσεις του. Έπειτα επιλέγει μια βάση δεδομένων και δημιουργεί και τροποποιεί σε αυτές προφίλ και προτιμήσεις, αποθηκεύει ένα πρότυπο ερώτημα PrefDB, φορτώνει ένα άλλο και το εκτελεί, και κάνει πλοήγηση στα αποτελέσματα . Όλα τα παραπάνω θα παρουσιαστούν με screenshots από το σύστημα.

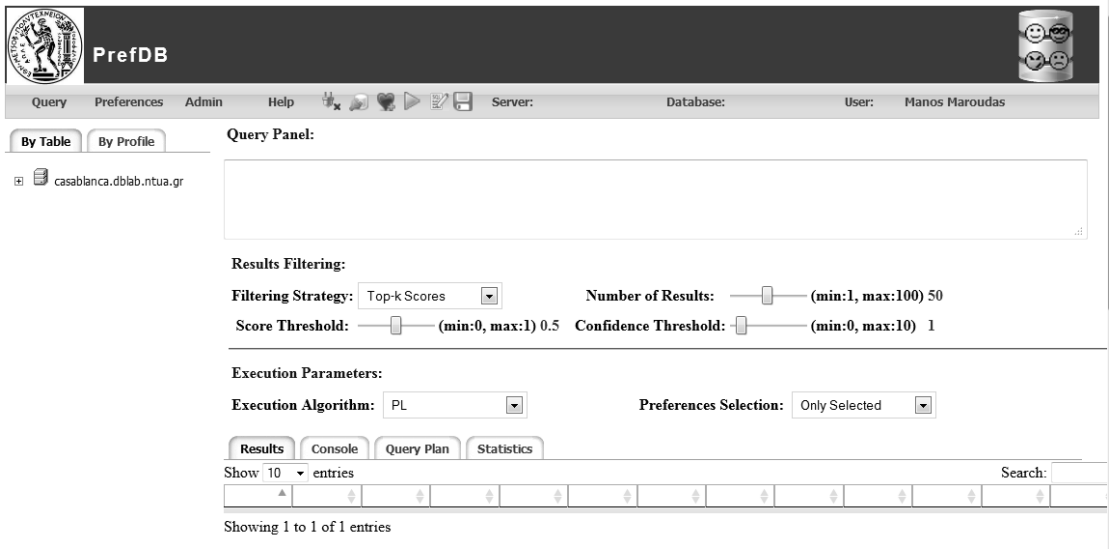
7.2 Αναλυτική παρουσίαση ελέγχου

Αρχικά ο χρήστης μπαίνει στη σελίδα της εφαρμογής και εμφανίζεται ένα popup για το login (το σύστημα δε μπορεί να χρησιμοποιηθεί από μη εγγεγραμμένους χρήστες)



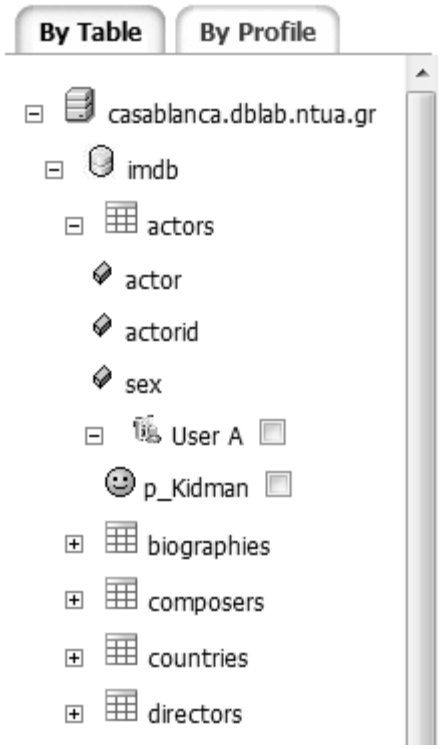
Εικόνα 1: Εισαγωγή στοιχείων χρήστη

Αφού ο χρήστης δώσει σωστά τα στοιχεία του, βλέπει την αρχική οθόνη του συστήματος:

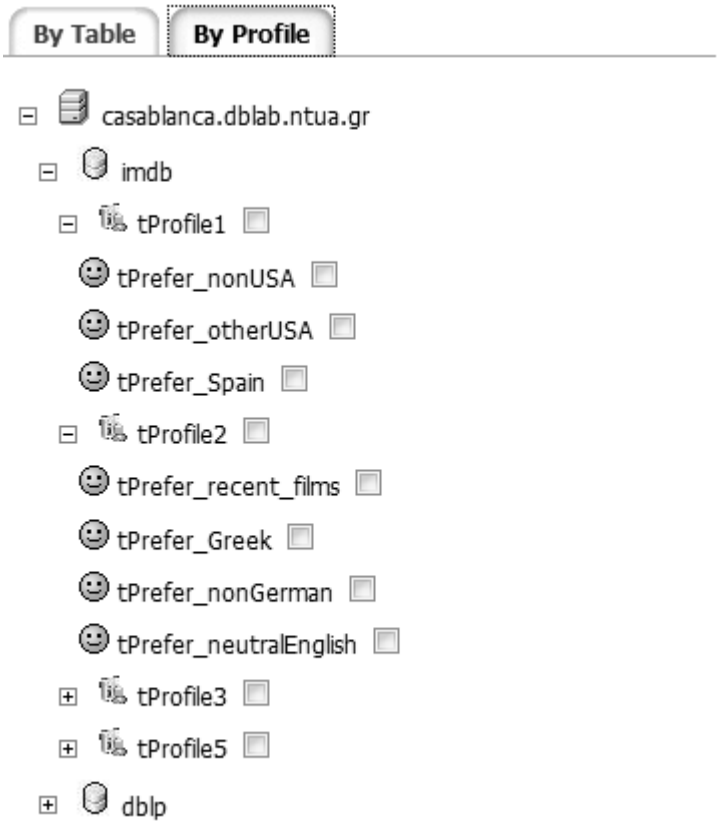


Εικόνα 2: Κεντρική σελίδα

Ο χρήστης κάνει πλοήγηση στη βάση κατά πίνακα είτε κατά προφίλ. Στην πρώτη περίπτωση, κάτω από τον κάθε πίνακα εμφανίζονται μόνο τα προφίλ και οι προτιμήσεις που σχετίζονται με αυτόν.

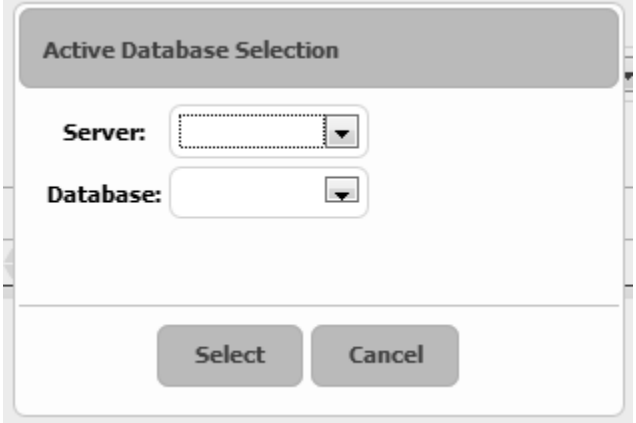


Εικόνα 3: Πλοήγηση κατά πίνακα

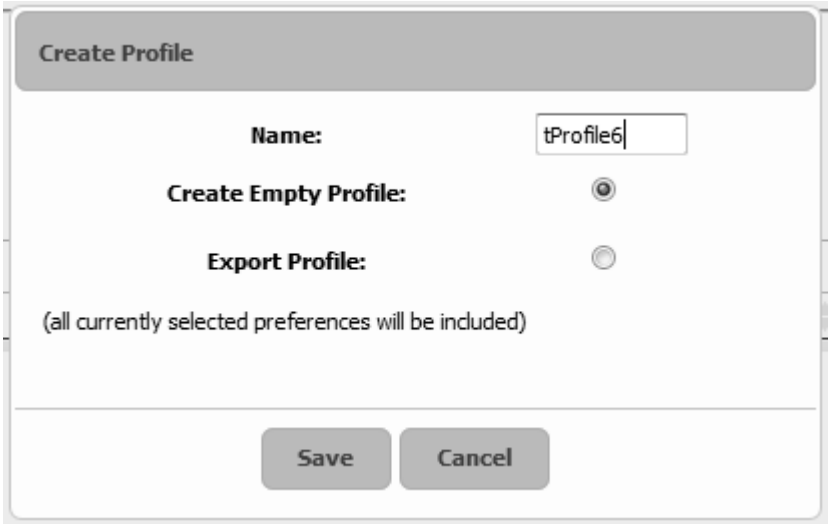


Εικόνα 4: Πλοήγηση κατά προφίλ

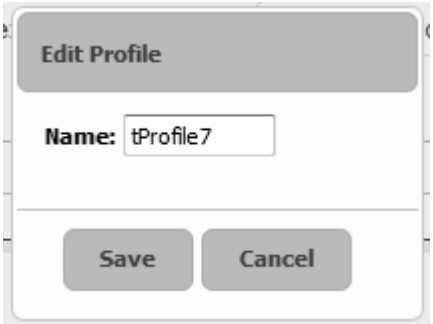
Έπειτα ο χρήστης επιλέγει τη βάση στην οποία εργάζεται, δημιουργεί ένα προφίλ και στη συνέχεια το μετονομάζει.



Εικόνα 5: Επιλογή βάσης

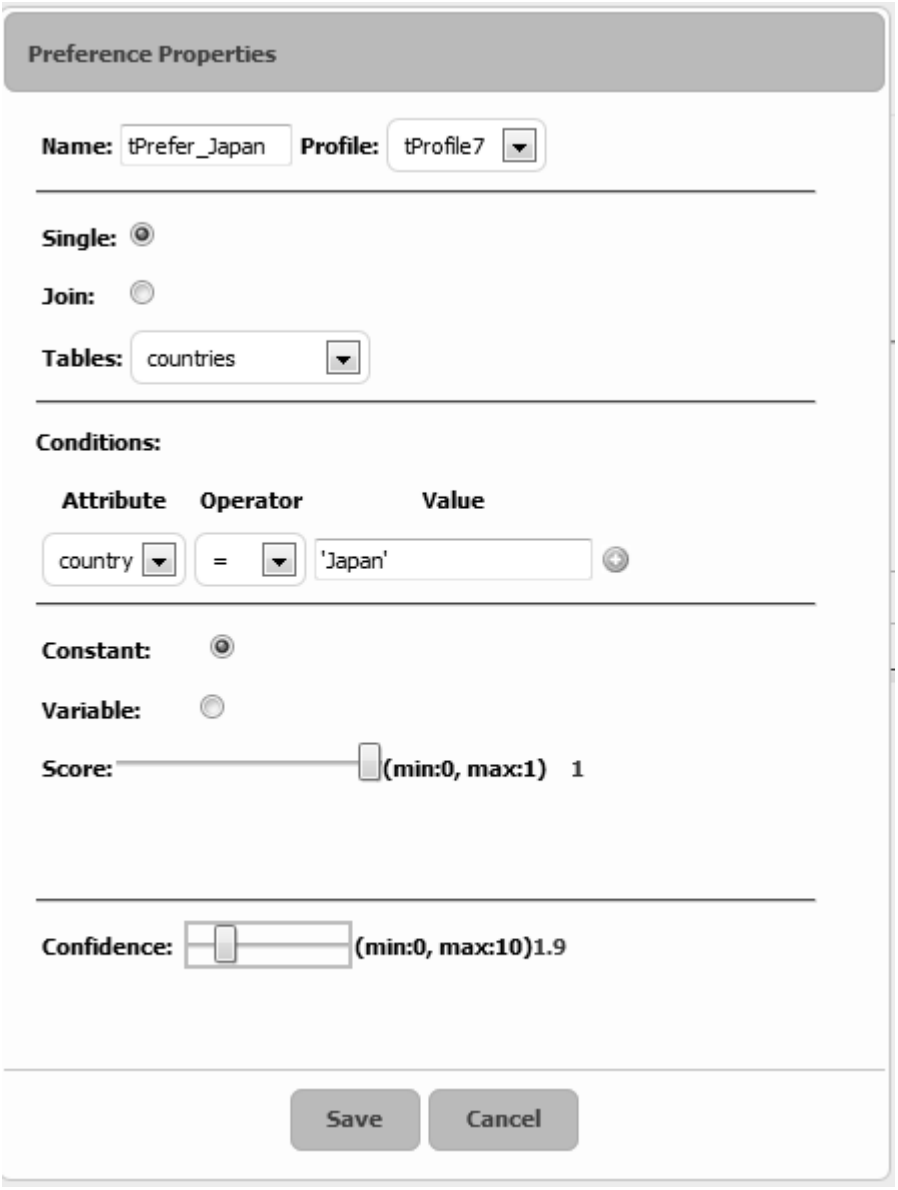


Εικόνα 6: Δημιουργία προφίλ

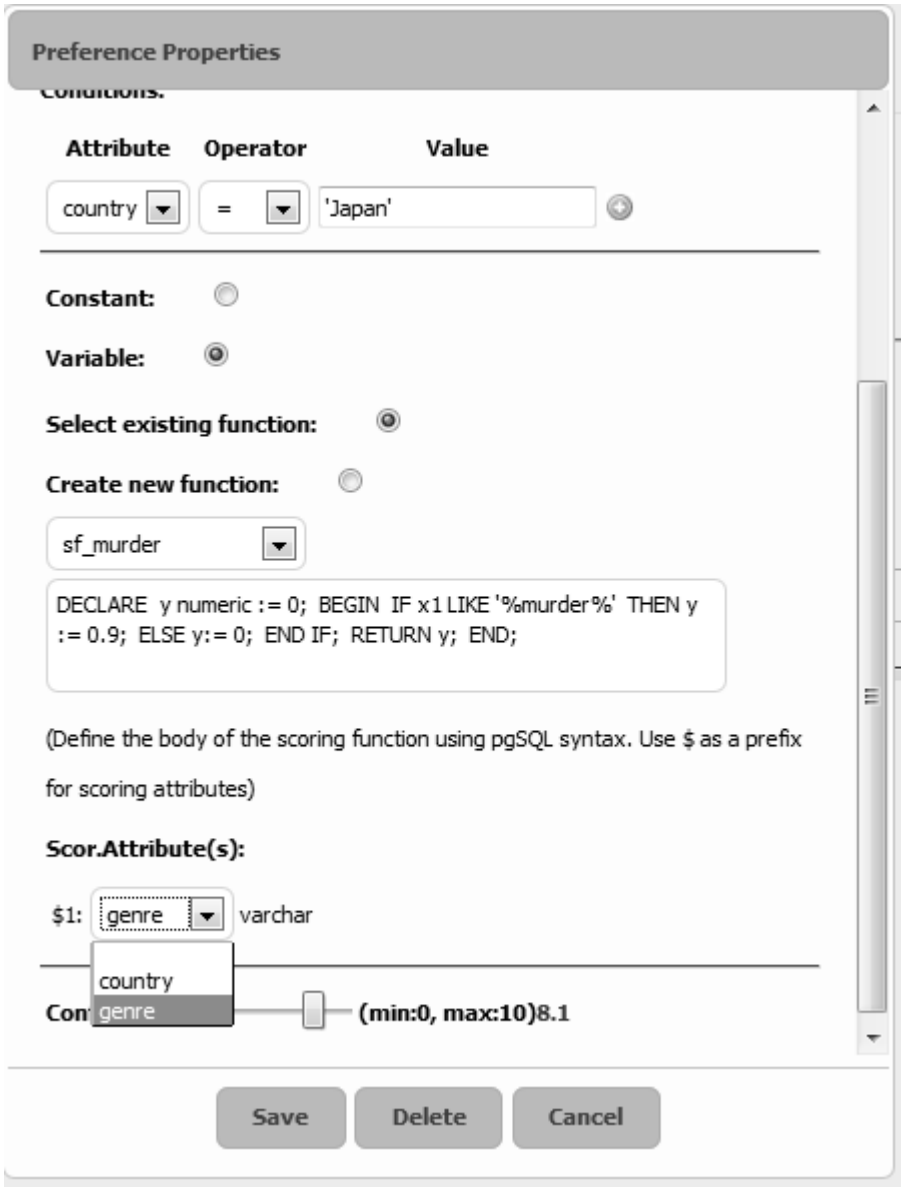


Εικόνα 7: Μετονομασία προφίλ

Προχωρόντας με το σενάριο , ο χρήστης δημιουργεί μια προτίμηση, την οποία στη συνέχεια τροποποιεί συνδέοντάς την με μια συνάρτηση βαθμολόγησης.



Εικόνα 8: Δημιουργία προτίμησης

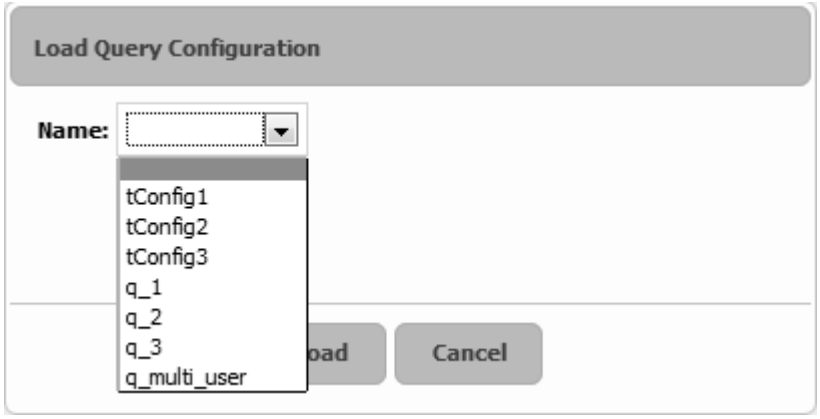


Εικόνα 9: Τροποποίηση προτίμησης

Έπειτα ο χρήστης, επιλέγοντας τη νέα και κάποιες άλλες προτιμήσεις, δημιουργεί ένα ερώτημα με παραμέτρους και το αποθηκεύει.

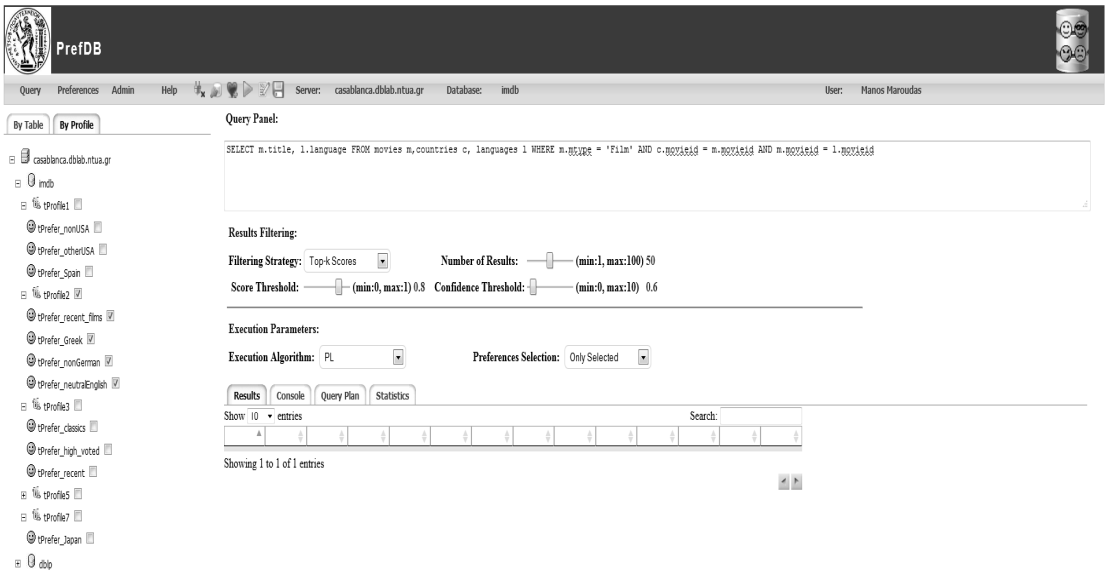
Εικόνα 10: Αποθήκευση ερωτήματος

Στη συνέχεια ο χρήστης φορτώνει ένα άλλο αποθηκευμένο ερώτημα από τη βάση και το εκτελεί.

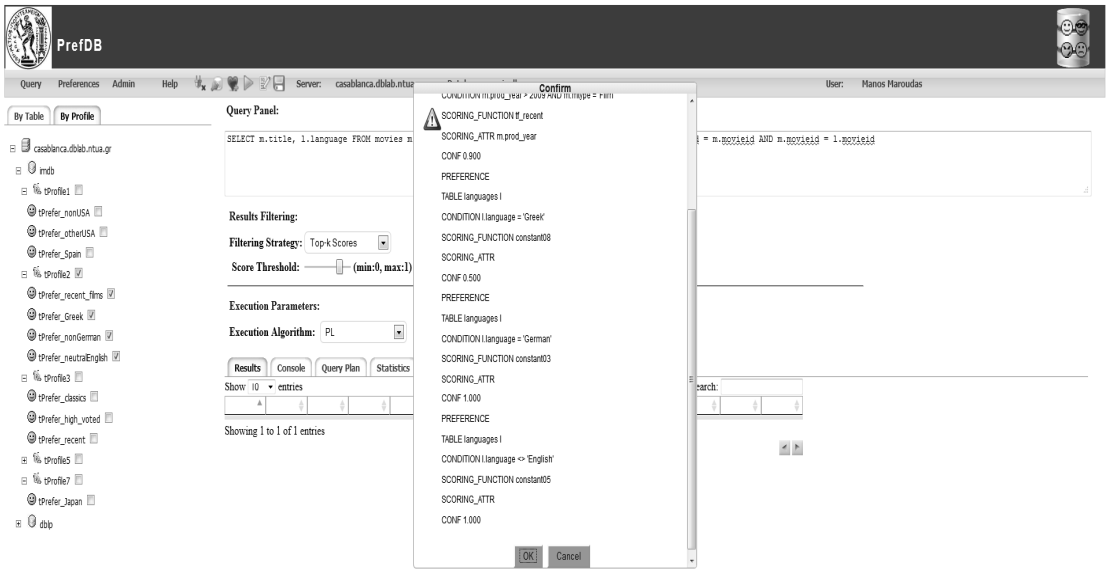


Εικόνα 11: Φόρτωση ερωτήματος

Η κεντρική σελίδα μετά τη φόρτωση του ερωτήματος:

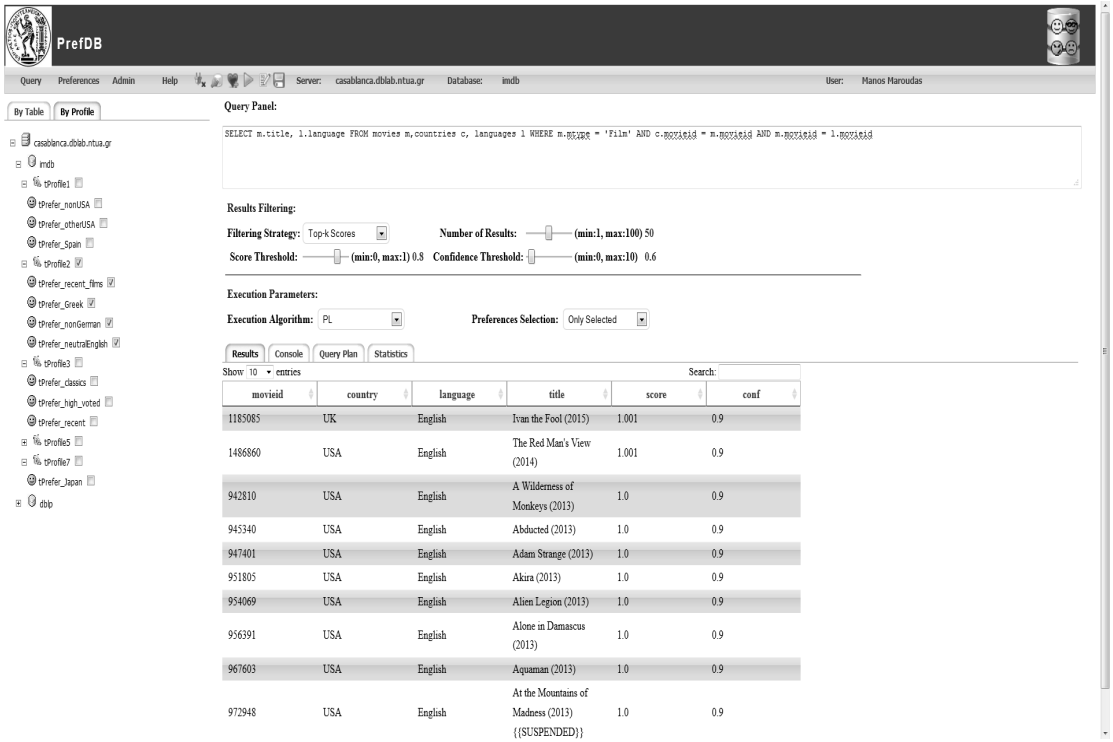


Εικόνα 12: Κεντρική σελίδα, μετά τη φόρτωση



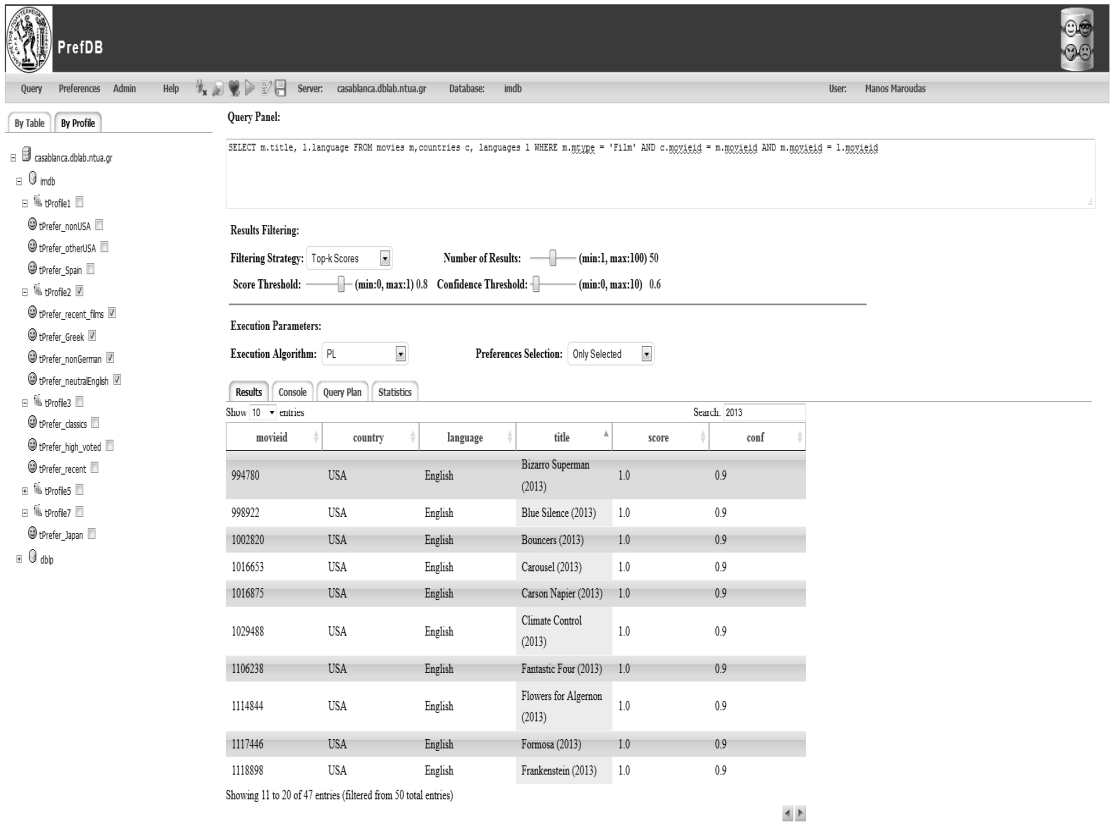
Εικόνα 13: Τελική μορφή ερωτήματος, επιβεβαίωση

Τέλος, παίρνει τα αποτελέσματα για το ερώτημα και τους αλλάζει την ταξινόμηση .



Εικόνα 14: Μη ταξινομημένα αποτελέσματα

Ταξινομημένα αποτελέσματα με βάση τον τίτλο της ταινίας:



Εικόνα 15: Ταξινομημένα αποτελέσματα με βάση τον τίτλο και με κριτήριο αναζήτησης

8 Επίλογος

8.1 Σύνοψη και συμπεράσματα

Ολοκληρώνοντας την εφαρμογή, είναι έτοιμο ένα εργαλείο που ικανοποιεί όλες τις προδιαγραφές που είχαν καθοριστεί εξαρχής για το σύστημα. Η διαδικασία των δοκιμών του συστήματος και της ανατροφοδότησης ανάλογα με τις παρατηρήσεις που προέκυψαν, βοήθησε ώστε το αποτέλεσμα να είναι ιδιαίτερα φιλικό προς το χρήστη, να είναι οικείο προς κάποιον που έχει χρησιμοποιήσει άλλα συστήματα διαχείρισης βάσεων δεδομένων και να λειτουργεί χωρίς προβλήματα.

8.2 Μελλοντικές επεκτάσεις

Μια ενδιαφέρουσα επέκταση της λειτουργικότητας του συστήματος, θα ήταν να μπορεί ο χρήστης να αποθηκεύει στα προφίλ αγαπημένες πλειάδες από κάθε βάση μετά την εκτέλεση των ερωτημάτων και το σύστημα να εξάγει από αυτή τη λίστα «αγαπημένων» προτιμήσεις του χρήστη. Εναλλακτικά ο χρήστης θα μπορούσε να βαθμολογεί στο σύνολό τους ή κατά πλειάδα τα αποτελέσματα ενός ερωτήματος ώστε να υπάρχει διαδικασία ανατροφοδότησης για το σύστημα. Επιπρόσθετα, το σύστημα θα μπορούσε να συλλέγει στατιστικά δεδομένα για τις προτιμήσεις που χρησιμοποιούν οι χρήστες στα ερωτήματα και να τα χρησιμοποιεί ως ανατροφοδότηση για το σύστημα.

9 *Βιβλιογραφία*

[AK11] Anastasios Arvanitis , Georgia Koutrika, Towards Preference-aware Relational Databases. ICDE ’12. 2012

[AW00] R. Agrawal and E. L. Wimmers, “A framework for expressing and combining preferences,” in SIGMOD, 2000, pp. 297–306.

[BKS01] Borzsonyi, Kossman, Stoker, The skyline Operator, ICDE 2001

[BW07] Nicolas Bruno, Hui Wendy Wang, The Threshold Algorithm: from Middleware Systems to the Relational Engine. IEEE Transactions on Knowledge and Data Engineering ,19(4):523-537 (2007)

[CGG+03] Jan Chomickı, Parke Godfrey, Jarek Gryz, Dongming Liang, Skyline with Presorting, ICDE 203

[Cho03] J. Chomicki, “Preference formulas in relational queries,” ACM Transactions on Database Systems, vol. 28, no. 4, pp. 427–466, 2003

[Fis99] P.C. Fishburn, Preference structures and their numerical representations. Theoretical Computer Science, 217(2):359-383 (1999)

[KK02] Werner Kießling, Gerhard Köstler, Preference SQL - Design, Implementation, Experiences, VLDB 2002:990-1001

[KI04] G. Koutrika and Y. E. Ioannidis, “Personalization of queries in database systems,” in ICDE, 2004, pp. 597–608.

[LKM10] Justin J. Levandoski, Mohamed E. Khalefa, Mohamed F. Mokbel. CareDB: A Context and Preference-Aware Location-Based Database System. PVLDB 3(2):1529-1532 (2010)

[PTFS03] Papadias, Tao, Fu, SeegerAn Optimal and Progressive Algorithm for Skyline Queries, SIGMOD 2003

[SKP11] Kostas Stefanidis, Georgia Koutrika, Evaggelia Pitoura, A Survey on Representation, Composition and Application of Preferences in Database Systems. ACM Transactions on Database Systems (TODS) 36(3):19 (2011)

[YOK04] Kyung Hoon Yang, David Olson, Jaekyung Kim, Comparison of first order predicate logic, fuzzy logic and non-monotonic logic as knowledge representation methodology, 2004, Expert Systems with Applications 27:501–519 (2004)