ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία Γραφικής Γλώσσας Αιτημάτων

Του φοιτητή

Κατσικιώτη Χρήστου

κ Χαριτίδη Κωνσταντίνου

Αρ. Μητρώου: 103574 κ 103570

Επιβλέπων καθηγητής

Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2015

Πρόλογος

Η διαχείριση δεδομένων αποτελεί αναπόσπαστο κομμάτι πολλών κλάδων αλλά και της καθημερινότητας μας. Καθώς ο όγκος των δεδομένων αυξάνεται, η χρήση του Η/Υ καθίσταται απαραίτητη. Η διαχείριση όμως των δεδομένων μέσω του Η/Υ δεν είναι και τόσο απλή, ειδικά για έναν απλό χρήστη. Για τον λόγο αυτό, αποφασίσαμε να κατασκευάσουμε μια εφαρμογή που μέσα από γνωστά στον χρήστη σύμβολα θα μπορεί να δέχεται ερωτήσεις από τον τελευταίο και θα του επιστρέφει απαντήσεις.

Περίληψη

Σκοπός της πτυχιακής μας είναι η κατασκευή μιας γραφικής γλώσσας ερωταποκρίσεων όπου ο χρήστης θα κατασκευάζει ένα ερώτημα (query) στην γλώσσα ερωτημάτων XQuery μέσα από γνωστά εικονίδια του K.O.K. (metaphors), η εφαρμογή θα χτυπάει το ερώτημα σε XML, σχεσιακό και αντικειμενοσχεσιακό μοντέλο και θα επιστρέφει τα αποτελέσματα σε μορφή XML. Αρχικά, δίνουμε μια σύντομη περιγραφή της XML, της XQuery και των διάφορων DBMS που χρησιμοποιήσαμε δίνοντας παραδείγματα. Στην συνέχεια, εξηγούμε την χρήση των metaphors, την λειτουργία της βάσης XML «Sedna», το NetBeans IDE και το PostgreSQL DBMS. Τέλος, παρουσιάζουμε την λειτουργία της εφαρμογής με παραδείγματα και βήματα για την κατανόηση της από τον απλό χρήστη. Τα metaphors του K.O.K. βασίστηκαν στην πτυχιακή εργασία του Α.Πλιάκα και Κ.Τσέκου με τίτλο «Υλοποίηση εργαλείου γραφικής απεικόνισης της XQuery».

Abstract

The objective of our thesis is the construction of a graphical query language of questions and answers which provides to user the capability to construct a question (query) in XQuery language through known icons of traffic regulations (metaphors), the application hits the query in XML, relational and object-relational model and returns results in XML form. At first, we give a brief description of XML, XQuery and the different DBMS we have used by giving examples. Then, we explain the use of the metaphors, the functionality of “Sedna” XML Database, the NetBeans IDE and the PostgreSQL DBMS. In the end, we present the function of the application with examples and through steps for the simple user to understand.

Ευχαριστίες

Η πραγματοποίηση της συγκεκριμένης πτυχιακής εργασίας δεν θα ήταν εφικτή χωρίς το Αλεξάνδρειο Τεχνολογικό Ίδρυμα της Θεσσαλονίκης, το τμήμα Πληροφορικής καθώς και τους καθηγητές που μας δίδαξαν την επιστήμη των υπολογιστών. Θα θέλαμε να ευχαριστήσουμε ιδιαίτερα τον κ. Κεραμόπουλο όχι μόνο για την συμβολή του στην διδαχή των μαθημάτων αλλά και για την υπομονή και την θέληση του να συνεργαστούμε για την εκπόνηση αυτής της πτυχιακής εργασίας.

Περιεχόμενα

Ευρετήριο σχημάτων

Ευρετήριο πινάκων

1 Ανάλυση θεωρητικών τεχνολογιών

1.1 Εισαγωγή

Ο σκοπός του πρώτου κεφαλαίου είναι ο χρήστης να μάθει και να κατανοήσει τα βασικά των τεχνολογιών που χρησιμοποιήθηκαν στη πτυχιακή εργασία. Ο χρήστης μετά την ανάγνωση του πρώτου κεφαλαίου θα γνωρίζει τα βασικά πάνω στην XML, το σχεσιακό και το αντικειμενοσχεσιακό μοντέλο βάσεων δεδομένων. Για την κατανόηση αυτού του κεφαλαίου δεν χρειάζονται ιδιαίτερες προαπαιτούμενες γνώσεις παρά μόνο βασικές γνώσεις στον τομέα της πληροφορικής.

1.2 XML

1.2.1 Τι είναι η XML;

Η eXtensible Markup Language (XML) είναι ένα απλό και ανοικτό πρότυπο-γλώσσα αποθήκευσης καλά δομημένων δεδομένων (έγγραφα, βιβλία, αγορές και άλλα) βασισμένο σε κείμενο.[1,2,3,4] Σχεδιάστηκε με κύριο γνώμονα την απλότητα και την χρήση μέσω του διαδικτύου [5] και εστιάζει στον τρόπο δόμησης των δεδομένων και όχι στην παρουσίαση τους. Η XML δημιουργήθηκε από μέλη του World Wide Web Consortium (W3C) [6] και είναι απόγονος και υποσύνολο της γλώσσας SGML. Τον Φεβρουάριο του 1998 κυκλοφόρησε επίσημα η πρώτη έκδοση της γλώσσας [7].

1.2.2 Δομή της XML

Η δομή της XML είναι πολύ απλή και εύκολη στην χρήση. Η αναπαράσταση των δεδομένων γίνεται με την χρήση δυο τύπων αντικειμένων. Ο ένας τύπος αντικειμένου είναι το στοιχείο (node) και ο δεύτερος είναι η ιδιότητα (attribute) [8].

Θα μπορούσαμε να παρομοιάσουμε το στοιχείο με ένα παιδί π.χ.. τον Νίκο και την ιδιότητα με ένα από τα χαρακτηριστικά του παιδιού π.χ.. το ύψος. Κατευθείαν μπορούμε να συμπεράνουμε πως ένα στοιχείο μπορεί να έχει πολλές ιδιότητες, στην δικιά μας την περίπτωση κάποια άλλη ιδιότητα θα μπορούσε να είναι το βάρος, το φύλο κ.α. Επίσης, ίσως να θέλαμε να αποθηκεύσουμε και άλλα στοιχεία για το παιδί με παραπάνω πληροφορίες π.χ.. το μπλουζάκι που φοράει και τι χρώμα είναι αυτό, τι υλικό είναι και άλλα. Η XML μας δίνει την δυνατότητα να το κάνουμε αυτό ορίζοντας την μπλούζα ως ένα στοιχείο το οποίο μπορεί να έχει όσες ιδιότητες χρειαζόμαστε.

Για να δημιουργήσουμε ένα στοιχείο πρέπει η εντολή μας να έχει την εξής δομή:

```
<child> … </child>
```

Όπως βλέπουμε ο τύπος του στοιχείου πρέπει να γράφεται ανάμεσα στα δυο σήματα της ανισότητας, μια φορά στην αρχή και μια φορά στο τέλος με την διαφορά ότι στο τέλος πρέπει να υπάρχει μια κάθετος (/) όπως στο παράδειγμα. Χρησιμοποιούμε αυτήν την δομή για να δηλώσουμε ότι ανάμεσα στις δυο αυτές λέξεις περιέχονται οι πληροφορίες για αυτό το στοιχείο. Είναι πολύ σημαντικό να μην ξεχνάμε να εισάγουμε τις λέξεις αυτές γιατί αλλιώς το αρχείο XML δεν θα θεωρείται σωστά δομημένο (βλ. Κεφ. 1.2.3 - Εγκυρότητα ενός αρχείου XML).

Για να εισάγουμε ιδιότητες σε αυτό το στοιχείο, αυτό που έχουμε να κάνουμε είναι να τις γράψουμε στην πρώτη λέξη, εκεί που «ανοίγει» το στοιχείο μας. Για να το κάνουμε αυτό πρέπει να ακολουθήσουμε την εξής δομή:

```
<child name=”Νίκος” height=”1.50”> … </child>
```

Μια πιο ολοκληρωμένη δομή του προηγούμενου παραδείγματος θα ήταν:

```
<child name=”Νίκος” height=”1.50”>
        <t-shirt color=”κόκκινο” type=”βαμβακερό”></t-shirt>
</child>
```

Το πιο σύνηθες λάθος που συμβαίνει στα αρχεία xml είναι η λάθος τοποθέτηση των κλεισιμάτων των στοιχείων. Θα πρέπει ο χρήστης να δίνει ιδιαίτερη προσοχή ώστε τα εσωτερικά στοιχεία, δηλαδή τα στοιχεία μέσα σε άλλα στοιχεία να κλείνουν πριν κλείσουν τα εξωτερικά.

Επίσης σημαντικό θα είναι ο χρήστης να μην ξεχνά στην αρχή του αρχείου XML να εισάγει την δήλωση(declaration) που ορίζει τις βασικές λεπτομέρειες του αρχείου όπως την έκδοση της XML, την κωδικοποίηση και αν πρέπει να συμφωνεί με ένα πρότυπο DTD.[9] Ένα παράδειγμα μιας δήλωσης είναι:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

Παρατηρώντας το ολοκληρωμένο παράδειγμα στο τέλος της προηγούμενης σελίδας, καταλαβαίνουμε πως θα μπορούσαμε να μην χρησιμοποιήσουμε ιδιότητες αλλά στοιχεία για να αποθηκεύσουμε κάποιες πληροφορίες του παιδιού. Μια εναλλακτική δομή θα ήταν:

```
<child>
        <name>Νίκος</name>
        <height>1.50</height>
        <t-shirt>
                <color>κόκκινο</color>
                <type>βαμβακερό</type>
        </t-shirt>
</child>
```

Δεν υπάρχει κάποιος κανόνας που να ορίζει ποια είναι σωστότερη μέθοδος αλλά προτιμάται η χρήση των στοιχείων έναντι των ιδιοτήτων λόγω της δυνατότητας να έχουν πολλαπλές τιμές και να επεξεργάζονται πιο εύκολα από τους XML αναγνώστες (parsers).[10]

1.2.3 Εγκυρότητα ενός αρχείου XML

Υπάρχουν δυο όροι οι οποίοι βοηθάνε τον χρήστη να καταλάβει πόσο σωστά δομημένο είναι ένα αρχείο XML.[11,12,13]

Ένα αρχείο XML θεωρείται «καλώς ορισμένο» (well formed) όταν ακολουθεί τους κανόνες σύνταξης της xml όπως:

- Τα αρχεία XML πρέπει να έχουν ένα αρχικό αντικείμενο το οποίο περιλαμβάνει τα υπόλοιπα
- Τα στοιχεία πρέπει να κλείνουν
- Οι ιδιότητες πρέπει να περιέχονται ανάμεσα από εισαγωγικά
- Τα στοιχεία που βρίσκονται εσωτερικά σε άλλα στοιχεία, πρέπει να κλείνονται πρώτα σε σχέση με τα εξωτερικά
- Τα ονόματα των στοιχείων είναι case-sensitive

Ένα αρχείο XML θεωρείται «έγκυρο» (valid) όταν είναι καλώς ορισμένο καθώς και όταν η δομή του συμφωνεί με ένα πρότυπο «DTD», δηλαδή με κάποιους κανόνες δομής.

Θα μπορούσε κάποιος να πει ότι ο όρος «καλώς ορισμένο» αναφέρεται στην σύνταξη του αρχείου XML ενώ ο όρος «έγκυρο» αναφέρεται και στην σύνταξη αλλά και στην δομή του.

1.2.4 Πλεονεκτήματα της XML

Η λίστα των πλεονεκτημάτων χρήσης της XML είναι μεγάλη.[14, 15, 16]. Τα σημαντικότερα είναι:

- Εύκολη στην ανάγνωση και στην κατανόηση από τους υπολογιστές αλλά και από τους ανθρώπους. Αυτό οφείλεται στο ότι βασίζεται σε κείμενο, έτσι αν ο άνθρωπος γνωρίζει την δομή, τα στοιχεία και τις ιδιότητες, τότε μπορεί πολύ εύκολα να κατανοήσει τι αναπαριστά η πληροφορία
- Είναι διεθνές πρότυπο. Το W3C έχει φροντίσει να προτυποποιήσει την γλώσσα και να εκδώσει όλες τις απαραίτητες πληροφορίες ώστε να επιταχύνει την διάδοση της XML. Σε όλες τις γνωστές γλώσσες προγραμματισμού υπάρχουν βιβλιοθήκες ώστε να γίνεται εύκολα η ανάγνωση και η επεξεργασία των αρχείων XML
- Επεκτάσιμη. Όπως το λέει και το όνομα της, η XML δεν έχει κάποιο περιορισμένο σετ στοιχείων και ιδιοτήτων. Η XML είναι ανοιχτή ώστε ο χρήστης να μπορεί να ορίσει οποιαδήποτε δομή δεδομένων χωρίς να χάνει καμία λειτουργικότητα
- Διεθνής γλώσσα. Υποστηρίζει το πρότυπο Unicode για την αποθήκευση των χαρακτήρων κάνοντας την εφικτή για χρήση της με οποιαδήποτε γλώσσα που περιέχει η Unicode
- Μεγάλη εξάπλωση σε εταιρίες πληροφορικής όπως AG, Microsoft, IBM και άλλες
- Έχει ομοιότητες με την γλώσσα HTML. Αυτό την καθιστά μια σχετικά πιο γνώριμη γλώσσα στους προγραμματιστές και βοηθάει την εξάπλωση της
- Μπορεί να ελεγχθεί αν υπακούει κάποιους κανόνες που ορίζει ο προγραμματιστής. Έχει την δυνατότητα να χρησιμοποιηθούν δηλαδή κάποια πρότυπα με την βοήθεια των Document Type Definition (DTD)
- Μπορεί να περιέχει πολλές μορφές δεδομένων από αρχεία multimedia μέχρι και μικρό-εφαρμογές της Java
- Έχει την δυνατότητα του ελέγχου του τρόπου παρουσίασης σε συνεργασία με άλλες τεχνολογίες όπως XSL και CSS

1.3 XQuery

1.3.1 Τι είναι η XQuery;

Με την ολοένα και αυξανόμενη εισχώρηση του ηλεκτρονικού υπολογιστή στην καθημερινότητα αυξάνεται και η αξία της πληροφορίας και των δεδομένων. Ένα τεράστιο ποσοστό δεδομένων αποθηκεύονται πλέον σε μορφή XML είτε με πλήρως

δομημένο, είτε με ημι-δομημένο τρόπο ή με συσχετιζόμενα αδόμητα δεδομένα. Με την αύξηση όμως του όγκου των δεδομένων δημιουργήθηκε η ανάγκη εύρεσης ενός τρόπου για εύκολη προσπέλαση των δεδομένων αυτών όσο το δυνατόν πιο αποτελεσματικά και πιο ολοκληρωμένα. Η XQuery είναι μια γλώσσα ερωτοαπαντήσεων βάσεων δεδομένων μορφής XML. Η XQuery σχεδιάστηκε από το W3C με σκοπό να ικανοποιήσει την ανάγκη αυτή. [17,18]

1.3.2 Ιστορία της XQuery

1998: Το W3C υποστηρίζει ένα workshop για τη χρήση ερωτημάτων σε βάσεις XML

1999: Με τη βοήθεια του W3C συγκροτείται η υπεύθυνη ομάδα με τριάντα εννιά μέλη

2000: Η υπεύθυνη ομάδα δημοσιεύει τις απαιτήσεις, τις χρήσεις περιπτώσεων και το μοντέλο δεδομένων

2001: Η υπεύθυνη ομάδα δημοσιεύει τις μη-τελικές απαιτήσεις της γλώσσας

2003: Τελική δημοσίευση των απαιτήσεων και των χρήσεων περιπτώσεων της πρώτης έκδοσης της γλώσσας με το όνομα XQuery

2007: Τελική δημοσίευση της πρώτης έκδοσης της XQuery

2014: Κυκλοφορία της τρίτη έκδοσης της XQuery και προετοιμασία για την 3.1 έκδοση

1.3.3 Δυνατότητες της XQuery

Οι βασικές δυνατότητες της XQuery είναι[19]:

- Δυνατότητα επιστροφής δεδομένων με χρήση κριτηρίων
- Φιλτράρισμα ανεπιθύμητων πληροφοριών
- Εύρεση δεδομένων από ένα σύνολο βάσεων δεδομένων
- Σύζευξη δεδομένων από έναν αριθμό βάσεων δεδομένων
- Ταξινόμηση, ομαδοποίηση δεδομένων
- Μετατροπή δεδομένων από μορφή XML σε άλλη μορφή
- Αριθμητικές πράξεις
- Επεξεργασία κειμένου

1.3.4 Χρήσεις της XQuery

Μερικές χρήσεις της XQuery[20]:

- Για συνδυασμό δεδομένων από παραδοσιακές μη-XML πηγές δεδομένων
- Επιστροφή αποτελεσμάτων από μια βάση δεδομένων στο διαδίκτυο με την μορφή XHTML
- Εκτέλεση ερωτημάτων σε XML βάσεις δεδομένων
- Εκτέλεση ερωτημάτων σε σχεσιακή βάση δεδομένων για χρήση σε υπηρεσία διαδικτύου

1.3.5 Απαιτήσεις σχεδιασμού της XQuery

Ο σχεδιασμός της XQuery ξεκίνησε το 1999 από το World Wide Consortium (W3C) με αρχικό όνομα, Quilt, λόγω του ότι βασιζόταν στην XQL και την XML-QL. Η ομάδα ως στόχους σχεδιασμού είχε τους εξής:

- Να μπορεί να χρησιμοποιηθεί τόσο για δομημένα όσο και για ημι-δομημένα έγγραφα
- Να μην βασίζεται σε κάποιο πρωτόκολλο, καθιστώντας την ικανή να τρέξει σε οποιοδήποτε σύστημα το ίδιο σωστά
- Να είναι μια επεξηγηματική γλώσσα και όχι διαδικαστική
- Να είναι αυστηρά καθορισμένη επιτρέποντας έτσι να εκτελεσθούν τα ερωτήματα ώστε να βρίσκονται πιθανά λάθη και να βελτιστοποιηθεί η αξιολόγηση τους
- Δυνατότητα εκτέλεσης ερωτημάτων σε ένα σύνολο από έγγραφα
- Υιοθέτηση όσο περισσότερων προτύπων του W3C όπως XPath, XML 1.0 και άλλα

1.3.6 XQuery και XPath

Η XPath είναι μια γλώσσα που σχεδιάστηκε με σκοπό να επιλέγει στοιχεία και ιδιότητες από ένα XML έγγραφο, χρησιμοποιώντας διάφορα κριτήρια. Η XPath 1.0 ήταν η πρώτη υλοποίηση της γλώσσας με σχετικά απλά ερωτήματα και περιορισμένο αριθμό συναρτήσεων. Αυτό άλλαξε, με την δεύτερη υλοποίηση της γλώσσας την XPath 2.0 προσθέτοντας μια μεγάλη συλλογή ερωτημάτων και συναρτήσεων στοχεύοντας στην διατήρηση της συμβατότητας με την προηγούμενη υλοποίηση. Όπως θα δούμε και στην συνέχεια, η XQuery και η XPath 2.0 έχουν πάρα πολλά κοινά στοιχεία που θα μπορούσαμε να πούμε ότι η XPath 2.0 είναι ένα υποσύνολο της XQuery, διότι δεν περιέχει δυνατότητες όπως η FLWOR.[21,22]

1.3.7 XQuery και XSLT

Η XSLT είναι μια γλώσσα που έχει ως στόχο την μετατροπή των XML αρχείων σε άλλη μορφή. Επίσης έχει την δυνατότητας της ανασχηματοποίηση των αρχείων XML, δηλαδή την αλλαγής της δομής του. Όπως και η XQuery, έτσι και η XSLT από την δεύτερη έκδοση της βασίζεται στην XPath με αποτέλεσμα να έχουν αρκετές ομοιότητες μεταξύ τους όπως στην χρήση συναρτήσεων και τελεστών αλλά υπάρχουν ορισμένες βασικές διαφορές μεταξύ τους όπως:

- Η XSLT μετατρέπει ολόκληρα τα αρχεία XML ενώ η XQuery είναι σχεδιασμένη ώστε να επεξεργάζεται μέρη του αρχείου
- Η σύνταξη ενός ερωτήματος της XQuery δεν έχει πολλές ομοιότητες με την σύνταξη της XML καθιστώντας την πιο εύκολη στην ανάγνωση και στη σύνταξη
- Παρόλο που και οι δυο μπορούν να λειτουργήσουν με πολλαπλά αρχεία, ο διερμηνέας της XQuery είναι βελτιστοποιημένος για αυτό.

Με λίγα λόγια, αν υπάρχει ανάγκη για μετατροπή ολόκληρου του XML αρχείου ενδείκνυται η χρήση της XSLT, σε διαφορετική περίπτωση ενδείκνυται η χρήση της XQuery. [23,24]

1.3.8 Σύγκριση XQuery και SQL

Η XQuery χρησιμοποιεί πολλά χαρακτηριστικά της SQL. Στην πραγματικότητα πολλοί σχεδιαστές της XQuery ασχολήθηκαν στον παρελθόν με τον σχεδιασμό της SQL. Παρόλο που στην θεωρία η διάκριση μεταξύ της χρήσης της XQuery και της SQL φαίνεται να είναι ότι η SQL χρησιμοποιείται για σχεσιακές βάσεις δεδομένων και η XQuery για αρχεία XML, στην πραγματικότητα δεν ισχύει κάτι τέτοιο. Ολοένα και περισσότερα προϊόντα διαχείρισης σχεσιακών βάσεων δεδομένων προσθέτουν την δυνατότητα αποθήκευσης αρχείων XML σε σχεσιακές βάσεις δεδομένων.

Δεν προβλέπεται στο μέλλον να σταματήσει μια από τις δυο να χρησιμοποιείται, αντιθέτως θα συνυπάρχουν και η κάθε μια θα χρησιμοποιείται στον αντίστοιχο τύπο βάσεων δεδομένων.[25,26]

1.3.9 XQuery και XML Schema

Η χρήση των σχημάτων στα δεδομένα μας θεωρείται πολύ σημαντική. Τα σχήματα μας επιτρέπουν να ορίσουμε μια αυστηρά καθορισμένη δομή των δεδομένων και να επιβεβαιώσουμε πως αυτά την ακολουθούν. Πιο συγκεκριμένα μπορούμε να προσδιορίσουμε ποια στοιχεία και ιδιότητες είναι αποδεκτά, αν τα στοιχεία μας είναι στην σωστή σειρά, αν υπάρχουν όλα τα απαραίτητα στοιχεία καθώς αν τα στοιχεία και οι ιδιότητες έχουν έγκυρες τιμές.

Εκτός λοιπόν από τον ορισμό της δομής των δεδομένων τα βασικότερα πλεονεκτήματα των σχημάτων είναι[27,28]:

- Ο μη-αναγκαίος προσδιορισμός των τύπων των δεδομένων
- Η εύκολη ανίχνευση των σφαλμάτων σε ότι έχει σχέση με τα δεδομένα όπως τα μονοπάτια, τις ιδιότητες και άλλα
- Βελτιστοποίηση του χρόνου εκτέλεσης ερωτημάτων
- Ειδική επεξεργασία δεδομένων με βάση τον τύπο τους

1.3.10 Namespace της XQuery

Εκτός από την χρήση σχημάτων, η XQuery μας επιτρέπει να ορίζουμε την σημασία των στοιχείων που επεξεργαζόμαστε. Ίσως να μην φαίνεται η αξία της χρήσης λεξικού σε απλές βάσεις δεδομένων με απλά ερωτήματα αλλά όταν ζητηθεί να αντληθούν δεδομένα από παραπάνω από μια βάση δεδομένων είναι σημαντικό ο χρήστης να γνωρίζει κάθε στοιχείο και ιδιότητα τι συμβολίζει.

1.3.11 Το ερώτημα – Query

Όπως λέει και η ονομασία του, είναι η ερώτηση του χρήστη προς την βάση δεδομένων μας στην οποία έχουμε την δυνατότητα να ζητήσουμε να επιστραφεί η απάντηση με συγκεκριμένη μορφή.

Τα ερωτήματα μπορούν να είναι αποθηκευμένα σε αρχεία κειμένου, στον κώδικα ενός προγράμματος είτε να τα πληκτρολογεί ο χρήστης σε πραγματικό χρόνο.

Ένα ερώτημα χωρίζεται σε δυο μέρη, τον πρόλογο και το κύριο μέρος. Στον πρόλογο γίνονται όλες οι δηλώσεις όπως ποια σχήματα θα χρησιμοποιηθούν καθώς και η δημιουργία μεταβλητών που έχουν άμεση σχέση με το ερώτημα. Το κύριο μέρος περιλαμβάνει τις εντολές-εκφράσεις του χρήστη χωρισμένες με κόμμα και φυσικά ορίζεται ο τρόπος που θα εμφανιστεί η απάντηση.

1.3.12 Επεξεργαστής ερωτημάτων – Query Processor

Ο επεξεργαστής ερωτημάτων είναι το λογισμικό το οποίο δέχεται το ερώτημα, το επεξεργάζεται και επιστρέφει πίσω το αποτέλεσμα. Ενδέχεται να αντιμετωπίσει διάφορα λάθη κατά την εκτέλεση που μπορεί να οφείλονται σε συντακτικά λάθη του ερωτήματος ή ακόμα και λογικά λάθη όπως διαίρεση με το μηδεν. Όλα τα λάθη έχουν ονόματα με 8 χαρακτήρες όπως «XPST0001»[29].

1.3.13 Το μοντέλο δεδομένων της XQuery

Το μοντέλο δεδομένων της XQuery χρησιμοποιείται για να προσδιορίσει τις τιμές από τα ερωτήματα συμπεριλαμβανομένου τα αρχεία που αντλούνται τα δεδομένα, τα αποτελέσματα καθώς και όλες τις ενδιάμεσες τιμές. Το επίσημο όνομα είναι XQuery 1.0 and XPath 2.0 Data Model (XDM). Λόγω του ότι πρέπει να υποστηρίξει τιμές που δεν είναι ολοκληρωμένα μοντέλα όπως ακολουθίες από στοιχεία και ατομικές τιμές δεν είναι το ίδιο με το Infoset, που είναι το μοντέλο του W3C για τα XML αρχεία.

Θα μπορούσε κάποιος να παρομοιάσει την κατανόηση του μοντέλου δεδομένων της XQuery με την εκμάθηση των πινάκων και των στηλών στην εκμάθηση της SQL. Είναι απαραίτητο ένας χρήστης να κατανοήσει αυτές τις έννοιες για να μπορέσει να συγγράψει ερωτήματα.

Οι βασικότερες έννοιες είναι[30,31,32]:

- Κόμβος – Node: Μια δομή της XML όπως ένα στοιχείο ή μια ιδιότητα
- Ατομική Τιμή – Atomic Value: Ένα απλό δεδομένο χωρίς κάποια άλλη πληροφορία σχετιζόμενη με αυτό
- Αντικείμενο – Item: Ένας γενικός χαρακτηρισμός που αναφέρεται είτε σε κόμβο είτε σε ατομική τιμή
- Ακολουθία: Μια λίστα με κανένα, ένα ή περισσότερα αντικείμενα

1.3.14 Τύποι δεδομένων στην XQuery – Types of XQuery Data

Όπως και οι πιο πολλές γλώσσες προγραμματισμού έτσι και η XQuery δίνει την δυνατότητα να ορίζουμε τον τύπο των δεδομένων χωρίς όμως να είναι υποχρεωτικό. Συνήθως η XQuery όταν διαβάζει δεδομένα χωρίς να έχει προσδιοριστεί ο τύπος τους, τα θεωρεί σαν αλφαριθμητικά και όχι σαν νούμερα, δηλαδή τα θεωρεί μια σειρά από χαρακτήρες χωρίς μαθηματικές ιδιότητες. Παραδείγματος χάρη αν ζητήσουμε από την XQuery να συγκρίνει τα νούμερα πενήντα (50) και εκατό-τριάντα (130), χωρίς να τα προσδιορίσουμε ότι είναι νούμερα θα μας επιστρέψει ότι το εκατό-τριάντα (130) είναι μικρότερο διότι ο χαρακτήρας ένα (1) είναι μικρότερος από το πέντε (5), κάτι που δεν είναι σωστό. Για αυτόν τον λόγο πριν ρωτήσουμε την XQuery θα πρέπει να προσδιορίσουμε τον τύπο των δεδομένων. Υπάρχουν όμως περιπτώσεις που γίνεται αυτομάτως αντιληπτό ότι τα δεδομένα μας είναι νούμερα και όχι αλφαριθμητικά όταν π.χ.. ζητάμε να πολλαπλασιάσουμε ή να διαιρέσουμε δυο τιμές. Επίσης όταν έχουμε χρησιμοποιήσει σχήματα για να ορίσουμε την δομή των δεδομένων, όπως είναι λογικό, δεν χρειάζεται να εισάγουμε κάθε φορά τον τύπο των δεδομένων γιατί είναι από πριν γνωστό.

Ένα από τα πλεονεκτήματα του καθορισμού του τύπου των δεδομένων είναι η δυνατότητα ανίχνευσης σφαλμάτων και εμφάνισης σαφούς μηνύματος σφάλματος χωρίς την εκτέλεση των ερωτημάτων. Αυτό σημαίνει ότι ο χρήστης μπορεί να καταλάβει την στιγμή της εισαγωγής των δεδομένων ότι κάτι δεν είναι σωστό όσον αφορά την φύση των δεδομένων χωρίς να επιθεωρήσει τον υπόλοιπο κώδικα. Επίσης η χρήση τύπου δεδομένων βελτιώνει την απόδοση της εκτέλεσης των ερωτημάτων.

Από την άλλην πλευρά, η χρήση τύπου δεδομένων αυξάνει την πολυπλοκότητα του λογισμικού καθώς χρειάζεται να γίνεται η μετατροπή των δεδομένων (casting) από τον έναν τύπο στον άλλον.

Η XQuery περιλαμβάνει δέκα-εννιά τύπους δεδομένων που μπορούν να χρησιμοποιηθούν και η ονομασία τους συνηθίζεται να χρησιμοποιείται με το πρόθεμα “xs” επειδή προέρχονται από το σχήμα της XML[33,34]. Ο παρακάτω πίνακας περιλαμβάνει τους βασικότερους τύπους δεδομένων:

xs:integer	Ακέραιοι αριθμοί
xs:double	Μη-ακέραιοι αριθμοί
xs:string	Αλφαριθμητικά
xs:Boolean	Τιμή αληθής-ψευδής
xs:date	Ημερομηνία
xs:anyURI	Τοποθεσία δεδομένων

Πίνακας 1: Τύποι δεδομένων της XQuery

1.3.15 Εκφράσεις

1.3.15.1 Βασικές εκφράσεις

1.3.15.1.1 Γενικές συγκρίσεις

Χρησιμοποιούνται για συγκρίσεις μεταξύ ατομικών τιμών ή κόμβων που περιέχουν ατομικές τιμές. Χρησιμοποιούν τους χαρακτήρες όπως <,>,=,>=, <=.

1.3.15.1.2 Γενικές συγκρίσεις σε ακολουθίες πολλαπλών αντικειμένων

Στις συγκρίσεις με ακολουθίες πολλαπλών αντικειμένων επιστρέφεται η τιμή ΑΛΗΘΗΣ αν τουλάχιστον μια από τις συγκρίσεις είναι αληθής. Για παράδειγμα η εξής εντολή:

```
doc("βιβλία.xml")/product/@dept = 'ACC'
```

αν υπάρχει έστω και ένα αντικείμενο στην κατηγορία product του αρχείου βιβλία.xml του οποίο η ιδιότητα «dept» ισούται με «ACC» τότε θα επιστρέψει ΑΛΗΘΗΣ.

1.3.16 Εκφράσεις συγκρίσεως τιμών

Χρησιμοποιούνται για να συγκρίνουν μόνο ατομικές τιμές. Δεν είναι δυνατόν να χρησιμοποιηθούν για να συγκρίνουν ακολουθίες αντικειμένων. Ιδιαίτερη προσοχή θα πρέπει να δώσουμε ώστε οι τιμές που θέλουμε να συγκρίνουμε είναι καταχωρημένες ως αριθμοί και όχι ως αλφαριθμητικά αλλιώς θα υπάρξει σφάλμα. Ο παρακάτω πίνακας περιέχει τους τελεστές σύγκρισης[35]:

eq	Ίσο με
ne	Δεν είναι ίσο με
lt	Μικρότερο από
le	Ίσο ή μικρότερο από
gt	Μεγαλύτερο από
ge	Ίσο ή μεγαλύτερο από

Πίνακας 2: Εκφράσεις συγκρίσεως τιμών

1.3.17 Εκφράσεις Ελέγχου

Η XQuery δίνει την δυνατότητα στον χρήστη να πραγματοποιεί ελέγχους και ανάλογα την περίπτωση να εκτελεί την ανάλογη πράξη. Αυτό γίνεται με την χρήση της if, then, else όπως στο παράδειγμα:

```
if (έκφραση1)
then (έκφραση2)
else (έκφραση3)
```

Στην πρώτη γραμμή, γίνεται ο έλεγχος. Αν η έκφραση είναι αληθής τότε εκτελείτε η έκφραση στην δεύτερη γραμμή, αν όχι τότε εκτελείται η έκφραση της τρίτης γραμμής. Επίσης έχουμε την δυνατότητα να κάνουμε παραπάνω από έναν έλεγχο εισάγοντας στην δεύτερη ή τρίτη γραμμή μια έκφραση if[36].

1.3.18 Λογικές εκφράσεις

Οι λογικές εκφράσεις χρησιμοποιούν τους τελεστές AND (και) και OR (ή). Χρησιμοποιούνται κυρίως στις εκφράσεις FLWOR που θα αναλυθούν παρακάτω και στις εκφράσεις ελέγχου. Ο τελεστής AND για να επιστρέψει την τιμή ΑΛΗΘΗΣ θα πρέπει και οι δυο εκφράσεις εκατέρωθεν του να είναι αληθής. Από την άλλη ο τελεστής OR για να επιστρέψει την τιμή ΑΛΗΘΗΣ θα πρέπει τουλάχιστον μια έκφραση να είναι αληθής. Οι λογικές εκφράσεις έχουν χαμηλότερη σειρά προτεραιότητας από τις εκφράσεις συγκρίσεως έτσι μπορούν να χρησιμοποιηθούν ταυτόχρονα χωρίς να χρειάζεται να χρησιμοποιήσουμε παρένθεση όπως φαίνεται στο παράδειγμα:

```
if ($k < 4 and $o > 0) then …
```

Επίσης, χρησιμοποιώντας τον τελεστή NOT μπορούμε να αντιστρέψουμε μια τιμή από αληθής να γίνει ψευδής.

1.3.19 Αριθμητικές εκφράσεις

Είναι οι εκφράσεις οι οποίες μας επιτρέπουν να συγκρίνουμε αριθμητικές τιμές. Η μεγαλύτερη διαφορά με τις εκφράσεις συγκρίσεως τιμών είναι ότι οι τελευταίες

συγκρίνουν τις τιμές σαν αλφαριθμητικά και όχι σαν νούμερα. Ο παρακάτω πίνακας περιλαμβάνει τους τελεστές των αριθμητικών εκφράσεων[37,38]:

+, -	Πρόσθεση και αφαίρεση
-	Μετατροπή σε αρνητικό αριθμό
*	Πολλαπλασιασμός
div	Διαίρεση
idiv	Διαίρεση με ακέραιο αποτέλεσμα
mod	Το υπόλοιπο της διαίρεσης

Πίνακας 3: Αριθμητικές εκφράσεις

Ιδιαίτερη προσοχή πρέπει να δοθεί στην σύνταξη των ερωτημάτων των αριθμητικών εκφράσεων όταν κάνουμε πράξεις με μια ακολουθία στοιχείων. Θα πρέπει να χρησιμοποιηθούν έτσι οι παρενθέσεις ώστε η XQuery να επεξεργάζεται κάθε στοιχείο ξεχωριστά και όχι όλη την ακολουθία των στοιχείων μαζί. Παρακάτω βλέπουμε τον εσφαλμένο τρόπο επεξεργασίας:

```
doc(“βιβλία.xml”)//τιμές * 2
```

Και τον σωστό τρόπο:

```
doc(“βιβλία.xml”)//(τιμές * 2)
```

1.3.20 Μονοπάτια – Path Expressions

Τα μονοπάτια ή αλλιώς τα XPath Path Expressions χρησιμοποιούνται για να διαλέξουμε οντότητες ή ιδιότητες μέσα από ένα συγκεκριμένο έγγραφο – βάση δεδομένων. Είναι εύκολες, γρήγορες και μας βοηθούν στο να επιστρέψουμε δεδομένα αλλά δεν μας επιτρέπουν να τα επεξεργαστούμε[39,40].

Στην αρχή πρέπει να ορίσουμε την πηγή των δεδομένων με την χρήση της λέξης doc ακολουθούμενης από το αρχείο. Όταν θέλουμε να κατέβουμε ένα επίπεδο πιο κάτω χρησιμοποιούμε πάντα την κάθετο /. Επίσης μπορούμε να χρησιμοποιήσουμε ένα μονοπάτι σε σχέση με μια μεταβλητή-μονοπάτι που έχουμε ορίσει πιο πριν.

π.χ.

```
doc("catalog.xml")/catalog/product
```

Στο συγκεκριμένο παράδειγμα, ζητάμε να προσπελάσουμε ένα αρχείο με όνομα “catalog.xml”, να επιλέξουμε το αμέσως επόμενο στοιχείο που βρίσκεται ένα

επίπεδο κάτω με όνομα “catalog” και στην συνέχεια να ζητήσουμε όσα στοιχεία υπάρχουν ένα επίπεδο πιο κάτω τύπου “product”.

Η XQuery δείχνει την ευελιξία της και τις δυνατότητες της δίνοντας μας κάποια βοηθητικά σύμβολα. Μπορούμε λοιπόν να ζητήσουμε να επιστραφούν οι ιδιότητες κάποιων στοιχείων εισάγοντας σύμβολο @ καθώς επίσης έχουμε την δυνατότητα να προσπελάσουμε “οποιοδήποτε“ σύμβολο εισάγοντας τον χαρακτήρα *. Μια άλλη δυνατότητα είναι να χρησιμοποιήσουμε το σύμβολο // για να προσπελάσουμε οποιοδήποτε στοιχείο έχει το όνομα που θα του ζητήσουμε. Και τέλος μπορούμε να ορίσουμε κάποια κριτήρια για να κάνουμε την προσπέλαση μας πιο συγκεκριμένη χρησιμοποιώντας την αγκύλη και το κριτήριο μας.

π.χ.

```
doc("catalog.xml")//product[@dept = “ACC”]
```

Στο συγκεκριμένο παράδειγμα η XQuery φορτώνει το αρχείο με όνομα “catalog.xml“, ψάχνει να βρει οπουδήποτε υπάρχει στοιχείο με όνομα “product“ και η ιδιότητα του “dept” να ισοδυναμεί με “ACC”.

Παρατηρούμε πως υπάρχουν ομοιότητες με την προσπέλαση αρχείων σε λειτουργικά συστήματα όπως Windows, Unix based κα.

1.3.21 Βασική Δομή XQuery εντολών – FLWOR

Η βασική δομή εντολών της XQuery είναι η FLWOR. Η λέξη FLWOR είναι συντομογραφία των λέξεων-εντολών For, Let, Where, Order by, Return. Σε αντίθεση με τα μονοπάτια που περιγράψαμε πιο πριν, μας επιτρέπουν να τροποποιήσουμε τα αποτελέσματα όπως επιθυμούμε. Ας ξεκινήσουμε να περιγράφουμε τις πέντε αυτές εντολές[41,42]:

1.3.21.1 For

Ορίζει μια προσπέλαση όλων των στοιχείων σε ένα συγκεκριμένο μονοπάτι. Μια εντολή for έχει την εξής δομή:

```
for $(μεταβλητή) in (μονοπάτι)
```

Η εντολή αυτή λέει στην XQuery να προσπελάσει το μονοπάτι, και για κάθε αντικείμενο που βρίσκει να το ορίζει στην μεταβλητή

1.3.21.2 Let

Μας επιτρέπει να ορίσουμε μια μεταβλητή της αρεσκείας μας. Παρόλο που μοιάζει με την εντολή “For” έχει μια σημαντική διαφορά. Η εντολή “For” για κάθε αντικείμενο που βρίσκει το καταχωρεί στην μεταβλητή και εκτελεί κάθε φορά τις υπόλοιπες εντολές ενώ η εντολή “Let” καταχωρεί στην μεταβλητή

όλα τα αντικείμενα και εκτελεί τις υπόλοιπες εντολές μια φορά[43]. Η δομή της εντολής “Let” είναι:

let $(μεταβλητή) := (μονοπάτι)

1.3.21.3 Where

Με αυτήν την εντολή ορίζουμε τα κριτήρια που πρέπει να τηρεί η for στην προσπέλαση των στοιχείων. Παίρνει τα ίδια κριτήρια με τα μονοπάτια που εξηγήσαμε παραπάνω.

Έχει την εξής δομή:

where $(μονοπάτι) = "ACC"

1.3.21.4 Order By

Μας επιτρέπει να ταξινομήσουμε τα αποτελέσματα που θα μας επιστραφούν σύμφωνα με ένα πεδίο, κάτι που δεν ήταν εφικτό με τα μονοπάτια

Έχει την εξής δομή:

order by $(μονοπάτι)

1.3.21.5 Return

Είναι η εντολή που επιστρέφει τα δεδομένα. Έχει την εξής δομή:

return $(μονοπάτι)

1.3.22 Προσθήκη στοιχείων και ιδιοτήτων

Σε περίπτωση που η υπάρχουσα δομή της βάσης μας δεν μας ικανοποιεί, η XQuery μας δίνει την δυνατότητα να την αλλάξουμε τροποποιώντας τα επιστρεφόμενα αποτελέσματα, κάνοντας την ανάλογη χρήση της FLWOR.

1.3.22.1 Προσθήκη στοιχείων

Υποθέτοντας ότι θέλουμε να επιστρέψουμε κάποια στοιχεία της βάσης μας και να τα εμφωλεύσουμε σε ένα λεξικό διαφορετικό της XML, όπως της XHTML. Με τη βοήθεια της XQuery αυτό είναι εφικτό. Ένα παράδειγμα είναι:

Query:

```
<ul>{
        for $product in doc("catalog.xml")/catalog/product
        where $product/@dept='ACC'
        order by $product/name
        return <name>data($product/name)</name>
}</ul>
```

Results:

```
<ul>
        <name>Deluxe Travel Bag</name>
        <name>Floppy Sun Hat</name>
</ul>
```

Στο παράδειγμα αυτό, παρατηρούμε ότι ζητάμε όλα τα στοιχεία που βρίσκονται στο μονοπάτι /catalog/product του αρχείου “catalog.xml” τα οποία έχουν την ιδιότητα “dept“ να είναι ίση με το “ACC”, να είναι ταξινομημένα αλφαβητικά με το έκαστο στοιχείο “name”. Αφού πάρουμε τα στοιχεία αυτά, ζητάμε το καθένα από αυτά να το εμφωλεύσουμε γύρω από ένα στοιχείο unordered list και το καθένα να έχει ιδιότητα “language” ίση με “en”.

1.3.22.2 Προσθήκη μεταβλητών

Μπορούμε πολύ εύκολα να κάνουμε προσθήκη ιδιοτήτων στα αποτελέσματα του προηγούμενου παραδείγματος εισάγοντας στην εντολή return την ιδιότητα μαζί με την τιμής της.

Π.χ..

Query:

```
<ul>{
        for $product in doc("catalog.xml")/catalog/product
        where $product/@dept='ACC'
        order by $product/name
        return <name language=”en”> data($product/name)</name>
}</ul>
```

Results:

```
<ul>
        <name language="en">Deluxe Travel Bag</name>
        <name language="en">Floppy Sun Hat</name>
</ul>
```

Βλέπουμε ότι στα αποτελέσματα προστέθηκε η ιδιότητα language=”en”.

1.3.23 Συναρτήσεις - Functions

Οι συναρτήσεις δίνουν την δυνατότητα στον χρήστη να διαχειρίζεται δεδομένα και ημερομηνίες, να εκτελούν μαθηματικές πράξεις και άλλα[44,45].

Μερικές από τις πιο βασικές ενσωματωμένες συναρτήσεων είναι:

1. *Μαθηματικές πράξεις*
    - abs(): Επιστρέφει την απόλυτη τιμή της μεταβλητής
    - avg(): Επιστρέφει την μέση τιμή των δεδομένων
    - max(): Επιστρέφει την μέγιστη τιμή
    - min(): Επιστρέφει την ελάχιστη τιμή
    - round(): Στρογγυλοποιεί την τιμή και την επιστρέφει
2. *Συναρτήσεις κειμένου*
    - lower-case(): Μετατροπή του κειμένου σε μικρά γράμματα
    - upper-case(): Μετατροπή του κειμένου σε μεγάλα γράμματα
    - string-length(): Μετράει τον αριθμό των χαρακτήρων
    - contains(): Ελέγχει αν ένα κείμενο περιέχει μια ακολουθία χαρακτήρων
    - compare(): Ελέγχει αν δυο κείμενα είναι ίδια
3. *Συναρτήσεις ημερομηνίας*
    - next-day(): Επιστρέφει την ημερομηνία της επόμενης μέρας από την ημέρα που δόθηκε σαν παράμετρο
    - is-leap-year(): Επιστρέφει τιμή Boolean αν το έτος που δόθηκε σαν παράμετρο είναι δίσεκτο.
    - current-date(): Επιστρέφει την τωρινή ημερομηνία
4. *Συναρτήσεις σχετικά με ακολουθίες δεδομένων*
    - count(): απαρίθμηση των αντικειμένων
    - distinct-values(): απαρίθμηση των μοναδικών αντικειμένων
    - last(): επιστροφή τελευταίου στοιχείου
    - reverse(): Αναποδογύρισμα της λίστας των αντικειμένων

Όπως είναι φυσικό, ο χρήστης δεν είναι περιορισμένος να χρησιμοποιεί μόνο τις ενσωματωμένες συναρτήσεις της XQuery. Δίνεται δυνατότητα στον χρήστη να δημιουργήσει δικές του συναρτήσεις και να τις χρησιμοποιεί όποτε επιθυμεί. Τα βασικότερα πλεονεκτήματα είναι:

- Επαναχρησιμοποίηση: Όταν ένας χρήστης χρησιμοποιεί την ίδια συνάρτηση από πολλά μέρη είναι πιο εύκολο να χρησιμοποιεί μια συνάρτηση και να την τροποποιεί μια φορά
- Επίγνωση: Είναι πιο εύκολο στον χρήστη να γνωρίζει εξ ολοκλήρου πως λειτουργεί μια συνάρτηση, την ονοματοδοσία καθώς και το σχήμα που ακολουθεί.
- Αυτόματη μετατροπή τύπου δεδομένων: Μπορεί ο χρήστης να αποφύγει περιττές μετατροπές τύπου δεδομένων με το να εισάγει την μετατροπή στην συνάρτηση

1.3.24 Σύζευξη

Ένα σημαντικό χαρακτηριστικό της XQuery είναι η δυνατότητα να πραγματοποιεί σύζευξη μεταξύ δυο ή και περισσότερων βάσεων δεδομένων με ιδιαίτερη ευκολία. Έστω ότι θέλουμε να κάνουμε σύζευξη των δεδομένων από τα αρχεία order.xml και catalog.xml, δεν χρειάζεται κάτι παραπάνω από δυο εντολές επανάληψης, είτε for είτε let, και το ανάλογο κριτήριο. Στην δικιά μας την περίπτωση το ερώτημα θα έπρεπε να ήταν κάτι αντίστοιχο με:

```
for $item in doc("order.xml")//item
let $name := doc("catalog.xml")//product[number = $item/@num]/name
return <item num="{$item/@num}"
        name="{$name}"
        quan="{$item/@quantity}"/>
```

1.3.25 Πράξεις με τα δεδομένα και ομαδοποίηση αποτελεσμάτων

Επίσης, η XQuery μας επιτρέπει εκτός από το να ομαδοποιούμε τα αποτελέσματα να πραγματοποιούμε πράξεις μεταξύ τους. Ένα παράδειγμα χρήσης είναι να ζητήσουμε να μάθουμε πόσα αντικείμενα έχει το κάθε παράρτημα μιας επιχείρησης. Φυσικά, σε τέτοιου είδους ερωτήματα κάνουμε χρήση των συναρτήσεων. Στην συγκεκριμένη περίπτωση χρησιμοποιούμε την συνάρτηση sum για να προσθέσουμε τις τιμές.

```
for $d in distinct-values(doc("order.xml")//item/@dept)
let $items := doc("order.xml")//item[@dept = $d]
order by $d
return <department name="{$d}" totQuantity="{sum($items/@quantity)}"/>
```

1.3.26 Χρήση και επεξεργασία των αλφαριθμητικών – Strings

1.3.26.1 Δημιουργία ενός αλφαριθμητικού

Υπάρχουν τρεις τρόποι δημιουργίας αλφαριθμητικών στην XQuery. Ο πρώτος είναι με την χρήση μονών ή διπλών εισαγωγικών, ο δεύτερος είναι με την χρήση του δημιουργού (constructor) και ο τρίτος είναι με την χρήση μιας ενσωματωμένης συνάρτησης της string.

1.3.26.2 Έλεγχος αλφαριθμητικών

Η XQuery έχει τουλάχιστον πέντε ενσωματωμένες συναρτήσεις οι οποίες επιτρέπουν τον έλεγχο του περιεχομένου ενός αλφαριθμητικού[46]:

- Compare(): Πραγματοποιείται έλεγχος μεταξύ δυο αλφαριθμητικών με την χρήση των συμβόλων ισότητας όπως =,<,>,>=,<=, !=. Ο έλεγχος ξεκινάει από τον πρώτο χαρακτήρα της κάθε λέξης και συνεχίζεται μέχρι τον τελευταίο
- Starts-with(): Ελέγχει αν το πρώτο αλφαριθμητικό ξεκινάει ακριβώς με τους ίδιους χαρακτήρες όπως το δεύτερο αλφαριθμητικό
- Ends-with(): Γίνεται ο έλεγχος αν το πρώτο αλφαριθμητικό τελειώνει ακριβώς όπως το δεύτερο αλφαριθμητικό
- Contains(): Εξετάζει αν το δεύτερο αλφαριθμητικό περιέχεται στο πρώτο ανεξάρτητα από την θέση του
- Matches(): Εξετάζει αν το πρώτο αλφαριθμητικό ταιριάζει στην έκφραση – pattern

1.3.26.2.1 Μέγεθος αλφαριθμητικού

Η ενσωματωμένη συνάρτηση string-length() δέχεται ως παράμετρο ένα αλφαριθμητικό και επιστρέφει έναν ακέραιο αριθμό ίσο με τον αριθμό των χαρακτήρων του.

1.3.26.2.2 Ένωση αλφαριθμητικών

Παρόλο που αρκετές γλώσσες προγραμματισμού επιτρέπουν την ένωση αλφαριθμητικών με την χρήση των χαρακτήρων όπως +, & ή || η XQuery δεν είναι μια από αυτές. Για αυτόν τον σκοπό υπάρχουν δυο συναρτήσεις:

- Concat(): Δέχεται όσα αλφαριθμητικά θέλει ο χρήστης και επιστρέφει ένα αλφαριθμητικό που περιέχει αυτά που εισάχθηκαν
- String-join(): Δέχεται ως ορίσματα, μια ακολουθία από αλφαριθμητικά και έναν χαρακτήρα-διαχωριστή. Επιστρέφει την ακολουθία από τα αλφαριθμητικά αφού παραβάλει ανάμεσα τους τον χαρακτήρα-διαχωριστή

1.3.26.3

1.3.26.3.1 Διαχωρισμός αλφαριθμητικών

Για να χωρίσει ο χρήστης ένα αλφαριθμητικό χρειάζεται να χρησιμοποιήσει την συνάρτηση tokenize(). Η tokenize() δέχεται το αλφαριθμητικό το οποίο είναι προς διαχωρισμό και ένα pattern για να γίνει ο διαχωρισμός μεταξύ των χαρακτήρων.

1.3.26.3.2 Αντικατάσταση γραμμάτων αλφαριθμητικού

Η συνάρτηση replace() έχει ως είσοδο δυο αλφαριθμητικά και ένα μοτίβο – pattern. Ο χρήστης εισάγει πρώτα το αλφαριθμητικό το οποίο θέλει να επεξεργαστεί, στην συνέχεια εισάγει το μοτίβο και μετά το αλφαριθμητικό που θα αντικαθιστά χαρακτήρες από το αρχικό αλφαριθμητικό.

1.3.26.3.3 Μετατροπές αλφαριθμητικών

Η XQuery δίνει την δυνατότητα στον χρήστη την εύκολη μετατροπή από κεφαλαία σε μικρά γράμματα και το ανάποδο. Αυτό γίνεται με την χρήση των συναρτήσεων:

- Upper-case()
- Lower-case()

1.3.26.3.4Απαλοιφές κενών

Για να μπορέσει ο προγραμματιστής να απαλείψει όσα παραπάνω κενά έχουν προκύψει από λάθος του χρήστη χρησιμοποιεί την συνάρτηση normalize-space(). Αυτή η συνάρτηση δέχεται το αλφαριθμητικό και το επιστρέφει αφού αφαιρέσει τα κενά που υπάρχουν πάνω από δυο φορές διαδοχικά.

1.4 Συστήματα Διαχείρισης Βάσεων Δεδομένων - DBMS

1.4.1 Βασικές πληροφορίες

Τα συστήματα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) είναι λογισμικό το οποίο επιτρέπει στον χρήστη να αλληλοεπιδρά με τις βάσεις δεδομένων του με σχετική ευκολία. Πιο συγκεκριμένα, επιτρέπει στον χρήστη να δημιουργεί, να επεξεργάζεται, να διαγράφει, να εκτελεί ερωτήματα και να διαχειρίζεται τις βάσεις δεδομένων του. Τα ΣΔΒΔ έδωσαν την λύση στο πρόβλημα της αποθήκευσης και αξιοποίησης των δεδομένων καθώς η αποθήκευση γίνεται σε πολλά αρχεία και πρέπει κάθε πρόγραμμα να περιέχει τον αντίστοιχο κώδικα για την επεξεργασία τους. Υπάρχουν πολλά συστήματα διαχείρισης βάσεων δεδομένων όπως η PostgreSQL, Oracle Database, MySQL, IBM DB2, Microsoft SQL Server που αν και έχουν ομοιότητες στην χρήση δεν σημαίνει ότι συνεργάζονται απαραίτητα μεταξύ τους. Για αυτόν τον λόγο υπάρχουν κάποιες βιβλιοθήκες που επιτρέπουν στον προγραμματιστή να γράψει λογισμικό το οποίο μπορεί να αντλήσει δεδομένα από παραπάνω από ένα σύστημα διαχείρισης βάσεων δεδομένων. Επίσης τα ΣΔΒΔ κατηγοριοποιούνται ανάλογα με τον μοντέλο της βάσης δεδομένων, το οποίο θα αναλυθεί παρακάτω[47,48,49].

1.4.2 Πλεονεκτήματα

Λόγω της μεγάλης αξίας των δεδομένων και κυρίως στην εποχή του διαδικτύου, οι απαιτήσεις στην σχεδίαση των ΣΔΒΔ είναι ιδιαίτερη απαιτητική. Τα βασικότερα πλεονεκτήματα των ΣΔΒΔ είναι[50,51]:

- Ανεξαρτησία των δεδομένων
- Ταχεία πρόσβαση στα δεδομένα: Δεδομένου ότι ο αποθηκευτικός χώρος έχει ένα μη-αμελητέο κόστος το ΣΔΒΔ πρέπει να αξιοποιεί στο έπακρο τις δυνατότητες του υλικού της εποχής
- Ταυτόχρονη πρόσβαση από χρήστες: Θα πρέπει να είναι σε θέση πολυάριθμοι χρήστες να έχουν πρόσβαση σε δεδομένα της βάσης ταυτόχρονα αλλά να υπάρχει και η δυνατότητα επαναφοράς από τυχόν βλάβες λόγω λάθους ενός χρήστη
- Ακεραιότητα δεδομένων και ασφάλεια: Επιτρέπει την επιβολή περιορισμού των δεδομένων καθώς και τον έλεγχο κατά την είσοδο αυτών αν συμφωνούν με τους περιορισμούς, μειώνοντας έτσι την πιθανότητα λάθους. Επίσης, θα

πρέπει τα δεδομένα να είναι διαθέσιμα για ανάγνωση και επεξεργασία μόνο σε αυτούς που έχουν το δικαίωμα προσπέλασης. Επίσης, θα πρέπει να τηρούνται οι κανόνες ασφαλείας σε περιπτώσεις υποκλοπών στην επικοινωνία μεταξύ του χρήστη και της βάσης δεδομένων (κρυπτογράφηση, επιβεβαίωση ταυτότητας χρήστη και άλλα)

- Εύκολη διαχείριση δεδομένων: Η δυνατότητα των χρηστών να διαχειρίζονται την βάση δεδομένων ορίζοντας βασικά χαρακτηριστικά όπως τα δικαιώματα προσπέλασης των χρηστών, βελτιστοποίησης του περιβάλλοντος της βάσης και άλλα
- Ευκολία σύνδεσης στην βάση με εξωτερικό πρόγραμμα: Αποτελεί ένα σημαντικό χαρακτηριστικό για τον προγραμματιστή να μπορεί να δημιουργήσει εύκολα ένα δικό του πρόγραμμα προσπέλασης της βάσης δεδομένων και αν θέλει να την συνδυάσει με άλλη βάση
- Backup και επαναφορά: Μια από τις πιο σημαντικές απαιτήσεις είναι το ΣΔΒΔ να μπορεί να κρατά σωστά αντίγραφα ασφαλείας και σε περίπτωση βλάβης είτε στο λογισμικό είτε στο υλικό να επαναφέρει την βάση στην λειτουργική της κατάσταση
- Καταγραφή ενεργειών: Για λόγους ασφαλείας και όχι μόνο, είναι ιδιαίτερα χρήσιμο για τον διαχειριστή της βάσης δεδομένων να είναι σε θέση να γνωρίζει τα γεγονότα που συνέβησαν πριν από μια βλάβη στην βάση δεδομένων. Αυτό θα οδηγήσει στην ταχύτερη και αποτελεσματικότερη αναγνώριση των σφαλμάτων καθώς και στην διόρθωση τους
- Μετατροπή σε άλλο τύπο ΣΔΒΔ: Είναι ιδιαίτερα ωφέλιμο ένα ΣΔΒΔ να υποστηρίζει την μετατροπή της βάσης του σε άλλο τύπο ΣΔΒΔ ώστε να επιτρέπει την εύκολη μεταφορά δεδομένων

1.4.3 Μειονεκτήματα

Εκτός από πλεονεκτήματα, τα ΣΔΒΔ έχουν φυσικά και έναν αριθμό από μειονεκτήματα[52,53]:

- Πολυπλοκότητα των ΣΔΒΔ: Παρόλο που είναι βελτιστοποιημένα για αποτελεσματική και ταχύτητα εκτέλεση σύνθετων και πολλαπλών αιτημάτων δεν σημαίνει ότι είναι κατάλληλα για εφαρμογές πραγματικού χρόνου ή για εφαρμογές που έχουν πολύ υψηλές απαιτήσεις
- Περιορισμοί της γλώσσας αιτημάτων: Υπάρχουν περιπτώσεις που ο χρήστης πρέπει να επεξεργαστεί τα δεδομένα που επιστρέφονται από την εκτέλεση των αιτημάτων και να μην μπορεί η γλώσσα αιτημάτων να τα επεξεργαστεί
- Κόστος: Η χρήση των ΣΔΒΔ απαιτεί εκτός από το ενδεχόμενο της αγοράς, αν δεν διανέμεται δωρεάν, την απόκτηση ενός υπολογιστικού συστήματος ικανό να ανταπεξέλθει στις απαιτήσεις του χρήστη. Επίσης πρέπει να προβλεφθεί η υποβάθμιση της απόδοσης του υπολογιστικού συστήματος κατά την αύξηση του όγκου της βάσης δεδομένων. Ο συνδυασμός αυτών προσθέτει ένα σημαντικό κόστος

1.4.4 Πως λειτουργούν

Έχοντας στο νου τις απαιτήσεις και τα πλεονεκτήματα της αποθήκευσης των δεδομένων σε αρχεία σε αποθηκευτικό χώρο που διαχειρίζεται το λειτουργικό σύστημα, τα ΣΔΒΔ αποθηκεύουν τα δεδομένα σε έναν δικό τους χώρο τον οποίο τον διαχειρίζονται μόνο αυτά. Αυτός ο χώρος μπορεί να είναι είτε ένα partition – δίσκος αποθήκευσης είτε ένα αρχείο μέσα στον προσωπικό φάκελο ενός χρήστη. Σύνηθες επιλογή είναι τα ΣΔΒΔ να είναι υπεύθυνα εξολοκλήρου για έναν δίσκο. Αυτό έχει ως αποτέλεσμα, γνωρίζοντας την μορφή των δεδομένων να βελτιστοποιεί τις επιδόσεις της προσπέλασης και συνάμα να ικανοποιεί όλες τις απαιτήσεις.[54] Μερικοί τύποι αποθήκευσης των δεδομένων στην βάση είναι με την χρήση:

- Ordered/unordered flat files
- Hash tables
- B+ trees
- ISAM
- Heaps

1.4.5 Βασικές έννοιες

Τα ΣΔΒΔ όπως έχει αναφερθεί περιέχουν ένα σύνολο από εργαλεία για τον σχεδιασμό, την κατασκευή και την διαχείριση των βάσεων δεδομένων. Τα δεδομένα που περιέχονται στις βάσεις δεδομένων συνήθως ακολουθούν το σχεσιακό μοντέλο δεδομένων, τα υπόλοιπα μοντέλα θα αναλυθούν παρακάτω. Το σχεσιακό μοντέλο επιτρέπει την αποθήκευση των δεδομένων με μορφή πινάκων, όπου κάθε οντότητα αντιστοιχεί σε μια πλειάδα και κάθε γνώρισμα των οντοτήτων αντιστοιχεί σε μια στήλη.

Για τον σχεδιασμό της βάσης δεδομένων χρησιμοποιείται το μοντέλο των Οντοτήτων Συσχετίσεων, δηλαδή κάθε αντικείμενο του φυσικού κόσμου περιγράφεται με ένα σύνολο οντοτήτων, όπου το καθ’ ένα από αυτά έχουν ένα σύνολο από γνωρίσματα-ιδιότητες και συνήθως μια μοναδική ταυτότητα να τα ξεχωρίζει από τα υπόλοιπα. Αυτό το μοντέλο μας επιτρέπει την άμεση μετατροπή ενός διαγράμματος Οντοτήτων Συσχετίσεων σε σχεσιακό σχήμα.

Πιο συγκεκριμένα το μοντέλο Οντοτήτων Συσχετίσεων έχει τις εξής βασικές έννοιες:

- Οντότητα: Αναπαριστά ένα αντικείμενο του φυσικού κόσμου και περιγράφεται με ένα σύνολο από γνωρίσματα-ιδιότητες. Μια οντότητα θα μπορούσε να είναι ένα άνθρωπος στα ληξιαρχικά μητρώα.
- Σύνολο Οντοτήτων: Είναι μια συλλογή από όμοιες οντότητες όπου όλες οι οντότητες έχουν τα ίδια γνωρίσματα. Συνήθως οι οντότητες έχουν τουλάχιστον μια μοναδική ταυτότητα(κλειδί) που τις κάνει να ξεχωρίζουν από τις άλλες. Ο συνδυασμός αυτών των μοναδικών ταυτοτήτων

ονομάζεται κύριο κλειδί. Θα μπορούσαμε να αναπαραστήσουμε ως σύνολο οντοτήτων το σύνολο των ανθρώπων στα ληξιαρχικά μητρώα και ως μοναδικό κλειδί το ΑΦΜ τους.
- Συσχέτιση: Είναι η σχέση μεταξύ οντοτήτων σε διαφορετικά σύνολα. Θα μπορούσαμε να αναπαραστήσουμε ως συσχέτιση την σχέση μεταξύ ενός ανθρώπου στα ληξιαρχικά μητρώα και στα φορολογικά μητρώα.

Επίσης έχει προληφθεί η κληρονομικότητα με την χρήση της δήλωσης «ISA». Με την κληρονομικότητα γίνεται εύκολη η προσθήκη γνωρισμάτων σε μια υποκλάση.

Τα ΣΔΒΔ έχοντας ως γνώμονα την ακεραιότητα, προσφέρουν την δυνατότητα στον διαχειριστή της βάσης να ορίσει ορισμένους περιορισμούς στα δεδομένα και στα γνωρίσματα τους. Τέτοιοι περιορισμοί μπορεί να είναι κάποιο πεδίο-γνώρισμα να μην μπορεί να μείνει κενό από τον χρήστη ή ένα πεδίο να μην δέχεται ημερομηνίες παλιότερες από το 2000.

Επιπροσθέτως, τα ΣΔΒΔ επιτρέπουν να οριστούν ορισμένα εναύσματα τα οποία εκτελούνται αυτόματα μετά από ένα γεγονός το οποίο ορίζει ο χρήστης της βάσης. Τα εναύσματα χωρίζονται σε τρια μέρη, την πράξη η οποία προκαλεί την εκτέλεση του, την συνθήκη για το αν θα εκτελεστεί και στο τέλος τι θα εκτελεστεί. Για παράδειγμα αν θέλουμε να καταγράφουμε στην βάση δεδομένων μας πότε προστέθηκε μια εγγραφή, θα χρησιμοποιήσουμε τα εναύσματα. Επίσης ένα άλλο χαρακτηριστικό είναι ότι μπορούμε να ορίσουμε αν το έναυσμα θα εκτελείται πριν την αρχική πράξη η μετά.

Ένα ακόμα σημαντικό χαρακτηριστικό των ΣΔΒΔ είναι η ύπαρξη και χρήση συναρτήσεων. Η SQL παρέχει ένα σύνολο από ενσωματωμένες συναρτήσεις όπως π.χ. την εύρεση της μέσης τιμής, την απαρίθμηση των στοιχείων κ.α. Εκτός αυτών όμως ο χρήστης είναι σε θέση να δημιουργήσει και τις δικές του συναρτήσεις.

1.4.6 Γλώσσα αιτημάτων και ερωτήματα - SQL

Η γλώσσα των αιτημάτων για σχεσιακά μοντέλα δεδομένων είναι η SQL. Στόχος της είναι ο ορισμός της δομής των δεδομένων καθώς και η δημιουργία και επεξεργασία των δεδομένων. Η SQL αναπτύχθηκε από την IBM την δεκαετία του 1970 και παρόλο που αποτελεί ένα πρότυπο, υπάρχουν ορισμένες διαφορές ανάμεσα στην χρήση της στα διάφορα ΣΔΒΔ[55].

Η γλώσσα SQL παρόλα τα πλεονεκτήματα και τις δυνατότητες που προσφέρει, έχει δεχτεί ιδιαίτερη κριτική. Λόγω του μεγάλου μεγέθους και της πολυπλοκότητας του προτύπου της πολλοί δημιουργοί ΣΔΒΔ έχουν επιλέξει να μην ακολουθήσουν πλήρως το πρότυπο, με αποτέλεσμα τα ΣΔΒΔ να έχουν ασυμβατότητες μεταξύ τους και να προσπαθούν να συναγωνιστούν στην απόδοση των ΣΔΒΔ και όχι στην συμβατότητα με το πρότυπο. Αυτή η κατάσταση έχει οδηγήσει σε μια αγορά

όπου οι χρήστες είναι σχεδόν περιορισμένοι σε ένα ΣΔΒΔ κάνοντας την μετάβαση αρκετά δύσκολη[56].

Υποθέτουμε ότι στην βάση δεδομένων μας έχουμε έναν πίνακα με όνομα «OOP101» ο οποίος περιέχει της πληροφορίες για ένα εργαστήριο μιας σχολής και έχει την εξής δομή:

AM	Απουσίες	Πρόοδος	Εξεταστική	T_Βαθμός

Πίνακας 4: Δομή πίνακα OOP101

Και έναν ακόμα πίνακα με όνομα «Φοιτητές» που περιέχει τα στοιχεία των φοιτητών της σχολής και έχει την εξής δομή:

Όνομα	Επίθετο	**AM**	Έτος Εισαγωγής	E-mail	Διεύθυνση	Τηλέφωνο

Πίνακας 5: Δομή πίνακα "Φοιτητές"

Με έντονα γράμματα σημειώνεται το κύριο κλειδί του κάθε πίνακα, δηλαδή το πεδίο το οποίο πρέπει να είναι μοναδικό για όλες τις εγγραφές. Και στους δυο πίνακες έχει δηλωθεί το AM ως κλειδί διότι όπως λένε και οι κανόνες του ιδρύματος, απαγορεύεται δύο φοιτητές να έχουν ίδιο AM.

1.4.6.1 Δημιουργία Πίνακα

Έστω ότι ο καθηγητής θέλει να δημιουργήσει τον πίνακα OOP101 στην βάση δεδομένων του. Όπως είπαμε η SQL επιτρέπει τον ορισμό της δομής των δεδομένων. Για να δημιουργήσει τον παραπάνω πίνακα θα πρέπει να τρέξει το παρακάτω ερώτημα:

```
Create Table OOP101 (
AM integer,
Απουσίες integer,
Πρόοδος float,
Εξεταστική float,
T_Βαθμός float,
Primary Key(AM)
);
```

Βλέποντας το ερώτημα παρατηρούμε πως γίνεται χρήση της συνάρτησης Create Table ακολουθούμενο από το όνομα του πίνακα και μέσα σε παρένθεση εισάγουμε τα ορίσματα(ή παραμέτρους). Για κάθε πεδίο-στήλη στον πίνακα εισάγουμε το όνομα, τον τύπο των δεδομένων και στο τέλος προαιρετικά τους περιορισμούς. Στο τέλος επίσης δεν ξεχνάμε να δηλώσουμε ποιο ή ποια πεδία θα αποτελούν το κύριο κλειδί (μπορεί να δηλωθεί και δίπλα από την δήλωση της μεταβλητής, δηλαδή έτσι «AM integer primary key»).

Αν θέλαμε να προσθέσουμε μια εγγραφή στον παραπάνω πίνακα θα έπρεπε η εντολή μας να μοιάζει με:

```
insert into OOP101 values(1111,1,5.2,6,5.8);
```

1.4.6.2 Αναζήτηση εγγραφών στην βάση δεδομένων

Για να μπορέσει ο καθηγητής του εργαστηρίου να δει ποιοι μαθητές έχουν δηλωθεί το μόνο που έχει να κάνει είναι να τρέχει το παρακάτω ερώτημα

```
Select *
From OOP101
```

Πιο αναλυτικά το ερώτημα αποτελείται από δυο γραμμές. Η πρώτη γραμμή με την εντολή «select» ορίζει ποιες στήλες θέλουμε να επιστραφούν, στην δικιά μας περίπτωση, θέλουμε όλες τις στήλες του πίνακα για αυτό έχουμε εισάγει τον αστερίσκο (*). Σε μια άλλη περίπτωση θα μπορούσαμε να αντικαταστήσουμε τον αστερίσκο με τις στήλες που επιθυμούμε χωρισμένες με κόμμα ανάμεσα τους. Στην δεύτερη γραμμή με την εντολή «from» ορίζουμε από ποιόν πίνακα θέλουμε να αντλήσουμε δεδομένα.

1.4.6.3 Αναζήτηση εγγραφών στην βάση δεδομένων με κριτήρια

Σε περίπτωση που ο καθηγητής θέλει να δει ποιοι φοιτητές έχουν γράψει πάνω από πέντε στην πρόοδο και μπορούν να γράψουν εξετάσεις θα πρέπει να τρέξει το παρακάτω ερώτημα

```
Select *
From OOP101
Where OOP101.Πρόοδος >= 5
```

Η μόνη διαφορά με το προηγούμενο ερώτημα είναι η τρίτη γραμμή. Η τρίτη γραμμή που ξεκινάει με το «where» εισάγει το κριτήριο, δηλαδή ζητάει όσους φοιτητές έχουν γράψει ίσο η παραπάνω από πέντε. Για να δηλώσουμε ποια στήλη ζητάμε γράφουμε το όνομα του πίνακα ακολουθούμενο με μια τελεία και το όνομα της στήλης (το όνομα του πίνακα μπορεί να παραληφθεί εφόσον χρησιμοποιούμε μόνο έναν).

1.4.6.4 Αναζήτηση εγγραφών στην βάση δεδομένων με κριτήρια με χρήση ψευδώνυμου

Αν ο καθηγητής ήθελε να πάρει μόνο το ΑΜ των μαθητών που μπορούν να γράψουν στις εξετάσεις θα έγραφε το εξής ερώτημα

```
Select O.AM
From OOP101 O
Where O.Πρόοδος >= 5 and
O.Απουσίες <= 2
```

Σε αυτό το ερώτημα βλέπουμε μια ευκολία της SQL. Η SQL δίνει την δυνατότητα στον χρήστη αντί να εισάγει κάθε φορά ολόκληρο το όνομα του πίνακα, να εισάγει ένα ψευδώνυμο του πίνακα και να χρησιμοποιεί αυτό. Για να ορίσουμε ένα ψευδώνυμο, εισάγουμε στην εντολή «from» αμέσως μετά το όνομα του πίνακα, το ψευδώνυμο. Επίσης, για να χρησιμοποιήσουμε δυο περιορισμούς, εισάγουμε την λέξη «and» στην εντολή «where».

1.4.6.5 Βασικές πράξεις με δεδομένα και ορισμός ονόματος επιστρεφόμενης στήλης

Αν ο καθηγητής θέλει να υπολογίσει τον μέσο όρο των βαθμών της προόδου όσων έχουν πετύχει, μπορεί να το κάνει με το παρακάτω ερώτημα:

```
Select avg(O.Πρόοδος) as “Μέσος
Όρος”
From OOP101 0
Where O.Πρόοδος >= 5
```

Στην εντολή «select» μπορούμε να εκτελέσουμε ορισμένες πράξεις με τα δεδομένα, εφόσον είναι αριθμητικά. Στο συγκεκριμένο ερώτημα ζητάμε τον μέσο όρο των βαθμών προόδου. Για να εκτελέσουμε μια από τις ενσωματωμένες πράξεις πρέπει να εισάγουμε το όνομα της πράξης και μέσα σε μια παρένθεση την στήλη που θα αντληθούν τα δεδομένα. Επίσης παρατηρούμε την χρήση του «as» στην εντολή select. Με αυτό ορίζουμε πως θα ονομάζεται η στήλη που θα επιστρέφεται από το ΣΔΒΔ.

Ο παρακάτω πίνακας περιέχει τις βασικές πράξεις-συναρτήσεις που είναι ενσωματωμένες στην SQL[57,58]:

avg	Μέσος όρος
count	Πλήθος
sum	Άθροισμα
max	Μέγιστος όρος
min	Ελάχιστος όρος

Πίνακας 6: Βασικές ενσωματωμένες πράξεις-συναρτήσεις της SQL

1.4.6.6

1.4.6.7 Άντληση δεδομένων από περισσότερους από έναν πίνακα

Ας δούμε την περίπτωση που ο καθηγητής θέλει να πάρει τα ονόματα όλων όσων πέτυχαν το μάθημα. Όπως γίνεται αντιληπτό πρέπει να αντλήσει δεδομένα και από τον πίνακα με όνομα «OOP101» αλλά και από το πίνακα «Φοιτητές». Το ερώτημα θα πρέπει να διαμορφωθεί κάπως έτσι:

```
Select Φ.Όνομα, Φ.Επίθετο
From OOP101.O, Φοιτητές Φ
Where O.AM = Φ.AM and O.T_Βαθμός >= 5
```

Το ιδιαίτερο σ αυτό το ερώτημα είναι η ένωση δύο πινάκων καθώς και η χρήση κριτηρίων. Το πρώτο σκέλος της «where» βλέπουμε ποια στήλη του ενός πίνακα συσχετίζεται με την στήλη του δεύτερου πίνακα, στην προκειμένη περίπτωση το ΑΜ είναι το κοινό στοιχείο. Επίσης ζητάμε από το ΣΔΒΔ να επιστρέψει μόνο φοιτητές που έχουν περάσει το μάθημα, όπως έχουμε δει και στα προηγούμενα ερωτήματα. Για να μπορέσουμε να συνδυάσουμε αυτούς τους δυο περιορισμούς, το καταφέρνουμε με την χρήση του συνδέσμου «and». Πρέπει επίσης να τονίσουμε ότι δεν έχει σημασία με ποια σειρά βάζεις τους περιορισμούς.

1.4.6.8 Ταξινόμηση επιστρεφόμενων αποτελεσμάτων

Μπορούμε επίσης να ταξινομήσουμε τα αποτελέσματα με βάση τις τιμές τους. Αυτό γίνεται με την εντολή «Order By» και μπορούμε να ορίσουμε αν θα αυξάνονται «ASC» ή θα μειώνονται «DESC». Ένα παράδειγμα είναι ο καθηγητής να ζητήσει τα ΑΜ των μαθητών στο εργαστήριο ταξινομημένα. Για να το κάνει αυτό πρέπει να τρέξει το παρακάτω ερώτημα:

```
Select O.AM
From OOP101 O
Sort By O.AM
```

1.4.6.9 Εμφωλευμένα ερωτήματα

Έχουμε επίσης την δυνατότητα να εκτελέσουμε δυο ερωτήματα ταυτόχρονα για να πάρουμε τα δεδομένα μας. Μια πιθανή περίπτωση θα ήταν να ζητήσει ο

καθηγητής να μάθει το όνομα του μαθητή που έγραψε τον καλύτερο βαθμό. Ένα πιθανό ερώτημα είναι:

```
Select Φ.Όνομα, Φ.Επίθετο
From OOP101 O, Φοιτητές Φ
Where O.AM = Φ.AM and O. T_Βαθμός = ( Select Max(T_Βαθμός)
                                      From OOP101)
```

Στο εμφωλευμένο ερώτημα ζητάμε από την SQL να μας επιστρέψει το ΑΜ του μαθητή που έχει τον μεγαλύτερο βαθμό. Αν παραπάνω από ένας μαθητής έχει γράψει τον μέγιστο βαθμό τότε θα επιστραφούν όλοι. Στην συνέχεια ζητάμε να μας επιστραφεί το όνομα και το επίθετο κάθε φοιτητή που το ΑΜ περιλαμβάνεται στην απάντηση του εμφωλευμένου ερωτήματος.

1.4.6.10 Group By

Μια άλλη ενδιαφέρουσα δυνατότητα είναι η ομαδοποίηση των δεδομένων με βάση τις τιμές. Ένα πιθανό παράδειγμα θα ήταν ο καθηγητής να θέλει να δει ποιος είναι ο μέσος όρος των απουσιών ανάλογα τον βαθμό. Για να το κάνει αυτό θα πρέπει να τρέξει το εξής ερώτημα:

```
Select O.T_Βαθμός, avg(O.Απουσίες)
From OOP101 O
Group By O.T_Βαθμός, O.Απουσίες
```

Σε αυτό το ερώτημα ζητάμε από την SQL να μας επιστρέψει τόσες εγγραφές όσες είναι οι διαφορετικές τιμές στην στήλη του «T_Βαθμός» στο πίνακα «OOP101». Στην συνέχεια για κάθε μια εγγραφή να υπολογίσει τον μέσο όρο.

1.4.6.11 Having

Παρόλο που μοιάζει με την εντολή «Where», η εντολή «Having» χρησιμοποιείται με άλλα δεδομένα. Η πρώτη εντολή χρησιμοποιεί δεδομένα κατευθείαν από τον πίνακα ενώ η δεύτερη εντολή χρησιμοποιεί δεδομένα αφού έχουν υποστεί κάποιον υπολογισμό όπως avg, max, sum και άλλα. Ένα παράδειγμα είναι να θέλουμε στο προηγούμενο ερώτημα να εμφανίσουμε την κατηγορία με τους τελικούς βαθμούς που ο μέσος όρος ξεπερνάει το πέντε.

```
Select O.T_Βαθμός, avg(O.Απουσίες)
From OOP101 O
Group By O.T_Βαθμός, O.Απουσίες
Having avg(O.T_Βαθμός) >= 5
```

1.4.7 Αντικειμενοσχεσιακό Μοντέλο

Εκτός από το σχεσιακό μοντέλο αναπαράστασης δεδομένων το οποίο περιεγράφηκε παραπάνω, υπάρχουν και άλλα μοντέλα όπως το αντικειμενοσχεσιακό καθώς και το αντικειμενοστρεφές (το οποίο βρίσκεται έξω από τα πλαίσια της πτυχιακής αυτής και δεν θα περιγραφθεί). Το αντικειμενοσχεσιακό μοντέλο είναι μια προσπάθεια επέκτασης του σχεσιακού μοντέλο με λειτουργία που απαιτείται για την υποστήριξη μιας ευρύτερης τάξης εφαρμογών. Ουσιαστικά αποτελεί το ενδιάμεσο στάδιο μεταξύ του σχεσιακού και αντικειμενοσχεσιακού μοντέλου. Παρόλο που η SQL αναπτυσσόταν από την δεκαετία του 1970, το πρώτο πρότυπο οριστικοποιήθηκε το 1987 με όνομα «SQL-86» και η υποστήριξη για τα δυο τελευταία μοντέλα δεδομένων έγινε το 1999 με το πρότυπο «SQL3» ή «SQL:1999». Μερικά από τα νέα χαρακτηριστικά που έφερε η SQL3 είναι:

- Δυνατότητα να προστεθούν νέοι τύποι δεδομένων ορισμένοι από τον χρήστη
- Μηχανισμός προσδιορισμού ταυτότητας αντικειμένου με χρήση του τύπου αναφοράς
- Υποστήριξη ενθυλάκωσης των τελεστών μέσω του μηχανισμού τύπων που ορίζονται από το χρήστη
- Μηχανισμοί κληρονομικότητας για την επέκταση κλάσεων
- Υποστήριξη μεθόδων στις κλάσεις

1.4.8 Χαρακτηριστικά και παραδείγματα χρήσης

Ο χρήστης, με την νέα έκδοση την SQL3 μπορεί να ορίσει δικούς του τύπους δεδομένων. Υπάρχουν δυο είδη τύπων.

Ο πρώτος τύπος είναι οι απλοί τύποι δεδομένων. Οι απλοί τύποι δεδομένων βασίζονται στους ενσωματωμένους τύπους του συστήματος όπως integer, varchar και δίνουν την δυνατότητα να διαχειριστούμε καλύτερα τη σημασιολογία των δεδομένων. Ένα απλό παράδειγμα δημιουργίας απλών τύπων είναι το εξής:

```
create type address as (street text, street_no int, postcode int, city text);
```

Αυτή η εντολή δημιουργεί έναν τύπο με όνομα «address» που βασίζεται σε έναν συνδυασμό τεσσάρων μεταβλητών οι οποίες βασίζονται σε ενσωματωμένους τύπους της SQL. Βλέπουμε πως η μεταβλητή «street» και η «city» είναι τύπου text, ενώ οι άλλες δυο, η «street_no» και η «postcode» είναι τύπου ακέραιου αριθμού. Οι μεταβλητές τύπου text δεν έχουν κάποιον περιορισμό στο μέγεθος και παρόλο που δεν ορίζεται στο πρότυπο της SQL τον υποστηρίζει η PostgreSQL και ένας αριθμός ΣΔΒΔ[59]. Οι μεταβλητές τύπου int έχουν μέγεθος μέχρι τέσσερα byte και συνεπώς έχουν σαν όριο τιμής το [-2147483648,2147483647].

Ο δεύτερος τύπος είναι οι σύνθετοι τύποι δεδομένων. Οι σύνθετοι τύποι δεδομένων περιέχουν και χαρακτηριστικά (attributes) που μπορεί να είναι απλού τύπου, σύνθετου τύπου, τύπου συλλογής ή ακόμα και αναφορά σε άλλο τύπο αντικειμένου, καθώς και μεθόδους. Ένα ενδεικτικό ερώτημα δημιουργίας ενός σύνθετου τύπου είναι:

```
create type person as (
name text,
surname text,
am integer,
reg_year integer,
email text,
phone integer,
address1 address);
```

Σε αυτήν την εντολή, βλέπουμε να έχει δημιουργηθεί ένας νέος τύπος με όνομα «person» που περιέχει έξι απλούς τύπους δεδομένων (name,surname,am,reg_year,email,phone) και έναν σύνθετο τύπο δεδομένων τον τύπο «address».

Σε αυτό το παράδειγμα, η μόνη διαφορά σε σχέση με το σχεσιακό μοντέλο είναι η τελευταία εντολή. Σε αυτήν την γραμμή παρατηρούμε πως η μεταβλητή με όνομα «address1» δεν είναι τύπου ενός από τους ενσωματωμένους της SQL αλλά από έναν που δημιουργήσαμε εμείς.

Αν θα θέλαμε να δημιουργήσουμε έναν πίνακα που να περιέχει ένα σύνολο από εγγραφές τύπου «person» η εντολή μας θα έμοιαζε:

```
create type student of person(
primary key(am)
);
```

Εφόσον δεν θέλουμε να προσθέσουμε κάποια άλλη ιδιότητα, και έχουμε περιορισμό στα δεδομένα μας, δηλαδή ότι δεν μπορούν να έχουν δυο μαθητές το ίδιο ΑΜ προσθέτουμε την προ-τελευταία εντολή στο ερώτημα μας. Σε αυτήν την εντολή «primary key(am)» ορίστηκε ότι το πεδίο του πίνακα μας με όνομα «ΑΜ» θα είναι το κλειδί, δηλαδή δεν θα επιτρέπει ίδιες τιμές σε παραπάνω από μια καταχώρηση.

Αν θα θέλαμε να προσθέσουμε δεδομένα σε αυτόν τον πίνακα θα έπρεπε να εκτελέσουμε ένα ερώτημα όπως το παρακάτω:

```
insert into student values(
'Nikos','Papadopoulos',1111,2005,'nikospap@example.com',555444,
ROW('Komninon',50,55130,'Thessaloniki'));
```

Η μοναδική διαφορά σε σχέση με το αντίστοιχο παράδειγμα στο σχεσιακό περιβάλλον είναι ότι για να δημιουργήσουμε ένα αντικείμενο τύπου address θα πρέπει να εισάγουμε τα δεδομένα του γράφοντας την λέξη «ROW» και εισάγοντας τα ορίσματα-τιμές μας, με την σειρά που έχει οριστεί.

Και οι δυο περιπτώσεις δημιουργίας δικών μας τύπων, μας θυμίζουν το μοτίβο «σύνθεσης» του αντικειμενοστρεφούς προγραμματισμού, διότι έτσι και εδώ συνδυάζονται στοιχεία για αναπαράσταση μιας οντότητας.

Εκτός από την σύνθεση, το αντικειμενοσχεσιακό μοντέλο δεδομένων μας επιτρέπει να χρησιμοποιήσουμε το μοτίβο της κληρονομικότητας στα δεδομένα μας, όπως και στον αντικειμενοστρεφή προγραμματισμό. Με την κληρονομικότητα έχουμε την δυνατότητα να επεκτείνουμε έναν υπάρχον τύπο, προσθέτοντας νέες ιδιότητες και αυτός ο νέος τύπος να κληρονομεί την ταυτότητα του. Η κλάση-τύπος η οποία κληροδοτεί, δηλαδή δίνει τις ιδιότητες της, ονομάζεται «υπερ-κλάση» ή αλλιώς «υπερ-τύπος», ενώ αντίθετα η κλάση-τύπος που κληρονομεί ονομάζεται «υπο-κλάση» ή «υπο-τύπος». Ιδιαίτερη προσοχή πρέπει να δοθεί στην χρήση των τύπων, μια υπο-κλάση μπορεί να αντικαταστήσει μια υπερ-κλάση, δεν ισχύει όμως το αντίστροφο. Στο αντικειμενοσχεσιακό μοντέλο δεδομένων, η κληρονομικότητα δεν περιορίζεται μόνο στον τύπο-κλάση αλλά και στους πίνακες.

Ένα παράδειγμα χρήσης, θα ήταν αν θέλαμε τον πίνακα «student» να τον επεκτείνουμε ώστε να περιέχει το ύψος και το βάρος του κάθε μαθητή. Το μόνο που θα έπρεπε να κάνουμε είναι να δημιουργήσουμε έναν νέο πίνακα, να προσθέσουμε τις δυο ιδιότητες που θέλουμε και να ορίσουμε ότι κληρονομεί τα στοιχεία του πίνακα «student», όπως φαίνεται και στο παράδειγμα:

```
create table measurements(
height int,
weight int
) inherits (student);
```

Και αν θα θέλαμε να προσθέσουμε δεδομένα στον νέο μας πίνακα θα μπορούσαμε να το κάνουμε κανονικά, προσθέτοντας τις τιμές για τις δυο νέες στήλες στο τέλος. Όπως φαίνεται στο παρακάτω παράδειγμα:

```
insert into measurements values(
'Nikos','Papadopoulos',1111,2005,'nikospap@example.com',555444,ROW('Komninon'
,50,55130,'Thessaloniki'),180,75);
```

1.5 Επίλογος

Μετά την ανάγνωση αυτού του κεφαλαίου, αναγνωρίζουμε πως ο αναγνώστης έχει κατανοήσει τα βασικά των τεχνολογιών χωρίς όμως να είναι σε θέση να ανταποκριθεί πλήρως και να προσπελάσει πληροφορίες που βρίσκονται αποθηκευμένες σε βάση δεδομένων. Για να εξαλείψουμε την τριβή του χρήστη στο πρώτο κεφάλαιο, στην συνέχεια θα παρουσιαστεί μια λύση για εύκολη προσπέλαση δεδομένων από τους αναγνώστες χωρίς κάποια ιδιαίτερη δυσκολία.

2 Ανάλυση της εφαρμογής και των βοηθητικών προγραμμάτων

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται περιγραφή της εφαρμογής. Περιγράφεται όχι μόνο το γραφικό περιβάλλον της εφαρμογής αλλά και όλα τα προγράμματα που χρειάστηκαν για την υλοποίηση της. Επίσης, ιδιαίτερη έμφαση δίνεται στις μεταφορές-metaphors, στην σημασία τους καθώς και στην χρήση τους. Οι μεταφορές που χρησιμοποιήθηκαν βασίστηκαν στις μεταφορές της πτυχιακής εργασίας του Α.Πλιάκα και Κ.Τσέκου με τίτλο «Υλοποίηση εργαλείου γραφικής απεικόνισης της XQuery»

2.2 Σκοπός της εφαρμογής

Δεδομένου ότι όχι μόνο η ανάγνωση αλλά και η επεξεργασία δεδομένων από μια βάση είναι δύσκολη για έναν απλό χρήστη, καθώς απαιτεί ένα βαθμό εξοικείωσης και κάποιες βασικές γνώσεις, η εφαρμογή αυτή αποτελεί μια λύση. Η εφαρμογή αυτή σχεδιάστηκε με στόχο να εξαλείψει αυτό το χάσμα και να απλοποιήσει την διαδικασία χρησιμοποιώντας μια σειρά από metaphors, δηλαδή από γνώριμες εικόνες-σήματα και κατανοητά μηνύματα που θα αντικαταστήσουν εντολές και μηνύματα λάθους. Επίσης, ιδιαίτερη προσοχή δόθηκε στην χρήση της εφαρμογής. Όπως μας επιτάσσει η τεχνολογία, το πλήθος των συσκευών και η αρχιτεκτονική τους, ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή από οποιαδήποτε συσκευή εφόσον είναι συνδεδεμένη στο διαδίκτυο ανεξάρτητα από το αν είναι υπολογιστής με Windows ή Linux, ή αν είναι tablet ή κινητό.

2.3 Παρουσίαση προσέγγισης και εικονιδίων

Η προσέγγιση που χρησιμοποιήθηκε κατά την επιλογή των metaphors είχε ως στόχο την περεταίρω αύξηση της οικειότητας των χρηστών με την εφαρμογή. Για να επιτευχθεί αυτό έπρεπε να επιλεχθούν εικονίδια από την καθημερινή ζωή που είναι γνώριμα στον μέσο χρήστη και δεν θα δυσκολευτεί να τα χρησιμοποιήσει στην πράξη. Ο στόχος για την επιλογή των εικονιδίων δεν ήταν η πιστή αναπαράσταση των εντολών αλλά η εύκολη και γρήγορη κατανόηση τους από τους χρήστες. Για αυτόν τον λόγο τα εικονίδια που χρησιμοποιήθηκαν για να αντικαταστήσουν τις εντολές πάρθηκαν από τα σήματα της οδικής κυκλοφορίας που συναντά κάποιος αρκετές φορές την ημέρα και είναι ευρέως γνωστά, όχι μόνο στην χώρα μας αλλά στις περισσότερες χώρες του κόσμου. Έτσι, οι χρήστες γνωρίζοντας ήδη τα σήματα, μπορούν να επικεντρωθούν απευθείας στην σύνταξη τους και μαθαίνοντας το γραφικό περιβάλλον της εφαρμογής. Οι εικόνες βασίστηκαν στην πτυχιακή εργασία

του Α.Πλιάκα και Κ.Τσέκου με τίτλο «Υλοποίηση εργαλείου γραφικής απεικόνισης της XQuery»

Εικόνα 1: Οδικά σήματα στις ΗΠΑ, Ιρλανδία και Πορτογαλία

Στις παρακάτω παραγράφους θα επεξηγηθούν τα metaphors που χρησιμοποιηθήκαν στην εφαρμογή, πως συντάσσονται, τι τιμές δέχονται καθώς και πως λειτουργούν.

2.3.1 For

Εικόνα 2: Metaphor For

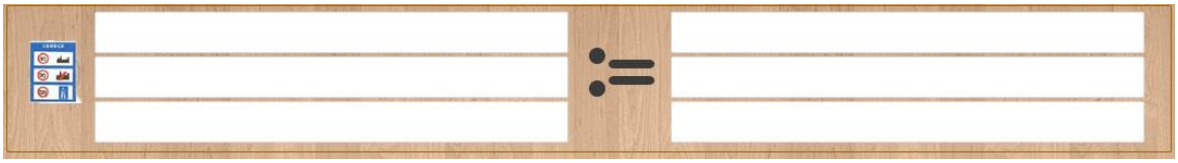
Όπως έχει αναφερθεί στο κεφάλαιο της XQuery, η εντολή for χρησιμοποιείται για να γίνει προσπέλαση των αντικειμένων στο συγκεκριμένο μονοπάτι που ορίζουμε. Παρόμοια σκέφτεται και ένας χρήστης σε έναν κυκλικό κόμβο, που δεν γνωρίζει τις επιλογές, πραγματοποιεί έναν κύκλο και διαβάζει τις ενδείξεις για κάθε μια από τις εξόδους. Στην εφαρμογή, αφού ο χρήστης πατήσει την metaphor «for» θα παρατηρήσει δυο textboxes και την λέξη in. Στο πρώτο textbox ο χρήστης θα εισάγει το όνομα της μεταβλητής που θα πάρει ως τιμή τα αντικείμενα και στο δεύτερο textbox θα εισάγει το μονοπάτι της βάσης δεδομένων που θα αντληθούν τα αντικείμενα.

Εικόνα 3: Χρήση του metaphor For στην εφαρμογή

2.3.2 Let

Εικόνα 4: Metaphor Let

Όπως αυτή η πινακίδα στους δρόμους ορίζει την μέγιστη ταχύτητα των οχημάτων ανά περιοχή έτσι και το metaphor ορίζει σε μια μεταβλητή τις τιμές της. Στην εφαρμογή, επιλέγοντας αυτό το metaphor εμφανίζονται δυο textboxes και το σύμβολο := ανάμεσα τους. Στο πρώτο textbox, ο χρήστης θα εισάγει το όνομα της μεταβλητής και στο δεύτερο θα εισάγει τις τιμές ή το μονοπάτι που θα πάρει η μεταβλητή στο πρώτο textbox.



Εικόνα 5: Χρήση της metaphor Let στην εφαρμογή

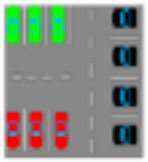
2.3.3 Where

Εικόνα 6: Metaphor Where

Ένας οδηγός όταν συναντάει έναν τροχονόμο, λαμβάνει εντολές για το πώς πρέπει να κινηθεί στην συνέχεια της πορείας του. Παρόμοια συμβαίνει και με το metaphor «where», όπου ο χρήστης ορίζει στην XQuery σύμφωνα με πιο κριτήριο να αναζητήσει και να επιστρέψει τα αποτελέσματα. Ο χρήστης στο πρώτο textbox ζητείται να εισάγει το μονοπάτι, δηλαδή το πεδίο στην βάση δεδομένων, στην συνέχεια εισάγει το κριτήριο ισότητας(<,<=,=,!=,>=,>) και στο επόμενο πεδίο εισάγει την τιμή με την οποία θα συγκριθεί. Σε περίπτωση που ο χρήστης θέλει να εισάγει και άλλο κριτήριο, το μόνο που έχει να κάνει είναι να πατήσει το κουμπί «and» ή «or» και θα δημιουργηθεί ένα νέο σύνολο με textboxes. Αυτό μπορεί να το επαναλάβει όσες φορές θέλει ο χρήστης.

Εικόνα 7: Χρήση του metaphor Where στην εφαρμογή

2.3.4 Order By



Εικόνα 8: Metaphor Order By

Αυτό το metaphor θέλει να δώσει στον χρήστη την ικανότητα να ταξινομεί τα επιστρεφόμενα αποτελέσματα σύμφωνα με ένα συγκεκριμένο πεδίο. Η προεπιλεγμένη ταξινόμηση είναι η αλφαβητική. Ο χρήστης στο μοναδικό textbox που του εμφανίζεται πρέπει να ορίσει το πεδίο ως προς το οποίο θα γίνει ταξινόμηση, σε περίπτωση που θέλει να ορίσει και ένα δεύτερο πεδίο για ταξινόμηση, θα πρέπει να το εισάγει αφού τοποθετήσει ένα κόμμα (,).

Εικόνα 9: Χρήση του metaphor Order By στην εφαρμογή

2.3.5 Return

Εικόνα 10: Metaphor Exit

Ένα από τα πιο σημαντικά metaphor είναι αυτό της εντολής return. Όπως και το σήμα του ΚΟΚ που ενημερώνει τον οδηγό ότι υπάρχει έξοδος από την κύριο αυτοκινητόδρομο έτσι και αυτό το metaphor εξάγει/επιστρέφει τα δεδομένα πίσω στον χρήστη. Δηλαδή αυτό ορίζει τι δεδομένα θα επιστραφούν και πως. Επίσης το metaphor της Return δέχεται κάποιες ενσωματωμένες συναρτήσεις της XQuery, ώστε να επεξεργάζονται οι τιμές πριν επιστραφούν στον χρήστη. Στην εφαρμογή ο χρήστης πατώντας το εικονίδιο της return, του επιστρέφει μια μπάρα με τις ενσωματωμένες συναρτήσεις που μπορεί να επιλέξει. Αυτή η μπάρα θα περιγραφθεί στην συνέχεια. Επιλέγοντας το metaphor return, εμφανίζεται ένα textbox που ο χρήστης ορίζει τα επιστρεφόμενα δεδομένα.

Εικόνα 11: Χρήση του metaphor Return

2.3.6 Μπάρα ενσωματωμένων συναρτήσεων της XQuery

Όπως προαναφέρθηκε, δίνεται στον χρήστη η δυνατότητα επεξεργασίας των δεδομένων που επιστρέφονται κάνοντας χρήση των ενσωματωμένων συναρτήσεων της XQuery. Αυτή ή μπάρα εμφανίζεται πατώντας το εικονίδιο(metaphor) της return, στο πλαίσιο που δημιουργείται αφού πατηθεί το κουμπί της return. Η μπάρα φαίνεται παρακάτω:

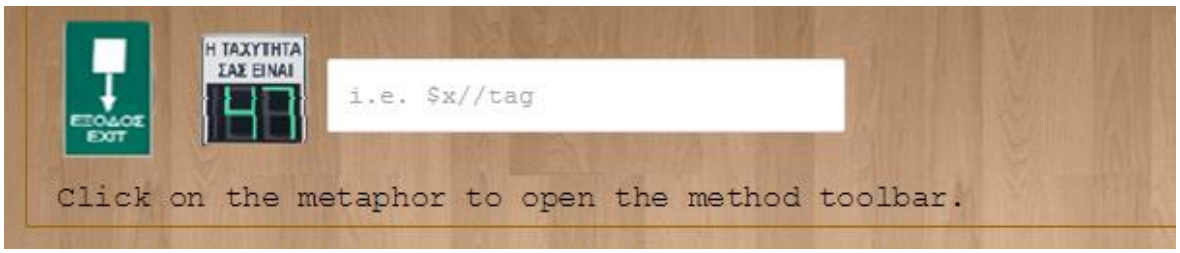
Εικόνα 12: Γραμμή ενσωματωμένων συναρτήσεων

Οι ενσωματωμένες συναρτήσεις είναι:

2.3.7 Count

Εικόνα 13: Metaphor Count

Στο προηγούμενο κεφάλαιο αναφέραμε πως η XQuery έχει ενσωματωμένο σύνολο από μεθόδους-συναρτήσεις. Η Count είναι μια από αυτές και επιλέγοντας την ο χρήστης έχει την δυνατότητα να μετρήσει το πλήθος το στοιχείων που επιστρέφονται.

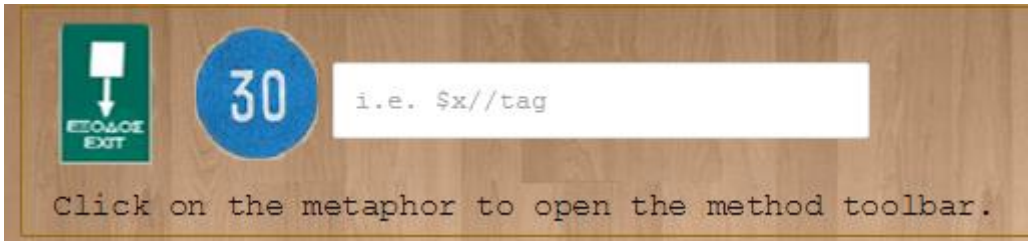


Εικόνα 14: Χρήση του metaphor Count μαζί με το metaphor Return

2.3.8 Min

Εικόνα 15: Metaphor Min

Άλλη μια ενσωματωμένη συνάρτηση της XQuery. Όπως το σήμα του κοκ που ορίζει την ελάχιστη ταχύτητα που θα κινούνται τα οχήματα, έτσι και αυτό το metaphor βοηθά τον χρήστη να βρει την ελάχιστη τιμή ανάμεσα σε ένα σύνολο από στοιχεία.



Εικόνα 16: Χρήση του metaphor Min μαζί με το metaphor Return

2.3.9 Max

Εικόνα 17. Metaphor Max

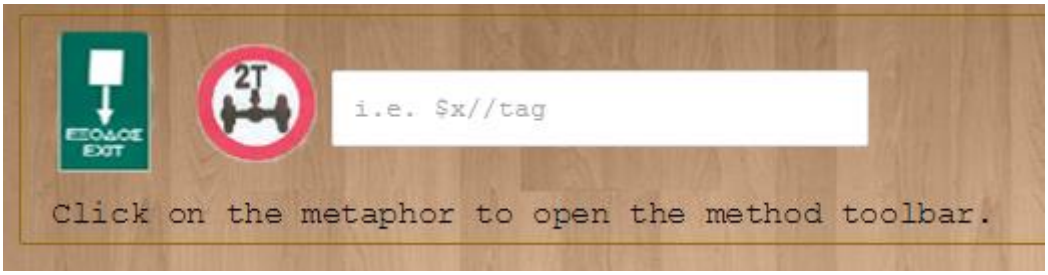
Αποτελεί το ακριβώς ανάποδο του προηγούμενου metaphor. Σε αντίθεση με το Min, ο χρήστης βρίσκει την μέγιστη τιμή ανάμεσα σε ένα σύνολο από στοιχεία.

Εικόνα 18: Χρήση του metaphor Max μαζί με το metaphor Return

2.3.10 Avg

Εικόνα 19: Metaphor Avg

Όπως φαίνεται και απο το όνομα του, αυτό το metaphor υπολογίζει το μέσο όρο των τιμών ενός πεδίου των αντικειμένων. Εκμεταλλεύεται την αντίστοιχη συνάρτηση της XQuery, την avg.



Εικόνα 20: Χρήση του metaphor Avg μαζί με το metaphor Order By

2.3.11 Sum

Εικόνα 21: Metaphor Sum

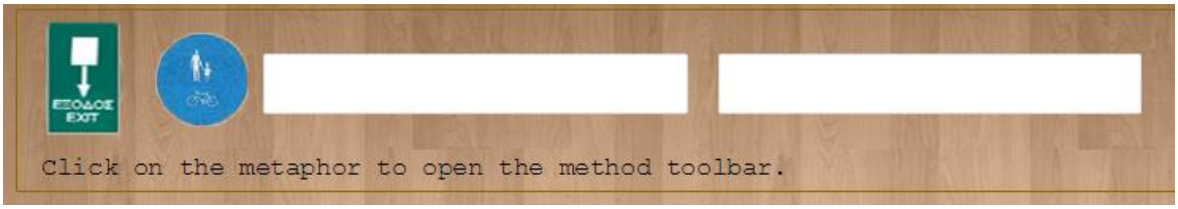
Όπως διαπιστώνουμε από το όνομα αλλά και από την εικόνα αυτού του metaphor, χρησιμοποιείται για να υπολογίσει το άθροισμα των στοιχείων του πεδίου που έχει επιλεχθεί να επιστραφεί. Αξιοποιεί την αντίστοιχη ενσωματωμένη συνάρτηση της XQuery, την sum.

Εικόνα 22: Χρήση του metaphor Sum μαζί με το metaphor Return

2.3.12 AND

Εικόνα 23: Metaphor And

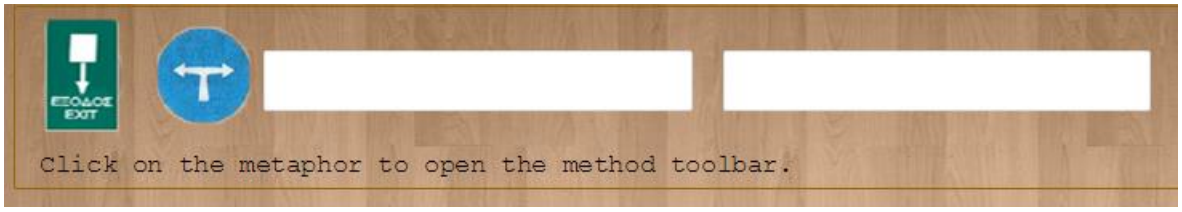
Παρομοίως με το σήμα του ΚΟΚ που επιτρέπει την προσέλευση στους ποδηλάτες αλλά και τους πεζούς, έτσι και αυτό το metaphor επιτρέπει στον χρήστη να ελέγξει αν δυο εκφράσεις είναι σωστές. Αν και μόνο αν και οι δυο είναι σωστές, δηλαδή true, τότε επιστρέφει στον χρήστη η λέξη «true», αν όχι τότε επιστρέφει την τιμή «false»



Εικόνα 24: Χρήση του metaphor And μαζί με το metaphor Return

2.3.13 OR

Εικόνα 25: Metaphor Or



Εικόνα 26: Χρήση του metaphor Or μαζί με το metaphor Return

Αυτό το metaphor επιτρέπει στον χρήστη να ελέγχει αν τουλάχιστον μια από τις δυο εκφράσεις που θα εισάγει στα δυο textboxes είναι σωστή. Αν είναι έστω μια σωστή τότε θα επιστρέψει την τιμή «true» αλλιώς θα επιστρέψει την τιμή «false».

2.3.14 If/Else

Εικόνα 27: Metaphor If/Else

Όπως το σήμα επιτρέπει στον οδηγό να συνεχίσει την πορεία του ευθεία ή να στρίψει δεξιά, έτσι παρόμοια λειτουργεί και το metaphor. Επιτρέπει στον χρήστη να συγκρίνει την τιμή ενός πεδίου, και αν είναι σωστή η σύγκριση να επιστρέψει μια συγκεκριμένη τιμή ή εναλλακτικά μια άλλη. Επιλέγοντας το ο χρήστης, εμφανίζονται τρια textboxes. Στο πρώτο εισάγει την υπόθεση, δηλαδή το κριτήριο, στο δεύτερο την τιμή αν η υπόθεση είναι αληθής και στο τρίτο textbox την επιστρεφόμενη τιμή αν η υπόθεση είναι λάθος.

Εικόνα 28: Χρήση του metaphor If/Else μαζί με το metaphor Order By

2.3.15 Remove

Εικόνα 29: Metaphor Remove

Αυτό το κουμπί σβήνει από την return όποιο metaphor ενσωματωμένης συνάρτησης έχει χρησιμοποιηθεί.

2.4 Sedna

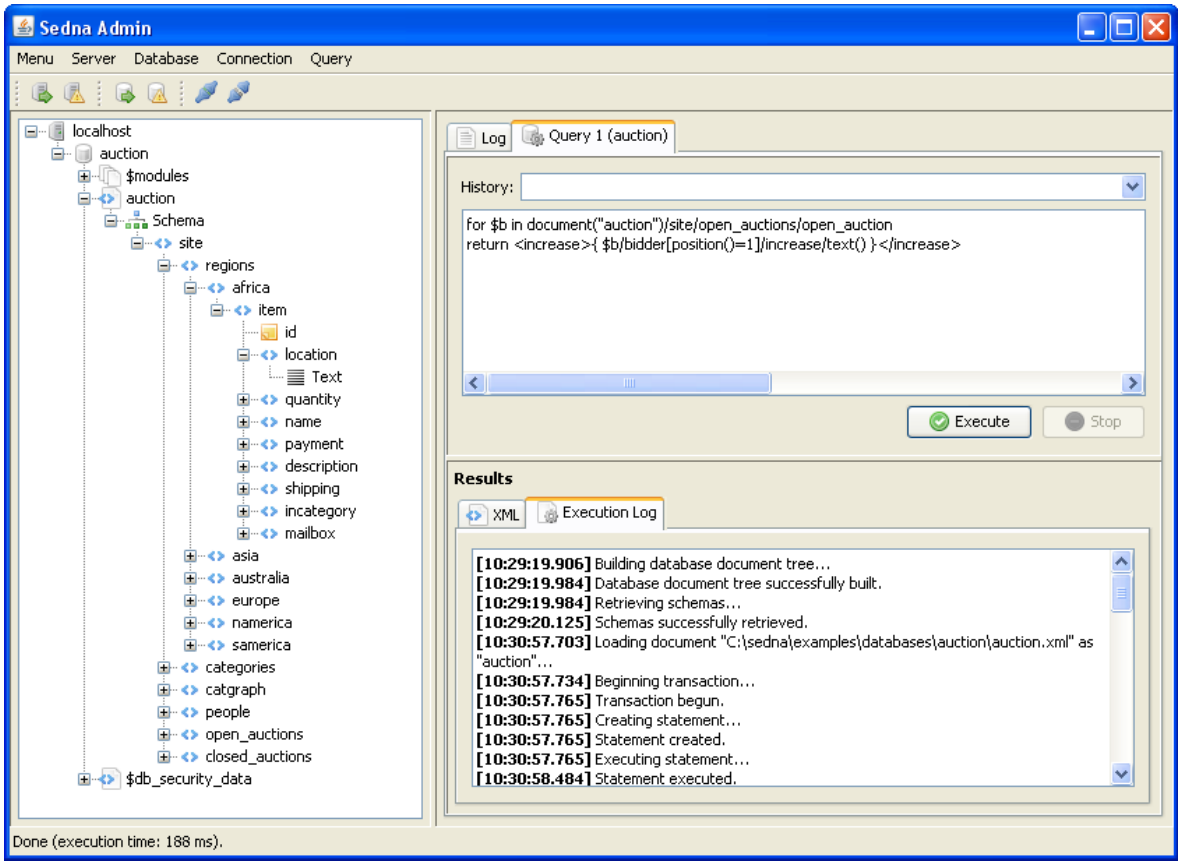
Εικόνα 30: Λογότυπο του Sedna

Η εφαρμογή sedna αποτελεί ένα σημαντικό και ουσιαστικό κομμάτι της πτυχιακής εργασίας. Θα μπορούσε να παραλληλιστεί με ένα ΣΔΒΔ αλλά για βάσεις με αρχεία XML. Το sedna είναι ελαφρύ και ανοιχτού τύπου λογισμικό το οποίο μας επιτρέπει να δημιουργήσουμε τοπικά μια βάση δεδομένων, να εκτελέσουμε ερωτήματα και να μας επιστρέψει δεδομένα. Η πρώτη έκδοση κυκλοφόρησε τον Ιούνιο του 2006 και ύστερα από πολλές προσθήκες και βελτιώσεις έφτασε στην έκδοση 3.5 τον Νοέμβριο του 2011. Οι βασικότεροι λόγοι που προτιμήθηκε η εφαρμογή αυτή ήταν η ύπαρξη του ανοιχτού κώδικα καθώς και η ευκολία στην εγκατάσταση και στην χρήση. Παρόλο που το sedna δεν έχει κάποια επίσημη γραφική διεπαφή, ο χρήστης μπορεί είτε να χρησιμοποιεί εντολές στο τερματικό είτε να χρησιμοποιήσει μια από τις ανεπίσημες διεπαφές. Στην ανάπτυξη της πτυχιακής χειριστήκαμε το sedna μέσω των εντολών του τερματικού.

Τα βασικότερα χαρακτηριστικά του sedna είναι[60]:

- Ανοικτού τύπου λογισμικό υπό την άδεια Apache License 2.0
- Εγγενής υποστήριξη XML αρχείων κάνοντας χρήση των γλωσσών C και C++
- Συμμόρφωση με τα πρότυπα του W3C
- Υποστήριξη εξωτερικών συναρτήσεων της XQuery
- Υποστήριξη κωδικοποίησης κειμένου Unicode

- Υποστήριξη χρήσης XML εναυσμάτων
- Διαθέσιμοι drivers για συνεργασία με εφαρμογές σε Java/C/PHP/Python/Ruby/Perl/C# και άλλες γλώσσες προγραμματισμού
- Ασφάλεια στις βάσεις δεδομένων με βάση τους χρήστες, τους ρόλους και τα δικαιώματα



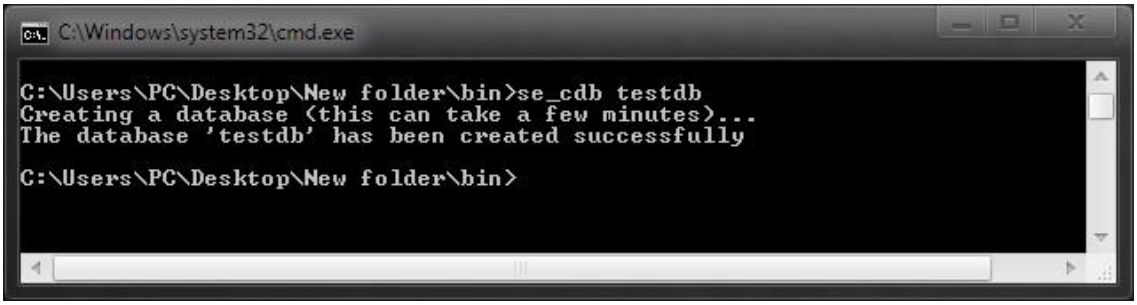
Εικόνα 31: Ένα γραφικό περιβάλλον του Sedna

Τα βήματα για την εκκίνηση του sedna, την κατασκευή της βάσης και την εκκίνηση αυτής παρουσιάζονται παρακάτω:

1. Κατέβασμα και εγκατάσταση του sedna σε έναν φάκελο της προτίμησης μας
2. Εκκίνηση του sedna με την παρακάτω εντολή αφού μεταφερθούμε στον φάκελο bin της εγκατάστασης του

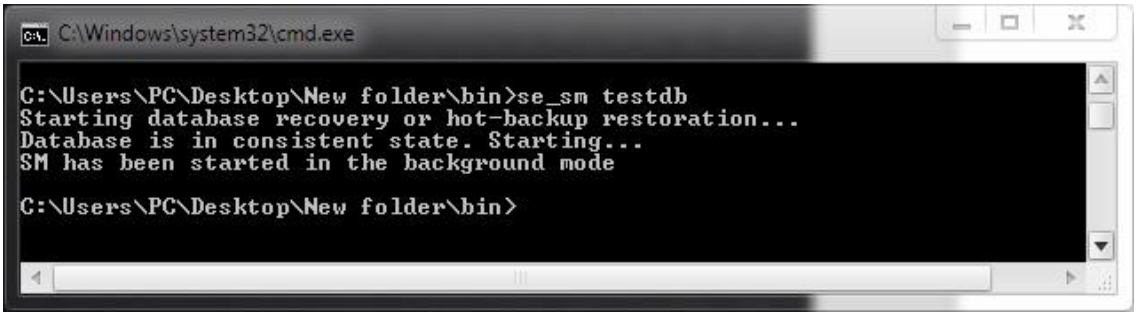
Εικόνα 32: Εκκίνηση του Sedna

3. Δημιουργία μιας τοπικής βάσης με το όνομα «testdb»



Εικόνα 33: Δημιουργία της βάσης μας

4. Εκκίνηση της βάση που μόλις δημιουργήσαμε



Εικόνα 34: Εκκίνηση της βάσης μας

5. Τερματισμός της βάσης και του Sedna

Εικόνα 35: Τερματισμός βάσης και του Sedna

Είναι αρκετά ενθαρρυντικό πως πριν το πέμπτο βήμα, οι δυο διεργασίες που τρέχουν στο παρασκήνιο, η «se_gov» και η «se_sm» συνολικά καταλαμβάνουν δεκατρία (13) MB στην μνήμη RAM.

2.5 Ανάλυση NetBeans

Εικόνα 36: Λογότυπο NetBeans

Η εφαρμογή NetBeans ανήκει στην κατηγορία των IDE (Integrated Development Environment), δηλαδή στη κατηγορία των λογισμικών που προσφέρουν ένα ολοκληρωμένο γραφικό περιβάλλον για την ανάπτυξη εφαρμογών. Αυτήν την στιγμή είναι ένα από τα πιο διάσημα λογισμικά ανάπτυξης εφαρμογών μαζί με το Eclipse και το IntelliJ IDEA της εταιρίας JetBrains. Είναι γραμμένο στην γλώσσα προγραμματισμού Java και έτσι μπορεί να εκτελεστεί στα πιο γνωστά λειτουργικά συστήματα όπως Windows, OS X, Linux, Solaris. Το 1996 ξεκίνησε η ανάπτυξη του NetBeans από έναν φοιτητή με όνομα Roman Staněk ως εργασία για το τμήμα των μαθηματικών και φυσικής του πανεπιστημίου Charles της Πράγας. Το 1997 δημιουργήθηκε μια εταιρία υπεύθυνη για την ανάπτυξη του λογισμικού. Το 1999 εξαγοράστηκε από την Sun Microsystems, η οποία άνοιξε τον κώδικα της εφαρμογής στο κοινό και από το 2010 η εφαρμογή ανήκει στην Oracle αφού η τελευταία εξαγόρασε την Sun Microsystems. Η εφαρμογή NetBeans σχεδιάστηκε για την ανάπτυξη εφαρμογών Java αλλά στην συνέχεια επεκτάθηκε με τις γλώσσες PHP, C/C++ και HTML 5. Ύστερα από τόσα χρόνια ανάπτυξης τον Νοέμβριο του 2014 έφτασε στην έκδοση 8.0.2 και σύμφωνα με το χρονοδιάγραμμα η 9η έκδοση αναμένεται τον Ιούλιο του 2015. Ένας από τους λόγους της επιλογής του NetBeans είναι το χαρακτηριστικό ότι έρχεται προεγκατεστημένο με ένα σύνολο από τεχνολογίες μειώνοντας τον φόρτο εργασίας. Μια από αυτές τις τεχνολογίες είναι ο Glassfish server που ενσωματώνει το Java Server Faces(JSF) και έτσι μπορούμε να δημιουργήσουμε την ιστοσελίδα μας και να την εμπλουτίσουμε με κώδικα Java για καλύτερη αλληλεπίδραση με τον χρήστη. Το κύριο χαρακτηριστικό της JSF είναι ότι δεν χρειάζεται η εκμάθηση νέας γλώσσας πέρα από html και java, διότι ουσιαστικά επεκτείνει τα στοιχεία της html με ιδιότητες οι οποίες εκτελούν μεθόδους από τον κώδικα της Java. Στην προκειμένη περίπτωση χρησιμοποιήσαμε το NetBeans ώστε να εμπλουτίσουμε την σελίδα μας δίνοντας την δυνατότητα στον χρήστη να ανεβάζει δικές του βάσεις, να θέτει τα ερωτήματα και να του επιστρέφονται τα αποτελέσματα[61,62].

Τα βασικά χαρακτηριστικά του που το κάνουν να υπερέχει έναντι του Eclipse είναι:

- Πιο εξελιγμένη παλέτα γραφικών εργαλείων: Το NetBeans παρέχει στον χρήστη το Standard Swing Toolkit το οποίο διαθέτει περισσότερα εργαλεία καθώς και υποστήριξη για εγκατάσταση νέων εργαλείων από χρήστες
- Καλύτερος σχεδιασμός γραφικής διεπαφής: Παρέχει την δυνατότητα στον χρήστη να παραμετροποιεί το γραφικό περιβάλλον της εφαρμογής με τη χρήση του drag-n-drop αναλόγως την αρέσκειά του, κάτι αντίστοιχο υπάρχει και στο eclipse αλλά έναντι αμοιβής
- Υποστήριξη εργαλείων: Αν και το Eclipse και το NetBeans παρέχουν υποστήριξη στο κοινώς αποδεκτό σύστημα εργαλείων το OSGi, το NetBeans παρέχει και το δικό του σύστημα εργαλείων βασισμένο στην φιλοσοφία της Java
- Δωρεάν Εκμάθηση: Παρέχεται από μεγάλους οργανισμούς (όπως πανεπιστήμια) δωρεάν εκμάθηση του προγράμματος σε αντίθεση με το Eclipse που δεν το παρέχει

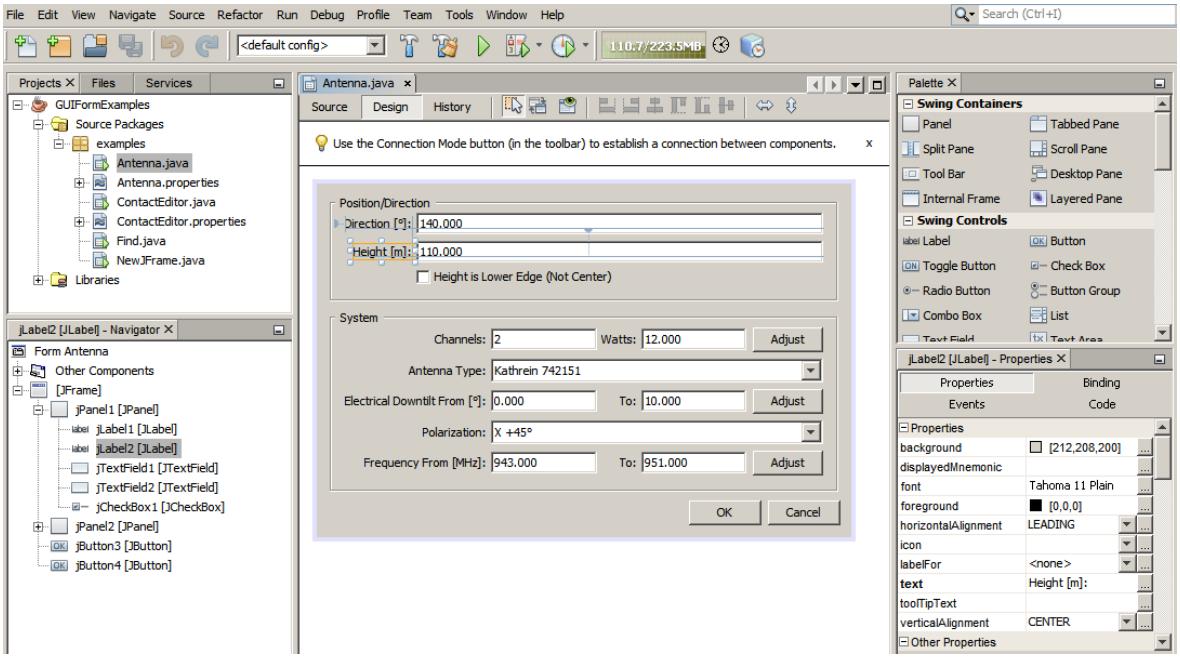
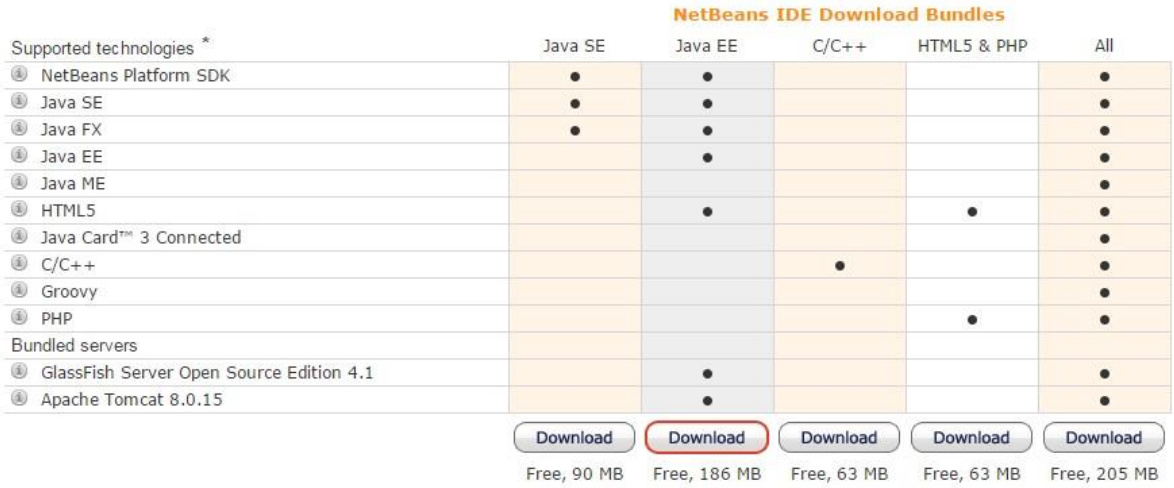


Figure 1. Ένα τυπικό περιβάλλον του NetBeans

Όπως είναι φυσικό, δεν περιμένουμε το NetBeans να έχει χαμηλή κατανάλωση στην μνήμη, μιας και το περιβάλλον του δεν είναι και ιδιαίτερα απλό. Με την εκκίνηση του NetBeans για την επεξεργασία του κώδικα χωρίς να τρέξουμε το project, το NetBeans καταναλώνει περίπου 479 MB της μνήμης RAM. Με την εκκίνηση του project και παράλληλα την εκκινηση του Glassfish Server, βλέπουμε το NetBeans να αυξάνει την «κατανάλωση» του στα 594 MB και παράλληλα ο Glassfish Server να χρησιμοποιεί 362 MB, δηλαδή συνολικά 956 MB.

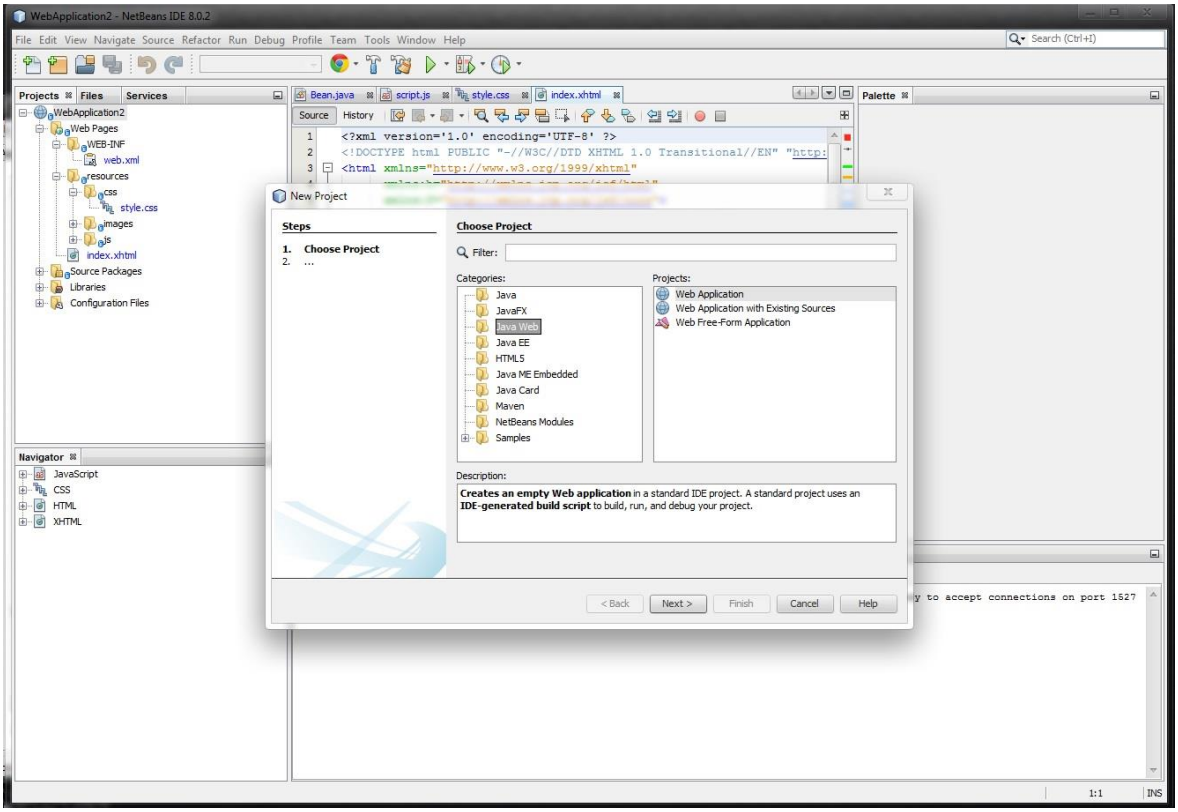
Ο κώδικας της πτυχιακής εργασίας γράφτηκε στο NetBeans και χρησιμοποιήθηκαν Java Server Faces, JavaScript, CSS, XHTML και δοκιμάστηκε στον Google Chrome για την εύρεση σφαλμάτων. Τα βήματα που πραγματοποιήθηκαν είναι τα εξής:

1. Κατέβασμα και εγκατάσταση της EE έκδοσης του NetBeans από την επίσημη ιστοσελίδα: https://netbeans.org/downloads/



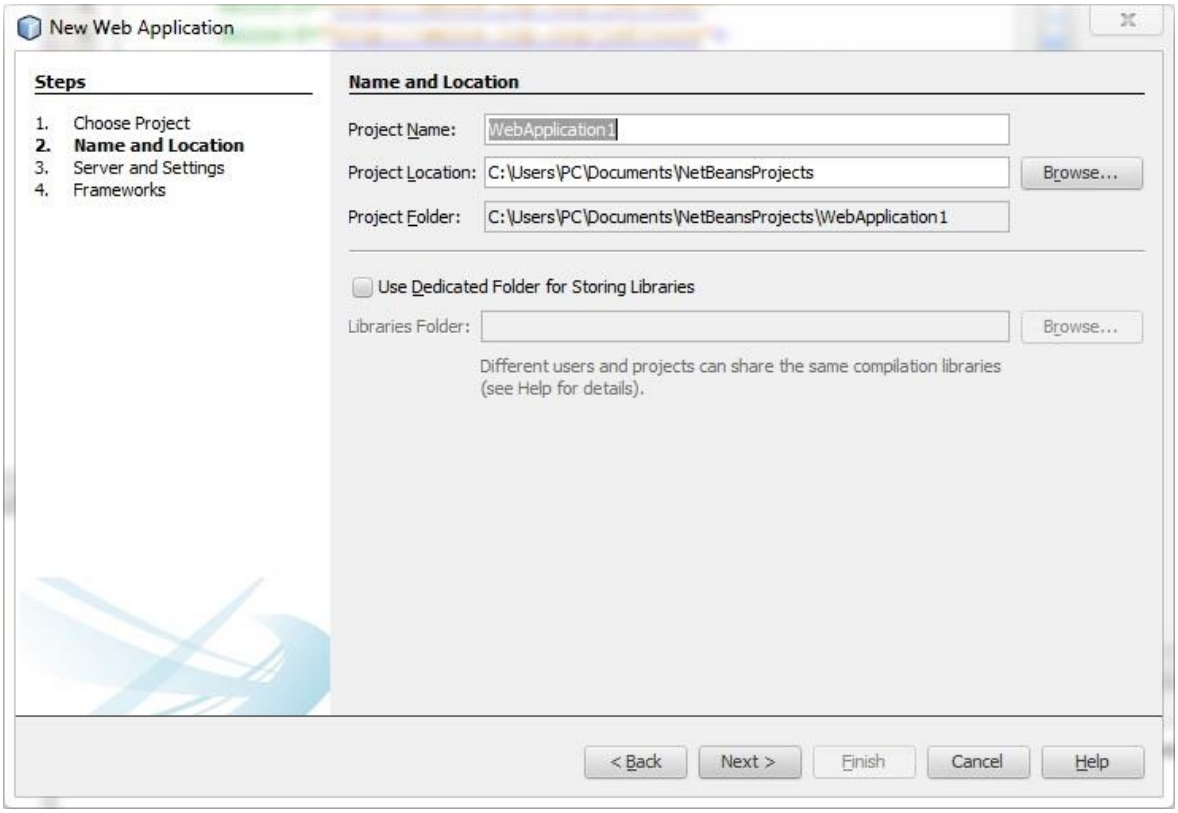
Εικόνα 37: Εκδόσεις και χαρακτηριστικά του NetBeans

2. Εκκίνηση της εφαρμογής και δημιουργίας νέου project τύπου Java Web Application



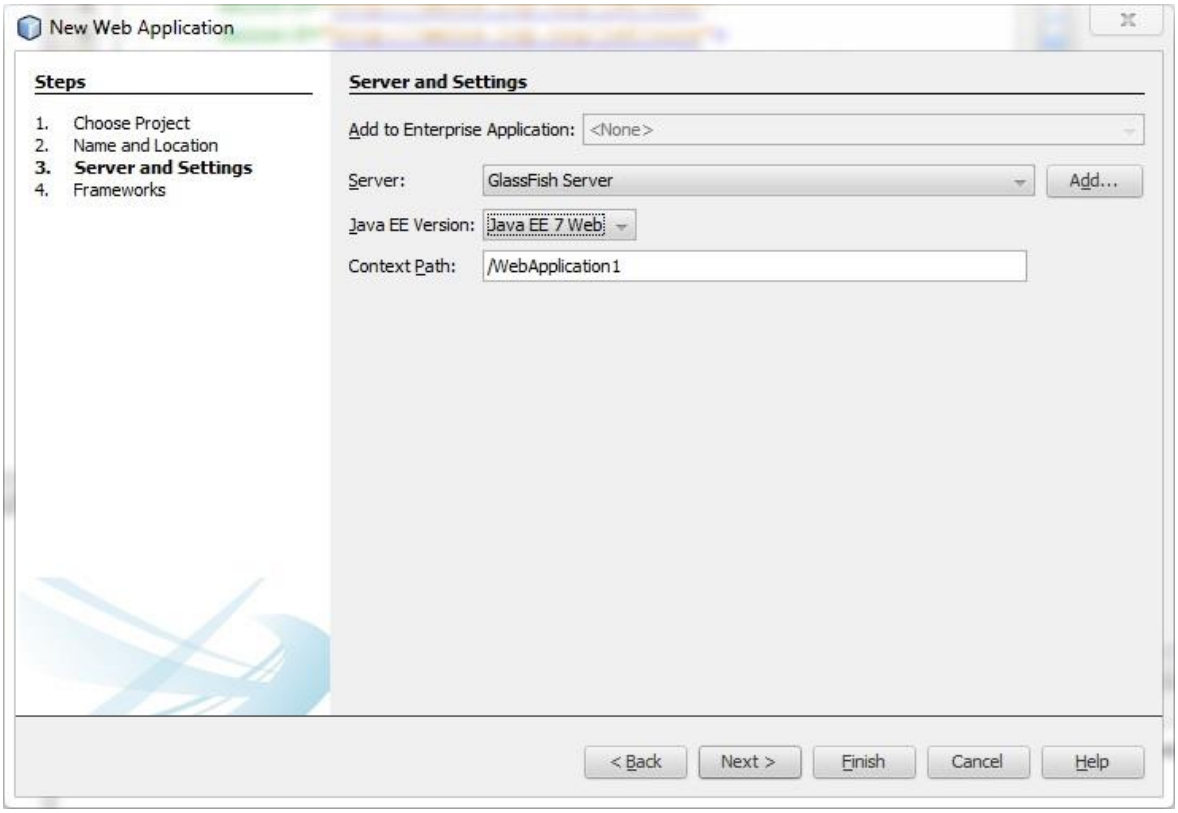
Εικόνα 38: Δημιουργία νέου project

3. Ορίζουμε το όνομα του project και τη διεύθυνση στον δίσκο



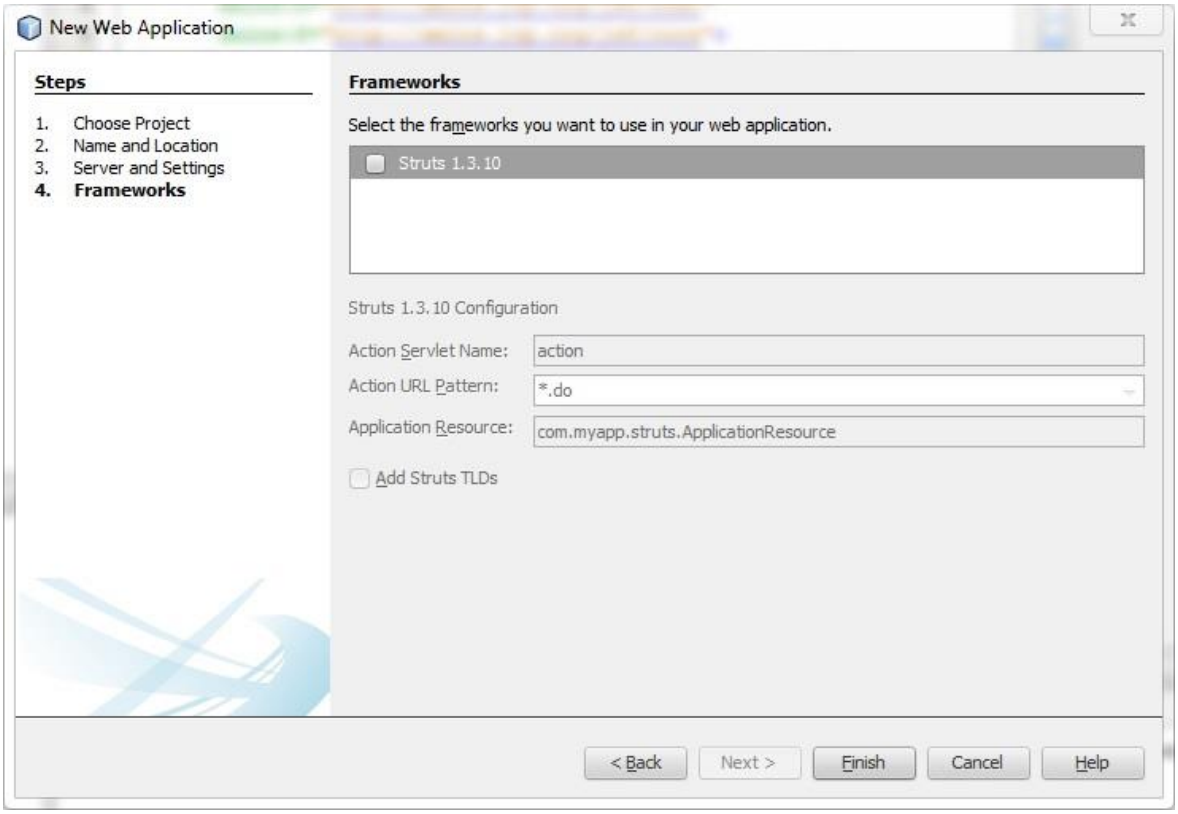
Εικόνα 39: Ορισμός ονόματος και τοποθεσίας του project

4. Ορίζουμε τον server μας και την έκδοση της Java EE



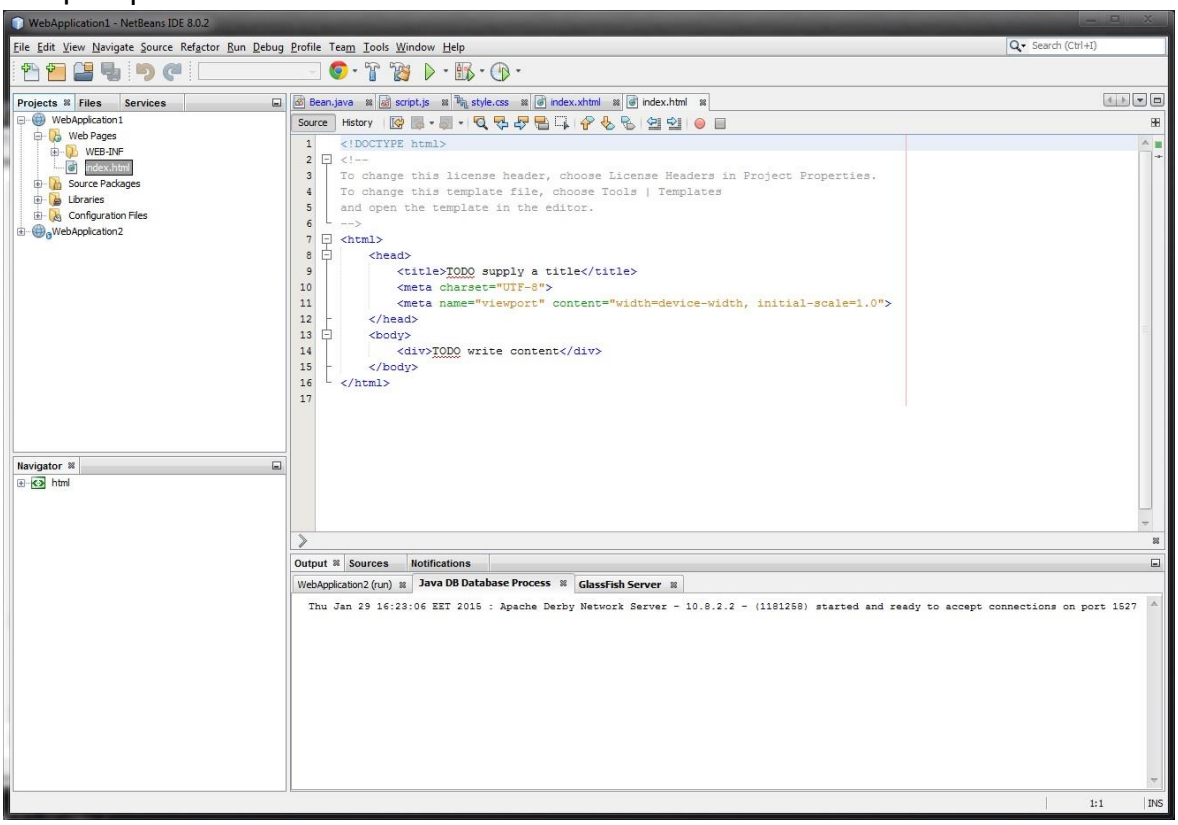
Εικόνα 40: Επιλογή server και έκδοσης Java EE

5. Στην συνέχεια, στο βήμα «Frameworks» δεν επιλέγουμε κάποιο από τα διαθέσιμα frameworks



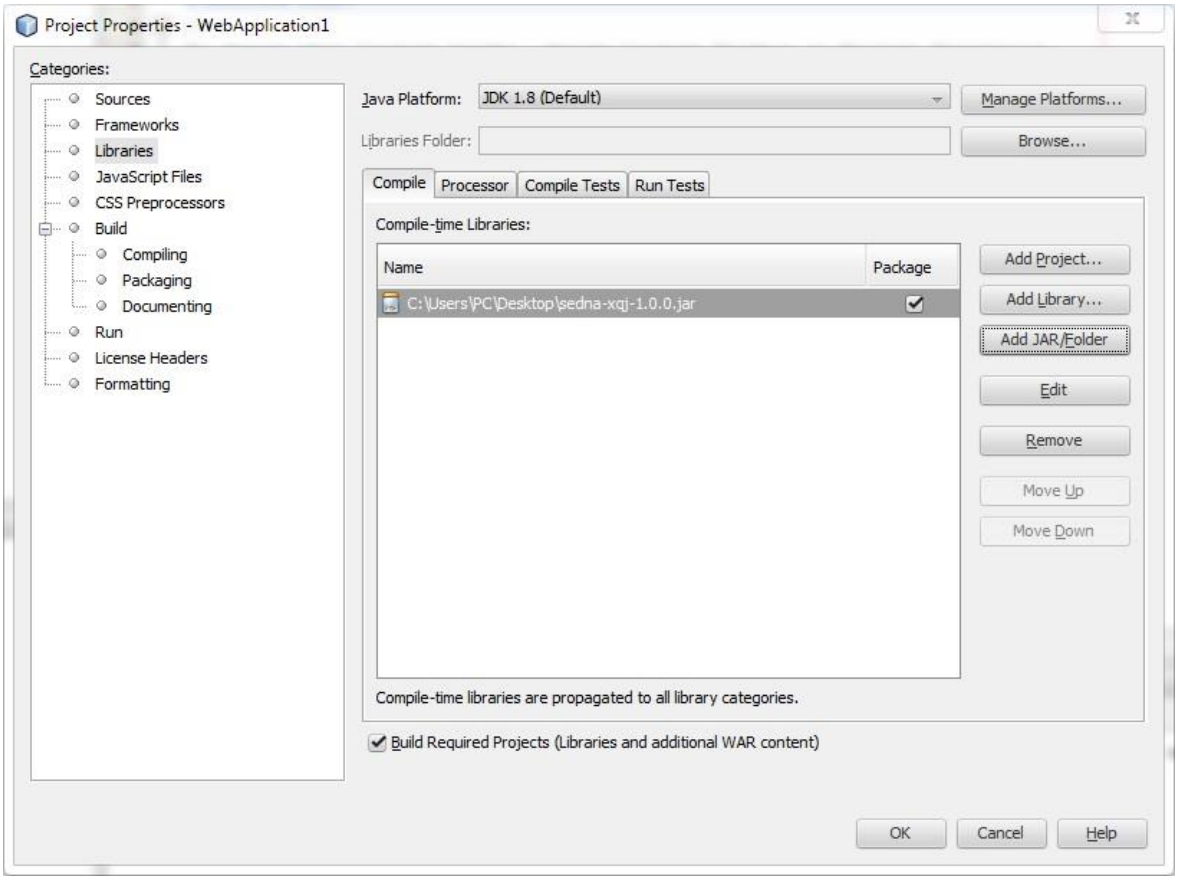
Εικόνα 41: Επιλογή Frameworks

6. Τώρα βλέπουμε το νέο μας project που περιλαμβάνει τα απολύτως απαραίτητα



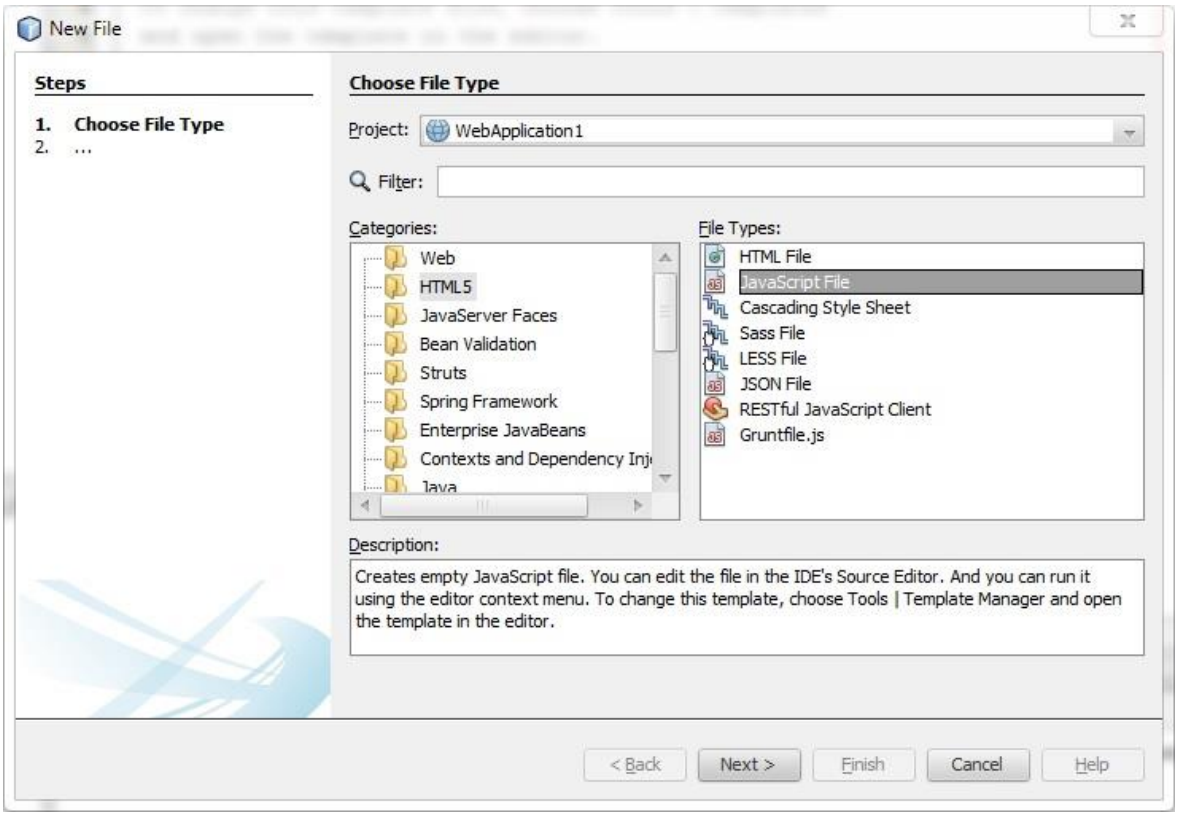
Εικόνα 42: Αρχική εικόνα project

7. Προσθέτουμε τις βιβλιοθήκες που θα μας χρειαστούν. Στην δικιά μας περίπτωση ενσωματώνουμε τις βιβλιοθήκες του sedna και του ΣΔΒΔ μας.



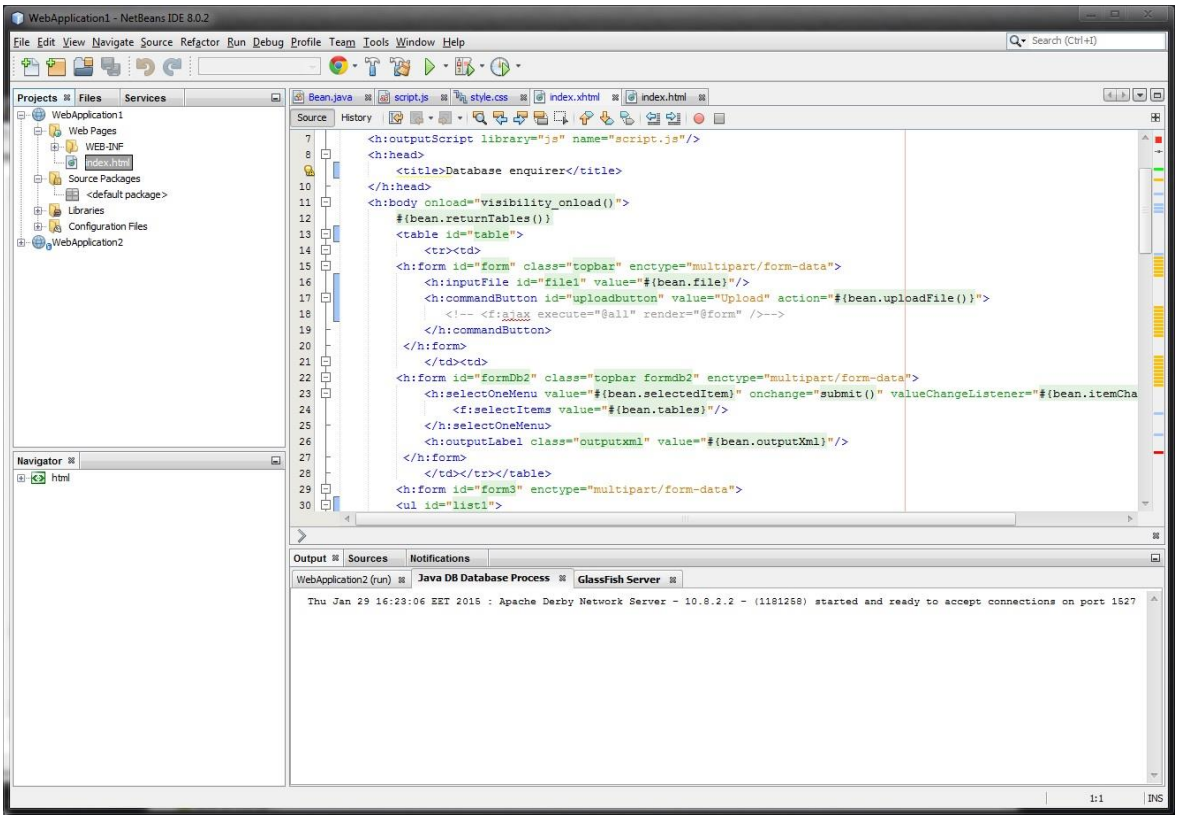
Εικόνα 43: Προσθήκη βιβλιοθηκών

8. Δημιουργούμε τα τυχόν αναγκαία αρχεία για το project



Εικόνα 44: Δημιουργία αρχείων

9. Συγγράφουμε τον κώδικα μας

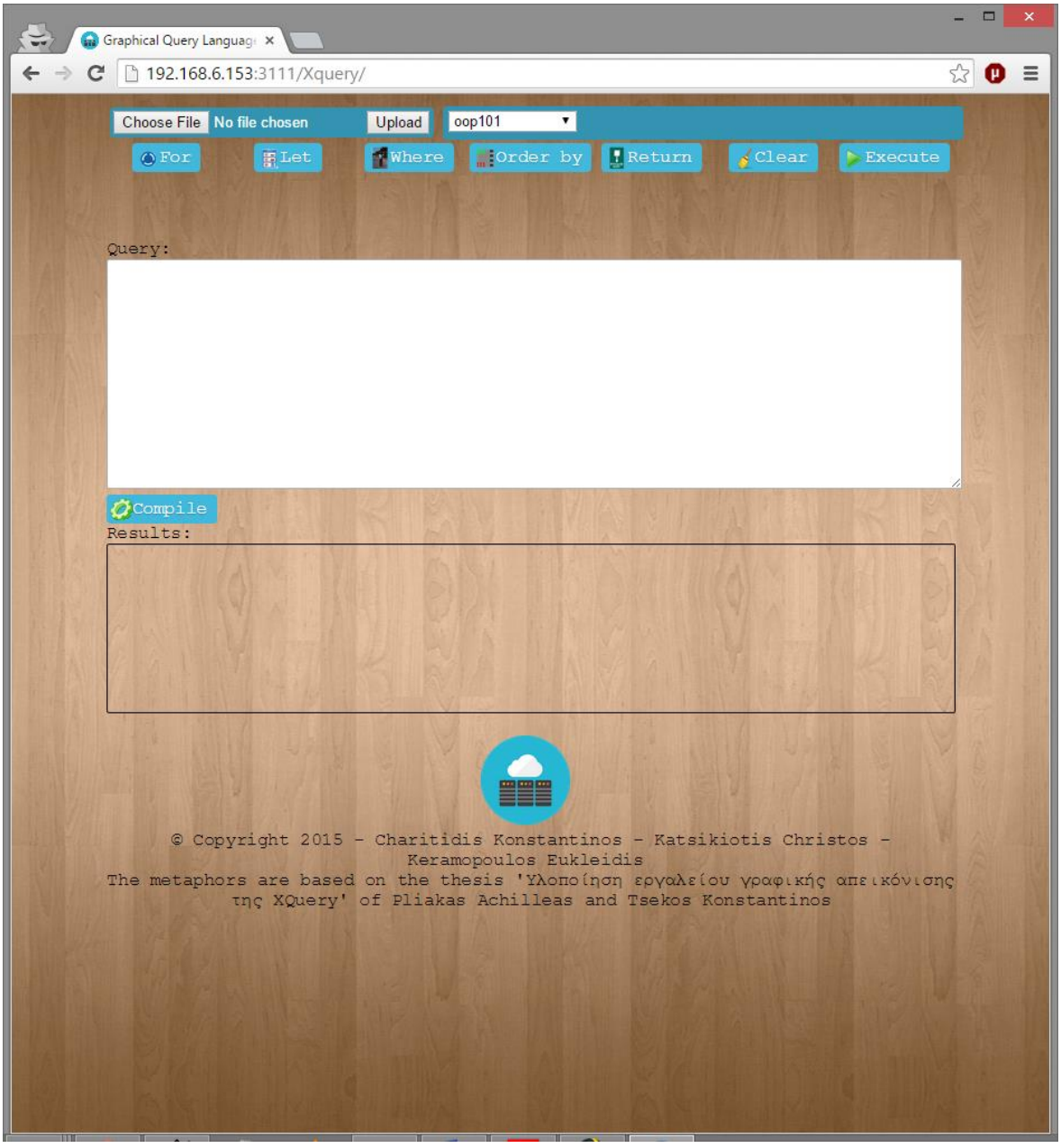


Εικόνα 45: Συγγραφή κώδικα

10. Εκκινούμε το project επιλέγοντας το πράσινο βελάκι στην μπάρα των εικονιδίων

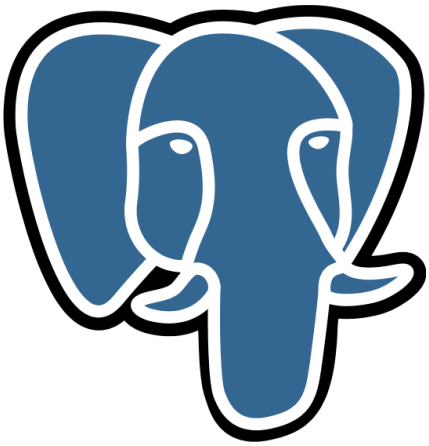
Εικόνα 46: Εκκίνηση προγράμματος

11. Τελικά, θα εμφανιστεί ο browser μας στην σελίδα του project μας



Εικόνα 47: Γραφικό περιβάλλον της εφαρμογής μας

2.6 PostgreSQL



Εικόνα 48: Λογότυπο της PostgreSQL

Με τα δυο προαναφερθέντα προγράμματα επετεύχθη η δημιουργία της διεπαφής για τον χρήστη και η σύνδεση του με την βάση XML. Την διαχείριση των σχεσιακών και αντικειμενοσχεσιακών βάσεων δεδομένων την πραγματοποιεί η PostgreSQL. Η PostgreSQL που άλλοτε αναφέρεται και ως Postgres είναι ένα ανοιχτού τύπου ΣΔΒΔ το οποίο δίνει έμφαση στην επεκτασιμότητα και στην υιοθέτηση προτύπων. Η πρώτη του έκδοση κυκλοφόρησε την 1η Μαΐου του 1995 και είναι η εξέλιξη του project Ingres στο πανεπιστήμιο Μπέρκλεϊ της Καλιφόρνιας και ύστερα από 9 χρόνια ανάπτυξης έφτασε στην έκδοση 9.4, τον Δεκέμβριο του 2014. Κατά την ανάπτυξη δόθηκε ιδιαίτερη έμφαση στην υποστήριξη των λειτουργικών συστημάτων και υποστηρίζει τα πιο πολλά λειτουργικά συστήματα της αγοράς όπως Linux, Windows, Mac OS X, Solaris και FreeBSD. Έχει κερδίσει πολλά βραβεία που εξυμνούν την ποιότητα του όπως το 2008 που ανακηρύχθηκε ως λογισμικό διαχείρισης βάσεων δεδομένων της χρονιάς ή το 2006 που οι συντάκτες του περιοδικού Linux το επέλεξαν επίσης ως λογισμικό διαχείρισης βάσεων της χρονιάς. Μαζί με το αρχείο εγκατάστασης, η PostgreSQL έρχεται και με μια γραφική διεπαφή, το pgAdmin III το οποίο είναι απευθείας ρυθμισμένο και δεν χρειάζεται κάποια ρύθμιση εκ μέρους του χρήστη, πέρα από το να συνδεθεί. Τα σημαντικότερα χαρακτηριστικά της PostgreSQL είναι[63,64,65]:

- Μεγάλη υποστήριξη από λογισμικό κλειστού τύπου: Εκτός από την εμπορικού τύπου υποστήριξη που παρέχουν και τα λογισμικά κλειστού τύπου η Postgres διαθέτει μια μεγάλη κοινότητα χρηστών που βοηθάνε στην επίλυση προβλημάτων
- Κανένα χρηματικό κόστος για την χρήση του: Λόγω του ότι είναι ανοιχτού τύπου δεν κοστίζει η αγορά του
- Υποστήριξη όλων των λειτουργικών συστημάτων της αγοράς
- Περιορισμένη ανάγκη συντήρησης της βάσης δεδομένων
- Εύκολη επεκτασιμότητα: Λόγω του ανοικτού κώδικα είναι εύκολη η επέκταση των δυνατοτήτων του λογισμικού

- Πληθώρα γραφικών διεπαφών: Όπως και στο sedna, έτσι και η Postgres υποστηρίζει την διαχείριση της βάσης δεδομένων μέσα από μια πληθώρα γραφικών διεπαφών
- Σχεδιασμένο για διαχείριση μεγάλου όγκου δεδομένων
- Μεγάλη σταθερότητα και αξιοπιστία

Τα βήματα που πραγματοποιήθηκαν στην πτυχιακή για όσον αφορά την Postgres είναι τα εξής:

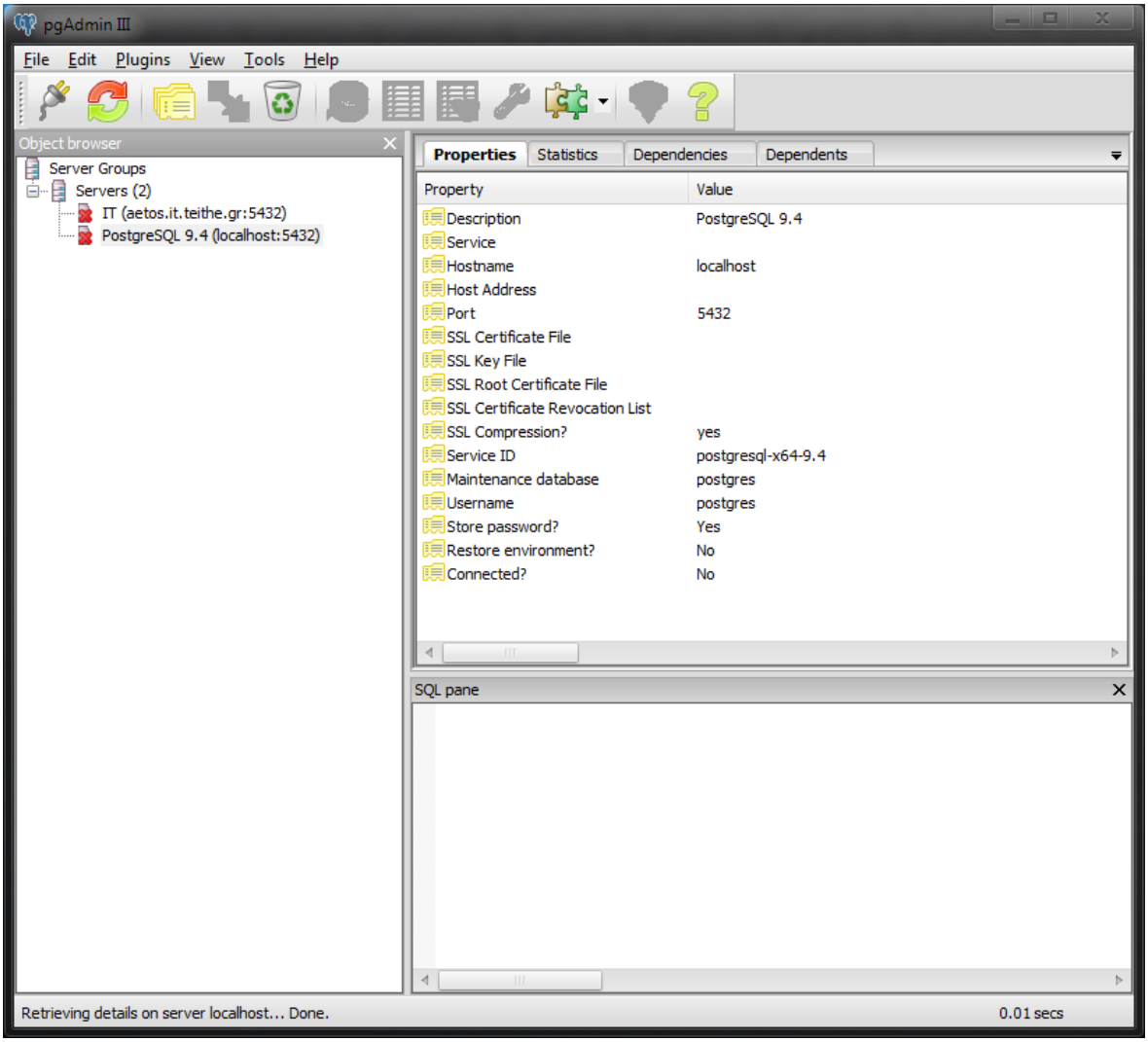
1. Μετάβαση στην επίσημη ιστοσελίδα και κατέβασμα του κατάλληλου αρχείου εγκατάστασης σύμφωνα με το λειτουργικό μας σύστημα

Εικόνα 49: Σελίδα κατεβάσματος της PostgreSQL

2. Εγκατάσταση του αρχείου

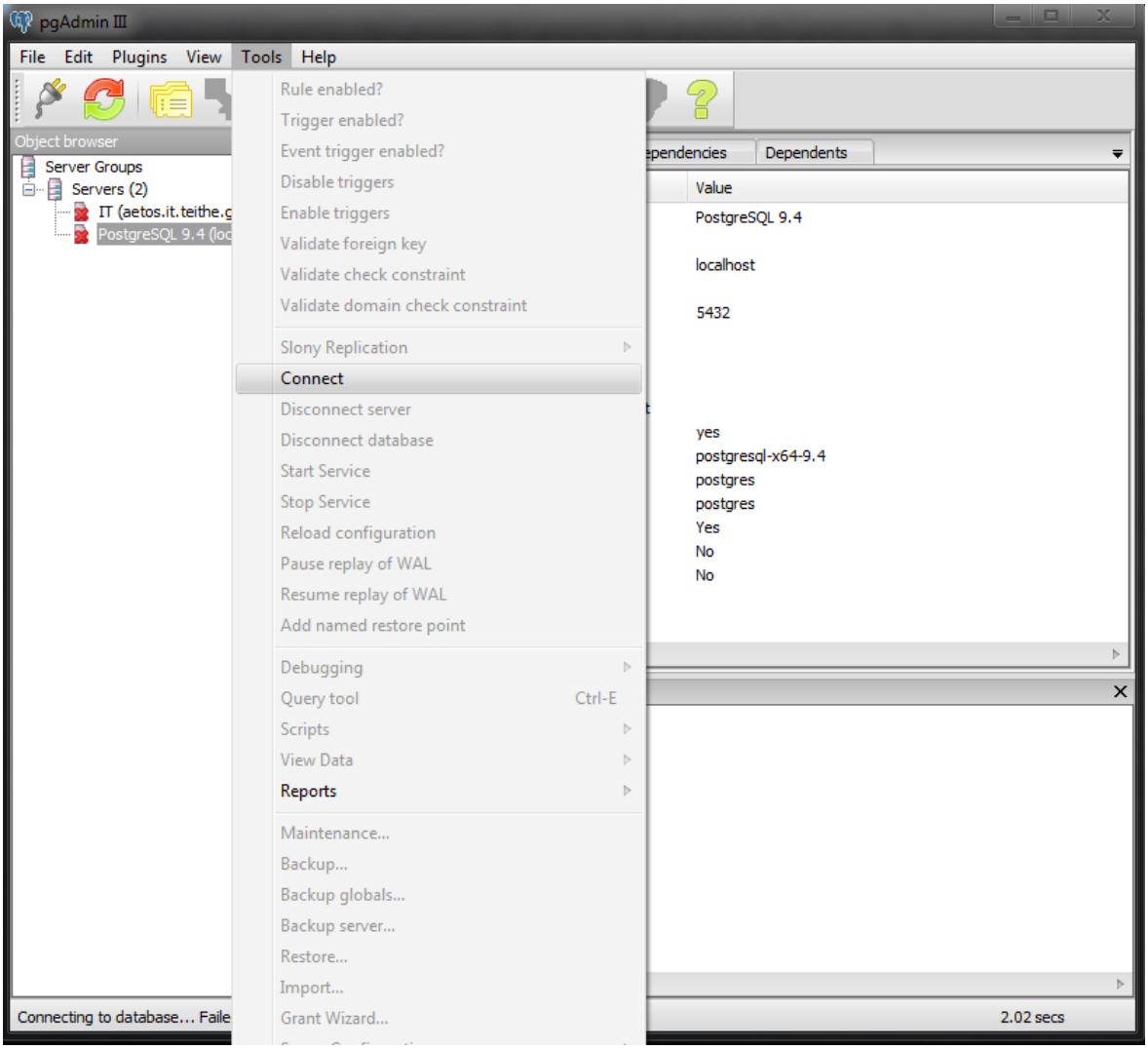
Εικόνα 50: Αρχική εικόνα εγκατάστασης

3. Άνοιγμα του προγράμματος pgAdmin III



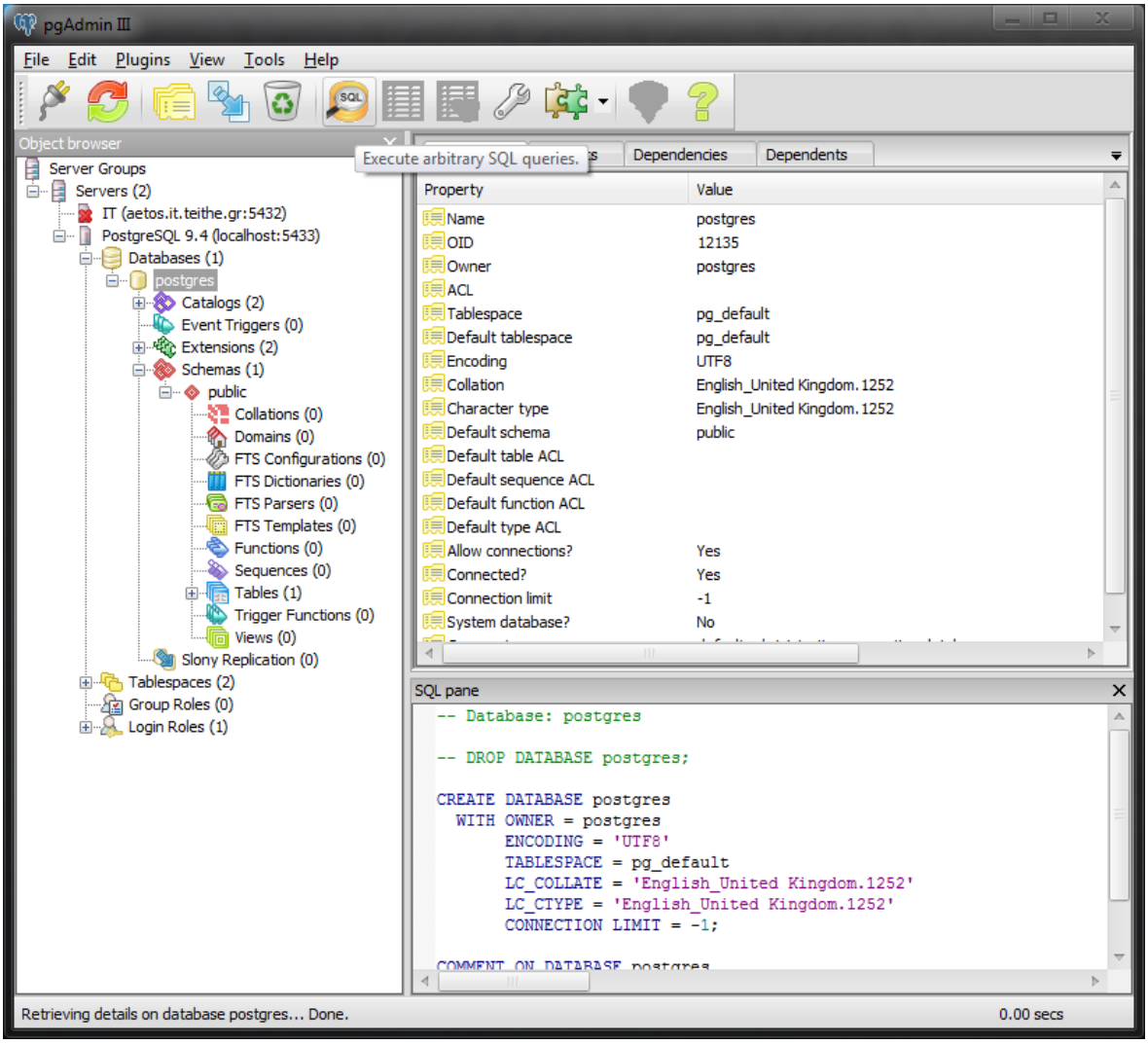
Εικόνα 51: Αρχική εικόνα του pgAdmin III

4. Επιλογή της σύνδεσης με την PostgreSQL που έχει αυτόματα δημιουργηθεί και σύνδεση με αυτήν



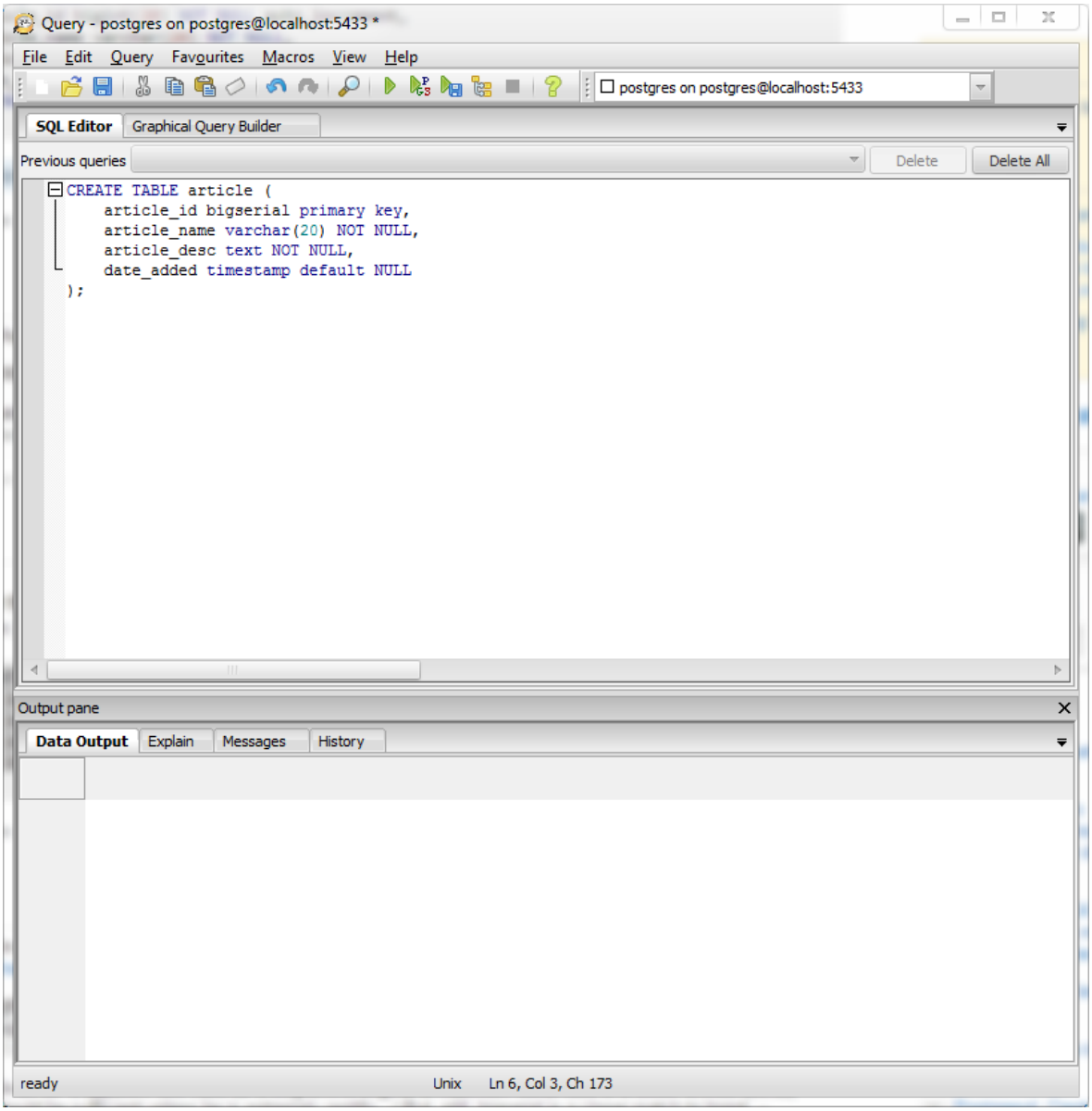
Εικόνα 52: Σύνδεση με την βάση μας

5. Επιλογή της βάσης μας από τον πίνακα περιεχομένου της σύνδεσης μας και άνοιγμα του παραθύρου για την εκτέλεση των ερωτημάτων μας



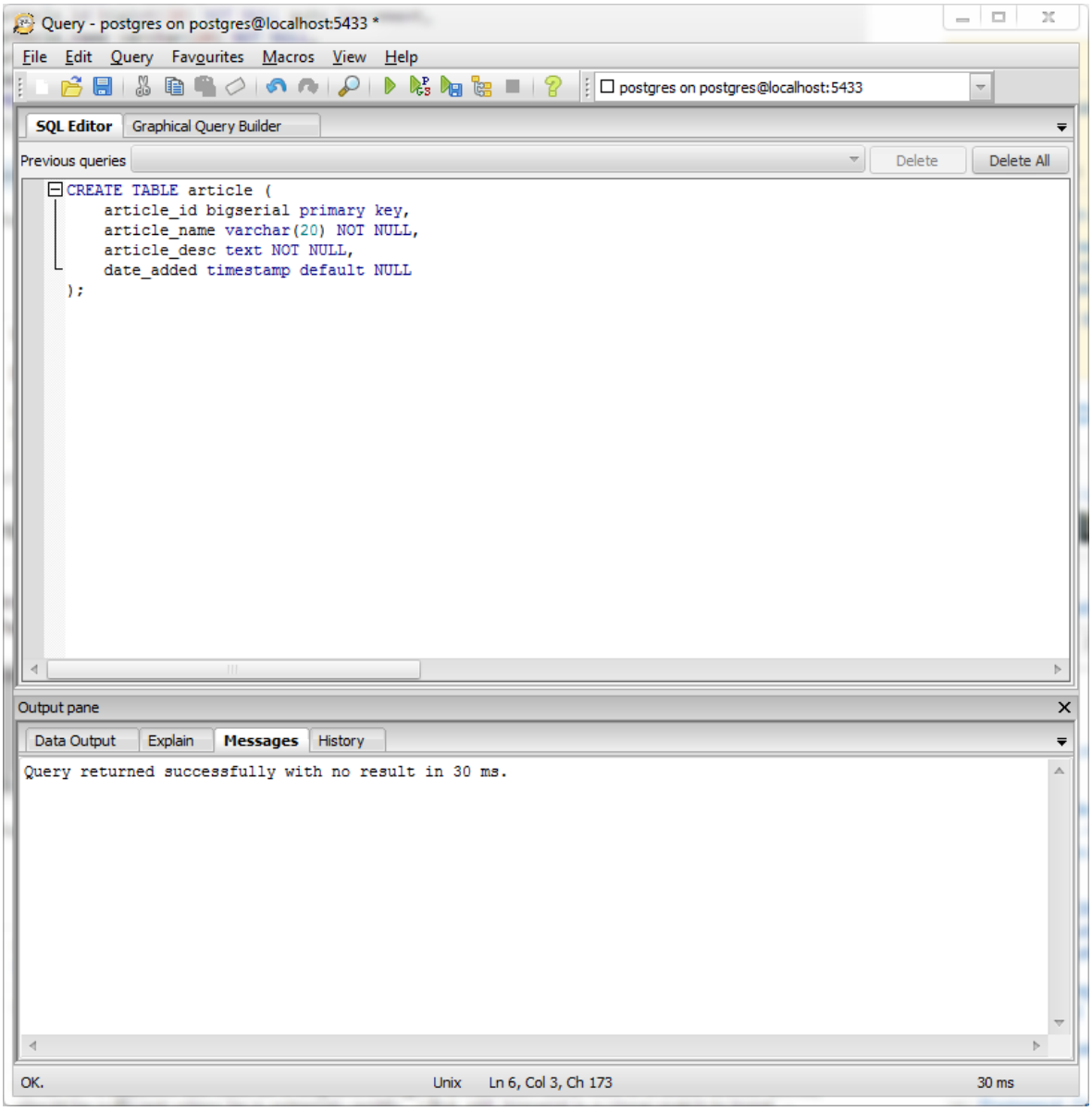
Εικόνα 53: Επιλογή της βάσης μας

6. Συγγραφή ερωτήματος SQL



Εικόνα 54: Συγγραφή ερωτήματος SQL

7. Επιβεβαιώνουμε ότι δεν υπήρξαν λάθη κατά την εκτέλεση του



Εικόνα 55: Εκτέλεση ερωτήματος SQL

2.7 Επίλογος

Πλέον ο χρήστης έχει κατανοήσει τον τρόπο που έχει δομηθεί η εφαρμογή, τις επιμέρους τεχνολογίες καθώς και τα βασικά στοιχεία της διεπαφής. Όμως παρόλο που έχει μια βασική κατανόηση της εφαρμογής, χρειάζεται μια αναλυτική περιγραφή της χρήσης της εφαρμογής συνοδευόμενη από κάποια παραδείγματα.

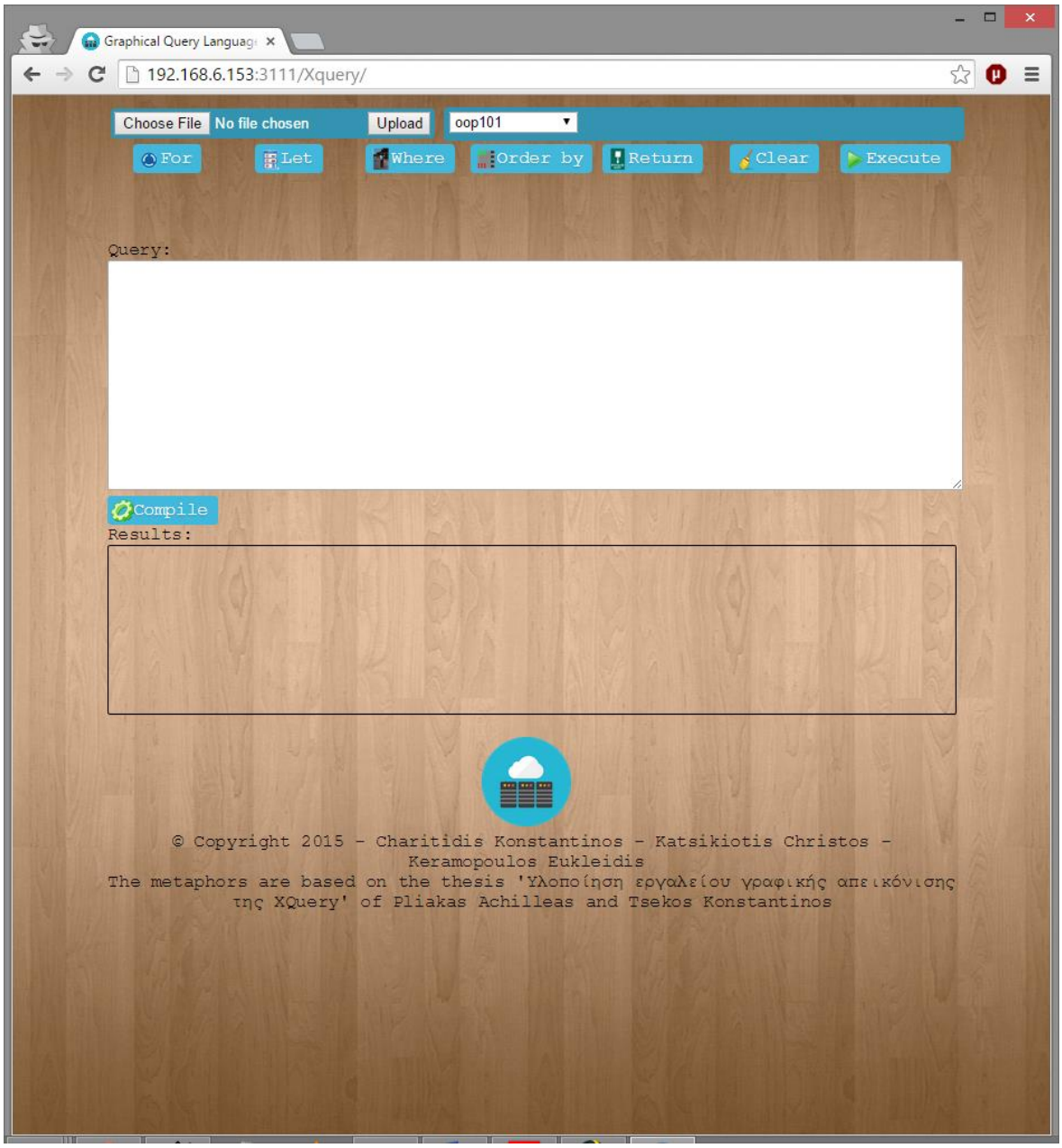
3 Περιγραφή του κώδικα της εφαρμογής

3.1 Εισαγωγή

Πλέον έχουμε αναλύσει τα επιμέρους στοιχεία της εφαρμογής αλλά δεν έχουμε αναλύσει την εφαρμογή ως σύνολο των στοιχείων. Έτσι, σε αυτό το κεφάλαιο περιγράφουμε όχι μόνο τις οδηγίες χρήσης για τον υποψήφιο χρήστη αλλά εκτελούμε και μερικά από τα ερωτήματα από το πρώτο κεφάλαιο για να επιβεβαιώσουμε τα αποτελέσματα.

3.2 Περιγραφή στοιχείων

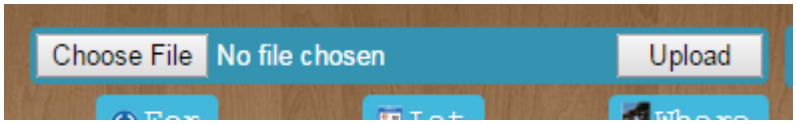
Ο χρήστης μεταβαίνοντας στην ιστοσελίδα της εφαρμογής βλέπει την παρακάτω εικόνα:



Εικόνα 56: Κεντρική εικόνα της εφαρμογής μας

Παρατηρούμε πως υπάρχουν δυο φόρμες, κάτω από αυτές εφτά κουμπιά, δυο περιοχές κειμένου καθώς και ένα κουμπί ανάμεσα τους.

3.2.1 Φόρμα ανεβάσματος αρχείων – βάσεων δεδομένων σε μορφή XML



Εικόνα 57: Φόρμα ανεβάσματος αρχείων

Αυτή η φόρμα είναι υπεύθυνη για το ανέβασμα των δεδομένων του χρήστη, τα οποία σκοπεύει να προσπελάσει. Περιέχει τρια στοιχεία, δυο κουμπιά και ένα κείμενο. Το κουμπί με το κείμενο «Choose File» ανοίγει ένα παράθυρο διαλόγου στο οποίο ο χρήστης επιλέγει το αρχείο xml από τον προσωπικό του υπολογιστή. Το κείμενο που περιέχει μηνύματα ως προς τον χρήστη για το αν έχει επιλέξει ένα αρχείο ή όχι και το τελευταίο κουμπί με κείμενο «Upload» όπου με το πάτημα του ξεκινάει η διαδικασία μεταφοράς της βάσης δεδομένων στον server που φιλοξενείται η ιστοσελίδα, στην περίπτωση μας στον server του τμήματος Πληροφορικής του ΑΤΕΙΘ.

3.2.2 Φόρμα επιλογής πίνακα

Εικόνα 58: Φόρμα επιλογής πίνακα

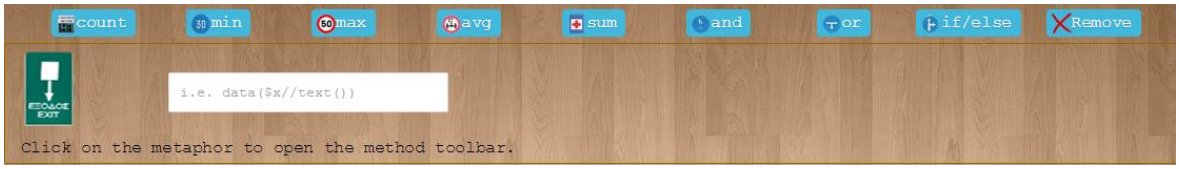
Θα μπορούσε να περιγραφθεί ως το αντίστοιχο της προηγούμενης φόρμας αλλά γίνεται επιλογή των δεδομένων από το αντίστοιχο ΣΔΒΔ την PostgreSQL. Περιέχει δυο στοιχεία, μια λίστα με τους πίνακες που υπάρχουν ήδη φορτωμένοι στην Postgres και ένα μήνυμα επιβεβαίωσης για τον χρήστη. Επιλέγοντας έναν πίνακα από την λίστα, αυτόματα η εφαρμογή μετατρέπει τον πίνακα σε μορφή xml και τον φορτώνει στο sedna. Αν γίνει επιτυχώς το κείμενο αλλάζει σε «Table has been exported to ‘όνομα-πίνακα.xml'»

3.2.3 Κουμπιά εντολών

Εικόνα 59: Λίστα κουμπιών εντολών

Περιέχονται εφτά κουμπιά, εκ των οποίων τα πέντε από αυτά αντιστοιχίζονται με τις βασικές εντολές της XQuery. Αυτά τα κουμπιά όταν πατηθούν από τον χρήστη εμφανίζουν νέα κουτιά κειμένου στα οποία καλείται ο χρήστης να εισάγει τιμές ώστε να προστεθεί στο ερώτημα η εντολή που σκοπεύει να εκτελέσει. Κάποια από αυτά

τα κουμπιά δέχονται νέες εντολές, έτσι εμφανίζονται μερικά ακόμα κουμπιά στην διάθεση του χρήστη. Αυτό συμβαίνει στο κουμπί με το κείμενο «Return» όπως φαίνεται στην παρακάτω εικόνα.

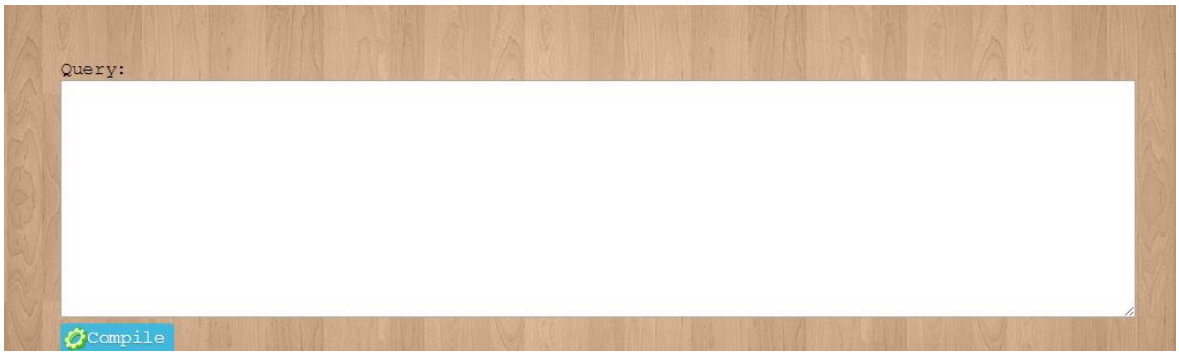


Εικόνα 60: Χρήση του metaphor Return

Το έκτο κουμπί, με το κείμενο «Clear» σβήνει το ερώτημα που έχουμε δημιουργήσει μέχρι αυτήν την στιγμή, καθώς κρύβει ότι νέα κουτιά κειμένου έχουν εμφανιστεί για κάθε εντολή.

Το τελευταίο κουμπί, με το κείμενο «Execute» τρέχει το ερώτημα που έχει κατασκευάσει ο χρήστης.

3.2.4 Περιοχή εντολής



Εικόνα 61: Κενή περιοχή εντολής

Ο χρήστης καθώς κατασκευάζει την εντολή, κάνοντας χρήση των κουμπιών εντολών και εισάγοντας τιμές, δημιουργείται παράλληλα από την εφαρμογή το ερώτημα ως προς εκτέλεση και εισάγεται στην περιοχή εντολής. Η περιοχή εντολής εκτός του ότι επιτρέπει στον χρήστη να επεξεργαστεί την εντολή, του επιτρέπει να γράψει μόνος την εντολή χωρίς την χρήση των κουμπιών και των αυτοματισμών. Έχοντας ολοκληρώσει το ερώτημα η περιοχή εντολής μοιάζει κάπως έτσι:

Εικόνα 62: Συμπληρωμένη περιοχή εντολής

3.2.5 Κουμπί Compile

Εικόνα 63: Κουμπί Compile

Πατώντας το κουμπί Compile, αποθηκεύεται στον server το ερώτημα που δημιούργησε ο χρήστης στα προηγούμενα βήματα σε μορφή xml, συγκεκριμένα σε ένα αρχείο με όνομα «query.xml».

3.2.6 Περιοχή αποτελεσμάτων

Εικόνα 64: Κενή περιοχή αποτελεσμάτων

Στην περιοχή αποτελεσμάτων επιστρέφονται τα αποτελέσματα των ερωτημάτων.

3.2.7 Κουμπί Execute

Εικόνα 65: Κουμπί Execute

Το κουμπί execute πυροδοτεί την εκτέλεση του ερωτήματος που δημιουργήθηκε πατώντας το κουμπί «compile», αποθηκεύει τα αποτελέσματα τοπικά σε μορφή xml(«result.xml») και τα επιστρέφει στον χρήστη στην περιοχή αποτελεσμάτων.

3.3 Βήματα χρήσης

Σε αυτήν την υπό-ενότητα θα περιγραφθούν αναλυτικά τα βήματα που πρέπει να εκτελέσει ο χρήστης ώστε να ανεβάσει ένα δικό του αρχείο xml με δικά του δεδομένα, καθώς και πως θα εκτελέσει ένα ερώτημα και θα λάβει τα αποτελέσματα.

Υποθέτουμε ότι στην περίπτωση μας ο χρήστης έχει δεδομένα που ταιριάζουν με το παράδειγμα της ανάλυσης της δομής της XML στο πρώτο κεφάλαιο, δηλαδή περιγραφή των μαθητών μια τάξης με την εξής δομή:

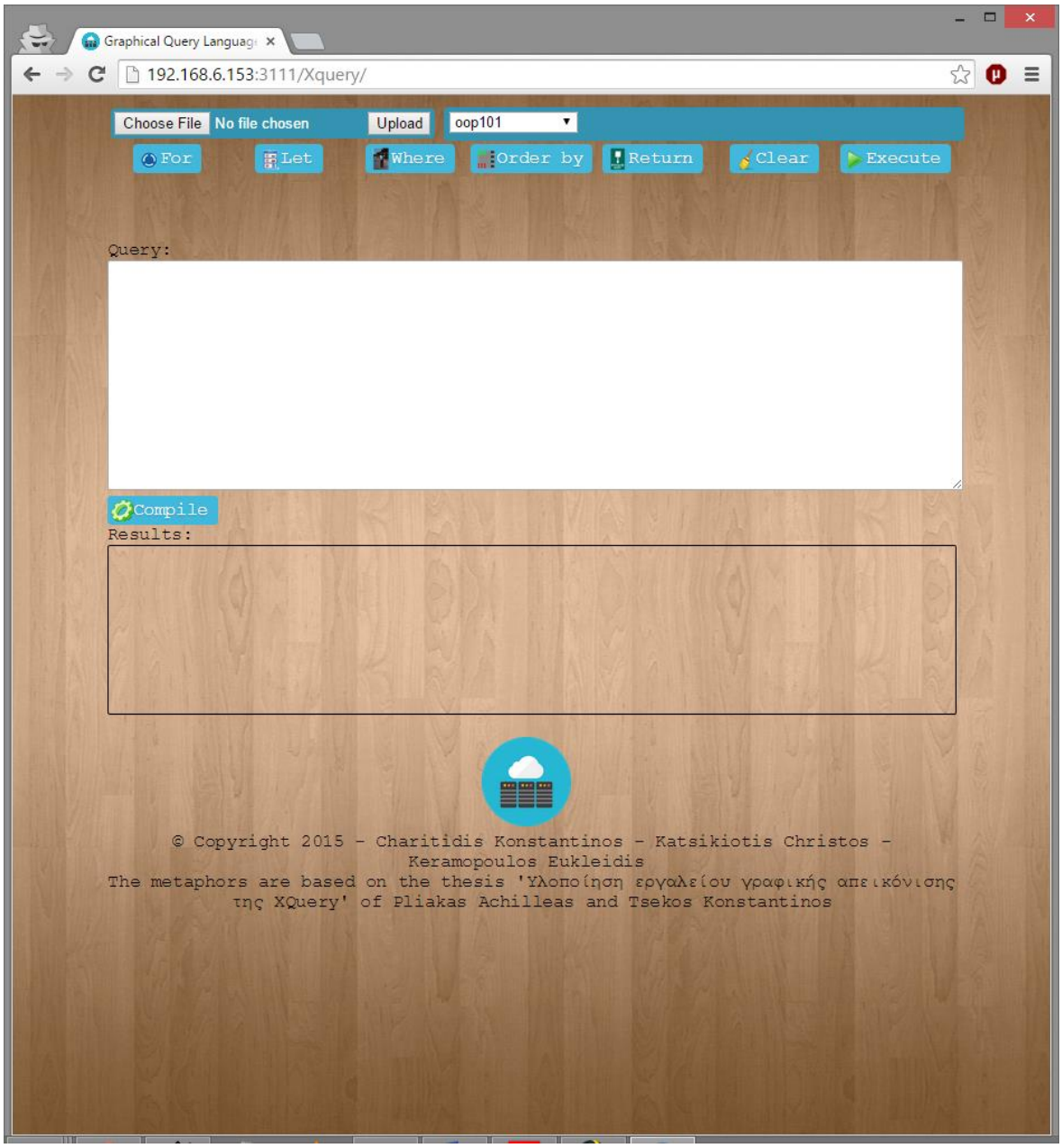
```
<student>
        <name></name>
        <surname></surname>
        <am></am>
        <reg_year></reg_year>
        <email></email>
        <phone></phone>
        <address1></address1>
</student >
```

Έστω το αρχείο με τα δεδομένα για τους μαθητές μιας τάξης έχει όνομα «student.xml» και περιέχει εννιά εγγραφές με μαθητές.

Παράδειγμα 1 – Αναζήτηση μαθητών που έχουν γραφτεί μετά το 2006

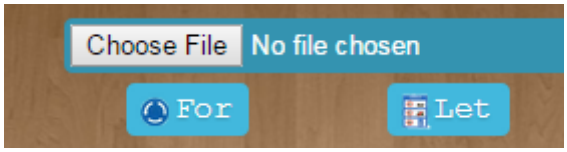
Τα βήματα που θα πρέπει να πραγματοποιήσει ο χρήστης είναι:

1. *Μετάβαση στην ιστοσελίδα της πτυχιακής εργασίας*



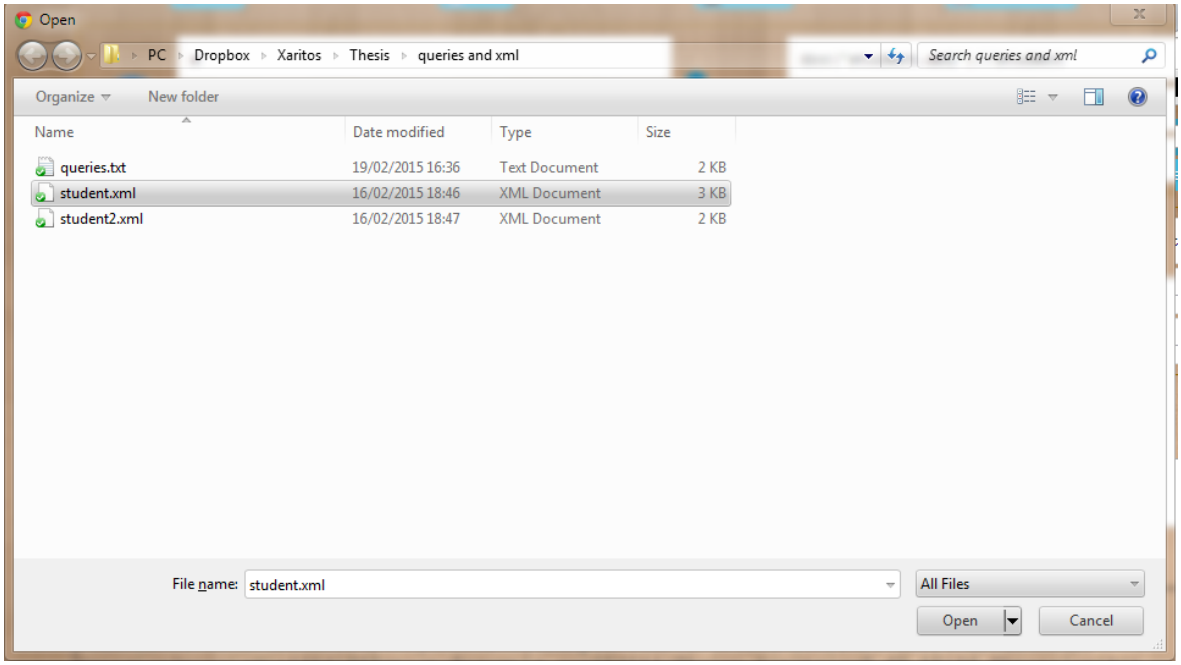
Εικόνα 66: Αρχική εικόνα εφαρμογής

2. *Πάτημα του κουμπιού «Choose File»*



Εικόνα 67: Επιλογή αρχείου

3. Επιλογή του αρχείου «student.xml» από τον τοπικό *δίσκο*

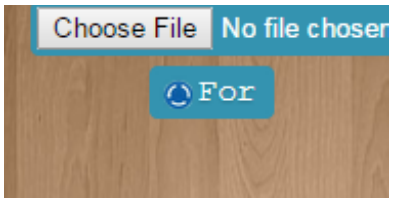


Εικόνα 68: Επιλογή αρχείου από τον τοπικό δίσκο

4. Ανέβασμα του αρχείου στον server πατώντας το κουμπί *«Upload»*

Εικόνα 69: Ανέβασμα επιλεγμένου αρχείου

5. *Πάτημα στο κουμπί for*

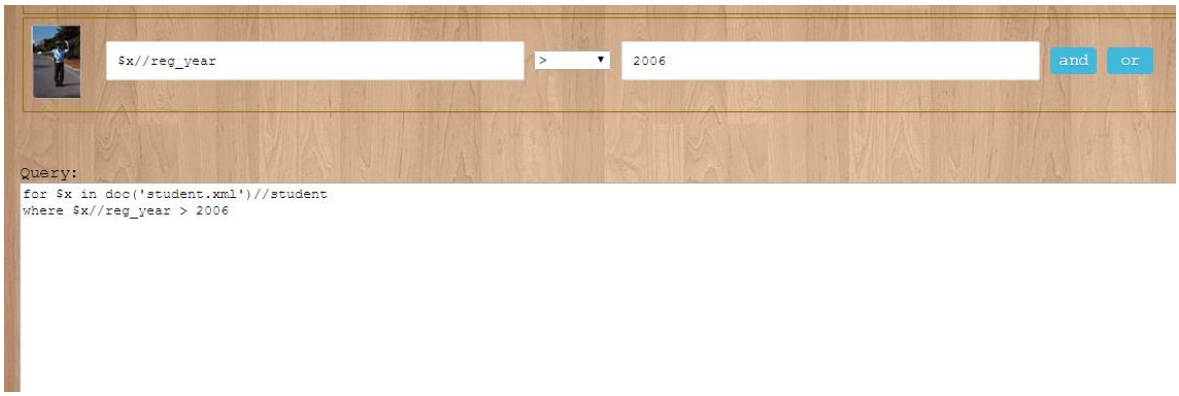


Εικόνα 70: Κουμπί for

6. *Πάτημα του κουμπιού του metaphor for και εισαγωγή της μεταβλητής μας καθώς και το όνομα του αρχείου με την βάση δεδομένων που ανεβάσαμε στο βήμα δυο*

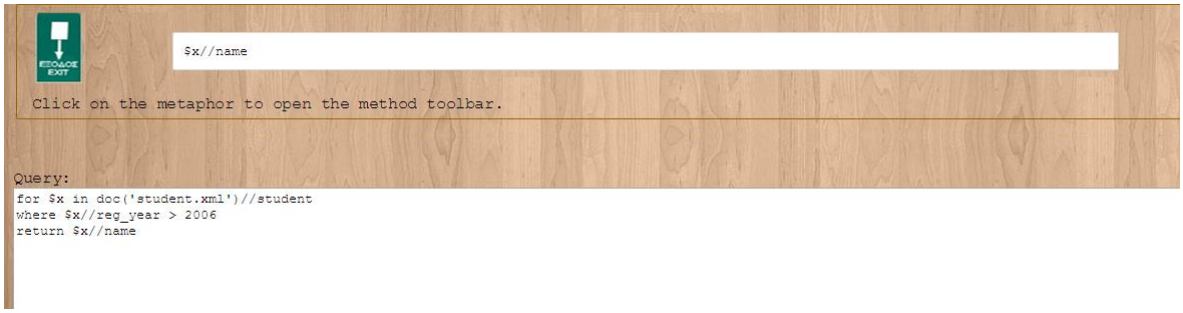
Εικόνα 71: Εισαγωγή εντολής for

7. *Εισαγωγή του κριτηρίου του ύψους πατώντας το metaphor where και εισάγοντας το ελάχιστον ύψος*



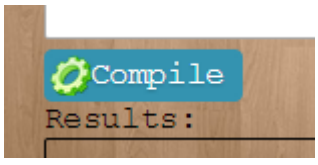
Εικόνα 72: Εισαγωγή εντολής where

8. *Εισαγωγή της εντολής επιστροφής του ονόματος των μαθητών που πληρούν το κριτήριο του προηγούμενου βήματος*



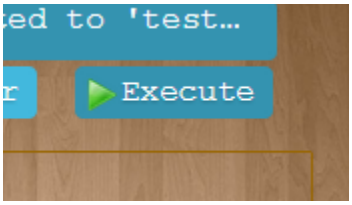
Εικόνα 73: Εισαγωγή εντολής return

9. *Πάτημα του κουμπιού «Compile»*



Εικόνα 74: Κουμπί compile

10. Πάτημα του κουμπιού «Execute»



Εικόνα 75: Κουμπί execute

11. Τα αποτελέσματα εμφανίζονται στην περιοχή κειμένου κάτω από το «Results».

Εικόνα 76: Εμφάνιση αποτελεσμάτων

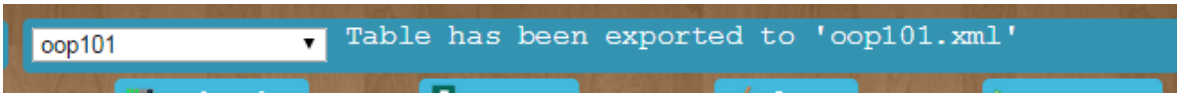
Παράδειγμα 2 –Ένωση δεδομένων από δυο πίνακες και χρήση της ενσωματωμένης συνάρτηση «concat»

Έστω ότι ο χρήστης έχει έναν πίνακα με όνομα «OOP101» στο ΣΔΒΔ σχεσιακού μοντέλου και έχει την παρακάτω δομή:

```
<oop101>
  <am></am>
  <absence></absence>
  <proodos></proodos>
  <exam></exam>
  <grade></grade>
</oop101>
```

Ο χρήστης θέλει να μάθει το όνομα των μαθητών που είναι εγγεγραμμένοι σε αυτό το μάθημα συνδέοντας το ΑΜ από αυτόν τον πίνακα με τον πίνακα «students.xml» που θεωρούμε ότι είναι ήδη ανεβασμένο στον server. Αυτό γίνεται με την εκτέλεση των παρακάτω βημάτων:

1. *Επιλογή του πίνακα από την φόρμα επιλογής πίνακα*



Εικόνα 77: Επιλογή πίνακα

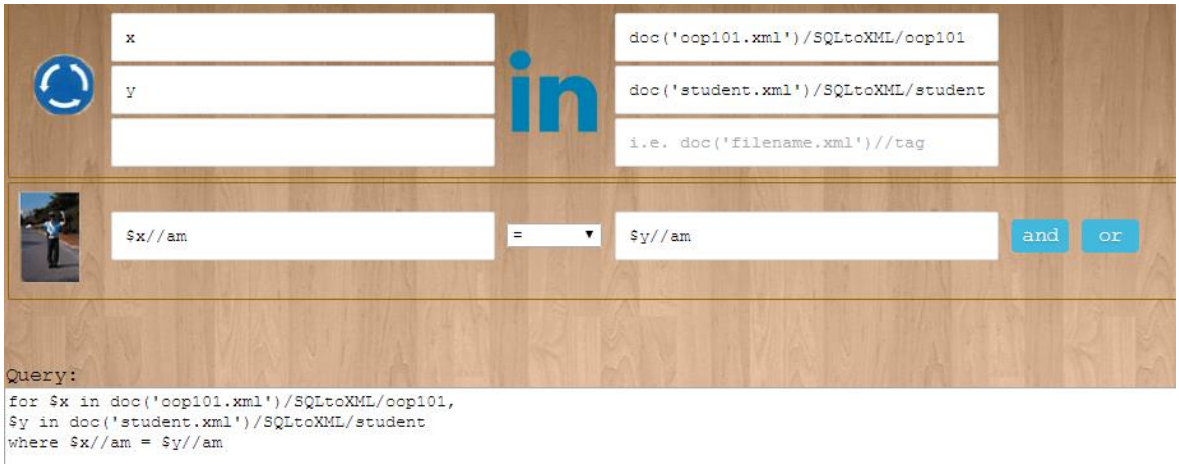
2. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101»*

Εικόνα 78: Προσπέλαση του 1ου πίνακα

3. *Συγγραφή εντολής για προσπέλαση του πίνακα με τους φοιτητές «student.xml»*

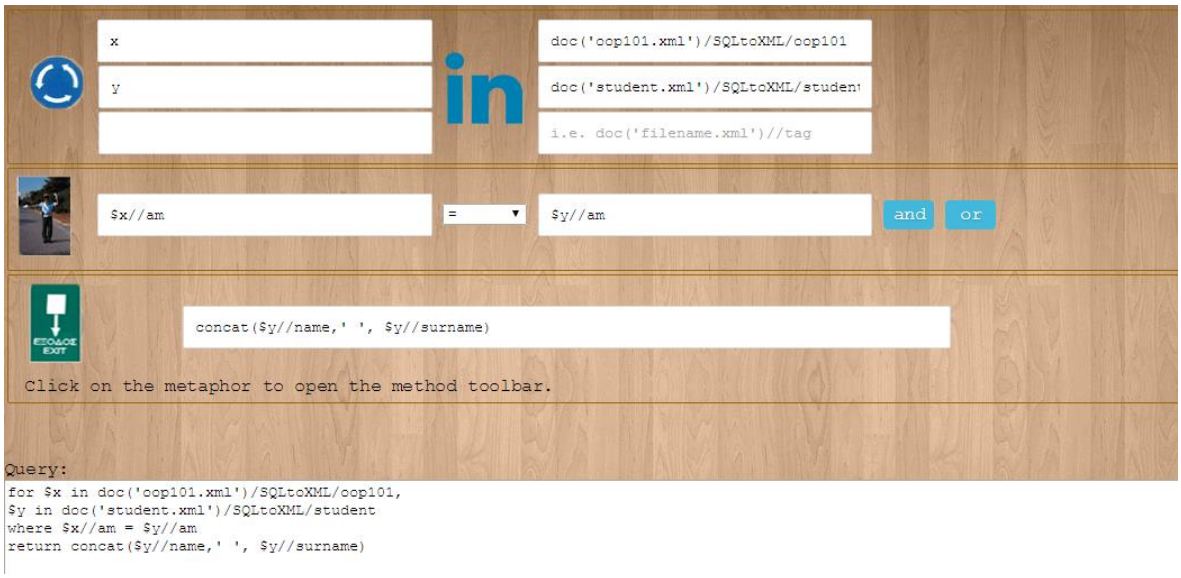
Εικόνα 79: Προσπέλαση του 2ου πίνακα

4. *Γίνεται η συσχέτιση των δεδομένων από τον πρώτο πίνακα με τον δεύτερο μέσω του πεδίου «ΑΜ»*



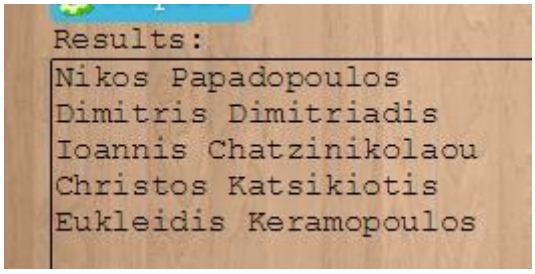
Εικόνα 80: Συσχέτιση δεδομένων των πινάκων μέσω της εντολής Where

5. *Επιστροφή των δεδομένων και ένωση δυο πεδίων με την χρήση την ενσωματωμένης συνάρτησης «concat»*



Εικόνα 81: Επιστροφή τιμών και συνένωση τους με την συνάρτηση concat

6. *Πάτημα του κουμπιού «Compile» και «Execute»*
7. *Προβολή αποτελεσμάτων*



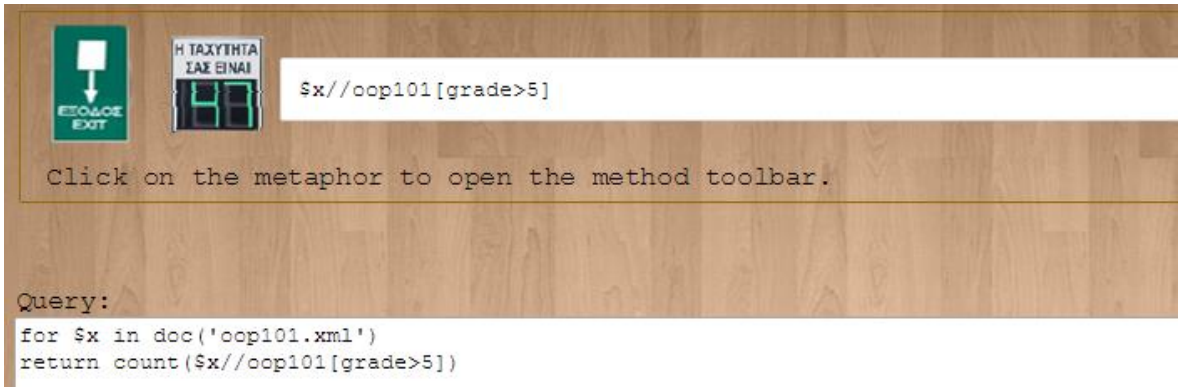
Εικόνα 82: Αποτελέσματα

Παράδειγμα 3 – Εύρεση αριθμού φοιτητών που έγραψαν πάνω από πέντε στο εργαστήριο «OOP101» με την χρήση του metaphor count

1. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101»*

Εικόνα 83: Συγγραφή εντολής for για προσπέλαση του πίνακα

2. *Εισαγωγή εντολής return μαζί με το metaphor count*



Εικόνα 84: Εισαγωγή εντολής return

3. *Πάτημα κουμπιών «compile» και «execute»*
4. *Προβολή αποτελεσμάτων*

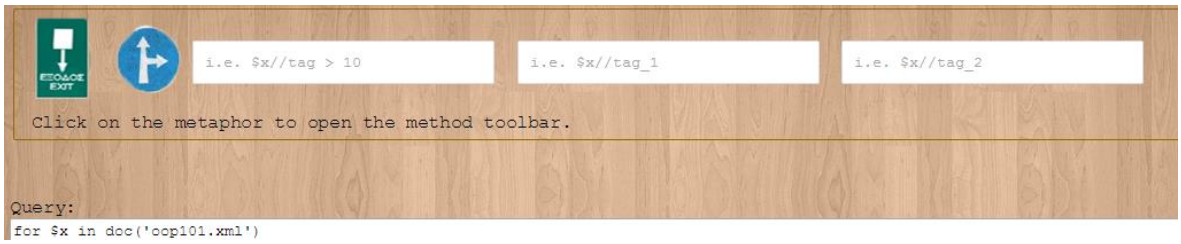
Εικόνα 85: Αποτελέσματα

Παράδειγμα 4 – Εύρεση αν όλοι οι φοιτητές πέρασαν το εργαστήριο «OOP101» με την χρήση του metaphor if/else και εμφάνιση κατάλληλου μηνύματος

1. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101»*

Εικόνα 86: Συγγραφή εντολής for για προσπέλαση του πίνακα

2. *Εισαγωγή του metaphor If/Else αφού επιλέξουμε την εντολή return*



Εικόνα 87: Εισαγωγή του metaphor if/else μαζί με την εντολή return

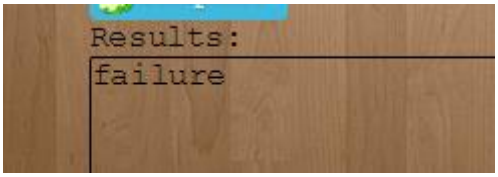
3. *Συμπλήρωση και των τρια πεδίων*

Εικόνα 88: Συμπλήρωση της If/Else

Θα πρέπει να θυμάται ο χρήστης ότι στην συγγραφή αλφαριθμητικών όπως στα πεδία Then και Else θα πρέπει να εισάγει διπλά εισαγωγικά “”.

4. *Πάτημα κουμπιών «compile» και «execute»*

5. *Προβολή αποτελεσμάτων*



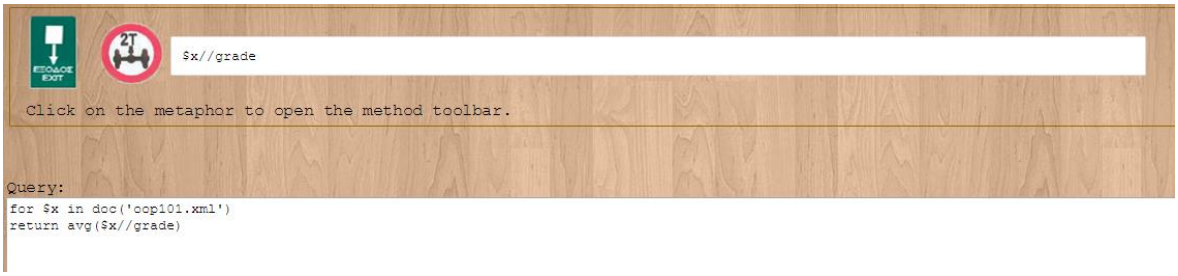
Εικόνα 89: Αποτελέσματα

Παράδειγμα 5 – Εύρεση μέσου όρου των φοιτητών του «OOP101» με την χρήση του metaphor avg

1. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101»*

Εικόνα 90: Συγγραφή εντολής for για προσπέλαση του πίνακα

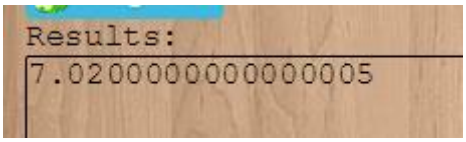
2. *Εισαγωγή της εντολής return με την χρήση του metaphor avg*



Εικόνα 91: Εισαγωγή του metaphor avg μαζί με την εντολή return

3. *Πάτημα κουμπιών "compile" και "execute"*

4. *Προβολή αποτελεσμάτων*

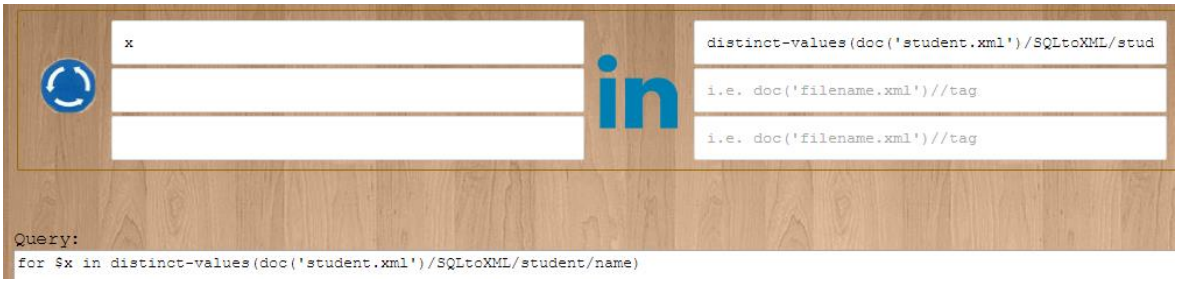


Εικόνα 92: Αποτελέσματα

Παρόμοιο τρόπο λειτουργείας έχει η χρήση του metaphor min και max.

Παράδειγμα 6 – Εύρεση των ονομάτων των φοιτητών του εργαστηρίου «OOP101», αφαίρεση διπλότυπων ονομάτων και ταξινόμηση τους αλφαβητικά

1. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101»*



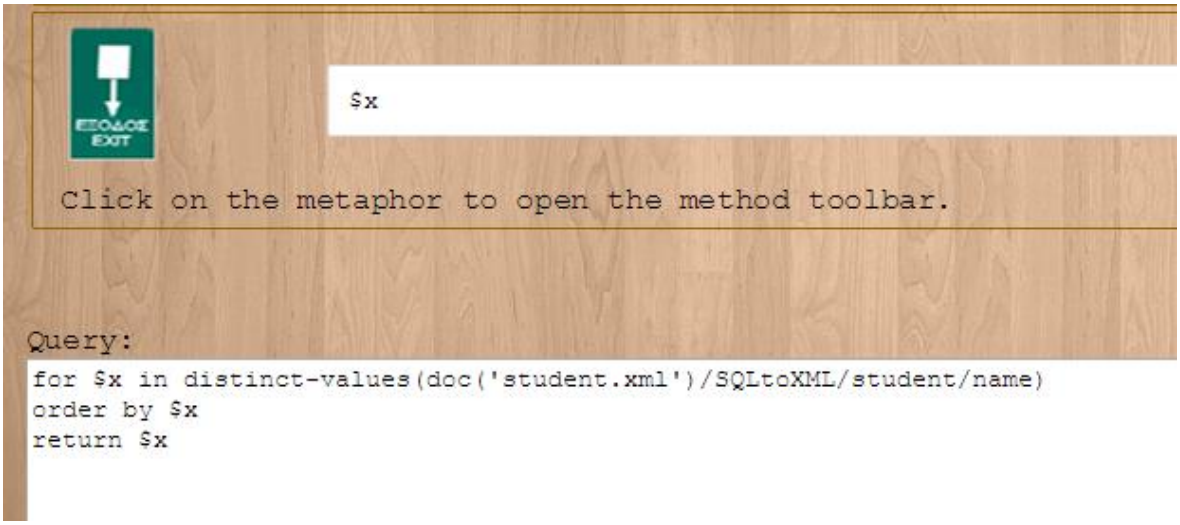
Εικόνα 93: Συγγραφή εντολής for για προσπέλαση του πίνακα

Σε αυτήν την εντολή χρησιμοποιείται η συνάρτηση distinct-values που αφαιρεί τις διπλότυπες τιμές όπως αναφέρθηκε στο κεφάλαιο «Συναρτήσεις».

2. *Εισαγωγή το metaphor Order By*

Εικόνα 94: Εισαγωγή metaphor Order By

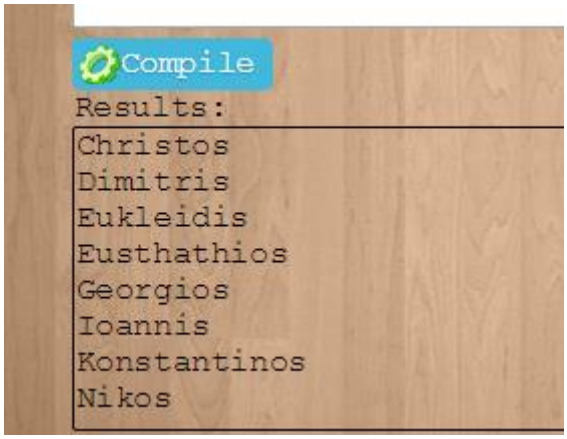
3. *Εισαγωγή εντολής return*



Εικόνα 95: Εισαγωγή εντολής return

4. *Πάτημα κουμπιών "compile" και "execute"*

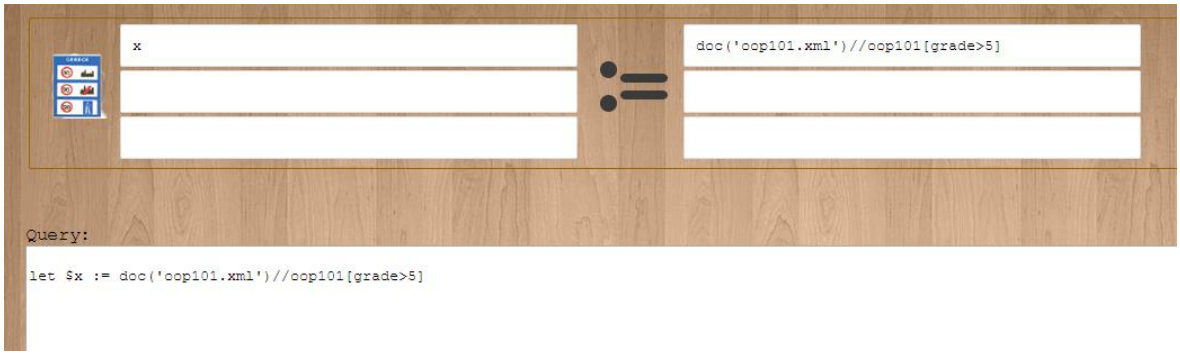
5. *Προβολή αποτελεσμάτων*



Εικόνα 96: Αποτελέσματα

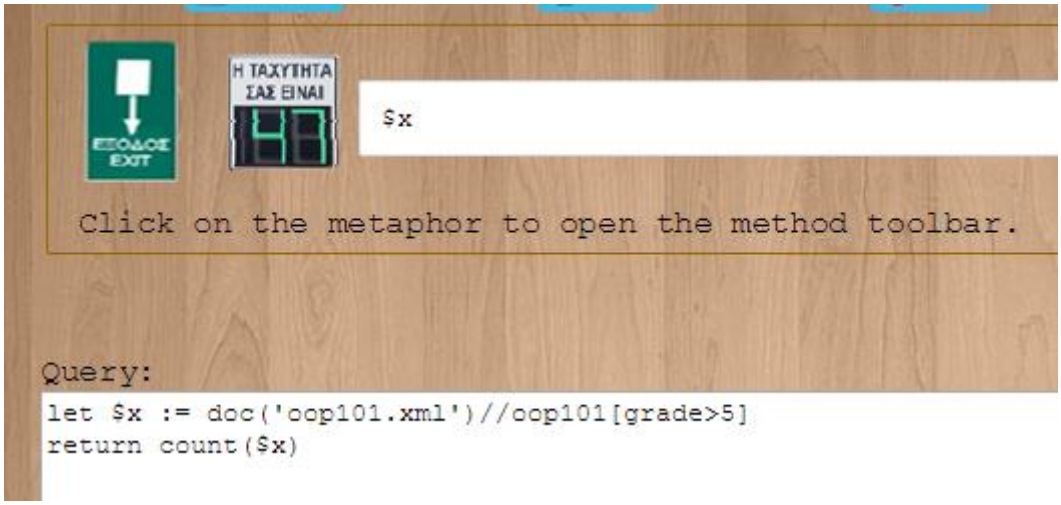
Παράδειγμα 7 – Εκτέλεση του τρίτου ερωτήματος με την χρήση του metaphor let

1. *Επιλογή της εντολής let μαζί με το κριτήριο για τον βαθμό να είναι μεγαλύτερος του πέντε*



Εικόνα 97: Εισαγωγή εντολής let μαζί με το κριτήριο

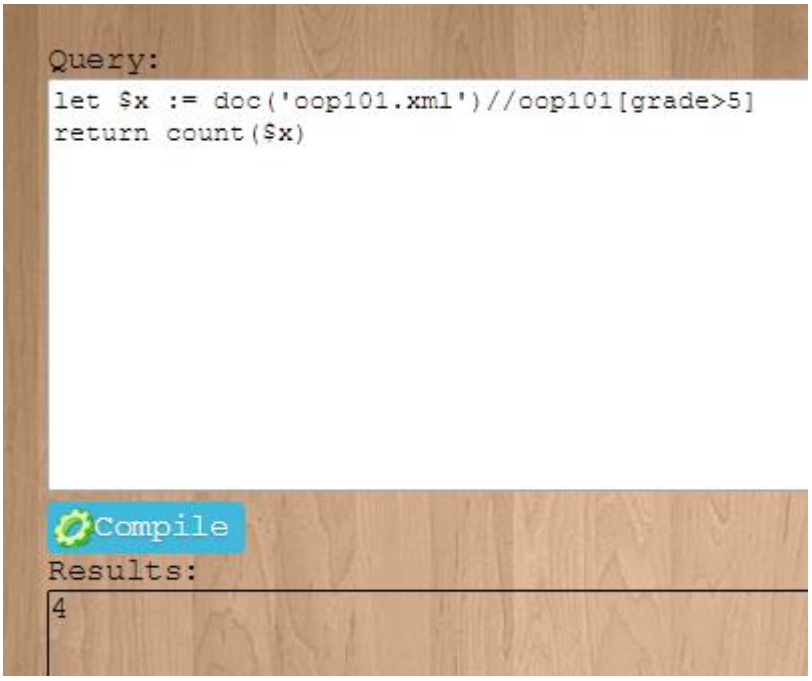
2. *Επιστροφή τιμών*



Εικόνα 98: Επιστροφή τιμών

3. *Πάτημα κουμπιών "compile" και "execute"*

4. *Προβολή αποτελεσμάτων*



Εικόνα 99: Αποτελέσματα

Όπως βλέπουμε το αποτέλεσμα είναι ίδιο με το παράδειγμα 3 χρησιμοποιώντας την εντολή let αντί της for.

Παράδειγμα 8 – Εύρεση του συνόλου των απουσιών που έκαναν οι φοιτητές του εργαστηρίου «OOP101»

1. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101»*

Εικόνα 100: Εντολή for

2. *Συγγραφή εντολής return μαζί με το metaphor sum για άθροισμα όλων των απουσιών*

Εικόνα 101: Χρήση sum με την εντολή return

3. *Πάτημα κουμπιών "compile" και "execute"*
4. *Προβολή αποτελεσμάτων*

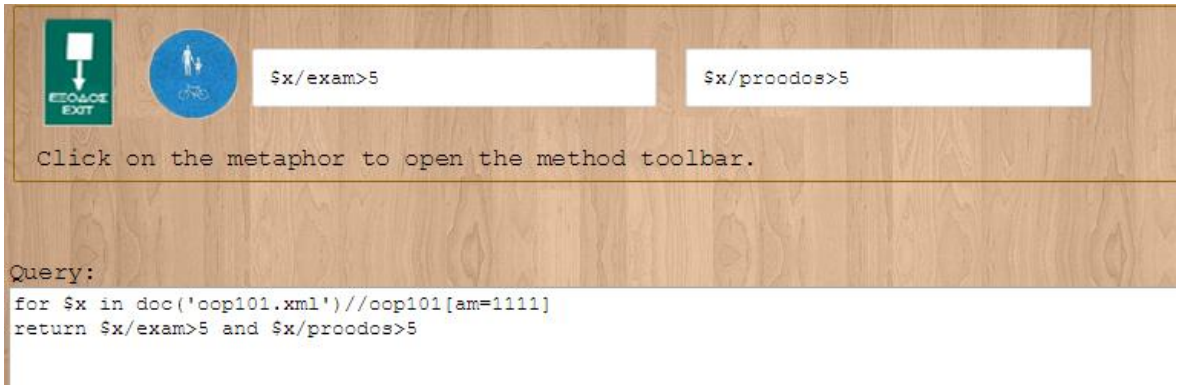
Εικόνα 102: Αποτελέσματα

Παράδειγμα 9 – Εύρεση αν ο φοιτητής με το ΑΜ 1111, έχει περάσει το μάθημα έχοντας και τους δυο βαθμούς (εξεταστικής και προόδου) μεγαλύτερους από πέντε κάνοντας χρήση του metaphor AND

1. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101» μαζί με τον περιορισμό για την εύρεση του συγκεκριμένου φοιτητή*

Εικόνα 103: Εντολή for μαζί με τους περιορισμούς

2. *Εισαγωγή της εντολής return μαζί με τους δυο ελέγχους*



Εικόνα 104: Εντολή return μαζί με το metaphor and

3. *Πάτημα κουμπιών "compile" και "execute"*

4. *Προβολή αποτελεσμάτων*

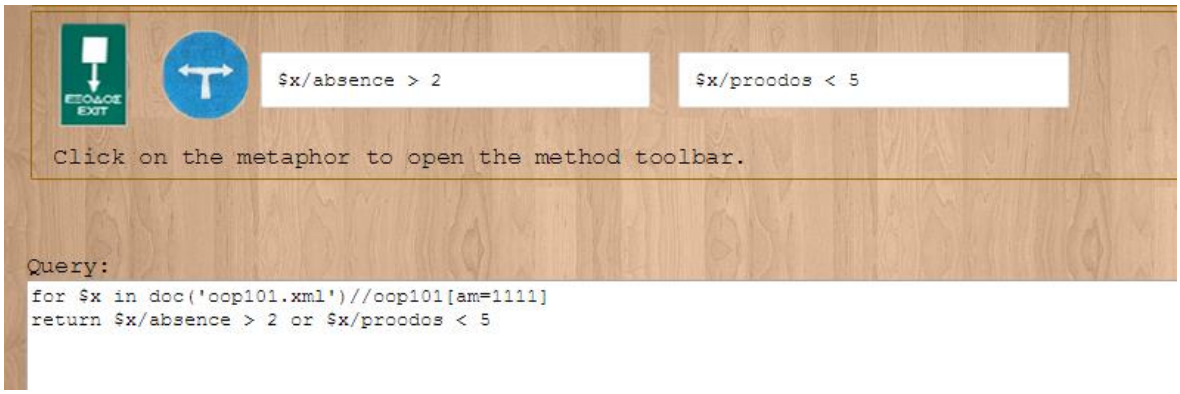
Εικόνα 105: Αποτελέσματα

Παράδειγμα 10 – Εύρεση αν ο φοιτητής με το ΑΜ 1111 έχασε το δικαίωμα συμμετοχής στην εξεταστική λόγω πολλών απουσιών ή μη-επαρκούς βαθμού στην πρόοδο

1. *Συγγραφή εντολής for για προσπέλαση του πίνακα του εργαστηρίου «oop101» μαζί με τον περιορισμό για την εύρεση του συγκεκριμένου φοιτητή*

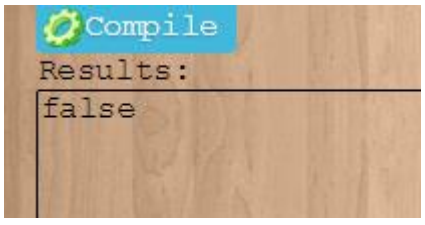
Εικόνα 106: Εντολή for μαζί με τους περιορισμούς

2. *Εισαγωγή εντολής return μαζί με το metaphor or για έλεγχο δικαιώματος πρόσβασης στην εξεταστική*



Εικόνα 107: Χρήση return με metaphor or

3. *Πάτημα κουμπιών "compile" και "execute"*
4. *Προβολή αποτελεσμάτων*



Εικόνα 108: Αποτελέσματα

4 Συμπεράσματα

Ο σκοπός της πτυχιακής εργασίας αυτής ήταν να βοηθήσουμε τον μέσο χρήστη να εκμεταλλευτεί τις ικανότητες της πληροφορικής στην διαχείριση των δεδομένων χωρίς κόπο και γνώσεις. Πιστεύουμε πως πραγματοποιήθηκε ως ένα βαθμό αλλά σίγουρα επιδέχεται αρκετές βελτιώσεις ακόμα. Σύντομα, στην προσπάθεια περαιτέρω βελτίωσης θα υπάρξει μια έρευνα με βοήθεια των χρηστών για το πόσο βοήθησε η εφαρμογή, αν αντιμετώπισαν δυσκολίες στην κατανόηση και στη χρήση της καθώς και αν υπάρχουν κάποιες ιδέες για βελτίωση. Τέλος, ένας από τους στόχους είναι η επέκταση της εφαρμογής για χρήση και από άτομα με ειδικές ανάγκες όπως π.χ. άτομα με μειωμένη όραση.

5 Βιβλιογραφία

Gehrke Johannes and Ramakrishnan Raghu, Database Management Systems, 3rd Edition 2011, pp 3-88

Priscilla Walmsley (April 2007), XQuery – Search across a variety of xml data, O'Reilly, United States of America.

Πλιάκας Αχιλλέας - Τσέκος Κωνσταντίνος, 2010. «Υλοποίηση γραφικής διεπαφής XQuery», τμήμα Μηχανικών Πληροφορικής Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης

---

[1] http://www.w3schools.com/xml/xml_whatis.asp

[2] http://www.w3.org/XML/

[3] http://searchsoa.techtarget.com/definition/XML

[4] http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231

[5] http://www.w3.org/TR/REC-xml/#sec-origin-goals

[6] http://www.w3.org/TR/REC-xml/

[7] http://www.w3.org/TR/1998/REC-xml-19980210

[8] http://www.w3schools.com/dtd/dtd_building.asp

[9] http://xmlwriter.net/xml_guide/xml_declaration.shtml

[10] http://aetos.it.teithe.gr/~iliou/cs4804/dialexeis/tmp/4.%20XML_Intro_I.pdf slide:11

[11] http://www.w3.org/TR/REC-xml/#sec-well-formed

[12] http://xml.silmaril.ie/validity.html

[13] http://www.w3schools.com/xml/xml_doctypes.asp

[14] http://www.mulberrytech.com/papers/HowAndWhyXML/slide009.html

[15] http://c2.com/cgi/wiki?BenefitsOfXml

[16] http://www.techmynd.com/advantages-disadvantages-of-xml/

[17] http://www.w3schools.com/xquery/

[18] http://www.w3.org/TR/xquery/#id-introduction

[19] http://www.w3.org/TR/xquery-30/

[20] http://en.wikipedia.org/wiki/XQuery#Applications

[21] http://www.w3schools.com/xpath/default.asp

[22] http://en.wikipedia.org/wiki/XQuery#Features

[23] http://grtjn.blogspot.nl/2011/10/xquery-novelties-revisited.html

[24] http://www.w3schools.com/xsl/

[25] http://en.wikibooks.org/wiki/XQuery/Benefits

[26] http://nativexmldatabase.com/2008/06/26/xquery-versus-sqlxml/

[27] http://searchsoa.techtarget.com/definition/XSD

[28] https://www.safaribooksonline.com/library/view/xquery/0596006349/ch13s02.html

[29] http://www.w3.org/2005/xqt-errors/

[30] http://www.w3schools.com/xquery/xquery_terms.asp

[31] http://en.wikibooks.org/wiki/XQuery/Sequences

[32] https://www.safaribooksonline.com/library/view/xquery/0596006349/ch02s04.html

[33] http://docs.oracle.com/cd/E13214_01/wli/docs92/xref/xqdtypes.html

[34] http://www.w3.org/TR/xpath-datamodel/

[35] http://www.w3schools.com/xquery/xquery_syntax.asp

[36] https://www.safaribooksonline.com/library/view/xquery/0596006349/ch03s10.html

[37] http://www.w3.org/TR/2002/WD-xquery-operators-20020816/

[38] https://www.progress.com/products/data-integration-suite/data-integration-suite-developer-center/data-integration-suite-tutorials/learning-xquery/xquery---a-guided-tour/xquery-operators

[39] http://www.w3schools.com/xpath/

[40] http://www.ibm.com/developerworks/library/x-xqueryxpath/

[41] http://www.w3schools.com/xquery/xquery_flwor.asp

[42] http://www.stylusstudio.com/xquery_flwor.html

[43] http://www.cs.princeton.edu/courses/archive/fall08/cos597A/Notes/xml_example.pdf

[44] http://www.w3schools.com/xquery/xquery_functions.asp

[45] http://www.w3schools.com/xpath/xpath_functions.asp

[46] http://www.xqueryfunctions.com/xq/c0008.html

[47] http://www.webopedia.com/TERM/D/database_management_system_DBMS.html

[48] http://www.britannica.com/EBchecked/topic/152201/database-management-system-DBMS

[49] http://searchsqlserver.techtarget.com/definition/database-management-system

[50] http://ecomputernotes.com/fundamental/what-is-a-database/advantages-and-disadvantages-of-dbms

[51] http://www.gitta.info/IntroToDBS/en/html/AdvantDisadv_learningObject2.html

[52] http://www.bcanotes.com/Download/DBMS/Rdbms/Advantages_&_Disadvantages%20of%20DBMS.pdf page:10-12

[53] http://www.gitta.info/IntroToDBS/en/html/AdvantDisadv_learningObject3.html

[54] http://en.wikipedia.org/wiki/Database#Storage

[55] http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf

[56] http://en.wikipedia.org/wiki/SQL#Criticism

[57] http://www.tutorialspoint.com/sql/sql-useful-functions.htm

[58] http://www.w3schools.com/sql/sql_functions.asp

[59] http://www.postgresql.org/docs/9.1/static/datatype-character.html

[60] http://www.sedna.org/

[61] https://netbeans.org/community/releases/80/

[62] http://en.wikipedia.org/wiki/NetBeans

[63] http://www.postgresql.org/about/advantages/

[64] http://www.postgresql.org/about/awards/

[65] http://en.wikipedia.org/wiki/PostgreSQL