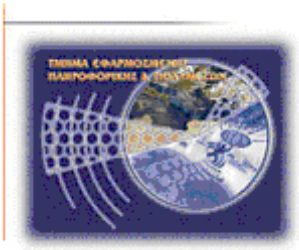
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία με θέμα

Δημιουργία interface σε φυσική γλώσσα (αγγλικά) για την γλώσσα επερωτήσεων SQL

του σπουδαστή: Σιρανίδη Κωνσταντίνου (ΑΜ: 2551)

Επιβλέπων καθηγητής : Νικόλας Παπαδάκης

Ηράκλειο, Δεκέμβριος 2013

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους γονείς μου για τη στήριξη και τη συμπαράσταση που μου παρείχαν κατά τη διάρκεια υλοποίησης αυτής της πτυχιακής. Τους φίλους μου, για την υποστήριξη και τη γνώμη τους, οποιαδήποτε στιγμή και αν τη ζητούσα, καθώς και τον εισηγητή μου Κύριο Νικόλαο Παπαδάκη για τη βοήθεια που μου παρείχε.

Abstract

In this thesis we created a web application which is capable, with defined grammar and syntax to transform simple queries in natural language, written by user, to equivalent SQL queries. Even thought SQL language is widely used it is very difficult for a typical user to use it due to syntax restricted format. The purpose of this application is to transform simple queries in natural language close to sql syntax queries. This is a result of a language grammar and syntax equal to sql language syntax query system. The application with some modification its can be easily expandable and can support multiply language as input.

Keywords: Natural language, sql, queries, semantics

Περίληψη

Σε αυτή την πτυχιακή αναπτύχθηκε μια εφαρμογή που είναι ικανή με βάση μια συγκεκριμένη γραμματική και συντακτικό που έχουμε ορίσει, να μετατρέψει μια πρόταση διατυπωμένη σε φυσική γλώσσα (αγγλικά) που δίνεται από τον χρήστη, σε ένα sql ερώτημα. Παρόλο που η γλώσσα sql χρησιμοποιείται ευρέως, η χρήση της από έναν απλό χρήστη δεν είναι κάτι εύκολο εάν δεν γνωρίζει ακριβώς την σύνταξή της. Ο σκοπός αυτής της εφαρμογής είναι να μετατρέψει απλά ερωτήματα διατυπωμένα σε φυσική γλώσσα στα αντίστοιχα sql ερωτήματα. Αυτό γίνεται με την δημιουργία μιας γραμματικής που μοιάζει συντακτικά με την γλώσσα επερωτήσεων sql. Η εφαρμογή είναι ικανή να εξελιχθεί και να υποστηρίζει παραπάνω από μια φυσικές γλώσσες με τις κατάλληλες μετατροπές.

Λέξεις κλειδιά: Φυσική γλώσσα, βάσεις δεδομένων, ερωτήσεις, semantics

Πίνακας Περιεχομένων

Πίνακας εικόνων

Πίνακας πινάκων

Γλωσσάρι

Compiler	Μεταγλωττιστής
Lexical analysis	Διαδικασία λεκτικής ανάλυσης
Syntactic analysis	Διαδικασία συντακτικής ανάλυσης
Semantic analysis	Διαδικασία σημασιολογικής ανάλυσης
Source code	Πηγαίος κώδικας
Parsing	Διαδικασία συντακτικής ανάλυσης
Lexer	Λεξικός αναλυτής
Parser	Συντακτικός αναλυτής
Grammar	Γραμματική
Rules	Κανόνες γραμματικής
Lexers	Λεκτικοί κανόνες γραμματικής
Tokens	Λεκτικές Μονάδες
Top-down Parser	Συντακτικός αναλυτής από πάνω προς τα κάτω
Bottom-up Parser	Συντακτικός αναλυτής από κάτω προς τα πάνω
Operation semantics	Λειτουργική σημασιολογία
Denotation semantics	Δηλωτική σημασιολογία
Axiomatic semantics	Αξιωτική σημασιολογία
Web application	Διαδικτυακή εφαρμογή
Antlr	Γεννήτρια συντακτικής ανάλυσης
Antlrworks	Γραφικό περιβάλλον δημιουργίας γραμματικής
BNF grammar	Backus–Naur Form γραμματική
EBNF grammar	Extended Backus–Naur Form γραμματική
Java	Γλώσσα προγραμματισμού
MySQL	Σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων
Apache tomcat	Διαδικτυακός σέρβερ
Servlet	Γλώσσα προγραμματισμού java
JSP	JavaServer Pages
CSS	Cascading Style Sheets- αλληλουχία φύλλων στυλ
JDBC	Συνδετικότητα Βάσης Δεδομένων JAVA - Java Database Connectivity

1 Εισαγωγή

1.1 Περίληψη

Σκοπός της πτυχιακής αυτής εργασίας είναι η μελέτη, η σχεδίαση και η ανάπτυξη μιας διεπαφής χρήστη όπου θα μπορεί οποιοσδήποτε χρήστης χωρίς απαραίτητες γνώσεις γλωσσών προγραμματισμού ή βάσεων δεδομένων, με κάποια ερωτήματα σε φυσική γλώσσα να εξάγει κάποια αποτελέσματα από μια βάση δεδομένων. Για το λόγο αυτό στα πλαίσια αυτής της πτυχιακής μελετήθηκε η φυσική γλώσσα, η γλώσσα ερωτήσεων SQL, η αρχιτεκτονική συστημάτων βάσεων δεδομένων, καθώς και η δημιουργία μιας γλώσσας που μοιάζει με το συντακτικό των ερωτήσεων SQL.

Για την υλοποίηση της διεπαφής μας, χρησιμοποιήθηκε η τεχνολογία Servlet και JSP, που παρέχεται από τη γλώσσα προγραμματισμού Java και εκτελείται στα περισσότερα σύγχρονα υπολογιστικά συστήματα. Για τη παρουσίαση της βάσης δεδομένων καθώς και τον σχεδιασμό των ερωτημάτων χρησιμοποιήθηκε το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων MySQL. Για το τελικό αποτέλεσμα της διεπαφής του χρήστη χρησιμοποιήθηκαν σύγχρονα εργαλεία όπως HTML, JSP, CSS, SQL, JDBC, ANTLR.

1.2 Κίνητρο για τη διεξαγωγή της εργασίας

Το σύστημα ερωταπαντήσεων σε φυσική γλώσσα φιλοδοξεί να γίνει η επόμενη γενιά στην επικοινωνία ανάμεσα στους χρήστες και στον υπολογιστή. Επιτρέπει στους χρήστες να αλληλεπιδρούν με βάσεις δεδομένων χωρίς να έχουν κάποιες εξειδικευμένες γνώσεις πάνω στις γλώσσες επερωτήσεων. Ο σκοπός αυτής της πτυχιακής είναι να παρέχει μια διασύνδεση σε φυσική γλώσσα για τη γλώσσα ερωτήσεων SQL.Επειδή η ανάκτηση πληροφοριών από σχεσιακές βάσεις δεδομένων απαιτεί ορισμένες γνώσεις κάποιων ειδικών γλωσσών προγραμματισμού όπως SQL, υπάρχει προφανής ανάγκη να δημιουργηθεί ένα σύστημα που θα επιτρέπει στον χρήστη να δημιουργεί ερωτήματα, όπως αυτά στη καθημερινή του ζωή, διατυπωμένα σε φυσική γλώσσα.

Πρέπει να επισημανθεί ότι η ανάγκη αυτή σημειώθηκε πολύ πριν στη δεκαετία του 1960.Το ISNLIS (Woods, 1973) (LUNAR SCIENCE NATURAL LANGUAGE INFORMATION SYSTEM) ήταν ένα από τα πρώτα πειραματικά συστήματα ερωταποκρίσεων, το οποίο επέτρεπε σε γεωλόγους να έχουν πρόσβαση, να αναλύουν και να συγκρίνουν δεδομένα που προκύπτουν από την ανάλυση των πετρωμάτων που, οι αποστολές Apollo είχαν πάρει από το φεγγάρι. Είναι προφανές ότι η ανάγκη να μάθουν η γεωλόγοι μια κατάλληλη γλώσσα προγραμματισμού για να εξάγουν τα

αποτελέσματα που ήθελαν θα είχε σοβαρές συνέπειες τόσο από άποψη κόστους όσο και χρόνου. Με αυτό το σκεπτικό σχεδιάστηκε και αναπτύχθηκε ένα εργαλείο υποβολής ερωτημάτων που συλλέγει και συναθροίζει δεδομένα

1.3 Σκοπός και στόχος εργασίας

Στόχος λοιπόν της συγκεκριμένης εργασίας σύμφωνα με όσα έχουν αναλυθεί στις προηγούμενες παραγράφους είναι η δημιουργία ενός συστήματος που θα λάβει σαν είσοδο ερωτήματα χρηστών σε μια βάση δεδομένων εκφρασμένα σε φυσική γλώσσα, και θα παράγεται ως αποτέλεσμα το αντίστοιχο ερώτημα σε SQL γλώσσα. Για την επίτευξη του σκοπού αυτού έχουμε δημιουργήσει μια γραμματική που βασίζεται στο συντακτικό της αγγλικής γλώσσας και η μέθοδος επεξεργασίας των ερωτημάτων γίνεται σε γλώσσα SQL.Ουσιαστικά έχουμε δημιουργήσει μια γλώσσα επερωτήσεων πολύ κοντά στη φυσική γλώσσα, η οποία δίνει την ικανότητα σε μη ειδικευμένους χρήστες να αναζητούν μια βάση δεδομένων.

Για την κατασκευή του συστήματος χρησιμοποιούμε λεξική και συντακτική ανάλυση των τεχνικών της φυσικής γλώσσας για την τελική επεξεργασία και παραγωγή του ερωτήματος που εισάγει ο χρήστης. Επίσης χρησιμοποιείται το εργαλείο ANTLR και ANTLRworks (τα οποία αναλύονται στα επόμενα κεφάλαια) για τη δημιουργία της γραμματικής, η γλώσσα java για την επεξεργασία και προσθήκη επιπλέον κώδικα στη γραμματική μας καθώς και οι τεχνολογίες της java, servlets και jsp για την τελική διεπαφή του χρήστη. Τέλος, γίνεται χρήση η τεχνολογία MySQL για την προβολή και εξαγωγή αποτελεσμάτων από τη βάση δεδομένων μας.

1.4 Δομή εργασίας

Η αναφορά αυτή αποτελείται από 6 κεφάλαια. Στο 1^{ο} κεφάλαιο έχουμε την περίληψη και τους στόχους της πτυχιακής αυτής. Στο 2^{ο} κεφάλαιο γίνεται μια θεωρητική ανάλυση της διαδικασίας της μεταγλώττισης με αναλυτικότερα τα στάδια της λεκτικής και συντακτικής ανάλυσης που είναι απαραίτητα γι αυτήν την πτυχιακή. Στο 3^{ο} κεφάλαιο γίνεται μια αναφορά στις τεχνολογίες υλοποίησης της πτυχιακής και στη συνεχεία παρουσιάζονται αναλυτικά όλες αυτές η τεχνολογίες. Στο 4^{ο} κεφάλαιο περιγράφεται η εφαρμογή σε θεωρητικό επίπεδο, κυρίως η ανάλυση της γραμματικής και πως αυτή χρησιμοποιείται, καθώς και η αρχιτεκτονική του συστήματος. Στο 5^{ο} κεφάλαιο παρουσιάζεται αναλυτικά και γραφικά η δημιουργία και το αποτέλεσμα της εφαρμογής και τέλος στο 6^{ο} κεφάλαιο παρουσιάζονται τα αποτελέσματα αυτής της πτυχιακής καθώς και οι μελλοντικές της επεκτάσεις.

2 Βασικές Αρχές Μεταγλωττιστών

Στο κεφάλαιο αυτό γίνεται μια παρουσίαση στις βασικές αρχές των μεταγλωττιστών, τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο. Προηγείται μια μικρή εισαγωγή στις βασικές έννοιες των γλωσσών προγραμματισμού και στη συνέχεια παρουσιάζεται το εργαλείο ANTLR, το οποίο είναι το βασικότερο εργαλείο που επιλέχθηκε για να χρησιμοποιηθεί στην εργασία αυτή για την κατασκευή του αρχείου της γραμματικής.

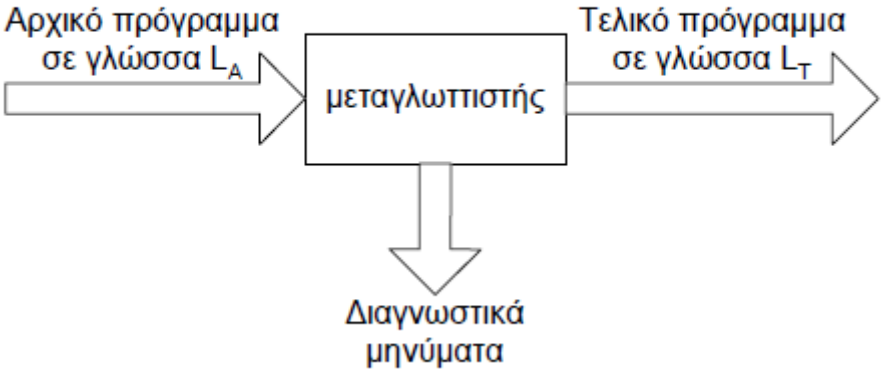
2.1 Γλώσσες Προγραμματισμού

Οι γλώσσες προγραμματισμού(programming languages) είναι συμβολισμοί που χρησιμοποιούνται για την περιγραφή υπολογισμών. Με τον όρο υπολογισμός, εννοούμε ακολουθίες βημάτων που μπορεί να εκτελέσει μια μηχανή για την επίλυση ενός προβλήματος. Μια συμβολική περιγραφή ενός υπολογισμού σε κάποια γλώσσα προγραμματισμού ονομάζεται πρόγραμμα. Ο προγραμματισμός στη γλώσσα μηχανής ενός υπολογιστή είναι ιδιαίτερα επίπονος και επιρρεπής σε σφάλματα, καθώς οι γλώσσες αυτές είναι συνήθως εξαιρετικά χαμηλού επιπέδου(low-level) και διαφέρουν κατά πολύ από τον τρόπο σκέψης του ανθρώπου. Για το λόγο αυτό, σχεδιάστηκαν και χρησιμοποιούνται οι γλώσσες προγραμματισμού υψηλού επιπέδου(high-level) με τις οποίες ο προγραμματισμός των υπολογιστών γίνεται πιο προσιτός στον άνθρωπο. Παραδείγματα διαδεδομένων γλωσσών προγραμματισμού υψηλού επιπέδου είναι:

- Pascal
- Fortran
- C
- C++
- Java
- Python
- Ruby
- Perl

Μια γλώσσα προγραμματισμού πρέπει να είναι κατανοητή από τον προγραμματιστή ώστε να μπορεί αυτός να υλοποιήσει προγράμματα με ευκολία και εκτελέσιμα από τον υπολογιστή. Προκειμένου ο υπολογιστής να εκτελέσει ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου, είναι απαραίτητη η ύπαρξη μιας υλοποίησης (implementation) της γλώσσας, η οποία αναλαμβάνει το ρόλο του μεταφραστή προς τον υπολογιστή. Οι υλοποιήσεις των γλωσσών προγραμματισμού κατατάσσονται σε δυο κύριες κατηγορίες: τους μεταγλωττιστές (compilers) και τους διερμηνείς (interpreters).

2.2 Μεταγλωττιστές



Εικόνα 1: Δομή μεταγλωττιστή

Ο μεταγλωττιστής (compiler) είναι ένα πρόγραμμα το οποίο δέχεται ως είσοδο το κείμενο ενός προγράμματος γραμμένο σε μια γλώσσα La, και παράγει ως έξοδο ένα ισοδύναμο πρόγραμμα γραμμένο σε μια γλώσσα Lt. Δύο προγράμματα θεωρούνται ισοδύναμα, αν κατά την εκτέλεση τους για τα ίδια δεδομένα παράγουν ακριβώς τα ίδια αποτελέσματα. Η παραπάνω διαδικασία ονομάζεται μεταγλώττιση (compilation).Η γλώσσα La ονομάζεται αρχική γλώσσα (source language) ενώ η γλώσσα Lt ονομάζεται τελική γλώσσα(target language).Το πρόγραμμα που δίνεται ως είσοδος στον μεταγλωττιστή λέγεται αρχικό πρόγραμμα (source program) και το ισοδύναμο πρόγραμμα που παράγεται στην έξοδο ονομάζεται τελικό πρόγραμμα(target program).

2.3 Φάσεις μεταγλώττισης

Εικόνα 2: Φάσεις Μεταγλώττισης

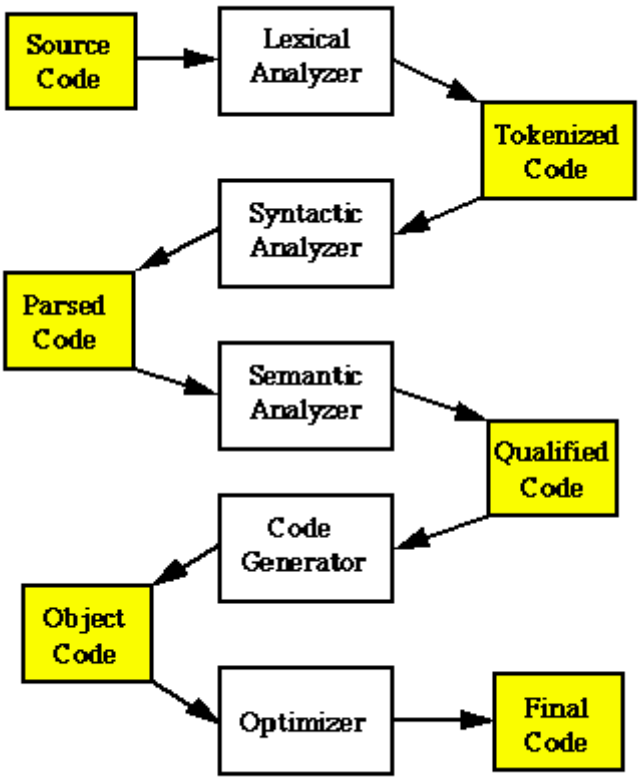
2.4 Δομή ενός Μεταγλωττιστή

Ένας μεταγλωττιστής αποτελεί τη γέφυρα μεταξύ προγραμμάτων πηγαίου κώδικα σε κάποια γλώσσα υψηλού επιπέδου και του υλικού. Πρέπει 1) να επαληθεύσει ότι τα προγράμματα έχουν σωστή σύνταξη, 2) να παράγει σωστό και γρήγορο αντικειμενικό κώδικα, 3) να οργανώσει το πώς εκτελείται το πρόγραμμα, και 4) να δώσει μορφή στην έξοδο που να είναι κατάλληλη για τον συμβολομεταφραστή ή τον συνδέτη. Ένας μεταγλωττιστής αποτελείται από τρία κύρια μέρη: το εμπρόσθιο, το ενδιάμεσο και το οπίσθιο τμήμα.

Το εμπρόσθιο τμήμα (front end) ελέγχει αν το πρόγραμμα είναι σωστά γραμμένο με βάση τη σύνταξη και τη σημασιολογία της γλώσσας προγραμματισμού. Σε αυτό το σημείο φαίνεται ποιο πρόγραμμα είναι έγκυρο και ποιο όχι, και εμφανίζονται μηνύματα σφάλματος. Εδώ επίσης ελέγχονται οι τύποι συλλέγοντας πληροφορία τύπων. Το εμπρόσθιο τμήμα παράγει επίσης μια ενδιάμεση αναπαράσταση (intermediate representation ή IR) του πηγαίου κώδικα, η οποία πρόκειται να δοθεί στο ενδιάμεσο τμήμα για επεξεργασία. Στο εμπρόσθιο τμήμα μπορούν να ανήκουν μεταξύ άλλων και τα εξής στάδια:

- ένα προεπεξεργαστή που αναλαμβάνει να επεξεργαστεί κάποιες ειδικές εντολές ή άλλα χαρακτηριστικά του πηγαίου κώδικα, ώστε να είναι σε κατάλληλη μορφή για τη μεταγλώττιση,
- ένα λεκτικό αναλυτή (lexer) που τεμαχίζει τον πηγαίο κώδικα σε λεκτικές μονάδες (tokens), ξεχωρίζοντας για παράδειγμα τις λέξεις-κλειδιά, τις εντολές της γλώσσας και τις τιμές του προγράμματος,

- ένα συντακτικό αναλυτή (parser) που συνθέτει τις λεκτικές μονάδες με βάση τη σύνταξη της γλώσσας, ώστε να προκύψει μια αφηρημένη μορφή του προγράμματος (συντακτικό δέντρο), κατάλληλη για περαιτέρω επεξεργασία.



Εικόνα 3: Διάγραμμα μεταγλώττισης

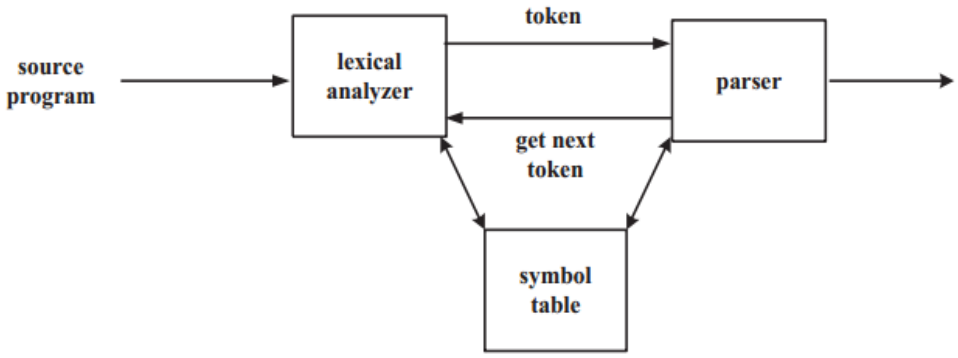
Τα τμήματα του λεκτικού αναλυτή και του συντακτικού αναλυτή είναι καθιερωμένο να υλοποιούνται με ειδικά εργαλεία για αυτό το σκοπό, τις γεννήτριες λεκτικών και συντακτικών αναλυτών. Στις γεννήτριες της πρώτης κατηγορίας, όπως το Lex, ο προγραμματιστής του μεταγλωττιστή ορίζει τις λεκτικές μονάδες που μπορεί να συναντηθούν στον πηγαίο κώδικα (όπως οι δεσμευμένες λέξεις και τα αλφαριθμητικά) και η γεννήτρια αναλαμβάνει να παράγει το αντίστοιχο τμήμα του μεταγλωττιστή. Αντίστοιχα, στις γεννήτριες της δεύτερης κατηγορίας, όπως το Yacc, ο προγραμματιστής ορίζει τη γραμματική της πηγαίας γλώσσας σε μια κατάλληλη μορφή (όπως η μορφή Μπάκους-Νάουρ) και στη συνέχεια παράγεται ο συντακτικός αναλυτής που διαβάζει αυτή τη γραμματική.

Στο ενδιάμεσο τμήμα (middle end) γίνονται οι βελτιστοποιήσεις. Συνηθισμένοι μετασχηματισμοί βελτιστοποίησης είναι η αφαίρεση άχρηστου ή απρόσιτου κώδικα, ο εντοπισμός και η διάδοση των σταθερών (constant propagation), η μεταφορά υπολογισμών εκτός συχνά χρησιμοποιούμενων τμημάτων (για παράδειγμα, μετακίνηση έξω από μια δομή επανάληψης), ή η εξειδίκευση ενός

υπολογισμού ανάλογα με τον κώδικα που τον περιβάλλει. Το ενδιάμεσο τμήμα παράγει στη συνέχεια μια άλλη ενδιάμεση αναπαράσταση, για το οπίσθιο τμήμα. Οι περισσότερες βελτιστοποιήσεις έχουν ήδη γίνει στο ενδιάμεσο τμήμα.

Το οπίσθιο τμήμα (back end) είναι υπεύθυνο για τη μετάφραση της ενδιάμεσης αναπαράστασης του ενδιάμεσου τμήματος σε γλώσσα μηχανής, συμβολική γλώσσα, γλώσσα προγραμματισμού (όπως η C) ή κώδικα για κάποια αφηρημένη μηχανή (abstract machine) όπως ο κώδικας byte (bytecode). Κάθε εντολή της ενδιάμεσης αναπαράστασης αντιστοιχεί σε κάποιες συμβολικές εντολές. Η κατανομή καταχωρητών αντιστοιχεί καταχωρητές στις μεταβλητές του προγράμματος. Το οπίσθιο τμήμα χρησιμοποιεί το υλικό με τέτοιο τρόπο ώστε να χρησιμοποιούνται όλες οι λειτουργικές μονάδες του υλικού με αποδοτικό τρόπο. Αν και οι περισσότεροι αλγόριθμοι βελτιστοποίησης για αυτά τα προβλήματα είναι πολυπλοκότητας NP, έχουν αναπτυχθεί και αρκετά προχωρημένες ευριστικές τεχνικές.

2.5 Λεκτική ανάλυση



Εικόνα 4: Λεκτική ανάλυση

Στη φάση της λεκτικής ανάλυσης(lexical analysis) ο μεταγλωττιστής δέχεται ως είσοδο ένα αρχικό πρόγραμμα με τη μορφή μιας συμβολοσειράς χαρακτήρων και δίνει ως έξοδο το ίδιο πρόγραμμα με τη μορφή μιας συμβολοσειράς λεκτικών μονάδων(tokens).Ουσιαστικά στη φάση αυτή γίνεται μια ομαδοποίηση των χαρακτήρων εισόδου σε λεκτικές μονάδες. Η φύση και η μορφή των λεκτικών μονάδων εξαρτώνται από την αρχική γλώσσα καθώς και από άλλους παράγοντες που σχετίζονται με τη σχεδίαση του μεταγλωττιστή. Οι λεκτικές μονάδες αποτελούν στην πραγματικότητα τα τερματικά σύμβολα της γραμματικής που περιγράφει τη σύνταξη της αρχικής γλώσσας προγραμματισμού στην επόμενη φάση της μεταγλώττισης, δηλαδή στην συντακτική ανάλυση. Έτσι μπορεί κανείς να θεωρήσει ότι η λεκτική ανάλυση είναι κατά ουσία μέρος της συντακτικής ανάλυσης. Ο λόγος που χρησιμοποιείται ξεχωριστά στη πράξη είναι διότι διευκολύνει τη σχεδίαση και την υλοποίηση του μεταγλωττιστή.

Το τμήμα του μεταγλωττιστή που υλοποιεί τη φάση της λεκτικής ανάλυσης λέγεται λεκτικός αναλυτής(lexical analyser, scanner).Εκτός από την εξαγωγή των λεκτικών μονάδων από το αρχικό πρόγραμμα ο λεκτικός αναλυτής είναι δυνατόν να κάνει και άλλες λειτουργίες, ανάλογα με τη φύση της αρχικής γλώσσας. Έτσι μπορεί να χρησιμοποιεί τον πίνακα συμβόλων για να αποθηκεύει ειδικές κατηγορίες λεκτικών μονάδων, όπως τα ονόματα και οι σταθερές, ή για να διαπιστώνει το είδος των λεκτικών μονάδων. Επίσης σε περίπτωση διαπίστωσης λεκτικών σφαλμάτων μπορεί να καλέσει τον διαχειριστή σφαλμάτων για την εκτύπωση κατάλληλων διαγνωστικών μηνυμάτων. Συνοψίζοντας ένας λεκτικός αναλυτής περιέχει τις παρακάτω λειτουργίες

- Διαχωρίζει το εισερχόμενο κείμενο σε λεκτικές μονάδες (tokens) και το μεταφέρει με τον τρόπο αυτό στο συντακτικό αναλυτή.
- Αποθηκεύει τα σύμβολα που διαβάζει σε πίνακα συμβόλων.
- Αποθηκεύει άλλα στοιχεία όπως τις συμβολοσειρές σε δυναμική μνήμη.
- Αναγνωρίζει λεκτικά λάθη στην είσοδο (π.χ. σύμβολα που δεν επιτρέπονται στη γλώσσα).
- Αφαιρεί τα σχόλια
- Συσχετίζει αριθμούς γραμμών με στοιχεία της εισόδου του.

Με τον τρόπο αυτό διαχωρίζονται οι εργασίες της λεκτικής και της συντακτικής ανάλυσης και κάθε μια υλοποιείται με τον πιο αποδοτικό τρόπο.

2.5.1 Tokenization

Tokenization ονομάζεται η διαδικασία με την οποία τεμαχίζεται μια συμβολοσειρά χαρακτήρων εισόδου σε ξεχωριστά μέρη, καθώς και η κατηγοριοποίηση αυτών των μερών. Οι λεκτικές μονάδες που προκύπτουν προωθούνται στη συνέχεια για περαιτέρω επεξεργασία. Η διαδικασία αυτή μπορεί να θεωρηθεί μέρος της συντακτικής ανάλυσης της εισόδου. Τυπικές λεκτικές μονάδες που ορίζονται είναι:

- Οι δεσμευμένες λέξεις (reserved words)
- Τα ονόματα (identifiers)
- Οι σταθερές (constants)
- Οι τελεστές (operators)
- Οι διαχωριστές (delimiters)
- Κάθε λεκτική μονάδα συσχετίζεται από το λεκτικό αναλυτή με τον τύπο της (π.χ. ακέραια σταθερά) και την τιμή της.
- Μη προκαθορισμένες λεκτικές μονάδες που δεν υποστηρίζονται άμεσα από τη γλώσσα υλοποίησης πρέπει να αποθηκευτούν σε πίνακα συμβόλων (για τα ονόματα) ή σε δυναμική μνήμη.
- Οι δεσμευμένες λέξεις τυπικά διαχωρίζονται μεταξύ τους κατά το στάδιο της λεκτικής ανάλυσης.

- Συχνά η αναγνώριση των δεσμευμένων λέξεων γίνεται με τη χρήση συναρτήσεων κατακερματισμού.

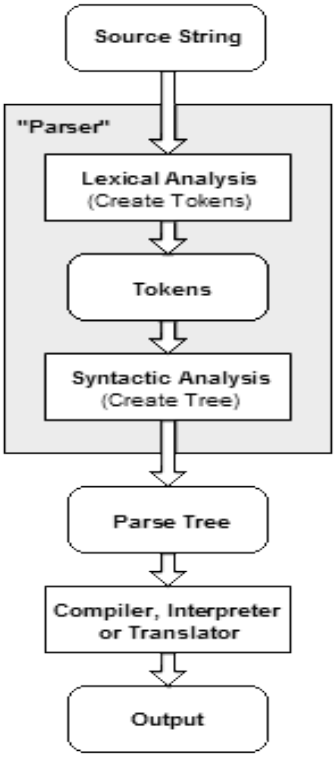
Έστω για παράδειγμα, η συμβολοσειρά:

The quick brown fox jumps over the lazy dog

Αν και ένας ομιλητής της Αγγλικής θα χώριζε την παραπάνω πρόταση με βάση τα κενά, αυτό δεν συμβαίνει γενικά κατά τη συντακτική ανάλυση. Η είσοδος αποτελείται από 43 χαρακτήρες και πρέπει ρητά να χωριστεί σε 9 λεκτικές μονάδες χρησιμοποιώντας σαν διαχωριστικό το κενό.

2.6 Συντακτική ανάλυση

Στη φάση της συντακτικής ανάλυσης (syntactic analysis, parsing) ελέγχεται αν το αρχικό πρόγραμμα ανήκει στη γλώσσα της οποίας η σύνταξη ορίζεται από μια δεδομένη γραμματική χωρίς συμφραζόμενα. Η ακολουθία λεκτικών μονάδων που προκύπτει από την λεκτική ανάλυση διαβάζεται στη φάση αυτή, και οι λεκτικές μονάδες ομαδοποιούνται σε συντακτικές μονάδες με βάση την παραπάνω γραμματική. Παράλληλα κατασκευάζεται το συντακτικό δέντρο (syntax tree) που απεικονίζει τη συντακτική δομή του αρχικού προγράμματος. Ανάλογα με τη σχεδίαση των επόμενων φάσεων της μεταγλώττισης είναι πιθανόν αυτό το δέντρο να αποτελεί βάση για την παραγωγή ενδιάμεσου κώδικα, οπότε και αποθηκεύεται για μεταγενέστερη χρήση. Σε αντίθετη περίπτωση, αν δηλαδή το δέντρο δεν είναι απαραίτητο για τη συνέχεια της μεταγλώττισης, κατασκευάζεται μόνο έμμεσα και δεν αποθηκεύεται. Το τμήμα του μεταγλωττιστή που κάνει αυτή τη δουλειά ονομάζεται συντακτικός αναλυτής(syntactic analyzer, parser).



Εικόνα 5: Συντακτική Ανάλυση / Parsing

Όπως αναφέρθηκε παραπάνω, η είσοδος ενός συντακτικού αναλυτή είναι το αρχικό πρόγραμμα με τη μορφή μιας ακολουθίας λεκτικών μονάδων, και η έξοδος του είναι το συντακτικό δέντρο που αντιστοιχεί σε αυτό το πρόγραμμα ή μια ένδειξη ότι το αρχικό πρόγραμμα δεν είναι συντακτικά ορθό. Στη πράξη όμως, ο συντακτικός αναλυτής συνεργάζεται στενά με τις επόμενες φάσεις της μεταγλώττισης και το συντακτικό δέντρο δεν είναι ορατό ως έξοδος αυτού του τμήματος.

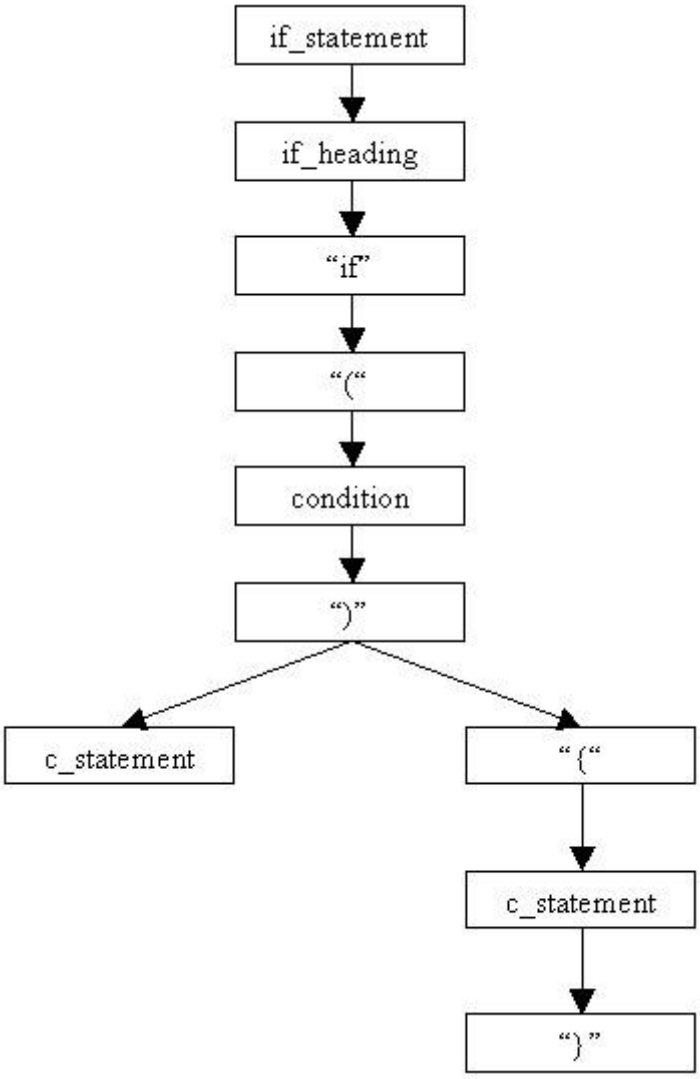
Συνοψίζοντας ο συντακτικός αναλυτής :

- υλοποιεί τη συντακτική ανάλυση μιας συγκεκριμένης γλώσσας,
- δέχεται ως είσοδο ένα πρόγραμμα με τη μορφή ακολουθίας λεκτικών μονάδων,
- ελέγχει αν το πρόγραμμα είναι σύμφωνο με τη γραμματική της γλώσσας που υλοποιεί (αν ανήκει στη συγκεκριμένη γλώσσα),
- σε περίπτωση συντακτικού λάθους ενημερώνει το χρήστη,
- παράγει το συντακτικό δένδρο που αντιστοιχεί στην ακολουθία εισόδου.

Διαχωρίζουμε δύο είδη συντακτικών αναλυτών ανάλογα με τον τρόπο που σχηματίζουν το δένδρο:

2.6.1 Συντακτικός αναλυτής από πάνω προς τα κάτω (top down parser)

Ο αναλυτής αυτός ξεκινά από τη ρίζα του δένδρου και αντικαθιστά μη τερματικά σύμβολα από το αριστερό τμήμα παραγωγών με τα αντίστοιχα δεξιά τους τμήματα μέχρι να αναγνωρίσει όλη την είσοδο. Το μη τερματικό σύμβολο που επιλέγεται για αντικατάσταση είναι κάθε φορά το αριστερότερο (leftmost) (L) μη τερματικό σύμβολο. Ακόμα, ο αναλυτής για να μπορέσει να υλοποιηθεί πρέπει να αρκεί να διαβάζει κάθε φορά ένα (1) μόνο σύμβολο από την είσοδο από αριστερά προς τα δεξιά (L). Έτσι ο αναλυτής και η αντίστοιχη γραμματική ονομάζονται LL(1).

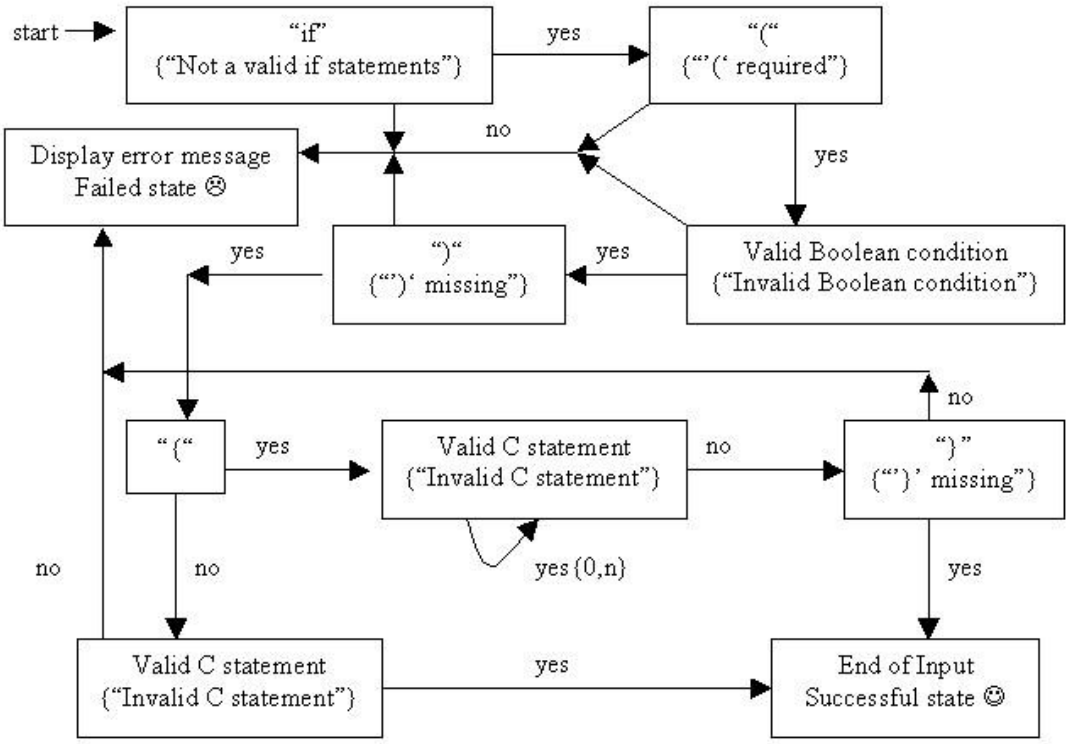


Εικόνα 6: Top Down Parser

Ο αναλυτής αυτός μπορεί να υλοποιηθεί είτε ως ένας συντακτικός αναλυτής αναδρομικής κατάβασης (recursive descent parser) είτε ως ένα ειδικό αυτόματο στοίβας. Ο συντακτικός αναλυτής αναδρομικής κατάβασης αποτελείται από συναρτήσεις, μια για κάθε μη τερματικό σύμβολο, που καλούν η μια την άλλη. Το ρόλο της στοίβας παίζει η στοίβα κλήσεων των συναρτήσεων που υλοποιεί ο επεξεργαστής. Για το λόγο αυτό οι συντακτικοί αυτοί αναλυτές είναι συχνά γρηγορότεροι από αυτούς που υλοποιούνται με αυτόματα.

2.6.2 Συντακτικός αναλυτής από κάτω προς τα πάνω (bottom up parser)

Ο αναλυτής αυτός αντικαθιστά δεξιά τμήματα παραγωγών με τα αντίστοιχα αριστερά τους τμήματα. Ο αναλυτής αυτός χρησιμοποιεί κάθε φορά τη δεξιότερη (rightmost) παραγωγή. Ανάλογα με το αν χρειάζεται να εξετάσει 0, 1 ή K σύμβολα εισόδου για να υλοποιήσει μια παραγωγή ονομάζεται LR(0), LR(1) ή LR(K). Η υλοποίηση του αναλυτή αυτού γίνεται από ειδικό αυτόματο στοίβας. Και στις δύο περιπτώσεις οι γραμματικές πρέπει να πληρούν ορισμένους κανόνες για να μπορούν να σχηματίσουν τον αντίστοιχο συντακτικό αναλυτή και κατηγοριοποιούνται ανάλογα με το όνομα του αναλυτή (LL(1), LR(1), ...).



Εικόνα 7: Bottom Up Parser

	Top-Down Parsers	Bottom-up parsers
Advantages	• Easy to code • Easy to debug • Smaller code • Can tokenise quickly	• Faster Handles left recursion
Disadvantages	• Slow Backtracking and recursion can be expensive. • Can not handle left recursion	• Fixed tokenisation • Hard to debug • Code can grow very large • Executing of actions can be unpredictable. • Tail recursion is handled poorly.

Πίνακας 1: Πλεονεκτήματα/Μειονεκτήματα Συντακτικών Αναλυτών

2.7 Σημασιολογική ανάλυση

Στη φάση της σημασιολογικής ανάλυσης (semantic analysis) ελέγχεται το πρόγραμμα για σημασιολογικά σφάλματα και συλλέγονται πληροφορίες που χρειάζονται για την παραγωγή ενδιάμεσου κώδικα και τη βελτιστοποίηση του. Το αποτέλεσμα αυτής της φάσης είναι και πάλι το συντακτικό δέντρο, πιθανώς τροποποιημένο και συμπληρωμένο με πληροφορίες σημασιολογικής φύσης.Η σημασιολογία διακρίνεται σε δυο κατηγορίες α) στη στατική και τη β)δυναμική σημασιολογία.

2.7.1 Στατική σημασιολογία

Ένα συντακτικά σωστό πρόγραμμα, που μπορεί να παραχθεί από μια γραμματική χωρίς συμφραζόμενα είναι πιθανό να περιέχει σφάλματα σημασιολογικής φύσης. Για παράδειγμα, ενώ οι περισσότερες γλώσσες προγραμματισμού απαγορεύουν τη δήλωση μιας μεταβλητής στην ίδια εμβέλεια περισσότερες από μια φορές, ο περιορισμός αυτός δεν μπορεί να επιβληθεί από μια γραμματική χωρίς συμφραζόμενα. Επίσης παρά το γεγονός ότι στις περισσότερες γλώσσες απαγορεύεται η πρόσθεση μιας συμβολοσειράς και ενός πίνακα ακέραιων αριθμών, η συντακτική ανάλυση ενός προγράμματος αγνοεί τους τύπους των μεταβλητών και κατά συνέπεια είναι αναγκασμένη να επιστρέψει κάθε έκφραση της μορφής a+b, ανεξαρτήτως των τύπων των a και b.

Για να εντοπισθούν σφάλματα όπως τα παραπάνω θα πρέπει να αποδοθεί κάποιου είδους ερμηνεία στη συντακτική δομή του προγράμματος. Αυτού του είδους ερμηνείες θα πρέπει να χρησιμοποιούνται συνέχεια προκειμένου να αποφεύγονται οι πολλαπλές δηλώσεις της ίδιας μεταβλητής μέσα σε μια εμβέλεια και να συσχετίζονται τα ονόματα μεταβλητών με τους τύπους που έχουν δηλωθεί για αυτές. Το τμήμα της σημασιολογικής περιγραφής μιας γλώσσας προγραμματισμού που ασχολείται με τέτοιου είδους θέματα απόδοσης και ερμηνείας ονομάζεται στατική σημασιολογία (static semantics).Δηλαδή κύριος σκοπός της στατικής σημασιολογίας είναι ο εντοπισμός σημασιολογικών σφαλμάτων κατά τη διάρκεια της μεταγλώττισης.

2.7.2 Δυναμική Σημασιολογία

Η απόδοση ερμηνείας στα προγράμματα μιας γλώσσας προγραμματισμού δε σταματά φυσικά στον έλεγχο τύπων και στον εντοπισμό σημασιολογικών σφαλμάτων. Εξίσου σημαντικό είναι να περιγραφεί η συμπεριφορά προγραμμάτων κατά την εκτέλεσης τους. Αυτό είναι το αντικείμενο της δυναμικής σημασιολογίας (dynamic semantics).Δηλαδή για ένα πρόγραμμα που είναι συντακτικά σωστό και δεν έχει στατικά σημασιολογικά σφάλματα η δυναμική σημασιολογία περιγράφει τι θα κάνει αυτό το πρόγραμμα όταν εκτελεστεί. Οι τυπικές περιγραφές δυναμικής σημασιολογίας κατατάσσονται σε τρεις μεγάλες κατηγορίες:

- Λειτουργική σημασιολογία (operation semantics), που περιγράφει την ερμηνεία ενός προγράμματος ως μια ακολουθία από τα υπολογιστικά βήματα που πραγματοποιούνται κατά την εκτέλεση.
- Δηλωτική σημασιολογία(denotational semantics), που περιγράφει την ερμηνεία ενός προγράμματος ως ένα μαθηματικό αντικείμενο, που συνήθως είναι μια συνάρτηση από το σύνολο των δεδομένων εισόδου του προγράμματος προς το σύνολο των αποτελεσμάτων του.
- Αξιωματική σημασιολογία(axiomatic semantics), που περιγράφει την ερμηνεία ενός προγράμματος με έμμεσο τρόπο, μέσω λογικών προτάσεων που περιγράφουν ιδιότητες αυτού.

2.8 Βελτιστοποίηση

Η βελτιστοποίηση έχει ως σκοπό να κάνει αποδοτικότερο το τελικό πρόγραμμα. Κύρια μέτρα απόδοσης είναι σε αυτή την περίπτωση η ταχύτητα εκτέλεσης του τελικού προγράμματος και το μέγεθος του. Η βελτιστοποίηση είναι φυσικά προαιρετική σε ένα μεταγλωττιστή και μπορεί να εφαρμοστεί στον ενδιάμεσο ή στον τελικό κώδικα και χωρίζεται σε δύο φάσεις η πρώτη βελτιστοποίηση μπορεί να γίνει στον ενδιάμεσο κώδικα και η δεύτερη στον τελικό κώδικα. Κάθε φάση

βελτιστοποίησης τροποποιεί τον κώδικα που δέχεται ως είσοδ, και παράγει στην έξοδο της τον ισοδύναμο κώδικα με βελτιωμένα χαρακτηριστικά απόδοσης.

2.9 Παραγωγή τελικού Κώδικα

Στη φάση αυτή μετατρέπεται ο ενδιάμεσος κώδικας σε τελικό κώδικα. Το αποτέλεσμα είναι το τελικό μεταγλωττισμένο μας πρόγραμμα.

3 Τεχνολογίες υλοποίησης πτυχιακής

Σε αυτό το κεφάλαιο θα παρουσιαστούν αναλυτικά οι βασικές τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση αυτής της πτυχιακής. Η web εφαρμογή που έχουμε δημιουργήσει βασίζεται στην τεχνολογία JSP και Servlets όπου ένα ερώτημα περνάει στο Servlet.Αυτό με τη σειρά του καλεί κάποιες java classes(lexer,parser) όπου σύμφωνα με τη γραμματική που έχουμε ορίσει παίρνουμε σαν αποτέλεσμα ένα SQL κώδικα. Στη συνέχεια εκτελείται το SQL ερώτημα προς μια βάση δεδομένων, όπου πάλι με τη χρήση Servlet και JDBC driver παίρνουμε σαν αποτέλεσμα τα δεδομένα του SQL ερωτήματος και εμφανίζονται στο χρήστη.

Στις επόμενες ενότητες θα αναλυθούν όλες οι παραπάνω τεχνολογίες καθώς και τα βήματα σε επίπεδο χρήστη που πρέπει να γίνουν ώστε να μπορέσει να τρέξει η εφαρμογή μας. Συνοπτικά στις επόμενες ενότητες θα δούμε:

- MySQL
- Apache Tomcat
- XAMPP
- JAVA
- NetBeans
- JSP
- Servlets
- JDBC
- CSS
- ANTLR
- ANTLRworks

3.1 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακής βάσης, ανοικτού κώδικα όπως λέγεται (relational database management system - RDBMS) που χρησιμοποιεί την Structured Query Language (SQL), την πιο γνωστή γλώσσα για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία Βάση Δεδομένων. Μια βάση δεδομένων σας επιτρέπει να αποθηκεύετε να αναζητάτε να ταξινομείτε και να

ανακαλείτε τα δεδομένα αποτελεσματικά. Επειδή είναι ανοικτού κώδικα (open source), οποιοσδήποτε μπορεί να κατεβάσει την MySQL και να την διαμορφώσει σύμφωνα με τις ανάγκες του σύμφωνα πάντα με την γενική άδεια που υπάρχει. Η MySQL είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία, και την ευελιξία που παρέχει. Οι περισσότεροι συμφωνούν ωστόσο ότι δουλεύει καλύτερα όταν διαχειρίζεται περιεχόμενο και όχι όταν εκτελεί συναλλαγές.

Η MySQL αυτή τη στιγμή μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows. Μια βάση δεδομένων (database) αποτελείται από έναν ή περισσότερους πίνακες (tables), ο καθένας από τους οποίους περιέχει μια λίστα από κάποια πράγματα. Για μια βάση δεδομένων πελατών (clients), είναι φυσικό να ξεκινήσουμε μ' έναν πίνακα με όνομα clients που θα περιέχει μια λίστα από στοιχεία πελατών.Κάθε πίνακας σε μια βάση δεδομένων περιέχει μια ή περισσότερες στήλες (columns) ή πεδία (fields), όπου η κάθε στήλη περιέχει μια συγκεκριμένη πληροφορία για τον κάθε πελάτη που υπάρχει στην βάση δεδομένων (database).Παράδειγμα χρήσης SQL κώδικα σε MySQL βάση δεδομένων:

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

Πίνακας 2: Παράδειγμα SQL κώδικα

3.2 Apache Tomcat

Το εργαλείο Apache Tomcat είναι μια ανοιχτού κώδικα υλοποίηση της τεχνολογίας των σελίδων διακομιστών Java (Servlet και JSP). Προσφέρει ώθηση στην ανάπτυξη κρίσιμων και μεγάλης κλίμακας δικτυακών εφαρμογών σε επιχειρήσεις και οργανισμούς. Όταν δημιουργούμε μια μεγάλη web εφαρμογή η οποία θα δέχεται requests ταυτόχρονα από διάφορους clients και όταν θα πρέπει να απαντάει σε όλους με ένα μεγάλο όγκο δεδομένων, κρίνεται αναγκαία η χρήση πολλών servers που θα εξυπηρετούν αυτά τα requests αλλά και κάποια λειτουργία που θα μοιράζει το βάρος (μνήμης, requests, responses, processes) κοινώς θα είναι ο balancer. Όλη την παραπάνω διαδικασία για μας λοιπόν, την κάνει ο Apache. Έτσι στήνουμε τον Apache και συνδέουμε με αυτόν τους διάφορους Tomcat που θα δημιουργήσουμε. Ο

Apache όμως, σε αντίθεση με τον Apache Tomcat παρέχει και κάποια εργαλεία τα οποία είναι απαραίτητα για την σωστή διαχείριση του server μας.

Με το εργαλείο apache tomcat παρέχεται η δυνατότητα:

- ρύθμισης του διαχειριστή ασφάλειας της Java
- χρησιμοποίησης πηγών δεδομένων μέσω JDBC
- υποστήριξης του πρωτοκόλλου SSL για κρυπτογραφημένες συνδέσεις
- χρήσης διαμεσολαβητών (proxy)
- εικονικής λειτουργίας (virtual hosting)
- Εκτέλεσης σεναρίων CGI.

3.3 XAMPP

Το XAMPP είναι ακρωνύμιο και αναφέρεται στα παρακάτω αρχικά:

- X (αναφέρεται στο «cross-platform» που σημαίνει λογισμικό ανεξάρτητο πλατφόρμας)
- Apache HTTP εξυπηρετητής
- MySQL
- PHP
- Perl

Το XAMPP είναι ένα ελεύθερο λογισμικό το οποίο περιέχει ένα εξυπηρετητή ιστοσελίδων το οποίο μπορεί να εξυπηρετεί και δυναμικές ιστοσελίδες τεχνολογίας PHP/MySQL. Είναι ανεξάρτητο πλατφόρμας και τρέχει σε Microsoft Windows, Linux, Solaris, and Mac OS X και χρησιμοποιείται ως πλατφόρμα για την σχεδίαση και ανάπτυξη ιστοσελίδων με την τεχνολογίες όπως PHP, JSP και Servlets. Στην πράξη το XAMPP ορισμένες φορές χρησιμοποιείται και για την φιλοξενία ιστοσελίδων.Υποστηρίζει την δημιουργία και διαχείριση βάσεων δεδομένων τύπου MySQL και SQLite. Όταν το XAMPP εγκατασταθεί στον τοπικό υπολογιστή διαχειρίζεται τον localhost ως ένα απομακρυσμένο κόμβο, ο οποίος συνδέεται με το πρωτόκολλο μεταφοράς αρχείων FTP.

3.4 Java

`

Η java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems Inc.. Ξεκίνησε σαν μέρος ενός μεγαλύτερου σχεδίου που αφορούσε την ανάπτυξη λογισμικού για καταναλωτικά ηλεκτρονικά προϊόντα που στην αρχή βασιζόντουσαν στην C++. Αρκετά προβλήματα όμως παρουσιάστηκαν και η γλώσσα C++ δεν μπόρεσε να εφαρμοστεί τελικά. Χρειάστηκε να αναπτυχθεί μία νέα γλώσσα: η Java. Η Java, στην τελική της μορφή, βρήκε περαιτέρω εφαρμογή στην επίλυση μερικών προβλημάτων του σημερινού προγραμματισμού, όπως animation, την αλληλεπίδραση πραγματικού χρόνου (real-time interaction) και την εξερεύνηση του Web (Web browsing).

Από την ημέρα της δημοσιοποίησης της τον Μάιο του 1995, η Java έχει εξαπλωθεί σε όλο το Internet. Η Java έχει κιόλας λύσει τα περισσότερα προβλήματα στο μοντέλο client/server και έχει προάγει την χρήση του World Wide Web. Η Java χαρακτηρίζεται από τα εξής: απλή, αντικειμενοστραφής, συμβατή με δικτυακά πρωτόκολλα, ουδέτερη της υποκείμενης αρχιτεκτονικής, φορητή ασφαλής, υψηλής απόδοσης, δυναμική, σταθερή, interpreted και multithreaded.

```
public class MyFirstJavaProgram {
   public static void main(String []args) {
      System.out.println("Hello World");
   }
}
```

Πίνακας 3: Παράδειγμα Κώδικα Java

3.4.1 NetBeans

Το NetBeans IDE είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης IDE - ένα εργαλείο στους προγραμματιστές για να γράψουν, να μεταγλωττίσουν, να αποσφαλματώσουν και να αναπτύξουν προγράμματα. Είναι γραμμένο σε Java, αλλά μπορεί να υποστηρίξει όλες τις γλώσσες προγραμματισμού. Υπάρχει επίσης ένας μεγάλος αριθμός υπομονάδων (modules) που βοηθάνε στην επέκταση της

λειτουργικότητας του NetBeans IDE. Το NetBeans IDE είναι ένα ελεύθερο προϊόν δίχως περιορισμούς στον τρόπο χρησιμοποίησής του. Το περιβάλλον ανάπτυξης γενικά παρέχει βοηθήματα στον προγραμματιστή τα οποία κάνουν ευκολότερη και αποδοτικότερη την ανάπτυξη των εφαρμογών. Τα βασικότερα από τα βοηθήματα αυτά αφορούν κυρίως τον Source Code Editor (Επισήμανση συντακτικού στον κώδικα, live επισήμανση λαθών και live parsing, pop up παράθυρα βοήθειας). Το εργαλείο NetBeans υποστηρίζει:

- Java SE, JavaFX
- Web & Java EE
- Java ME
- HTML και XHTML
- CSS
- Javascript
- PHP (Έκδοση 6.5 και μετά)
- C/C++
- Ruby

3.4.2 JSP

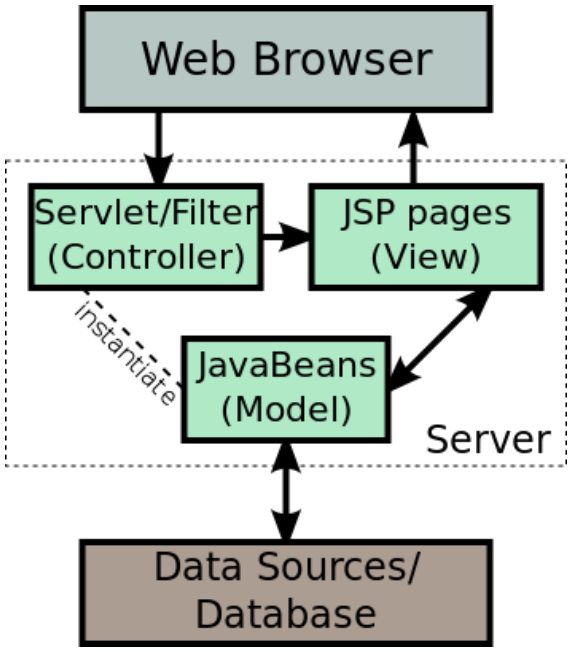
Οι JSPs (JavaServer Pages) είναι μια τεχνολογία που έχει δημιουργηθεί από την εταιρεία Sun Microsystems για να μπορεί να δημιουργεί δυναμικό περιεχόμενο (dynamic content) στο Web. Πρόκειται για HTML έγγραφα (ιστοσελίδες) τα οποία αναμειγνύονται με τη γλώσσα προγραμματισμού Java, η οποία και έχει τη δυνατότητα να παρέχει (δημιουργεί) αυτό το δυναμικό περιεχόμενο. Οι JSPs είναι μια εφαρμογή στην πλευρά του server (server-side application), που σημαίνει ότι δέχονται μια αίτηση (request) και παράγουν μια απόκριση ή απάντηση (response). Σε γενικές γραμμές, οι αιτήσεις γίνονται από έναν Web client και η απόκριση είναι ένα παραγόμενο HTML έγγραφο (ιστοσελίδα) το οποίο στέλνεται πίσω στον Web client.

Επειδή οι JSPs είναι μια εφαρμογή στην πλευρά του server, έχουν πρόσβαση σε πηγές (resources) στον server, όπως είναι τα Servlets, JavaBeans, EJBs, αλλά και σε βάσεις δεδομένων. Ένα άλλο σημαντικό πλεονέκτημα των JSPs είναι ο διαχωρισμός των ρόλων. Οι προδιαγραφές των JSPs επιτρέπουν να μοιραστεί το φορτίο σε δύο κατηγορίες : στο γραφικό περιεχόμενο της σελίδας και στο δυναμικό περιεχόμενο της σελίδας. Αυτό σημαίνει στην πράξη ότι η ομάδα που δεν γνωρίζει τη γλώσσα προγραμματισμού Java μπορεί να δημιουργήσει το γραφικό περιεχόμενο της

σελίδας και ένας προγραμματιστής της Java να δημιουργήσει το δυναμικό περιεχόμενο της σελίδας.

```
<HTML>
<BODY>
<%    System.out.println( "Evaluating date now" );
   java.util.Date date = new java.util.Date();
%>
Hello!  The  time  is  now  <%=  date  %>
</BODY>
</HTML>
```

Πίνακας 4: Παράδειγμα JSP κώδικα

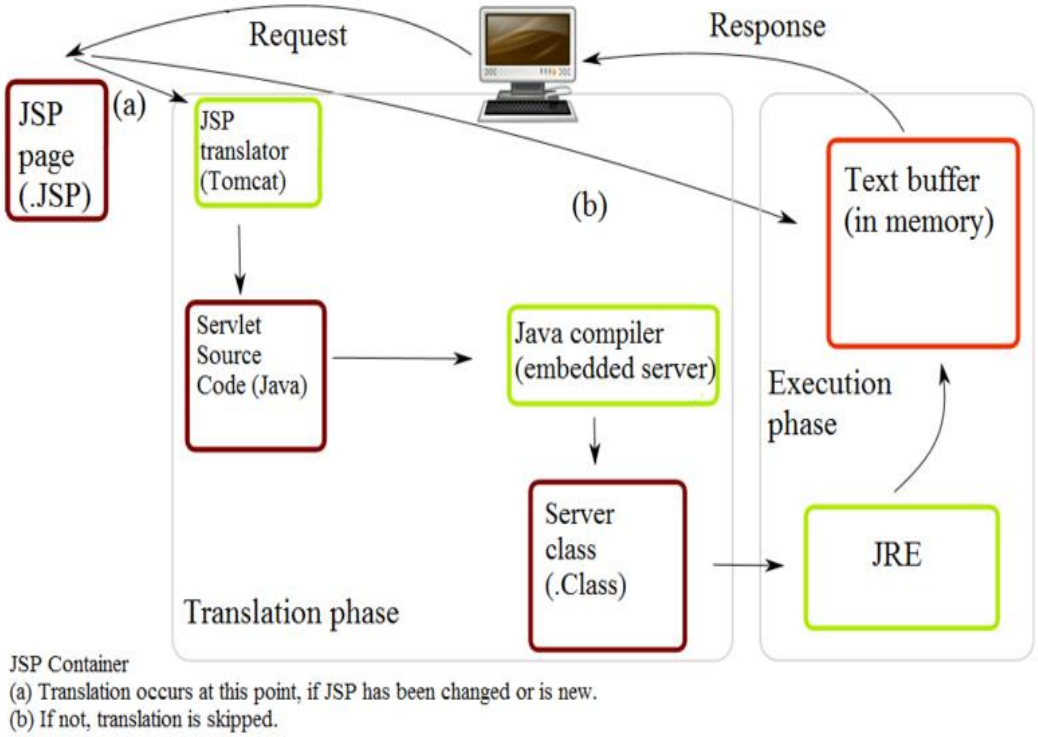


Εικόνα 8: Αρχιτεκτονική JSP

3.4.3 Servlets

Οι ιστοσελίδες που περιέχουν κώδικα JSP (JavaServer Pages), δηλαδή ανάμιξη κώδικα HTML με κώδικα Java, μετατρέπονται (μεταφράζονται ή μεταγλωττίζονται) σε Servlets πριν εκτελεσθούν στον server. Η κατανόηση αυτής της μετάφρασης (απόδοσης) σε κώδικα Servlet θα μας βοηθήσει να καταλάβουμε καλύτερα τις δραστηριότητες των JSPs στο παρασκήνιο. Ένα Servlet, στη γενική του μορφή, είναι μια τάξη (class) της Java που υλοποιεί (implements) το interface Servlet και δέχεται αιτήσεις (requests) και παράγει (δημιουργεί) αποκρίσεις (responses). Οι αιτήσεις μπορεί να προέρχονται από τάξεις της Java, από Web clients ή και από άλλα Servlets.Όταν υλοποιούμε ένα interface λέμε ότι η τάξη μάς παρέχει υλοποιήσεις για τις μεθόδους που είναι δηλωμένες στο interface.

Συνεπώς, όταν υλοποιούμε το interface Servlet δηλώνουμε ότι ο κώδικάς μας θα παρέχει υλοποιήσεις για τις μεθόδους που βρίσκονται στο interface Servlet. Θα ασχοληθούμε μ' ένα μόνο συγκεκριμένο είδος Servlet, το HttpServlet, το οποίο δέχεται HTTP requests και παράγει HTTP responses. Όταν γράφουμε το δικό μας HttpServlet, δεν υλοποιούμε το interface Servlet απευθείας, αλλά επεκτείνουμε (extend) την τάξη HttpServlet.



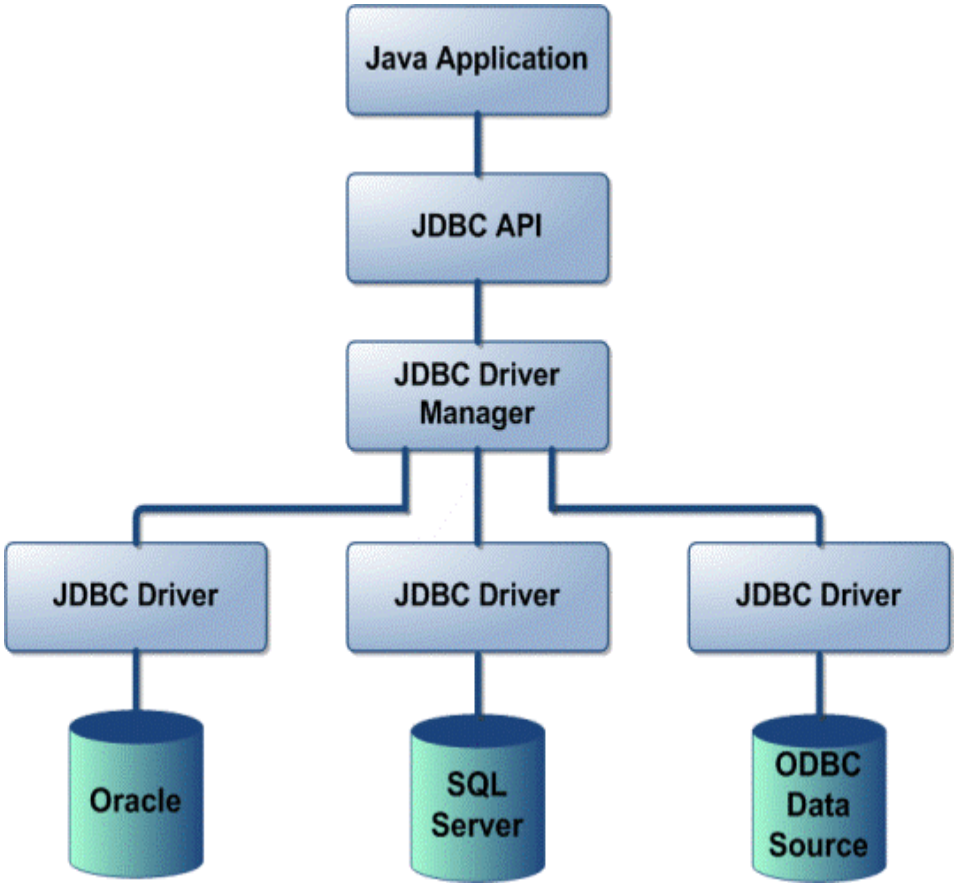
Εικόνα 9: Αρχιτεκτονική Java Servlet

3.4.4 JDBC

Η Συνδεσιμότητα Βάσης Δεδομένων JAVA (Java Database Connectivity, JDBC) είναι μία διεπαφή προγραμματισμού εφαρμογών (API) για την γλώσσα προγραμματισμού Java η οποία μας επιτρέπει να έχουμε πρόσβαση σε μία σχεσιακή βάση δεδομένων μέσα από το πρόγραμμά μας. Η τεχνολογία βασίζεται στην ύπαρξη οδηγών (drivers), οι οποίοι είναι αρμόδιοι για την επικοινωνία μεταξύ του προγράμματος και της βάσης. Με τη χρήση του κατάλληλου οδηγού συνδεόμαστε με τη βάση, στέλνουμε εντολές και λαμβάνουμε αποτελέσματα. Παρέχει συναρτήσεις για εξαγωγή, πρόσθεση, ανανέωση ή διαγραφή δεδομένων σε μια βάση. Οι

απαραίτητες κλάσεις ορίζονται από τη βιβλιοθήκη java.sql. Περιλαμβάνει συνοπτικά τα παρακάτω βήματα:

- Σύνδεση με τη βάση
- Φόρτωση οδηγού
- Δημιουργία σύνδεσης
- Δημιουργία εντολών με JDBC
- Ανάκτηση δεδομένων
- Αποσύνδεση



Εικόνα 10: Αρχιτεκτονική JDBC

3.5 CSS

Το CSS είναι μια απλή γλώσσα που μας βοηθάει να ορίσουμε με σαφήνεια και ιδιαίτερη ευελιξία τον τρόπο με τον οποίο θα εμφανίζονται τα διάφορα στοιχεία στην ιστοσελίδα μας. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη. Μερικά από τα πλεονεκτήματα που μας παρέχει το css είναι:

- Πολύ μεγαλύτερη ευελιξία. Το CSS κατέστησε εφικτές μορφοποιήσεις οι οποίες ήταν αδύνατες ή πολύ δύσκολες με την κλασσική HTML.
- Ευκολότερη συντήρηση των ιστοσελίδων. Η εμφάνιση ενός ολόκληρου site μπορεί να ελέγχεται από ένα μόνο εξωτερικό αρχείο CSS. Έτσι, κάθε αλλαγή στο στυλ της ιστοσελίδας μπορεί να γίνεται με μια μοναδική αλλαγή σε αυτό το αρχείο, αντί για την επεξεργασία πολλών σημείων σε κάθε σελίδα που υπάρχει στο site.
- Μικρότερο μέγεθος αρχείου, δεδομένου ότι ο κάθε κανόνας μορφοποίησης γράφεται μόνο μια φορά και όχι σε κάθε σημείο που εφαρμόζεται.
- Καλύτερο SEO (Search engine optimization). Οι μηχανές αναζήτησης δεν «μπερδεύονται» ανάμεσα σε περιεχόμενο και τη μορφοποίηση του, αλλά έχουν πρόσβαση στο περιεχόμενο σκέτο, οπότε είναι πολύ ευκολότερο να το καταγράψουν και να το αρχειοθετήσουν (indexing).
- Γρηγορότερες σελίδες. Όταν χρησιμοποιούμε εξωτερικό αρχείο CSS.Ο browser την πρώτη φορά που θα φορτώσει κάποια σελίδα του site μας το αποθηκεύει στην cache, οπότε δεν χρειάζεται να το κατεβάσει ξανά κάθε φορά που κατεβάζει ο χρήστης του κάποια άλλη σελίδα του site μας

3.6 ANTLR

Το εργαλείο ANTLR(Another Tool for Language Recognition) είναι ένα γλωσσικό εργαλείο αυτοματοποιημένης κατασκευής προγραμμάτων για την αναγνώριση διάφορων γλωσσών, βοηθώντας έτσι στην κατασκευή αναγνωριστών, μεταφραστών , μεταγλωττιστών. Το εργαλείο αυτό αναπτύσσεται από τον Terence Parr, καθηγητή του San Francisco, και τους συνεργάτες του από το 1989 και διανέμεται δωρεάν υπό τη μορφή πηγαίου κώδικα χωρίς κατοχυρωμένα πνευματικά δικαιώματα.

Κύριο χαρακτηριστικό του εργαλείου ANTLR είναι η κατασκευή συντακτικών αναλυτών αναδρομικής κατάβασης(recursive-descent parsers) από γραμματικές pred-LL(k), δηλαδή γραμματικές LL(k) για k>1 οι οποίες υποστηρίζουν κατηγορήματα (predicated).Ένας LL(k) συντακτικός αναλυτής, αναλύει τις λεκτικές μονάδες από πάνω προς τα κάτω χρησιμοποιώντας την αριστερότερη παραγωγή και χρησιμοποιώντας k προπορευόμενα σύμβολα. ένας συντακτικός αναλυτής αναδρομικής κατάβασης είναι ειδική υλοποίηση ενός LL συντακτικού αναλυτή, η οποία χρησιμοποιεί αναδρομική κλήση μεθόδων για την συντακτική ανάλυση αντί για να προσομοιώνει ένα αυτόματο στοίβας υλοποιημένο με πίνακα ακεραίων. Ο κώδικας που παράγεται μ'αυτή την υλοποίηση έχει τα παρακάτω πλεονεκτήματα σε σχέση με τις υλοποιήσεις που βασίζονται στην προσομοίωση αυτομάτων στοίβας:

- Είναι εύκολα αναγνώσιμος το οποίο διευκολύνει την απασφαλμάτωση
- Είναι αρκετά γρηγορότερος
- Διευκολύνει τον χειρισμό σφαλμάτων και την ανάληψη από αυτά

Τα κατηγορήματα επιτρέπουν αυθαίρετα σημασιολογικά και συντακτικά συμφραζόμενα να καθοδηγήσουν τη συντακτική ανάλυση με συστηματικό τρόπο. Έτσι ένας συντακτικός αναλυτής κατασκευασμένος από γραμματική pred-LL(k), μπορεί να αναγνωρίσει πολλές γλώσσες με συμφραζόμενα και πολλές μη-LL(k)/LR(k) γλώσσες χωρίς συμφραζόμενα. Τα σημασιολογικά κατηγορήματα καθορίζουν τη σημασιολογική εγκυρότητα εφαρμογής μιας παραγωγής, ενώ τα συντακτικά κατηγορήματα είναι τμήματα γραμματικής τα οποία περιγράφουν ένα συντακτικό συμφραζόμενο το οποίο πρέπει να ικανοποιηθεί πριν την αναγνώριση της αντίστοιχης παραγωγής.

Το εργαλείο ANTLR δέχεται σαν είσοδο μια γραμματική που καθορίζει μια γλώσσα και παράγει τον πηγαίο κώδικα εξόδου για ένα στοιχείο αναγνωρίσεις για αυτή την γλώσσα. Επιτρέπει επίσης τη δημιουργία lexers, parsers, tree parsers και συνδυασμό lexer-parser.Από προεπιλογή η ANTLR διαβάζει μια γραμματική και

δημιουργεί ένα αναγνωριστικό για τη γλώσσα που ορίζεται από τη γραμματική, δηλαδή ένα πρόγραμμα που διαβάζει ένα ρεύμα εισόδου και παράγει ένα σφάλμα αν το ρεύμα εισόδου δεν είναι σύμφωνο με τη σύνταξη που ορίζεται από τη γραμματική. Εάν δεν υπάρχουν συντακτικά λάθη, τότε η προεπιλεγμένη ενέργεια είναι απλώς μια έξοδο χωρίς να εκτυπωθεί κάποιο συγκεκριμένο μήνυμα. Η περιγραφή της αρχικής γλώσσας γίνεται με τρόπο τέτοιο ώστε να ενοποιείται ο καθορισμός της λεκτικής και συντακτικής ανάλυσης.

Για τον ορισμό της γραμματικής της αρχικής γλώσσας, το εργαλείο ANTLR χρησιμοποιεί τη σημειογραφία του εργαλείου YACC επεκταμένη με δομές Extended Backus-Naur Form(EBNF),δανεισμένες από τη σημειογραφία κανονικών εκφράσεων, για τις επαναλήψεις και την έκφραση επιλογών. Επίσης υπάρχει η δυνατότητα αναφοράς κυριολεκτικών(literals) απευθείας ως συμβολοσειρές στον ορισμό της γραμματικής. Τέλος, σε κάθε γραμματικό κανόνα υποστηρίζεται το πέρασμα παραμέτρων και επιστροφών τιμών, καθώς και η εκτέλεση αυθαίρετου κώδικα.

- Επιπλέον χαρακτηριστικά του εργαλείο ANTLR τα οποία το διαφοροποιούν από τα υπόλοιπα αντίστοιχα εργαλεία είναι τα εξής:
- Χειρισμός χωρίς επιπλέον κόστος Unicode χαρακτήρων (16-bit).
- Αυτόματες και μη λειτουργίες για χειρισμό λαθών.
- Πλήρως μεταφέρσιμος κώδικα, αφού είναι υλοποιημένος σε java.
- Παραγωγή κώδικα σε γλώσσες Java, C++, C#, Python, Perl, Ruby, JavaScript

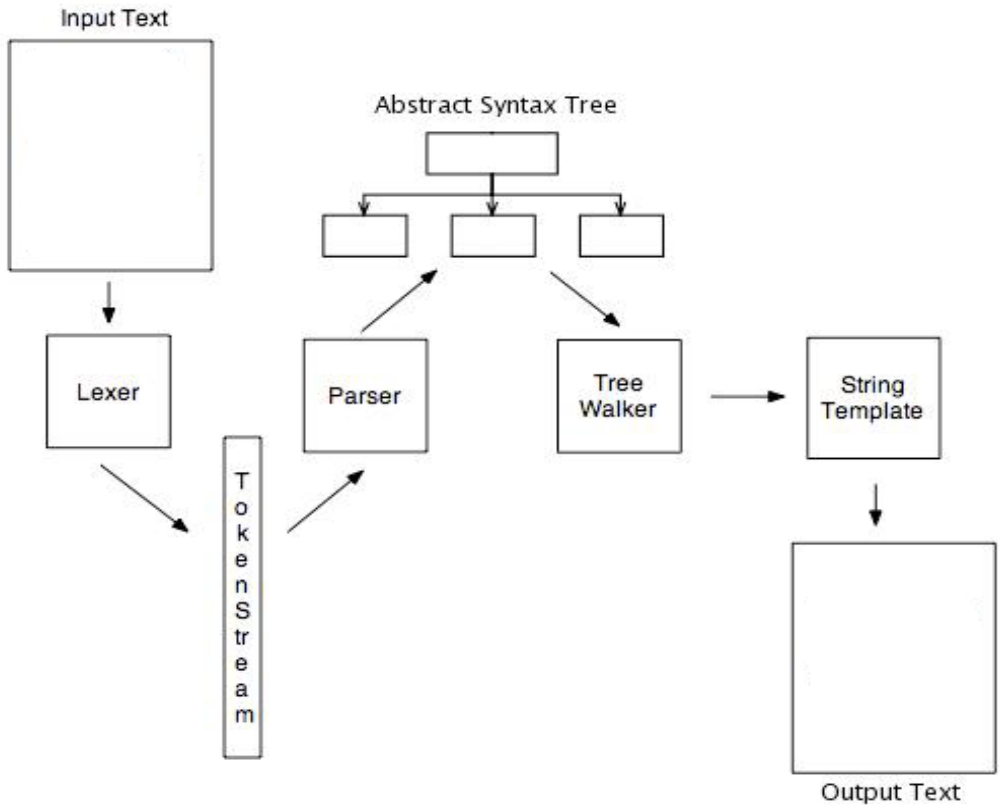
3.7 ANLTRWorks: ANTLR GUI Development Environment

Είναι ένα γραφικό περιβάλλον ανάπτυξης γραμματικής σχεδιασμένο να συνεργάζεται με τις γραμματικές που γράφονται για το εργαλείο ANLTR version3.Είναι δημιούργημα του Jean Bovet και προτείνεται σε αρκετές περιπτώσεις από τον Terence Parr,δημιουργό του εργαλείου ANLTR. Συνδυάζει μια εξαιρετική σύνταξη γραμματικής με διερμηνευτή για την ταχύτερη προτυποποίηση της γλώσσας και γρήγορο εντοπισμό σφαλμάτων για την εξάλειψη από γραμματικά λάθη. Το εργαλείο ANTLRWorks βοηθά στην εξάλειψη τον γραμματικών “nondeterminisms” ένα από τα πιο δύσκολα προβλήματα τόσο για αρχάριους όσο και για τους ποιο έμπειρους προγραμματιστές, επισημαίνοντας τα nondeterminisms μονοπάτια στο συντακτικό διάγραμμα που δημιουργεί και συνδέεται με μια γραμματική.

Κύριος στόχος του εργαλείου είναι να κάνει της γραμματικές πιο προσιτές στο μέσο προγραμματιστή, να βελτιώσει τη συντήρηση και την αναγνωσιμότητα των γραμματικών παρέχοντας άριστη πλοήγηση και εργαλεία επαναδημιουργίας της γραμματικής, και τέλος να αντιμετωπίσει τις πιο συχνές ερωτήσεις, προβλήματα και λάθη που αντιμετωπίζουν συχνά οι προγραμματιστές όταν δημιουργούν μια γραμματική.

4 Περιγραφή εφαρμογής

Σε αυτή την ενότητα παρουσιάζεται η αρχιτεκτονική και η λειτουργικότητα του συστήματος μας. Το παρακάτω διάγραμμα απεικονίζει την αρχιτεκτονική του συστήματος μας .



Εικόνα 11: Διάγραμμα εφαρμογής

Η κεντρική ιδέα στην οποία βασίζεται το σύστημα μας βασίζεται σε μια γραμματική που ακολουθεί το παρακάτω μοτίβο.

Grammar : [show|list|display] (me) (all) (our) <table_name>
SQL : SELECT * FROM <table_name>

Grammar : show (the) <column_name> of (our) <table_name>
SQL : SELECT <column_name> FROM <table_name>

Ψευδοκώδικας

```
statemens  : select

    select :  ask /(me)?/ /(all)?/ /(our)?/ table_name
                    { "SELECT * FROM table_name }

                  | ask /(με)?/ column_name /of/ /(our)?/
table_name [column_name, table_name]
                  { "SELECT column_name  FROM table_name" }

    ask : show|list|display

 table_name : TABLES

 column_name : COLUMNS
```

Πίνακας 5: Παράδειγμα ψευδοκώδικα

4.1 Αρχιτεκτονική του συστήματος

Το σύστημα μας αποτελείται από ένα σύνολο εργαλείων που συνεργάζονται και επικοινωνούν μεταξύ τους και κάθε ένα από αυτά έχει έναν καθορισμένο ρόλο. Τα τμήματα που αποτελείτε το σύστημα μας είναι τα εξής:

Βάση δεδομένων του σχήματος: Δεν είναι απαραίτητο ο χρήστης που θέλει να χρησιμοποίησει την εφαρμογή να ορίσει ή να δώσει σαν ξεχωριστό αρχείο τη βάση στην οποία θέλει να κάνει τα ερωτήματα. Ο μόνος περιορισμός είναι ότι η βάση πρέπει να είναι συνδεμένη και να εμφανίζονται όλα τα αποτελέσματα της στο γραφικό interface της web εφαρμογής που έχουμε δημιουργήσει.

Γραμματική: Δεν ορίζεται από τον χρήστη αλλά υπάρχει από πριν στο σύστημα μας και περιέχει τη σχέση των λέξεων-φράσεων που αντιστοιχούν σε λέξεις κλειδιά της sql γλώσσας. Επίσης περιέχει και κανόνες για τη συντακτική και λεκτική ανάλυση.

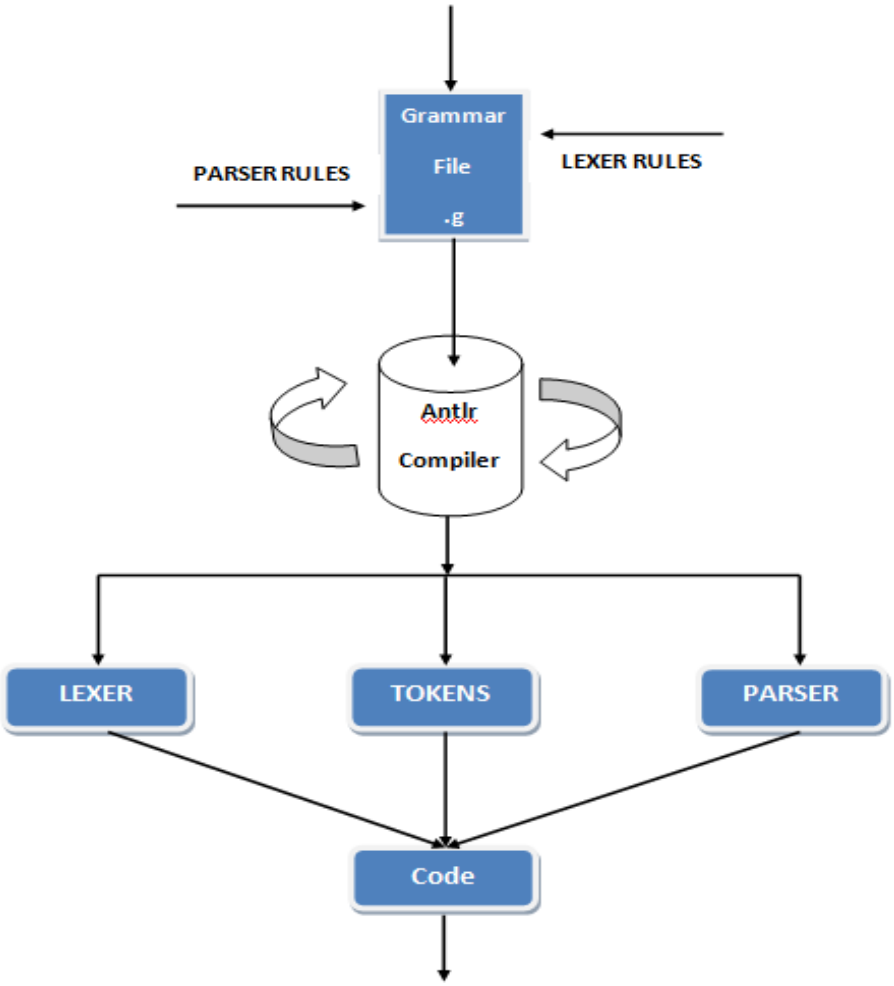
Ερώτημα: Είναι το κείμενο, πρόταση που πληκτρολογεί ο χρήστης προς το σύστημα μας για να εξάγει τα δεδομένα που θέλει.

Lexical analyzer: Είναι η λεξικογραφική ανάλυση πού εκτελείται στο ερώτημα που έθεσε ο χρήστης στο προηγούμενο στάδιο.

Syntax Analyzer: Εκτελείται η συντακτική ανάλυση προς τα λεξικολογικά δεδομένα που προήλθαν από τον λεκτικό αναλυτή καθώς και με την γραμματική που έχουμε ορίσει στο σύστημα μας.

Τελική Διαδικασία: Επεξεργασία των αποτελεσμάτων με βάση του συντακτικού αναλυτή και των κανόνων γραμματικής που έχουμε ορίσει στο αρχείο της γραμματικής.

SQL ερώτημα: Εκτέλεση κώδικα και παραγωγή του τελικού ερωτήματος σε sql μορφή



Εικόνα 12: Αρχιτεκτονική εφαρμογής (Α' Στάδιο)

4.2 Στάδια επεξεργασίας

Το σύστημα μας ακολουθεί τα εξής στάδια επεξεργασίας:

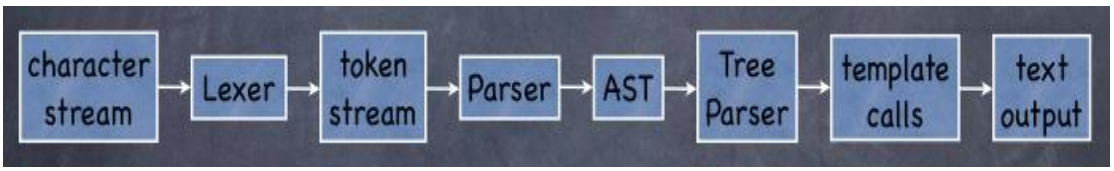
Αρχικά συνδέεται σε μια βάση δεδομένων για να εμφανίσει στον χρήστη που θέλει να εξάγει κάποια αποτελέσματα τα ακριβή ονόματα των πεδίων και της βάσης δεδομένων, τα οποία αποτελούν απαραίτητα στοιχεία για τη δημιουργία του ερωτήματος.

Στη συνέχεια σαν είσοδο το σύστημα δέχεται το ερώτημα που θέτει ο χρήστης σε μορφή κειμένου που το πληκτρολογεί ο χρήστης στο αντίστοιχο πεδίο. Το ερώτημα αυτό περνάει πρώτα από ένα lexical analyzer, ο οποίος διαβάζει και μετατρέπει την ακολουθία χαρακτήρων του ερωτήματος εισόδου σε λεξικά στοιχεία, τα λεγόμενα tokens.Για το συγκεκριμένο στάδιο χρησιμοποιήσαμε το lexical analyzer του εργαλείου antlr. Στη συνέχεια οι λεκτικές μονάδες που δημιουργήθηκαν από τον λεκτικό αναλυτή περνάνε στον συντακτικό αναλυτή και με βάση γραμματική που έχουμε ορίσει και τους κανόνες σε αυτήν παράγεται το sql κώδικας. Για το στάδιο αυτό χρησιμοποιήθηκε επίσης ο συντακτικός αναλυτής του εργαλείου antlr.

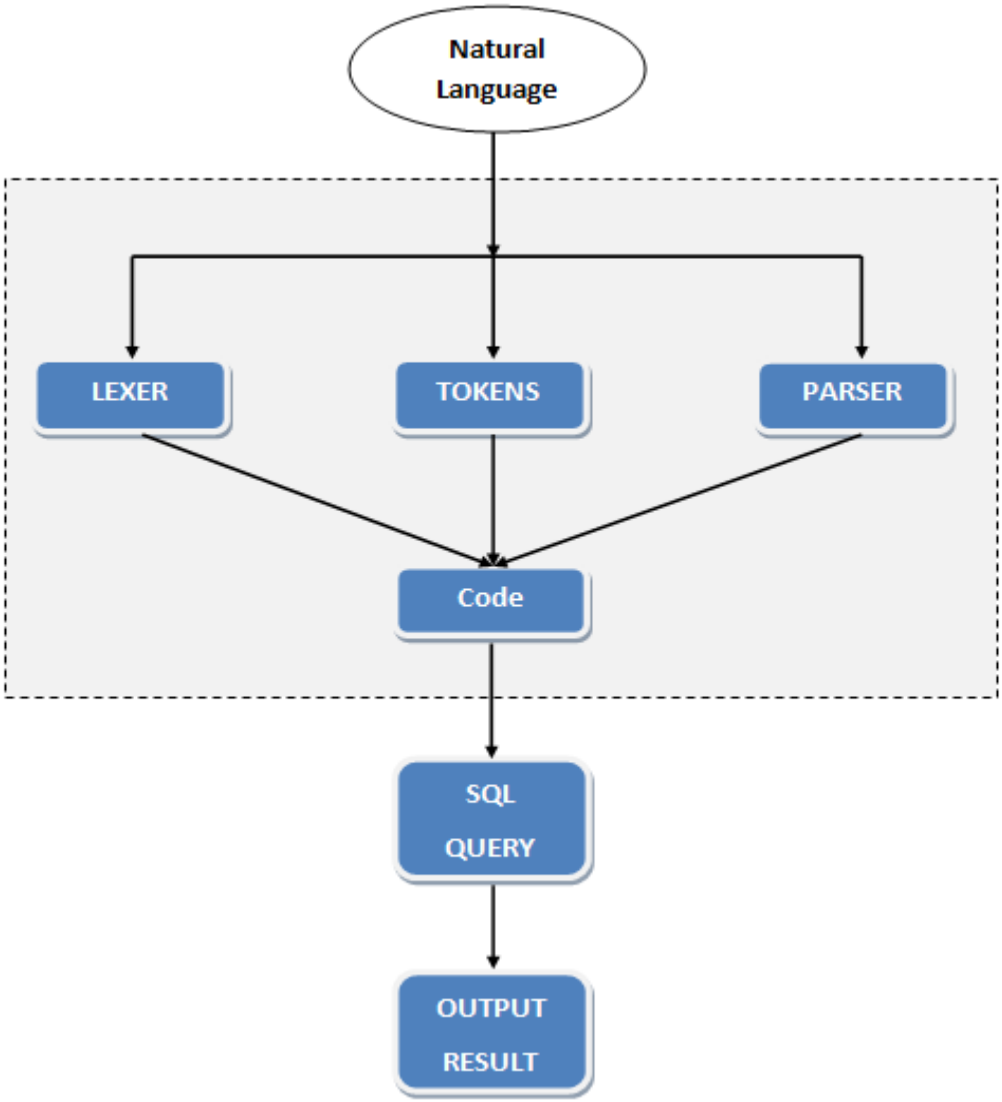
Στο τελικό στάδιο, ο κώδικας sql που δημιουργήθηκε ελέγχεται για την ορθότητα του και στη συνέχεια εκτελείτε από το γραφικό περιβάλλον που έχουμε δημιουργήσει με σκοπό να εμφανίσει στο χρήστη το αποτέλεσμα.

Το σύστημα μας λειτουργεί για απλά ερωτήματα sql της μορφής select <column_name> , <column_name> from <table_name> (where). Για να έχουμε ένα ερώτημα αποδεκτό από το σύστημα μας, πρέπει να τηρούνται κάποιες ορισμένες προϋποθέσεις και περιορισμοί στη σύνταξη των προτάσεων. Συγκεκριμένα, στο σύστημα μας υπάρχει διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων. Για παράδειγμα, η λέξη find δεν αντιμετωπίζεται όπως η λέξη fiND και αυτό οφείλεται στη γραμματική την οποία έχουμε ορίσει

Ο χρήστης προκείμενου να αναφερθεί σε ένα στοιχείο ενός πίνακα της βάσης δεδομένων πρέπει να το γράψει ακριβώς με το ίδιο όνομα που αναφέρεται στο πεδίο της συγκεκριμένης βάσης δεδομένων. Για να αναφερθεί σε έναν πίνακα, ο χρήστης πρέπει να αναφέρει το όνομα του πίνακα όπως ακριβώς αναφέρεται στη βάση δεδομένων.



Εικόνα 13: Βήματα επαλήθευσης εισόδου



Εικόνα 14: Αρχιτεκτονική εφαρμογής (Β' Στάδιο)

Η γενική μορφή της πρότασης είναι:

Find | show| displace *sentence1* from | of *sentence2* that | where |whose *sentence3*

Sentence1:

Σε αυτό το μέρος του ερωτήματος ο χρήστης πρέπει να αναφέρει τουλάχιστον ένα από τα στοιχείο των πινάκων της βάσης δεδομένων . Εάν θέλει να προσθέσει επιπλέον από ένα στοιχεία δηλαδή να εμφανίσει παραπάνω από ένα πεδία της βάσης, θα πρέπει να χωρίσει τα δύο πεδία με τη λέξη AND(και). Τέλος , ο χρήστης μπορεί επίσης να χρησιμοποιήσει τη λέξη all(*) που σχετίζεται με τη λέξη-κλειδί sql [*]. Σε αυτό το μέρος του ερωτήματος του χρήστη αναφέρεται στα στοιχεία που επιθυμεί να επιστρέφονται από το ερώτημα που θα πρέπει να γίνει στη βάση δεδομένων. Για παράδειγμα , ένα αποδεκτό sentence1 είναι :

find the name ή find the name and id

Sentence2:

Σε αυτό το μέρος του ερωτήματος ο χρήστης πρέπει να αναφέρει το όνομα τουλάχιστον ενός πίνακα της βάσης δεδομένων. Επίσης ο χρήστης πρέπει να αναφέρεται σε πίνακα του οποίου τα στοιχεία αναφέρονται στην ερώτηση που έχει διατυπώσει νωρίτερα. Παράδειγμα, ένα αποδεκτό sentence2 θα μπορούσε να είναι

of city ή of|from students

όπου τα city,students αποτελούν το όνομα του πίνακα της βάσης που εκτελείτε το κάθε ερώτημα.

Sentence3:

Σε αυτό το μέρος του ερωτήματος ο χρήστης πρέπει να αναφερθεί στις προϋποθέσεις που απαιτούνται για να λάβει το επιθυμητό αποτέλεσμα από το ερώτημα. Αυτό σημαίνει ότι ο χρήστης πρέπει να αναφερθεί στις προϋποθέσεις που θέλει το αποτέλεσμα να του εμφανίσει. Μεταξύ αυτών των προϋποθέσεων είναι γνωστές μας αριθμητικές πράξεις, μεγαλύτερο(greater than), μικρότερο (less than) ή ίσο(equal to). Μια αποδεκτή sentence3 θα μπορούσε να είναι:

that has population equal to 250000 ή has id greater than 12 ή that has name equal to john

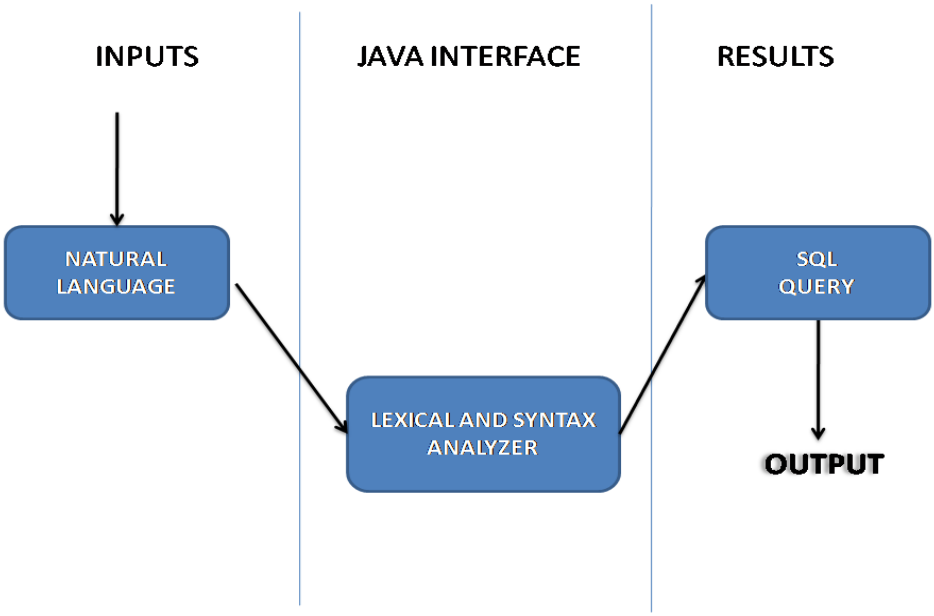
Δεν υπάρχει περιορισμός προς το χρήστη εάν θα δώσει συμβολοσειρά ή αριθμό μετά τις εκφράσεις equal to, greater than , less than. Ο μόνος περιορισμός του χρήστη σε αυτό το βήμα είναι ότι εάν δοθεί κάποια συμβολοσειρά που αναφέρεται προφανώς σε κάποιο από τα πεδία της βάσης δεδομένων, αυτή πρέπει να διατυπωθεί όπως ακριβώς εμφανίζεται στο πεδίο αυτό, δεν λαμβάνεται όμως υπόψη ο περιορισμός πεζών και κεφαλαίων γραμμάτων.

Κατά την συντακτική ανάλυση του ερωτήματος του χρήστη, παράγεται κώδικας sql.Όταν ολοκληρώνεται η ανάλυση ό κώδικας που παράγεται δεν μοιάζει τόσο με τον κώδικα ερωτήσεων sql. Χρειάζεται να επέμβουμε στον τελικό κώδικα της συντακτικής ανάλυσης ώστε όταν επαληθεύεται η χρήση ενός η παραπάνω κανόνων που υπάρχουν στη γραμματική να εκτελείτε κάποιος συγκεκριμένος κώδικας. Έτσι κάνοντας σωστή χρήση των δεδομένων της συντακτικής αλλά και τις λεκτικής ανάλυσης είμαστε σε θέση να δημιουργήσουμε το τελικό ερώτημα που μοιάζει περισσότερο με τον κώδικα sql ερωτημάτων.

Μερικά παραδείγματα τα οποία μπορεί να εισάγει ο χρήστης στο σύστημα μας και πως αυτά μετατρέπονται σε SQL ερωτήματα :

1- display the id of city that has population less than 440900 2- show all records from city 3- find countrycode of city that has id equal to 10 4- show the name of city that has population equal to 149544 5- find id and name of city that population greater than 731200
1- SELECT id FROM city WHERE population<440900 2- SELECT * FROM city 3- SELECT countrycode FROM city WHERE id='10' 4- SELECT name FROM city WHERE population='149544' 5- SELECT id,name FROM city WHERE population>731200

Πίνακας 6: Αποδεκτές προτάσεις συστήματος



Εικόνα 15: Αρχιτεκτονική εφαρμογής

4.3 Ανάλυση γραμματικής

4.3.1 Γραμματική BNF

Στη θεωρητική πληροφορική, η BNF (Κανονική μορφή του Μπάκους, αγγλ. Backus Normal Form ή Μορφή Μπάκους-Νάουρ, αγγλ. Backus–Naur Form) είναι μια τεχνική συμβολισμού (μετασύνταξη) για γραμματικές χωρίς συμφραζόμενα (context-free grammars),που συχνά χρησιμοποιείται για να περιγράψει τη σύνταξη μιας γλώσσας της πληροφορικής, όπως οι γλώσσες προγραμματισμού υπολογιστών, οι τύποι εγγράφων (document formats), τα σύνολα εντολών (instruction sets) και τα πρωτόκολλα επικοινωνιών. Εφαρμόζεται όπου χρειάζονται ακριβείς περιγραφές γλωσσών, για παράδειγμα σε επίσημους ορισμούς γλωσσών, σε εγχειρίδια, ή σε βιβλία για θεωρία γλωσσών προγραμματισμού.

Υπάρχουν πολλές επεκτάσεις και παραλλαγές της αρχικής BNF, κάποιες από αυτές είναι αυστηρά ορισμένες, όπως η Εκτεταμένη Μορφή Μπάκους-Νάουρ (Extended Backus–Naur Form, EBNF) και η Επαυξημένη Μορφή Μπάκους-Νάουρ (Augmented Backus–Naur Form, ABNF).

Υπάρχουν πολλές παραλλαγές και επεκτάσεις της BNF, συνήθως με σκοπό την απλότητα ή την οικονομία, ή για συγκεκριμένες εφαρμογές. Συνηθισμένο χαρακτηριστικό σε πολλές παραλλαγές είναι η χρήση τελεστών κανονικών εκφράσεων όπως ο * και ο +. Η Εκτεταμένη Μορφή Μπάκους-Νάουρ (Extended Backus–Naur Form, EBNF) είναι αρκετά συνηθισμένη - ακόμα και το παραπάνω παράδειγμα, δεν ήταν στην αρχική μορφή που χρησιμοποιήθηκε στην αναφορά της ALGOL 60. Η σημειογραφία με τις αγκύλες "[ ]" εισήχθηκε κάποια χρόνια αργότερα, από τον ορισμό της IBM για την PL/I αλλά σήμερα είναι ευρέως αποδεκτός. Η Επαυξημένη Μορφή Μάκους-Νάουρ (Augmented Backus–Naur Form, ABNF) και η RBNF είναι δύο άλλες επεκτάσεις που χρησιμοποιούνται συχνά για την περιγραφών πρωτοκόλλων της Internet Engineering Task Force (IETF).

Οι γραμματικές συντακτικής ανάλυσης εκφράσεων (parsing expression grammars) βασίζονται στη BNF και σε συμβολισμούς κανονικών εκφράσεων για το σχηματισμό μιας εναλλακτικής κλάσης τυπικών γραμματικών, που είναι αναλυτικές, αντί για παραγωγικές γραμματικές (generative grammars).

Πολλοί ορισμοί BNF που υπάρχουν διαθέσιμοι στο Διαδίκτυο σήμερα προσπαθούν να είναι ευανάγνωστοι και όχι τυπικοί. Συχνά περιλαμβάνουν τους εξής συντακτικούς κανόνες και επεκτάσεις:

- Προαιρετικά αντικείμενα μέσα σε τετράγωνες αγκύλες, π.χ. [<αντικείμενο-x>]
- Αντικείμενα που επαναλαμβάνονται 0 ή περισσότερες φορές τοποθετούνται σε αγκύλες ή στο τέλος τους προστίθεται ένας αστερίσκος, π.χ. <λέξη> ::= <γράμμα> {<γράμμα>}
- Αντικείμενα που επαναλαμβάνονται 1 ή περισσότερες φορές ακολουθούνται από '+'

- Τα τερματικά σύμβολα μπορούν να εμφανίζονται με έντονη γραφή και τα μη-τερματικά σε απλή γραφή (και όχι σε πλάγια γραφή και μέσα σε σύμβολα <, >)
- Η επιλογή μεταξύ εναλλακτικών σε μια παραγωγή δείχνεται με το σύμβολο '|'. Π.χ., <εναλλακτική-Α> | <εναλλακτική-Β>
- Η ομαδοποίηση αντικειμένων γίνεται με παρενθέσεις

Λογισμικό που χρησιμοποιεί BNF γραμματική

- Yacc, γεννήτρια συντακτικών αναλυτών (χρησιμοποιείται μαζί με το Lex).
- ANTLR, γεννήτρια συντακτικών αναλυτών γραμμένη σε Java.
- BNF Converter (BNFC) (BNFC).
- Coco/R, γεννήτρια μεταγλωττιστών που δέχεται γραμματικές χαρακτηριστικών (attributed grammars) σε EBNF.
- GNU bison, η έκδοση GNU του yacc.
- RPA, BNF συντακτική ανάλυση. Επίδειξη συντακτικής ανάλυσης με σελίδα PHP: JavaScript, XML.

4.4 Παράδειγμα χρήσης γραμματικής

Όπως αναφέρθηκε και στα προηγούμενα κεφάλαια απαραίτητο χαρακτηριστικό στοιχείο της εφαρμογής μας είναι η γραμματική που έχουμε ορίσει και αποτελεί το κύριο μέρος αυτής της πτυχιακής. Η γραμματική αναπτύχθηκε με το εργαλείο ANTLR και ακολουθεί τους κανόνες της κατηγορίας των BNF γραμματικών.Για να μπορέσει να γίνει σωστά αποδεκτή από το σύστημα μας πρέπει να πλήρη κάποιες προϋποθέσεις, ως προς το πως θα ορίσουμε τα δεδομένα μας, τι λεξιλόγια θα χρησιμοποιήσουμε, ποιες είναι οι αποδεκτές λέξεις και εκφράσεις καθώς και το συντακτικό της γραμματικής. Το πώς θα ορίσουμε τη γραμματική μας έχει κάποιους αυστηρούς κανόνες που πρέπει να τηρούνται ρητά. Στα επόμενο κεφάλαιο θα αναπτυχθεί αναλυτικά το πώς πρέπει να γράψουμε μια γραμματική που είναι αποδεκτή από το εργαλεία ANTLR καθώς και πως θα μας βοηθήσει το εργαλείο ANTLRWorks σε αυτή την διαδικασία.

Απαραίτητα βήματα που πρέπει να ακολουθήσουμε ώστε η γραμματική μας να γίνεται αποδεκτή από το εργαλείο ANTLR:

Το όνομα της γραμματικής και το όνομα του αρχείου μας πρέπει να είναι πανομοιότυπα. Το αρχείο πρέπει να έχει την κατάληξη .g και επιπλέον κάθε αρχείο γραμματικής είναι απαραίτητο να ξεκινάει με την ονομαστική δήλωση της γραμματικής. Παράδειγμα:

```
grammar MyGrammarName
```

Πίνακας 7: Όνομα γραμματικής

Επειδή η τελική μας γλώσσα είναι η java και κάθε κλάση στη java πρέπει να δηλώνεται και το πακέτο η Antlr δεν δημιουργεί δηλώσει πακέτων στην κορυφή των παραγόμενων java κλάσεων. Πρέπει να χρησιμοποιήσουμε την εντολή @parser::header και @parser::lexer για να αναγκάσουμε να δηλωθούν και τα πακέτα. Παράδειγμα:

```
@lexer::header {
  package gr.org.epp.parsers;
}
@parser::header {
  package gr.org.epp.parsers;
}
```

Πίνακας 8: Options γραμματικής

Κάθε αρχείο γραμματικής πρέπει να έχει τουλάχιστον έναν κανόνα lexer. Κάθε κανόνας lexer πρέπει να αρχίζει με κεφαλαίο γράμμα. Στο παράδειγμα που ακολουθεί έχουμε δύο κανόνες ο πρώτος ορίζει ένα σύμβολο χαιρετισμού και ο δεύτερος ένα διακριτικό τερματικού συμβόλου. Ο χαιρετισμός είναι “hello word” και το τερματικό σύμβολο “ ! “.

```
SALUTATION:'Hello word'

ENDSYMBOL:'!'
```

Πίνακας 9: Παράδειγμα λεκτικού κανόνα

Ομοίως κάθε αρχείο γραμματικής πρέπει να έχει τουλάχιστον έναν κανόνα parser. Κάθε κανόνας parser πρέπει να αρχίζει με μικρό γράμμα. Στο παράδειγμα έχουμε μόνο έναν κανόνα Οποιαδήποτε έκφραση στη γραμματική μας πρέπει να αποτελείτε από έναν χαιρετισμό και να ακολουθείτε από ένα τερματικό σύμβολο :

```
expression : SALUTATION ENDSYMBOL
```

Πίνακας 10: Παράδειγμα κανόνα γραμματικής

Σύμφωνα λοιπόν με τα παραπάνω η γραμματική που δημιουργήσαμε και είναι αποδεκτή από το σύστημα ANTLR παρουσιάζεται παρακάτω:

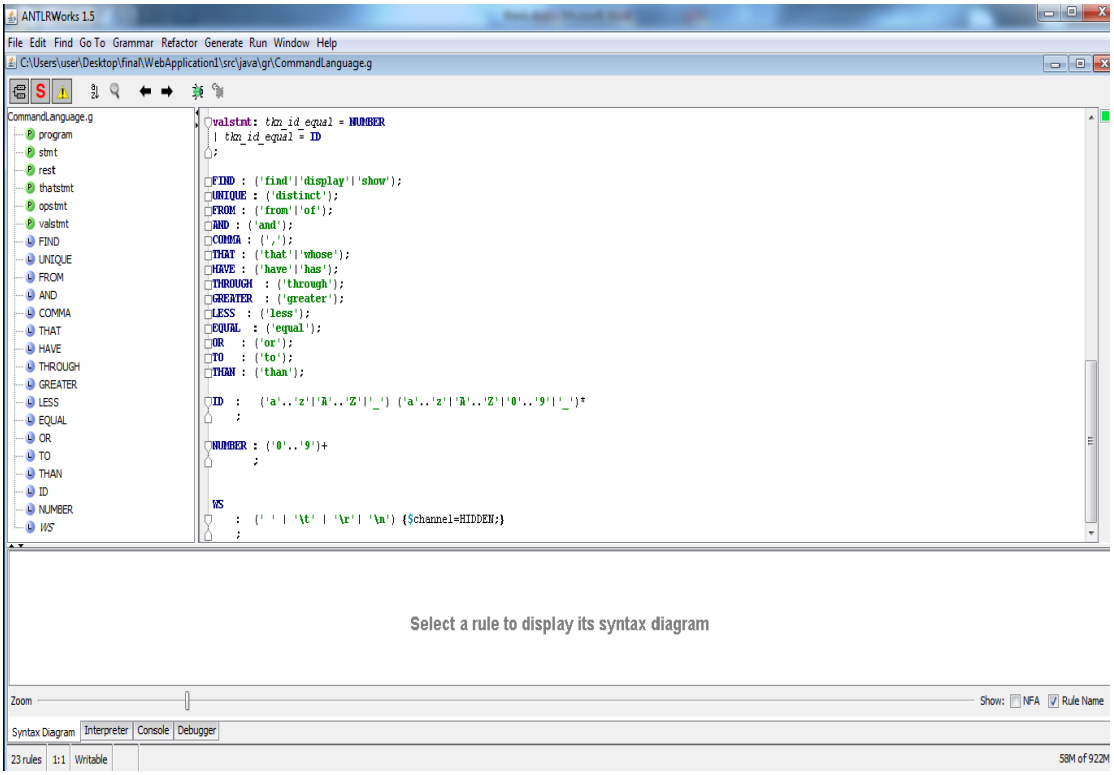
```
/*------------------------------------------------------------------
 * PARSER RULES
 *------------------------------------------------------------------*/
program: stmt+
;
stmt: FIND ALL (ID)? FROM ID
| FIND ALL (ID)? FROM ID rest
| FIND (ID)? ID rest
;

rest: FROM ID thatstmt
| FROM ID EOF
| AND ID rest
| AND ID thatstmt
;

thatstmt: THAT (ID)? ID opstmt
;

opstmt: (IS)? GREATER THAN valstmt
| (IS)? LESS THAN valstmt
| (IS)? EQUAL TO valstmt
;
valstmt: NUMBER
| ID
;
/*------------------------------------------------------------------
 * LEXER RULES
 *------------------------------------------------------------------*/
FIND : ('find'|'display'|'show');
FROM : ('from'|'of');
AND : ('and');
COMMA : (',');
THAT : ('that'|'whose');
GREATER  : ('greater');
LESS  : ('less');
EQUAL  : ('equal');
OR   : ('or');
TO   : ('to');
THAN : ('than');
IS: ('is');
ALL: ('all');
ID  :       ('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_')*
    ;
NUMBER : ('0'..'9')+
       ;
WS
    :  (' ' | '\t' | '\r'| '\n') {$channel=HIDDEN;}
    ;
```

Πίνακας 11: Η γραμματική του συστήματος

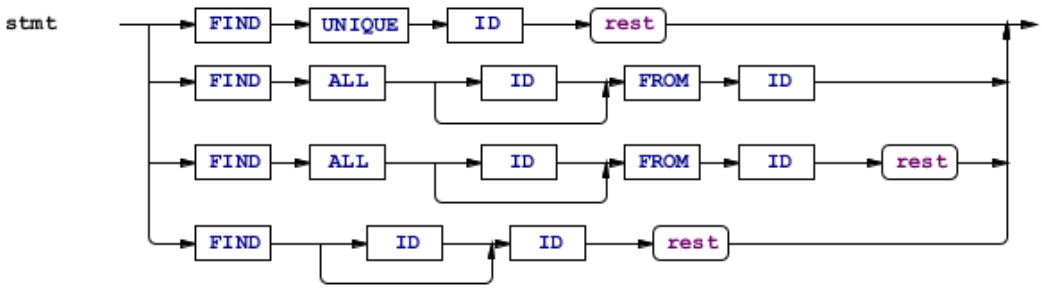
Σύμφωνα λοιπόν με όσα αναλύθηκαν παραπάνω η γραμματική μας συμφωνεί στο πως γράφονται οι lexer και parser rules καθώς και στο υπόλοιπο συντακτικό της.

4.5 Η γραμματική στο Antlrworks

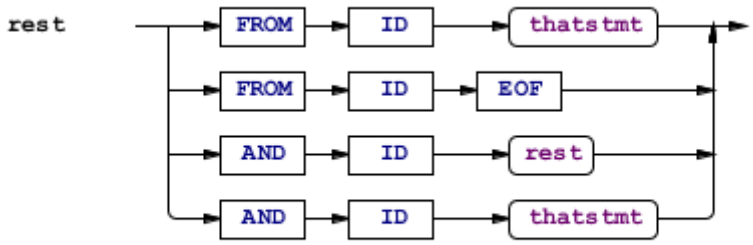
Το εργαλείο που μας βοηθήσει και αποτέλεσε επίσης ένα μεγάλο κομμάτι της πτυχιακής είναι το εργαλείο ANTLRWorks το οποίο αναλύθηκε σε προηγούμενο κεφάλαιο, είναι μια γραφική απεικόνιση της γλώσσας μας ώστε να μπορέσουμε να αποφύγουμε λάθη και να γίνεται ακόμα ποιο κατανοητή. Να διαπιστώσουμε τα σημεία στα οποία υστερεί η γραμματικής μας διορθώνοντας τας αρκετά γρήγορα, διευκολύνοντας έτσι την διαδικασία της απασφαλμάτωσης.



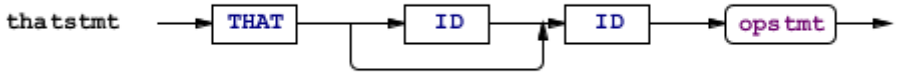
Εικόνα 16: Περιβάλλον ANTLRWorks



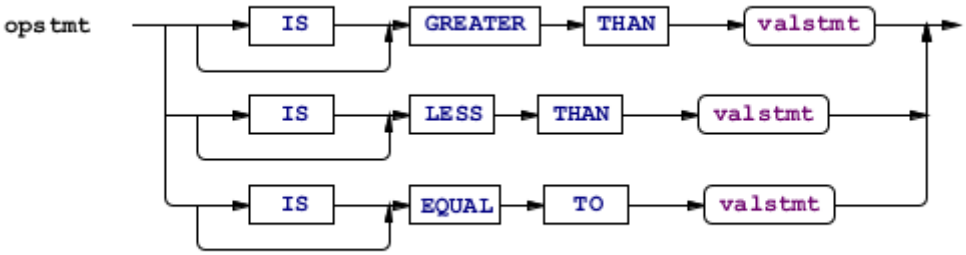
Εικόνα 17: Κανόνας γραμματικής stmt



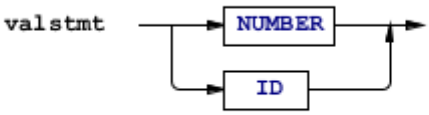
Εικόνα 18: Κανόνας γραμματικής rest



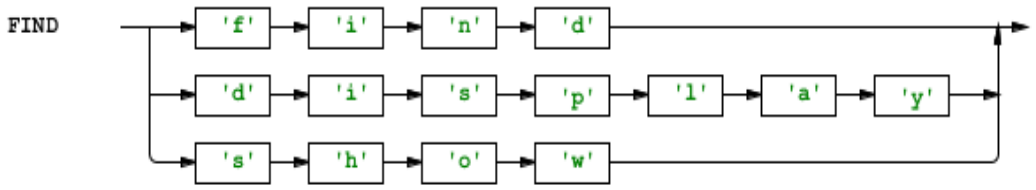
Εικόνα 19: Κανόνας γραμματικής thatstmt



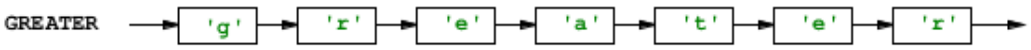
Εικόνα 20: Κανόνας γραμματικής opstmt



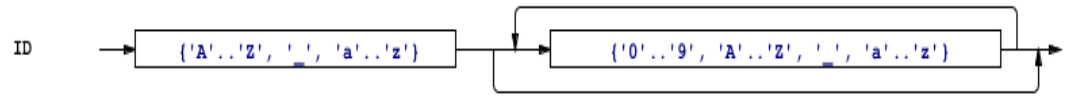
Εικόνα 21: Κανόνας γραμματικής valstmt



Εικόνα 22: Παράδειγμα λεκτικού κανόνα γραμματικής 1



Εικόνα 23: Παράδειγμα λεκτικού κανόνα γραμματικής 2



Εικόνα 24: Παράδειγμα λεκτικού κανόνα γραμματικής 3

Αφού λοιπόν υλοποιήσαμε την γραμματική μας το βήμα που πρέπει να ακολουθήσουμε είναι να περάσει η γραμματική μας από lexical analyzer ώστε να πάρουμε τα λεγόμενα Tokens, στη συνέχεια από έναν syntactic analyzer και τέλος από έναν semantic analyzer.Με άλλα λόγια η γραμματική μας πρέπει να γίνει compile ώστε να πάρουμε σε java γλώσσα των κώδικα της γραμματικής μας. Αφού πάρουμε τον κώδικα αυτός θα μας βοηθήσει να γράψουμε επιπλέον κώδικα που θέλουμε ώστε να υλοποιήσουμε την εφαρμογή μας είναι να αλλάξουμε όποια άλλα σημεία επιθυμούμε. Για τις διαδικασίες lexical analyzer και syntactic analyzer θα χρησιμοποιούμε το λεκτικό και συντακτικό αναλυτή του εργαλείου ANTLR.

Στη φάση της λεκτικής ανάλυσης, το ρεύμα χαρακτήρων εισόδου επεξεργάζεται για τον διαχωρισμό των χαρακτήρων σε λεκτικές μονάδες και την κατηγοριοποίηση αυτών. Η έξοδος του λεκτικού αναλυτή είναι το ρεύμα των λεκτικών μονάδων που προκύπτουν

Στη φάση της συντακτικής ανάλυσης, το ρεύμα λεκτικών μονάδων που προέκυψε από την λεκτική ανάλυση του πηγαίου κώδικα γραμματικής, επεξεργάζεται

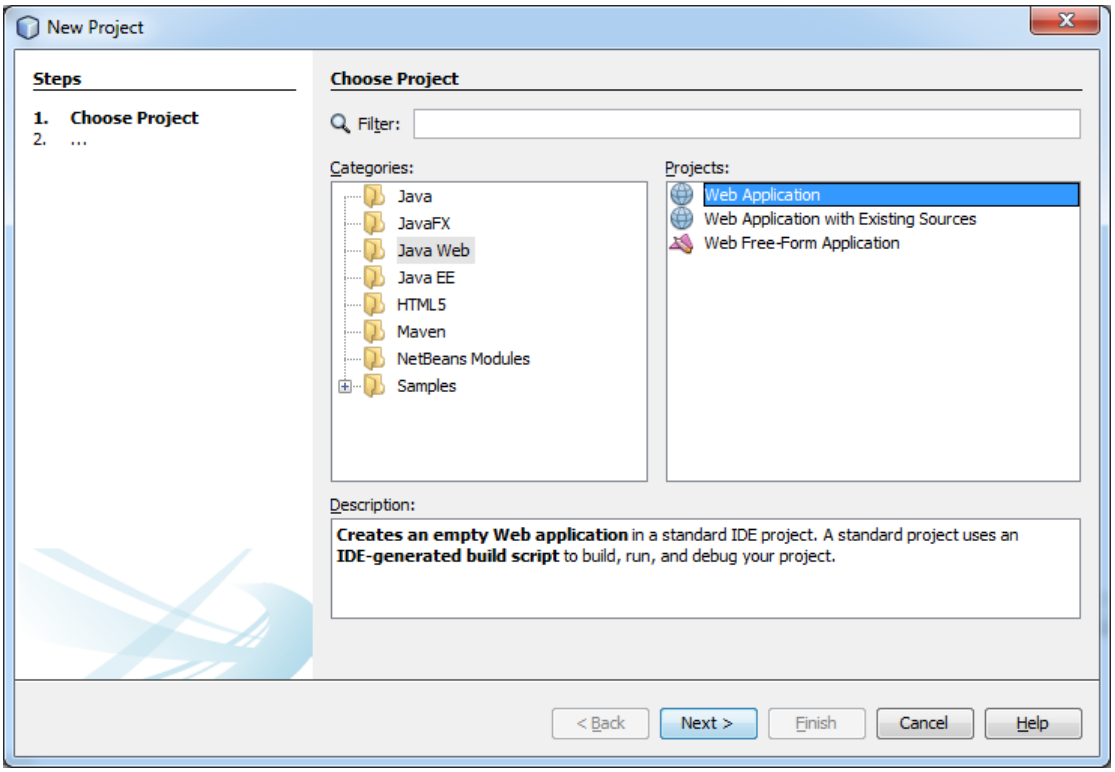
για τον έλεγχο συντακτικών λαθών σύμφωνα με τον ορισμό της γραμματικής της γλώσσας, και για τη κατασκευή του αφηρημένου συντακτικού δέντρου το οποίο αναπαριστά τον πηγαίο κώδικα σε πιο επεξεργάσιμη μορφή για την δομή του μεταφραστή. Το δέντρο αυτό αποτελεί και την έξοδο του συντακτικού αναλυτή

5 Υλοποίηση Εφαρμογής Servlet

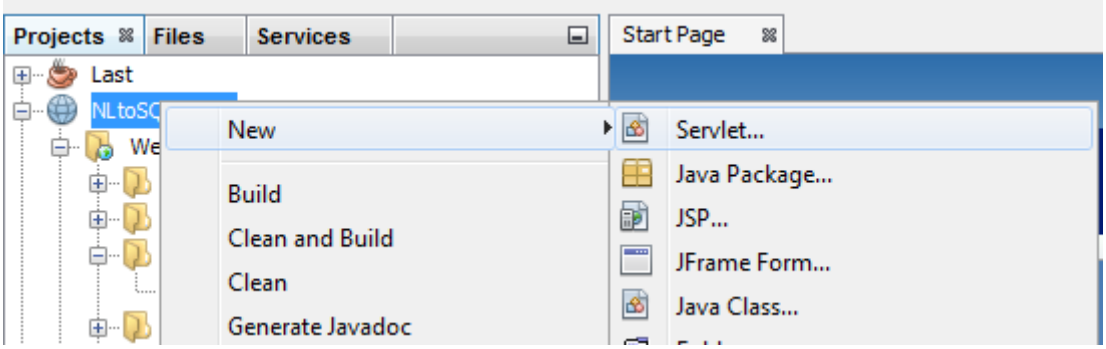
5.1 Υλοποίηση Servlet

Αφού λοιπόν δημιουργήσουμε τον τελικό μας κώδικα πρέπει με κάποιον τρόπο να καλέσουμε τις κατάλληλες μεθόδους για να διαπιστώσουμε ένα η γραμματική μας είναι σωστή, έχουν γίνει όλα τα στάδια με επιτυχία και να ελέγχουμε εάν τελικά κάνει αυτό που της έχουμε ορίσει. Για να γίνει όλη η διαδικασία σωστά δημιουργούμε ένα Web Application στο εργαλείο NetBeans IDE που χρησιμοποιούμε για να υλοποιήσουμε την εφαρμογή μας και κάνει χρήση του tomcat server , ακολουθώντας τα παρακάτω βήματα:

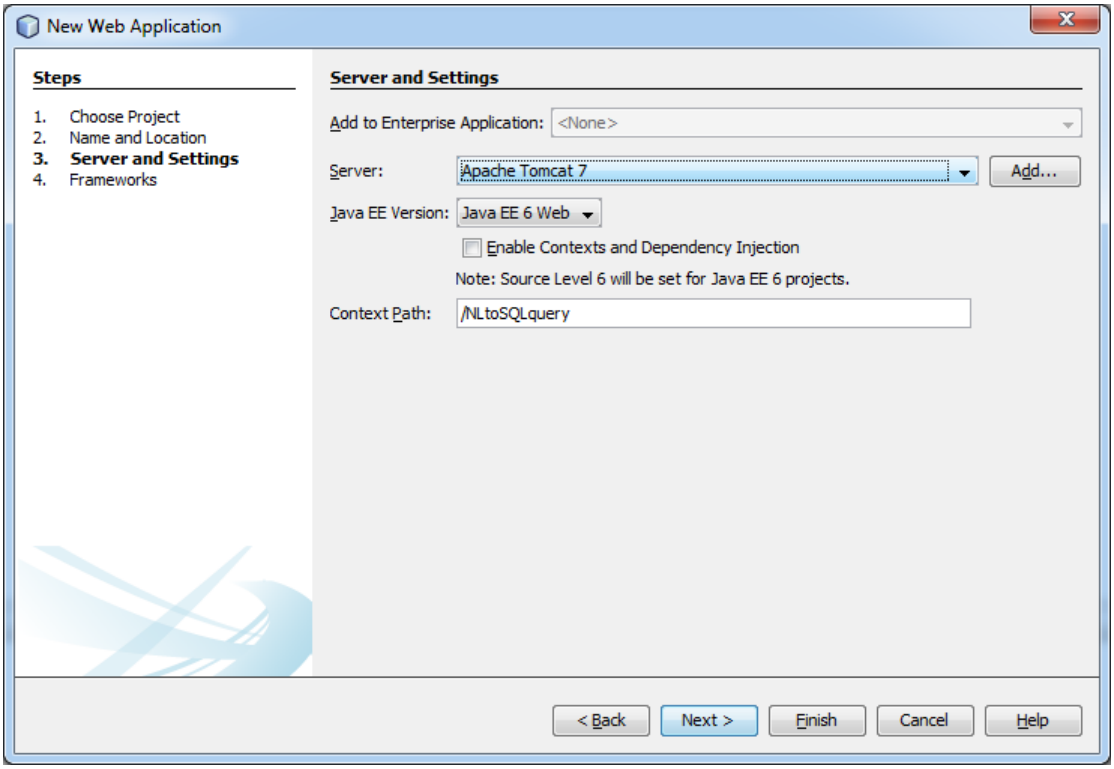
File->New Project->Java Web->Web Application



Εικόνα 25: Δημιουργία WebApplication σε NETBEANS IDE

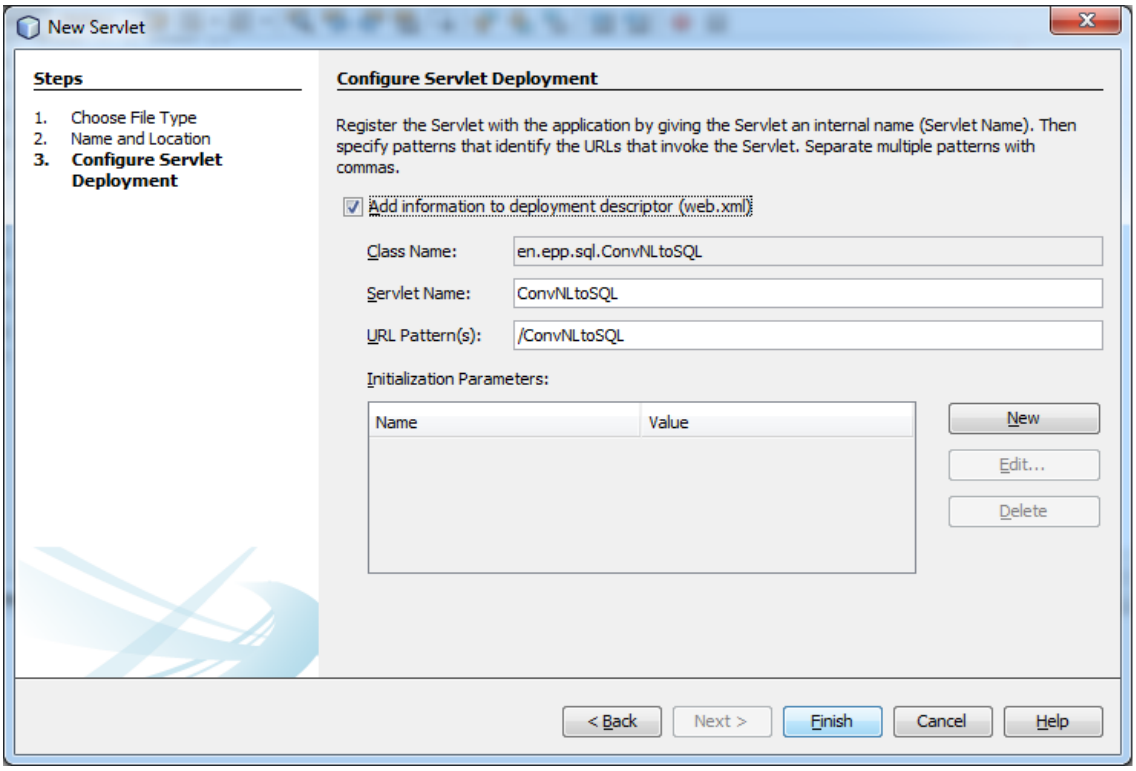


Εικόνα 26: Δημιουργία Servlet σε NETBEANS IDE



Εικόνα 27: Επιλογή Servlet Server

.



Εικόνα 28: Deploy web.xml στο Servlet

Στη συνέχεια γράφουμε τον κώδικα που χρειάζεται. Αρχικά πρέπει η είσοδος να μετατραπεί σε ένα org.antlr.runtime.CharStream.

```
ANTLRStringStream in = new ANTLRStringStream("Some text here")
```

Πίνακας 12: Βήμα 1 επαλήθευσης Γραμματικής

Αφού λοιπόν έχει γίνει η μετατροπή σε StringStream το επόμενο στάδιο είναι ο κώδικας αυτός να περάσει από τον λεκτικό αναλυτή ώστε να πάρουμε σαν αποτέλεσμα το TokenStream

```
CommandLanguageLexer lexer = new CommandLanguageLexer(in)
CommonTokenStream tokens = new CommonTokenStream(lexer)
```

Πίνακας 13: Βήμα 2 επαλήθευσης Γραμματικής

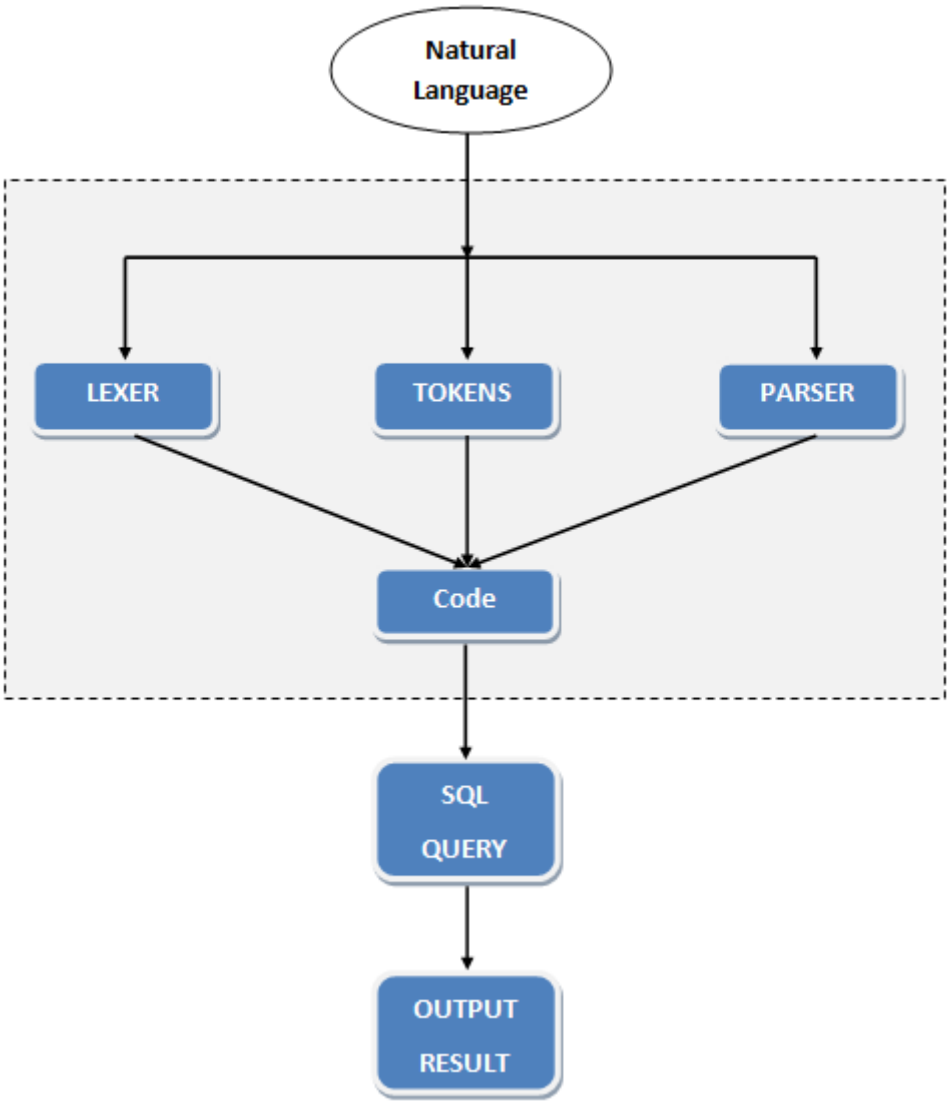
Αφού ολοκληρωθεί και η διαδικασία του TokenStream και έχει δημιουργήσει ο κώδικας μας τα Tokens σύμφωνα με την ακολουθία την οποία του έχουμε δώσει στη συνέχεια μπορούμε να δημιουργήσουμε ένα στιγμιότυπο του parser.Ουσιαστικά

αυτό που γίνεται δηλαδή είναι να περάσει το TokenStream που έχουμε πάρει σαν αποτέλεσμα από τον λεκτικό αναλυτή, στον συντακτικό αναλυτή.

```
CommandLanguageParser parser = new CommandLanguageParser(tokens)
```

Πίνακας 14: Βήμα 3 επαλήθευσης Γραμματικής

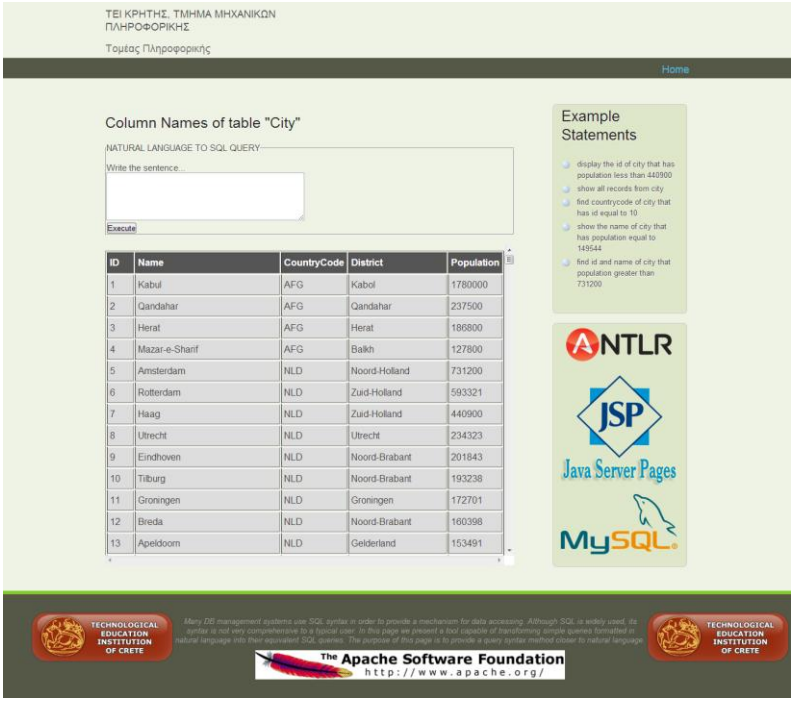
Το αποτέλεσμα που παίρνουμε από την συγκεκριμένα διαδικασία και σύμφωνα πάντα με την γραμματική που έχουμε ορίσει είναι ένα String το οποίο είναι το αποτέλεσμα ενός ερωτήματος σε φυσική γλώσσα και έχει τη μορφή SQL ερωτήματος.



Εικόνα 29: Τελική Εφαρμογή Servlet

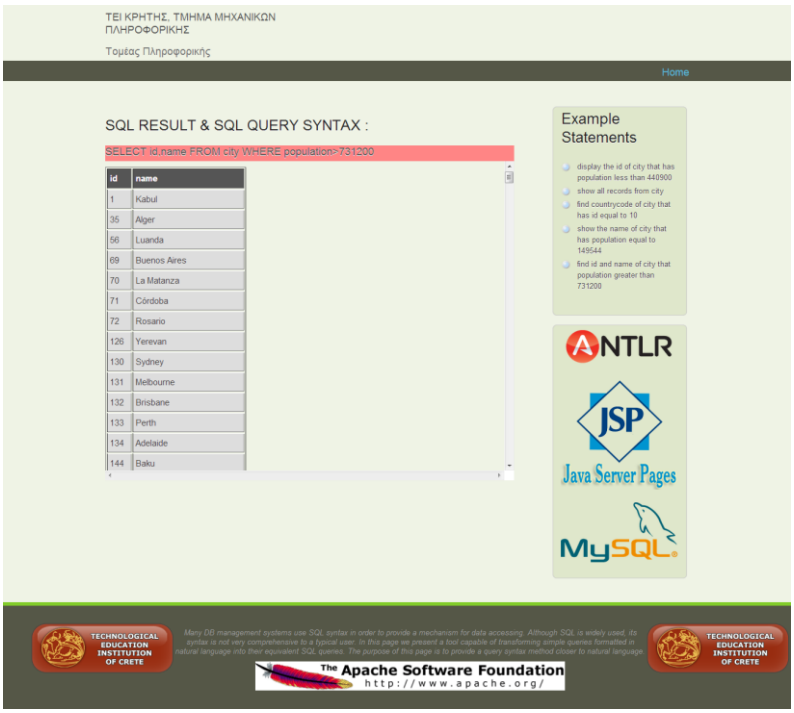
5.2 Αποτέλεσμα Servlet

Αφού δημιουργήσαμε με επιτυχία το servlet μας αυτό που παίρνουμε σαν αποτέλεσμα είναι το Interface της εφαρμογής μας το οποίο φαίνεται στην παρακάτω εικόνα.



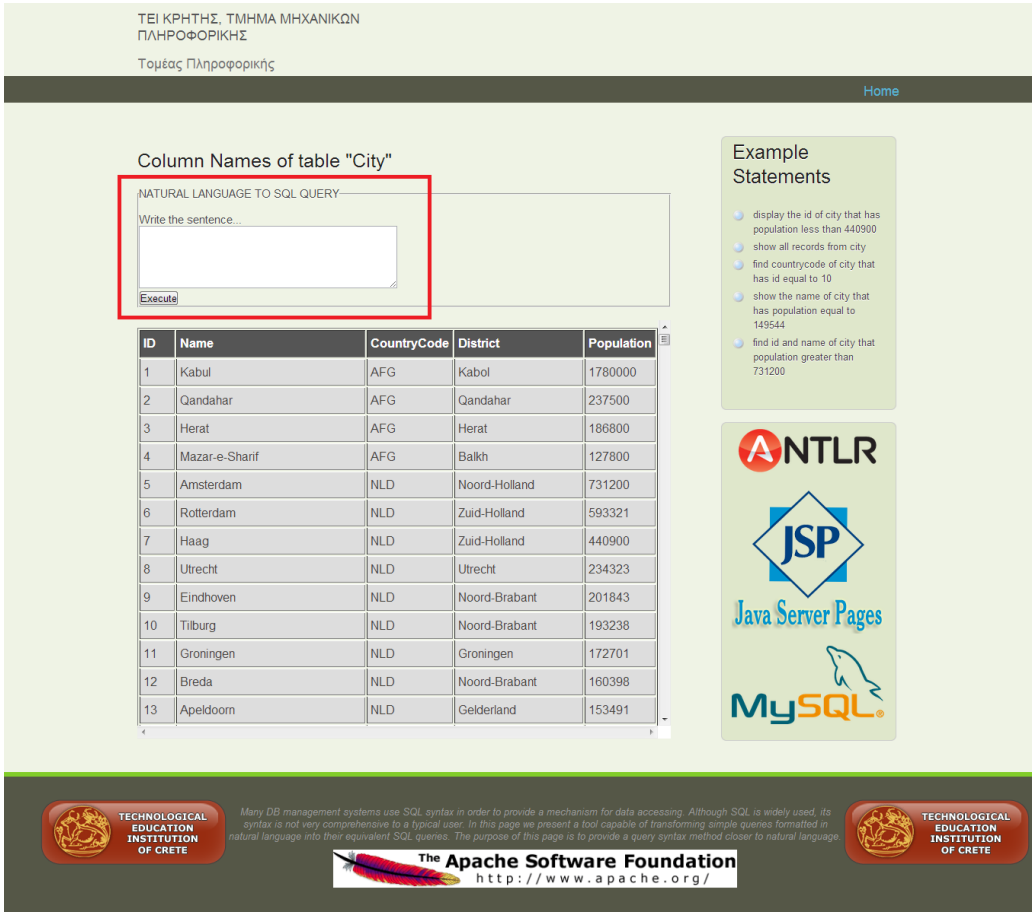
Εικόνα 30: Διεπαφή εφαρμογής

Σαν αποτέλεσμα του ερωτήματος του χρήση προς το σύστημα μας παίρνουμε την παρακάτω εικόνα με τα αποτελέσματα καθώς και το SQL ερώτημα:



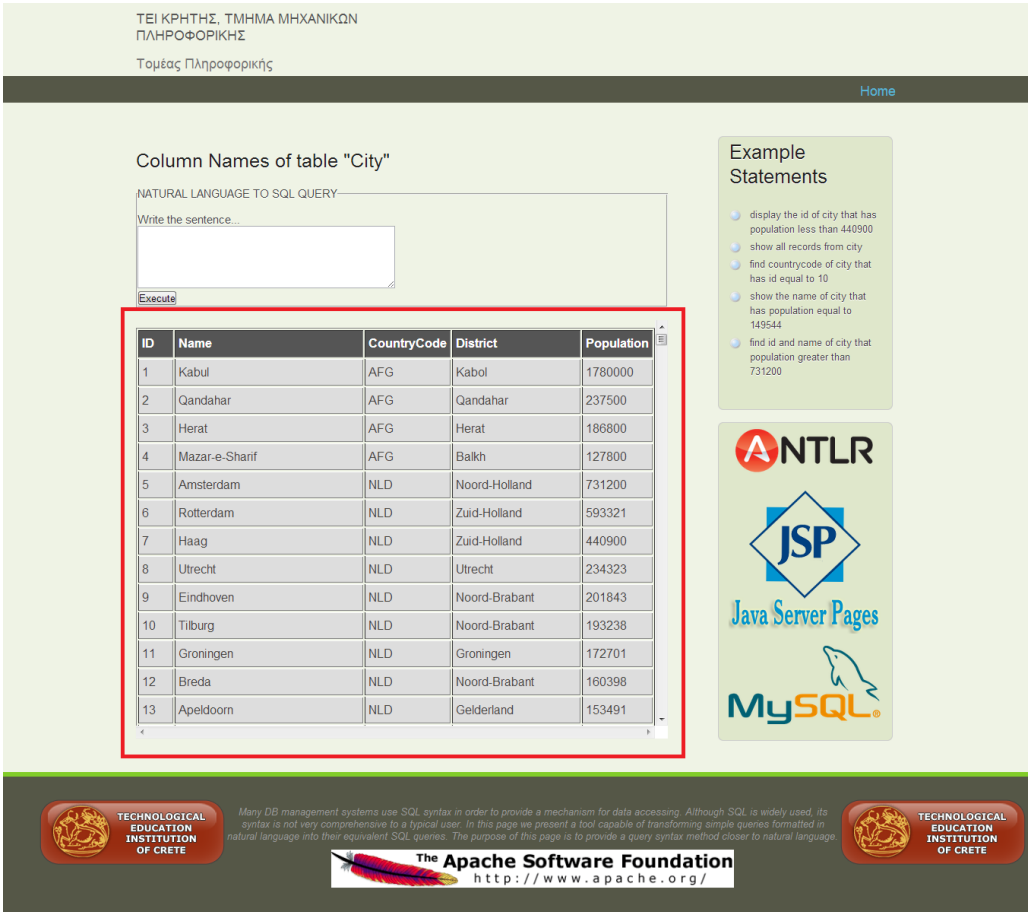
Εικόνα 31: Αποτέλεσμα εκτέλεσης ερωτήματος

Αναλυτικότερα στο πεδίο “TextArea” ο χρήστης πληκτρολογεί διατυπωμένη στα αγγλικά την πρόταση του προς το σύστημα σύμφωνα πάντα με τους περιορισμούς που έχουμε θέσει σε αυτό οι οποίοι αναλύονται στο κεφάλαιο 4.2. Ο χρήστης προκείμενου να αναφερθεί σε ένα στοιχείο ενός πίνακα της βάσης δεδομένων πρέπει να το γράψει ακριβώς με το ίδιο όνομα που αναφέρεται στο πεδίο της συγκεκριμένης βάσης δεδομένων. Για να αναφερθεί σε έναν πίνακα, ο χρήστης πρέπει να αναφέρει το όνομα του πίνακα όπως ακριβώς αναφέρεται στη βάση δεδομένων. Δίπλα στο πεδίο “Example Statements” μπορεί να δει παραδείγματα αποδεκτών προτάσεων.



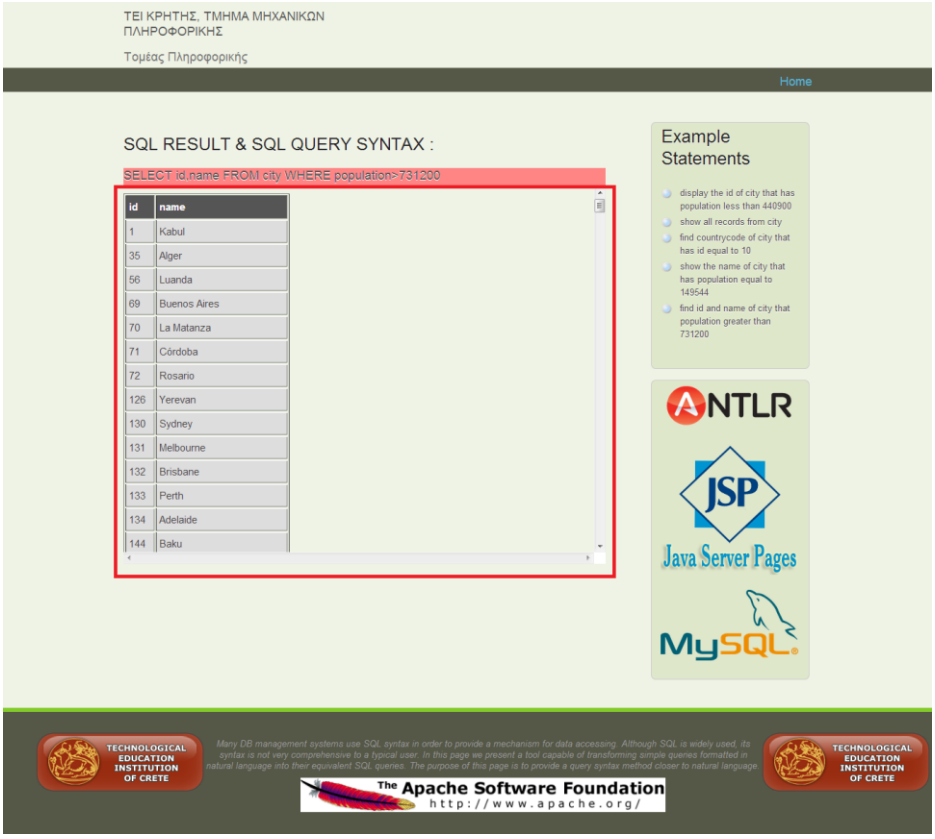
Εικόνα 32: Διεπαφή εφαρμογής, πεδίο input

Στο παρακάτω εικόνα φαίνεται η βάση δεδομένων με τα πεδία που έχουμε συνδέσει στο σύστημα μας και στην οποία κάνει την ερώτηση ο χρήστης. Ισχύουν και εδώ οι περιορισμοί που έχουμε θέσει στο σύστημα μας.

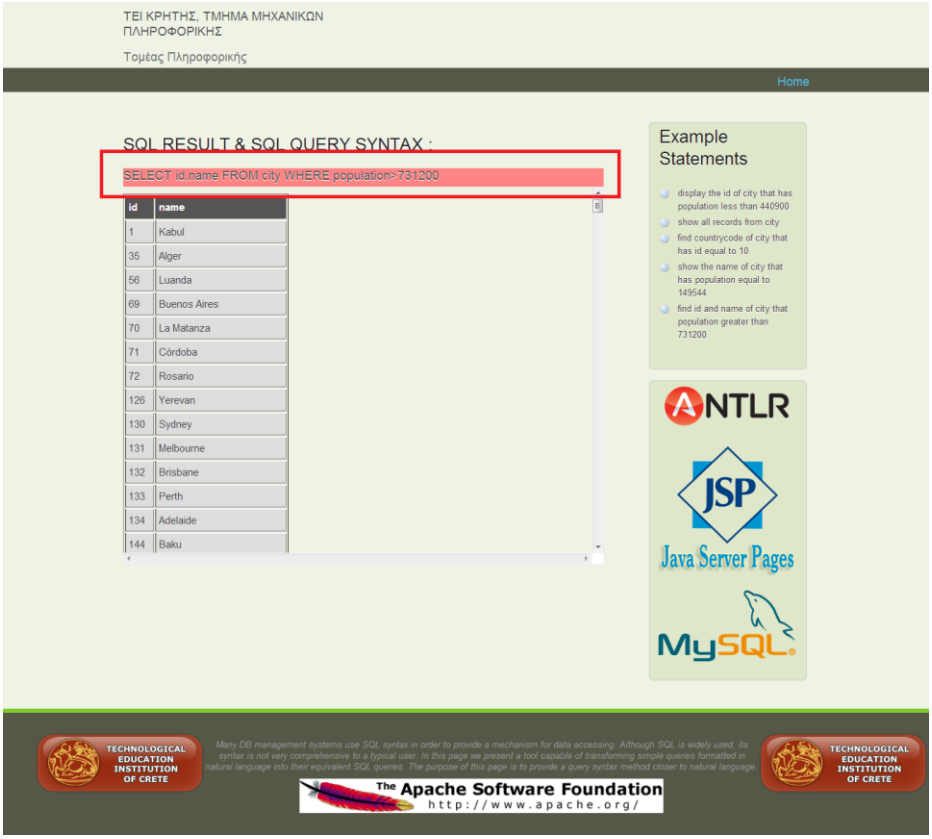


Εικόνα 33: Διεπαφή εφαρμογής, Βάση Δεδομένων

Στης επόμενες δύο εικόνες βλέπουμε το αποτέλεσμα της εκτέλεσης του ερωτήματος του χρήστη, δηλαδή τα πεδία που επέλεξε να του εμφανίσει σαν έξοδο η εφαρμογή καθώς και το αποτέλεσμα της γλώσσας SQL.

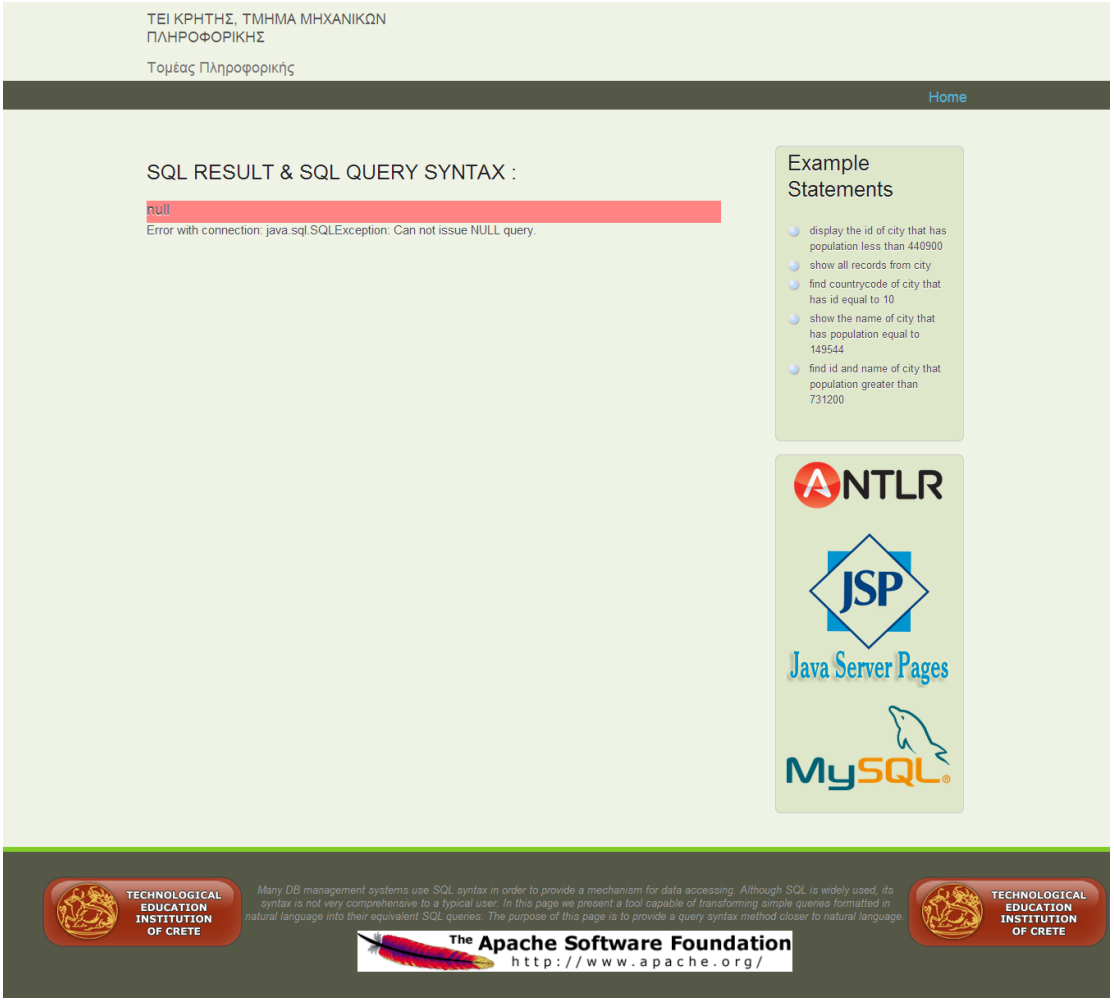


Εικόνα 34: Αποτέλεσμα ερωτήματος Χρήστη



Εικόνα 35: Αποτέλεσμα SQL ερωτήματος χρήστη

Σε περίπτωση που ο χρήστης δώσει μια πρόταση διατυπωμένη λάθος, με τον όρο λάθος εννοούμαι μια πρόταση που δεν συμφωνεί με το συντακτικό της γραμματικής της οποίας έχουμε ορίσει και δέχεται το σύστημα, τότε η έξοδος της διεπαφής θα είναι η παρακάτω. Οπότε ο χρήστης πρέπει να επιστρέψει στην αρχική σελίδα τις εφαρμογής και να μελετήσει τα προτεινόμενα παραδείγματα εκτέλεσης ερωτήματος.



Εικόνα 36: Έξοδος μη αποδεκτής πρότασης

6 Συμπεράσματα

Στα πλαίσια της πτυχιακής αυτής αναπτύχθηκε μια εφαρμογή η οποία επιτρέπει σε έναν απλό χρήστη χωρίς ιδιαίτερες γνώσεις προγραμματισμού ή γλώσσα βάσεων δεδομένων να εξάγει δεδομένα από μια βάση δεδομένων. Η διαδικασία αυτή βασίζεται στην φυσική γλώσσα που χρησιμοποιεί ο χρήστης για να διατυπώσει τα ερωτήματα η οποία μοιάζει στο συντακτικό με την γλώσσα ερωτήσεων SQL.Η πτυχιακή αποτελείται από δύο επίπεδα. Ένα είναι το κομμάτι του σερβερ το οποίο αφού πάρει το ερώτημα από το χρήστη αναλαμβάνει να το μετατρέψει σε SQL ερώτημα μέσα από τη διαδικασία της λεκτικής και συντακτικής ανάλυσης σύμφωνα πάντα με τη γραμματική που έχουμε ορίσει στο σύστημα μας. Το δεύτερο κομμάτι αποτελεί το γραφικό περιβάλλον, το οποίο στοχεύει στο χρήστη και έχει γίνει όσο πιο απλό γίνεται. Αποτελείται από ένα πεδίο στο οποίο ο χρήστης γράφει το ερώτημα του και ένα ακόμα πεδίο στο οποίο εμφανίζεται η βάση δεδομένων με την οποία θέλει να αλληλεπίδραση ο χρήστης, αλλά και να του εμφανίσει το αποτέλεσμα.

Όπως έχουμε αναφέρει και στα προηγούμενα κεφάλαια, το σύστημα μας λειτουργεί μόνο για απλές εκφράσεις της sql γλώσσας. Για τη λειτουργία του συστήματος ακολουθήθηκαν κάποιες ειδικές τεχνικές, που το σύστημα με βάση αυτά, μπορεί να αναπτυχθεί για την κάλυψη περισσότερων και πιο πολύπλοκων εκφράσεων. Λόγω της φύσης της γραμματικής που έχουμε ορίσει το σύστημα μας είναι ικανό να υποστηρίξει εύκολα και άλλες γλώσσες, εκτός από αυτήν που έχουμε υλοποίησει, αλλά και επίσης μπορεί να καλύψει και ακόμα περισσότερες εκφράσεις. Οπότε εμφανίζεται το θέμα της επέκτασης του συστήματος μας, πράγμα που σημαίνει ότι είναι αρκετά εύκολο για κάποιον, με τις απαραίτητες γνώσεις να επεκτείνει την εφαρμογή ώστε να καλύψει την επιθυμητή γλώσσα που θέλει ο ίδιος.

Το παρόν σύστημα που υλοποιήσαμε σε αυτή την πτυχιακή μπορεί να φανεί χρήσιμο σε αρκετές περιπτώσεις. Σκεφτείτε μια εταιρία η οποία διαθέτει μεγάλες βάσεις δεδομένων και τους υπαλλήλους της να αλληλεπιδρούν με αυτές. Κανονικά θα έπρεπε να εκπαιδεύει τους υπαλλήλους της ώστε να μάθουν να χρησιμοποιούν τη γλώσσα ερωτήσεων SQL ή να προσλαμβάνει κάποιους που ήδη την γνωρίζουν. Όλα αυτά είναι χρονοβόρα και επιπλέον κοστίζουν στην επιχείρηση. Ένα παρόμοιο σύστημα με αυτό που δημιουργήσαμε θα μπορούσε εύκολα να χρησιμοποιηθεί από έναν υπάλληλο ή και απλό χρήστη ώστε να κάνει εύκολα και γρήγορα τη δουλειά του, γλιτώνοντας έτσι την εταιρία από περαιτέρω έξοδα.

7 Βιβλιογραφία

[1] Nikos Papadakis and Pavlos Kefalas , A tool for access to Relational Databases in natural language , Department of Sciences, Techological Educational Institute of Crete

[2] Natural Language Query Processing Tom Harrison et al. Application No- 10/251,929 Patent No- US7,403,938,B2 Date of Patent- 22 July 2008

[3] NATURAL LANGUAGE TO SQL CONVERSION SYSTEM ANIL M. BHADGALE, SANHITA R. GAVAS, MEGHANA M. PATIL & PINKI R. GOYAL PVG‟S COET , Pune, Maharashtra, India

[4] Using Natural Language Processing in Order to Create SQL Queries F.Siasar djahantighi, M.Norouzifard, S.H.Davarpanah, M.H.Shenassa Islamic Azad University science and research Branch, Tehran, Iran University of Science and Culture, Iran K.N.Toosi University of Technology, Iran

[5] NLP TOKEN MATCHING ON DATABASE USING BINARY SEARCH Ekta Agrawal Shreeja Nair Department of of Computer Science & Engineering Department of of Computer Science & Engineering Oriental Institute of Technology, Bhopal Oriental Institute of Technology, Bhopal

[6] A Natural Language Database Interface For SQL-Tutor 5th November 1999 Honours Project by Seymour Knowles Supervisor: Tanja Mitrovic

[7] Natural Language Database Interface Afya Nahas Dhankawadi, Pune 411 043, India

[8] Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked Alessandra Giordani and Alessandro Moschitti Computer Science and Engineering Department University of Trento, Povo (TN), Italy

[9] Natural Language Database Interface for Selection of Data Using Grammar and Parsing N. D. Karande, and G. A. Patil

[10] ANTLR
http://www.placidsystems.com/articles/article-grammarlayout/grammarLayout.htm
http://www.antlr.org/wiki/display/ANTLR3/ANTLR+3+Wiki+Home
http://meri-stuff.blogspot.gr/2011/08/antlr-tutorial-hello-word.html
http://www.javacodegeeks.com/2012/06/antlr-getting-started.html
http://antlr3.org/api/Java/index.html

[11] BNF grammar

http://www.cs.utsa.edu/~wagner/CS3723/grammar/examples2.html

[12] NLP Tutorial
http://www.programcreek.com/2012/05/opennlp-tutorial/

[13] SQL
http://www.w3schools.com/sql/
http://blogs.sch.gr/giannopk/files/2010/12/phpmysql.pdf

[14] Java
http://docs.oracle.com/javase/7/docs/
http://www.islab.demokritos.gr/gr/html/ptixiakes/kostas-aris_ptyxiakh/Phtml/java.htm

[15] Tomcat
http://opensci.grnet.gr/os_catalog/softwares/apache-tomcat

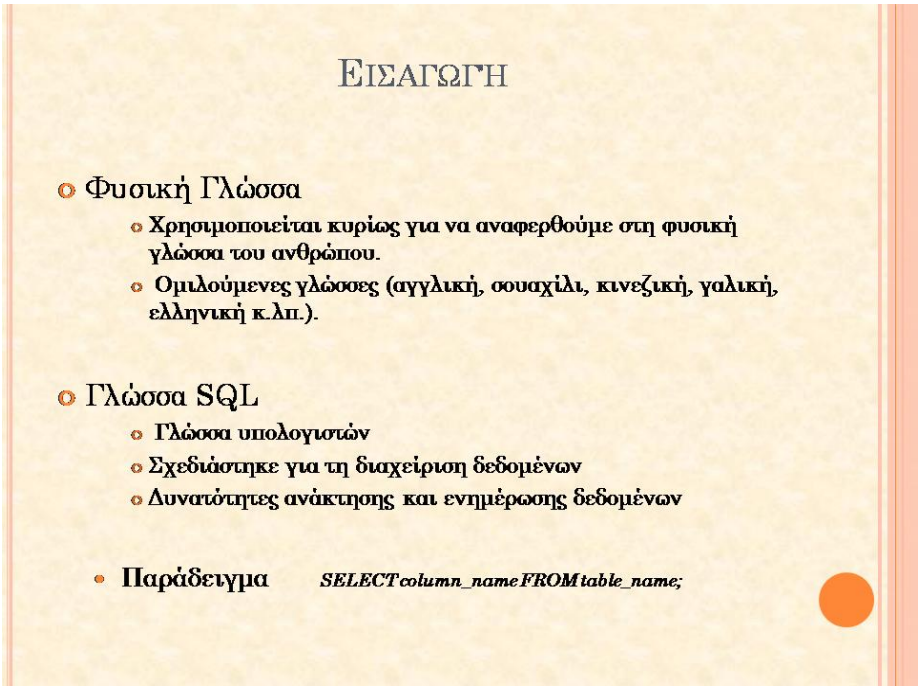
[16] NetBeans
http://opensci.grnet.gr/os_catalog/softwares/netbeans
http://www.cs.uoi.gr/~zarras/se-notes/NetBeansPresentation.pdf

[17] JSP
http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-JSP-1-Introductiion.html
http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-JSP-1-Introductiion.html

[18] CSS
http://pages.cs.aueb.gr/courses/epl131/files/CSS_notes.pdf

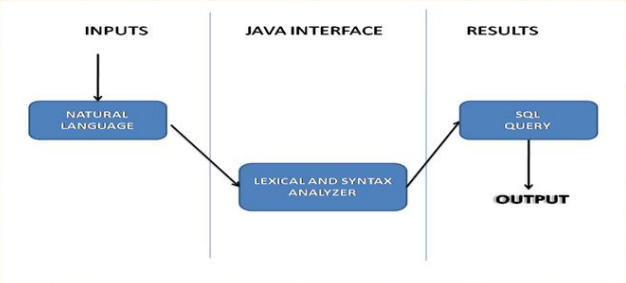
8 Παράρτημα

Ακολουθούν οι διαφάνειες παρουσίασης του PowerPoint.



ΣΤΟΧΟΙ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

- Δημιουργία μιας διαεπαφής.
- Είσοδος: Ερώτημα του χρήστη διατυπωμένο σε φυσική γλώσσα.
- Επεξεργασία.
- Έξοδος: Αποτέλεσμα ερωτήματος σε SQL γλώσσα.



ΤΕΧΝΟΛΟΓΙΕΣ ΥΛΟΠΟΙΗΣΗΣ

- **Επίπεδο Γραμματικής**
- **ANTLR**
- **ANTLRworks**

- **Επίπεδο Εφαρμογής**
- **MySQL**
- **XAMPP**
- **Apache Tomcat**
- **JAVA**
  - **NetBeans**
  - **JSP**
  - **Servlets**
  - **JDBC**
- **CSS**

ΚΕΝΤΡΙΚΉ ΙΔΈΑ

- Δημιουργία μιας γραμματικής που ακολουθεί το εξής μοτίβο.

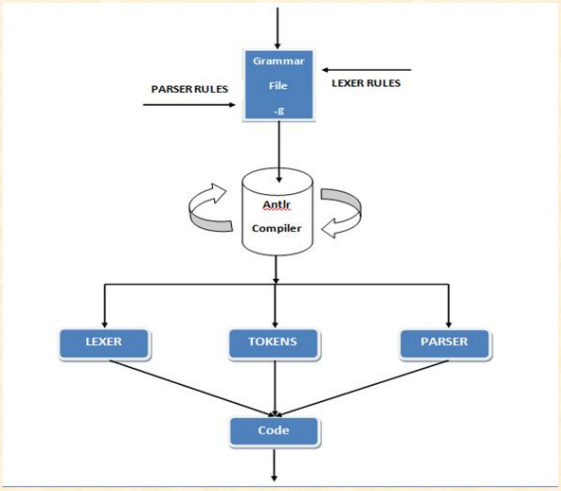
```
Grammar   : [show | list | display] (me) (all) (our) <table_name>
SQL       : SELECT * FROM <table_name>

Grammar   : show (the) <column_name> of (our) <table_name>
SQL       : SELECT <column_name> FROM <table_name>
```

```
statemens : select

  select :  ask /(me)?/ /(all)?/ /(our)?/ table_name
            {"SELECT * FROM table_name}

          | ask /(pe)?/ column_name /of/ /(our)?/ table_name [column_name, table_name]
          {"SELECT column_name  FROM table_name"}

  ask : show | list | display

table_name : TABLES

column_name : COLUMNS
```

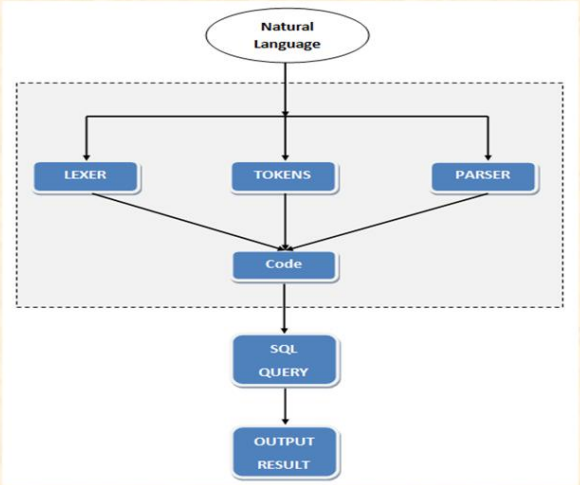
ΧΑΡΑΚΤΗΡΙΣΤΙΚΆ ΓΡΑΜΜΑΤΙΚΉΣ

- Μια γραμματική αποτελείτε από:
  - Λεκτικούς κανόνες: Αρχίζουν με κεφαλαίο γράμμα
  - Συντακτικούς κανόνες: Αρχίζουν με πεζά γράμματα.
- Διαδικασία Μεταγλώττισης Γραμματικής
  - Λεκτική Ανάλυση.
  - Λεκτικές Μονάδες.
  - Συντακτική Ανάλυση.
  - Σημασιολογική Ανάλυση.
  - Παραγωγή Τελικού κώδικα.

Διαδικασία μεταγλώττισης



Τελικό στάδιο μεταγλώττισης



ΣΤΙΓΜΙΌΤΥΠΟ ΓΡΑΜΜΑΤΙΚΉΣ

Parser Rules

```
program: stmt+
;

stmt: FIND UNIQUE ID rest
| FIND ALL (ID)? FROM ID
| FIND ALL (ID)? FROM ID rest
| FIND (ID)? ID rest
;

rest: FROM ID thatstmt
| FROM ID
| AND ID rest
| AND ID thatstmt
;

thatstmt: THAT (ID)?  ID opstmt
;
```

Lexer Rules

```
FIND : ('find' | 'display' | 'show');

FROM : ('from' | 'of');
AND : ('and');
COMMA : (',');

THAT : ('that' | 'whose');
ALL: ('all');

ID :
        ('a'..'z' | 'A'..'Z' | '_')('a'..'z' | 'A'..'Z'
    | '0'..'9' | '_')*
;
```

ΑΠΟΔΕΚΤΉ ΠΡΌΤΑΣΗ

- Η γενική μορφή της πρότασης είναι:
  - **[Find | show | displace] *sentence1* [from | of] *sentence2* [that | where | whose] *sentence3***

1- display the id of city that has population less than 440900	1- SELECT id FROM city WHERE population<440900
2- show all records from city	2- SELECT * FROM city
3- find countrycode of city that has id equal to 10	3- SELECT countrycode FROM city WHERE id='10'


ΤΕΙ ΚΡΗΤΗΣ, ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
Τομέας Πληροφορικής
Home
Column Names of table "City"
NATURAL LANGUAGE TO SQL QUERY
Write the sentence...
Execute
Example Statements
display the id of city that has population less than 440900
show all records from city
find countrycode of city that has id equal to 10
show the name of city that has population equal to 149544
find id and name of city that population greater than 731200
ID Name CountryCode District Population
1 Kabul AFG Kabol 1780000
2 Qandahar AFG Qandahar 237500
3 Herat AFG Herat 186800
4 Mazar-e-Sharif AFG Balkh 127800
5 Amsterdam NLD Noord-Holland 731200
6 Rotterdam NLD Zuid-Holland 593321
7 Haag NLD Zuid-Holland 440900
8 Utrecht NLD Utrecht 234323
9 Eindhoven NLD Noord-Brabant 201843
10 Tilburg NLD Noord-Brabant 193238
11 Groningen NLD Groningen 172701
12 Breda NLD Noord-Brabant 160398
13 Apeldoorn NLD Gelderland 153491
ANTLR
JSP
Java Server Pages
MySQL
TECHNOLOGICAL EDUCATION INSTITUTION OF CRETE
The Apache Software Foundation
http://www.apache.org/
Φυσική Γλώσσα
Είσοδος του χρήστη
Βάση Δεδομένων
Συνδεμένη Στο Σύστημα μας


ΤΕΙ ΚΡΗΤΗΣ, ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
Τομέας Πληροφορικής
Home
SQL RESULT & SQL QUERY SYNTAX :
SELECT id,name FROM city WHERE population>731200
id name
1 Kabul
35 Alger
56 Luanda
69 Buenos Aires
70 La Matanza
71 Córdoba
72 Rosario
126 Yerevan
130 Sydney
131 Melbourne
132 Brisbane
133 Perth
134 Adelaide
144 Baku
Example Statements
display the id of city that has population less than 440900
show all records from city
find countrycode of city that has id equal to 10
show the name of city that has population equal to 149544
find id and name of city that population greater than 731200
ANTLR
JSP
Java Server Pages
MySQL
TECHNOLOGICAL EDUCATION INSTITUTION OF CRETE
The Apache Software Foundation
http://www.apache.org/
SQL Αποτέλεσμα
Ερωτήματος Χρήστη
Αποτέλεσμα
Ερωτήματος Χρήστη

Επεκτάσεις

- Το σύστημα είναι επεκτάσιμο
  - **Μπορεί να καλύψει επιπλέον λέξεις και φράσεις πολύ εύκολα.**
  - **Λόγω της φύσεως της γραμματικής μπορεί να καλύψει περισσότερες από μια γλώσσες.**
- Εφαρμογές
  - **Γρήγορη εξαγωγή αποτελεσμάτων για αρχάριους χρήστες.**
  - **Εταιρίες χωρίς εξειδικευμένους χρήστες.**
    - **Πρόσληψη χωρίς ιδιαίτερα προσόντα (γλώσσες προγραμματισμού).**
    - **Μείωση κόστους εκπαίδευσης.**

Τέλος Παρουσίασης

Σας ευχαριστώ πολύ

Σιρανίδης Κώστας

TECHNOLOGICAL EDUCATION INSTITUTION OF CRETE