ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ

ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Εργαλείο μέτρησης ετεροαναφορών βασισμένο στον Google Scholar

Φοιτητής: Παλόλλης Γιώργος

Εξεταστική Επιτροπή

Επ.Καθηγητής Δεληγιαννάκης Αντώνιος (επιβλέπων)

Καθηγητής Χριστοδουλάκης Σταύρος

Καθηγητής Γαροφαλάκης Μίνως

Χανιά 2012

Ευχαριστίες

Θα ήθελα πάνω απ' όλα να ευχαριστήσω την οικογένειά μου για την συνεχή και αμέριστη συμπαράστασή τους, την υλική και κυρίως την ψυχολογική στήριξή τους καθ' όλη την διάρκεια των προπτυχιακών σπουδών μου, η οποία συνέβαλε καθοριστικά στην αποπεράτωση τους.

Επίσης θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Δεληγιαννάκη Αντώνιο για την επίβλεψη της διπλωματικής μου εργασίας, καθώς και τους Καθηγητές κ. Χριστοδουλάκη Σταύρο και κ. Γαροφαλάκη Μίνω για την συμμετοχή τους στην εξεταστική επιτροπή.

Περίληψη

Στα πλαίσια της παρούσας διπλωματικής εργασίας αναπτύχθηκε ένα σύστημα το οποίο έχει σαν βασική λειτουργία την εύρεση συγκεκριμένου αριθμού αναφορών (citations) που υπάρχουν σε επιστημονικές δημοσιεύσεις (papers) που είναι αναρτημένες στον Google Scholar. Η ιδιαιτερότητα σε αυτόν τον αριθμό είναι πως δεν συνυπολογίζονται οι αυτοαναφορές (self-citations), δηλαδή οι αναφορές που προέρχονται από δημοσιεύσεις των ίδιων των συγγραφέων της πρώτης δημοσίευσης.

Επίσης η εφαρμογή έχει την δυνατότητα να βρει τις αναφορές που γίνονται στις δημοσιεύσεις ενός συγγραφέα σε συγκεκριμένο χρονικό εύρος (πχ οι αναφορές που έχει ο συγγραφέας Α τα έτη 2001 έως και σήμερα).

Τέλος υπάρχει η δυνατότητα εμφάνισης όλων των αυτοαναφορών (self-citations) και ετεροαναφορών (non self citations) για την κάθε δημοσίευση των αποτελεσμάτων ξεχωριστά και στις δύο παραπάνω κατηγορίες.

Αντίθετα, η εμφάνιση των αποτελεσμάτων στην αναζήτηση που προσφέρει το Google Scholar (scholar.google.com) δεν περιλαμβάνει την μη συμπερίληψη των ετεροαναφορών στον υπολογισμό του συνόλου των αναφορών καθώς και δεν υποστηρίζει αναζήτηση αναφορών σε εύρος χρόνων.

Περιεχόμενα

Εισαγωγή

Τα τελευταία χρόνια έχουν αναπτυχθεί αρκετές μηχανές αναζήτησης που περιέχουν βάσεις δεδομένων με πληροφορίες για τα επιστημονικά (και όχι μόνο) δημοσιεύματα. Δυο-τρεις απ' αυτές έχουν εκτεταμένη, σχεδόν ολική, κάλυψη όλων των επιστημονικών κλάδων. Κάποιες από αυτές είναι ελεύθερες για κάθε ενδιαφερόμενο, ενώ άλλες ανοίγουν τα περιεχόμενά τους μόνο σε συνδρομητές (πανεπιστήμια, ερευνητικά κέντρα κ.ά.).

Μία από τις παραπάνω μηχανές αναζήτησης που είναι ελεύθερα προσβάσιμη στο διαδίκτυο είναι αυτή του Google Scholar, τα αποτελέσματα αναζήτησης του οποίου χρησιμοποιήσαμε στην υλοποίηση τους συστήματος που διαπραγματεύεται η παρούσα διπλωματική εργασία.

Τα θέματα που τέθηκαν προς επίλυση στην εργασία αυτή είναι δύο:

1ον- Ο υπολογισμός και η εμφάνιση των:

- ετεροαναφορών, δηλαδή των αναφορών στον ζητούμενο επιστημονικό συγγραφέα χωρίς όμως να λαμβάνονται υπόψη οι αναφορές που έχει κάνει ο συγγραφέας σε δημοσιεύσεις του,
- αυτοαναφορών, δηλαδή του συνόλου των αναφορών που προέρχονται από τον ίδιο τον ζητούμενο συγγραφέα,
- συνόλου των αναφορών, δηλαδή του αθροίσματος των αυτοαναφορών και των ετεροαναφορών

επί του συνόλου των δημοσιεύσεων αλλά και για την κάθε δημοσίευση των αποτελεσμάτων ξεχωριστά.

2ον - Η εύρεση των αναφορών που έχουν γίνει στις δημοσιεύσεις του ζητούμενου επιστημονικού συγγραφέα σε ένα εύρος ετών, όπως για παράδειγμα: υπολογισμός και εμφάνιση του συνόλου των αναφορών που έχει ο συγγραφέας Α από το έτος 2005 μέχρι το 2011.
Στα αποτελέσματα που προκύπτουν μπορούν να εφαρμοστούν τα ερωτήματα και να υπολογιστούν τα δεδομένα του προηγούμενου βήματος.

Τα παραπάνω είναι χρήσιμα καθώς με αυτόν τον τρόπο γίνεται μια πιο έγκυρη αξιολόγηση των επιστημονικών συγγραφέων αλλά και των δημοσιεύσεών τους, καθώς δείχνουν τον πραγματικό των αριθμό των αναφορών.

Ένας παραπάνω λόγος επίσης είναι πως καμία μηχανή αναζήτησης δεν εκτελεί τις δύο λειτουργιές που προαναφέρθηκαν.
Το Google Scholar, στην περίπτωσή μας, υπολογίζει αποκλειστικά τον συνολικό αριθμό των αναφορών χωρίς να ξεχωρίζει αυτοαναφορές και ετεροαναφορές.

Ενώ όσον αφορά το χρονικό εύρος αναζήτησης, αυτή γίνεται μόνο για τις δημοσιεύσεις που έχουν συγγραφεί από τον ζητούμενο επιστημονικό συγγραφέα και όχι για τις αναφορές που έχει επί των δημοσιεύσεών του.

Απαιτήσεις από το σύστημα του χρήστη

Για να εκτελεστεί το σύστημα της παρούσας διπλωματικής εργασίας, ο χρήστης θα πρέπει να έχει εγκατεστημένα στο λειτουργικό του σύστημα:

- το Σύστημα Διαχείρισης Βάσεων Δεδομένων MySQL,
- έναν χρήστη στην MySQL ο οποίος θα έχει δικαιώματα δημιουργίας σχήματος, στην βάση δεδομένων (CREATE SCHEMA), δημιουργίας πινάκων (CREATE TABLE), διαγραφής περιεχομένων πινάκων (TRUNCATE TABLE), δημιουργίας ξένων κλειδιών (CONSTRAINT FOREIGN KEY) και τέλος να μπορεί να ενεργοποιεί/απενεργοποιεί τους ελέγχους για τα ξένα κλειδιά (FOREIGN_KEY_CHECKS=0/1)
- το Java Runtime Environment (JRE),
- σταθερή σύνδεση στο διαδίκτυο,
- προαιρετικά, το οπτικό εργαλείο MySQL Workbench
- και κυρίως, αρκετή υπομονή.

Απαιτήσεις από την εφαρμογή

Οι απαιτήσεις από την εφαρμογή είναι οι εξής:

1. Δημιουργία της βάσης δεδομένων και των πινάκων που θα χρειαστούν για την αποθήκευση των δημοσιεύσεων, των αναφορών και των συγγραφέων τους
2. Η εφαρμογή να κρατάει στην βάση δεδομένων τα αποτελέσματα ενός συγγραφέα την φορά
3. Εμφάνιση της κάθε δημοσίευσης του επιστημονικού συγγραφέα που εισήχθη προς αναζήτηση και των στοιχείων της αναλυτικά, περιλαμβάνοντας τα παρακάτω:
    - τίτλος,
    - συγγραφείς,
    - που εκδόθηκε,
    - έτος έκδοσής,
    - συνολικό αριθμό των αναφορών καθώς και
    - συνολικό αριθμό των ετεροαναφορών

4. Εμφάνιση συνολικού αριθμού δημοσιεύσεων του εισαχθέντος επιστημονικού συγγραφέα καθώς επίσης του συνολικού αριθμού των αναφορών και του συνολικού αριθμού των ετεροαναφορών
5. Για την εκάστοτε δημοσίευση να εμφανίζεται ποιες και πόσες είναι οι αναφορές, οι αυτοαναφορές και οι ετεροαναφορές σε αυτήν
6. Εμφάνιση των δημοσιεύσεων που έχουν αναφορές σε ένα συγκεκριμένο εύρος ετών, υποστηρίζοντας κάθε μία από τις λειτουργίες 3 έως 5

Μεθοδολογία υλοποίησης του συστήματος

Η μεθοδολογία που ακολουθήθηκε για την υλοποίηση του συστήματος χωρίστηκε σε τέσσερεις φάσεις:

1. *Συνομιλία με έναν τελικό χρήστη της εφαρμογής*
   Σε αυτή την φάση καθορίστηκαν οι απαιτήσεις και όλες οι παράμετροι που θα θέλαμε να ικανοποιεί η εφαρμογή μας.

2. *Σχεδιασμός του συστήματος*
   Εδώ πραγματοποιήθηκε ένας αρχικός σχεδιασμός που αφορά τον τρόπο υλοποίησης του συστήματος με βάση τις απαιτήσεις που προέκυψαν από το προηγούμενο βήμα.
   Όπως είναι φυσιολογικό, είναι η φάση η οποία καθ' όλη την διάρκεια της ανάπτυξης δεχόταν τις μεγαλύτερες αλλαγές, ιδιαίτερα μετά το βήμα της ανάδρασης.

3. *Υλοποίηση του συστήματος*
   Με βάση τον παραπάνω σχεδιασμό προχωρούσε και η υλοποίηση του συστήματος. Λόγω αυτής της στενής σχέσης Σχεδιασμού – Υλοποίησης, και αυτή η φάση δεχόταν αρκετές αλλαγές κατά την διάρκεια της ανάπτυξης του συστήματος.

4. *Ανάδραση*
   Στην συγκεκριμένη φάση η εφαρμογή ήταν έτοιμη να περάσει στην στους τελικούς χρήστες για δοκιμές και ανάλογα με τα σχόλιά τους να γίνουν οι απαραίτητες αλλαγές.
   Η συγκεκριμένη φάση ήταν η πιο σημαντική και η πιο ουσιαστική καθώς αναγνωρίστηκαν τα σημεία του σχεδιασμού και της υλοποίησης του συστήματος που έχριζαν βελτίωση.
   Πέραν του επιβλέποντα καθηγητή (κ.Δεληγιαννάκη) το σύστημα δοκιμάστηκε από δύο μεταπτυχιακούς φοιτητές του Πανεπιστημίου Πειραιώς και έναν προπτυχιακό τελειόφοιτο φοιτητή του Πολυτεχνείου Κρήτης. Όλοι τους ήταν γνώστες του Google Scholar και οι τρεις από του τέσσερεις χρησιμοποιούσαν τις υπηρεσίες του σε τακτική βάση.
   Η συμβολή τους ήταν καθοριστική στην βελτιστοποίηση της εφαρμογής καθώς κατά την διάρκεια αυτής της φάσης και με βάση τα σχόλια και της ιδέες τους που συγκέντρωσα, πραγματοποιήθηκαν κρίσιμες αλλαγές κυρίως στο γραφικό περιβάλλον της διεπαφής χρήστη και στην βάση δεδομένων.

   Στην διεπαφή χρήστη οι αλλαγές που προέκυψαν είχαν να κάνουν:
   - με τον τρόπο εμφάνισης των αναφορών των δημοσιεύσεων,

- την τοποθέτηση μπάρας που θα δείχνει την πρόοδο της αναζήτησης,
- την τοποθέτηση κουμπιού που θα σταματούσε την αναζήτηση εφόσον το επιθυμούσε ο χρήστης και
- την εμφάνιση αποτελεσμάτων καθώς εκτελείτε η αναζήτηση.

Ενώ στην βάση δεδομένων η αλλαγές που προέκυψαν είναι:

- προσθήκη επιπλέον γνωρισμάτων για να είναι το περιεχόμενο των πινάκων πιο πλήρες (πχ προσθήκη του γνωρίσματος ¨Έτος δημοσίευσης¨ στον πίνακα με τις αναφορές)
- δημιουργία δύο νέων πινάκων (paperAuthors: πίνακας που κρατάει τον κάθε συγγραφέα της εκάστοτε δημοσίευσης ξεχωριστά και citationAuthors: πίνακας που κρατάει τον κάθε συγγραφέα της εκάστοτε αναφοράς ξεχωριστά) που τελικώς ήταν απαραίτητοι στην εξαγωγή/αναγνώριση των αυτοαναφορών και των ετεροαναφορών.

Προβλήματα που παρουσιάστηκαν και αντιμετωπίστηκαν

Κατά την υλοποίηση της εφαρμογής παρουσιάστηκαν ορισμένα προβλήματα που έχουν να κάνουν κυρίως με τις συνδέσεις στο διακομιστή του Google Scholar και στην εμφάνιση των αποτελεσμάτων στην επιθυμητή για εμάς μορφή.

Το πρώτο πρόβλημα που αντιμετωπίστηκε είχε να κάνει με το γεγονός πως το Google Scholar εξ'ορισμού δεν έχει ενεργοποιημένη την επιλογή (σύνδεσμο) για την εμφάνιση των στοιχείων των δημοσιεύσεων στην μορφή BibTex στις σελίδες με τα αποτελέσματα. Για να γίνει αυτό θα πρέπει να ενεργοποιηθεί η αντίστοιχη επιλογή από τις ρυθμίσεις του.
Με αναζήτηση σε διάφορα φόρουμ βρήκαμε πως για την περίπτωση του συστήματός μας το πρόβλημα λύνεται προσθέτοντας την κατάληξη ¨:CF=4¨ στο τέλος των cookies* που στέλνουμε με το κάθε αίτηση στον διακομιστή του Google Scholar.
Με αυτόν τον τρόπο εμφανίζεται ο σύνδεσμος που οδηγεί στην εξαγωγή των στοιχείων των δημοσιεύσεων στην ζητούμενη μορφή για την εφαρμογή μας.

Ένα άλλο πρόβλημα που αντιμετωπίσαμε, αφορά τα ονόματα των συγγραφέων στις δημοσιεύσεις.
Πολλές φορές οι συγγραφείς πέραν του επωνύμου και του πρώτου αρχικού του ονόματός τους, πρόσθεταν κι άλλα αρχικά, όπως για παράδειγμα το όνομα του κ.Σιδηρόπουλου, καθηγητή του Πολυτεχνείου Κρήτης, σε ορισμένες δημοσιεύσεις του είναι γραμμένο ως Σιδηρόπουλος Ν. ενώ σε άλλες ως Σιδηρόπουλος Ν.Δ.
Το αποτέλεσμα ήταν να μην μπορεί να γίνει ακριβής και σωστή σύγκριση για την εξαγωγή των αυτοαναφορών.
Στο σύστημα μας για να λυθεί αυτό το πρόβλημα, διώχνουμε ό, τι υπάρχει πέραν του πρώτου αρχικού γράμματος μετά το επώνυμου του συγγραφέα.
Με αυτόν τον τρόπο η σύγκριση γίνεται περισσότερο ακριβής απ'ότι ήταν πριν.

Το βασικότερο πρόβλημα που αντιμετωπίστηκε έχει να κάνει με τις συνδέσεις στον διακομιστή του Google Scholar.
Συγκεκριμένα, όταν ο διακομιστής αυτός δέχεται μεγάλο αριθμό αιτημάτων, όπως στην περίπτωσή μας**, θεωρεί πως δέχεται επίθεση από κακόβουλο λογισμικό με αποτέλεσμα να κλειδώνει την IP για κάποιο χρονικό διάστημα που συνήθως είναι από μία μέχρι τρεις ημέρες ανάλογα με το πόσο ¨ύποπτη¨ είναι η συγκεκριμένη IP, δηλαδή αν έχει προκαλέσει το ίδιο πρόβλημα στο άμεσο παρελθόν.
Ό έλεγχος που γίνεται από τον παραπάνω διακομιστή είναι διπλός.
Πρώτα ελέγχει το cookie του κάθε request και εάν δει πως είναι το ίδιο κάθε φορά κλειδώνει το συγκεκριμένο cookie και ουσιαστικά κλειδώνει και την εφαρμογή μας.

Ο δεύτερος έλεγχος έχει να κάνει με την IP που έχει ο υπολογιστής που εκτελεί την εφαρμογή μας. Αν τα requests είναι πολλά, ακόμα και με διαφορετικό cookie, ελέγχεται η IP και εφόσον προέρχονται από την ίδια, τότε την κλειδώνει.

Οι λύσεις για το παραπάνω ζήτημα ήταν δύο:

- Αφού κατέγραψα την μορφή του cookie* που αποθηκεύει στον φυλλομετρητή μου ο διακομιστής του Google Scholar, προχώρησα στην υλοποίηση συνάρτησης η οποία φτιάχνει αντίστοιχα cookies. Έτσι σε κάθε request στέλνω διαφορετικό cookie.
  Η λύση αυτή, όπως περιγράψαμε παραπάνω, δεν λύνει το πρόβλημα ωστόσο ¨ξεκλειδώνει¨ την εφαρμογή εφόσον ο χρήστης αλλάξει την IP του ή την εκτελέσει σε υπολογιστή που έχει διαφορετική σύνδεση στο διαδίκτυο.
  Στην αντίθετη περίπτωση κατά τη οποία δεν ανανεώνεται και η εφαρμογή έχει το ίδιο πάντα cookie, η εφαρμογή αποτελεί εύκολο και άμεσο στόχο.
  Επιπλέον σε αυτήν την περίπτωση αν κλειδωθεί, δεν μπορεί να εκτελεστεί σε κανέναν άλλον υπολογιστή μέχρι να υπάρξει ξεκλείδωμα από τον διακομιστή του Google Scholar.
- Η χρονοκαθυστέρηση, δηλαδή να υπάρχει ένας χρόνος αναμονής ανάμεσα στα requests που θα στέλνονται.
  Αυτή είναι μία σίγουρη λύση αλλά καταλήγει να κάνει την εφαρμογή μας αρκετά αργή στην εμφάνιση των αποτελεσμάτων.
  Παρότι η εφαρμογή μας επιτρέπει στον χρήστη να επιλέξει εκείνος το πόση χρονοκαθυστέρηση θα υπάρχει ανά αίτημα, προτείνεται ανεπιφύλακτα να εισάγει δεκαοχτώ δευτερόλεπτα ανά ένα αίτημα, καθώς ύστερα από μεγάλο αριθμό δοκιμών παρατηρήθηκε πως μετά τον παραπάνω χρόνο ο διακομιστής του Google Scholar δεν κλειδώνει την εφαρμογή μας, ανεξαρτήτως του αριθμού των δημοσιεύσεων που θα προκύψουν από την αναζήτηση.
  Με άλλα λόγια τα δεκαοχτώ δευτερόλεπτα αποτελούν άνω αλλά και κάτω όριο για το κλείδωμα.

Τέλος, ορισμένες δημοσιεύσεις, και κυρίως αναφορές προς αυτές, δεν συμπεριλαμβάνουν στα στοιχεία τους το έτος της δημοσίευσης καθώς και που έχουν δημοσιευτεί.
Σε αυτήν την περίπτωση, στο έτος δημοσίευσης βάζουμε ¨0¨ ενώ στο μέρος δημοσίευσης ¨Not Available¨

*Cookies: Τα Cookies είναι μικρά "αρχεία" που εγκαθίστανται στο σκληρό δίσκο του υπολογιστή, όταν επισκεπτόμαστε τοποθεσίες για πρώτη φορά και περιέχουν πληροφορίες τις οποίες χρησιμοποιούν οι ιστοσελίδες για την αναγνώρισή μας.

Ο κωδικός των cookies της Google έχει την μορφή:
PREF=ID= c08ea29d762de76b,
όπου τα τελευταία 16 αλφαριθμητικά είναι αριθμοί γραμμένοι σε δεκαεξαδική μορφή

Ο κωδικός των cookies του Scholar έχει την μορφή:
GSP=ID= c08ea29d762de76b:CF=4,
όπου το πεδίο CF=4 ενεργοποιεί την εμφάνιση του συνδέσμου μέσω του οποίου μπορεί να γίνει η εξαγωγή των στοιχείων της δημοσίευσης,
και τα 16 αλφαριθμητικά είναι αριθμοί γραμμένοι σε δεκαεξαδική μορφή.

**Υπολογισμός συνολικού αριθμού των συνδέσεων στον διακομιστή του Google Scholar.*

Αν ο επιστημονικός συγγραφέας έχει n δημοσιεύσεις και m αναφορές έχουμε k συνολικά requests όπου:

k =~ 1 + n + n + m

το 1 είναι το πρώτο request με το οποίο ζητάμε τις δημοσιεύσεις του συγγραφέα που εισήγαγε ο χρήστης για αναζήτηση,
το πρώτο n είναι για να πάρουμε τα στοιχεία της κάθε δημοσίευσης (BibTex link),
το δεύτερο n είναι για να πάρουμε τις αναφορές της κάθε δημοσίευσης (Cited By link),
και τέλος, το m είναι για να πάρουμε τα στοιχεία της κάθε αναφοράς (BibTex link)

Αριθμητικό παράδειγμα:
Συγγραφέας με 45 δημοσιεύσεις και 839 αναφορές:

k =~ 1+ 45 + 45 +839 = 930 requests

Τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν

Εισαγωγή

Από το μεγάλο εύρος τεχνολογιών και εργαλείων που υπάρχει αυτή την στιγμή διαθέσιμο για τους προγραμματιστές, επέλεξα να χρησιμοποιήσω τις τεχνολογίες και τα εργαλεία ανοικτού κώδικα.
Ο λόγος που κατέληξα στην παραπάνω απόφαση έχει να κάνει με το γεγονός πως η χρησιμοποίηση τους δεν απαιτεί άδεια, προσφέρουν αρκετά μεγάλη ευελιξία και κυρίως τεκμηρίωση μέσω παραδειγμάτων και βιβλιογραφικών αναφορών.
Η κάθε τεχνολογία που χρησιμοποιήθηκε περιγράφεται αναλυτικά παρακάτω και εξηγούνται οι λόγοι που επέλεξα την κάθε μία από αυτές.

JAVA

JRE (Java Runtime Environment): v. 1.7.0_07

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems.
Η Java είναι γλώσσα προγραμματισμού και πλατφόρμα.
Σαν γλώσσα προγραμματισμού χαρακτηρίζεται από τα εξής:
απλή, αντικειμενοστραφής, συμβατή με δικτυακά πρωτόκολλα, ουδέτερη της υποκείμενης αρχιτεκτονικής, φορητή, ασφαλής, υψηλής απόδοσης, δυναμική, σταθερή, interpreted και multithreaded.
Η Java σαν πλατφόρμα: Πλατφόρμα είναι το hardware ή software περιβάλλον όπου τρέχει ένα πρόγραμμα. Η Java πλατφόρμα διαφέρει από τις άλλες πλατφόρμες, γιατί είναι μία software-only που τρέχει πάνω από άλλες hardware πλατφόρμες.
Η Java πλατφόρμα έχει δύο στοιχεία: την Java Virtual Machine (JVM) και το Java Application Programming Interface (Java API).

Java Virtual Machine (JVM): Η Java παρέχει την δυνατότητα "write once, run everywhere" μέσω της JVM. Η JVM εφαρμόζεται πάνω από το λειτουργικό σύστημα της μηχανής και τα προγράμματα σε Java τρέχουν πάνω από την virtual machine. Σκοπός της είναι η απομόνωση του προγράμματος από τις διαφορές μεταξύ των υποκείμενων λειτουργικών συστημάτων και CPUs.
Java Application Programming Interface (Java API): Συλλογή από έτοιμα λογισμικά εργαλεία που προσφέρουν πολλές χρήσιμες δυνατότητες (π.χ. Graphical User Interface GUI). Το Java API είναι ομαδοποιημένο σε βιβλιοθήκες συσχετιζόμενων εργαλείων.

Γιατί Java;
Οι δύο τελευταίες παράγραφοι απαντούν στην ερώτηση: Η φορητότητα, δηλαδή η

κάθε εφαρμογή γραμμένη σε γλώσσα προγραμματισμού Java μπορεί να εκτελείτε σε κάθε υπολογιστή ανεξαρτήτως λειτουργικού συστήματος και επεξεργαστή, καθώς και οι έτοιμες βιβλιοθήκες που προσφέρει (πχ δημιουργίας γραφικών περιβαλλόντων) την καθιστούν ως μία από τις καλύτερες επιλογές για ανάπτυξη εφαρμογών.

MySQL

Server Version: 5.5.27 MySQL Community Version

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακής βάσης δεδομένων ανοικτού κώδικα (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 επιτρέπει την αποθήκευση, την αναζήτηση, την ταξινόμηση και την ανάκτηση των δεδομένων αποτελεσματικά . Ο MySQL διακομιστής ελέγχει την πρόσβαση των δεδομένων έτσι ώστε:
να μπορούν να δουλεύουν πολλοί χρήστες ταυτόχρονα, να παρέχει γρήγορη πρόσβαση και να διασφαλίζει ότι μόνο πιστοποιημένοι χρήστες μπορούν να έχουν πρόσβαση.
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. Επίσης είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία, και την ευελιξία που παρέχει.
Τέλος, μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows.

Γιατί Mysql;

Επελέγη το σύστημα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) της MySQL καθώς είναι ελεύθερης διανομής. Αυτό σημαίνει πως η μεταφόρτωση και η χρησιμοποίησή της είναι ελεύθερη και δεν υπόκειται σε περιορισμούς. Το γεγονός αυτό το κάνει ιδιαίτερα δημοφιλές και είναι ένα από τα πιο συχνά χρησιμοποιούμενα ΣΔΒΔ.
Τέλος, επειδή σκοπός είναι το σύστημά μας να τρέχει σε όσο το δυνατόν περισσότερους υπολογιστές και το ότι η MySQL μπορεί να λειτουργήσει σε διαφορετικά λειτουργικά συστήματα (κατά κύριο λόγο UNIX και MS Windows), η επιλογή του παραπάνω ΣΔΒΔ γίνεται εξαιρετικά ελκυστική.

SQL

Η SQL (Structured Query Language) είναι μία γλώσσα υπολογιστών στις βάσεις δεδομένων, που σχεδιάστηκε για τη διαχείριση δεδομένων, σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System, RDBMS) και η οποία, αρχικά, βασίστηκε στη σχεσιακή άλγεβρα. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης (SELECT) και ενημέρωσης δεδομένων (UPDATE), δημιουργίας (CREATE), τροποποίησης σχημάτων (ALTER SCHEMA) και σχεσιακών πινάκων (ALTER TABLE), αλλά και ελέγχου πρόσβασης στα δεδομένα (πχ SET FOREIGN_KEY_CHECKS=0;).
Η SQL ήταν μία από τις πρώτες γλώσσες για το σχεσιακό μοντέλο του *Edgar F. Codd*, στο σημαντικό άρθρο του το 1970, και έγινε η πιο ευρέως χρησιμοποιούμενη γλώσσα για τις σχεσιακές βάσεις δεδομένων.

Γιατί SQL;

Υπάρχουν δύο βασικοί σκοποί της γλώσσας προγραμματισμού των βάσεων δεδομένων:
για να δημιουργήσουμε και να επεξεργαστούμε τις βάσεις δεδομένων καθώς και για να εκτελέσουμε επερωτήσεις πάνω στους πίνακες και στα περιεχόμενά τους.
Και οι δύο παραπάνω λειτουργίες εκτελούνται με εξαιρετική ευκολία μέσω της SQL. Ο παραπάνω λόγος καθώς και το ότι κανείς δεν έχει καταλήξει σε κάποιον καλύτερο και ενιαία αναγνωρίσιμο τρόπο εργασίας με τα δεδομένα σε μία σχεσιακή βάση δεδομένων, καθιστούν την επιλογή της SQL ως την καλύτερη επιλογή.

Eclipse

Eclipse Indigo v.1.4.1

Το eclipse αποτελεί ένα SDK (Software Development Kit), ένα ολοκληρωμένο δηλαδή περιβάλλον μέσα από το οποίο μπορούμε να γράψουμε και να εκτελέσουμε κώδικα. Το περιβάλλον αυτό καθώς και όλες οι υπόλοιπες επεκτάσεις (plugins) που χρειάζονται για να εκτελέσουμε κώδικα σε Java ή/και C/C++ είναι ελεύθερης διανομής (freeware) και ανοικτού κώδικα (open source). Επιπλέον αποτελεί το ανερχόμενο περιβάλλον ανάπτυξης κώδικα, καθώς χρησιμοποιείται από ολοένα και περισσότερους χρήστες αλλά και εταιρείες.

Γιατί Eclipse IDE

Το Eclipse παρότι υποστηρίζει την ανάπτυξη εφαρμογών στις περισσότερες γλώσσες (πχ C, C++, python κλπ) με την εγκατάσταση των αντίστοιχων επεκτάσεων, η βασική γλώσσα για την οποία έχει δημιουργηθεί, είναι η Java.
Το παραπάνω σε συνδυασμό με το γεγονός πως είναι ελεύθερης διανομής και έχει ένα εξαιρετικά φιλικό στον χρήστη γραφικό περιβάλλον, το κατατάσσουν στις πρώτες επιλογές των προγραμματιστών για την ανάπτυξη Java εφαρμογών.

MySQL Workbench
version 5.2

Το MySQL Workbench είναι ένα cross-platform, οπτικό εργαλείο σχεδιασμού της βάσης δεδομένων που αναπτύχθηκε από την MySQL. Αναπτύχθηκε για να είναι η επόμενη γενιά οπτικών εφαρμογών σχεδιασμού της βάσης δεδομένων που μπορούν να χρησιμοποιηθούν για τον αποτελεσματικό σχεδιασμό (Design), τη διαχείριση (Administrate), την ανάπτυξη (Develop) και τώρα πλέον και την μετεγκατάσταση (Database Migration) από άλλα ΣΔΒΔ (όπως Microsoft SQL Server, PostgreSQL κ.α.).
Πρόκειται για το πολυαναμενόμενο διάδοχο εφαρμογή του DBDesigner4 έργου.

Γιατί MySQL Workbench;

Το MySQL Workbench αποτελεί αυτή την στιγμή ένα από τα κορυφαία οπτικά εργαλεία για τις λειτουργίες που περιγράφηκαν παραπάνω. Ανανεώνεται πολύ συχνά και εμπλουτίζεται με καινούργιες λειτουργίες διαρκώς.
Επιπλέον έχει το προνόμιο του να αναπτύσσεται από την εταιρία Oracle, κάτι που εγγυάται την καλύτερη συνεργασία με το ΣΔΒΔ MySQL, το οποίο επίσης υποστηρίζεται από την ίδια εταιρία.

JDBC
Η Συνδετικότητα Βάσης Δεδομένων JAVA (Java Database Connectivity - JDBC) είναι μία διεπαφή προγραμματισμού εφαρμογών (API) για την γλώσσα προγραμματισμού Java η οποία ορίζει την πρόσβαση ενός χρήστη σε

μια βάση δεδομένων. Παρέχει συναρτήσεις για εξαγωγή, πρόσθεση, ανανέωση ή διαγραφή δεδομένων σε μια βάση.

Γιατί JDBC

Η έλλειψη αξιόλογων εναλλακτικών επιλογών και το γεγονός πως ικανοποιεί τις απαιτήσεις του συστήματος, οδηγεί εύκολα στην επιλογή του παραπάνω οδηγού.

MySQL Query Browser

Ο MySQL Query Browser είναι ένα γραφικό εργαλείο, το οποίο παρέχεται από την MySQL για την δημιουργία, εκτέλεση και βελτιστοποίηση αιτημάτων σε γραφικό περιβάλλον. Έχει σχεδιαστεί προς βοήθεια στην ανάθεση αιτημάτων και ανάλυσης δεδομένων, τα οποία είναι αποθηκευμένα στην MySQL βάση δεδομένων.
Όλα τα αιτήματα που μπορούν να εκτελεστούν με τον MySQL Query Browser, μπορούν επίσης να εκτελεστούν με την εφαρμογή γραμμής εντολών mysql, όμως ο MySQL Query Browser επιτρέπει την υποβολή αιτημάτων και την επεξεργασία των δεδομένων με ένα πιο διαισθητικό, γραφικό τρόπο.

Γιατί MySQL Query Browser

Η επιλογή έγινε λόγω του εξαιρετικά φιλικού περιβάλλον χρήστη που έχει, των πολλών επιλογών που δίνει στον προγραμματιστή όσον αφορά τα αιτήματα στο ΣΔΒΔ και του γεγονότος ότι συνυπάρχει στο πακέτο του MySQL Workbench

Google Scholar

Τι είναι το Google Scholar

Το Google Scholar είναι μια ελεύθερα προσβάσιμη πανίσχυρη μηχανή αναζήτησης η οποία παρέχει έναν απλό τρόπο ευρείας αναζήτησης στην ακαδημαϊκή βιβλιογραφία.

Από ένα σημείο, μπορεί να γίνει αναζήτηση σε πολλά ερευνητικά πεδία και πηγές: εργασίες που έχουν αξιολογηθεί από ομότιμους επιστήμονες, διατριβές, βιβλία, περιλήψεις και άρθρα, από ακαδημαϊκούς εκδότες, επαγγελματικές ενώσεις, πηγές προδημοσιεύσεων, πανεπιστήμια και άλλους ακαδημαϊκούς οργανισμούς. Ο Google Scholar βοηθά να γίνει ο εντοπισμός των πλέον συναφή ερευνητικών εργασιών στον κόσμο της ακαδημαϊκής έρευνας.

Δυνατότητες του Google Scholar

- **Αναζήτηση** διαφόρων πηγών από ένα βολικό σημείο
- **Εύρεση** εργασιών, περιλήψεων και βιβλιογραφικών αναφορών
- **Εντοπισμός** ολόκληρης της εργασίας μέσω της βιβλιοθήκης σας ή στον ιστό
- **Μάθετε** για τις βασικές εργασίες σε οποιοδήποτε πεδίο έρευνας

Πώς κατατάσσονται τα άρθρα

Το Google Scholar στοχεύει στο να κατατάσσει τα άρθρα με τον ίδιο τρόπο που το κάνουν οι ερευνητές, σταθμίζοντας το πλήρες κείμενο κάθε άρθρου, τον συγγραφέα, το έντυπο στο οποίο εμφανίζεται το άρθρο καθώς και τη συχνότητα βιβλιογραφικής αναφοράς του άρθρου στην υπόλοιπη βιβλιογραφία. Τα πλέον συναφή αποτελέσματα θα εμφανίζονται πάντα στην πρώτη σελίδα.

Μορφή στην οποία εμφανίζονται τα αποτελέσματα

Μέσα από τις επιλογές του Google Scholar υπάρχει η δυνατότητα ενεργοποίησης επιπλέον συνδέσμου ο οποίος μπορεί να εμφανίσει τα στοιχεία της δημοσίευσης σε τρία format:

- BibTex
- End Note
- RefMan
- RefWorks

Στην υλοποίηση της παρούσας διπλωματικής επιλέχθηκε το format BibTex καθώς σε αυτή την μορφή γινόταν πιο εύκολα η ανάλυση και αποθήκευση των στοιχείων της δημοσίευσης (τίτλος, συγγραφείς, που δημοσιεύτηκε, έτος δημοσίευσης).

Γιατί επιλέξαμε τον Google Scholar

Οι βασικότεροι λόγοι που επιλέξαμε την παραπάνω μηχανή αναζήτησης είναι οι εξής τρεις:

1. Τα περιεχόμενα της βάσης της παρέχονται ελεύθερα και όχι μόνο σε συνδρομητές (πανεπιστήμια, ερευνητικά κέντρα κλπ)
2. Ο όγκος της πληροφορίας είναι τεράστιος. Μάλιστα, διαπιστώνεται μεγαλύτερος βαθμός κάλυψης επιστημονικών εκδόσεων και επίσης καλύπτει σημαντικό αριθμό διεθνών συνεδριών.
3. Τέλος, προσφέρει τα στοιχεία των δημοσιεύσεων (τίτλος, συγγραφείς, έτος έκδοσης, σημείο έκδοσης) καθώς και των αναφορών τους σε πολλές μορφές (EndNote, RefMan, RefWorks και BibTex). Η τελευταία μορφή είναι αυτή που χρησιμοποιούμε στην υλοποίηση του συστήματός μας.

Βάση Δεδομένων της εφαρμογής

Η βάση δεδομένων του συστήματος αποτελείτε από τέσσερεις πίνακες.
Συγκεκριμένα έχουμε τους ακόλουθους πίνακες:

Ο πίνακας papers περιέχει τις δημοσιεύσεις που προέκυψαν σαν αποτέλεσμα από τον συγγραφέα που εισήγαγε ο χρήστης στην αναζήτηση.
Έχει τα γνωρίσματα:

- id - *PrimaryKey*
- title
- authors
- publication
- yearOfPublication
- citations

Το γνώρισμα id περιέχει τον κωδικό που καθιστά την δημοσίευση μοναδική.

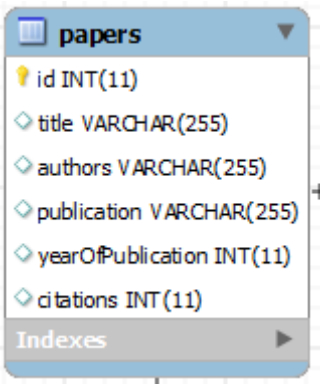
Το γνώρισμα title περιέχει τον τίτλος της δημοσίευσης.

Το γνώρισμα authors περιέχει τους συγγραφείς της δημοσίευσης

Το γνώρισμα publication περιέχει το που έχει δημοσιευτεί

Το γνώρισμα yearOfPublication περιέχει την χρόνια δημοσίευσης

Και τέλος το γνώρισμα citations περιέχει τον αριθμό του συνόλου των αναφορών που έχει η δημοσίευση.



Πίνακας papers

Ο πίνακας authors περιέχει τον κάθε ένα συγγραφέα της εκάστοτε δημοσίευσης ξεχωριστά.

Ο λόγος που έχουμε τον κάθε συγγραφέα ξεχωριστά είναι για να ελέγξουμε την ύπαρξη αυτοαναφορών και τις ετεροαναφορών.

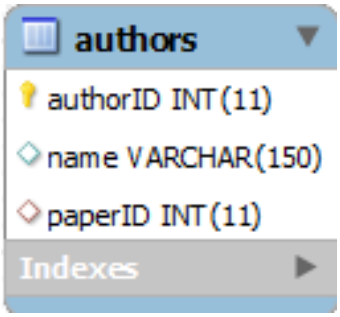
Ο παραπάνω πίνακας έχει τα γνωρίσματα:

- authorID - *PrimaryKey*
- name
- papered - *ForeignKey*

Το γνώρισμα id περιέχει τον κωδικό που καθιστά τον κάθε συγγραφέα της εκάστοτε δημοσίευσης μοναδικό.

Το γνώρισμα name περιέχει το όνομα του συγγραφέα

Και τέλος το γνώρισμα paperID είναι ξένο κλειδί και συνδέεται με το id του πίνακα paper που είδαμε πριν από λίγο.
Έτσι ώστε μία δημοσίευση να μπορεί να συνδέεται με τον κάθε συγγραφέα της χωριστά μέσω του id της και του paperID που υπάρχει στον πίνακα authors



Πίνακας authors

Ο πίνακας citations περιέχει τις αναφορές για κάθε μία από τις δημοσιεύσεις που προέκυψαν από την αναζήτηση.

Ο πίνακας αυτός έχει τα παρακάτω γνωρίσματα:

- citationID - *PrimaryKey*
- title
- authors
- yearOfPublication
- papered - *ForeignKey*

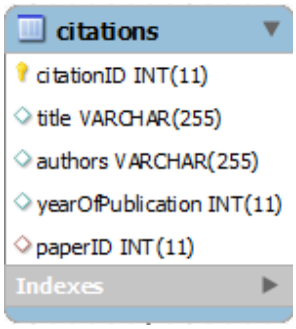
Το γνώρισμα citationID περιέχει τον κωδικό που καθιστά την κάθε αναφορά μοναδική.

Το γνώρισμα title περιέχει τον τίτλο της δημοσίευσης που κάνει την αναφορά

Το γνώρισμα authors περιέχει τους συγγραφείς της δημοσίευσης που κάνει την αναφορά

Το γνώρισμα yearOfPublication περιέχει την χρόνια δημοσίευσης που κάνει την αναφορά

Και τέλος το γνώρισμα paperID είναι ξένο κλειδί και συνδέεται με το id του πίνακα papers που είδαμε πριν από λίγο.
Έτσι μία δημοσίευση να μπορεί να συνδέεται με τον κάθε αναφορά της μέσω του citationID της και του paperID που υπάρχει στον πίνακα authors



Πίνακας citations

Τέλος, έχουμε τον πίνακα citationsauthors ο οποίος έχει τις αντίστοιχες ιδιότητες του πίνακα authors που είδαμε πριν από λίγο, αλλά σε αυτή την περίπτωση ο πίνακας αυτός περιέχει τους συγγραφείς, της κάθε δημοσίευσης που κάνει αναφορά, ξεχωριστά.

Ο λόγος που έχουμε πάλι τον κάθε συγγραφέα ξεχωριστά είναι για να ελέγξουμε την ύπαρξη αυτοαναφορών και τις ετεροαναφορών.

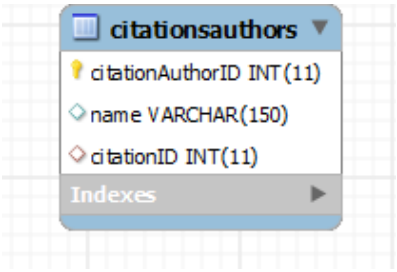
Ο παραπάνω πίνακας έχει τα γνωρίσματα:

- citationAuthorID - *PrimaryKey*
- name
- citationID - *ForeignKey*

Το γνώρισμα id περιέχει τον κωδικό που καθιστά τον κάθε συγγραφέα της εκάστοτε δημοσίευσης που κάνει αναφορά μοναδικό.

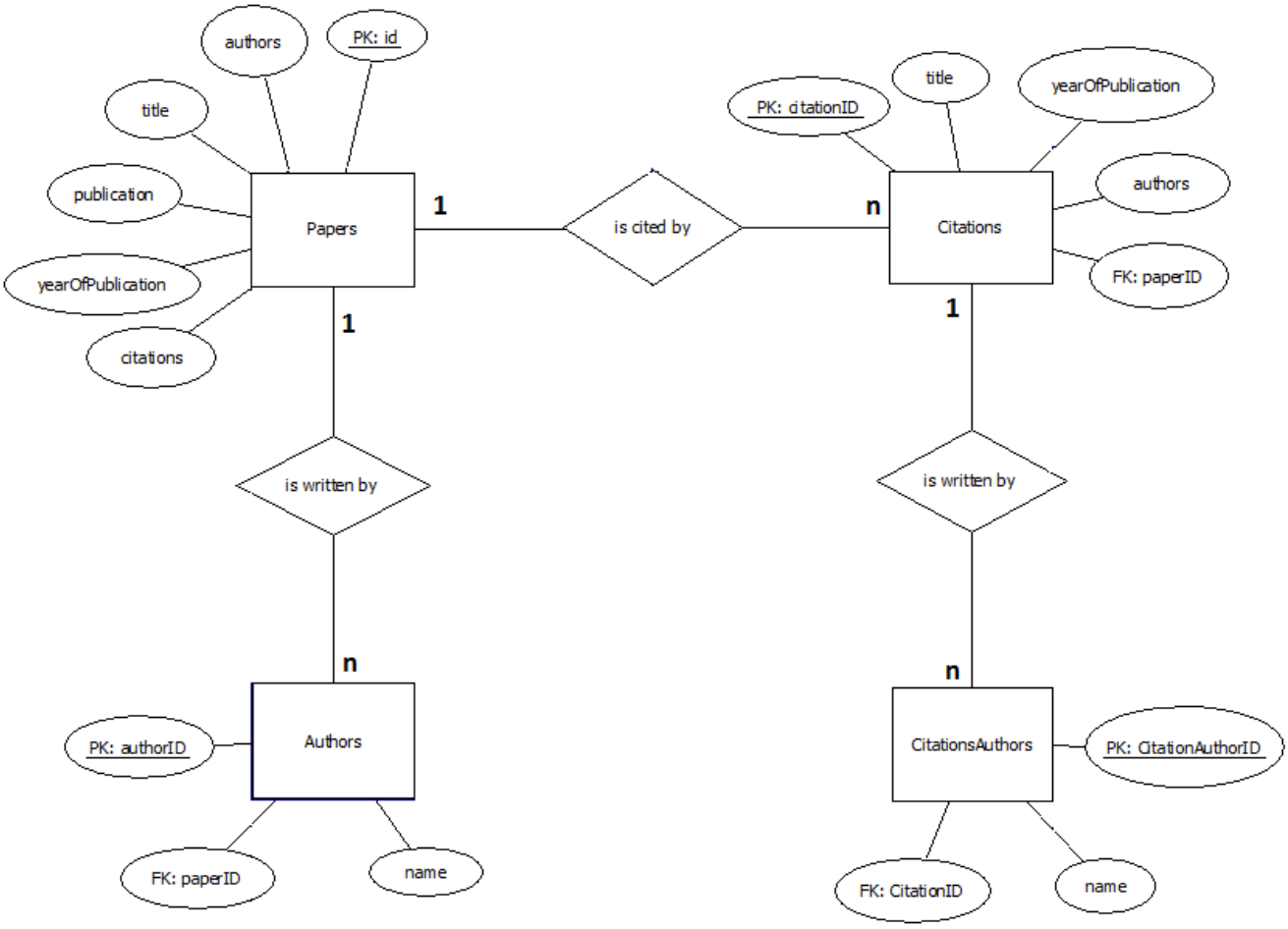
Το γνώρισμα name περιέχει το όνομα του συγγραφέα

Και τέλος το γνώρισμα citationID είναι ξένο κλειδί και συνδέεται με το cititationID του πίνακα citations που είδαμε πριν από λίγο.
Έτσι ώστε μία δημοσίευση που κάνει αναφορά να μπορεί να συνδέεται με τον κάθε συγγραφέα της χωριστά μέσω του citationID της και του citationID που υπάρχει στον πίνακα citationauthors



Πίνακας citationsauthors

Παρακάτω φαίνεται και το διάγραμμα Οντοτήτων-Συσχετήσεων της βάσης δεδομένων (Entity – Relationship Diagramm)



Διάγραμμα Οντοτήτων-Συσχετήσεων (E-R Diagramm)

Όπως παρατηρούμε και από το διάγραμμα:

Ο πίνακας papers συνδέεται με τον πίνακα authors με σχέση ένα προς πολλά καθώς μία δημοσίευση μπορεί να έχει από έναν μέχρι πολλούς συγγραφείς

Ο πίνακας papers συνδέεται με τον πίνακα citations με σχέση μία προς πολλά καθώς μία δημοσίευση μπορεί να έχει από μία/καμία μέχρι πολλές αναφορές

Ο πίνακας citations συνδέεται με τον πίνακα citationauthors με σχέση ένα προς πολλά καθώς μία αναφορά μπορεί να έχει από έναν μέχρι πολλούς συγγραφείς.

Περιγραφή και οργάνωση κώδικα Java του συστήματος

Για την δημιουργία του συστήματος υλοποιήθηκαν δεκαπέντε (15) κλάσεις μοιρασμένες σε πέντε (5) πακέτα.
Ο λόγος που η εφαρμογή έχει υλοποιηθεί σε μεγάλο αριθμό κλάσεων είναι η εύκολη εγγραφή, ανάγνωση και κυρίως αποσφαλμάτωση κώδικα.
Η λειτουργία της καθεμιάς εκ των κλάσεων αναλύεται παρακάτω.

Επιγραμματικά οι κλάσεις που υλοποιήθηκαν και τα πακέτα στα οποία ανήκουν σε αλφαβητική σειρά:

Πακέτα

Πακέτο gui

Το πακέτο gui περιέχει όλες τις κλάσεις εκείνες που δημιουργούν το γραφικό περιβάλλον του συστήματος, όπως για παράδειγμα τα παράθυρα και τα προειδοποιητικά μηνύματα καθώς επίσης και την κλάση που δημιουργεί τα νήματα έτσι ώστε να υπάρχει η δυνατότητα να εκτελούνται παραπάνω παράθυρα ταυτόχρονα

- CitationsWindow.java
- DelayWindow.java
- LoginWindow.java
- MainWindow.java
- YearsRangeWindow.java
- ProgressBars.java
- MyThread.java

Πακέτο database

Το πακέτο database περιέχει μία κλάση, η οποία υλοποιεί όλες τις μεθόδους που έχουν σχέση με την βάση δεδομένων, όπως για παράδειγμα σύνδεση στην βάση και πραγματοποίηση αιτημάτων

- ScholarDB.java

Πακέτο citations

Το πακέτο citations περιέχει όλες τις κλάσεις που σχετίζονται και χειρίζονται τις δημοσιεύσεις που κάνουν αναφορές στον επιστημονικού συγγραφέα που εισήχθη προς αναζήτηση

- CitationsFetch.java

- CitationsTable.java
- PartialCitations.java

Πακέτο methods
Το πακέτο methods περιέχει μία κλάση η οποία υλοποιεί όλες τις συναρτήσεις που εκτελούνται πίσω από το γραφικό περιβάλλον, όπως είναι η δημιουργία του αιτήματος (request) και σύνδεση στον Google Scholar και η ανάλυση (parse) των αποτελεσμάτων

- Core.java

Πακέτο mainPapers
Το πακέτο mainPapers περιέχει τις κλάσεις που σχετίζονται και χειρίζονται τις δημοσιεύσεις που προκύπτουν σαν αποτέλεσμα της αναζήτησης από τον εισαχθέντα επιστημονικό συγγραφέα

- MainPapersJtable.java
- MainPaperTable.java
- PartialMainPapersJtable.java

Λειτουργία της κάθε κλάσης ξεχωριστά:

CitationsFetch.java:
Στην κλάση αυτή υλοποιούνται τα αιτήματα προς την βάση δεδομένων και έχουν να κάνουν με:

- την εμφάνιση όλων των αναφορών,
- την εμφάνιση των αυτοαναφορών
- και την εμφάνιση των ετεροαναφορών

στον πίνακα με τα αποτελέσματα του παραθύρου.

CitationsTable.java
Στην παραπάνω κλάση δημιουργείται ο κορμός του πίνακα που υπάρχει στην σελίδα με τις αναφορές (πχ αριθμός γραμμών, αριθμός στηλών, ονόματα στηλών κλπ).

CitationsWindow.java

Στην κλάση αυτή δημιουργείται το παράθυρο των αναφορών καθώς και το μενού που υπάρχει σε αυτήν.

Core.java

Η κλάση Core.java είναι η πιο σύνθετη και ίσως η σημαντικότερη κλάση του συστήματος.
Στην κλάση αυτή ετοιμάζετε το request προς τον Google Scholar με βάση τον συγγραφέα που έδωσε ο χρήστης, πραγματοποιείτε η σύνδεση στον Google Scholar, γίνεται ανάλυση (parsing) των αποτελεσμάτων και αποθήκευση τους στην βάση δεδομένων.

DelayWindow.java

Η παραπάνω κλάση φτιάχνει το παράθυρο το οποίο ζητάει από τον χρήστη να εισάγει την χρονοκαθυστέρηση και ανά πόσες συνδέσεις (hits) να πραγματοποιείται αυτή.
Η χρονοκαθυστέρηση, όπως θα αναλυθεί παρακάτω, είναι απαραίτητη καθώς ο διακομιστής του Google Scholar αν αντιληφθεί πολλές συνδέσεις σε σύντομο χρονικό διάστημα, θεωρεί πως δέχεται επίθεση και κλειδώνει την IP από την οποία προέρχονται τα αιτήματα.

LoginWindow.java

Η κλάση αυτή δημιουργεί το πρώτο παράθυρο που εμφανίζεται κατά την εκτέλεση του προγράμματος και ζητάει το όνομα χρήστη και τον κωδικό της βάσης δεδομένων έτσι ώστε να συνδεθεί σε αυτήν.

MainPapersFetch.java

Στην κλάση αυτή υλοποιείται το αίτημα προς την βάση δεδομένων μέσω της οποίας ανακτώνται τα στοιχεία των δημοσιεύσεων (τίτλος, συγγραφείς, που δημοσιεύθηκε, το έτος δημοσίευσης, ο αριθμός των αναφορών καθώς και ο αριθμός των ετεροαναφορών).
 και έχουν να κάνουν με στον πίνακα με τα αποτελέσματα του παραθύρου.
Τα παραπάνω αποτελέσματα εμφανίζονται σε μορφή πίνακα.

MainPaperTable.java

Σε αυτήν την κλάση δημιουργείται ο κορμός του πίνακα που υπάρχει στην σελίδα με τις δημοσιεύσεις (πχ αριθμός γραμμών, αριθμός στηλών, ονόματα στηλών κλπ).

MainWindow.java
Η κλάση αυτή δημιουργεί το κυρίως παράθυρο της εφαρμογής και είναι το πρώτο παράθυρο που αντικρίζει ο χρήστης μόλις συνδεθεί επιτυχώς στην βάση δεδομένων.

MyThread.java
Η παραπάνω κλάση δημιουργεί τα νήματα, τα οποία φτιάχνονται με σκοπό να επιτρέψουν στην εφαρμογή να εκτελεί παραπάνω από ένα παράθυρο ταυτόχρονα.

CitationsBetweenYears.java
Σε αυτή την κλάση υλοποιούνται τα αιτήματα προς την βάση δεδομένων τα οποία ανακτούν τις αναφορές που έχουν γίνει σε συγκεκριμένο εύρος ετών στον ζητούμενο επιστημονικό συγγραφέα.

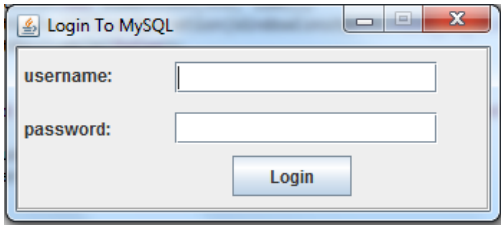
MainPapersBetweenYears.java
Αντίστοιχα με την προηγούμενη κλάση που είδαμε, σε αυτή την κλάση υλοποιούνται τα αιτήματα προς την βάση δεδομένων τα οποία ανακτούν τις δημοσιεύσεις που έχουν αναφορές στο επιλεγμένο εύρος ετών.

ProgressBars.java
Εδώ υλοποιείται ο κώδικας για την δημιουργία των δύο μπαρών προόδου.
Η πρώτη σχετίζεται με την πρόοδο κατεβάσματος δημοσιεύσεων και η δεύτερη με την πρόοδο κατεβάσματος των αναφορών για την εκάστοτε δημοσίευση ξεχωριστά.

YearsRangeWindow.java
Η παραπάνω κλάση δημιουργεί το παράθυρο το οποίο εμφανίζεται στον χρήστη για να εισάγει το χρονικό εύρος στο οποίο θα γίνει η αναζήτηση των αναφορών.

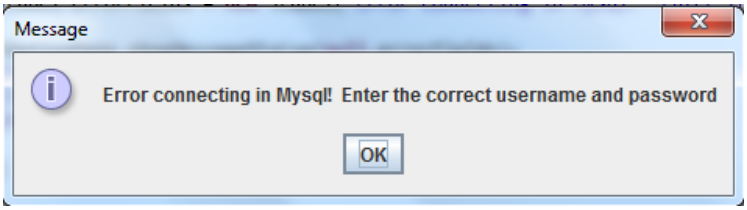
Λειτουργία του Συστήματος

Το σύστημα κατά την εκκίνησή του ζητάει να εισαχθεί όνομα χρήστη καθώς και κωδικός πρόσβασης στην βάση δεδομένων με σκοπό την σύνδεση σε αυτήν.



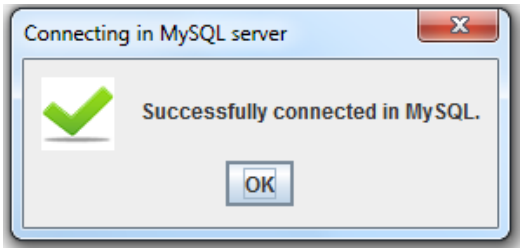
Εικόνα 1 – Παράθυρο που προτρέπει τον χρήστη να εισάγει το όνομα χρήστη και τον κωδικό πρόσβασης για την σύνδεση στην Βάση Δεδομένων

Τα στοιχεία που εισάγονται ελέγχονται για την ορθότητά τους και εφόσον έχει γίνει λάθος στο όνομα χρήστη ή στον κωδικό εμφανίζεται προειδοποιητικό παράθυρο που ενημερώνει τον χρήστη να εισάγει τα σωστά στοιχεία.



Εικόνα 2α – Παράθυρο που ενημερώνει τον χρήση πως έχει εισάγει λάθος όνομα χρήστη η κωδικό πρόσβασης

Ενώ στην περίπτωση που έχουν εισαχθεί τα σωστά στοιχεία, εμφανίζεται το μήνυμα που ενημερώνει πως έχει πραγματοποιηθεί επιτυχώς σύνδεση στο σύστημα διαχείρισης βάσεων δεδομένων της MySQL.



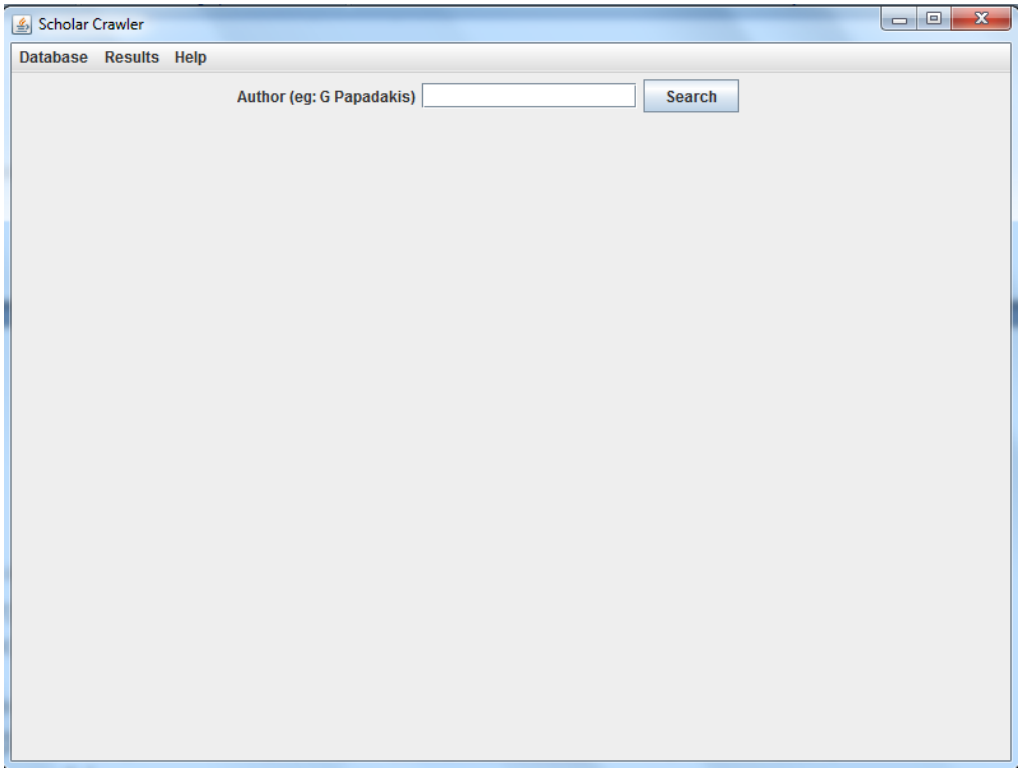
Εικόνα 2β – Παράθυρο που ενημερώνει τον χρήση πως πραγματοποιήθηκε επιτυχώς σύνδεση στην βάση δεδομένων

Όπως είδαμε και παραπάνω, υπάρχει ο περιορισμός πως ο χρήστης που θα συνδεθεί πρέπει να έχει δικαιώματα δημιουργίας σχήματος στην βάση δεδομένων, δημιουργίας πινάκων, διαγραφής περιεχομένων πινάκων (TRUNCATE TABLE) και

τέλος να μπορεί να ενεργοποιεί/απενεργοποιεί τους ελέγχους για τα ξένα κλειδιά (FOREIGN_KEY_CHECKS=0/1)

Όταν ο χρήστης συνδεθεί επιτυχώς, το σύστημα δημιουργεί το αρχείο login.txt με τα στοιχεία σύνδεσης στην βάση δεδομένων έτσι ώστε να τα έχει στην διάθεσή του και να μην τα ξαναζητήσει κατά την διάρκεια της εκτέλεσης.
Στην συνεχεία συνδέεται στην MySQL και εφόσον δεν υπάρχει ήδη η βάση δεδομένων, εκτελεί τον κώδικα sql που υπάρχει στο αρχείο db_script.sql μέσω του οποίου δημιουργούνται αυτόματα η βάση δεδομένων καθώς και οι απαιτούμενοι πίνακες που είναι απαραίτητοι για την αποθήκευση των δεδομένων του συστήματος.

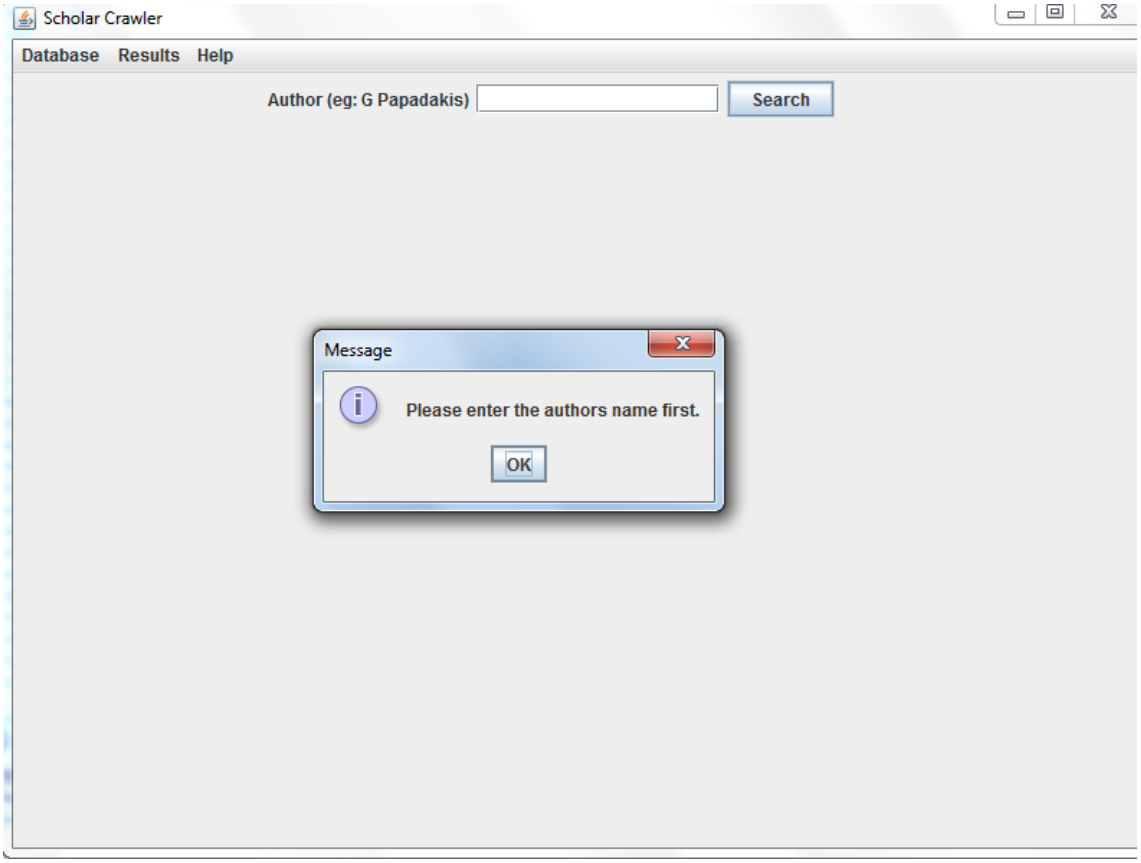
Στην συνέχεια εμφανίζεται το κυρίως παράθυρο:



Εικόνα 3a – Κυρίως παράθυρο του συστήματος

Στο κυρίως παράθυρο ο χρήστης μπορεί να εισάγει το όνομα του επιστημονικού συγγραφέα γράφοντας το αρχικό γράμμα του ονόματός του και το επώνυμό του με αγγλικούς χαρακτήρες (πχ G Papadakis) χωρίς να βάλει ομοιωματικά στην αρχή και στο τέλος του ονόματος.

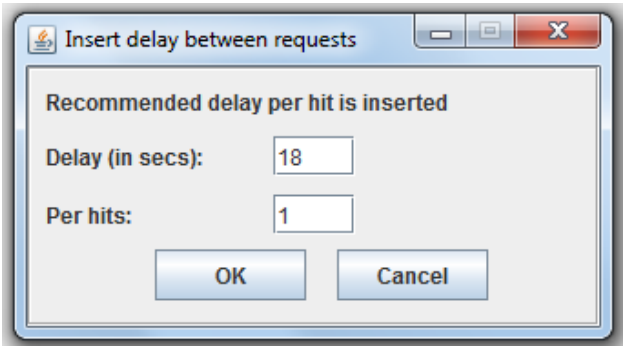
Όταν ο χρήστης πατήσει το κουμπί «Search» γίνεται έλεγχος για το αν εισήγαγε το όνομα του συγγραφέα. Αν δεν το έκανε, εμφανίζεται προειδοποιητικό μήνυμα που τον ενημερώνει πως πρέπει να εισάγει τον όνομα του συγγραφέα για να εκτελεστεί η αναζήτηση



Εικόνα 3b – Παράθυρο που ζητάει από τον χρήστη να εισάγει τον συγγραφέα πριν εκτελέσει την αναζήτηση

Μόλις εισάγει ο χρήστης το όνομα του συγγραφέα και πριν εκτελεστεί η αναζήτηση, διαγράφονται τα περιεχόμενα της βάσης δεδομένων και στην συνέχεια παραμετροποιείται το όνομα του συγγραφέα που έχει εισαχθεί.
Συγκεκριμένα από την μορφή G Papadakis γίνεται "G+Papadakis" και τοποθετείτε στο σωστό πεδίο του request που θα σταλεί στον scholar.

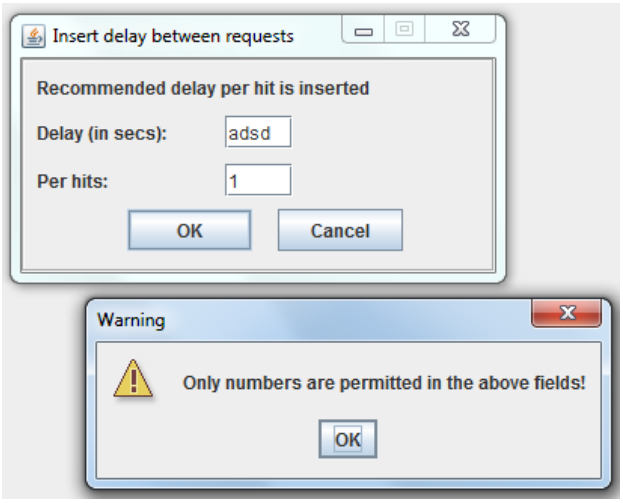
Μετά την εισαγωγή του επιστημονικού συγγραφέα και πριν την εκτέλεση της αναζήτησης, το σύστημα ρωτάει τον χρήστη για την χρονοκαθυστέρηση και ανά πόσες συνδέσεις θα πραγματοποιείται (ο λόγος αναλύθηκε παραπάνω).



Εικόνα 4α – Παράθυρο που ζητάει από τον χρήστη να εισάγει την χρονοκαθυστέρηση και ανά πόσες συνδέσεις να πραγματοποιείται αυτή

Όπως παρατηρούμε και από την Εικόνα 4, το σύστημα έχει ήδη εισάγει την προτεινόμενη καθυστέρηση ανά αίτημα στον διακομιστή του Google Scholar.

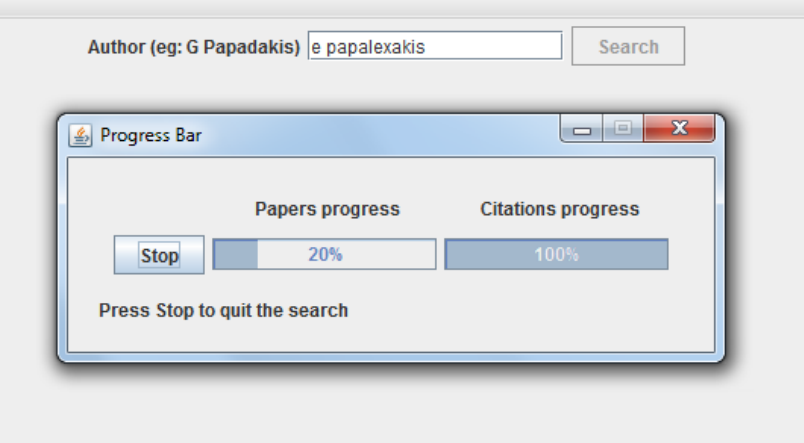
Μόλις ο χρήστης εισάγει την χρονοκαθυστέρηση και πατήσει το κουμπί "OK", γίνετε έλεγχος αν η είσοδος είναι αριθμός. Στην περίπτωση που δεν είναι αριθμός αλλά είναι γράμμα ή άλλη μη επιτρεπόμενη είσοδος, εμφανίζεται μήνυμα που επισημαίνει την λάθος είσοδο και τον παροτρύνει να εισάγει αριθμό στο αντίστοιχο πεδίο.



Εικόνα 4β – Παράθυρο που ενημερώνει τον χρήστη πως μπορεί να εισάγει μόνο αριθμούς στα ζητούμενα πεδία

Καθώς πραγματοποιείται η αναζήτηση, απενεργοποιείται το κουμπί της αναζήτησης μέχρις ότου τελειώσει η αναζήτηση και παράλληλα εμφανίζεται παράθυρο το οποίο δείχνει την πρόοδο της αναζήτησης. Συγκεκριμένα εμφανίζονται δύο μπάρες, με την πρώτη να αφορά την πρόοδο της αναζήτησης των δημοσιεύσεων του ζητούμενου επιστημονικού συγγραφέα και με την δεύτερη να αφορά την πρόοδο της αναζήτησης των αναφορών της εκάστοτε δημοσίευσης.

Εφόσον ο χρήστης επιθυμεί, μπορεί να διακόψει την αναζήτηση πατώντας το κουμπί ¨Stop¨, επανενεργοποιώντας το κουμπί της αναζήτησης του κυρίως παραθύρου.



Εικόνα 5 – Παράθυρο που δείχνει την πρόοδο αναζήτησης των δημοσιεύσεων (πρώτη μπάρα) και την πρόοδο αναζήτησης των αναφορών ανά δημοσίευση (δεύτερη μπάρα).
Όσο η αναζήτηση είναι σε εξέλιξη, το κουμπί της αναζήτησης παραμένει απενεργοποιημένο.

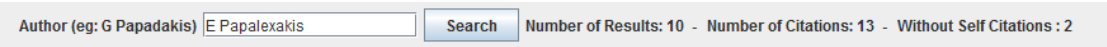
Κατά την διάρκεια της αναζήτησης, γίνεται ανάλυση των αποτελεσμάτων που επιστρέφει ο διακομιστής του GoogleScholar και αποθηκεύονται στους πίνακες της βάσης δεδομένων τα στοιχεία που είναι χρήσιμα για την εφαρμογή μας.

Συγκεκριμένα στους παρακάτω πίνακες αποθηκεύονται τα εξής στοιχεία:

- πίνακας papers
  σε αυτόν τον πίνακα γίνεται εισαγωγή των δημοσιεύσεων του ζητούμενου επιστημονικού συγγραφέα
- πίνακας authors
  σε αυτόν τον πίνακα γίνεται εισαγωγή του κάθε συγγραφέα ξεχωριστά για την κάθε δημοσίευση. Ο λόγος είναι για να μπορέσουμε να ελέγξουμε αργότερα τις αυτοαναφορές και τις ετεροαναφορές
- πίνακας citations
  σε αυτόν τον πίνακα γίνεται εισαγωγή των δημοσιεύσεων που κάνουν αναφορά στον ζητούμενο επιστημονικο συγγραφέα
- πίνακας citationAuthors
  σε αυτόν τον πίνακα γίνεται εισαγωγή του κάθε συγγραφέα ξεχωριστά για την κάθε δημοσίευση της αναφοράς. Ο λόγος είναι για να μπορέσουμε να ελέγξουμε αργότερα τις αυτοαναφορές και τις ετεροαναφορές

Όταν ολοκληρωθεί η αναζήτηση, το σύστημα επιστρέφει στο κυρίως παράθυρο όπου πλέον εμφανίζονται τα εξής αποτελέσματα:

- το σύνολο των επιστημονικών δημοσιεύσεων που έχουν γραφτεί από τον ίδιον,
- το σύνολο των αναφορών επί του συνόλου των δημοσιεύσεών του,
- το σύνολο των ετεροαναφορών (αναφορές αποκλειστικά από άλλους συγγραφείς) επί του συνόλου των δημοσιεύσεών του.



Εικόνα 6 – Εμφάνιση αριθμού αποτελεσμάτων, αναφορών και ετεροαναφορών

Επίσης, στο ίδιο παράθυρο και ακριβώς από κάτω, εμφανίζεται πίνακας ο οποίος περιέχει την κάθε επιστημονική δημοσίευση ξεχωριστά με όλα τα στοιχεία της (τίτλος, συγγραφείς, που δημοσιεύθηκε έτος δημοσίευσης αναφορές και ετεροαναφορές).

Title	Authors	Publicaiton	Year of Publication	Citation
Co-clustering as multilinear decomposition with sparse latent factors	Papalexakis E.E. , Sidiropoulos N.D.	Acoustics Speech and Signal Processing (ICASSP) 2011 IEEE Int...	2011	8
Reviewer profiling using sparse matrix regression	Papalexakis E.E. , Sidiropoulos N.D. , Garofalakis M.N.	Data Mining Workshops (ICDMW) 2010 IEEE International Confer...	2010	4
Coclusteringa useful tool for chemometrics	Bro R. , Papalexakis E.E. , Acar E. , Sidiropoulos N.D.	Journal of Chemometrics	2012	1
Network Anomaly Detection using Co-clustering	Papalexakis E.E. , Beutel A. , Steenkiste P.	Not Available	0	0
SigSpot: mining significant anomalous regions from time-evolving ...	Mongiovi M. , Bogdanov P. , Ranca R. , Singh A.K. , Papalexa...	Proceedings of the 2012 international conference on Managemen...	2012	0
GigaTensor: scaling tensor analysis up by 100 times-algorithms an...	Kang U. , Papalexakis E. , Harpale A. , Faloutsos C.	Proceedings of the 18th ACM SIGKDD international conference o...	2012	0
Reviewer Profiling Using Factor Analysis	Papalexakis E.E.	Not Available	2010	0
ParCube: Sparse Parallelizable Tensor Decompositions	Papalexakis E. , Faloutsos C. , Sidiropoulos N.	Machine Learning and Knowledge Discovery in Databases	2012	0
From K-means to higher-way co-clustering: multilinear decompositi...	Papalexakis E.E. , Sidiropoulos N.D. , Bro R.	Not Available	0	0
TENSORSPLAT: Spotting Latent Anomalies in Time	Koutra D. , Papalexakis E.E. , Faloutsos C.	Not Available	0	0

Εικόνα 7 – Πίνακας με τα αποτελέσματα

Κάνοντας διπλό κλικ σε μία από της δημοσιεύσεις, δημιουργείται ένα νέο δευτερεύων παράθυρο στο οποίο εμφανίζονται οι αναφορές για την συγκεκριμένη δημοσίευση, με τον τίτλο τους, τους συγγραφείς και το έτος δημοσίευσής τους)

Cited by 8 Papers	Cited By Authors	Year of Publication
Coclusteringa useful tool for chemometrics	Bro R. , Papalexakis E.E. , Acar E. , Sidiropoulos N.D.	2012
Network Anomaly Detection using Co-clustering	Papalexakis E.E. , Beutel A. , Steenkiste P.	0
Algorithms for Graph Similarity and Subgraph Matching	Koutra D. , Parikh A. , Ramdas A. , Xiang J.	2011
Sparse robust matrix tri-factorization with application to cancer genomics	Kim S.J. , Hwang T.H. , Giannakis G.B.	2012
GigaTensor: scaling tensor analysis up by 100 times-algorithms and discoveries	Kang U. , Papalexakis E. , Harpale A. , Faloutsos C.	2012
Multi-Way Compressed Sensing for Sparse Low-Rank Tensors	Sidiropoulos N.D. , Kyrillidis A.	2012
ParCube: Sparse Parallelizable Tensor Decompositions	Papalexakis E. , Faloutsos C. , Sidiropoulos N.	2012
From K-means to higher-way co-clustering: multilinear decomposition with sparse latent factors	Papalexakis E.E. , Sidiropoulos N.D. , Bro R.	0

Εικόνα 8 – Πίνακας με τις αναφορές της δημοσίευσης που επιλέχθηκε

Από το μενού επιλογών που υπάρχει επάνω αριστερά, ο χρήστης μπορεί να επιλέξει:

- Εμφάνιση αυτοαναφορών (Show SelfCitations)
- Εμφάνιση ετεροαναφορών (Hide SelfCitations)
- Εμφάνιση όλων των αναφορών (Show all citations)

Εικόνα 9 – Menu με τις επιλογές που αφορούν τις αναφορές

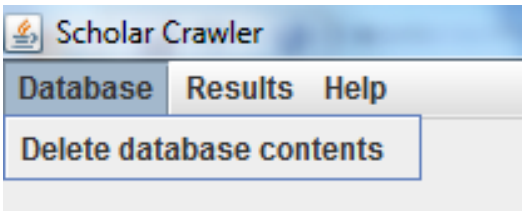
Με την πρώτη επιλογή, τα περιεχόμενα του πίνακα ανανεώνονται και πλέον εμφανίζονται οι αναφορές που έχουν γίνει από τους ίδιους συγγραφείς της επιλεχθέντας δημοσίευσης σε αυτήν (SelfCitations).

Με την δεύτερη επιλογή, τα περιεχόμενα του πίνακα ανανεώνονται και πλέον εμφανίζονται οι αναφορές που έχουν γίνει στην δημοσίευση που επιλέχθηκε από άλλους συγγραφείς (Non SelfCitations).

Τέλος με την τρίτη επιλογή, τα περιεχόμενα του πίνακα ανανεώνονται και πλέον εμφανίζεται το σύνολο των αναφορών που έχουν γίνει στην δημοσίευση που επιλέχθηκε.

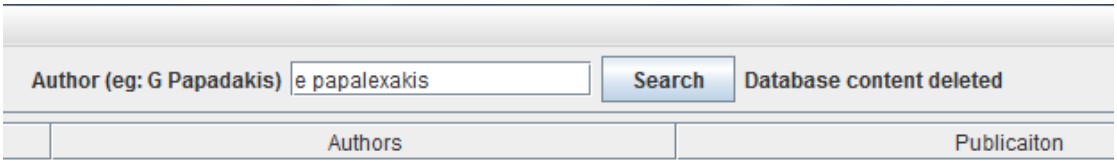
Μενού Επιλογών

Στην γραμμή μενού που υπάρχει επάνω αριστερά και συγκεκριμένα στο μενού Database – Delete database contents, ο χρήστης έχει την επιλογή να διαγράψει (TRUNCATE) τα περιεχόμενα που τυχόν έχει η βάση δεδομένων



Εικόνα 10 – Menu με την επιλογή διαγραφής των περιεχομένων της βάσης δεδομένων

Με την εκτέλεση της εντολής, εμφανίζεται μήνυμα το οποίο ενημερώνει τον χρήστη πως πραγματοποιήθηκε διαγραφή περιεχομένων της βάσης δεδομένων

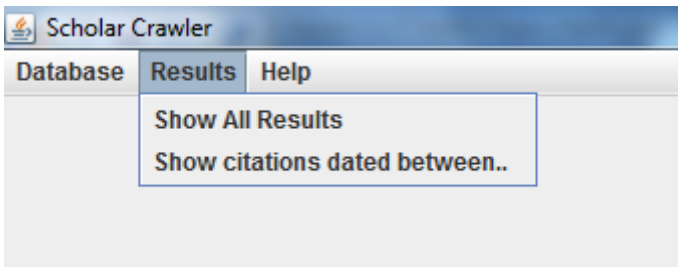


Εικόνα 11 – Εμφάνιση ενημερωτικού μηνύματος πραγματοποίησης διαγραφής

Στο μενού Results υπάρχουν δύο επιλογές:

- Show Results
- Show citations dated between

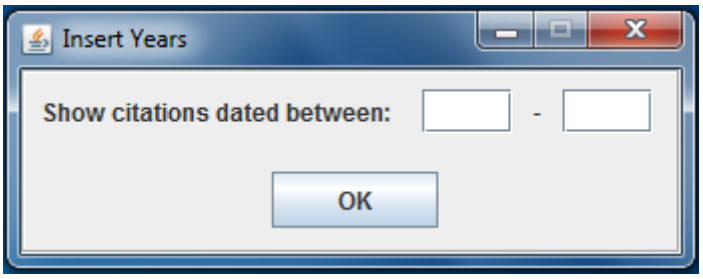
Με την πρώτη επιλογή, ο πίνακας ανανεώνεται και εμφανίζει το σύνολο των αποτελεσμάτων-δημοσιεύσεων με βάση των συγγραφέα της αναζήτησης



Εικόνα 12 – Menu επιλογών για τα αποτελέσματα που έχουν προκύψει για τον ζητούμενο επιστημονικό συγγραφέα

Με την δεύτερη επιλογή εμφανίζεται πίνακας με τις αναφορές σε ένα συγκεκριμένο εύρος χρόνων που έχουν γίνει στον συγγραφέα της αναζήτησης και σε κάθε μία από

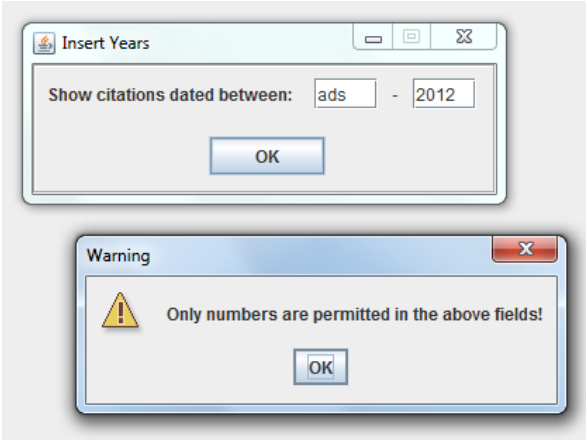
τις δημοσιεύσεις του ξεχωριστά.
Με την επιλογή του συγκεκριμένου μενού, εμφανίζεται ένα μικρό παράθυρο στον χρήστη που τον καλεί να εισάγει σε δύο πεδία το χρονικό εύρος.



Εικόνα 13 – Παράθυρο στο οποίο εισάγεται το χρονικό εύρος των αναφορών

Αν το πρώτο πεδίο αφεθεί κενό, το σύστημα δίνει αυτόματα την τιμή 0 έτσι ώστε να μην υπάρχει κάτω όριο χρόνων στην αναζήτησης, ενώ αν αφεθεί το δεύτερο πεδίο κενό το σύστημα δίνει αυτόματα το τρέχον έτος.

Μόλις ο χρήστης εισάγει το χρονικό εύρος των αναφορών και πατήσει το κουμπί "OK", γίνετε έλεγχος αν η είσοδος είναι αριθμός. Στην περίπτωση που δεν είναι αριθμός αλλά είναι γράμμα ή άλλη μη επιτρεπόμενη είσοδος, εμφανίζεται μήνυμα που επισημαίνει την λάθος είσοδο και τον παροτρύνει να εισάγει αριθμό στο αντίστοιχο πεδίο.

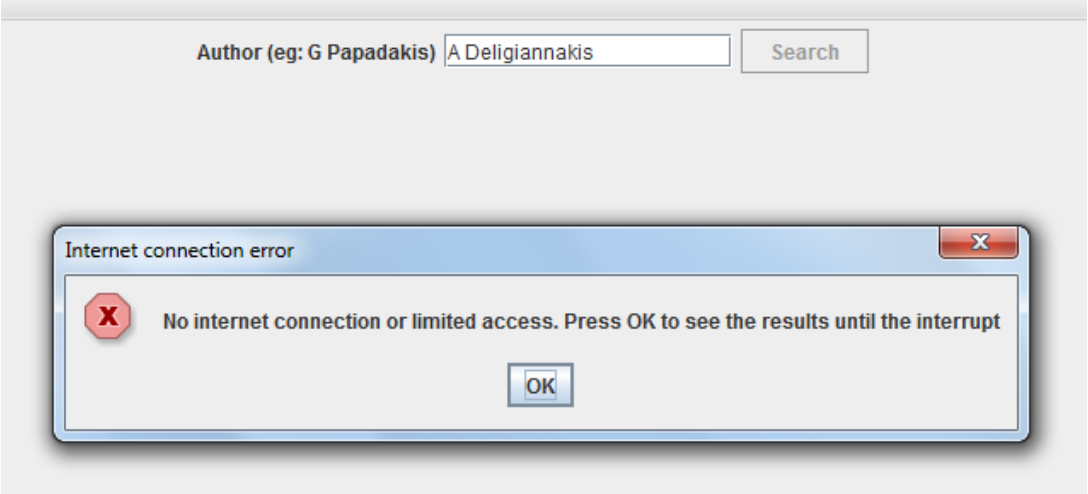


Εικόνα 13β - Παράθυρο που ενημερώνει τον χρήστη πως μπορεί να εισάγει μόνο αριθμούς στα ζητούμενα πεδία

Όταν πατηθεί το κουμπί OK, ο πίνακας με τις δημοσιεύσεις καθώς επίσης και τα αποτελέσματα ανανεώνονται ανταποκρινόμενα στο νέο χρονικό εύρος των αναφορών που εισήγαγε ο χρήστης.

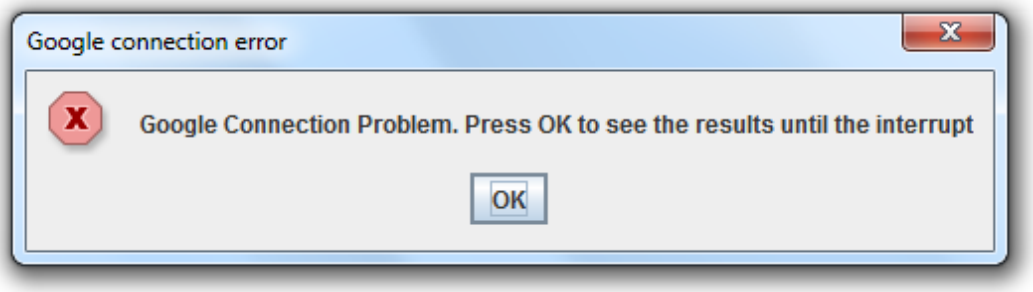
Περιπτώσεις τερματισμού του συστήματος χωρίς την εντολή του χρήστη

Υπάρχουν δύο περιπτώσεις κατά τις οποίες το σύστημα θα τερματίσει την αναζήτηση των δημοσιεύσεων και θα εμφανίσει τα αποτελέσματα που έχουν αποθηκευτεί μέχρι εκείνη την ώρα.
Η πρώτη περίπτωση είναι να υπάρξει διακοπή της σύνδεσης στο διαδίκτυο.
Σε αυτή την περίπτωση, το σύστημα εμφανίζοντας ένα παράθυρο ενημερώνει τον χρήστη για το πρόβλημα που προέκυψε και εμφανίζει τα αποτελέσματα της αναζήτησης μέχρι εκείνη την στιγμή.



Εικόνα 14α – Παράθυρο που ενημερώνει τον χρήστη πως υπάρχει πρόβλημα της σύνδεσης διαδικτύου

Η δεύτερη περίπτωση είναι ο διακομιστής του Google Scholar να κλειδώσει το σύστημα καθώς πραγματοποιήθηκαν προς αυτόν πολλά αιτήματα σε σύντομο χρονικό διάστημα.
Όπως πριν, έτσι και τώρα το σύστημα εμφανίζει ένα παράθυρο το οποίο ενημερώνει τον χρήστη για το πρόβλημα που προέκυψε και εμφανίζει τα αποτελέσματα της αναζήτησης μέχρι εκείνη την στιγμή.



Εικόνα 14β - Παράθυρο που ενημερώνει τον χρήστη πως υπάρχει πρόβλημα της σύνδεσης στο διακομιστή του Google Scholar

Η λύση του τελευταίου προβλήματος είναι το σύστημα να εκτελεστεί σε σύνδεση διαδικτύου με διαφορετική IP από την τελευταία κατά την οποία προέκυψε το κλείδωμα.
Με αυτόν τον τρόπο και αλλάζοντας cookies κατά τις αναζητήσεις (πραγματοποιείται αυτόματα από συνάρτηση που υπάρχει στον κώδικα του προγράμματος η οποία παράγει cookies για τον διακομιστή του Scholar καθώς και αυτόν της Google), το πρόγραμμα εκτελείτε κανονικά χωρίς να υπάρχει πρόβλημα.

Σχολιασμός και μελλοντική επεκτασιμότητα εργασίας

Σχολιασμός

Η διπλωματική εργασία αυτή ολοκληρώθηκε μέσα από μία διαδικασία αναζήτησης, εκμάθησης και εφαρμογής νέων αλλά και παλαιών τεχνολογιών.
Επιπλέον, με την ολοκλήρωση της υλοποίησης του συστήματος που διαπραγματεύεται η παρούσα διπλωματική εργασία, καταφέρνουμε να εμφανίσουμε και να υπολογίσουμε στατιστικά στοιχεία για της αναφορές επί των δημοσιεύσεων των επιστημονικών συγγραφέων, τα οποία δεν υπολογίζονται από άλλες αντίστοιχες εφαρμογές (Publish or Perish) αλλά ούτε και από το ίδιο το Google Scholar.
Και στις δύο περιπτώσεις η εμφάνιση και ο υπολογισμός των παραπάνω στατιστικών στοιχείων υπολογίζεται μόνο πάνω στις δημοσιεύσεις του εισαχθέντος συγγραφέα αλλά όχι και στις αναφορές που έχει η κάθε μία από αυτές.
Τέλος, με την εξαγωγή των αυτοαναφορών από τον συνολικό αριθμό των αναφορών και με την εμφάνιση μόνο των ετεροαναφορών, δίνουμε μία αντικειμενικότερη και περισσότερο ρεαλιστική εικόνα για την αξία της κάθε δημοσίευσης καθώς και των συγγραφέων τους.

Μελλοντικές Επεκτάσεις

Η εργασία αυτή μπορεί να επεκτείνει της δυνατότητες της εάν προστεθούν ορισμένα καινούργια στοιχεία.
Τα κυριότερα εξ' αυτών είναι:

1. Δημιουργία και υλοποίηση κουμπιού το οποίο θα επιτρέπει την συνέχιση της αναζήτησης στην περίπτωση που υπάρξει διακοπή της σύνδεσης στο διαδίκτυο
   Η εφαρμογή λόγω της χρονοκαθυστέρησης για αναζητήσεις με συγγραφείς οι οποίοι έχουν μεγάλο αριθμό δημοσιεύσεων, αργεί σημαντικά να εμφανίσει τα αποτελέσματα. Στην περίπτωση λοιπόν που για τον οποιονδήποτε λόγο υπάρξει διακοπή της σύνδεσης στο διαδίκτυο (πχ επανεκκίνηση δρομολογητή ή κλείδωμα από τον διακομιστή του google scholar) τότε υπάρχει και διακοπή του προγράμματος με αποτέλεσμα ο χρήστης να είναι αναγκασμένος εκτελέσει εκ νέου την εφαρμογή από την αρχή.
   Η τοποθέτηση όμως ενός κουμπιού όπως περιγράφεται παραπάνω, θα επέτρεπε την συνέχιση της αναζήτησης από το σημείο το οποίο σταμάτησε, με αποτέλεσμα να γίνει σημαντική εξοικονόμηση χρόνου.

2. Δυνατότητα ο χρήστης να συνδέεται στο σύστημα διαχείρισης βάσεων δεδομένων της MySQL που βρίσκεται σε κάποιον άλλον υπολογιστή και όχι απαραίτητα στο δικό του.
   Ουσιαστικά δηλαδή να μην υποχρεώνει τον χρήστη να έχει ήδη εγκατεστημένη η να εγκαθιστά την MySQL για να εκτελέσει την εφαρμογή αλλά να έχει την ελευθερία να αποθηκεύει τα αποτελέσματα της αναζήτησής του σε βάση δεδομένων άλλου υπολογιστή.

3. Να γίνει το πρόγραμμα κατανεμημένο. Αυτή θα ήταν και η σημαντικότερη επέκταση στην εφαρμογή.
   Η μεγαλύτερη αδυναμία του προγράμματος είναι η αργή εμφάνιση των αποτελεσμάτων λόγω της ηθελημένης χρονοκαθυστέρησης που εισάγεται για την αποφυγή του κλειδώματος από τον google scholar.
   Στην περίπτωση όμως που το σύστημα γινόταν κατανεμημένο (πχ με την υποστήριξη του κατανεμημένου συστήματος αρχείων hadoop) η εκτέλεση του κάθε αιτήματος προς τον διακομιστή του google scholar θα γινόταν από διαφορετικό κόμβο. Ο κάθε παραπάνω κόμβος θα μείωνε σημαντικά το χρόνο πραγματοποίησης της αναζήτησης.
   Για παράδειγμα, μία αναζήτηση δημοσιεύσεων από συγγραφέα που η εφαρμογή θα χρειαζόταν πέντε (5) ώρες για την εκτέλεσή του, με την ύπαρξη δέκα (10) κόμβων να εκτελούν τα αιτήματα παράλληλα, ο χρόνος αυτός θα μειωνόταν στα 30 λεπτά (300λεπτά/10κόμβοι =30 λεπτά)

Αναφορές

[1] sql: http://el.wikipedia.org/wiki/SQL

http://www.databasedev.co.uk/why-use-sql.html

[2] java: http://el.wikipedia.org/wiki/Java

Java me UML, Lervik Havdal, εκδόσεις Κλειδάριθμος

[3] mysql: http://www.internetnow.gr/node/70

http://dnhost.gr/kb/article/AA-00274/0/%CE%A4%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%B7-MySQL-%CE%B2%CE%AC%CF%83%CE%B7-%CE%B4%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89%CE%BD.html

Θεμελιώδεις Αρχές ΣΔΒΔ – R. Elmasri – S.B. Navathe, εκδόσεις Δίαυλος

[4] eclipse: www.eclipse.org

http://users.teicrete.gr/taxd/02/notes/eclipse/abouteclipsesdk.htm

[5] jdbc: http://el.wikipedia.org/wiki/JDBC

http://docs.oracle.com/javase/tutorial/jdbc/index.html

[6] google scholar: http://scholar.google.gr/intl/el/scholar/about.html

http://www.slideshare.net/PanagiotaAltanopoulou/google-scholar-h

Παραρτημα 1

Ο κώδικας SQL που δημιουργεί την βάση δεδομένων και τους πίνακές της

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `scholardb` DEFAULT CHARACTER SET utf8 ;
USE `scholardb` ;


CREATE  TABLE IF NOT EXISTS `scholardb`.`papers` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `title` VARCHAR(255) NULL DEFAULT NULL ,
  `authors` VARCHAR(255) NULL DEFAULT NULL ,
  `publication` VARCHAR(255) NULL DEFAULT NULL ,
  `yearOfPublication` INT(11) NULL DEFAULT NULL ,
  `citations` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


CREATE  TABLE IF NOT EXISTS `scholardb`.`authors` (
  `authorID` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(255) NULL DEFAULT NULL ,
  `paperID` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`authorID`) ,
  INDEX `paperID_FK2_idx` (`paperID` ASC) ,
  CONSTRAINT `paperID_FK2`
    FOREIGN KEY (`paperID` )
    REFERENCES `scholardb`.`papers` (`id` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


CREATE  TABLE IF NOT EXISTS `scholardb`.`citations` (
  `citationID` INT(11) NOT NULL AUTO_INCREMENT ,
  `title` VARCHAR(255) NULL DEFAULT NULL ,
  `authors` VARCHAR(255) NULL DEFAULT NULL ,
  `yearOfPublication` INT(11) NULL DEFAULT NULL ,
  `paperID` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`citationID`) ,
  INDEX `paperID_FK1_idx` (`paperID` ASC) ,
  CONSTRAINT `paperID_FK1`
    FOREIGN KEY (`paperID` )
    REFERENCES `scholardb`.`papers` (`id` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

```
CREATE  TABLE IF NOT EXISTS `scholardb`.`citationsauthors` (
  `citationAuthorID` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(255) NULL DEFAULT NULL ,
  `citationID` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`citationAuthorID`) ,
  INDEX `citationID_FK3_idx` (`citationID` ASC) ,
  CONSTRAINT `citationID_FK3`
    FOREIGN KEY (`citationID` )
    REFERENCES `scholardb`.`citations` (`citationID` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Παραρτημα 2

Ο κώδικας JAVA που υλοποιεί το σύστημα

Κλάση CitationsBetweenYears:

```
import javax.swing.*;
import java.awt.*;
import java.sql.*;
import java.util.*;


public class CitationsBetweenYears extends JPanel {
    private Connection con = null;
    private Statement query = null;
    private ResultSet rs = null;

    public CitationsBetweenYears(){
    }


    public CitationsBetweenYears(int p1id, String flag, String from,
String to){
        try{
            con = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
            query = con.createStatement();

            if(flag=="Show All Citations"){//emfanisi olwn twn
apotelesmatwn
                rs = query.executeQuery("SELECT citations.citationID,
citations.title, citations.authors, citations.yearOfPublication" +
                        " from scholardb.papers, scholardb.citations
" +
                        "WHERE papers.id="+p1id+" and
papers.id=citations.paperID and (citations.yearOfPublication >=
"+from+" and citations.yearOfPublication <= "+to+") ");

            }
            else if(flag=="Hide Self Citations"){
                rs = query.executeQuery("SELECT pp2.citationID,
pp2.title, pp2.authors, pp2.yearOfPublication " +
                        "from papers, citations as pp2 " +
                        "where papers.id=pp2.paperID and
pp2.yearOfPublication>= "+from+" and pp2.yearOfPublication<= "+to+"
and papers.id="+p1id+" and pp2.citationID not in "+
                            "(select p2.citationID " +
                            "from citations as p2, citationsauthors "
+
                            "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                                "(select name " +
                                "from authors " +
                                "where authors.paperID = papers.id) "
+
                        "group by p2.citationID " +
                        "having count(*) > 0 )");

            }
            else if(flag=="Show Self Citations"){
```

```
                rs = query.executeQuery("SELECT pp2.citationID,
pp2.title, pp2.authors, pp2.yearOfPublication " +
                        "from papers, citations as pp2 " +
                        "where papers.id=pp2.paperID and
pp2.yearOfPublication>= "+from+" and pp2.yearOfPublication<= "+to+"
and papers.id="+p1id+" and pp2.citationID in "+
                            "(select p2.citationID " +
                            "from citations as p2, citationsauthors "
+
                            "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                                "(select name " +
                                "from authors " +
                                "where authors.paperID = papers.id) "
+
                        "group by p2.citationID " +
                        "having count(*) > 0 )");
            }

            ResultSetMetaData md = rs.getMetaData();
            int columnCount = md.getColumnCount();

            Vector columns = new Vector(columnCount);

            Vector<RowsOfCitationsTable> rowCitedByPaper= new
Vector<RowsOfCitationsTable>();
            //store row data

            while(rs.next()){
                RowsOfCitationsTable rowTable = new
RowsOfCitationsTable();

                rowTable.citedByPaper=rs.getString(2);
                rowTable.citedByAuthors=rs.getString(3);
                rowTable.yearOfPublication=rs.getString(4);

                rowCitedByPaper.add(rowTable);
            }

            final JTable table = new JTable(new
CitationsTable(rowCitedByPaper));
            table.setPreferredScrollableViewportSize(new
Dimension(1200, 700));
            table.setFillsViewportHeight(true);
            table.setVisible(true);
            table.validate();

table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);

table.getColumnModel().getColumn(0).setPreferredWidth(670);

table.getColumnModel().getColumn(1).setPreferredWidth(400);

table.getColumnModel().getColumn(2).setPreferredWidth(130);

            JScrollPane scrollPane = new JScrollPane(table);
            add(scrollPane);

        }
```

```
        catch(SQLException sqle){
            System.out.println(sqle);
            sqle.printStackTrace();
        }
    }

 public void paintCitedByTable(JFrame frame, int p1id, String flag,
String from, String to){
        JScrollPane scrollPane = new JScrollPane(new
CitationsBetweenYears(p1id, flag, from, to));
        frame.getContentPane().add(scrollPane);
    }
}
```

Κλάση CitationsFetch:

```
import javax.swing.*;
import java.awt.*;
import java.sql.*;
import java.util.*;


public class CitationsFetch extends JPanel {
    private Connection con = null;
    private Statement query = null;
    private ResultSet rs = null;

    public CitationsFetch(){
    }


    public CitationsFetch(int p1id, String flag){
        try{
            con = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
            query = con.createStatement();

            if(flag=="Show All Citations"){
                rs = query.executeQuery("SELECT * FROM
scholardb.citations WHERE paperID="+p1id);//emfanisi olwn twn
apotelesmatwn
            }
            else if(flag=="Hide Self Citations"){   //emfanisi mono
twn apotelesmatwn pou den einai self-citations
                rs = query.executeQuery("SELECT pp2.citationID,
pp2.title, pp2.authors, pp2.yearOfPublication " +
                        "from scholardb.papers, scholardb.citations
as pp2 " +
                        "where papers.id=pp2.paperID  and
papers.id="+p1id+" and pp2.citationID not in "+
                            "(select p2.citationID " +
                            "from scholardb.citations as p2,
scholardb.citationsauthors " +
                            "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                                "(select name " +
```

```
                                    "from scholardb.authors " +
                                    "where authors.paperID = papers.id) "
+
                                    "group by p2.citationID " +
                                    "having count(*) > 0 )");
            }
            else if(flag=="Show Self Citations"){   //emfanisi mono
twn apotelesmatwn pou einai self-citations
                rs = query.executeQuery("SELECT pp2.citationID,
pp2.title, pp2.authors, pp2.yearOfPublication " +
                        "from scholardb.papers, scholardb.citations
as pp2 " +
                        "where papers.id=pp2.paperID  and
papers.id="+p1id+" and pp2.citationID in "+
                            "(select p2.citationID " +
                            "from scholardb.citations as p2,
scholardb.citationsauthors " +
                            "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                                    "(select name " +
                                    "from scholardb.authors " +
                                    "where authors.paperID = papers.id) "
+
                                    "group by p2.citationID " +
                                    "having count(*) > 0 )");
            }

            ResultSetMetaData md = rs.getMetaData();
            int columnCount = md.getColumnCount();

            Vector columns = new Vector(columnCount);

            Vector<RowsOfCitationsTable> rowCitedByPaper= new
Vector<RowsOfCitationsTable>();
            //store row data

            while(rs.next()){
                RowsOfCitationsTable rowTable = new
RowsOfCitationsTable();

                rowTable.citedByPaper=rs.getString(2);
                rowTable.citedByAuthors=rs.getString(3);
                rowTable.yearOfPublication=rs.getString(4);

                rowCitedByPaper.add(rowTable);
            }

            final JTable table = new JTable(new
CitationsTable(rowCitedByPaper));
            table.setPreferredScrollableViewportSize(new
Dimension(1200, 700));
            table.setFillsViewportHeight(true);
            table.setVisible(true);
            table.validate();

table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);

table.getColumnModel().getColumn(0).setPreferredWidth(670);
```

```
table.getColumnModel().getColumn(1).setPreferredWidth(400);

table.getColumnModel().getColumn(2).setPreferredWidth(130);


            JScrollPane scrollPane = new JScrollPane(table);
            add(scrollPane);

        }
        catch(SQLException sqle){
            System.out.println(sqle);
            sqle.printStackTrace();
        }
    }

 public void paintCitedByTable(JFrame frame, int p1id, String flag){
        JScrollPane scrollPane = new JScrollPane(new
CitationsFetch(p1id, flag));
        frame.getContentPane().add(scrollPane);
    }
}
```

Κλάση CitationsTable:

```
import java.util.Vector;
import javax.swing.table.AbstractTableModel;


public class CitationsTable extends AbstractTableModel{
    Vector<RowsOfCitationsTable> v;

    public CitationsTable(Vector<RowsOfCitationsTable> p) {
        v = p;
    }

    public int getColumnCount() {
        return 3;
    }

    public Object getValueAt(int row, int column){
        if (column==0)
            return  (v.get(row)).citedByPaper;
        else if (column==1)
            return(v.get(row)).citedByAuthors;
        else if (column==2)
            return(v.get(row)).yearOfPublication;
        return null;
    }

    public int getRowCount(){
        return v.size();
    }

    public String  getColumnName(int columnIndex) {
        if (columnIndex==0)
            return "Cited by "+v.size()+" Papers";
```

```
        else if (columnIndex==1)
            return "Cited By Authors";
        else if (columnIndex==2)
            return "Year of Publication";
        return null;
    }
}
```

Κλάση CitationsWindow:

```
import java.awt.Frame;
import java.awt.event.*;
import javax.swing.*;


public class CitationsWindow extends JFrame{
    public int p1ID;
    public int flag2;
    private String from;
    private String to;
    public CitationsWindow(String title, int p1id, int FLAG2, String
From, String To){
        from = From;
        to = To;
        flag2=FLAG2;
        setTitle(title);

        if(flag2==1){
             new CitationsFetch().paintCitedByTable(this, p1id, "Show
All Citations");
        }else if(flag2==2){
             new CitationsBetweenYears().paintCitedByTable(this,
p1id, "Show All Citations", from, to);
        }


     p1ID=p1id;
     SelfCitListener theSelfCitationListener = new SelfCitListener();
     JMenu selfCitMenu = new JMenu("Self Citations");
        JMenuItem dbMenuItem = new JMenuItem("Show Self Citations");
        selfCitMenu.add(dbMenuItem);
        dbMenuItem.addActionListener(theSelfCitationListener);

        dbMenuItem = new JMenuItem("Hide Self Citations");
        selfCitMenu.add(dbMenuItem);
        dbMenuItem.addActionListener(theSelfCitationListener);

        dbMenuItem = new JMenuItem("Show All Citations");
        selfCitMenu.add(dbMenuItem);
        dbMenuItem.addActionListener(theSelfCitationListener);

        JMenuBar menuBar = new JMenuBar();
        menuBar.add(selfCitMenu);

        setJMenuBar(menuBar);
        setSize(800, 600);
        setExtendedState(Frame.MAXIMIZED_BOTH);
     setVisible(true);
     setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
```

```
    }


    private class SelfCitListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {
          String command = event.getActionCommand();
          if (command.equals("Show Self Citations")){

              repaintCitedByTable("Show Self Citations");
             }
          else if (command.equals("Hide Self Citations")){

              repaintCitedByTable("Hide Self Citations");
          }
          else if (command.equals("Show All Citations")){

              repaintCitedByTable("Show All Citations");
          }
        }
      }

    public void repaintCitedByTable(String flag){

        getContentPane().removeAll();
        repaint();

        if(flag2==1){

            new CitationsFetch().paintCitedByTable(this, p1ID, flag);
        }else if(flag2==2){

            new CitationsBetweenYears().paintCitedByTable(this, p1ID,
flag, from, to);
        }

        setVisible(true);
    }
}
```

Κλάση Core:

```
import java.net.*;
import java.io.*;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.util.*;
import java.sql.*;
import javax.swing.JLabel;
import javax.swing.JOptionPane;


public class Core {

    private int nofConnections = 0; //counter for number of
connections
    private String scholarCookie;   //To scholar cookie
    private String googleCookie;        //To google cookie
    private String myText, P1title, P2title, P1Author, P2Author;
//epithymito html keimeno gia to kathe paper, titlos Paper1
```

```
    private int P1yearOfPublication=0, P2yearOfPublication=0;
    private MainWindow myMainWindow;
    private int delayPerHit=4;
    private int delaySecs=0;
    private String request ="";
    private int numOfResults=0;
    private ProgressBars pbars;
    public Thread newThread;


//**********************************************************************
**********************************************************************
**********//

    public  void parseResults(BufferedReader in) {

        String regexMyText="(<div class=\"gs_fl\">)(.*?)(\">Import
into BibTeX</a>)";
        Pattern patternMyText=Pattern.compile(regexMyText);

        String regex = "Cited\\s*by\\s*\\d+</a>";   //link pou mas
deixnei posa papers to kanoun cite
        Pattern pattern = Pattern.compile(regex);

        String regexBibTex="(href=\"/scholar.bib)(.*?)(\">Import into
BibTeX</a>)"; //link pou tha paroume ta stoixeia tou paper
        Pattern patternBibTex=Pattern.compile(regexBibTex);

        String
regexCitedBy="(href=\"/scholar\\?cites=)(.*?)(\">Cited\\s*by\\s*\\d+<
/a>)";  //link pou tha paroume to poia papers to kanoun cite
        Pattern patternCitedBy=Pattern.compile(regexCitedBy);

        String bibTexLink = null, citedByLink = null, citedBy = null,
line = "";
        Matcher matcher, matcher2;

        try {
            while ((line = in.readLine()) != null){

                matcher2=patternMyText.matcher(line);
                while(matcher2.find()){

                    myText = matcher2.group();
                    //System.out.println(myText);

                    //NumOf Citations
                    matcher = pattern.matcher(myText);//Creates a
matcher that will match the given input against this pattern
                    if(matcher.find()) {//Attempts to find the next
subsequence of the input sequence that matches the pattern
                        citedBy = matcher.group();//Returns the input
subsequence matched by the previous match
                        citedBy =
citedBy.replaceFirst("Cited\\s*by\\s*","");//Afairei to "Cited by"
kai
                        citedBy = citedBy.replaceFirst("</a>","");//
to </a> gia na krathsei mono ton arithmo twn citations

                        pbars.paper2Counter=0;
```

```
pbars.pb2.setMaximum(Integer.parseInt(citedBy));

                    }
                    else{
                        pbars.paper2Counter=0;
                        pbars.pb2.setMaximum(1);
                        citedBy = "0";
                    }

                    //Paper 1 BibTex link
                    matcher=patternBibTex.matcher(myText);
                    if(matcher.find()){

bibTexLink="http://scholar.google.com/scholar.bib"+matcher.group(2);//
//synenwsi periexomenou tou (.*) sto teliko link
                        bibTexLink=bibTexLink.replaceAll("amp;", "");

                        getPaperInfo(bibTexLink, 0,
Integer.parseInt(citedBy)); //Apothikeyw ta stoixeia tou Paper 1 sta
antistoixa vectors
                        //System.out.println("This HIT is from
parseResults to take Paper1 BibTex"+"\n");
                    }

                    //Papers 2 link
                    matcher=patternCitedBy.matcher(myText);
                    if(matcher.find()){

citedByLink="http://scholar.google.com/scholar?cites="+matcher.group(
2);
                        citedByLink=citedByLink.replaceAll("amp;",
"");

citedByLink=citedByLink.replaceAll("&num=\\d+", "&num=100");

citedByLink=citedByLink.replaceAll("&oe=ASCII", "");
                        //System.out.println("citedByLink AFTER:
"+citedByLink);

                        selfCitation(citedByLink, citedBy);

                    }
                    pbars.p1Increment();
                }
            }
        } catch (IOException ioe) {
          System.err.println(ioe);
          System.exit(1);
        }

    }

//*****************************************************************
*****************************************************************
**********//

   public void selfCitation(String citedByLink, String citedBy){

       citedByLink = citedByLink+"&start=0";
```

```
        for(int i=0; i<Integer.parseInt(citedBy) ; i+=100){

        citedByLink = citedByLink.replaceAll("start=\\d+",
"start="+i);

        BufferedReader in = openConnection(citedByLink);
        //System.out.println("This HIT is from selfCitation to open
Cited By link"+"\n");
        String regexBibTex="(href=\"/scholar.bib)(.*?)(\">Import into
BibTeX</a>)";//to xwrizei se tria kommatia
        Pattern patternBibTex=Pattern.compile(regexBibTex);
        String bibTexLink = null, line = "";
        Matcher matcher;

        try {
             while ((line = in.readLine()) != null){

                   matcher=patternBibTex.matcher(line);
                   while(matcher.find()){    //find BibTex link

bibTexLink="http://scholar.google.com/scholar.bib"+matcher.group(2);
                      bibTexLink=bibTexLink.replaceAll("amp;", "");

                     pbars.p2Increment();
                       getPaperInfo(bibTexLink, 1, 0);
                     //System.out.println("This HIT is from
parseResults to take Paper2 BibTex\n");
                    }
                 }
             } catch (IOException ioe) {
                 System.err.println(ioe);
                 //System.exit(1);
             }
        }

    }


//*****************************************************************
*****************************************************************
**********//


   public void getPaperInfo(String bibTexLink, int flag, int
numOfCitations){

         String tempLine;
         BufferedReader in = openConnection(bibTexLink);
         String ttitle = null, tauthor = null, tpublication = "Not
Available";
         int tyear = 0;

         ScholarDB scholarDB = new ScholarDB();
         Connection conn = new
ScholarDB().openConnection(readLoginData("user"),
readLoginData("pass"),2);


         try {
             while((tempLine=in.readLine())!=null){
```

```
                if(tempLine.contains("  title")){
                    ttitle=tempLine.replaceAll("  title=\\{", "");
                    ttitle=ttitle.replace("},", "");
                    ttitle=ttitle.replace("}", "");
                    ttitle=ttitle.replace("{", "");
                    ttitle=ttitle.replace("'", " ");
                //  System.out.println("Title: "+ttitle);

                    if(ttitle.length()>250){
                        System.out.println("Title: "+ttitle);
                        System.out.println("Title SIZE:
"+ttitle.length());
                        ttitle=ttitle.substring(0, 250);
                    }


                    if(flag==0){//periptwsh author gia to PAPER1
                        P1title= ttitle;
                        }
                    else{//periptwsh author gia to PAPER2
                        P2title= ttitle;
                    }


                }else if(tempLine.contains("  author")){
                    tauthor=tempLine.replaceAll("  author=\\{", "");
                    tauthor=tauthor.replace("},", "");
                    tauthor=tauthor.replace("}", "");
                    tauthor=tauthor.replace("{", "");
                    tauthor=tauthor.replace(",", "");
                    tauthor=tauthor.replace("'", " ");
//                  tauthor=tauthor.replace(" ", "`");
                    tauthor=tauthor.replace("and", ", ");
                    tauthor=tauthor.replace("\\", "");

                    if(flag==0){//periptwsh author gia to PAPER1
                        P1Author= tauthor;
                        }
                    else{//periptwsh author gia to PAPER2
                        P2Author= tauthor;
                    }


                }else if((tempLine.contains("
booktitle")||tempLine.contains("  journal"))&&(flag==0)){
                    tpublication=tempLine.replaceAll("
booktitle=\\{", "");
                    tpublication=tpublication.replaceAll("
journal=\\{", "");
                    tpublication=tpublication.replace("}", "");
                    tpublication=tpublication.replace(",", "");
                    tpublication=tpublication.replace("'", " ");
//                  tpublication=tpublication.replaceAll("[^A-Za-z0-
9\\s.,-/:&?()]+", "");
//                  System.out.println("Publication: "+tpublication);

                    if(tpublication.length()>250){
                        System.out.println("Publication:
"+tpublication);
                        System.out.println("Publication SIZE:
"+tpublication.length());
```

```
                        tpublication=tpublication.substring(0, 250);
                    }

                }else if(tempLine.contains("  year")){
                    tempLine=tempLine.replaceAll("  year=\\{", "");
                    tempLine=tempLine.replace("}", "");
                    tempLine=tempLine.replace(",", "");

                    tyear = Integer.parseInt(tempLine);

                    if(flag==0){//periptwsh author gia to PAPER1
                        P1yearOfPublication= tyear;
                        }
                    else{//periptwsh author gia to PAPER2
                        P2yearOfPublication= tyear;
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }


        if(flag==0){
            scholarDB.insertP1(P1title, P1Author, tpublication,
numOfCitations, tyear);
            splitAuthors(P1title, tauthor, "papers");
//          myMainWindow.rePaint(1, null, null);

        }
        else if(flag==1){

            scholarDB.insertP2(P1title, P2title, P2Author, tyear);
            splitAuthors(P2title, tauthor, "citations");
        }

        try {
            conn.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        tyear=0;
    }



//*********************************************************************
*********************************************************************
***********//


   public void splitAuthors(String Ptitle, String authors, String
paper){

     ScholarDB scholarDB = new ScholarDB();
     String[] temp;
     String delimiter = "\\s*,\\s*";
     temp = authors.split(delimiter);

     for(int i=0; i < temp.length ; i++){
```

```
            scholarDB.insertAuthors(Ptitle, paper,
removeSecondName(temp[i]));
            //System.out.println(temp[i]);
        }
    }

    public String removeSecondName(String author){

        String[] temp;
        String result=null;
        String delimiter = "\\.";
        temp = author.split(delimiter);
        int j=0;

        for(int i=0; i < temp.length ; i++){
            if(j==0) result = temp[i]+".";
            j++;
        }

            return result;
    }

//*********************************************************************
**********************************************************************
***********//


  public BufferedReader openConnection(String request){

      nofConnections++;
      System.out.println("Hit No: "+ nofConnections);

      if(nofConnections%delayPerHit==0){

          System.out.println("\nWaiting...\n");


          try{
              Thread.sleep(delaySecs);
          }catch (InterruptedException ie){
              System.out.println(ie.getMessage());
          }
      }

      scholarCookie = "GSP=ID="+cookieGenerator()+":CF=4";
      googleCookie = "PREF=ID="+cookieGenerator();


        BufferedReader in = null;
        try {
            HttpURLConnection con = (HttpURLConnection) (new
URL(request)).openConnection();

            con.setRequestProperty("User-Agent","Mozilla/5.0 (Windows
NT 6.1; rv:15.0) Gecko/20120716 Firefox/15.0a2");
            con.setRequestProperty("Cookie",
"GPC=FW=0:GHV=0:SG=0:TS=1:TV=0:SIG=ROcjJYtWPqQ84o7y; " +
                    "rememberme=false; "+googleCookie+":LD=en:CR=2:"
+
                    "TM=1324488803:LM=1324488836:S=Ay3RYbVTIbxBqGTL;
"+scholarCookie);
```

```
            in = new BufferedReader(new
InputStreamReader(con.getInputStream()));

        }catch (UnknownHostException  e1) {
            System.err.println(e1);
            System.out.println("Internet Connection Problem");
            JLabel errorFields = new JLabel("No internet connection
or limited access. Press OK to see the results until the interrupt");
            JOptionPane.showMessageDialog(null, errorFields,
"Internet connection error", JOptionPane.ERROR_MESSAGE);
              myMainWindow.exeButton.setEnabled(true);
              pbars.frame.setVisible(false);
              myMainWindow.showResults();
              newThread.stop();

        }catch (SocketException  se1) {
            System.err.println(se1);
            System.out.println("Internet Connection Problem");
            JLabel errorFields = new JLabel("No internet connection
or limited access. Press OK to see the results until the interrupt");
            JOptionPane.showMessageDialog(null, errorFields,
"Internet connection error", JOptionPane.ERROR_MESSAGE);
              myMainWindow.exeButton.setEnabled(true);
              pbars.frame.setVisible(false);
              myMainWindow.showResults();
              newThread.stop();

        }catch (IOException ioe) {
          System.err.println(ioe);
          System.out.println("Google Connection Problem");
            JLabel errorFields = new JLabel("Google Connection
Problem. Press OK to see the results until the interrupt");
            JOptionPane.showMessageDialog(null, errorFields, "Google
connection error",JOptionPane.ERROR_MESSAGE);
              myMainWindow.exeButton.setEnabled(true);
//            pbars.frame.setVisible(false);
              myMainWindow.showResults();
//            newThread.stop();
        }
        return in;
  }


  public int findNumOfResults(String link){

      BufferedReader in = openConnection(link);

      String regexNoFResults = "\\s*\\d+\\s*results";
      Pattern patternNoFResults = Pattern.compile(regexNoFResults);
      String results=null, line = "";
      Matcher matcher;

    try {
        while ((line = in.readLine()) != null){

            matcher = patternNoFResults.matcher(line);
            while(matcher.find()) {
                results = matcher.group();
                results = results.replaceAll("\\s*", "");
                results = results.replaceAll("\\s*results", "");
```

```
                System.out.println("Number of Results: "+results);
            }
        }

   } catch (IOException ioe) {
      System.err.println(ioe);
      System.exit(1);
    }

      if(results==null){
          numOfResults=0;
          return 0;
      }else{
         numOfResults=Integer.parseInt(results);
         return Integer.parseInt(results);
      }

  }


 public static String cookieGenerator(){

       Random r = new Random();
       String alphabet = "0123456789abcdef";
       String cookie = "";

       for (int i = 0; i < 16; i++) {
           cookie = cookie +
alphabet.charAt(r.nextInt(alphabet.length()));
       }
      return cookie;
  }

 public void makeRequest(String author, MainWindow frame){
     myMainWindow=frame;
     String[] temp;
     String authorName = "";
     String delimiter = "\\s+";
     temp = author.split(delimiter);

       for(int i=0; i < temp.length ; i++){
           if(authorName=="")
               authorName = temp[i];
           else
               authorName = authorName + "+" + temp[i];
           }

     request =
"http://scholar.google.com/scholar?as_q=&as_sauthors=%22"+authorName+
"%22&as_publication=&as_ylo=&as_yhi=&btnG=&hl=en&as_sdt=0%2C5";


     int numOfResults = findNumOfResults(request);

     new DelayWindow(this,numOfResults, myMainWindow).requestFocus();

 }

 public void search(){

     pbars = new ProgressBars(myMainWindow, newThread);
```

```
     pbars.pb1.setMaximum(numOfResults);

     request = request + "&num=100&start=0";


        for(int i=0; i<numOfResults; i+=100){

            request = request.replaceAll("start=\\d+", "start="+i);
            BufferedReader in = openConnection(request);

            parseResults(in);

            try {
                in.close();
            } catch (IOException e) {

                e.printStackTrace();
            }
        }
        pbars.frame.setVisible(false);
        myMainWindow.showResults();
 }


    public void setDelay(int[] delayData){

        delaySecs = delayData[0];
        delayPerHit = delayData[1];
    }

    public void writeLoginData(String user, String pass){

        try{
            // Create file
            FileWriter fstream = new FileWriter("LoginData.txt");
            BufferedWriter out = new BufferedWriter(fstream);
            out.write(user+"\n"+pass);
            //Close the output stream
            out.close();
            }catch (Exception e){//Catch exception if any
                System.err.println("Error: " + e.getMessage());
            }
    }


    public String readLoginData(String whatToReturn){
        String user="";
        String pass="";
        int temp=0;
        try{
              FileInputStream fstream = new
FileInputStream("LoginData.txt");
              DataInputStream in = new DataInputStream(fstream);
              BufferedReader br = new BufferedReader(new
InputStreamReader(in));
              String strLine;
              while ((strLine = br.readLine()) != null)   {

                  if (temp==0) user = strLine;

                  if (temp==1)pass = strLine;
```

```
                temp++;
            }
            //Close the input stream
            in.close();
              }catch (Exception e){//Catch exception if any
            System.err.println("Error: " + e.getMessage());
            }

        if(whatToReturn=="user"){
            return user;
        }
        else {
            return pass;
        }
    }
} // end of class
```

Κλάση DelayWindow:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class DelayWindow extends JFrame{

    private JLabel delay;
    private JLabel perHit;
    private JLabel recommendations;
    private JTextField delayField;
    private JTextField perHitField;
    private JButton okButton;
    private JPanel contentPane;
    private String delayms;
    private String perHitNo;
    private Core myCore;
    private int numOfResults;
    private JButton cancelButton;
    private MyThread myThread;
    private MainWindow myMainWindow;


    public DelayWindow(Core theCore, int numOfRes, MainWindow mWin){
        myCore=theCore;
        myMainWindow = mWin;
        numOfResults = numOfRes;
        myThread = new MyThread(myCore);
        createWindow();
    }


    public void createWindow(){
        delay = new JLabel();
        perHit = new JLabel();
        recommendations = new JLabel();
        delayField = new JTextField();
```

```
        delayField.requestFocus();
        delayField.setText("18");
        perHitField = new JTextField();
        perHitField.setText("1");
        okButton = new JButton();
        cancelButton = new JButton();
        cancelButtonListener theCancelButtonListener = new
cancelButtonListener();
        cancelButton.addActionListener(theCancelButtonListener);
        OKButtonListener theButtonListener = new OKButtonListener();
        okButton.addActionListener(theButtonListener);

        contentPane = (JPanel)this.getContentPane();
        delay.setHorizontalAlignment(SwingConstants.LEFT);
        delay.setText("Delay (in secs): ");
        perHit.setHorizontalAlignment(SwingConstants.LEFT);
        perHit.setText("Per hits: ");
        recommendations.setHorizontalAlignment(SwingConstants.LEFT);
        recommendations.setText("Recommended delay per hit is
inserted");
        okButton.setText("OK");
        cancelButton.setText("Cancel");

        contentPane.setLayout(null);
        contentPane.setBorder(BorderFactory.createEtchedBorder());
        addComponent(contentPane, recommendations, 10, 10, 230, 18);
        addComponent(contentPane, delay, 10,40,200,18);
        addComponent(contentPane, perHit, 10,72,97,18);
        addComponent(contentPane, delayField, 135,38,45,22);
        addComponent(contentPane, perHitField, 135,70,45,22);
        addComponent(contentPane, okButton, 70,100,83,28);
        addComponent(contentPane, cancelButton, 170,100,83,28);

        KeyStroke enterKey =
KeyStroke.getKeyStroke(KeyEvent.VK_ENTER,0);
        okButton.registerKeyboardAction(theButtonListener, "OK",
enterKey, JComponent.WHEN_IN_FOCUSED_WINDOW);

        this.setTitle("Insert delay between requests");
        this.setLocation(new Point(500, 182));
        this.setSize(new Dimension(320, 170));

this.setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);
//      this.setUndecorated(true);

        this.setResizable(false);
        delayField.requestFocus();
        this.setVisible(true);
    }

    private void addComponent(Container container,Component c,int
x,int y,int width,int height)
    {
        c.setBounds(x,y,width,height);
        container.add(c);
    }


      private class OKButtonListener implements ActionListener {
                public void actionPerformed(ActionEvent event) {
                    int[] delayData = {10, 1};
```

```
                    delayms = delayField.getText();
                    perHitNo = perHitField.getText();



if((!isInteger(delayms)&&(!delayms.matches("\\s*")))||!isInteger(perH
itNo)&&(!perHitNo.matches("\\s*"))){

                        JLabel errorFields = new JLabel("Only numbers
are permitted in the above fields!");
                        JOptionPane.showMessageDialog(null,
errorFields, "Warning",JOptionPane.WARNING_MESSAGE);

                        delayField.setText("");
                        delayField.requestFocus();
                        perHitField.setText("");
                        setVisible(true);

                    }

                    else{
                        if(delayms.matches("\\s*") ||
perHitNo.matches("\\s*")){
                            JLabel errorFields = new JLabel("You must
fill both fields to continue");
                            JOptionPane.showMessageDialog(null,
errorFields, "Warning",JOptionPane.WARNING_MESSAGE);
                        }
                        else{

                            delayms = delayms + "000";

                            delayData[0] = Integer.parseInt(delayms);
                            delayData[1] =
Integer.parseInt(perHitNo);

                            System.out.println("Delay:
"+delayData[0]);
                            System.out.println("Per Hit:
"+delayData[1]);
                            myCore.setDelay(delayData);
                            setVisible(false);

                            Thread thread = new Thread(myThread);
                            myCore.newThread=thread;
                            thread.start();

                    }
                }
            }
        }


      private class cancelButtonListener implements ActionListener{
          public void actionPerformed(ActionEvent event) {

              myMainWindow.exeButton.setEnabled(true);
              setVisible(false);

          }
```

```
        }

        public boolean isInteger(String input){
            try{
                Integer.parseInt( input );
                return true;
            }
            catch(Exception e)
            {
                return false;
           }
        }
}
```

Κλάση LoginWindow:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class LoginWindow extends JFrame{
    // Variables declaration
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JTextField jTextField1;
    private JPasswordField jPasswordField1;
    private JButton loginButton;
    private JPanel contentPane;
    ImageIcon icon = new ImageIcon("success.png");

        //constructor
    public LoginWindow(){
        super();
        create();
        this.setVisible(true);
    }

    //creation of login menu
    private void create(){
        // Variables initialization
        jLabel1 = new JLabel();
        jLabel2 = new JLabel();
        jTextField1 = new JTextField();
        jPasswordField1 = new JPasswordField();
        loginButton = new JButton();
        contentPane = (JPanel)this.getContentPane();
        jLabel1.setHorizontalAlignment(SwingConstants.LEFT);
        jLabel1.setText("username:");
        jLabel2.setHorizontalAlignment(SwingConstants.LEFT);
        jLabel2.setText("password:");
        loginButton.setText("Login");

            //Jbutton Listener
        loginButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
```

```
                loginButton_actionPerformed(e);
            }
        });

        //adding components
        contentPane.setLayout(null);
        contentPane.setBorder(BorderFactory.createEtchedBorder());
        addComponent(contentPane, jLabel1, 5,10,106,18);
        addComponent(contentPane, jLabel2, 5,47,97,18);
        addComponent(contentPane, jTextField1, 110,10,183,22);
        addComponent(contentPane, jPasswordField1, 110,45,183,22);
        addComponent(contentPane, loginButton, 150,75,83,28);

        //Enter Key
        KeyStroke enterKey =
KeyStroke.getKeyStroke(KeyEvent.VK_ENTER,0);
        loginButton.registerKeyboardAction((new ActionListener() {
            public void actionPerformed(ActionEvent e){
                loginButton_actionPerformed(e);
            }
        }), "Login", enterKey, JComponent.WHEN_IN_FOCUSED_WINDOW);


        this.setTitle("Login To MySQL");
        this.setLocation(new Point(500, 182));
        this.setSize(new Dimension(335, 141));
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        this.setResizable(false);
    }

    /** Add Component Without a Layout Manager (Absolute Positioning)
*/
    private void addComponent(Container container,Component c,int
x,int y,int width,int height)
    {
        c.setBounds(x,y,width,height);
        container.add(c);
    }

    //Button Action Listener
    private void loginButton_actionPerformed(ActionEvent e){

        String username = new String(jTextField1.getText());
        String password = new String(jPasswordField1.getPassword());
        if(new ScholarDB().checkLogin(username, password)){
            JLabel errorFields = new JLabel("Successfully connected
in MySQL.");

JOptionPane.showMessageDialog(null,errorFields,"Connecting in MySQL
server", JOptionPane.INFORMATION_MESSAGE, icon);

            new Core().writeLoginData(username, password);
            new ScholarDB().createDB(username, password);
            createMainFrame();
            this.setVisible(false);
        }
        else{
            JLabel errorFields = new JLabel("Enter the correct
username and password");
            JOptionPane.showMessageDialog(null, errorFields, "Error
connecting in MySQL", JOptionPane.ERROR_MESSAGE);
```

```
            jTextField1.setText("");
            jTextField1.requestFocus();
            jPasswordField1.setText("");
               this.setVisible(true);
            }
        }


        public static void main(String[] args){
            new LoginWindow();
        }


        public static void createMainFrame(){
            MainWindow theMainWindow = new MainWindow();
            theMainWindow.setTitle("Scholar Crawler");
            theMainWindow.setSize(800, 600);
            theMainWindow.setExtendedState(Frame.MAXIMIZED_BOTH);
            theMainWindow.setVisible(true);

theMainWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```

Κλάση MainPapersBetweenYears:

```
import javax.swing.*;
import java.awt.Dimension;
import java.awt.event.*;
import java.sql.*;
import java.util.*;


public class MainPapersBetweenYears extends JPanel {
    public Connection con = null;
    public Statement query = null;
    public ResultSet rs = null;
    public String from;
    public String to;

    public MainPapersBetweenYears(String From, String To){
        from = From;
        to = To;
        try{
            con = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
            query = con.createStatement();
            rs = query.executeQuery("SELECT papers.id, papers.title,
papers.authors, papers.publication, papers.yearOfPublication ,
papers.citations from papers, citations " +
                        "where papers.id=citations.paperID and
(citations.yearOfPublication >= "+from+" and
citations.yearOfPublication <= "+to+" ) " +
                        "group by papers.id");


            ResultSetMetaData md = rs.getMetaData();
            int columnCount = md.getColumnCount();
```

```
            Vector columns = new Vector(columnCount);

            Vector<RowsOfMainPaperTable> rowMainPaper= new
Vector<RowsOfMainPaperTable>();
            //store row data

            while(rs.next()){
                RowsOfMainPaperTable rowTable = new
RowsOfMainPaperTable();

                rowTable.title=rs.getString(2);
                rowTable.authors=rs.getString(3);
                rowTable.publication=rs.getString(4);
                rowTable.yearOfPublication=rs.getString(5);
                rowTable.citations=new
ScholarDB().numOfCitBetween(rs.getString(1), from, to);
                rowTable.withoutSelfCitations=new
ScholarDB().removeSelfCitationsBetween(rs.getString(1),from,to);

                rowMainPaper.add(rowTable);
                }

            final JTable table = new JTable(new
MainPaperTable(rowMainPaper));
            table.setPreferredScrollableViewportSize(new
Dimension(1280, 768));
            table.setFillsViewportHeight(true);
            table.setVisible(true);
            table.validate();

table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);

table.getColumnModel().getColumn(0).setPreferredWidth(380);

table.getColumnModel().getColumn(1).setPreferredWidth(350);

table.getColumnModel().getColumn(2).setPreferredWidth(370);

table.getColumnModel().getColumn(3).setPreferredWidth(120);

table.getColumnModel().getColumn(4).setPreferredWidth(80);

table.getColumnModel().getColumn(5).setPreferredWidth(130);

            table.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    if (e.getClickCount() == 2) {
                        int selectedRow = table.getSelectedRow()+1;
                        if(selectedRow>0){
                            String mainPaper =
(String)table.getValueAt(selectedRow-1, 0);
                            new CitationsWindow(mainPaper,
selectedRow, 2, from, to);
                        }
                    }
                    }
                });

            //Create the scroll pane and add the table to it.
```

```
            JScrollPane scrollPane = new JScrollPane(table);
            add(scrollPane);

        }
        catch(SQLException sqle){
            System.out.println(sqle);
            sqle.printStackTrace();
        }
    }
}
```

Κλάση MainPapersFetch:

```
import javax.swing.*;
import java.awt.Dimension;
import java.awt.event.*;
import java.sql.*;
import java.util.*;


public class MainPapersFetch extends JPanel {
    public Connection con = null;
    public Statement query = null;
    public ResultSet rs = null;


    public MainPapersFetch(){
        try{
            con = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
            query = con.createStatement();
            rs = query.executeQuery("SELECT * FROM
scholardb.papers");


            ResultSetMetaData md = rs.getMetaData();
            int columnCount = md.getColumnCount();

            Vector columns = new Vector(columnCount);

            Vector<RowsOfMainPaperTable> rowMainPaper= new
Vector<RowsOfMainPaperTable>();

            while(rs.next()){
                RowsOfMainPaperTable rowTable = new
RowsOfMainPaperTable();

                rowTable.title=rs.getString(2);
                rowTable.authors=rs.getString(3);
                rowTable.publication=rs.getString(4);
                rowTable.yearOfPublication=rs.getString(5);
                rowTable.citations=rs.getString(6);
                rowTable.withoutSelfCitations=new
ScholarDB().removeSelfCitations(rs.getString(1));

                rowMainPaper.add(rowTable);

//              System.out.println("From JTable:
"+rowTable.withoutSelfCitations);
```

```
            }

            final JTable table = new JTable(new
MainPaperTable(rowMainPaper));
            table.setPreferredScrollableViewportSize(new
Dimension(1280, 700));
            table.setFillsViewportHeight(true);
            table.setVisible(true);
            table.validate();

table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);

table.getColumnModel().getColumn(0).setPreferredWidth(380);

table.getColumnModel().getColumn(1).setPreferredWidth(350);

table.getColumnModel().getColumn(2).setPreferredWidth(370);

table.getColumnModel().getColumn(3).setPreferredWidth(120);

table.getColumnModel().getColumn(4).setPreferredWidth(80);

table.getColumnModel().getColumn(5).setPreferredWidth(130);

            table.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    if (e.getClickCount() == 2) {
                        int selectedRow = table.getSelectedRow()+1;
                        if(selectedRow>0){
                            String mainPaper =
(String)table.getValueAt(selectedRow-1, 0);
                            new CitationsWindow(mainPaper,
selectedRow, 1, null, null);
                        }
                    }
                    }
                });

            //Create the scroll pane and add the table to it.
            JScrollPane scrollPane = new JScrollPane(table);
            add(scrollPane);

        }
        catch(SQLException sqle){
            System.out.println(sqle);
            sqle.printStackTrace();
        }
    }
}
```

Κλάση MainPapersTable:

```
import java.util.Vector;
import javax.swing.table.AbstractTableModel;


public class MainPapersTable extends AbstractTableModel{
    Vector<RowsOfMainPaperTable> v;
```

```
    public MainPaperTable(Vector<RowsOfMainPaperTable> p) {
        v = p;
    }


    public int getColumnCount() {
        return 6;
    }


    public Object getValueAt(int row, int column){
        if (column==0)
            return  (v.get(row)).title;
        else if (column==1)
            return(v.get(row)).authors;
        else if (column==2)
            return(v.get(row)).publication;
        else if (column==3)
            return(v.get(row)).yearOfPublication;
        else if (column==4)
            return(v.get(row)).citations;
        else if (column==5)
            return(v.get(row)).withoutSelfCitations;
        return null;
    }


    public int getRowCount(){
        return v.size();
    }


    public String  getColumnName(int columnIndex) {
        if (columnIndex==0)
            return "Title";
        else if (columnIndex==1)
            return "Authors";
        else if (columnIndex==2)
            return "Publicaiton";
        else if (columnIndex==3)
            return "Year of Publication";
        else if (columnIndex==4)
            return "Citations";
        else if (columnIndex==5)
            return "Without SelfCitations";
        return null;
    }
}
```

Κλάση MainWindow:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class MainWindow extends JFrame {
    private Container guiContainer;
```

```
    private JLabel authorLabel = new JLabel("Author (eg: G
Papadakis)");
    private JTextField  authorField = new JTextField(15);
    public  JButton exeButton = new JButton("Search");
    private JLabel messages = new JLabel();
    private JLabel numOfResults = new JLabel();
    private JLabel sumOfCitations = new JLabel();
    private JLabel sumWithoutSelfCiations = new JLabel();
    private JLabel sumOfCitationsBetween = new JLabel();
    private JLabel sumWithoutSelfCiationsBetween = new JLabel();
    public JLabel queryStopped = new JLabel();
    private JScrollPane scrollPane;

    public MainWindow() {

        guiContainer = getContentPane();
        guiContainer.setLayout(new FlowLayout());
        ButtonListener theButtonListener = new ButtonListener();
        exeButton.addActionListener(theButtonListener);
        guiContainer.add(authorLabel);
        guiContainer.add(authorField);
        guiContainer.add(exeButton);
        guiContainer.add(messages);
        guiContainer.add(numOfResults);
        guiContainer.add(sumOfCitations);
        guiContainer.add(sumWithoutSelfCiations);
        guiContainer.add(sumOfCitationsBetween);
        guiContainer.add(sumWithoutSelfCiationsBetween);
        guiContainer.add(queryStopped);

        KeyStroke enterKey =
KeyStroke.getKeyStroke(KeyEvent.VK_ENTER,0);
        exeButton.registerKeyboardAction(theButtonListener, "Execute
Query", enterKey, JComponent.WHEN_IN_FOCUSED_WINDOW);

        /**********************DataBase
Menu************************/
        DBMenuListener DBListener = new DBMenuListener();

        JMenu dbMenu = new JMenu("Database");

        JMenuItem dbMenuItem = new JMenuItem("Delete database
contents");
        dbMenu.add(dbMenuItem);
        dbMenuItem.addActionListener(DBListener);

        JMenuBar menuBar = new JMenuBar();
        menuBar.add(dbMenu);

        /**********************Results Menu**********************/

        ResultsMenuListener ResultsListener = new
ResultsMenuListener(this);

        JMenu resultsMenu = new JMenu("Results");
        JMenuItem ResultsMenuItem = new JMenuItem("Show All
Results");
        resultsMenu.add(ResultsMenuItem);
        ResultsMenuItem.addActionListener(ResultsListener);
```

```
        ResultsMenuItem = new JMenuItem("Show citations dated
between..");
        resultsMenu.add(ResultsMenuItem);
        ResultsMenuItem.addActionListener(ResultsListener);

        menuBar.add(resultsMenu);
        setJMenuBar(menuBar);

        /***************************Help
Menu**************************/

        HelpListener theHelpListener = new  HelpListener();

        JMenu helpMenu = new JMenu("Help");

        JMenuItem menuItem2 = new JMenuItem("Help");
        helpMenu.add(menuItem2);
        menuItem2.addActionListener(theHelpListener);

        menuItem2 = new JMenuItem("About the program...");
        helpMenu.add(menuItem2);
        menuItem2.addActionListener(theHelpListener);

        menuBar.add(helpMenu);
        scrollPane = new JScrollPane(new MainPapersFetch());
        getContentPane().add(scrollPane).setVisible(false);
  }

      /***********************************L I S T E N E R
S*********************************/

      private class DBMenuListener implements ActionListener {
            public void actionPerformed(ActionEvent event) {
                String command = event.getActionCommand();
                if (command.equals("Delete database contents")){
                    new ScholarDB().truncate();
                    messages.setText("Database content deleted");
                    messages.setVisible(true);
                    numOfResults.setVisible(false);
                    sumOfCitations.setVisible(false);
                    sumWithoutSelfCiations.setVisible(false);
                    sumOfCitationsBetween.setVisible(false);
                    sumWithoutSelfCiationsBetween.setVisible(false);
                    queryStopped.setVisible(false);
                    rePaint(1, null, null);
                }
            }
      }

      private class ResultsMenuListener implements ActionListener {
          MainWindow frame;

          public ResultsMenuListener(MainWindow f){
              frame =f;
          }
            public void actionPerformed(ActionEvent event) {
                queryStopped.setVisible(false);
              String command = event.getActionCommand();
              if (command.equals("Show All Results")) {
                  numOfResults.setText("Number of Results: "+new
ScholarDB().numOfResults()+"  - ");
```

```
                    sumOfCitations.setText("Number of Citations: "+new
ScholarDB().totalNumOfCitations()+"  - ");
                    sumWithoutSelfCiations.setText("Without Self
Citations: "+new ScholarDB().totalNumWithoutSelfCitations());
                    numOfResults.setVisible(true);
                    sumOfCitations.setVisible(true);
                    sumWithoutSelfCiations.setVisible(true);
                    messages.setVisible(false);
                    sumOfCitationsBetween.setVisible(false);
                    sumWithoutSelfCiationsBetween.setVisible(false);
                    rePaint(1, null, null);
                    }
                else if (command.equals("Show citations dated
between..")){
                    messages.setVisible(false);
                    sumOfCitations.setVisible(false);
                    sumWithoutSelfCiations.setVisible(false);
                    numOfResults.setVisible(false);
                    YearsRangeWindow yearsRange =  new
YearsRangeWindow(frame);

                    }
                }
            }


      private class HelpListener implements ActionListener {
          public void actionPerformed(ActionEvent event) {
              String command = event.getActionCommand();
              if (command.equals("Help")) {
                  JOptionPane.showMessageDialog(null,"Double Click to
see Citations of the specific paper");
                  }else{
                      JOptionPane.showMessageDialog(null,"This
program is part of my diploma thesis. -George Palolli.");
                  }
              }
          }



      private class ButtonListener implements ActionListener {
            public void actionPerformed(ActionEvent event) {

                if(authorField.getText().equalsIgnoreCase("")){

                    JLabel errorFields = new JLabel("Please enter the
authors name you want to search");
                    JOptionPane.showMessageDialog(null, errorFields,
"Warning",JOptionPane.WARNING_MESSAGE);
                    authorField.requestFocus();

                }
                else{
                    executeQuery();
                }
            }
      }

      public void executeQuery(){
```

```
            exeButton.setEnabled(false);
            messages.setVisible(false);
            new ScholarDB().truncate();
            String name = authorField.getText();

            new Core().makeRequest(name, this);

        }

        public void showResults(){

            numOfResults.setText("Number of Results: "+new
ScholarDB().numOfResults()+"  - ");
            sumOfCitations.setText("Number of Citations: "+new
ScholarDB().totalNumOfCitations()+"  - ");
            sumWithoutSelfCiations.setText("Without Self Citations :
"+new ScholarDB().totalNumWithoutSelfCitations());
            numOfResults.setVisible(true);
            sumOfCitations.setVisible(true);
            sumWithoutSelfCiations.setVisible(true);
            sumOfCitationsBetween.setVisible(false);
            sumWithoutSelfCiationsBetween.setVisible(false);


            rePaint(1, null, null);
            exeButton.setEnabled(true);
        }

        //repaint main papers jtable
        public void rePaint(int flag, String from, String to){

            guiContainer.remove(scrollPane);
            if(flag==1){//flag=1 periptwsh OLWN twn main papers

                scrollPane = new JScrollPane(new MainPapersFetch());
            }
            else if(flag==2){ //flag=1 periptwsh twn main papers
BETWEEN

                scrollPane = new JScrollPane(new
MainPapersBetweenYears(from, to));
            }
            guiContainer.add(scrollPane).setVisible(true);
            requestFocus(false);
            setVisible(true);
            requestFocus(false);
          }

        public void putResultMessages(String FROM, String TO){
            numOfResults.setText("Number of Results: "+new
ScholarDB().totalNumOfResultsBetween(FROM, TO)+"  - ");
            sumOfCitationsBetween.setText("Number of Citations: "+new
ScholarDB().totalNumOfCitationsBetween(FROM, TO)+"  - ");
            sumWithoutSelfCiationsBetween.setText("Without Self
Citations: "+new
ScholarDB().totalNumWithoutSelfCitationsBetween(FROM, TO));
            sumOfCitationsBetween.setVisible(true);
            sumWithoutSelfCiationsBetween.setVisible(true);
            numOfResults.setVisible(true);
        }
}
```

Κλάση MyThread:

```
public class MyThread extends Thread{
Core myCore;
    public MyThread(Core core){

        myCore = core;
    }

    public void run(){
        myCore.search();

    }
}
```

Κλάση ProgressBars

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ProgressBars{

    private JLabel label;
    private JLabel papersLabel;
    private JLabel citationsLabel;
    public JProgressBar pb1;
    public JProgressBar pb2;
    private JButton button;
    public int paper1Counter;
    public int paper2Counter;
    public JFrame frame;
    private MainWindow mWindow;
    private Thread mThread;

    public ProgressBars(MainWindow window, Thread thread) {

        mThread=thread;
        mWindow=window;

        frame = new JFrame("Progress Bar");
        button = new JButton("Stop");
        button.addActionListener(new stopButtonListener());

        pb1 = new JProgressBar(0, 20);
        pb1.setValue(0);
        pb1.setStringPainted(true);

        pb2 = new JProgressBar(0, 20);
        pb2.setValue(0);
        pb2.setStringPainted(true);

        label = new JLabel("Press Stop to quit the search");
        papersLabel = new JLabel("                          Papers
progress");
```

```
        citationsLabel = new JLabel("                Citations
progress");


        JPanel panel = new JPanel();
        panel.add(button);
        panel.add(pb1);
        panel.add(pb2);

        JPanel panel2 = new JPanel();
        panel2.add(papersLabel);
        panel2.add(citationsLabel);

        JPanel panel1 = new JPanel();
        panel1.setLayout(new BorderLayout());
        panel1.add(panel2, BorderLayout.NORTH);
        panel1.add(panel, BorderLayout.CENTER);
        panel1.add(label, BorderLayout.SOUTH);
        panel1.setBorder(BorderFactory.createEmptyBorder(20, 20, 20,
20));
        frame.setContentPane(panel1);
        frame.pack();
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        frame.setResizable(false);
        frame.setLocation(new Point(500, 182));
    }

    public void p1Increment(){

        paper1Counter++;
        pb1.setValue(paper1Counter);
    }


    public void p2Increment(){

        paper2Counter++;
        pb2.setValue(paper2Counter);
    }


      private class stopButtonListener implements ActionListener{

          public void actionPerformed(ActionEvent event){

                mWindow.exeButton.setEnabled(true);
                mWindow.queryStopped.setText("Searching Stopped");
                mWindow.queryStopped.setVisible(true);
                mThread.stop();
              frame.setVisible(false);
              mWindow.rePaint(1, null, null);

          }
      }
  }
```

Κλάση ScholarDB:

```
import java.io.*;
import java.sql.*;

public class ScholarDB {

    public void createDB(String username, String password){

        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),1);

        try{
            FileInputStream fstream = new
FileInputStream("db_script.sql");
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new
InputStreamReader(in));
            String strLine;
            String script="";

            while ((strLine = br.readLine()) != null)   {
                script = script+strLine+" ";
            }

            in.close();

            String[] temp;
            String delimiter = ";";
            temp = script.split(delimiter);

            try {
                Statement stmt = conn.createStatement();
                for(int i=0; i < temp.length ; i++){
                    String query = temp[i]+";";
                    if(query.equalsIgnoreCase(" ;")) break;
                    //System.out.println(query);
                    stmt.executeUpdate(query);
                }

                stmt.close();
                conn.close();

               } catch (SQLException e) {
                e.printStackTrace();
               }

            }catch (Exception e){
                System.err.println("Error: " + e.getMessage());
        }
    }


    public boolean checkLogin(String user, String pass){
       boolean flag = true;
       final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
       final String DB_URL = "jdbc:mysql://localhost";
       Connection conn = null;

           try{
               Class.forName(JDBC_DRIVER);

               conn = DriverManager.getConnection(DB_URL,user,pass);
```

```
            }catch (ClassNotFoundException e) {
                String msg = "The com.mysql.jdbc.Driver is missing\n"
                        + "install and rerun the application";
                System.out.println(msg);
                System.exit(1);
            }catch (SQLException e) {
                String msg = "Check Login - Error Connecting to
Database:\n"
                        + e.getMessage();
                flag = false;
                System.out.println(msg);
            }catch(Exception e){
               //Handle errors for Class.forName
               e.printStackTrace();
            }//end try

            return flag;

    }



   public Connection openConnection(String user, String pass, int
flag){

        // JDBC driver name and database URL
        String JDBC_DRIVER = "com.mysql.jdbc.Driver";
        String DB_URL;

        if (flag==1){
            DB_URL = "jdbc:mysql://localhost";
        }
        else{
            DB_URL = "jdbc:mysql://localhost/scholardb";
        }

        Connection conn = null;

        try{
           //STEP 2: Register JDBC driver
           Class.forName(JDBC_DRIVER);


           //STEP 3: Open a connection
           //System.out.println("Connecting to database...");
           conn = DriverManager.getConnection(DB_URL,user,pass);

        }catch (ClassNotFoundException e) {
            String msg = "The com.mysql.jdbc.Driver is missing\n"
                    + "install and rerun the application";
            System.out.println(msg);
            System.exit(1);
        }catch (SQLException e) {
            String msg = "Open Connection - Error Connecting to
Database:\n"
                    + e.getMessage();
            System.out.println(msg);
        }catch(Exception e){
           //Handle errors for Class.forName
           e.printStackTrace();
```

```
        }//end try

        return conn;
    }



    public void insertP1(String title, String authors, String
publication, int citations, int year){

        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);

        try {
            Statement stmt = conn.createStatement();
            String query = "INSERT INTO papers (title, authors,
publication, yearOfPublication, citations) VALUES" + "
('"+title+"','"+authors+"','"+publication+"','"+year+"','"+citations+
"')";
            stmt.executeUpdate(query);

            stmt.close();
            conn.close();

        } catch (SQLException e) {
         e.printStackTrace();
        }
    }


    public String removeSelfCitations(String p1id){
        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
        String numOfResults = "";

        try {
            Statement stmt = conn.createStatement();

            String query = "SELECT count(*) " +
                 "from papers, citations as pp2 " +
                 "where papers.id=pp2.paperID  and papers.id="+p1id+"
and pp2.citationID not in " +
                    "(select p2.citationID " +
                    "from citations as p2, citationsauthors " +
                    "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                    "(select name " +
                    "from authors " +
                    "where authors.paperID = papers.id) " +
                    "group by p2.citationID " +
                    "having count(*) > 0 )";
            ResultSet rs = stmt.executeQuery(query);
            while(rs.next()){

                  numOfResults=rs.getString("count(*)");
            }
            rs.close();
            stmt.close();
            conn.close();
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
       return numOfResults;
   }


   public String removeSelfCitationsBetween(String p1id, String from,
String to){
       Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
       String numOfResults = "";

       try {
           Statement stmt = conn.createStatement();

           String query = "SELECT count(*) " +
                "from papers, citations as pp2 " +
                "where papers.id=pp2.paperID and papers.id="+p1id+"
and pp2.yearOfPublication>= "+from+" and pp2.yearOfPublication<=
"+to+" and pp2.citationID not in " +
                    "(select p2.citationID " +
                    "from citations as p2, citationsauthors " +
                    "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                        "(select name " +
                        "from authors " +
                        "where authors.paperID = papers.id) " +
                        "group by p2.citationID " +
                        "having count(*) > 0 )";

           ResultSet rs = stmt.executeQuery(query);
           while(rs.next()){

                 numOfResults=rs.getString("count(*)");
           }
           rs.close();
           stmt.close();
           conn.close();

        } catch (SQLException e) {
            e.printStackTrace();
        }
       return numOfResults;
   }


   public int totalNumOfCitationsBetween(String from, String to){

       Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
       int numOfCitations=0;
        try {
           Statement stmt = conn.createStatement();

           String query = "SELECT count(*) " +
                   "from papers, citations " +
                   "where papers.id=citations.paperID and
(citations.yearOfPublication >= "+from+" and
citations.yearOfPublication <= "+to+" )";
```

```
            ResultSet rs = stmt.executeQuery(query);
             while(rs.next()){

                 numOfCitations =
numOfCitations+rs.getInt("count(*)");
             }

                rs.close();
                stmt.close();
                conn.close();

            } catch (SQLException e) {
                e.printStackTrace();
            }

        return numOfCitations;
    }

    public int totalNumOfResultsBetween(String from, String to){
        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
        int numOfCitations=0;

         try {
            Statement stmt = conn.createStatement();

            String query = "SELECT count(*)" +
                    "from papers as pp1 " +
                    "where pp1.title in " +
                    "(SELECT papers.title " +
                        "from citations, papers " +
                        "where papers.id = citations.paperID and
(citations.yearOfPublication >= "+from+" and
citations.yearOfPublication <=  "+to+" ))";

            ResultSet rs = stmt.executeQuery(query);

            while(rs.next()){

                 numOfCitations =
numOfCitations+rs.getInt("count(*)");
             }

                rs.close();
                stmt.close();
                conn.close();

            } catch (SQLException e) {
                e.printStackTrace();
            }


        return numOfCitations;
    }

    public String numOfCitBetween(String p1id, String from, String
to){

        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
        int numOfCitations=0;
```

```
        try {
            Statement stmt = conn.createStatement();

            String findP1id = "SELECT count(*) " +
                    "from papers, citations " +
                    "where papers.id="+p1id+" and
papers.id=citations.paperID and (citations.yearOfPublication >=
"+from+" and citations.yearOfPublication <= "+to+" )";
            ResultSet rs = stmt.executeQuery(findP1id);
             while(rs.next()){

                 numOfCitations = rs.getInt("count(*)");
             }

                rs.close();
                stmt.close();
                conn.close();

            } catch (SQLException e) {
                e.printStackTrace();
            }

        String StringNumOfCitations = ""+numOfCitations;

        return StringNumOfCitations;
    }


    public int numOfResults(){
        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
        int numOfCitations=0;

        try {
            Statement stmt = conn.createStatement();

            String findP1id = "SELECT count(*) from papers ";

            ResultSet rs = stmt.executeQuery(findP1id);
             while(rs.next()){

                 numOfCitations = rs.getInt("count(*)");
             }
                rs.close();
                stmt.close();
                conn.close();

            } catch (SQLException e) {
                e.printStackTrace();
            }

        return numOfCitations;
    }


    public int totalNumOfCitations(){
        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
        int numOfCitations=0;
         try {
```

```
            Statement stmt = conn.createStatement();

            String findP1id = "SELECT citations FROM
scholardb.papers";
            ResultSet rs = stmt.executeQuery(findP1id);
             while(rs.next()){

                 numOfCitations =
numOfCitations+rs.getInt("citations");
             }

//            System.out.println("Total num of Citations:
"+numOfCitations);
                rs.close();
                stmt.close();
                conn.close();

            } catch (SQLException e) {
                e.printStackTrace();
            }

        return numOfCitations;
   }


   public int totalNumWithoutSelfCitations(){

       Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
       int numOfResults = 0;

       try {
           Statement stmt = conn.createStatement();

           String query = "SELECT count(*) " +
                "from papers, citations as pp2 " +
                "where papers.id=pp2.paperID and pp2.citationID not
in " +
                    "(select p2.citationID " +
                    "from citations as p2, citationsauthors " +
                    "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                        "(select name " +
                        "from authors " +
                        "where authors.paperID = papers.id) " +
                        "group by p2.citationID " +
                        "having count(*) > 0 )";

           ResultSet rs = stmt.executeQuery(query);
           while(rs.next()){

                numOfResults=rs.getInt("count(*)");
           }

           rs.close();
           stmt.close();
           conn.close();

        } catch (SQLException e) {
            e.printStackTrace();
```

```
        }
        return numOfResults;
    }



    public int totalNumWithoutSelfCitationsBetween(String from, String
to){

        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
        int numOfResults = 0;

        try {
            Statement stmt = conn.createStatement();

            String query = "SELECT count(*) " +
                    "from papers, citations as pp2 " +
                    "where papers.id=pp2.paperID and
pp2.yearOfPublication>="+from+" and pp2.yearOfPublication<="+to+" and
pp2.citationID not in " +
                        "(select p2.citationID " +
                        "from citations as p2, citationsauthors " +
                        "where p2.citationID = pp2.citationID and
p2.citationID=citationsauthors.citationID and citationsauthors.name
in " +
                                "(select name " +
                                "from authors " +
                                "where authors.paperID = papers.id) " +
                                "group by p2.citationID " +
                                "having count(*) > 0 )";

            ResultSet rs = stmt.executeQuery(query);
            while(rs.next()){

                  numOfResults=rs.getInt("count(*)");
            }

            rs.close();
            stmt.close();
            conn.close();

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return numOfResults;
    }



    public void insertP2(String P1title ,String P2title, String
authors, int year){

        Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
        int P1id=0;
         try {
            Statement stmt = conn.createStatement();

            String findP1id = "SELECT id FROM scholardb.papers where
title = "+"'"+P1title+"'";
```

```
            ResultSet rs = stmt.executeQuery(findP1id);
             while(rs.next()){

                 P1id  = rs.getInt("id");
             }

            String query = "INSERT INTO scholardb.citations (title,
authors, yearOfPublication, paperID) VALUES" + "
('"+P2title+"','"+authors+"','"+year+"','"+P1id+"')";
            stmt.executeUpdate(query);

            rs.close();
            stmt.close();
            conn.close();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }


   public void insertAuthors(String Ptitle, String paper, String
author){

       Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);
       String PXid;
       int Pid=0;
       if(paper.equalsIgnoreCase("papers")){
           PXid= "paperID";

            try {
                Statement stmt = conn.createStatement();

                String findPid = "SELECT id FROM "+paper+" where
title = "+"'"+Ptitle+"'";
                ResultSet rs = stmt.executeQuery(findPid);
                 while(rs.next()){

                     Pid  = rs.getInt("id");
                 }

                String query = "INSERT INTO authors (name, "+PXid+")
VALUES" + " ('"+author+"', '"+Pid+"')";
                stmt.executeUpdate(query);

                rs.close();
                stmt.close();
                conn.close();

            } catch (SQLException e) {
                e.printStackTrace();
            }


       }else{
           PXid = "citationID";

        try {
            Statement stmt = conn.createStatement();
```

```
            String findPid = "SELECT citationID FROM "+paper+" where
title = "+"'"+Ptitle+"'";
            ResultSet rs = stmt.executeQuery(findPid);
             while(rs.next()){

                 Pid  = rs.getInt("citationID");
             }

            String query = "INSERT INTO citationsauthors (name,
"+PXid+") VALUES" + "('"+author+"', '"+Pid+"')";
            stmt.executeUpdate(query);

            rs.close();
            stmt.close();
            conn.close();

        } catch (SQLException e) {
            e.printStackTrace();
        }
       }
   }



   public void truncate(){
       Connection conn = new ScholarDB().openConnection(new
Core().readLoginData("user"),new Core().readLoginData("pass"),2);

       try {
           Statement stmt = conn.createStatement();
           String checks0 = "SET FOREIGN_KEY_CHECKS=0;";
           stmt.executeQuery(checks0);

           String truncatePaper1 = "TRUNCATE scholardb.papers;";
           stmt.executeQuery(truncatePaper1);

           String truncatePaper1Authors = "TRUNCATE
scholardb.authors;";
           stmt.executeQuery(truncatePaper1Authors);

           String truncatePaper2 = "TRUNCATE scholardb.citations;";
           stmt.executeQuery(truncatePaper2);

           String truncatePaper2Authors = "TRUNCATE
scholardb.citationsauthors;";
           stmt.executeQuery(truncatePaper2Authors);

           String checks1 = "SET FOREIGN_KEY_CHECKS=1;";
           stmt.executeQuery(checks1);


           stmt.close();
           conn.close();

    } catch (SQLException e) {

        e.printStackTrace();
    }
   }
}
```

Κλάση YearsRangeWindow:

```
import java.awt.*;
import java.awt.event.*;
import java.util.Calendar;

import javax.swing.*;


public class YearsRangeWindow extends JFrame{

    private JLabel fromYearLabel;
    private JLabel toYearLabel;
    private JTextField fromYearField;
    private JTextField toYearField;
    private JButton okButton;
    private JPanel contentPane;
    private MainWindow mainWindowFrame;
    private String FROM;
    private String TO;


    public YearsRangeWindow(MainWindow frame){
        mainWindowFrame = frame;
        createWindow();
        this.setVisible(true);
    }

    public void createWindow(){
        fromYearLabel = new JLabel();
        toYearLabel = new JLabel();
        fromYearField = new JTextField();
        toYearField = new JTextField();
        okButton = new JButton();
        OKButtonListener theButtonListener = new OKButtonListener();
        okButton.addActionListener(theButtonListener);

        KeyStroke enterKey =
KeyStroke.getKeyStroke(KeyEvent.VK_ENTER,0);
        okButton.registerKeyboardAction(theButtonListener, "Ok",
enterKey, JComponent.WHEN_IN_FOCUSED_WINDOW);

        contentPane = (JPanel)this.getContentPane();
        fromYearLabel.setHorizontalAlignment(SwingConstants.LEFT);
        fromYearLabel.setText("Show citations dated between: ");
        toYearLabel.setHorizontalAlignment(SwingConstants.LEFT);
        toYearLabel.setText("-");
        okButton.setText("OK");

        contentPane.setLayout(null);
        contentPane.setBorder(BorderFactory.createEtchedBorder());
        addComponent(contentPane, fromYearLabel, 10,12,200,18);
        addComponent(contentPane, toYearLabel, 255,11,97,18);
        addComponent(contentPane, fromYearField, 200,10,45,22);
        addComponent(contentPane, toYearField, 270,10,45,22);
        addComponent(contentPane, okButton, 125,51,83,28);
```

```
        this.setTitle("Insert Years");
        this.setLocation(new Point(500, 182));
        this.setSize(new Dimension(335, 120));
        this.setDefaultCloseOperation(WindowConstants.HIDE_ON_CLOSE);
        this.setResizable(false);
    }

    private void addComponent(Container container,Component c,int
x,int y,int width,int height)
    {
        c.setBounds(x,y,width,height);
        container.add(c);
    }


      private class OKButtonListener implements ActionListener {
                public void actionPerformed(ActionEvent event) {
                    FROM = fromYearField.getText();
                    TO = toYearField.getText();


if((!isInteger(FROM)&&(!FROM.matches("\\s*")))||!isInteger(TO)&&(!TO.
matches("\\s*"))){
                        JLabel errorFields = new JLabel("Only numbers
are permitted in the above fields!");
                        JOptionPane.showMessageDialog(null,
errorFields, "Warning",JOptionPane.WARNING_MESSAGE);

                        fromYearField.setText("");
                        fromYearField.requestFocus();
                        toYearField.setText("");
                        setVisible(true);

                    }else{

                        if(FROM.matches("\\s*") &&
TO.matches("\\s*")){
                            FROM="0";

TO=""+Calendar.getInstance().get(Calendar.YEAR);
                    }
                        else if(FROM.matches("\\s*")){
                            FROM="0";
                    }
                        else if(TO.matches("\\s*")){

TO=""+Calendar.getInstance().get(Calendar.YEAR);
                    }

                    mainWindowFrame.rePaint(2, FROM, TO);
                    mainWindowFrame.putResultMessages(FROM, TO);
                    setVisible(false);
                    }
            }
      }


      public boolean isInteger(String input){
          try{
              Integer.parseInt( input );
              return true;
```

```
            }
            catch(Exception e)
            {
                return false;
            }
        }
}
```