ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΞΑΜΗΝΙΑΙΟ PROJECT ΜΑΘΗΜΑΤΟΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

2023-24

ΜΠΕΡΤΣΕΚΑΣ ΠΑΡΑΣΚΕΥΑΣ-ΣΩΤΗΡΙΟΣ (ΑΜ: 1093445)

up1093445@ac.upatras.gr

ΔΗΜΗΤΡΑΚΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ (ΑΜ: 1079500)

chris.dimitrako@ac.upatras.gr

ΛΑΟΥΡΕΝΤΙΟΥΣ ΙΩΑΝΝΗΣ (ΑΜ: 1093411)

up1093411@ac.upatras.gr

ΜΠΕΡΤΣΕΚΑΣ ΠΑΡΑΣΚΕΥΑΣ-ΣΩΤΗΡΙΟΣ (ΑΜ: 1093445)
ΔΗΜΗΤΡΑΚΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ (ΑΜ: 1079500)
ΛΑΟΥΡΕΝΤΙΟΥΣ ΙΩΑΝΝΗΣ (ΑΜ: 1093411)

ΠΕΡΙΕΧΟΜΕΝΑ

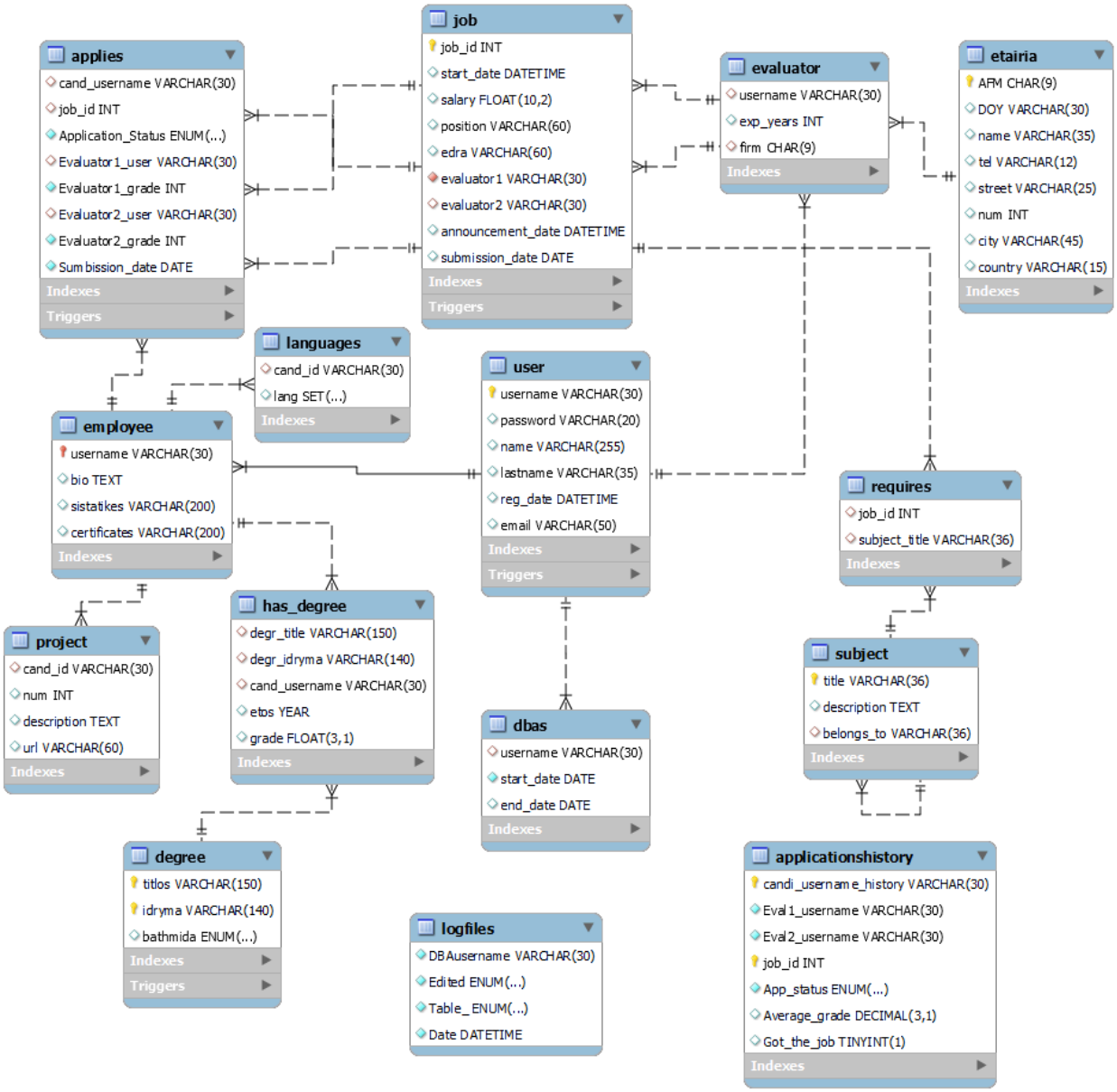
Κεφάλαιο 1: Περιγραφή της Βάσης Δεδομένων....4
ER....4
Περιγραφή tables....5
Κεφάλαιο 2: Περιγραφή SQL....7
Γενικά....7
Σχεδιασμός SQL ανα ερώτημα....7
3.1.2.1:....7
3.1.2.2:....9
3.1.2.3:....10
3.1.2.4:....10
3.1.3.1:....11
3.1.3.2:....11
3.1.3.3:....11
3.1.3.4:....12
Θεωρία....12
Ζητούμενο α)....12
Ζητούμενο β)....12
Χρόνοι....12
3.1.4.1:....13
3.1.4.2:....13
3.1.4.3:....13
Κεφάλαιο 2: Κώδικας ανα ερώτημα και παραδείγματα procedure....14
3.1.2.1....14
ApplicationDeadlineCheck....14
Example....14
CheckIfApplicantHasLessThan3ApplicationsActive....15
Example....15
CheckIfApplicationIsCancellable....16
Example....16
3.1.2.2....16
CalculateAndAssignGrades....16

Example 21
WhatDidHeSaaaay 21
Example 24
3.1.3.1 24
EvaluatorGradeCheck 24
Example 26
3.1.3.2 27
InsertCancelIREactivate_applications 27
Example 30
3.1.3.3 32
CallAllToFinishJob 32
Example 33
3.1.3.4 34
α) procedure2 34
Example 35
β) procedure3 35
Example 36
Κεφάλαιο 3: Κώδικας και παραδείγματα triggers 37
3.1.4.1 37
job_insert_trigger 37
Example 37
job_update_trigger 38
job_delete_trigger 38
user_insertttrigger 38
Example 39
user_updatetrigger 39
user_deletetrigger 39
degree_insert_trigger 40
Example 40
degree_update_trigger 41
degree_delete_trigger 41
3.1.4.2 42
CheckBeforeApply 42
Example 42
3.1.4.3 43
UpdateApplications 43
Example 44

Κεφάλαιο 4: GUI.....44
Γενικά.....44
Screenshot λειτουργίας.....44
Screenshot λειτουργίας trigger.....49
Σχεδιασμός κώδικα.....50
Bonus χαρακτηριστικά.....56
Log In Interface.....56
Κώδικας.....57
Log Out Button.....58
Κώδικας.....60
COMBOBOX με δυνατότητα μετάβασης σε άλλο πίνακα σε κάθε σελίδα.....60
Κώδικας.....60
Click-To-Select σειρές του πίνακα και άμεσο “πέρασμα” τιμών στα textbox.....63
Κώδικας.....64

Κεφάλαιο 1: Περιγραφή της Βάσης Δεδομένων

ER



Περιγραφή tables

Τα tables που χρησιμοποιήθηκαν για την υλοποίηση της Βάσης Δεδομένων είναι τα κάτωθι:

· user: εδώ βρίσκονται τα στοιχεία του χρήστη και έχει γνωρίσματα το username, password, όνομα, επώνυμο, ημερομηνία εγγραφής, το email. Το username είναι κλειδί αυτού του πίνακα και συσχετίζεται με τους επόμενους πίνακες employee, evaluator και DBAs.

· etairia: έχει τα γνωρίσματα AFM, DOY, name, tel, street, num, city και country. Κλειδί του πίνακα etairia είναι το AFM και συσχετίζεται με τον επόμενο πίνακα evaluator.

· evaluator: έχει τα γνωρίσματα username, όπου είναι μοναδικό και είναι ξένο κλειδί από τον πίνακα user, exp_years και firm, όπου και αυτό είναι ξένο κλειδί από τον πίνακα etairia (AFM). Ο πίνακας αυτός συσχετίζεται με τους παρακάτω πίνακες job και applies.

· employee: γνωρίσματά του είναι τα username, όπου είναι κλειδί του πίνακα αυτού αλλά και ξένο κλειδί από τον πίνακα user, το bio, sistatikes και certificates. Ο πίνακας αυτός συσχετίζεται με τους παρακάτω πίνακες languages, project, applies, has_degree, και user.

· DBAs(Data Base Admin): έχει γνωρίσματα τα username, όπου είναι ξένο κλειδί από τον πίνακα user, start_date και end_date. Συσχετίζεται με τον πίνακα user

· languages: έχει τα γνωρίσματα lang και cand_id, που είναι μοναδικό για κάθε employee και είναι ξένο κλειδί από τον πίνακα employee(username). Επίσης, από την προπαρασκευαστική φάση ζητείται να υπάρχουν 3 επιλογές γλώσσας: EN, GE, GR. Οπότε χρησιμοποιούμε SET('EN,GE,GR'). Ο πίνακας συσχετίζεται με τον employee.

· project: Εδώ βρίσκονται τα projects που υλοποιούν οι εργαζόμενοι. Έχει γνωρίσματα τα cand_id, όπου είναι ξένο κλειδί από τον πίνακα employee(username), num (κατ' αύξοντα αριθμό μοναδικό για κάθε project), description και url. Ο πίνακας αυτός συσχετίζεται με τον πίνακα employee.

· job: έχει γνωρίσματα τα job_id, όπου είναι κλειδί του πίνακα και το κάθε id είναι μοναδικό, start_date, salary, position, edra, evaluator1, όπου είναι ξένο κλειδί από τον πίνακα evaluator(username), evaluator2, όπου και αυτό είναι ξένο κλειδί από τον πίνακα evaluator(username), announcement_date και submission_date. Συσχετίζεται με τους πίνακες evaluator, requires και applies.

· *subject:* έχει γνωρίσματα τα title, που είναι κλειδί του πίνακα, description και belongs_to. Συσχετίζεται με τον πίνακα *requires*.

· *requires:* έχει γνωρίσματα τα job_id, όπου είναι ξένο κλειδί από τον πίνακα *job(job_id),* subject_title*,* όπου είναι ξένο κλειδί από τον πίνακα *subject(title).* Συσχετίζεται με τους πίνακες *job και subject.*

· *applies:* Γνωρίσματά του είναι τα cand_username, που είναι ξένο κλειδί από τον πίνακα *employee(username)*, job_id, που είναι ξένο κλειδί από τον πίνακα *job(job_id)* , Application_Status, που δίνει επιλογές ¨active¨, ¨finished¨, ¨cancelled¨, Evaluator1_user, που είναι ξένο κλειδί από τον πίνακα *job(evaluator1)*, Evaluator1_grade, Evaluator2_user, που είναι ξένο κλειδί από τον πίνακα *job(evaluator1)*, Evaluator2_grade και Submission_date. Ο πίνακας αυτός συσχετίζεται με τους *job, employee και job.*

· *degree*: έχει γνωρίσματα τα titlos, idryma και bathmida, όπου τα titlos και idryma είναι κλειδιά του πίνακα. Συσχετίζεται με τον πίνακα *has_degree.*

· *has_degree:* Γνωρίσματά του είναι τα degr_title, που είναι ξένο κλειδί από τον πίνακα *degree(titlos)* είναι degr_idryma, που είναι ξένο κλειδί από τον πίνακα *degree(idryma)*, cand_username, που είναι ξένο κλειδί από τον πίνακα *employee(username),* etos και grade. Συσχετίζεται ο πίνακας αυτός με τους πίνακες *employee και degree.*

· *ApplicationsHistory:* Πρακτικά, αυτός είναι ο πίνακας “ιστορικό” που ζητείται να υλοποιήσουμε, και έχει γνωρίσματα τα candi_username_history, που είναι οι ολοκληρωμένες αιτήσεις κάθε υποψήφιου, Eval1_username, Eval2_username, job_id, App_status, Average_grade, Got_the_job,

· *logFlies:* Εδώ έχουμε έναν πίνακα που καταγράφει τις ενέργειες των user, και έχει γνωρίσματα τα DBAusername (ο αρχικός σκοπός ήταν να αποθηκεύουμε τις κινήσεις που κάνουν οι DBAs [βλ. Ερώτημα 3.1.2.4]), Edited, που δείχνει τις ενέργειες που έχουν γίνει, και τέλος το Table_, που δείχνει απο ποιόν πίνακα έγιναν οι ενέργειες απο το user.

Κεφάλαιο 2: Περιγραφή SQL

Γενικά

Σε γενικές γραμμές η υλοποίηση έγινε ακριβώς όπως περιγράφεται στα ζητούμενα, χωρίς πολλές αλλαγές στο δοθέν ER. Σημειώνεται πως όλα τα ζητούμενα λειτουργούν όπως ζητούνται, με πολύ μικρές διαφοροποιήσεις. Επίσης σε κάποια ερωτήματα ζητείται η επανάληψη μερικών “μεθόδων” και για λόγους εξασφάλισης του αποτελέσματος δεν φτιάξαμε distinct μεθόδους για όλα τα ζητούμενα (αφορά την περίπτωση που για κάποιο λόγο κάποια μέθοδος δεν δουλέψει τη στιγμή που χρειάζεται). Αντίστοιχα, για τους triggers, παρόλο που έχουν δημιουργηθεί για να ελέγχεται η εγκυρότητα των αιτήσεων, ο έλεγχος γίνεται και στην αντίστοιχη procedure προτού αναλάβει ο trigger. Επίσης, έχουμε δημιουργήσει μια procedure με όνομα debuggg, την οποία μπορείτε να τρέξετε για να δείτε όλες τις λειτουργίες, με τα αποτελέσματά τους. Επιπλέον, μια παραδοχή που χρειάζεται να γίνει, είναι ότι ο κάθε employee, άπαξ και κάνει ένα apply για ένα συγκεκριμένο job_id, δεν πρέπει να κάνει δεύτερο με το ίδιο job_id.

Σχεδιασμός SQL ανα ερώτημα

3.1.2.1:

Το ερώτημα 3.1.2.1 μας ζητά να φτιάξουμε μηχανισμό για την εισαγωγή αλλά και για τη διαχείριση των αιτήσεων που κάνουν οι employees, με απαίτηση η κάθε αίτηση να έχει status (active, cancelled, finished). Όπως περιγράφεται στο κεφάλαιο 1, έχει προστεθεί ένα πεδίο με όνομα application_status που χρησιμοποιείται για αυτόν τον σκοπό.

Επίσης, εφόσον ζητείται οι νέες αιτήσεις να μπορούν να καταχωρηθούν το αργότερο 15 μέρες πριν την έναρξη της δουλειάς, φτιάξαμε το function με όνομα ApplicationDeadlineCheck, το οποίο παίρνει ως είσοδο το job_id της δουλειάς που θέλουμε να ελέγξουμε, και επιστρέφει TRUE όταν δεν έχουν περάσει οι 15 μέρες.

```
DROP FUNCTION IF EXISTS ApplicationDeadlineCheck;
DELIMITER $$
```

```
CREATE FUNCTION ApplicationDeadlineCheck (_job_id INT(11))
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
        DECLARE Job_start_date DATE;
  DECLARE DateDifference INT;
  SELECT start_date FROM job WHERE Job_id = _job_id INTO Job_start_date;
  IF((SELECT DATEDIFF(Job_Start_date, CURDATE()) AS DAYS) >=15)THEN
                RETURN TRUE;
        ELSE
                RETURN FALSE;
        END IF;
END$$
DELIMITER ;
```

Ακόμα μια απαίτηση που ζητείται, είναι να ελέγχεται αν κάποιος εργαζόμενος που προσπαθεί να κάνει αίτηση έχει ήδη 3 ενεργές αιτήσεις. Αν έχει 3, τότε δε θα μπορεί να αιτηθεί επιπλέον. Αυτό μπορεί να γίνει απλά με COUNT στον πίνακα applies. Με ίδιο σκεπτικό όπως πριν, δημιουργήσαμε function το οποίο δέχεται ως είσοδο το username του employee και επιστρέφει TRUE μόνο εάν έχει 1 ή 2 ενεργές αιτήσεις.

```
DROP FUNCTION IF EXISTS CheckIfApplicantHasLessThan3ApplicationsActive;
DELIMITER $$
CREATE FUNCTION CheckIfApplicantHasLessThan3ApplicationsActive (username
VARCHAR(30))
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
        DECLARE NoJobsApplied INT;
        SELECT COUNT(*) INTO NoJobsApplied FROM APPLIES WHERE cand_username =
username AND application_status = 'active';
  IF NoJobsApplied < 3 THEN
                RETURN TRUE;
  ELSE
                RETURN FALSE;
        END IF;
END$$
DELIMITER ;
```

Αντίστοιχα με το πρώτο ζητούμενο, το τρίτο ζητά να ελέγχουμε πριν την ακύρωση κάποιας αίτησης, εάν γίνεται το αργότερο 10 μέρες πριν την έναρξη της δουλειάς. Η CheckIfApplicationIsCancellable δέχεται ως είσοδο το username του employee αλλά και το job_id της αίτησης που είχε κάνει. Επιστρέφει FALSE εάν έχουν περάσει >= 11 μέρες. Δηλ. Αν η μέρα της ακύρωσης είναι μέχρι και η 10η, θα επιστρέψει TRUE, αν όμως είναι 11η κλπ θα επιστρέφει FALSE.

```
DROP FUNCTION IF EXISTS CheckIfApplicationIsCancellable; #WORKING
DELIMITER $$
CREATE FUNCTION CheckIfApplicationIsCancellable (username VARCHAR(30), _job_id INT(11))
#Elegxei an h mera pou paei na ginei cancell kapoia aithsh einai 10 meres prin thn enarksh (an einai 10 epistrefei true)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
        DECLARE Job_start_date DATE;
        #DECLARE DateDifference INT;
        SELECT start_date INTO Job_start_date FROM job WHERE job_id = _job_id;

  IF((SELECT DATEDIFF(Job_Start_date, CURDATE()) AS DAYS) >=10)THEN
                RETURN TRUE;
        ELSE
                RETURN FALSE;
        END IF;
END $$
DELIMITER ;
```

3.1.2.2:

Σε αυτό το ερώτημα, ζητείται μηχανισμός για την αξιολόγηση και την εξαγωγή (άρα και εισαγωγή στον πίνακα ApplicationsHistory) αποτελεσμάτων των αιτήσεων. Αρχικά, οι προϋποθέσεις είναι:

α) Αιτήσεις που είναι ακυρωμένες, να μην συμμετέχουν. Αυτό λύνεται με ένα WHERE application_status = ‘active’.

β) Κάθε αίτηση να αξιολογείται απο 2 evaluators. Εφόσον στην προπαρασκευαστική φάση, ο πίνακας job είχε θέση μόνο για έναν evaluator, τον αλλάξαμε και προσθέσαμε evaluator1 και evaluator2. *ΠΑΡΑΔΟΧΗ*: Θα πρέπει οπωσδήποτε να υπάρχει ένας evaluator για κάθε job (ουσιαστικά ο evaluator1 είναι αυτός που κάνει announce το εκάστοτε job). Αντίστοιχα διαμορφώσαμε τον πίνακα applies, έτσι ώστε να κρατά τα usernames των evaluator και τους βαθμούς που δίνουν.

γ) Την θέση την παίρνει ο employee με τον μεγαλύτερο μέσο όρο των βαθμών των ααξιολογητών. Σε περίπτωση ισοβαθμίας, επιλέγεται αυτός που έκανε νωρίτερα την αίτηση. (procedure WhatDidHeSaaaay)

δ) Μας δίνεται ο Πίνακας2 (στο pdf σελ6) από τον οποίο πρέπει να υπολογίζουμε τη βαθμολογία κάποιου evaluator που δεν έχει ορίσει βαθμό, σε κάποιον employee. Έτσι, κατασκευάσαμε την procedure CalculateAndAssignGrades η οποία, όταν βρίσκει βαθμό στους evaluator, τους περνάει στην ApplicationsHistory. Αν δεν βρει, χρησιμοποιεί τα query

```
SET total_points_degree_ = (SELECT SUM(CASE d.bathmida WHEN 'BSc' THEN 1 WHEN 'MSc' THEN 2 WHEN 'PhD' THEN 3 ELSE 0 END) FROM (has_degree hd JOIN degree d ON
```

```
    hd.degr_title = d.titlos AND hd.degr_idryma = d.idryma) WHERE hd.cand_username =
    candi_username);
SELECT SUM(num) INTO points_projects FROM project WHERE cand_id = candi_username;
SELECT COUNT(DISTINCT lang) INTO total_points_lang FROM languages WHERE cand_id =
    candi_username AND (FIND_IN_SET('EN', lang) > 0 OR FIND_IN_SET('GE', lang) > 0);
    #PaRAdOxi oti def lang gr
SET calculated_ev_1_points = total_points_degree_ + total_points_lang + points_projects;
SET avg_grade = (calculated_ev_1_points + v2_grade) / 2;
```

για να υπολογίσει βαθμό με βάση τον προαναφερθέντα πίνακα. *ΠΑΡΑΔΟΧΗ:* Οι employees έχουν ως μητρική γλώσσα τα Ελληνικά.

ε) Τέλος, όλες οι αιτήσεις για τις οποίες έχει υπολογιστεί βαθμός θα πρέπει να σβήνονται από τον πίνακα applies, και να μεταφέρονται με status finished στο ιστορικό (ApplicationsHistory).

Αξίζει να σημειωθεί ότι για την πιο εύκολη κατανόηση και ανάγνωση του αρχείου που περιλαμβάνει όλες τις λειτουργίες που ζητούνται, αποφασίσαμε να σπάσουμε σε μικρά procedures τα μεγάλα ζητούμενα. Για το ερώτημα 3.1.2.2, η procedure CalculateAndAssignGrades βρίσκει ή υπολογίζει τους βαθμούς, δημιουργεί έναν local πίνακα, στον οποίο κρατάμε όλα τα usernames και τους βαθμούς του job_id που δώθηκε, και αμέσως μετά τον υπολογισμό βρίσκουμε τον μεγαλύτερο μέσο όρο. Γνωρίζοντας τον μεγαλύτερο μέσο όρο, ασχέτως αν υπάρχει ισοβαθμία μπορούμε να μεταφέρουμε όλα τα δεδομένα στην applicationsHistory, με Got_the_job = FALSE (BOOL η οποία όταν είναι TRUE σημαίνει ότι ο employee πήρε τη θέση) όπου ο βαθμός που υπολογίστηκε δεν είναι ίσος με τον μεγαλύτερο μέσο όρο. Όπου ο βαθμός που υπολογίστηκε είναι ίσος με τον μεγαλύτερο μέσο όρο, δίνεται Got_the_job = TRUE.

Πριν τελειώσει η procedure, καλεί την procedure WhatDidHeSaaaay η οποία ελέγχει αν για το job_id που μας ενδιαφέρει υπάρχουν πάνω από 1 Got_the_job = TRUE. Αν ναι, τότε ανάμεσα στους employees που μας αφορούν, βρίσκει τις ημερομηνίες που έκαναν την αίτηση, και δίνει Got_the_job = TRUE μόνο, στον παλίοτερο.

Στο τέλος, διαγράφει τις αντίστοιχες εγγραφές από τον πίνακα applies.

3.1.2.3:

Το ιστορικό αιτήσεων εξηγήθηκε ήδη στα προηγούμενα ερωτήματα.

3.1.2.4:

Ζητείται να δημιουργήσουμε ένα νέο είδος χρήστη, τους διαχειριστές βάσης δεδομένων. (DBAs, Βλ. Περιγραφή βάσης δεδομένων σελ #####). Επίσης ζητείται να αποθηκεύουμε τις ενέργειες που κάνουν σε έναν πίνακα log (στην βάση ο πίνακας λέγεται logfiles). Δεν καταφέραμε να αποθηκεύονται οι ενέργειες *ΜΟΝΟ* των DBA καθώς χρειάζεται sessions, όμως αποθηκεύονται όλες οι ενέργειες των local users όπως ζητείται στο ερώτημα 3.1.4.1.

3.1.3.1:

Ζητείται stored procedure η οποία δέχεται ως είσοδο username κάποιου evaluator, username employee και job_id, για να ελεγχθεί ο βαθμός που έχει δοθεί, αν έχει δοθεί. Ζητείται μόνο η εμφάνιση του κάθε αποτελέσματος, όχι η εισαγωγή. Άρα και στην procedure EvaluatorGradeCheck που φαίνεται στο παράρτημα με τους κώδικες ανά ερώτημα, έχει μόνο select. Συνοπτικά, ο τρόπος λειτουργίας της, είναι όπως και στην procedure CalculateAndAssignGrades του ερωτήματος 3.1.2.2.

3.1.3.2:

Σε αυτό το ερώτημα, ζητείται να φτιάξουμε, κάτι σαν UI. Δηλαδή, ανάλογα την είσοδο που δίνεται (i,c ή a) να εκτελούνται και ανάλογες λειτουργίες. Πιο συγκεκριμένα, η procedure InsertCancelIREactivate_applications δέχεται ως είσοδο username employee, job_id και μία από τις τρεις προαναφερθέντες επιλογές. Με την χρήση των case clause, επιτυγχάνονται οι λειτουργίες της εισαγωγής, ακύρωσης ή της επανενεργοποίησης κάποιας ακυρωμένης θέσης. Άξιος αναφοράς είναι ο τρόπος με τον οποίο επιλέγεται κάποιος evaluator (*ισχύει η παραδοχή ότι πρέπει να υπάρχει τουλάχιστον ένας evaluator στον πίνακα job)*. Εφόσον δεν υπάρχει άλλο κριτήριο επιλογής evaluator για την κάλυψη κενής θέσης, εκτός από το ότι πρέπει να είναι και οι δύο από την ίδια εταιρεία, και μπορούν να υπάρχουν πολλοί evaluators σε μια εταιρεία (αυτό σημαίνει ότι ένα απλό select evaluator.username WHERE firm = eval1.firm επιστρέφει πολλά rows) χρησιμοποιήσαμε απλά MIN() στην επιλογή username.

3.1.3.3:

Η συγκεκριμενη procedure CallAlltoFinishJob, παίρνει ως είσοδο το job_id και καλεί μονάχα την CalculateAndAssignGrades και την WhatDidHeSayyy.

```
DROP PROCEDURE IF EXISTS CallAllToFinishJob;
DELIMITER $$
create procedure CallAllToFinishJob (IN jobid INT(11))
BEGIN
        CALL CalculateAndAssignGrades(jobid);
        CALL WhatDidHeSaaaay(jobid);
END $$
DELIMITER ;
```

3.1.3.4:

Θεωρία

Το ζητούμενο index είναι το παρακάτω. Αφορά τον πίνακα ApplicationsHistory, και συγκεκριμένα τις στήλες candi_username_history και job_id, μιας και αυτά θέλουμε να είναι τα αποτελέσματα των procedure που ζητούνται παρακάτω.

```
CREATE INDEX vaggel ON ApplicationsHistory(candi_username_history,job_id);
```

Ζητούμενο α)

Ζητείται procedure (procedure2) η οποία παίρνει ως είσοδο 2 βαθμούς και επιστρέφει τα usernames των employees και τα job_id των αιτήσεων που πήραν βαθμό ανάμεσα στο διάστημα των 2 βαθμών που δόθηκαν.

```
DROP procedure if exists procedure2;
DELIMITER $$
create procedure procedure2 (IN vathmos1 Decimal(3,1), vathmos2 Decimal(3,1))
BEGIN
    SELECT candi_username_history, job_id FROM ApplicationsHistory WHERE
(Average_grade BETWEEN vathmos1 AND vathmos2);
END $$
DELIMITER ;
```

Ζητούμενο β)

Ζητείται procedure (procedure3) η οποία παίρνει ως είσοδο το username κάποιου evaluator και εμφανίζει τα job_id και τα usernames των employees που έκανε evaluate.

```
DROP procedure if exists procedure3; # WORKING
DELIMITER $$
create procedure procedure3 (IN eva_username varchar(30))
BEGIN
    SELECT candi_username_history, job_id FROM ApplicationsHistory WHERE
Eval1_username = eva_username OR Eval2_username = eva_username;
END $$
DELIMITER ;
```

Χρόνοι

Θα εκτελέσουμε το παρακάτω query μια φορά χωρίς να έχουμε το index και μια με το index στον applicationshistory με 77564 rows.

```
select count(*) from applicationshistory;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

count(*)
77564

Procedure2 χωρίς index:

CALL procedure2(5,17)	52143 row(s) returned	0.000 sec / 0.047 sec

Procedure2 με index:

CALL procedure2(5,17)	52143 row(s) returned	0.000 sec / 0.047 sec

3.1.4.1:

Όπως αναφέρθηκε σε προηγούμενο ερώτημα, χωρίς τη χρήση log in και session, δεν γίνεται να γνωρίζουμε ποιος χρήστης είναι συνδεδεμένος, συνεπώς στα συγκεκριμένα trigger, το πεδίο user που ζητείται είναι απλά *CURRENT_USER()*. Επειδή θέλουμε να βεβαιωθούμε ότι ο πίνακας logfiles δεν θα έχει rows που για κάποιο λόγο δεν κατάφεραν να εκτελεστούν, σε όλα τα triggers του συγκεκριμένου ερωτήματος, χρησιμοποιούμε AFTER.

3.1.4.2:

Ζητείται trigger για τον έλεγχο μιας αίτησης πριν την εισαγωγή της. Άρα θα χρησιμοποιήσουμε BEFORE. Επίσης, εφόσον οι έλεγχοι είναι ίδιοι με αυτούς που κάναμε στα πρώτα ερωτήματα, καλώ τα αντίστοιχα functions.

3.1.4.3:

Αντίστοιχα με το προηγούμενο ερώτημα.

Κεφάλαιο 2: Κώδικας ανα ερώτημα και παραδείγματα procedure

3.1.2.1

ApplicationDeadlineCheck

```
DROP FUNCTION IF EXISTS ApplicationDeadlineCheck; #WORKING
DELIMITER $$
CREATE FUNCTION ApplicationDeadlineCheck (_job_id INT(11))
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
        DECLARE Job_start_date DATE;
  DECLARE DateDifference INT;
  SELECT start_date FROM job WHERE Job_id = _job_id INTO Job_start_date;
  IF((SELECT DATEDIFF(Job_Start_date, CURDATE()) AS DAYS) >=15)THEN
                RETURN TRUE;
        ELSE
                RETURN FALSE;
        END IF;
END$$
DELIMITER ;
```

Example

```
select * from job as message;
SET deadline = ApplicationDeadlineCheck(1);
SELECT 'deadline' AS 'Procedure', '1' AS 'Job', deadline AS 'In deadline';
SET deadline = ApplicationDeadlineCheck(2);
SELECT 'Deadline' AS 'Procedure', '2' AS 'Job', deadline AS 'In deadline';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_id	start_date	salary	position	edra	evaluator1	evaluator2	announcement_date	submission_date
1	2024-03-01 00:00:00	55000.00	Network Engineer	Berlin	daniel.jackson	john_doe	2023-01-01 00:00:00	2023-01-15
2	2022-11-15 00:00:00	68000.00	Data Analyst	Sydney	sophia.lopez	alice.smith	2023-01-05 00:00:00	2023-01-20

Procedure	Job	In deadline
deadline	1	1

Procedure	Job	In deadline
Deadline	2	0

CheckIfApplicantHasLessThan3ApplicationsActive

```
DROP FUNCTION IF EXISTS
CheckIfApplicantHasLessThan3ApplicationsActive; #WORKING
DELIMITER $$
CREATE FUNCTION CheckIfApplicantHasLessThan3ApplicationsActive
(username VARCHAR(30))
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
        DECLARE NoJobsApplied INT;
        SELECT COUNT(*) INTO NoJobsApplied FROM APPLIES WHERE
cand_username = username AND application_status = 'active';
 IF NoJobsApplied < 3 THEN
                RETURN TRUE;
 ELSE
                RETURN FALSE;
        END IF;
END$$
DELIMITER ;
```

Example

```
select * from applies;
SET less_than_3 = CheckIfApplicantHasLessThan3ApplicationsActive('olivia.martin');
SELECT 'Less than 3 applications' AS 'Procedure', 'olivia.martin' AS 'User', less_than_3 AS 'True / False';
SET less_than_3 = CheckIfApplicantHasLessThan3ApplicationsActive('chris.jones');
SELECT 'Less than 3 applications' AS 'Procedure', 'chris.jones' AS 'User', less_than_3 AS 'True / False';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
alex.green	10	cancelled	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06

Procedure	User	True / False
Less than 3 applications	olivia.martin	1

Procedure	User	True / False
Less than 3 applications	chris.jones	0

CheckIfApplicationIsCancellable

```
DROP FUNCTION IF EXISTS CheckIfApplicationIsCancellable; #WORKING
DELIMITER $$
CREATE FUNCTION CheckIfApplicationIsCancellable (username
VARCHAR(30), _job_id INT(11))
#Elegxei an h mera pou paei na ginei cancell kapoia aithsh einai 10 meres prin
thn enarksh (an einai 10 epistrefei true)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
        DECLARE Job_start_date DATE;
        #DECLARE DateDifference INT;
        SELECT start_date INTO Job_start_date FROM job WHERE job_id =
_job_id;

   IF((SELECT DATEDIFF(Job_Start_date, CURDATE()) AS DAYS) >=10)THEN
                RETURN TRUE;
        ELSE
                RETURN FALSE;
        END IF;
END $$
DELIMITER ;
```

Example

```
SET cancellable = CheckIfApplicationIsCancellable('dimitris_restas',10);
SELECT 'Cancellable' AS 'Procedure', 'dimitris_restas' AS 'User', '10' AS 'Job', cancellable AS 'Cancellable';
SET cancellable = CheckIfApplicationIsCancellable('jessica.clark',2);
SELECT 'Cancellable' AS 'Procedure', 'jessica.clark' AS 'User', '2' AS 'Job', cancellable AS 'Cancellable';
```

Procedure	User	Job	Cancellable
Cancellable	dimitris_restas	10	1

Procedure	User	Job	Cancellable
Cancellable	jessica.clark	2	0

3.1.2.2

CalculateAndAssignGrades

```
DROP PROCEDURE IF EXISTS CalculateAndAssignGrades; #NA SVHSW TA
SXOLIA HEHE
DELIMITER $$
CREATE PROCEDURE CalculateAndAssignGrades(IN _job_id INT(11))
```

```
BEGIN

DECLARE done INT DEFAULT FALSE;
DECLARE candi_username VARCHAR(30);
DECLARE username_on_ltable VARCHAR(30);
DECLARE avg_grade FLOAT;
DECLARE high_avg_grade DECIMAL(3,1);
DECLARE v1_grade INT;
DECLARE v2_grade INT;
DECLARE ev1_usr VARCHAR(30);
DECLARE ev2_usr VARCHAR(30);
DECLARE total_points FLOAT;
DECLARE total_points_lang INT;
DECLARE total_points_degree_ INT;
DECLARE points_projects INT;
DECLARE calculated_ev_1_points DECIMAL(2,1);
DECLARE calculated_ev_2_points DECIMAL(2,1);
DECLARE cur CURSOR FOR SELECT cand_username, Evaluator1_user,
Evaluator1_grade, Evaluator2_user, Evaluator2_grade FROM applies WHERE
job_id = _job_id AND application_status = 'ACTIVE';
DECLARE cur_for_cancelled CURSOR FOR SELECT cand_username,
Evaluator1_user, Evaluator2_user FROM applies WHERE job_id = _job_id AND
application_status = 'CANCELLED';
DECLARE cursOnlocal_table CURSOR FOR SELECT name_ FROM
Users_and_theirAverage WHERE job_id = _job_id;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TABLE IF EXISTS Users_and_theirAverage ;
CREATE TABLE Users_and_theirAverage (
        name_ VARCHAR(30),
   average DECIMAL(3,1),
   job_id INT(11)
);

        OPEN cur_for_cancelled;
   cancelled_applications: LOOP
                FETCH cur_for_cancelled INTO candi_username, ev1_usr,
ev2_usr;
      IF done THEN LEAVE cancelled_applications; END IF;
```

```
        INSERT INTO ApplicationsHistory (candi_username_history,
Eval1_username, Eval2_username, job_id, App_status, Average_grade,
Got_the_job)
                                VALUES (candi_username, ev1_usr, ev2_usr, _job_id,
DEFAULT, 0, FALSE);
    END LOOP;
    CLOSE cur_for_cancelled;
    SET done = FALSE;

  OPEN cur;
  read_loop: LOOP
    FETCH cur INTO candi_username, ev1_usr, v1_grade, ev2_usr, v2_grade;
    IF done THEN LEAVE read_loop; END IF;
                                        IF v1_grade < 1 AND v2_grade>1 THEN
                                                SET total_points_degree_ = (SELECT
SUM(CASE d.bathmida WHEN 'BSc' THEN 1 WHEN 'MSc' THEN 2 WHEN 'PhD'
THEN 3 ELSE 0 END)
                                                FROM (has_degree hd JOIN degree d ON
hd.degr_title = d.titlos AND hd.degr_idryma = d.idryma) WHERE
hd.cand_username = candi_username);

                                                SELECT SUM(num) INTO points_projects
FROM project WHERE cand_id = candi_username;
                                                SELECT COUNT(DISTINCT lang) INTO
total_points_lang FROM languages
                                                        WHERE cand_id = candi_username
AND (FIND_IN_SET('EN', lang) > 0 OR FIND_IN_SET('GE', lang) > 0);
#PaRAdOxi oti def lang gr
                                                SET calculated_ev_1_points =
total_points_degree_ + total_points_lang + points_projects;
                                                SET avg_grade = (calculated_ev_1_points
+ v2_grade) / 2;

                                        ELSEIF v2_grade < 1 AND v1_grade > 1 THEN
                                                SET total_points_degree_ = (SELECT
SUM(CASE d.bathmida WHEN 'BSc' THEN 1 WHEN 'MSc' THEN 2 WHEN 'PhD'
THEN 3 ELSE 0 END)
                                                FROM (has_degree hd JOIN degree d ON
hd.degr_title = d.titlos AND hd.degr_idryma = d.idryma) WHERE
hd.cand_username = candi_username);
```

```
                                        SELECT SUM(num) INTO points_projects
FROM project WHERE cand_id = candi_username;
                                        SELECT COUNT(DISTINCT lang) INTO
total_points_lang FROM languages
                                                WHERE cand_id = candi_username
AND (FIND_IN_SET('EN', lang) > 0 OR FIND_IN_SET('GE', lang) > 0);
#PaRAdOxi oti def lang gr
                                        SET calculated_ev_2_points =
total_points_degree_ + total_points_lang + points_projects;
                                        SET avg_grade = (v1_grade +
calculated_ev_2_points) / 2;

                                ELSEIF v2_grade < 1 AND v1_grade < 1 THEN
                                        SET total_points_degree_ = (SELECT
SUM(CASE d.bathmida WHEN 'BSc' THEN 1 WHEN 'MSc' THEN 2 WHEN 'PhD'
THEN 3 ELSE 0 END)
                                        FROM (has_degree hd JOIN degree d ON
hd.degr_title = d.titlos AND hd.degr_idryma = d.idryma) WHERE
hd.cand_username = candi_username);

                                        SELECT SUM(num) INTO points_projects
FROM project WHERE cand_id = candi_username;
                                        SELECT COUNT(DISTINCT lang) INTO
total_points_lang FROM languages
                                                WHERE cand_id = candi_username
AND (FIND_IN_SET('EN', lang) > 0 OR FIND_IN_SET('GE', lang) > 0);
#PaRAdOxi oti def lang gr
                                        SET calculated_ev_2_points =
total_points_degree_ + total_points_lang + points_projects;
                                        SET calculated_ev_1_points =
total_points_degree_ + total_points_lang + points_projects;
                                        SET avg_grade = (v1_grade + v2_grade) /
2;
                        ELSE
                                SET avg_grade = (v1_grade + v2_grade) / 2;
                                SET avg_grade = (SELECT CONVERT(avg_grade,
DECIMAL(3,1)));
                        END IF;
```

```
        INSERT INTO ApplicationsHistory (candi_username_history,
Eval1_username, Eval2_username, job_id, App_status, Average_grade,
Got_the_job)
                                VALUES (candi_username, ev1_usr, ev2_usr, _job_id,
DEFAULT, avg_grade, FALSE);

        INSERT INTO Users_and_theirAverage VALUES
(candi_username,avg_grade,_job_id);
            END LOOP;
    SET done = FALSE;
            CLOSE cur;

    SET high_avg_grade = (SELECT MAX(average) FROM
Users_and_theirAverage WHERE job_id = _job_id);
            OPEN cursOnlocal_table;

    Update_application_history: LOOP #Me got the job true opou
high_average_grade = me average
        FETCH cursOnlocal_table INTO username_on_ltable;
        IF done THEN LEAVE Update_application_history; END IF;
                    UPDATE ApplicationsHistory SET Got_the_job = TRUE WHERE
job_id = _job_id AND username_on_ltable = candi_username_history AND
average_grade = high_avg_grade;
    END LOOP;
            DROP TABLE IF EXISTS Users_and_theirAverage;
    CLOSE cursOnlocal_table;

    CALL WhatDidHeSaaaay(_job_id);
END $$
DELIMITER ;
```

Example

```
#slect 'calculate and asign grades' as message;
select * from applies where job_id = 10;
select * from applicationshistory;
call CalculateAndAssignGrades(10);
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_da
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
alex.green	10	cancelled	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06

```
select * from applicationshistory;
call CalculateAndAssignGrades(10);
```

lt Grid | Filter Rows: | Export: | Wrap Cell Content:

candi_username_history	Eval1_username	Eval2_username	job_id	App_status	Average_grade	Got_the_job

```
call CalculateAndAssignGrades(10);
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

candi_username_history	Eval1_username	Eval2_username	job_id	App_status	Average_grade	Got_the_job
alex.green	vlasis_restas	koukloux_klanios	10	finished	0.0	0
chris.jones	vlasis_restas	koukloux_klanios	10	finished	17.5	1
dimitris_restas	vlasis_restas	koukloux_klanios	10	finished	17.5	1
jessica.clark	vlasis_restas	koukloux_klanios	10	finished	5.5	0
olivia.martin	vlasis_restas	koukloux_klanios	10	finished	17.5	1

WhatDidHeSaaaay

DROP PROCEDURE IF EXISTS WhatDidHeSaaaay; #WORKING

```
DELIMITER $$
CREATE PROCEDURE WhatDidHeSaaaay (IN jobid INT(11))
BEGIN
        DECLARE done INT DEFAULT FALSE;
        DECLARE temp_applicant_username1 VARCHAR(30);
   DECLARE temp_applicant_username2 VARCHAR(30);
        DECLARE PosaGotTheJobUparxoun INT;
        DECLARE temp_date1 DATE;
        DECLARE temp_date2 DATE;
        DECLARE GotTheJobC bool;
   DECLARE usr VARCHAR(30);
        DECLARE got_the_job_true_cursor CURSOR FOR SELECT
Got_the_job, candi_username_history FROM ApplicationsHistory WHERE job_id
= jobid AND Got_the_job = TRUE;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;
   SET temp_applicant_username1 = NULL;
   SET temp_applicant_username2 = NULL;

                OPEN got_the_job_true_cursor;
                SELECT COUNT(*) FROM ApplicationsHistory WHERE
Got_the_job = TRUE INTO PosaGotTheJobUparxoun;
                        IF (PosaGotTheJobUparxoun > 1) THEN

                                Loopa: LOOP
                                        FETCH got_the_job_true_cursor INTO
GotTheJobC, usr;
            IF done THEN LEAVE Loopa; END IF;
                                                IF (temp_applicant_username1 IS
NULL) THEN
                                                        SET
temp_applicant_username1 = usr;
                    SELECT Sumbission_date INTO temp_date1 FROM applies
WHERE cand_username = temp_applicant_username1 AND job_id = jobid;
                                                        SELECT
temp_applicant_username1 AS MESSAGE;
                        SELECT temp_date1 AS MESSAGE;

                    FETCH got_the_job_true_cursor INTO GotTheJobC, usr;
                                                        SELECT usr AS
MESSAGE;
```

```
                    SET temp_applicant_username2 = usr;
                    SELECT Sumbission_date INTO temp_date2 FROM applies
WHERE cand_username = temp_applicant_username2 AND job_id = jobid;
                                                        SELECT temp_date2
AS MESSAGE;
                                        ELSEIF
(temp_applicant_username2 IS NULL) THEN
                                                SELECT Sumbission_date
INTO temp_date2 FROM applies WHERE cand_username = joker_usr AND
job_id = jobid;
                                        END IF;

                IF((SELECT DATEDIFF(temp_date1, temp_date2) AS DAYS) < 0)
THEN
                                                UPDATE ApplicationsHistory
SET Got_the_job = FALSE WHERE candi_username_history =
temp_applicant_username2 AND job_id = jobid;
                                                SET
temp_applicant_username1 = temp_applicant_username2;
                    SET temp_applicant_username2 = NULL;
                                        ELSEIF ((SELECT
DATEDIFF(temp_date1, temp_date2) AS DAYS) > 0) THEN
                                                UPDATE ApplicationsHistory
SET Got_the_job = FALSE WHERE candi_username_history =
temp_applicant_username1 AND job_id = jobid;
                                        ELSE
                                                SELECT 'einai idia dates den
kserame ti na kanoume sorry' AS MESSAGE;
                END IF;
                            END LOOP Loopa;
                            CLOSE got_the_job_true_cursor;
                        END IF;
    DELETE FROM applies WHERE job_id = jobid;
END$$
DELIMITER ;
```

Example

```
call WhatDidHeSaaaay(10);
select * from applicationshistory;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

candi_username_history	Eval1_username	Eval2_username	job_id	App_status	Average_grade	Got_the_jo
alex.green	vlasis_restas	koukloux_klanios	10	finished	0.0	0
chris.jones	vlasis_restas	koukloux_klanios	10	finished	17.5	1
dimitris_restas	vlasis_restas	koukloux_klanios	10	finished	17.5	0
jessica.clark	vlasis_restas	koukloux_klanios	10	finished	5.5	0
olivia.martin	vlasis_restas	koukloux_klanios	10	finished	17.5	0

3.1.3.1

EvaluatorGradeCheck

```
DROP procedure IF EXISTS EvaluatorGradeCheck; #WORKING
DELIMITER $$
CREATE PROCEDURE EvaluatorGradeCheck(IN eval_username
VARCHAR(30), empl_username VARCHAR(30), id_job INT(11), OUT grade
DECIMAL(2,1))
BEGIN
        DECLARE evaluator VARCHAR(30);
        DECLARE employee VARCHAR(30);
        DECLARE id INT(11);
        DECLARE total_points_degree_ INT;
        DECLARE final_grade INT;
        DECLARE points_projects INT;
        DECLARE total_points_lang INT;
        DECLARE eval1_user VARCHAR(30);
        DECLARE eval2_user VARCHAR(30);

        IF ((SELECT Evaluator1_user FROM applies WHERE job_id = id_job
AND cand_username = empl_username) IS NOT NULL) THEN
                SET eval1_user = (SELECT evaluator1_user FROM applies
WHERE job_id = id_job AND cand_username = empl_username);
                IF (eval1_user = eval_username) THEN
```

```
                    SET final_grade = (SELECT Evaluator1_grade FROM
applies WHERE job_id = id_job AND cand_username = empl_username);
                    IF (final_grade < 1) THEN
                        #ypologismos vathmou apo pinaka 2
                        SET total_points_degree_ = (SELECT SUM(CASE
d.bathmida WHEN 'BSc' THEN 1 WHEN 'MSc' THEN 2 WHEN 'PhD' THEN 3
ELSE 0 END)

FROM has_degree hd JOIN degree d ON hd.degr_title = d.titlos AND
hd.degr_idryma = d.idryma

WHERE hd.cand_username = empl_username);
                        SELECT num INTO points_projects FROM project
WHERE cand_id = empl_username;
                        SELECT COUNT(DISTINCT lang) INTO
total_points_lang FROM languages
                            WHERE cand_id = empl_username AND
(FIND_IN_SET('EN', lang) > 0 OR FIND_IN_SET('GE', lang) > 0); #PaRAdOxi oti
def lang gr
            SET final_grade = total_points_degree_ + total_points_lang +
points_projects;
                    END IF;
                ELSEIF ((SELECT Evaluator2_user FROM applies WHERE job_id
= id_job AND cand_username = empl_username) IS NOT NULL) THEN
                    SET eval2_user = (SELECT evaluator1_user FROM
applies WHERE job_id = id_job AND cand_username = empl_username);
                    IF (eval2_user = eval_username) THEN
                        SET final_grade = (SELECT Evaluator2_grade
FROM applies WHERE job_id = id_job AND cand_username = empl_username);
                        IF (final_grade < 1) THEN
                            #ypologismos vathmou apo pinaka 2
                            SET total_points_degree_ = (SELECT
SUM(CASE d.bathmida WHEN 'BSc' THEN 1 WHEN 'MSc' THEN 2 WHEN 'PhD'
THEN 3 ELSE 0 END)

FROM has_degree hd JOIN degree d ON hd.degr_title = d.titlos AND
hd.degr_idryma = d.idryma
                        WHERE hd.cand_username = empl_username);
                            SELECT num INTO points_projects FROM
project WHERE cand_id = empl_username;
```

```
                                        SELECT COUNT(DISTINCT lang) INTO
total_points_lang FROM languages
                                                WHERE cand_id = empl_username
AND (FIND_IN_SET('EN', lang) > 0 OR FIND_IN_SET('GE', lang) > 0);
#PaRAdOxi oti def lang gr
                SET final_grade = total_points_degree_ + total_points_lang +
points_projects;
                                END IF;
                        ELSE
                                SET final_grade = 0;
                        END IF;
                END IF;
        ELSE
                SET final_grade = 0;
        END IF;

SELECT final_grade;
END$$
DELIMITER ;
```

Example

```
575    SELECT * from applies where cand_username = 'jessica.clark' and job_id = 10;
576    select * from languages where cand_id = 'jessica.clark';
577    select * FROM (has_degree hd JOIN degree d ON hd.degr_title = d.titlos AND hd.degr_idryma = d.idryma)
578                                    WHERE hd.cand_username = 'jessica.clark';
579    CALL EvaluatorGradeCheck('vlasis_restas', 'jessica.clark', 10, test);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06

cand_id	lang
jessica.clark	GR

degr_title	degr_idryma	cand_username	etos	grade	titlos	idryma	bathmida
Customer Relations	University of Toronto	jessica.clark	2017	8.7	Customer Relations	University of Toronto	MSc

final_grade
3

3.1.3.2

InsertCancellREactivate_applications

```
DROP PROCEDURE IF EXISTS InsertCancellREactivate_applications; #
WORKING
DELIMITER $$
CREATE PROCEDURE InsertCancellREactivate_applications(IN applicant
VARCHAR(30), id INT(11), selection_ CHAR(1))
BEGIN
        DECLARE selection CHAR(1);
   DECLARE firm CHAR(9);
        DECLARE eval1_uname VARCHAR(30);
        DECLARE eval2_uname VARCHAR(30);
        DECLARE ApplicationDeadlineCheck BOOL;
        DECLARE CheckIfApplicantHasLessThan3ApplicationsActive BOOL;
        DECLARE CheckIfApplicationIsCancellable BOOL;
   declare username VARCHAR(30);
   declare jobid INT(11);
   declare today date;
   SET today = (SELECT (CURDATE()));
   SET CheckIfApplicationIsCancellable =
CheckIfApplicationIsCancellable(applicant,id);
        SET ApplicationDeadlineCheck = ApplicationDeadlineCheck(id);
        SET CheckIfApplicantHasLessThan3ApplicationsActive =
CheckIfApplicantHasLessThan3ApplicationsActive(applicant);

   select CheckIfApplicationIsCancellable as message;


        CASE selection_

                WHEN 'i' THEN  #WORKING
                        IF (ApplicationDeadlineCheck AND
CheckIfApplicantHasLessThan3ApplicationsActive) THEN
                                set eval1_uname = (SELECT evaluator1 FROM job
WHERE job_id = id);
            set eval2_uname = (SELECT evaluator2 FROM job WHERE job_id =
id);
```

```
            SET firm = (SELECT firm FROM evaluator WHERE username =
eval1_uname);

                                    IF eval2_uname IS NULL THEN
                                            #asign eval2
                set eval2_uname = (SELECT MIN(username) FROM evaluator
WHERE firm = firm ORDER BY username LIMIT 1);
                SELECT eval1_uname, eval2_uname, firm;
                UPDATE job SET evaluator2 = eval2_uname WHERE id = id;
                INSERT INTO applies VALUES
(applicant,id,DEFAULT,eval1_uname,0,eval2_uname,0,CURDATE());
                                            SELECT 'Application sumbited.' AS
MESAGE;
            ELSE
                                            #proceed to application
                INSERT INTO applies VALUES
(applicant,id,DEFAULT,eval1_uname,0,eval2_uname,0,CURDATE());
                                            SELECT 'Application sumbited.' AS
MESAGE;
            END IF;
          ELSE
                                    SELECT 'Either deadline for applications has
passed, or user already has 3 applications active.' AS MESSAGE;
                            END IF;



   WHEN 'c' THEN #WORKING
                            IF (CheckIfApplicationIsCancellable) THEN
                                    IF (SELECT Application_status FROM applies
WHERE cand_username = applicant AND job_id = id) = 'active' THEN
                                            UPDATE applies SET cand_username =
applicant, Application_status = 'cancelled', job_id = id WHERE cand_username =
applicant AND job_id = id;
                SELECT 'Application cancelled.' AS MESSAGE;
                                    ELSEIF (SELECT Application_status FROM
applies WHERE cand_username = applicant AND job_id = id) = 'cancelled'
THEN
                                            SELECT 'Application allready cancelled.' AS
MESSAGE;
                                    ELSE
```

```
                                        SELECT 'Application not found.' AS
MESSAGE;
                                END IF;
                        ELSE
                                SELECT 'Application cannot be cancelled as the 10
days before cancellation requirement have passed.' AS MESSAGE;
                        END IF;



    WHEN 'a' THEN #WORKING
                        IF (ApplicationDeadlineCheck AND
CheckIfApplicantHasLessThan3ApplicationsActive) THEN
                                IF (SELECT Application_status FROM applies
WHERE cand_username = applicant AND job_id = id) = 'cancelled' THEN
                                        UPDATE applies SET Application_status =
'active' WHERE cand_username = applicant AND job_id = id;
                SELECT 'Application activated' AS MESSAGE;
                                ELSEIF (SELECT Application_status FROM
applies WHERE cand_username = applicant AND job_id = id) = 'active' THEN
                                        SELECT 'Application allready active.' AS
MESSAGE;
                                ELSE
                                        SELECT 'Application not found.' AS
MESSAGE;
                                END IF;
                        ELSE
                                SELECT 'Either deadline for applications has
passed, or user already has 3 applications active.' AS MESSAGE;
                        END IF;
    ELSE
                SELECT 'Wrong input' AS MESSAGE;
        END CASE;
END$$
DELIMITER ;
```

Example

```
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',10,'a');
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',10,'c');
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',1,'i');
SELECT * FROM applies;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
alex.green	10	cancelled	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06
jessica.clark	2	active	sophia.lopez	0	alice.smith	0	2023-02-06

```
CALL InsertCancellREactivate_applications('alex.green',10,'a');
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',10,'c');
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',1,'i');
SELECT * FROM applies;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MESSAGE
Application activated

```
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',10,'c');
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',1,'i');
SELECT * FROM applies;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
alex.green	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06
jessica.clark	2	active	sophia.lopez	0	alice.smith	0	2023-02-06

```
CALL InsertCancellREactivate_applications('alex.green',10,'c');
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',1,'i');
SELECT * FROM applies;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MESSAGE
Application cancelled.

```
SELECT * FROM applies;
CALL InsertCancellREactivate_applications('alex.green',1,'i');
SELECT * FROM applies;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
alex.green	10	cancelled	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06
jessica.clark	2	active	sophia.lopez	0	alice.smith	0	2023-02-06

```
606    CALL InsertCancellREactivate_applications('alex.green',1,'i');
607    SELECT * FROM applies;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MESAGE
Application sumbited.

```
607    SELECT * FROM applies;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
alex.green	10	cancelled	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06
jessica.clark	2	active	sophia.lopez	0	alice.smith	0	2023-02-06
alex.green	1	active	daniel.jackson	0	john_doe	0	2024-01-22

3.1.3.3

CallAllToFinishJob

```
DROP PROCEDURE IF EXISTS CallAllToFinishJob;
DELIMITER $$
create procedure CallAllToFinishJob (IN jobid INT(11))
BEGIN
        CALL CalculateAndAssignGrades(jobid);
   CALL WhatDidHeSaaaay(jobid);
END $$
DELIMITER ;
```

Example

```
SELECT * FROM APPLIES;
SELECT * FROM APPLICATIONSHISTORY;
CALL CallAllToFinishJob(10);
SELECT * FROM APPLIES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
alex.green	10	cancelled	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06
jessica.clark	2	active	sophia.lopez	0	alice.smith	0	2023-02-06

```
SELECT * FROM APPLIES;
SELECT * FROM APPLICATIONSHISTORY;
CALL CallAllToFinishJob(10);
SELECT * FROM APPLIES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
jessica.clark	2	active	sophia.lopez	0	alice.smith	0	2023-02-06

```
SELECT * FROM APPLIES;
SELECT * FROM APPLICATIONSHISTORY;
CALL CallAllToFinishJob(10);
SELECT * FROM APPLIES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

candi_username_history	Eval1_username	Eval2_username	job_id	App_status	Average_grade	Got_the_job
alex.green	vlasis_restas	koukloux_klanios	10	finished	0.0	0
chris.jones	vlasis_restas	koukloux_klanios	10	finished	17.5	1
dimitris_restas	vlasis_restas	koukloux_klanios	10	finished	17.5	0
jessica.clark	vlasis_restas	koukloux_klanios	10	finished	5.5	0
olivia.martin	vlasis_restas	koukloux_klanios	10	finished	17.5	0

```
SELECT * FROM APPLIES;
SELECT * FROM APPLICATIONSHISTORY;
CALL CallAllToFinishJob(10);
SELECT * FROM APPLIES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

candi_username_history	Eval1_username	Eval2_username	job_id	App_status	Average_grade	Got_the_job

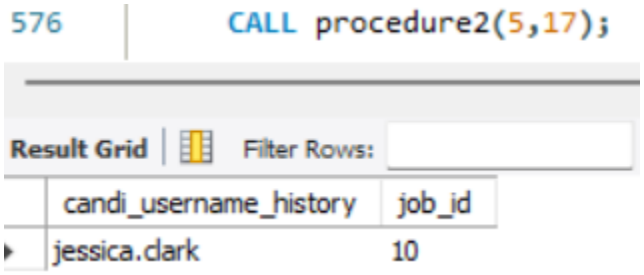
3.1.3.4

α) procedure2

DROP procedure if exists procedure2; # WORKING

```
DELIMITER $$
create procedure procedure2 (IN vathmos1 Decimal(3,1), vathmos2
Decimal(3,1))
BEGIN
        SELECT candi_username_history, job_id FROM ApplicationsHistory
WHERE (Average_grade BETWEEN vathmos1 AND vathmos2);
END $$
DELIMITER ;
```

Example



β) procedure3

```
DROP procedure if exists procedure3; # WORKING
DELIMITER $$
create procedure procedure3 (IN eva_username varchar(30))
BEGIN
        SELECT candi_username_history, job_id FROM ApplicationsHistory
WHERE Eval1_username = eva_username OR Eval2_username =
eva_username;
END $$
DELIMITER ;
```

Example

```
576    CALL procedure3('vlasis_restas');
```

Result Grid | Filter Rows: | Export:

candi_username_history	job_id
alex.green	10
chris.jones	10
dimitris_restas	10
jessica.clark	10
olivia.martin	10

Κεφάλαιο 3: Κώδικας και παραδείγματα triggers

3.1.4.1

job_insert_trigger

```
DROP TRIGGER IF EXISTS job_insert_trigger; # WORKING
DELIMITER $$
CREATE TRIGGER job_insert_trigger AFTER INSERT ON job FOR EACH ROW
BEGIN
        INSERT INTO logfiles
                VALUES (CURRENT_USER(),'INSERT', 'job', NOW());
END $$
DELIMITER ;
```

Example

Μιας και όλα τα triggers λειτουργούν με ακριβώς ίδιο σκεπτικό, και μόνες οι διαφορές μεταξύ τους είναι τα tables, τα παραδείγματα για τις update και delete μπορούν να παραβλεθούν.

```
select * from logfiles;
insert into job values (NULL,'2024-08-03',14000,'test','kolokotronitsi','vlasis_restas','sophia.lopez',CURDATE(),CURDATE());
select * from job where edra = 'kolokotronitsi';
select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DBAusername	Edited	Table_	Date

```
insert into job values (NULL,'2024-08-03',14000,'test','kolokotronitsi','vlasis_restas','sophia.lopez',CURDATE(),CURDATE());
select * from job where edra = 'kolokotronitsi';
select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_id	start_date	salary	position	edra	evaluator1	evaluator2	announcement_date	submission_date
12	2024-08-03 00:00:00	14000.00	test	kolokotronitsi	vlasis_restas	sophia.lopez	2024-01-22 00:00:00	2024-01-22

```
select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wr

DBAusername	Edited	Table_	Date
root@localhost	insert	job	2024-01-22 05:04:13

job_update_trigger

```
DROP TRIGGER IF EXISTS job_update_trigger; # WORKING
DELIMITER $$
CREATE TRIGGER job_update_trigger AFTER UPDATE ON job FOR EACH
ROW
BEGIN
	INSERT INTO logfiles
		VALUES (CURRENT_USER(),'UPDATE', 'job', NOW());
END $$
DELIMITER ;
```

job_delete_trigger

```
DROP TRIGGER IF EXISTS job_delete_trigger; # WORKING
DELIMITER $$
CREATE TRIGGER job_delete_trigger AFTER DELETE ON job FOR EACH
ROW
BEGIN
	INSERT INTO logfiles
		VALUES (CURRENT_USER(),'DELETE', 'job', NOW());
END $$
DELIMITER ;
```

user_inserttrigger

```
DROP TRIGGER IF EXISTS user_inserttrigger; # WORKING
DELIMITER $$
CREATE TRIGGER user_inserttrigger
AFTER INSERT ON user
FOR EACH ROW
BEGIN
	INSERT INTO logfiles
		VALUES (CURRENT_USER(),'INSERT', 'user', NOW());
END $$
DELIMITER ;
```

Example

Μιας και όλα τα triggers λειτουργούν με ακριβώς ίδιο σκεπτικό, και μόνες οι διαφορές μεταξύ τους είναι τα tables, τα παραδείγματα για τις update και delete μπορούν να παραβλεθούν.

```
577    select * from logfiles;
578    insert into user values ('pigkouinos','pigkouinaki','kurios','pigkouinos',CURDATE(),'kurios.pigkouinos@igklou.is');
579    select * from user where username = 'pigkouinos';
580    select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DBAusername	Edited	Table_	Date

```
578    insert into user values ('pigkouinos','pigkouinaki','kurios','pigkouinos',CURDATE(),'kurios.pigkouinos@igklou.is');
579    select * from user where username = 'pigkouinos';
580    select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

username	password	name	lastname	reg_date	email
pigkouinos	pigkouinaki	kurios	pigkouinos	2024-01-22 00:00:00	kurios.pigkouinos@igklou.is

```
580    select * from logfiles;
```

Result Grid | Filter Rows: | Export:

DBAusername	Edited	Table_	Date
root@localhost	insert	user	2024-01-22 05:11:10

user_updatetrigger

```
DROP TRIGGER IF EXISTS user_updatetrigger; # WORKING
DELIMITER $$
CREATE TRIGGER user_updatetrigger
AFTER UPDATE ON user
FOR EACH ROW
BEGIN
        INSERT INTO logfiles
                VALUES (CURRENT_USER(),'UPDATE', 'user', NOW());
END $$
DELIMITER ;
```

user_deletetrigger

```
DROP TRIGGER IF EXISTS user_deletetrigger; # WORKING
```

```
DELIMITER $$
CREATE TRIGGER user_deletetrigger
AFTER DELETE ON user
FOR EACH ROW
BEGIN
	INSERT INTO logfiles
		VALUES (CURRENT_USER(),'DELETE', 'user', NOW());
END $$
DELIMITER ;
```

degree_insert_trigger

```
DROP TRIGGER IF EXISTS degree_insert_trigger; # WORKING
DELIMITER $$
CREATE TRIGGER degree_insert_trigger
AFTER INSERT ON degree
FOR EACH ROW
BEGIN
	INSERT INTO logfiles (action_type, table_name, username)
		VALUES (CURRENT_USER(),'INSERT', 'degree', NOW());
END $$
DELIMITER ;
```

Example

Μιας και όλα τα triggers λειτουργούν με ακριβώς ίδιο σκεπτικό, και μόνες οι διαφορές μεταξύ τους είναι τα tables, τα παραδείγματα για τις update και delete μπορούν να παραβλεθούν.

```
insert into degree values ('true_ceidas','CEID','BSc');
select * from degree where idryma = 'CEID';
select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

titlos	idryma	bathmida
true_ceidas	CEID	BSc

```
select * from logfiles;
insert into degree values ('true_ceidas','CEID','BSc');
select * from degree where idryma = 'CEID';
select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DBAusername	Edited	Table_	Date

```
select * from logfiles;
```

Result Grid | Filter Rows: | Export:

DBAusername	Edited	Table_	Date
root@localhost	insert	degree	2024-01-22 05:17:19

degree_update_trigger

```
DROP TRIGGER IF EXISTS degree_update_trigger; # WORKING
DELIMITER $$
CREATE TRIGGER degree_update_trigger
AFTER UPDATE ON degree
FOR EACH ROW
BEGIN
        INSERT INTO logfiles
                VALUES (CURRENT_USER(),'UPDATE', 'degree', NOW());
END $$
DELIMITER ;
```

degree_delete_trigger

```
DROP TRIGGER IF EXISTS degree_delete_trigger; # WORKING
DELIMITER $$
CREATE TRIGGER degree_delete_trigger
AFTER DELETE ON degree
FOR EACH ROW
```

```
BEGIN
        INSERT INTO logfiles  (action_type, table_name, username)
                VALUES (CURRENT_USER(),'DELETE', 'degree', NOW());
END $$
DELIMITER ;
```

3.1.4.2

CheckBeforeApply

```
DROP TRIGGER IF EXISTS CheckBeforeApply; # WORKING
DELIMITER $$
CREATE TRIGGER CheckBeforeApply BEFORE INSERT ON applies FOR
EACH ROW
BEGIN
        DECLARE applications BOOL;
        DECLARE deadline BOOL;
        SET applications =
CheckIfApplicantHasLessThan3ApplicationsActive(new.cand_username);
        SET deadline = ApplicationDeadlineCheck(new.job_id);
        IF (applications = FALSE AND deadline = FALSE) THEN
                signal sqlstate '45000' set message_text = 'Either deadline for
applications has passed, or user already has 3 applications active.';
        END IF;
END $$
DELIMITER ;
```

Example

Μιας και τα συγκεκριμένα triggers χρησιμοποιούν τις functions που έχουμε ήδη δει στα πρώτα ζητούμενα, γνωρίζουμε ότι δουλεύουν σε κάθε περίπτωση.

```
select * from applies;
insert into applies values ('mark.evans','11','active','vlasis_restas',0,'koukloux_klanios',0,CURDATE());
select * from applies where cand_username = 'mark.evans';
select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
dimitris_restas	10	active	vlasis_restas	18	koukloux_klanios	17	2023-04-10
chris.jones	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	11	active	vlasis_restas	18	koukloux_klanios	17	2023-02-01
chris.jones	1	active	daniel.jackson	18	john_doe	17	2023-02-01
alex.green	10	cancelled	vlasis_restas	18	koukloux_klanios	17	2023-02-05
olivia.martin	10	active	vlasis_restas	18	koukloux_klanios	17	2023-02-06
jessica.clark	10	active	vlasis_restas	0	koukloux_klanios	8	2023-02-06
jessica.clark	2	active	sophia.lopez	0	alice.smith	0	2023-02-06

```
insert into applies values ('mark.evans','11','active','vlasis_restas',0,'koukloux_klanios',0,CURDATE());
select * from applies where cand_username = 'mark.evans';
select * from logfiles;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cand_username	job_id	Application_Status	Evaluator1_user	Evaluator1_grade	Evaluator2_user	Evaluator2_grade	Sumbission_date
mark.evans	11	active	vlasis_restas	0	koukloux_klanios	0	2024-01-22

3

3.1.4.3

UpdateApplications

```
DROP TRIGGER IF EXISTS UpdateApplications; # WORKING
DELIMITER $$
CREATE TRIGGER UpdateApplications BEFORE UPDATE ON applies FOR
EACH ROW
BEGIN
        DECLARE applications BOOL;
        DECLARE cancellable BOOL;
        SET cancellable = CheckIfApplicationIsCancellable(new.cand_username,
new.job_id);
        SET applications =
CheckIfApplicantHasLessThan3ApplicationsActive(new.cand_username);

        IF (cancellable = FALSE) THEN
```

```
                signal sqlstate '45000' set message_text = '15 days before
cancellation of application have passed. Application can not be cancelled.';
        END IF;
        IF (applications = FALSE) THEN
                signal sqlstate '45000' set message_text = 'Candidate allready has
3 active applications. Application can not be cancelled.';
        END IF;
END $$
DELIMITER ;
```

Example

Αντίστοιχα και σε αυτόν τον trigger. Η λειτουργία του έχει δοκιμαστεί νωρίτερα.

Κεφάλαιο 4: GUI

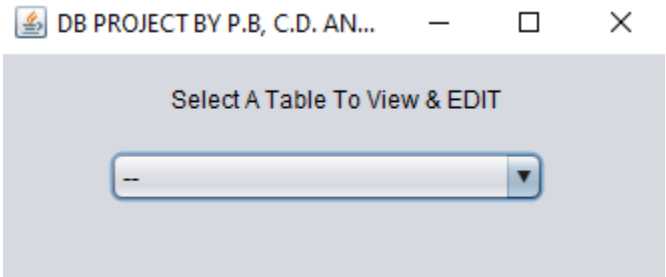
Γενικά

Στο συγκεκριμένο GUI υλοποιούνται οι λειτουργίες που ζητούνται ακριβώς, δηλαδή υπάρχουν οι εξής επιλογές:

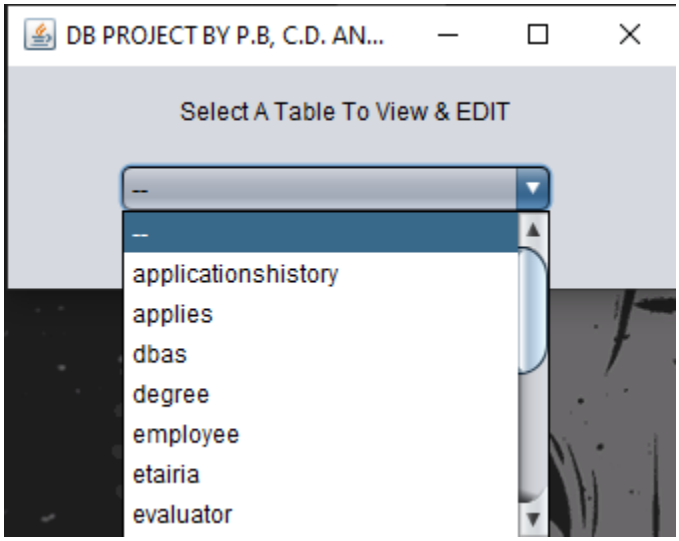
- Ενημέρωση πεδίων πίνακα (update)
- Εισαγωγή νέων πεδίων στον πίνακα (insert)
- Διαγραφή δεδομένων από τον πίνακα (delete)

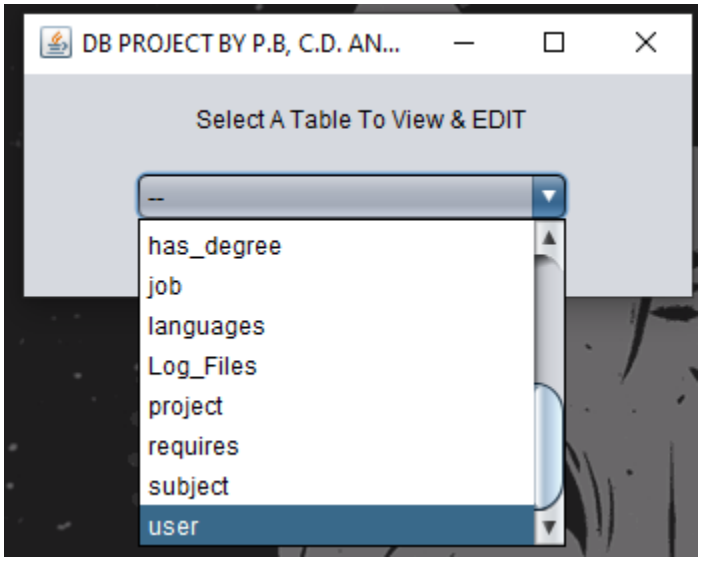
Όλα αυτά γίνονται αφού ο χρήστης επιλέξει ποιόν πίνακα θέλει να τροποποιήσει. Παρακάτω ακολουθούν τα αντίστοιχα screenshots με τη γραφική διεπαφή σε δοκιμαστικό dataset.

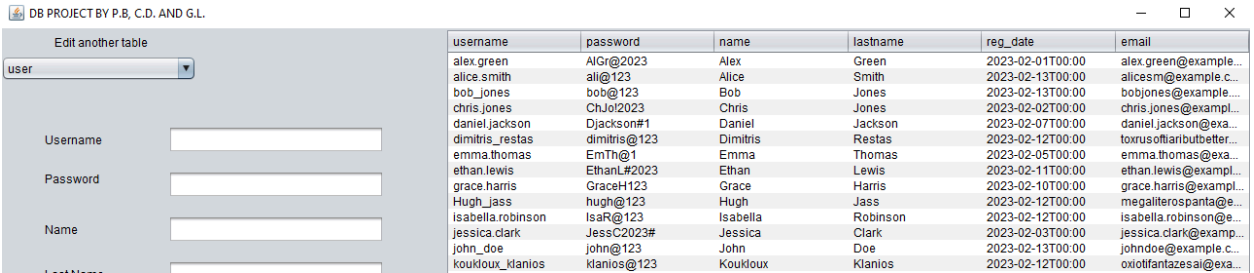
Screenshot λειτουργίας



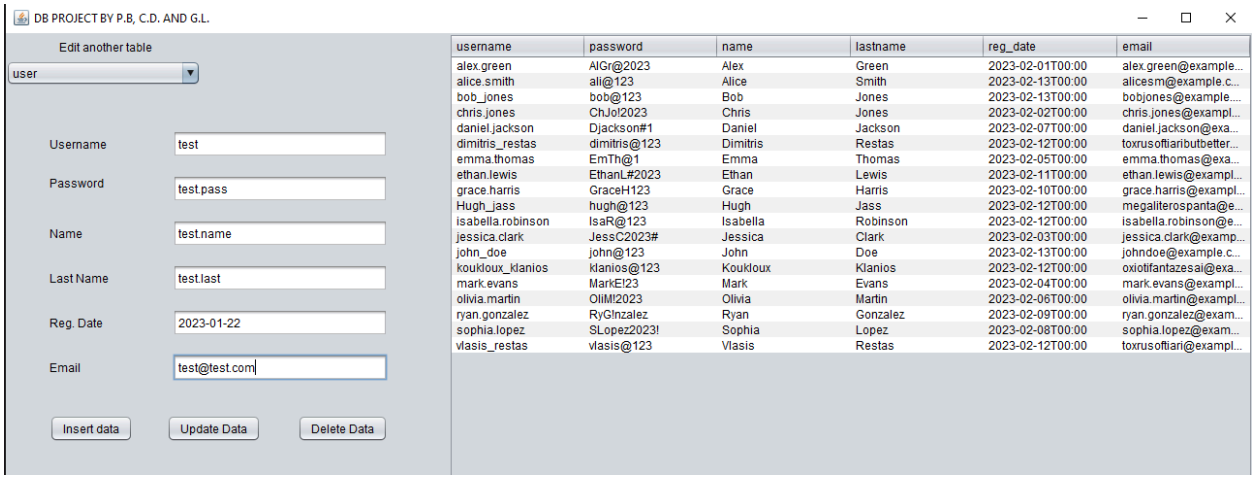
ΜΠΕΡΤΣΕΚΑΣ ΠΑΡΑΣΚΕΥΑΣ-ΣΩΤΗΡΙΟΣ (AM: 1093445)
ΔΗΜΗΤΡΑΚΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ (AM: 1079500)
ΛΑΟΥΡΕΝΤΙΟΥΣ ΙΩΑΝΝΗΣ (AM: 1093411)

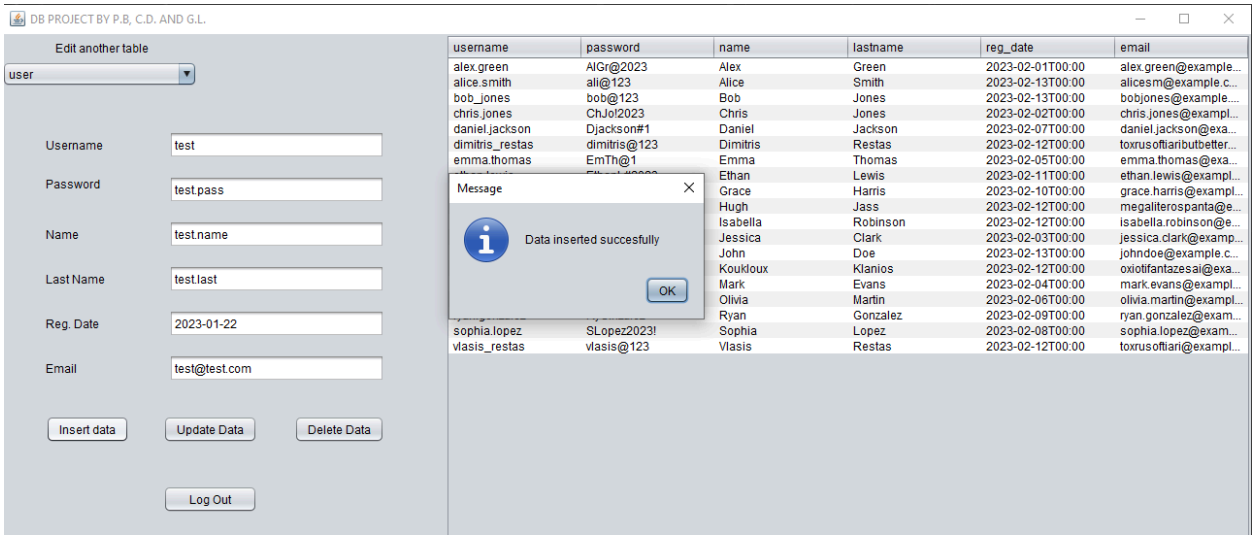


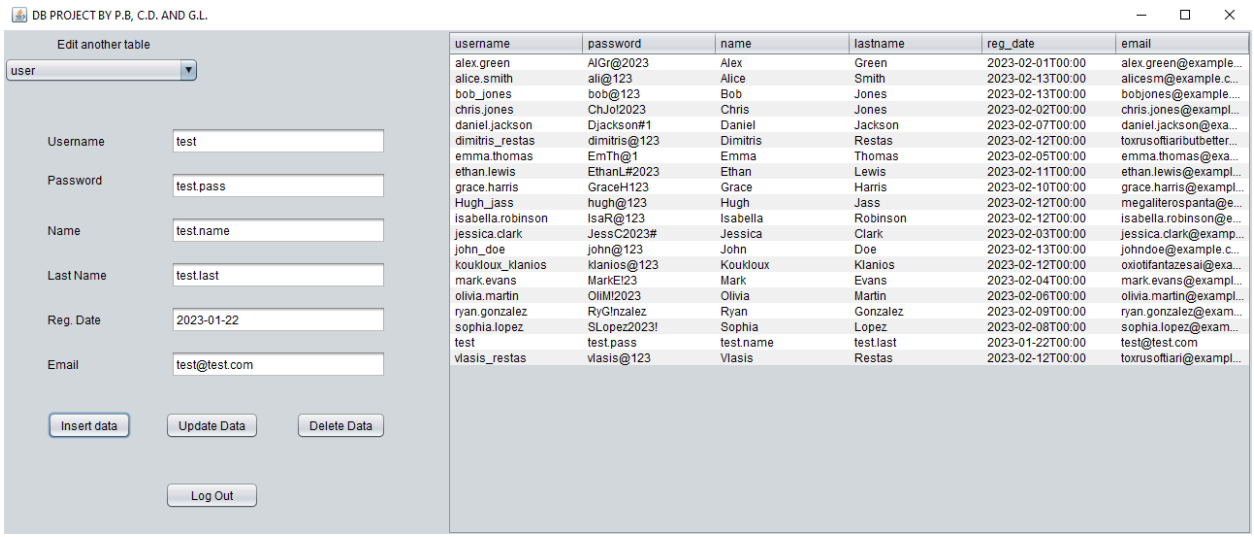


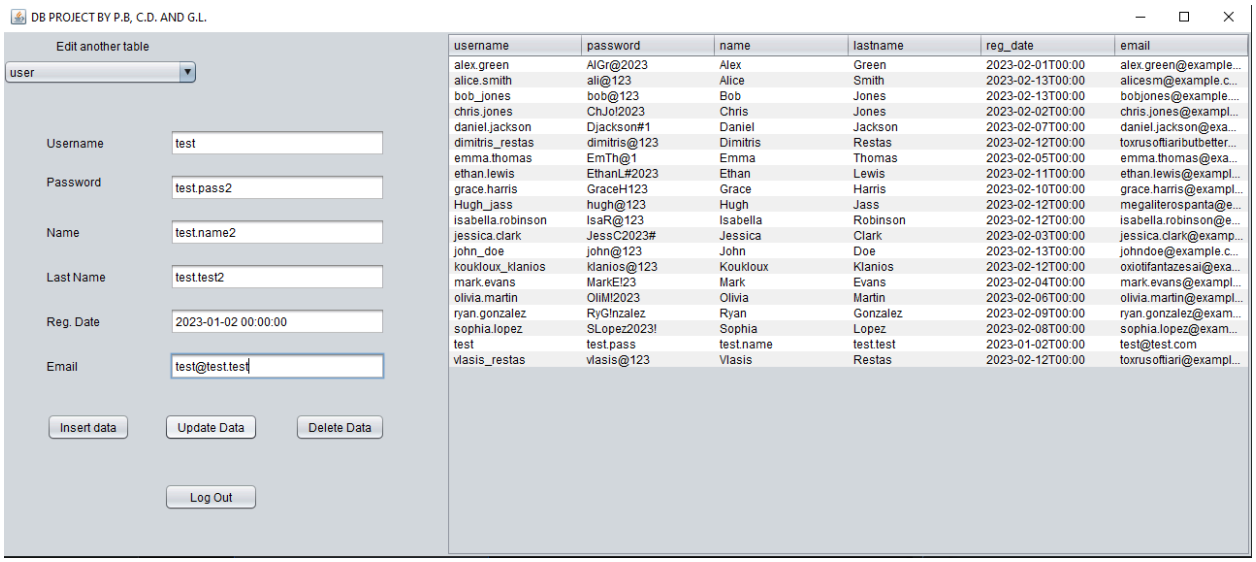


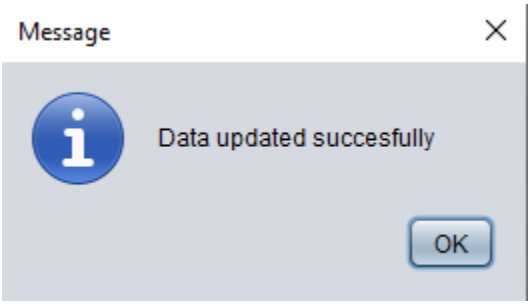
username	password	name	lastname	reg_date	email
alex.green	AlGr@2023	Alex	Green	2023-02-01T00:00	alex.green@example...
alice.smith	ali@123	Alice	Smith	2023-02-13T00:00	alicesm@example.c...
bob_jones	bob@123	Bob	Jones	2023-02-13T00:00	bobjones@example....
chris.jones	ChJo!2023	Chris	Jones	2023-02-02T00:00	chris.jones@exampl...
daniel.jackson	Djackson#1	Daniel	Jackson	2023-02-07T00:00	daniel.jackson@exa...
dimitris_restas	dimitris@123	Dimitris	Restas	2023-02-12T00:00	toxrusoftiaributbetter...
emma.thomas	EmTh@1	Emma	Thomas	2023-02-05T00:00	emma.thomas@exa...
ethan.lewis	EthanL#2023	Ethan	Lewis	2023-02-11T00:00	ethan.lewis@exampl...
grace.harris	GraceH123	Grace	Harris	2023-02-10T00:00	grace.harris@exampl...
Hugh_jass	hugh@123	Hugh	Jass	2023-02-12T00:00	megaliterospanta@e...
isabella.robinson	IsaR@123	Isabella	Robinson	2023-02-12T00:00	isabella.robinson@e...
jessica.clark	JessC2023#	Jessica	Clark	2023-02-03T00:00	jessica.clark@examp...
john_doe	john@123	John	Doe	2023-02-13T00:00	johndoe@example.c...
koukloux_klanios	klanios@123	Koukloux	Klanios	2023-02-12T00:00	oxiotifantazesai@exa...

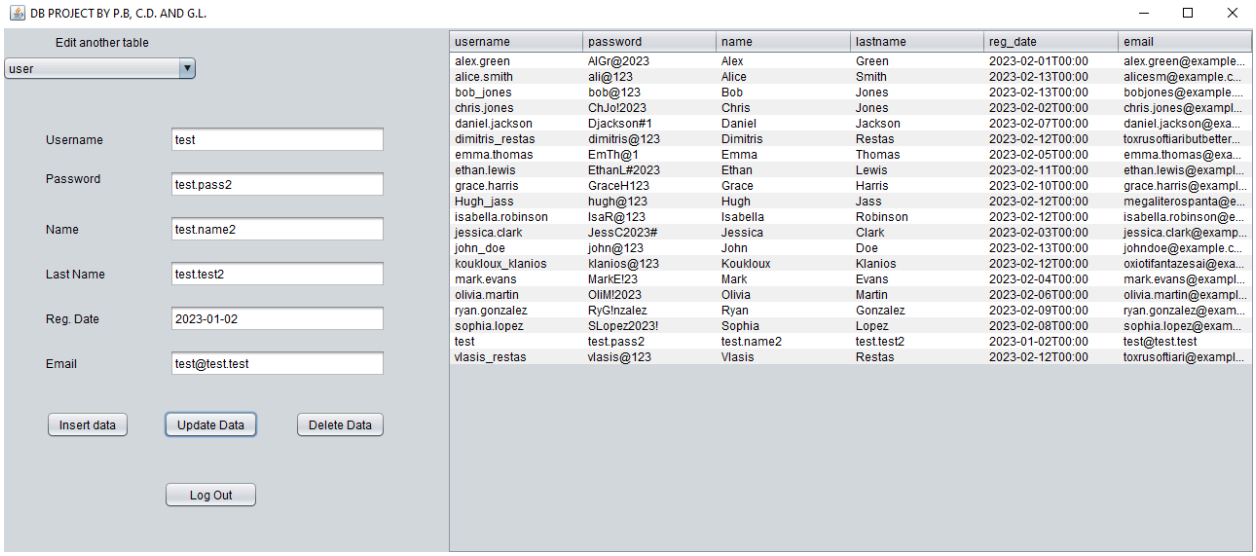

DB PROJECT BY P.B, C.D. AND G.L.
Edit another table
user
Username
test
Password
test.pass
Name
test.name
Last Name
test.last
Reg. Date
2023-01-22
Email
test@test.com
Insert data
Update Data
Delete Data
username | password | name | lastname | reg_date | email
alex.green | AlGr@2023 | Alex | Green | 2023-02-01T00:00 | alex.green@example...
alice.smith | ali@123 | Alice | Smith | 2023-02-13T00:00 | alicesm@example.c...
bob_jones | bob@123 | Bob | Jones | 2023-02-13T00:00 | bobjones@example....
chris.jones | ChJo!2023 | Chris | Jones | 2023-02-02T00:00 | chris.jones@exampl...
daniel.jackson | Djackson#1 | Daniel | Jackson | 2023-02-07T00:00 | daniel.jackson@exa...
dimitris_restas | dimitris@123 | Dimitris | Restas | 2023-02-12T00:00 | toxrusoftiaributbetter...
emma.thomas | EmTh@1 | Emma | Thomas | 2023-02-05T00:00 | emma.thomas@exa...
ethan.lewis | EthanL#2023 | Ethan | Lewis | 2023-02-11T00:00 | ethan.lewis@exampl...
grace.harris | GraceH123 | Grace | Harris | 2023-02-10T00:00 | grace.harris@exampl...
Hugh_jass | hugh@123 | Hugh | Jass | 2023-02-12T00:00 | megaliterospanta@e...
isabella.robinson | IsaR@123 | Isabella | Robinson | 2023-02-12T00:00 | isabella.robinson@e...
jessica.clark | JessC2023# | Jessica | Clark | 2023-02-03T00:00 | jessica.clark@examp...
john_doe | john@123 | John | Doe | 2023-02-13T00:00 | johndoe@example.c...
koukloux_klanios | klanios@123 | Koukloux | Klanios | 2023-02-12T00:00 | oxiotifantazesai@exa...
mark.evans | MarkE!23 | Mark | Evans | 2023-02-04T00:00 | mark.evans@exampl...
olivia.martin | OliM!2023 | Olivia | Martin | 2023-02-06T00:00 | olivia.martin@exampl...
ryan.gonzalez | RyG!nzalez | Ryan | Gonzalez | 2023-02-09T00:00 | ryan.gonzalez@exam...
sophia.lopez | SLopez2023! | Sophia | Lopez | 2023-02-08T00:00 | sophia.lopez@exam...
vlasis_restas | vlasis@123 | Vlasis | Restas | 2023-02-12T00:00 | toxrusoftiari@exampl...

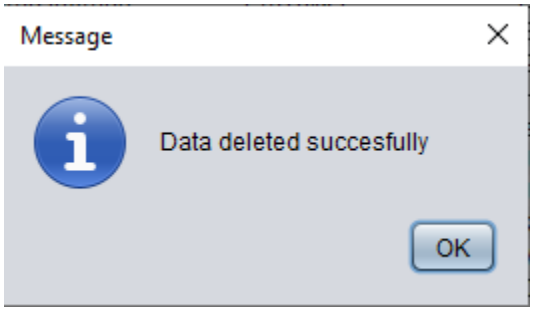

DB PROJECT BY P.B, C.D. AND G.L.
Edit another table
user
Username
test
Password
test.pass
Name
test.name
Last Name
test.last
Reg. Date
2023-01-22
Email
test@test.com
Insert data
Update Data
Delete Data
Log Out
Message
Data inserted succesfully
OK
username | password | name | lastname | reg_date | email

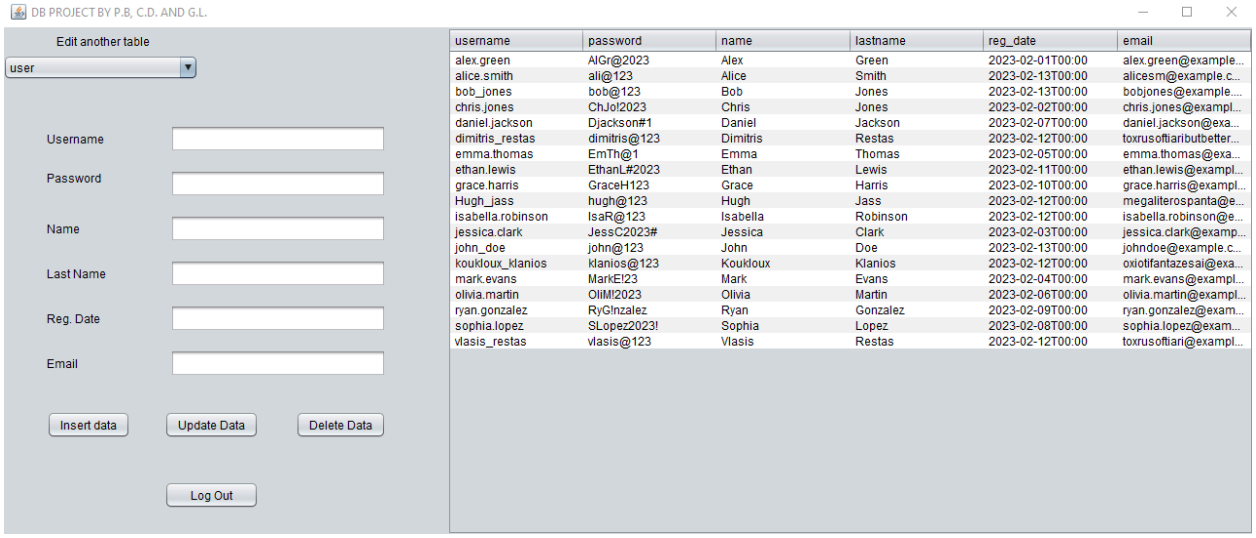

DB PROJECT BY P.B, C.D. AND G.L.
Edit another table
user
Username
test
Password
test.pass
Name
test.name
Last Name
test.last
Reg. Date
2023-01-22
Email
test@test.com
Insert data
Update Data
Delete Data
Log Out
username password name lastname reg_date email
alex.green AlGr@2023 Alex Green 2023-02-01T00:00 alex.green@example...
alice.smith ali@123 Alice Smith 2023-02-13T00:00 alicesm@example.c...
bob_jones bob@123 Bob Jones 2023-02-13T00:00 bobjones@example....
chris.jones ChJol2023 Chris Jones 2023-02-02T00:00 chris.jones@exampl...
daniel.jackson Djackson#1 Daniel Jackson 2023-02-07T00:00 daniel.jackson@exa...
dimitris_restas dimitris@123 Dimitris Restas 2023-02-12T00:00 toxrusoftiaributbetter...
emma.thomas EmTh@1 Emma Thomas 2023-02-05T00:00 emma.thomas@exa...
ethan.lewis EthanL#2023 Ethan Lewis 2023-02-11T00:00 ethan.lewis@exampl...
grace.harris GraceH123 Grace Harris 2023-02-10T00:00 grace.harris@exampl...
Hugh_jass hugh@123 Hugh Jass 2023-02-12T00:00 megaliterospanta@e...
isabella.robinson IsaR@123 Isabella Robinson 2023-02-12T00:00 isabella.robinson@e...
jessica.clark JessC2023# Jessica Clark 2023-02-03T00:00 jessica.clark@examp...
john_doe john@123 John Doe 2023-02-13T00:00 johndoe@example.c...
koukloux_klanios klanios@123 Koukloux Klanios 2023-02-12T00:00 oxiotifantazesai@exa...
mark.evans MarkEl23 Mark Evans 2023-02-04T00:00 mark.evans@exampl...
olivia.martin OliM!2023 Olivia Martin 2023-02-06T00:00 olivia.martin@exampl...
ryan.gonzalez RyGlnzalez Ryan Gonzalez 2023-02-09T00:00 ryan.gonzalez@exam...
sophia.lopez SLopez2023! Sophia Lopez 2023-02-08T00:00 sophia.lopez@exam...
test test.pass test.name test.last 2023-01-22T00:00 test@test.com
vlasis_restas vlasis@123 Vlasis Restas 2023-02-12T00:00 toxrusoftiari@exampl...


DB PROJECT BY P.B, C.D. AND G.L.
Edit another table
user
Username
test
Password
test.pass2
Name
test.name2
Last Name
test.test2
Reg. Date
2023-01-02 00:00:00
Email
test@test.test
Insert data
Update Data
Delete Data
Log Out
username password name lastname reg_date email
alex.green AlGr@2023 Alex Green 2023-02-01T00:00 alex.green@example...
alice.smith ali@123 Alice Smith 2023-02-13T00:00 alicesm@example.c...
bob_jones bob@123 Bob Jones 2023-02-13T00:00 bobjones@example....
chris.jones ChJol2023 Chris Jones 2023-02-02T00:00 chris.jones@exampl...
daniel.jackson Djackson#1 Daniel Jackson 2023-02-07T00:00 daniel.jackson@exa...
dimitris_restas dimitris@123 Dimitris Restas 2023-02-12T00:00 toxrusoftiaributbetter...
emma.thomas EmTh@1 Emma Thomas 2023-02-05T00:00 emma.thomas@exa...
ethan.lewis EthanL#2023 Ethan Lewis 2023-02-11T00:00 ethan.lewis@exampl...
grace.harris GraceH123 Grace Harris 2023-02-10T00:00 grace.harris@exampl...
Hugh_jass hugh@123 Hugh Jass 2023-02-12T00:00 megaliterospanta@e...
isabella.robinson IsaR@123 Isabella Robinson 2023-02-12T00:00 isabella.robinson@e...
jessica.clark JessC2023# Jessica Clark 2023-02-03T00:00 jessica.clark@examp...
john_doe john@123 John Doe 2023-02-13T00:00 johndoe@example.c...
koukloux_klanios klanios@123 Koukloux Klanios 2023-02-12T00:00 oxiotifantazesai@exa...
mark.evans MarkEl23 Mark Evans 2023-02-04T00:00 mark.evans@exampl...
olivia.martin OliM!2023 Olivia Martin 2023-02-06T00:00 olivia.martin@exampl...
ryan.gonzalez RyGlnzalez Ryan Gonzalez 2023-02-09T00:00 ryan.gonzalez@exam...
sophia.lopez SLopez2023! Sophia Lopez 2023-02-08T00:00 sophia.lopez@exam...
test test.pass test.name test.test 2023-01-02T00:00 test@test.com
vlasis_restas vlasis@123 Vlasis Restas 2023-02-12T00:00 toxrusoftiari@exampl...


Message
Data updated succesfully
OK



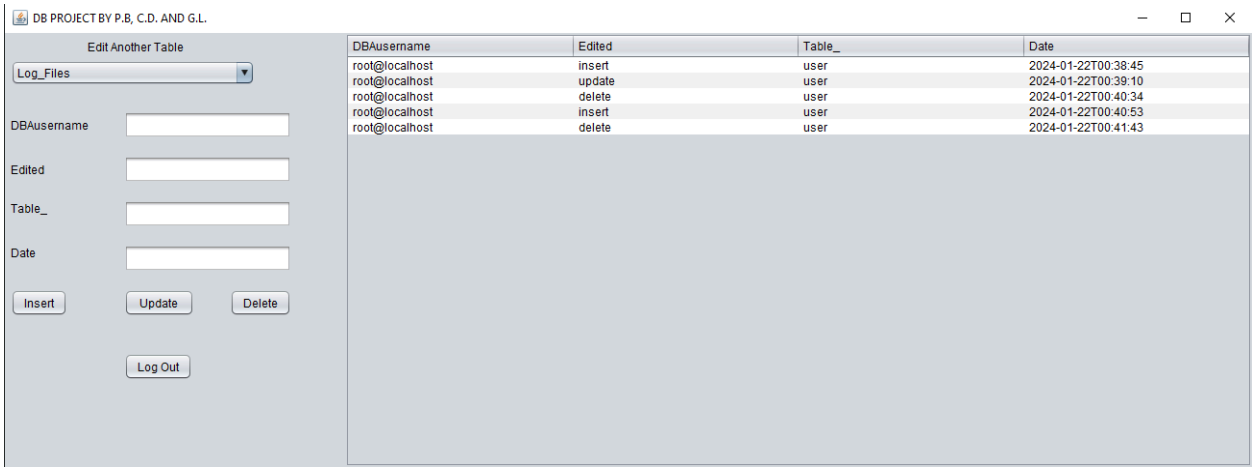




Τα παραπάνω screenshot, αποτελούν παραδείγματα insert, update και delete σε πίνακα. Η άψογη συνεργασία με τον mySQL κώδικα, φαίνεται, διότι, αν παρατηρήσουμε τον πίνακα logfiles

από το γραφικό περιβάλλον μας θα δούμε ότι όποια αλλαγη κάναμε, έχει περαστεί σε αυτό τον πίνακα με την χρήση των ζητούμενων trigger.

Screenshot λειτουργίας trigger



Σχεδιασμός κώδικα

Αρχικά, χρησιμοποιήθηκε επιπλέον το αρχείο jar mySQL connector[1] για να συνδέσουμε τη mySQL βάση δεδομένων με τη JAVA. Ο κώδικας που χρησιμοποιείται για τη σύνδεση και κλήση της κλάσης κάθε φορά παρατίθεται παρακάτω.

```
import java.sql.*;
import javax.swing.*;
public class connect {

    Connection conn= null;
    public static Connection connectDb(){
    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
    Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/project", "root",
"root");
    //JOptionPane.showMessageDialog(null,"Connected to database succesfully");
    return conn;
    }catch(Exception e){
        JOptionPane.showMessageDialog(null,e);
        return null;
    }
}
}
```

Αρχικά γίνεται εισαγωγή των πακέτων import java.sql.* και import javax.swing.* όπου το java.sql χρησιμοποιείται για τη σύνδεση της βάσης δεδομένων και το javax.swing χρησιμοποιείται για να βγάζει μηνύματα διεπαφής με το χρήστη για debugging.

Έπειτα εισάγεται η κλάση table info, όπου είναι και η πρώτη γραφική διεπαφή με τον χρήστη, εξού και το extension από την javax.swing.Jframe. Σχηματίζεται το frame και μέσα του υπάρχει ένα ComboBox, το οποίο λειτουργεί ως διακόπτης (switch) για να οδηγήσει τον χρήστη στον πίνακα που θέλει να τροποποιήσει. Επίσης εισάγεται η συνάρτηση close(), όπου, χρησιμοποιείται για να “κλείνει” το συγκεκριμένο παράθυρο, αφού ο χρήστης κάνει την επιλογή του.

```
import javax.swing.*;
public class Tableinfo extends javax.swing.JFrame{
```

[1] https://dev.mysql.com/downloads/connector/j/

```
    public Tableinfo() {
        initComponents();


    }
  public void close() {
    this.setVisible(false);
    this.dispose();
}
    private void jComboBox1ItemStateChanged(java.awt.event.ItemEvent evt) {

    }

    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
      String selectedOption = jComboBox1.getSelectedItem().toString();
      switch (selectedOption) {
          case "--":

            break;
          case "applicationshistory":
            applicationshistory appl_his = new applicationshistory();
            appl_his.setVisible(true);
            appl_his.pack();
            appl_his.setLocationRelativeTo(null);
            appl_his.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            close();
              break;

          case "applies":
            Applies appl = new Applies();
            appl.setVisible(true);
            appl.pack();
            appl.setLocationRelativeTo(null);
            appl.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
              close();
              break;
          case "dbas":
              dbas dba = new dbas();
            dba.setVisible(true);
            dba.pack();
            dba.setLocationRelativeTo(null);
            dba.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            close();
          break;

          case "degree":
              degree deg = new degree();
            deg.setVisible(true);
            deg.pack();
```

```
  deg.setLocationRelativeTo(null);
  deg.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "employee":
    employee emp = new employee();
  emp.setVisible(true);
  emp.pack();
  emp.setLocationRelativeTo(null);
  emp.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "etairia":
    etairia et = new etairia();
  et.setVisible(true);
  et.pack();
  et.setLocationRelativeTo(null);
  et.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "evaluator":
    evaluator ev = new evaluator();
  ev.setVisible(true);
  ev.pack();
  ev.setLocationRelativeTo(null);
  ev.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "has_degree":
    has_degree hd = new has_degree();
  hd.setVisible(true);
  hd.pack();
  hd.setLocationRelativeTo(null);
  hd.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "job":
    job jb = new job();
  jb.setVisible(true);
  jb.pack();
  jb.setLocationRelativeTo(null);
  jb.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "languages":
    languages lang = new languages();
  lang.setVisible(true);
  lang.pack();
  lang.setLocationRelativeTo(null);
  lang.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
            close();
        break;
        case "Log_Files":
            Logfiles lg = new Logfiles();
            lg.setVisible(true);
            lg.pack();
            lg.setLocationRelativeTo(null);
            lg.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            close();
        break;
        case "project":
            project pj = new project();
            pj.setVisible(true);
            pj.pack();
            pj.setLocationRelativeTo(null);
            pj.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            close();
        break;
        case "requires":
            requires rq = new requires();
            rq.setVisible(true);
            rq.pack();
            rq.setLocationRelativeTo(null);
            rq.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            close();
        break;
        case "subject":
            subject sb = new subject();
            sb.setVisible(true);
            sb.pack();
            sb.setLocationRelativeTo(null);
            sb.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            close();
        break;
        case "user":
            user ur = new user();
            ur.setVisible(true);
            ur.pack();
            ur.setLocationRelativeTo(null);
            ur.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            close();
        break;

        default:
            break;
    }
}
```

Φτιάχνεται μια ξεχωριστή κλάση για κάθε πίνακα, καθώς έτσι είναι πιο αποδοτική και εύκολη η επιλογή πίνακα και η διαχείρισή του. Οι κώδικες για κάθε πίνακα είναι όμοιοι και αλλάζουν ελάχιστα, ανάλογα με την δομή του κάθε πίνακα (Βλ. ER). Παρακάτω, είναι ο κώδικας για την κλάση User, (χωρίς την ανάπτυξη του Frame) που έχει να κάνει με τον πίνακα user (η ανάπτυξη έγινε με χρήση του Netbeans IDE.

```
import java.sql.*;
import javax.swing.*;
import net.proteanit.sql.DbUtils;
public class user extends javax.swing.JFrame {
Connection conn=null;
ResultSet rs =null;
PreparedStatement pst=null;
public user() {
        initComponents();
        conn=connect.connectDb();
        Update_Table();
    }
 private void Update_Table(){
        try {
                String sql = "SELECT * FROM user" ;
                pst=conn.prepareStatement(sql);
                rs=pst.executeQuery();
                  Table_show.setModel(DbUtils.resultSetToTableModel(rs));
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(null,ex);

    }
    }
 private void insert_btnActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            String sql ="INSERT INTO User (username,password,name,lastname,reg_date,email)
VALUES (?, ?, ?, ?, ?, ?)";
            pst=conn.prepareStatement(sql);
            pst.setString(1, txt_username.getText());
            pst.setString(2, txt_password.getText());
            pst.setString(3, txt_name.getText());
            pst.setString(4, txt_lastname.getText());
            pst.setString(5, txt_reg_date.getText());
            pst.setString(6, txt_email.getText());
            pst.execute();
            JOptionPane.showMessageDialog(null, "Data inserted successfully");
        }
        catch(Exception ex){
        JOptionPane.showMessageDialog(null, ex);}
        Update_Table();
    }
```

```
private void delete_btnActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        String sql ="DELETE FROM User WHERE username = ?";
        pst=conn.prepareStatement(sql);
        pst.setString(1, txt_username.getText());
        pst.execute();
        JOptionPane.showMessageDialog(null, "Data deleted successfully");
    }
    catch(Exception ex){
    JOptionPane.showMessageDialog(null, ex);}
    Update_Table();
}

private void update_btnActionPerformed(java.awt.event.ActionEvent evt) {
    try{

        String value1= txt_username.getText();
        String value2= txt_password.getText();
        String value3= txt_name.getText();
        String value4= txt_lastname.getText();
        String value5= txt_reg_date.getText();
        String value6= txt_email.getText();
        String sql ="UPDATE  User SET username='"+value1+"', password='"+value2+"',
name='"+value3+"', lastname='"+value4+"', reg_date='"+value5+"', email='"+value6+"' WHERE
username='"+value1+"' ";
        pst=conn.prepareStatement(sql);
        pst.execute();
        JOptionPane.showMessageDialog(null, "Data updated successfully");
    }
    catch(Exception ex){
    JOptionPane.showMessageDialog(null, ex);}
    Update_Table();
}
```

Χρησιμοποιείται η java.sql.* και η java.swing.* για τους ίδιους λόγους, το java.sql χρησιμοποιείται για τη σύνδεση της βάσης δεδομένων και το javax.swing χρησιμοποιείται για να βγάζει μηνύματα διεπαφής με το χρήστη για debugging. Επίσης, η net.proteanit.sql.DbUtils που προέρχεται απο το jar rs2xml βοηθά στην εισαγωγή των δεδομένων του resultset (rs) στον πίνακά μας κάθε φορα. Αυτό γίνεται με την συνάρτηση Update_Table(), όπου παίρνει όλες τις εγγραφές από τον πίνακα user, τις βάζει σε νέο πίνακα και ειδοποιεί για την διαδικασία της ενέργειας αυτής τον χρήστη.

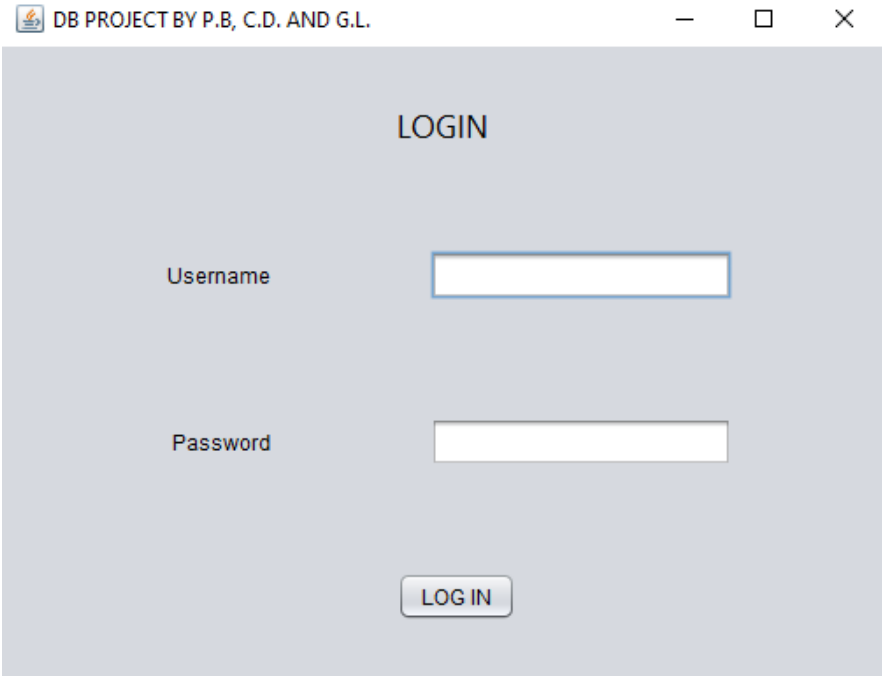
Χρησιμοποιούμε τα κουμπιά insert_btn, delete_btn και update_btn για να κάνουμε αντίστοιχα τις λειτουργίες INSERT, UPDATE και DELETE, όταν αυτά πατηθούν. Κατά το πάτημα των κουμπιών γίνεται αναγνώριση των δεδομένων που έχουμε μέσα στα textbox txt_username, txt_password, txt_lastname, txt_reg_date, txt_email (αντίστοιχα

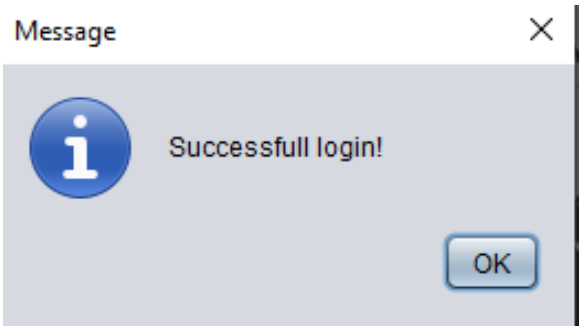
πεδία πίνακα username, password, lastname, reg_date και email) ως string, προκειμένου να κάνουμε τις κατάλληλες ενέργειες και να εκτελέσουμε τα Query μας με pst=conn.prepareStatement(sql); και pst.execute();. Ύστερα, ενημερώνεται ο χρήστης για την ενέργεια που εκτελέστηκε με το αντίστοιχο μύνημα. Σε περίπτωση αποτυχίας, επιστρέφεται το error. Μετά από κάθε ενέργεια, καλείται η Update_Table(); για να ανανεώσει τις τιμές στον πίνακα. Παρόμοιος κώδικας, χρησιμοποιείται σε όλα ρα υπόλοιπα class.

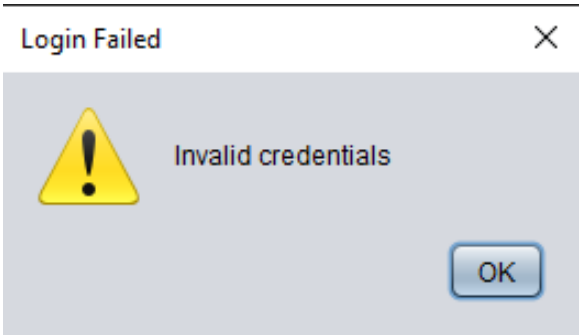
Bonus χαρακτηριστικά

Log In Interface

Συνειδητοποιώντας οτι σε ένα τέτοιο σύστημα η ασφάλεια μετράει, αποφασίσαμε στην αρχή το προγράμματος να υπάρχει μια διεπαφή εισόδου του χρήστη (“login”). Στην συγκεκριμένη επαφή, επιτρέπεται η είσοδος μόνο στους DBA που δεν έχουν end date. Σε περίπτωση σωστών στοιχείων και τήρηση των προϋποθέσεων, ο χρήστης ενημερώνεται με μήνυμα και του επιτρέπεται η είσοδος. Σε περίπτωση λάθους ή μη τήρησης των προυποθέσεων, ο χρήστης ενημερώνεται με μήνυμα και δεν του επιτρέπεται η είσοδος. Παρακάτω παρατίθεται screenshot της διεπαφής για να παρουσιαστεί η γραφική λειτουργία.







Κώδικας

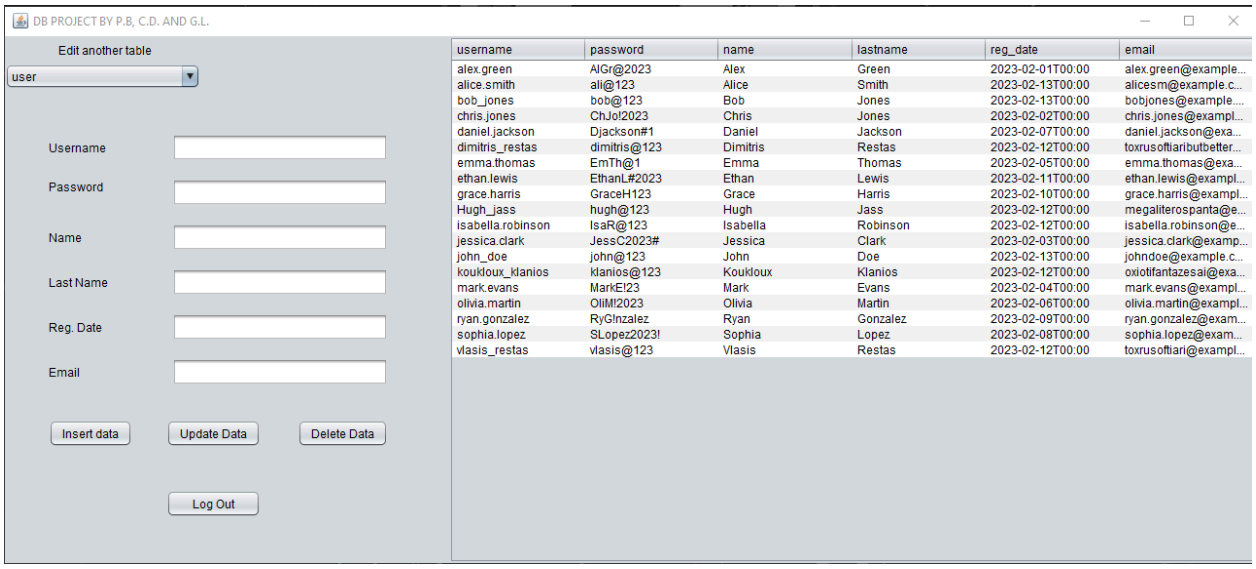
```
import java.sql.*;
import javax.swing.*;

public class Welcome_DBA extends javax.swing.JFrame {
Connection conn=null;
ResultSet rs =null;
PreparedStatement pst=null;
    /**
     * Creates new form Welcome_DBA
     */
    public Welcome_DBA() {
        initComponents();
        conn=connect.connectDb();
    }
   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
          String sql="SELECT u.username, u.password FROM user u JOIN DBAs d ON u.username = d.username
WHERE d.end_date IS NULL AND u.username = ? AND u.password = ?" ;
            try{
                pst=conn.prepareStatement(sql);
                pst.setString(1, txt_username.getText());
                pst.setString(2, new String(txt_password.getPassword()));
```

```
                rs=pst.executeQuery();
                  if(rs.next()){
                  pst.close();
                  rs.close();
                  close();
                  JOptionPane.showMessageDialog(rootPane, "Successful login!");
                  Tableinfo t = new Tableinfo();
                  t.setVisible(true);


                          }
              else {
                    JOptionPane.showMessageDialog(rootPane, "Invalid credentials", "Login Failed", 2);
                  }
              }
          catch(Exception e){
                  JOptionPane.showMessageDialog(null, e);
                  }
    }
  public void close() {
    this.setVisible(false);
    this.dispose();
  }
```

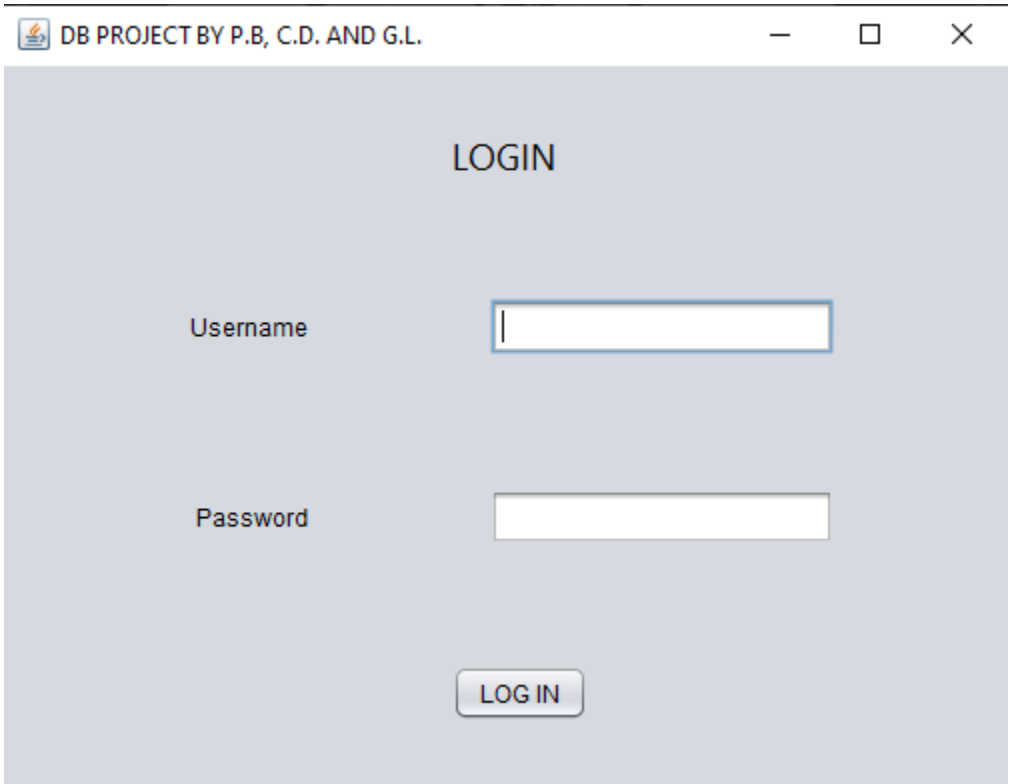
Αυτή η κλάση, όπως και οι υπόλοιπες, είναι συνδεδεμένη με τη βάση δεδομένων με το java.sql.* και βγάζει μηνύματα με το java.swing.* Ο έλεγχος που χρησιμοποιείται για τον έλεγχο των στοιχείων είναι μέσω του Query String sql="SELECT username, password FROM user WHERE username = ? IN (SELECT username = username FROM user INTERSECT SELECT username = username FROM dbas WHERE end_date IS NULL) AND password = ?" ; όπου με ? είναι με τη σειρά τα στοιχεία που εισάγονται στα textbox txt_username και txt_password . Ύστερα ενημερώνει τον χρήστη για το αποτέλεσμα και αν τα στοιχεία του είναι σωστά κλείνει το παράθυρο αυτό (με την χρήση της συνάρτησης close) και μεταβαίνει στο παράθυρο Tableinfo, Όπου είναι το μενού επιλογής του πίνακα προς επεξεργασία.

Log Out Button

Με παρόμοια λογική κατα νου, υλοποιήσαμε το log out κουμπί, οπου επιστρέφει τον χρήστη στην αρχική σελίδα μετά το πάτημά του. Παρακάτω παρατίθεται η λειτουργία του με screenshots.



username	password	name	lastname	reg_date	email
alex.green	AlGr@2023	Alex	Green	2023-02-01T00:00	alex.green@example...
alice.smith	ali@123	Alice	Smith	2023-02-13T00:00	alicesm@example.c...
bob_jones	bob@123	Bob	Jones	2023-02-13T00:00	bobjones@example....
chris.jones	ChJo!2023	Chris	Jones	2023-02-02T00:00	chris.jones@exampl...
daniel.jackson	Djackson#1	Daniel	Jackson	2023-02-07T00:00	daniel.jackson@exa...
dimitris_restas	dimitris@123	Dimitris	Restas	2023-02-12T00:00	toxrusoftiaributbetter...
emma.thomas	EmTh@1	Emma	Thomas	2023-02-05T00:00	emma.thomas@exa...
ethan.lewis	EthanL#2023	Ethan	Lewis	2023-02-11T00:00	ethan.lewis@exampl...
grace.harris	GraceH123	Grace	Harris	2023-02-10T00:00	grace.harris@exampl...
Hugh_jass	hugh@123	Hugh	Jass	2023-02-12T00:00	megaliterospanta@e...
isabella.robinson	IsaR@123	Isabella	Robinson	2023-02-12T00:00	isabella.robinson@e...
jessica.clark	JessC2023#	Jessica	Clark	2023-02-03T00:00	jessica.clark@examp...
john_doe	john@123	John	Doe	2023-02-13T00:00	johndoe@example.c...
koukloux_klanios	klanios@123	Koukloux	Klanios	2023-02-12T00:00	oxiotifantazesai@exa...
mark.evans	MarkE!23	Mark	Evans	2023-02-04T00:00	mark.evans@exampl...
olivia.martin	OliM!2023	Olivia	Martin	2023-02-06T00:00	olivia.martin@exampl...
ryan.gonzalez	RyG!nzalez	Ryan	Gonzalez	2023-02-09T00:00	ryan.gonzalez@exam...
sophia.lopez	SLopez2023!	Sophia	Lopez	2023-02-08T00:00	sophia.lopez@exam...
vlasis_restas	vlasis@123	Vlasis	Restas	2023-02-12T00:00	toxrusoftiari@exampl...



Κώδικας

Ακολουθεί ο κώδικας για το συγκεκριμένο κουμπί, που είναι ίδιος σε κάθε class πίνακα. Καλείται κάθε φορά που ο χρήστης πατήσει το κουμπί και κλείνει το παράθυρο που βρίσκεται και εμφανίζει ξανά την σελίδα εισόδου (Welcome_DBA) που πρέπει να κάνει log in για να χρησιμοποιήσει την εφαρμογή.

```
private void log_out_btnActionPerformed(java.awt.event.ActionEvent evt) {
    Welcome_DBA wd = new Welcome_DBA();
        wd.setVisible(true);
        wd.pack();
        wd.setLocationRelativeTo(null);
        wd.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        close();
  }
```

COMBOBOX με δυνατότητα μετάβασης σε άλλο πίνακα σε κάθε σελίδα

Τέλος, όμοια με την πρώτη σελίδα, με την προσθήκη αυτού του Combobox και την ενσωμάτωσή του ως Switch με δυνατότητα μετάβασης σε όλους τους πίνακες από οποιαδήποτε άλλη σελίδα (αρκεί ο χρήστης να έχει κάνει log in) πετυχαίνουμε καλύτερη χρηστικότητα γενικά και πιο ομαλές και εύκολες μεταβάσεις από πίνακα σε πίνακα. Χρησιμοποιείται μαζί με την συνάρτηση Close για το κάθε παράθυρο και αναγνωρίζει την επιλογή του χρήστη, χρησιμοποιώντας String Recognition.

Κώδικας

```
 private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
String selectedOption = jComboBox1.getSelectedItem().toString();
    switch (selectedOption) {
        case "user":
          break;
        case "applicationshistory":
          applicationshistory appl_his = new applicationshistory();
          appl_his.setVisible(true);
          appl_his.pack();
          appl_his.setLocationRelativeTo(null);
```

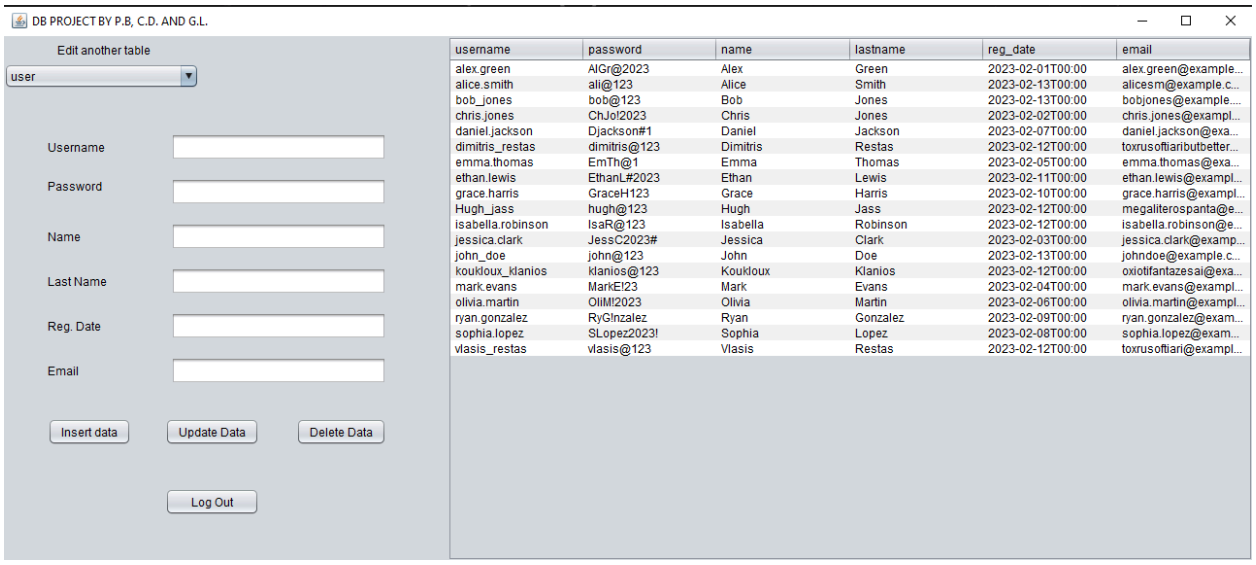
```
  appl_his.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
    break;

case "applies":
  Applies appl = new Applies();
  appl.setVisible(true);
  appl.pack();
  appl.setLocationRelativeTo(null);
  appl.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    close();
    break;
case "dbas":
    dbas dba = new dbas();
  dba.setVisible(true);
  dba.pack();
  dba.setLocationRelativeTo(null);
  dba.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;

case "degree":
    degree deg = new degree();
  deg.setVisible(true);
  deg.pack();
  deg.setLocationRelativeTo(null);
  deg.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "employee":
    employee emp = new employee();
  emp.setVisible(true);
  emp.pack();
  emp.setLocationRelativeTo(null);
  emp.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "etairia":
    etairia et = new etairia();
  et.setVisible(true);
  et.pack();
  et.setLocationRelativeTo(null);
  et.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "evaluator":
    evaluator ev = new evaluator();
  ev.setVisible(true);
  ev.pack();
  ev.setLocationRelativeTo(null);
```

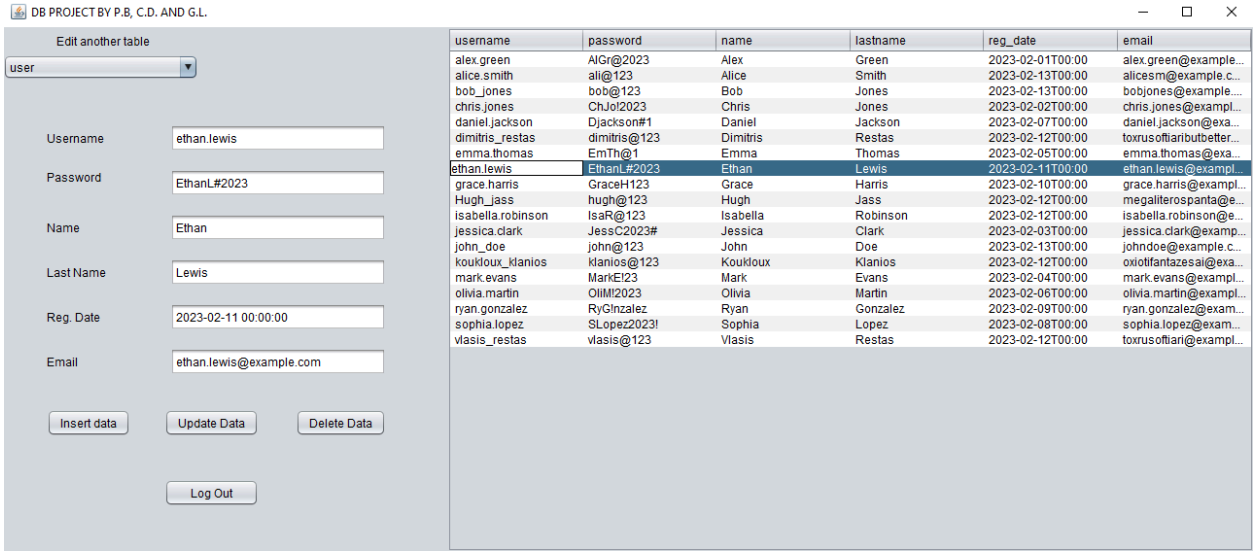
```
  ev.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "has_degree":
    has_degree hd = new has_degree();
  hd.setVisible(true);
  hd.pack();
  hd.setLocationRelativeTo(null);
  hd.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "job":
    job jb = new job();
  jb.setVisible(true);
  jb.pack();
  jb.setLocationRelativeTo(null);
  jb.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "languages":
    languages lang = new languages();
  lang.setVisible(true);
  lang.pack();
  lang.setLocationRelativeTo(null);
  lang.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "Log_Files":
    Logfiles lg = new Logfiles();
  lg.setVisible(true);
  lg.pack();
  lg.setLocationRelativeTo(null);
  lg.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "project":
    project pj = new project();
  pj.setVisible(true);
  pj.pack();
  pj.setLocationRelativeTo(null);
  pj.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
break;
case "requires":
    requires rq = new requires();
  rq.setVisible(true);
  rq.pack();
  rq.setLocationRelativeTo(null);
  rq.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  close();
```

```
            break;
            case "subject":
                subject sb = new subject();
              sb.setVisible(true);
              sb.pack();
              sb.setLocationRelativeTo(null);
              sb.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
              close();
            break;
            default:
                break;
        }
    }
```

Click-To-Select σειρές του πίνακα και άμεσο “πέρασμα” τιμών στα textbox

Ένα ακόμα χαρακτηριστικό που αποφασίσαμε ότι θα ήταν καλό να υπάρχει για την διευκόλυνση του χρήστη ως προς την επιλογή δεδομένων είναι το παραπάνω. Ο χρήστης πλέον μπορεί πολύ απλά να επιλέγει με ένα κλικ του ποντικιού την γραμμή του πίνακα που θέλει να αλλάξει και τα στοιχεία των κελιών αυτής της γραμμής περνάνε αυτόματα στα textboxes για ενημέρωση. Έτσι, διευκολύνεται η εμπειρία του χρήστη, και αποφεύγονται τυχόν λάθη. Παρακάτω ακολουθεί ο κώδικας για το συγκεκριμένο χαρακτηριστικό, και screenshot με την λειτουργία του. Χρησιμοποιείται String Recognition για την αναγνώριση της σειράς που θέλει να επιλέξει ο χρήστης και τα στοιχεία μπαίνουν αυτοματα στα κελιά με χρήση SQL Query και βηματικής συνάρτησης (add).





Κώδικας

```
private void Table_showMouseClicked(java.awt.event.MouseEvent evt) {
        try{
            int row = Table_show.getSelectedRow();
            String Table_click=(Table_show.getModel().getValueAt(row,0).toString());
            String sql = "select * from user where username='"+Table_click+"'";
            pst= conn.prepareStatement(sql);
            rs=pst.executeQuery();
            if(rs.next()){
                String add1 =rs.getString("username");
                txt_username.setText(add1);
                String add2 =rs.getString("password");
                txt_password.setText(add2);
                String add3 =rs.getString("name");
                txt_name.setText(add3);
                String add4 =rs.getString("lastname");
                txt_lastname.setText(add4);
                String add5 =rs.getString("reg_date");
                txt_reg_date.setText(add5);
                String add6 =rs.getString("email");
                txt_email.setText(add6);

            }
        }
        catch(Exception ex){
            JOptionPane.showMessageDialog(null, ex);
        }
    }
```