ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ 2023-2024

2024

ΣΥΝΤΑΚΤΕΣ

Βαμβακίδης Αλέξανδρος 1093329

Ζουμπουλάκης Ιωάννης 1093365

Μάγαλου Κωνσταντίνα 1093419

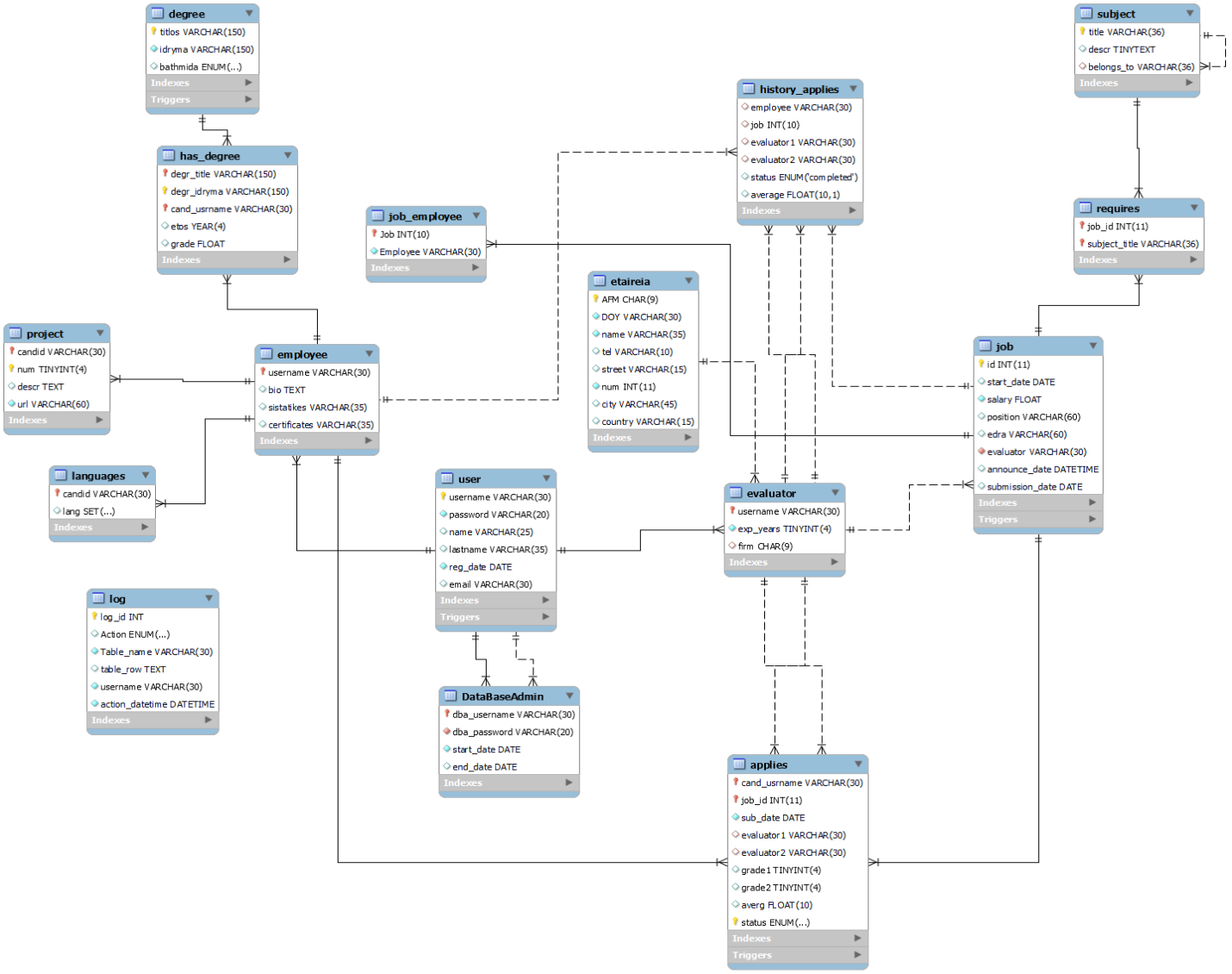
22/1/2024

ΠΕΡΙΕΧΟΜΕΝΑ

ΜΕΡΟΣ Α: ΣΧΕΔΙΑΣΜΟΣ ΒΔ & SQL
ΚΕΦΑΛΑΙΟ 1:

Στο κεφάλαιο αυτό, στόχος είναι η παρουσίαση του σχεσιακού διαγράμματος της συνολικής αναθεωρημένης Βάσης Δεδομένων (ΒΔ). Παρακάτω, παρουσιάζονται οι τροποποιήσεις οι οποίες πραγματοποιήθηκαν και οι νέοι πίνακες που δημιουργήθηκαν, καθώς και τα σχόλια για κάθε σημαντική αλλαγή.

ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ ΑΝΑΘΕΩΡΗΜΕΝΗΣ ΒΔ:



ΠΡΟΠΑΡΑΣΚΕΥΑΣΤΙΚΗ ΦΑΣΗ:

Προς τις απαιτήσεις του προπαρασκευαστικού μέρους του project, η ομάδα μας προχώρησε στη δημιουργία των πινάκων της Βάσης Δεδομένων, λαμβάνοντας υπόψη το σχεσιακό διάγραμμα που παρείχε η εκφώνηση. Κάθε πίνακας δημιουργήθηκε σύμφωνα με τις προδιαγραφές που περιγράφονταν σε αυτό, εξασφαλίζοντας έτσι τη συμβατότητα της βάσης δεδομένων με τις απαιτήσεις του project.

Για τη συμπλήρωση των πινάκων με κατάλληλα δεδομένα, η ομάδα προχώρησε στην εισαγωγή του κατάλληλου αριθμού πληροφοριών σε κάθε πεδίο, προσαρμόζοντας τα δεδομένα στις αναφερθείσες ανάγκες και προδιαγραφές.

ΝΕΕΣ ΑΠΑΙΤΗΣΕΙΣ 3.1.2:

ΑΠΑΙΤΗΣΗ 1:

Για την πραγματοποίηση της συγκεκριμένης απαίτησης, εκτελέστηκαν επεκτάσεις στη δομή του πίνακα **applies** με σκοπό την ολοκληρωτική κάλυψη των αναγκών του έργου. Οι αλλαγές που πραγματοποιήθηκαν είναι οι εξής:

- Δημιουργία πεδίου **status**:

Προστέθηκε το πεδίο status με σκοπό την καταγραφή της τρέχουσας κατάστασης της αίτησης. Το πεδίο αυτό λαμβάνει τις τιμές 'active', 'completed' ή 'canceled', προσφέροντας έτσι πληροφορίες σχετικά με την εξέλιξη της αίτησης.

- Δημιουργία πεδίου **sub_date**:

Το πεδίο sub_date προστέθηκε με σκοπό την αποθήκευση της ημερομηνίας υποβολής της αίτησης. Αυτή η προσθήκη επιτρέπει την ακριβέστερη καταγραφή του χρόνου υποβολής, παρέχοντας σημαντική πληροφορία για την αίτηση.

Η ολοκληρωτική εκπλήρωση των απαιτήσεων προβλέπεται με τη δημιουργία του σχετικού trigger (3.1.4.2), το οποίο αναλύεται λεπτομερέστερα στη συνέχεια.

❖ **ΣΗΜΕΙΩΣΗ:**

Στην πραγματικότητα, η τροποποίηση της δομής κάθε πίνακα σε μια βάση δεδομένων συνήθως πραγματοποιείται με τη χρήση της εντολής **ALTER**. Ωστόσο, επιλέξαμε την άμεση τροποποίηση του αρχικού πίνακα για λόγους ευανάγνωστης παρουσίασης του κώδικα. Αν και η χρήση της εντολής ALTER είναι η συνηθισμένη πρακτική για τέτοιες αλλαγές, η προσέγγισή μας επιδιώκει την απλότητα και την εύκολη κατανόηση του κώδικα.

ΑΠΑΙΤΗΣΗ 2:

Για την υλοποίηση της εν λόγω απαίτησης, προβήκαμε σε περαιτέρω επεκτάσεις της δομής του πίνακα **applies**. Οι αλλαγές οι οποίες πραγματοποιήθηκαν είναι οι ακόλουθες:

- Δημιουργία των πεδίων **evaluator1** & **evaluator2**:

Δημιουργήθηκαν δύο νέα πεδία, τα οποία αντιπροσωπεύουν τους δύο αξιολογητές που είναι υπεύθυνοι για τη βαθμολόγηση της αίτησης κάθε εργαζομένου.

- Δημιουργία των πεδίων **grade1** & **grade2**:

Δημιουργήθηκαν δύο νέα πεδία για την αποθήκευση των βαθμολογιών που καταχωρούν οι δύο αξιολογητές για την αίτηση του κάθε εργαζομένου.

- Δημιουργία του πεδίου **averg**:

Δημιουργήθηκε ένα νέο πεδίο με την ονομασία averg, το οποίο αφορά τον μέσο όρο των δύο βαθμολογιών που λαμβάνει η αίτηση του εργαζομένου από τους αξιολογητές.

Παράλληλα με αυτή την τροποποίηση, η ομάδα οδηγήθηκε στη δημιουργία του πίνακα job_employee. Πιο αναλυτικά:

❖ **Πίνακας job_employee**:

Παρατίθεται ο κώδικας του προκείμενου πίνακα:

```
Job int(10) not null,
Employee varchar(30) not null default 'unknown',
primary key (Job),
constraint JB
foreign key (Job) references job(id)
on update cascade on delete cascade
);
```

Η δημιουργία του πίνακα αυτού, προέκυψε από την ανάγκη καταγραφής της σχέσης εργασιών και υπαλλήλων. Πιο αναλυτικά:

- Η στήλη Job αναπαριστά τον αριθμό ταυτοποίησης της εργασίας και είναι καθοριστική ως πρωτεύον κλειδί.
- Η στήλη Employee καθορίζει τον υπάλληλο που σχετίζεται με την εργασία.
- Υπάρχει ο περιορισμός JB όπου η στήλη Job αναφέρεται στο πεδίο id του πίνακα **job**. Σε περίπτωση ενημέρωσης ή διαγραφής στον πίνακα job, οι αλλαγές αντικατοπτρίζονται και στον πίνακα job_employee.

Για την αξιολόγηση των αιτήσεων, οι βαθμολογίες εισήχθησαν από εμάς μέσω της εντολής insert. Η ολοκληρωτική εκπλήρωση των απαιτήσεων προβλέπεται με τη δημιουργία του σχετικού procedure (3.1.3.1) , το οποίο αναλύεται λεπτομερέστερα στη συνέχεια.

ΑΠΑΙΤΗΣΗ 3:

Για την πραγματοποίηση της συγκεκριμένης απαίτησης, απαραίτητη ήταν η δημιουργία ενός *νέου πίνακα* τον οποίο ονομάσαμε **history_applies.** Πιο συγκεκριμένα:

❖ **Πίνακας history_applies:**

Παρατίθεται ο κώδικας του προκείμενου πίνακα:

```
CREATE TABLE history_applies (
   employee VARCHAR(30),
   job INT(10),
   evaluator1 VARCHAR(30),
   evaluator2 VARCHAR(30),
   status ENUM('completed') DEFAULT 'completed',
   average FLOAT(10,1),
   CONSTRAINT EVALGR1 FOREIGN KEY (evaluator1) REFERENCES evaluator(username) ON DELETE NO ACTION ON UPDATE CASCADE,
   CONSTRAINT EVALGR2 FOREIGN KEY (evaluator2) REFERENCES evaluator(username) ON DELETE NO ACTION ON UPDATE CASCADE,
   CONSTRAINT empl FOREIGN KEY (employee) REFERENCES employee(username) ON DELETE NO ACTION ON UPDATE CASCADE,
   CONSTRAINT Jobmpl FOREIGN KEY (job) REFERENCES job(id) ON DELETE NO ACTION ON UPDATE CASCADE
);
```

Ο σχεδιασμός του συγκεκριμένου πίνακα προέκυψε από την δεύτερη και τρίτη απαίτηση. Πιο συγκεκριμένα, ο πίνακας αυτός αξιοποιήθηκε για την αποθήκευση των ολοκληρωμένων αιτήσεων. Έτσι, με το πέρας της επεξεργασίας και την απόφανση του αποτελέσματος που προκύπτει από τους δύο αξιολογητές (**evaluator**) και αφορά τον εργαζόμενο (**employee**), η αίτηση μεταφέρεται στον πίνακα ιστορικού. Για το λόγο αυτό, στο σχεδιασμό του πίνακα αυτού χρησιμοποιήθηκαν οι περιορισμοί (CONSTRAINT) και τα εξωτερικά κλειδιά τα οποία επιδεικνύουν τη άμεση σύνδεση αυτών των πινάκων (history_applies, evaluator, employee, job). Πιο αναλυτικά:

- Οι στήλες *employee* και *evaluator* αντιπροσωπεύουν τον υπάλληλο και τον αριθμό ταυτοποίησης της θέσης εργασίας, αντίστοιχα.
- Οι στήλες *evaluator1* και *evaluator2* αντιπροσωπεύουν τους αξιολογητές οι οποίοι βαθμολογούν τον εκάστοτε εργαζόμενο.
- Ο πίνακας περιλαμβάνει περιορισμούς (*CONSTRAINTS*) για εξασφάλιση σωστών συσχετίσεων με άλλους πίνακες (evaluator, employee, job).
- Οι στήλες *status* και *average* έχουν προκαθορισμένες τιμές και αντιπροσωπεύουν την κατάσταση της αίτησης και τον μέσο όρο αξιολόγησης, αντίστοιχα.

Για την **εισαγωγή δεδομένων** στον πίνακα, απαιτήθηκε να εισαχθούν σε αυτόν 60.000 εγγραφές. Συνολικά ο κώδικας δημιουργεί μια προσομοίωση 60.000 ολοκληρωμένων αιτήσεων με τυχαίες τιμές βαθμολογίας, λαμβάνοντας υπόψη τα υπάρχοντα δεδομένα στους πίνακες ‘employee’, ‘job’και ‘evaluator’. Πιο αναλυτικά:

I.Προέλευση Δεδομένων:

Οι τιμές προέρχονται από τους πίνακες employee, job, και evaluator χρησιμοποιώντας συντονισμό (CROSS JOIN). Αυτό σημαίνει ότι κάθε στοιχείο από κάθε πίνακα συνδυάζεται με κάθε άλλο, προκειμένου να παραχθούν όλοι οι δυνατοί συνδυασμοί.

II.Τυχαιότητα Στοιχείων:

Οι τυχαίες τιμές επιλέγονται χρησιμοποιώντας τη συνάρτηση RAND(). Η τιμή RAND() * 20 παράγει τυχαίους αριθμούς από 0 έως 20.
Η συνάρτηση ROUND(RAND() * 20, 1) χρησιμοποιείται για να περιορίσει τον βαθμό αξιολόγησης σε ένα δεκαδικό ψηφίο.

III. *Τελική Εισαγωγή:*

Η εντολή INSERT INTO ... SELECT ... χρησιμοποιείται για να εισάγει τα δεδομένα που επιλέγονται από τον συντονισμένο πίνακα στον πίνακα history_applies.
Οι εγγραφές που εισάγονται είναι τυχαίες λόγω του ORDER BY RAND().

IV. *Περιορισμός Εγγραφών:*

Χρησιμοποιείται η δήλωση LIMIT 60000 για να περιοριστεί ο αριθμός των εγγραφών που εισάγονται στις 60.000.

Ο κώδικας που χρησιμοποιήθηκε για την εισαγωγή εγγραφών στον πίνακα είναι ο εξής:

```
INSERT INTO history_applies (employee, job, evaluator1, evaluator2, status, average)
SELECT
   e.username AS employee,
   j.id AS job,
   ev1.username AS evaluator1,
   ev2.username AS evaluator2,
   'completed' AS status,
ROUND(RAND() * 20, 1) AS average  -- Χρήση της συνάρτησης ROUND για δύο
     δεκαδικά ψηφία
FROM employee e, job j, evaluator ev1,evaluator ev2
ORDER BY  RAND() LIMIT 60000;
```

ΑΠΑΙΤΗΣΗ 4:

Για την περάτωση της προκείμενης απαίτησης δημιουργήθηκαν ακόμη δύο πίνακες στη Βάση Δεδομένων, ο **DataBaseAdmin** και ο **log**. Η ανάλυση των πινάκων αυτών παρουσιάζεται εκτενώς παρακάτω:

- **Πίνακας DataBaseAdmin:**

Παρατίθεται ο κώδικας του προκείμενου πίνακα:

```
CREATE TABLE DataBaseAdmin(
dba_username varchar(30) not null,
dba_password varchar(20) not null,
start_date DATE not null,
end_date DATE null,
primary key (dba_username),
constraint dbausr
foreign key (dba_username) references user(username)
on update cascade on delete cascade,
constraint dbapass
foreign key (dba_password) references user(password)
on update cascade on delete cascade
);
```

Στην τέταρτη απαίτηση, χρειάστηκε η δημιουργία του συγκεκριμένου πίνακα ο οποίος αναπαριστά τους Διαχειριστές της Βάσης Δεδομένων, μιας νέας κατηγορίας users. Πιο αναλυτικά:

- Η στήλη dba_username είναι ορισμένη ως πρωτεύον κλειδί και δεν μπορεί να παραμείνει κενή (not null).
- Το πεδίο start_date αντιπροσωπεύει την ημερομηνία έναρξης, ενώ το πεδίο end_date αντιπροσωπεύει την ημερομηνία λήξης, είναι προαιρετικό (μπορεί να παραμείνει κενό).
- Υπάρχει εξωτερική σχέση με τον πίνακα user, όπου η στήλη dba_username αναφέρεται στο πεδίο username του πίνακα **user** και το όμοιο ισχύει για το dba_password - password (του πίνακα user). Σε περίπτωση ενημέρωσης ή διαγραφής στον πίνακα user, οι αλλαγές αντικατοπτρίζονται και στον πίνακα DataBaseAdmin.

Ο κώδικας που χρησιμοποιήθηκε για την εισαγωγή εγγραφών στον πίνακα είναι ο εξής:

```
INSERT INTO DataBaseAdmin
VALUES ('alex','123','2020-09-14',null),
                ('zoubou','456','2019-06-09',null);
```

❖ **Πίνακας log:**

Παρατίθεται ο κώδικας του προκείμενου πίνακα:

```
CREATE TABLE log(
log_id INT AUTO_INCREMENT,
Action enum ('Insert',  'Update', 'Delete'),
Table_name VARCHAR(30) NOT NULL ,
table_row text,
username VARCHAR(30) NOT NULL,
action_datetime DATETIME NOT NULL,
PRIMARY KEY (log_id)
);
```

Ο συγκεκριμένος πίνακας, δημιουργήθηκε για να καταγράψει τις ενέργειες των Διαχειριστών Βάσης Δεδομένων. Συγκεκριμένα, ενημερώνεται σε κάθε αλλαγή ενέργειας (insert, update, delete) σε κάποιον από τους εξής πίνακες: **job, user, degree**. Αποτελεί ισχυρό εργαλείο για τον έλεγχο και την ανίχνευση αλλαγών στη Βάση Δεδομένων, επιτρέποντας την ανάκτηση πληροφοριών σχετικά με την ιστορία των ενεργειών που πραγματοποιήθηκαν. Πιο αναλυτικά:

- log_id: Πρωτεύον κλειδί που είναι αυξανόμενος αριθμός, παρέχοντας μοναδική αναγνώριση για κάθε καταγραφή.
- Action: Το πεδίο καθορίζει το είδος της δραστηριότητας που καταγράφεται, είτε πρόκειται για εισαγωγή, ενημέρωση ή διαγραφή.
- Table_name: Κρατάει το όνομα του πίνακα στον οποίο αναφέρεται η ενέργεια. Αυτό είναι χρήσιμο για να κατανοηθεί ποιο τμήμα της βάσης δεδομένων επηρεάστηκε.
- table_row: Αυτό το πεδίο καταγράφει τα δεδομένα που άλλαξαν κατά τη διάρκεια της ενέργειας. Αναφέρεται στις πληροφορίες που επηρεάζονται από την εισαγωγή, ενημέρωση ή διαγραφή. Η χρήση του τύπου TEXT επιτρέπει την αποθήκευση μεγάλων όγκων δεδομένων.
- username: Αναφέρεται στο όνομα χρήστη που πραγματοποίησε τη δραστηριότητα. Αυτό επιτρέπει τον εντοπισμό του χρήστη που πραγματοποίησε την αλλαγή.
- action_datetime: Κρατάει τη χρονική στιγμή εκτέλεσης της δραστηριότητας. Αυτό παρέχει πληροφορίες σχετικά με το πότε πραγματοποιήθηκε η ενέργεια.

Με αφορμή όλα τα παραπάνω, γίνεται σαφές ότι στον πίνακα αυτό δεν υπάρχει ανάγκη παρέμβασης για την εισαγωγή σχετικών δεδομένων. Η καταγραφή των δραστηριοτήτων πραγματοποιείται **δυναμικά** από το σύστημα, παρέχοντας ουσιαστικές πληροφορίες για τις αλλαγές στη βάση δεδομένων, οι οποίες συνδέονται άρρηκτα με τους πίνακες job, user, degree.

ΚΕΦΑΛΑΙΟ 2:

ΔΗΜΙΟΥΡΓΙΑ STORED PROCEDURE (3.1.3):

STORED PROCEDURE 1:

Παρατίθεται ο κώδικας της προκείμενης stored procedure:

```
CREATE PROCEDURE CHECKEVALUATIONGRADE(
   IN EVALUATOR_USERNAME VARCHAR(30),
   IN EMPLOYEE_USERNAME VARCHAR(30),
   IN JOB_ID INT,
   OUT GRADE INT
)
BEGIN
   DECLARE GRADE_COUNT INT;
   DECLARE EVAL1 VARCHAR(255);
   DECLARE EVAL2 VARCHAR(255);
   DECLARE GRAD1 INT;
   DECLARE GRAD2 INT;

   SELECT COUNT(*), EVALUATOR1, EVALUATOR2, GRADE1, GRADE2
     INTO GRADE_COUNT, EVAL1, EVAL2, GRAD1, GRAD2
     FROM APPLIES A
     WHERE CAND_USRNAME = EMPLOYEE_USERNAME
     AND A.JOB_ID = JOB_ID
     AND (EVALUATOR1 = EVALUATOR_USERNAME OR EVALUATOR2 = EVALUATOR_USERNAME)
     GROUP BY CAND_USRNAME, JOB_ID, EVALUATOR1, EVALUATOR2, GRADE1, GRADE2;


   IF GRADE_COUNT IS NULL  THEN
      SET GRADE = 0;
   ELSE
     CASE
     WHEN EVAL1 = EVALUATOR_USERNAME AND GRAD1 IS NOT NULL THEN
               SET GRADE = GRAD1;
     WHEN EVAL1 = EVALUATOR_USERNAME AND GRAD1 IS NULL THEN
               SET GRAD1 = 0;

     SET GRAD1 = GRAD1 + IFNULL((SELECT COUNT(*)
FROM HAS_DEGREE HD WHERE HD.CAND_USRNAME = EMPLOYEE_USERNAME AND HD.DEGR_TITLE
= 'BSC'), 0) * 1 + IFNULL((SELECT COUNT(*)
FROM HAS_DEGREE HD
WHERE HD.CAND_USRNAME = EMPLOYEE_USERNAME AND HD.DEGR_TITLE = 'MSC'), 0) * 2
     + IFNULL((SELECT COUNT(*) FROM HAS_DEGREE HD
WHERE HD.CAND_USRNAME = EMPLOYEE_USERNAME AND HD.DEGR_TITLE = 'PHD'), 0) * 3
```

```
                                                    + IFNULL((SELECT COUNT(*) FROM
LANGUAGES L WHERE L.CANDID = EMPLOYEE_USERNAME), 0) * 1
                                                    + IFNULL((SELECT COUNT(*) FROM PROJECT
P WHERE P.CANDID = EMPLOYEE_USERNAME), 0) * 1;


UPDATE APPLIES
        SET GRADE1 = GRAD1
WHERE CAND_USRNAME = EMPLOYEE_USERNAME AND APPLIES.JOB_ID = JOB_ID;
         SET GRADE = GRAD1;
WHEN EVAL2 = EVALUATOR_USERNAME AND GRAD2 IS NOT NULL THEN
          SET GRADE = GRAD2;
WHEN EVAL2 = EVALUATOR_USERNAME AND GRAD2 IS NULL THEN
           SET GRAD2 = 0;

SET GRAD2 = GRAD2 + IFNULL((SELECT COUNT(*) FROM HAS_DEGREE HD WHERE
HD.CAND_USRNAME = EMPLOYEE_USERNAME AND HD.DEGR_TITLE = 'BSC'), 0) * 1
                                                    + IFNULL((SELECT COUNT(*) FROM
HAS_DEGREE HD WHERE HD.CAND_USRNAME = EMPLOYEE_USERNAME AND HD.DEGR_TITLE =
'MSC'), 0) * 2
                                                    + IFNULL((SELECT COUNT(*) FROM
HAS_DEGREE HD WHERE HD.CAND_USRNAME = EMPLOYEE_USERNAME AND HD.DEGR_TITLE =
'PHD'), 0) * 3
                                                    + IFNULL((SELECT COUNT(*) FROM
LANGUAGES L WHERE L.CANDID = EMPLOYEE_USERNAME), 0) * 1
                                                    + IFNULL((SELECT COUNT(*) FROM PROJECT
P WHERE P.CANDID = EMPLOYEE_USERNAME), 0) * 1;
                    SET GRADE = GRAD2;

UPDATE APPLIES
SET GRADE2 = GRAD2
WHERE CAND_USRNAME = EMPLOYEE_USERNAME AND APPLIES.JOB_ID = JOB_ID;

              END CASE;
      END IF;

END //
```

Σκοπός της συγκεκριμένης stored procedure είναι ο έλεγχος και υπολογισμός των βαθμών αξιολόγησης για έναν συγκεκριμένο αξιολογητή (evaluator), έναν εργαζόμενο (employee) και μια συγκεκριμένη θέση εργασίας (job). Η λογική προορίζεται να διαχειριστεί περιπτώσεις όπου υπάρχουν διάφοροι αξιολογητές για μια αίτηση εργασίας. Αναλυτικότερα, σχετικά με τον κώδικα:

- *Δήλωση Παραμέτρων:*
  Η αποθηκευμένη διαδικασία δέχεται τρεις εισόδους (IN): το όνομα του αξιολογητή (evaluator_username), το όνομα του εργαζομένου (employee_username), και το αναγνωριστικό της θέσης εργασίας (job_id). Επίσης, έχει μια έξοδο (OUT) παράμετρο με το όνομα grade τύπου INT.
- *Μεταβλητές:*
  Δηλώνονται τέσσερις μεταβλητές (grade_count, eval1, eval2, grade), που χρησιμοποιούνται για την αποθήκευση αποτελεσμάτων ερωτημάτων.

- *Ερώτημα SELECT:*
  Χρησιμοποιείται ένα ερώτημα SELECT για τον έλεγχο του αριθμού των εγγραφών που πληρούν συγκεκριμένες συνθήκες. Τα αποτελέσματα αποθηκεύονται στις αντίστοιχες μεταβλητές.
- *Δομή CASE:*
  Έχουμε μια δομή CASE που ελέγχει τον αξιολογητή (evaluator_username) και τους βαθμούς (grade1, grade2) που έχει δώσει. Ανάλογα με τον αξιολογητή, υπολογίζονται και ενημερώνονται οι βαθμοί.
- *Ενημέρωση Δεδομένων:*
  Χρησιμοποιούνται εντολές UPDATE για να ενημερώσουν τα πεδία grade1 ή grade2 του πίνακα applies με τους υπολογισμένους βαθμούς.

Εκτελώντας την διαδικασία **CheckEvaluationGrade**, τα αποτελέσματα που εκτυπώνονται στην οθόνη είναι τα εξής:

STORED PROCEDURE 2:

Παρατίθεται ο κώδικας της προκείμενης stored procedure:

```
CREATE PROCEDURE EditApplies(
   IN p_employee_username VARCHAR(30),
   IN p_job_id INT,
   IN p_sub_date DATE,
   IN p_action CHAR(1)
)
BEGIN
   DECLARE evaluator1_username VARCHAR(30);
   DECLARE evaluator2_username VARCHAR(30);
   DECLARE existing_applications INT;
   DECLARE p_status enum('active','canceled');
   DECLARE eval_firm int(10);

   SELECT COUNT(*)
   INTO existing_applications
   FROM applies
   WHERE cand_usrname = p_employee_username AND job_id = p_job_id;
```

```
    SELECT status INTO p_status
    FROM applies
    WHERE job_id = p_job_id AND cand_usrname = p_employee_username;

    SELECT evaluator1, evaluator2 INTO evaluator1_username, evaluator2_username
    FROM applies
    WHERE job_id = p_job_id AND cand_usrname = p_employee_username;

    IF p_action = 'i' THEN
        IF existing_applications > 0 and p_status = 'active' THEN
            IF evaluator1_username IS NULL  THEN
              SET eval_firm = (SELECT firm FROM evaluator where username = evaluator2_username);
               SET evaluator1_username = (SELECT username FROM evaluator  WHERE firm = eval_firm AND
     username != evaluator2_username ORDER BY RAND()  LIMIT 1);
               UPDATE applies
               SET evaluator1 = evaluator1_username
               WHERE cand_usrname = p_employee_username AND job_id = p_job_id;
              SET eval_firm = (SELECT firm FROM evaluator where username = evaluator2_username);
                        ELSEIF evaluator2_username IS NULL  THEN
              SET eval_firm = (SELECT firm FROM evaluator where username = evaluator1_username);
               SET evaluator2_username = (SELECT username FROM evaluator  WHERE firm = eval_firm AND
                 username != evaluator1_username ORDER BY RAND() LIMIT 1);
               UPDATE applies
                SET evaluator2 = evaluator2_username
                WHERE cand_usrname = p_employee_username AND job_id = p_job_id;
                          ELSEIF evaluator1_username IS NULL AND evaluator2_username IS NULL THEN
                SET evaluator1_username = (SELECT username FROM evaluator  ORDER BY RAND()  LIMIT 1);
               SET eval_firm = (SELECT firm FROM evaluator where username = evaluator1_username);
                evaluator2_username = (SELECT username FROM evaluator  WHERE firm = eval_firm  AND
              username != evaluator1_username ORDER BY RAND()  LIMIT 1);
               UPDATE applies
               SET evaluator1 = evaluator1_username,evaluator2 = evaluator2_username
               WHERE cand_usrname = p_employee_username AND job_id = p_job_id;
                        ELSE
               SIGNAL SQLSTATE '45000'
               SET MESSAGE_TEXT = 'Application already exists and all evaluator positions are covered';

                        END IF;
                        ELSE
               SET evaluator1_username = (SELECT username FROM evaluator ORDER BY RAND()  LIMIT 1);
              SET eval_firm = (SELECT firm FROM evaluator where username = evaluator1_username);
               SET evaluator2_username = (SELECT username FROM evaluator WHERE firm = eval_firm AND
                    username != evaluator1_username ORDER BY RAND()  LIMIT 1);
INSERT INTO applies (cand_usrname, job_id, sub_date, evaluator1, evaluator2)
    VALUES (p_employee_username, p_job_id, p_sub_date, evaluator1_username, evaluator2_username);
                        END IF;

    ELSEIF p_action = 'c' THEN
       IF existing_applications > 0 THEN
          IF p_status = 'canceled' THEN
             SIGNAL SQLSTATE '45000'
             SET MESSAGE_TEXT = 'Already canceled application';
          ELSE
             UPDATE applies
             SET status = 'canceled', sub_date = p_sub_date
```

```
            WHERE cand_usrname = p_employee_username AND job_id = p_job_id;

        END IF;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Application does not exist';
    END IF;

ELSEIF p_action = 'a' THEN
    IF (SELECT COUNT(*) FROM applies WHERE cand_usrname = p_employee_username AND job_id =
p_job_id AND status = 'canceled') > 0 THEN
        UPDATE applies
        SET status = 'active'
        WHERE cand_usrname = p_employee_username AND job_id = p_job_id;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT =  'No canceled application found for the given employee and job.';
    END IF;

ELSE
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Invalid action. Please provide \'i\', \'c\', or \'a\' as the action.';
END IF;
END //
```

Η διαδικασία **EditApplies** αποτελεί μια διαδικασία επεξεργασίας αιτήσεων για θέσεις εργασίας. Ο χαρακτήρας της εισόδου σε αυτήν, καθορίζει την πορεία και την εξέλιξη της αίτησης, ενώ παρέχει λεπτομερείς διαχειριστικές λειτουργίες για διάφορα σενάρια.
Η διαδικασία αυτή προβλέπει διάφορες ενέργειες ανάλογα με τον εισαγόμενο χαρακτήρα (p_action). Συγκεκριμένα, διαχειρίζεται τη διαδικασία εισαγωγής νέας αίτησης με την τυχαία επιλογή αξιολογητών, επαναχρησιμοποιεί υπάρχοντες αξιολογητές για υπάρχουσες αιτήσεις (χαρακτήρας 'i'), ακυρώνει υπάρχουσες αιτήσεις (χαρακτήρας 'c'), και ενεργοποιεί προηγουμένως ακυρωμένες αιτήσεις (χαρακτήρας 'a').
Μάλιστα, εξασφαλίζεται η ακεραιότητα των δεδομένων με την εφαρμογή αποτελεσματικών ελέγχων σφαλμάτων, όπως η αποτροπή εισαγωγής υπάρχουσας αίτησης και η αποτροπή ακύρωσης ή ενεργοποίησης μιας ήδη ακυρωμένης αίτησης. Η συγκεκριμένη διαδικασία είναι καίρια για την αποτελεσματική διαχείριση των αιτήσεων εργασίας στο πλαίσιο του συστήματος, εξασφαλίζοντας ομαλή λειτουργία και ακρίβεια στη διαχείριση των αιτήσεων εκ μέρους του προσωπικού διαχείρισης. Παρακάτω αναλύονται τα κύρια σημεία του κώδικα:

➢ *Παράμετροι εισόδου (IN):*

- p_employee_username: Το όνομα χρήστη του εργαζομένου.
- p_job_id: Το αναγνωριστικό id της θέσης εργασίας.
- p_sub_date: Η ημερομηνία υποβολής της αίτησης.
- p_action: Η ενέργεια που θα πραγματοποιηθεί ('i' για εισαγωγή, 'c' για ακύρωση, 'a' για ενεργοποίηση).

➢ *Δηλώσεις Μεταβλητών:*
- evaluator1_username και evaluator2_username: Τα ονόματα χρηστών των δύο αξιολογητών.
- existing_applications: Ο αριθμός των υπαρκτών αιτήσεων για τη συγκεκριμένη θέση εργασίας.
- p_status: Η κατάσταση της αίτησης ('active' ή 'canceled').
- eval_firm: Η εταιρεία στην οποία ανήκει ο αξιολογητής.

➢ *Έλεγχος Ύπαρξης Αίτησης:*
- Έλεγχος για το αν η αίτηση υπάρχει ήδη με βάση τον αριθμό των υπαρκτών αιτήσεων.

➢ *Καθορισμός Κατάστασης Αίτησης:*
- Καθορισμός της κατάστασης της αίτησης ('active' ή 'canceled').

➢ *Επιλογή Αξιολογητών:*
- Επιλογή των δύο αξιολογητών για τη θέση εργασίας από την ίδια εταιρεία.
- Αν η ενέργεια είναι εισαγωγή, γίνεται τυχαία επιλογή αξιολογητών, ενώ αν η αίτηση υπάρχει ήδη, επαναχρησιμοποιούνται οι υπάρχοντες αξιολογητές.

➢ *Ενέργειες Ανάλογα με την Ενέργεια (p_action):*
- Εισαγωγή νέας αίτησης με τους νέους αξιολογητές.
- Ακύρωση υπάρχουσας αίτησης.
- Ενεργοποίηση ακυρωμένης αίτησης.

Εκτελώντας τη διαδικασία **EditApplies**, τα αποτελέσματα που εμφανίζονται στην οθόνη είναι τα εξής:

```
1034    select * from applies;
1035
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

cand_usrname	job_id	sub_date	evaluator1	evaluator2	grade1	grade2	averg	status
amarildolou	6606	2020-10-01	raptor	NULL	2	11	0	active
annitapania	4404	2019-09-25	konmag	nana	4	8	0	active
bardos	222938743	2017-05-09	flylo	gianniszoub	15	13	0	active
choojuan78	9876543	2022-04-20	konmag	nana	11	2	0	active
chrisboo	3303	2022-10-02	maryjane12	miguelES	16	7	0	active
ElMull12	2345678	2022-06-10	raptor	alexvamv	18	9	0	active
ElMull12	9876543	2022-05-01	venetiaben	BraveSpirit	13	16	0	active
georgepap98	123456789	2022-03-15	maryjane12	miguelES	7	15	0	active
helenant	1101	2020-08-10	johnD78	Son	12	10	0	active

```
1036 •   call EditApplies('georgepap98','123456789', '2012-03-15', 'c');
1037 •   select * from applies;
1038
1039
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conte

cand_usrname	job_id	sub_date	evaluator1	evaluator2	grade1	grade2	averg	status
amarildolou	6606	2020-10-01	raptor	NULL	2	11	0	active
annitapania	4404	2019-09-25	konmag	nana	4	8	0	active
bardos	222938743	2017-05-09	flylo	gianniszoub	15	13	0	active
choojuan78	9876543	2022-04-20	konmag	nana	11	2	0	active
chrisboo	3303	2022-10-02	maryjane12	miguelES	16	7	0	active
ElMull12	2345678	2022-06-10	raptor	alexvamv	18	9	0	active
ElMull12	9876543	2022-05-01	venetiaben	BraveSpirit	13	16	0	active
georgepap98	123456789	2012-03-15	maryjane12	miguelES	7	15	0	canceled
helenant	1101	2020-08-10	johnD78	Son	12	10	0	active

```
1040 •   call EditApplies('georgepap98','123456789', '2012-03-15', 'a');
1041 •   select * from applies;
1042
1043
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

cand_usrname	job_id	sub_date	evaluator1	evaluator2	grade1	grade2	averg	status
amarildolou	6606	2020-10-01	raptor	NULL	2	11	0	active
annitapania	4404	2019-09-25	konmag	nana	4	8	0	active
bardos	222938743	2017-05-09	flylo	gianniszoub	15	13	0	active
choojuan78	9876543	2022-04-20	konmag	nana	11	2	0	active
chrisboo	3303	2022-10-02	maryjane12	miguelES	16	7	0	active
ElMull12	2345678	2022-06-10	raptor	alexvamv	18	9	0	active
ElMull12	9876543	2022-05-01	venetiaben	BraveSpirit	13	16	0	active
georgepap98	123456789	2012-03-15	maryjane12	miguelES	7	15	0	active
helenant	1101	2020-08-10	johnD78	Son	12	10	0	active

```
1044 •   call EditApplies('georgepap98','1101', '2012-03-15', 'i');
1045 •   select * from applies;
1046
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

cand_usrname	job_id	sub_date	evaluator1	evaluator2	grade1	grade2	averg	status
amarildolou	6606	2020-10-01	raptor	NULL	2	11	0	active
annitapania	4404	2019-09-25	konmag	nana	4	8	0	active
bardos	222938743	2017-05-09	flylo	gianniszoub	15	13	0	active
choojuan78	9876543	2022-04-20	konmag	nana	11	2	0	active
chrisboo	3303	2022-10-02	maryjane12	miguelES	16	7	0	active
ElMull12	2345678	2022-06-10	raptor	alexvamv	18	9	0	active
ElMull12	9876543	2022-05-01	venetiaben	BraveSpirit	13	16	0	active
georgepap98	1101	2012-03-15	Son	bomber	NULL	NULL	0	active
georgepap98	123456789	2012-03-15	maryjane12	miguelES	7	15	0	active

STORED PROCEDURE 3:

Παρατίθεται ο κώδικας της προκείμενης stored procedure:

```
CREATE PROCEDURE apply_evaluation_and_results(IN job int(10))
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE cand_usrnm VARCHAR(30);
   DECLARE eval1 VARCHAR(30);
   DECLARE eval2 VARCHAR(30);
   DECLARE grd1 INT(10);
   DECLARE grd2 INT(10);
   DECLARE submission_date DATE;
   DECLARE avg_grade float;

   DECLARE winning_employee VARCHAR(30);
   DECLARE winning_eval1 VARCHAR(30);
   DECLARE winning_eval2 VARCHAR(30);
   DECLARE winning_avg_grade float;
   DECLARE winner_sub_date DATE;

   DECLARE cur_promotions CURSOR FOR
    SELECT
         cand_usrname,
         grade1,
          grade2,
         evaluator1,
         evaluator2,
         sub_date
       FROM
         applies
       WHERE status = 'active' and job_id = job;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
OPEN cur_promotions;
promotion_loop: LOOP
FETCH cur_promotions INTO cand_usrnm,grd1,grd2,eval1,eval2,submission_date;

IF done THEN
          LEAVE promotion_loop;
END IF;

call CheckEvaluationGrade(eval1,cand_usrnm,job,grd1);
call CheckEvaluationGrade(eval2,cand_usrnm,job,grd2);

SET avg_grade = (grd1 + grd2) / 2;

IF (avg_grade > (SELECT MAX(averg) FROM applies WHERE job_id = job )) OR (avg_grade = (SELECT
MAX(averg) FROM applies WHERE job_id = job ) ) AND submission_date < winner_sub_date THEN
UPDATE job_employee je
SET Employee = cand_usrnm
WHERE je.Job = job;

IF winning_employee IS NOT NULL THEN
   UPDATE history_applies
```

```
            SET average = 0
            WHERE history_applies.job = job AND history_applies.employee = winning_employee;
        END IF;

    SET winning_employee = cand_usrnm;
        SET winning_eval1 = eval1;
        SET winning_eval2 = eval2;
        SET winning_avg_grade = avg_grade;
        SET winner_sub_date = submission_date;


        INSERT INTO history_applies (employee, job, evaluator1, evaluator2,status, average)
            VALUES (winning_employee, job, winning_eval1, winning_eval2,'completed', winning_avg_grade);

          ELSE
            INSERT INTO history_applies
                VALUES (cand_usrnm, job,eval1, eval2,'completed', 0);
        END IF;


        UPDATE applies
        SET averg = avg_grade
        WHERE cand_usrname = cand_usrnm AND job_id = job;

        END LOOP;

        CLOSE cur_promotions;
        DELETE FROM applies WHERE job_id = job;

        END //
```

Η διαδικασία **apply_evaluation_and_results** υλοποιεί έναν μηχανισμό αξιολόγησης και καταγραφής αποτελεσμάτων για τις αιτήσεις εργασίας σε συγκεκριμένες θέσεις. Συνολικά, διαχειρίζεται τις αιτήσεις, υποβάλλοντάς τις για αξιολόγηση, υπολογίζοντας το μέσο όρο των βαθμών και καταγράφοντας τα αποτελέσματα στους αντίστοιχους πίνακες. Παρακάτω αναλύονται τα κύρια σημεία του κώδικα:

- *Δήλωση Μεταβλητών:*
  - done: Μεταβλητή που χρησιμοποιείται για τον έλεγχο του cursor loop.
  - Πολλαπλές μεταβλητές για την αποθήκευση δεδομένων της αίτησης, όπως cand_usrnm, grd1, grd2, eval1, eval2, submission_date, avg_grade, winning_employee, winning_eval1, winning_eval2, winning_avg_grade, winner_sub_date.

- *Δήλωση Cursor:*
  - Ένας cursor (cur_promotions) που επιστρέφει τα στοιχεία αίτησης για εργασία με συγκεκριμένη κατάσταση ('active') και θέση εργασίας (job).

- *Δήλωση CONTINUE HANDLER:*
  - Ένας handler που ρυθμίζει τη μεταβλητή done όταν δεν υπάρχουν άλλες εγγραφές για επεξεργασία.

- *Άνοιγμα του Cursor:*
  - Ανοίγει τον cursor για επεξεργασία των αιτήσεων εργασίας.

- *Επαναληπτική Διαδικασία:*
  - Ένα loop (promotion_loop) που επεξεργάζεται κάθε αίτηση εργασίας από τον cursor.

- *Εκτέλεση Διαδικασιών:*
  - Επικαλείται τη διαδικασία CheckEvaluationGrade για τον υπολογισμό της βαθμολογίας (grade) και τους ελέγχους ανάθεσης αξιολογητών.
  - Υπολογίζει το μέσο όρο (avg_grade) των βαθμών.

- *Ενημέρωση Πίνακα job_employee:*
  - Εάν ο υπολογισμένος μέσος όρος είναι υψηλότερος από τους υφιστάμενους, τότε ενημερώνει τον πίνακα job_employee με το νέο υποψήφιο.

- *Ενημέρωση Πίνακα history_applies:*
  - Εάν υπάρχει νικητής, ενημερώνει το ιστορικό αιτήσεων και ανακηρύσσει νέο νικητή.

- *Ενημέρωση Πίνακα applies:*
  - Ενημερώνει τον πίνακα applies με τον υπολογισμένο μέσο όρο.

- *Κλείσιμο Cursor:*
  - Κλείνει τον cursor.

- *Διαγραφή από τον Πίνακα applies:*
  - Διαγράφει τις εγγραφές από τον πίνακα applies που αφορούν τη συγκεκριμένη θέση εργασίας.

Εκτελώντας τη διαδικασία **apply_evaluation_and_results**, τα αποτελέσματα που εμφανίζονται στην οθόνη είναι τα εξής:

```
1045    call apply_evaluation_and_results('5505');
1046 •  select * from history_applies;
```

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

employee	job	evaluator1	evaluator2	status	average
NatNowak	8808	albysmth	raptor	completed	12.0
amarildolou	4404	venetiaben	BraveSpirit	completed	18.1
moneymaker	1234567	nana	Gab4unc4	completed	9.1
JSmith09	876543210	Son	maryjane12	completed	14.7
georgepap98	876543210	gianniszoub	johnD78	completed	19.9
NatNowak	222938743	miguelES	albysmth	completed	12.3
kalos	2202	konmag	maryjane12	completed	0.2
StephanosN9	382693468	flylo	miguelES	completed	0.7
simos	1101	albysmth	johnD78	completed	2.4

history_applies 9 ×

STORED PROCEDURE 4:

Παρατίθεται ο κώδικας της προκείμενης stored procedure:

```
-- a)
CREATE INDEX idx_average ON history_applies(average)//

CREATE PROCEDURE SearchByGradeRange(IN start_grade FLOAT, IN end_grade FLOAT)
BEGIN
   SELECT employee, job
   FROM history_applies
   WHERE average BETWEEN start_grade AND end_grade;
END //

-- b)
CREATE INDEX idx2_average ON history_applies(evaluator1, evaluator2)//

CREATE PROCEDURE SearchByEvaluator(IN evaluator VARCHAR(255))
BEGIN
   SELECT employee, job
   FROM history_applies
   WHERE evaluator=evaluator1 OR evaluator=evaluator2;
END //
```

Απαραίτητο σε αυτή την procedure κρίθηκε η χρήση της εντολής CREATE INDEX. Η ανάγη αυτή, προέκυψε από το πολύ μεγάλο όγκο εγγραφών (60.000) στον πίνακα history_applies με σκοπό την επιτάχυνση της αναζήτησης.

a)
Αρχικά, έγινε η δημιουργία του δείκτη (idx_average) στον πίνακα history_applies βάσει της στήλης average. Αυτός ο δείκτης βοηθά στη γρήγορη αναζήτηση εγγραφών με βάση την τιμή του πεδίου average. Όταν εκτελούνται ερωτήματα που έχουν συνθήκες όπου εμπλέκεται το πεδίο average, ο δείκτης μπορεί να επιταχύνει τη διαδικασία της αναζήτησης.
Η διαδικασία **SearchByGradeRange** δέχεται δύο παραμέτρου και επιστρέφει τις στήλες employee και job από τον πίνακα history_applies για τις εγγραφές που έχουν το πεδίο average εντός του εύρους που καθορίζεται από τις παραμέτρους start_grade και end_grade.

b)
Αρχικά, δημιουργήθηκε ένας δείκτης (idx2_average) στον πίνακα history_applies βάσει των στηλών evaluator1 και evaluator2 με σκοπό την γρήγορη αναζήτηση εγγραφών με βάση τους αξιολογητές (evaluator1 και evaluator2).
Η διαδικασία **SearchByEvaluator** δέχεται μία παράμετρο evaluator και επιστρέφει τις στήλες employee και job από τον πίνακα history_applies για τις εγγραφές όπου το evaluator ταιριάζει με τα πεδία evaluator1 ή evaluator2.

Εκτελώντας τη διαδικασία **SearchByGradeRange**, τα αποτελέσματα που εμφανίζονται στην οθόνη είναι τα εξής:

```
1048    call SearchByGradeRange('5', '8');
```

Result Grid | Filter Rows: | Export:

employee	job
mairalag	203749373
moneymaker	203749373
JSmith09	5505
NatNowak	876543210
georgepap98	87654321
choojuan78	8808
georgepap98	7707
choojuan78	222938743
simos	87654321
StephanosN9	6606
choojuan78	5505

Result 12

Εκτελώντας τη διαδικασία **SearchByEvaluator**, τα αποτελέσματα που εμφανίζονται στην οθόνη είναι τα εξής:

```
1048    call SearchByEvaluator('konmag');
```

Result Grid | Filter Rows: | Export:

employee	job
georgepap98	1234567
ElMull12	203749373
simos	8808
choojuan78	876543210
JSmith09	2345678
helenant	4404
JSmith09	8808
leonardodi	3303
amarildolou	382693468
StephanosN9	382693468
megalos	251382255
StephanosN9	2345678

Result 13

ΚΕΦΑΛΑΙΟ 3:

ΔΗΜΙΟΥΡΓΙΑ TRIGGER (3.1.4)

TRIGGER 1:

Παρατίθεται ο κώδικας του προκείμενου trigger:

```
/* Trigger για τον πίνακα job */
CREATE TRIGGER jobINSERT_logTrigger
AFTER INSERT ON job
FOR EACH ROW
BEGIN
   INSERT INTO log (action,table_name ,table_row,username,action_datetime)
       VALUES ('Insert','Job' ,new.id,user(),now());
END //

CREATE TRIGGER jobUPDATE_logTrigger
AFTER UPDATE ON job
FOR EACH ROW
BEGIN
   INSERT INTO log (action,table_name ,table_row,username,action_datetime)
      VALUES ('Update','Job' ,new.id,user(),now());
END //

CREATE TRIGGER jobDELETE_logTrigger
AFTER DELETE ON job
FOR EACH ROW
BEGIN
     INSERT INTO log (action,table_name ,table_row,username,action_datetime)
        VALUES ('Delete','Job' ,old.id,user(),now());
END //

/* Trigger για τον πίνακα user */
CREATE TRIGGER userINSERT_logTrigger
AFTER INSERT ON user
FOR EACH ROW
BEGIN
     INSERT INTO log (action,table_name ,table_row,username,action_datetime)
        VALUES ('Insert','User',new.username,user(),now());
END //

CREATE TRIGGER userUPDATE_logTrigger
AFTER UPDATE ON user
FOR EACH ROW
BEGIN
     INSERT INTO log (action,table_name ,table_row,username,action_datetime)
         VALUES ('Update','User' ,new.username,user(),now());
END //

CREATE TRIGGER userDELETE_logTrigger
AFTER DELETE ON user
FOR EACH ROW
BEGIN
     INSERT INTO log (action,table_name ,table_row,username,action_datetime)
         VALUES ('Delete','User' ,old.username,user(),now());
```

```
END //

/* Trigger για τον πίνακα degree */
CREATE TRIGGER degreeINSERT_logTrigger
AFTER INSERT ON degree
FOR EACH ROW
BEGIN
    INSERT INTO log (action,table_name ,table_row,username,action_datetime)
        VALUES ('Insert','Degree' ,new.titlos,user(),now());
END //

CREATE TRIGGER degreeUPDATE_logTrigger
AFTER UPDATE ON degree
FOR EACH ROW
BEGIN
    INSERT INTO log (action,table_name ,table_row,username,action_datetime)
        VALUES ('Update','Degree' ,new.titlos,user(),now());
END //

CREATE TRIGGER degreeDELETE_logTrigger
AFTER DELETE ON degree
FOR EACH ROW
BEGIN
    INSERT INTO log (action,table_name ,table_row,username,action_datetime)
        VALUES ('Delete','Degree' ,old.titlos,user(),now());
END //

DELIMITER ;
```

Ο παραπάνω κώδικας περιλαμβάνει Trigger που σχετίζονται με τους πίνακες job, user, και degree. Συγκεκριμένα, οποιαδήποτε ενέργεια (insert, update, delete) εφαρμοστεί σε έναν από αυτούς τους τρεις πίνακες, αυτόματα ενημερώνεται και εφαρμόζεται και στοον πίνακα log. Παρακάτω αναλύεται η λειτουργία του κάθε trigger:

➢ ***Για τον πίνακα job:***

❖ Trigger AFTER INSERT (jobINSERT_logTrigger):

- Ενεργοποιείται μετά την εισαγωγή μιας νέας εγγραφής στον πίνακα job.
- Καταγράφει την ενέργεια (Insert), το όνομα του πίνακα (Job), τη γραμμή που εισήχθη (new.id), το όνομα χρήστη του εκτελεστή (user()), και την τρέχουσα ημερομηνία (now()).

❖ Trigger AFTER UPDATE (jobUPDATE_logTrigger):

- Ενεργοποιείται μετά την ενημέρωση μιας υπάρχουσας εγγραφής στον πίνακα job.
- Καταγράφει την ενέργεια (Update), το όνομα του πίνακα (Job), τη γραμμή που ενημερώθηκε (new.id), το όνομα χρήστη του εκτελεστή (user()), και την τρέχουσα ημερομηνία (now()).

❖ Trigger AFTER DELETE (jobDELETE_logTrigger):

- Ενεργοποιείται μετά τη διαγραφή μιας υπάρχουσας εγγραφής από τον πίνακα job.
- Καταγράφει την ενέργεια (Delete), το όνομα του πίνακα (Job), τη γραμμή που διαγράφηκε (old.id), το όνομα χρήστη του εκτελεστή (user()), και την τρέχουσα ημερομηνία (now()).

➢ ***Για τον πίνακα user:***

- Οι Trigger για τον πίνακα user λειτουργούν με αντίστοιχο τρόπο με εκείνους του πίνακα job. Κάθε Trigger (userINSERT_logTrigger, userUPDATE_logTrigger, userDELETE_logTrigger) καταγράφει τις αντίστοιχες ενέργειες για τον πίνακα user.
- Για τον πίνακα degree:
- Οι Trigger για τον πίνακα degree λειτουργούν με αντίστοιχο τρόπο με εκείνους του πίνακα job. Κάθε Trigger (degreeINSERT_logTrigger, degreeUPDATE_logTrigger, degreeDELETE_logTrigger) καταγράφει τις αντίστοιχες ενέργειες για τον πίνακα degree.
- Αυτοί οι Trigger βοηθούν στη διατήρηση ενός αρχείου καταγραφής (log) που καταγράφει σημαντικές ενέργειες που συμβαίνουν στους πίνακες της βάσης δεδομένων, παρέχοντας έτσι ιστορικό των αλλαγών.

➢ ***Για τον πίνακα degree:***

- Οι Trigger για τον πίνακα degree λειτουργούν με αντίστοιχο τρόπο με εκείνους του πίνακα job. Κάθε Trigger (degreeINSERT_logTrigger, degreeUPDATE_logTrigger, degreeDELETE_logTrigger) καταγράφει τις αντίστοιχες ενέργειες για τον πίνακα degree.

  Αυτοί οι Trigger βοηθούν στη διατήρηση ενός αρχείου καταγραφής (log) που καταγράφει σημαντικές ενέργειες που συμβαίνουν στους συγκεκριμένους πίνακες (job, user, degree), παρέχοντας έτσι ιστορικό των αλλαγών.

```
1050    INSERT INTO job VALUES ('6726223','2020-06-7','1200','manager','Athens','raptor','2020-1-5 11:00:00','2020-2-3');
1051 •  select * from log;
1052
1053 •  select * from job;
1054 •  UPDATE job
1055    SET start_date = '2022-12-09'
1056    WHERE id = 6726223;
1057 •  select * from log;
1058
1059 •  DELETE FROM user WHERE username LIKE 'kon%';
1060 •  select * from log;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

log_id	Action	Table_name	table_row	username	action_datetime
1	Insert	Job	6726223	root@localhost	2024-01-21 22:53:44
2	Update	Job	6726223	root@localhost	2024-01-21 22:53:44
3	Delete	User	konmag	root@localhost	2024-01-21 22:53:44
NULL	NULL	NULL	NULL	NULL	NULL

TRIGGER 2:

Παρατίθεται ο κώδικας του προκείμενου trigger:

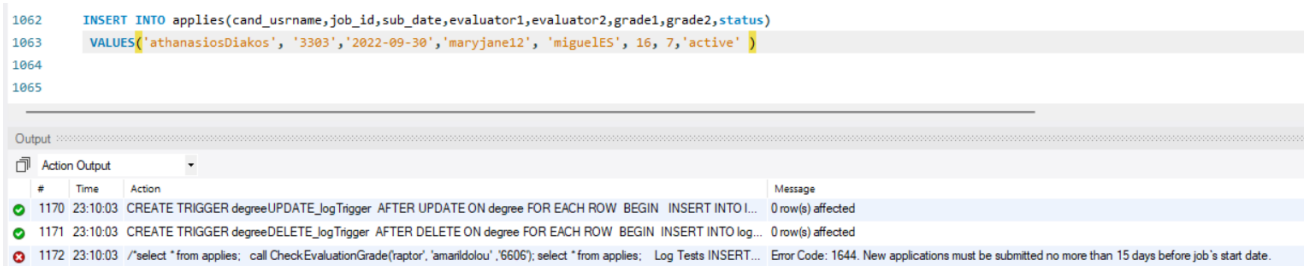
```
CREATE TRIGGER AppliesCheck1
BEFORE INSERT ON applies
FOR EACH ROW
BEGIN
DECLARE active_applications INT(4);
DECLARE job_start_date DATE;

SELECT COUNT(*) INTO active_applications
FROM applies
WHERE cand_usrname=NEW.cand_usrname AND status='active';

 IF active_applications >= 3 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot have more than 3 active applications.';
    END IF;

    SELECT start_date INTO job_start_date
    FROM job
    WHERE job.id = NEW.job_id;

    IF  DATEDIFF(job_start_date, NEW.sub_date) < 15 or NEW.sub_date > job_start_date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'New applications must be submitted no more than 15 days before job`s start
date.';
    END IF;
END//
```

Ο παραπάνω Trigger, με την ονομασία **AppliesCheck1**, εκτελείται πριν από την εισαγωγή μιας νέας εγγραφής στον πίνακα applies. Συγκεκριμένα, ελέγχει αν η εισαγωγή της αίτησης απέχει λιγότερο από 15 ημέρες σε σχέση με την ημερομηνία έναρξης της θέσης ή αν ο εργαζόμενος ο οποίος αιτείται την θέση έχει ήδη τρεις ενεργές αιτήσεις, και αν κάποια από τις συνθήκες αυτές ισχύει, αποτρέπει την αίτηση. Ας αναλύσουμε τη λειτουργία του:

- *Δήλωση μεταβλητών:*
  - active_applications: Αναπαριστά τον αριθμό των ενεργών αιτήσεων που έχει καταθέσει ο συγκεκριμένος υποψήφιος.
  - job_start_date: Αναπαριστά την ημερομηνία έναρξης ενός συγκεκριμένου έργου.
  - Έλεγχος αριθμού ενεργών αιτήσεων:
  - Ένα από τα χαρακτηριστικά αυτού του Trigger είναι να ελέγχει αν ο υποψήφιος έχει ήδη τρεις ενεργές αιτήσεις. Αυτό επιτυγχάνεται με τον έλεγχο της μεταβλητής active_applications.
  - Εάν ο υποψήφιος έχει ήδη τρεις ενεργές αιτήσεις, το Trigger εκτελεί μια εξαίρεση (SIGNAL SQLSTATE '45000') και επιστρέφει ένα μήνυμα λάθους ('Cannot have more than 3 active applications.').

- *Έλεγχος ημερομηνίας υποβολής:*
  - Αντίστοιχα, το Trigger ελέγχει αν η νέα αίτηση υποβάλλεται περισσότερο από 15 ημέρες πριν από την ημερομηνία έναρξης της εργασίας (job_start_date).

- Εάν η αίτηση υποβάλλεται περισσότερο από 15 ημέρες πριν ή μετά την ημερομηνία έναρξης της εργασίας, το Trigger εκτελεί μια εξαίρεση και επιστρέφει ένα αντίστοιχο μήνυμα λάθους ('New applications must be submitted no more than 15 days before jobs start date.'`).
- Συνολικά, αυτός ο Trigger προσφέρει έλεγχο στον αριθμό των ενεργών αιτήσεων και στη σωστή χρονική στιγμή υποβολής νέας αίτησης σε σχέση με την ημερομηνία έναρξης μιας εργασίας.



TRIGGER 3:

Παρατίθεται ο κώδικας του προκείμενου trigger:

```
CREATE TRIGGER AppliesCheck2
BEFORE UPDATE ON applies
FOR EACH ROW
BEGIN
DECLARE job_start_date DATE;
DECLARE c_status enum('active','canceled');

SELECT new.status INTO c_status
  FROM applies
  WHERE job_id = new.job_id AND cand_usrname = new.cand_usrname;

SELECT start_date INTO job_start_date
   FROM job
   WHERE job.id = NEW.job_id;

   IF  c_status = 'canceled' AND DATEDIFF(job_start_date, NEW.sub_date) < 10 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Applications can be canceled up to 10 days before job start_date.';
      END IF;
END//
```

Ο παραπάνω Trigger, **AppliesCheck2**, εκτελείται πριν από μια ενημέρωση (UPDATE) στον πίνακα applies. Συγκεκριμένα, αποτρέπει την ακύρωση μιας αίτησης για μια θέση εργασίας αν η ημερομηνία που γίνεται η ακύρωση της αίτησης απέχει λιγότερο από 10 μέρες από την ημερομηνία έναρξης της θέσης (start_date). Επίσης αποτρέπει την ενεργοποίηση ακυρωμένης αίτησης αν ο συγκεκριμένος εργαζόμενος έχει ήδη τρεις ενεργές αιτήσεις. Παρακάτω παρουσιάζεται η ανάλυση του κώδικα:

➢ *Δήλωση Μεταβλητών:*

- job_start_date: Αναπαριστά την ημερομηνία έναρξης μιας εργασίας που σχετίζεται με την αίτηση.
- c_status: Αναπαριστά την κατάσταση (status) της αίτησης.

➢ *Ανάκτηση Κατάστασης και Ημερομηνίας:*

- Χρησιμοποιούνται δύο εντολές SELECT για να ανακτηθεί η κατάσταση (status) και η ημερομηνία έναρξης (start_date) που σχετίζονται με την αίτηση που θα ενημερωθεί.

➢ *Έλεγχος Ακύρωσης Αίτησης:*

- Το Trigger ελέγχει αν η αίτηση είναι σε κατάσταση 'canceled' (c_status = 'canceled').
- Αν η κατάσταση είναι 'canceled', ελέγχει επίσης αν η ημερομηνία υποβολής της αίτησης (NEW.sub_date) βρίσκεται λιγότερο από 10 ημέρες πριν από την ημερομηνία έναρξης της εργασίας (job_start_date).
- Εάν και οι δύο συνθήκες ισχύουν, τότε εκτελείται μια εξαίρεση (SIGNAL SQLSTATE '45000') και επιστρέφει ένα μήνυμα λάθους ('Applications can be canceled up to 10 days before job start_date.').

Συνολικά, αυτός ο Trigger προσφέρει έναν έλεγχο για το εάν είναι εφικτή η ακύρωση μιας αίτησης, λαμβάνοντας υπόψη την κατάσταση της αίτησης και την απόσταση σε ημέρες μεταξύ της ημερομηνίας υποβολής και της ημερομηνίας έναρξης της εργασίας.

```
1062    UPDATE applies
1063    SET status = 'canceled' and sub_date = '2022-09-30'
1064    WHERE  job_id = '3303';
1065 •  SELECT * FROM applies;
```

Output

Action Output

#	Time	Action	Message
1277	23:23:43	CREATE TRIGGER userDELETE_logTrigger AFTER DELETE ON user FOR EACH ROW BEGIN INSERT INTO log (acti...	0 row(s) affected
1278	23:23:43	CREATE TRIGGER degreeINSERT_logTrigger AFTER INSERT ON degree FOR EACH ROW BEGIN INSERT INTO log ...	0 row(s) affected
1279	23:23:43	CREATE TRIGGER degreeUPDATE_logTrigger AFTER UPDATE ON degree FOR EACH ROW BEGIN INSERT INTO l...	0 row(s) affected
1280	23:23:43	CREATE TRIGGER degreeDELETE_logTrigger AFTER DELETE ON degree FOR EACH ROW BEGIN INSERT INTO log...	0 row(s) affected
1281	23:23:43	/*select * from applies; call CheckEvaluationGrade('raptor', 'amarildolou' ,'6606'); select * from applies; Log Tests INSERT...	Error Code: 1292. Truncated incorrect DOUBLE value: 'canceled'

ΜΕΡΟΣ Β: GUI
ΚΕΦΑΛΑΙΟ 4:

Η γραφική διεπαφή(GUI), που σχεδιάσαμε ενσωματώνει όλες τις λειτουργίες της βάσης δεδομένων που κατασκευάσαμε, ενώ δίνει την δυνατότητα στους πιστοποιημένους διαχειριστές της(Data Base Admins) να επιλέξουν όποιον πίνακα αυτοί θέλουν και να εκτελέσουν τις εξής λειτουργίες :

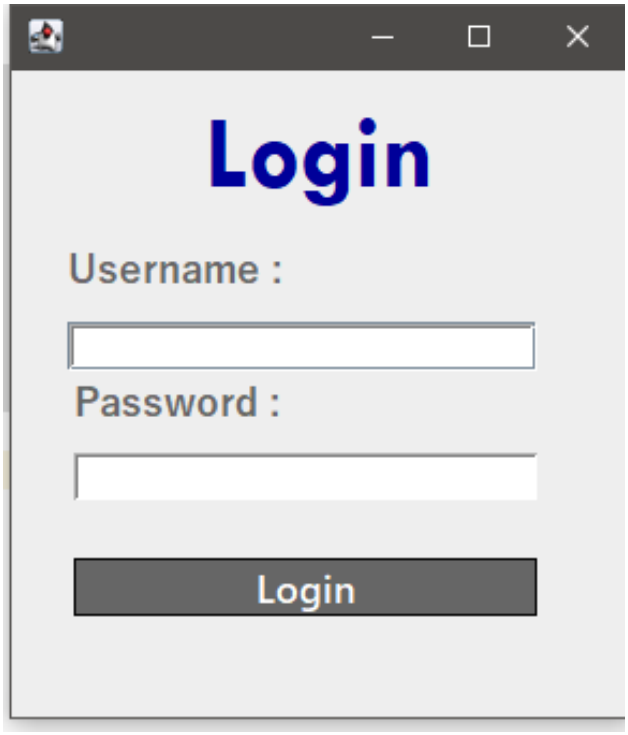
- Εμφάνιση όλων των δεδομένων του εκάστοτε πίνακα που διάλεξε ο admin.
- Εισαγωγή μιας νέας εγγραφής στον πίνακα.

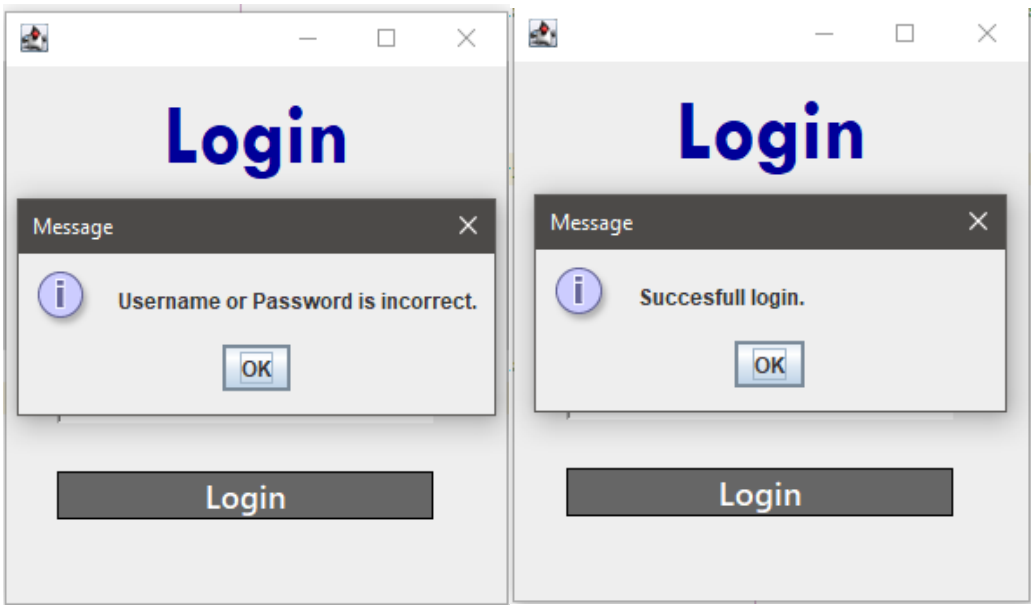
- Ενημέρωση μίας ήδη υπάρχουσας εγγραφής με μία καινούργια τιμή σε οποιοδήποτε πεδίο της.
- Διαγραφή της επιλεγμένης εγγραφής.
- Εκτέλεση των διαδικασιών(procedures) που δημιουργήσαμε στο προηγούμενο μέρος του project.

ΠΕΡΙΓΡΑΦΗ ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ

ΠΑΡΑΘΥΡΟ LOGIN:

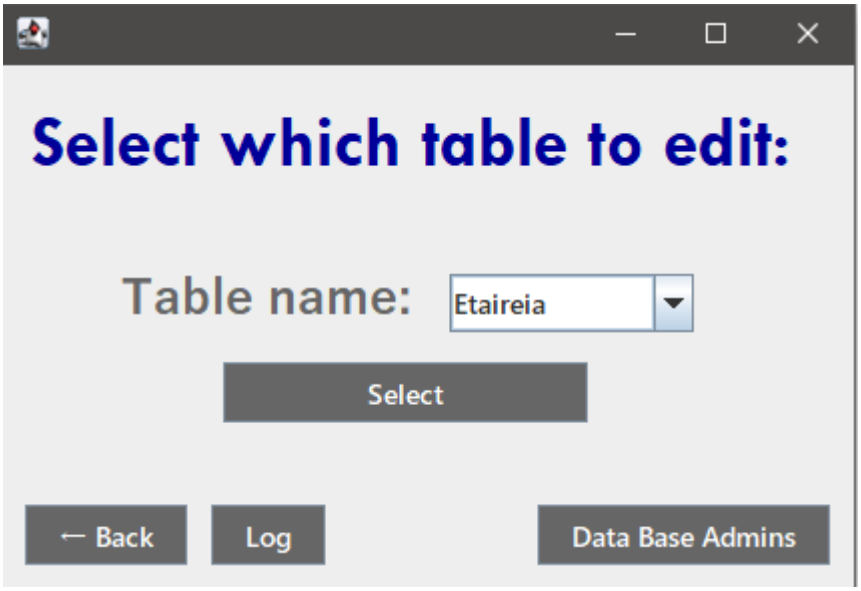
Τον χρήστη, κατά την εκτέλεση του προγράμματος, αρχικά υποδέχεται ένα παράθυρο είσοδου login. Το παράθυρο αυτό, περίμενει από το χρήστη να είσαγει τα στοιχεία του (credentials), ώστε να του δωθεί πρόσβαση στην βάση εάν αυτά είναι σωστά. Το πρόγραμμα, λοίπον, ελέγχει εάν τα στοίχεια που έδωσε ο χρήστης (username και password), υπάρχουν στον πίνακα DataBaseAdmin της βάσης και πέταει το αναλογό μύνημα εάν αυτα είναι σώστα ή όχι.

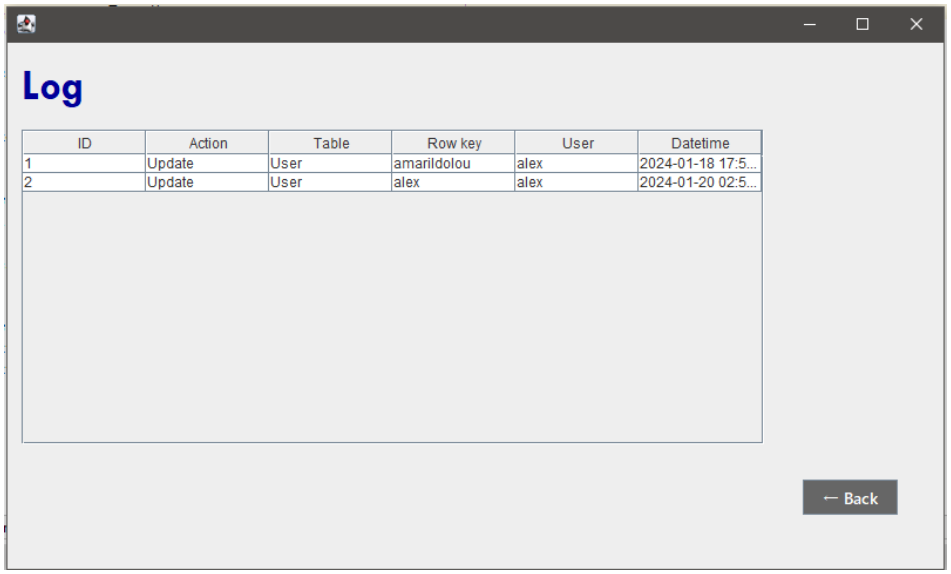




ΠΑΡΑΘΥΡΟ TABLE_SELECT & LOG:

Αφού πραγματοποιηθεί η πιστοποίηση του χρήστη με επιτυχία, το login παράθυρο κλείνει και την θέση του παίρνει η διεπαφή επιλογής πίνακα. Η διεπαφή αυτή περιέχει, αρχικά, ένα μενού που περιλαμβάνει όλους τους πίνακες της βάσης, ώστε να διαλέξει ο χρήστης ποιόν πίνακα θέλει να τροποποιήσει, και επιπλέον περιέχει κουμπί **Back,**το οποίο επιστρέφει στο παράθυρο login, κουμπί **Log** , το οποίο εμφανίζει ένα νέο παράθυρο που περιέχει τα δεδομένα του πίνακα log από την βάση, και ένα ακόμα κουμπί, που οδηγεί σε ενα, επίσης, νέο παράθυρο με τις εισαγωγές του πίνακα DataBaseAdmin.

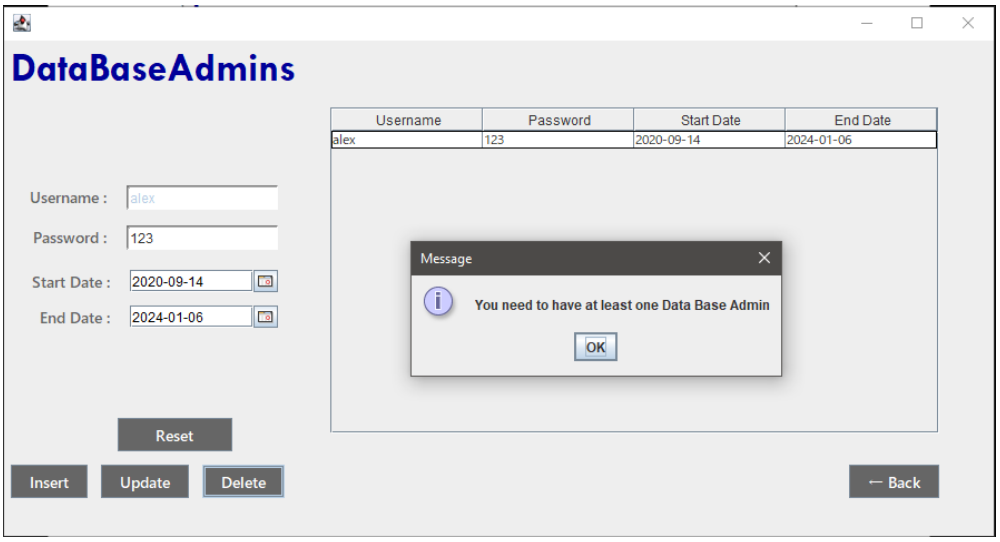




ΠΑΡΑΘΥΡΑ ΠΙΝΑΚΩΝ:

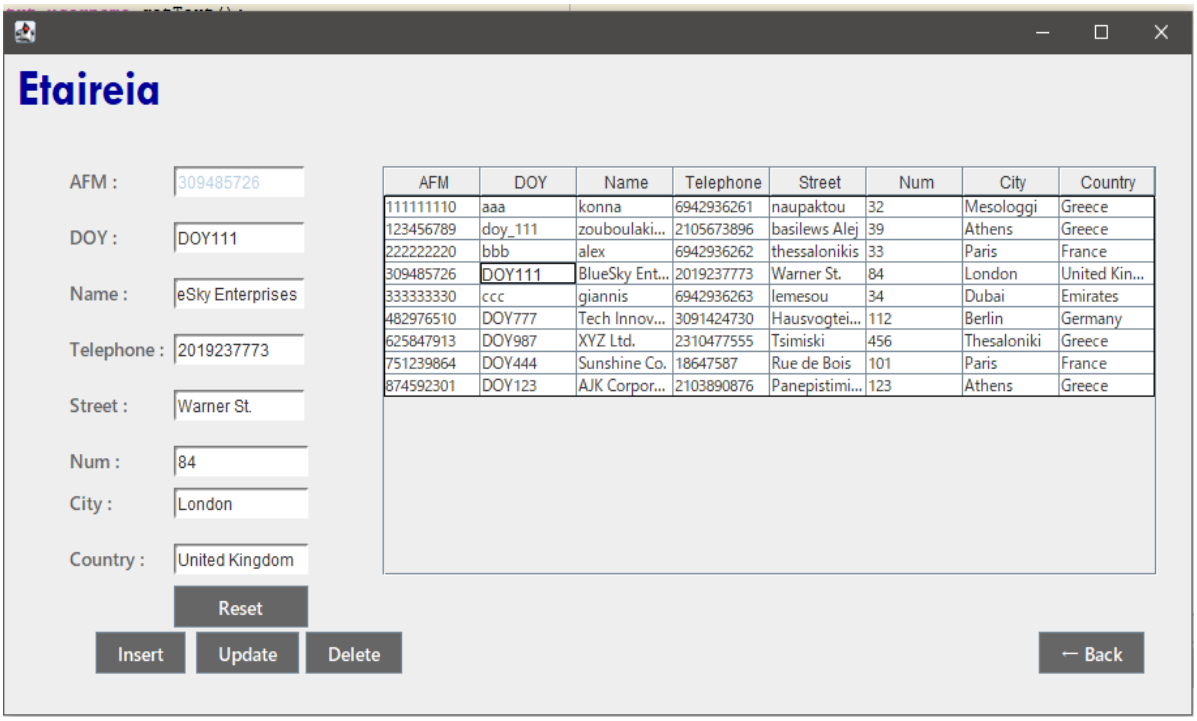
DataBaseAdmin:

Ας ξεκινήσουμε με τον πίνακα DataBaseAdmin, που αναφέρθηκε και προηγουμένως. Στο παράθυρο αυτό εμφανίζονται όλες οι εγγραφές του πίνακα και δίνεται η δυνατότητα στον χρήστη να μπορεί να εισάγει καινούργια, να ενημερώσει κάποια ήδη υπάρχουσα ή να διαγράψει κάποια εγγραφή. Όσων αφορά την ενημέρωση των δεδομένων, ο χρήστης μπορεί ,αφού διαλέξει ποια εγγραφή θέλει να ενημερώσει κάνοντας κλικ στον πίνακα, να αλλάξει τις τιμές σε όλα τα πεδία εκτός του username ,το οποίο γίνεται disabled όταν πραγματοποιηθεί κλικ στον πίνακα, διότι το πεδίο αυτό χρησιμοποιείται σαν κλειδί. Επίσης, στην περίπτωση που υπάρχει μόνο μια εγγραφή στον πίνακα, εάν ο χρήστης την επιλέξει και κλικάρει στο κουμπί της διαγράφης, το πρόγραμμα θα εμφανίσει μήνυμα που τον ενημερώνει ότι δεν μπορεί να διαγράψει αυτήν την εγγραφή, διότι είναι η τελευταία και χωρίς καμία εγγραφή δεν θα μπόρει να συνδεθεί στην βάση. Τέλος, υπάρχει και το κουμπί **Reset ,** το οποίο καθαρίζει τα πεδία κειμένου της διεπαφής.



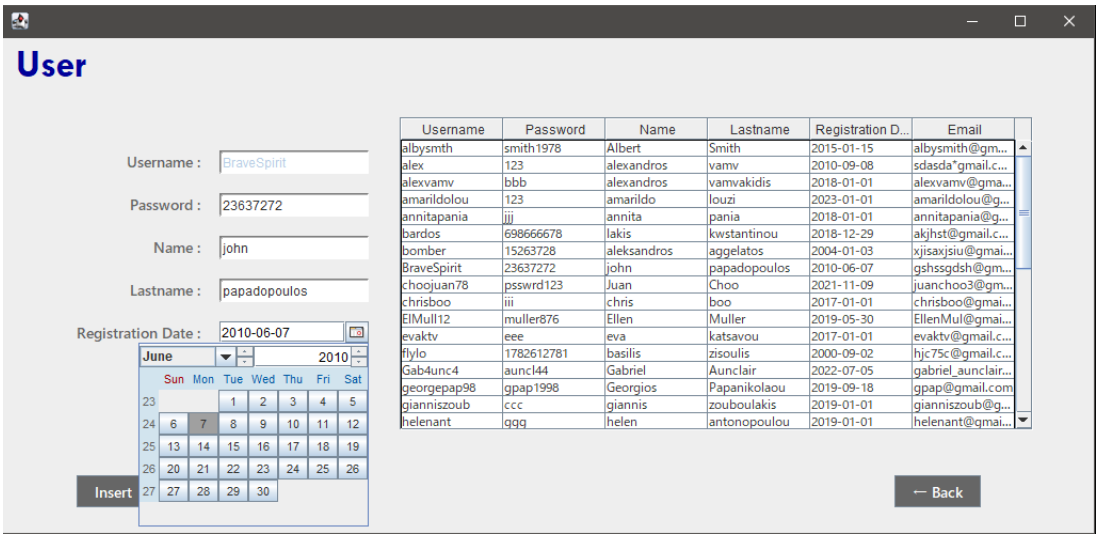
Etairia:

Το παράθυρο για τον πίνακα Etaireia έχει παρόμοια δομή με αυτό του προηγούμενου πίνακα. Όταν επιλέγει δηλαδή ο χρήστης σε κάποια από τις εγγραφές, μεταφέρονται οι τιμές της στα αντίστοιχα πεδία της διεπαφής, ώστε να μπορούν να τροποποιηθούν. Περιέχει τα ίδια κουμπιά με την προηγούμενη διεπαφή και γενικά εκτελεί τις ίδιες λειτουργίες. Και εδώ απενεργοποιείται το πεδίο AFM όταν επιλεχθεί μια εγγραφή, έτσι ώστε να μην μπορεί ο χρήστης να επέμβει στα πρωτεύων κλειδιά του πίνακα και να προκαλέσει πρόβλημα στην βάση.



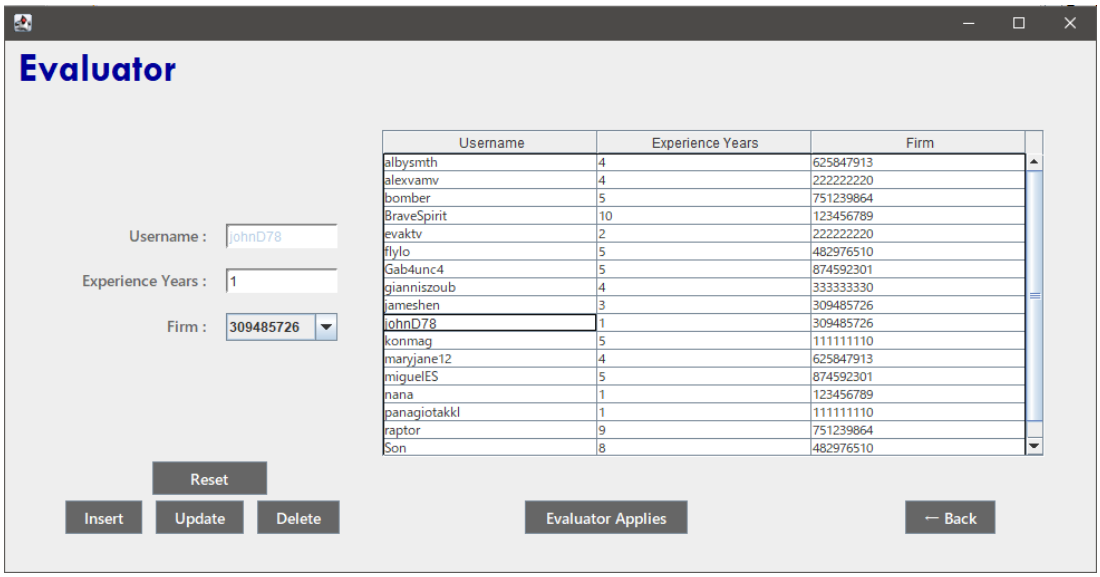
User:

Ακρίβως τα ίδια ισχύουν και εδώ για τον πίνακα User. Στην διεπάφη αυτή, όπως και σε αυτή του DataBaseAdmin, εχεί χρήσιμοποιήθει ενα ειδικό πεδίο jDateChooser, το οποίο εμφανίζει ένα παραθυράκι επιλογής ημερομηνίας μέσα από ημερολόγιο.



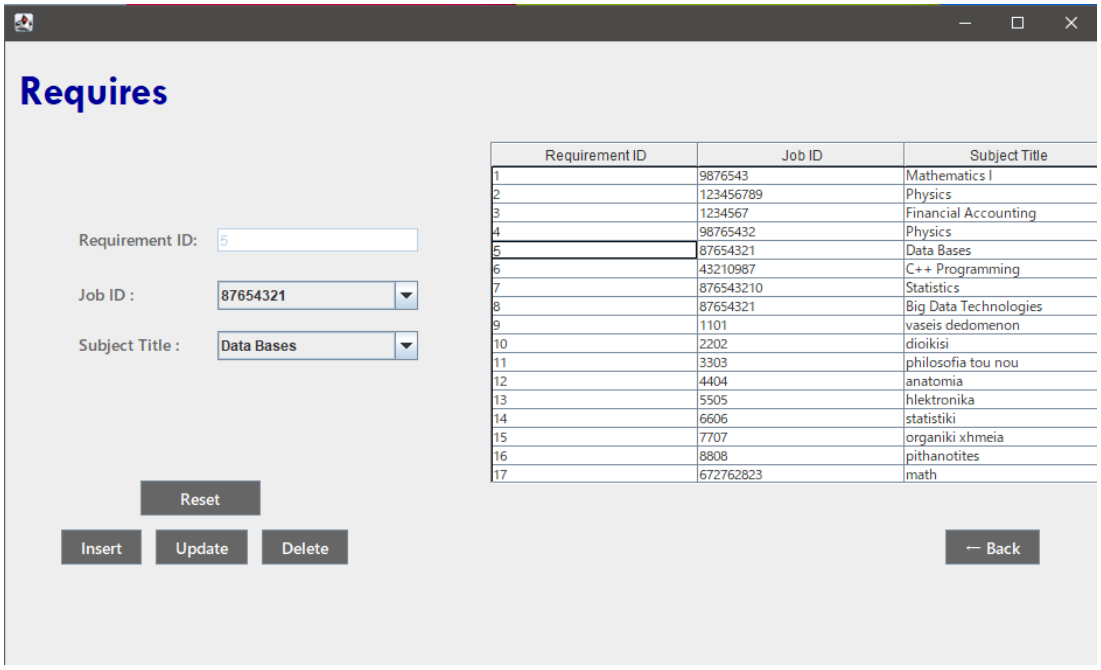
Evaluator:

Ίδια δομή υπάρχει και στο παράθυρο του Evaluator, απλά εδώ έχει προστέθει ενα ComboBox, το οποίο περιέχει τα usernames όλων των εγγραφών του πίνακα User, διότι το πεδίο username στον πίνακα Evaluator είναι ξένο κλειδί στο κλειδί username του πίνακα User. Επιπλέον, υπάρχει και το κουμπί **Evaluator Applies,** το οποίο θα αναλυθει παρακάτω στην τεκμηρίωση του ερωτήματος 3.2.3.



Requires:

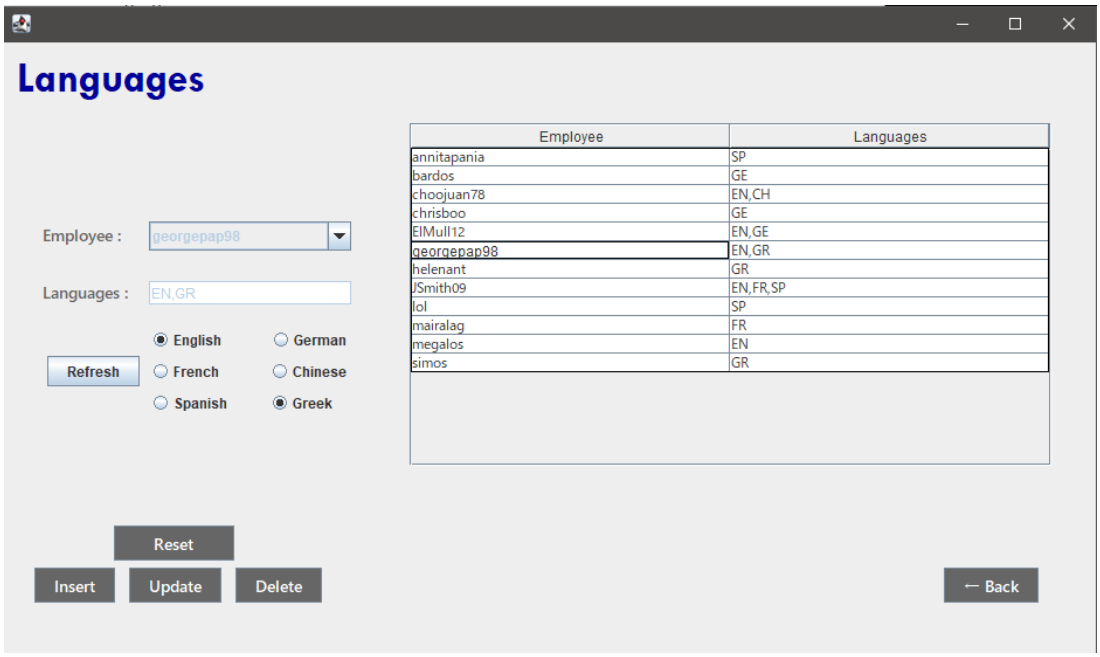
Στον πίνακα requires, αποφασίσαμε να προσθέσουμε ένα επιπλέον πεδίο το οποίο ονομάσαμε requirement id. Το πεδίο αυτό λειτουργεί ως κλειδί για τον πίνακα requires και το προσθέσαμε, ώστε να μπορεί ο χρήστης να πραγματοποιήσει τις επιθυμητές λειτουργίες στον πίνακα, αφού το πεδίο Job ID αλλά και το Subject Title ήταν πρωτεύον κλειδιά πριν την αλλαγή μας.



Requirement ID	Job ID	Subject Title
1	9876543	Mathematics I
2	123456789	Physics
3	1234567	Financial Accounting
4	98765432	Physics
5	87654321	Data Bases
6	43210987	C++ Programming
7	876543210	Statistics
8	87654321	Big Data Technologies
9	1101	vaseis dedomenon
10	2202	dioikisi
11	3303	philosofia tou nou
12	4404	anatomia
13	5505	hlektronika
14	6606	statistiki
15	7707	organiki xhmeia
16	8808	pithanotites
17	672762823	math

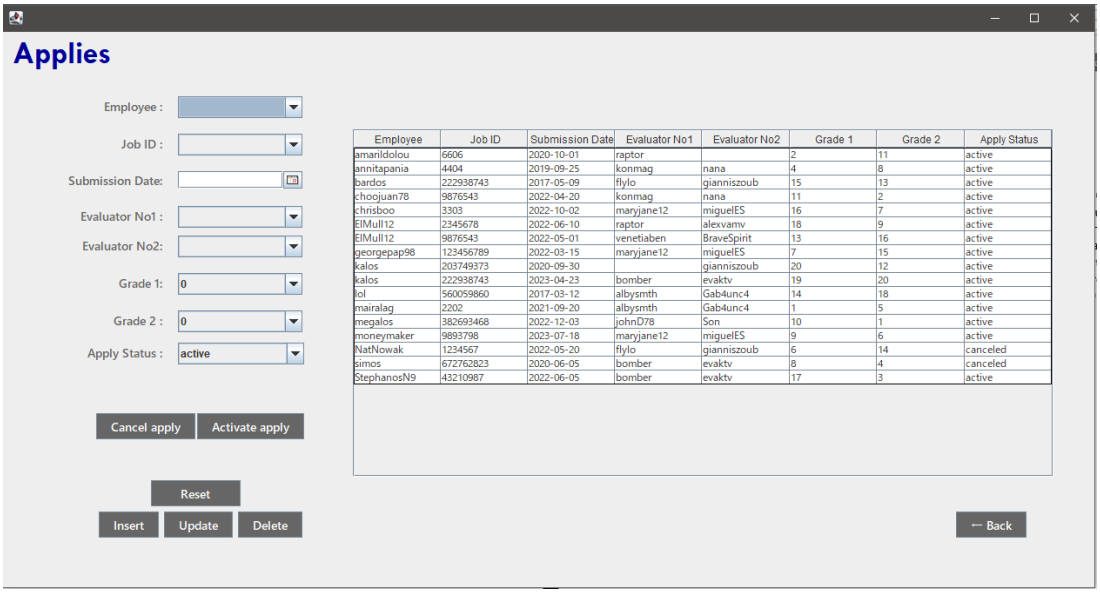
Languages:

Στο παράθυρο του πίνακα languages, θέλαμε να περιορίσουμε τις επιλογές γλώσσας που μπορεί να πληκτρολογήσει ο χρήστης, ώστε να αποφευχθούν θέματα στην εισαγωγή ή ενημέρωση εγγραφών. Αυτό γιατί το πεδίο lang στο πίνακα μας είναι μορφής set και ο χρήστης πρέπει να επιλέξει συγκεκριμένες γλώσσες και να τις πληκτρολογήσει με συγκεκριμένο τρόπο (την μία μετά την άλλη με κόμμα ανάμεσα). Για να το υλοποιήσουμε αυτό, χρησιμοποιήσαμε τα radio buttons που διαθέτει η java και ένα επιπλέον κουμπί **Refresh**. Αρχικά, όπως και στα άλλα παράθυρα που δημιουργήσαμε έτσι και εδώ, όταν ο χρήστης επιλέγει στον πίνακα των εγγραφών οι τιμές της εγγραφής που επέλεξε μεταφέρονται στα αντίστοιχα πεδία κειμένου της διεπαφής. Συγκεκριμένα, λοιπόν, εδώ την στιγμή του κλικ αντιγράφεται το String που υπάρχει στο πεδίο languages της διεπαφής σε ένα άλλο String (existingtext) και ανάλογα με τις τιμές που περιέχει, επιλέγονται και τα κατάλληλα buttons των γλωσσών. Έπειτα ο χρήστης διαλέγει ποιες γλώσσες θέλει να προσθέσει ή να αφαιρέσει και πατάει το κουμπί Refresh. Το κουμπί Refresh, αφού πατηθεί καθαρίζει το existingtext και με διάφορες δομές if ελέγχει ποιες γλώσσες έχουν επιλεχθεί και ανανεώνει το πεδίο κειμένου languages.



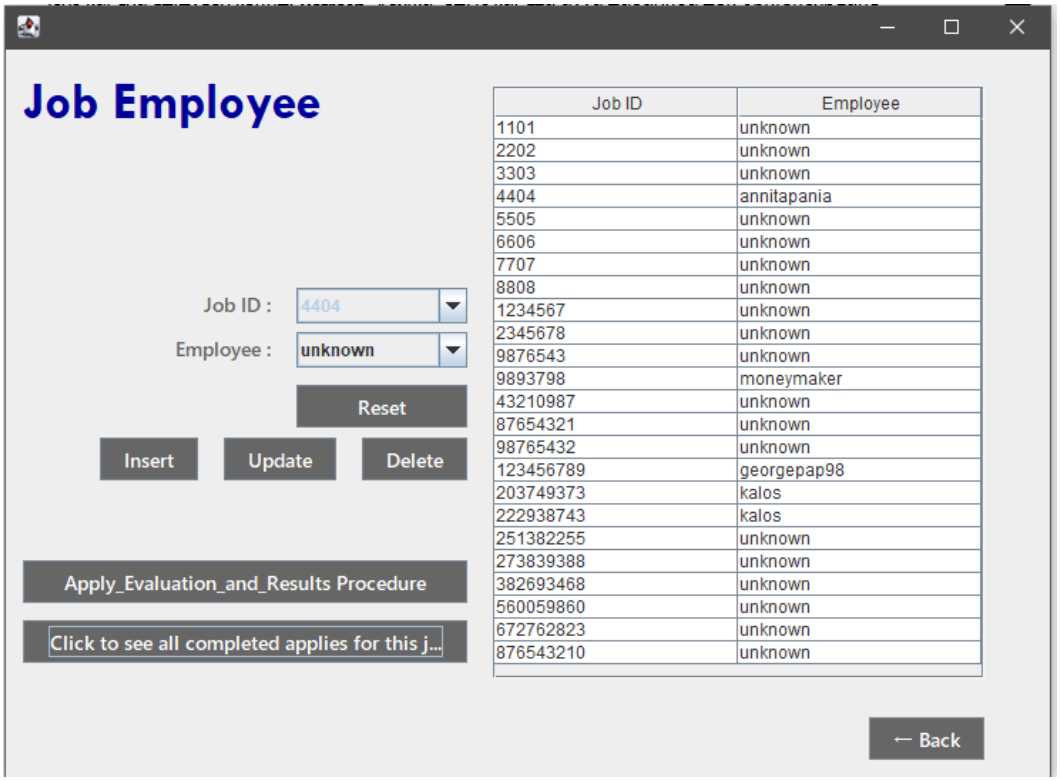
Applies:

Στην συγκεκριμένη διεπαφή, ενσωματώσαμε την λειτουργία της procedure **EditApplies**, που δημιουργήσαμε στο SQL μέρος του project. Όταν, λοιπόν, ο χρήστης εισάγει νέα εγγραφή με το κουμπί insert, τότε εάν μία από τις θέσεις των evaluators είναι κενή θα εκτελέσει την EditApplies και στην θέση του ορίσματος action θα τοποθετήσει μόνο του το πρόγραμμα την τιμή “i” εμφανίζοντας και το ανάλογο μήνυμα. Έτσι, και με τα κουμπιά Cancel Apply και Activate apply εκτελεί και πάλι την procedure **EditApplies,** αλλά τώρα στο πεδίο action, το πρόγραμμα εισάγει τις τιμές ‘”c” και “a” αντίστοιχα, ενώ εμφανίζει και τα ανάλογα μνήματα.



Job_Employee:

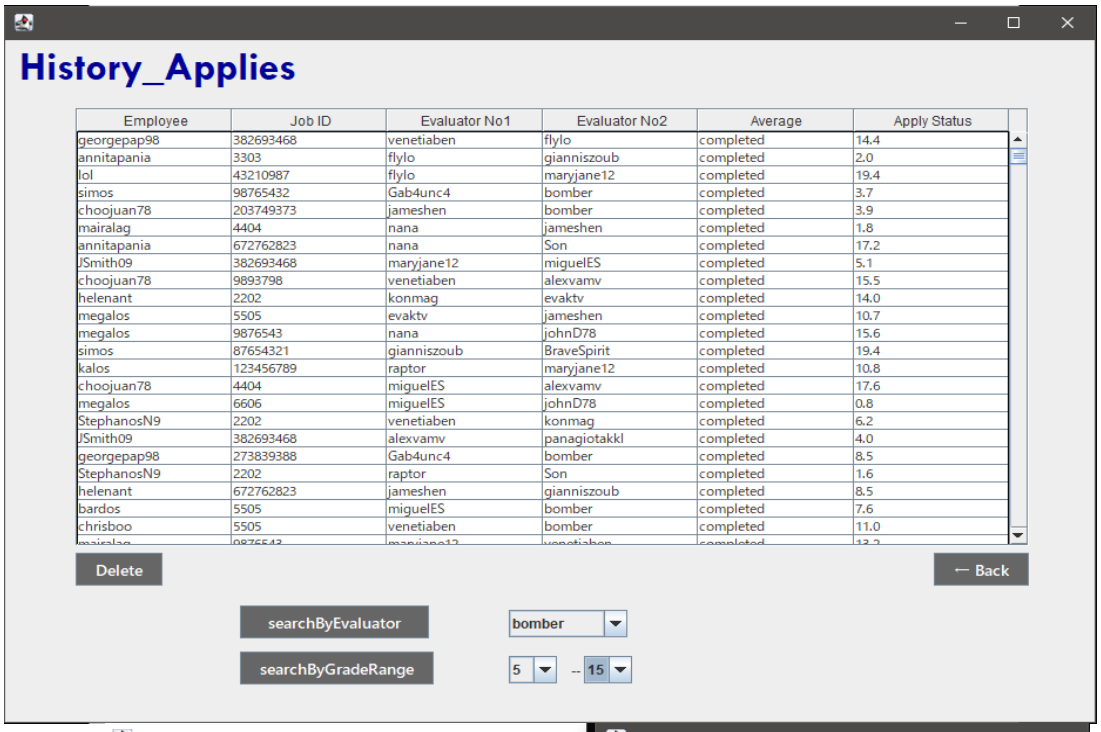
Στο παράθυρο του Job_Employee, εντάξαμε δυο ακόμα κουμπιά, το **Apply_Evaluation_and_Results** και το **Click to see all completed applies for this job**. Το πρώτο εκτελεί την λειτουργία της ομώνυμης procedure της βάσης για το job_id που έχει επιλέξει ο χρήστης. Το δεύτερο θα αναλυθεί στην τεκμηρίωση του ερωτήματος 3.2.3.



Job ID	Employee
1101	unknown
2202	unknown
3303	unknown
4404	annitapania
5505	unknown
6606	unknown
7707	unknown
8808	unknown
1234567	unknown
2345678	unknown
9876543	unknown
9893798	moneymaker
43210987	unknown
87654321	unknown
98765432	unknown
123456789	georgepap98
203749373	kalos
222938743	kalos
251382255	unknown
273839388	unknown
382693468	unknown
560059860	unknown
672762823	unknown
876543210	unknown

History_applies:

Στο history_applies παράθυρο κρίναμε περιττό ο χρήστης να μπορεί να εισάγει και να ενημερώσει κάποια εγγραφή. Επίσης προσθέσαμε με μορφή κουμπιών τις λειτουργίες των δυο procedures που δημιουργήσαμε, που εκμεταλλεύονται την παρουσία των indexes στον πίνακα history_applies. Στο κουμπί στης SearchByEvaluator procedure ο χρήστης διαλέγει από το μενού τον evaluator που θέλει και πατώντας το εμφανίζονται σε ένα jDialog όλες οι αξιολογήσεις του evaluator στον πίνακα history_applies. Το ίδιο ισχύει και για το κουμπί της SearchByGradeRange procedure, απλά εδώ ο χρήστης διαλέγει το άνω και κάτω όριο των βαθμών των εγγράφων που επιθυμεί.



Υπόλοιπα Παράθυρα Πινάκων:

Τα παράθυρα των πινάκων που δεν αναφέρθηκαν παραπάνω (Employee, Job, Subject, Project, Degree, Has_Degree) περιέχουν στοιχεία που αναλύθηκαν και τεκμηριώθηκαν στις παραπάνω παραγράφους.

ΑΝΑΦΟΡΕΣ ΣΤΟΝ ΚΩΔΙΚΑ:

UpdateDB:

Σε όλα τα παράθυρα των πινάκων που δημιουργήσαμε χρησιμοποιήθηκε η συνάρτηση UpdateDB. Η συνάρτηση παίρνει όλες τις εγγραφές του επιλεγμένου πίνακα και τις μεταφέρει στο jTable που υπάρχει στην γραφική μας διεπαφή. Η συνάρτηση αυτή εκτελείται αρχικά με την πρώτη εκτέλεση της διεπαφής και υστέρα κάθε φορά που ο χρήστης πραγματοποιεί εισαγωγή, ενημέρωση ή διαγραφή κάποιας εγγραφής.

Κώδικας :

```
public  void UpdateDB(){
      try{
         Class.forName("com.mysql.jdbc.Driver");
         Connection con = DriverManager.getConnection(DB_URL,"root","AlexBam03!");
         PreparedStatement stmnt = con.prepareStatement("SELECT * FROM user");
         ResultSet resultSet = stmnt.executeQuery();

         ResultSetMetaData metadata = resultSet.getMetaData();
         int numberOfColumns = metadata.getColumnCount();

         DefaultTableModel RecordTable = (DefaultTableModel)User_Table.getModel();
               RecordTable.setRowCount(0);

          while (resultSet.next()){
          Vector columnData = new Vector();
            for (int i = 1; i <= numberOfColumns; i++){
                columnData.add(resultSet.getString("Username"));
                columnData.add(resultSet.getString("Password"));
                columnData.add(resultSet.getString("Name"));
                columnData.add(resultSet.getString("Lastname"));
                columnData.add(resultSet.getString("reg_date"));
                columnData.add(resultSet.getString("Email"));
            }
            RecordTable.addRow(columnData);
          }
      }catch (Exception ex) {
          JOptionPane.showMessageDialog(null, ex);
      }
   }
```

Στον πίνακα user για παράδειγμα εκτελεί το query “Select * from user” και εισάγει κάθε γραμμή του αποτελέσματος στο jTable της διεπαφής που ονομάσαμε User_Table.

Updatecombo_combobox_name:

Η συγκεκριμένη συνάρτηση ενημερώνει το/τα combobox της εκάστοτε διεπαφής με τις τιμές του πεδίου του πίνακα που θα επιλέξουμε. Αυτή η συνάρτηση όπως και η UpdateDB εκτελείται αρχικά με την πρώτη εκτέλεση της διεπαφής, με μόνη διαφορά ότι αυτή δεν ξανά εκτελείται εκτός αν κλείσουμε και ξανανοίξουμε την συγκεκριμένη διεπαφή.

Κώδικας :

```
public void UpdateCombo_title(){
        String dropdown = "Select * from degree";
        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection(DB_URL,"root","AlexBam03!");
            PreparedStatement stmnt = con.prepareStatement(dropdown);
            ResultSet rs = stmnt.executeQuery();
            while(rs.next()){
                txtHDeg_titlos.addItem(rs.getString("titlos"));
            }
        }catch (Exception e){
    }
```

Εδώ παρουσιάζεται ο κώδικας ανανέωσης του μενού επιλογής ονόματος πτυχίου στον πίνακα requires. Όπως φαίνεται, στο μενού εισάγονται όλες οι εγγραφές του πεδίου title του πίνακα degree, ώστε ο χρήστης να διαλέξει μέσα από συγκεκριμένες επιλογές για να μην γίνονται λάθη.

ΕΓΚΥΡΟΤΗΤΑ ΔΕΔΟΜΕΝΩΝ:

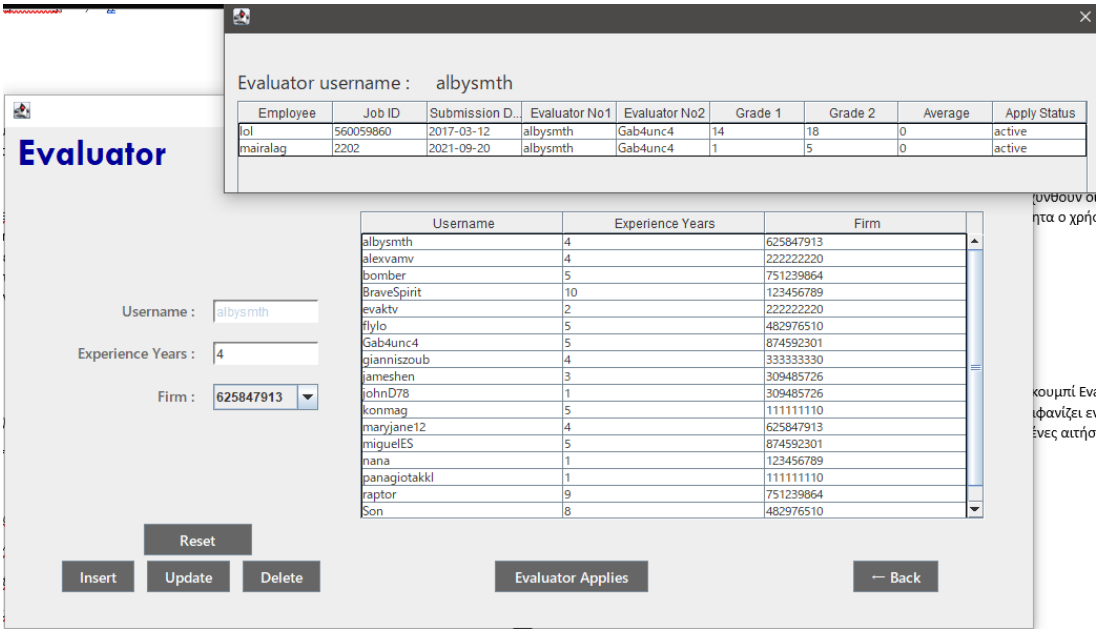
Σε όλες της διεπαφές που δημιουργήσαμε, χρησιμοποιήσαμε κάποιες μεθόδους αντιμετώπισης προβλημάτων εγκυρότητας , που σκοπό είχαν να περιορίσουν, πρακτικά, τις ελεύθερες επιλογές του χρήστη όσων αφορά την τροποποίηση των εγγραφών στους πίνακες αλλά και την εισάγωγη εγγραφών σε ορισμένους από αυτούς. Αρχίκα, αφαιρέσαμε από τον χρήστη την δυνατότητα εάν επίλεξει μια εγγραφή να τροποποιήσει το πρωτεύον κλειδί της, απενεργοποιώντας το συγκεκριμένο πεδίο. Έπειτα, σε όποια πεδία αποτελούσαν ξένα κλειδιά σε άλλους πίνακες (βλπ. Πεδίο username του πίνακα evaluator στο πεδίο username τοτ πίνακα user) αντικαταστήσαμε τα πεδία κειμένου της διεπαφής με μενου τύπου comboBox, ώστε ο χρήστης να επιλέγει από συγκεκριμένες τιμές ώστε να αποφευχθούν λάθη και επιταχυνθούν οι διαδικασίες. Επιπλέον προσθέσαμε σε όποιες διεπαφές ήταν δυνατό, την δυνατότητα ο χρήστης να επιλέγει ημερομηνία μέσα απο ενα μικρό παράθυρο ημερολογίου.

BONUS ZHTOYMENA:

Οι έξτρα λειτούργιες που προσθέσαμε είναι οι εξής :

EVALUATOR APPLIES

Στη διεπαφή του πίνακα evaluator προσθέσαμε το κουμπί Evaluator Applies, το οποίο παίρνει σαν όρισμα τον επιλεγμένο evaluator και εμφανίζει ένα jDialog παράθυρο που περιέχει έναν πίνακα με όλες τις ενεργές ή ακυρωμένες αιτήσεις που έχει αξιολογήσει ο συγκεκριμένος evaluator.



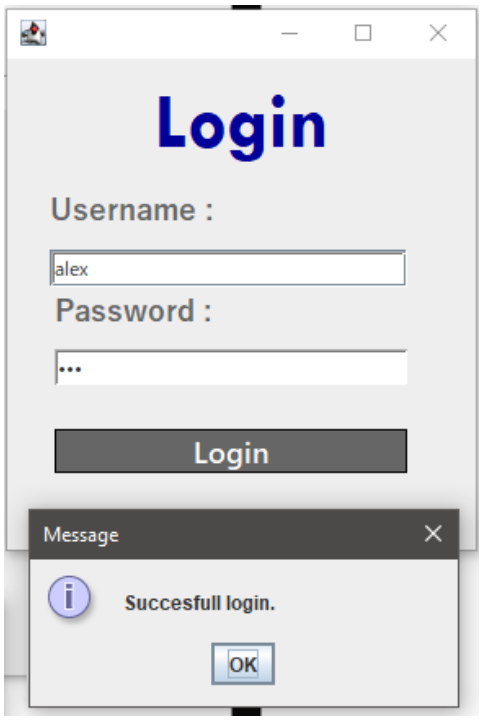
CLICK TO SEE ALL COMPLETED APPLIES FOR THIS JOB

Στη διεπαφή του πίνακα job_employee προσθέσαμε το κουμπί Click to see all completed applies for this job. Το κουμπί αυτό εμφανίζει ενα jDialog παράθυρο που περιέχει έναν πίνακα με όλες τις εγγραφές από τον πίνακα history_applies, όπου το status είναι complete και το average 0, δήλαδη όλες τις εγγραφές που απέτυχαν για την συγκεκριμένη θέση εργασίας.

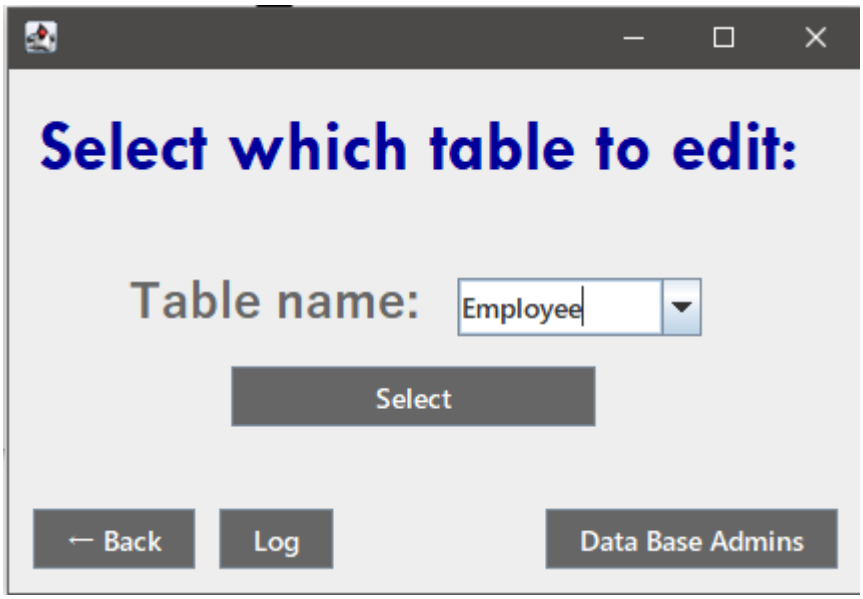
ΠΙΘΑΝΟ ΣΕΝΑΡΙΟ ΧΡΗΣΗΣ:

Έστω οτι ο διαχειριστής με username “alex” και password “123” θέλει να εισέλθει στην γραφική διεπαφή της βάσης, να προστέσει έναν νέο εργαζόμενο, να δημιουργήσει μια αίτηση του εργαζόμενου για μία θέση εργασίας και να δει έαν ο εργαζόμενος θα πάρει την θέση ή όχι.

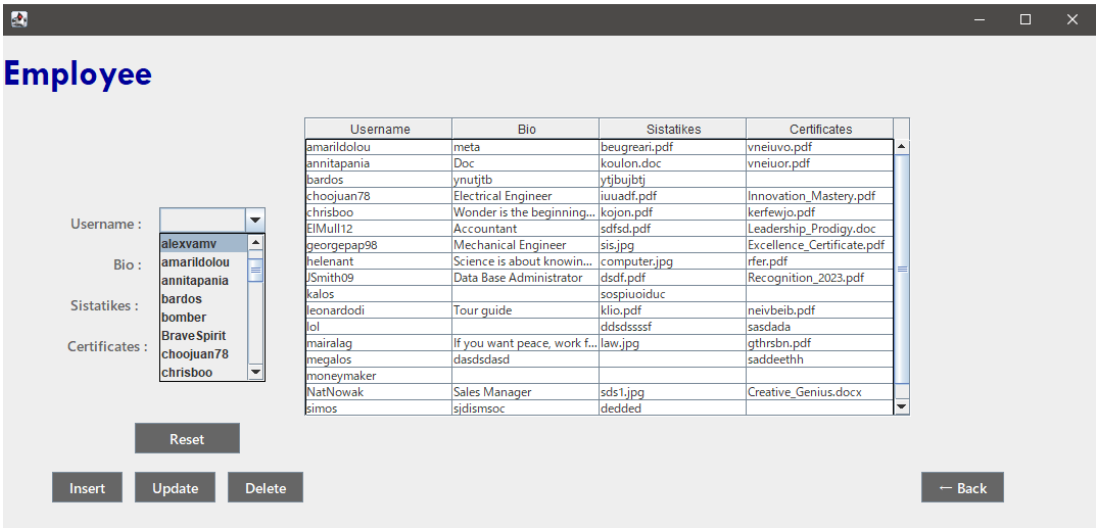
- Αρχικά, λοίπον, ο διαχειριστής εισάγει τα στοιχεία του στο παράθυρο login της βάσης, ταυτοποιείται και εισέρχεται στο βασικό μενού.



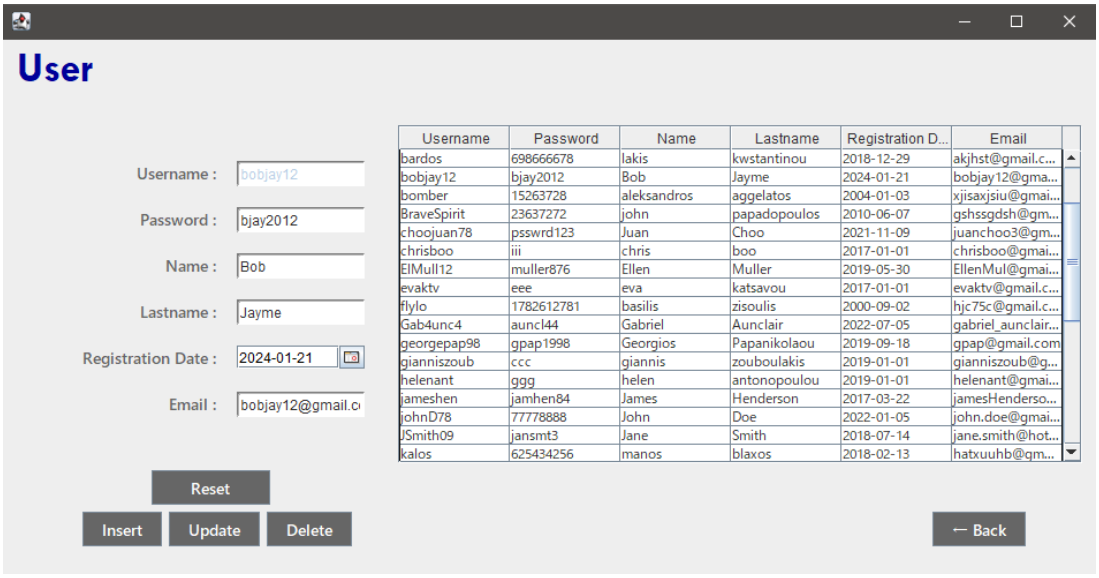
- Ύστερα, επιλέγει τον πίνακα Employee και μετά πατάει το κουμπί Select.



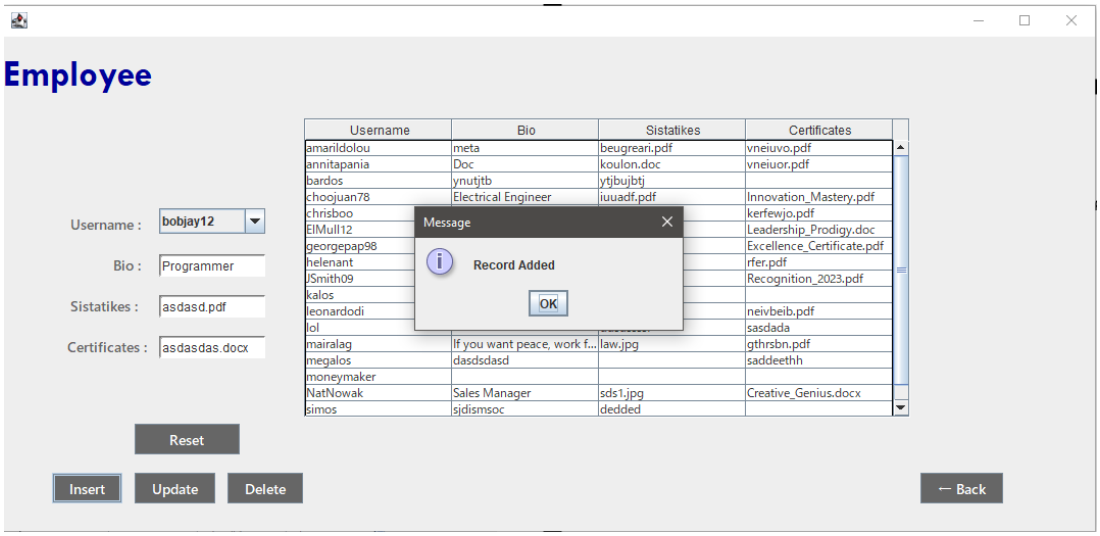
- Ο νέος εργαζόμενος που θέλει να δημιουργήσει θα έχει username : bobjay12, παρ'ολ'αυτά παρατήρει οτι δεν μπορεί να πληκτρολογήσει όποιο username θέλει στο πρώτο πεδίο της διεπαφής. Αυτό συμβαίνει, διότι το πεδίο username είναι ξένο κλειδί στον πίνακα User και η επιλογή της τιμής του γίνεται μέσα απο μενού επιλόγων, όπως αναφέρθηκε και νωρίτερα. Επομένως, θα πρέπει πρώτα να δημιουργήσει καινούργιο χρήστη στην βάση, άρα πατάει το κουμπί Back για να επιστρέψει στο μενού επιλογής πινάκων.



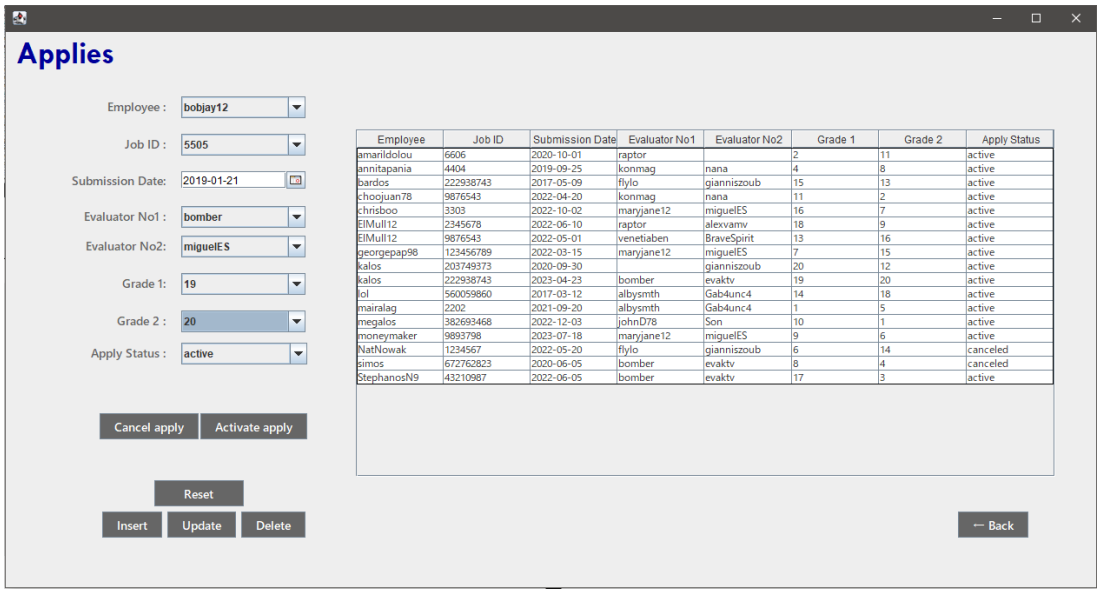
- Στον πίνακα User, τώρα θα δημιουργήσει μια νέα εγγραφή με τα στοιχεία που θέλει.



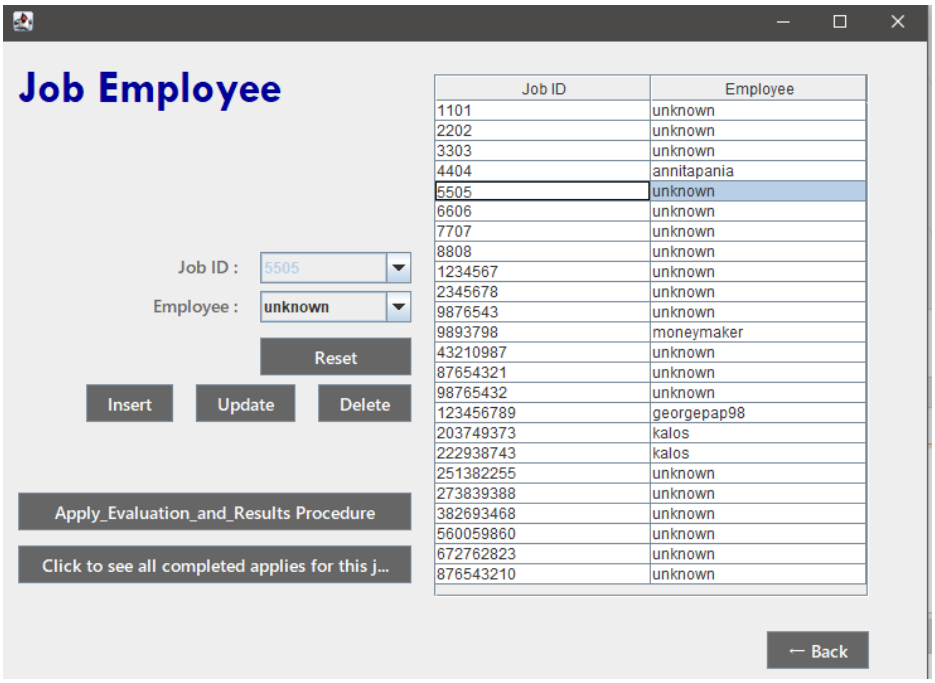
- Τώρα μπορεί να πάει στον πίνακα Employee και να δημιουργήσει τον εργαζόμενο.



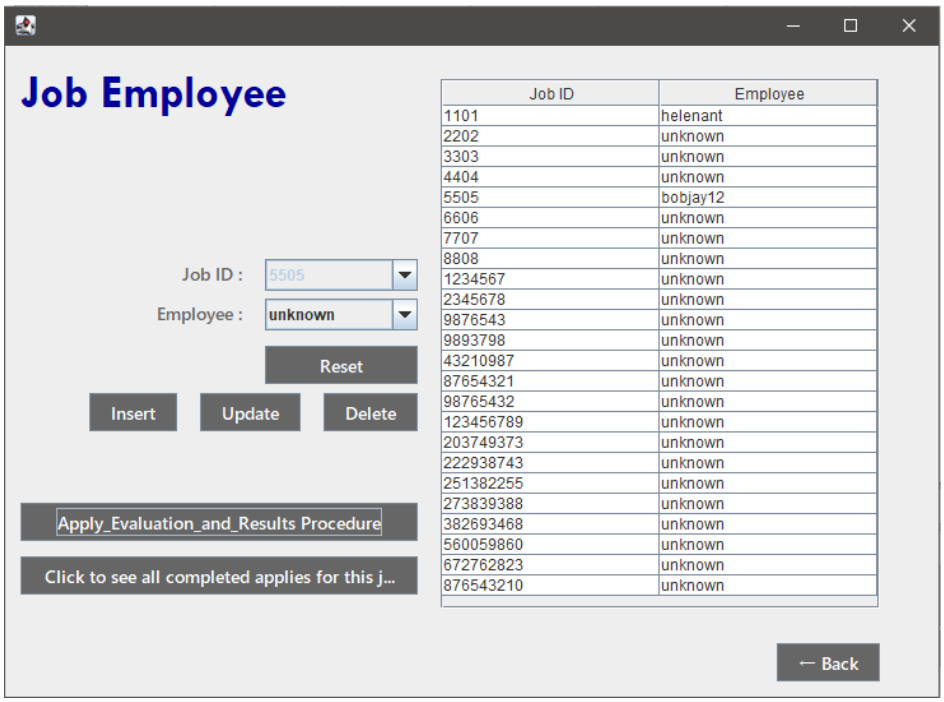
- Έπειτα, πατάει το κουμπί Back, επιστρέφει στο μενού επιλογής πινάκων και διαλέγει τον πίνακα Applies, ώστε να δημιουργήσει την καινούργια αίτηση.



- Ύστερα, επιστρέφει ξανά και διαλέγει το πίνακα Job Employee και στο παράθυρο αυτού του πίνακα επιλέγει το Job ID της θέσης που έκανε την αίτηση.



- Τέλος, κλικάρει στο κουμπί Apply_Evaluation_and_Results, και ο πίνακας ενημερώνεται με το όνομα του εργαζομένου που πήρε την θέση.Τελικά, ο εργαζόμενος που δημιούργησε πήρε την θέση !



- Έπειτα, αν θέλει μπόρει να πατήσει το κουμπί Click to see all completed applies for this job και να δει ποιοι συμμετείχαν στην αξιολόγηση.