

Αναφορά Προτζεκτ Βασεις Δεδομενων 2023-24

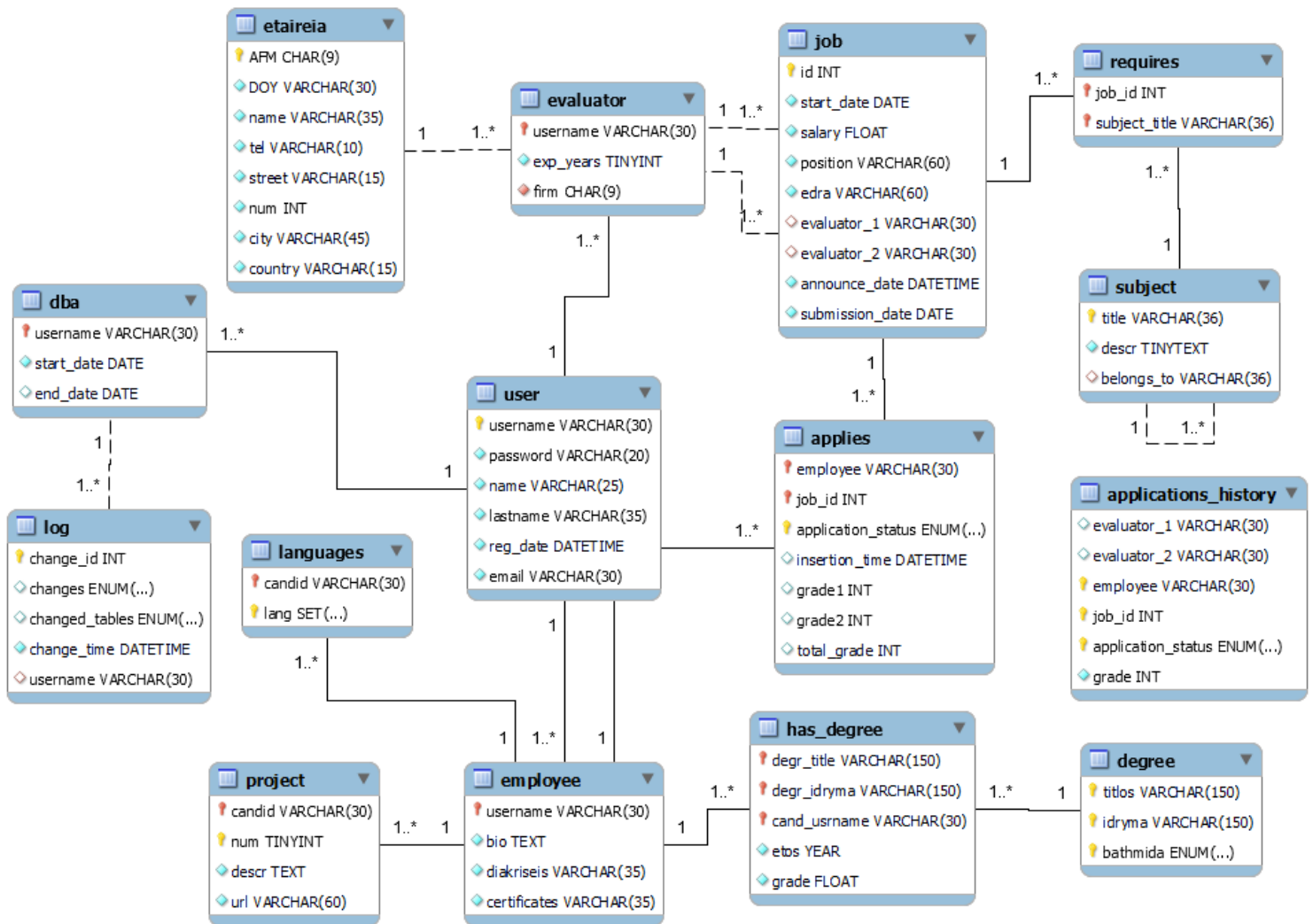
Άγγελος Μενεγατος, 1093426, 3ο έτος
Παντελής Φλουρής, 1093507, 3ο έτος

Περιεχόμενα

Κεφάλαιο 1:.....	2
Σχεσιακό διάγραμμα της συνολικής.....	2
αναθεωρημένης ΒΔ.....	2
αναθεωρημένης ΒΔ:.....	2
Περιγραφή:.....	3
Εντολές MySQL για τη δημιουργία της ΒΔ:.....	4
Κεφάλαιο 2:.....	19
Περιγραφή:.....	19
Παραδείγματα από την εκτέλεση των stored procedures:.....	21
Κώδικας Stored procedures.....	26
Κεφάλαιο 3:.....	36
Περιγραφή:.....	36
Παραδείγματα από την εκτέλεση των triggers:.....	37
Κώδικας triggers:.....	41
Κεφάλαιο 4.....	47
Περιγραφή.....	47
Σενάριο Χρήσης.....	51

Κεφάλαιο 1:

Σχεσιακό διάγραμμα της συνολικής
αναθεωρημένης ΒΔ



Παραδοχές:

Υποθέσαμε ότι οι ίδιοι δυο αξιολογητές αξιολογούν όλες τις αιτήσεις για μια θέση εργασίας.

Κατασκευαστικές επιλογές:

Επιλέξαμε να κάνουμε drop τους πίνακες όταν κάναμε κάποια αλλαγή, αντί του ALTER TABLE, λόγω της πιο οργανωμένης εμφάνισης του κώδικα, και την μικρή έκταση της βάσης μας που το επέτρεπε.

Περιγραφή:

3.1.2.1

Για αυτό το ερώτημα, αρχικά δημιουργήσαμε έναν πίνακα. Έπειτα συμπεράναμε ότι ήταν αρκετή η επέκταση του πίνακα `applies`, έτσι ώστε να περιέχει την κατάσταση της αίτησης. Τα υπόλοιπα ζητούμενα της απαίτησης καλύπτονται μέσω `triggers` και `procedures` που θα δούμε παρακάτω.

3.1.2.2

Σε αυτή την απαίτηση, σε πρώτη φάση επεκτείνουμε τον πίνακα `job` έτσι ώστε να περιέχει δυο αξιολογητές οι οποίοι αξιολογούν την κάθε θέση εργασίας. Ακόμη, έπρεπε να επεκτείνουμε τον πίνακα `applies` έτσι ώστε να περιέχει την ώρα που έγινε η αίτηση (`insertion_time`), με default τιμή `NOW()`, έτσι ώστε να αποθηκεύεται αυτόματα η στιγμή που γίνεται η αίτηση. Έπρεπε να επεκτείνουμε περεταίρω τον πίνακα βάζοντας τρεις βαθμολογίες (`grade1`, `grade2`, `total_grade`), στις οποίες αποθηκεύονται οι βαθμολογίες του πρώτου και του δεύτερου αξιολογητή, και ο συνολικός βαθμός για αυτή την αίτηση αντίστοιχα. Τα `grade1` και `grade2` έχουν default τιμή το -1 για λογούς ευκολίας στην υλοποίηση των επόμενων ερωτημάτων.

3.1.2.3

Εδώ απλά φτιάξαμε έναν επιπλέον πίνακα, τον `applications_history`, με τις κατάλληλες στήλες. Έπειτα βρήκαμε μια σελίδα που φτιάχνει `csv` αρχεία και φτιάξαμε ένα αρχείο με 60.000 εγγραφές, όπως λέει η εκφώνηση. Στην πορεία, φτιάξαμε άλλο ένα τέτοιο αρχείο 100.000 εγγραφών, για να φανεί η αποδοτικότητα των `indexes`. Μεγάλο πρόβλημα αντιμετωπίσαμε στο πώς θα διαβάσει τα αρχεία αυτά το πρόγραμμά μας. Καταλήξαμε, ότι το πιο απλό είναι να βάλουμε τα αρχεία αυτά στο `directory` που αποθηκεύει η `MySQL` την βάση δεδομένων μας. Τέλος με ένα απλό `LOAD DATA IN FILE` διαβάσαμε όλα τα δεδομένα.

3.1.2.4

Για την υλοποίηση αυτού του ζητήματος φτιάξαμε έναν νέο πίνακα, τον `dba`, ο οποίος είναι μια νέα κατηγορία `user`. Μετά τις εισαγωγές σε αυτόν τον πίνακα, πήγαμε χειροκίνητα και για τον καθένα από αυτούς τρέξαμε τις εντολές `create user`, και `grant all privileges`, έτσι ώστε να αναγνωρίζονται όντως ως `dba` από την `MySQL`. Η σκέψη μας αρχικά ήταν αυτό να μην γίνεται χειροκίνητα, αλλά αυτόματα με ένα `trigger`. Η σκέψη αυτή δεν δούλεψε καθώς προκύπταν προβλήματα, τα οποία δεν ξέραμε πως να λύσουμε. Τέλος φτιάξαμε και τον πίνακα `log`, στον οποίο θα αποθηκεύουμε ποιοι χρήστες κάνουν αλλαγές, αλλά για χάριν ευκολίας δεν θα αποθηκεύουμε ούτε την κατάσταση του πίνακα πριν την αλλαγή, ούτε τον τροποποιημένο πίνακα. Το ζητούμενο `trigger` υλοποιήθηκε ταυτόχρονα με αυτό το ερώτημα, αλλά η λειτουργία του θα περιγραφεί παρακάτω.

Εντολές MySQL για τη δημιουργία της ΒΔ:

```
CREATE DATABASE IF NOT EXISTS proparaskeuastiko;
```

```
USE proparaskeuastiko;
```

```
-- Tables with constraints
```

```
DROP TABLE IF EXISTS applications_history;
```

```
DROP TABLE IF EXISTS requires;
```

```
DROP TABLE IF EXISTS applies;
```

```
DROP TABLE IF EXISTS project;
```

```
DROP TABLE IF EXISTS languages;
```

```
DROP TABLE IF EXISTS has_degree;
```

```
DROP TABLE IF EXISTS employee;
```

```
DROP TABLE IF EXISTS applications_history;
```

```
-- Tables with no constraints
```

```
DROP TABLE IF EXISTS job;
```

```
DROP TABLE IF EXISTS log;
```

```
DROP TABLE IF EXISTS dba;
```

```
DROP TABLE IF EXISTS degree;
```

```
DROP TABLE IF EXISTS subject;
```

```
DROP TABLE IF EXISTS evaluator;
```

```
DROP TABLE IF EXISTS etaireia;
```

```
DROP TABLE IF EXISTS user;
```

```
CREATE TABLE IF NOT EXISTS etaireia(
```

```
AFM char(9) NOT NULL,
```

```
DOY varchar(30) DEFAULT 'unknown' NOT NULL,
```

```
name varchar(35) DEFAULT 'unknown' NOT NULL,
```

```
tel varchar(10) DEFAULT 'unknown' NOT NULL,
```

```
street varchar(15) DEFAULT 'unknown' NOT NULL,
```

```
num int(11) DEFAULT '0' NOT NULL,
```

```
city varchar(45) DEFAULT 'unknown' NOT NULL,  
country varchar(15) DEFAULT 'unknown' NOT NULL,
```

```
PRIMARY KEY (AFM),
```

```
UNIQUE (tel)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS user(  
username varchar(30) NOT NULL,
```

```
password varchar(20) DEFAULT 'unknown' NOT NULL,
```

```
name varchar(25) DEFAULT 'unknown' NOT NULL,
```

```
lastname varchar(35) DEFAULT 'unknown' NOT NULL,
```

```
last_name varchar(35) DEFAULT 'unknown' NOT NULL,
```

```
reg_date datetime NOT NULL,
```

```
email varchar(30) DEFAULT 'unknown' NOT NULL,
```

```
PRIMARY KEY (username)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS evaluator(  
username varchar(30) NOT NULL,
```

```
exp_years tinyint(4) DEFAULT '0' NOT NULL,
```

```
firm char(9) DEFAULT 'unknown' NOT NULL,
```

```
PRIMARY KEY (username),
```

```
CONSTRAINT username_con_1
```

```
FOREIGN KEY (firm)
```

```
REFERENCES etaireia(AFM)
```

```
ON DELETE CASCADE ON UPDATE CASCADE,
```

```
CONSTRAINT username_con_2
```

```
FOREIGN KEY (username)
```

```
REFERENCES user(username)
```

```
ON DELETE CASCADE ON UPDATE CASCADE
```

```
);
```

```
CREATE TABLE IF NOT EXISTS employee(  
    username varchar(30) NOT NULL,  
    bio text NOT NULL,  
    diakriseis varchar(35) DEFAULT 'none' NOT NULL,  
    certificates varchar(35) DEFAULT 'none' NOT NULL,  
  
    PRIMARY KEY(username),  
    CONSTRAINT employ_con  
    FOREIGN KEY (username)  
    REFERENCES user(username)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS languages(  
    candid varchar(30) DEFAULT 'unknown' NOT NULL,  
    lang set('EN', 'FR', 'SP', 'GE', 'CH', 'GR') ,  
  
    PRIMARY KEY (candid,lang),  
    CONSTRAINT languages_con  
    FOREIGN KEY(candid)  
    REFERENCES employee(username)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS project(  
    candid varchar(30) DEFAULT 'unknown' NOT NULL,  
    num tinyint(4) NOT NULL,  
    descr text NOT NULL,  
    url varchar(60) DEFAULT 'unknown' NOT NULL,  
  
    PRIMARY KEY(candid,num),  
    CONSTRAINT project_con  
    FOREIGN KEY(candid)  
    REFERENCES employee(username)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS degree(  

```

```
titlos varchar(150) DEFAULT 'unknown' NOT NULL,  
idryma varchar(150) DEFAULT 'unknown' NOT NULL,  
bathmida enum('BSc', 'MSc', 'PhD'),
```

```
PRIMARY KEY( titlos, idryma,bathmida)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS has_degree(  
degr_title varchar(150) DEFAULT 'unknown' NOT NULL,  
degr_idryma varchar(150) DEFAULT 'unknown' NOT NULL,  
cand_username varchar(30) DEFAULT 'unknown' NOT NULL,  
etos year(4) NOT NULL,  
grade float DEFAULT '0' NOT NULL,
```

```
PRIMARY KEY(degr_title, degr_idryma, cand_username),
```

```
CONSTRAINT has_degree_con_1
```

```
FOREIGN KEY (cand_username)
```

```
REFERENCES employee(username)
```

```
ON DELETE CASCADE ON UPDATE CASCADE,
```

```
CONSTRAINT has_degree_con_2
```

```
FOREIGN KEY(degr_title, degr_idryma)
```

```
REFERENCES degree(titlos, idryma)
```

```
ON DELETE CASCADE ON UPDATE CASCADE
```

```
);
```

```
CREATE TABLE IF NOT EXISTS job(  
id int(11) NOT NULL AUTO_INCREMENT,
```

```
start_date date NOT NULL,
```

```
salary float DEFAULT '0' NOT NULL,
```

```
position varchar(60) DEFAULT 'unknown' NOT NULL,
```

```
edra varchar(60) DEFAULT 'unknown' NOT NULL,
```

```
evaluator_1 varchar(30) DEFAULT NULL,
```

```
evaluator_2 varchar(30) DEFAULT NULL,
```

```
announce_date datetime NOT NULL,
```

```
submission_date date NOT NULL,
```

```
PRIMARY KEY(id),
```

```
CONSTRAINT job_con
```

```
FOREIGN KEY(evaluator_1)
```

```
REFERENCES evaluator(username)
```



```
ON DELETE CASCADE ON UPDATE CASCADE,  
  
CONSTRAINT job_con2  
  
FOREIGN KEY(evaluator_2)  
  
REFERENCES evaluator(username)  
  
ON DELETE CASCADE ON UPDATE CASCADE  
  
);
```

```
CREATE TABLE IF NOT EXISTS applies(  
  
employee varchar(30) NOT NULL,  
  
job_id int(11) NOT NULL,  
  
application_status ENUM ('active', 'canceled', 'finished') DEFAULT 'active',  
insertion_time DATETIME DEFAULT NOW(),  
  
grade1 int DEFAULT -1,  
grade2 int DEFAULT -1,  
total_grade int DEFAULT 0,
```

```
  
PRIMARY KEY(employee, job_id,application_status),  
  
CONSTRAINT applies_con_1  
  
FOREIGN KEY (employee)  
  
REFERENCES employee(username)  
  
ON DELETE CASCADE ON UPDATE CASCADE,  
  
CONSTRAINT applies_con_2  
  
FOREIGN KEY(job_id)  
  
REFERENCES job(id)  
  
ON DELETE CASCADE ON UPDATE CASCADE  
  
);
```

```
CREATE TABLE IF NOT EXISTS subject(  
  
title varchar(36) DEFAULT 'unknown' NOT NULL,  
  
descr tinytext NOT NULL,  
  
belongs_to varchar(36) DEFAULT 'unknown',  
  
#katakash ENUM ('active', 'canceled', 'finished'),
```

```
  
PRIMARY KEY(title),  
  
CONSTRAINT subject_con  
  
FOREIGN KEY (belongs_to)  
  
REFERENCES subject(title)  
  
ON DELETE CASCADE ON UPDATE CASCADE
```

);

```
CREATE TABLE IF NOT EXISTS requires(  
  job_id int(11) NOT NULL,  
  subject_title varchar(36) DEFAULT 'unknown' NOT NULL,  
  
  PRIMARY KEY (job_id, subject_title),  
  CONSTRAINT requires_con_1  
  FOREIGN KEY (job_id)  
  REFERENCES job(id)  
  ON DELETE CASCADE ON UPDATE CASCADE,  
  CONSTRAINT requires_con_2  
  FOREIGN KEY (subject_title)  
  REFERENCES subject(title)  
  ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS applications_history(  
  evaluator_1 varchar(30),  
  evaluator_2 varchar(30),  
  employee varchar(30) NOT NULL,  
  job_id int(11) NOT NULL,  
  application_status ENUM ('active', 'canceled', 'finished'),  
  grade int DEFAULT '0' NOT NULL,  
  
  PRIMARY KEY (employee, job_id, application_status)  
);
```

```
#DROP INDEX idx_grade ON applications_history;  
#DROP INDEX idx_evaluator_1 ON applications_history;  
#DROP INDEX idx_evaluator_2 ON applications_history;  
#SHOW INDEXES FROM applications_history;
```

```
CREATE INDEX idx_grade On applications_history(grade);  
CREATE INDEX idx_evaluator_1 On applications_history(evaluator_1);  
CREATE INDEX idx_evaluator_2 On applications_history(evaluator_2);
```

```

CREATE TABLE IF NOT EXISTS dba(
username varchar(30) NOT NULL,
start_date date NOT NULL,
end_date date,

PRIMARY KEY (username),
CONSTRAINT dba_con
FOREIGN KEY (username)
REFERENCES user(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

```

```

CREATE TABLE IF NOT EXISTS log(
change_id int(11) NOT NULL AUTO_INCREMENT,
changes ENUM('INSERT', 'UPDATE', 'DELETE'), #allagh se kefalaia
changed_tables ENUM('job', 'user', 'degree'),
change_time datetime NOT NULL,
username varchar(30),

PRIMARY KEY(change_id),
CONSTRAINT log_con
foreign key (username)
REFERENCES dba(username)
ON DELETE SET NULL ON UPDATE CASCADE
);

```

--Insert-----

```

INSERT INTO etaireia (AFM, DOY, name, tel, street, num, city,country) VALUES
('111111111', 'Δ.Ο.Υ. Α ΠΑΤΡΩΝ', 'h gonia tou mpamph', '2610422055', 'konstantinou', 25, 'patras', 'Greece'),
('222222222', 'Δ.Ο.Υ. Α ΠΑΤΡΩΝ', 'geush', '2610451397', 'stratioth', 40, 'patras', 'Greece'),
('333333333', 'Δ.Ο.Υ. Α ΑΘΗΝΑΣ', 'crust', '2731098989', 'amerikhs', 107, 'athens', 'Greece'),
('444444444', 'Δ.Ο.Υ. Β ΑΘΗΝΑΣ', 'lux', '2731076543', 'agiou', 303, 'athens', 'Greece'),
('555555555', 'Δ.Ο.Υ. Γ ΓΕΡΜΑΝΙΑΣ', 'bruchten', '355567665', 'deuchspiel', 55, 'berlin', 'Germany'),

```

```
('666666666', 'D.O.Y. G SPAIN', 'the amazest hotel', '4653422436', 'elf street', 299, 'las palmas', 'Spain');
```

```
# select * from etaireia;
```

```
INSERT INTO etaireia VALUE
```

```
(777777777,'Δ.O.Y. ΗΠΑΚΛΕΙΟΥ','RUNG','2810334006','SILICON VALLEY',11,'HERAKLION','GREECE'),
```

```
(888888888,'Δ.O.Y. Α ΠΑΤΡΩΝ','YOUCHEWB','2610987665','EL. STRATIOTOU',33,'PATRAS','GREECE'),
```

```
(999999999,'Δ.O.Y. Α ΖΑΚΥΝΘΟΥ','LEyesVO','2945033944','KORINTHOU',123,'ZAKINTHOS','GREECE');
```

```
INSERT INTO user (username, password, name, lastname, reg_date, email) VALUES # for evaluator
```

```
('john_doe', 'password123', 'John', 'Doe', '2023-01-01 12:00:00', 'john.doe@example.com'),
```

```
('alice_smith', 'pass456', 'Alice', 'Smith', '2023-02-15 15:30:00', 'alice.smith@example.com'),
```

```
('bob_jones', 'secure789', 'Bob', 'Jones', '2023-03-20 10:45:00', 'bob.jones@example.com'),
```

```
('emma_wilson', 'emma123', 'Emma', 'Wilson', '2023-04-05 08:15:00', 'emma.wilson@example.com'),
```

```
('alex_miller', 'pass123', 'Alex', 'Miller', '2023-05-10 17:00:00', 'alex.miller@example.com'),
```

```
('sara_jackson', 'sara456', 'Sara', 'Jackson', '2023-06-25 14:20:00', 'sara.jackson@example.com');
```

```
#select * from user;
```

```
INSERT INTO user VALUES
```

```
('johnnyboy','1234','John','Adams','2023-11-14','johnny420@gmail.com'),
```

```
('jacko','4321','Jack','Jones','2021-01-14 12:11:59','jackOlantern@gmail.com'),
```

```
('Mucas1940','Di6nee','Amanda','Peters','2020-11-11 14:02:02','AmandaLPeters@armyspy.com'),
```

```
('Ancingingin','zah0Keg2c','Terry','Dodd','2023-12-05 17:05:02','TerryLDodd@armyspy.com'),
```

```
('Alke1996','taeK3NieSh','Milton','McSherry','2021-02-11 14:13:55','MiltonSMcSherry@dayrep.com'),
```

```
('DreamyCoder', 'SecurePass789', 'Emma', 'Johnson', '1989-03-15', 'emma.johnson@example.com'),
```

```
('Fortune','biJ8lue7lah','Thomas','Martin','2019-11-26 20:45:34','ThomasTMartin@armyspy.com'),
```

```
('Yeepy','Eishi2Achae','Earl','Martin','2015-10-11 09:14:16','EarlRMartin@rhyta.com'),
```

```
('Pilve1984','naiCh6uph7','Martin','Gordon','2009-05-25 14:14:14','MadelineGordon@jourrapide.com'),
```

```
('Tord2003','YeeK0mi4','George','Alexander','2022-03-15 15:04:29','GeorgeAlexander@armyspy.com'),
```

```
('Alte1970','Beihoa3xoh','keir','Munro','2021-01-21 09:16:00','KeirMunro@rhyta.com'),
```

```
('Hunitesig','Hephae7ai','Drew','Moore','2020-06-30 16:30:46','DrewMoore@rhyta.com');
```

```
INSERT INTO user (username, password, name, lastname, reg_date, email) VALUES #for employee
```

```
('mark_smith', 'mark789', 'Mark', 'Smith', '2023-07-10 09:30:00', 'mark.smith@example.com'),
```

```
('lisa_jones', 'lisa456', 'Lisa', 'Jones', '2023-08-15 13:45:00', 'lisa.jones@example.com'),
```

```
('kevin_davis', 'kevin123', 'Kevin', 'Davis', '2023-09-20 18:00:00', 'kevin.davis@example.com'),
('natalie_white', 'natalie456', 'Natalie', 'White', '2023-10-05 11:15:00', 'natalie.white@example.com'),
('steve_martin', 'steve789', 'Steve', 'Martin', '2023-11-10 14:45:00', 'steve.martin@example.com'),
('emily_wilson', 'emily123', 'Emily', 'Wilson', '2023-12-25 20:30:00', 'emily.wilson@example.com');
```

```
INSERT INTO user (username, password, name, lastname, reg_date, email) VALUES      # for dba
('mark_anderson', 'markpass123', 'Mark', 'Anderson', '2023-07-12 09:30:00', 'mark.anderson@example.com'),
('laura_white', 'laura789', 'Laura', 'White', '2023-08-18 13:45:00', 'laura.white@example.com'),
('kevin_martin', 'kevinpass', 'Kevin', 'Martin', '2023-09-22 11:00:00', 'kevin.martin@example.com'),
('natalie_green', 'natalie123', 'Natalie', 'Green', '2023-10-30 16:15:00', 'natalie.green@example.com'),
('michael_turner', 'mike456', 'Michael', 'Turner', '2023-11-05 20:00:00', 'michael.turner@example.com'),
('olivia_carter', 'olivia123', 'Olivia', 'Carter', '2023-12-15 18:30:00', 'olivia.carter@example.com');
#select * from user;
```

```
INSERT INTO evaluator (username, exp_years, firm) VALUES
('john_doe', 3, '111111111'),
('alice_smith', 5, '222222222'),
('bob_jones', 2, '333333333'),
('emma_wilson', 4, '444444444'),
('alex_miller', 6, '555555555'),
('sara_jackson', 1, '666666666');
```

```
INSERT INTO evaluator VALUE
('Ancingingeng',1,'777777777'),
('johnnyboy',3,'888888888'),
('jacko',1,'999999999'),
('Mucas1940',6,'888888888'),
('Alke1996',2,'777777777'),
('DreamyCoder',11,'888888888');
```

```
INSERT INTO employee (username, bio, diakriseis, certificates) VALUES
('mark_smith', 'Experienced IT professional with a focus on technical support.', 'mark_smith_diakriseis.pdf',
'mark_smith_cert.pdf'),
('lisa_jones', 'Dedicated finance professional with expertise in financial analysis.', 'lisa_jones_diakriseis.pdf',
'lisa_jones_cert.pdf'),
```

```
('kevin_davis', 'Detail-oriented individual with skills in manufacturing processes.', 'kevin_davis_diakriseis.pdf',
'kevin_davis_cert.pdf'),

('natalie_white', 'Compassionate healthcare professional committed to patient care.', 'natalie_white_diakriseis.pdf',
'natalie_white_cert.pdf'),

('steve_martin', 'Creative marketing specialist with a proven track record of successful campaigns.', 'steve_martin_diakriseis.pdf',
'steve_martin_cert.pdf'),

('emily_wilson', 'Passionate educator fostering a positive learning environment.', 'emily_wilson_diakriseis.pdf',
'emily_wilson_cert.pdf');
```

INSERT INTO employee VALUES

```
('Fortune', 'Software Engineer', 'Fortune_diakriseis.pdf', 'Fortune_cert.pdf'),
('Yeepy', 'Digital Marketer', 'Yeepy_diakriseis.pdf', 'Yeepy_cert.pdf'),
('Pilve1984', 'Graphic Designer', 'Pilve1984_diakriseis.pdf', 'Pilve1984_cert.pdf'),
('Tord2003', 'UI/UX Designer', 'Tord2003_diakriseis.pdf', 'Tord2003_cert.pdf'),
('Alte1970', 'Front-end Developer', 'Alte1970_diakriseis.pdf', 'Alte1970_cert.pdf'),
('Hunitesige', 'Web Developer', 'Hunitesige_diakriseis.pdf', 'Hunitesige_cert.pdf');
```

INSERT INTO languages (candid, lang) VALUES

```
('mark_smith', 'EN,FR'),
('lisa_jones', 'SP'),
('kevin_davis', 'EN,CH'),
('natalie_white', 'EN'),
('steve_martin', 'FR,GE'),
('emily_wilson', 'SP,CH');
```

INSERT INTO languages VALUE

```
('Fortune', 'GR,EN'),
('Yeepy', 'FR,SP,GR'),
('Pilve1984', 'EN,CH,GR'),
('Alte1970', 'FR,SP,GR,EN');
```

INSERT INTO project (candid, num, descr, url) VALUES

```
('mark_smith', 1, 'IT Support System Upgrade', 'https://example.com/project1'),
('lisa_jones', 1, 'Financial Analysis Tool Development', 'https://example.com/project2'),
('kevin_davis', 1, 'Manufacturing Process Optimization', 'https://example.com/project3'),
('natalie_white', 1, 'Patient Care App Implementation', 'https://example.com/project4'),
('steve_martin', 1, 'Marketing Campaign for Product Launch', 'https://example.com/project5'),
```

```
('emily_wilson', 1, 'Educational Platform Enhancement', 'https://example.com/project6'),
('lisa_jones', 1, 'mysql projecgt', 'https://example.com/project7');
```

```
INSERT INTO project VALUES
```

```
('Hunitesige',1,'Web Development for XYZ Corp','www.xyzcorp.com'),
('Hunitesige',2,'Mobile App Design for ABC Inc','www.abcinc.com'),
('Hunitesige',3,'Data Analysis for DEF Company','www.defcompany.com'),
('Tord2003',1,'UI/UX Redesign for GHI Industries','www.ghiindustries.com'),
('Yeepy',1,'Marketing Campaign for JKL Enterprises','www.jklenterprises.com'),
('Fortune',1,'Software Development for MNO Ltd','www.mnold.com'),
('Pilve1984',1,'Graphic Design for PQR Co.','www.pqrco.com');
```

```
INSERT INTO job (id, start_date, salary, position, edra, evaluator_1, evaluator_2, announce_date, submission_date) VALUES
(NULL, '2025-01-15', 50000.00, 'IT Support Specialist', 'Athens', 'john_doe', 'alice_smith', '2025-01-05 08:00:00', '2025-02-01'),
(NULL, '2025-02-20', 60000.00, 'Financial Analyst', 'Paris', 'alice_smith', 'bob_jones', '2025-02-10 10:30:00', '2025-03-01'),
(NULL, '2025-03-25', 70000.00, 'Production Coordinator', 'Berlin', 'bob_jones', 'emma_wilson', '2025-03-15 12:45:00', '2025-04-01'),
(NULL, '2025-04-10', 55000.00, 'Registered Nurse', 'Madrid', 'emma_wilson', 'alex_miller', '2025-04-01 15:15:00', '2025-05-01'),
(NULL, '2025-05-15', 65000.00, 'Marketing Specialist', 'London', 'alex_miller', 'sara_jackson', '2025-05-05 17:30:00', '2025-06-01'),
(NULL, '2025-06-20', 75000.00, 'Teacher', 'Barcelona', 'sara_jackson', 'john_doe', '2025-06-10 20:45:00', '2025-07-01'),
(NULL, '2025-07-15', 80000.00, 'Software Developer', 'Munich', 'john_doe', 'emma_wilson', '2025-07-05 08:30:00', '2025-08-01'),
(NULL, '2025-08-20', 90000.00, 'Data Scientist', 'Amsterdam', 'emma_wilson', 'alex_miller', '2025-08-10 11:00:00', '2025-09-01'),
(DEFAULT, '2025-12-12', 60000.00, 'MECHANICAL ENGINEER', 'PATRAS', 'jacko', 'Ancingen', '2025-12-13 08:15:34', '2025-12-23'),
(DEFAULT, '2025-01-15', 50000.00, 'Software Engineer', 'New York', 'Ancingen', 'johnnyboy', '2025-01-10 08:00:00', '2025-02-28'),
(DEFAULT, '2025-02-20', 60000.00, 'Data Analyst', 'California', 'johnnyboy', 'jacko', '2025-02-15 10:30:00', '2025-04-15'),
(DEFAULT, '2025-03-10', 70000.00, 'Marketing Manager', 'Texas', 'jacko', 'Mucas1940', '2025-03-05 12:45:00', '2025-05-30'),
(DEFAULT, '2025-04-05', 55000.00, 'Financial Advisor', 'Florida', 'Mucas1940', 'Alke1996', '2025-03-30 09:15:00', '2025-06-10'),
(DEFAULT, '2025-05-22', 65000.00, 'HR Specialist', 'Washington', 'Alke1996', 'DreamyCoder', '2025-05-15 11:00:00', '2025-07-20'),
```

(DEFAULT, '2025-06-30', 75000.00, 'Project Manager', 'Illinois', 'DreamyCoder', 'Ancingen', '2025-06-25 14:20:00', '2025-08-31'),

(DEFAULT, '2025-07-12', 58000.00, 'Sales Representative', 'Arizona', 'Ancingen', 'johnnyboy', '2025-07-05 09:45:00', '2025-09-25');

INSERT INTO degree (titlos, idryma, bathmida) VALUES

('Bachelor of Computer Science and Math', 'University of Athens', 'BSc'),

('Bachelor of Data Science', 'Stanford University', 'BSc'),

('Bachelor of Electrical Engineering', 'Massachusetts Institute of Technology', 'BSc'),

('Bachelor of Mechanical Engineering', 'University of Cambridge', 'BSc'),

('Bachelor of Computer Science', 'Harvard University', 'BSc'),

('Bachelor of Finance', 'Paris University', 'BSc'),

('Bachelor of Marketing', 'London Business School', 'BSc'),

('Bachelor of Computer Engineering', 'University of Patras', 'BSc'),

('Master of Finance', 'Paris University', 'MSc'),

('Master of Marketing', 'London Business School', 'MSc'),

('Master of Electrical Engineering', 'Massachusetts Institute of Technology', 'MSc'),

('Master of Computer Science and Math', 'University of Athens', 'MSc'),

('Master of Computer Science', 'Harvard University', 'MSc'),

('Doctorate in Computer Science and Math', 'University of Athens', 'PhD'),

('Doctorate in Mechanical Engineering', 'University of Cambridge', 'PhD'),

('Doctorate in Computer Science', 'Harvard University', 'PhD');

INSERT INTO has_degree (degr_title, degr_idryma, cand_username, etos, grade) VALUES

('Bachelor of Computer Science and Math', 'University of Athens', 'mark_smith', 2018, 5.0),

('Bachelor of Finance', 'Paris University', 'lisa_jones', 2020, 5.0),

('Bachelor of Electrical Engineering', 'Massachusetts Institute of Technology', 'kevin_davis', 2015, 5.0),

('Bachelor of Marketing', 'London Business School', 'steve_martin', 2022, 5.0),

('Bachelor of Computer Engineering', 'University of Patras', 'Fortune', 2018, 8.45),

('Bachelor of Computer Science', 'Harvard University', 'Yeepy', 2015, 9.5),

('Bachelor of Computer Science', 'Harvard University', 'Pilve1984', 2020, 9.5),

('Bachelor of Computer Science and Math', 'University of Athens', 'Tord2003', 2017, 8.0),

('Bachelor of Computer Science', 'Harvard University', 'Alte1970', 2018, 7.2),

('Master of Finance', 'Paris University', 'lisa_jones', 2020, 7.0),
 ('Master of Marketing', 'London Business School', 'steve_martin', 2022, 6.2),
 ('Master of Computer Science', 'Harvard University', 'Fortune', 2018, 8.45),
 ('Master of Computer Science and Math', 'University of Athens', 'Pilve1984', 2021, 8.9),
 ('Master of Computer Science', 'Harvard University', 'Yeepy', 2019, 6.1),
 ('Master of Electrical Engineering', 'Massachusetts Institute of Technology', 'Alte1970', 2019, 9.2),

 ('Doctorate in Computer Science and Math', 'University of Athens', 'Yeepy', 2022, 8.7),
 ('Doctorate in Computer Science', 'Harvard University', 'Pilve1984', 2016, 7.85),
 ('Doctorate in Mechanical Engineering', 'University of Cambridge', 'Alte1970', 2019, 9.2);

INSERT INTO subject (title, descr, belongs_to) VALUES

('Math', 'Mathematics', NULL),
 ('Algebra', 'Basic algebra concepts', 'Math'),
 ('Geometry', 'Geometry principles', 'Math'), #for check
 ('Calculus', 'Calculus principles', 'Math'),
 ('Physics', 'Study of matter and energy', NULL),
 ('Mechanics', 'Study of motion and forces', 'Physics'),
 ('English', 'basic knowledge', NULL),
 ('Academic Skills', 'basic academic skills', NULL);

INSERT INTO subject (title, descr,belongs_to) VALUES

('Web Development', 'Creating dynamic and responsive websites using modern technologies.',NULL),
 ('JavaScript Fundamentals', 'Fundamental concepts and syntax of JavaScript programming language.',NULL),
 ('CSS Styling', 'Cascading Style Sheets for web page styling and design.',NULL);

INSERT INTO subject (title, descr, belongs_to) VALUES

('HTML Basics', 'Introduction to HTML tags, structure, and elements.', 'Web Development'),
 ('React Framework', 'Building interactive UIs using the React library.', 'JavaScript Fundamentals'),
 ('Node.js Backend', 'Building server-side applications using Node.js.', 'JavaScript Fundamentals'),
 ('Database Management', 'Introduction to SQL, database design, and management.', 'Web Development'),

```
('Responsive Design', 'Optimizing websites for various devices and screen sizes.', 'CSS Styling');
```

```
INSERT INTO requires (job_id, subject_title) VALUES
```

```
(1, 'Math'),  
(1, 'Physics'),  
(2, 'Math'),  
(2, 'Algebra'),  
(2, 'Mechanics'),  
(3, 'Geometry'),      #for check  
(1, 'Algebra'),  
(2, 'Physics'),  
(4, 'Mechanics');
```

```
INSERT INTO requires VALUES
```

```
(1, 'Web Development'),  
(2, 'HTML Basics'),  
(3, 'CSS Styling'),  
(4, 'JavaScript Fundamentals'),  
(5, 'React Framework'),  
(6, 'Node.js Backend'),  
(7, 'Database Management'),  
(8, 'Responsive Design');
```

```
INSERT INTO dba (username, start_date, end_date) VALUES
```

```
('mark_anderson', '2023-07-12', NULL),  
( 'laura_white', '2023-08-18', NULL),  
( 'kevin_martin', '2023-09-22', NULL),  
( 'natalie_green', '2023-10-30', NULL),  
( 'michael_turner', '2023-11-05', NULL),  
( 'olivia_carter', '2023-12-15', NULL);
```

INSERT INTO applies VALUES

```
('mark_smith', 1, DEFAULT, NOW(), 15, 18, DEFAULT),  
('Tord2003', 1, DEFAULT, NOW(), 19, 16, DEFAULT),  
('emily_wilson', 1, DEFAULT, NOW(), 15, 13, DEFAULT),  
('natalie_white', 1, DEFAULT, NOW(), 20, 19, DEFAULT),  
('Alte1970', 1, DEFAULT, NOW(), 15, 17, DEFAULT),  
('lisa_jones', 2, DEFAULT, NOW(), 11, 14, DEFAULT),  
('steve_martin', 2, DEFAULT, NOW(), 16, 15, DEFAULT),  
('Pilve1984', 2, DEFAULT, NOW(), 20, 18, DEFAULT),  
('kevin_davis', 3, DEFAULT, NOW(), 19, 17, DEFAULT),  
('Yeepy', 3, DEFAULT, NOW(), 16, 18, DEFAULT),  
('Alte1970', 4, DEFAULT, NOW(), 13, 14, DEFAULT),  
('Hunitesige', 4, DEFAULT, NOW(), 15, 16, DEFAULT),  
('Hunitesige', 5, DEFAULT, NOW(), 16, 16, DEFAULT),  
('Hunitesige', 6, DEFAULT, NOW(), 12, 12, DEFAULT),  
('Fortune', 6, DEFAULT, NOW(), 11, 12, DEFAULT);
```

Κεφάλαιο 2:

Περιγραφή:

3.1.3.1

Για αυτο το stored procedure χρειαστηκε να φτιαξουμε δυο, ενα βοηθητικο για την αυτοματη βαθμολογηση σε περιπτωση που ο αξιολογητης αξιολογει τον συγκεκριμενο εργαζομενο και δεν εχει καταχωριστεί ο βαθμος αξιολογησης (auto_grading), και ενα για τα υπολοιπα ζητουμενα (evaluators_grade). Θα μπορούσαμε να το υλοποιησουμε και σε ενα, αλλα αυτη η μεθοδος μας διευκολυνε, καθως “καθαρισε” αρκετα τον κωδικα.

Για το auto_grading:

Χρησιμοποιούμε εναν cursor για να ελεγχουμε ολα τα πτυχια που εχει ενας εργαζομενος, και μεσα σε ενα while loop προσθετουμε τους βαθμους που αναλογουν για καθε πτυχιο. Μετα ελεγχουμε ποσα προτζεκτ εχει υλοποιησει ο εργαζομενος, προσθετοντας εναν βαθμο για καθε προτζεκτ, και εαν γνωριζει 2+ γλωσσες, προσθετουμε αλλον ενα.

Για να δουμε ποσες γλωσσες ξερει, βρισκουμε το μηκος της θεσης lang στην εγγραφη και το αφαιρουμε απο το μηκος της θεσης χωρις τα κόμματα . Ετσι, μας μενουν μονο τα κόμματα , οπου εφοσον καθε γλωσσα ειναι διαχωρισμενη με κομμα, ο αριθμος των κομμάτων ειναι ο αριθμος των γλωσσων -1. Αρα προσθετουμε 1 και παιρνουμε τον αριθμο των γλωσσων.

Για το evaluators_grade:

Αποθηκευουμε αρχικά σε μια μεταβλητη την τιμη του grade1 απο τον πινακα applies. Εαν η μεταβλητη μας εχει τιμη NULL μετα την αναθεση (αρα ο evaluator δεν ειναι ο evaluator1), επαναλαμβανουμε την διαδικασια για την τιμη του grade2.

Μετα εαν η τιμη της μεταβλητης ειναι ακομα NULL, ο συγκεκριμενος evaluator δεν αξιολογει τον συγκεκριμενο εργαζομενο στην συγκεκριμενη δουλεια, και επιστρεφεται η τιμη 0.

Αλλιως, εαν η μεταβλητη εχει τιμη -1 (αρα βαθμολογει και δεν εχει καταχωρηθει βαθμολογια) καλειται το procedure `auto_grading` και επιστρεφεται η τιμη του. Τελος, εαν δεν ισχυει κατι απο τα παραπανω σημαινει πως ο αξιολογητης και αξιολογει, και εχει καταχωρησει βαθμολογια, και επιστρεφεται η βαθμολογια του.

3.1.3.2

Για αυτο το ερωτημα δημιουργησαμε ενα procedure (`application_handler`).

Στην περιπτωση που ο χρηστης εχει επιλεξει “i”, αποθηκευονται τα ονοματα των αξιολογητων για αυτη την θεση εργασιας σε δυο μεταβλητες.

Επειτα, ελεγχονται, και εαν καποια μεταβλητη ειναι NULL (αρα δεν εχει συμπληρωθει ο αξιολογητης), επιλεγεται τυχαια ενας αξιολογητης απο την ιδια εταιρεια που ειναι ο αλλος αξιολογητης. Δεν γινεται να μην εχει συμπληρωθει κανενας απο τους δυο αξιολογητες, καθως τουλαχιστον ο ενας απο αυτους ανακοινωνει την θεση εργασιας.

Μετα την τυχον επιλογη αξιολογητη, δημιουργειται μια αιτηση για τον εργαζομενο που μας δινεται.

Στην περιπτωση που ο χρηστης εχει επιλεξει “c”, ελεγχεται εαν υπαρχει εργαζομενος με ενεργη αιτηση για αυτη την θεση εργασιας. Εαν υπαρχει, ενημερωνεται σε ακυρωμενη. Εαν οχι, εκτυπωνεται αντιστοιχο μηνυμα.

Τελος, εαν ο χρηστης εχει επιλεξει “a”, ελεγχεται εαν υπαρχει εργαζομενος με ακυρωμενη αιτηση για αυτη την θεση εργασιας. Εαν υπαρχει, ενημερωνεται σε ενεργη. Εαν οχι, εκτυπωνεται αντιστοιχο μηνυμα.

3.1.3.3

Για αυτο το ερωτημα δημιουργησαμε δυο procedures, ενα βοηθητικο (`final_grading`), και ενα που εκτελει ολες τις ενεργειες και καλει το βοηθητικο (`result_extraction`).

Για το βοηθητικο, ελεγχουμε εαν η αιτηση με τα δεδομενα που παιρνει σαν εισοδο ειναι ακυρωμενη. Εαν ειναι, δεν γινεται τιποτα. Εαν δεν ειναι, σε περιπτωση που δεν εχει βαθμος απο καποιον αξιολογητη, υπολογιζονται με βαση τον πινακα και ενημερωνεται η εγγραφη. Πριν ολοκληρωθει, ανανεωνει τον πινακα ξανα, και ενημερωνει την τελικη βαθμολογια με τον μεσο ορο των βαθμολογιων των δυο αξιολογητων.

Για το `result_extraction`, αρχικα δημιουργουμε ενα cursor για τους εργαζομενους που εχουν κανει αιτηση για την συγκεκριμενη θεση εργασιας.

Για καθε εργαζομενο, αποθηκευουμε ολα τα δεδομενα της αιτησης σε μεταβλητές, καλουμε το βοηθητικο για να παρουμε τα τελικα δεδομενα και τα κανουμε insert στον πινακα applications_history.

Μετα που θα αντιγραφουν ολα τα δεδομενα στον πινακα ιστορικού, θα επιλεξουμε τον εργαζομενο με τον μεγαλυτερο τελικο βαθμο και ενεργη αιτηση, και θα τον αποθηκευσουμε στην μεταβλητη εξοδου. Για να αποφυγουμε τυχον προβληματα σε περιπτωση ισοβαθμιας, επιλεγεται παντα ο εργαζομενος που εκανε την αιτηση πρωτος.

Τελος, διαγραφουμε ολες τις εγγραφες για αυτη την θεση εργασιας απο τον πινακα applies.

3.1.3.4

Η υλοποίηση των δύο αυτών StoredProcedures ήταν τετριμμένη. Στην συνέχεια έφτιαξα indexes για τις στήλες grade, evaluator_1, και evaluator_2. Υπήρξε η σκέψη να φτιαχτεί ένα μόνο σύνθετο indexe για τις στήλες evaluator_1, και evaluator_2, αλλά αυτό δεν συνέβαλλε θετικά στην μείωση του χρόνου εκτελέσεως των StoredProcedures. Για την καταγραφή του χρόνου θα χρησιμοποιήσουμε την εντολή show profiles.

Παραδείγματα από την εκτέλεση των stored procedures:

3.1.3.1

```
select evaluator_1, employee, job_id , grade1
from job
inner join applies
on job_id = id AND employee = 'Alte1970';
```

	evaluator_1	employee	job_id	grade1
▶	john_doe	Alte1970	1	15
	emma_wilson	Alte1970	4	13

```
call evaluators_grade('john_doe', 'Alte1970', 1, @res);
select @res;
```

	@res
▶	15

3.1.3.2

INSERT INTO job VALUES

(NULL, '2025-01-15', 50000.00, 'IT Support Specialist', 'Athens', 'john_doe', DEFAULT, '2025-01-05 08:00:00', '2025-02-01');

select id, evaluator_1, evaluator_2 from job order by id desc limit 1;

	id	evaluator_1	evaluator_2
▶	17	john_doe	NULL
*	NULL	NULL	NULL

select * from applies where employee = "Tord2003";

	employee	job_id	application_status	insertion_time	grade1	grade2	total_grade
▶	Tord2003	1	active	2024-01-21 19:30:44	19	16	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

call application_handler('Tord2003', 17,'i');

select * from applies where employee = "Tord2003";

	employee	job_id	application_status	insertion_time	grade1	grade2	total_grade
▶	Tord2003	1	active	2024-01-21 19:30:44	19	16	0
	Tord2003	17	active	2024-01-21 19:32:12	-1	-1	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

select id, evaluator_1, evaluator_2 from job order by id desc limit 1;

	id	evaluator_1	evaluator_2
▶	17	john_doe	Ancingen
*	NULL	NULL	NULL

3.1.3.3

```
select * from applies where job_id =1;
```

	employee	job_id	application_status	insertion_time	grade1	grade2	total_grade
▶	Alte1970	1	active	2024-01-21 17:05:05	15	17	0
	emily_wilson	1	active	2024-01-21 17:05:05	15	13	0
	mark_smith	1	active	2024-01-21 17:05:05	15	18	0
	natalie_white	1	active	2024-01-21 17:05:05	20	19	0
	Tord2003	1	active	2024-01-21 17:05:05	19	16	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
select * from applications_history WHERE job_id =1;
```

	evaluator_1	evaluator_2	employee	job_id	application_status	grade
▶	Hilda	Tremblay	Luna	1	finished	2
*	NULL	NULL	NULL	NULL	NULL	NULL

```
call result_extraction(1,@res);
```

```
select @res;
```

	@res
▶	natalie_white

```
select * from applies WHERE job_id =1;
```

	employee	job_id	application_status	insertion_time	grade1	grade2	total_grade
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
select * from applications_history WHERE job_id =1;
```

	evaluator_1	evaluator_2	employee	job_id	application_status	grade
▶	john_doe	alice_smith	Alte1970	1	finished	16
	john_doe	alice_smith	emily_wilson	1	finished	14
	Hilda	Tremblay	Luna	1	finished	2
	john_doe	alice_smith	mark_smith	1	finished	17
	john_doe	alice_smith	natalie_white	1	finished	20
	john_doe	alice_smith	Tord2003	1	finished	18
*	NULL	NULL	NULL	NULL	NULL	NULL

3.1.3.4

Χωρίς την χρήση Indexes

show profiles;

	Query_ID	Duration	Query
▶	1	0.00017075	SHOW WARNINGS
CALL search_a(6,13);	2	0.05423700	SELECT employee, job_id FROM applications_hi...
CALL search_b('Edee');	3	0.05389875	SELECT employee, job_id FROM applications_hi...

Με την χρήση Indexes

show profiles;

CALL search_a(6,13);	7	0.02773375	SELECT employee, job_id FROM applications_hi...
CALL search_b('Edee');	8	0.01489725	SELECT employee, job_id FROM applications_hi...

Κώδικας Stored procedures

-- 3.1.3.1 -----

```
DELIMITER $

CREATE PROCEDURE auto_grading(
candidate varchar(30), OUT grade int)

BEGIN

    DECLARE level enum('BSc', 'MSc', 'PhD');

    DECLARE project_num INT DEFAULT 0;

    DECLARE languages INT DEFAULT 0;

    DECLARE flag INT;


    DECLARE bcursor CURSOR FOR

        SELECT bathmida

    FROM degree

    inner join has_degree ON titlos = degr_title AND degr_idryma = idryma

    WHERE cand_username = candidate;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag=1;

    set flag=0;


    set grade = 0;

    OPEN bcursor;


    FETCH bcursor INTO level;


    WHILE(flag=0)

DO

SET grade =

    CASE

        WHEN level = 'BSc' THEN grade + 1

        WHEN level = 'MSc' THEN grade + 2

        ELSE grade + 3

    END;

END;
```

```

        FETCH bcursor INTO level;

END WHILE;


CLOSE bcursor;


        select MAX(num) INTO project_num
FROM project WHERE candid = candidate;
IF project_num IS NULL
THEN
        SET project_num = 0;
END IF;


SELECT LENGTH(lang) - LENGTH(REPLACE(lang, ',', '')) + 1 INTO languages
FROM languages
WHERE candid = candidate;

set grade = grade + project_num;


IF(languages >= 2)
THEN
        set grade = grade + 1;
END IF;
END$
DELIMITER ;


DELIMITER $
CREATE PROCEDURE evaluators_grade(
evaluator varchar(30), employee_username varchar(30), job int, OUT grade int)
BEGIN
        DECLARE grade_out int;

SELECT grade1 INTO grade_out
FROM applies
INNER JOIN job ON job_id = id
WHERE evaluator_1 = evaluator AND employee = employee_username AND job_id = job;

```

```

IF grade_out IS NULL THEN

    SELECT grade2 INTO grade_out

    FROM applies

    INNER JOIN job ON job_id = id

    WHERE evaluator_2 = evaluator AND employee = employee_username AND job_id = job;

END IF;

IF grade_out IS NULL THEN

    SET grade = 0;

ELSEIF grade_out = -1 THEN

    call auto_grading(employee_username, grade);

ELSE

    SET grade = grade_out;

END IF;

END$

DELIMITER ;

-- TEST --

/*

select evaluator_1, employee, job_id , grade1

from job

inner join applies

on job_id = id AND employee = 'Alte1970';

call evaluators_grade('john_doe', 'Alte1970', 1, @res);

select @res;

*/

-- 3.1.3.2 -----

DELIMITER $

CREATE PROCEDURE application_handler(

employee_username varchar(30), job int, operation char(1))

BEGIN

    DECLARE eval1 varchar(30);

    DECLARE eval2 varchar(30);

```

```

declare firm1 int;

IF operation = 'i' THEN

    SELECT evaluator_1, evaluator_2 INTO eval1, eval2

FROM job

WHERE id = job;

IF eval1 IS NULL THEN

    SELECT firm INTO firm1

FROM evaluator

WHERE username = eval2;

    UPDATE job

    SET evaluator_1 = (SELECT evaluator.username FROM evaluator

    WHERE evaluator.firm = firm1 AND username != eval2

    ORDER BY RAND()

    LIMIT 0,1)

WHERE id = job;

    END IF;

SELECT eval2;

    IF eval2 IS NULL THEN

        SELECT firm INTO firm1

FROM evaluator

WHERE username = eval1;

        UPDATE job

        SET evaluator_2 = (SELECT evaluator.username FROM evaluator

        WHERE evaluator.firm = firm1 AND username != eval1

        ORDER BY RAND()

        LIMIT 0,1)

        WHERE id = job;

        END IF;

INSERT INTO applies VALUES(employee_username, job, DEFAULT, NOW(), DEFAULT, DEFAULT,
DEFAULT);

ELSEIF operation = 'c' THEN

    IF EXISTS(SELECT employee FROM applies WHERE employee = employee_username AND job_id =
job AND application_status = 'active') THEN

```

```

UPDATE applies

SET application_status = 'canceled'

WHERE employee = employee_username AND job_id = job AND application_status = 'active';

ELSE

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Employee doesnt have active application or it has already been
canceled';

        END IF;

ELSEIF operation = 'a' THEN

        IF EXISTS(SELECT employee FROM applies WHERE employee = employee_username AND job_id =
job AND application_status = 'canceled') THEN

                UPDATE applies

                SET application_status = 'active'

                WHERE employee = employee_username AND job_id = job AND application_status = 'canceled';

                ELSE

                        SIGNAL SQLSTATE '45000'

                        SET MESSAGE_TEXT = 'The application doesnt exist or employee already has active
application for this position ';

                        END IF;

                ELSE

                        SIGNAL SQLSTATE '45000'

                        SET MESSAGE_TEXT = 'Wrong input! ';

                END IF;

                select 'Operation success! ' ;

ENDS$

DELIMITER ;

/*

-- TEST FOR 3.1.3.2

INSERT INTO job VALUES

(NULL, '2025-01-15', 50000.00, 'IT Support Specialist', 'Athens', 'john_doe', DEFAULT, '2025-01-05 08:00:00', '2025-02-01');

select id, evaluator_1, evaluator_2 from job order by id desc limit 1;

select * from applies where employee = "Tord2003";

call application_handler('Tord2003', 17,'i');

select * from applies where employee = "Tord2003";

```

```

select id, evaluator_1, evaluator_2 from job order by id desc limit 1;

select firm from evaluator where username = "john_doe" OR username = "Ancingen";

call application_handler('Tord2003', 2, 'c');

call application_handler('Tord2003', 2, 'a');


call application_handler('Tord2003', 17, 'c');

select * from applies where employee = "Tord2003";

call application_handler('Tord2003', 17, 'a');

select * from applies where employee = "Tord2003";

*/

```

-- 3.1.3.3 -----

```

DELIMITER $

CREATE PROCEDURE final_grading (
candidate VARCHAR(30), job INT , status ENUM ('active', 'canceled', 'finished'))
BEGIN
    DECLARE grade1_result INT;
        DECLARE grade2_result INT;

select grade1, grade2 INTO grade1_result, grade2_result
FROM applies
WHERE candidate = employee AND job_id = job AND status = application_status;

    if(status != 'canceled')
    THEN
        IF(grade1_result = -1)
        THEN
            CALL auto_grading(candidate, grade1_result);

            UPDATE applies
            SET grade1 = grade1_result
            WHERE candidate = employee AND job = job_id AND application_status = status;
        END IF;

        IF(grade2_result = -1)
        THEN
            CALL auto_grading(candidate, grade2_result);

```



```

        UPDATE applies
        SET grade2 = grade2_result
        WHERE candidate = employee AND job = job_id AND application_status = status;

        END IF;

        UPDATE applies
        SET total_grade = (grade1_result + grade2_result) / 2
        WHERE candidate = employee AND job = job_id AND application_status = status;

        END IF;

END$

DELIMITER ;

/*

select * from applies where job_id =1;
call final_grading("Alte1970",1,"active");

*/

DELIMITER $

CREATE PROCEDURE result_extraction(
job int, OUT Results varchar(30))
BEGIN

    DECLARE flag INT;

    DECLARE candidate varchar(30);

    DECLARE state ENUM ('active', 'canceled', 'finished');

    DECLARE winner varchar(30);

    DECLARE grade INT;

    DECLARE evaluator1 varchar(30);

    DECLARE evaluator2 varchar(30);

    DECLARE bcursor CURSOR FOR

    SELECT employee FROM applies

    where job_id = job;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag=1;

    set flag=0;

    OPEN bcursor;

    FETCH bcursor INTO candidate;

```

```

WHILE (flag=0)
DO
    select application_status into state
    from applies
    where employee = candidate AND job_id = job;

    SELECT evaluator_1, evaluator_2 INTO evaluator1, evaluator2
    FROM job
    WHERE id =job;

    call final_grading(candidate, job, state);

    SELECT total_grade INTO grade
    FROM applies
    WHERE employee = candidate AND job_id = job;

    IF( grade IS NULL ) THEN
        set grade =0;
    END IF;

    INSERT INTO applications_history VALUES
    (evaluator1, evaluator2, candidate, job, 'finished', grade);

    FETCH bcursor INTO candidate;

END WHILE;

CLOSE bcursor;

SELECT employee INTO winner
FROM applies
WHERE total_grade = (
SELECT MAX(total_grade)
FROM applies
WHERE job_id = job AND application_status = 'active'
)

ORDER BY insertion_time DESC limit 0,1;

set Results = winner;

```

```
DELETE FROM applies WHERE job_id = job;

END$

DELIMITER ;
```

```
-- TEST FOR 3.1.3.3

select * from applies WHERE job_id =1;

select * from applications_history WHERE job_id =1;

call result_extraction(1,@res);

select @res;

select * from applies WHERE job_id =1;

select * from applications_history WHERE job_id =1;

*/
```

-- 3.1.3.4 -----

-- 3.1.3.4a-----

```
DELIMITER $

CREATE PROCEDURE search_a(
IN grade_1 INT, IN grade_2 INT
)
BEGIN
DECLARE employee_username varchar(30);
DECLARE job int(11);

SELECT employee, job_id
FROM applications_history
WHERE grade > grade_1 AND grade < grade_2;

END$

DELIMITER ;

# CALL search_a(6,13);
```

-- 3.1.3.4b -----

```
DELIMITER $

CREATE PROCEDURE search_b(
  IN evaluator varchar(30)
)
BEGIN
  DECLARE employee_username varchar(30);
  DECLARE job int(11);
  SELECT employee, job_id
  FROM applications_history
  WHERE evaluator_1 = evaluator OR evaluator_2 = evaluator;
END$

DELIMITER ;
```

Κεφάλαιο 3:

Περιγραφή:

3.1.4.1

Η πραγμάτωση των ζητούμενων αυτού του ερωτήματος χρειαζόταν 6 triggers. Για κάθε έναν από τους πίνακες `job`, `user`, `degree` έφτιαξα από 3 triggers για κάθε ενέργεια εισαγωγής, ενημέρωσης ή διαγραφής. Ακολουθεί περιγραφή για ένα μόνο trigger από τα 6, και η περιγραφή για τα υπόλοιπα είναι αντίστοιχη. Αρχικά, ένας από τους dba χρήστες πρέπει να συνδεθεί στην βάση μας. Η εντολή `USER()` μας επιστρέφει το όνομα του χρήστη που είναι συνδεδεμένος, ακολουθούμενο από `@` και κάποια άλλα πράγματα, τα οποία δεν χρειαζόμαστε. Για να πάρουμε μόνο το όνομα του χρήστη εκτελούμε την εντολή `SUBSTRING_INDEX(USER(),'@',1)`. Ύστερα ελέγχουμε αν ο χρήστης που είναι συνδεδεμένος δεν είναι ο `root`, και στην συνέχεια εισάγουμε τα στοιχεία του στον πίνακα `log`.

3.1.4.2

Η υλοποίηση αυτού του trigger ήταν πολύ απλή, καθώς απλά μετράει τον αριθμό των ενεργών αιτήσεων του συγκεκριμένου εργαζομένου με μια `select count(*)` και αποθηκεύει την ημερομηνία έναρξης της θέσης σε μια μεταβλητή (για ευκολία ανάγνωσης). Μετά υπάρχει ένα `if` που ελέγχει εάν πληρούνται οι προϋποθέσεις, και εάν όχι, ακυρώνει το `insert` (γι' αυτό και `before insert`) και τυπώνει το απαραίτητο μήνυμα.

Ο έλεγχος για τις ημερομηνίες γίνεται με χρήση της συνάρτησης `DATEDIFF`.

3.1.4.3

Αυτό το trigger ακολουθεί την ίδια λογική με το προηγούμενο. Καλείται `PPIN` την ενημέρωση στον πίνακα `applies`, και αποθηκεύει τον αριθμό των ενεργών αιτήσεων και την ημερομηνία έναρξης της θέσης.

Μετά εάν ο χρήστης προσπαθεί να ενεργοποιήσει κάποια ακυρωμένη εγγραφή και έχει 3 ήδη ενεργές, ακυρώνει την ενημέρωση και εκτυπώνει αντίστοιχο μήνυμα.

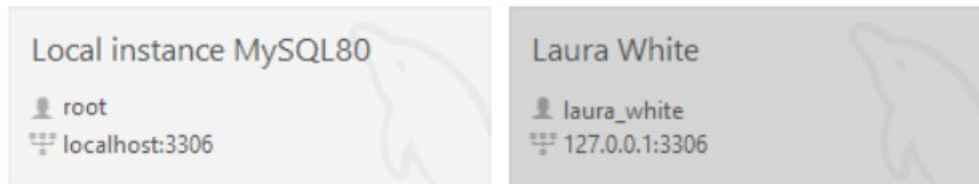
Εάν πάλι, προσπαθεί να ακυρώσει μια μη ακυρωμένη εγγραφή και η ημερομηνία έναρξης είναι σε λιγότερο από 10 μέρες, δεν το επιτρέπει, και τυπώνει αντίστοιχο μήνυμα.

Ο έλεγχος των ημερομηνιών γίνεται με χρήση της συνάρτησης `CURDATE()`.

Παραδείγματα από την εκτέλεση των triggers:

3.1.4.1

MySQL Connections



```
DELETE FROM job
WHERE id= 9;
```

```
INSERT INTO job (id, start_date, salary, position, edra, announce_date,
submission_date)
```

```
VALUES (NULL, '2023-01-15', 50000, 'Software Engineer', 'Athens', '2022-12-01',
'2022-12-15');
```

```
UPDATE job
```

```
SET salary = 69
```

```
WHERE id = 10;
```

```
select * from log;
```

	change_id	changes	changed_tables	change_time	username
▶	1	DELETE	job	2024-01-21 17:32:50	laura_white
	2	INSERT	job	2024-01-21 17:34:38	laura_white
	3	UPDATE	job	2024-01-21 17:34:55	laura_white
*	NULL	NULL	NULL	NULL	NULL

3.1.4.2

-- make a job less than 15 days away

```
INSERT INTO job VALUES
(NULL, DATE_ADD(DATE(NOW()), INTERVAL 14 DAY), 50000.00, 'TEST', 'TEST',
'john_doe', 'alice_smith', '2025-01-05 08:00:00', '2025-02-01');
```

-- see the job id

```
select id from job order by id DESC limit 1;
```

Result Grid	
	id
▶	17
*	NULL

-- try to insert an application for this job

```
INSERT INTO applies VALUES
('mark_smith', 17, DEFAULT, NOW(), 15, 18, DEFAULT);
```

✓	588	18:49:39	call application_handler('Hunitesig', 4,'a')	1 row(s) returned
✓	589	18:49:41	select employee, job_id, application_status from applies where employee = "Hunitesig" LIMIT 0, 1000	3 row(s) returned
✓	590	18:53:29	INSERT INTO job VALUES (NULL, DATE_ADD(DATE(NOW()), INTERVAL 14 DAY), 50000.00, 'TEST'...	1 row(s) affected
✓	591	18:53:32	select id from job order by id DESC limit 1	1 row(s) returned
✗	592	18:54:45	INSERT INTO applies VALUES ('mark_smith', 17, DEFAULT, NOW(), 15, 18, DEFAULT)	Error Code: 1644, ERROR: Employee cannot apply for this position

-- see how many active applicaitons hunitesig has

```
select employee, job_id, application_status from applies where employee =
"Hunitesig";
```

	employee	job_id	application_status
▶	Hunitesig	4	active
	Hunitesig	5	active
	Hunitesig	6	active
*	NULL	NULL	NULL

-- try to insert another

INSERT INTO applies VALUES

('Hunitesig', 8, DEFAULT, NOW(), 15, 18, DEFAULT);

✓	589	18:49:41	select employee, job_id, application_status from applies where employee = "Hunitesig" LIMIT 0, 1000	3 row(s) returned
✓	590	18:53:29	INSERT INTO job VALUES (NULL, DATE_ADD(DATE(NOW()), INTERVAL 14 DAY), 50000.00, 'TEST'...	1 row(s) affected
✓	591	18:53:32	select id from job order by id DESC limit 1	1 row(s) returned
✗	592	18:54:45	INSERT INTO applies VALUES ('mark_smith', 17, DEFAULT, NOW(), 15, 18, DEFAULT)	Error Code: 1644. ERROR: Employee cannot apply for this position
✓	593	18:57:03	select employee, job_id, application_status from applies where employee = "Hunitesig" LIMIT 0, 1000	3 row(s) returned
✗	594	18:57:32	INSERT INTO applies VALUES ('Hunitesig', 8, DEFAULT, NOW(), 15, 18, DEFAULT)	Error Code: 1644. ERROR: Employee cannot apply for this position

3.1.4.3

-- create a job that starts in more than 9 days

INSERT INTO job VALUES

(NULL, '2024-03-11', 50000.00, 'IT Support Specialist', 'Athens', 'john_doe',
'alice_smith', '2023-01-05 08:00:00', '2024-01-15');

-- see the job_id

select id from job order by id DESC limit 1;

	id
▶	18
*	NULL

-- Make an application for this job

call application_handler('Alte1970', 18, 'i');

-- make it so the job starts in less than 10 days

update job

SET start_date = DATE_ADD(DATE(NOW()), INTERVAL 9 DAY)

WHERE id =18;

-- try to cancel his application

call application_handler('Alte1970', 18, 'c');

✓	706	19:04:19	INSERT INTO job VALUES (NULL, DATE_ADD(DATE(NOW()), INTERVAL 14 DAY), 50000.00, 'TEST'...	1 row(s) affected
✓	707	19:04:29	INSERT INTO job VALUES (NULL, '2024-03-11', 50000.00, 'IT Support Specialist', 'Athens', 'john_doe', '...	1 row(s) affected
✓	708	19:04:31	select id from job order by id DESC limit 1	1 row(s) returned
✓	709	19:04:33	call application_handler('Alte1970', 18, 'i')	1 row(s) returned
✓	710	19:04:36	update job SET start_date = DATE_ADD(DATE(NOW()), INTERVAL 9 DAY) WHERE id =18	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✗	711	19:04:41	call application_handler('Alte1970', 18, 'c')	Error Code: 1644. ERROR: Starting date is less than 10 days away

-- see active applications for x employee

select employee, job_id, application_status from applies where employee = "Hunitesige";

	employee	job_id	application_status
▶	Hunitesige	4	active
	Hunitesige	5	active
	Hunitesige	6	active
*	NULL	NULL	NULL

-- cancel his third application

call application_handler('Hunitesige', 4,'c');

	employee	job_id	application_status
▶	Hunitesige	4	canceled
	Hunitesige	5	active
	Hunitesige	6	active
*	NULL	NULL	NULL

-- make a new one

call application_handler('Hunitesige', 7,'i');

	employee	job_id	application_status
▶	Hunitesige	4	canceled
	Hunitesige	5	active
	Hunitesige	6	active
	Hunitesige	7	active
*	NULL	NULL	NULL

-- try to activate his canceled application

call application_handler('Hunitesige', 4,'a');

✓	712	19:05:16	select employee, job_id, application_status from applies where employee = "Hunitesige" LIMIT 0, 1000	3 row(s) returned
✓	713	19:05:43	call application_handler('Hunitesige', 4,'c')	1 row(s) returned
✓	714	19:05:48	select employee, job_id, application_status from applies where employee = "Hunitesige" LIMIT 0, 1000	3 row(s) returned
✓	715	19:06:11	call application_handler('Hunitesige', 7,'i')	1 row(s) returned
✓	716	19:06:13	select employee, job_id, application_status from applies where employee = "Hunitesige" LIMIT 0, 1000	4 row(s) returned
✗	717	19:06:42	call application_handler('Hunitesige', 4,'a')	Error Code: 1644. ERROR: Employee already has 3 active applications

Κώδικας triggers:

```
DELIMITER $
CREATE TRIGGER project_trigger
BEFORE INSERT ON project
FOR EACH ROW
BEGIN
DECLARE max_num INT;

# Get the maximum num for the current employee
SELECT MAX(num) INTO max_num
FROM project
WHERE candid= NEW.candid;

# Set the new num value
SET NEW.num = COALESCE(max_num, 0) + 1;

END $
DELIMITER ;
```

/* ta triggers gia to erotima 3.1.4.1.*/

```
DELIMITER $
CREATE TRIGGER job_trigger_1
AFTER INSERT ON job
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);
SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('INSERT', 'job', now(), t_username);
END IF;

END $
DELIMITER ;

-----

DELIMITER $
CREATE TRIGGER job_trigger_2
AFTER DELETE ON job
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);
```

```

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('DELETE', 'job', now(), t_username);
END IF;

END $
DELIMITER ;
-----

DELIMITER $
CREATE TRIGGER job_trigger_3
AFTER UPDATE ON job
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('UPDATE', 'job', now(), t_username);
END IF;

END $
DELIMITER ;
-----

DELIMITER $
CREATE TRIGGER user_trigger_1
AFTER INSERT ON user
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('INSERT', 'user', now(), t_username);
END IF;

END $
DELIMITER ;
-----

DELIMITER $
CREATE TRIGGER user_trigger_2
AFTER DELETE ON user
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('DELETE', 'user', now(), t_username);

```

```

END IF;

END $
DELIMITER ;

-----

DELIMITER $
CREATE TRIGGER user_trigger_3
AFTER UPDATE ON user
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('UPDATE', 'user', now(), t_username);
END IF;

END $
DELIMITER ;

-----

DELIMITER $
CREATE TRIGGER degree_trigger_1
AFTER INSERT ON degree
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('INSERT', 'degree', now(), t_username);
END IF;

END $
DELIMITER ;

-----

DELIMITER $
CREATE TRIGGER degree_trigger_2
AFTER DELETE ON degree
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('DELETE', 'degree', now(), t_username);
END IF;

END $
DELIMITER ;

-----

```

```

DELIMITER $
CREATE TRIGGER degree_trigger_3
AFTER UPDATE ON degree
FOR EACH ROW
BEGIN
DECLARE t_username VARCHAR(30);

SET t_username = SUBSTRING_INDEX(USER(), '@', 1);

-- Now t_username contains the extracted username

IF t_username <> 'root' THEN
INSERT INTO log (changes, changed_tables, change_time, username)
VALUES('UPDATE', 'degree', now(), t_username);
END IF;

END $
DELIMITER ;

```

/* ta triggers gia to erotima 3.1.4.2*/

```

DELIMITER $
CREATE TRIGGER Application_Status
BEFORE INSERT ON applies
FOR EACH ROW
BEGIN
                                DECLARE applications int;
    DECLARE job_start date;

    select count(*) into applications
    from applies
    where employee = new.employee AND application_status = 'active';

    select job.start_date into job_start
    from job
    where job.id = new.job_id;

                                IF(applications >= 3 OR DATEDIFF(job_start, DATE( NOW() ) ) < 15)
    THEN
                                SIGNAL SQLSTATE '45000'
                                SET MESSAGE_TEXT = 'ERROR: Employee cannot apply for this position ';
                                END IF;

END $
DELIMITER ;

/*
-- make a job less than 15 days away
INSERT INTO job VALUES
(NULL, DATE_ADD(DATE(NOW()), INTERVAL 14 DAY), 50000.00, 'TEST', 'TEST', 'john_doe', 'alice_smith', '2025-01-05 08:00:00',
'2025-02-01');
-- see the job id
select id from job order by id DESC limit 1;

-- try to insert an application for this job
INSERT INTO applies VALUES
('mark_smith', 17, DEFAULT, NOW(), 15, 18, DEFAULT);

-- see how many active applicaitons hunitesige has
select employee, job_id, application_status from applies where employee = "Hunitesige";
-- try to insert another
INSERT INTO applies VALUES
('Hunitesige', 8, DEFAULT, NOW(), 15, 18, DEFAULT);

```

*/

/* ta triggers gia to erotima 3.1.4.3*/

```
DELIMITER $
CREATE TRIGGER cancel_enable_prevention
BEFORE UPDATE ON applies
FOR EACH ROW
BEGIN
    DECLARE starting_date DATE;
    DECLARE applications INT;

    SELECT start_date INTO starting_date
    FROM job
    WHERE job.id = old.job_id;

    select count(*) into applications
    from applies
    where employee = new.employee AND application_status = 'active';

    IF(applications >= 3 AND new.application_status = 'active')
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ERROR: Employee already has 3 active applications ';
    END IF;

    IF new.application_status = 'canceled' and old.application_status != 'canceled' THEN
        IF starting_date - CURDATE() < 10 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'ERROR: Starting date is less than 10 days away ';
        END IF;
    END IF;
END$
DELIMITER ;

/* -- TEST FOR 3.1.4.3
-- create a job
INSERT INTO job VALUES
(NULL, '2024-03-11', 50000.00, 'IT Support Specialist', 'Athens', 'john_doe', 'alice_smith', '2023-01-05 08:00:00', '2024-01-15');
-- see the job_id
select id from job order by id DESC limit 1;
-- Make an application for this job
call application_handler('Alte1970', 18, 'i');
-- make it so the job starts in less than 10 days
update job
SET start_date = DATE_ADD(DATE(NOW()), INTERVAL 9 DAY)
WHERE id = 18;
-- try to cancel his application
call application_handler('Alte1970', 18, 'c');
-- see active applications for x employee
select employee, job_id, application_status from applies where employee = "Hunitesige";
-- cancel his third application
call application_handler('Hunitesige', 4, 'c');
-- make a new one
call application_handler('Hunitesige', 7, 'i');
-- try to activate his canceled application
call application_handler('Hunitesige', 4, 'a'); */
```

Κεφάλαιο 4

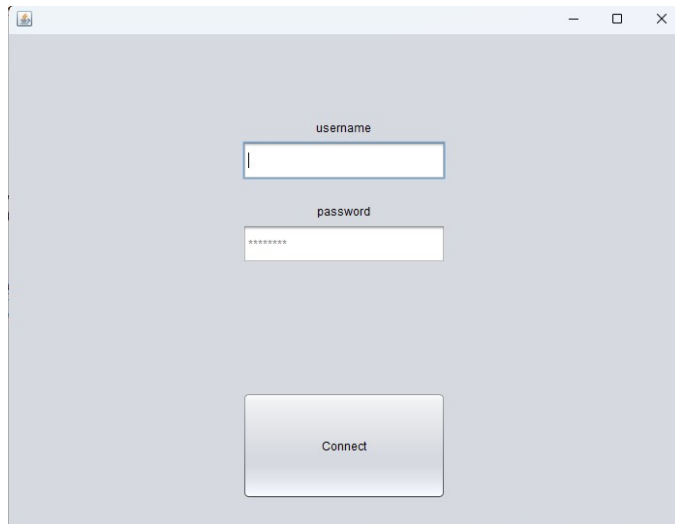
Περιγραφή

Απο το κεφαλαιο 4 υλοποιησαμε μονο το 3.2.1

Χρησιμοποιήσαμε 2 κλασεις, την `main – ProjectGUI` και την `ConnectToDB`, που μονη της δουλεια ειναι οι μεθοδοι συνδεσης με την βαση.

Όταν αρχίζει το πρόγραμμα, δημιουργείται ένα στιγμιότυπο της μεθόδου ProjectGUI, και το κάνει ορατό.

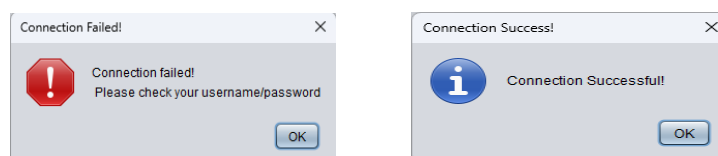
Όταν δημιουργείται το στιγμιότυπο, πρώτα γίνονται τα “μαγικά” του NetBeans για την αρχικοποίηση των απαραίτητων στο GUI που έχουμε δημιουργήσει γραφικά, και στον χρήστη εμφανίζεται το παρακάτω παραθυρό:



Ένα Frame που περιέχει δυο labels που δείχνουν στον χρήστη τι πρέπει να συμπληρώσει σε καθένα από τα δυο πλαίσια: το username και το password.

Παρατήρηση: Στα πλαίσια έχουμε επιλέξει να φαίνεται με γκρι χρώμα το όνομα του αντιστοιχού πλαισίου (username, password). Αυτό το κάναμε αρχικοποιώντας τα πλαίσια έτσι, και οπότε ο χρήστης κάνει focus στο πλαίσιο, εάν η τιμή είναι username/password, την σβήνει, επιτρέποντας στον χρήστη να πληκτρολογήσει τα στοιχεία του. Όταν χάνεται το focus από το πλαίσιο, ελέγχεται εάν η τιμή έχει αλλάξει και εάν όχι, επιστρέφει στις γκρι τιμές.

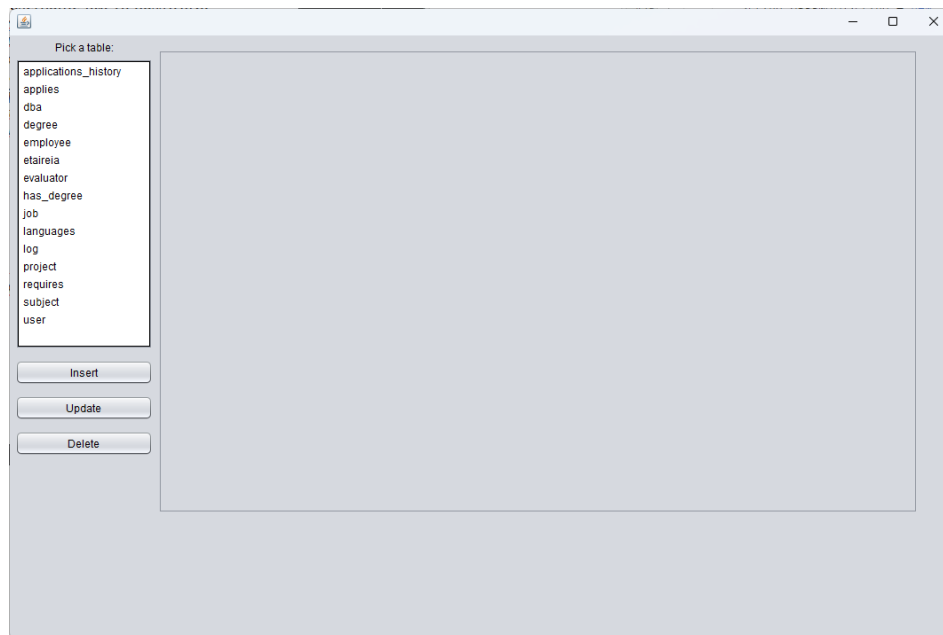
Τέλος, το κουμπί Connect, που όταν πατηθεί, επιχειρείται η σύνδεση με την βάση. Εάν αυτή αποτύχει, εκτυπώνεται αντιστοιχό μήνυμα στο cli και εμφανίζεται ένα ενημερωτικό παραθυρό στον χρήστη.



Εάν η σύνδεση είναι επιτυχής, εμφανίζεται ένα ενημερωτικό παραθυρό.

Τα παραθυρά των μηνυμάτων υλοποιήθηκαν με χρήση της μεθόδου showMessageDialog της JOptionPane.

Όταν ο χρήστης πατήσει OK (μετά από μια επιτυχή σύνδεση), διαγράφεται το προηγούμενο frame, και αντικαθίσταται από αυτό:



Στο παρασκήνιο γίνονται 3 πράγματα:

- Καλούμε την μεθοδο `ShowTablesQuery()` μεσα στην λιστα (που βρίσκεται πανω αριστερα στο frame).

Η μεθοδος αυτη εκτελει ενα `SHOW TABLES` στην βαση μας, αποθηκευει τα αποτελεσματα σε ενα `ArrayList` και επιστρεφει εναν πινακα απο `Strings` που περιεχει τους πινακες.

Ετσι, υπολογιζονται αυτοματα οι πινακες της βασης και εμφανιζονται στην λιστα.

- Κανουμε ορατό το καινουργιο Frame
- Διαγραφουμε το προηγουμενο Frame

Αυτο το Frame αποτελείται απο την λιστα, τρια κουμπια για καθεμια απο τις ενεργειες, και εναν αρχικα αδειο πινακα.

Οταν ο χρηστης επιλέξει ενα πινακα απο την λιστα, εκτελείται μια `SELECT` για ολα τα στοιχεια του πινακα, και αποθηκευουμε τα `metadata` του αποτελεσματος σε μια μεταβλητη (για τα ονοματα των στηλών).

Μηδενιζουμε τον πινακα για να φυγουν τυχον προηγουμενες πληροφοριες, δημιουργουμε εναν μονοδιαστατο πινακα `String` με τις τιμες των στηλών, και τις βαζουμε σαν τιτλους του πινακα (μεσω `setColumnIdentifiers`).

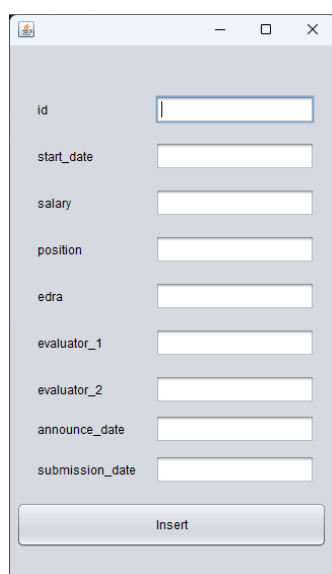
Μετα τοποθετουμε τις εγγραφες στον πινακα με παρομοιο τροπο (`addRow`).

Ετσι, όταν ο χρήστης επιλέγει έναν πίνακα από την λίστα, εμφανίζονται αυτοματα όλες οι εγγραφές και τα ονοματα αυτών στον πίνακα στην οθονη.

- **Κουμπι insert:**

Προκειμένου να λειτουργήσει, πρέπει να έχει επιλεγθεί ο πίνακας από την λίστα.

Όταν πατηθεί το κουμπι insert, εμφανίζεται ένα τέτοιο παραθυρο:



Στο παρασκήνιο καλείται μια SELECT για τον πίνακα που έχει επιλέξει ο χρήστης, τα ονοματα των labels γίνονται τα ονοματα των στηλών του πίνακα, και εμφανίζονται στο νέο frame μόνο όσα πλαίσια – labels υπάρχουν στον πίνακα.

Στην πραγματικότητα, αυτό το νέο frame περιέχει 10 διαφορετικά σέτ labels – textFields και δυο κουμπια (insert – update), και αναλογα τον πίνακα που έχει επιλέξει ο χρήστης γίνονται ορατά μόνο όσα σέτ απαιτούνται από τον συγκεκριμένο πίνακα.

Ο χρήστης βάζει τις τιμές που θέλει να κάνει insert στα αντιστοιχα πλαίσια και πατάει το κουμπι insert.

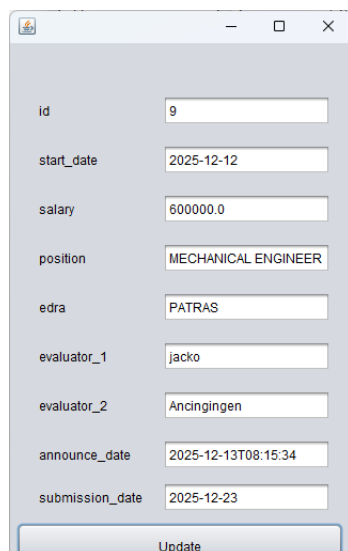
Το κουμπι insert συνθέτει την εντολή INSERT και την εκτελεί. Εάν πετύχει εμφανίζει καταλλήλο παραθυρο, όπως και εάν αποτύχει. Εάν πετύχει, το παραθυρο αυτό κλείνει.

ΣΗΜΕΙΩΣΗ: Για την συνδεση της insert, εάν ο χρήστης έχει αφήσει κενό σε κάποιο πεδίο η τιμή του πεδίου στην insert γίνεται NULL, και εάν είναι default, DEFAULT, null, NULL δεν τοποθετείται σε εισαγωγικά για την σωστή λειτουργία της.

- **Κουμπι update:**

Προκειμένου να λειτουργήσει, πρέπει να έχει επιλεγθεί ο πίνακας και μια εγγραφή από αυτόν.

Όταν πατηθεί το κουμπί update, εμφανίζεται ένα τέτοιο παράθυρο:



Στην πραγματικότητα τα δυο παράθυρα για την insert και για την update είναι το ίδιο παράθυρο, απλώς όταν καλείται από την insert, δεν εμφανίζεται το κουμπί update και το αντιστρόφω.

Με παρόμοιο τρόπο εμφανίζονται τα στοιχεία και εδώ, μόνο που αρχικοποιούνται και οι τιμές σε κάθε πλαίσιο σύμφωνα με την εγγραφή που έχει επιλέξει ο χρήστης.

Το κουμπί Update συνθέτει την εντολή UPDATE και την εκτελεί. Και πάλι εμφανίζει κατάλληλο παράθυρο

αναλογως εαν πετυχε / απετυχε η εντολη.

- **Κουμπί delete:**

Προκειμένου να λειτουργήσει, πρέπει να έχει επιλεγθεί ο πίνακας και μια εγγραφή από αυτόν.

Όταν πατηθεί το κουμπί delete, συντίθεται αυτοματα η εντολη και εκτελειται, εμφανιζοντας καταλληλο μηνυμα.

Και πάλι υπάρχει έλεγχος, ώστε εάν το συγκεκριμένο δεδομένο είναι ακέραιος να μην τοποθετούνται εισαγωγικά στην σύνταξη της εντολής.

Σενάριο Χρήσης

Εστω πως ο DBA της βάσης Mark Anderson θέλει να ενημερώσει μια θέση εργασίας που είχε λάθος ημερομηνία έναρξης. Τα βήματα που πρέπει να ακολουθήσει είναι τα εξής:

1. Συνδεση με τα στοιχεία του.

username

mark_anderson

password

Connection Success!

Connection Successful!

OK

Connect

2. Επιλογή του πίνακα job και της εγγραφής που θέλει να αλλάξει

Pick a table:

- applications_history
- applies
- dba
- degree
- employee
- etaireia
- evaluator
- has_degree
- job**
- languages
- log
- project
- requires
- subject
- user

Insert

Update

Delete

id	start_date	salary	position	edra	evaluator_1	evaluator_2	announce_da...	submission_...
1	2025-01-15	50000.0	IT Support Sp...	Athens	john_doe	alice_smith	2025-01-05T...	2025-02-01
2	2025-02-20	60000.0	Financial Ana...	Paris	alice_smith	bob_jones	2025-02-10T...	2025-03-01
3	2025-03-25	70000.0	Production C...	Berlin	bob_jones	emma_wilson	2025-03-15T...	2025-04-01
4	2025-04-10	55000.0	Registered N...	Madrid	emma_wilson	alex_miller	2025-04-01T...	2025-05-01
5	2025-05-15	65000.0	Marketing Sp...	London	alex_miller	sara_jackson	2025-05-05T...	2025-06-01
6	2025-06-20	75000.0	Teacher	Barcelona	sara_jackson	john_doe	2025-06-10T...	2025-07-01
7	2025-07-15	80000.0	Software Dev...	Munich	john_doe	emma_wilson	2025-07-05T...	2025-08-01
8	2025-08-20	90000.0	Data Scientist	Amsterdam	emma_wilson	alex_miller	2025-08-10T...	2025-09-01
9	2025-12-12	60000.0	MECHANICA...	PATRAS	jacko	Ancingen	2025-12-13T...	2025-12-23
10	2025-01-15	50000.0	Software Engi...	New York	Ancingen	johnnyboy	2025-01-10T...	2025-02-28
11	2025-02-20	60000.0	Data Analyst	California	johnnyboy	jacko	2025-02-15T...	2025-04-15
12	2025-03-10	70000.0	Marketing Ma...	Texas	jacko	Mucas1940	2025-03-05T...	2025-05-30
13	2025-04-05	55000.0	Financial Adv...	Florida	Mucas1940	Alke1996	2025-03-30T...	2025-06-10
14	2025-05-22	65000.0	HR Specialist	Washington	Alke1996	DreamyCoder	2025-05-15T...	2025-07-20
15	2025-06-30	75000.0	Project Mana...	Illinois	DreamyCoder	Ancingen	2025-06-25T...	2025-08-31
16	2025-07-12	58000.0	Sales Repres...	Arizona	Ancingen	johnnyboy	2025-07-05T...	2025-09-25
17	2025-01-15	50000.0	IT Support Sp...	Athens	john_doe	Ancingen	2025-01-05T...	2025-02-01

3. Πατημα του κουμπιού Update και επιθυμητή αλλαγή

id

9

start_date

2025-12-12

salary

60000.0

position

MECHANICAL ENGINEER

edra

PATRAS

evaluator_1

jacko

evaluator_2

Ancingen

announce_date

2025-12-13T08:15:34

submission_date

2025-12-23

Update

id

9

start_date

2025-12-12

salary

60000.0

position

MECHANICAL ENGINEER

edra

PATRAS

evaluator_1

jacko

evaluator_2

Ancingen

announce_date

2025-12-13T08:15:34

submission_date

2025-12-23

Update

4. Ολοκλήρωση της διαδικασίας με το πατημα του κουμπιού Update

Update Successfull

i

Update Success!

OK

Pick a table:

applications_history

applies

dba

degree

employee

etaireia

evaluator

has_degree

job

languages

log

project

requires

subject

user

Insert

Update

Delete

id	start_date	salary	position	edra	evaluator_1	evaluator_2	announce_da...	submission_...
1	2025-01-15	50000.0	IT Support Sp...	Athens	john_doe	alice_smith	2025-01-05T...	2025-02-01
2	2025-02-20	60000.0	Financial Ana...	Paris	alice_smith	bob_jones	2025-02-10T...	2025-03-01
3	2025-03-25	70000.0	Production C...	Berlin	bob_jones	emma_wilson	2025-03-15T...	2025-04-01
4	2025-04-10	55000.0	Registered N...	Madrid	emma_wilson	alex_miller	2025-04-01T...	2025-05-01
5	2025-05-15	65000.0	Marketing Sp...	London	alex_miller	sara_jackson	2025-05-05T...	2025-06-01
6	2025-06-20	75000.0	Teacher	Barcelona	sara_jackson	john_doe	2025-06-10T...	2025-07-01
7	2025-07-15	80000.0	Software Dev...	Munich	john_doe	emma_wilson	2025-07-05T...	2025-08-01
8	2025-08-20	90000.0	Data Scientist	Amsterdam	emma_wilson	alex_miller	2025-08-10T...	2025-09-01
9	2025-02-12	60000.0	MECHANICA...	PATRAS	jacko	Ancingen	2025-12-13T...	2025-12-23
10	2025-01-15	50000.0	Software Engi...	New York	Ancingen	johnnyboy	2025-01-10T...	2025-02-28
11	2025-02-20	60000.0	Data Analyst	California	johnnyboy	jacko	2025-02-15T...	2025-04-15
12	2025-03-10	70000.0	Marketing Ma...	Texas	jacko	Mucas1940	2025-03-05T...	2025-05-30
13	2025-04-05	55000.0	Financial Adv...	Florida	Mucas1940	Alke1996	2025-03-30T...	2025-06-10
14	2025-05-22	65000.0	HR Specialist	Washington	Alke1996	DreamyCoder	2025-05-15T...	2025-07-20
15	2025-06-30	75000.0	Project Mana...	Illinois	DreamyCoder	Ancingen	2025-06-25T...	2025-08-31
16	2025-07-12	58000.0	Sales Repres...	Arizona	Ancingen	johnnyboy	2025-07-05T...	2025-09-25
17	2025-01-15	50000.0	IT Support Sp...	Athens	john_doe	Ancingen	2025-01-05T...	2025-02-01