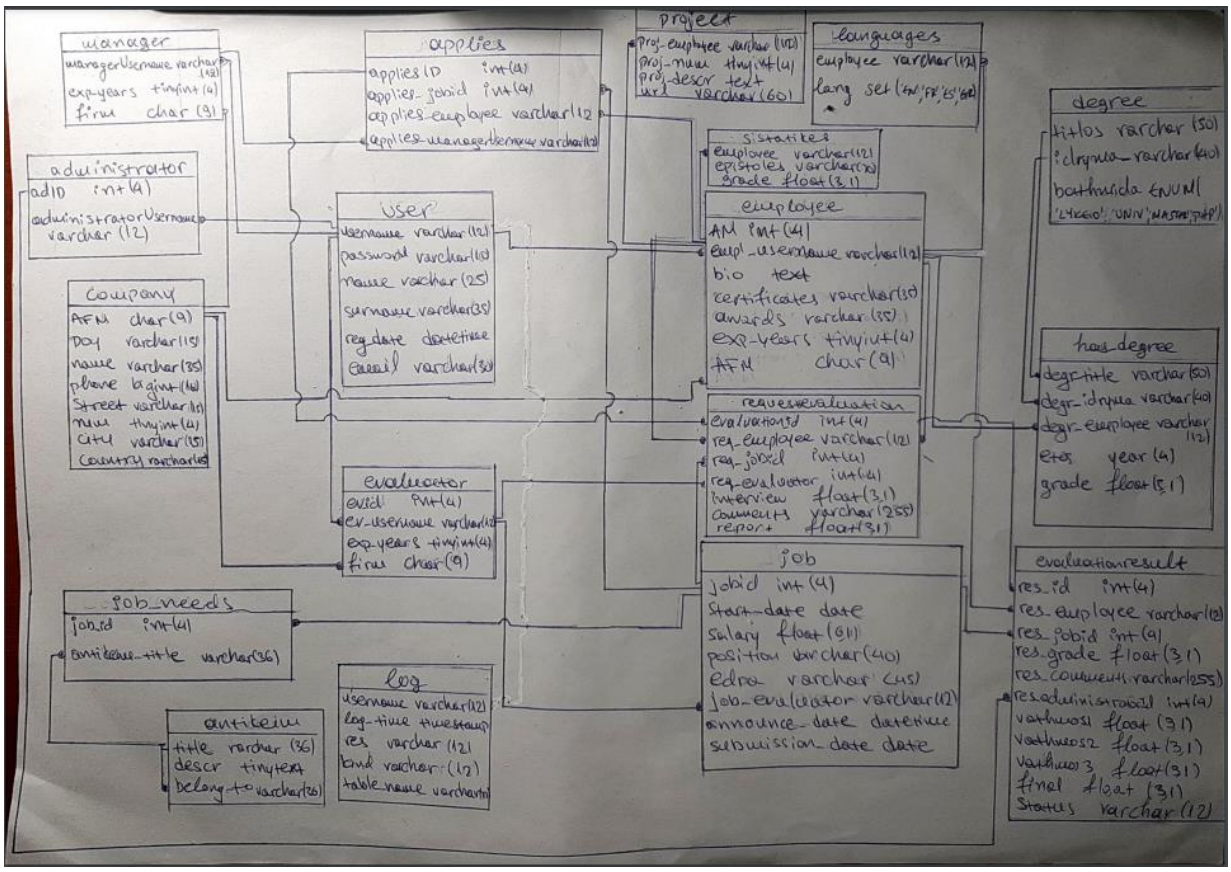
Εργασία Βάσεων Δεδομένων- Ειδική Ρύθμιση Νοέμβριος 2021

Χρυσανθοπούλου Αριστέα: 1067483

Κούρου Αγγελική:1067499

Ερώτημα 1: Παρακάτω παρουσιάζεται το τελικό σχεσιακό διάγραμμα της βάσης που κατασκευάσαμε.



Στην συνέχεια θα παρουσιαστούν οι κώδικες κατασκευής της βάσης, των procedures και των απαιτούμενων triggers.

Παραδοχές:

1. Υπάρχει μόνο ένας συμβατός καταγεγραμμένος administrator στην βάση για κάθε εταιρεία.
2. Ο αξιολογητής μόνο αξιολογεί. Ο manager είναι αυτός που ανακοινώνει τις θέσεις προς αίτηση, για ευκολότερη πρόσβαση στην δομή της βάσης.
3. Η υποβολή αίτησης γίνεται μέσω του πίνακα applies για λόγους λειτουργικότητας. Στην συνέχεια οι αιτήσεις μεταφέρονται στον πίνακα requestevaluation όπου γίνεται η διαδικασία αξιολόγησης και τα αποτελέσματα αποθηκεύονται στον evaluationresult, από τον οποίο και τα λαμβάνει ο χρήστης.
4. Στον πίνακα requestevaluation έχουμε προσθέσει τις μεταβλητές interview και report, που αναφέρονται στις αντίστοιχες βαθμολογίες της αξιολόγησης, καθώς και τα αντίστοιχα comments. Επίσης έχουμε συνδέσει κάθε request με έναν υπεύθυνο αξιολογητή και το αντίστοιχο applies_id.

5. Στον πίνακα evaluationresult έχουμε προσθέσει 3 μεταβλητές όπου αποθηκεύεται η ακολουθία κάθε φάσης της αξιολόγησης, τα comments και η έγκριση ή όχι της αίτησης.
6. Για τις συστατικές επιστολές δημιουργήσαμε έναν πίνακα όπου υπάρχουν οι επιστολές κάθε εργαζόμενου βαθμολογημένες. Ο πίνακας sistatikes συνδέεται άμεσα με τον πίνακα employee.
7. Τα αποτελέσματα τα ανακοινώνει ο διαχειριστής του συστήματος αξιολόγησης της εταιρείας.
8. Δημιουργήσαμε ένα procedure anakoinwsh το οποίο για κάθε νέα προστιθέμενη ανοιχτή θέση εργασίας ανακοινώνει με της είσοδο του jobid ότι έχει ανοίξει η υποβολή των αιτήσεων.

Στις επόμενες σελίδες θα παρουσιαστούν κατά σειρά τα create, insert, stored procedures και triggers που γράψαμε για να κατασκευάσουμε την βάση.

CREATE

```
DROP DATABASE IF EXISTS StaffEvaluation;

CREATE DATABASE StaffEvaluation;

USE StaffEvaluation;


CREATE TABLE IF NOT EXISTS user(
        username VARCHAR(12) NOT NULL,
        password VARCHAR(10) NOT NULL,
        name VARCHAR(25) NOT NULL,
        surname VARCHAR(35) NOT NULL,
        reg_date DATETIME,
        email VARCHAR(30),
        PRIMARY KEY(username)
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS company(
        AFM CHAR(9) NOT NULL,
        DOY VARCHAR(15) NOT NULL,
        name VARCHAR(35) NOT NULL,
        phone BIGINT(16) NOT NULL,
```

```
        street VARCHAR(15) NOT NULL,
        num TINYINT(4) NOT NULL,
    city VARCHAR(15) NOT NULL,
    country VARCHAR(15),
    PRIMARY KEY(AFM)
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS manager(
        managerUsername VARCHAR(12) NOT NULL,
    exp_years TINYINT(4),
    firm CHAR(9),
    PRIMARY KEY(managerUsername),
    FOREIGN KEY (managerUsername) REFERENCES user(username)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(firm) REFERENCES company(AFM)
    ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS evaluator(
        ev_id INT(4) NOT NULL,
        ev_username VARCHAR(12),
    exp_years TINYINT(4),
    firm CHAR(9),
    PRIMARY KEY(ev_id),
    FOREIGN KEY (ev_username) REFERENCES user(username)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(firm) REFERENCES company(AFM)
    ON DELETE CASCADE ON UPDATE CASCADE
```

```
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS employee(
        AM INT(4) NOT NULL,
        empl_username VARCHAR(12) NOT NULL,
        bio TEXT,
        certificates VARCHAR(35),
        awards VARCHAR(35),
        exp_years TINYINT(4),
        AFM CHAR(9) NOT NULL,
        PRIMARY KEY(AM,empl_username),
        FOREIGN KEY (AFM) REFERENCES company(AFM)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (empl_username) REFERENCES user(username)
   ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS languages(
        employee VARCHAR(12) NOT NULL,
        lang SET('EN','FR','ES','GR'),
   PRIMARY KEY(employee),
        FOREIGN KEY (employee) REFERENCES employee(empl_username)
   ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS job(
        jobid INT(4) NOT NULL AUTO_INCREMENT,
        start_date DATE NOT NULL,
        salary FLOAT(6,1) NOT NULL,
```

```
        position VARCHAR(40) NOT NULL,
        edra VARCHAR(45) NOT NULL,
    job_evaluator VARCHAR(12) NOT NULL,
    announce_date DATETIME NOT NULL,
    submission_date DATE NOT NULL,
    PRIMARY KEY(jobid),
    FOREIGN KEY (job_evaluator) REFERENCES evaluator(ev_username)
    ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;


CREATE TABLE IF NOT EXISTS antikeim(
    title VARCHAR(36) NOT NULL,
    descr TINYTEXT,
    belongs_to VARCHAR(36),
    PRIMARY KEY(title),
    FOREIGN KEY (belongs_to) REFERENCES antikeim(title)
    ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS applies(
        appliesID INT(4) NOT NULL,
        applies_jobid INT(4) NOT NULL,
        applies_employee VARCHAR(12) NOT NULL,
        applies_managerUsername VARCHAR(12) NOT NULL,
        PRIMARY KEY (appliesID),
        FOREIGN KEY (applies_jobid) REFERENCES job(jobid)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (applies_employee) REFERENCES employee(empl_username)
```

```
   ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (applies_managerUsername) REFERENCES manager(managerUsername)
   ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS job_needs(
        job_id INT(4) NOT NULL,
        antikeim_title VARCHAR(36) NOT NULL,
        PRIMARY KEY(job_id,antikeim_title),
        FOREIGN KEY (job_id) REFERENCES job(jobid)
   ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (antikeim_title) REFERENCES antikeim(title)
   ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS degree(
        titlos VARCHAR(50),
        idryma VARCHAR(40),
        bathmida ENUM('LYKEIO','UNIV','MASTER','PHD'),
        PRIMARY KEY(titlos,idryma)
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS has_degree(
  degr_title VARCHAR(50) ,
  degr_idryma VARCHAR(40) ,
  degr_employee VARCHAR(12) NOT NULL,
  etos YEAR(4) NOT NULL,
  grade FLOAT(3,1) NOT NULL,
  PRIMARY KEY(degr_title,degr_idryma,degr_employee),
  FOREIGN KEY (degr_employee) REFERENCES employee(empl_username)
```

```
  ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (degr_title, degr_idryma) REFERENCES degree(titlos,idryma)
  ON DELETE CASCADE ON UPDATE CASCADE

 )ENGINE=InnoDB;


CREATE TABLE IF NOT EXISTS project(
        proj_employee VARCHAR(12) NOT NULL,
        proj_num TINYINT(4) NOT NULL AUTO_INCREMENT,
        proj_descr TEXT,
        url VARCHAR(60),
        PRIMARY KEY(proj_num),
        FOREIGN KEY (proj_employee) REFERENCES employee(empl_username)
   ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;


CREATE TABLE IF NOT EXISTS administrator(
        adID INT(4) NOT NULL,
        administratorUsername VARCHAR(12) NOT NULL,
        PRIMARY KEY(adID),
   FOREIGN KEY (administratorUsername) REFERENCES user(username)
   ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS requestsevaluation(
        evaluationId INT(4) NOT NULL ,
        req_employee VARCHAR(12) NOT NULL,
        req_jobid INT(4) NOT NULL,
        req_evaluator VARCHAR(12) NOT NULL,
```

```
        interview FLOAT(3,1),
        comments VARCHAR(255) NOT NULL,
        report FLOAT(3,1),
        status VARCHAR(12),
        PRIMARY KEY(evaluationId),
        FOREIGN KEY (req_employee) REFERENCES employee(empl_username)
  ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (req_jobid) REFERENCES job(jobid)
  ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (req_evaluator) REFERENCES evaluator(ev_username)
  ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (evaluationId) REFERENCES applies(appliesID)
  ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;


CREATE TABLE IF NOT EXISTS evaluationresult(
        res_id INT(4) NOT NULL,
        res_employee VARCHAR(12) NOT NULL,
        res_jobid INT(4) NOT NULL,
        res_grade FLOAT(3,1) NOT NULL,
        res_comments VARCHAR(255) NOT NULL,
        res_administratorId INT(4) NOT NULL,
        vathmos1 FLOAT(3,1) NOT NULL,
        vathmos2 FLOAT(3,1) NOT NULL,
        vathmos3 FLOAT(3,1) NOT NULL,
        final FLOAT(3,1),
        status VARCHAR(12),
        PRIMARY KEY(res_id),
        FOREIGN KEY (res_employee) REFERENCES employee(empl_username)
  ON DELETE CASCADE ON UPDATE CASCADE,
```

```
   FOREIGN KEY (res_jobid) REFERENCES job(jobid)
   ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (res_administratorId) REFERENCES administrator(adID)
   ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (res_id) REFERENCES requestsevaluation(evaluationId)
   ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;



CREATE TABLE IF NOT EXISTS sistatikes(
      employee VARCHAR(12) NOT NULL,
      epistoles VARCHAR(30),
      grade FLOAT(3,1),
           FOREIGN KEY (employee) REFERENCES employee(empl_username)
    ON DELETE CASCADE ON UPDATE CASCADE

)ENGINE=InnoDB;

CREATE TABLE log
(
 username VARCHAR(12) NOT NULL,
 log_time TIMESTAMP NOT NULL,
 res VARCHAR(12) NOT NULL,
 kind VARCHAR(12) NOT NULL,
 table_name VARCHAR(12) NOT NULL,
 PRIMARY KEY (log_time)
)ENGINE=InnoDB;
```

INSERT

```
INSERT INTO user VALUES
```

('erachrr','12345','Aristea','Chrysanthopoulou','2010-10-1','crysanth@gmail.com'), #MANAGER

('manwlhs','67890','Emmanouhl','Skoteinos','2010-10-20','manosf@gmail.com'), # EVALUATOR 1

('fayboo','54321','Fwteinh','Aggelaki','2010-10-21','kassandra@gmail.com'), #EVALUATOR 2

('makhst','09876','Mixahl','Theodorou','2010-10-22','mixalhs@gmail.com'), #EVALUATOR 3

('vlachospy','abcde','Spyros','Vlachos','2010-10-30','spyvlach@gmail.com'), #EMPLOYEE 1

('edwardst','fghij','Edward','Stergiou','2010-11-1','edward@gmail.com'), #EMPLOYEE 2

('tsaggelikh','klam4','Angeliki','Kourou','2010-11-1','teapot@gmail.com'), #EMPLOYEE 3

('mavrapol','kl1no','Mavra','Polydorou','2010-11-2','mavreta@gmail.com'); #ADMIN

INSERT INTO company VALUES

('1256','A Patras','Ubisoft','2610430890','Agiou Andreou',2,'Patra','Greece'),

('1285','B Patras','Intracom','2610639026','Kolokotrwni',4,'Patra','Greece'),

('1345','A Athinwn','Cosmote','2108353469','Agioy Iwannou',3,'Athina','Greece'),

('9367','A Patras','Vodasoft','2610567890','Amalias Av.',58,'Patra','Greece');

INSERT INTO manager VALUES

('erachrr',10,'1256');

INSERT INTO evaluator VALUES(7540,'manwlhs',2,'1256');

INSERT INTO evaluator VALUES(7541,'fayboo',1,'1256');

INSERT INTO evaluator VALUES(7542,'makhst',4,'1256');

INSERT INTO employee VALUES

(7543,'vlachospy','cv1.txt','cert1.txt','awards1.txt',7,1256),

(7544,'edwardst','cv2.txt','cert2.txt','awards2.txt',8,1256),

(7545,'tsaggelikh','cv3.txt','cert3.txt','awards3.txt',4,1256);

INSERT INTO languages VALUES

('vlachospy','ES'),

('tsaggelikh', 'ES'),

('edwardst', 'FR');

INSERT INTO job VALUES

(3301,'2017-01-26',10000,'Human Resources Manager','Patra','manwlhs','2021-02-03 15:30:00','2021-07-30 15:30:00'),

(3302,'2021-03-06',20000,'Spaceship Engineer','Patra','fayboo','2021-04-14 15:30:00','2021-08-18 15:30:00'),

(3303,'2017-04-06',5000,'Junior Programming Assistant','Patra','makhst','2021-03-03 14:28:00','2021-10-05 15:30:00');

INSERT INTO antikeim VALUES

('Manager','Interfering with the employee',NULL),

('Engineer','Build, fix, develop schemas for a new ship',NULL);

INSERT INTO applies (appliesID ,applies_jobid,applies_employee , applies_managerUsername) VALUES

(4555,3301,'edwardst','erachrr'),

(4556,3302,'edwardst','erachrr');

INSERT INTO applies(appliesID ,applies_jobid,applies_employee , applies_managerUsername) VALUES

(4557,3302,'edwardst','erachrr');

INSERT INTO job_needs VALUES

(3301,'Manager'),

(3302,'Engineer');

INSERT INTO degree VALUES

('Computer Engineering','CEID','MASTER'), #Employee 1

('Economics','Technical University of Larissa','UNIV'), #Employee1

('Management & Economics','University of Pireaus','PHD'), #Employee2

('Computer Science','EKPA','PHD');

INSERT INTO has_degree VALUES

('Computer Engineering','CEID','vlachospy','2008',9.0), #Employee 1

('Economics','Technical University of Larissa','vlachospy','2007',7.0), #Employee1

('Management & Economics','University of Pireaus','edwardst','2006',8.0), #Employee2

('Computer Science','EKPA','edwardst','2004',9.0);

INSERT INTO project VALUES

('vlachospy',1,'my cloud.com','Multidimensional databases and the problems that occur'),

('edwardst',2,'my cloud.com','Economics and their effect on a budget');

INSERT INTO administrator VALUES(1097,'mavrapol');

INSERT INTO
requestsevaluation(evaluationId,req_employee,req_jobid,req_evaluator,interview
,comments,report) VALUES

(4555,'edwardst',3301,'manwlhs',9.0,'Candidate did an intermediate job',5.0),

(4556,'edwardst',3301,'manwlhs',8.0,'Candidate was prepared for a different
interview!',4.0),

(4557,'edwardst',3302,'fayboo',0.0,'Candidate wasnt prepared well.',0.0);

INSERT INTO sistatikes VALUES

('edwardst','sis.txt','7.2'),

('edwardst','sis.txt','3.5');

STORED PROCEDURES

mysql.exe -u root

```
DELIMITER $

DROP PROCEDURE IF EXISTS Employee$

CREATE PROCEDURE Employee(IN usernameOfEmployee VARCHAR(12))
BEGIN
DECLARE numOfapplies INT(4);
DECLARE numOfevaluations INT(4);
DECLARE finished_flag INT(4);
DECLARE evalusrname VARCHAR(12) ;
DECLARE nm VARCHAR(12);
DECLARE srnm VARCHAR(12);

DECLARE Evaluator CURSOR FOR
 SELECT req_evaluator
 FROM requestsevaluation
 LEFT JOIN evaluator ON evaluator.ev_id = requestsevaluation.req_evaluator
 LEFT JOIN employee ON employee.empl_username = requestsevaluation.req_employee
 WHERE employee.empl_username = usernameOfEmployee;


DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished_flag=1;

OPEN  Evaluator;
SET finished_flag = 0;

SELECT count(*) INTO numOfapplies
FROM applies
```

```
WHERE applies.applies_employee = usernameOfEmployee;

SELECT count(*) INTO numOfevaluations
FROM evaluationresult
WHERE evaluationresult.res_employee=usernameOfEmployee;

SELECT numOfapplies  AS 'Arithmos Aitisewn', numOfevaluations AS 'Arithmos Oloklиromenon Aitisewn';

WHILE(finished_flag=0) DO

FETCH Evaluator INTO evalusrname ;

   SELECT user.name AS 'ONOMA Aksiologiti' , user.surname AS 'EPWNYMO Aksiologitis'
   FROM user
   LEFT JOIN evaluator ON evaluator.ev_username = user.username
   WHERE  evaluator.ev_username =evalusrname ;

END WHILE;
CLOSE Evaluator;

IF(numOfapplies>numOfevaluations) THEN

SELECT 'Aksiologisi se ekseliksi';

SELECT *
FROM requestsevaluation
WHERE requestsevaluation.evaluationId  NOT IN (SELECT evaluationresult.res_id FROM evaluationresult);
END IF;
```

```
END$


DELIMITER ;
CALL Employee('edwardst');




DROP PROCEDURE IF EXISTS evaluation;
DELIMITER $
CREATE PROCEDURE evaluation(IN IdEvaluation INT(4),IN usernameofEmployee VARCHAR(12),IN IdJob INT(4),IN Evaluator VARCHAR(12))
BEGIN
DECLARE vathmos1 FLOAT(3,1);
DECLARE comm VARCHAR(255);
DECLARE vathmos2 FLOAT(3,1);
DECLARE gradeofdegree FLOAT(3,1);
DECLARE gradeofsistatikes FLOAT(3,1);
DECLARE numOfproject FLOAT(3,1);
DECLARE vathmos3 FLOAT(3,1);
DECLARE final FLOAT(3,1);



SELECT interview INTO vathmos1
FROM requestsevaluation
WHERE requestsevaluation.evaluationId=IdEvaluation;
SELECT vathmos1;

SELECT comments INTO comm
FROM requestsevaluation
WHERE requestsevaluation.evaluationId=IdEvaluation;
```

```
SELECT report INTO vathmos2
FROM requestsevaluation
WHERE requestsevaluation.evaluationId=IdEvaluation;


SELECT AVG(grade) INTO gradeofdegree
FROM has_degree
WHERE has_degree.degr_employee=usernameofEmployee;


SELECT AVG(grade) INTO gradeofsistatikes
FROM sistatikes
WHERE sistatikes.employee=usernameofEmployee;

SELECT count(*) INTO numOfproject
FROM project
WHERE project.proj_employee=usernameofEmployee;
SELECT numOfproject;

SET vathmos3=(numOfproject+gradeofdegree+gradeofsistatikes)/3;
SELECT vathmos3;
SET final= (vathmos1*0.4 + vathmos2*0.4+ vathmos3*0.2);
SELECT final;
IF(final>=4  ) THEN
INSERT INTO evaluationresult (res_id, res_employee,res_jobid, res_grade, res_comments,res_administratorId,vathmos1,vathmos2,vathmos3,status)
VALUES(IdEvaluation,usernameofEmployee,IdJob,final,comm,1097,vathmos1,vathmos2,vathmos3,'dekth');
ELSE IF (final < 4 ) THEN
INSERT INTO evaluationresult (res_id, res_employee,res_jobid, res_grade, res_comments,res_administratorId,vathmos1,vathmos2,vathmos3,status)
```

```
VALUES(IdEvaluation,usernameofEmployee,IdJob,final,comm,1097,vathmos1,vathmos2,vathmos3,'aporriptetai');


END IF;

END IF;

END$


DELIMITER ;

CALL evaluation(4555,'edwardst',3301,1097);

CALL evaluation(4556,'edwardst',3301,1097);

CALL evaluation(4557,'edwardst',3302,1097);




DROP PROCEDURE IF EXISTS finished_evals;

DELIMITER $

CREATE PROCEDURE finished_evals(IN jobcode INT(4))

BEGIN


  DECLARE count1 INT;

  DECLARE count2 INT;

  DECLARE result FLOAT(3,1);


  SELECT COUNT(*) INTO count1 FROM applies WHERE applies_jobid=jobcode ;

  SELECT COUNT(*) INTO count2 FROM evaluationresult WHERE res_jobid=jobcode;



  IF(count1 > count2) THEN

  SELECT count1 - count2 AS ' There are still requests pending evaluation: ';


  SELECT * FROM evaluationresult WHERE res_jobid=jobcode ORDER BY final DESC;
```

```
  ELSE
  SELECT * FROM evaluationresult WHERE res_jobid=jobcode ORDER BY final DESC;
  END IF;

END $
DELIMITER ;
CALL finished_evals(3301);



DROP PROCEDURE IF EXISTS anakoinwsh;
DELIMITER $
CREATE PROCEDURE anakoinwsh(IN IdJob INT(4))
BEGIN

SELECT position AS 'YPOVOLI AITHSEWN GIA THN THESI:'
FROM job
WHERE jobid=IdJob;

END$

DELIMITER ;

CALL anakoinwsh(3302);
```

TRIGGERS

```
DROP TRIGGER IF EXISTS insJob;
DELIMITER $
CREATE TRIGGER insJob
AFTER INSERT ON job
```

```
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (NEW.job_evaluator,current_timestamp,'insert','success','job');

END $
DELIMITER ;



DROP TRIGGER IF EXISTS updtJob;
DELIMITER $
CREATE TRIGGER updtJob
AFTER UPDATE ON job
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (NEW.job_evaluator,current_timestamp,'update','success','job');

END $
DELIMITER ;

DROP TRIGGER IF EXISTS dltJob;
DELIMITER $
CREATE TRIGGER dltJob
BEFORE DELETE ON job
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
```

```
   (OLD.job_evaluator,current_timestamp,'delete','success','job');

END $
DELIMITER ;

DROP TRIGGER IF EXISTS insEmployee;
DELIMITER $
CREATE TRIGGER insEmployee
AFTER INSERT ON employee
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (NEW.empl_username,current_timestamp,'insert','success','employee');

END $
DELIMITER ;


DROP TRIGGER IF EXISTS updtEmployee;
DELIMITER $
CREATE TRIGGER updtEmployee
AFTER UPDATE ON employee
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (NEW.empl_username,current_timestamp,'update','success','employee');

END $
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS dltEmployee;
DELIMITER $
CREATE TRIGGER dltEmployee
BEFORE DELETE ON employee
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (OLD.empl_username,current_timestamp,'delete','success','employee');

END $
DELIMITER ;



CREATE TRIGGER insRequestsevaluation
AFTER INSERT ON insRequestsevaluation
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (NEW.req_evaluator,current_timestamp,'insert','success','requestsevaluation');

END $
DELIMITER ;



DROP TRIGGER IF EXISTS updtRequestsevaluation;
DELIMITER $
CREATE TRIGGER updtRequestsevaluation
```

```
AFTER UPDATE ON job
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (NEW.req_evaluator,current_timestamp,'update','success','requestsevaluation');

END $
DELIMITER ;



DROP TRIGGER IF EXISTS dltRequestsevaluation;
DELIMITER $
CREATE TRIGGER dltRequestsevaluation
BEFORE DELETE ON job
FOR EACH ROW
BEGIN

INSERT INTO log (username,log_time,res,kind,table_name) VALUES
   (OLD.req_evaluator,current_timestamp,'delete','success','requestsevaluation');

END $
DELIMITER ;
```

```
INSERT INTO job VALUES

(3004,'2021-04-06',20000,'Senior  Programmer','Patra','fayboo','2021-05-14  15:30:00','2021-09-18 15:30:00');



 DELETE FROM job WHERE jobid=3004;



DROP TRIGGER IF EXISTS updateCompany;

DELIMITER $

CREATE TRIGGER updateCompany

BEFORE UPDATE ON company

FOR EACH ROW

BEGIN


SET NEW.name = OLD.name;

SET NEW.AFM = OLD.AFM;

SET NEW.DOY = OLD.DOY;


END $

DELIMITER ;


UPDATE company SET AFM='1256' WHERE AFM='1257';
```

```
DROP TRIGGER IF EXISTS updateUser;
DELIMITER $
CREATE TRIGGER updateUser
BEFORE UPDATE ON user
FOR EACH ROW
BEGIN

DECLARE finished_flag INT(4);
DECLARE admUsername VARCHAR(12);

DECLARE admn CURSOR FOR
 SELECT administratorUsername
 FROM administrator;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished_flag=1;

OPEN  admn;
SET finished_flag = 0;

WHILE(finished_flag=0) DO


FETCH admn INTO admUsername;

IF(OLD.username  LIKE admUsername ) THEN
        SET NEW.username = OLD.username;
        SET NEW.password = OLD.password;
        SET NEW.name = OLD.name;
```

```
        SET NEW.surname = OLD.surname;
        SET NEW.reg_date = OLD.reg_date;
        SET NEW.email = OLD.email;



END IF;


END WHILE;
CLOSE admn;


END $
DELIMITER ;



UPDATE user SET username='Fay' WHERE username='fayboo';
UPDATE user SET username='Mavra' WHERE username='mavrapol';
```