

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ ΣΤΟ ΕΡΓΑΣΤΗΡΙΟ ΤΟΥ ΜΑΘΗΜΑΤΟΣ “ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ”

ΚΟΛΛΙΑΣ ΙΩΑΝΝΗΣ

1064886

DROP DATABASE IF EXISTS staffevaluation;

CREATE DATABASE staffevaluation;

USE staffevaluation;

CREATE TABLE user(

    username VARCHAR(12) NOT NULL,

    password VARCHAR(5) NOT NULL,

    name VARCHAR(25),

    surname VARCHAR(35),

    email VARCHAR(30),

--  reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   /\* AYTOMATH XRHSH HMEROMHNIAS EGGRAFHS \*/

    reg\_date DATE,

    userkind ENUM('MANAGER','EVALUATOR','EMPLOYEE','ADMINISTRATOR'),

    INDEX UK(userkind),

    PRIMARY KEY (username)

);

CREATE TABLE company(

    AFM CHAR(9) NOT NULL,

    DOY VARCHAR(15),

    compname VARCHAR(35),

    phone BIGINT(16),

    street VARCHAR(15),

    num TINYINT(4),

    city VARCHAR (15),

    country VARCHAR(15),

    -- edra VARCHAR(55) generated always as (CONCAT(street,num,city,country)),

    -- INDEX EDRA(edra),

    PRIMARY KEY(AFM)

);

CREATE TABLE manager(

    manager\_username VARCHAR(12) NOT NULL,

    exp\_years TINYINT(4),

    AFM CHAR(9) NOT NULL,

    PRIMARY KEY (manager\_username),

    CONSTRAINT const1

    FOREIGN KEY(manager\_username)

    REFERENCES user(username)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const2

    FOREIGN KEY(AFM)

    REFERENCES company(AFM)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE evaluator(

    evaluator\_username VARCHAR(12) NOT NULL,

    AFM CHAR(9) NOT NULL,

    exp\_years TINYINT(4) NOT NULL,

    avr\_grade FLOAT(4,1),

    PRIMARY KEY (evaluator\_username),

    CONSTRAINT const3

    FOREIGN KEY(evaluator\_username)

    REFERENCES user(username)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const4

    FOREIGN KEY(AFM)

    REFERENCES company(AFM)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE employee(

    empl\_username VARCHAR(12) NOT NULL,

    AFM CHAR(9) NOT NULL,

    exp\_years TINYINT(4) NOT NULL,

    bio TEXT,

    sistatikes VARCHAR(35),

    certificates VARCHAR(35),

    awards VARCHAR(35),

    PRIMARY KEY(empl\_username),

    CONSTRAINT const5

    FOREIGN KEY(AFM)

    REFERENCES company(AFM)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const6

    FOREIGN KEY(empl\_username)

    REFERENCES user(username)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE languages(

    employee VARCHAR(12) NOT NULL,

    lang set('EN','FR','SP','GR'),

    PRIMARY KEY(employee,lang),

    CONSTRAINT const17

    FOREIGN KEY(employee)

    REFERENCES employee(empl\_username)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE job(

    job\_id INT(4) NOT NULL AUTO\_INCREMENT,

    AFM CHAR(9) NOT NULL,

    evaluator\_username VARCHAR(12) NOT NULL,

    salary FLOAT(6,1),

    position VARCHAR(40),

    edra VARCHAR(55),

    announcedate DATETIME DEFAULT NOW(),

    SubmissionDate DATE NOT NULL,

    PRIMARY KEY(job\_id,AFM),

    INDEX SUB(SubmissionDate),

    CONSTRAINT const7

    FOREIGN KEY(AFM)

    REFERENCES company(AFM)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const8

    FOREIGN KEY(evaluator\_username)

    REFERENCES evaluator(evaluator\_username)

    ON DELETE CASCADE ON UPDATE CASCADE

    -- CONSTRAINT const9

    -- FOREIGN KEY(edra)

    -- REFERENCES company(edra)

    -- ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE antikeim(

    antikeim\_title VARCHAR(36) NOT NULL,

    descr TINYTEXT,

    belongs\_to VARCHAR(36) NOT NULL,

    PRIMARY KEY(antikeim\_title),

    CONSTRAINT const10

    FOREIGN KEY(belongs\_to)

    REFERENCES antikeim(antikeim\_title)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE needs(

    job\_id INT(4) NOT NULL,

    antikeim\_title VARCHAR(36) NOT NULL,

    PRIMARY KEY(job\_id,antikeim\_title),

    CONSTRAINT const12

    FOREIGN KEY(job\_id)

    REFERENCES job(job\_id)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const11

    FOREIGN KEY(antikeim\_title)

    REFERENCES antikeim(antikeim\_title)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE project(

    num TINYINT(4) AUTO\_INCREMENT NOT NULL,

    empl\_username VARCHAR(12) NOT NULL,

    descr TEXT,

    url VARCHAR(60),

    PRIMARY KEY(num,empl\_username),

    UNIQUE (url),

    CONSTRAINT const13

    FOREIGN KEY(empl\_username)

    REFERENCES employee(empl\_username)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE availablejob(

    empl\_username VARCHAR(15) NOT NULL,

    job\_id INT(4) NOT NULL,

    PRIMARY KEY(empl\_username,job\_id),

    CONSTRAINT const23

    FOREIGN KEY(empl\_username)

    REFERENCES employee(empl\_username)

    ON UPDATE CASCADE ON DELETE CASCADE,

    CONSTRAINT const24

    FOREIGN KEY(job\_id)

    REFERENCES job(job\_id)

    ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE requestevaluation(

    empl\_username VARCHAR(12) NOT NULL,

    job\_id INT(4) NOT NULL,

    SubmissionDate DATE NOT NULL,

    empl\_interest BOOL DEFAULT FALSE,

    PRIMARY KEY(empl\_username,job\_id),

    CONSTRAINT const14

    FOREIGN KEY(empl\_username)

    REFERENCES availablejob(empl\_username)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const15

    FOREIGN KEY(job\_id)

    REFERENCES job(job\_id)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const16

    FOREIGN KEY(SubmissionDate)

    REFERENCES job(SubmissionDate)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE degree(

    titlos VARCHAR(50) NOT NULL,

    idryma VARCHAR(40) NOT NULL,

    INDEX IDEYM(idryma),

    numgraduates INT(4),

    bathmida ENUM('LYKEIO','UNIV','MASTER','PHD'),

    PRIMARY KEY(titlos,idryma)

);

CREATE TABLE has\_degree(

    empl\_username VARCHAR(12) NOT NULL,

    titlos VARCHAR(50) NOT NULL,

    idryma VARCHAR(40) NOT NULL,

    etos YEAR(4),

    grade FLOAT(3,1),

    bathmida ENUM('LYKEIO','UNIV','MASTER','PHD'),

    PRIMARY KEY(titlos,idryma,empl\_username),

    CONSTRAINT const20

    FOREIGN KEY(empl\_username)

    REFERENCES employee(empl\_username)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE evaluationresult(

    Evld INT(4) NOT NULL,

    empl\_username VARCHAR(12) NOT NULL,

    evaluator\_username VARCHAR(12) NOT NULL,

    job\_id INT(4),

    F1 INT(4),

    F2 INT(4),

    F3 INT(4),

    grade INT(4),

    comments VARCHAR(255),

    PRIMARY KEY(Evld,empl\_username),

    CONSTRAINT const21

    FOREIGN KEY(empl\_username)

    REFERENCES requestevaluation(empl\_username)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const22

    FOREIGN KEY(job\_id)

    REFERENCES requestevaluation(job\_id)

    ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE logs(

    order\_of\_action int(8) auto\_increment not null,

    username VARCHAR(12) NOT NULL,

    userkind ENUM('MANAGER','EVALUTOR','EMPLOYEE','ADMINISTRATOR') NOT NULL,

    table\_of\_incident ENUM('job','employee','evaluationresult') NOT NULL,

    time\_of\_incident DATETIME,

    type\_of\_incident ENUM('INSERT','UPDATE','DELETE') not null,

    success enum('YES','NO'),

    PRIMARY KEY(order\_of\_action,username),

    CONSTRAINT const26

    FOREIGN KEY(username)

    REFERENCES user(username)

    ON DELETE CASCADE ON UPDATE CASCADE,

    CONSTRAINT const25

    FOREIGN KEY(userkind)

    REFERENCES user(userkind)

    ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO company VALUES('143792558','NAYFPLIO','SaillokStudio','2752017613','Omiroy','26','Nafplio','Greece'/\*,DEFAULT\*/);

INSERT INTO company VALUES('268926487','AMAROYSIOY','Oikomat','2103558645','Eyripidi','1','A8hna','Greece'/\*,DEFAULT\*/);

INSERT INTO company VALUES('197832746','LIBADEIAS','ArgoFarm','2261085946','Konstantinoy','102','Livadeia','Greece'/\*,DEFAULT\*/);

INSERT INTO user VALUES('saillok','12345','IOANNIS','KOLLIAS','saillok@gmail.com','1999-09-18','MANAGER');

INSERT INTO user VALUES('alexiou','12345','STAVROS','ALEXIOU','aleksiou@gmail.com','2005-11-27','MANAGER');

INSERT INTO user VALUES('petselis','12345','IOANNIS','PETSELIS','petselis@gmail.com','2012-05-03','MANAGER');

INSERT INTO user VALUES('xoulis','12345','TA3IARXHS','LYGIZOS','xoulis@gmail.com','2003-09-12','MANAGER');

INSERT INTO user VALUES('dorzi','12345','NESLIE','DORZI','dorzi@gmail.com','1997-01-19','MANAGER');

INSERT INTO user VALUES('kerkidoy','12345','KONSTANTINA','KERKIDOY','kerkidoy@gmail.com','2000-01-07','EVALUATOR');

INSERT INTO user VALUES('papagiannhs','12345','PAULOS','PAPAGIANNHS','papagiannhs@gmail.com','2008-12-19','EVALUATOR');

INSERT INTO user VALUES('papanikolaou','12345','KOSTAS','PAPANIKOLAOU','papanikolaou@gmail.com','2004-02-22','EVALUATOR');

INSERT INTO user VALUES('afentakh','12345','FLWRENTIA','AFENTAKH','afentakh@gmail.com','2010-10-28','EMPLOYEE');

INSERT INTO user VALUES('paylidh','12345','SOFIA','PAYLIDH','paylidh@gmail.com','2016-03-15','EMPLOYEE');

INSERT INTO user VALUES('papagewrgioy','12345','THANOS','PAPAGEWRGIOY','papagewrgioy@gmail.com','2013-05-14','EMPLOYEE');

INSERT INTO user VALUES('spandwnh','12345','HLIANA','SPANDWNH','spandwnh@gmail.com','2017-04-13','EMPLOYEE');

INSERT INTO user VALUES('karagiannhs','12345','HLIAS','KARAGIANNHS','karagiannhs@gmail.com','2020-10-17','EMPLOYEE');

INSERT INTO user VALUES('mpakalhs','12345','SWTHRHS','MPAKALHS','mpakalhs@gmail.com','2019-11-23','EMPLOYEE');

INSERT INTO user VALUES('ntova','12345','RAFAHLIA','NTOVA','ntova@gmail.com','2018-01-30','EMPLOYEE');

INSERT INTO user VALUES('lame','12345','MPROUNA','LAME','lame@gmail.com','2016-02-25','EMPLOYEE');

INSERT INTO user VALUES('kallaras','12345','NTINOS','KALLARAS','kallaras@gmail.com','2015-06-20','EMPLOYEE');

INSERT INTO user VALUES('mellos','12345','THODORIS','MELLOS','mellos@gmail.com','1997-05-11','ADMINISTRATOR');

INSERT INTO manager VALUES('saillok',15,'143792558');

INSERT INTO manager VALUES('alexiou',9,'143792558');

INSERT INTO manager VALUES('petselis',23,'268926487');

INSERT INTO manager VALUES('xoulis',18,'268926487');

INSERT INTO manager VALUES('dorzi',13,'197832746');

INSERT INTO evaluator VALUES('kerkidoy','143792558',8,NULL);

INSERT INTO evaluator VALUES('papagiannhs','268926487',5,NULL);

INSERT INTO evaluator VALUES('papanikolaou','197832746',10,NULL);

INSERT INTO employee VALUES('afentakh','143792558',5,'phre me meso th doyleia','ISBL,Seminario prwtwn voh8eiwn',NULL,NULL);

INSERT INTO employee VALUES('paylidh','143792558',10,NULL,NULL,NULL,NULL);

INSERT INTO employee VALUES('papagewrgioy','143792558',12,NULL,NULL,NULL,NULL);

INSERT INTO employee VALUES('spandwnh','143792558',2,NULL,NULL,NULL,NULL);

INSERT INTO employee VALUES('karagiannhs','268926487',6,NULL,NULL,NULL,NULL);

INSERT INTO employee VALUES('mpakalhs','268926487',8,NULL,NULL,NULL,NULL);

INSERT INTO employee VALUES('ntova','268926487',15,NULL,NULL,NULL,NULL);

INSERT INTO employee VALUES('lame','197832746',3,NULL,NULL,NULL,NULL);

INSERT INTO employee VALUES('kallaras','197832746',1,NULL,NULL,NULL,NULL);

INSERT INTO job(AFM,evaluator\_username,salary,position,edra,SubmissionDate) VALUES('143792558','kerkidoy',1200,'Omiroy 26 Nafplio Greece','hxolhpths','2021-10-19');

INSERT INTO job(AFM,evaluator\_username,salary,position,edra,SubmissionDate) VALUES('143792558','kerkidoy',1300,'Omiroy 26 Nafplio Greece','Drumer','2021-05-08');

INSERT INTO job(AFM,evaluator\_username,salary,position,edra,SubmissionDate) VALUES('268926487','papagiannhs',900,'Eyripidi 1 A8hna Greece','Pwlhths','2021-11-06');

INSERT INTO job(AFM,evaluator\_username,salary,position,edra,SubmissionDate) VALUES('268926487','papagiannhs',1250,'Eyripidi 1 A8hna Greece','Synthrhths','2021-02-10');

INSERT INTO job(AFM,evaluator\_username,salary,position,edra,SubmissionDate) VALUES('197832746','papanikolaou',1400,'Konstantinoy 102 Livadeia Greece','Geoponos','2021-04-19');

INSERT INTO job(AFM,evaluator\_username,salary,position,edra,SubmissionDate) VALUES('197832746','papanikolaou',850,'Konstantinoy 102 Livadeia Greece','Pwlhths','2021-08-25');

INSERT INTO has\_degree VALUES ('afentakh','COMPUTERS','PLHROFORIKH',2009,8,'MASTER');

INSERT INTO has\_degree VALUES ('paylidh','COMPUTERS','CEID',2006,7,'MASTER');

-- INSERT INTO needs(job\_id,antikeim\_title) VALUES (1,'SQL');

-- INSERT INTO needs(job\_id,antikeim\_title) VALUES (1,'C');

-- INSERT INTO antikeim(antikeim\_title) VALUES ('SQL');

SET @current\_username='';

SET @current\_userkind='';

/\* ERWTHMA 4c \*/

DELIMITER $

CREATE TRIGGER UsernameChange

BEFORE UPDATE ON user

FOR EACH ROW

BEGIN

    IF (@current\_userkind<>'ADMINISTRATOR') THEN

        IF (OLD.username<>NEW.username OR OLD.userkind<>NEW.userkind OR OLD.reg\_date<>NEW.reg\_date OR OLD.name<>NEW.name OR OLD.surname<>NEW.surname) THEN

            SIGNAL SQLSTATE VALUE '45000' SET message\_text = 'DEN EXEIS DIKAIWMATA NA ALLA3EIS AYTHN TIMH';

        END IF;

    END IF;

END$

DELIMITER ;

/\*

CREATE TRIGGER InsertDate

AFTER INSERT ON user

FOR EACH ROW

SET reg\_date=CURDATE();

vgazei sfalma reg\_date

\*/

/\* ERWTHMA 4b \*/

DELIMITER $

CREATE TRIGGER UnchangeableColumns

BEFORE UPDATE ON company

FOR EACH ROW

BEGIN

    IF (NEW.AFM<>OLD.AFM) THEN

        SIGNAL SQLSTATE VALUE '45000' SET message\_text =  'H TIMH AYTH DEN ALLAZEI.';

        Set NEW.AFM=OLD.AFM;

    END IF;

    IF (NEW.DOY<>OLD.DOY) THEN

        SIGNAL SQLSTATE VALUE '45000' SET message\_text =  'H TIMH AYTH DEN ALLAZEI.';

        Set NEW.AFM=OLD.AFM;

    END IF;

    IF (NEW.compname <> OLD.compname) THEN

        SIGNAL SQLSTATE VALUE '45000' SET message\_text =  'H TIMH AYTH DEN ALLAZEI.';

        Set NEW.compname=OLD.compname;

     END IF;

END$

DELIMITER ;

/\* ERWTHMA 4a \*/

/\* INSERT UPDATE DELETE EMPLOYEE \*/

DELIMITER $

CREATE TRIGGER EmplInsertLog

AFTER INSERT ON employee

FOR EACH ROW

BEGIN

    CALL SuccessLog(@current\_userkind,'employee',@current\_username,'INSERT');

END$

DELIMITER ;

DELIMITER $

CREATE TRIGGER EmplUpdateLog

AFTER UPDATE ON employee

FOR EACH ROW

BEGIN

    CALL SuccessLog(@current\_userkind,'employee',@current\_username,'UPDATE');

END$

DELIMITER ;

DELIMITER $

CREATE TRIGGER EmplDeleteLog

AFTER DELETE ON employee

FOR EACH ROW

BEGIN

    CALL SuccessLog(@current\_userkind,'employee',@current\_username,'DELETE');

END$

DELIMITER ;

/\* INSERT UPDATE DELETE DEGREES exw 8ema degr\_idryma \*/

DELIMITER $

CREATE TRIGGER InsertDegree

AFTER INSERT ON has\_degree

FOR EACH ROW

BEGIN

    INSERT INTO degree(titlos,idryma,numgraduates) VALUES (NEW.titlos,NEW.idryma,1)

        ON DUPLICATE KEY UPDATE numgraduates=numgraduates+1;

END$

CREATE TRIGGER DeleteDegree

AFTER DELETE ON has\_degree

FOR EACH ROW

BEGIN

    DECLARE deleted\_numgraduates int(4);

    SELECT numgraduates INTO deleted\_numgraduates FROM degree WHERE degree.titlos=OLD.titlos AND degree.idryma=OLD.idryma;

    SET deleted\_numgraduates=deleted\_numgraduates-1;

    IF (deleted\_numgraduates = 0) THEN

        DELETE FROM degree WHERE has\_degree.titlos=OLD.titlos AND has\_degree.idryma=OLD.idryma;

    ELSE

        UPDATE degree SET numgraduates=deleted\_numgraduates WHERE degree.titlos=OLD.titlos AND degree.idryma=OLD.idryma;

    END IF;

END$

CREATE TRIGGER UpdateDegree

AFTER UPDATE ON has\_degree

FOR EACH ROW

BEGIN

    DECLARE deleted\_numgraduates int(4);

    IF (OLD.titlos <> NEW.titlos or OLD.idryma<>NEW.idryma) THEN

        SELECT numgraduates INTO deleted\_numgraduates FROM degree WHERE degree.titlos=OLD.titlos AND degree.idryma=OLD.idryma;

        SET deleted\_numgraduates=deleted\_numgraduates-1;

        IF (deleted\_numgraduates = 0) THEN

            DELETE FROM degree WHERE has\_degree.titlos=Old.titlos and  has\_degree.idryma=OLD.idryma;

        ELSE

            UPDATE degree set numgraduates=deleted\_numgraduates WHERE degree.titlos=OLD.titlos and degree.idryma=OLD.idryma;

        END IF;

        INSERT INTO degree(titlos,idryma,numgraduates) VALUES (NEW.titlos,NEW.idryma,1)

        ON DUPLICATE KEY UPDATE numgraduates=numgraduates+1;

    END IF;

END$

DELIMITER ;

/\* APARAITHTES DHLWSEIS GIA JOB \*/

DELIMITER $

CREATE TRIGGER SendAvailability

AFTER INSERT ON job

FOR EACH ROW

BEGIN

    DECLARE finished bool;

    DECLARE endex\_requester VARCHAR(12);

    DECLARE SendCursor CURSOR FOR SELECT empl\_username FROM employee WHERE AFM=NEW.AFM;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=FALSE;

    OPEN SendCursor;

    SET finished=TRUE;

    FETCH SendCursor INTO endex\_requester;

    WHILE (finished=TRUE) DO

        INSERT INTO requestevaluation(empl\_username,job\_id,SubmissionDate) VALUES (endex\_requester,NEW.job\_id,NEW.SubmissionDate);

        FETCH SendCursor INTO endex\_requester;

    END WHILE;

    CLOSE SendCursor;

END$

DELIMITER ;

DELIMITER $

CREATE TRIGGER SAUpdatedDate

AFTER UPDATE ON job

FOR EACH ROW

BEGIN

    DECLARE finished bool;

    DECLARE endex\_requester VARCHAR(12);

    DECLARE SendCursor CURSOR FOR SELECT empl\_username FROM employee WHERE AFM=NEW.AFM;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=FALSE;

    IF (OLD.SubmissionDate<>NEW.SubmissionDate) THEN

        OPEN SendCursor;

        SET finished=TRUE;

        FETCH SendCursor INTO endex\_requester;

        WHILE (finished=TRUE) do

            UPDATE requestevaluation SET SubmissionDate=NEW.SubmissionDate

            WHERE requestevaluation.empl\_username=endex\_requester AND requestevaluation.job\_id=NEW.job\_id;

            FETCH SendCursor INTO endex\_requester;

        END WHILE;

    END IF;

    CLOSE SendCursor;

END $

DELIMITER ;

/\* INSERT UPDATE DELETE JOB \*/

DELIMITER $

CREATE TRIGGER JobInsertLog

AFTER INSERT ON job

FOR EACH ROW

BEGIN

    CALL SuccessLog('job','INSERT');

END$

CREATE TRIGGER JobUpdateLog

AFTER UPDATE ON job

FOR EACH ROW

BEGIN

    CALL SuccessLog('job','UPDATE');

END$

CREATE TRIGGER JobDeleteLog

AFTER DELETE ON job

FOR EACH ROW

BEGIN

    CALL SuccessLog('job','DELETE');

END$

DELIMITER ;

/\* APARAITHTES DHLWSEIS GIA EVALUATION RESULT \*/

DELIMITER $

CREATE TRIGGER UnchangeableGrade

BEFORE UPDATE ON evaluationresult

FOR EACH ROW

BEGIN

    IF (OLD.grade <> NEW.grade AND OLD.grade IS NOT NULL AND @current\_userkind<>'ADMINISTRATOR') THEN

        CALL FailureLog('evaluationresult','UPDATE');

        SIGNAL SQLSTATE VALUE '45000' SET message\_text = 'O vathmos den mporei na allaxtei ap th stigmh poy exei oristikopoih8ei.';

    END IF;

END$

DELIMITER ;

/\* INSERT UPDATE DELETE EVALUATION RESULT \*/

DELIMITER $

CREATE TRIGGER evaluationresultInsertLog

AFTER INSERT ON evaluationresult

FOR EACH ROW

BEGIN

    CALL SuccessLog('evaluationresult','INSERT');

END$

CREATE TRIGGER evaluationresulteUpdateLog

AFTER UPDATE ON evaluationresult

FOR EACH ROW

BEGIN

    CALL SuccessLog('evaluationresult','UPDATE');

END$

CREATE TRIGGER evaluationresultDeleteLog

AFTER DELETE ON evaluationresult

FOR EACH ROW

BEGIN

    CALL SuccessLog('evaluationresult','DELETE');

END$

DELIMITER ;

/\* SUCCESS / FAILURE LOG \*/

DELIMITER $

CREATE PROCEDURE SuccessLog(IN table\_of\_incident ENUM('job','employee','evaluationresult'), IN type\_of\_incident ENUM('INSERT','UPDATE','DELETE'))

BEGIN

    INSERT INTO  logs(userkind,table\_of\_incident,username,time\_of\_incident,type\_of\_incident,success) VALUES (@current\_userkind,table\_of\_incident,@current\_username,now(),type\_of\_incident,'YES');

END$

CREATE PROCEDURE FailureLog(IN table\_of\_incident ENUM('job','employee','evaluationresult'), IN type\_of\_incident ENUM('INSERT','UPDATE','DELETE'))

BEGIN

    INSERT INTO logs(userkind,table\_of\_incident,username,time\_of\_incident,type\_of\_incident,success) VALUES (@current\_userkind,table\_of\_incident,@current\_username,now(),type\_of\_incident,'NO');

END$

DELIMITER ;

/\* Stored Procedures \*/

/\* A \*/

DELIMITER $

CREATE PROCEDURE LoginAccount(IN endex\_username VARCHAR(12),in endex\_password VARCHAR(5))

BEGIN

/\*KA8ARISMOS DEDOMENWN\*/

    SELECT "Disconnecting from previous user...";

    SET @current\_username=NULL;

    SET @current\_password=NULL;

    SET @current\_userkind=NULL;

    SET @current\_username=endex\_username;

    SET @current\_password=endex\_password;

    /\*PSAXNW STO user.ADMINISTRATOR YPAREXEI.\*/

    SELECT userkind INTO @current\_userkind FROM user WHERE username=@current\_username AND password=@current\_password;

    /\* IF userkind=NULL => DE BRE8HKE\*/

     IF (@current\_userkind IS NULL) THEN

        SELECT 'DE BRE8HKE TO SYGKEKRIMENO ACCOUNT.PLHKTROLOGEISTE "CALL LoginAccount(your Username,your Password);"';

        ELSE

        SELECT 'SYNDE8HKATE EPITYXWS ',@current\_username,',H IDIOTHTA SAS EIANI:',@current\_userkind;

    END IF;

 END$

 DELIMITER ;

/\* B \*/

DELIMITER $

CREATE PROCEDURE Average (in specific\_evaluator\_username VARCHAR(12))

BEGIN

    DECLARE specific\_grade INT;

    DECLARE finished bool;

    DECLARE numloopsfinalization int(4);

    DECLARE specific\_avg FLOAT(4,1);

    DECLARE EvalCursor CURSOR FOR SELECT grade  FROM  evaluationresult WHERE evaluationresult.evaluator\_username= specific\_evaluator\_username AND evaluationresult.grade IS NOT NULL;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=false;

    SET specific\_avg=0;

    SET numloopsfinalization=0;

    OPEN EvalCursor;

    SET finished=true;

    FETCH EvalCursor INTO specific\_grade;

    WHILE (finished=true) DO

        SET numloopsfinalization=numloopsfinalization+1;

        SET specific\_avg=specific\_avg+specific\_grade;

        FETCH EvalCursor INTO specific\_grade;

    END WHILE;

CLOSE EvalCursor;

SET specific\_avg=specific\_avg/numloopsfinalization;

UPDATE evaluator SET evaluator.avr\_grade=specific\_avg WHERE evaluator.evaluator\_username=specific\_evaluator\_username;

END$

DELIMITER ;

/\* C \*/

DELIMITER $

CREATE PROCEDURE RequestEvaluation(IN req\_empl\_username VARCHAR(12),IN req\_job\_id INT(8))

BEGIN

DECLARE req\_SubmissionDate DATE;

DECLARE req\_evaluator\_username VARCHAR(12);

SELECT SubmissionDate INTO req\_SubmissionDate FROM requestevaluation WHERE requestevaluation.empl\_username=req\_empl\_username AND requestevaluation.job\_id=req\_job\_id;

IF  (CURDATE()<=req\_SubmissionDate) THEN

    SELECT evaluator\_username into req\_evaluator\_username FROM evaluator WHERE job.job\_id=req\_job\_id;

    INSERT INTO evaluationresult (empl\_username,job\_id,evaluator\_username) VALUES (req\_empl\_username,req\_job\_id,req\_evaluator\_username);

    UPDATE requestevaluation

    SET empl\_interest=TRUE WHERE requestevaluation.empl\_username=req\_empl\_username AND requestevaluation.job\_id=req\_job\_id;

ELSE

    SELECT 'ELH3E TO XRONIKO DIASTHMA YPOBOLHS GIA AYTHN TH DOYLEIA.';

END IF;

END$

DELIMITER ;

/\* D \*/

DELIMITER $

CREATE PROCEDURE FinEvaluations (IN Particular\_job\_id INT)

BEGIN

DECLARE Particular\_empl\_username VARCHAR(12);

DECLARE Particular\_evaluator\_username VARCHAR(12);

DECLARE Particular\_F1 INT(4);

DECLARE Particular\_F2 INT(4);

DECLARE Particular\_F3 INT(4);

DECLARE Particular\_grade INT(4);

DECLARE finished bool;

DECLARE FinCursor CURSOR FOR SELECT username\_employee,username\_evaluator,phase1,phase2,phase3,Grade FROM evaluationresult WHERE evaluationresult.job\_id=Particular\_job\_id ;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=false;

OPEN FinCursor;

SET finished=true;

FETCH FinCursor INTO Particular\_empl\_username,Particular\_evaluator\_username,Particular\_F1,Particular\_F2,Particular\_F3,Particular\_grade;

WHILE (finished=true) DO

    IF (Particular\_F1 IS NOT NULL AND Particular\_F2 IS NOT NULL AND Particular\_F3 IS NOT NULL AND Particular\_grade IS  NULL) THEN

        SET Particular\_grade=Particular\_F1+Particular\_F2+Particular\_F3 ;

        SELECT Particular\_grade;

        UPDATE evaluationresult

        SET grade=Particular\_grade WHERE empl\_username=Particular\_empl\_username AND job\_id=Particular\_job\_id;

        SELECT Particular\_evaluator\_username AS 'EMPLOYEE', Particular\_grade as 'grade';

    END IF;

    FETCH FinCursor INTO Particular\_empl\_username,Particular\_evaluator\_username,Particular\_F1,Particular\_F2,Particular\_F3,Particular\_grade;

END WHILE;

CLOSE FinCursor;

END$

DELIMITER ;

/\* E \*/

DELIMITER $

CREATE PROCEDURE FinishedEvaluations (in Demanded\_job\_id int )

BEGIN

   DECLARE NumRequests INT(8);

   DECLARE NumUnansweredRequests INT(8);

   Select COUNT(job\_id) INTO NumRequests FROM evaluationresult WHERE job\_id=Demanded\_job\_id;

   SELECT COUNT(job\_id) INTO NumUnansweredRequests FROM evaluationresult WHERE job\_id=Demanded\_job\_id AND grade IS NULL;

   IF  (NumRequests>0) then               /\*Diladi uparxoun request pou exonu ginei ews tora gia na proxorisoume\*/

      IF (NumUnansweredRequests=0) then Select 'OLA TA AITHMATA GIA AYTHN TH DOYLEIA EXOYN A3IOLOGH8EI PLHRWS.';

      END IF;

      SELECT empl\_username AS 'Candidate',grade FROM evaluationresult WHERE job\_id=Demanded\_job\_id AND grade IS NOT NULL ORDER BY grade Desc;

      /\*Dinoume sto username eidiko onoma kai oxi sto grade,giati to grade einai column eidiko

      gia to table auto eno to empl\_username einai genika opou uparxei username ypallilou\*/

      IF (NumUnansweredRequests>0) THEN SELECT NumUnansweredRequests;

      END IF;

   ELSE SELECT 'DEN EXEI ZHTHSEI KANEIS AYTH TH 8ESH.';

   END IF ;

END $

DELIMITER ;

/\* ST \*/

DELIMITER $

CREATE PROCEDURE ParticularEmployeeRequests (in particular\_name varchar(25),in particular\_surname varchar(35))

BEGIN

    DECLARE particular\_job\_id INT(8);

    DECLARE particular\_grade INT(4);

    DECLARE finished TINYINT(2);

    DECLARE particular\_evaluator\_username VARCHAR(12);

    DECLARE particular\_empl\_username VARCHAR(12);

    DECLARE ReqCursor CURSOR FOR SELECT job\_id,grade,evaluator\_username FROM evaluationresult WHERE evaluationresult.empl\_username=particular\_empl\_username;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=1;

    SELECT username INTO particular\_empl\_username FROM user WHERE user.name=Particular\_name and user.surname=Particular\_surname AND userkind='EMPLOYEE';

    OPEN ReqCursor;

    SET finished=0;

    FETCH ReqCursor INTO particular\_job\_id,particular\_grade,particular\_evaluator\_username;

    WHILE (finished=0) DO

       SELECT particular\_job\_id AS 'Doyleia poy zhth8hke ap ton ypallhlo.',particular\_grade AS 'Vathmos,An to aithma exei a3iologh8ei plhrws.';

       SELECT name,surname AS 'O a3iologhths ths doyleias' FROM user WHERE particular\_evaluator\_username=user.username and userkind='EVALUATOR';

       FETCH ReqCursor INTO particular\_job\_id,particular\_grade,particular\_evaluator\_username;

    END WHILE;

    CLOSE ReqCursor;

END$

DELIMITER ;

Ακολουθεί η περιγραφή του κώδικα. Οι προσθέσεις και αλλαγές της βάσης φαίνονται παρακάτω.

Αρχικά καλείται το Procedure LoginAccount που φαίνεται τι δικαιώματα έχει ο κάθε user στη βάση δεδομένων. Με τα current\_username και current\_userkind στο login αυτόματα βρίσκω πού κατατάσσεται ο εκάστοτε χρήστης. Ο Administrator έχει ήδη τοποθετηθεί στη βάση και έχει τα περισσότερα δικαιώματα. Το current\_username χρησιμοποιείται κυρίως στη Log ενώ το current\_userkind περιορίζει την πρόσβαση σε όποια είδη χρηστών δεν έχουν τα δικαιώματα να αλλάξουν στοιχεία της βάσης, μέσω trigger. Συγκεκριμένα τα trigger Username Change και UnchangeableColumns είναι αυτά που δημιουργούν τους περιορισμούς αλλαγής στοιχείων. Οι δύο παραπάνω μεταβλητές είναι session variable, δηλαδή παραμένουν ενεργές έως ότου κλείσουμε τη σύνδεση ή ξανακαλέσουμε την “LoginAccount”.

Στον πίνακα user πρόσθεσα τη στήλη userkind έτσι ώστε να είναι πιο εύκολο να βρούμε κάποιο username και να μην ψάχνουμε στον κάθε πίνακα ξεχωριστά (employee, evaluator κ.λ.π).

Για τη στήλη edra χρησιμοποίησα την συνάρτηση CONCAT έτσι ώστε να προστεθούν τα διάφορα strings σε μία στήλη (country, city, street, num). Τέλος αναφορικά με το table company μέσω του trigger UnchangeableColumns κρατά αναλλοίωτες τις αλλαγές σε AFM,DOY και compname απ’ όποιον προσπαθεί να τις αλλάξει και δεν είναι administrator. Δυστυχώς η στήλη edra δεν λειτούργησε ως FK παρά μόνο στο xampp, ακριβώς επειδή έχω προσπαθήσει και το gui βρίσκεται στον κώδικά μου αλλά σε σχόλιο.

Μέσω του Procedure Average φαίνονται στη στήλη avr\_grade οι βαθμολογίες που οριστικοποιήθηκαν από αναφορές του συγκεκριμένου αξιολογητή. Το procedure αυτό παίρνει το username του evaluator απ’ όλα τα request\_evaluation σε job υπό την εποπτεία του και αποθηκεύεται στο avr\_grade.

**ΣΗΜΕΙΩΣΗ:** Πριν δείτε στο Select, τρέξτε τη Average ώστε να φανεί ο καινούριος μέσος όρος.

Στο table has\_degree περιέχει ένα πτυχίο που μπορεί να έχει ένας χρήστης. Ο κάθε employee μπορεί να έχει πολλά πτυχία. Απ’ τη στιγμή που το primary key είναι το empl\_username, titlos, idryma δεν τίθεται ζήτημα μοναδικότητας.

Απ’ την άλλη στο degree φαίνεται τι πτυχία έχουν οι χρήστες που εξετάζονται και πόσοι έχουν το ίδιο πτυχίο. Έτσι σε αυτό το table προστέθηκε και to column numgraduates.

Θα παρατηρήσετε πως δεν υπάρχουν τα constraint για τα δύο tables και αυτό γιατί δημιουργούνταν πολλές συγκρούσεις αναφορικά με τη μοναδικότητα των κλειδιών. Για αυτό χρησιμοποιώ triggers insert, delete, update. Το degree παίρνει από το insert του has\_degree, οπότε δεν ενημερώνεται απ’ το χρήστη. Αν υπάρχει ήδη τιμή στο degree με ίδιο titlos και idryma, τότε numgraduates +. Αν γίνει delete ένα στοιχείο του has\_degree, το numgraduates θα μειωθεί κατά 1. Αν γίνει update θα αφαιρεθεί ένα στα OLD titlos και idryma και θα προστεθεί ένα στα NEW titlos και idryma.

Στο table job, η στήλη job\_id έχει οριστεί με auto\_increment για να παίρνει αυτόματα κάποιον αριθμό ως id της δουλειάς. Το ίδιο ισχύει και με το announcedate. Υπάρχει Trigger για την επιτυχή ενημέρωση στο log.

Το SendAvailability trigger χρησιμοποιείται έτσι ώστε όταν προστεθεί καινούρια δουλειά, προσθέτει για κάθε εργαζόμενο που ανήκει στην εταιρία όπου δημιουργήθηκε η δουλειά, ένα insert στον πίνακα request. Απ’ την άλλη μεριά στο SAUpdatedDate αν αλλάξει το SubmissionDate ενημερώνει κάθε στοιχείο του requestevaluation από το συγκεκριμένο id\_job.

Στο requestevaluation έχω προσθέσει μία Boolean στήλη ( empl\_interest ) για τη δήλωση ενδιαφέροντος. Καλώντας το procedure RequestEvaluation από τον ενδιαφερόμενο υπάλληλο ( αν δεν έχει λήξει η ημερομηνία αιτήσεων ), δημιουργείται αυτόματα insert στο table evaluationresult και το Boolean empl\_interested γίνεται true.

Στο table evaluationresult ο κάθε εργαζόμενος μπορείς να στείλει μόνο μία αναφορά για μία δουλειά. Τα triggers evaluationresult(Insert/Update/Delete)Log είναι για να ενημερώσουν το table logs.

Το grade στο table evaluationresult δεν συμπληρώνεται τη στιγμή που παίρνουν τιμές τα διάφορα F1, F2, F3, αλλά ο evaluator καλεί την FinEvaluations. Αν έχει τιμή το grade, δεν μπορεί να αλλαχτεί χάρη στο Trigger UnchangeableGrade το οποίο πρέπει να καλεστεί απ’ το χρήστη.

Το procedure FinishedEvaluations δείχνει τον αριθμό όσων αιτημάτων για δουλειές που έχουν απαντηθεί δείχνοντας και τους υπαλλήλους που την έχουν ζητήσει και την βαθμολογία που τους δόθηκε, αλλά και τον αριθμό όσων δεν έχουν απαντηθεί.

Το τελευταίο procedure ParticularEmployeeRequests είναι μια μορφή αναζήτησης με βάση το ονοματεπώνυμο του υπαλλήλου έτσι ώστε να βρίσκουμε όλες τις δουλειές που έχει ζητήσει για αξιολόγηση, και για όσες υπάρχουν, παρουσιάζεται και η βαθμολογία αλλά και ο evaluator για τη δουλειά αυτή.

Τέλος αξίζει να σημειώσω τη χρήση των Trigger Success και Failure log για όπου για “YES” ή “NO” ενημερώνονται οι αντίστοιχες στήλες, και από τα διάφορα triggers που έχουν τη λέξη log ενημερωνόμαστε για την επιτυχία της ενημέρωσης.

**ΠΑΡΑΤΗΡΗΣΗ:** Λόγω των @current\_username και @current\_userkind δεν χρειάζεται να γράφουμε κάθε φορα τα στοιχεία του user αλλά και η order\_of\_action είναι αυτόματη.