ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ ΣΤΟ ΕΡΓΑΣΤΗΡΙΟ ΤΟΥ ΜΑΘΗΜΑΤΟΣ
“ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ”

ΚΟΛΛΙΑΣ ΙΩΑΝΝΗΣ

1064886

```sql
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 HM
EROMHNIAS 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 NUL
L,
    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','2752
017613','Omiroy','26','Nafplio','Greece'/*,DEFAULT*/);
INSERT INTO company VALUES('268926487','AMAROYSIOY','Oikomat','21035586
45','Eyripidi','1','A8hna','Greece'/*,DEFAULT*/);
INSERT INTO company VALUES('197832746','LIBADEIAS','ArgoFarm','22610859
46','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','petsel
is@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.c
om','1997-01-19','MANAGER');

INSERT INTO user VALUES('kerkidoy','12345','KONSTANTINA','KERKIDOY','ke
rkidoy@gmail.com','2000-01-07','EVALUATOR');
INSERT INTO user VALUES('papagiannhs','12345','PAULOS','PAPAGIANNHS','p
apagiannhs@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,NUL
L);


INSERT INTO job(AFM,evaluator_username,salary,position,edra,SubmissionD
ate) VALUES('143792558','kerkidoy',1200,'Omiroy 26 Nafplio Greece','hxo
lhpths','2021-10-19');
INSERT INTO job(AFM,evaluator_username,salary,position,edra,SubmissionD
ate) VALUES('143792558','kerkidoy',1300,'Omiroy 26 Nafplio Greece','Dru
mer','2021-05-08');
INSERT INTO job(AFM,evaluator_username,salary,position,edra,SubmissionD
ate) VALUES('268926487','papagiannhs',900,'Eyripidi 1 A8hna Greece','Pw
lhths','2021-11-06');
INSERT INTO job(AFM,evaluator_username,salary,position,edra,SubmissionD
ate) VALUES('268926487','papagiannhs',1250,'Eyripidi 1 A8hna Greece','S
ynthrhths','2021-02-10');
INSERT INTO job(AFM,evaluator_username,salary,position,edra,SubmissionD
ate) VALUES('197832746','papanikolaou',1400,'Konstantinoy 102 Livadeia
Greece','Geoponos','2021-04-19');
INSERT INTO job(AFM,evaluator_username,salary,position,edra,SubmissionD
ate) VALUES('197832746','papanikolaou',850,'Konstantinoy 102 Livadeia G
reece','Pwlhths','2021-08-25');


INSERT INTO has_degree VALUES ('afentakh','COMPUTERS','PLHROFORIKH',200
9,8,'MASTER');
INSERT INTO has_degree VALUES ('paylidh','COMPUTERS','CEID',2006,7,'MAS
TER');


-- 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.s
urname) 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,'INS
ERT');
END$
DELIMITER ;

DELIMITER $
CREATE TRIGGER EmplUpdateLog
AFTER UPDATE ON employee
FOR EACH ROW
BEGIN
    CALL SuccessLog(@current_userkind,'employee',@current_username,'UPD
ATE');
END$
DELIMITER ;

DELIMITER $
CREATE TRIGGER EmplDeleteLog
AFTER DELETE ON employee
FOR EACH ROW
BEGIN
    CALL SuccessLog(@current_userkind,'employee',@current_username,'DEL
ETE');
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,N
EW.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 deg
ree.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_d
egree.idryma=OLD.idryma;
    ELSE
        UPDATE degree SET numgraduates=deleted_numgraduates WHERE degre
e.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 d
egree.titlos=OLD.titlos and degree.idryma=OLD.idryma;
        END IF;
        INSERT INTO degree(titlos,idryma,numgraduates) VALUES (NEW.titl
os,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 WH
ERE 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,SubmissionDa
te) 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 WH
ERE 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.SubmissionD
ate
            WHERE requestevaluation.empl_username=endex_requester AND r
equestevaluation.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_u
serkind<>'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_incid
ent,type_of_incident,success) VALUES (@current_userkind,table_of_incide
nt,@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_incide
nt,type_of_incident,success) VALUES (@current_userkind,table_of_inciden
t,@current_username,now(),type_of_incident,'NO');
END$
DELIMITER ;




/* Stored Procedures */

/* A */
DELIMITER $
CREATE PROCEDURE LoginAccount(IN endex_username VARCHAR(12),in endex_pa
ssword 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=@cu
rrent_username AND password=@current_password;
    /* IF userkind=NULL => DE BRE8HKE*/
     IF (@current_userkind IS NULL) THEN
        SELECT 'DE BRE8HKE TO SYGKEKRIMENO ACCOUNT.PLHKTROLOGEISTE "CAL
L LoginAccount(your Username,your Password);"';
        ELSE
        SELECT 'SYNDE8HKATE EPITYXWS ',@current_username,',H IDIOTHTA S
AS 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.e
valuator_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 WH
ERE requestevaluation.empl_username=req_empl_username AND requestevalua
tion.job_id=req_job_id;

IF  (CURDATE()<=req_SubmissionDate) THEN
    SELECT evaluator_username into req_evaluator_username FROM evaluato
r WHERE job.job_id=req_job_id;
    INSERT INTO evaluationresult (empl_username,job_id,evaluator_userna
me) VALUES (req_empl_username,req_job_id,req_evaluator_username);

    UPDATE requestevaluation
    SET empl_interest=TRUE WHERE requestevaluation.empl_username=req_em
pl_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_evaluato
r,phase1,phase2,phase3,Grade FROM evaluationresult WHERE evaluationresu
lt.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_user
name,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 Par
ticular_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 jo
b_id=Demanded_job_id;
   SELECT COUNT(job_id) INTO NumUnansweredRequests FROM evaluationresul
t 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 AYT
HN TH DOYLEIA EXOYN A3IOLOGH8EI PLHRWS.';
      END IF;
      SELECT empl_username AS 'Candidate',grade FROM evaluationresult W
HERE 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 gr
ade 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 ypallhl
o.',particular_grade AS 'Vathmos,An to aithma exei a3iologh8ei plhrws.'
;
       SELECT name,surname AS 'O a3iologhths ths doyleias' FROM user WH
ERE particular_evaluator_username=user.username and userkind='EVALUATOR
';
       FETCH ReqCursor INTO particular_job_id,particular_grade,particul
ar_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 το 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.