**ΕΡΓΑΣΙΑ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ**

Ονοματεπώνυμο: Τριαντάφυλλος Πράππας ΑΜ: 1067504

email:up1067504@upnet.gr

Ονοματεπώνυμο: Γιώργος Παπανδρέου ΑΜ: 1067506

**Α’ ΜΕΡΟΣ: ΚΩΔΙΚΑΣ MYSQL(WORKBENCH)**

**ΕΤΑΙΡΕΙΑ**

CREATE TABLE `company` (

`AFM` char(9) NOT NULL,

`DOY` varchar(15) NOT NULL,

`name` varchar(35) NOT NULL,

`phone` bigint NOT NULL,

`street` varchar(15) NOT NULL,

`number` tinyint NOT NULL,

`city` varchar(15) NOT NULL,

`country` varchar(15) NOT NULL,

PRIMARY KEY (`AFM`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον πίνακα σχετικά με τις εταιρείες ακολούθησα το διάγραμμα που δόθηκε στο αρχείο του project. Έθεσα ως primary key το AFM το οποίο στην συνέχεια μπορεί να υπάρχει και ως firm

**ΧΡΗΣΤΗΣ**

CREATE TABLE `user` (

`username` varchar(12) NOT NULL,

`password` varchar(10) NOT NULL,

`name` varchar(25) NOT NULL,

`surname` varchar(35) NOT NULL,

`reg\_date` date NOT NULL,

`email` varchar(30) NOT NULL,

PRIMARY KEY (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον πίνακα των χρηστών ακολούθησα τον αντίστοιχο του διαγράμματος με primary key το username

**ΔΙΕΥΘΥΝΤΗΣ**

CREATE TABLE `manager` (

`managerUsername` varchar(12) NOT NULL,

`exp\_years` tinyint DEFAULT '0',

`firm` char(9) NOT NULL,

KEY `mngrUsr\_idx` (`managerUsername`),

KEY `mngrComp\_idx` (`firm`),

CONSTRAINT `mngrComp` FOREIGN KEY (`firm`) REFERENCES `company` (`AFM`),

CONSTRAINT `mngrUsr` FOREIGN KEY (`managerUsername`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Σχετικά με τον πίνακα του διευθυντή είναι ίδιος με αυτόν του διαγράμματος δεν έχει primary key. Τον έχω συνδέσει με Foreign Keys με τον πίνακα των users όπου το managerUsername ταυτίζεται με το username. Και με τον πίνακα των εταιρειών μίας και κάθε εταιρεία έχει έναν manager η σύνδεση έγινε μέσω του firm το οποίο ταυτίζεται με το αντίστοιχο AFM στον πίνακα company.

**ΥΠΑΛΛΗΛΟΣ**

CREATE TABLE `employee` (

`emplUsername` varchar(12) NOT NULL,

`AM` int NOT NULL,

`AFM` varchar(45) NOT NULL,

`proypiresia` tinyint DEFAULT '0',

`bio` text NOT NULL,

`sistatikes` varchar(35) NOT NULL,

`certificates` varchar(35) DEFAULT 'Δεν έχει πιστοποιήσεις',

`awards` varchar(35) DEFAULT 'Δεν έχει βραβεία',

UNIQUE KEY `AM\_UNIQUE` (`AM`),

KEY `emplUsr\_idx` (`emplUsername`),

CONSTRAINT `emplUsr` FOREIGN KEY (`emplUsername`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον πίνακα employee δεν πρόσθεσα κάτι παραπάνω πέρα από αυτά που μας ζητάει. Απλώς μέσω foreign key ένωσα τους πίνακες employee και user , δηλαδή το emplUsername ταυτίζεται με το user username.

**PROJECT**

CREATE TABLE `project` (

`employee` varchar(12) NOT NULL,

`number` int NOT NULL,

`descr` text NOT NULL,

`url` varchar(60) NOT NULL,

KEY `PrjctEmpl\_idx` (`employee`) /\*!80000 INVISIBLE \*/,

CONSTRAINT `PrjctEmpl` FOREIGN KEY (`employee`) REFERENCES `employee` (`emplUsername`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον πίνακα Project δεν πρόσθεσα κάτι επιπλέον.

**ΞΕΝΕΣ ΓΛΩΣΣΕΣ**

CREATE TABLE `languages` (

`employee` varchar(12) NOT NULL,

`lang` set('EN','FR','SP','GR') NOT NULL,

KEY `langEmpl\_idx` (`employee`),

CONSTRAINT `langEmpl` FOREIGN KEY (`employee`) REFERENCES `employee` (`emplUsername`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον πίνακα languages δεν πρόσθεσα κάτι επιπλέον.

**ΠΤΥΧΙΟ**

CREATE TABLE `degree` (

`titlos` varchar(50) NOT NULL,

`idryma` varchar(40) NOT NULL,

`bathmida` enum('LYKEIO','UNIV','MASTER','PHD') NOT NULL,

`etos` year NOT NULL,

`grade` float(4,1) NOT NULL,

`employee` varchar(12) NOT NULL,

// UNIQUE KEY `titlosidryma` (`titlos`,`idryma`),//

KEY `dgrEmpl\_idx` (`employee`),

CONSTRAINT `dgrEmpl` FOREIGN KEY (`employee`) REFERENCES `employee` (`emplUsername`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον πίνακα degree δεν ακολούθησα το σχεσιακό ακριβώς. Αποφάσισα να δημιουργήσω έναν ακριβώς πίνακα που να τα περιέχει όλα. Επιπλέον ενώ πρόσθεσα στον παραπάνω κώδικα αυτό που ζητάει δηλαδή ο τίτλος και το ίδρυμα να είναι μοναδικά. Θεωρώ ότι δεν βγάζει πολύ νόημα μιας και αν τα κάνουμε όντως μοναδικά θα σημαίνει ότι μετά από ένα INSERT συνδυασμού αυτών των δύο για ένα συγκεκριμένο εργαζόμενο ένας άλλος εργαζόμενος δεν θα μπορεί να έχει σπουδάσει στο ίδιο ίδρυμα.

**ΑΞΙΟΛΟΓΗΤΗΣ**

CREATE TABLE `evaluator` (

`evaluatorUsername` varchar(12) NOT NULL,

// `id` int NOT NULL,//

`exp\_years` tinyint NOT NULL DEFAULT '0',

`firm` char(9) NOT NULL,

KEY `evalUser\_idx` (`evaluatorUsername`),

KEY `evalComp\_idx` (`firm`),

CONSTRAINT `evalComp` FOREIGN KEY (`firm`) REFERENCES `company` (`AFM`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `evalUser` FOREIGN KEY (`evaluatorUsername`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον αξιολογητή ακολούθησα αυτά που ζητούσε. Τον συνέδεσα με τον πίνακα της εταιρείας έτσι ώστε να φαίνεται σε ποια εταιρεία δουλεύει καθώς και έγινε σύνδεση με τον πίνακα των user για να μπορούμε να παίρνουμε τα στοιχεία του. (Σε αυτό το σημείο θα ήθελα να τονίσω ότι εδώ πρόσθεσα το id καθώς δημιουργούσα την βάση και αφού έκανα τα insert τιμών πρόσεξα ότι ζητούσε id. Γενικά δεν αλλάζουν πολλά απλώς στις stored procedure που ζητάει το id του evaluator χρησιμοποίησα το username).

**ΔΟΥΛΕΙΑ**

CREATE TABLE `job` (

`id` int NOT NULL,

`salary` float(6,1) NOT NULL,

`position` varchar(40) NOT NULL,

`edra` varchar(45) NOT NULL,

`evaluator` varchar(12) NOT NULL,

`announce\_date` datetime NOT NULL,

`submission\_date` datetime NOT NULL,

PRIMARY KEY (`id`),

KEY `jbEval\_idx` (`evaluator`),

CONSTRAINT `jbEval` FOREIGN KEY (`evaluator`) REFERENCES `evaluator` (`evaluatorUsername`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Όσον αφορά τον πίνακα job ακολούθησα το σχεδιάγραμμα που μας δινόταν. Και επιπλέον πρόσθεσα σαν foreign key τον evaluator.

**ΧΡΕΙΑΖΕΤΑΙ (ΓΙΑ ΘΕΣΗ ΕΡΓΑΣΙΑΣ)**

CREATE TABLE `needs` (

`job\_id` int NOT NULL,

`antikeim` varchar(35) NOT NULL,

KEY `needsjob\_idx` (`job\_id`),

KEY `needsantikeim\_idx` (`antikeim`),

CONSTRAINT `needsantikeim` FOREIGN KEY (`antikeim`) REFERENCES `antikeimeno` (`title`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `needsjob` FOREIGN KEY (`job\_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Για τον πίνακα needs ακολούθησα το σχεσιακό διάγραμμα που μας δινόταν. Και τον συνέδεσα με τον πίνακα job και τον πίνακα antikeimeno που θα εξηγηθεί παρακάτω μέσα από τα αντίστοιχα foreign key.

**ΑΝΤΙΚΕΙΜΕΝΟ**

CREATE TABLE `antikeimeno` (

`title` varchar(35) NOT NULL,

`descr` tinytext NOT NULL,

`belongs\_to` varchar(35) DEFAULT NULL,

PRIMARY KEY (`title`),

KEY `belong\_idx` (`belongs\_to`),

CONSTRAINT `belong` FOREIGN KEY (`belongs\_to`) REFERENCES `antikeimeno` (`title`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Για τον πίνακα αντικείμενο ακολούθησα το σχεσιακό διάγραμμα. Επιπλέον να σχολιάσω την αναδρομή που υπάρχει ουσιαστικά ένα αντικείμενο μπορεί να ανήκει σε ένα άλλο αντικείμενο. Για να κάνουμε insert ένα τέτοιο αντικείμενο θα πρέπει να έχουμε δημιουργήσει πρώτα το αντικείμενο στο οποίο ανήκει πχ.

INSERT INTO `antikeimeno` VALUES

(“antikeimeno1” ,”descr” , NULL);

Και στην συνέχεια μπορούμε να κάνουμε το:

INSERT INTO `antikeimeno` VALUES

(“antikeimeno2” ,”descr” , “antikeimeno1”);

Αν προσπαθήσουμε να κάνουμε πρώτα το δεύτερο INSERT θα μας βγάλει error.

**ΑΙΤΗΣΗ ΑΞΙΟΛΟΓΗΣΗΣ**

CREATE TABLE `request\_evaluation` (

`employee` varchar(12) NOT NULL,

`job\_id` int NOT NULL,

UNIQUE KEY `empljob` (`employee`,`job\_id`),

KEY `reqEmpl\_idx` (`employee`),

KEY `reqJob\_idx` (`job\_id`) /\*!80000 INVISIBLE \*/,

CONSTRAINT `reqEmpl` FOREIGN KEY (`employee`) REFERENCES `employee` (`emplUsername`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `reqJob` FOREIGN KEY (`job\_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Στον πίνακα request\_evaluation ένας employee μπορεί να κάνει INSERT μια αίτηση βάζοντας το username και τον κωδικό της δουλείας που τον ενδιαφέρει. Επιπλέον έκανα σύνδεση του πίνακα με τους πίνακες employee και job με τον αντίστοιχο τρόπο.

**ΑΠΟΤΕΛΕΣΜΑ ΑΞΙΟΛΟΓΗΣΗΣ**

CREATE TABLE `evaluation\_results` (

`evaluation\_id` int NOT NULL,

`employee` varchar(12) NOT NULL,

`job\_id` int NOT NULL,

`comments` varchar(225) DEFAULT 'NO COMMENTS',

`evaluator` enum('0','1','2','3','4','Missing Grade') DEFAULT 'Missing Grade',

`report` enum('0','1','2','3','4','Missing Grade') DEFAULT 'Missing Grade',

`total\_evaluation` enum('0','1','2','Missing Grade') DEFAULT 'Missing Grade',

`grade` varchar(35) DEFAULT 'Pending',

PRIMARY KEY (`evaluation\_id`),

UNIQUE KEY `employeejob` (`employee`,`job\_id`),

KEY `evalEmpl\_idx` (`employee`),

KEY `evalJob\_idx` (`job\_id`),

CONSTRAINT `evalEmpl` FOREIGN KEY (`employee`) REFERENCES `employee` (`emplUsername`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `evalJob` FOREIGN KEY (`job\_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Σχετικά με τον πίνακα evaluation\_results ακολούθησα το σχεσιακό διάγραμμα. Και επιπλέον τον συνέδεσα με τους πίνακες employee και job με τον αντίστοιχο τρόπο.

**LOGS**

CREATE TABLE `logs` (

`username` varchar(12) DEFAULT NULL,

`action` varchar(45) DEFAULT NULL,

`timestamp` timestamp NULL DEFAULT NULL,

KEY `emplogs\_idx` (`username`),

CONSTRAINT `emplogs` FOREIGN KEY (`username`) REFERENCES `employee` (`emplUsername`),

CONSTRAINT `joblogs` FOREIGN KEY (`username`) REFERENCES `job` (`evaluator`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

**Triggers**

**COMPANY**

CREATE DEFINER=`root`@`localhost` TRIGGER `DenyCompanyDataChange` BEFORE UPDATE ON `company` FOR EACH ROW BEGIN

SET @connectedUser = USER();

IF( connectedUser.Type != 0)

THEN

SET NEW.AFM = OLD.AFM;

SET NEW.DOY = OLD.DOY;

SET NEW.name = OLD.name;

SET NEW.phone = OLD.phone;

SET NEW.street= OLD.street;

SET NEW.number= OLD.number;

SET NEW.city= OLD.city;

SET NEW.country= OLD.country;

ELSE IF ( connectedUser.Type = 0)

THEN

SET NEW.AFM = OLD.AFM;

SET NEW.DOY = OLD.DOY;

SET NEW.name = OLD.name;

END IF;

END IF;

END

**USER**

CREATE DEFINER=`root`@`localhost` TRIGGER `DenyUserDataChange` BEFORE UPDATE ON `user` FOR EACH ROW BEGIN

SET @connectedUser = USER();

IF( connectedUser.Type != 0)

THEN

SET NEW.username = OLD.username;

SET NEW.reg\_date = OLD.reg\_date;

ELSE IF ( connectedUser.Type = 0)

THEN

SET NEW.username = OLD.username;

END IF;

END IF;

END

**MANAGER**

CREATE DEFINER=`root`@`localhost` TRIGGER `DenyManagerDataChange` BEFORE UPDATE ON `manager` FOR EACH ROW BEGIN

SET @connectedUser = USER();

IF( connectedUser.Type != 0)

THEN

SET NEW.managerUsername = OLD.managerUsername;

SET NEW.exp\_years = OLD.exp\_years;

SET NEW.firm = OLD.firm;

ELSE IF ( connectedUser.Type = 0)

THEN

SET NEW.managerUsername = OLD.managerUsername;

END IF;

END IF;

END

**EMPLOYEE**

CREATE DEFINER=`root`@`localhost` TRIGGER `DenyEmployeeDataChange` BEFORE UPDATE ON `employee` FOR EACH ROW BEGIN

SET @connectedUser = USER();

IF( connectedUser.Type != 0)

THEN

SET NEW.emplUsername = OLD.emplUsername;

SET NEW.AM = OLD.AM;

SET NEW.AFM = OLD.AFM;

SET NEW.proypiresia = OLD.proypiresia;

SET NEW.bio= OLD.bio;

SET NEW.sistatikes= OLD.sistatikes;

SET NEW.certificates= OLD.certificates;

SET NEW.awards= OLD.awards;

ELSE IF ( connectedUser.Type = 0)

THEN

SET NEW.emplUsername = OLD.emplUsername;

END IF;

END IF;

END

**EVALUATOR**

CREATE DEFINER=`root`@`localhost` TRIGGER `DenyEvaluatorDataCahnge` BEFORE UPDATE ON `evaluator` FOR EACH ROW BEGIN

/\* Gets the connected user \*/

SET @connectedUser = USER();

/\* If the users type is NOT 0 == Admin... \*/

IF( connectedUser.Type != 0)

THEN

/\* Do NOT apply the changes keep the od values \*/

SET NEW.evaluatorUsrname = OLD.evaluatorUsrname;

SET NEW.exp\_years = OLD.exp\_years;

SET NEW.firm = OLD.firm;

ELSE IF ( connectedUser.Type = 0)

THEN

SET NEW.evaluatorUsrname = OLD.evaluatorUsrname;

END IF;

END IF;

END

Τα παραπάνω triggers απορρίπτουν την αλλαγή δεδομένων σε κάποιους πίνακες. Παρόλα αυτά την αλλαγή δεδομένων την καθορίζουμε εμείς στο GUI ανάλογα με το ποια δικαιώματα δίνουμε στο κάθε είδους χρήστη που κάνει login στην εφαρμογή μας.

**EVALUATION RESULT**

CREATE DEFINER=`root`@`localhost` TRIGGER `FinalGrade` BEFORE INSERT ON `evaluation\_results` FOR EACH ROW BEGIN

IF (NEW.evaluator = 'Missing Grade' OR NEW.report = 'Missing Grade' OR NEW.total\_evaluation = 'Missing Grade') THEN

SET NEW.grade = 'Pending';

ELSE

SET NEW.grade = NEW.evaluator + NEW.report + NEW.total\_evaluation - '3';

END IF;

END

Με το συγκεκριμένο trigger μπορούμε να πάρουμε τους βαθμούς στον πίνακα evaluation\_results και να τους προσθέσουμε και να πάρουμε τον τελικό βαθμό στο column grade. Επιπλέον με αυτό το trigger κάθε φορά που λείπει ένας από τους 3 βαθμούς και αφού έχω ορίσει σαν DEFAULT value τον βαθμό που λείπουν το ‘Missing Grade’ στο column grade θα εμφανίζεται η τιμή ‘Pending’ η οποία αργότερα θα μας χρειαστεί και για τα stored procedure που μας ζητάει το project.

**STORED PROCEDURES**

**3.1 Stored Procedure**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `Procedure1`(IN empl\_name VARCHAR(25), IN empl\_surname VARCHAR(35))

BEGIN

SELECT empl\_name , empl\_surname , evaluatorUsername , request\_evaluation.job\_id, evaluation\_id AS 'JobApply', grade FROM request\_evaluation

INNER JOIN job ON request\_evaluation.job\_id = job.id

INNER JOIN evaluator ON job.evaluator = evaluator.evaluatorUsername

INNER JOIN employee ON request\_evaluation.employee = employee.emplUsername

INNER JOIN evaluation\_results ON employee.emplUsername = evaluation\_results.employee AND job.id = evaluation\_results.job\_id

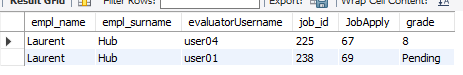
INNER JOIN user ON employee.emplUsername = user.username

WHERE (user.name = empl\_name) AND (user.surname = empl\_surname);

END$$

DELIMITER;

CALL Procedure1('Laurent' , 'Hub');



**3.2 Stored Procedure**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `Procedure2`(IN id INT, IN evalUsername VARCHAR(12))

BEGIN

SELECT evaluation\_id AS 'JobApply', job.id AS 'JobID', job.evaluator, grade AS 'Finalized' FROM job

INNER JOIN evaluation\_results ON job.id = evaluation\_results.job\_id

WHERE (job.id = id) AND (job.evaluator = evalUsername) AND (grade <> 'Pending');

END$$

DELIMITER ;

CALL Procedure2(225 , 'user04');



Αν υπάρχει βαθμός τελικός ύστερα από την αξιολόγηση θεωρείται Finalized. Αλλιώς παραμένει ως Pending στον πίνακα evaluation\_results.

**3.3 Stored Procedure**

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `Procedure3`(IN JobID INT)

BEGIN

DECLARE mistake VARCHAR(30);

IF EXISTS (SELECT name , surname, evaluation\_id, job\_id,

grade, IF (grade <> 'Pending' , 'Komple' , 'oxikomple') AS 'OLOKLIRWTHIKE'

FROM evaluation\_results

INNER JOIN employee ON evaluation\_results.employee = employee.emplUsername

INNER JOIN user ON employee.emplUsername = user.username

WHERE (evaluation\_results.job\_id = JobID)

ORDER BY grade)

THEN

SELECT name , surname, evaluation\_id, job\_id,

grade, IF (grade <> 'Pending' , 'Completed' , 'Not Completed') AS 'OLOKLIRWTHIKE'

FROM evaluation\_results

INNER JOIN employee ON evaluation\_results.employee = employee.emplUsername

INNER JOIN user ON employee.emplUsername = user.username

WHERE (evaluation\_results.job\_id = JobID)

ORDER BY grade;

ELSE

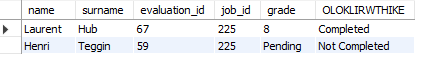
SELECT 'DOES NOT EXIST' AS 'JobEvaluation';

END IF;

END$$

DELIMITER ;

CALL Procedure3(225);

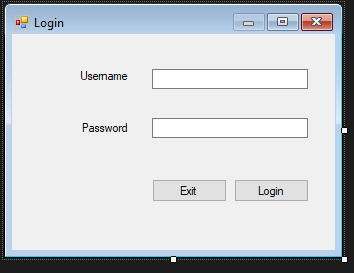


Με το συγκεκριμένο procedure μπορούμε να εισάγουμε τον κωδικό μιας θέσης εργασίας και στην συνέχεια να μας εμφανιστούν οι βαθμοί των υποψηφίων σε φθίνουσα σειρά και παράλληλα αν υπάρχει βαθμός να θεωρείται ολοκληρωμένη η βαθμολογία ενώ αν δεν υπάρχει να θεωρείται ότι δεν έχει ολοκληρωθεί ακόμα. Γενικά μετά το τέλος της εργασίας σχετικά με τα procedures ίσως έκανα κάποιες αλλαγές τις οποίες θα αναφέρω στο τέλος.

**Β’ ΜΕΡΟΣ GUI**

Για το GUI χρησιμοποίησα το Winforms(C#) στο Visual Studio. Επιπλέον να προσθέσω ότι για το Β μέρος υλοποιήθηκε η κατηγορία του Evaluator και η κατηγορία Employee.

**Login Page**



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace MySQLConnectionVS

{

public partial class Login : Form

{

public Login()

{

InitializeComponent();

}

private void btnLogin\_Click(object sender, EventArgs e)

{

database db = new database();

String username = txtUsername.Text;

String password = txtPassword.Text;

DataTable managTable = new DataTable();

DataTable emplTable = new DataTable();

DataTable evalTable = new DataTable();

MySqlDataAdapter managAdapter = new MySqlDataAdapter();

MySqlDataAdapter emplAdapter = new MySqlDataAdapter();

MySqlDataAdapter evalAdapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("select \* from `user` where `username` = @usn and `password` = @pass", db.getConnection());

MySqlCommand getManager = new MySqlCommand("select \* from `manager`INNER JOIN user ON manager.managerUsername = user.username where `managerUsername` = @usn and password = @pass ", db.getConnection());

MySqlCommand getEmployee = new MySqlCommand("select \* from `employee` INNER JOIN user ON employee.emplUsername = user.username where `emplUsername` = @usn and password = @pass", db.getConnection());

MySqlCommand getEvaluator = new MySqlCommand("select \* from `evaluator` INNER JOIN user ON evaluator.evaluatorUsername = user.username where `evaluatorUsername` = @usn and password = @pass", db.getConnection());

getManager.Parameters.Add("@usn", MySqlDbType.VarChar).Value = username;

getManager.Parameters.Add("@pass", MySqlDbType.VarChar).Value = password;

getEmployee.Parameters.Add("@usn", MySqlDbType.VarChar).Value = username;

getEmployee.Parameters.Add("@pass", MySqlDbType.VarChar).Value = password;

getEvaluator.Parameters.Add("@usn", MySqlDbType.VarChar).Value = username;

getEvaluator.Parameters.Add("@pass", MySqlDbType.VarChar).Value = password;

managAdapter.SelectCommand = getManager;

emplAdapter.SelectCommand = getEmployee;

evalAdapter.SelectCommand = getEvaluator;

managAdapter.Fill(managTable);

emplAdapter.Fill(emplTable);

evalAdapter.Fill(evalTable);

if (managTable.Rows.Count > 0)

{

frmManager objFrmManager = new frmManager();

this.Hide();

objFrmManager.Show();

}

else if (emplTable.Rows.Count >0)

{

frmEmployee objFrmEmployee = new frmEmployee(txtUsername.Text);

this.Hide();

objFrmEmployee.Show();

}

else if (evalTable.Rows.Count > 0)

{

frmEvaluator objFrmEvaluator = new frmEvaluator(txtUsername.Text);

this.Hide();

objFrmEvaluator.Show();

}

else

{

MessageBox.Show("Check Username And Password");

}

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

}

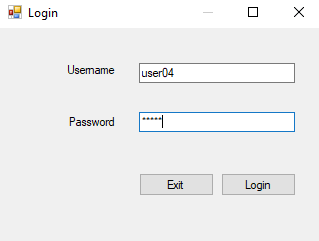
}

}

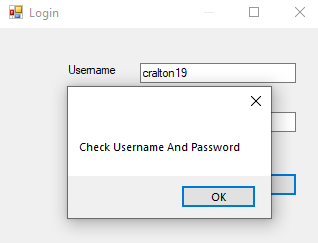
Ο παραπάνω είναι ο κώδικας για την σύνδεση ενός χρήστη και στην συνέχεια αφού γίνει έλεγχος ότι το username με το password αντιστοιχούν στην βάση εμφανίζεται μια επόμενη σελίδα ανάλογα με το τι τύπος είναι ο user, δηλαδή υπάλληλος, αξιολογητής ή μάνατζερ.

Ας δείξουμε πρώτα παραδείγματα για το καθένα.

Στα κενά συμπληρώνουμε το username και τον κωδικό αν είναι σωστός θα μας πετάξει στην ανάλογη σελίδα.



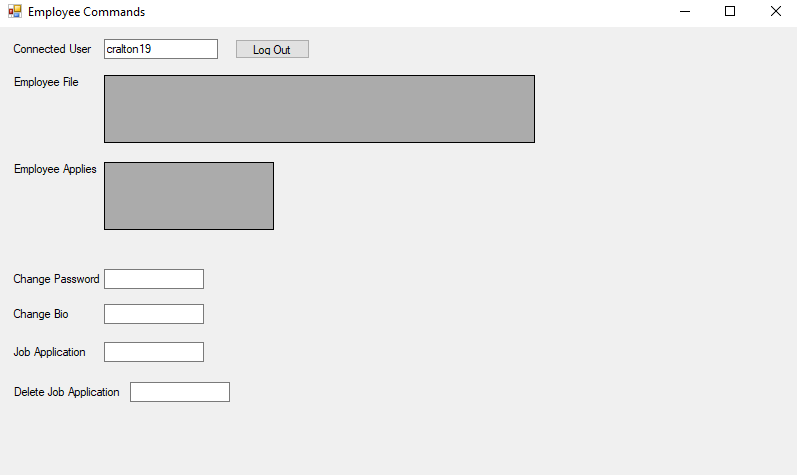
Αν είναι λάθος θα μας εμφανίσει το αντίστοιχο λάθος.

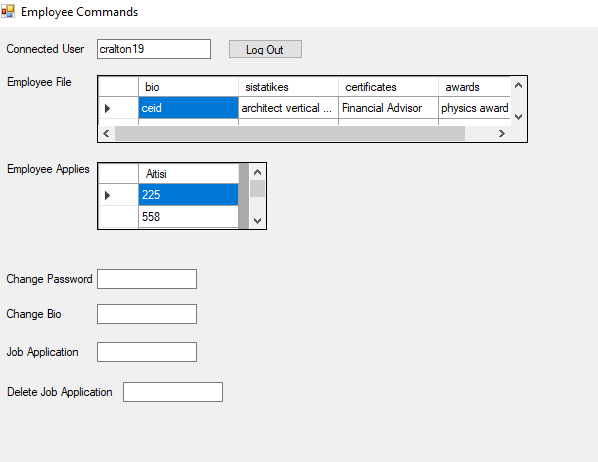


Επιπλέον πατώντας το κουμπί exit θα κλείσουμε το πρόγραμμα.

Η παρακάτω φωτογραφία είναι το περιβάλλον ενός υπαλλήλου πάνω υπάρχει το username του καθώς και ένα κουμπί Log Out αν θέλουμε να αλλάξουμε χρήστη χωρίς να χρειαστεί να ξανά κάνουμε compile σαν developers. Επιπλέον πατώντας το Label Employee File ο χρήστης μπορεί να δει τα στοιχεία του όπως τα awards ,certificates, bio κτλ. Πατώντας το Employee Applies εμφανίζονται όλες οι αιτήσεις που έχει κάνει για κάποιες θέσεις εργασίας που θέλει.

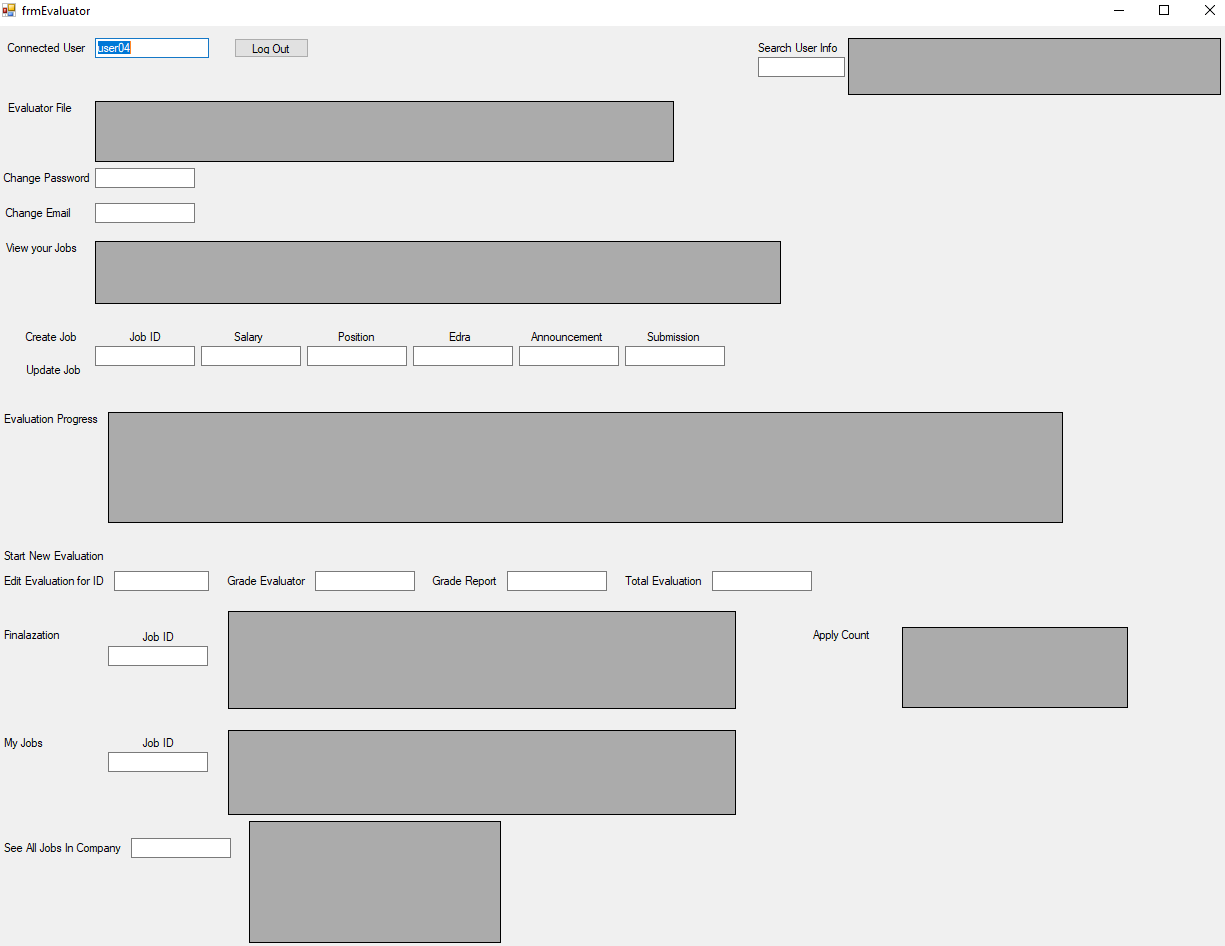
Και στην συνέχεια υπάρχουν και τα άλλα labels με τα οποία αφού συμπληρώσουμε το text box θα μπορεί ο υπάλληλος να αλλάξει τον κωδικό του , να αλλάξει το bio του , να κάνει αίτηση για μια δουλεία η οποία αίτηση αφού γίνει θα εμφανιστεί στον πίνακα του Employee Applies και αντίστοιχα θα διαγραφεί μια θέση από Employee Applies με την χρήση του Delete Job Application.

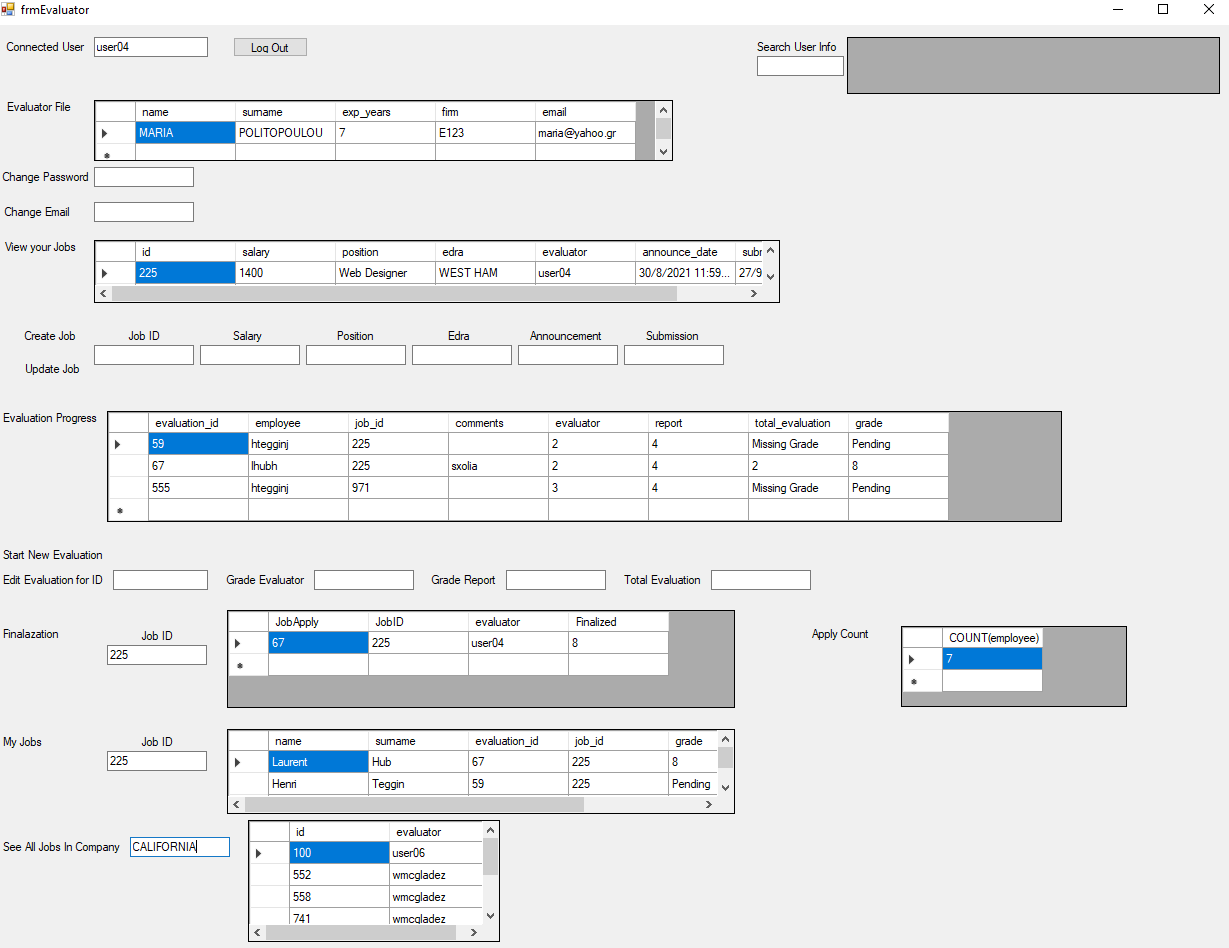




Τα παρακάτω screenshots αφορούν το περιβάλλον που μπορεί να χρησιμοποιήσει ένας Evaluator. Κάποια πράγματα όπως το Log out το ποιος χρήστης είναι συνδεδεμένος το Evaluator File που ουσιαστικά μας δίνει τα στοιχεία του χρήστη ανάλογο με το Employee File καθώς και η αλλαγή email και κωδικού παρόμοια με αυτά του employee. Επιπλέον για τον Evaluator του δίνουμε την δυνατότητα να δει τις δουλείες για τις οποίες είναι υπεύθυνος καθώς και να δημιουργήσει μια καινούργια δουλειά αλλά και να ενημερώσει μια παλιά ουσιαστικά να αλλάξει τον μισθό καθώς και την ημερομηνία που μπορούν να ξεκινήσουν οι αιτήσεις αλλά και το πότε τελειώνουν οι υποβολές ενώ παράλληλα μπορεί να βάζει βαθμούς στις ήδη υπάρχουσες. Επιπλέον μπορεί να δει για μια συγκεκριμένη δουλεία για την οποία είναι υπεύθυνος ποιες βαθμολογίες έχουν οριστικοποιηθεί μέσω του Finalization. Επίσης μπορεί μέσω του My Jobs να δει για μια συγκεκριμένη θέση που είναι υπεύθυνος για το ποιοι βαθμοί είναι ολοκληρωμένοι και ποιοι δεν είναι. Επίσης με το Apply Count ο evaluator μπορεί να δει πόσες αιτήσεις έχει για τις θέσεις που είναι υπεύθυνος. Ακόμα μέσω του See All Jobs μπορεί να δει τις αιτήσεις στην εταιρεία του αλλά και σε άλλες εταιρείες για να κρίνει την ζήτηση δουλειών.

Με το Evaluation Progress μπορεί να δει το πως πάνε οι αξιολογήσεις τους βαθμούς κτλ. Προσέθεσα επίσης και τρόπο για να μπορεί να βάζει βαθμούς εκεί που λείπουν και να αλλάζει ο τελικός βαθμός. Παρόλα αυτά ενώ το procedure στο workbench τρέχει κανονικά στο winforms μου πετάει error το οποίο δεν κατάφερα να επιλύσω.





Όλα τα παραπάνω έχουν επιτευχθεί με stored procedure που δημιουργήθηκαν στην mysql και στην συνέχεια καλούνται στο Visual Studio έτσι ώστε να εκτελεσθεί η εντολή που θέλουμε, με παρόμοιο τρόπο θα δημιουργούσα και stored procedures για την κατηγορία manager για να εκτελεί τις εντολές που μας ζητούνται. Επιπλέον σε αυτό το σημείο να προσθέσω ότι για τον πίνακα logs προσθέτοντας triggers στους πίνακες που μας ενδιαφέρει πχ. Για τον πίνακα employee έχουμε:

CREATE DEFINER=`root`@`localhost` TRIGGER `insertLog` AFTER INSERT ON `employee` FOR EACH ROW BEGIN

INSERT INTO logs

VALUES (NEW.emplUsername, 'Insert Bio', TIMESTAMP);

END

CREATE DEFINER=`root`@`localhost` TRIGGER `employee\_AFTER\_UPDATE` AFTER UPDATE ON `employee` FOR EACH ROW BEGIN

INSERT INTO logs

VALUES (NEW.emplUsername, 'Update Bio', TIMESTAMP);

END

CREATE DEFINER=`root`@`localhost` TRIGGER `employee\_AFTER\_DELETE` AFTER DELETE ON `employee` FOR EACH ROW BEGIN

INSERT INTO logs

VALUES (emplUsername, 'Delete Bio', TIMESTAMP);

END

Με τα παραπάνω και αφού κάνουμε αλλαγές πχ στο Bio στο πρόγραμμα που έχουμε φτιάξει

Και στην συνέχεια κάνουμε Select \*from logs; θα εμφανιστεί η αλλαγή που κάναμε και το όνομα του χρήστη.

Οι κώδικες για τα stored και για τις κλάσεις του GUI βρίσκονται σε txt αρχεία στο .zip του δεύτερου μέρους της παράδοσης.

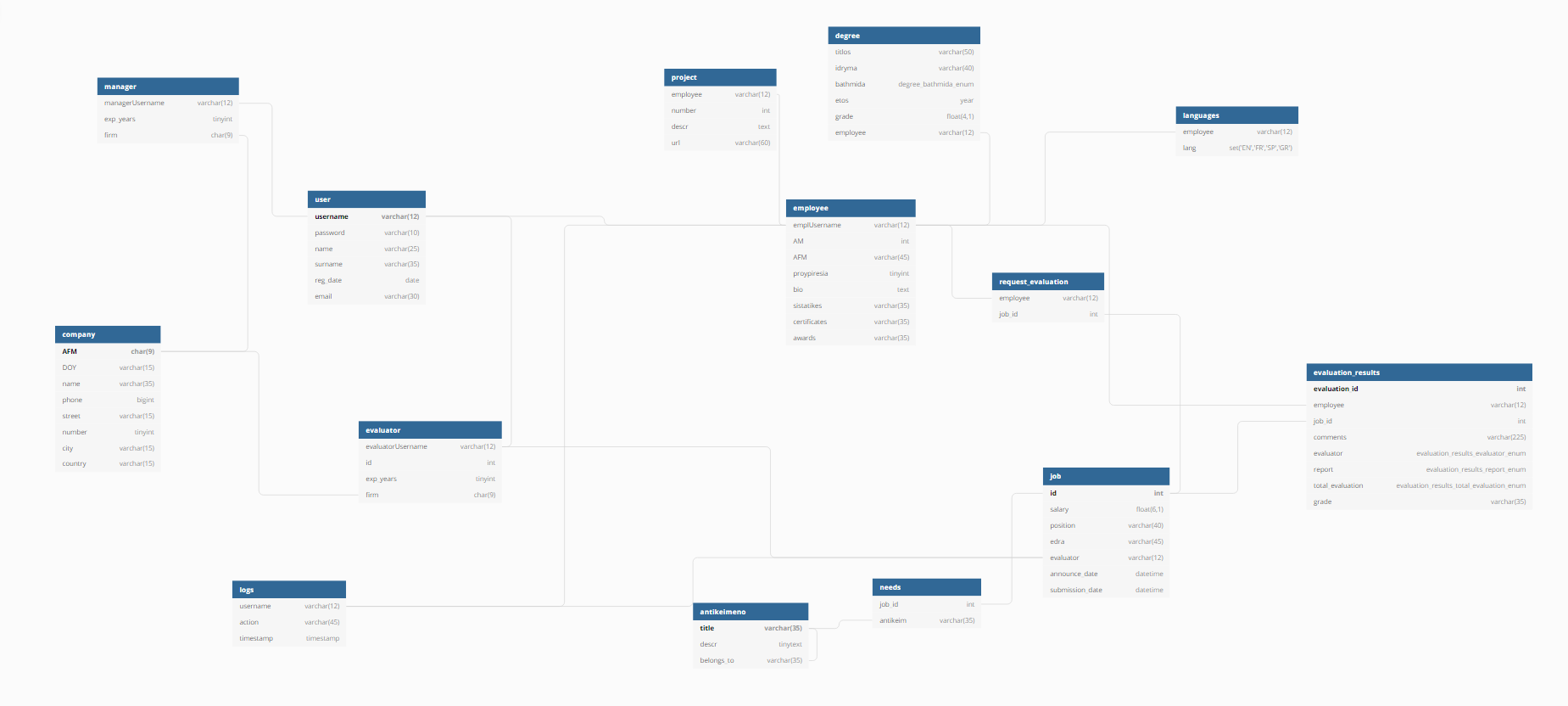
Αλλαγές που χρειάζονται έτσι ώστε να διευκολυνόταν η δημιουργία του GUI:

Γενικά θα άλλαζα κάποιους πίνακες θα προσέθετα παραπάνω columns για καλύτερη ανάλυσή τους.

Επίσης θα έκανα περισσότερες συνδέσεις πινάκων ιδιαίτερα μεταξύ των evaluation\_results και request\_evaluation.

Τέλος στοιχεία όπως είναι τα αντικείμενα πτυχία γλώσσες κτλ. Ίσως να είναι καλύτερο να βρίσκονταν σαν columns των πινάκων job, employee αντίστοιχα.

**ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ**

****

(Επειδή δεν φαίνεται πολύ καλά θα είναι και μέσα στο αρχείο .zip)