

## ΑΝΑΦΟΡΑ PROJECT ΣΤΟ ΕΡΓΑΣΤΗΡΙΟ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ 2020

### ΜΕΛΗ ΟΜΑΔΑΣ

**ΤΣΑΜΠΑΣ ΣΤΥΛΙΑΝΟΣ 1039884**

**ΣΙΑΜΟΓΛΟΥ ΧΑΡΑΛΑΜΠΟΣ 1041601**

### Περιγραφή

Στόχος της άσκησης ήταν η δημιουργία μίας βάσης δεδομένων και της αντίστοιχης διεπαφής για την διαδικασία αξιολογήσεων προσλήψεων σε έναν όμηλο εταιριών.

Το σχήμα της βάσης μας βασίστηκε στο σχήμα που δινόταν σαν παράδειγμα στην εκφώνηση στο οποίο προστέθηκαν οι παρακάτω πίνακες.

**admin :** Καταγράφει τους χρήστες που έχουν την ιδιότητα του Administrator. Περιέχει το πεδίο username το οποίο κληρωνομεί από τον πίνακα user.

**evaluationprocess :** Καταγράφει την διαδικασία αξιολόγησης. Έχει ως κύριο κλειδί τον συνδιασμό των empl\_username και job\_id που προέρχονται από τους πίνακες employee και job αντίστοιχα. Περιέχει επίσης τα πεδία phase\_1, phase\_2 και phase\_3 για τις βαθμολογίες στις τρεις φάσεις αξιολόγησης. το πεδίο comment για σχόλια σχετικά με τη συνέντευξη, το report για την αναφορά του manager καθώς και πεδίο mngr\_username με το όνομα του manager που κατέθεσε το report.

**evaluationresult :** Καταγράφει τα αποτελέσματα των αξιολογήσεων. Περιέχει τα πεδία id και empl\_username που αποτελούν τα πρωτεύοντα κλειδιά, πεδίο job\_id ως δευτερεύον κλειδί από τον πίνακα job, grade για τον τελικό βαθμό καθώς και comments για τα σχόλια του evaluator από την συνέντευξη.

**log :** Καταγράφει τις ενέργειες που συμβαίνουν σε συγκεκριμένους πίνακες. Ως πρωτεύον κλειδί έχει ένα id. Τα υπόλοιπα πεδία του είναι username για το όνομα του χρήστη που έκανε την ενέργεια, action για το είδος της ενέργειας (insert/update), table για τον πίνακα στον οποίο έγινε η ενέργεια, result για τον αν η ενέργεια ήταν επιτυχής ή όχι και datetime για την ημερομηνία και ώρα της ενέργειας.

### Triggers

Περιγράφονται με σχόλια στον επισυναπτόμενο κώδικα.

### Procedures

Περιγράφονται με σχόλια στον επισυναπτόμενο κώδικα.

## Διεπαφή Χρήστη

Το interface σε java δεν το ολοκληρώσαμε από άποψη λειτουργικότητας. Υλοποιήσαμε το login των χρηστών καθώς και την λήψη και εμφάνιση μερικών δεδομένων για τον administrator. Παρακάτω φαίνονται screenshots από το design της διεπαφής.

### Login

Username:  aa, aa aa ▼

Password:

Login

### Administrator

username	password	name	surname	reg_date	email
aa	aa	aa	aa	2021-01-04 18:25...	aa@aa.com
bb	bb	bb	bb	2021-01-04 18:25...	bb@bb.com
cc	cc	cc	cc	2021-01-04 18:25...	cc@cc.com
dd	dd	dd	dd	2021-01-04 18:25...	dd@dd.com
ee	ee	ee	ee	2021-01-04 18:25...	ee@ee.com
ff	ff	ff	ff	2021-01-04 18:25...	ff@ff.com
gg	gg	gg	gg	2021-01-04 18:25...	gg@gg.com
hh	hh	hh	hh	2021-01-04 18:25...	hh@hh.com
ii	ii	ii	ii	2021-01-04 18:25...	ii@ii.com
jj	jj	jj	jj	2021-02-13 16:25...	jj@jj.com
kk	kk	kk	kk	2021-02-13 16:51...	kk@kk.com
ll	ll	ll	ll	2021-02-13 16:54...	ll@ll.com
zz	zz	zz	zz	2021-01-07 08:02...	zz@zz.com

Add User

Username:  Password:

Name:  Surname:

Email:  Role: ▼ Company: ▼

Add

Edit

afm	doy	name	phone	street	num	city	country
123456789	A Patrwn	Vyzakias A.E.	6986959596	Aithera	15	Patra	GR
987654321	B Patrwn	Kwlakias A.E.	6988888888	Aithera	16	Patra	GR

Add Company

AFM:  DOY:

Name:  Phone:

Street:  Num:

City:  Country:

Add

Edit

id	username	action	table	result	datetime
1	zz	insert	employee	success	2021-02-13 16:29...

Add Activity

Title:  Parent: ▼

Description:

Add

Fetch

Manager

PersonalCompanyJobsEmployeesEvaluationsEvaluators

PersonalCompanyJobsEmployeesEvaluationsEvaluators

Password:

Email:

AFM:

DOY:

Update

Name:

Phone:

Street:

Num:

City:

Country:

Update

PersonalCompanyJobsEmployeesEvaluationsEvaluators

PersonalCompanyJobsEmployeesEvaluationsEvaluators

Position:

Salary:

Update

References

Certificates

Awards

Update

PersonalCompanyJobsEmployeesEvaluationsEvaluators

PersonalCompanyJobsEmployeesEvaluationsEvaluators

Name:

Surname:

Fetch

Fetch

Finalized

In Progress

# Evaluator

PersonalJobsEvaluations

PersonalJobsEvaluations

Username:

Password:

Name:

Surname:

Email:

Update

Edit Jobs

Edit

ID:

Date:

Start Date:

Deadline:

Salary:

Position:

Location:

Activity:

Add/Update

Activities

Title:

Parent:

Description

Add

PersonalJobsEvaluations

Edit Evaluations

Job:

# of Requests:

Update

Employee

PersonalRequests

Username: Password: Name: Surname: Email: Company: Bio

References

Certificates

Awards

PersonalRequests

Open

In Progress

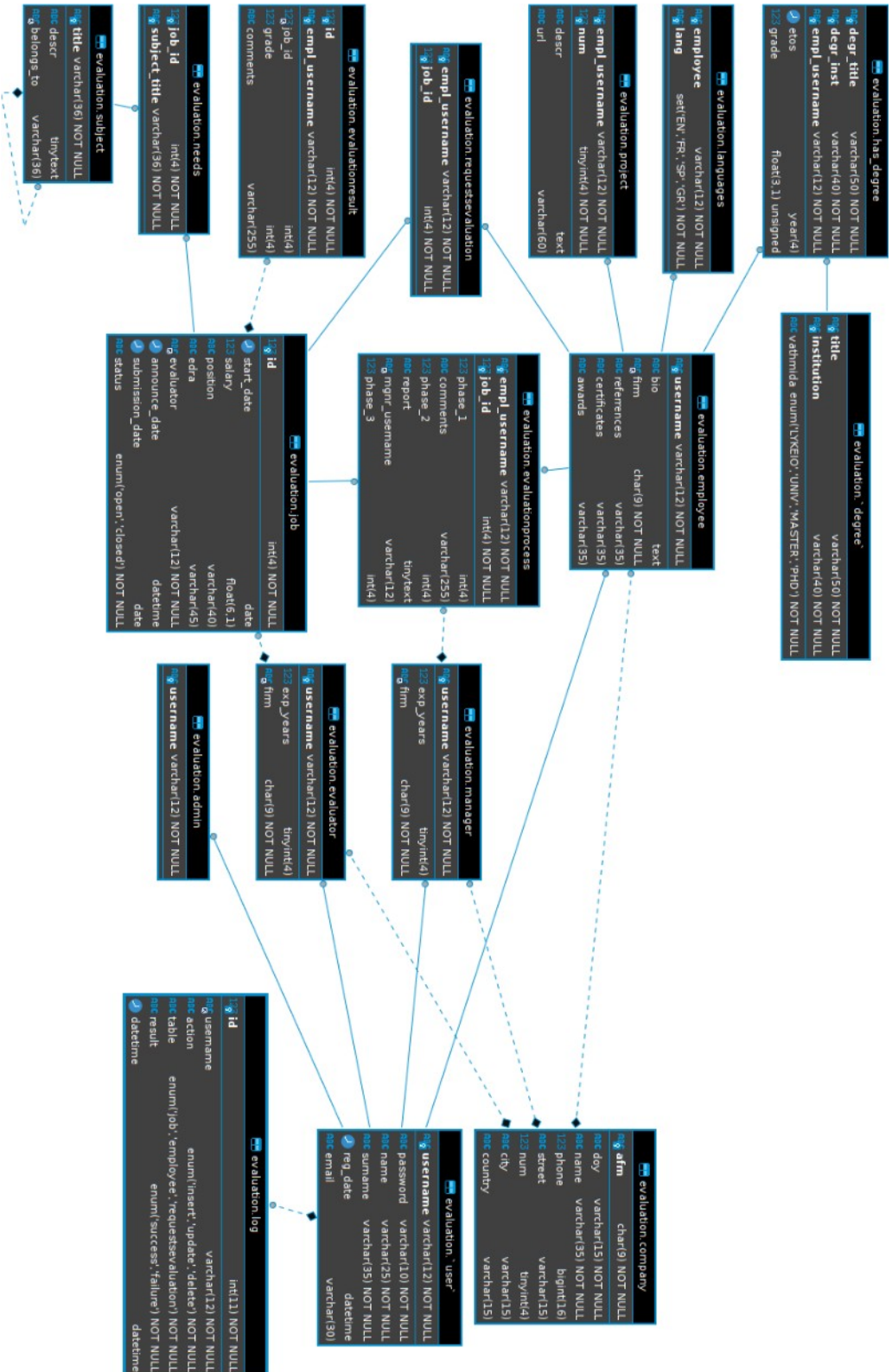
Finalized

Update

Submit

UpdateDelete

## Σχήμα Βάσης



```

1 DROP DATABASE `evaluation`;
2 CREATE DATABASE `evaluation`;
3 USE `evaluation`;
4
5 --
6 -- Table structure for table `user`
7 --
8 DROP TABLE IF EXISTS `user`;
9 CREATE TABLE `user` (
10   `username` varchar(12) NOT NULL,
11   `password` varchar(10) NOT NULL,
12   `name` varchar(25) NOT NULL,
13   `surname` varchar(35) NOT NULL,
14   `reg_date` datetime DEFAULT current_timestamp(),
15   `email` varchar(30) NULL,
16   PRIMARY KEY (`username`)
17 ) ENGINE=InnoDB;
18
19 --
20 -- Table structure for table `company`
21 --
22 DROP TABLE IF EXISTS `company`;
23 CREATE TABLE `company` (
24   `afm` char(9) NOT NULL,
25   `doy` varchar(15) NOT NULL,
26   `name` varchar(35) NOT NULL,
27   `phone` bigint(16) NULL,
28   `street` varchar(15) NULL,
29   `num` tinyint(4) NULL,
30   `city` varchar(15) NULL,
31   `country` varchar(15) NULL,
32   PRIMARY KEY (`afm`)
33 ) ENGINE=InnoDB;
34
35 --
36 -- Table structure for table `degree`
37 --
38 DROP TABLE IF EXISTS `degree`;
39 CREATE TABLE `degree` (
40   `title` varchar(50) NOT NULL,
41   `institution` varchar(40) NOT NULL,
42   `vathmida` enum('LYKEIO','UNIV','MASTER','PHD') NOT NULL,
43   PRIMARY KEY (`title`,`institution`)
44 ) ENGINE=InnoDB;
45
46 --
47 -- Table structure for table `admin`
48 --
49 DROP TABLE IF EXISTS `admin`;
50 CREATE TABLE `admin` (
51   `username` varchar(12) NOT NULL,
52   PRIMARY KEY (`username`),
53   CONSTRAINT `fk_admin_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
54 ) ENGINE=InnoDB;
55
56 --
57 -- Table structure for table `manager`
58 --
59 DROP TABLE IF EXISTS `manager`;
60 CREATE TABLE `manager` (
61   `username` varchar(12) NOT NULL,
62   `exp_years` tinyint(4) NULL,
63   `firm` char(9) NOT NULL,
64   PRIMARY KEY (`username`),
65   KEY `fk_manager_company` (`firm`),
66   CONSTRAINT `fk_manager_company` FOREIGN KEY (`firm`) REFERENCES `company` (`afm`) ON DELETE CASCADE ON UPDATE CASCADE,
67   CONSTRAINT `fk_manager_users` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
68 ) ENGINE=InnoDB;
69
70 --
71 -- Table structure for table `evaluator`
72 --
73 DROP TABLE IF EXISTS `evaluator`;
74 CREATE TABLE `evaluator` (
75   `username` varchar(12) NOT NULL,
76   `exp_years` tinyint(4) NULL,
77   `firm` char(9) NOT NULL,
78   PRIMARY KEY (`username`),
79   KEY `fk_evaluator_company` (`firm`),
80   CONSTRAINT `fk_evaluator_company` FOREIGN KEY (`firm`) REFERENCES `company` (`afm`) ON DELETE CASCADE ON UPDATE CASCADE,
81   CONSTRAINT `fk_evaluator_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
82 ) ENGINE=InnoDB;
83
84 --
85 -- Table structure for table `employee`
86 --
87 DROP TABLE IF EXISTS `employee`;
88 CREATE TABLE `employee` (
89   `username` varchar(12) NOT NULL,
90   `bio` text NULL,

```



```

91     `firm` char(9) NOT NULL,
92     `references` varchar(35) NULL,
93     `certificates` varchar(35) NULL,
94     `awards` varchar(35) NULL,
95     PRIMARY KEY (`username`),
96     KEY `fk_employee_company` (`firm`),
97     CONSTRAINT `fk_employee_company` FOREIGN KEY (`firm`) REFERENCES `company` (`afm`) ON DELETE CASCADE ON UPDATE CASCADE,
98     CONSTRAINT `fk_employee_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
99 ) ENGINE=InnoDB;
100
101 --
102 -- Table structure for table `job`
103 --
104 DROP TABLE IF EXISTS `job`;
105 CREATE TABLE `job` (
106     `id` int(4) NOT NULL AUTO_INCREMENT,
107     `start_date` date NULL,
108     `salary` float(6,1) NULL,
109     `position` varchar(40) NULL,
110     `edra` varchar(45) NULL,
111     `evaluator` varchar(12) NOT NULL,
112     `announce_date` datetime NULL,
113     `submission_date` date NULL,
114     `status` enum('open','closed') NOT NULL DEFAULT 'open',
115     PRIMARY KEY (`id`),
116     KEY `fk_job_evaluator` (`evaluator`),
117     CONSTRAINT `fk_job_evaluator` FOREIGN KEY (`evaluator`) REFERENCES `evaluator` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
118 ) ENGINE=InnoDB;
119
120 --
121 -- Table structure for table `evaluationprocess`
122 --
123 DROP TABLE IF EXISTS `evaluationprocess`;
124 CREATE TABLE `evaluationprocess` (
125     `empl_username` varchar(12) NOT NULL,
126     `job_id` int(4) NOT NULL,
127     `phase_1` int(4) DEFAULT NULL,
128     `comments` varchar(255) NULL,
129     `phase_2` int(4) DEFAULT NULL,
130     `report` tinytext NULL,
131     `mgr_username` varchar(12) NULL,
132     `phase_3` int(4) DEFAULT NULL,
133     PRIMARY KEY (`empl_username`,`job_id`),
134     KEY `fk_evaluationprocess_job` (`job_id`),
135     KEY `fk_evaluationprocess_manager` (`mgr_username`),
136     CONSTRAINT `fk_evaluationprocess_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON
137 UPDATE CASCADE,
138     CONSTRAINT `fk_evaluationprocess_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
139     CONSTRAINT `fk_evaluationprocess_manager` FOREIGN KEY (`mgr_username`) REFERENCES `manager` (`username`) ON DELETE CASCADE ON UPDATE
140 CASCADE
141 ) ENGINE=InnoDB;
142
143 --
144 -- Table structure for table `evaluationresult`
145 --
146 DROP TABLE IF EXISTS `evaluationresult`;
147 CREATE TABLE `evaluationresult` (
148     `id` int(4) NOT NULL AUTO_INCREMENT,
149     `empl_username` varchar(12) NOT NULL,
150     `job_id` int(4) NULL,
151     `grade` int(4) NULL,
152     `comments` varchar(255) NULL,
153     PRIMARY KEY (`id`,`empl_username`),
154     KEY `fk_evaluationresult_employee` (`empl_username`),
155     KEY `fk_evaluationresult_job` (`job_id`),
156     CONSTRAINT `fk_evaluationresult_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE
157 CASCADE,
158     CONSTRAINT `fk_evaluationresult_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE SET NULL ON UPDATE CASCADE
159 ) ENGINE=InnoDB;
160
161 --
162 -- Table structure for table `has_degree`
163 --
164 DROP TABLE IF EXISTS `has_degree`;
165 CREATE TABLE `has_degree` (
166     `degr_title` varchar(50) NOT NULL,
167     `degr_inst` varchar(40) NOT NULL,
168     `empl_username` varchar(12) NOT NULL,
169     `etos` year(4) NULL,
170     `grade` float(3,1) unsigned NULL,
171     PRIMARY KEY (`degr_title`,`degr_inst`,`empl_username`),
172     KEY `fk_has_degree_employee` (`empl_username`),
173     CONSTRAINT `fk_has_degree_degree` FOREIGN KEY (`degr_title`,`degr_inst`) REFERENCES `degree` (`title`,`institution`) ON DELETE
174 CASCADE ON UPDATE CASCADE,
175     CONSTRAINT `fk_has_degree_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE
176 CASCADE
177 ) ENGINE=InnoDB;
178
179 --
180 -- Table structure for table `languages`

```

```

176 --
177 DROP TABLE IF EXISTS `languages`;
178 CREATE TABLE `languages` (
179   `employee` varchar(12) NOT NULL,
180   `lang` set('EN','FR','SP','GR') NOT NULL,
181   PRIMARY KEY (`employee`,`lang`),
182   CONSTRAINT `fk_languages_employee` FOREIGN KEY (`employee`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
183 ) ENGINE=InnoDB;
184
185 --
186 -- Table structure for table `log`
187 --
188 DROP TABLE IF EXISTS `log`;
189 CREATE TABLE `log` (
190   `id` int(11) NOT NULL AUTO_INCREMENT,
191   `username` varchar(12) NOT NULL,
192   `action` enum('insert','update','delete') NOT NULL,
193   `table` enum('job','employee','requestsevaluation') NOT NULL,
194   `result` enum('success','failure') NOT NULL,
195   `datetime` datetime DEFAULT current_timestamp(),
196   PRIMARY KEY (`id`),
197   KEY `fk_log_user` (`username`),
198   CONSTRAINT `fk_log_user` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE RESTRICT ON UPDATE CASCADE
199 ) ENGINE=InnoDB;
200
201 --
202 -- Table structure for table `subject`
203 --
204 DROP TABLE IF EXISTS `subject`;
205 CREATE TABLE `subject` (
206   `title` varchar(36) NOT NULL,
207   `descr` tinytext NULL,
208   `belongs_to` varchar(36) NULL,
209   PRIMARY KEY (`title`),
210   UNIQUE KEY `title_UNIQUE` (`title`),
211   KEY `fk_subject_subject` (`belongs_to`),
212   CONSTRAINT `fk_subject_subject` FOREIGN KEY (`belongs_to`) REFERENCES `subject` (`title`) ON DELETE SET NULL ON UPDATE CASCADE
213 ) ENGINE=InnoDB;
214
215 --
216 -- Table structure for table `needs`
217 --
218 DROP TABLE IF EXISTS `needs`;
219 CREATE TABLE `needs` (
220   `job_id` int(4) NOT NULL,
221   `subject_title` varchar(36) NOT NULL,
222   PRIMARY KEY (`job_id`,`subject_title`),
223   KEY `fk_needs_subject` (`subject_title`),
224   CONSTRAINT `fk_needs_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
225   CONSTRAINT `fk_needs_subject` FOREIGN KEY (`subject_title`) REFERENCES `subject` (`title`) ON DELETE CASCADE ON UPDATE CASCADE
226 ) ENGINE=InnoDB;
227
228 --
229 -- Table structure for table `requestsevaluation`
230 --
231 DROP TABLE IF EXISTS `requestsevaluation`;
232 CREATE TABLE `requestsevaluation` (
233   `empl_username` varchar(12) NOT NULL,
234   `job_id` int(4) NOT NULL,
235   PRIMARY KEY (`empl_username`,`job_id`),
236   KEY `fk_requestsevaluation_job` (`job_id`),
237   CONSTRAINT `fk_requestsevaluation_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON
UPDATE CASCADE,
238   CONSTRAINT `fk_requestsevaluation_job` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
239 ) ENGINE=InnoDB;
240
241 --
242 -- Table structure for table `project`
243 --
244 DROP TABLE IF EXISTS `project`;
245 CREATE TABLE `project` (
246   `empl_username` varchar(12) NOT NULL,
247   `num` tinyint(4) NOT NULL,
248   `descr` text NULL,
249   `url` varchar(60) NULL,
250   PRIMARY KEY (`empl_username`,`num`),
251   CONSTRAINT `fk_project_employee` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
252 ) ENGINE=InnoDB;
253
254 --
255 -- Triggers for table `company`
256 --
257 -- Αποτρέπει την αλλαγή συγκεκριμένων στοιχείων της εταιρίας
258 DELIMITER ;;
259 CREATE TRIGGER `company_BEFORE_UPDATE` BEFORE UPDATE ON `company` FOR EACH ROW
260 BEGIN
261   SET NEW.afm = OLD.afm;
262   SET NEW.doy = OLD.doy;
263   SET NEW.name = OLD.name;
264 END;;

```

```

265 DELIMITER ;
266
267 --
268 -- Triggers for table `employee`
269 --
270 -- Συνδιασμός απο triggers που καταγράφουν στον `log` τις αλλαγές που συμβαίνουν
271 -- στον `employee`. Καλούν συγκεκριμένα procedures. Η λειτουργία τους αναλύεται εκεί
272 DELIMITER ;;
273 CREATE TRIGGER `employee_BEFORE_INSERT` BEFORE INSERT ON `employee` FOR EACH ROW
274 BEGIN
275     CALL proc_log_before("insert", "employee", "failure");
276 END;;
277 DELIMITER ;
278 DELIMITER ;;
279 CREATE TRIGGER `employee_AFTER_INSERT` AFTER INSERT ON `employee` FOR EACH ROW
280 BEGIN
281     CALL proc_log_after();
282 END;;
283 DELIMITER ;
284 DELIMITER ;;
285 CREATE TRIGGER `employee_BEFORE_UPDATE` BEFORE UPDATE ON `employee` FOR EACH ROW
286 BEGIN
287     CALL proc_log_before("update", "employee", "failure");
288 END;;
289 DELIMITER ;
290 DELIMITER ;;
291 CREATE TRIGGER `employee_AFTER_UPDATE` AFTER UPDATE ON `employee` FOR EACH ROW
292 BEGIN
293     CALL proc_log_after();
294 END;;
295 DELIMITER ;
296
297 --
298 -- Triggers for table `job`
299 --
300 -- Συνδιασμός απο triggers που καταγράφουν στον `log` τις αλλαγές που συμβαίνουν
301 -- στον `job`. Καλούν συγκεκριμένα procedures. Η λειτουργία τους αναλύεται εκεί
302 DELIMITER ;;
303 CREATE TRIGGER `job_BEFORE_INSERT` BEFORE INSERT ON `job` FOR EACH ROW
304 BEGIN
305     CALL proc_log_before("insert", "job", "failure");
306 END;;
307 DELIMITER ;
308 DELIMITER ;;
309 CREATE TRIGGER `job_AFTER_INSERT` AFTER INSERT ON `job` FOR EACH ROW
310 BEGIN
311     CALL proc_log_after();
312 END;;
313 DELIMITER ;
314 DELIMITER ;;
315 CREATE TRIGGER `job_BEFORE_UPDATE` BEFORE UPDATE ON `job` FOR EACH ROW
316 BEGIN
317     CALL proc_log_before("update", "job", "failure");
318 END;;
319 DELIMITER ;
320 DELIMITER ;;
321 CREATE TRIGGER `job_AFTER_UPDATE` AFTER UPDATE ON `job` FOR EACH ROW
322 BEGIN
323     CALL proc_log_after();
324 END;;
325 DELIMITER ;
326
327 --
328 -- Triggers for table `requestsevaluation`
329 --
330 -- Συνδιασμός απο triggers που καταγράφουν στον `log` τις αλλαγές που συμβαίνουν
331 -- στον `requestsevaluation`. Καλούν συγκεκριμένα procedures. Η λειτουργία τους αναλύεται εκεί
332 DELIMITER ;;
333 CREATE TRIGGER `requestsevaluation_BEFORE_INSERT` BEFORE INSERT ON `requestsevaluation` FOR EACH ROW
334 BEGIN
335     CALL proc_log_before("insert", "requestsevaluation", "failure");
336 END;;
337 DELIMITER ;
338 DELIMITER ;;
339 CREATE TRIGGER `requestsevaluation_AFTER_INSERT` AFTER INSERT ON `requestsevaluation` FOR EACH ROW
340 BEGIN
341     CALL proc_log_after();
342 END;;
343 DELIMITER ;
344 DELIMITER ;;
345 CREATE TRIGGER `requestsevaluation_BEFORE_UPDATE` BEFORE UPDATE ON `requestsevaluation` FOR EACH ROW
346 BEGIN
347     CALL proc_log_before("update", "requestsevaluation", "failure");
348 END;;
349 DELIMITER ;
350 DELIMITER ;;
351 CREATE TRIGGER `requestsevaluation_AFTER_UPDATE` AFTER UPDATE ON `requestsevaluation` FOR EACH ROW
352 BEGIN
353     CALL proc_log_after();
354 END;;

```

```

355 DELIMITER ;
356
357 --
358 -- Triggers for table `evaluationprocess`
359 --
360 -- Triggers που ελέγχουν αν υπάρχει βαθμός και στις τρεις φάσεις αξιολόγησης μετά από είσοδο ή
361 -- αλλαγή δεδομένων. Αν υπάρχει και στις τρεις φάσεις, καταγράφουν το αποτέλεσμα της αξιολόγησης
362 -- στον `evaluationresult`. Το υλοποιήσαμε πριν προχωρήσουμε στα απαιτούμενα procedures και το αφήσαμε
363 -- σαν δικλείδα ασφαλείας. Επίσης, αχρηστεύει το αντίστοιχο procedure.
364 DELIMITER ;;
365 CREATE TRIGGER `evaluationprocess_AFTER_INSERT` AFTER INSERT ON `evaluationprocess` FOR EACH ROW
366 BEGIN
367     DECLARE _grade_1 INT(4);
368     DECLARE _grade_2 INT(4);
369     DECLARE _grade_3 INT(4);
370     DECLARE _comments VARCHAR(255);
371     DECLARE _username VARCHAR(12);
372     DECLARE _job_id INT(4);
373     SELECT
374         empl_username, job_id, phase_1, comments, phase_2, phase_3
375     FROM
376         evaluation.evaluationprocess WHERE empl_username = NEW.empl_username AND job_id = NEW.job_id
377     INTO
378         _username, _job_id, _grade_1, _comments, _grade_2, _grade_3;
379
380     IF ((_grade_1 IS NOT NULL) AND (_grade_2 IS NOT NULL) AND (_grade_3 IS NOT NULL)) THEN
381         INSERT INTO evaluation.evaluationresult(empl_username, job_id, grade, comments)
382         VALUES(_username, _job_id, _grade_1+_grade_2+_grade_3, _comments);
383     END IF;
384 END;;
385 DELIMITER ;
386 DELIMITER ;;
387 CREATE TRIGGER `evaluationprocess_AFTER_UPDATE` AFTER UPDATE ON `evaluationprocess` FOR EACH ROW
388 BEGIN
389     DECLARE _grade_1 INT(4);
390     DECLARE _grade_2 INT(4);
391     DECLARE _grade_3 INT(4);
392     DECLARE _comments VARCHAR(255);
393     DECLARE _username VARCHAR(12);
394     DECLARE _job_id INT(4);
395     SELECT
396         empl_username, job_id, phase_1, comments, phase_2, phase_3
397     FROM
398         evaluation.evaluationprocess WHERE empl_username = NEW.empl_username AND job_id = NEW.job_id
399     INTO
400         _username, _job_id, _grade_1, _comments, _grade_2, _grade_3;
401
402     IF ((_grade_1 IS NOT NULL) AND (_grade_2 IS NOT NULL) AND (_grade_3 IS NOT NULL)) THEN
403         INSERT INTO evaluation.evaluationresult(empl_username, job_id, grade, comments)
404         VALUES(_username, _job_id, _grade_1+_grade_2+_grade_3, _comments);
405     END IF;
406 END;;
407 DELIMITER ;
408
409 --
410 -- Triggers for table `evaluationresult`
411 --
412 -- Trigger που ελέγχει αν μία αξιολόγηση έχει ήδη ολοκληρωθεί και δεν επιτρέπει την εισαγωγή
413 -- νέας εγγραφής για τον ίδιο `employee` και `job_id`.
414 DELIMITER ;;
415 CREATE TRIGGER `evaluationresult_BEFORE_INSERT` BEFORE INSERT ON `evaluationresult` FOR EACH ROW
416 BEGIN
417     DECLARE _exists INT;
418
419     SELECT
420         EXISTS(
421             SELECT * FROM `evaluation`.`evaluationresult`
422             WHERE empl_username = NEW.empl_username AND job_id = NEW.job_id)
423     INTO _exists;
424
425     IF (_exists) THEN
426         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Evaluation already finalized';
427     END IF;
428 END;;
429 DELIMITER ;
430 -- Δεν επιτρέπει την αλλαγή των αποτελεσμάτων των αξιολογήσεων απενεργοποιώντας
431 -- συνολικά τα updates στον πίνακα.
432 DELIMITER ;;
433 CREATE TRIGGER `evaluationresult_BEFORE_UPDATE` BEFORE UPDATE ON `evaluationresult` FOR EACH ROW
434 BEGIN
435     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Updates are disabled. Results are final';
436 END;;
437 DELIMITER ;
438
439 --
440 -- Triggers for table `user`
441 --
442 -- Αποτρέπει την εισαγωγή δεδομένων στον πίνακα αν ο χρήστης δεν έχει ρόλο administrator
443 -- Επίσης αποτρέπει μια διαφορετικό μήνυμα την εισαγωγή αν δεν έχει δηλωθεί ρόλος
444 DELIMITER ;;

```

```

445 CREATE TRIGGER `user_BEFORE_INSERT` BEFORE INSERT ON `user` FOR EACH ROW
446 BEGIN
447     IF (@evaluation_current_role IS NULL) THEN
448         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'evaluation_current_role is unset';
449     END IF;
450
451     IF (@evaluation_current_role != "administrator") THEN
452         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'evaluation_current_role is not administrator';
453     END IF;
454 END;;
455 DELIMITER ;
456 -- Υλοποιεί τα δικαιώματα των διαφορετικών ρόλων στην αλλαγή στοιχείων στον πίνακα `user`
457 -- Administrator: αλλάζει τα πάντα εκτός από το reg_date
458 -- Evaluator: δεν αλλάζει το username και reg_date
459 -- Manager: Δεν αλλάζει τα username, name, surname, reg_date
460 -- Employee: Δεν αλλάζει τα username, name, surname, reg_date, email
461 DELIMITER ;;
462 CREATE TRIGGER `user_BEFORE_UPDATE` BEFORE UPDATE ON `user` FOR EACH ROW
463 BEGIN
464     IF (@evaluation_current_role IS NULL) THEN
465         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'evaluation_current_role is unset';
466     END IF;
467     SET NEW.reg_date = OLD.reg_date;
468     IF (@evaluation_current_role != "administrator") THEN
469         SET NEW.username = OLD.username;
470         IF (@evaluation_current_role = "manager") THEN
471             SET NEW.name = OLD.name;
472             SET NEW.surname = OLD.surname;
473         END IF;
474         IF (@evaluation_current_role = "employee") THEN
475             SET NEW.name = OLD.name;
476             SET NEW.surname = OLD.surname;
477             SET NEW.email = OLD.email;
478         END IF;
479     END IF;
480 END;;
481 DELIMITER ;
482
483 --
484 -- Dumping routines for database 'evaluation'
485 --
486 -- Procedure που ζητήθηκε σαν ερώτημα, οριστικοποιεί τις αξιολογήσεις που έχουν βαθμούς και στις
487 -- τρεις φάσεις αλλά δεν υπάρχουν αντίστοιχες εγγραφές στον πίνακα `evaluationresult`.
488 DROP PROCEDURE IF EXISTS `proc_finalize_employee_evaluation`;
489 DELIMITER ;;
490 CREATE PROCEDURE `proc_finalize_employee_evaluation`(IN in_job_id INT, IN in_evaluator VARCHAR(12))
491 BEGIN
492     DECLARE _username VARCHAR(12);
493     DECLARE _job_id INT(4);
494     DECLARE _grade INT(4);
495     DECLARE _comments VARCHAR(255);
496     DECLARE _done BOOL DEFAULT FALSE;
497     DECLARE _exists INT;
498     DECLARE _cur CURSOR FOR
499         SELECT
500             empl_username, job_id, phase_1+phase_2+phase_3 as grade, comments
501         FROM
502             evaluationprocess INNER JOIN job ON evaluationprocess.job_id = job.id
503         WHERE
504             evaluator = in_evaluator AND job_id = in_job_id;
505     DECLARE CONTINUE HANDLER FOR NOT FOUND SET _done = TRUE;
506     -- Handler για exception που μπορεί να προκύψει από το before insert trigger του `user`
507     DECLARE CONTINUE HANDLER FOR SQLSTATE '45000' BEGIN END;
508
509     OPEN _cur;
510     insert_loop: LOOP
511         FETCH _cur INTO _username, _job_id, _grade, _comments;
512         IF _done THEN
513             LEAVE insert_loop;
514         END IF;
515         SELECT
516             EXISTS(
517                 SELECT * FROM evaluationresult
518                 WHERE empl_username = _username AND job_id = _job_id)
519         INTO _exists;
520         IF ((_grade IS NOT NULL) AND (!_exists)) THEN
521             INSERT INTO evaluationresult (empl_username, job_id, grade, comments)
522             VALUES(_username, _job_id, _grade, _comments);
523         END IF;
524     END LOOP;
525     CLOSE _cur;
526 END ;;
527 DELIMITER ;
528
529 -- Procedure που ζητήθηκε, βρίσκει τις αιτήσεις ενός employee, τις ολοκληρωμένες αξιολογήσεις του,
530 -- τον αντίστοιχο αξιολογητή, εμφανίζει και τις μη ολοκληρωμένες
531 DROP PROCEDURE IF EXISTS `proc_find_employee_evaluation`;
532 DELIMITER ;;
533 CREATE PROCEDURE `proc_find_employee_evaluation`
534 (IN in_name varchar(25), IN in_surname varchar(35))

```

```

535 BEGIN
536     declare _username varchar(12);
537     declare _num_eval_req int;
538     declare _num_eval_compl int;
539
540     SELECT username INTO _username
541     FROM user
542     WHERE user.name=in_name AND user.surname=in_surname;
543
544     SELECT count(*) INTO _num_eval_req
545     FROM requestsevaluation
546     WHERE requestsevaluation.empl_username=_username;
547
548     SELECT
549     user.username, requestsevaluation.job_id, grade, evaluator, _evaluator.name, _evaluator.surname
550 FROM user
551     INNER JOIN requestsevaluation
552         ON user.username = requestsevaluation.empl_username
553     INNER JOIN job
554         ON requestsevaluation.job_id = job.id
555     INNER JOIN evaluationresult
556         ON evaluationresult.empl_username = user.username AND evaluationresult.job_id = job.id
557     INNER JOIN user AS _evaluator
558         ON job.evaluator = _evaluator.username
559 WHERE user.name = in_name AND user.surname = in_surname;
560 SELECT FOUND_ROWS() INTO _num_eval_compl;
561
562 IF (_num_eval_compl<_num_eval_req) THEN
563     SELECT 'Evaluations in progress' AS message;
564     SELECT
565     user.username, requestsevaluation.job_id, phase_1, phase_2, phase_3, evaluator, _evaluator.name, _evaluator.surname
566 FROM user
567     INNER JOIN requestsevaluation
568         ON user.username = requestsevaluation.empl_username
569     INNER JOIN job
570         ON requestsevaluation.job_id = job.id
571     INNER JOIN evaluationprocess
572         ON evaluationprocess.empl_username = user.username AND evaluationprocess.job_id = job.id
573     INNER JOIN user AS _evaluator
574         ON job.evaluator = _evaluator.username
575 WHERE user.name = in_name
576     AND user.surname = in_surname
577     AND ((evaluationprocess.phase_1 IS NULL)
578         OR (evaluationprocess.phase_2 IS NULL)
579         OR (evaluationprocess.phase_1 IS NULL));
580     END IF;
581 END ;;
582 DELIMITER ;
583
584 -- Procedure για το login ενός user. Παίρνει σαν όρισμα το username και το password, βρίσκει τον χρήστη καθώς
585 -- και την ιδιότητα του και επιστρέφει τον ρόλο του. Επίσης θέτει αντίστοιχα session variables για χρήση από triggers
586 -- και άλλα procedures.
587 DROP PROCEDURE IF EXISTS `proc_login_user`;
588 DROP PROCEDURE IF EXISTS `evaluation`.`proc_login_user`;
589 DELIMITER ;;
590 CREATE PROCEDURE `proc_login_user`(IN in_username varchar(12), IN in_password varchar(10), OUT out_role varchar(15))
591 BEGIN
592     DECLARE _exists INT;
593     SELECT
594     EXISTS(SELECT * FROM `user`
595         WHERE `user`.`username` = in_username
596         AND `user`.`password` = in_password) INTO _exists;
597
598     IF (!_exists) THEN
599         SELECT NULL INTO out_role;
600     ELSE
601         SELECT
602         CASE
603             WHEN EXISTS(SELECT * FROM `admin` WHERE `admin`.`username` = in_username) THEN "administrator"
604             WHEN EXISTS(SELECT * FROM `manager` WHERE `manager`.`username` = in_username) THEN "manager"
605             WHEN EXISTS(SELECT * FROM `evaluator` WHERE `evaluator`.`username` = in_username) THEN "evaluator"
606             WHEN EXISTS(SELECT * FROM `employee` WHERE `employee`.`username` = in_username) THEN "employee"
607             ELSE NULL
608         END INTO out_role;
609         SET @evaluation_current_user = in_username;
610         SET @evaluation_current_role = out_role;
611     END IF;
612 END;;
613 DELIMITER ;
614
615 -- Triggers για την καταγραφή στον `log`
616 -- Για χρήση με τα αντίστοιχα before triggers. Καταγράφει στον `log` την ενέργεια ενός χρήστη.
617 -- το result καταγράφεται πάντα ως failure.
618 DROP PROCEDURE IF EXISTS `proc_log_before`;
619 DELIMITER ;;
620 CREATE PROCEDURE `proc_log_before`(
621     IN in_action ENUM('insert','update','delete'),
622     IN in_table ENUM('job','employee','requestsevaluation'),
623     IN in_result ENUM('success','failure'))
624 BEGIN

```

```

625 DECLARE _now DATETIME;
626 SELECT NOW() INTO _now;
627 INSERT INTO `evaluation`.`log`
628     (`username`, `action`, `table`, `result`, `datetime`)
629     VALUES (@evaluation_current_user,
630             in_action,
631             in_table,
632             'failure',
633             _now);
634 END ;;
635 DELIMITER ;
636 -- Για χρήση με τα αντίστοιχα after triggers. Επιλέγει το τελευταίο στοιχείο στον `log` και αλλάζει το
637 -- αποτέλεσμα από 'failure' σε 'success'. Βασίζεται στο ότι το after trigger εκτελείται μόνο όταν ενέργεια
638 -- ολοκληρωθεί σωστά. Υποθέτει δύο συνθήκες, ότι ένας χρήστης είναι συνδεδεμένος και κάνει αλλαγές και ότι
639 -- δεν συμβαίνουν batch inserts/updates.
640 DROP PROCEDURE IF EXISTS `proc_log_after`;
641 DELIMITER ;;
642 CREATE PROCEDURE `proc_log_after`()
643 BEGIN
644     DECLARE _id INT;
645     SELECT id FROM log ORDER BY id DESC LIMIT 1 INTO _id;
646     UPDATE log
647         SET result = "success"
648         WHERE log.id = _id;
649 END ;;
650 DELIMITER ;
651
652 -- Procedure που ζητήθηκε, εμφανίζει τις ολοκληρωμένες αξιολογήσεις για μία θέση. Αν υπάρχουν
653 -- εμφανίζει και τον αριθμό των αιτήσεων σε εκκρεμότητα.
654 DROP PROCEDURE IF EXISTS `proc_print_employee_evaluation`;
655 DELIMITER ;;
656 CREATE PROCEDURE `proc_print_employee_evaluation`(IN in_job_id INT)
657 BEGIN
658     DECLARE _active_evaluations INT;
659     DECLARE _final_evaluations INT;
660
661     SELECT COUNT(*) FROM evaluationresult WHERE job_id = in_job_id INTO _final_evaluations;
662     SELECT COUNT(*) FROM requestsevaluation WHERE job_id = in_job_id INTO _active_evaluations;
663
664     IF (!_active_evaluations) THEN
665         SELECT "No candidates for this job." AS "Status";
666     ELSE
667         IF (_active_evaluations = _final_evaluations) THEN
668             SELECT "Finalized evaluation." AS "Status";
669             SELECT * FROM evaluationresult WHERE job_id = in_job_id ORDER BY grade DESC;
670         ELSEIF (_active_evaluations > _final_evaluations) THEN
671             SELECT * FROM evaluationresult WHERE job_id = in_job_id ORDER BY grade DESC;
672             SELECT CONCAT("Evaluation in progress. ", _active_evaluations-_final_evaluations, " requests remaining.") AS "Status";
673         ELSE
674             SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'This should not happen';
675         END IF;
676     END IF;
677 END ;;
678 DELIMITER ;
679

```