PROJECT ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΑΝΑΦΟΡΑ

2020-2021

ΕΜΜΑΝΟΥΗΛ ΠΙΤΣΙΓΑΥΔΑΚΗΣ ΑΜ:1054405 Έ ΕΤΟΣ

ΠΑΝΑΓΙΩΤΗΣ ΣΦΕΝΔΟΥΡΑΚΗΣ ΑΜ : 1054282 Έ ΕΤΟΣ

ΜΕΡΟΣ Α)

Αρχικά φτιάχνουμε την βάση δεδομένων που μας δίνεται στην εκφώνηση.

1)

Έπειτα για το ερώτημα 1 θα πρέπει να προσθέσουμε 3 επιπλέον πίνακες.Ο πρώτος,ο ‘evaluation’ , θα περιέχει πεδία για

τον βαθμό της συνέντευξης του υπάλληλου

τον βαθμό της αξιολόγησης του

τον βαθμό της αξιολόγησης των πτυχίων του

και έπειτα θα υπολογίζεται αυτόματα σε ένα πεδίο του πίνακα το evaluationresult.Τα comments μπορούμε να τα βάλουμε ή να τα παραλείψουμε εφόσον δεν αλλάζουν το αποτέλεσμα.Ο δεύτερος,ο ‘job_application’, θα περιέχει τις αιτήσεις για τις θέσεις εργασίας και ο τρίτος θα είναι ο πίνακας ενεργειών log που λειτουργεί ενημερώνεται με triggers.

2) προσθέτουμε δεδομένα ώστε να εξασφαλίσουμε την λειτουργία της βάσης

3)

3.1) Για το πρώτο procedure βάζουμε ως ορίσματα το όνομα και το επώνυμο του υπάλληλου και έπειτα δηλώνουμε αυτά που ζητούνται και τα εμφανίζουμε με τις εντολές select από τους πίνακες που βρίσκονται,με την προυπόθεση ότι το username είναι ιδιο με το όνομα του υπαλλήλου . Ενώ για να εμφανιστεί το μήνυμα 'Βρίσκεται σε αξιολόγηση' χρησιμοποιούμε εντολή if.

3.2)Το επόμενο procedure φτιάχνεται με παρόμοια διαδικασία με το προηγούμενο δηλαδή εισάγουμε τα ορίσματα και έπειτα επιλέγονται τα στοιχεία αξιολόγησης για την συγκεκριμένη θέση εργασίας και με μια εντολή if ενημερώνεται ο πίνακας evaluation result αν υπάρχουν όλοι οι βαθμοί.

3.3)Το τελευταίο procedure ελέγχει με εντολή if αν έχουν ολοκληρωθεί οι αξιολογήσεις και με επανάληψη(cursor) εμφανίζει τις ολοκληρωμένες .Για να εκτυπωθεί το ανάλογο μήνυμα ,αν είναι σε εξέλιξη, χρησιμοποιούμε μια εμφωλευμένη if και εμφανίζεται το n που μέτρησε μαζί με το μήνυμα που ο n είναι ο συνολικός αριθμός των αξιολογήσεων σε εξέλιξη.

4)Τα triggers εισαγωγής, ενημέρωσης και διαγραφής για τους συγκεκριμένους πίνακες θα χρειαστεί να προστεθούν ξεχωριστά για κάθε πίνακα για όλες τις ενέργειες .Στην συνέχεια ο trigger που δεν επιτρέπει αλλαγή στα πεδία ΑΦΜ, ΔΟΥ , και όνομα βάζει στα στοιχεία την προηγούμενη τιμή τους. Τέλος ο trigger που αποτρέπει σε άλλους εκτός του διαχειριστή να αλλάξει στοιχεία στο προφίλ

υλοποιείται με μια εντολή if και με την εμφάνιση προειδοποιητικού μυνήματος

```
CREATE TABLE `antikeim` (
  `title` varchar(36) NOT NULL,
  `descr` text,
  `belongs_to` varchar(36),
  PRIMARY KEY (`title`),
  CONSTRAINT `ANTIKEIM_TITLE` FOREIGN KEY (`belongs_to`) REFERENCES `antikeim` (`title`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `user` (
  `username` varchar(12) NOT NULL,
  `password` varchar(10),
  `name` varchar(25) NOT NULL,
  `surname` varchar(35) NOT NULL,
  `reg_date` date,
  `email` varchar(30),
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `company` (
```

```
  `AFM` char(9) NOT NULL,
  `DOY` varchar(15),
  `name` varchar(35) NOT NULL,
  `phone` bigint(16),
  `street` varchar(15),
  `num` tinyint(4),
  `city` varchar(15),
  `country` varchar(15),
  PRIMARY KEY (`AFM`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `degree` (
  `titlos` varchar(50),
  `idryma` varchar(40),
  `bathmida` enum('LYKEIO','UNIV','MASTER','PHD') NOT NULL,
  PRIMARY KEY (`titlos`,`idryma`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `employee` (
  `username` varchar(12) NOT NULL,
  `bio` text,
  `sistatikes` varchar(35),
```

```
  `certificates` varchar(35),
  `awards` varchar(35),
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `evaluator` (
  `username` varchar(12) NOT NULL,
  `exp_years` tinyint(4),
  `firm` char(9),
  PRIMARY KEY (`username`),
  CONSTRAINT `EVALR_USER` FOREIGN KEY (`username`) REFERENCES
`user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `EVALR_COMP` FOREIGN KEY (`firm`) REFERENCES
`company` (`AFM`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `job` (
  `id` int(4) NOT NULL,
  `start_date` date,
  `salary` float(6,1),
  `position` varchar(40),
  `edra` varchar(45),
  `evaluator` varchar(12),
```

```
  `announce_date` datetime,
  `submission_date` date,
  PRIMARY KEY (`id`),
  CONSTRAINT `JOB_EVALR` FOREIGN KEY (`evaluator`) REFERENCES `evaluator` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `evaluation` (
  `empl_username` varchar(12) NOT NULL,
  `job_id` int(4),
  `interview` int(11),
  `report` int(11),
  `achievements` int(11),
  `evaluation_result` tinyint(4),
  PRIMARY KEY (`empl_username`),
  CONSTRAINT `EVAL_EMPL` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `evaluationresult` (
  `EvId` int(4) NOT NULL AUTO_INCREMENT,
  `empl_username` varchar(12) NOT NULL,
```

```
  `job_id` int(4) NOT NULL,
  `grade` int(4),
  `comments` varchar(255),
  PRIMARY KEY (`EvId`,`empl_username`),
  CONSTRAINT `EVALRES_EMPL` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `EVALRES_JOB` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `has_degree` (
  `degr_title` varchar(50) NOT NULL,
  `degr_idryma` varchar(40) NOT NULL,
  `empl_username` varchar(12) NOT NULL,
  `etos` year(4),
  `grade` float(3,1),
  PRIMARY KEY (`degr_title`,`degr_idryma`,`empl_username`),
  CONSTRAINT `HASDGR_DGR` FOREIGN KEY (`degr_title`) REFERENCES `degree` (`titlos`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `HASDGR_EMPL` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
CREATE TABLE `job_application` (
  `job_id` int(11) NOT NULL,
  `empl_username` varchar(12) NOT NULL,
  PRIMARY KEY (`job_id`,`empl_username`),
  CONSTRAINT `JOBAPL_EMPL` FOREIGN KEY (`empl_username`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `JOBAPL_JOB` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `languages` (
  `employee` varchar(12) NOT NULL,
  `lang` set('EN','FR','SP','GR') NOT NULL,
  PRIMARY KEY (`employee`,`lang`),
  CONSTRAINT `LANG_EMPL` FOREIGN KEY (`employee`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `log` (
  `username` varchar(12) NOT NULL,
```

```
  `date_time` datetime,
  `success` tinyint(1),
  `kind` enum('insert','update','delete',''),
  `table_name` varchar(25),
  PRIMARY KEY (`username`),
  CONSTRAINT `LOG_USER` FOREIGN KEY (`username`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `manager` (
  `managerUsername` varchar(12) NOT NULL,
  `exp_years` tinyint(4),
  `firm` char(9),
  PRIMARY KEY (`namagerUsername`),
  CONSTRAINT `MAN_COMP` FOREIGN KEY (`firm`) REFERENCES `company` (`AFM`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `MAN_USER` FOREIGN KEY (`namagerUsername`) REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `needs` (
  `job_id` int(4) NOT NULL,
```

```
  `antikeim_title` varchar(36) NOT NULL,
  PRIMARY KEY (`job_id`),
  CONSTRAINT `NEEDS_ANTIKEIM` FOREIGN KEY (`antikeim_title`) REFERENCES `antikeim` (`title`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `NEEDS_JOB` FOREIGN KEY (`job_id`) REFERENCES `job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `project` (
  `empl` varchar(12) NOT NULL,
  `num` tinyint(4) NOT NULL,
  `descr` text,
  `url` varchar(60),
  PRIMARY KEY (`empl`,`num`),
  CONSTRAINT `PROJECT_EMPL` FOREIGN KEY (`empl`) REFERENCES `employee` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `requestevaluation` (
  `empl_username` varchar(12) NOT NULL,
  `job_id` int(4) NOT NULL,
  PRIMARY KEY (`empl_username`,`job_id`),
```

```
  CONSTRAINT `REQEVAL_JOB` FOREIGN KEY (`job_id`) REFERENCES
`job` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

  CONSTRAINT `REQEVAL_LANG` FOREIGN KEY (`empl_username`)
REFERENCES `languages` (`employee`) ON DELETE CASCADE ON
UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;



CREATE TABLE `user_role` (

  `username` varchar(12) NOT NULL,

  `role` set('manager','evaluator','employee','admin') NOT NULL,

  PRIMARY KEY (`username`, `role`),

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

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;



INSERT INTO company
```

(AFM, DOY, name, phone, street, num, city, country) VALUES

(1234, 'CENTRAL ATHENS', 'MASOUTHS', 2102935762, 'MIAOULH', NULL, 'ATHENS', 'GREECE'),

(2345, 'THESSALONIKIS', 'LAVDAS', 2396067901, 'KAROLOY', NULL, 'THESSALONIKI', 'GREECE'),

(3456, 'XANIWN', 'STEIAKAKHS', 2810323459, 'XATZIDAKH', NULL, 'CRETE', 'GREECE');

INSERT INTO degree

(titlos, idryma, bathmida) VALUES

('MHXANIKOS H/Y', 'UNIVERSITY OF PATRAS', 'MASTER'),

('HLEKTROLOGOS MHXANIKOS', 'TEI DUTIKHS ATTIKHS', 'UNIV'),

('APOLYTHRIO', '57o LUKEIO ATHINWN', 'LYKEIO');

INSERT INTO `user`

(username , `password`, name , surname , reg_date , email) VALUES

('mark', '12', 'markos', 'seferlis', '2019-12-20', 'mark@hotmail.com'),

('nick', '123', 'nikos', 'papadopoulos', '2019-12-21', 'nick@hotmail.com'),

('mary', '1234', 'maria', 'makri', '2019-12-22', 'mary@hotmail.com'),

('kostas1', '12345', 'kostas', 'kakouros', '2019-09-11', 'kostas@hotmail.com'),

('marios1', '123456', 'marios', 'martakis', '2019-09-12', 'marios@hotmail.com'),

('dimitris1', '1234567', 'dimitris', 'manis', '2019-09-13', 'dimitris@hotmail.com'),

('manos1', '12345678', 'manos', 'papas', '2019-09-14', 'manos@hotmail.com'),

('baggelis1', '1234567890', 'baggelis', 'lagos', '2019-09-15', 'baggelis@hotmail.com'),

('takis1', '12345678901', 'takis', 'ksenos', '2019-09-16', 'takis@hotmail.com');

INSERT INTO employee

(username, bio, sistatikes, certificates, awards) VALUES

('mark', 'He plays professional football and have worked at walmart', 'nothing', 'certificate of highschool graduation', 'most football goals in a year award'),

('nick', 'He worked for 5 companies and he is as experiences as possible', 'mr nick was the best employee of the month at our company for 12 months', 'highschool diploma,degree of computer engineering', 'worlds best artificial intelligence project of year 2014-2015 award'),

('mary', 'She just completed her senior year of college and plays amateur volleyball', 'mrs maria was one of our best students',

```
                                                    'degree of hlektrologon
mhxanikon',                                         'nothing ');

INSERT INTO manager
(managerUsername,  exp_years,firm) VALUES
('kostas1',        5,                1234),
('marios1',              6,                2345),
('dimitris1',            7,                3456);

INSERT INTO needs
(job_id,     antikeim_title) VALUES
(1,          'servant' ),
(2,          'programming'),
(3,          'director of the company ceo');

INSERT INTO evaluator
(username  ,     exp_years  ,     firm) VALUES
('manos1',       2,               1234),
('baggelis1',    3,                    2345),
('takis1',       4,                    3456) ;

INSERT INTO antikeim
(title ,                                    descr,
                                            belongs_to) VALUES
```

```
('servant',                    'bringing coffee',
                               'n/a'),
('programming ',               'programming at java
language',                     'computer technology'),
('director of the company ceo', 'making important decisions for the
company', 'n/a');

INSERT INTO project
(empl,          num, descr  ,


          url) VALUES
('nick',    1,          'a program of artificial inteligence which plays
chess ,is based on machine learning and is getting better by every
game that it plays' ,   'nickproject.gr'),
('mark',    0,          'n/a',


          'n/a'),
('mary',    0,          'n/a',


          'n/a');

INSERT INTO languages
(employee,      lang ) VALUES
('mark',    'EN'),
('nick',    'EN, FR, GR'),
```

```
('mary',     'EN, SP');

INSERT INTO requestevaluation
(empl_username,      job_id ) VALUES
('mark',             '1'),
('nick',             '3'),
('mary',             '2');

INSERT INTO job
(id,   start_date,          salary,          `position`,      edra,
                evaluator,      announce_date,   submission_date)
VALUES
(1 ,   '2020-01-12',   '1000',   'servant',       'thessaloniki',
       'manos1',       '2019-12-25',          '2020-01-29'),
(2 ,   '2020-01-15',   '10000',  'programmer',    'xania',
       'baggelis1',    '2019-12-25',          '2020-01-29'),
(3 ,   '2020-01-18',   '100000', 'ceo',                'athens',
            'takis1',      '2019-12-25',      '2020-01-29');

INSERT INTO has_degree
(degr_title,                    degr_idryma,
       empl_username,           etos,            grade) VALUES
('MHXANIKOS H/Y',   'UNIVERSITY OF PATRAS',   'nick',
            5,                  10),
('HLEKTROLOGOS MHXANIKOS', 'TEI DUTIKHS ATTIKHS ',    'mary',
                    5,              9) ;
```

```
INSERT INTO evaluationresult
(EVid, empl_username , job_id, grade, comments) VALUES
(12, 'mark', 1, 2, 'mr markos was 20 min late'),
(13, 'nick', 3, 10, 'mr nikos was very polite'),
(14, 'mary', 2, 7, 'mrs maria was very confident');

INSERT INTO evaluation
(empl_username, job_id, interview, report, achievements, evaluation_result) VALUES
('mark', 1, 1, 1, 0, 2),
('nick', 3, 4, 4, 2, 10),
('mary', 2, 3, 3, 1, 7);

INSERT INTO job_application
(job_id, empl_username) VALUES
(1, 'mark'),
(3, 'nick'),
(2, 'mary');

INSERT INTO `log`
```

```
(username , date_time, success, kind, table_name )
VALUES
('kostas1', '2020-01-01', 1, 'delete', 'evaluationresult'),
('manos1', '2020-01-02', 1, 'update',
'evaluation '),
('baggelis1', '2020-01-03', 1, 'insert', 'evaluation');

INSERT INTO user_role
(username, role) VALUES
('mark', 'employee'),
('nick', 'employee'),
('mary', 'employee'),
('kostas1', 'manager'),
('marios1', 'manager'),
('dimitris1', 'manager'),
('manos1', 'evaluator'),
('baggelis1', 'evaluator'),
('takis1', 'evaluator');
```

```
DROP PROCEDURE IF EXISTS employeejobprogress;

DELIMITER $

CREATE PROCEDURE employeejobprogress(IN empl_name
VARCHAR(25), IN empl_surname VARCHAR(35))

BEGIN

        DECLARE usrname VARCHAR(12);

        DECLARE jobid INT;

        DECLARE evusrname VARCHAR(12);


        SELECT username INTO usrname FROM user WHERE
name=empl_name and surname=empl_surname;


   SELECT * FROM job_applications WHERE
empl_username=usrname;

        IF (SELECT evaluation_result FROM evaluation WHERE
empl_username=usrname) = NULL THEN

                SELECT 'Evaluation still in progress';

        ELSE

                SELECT * FROM evaluation WHERE
empl_username=usrname;

        END IF;
```

```
        SELECT job_id INTO jobid FROM evaluation_result WHERE
empl_username=usrname;
        SELECT evaluator INTO evusrname FROM job WHERE id=jobid;
        SELECT name FROM user WHERE username=evusrname;
    SELECT surname FROM user WHERE username=evusrname;
END$
DELIMITER ;

DROP PROCEDURE IF EXISTS getevaluationprogress;
DELIMITER $
CREATE PROCEDURE getevaluationprogress(IN jobid INT)
BEGIN
        DECLARE empl_usrname VARCHAR(12);
        DECLARE jinterview INT;
        DECLARE jreport INT;
        DECLARE jachievements INT;
        DECLARE final INT;

        SELECT interview INTO jinterview FROM evaluation WHERE
job_id=jobid;
        SELECT report INTO jreport FROM evaluation WHERE
job_id=jobid;
        SELECT achievements INTO jachievements FROM evaluation
WHERE job_id=jobid;
        SELECT empl_username INTO empl_usrname FROM evaluation
WHERE job_id=jobid;
```

```
        IF (jinterview!=NULL AND jreport!=NULL AND
jachievements!=NULL) THEN
        SET final = jinterview + jreport + jachievements;
        INSERT INTO evaluationresult (empl_username, job_id, grade,
comments) VALUES (empl_usrname, jobid, final, '');
        END IF;
END$
DELIMITER ;


DROP PROCEDURE IF EXISTS getevaluationemployees;
DELIMITER $
CREATE PROCEDURE getevaluationemployees(IN jobid INT)
BEGIN
        DECLARE finished INT DEFAULT 0;
        DECLARE eval INT DEFAULT 0;
        DECLARE empl VARCHAR(12);


        DEClARE curEval
                CURSOR FOR
                        SELECT evaluation_result FROM evaluation WHERE
job_id=jobid;


        DEClARE curEmpl
                CURSOR FOR
```

```
                SELECT empl_username FROM evaluation WHERE
job_id=jobid;

    DECLARE CONTINUE HANDLER
   FOR NOT FOUND SET finished = 1;

    CREATE TABLE temp (empl_usrname VARCHAR(12), eval_result
INT);

    OPEN curEval;
    OPEN curEmpl;

    getEval: LOOP
        FETCH curEval INTO eval;
        IF finished = 1 THEN
            LEAVE getEval;
        END IF;

        FETCH curEmpl INTO empl;
        IF finished = 1 THEN
            LEAVE getEval;
        END IF;

        IF eval != NULL THEN
            INSERT INTO temp VALUES (empl, eval);
```

```
            END IF;
        END LOOP getEval;
        ClOSE curEval;
        CLOSE curEmpl;

        SELECT * FROM temp
        ORDER BY eval_result DESC;

        DROP TABLE temp;
END$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER job_ins AFTER INSERT ON job
FOR EACH ROW
BEGIN
INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"insert", 'job');
```

```
END $$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER job_upd AFTER UPDATE ON job
FOR EACH ROW
BEGIN
INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"update", 'job');
END $$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER job_del AFTER DELETE ON job
FOR EACH ROW
BEGIN
INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"delete", 'job');
END $$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER employee_ins AFTER INSERT ON employee
FOR EACH ROW
BEGIN
```

```
INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"insert", 'employee');

END $$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER employee_upd AFTER UPDATE ON employee

FOR EACH ROW

BEGIN

INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"update", 'employee');

END $$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER employee_del AFTER DELETE ON employee

FOR EACH ROW

BEGIN

INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"delete", 'employee');

END $$

DELIMITER ;


DELIMITER $$
```

```
CREATE TRIGGER requestevaluation_ins AFTER INSERT ON
requestevaluation

FOR EACH ROW

BEGIN

INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"insert", 'requestevaluation');

END $$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER requestevaluation_upd AFTER UPDATE ON
requestevaluation

FOR EACH ROW

BEGIN

INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"update", 'requestevaluation');

END $$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER requestevaluation_del AFTER DELETE ON
requestevaluation

FOR EACH ROW

BEGIN

INSERT INTO `log` VALUES (username, CURRENT_TIMESTAMP(), 1,
"delete", 'requestevaluation');
```

```
END $$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER prevent_comp_update BEFORE UPDATE ON
company
FOR EACH ROW
BEGIN
SET NEW.AFM = OLD.AFM;
SET NEW.DOY = OLD.DOY;
SET NEW.name = OLD.name;
END $$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER prevent_user_update BEFORE UPDATE ON `user`
FOR EACH ROW
BEGIN
IF (SELECT role FROM user_role WHERE `user`.username=username)
!= "admin" THEN
SIGNAL SQLSTATE '45000'
   SET MESSAGE_TEXT = 'Not authorized for this action';
END IF;
END $$
DELIMITER ;
```