ΕΡΓΑΣΊΑ ΒΆΣΕΩΝ ΔΕΔΟΜΈΝΩΝ

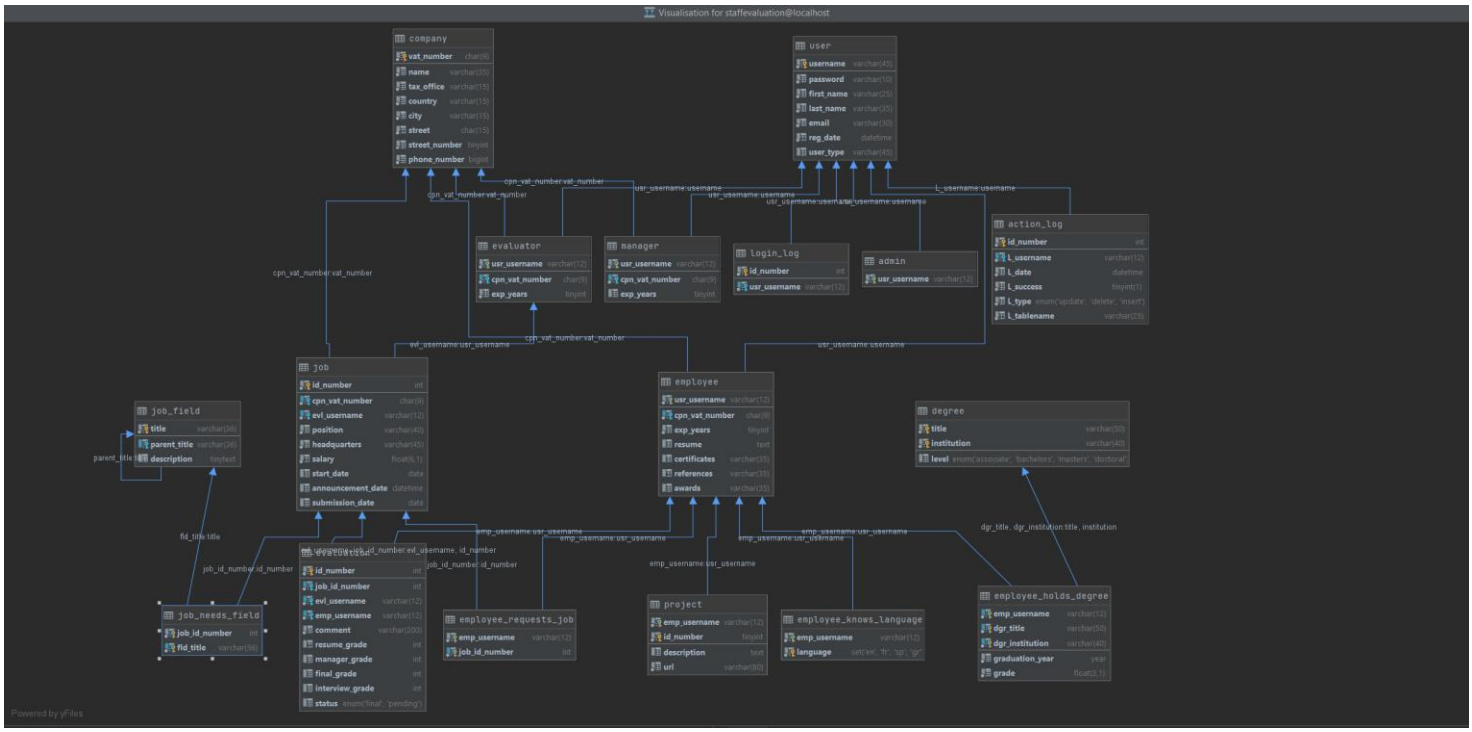
Ορέστης Περράκης ΑΜ: 1067403 3ο Έτος

Γιώργος Γκόνης ΑΜ: 1067390 3ο Έτος

ΜΈΡΟΣ Α

1) Σχεσιακό διάγραμμα και παραδοχές
    a. Στο σχεσιακό διάγραμμα που κατασκευάσαμε χρειάστηκε να προσθέσουμε έναν καινούργιο πίνακα τον log που θα αποθηκεύει τα στοιχεία των κινήσεων των χρηστών στο σύστημα όπως μας ζητήθηκε.
    b. Παράλληλα, έπρεπε να προσθέσουμε διάφορα στοιχεία σε ήδη υπάρχοντες πίνακες για να εξυπηρετήσουμε τις ανάγκες μας. Πιο συγκεκριμένα:
        i. Οι στήλες firm και exp_years ήταν απαραίτητες στο table employee για να αποθηκεύουμε την εταιρεία στην οποία εργάζεται ο εργαζόμενος και τα χρόνια εμπειρίας του.
        ii. Στο table evaluationresults προσθέσαμε τις στήλες:
            1. Interview_grade για τον βαθμό που θα του βάλει ο evaluator μετά την συνέντευξη του. (0<=interview_grade<=4)
            2. Manager_grade για τον βαθμό που θα ορίσει ο manager στον εργαζόμενο σύμφωνα με την κρίση του. (0<=manager_grade<=4)
            3. General_grade για τον βαθμό που θα ανατεθεί στον εργαζόμενο με μοναδικό κριτήριο τις συστατικές και τα certificates που κατέχει παράλληλα με τα project που έχει ολοκληρώσει. (0<=general_grade<=2)
            4. Επίσης, προστέθηκε Trigger στο table αυτό ώστε όταν συμπληρωθούν και οι τρείς βαθμοί (είτε μέσω Insert είτε μέσω Update) ο τελικός βαθμός grade να υπολογίζεται και να αποθηκεύεται αυτόματα στο σύστημα.
            5. Τέλος, ένα πολύ απλό Trigger ελέγχου κατά την εισαγωγή εφαρμόστηκε ώστε να τηρούνται τα όρια των interview_grade, manager_grade και general_grade.

2) Η τελική μορφή του διαγράμματος ήταν η εξής:



3) Τα κατάλληλα triggers και procedures δημιουργήθηκαν για την εξυπηρέτηση των ερωτημάτων της εργασίας
Πιο συγκεκριμένα δημιουργήθηκαν 3 triggers για την υλοποίηση της log:

- DeleteTriggerEmployee
- InsertTriggerEmployee
- logDateTrigger
- UpdateTriggerEmployee

Τα οποία κάνουν αυτό που περιγράφουν το όνομα τους στο log.

Εκτός από την logDateTrigger η οποία εξυπηρετεί έναν συγκεκριμένο σκοπό ο οποίος είναι να θέτει την ώρα ενημέρωσης σύμφωνα με την πραγματική ώρα

Παράλληλα για να υλοποιήσουμε την κατάλληλη αρίθμηση των project κάθε υπαλλήλου χρειάστηκε να δημιουργήσουμε άλλα δύο triggers:

- ProjectBeforeInsert
- ProjectNumberUpdaterAfterDelete

Και τα δύο αφορούν την ενημέρωση του αριθμού των πρότζεκτ.

4) Όσον αφορά τα Procedures που μας ζητήθηκαν ο κώδικας του καθενός ακολουθεί ανάλογα:

3.1

```
DELIMITER $
DROP PROCEDURE IF EXISTS employee_promotion_status$
CREATE PROCEDURE employee_promotion_status(IN e_name VARCHAR(25), IN e_surr VARCHAR(35))
BEGIN
    DECLARE uname VARCHAR(12);
    DECLARE ejid INT(4);
    DECLARE evname VARCHAR(12);
    SELECT username INTO uname FROM user WHERE e_name = user.name AND e_surr = user.surname;

    SELECT job_id INTO ejid FROM requestevaluation WHERE empl_usrname = uname;
    SELECT evaluator INTO evname FROM job WHERE id = ejid;

    SELECT 'The evaluation request list of this employee:';
    SELECT * FROM requestevaluation WHERE empl_usrname = uname;

    SELECT 'The completed evaluations of this employee:';
    SELECT * FROM evaluationresults WHERE ejid = job_id AND grade IS NOT NULL;

    SELECT 'The name and surname of the employees evaluators:';
    SELECT name, surname FROM user WHERE evname = user.username;


END$

DELIMITER ;
```

3.2

```
DELIMITER $
DROP PROCEDURE IF EXISTS evaluation_status$
CREATE PROCEDURE evaluation_status(IN tempjid INT(4), IN teval VARCHAR(12))
BEGIN
    DECLARE tempGrade INT(4);
    DECLARE tempEmpl VARCHAR(12);
    DECLARE tempStatus ENUM('FINAL','PENDING');
    DECLARE cflag INT;
    DECLARE gradeCursor CURSOR FOR
        SELECT grade,empl_usrname FROM evaluationresults WHERE tempjid = job_id AND teval = eval;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET cflag = 1;
    OPEN gradeCursor;
    SET cflag = 0;
    FETCH gradeCursor INTO tempGrade, tempEmpl;
    WHILE (cflag = 0) DO
        IF ( tempGrade IS NOT NULL ) THEN
            SET tempStatus = 'FINAL';
        ELSE
            SET tempStatus = 'PENDING';
        END IF;

            UPDATE evaluationresults
                SET status = tempStatus
                WHERE job_id = tempjid AND teval = eval AND empl_usrname = tempEmpl;

        FETCH gradeCursor INTO tempGrade, tempEmpl;
    END WHILE;
    CLOSE gradeCursor;

END$
DELIMITER ;
```

3.3

```
DELIMITER $
DROP PROCEDURE IF EXISTS job_candidate_decider$
CREATE PROCEDURE job_candidate_decider(IN jid INT(4))
BEGIN

    DECLARE tempGrade INT(4);
    DECLARE tempEmpl VARCHAR(12);
    DECLARE tempStatus ENUM('FINAL','PENDING');
    DECLARE counter1 INT;
    DECLARE counter2 INT;
    DECLARE diff INT;
    DECLARE cflag INT;
    DECLARE statusCursor CURSOR FOR
        SELECT status FROM evaluationresults WHERE job_id = jid;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET cflag = 1;
    OPEN statusCursor;
    SET cflag = 0;
    SET counter1 = 0;
    SET counter2 = 0;
    FETCH statusCursor INTO tempStatus;
    WHILE (cflag = 0) DO
        SET counter1 = counter1 + 1;
        IF(tempStatus = 'FINAL') THEN
            SET counter2 = counter2 + 1;
        END IF;
        FETCH statusCursor INTO tempStatus;
    END WHILE ;
    CLOSE statusCursor;
    SELECT counter1,counter2;

    IF(counter1 = counter2) THEN
        SELECT 'Finalised evaluations';
        SELECT empl_usrname,grade FROM evaluationresults WHERE job_id = jid ORDER BY grade DESC;
    ELSE
        SELECT 'Finalised evaluations';
        SELECT empl_usrname,grade FROM evaluationresults WHERE job_id = jid AND status = 'FINAL' ORDER BY grade DESC;
        SET diff = counter1 - counter2;
        SELECT 'The evaluations that haven not been finalized are:',diff;
    END IF;
END$
DELIMITER ;
```