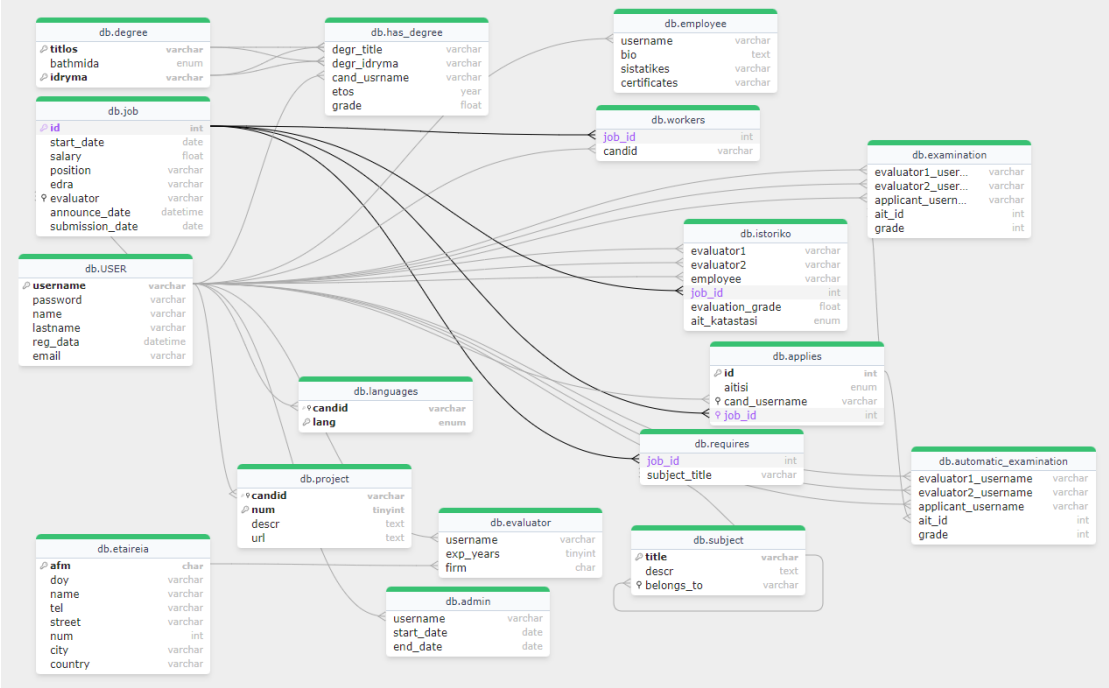
PROJECT: BUSINESS EVALUATION PLATFORM DATABASE SIMULATION

CREATOR: GEORGE D. LALAS

REPORT:

This database project is a simulation of a business evaluation firm that evaluates employees for businesses based on their qualifications. All of the files have executable examples.

Relational Diagram:



1) DATABASE CREATION

Implemented in the files database_project and database_project_data.

Initially, the Database was created, and values were inserted.

USER: Stores user information such as username, password, personal details, registration date, and email. The username is the primary key.

employee: Contains details of employees, such as bio, statistics, and certificates, and links to the USER table via username.

languages: Stores the languages spoken by users (candid is the user) with a set of languages. Links to the USER table.

project: Stores project details for users, such as project number, description, and URL. Links to the USER table.

degree: Stores academic degrees with a title, degree level (BSc, MSc, PhD), and institution. Indexed by idryma (institution).

has_degree: Links users to their academic degrees, containing information about the degree title, institution, grade, and year obtained.

Etaireia (company): Stores company details such as tax identification number (AFM), company name, address, and other related information.

evaluator: Stores evaluators (users who evaluate jobs), along with their years of experience and the company they belong to. Linked to both USER and etaireia.

job: Stores job information such as the job position, salary, and the evaluator who announced the job. Links to the USER table for evaluators.

applies: Represents job applications, with a status for the application (energi (active), oloklиromeni (completed), akyromeni (canceled)) and the link between the candidate (cand_username) and the job.

subject: Stores information about subjects (possibly courses or job requirements) and links to other subjects in a hierarchical manner.

requires: Links jobs to required subjects (skills or qualifications), connecting the job table with the subject table.

For the database, I used the tables from a previous submission. New data was inserted into the database. These data entries have simple names and numbers, with usernames following the format "userX," where X is a number. Additionally, the necessary consistency between the tables was maintained, ensuring that jobs require the appropriate qualifications and that degrees align with those qualifications. Users are registered in ascending numerical order across all entries: user1 to user18 are registered as employees. user19 to user36 are registered as evaluators. Later, user37 to user40 were added as the Database Administrators.

1) APPLICATION MANAGEMENT

Implemented in the files ait, ait1, and ait2.

The first change was made to the tables, adding the attribute aitisi enum("energi" (active), "oloklиromeni" (completed), "akyromeni"(cancelled)). New applications are inserted with the status 'energi'.

In ait, a stored procedure named complete was created. When called, it sets all active applications where the current date (using CURDATE) has exceeded the job start date to 'oloklиromeni'. This is done using the DATEDIFF function, where the difference is less than 0.

Next, a trigger named ait was created, which is triggered before insertion into the applies table. Again, using the CURDATE function, if the difference between the

current date and the start date is greater than 15 days, the application is set to 'energi'; otherwise, it is set to 'akyromeni'.

In ait1, there are two triggers. The first is triggered before insertion into the applies table, selecting the number of active applications for the employee and checking if it is greater than 3. If so, the application is inserted as 'akyromeni'. The second trigger, ait1up, is triggered before an update on the applies table and prevents the reactivation of canceled applications for employees with 3 or more active applications.

In ait2, a trigger named ait2 is created, which is set off after an update on the applies table. Similar to ait, it finds the difference between the current date (CURDATE) and the start_date. If the application was active and the difference (DATEDIFF) is greater than 10, it cancels the application.

It is worth noting that due to the CURDATE function, start dates may need to be updated for a more accurate presentation.

3) EVALUATION

Implemented in the files eva, eva2, w, and delete_examination.

TABLES CREATED FOR THIS SEGMENT:

The examination table is a temporary table where the evaluation process takes place. Applications from employees are entered here for evaluation, and grades from evaluators or automatic evaluation results are recorded.

The automatic_examination table is also a temporary table where data is entered when there is no evaluator, and the automatic evaluation process takes place. The results are then passed to the examination table. Its attributes are the same as the examination table.

The workers table contains the employees who will ultimately get the job position because they had the highest grade.

Initially, the examination table was created, and some values were inserted into it.

In the eva file, a complex procedure named evaluation was implemented. This procedure simulates the evaluation of an application by evaluators. Each evaluator "assigns" their grade in the form of a random integer between 0 and 20.

The procedure takes as input the names of two evaluators (which can be NULL), the name of an employee, and the ID of a job position. Depending on the cases:

- If both evaluators exist, the application is graded by both, and the average grade is calculated.
- If only one evaluator exists, only that evaluator grades the application, and the result is inserted into the examination and automatic_examination tables.
- If no evaluators exist, the grade is set to 0 and inserted into both tables.

Because MySQL did not allow updating the table for which a trigger is triggered, a new table automatic_examination is created for the automatic evaluation. The entries where at least one evaluator is missing are inserted into this table from the procedure: evaluation. The automatic evaluation process will be performed in this table. Its attributes are the same as examination.

In eva2, a trigger is implemented, automatic_evaluation, which is triggered before each entry into the automatic_examination table.

Initially, the appropriate connections are made with the other tables so that the correct values for the employee's qualifications are selected.

The grade that has been passed is selected and two cases are taken.

If it is 0, then the result is calculated entirely from the employee's qualifications with points awarded for their degrees, languages and the projects they participated in.

If it is not 0 then the result is calculated as the average of the grade that the evaluator passed in the eva file and the bonus points from the employee's qualifications.

In both cases the trigger updates the examination table replacing the existing result with the final result.

There were some issues in the execution of eva2 & eva.

The most serious is that in the update, after running with 2 inputs to eva with NULL as arguments, only the first one is updated in examination. Also, if 1 call is made to eva2, it will not update examination at all and if there are more than 2 calls to eva2, it gives an error.

Many solutions were tried in eva2 such as using a cursor or via DELETE & INSERT without the desired results.

It is worth noting that eva2 must be run before eva to obtain the results of the automatic evaluation.

In the file w, the mechanism for selecting employees for a job position is implemented.

Then in w, the procedure: AssignJobPosition wascreated, which takes the ID of a job position as an argument.

The procedure will select the employee with the highest final grade and the lowest application ID in case of a tie. Because there is no mechanism for recording the application date and the previous queries are made with CURDATE, the one with the lowest application ID is selected, because the IDs in applies are auto_increment. This means that the one with the lowest ID number has applied earlier.

Finally, it inserts the job ID and the username of the employee selected in the workers table.

Then eva was modified and an extra condition was added that checks if the application is canceled. In case it is, the result is 0 and it is set as complete so that it can be entered into the history with a procedure, insert_on_istoriko, which will be presented later. Also in all other cases of eva the application is also set to complete.

In the delete_examination file a procedure named delete_completed_examinations is implemented. It is a simple procedure which, using a nested select in delete to cross-reference the results, deletes all entries in the examination table as well as in the automatic_examination table where the application status is complete.

It is worth noting that it must be run after ist otherwise the entries will be lost.

4) HISTORY

Implemented in the ist file

TABLES CREATED FOR THIS SEGMENT:

The istoriko table contains the history of evaluations, where all completed applications are recorded.

In the ist file, a procedure named insert_on_istoriko is implemented. This procedure uses a cursor to read data from an INNER JOIN of the examination and applies tables for applications with the status 'oloklиromeni' and inserts them into the istoriko table.

5) EVALUATOR CHECK

Implemented in the specific_eval_check file.

A procedure named check_evaluation is implemented, which takes as input the username of an evaluator, the username of an employee, and the job ID for which the employee was evaluated. It checks if the evaluator has evaluated the employee for that job and returns the grade.

6) INSERT/CANCEL/ACTIVATE PROCEDURE

Implemented in the ex file.

A procedure named ex is implemented, which takes as input the username of an employee, a job ID, and one of three letters: 'i', 'c', or 'a'. Depending on the letter, it either inserts a new application, cancels an active application, or reactivates a canceled application.

Initially, a check is made for which letter the user has selected.

If he has selected i

an entry is made in the applies and examination tables with two randomly selected evaluators for the evaluation.

If he has selected c and the application is active, it cancels it and displays a success message. If it is not active, no change is made and the appropriate message is displayed.

If he has selected a and the application is canceled, it activates it and displays a success message. If it is not canceled, no change is made and the appropriate message is displayed.

7) 60.000 INSERTS ON THE HISTORY TABLE

In the ist_60k file, a procedure named insert_on_istoriko_60k is implemented
In the procedure there is a mechanism that, using a loop, fills the istoriko table with 60000 records.

The names of the evaluators, employees thanks to the fact that they are of the form userX using CONCAT and RAND I connected the string ‘user’ with a randomly generated number. For the evaluators the random numbers are from 19-36 while for the employees they are from 1-18. Also, the result was randomly entered with a random number from 0-20 and the job id from 1-23. The applications are automatically entered as complete without further checks.

All these random values are entered in the istoriko table until the counter reaches the number I defined.

It is worth noting that it was difficult for me, due to my low end laptop, to execute the command, as MySQL crashed. For this reason, I set the procedure to a smaller number and called it more times.

8) INDEXES

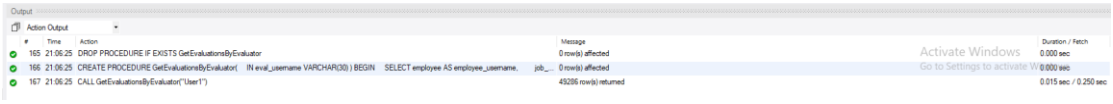
Implemented in the files database_project, index1_procedure, and index2_procedure. Indexes were added to the database for faster searching.

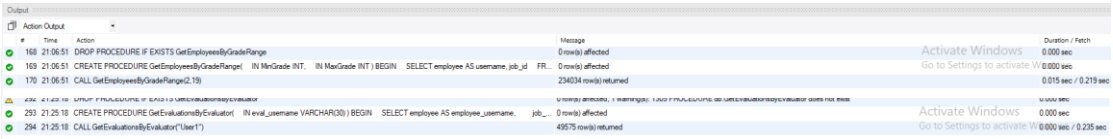
In index1_procedure, a procedure is implemented that takes two values (MinGrade, MaxGrade) and returns the employees and job positions for those who have been graded within that range.

In index2_procedure, a procedure is implemented that returns the employees and job IDs from the istoriko table when a specific evaluator, whose value is given in the eval_username variable, participates as evaluator1 or evaluator2.

Below are the execution times before and after adding the index indexes. At first many inserts were made using the ist_60k file and then the index1_procedure and index2_procedure were executed.

Without the use of the index:





With the use of the index:

The time required to execute the procedures is reduced through the use of indexes in the istoriko table **.**

9) ADMINISTRATORS

Implemented in the DBA file.

TABLES CREATED FOR THIS SEGMENT:

The DBA table contains the Database Administrators.

The log table records the actions of the administrators.

Initially, the log table was created, with enums for the names of the three actions INSERT UPDATE DELETE and three tables degree, job and user, the name of the administrator, and the date and time of the action.

Nine triggers were created, which fire after the three actions on the three tables described in the exercise. Each trigger writes the action and the table it was triggered on to the log table, selecting the appropriate value. It also assigns a randomly selected administrator and the current_timestamp.