

Ανδρέας Λεβής (ΑΜ: 03119160)  
Μαρία-Λαμπρινή Μπαρτσιώκα (ΑΜ: 03119100)  
Χρυσούλα Πανηγυράκη (ΑΜ: 03119010)

**Databases Project**

**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ % ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

*ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ*

Contents

[I. Relational Diagram 2](#_Toc105190630)

[II. Database in SQL 3](#_Toc105190631)

[Tables 3](#_Toc105190632)

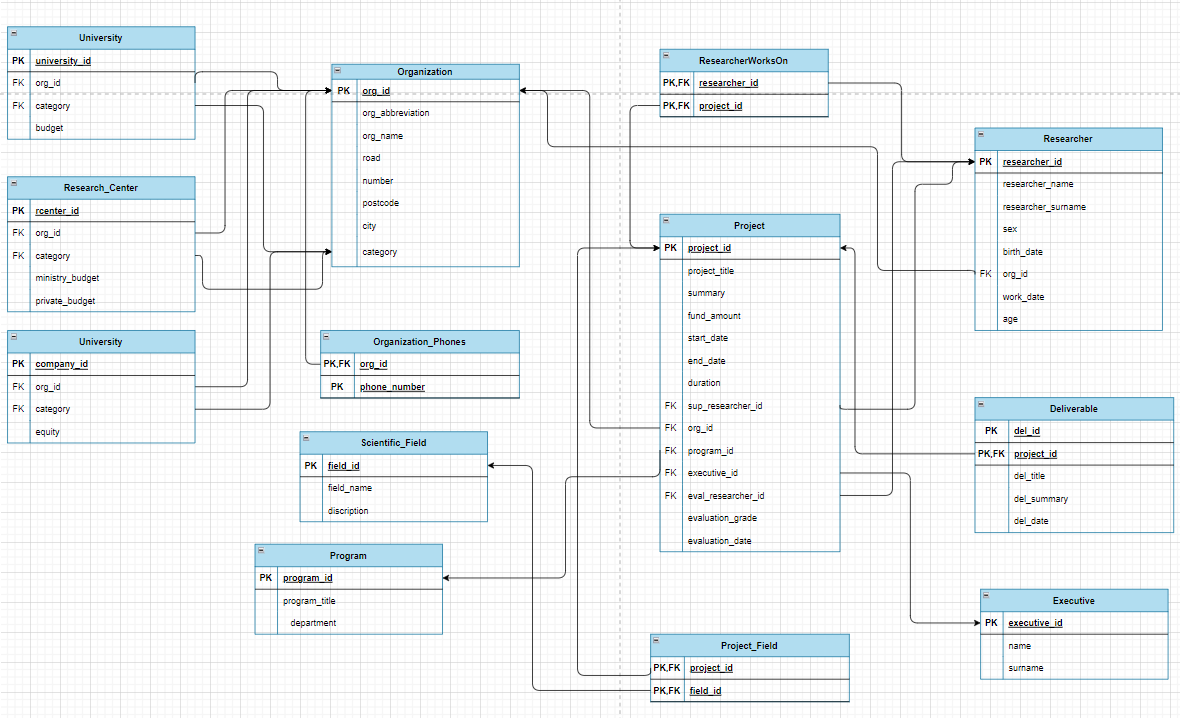
[Other 7](#_Toc105190633)

[III. Organization of Files 8](#_Toc105190634)

[IV. Prerequisites for Database and App Development 9](#_Toc105190635)

[V. Steps for Installation 9](#_Toc105190636)

# Relational Diagram



ER Diagram and Relational Diagram illustrate the logic within a database and how individual components relate to one another. Such diagrams lay out the wireframe of internal database structures to aid in understanding, improving efficiency, and debugging logical errors. The key pieces are the entities, attributes, and the relationships between them.

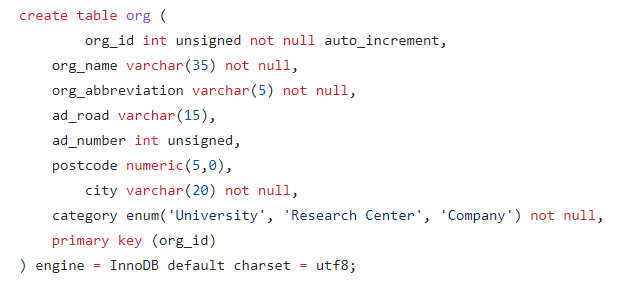
The above relational diagram describes the design of the database of a system which stores, manages and analyzes information for the Hellenic Foundation for Research and Innovation (ELIDEK). This database stores *organizations* (universities, research centers or companies) that manage *projects* and employ *researchers* to carry out these projects. The projects belong to a scientific *field*, while they are financed by ELIDEK *programs*, they have employees, *supervisors*, *evaluators* and *managers* and they may have individual *deliverables*. The necessary data for what was mentioned is stored in the database and the relationships between them are indicated in the diagram with arrows. The separation between entities of the same category is done mainly with the primary keys - IDs.

# Database in SQL

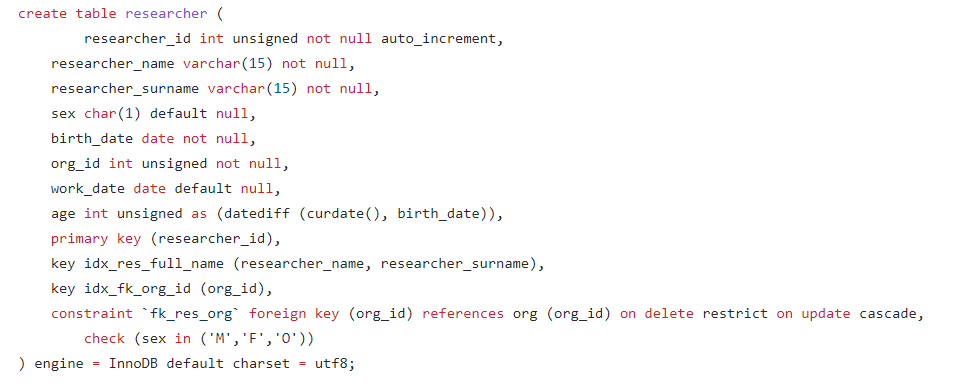
## Tables

The following is the design of the database in SQL language. The tables that are depicted in the relational diagram are created and their primary and foreign keys are defined, based on their importance and the relationships that connect the tables. As for the characteristics of the tables, they are set to NOT NULL when they are deemed necessary for the existence of each element of the respective table and are needed during its initialization. For this reason, the primary keys are always NOT NULL. As primary keys in each table are considered the IDs that are unique for each entity. Foreign keys are also set to NOT NULL, in order to be consistent with the primary keys of the relationships to which they belong.

* Organization

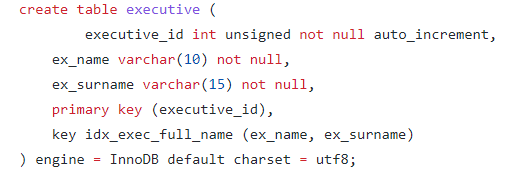


* Researcher

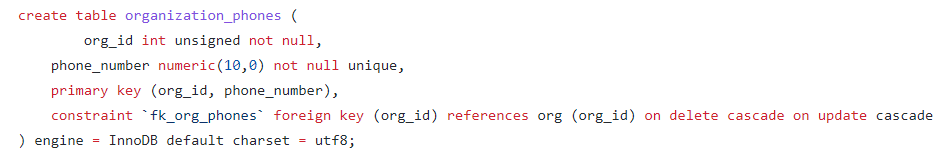


Attribute org\_id is foreign key that matches in a relationship one organization to many researchers. An index res\_full\_name is also made to contain the name and suername of a researcher in one attribute.

* Executive

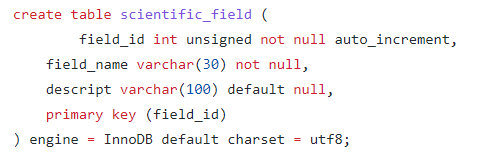


* Organization Phones

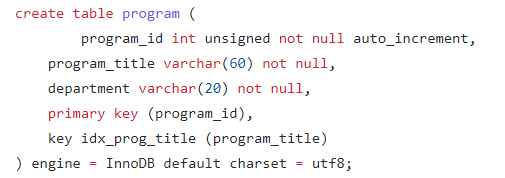


Attribute org\_id is foreign key that matches in a relationship one organization to many phones.

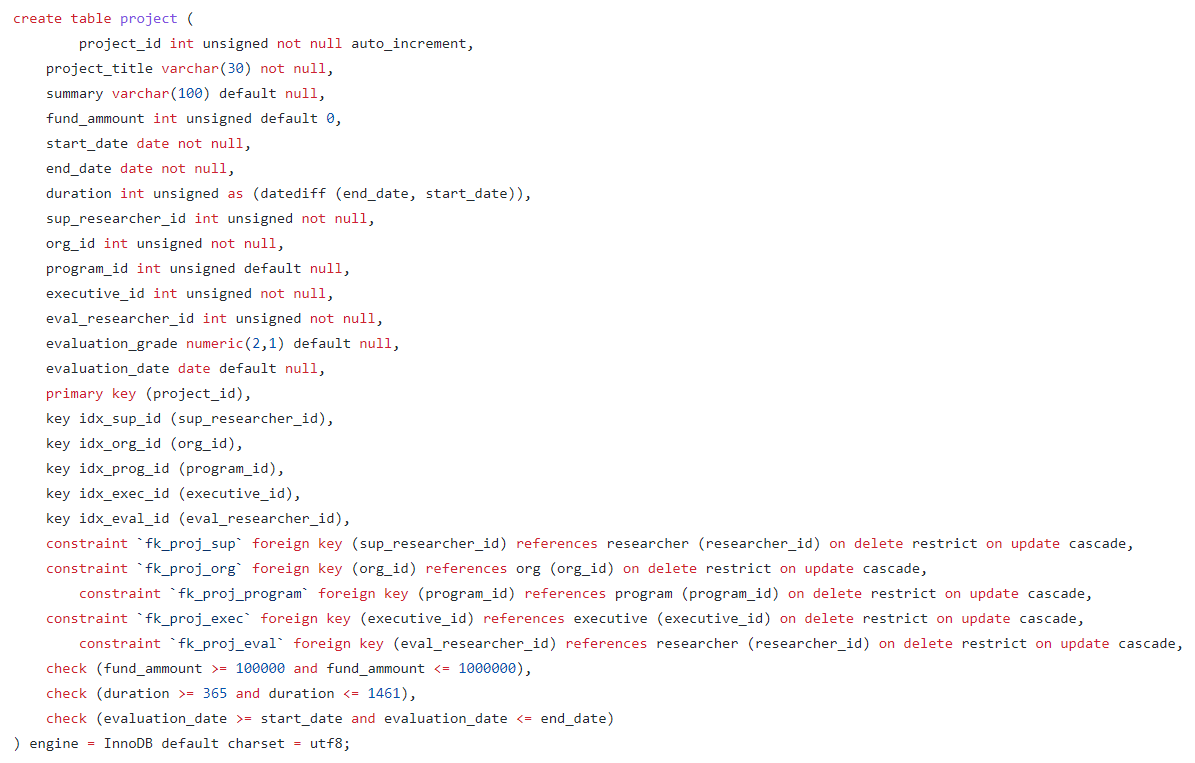
* Scientific Field



* Program

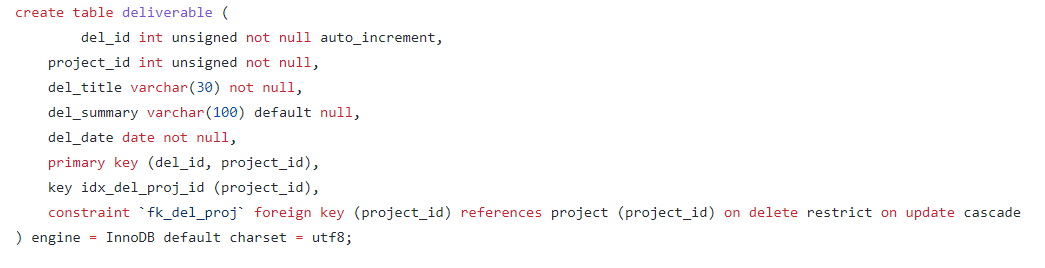


* Project



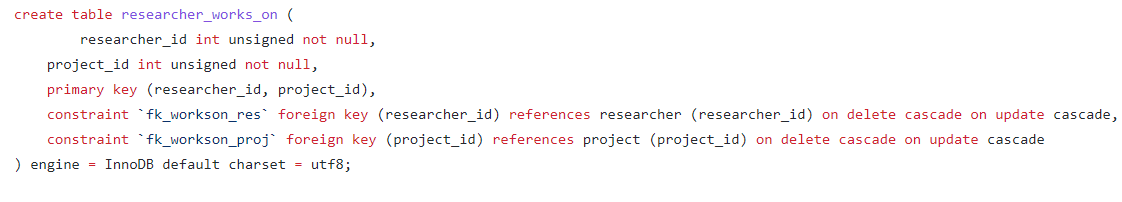
Attribute sup\_researcher\_id is foreign key that matches in a relationship one supervisor to one project. Attribute org\_id is foreign key that matches in a relationship one organization to many projects. Attribute program\_id is foreign key that matches in a relationship many projects to one program. Attribute executive\_id is foreign key that matches in a relationship one executive to one project. Attribute eval\_researcher\_id is foreign key that matches in a relationship one evaluator to one project. In addition, the values of the fund amounts, the duration and the evaluation date of the projects are checked to be within the acceptable limits.

* Deliverable



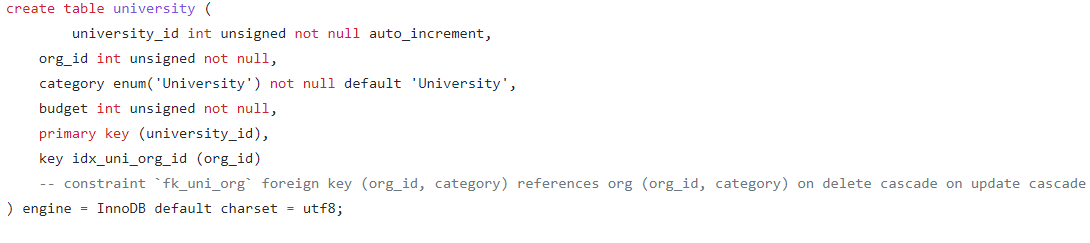
Attribute project\_id is foreign key that matches in a relationship one project to many deliverables.

* Researcher Works On



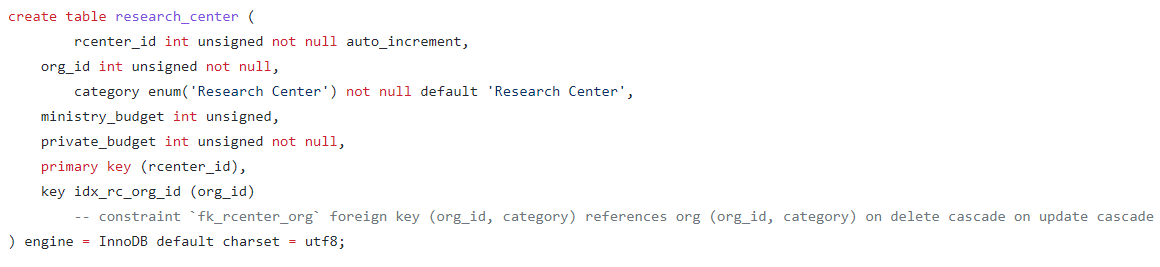
The above table demonstrates the relationship “works on” between researcher who works at a project and is has no specific role. Attributes researcher\_id and project\_id are foreign keys that matches in a relationship many researchers to one project.

* University

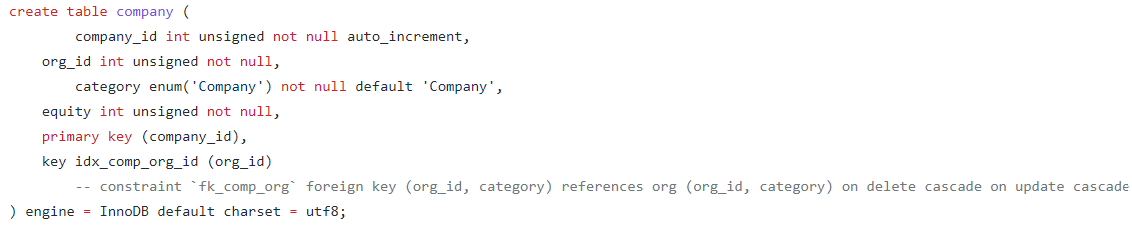


An organization can be either a University or a Research Center or a Company. So an index is created for foreign keys org\_id and category in order to match in a relationship many universities to one organization. The tables for research center and company are created below in a same way.

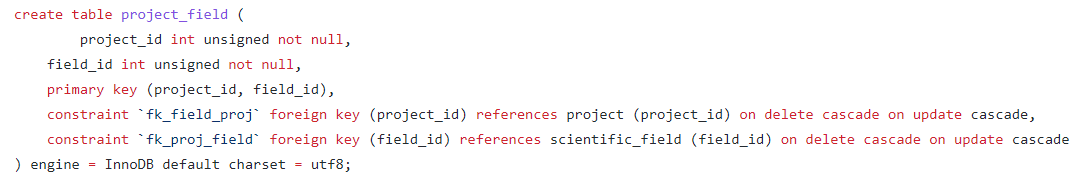
* Research Center



* Company



* Project Field

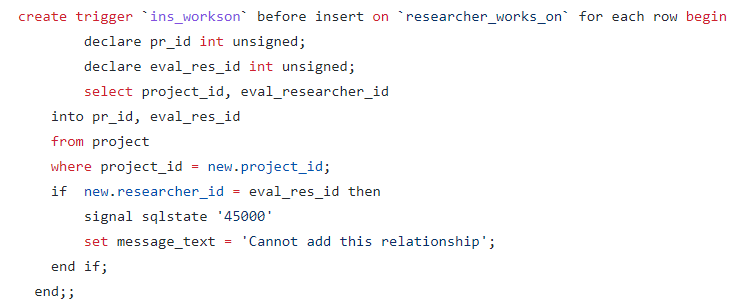


The above table demonstrates the relationship “belongs” between project which belongs to a scientific field. Attributes project\_id and researcher\_id are foreign keys that match in this relationship many projects to one field.

## Other

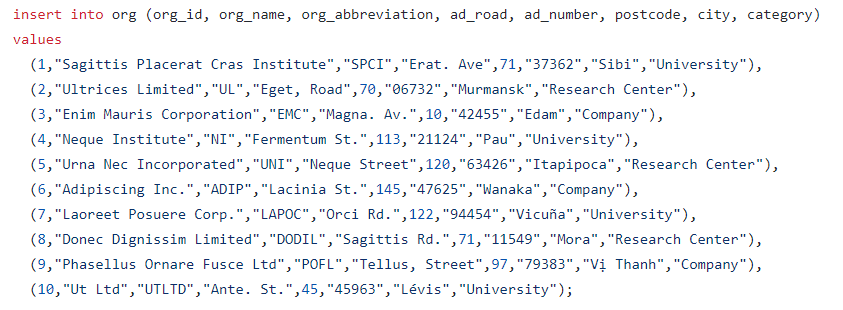
**Indexes** are defined for the attributes that are used the most so as to help queries run faster. All of them are obtained in the tables and have already been analyzed.

**Triggers** are stored procedures which automatically invokes whenever a special event occurs in the database and especially when a row is inserted or a column is being updated on a table. One of them is given as an example.



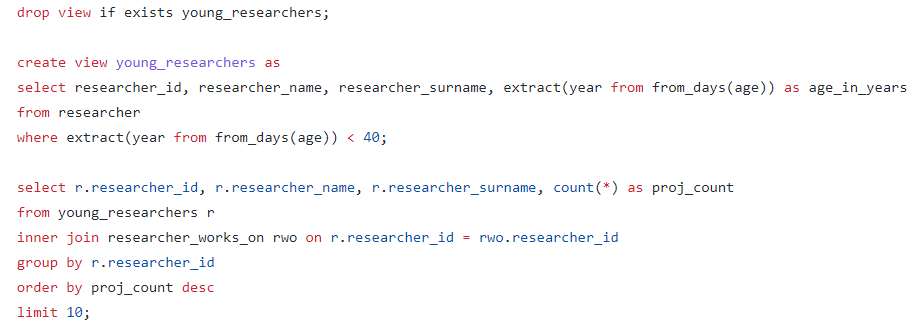
This trigger gives an error message if a new researcher who is entered in the table “researcher\_works\_on” has already been an evaluator for another project.

**Insert into** is a mandatory statement for every database in order to have values in its tables and to be able to run its queries. At first all the data are being deleted and then new data are inserted in the database. User can also add an insert into command to add individual entities.



Data are inserted by rows accordingly to the attributes of each table, as it is shown above. In our implementation the database runs fake data generated from online app.

**Queries** request for data or information from a database table or combination of tables. Here queries implement the questions of the project. Queries are often combined with **Views**, that is virtual tables based on the result-set of an SQL query. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. For instance:



At the beginning of every view we have to delete objects from this view, that may be pre-existing, with drop command. For answering question 3.6 we created a view table which contains all the researchers (independently of their role) that are under 40 years old. Then, we created a query which selects from this view, joined with the relationship table “researcher\_works\_on”, the number of active projects that every researcher works on.

# Organization of Files

In addition to the files analyzed above and relating to the database in SQL, there are a number of other files for the implementation of the server side and client side of the application.

**Routes** are the endpoints of the application. This means that they form a unique URL where a specific task is completed. For example, in our case http://localhost:3000/young\_res is used for hosting the table of young researchers working in ELIDEK project.

**Controllers** are the logic of the routes. In these files the sql queries take place, and then the result can be either returned rendered or in other ways (like JSON).

**Views** contain the web pages of the project.

**Config** contains the connection of the database.

# Prerequisites for Database and App Development

* Open-source, cross-platform, back-end JavaScript runtime environment for executing JavaScript code outside a web browser: <https://nodejs.org/en/>
* Relational database management system: <https://dev.mysql.com/downloads/installer/>

# Steps for Installation

For the management and development of the database we used “*MySQL WorkBench*” and “*XAMP*”, while for building and debugging the web application we used “*Visual Studio Code*” in JavaScript.

You need to download all the files from GitHub in the format they already have and save them in a parent folder. The you have to load them in the environment you prefer.

In order to run the files, there are some required NodeJS dependencies:

* Minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications: <https://www.npmjs.com/package/express>
* Mysql driver: <https://www.npmjs.com/package/mysql2>
* A templating engine: <https://www.npmjs.com/package/ejs>
* A command-line interface (CLI) utility that wraps your Node app, watches the file system, and automatically restarts the process: <https://www.npmjs.com/package/nodemon>
* First thing to do is the creation of the tables and the inserting of the data in them. For this purpose, open a terminal and write: «npm run insert» and wait for confirmation message.
* Suppose you have already install all the above dependencies, you are ready to type on the terminal: «npm start»
* When the message: <Server running on port 3000!> appears, you can go to page of <https://localhost:3000> on your web browser and browse on ELIDEK website.