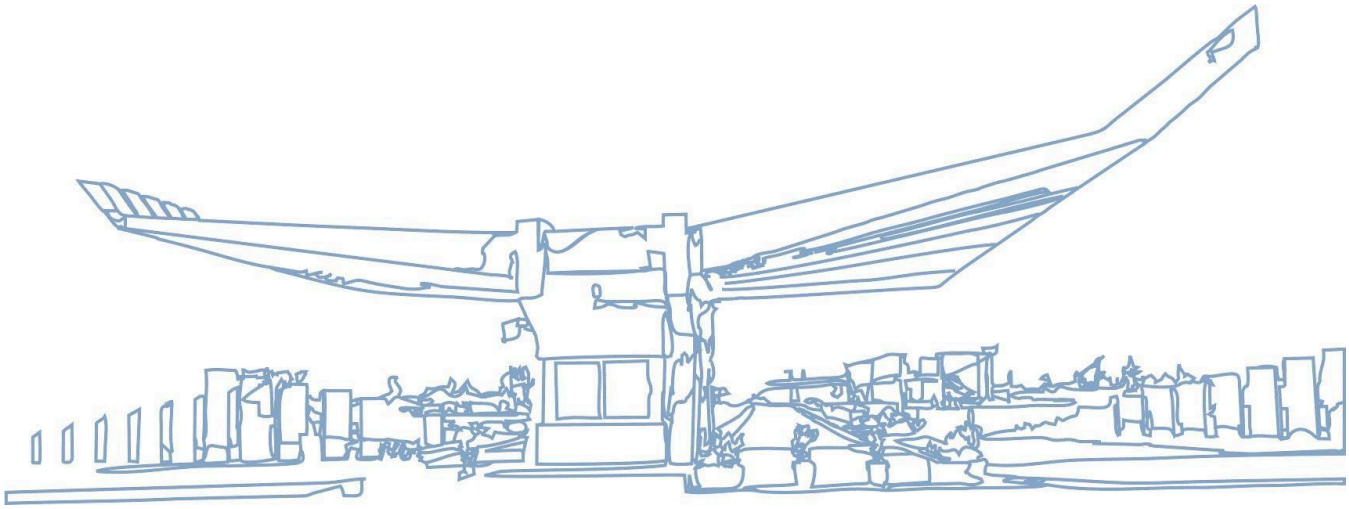
DATABASE MANAGEMENT SYSTEMS

[EIS - EPOKA INTERACTIVE SYSTEM]



Maida Daulle

[EIS] Requirements Specification

Table of Contents

Project Outline And Planning

1.1 Project Description

Epoka Interactive System *(EIS)* is a system where most of the procedures can be managed within the computer medium. It is exclusively accessed by Epoka University students, lecturers as well as staff members. Previously only found as a web page, now it is also available in the form of an app making its accessibility easier and faster. Its objective is for students to engage, succeed, and get the most out of their time at university.

The accounts are created by authorized members adding all the necessary identifying information for each individual when they become a part of the university. Then it can be easily accessed by the Epoka email followed by a password, as a result carrying out a requested action that is offered.

EIS allows students to:

- View their personal information
- Select the courses for each semester and have access to their academic records
- Apply for document requests
- Register for additional or resit exams
- Keep track of their financial status
- Have a personalized timetable according to their selected courses
- Check their attendance
- Receive up-to-the-minute notifications

With an increasing amount of data every year, it is very important to store and organize it efficiently. A well organized database can help maintain the system in the desired ways. Designing a database is considered a task that will always require precise planning and implementation. When designing a database the main focus should always be on the requirements and what the system offers.

After being assigned the project, we got together as a team and started planning an idea of how we would manage to use the time and resources we had available in order to help us in the realization of this project, as well as discussed the tasks that each of us would be responsible for in the creation of the database for EIS.

1.2 Purpose and scope of this specification

Our strategies and procedures for the database project development:

Step 1: *Requirements collection, definition, and visualization*

As a starting point we consider the detailed observation of every possible service EIS offers regarding a student's point of view and a lecturer's point of view. After gathering all the information that would be required for the future database to contain, including how it would store the data, what features and capacities it would have, and so on, all of the work was well-documented and kept up to date in a shared document so that we could always track each other's progress to then later be followed by the visualization of our idea.

Step 2: *Conceptual database model*

The following step was creating a conceptual data modeling technique (ER Diagrams in our case) to visualize our work. Attached with all the necessary notes made throughout our work to have a better understanding of what has been done as well as expressing ideas, proposals, improvements and more.

Step 3: *Database modeling (logical database modeling)*

In this step we provided the associated RS schema, with notes on necessary conversions between (ER Diagrams <-> Relational Schemas).

Step 4: *Database implementation*

Usage of a DBMS to implement the database model as an actual database.

Step 5: *Database deployment*

Enabling end users to access the database system.

Step 6: *Database use*

The insertion, modification, deletion and retrieval of the data in the database system.

Step 7: *Database administration and maintenance*

Ensuring the security of the data stored in the database, making sure the database's content has adequate space as well as Implementing the backup and recovery procedures.

Every step prior leads to the mirroring of the information in the form of queries executed in SQL.

Research And Idea Consolidation

2.1 Proposals For Technologies And Techniques To Use

The development of a successful database can be achieved by following a guidance consisting of steps we had mentioned and implemented prior, considering it as a form of organization to not miss an important factor. The database being created in this project is considered as a large one regarding the fact that it can be used by hundreds of end users. Despite the scope of the database they all go through the same essential development processes, regardless of their different extents.

In order for a database to be kept up to date some features must be available when it comes to updating the data such as:

- *Insert Operation* - Adding new data in the relation
- *Delete Operation* - Deleting existing data from the relation
- *Modify Operation* - Changing existing data in the relation

Another important feature we were careful about was *Updating Anomalies* caused by the primarily mentioned relations. We wanted to make sure that mistakes or inconsistencies that arise from modifying data in a database wouldn't happen in our project.

There is a large number of entities made of even a larger number of attributes when it comes to EIS, despite this fact a key idea in database architecture called *Functional Dependencies*, which can be used to connect and create relationships between the database's attributes as well as help to minimize data redundancy and improve data integrity. These were two of the reasons why this was a key point in the building of our database.

The types of the functional dependencies used in this database project for the normalization process are as follows:

- ***Partial*** functional dependency
- ***Full key*** functional dependency
- ***Transitive*** functional dependency

2.2 Core Concept Of Improvement

One of the main goals of ours was to improve the design of our relational databases and to be able to do that we used a process well known in database development known as *Normalization*. The fundamental idea behind this approach is to divide up the larger tables into smaller ones and identify the linkages between those tables using rules that aim to protect the data and increase database flexibility by reducing redundant information and inconsistent dependencies.

How this can be done is by using *normal forms* and that is exactly what we did. The main aim of it is not only the reduction of redundancy but also the elimination of it completely. There are several levels of normalization, each with its own set of guidelines, but the ones used in this project are:

- First Normal Form **(1NF)**
- Second Normal Form **(2NF)**
- Third Normal Form **(3NF)**

A short idea of how we executed this process is that we firstly started by carefully examining each table to see if it complies with a specific normal form or not, if a table satisfies a particular normal form than we check if the relation satisfies the next normal higher form, in the event that a table fails to meet a specific normal form, steps are taken to transform it into a collection of tables that do.

The application of 1NF was carried out on non-relational tables to transform them into relational databases. As for the other normal forms (e.g., 2NF, 3NF) we used it to minimize the issue of update anomalies and enhance the design of relational databases that contained duplicated data.

The tables that we considered being in the **1NF** are the tables that fulfilled these rules:

- Each row is unique and no column in any row contains multiple values

The tables that we considered being in the **2NF** are the tables that fulfilled these rules:

- If it is in 1NF and if it does not contain partial functional dependencies

The tables that we considered being in the **3NF** are the tables that fulfilled these rules:

- If it is in 2NF and if it does not contain transitive functional dependencies

We should always keep in mind that after creating a database another important step is how we maintain it and always trying to find ways to improve its performance. Different issues can arise

and that is normal, but the true ability is not only to detect them and on time and fix them it is also to come back with an improved version of the database.

Recognising the existence of a problem is always thought of as the first step in resolving any performance issue. Continuous performance monitoring is the best approach to keep ahead of performance problems. By doing this, you can track changes in performance over time. When the rate of change gets faster, or when you reach a certain threshold, this might mean something has to be done.

Problems can occasionally persist between layers. These problems could be caused by hardware malfunctions, software updates, resource congestion, virus scanners, or network problems. Remedial work can begin as soon as the issue has been identified and linked to the apparent database performance problem.

As the size of the data set increases, a query becomes slower. How you make the database faster is obvious because you've measured from the beginning: you do a database refactor and eliminate the offending string concatenation in the query.

An additional feature that we implemented in our database for EIS is Clubs, Erasmus (together with the process of Applications) and being able to apply to different partner companies when a job is available for your profile..

2.3 Description Of The Specifics Of Improvements Made

Normalization process was a crucial element in the building of this database not only by improving the design part, but also by improving its performance. It helps in improving the reliability of the database ensuring that data is stored in a consistent, accurate, and secure manner.

The steps we followed in achieving the normalization of tables:

1. *Identify the entities*
2. *Creating tables for each entity containing all of the attributes*
3. *Identify the primary key*
4. *Define relationships*
5. *Normalize*

Student Table:

Identification (Personal Information):

- Epoka ID No. (Student ID) - ***Primary Key***
- Student Name
- Father Name
- Surname
- Title
- Gender
- Birthday (Year/Month/Day)
- Country
- City
- Citizenship
- Passport No.
- ID Card No.
- Primary Email (Epoka Email)
- Secondary Email
- Mobile Phone Number
- Marital Status
- Blood Group
- English level
- Program ID - ***Foreign Key***

Student Courses Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- CourseID - ***Foreign Key***
- Letter Grade
- Grade

Student Role Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Unit
- Position
- Project
- Role
- Start Date
- End Date
- Status *(Active, Inactive, Completed)*

Student Residence Permit Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Assigned Residence Permit
- Expiration Date
- Issuing Authority
- Permit Type
- Entry Date
- Exit Date
- Additional Details

Student Address Table:

- Epoka ID No. (Student ID) - ***Foreign Key***

- Street
- City
- Country
- Address Details

Student Family Details Table:

- Epoka ID No. (Student ID) - ***Foreign Key***

Father of Student:

- Father Name
- Surname
- Phone Number
- Work
- Profession

Mother of Student:

- Mother Name
- Surname
- Phone Number
- Work
- Profession

Faculty Table:

- Faculty Profile
- Faculty ID - ***Primary Key***

Program Table:

- Program Code
- Program ID - ***Primary Key***
- Program name
- Description
- Semester

- Nr. of Courses
- Course ID - ***Foreign Key***

Student Group (Details):

- Group Type (*Primary / Laboratory / [Seminar / Practice] Groups)*

Primary Groups:

- Group A (Epoka ID No. (Student ID) - last digit Odd)
- Group B (Epoka ID No. (Student ID)- last digit Even)

Laboratory Groups :

(Group A - Separated A1[A] and A2[C]) (Group B - Separated B1[B] and B2[D])

- Group A [A1] (First Half of Primary Group A)
- Group B [B1] (First Half of Primary Group B)
- Group C [A2] (Second Half of Primary Group A)
- Group D [B2] (Second Half of Primary Group B)

[Seminar / Practice] Groups :

- Group A (First Half of Primary Group A)
- Group B (Second Half of Primary Group B)
- Group C (Second Half of Primary Group A + First Half of Primary Group B)

Student Group Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- ProgramID ***- Foreign Key***
- Year of Study
- Student Group
- Hour Type

Department Table:

- Faculty ID - ***Foreign Key***
- DepartmentID - ***Primary Key***
- Department Name

- Department Description
- Nr. of Programs
- ProgramID - ***Foreign Key***

Courses Table:

- CourseID - ***Primary Key***
- Course Name
- Course Type (*Compulsory Courses / Non-Area Elective / General)*
- Lecturer No.
- ECTS per Course
- Credits per Course
- No. of Students
- Average Grades

Lecturer Table:

General Info (Identification):

- LecturerID - ***Primary Key***
- Lecturer Name
- Father Name
- Surname
- Title
- Gender
- Birthday (Year/Month/Day)
- Birthplace
- Country
- City
- Citizenship
- Passport No.
- ID Card No.
- Mobile Phone Number

- Home Phone
- Marital Status
- Blood Group

Lecturer Email Table:

- Primary Email (Epoka Email)
- Secondary Email
- LecturerID - ***Foreign Key***

Lecturer Role Table:

- LecturerID - ***Foreign Key***
- DepartmentID - ***Foreign Key***
- Position
- Role
- Start Date
- Status *(Active, Inactive, Completed)*

Lecturer Residence Permit Table:

- Assigned Residence Permit
- Expiration Date
- Issuing Authority
- Permit Type
- Entry Date
- Exit Date
- Additional Details
- Lecturer ID - ***Foreign Key***

Lecturer Address Table:

- Lecturer ID - ***Foreign Key***
- Street

- City
- Country
- Address Details

Teached Courses Table:

- LecturerID - ***Foreign Key***
- CourseID - ***Foreign Key***
- Semester Year
- Lecturer Role

Lecturer Office Table:

- LecturerID - ***Foreign Key***
- Office Phone
- Office Number

Discipline Table:

- Discipline Type - ***Primary Key***
- Discipline Description
- Time Period

Student Discipline Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Discipline Type - ***Foreign Key***
- Student Status
- Start Time
- End Time

Advisor Table:

- AdvisorID - ***Primary Key***
- ProgramID - ***Foreign Key***

- Year of Study
- No. of Students

Bank Info Table:

- Bank Name
- Account Name
- Account Number
- Account currency
- IBAN
- Swift Code
- Payment Description

Finance Table:

Financial information(tuition , fees, debts)

- Epoka ID No. (Student ID) - ***Foreign Key***
- Academic Year
- Debt
- Currency
- Total

Documents Table:

- Records
- Nr. Count (number of document)
- Epoka ID No. (Student ID) - ***Foreign Key***
- Document Type
- RequestID - ***Primary Key***
- Request Date
- Copy Requested
- Payment
- Price

- Document Status (when status is Ready you may receive the document)
- Actions

Grades Table:

- Letter Grade - ***Primary Key***
- GPA Value
- Points
- Grade Description
- Albanian Grade System
- Average Included

Academic Calendar Table:

- Event Name
- Period of Time
- Event Description

Transport Table:

- Route ID - ***Primary Key***
- Route (Tirana-Campus, Campus-Tirana, Durres-Campus, Campus-Durres)
- Departure Time
- Route Status (Has departed / Hasn't departed yet)
- Bus Stations
- Station Time

Transport Reservation:

- Epoka ID No. (Student ID) - ***Foreign Key***
- RouteID ***- Foreign Key***

Company Table:

- CompanyID - ***Primary Key***

- Company Name
- Company Description

Application Table:

- ApplicationID - ***Primary Key***
- Application Name
- CompanyID - ***Foreign Key***
- Date of Application

Professional Practice:

- Epoka ID No. (Student ID) - ***Foreign Key***
- ApplicationID - ***Foreign Key***
- CompanyID - ***Foreign Key***

Class Table:

- ClassID - ***Primary Key***
- Building
- Floor Level
- Class number
- Class Description (optional)

Erasmus Table:

- Erasmus code - ***Primary Key***
- University name
- State
- Study program
- Year of study
- Season
- Description

Erasmus Application Table:

- ApplicationID **-** ***Primary Key***
- Erasmus Code - ***Foreign Key***
- Epoka ID No. (Student ID) - ***Foreign Key***

Clubs Table:

- ClubID - ***Primary Key***
- Club name
- Club link
- No. of Students

Club Participants Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- ClubID - ***Foreign Key***
- Student role

Clubs Event Table:

- ClubID - ***Foreign Key***
- Event Name
- Event Date
- Start Time
- End Time

Timetable:

- Season
- Student Type
- ProgramID - ***Foreign Key***
- Program Year
- Year Group
- Day of the week

- CourseID - ***Foreign Key***
- Start Time
- End Time
- LecturerID - ***Foreign Key***
- ClassID - ***Foreign Key***

Personalized Timetable:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Day of the Week
- Course ID - ***Foreign Key***
- Start Time
- End Time
- LecturerID - ***Foreign Key***
- ClassID - ***Foreign Key***

Course Selection Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Course Type
- Course ID - ***Foreign Key***
- Advisor ID - ***Foreign Key***
- Selected
- Confirmation Date

Attendance Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Records
- CourseID - ***Foreign Key***
- Nr. of the Week
- Topic
- Hour Type

- Attended Hours
- Total Hours
- Course Hours Date

Survey Table:

- ProgramID **-** ***Foreign Key***
- Epoka ID No. (Student ID) - ***Foreign Key***
- CourseID - ***Foreign Key***
- LecturerID - ***Foreign Key***
- Question
- Evaluation
- Additional Review

Resit Exam Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Course ID - ***Foreign Key***
- Letter Grade - ***Foreign Key***
- Select Status

Additional Exams Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Academic Year
- Course ID - ***Foreign Key***
- Additional Date
- Exam Status
- Actions

3rd Semester Table:

- Epoka ID No. (Student ID) - ***Foreign Key***
- Course ID - ***Foreign Key***

- Course Status

Selected Courses Table:

- Course Code - ***Foreign Key***
- Approval

Epoka ID No. (Student ID) (for description purposes)::

A unique number created by:

- Faculty
- Program of study
- Year Of Registration
- No. Of Registration

LogIn Table (for description purposes):

- Epoka email
- First letter of name
- Last name
- Year of registration
- Password (hash codes)

Relationships:

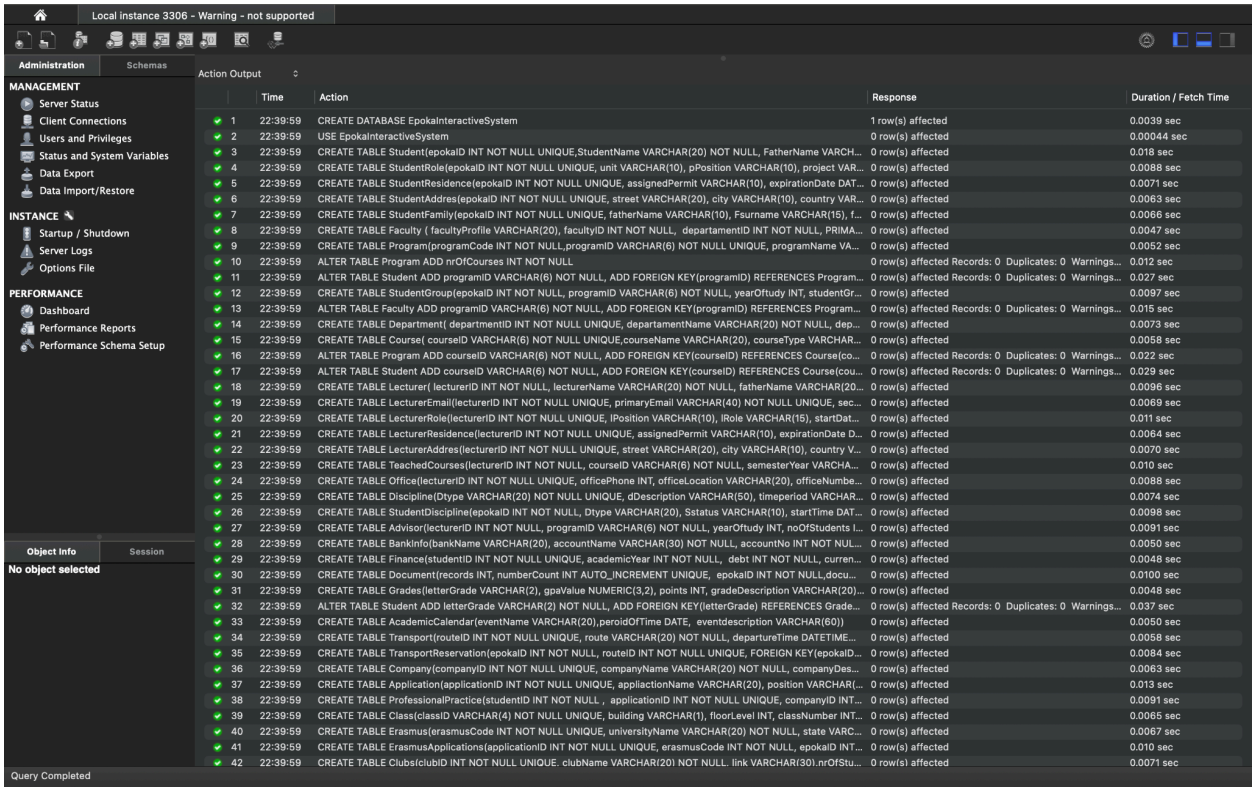
- Epoka has 3 faculties.
- Each faculty has at least one departament.
- A departament has many programs.
- A program has at least 4 semesters.
- A semester has many mandatory courses.
- A course is to be selected by many students.
- A student takes many courses.

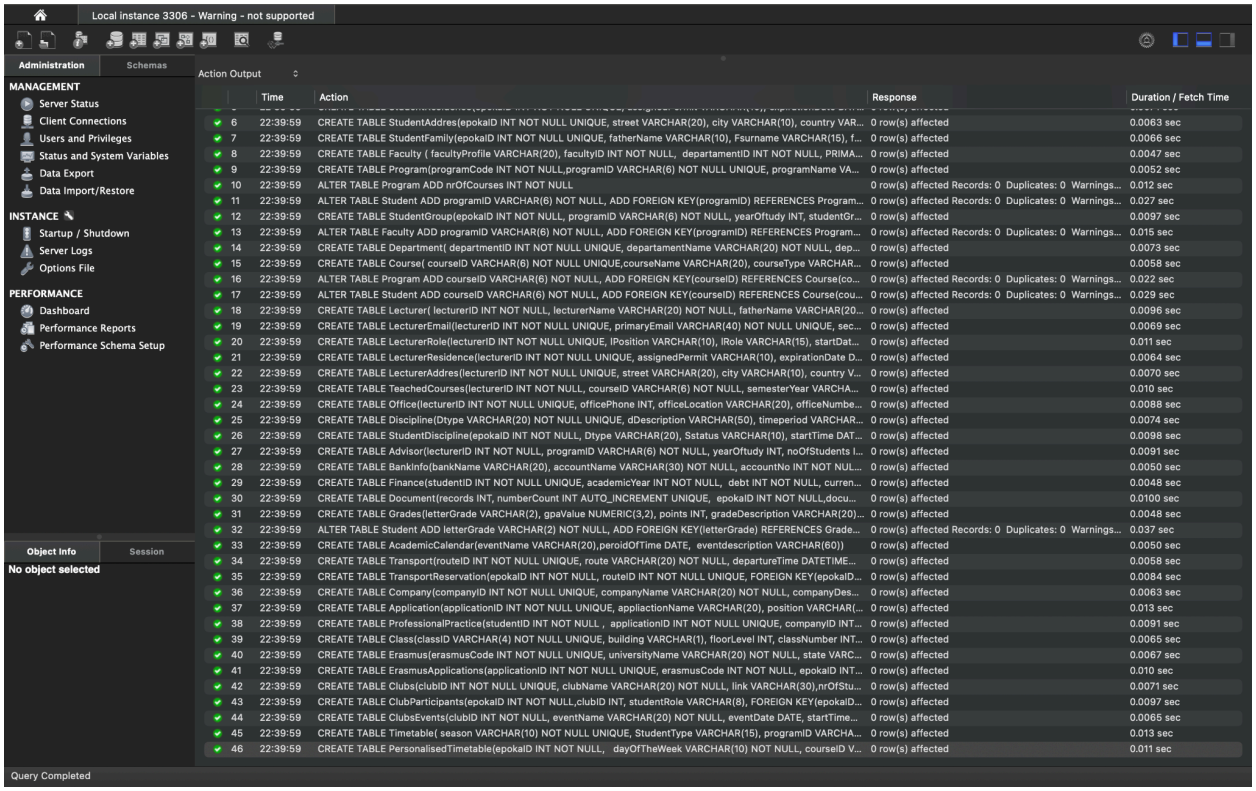
- A course is given by at least one lecturer.
- A lecturer can give one or many courses.
- A lecturer can give lectures in more than one faculty.
- A lecturer has many students.
- A student has one email and password to log in.
- A lecturer has one email and password to log in.
- One student must take at least 5 courses per semester. ***(FAE - Faculty Architecture & Engineering)***
- A course can be held in one classroom at a time.
- A student can only have one final letter grade for a course.
- A student is at least in one group.
- 1 group has many students.
- Many students have one advisor.
- One advisor is responsible for only one year of a specific program.
- Epoka offers many documents.
- A student can request one or many documents.
- Epoka has exactly 2 bank accounts.
- Each student pays only one tuition fee.
- One student has exactly one transcript.
- 1 student can make none or many professional practice applications.
- Many companies offer one or many job positions.
- In one company none or many applications can be made.
- Epoka has one academic calendar per year.
- Erasmus has at least one university option.
- Erasmus has at least one state option.
- One state has one or many university options.
- 1 university offers at least one program.
- 1 club can have many students.
- Each student can be registered to none or many clubs.
- 1 club has at least one event.
- Transport has exactly 4 routes.

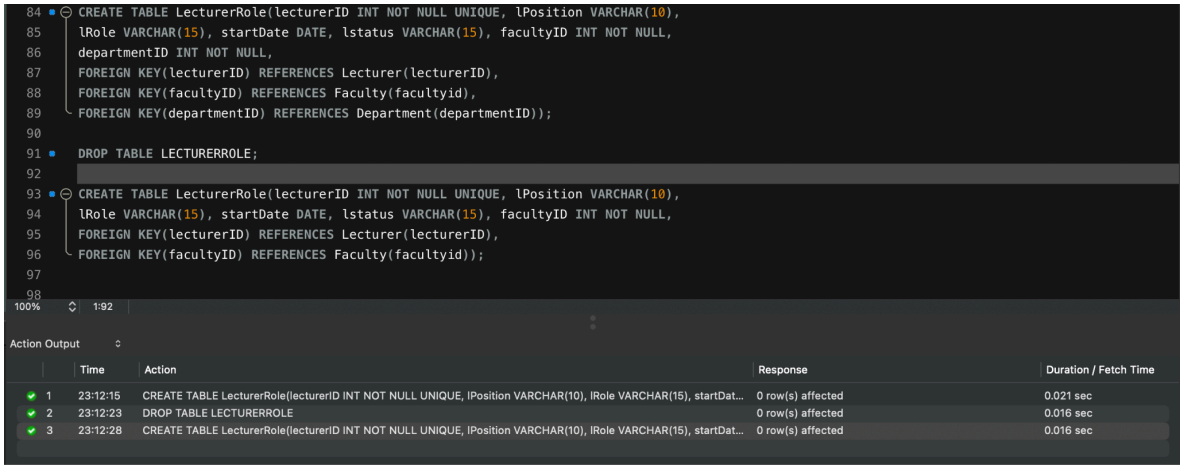
- 1 route is made at least one time per day.
- 1 route has at least one station.
- 1 route can be taken by many students or lecturers.

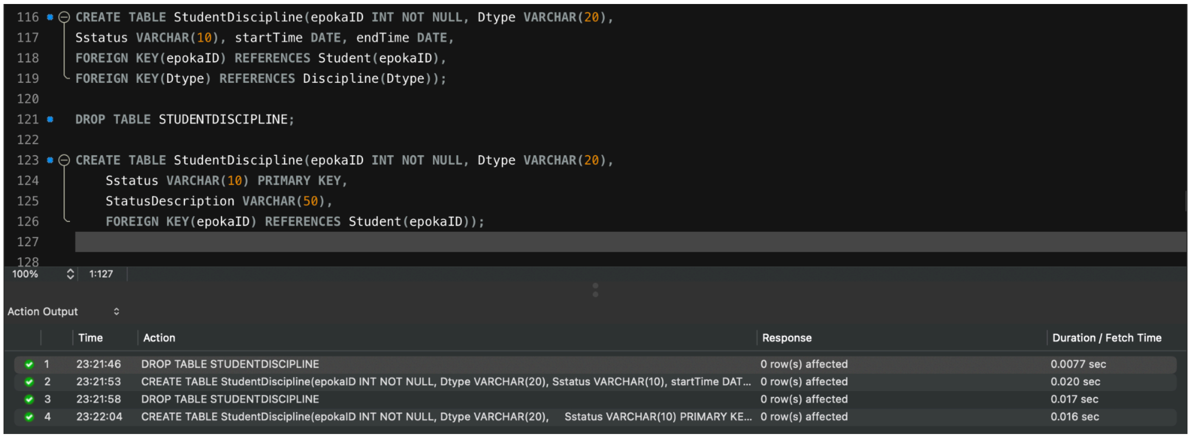
2.4 Concrete Measurement Description

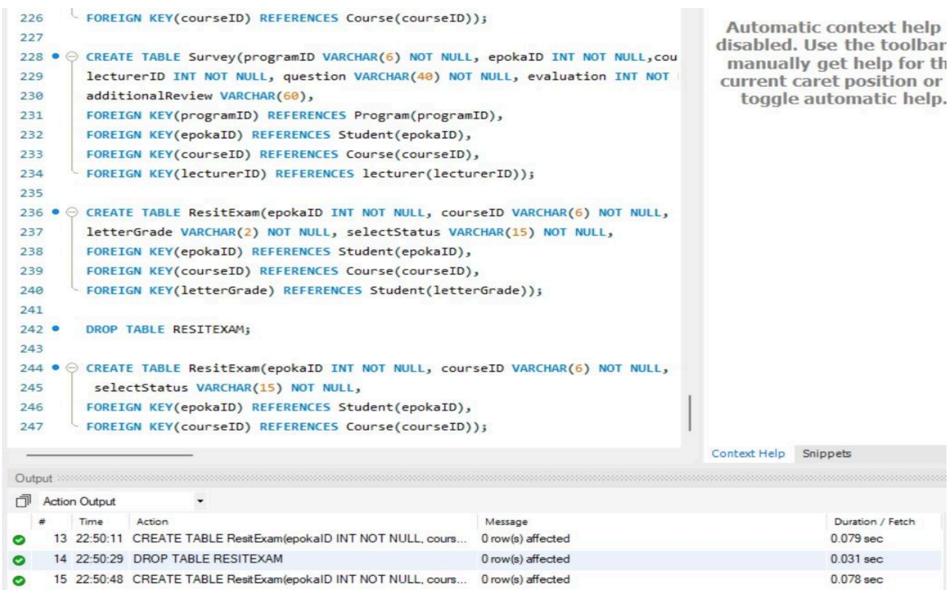
While the SQL files and queries were being created we took concrete measurements of the query speed and tried different ways to improve them. We did this by using short concentrated tables containing a basic concept explaining it and not long overloaded tables with unnecessary repeated and spread information. The usage of primary keys, unique values and not allowing empty spaces lead to an improved performance.




Local instance 3306 - Warning - not supported
Administration
Schemas
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Object Info
Session
No object selected
Action Output
Time
Action
Response
Duration / Fetch Time
Query Completed


CREATE TABLE LecturerRole(lecturerID INT NOT NULL UNIQUE, lPosition VARCHAR(10),
lRole VARCHAR(15), startDate DATE, lstatus VARCHAR(15), facultyID INT NOT NULL,
departmentID INT NOT NULL,
FOREIGN KEY(lecturerID) REFERENCES Lecturer(lecturerID),
FOREIGN KEY(facultyID) REFERENCES Faculty(facultyid),
FOREIGN KEY(departmentID) REFERENCES Department(departmentID));
DROP TABLE LECTURERROLE;
CREATE TABLE LecturerRole(lecturerID INT NOT NULL UNIQUE, lPosition VARCHAR(10),
lRole VARCHAR(15), startDate DATE, lstatus VARCHAR(15), facultyID INT NOT NULL,
FOREIGN KEY(lecturerID) REFERENCES Lecturer(lecturerID),
FOREIGN KEY(facultyID) REFERENCES Faculty(facultyid));
100%
1:92
Action Output
Time
Action
Response
Duration / Fetch Time
1 23:12:15 CREATE TABLE LecturerRole(lecturerID INT NOT NULL UNIQUE, lPosition VARCHAR(10), lRole VARCHAR(15), startDat... 0 row(s) affected 0.021 sec
2 23:12:23 DROP TABLE LECTURERROLE 0 row(s) affected 0.016 sec
3 23:12:28 CREATE TABLE LecturerRole(lecturerID INT NOT NULL UNIQUE, lPosition VARCHAR(10), lRole VARCHAR(15), startDat... 0 row(s) affected 0.016 sec


CREATE TABLE StudentDiscipline(epokaID INT NOT NULL, Dtype VARCHAR(20),
Sstatus VARCHAR(10), startTime DATE, endTime DATE,
FOREIGN KEY(epokaID) REFERENCES Student(epokaID),
FOREIGN KEY(Dtype) REFERENCES Discipline(Dtype));
DROP TABLE STUDENTDISCIPLINE;
CREATE TABLE StudentDiscipline(epokaID INT NOT NULL, Dtype VARCHAR(20),
Sstatus VARCHAR(10) PRIMARY KEY,
StatusDescription VARCHAR(50),
FOREIGN KEY(epokaID) REFERENCES Student(epokaID));
Action Output
Time
Action
Response
Duration / Fetch Time
1 23:21:46 DROP TABLE STUDENTDISCIPLINE 0 row(s) affected 0.0077 sec
2 23:21:53 CREATE TABLE StudentDiscipline(epokaID INT NOT NULL, Dtype VARCHAR(20), Sstatus VARCHAR(10), startTime DAT... 0 row(s) affected 0.020 sec
3 23:21:58 DROP TABLE STUDENTDISCIPLINE 0 row(s) affected 0.017 sec
4 23:22:04 CREATE TABLE StudentDiscipline(epokaID INT NOT NULL, Dtype VARCHAR(20), Sstatus VARCHAR(10) PRIMARY KE... 0 row(s) affected 0.016 sec


FOREIGN KEY(courseID) REFERENCES Course(courseID));
CREATE TABLE Survey(programID VARCHAR(6) NOT NULL, epokaID INT NOT NULL,cou
lecturerID INT NOT NULL, question VARCHAR(40) NOT NULL, evaluation INT NOT
additionalReview VARCHAR(60),
FOREIGN KEY(programID) REFERENCES Program(programID),
FOREIGN KEY(epokaID) REFERENCES Student(epokaID),
FOREIGN KEY(courseID) REFERENCES Course(courseID),
FOREIGN KEY(lecturerID) REFERENCES lecturer(lecturerID));
CREATE TABLE ResitExam(epokaID INT NOT NULL, courseID VARCHAR(6) NOT NULL,
letterGrade VARCHAR(2) NOT NULL, selectStatus VARCHAR(15) NOT NULL,
FOREIGN KEY(epokaID) REFERENCES Student(epokaID),
FOREIGN KEY(courseID) REFERENCES Course(courseID),
FOREIGN KEY(letterGrade) REFERENCES Student(letterGrade));
DROP TABLE RESITEXAM;
CREATE TABLE ResitExam(epokaID INT NOT NULL, courseID VARCHAR(6) NOT NULL,
selectStatus VARCHAR(15) NOT NULL,
FOREIGN KEY(epokaID) REFERENCES Student(epokaID),
FOREIGN KEY(courseID) REFERENCES Course(courseID));
Automatic context help disabled. Use the toolbar manually get help for th current caret position or toggle automatic help.
Context Help
Snippets
Output
Action Output
13 22:50:11 CREATE TABLE ResitExam(epokaID INT NOT NULL, cours... 0 row(s) affected 0.079 sec
14 22:50:29 DROP TABLE RESITEXAM 0 row(s) affected 0.031 sec
15 22:50:48 CREATE TABLE ResitExam(epokaID INT NOT NULL, cours... 0 row(s) affected 0.078 sec

Project Execution

All of the diagrams presented in this document have been worked on ERDPlus.

3.1 ER Diagram Solution

Entity - Relationship is a conceptual data model that explains how items relate to one another in a specific field. A database's structure is generally described in detail by the ER model.

The ER Model consists of three main components:

- ***Entities:*** Objects / Ideas are represented by the entities and in a database each table is associated with an entity.
- ***Attributes:*** The collection of qualities that characterize an entity and in the database they correspond to the columns of the table. They can be of different types: unique, composite, derived, optional.
- ***Relationships:*** Represent how the relationships between different entities and map foreign key associations in a database. There are different types of relationships such as: one to many, many to many and one to one. They can be either mandatory or optional.

It is very helpful when you want to achieve a well built project and is considered as a guidance in the implementation of a database.

The way we executed this requirement in our project was step-by-step:

- ***Step 1:*** After having identified all the entities that make up EIS, we started by making individual diagrams with one entity and all the attributes that described it.
- ***Step 2:*** After the first step we started by writing all the relationships (connections) to form a full ER diagram. The reason we did it this way is so the ER diagrams wouldn't be overly populated and the relationship could be clearly visible.

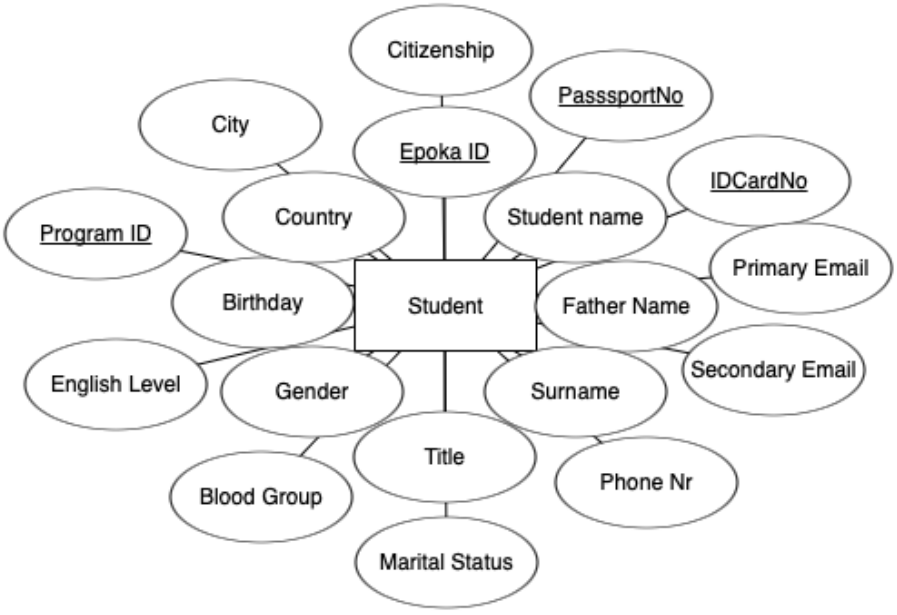
An ER model is an abstract description of a database structure, while an ER diagram is a visual representation of an ER model.

As previously mentioned the ER Diagram is build through relationship which are of type:

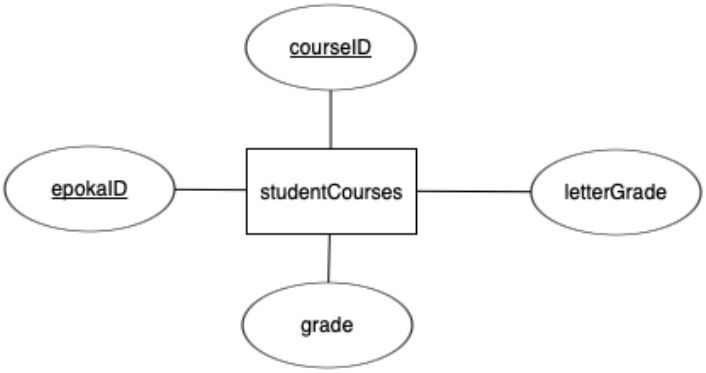
- Optional Many
- Mandatory Many
- Optional One
- Mandatory One

ER Models

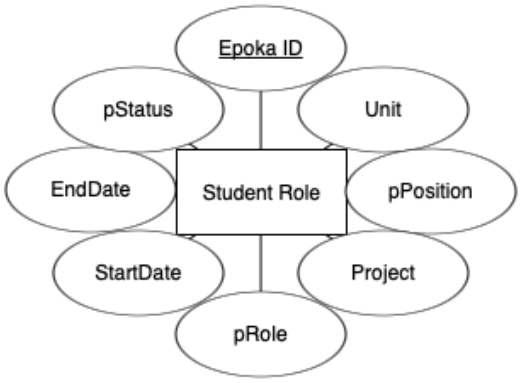
1. Student:



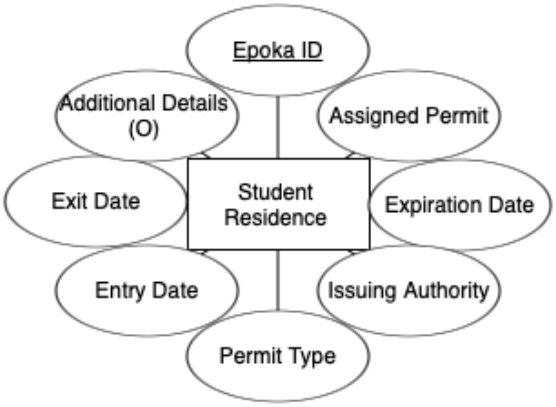
2. Student Courses:



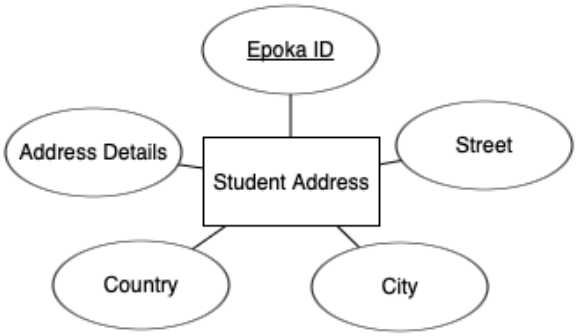
3. Student Role:



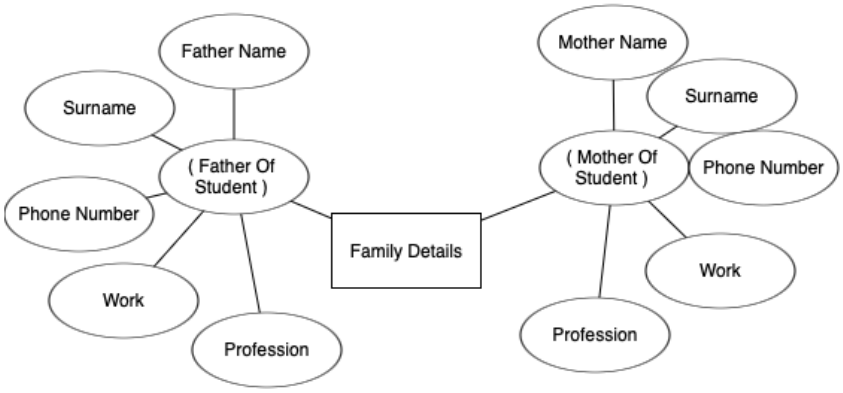
4. Student Residence Permit:



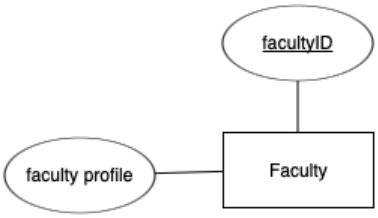
5. Student Address:



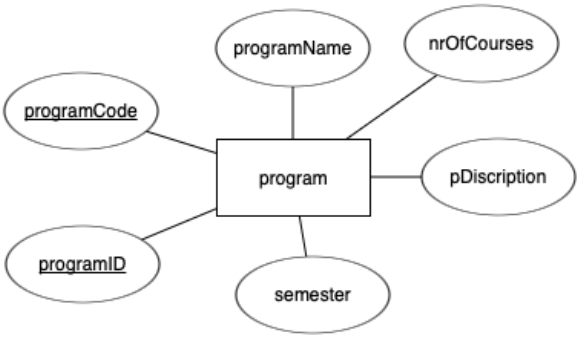
6. Student Family Details:



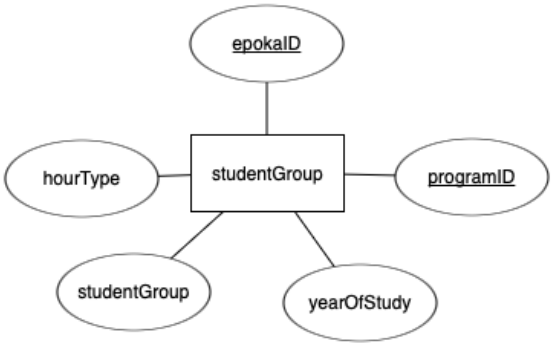
7. Faculty:



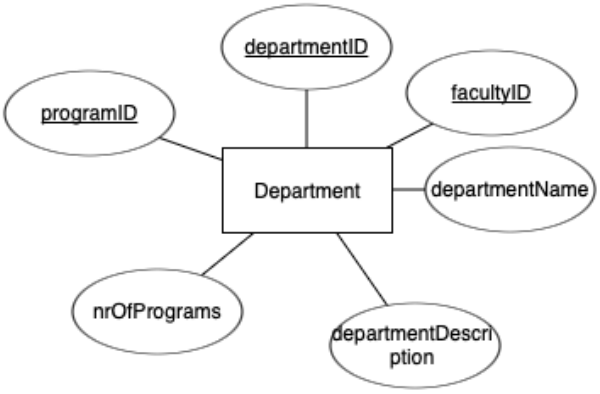
8. Program:



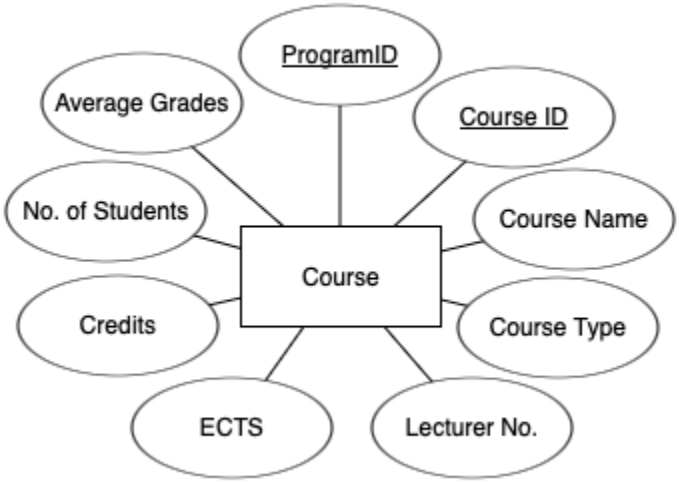
9. Student Group:



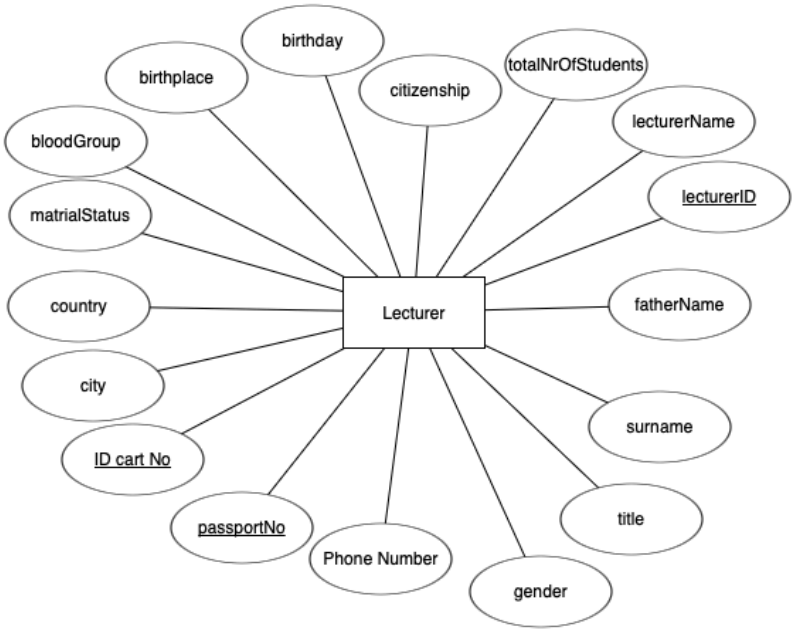
10. Department:



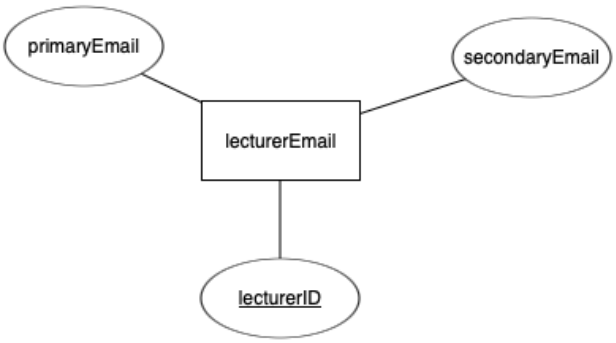
11. Course:



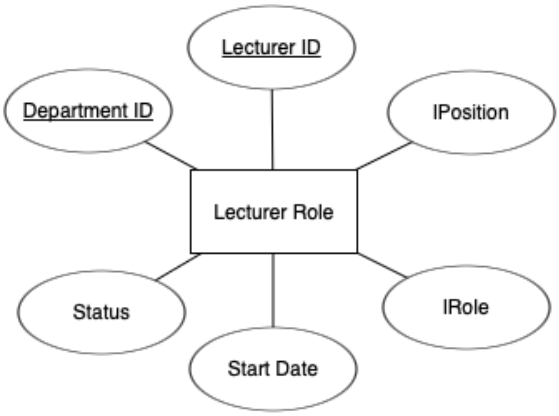
12. Lecturer:



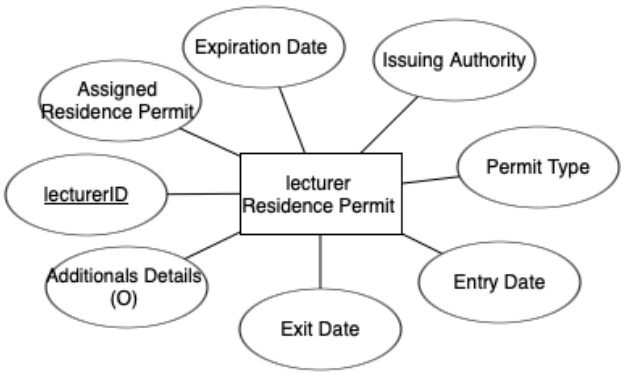
13. Lecturer Email:



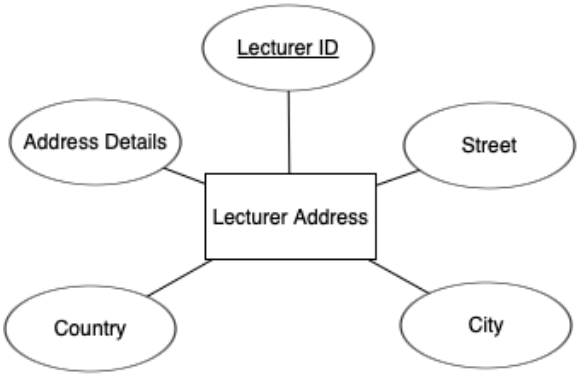
14. Lecturer Role:



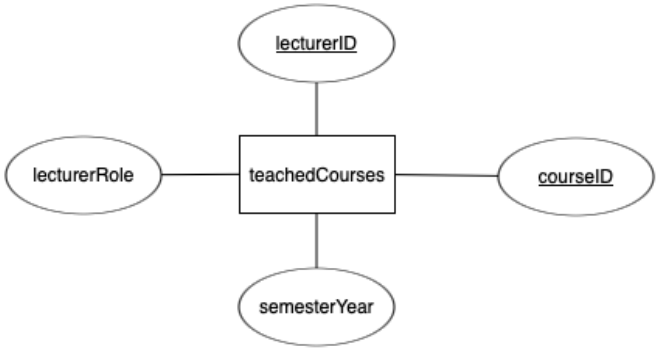
15. Lecturer Residence Permit:



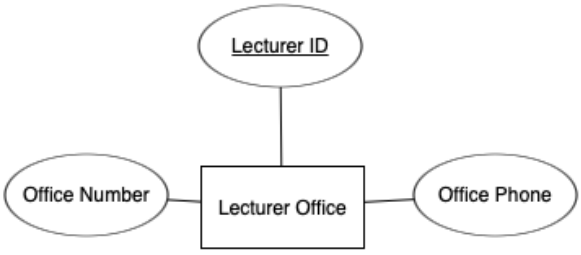
16. Lecturer Address:



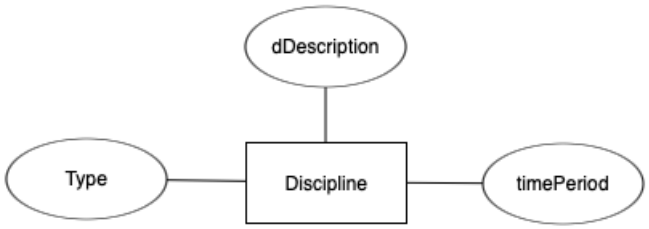
17. Teached Courses:



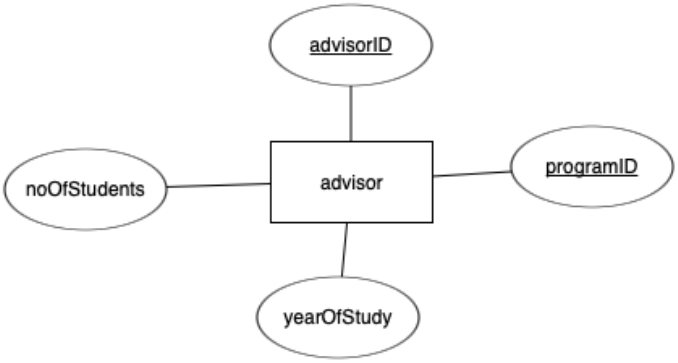
18. Lecturer Office:



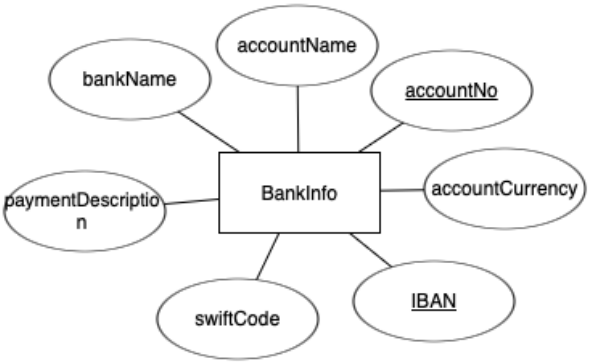
19. Discipline:



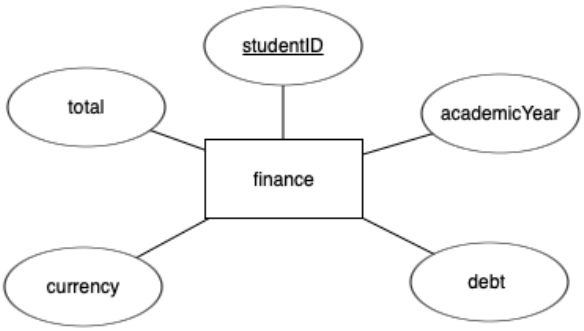
20. Advisor:



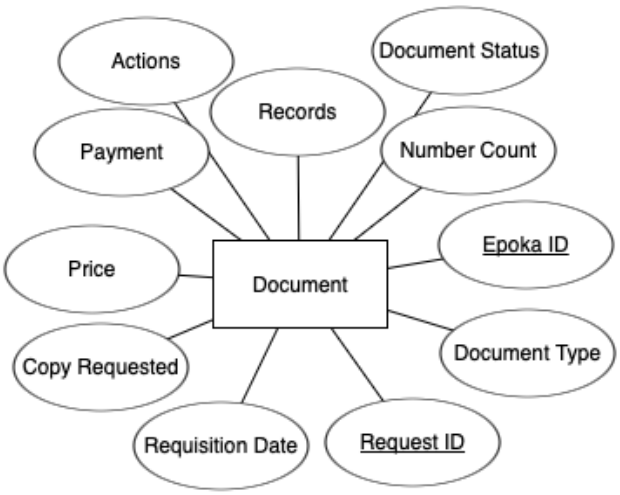
21. Bank Info:



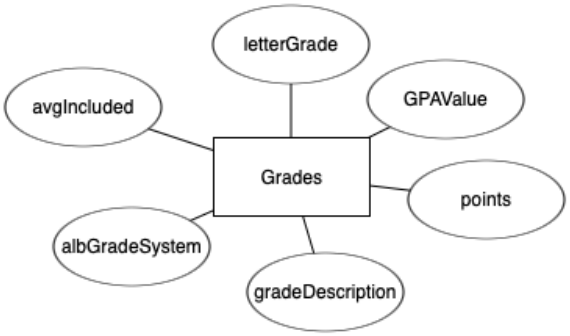
22. Finance:



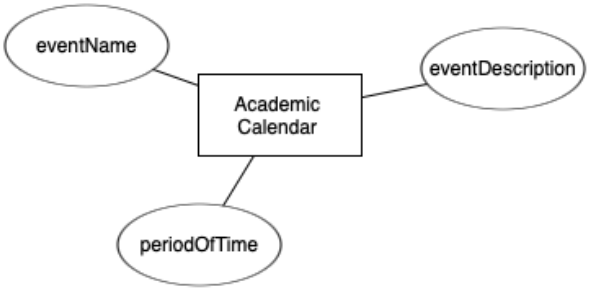
23. Document Request:



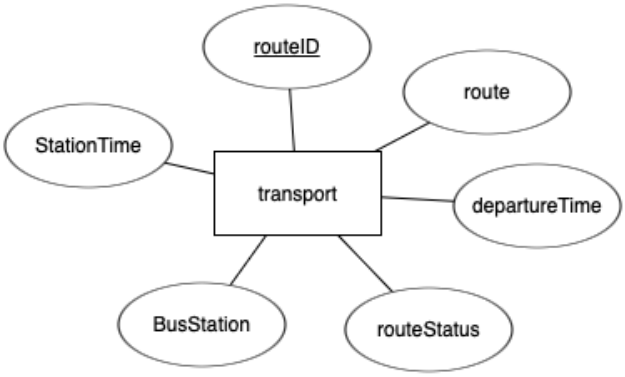
24. Grades:



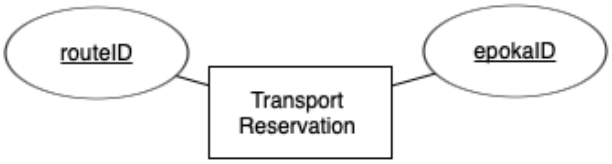
25. Academic Calendar:



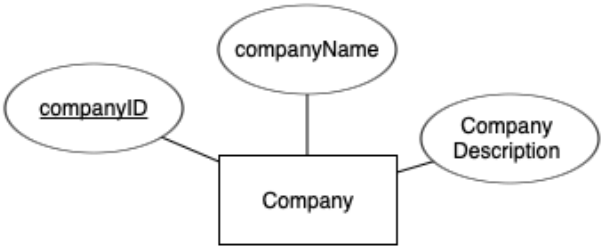
26. Transport:



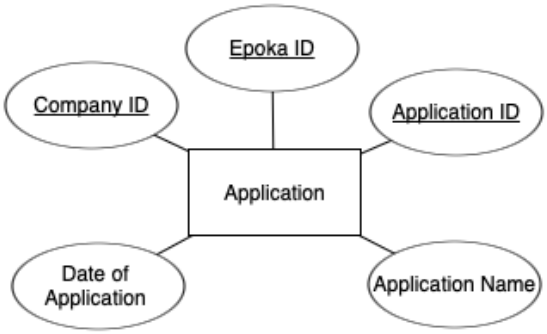
27. Transport Reservation:



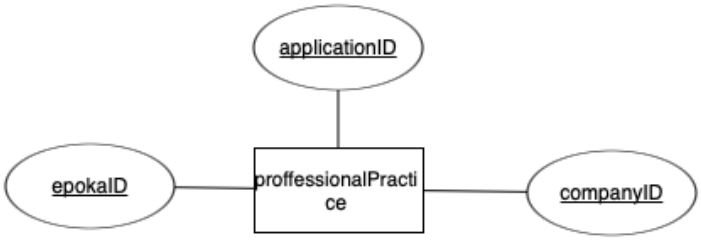
28. Company:



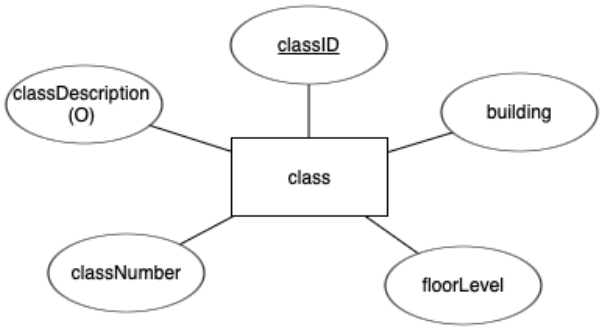
29. Application:



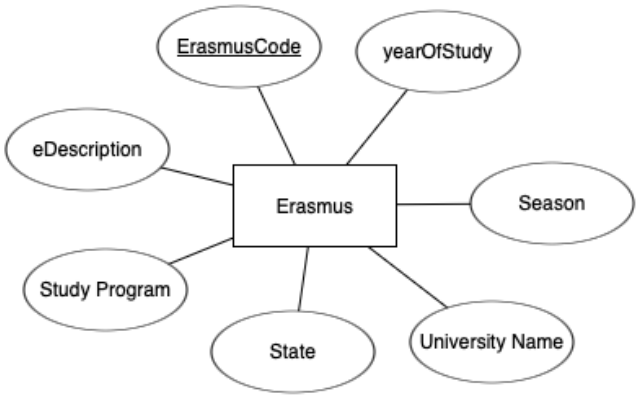
30. Professional Practice:



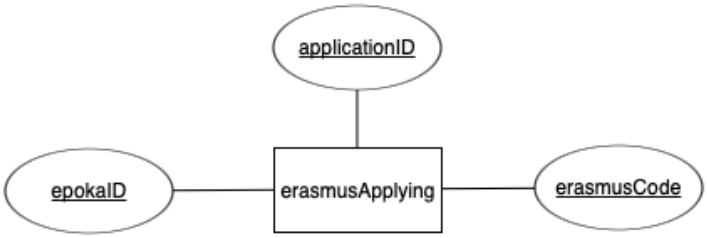
31. Class:



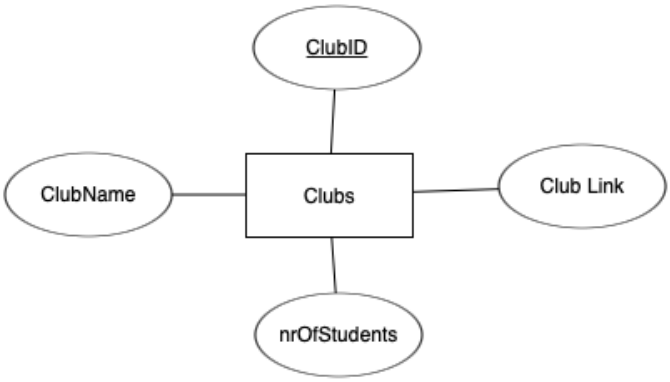
32. Erasmus:



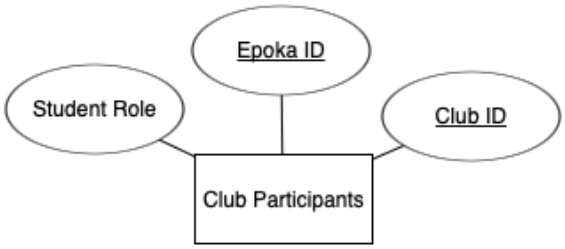
33. Erasmus Application:



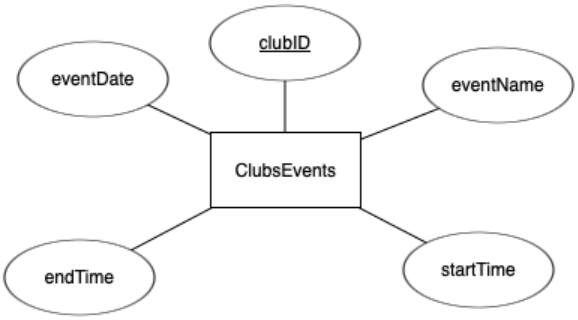
34. Club:



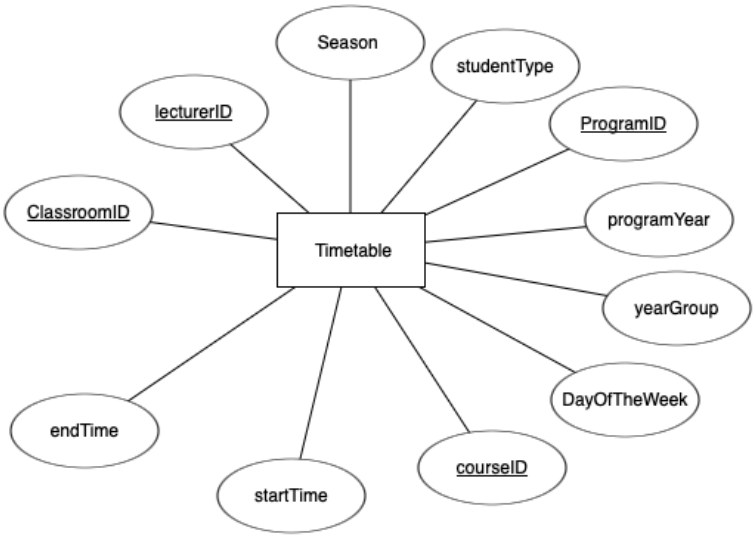
35. Club Participants:



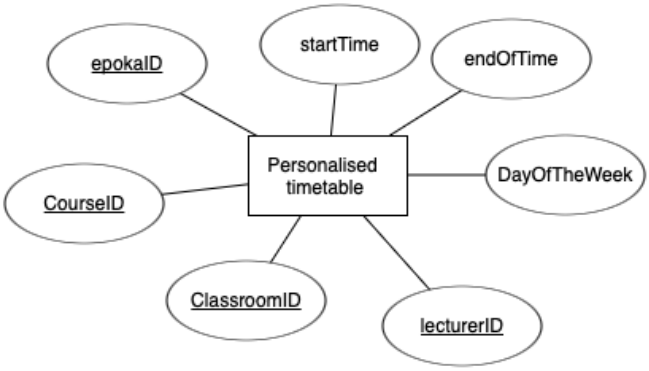
36. Club Event:



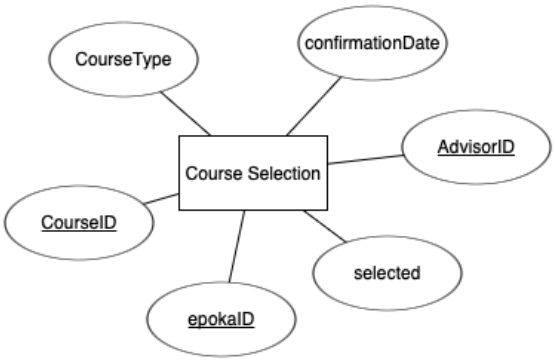
37. Timetable:



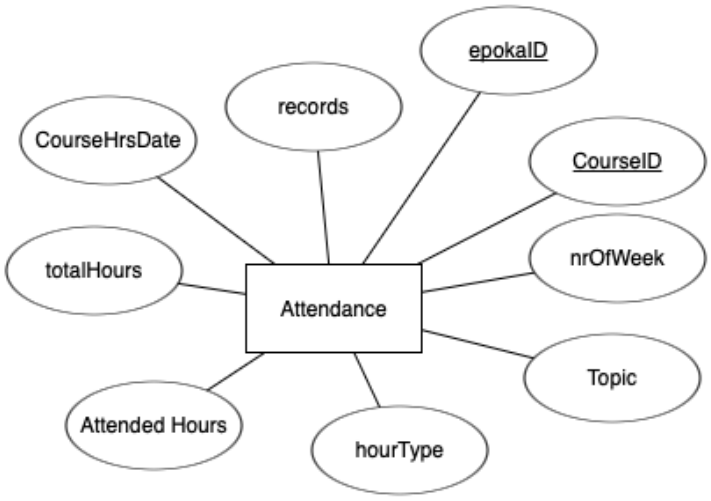
38. Personalized Timetable:



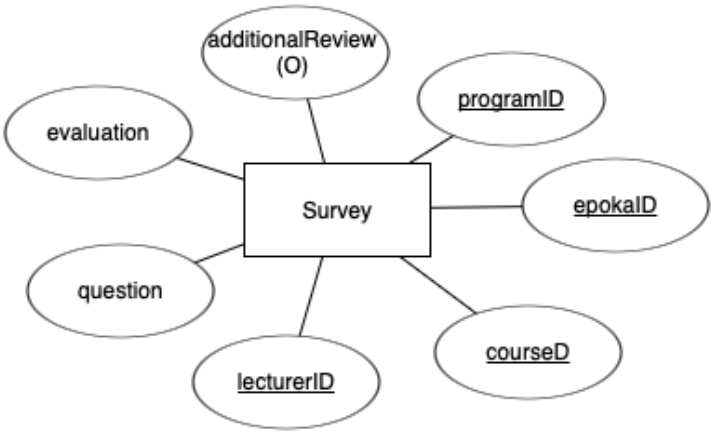
39. Course Selection:



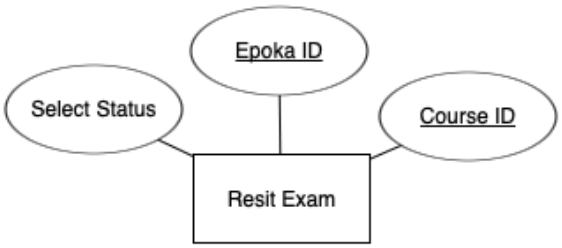
40. Attendance:



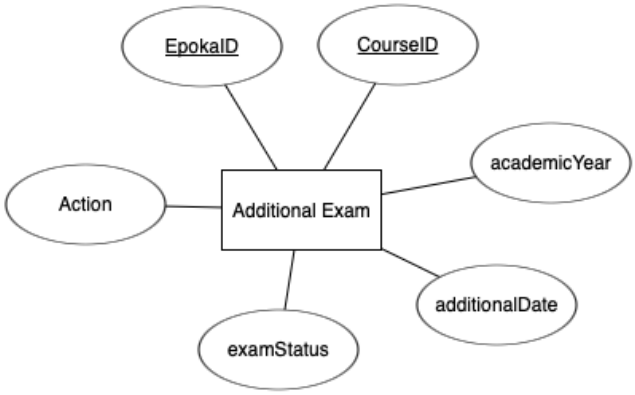
41. Survey:



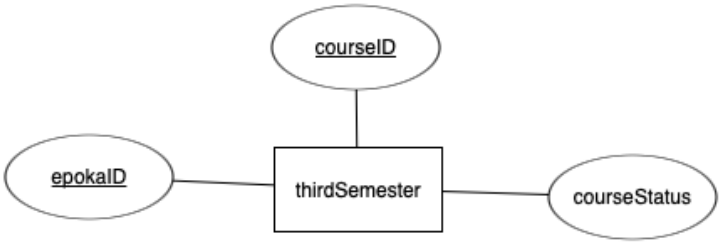
42. Resit Exam:



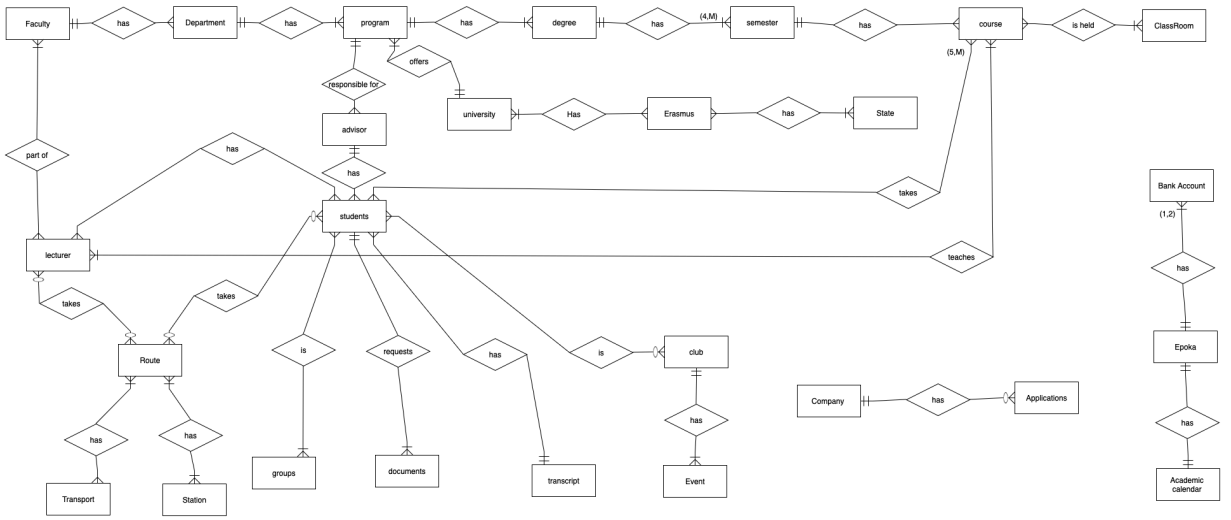
43. Additional Exams:



44. 3rd Semester:



ER Diagrams



3.2 RS Schema

A relational schema is a type of data model for logically representing the database as a collection of related tables. Each named relation in a relational database definition has its own set of attributes and instances of entities.

An attribute can be considered as a key as well:

- ***Primary Key:*** Column (or a set of columns) whose value is unique for each row.
  - We can also have <u>composite</u> primary keys.
- ***Foreign Key:*** Column in a relation that refers to a primary key column in another (referred) relation

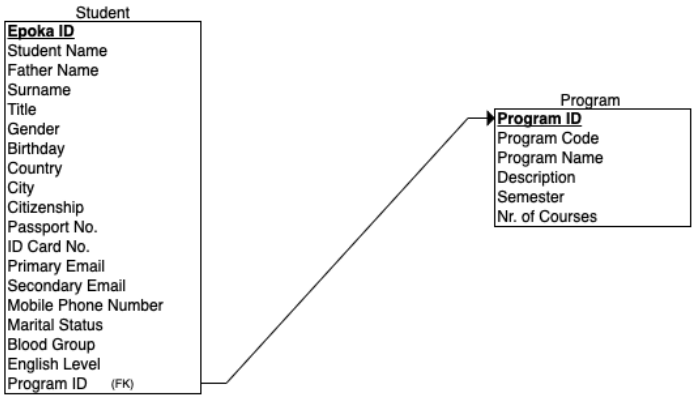
Comparison between ER Diagram and Relational Schema

The structure of a database is represented by both the relational schema and the ER model. Nonetheless, their goals and degrees of abstraction differ. While the relational schema is a logical model to build a specific database schema that can be implemented in a particular DBMS, the ER Model is a conceptual model used to design a database.

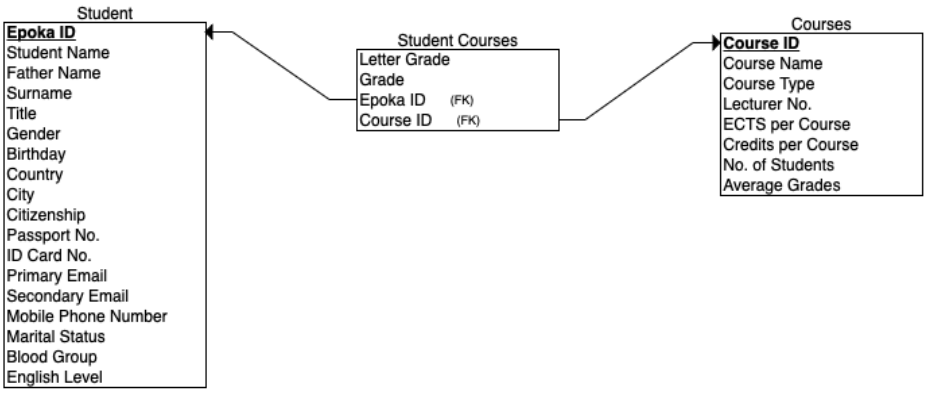
RS Schemas (with short explanations)

While creating the Relational Schemas we tried to focus on showing the new connections formed

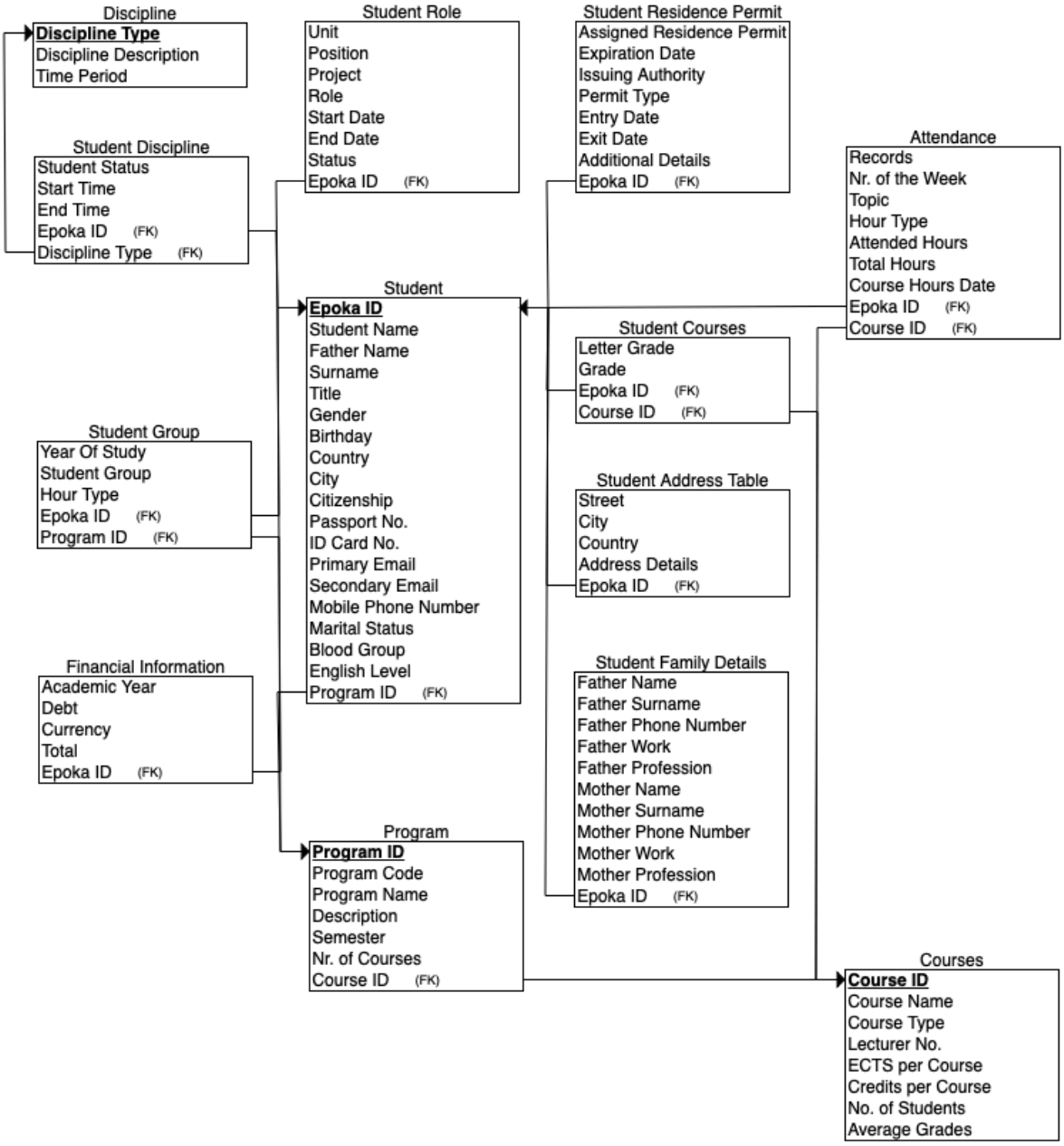
as well as paying close attention to primary keys and foreign keys, rather than repeating the same information on previously mentioned tables.



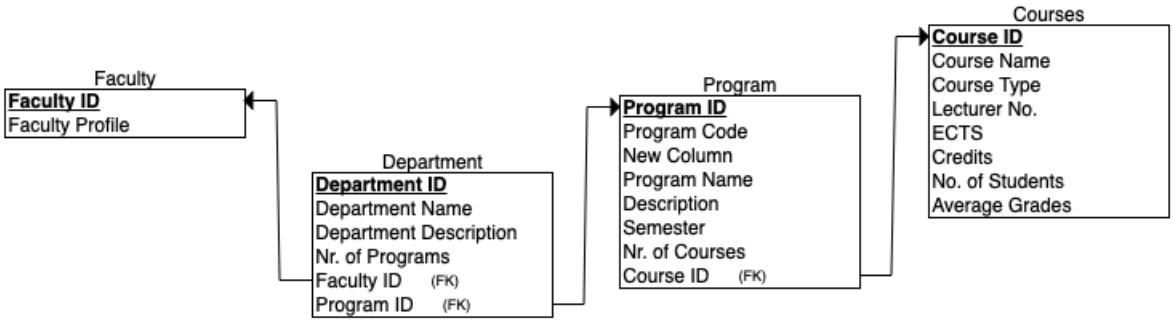
The first Relational Schema presents the *Student Table* which has the Epoka ID marked as its primary key connected with the *Program Table* that has Program ID marked as its primary key. Through this relation we can see that the primary key of the Program Table passes as a foreign key to the Student table.



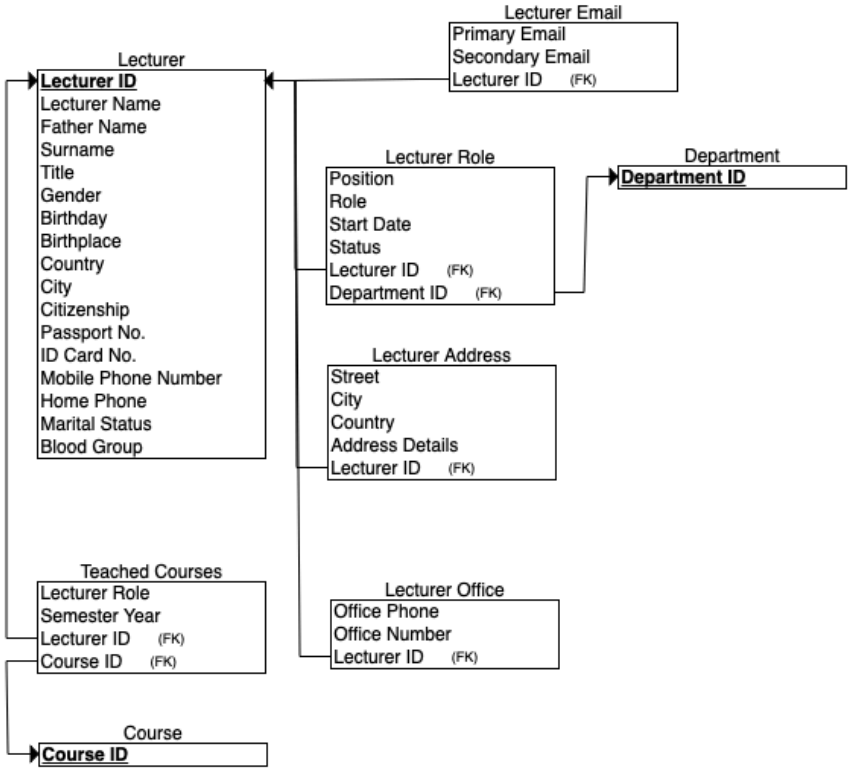
The second schema presents a new table named *Courses Table* with a primary key named Course ID, the table *Student* from the previous and the table *Student Courses* which doesn't have a primary key of its own, but has two foreign keys taken from the two other tables as shown above.



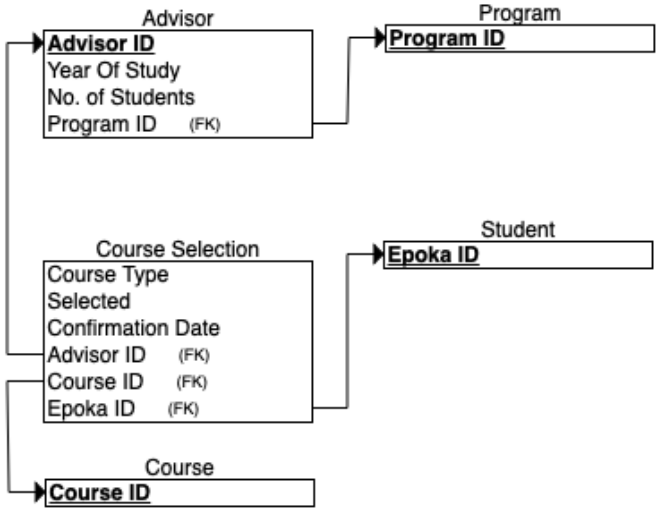
The third schema is a longer one and made up of more components all related to the *Student Table*.



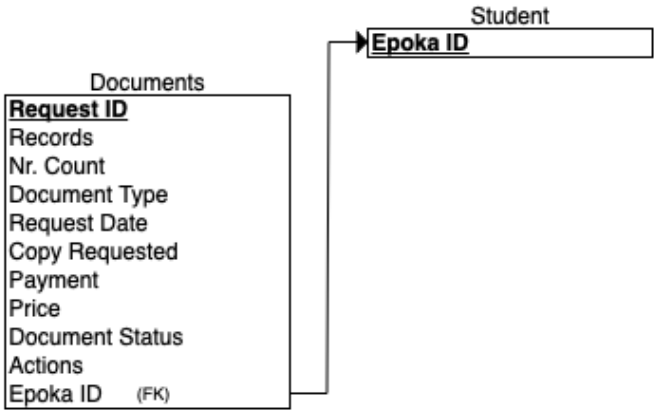
The fourth schema shows a breakdown of the main components that make up a university, together with the way that they relate to one another through primary and foreign keys.



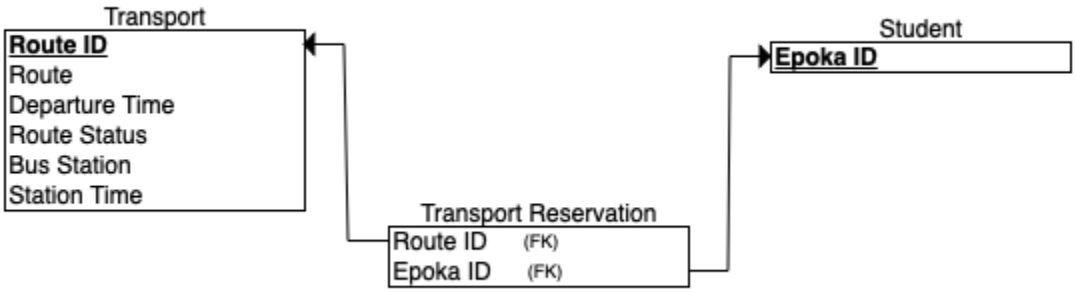
Until now we have seen the student part that makes up EIS, this schema is the representation of the primary components of the lecturer to further expand.



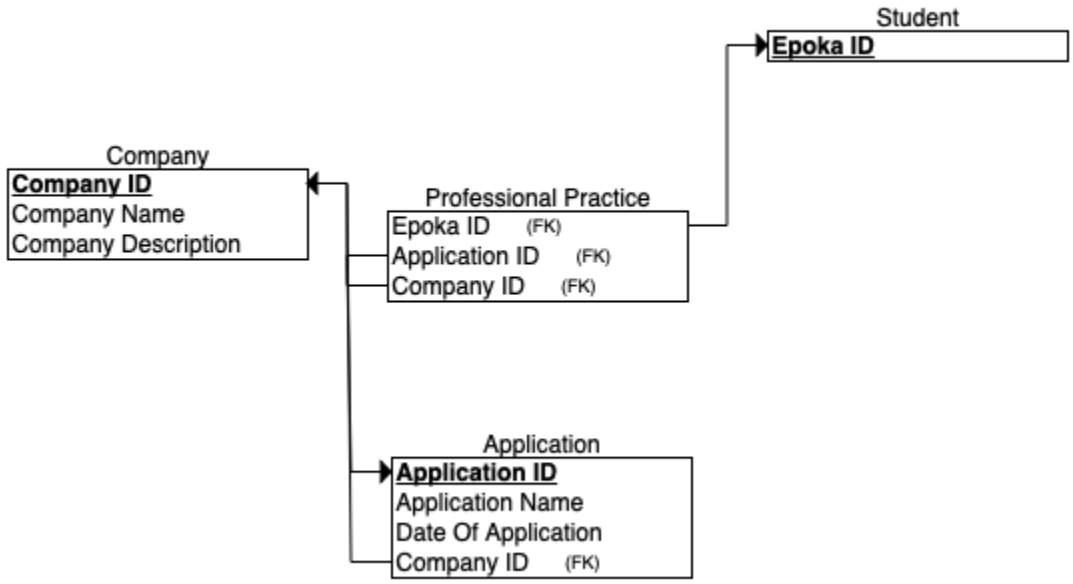
This schema shows the relation between the Advisor (who firstly needs to be a Lecturer) and the Students. What this means is that one group of one year of one program can have one Advisor, another role associated with it is the approval of courses chosen by students in this program.



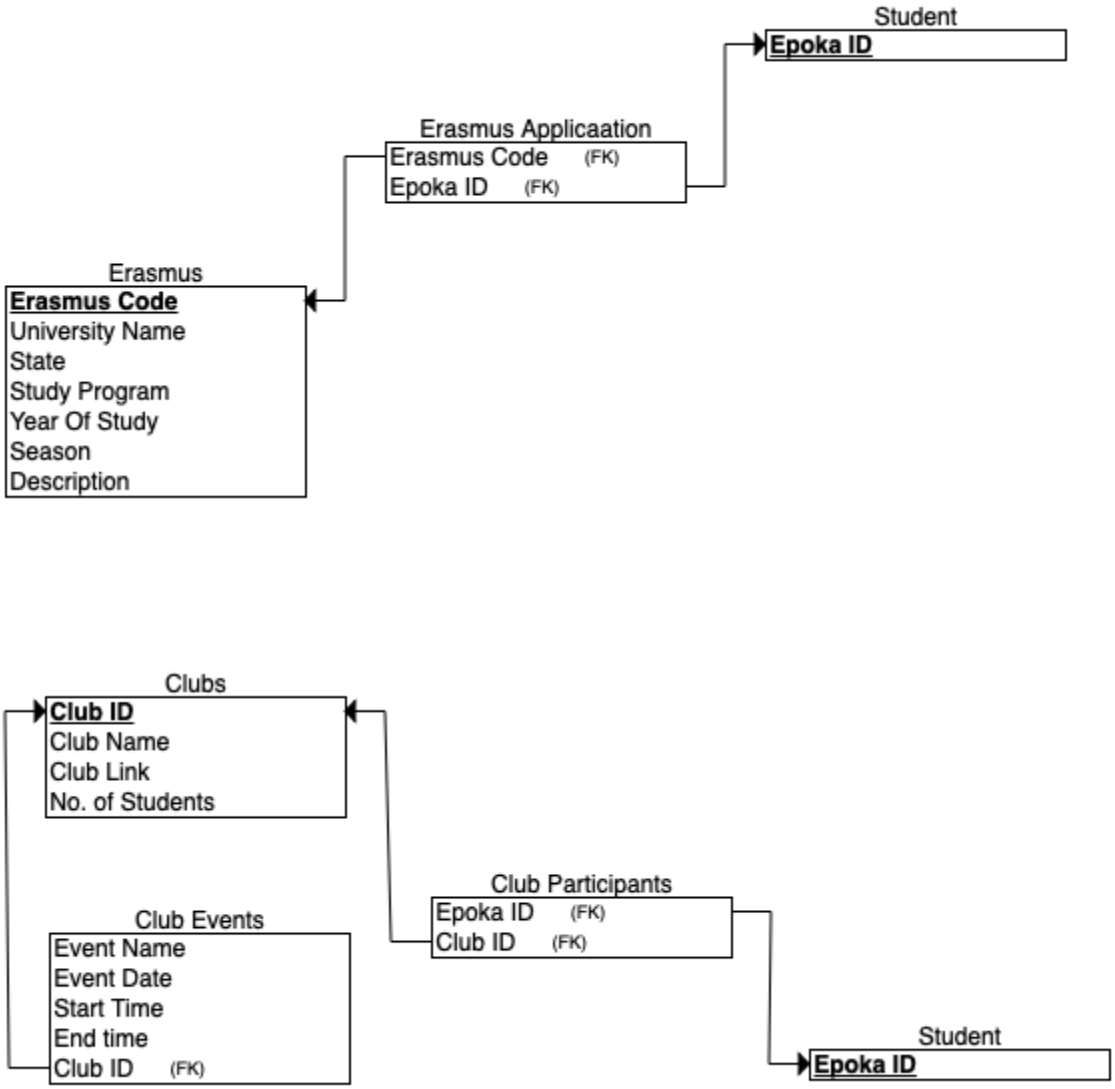
This schema represents a service that can be offered to all Students.



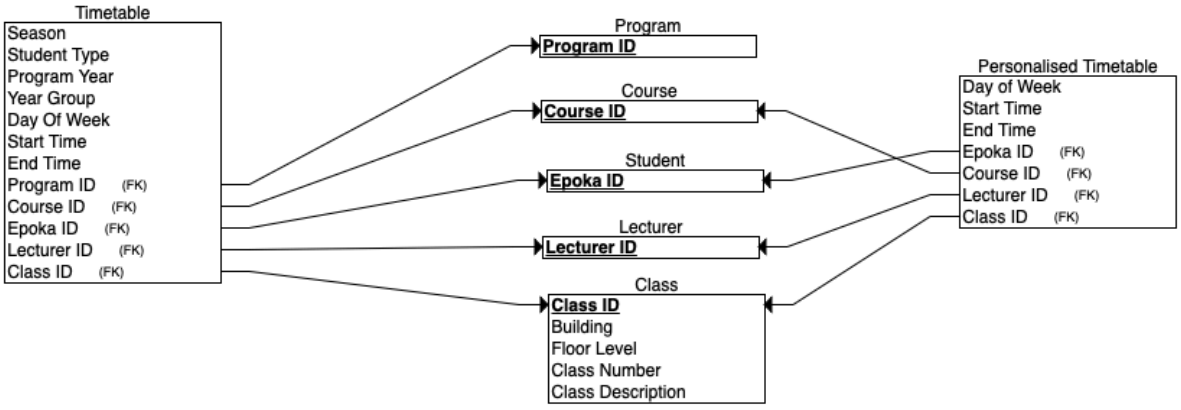
Here we have represented the possibility of the *reservation* of your seat in one of the desired routes that Epoka University offers.



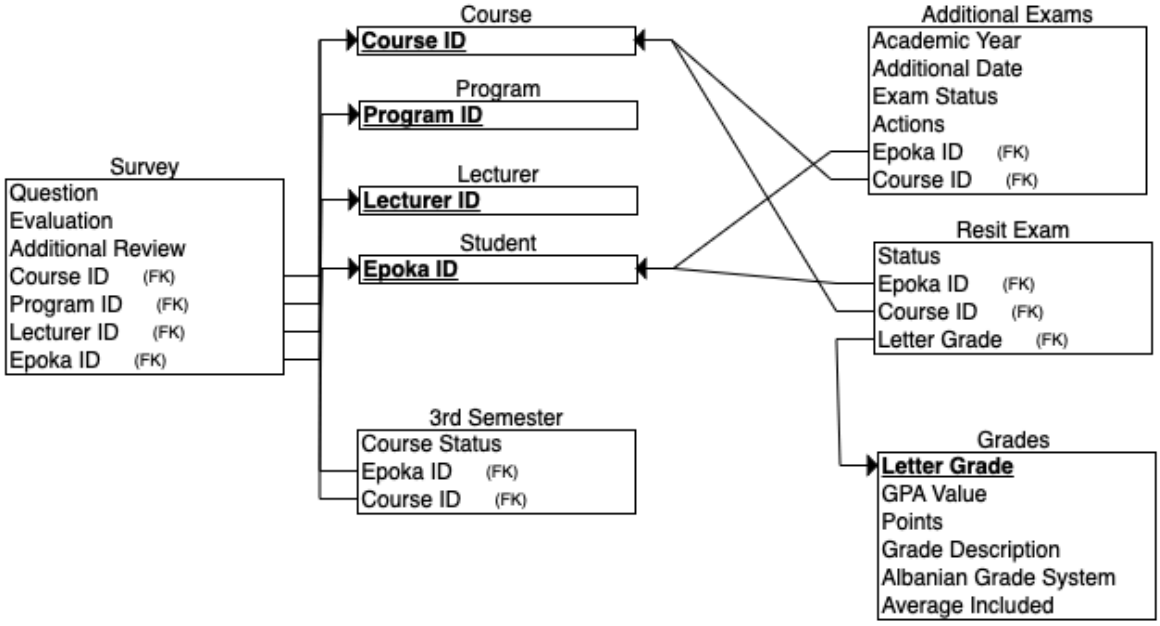
As a third year student you are obligated to take the course *Professional Practice* and to make the *Application* easier to the students a list of *companies* is presented to them later to be able to apply to the companies that offer jobs inside your profile.



These two schemas are the new implementations we decided to make on our project as a result of thinking it would be easier, faster and overall more efficient if students could apply for Erasmus or become a part of the schools club directly from EIS since thor information is already existing.



Personalized Timetable is a very helpful feature especially for second and third year students due to most of them not only taking the courses of that semester, but retake ones as well. Classes that are separated into more than one group(e.g for laboratory hours, seminars,etc.) have also benefited from this. Despite that, a public timetable is available for everyone.



The survey is a form of communication between the students and lecturers through EIS where the right of evaluation is given to the students in an anonymous form.

3.3 Database Dumb (sql files & queries)

Created Tables using My SQL

Table: academiccalendar

Columns:

eventName	varchar(40)
peroidOfTime	varchar(20)
eventdescription	varchar(60)

Table: additionalexams

Columns:

epokaID	int
academicYear	int
courseID	varchar(6)
additionalDate	date
examStatus	varchar(10)
actions	varchar(10)

Table: application

Columns:

epokaID	int
applicationID	int PK
appliactionName	varchar(20
position	varchar(20
companyID	int
dateOfApplication	date

Table: advisor

Columns:

advisorID	int
programID	varchar(6)
yearOftudy	int
noOfStudents	int

Table: attendance

Columns:

epokaID	int
records	int AI PK
courseID	varchar(6)
nrOfWeek	varchar(6)
topic	varchar(40)
hourType	varchar(15)
attendedHours	int
totalHours	int
courseHrsDate	date

Table: bankinfo

Columns:

bankName	varchar(20)
accountName	varchar(30)
accountNo	bigint
accountCurrency	varchar(5)
IBAN	varchar(30)
swiftCode	varchar(10)
paymentDescription	varchar(30)

Table: class

Columns:

classID	varchar(4) PK
building	varchar(1)
floorLevel	int
classNumber	int
classDescription	varchar(30)

Table: clubparticipants

Columns:

epokaID	int
clubID	int
studentRole	varchar(8)

Table: clubs

Columns:

clubID	int PK
clubName	varchar(20)
link	varchar(30)
nrOfStudents	int

Table: clubsevents

Columns:

clubID	int
eventName	varchar(20)
eventDate	date
startTime	time
endTime	time

Table: company

Columns:

companyID	int PK
companyName	varchar(20)
companyDescription	varchar(100)

Table: course

Columns:

programID	varchar(6)
courseID	varchar(6) PK
courseName	varchar(40)
courseType	varchar(20)
lecturerno	int
ECTS	int
Credits	int
noOfStudents	int
averageGrades	decimal(4,2)

Table: courseselection

Columns:

epokaID	int
courseType	varchar(13)
courseID	varchar(6)
advisorID	int
selected	varchar(13)
confirmationDate	date

Table: department

Columns:

facultyID	int
departmentID	int
departamentName	varchar(20)
depDescription	varchar(100)
nrOfPrograms	int
programID	varchar(6) PK

Table: discipline

Columns:

Dtype	varchar(20) PK
dDescription	varchar(100)
timeperiod	varchar(8)

Table: document

Columns:

records	int
numberCount	int AI
epokaID	int
documentType	varchar(20)
requestID	int PK
requestionDate	date
copyRequested	int
price	varchar(10)
payment	varchar(8)
documentStatus	varchar(10)
actions	varchar(40)

Table: erasmus

Columns:

erasmusCode	int PK
universityName	varchar(20)
state	varchar(20)
studyProgram	varchar(20)
yearOfStudy	year
season	int
eDescription	varchar(50)

Table: erasmusapplications

Columns:

applicationID	int PK
erasmusCode	int
epokaID	int

Table: faculty

Columns:

facultyProfile	varchar(40)
facultyID	int PK

Table: finance

Columns:

epokaID	int PK
academicYear	int
debt	int
currency	varchar(5)
total	int

Table: grades

Columns:

letterGrade	varchar(2) PK
gpaValue	decimal(3,2)
points	varchar(6)
gradeDescription	varchar(20)
albGradeSystem	int
avgIncluded	varchar(3)

Table: lecturer

Columns:

lecturerID	int PK
lecturerName	varchar(20)
fatherName	varchar(20)
surname	varchar(20)
title	varchar(20)
gender	varchar(6)
passportNo	varchar(15)
IDCartNo	varchar(15)
birthday	date
birthplace	varchar(15)
citizenship	varchar(15)
phoneNumber	bigint
city	varchar(20)
country	varchar(20)
matrialStatus	varchar(20)
bloodGroup	varchar(20)

Table: lectureraddres

Columns:

lecturerID	int PK
street	varchar(20)
city	varchar(10)
country	varchar(15)
addresDetails	varchar(30)

Table: lectureremail

Columns:

lecturerID	int PK
primaryEmail	varchar(40)
secondaryEmail	varchar(40)

,

Table: lecturerresidence

Columns:

lecturerID	int PK
assignedPermit	varchar(10)
expirationDate	date
issuingAuthority	varchar(10)
permitType	varchar(10)
entryDate	date
exitDate	date
additionalDetails	varchar(30)

Table: lecturerrole

Columns:

lecturerID	int PK
lPosition	varchar(20)
lRole	varchar(20)
startDate	date
lstatus	varchar(15)
departmentID	int

Table: office

Columns:

lecturerID	int PK
officePhone	int
officeNumber	varchar(4)

Table: personalisedtimetable

Columns:

epokaID	int
dayOfTheWeek	varchar(10)
courseID	varchar(6)
startTime	time
endTime	time
lecturerID	int
classID	varchar(4)

Table: professionalpractice

Columns:

epokaID	int
applicationID	int PK
companyID	int

Table: program

Columns:

programCode	int
programID	varchar(6) PK
programName	varchar(40)
pDescription	varchar(100)
semester	int
nrOfCourses	int

Table: student

Columns:

epokaID	int PK
StudentName	varchar(20)
FatherName	varchar(20)
surname	varchar(20)
title	varchar(10)
gender	varchar(7)
birthday	date
country	varchar(20)
city	varchar(20)
citizenship	varchar(20)
passportNo	varchar(15)
IDCardNo	varchar(15)
primaryEmil	varchar(40)
secondaryEmail	varchar(40)
phoneNr	bigint
matrialStatus	varchar(10)
bloodGroup	varchar(10)
englishLevel	varchar(3)
programID	varchar(6)

Table: resitexam

Columns:

epokaID	int
courseID	varchar(6)
selectStatus	varchar(15)

Table: studentaddres

Columns:

epokaID	int PK
street	varchar(20)
city	varchar(10)
country	varchar(15)
addresDetails	varchar(30)

Table: studentcourses

Columns:

epokaID	int
courseID	varchar(6)
letterGrade	varchar(2)
grade	decimal(3,2)

Table: studentdiscipline

Columns:

epokaID	int
Dtype	varchar(20)
Sstatus	varchar(10)
startTime	date
endTime	date

Table: studentfamily

Columns:

epokaID	int PK
fatherName	varchar(10)
Fsurname	varchar(15)
fPhoneNo	bigint
fWork	varchar(20)
fProfession	varchar(20)
motherName	varchar(10)
Msurname	varchar(15)
mPhoneNo	bigint
mWork	varchar(20)
mProfession	varchar(20)

Table: studentgroup

Columns:	
epokaID	int
programID	varchar(6)
yearOftudy	int
studentGroup	varchar(1)
hourType	varchar(10)

Table: studentresidence

Columns:	
epokaID	int PK
assignedPermit	varchar(10)
expirationDate	date
issuingAuthority	varchar(10)
permitType	varchar(10)
entryDate	date
exitDate	date
additionalDetails	varchar(30)

Table: studentrole

Columns:	
epokaID	int PK
unit	varchar(10)
pPosition	varchar(10)
project	varchar(20)
pRole	varchar(10)
startDate	date
endDate	date
pStatus	varchar(10)

Table: survey

Columns:	
programID	varchar(6)
epokaID	int
courseID	varchar(6)
lecturerID	int
question	varchar(40)
evaluation	int
additionalReview	varchar(60)

Table: teachedcourses

Columns:	
lecturerID	int
courseID	varchar(6)
semesterYear	varchar(10)
lecturerRole	varchar(10)

Table: thirdsemester

Columns:	
epokaID	int
courseID	varchar(6)
courseStatus	varchar(10)

Table: timetable

Columns:	
season	varchar(10)
StudentType	varchar(15)
programID	varchar(6)
programYear	int
yearGroup	varchar(1)
dayOfTheWeek	varchar(10)
courseID	varchar(6)
startTime	time
endTime	time
lecturerID	int
classID	varchar(4)

Table: transport

Columns:	
routeID	int PK
route	varchar(20)
departureTime	datetime
routestatus	varchar(15)
busStation	varchar(20)
stationTime	time

Table: transportreservation

Columns:	
epokaID	int
routeID	int PK

3.3.1 Create & Insert data

Here is a representation of the way the tables were created and then populated with data. Below are included only a few of the main tables.

Create Student Table

```sql
CREATE TABLE Student(epokaID INT NOT NULL UNIQUE,StudentName VARCHAR(20) NOT NULL, FatherName VARCHAR(20),
surname VARCHAR(20) NOT NULL ,  title VARCHAR(10), gender VARCHAR(7),
birthday DATE NOT NULL, country VARCHAR(20), city VARCHAR(20), citizenship VARCHAR(20),
passportNo VARCHAR(15) NOT NULL UNIQUE, IDCardNo VARCHAR(15) NOT NULL UNIQUE,
primaryEmil VARCHAR(40) NOT NULL UNIQUE, secondaryEmail VARCHAR(40) NOT NULL UNIQUE,
phoneNr BIGINT NOT NULL, matrialStatus VARCHAR(10),
bloodGroup VARCHAR(10), englishLevel VARCHAR(3) NOT NULL,
PRIMARY KEY(epokaID));
```

Insert Data in Student Table

```sql
INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship,
passportNo, IDCardNo, primaryEmil, secondaryEmail, phoneNr, matrialStatus, bloodGroup, englishLevel,programID)
VALUES("020522110","Michael","Jack","Smith","Male","2004-05-04","Albania","Fier,ALB","Albanian","K12343908K",
"1234K908","msmith22@epoka.edu.al","michaelsmith54@gmail.com","0693375758","single","0+","B2","SWE");
INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship,
passportNo, IDCardNo, primaryEmil, secondaryEmail, phoneNr, matrialStatus, bloodGroup, englishLevel,programID)
VALUES("020321090","Ana","Besmir","Hoxha","Female","2002-11-04","Albania","Tirane,ALB","Albanian","K18743908E",
"K18743908E","ahoxha21@epoka.edu.al","annahoxha@gmail.com","0693355858","single","A+","B2","CEN");
INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship,
passportNo, IDCardNo, primaryEmil, secondaryEmail, phoneNr, matrialStatus, bloodGroup, englishLevel,programID)
VALUES("02052286","Maida","Idris","Daulle","Female","2004-3-14","Albania","Lushnje,ALB","Albanian","K1154908E",
"K1154908E","mdaulle22@epoka.edu.al","maidadaulle04@gmail.com","0693623672","single","0+","B2","SWE");
INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship,
passportNo, IDCardNo, primaryEmil, secondaryEmail, phoneNr, matrialStatus, bloodGroup, englishLevel,programID)
VALUES("01042055","Ema","Ilir","Lika","Female","2002-5-14","Albania","Shkoder,ALB","Albanian","K1154458P",
"K1154458P","elika20@epoka.edu.al","emalika14@gmail.com","0693488672","single","0+","B2","BINF"),
("01032201","Arben","Shpetim","Dervishi","Male","2003-12-30","Albania","Tirane,ALB","Albanian","K0002908E",
"K0002908E","adervishi22@epoka.edu.al","arbendervishi@gmail.com","069364499","single","B-","B2","BAF");
```

Create and Insert Data in Lecturer Table

```sql
CREATE TABLE Lecturer( lecturerID INT NOT NULL, lecturerName VARCHAR(20) NOT NULL,
fatherName VARCHAR(20),surname VARCHAR(20) NOT NULL, title VARCHAR(20) NOT NULL, gender VARCHAR(6),
passportNo VARCHAR(15) NOT NULL UNIQUE, IDCartNo VARCHAR(15) NOT NULL UNIQUE ,birthday DATE, birthplace VARCHAR(15),
citizenship VARCHAR(15),phoneNumber BIGINT NOT NULL,city VARCHAR(20),
country VARCHAR(20) ,matrialStatus VARCHAR(20),bloodGroup VARCHAR(20),
PRIMARY KEY (lecturerID));
INSERT INTO Lecturer(lecturerID,lecturerName,fatherName,surname,title,gender,passportNo,IDCartNo,birthday,birthplace,
citizenship,phoneNumber,city, country, matrialStatus,bloodGroup)
VALUES("1000","Arban","Arben","Uka","DR.","Male","P7869425K","P7869425K","1980-5-28","Tirane,ALB","Albanian","0684587624","Tirane","Albania","married","0+");
INSERT INTO Lecturer(lecturerID,lecturerName,fatherName,surname,title,gender,passportNo,IDCartNo,birthday,birthplace,
citizenship,phoneNumber,city, country, matrialStatus,bloodGroup)
VALUES("1001","Igli","Besmir","Drraci","M. SC.","Male","F7869425E","F7869425E","1994-5-8","Tirane,ALB","Albanian","0690087624","Tirane","Albania","single","A-"),
("1003","Redjola","Ilir","Manaj","M. SC.","Female","P7895567R","P7895567R","1992-12-01","Vlore,ALB","Albanian","0684578464","Tirane","Albania","single","0+"),
("1004","Florenc","Ilir","Skuka","M. SC.","Male","K7849562P","K7849562P","1985-01-19","Tirane,ALB","Albanian","0698758691","Tirane","Albania","married","B+"),
("1005","Sabrina","Gezim","Begaj","M. SC.","Female","P7859004K","P7859004K","1993-03-19","Tirane,ALB","Albanian","0698700241","Tirane","Albania","married","B+");
INSERT INTO Lecturer(lecturerID,lecturerName,fatherName,surname,title,gender,passportNo,IDCartNo,birthday,birthplace,
citizenship,phoneNumber,city, country, matrialStatus,bloodGroup)
VALUES("1006","Eriselda","Artan","Goga","DR.","Female","P7869004K","P7869004K","1992-7-2","Fier,ALB","Albanian","0690087624","Tirane","Albania","single","0+"),
("1007","Ari","Astrit","Gjerazi","DR.","Male","P7024580M","P7024580M","1994-9-12","Fier,ALB","Albanian","0690579842","Tirane","Albania","single","A+");
INSERT INTO Lecturer(lecturerID,lecturerName,fatherName,surname,title,gender,passportNo,IDCartNo,birthday,birthplace,
citizenship,phoneNumber,city, country, matrialStatus,bloodGroup)
VALUEs("2001","Chrysanthi","Jani","Balomenou","DR.","Female","47861021P","47861021P","1960-8-01","Athens,GR","Greek","0698749568","Tirane","Albania","married","B+
```

Create and Insert Data in Courses Table

```
CREATE TABLE Course( programID VARCHAR(6) NOT NULL,courseID VARCHAR(6) NOT NULL UNIQUE,courseName VARCHAR(40),
courseType VARCHAR(20), lecturerno INT,ECTS INT, Credits INT, noOfStudents INT,
 averageGrades NUMERIC(4,2),
PRIMARY KEY (courseID),
FOREIGN KEY(programID) REFERENCES Program(programID));
INSERT INTO Course (programID,courseID, courseName, courseType, lecturerno, ECTS, Credits,noOfStudents)
VALUES("SWE","CEN109","Introduction to Algorithms","A","3","7","4","120"),("ARCH","MTH125","Bacis Mathematics","A","1","4","3","100");
INSERT INTO Course (programID,courseID, courseName, courseType, lecturerno, ECTS, Credits,noOfStudents)
VALUES("CEN","ENG102","ENGLISH II","E","1","5","4","110"),("SWE","CEN203","DATABASE MANAGEMENT SYSTEMS","B","2","7","4","135"),
("SWE","SWE101","INTRO TO SOFTWARE ENGINEERING","B","3","7","4","128"),("BINF","BUS101","MATH. FOR ECONOMICS AND BUSINESS I","A","1","5","4","89"),
("BAF","BAF101","INTRODUCTION TO ECONOMICS","A","1","5","3","30");
INSERT INTO Course(programID,courseID, courseName, courseType, lecturerno, ECTS, Credits,noOfStudents)
VALUES("SWE","MTH207","FUNDAMENTALS OF PROBABILITY","A","2","6","3","150"),("SWE","CEN219","COMPUTER ORGANISATION","C","1","6","3","120"),
("CEN","CEN215","OBJECT ORIENTED PROGRAMIMG","B","3","7","4","200");
```

Create and Insert Data in Grades Table

```
CREATE TABLE Grades(letterGrade VARCHAR(2), gpaValue NUMERIC(3,2),
points INT, gradeDescription VARCHAR(20), albGradeSystem INT, avgIncluded VARCHAR(3),
PRIMARY KEY(letterGrade));
ALTER TABLE Grades MODIFY COLUMN points VARCHAR(6);
INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded)
VALUES("AA","4.00","90-100","Excellent","10","I");
INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded)
VALUES("DC","1.50","65-69","ON PROBATION","6","I");
INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded)
VALUES("BA","3.50","85-89","Excellent","9","I"), ("BB","3.00","80-84","SUCCESS","8","I"),
("CB","2.50","75-79","SUCCESS","8","I"), ("CC","2.00","70-74","SUCCESS","7","I");
INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded)
VALUES("DD","1.00","60-64","ON PROBSCION","5","I"), ("FD","0.50","50-59","FAIL","4","I"),
("FF","0.00","0-49","FAIL","0","I"),("NA","0.00","0","FAIL","0","NI");

```

Create and Insert Data in Transport Table

```
 CREATE TABLE Transport(routeID INT NOT NULL , route VARCHAR(20) NOT NULL,
departureTime DATETIME NOT NULL, routeStatus VARCHAR(10),
busStation VARCHAR(20), stationTime TIME NOT NULL,
PRIMARY KEY (routeID));
ALTER TABLE TRANSPORT MODIFY COLUMN routestatus VARCHAR(15);
INSERT INTO Transport(routeID, route,departureTime, routeStatus, busStation, stationTime)
VALUES("001","Tirana-Campus","2023-11-15 7:30","Departed","Tirana Center","8:00");
INSERT INTO Transport(routeID, route,departureTime, routeStatus, busStation, stationTime)
VALUES("002","Campus-Durres","2024-01-09 12:30","NotDeparted","Campus","12:30");
```

3.3.2 Managerial Queries

Find attached 20 queries that we have created as well as a short description of them.

```
429    -- QUERY 1
430    SELECT studentName,surname,courseID,grade
431    FROM student,studentCourses
432    WHERE student.epokaID=studentCourses.epokaID
433    HAVING grade>=2;
434
435    -- QUERY 2
436    SELECT s.epokaID, s.StudentName, s.surname, AVG(sc.grade) AS average_grade
437    FROM Student s
438    JOIN StudentCourses sc ON s.epokaID = sc.epokaID
439    GROUP BY s.epokaID, s.StudentName, s.surname;
440
441    -- QUERY 3
442    SELECT c.programID, c.courseID, AVG(sc.grade) AS average
443    FROM Course c
444    JOIN StudentCourses sc ON c.courseID = sc.courseID
445    JOIN Student s ON sc.epokaID = s.epokaID
446    GROUP BY c.programID, c.courseID;
447
448    -- QUERY 4
449    SELECT tc.lecturerID,l.lecturerName,l.surname,SUM(noOfStudents) AS totalNrOfStudents
450    FROM teachedCourses tc
451    JOIN course c ON tc.courseID=c.courseID
452    JOIN Lecturer l ON tc.lecturerID=l.lecturerID
453    GROUP BY tc.lecturerID, l.lecturerName, l.surname;
```

➔ ***Query 1***

The query displays student name, surname, course ID and grade of students that have the course grade higher or equal to 2.

➔ ***Query 2***

The query displays information related to students such as Epoka ID, student name and surname and it calculates the overall average of each student and creates a new column to display it.

➔ ***Query 3***

The query displays the Course ID and the corresponding Program ID for each course and one column is added to calculate the course average by the grades of all students that have selected the specified course.

➔ ***Query 4***

The query displays the Lecturer ID , name and surname and adds another column in which is calculated the total number of students they have in all courses they teach.

```
Limit to 1000 rows
455  -- QUERY 5
456  SELECT a.programID,le.primaryEmail
457  FROM lecturerEmail le
458  JOIN advisor a ON le.lecturerID=a.advisorID
459  GROUP BY a.programID,le.primaryEmail;
460
461  -- QUERY 6
462  SELECT officeNumber, COUNT(lecturerID) AS lecturerCount
463  FROM Office
464  GROUP BY officeNumber;
465
466  -- QUERY 7
467  SELECT s.epokaID, s.programID, sc.courseID, c.courseName
468  FROM Student s
469  JOIN StudentCourses sc ON s.epokaID = sc.epokaID
470  JOIN Course c ON sc.courseID = c.courseID
471  WHERE s.programID IN (
472      SELECT p.programID
473      FROM Program p
474      WHERE p.programName = "Software Engineering");
475
476  -- QUERY 8
477  SELECT l.lecturerID, l.lecturerName,l.surname,s.courseID ,AVG(s.evaluation) AS avgEvaluationScore
478  FROM Lecturer l
479  JOIN Survey s ON l.lecturerID = s.lecturerID
480  GROUP BY l.lecturerID, l.lecturerName,l.surname,s.courseID
481  HAVING avgEvaluationScore < 2;
482
483  -- QUERY 9
484  SELECT courseID, lecturerID, AVG(evaluation) AS AvgEvaluation
485  FROM Survey
486  GROUP BY courseID, lecturerID;
487
```

➔ ***Query 5***

It displays the Program ID and the corresponding email for the advisor of the program.

➔ ***Query 6***

The query displays the office number and it counts how many lecturers are located in that office.

➔ ***Query 7***

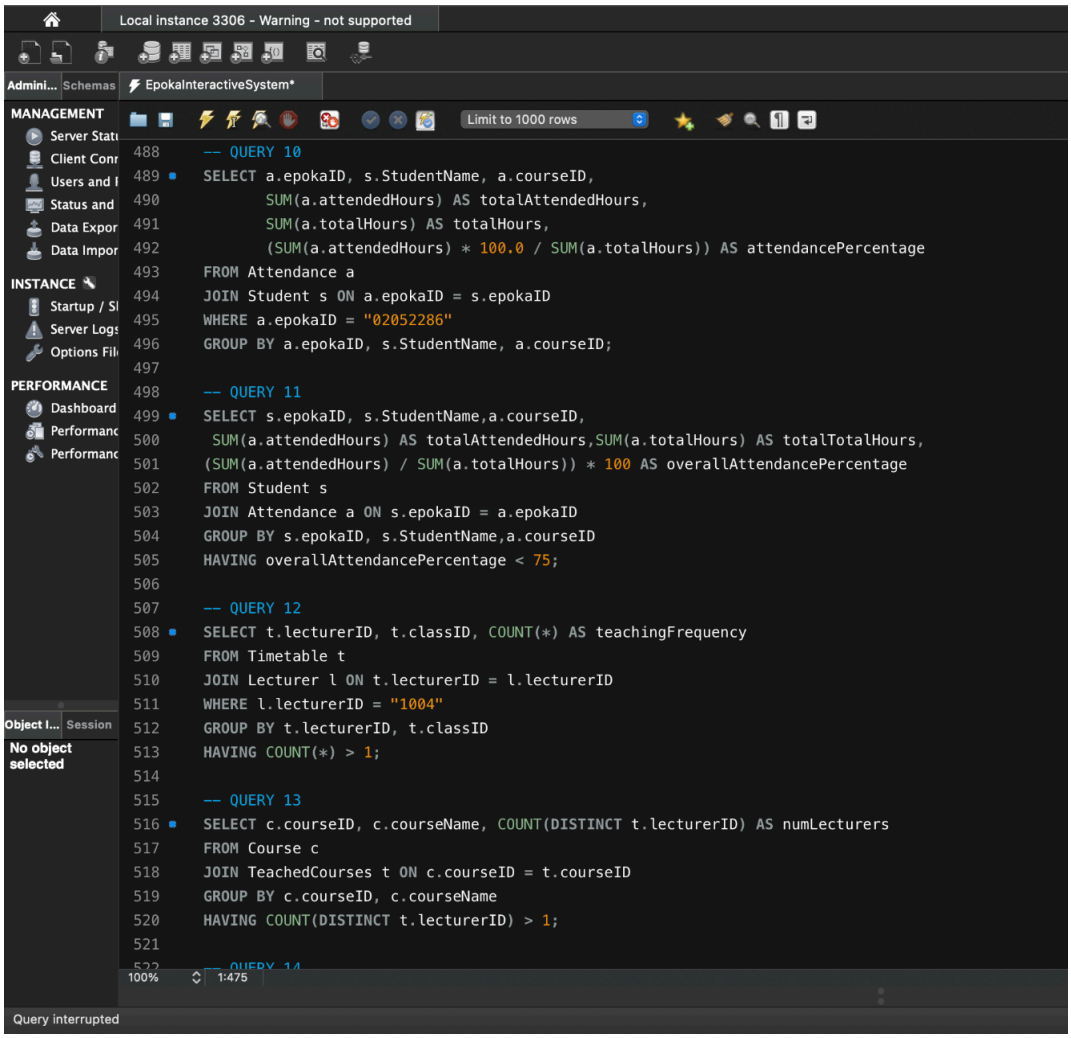
Inner query gives the program name corresponding Program ID and shows the student Epoka ID, Program ID and the courses they have selected from that program giving the Course ID and name.

➔ ***Query 8***

The query displays Lecturer ID, lecturer name and surname and the Course ID that they teach when their average evaluation from surveys of students is below 2 and it also shows the values of the avg .

➔ *Query 9*

This query displays the Course ID, the lecturer that teaches, the Course ID and the average of the evaluation of the surveys for that lecturer.




```
488   -- QUERY 10
489   SELECT a.epokaID, s.StudentName, a.courseID,
490          SUM(a.attendedHours) AS totalAttendedHours,
491          SUM(a.totalHours) AS totalHours,
492          (SUM(a.attendedHours) * 100.0 / SUM(a.totalHours)) AS attendancePercentage
493   FROM Attendance a
494   JOIN Student s ON a.epokaID = s.epokaID
495   WHERE a.epokaID = "02052286"
496   GROUP BY a.epokaID, s.StudentName, a.courseID;
497
498   -- QUERY 11
499   SELECT s.epokaID, s.StudentName,a.courseID,
500    SUM(a.attendedHours) AS totalAttendedHours,SUM(a.totalHours) AS totalTotalHours,
501   (SUM(a.attendedHours) / SUM(a.totalHours)) * 100 AS overallAttendancePercentage
502   FROM Student s
503   JOIN Attendance a ON s.epokaID = a.epokaID
504   GROUP BY s.epokaID, s.StudentName,a.courseID
505   HAVING overallAttendancePercentage < 75;
506
507   -- QUERY 12
508   SELECT t.lecturerID, t.classID, COUNT(*) AS teachingFrequency
509   FROM Timetable t
510   JOIN Lecturer l ON t.lecturerID = l.lecturerID
511   WHERE l.lecturerID = "1004"
512   GROUP BY t.lecturerID, t.classID
513   HAVING COUNT(*) > 1;
514
515   -- QUERY 13
516   SELECT c.courseID, c.courseName, COUNT(DISTINCT t.lecturerID) AS numLecturers
517   FROM Course c
518   JOIN TeachedCourses t ON c.courseID = t.courseID
519   GROUP BY c.courseID, c.courseName
520   HAVING COUNT(DISTINCT t.lecturerID) > 1;
521
```

➔ *Query 10*

This query calculates the attendance of a specific student Epoka ID and shows the Student ID, the student name, attended courses, the attended hours, total number of hours and the attendance percentage in a new

column

➔ ***Query 11***

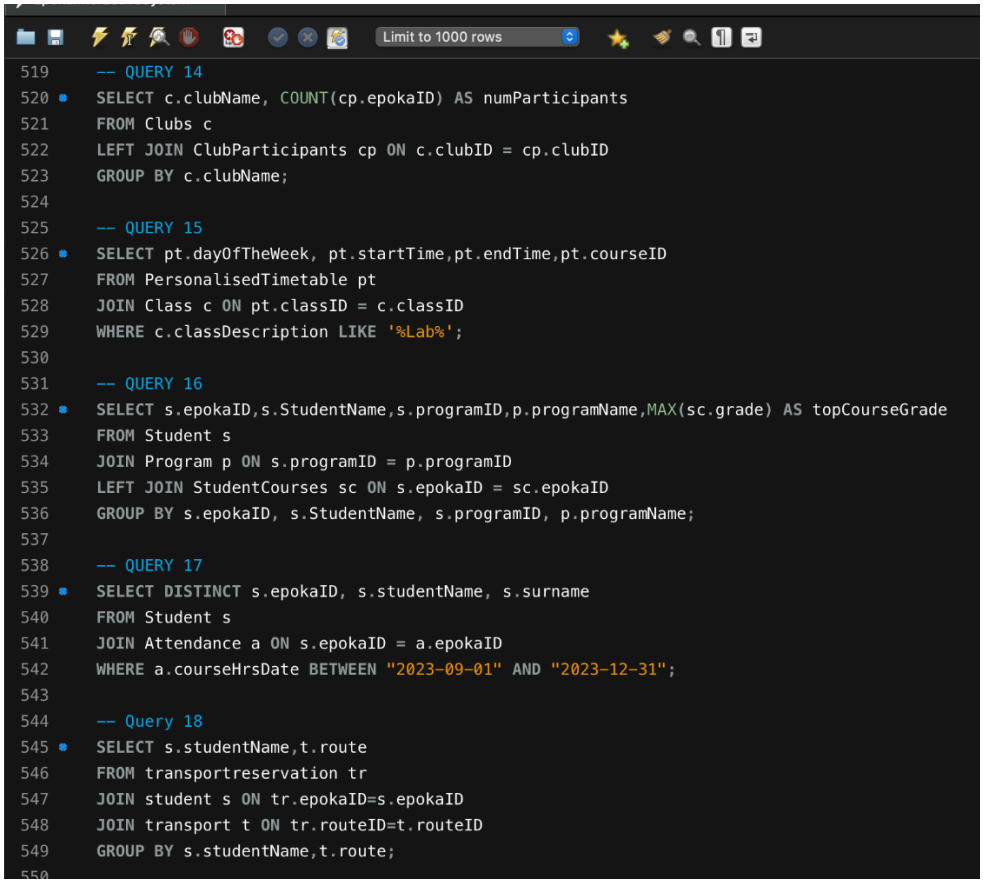
This query calculates each student attendance percentage and it shows the Epoka ID, name, Course ID, attendant hours total hours and the attendance percentage of the student whose percentage is below 75

➔ ***Query 12***

This query displays the Lecturer ID and the classroom ID, if the lecturer teaches more than once in the same classroom during the week and it also shows the frequency of this happening.

➔ ***Query 13***

This query calculates the number of lecturers on each course and displays the Course ID, name and the number of lecturers that give that course if the number is larger than 1.



```
519   -- QUERY 14
520   SELECT c.clubName, COUNT(cp.epokaID) AS numParticipants
521   FROM Clubs c
522   LEFT JOIN ClubParticipants cp ON c.clubID = cp.clubID
523   GROUP BY c.clubName;
524
525   -- QUERY 15
526   SELECT pt.dayOfTheWeek, pt.startTime,pt.endTime,pt.courseID
527   FROM PersonalisedTimetable pt
528   JOIN Class c ON pt.classID = c.classID
529   WHERE c.classDescription LIKE '%Lab%';
530
531   -- QUERY 16
532   SELECT s.epokaID,s.StudentName,s.programID,p.programName,MAX(sc.grade) AS topCourseGrade
533   FROM Student s
534   JOIN Program p ON s.programID = p.programID
535   LEFT JOIN StudentCourses sc ON s.epokaID = sc.epokaID
536   GROUP BY s.epokaID, s.StudentName, s.programID, p.programName;
537
538   -- QUERY 17
539   SELECT DISTINCT s.epokaID, s.studentName, s.surname
540   FROM Student s
541   JOIN Attendance a ON s.epokaID = a.epokaID
542   WHERE a.courseHrsDate BETWEEN "2023-09-01" AND "2023-12-31";
543
544   -- Query 18
545   SELECT s.studentName,t.route
546   FROM transportreservation tr
547   JOIN student s ON tr.epokaID=s.epokaID
548   JOIN transport t ON tr.routeID=t.routeID
549   GROUP BY s.studentName,t.route;
```

➔ ***Query 14***

This query displays the club name and the number of students that are participants in that club. *The Left join* allows to show on the table venue the club name whole participant number is 0

➔ ***Query 15***

This query displays the day start time, end time of the courses that take place in a lab type of classroom.

➔ ***Query 16***

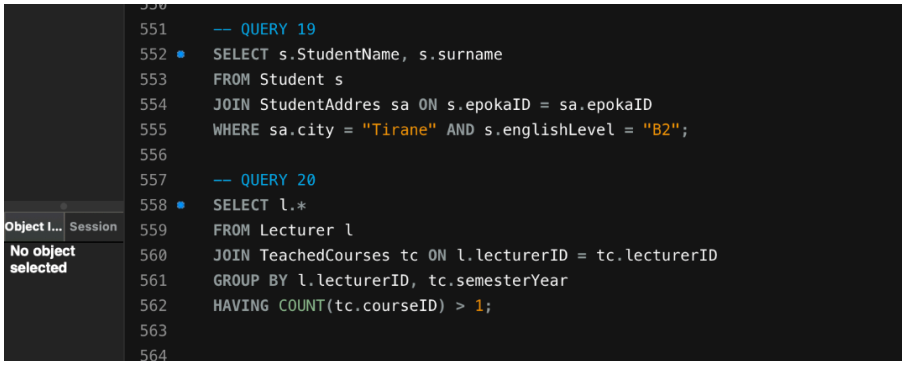
This query finds the maximum grade of each student of all their selected courses and displays their Epoka ID, name, Program ID and name and also the corresponding maximum grade of each one of them.

➔ ***Query 17***

This query displays Epoka ID, student name and surname of the students that have attended any course hour during the specified period of time

➔ ***Query 18***

This query displays the student name and the route name of the corresponding Epoka ID and the Route ID reservation made.



➔ ***Query 19***

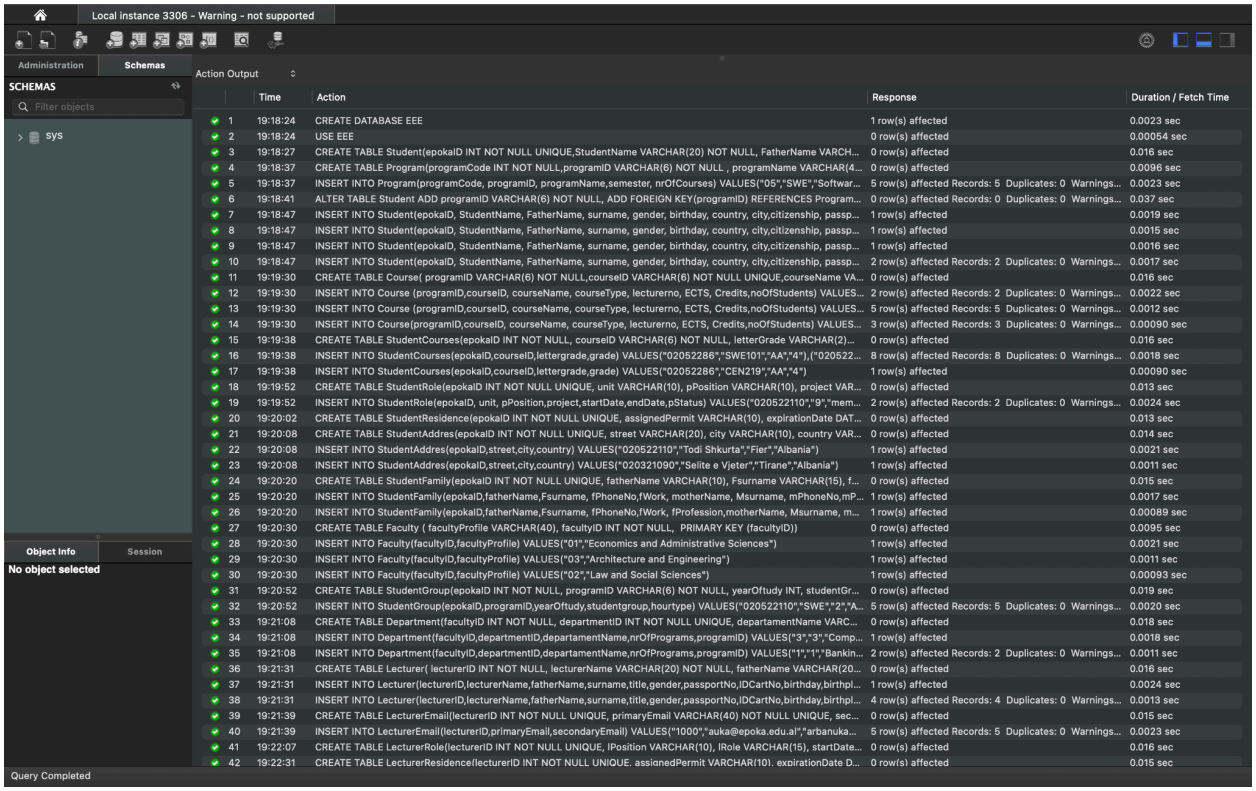
This query displays the name and surname of the student who lives in Tirana and also has an English level of B2 , for the name to be shown both conditions should be satisfied.

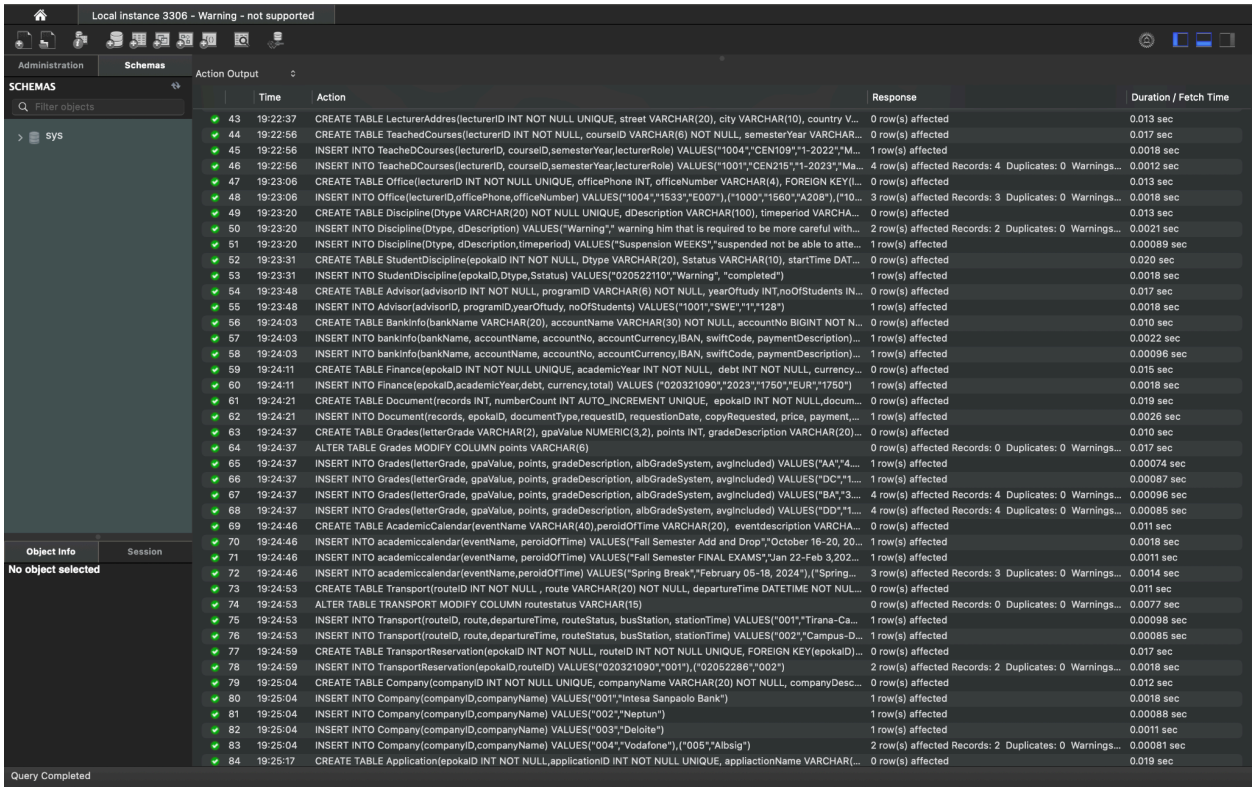
➔ ***Query 20***

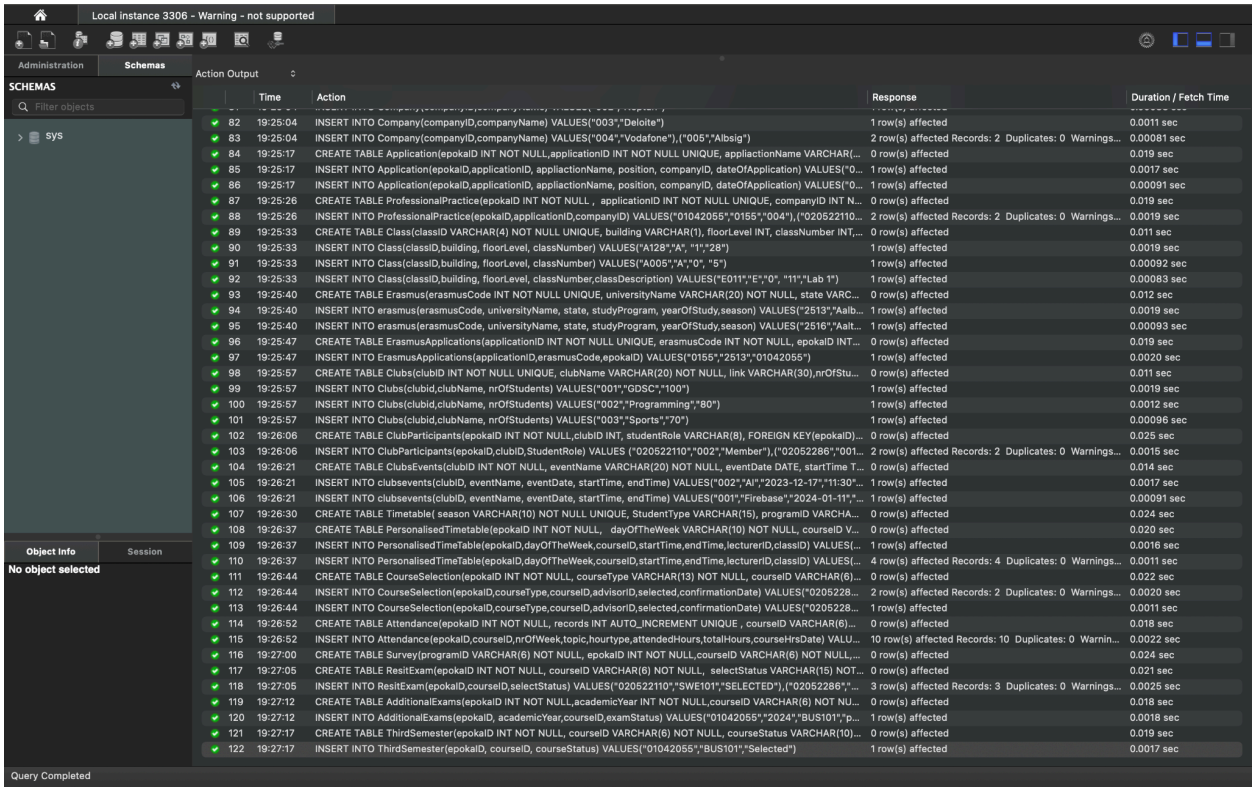
This query displays all information from the Lecturer table who have teached more than one course during the same semester.

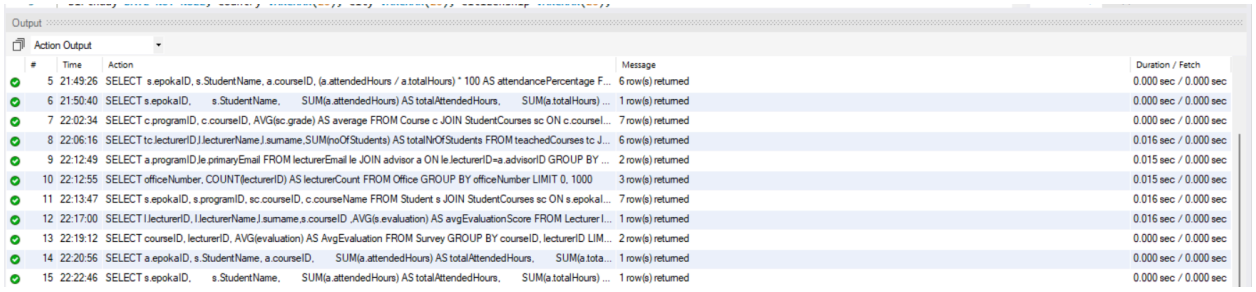
3.3.3 Performance

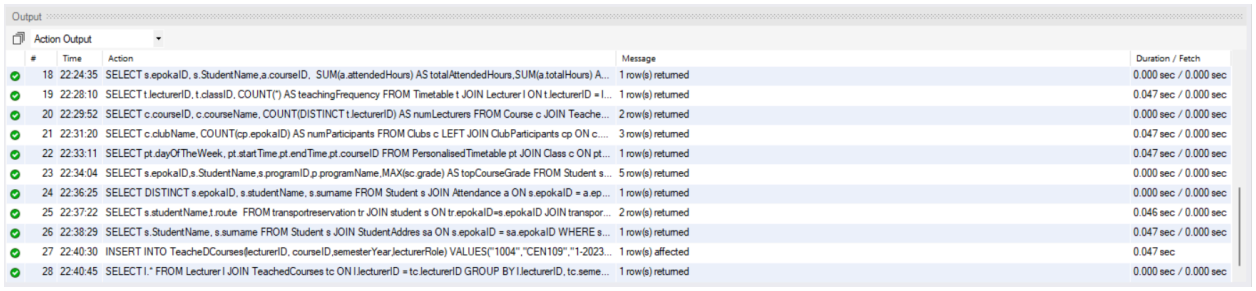
As previously mentioned in the ***2.4*** part of our documentation, throughout the creation of this project we have taken measurement of the performance time and always had the intention to improve our performance. This is also presented below:


Local instance 3306 - Warning - not supported
Administration
Schemas
SCHEMAS
Filter objects
sys
Object Info
Session
No object selected
Action Output
Time
Action
Response
Duration / Fetch Time
1 19:18:24 CREATE DATABASE EEE 1 row(s) affected 0.0023 sec
2 19:18:24 USE EEE 0 row(s) affected 0.00054 sec
3 19:18:27 CREATE TABLE Student(epokaID INT NOT NULL UNIQUE,StudentName VARCHAR(20) NOT NULL, FatherName VARCH... 0 row(s) affected 0.016 sec
4 19:18:37 CREATE TABLE Program(programCode INT NOT NULL,programID VARCHAR(6) NOT NULL , programName VARCHAR(4... 0 row(s) affected 0.0096 sec
5 19:18:37 INSERT INTO Program(programCode, programID, programName,semester, nrOfCourses) VALUES("05","SWE","Softwar... 5 row(s) affected Records: 5 Duplicates: 0 Warnings... 0.0023 sec
6 19:18:41 ALTER TABLE Student ADD programID VARCHAR(6) NOT NULL, ADD FOREIGN KEY(programID) REFERENCES Program... 0 row(s) affected Records: 0 Duplicates: 0 Warnings... 0.037 sec
7 19:18:47 INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship, passp... 1 row(s) affected 0.0019 sec
8 19:18:47 INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship, passp... 1 row(s) affected 0.0015 sec
9 19:18:47 INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship, passp... 1 row(s) affected 0.0016 sec
10 19:18:47 INSERT INTO Student(epokaID, StudentName, FatherName, surname, gender, birthday, country, city,citizenship, passp... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0017 sec
11 19:19:30 CREATE TABLE Course(programID VARCHAR(6) NOT NULL,courseID VARCHAR(6) NOT NULL UNIQUE,courseName VA... 0 row(s) affected 0.016 sec
12 19:19:30 INSERT INTO Course (programID,courseID, courseName, courseType, lecturerno, ECTS, Credits,noOfStudents) VALUES... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0022 sec
13 19:19:30 INSERT INTO Course (programID,courseID, courseName, courseType, lecturerno, ECTS, Credits,noOfStudents) VALUES... 5 row(s) affected Records: 5 Duplicates: 0 Warnings... 0.0012 sec
14 19:19:30 INSERT INTO Course(programID,courseID, courseName, courseType, lecturerno, ECTS, Credits,noOfStudents) VALUES... 3 row(s) affected Records: 3 Duplicates: 0 Warnings... 0.00090 sec
15 19:19:38 CREATE TABLE StudentCourses(epokaID INT NOT NULL, courseID VARCHAR(6) NOT NULL, letterGrade VARCHAR(2)... 0 row(s) affected 0.016 sec
16 19:19:38 INSERT INTO StudentCourses(epokaID,courseID,lettergrade,grade) VALUES("02052286","SWE101","AA","4"),("020522... 8 row(s) affected Records: 8 Duplicates: 0 Warnings... 0.0018 sec
17 19:19:38 INSERT INTO StudentCourses(epokaID,courseID,lettergrade,grade) VALUES("02052286","CEN219","AA","4") 1 row(s) affected 0.00090 sec
18 19:19:52 CREATE TABLE StudentRole(epokaID INT NOT NULL UNIQUE, unit VARCHAR(10), pPosition VARCHAR(10), project VAR... 0 row(s) affected 0.013 sec
19 19:19:52 INSERT INTO StudentRole(epokaID, unit, pPosition,project,startDate,endDate,pStatus) VALUES("020522110","9","mem... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0024 sec
20 19:20:02 CREATE TABLE StudentResidence(epokaID INT NOT NULL UNIQUE, assignedPermit VARCHAR(10), expirationDate DAT... 0 row(s) affected 0.013 sec
21 19:20:08 CREATE TABLE StudentAddres(epokaID INT NOT NULL UNIQUE, street VARCHAR(20), city VARCHAR(10), country VAR... 0 row(s) affected 0.014 sec
22 19:20:08 INSERT INTO StudentAddres(epokaID,street,city,country) VALUES("020522110","Todi Shkurta","Fier","Albania") 1 row(s) affected 0.0021 sec
23 19:20:08 INSERT INTO StudentAddres(epokaID,street,city,country) VALUES("020321090","Selite e Vjeter","Tirane","Albania") 1 row(s) affected 0.0011 sec
24 19:20:20 CREATE TABLE StudentFamily(epokaID INT NOT NULL UNIQUE, fatherName VARCHAR(10), Fsurname VARCHAR(15), f... 0 row(s) affected 0.015 sec
25 19:20:20 INSERT INTO StudentFamily(epokaID,fatherName,Fsurname, fPhoneNo,fWork, motherName, Msurname, mPhoneNo,mP... 1 row(s) affected 0.0017 sec
26 19:20:20 INSERT INTO StudentFamily(epokaID,fatherName,Fsurname, fPhoneNo,fWork, fProfession,motherName, Msurname, m... 1 row(s) affected 0.00089 sec
27 19:20:30 CREATE TABLE Faculty (facultyProfile VARCHAR(40), facultyID INT NOT NULL, PRIMARY KEY (facultyID)) 0 row(s) affected 0.0095 sec
28 19:20:30 INSERT INTO Faculty(facultyID,facultyProfile) VALUES("01","Economics and Administrative Sciences") 1 row(s) affected 0.0021 sec
29 19:20:30 INSERT INTO Faculty(facultyID,facultyProfile) VALUES("03","Architecture and Engineering") 1 row(s) affected 0.0011 sec
30 19:20:30 INSERT INTO Faculty(facultyID,facultyProfile) VALUES("02","Law and Social Sciences") 1 row(s) affected 0.00093 sec
31 19:20:52 CREATE TABLE StudentGroup(epokaID INT NOT NULL, programID VARCHAR(6) NOT NULL, yearOftudy INT, studentGr... 0 row(s) affected 0.019 sec
32 19:20:52 INSERT INTO StudentGroup(epokaID,programID,yearOftudy,studentgroup,hourtype) VALUES("020522110","SWE","2","A... 5 row(s) affected Records: 5 Duplicates: 0 Warnings... 0.0020 sec
33 19:21:08 CREATE TABLE Department(facultyID INT NOT NULL, departmentID INT NOT NULL UNIQUE, departamentName VARC... 0 row(s) affected 0.018 sec
34 19:21:08 INSERT INTO Department(facultyID,departmentID,departamentName,nrOfPrograms,programID) VALUES("3","3","Comp... 1 row(s) affected 0.0018 sec
35 19:21:08 INSERT INTO Department(facultyID,departmentID,departamentName,nrOfPrograms,programID) VALUES("1","1","Bankin... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0011 sec
36 19:21:31 CREATE TABLE Lecturer(lecturerID INT NOT NULL, lecturerName VARCHAR(20) NOT NULL, fatherName VARCHAR(20... 0 row(s) affected 0.016 sec
37 19:21:31 INSERT INTO Lecturer(lecturerID,lecturerName,fatherName,surname,title,gender,passportNo,IDCartNo,birthday,birthpl... 1 row(s) affected 0.0024 sec
38 19:21:31 INSERT INTO Lecturer(lecturerID,lecturerName,fatherName,surname,title,gender,passportNo,IDCartNo,birthday,birthpl... 4 row(s) affected Records: 4 Duplicates: 0 Warnings... 0.0013 sec
39 19:21:39 CREATE TABLE LecturerEmail(lecturerID INT NOT NULL UNIQUE, primaryEmail VARCHAR(40) NOT NULL UNIQUE, sec... 0 row(s) affected 0.015 sec
40 19:21:39 INSERT INTO LecturerEmail(lecturerID,primaryEmail,secondaryEmail) VALUES("1000","auka@epoka.edu.al","arbanuka... 5 row(s) affected Records: 5 Duplicates: 0 Warnings... 0.0023 sec
41 19:22:07 CREATE TABLE LecturerRole(lecturerID INT NOT NULL UNIQUE, lPosition VARCHAR(10), lRole VARCHAR(15), startDate... 0 row(s) affected 0.016 sec
42 19:22:31 CREATE TABLE LecturerResidence(lecturerID INT NOT NULL UNIQUE, assignedPermit VARCHAR(10), expirationDate D... 0 row(s) affected 0.015 sec
Query Completed


Local instance 3306 - Warning - not supported
Administration
Schemas
SCHEMAS
Filter objects
sys
Object Info
Session
No object selected
Action Output
Time
Action
Response
Duration / Fetch Time
43 19:22:37 CREATE TABLE LecturerAddres(lecturerID INT NOT NULL UNIQUE, street VARCHAR(20), city VARCHAR(10), country V... 0 row(s) affected 0.013 sec
44 19:22:56 CREATE TABLE TeachedCourses(lecturerID INT NOT NULL, courseID VARCHAR(6) NOT NULL, semesterYear VARCHAR... 0 row(s) affected 0.017 sec
45 19:22:56 INSERT INTO TeacheDCourses(lecturerID, courseID,semesterYear,lecturerRole) VALUES("1004","CEN109","1-2022","M... 1 row(s) affected 0.0018 sec
46 19:22:56 INSERT INTO TeacheDCourses(lecturerID, courseID,semesterYear,lecturerRole) VALUES("1001","CEN215","1-2023","Ma... 4 row(s) affected Records: 4 Duplicates: 0 Warnings... 0.0012 sec
47 19:23:06 CREATE TABLE Office(lecturerID INT NOT NULL UNIQUE, officePhone INT, officeNumber VARCHAR(4), FOREIGN KEY(l... 0 row(s) affected 0.013 sec
48 19:23:06 INSERT INTO Office(lecturerID,officePhone,officeNumber) VALUES("1004","1533","E007"),("1000","1560","A208"),("10... 3 row(s) affected Records: 3 Duplicates: 0 Warnings... 0.0018 sec
49 19:23:20 CREATE TABLE Discipline(Dtype VARCHAR(20) NOT NULL UNIQUE, dDescription VARCHAR(100), timeperiod VARCHA... 0 row(s) affected 0.013 sec
50 19:23:20 INSERT INTO Discipline(Dtype, dDescription) VALUES("Warning"," warning him that is required to be more careful with... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0021 sec
51 19:23:20 INSERT INTO Discipline(Dtype, dDescription,timeperiod) VALUES("Suspension WEEKS","suspended not be able to atte... 1 row(s) affected 0.00089 sec
52 19:23:31 CREATE TABLE StudentDiscipline(epokaID INT NOT NULL, Dtype VARCHAR(20), Sstatus VARCHAR(10), startTime DAT... 0 row(s) affected 0.020 sec
53 19:23:31 INSERT INTO StudentDiscipline(epokaID,Dtype,Sstatus) VALUES("020522110","Warning", "completed") 1 row(s) affected 0.0018 sec
54 19:23:48 CREATE TABLE Advisor(advisorID INT NOT NULL, programID VARCHAR(6) NOT NULL, yearOftudy INT,noOfStudents IN... 0 row(s) affected 0.017 sec
55 19:23:48 INSERT INTO Advisor(advisorID, programID,yearOftudy, noOfStudents) VALUES("1001","SWE","1","128") 1 row(s) affected 0.0018 sec
56 19:24:03 CREATE TABLE BankInfo(bankName VARCHAR(20), accountName VARCHAR(30) NOT NULL, accountNo BIGINT NOT N... 0 row(s) affected 0.010 sec
57 19:24:03 INSERT INTO bankInfo(bankName, accountName, accountNo, accountCurrency,IBAN, swiftCode, paymentDescription)... 1 row(s) affected 0.0022 sec
58 19:24:03 INSERT INTO bankInfo(bankName, accountName, accountNo, accountCurrency,IBAN, swiftCode, paymentDescription)... 1 row(s) affected 0.00096 sec
59 19:24:11 CREATE TABLE Finance(epokaID INT NOT NULL UNIQUE, academicYear INT NOT NULL, debt INT NOT NULL, currency... 0 row(s) affected 0.015 sec
60 19:24:11 INSERT INTO Finance(epokaID,academicYear,debt, currency,total) VALUES ("020321090","2023","1750","EUR","1750") 1 row(s) affected 0.0018 sec
61 19:24:21 CREATE TABLE Document(records INT, numberCount INT AUTO_INCREMENT UNIQUE, epokaID INT NOT NULL,docum... 0 row(s) affected 0.019 sec
62 19:24:21 INSERT INTO Document(records, epokaID, documentType,requestID, requestionDate, copyRequested, price, payment,... 1 row(s) affected 0.0026 sec
63 19:24:37 CREATE TABLE Grades(letterGrade VARCHAR(2), gpaValue NUMERIC(3,2), points INT, gradeDescription VARCHAR(20)... 0 row(s) affected 0.010 sec
64 19:24:37 ALTER TABLE Grades MODIFY COLUMN points VARCHAR(6) 0 row(s) affected Records: 0 Duplicates: 0 Warnings... 0.017 sec
65 19:24:37 INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded) VALUES("AA","4.... 1 row(s) affected 0.00074 sec
66 19:24:37 INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded) VALUES("DC","1.... 1 row(s) affected 0.00087 sec
67 19:24:37 INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded) VALUES("BA","3.... 4 row(s) affected Records: 4 Duplicates: 0 Warnings... 0.00096 sec
68 19:24:37 INSERT INTO Grades(letterGrade, gpaValue, points, gradeDescription, albGradeSystem, avgIncluded) VALUES("DD","1.... 4 row(s) affected Records: 4 Duplicates: 0 Warnings... 0.00085 sec
69 19:24:46 CREATE TABLE AcademicCalendar(eventName VARCHAR(40),peroidOfTime VARCHAR(20), eventdescription VARCHA... 0 row(s) affected 0.011 sec
70 19:24:46 INSERT INTO academiccalendar(eventName, peroidOfTime) VALUES("Fall Semester Add and Drop","October 16-20, 20... 1 row(s) affected 0.0018 sec
71 19:24:46 INSERT INTO academiccalendar(eventName, peroidOfTime) VALUES("Fall Semester FINAL EXAMS","Jan 22-Feb 3,202... 1 row(s) affected 0.0011 sec
72 19:24:46 INSERT INTO academiccalendar(eventName,peroidOfTime) VALUES("Spring Break","February 05-18, 2024"),("Spring... 3 row(s) affected Records: 3 Duplicates: 0 Warnings... 0.0014 sec
73 19:24:53 CREATE TABLE Transport(routeID INT NOT NULL , route VARCHAR(20) NOT NULL, departureTime DATETIME NOT NUL... 0 row(s) affected 0.011 sec
74 19:24:53 ALTER TABLE TRANSPORT MODIFY COLUMN routestatus VARCHAR(15) 0 row(s) affected Records: 0 Duplicates: 0 Warnings... 0.0077 sec
75 19:24:53 INSERT INTO Transport(routeID, route,departureTime, routeStatus, busStation, stationTime) VALUES("001","Tirana-Ca... 1 row(s) affected 0.00098 sec
76 19:24:53 INSERT INTO Transport(routeID, route,departureTime, routeStatus, busStation, stationTime) VALUES("002","Campus-D... 1 row(s) affected 0.00085 sec
77 19:24:59 CREATE TABLE TransportReservation(epokaID INT NOT NULL, routeID INT NOT NULL UNIQUE, FOREIGN KEY(epokaID)... 0 row(s) affected 0.017 sec
78 19:24:59 INSERT INTO TransportReservation(epokaID,routeID) VALUES("020321090","001"),("02052286","002") 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0018 sec
79 19:25:04 CREATE TABLE Company(companyID INT NOT NULL UNIQUE, companyName VARCHAR(20) NOT NULL, companyDesc... 0 row(s) affected 0.012 sec
80 19:25:04 INSERT INTO Company(companyID,companyName) VALUES("001","Intesa Sanpaolo Bank") 1 row(s) affected 0.0018 sec
81 19:25:04 INSERT INTO Company(companyID,companyName) VALUES("002","Neptun") 1 row(s) affected 0.00088 sec
82 19:25:04 INSERT INTO Company(companyID,companyName) VALUES("003","Deloite") 1 row(s) affected 0.0011 sec
83 19:25:04 INSERT INTO Company(companyID,companyName) VALUES("004","Vodafone"),("005","Albsig") 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.00081 sec
84 19:25:17 CREATE TABLE Application(epokaID INT NOT NULL,applicationID INT NOT NULL UNIQUE, appliactionName VARCHAR(... 0 row(s) affected 0.019 sec
Query Completed


Local instance 3306 - Warning - not supported
Administration
Schemas
SCHEMAS
Filter objects
sys
Object Info
Session
No object selected
Action Output
Time
Action
Response
Duration / Fetch Time
82 19:25:04 INSERT INTO Company(companyID,companyName) VALUES("003","Deloite") 1 row(s) affected 0.0011 sec
83 19:25:04 INSERT INTO Company(companyID,companyName) VALUES("004","Vodafone"),("005","Albsig") 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.00081 sec
84 19:25:17 CREATE TABLE Application(epokaID INT NOT NULL,applicationID INT NOT NULL UNIQUE, appliactionName VARCHAR(... 0 row(s) affected 0.019 sec
85 19:25:17 INSERT INTO Application(epokaID,applicationID, appliactionName, position, companyID, dateOfApplication) VALUES("0... 1 row(s) affected 0.0017 sec
86 19:25:17 INSERT INTO Application(epokaID,applicationID, appliactionName, position, companyID, dateOfApplication) VALUES("0... 1 row(s) affected 0.00091 sec
87 19:25:26 CREATE TABLE ProfessionalPractice(epokaID INT NOT NULL , applicationID INT NOT NULL UNIQUE, companyID INT N... 0 row(s) affected 0.019 sec
88 19:25:26 INSERT INTO ProfessionalPractice(epokaID,applicationID,companyID) VALUES("01042055","0155","004"),("020522110... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0019 sec
89 19:25:33 CREATE TABLE Class(classID VARCHAR(4) NOT NULL UNIQUE, building VARCHAR(1), floorLevel INT, classNumber INT,... 0 row(s) affected 0.011 sec
90 19:25:33 INSERT INTO Class(classID,building, floorLevel, classNumber) VALUES("A128","A", "1","28") 1 row(s) affected 0.0019 sec
91 19:25:33 INSERT INTO Class(classID,building, floorLevel, classNumber) VALUES("A005","A","0", "5") 1 row(s) affected 0.00092 sec
92 19:25:33 INSERT INTO Class(classID,building, floorLevel, classNumber,classDescription) VALUES("E011","E","0", "11","Lab 1") 1 row(s) affected 0.00083 sec
93 19:25:40 CREATE TABLE Erasmus(erasmusCode INT NOT NULL UNIQUE, universityName VARCHAR(20) NOT NULL, state VARC... 0 row(s) affected 0.012 sec
94 19:25:40 INSERT INTO erasmus(erasmusCode, universityName, state, studyProgram, yearOfStudy,season) VALUES("2513","Aalb... 1 row(s) affected 0.0019 sec
95 19:25:40 INSERT INTO erasmus(erasmusCode, universityName, state, studyProgram, yearOfStudy,season) VALUES("2516","Aalt... 1 row(s) affected 0.00093 sec
96 19:25:47 CREATE TABLE ErasmusApplications(applicationID INT NOT NULL UNIQUE, erasmusCode INT NOT NULL, epokaID INT... 0 row(s) affected 0.019 sec
97 19:25:47 INSERT INTO ErasmusApplications(applicationID,erasmusCode,epokaID) VALUES("0155","2513","01042055") 1 row(s) affected 0.0020 sec
98 19:25:57 CREATE TABLE Clubs(clubID INT NOT NULL UNIQUE, clubName VARCHAR(20) NOT NULL, link VARCHAR(30),nrOfStu... 0 row(s) affected 0.011 sec
99 19:25:57 INSERT INTO Clubs(clubid,clubName, nrOfStudents) VALUES("001","GDSC","100") 1 row(s) affected 0.0019 sec
100 19:25:57 INSERT INTO Clubs(clubid,clubName, nrOfStudents) VALUES("002","Programming","80") 1 row(s) affected 0.0012 sec
101 19:25:57 INSERT INTO Clubs(clubid,clubName, nrOfStudents) VALUES("003","Sports","70") 1 row(s) affected 0.00096 sec
102 19:26:06 CREATE TABLE ClubParticipants(epokaID INT NOT NULL,clubID INT, studentRole VARCHAR(8), FOREIGN KEY(epokaID)... 0 row(s) affected 0.025 sec
103 19:26:06 INSERT INTO ClubParticipants(epokaID,clubID,StudentRole) VALUES ("020522110","002","Member"),("02052286","001... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0015 sec
104 19:26:21 CREATE TABLE ClubsEvents(clubID INT NOT NULL, eventName VARCHAR(20) NOT NULL, eventDate DATE, startTime T... 0 row(s) affected 0.014 sec
105 19:26:21 INSERT INTO clubsevents(clubID, eventName, eventDate, startTime, endTime) VALUES("002","AI","2023-12-17","11:30"... 1 row(s) affected 0.0017 sec
106 19:26:21 INSERT INTO clubsevents(clubID, eventName, eventDate, startTime, endTime) VALUES("001","Firebase","2024-01-11","... 1 row(s) affected 0.00091 sec
107 19:26:30 CREATE TABLE Timetable(season VARCHAR(10) NOT NULL UNIQUE, StudentType VARCHAR(15), programID VARCHA... 0 row(s) affected 0.024 sec
108 19:26:37 CREATE TABLE PersonalisedTimetable(epokaID INT NOT NULL, dayOfTheWeek VARCHAR(10) NOT NULL, courseID V... 0 row(s) affected 0.020 sec
109 19:26:37 INSERT INTO PersonalisedTimeTable(epokaID,dayOfTheWeek,courseID,startTime,endTime,lecturerID,classID) VALUES(... 1 row(s) affected 0.0016 sec
110 19:26:37 INSERT INTO PersonalisedTimeTable(epokaID,dayOfTheWeek,courseID,startTime,endTime,lecturerID,classID) VALUES(... 4 row(s) affected Records: 4 Duplicates: 0 Warnings... 0.0011 sec
111 19:26:44 CREATE TABLE CourseSelection(epokaID INT NOT NULL, courseType VARCHAR(13) NOT NULL, courseID VARCHAR(6)... 0 row(s) affected 0.022 sec
112 19:26:44 INSERT INTO CourseSelection(epokaID,courseType,courseID,advisorID,selected,confirmationDate) VALUES("0205228... 2 row(s) affected Records: 2 Duplicates: 0 Warnings... 0.0020 sec
113 19:26:44 INSERT INTO CourseSelection(epokaID,courseType,courseID,advisorID,selected,confirmationDate) VALUES("0205228... 1 row(s) affected 0.0011 sec
114 19:26:52 CREATE TABLE Attendance(epokaID INT NOT NULL, records INT AUTO_INCREMENT UNIQUE , courseID VARCHAR(6)... 0 row(s) affected 0.018 sec
115 19:26:52 INSERT INTO Attendance(epokaID,courseID,nrOfWeek,topic,hourtype,attendedHours,totalHours,courseHrsDate) VALU... 10 row(s) affected Records: 10 Duplicates: 0 Warnin... 0.0022 sec
116 19:27:00 CREATE TABLE Survey(programID VARCHAR(6) NOT NULL, epokaID INT NOT NULL,courseID VARCHAR(6) NOT NULL,... 0 row(s) affected 0.024 sec
117 19:27:05 CREATE TABLE ResitExam(epokaID INT NOT NULL, courseID VARCHAR(6) NOT NULL, selectStatus VARCHAR(15) NOT... 0 row(s) affected 0.021 sec
118 19:27:05 INSERT INTO ResitExam(epokaID,courseID,selectStatus) VALUES("020522110","SWE101","SELECTED"),("02052286","... 3 row(s) affected Records: 3 Duplicates: 0 Warnings... 0.0025 sec
119 19:27:12 CREATE TABLE AdditionalExams(epokaID INT NOT NULL,academicYear INT NOT NULL,courseID VARCHAR(6) NOT NU... 0 row(s) affected 0.018 sec
120 19:27:12 INSERT INTO AdditionalExams(epokaID, academicYear,courseID,examStatus) VALUES("01042055","2024","BUS101","p... 1 row(s) affected 0.0018 sec
121 19:27:17 CREATE TABLE ThirdSemester(epokaID INT NOT NULL, courseID VARCHAR(6) NOT NULL, courseStatus VARCHAR(10)... 0 row(s) affected 0.019 sec
122 19:27:17 INSERT INTO ThirdSemester(epokaID, courseID, courseStatus) VALUES("01042055","BUS101","Selected") 1 row(s) affected 0.0017 sec
Query Completed


Output
Action Output
Time Action Message Duration / Fetch
5 21:49:26 SELECT s.epokaID, s.StudentName, a.courseID, (a.attendedHours / a.totalHours) * 100 AS attendancePercentage F... 6 row(s) returned 0.000 sec / 0.000 sec
6 21:50:40 SELECT s.epokaID, s.StudentName, SUM(a.attendedHours) AS totalAttendedHours, SUM(a.totalHours) ... 1 row(s) returned 0.000 sec / 0.000 sec
7 22:02:34 SELECT c.programID, c.courseID, AVG(sc.grade) AS average FROM Course c JOIN StudentCourses sc ON c.courseI... 7 row(s) returned 0.000 sec / 0.000 sec
8 22:06:16 SELECT tc.lecturerID,l.lecturerName,l.surname,SUM(noOfStudents) AS totalNrOfStudents FROM teachedCourses tc J... 6 row(s) returned 0.016 sec / 0.000 sec
9 22:12:49 SELECT a.programID,le.primaryEmail FROM lecturerEmail le JOIN advisor a ON le.lecturerID=a.advisorID GROUP BY ... 2 row(s) returned 0.015 sec / 0.000 sec
10 22:12:55 SELECT officeNumber, COUNT(lecturerID) AS lecturerCount FROM Office GROUP BY officeNumber LIMIT 0, 1000 3 row(s) returned 0.015 sec / 0.000 sec
11 22:13:47 SELECT s.epokaID, s.programID, sc.courseID, c.courseName FROM Student s JOIN StudentCourses sc ON s.epokaI... 7 row(s) returned 0.016 sec / 0.000 sec
12 22:17:00 SELECT l.lecturerID, l.lecturerName,l.surname,s.courseID ,AVG(s.evaluation) AS avgEvaluationScore FROM Lecturer l... 1 row(s) returned 0.016 sec / 0.000 sec
13 22:19:12 SELECT courseID, lecturerID, AVG(evaluation) AS AvgEvaluation FROM Survey GROUP BY courseID, lecturerID LIM... 2 row(s) returned 0.000 sec / 0.000 sec
14 22:20:56 SELECT a.epokaID, s.StudentName, a.courseID, SUM(a.attendedHours) AS totalAttendedHours, SUM(a.tota... 1 row(s) returned 0.000 sec / 0.000 sec
15 22:22:46 SELECT s.epokaID, s.StudentName, SUM(a.attendedHours) AS totalAttendedHours, SUM(a.totalHours) ... 1 row(s) returned 0.000 sec / 0.000 sec


Output
Action Output
Time Action Message Duration / Fetch
18 22:24:35 SELECT s.epokaID, s.StudentName,a.courseID, SUM(a.attendedHours) AS totalAttendedHours,SUM(a.totalHours) A... 1 row(s) returned 0.000 sec / 0.000 sec
19 22:28:10 SELECT t.lecturerID, t.classID, COUNT(*) AS teachingFrequency FROM Timetable t JOIN Lecturer l ON t.lecturerID = l... 1 row(s) returned 0.047 sec / 0.000 sec
20 22:29:52 SELECT c.courseID, c.courseName, COUNT(DISTINCT t.lecturerID) AS numLecturers FROM Course c JOIN Teache... 2 row(s) returned 0.000 sec / 0.000 sec
21 22:31:20 SELECT c.clubName, COUNT(cp.epokaID) AS numParticipants FROM Clubs c LEFT JOIN ClubParticipants cp ON c.... 3 row(s) returned 0.047 sec / 0.000 sec
22 22:33:11 SELECT pt.dayOfTheWeek, pt.startTime,pt.endTime,pt.courseID FROM PersonalisedTimetable pt JOIN Class c ON pt.... 1 row(s) returned 0.047 sec / 0.000 sec
23 22:34:04 SELECT s.epokaID,s.StudentName,s.programID,p.programName,MAX(sc.grade) AS topCourseGrade FROM Student s... 5 row(s) returned 0.000 sec / 0.000 sec
24 22:36:25 SELECT DISTINCT s.epokaID, s.studentName, s.surname FROM Student s JOIN Attendance a ON s.epokaID = a.ep... 1 row(s) returned 0.000 sec / 0.000 sec
25 22:37:22 SELECT s.studentName,t.route FROM transportreservation tr JOIN student s ON tr.epokaID=s.epokaID JOIN transpor... 2 row(s) returned 0.046 sec / 0.000 sec
26 22:38:29 SELECT s.StudentName, s.surname FROM Student s JOIN StudentAddres sa ON s.epokaID = sa.epokaID WHERE s... 1 row(s) returned 0.047 sec / 0.000 sec
27 22:40:30 INSERT INTO TeacheDCourses(lecturerID, courseID,semesterYear,lecturerRole) VALUES("1004","CEN109","1-2023... 1 row(s) affected 0.047 sec
28 22:40:45 SELECT l.* FROM Lecturer l JOIN TeachedCourses tc ON l.lecturerID = tc.lecturerID GROUP BY l.lecturerID, tc.seme... 1 row(s) returned 0.000 sec / 0.000 sec

References

Zanini, Antonello. “Comparing Er Diagrams, ER Models and Relational Schemas.” *DbVisualizer*, DbVis Software AB, 11 Sept. 2023, www.dbvis.com/thetable/er-diagrams-vs-er-models-vs-relational-schemas/.

Phil Vuollet. “Database Performance Improvement - How-to in 4 Easy Steps.” *Raygun Blog*, Raygun Blog, 15 Jan. 2020, raygun.com/blog/database-performance-improvements/.