**Czech University of Life Sciences**

**Faculty of Economics and Management**



**Database Systems project:**

**Database for University Students Grades**

**Autor:**

**Aygun Guliyeva**, Bachelor’s

FEM Informatics B-INFOAP

Identification number: 224076

pres [term 3, year 2]

Study group no.: 1

**Introduction**

This work designs a database that contains information about Students’ University Grades from different subjects at the university. It intends to capture basic information needed for monitoring the results. And it’s based on following assumptions:

* One subject can be taught in different study programs and during different study years
* Each Subject has only one Teacher and the corresponding exam is graded by the same teacher
* An Address can be same for two students or one student and one teacher if they are family members but their contacts should be different
* Students table always show the current information about the student, but Grades table consist of historical data.

This project does not present multiple credit test grades from different subjects, also the information about faculties which each subject/program belongs to, it also ignores several important attributes that would be required in real world system.

**Outline**

[1. Possible use cases for the model 3](#_Toc532980131)

[2. Entity relationship diagrams 3](#_Toc532980132)

[2.1. Conceptual ERD 3](#_Toc532980133)

[2.2. Logical ERD 4](#_Toc532980134)

[2.3. Physical ERD 4](#_Toc532980135)

[3. SQL Implementation 5](#_Toc532980136)

[3.1. DDL : Defining the database objects 5](#_Toc532980137)

[3.2. DML: Inserting the data (examples) 6](#_Toc532980138)

[3.3. SQL Queries 8](#_Toc532980139)

# Possible use cases for the model

* See all student grades together with the exam date, the student’s program, study year and term and the subject teacher as the examiner
* Calculate the average student grade for each subject for the whole study period or in a particular year/semester
* Retrieve the list of students which failed in any subject exam
* Find information about teachers and calculate the workload of the teacher based on his/her subject’s number of credits.
* Identifying same family members among students and teachers (based on their address)

# Entity relationship diagrams

Following section captures the proposed structure of database using entity relationship diagram. The diagrams were created in <https://www.lucidchart.com/>.

## Conceptual ERD

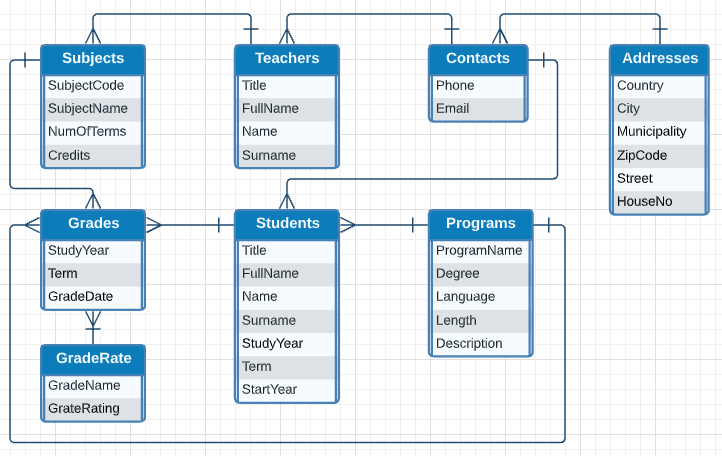


Figure 1: Conceptual ERD

## Logical ERD

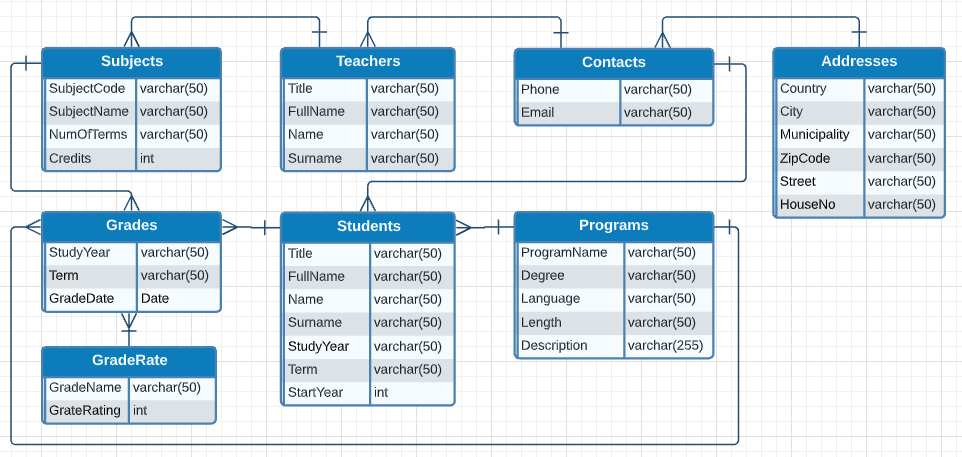


Figure 2: Logical ERD

## Physical ERD

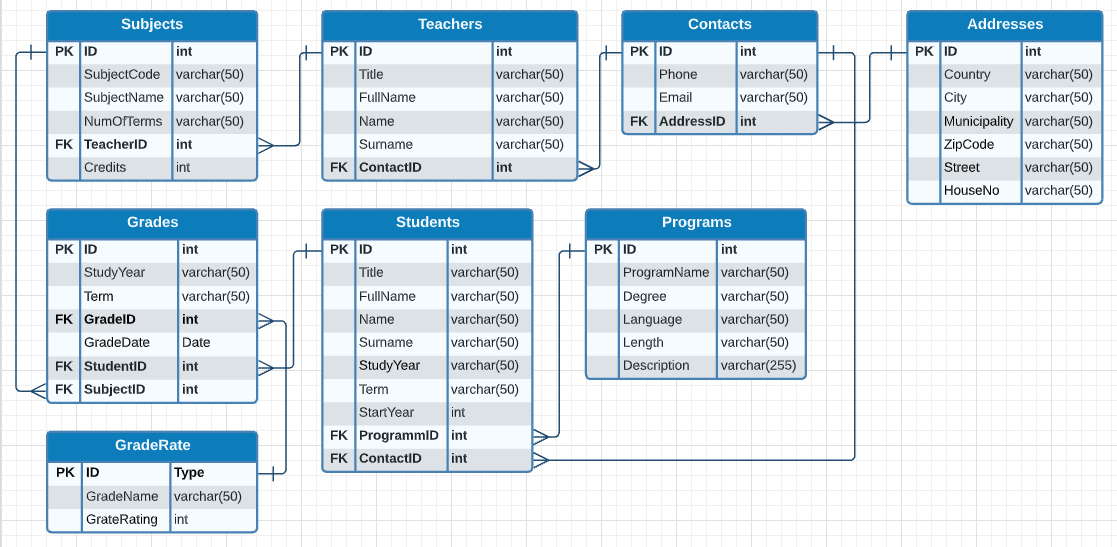


Figure 3: Physical ERD

# SQL Implementation

The database was implemented in SQL Server, which uses Microsoft SQL Server Management Studio 18 - 64bit Production as a DBMS.

## DDL : Defining the database objects

CREATE TABLE Students

(ID INT PRIMARY KEY, Title varchar(50) NOT NULL, FullName varchar(50) NOT NULL, Name varchar(50) NOT NULL, Surname varchar(50) NOT NULL, StudyYear varchar(50) NOT NULL, Term varchar(50) NOT NULL, StartYear INT NOT NULL, ProgramID INT NOT NULL, ContactID INT NOT NULL);

CREATE TABLE Teachers

(ID INT PRIMARY KEY, Title varchar(50) NOT NULL, FullName varchar(50) NOT NULL, Name varchar(50) NOT NULL, Surname varchar(50) NOT NULL, ContactID INT NOT NULL);

CREATE TABLE Programs

(ID INT PRIMARY KEY, ProgramName varchar(50) NOT NULL, Degree varchar(50) NOT NULL, Language varchar(50) NOT NULL, Length varchar(50) NOT NULL, Description varchar(255));

CREATE TABLE Subjects

(ID INT PRIMARY KEY, SubjectCode varchar(50) NOT NULL, SubjectName varchar(50) NOT NULL, NumOfTerms INT NOT NULL, TeacherID INT NOT NULL, Credits INT NOT NULL);

CREATE TABLE Contacts

(ID INT PRIMARY KEY, Phone varchar(50) NOT NULL, Email varchar(50) NOT NULL, AddressID INT NOT NULL);

CREATE TABLE Addresses

(ID INT PRIMARY KEY, Country varchar(50) NOT NULL, City varchar(50) NOT NULL, Municipality varchar(50) NOT NULL, ZipCode varchar(50) NOT NULL, Street varchar(50) NOT NULL, HouseNo varchar(50) NOT NULL);

CREATE TABLE Grades

(ID INT PRIMARY KEY, StudyYear varchar(50) NOT NULL, Term varchar(50) NOT NULL, GradeID INT NOT NULL, GradeDate Date NOT NULL, StudentID INT NOT NULL, SubjectID INT NOT NULL);

CREATE TABLE GradeRate

(ID INT PRIMARY KEY, GradeName varchar(50) NOT NULL, GradeRating INT NOT NULL);

**Constraints:**

ALTER TABLE Students

ADD CONSTRAINT StudentProgram FOREIGN KEY (ProgramID) REFERENCES Programs (ID);

ALTER TABLE Students

ADD CONSTRAINT StudentContact FOREIGN KEY (ContactID) REFERENCES Contacts (ID);

ALTER TABLE Teachers

ADD CONSTRAINT TeacherContact FOREIGN KEY (ContactID) REFERENCES Contacts (ID);

ALTER TABLE Contacts

ADD CONSTRAINT COntactAddress FOREIGN KEY (AddressID) REFERENCES Addresses(ID);

ALTER TABLE Subjects

ADD CONSTRAINT SubjectTeacher FOREIGN KEY (TeacherID) REFERENCES Teachers (ID);

ALTER TABLE Grades

ADD CONSTRAINT GradeStudent FOREIGN KEY (StudentID) REFERENCES Students (ID);

ALTER TABLE Grades

ADD CONSTRAINT GradeSubject FOREIGN KEY (SubjectID) REFERENCES Subjects (ID);

ALTER TABLE Grades

ADD CONSTRAINT GradeRates FOREIGN KEY (GradeID) REFERENCES GradeRate(ID);

## DML: Inserting the data (examples)

INSERT INTO Programs (ID, ProgramName, Degree, Language, Length, Description) VALUES

(1, 'INFOAP', 'Bachelors', 'English', '3 years', 'Informatics')

, (2, 'COMSCI', 'Bachelors', 'English', '3 years', 'ComputerScience')

Table: Programs



INSERT INTO Addresses (ID, Country, City, Municipality, ZipCode, Street, HouseNo) VALUES

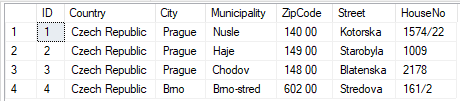
(1, 'Czech Republic', 'Prague', 'Nusle', '140 00', 'Kotorska', '1574/22')

, (2, 'Czech Republic', 'Prague', 'Haje', '149 00', 'Starobyla', '1009')

, (3, 'Czech Republic', 'Prague', 'Chodov', '148 00', 'Blatenska', '2178')

, (4, 'Czech Republic', 'Brno', 'Brno-stred', '602 00', 'Stredova', '161/2')

Table: Addresses



INSERT INTO Contacts (ID, Phone, Email, AddressID) VALUES

(1, '+420775343656', 'Aygun.Guliyeva@studenti.czu.cz', 1)

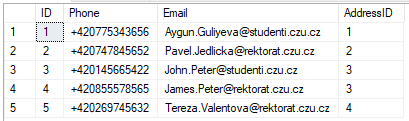
, (2, '+420747845652', 'Pavel.Jedlicka@rektorat.czu.cz', 2)

, (3, '+420145665422', 'John.Peter@studenti.czu.cz', 3)

, (4, '+420855578565', 'James.Peter@rektorat.czu.cz', 3)

, (5, '+420269745632', 'Tereza.Valentova@rektorat.czu.cz', 4)

Table: Contacts

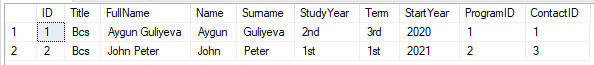


INSERT INTO Students (ID, Title, FullName, Name, Surname, StudyYear, Term, StartYear, ProgramID, ContactID) VALUES

(1, 'Bcs', 'Aygun Guliyeva', 'Aygun', 'Guliyeva', '2nd', '3rd', 2020, 1, 1)

, (2, 'Bcs', 'John Peter', 'John', 'Peter', '1st', '1st', 2021, 2, 3)

Table: Students



INSERT INTO Teachers (ID, Title, FullName, Name, Surname, ContactID) VALUES

(1, 'Prof.', 'Pavel Jedlicka', 'Pavel', 'Jedlicka', 2)

, (3, 'Prof.', 'James Peter', 'James', 'Peter', 4)

, (2, 'Prof.', 'Tereza Valentova', 'Tereza', 'Valentova', 5)

Table: Teachers



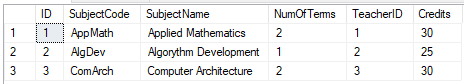
INSERT INTO Subjects (ID, SubjectCode, SubjectName, NumOfTerms, Credits, TeacherID) VALUES

(1, 'AppMath', 'Applied Mathematics', 2, 30, 1)

, (2, 'AlgDev', 'Algorythm Development', 1, 25, 2)

, (3, 'ComArch', 'Computer Architecture', 2, 30, 3)

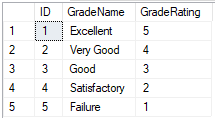
Table: Subjects



INSERT INTO GradeRate (ID, GradeName, GradeRating) VALUES

(1, 'Excellent', 5), (2, 'Very Good', 4), (3, 'Good', 3), (4, 'Satisfactory', 2), (5, 'Failure', 1)

Table: GradeRate



INSERT INTO Grades (ID, StudyYear, Term, GradeID, GradeDate, StudentID, SubjectID) VALUES

(1, '1st', '1st', 2, '2020-11-29', 1, 1)

, (2, '1st', '1st', 1, '2020-12-15', 1, 2)

, (3, '1st', '1st', 1, '2020-12-28', 1, 3)

, (4, '1st', '2nd', 2, '2021-05-25', 1, 1)

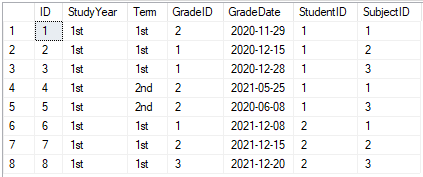
, (5, '1st', '2nd', 2, '2020-06-08', 1, 3)

, (6, '1st', '1st', 1, '2021-12-08', 2, 1)

, (7, '1st', '1st', 2, '2021-12-15', 2, 2)

, (8, '1st', '1st', 3, '2021-12-20', 2, 3)

Table: Grades



## SQL Queries

* **See all student grades together with the exam date, the student’s program, study year and term and the subject teacher as the examiner**

SELECT st.FullName AS StudentName, pr.ProgramName, st.StudyYear AS CurrentYear, st.Term AS CurrentTerm, sb.SubjectName, gr.StudyYear, gr.Term, rt.GradeName, gr.GradeDate, tc.FullName AS Examiner FROM Grades gr

LEFT JOIN GradeRate rt ON gr.GradeID = rt.ID

LEFT JOIN Students st ON gr.StudentID = st.ID

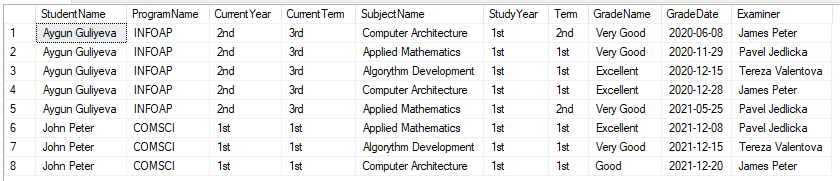
LEFT JOIN Subjects sb ON gr.SubjectID = sb.ID

LEFT JOIN Teachers tc ON sb.TeacherID = tc.ID

LEFT JOIN Programs pr ON st.ProgramID = pr.ID

ORDER BY StudentName, GradeDate

Query sample result



* **Calculate the average student grade for each subject for the whole study period or in a particular year/semester**

SELECT st.FullName AS StudentName, sb.SubjectName, AVG(rt.GradeRating) AS AvgGrade FROM Grades gr

LEFT JOIN GradeRate rt ON gr.GradeID = rt.ID

LEFT JOIN Subjects sb ON gr.SubjectID = sb.ID

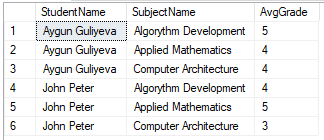
LEFT JOIN Students st ON gr.StudentID = st.ID

LEFT JOIN Programs pr ON st.ProgramID = pr.ID

GROUP BY st.FullName, sb.SubjectName

ORDER BY StudentName

Query sample result



* **Retrieve the list of students which failed in any subject exam**

SELECT st.FullName AS StudentName, sb.SubjectName, gr.StudyYear, gr.Term, rt.GradeName, gr.GradeDate, tc.FullName AS Examiner FROM Grades gr

LEFT JOIN GradeRate rt ON gr.GradeID = rt.ID

LEFT JOIN Students st ON gr.StudentID = st.ID

LEFT JOIN Subjects sb ON gr.SubjectID = sb.ID

LEFT JOIN Teachers tc ON sb.TeacherID = tc.ID

WHERE GradeName LIKE '%fail%'

ORDER BY StudentName, GradeDate

Query sample result (no failed students)



* **Find information about teachers and calculate the workload ranking of the teacher based on his/her subject’s number of credits.**

SELECT tc.FullName AS TeacherName, sb.SubjectName, sb.NumOfTerms\*sb.Credits AS YearlyCredits, DENSE\_RANK() OVER(ORDER BY sb.NumOfTerms\*sb.Credits DESC) AS CreditsRank, cn.Phone, cn.Email, ad.City

FROM Subjects sb

LEFT JOIN Teachers tc ON sb.TeacherID = tc.ID

LEFT JOIN Contacts cn ON tc.ContactID = cn.ID

LEFT JOIN Addresses ad ON cn.AddressID = ad.ID

ORDER BY tc.FullName

Query sample result



* **Identifying same family members among students and teachers (based on their address)**

SELECT ppl.Title, ppl.FullName, cn.Phone, cn.Email, ad.ID, ad. Country, ad.City, ad.Municipality, ad.Street, ad.HouseNo, ad.ZipCode

FROM ( SELECT Title, FullName, ContactID FROM Students

UNION ALL

SELECT Title, FullName, ContactID FROM Teachers) ppl

LEFT JOIN Contacts cn ON ppl.ContactID = cn.ID

LEFT JOIN Addresses ad ON cn.AddressID = ad.ID

INNER JOIN

(SELECT cn.AddressID, COUNT(cn.AddressID) AS Cnt

FROM Contacts cn

GROUP BY cn.AddressID

HAVING COUNT(cn.AddressID) > 1) fml ON cn.AddressID = fml.AddressID

Query sample result



# Conclusion

This project contains a basic proposal for a database, which can be used in a Student Grades Management System. It contains definitions of essential database objects, and examples of possible use cases realised in the form of SQL queries. This project is an essential part of a possible implementation of full-scale systems.