**Akademia Finansów i Biznesu Vistula**

**Wydział Informatyki, Grafiki i Architektury**

**Kierunek studiów Informatyka**

**Nazarii Lozynskyi**

Numer albumu: 72102

**PROJEKTOWANIE BAZY DANYCH NA TEMAT „SYSTEM ZARZĄDZANIA OCENAMI I FREKWENCJĄ UCZNIÓW”**

napisana pod kierunkiem

Promotor – Mgr inż. Mirosław Lisiecki

Warszawa, 2025

**Spis treści**

[**1** **Cele i założenia projektu** 3](#_Toc189491025)

[**2** **Model diagramu koncepcyjnego** 4](#_Toc189491026)

[**2.1** **Cele i założenia diagramu koncepcyjnego** 4](#_Toc189491027)

[**2.2** **Byty (Entity)** 4](#_Toc189491028)

[**2.3** **Relacje lub interakcje między jednostkami** 5](#_Toc189491029)

[**3** **Model ERD** 6](#_Toc189491030)

[**4** **Schemat fizyczny** 7](#_Toc189491031)

[**5** **Procedury i triggery** 10](#_Toc189491032)

[**5.1** **Procedura dla dopisania i aktualizacji danych** 10](#_Toc189491033)

[**5.2** **Procedura dla kasowania danych** 12](#_Toc189491034)

[**5.3** **Wyzwalacz dla obliczenia oceny semestralnej** 13](#_Toc189491035)

[**6** **Testowy zestaw danych** 15](#_Toc189491036)

[**7** **Streszczenie** 23](#_Toc189491037)

1. **Cele i założenia projektu**

Tematem projektowanej bazy danych jest: „System zarządzania ocenami i frekwencją uczniów”. Celem tej bazy danych jest ułatwienie i zautomatyzowanie procesu wystawiania ocen i frekwencji w ośrodku edukacyjnym.

Kluczowe założenia i ograniczenia projektowanej bazy danych opisano poniżej:

1. **Użytkownicy systemu:**

* Wszyscy użytkownicy (studenci, nauczyciele, administratorzy) znajdują się w jednej tabeli Users;
* Każdy użytkownik posiada unikalny identyfikator, a także login i hasło dla dostępu do systemu, adres e-mail oraz numer kontaktowy;
* Każdy użytkownik ma swój status (student, nauczyciel, administrator);

1. **Studenci i grupy:**

* Każdy student należy do określonej grupy;
* Każdy student ma swój unikalny numer Albumu i okres studiów (daty rozpoczęcia i zakończenia studiów);

1. **Przedmioty i plan zajęć:**

* Plan zajęć zawiera informacje o grupie, w której przedmiot jest prowadzony, nazwę przedmiotu, nauczyciela prowadzącego przedmiot w odpowiedniej grupie oraz dzień tygodnia, w którym przedmiot ma się odbyć;

1. **Oceny:**

* Wszystkie oceny znajdują się w jednej tabeli i zawierają informacje o uczniu, przedmiocie, dacie otrzymania i nauczycielu, który wystawił ocenę.
* Oceny końcowe za semestr znajdują się w oddzielnej tabeli i zawierają informacje o uczniu, przedmiocie oraz numerze semestru;
* Oceny końcowe liczą się automatycznie za pomocą wyzwalacza;

1. **Frekwencja:**

* Oddzielna tabela zawiera dane dotyczące nieobecności uczniów, w tym datę, przedmiot, powód i nauczyciela, który zaznaczył nieobecność;
* Istnieją dwa powody nieobecności: z uzasadnionym powodem i bez uzasadnionego powodu;

1. **Ograniczenia:**

* Każdy student może być zapisany tylko do jednej grupy;
* Nie jest możliwe zapisanie oceny bez wpisania odpowiedniego studenta, nauczyciela i przedmiotu;
* Nieobecność studenta jest rejestrowana tylko wtedy, gdy jest on oficjalnie zapisany na kurs;
* Każdy użytkownik musi mieć unikalny login;

1. **Model diagramu koncepcyjnego**
2. **Cele i założenia diagramu koncepcyjnego**

Cele systemu:

* Przedstawienie jasnej reprezentacji systemu zarządzania ocenami i frekwencją studentów;
* Identyfikacja kluczowych relacji i zależności między danymi;
* Umożliwienie analizy struktury systemu i wykrycia potencjalnych problemów;
* Wsparcie dla dalszego projektowania poprzez zdefiniowanie kluczowych bytów i relacji;

1. **Byty (Entity)**

* **Użytkownicy (**Users**)** – Tabela przechowuje dane wszystkich użytkowników systemu, w tym studentów, nauczycieli i administratorów. Każdy użytkownik ma swój unikalny UserID, login, hasło, e-mail i numer telefonu.
* **Studenci (**Students**)** – Podzbiór użytkowników, którzy są zapisani do określonych grup (Groups). Zawiera datę rozpoczęcia i zakończenia nauki oraz numer albumu.
* **Grupy (**Groups**)** – Reprezentuje przypisanie studentów do danej grupy. Każdy student należy do jednej grupy.
* **Przedmioty (**Subjects**)** – Lista przedmiotów, które są nauczane.
* **Harmonogram (**Schedule**)** – Definiuje, kiedy i jakie zajęcia odbywają się dla danej grupy, określając przedmiot, nauczyciela oraz dzień tygodnia.
* **Oceny (**Grades**)** – Rejestr ocen studentów dla konkretnych przedmiotów. Zawiera informacje o wystawionej ocenie, dacie jej wpisania oraz nauczycielu, który jej udzielił.
* **Oceny końcowe (**Final\_Grades**)** – Podsumowanie ocen w semestrze dla każdego studenta z konkretnego przedmiotu.
* **Nieobecności (**Absence**)** – Przechowuje informacje o nieobecnościach studentów, z określeniem przedmiotu, daty oraz powodu.
* **Powody nieobecności (**Reasons**)** – Lista możliwych przyczyn nieobecności.
* **Status użytkownika** (Status) – Określa rolę użytkownika w systemie: student, nauczyciel lub administrator.

1. **Relacje lub interakcje między jednostkami**

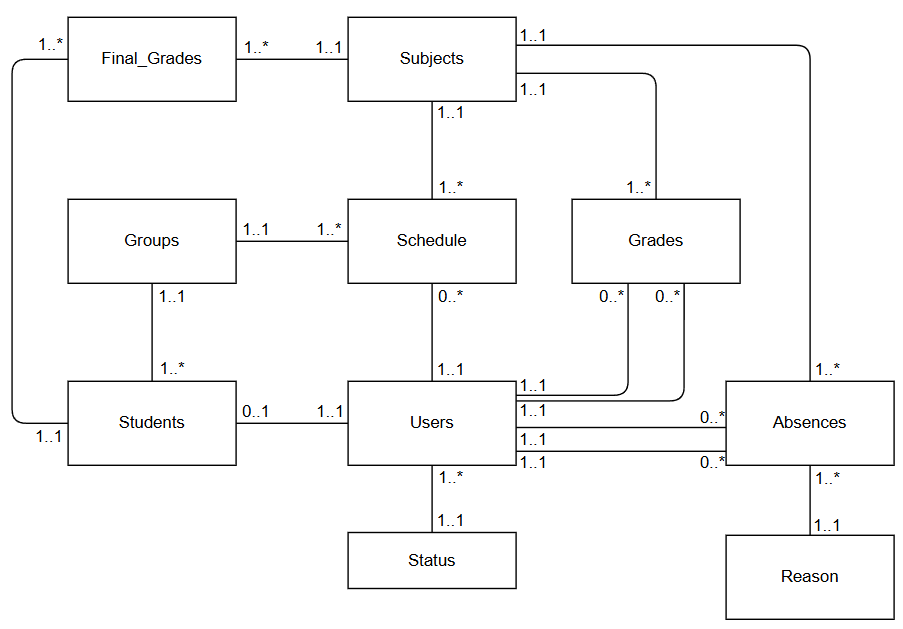
Relacje pomiędzy bytami przedstawia schemat pokazany na rysunku 1. Schemat pokazuje, że zostały użyte 4 typy relacji:

* 0..1 – Relacja jest nieobowiązkowa, maksymalnie 1 zapis;
* 0..\* – Relacja jest nieobowiązkowa, może zawierać 0, 1 lub więcej zapisów;
* 1..1 – Relacja jest obowiązkowa, maksymalnie 1 zapis;
* 1..\* – Relacja jest obowiązkowa, może zawierać 0, 1 lub więcej zapisów;

**Uwagi:**

Tabele Users i Students mają relację typu 1..1 do 0..1, co oznacza:

* Każdy Student musi być powiązany dokładnie z jednym rekordem w Users.
* Każdy rekord w Users może być powiązany z maksymalnie jednym Studentem lub nie być powiązany wcale.

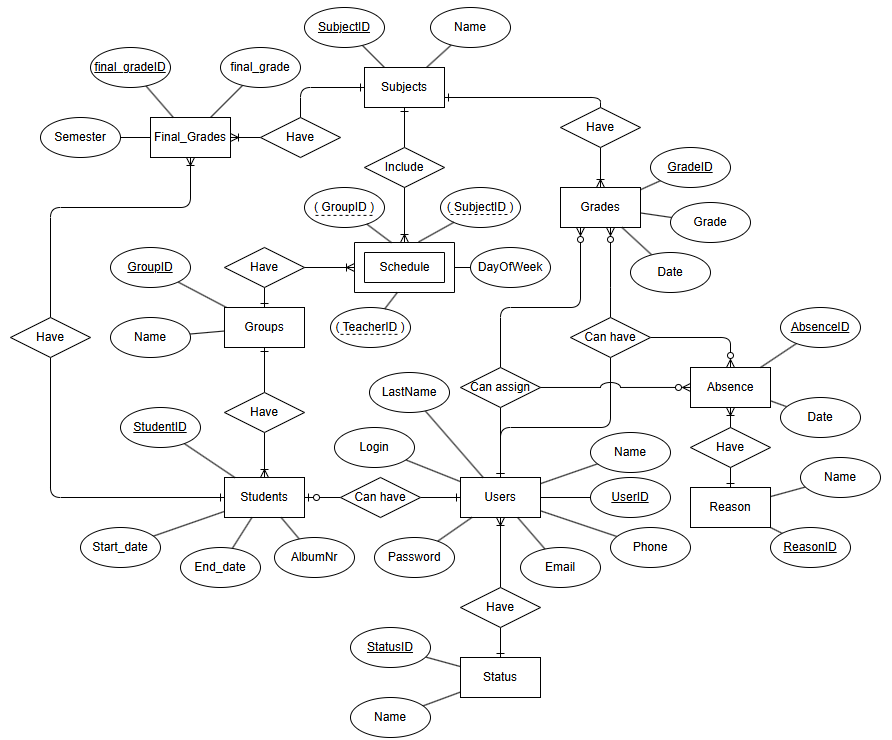
****

Rysunek 1 – Diagram koncepcyjny

1. **Model ERD**

Podczas tworzenia diagramu, przedstawionego na rysunku 2, wykorzystano następujące elementy: encje (prostokąty), atrybuty (owale) i relacje (romby). Atrybuty podkreślone linią ciągłą są kluczami podstawowymi, a zaznaczone linią przerywaną są kluczami częściowymi. Słabe byty nie mają własnego unikalnego klucza głównego i zależą od silnego bytu, czyli nie mogą istnieć bez niego. Takie byty używają klucza częściowego, który w połączeniu z kluczem silnego bytu staje się unikalny. Takie jednostki są oznaczone podwójnym prostokątem.

Oprócz tego, podczas tworzenia diagramu wykorzystano identyczne relacje jak w poprzednim diagramie, tylko przedstawione w formie graficznej.

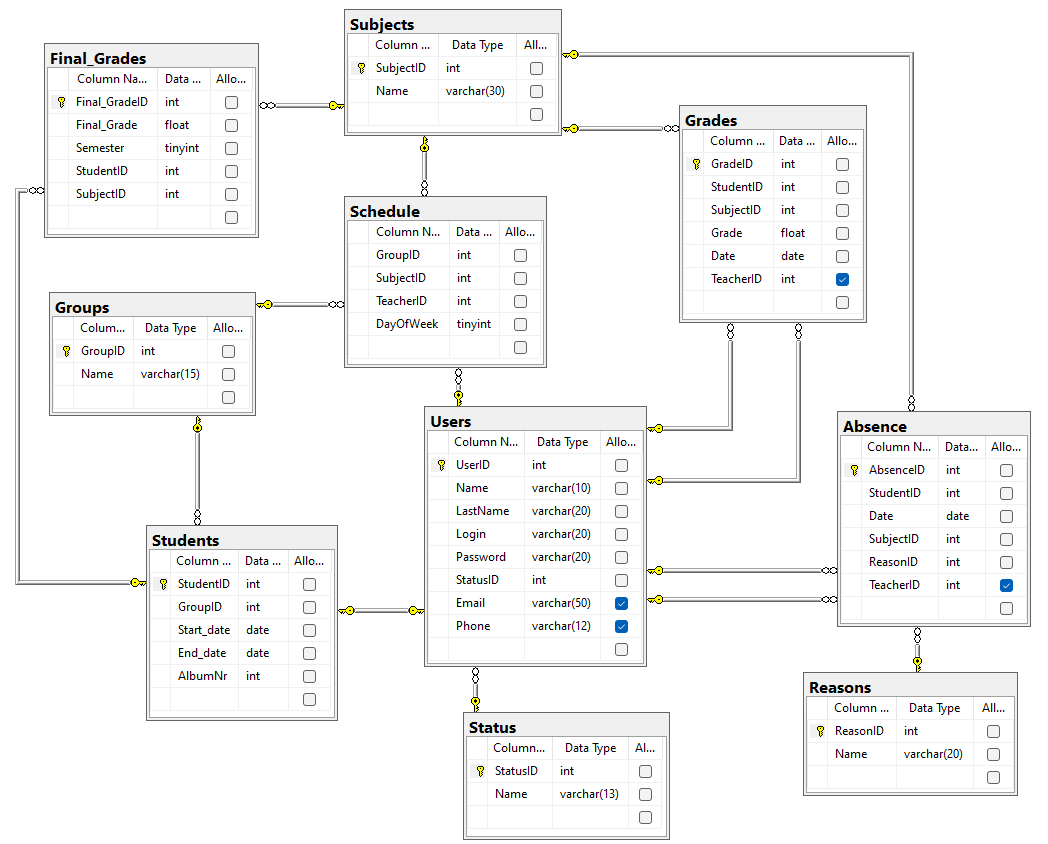


Rysunek 2 – Model ERD

1. **Schemat fizyczny**

Aby utworzyć schemat fizyczny, należy najpierw utworzyć samą bazę danych. Kod użyty do utworzenia tej bazy danych jest pokazany poniżej, a schemat fizyczny jest przedstawiony na rysunku 3.

|  |
| --- |
| create table Groups(  GroupID int identity primary key,  [Name] varchar(15) not null unique  )  create table [Status](  StatusID int identity primary key,  [Name] varchar(13) not null unique  )  create table Users (  UserID int identity primary key,  [Name] varchar(10) not null,  LastName varchar(20) not null,  [Login] varchar(20) not null unique,  [Password] varchar(20) not null,  StatusID int not null,  Email varchar (50) unique,  Phone varchar(12) unique,  foreign key (StatusID) references [Status](StatusID)  )  create table Students(  StudentID int primary key,  GroupID int not null,  [Start\_date] date not null,  [End\_date] date not null,  AlbumNr int not null unique,  foreign key (GroupID) references Groups(GroupID),  foreign key (StudentID) references Users(UserID)  )  create table Subjects(  SubjectID int identity primary key,  [Name] varchar(30) not null unique  )  create table Reasons(  ReasonID int identity primary key,  [Name] varchar(20) not null unique  )  create table Absence(  AbsenceID int identity primary key,  StudentID int not null,  [Date] date not null,  SubjectID int not null,  ReasonID int not null,  TeacherID int,  foreign key (StudentID) references Users(UserID),  foreign key (TeacherID) references Users(UserID),  foreign key (ReasonID) references Reasons (ReasonID),  foreign key (SubjectID) references Subjects(SubjectID)  )  create table Grades(  GradeID int identity primary key,  StudentID int not null,  SubjectID int not null,  Grade float not null,  [Date] date not null,  TeacherID int,  foreign key (StudentID) references Users(UserID),  foreign key (TeacherID) references Users(UserID),  foreign key (SubjectID) references Subjects(SubjectID)  )  create table Schedule(  GroupID int not null,  SubjectID int not null,  TeacherID int not null,  [DayOfWeek] tinyint not null  foreign key (GroupID) references Groups(GroupID),  foreign key (SubjectID) references Subjects(SubjectID),  foreign key (TeacherID) references Users(UserID)  )  create table Final\_Grades(  Final\_GradeID int identity primary key,  Final\_Grade float not null,  Semester tinyint not null,  StudentID int not null,  SubjectID int not null,  foreign key (StudentID) references Students(StudentID),  foreign key (SubjectID) references Subjects(SubjectID)  ) |

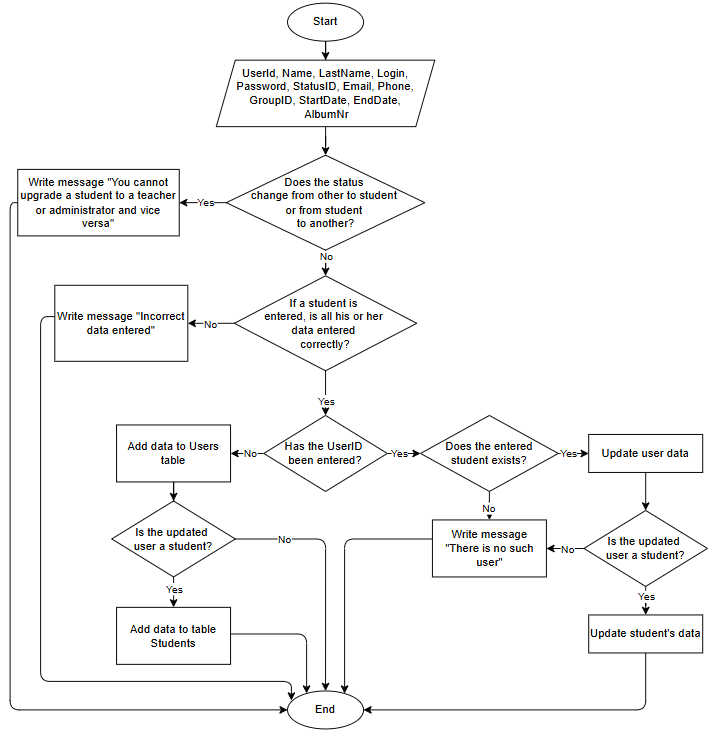


Rysunek 3 – Schemat fizyczny

1. **Procedury i triggery**
2. **Procedura dla dopisania i aktualizacji danych**

Następnym krokiem jest utworzenie procedury, która może dodawać i aktualizować dane w tabeli Users. Kod odpowiedniej procedury został przedstawiony poniżej, a schemat blokowy, który pokazuje, jak ta procedura działa, pokazano na rysunku 4.

|  |
| --- |
| alter procedure AddUpdate  @UserId int = null,  @Name varchar(10),  @LastName varchar(20),  @Login varchar(20),  @Password varchar(20),  @StatusID int,  @Email varchar(50) = null,  @Phone varchar(12) = null,  @GroupID int = null,  @StartDate date = null,  @EndDate date = null,  @AlbumNr int = null  as  begin  declare @CurrentStatusID int;  select @CurrentStatusID = StatusID from Users where UserID = @UserId;  if (@StatusID = 1 and @CurrentStatusID in (2, 3)) or (@StatusID in (2, 3) and @CurrentStatusID = 1)  begin  print 'You cannot upgrade a student to a teacher or administrator and vice versa'  return  end  if @StatusID = 1 and (@GroupID is null or @StartDate is null or @EndDate is null or @AlbumNr is null or @EndDate <= @StartDate)  begin  print 'Incorrect data entered'  return  end  if @UserId is not null  begin  if exists (select 1 from Users where UserID = @UserId)  begin  update Users  set  [Name] = @Name,  LastName = @LastName,  [Login] = @Login,  [Password] = @Password,  StatusID = @StatusID,  Email = @Email,  Phone = @Phone  where UserID = @UserId;  if @StatusID = 1  begin  update Students  set  GroupID = @GroupID,  [Start\_date] = @StartDate,  End\_date = @EndDate,  AlbumNr = @AlbumNr  where StudentID = @UserId  end  end  else  begin  print 'There is no such user'  end  end  else  begin  insert into Users ([Name], LastName, [Login], [Password], StatusID, Email, Phone) values  (@Name, @LastName, @Login, @Password, @StatusID, @Email, @Phone)  if @StatusID = 1  begin  insert into Students (StudentID, GroupID, [Start\_date], End\_date, AlbumNr) values  ((select UserID from Users where [Login] = @Login), @GroupID, @StartDate, @EndDate, @AlbumNr)  end  end  end |

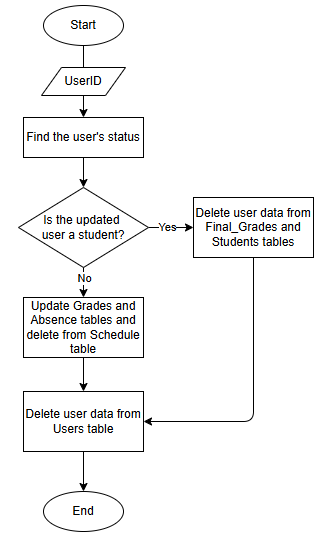


Rysunek 4 – Schemat blokowy procedury AddUpdate

1. **Procedura dla kasowania danych**

Następnie należy utworzyć procedurę, która usunie dane z tabeli Users i sprawdzi, czy usuwany użytkownik nie jest studentem, jeśli tak, to należy usunąć dane z tabel Finle\_Grades i Students. Kod odpowiedniej procedury został przedstawiony poniżej, a schemat blokowy, który pokazuje, jak ta procedura działa, pokazano na rysunku 5.

|  |
| --- |
| create procedure DeleteUser  @UserID int  as  begin  declare @UserStatus int  select @UserStatus = StatusID from Users where UserID = @UserID  if @UserStatus = 1  begin  delete from Final\_Grades where StudentID = @UserID  delete from Students where StudentID = @UserID  end  else  begin  delete from Schedule where TeacherID = @UserID  update Grades set TeacherID = NULL where TeacherID = @UserID  update Absence set TeacherID = NULL where TeacherID = @UserID  end  delete from Users where UserID = @UserID  end |

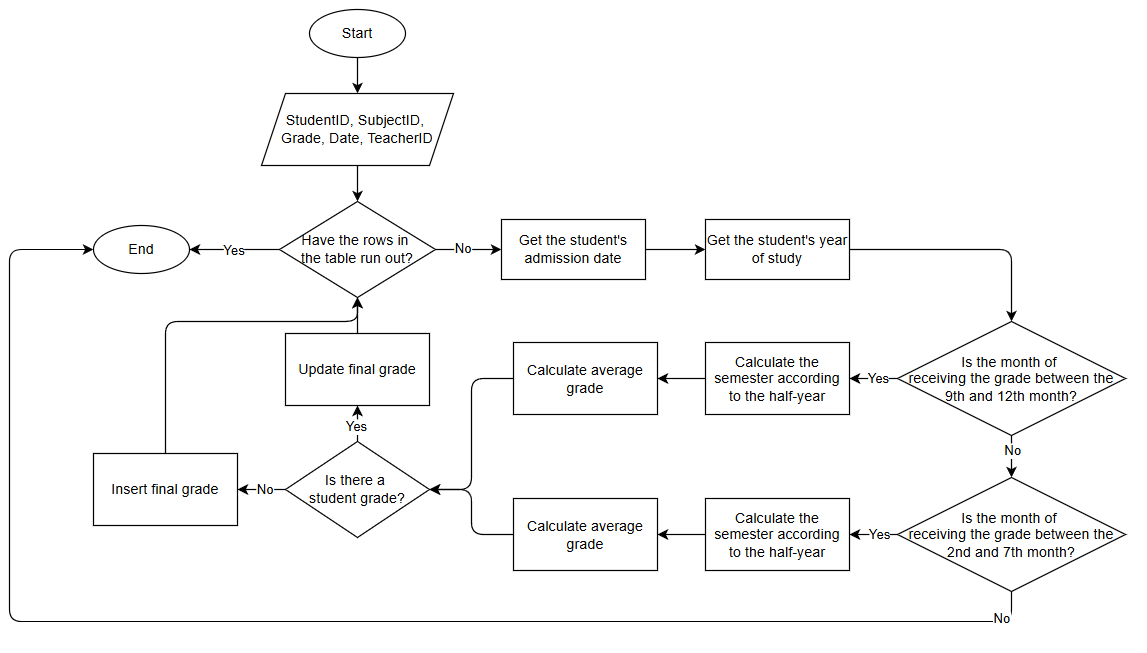


Rysunek 5 – Schemat blokowy procedury DeleteUser

1. **Wyzwalacz dla obliczenia oceny semestralnej**

Po utworzeniu procedur został dodany wyzwalacz, który oblicza ocenę semestralną na podstawie już przypisanych ocen i uzupełnia dane w tabeli Final\_Grades. Kod wyzwalacz przedstawiono poniżej, a schemat blokowy, który pokazuje, jak ten wyzwalacz działa, pokazano na rysunku 6.

|  |
| --- |
| alter trigger trg\_Final\_Grade  on Grades  after insert  as  begin  declare @StartDate date  declare @GradeDate date  declare @Semester int  declare @StudentID int  declare @SubjectID int  declare @FinalGrade float  declare @Year int  declare inserted\_cursor cursor for  select StudentID, [Date], SubjectID from inserted  open inserted\_cursor  fetch next from inserted\_cursor into @StudentID, @GradeDate, @SubjectID  while @@FETCH\_STATUS = 0  begin  select @StartDate = [Start\_date]  from Students  where StudentID = @StudentID  set @Year = year(@GradeDate) - year(@StartDate)  if month(@GradeDate) < 9  begin  set @Year = @Year - 1  end  set @Year = @Year + 1  if MONTH(@GradeDate) between 9 and 12  begin  set @Semester = (@Year - 1) \* 2 + 1    select @FinalGrade = AVG(Grade)  from Grades  where YEAR([Date]) = YEAR(@GradeDate) and MONTH([Date]) >= 9 and StudentID = @StudentID;  end  else if MONTH(@GradeDate) between 2 and 7  begin  set @Semester = (@Year - 1) \* 2 + 2    select @FinalGrade = AVG(Grade)  from Grades  where YEAR([Date]) = YEAR(@GradeDate) and MONTH([Date]) <= 6 and StudentID = @StudentID;  end  else  begin  return  end  if exists (  select 1  from Final\_Grades  where StudentID = @StudentID and SubjectID = @SubjectID and Semester = @Semester  )  begin  update Final\_Grades  set Final\_Grade = @FinalGrade  where StudentID = @StudentID and SubjectID = @SubjectID and Semester = @Semester;  end  else  begin  insert into Final\_Grades (Final\_Grade, Semester, StudentID, SubjectID)  values (@FinalGrade, @Semester, @StudentID, @SubjectID);  end  fetch next from inserted\_cursor into @StudentID, @GradeDate, @SubjectID  end  close inserted\_cursor  deallocate inserted\_cursor  end |



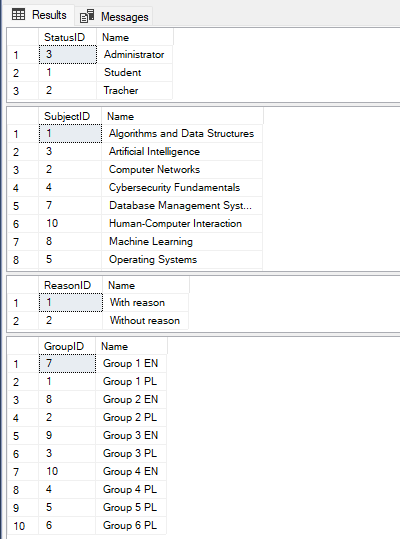
Rysunek 6 – Schemat blokowy wyzwalacza trg\_Final\_Grade

1. **Testowy zestaw danych**

Najpierw warto dodać podstawowe dane, takie jak statusy użytkowników, nazwy grup, tematy i powody nieobecności. Kod użyty do dodania danych jest przedstawiony poniżej, a wynik pokazano na rysunku 7.

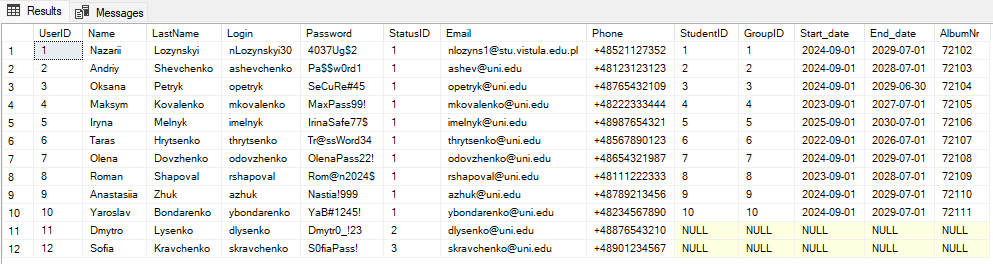
Następnie, korzystając z wcześniej utworzonej procedury, dodano 12 użytkowników: 10 studentów, 1 nauczyciela i 1 administratora. Kod użyty do dodania danych jest przedstawiony poniżej, a wynik pokazano na rysunku 8.

|  |
| --- |
| insert into [Status]([Name]) values  ('Student'),  ('Tracher'),  ('Administrator')  insert into Groups([Name]) values  ('Group 1 PL'),  ('Group 2 PL'),  ('Group 3 PL'),  ('Group 4 PL'),  ('Group 5 PL'),  ('Group 6 PL'),  ('Group 1 EN'),  ('Group 2 EN'),  ('Group 3 EN'),  ('Group 4 EN')  insert into Subjects([Name]) values  ('Algorithms and Data Structures'),  ('Computer Networks'),  ('Artificial Intelligence'),  ('Cybersecurity Fundamentals'),  ('Operating Systems'),  ('Software Engineering'),  ('Database Management Systems'),  ('Machine Learning'),  ('Web Development'),  ('Human-Computer Interaction')  insert into Reasons([Name]) values  ('With reason'),  ('Without reason')  select \* from [Status]  select \* from Groups  select \* from Subjects  select \* from Reasons |



Rysunek 7 – Dodane dane

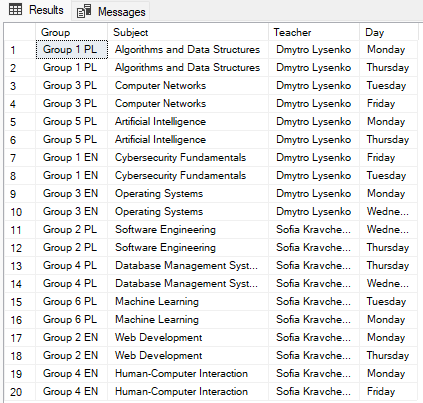
|  |
| --- |
| EXEC AddUpdate NULL, 'Nazarii', 'Lozynskyi', 'nLozynskyi30', '4037Ug$2', 1, 'nlozyns1@stu.vistula.edu.pl', '+48521127352', 1, '2024-09-01', '2029-07-01', 72102  EXEC AddUpdate NULL, 'Andriy', 'Shevchenko', 'ashevchenko', 'Pa$$w0rd1', 1, 'ashev@uni.edu', '+48123123123', 2, '2024-09-01', '2028-07-01', 72103;  EXEC AddUpdate NULL, 'Oksana', 'Petryk', 'opetryk', 'SeCuRe#45', 1, 'opetryk@uni.edu', '+48765432109', 3, '2024-09-01', '2029-06-30', 72104;  EXEC AddUpdate NULL, 'Maksym', 'Kovalenko', 'mkovalenko', 'MaxPass99!', 1, 'mkovalenko@uni.edu', '+48222333444', 4, '2023-09-01', '2027-07-01', 72105;  EXEC AddUpdate NULL, 'Iryna', 'Melnyk', 'imelnyk', 'IrinaSafe77$', 1, 'imelnyk@uni.edu', '+48987654321', 5, '2025-09-01', '2030-07-01', 72106;  EXEC AddUpdate NULL, 'Taras', 'Hrytsenko', 'thrytsenko', 'Tr@ssWord34', 1, 'thrytsenko@uni.edu', '+48567890123', 6, '2022-09-01', '2026-07-01', 72107;  EXEC AddUpdate NULL, 'Olena', 'Dovzhenko', 'odovzhenko', 'OlenaPass22!', 1, 'odovzhenko@uni.edu', '+48654321987', 7, '2024-09-01', '2029-07-01', 72108;  EXEC AddUpdate NULL, 'Roman', 'Shapoval', 'rshapoval', 'Rom@n2024$', 1, 'rshapoval@uni.edu', '+48111222333', 8, '2023-09-01', '2028-07-01', 72109;  EXEC AddUpdate NULL, 'Anastasiia', 'Zhuk', 'azhuk', 'Nastia!999', 1, 'azhuk@uni.edu', '+48789213456', 9, '2024-09-01', '2029-07-01', 72110;  EXEC AddUpdate NULL, 'Yaroslav', 'Bondarenko', 'ybondarenko', 'YaB#1245!', 1, 'ybondarenko@uni.edu', '+48234567890', 10, '2024-09-01', '2029-07-01', 72111;  EXEC AddUpdate NULL, 'Dmytro', 'Lysenko', 'dlysenko', 'Dmytr0\_!23', 2, 'dlysenko@uni.edu', '+48876543210';  EXEC AddUpdate NULL, 'Sofia', 'Kravchenko', 'skravchenko', 'S0fiaPass!', 3, 'skravchenko@uni.edu', '+48901234567';  select \* from Users  left join Students on Users.UserID = Students.StudentID |



Rysunek 8 – Dodani użytkownicy

Następnie należy wypełnić tabelę z harmonogramem zajęć. Kod użyty do dodania danych jest przedstawiony poniżej, a wynik pokazano na rysunku 9.

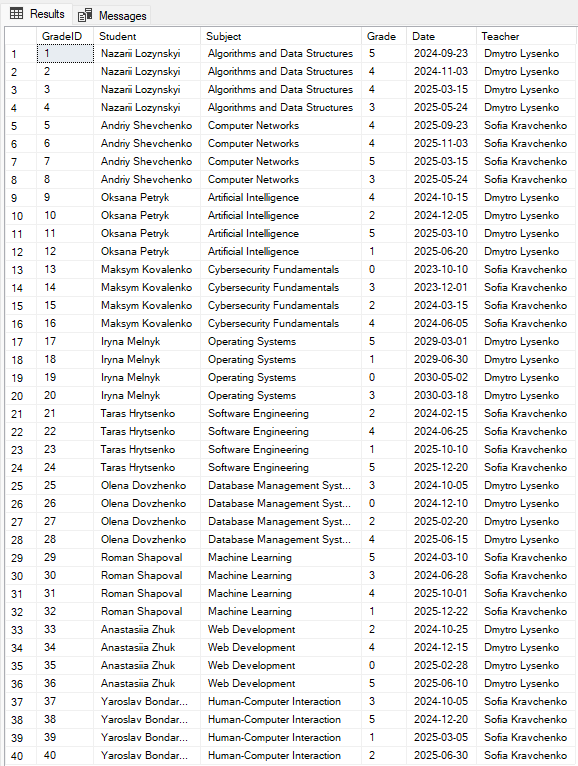
|  |
| --- |
| insert into Schedule(GroupID, SubjectID, TeacherID, [DayOfWeek]) values  (1, 1, 11, 1),  (1, 1, 11, 4),  (3, 2, 11, 2),  (3, 2, 11, 5),  (5, 3, 11, 1),  (5, 3, 11, 4),  (7, 4, 11, 5),  (7, 4, 11, 2),  (9, 5, 11, 1),  (9, 5, 11, 3),  (2, 6, 12, 3),  (2, 6, 12, 4),  (4, 7, 12, 4),  (4, 7, 12, 3),  (6, 8, 12, 2),  (6, 8, 12, 1),  (8, 9, 12, 1),  (8, 9, 12, 4),  (10, 10, 12, 1),  (10, 10, 12, 5)  select Groups.[Name] as [Group], Subjects.[Name] as [Subject], concat(Users.[Name], ' ', Users.LastName) as Teacher, DATENAME(WEEKDAY, DATEADD(DAY, [DayOfWeek] - 1, '19000101')) AS [Day] from Schedule  join Groups on Schedule.GroupID = Groups.GroupID  join Subjects on Schedule.SubjectID = Subjects.SubjectID  join Users on Schedule.TeacherID = Users.UserID |



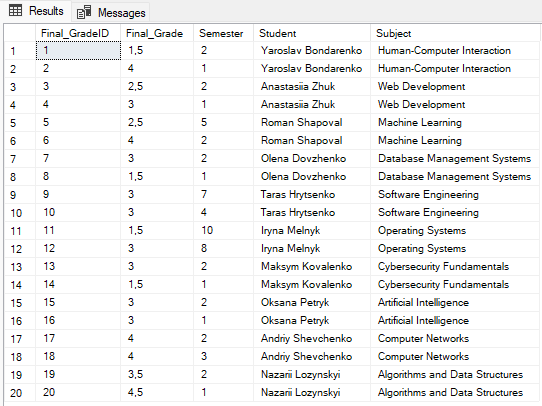
Rysunek 9 – Wypełniony harmonogram zajęć

Także należy dodać oceny do uczniów i sprawdzić działanie utworzonego wyzwalacza. Kod użyty do dodania danych jest przedstawiony poniżej, a wynik pokazano na rysunkach 10 i 11.

|  |
| --- |
| insert into Grades(StudentID, SubjectID, Grade, [Date], TeacherID) values  (1, 1, 5, '2024-9-23', 11),  (1, 1, 4, '2024-11-03', 11),  (1, 1, 4, '2025-03-15', 11),  (1, 1, 3, '2025-05-24', 11),  (2, 2, 4, '2025-9-23', 12),  (2, 2, 4, '2025-11-03', 12),  (2, 2, 5, '2025-03-15', 12),  (2, 2, 3, '2025-05-24', 12),  (3, 3, 4, '2024-10-15', 11),  (3, 3, 2, '2024-12-05', 11),  (3, 3, 5, '2025-03-10', 11),  (3, 3, 1, '2025-06-20', 11),  (4, 4, 0, '2023-10-10', 12),  (4, 4, 3, '2023-12-01', 12),  (4, 4, 2, '2024-03-15', 12),  (4, 4, 4, '2024-06-05', 12),  (5, 5, 5, '2029-03-01', 11),  (5, 5, 1, '2029-06-30', 11),  (5, 5, 0, '2030-05-02', 11),  (5, 5, 3, '2030-03-18', 11),  (6, 6, 2, '2024-02-15', 12),  (6, 6, 4, '2024-06-25', 12),  (6, 6, 1, '2025-10-10', 12),  (6, 6, 5, '2025-12-20', 12),  (7, 7, 3, '2024-10-05', 11),  (7, 7, 0, '2024-12-10', 11),  (7, 7, 2, '2025-02-20', 11),  (7, 7, 4, '2025-06-15', 11),  (8, 8, 5, '2024-03-10', 12),  (8, 8, 3, '2024-06-28', 12),  (8, 8, 4, '2025-10-01', 12),  (8, 8, 1, '2025-12-22', 12),  (9, 9, 2, '2024-10-25', 11),  (9, 9, 4, '2024-12-15', 11),  (9, 9, 0, '2025-02-28', 11),  (9, 9, 5, '2025-06-10', 11),  (10, 10, 3, '2024-10-05', 12),  (10, 10, 5, '2024-12-20', 12),  (10, 10, 1, '2025-03-05', 12),  (10, 10, 2, '2025-06-30', 12)  select GradeID, CONCAT(student.[Name], ' ', student.LastName) as Student, Subjects.[Name] as Subject, Grade, [Date], CONCAT(teacher.[Name], ' ', teacher.[LastName]) as Teacher from Grades  join Users student on Grades.StudentID = student.UserID  join Users teacher on Grades.TeacherID = teacher.UserID  join Subjects on Grades.SubjectID = Subjects.SubjectID  select Final\_GradeID, Final\_Grade, Semester, concat(Users.[Name], ' ', Users.LastName) as Student, Subjects.[Name] as [Subject] from Final\_Grades  join Users on Final\_Grades.StudentID = Users.UserID  join Subjects on Final\_Grades.SubjectID = Subjects.SubjectID |



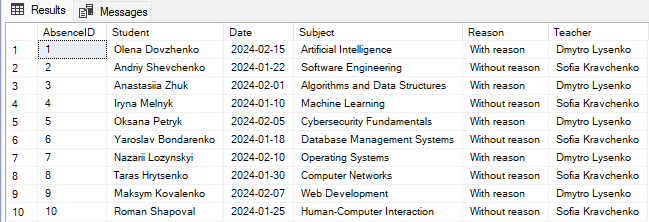
Rysunek 10 – Dodane oceny



Rysunek 11 – Obliczone oceny semestralne

Należy również dodać dane do tabeli Nieobecności, która pokazuje nieobecności uczniów. Kod użyty do dodania danych jest przedstawiony poniżej, a wynik pokazano na rysunku 12.

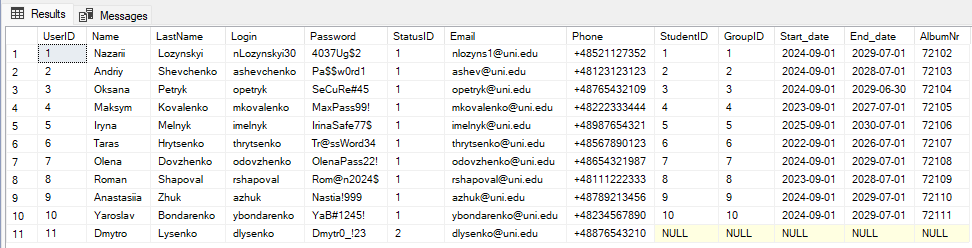
|  |
| --- |
| insert into Absence(StudentID, [Date], SubjectID, ReasonID, TeacherID) values  (7, '2024-02-15', 3, 1, 11),  (2, '2024-01-22', 6, 2, 12),  (9, '2024-02-01', 1, 1, 11),  (5, '2024-01-10', 8, 2, 12),  (3, '2024-02-05', 4, 1, 11),  (10, '2024-01-18', 7, 2, 12),  (1, '2024-02-10', 5, 1, 11),  (6, '2024-01-30', 2, 2, 12),  (4, '2024-02-07', 9, 1, 11),  (8, '2024-01-25', 10, 2, 12);  select AbsenceID, CONCAT(student.[Name], ' ', student.LastName) as Student, [Date], Subjects.[Name] as [Subject], Reasons.[Name] as Reason, CONCAT(teacher.[Name], ' ', teacher.[LastName]) as Teacher from Absence  join Users student on Absence.StudentID = student.UserID  join Users teacher on Absence.TeacherID = teacher.UserID  join Subjects on Absence.SubjectID = Subjects.SubjectID  join Reasons on Reasons.ReasonID = Absence.ReasonID |



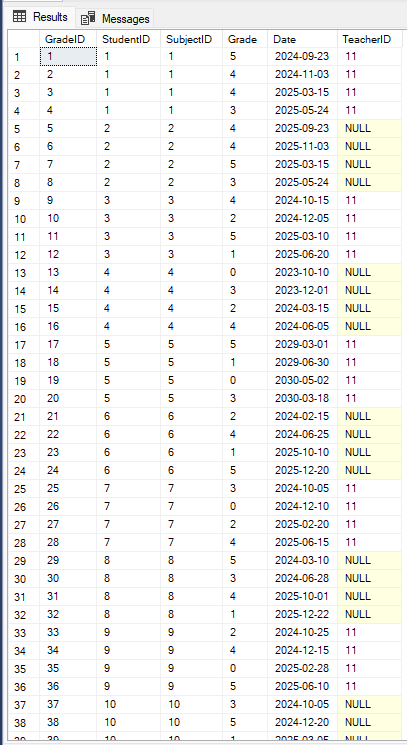
Rysunek 12 – Nieobecność studentów

Na koniec przetestowano procedury aktualizacji danych i usuwania użytkowników. Kod użyty dla aktualizacji i usuwania danych jest przedstawiony poniżej, a wynik pokazano na rysunkach 13 i 14.

|  |
| --- |
| EXEC AddUpdate 1, 'Nazarii', 'Lozynskyi', 'nLozynskyi30', '4037Ug$2', 1, 'nlozyns1@uni.edu', '+48521127352', 1, '2024-09-01', '2029-07-01', 72102  EXEC DeleteUser 12 |



Rysunek 13 – Aktualizacja danych



Rysunek 14 - Wynik usunięcia nauczyciela

1. **Streszczenie**

W wyniku tego projektu stworzono bazę danych z 10 tabelami, która ma ułatwić proces oceniania i rejestrowania nieobecności uczniów. Ponadto stworzono dwie procedury ułatwiające proces dodawania, edytowania i usuwania danych. Dodano również wyzwalacz do tabeli Grades, aby automatycznie obliczać średnią ocen za semestr. Do bazy danych zbudowano trzy diagramy: koncepcyjny, ERD i fizyczny. Do procedur i wyzwalacza dołączono schematy blokowe. Na koniec baza danych została przetestowana poprzez dodanie testowego zestawu danych.