Uniwersytet Gdański

Baza Danych Szkoły

Projekt wykonał:
Wojciech Rosiński
Nr albumu: 240425

Gdańsk 2018

Spis treści

Podstawowe założenia projektu

Projekt ten został wykonany na potrzeby zaliczenia przedmiotu Bazy danych na kierunku Informatyka. Fundamentalnym założeniem projektu jest stworzenie zaplecza do aplikacji bazodanowej pozwalającej na ewidencję i jednocześnie organizację informacji występujących w szkole. Na te informacje składają się dane o nauczycielach, uczniach i ich ocenach semestralnych, przedmiotach oraz zajęciach. Stworzony projekt powinien umożliwiać wprowadzanie, aktualizowanie i usuwanie tego typu danych. Baza danych jest odpowiednio oprogramowana poprzez procedury oraz wyzwalacze omówione w dalszej części dokumentu. Co więcej, dostarcza również wcześniej zdefiniowane widoki, które przedstawiają pewne statystyki.

Jedynym ograniczeniem napotkanym w trakcie tworzenia projektu był problem z wprowadzeniem przykładowych danych. Wynikało to z ilości atrybutów w poszczególnych tabelach, jak i również ogromnej ilości ocen do wprowadzenia dla każdego ucznia. Drugi problem został rozwiązany poprzez zaprojektowany dodatkowo algorytm wprowadzający losowe oceny z przedziału 1-6 do tabeli Oceny dla każdego ucznia. Kod ten jest zaprezentowany w dalszej części dokumentu.

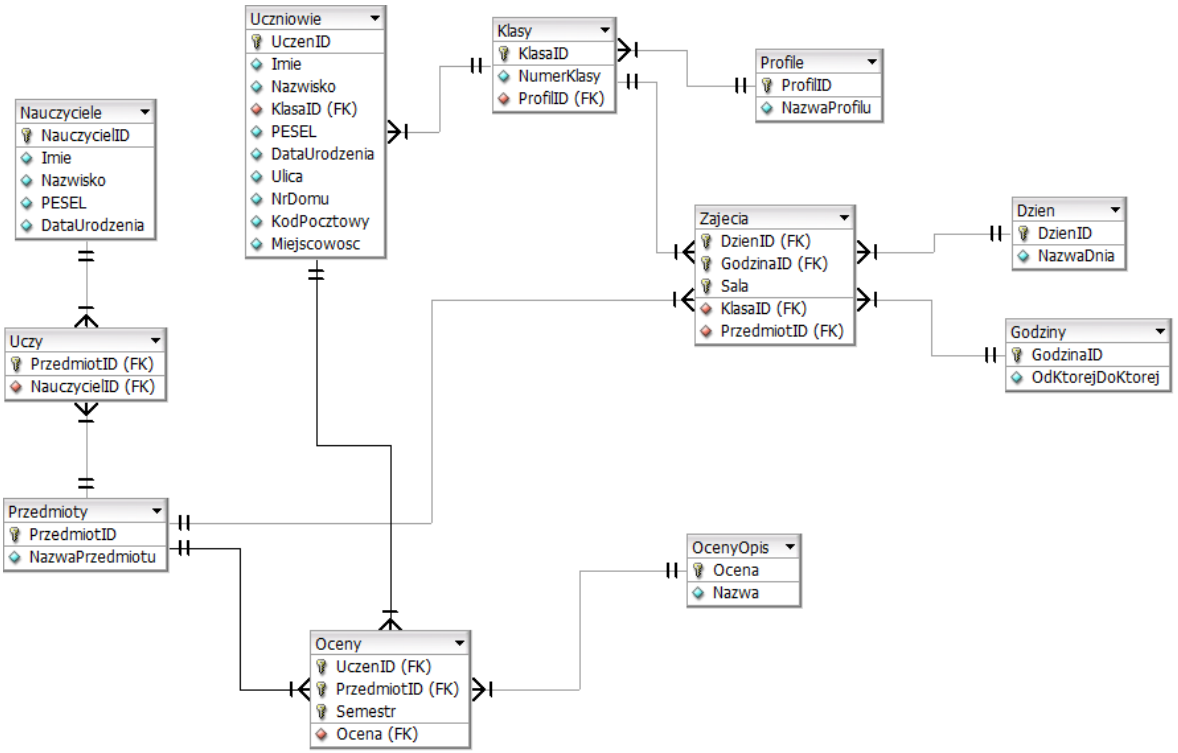
W projekcie przyjęto następujące uproszczenia:

a) każdy przedmiot jest prowadzony tylko przez jednego nauczyciela,
b) brak dziennika lekcyjnego (ewidencjonowane są tylko oceny semestralne).

W niektórych tabelach przy pewnych więzach kluczy obcych celowo nie zdefiniowano kaskadowego usuwania i aktualizacji wierszy. Z tego względu, że niektóre tabele reprezentują stałe wartości, które prawdopodobnie nigdy się nie zmienią. Mowa tutaj przykładowo o tabelach Dzień (zawiera identyfikatory i nazwy poszczególnych dni tygodnia) lub Godziny (zawiera identyfikatory i przedziały czasowe kolejnych zajęć).

Schemat bazy danych (diagram relacji)

Rysunek 1 Schemat bazy danych szkoły



Źródło: Opracowanie własne.

Dodatkowe więzy integralności danych

Akcje kaskadowe dla więzów klucza obcego zostały zdefiniowane dla tabel:

1. Klasy (ProfilID odwołuje się do klucza w tabeli Profile).
2. Uczy (NauczycielID odwołuje się do klucza w tabeli Nauczyciele).
3. Zajecia (KlasaID odwołuje się do klucza w tabeli Klasy).
4. Oceny (PrzedmiotID odwołuje się do klucza w tabeli Przedmioty).
5. Oceny (UczenID odwołuje się do klucza w tabeli Uczniowie).

Widoki

1. Średnie ocen wszystkich uczniów dla semestru pierwszego.

```
CREATE VIEW SrednieOcen_1
AS
SELECT UczenID, CAST(AVG(CAST(Ocena AS DECIMAL(3,2))) AS DECIMAL(3,2)) AS Srednia FROM
Oceny
WHERE Semestr = 1
GROUP BY UczenID
```

2. Średnie ocen wszystkich uczniów dla semestru drugiego.

```
CREATE VIEW SrednieOcen_2
AS
SELECT UczenID, CAST(AVG(CAST(Ocena AS DECIMAL(3,2))) AS DECIMAL(3,2)) AS Srednia FROM
Oceny
WHERE Semestr = 2
GROUP BY UczenID
```

3. Uczniowie, którzy mają w drugim semestrze średnią ocen wyższą niż w pierwszym.

```
CREATE VIEW UczniowieProgres
AS
SELECT A.UczenID, (U.Imie + ' ' + U.Nazwisko) AS 'Imie i nazwisko', (CAST(K.NumerKlasy
AS NCHAR(1)) + K.ProfilID) AS Klasa
FROM SrednieOcen_2 A JOIN Uczniowie U
ON A.UczenID = U.UczenID
JOIN Klasy K ON U.KlasaID = K.KlasaID
WHERE A.Srednia > (SELECT B.Srednia FROM SrednieOcen_1 B WHERE B.UczenID = A.UczenID)
```

4. Liczba uczniów pochodzących z poszczególnych miejscowości.

```
CREATE VIEW LiczbaUczniow_Miejscowosc
AS
SELECT Miejscowosc, COUNT(*) AS LiczbaUczniow FROM Uczniowie
GROUP BY Miejscowosc
```

5. Statystyka prezentująca liczbę wystawionych ocen niedostatecznych w podziale na dwa semestry.

```
CREATE VIEW LiczbaJedynek
AS
SELECT Semestr, COUNT(*) AS LiczbaJedynek FROM Oceny
WHERE Ocena = 1
GROUP BY Semestr
```

6. Statystyka prezentująca informacje o uczniach i przedmiotach, z których wychodzi im negatywna ocena całoroczna.

```
CREATE VIEW Uczniowie_NDST
AS
SELECT O.UczenID, (U.Imie + ' ' + U.Nazwisko) AS 'Imie i nazwisko', (CAST(K.NumerKlasy
AS NCHAR(1)) + K.ProfilID) AS Klasa, P.NazwaPrzedmiotu
FROM Oceny O JOIN Uczniowie U ON O.UczenID = U.UczenID
JOIN Przedmioty P ON O.PrzedmiotID = P.PrzedmiotID
JOIN Klasy K ON U.KlasaID = K.KlasaID
WHERE (SELECT AVG(Ocena) FROM Oceny WHERE UczenID = O.UczenID AND PrzedmiotID =
O.PrzedmiotID) = 1
```

7. Statystyka prezentująca informacje o uczniach i przedmiotach, z których otrzymali oni ocenę niedostateczną w pierwszym semestrze, a w następnym nie poprawili się na tyle, aby uzyskać pozytywną ocenę całoroczną.

```
CREATE VIEW Uczniowie_Brak_Poprawy
AS
SELECT O.UczenID, (U.Imie + ' ' + U.Nazwisko) AS 'Imie i nazwisko', (CAST(K.NumerKlasy
AS NCHAR(1)) + K.ProfilID) AS Klasa, P.NazwaPrzedmiotu
FROM Oceny O JOIN Uczniowie U ON O.UczenID = U.UczenID
JOIN Przedmioty P ON O.PrzedmiotID = P.PrzedmiotID
```

```
JOIN Klasy K ON U.KlasaID = K.KlasaID
WHERE O.Ocena = 1 AND Semestr = 1 AND EXISTS (SELECT * FROM Oceny WHERE UczenID =
O.UczenID AND PrzedmiotID = O.PrzedmiotID AND Semestr = 2 AND Ocena < 3)
ORDER BY UczenID
```

8. Statystyka prezentująca informacje o 10 najlepszych uczniach (mających najlepsze średnie w całej szkole w drugim semestrze).

```
CREATE VIEW Najlepsza10
AS
SELECT TOP 10 U.Imie, U.Nazwisko, (CAST(K.NumerKlasy AS NCHAR(1)) + K.ProfilID) AS
Klasa, S.Srednia
FROM SrednieOcen_2 S JOIN Uczniowie U
ON S.UczenID = U.UczenID JOIN Klasy K ON U.KlasaID = K.KlasaID
ORDER BY S.Srednia DESC
```

9. Ranking klas w szkole – informacje o klasie, profilu i łącznej średniej wszystkich uczniów w klasie.

```
CREATE VIEW Ranking
AS
SELECT (CAST(K.NumerKlasy AS NCHAR(1))+K.ProfilID) AS Klasa, P.NazwaProfilu,
AVG(Srednia) SredniaKlasy FROM SrednieOcen_2 S JOIN Uczniowie U
ON S.UczenID = U.UczenID JOIN Klasy K ON U.KlasaID = K.KlasaID
JOIN Profile P ON K.ProfilID = P.ProfilID
GROUP BY (CAST(K.NumerKlasy AS NCHAR(1))+K.ProfilID), P.NazwaProfilu
ORDER BY SredniaKlasy DESC
```

10. Informacje o ilości zajęć odbywających się w tygodniu w podziale na konkretne przedmioty.

```
CREATE VIEW IleZajecWTygodniu
AS
SELECT NazwaPrzedmiotu, COUNT(*) FROM Zajecia Z JOIN Przedmioty P
ON Z.PrzedmiotID = P.PrzedmiotID
GROUP BY NazwaPrzedmiotu
```

Procedury składowane

1. Procedura otrzymująca nazwę klasy w formacie numerklasy/profil (np. 1A), a zwracająca identyfikator klasy będący liczbą całkowitą.

```
CREATE PROC P0 (@Klasa NCHAR(2), @KID TINYINT OUT)
AS
DECLARE @Nr TINYINT
DECLARE @Profil NCHAR(1)
SET @Nr = CAST((SELECT LEFT(@Klasa, 1)) AS TINYINT)
SET @Profil = (SELECT RIGHT(@Klasa, 1))

SELECT @KID = KlasaID FROM Klasy WHERE NumerKlasy = @Nr AND ProfilID = @Profil
GO
```

2. Procedura otrzymująca nazwę klasy w formacie numerklasy/profil (np. 1A), a zwracająca plan lekcji tej klasy na cały tydzień. Procedura ta używa wcześniej zdefiniowanej procedury nr 1.

```
CREATE PROC P1 (@Klasa NCHAR(2))
AS
DECLARE @KID TINYINT
```

```
EXEC P0 @Klasa, @KID OUT

SELECT D.DzienID, D.NazwaDnia, G.OdKtorejDoKtorej, P.NazwaPrzedmiotu, Z.Sala FROM
Zajecia Z JOIN Dzien D ON Z.DzienID = D.DzienID
JOIN Godziny G ON Z.GodzinaID = G.GodzinaID JOIN Przedmioty P ON Z.PrzedmiotID =
P.PrzedmiotID
WHERE KlasaID = @KID
ORDER BY D.DzienID
GO
```

3. Procedura otrzymująca imię, nazwisko, klasę (format numerklasy/profil np. 1A), semestr, a zwracająca oceny z poszczególnych przedmiotów i średnią do zmiennej podanej jako parametr.

```
CREATE PROC P2 (@Imie NVARCHAR(15), @Nazwisko NVARCHAR(30), @Klasa NCHAR(2), @Sem
TINYINT, @Srednia DECIMAL(3,2) OUT)
AS
DECLARE @KID TINYINT
EXEC P0 @Klasa, @KID OUT

DECLARE @UID INT
SELECT @UID = UczenID FROM Uczniowie
WHERE Imie = @Imie AND Nazwisko = @Nazwisko AND KlasaID = @KID

SELECT P.NazwaPrzedmiotu, O.Ocena FROM Oceny O JOIN Przedmioty P
ON O.PrzedmiotID = P.PrzedmiotID
WHERE O.Semestr = @Sem AND UczenID = @KID

SELECT @Srednia = CAST(AVG(CAST(Ocena AS DECIMAL(3,2))) AS DECIMAL(3,2)) FROM Oceny
WHERE Semestr = @Sem AND UczenID = @KID
GO
```

4. Procedura otrzymująca identyfikator ucznia – zwraca wszystkie dane o uczniu.

```
CREATE PROC P3 (@ID INT)
AS
SELECT * FROM Uczniowie
WHERE UczenID = @ID
GO
```

5. Procedura otrzymująca nazwę przedmiotu, a zwracająca imię i nazwisko nauczyciela prowadzącego oraz salę, w której odbywają się te zajęcia.

```
CREATE PROC P4 (@przedmiot NVARCHAR(50))
AS
SELECT DISTINCT (N.Imie + ' ' + N.Nazwisko) AS 'Imie i nazwisko', Sala FROM Uczy U
JOIN Nauczyciele N ON U.NauczycielID = N.NauczycielID JOIN Przedmioty P
ON U.PrzedmiotID = P.PrzedmiotID JOIN Zajecia Z ON P.PrzedmiotID = Z.PrzedmiotID
WHERE P.NazwaPrzedmiotu = @przedmiot
GO
```

6. Procedura pobierająca imię i nazwisko nauczyciela, a zwracająca w jednym wierszu wszystkie dni i godziny, w których odbywają się jego zajęcia.

```
CREATE PROC P5 (@imie NVARCHAR(15), @nazwisko NVARCHAR(30))
AS
DECLARE @informacja NVARCHAR(1000)
SET @informacja = ''

SELECT @informacja = @informacja + D.NazwaDnia + ' ' + G.OdKtorejDoKtorej + '; ' FROM
Zajecia Z JOIN Uczy U
ON Z.PrzedmiotID = U.PrzedmiotID JOIN Dzien D
```

```
ON Z.DzienID = D.DzienID JOIN Godziny G
ON Z.GodzinaID = G.GodzinaID JOIN Nauczyciele N
ON U.NauczycielID = N.NauczycielID
WHERE N.Imie = @imie AND N.Nazwisko = @nazwisko

SELECT @informacja
GO
```

Wyzwalacze

1. Wyzwalacz na tabeli Uczniowie, który po usunięciu danego ucznia kopiuje jego wszystkie oceny do nowoutworzonej tabeli OcenyTemp. Wyzwalacz sprawdza, czy tabela została już otworzona. Jeśli nie, tworzy ją. W przeciwnym wypadku od razu przechodzi do kopiowania ocen.

```
CREATE TRIGGER Tr0 ON Uczniowie
INSTEAD OF DELETE
AS

	IF OBJECT_ID('dbo.OcenyTemp', 'U') IS NULL
		CREATE TABLE OcenyTemp
		(
		PrzedmiotID TINYINT,
		UczenID INT,
		Ocena TINYINT,
		Semestr TINYINT
		);

	INSERT INTO OcenyTemp
	SELECT * FROM Oceny
	WHERE UczenID IN (SELECT UczenID FROM deleted)

	DELETE FROM Uczniowie
	WHERE UczenID IN (SELECT UczenID FROM deleted)

	SELECT * FROM OcenyTemp
GO
```

2. Wyzwalacz na tabeli Klasy, który po usunięciu danej klasy kopiuje jej zajęcia z całego tygodnia do nowoutworzonej tabeli ZajeciaTemp. Ponadto, wyzwalacz kopiuje informacje o wszystkich uczniach należących do tej klasy do nowoutworzonej tabeli UczniowieTemp. Wyzwalacz kopiuje również oceny wszystkich usuniętych uczniów do tabeli OcenyTemp (wyzwalacz uruchamia inny wyzwalacz – nr 1).

```
CREATE TRIGGER TR1 ON Klasy
INSTEAD OF DELETE
AS
	IF OBJECT_ID('dbo.ZajeciaTemp', 'U') IS NULL
		CREATE TABLE ZajeciaTemp
		(
		DzienID TINYINT,
		GodzinaID TINYINT,
		KlasaID TINYINT,
		PrzedmiotID TINYINT,
		Sala NVARCHAR(3)
		);
	IF OBJECT_ID('dbo.UczniowieTemp', 'U') IS NULL
		CREATE TABLE UczniowieTemp
		(
```

```
            UczenID INTEGER,
            Imie NVARCHAR(15),
            Nazwisko NVARCHAR(30),
            KlasaID TINYINT,
            PESEL NVARCHAR(30),
            DataUrodzenia DATE,
            Ulica NVARCHAR(40),
            NrDomu TINYINT,
            KodPocztowy NVARCHAR(6),
            Miejscowosc NVARCHAR(15)
            )

        INSERT INTO ZajeciaTemp
        SELECT * FROM Zajecia
        WHERE KlasaID IN (SELECT KlasaID FROM deleted)

        INSERT INTO UczniowieTemp
        SELECT * FROM Uczniowie
        WHERE KlasaID IN (SELECT KlasaID FROM deleted)

        DELETE FROM Uczniowie
        WHERE KlasaID IN (SELECT KlasaID FROM deleted)

        DELETE FROM Klasy
        WHERE KlasaID IN (SELECT KlasaID FROM deleted)

        SELECT * FROM ZajeciaTemp
        SELECT * FROM UczniowieTemp
GO
```

3. Wyzwalacz na tabeli Dzien, który w momencie próby usunięcia z niej danych informuje o braku takiej możliwości wraz z wyjaśnieniem.

```
CREATE TRIGGER TR2 ON Dzien
INSTEAD OF DELETE
AS
RAISERROR('Nie wolno usuwac danych z tabeli Dzien, poniewaz klucz glowny tej tabeli
stanowi klucz obcy w tabeli Zajecia.', 5, 1)
GO
```

4. Wyzwalacz na tabeli Profile, który po wstawieniu wierszy do tej tabeli wyświetla komunikat o identyfikatorze i nazwie dodanego profilu oraz podpowiada, co należy zrobić dalej.

```
CREATE TRIGGER TR3 ON Profile
AFTER INSERT
AS
DECLARE @pid NCHAR(1)
DECLARE @nazwa VARCHAR(50)

SELECT @pid = ProfilID FROM inserted
SELECT @nazwa = NazwaProfilu FROM inserted

PRINT('Po dodaniu nowego profilu ' + @pid + ' o nazwie ' + @nazwa + ' nalezy
zdefiniowac nowa klase w tabeli Klasy!')
GO
```

5. Wyzwalacz na tabeli Uczniowie, który po zaktualizowaniu danych ucznia wyświetla informacje o tym, które dane zostały zmienione. Ze względu na fakt, że instrukcje UPDATE można wykonać dla wielu wierszy konieczne jest przekopiowanie zaktualizowanych danych do nowej tabeli z dodatkowym indeksem, który pozwala na iterowanie po poszczególnych wierszach.

```
CREATE TRIGGER TR4 ON Uczniowie
AFTER UPDATE
AS

SET NOCOUNT ON

CREATE TABLE #temp
(
id INT IDENTITY(1,1),
UczenID INTEGER,
Imie NVARCHAR(15),
Nazwisko NVARCHAR(30),
KlasaID TINYINT,
PESEL NVARCHAR(30),
DataUrodzenia DATE,
Ulica NVARCHAR(40),
NrDomu TINYINT,
KodPocztowy NVARCHAR(6),
Miejscowosc NVARCHAR(15)
)

INSERT INTO #temp
SELECT * FROM inserted

DECLARE @id INT
SET @id = (SELECT TOP 1 id FROM #temp ORDER BY id)

DECLARE @uid INT
WHILE EXISTS (SELECT * FROM #temp WHERE id = @id)
BEGIN
	SET @uid = (SELECT UczenID FROM #temp WHERE id = @id)
	PRINT('Dla ucznia o ID: ' + CAST(@uid AS VARCHAR(1000)) + ' zaktualizowano
nastepujace pozycje:')
	IF (SELECT UczenID FROM deleted WHERE UczenID = @uid) != (SELECT UczenID FROM
inserted WHERE UczenID = @uid)
	PRINT(' UczenID ');
	IF (SELECT Imie FROM deleted WHERE UczenID = @uid) != (SELECT Imie FROM inserted
WHERE UczenID = @uid)
	PRINT(' Imie ');
	IF (SELECT Nazwisko FROM deleted WHERE UczenID = @uid) != (SELECT Nazwisko FROM
inserted WHERE UczenID = @uid)
	PRINT(' Nazwisko ');
	IF (SELECT KlasaID FROM deleted WHERE UczenID = @uid) != (SELECT KlasaID FROM
inserted WHERE UczenID = @uid)
	PRINT(' KlasaID ');
	IF (SELECT PESEL FROM deleted WHERE UczenID = @uid) != (SELECT PESEL FROM
inserted WHERE UczenID = @uid)
	PRINT(' PESEL ');
	IF (SELECT DataUrodzenia FROM deleted WHERE UczenID = @uid) != (SELECT
DataUrodzenia FROM inserted WHERE UczenID = @uid)
	PRINT(' DataUrodzenia ');
	IF (SELECT Ulica FROM deleted WHERE UczenID = @uid) != (SELECT Ulica FROM
inserted WHERE UczenID = @uid)
	PRINT(' Ulica ');
```

```
	IF (SELECT NrDomu FROM deleted WHERE UczenID = @uid) != (SELECT NrDomu FROM
inserted WHERE UczenID = @uid)
	PRINT(' NrDomu ');
	IF (SELECT KodPocztowy FROM deleted WHERE UczenID = @uid) != (SELECT KodPocztowy
FROM inserted WHERE UczenID = @uid)
	PRINT(' KodPocztowy ');
	IF (SELECT Miejscowosc FROM deleted WHERE UczenID = @uid) != (SELECT Miejscowosc
FROM inserted WHERE UczenID = @uid)
	PRINT(' Miejscowosc ');

	SET @id = @id + 1
END
SET NOCOUNT OFF
GO
```

Skrypt tworzący bazę danych

```
use Szkola

CREATE TABLE Przedmioty (
  PrzedmiotID TINYINT IDENTITY(1,1),
  NazwaPrzedmiotu NVARCHAR(50) NOT NULL,
  PRIMARY KEY(PrzedmiotID)
);

CREATE TABLE Profile (
  ProfilID NCHAR(1) NOT NULL,
  NazwaProfilu NVARCHAR(50) NOT NULL,
  PRIMARY KEY(ProfilID)
);

CREATE TABLE OcenyOpis (
  Ocena TINYINT IDENTITY(1,1),
  Nazwa NVARCHAR(15) NOT NULL,
  PRIMARY KEY(Ocena)
);

CREATE TABLE Godziny (
  GodzinaID TINYINT IDENTITY(1,1),
  OdKtorejDoKtorej NVARCHAR(11) NOT NULL,
  PRIMARY KEY(GodzinaID)
);

CREATE TABLE Dzien (
  DzienID TINYINT IDENTITY(1,1),
  NazwaDnia NVARCHAR(12) NOT NULL,
  PRIMARY KEY(DzienID)
);

CREATE TABLE Nauczyciele (
  NauczycielID INTEGER IDENTITY(1,1),
  Imie NVARCHAR(15) NOT NULL,
  Nazwisko NVARCHAR(30) NOT NULL,
  PESEL NVARCHAR(11) NOT NULL,
  DataUrodzenia DATE NOT NULL,
  PRIMARY KEY(NauczycielID)
);

CREATE TABLE Klasy (
  KlasaID TINYINT IDENTITY(1,1),
  NumerKlasy TINYINT NOT NULL,
```

```
  ProfilID NCHAR(1) NOT NULL,
  PRIMARY KEY(KlasaID),
  FOREIGN KEY(ProfilID) REFERENCES Profile(ProfilID)
  ON UPDATE CASCADE
);

CREATE TABLE Uczy (
  PrzedmiotID TINYINT NOT NULL,
  NauczycielID INTEGER NOT NULL,
  PRIMARY KEY(PrzedmiotID),
  FOREIGN KEY(PrzedmiotID) REFERENCES Przedmioty(PrzedmiotID),
  FOREIGN KEY(NauczycielID) REFERENCES Nauczyciele(NauczycielID)
  ON UPDATE CASCADE
);

CREATE TABLE Zajecia (
  DzienID TINYINT NOT NULL,
  GodzinaID TINYINT NOT NULL,
  KlasaID TINYINT NOT NULL,
  PrzedmiotID TINYINT NOT NULL,
  Sala NVARCHAR(3) NOT NULL,
  PRIMARY KEY(DzienID, GodzinaID, Sala),
  FOREIGN KEY(DzienID) REFERENCES Dzien(DzienID),
  FOREIGN KEY(GodzinaID) REFERENCES Godziny(GodzinaID),
  FOREIGN KEY(KlasaID) REFERENCES Klasy(KlasaID)
  ON DELETE CASCADE,
  FOREIGN KEY(PrzedmiotID) REFERENCES Przedmioty(PrzedmiotID)
);

CREATE TABLE Uczniowie (
  UczenID INTEGER IDENTITY(1,1),
  Imie NVARCHAR(15) NOT NULL,
  Nazwisko NVARCHAR(30) NOT NULL,
  KlasaID TINYINT NOT NULL,
  PESEL NVARCHAR(30) NOT NULL,
  DataUrodzenia DATE NOT NULL,
  Ulica NVARCHAR(40) NOT NULL,
  NrDomu TINYINT NOT NULL,
  KodPocztowy NVARCHAR(6) NOT NULL,
  Miejscowosc NVARCHAR(15) NOT NULL,
  PRIMARY KEY(UczenID),
  FOREIGN KEY(KlasaID) REFERENCES Klasy(KlasaID)
);

CREATE TABLE Oceny (
  PrzedmiotID TINYINT NOT NULL,
  UczenID INTEGER NOT NULL,
  Ocena TINYINT NOT NULL,
  Semestr TINYINT NOT NULL,
  PRIMARY KEY(PrzedmiotID, UczenID, Semestr),
  FOREIGN KEY(PrzedmiotID) REFERENCES Przedmioty(PrzedmiotID)
  ON UPDATE CASCADE,
  FOREIGN KEY(UczenID) REFERENCES Uczniowie(UczenID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY(Ocena) REFERENCES OcenyOpis(Ocena)
);
```

Skrypt wprowadzający rekordy do bazy danych

```
use Szkola

INSERT INTO Profile VALUES
('A', 'Humanistyczny'),
('B', 'Matematyczno-fizyczny')

INSERT INTO Klasy VALUES
(1, 'A'),
(1, 'B'),
(2, 'A'),
(2, 'B'),
(3, 'A'),
(3, 'B')

INSERT INTO Uczniowie VALUES
('Jan','Kowalski', 1, '95020403876', '1995-02-04', 'Kochanowskiego', 50, '80-402',
'Gdansk'),
('Marcin','Szczucki', 1, '95031208764', '1995-03-31', 'Grunwaldzka', 42, '80-430',
'Gdansk'),
('Ewelina','Makowiecka', 1, '94022003876', '1995-02-20', 'Dzielna', 12, '50-502',
'Sopot'),
('Katarzyna','Nosowska', 1, '95092005534', '1995-09-20', 'Biala', 4, '80-200',
'Gdansk'),
('Wojciech','Zorawski', 1, '95120905344', '1995-12-09', 'Polska', 21, '80-190',
'Gdynia'),
('Kamil','Nowy', 2, '95040555534', '1995-04-05', 'Mala', 50, '80-230', 'Gdansk'),
('Jan','Bielski', 2, '95081533334', '1995-08-15', 'Duza', 80, '50-239', 'Sopot'),
('Magdalena','Wrzosk', 2, '95013029534', '1995-01-30', 'Brytyjska', 50, '80-555',
'Gdansk'),
('Patryk','Wojcik', 2, '95110705324', '1995-11-07', 'Malownicza', 22, '50-666',
'Sopot'),
('Natalia','Kowalska', 2, '95052305534', '1995-05-23', 'Sembrzyckiego', 15, '80-012',
'Gdansk'),
('Marta','Pewna', 3, '94042005034', '1994-04-20', 'Drzewna', 23, '80-902', 'Gdansk'),
('Szymon','Szary', 3, '94063005834', '1994-06-30', 'Istna', 2, '80-304', 'Gdansk'),
('Wojciech','Solidny', 3, '94071505224', '1994-07-15', 'Warszawska', 18, '50-111',
'Sopot'),
('Jadwiga','Pastuszka', 3, '94083005134', '1994-08-30', 'Marszalkowska', 12, '50-521',
'Sopot'),
('Natalia','Niepewna', 3, '94062398765', '1994-06-23', 'Mokotowska', 23, '80-333',
'Gdansk'),
('Jacek','Rzodkiewicz', 4, '94122005134', '1994-12-20', 'Znana', 80, '80-354',
'Gdansk'),
('Tomasz','Rzodkiewicz', 4, '94122005144', '1994-12-20', 'Znana', 80, '80-354',
'Gdansk'),
('Pawel','Maslany', 4, '94102405664', '1994-10-24', 'Zawodna', 4, '50-121', 'Sopot'),
('Franciszek','Mazur', 4, '94031705254', '1994-03-17', 'Mickiewicza', 66, '80-904',
'Gdansk'),
('Paulina','Brzoza', 4, '94071505511', '1994-07-15', 'Ustrojowa', 100, '20-555',
'Gdynia'),
('Krzysztof','Lewandowski', 5, '93011905302', '1993-01-19', 'Piwna', 42, '80-708',
'Gdansk'),
('Maurycy','Piecewski', 5, '93112905311', '1993-11-29', 'Ogarna', 50, '20-321',
'Gdynia'),
('Monika','Czarna', 5, '93032203432', '1993-03-22', 'Piwna', 21, '80-999', 'Gdansk'),
('Ewelina','Jakubowska', 5, '93050302752', '1993-05-03', 'Oparta', 78, '50-708',
'Sopot'),
```

```
('Daniel','Majewski', 5, '93022801112', '1993-02-28', 'Piastowska', 100, '80-502',
'Gdansk'),
('Szymon','Zberezny', 6, '93111804471', '1993-11-18', 'Nowowiejska', 80, '20-222',
'Gdynia'),
('Krzysztof','Zawodny', 6, '93062003331', '1993-06-20', 'Miejska', 120, '20-521',
'Gdynia'),
('Matylda','Wierna', 6, '93011702411', '1993-01-17', 'Serowa', 5, '20-452', 'Gdynia'),
('Filip','Marudny', 6, '93032703331', '1993-03-27', 'Nowowiejska', 8, '80-372',
'Gdansk'),
('Aleksandra','Nasza', 6, '93040403331', '1993-04-04', 'Staropolska', 7, '80-144',
'Gdansk')

INSERT INTO Nauczyciele VALUES
('Franciszek','Brzozowski','60021690888', '1960-02-16'), --1
('Alicja','Ciunelis','55031790111', '1955-03-17'), -- 2
('Anna','Chojecka','70071590212', '1970-07-15'), -- 3
('Dariusz','Bujak','60021690888', '1960-02-16'), -- 4
('Anna','Dmitruk','60071990171', '1960-07-19'), -- 5
('Agnieszka','Kubinska','63051490248', '1963-05-14'), -- 6
('Pawel','Drobisz','79041990158', '1979-04-19'), --7
('Wioletta','Ewartowska','70111477321', '1970-11-14'), --8
('Iwona','Gesek','79120155555','1979-12-01'), --9
('Rafal','Grzebyk','77090790523', '1977-09-07'), --10
('Iwona','Jakimowicz','77031491218', '1977-03-14'), --- 11
('Ewa','Legucka','79061788554', '1979-06-17'), ---12
('Jolanta','Vetter','68080156291', '1968-08-01'), ---13
('Monika','Szczepanska','78041890221', '1978-04-18') --14

INSERT INTO Przedmioty VALUES
('Jezyk polski'),
('Matematyka'),
('Biologia'),
('Chemia'),
('Wychowanie fizyczne'),
('Jezyk niemiecki'),
('Fizyka'),
('Religia'),
('Jezyk angielski'),
('Historia'),
('Geografia'),
('Podstawy przedsiebiorczosci'),
('Wiedza o kulturze'),
('Informatyka')

INSERT INTO Uczy VALUES
(1, 9), (2, 5), (3, 2), (4, 3), (5, 7), (6, 6),
(7, 8), (8, 10), (9, 11), (10, 1), (11, 12), (12, 14),
(13, 13), (14, 4)

INSERT INTO Dzien VALUES
('Poniedzialek'),
('Wtorek'),
('Sroda'),
('Czwartek'),
('Piatek')

INSERT INTO Godziny VALUES
('8:00-8:45'),
('8:50-9:35'),
('9:40-10:25'),
('10:40-11:25'),
('11:30-12:15'),
```

```
('12:20-13:05'),
('13:10-13:55'),
('14:00-14:45')

INSERT INTO Zajecia VALUES
(1, 1, 1, 1, 101), (1, 2, 1, 2, 107), (1, 3, 1, 3, 108), (1, 4, 1, 4, 102),
(1, 1, 2, 2, 107), (1, 2, 2, 1, 101), (1, 3, 2, 4, 102), (1, 4, 2, 3, 108),
(1, 1, 3, 5, 15), (1, 2, 3, 6, 221), (1, 3, 3, 7, 222), (1, 4, 3, 8, 229),
(1, 1, 4, 6, 221), (1, 2, 4, 5, 15), (1, 3, 4, 8, 229), (1, 4, 4, 7, 222),
(1, 1, 5, 9, 321), (1, 2, 5, 10, 333), (1, 3, 5, 11, 343), (1, 4, 5, 12, 350),
(1, 1, 6, 10, 353), (1, 2, 6, 9, 321), (1, 3, 6, 12, 350), (1, 4, 6, 11, 343),

(2, 2, 1, 5, 15), (2, 3, 1, 6, 221), (2, 4, 1, 7, 222), (2, 5, 1, 8, 229),
(2, 2, 2, 6, 221), (2, 3, 2, 5, 15), (2, 4, 2, 8, 229), (2, 5, 2, 7, 222),
(2, 2, 3, 9, 321), (2, 3, 3, 10, 333), (2, 4, 3, 11, 343), (2, 5, 3, 12, 350),
(2, 2, 4, 10, 333), (2, 3, 4, 9, 321), (2, 4, 4, 12, 350), (2, 5, 4, 11, 343),
(2, 2, 5, 13, 55), (2, 3, 5, 14, 56), (2, 4, 5, 1, 101), (2, 5, 5, 2, 107),
(2, 2, 6, 14, 56), (2, 3, 6, 13, 55), (2, 4, 6, 2, 107), (2, 5, 6, 1, 101),

(3, 3, 1, 9, 321), (3, 4, 1, 10, 333), (3, 5, 1, 11, 343), (3, 6, 1, 12, 350),
(3, 3, 2, 10, 333), (3, 4, 2, 9, 321), (3, 5, 2, 12, 350), (3, 6, 2, 11, 343),
(3, 3, 3, 13, 55), (3, 4, 3, 14, 56), (3, 5, 3, 1, 101), (3, 6, 3, 2, 107),
(3, 3, 4, 14, 56), (3, 4, 4, 13, 55), (3, 5, 4, 2, 107), (3, 6, 4, 1, 101),
(3, 3, 5, 3, 108), (3, 4, 5, 4, 102), (3, 5, 5, 5, 15), (3, 6, 5, 6, 221),
(3, 3, 6, 4, 102), (3, 4, 6, 3, 108), (3, 5, 6, 6, 221), (3, 6, 6, 5, 15),

(4, 4, 1, 13, 55), (4, 5, 1, 14, 56), (4, 6, 1, 1, 101), (4, 7, 1, 2, 107),
(4, 4, 2, 14, 56), (4, 5, 2, 13, 55), (4, 6, 2, 2, 107), (4, 7, 2, 1, 101),
(4, 4, 3, 3, 108), (4, 5, 3, 4, 102), (4, 6, 3, 5, 15), (4, 7, 3, 6, 221),
(4, 4, 4, 4, 102), (4, 5, 4, 3, 108), (4, 6, 4, 6, 221), (4, 7, 4, 5, 15),
(4, 4, 5, 7, 222), (4, 5, 5, 8, 229), (4, 6, 5, 9, 321), (4, 7, 5, 10, 333),
(4, 4, 6, 8, 229), (4, 5, 6, 7, 222), (4, 6, 6, 10, 333), (4, 7, 6, 9, 321),

(5, 5, 1, 3, 108), (5, 6, 1, 4, 102), (5, 7, 1, 5, 15), (5, 8, 1, 6, 221),
(5, 5, 2, 4, 102), (5, 6, 2, 3, 108), (5, 7, 2, 6, 221), (5, 8, 2, 5, 15),
(5, 5, 3, 7, 222), (5, 6, 3, 8, 229), (5, 7, 3, 10, 333), (5, 8, 3, 11, 343),
(5, 5, 4, 8, 229), (5, 6, 4, 7, 222), (5, 7, 4, 11, 343), (5, 8, 4, 10, 333),
(5, 5, 5, 11, 343), (5, 6, 5, 12, 350), (5, 7, 5, 13, 55), (5, 8, 5, 14, 56),
(5, 5, 6, 12, 350), (5, 6, 6, 11, 343), (5, 7, 6, 14, 56), (5, 8, 6, 13, 55)

INSERT INTO OcenyOpis VALUES
('Niedostateczny'),
('Dopuszczajacy'),
('Dostateczny'),
('Dobry'),
('Bardzo dobry'),
('Celujacy')

------ Generowanie ocen z 14 przedmiotow dla kazdego ucznia z uwzglednieniem podzialu
na dwa semestry

DECLARE @uczenid INT
SET @uczenid = 1
DECLARE @przedmiotid INT
SET @przedmiotid = 1
DECLARE @ocena INT

WHILE (SELECT @uczenid) != 31
BEGIN
	WHILE (SELECT @przedmiotid) != 15
	BEGIN
		SELECT @ocena = 1 + ABS(Checksum(NewID()) % 6)
```

```
                INSERT INTO Oceny VALUES
                (@przedmiotid, @uczenid, @ocena, 1)
                SET @przedmiotid = @przedmiotid + 1
        END
        SET @przedmiotid = 1
        SET @uczenid = @uczenid + 1
END

SET @uczenid = 1

WHILE (SELECT @uczenid) != 31
BEGIN
        WHILE (SELECT @przedmiotid) != 15
        BEGIN
                SELECT @ocena = 1 + ABS(Checksum(NewID()) % 6)
                INSERT INTO Oceny VALUES
                (@przedmiotid, @uczenid, @ocena, 2)
                SET @przedmiotid = @przedmiotid + 1
        END
        SET @przedmiotid = 1
        SET @uczenid = @uczenid + 1
END
```

Przykładowe zapytania

1. Zapytanie zwracające całotygodniowy plan zajęć klasy, której identyfikator podany jest w warunku WHERE.

```
SELECT D.DzienID, D.NazwaDnia, G.OdKtorejDoKtorej, P.NazwaPrzedmiotu, Z.Sala FROM
Zajecia Z JOIN Dzien D ON Z.DzienID = D.DzienID
JOIN Godziny G ON Z.GodzinaID = G.GodzinaID JOIN Przedmioty P ON Z.PrzedmiotID =
P.PrzedmiotID
WHERE KlasaID = 1
ORDER BY D.DzienID
GO
```

2. Zapytanie zwracające średnią danego ucznia dla określonego semestru. Identyfikator ucznia i numer semestru podany jest w warunku WHERE.

```
SELECT CAST(AVG(CAST(Ocena AS DECIMAL(3,2))) AS DECIMAL(3,2)) FROM Oceny
WHERE UczenID = 1 AND Semestr = 1
```

3. Zapytanie zwracające wszystkie dane uczniów mieszkających w Gdańsku lub Sopocie.

```
SELECT * FROM Uczniowie
WHERE Miejscowosc IN ('Gdansk','Sopot')
```

4. Zapytanie zwracające ilość ocen niedostatecznych z poszczególnych przedmiotów w klasie o identyfikatorze nr 1 na koniec drugiego semestru. Identyfikator klasy oraz numer semestru podany jest w warunku WHERE.

```
SELECT P.NazwaPrzedmiotu, COUNT(O.Ocena) FROM Oceny O JOIN OcenyOpis OO
ON O.Ocena = OO.Ocena JOIN Uczniowie U
ON O.UczenID = U.UczenID JOIN Przedmioty P
ON O.PrzedmiotID = P.PrzedmiotID
WHERE U.KlasaID = 1 AND O.Ocena = 1 AND O.Semestr = 2
GROUP BY P.NazwaPrzedmiotu
```

*Źródło grafiki ze strony tytułowej: https://media.giphy.com/media/NnJzHVQVOUOmA/giphy.gif