Projekt bazy danych szkoły, wykonany na potrzeby zajęć laboratoryjnych z kursu *Bazy Danych*.

Wykonanie:

Mikołaj Barsznica

Wojciech Szlosek

Założenia projektu i krótkie wprowadzenie

- Cel:

Realizacja bazy danych szkoły przechowującej podstawowe informacje o niej (takie jak informacje na temat pracowników i uczniów), wraz ze stworzeniem prostych funkcjonalności umożliwiającej jej obsługę.

- Podstawowe założenia oraz ograniczenia:

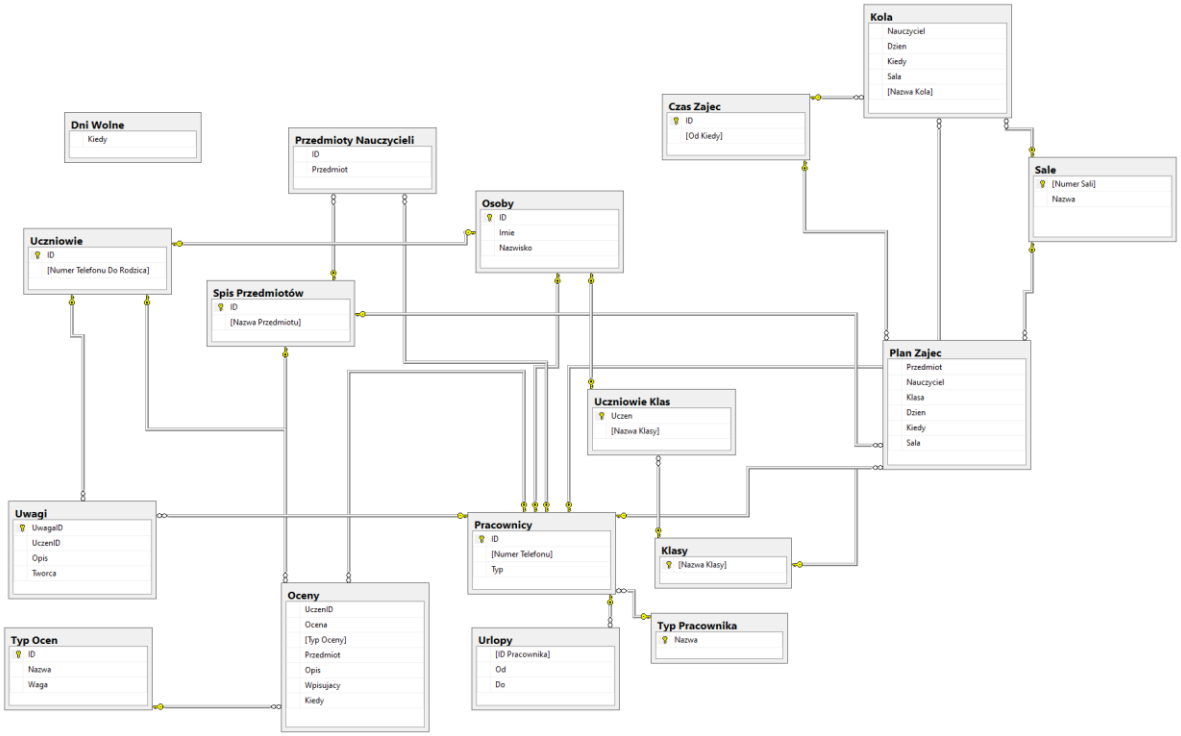
Głównym celem projektu jest stworzenie bazy danych szkoły, która przechowuje informacje na temat pracowników budynku, informacji na temat uczniów, ocen uczniów i planu zajęć. Projekt jest z założenia ograniczony tylko do pokazania podstawowych funkcji z możliwością rozwoju bazy danych w przyszłości do odpowiednich potrzeb. Jego przeznaczeniem jest używanie go w danej placówce szkolnej w celach administracyjnych budynku jak i zarządzanie tzw. "E-dziennikiem" jako system wewnętrzny. Dzięki temu oduzależniamy daną placówkę od czynników zewnętrznych, takich jak osobne "E-dzienniki".

Ograniczyliśmy projekt do podstaw, tzw. dane przykładowe są podane tylko dla jednej klasy i wypełnione danymi tylko dla jednej osoby ze względu na skalę projektu (w rzeczywistości taka baza miałaby np. tysiące rekordów z ocenami). Więc prezentację projektu utrzymujemy w założeniu do minimum funkcjonalności na przykładowych danych. Dodatkowo kolejnym ograniczeniem jak już wcześniej zostało powiedziane jest skala projektu - znacznie uprościliśmy system, a ściślej mówiąc = stworzyliśmy podstawę pod dalszy rozwój.

- Strategia pielęgnacji bazy danych:

Co tydzień pełna kopia zapasowa bazy danych. Codziennie po północy: różnicowa kopia zapasowa.
Taka strategia pozwoli zniwelować skutki ewentualnych awarii – i odzyskać utracone w skutek nań dane.

- Diagram ER oraz schemat bazy danych:



Oceny

Column Name	Data Type	Allow Nulls
UczenID	int	☐
Ocena	tinyint	☐
[Typ Oceny]	tinyint	☐
Przedmiot	tinyint	☐
Opis	text	☑
Wpisujacy	int	☐
Kiedy	datetime2(7)	☑
		☐

Osoby

Column Name	Data Type	Allow Nulls
ID	int	☐
Imie	varchar(255)	☐
Nazwisko	varchar(255)	☐
		☐

Plan Zajec

Column Name	Data Type	Allow Nulls
Przedmiot	tinyint	☐
Nauczyciel	int	☐
Klasa	varchar(10)	☐
Dzien	tinyint	☐
Kiedy	tinyint	☐
Sala	int	☐
		☐

Uwagi

Column Name	Data Type	Allow Nulls
UwagaID	int	☐
UczenID	int	☐
Opis	text	☐
Tworca	int	☐
		☐

Czas Zajec

Column Name	Data Type	Allow Nulls
ID	tinyint	☐
[Od Kiedy]	varchar(20)	☑
		☐

Dni Wolne

Column Name	Data Type	Allow Nulls
Kiedy	datetime	☑
		☐

Klasy

Column Name	Data Type	Allow Nulls
[Nazwa Klasy]	varchar(10)	☐
		☐

Koła

Column Name	Data Type	Allow Nulls
Nauczyciel	int	☐
Dzien	tinyint	☐
Kiedy	tinyint	☐
Sala	int	☐
[Nazwa Kola]	varchar(255)	☐
		☐

Przedmioty Nauczycieli

Column Name	Data Type	Allow Nulls
ID	int	☐
Przedmiot	tinyint	☐
		☐

Pracownicy

Column Name	Data Type	Allow Nulls
ID	int	☐
[Numer Telefonu]	varchar(16)	☑
Typ	varchar(255)	☐
		☐

Sale

Column Name	Data Type	Allow Nulls
[Numer Sali]	int	☐
Nazwa	varchar(255)	☑
		☐

Schemat relacji bazy danych SzkolaDB

Urlopy

Column Name	Data Type	Allow Nulls
[ID Pracownika]	int	☐
Od	time(7)	☑
Do	time(7)	☑
		☐

Uczniowie Klas

Column Name	Data Type	Allow Nulls
Uczen	int	☐
[Nazwa Klasy]	varchar(10)	☐
		☐

Uczniowie

Column Name	Data Type	Allow Nulls
ID	int	☐
[Numer Telefonu Do Rodzica]	varchar(16)	☑
		☐

Typ Pracownika

Column Name	Data Type	Allow Nulls
Nazwa	varchar(255)	☐
		☐

Typ Ocen

Column Name	Data Type	Allow Nulls
ID	tinyint	☐
Nazwa	varchar(255)	☐
Waga	tinyint	☑
		☐

Spis Przedmiotów

Column Name	Data Type	Allow Nulls
ID	tinyint	☐
[Nazwa Przedmiotu]	varchar(255)	☐
		☐

Ścisła dokumentacja. Utworzone tabele oraz ich opis.

Zgodnie z wymaganiami projektowymi, mieliśmy utworzyć minimum 16 tabel (po 8 na osobę). Ostatecznie wykonaliśmy 17 tabel, które zostaną poniżej nieco opisane.
W każdym z punktów przyjmujemy układ, kolejno: kwerenda tworząca tabelę; kwerenda ze wstawieniem przykładowych danych (w większości z tabel); krótki opis tabeli.

1. Typ pracownika

```
-- Stworzenie tabeli Typ Pracownika
CREATE TABLE [Typ Pracownika] (
    Nazwa VARCHAR(255) NOT NULL PRIMARY KEY
);

-- Wstawienie przykładowych rekordów do tabeli
INSERT INTO [Typ Pracownika](Nazwa) VALUES
('Dyrektor'),
('Administracja'),
('Ekipa sprzątająca'),
('Nauczyciel')
```

Tabela zawierająca nazwę typu pracownika. W domyśle ma przyjmować stanowisko administracyjne lub zawodowe danego pracownika (np. nauczyciel lub dyrektor).

2. Pracownicy

```
-- Stworzenie tabeli Pracownicy
CREATE TABLE Pracownicy (
    ID INT REFERENCES Osoby(ID) ON DELETE CASCADE PRIMARY KEY NOT NULL,
    [Numer Telefonu] VARCHAR(16),
    Typ VARCHAR(255) REFERENCES [Typ Pracownika](Nazwa) NOT NULL
);

INSERT INTO Pracownicy(ID, [Numer Telefonu], Typ) VALUES
(1,NULL,'Dyrektor'),
(2,'152512241','Administracja'),
(3,'612613612','Administracja'),
(4,'125151111','Ekipa sprzątająca'),
(5,NULL,'Ekipa sprzątająca'),
(6,NULL,'Nauczyciel'),
(7,NULL,'Nauczyciel'),
(8,'889741523','Nauczyciel'),
(9,'352776418','Nauczyciel'),
(10,'556789921','Nauczyciel'),
(11,'663992881','Nauczyciel'),
(12,'559876543','Nauczyciel'),
(13,'886543112','Nauczyciel'),
(14,'534665410','Nauczyciel'),
(15,'643997612','Nauczyciel'),
(16,'559813456','Nauczyciel'),
(17,'755086732','Nauczyciel'),
(18,'901887322','Nauczyciel'),
(19,'654337820','Nauczyciel'),
(20,'998451002','Nauczyciel')
```

Tabela zawierająca podstawowe dane o pracowniku: jego ID, numer telefonu oraz typ (piastowane stanowisko).

3. Typ ocen

```
-- Stworzenie tabeli Typ Ocen
CREATE TABLE [Typ Ocen] (
    ID TINYINT IDENTITY(1,1) PRIMARY KEY,
    Nazwa VARCHAR(255) NOT NULL,
    Waga TINYINT DEFAULT 1
);

INSERT INTO [Typ Ocen](Nazwa, Waga) VALUES
('Aktywność',1),
('Kartkówka',2),
('Sprawdzian',3),
('Projekt',3),
('Praca domowa',1)
```

Tabela zawierająca typy ocen wraz z ich wagami (liczbową „silnością” danej oceny).

4. Czas zajęć

```
-- Stworzenie tabeli Czas Zajec
CREATE TABLE [Czas Zajec] (
    ID TINYINT IDENTITY(1,1) PRIMARY KEY,
    [Od Kiedy] VARCHAR(20)

)

INSERT INTO [Czas Zajec]([Od Kiedy]) VALUES
('8:00'),
('8:55'),
```

```
('9:50'),
('10:45'),
('11:40'),
('12:45'),
('13:40'),
('14:35')
```

Tabela zawierająca z informacje o godzinach rozpoczęcia kolejnych zajęć lekcyjnych.

5. Sale

```
-- Stworzenie tabeli Sale
CREATE TABLE Sale (
    [Numer Sali] INT PRIMARY KEY NOT NULL,
    Nazwa VARCHAR(255)
)

INSERT INTO Sale([Numer Sali], Nazwa) VALUES
('101', 'Sala językowa'),
('102', 'Sala językowa'),
('103', 'Sala językowa'),
('104', 'Sala językowa'),
('105', 'Sala językowa'),
('106', 'Sala od matematyki'),
('107', 'Sala od matematyki'),
('108', 'Sekretariat'),
('109', 'Dyrekcja')
```

Tabela zawierająca informacje o salach: ich numerze i danej nazwie (typu „sala językowa”).

6. Dni wolne

```
-- Stworzenie tabeli Dni Wolne
CREATE TABLE [Dni Wolne] (
    Kiedy DATETIME
)
```

```
INSERT INTO [Dni Wolne](Kiedy) VALUES
('20200108'),
('20200121'),
('20200214'),
('20200312'),
('20200421'),
('20200921')
```

Tabela z informacjami o dniach wolnych od nauki szkolnej.

7. Osoby

```
-- Stworzenie tabeli Osoby
CREATE TABLE Osoby (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Imie VARCHAR(255) NOT NULL,
    Nazwisko VARCHAR(255) NOT NULL
)

INSERT INTO Osoby(Imie, Nazwisko) VALUES
('Zenek','Martyniuk'),
('Władysław','Jagiełło'),
('Jan','Seriusz'),
('Ken','Damino'),
('Jan','Anatomiusz'),
('Maciej','Jankowski'),
('Karol','Kanodziej'),
('Oskar','Przybylski'),
('Artur','Krol'),
('Henryk','Sienkiewicz'),
('Matylda','Nowak'),
('Lucyna','Moskowiak'),
('Aleksandra','Bytomska'),
('Nadia','Milowicz'),
('Agnieszka','Lipka'),
('Tatiana','Zlotko'),
('Honorata','Siemowicz'),
('Milena','Anastowicz'),
('Urszula','Kinek'),
('Irena','Portowicz'),
('Carlos','Kelly'),
('Harold','James'),
('Henry','Martin'),
('Earl','Powell'),
('Martin','Nelson'),
('Brian','Campbell'),
('Alice','Bailey')
```

Jedna z najważniejszych tabel. Zawiera ID, imię oraz nazwisko danej osoby w szkole, a zatem łączy pracowników i uczniów – umożliwia nam to użycie dziedziczenia w projekcie.

8. Urlopy

```
-- Stworzenie tabeli Urlopy
CREATE TABLE Urlopy (
    [ID Pracownika] INT REFERENCES Pracownicy(ID) ON DELETE CASCADE NOT
NULL,
    Od TIME,
    Do TIME
)
```

Tabela zawierające informacje o przedziale czasowym, kiedy pracownik o danym ID posiada urlop.

9. Spis przedmiotów

```
-- Stworzenie tabeli Przedmioty
CREATE TABLE [Spis Przedmiotów] (
    ID TINYINT IDENTITY(1,1) PRIMARY KEY,
    [Nazwa Przedmiotu] VARCHAR(255) NOT NULL
);

INSERT INTO [Spis Przedmiotów]([Nazwa Przedmiotu]) VALUES
('Język polski'),
('Język angielski'),
('Język niemiecki'),
('Matematyka'),
```

```
('Informatyka'),
('Fizyka'),
('Chemia'),
('Biologia'),
('Wychowanie fizyczne'),
('Historia')
```

Tabela wymieniająca nazwy wszystkich przedmiotów w naszej szkole.

10. Przedmioty nauczycieli

```
-- Stworzenie tabeli Przedmioty Nauczycieli
CREATE TABLE [Przedmioty Nauczycieli] (
    ID INT REFERENCES Pracownicy(ID) ON DELETE CASCADE NOT NULL,
    Przedmiot TINYINT REFERENCES [Spis Przedmiotów](ID) ON DELETE CASCADE
NOT NULL
);

INSERT INTO [Przedmioty Nauczycieli](ID, Przedmiot) VALUES
(6, 1),(6, 2),(7, 3),(8, 4),(9, 5),(10, 6),(11, 7),(12, 1),(13, 8),(14,
9),(15, 10)
```

Tabela informująca o przydzieleniu danego przedmiotu do nauczyciela (o danym ID).

11. Klasy

```
-- Stworzenie tabeli Klasy
CREATE TABLE Klasy (
    [Nazwa Klasy] VARCHAR(10) PRIMARY KEY NOT NULL
);

INSERT INTO Klasy([Nazwa Klasy]) VALUES
```

```
('1A'),('1B'),('1C'),('1D'),('1E'),('1F'),
('2A'),('2B'),('2C'),('2D'),('2E'),
('3A'),('3B'),('3C'),('3D'),('3E')
```

Tabela z nazwami klas występujących w szkole.

12. Uczniowie

```
-- Stworzenie tabeli Uczniowie
CREATE TABLE Uczniowie (
    ID INT REFERENCES Osoby(ID) ON DELETE CASCADE PRIMARY KEY NOT NULL,
    [Numer Telefonu Do Rodzica] VARCHAR(16)
);
INSERT INTO Uczniowie(ID, [Numer Telefonu Do Rodzica]) VALUES
(21,'674692364'),
(22,'336860628'),
(23,'947446770'),
(24,'091843778'),
(25,'588758593'),
(26,'480280287'),
(27,'146215678'),
(28,'978789825'),
(29,'118917758'),
(30,'369744971'),
(31,'676620670'),
(32,'851529601'),
(33,'416956734'),
(34,'667584689')
```

Tabela z listą uczniów: zawiera ich ID oraz numer telefonu do rodzica.

13. Uczniowie klas

```
-- Stworzenie tabeli Uczniowie Klas
```

```
CREATE TABLE [Uczniowie Klas] (
    Uczen INT REFERENCES Osoby(ID) PRIMARY KEY NOT NULL,
    [Nazwa Klasy] VARCHAR(10) REFERENCES Klasy([Nazwa Klasy]) NOT NULL
);

INSERT INTO [Uczniowie Klas]([Nazwa Klasy], Uczen) VALUES
('3B',21),('3B',22),('3E',23),('3A',24),('2E',25),('2D',26),('2C',27),('2B
',28),('2B',29),('1F',30),('3D',31),('3C',32),('1D',33),('2D',34)
```

Tabela zawierająca informacje o nazwie klasy danego ucznia.

14. Uwagi

```
-- Stworzenie tabeli Uwagi
CREATE TABLE Uwagi (
    UwagaID INT IDENTITY(1,1) PRIMARY KEY,
    UczenID INT REFERENCES Uczniowie(ID) ON DELETE CASCADE NOT NULL,
    Opis TEXT NOT NULL,
    Tworca INT REFERENCES Pracownicy(ID) NOT NULL
);

INSERT INTO Uwagi(UczenID, Opis, Tworca) VALUES
(21, 'Wdał się w bójkę z innym uczniem', 1),
(21, 'Grał na telefonie', 11)
```

Tabela z uwagami uczniów naszej szkoły.

15. Oceny

```
-- Stworzenie tabeli Oceny
CREATE TABLE Oceny (
    UczenID INT REFERENCES Uczniowie(ID) NOT NULL,
    Ocena TINYINT NOT NULL,
    [Typ Oceny] TINYINT REFERENCES [Typ Ocen](ID) NOT NULL,
```

```
    Przedmiot TINYINT REFERENCES [Spis Przedmiotów](ID) ON DELETE CASCADE
NOT NULL,
    Opis TEXT,
    Wpisujacy INT REFERENCES Pracownicy(ID) ON DELETE CASCADE NOT NULL,
    Kiedy DATETIME2 DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO Oceny(UczenID, Ocena, [Typ Oceny], Przedmiot, Opis, Wpisujacy,
Kiedy) VALUES
(21,6,7,11,'',8,'2018-06-23 07:30:20'),(21,5,4,12,'',9,'2018-06-23
07:30:20'),(21,3,4,13,'',19,'2018-06-23 07:30:20'),(21,6,1,5,'',7,'2018-
06-23 07:30:20'),(21,2,2,5,'',12,'2018-06-23
07:30:20'),(21,4,1,14,'',13,'2018-06-23 07:30:20')
```

Tabela zawierająca informacje o ocenach (m.in. o wartości oceny, nazwie przedmiotu itd.).

16. Plan zajęć

```
-- Stworzenie tabeli Plan Zajec
CREATE TABLE [Plan Zajec] (
    Przedmiot TINYINT REFERENCES [Spis Przedmiotów](ID) ON DELETE CASCADE
NOT NULL,
    Nauczyciel INT REFERENCES Pracownicy(ID) NOT NULL,
    Klasa VARCHAR(10) REFERENCES Klasy([Nazwa Klasy]) NOT NULL,
    Dzien TINYINT NOT NULL,
    Kiedy TINYINT REFERENCES [Czas Zajec](ID) NOT NULL,
    Sala INT REFERENCES Sale([Numer Sali]) NOT NULL
);
INSERT INTO SzkolaDB.dbo.[Plan Zajec](Przedmiot, Nauczyciel, Klasa, Dzien,
Kiedy, Sala) VALUES
(10,11,'1A',1,2,'204'),(14,18,'1A',1,3,'202'),(12,20,'1A',1,4,'202'),(2,8,
'1A',1,5,'204'),(14,7,'1A',1,6,'102'),(3,18,'1A',1,7,'203'),(7,19,'1A',1,8
,'102'),(1,6,'1A',1,9,'106'),(15,17,'1A',1,10,'104'),(2,6,'1A',2,3,'205'),
(1,15,'1A',2,4,'104'),(12,19,'1A',2,5,'301'),(16,8,'1A',2,6,'104'),(10,7,'
1A',2,7,'301'),(7,11,'1A',3,1,'110'),(16,13,'1A',3,2,'203'),(12,14,'1A',3,
3,'110'),(6,6,'1A',3,4,'103'),(12,13,'1A',3,5,'204'),(6,10,'1A',3,6,'110')
,(1,8,'1A',3,7,'201'),(14,11,'1A',4,4,'201'),(16,14,'1A',4,5,'110'),(11,8,
'1A',4,6,'204'),(12,14,'1A',4,7,'110'),(7,13,'1A',4,8,'203'),(12,10,'1A',4
,9,'205'),(1,18,'1A',4,10,'205'),(5,11,'1A',5,3,'301'),(10,19,'1A',5,4,'20
4'),(16,10,'1A',5,5,'201'),(7,12,'1A',5,6,'203'),(15,20,'1A',5,7,'204'),(1
6,9,'1A',5,8,'103'),(15,13,'1A',5,9,'205'),(13,9,'1A',5,10,'101'),(15,19,'
1A',5,11,'107')
```

Tabela zawierająca informacje o planach lekcji.

17. Koła

```
-- Stworzenie tabeli Kola
CREATE TABLE Kola (
    Nauczyciel INT REFERENCES Pracownicy(ID) ON DELETE CASCADE NOT NULL,
    Dzien TINYINT NOT NULL,
    Kiedy TINYINT REFERENCES [Czas Zajec](ID) NOT NULL,
    Sala INT REFERENCES Sale([Numer Sali]) NOT NULL,
    [Nazwa Kola] VARCHAR(255) NOT NULL
);
```

Tabela z kołami (np. naukowymi) w szkole.

Siedemnaście powyższych tabel stanowi podstawowe informacje o każdej szkole. Ponadto umożliwia tworzenie sensownych funkcji czy widoków. Wszystkie te możliwości składają się na finalny wynik projektu.

Widoki i funkcje. Kwerendy wraz z opisami.

Zgodnie z wymaganiami projektowymi, mieliśmy utworzyć 10 widoków lub funkcji.

Widoki

```
CREATE VIEW wyswietlanie_uczniow AS
SELECT O.Imie, O.Nazwisko, U.[Numer Telefonu Do Rodzica], K.[Nazwa Klasy]
FROM Osoby O JOIN Uczniowie U
ON O.ID = U.ID
JOIN [Uczniowie Klas] K
ON U.ID = K.Uczen
ORDER BY [Nazwa Klasy], Nazwisko, Imie
OFFSET 0 ROWS
```

Wypisanie uczniów w kolejności klas (od 1A licząc), wewnątrz danej klasy sortujemy za nazwiskiem, a następnie za imieniem

```
CREATE VIEW wyswietlanie_nauczycieli AS
```

```sql
SELECT O.Imie, O.Nazwisko, P.[Numer Telefonu]
FROM Osoby O LEFT JOIN Pracownicy P
ON O.ID = P.ID
WHERE P.Typ = 'Nauczyciel'
```

Wyświetla nauczycieli: ich imiona, nazwiska, numery telefonu.

```sql
CREATE VIEW hierarchia AS
SELECT T.Nazwa, O.Imie, O.Nazwisko
FROM Pracownicy P
JOIN [Typ Pracownika] T
ON P.Typ = T.Nazwa
JOIN Osoby O
ON O.ID = P.ID
ORDER BY CASE WHEN Typ = 'Dyrektor' THEN 1
                  WHEN Typ = 'Administracja' THEN 2
                  WHEN Typ = 'Nauczyciel' THEN 3
                  WHEN Typ = 'Ekipa Sprzątająca' THEN 4
                  ELSE 5
            END ASC
OFFSET 0 ROWS
```

Widok "hierarchia" przedstawia pracowników oraz ich stanowiska w kolejności: dyrektor, administracja, nauczyciel, ekipa sprzątająca - w zamyśle, widok ten przeznaczony jest do pokazania 'hierarchii ważności stanowiskowej' w szkole.

```sql
CREATE VIEW urlopy_pracownikow AS
SELECT O.Imie, O.Nazwisko,SUM(DATEDIFF(hour, U.Od ,U.do)) [Suma urlopów (w
godzinach)]
FROM Urlopy U
JOIN Osoby O
ON U.[ID Pracownika] = O.ID
GROUP BY IMIE, Nazwisko
ORDER BY SUM(DATEDIFF(hour, U.Od ,U.do)) DESC
OFFSET 0 ROWS
```

Widok przedstawia sumę godzinową urlopów każdego z pracowników (od pracownika, który tych urlopów ma godzinowo najmniej, do tego, który ma najwięcej).

Funkcje

```
GO
CREATE FUNCTION dbo.wyswietl_ucznia (@ID AS INT)
RETURNS TABLE

AS

RETURN
SELECT O.Imie, O.Nazwisko, U.[Numer Telefonu Do Rodzica], K.[Nazwa Klasy]
FROM Osoby O JOIN Uczniowie U
ON O.ID = U.ID
JOIN [Uczniowie Klas] K
ON U.ID = K.Uczen
WHERE U.ID = @ID

GO

-- przykładowe wywołanie funkcji (wypisanie danych ucznia o ID = 24):

SELECT * FROM dbo.wyswietl_ucznia(24)
```

Funkcja wyświetlająca dane ucznia o ID danym argumentem.

```
GO
CREATE FUNCTION dbo.wypisz_typem (@Typ AS VARCHAR(255))
RETURNS TABLE

AS

RETURN
```

```
SELECT O.Imie, O.Nazwisko, P.[Numer Telefonu], Typ
FROM Osoby O LEFT JOIN Pracownicy P
ON O.ID = P.ID
WHERE Typ = @Typ

GO


-- przykładowe wywołanie powyższej funkcji:

SELECT * FROM dbo.wypisz_typem('Administracja')
```

Funkcja wypisuje pracowników o typie podanym w argumencie.

```
GO
CREATE FUNCTION dbo.wypisz_oceny (@ID AS INT)
RETURNS TABLE

AS

RETURN
SELECT Oc.Ocena [Ocena], T.Nazwa, T.Waga, Oc.Opis, S.[Nazwa Przedmiotu],
Os.Imie [Imie Nauczyciela], Os.Nazwisko[Nazwisko Nauczyciela], Oc.Kiedy
FROM Osoby O
LEFT JOIN Oceny Oc
ON Oc.UczenID = O.ID
LEFT JOIN [Typ Ocen] T
ON T.ID = Oc.[Typ Oceny]
LEFT JOIN [Spis Przedmiotów] S
ON S.ID = Oc.Przedmiot
LEFT JOIN Osoby Os
ON Os.ID = Oc.Wpisujacy
WHERE O.ID = @ID AND Oc.Ocena IS NOT NULL

GO


-- przykładowe wywołanie funkcji, wypisujemy dane ucznia o ID = 21, wraz z
jego ocenami:

SELECT * FROM dbo.wypisz_oceny(21)
```

Funkcja wypisująca oceny ucznia o ID danym argumentem, wraz ze szczegółami danych ocen.

```
GO
CREATE FUNCTION dbo.srednia_wazona (@ID AS INT, @Przedmiot AS
NVARCHAR(255))
RETURNS TABLE

AS
RETURN
SELECT (1. * SUM(Ocena * W)/SUM(W)) as Srednia FROM
(
SELECT O.ID iden, O.Imie, O.Nazwisko, Oc.Ocena Ocena, T.Nazwa, T.Waga W,
S.[Nazwa Przedmiotu] naz
FROM (Osoby O
LEFT JOIN Oceny Oc
ON Oc.UczenID = O.ID
LEFT JOIN [Typ Ocen] T
ON T.ID = Oc.[Typ Oceny]
LEFT JOIN [Spis Przedmiotów] S
ON S.ID = OC.Przedmiot)
) x
WHERE (iden = @ID AND naz = @Przedmiot)
GO


-- przykład wyświetlający średnią ważoną ocen ucznia o ID = 21 z języka
niemieckiego:

SELECT * FROM dbo.srednia_wazona(21, 'Język niemiecki')
```

Funkcja obliczająca średnią ważoną ocen danego ucznia z danego przedmiotu (argumenty to ID ucznia i nazwa przedmiotu).

```
GO
CREATE FUNCTION dbo.wypisz_uwagi (@ID AS INT)
RETURNS TABLE

AS

RETURN
SELECT O.Imie, O.Nazwisko, G.Opis [Opis Uwagi],
```

```
P.Imie [Imie Nauczyciela], P.Nazwisko [Nazwisko Nauczyciela]
FROM Osoby O
LEFT JOIN Uczniowie U
ON O.ID = U.ID
LEFT JOIN Uwagi G
ON O.ID = G.UczenID
LEFT JOIN Osoby P
ON G.Tworca = P.ID
WHERE O.ID = @ID

GO


-- przykład wywołania funkcji, wypisze uwagi ucznia o ID = 23:

SELECT * FROM dbo.wypisz_uwagi(23)
```

Funkcja odpowiedzialna za wypisanie wszystkich uwag danego ucznia (którego ID podamy w argumencie) wraz z jej opisem i danymi autora.

```
GO
CREATE FUNCTION dbo.plan_lekcji (@Klasa AS VARCHAR(10))
RETURNS TABLE

AS

RETURN
SELECT SP.[Nazwa Przedmiotu] [Przedmiot], O.Imie [Imie Nauczyciela],
O.Nazwisko [Nazwisko Nauczyciela], PZ.Nauczyciel, PZ.Dzien, PZ.Kiedy,
PZ.Sala
FROM SzkolaDB.dbo.[Plan Zajec] PZ
JOIN SzkolaDB.dbo.[Spis Przedmiotów] SP
ON PZ.Przedmiot = SP.ID
JOIN SzkolaDB.dbo.[Osoby] O
ON PZ.Nauczyciel = O.ID
WHERE Klasa = @Klasa

GO


-- przykładowe wywołanie powyższej funkcji:

SELECT * FROM dbo.plan_lekcji('1A')
```

Powyższa funkcja wypisuje plan lekcji danej klasy.

Procedury składowane.

Zgodnie z wymaganiami projektowymi, stworzyliśmy pięć procedur. Każda z nich jest zabezpieczona przed najpopularniejszymi błędami, które mogłyby się przytrafić.

```
GO
CREATE PROC dbo.dodaj_pracownika

@Imie VARCHAR(255) = NULL,
@Nazwisko VARCHAR(255) = NULL,
@Numer VARCHAR(16),
@Typ  VARCHAR(255) = NULL,
@Przedmiot TINYINT = 0

AS

DECLARE @blad AS NVARCHAR(500);

IF @Imie IS NULL OR @Nazwisko IS NULL OR @Typ IS NULL
OR (@Typ = 'Nauczyciel' AND @Przedmiot = 0)
BEGIN
     SET @blad = 'Błędne dane, sprawdź podane argumenty.';
     RAISERROR(@blad, 16,1);
     RETURN;
END

DECLARE @cd INT
SELECT @cd = COUNT(*) FROM Osoby
SET @cd = @cd + 1

INSERT INTO Osoby(Imie, Nazwisko)
VALUES (@Imie, @Nazwisko);
```

```
INSERT INTO Pracownicy(ID, [Numer Telefonu], Typ)
VALUES(@cd, @Numer, @Typ)

-- jeśli to nauczyciel, to dokładamy jego przedmiot do danej tabeli
IF @Typ = 'Nauczyciel'
INSERT INTO [Przedmioty Nauczycieli](ID, Przedmiot)
VALUES(@cd, @Przedmiot)

GO

-- przykładowe wywołanie powyższej procedury, dodanie pracownika
(nauczyciela) Tomasza Zauchę:

EXEC dbo.dodaj_pracownika @Imie = 'Tomasz', @Nazwisko = 'Zaucha',
@Numer = '523456789', @Typ = 'Nauczyciel', @Przedmiot = 5
GO
```

Procedura ta dodaje nowego pracownika do naszej bazy, poprzez dodanie do tabel Osoby, Pracownicy (i ewentualnie Przedmioty Nauczycieli) osoby o tym samym numerze ID.

```
CREATE PROC dbo.dodaj_ucznia

@Imie VARCHAR(255) = NULL,
@Nazwisko VARCHAR(255) = NULL,
@Numer VARCHAR(16),
@Klasa  VARCHAR(10) = NULL

AS

DECLARE @blad AS NVARCHAR(500);

IF @Imie IS NULL OR @Nazwisko IS NULL OR @Klasa IS NULL

BEGIN
      SET @blad = 'Błędne dane, sprawdź podane argumenty.';
      RAISERROR(@blad, 16,1);
    RETURN;
END

DECLARE @cd INT
SELECT @cd = COUNT(*) FROM Osoby
SET @cd = @cd + 1

INSERT INTO Osoby(Imie, Nazwisko)
```

```
VALUES (@Imie, @Nazwisko);

INSERT INTO Uczniowie(ID, [Numer Telefonu Do Rodzica])
VALUES(@cd, @Numer)

INSERT INTO [Uczniowie Klas]([Nazwa Klasy], Uczen)
VALUES(@Klasa,@cd)

GO

-- przykładowe wywołanie powyższej procedury (dodanie ucznia Tomka
Mikulskiego z klasy 2A)

EXEC dbo.dodaj_ucznia @Imie = 'Tomek', @Nazwisko = 'Mikulski',
@Numer = '123456789', @Klasa = '2A'
GO
```

Procedura dodaje nowego ucznia do naszej bazy posiadającego swoje dane osobiste oraz klasę.

```
GO
CREATE PROC dbo.dodaj_uwage

@ID INT = NULL,
@Opis TEXT = NULL,
@Tworca INT = NULL

AS

DECLARE @blad AS NVARCHAR(500);

IF @ID IS NULL OR @Opis IS NULL OR @Tworca IS NULL OR
@ID NOT IN (SELECT ID FROM Uczniowie) OR @Tworca NOT IN (SELECT ID FROM
Pracownicy)
BEGIN
     SET @blad = 'Błędne dane, sprawdź podane argumenty.';
     RAISERROR(@blad, 16,1);
     RETURN;
END

INSERT INTO Uwagi
VALUES (@ID, @Opis, @Tworca)
```

```
GO


-- przykład:

EXEC dbo.dodaj_uwage @ID = 26, @Opis = 'Naganne zachowania ucznia podczas
lekcji',
@Tworca = 8
GO
```

Procedura wstawiania uwagi, jako argumenty podajemy ID ucznia, opis uwagi oraz ID twórcy (pracownika) wstawiającego uwagę
-- zwracanie błędu, gdy: jakikolwiek argument jest NULLem LUB gdy ID ucznia lub pracownika jest niepoprawne (nie istnieje w bazie).

```
GO
CREATE PROC dbo.dodaj_ocene

@ID INT = NULL,
@Ocena TINYINT = NULL,
@Typ TINYINT = NULL,
@Przedmiot TINYINT = NULL,
@Opis TEXT = NULL,
@Wpisujacy INT = NULL

AS

DECLARE @blad AS NVARCHAR(500);

IF @ID IS NULL OR @Ocena IS NULL OR @Typ IS NULL
OR @Przedmiot IS NULL OR @Wpisujacy IS NULL

BEGIN
      SET @blad = 'Błędne dane, sprawdź podane argumenty.';
      RAISERROR(@blad, 16,1);
    RETURN;
END

IF @Ocena > 6 OR @Ocena < 1

BEGIN
      SET @blad = 'Zła wartość oceny';
      RAISERROR(@blad, 16,1);
```

```
    RETURN;
END

INSERT INTO Oceny(UczenID, Ocena, [Typ Oceny], Przedmiot, Opis, Wpisujacy,
Kiedy)
VALUES (@ID, @Ocena, @Typ, @Przedmiot, @Opis, @Wpisujacy, GETDATE());

GO


-- przykład, dodajemy uczniowi o ID = 21 ocenę 4 z podanymi innymi danymi,
daty nie trzeba podawać:

EXEC dbo.dodaj_ocene @ID = 21, @Ocena = 4, @Typ = 2, @Przedmiot = 3,
@Opis = 'Opis oceny', @Wpisujacy = 2
GO
```

Procedura dodająca ocenę uczniowi o danym ID.

```
GO

CREATE PROC dbo.dodaj_dzien_wolny

@Kiedy DATETIME

AS

DECLARE @blad AS NVARCHAR(500);


IF @Kiedy IN (SELECT * FROM dbo.[Dni Wolne])

BEGIN

     SET @blad = 'Podana data istnieje juz w tabeli dni wolnych.';

     RAISERROR(@blad, 17, 1);

     RETURN;

END


INSERT INTO dbo.[Dni Wolne]

VALUES (@Kiedy)

GO


-- przykład:
```

```
EXEC dbo.dodaj_dzien_wolny @Kiedy = '2007-05-08 12:35:29'
GO
```

Procedura dodawania dnia wolnego, jako argument podajemy datetime, zwracanie błędu, gdy: podany datetime istnieje.

Wyzwalacze (triggery).

Proste wyzwalacze. Głównie informują o zdarzeniach zrealizowanych przez użytkownika, takich jak np. dodanie nowego ucznia (poprzez automatyczne wypisanie komunikatu). Dodatkowo wyświetlany jest komunikat o liczebności danej tabeli (np. o aktualnej liczbie uczniów w bazie) Podobna realizacja dostępna była w niektórych starszych wersjach dzienników elektronicznych.

```
IF OBJECT_ID('dodano_ucznia', 'TR') IS NOT NULL
      DROP TRIGGER dodano_ucznia
GO

CREATE TRIGGER dodano_ucznia ON Uczniowie
AFTER INSERT
AS BEGIN

DECLARE @msg NVARCHAR(55) = NULL
DECLARE @nr INT
SELECT @nr = COUNT(*) FROM dbo.Uczniowie
SET @msg = 'Mamy obecnie ' + CAST(@nr AS VARCHAR(10)) + ' uczniów w naszej
bazie danych.'
SELECT 'POMYŚLNIE DODANO UCZNIA' [Komunikat 1], @msg [Komunikat 2]

END
GO
```

Po pomyślnym dodaniu ucznia ukazuje się komunikat o liczbie uczniów w naszej bazie danych (szkole).

```
IF OBJECT_ID('dodano_pracownika', 'TR') IS NOT NULL
      DROP TRIGGER dodano_pracownika
GO

CREATE TRIGGER dodano_pracownika ON Pracownicy
AFTER INSERT
AS BEGIN

DECLARE @msg NVARCHAR(55) = NULL
DECLARE @nr INT
SELECT @nr = COUNT(*) FROM dbo.Pracownicy
SET @msg = 'Mamy obecnie ' + CAST(@nr AS VARCHAR(10)) + ' pracowników w
naszej bazie danych.'
SELECT 'POMYŚLNIE DODANO PRACOWNIKA' [Komunikat 1], @msg [Komunikat 2]

END
GO
```

Po pomyślnym dodaniu pracownika ukazuje się komunikat o liczbie pracowników w naszej bazie danych (szkole).

```
IF OBJECT_ID('dodano_ocene', 'TR') IS NOT NULL
      DROP TRIGGER dodano_ocene
GO

CREATE TRIGGER dodano_ocene ON Oceny
AFTER INSERT
AS BEGIN
SELECT 'POMYŚLNIE DODANO NOWĄ OCENĘ'
END
GO
```

Informacja o pomyślnym dodaniu nowej oceny.

```
IF OBJECT_ID('dodano_ucznia_do_klasy', 'TR') IS NOT NULL
DROP TRIGGER dodano_ucznia_do_klasy
GO

CREATE TRIGGER dodano_ucznia_do_klasy ON [Uczniowie Klas]
AFTER INSERT
AS BEGIN
SELECT [Nazwa Klasy], COUNT([Nazwa Klasy]) AS [Ilosc uczniow w klasie]
FROM [Uczniowie Klas] GROUP BY [Nazwa Klasy]
END
GO
```

Informacja o pomyślnym dodaniu nowego ucznia do klasy.

```
IF OBJECT_ID('usunieto_ucznia_do_klasy', 'TR') IS NOT NULL
DROP TRIGGER usunieto_ucznia_do_klasy
GO

CREATE TRIGGER usunieto_ucznia_do_klasy ON [Uczniowie Klas]
AFTER DELETE
AS BEGIN
SELECT [Nazwa Klasy], COUNT([Nazwa Klasy]) AS [Ilosc uczniow w klasie]
FROM [Uczniowie Klas] GROUP BY [Nazwa Klasy]
END
GO
```

Komunikat po usunięciu ucznia z klasy.

Aplikacja kliencka

"E-dziennik" to aplikacja kliencka przedstawiająca dane SzkolyDB w formie, jak powinna zostać użyta. Logując się do e-dziennika (w tym przypadku z góry jesteśmy Carlosem Kelly) otrzymujemy podstawowe informacje o uczniu, jego planu lekcji, jego ocenach i uwagach. Aplikacja pokazuje szczegóły takie jak: waga danej oceny, kto ją wystawił, w której sali odbywają się zajęcia itd. Aplikacja została napisana za pomocą framework'a do JavaScript'u o nazwie Electron. Dodatkowo aplikacja ma możliwość dalszego rozwoju, dzięki czemu można by było stworzyć interfejs dla nauczyciela do wstawiania ocen czy możliwość ukazywania zadania domowego (o ile stworzymy odpowiednią strukturę w SzkolaDB).