Wykonanie: Strach Maciej,

Tytuł: Uproszczona baza danych szkoły śreniej

Opis projektu:
Projekt zawiera budowę podstawowej bazy danych przeznaczonej do przechowywania informacji potrzebnych do funkcjonowania szkoły, stworzone z myślą o szkole średniej (technikum, liceum, szkoła branżowa). Wstępnie składa się z 10 tabel mających zawierać dane o: uczniach, nauczycielach, klasach, przedmiotach, ocenach, zajęciach, salach lekcyjnych, rodzicach uczniów. Bazę danch znormalizowano w miarę możliwości.
Dla uproszczenia przyjęto następujące założenia:
Nie ma dwóch nauczycieli o tym samym imieniu i nazwisku, rodzice mają ten sam adres co ich dziecko, nie planujemy zastępstw w bazie danych, nie ma ocen z plusem lub typu +/-.

0. Utworzenie bazy danych

```
DROP DATABASE IF EXISTS Szkola;

CREATE DATABASE Szkola;

USE Szkola;
```

1. Tabele

1.1. Tabela `Uczen`

Tabela `Uczen` przechowuje podstawowe informacje o każdym aktualnie uczęszczającym uczniu, jej kluczem głównym jest kolumna `UczenId`, której wartości są generowane automatycznie dzięki sekwencji `SEQ_UczenId`.

```
CREATE SEQUENCE SEQ_UczenId AS INT START
WITH
    1 INCREMENT BY 1 MINVALUE 1 CACHE 10;

CREATE TABLE
    Uczen (
        UczenId INT NOT NULL DEFAULT NEXT VALUE FOR SEQ_UczenId,
        KlasaId INT,
        Imie VARCHAR(30) NOT NULL,
        Nazwisko VARCHAR(50) NOT NULL,
        PESEL VARCHAR(11) NOT NULL,
        Miejscowosc VARCHAR(40) NOT NULL,
        Ulica VARCHAR(60),
        NumerDomu VARCHAR(10) NOT NULL,
        KodPocztowy VARCHAR(6) NOT NULL
    );

ALTER TABLE Uczen ADD CONSTRAINT PK_UczenId PRIMARY KEY (UczenId);
```

1.2. Tabela `Nauczyciel`

Tabela `Nauczyciel`, podobnie jak `Uczen`, zawiera podstawawe informacje o pracujących nauczycielach w szkole. Każdemu jest przypisywany unikalny ID Nauczyciela za pomocą sekwencji. Pole `NauczycielId` jest kluczem podstawowym tabeli.

```
CREATE SEQUENCE SEQ_NauczycielId AS INT START
WITH
    1 INCREMENT BY 1 MINVALUE 1 CACHE 10;

CREATE TABLE
    Nauczyciel (
        NauczycielId INT NOT NULL DEFAULT NEXT VALUE FOR SEQ_NauczycielId,
        Imie VARCHAR(30) NOT NULL,
        Nazwisko VARCHAR(50) NOT NULL,
        PESEL VARCHAR(11) NOT NULL,
        Miejscowosc VARCHAR(40) NOT NULL,
        Ulica VARCHAR(60),
        NumerDomu VARCHAR(10) NOT NULL,
        KodPocztowy VARCHAR(6) NOT NULL
    );

ALTER TABLE Nauczyciel ADD CONSTRAINT PK_NauczycielId PRIMARY KEY
(NauczycielId);
```

1.3. Tabela `Rodzic`

Tabela `Rodzic` składuje dane o rodzicach ucznia. Przechowuje imiona, nazwiska, telefony rodziców, dla uproszczenia zakładamy że adres rodziców/rodzica jest ten sam co ucznia. Tabela nie zawiera klucza podstawowego, ale ma klucz obcy na kolumnie `UczenId`, która odwołuje się do tabeli `Uczen` - relacja „1 do n” z „n” po stronie rodzica (rodzic może mieć dwójkę dzieci w tej samej szkole), w przypadku usuni ęcia ucznia z tabeli rodzice również zostają usunięci. Dodatkowo tabela ma ograniczenia: wymagany jest pełny zestaw informacji przynajmniej do jednego rodzica.

```
CREATE TABLE
    Rodzic (
        UczenId INT NOT NULL,
        ImieMatki VARCHAR(30),
        NazwiskoMatki VARCHAR(50),
        TelMatki VARCHAR(12),
        ImieOjca VARCHAR(30),
        NazwiskoOjca VARCHAR(50),
        TelOjca VARCHAR(12)
    );

ALTER TABLE Rodzic ADD CONSTRAINT FK_Rodzic_UczenId FOREIGN KEY (UczenId)
REFERENCES Uczen (UczenId) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE Rodzic ADD CONSTRAINT CHK_Rodzice CHECK (
    (
        ImieMatki IS NOT NULL
        AND NazwiskoMatki IS NOT NULL
        AND TelMatki IS NOT NULL
    )
    OR (
        ImieOjca IS NOT NULL
        AND NazwiskoOjca IS NOT NULL
        AND TelOjca IS NOT NULL
    )
);
```

1.4. Tabela `Przedmiot`

Ta tabela przechowuje jedynie nazwy przedmiotów i każdemu z nich przypisuje jednoznacznie określającą wartość w kolumnie `PrzedmiotId` za pomocą sekwencji `SEQ_PrzedmiotId`.

```
CREATE SEQUENCE SEQ_PrzedmiotId AS INT START
WITH
    1 INCREMENT BY 1 MINVALUE 1 CACHE 10;

CREATE TABLE
    Przedmiot (
        PrzedmiotId INT NOT NULL DEFAULT NEXT VALUE FOR SEQ_PrzedmiotId,
        NazwaPrzedmiotu VARCHAR(30) NOT NULL UNIQUE
    );

ALTER TABLE Przedmiot ADD CONSTRAINT PK_Przedmiot PRIMARY KEY (PrzedmiotId);
```

1.5. Tabela `PrzedmiotNauczyciel`

Tabela ma na celu powiązać nauczycieli z przedmiotami, których nauczają. Tabela ma dwie kolumny będące kluczami obcymi do kolumn w tabelach `Nauczyciel` i `Przedmiot`. Powstała z powodu tego, że nauczyciel może uczyć dwóch przedmiotów, więc jest to tabela dla relacji „n do n”.

```
CREATE TABLE
    PrzedmiotNauczyciel (
        PrzedmiotId INT NOT NULL,
        NauczycielId INT NOT NULL
    );

ALTER TABLE PrzedmiotNauczyciel ADD CONSTRAINT
FK_PrzedmiotNauczyciel_PrzedmiotId FOREIGN KEY (PrzedmiotId) REFERENCES
Przedmiot (PrzedmiotId) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE PrzedmiotNauczyciel ADD CONSTRAINT
FK_PrzedmiotNauczyciel_NauczycielId FOREIGN KEY (NauczycielId) REFERENCES
Nauczyciel (NauczycielId) ON DELETE CASCADE ON UPDATE CASCADE;
```

1.6. Tabela `Sala`

Tabela `Sala` ma za zadanie przetrzymywać informacje o salach lekcyjnych. Zawiera informacje o jej lokalizacji i ilości uczniów, która się do niej zmieści. Dodatkowo można zaznaczyć czy jest to sala przystosowana do prowadzenia konkretnego przedmiotu (np. informatyczna) czy zwykła sala ogólna bez specjalnych materiałów naukowych (kolumna `CzyPrzedmiotowa` z wartością 0 dla „nie” i 1 dla „tak”, oraz kolumna `NazwaPrzedmiotu`). Można też przypisać jej opiekuna nauczyciela. Jej kluczem głównym jest kolumna `NrSali`, zawiera dwa klucze obce na kolumnach `OpiekunId` odwołujący się do tabeli `Nauczyciel` i `NazwaPrzedmiotu` odwołujący się do tabeli `Przedmiot`. Obie relacje odpowiednio się zachowują w przypadku usunięcia nauczyciela lub przedmiotu w ich tabelach. Dodatkowo ograniczenie CHECK `CHK_Przedmiot` sprawdza ,czy jeżeli sala jest przedmiotowa, to czy podano nazwę przedmiotu, którego w niej należy uczyć.

```
CREATE TABLE
    Sala (
        NrSali INT NOT NULL,
        Pietro INT NOT NULL,
        LiczbaMiejsc INT NOT NULL,
        CzyPrzedmiotowa BIT NOT NULL,
        NazwaPrzedmiotu VARCHAR(30),
        OpiekunId INT
    );

ALTER TABLE Sala ADD CONSTRAINT PK_NrSali PRIMARY KEY (NrSali);

ALTER TABLE Sala ADD CONSTRAINT FK_Sala_OpiekunId FOREIGN KEY (OpiekunId)
REFERENCES Nauczyciel (NauczycielId) ON DELETE SET NULL ON UPDATE CASCADE;

ALTER TABLE Sala ADD CONSTRAINT FK_Sala_NazwaPrzedmiotu FOREIGN KEY
(NazwaPrzedmiotu) REFERENCES Przedmiot (NazwaPrzedmiotu) ON DELETE SET NULL ON
UPDATE CASCADE;

ALTER TABLE Sala ADD CONSTRAINT CHK_Przedmiot CHECK (
    (
        CzyPrzedmiotowa = 1
        AND NazwaPrzedmiotu IS NOT NULL
    )
    OR (
        CzyPrzedmiotowa = 0
        AND NazwaPrzedmiotu IS NULL
    )
);
```

1.7. Tabela `Klasa`

Ta tabela przechowuje informacje o wszystkich klasach uczniów znajdujących się w szkole. Zawiera nazwę klasy i jej oznaczenie (`NrKlasy` np. 2 i `Oznaczenie` np. `B` dla klasy „II B”), czy jest to klasa licealna, technikalna, czy zawodowa (pole `Typ`); także informacje o wychowawcy i sali którą dana klasa się opiekuje. Kluczem postawowym tabeli jest atrybut `KlasaId` generowany automatycznie przez sekwencję. Łączy się relacją z tabelą `Nauczyciel` na polu `OpiekunId` i `Sala` na polu `NrSali`, aby uniknąć pomyłek. Tabela ma ograniczenie CHECK sprawdzające czy został podany odpowiedni numer (rok) i odpowiedni typ klasy.

```
CREATE SEQUENCE SEQ_KlasaId AS INT START
WITH
    1 INCREMENT BY 1 MINVALUE 1 CACHE 10;

CREATE TABLE
    Klasa (
        KlasaId INT NOT NULL DEFAULT NEXT VALUE FOR SEQ_KlasaId,
        NrKlasy INT NOT NULL,
        Oznaczenie VARCHAR(5),
        Typ VARCHAR(10),
        OpiekunId INT,
        NrSali INT
    );

ALTER TABLE Klasa ADD CONSTRAINT PK_KlasaId PRIMARY KEY (KlasaId);

ALTER TABLE Klasa ADD CONSTRAINT FK_Klasa_OpiekunId FOREIGN KEY (OpiekunId)
REFERENCES Nauczyciel (NauczycielId) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE Klasa ADD CONSTRAINT FK_Klasa_NrSali FOREIGN KEY (NrSali)
REFERENCES Sala (NrSali) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE Klasa ADD CONSTRAINT CHK_Klasa CHECK (
    (NrKlasy BETWEEN 1 AND 5)
    AND (Typ IN ('technikum', 'liceum', 'branżowa'))
);
```

Po utworzeniu tabeli `Klasa` możemy dodać na tabeli `Uczen` klucz obcy na kolumnie `KlasaId` pozwalający jednoznacznie przypisać ucznia do konkretnej klasy.

```
ALTER TABLE Uczen ADD CONSTRAINT FK_Uczen_KlasaId FOREIGN KEY (KlasaId)
REFERENCES Klasa (KlasaId) ON DELETE SET NULL ON UPDATE CASCADE;
```

Dyrektywy `NO ACTION` zostały nałożone na tą tabelę (relację), jak i następne ze względów późniejszego dodania obiektów takich jak funkcje procedury i wyzwalacze, a pozostałe wartości, czyli np. `SET NULL`, `CASCADE`, `SET DEFAULT` powodowały błędy.

1.8. Tabela `Ocena`

Wyżej wymieniona tabela ma za zadanie przechowywać wszystkie oceny wystawiane uczniom. Każda ocena ma swój identyfikator (Kolumna `OcenaId`) utworzony przez sekwencję i jest to szuczny klucz główny tabeli. Każda ocena ma przypisanego ucznia któremu wystawiono ocenę, przedmiot z którego została ocena wystawiona, wartość oceny sprawdzaną przez ograniczenie CHECK `CHK_Ocena`, przypisanego nauczyciela oraz datę i godzinę wstawienia oceny. Tabela ma 3 klucze obce definiujące relacje z tabelami `Uczen`, `Przedmiot`, `Nauczyciel` na ich kluczach głównych.

```
CREATE SEQUENCE SEQ_OcenaId AS INT START
WITH
    1 INCREMENT BY 1 MINVALUE 1 CACHE 10;

CREATE TABLE
    Ocena (
        OcenaId INT NOT NULL DEFAULT NEXT VALUE FOR SEQ_OcenaId,
        UczenId INT NOT NULL,
        PrzedmiotId INT NOT NULL,
        Ocena TINYINT NOT NULL,
        NauczycielId INT NOT NULL,
        DataWystawienia DATE DEFAULT GETDATE()
    );

ALTER TABLE Ocena ADD CONSTRAINT PK_Ocena PRIMARY KEY (OcenaId);

ALTER TABLE Ocena ADD CONSTRAINT FK_Ocena_UczenId FOREIGN KEY (UczenId)
REFERENCES Uczen (UczenId) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Ocena ADD CONSTRAINT FK_Ocena_PrzedmiotId FOREIGN KEY
(PrzedmiotId) REFERENCES Przedmiot (PrzedmiotId) ON DELETE CASCADE ON UPDATE
CASCADE;

ALTER TABLE Ocena ADD CONSTRAINT FK_Ocena_NauczycielId FOREIGN KEY
(NauczycielId) REFERENCES Nauczyciel (NauczycielId) ON DELETE NO ACTION ON
UPDATE NO ACTION;

ALTER TABLE Ocena ADD CONSTRAINT CHK_Ocena CHECK (Ocena IN (1, 2, 3, 4, 5,
6));
```

1.9. Tabela `Lekcja`

Ostatnią tabelą jest tabela `Lekcja` mająca najwięcej połączeń z innymi tabalami. Jej celem jest przechowywać informacje o wszystkich zajęciach wszystkich uczniów i nauczycieli w jednym tygodniu z określeniem godziny lekcyjnej (pole `NrLekcji`) i pomieszczenia (atrybut `NrSali`). Nie ma klucza głównego. Jej relacje definiują klucze obce `PrzedmiotId`, `NauczycielId`, `KlasaId` i `NrSali`. Sprawdzana jest poprawność dnia tygodnia w ograniczeniu CHECK `CHK_Lekcja`.

```
CREATE TABLE
    Lekcja (
        DzienTygodnia VARCHAR(12) NOT NULL,
        NrLekcji INT NOT NULL,
        PrzedmiotId INT NOT NULL,
        NauczycielId INT NOT NULL,
        KlasaId INT NOT NULL,
        NrSali INT NOT NULL
    );

ALTER TABLE Lekcja ADD CONSTRAINT FK_Lekcja_PrzedmiotId FOREIGN KEY
(PrzedmiotId) REFERENCES Przedmiot (PrzedmiotId) ON DELETE CASCADE ON UPDATE
CASCADE;

ALTER TABLE Lekcja ADD CONSTRAINT FK_Lekcja_NauczycielId FOREIGN KEY
(NauczycielId) REFERENCES Nauczyciel (NauczycielId) ON DELETE CASCADE ON
UPDATE CASCADE;

ALTER TABLE Lekcja ADD CONSTRAINT FK_Lekcja_KlasaId FOREIGN KEY (KlasaId)
REFERENCES Klasa (KlasaId) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE Lekcja ADD CONSTRAINT FK_Lekcja_NrSali FOREIGN KEY (NrSali)
REFERENCES Sala (NrSali) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE Lekcja ADD CONSTRAINT CHK_Lekcja CHECK (
    DzienTygodnia IN (
        'Poniedziałek',
        'Wtorek',
        'Środa',
        'Czwartek',
        'Piątek'
    )
);
```

1.10. Dodatkowa tabela `ZarchiwizowanyUczen`

Utworzona na cele jednego z wyzwalaczy, do archiwizacji uczniów. Jej przeznaczenie zostanie dokładniej omówione później.

```
CREATE TABLE ZarchiwizowanyUczen (
        UczenId INT PRIMARY KEY NOT NULL,
        Imie VARCHAR(30) NOT NULL,
        Nazwisko VARCHAR(50) NOT NULL,
        Miejscowosc VARCHAR(40) NOT NULL,
        Ulica VARCHAR(60),
        NumerDomu VARCHAR(10) NOT NULL,
        KodPocztowy VARCHAR(6) NOT NULL
);
```

2. Widoki

2.1. Widok `SredniaKlasy`

Widok zwraca zestawienie wszystkich klas i średnią ze wszystkich przedmiotów wszystkich ocen dla wszystkich uczniów danej klasy.

```
CREATE VIEW
    SredniaOcenKlasy AS
SELECT
    Klasa.NrKlasy,
    Klasa.Oznaczenie,
    AVG(Ocena.Ocena) AS SredniaOcenKlasy
FROM
    (
        Klasa
        INNER JOIN Uczen ON Klasa.KlasaId = uczen.UczenId
    )
    INNER JOIN Ocena ON Uczen.UczenId = Ocena.UczenId
GROUP BY
    Uczen.KlasaId,
    Klasa.NrKlasy,
    Klasa.Oznaczenie;
```

2.2. Widok `IlosciSalPrzedmiotowych`

Wywołanie tego widoku zwróci nam listę wszystkich przedmiotów oraz ilość sal, które są specjalnie przystosowane do nauki tego przedmiotu.

```
CREATE VIEW
    IlosciSalPrzedmiotowych AS
SELECT
    Sala.NazwaPrzedmiotu,
    COUNT(NrSali) AS IloscSal
FROM
    Sala
GROUP BY
    Sala.NazwaPrzedmiotu;
```

2.3. Widok `NauczycieleWychowawcy`

Ten widok pozwoli nam się dowiedzieć kto jest wychowawcą danej klasy w szkole.

```
CREATE VIEW
    NauczycieleWychowawcy AS
SELECT
    Klasa.NrKlasy,
    Klasa.Oznaczenie,
    Nauczyciel.Imie,
    Nauczyciel.Nazwisko
FROM
    Klasa
    INNER JOIN Nauczyciel ON Klasa.OpiekunId = Nauczyciel.NauczycielId;
```

2.4. Widok `SrednieUczniow`

Użycie twgo widoku pozwala zobaczyć śrendią arytmetyczną wszystkich ocen każdego ucznia. Zwraca imię, nazwisko, klasę, typ klasy i średnią ocen ucznia.

```
CREATE VIEW
    SrednieUczniow AS
SELECT
    Uczen.Imie,
    Uczen.Nazwisko,
    Klasa.NrKlasy,
    Klasa.Oznaczenie,
    Klasa.Typ,
    AVG(Ocena.Ocena) AS SredniaOcen
FROM
    (
        Uczen
        INNER JOIN Klasa ON Uczen.KlasaId = Klasa.KlasaId
    )
    INNER JOIN Ocena ON Uczen.UczenId = Ocena.OcenaId
GROUP BY
    Uczen.Imie,
    Uczen.Nazwisko,
    Klasa.NrKlasy,
    Klasa.Oznaczenie,
    Klasa.Typ;
```

2.5. Widok `IloscOcenNauczyciela`

Ten widok zwróci nam zestawienie wszystkich nauczycieli i ilość ocen jaką wystawił wszystkim uczniom. Ujżymy imię i nazwisko nauczyciela i ilość wystawionych ocen.

```
CREATE VIEW
    IloscOcenNauczyciela AS
SELECT
    Nauczyciel.Imie,
    Nauczyciel.Nazwisko,
    COUNT(Ocena.OcenaId) AS IloscWystawionychOcen
FROM
    Nauczyciel
    LEFT JOIN Ocena ON Nauczyciel.NauczycielId = Ocena.NauczycielId
GROUP BY
    Nauczyciel.NauczycielId,
    Nauczyciel.Imie,
    Nauczyciel.Nazwisko;
```

2.6. Widok `IloscGodzinTygodniowo`

Jeżeli chcielibyśmy sprawdzić ile godzin w tygodniu ma dana klasa to najprostszym sposobem będzie użyc widoku `IloscGodzinTygodniowo`. Zwraca ono zestawienie klas i ich ilości lekcji/tydzień.

```
CREATE VIEW
    IloscGodzinTygodniowoKlasa AS
SELECT
    Klasa.NrKlasy,
    Klasa.Oznaczenie,
    COUNT(Lekcja.KlasaId) AS IloscLekcjiTygodniowo
FROM
    Klasa
    INNER JOIN Lekcja ON Klasa.KlasaId = Lekcja.KlasaId
GROUP BY
    Klasa.NrKlasy,
    Klasa.Oznaczenie;
```

3. Funkcje

3.1. Funkcja `SrednieUczniowKlasaPrzedmiot`

Funkcja zwraca tabelę ukazującą średnią arytmetyczną ocen danej klasy z konkretnego przedmiotu. Przyjmuje 3 parametry: `@NrKlasy` – numer klasy (rok), `@OznaczenieKlasy` – oznaczenie literowe klasy i `@NazwaPrzedmiotu` – przedmiot, którego średnią ocen chcemy zobaczyć.

```
CREATE FUNCTION SrednieUczniowKlasaPrzedmiot (
    @NrKlasy INT,
    @OznaczenieKlasy VARCHAR(5),
    @NazwaPrzedmiotu VARCHAR(15)
)
RETURNS TABLE AS
RETURN
    SELECT
        Uczen.Imie,
        Uczen.Nazwisko,
        AVG(Ocena.Ocena) AS SredniaOcen
        FROM ((Uczen INNER JOIN Ocena ON Uczen.UczenId = Ocena.UczenId) INNER
JOIN Przedmiot ON Ocena.PrzedmiotId = Przedmiot.PrzedmiotId) INNER JOIN Klasa
ON Uczen.KlasaId = Klasa.KlasaId
        WHERE Przedmiot.NazwaPrzedmiotu = @NazwaPrzedmiotu AND Klasa.NrKlasy =
@NrKlasy AND Klasa.Oznaczenie = @OznaczenieKlasy
        GROUP BY Uczen.Imie, Uczen.Nazwisko;
```

3.2. Funkcja `PlanNauczycielaDzienTygodnia`

Ta funkcja zwraca tabelę, która ukazuje dla danego nauczyciela jego plan lekcji danego dnia. Konieczne jest podanie 3 argumentów: `@ImieNauczyciela`, `@NazwiskoNauczyciela` i `@DzienTygodnia`, nazwy są wymowne.

```
CREATE FUNCTION PlanNauczycielaDzienTygodnia (
    @ImieNauczyciela VARCHAR(30),
    @NazwiskoNauczyciela VARCHAR(50),
    @DzienTygodnia VARCHAR(12)
)
RETURNS TABLE AS
RETURN
    SELECT
        Lekcja.DzienTygodnia,
        Lekcja.NrLekcji,
        Przedmiot.NazwaPrzedmiotu
        FROM (Lekcja INNER JOIN Przedmiot ON Lekcja.PrzedmiotId =
Przedmiot.PrzedmiotId) INNER JOIN Nauczyciel ON Lekcja.NauczycielId =
Nauczyciel.NauczycielId
        WHERE Nauczyciel.Imie = @ImieNauczyciela AND Nauczyciel.Nazwisko =
@NazwiskoNauczyciela AND Lekcja.DzienTygodnia = @DzienTygodnia;
```

4. Procedury

4.1. Procedura `DodajLekcję`

Celem procedury jest dodanie lekcji dla danej klasy w danym dniu tygodnia na danej godzinie lekcyjnej danego przedmiotu z danym nauczycielem i w danej sali, i takie są wymagane parametry tej procedury. Początkowo procedura sprawdza czy ten nauczyciel nie jest już zajęty tego dnia o tej godzinie i jeżeli jest wolny to dopisuje lekcję do tabeli `Lekcje`

```
CREATE PROCEDURE DodajLekcję (
    @NrKlasy INT,
    @OznaczenieKlasy VARCHAR(5),
    @DzienTygodnia VARCHAR(12),
    @NrLekcji INT,
    @NazwaPrzedmiotu VARCHAR(30),
    @ImieNauczyciela VARCHAR(30),
    @NazwiskoNauczyciela VARCHAR(50),
    @NrSali INT
)
AS
BEGIN
IF (SELECT Lekcja.NauczycielId FROM Lekcja INNER JOIN Nauczyciel ON
Lekcja.NauczycielId = Nauczyciel.NauczycielId WHERE Lekcja.DzienTygodnia =
@DzienTygodnia AND Lekcja.NrLekcji = @NrLekcji AND Nauczyciel.Imie =
@ImieNauczyciela AND Nauczyciel.Nazwisko = @NazwiskoNauczyciela) IS NULL
    BEGIN
        DECLARE @PrzedmiotId INT, @NauczycielId INT, @KlasaId INT;
        SELECT @PrzedmiotId = Przedmiot.PrzedmiotId FROM Przedmiot WHERE
Przedmiot.NazwaPrzedmiotu = @NazwaPrzedmiotu;
        SELECT @NauczycielId = Nauczyciel.NauczycielId FROM Nauczyciel WHERE
Nauczyciel.Imie = @ImieNauczyciela AND Nauczyciel.Nazwisko =
@NazwiskoNauczyciela;
        SELECT @KlasaId = Klasa.KlasaId FROM KLasa WHERE Klasa.NrKlasy =
@NrKlasy AND Klasa.Oznaczenie = @OznaczenieKlasy;

        INSERT INTO Lekcja VALUES (@DzienTygodnia, @NrLekcji, @PrzedmiotId,
@NauczycielId, @KlasaId, @NrSali);
    END;
ELSE
    SELECT 'Nauczyciel już jest zajęty tego dnia na tej lekcji';
END;
```

4.2. Procedura `ArchiwizujCzyscTabele`

Ta procedura jest specyficzna, jako parametr podajemy jej nazwę tabeli, którą chcemy zarchiwizować. Procedura szuka po nazwie parametru konkretnego przypadku i dla znalezionej tabeli wykonuje następujące polecenia: kopiuje jej zawartość do odpowiedniej tabeli archiwizującej, modyfikuje ograniczenia tabeli, aby wartości kluczy obcych nie były sprawdzane; usuwa zawartość tabeli i spowrotem modyfikuje ograniczenia, aby wartości kluczy były ponownie sprawdzane.

```
CREATE PROCEDURE ArchiwizujCzyscTabele (
    @NazwaTabeli VARCHAR(15)
)
AS
BEGIN
    IF @NazwaTabeli LIKE 'Uczen'
        BEGIN
            SELECT * INTO ArchiwumUczen FROM Uczen;
            ALTER TABLE Uczen NOCHECK CONSTRAINT ALL;
            DELETE Uczen;
            ALTER TABLE Uczen CHECK CONSTRAINT ALL;
        END;
    ELSE IF @NazwaTabeli LIKE 'Nauczyciel'
        BEGIN
            SELECT * INTO ArchiwumNauczyciel FROM Nauczyciel;
            ALTER TABLE Nauczyciel NOCHECK CONSTRAINT ALL;
            DELETE Nauczyciel;
            ALTER TABLE Nauczyciel CHECK CONSTRAINT ALL;
        END;
    ELSE IF @NazwaTabeli LIKE 'Klasa'
        BEGIN
            SELECT * INTO ArchiwumKlasa FROM Klasa;
            ALTER TABLE Klasa NOCHECK CONSTRAINT ALL;
            DELETE Klasa;
            ALTER TABLE Klasa CHECK CONSTRAINT ALL;
        END;
      . . . -- i tak dalej dla każdej tabeli
    ELSE
        SELECT 'Nie ma takiej tabeli';
END;
```

Nie zostało użyte polecenie `TRUNCATE TABLE [Tabela]`, ponieważ wywoływało ono błędy klucza obcego nawet po zdjęciu ich sprawdzania.

5. Wyzwalacze

5.1. Wyzwalacz `ArchiwizujUcznia`

Ten wyzwalacz jest odpowiedzialny za przeniesienie ucznia do tabeli `ZarchiwizowanyUczen`, czyli tej dodatkowej tabeli wspomnianej w punkcie 1.10. Kopiuje wszystkie wartości z tabeli `Uczen` z wyjątkiem `PESEL` i `KlasaId`.

```
CREATE TRIGGER TR_ArchiwizujUcznia ON Uczen
FOR DELETE
AS
BEGIN
    INSERT INTO ZarchiwizowanyUczen SELECT UczenId, Imie, Nazwisko,
Miejscowosc, Ulica, NumerDomu, KodPocztowy FROM deleted;
END;
```

5.2. Wyzwalacz `SprawdzKlase`

Celem wyzwalacza jest sprawdzenie, czy osoba dodająca klasę do szkoły nie dodaje przypadkiem drugiej takiej samej klasy. Wyzwalacz sprawdza czy w zbiorze klas nie istnieje już jedna o takiej samej nazwie np. „II B”, w razie potrzeby usuwa nadprogramową klasę i wyświetla odpowiedni komunikat. Jeżeli jednak podana klasa jeszcze nie istnieje to zostaje normalnie dopisana.

```
CREATE TRIGGER TR_SprawdzKlase ON Klasa
INSTEAD OF INSERT, UPDATE
AS
BEGIN
    IF EXISTS (SELECT Klasa.KlasaId FROM Klasa, inserted WHERE Klasa.NrKlasy =
inserted.NrKlasy AND Klasa.Oznaczenie = inserted.Oznaczenie)
    BEGIN
        SELECT 'Istnieje już taka klasa';
        DELETE Klasa FROM Klasa INNER JOIN inserted ON Klasa.KlasaId =
inserted.KlasaId;
    END;
    ELSE
        INSERT INTO Klasa SELECT * FROM inserted;
END;
```

Diagram bazy danych został umieszczony w osobnym pliku ze względu na jego złożoność.
Jest to plik o nazwie: Diagram_ER_szkola.png, LINK