Baza danych przedsiębiorstwa komunikacji miejskiej

Założenia projektu

- Baza danych przechowuje informacje o osobach związanych z przedsiębiorstwem (pracownikach, ukaranych mandatami). Przechowuje tylko niezbędne informacje na temat tych osób.
- Baza danych nie zajmuje się ewidencją bądź wyliczaniem wypłat dla pracowników (można to zaimplementować).
- Baza danych przechowuje częściowe informacje na temat infrastruktury miejskiej związanej z komunikacją miejską, czyli o przystankach, strefach biletowych i ich taryfach, liniach oraz kursach. Umożliwia generowanie rozkładów jazdy dla linii oraz przystanków.
- Baza danych gromadzi informacje na temat posiadanych przez siebie pojazdów oraz kosztów eksploatacji z nimi związanymi. Nie prowadzi jednak ewidencji faktur (można to zaimplementować).
- Baza danych powinna być obsługiwana poprzez użytkowników o różnym stopniu uprawnień. Przykładowo, kontrolerzy biletów mogą jedynie wystawiać mandaty, pracownicy kadrowi *nie mogą* jednak w nie ingerować, ale mogą zmieniać pensje pracowników, klienci (pasażerowie) mogą generować rozkłady jazdy, ale nic poza tym.

Tabele (20)

- Osoby
  Informacje nt. wszelkich osób fizycznych związanych z przedsiębiorstwem.
- Stanowiska
  Informacje nt. różnych stanowisk w przedsiębiorstwie.
- Pracownicy
  Informacje nt. aktualnie zatrudnionych pracowników.
- Kierowcy
  Informacje nt. aktualnie zatrudnionych kierowców.
- Kontrolerzy
  Informacje nt. aktualnie zatrudnionych kontrolerów.
- Strefy
  Iinformacje nt. stref biletowych.
- Bilety
  Informacje nt. taryf biletowych.
- Progresja kar
  Informacje nt. progresywnych stawek mandatów w zależności od liczby ukarań.
- Przewinienia
  Informacje nt. przewinień karalnych mandatem.
- Nałożone kary
  Informacje nt. nałożonych mandatów.
- Autobusy
  Informacje nt. posiadanych przez przedsiębiorstwo autobusów.
- Czasy przejazdu
  Informacje nt. czasów przejazdu na danej trasie.

- Koszty eksploatacji
  Informacje nt. kosztów eksploatacji autobusu.
- Kursy
  Informacje nt. kursów autobusów.
- Linie
  Informacje nt. linii autobusowych.
- Modele autobusów
  Informacje nt. modeli autobusów posiadanych przez przedsiębiorstwo.
- Przystanki
  Informacje nt. przystanków.
- Trasy
  Informacje nt. tras autobusów.
- Usługi
  Informacje nt. usług związanych z eksploatacją autobusów.
- Wykonane kursy
  Informacje nt. wykonanych kursów.

Funkcje (7)

- Kwota kary
  Zwraca obliczoną kwotę mandatu, przyjmując jako parametry dane wystawiającego mandat kontrolera, osoby karanej oraz rodzaju przewinienia.
- Pracownicy na stanowisku
  Zwraca tabelę pracowników na stanowisku zadanym parametrem.
- Addtime
  Zwraca wynik dodawania dwóch wartości formatu TIME do siebie (na potrzeby obliczania czasów dojazdu do poszczególnych przystanków).
- Harmonogram jazdy autobusu
  Zwraca tabelę reprezentującą harmonogram jazdy autobusu dla danego dnia tygodnia, tzn. w jakich godzinach wykonywane są nim poszczególne kursy i na jakich przystankach się one zaczynają i kończą
- Harmonogram pracy kierowcy
  Zwraca tabelę reprezentującą harmonogram pracy kierowcy dla danego dnia tygodnia, tzn. w jakich godzinach wykonuje poszczególne kursy i na jakich przystankach się one zaczynają i kończą,
- Rozkład jazdy dla kursu
  Zwraca tabelę reprezentująca czasy dojazdu do poszczególnych przystanków na trasie danego kursu.
- Rozkład jazdy dla przystanku
  Zwraca tabelę reprezentującą rozkład jazdy dla danego przystanku i dnia tygodnia, wypisuje godziny w jakich zatrzymują się na nim autobusy wszystkich linii przez niego przejeżdżających.

Widoki (4)

- Przedawnione kary
  Pokazuje wszystkie nieopłacone mandaty, które są starsze niż rok.
- Kierowcy badania
  Pokazuje wszystkich kierowców, których badania lekarskie (pracownicze lub związane z prawem jazdy) tracą ważność w przeciągu trzech miesięcy.

- Ranking kontrolerów
  Pokazuje dla każdego kontrolera, który wystawił mandat w przeciągu ostatnich 30 dni sumę kwot wystawionych przez niego mandatów.
- Czasy trwania kursów
  Pokazuje dla każdego kursu godzinę rozpoczęcia i zakończenia wraz z końcowymi przystankami.

Procedury (6)

- Opłacenie mandatu
  Służy opłacaniu danego mandatu, opcjonalnie zwraca nadmiarową, zapłaconą kwotę.
- Wystaw mandat
  Służy wystawaniu mandatu danej osobie za dane przewinienie. Posiada specjalną flagę, pozwalającą poprawić nieaktualne lub niepoprawnie wstawione dane danej osoby.
- Aktualizacja mandatów
  Służy usunięciu z bazy danych wszystkich opłaconych mandatów starszych niż rok.
- Zmiana kwot kar
  Służy do proporcjonalnej zmiany kwot wszystkich kar o wartość zadaną parametrem.
- Zastąp autobus
  Służy zastąpieniu jednego autobusu innym we wszystkich kursach w tabeli Kursy.
- Zastąp kierowcę
  Służy przypisaniu wszystkich kursy jednego kierowcy innemu.

Wyzwalacze (5)

- Osoby INSERT
  Nie pozwala na dodanie rekordów do tabeli Osoby, w których PESEL lub numer dowodu osobistego są niepoprawne.
- Osoby UPDATE
  Nie pozwala na edycję rekordów w tabeli Osoby w taki sposób, że PESEL lub numer dowodu osobistego będą niepoprawne.
- NałożoneKary ALL
  Nie pozwala na jakąkolwiek manualną ingerencję w tabeli Nałożone kary.
- Kursy INSERT
  Sprawdza czy wstawione kursy mogą być wykonywane przez danego kierowcę danym autobusem, kontroluje by każdy kurs był przypisany do nie więcej niż jednego z dni.
- Kursy UPDATE
  Uniemożliwia zmianę godziny rozpoczęcia kursu ze względu na konieczność zachowania integralności tabeli wykonane kursy, sprawdza czy wstawione kursy mogą być wykonywane przez danego kierowcę danym autobusem, kontroluje by każdy kurs był przypisany do nie więcej niż jednego z dni,

Dodatkowe więzy integralności

- W tabeli *Kursy* przy wstawianiu lub aktualizacji danych przez wyzwalacze realizowane jest sprawdzanie czy dany kurs może być wykonany przez danego kierowcę danym autobusem tzn. czy w tym samym czasie kierowca lub autobus nie wykonują jakiegoś innego kursu.

- Jedyna możliwość ingerencji w tabelę *Nałożone kary* jest realizowana poprzez procedury *Wystaw mandat*, *Opłacenie mandatu* oraz *Aktualizacja mandatów* w związku z krytycznym charakterem danych zawartych w tej tabeli.
- Tabele *Kierowcy* oraz *Kontrolerzy* dziedziczą z tabeli *Pracownicy*, a ta dziedziczy z tabeli *Osoby*. W każdej z tych tabel dodawany jest dla danej osoby dodatkowy numer identyfikacyjny. Służy to temu, aby móc przypisać pracownikom numery porządkowe dodatkowo związane z ich stanowiskiem. Ponadto, przykładowo zmiana stanowiska kierowcy nie wymusza usunięcia jego rekordów z tabel *Osoby* lub *Pracownicy*.

Strategia pielęgnacji bazy danych

Baza danych zawiera informacje, które w większości nie są zmieniane szybko w czasie (przystanki, pracownicy, taryfy biletów), jednakże zawiera także dane o wykonanych kursach czy nałożonych (i opłaconych) mandatach - te ulegają zmieną wręcz nieustannie. W związku z tym, istotne jest wykonywanie regularnych kopii zapasowych. Cotygodniowe kopie zapasowe, wykonywane w niedziele, kiedy przedsiębiorstwo zdecydowanie korzysta z bazy danych najmniej oraz kiedy nie są rejestrowane przelewy bankowe powinny być w takiej sytuacji wystarczające.

Typowe zapytania

Są realizowane przez procedury, funkcje oraz widoki.

Kod źródłowy

Funkcje

Funkcja ADDTIME

```
CREATE FUNCTION ADDTIME (@StartTime TIME,@Offset TIME)
RETURNS TIME
AS
BEGIN
    SET @StartTime = DATEADD (hour, DATEPART(hh,@Offset),@StartTime)
    SET @StartTime = DATEADD (n, DATEPART(n,@Offset),@StartTime)

    RETURN @StartTime
END
GO
```

Funkcja Harmonogram jazdy autobusu

```
CREATE FUNCTION HarmonogramJazdyAutobusu(@AutobusID INT,@DzienTyg INT)
RETURNS @KursyAutobusu TABLE
```

```
(
    GodzinaOdjazdu TIME,
    PrzystanekOdjazdu NVARCHAR,
    GodzinaPrzyjazdu TIME,
    PrzystanekPrzyjazdu NVARCHAR
)
AS
    BEGIN
        INSERT INTO @KursyAutobusu
            SELECT GodzinaOdjazdu,PrzystanekOdjazdu,
            GodzinaPrzyjazdu,PrzystanekPrzyjazdu
            FROM CzasyTrwaniaKursów
            WHERE AutobusID=@AutobusID AND
            ((Soboty=1 AND @DzienTyg=6) OR
            (Niedziele=1 AND @DzienTyg=7) OR
            (DniPowszednie=1 AND @DzienTyg NOT IN(6,7)))
        RETURN
    END
GO
```

Funkcja harmonogram pracy kierowcy

```
CREATE FUNCTION HarmonogramPracyKierowcy(@PracownikID INT,@DzienTyg INT)
RETURNS @KursyKierowcy TABLE
(
    GodzinaOdjazdu TIME,
    PrzystanekOdjazdu NVARCHAR,
    GodzinaPrzyjazdu TIME,
    PrzystanekPrzyjazdu NVARCHAR
)
AS
    BEGIN
        INSERT INTO @KursyKierowcy
            SELECT GodzinaOdjazdu,PrzystanekOdjazdu,
            GodzinaPrzyjazdu,PrzystanekPrzyjazdu
            FROM CzasyTrwaniaKursów
            WHERE PracownikID=@PracownikID AND
            ((Soboty=1 AND @DzienTyg=6) OR
            (Niedziele=1 AND @DzienTyg=7) OR
            (DniPowszednie=1 AND @DzienTyg NOT IN(6,7)))
        RETURN
    END
GO
```

Funkcja Kwota kary

```
CREATE FUNCTION KwotaKary(
  @ukaranyID INT,
  @przewinienieID INT
```

```
) RETURNS MONEY
AS
BEGIN
  DECLARE @dzisiaj DATE = GETDATE();
  DECLARE @modyfikator DECIMAL = (
    SELECT TOP 1 PK.Modyfikator
    FROM ProgresjaKar PK
    WHERE PK.LiczbaPrzewinień >= (
      SELECT COUNT(NK.UkaranyID) LiczbaPrzewinień
      FROM NałożoneKary AS NK
      WHERE NK.UkaranyID = @ukaranyID AND DATEDIFF(DAY, @dzisiaj, NK.DataUkarania)
< 365
    )
    ORDER BY PK.LiczbaPrzewinień ASC
  )
  DECLARE @kwotaBazowa MONEY = (
    SELECT P.AktualnaKwotaKary
    FROM Przewinienia P
    WHERE @przewinienieID = P.PrzewinienieID
  )
  DECLARE @wynik MONEY = @kwotaBazowa * @modyfikator;
  RETURN @wynik;
END;
GO
```

Funkcja Pracownicy na stanowisku

```
CREATE FUNCTION PracownicyNaStanowisku(@stanowisko NVARCHAR(256))
RETURNS TABLE
AS
  RETURN(
    SELECT P.PracownikID, O.Imię, O.Nazwisko
    FROM Pracownicy P
    JOIN Osoby O
    ON P.OsobaID = P.OsobaID
    WHERE P.Stanowisko = @stanowisko)
GO
```

Funkcja Rozkład jazdy dla kursu

```
CREATE FUNCTION RozkładJazdyDlaKursu ( @KursID INT)
RETURNS @RozkladTab TABLE(PrzystanekID NVARCHAR, Godzina Time)
AS
BEGIN
    INSERT INTO @RozkladTab
        SELECT P.PrzystanekID, dbo.ADDTIME(K.GodzinaOdjazdu,C.CzasPrzejZPoczątku)
        FROM Przystanki P JOIN CzasyPrzejazdu C
        ON P.PrzystanekID=C.PrzystanekID
        JOIN Trasy T ON C.TrasaID=T.TrasaID
```

```
        JOIN Kursy K ON T.TrasaID=K.TrasaID
        WHERE K.KursID=@KursID

    INSERT INTO @RozkladTab
        SELECT P.PrzystanekID, K.GodzinaOdjazdu Godzina
        FROM Przystanki P JOIN Trasy T
        ON P.PrzystanekID=T.PrzystanekPoczątkowy
        JOIN Kursy K ON K.TrasaID=T.TrasaID
        WHERE K.KursID=@KursID
    RETURN
END
GO
```

Funkcja Rozkład jazdy dla przystanku

```
CREATE FUNCTION RozkładJazdyDlaPrzystanku ( @PrzystanekID INT,@DzienTyg INT)
RETURNS @RozkladTab TABLE(NazwaLinii NVARCHAR, KursID INT, Godzina Time)
AS
BEGIN
    INSERT INTO @RozkladTab
        SELECT T.LiniaID, K.KursID,
dbo.ADDTIME(K.GodzinaOdjazdu,C.CzasPrzejZPoczątku) Godzina
        FROM Przystanki P JOIN CzasyPrzejazdu C
        ON P.PrzystanekID=C.PrzystanekID
        JOIN Trasy T ON C.TrasaID=T.TrasaID
        JOIN Kursy K ON T.TrasaID=K.TrasaID
        WHERE P.PrzystanekID=@PrzystanekID AND
        ((K.Soboty=1 AND @DzienTyg=6) OR
        (K.Niedziele=1 AND @DzienTyg=7) OR
        (K.DniPowszednie=1 AND @DzienTyg NOT IN(6,7)))

    INSERT INTO @RozkladTab
        SELECT T.LiniaID, K.KursID, K.GodzinaOdjazdu Godzina
        FROM Przystanki P JOIN Trasy T
        ON P.PrzystanekID=T.PrzystanekPoczątkowy
        JOIN Kursy K ON K.TrasaID=T.TrasaID
        WHERE P.PrzystanekID=@PrzystanekID AND
        ((K.Soboty=1 AND @DzienTyg=6) OR
        (K.Niedziele=1 AND @DzienTyg=7) OR
        (K.DniPowszednie=1 AND @DzienTyg NOT IN(6,7)))
    RETURN
END
GO
```

Procedury

Procedura Aktualizacja mandatów

```
CREATE PROCEDURE AktualizacjaMandatów
AS
  ALTER TABLE NałożoneKary DISABLE TRIGGER NałożoneKaryALL
  DELETE FROM NałożoneKary
  WHERE DataOpłacenia IS NOT NULL AND DATEDIFF(DAY, DataUkarania, GETDATE()) > 365
  ALTER TABLE NałożoneKary ENABLE TRIGGER NałożoneKaryALL
GO
```

Procedura Opłacenie mandatu

```
CREATE PROCEDURE OpłacenieMandatu(
  @mandatID INT,
  @kwota MONEY,
  @reszta MONEY OUTPUT)
AS
  IF @mandatID NOT IN(
    SELECT KaraID FROM NałożoneKary)
    RAISERROR('Niepoprawny numer mandatu!', 0, 2)
    RETURN
  SET @reszta = (
  SELECT KwotaKary
  FROM NałożoneKary
  WHERE @mandatID = KaraID) - @kwota
  IF @reszta < 0
    RAISERROR('Za mała kwota do opłacenia mandatu!', 0, 1)
    RETURN
  ALTER TABLE NałożoneKary DISABLE TRIGGER NałożoneKaryALL
  UPDATE NałożoneKary
    SET DataOpłacenia = GETDATE()
    WHERE @mandatID = KaraID
  ALTER TABLE NałożoneKary ENABLE TRIGGER NałożoneKaryALL
GO
```

Procedura Wystaw mandat

```
CREATE PROCEDURE WystawMandat(
  @kontrolerID INT,
  @ukaranyImię NVARCHAR(256),
  @ukaranyNazwisko NVARCHAR(256),
  @ukaranyPESEL INT,
  @ukaranyNrDowoduOsobistego INT,
  @przewinienieID INT,
  @wymuśNadpisanieDanych BIT,
  @kwota MONEY OUT)
AS
  IF (@ukaranyPESEL IS NULL AND @ukaranyNrDowoduOsobistego IS NULL) OR
@kontrolerID IS NULL OR @ukaranyImię IS NULL OR @ukaranyNazwisko IS NULL OR
@przewinienieID IS NULL
```

```
    RAISERROR('Błędnie wprowadzone dane, nie wystawiono mandatu!', 0, 1)
    RETURN

  IF @kontrolerID NOT IN(
    SELECT K.KontrolerID
    FROM Kontrolerzy K)
    RAISERROR('Nieznany numer kontrolera!', 0, 6)
    RETURN

  IF @przewinienieID NOT IN(
    SELECT P.PrzewinienieID
    FROM Przewinienia P)
    RAISERROR('Nieznane ID przewinienia!', 0, 7)
    RETURN

  IF @ukaranyPESEL IS NOT NULL AND @ukaranyPESEL NOT IN(SELECT O.PESEL FROM Osoby
O) AND
  @ukaranyNrDowoduOsobistego IS NOT NULL AND @ukaranyNrDowoduOsobistego NOT IN
(SELECT O.NrDowoduOsobistego FROM Osoby O)
    INSERT INTO Osoby (Imię, Nazwisko, PESEL, NrDowoduOsobistego)
    VALUES (@ukaranyImię, @ukaranyNazwisko, @ukaranyPESEL,
@ukaranyNrDowoduOsobistego)

  DECLARE @daneOsobaID INT, @daneImię INT, @daneNazwisko NVARCHAR(256), @danePESEL
NVARCHAR(11), @daneNrDowoduOsobistego NVARCHAR(9)

  SELECT
    @daneOsobaID = OsobaID,
    @daneImię = Imię,
    @daneNazwisko = Nazwisko,
    @danePESEL = PESEL,
    @daneNrDowoduOsobistego = NrDowoduOsobistego
  FROM Osoby
  WHERE @ukaranyPESEL IS NOT NULL AND @ukaranyPESEL = PESEL OR
  @ukaranyNrDowoduOsobistego IS NOT NULL AND @ukaranyNrDowoduOsobistego =
NrDowoduOsobistego

  IF @wymuśNadpisanieDanych = 1
    BEGIN TRY
      UPDATE Osoby
        SET
          Imię = @ukaranyImię,
          Nazwisko = @ukaranyNazwisko,
          PESEL = COALESCE(@ukaranyPESEL, PESEL),
          NrDowoduOsobistego = COALESCE(@ukaranyNrDowoduOsobistego,
NrDowoduOsobistego)
      WHERE @ukaranyPESEL IS NOT NULL AND @ukaranyPESEL = PESEL OR
        @ukaranyNrDowoduOsobistego IS NOT NULL AND @ukaranyNrDowoduOsobistego =
NrDowoduOsobistego
    END TRY
    BEGIN CATCH
      RAISERROR('Niepoprawny PESEL lub numer dowodu osobistego!', 0, 8)
      RETURN
    END CATCH
```

```
  ELSE
    DECLARE @errorFlag BIT = 0
    IF @daneImię != @ukaranyImię
      RAISERROR('Imię niezgodne z bazą danych!', 0, 2)
      SET @errorFlag = 1
    IF @daneNazwisko != @ukaranyNazwisko
      RAISERROR('Nazwisko niezgodne z bazą danych!', 0, 3)
      SET @errorFlag = 1
    IF @danePESEL IS NOT NULL AND @danePESEL IS NOT NULL AND @danePESEL !=
@ukaranyPESEL
      RAISERROR('PESEL niezgodny z bazą danych!', 0, 4)
      SET @errorFlag = 1
    IF @daneNrDowoduOsobistego IS NOT NULL AND @daneNrDowoduOsobistego IS NOT NULL
AND @daneNrDowoduOsobistego != @ukaranyNrDowoduOsobistego
      RAISERROR('Numer dowodu osobistego niezgodny z bazą danych!', 0, 5)
      SET @errorFlag = 1
    IF @errorFlag = 1
      RETURN

  SET @kwota = (SELECT * FROM KwotaKary(@daneOsobaID, @przewinienieID))

  ALTER TABLE NałożoneKary DISABLE TRIGGER NałożoneKaryALL
  INSERT INTO NałożoneKary(KontrolerID, PrzewinienieID, UkaranyID, DataUkarania,
KwotaKary, DataOpłacenia)
  VALUES(@kontrolerID, @przewinienieID, @daneOsobaID, GETDATE(), @kwota, NULL)
  ALTER TABLE NałożoneKary ENABLE TRIGGER NałożoneKaryALL

GO
```

Procedura Zastąp autobus

```
CREATE PROCEDURE ZastąpAutobus(@zastępowany INT,@zastępujący INT)
AS
    UPDATE Kursy
    SET AutobusID=@zastępujący
    WHERE AutobusID=@zastępowany
GO
```

Procedura Zastąp kierowcę

```
CREATE PROCEDURE ZastąpKierowcę(@zastępowany INT,@zastępujący INT)
AS
    UPDATE Kursy
    SET PracownikID=@zastępujący
    WHERE PracownikID=@zastępowany
GO
```

Procedura Zmień kary

```
CREATE PROCEDURE ZmieńKary(
  @proporcja DECIMAL)
AS
  IF @proporcja <= 0
    RAISERROR('Niepoprawna proporcja!', 0, 1)
    RETURN
  UPDATE Przewinienia
  SET
    AktualnaKwotaKary = AktualnaKwotaKary * @proporcja
GO
```

Tabele

Tabela Autobusy

```
CREATE TABLE Autobusy(
    AutobusID INT,
    NumerRejestracyjny NVARCHAR NOT NULL UNIQUE,
    ModelID INT NOT NULL,
    RokProdukcji INT NOT NULL,
    DataRozpEksploatacji DATE,
    DataWaznosciPrzegladu DATE NOT NULL,
    AutomatyBiletowe TINYINT

    PRIMARY KEY(AutobusID),
    FOREIGN KEY(ModelID) REFERENCES ModeleAutobusów(ModelID)
);
GO
```

Tabela Bilety

```
CREATE TABLE Bilety(
  StrefaID INT NOT NULL,
  CenaNor20min MONEY NOT NULL,
  CenaUlg20min MONEY NOT NULL,
  CenaNor60min MONEY NOT NULL,
  CenaUlg60min MONEY NOT NULL,
  CenaNor24h MONEY NOT NULL,
  CenaUlg24h MONEY NOT NULL,
  CenaNorMsc MONEY NOT NULL,
  CenaUlgMsc MONEY NOT NULL

  PRIMARY KEY(StrefaID),
  FOREIGN KEY(StrefaID) REFERENCES Strefy(StrefaID)
);
GO
```

Tabela Czasy przejazdu

```
CREATE TABLE CzasyPrzejazdu(
    TrasaID INT,
    KolejnośćPrzystanku INT NOT NULL,
    PrzystanekID NVARCHAR(4) NOT NULL,
    CzasPrzejZPoczątku TIME NOT NULL

    PRIMARY KEY(TrasaID,KolejnośćPrzystanku),
    FOREIGN KEY(PrzystanekID) REFERENCES Przystanki(PrzystanekID),
    FOREIGN KEY(TrasaID) REFERENCES Trasy(TrasaID)
);
GO
```

Tabela Kierowcy

```
CREATE TABLE Kierowcy(
  PracownikID INT NOT NULL,
  NrTelefonuSłużbowego NVARCHAR(9),
  DataUzyskaniaPrawaJazdy DATE NOT NULL,
  DataWażnościPrawaJazdy DATE NOT NULL,
  DataWażnościBadańLekarskich DATE NOT NULL,

  PRIMARY KEY (PracownikID),
  FOREIGN KEY (PracownikID) REFERENCES Pracownicy(PracownikID)
);
GO
```

Tabela Kontrolerzy

```
  CREATE TABLE Kontrolerzy(
  PracownikID INT NOT NULL,
  KontrolerID INT UNIQUE IDENTITY(1, 1),

  PRIMARY KEY(PracownikID, KontrolerID),
  FOREIGN KEY (PracownikID) REFERENCES Pracownicy(PracownikID)
);
GO
```

Tabela Koszty eksploatacji

```
CREATE TABLE KosztyEksploatacji(
    AutobusID INT,
    DataWykonania DATE,
    NazwaUsługi NVARCHAR NOT NULL,
```

```
    Kwota MONEY NOT NULL

    PRIMARY KEY(AutobusID,DataWykonania),
    FOREIGN KEY(AutobusID) REFERENCES Autobusy(AutobusID),
    FOREIGN KEY(NazwaUsługi) REFERENCES Usługi(NazwaUsługi)
);
GO
```

Tabela Kursy

```
CREATE TABLE Kursy(
    KursID INT,
    TrasaID INT NOT NULL,
    GodzinaOdjazdu TIME NOT NULL,
    DniPowszednie BIT NOT NULL,
    Soboty BIT NOT NULL,
    Niedziele BIT NOT NULL,
    AutobusID INT,
    PracownikID INT

    PRIMARY KEY(KursID),
    FOREIGN KEY(TrasaID) REFERENCES Trasy(TrasaID),
    FOREIGN KEY(AutobusID) REFERENCES Autobusy(AutobusID),
    FOREIGN KEY (PracownikID) REFERENCES Kierowcy(PracownikID)
);
GO
```

Tabela Linie

```
CREATE TABLE Linie(
    LiniaID INT,
    NazwaLinii NVARCHAR NOT NULL,
    Nocna BIT,
    Przyspieszona BIT

    PRIMARY KEY(LiniaID)
);
GO
```

Tabela Modele autobusów

```
CREATE TABLE ModeleAutobusów(
    ModelID INT,
    Producent NVARCHAR NOT NULL,
    NazwaModelu NVARCHAR NOT NULL,
    MiejscaSiedzące INT NOT NULL,
```

```
    MiejscaStojące INT NOT NULL,
    MiejscaNaRowery INT,
    MiejscaNaWózki INT,
    Niskopodłogowy BIT,
    Przegubowy BIT,
    Napęd NVARCHAR


    PRIMARY KEY(ModelID),
);
GO
```

Tabela Nałożone kary

```
CREATE TABLE NałożoneKary(
  KaraID INT IDENTITY(1, 1),
  KontrolerID INT NOT NULL,
  PrzewinienieID INT NOT NULL,
  UkaranyID INT NOT NULL,
  DataUkarania DATE NOT NULL,
  KwotaKary MONEY NOT NULL,
  DataOpłacenia DATE,

  PRIMARY KEY (KaraID),
  FOREIGN KEY (KontrolerID) REFERENCES Kontrolerzy(KontrolerID),
  FOREIGN KEY (PrzewinienieID) REFERENCES Przewinienia(PrzewinienieID)
);
GO
```

Tabela Osoby

```
CREATE TABLE Osoby(
  OsobaID INT UNIQUE IDENTITY(1, 1),
  Imię NVARCHAR(256) NOT NULL,
  Nazwisko NVARCHAR(256) NOT NULL,
  PESEL NVARCHAR(11) UNIQUE,
  NrDowoduOsobistego NVARCHAR(9) UNIQUE,
  PRIMARY KEY (OsobaID)
);
GO
```

Tabela Pracownicy

```
CREATE TABLE Pracownicy(
  OsobaID INT NOT NULL,
  PracownikID INT UNIQUE IDENTITY(1, 1),
```

```
  DataUrodzenia DATE NOT NULL,
  AdresZamieszkania NVARCHAR(256),
  NumerTelefonu NVARCHAR(9),
  Stanowisko NVARCHAR(256) NOT NULL,
  DataZatrudnienia DATE NOT NULL,
  StawkaGodzinowa MONEY NOT NULL,
  WymiarEtatu DECIMAL

  PRIMARY KEY(OsobaID),
  FOREIGN KEY(OsobaID) REFERENCES Osoby(OsobaID),
  FOREIGN KEY(Stanowisko) REFERENCES Stanowiska(NazwaStanowiska)
);
GO
```

Tabela Progresja Kar

```
CREATE TABLE ProgresjaKar(
  LiczbaPrzewinień INT UNIQUE NOT NULL,
  Modyfikator DECIMAL NOT NULL,
  PRIMARY KEY (LiczbaPrzewinień)
);
GO
```

Tabela Przewinienia

```
CREATE TABLE Przewinienia(
  PrzewinienieID INT IDENTITY(1,1),
  PrzewinienieRodzaj NVARCHAR(256) UNIQUE NOT NULL,
  AktualnaKwotaKary MONEY NOT NULL,

  PRIMARY KEY(PrzewinienieID)
);
GO
```

Tabela Przystanki

```
CREATE TABLE Przystanki(
    PrzystanekID NVARCHAR(4),
    NazwaPrzystanku NVARCHAR NOT NULL,
    IlośćWiat TINYINT NOT NULL,
    NaŻądanie BIT,
    ElektronicznaInformacja BIT,
    AutomatBiletowy BIT,
    Strefa INT

    PRIMARY KEY(PrzystanekID),
```

```
    FOREIGN KEY(Strefa) REFERENCES Strefy(StrefaID)
);
GO
```

Tabela Stanowiska

```
CREATE TABLE Stanowiska(
  NazwaStanowiska NVARCHAR(256) NOT NULL,
  PRIMARY KEY (NazwaStanowiska)
);
GO
```

Tabela Strefy

```
CREATE TABLE Strefy(
  StrefaID INT IDENTITY(1, 1),

  PRIMARY KEY (StrefaID)
);
GO
```

Tabela Trasy

```
CREATE TABLE Trasy(
    TrasaID INT,
    LiniaID INT NOT NULL,
    PrzystanekPoczątkowy NVARCHAR(4) NOT NULL

    PRIMARY KEY(TrasaID),
    FOREIGN KEY(LiniaID) REFERENCES Linie(LiniaID),
    FOREIGN KEY(PrzystanekPoczątkowy) REFERENCES Przystanki(PrzystanekID)
);
GO
```

Tabela Usługi

```
CREATE TABLE Usługi(
    NazwaUsługi NVARCHAR

    PRIMARY KEY(NazwaUsługi)
);
GO
```

Tabela Wykonane kursy

```
CREATE TABLE WykonaneKursy(
    KursID INT,
    DataKursu DATE NOT NULL,
    PracownikID INT NOT NULL,
    AutobusID INT NOT NULL

    PRIMARY KEY(KursID,DataKursu),
    FOREIGN KEY(KursID) REFERENCES Kursy(KursID),
    FOREIGN KEY(PracownikID) REFERENCES Kierowcy(PracownikID),
    FOREIGN KEY(AutobusID) REFERENCES Autobusy(AutobusID)
);
GO
```

Wyzwalacze

Wyzwalacz Kursy INSERT

```
CREATE TRIGGER KursyINSERT ON Kursy
AFTER INSERT
AS
    IF EXISTS(SELECT * FROM inserted WHERE (DniPowszednie=1 AND Soboty=1)
    OR (DniPowszednie=1 AND Niedziele=1) OR (Soboty=1 AND Niedziele=1))
    BEGIN
        RAISERROR('Niejednoznaczne przypisanie dnia kursu', 0, 1)
        ROLLBACK TRANSACTION
    END

    IF EXISTS(SELECT K.KursID,K.GodzinaOdjazdu,
    dbo.ADDTIME(S.Czas,K.GodzinaOdjazdu) GodzinaPrzyjazdu,
    K.PracownikID,K.AutobusID
    FROM (SELECT Cz.TrasaID,
        MAX(CzasPrzejZPoczątku) Czas
        FROM CzasyPrzejazdu Cz
        GROUP BY(Cz.TrasaID)) S
    JOIN CzasyPrzejazdu C ON S.Czas=C.CzasPrzejZPoczątku AND S.TrasaID=C.TrasaID
    JOIN inserted K ON C.TrasaID=K.TrasaID JOIN Trasy T ON K.TrasaID=T.TrasaID
CROSS APPLY

HarmonogramPracyKierowcy(K.PracownikID,IIF(K.Soboty=1,6,IIF(K.Niedziele=1,7,1))) H
    WHERE (K.GodzinaOdjazdu<H.GodzinaOdjazdu AND
GodzinaPrzyjazdu>H.GodzinaOdjazdu) OR
    (K.GodzinaOdjazdu<H.GodzinaPrzyjazdu AND K.GodzinaOdjazdu>H.GodzinaOdjazdu))
    BEGIN
        RAISERROR('Kierowca ma już inny kurs w harmonogramie w tym czasie', 0, 1)
        ROLLBACK TRANSACTION
    END
```

```
    IF EXISTS(SELECT K.KursID,K.GodzinaOdjazdu,
    dbo.ADDTIME(S.Czas,K.GodzinaOdjazdu) GodzinaPrzyjazdu,
    K.PracownikID,K.AutobusID
    FROM (SELECT Cz.TrasaID,
        MAX(CzasPrzejZPoczątku) Czas
        FROM CzasyPrzejazdu Cz
        GROUP BY(Cz.TrasaID)) S
    JOIN CzasyPrzejazdu C ON S.Czas=C.CzasPrzejZPoczątku AND S.TrasaID=C.TrasaID
    JOIN inserted K ON C.TrasaID=K.TrasaID JOIN Trasy T ON K.TrasaID=T.TrasaID
CROSS APPLY

HarmonogramJazdyAutobusu(K.PracownikID,IIF(K.Soboty=1,6,IIF(K.Niedziele=1,7,1))) H
    WHERE (K.GodzinaOdjazdu<H.GodzinaOdjazdu AND
GodzinaPrzyjazdu>H.GodzinaOdjazdu) OR
    (K.GodzinaOdjazdu<H.GodzinaPrzyjazdu AND K.GodzinaOdjazdu>H.GodzinaOdjazdu))
    BEGIN
        RAISERROR('Autobus ma już inny kurs w harmonogramie w tym czasie', 0, 1)
        ROLLBACK TRANSACTION
    END
GO
```

Wyzwalacz Kursy Update

```
CREATE TRIGGER KursyUPDATE ON Kursy
AFTER UPDATE
AS
    IF UPDATE(GodzinaOdjazdu)
    BEGIN
        RAISERROR('Nie można zmienić godziny odjazdu, dodaj nowy kurs', 0, 1)
        ROLLBACK TRANSACTION
    END
    IF EXISTS(SELECT * FROM inserted WHERE (DniPowszednie=1 AND Soboty=1)
    OR (DniPowszednie=1 AND Niedziele=1) OR (Soboty=1 AND Niedziele=1))
    BEGIN
        RAISERROR('Niejednoznaczne przypisanie dnia kursu', 0, 1)
        ROLLBACK TRANSACTION
    END

    IF UPDATE(PracownikID)
    BEGIN
        IF EXISTS(SELECT K.KursID,K.GodzinaOdjazdu,
        dbo.ADDTIME(S.Czas,K.GodzinaOdjazdu) GodzinaPrzyjazdu,
        K.PracownikID,K.AutobusID
        FROM (SELECT Cz.TrasaID,
            MAX(CzasPrzejZPoczątku) Czas
            FROM CzasyPrzejazdu Cz
            GROUP BY(Cz.TrasaID)) S
        JOIN CzasyPrzejazdu C ON S.Czas=C.CzasPrzejZPoczątku AND
S.TrasaID=C.TrasaID
        JOIN inserted K ON C.TrasaID=K.TrasaID JOIN Trasy T ON K.TrasaID=T.TrasaID
CROSS APPLY
```

```
HarmonogramPracyKierowcy(K.PracownikID,IIF(K.Soboty=1,6,IIF(K.Niedziele=1,7,1))) H
        WHERE (K.GodzinaOdjazdu<H.GodzinaOdjazdu AND
GodzinaPrzyjazdu>H.GodzinaOdjazdu) OR
        (K.GodzinaOdjazdu<H.GodzinaPrzyjazdu AND
K.GodzinaOdjazdu>H.GodzinaOdjazdu))
        BEGIN
            RAISERROR('Kierowca ma już inny kurs w harmonogramie w tym czasie', 0,
1)
            ROLLBACK TRANSACTION
        END
    END

    IF UPDATE(AutobusID)
    BEGIN
        IF EXISTS(SELECT K.KursID,K.GodzinaOdjazdu,
        dbo.ADDTIME(S.Czas,K.GodzinaOdjazdu) GodzinaPrzyjazdu,
        K.PracownikID,K.AutobusID
        FROM (SELECT Cz.TrasaID,
            MAX(CzasPrzejZPoczątku) Czas
            FROM CzasyPrzejazdu Cz
            GROUP BY(Cz.TrasaID)) S
        JOIN CzasyPrzejazdu C ON S.Czas=C.CzasPrzejZPoczątku AND
S.TrasaID=C.TrasaID
        JOIN inserted K ON C.TrasaID=K.TrasaID JOIN Trasy T ON K.TrasaID=T.TrasaID
CROSS APPLY

HarmonogramJazdyAutobusu(K.PracownikID,IIF(K.Soboty=1,6,IIF(K.Niedziele=1,7,1))) H
        WHERE (K.GodzinaOdjazdu<H.GodzinaOdjazdu AND
GodzinaPrzyjazdu>H.GodzinaOdjazdu) OR
        (K.GodzinaOdjazdu<H.GodzinaPrzyjazdu AND
K.GodzinaOdjazdu>H.GodzinaOdjazdu))
        BEGIN
            RAISERROR('Autobus ma już inny kurs w harmonogramie w tym czasie', 0,
1)
            ROLLBACK TRANSACTION
        END
    END
GO
```

Wyzwalacz NałożoneKary ALL

```
CREATE TRIGGER NałożoneKaryALL
ON NałożoneKary
INSTEAD OF INSERT, UPDATE, DELETE
AS
  RAISERROR('Zakaz manualnej ingerencji w mandaty!', 0, 1)
GO
```

Wyzwalacz Osoby INSERT

```
CREATE TRIGGER OsobyINSERT
ON Osoby
INSTEAD OF INSERT
AS
  IF EXISTS(
    SELECT * FROM inserted
    WHERE PESEL IS NOT NULL AND PESEL NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9][0-9][0-9]' OR
      NrDowoduOsobistego IS NOT NULL AND NrDowoduOsobistego NOT LIKE '[A-Z][A-Z]
[A-Z][0-9][0-9][0-9][0-9][0-9][0-9]')
    RAISERROR('Niepoprawne dane!', 0, 1)

  INSERT INTO Osoby
  SELECT Imię, Nazwisko, PESEL, NrDowoduOsobistego FROM inserted
  WHERE (PESEL IS NULL OR PESEL LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9]') AND
      (NrDowoduOsobistego IS NULL OR NrDowoduOsobistego LIKE '[A-Z][A-Z][A-Z][0-9]
[0-9][0-9][0-9][0-9][0-9]')
GO
```

Wyzwalacz Osoby UPDATE

```
CREATE TRIGGER OsobyUPDATE
ON Osoby
INSTEAD OF UPDATE
AS
  IF EXISTS(
    SELECT * FROM inserted
    WHERE PESEL IS NOT NULL AND PESEL NOT LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9][0-9][0-9]' OR
      NrDowoduOsobistego IS NOT NULL AND NrDowoduOsobistego NOT LIKE '[A-Z][A-Z]
[A-Z][0-9][0-9][0-9][0-9][0-9][0-9]')
    RAISERROR('Niepoprawne dane!', 0, 1)

  UPDATE
    Osoby
  SET
    Imię = I.Imię,
    Nazwisko = I.Nazwisko,
    PESEL = I.PESEL,
    NrDowoduOsobistego = I.NrDowoduOsobistego
  FROM
    Osoby O
  INNER JOIN inserted I
  ON O.OsobaID = I.OsobaID
  WHERE (I.PESEL IS NULL OR I.PESEL LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9]') AND
      (I.NrDowoduOsobistego IS NULL OR I.NrDowoduOsobistego LIKE '[A-Z][A-Z][A-Z]
[0-9][0-9][0-9][0-9][0-9][0-9]') AND
```

```
      O.OsobaID = I.OsobaID
GO
```

Widoki

Widok Czasy trwania kursów

```
CREATE VIEW CzasyTrwaniaKursów
AS
    SELECT K.KursID,K.GodzinaOdjazdu,T.PrzystanekPoczątkowy PrzystanekOdjazdu,
    dbo.ADDTIME(S.Czas,GodzinaOdjazdu) GodzinaPrzyjazdu, C.PrzystanekID
PrzystanekPrzyjazdu,
    K.DniPowszednie,K.Soboty,K.Niedziele,K.PracownikID,K.AutobusID
    FROM (SELECT Cz.TrasaID,
        MAX(CzasPrzejZPoczątku) Czas
        FROM CzasyPrzejazdu Cz
        GROUP BY(Cz.TrasaID)) S
    JOIN CzasyPrzejazdu C ON S.Czas=C.CzasPrzejZPoczątku AND S.TrasaID=C.TrasaID
    JOIN Kursy K ON C.TrasaID=K.TrasaID JOIN Trasy T ON K.TrasaID=T.TrasaID
GO
```

Widok Kierowcy badania

```
CREATE VIEW KierowcyBadania
AS
SELECT P.PracownikID, O.Imię, O.Nazwisko, T.UpływająBadaniaZPrawaJazdy,
T.UpływająBadaniaLekarskie
FROM (
  SELECT K.PracownikID,
    CASE
      WHEN DATEDIFF(MONTH, GETDATE(), K.DataWażnościPrawaJazdy) < 3
        THEN N'TAK'
      ELSE N'NIE'
    END
    UpływająBadaniaZPrawaJazdy,
    CASE
      WHEN DATEDIFF(MONTH, GETDATE(), K.DataWażnościBadańLekarskich) < 3
       THEN N'TAK'
      ELSE N'NIE'
    END
    UpływająBadaniaLekarskie
  FROM Kierowcy K
) AS T
JOIN Pracownicy P
ON P.PracownikID = T.PracownikID
JOIN Osoby O
```

```
ON P.OsobaID = O.OsobaID
GO
```

Widok Przedawnione kary

```
CREATE VIEW PrzedawnioneKary
AS
SELECT O.Imię, O.Nazwisko, O.PESEL, O.NrDowoduOsobistego, T.Suma
FROM Osoby O
JOIN (
  SELECT NK.UkaranyID, SUM(NK.KwotaKary) Suma
  FROM NałożoneKary NK
  WHERE NK.DataOpłacenia IS NOT NULL AND
    DATEDIFF(DAY, GETDATE(), NK.DataUkarania) > 30
  GROUP BY NK.UkaranyID) AS T
ON O.OsobaID = T.UkaranyID
GO
```

Widok Ranking kontrolerów

```
CREATE VIEW RankingKontrolerów
AS
SELECT P.PracownikID, O.Imię, O.Nazwisko, T.SumaKar
FROM (
  SELECT NK.KontrolerID, SUM(NK.KwotaKary) SumaKar
  FROM NałożoneKary NK
  WHERE DATEDIFF(DAY, GETDATE(), NK.DataUkarania) < 30
  GROUP BY NK.KontrolerID
) AS T
JOIN Kontrolerzy K
ON K.KontrolerID = T.KontrolerID
JOIN Pracownicy P
ON P.PracownikID = K.PracownikID
JOIN Osoby O
ON O.OsobaID = P.OsobaID
GO
```