Projekt relacyjnej bazy danych komunikacji miejskiej

Paulina Miśkowiec, Patrycja Sarga

20 lutego 2021

Spis treści

1 Podstawowe założenia projektu

Baza powstała w celu przechowywania informacji związanych z działaniem przedsiębiorstwa komunikacji miejskiej obsługującej połączenia autobusowe. Zawiera ona w takim razie informacje o rozkładach poszczególnych linii, przystankach, biletach; informacje o pracownikach, ich grafikach; dane o poszczególnych pojazdach.
Baza umożliwia wyszukiwanie interesujących daną osobę połączeń, rozkładów, informacji na temat linii, wyposażenia autobusów (np. czy posiada miejsce dla rowerów), czy danych statystycznych na temat awarii pojazdów.
Przy projektowaniu bazy przyjęte zostały ograniczenia upraszczające jej projektowanie:

- linie ograniczone są do jednego miasta tzn. podając np. nazwę ulicy na której znajduje się dany przystanek, wiadomo że jest to jedyna taka ulica w tym mieście;
- poszczególne linie mają dokładnie takie same trasy w obu kierunkach, tzn. nie jest dopuszczana sytaucja, gdy dana linia jadąc z przystanku A do B zatrzymuje się na przystanku, na którym nie zatrzymuje się jadąc z B do A;
- dwa (lub więcej) przystanki o danej nazwie znajdują się przy tej samej ulicy (nie ma potrzeby dalszego ich rozróżniania).

2 Tabele bazy danych

Tabela przechowująca dane o działach, jakie istnieją w spółce:

Dzialy

	Column Name	Condensed Type	Nullable
🔑	IdDzialu	int	No
	NazwaDzialu	varchar(50)	No
	Adres	varchar(50)	No
	NumerTelefo...	varchar(9)	No

Tabela przechowująca dane o stanowiskach:

Stanowiska

	Column Name	Condensed Type	Nullable
🔑	Stanowisko	varchar(50)	No
	IdDzialu	int	No
	Pensja	money	No
	Hierarchia	int	No

Tabela przechowująca dane o pracownikach:

Pracownicy

	Column Name	Condensed Type	Nullable
🔑	IdPracownika	int	No
	Imie	varchar(50)	No
	Nazwisko	varchar(50)	No
	PESEL	char(11)	No
	DataZatrudnienia	date	No
	NumerPrawaJazdy	char(10)	Yes
	Stanowisko	varchar(50)	No
	IdPrzelozonego	int	Yes

Tabela przechowująca dane o urlopach pracowników:

Urlopy_Pracowników

	Column Name	Condensed Type	Nullable
🔑	IdPracownika	int	No
🔑	OdKiedy	date	No
	DoKiedy	date	No

Tabela przechowująca dane o rodzajach biletów dostępnych dla użytkowików komunikacji miejskiej:

RodzajeBiletow

	Column Name	Condensed Type	Nullable
🔑	RodzajBiletu	varchar(50)	No
	Opis	varchar(256)	Yes

Tabela przechowująca dane o biletach:

Bilety

	Column Name	Condensed Type	Nullable
🔑	NazwaBiletu	varchar(50)	No
🔑	RodzajBiletu	varchar(50)	No
	Cena	money	No

Tabela przechowująca dane o przystankach, które są obsługiwane przez wszystkie linie komunikacji miejskiej:

Przystanki

	Column Name	Condensed Type	Nullable
🔑	IdPrzystanku	int	No
	NazwaPrzystanku	varchar(50)	No
	Ulica	varchar(50)	No
	Rodzaj	varchar(50)	Yes

Tabela przechowująca dane o liniach autobusowych:

Linie

	Column Name	Condensed Type	Nullable
🔑	NumerLinii	int	No
	RodzajLinii	varchar(50)	Yes
	IdPrzystankuPoczatkowego	int	No
	IdPrzystankuKoncowego	int	No

Tabela przechowująca dane o trasach, jakimi poruszają się autobusy na konkretnych liniach:

Trasy

	Column Name	Condensed Type	Nullable
🔑	NumerLinii	int	No
🔑	NumerPorządkowyPrzystanku	int	No
	IdPrzystanku	int	No

Tabela przechowująca dane o zajezdniach autobusowych:

Zajezdnie

Column Name	Condensed Type	Nullable
IdZajezdni	int	No
NazwaZajezdni	varchar(50)	No
AdresZajezdni	varchar(100)	No

Tabela przechowująca dane o autobusach:

Autobusy

Column Name	Condensed Type	Nullable
IdAutobusu	int	No
NrRejestracyjny	varchar(10)	No
OdKiedy	datetime	No
IloscMiejscSiedzacych	int	No
IloscMiejscStojacych	int	No
Rodzaj	varchar(20)	No
Biletomat	bit	No
MiejsceDlaRowerow	bit	No
MiejsceDlaWozkow	bit	No
Przegląd	varchar(20)	Yes
DataAktualnegoPrzegladu	datetime	No
IdZajezdni	int	No

Tabela przechowująca dane o obsłudze technicznej, dbającej o sprawność pojazdów i bezpieczeństwo pasażerów:

Obsluga_techniczna

Column Name	Condensed Type	Nullable
IdPojazdu	int	No
NrRejestracyjny	varchar(10)	No
OdKiedy	datetime	No
DataAktualnegoPrzeglądu	datetime	No
Rodzaj	varchar(50)	Yes

Tabela przechowująca dane o awariach pojazdów:

DziennikAwarii

Column Name	Condensed Type	Nullable
IdAutobusu	int	No
IdPojazduObslugi	int	No
Data	date	No
[Opis zdarzenia]	varchar(256)	Yes

Tabela przechowująca dane o kursach autobusów:

Kursy

Column Name	Condensed Type	Nullable
IdKursu	int	No
NumerLinii	int	No
Kierunek	int	No

Tabela przechowująca dane o rozkładach:

Rozklady

Column Name	Condensed Type	Nullable
IdPrzystanku	int	No
IdKursu	int	No
Godzina	time(0)	No
Dzien	varchar(10)	No

Tabela przechowująca dane o tym, jakie autobusy kursują na konkretnych liniach:

Pojazdy_i_Linie

Column Name	Condensed Type	Nullable
IdAutobusu	int	No
NumerLinii	int	No

Tabela przechowująca dane o grafikach pracowników:

Grafiki_Pracownikow

Column Name	Condensed Type	Nullable
IdKursu	int	No
IdPracownika	int	No
Godzina	time(0)	No

Tabela przechowująca dane o zależnościach służbowych między pacownikami:

*powinna być aktualizowana po każdej aktualizacji w tabeli Pracownicy

Przelozeni

Column Name	Condensed Type	Nullable
IdPrzelozonego	int	Yes
IdPodwladnego	int	Yes

3 Diagramy bazy danych

Diagram relacji bazy danych

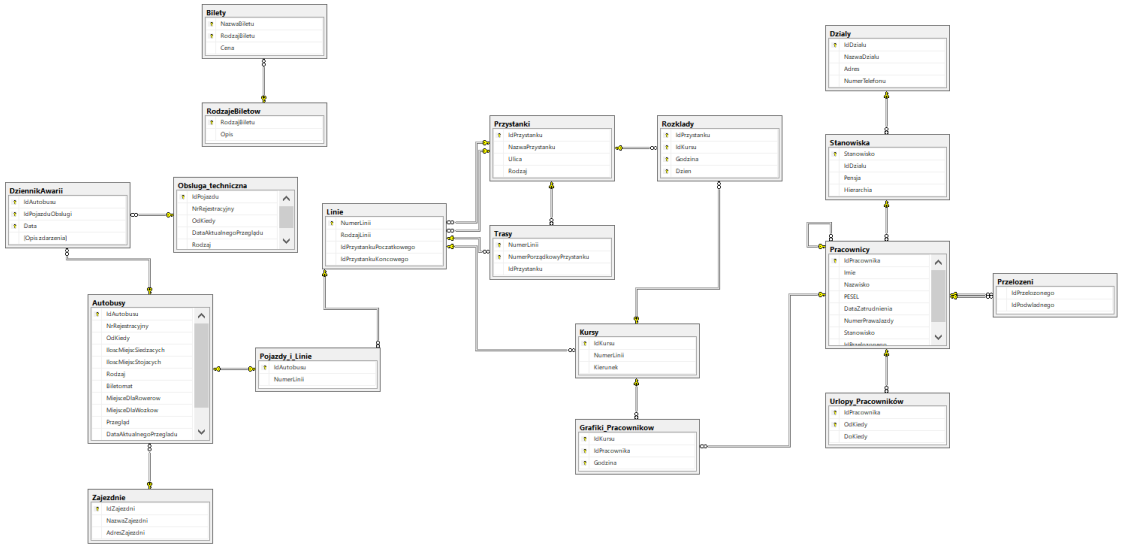
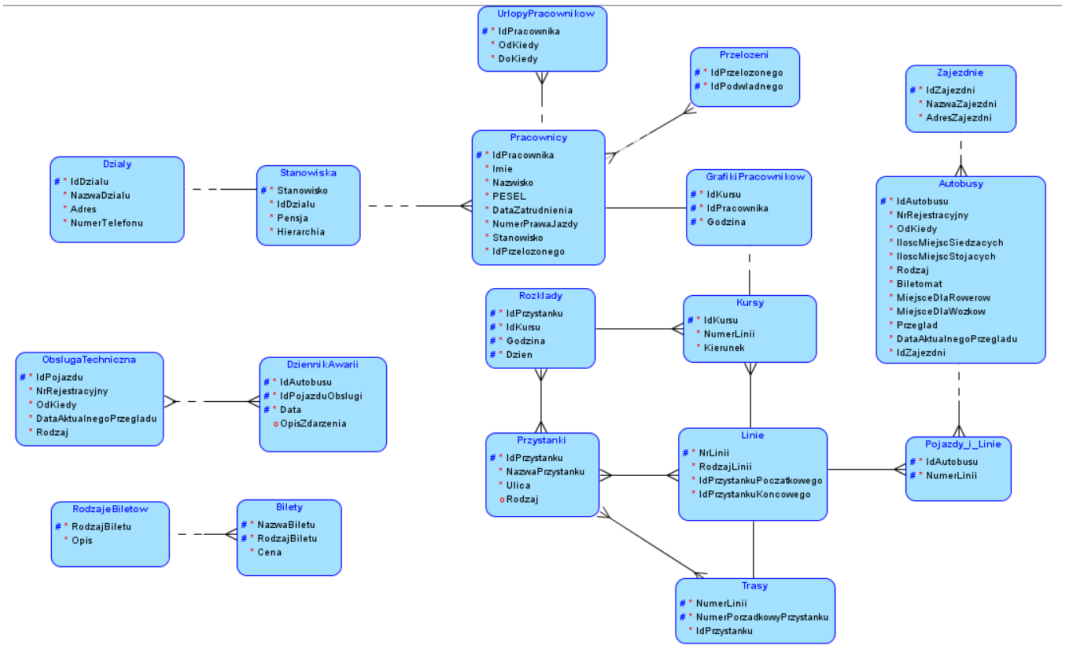


Diagram ER


UrlopyPracownikow
IdPracownika
OdKiedy
DoKiedy
Przelozeni
IdPrzelozonego
IdPodwladnego
Zajezdnie
IdZajezdni
NazwaZajezdni
AdresZajezdni
Dzialy
IdDzialu
NazwaDzialu
Adres
NumerTelefonu
Stanowiska
Stanowisko
IdDzialu
Pensja
Hierarchia
Pracownicy
IdPracownika
Imie
Nazwisko
PESEL
DataZatrudnienia
NumerPrawaJazdy
Stanowisko
IdPrzelozonego
GrafikiPracownikow
IdKursu
IdPracownika
Godzina
Autobusy
IdAutobusu
NrRejestracyjny
OdKiedy
IloscMiejscSiedzacych
IloscMiejscStojacych
Rodzaj
Biletomat
MiejsceDlaRowerow
MiejsceDlaWozkow
Przeglad
DataAktualnegoPrzegladu
IdZajezdni
Rozklady
IdPrzystanku
IdKursu
Godzina
Dzien
Kursy
IdKursu
NumerLinii
Kierunek
ObslugaTechniczna
IdPojazdu
NrRejestracyjny
OdKiedy
DataAktualnegoPrzegladu
Rodzaj
DziennikAwarii
IdAutobusu
IdPojazduObslugi
Data
OpisZdarzenia
Przystanki
IdPrzystanku
NazwaPrzystanku
Ulica
Rodzaj
Linie
NrLinii
RodzajLinii
IdPrzystankuPoczatkowego
IdPrzystankuKoncowego
Pojazdy_i_Linie
IdAutobusu
NumerLinii
RodzajeBiletow
RodzajBiletu
Opis
Bilety
NazwaBiletu
RodzajBiletu
Cena
Trasy
NumerLinii
NumerPorzadkowyPrzystanku
IdPrzystanku

4 Wyzwalacze

Wyzwalacz kontrolujący dane kierowców

```
GO
CREATE TRIGGER tr1_instead_of_insert on Pracownicy
INSTEAD OF INSERT
AS
DECLARE @COUNT1 INT
SELECT @COUNT1 = COUNT(*) FROM dbo.Pracownicy

DECLARE @COUNT_IN INT
SELECT @COUNT_IN = COUNT(*) FROM inserted

 INSERT INTO Pracownicy
    SELECT * FROM   inserted
    WHERE  EXISTS (SELECT 1
WHERE  ((Stanowisko = N'Kierowca' AND NumerPrawaJazdy IS NOT NULL) OR (Stanowisko <> N'Kierowca')))

DECLARE @COUNT2 INT
SELECT @COUNT2 = COUNT(*) FROM dbo.Pracownicy

IF ( @COUNT_IN + @COUNT1 > @COUNT2 )
PRINT 'Error: Nie wszystkie rekordy zostały dodane! Kierowca musi posiadac numer prawa jazdy!'
ELSE
PRINT 'Rekord(y) został dodany.'
GO
```

Jak działa? Wyzwalacz ten służy do kontroli warunku, czy zatrudnieni na stanowisku kierowcy mają uzupełnioną kolumnę z numerem prawa jazdy (nie może dla nich ona zawierać NULL). Po wprowadzeniu nowych rekordów sprawdzany jest ten warunek i jeśli jest on spełniony dla wszystkich nowych rekordów, to są one wpisywane do tabeli Pracownicy. Rekordy niespełniające warunku nie są do niej wpisywane i wyświetlany jest odpowiedni komunikat.

Wyzwalacz sprawdzający pojemność zajezdni

```
GO
CREATE TRIGGER tr2_instead_of_insert ON Autobusy
INSTEAD OF INSERT
AS
    DECLARE @NrZajezdni INT
    SELECT  @NrZajezdni = IdZajezdni FROM Inserted

    DECLARE @LiczbaAutobusow INT
    SELECT @LiczbaAutobusow = COUNT(IdAutobusu) FROM Autobusy WHERE IdZajezdni=@NrZajezdni
  IF @LiczbaAutobusow <10
    BEGIN
    INSERT INTO Autobusy
    SELECT * FROM inserted
    PRINT 'Oddano autobus do uzytku'
    END
  ELSE
    BEGIN
    PRINT 'Limit osiągniety! Umiesc autobus w innej zajezdni.'
    END
GO
```

Jak działa? Wyzwalacz sprawdza ile autobusów jest aktualnie w zajezdni, do której podjęto próbę przypisania autobusu (poprzez wstawianie rekordu). Jeżeli limit liczby autobusów (tutaj 10) nie został osiągnięty, następuje wstawienie rekordu. W przeciwnym razie wyświetlany jest komunikat, aby umieścić autobus w innej zajezdni.

Wyzwalacz kontrolujący grafik pracowników

```
CREATE TRIGGER tr3_instead_of_insert
ON Grafiki_Pracownikow
INSTEAD OF INSERT
AS
    DECLARE @czaspracy TIME(0)
    SELECT @czaspracy = Godzina FROM inserted
    DECLARE @IdKursuG INT
    SELECT @IdKursuG = IdKursu FROM inserted
    DECLARE @Pracownik INT
    SELECT @Pracownik = IdPracownika FROM inserted

IF EXISTS (SELECT IdPracownika =@Pracownik FROM Pracownicy WHERE Stanowisko NOT LIKE 'kierowca' )
BEGIN
    INSERT INTO Grafiki_Pracownikow
    SELECT * FROM inserted
    PRINT 'DODANO!'
END
```

```
ELSE IF EXISTS (SELECT IdPracownika =@Pracownik FROM Pracownicy WHERE Stanowisko LIKE 'kierowca' )
BEGIN
    IF EXISTS (SELECT * FROM Rozklady WHERE IdKursu = @IdKursuG AND Godzina = @czaspracy)
    BEGIN
        INSERT INTO Grafiki_Pracownikow
        SELECT * FROM inserted
        PRINT 'DODANO!'
    END
    ELSE
    BEGIN
        PRINT 'Nie można dodać rekordu! Podany kurs lub godzina odjazdu nie istnieje w rozkładzie!'
    END
END
GO
```

Jak działa? Przy próbie wstawienia rekordu do tabeli dotyczącej grafików pracowników, w przypadku gdy rekord dotyczy pracownika na stanowisku kierowcy, sprawdzane jest czy godzina przypisywana danemu pracownikowi ma swoje pokrycie w rozkładzie jazdy - zapobiega to przypisywaniu kierowcy godzin pracy w przypadku gdy de facto tej pracy dla niego nie ma.

Wyzwalacz kontrolujący wysokość pensji

```
GO
CREATE TRIGGER tr4_instead_of_update ON Stanowiska
INSTEAD OF UPDATE
AS
    DECLARE @hierarchia INT
    SELECT @hierarchia = Hierarchia FROM inserted

    DECLARE @pensja MONEY
    SELECT @pensja = Pensja FROM inserted

IF EXISTS
(SELECT * FROM Stanowiska WHERE Hierarchia + 1 = @hierarchia AND Pensja<=@pensja)
    BEGIN
    PRINT 'Wprowadzana wartość pensji jest za wysoka dla tego stanowiska!'
    END
ELSE IF EXISTS
(SELECT * FROM Stanowiska WHERE Hierarchia - 1 = @hierarchia AND Pensja>=@pensja)
    BEGIN
    PRINT 'Wprowadzana wartość pensji jest za niska dla tego stanowiska!'
    END
ELSE
    BEGIN
    UPDATE Stanowiska
    SET Pensja = @pensja
    WHERE Hierarchia = @hierarchia
    PRINT 'Zaktualizowano!'
    END
GO
```

Jak działa? Wyzwalacz ma za zadanie dopilnować, aby w wyniku aktualizacji wartości w bazie, pensja pracownika na niższym stanowisku nie przekroczyła pensji pracownika na wyższym stanowisku - zapobiega to chociażby takiej sytuacji, kiedy pracownik zarabia więcej niż jego przełożony.

Wyzwalacz kontrolujący stan przeglądu technicznego pojazdów komunikacji miejskiej

```
GO
CREATE TRIGGER tr5_after_update ON Autobusy
AFTER UPDATE
AS
    DECLARE @aktualnadata DATETIME
    SELECT @aktualnadata = GETDATE()
    DECLARE @dataprzegladu DATETIME
    SELECT @dataprzegladu = DataAktualnegoPrzegladu FROM inserted

IF @dataprzegladu>=@aktualnadata
    BEGIN
    UPDATE Autobusy
    SET Przegląd = 'aktualny'
    WHERE IdAutobusu = (SELECT IdAutobusu FROM inserted)
    PRINT 'Zaktualizowano stan przeglądu!'
    END

IF EXISTS
(SELECT * FROM Autobusy WHERE DataAktualnegoPrzegladu<@aktualnadata)
    BEGIN
    UPDATE Autobusy
    SET Przegląd = 'nieaktualny'
    WHERE DataAktualnegoPrzegladu<@aktualnadata
    PRINT 'Upłynął termin ważności przeglądu technicznego!'
    END
GO
```

Jak działa? Wyzwalacz zmienia wartość w kolumnie mówiącej o stanie przeglądów technicznych pojazdów po aktualizacji daty przeglądu na "aktywny" (w przypadku gdy nie zrobiono przeglądu przed zakończeniem ważności poprzedniego przeglądu i wartość stanu przeglądu zdążyła się zmienić na "nieaktywny").
Ponadto przy okazji każdej aktualizacji sprawdzany jest stan przeglądu: jeżeli minęła data ważności przeglądu, stan przeglądu zmienia się na "nieaktywny" i wyświetlany jest komunikat o konieczności zrobienia przeglądu.

5 Procedury składowane

Zmiana cen biletów

```
GO
CREATE PROCEDURE ZmianaCen (@zmiana FLOAT)
AS
BEGIN
    UPDATE Bilety SET Cena = ROUND((Cena + (Cena * @zmiana)),1)
END
GO
```

Jak działa? Procedura podnosi (lub obniża) cenę wszystkich biletów o podany procent wyrażony ułamkiem dziesiętnym.

Dane statystyczne o awariach autobusów

```
GO
CREATE PROCEDURE IleAwarii
(@OdKiedy DATE, @DoKiedy DATE, @ile INT )
AS BEGIN
    SELECT P.IdAutobusu, P.Data, P.[Opis zdarzenia] FROM DziennikAwarii AS P LEFT JOIN
    (SELECT IdAutobusu, COUNT(*) AS Licznik FROM DziennikAwarii
    WHERE Data BETWEEN @OdKiedy AND @DoKiedy
    GROUP BY IdAutobusu) AS K
    ON P.IdAutobusu = K.IdAutobusu
    WHERE K.Licznik >= @ile AND P.Data BETWEEN @OdKiedy AND @DoKiedy
END
GO
```

Jak działa? Procedura pozwala na monitorowanie ilości awarii autobusów - wypisuje on spis awarii autobusów, które w podanych ramach czasowych miały co najmniej tyle awarii, ile wynosi podana na wejściu wartość.

Aktualizacja daty przeglądu technicznego

```
GO
CREATE PROCEDURE AktualizacjaDaty (@numer INT, @data DATETIME)
AS
BEGIN
    UPDATE Obsluga_techniczna
    SET DataAktualnegoPrzeglądu = @data
    WHERE IdPojazdu = @numer
END
GO
```

Jak działa? Procedura pozwala na szybką aktualizację daty ważności przeglądu technicznego pojazdu obsługi technicznej. Jest to swego rodzaju alternatywa dla wcześniej przedstawionego wyzwalacza.

Ilość kursów w podanym dniu

```
GO
CREATE PROCEDURE Ile_kursow (@dzien VARCHAR(10))
AS
BEGIN
    SELECT DISTINCT K.NumerLinii, COUNT(*) AS LiczbaKursow FROM
    (Rozklady R JOIN Kursy K ON R.IdKursu=K.IdKursu) JOIN Linie L ON K.NumerLinii=L.NumerLinii
    WHERE R.IdPrzystanku=L.IdPrzystankuPoczatkowego AND R.Dzien=@dzien
    GROUP BY K.NumerLinii, Dzien
END
GO
```

Jak działa? Procedura wyświetla dla kadej linii ilość kursów, jakie są wykonywane w podanym dniu (powszedni/sobota/święta). Procedura może być wykorzystywana w celu porównania ilości kursów wykonywanych przez dane linie w danym typie dnia, by móc ewentualnie tę ilość wyregulować.

Udogonienia dla pasażerów w autobusach

```
GO
CREATE PROCEDURE LiniaKryteria
(@rodzaj VARCHAR(50), @biletomat BIT, @rower BIT, @wozek BIT)
AS BEGIN

IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 1 AND @rower = 1 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy'
    AND Biletomat = 1 AND MiejsceDlaRowerow = 1 AND MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 1 AND @rower = 1 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy'
    AND Biletomat = 1 AND MiejsceDlaRowerow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu
```

```sql
IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 1 AND @rower = 0 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy' AND Biletomat = 1
    AND MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 0 AND @rower = 1 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy' AND
    MiejsceDlaRowerow = 1 AND MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 1 AND @rower = 1 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE Biletomat = 1 AND MiejsceDlaRowerow = 1
    AND MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu
IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 1 AND @rower = 0 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy' AND Biletomat = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 0 AND @rower = 0 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy' AND MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 0 AND @rower = 1 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy' AND MiejsceDlaRowerow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 1 AND @rower = 1 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE Biletomat = 1 AND MiejsceDlaRowerow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 1 AND @rower = 0 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE Biletomat = 1 AND MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 0 AND @rower = 1 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE MiejsceDlaRowerow = 1 AND MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu
```

```
IF (@rodzaj = 'niskopodlogowy' AND @biletomat = 0 AND @rower = 0 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE rodzaj = 'niskopodlogowy') AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 1 AND @rower = 0 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE Biletomat = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 0 AND @rower = 1 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE MiejsceDlaRowerow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 0 AND @rower = 0 AND @wozek = 1)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy WHERE MiejsceDlaWozkow = 1) AS S
    ON PL.IdAutobusu = S.IdAutobusu

IF (@rodzaj = 'wysokopodlogowy' AND @biletomat = 0 AND @rower = 0 AND @wozek = 0)
    SELECT DISTINCT PL.NumerLinii FROM Pojazdy_i_Linie AS PL RIGHT JOIN (
    SELECT IdAutobusu FROM Autobusy) AS S
    ON PL.IdAutobusu = S.IdAutobusu
END
GO
```

Jak działa? Procedura pozwala na wyszukiwanie linii, na których jeżdżą autobusy spełniające określone kryteria: nisko- lub wysokopodłogowe, obecność biletomatu, miejsce dla rowerów, miejsce dla wózków, przy czym jeśli przy wywołaniu procedury podane będzie, że jakieś kryterium jest nieważne to będą wyszukiwane pojazdy, które to kryterium spełniają i te, które go nie spełniają - przykładowo przy podaniu, że nie musi być obecny biletomat, procedura bierze pod uwagę zarówno te pojazdy, które go posiadają, jak i te w których nie jest on obecny.

6 Funkcje

Funkcja wypisująca rozkład jazdy dla podanej linii z podanego przystanku

```
GO
CREATE FUNCTION RozkladDlaPrzystanku (@nazwa VARCHAR(50), @numer INT)
RETURNS @RozkladPrzystankow TABLE (GodzinaOdjazdu TIME(0), Kierunek VARCHAR(50)
) AS BEGIN

    DECLARE @przystanek INT
    SET @przystanek = (SELECT IdPrzystanku FROM Przystanki WHERE NazwaPrzystanku = @nazwa)

    INSERT INTO @RozkladPrzystankow
    SELECT RK.Godzina AS [Godzina Odjazdu], P.NazwaPrzystanku AS Kierunek FROM (
    SELECT R.Godzina, K.Kierunek FROM Rozklady AS R LEFT JOIN Kursy AS K
    ON R.IdKursu = K.IdKursu
    WHERE K.NumerLinii = @numer AND R.IdPrzystanku = @przystanek)
    AS RK LEFT JOIN Przystanki AS P
    ON RK.Kierunek = P.IdPrzystanku
    RETURN
END
GO
```

Jak działa? Po podaniu nazwy przystanku i numeru linii wypisuje ona rozkład tej linii, a więc godzinę i kierunek.

Funkcja podająca spis linii zatrzymujących się na danym przystanku

```
GO
CREATE FUNCTION JakieLinie (@nazwa VARCHAR(50))
RETURNS @LinieKierunki TABLE (NumerLinii INT, Kierunek VARCHAR(50)
) AS BEGIN
    INSERT INTO @LinieKierunki
    SELECT KPR.NumerLinii, PP.NazwaPrzystanku AS Kierunek FROM (
    SELECT K.NumerLinii, K.Kierunek FROM Kursy AS K RIGHT JOIN (
    SELECT DISTINCT R.IdKursu FROM Rozklady AS R LEFT JOIN Przystanki AS P
    ON R.IdPrzystanku = P.IdPrzystanku
    WHERE P.NazwaPrzystanku = @nazwa ) AS PR
    ON K.IdKursu = PR.IdKursu) AS KPR LEFT JOIN
    Przystanki AS PP
    ON KPR.Kierunek = PP.IdPrzystanku
    RETURN
END
GO
```

Jak działa? Funkcja po podaniu nazwy przystanku wypisuje wszystkie linie, które się na niej zatrzymują, wraz z kierunkami, w którym jadą.

Funkcja wywołująca wszystkie przystanki podanej linii

```
GO
CREATE FUNCTION JakiePrzystanki (@numer INT)
RETURNS @RozkladPrzystankow TABLE (NazwaPrzystanku VARCHAR(50)
) AS BEGIN
    INSERT INTO @RozkladPrzystankow
    SELECT P.NazwaPrzystanku FROM (
    SELECT IdPrzystanku FROM Trasy
    WHERE NumerLinii = @numer) AS T LEFT JOIN Przystanki AS P
    ON T.IdPrzystanku = P.IdPrzystanku
    RETURN
END
GO
```

Jak działa? Wywołana funkcja po podaniu numeru linii, którą jesteśmy zainteresowani, wyświetla wszystkie przystanki, które pojawiają się na jej trasie.

Funkcja wywołująca wszystkie przystanki z podanej ulicy

```
GO
CREATE FUNCTION PrzystankiZUlicy (@nazwa VARCHAR(50))
RETURNS @RozkladPrzystankow TABLE (NazwaPrzystanku VARCHAR(50))
AS BEGIN
    INSERT INTO @RozkladPrzystankow
    SELECT NazwaPrzystanku FROM Przystanki
    WHERE Ulica = @nazwa
    RETURN
END
GO
```

Jak działa? Funkcja wypisuje wszystkie przystanki jakie znajdują się przy podanej ulicy.

Funkcja sprawdzająca jakim autobusom zbliża się data przeglądu

```
GO
CREATE FUNCTION AutobusyDoPrzegladu ()
RETURNS @Autobusy TABLE (IdAutobusu INT, [Data aktualnego przegladu] DATE
) AS BEGIN
    INSERT INTO @Autobusy
    SELECT IdAutobusu, DataAktualnegoPrzegladu FROM Autobusy
    WHERE DATEDIFF(MONTH, GETDATE(), DataAktualnegoPrzegladu) <= 1
    RETURN
END
GO
```

Jak działa? Funkcja wypisuje wszystkie autobusy, które mają mniej niż miesiąc do kolejnego przeglądu (lub data ta już minęła).

Funkcja - wyszukiwanie trasy

```
GO
CREATE FUNCTION JakaLiniaDojechac (@nazwa1 VARCHAR(50), @nazwa2 VARCHAR(50))
RETURNS @Linie TABLE ([Numer Linii] INT
) AS BEGIN
    INSERT INTO @Linie
    SELECT R.NumerLinii FROM (
    SELECT K.NumerLinii, COUNT(*) AS Licznik FROM (
    SELECT T.NumerLinii, P.NazwaPrzystanku FROM Trasy AS T LEFT JOIN Przystanki AS P
    ON T.IdPrzystanku = P.IdPrzystanku
    WHERE P.NazwaPrzystanku = @nazwa1 OR P.NazwaPrzystanku = @nazwa2) AS K
    GROUP BY K.NumerLinii) AS R
    WHERE R.Licznik = 2
    RETURN
END
GO
```

Jak działa? Funkcja ta po podaniu nazw dwóch przystanków podaje numery wszystkich linii, którymi można się bezpośrednio między tymi przystankami przemieścić (działa to tylko w przypadku, gdy oba przystanki znajdują na trasie jakiejś linii).

7 Widoki

Widok tabeli Autobusy dla użytkownika

```
GO
CREATE VIEW AutobusyDlaUzytkownika AS
SELECT IdAutobusu, IloscMiejscSiedzacych, IloscMiejscStojacych,
Rodzaj, Biletomat, MiejsceDlaRowerow, MiejsceDlaWozkow FROM Autobusy
GO
```

Widok pokazuje informacje, które będą przydatne dla potencjalnego pasażera: numer autobusu, ilość miejsc siedzących i stojących, czy autobus jest nisko- czy wysokopodłogowy (rodzaj) oraz czy znajdują się w autobusie takie udogodnienia jak biletomat, miejsce dla roweru czy wózka.

Widok kierowców na zmianie

```
GO
CREATE VIEW KierowcyWPracy AS
SELECT DISTINCT P.IdPracownika, Imie, Nazwisko FROM Pracownicy P LEFT JOIN
Urlopy_Pracowników U ON P.IdPracownika=U.IdPracownika
WHERE Stanowisko = 'kierowca' AND (P.IdPracownika NOT IN
(SELECT IdPracownika FROM Urlopy_Pracowników) OR DoKiedy<GETDATE()OR
(DoKiedy>GETDATE() AND OdKiedy>GETDATE()))
GO
```

Widok wyświetla podstawowe informacje o kierowcach, którzy powinni być w pracy, tj. nie są na urlopie.

Widok linii nocnych

```
GO
CREATE VIEW LinieNocne AS
SELECT * FROM Linie WHERE RodzajLinii LIKE 'nocna'
GO
```

Widok wyświetla informacje o liniach nocnych.

Widok rozkładów świątecznych

```
GO
CREATE VIEW RozkladySwieta AS
SELECT * FROM Rozklady WHERE Dzien LIKE 'święta'
GO
```

Widok wyświetla rozkłady jazdy dla dni świątecznych.

8 Strategia pielęgnacji bazy danych

Stworzona baza danych nie zawiera zbyt wielu danych, których historia (logi) powinna być zapamiętywana, jak w przypadku np. banków. Tutaj wszystkie istotne dane zostają zapisane w rekordach tabel i to właśnie wyłącznie te tabele wymagają stworzenia kopii zapasowych.

Przedstawiona baza danych zawiera dość stałe informacje, których aktualizacja będzie się odbywać stosunkowo rzadko. Z tego powodu nie jest konieczne regularne tworzenie kopii zapasowych - wystarczy o nie zadbać po każdej aktualizacji.

Warto też pomyśleć o czyszczeniu bazy w celu pozyskania niepotrzebnie zajmowanej pamięci na dysku - takie czyszczenie mogłoby się odbywać raz w tygodniu.

9 Typowe zapytania

```
SELECT IdPracownika, IdKursu FROM Grafiki_Pracownikow
WHERE Godzina LIKE '11:30:00'
ORDER BY IdKursu

SELECT NazwaPrzystanku, Rodzaj FROM Przystanki
WHERE Ulica LIKE 'm%'
ORDER BY NazwaPrzystanku

SELECT Rodzaj, COUNT(IdAutobusu) FROM Autobusy
WHERE IdZajezdni = 3
GROUP BY Rodzaj

SELECT Imie, Nazwisko, PESEL FROM Pracownicy
WHERE DataZatrudnienia>'2017-09-01'
ORDER BY Imie, Nazwisko

SELECT NumerLinii, NumerPorządkowyPrzystanku FROM Trasy
WHERE IdPrzystanku=1

SELECT NumerLinii, IdPrzystankuKoncowego FROM Linie
WHERE IdPrzystankuPoczatkowego=1
```