WOJSKOWA AKADEMIA TECHNICZNA

Wydział Cybernetyki

SPRAWOZDANIE
Z PROJEKTU
BAZY DANYCH

Temat:

SPEDYCJA

Prowadzący: ppłk. dr inż. Jarosław KOSZELA

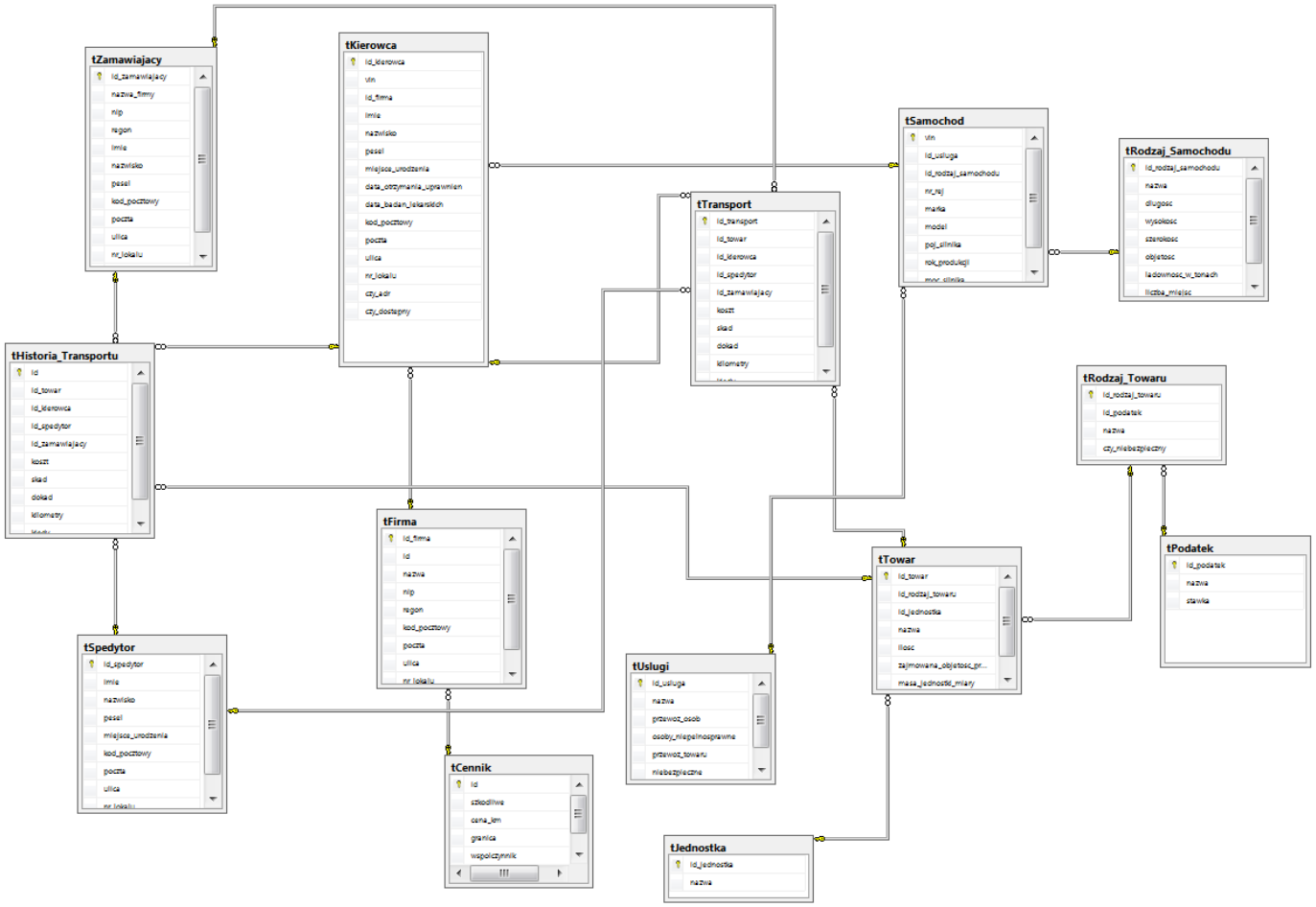
Wykonał: kpr. pchor. Damian KRATA (Nr albumu: 59223)

Grupa: I4X3S1

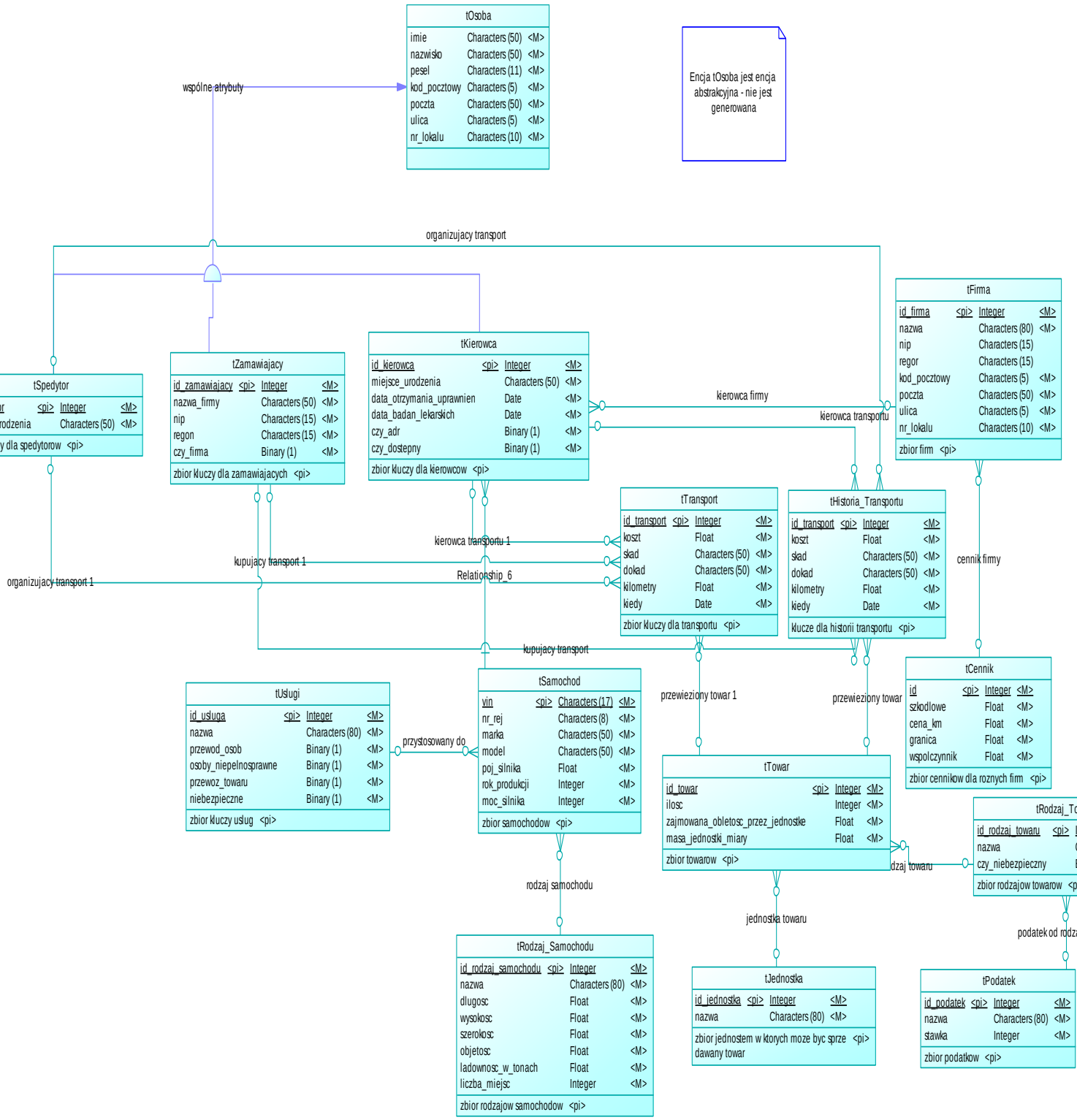
Data wykonania: 21.01.2016 r.

Baza danych Projekt przeznaczona jest do wsparcia informatycznego działalności organizacyjno-uzupełnieniowej, ewidencyjnej i sprawozdawczej w środowisku spedycyjnym. Zawiera tabele odnoszące się do firm spedycyjnych, ludzi zajmujących się przydzielaniem tras dla poszczególnych zamówień etc. jak także pozwala na uzyskanie danych, statystyk o kierowcach, zarobkach firm. Ponadto zawiera kilka użyteczności, które pozwalają na weryfikację wprowadzonych danych, jak także procedury automatyzujące wybór charakterystyk dla optymalnego zamówienia. Wprowadzone do bazy danych Projekt informacje obejmują:

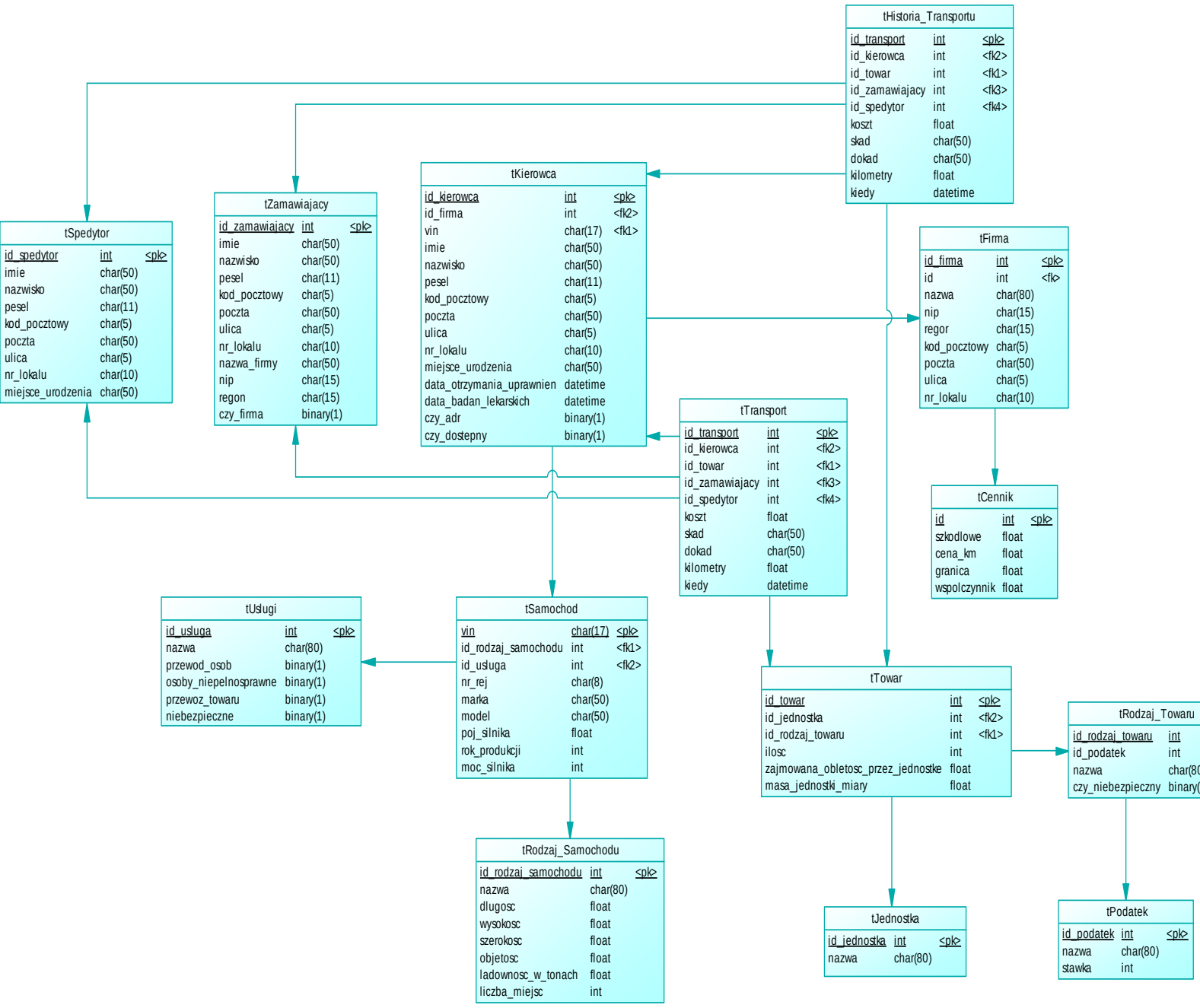
- Dane o zamawiających
- Dane o spedytorach
- Dane o kierowcach
- Dane o transporcie
- Dane firm i ich cenniki
- Samochody przydzielone poszczególnym kierowcom wraz z ich właściwościami itp.



Model konceptualny:



Model fizyczny:



WIDOKI

Widok widok_pierwszy ma za zadanie wypisać imie, nazwisko, id_kierowcy, jego wiek (w latach odczytany z numeru PESEL), kilometraż przejechany do tej pory (brany z tabel tTransport oraz tHistoria_Transportu), oraz mówić ile razy dany kierowca był w trasie. Ponadto widok informuje nas czy kierowca jest dostępny i może podjąć jakąś usługę jak również czy może przewozić towary niebezpieczne i czy ma ważne badania lekarskie (jeśli ma jeszcze je ważne mniej niż sto dni wyświetlany jest komunikat SKIERUJ PRACOWNIKA NA BADANIA).

```
use Projekt
go
ALTER VIEW widok_pierwszy
AS
SELECT DISTINCT K.imie, K.nazwisko,K.id_kierowca,
--kolumna wiek
DateDIFF(year, cast('19'+SUBSTRING(K.pesel,1,2)+'-05-06' as date), Cast(getdate() as
date)) as wiek,
SUM(kilometry) as kilometraz,
YEAR(getdate())-YEAR(K.data_otrzymania_uprawnien) as
ile_lat_uprawnienia,count(T.id_transport) as ile_razy,
[CzyDostepny] = CASE WHEN (K.czy_dostepny=0) THEN 'W TRASIE' ELSE 'GOTOWY DO PRACY'
END,
[czy moze niebezpieczne] = CASE WHEN (K.czy_adr=0) THEN 'NIE' ELSE 'TAK' END,
[CZY WAZNE BADANIA] = CASE WHEN (datediff (day,Cast(getdate() as date),
K.data_badan_lekarskich)<100) THEN 'SKIERUJ PRACOWNIKA NA BADANIA' ELSE 'WSZYSTKO OK'
END

FROM tKierowca AS K
JOIN tTransport AS T
ON T.id_kierowca=K.id_Kierowca
join tTowar AS Towar
on T.id_towar=Towar.id_towar

group by K.imie, K.nazwisko, K.id_kierowca, K.pesel, K.data_otrzymania_uprawnien,
K.czy_dostepny,K.czy_adr,K.data_badan_lekarskich


select * from widok_pierwszy
```

	imie	nazwisko	id_kierowca	wiek	kilometraz	ile_lat_uprawnienia	ile_razy	CzyDostepny	czy moze niebezpieczne	CZY WAZNE BADANIA
1	Hanna	Caban	1	32	300	11	3	GOTOWY DO PRACY	NIE	SKIERUJ PRACOWNIKA NA BADANIA
2	Aleksander	Lewicki	2	49	1878	29	17	GOTOWY DO PRACY	TAK	SKIERUJ PRACOWNIKA NA BADANIA
3	Damian	Dudkiewicz	3	42	5088	22	7	GOTOWY DO PRACY	TAK	SKIERUJ PRACOWNIKA NA BADANIA
4	Dawid	Duda	6	42	189	22	1	GOTOWY DO PRACY	TAK	WSZYSTKO OK
5	Piotr	Komorowski	7	56	4436	36	2	GOTOWY DO PRACY	TAK	WSZYSTKO OK
6	Mariusz	Zarzycki	10	43	574	23	1	GOTOWY DO PRACY	TAK	WSZYSTKO OK
7	Jeremiasz	Walezy	12	41	713	21	1	GOTOWY DO PRACY	TAK	WSZYSTKO OK
8	Hanna	Caban-mila-hej	20	31	189	4	1	GOTOWY DO PRACY	TAK	SKIERUJ PRACOWNIKA NA BADANIA

widok_drugi to statystyki firm. Wybierane są trzy najlepsze firmy, jeżeli chodzi o liczbę zarobionych pieniędzy w ciągu ostatnich dwóch lat. Ponadto wyświetlana jest nazwa firmy, jej id, adres (wybrany z kilku tabel i połączony w jeden string), średnia kilometrów na trasę oraz to czy firma przynosi zyski czy nie.

```
ALTER VIEW widok_drugi
AS
SELECT top 3 F.nazwa,F.id_firma,(F.ulica+' '+F.nr_lokalu+',
'+SUBSTRING(F.kod_pocztowy,1,2)+'-'+SUBSTRING(F.kod_pocztowy,3,3)+' '+F.poczta) AS
adres,SUM(T.kilometry) AS kilometraz, SUM(T.koszt) AS zarobione,
ROUND(AVG(T.kilometry),2) AS srednia,[CZY RENTOWNA] = CASE WHEN
(COUNT(K.id_kierowca)*3000< SUM(T.koszt)) THEN 'FIRMA JEST RENTOWNA' ELSE 'FIRMA MA
TYLY' END, count(T.id_kierowca) as [Liczba tras firmy]
from tFirma as F
join tKierowca as K
on F.id_firma=K.id_firma
inner join tTransport as T
on T.id_kierowca=K.id_kierowca
join tTowar as TOWAR
on TOWAR.id_towar=T.id_towar
WHERE ((YEAR(CAST(GETDATE() As date)) -YEAR(T.kiedy))<2)
GROUP BY F.id_firma, F.nazwa,F.poczta,F.ulica, F.kod_pocztowy, F.nr_lokalu
HAVING COUNT(T.id_towar)>1 AND SUM(T.koszt)>1000
order by zarobione desc

select * from widok_drugi
```

	nazwa	id_firma	adres	kilometraz	zarobione	srednia	CZY RENTOWNA	Liczba tras firmy
1	EUROTRANS	1	3 Maja 9 , 61-564 Abramow	6314	13700	332,32	FIRMA MA TYLY	19
2	EUROCARGO	2	Kolejowa 8 , 21-143 Warszawa	5088	17327,12	726,86	FIRMA MA TYLY	7

Widok_trzeci ma za zadanie pokazać statystyki spedytorów. Wypisuje ich imię, nazwisko, wiek odczytany z numeru PESEL, liczbę zleceń, którą wykonali a ponadto sumę tych zleceń i średnią za zlecenie.

```
ALTER VIEW widok_trzeci
AS
select imie,nazwisko,wiek,SUM(liczba_przyjetych_zlecen) AS liczba_zlecen,
SUM(suma_zlecen) AS suma,ROUND ((SUM(suma_zlecen)/ SUM(liczba_przyjetych_zlecen)),2)
AS srednia from
(
select DISTINCT S.imie, S.nazwisko,
--kolumna wiek
DateDIFF(year, cast('19'+SUBSTRING(S.pesel,1,2)+'-05-06' as date), Cast(getdate() as
date)) as wiek,

COUNT(TRANS.id_spedytor) AS liczba_przyjetych_zlecen, SUM(TRANS.koszt) AS suma_zlecen,
AVG(TRANS.koszt) AS srednia
from tTowar AS T
join tTransport AS TRANS
on TRANS.id_towar=T.id_towar
join tSpedytor       AS S
on S.id_spedytor=TRANS.id_spedytor
join tKierowca AS K
on K.id_kierowca=TRANS.id_kierowca
join tFirma AS F
on F.id_firma=K.id_firma
GROUP BY S.imie, S.nazwisko,S.pesel) jakies
GROUP BY imie,nazwisko,wiek

select * from widok_trzeci
```

	imie	nazwisko	wiek	liczba_zlecen	suma	srednia
1	Adam	Kabza	43	27	20663,6	765,32
2	Milena	Korbus	50	4	12813,52	3203,38
3	Patryk	Lewandowski	32	2	12000	6000

Widok zmaterializowany jest typem obiektu który posiada lokalnie zapisany wynik zapytania SQL na wybranej bazie (np. lokalnej lub rozproszonej zdecentralizowanej bazie danych). Ponadto obiekt ten ma asynchroniczny sposób odświeżania wyniku zapytania względem rzeczywistego wyniku na bazie macierzystej. Dzięki tym właściwościom znacząco poprawia się wydajność zapytania na tym obiekcie (w zależności od wyliczonych kosztów optymalizator może wykonać zapytanie na widoku zmaterializowanym zamiast tablicy macierzystej). W moim przykładzie wyświetlamy liczbę przejechanych km, imię oraz nazwisko kierowcy.

```
USE Projekt
GO
ALTER VIEW dbo.widokC
WITH SCHEMABINDING
AS
    SELECT SUM(T.kilometry) AS liczba_przejechanych_km,
       MAX( K.imie) AS imie , MAX(K.nazwisko) AS nazwisko, COUNT_BIG(*) AS tmp
    FROM dbo.tTransport T
        join dbo.tKierowca K
    on T.id_kierowca = K.id_kierowca
    GROUP BY K.pesel
GO
--Create an index on the view.
CREATE UNIQUE CLUSTERED INDEX IDX_V2
    ON  dbo.widokC(pesel);
GO

SELECT * FROM dbo.widokC
```

Results | Messages

	liczba_przejechanych_km	imie	nazwisko	tmp
1	300	Hanna	Caban	3
2	189	Hanna	Caban-mila-hej	1
3	189	Dawid	Duda	1
4	7893	Damian	Dudkiewicz	8
5	4436	Piotr	Komorowski	2
6	189	Andrzej	Krata	1
7	1878	Aleksander	Lewicki	17
8	713	Jeremiasz	Walezy	1
9	574	Mariusz	Zarzycki	1

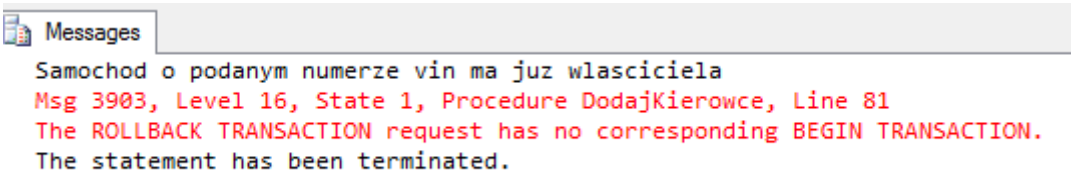
TRIGGERY

Pierwszy trigger ma za zadanie nie dopuścić do wpisania przez użytkownika do tabeli kierowców osoby, która ma ponad 67 (w mojej bazie założyłem że wiek emerytalny wynosi 67 lat). Ponadto trigger zabezpiecza przed wpisaniem osoby, która nie ma badań lekarskich lub nie ma uprawnień, albo dopiero będzie je miała. Wyzwalacz sprawdza także, czy dany samochód ma już właściciela, jeśli tak, to także nie można wpisać.

```
USE [Projekt]
GO
/****** Object:  Trigger [dbo].[DodajKierowce]    Script Date: 2016-01-20 12:08:17
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER TRIGGER [dbo].[DodajKierowce]
	on [dbo].[tKierowca]
FOR INSERT
AS
BEGIN

DECLARE @idkierowca INTEGER
	SET NOCOUNT ON;
SET @idkierowca=(SELECT MAX(id_kierowca) FROM tKierowca)

IF ((SELECT vin as samochod from tKierowca where id_kierowca=@idkierowca) IN (SELECT
vin from tSamochod))
	BEGIN
			ROLLBACK TRANSACTION
			PRINT 'Samochod o podanym numerze vin ma juz wlasciciela'
	END

IF ((SELECT DateDIFF(year, cast('19'+SUBSTRING(pesel,1,2)+'-05-06' as date),
Cast(getdate() as date))  wiek
from tKierowca where id_kierowca=@idkierowca) >67)
	BEGIN
			ROLLBACK TRANSACTION
			PRINT 'Ta osoba powinna juz byc na emeryturze'
	END

IF ((SELECT DateDIFF(year, data_otrzymania_uprawnien, Cast(getdate() as date)) as
kiedy
	from tKierowca where id_kierowca=@idkierowca) IS NULL)
	BEGIN
			ROLLBACK TRANSACTION
			PRINT 'ta osoba nie ma uprawnien'
	END
ELSE IF ((SELECT DateDIFF(year, data_otrzymania_uprawnien, Cast(getdate() as date)) as
kiedy
	from tKierowca where id_kierowca=@idkierowca)<0 )
		BEGIN
			ROLLBACK TRANSACTION
			PRINT 'ta osoba jeszcze nie ma uprawnien'
	END
```

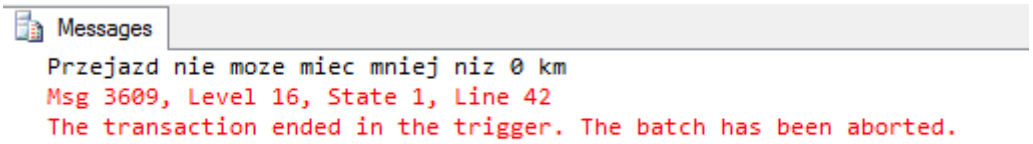
```
IF ((SELECT DateDIFF(day, data_badan_lekarskich, Cast(getdate() as date)) as badania
       from tKierowca where id_kierowca=@idkierowca) >0)
       BEGIN
                     ROLLBACK TRANSACTION
                     PRINT 'osoba nie ma badan lekarskich'
       END

END
```

```
INSERT INTO tKierowca VALUES
('WVWZZZ31ZNW000001',              '5',          'Hanna',      'CABAN'       ,
       '30051217589',       'Kalisz',             '2012-01-02', '2015-01-01', '40687',
       'Katowice',   'Sosnowa',    '5',0,1);
```



Kolejny trigger pozwala kontrolować sytuację wpisywania danych do tabeli tTransport. Nie możemy wpisać ujemnego kosztu transportu, ujemnej ilości kilometrów w transporcie, ponadto nie można wpisać trasy z ponad 20 dniowym wyprzedzeniem (ponieważ nie wiemy, czy np. kierowca nie weźmie urlopu itd.) oraz nie można wpisać trasy która już była (tak jak w sklepie nie można wydrukować w bieżącym dniu paragonu z zeszłego roku)

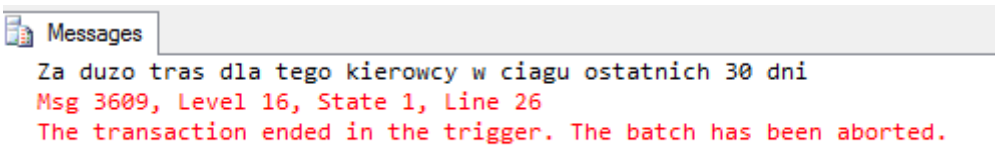
```
USE [Projekt]
GO
/****** Object:  Trigger [dbo].[DodajTransport]    Script Date: 2016-01-20 12:13:43
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[DodajTransport]
       on [dbo].[tTransport]
FOR INSERT
AS
BEGIN
DECLARE @idtransport integer
       SET NOCOUNT ON;
SET @idtransport=(SELECT MAX(id_transport) FROM tTransport)

IF ((SELECT koszt as koszt from tTransport where id_transport=@idtransport)<0)
       BEGIN
                     ROLLBACK TRANSACTION
                     PRINT 'Przejazd nie moze kosztowac mniej niz 0 zlotych'
       END

IF ((SELECT kilometry as kilometry from tTransport where id_transport=@idtransport)<0)
       BEGIN
                     ROLLBACK TRANSACTION
                     PRINT 'Przejazd nie moze miec mniej niz 0 km'
       END

IF (DATEDIFF(day,Cast(getdate() as date),(SELECT kiedy as data from tTransport where
id_transport=@idtransport))>20)
       BEGIN
                     ROLLBACK TRANSACTION
                     PRINT 'Nie mozna zaplanowac trasy z takim wyprzedzeniem'
       END
IF (DATEDIFF(day,Cast(getdate() as date),(SELECT kiedy as data from tTransport where
id_transport=@idtransport))<0)
       BEGIN
                     ROLLBACK TRANSACTION
                     PRINT 'podales zla date'
       END
END

insert into tTransport VALUES
('1','2','1','2','100','Widawa','Warszawa','-5','2016-01-22')
```



Ostatni trigger nie pozwala dopuścić do sytuacji, w której jednemu kierowcy w czasie ostatnich 30 dni przydzielimy więcej niż 3 trasy. Jest to zabezpieczenie wynikające z możliwości fizycznych człowieka, nie jesteśmy w stanie np. wykonać 60 tras w 30 dni.

```
USE [Projekt]
GO
/****** Object:  Trigger [dbo].[Kursy]    Script Date: 2016-01-20 12:19:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[Kursy]
       on [dbo].[tTransport]
FOR INSERT
AS
BEGIN
DECLARE @idkierowca integer

set @idkierowca=(SELECT id_kierowca from inserted)

       SET NOCOUNT ON;
IF ((SELECT COUNT(id_transport) as liczba from tTransport where
id_kierowca=@idkierowca AND DATEDIFF(d, kiedy, Cast(getdate() as date)) < 30) > 3)
       BEGIN
                     ROLLBACK TRANSACTION
                     PRINT 'Za duzo tras dla tego kierowcy w ciagu ostatnich 30 dni'
       END

END
```

```
insert into tTransport VALUES
('1','1','1','2','100','Widawa','Warszawa','100','2016-01-22')
```



PROCEDURY

Pierwsza procedura z kursorem, ma za zadanie przeniesienie danych z tabeli tTransport do tHistoria_Transportu o ile znalezione zostaną rekordy, dla których data jest starsza niż 50 dni.

```
USE Projekt
GO
SET ANSI_NULLS ON
GO
SET quoted_identifier ON
GO
alter PROCEDURE [dbo].[przenies]
AS
DECLARE kursor CURSOR FOR
select id_transport, id_towar,
id_kierowca,id_spedytor,id_zamawiajacy,koszt,skad,dokad,kilometry,kiedy from
tTransport
WHERE DateDIFF(day, kiedy, Cast(getdate() as date))>50

DECLARE @id_transport integer
DECLARE @id_towar integer
DECLARE @id_kierowca  integer
DECLARE @id_spedytor integer
DECLARE @id_zamawiajacy integer
DECLARE @koszt FLOAT
DECLARE @skad nvarchar(50)
DECLARE @dokad nvarchar(50)
DECLARE @kilometry FLOAT
DECLARE @kiedy DATE

PRINT 'Przenosze z tabeli tTransport do THistoria_Transportu'
SET IDENTITY_INSERT tHistoria_Transportu ON
OPEN kursor
FETCH NEXT FROM kursor INTO @id_transport, @id_towar,
@id_kierowca,@id_spedytor,@id_zamawiajacy,@koszt,@skad,@dokad,@kilometry,@kiedy
WHILE @@FETCH_STATUS = 0
       BEGIN
       SET IDENTITY_INSERT tHistoria_Transportu ON
       INSERT INTO tHistoria_Transportu (id, id_towar,
id_kierowca,id_spedytor,id_zamawiajacy,koszt,skad,dokad,kilometry,kiedy) VALUES
       (@id_transport, @id_towar,
@id_kierowca,@id_spedytor,@id_zamawiajacy,@koszt,@skad,@dokad,@kilometry,@kiedy)
       DELETE FROM tTransport WHERE id_transport=@id_transport
       FETCH NEXT FROM kursor INTO @id_transport, @id_towar,
@id_kierowca,@id_spedytor,@id_zamawiajacy,@koszt,@skad,@dokad,@kilometry,@kiedy
       SET IDENTITY_INSERT tHistoria_Transportu OFF
       END
CLOSE kursor
DEALLOCATE kursor
```

```
Przed wykonaniem procedury mamy
select * from tHistoria_Transportu
```

Results | Messages

id	id_towar	id_kierowca	id_spedytor	id_zamawiajacy	koszt	skad	dokad	kilometry	kiedy

```
exec dbo.przenies
```

Po wykonaniu procedury, otrzymujemy:

```
select * from tHistoria_Transportu
```

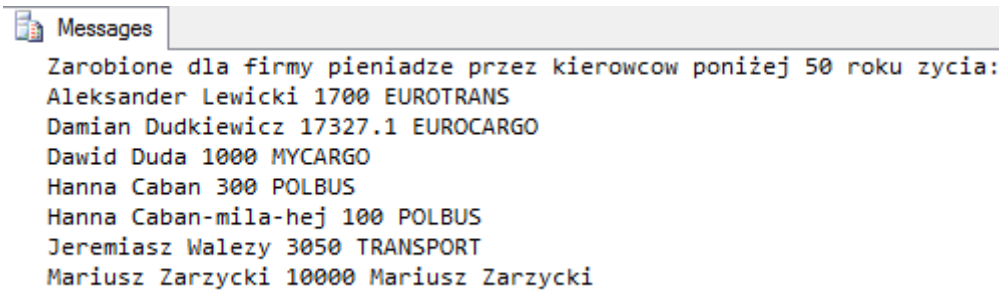
Results | Messages

	id	id_towar	id_kierowca	id_spedytor	id_zamawiajacy	koszt	skad	dokad	kilometry	kiedy
1	1	1	3	1	6	500	Widawa	Lublin	287	2015-11-01
2	2	2	6	2	7	1000	Widawa	Warszawa	189	2015-11-02
3	3	3	7	3	5	6000	Madryt	Praga	2218	2015-11-03
4	4	4	10	1	3	10000	Berlin	Warszawa	574	2015-11-04
5	5	5	12	2	2	3050	Gdynia	Zakopane	713	2015-11-05
6	6	1	3	1	6	500	Widawa	Gdynia	405	2015-05-01
7	7	2	3	2	7	1000	Widawa	Warszawa	189	2015-06-02
8	8	3	7	3	5	6000	Madryt	Praga	2218	2015-03-03

Druga procedura z kursorem, ma za zadanie zwrócić zarobione dla firmy pieniądze przez kierowców poniżej 50 roku życia.

```
USE [Projekt]
GO
/****** Object:  StoredProcedure [dbo].[zarobek]    Script Date: 2016-01-19 22:53:59
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[zarobek]
AS
DECLARE kursor CURSOR FOR
       select K.imie,K.nazwisko, SUM(T.koszt) as zarobek,  F.nazwa from
tKierowca AS K
join tTransport AS T
on T.id_kierowca=K.id_kierowca
join tFirma AS F
on K.id_firma=F.id_firma
WHERE DateDIFF(year, cast('19'+SUBSTRING(K.pesel,1,2)+'-05-06' as date),
Cast(getdate() as date))<50
GROUP BY K.imie,K.nazwisko, F.nazwa


DECLARE @imie nvarchar(50)
DECLARE @nazwisko nvarchar(50)
DECLARE @zarobek FLOAT
DECLARE @nazwafirmy nvarchar(50)
PRINT 'Zarobione dla firmy pieniadze przez kierowcow poniżej 50 roku zycia:'

OPEN kursor
FETCH NEXT FROM kursor INTO @imie, @nazwisko, @zarobek, @nazwafirmy
WHILE @@FETCH_STATUS = 0
       BEGIN
       PRINT @imie + ' '+@nazwisko + ' ' + CAST(@zarobek AS nvarchar(50)) +'
'+@nazwafirmy
      FETCH NEXT FROM kursor INTO @imie, @nazwisko, @zarobek, @nazwafirmy
   END
CLOSE kursor
DEALLOCATE kursor
```

exec dbo.zarobek



Trzecia procedura to procedura z tranzakcją. Pozwala ona na pełne zautomatyzowanie procesu wpisywania tak jakby nowego zlecenia do tabeli tTransport. Parametrami wejściowymi są

- Miasto, do którego chcemy jechać,
- Miasto, z którego chcemy wyruszać,
- Datę, kiedy będzie realizowany przewóz,
- Id_Towaru, który będzie przewożony,
- Czy będzie trzeba przekroczyć granicę (0-nie, 1-tak),
- Id_spedytora, który wykonał to dopasowanie,
- Id_zamawiającego, który złożył zapotrzebowanie.

Procedura łączy się z Google Maps Distance Matrix API, które zwraca XML z danymi, które później są wyciągane i używane. Wpisujemy dwa miasta, pomiędzy którymi będziemy chcieli zrealizować przewóz a zwracana jest odległość w kilometrach. Na tej podstawie, w zależności od przewożonego towaru, znajdowany jest samochód, który jest fizycznie w stanie przewieźć dany towar. Następnie spośród dostępnych samochodów, które spełniają kryteria tonażu, wybierany jest kierowca, dla którego firma, w której on pracuje oferuje najmniejszy koszt za przewóz tego towaru. Cena za przejazd jest odpowiednio wyliczana i zależy np. od takich parametrów jak: czy była przekraczana granica, czy przewożone były towary niebezpieczne, cena za kilometr oraz współczynnik przez który mnożony jest ostateczny koszt specyficzny dla każdej firmy. Następnie wybierany jest najmniejszy koszt, i jest on automatycznie za pomocą transakcji wpisywany do tabeli tTransport.

```
USE [Projekt]
GO
/****** Object:  StoredProcedure [dbo].[dystans]    Script Date: 2016-01-19 22:53:25
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[dystans] --exec dystans 'Widawa','Warszawa','',''…
(
@ToCity varchar(100)=null,
@FromCity varchar(100)=null,
@Data DATE,
@idTowar INTEGER,
@Granica INTEGER,
@idSpedytor nvarchar(50),
@idZamawiajacy nvarchar(50),
@DistanceInKilometers varchar(100) output,
@TimeinHours varchar(100) output
)
```

```
AS

     Declare @Object as Int;
Declare @ResponseText as Varchar(8000);
Declare @serviceUrl as varchar(500)
  set @serviceUrl = 'http://maps.googleapis.com/maps/api/distancematrix/xml?origins='
+@ToCity+ '&destinations=' +@FromCity+'&mode=driving&language=en-EN&units=metric;'
Exec sp_OACreate 'MSXML2.XMLHTTP', @Object OUT;
Exec sp_OAMethod @Object, 'open', NULL, 'get',
                @serviceUrl, --Your Web Service Url (invoked)
                 'false'
Exec sp_OAMethod @Object, 'send'
Exec sp_OAMethod @Object, 'responseText', @ResponseText OUTPUT

Declare @Response as XML

Select @ResponseText as XMLList

SET @Response = CAST(@ResponseText AS XML);

  Declare @Status as varchar(20)
   Declare @Distance as varchar(20)
   Declare @Time as varchar(20)
DECLARE @kierowca as INTEGER
DECLARE @koszt as FLOAT

  set @Status= @Response.value('(DistanceMatrixResponse/row/element/status)[1]',
'varchar(20)')
  print @Status
  if(@Status='ZERO_RESULTS')

   Begin

                    set @Distance=@Status
                                set @Time=@Status
   End
   else

   Begin
   set
@Distance=@Response.value('(DistanceMatrixResponse/row/element/distance/text)[1]',
'varchar(20)')
   set @Time=@Response.value('(DistanceMatrixResponse/row/element/duration/value)[1]',
'varchar(20)')
   End
  --Select @Response.value('(DistanceMatrixResponse/row/element/distance/text)[1]',
'varchar(10)') as Distance
  select @Distance as Odleglosc
  DECLARE @liczba_km AS INTEGER
  IF (SUBSTRING(@Distance, 2, 1)=',')
       BEGIN
  SET @liczba_km=CAST(((SUBSTRING(@Distance, 1, 1))+(SUBSTRING(@Distance, 3, 3))) as
INTEGER)
       END
  ELSE
       BEGIN
  SET @liczba_km=CAST((SUBSTRING(@Distance, 1, 4)) as INTEGER)
       END

SET @koszt=(SELECT dodatkowa.koszt
FROM
(SELECT TOP 1 MALPA.id_kierowca as id_kierowca, MALPA.KOSZT as koszt
```

```
from
(SELECT
[KOSZT]=((@liczba_km*LIPA.cena_km+@Granica*LIPA.granica+LIPA.niebezpieczne*LIPA.szkodl
iwe)*LIPA.wspolczynnik),LIPA.id_kierowca as id_kierowca--,
LIPA.cena_id,LIPA.cena_km,LIPA.granica,LIPA.niebezpieczne,LIPA.szkodliwe,LIPA.wspolczy
nnik
FROM
(SELECT RO.nazwa as nazwa, RO.ladownosc_w_tonach AS ladownosc,S.vin as
vin,K.id_kierowca as id_kierowca,K.imie as imie, K.nazwisko as nazwisko, F.id_firma as
firma_id,U.niebezpieczne as niebezpieczne ,C.id AS cena_id,C.cena_km as
cena_km,C.granica as granica,C.szkodliwe as szkodliwe,C.wspolczynnik as wspolczynnik
FROM tRodzaj_Samochodu as RO
join tSamochod AS S
on S.id_rodzaj_samochodu=RO.id_rodzaj_samochodu
join tKierowca AS K
on K.vin=S.vin
join tFirma AS F
on K.id_firma=F.id_firma
join tUslugi AS U
on U.id_usluga=S.id_usluga
join tCennik AS C
on C.id=F.id
WHERE RO.ladownosc_w_tonach>
((SELECT LOL.ladownosc from
(SELECT T.id_towar,T.nazwa,
T.ilosc,T.masa_jednostki_miary,R.czy_niebezpieczny,T.ilosc*T.masa_jednostki_miary as
ladownosc
FROM tTowar AS T
join tRodzaj_Towaru AS R
on R.id_rodzaj_towaru=T.id_rodzaj_towaru
WHERE T.id_towar=@idTowar) AS LOL)/1000)) AS LIPA
GROUP BY LIPA.id_kierowca ,LIPA.cena_km,
LIPA.granica,LIPA.niebezpieczne,LIPA.szkodliwe,LIPA.wspolczynnik) AS MALPA
GROUP BY MALPA.id_kierowca,MALPA.KOSZT
ORDER BY KOSZT) AS dodatkowa)

SET @kierowca=(SELECT dodatkowa.id_kierowca
FROM
(SELECT TOP 1 MALPA.id_kierowca as id_kierowca, MALPA.KOSZT as koszt
from
(SELECT
[KOSZT]=((@liczba_km*LIPA.cena_km+@Granica*LIPA.granica+LIPA.niebezpieczne*LIPA.szkodl
iwe)*LIPA.wspolczynnik),LIPA.id_kierowca as id_kierowca--,
LIPA.cena_id,LIPA.cena_km,LIPA.granica,LIPA.niebezpieczne,LIPA.szkodliwe,LIPA.wspolczy
nnik
FROM
(SELECT RO.nazwa as nazwa, RO.ladownosc_w_tonach AS ladownosc,S.vin as
vin,K.id_kierowca as id_kierowca,K.imie as imie, K.nazwisko as nazwisko, F.id_firma as
firma_id,U.niebezpieczne as niebezpieczne ,C.id AS cena_id,C.cena_km as
cena_km,C.granica as granica,C.szkodliwe as szkodliwe,C.wspolczynnik as wspolczynnik
FROM tRodzaj_Samochodu as RO
join tSamochod AS S
on S.id_rodzaj_samochodu=RO.id_rodzaj_samochodu
join tKierowca AS K
on K.vin=S.vin
join tFirma AS F
on K.id_firma=F.id_firma
join tUslugi AS U
on U.id_usluga=S.id_usluga
join tCennik AS C
on C.id=F.id
WHERE RO.ladownosc_w_tonach>
```

```
((SELECT LOL.ladownosc from
(SELECT T.id_towar,T.nazwa,
T.ilosc,T.masa_jednostki_miary,R.czy_niebezpieczny,T.ilosc*T.masa_jednostki_miary as
ladownosc
FROM tTowar AS T
join tRodzaj_Towaru AS R
on R.id_rodzaj_towaru=T.id_rodzaj_towaru
WHERE T.id_towar=@idTowar) AS LOL)/1000)) AS LIPA
GROUP BY LIPA.id_kierowca ,LIPA.cena_km,
LIPA.granica,LIPA.niebezpieczne,LIPA.szkodliwe,LIPA.wspolczynnik) AS MALPA
GROUP BY MALPA.id_kierowca,MALPA.KOSZT
ORDER BY KOSZT) AS dodatkowa)

  select @Time/3600 as CzasGodziny
  --select (@Time-(@Time/3600)*3600)/60 as CzasMinuty

BEGIN TRANSACTION [Tran1]

BEGIN TRY

INSERT INTO tTransport

VALUES (@idTowar,@kierowca,@idSpedytor,@idZamawiajacy,@koszt, @FromCity,
@ToCity,@liczba_km, @Data )

COMMIT TRANSACTION [Tran1]

END TRY
BEGIN CATCH
  ROLLBACK TRANSACTION [Tran1]
END CATCH


exec dystans 'Widawa','Warszawa','2016-01-22','1','1','2','5','',''
select * from tTransport
```

33	47	1	1	1	2	100	Widawa	Warszawa	100	2016-01-22
34	51	1	4	2	5	907,2	Warszawa	Widawa	189	2016-01-22

Widzimy, że wszystko ładnie zostało dodane.

(Na czas przeprowadzania doświadczenia, musiałem w triggerze odnoszącym się do ilości kursów zmienić wartość dla danego kierowcy na 20 (ponieważ tyle razy przeprowadzałem próby, że wszystko mi się wyczerpało)

```
exec dystans 'Gdynia','Madryt','2016-01-22','1','1','2','2','',''


select * from tTransport
```

33	47	1	1	1	2	100	Widawa	Warszawa	100	2016-01-22
34	51	1	4	2	5	907,2	Warsz...	Widawa	189	2016-01-22
35	63	1	3	2	2	7618,8	Madryt	Gdynia	2805	2016-01-22

FUNKCJE

Funkcja2 jest to funkcja z działu table-valued Functions, która za parametr wejściowy przyjmuje id_kierowcy. Następnie liczymy dla niego liczbę dni, która mówi nam ile czasu jest jeszcze ważne badanie lekarskie dla danego kierowcy. Funkcja zwraca tabelę z imieniem, nazwiskiem oraz liczbą dni.

```
USE [Projekt]
GO
/****** Object:  UserDefinedFunction [dbo].[funkcja2]    Script Date: 2016-01-20
13:43:06 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER FUNCTION [dbo].[funkcja2]
(@id_kierowcy INTEGER)
RETURNS @ret table
( [ID] int primary key NOT NULL,
       [IMIE] nvarchar(50) NOT NULL,
       [NAZWISKO] NVARCHAR(50) NOT NULL,
       [ILOSC_DNI] INTEGER )

BEGIN
DECLARE @pomoc DATE
DECLARE @idosoba integer
DECLARE @imie nvarchar(50)
DECLARE @nazwisko nvarchar(50)
DECLARE @liczba_dni INTEGER
SET @idosoba=@id_kierowcy
SET @imie=(SELECT imie from tKierowca where id_kierowca=@id_kierowcy)
SET @nazwisko=(SELECT nazwisko from tKierowca where id_kierowca=@id_kierowcy)
SET @pomoc=(SELECT data_badan_lekarskich from tKierowca where
id_kierowca=@id_kierowcy)
SET @liczba_dni=CAST((DATEDIFF(DD,GETDATE(),@pomoc)) AS INTEGER)

INSERT @ret
SELECT @idosoba, @imie, @nazwisko, @liczba_dni
RETURN
END
```

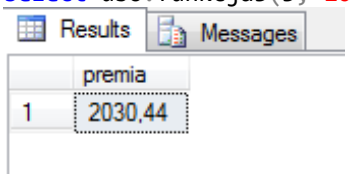
```
select * from dbo.funkcja2(3)
```

Results | Messages

	ID	IMIE	NAZWISKO	ILOSC_DNI
1	3	Damian	Dudkiewicz	43

Funkcja3 przyjmuje wartości (id_kierowcy oraz datę od której liczona jest premia, jest ona zależna od ilości tras, liczby przejechanych kilometrów itp.) i zwraca wartość premii, oznacza to że jest funkcją typu scalar-valued Function.

```
SET ANSI_NULLS ON GO SET QUOTED_IDENTIFIER ON

GO
ALTER FUNCTION [dbo].[funkcja3](
@id_kierowcy INTEGER,
@data_od DATE)
RETURNS FLOAT
BEGIN
DECLARE @pomoc FLOAT
DECLARE @pomoc_dwa FLOAT
DECLARE @pomoc_trzy FLOAT
DECLARE @adr INTEGER
DECLARE @premia FLOAT
SET @pomoc=(
select SUM(T.koszt)
from tKierowca as K
join (
select * from tTransport
union
select * from tHistoria_Transportu) AS T
on K.id_kierowca=T.id_kierowca
where T.kiedy between @data_od AND GETDATE() AND K.id_kierowca=@id_kierowcy)
SET @pomoc_dwa=(
select count(T.koszt)
from tKierowca as K
join (
select * from tTransport
union
select * from tHistoria_Transportu) AS T
on K.id_kierowca=T.id_kierowca
where T.kiedy between @data_od AND GETDATE() AND K.id_kierowca=@id_kierowcy)
SET @pomoc_trzy=(
select SUM(T.kilometry)
from tKierowca as K
join (
select * from tTransport
union
select * from tHistoria_Transportu) AS T
on K.id_kierowca=T.id_kierowca
where T.kiedy between @data_od AND GETDATE() AND K.id_kierowca=@id_kierowcy)
SET @adr=(
SELECT czy_adr from tKierowca
where id_kierowca=@id_kierowcy)
IF(@pomoc_trzy<1000)
SET @premia=@pomoc*0.05+@pomoc_dwa*50+@adr*200
ELSE
SET @premia=@pomoc*0.1+@pomoc_dwa*60+@adr*200
RETURN @premia
END
select dbo.funkcja3(3,'2015-01-01') as premia
```



Funkcja4 nie przyjmuje parametrów i zwraca imię, nazwisko oraz liczbę przejechanych kilometrów przez kierowców, którzy nie mają jeszcze 50 lat.

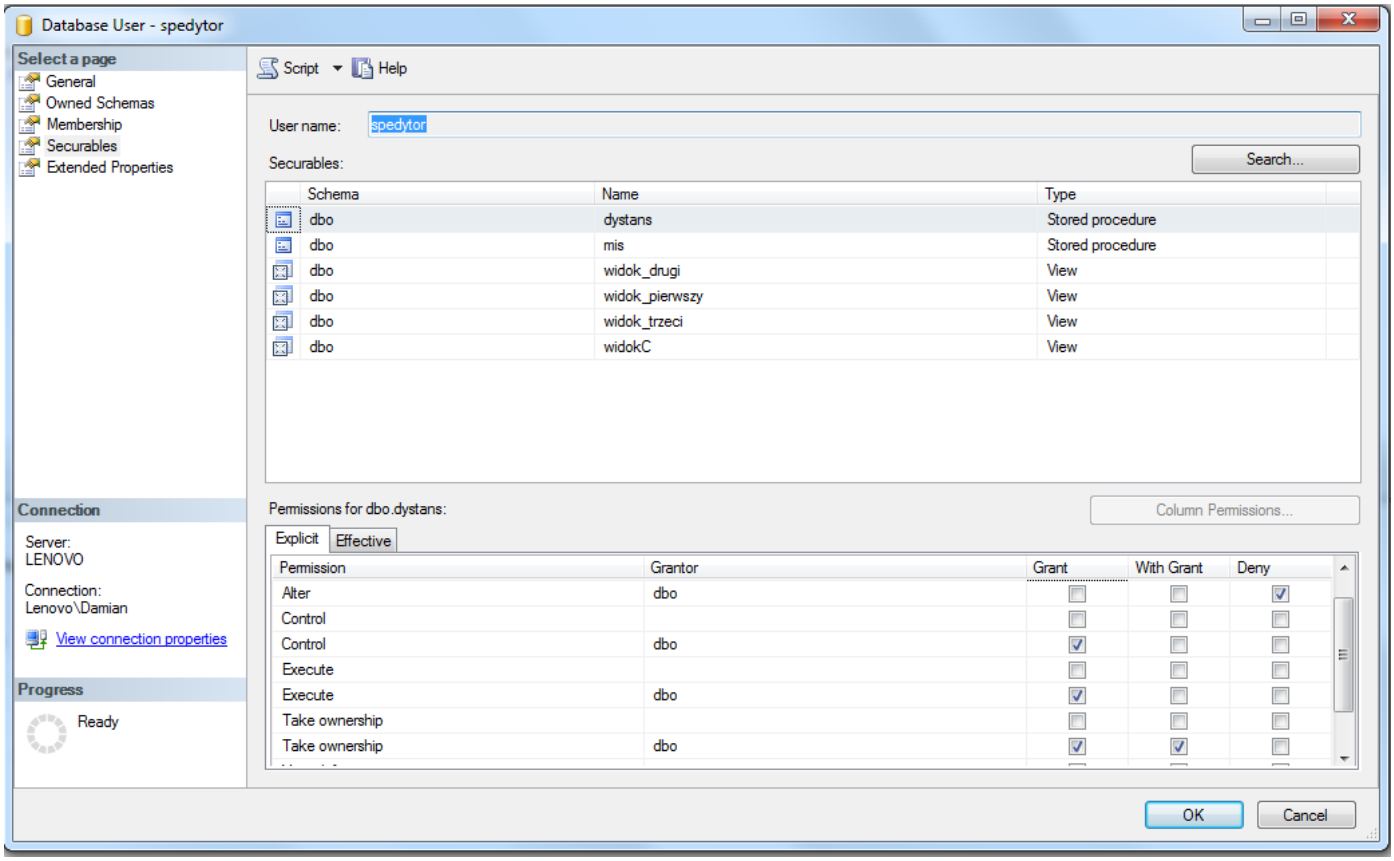
```
USE [Projekt]
GO
/****** Object:  UserDefinedFunction [dbo].[funkcja4]    Script Date: 2016-01-20
14:01:17 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER FUNCTION [dbo].[funkcja4]
()
RETURNS @ret table (
[IMIE] nvarchar(50) NOT NULL,
       [NAZWISKO] NVARCHAR(50) NOT NULL,
       [liczba] FLOAT )
BEGIN

DECLARE kursor CURSOR FOR
select K.imie, K.nazwisko,SUM(T.kilometry)
from tKierowca AS K
join(
select * from tTransport
union
select * from tHistoria_Transportu) AS T
on K.id_kierowca=T.id_kierowca
WHERE DateDIFF(year, cast('19'+SUBSTRING(K.pesel,1,2)+'-05-06' as date),
Cast(getdate() as date))<50
GROUP BY K.imie,K.nazwisko

DECLARE @imie nvarchar(50)
DECLARE @nazwisko nvarchar(50)
DECLARE @liczba FLOAT

OPEN kursor
FETCH NEXT FROM kursor INTO @imie, @nazwisko, @liczba
WHILE @@FETCH_STATUS = 0
       BEGIN
       INSERT @ret
       SELECT @imie, @nazwisko, @liczba
       FETCH NEXT FROM kursor INTO @imie, @nazwisko, @liczba
       END
CLOSE kursor
DEALLOCATE kursor
RETURN
END
select * from dbo.funkcja4()
```

Results | Messages

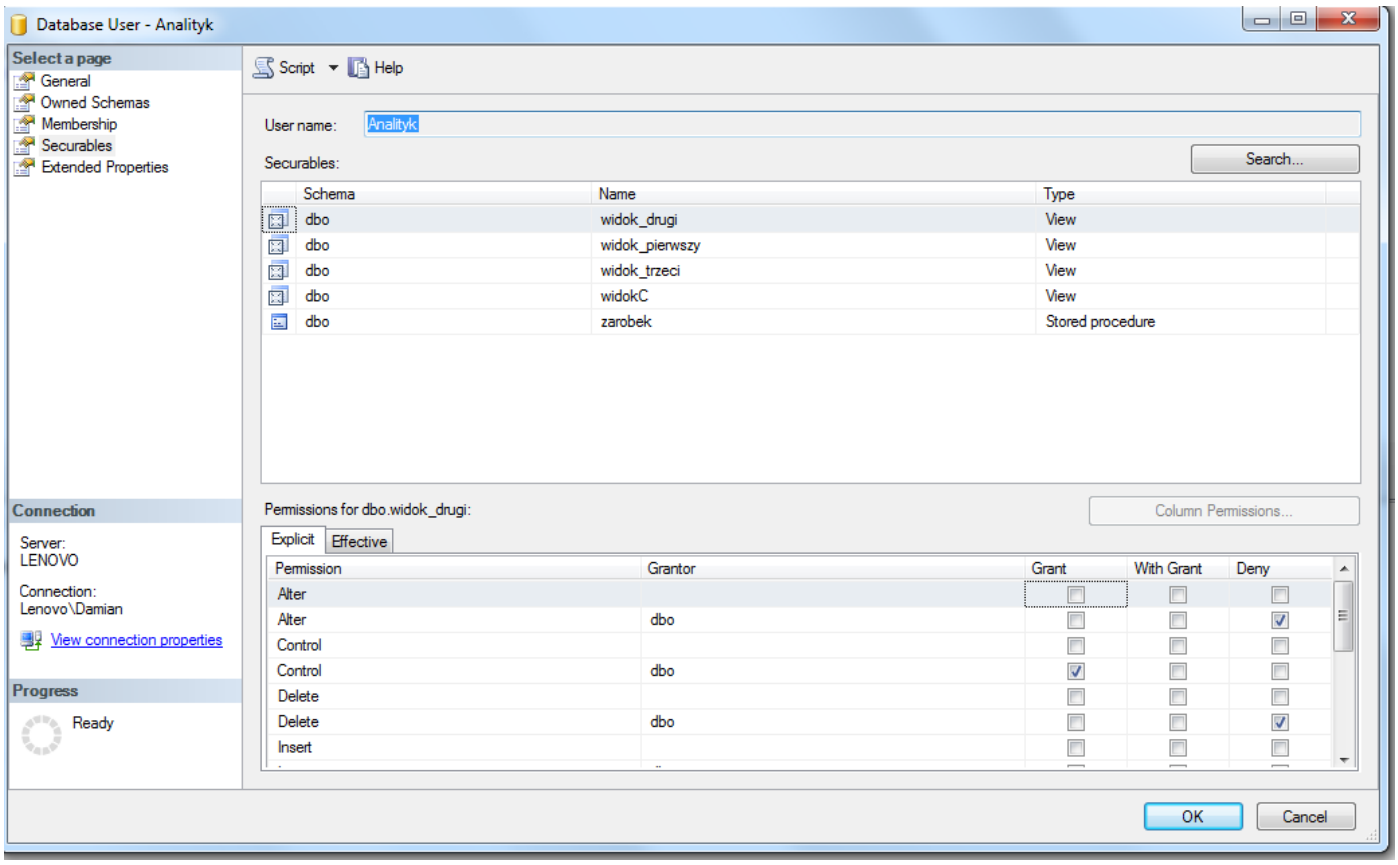
	IMIE	NAZWISKO	liczba
1	Hanna	Caban	300
2	Hanna	Caban-mila-hej	189
3	Dawid	Duda	189
4	Damian	Dudkiewicz	7893
5	Andrzej	Krata	189
6	Aleksander	Lewicki	1878
7	Jeremiasz	Walezy	713
8	Mariusz	Zarzycki	574

TRZECH UŻYTKOWNIKÓW

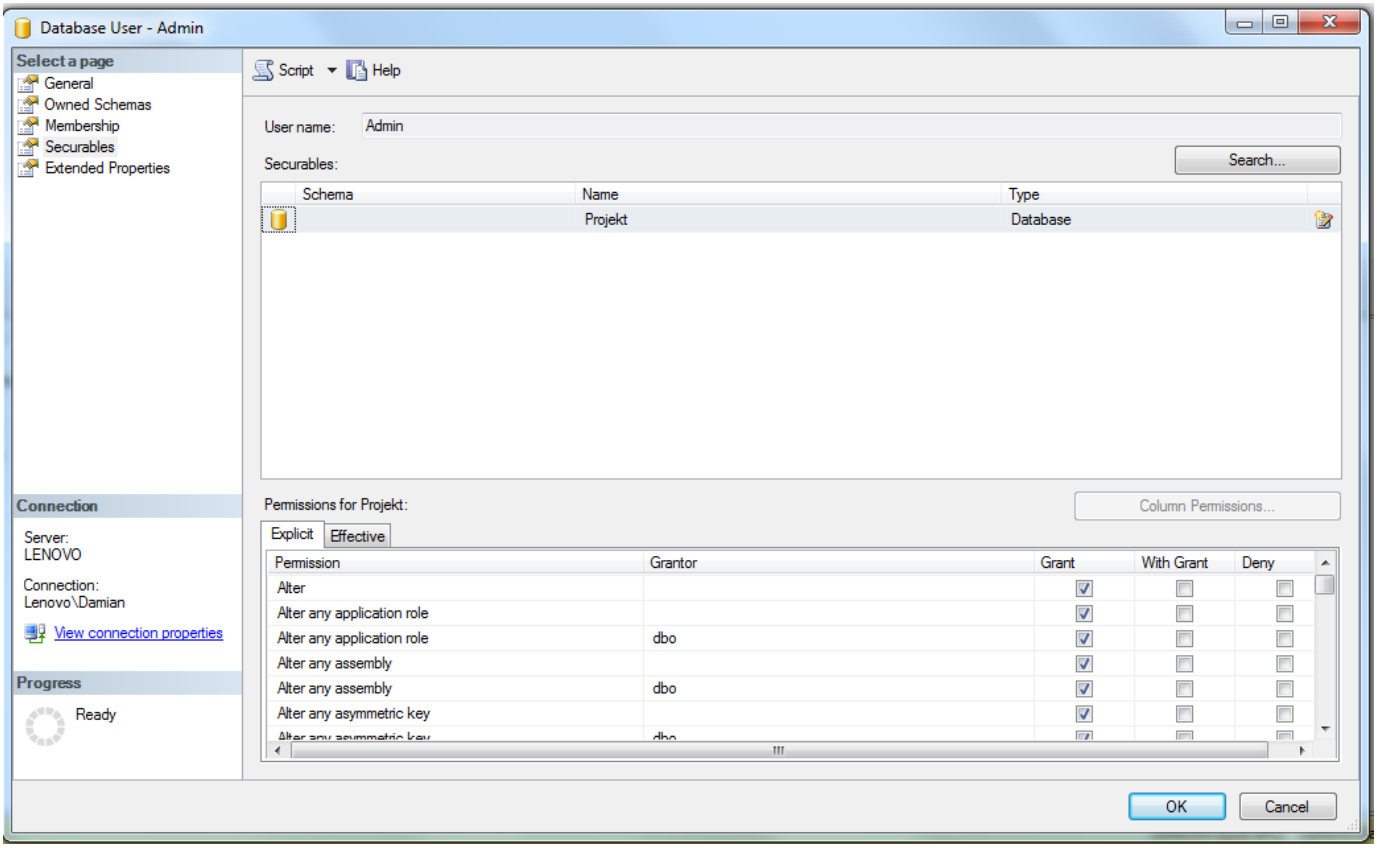
Spedytor



Analityk



Administrator



UŻYCIE SEQUENCERÓW

W mojej bazie danych można w TSQL wprowadzić nową tabelę na dwa sposoby:

```
CREATE TABLE tOleje2
(id int IDENTITY(1,1) NOT NULL,
PRIMARY KEY(id),
nazwa NVARCHAR(50) NOT NULL,
);
```

Albo z użyciem licznika

```
CREATE SEQUENCE licznik1
       START WITH 1
    INCREMENT BY 1 ;
GO

CREATE TABLE tOleje1
(id INTEGER DEFAULT next value for licznik1 NOT NULL,
PRIMARY KEY (id),
nazwa NVARCHAR(50) NOT NULL,
);
```