Projekt P3

Hurtownia danych - Bank Danych Lokalnych - opis działania i konfiguracji

Autorzy:
Jakub Porębski
Mateusz Bruchnicki
Maciej Zamorski

Kraków 2020

Wstęp

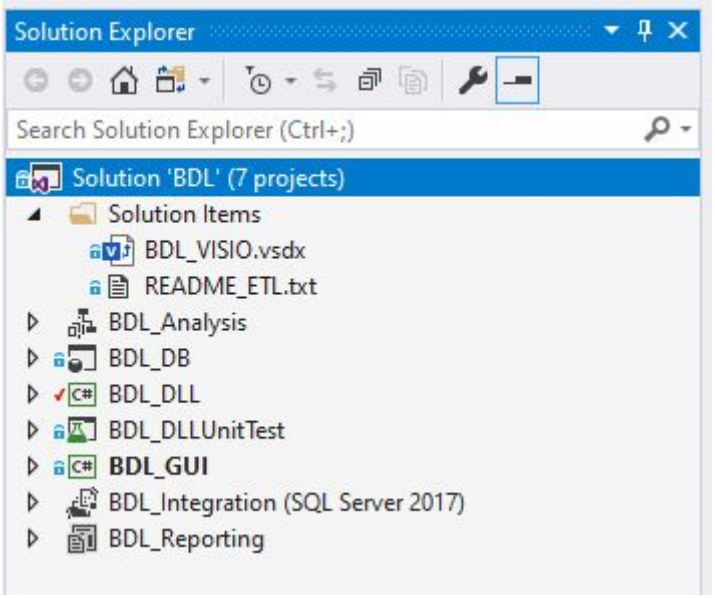
Przyświecającą nam ideą jest zaprezentowanie możliwości i funkcjonalności SQL Server w zakresie pobierania, przechowywania i raportowania danych lokalnie na serwerze, danych pobranych z Banku Danych Lokalnych.
Merytorycznym celem było stworzenie systemu raportowego dla Ministerstwa Transportu i Infrastruktury. Głównym zamierzeniem projektu jest zatem wygenerowanie raportu dla Ministra Transportu i Infrastruktury Rzeczpospolitej Polskiej. Przedstawimy aktualną sytuację z zakresu transportu i łączności oraz inne dane powiązane z tym tematem. Będą prezentowane ciekawe statystyki, wskaźniki oraz sugestie dla ministra, które pomogą mu podjąć określone działania.
Raport ten ma wywołać w ministrze impuls do podjęcia działań w najpilniejszych tematach.

Architektura projektu

Projekt zbudowany jest wielowarstwowo. Technologicznie rozwiązano to poprzez utworzenie solucji Visual Studio zawierającej rozmaite projekty:

1. Projekt DLL, jak również projekt testów jednostkowych do biblioteki
2. Projekt bazodanowy
3. Projekt SSIS (Integration Services)
4. Projekt kostek (SSAS - Analysis Services)
5. Projekt Reporting Services

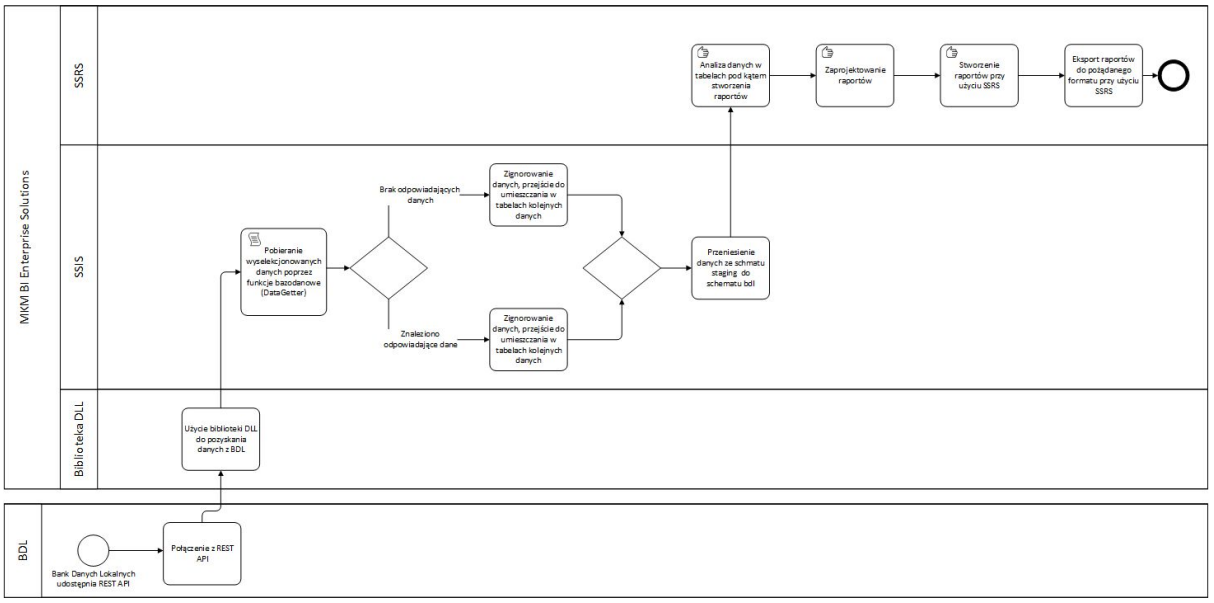
Całość zawarta jest w jednym repozytorium GIT, co czyni solucję bardzo przenośną. Widok na organizację wszelkich projektów w solucji przedstawia obrazek poniżej.



Ogólny schemat działania

Do pobierania danych służy biblioteka BDL.dll.
Dane pobierane są w trakcie procesu SSAS.
Przetwarzanie danych następuje w Analysis Services za pomocą tzw. kostek.
Wyświetlamy dane w formie raportów korzystając z Reporting Services.

Rysunek poniżej przedstawia ogólny proces.



Źródła danych

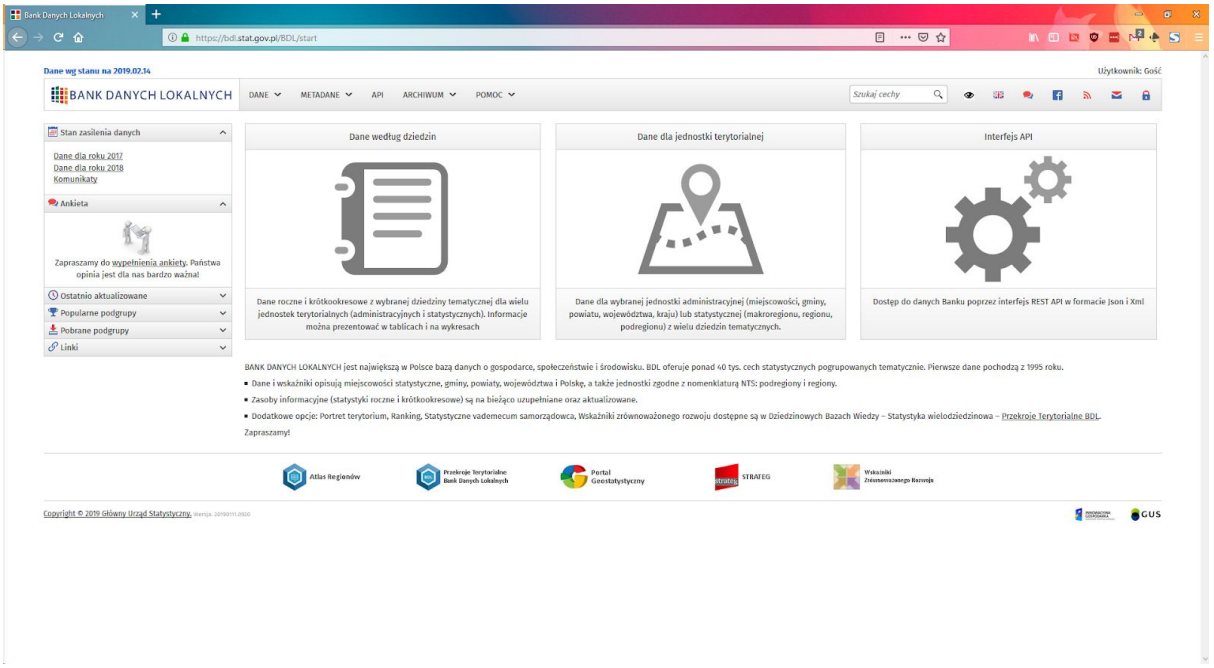
Źródłem danych jest Bank Danych Lokalnych[1]. Jest to polski projekt Głównego Urzędu Statystycznego. Udostępnionych jest kilkadziesiąt tysięcy wskaźników i powiązane z nimi dane. W szczególności dane z grupy “Transport i łączność”:

1. Drogi publiczne
2. Drogi publiczne gminne / powiatowe
3. Działalność transportowa
4. Pojazdy
5. Komunikacja miejska
6. Linie komunikacji autobusowej - transport zbiorowy
7. Wypadki drogowe
8. Przestępczość drogowa

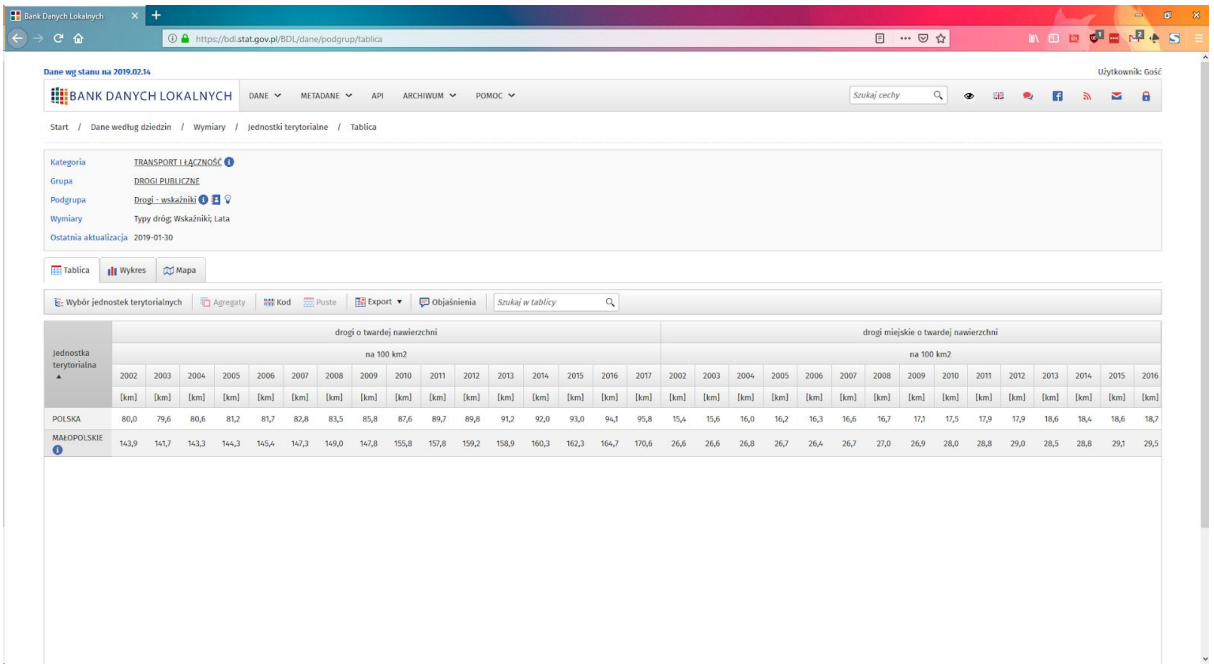
[1] https://bdl.stat.gov.pl

Przykładowe dane

Bank Danych Lokalnych udostępnia aplikację internetową. Widok na stronę główną Banku Danych Lokalnych i aplikacji:



Widok na podgląd przykładowych interesujących nas danych:



Takie właśnie dane chcemy zaimportować lokalnie do hurtowni danych, przetworzyć je, a następnie zaprezentować w formie raportu.

Opis biblioteki DLL

Ogólne założenia

System bazodanowy SQL serwer pozwala na dołączanie do baz tzw. assembly – wtyczek, bibliotek pisanych na platformę CLR (Common Language Runtime). Mogą być to biblioteki pisane w języku C#, ale również np. Visual Basic bądź też F#.

Niniejsza biblioteka działa właśnie jako assembly. Językiem biblioteki jest C#, użyte weń mechanizmy i składnia języka wymagają korzystania z .net Framework w wersji najwyższej (na czas pisania jest to 4.7).

Metoda wykorzystania takiej wtyczki pozwoli na elastyczne i łatwe połączenie SQL Server z serwisem REST Banku Danych Lokalnych. Biblioteka realizując zadanie łączenia się poprzez udostępnione API działa niezależnie od serwera SQL. Separacja odpowiedzialności jest tu bardzo ważna – każdy fragment projektu wykonuje to co potrafi najlepiej. SQL Server odpowiada za sprawne zarządzanie danymi, natomiast w kwestii połączeń z serwisami REST biblioteki .net są bezkonkurencyjne - w porównaniu z wbudowanymi w SQL Server mechanizmami.

Metoda ta jest również odporna na zmiany. W przypadku zmiany sposobu łączenia się z API Banku Danych Lokalnych wystarczające będzie odpowiednie dostosowanie niniejszej biblioteki, natomiast mechanizmy po stronie SQL Server pozostaną bez zmian.

Diagram klas

Poniższy ogólny diagram przedstawia klasy reprezentujące encje - rekordy.

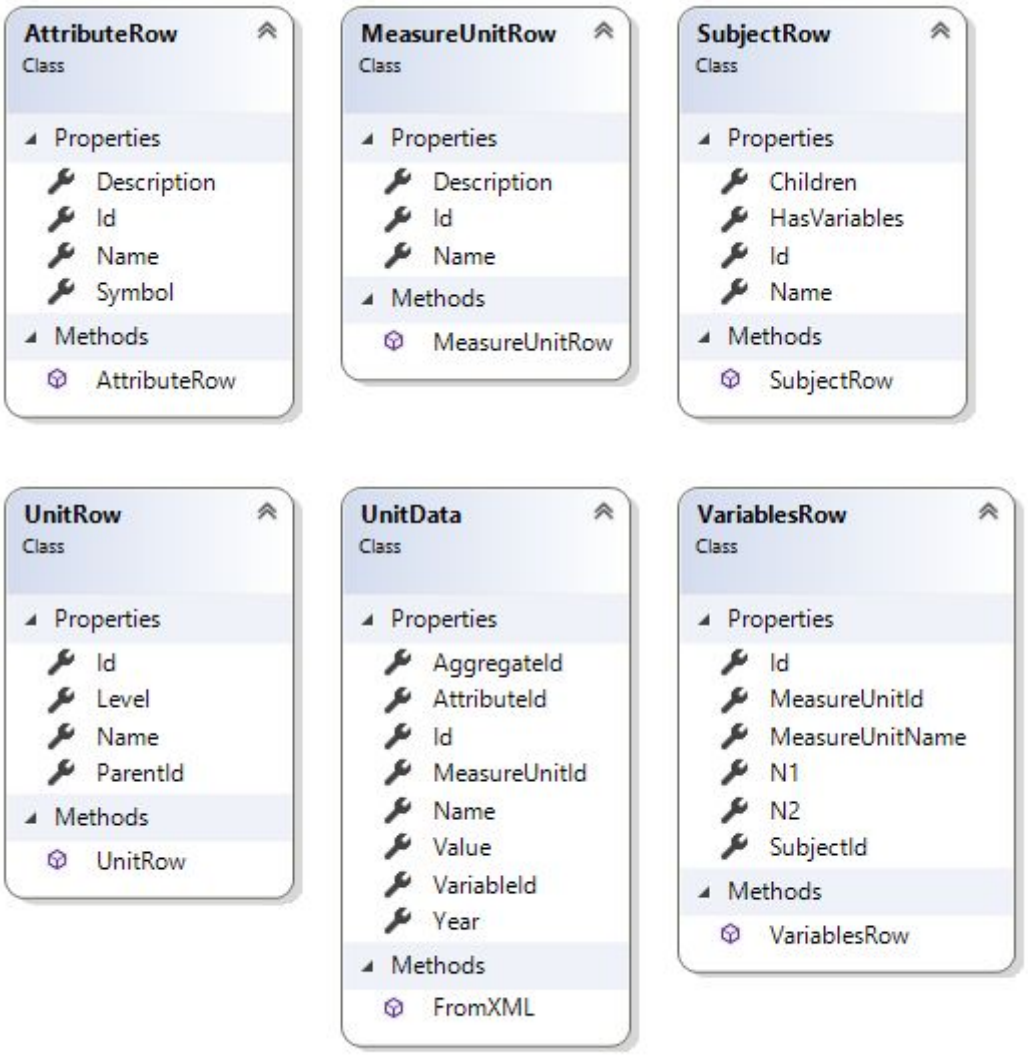
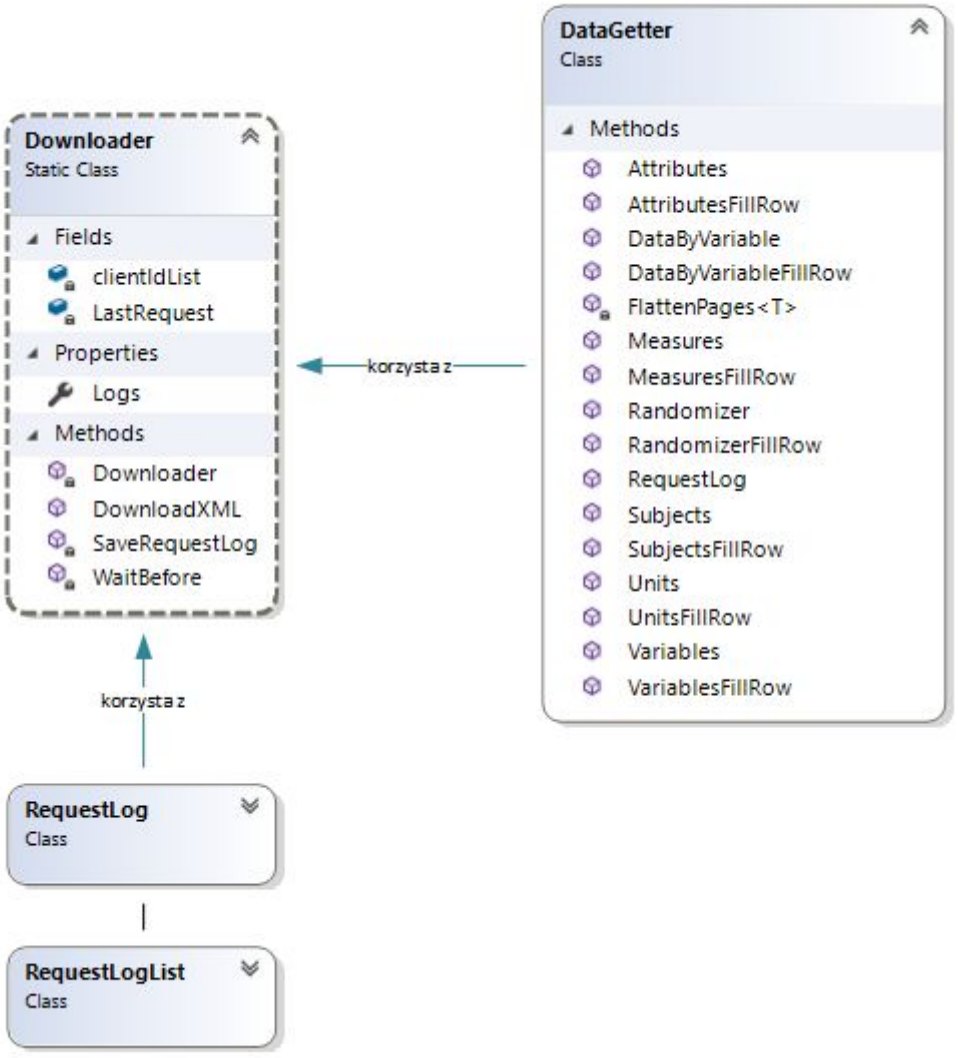


Diagram przedstawiający klasy pomocnicze oraz główną klasę wykorzystywaną po stronie SQL Server obrazuje kolejny rysunek poniżej.



Opis klas encji

Są to klasy reprezentujące pojedyncze encje bazodanowe. Każda instancja klasy reprezentuje jeden wiersz, który zwraca funkcja tabelaryczna bazodanowa. Są to proste obiekty DTO (data transfer object). Rola tych obiektów sprowadza się konwersji danych z odpowiedniej metody API REST Banku Danych Lokalnych i przechowywania tych danych w pamięci RAM na okres ich przetwarzania. Dane zapisane w klasie służą więc jako pośredniczące – odbierając z serwisu BDL i przekazując je do SQL Servera.

AttributeRow

Klasa reprezentuje atrybut. Atrybutem w sensie BDL jest opis specyficznych sytuacji w danych, które są powiązane z każdą wartością liczbową.

Właściwość	Typ	Opis
Id	Int	Unikalny identyfikator
Name	String	Nazwa atrybutu
Symbol	String	Symbol atrybutu
Description	String	Opis atrybutu

MeasureUnitRow

Klasa reprezentuje jednostkę miary. Zwykle jest to jednostka ze standardowego układu jednostek SI, ale poza nimi występują również jednostki walutowe (zł) lub opisujące liczby (promil).

Właściwość	Typ	Opis
Id	Int	Unikalny identyfikator
Name	String	Nazwa jednostki miary
Description	String	Opis

SubjectRow

Klasa reprezentuje ogólnie pojęty temat. Temat opisuje zgrubnie zbiór danych reprezentowanych – jest to tzw. metadana. Tematy są hierarchiczne, tj. dzielą się na nadrzędne oraz podrzędne. Każdy temat najwyższego poziomu zawiera listę tematów niższego poziomu. Tematy charakteryzują się również tym, że mogą posiadać zmienne – te które mogą posiadać zmienne są tematami „liśćmi” – nie mają tematów podrzędnych, ale można pobierać zmienne, których temat dotyczy – w przeciwieństwie do tematów, które „liścmi” nie są.

Właściwość	Typ	Opis
Id	Int	Unikalny identyfikator
Name	String	Nazwa tematu

Children	String	Lista tematów podrzędnych (oddzielona przecinkami)
HasVariables	Boolean	Czy temat posiada zmienne

UnitRow

Klasa reprezentuję jednostkę terytorialną. Przykładem jednostki jest województwo, gmina, powiat, miejscowość, itp. Jednostki naturalnie układają się w strukturę hierarchiczną – województwa, następnie powiaty, gminy, itd.

Właściwość	Typ	Opis
Id	Int	Unikalny identyfikator
Name	String	Nazwa jednostki (nazwa województwa, powiatu, itd.)
ParentId	String	Jednostka terytorialna nadrzędna
Level	Int	Poziom zagnieżdżenia jednostki

VariablesRow

Klasa reprezentująca zmienną – w sensie Banku Danych Lokalnych. Definicja zmiennej[1]

wielowymiarowe cechy reprezentujące określone zjawisko, z określonym obowiązywaniem w latach i na konkretnym poziomie jednostek, np. liczba pracujących dla wieku '20-26' i płci 'mężczyźni'

Każda zmienna przynależy do jakiegoś tematu. Dany temat może zawierać wiele zmiennych.

Właściwość	Typ	Opis
Id	Int	Unikalny identyfikator
SubjectId	String	Id tematu do którego przynależy zmienna
N1	String	Nazwa ogólna zmiennej
N2	String	Szczegóły zmiennej (np. dodatkowa dookreślająca nazwa)
MeasureUnitId	Int	Id jednostki miary
MeasureUnitName	String	Nazwa jednostki miary

UnitData

Klasa reprezentująca pojedynczy wiersz konkretnych już danych. W terminologii hurtownii danych można poprzez analogię rozumieć tę klasę jako reprezentującą pojedynczy wiersz z tabeli faktów. Mamy zatem odniesienia do wszystkich wyżej wymienionych klas, jak i również konkretne wartości danych. Klasa przyjmuje dowolne wartości z Banku Danych Lokalnych.

Właściwość	Typ	Opis
Id	Int	Unikalny identyfikator
Name	String	Nazwa
Year	String	Rok w którym występowała dana
Value	string	Wartość
VariableId	int	Id zmiennej
MeasureUnitId	int	Id jednostki miary
AggregateId	int	Id poziomu agregowania danych
AttributeId	int	Id atrybutu

Opis klas reprezentujących logikę biznesową

Poniżej przedstawiono klasy narzędziowe i klasy realizujące logikę biznesową.

Downloader

Klasa statyczna której celem jest połączenie się, z użyciem protokołu HTTPS, do serwisu REST Banku Danych Lokalnych. Wymaganiem takiego połączenia jest podanie tokena uwierzytelniającego (w nagłówku HTTP), który klasa dostarcza z listy tokenów. Ze względu na limity połączeń dla pojedynczego tokena, klasa posiada listę kilku tokenów, którymi się uwierzytelniamy (korzystamy tu z klasy NeverendingList). Dodatkowo, ze względu na ograniczenia serwisu API, przed każdym żądaniem HTTP czekamy określony czas – nie przekroczymy w ten sposób limitów.

RequestLog, RequestLogList

Klasy przechowujące informacje o wykonanych do tej pory żądaniach REST do serwisu BDL. Obsługują serializację i deserializację z pliku XML zapisywanego lokalnie na dysku serwera. Dzięki temu mamy możliwość śledzenia zapytań nawet i po restarcie SQL Servera.

DataGetter

Centralna klasa, która udostępniana jest na serwerze. Metody tej klasy, które są widoczne po stronie bazy danych opatrzone są adnotacją SqlFunction. Większość z tych funkcji działa po stronie bazodanowej jako funkcja tabelaryczna zwracająca określoną tabelę z jasno określonymi polami.

Lista metod niniejszej klasy:

1. RequestLog. Metoda zwraca listę zapisanych żądań do API Banku Danych Lokalnych.

Po stronie SQL Server widoczna jako procedura, która pokazuje listę w postaci kolejnych wierszy w zakładce Messages.

2. Subjects. Metoda zwraca listę tematów – w zależności od parametru. Jeśli parametr jest null, to metoda zwraca tematy nadrzędne w formie listy encji SubjectRow.

3. Variables. Metoda zwraca listę zmiennych dla danego tematu w postaci listy encji VariablesRow.

4. Measures. Metoda zwraca listę wszelkich jednostek miar używanych w BDL; postać listy encji MeasureUnitRow.

5. Attributes. Metoda zwraca listę wszelkich atrybutów w formie listy encji AttributeRow.

6. DataByVariable. Metoda zwraca listę konkretnych już danych – dla podanego id zmiennej, zakresu lat i zasięgu „w głąb” (poziomu zagnieżdżenia wg jednostek terytorialnych). Forma listy encji UnitData.

Listy encji odczytywane są przez SQL Server jako kolejne wiersze tabeli, którą funkcja zwraca.

Inne klasy

Wykorzystuję własnoręcznie napisaną klasę NeverendingList, która reprezentuje interfejs „niekończącej się listy”. Pobierając kolejny element listy wewnętrzny wskaźnik przesuwany jest o jeden do przodu. Gdy dojdziemy do końca listy, wskaźnik przesuwany jest na początek. Wykorzystanie tej klasy znajduje się w klasie Downloader.

Działanie mechanizmów

Klasa DataGetter korzysta z metod klasy Downloader. Klasa Downloader korzysta z kolei z klasy RequestLog, która zależy od RequestLogList. Mamy weń również klasę NeverendingList.

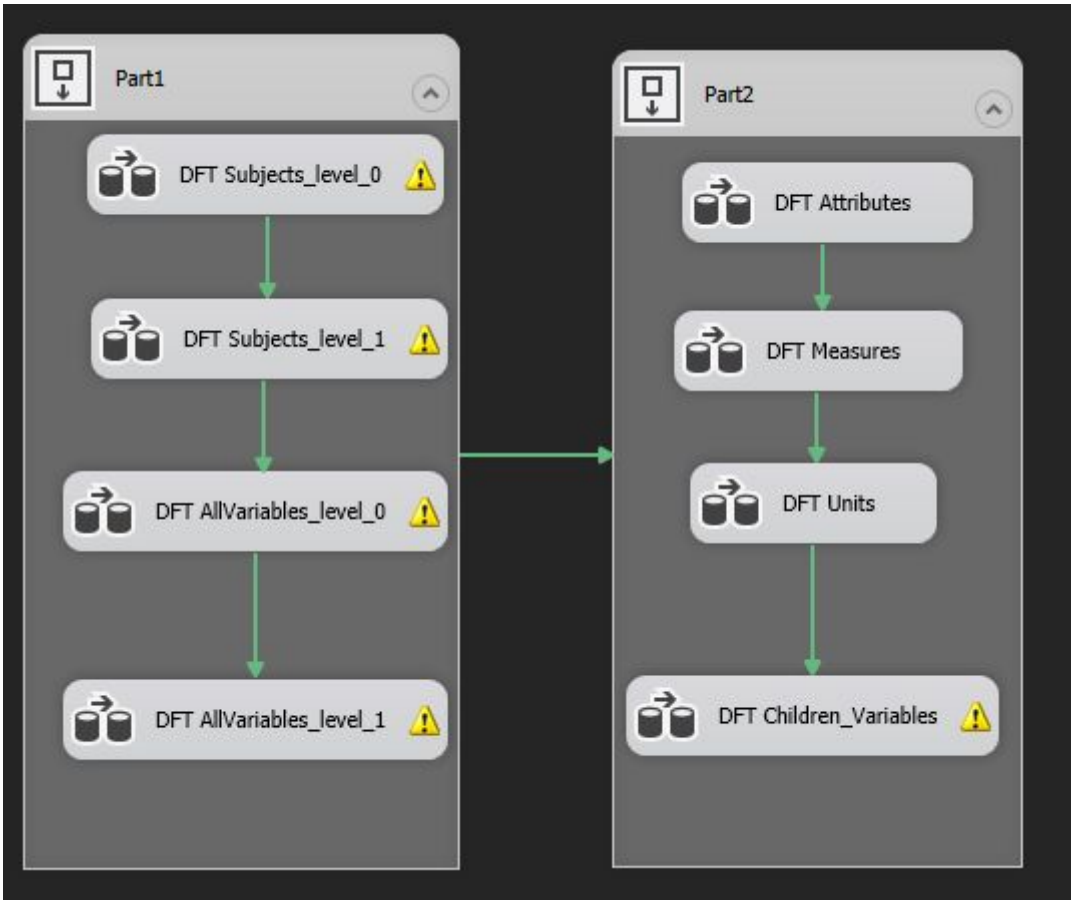
Przykłady wywołań

Przykłady wywołań klas znajdują się w projekcie testów jednostkowych.

Wyposażeni w wiedzę na temat głównej biblioteki DLL naszego projektu możemy przejść dalej do opisu kolejnych ważnych mechanizmów.

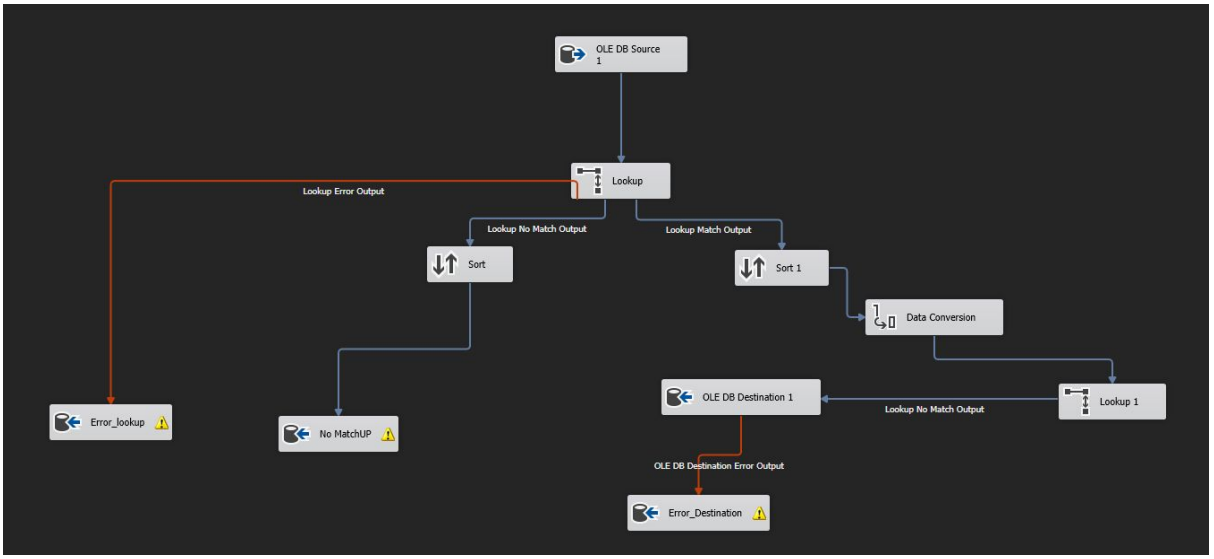
Proces transferu danych w SSIS

Pierwsza paczka pobiera po kolei wszystkie potrzebne dane, czyli tematy, podtematy, zmienne, atrybuty, miary i jednostki.



1. **Pobranie tematów.**
   Wykonywany jest skrypt
   “select *
   from dbo.Subjects(null, 100)”

   Proces ETL:



Pobrane dane trafiają do tabeli [DimTopSubjects] w schemacie [BDL].

Definicja tabeli:

```
CREATE TABLE [bdl].[DimTopSubjects](
    [Id] [nvarchar](10) NOT NULL,
    [Name] [nvarchar](255) NULL,
    [HasVariables] [bit] NULL,
    [Children] [nvarchar](1024) NULL,
    [CODE] [nvarchar](255) NULL,
 CONSTRAINT [PK_TOPSUBJECTS] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

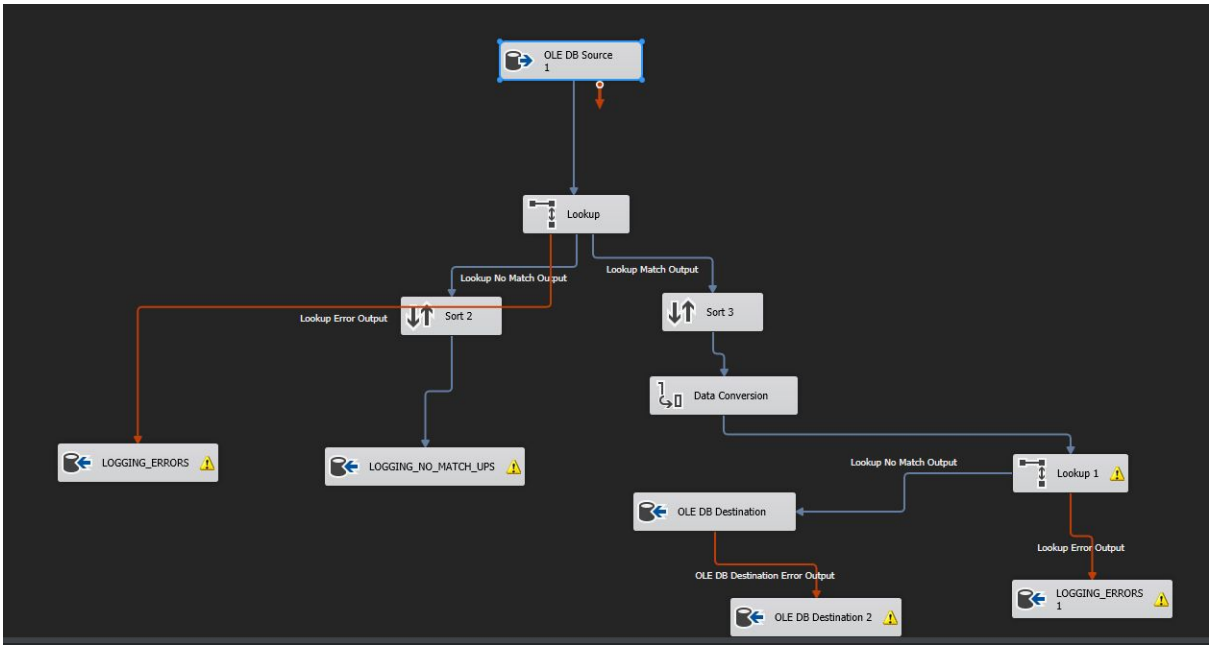
2. **Pobranie podtematów**

Wykonywany skrypt:

“with cte as
(
   select Code as ParentID from bdl.DimTopSubjects cross apply string_split(Children, ',') child
)
   SELECT
         distinct *, REPLACE(PARENTID,'K','') AS PARENT_CODE
    from cte cross apply dbo.Subjects(ParentID,100)”

REPLACE został użyty, aby kolumna z założonym PK była typu INTEGER.

Proces ETL:



Dane trafiają do tabeli DimSubjects w schemacie [BDL].

Definicja tabeli:

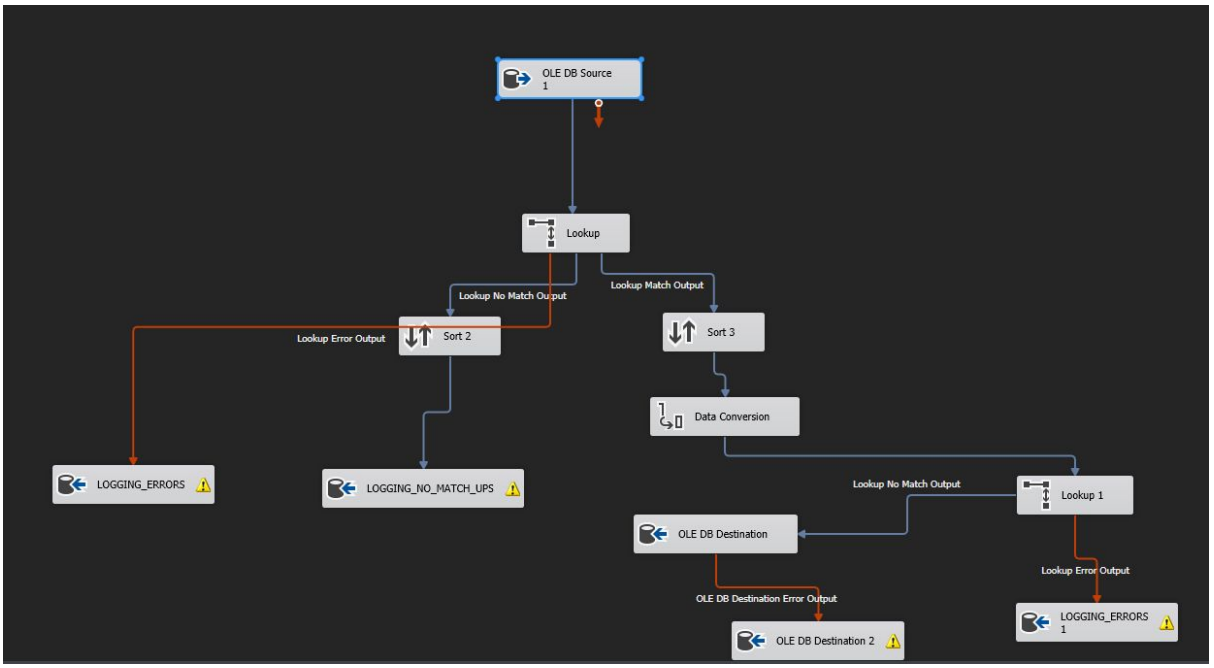
```
CREATE TABLE [bdl].[DimSubjects](
    [parentId] [nvarchar](10) NOT NULL,
    [Id] [nvarchar](10) NOT NULL,
    [Name] [nvarchar](255) NULL,
    [HasVariables] [bit] NULL,
    [Children] [nvarchar](1024) NULL,
    [CODE] [nvarchar](255) NULL,
 CONSTRAINT [PK_SUBJECTS] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [bdl].[DimSubjects] ADD CONSTRAINT [SUBJECT_HAS_TOP] FOREIGN KEY([parentId])
REFERENCES [bdl].[DimTopSubjects] ([Id])
GO

ALTER TABLE [bdl].[DimSubjects] CHECK CONSTRAINT [SUBJECT_HAS_TOP]
GO
```

3.Pobranie zmiennych nadrzędnych

Wykonywany skrypt:

“with cte as
(
    select Code as ParentID from bdl.DimSubjects cross apply string_split(Children, ',') child
)
    SELECT
            distinct *, REPLACE(PARENTID,'G','') AS PARENT_CODE
     from cte cross apply dbo.Subjects(ParentID,100)”

Proces ETL:



Dane trafiają do tabeli DimVariables w schemacie [BDL].
Definicja tabeli:

```
CREATE TABLE [bdl].[DimVariables](
    [parentId] [nvarchar](10) NOT NULL,
    [Id] [nvarchar](10) NOT NULL,
    [Name] [nvarchar](255) NULL,
    [HasVariables] [bit] NULL,
    [CODE] [nvarchar](255) NULL,
 CONSTRAINT [PK_VARIABLES] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [bdl].[DimVariables] ADD  CONSTRAINT [VARIABLE_HAS_TOP] FOREIGN KEY([parentId])
REFERENCES [bdl].[DimSubjects] ([Id])
GO

ALTER TABLE [bdl].[DimVariables] CHECK CONSTRAINT [VARIABLE_HAS_TOP]
GO
```

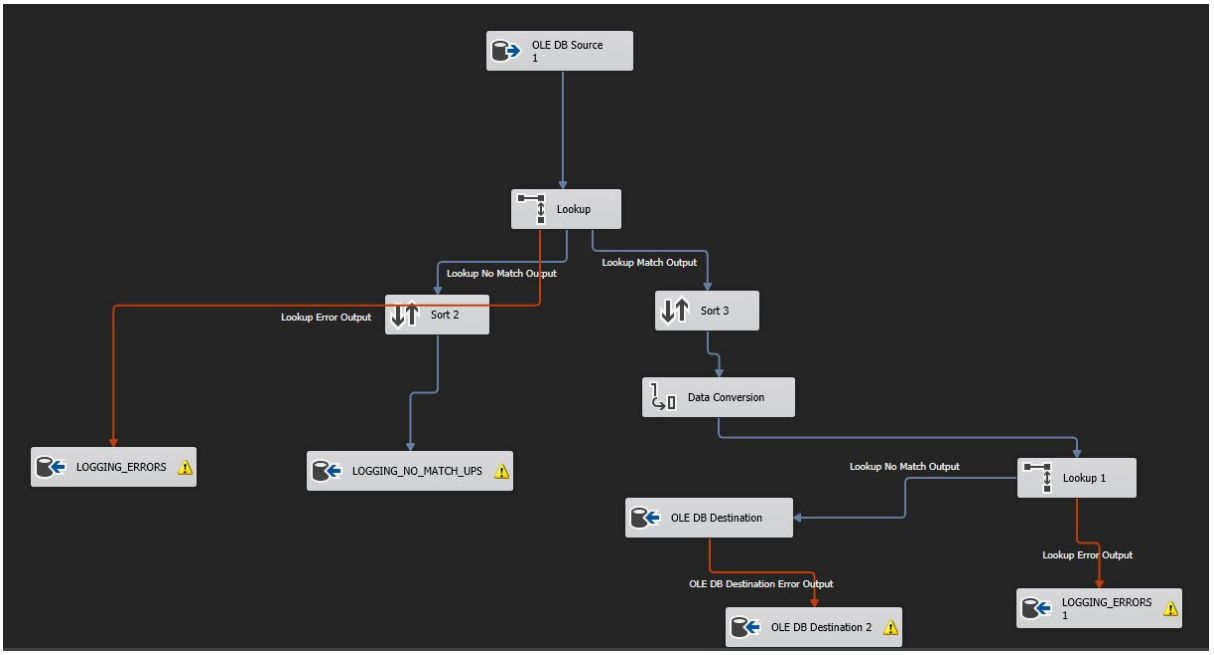
4. Pobranie zmiennych podrzędnych

Wykonywany skrypt:

“with cte as
(
  select Code as ParentID from bdl.DimSubjects cross apply string_split(Children, ',') child
  where value in ('P1636','P1720','P1722','P1723','P1724','P1729','P1733','P1736','P1746','P1747','P1748','P2418','P2420','P2655','P2825','P3298','P3434','P3548','P3573','P3583','P3584')
)
  SELECT
        distinct *, REPLACE(PARENTID,'G','') AS PARENT_CODE
  from cte cross apply dbo.Subjects(ParentID,100)”

Pobiera on tylko zmienne dotyczące transportu drogowego.

Proces ETL:



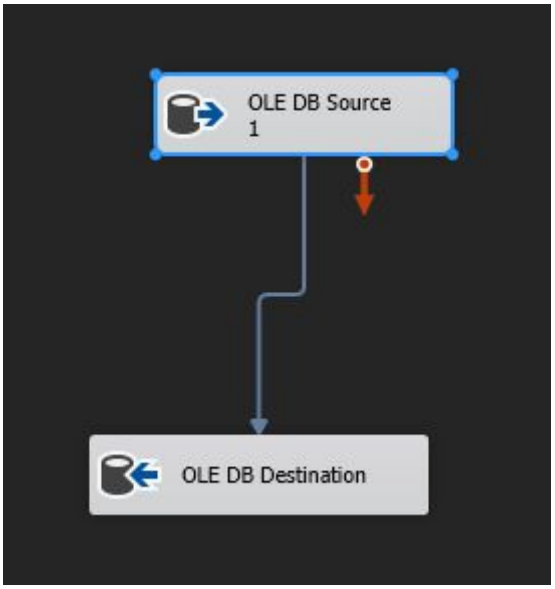
Dane trafiają do tej samej tabeli co zmienne nadrzędne.

5. Pobranie atrybutów

Wykonywany skrypt:

“select a.* from dbo.Attributes() a where a.id not in (select id from bdl.DimAttributes)”

Proces ETL:



Dane trafiają do tabeli DimAttributes w schemacie [BDL].

Definicja tabeli:

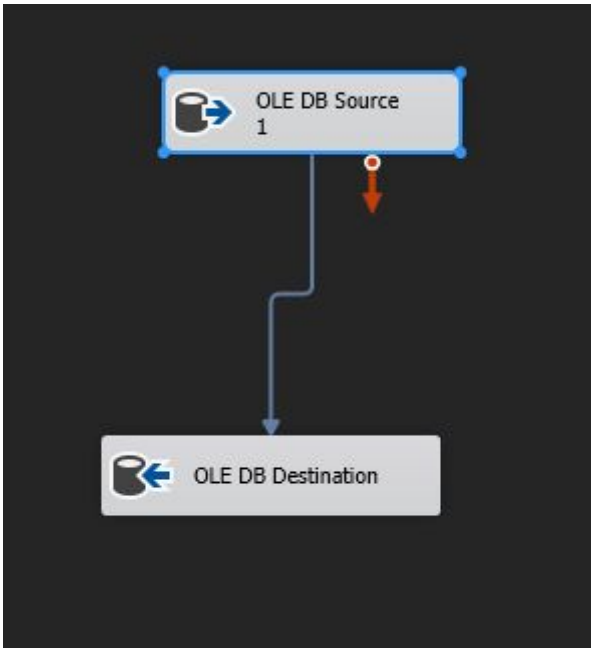
```
CREATE TABLE bdl.DimAttributes
(
    [Id] [int],
    [Name] [nvarchar](255),
    [Symbol] [nvarchar](255) NULL,
    [Description] [nvarchar](255) NULL,
CONSTRAINT [PK_ATTRIBUTES] PRIMARY KEY CLUSTERED
(
    [Id] ASC
))
```

6. Pobranie miar

Wykonywany skrypt:

“select a.* from dbo.Measures() a where a.id not in (select id from bdl.DimMeasures)”

Proces ETL:



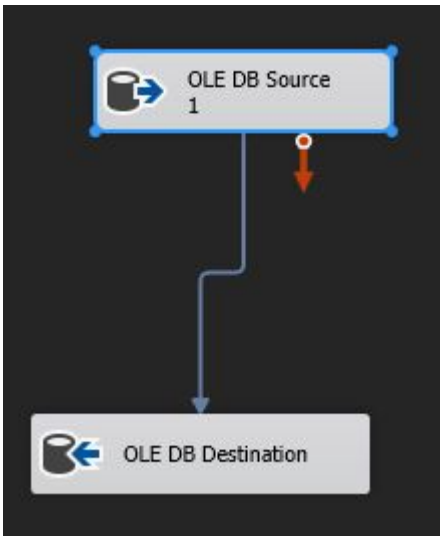
Dane trafiają do tabeli DimMeasures w schemacie [BDL].

Definicja tabeli:

```
CREATE TABLE [bdl].[DimMeasures]
(
    [Id] [int],
    [Name] [nvarchar](255),
    [Description] [nvarchar](255) NULL,
CONSTRAINT PK_MEASURES PRIMARY KEY CLUSTERED
(
ID ASC
))
```

7. Pobranie Jednostek
Wykonywany skrypt:
“select a.* from dbo.Units(100) a where a.id not in (select id from bdl.DimUnits)”

Proces ETL:



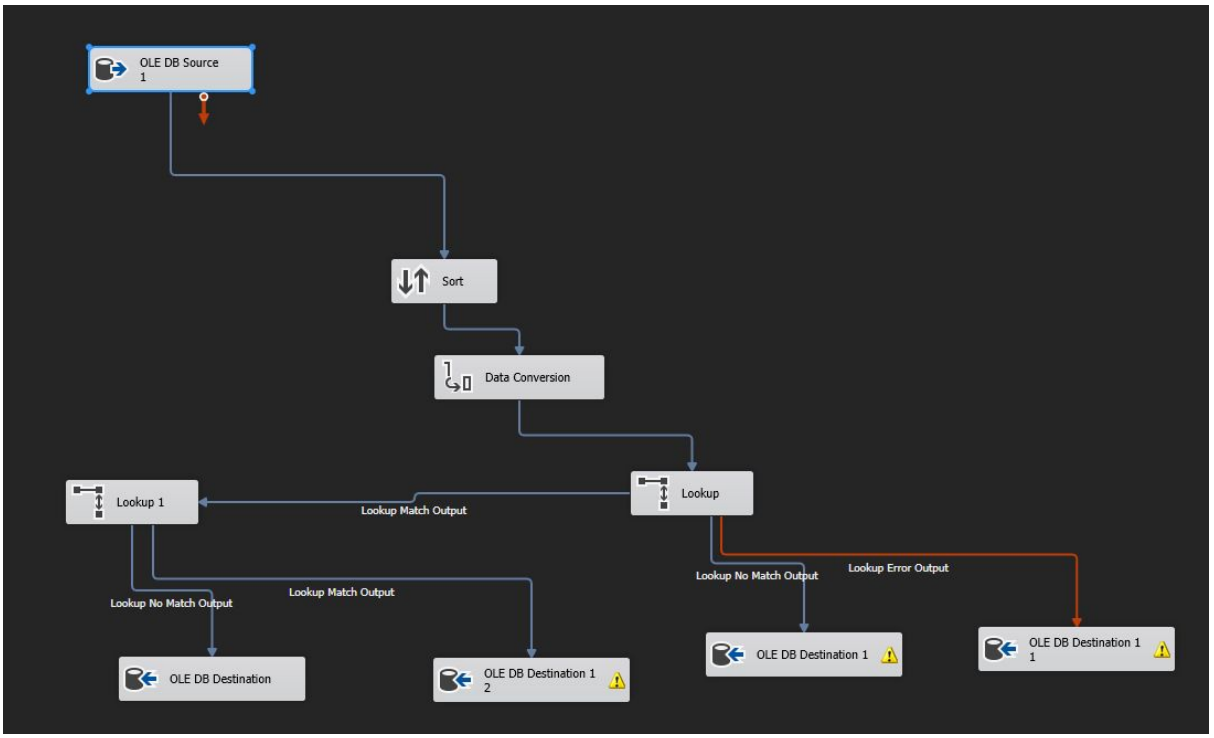
Dane trafiają do tabeli DimUnits w schemacie [BDL].

Definicja tabeli:

```
create table [bdl].[DimUnits]
(
    [Id] [nvarchar](30) PRIMARY KEY NOT NULL,
    [Name] [nvarchar](255) NULL,
    [ParentId] [nvarchar](30) NULL,
    [Level] [int] NULL,
)
```

8. Pobranie najniższego poziomu zmiennych:
“SELECT
distinct av.code,
REPLACE(dbo.variables.SUBJECTID,'P','') AS PARENTID
, dbo.variables.ID
, dbo.variables.N1
, dbo.variables.N2
, dbo.variables.MEASUREUNITID
, dbo.variables.MEASUREUNITNAME
FROM bdl.DimVariables av CROSS APPLY DBO.VARIABLES(av.CODE,100)
WHERE PARENTID IN (208, 209, 213, 239)”

Proces ETL:

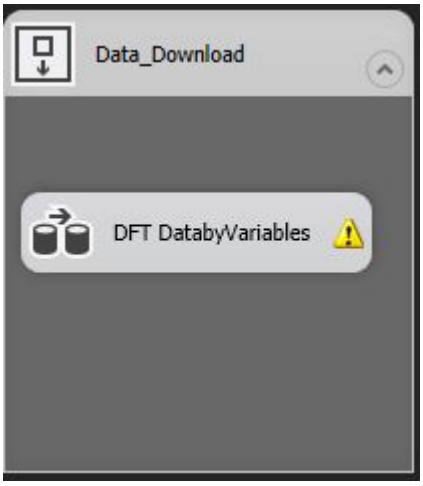


Dane trafiają do tabeli Variables_Dimms w schemacie [STAGING].

Definicja tabeli:

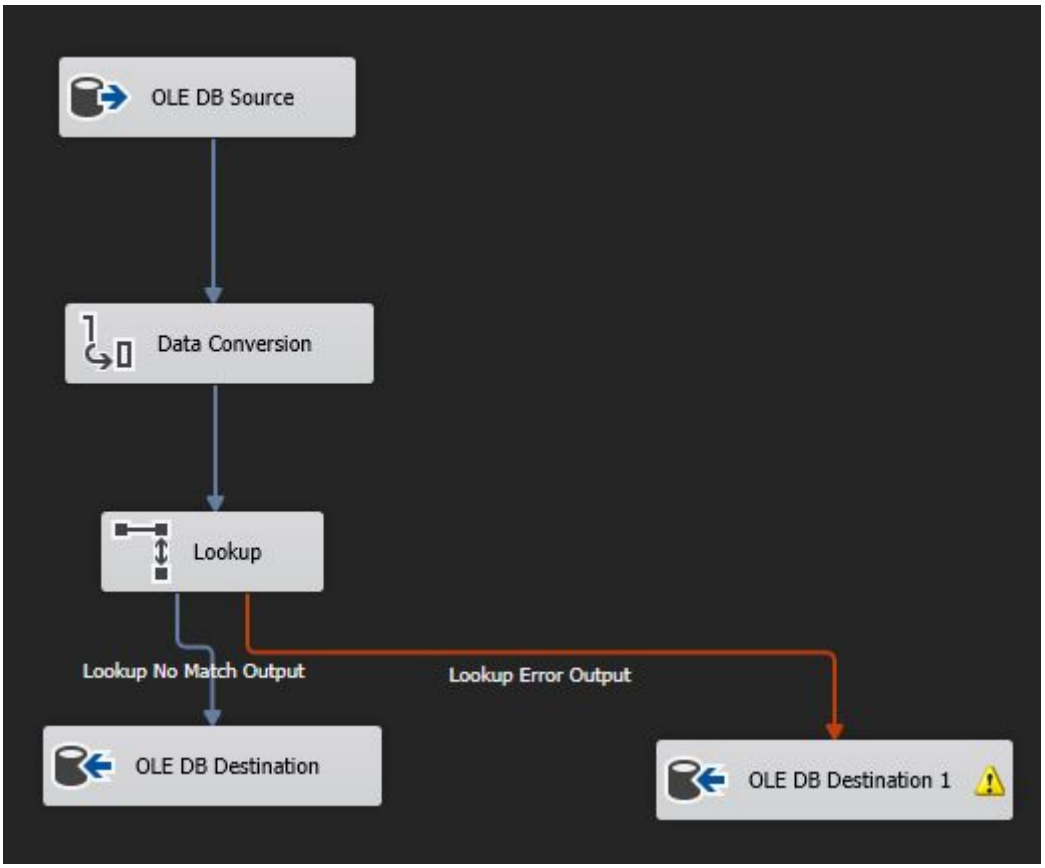
```
CREATE TABLE [staging].[VARIABLES_DIMMS](
    [parentId] [nvarchar](10) NOT NULL,
    [Id] [nvarchar](10) NOT NULL,
    [N1] [nvarchar](255) NULL,
    [N2] [NVARCHAR](255) NULL,
    [MEASUREUNITID] [INT],
 CONSTRAINT [PK_VARIABLES_DIMMS] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
) ON [PRIMARY]
GO

ALTER TABLE [staging].[VARIABLES_DIMMS]  ADD  CONSTRAINT [DIMENSION_HAS_VARIABLE] FOREIGN KEY([parentId])
REFERENCES [bdl].[DimVariables] ([Id])
GO

ALTER TABLE [staging].[VARIABLES_DIMMS] CHECK CONSTRAINT [DIMENSION_HAS_VARIABLE]
GO

ALTER TABLE [staging].[VARIABLES_DIMMS]  ADD  CONSTRAINT [VARIABLE_HAS_MEASURE] FOREIGN KEY([MEASUREUNITID])
REFERENCES [bdl].[DimMeasures] ([Id])
GO
```

Tutaj następuje pobranie danych z REST API do schematu TEMP.
Paczka:



9. Pobranie danych do tabeli Data_tmp.
Wykonywany skrypt:
“select
    zmienna.Id as VariableId
,   zmienna.MeasureUnitId
,   AggregateId
,   dane_dla_zmiennej.Id as UnitId
,   N1
,   N2
,   Name
,   Year
,   Value
,   AttributeId
,   REPLACE(zmienna.parentId,'P','') AS PARENTID
from staging.VARIABLES_DIMMS as zmienna
    cross apply dbo.DataByVariable(zmienna.Id, null, 2015, 2019, 2, 100) as dane_dla_zmiennej”

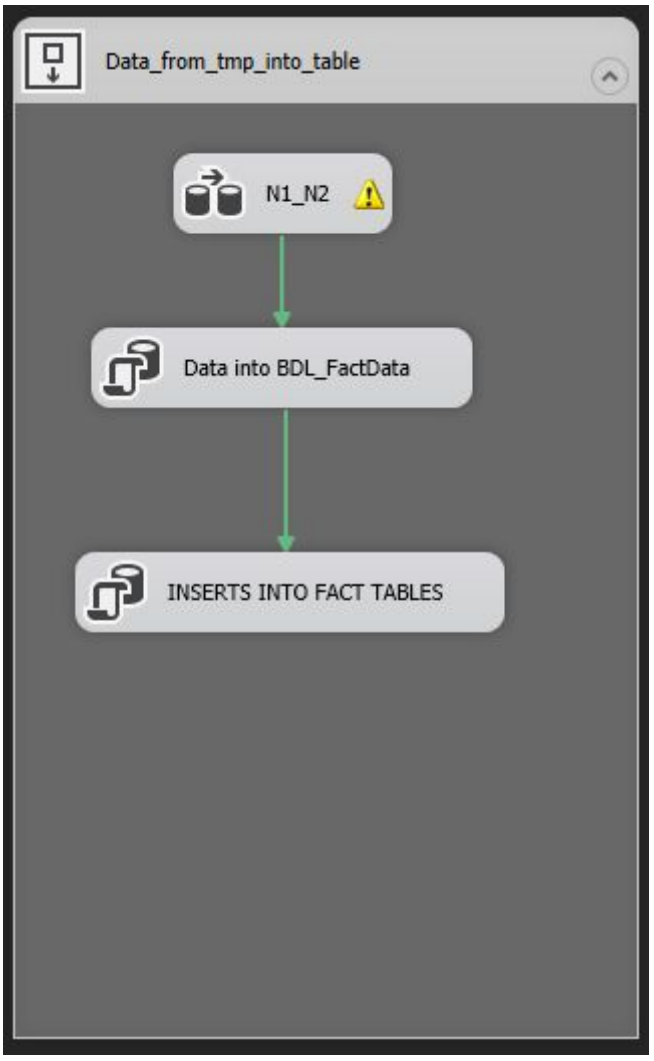
Proces ETL:



Dane trafiają do tabeli Data_tmp w schemacie [TEMP].

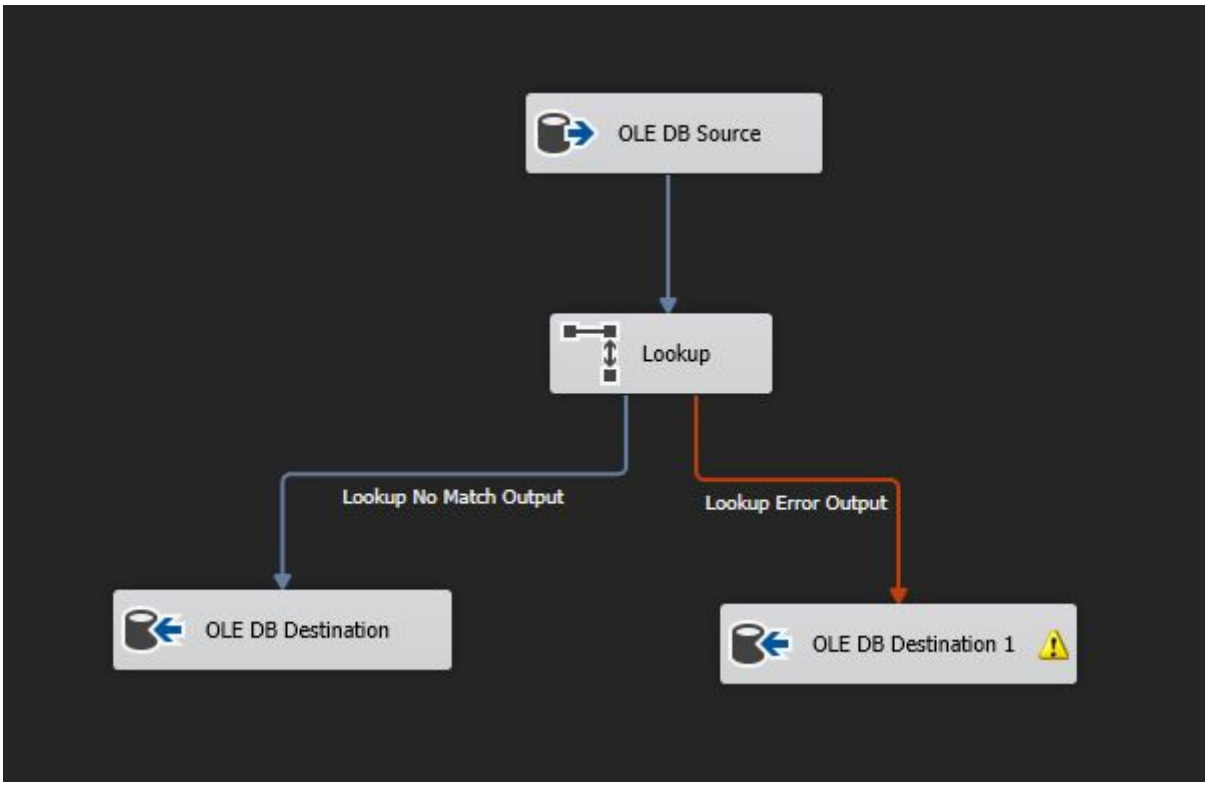
Paczka pobierająca dane z API jest uruchamiana raz na dobę za pomocą joba bazodanowego “Download_data_tmp”.

W tym momencie nastepuje przeniesienie danych z tabeli Data_tmp, a następnie za pośrednictwem schematu [BDL] zasilenie docelowej tabeli faktów. Wypełnione zostają również wymiary.
Paczka:



10.Przeniesienie danych z tabeli Data_tmp.

Proces ETL:



Dane trafiają do tabeli TemDataFromVariable w schemacie [TEMP].

Definicja tabeli:

```
CREATE TABLE [temp].[TempDataByVariable]
(
    [VariableId] [nvarchar](10) NULL,
    [SubjectId] [nvarchar](10) NULL,
    [N1] [nvarchar](255) NULL,
    [N2] [nvarchar](255) NULL,
    [MeasureUnitId] [int] NULL,
    [MeasureUnitName] [nvarchar](30) NULL,
    [AggregateId] [int] NULL,
    [TerritorialUnitId] [nvarchar](100) NULL,
    [Name] [nvarchar](255) NULL,
    [Year] [int] NULL,
    [Value] [nvarchar](255) NULL,
    [AttributeId] [int] NULL
)
```

11. Zasilenie tabeli FactData w schemacie [BDL]
Uruchamiany jest skrypt:
“INSERT INTO bdl.FactData
(ID, MEASUREUNITID, AGGREGATEID,UNITID,N1,N2,
NAME,YEAR,VALUE,ATTRIBUTEID,parentId)
SELECT VariableId, MeasureUnitId, AggregateId, UNITID, N1, N2, Name,
[Year], [Value], AttributeId, parentId
FROM [temp].Data_tmp where VariableID not in (select ID from bdl.FactData)”

Definicja tabeli:

```
CREATE TABLE [bdl].[FactData](
    [Id] INT NOT NULL,
    [parentId] [nvarchar](10) NOT NULL,
    [MEASUREUNITID] [INT],
    [UNITID] [NVARCHAR](30) NULL,
    [N1] [NVARCHAR](255),
    [N2] [NVARCHAR](255),
    [NAME] [NVARCHAR](255) NULL,
    [YEAR] [INT],
    [VALUE][NVARCHAR](255) NULL,
    [ATTRIBUTEID] [INT],
    [AGGREGATEID] [INT]
    )
```

12. Zasilenie tabel schematu [FACT].
Uruchamiana jest procedura składowana
“EXEC [DBO].[INSERTS]”

```
CREATE PROCEDURE [dbo].[INSERTS]
AS
SET NOCOUNT ON
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
EXEC DIMATTRIBUTES_IN
EXEC DIMMEASURES_IN
EXEC DIMTOPSUBJECTS_IN
EXEC DIMSUBJECTS_IN
EXEC DIMUNITS_IN
EXEC DIMVARIABLES_IN
EXEC CATEGORY_IN
EXEC SUBCATEGORY_IN
EXEC FACTDATA_IN

RETURN (@@ERROR)
```

Kolejno uruchamiane funkcje:

```
CREATE PROCEDURE DIMATTRIBUTES_IN
AS BEGIN
INSERT INTO FACT.DimAttributes
 SELECT [Id]
       ,[Name]
       ,[Symbol]
       ,[Description]
 FROM [bdl].[DimAttributes] A
 WHERE A.ID NOT IN (SELECT ID FROM FACT.DimAttributes)
END
```

```
CREATE PROCEDURE [dbo].[DIMMEASURES_IN]
AS BEGIN
INSERT INTO FACT.DimMeasures
 SELECT [Id]
       ,[Name]
       ,[Description]
 FROM [BDL].[DimMeasures] A
 WHERE A.ID NOT IN (SELECT ID FROM FACT.DimMeasures)
 END
```

```
CREATE PROCEDURE [dbo].[DIMTOPSUBJECTS_IN]
AS BEGIN
INSERT INTO FACT.DimTopSubjects
SELECT [Id]
    ,[Name]
    ,[HasVariables]
    ,[Children]
    ,[CODE]
FROM [BDL].[DimTopSubjects] A
WHERE A.ID NOT IN (SELECT ID FROM FACT.DimTopSubjects)
END
```

```
CREATE PROCEDURE [dbo].[DIMSUBJECTS_IN]
AS BEGIN
INSERT INTO FACT.DimSubjects
SELECT [parentId]
      ,[Id]
      ,[Name]
      ,[HasVariables]
      ,[Children]
      ,[CODE]
FROM [BDL].[DimSubjects] A
WHERE A.ID NOT IN (SELECT ID FROM FACT.DimSubjects)
END
```

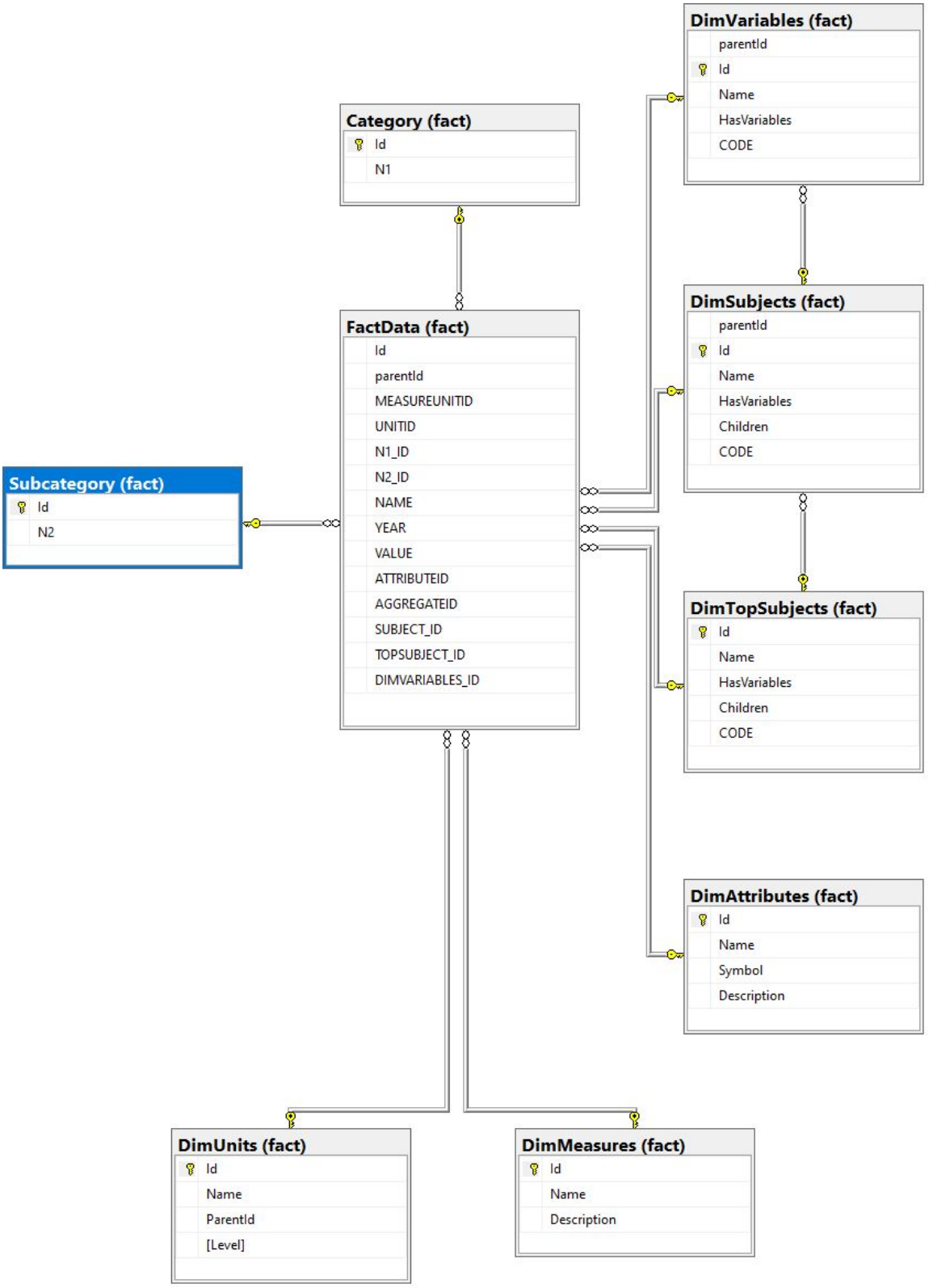
```
CREATE PROCEDURE [dbo].[DIMUNITS_IN]
AS BEGIN
INSERT INTO FACT.DimUnits
SELECT [Id]
      ,[Name]
      ,[ParentId]
      ,[Level]
FROM [bdl].[DimUnits] A
WHERE A.ID NOT IN (SELECT ID FROM FACT.DimUnits)
END
```

```sql
CREATE PROCEDURE [dbo].[DIMVARIABLES_IN]
AS BEGIN
INSERT INTO FACT.DimVariables
SELECT [parentId]
      ,[Id]
      ,[Name]
      ,[HasVariables]
      ,[CODE]
FROM [bdl].[DimVariables] A
WHERE A.ID NOT IN (SELECT ID FROM FACT.DimVariables)
END
```

```sql
CREATE PROCEDURE DBO.CATEGORY_IN
AS BEGIN
insert into fact.Category
select ROW_NUMBER() over (order by a.n1 asc) as ID, a.n1 from (select distinct n1 from temp.TempDataByVariable where n1 not in (select N1 from fact.Category)) a
END
```

```sql
CREATE PROCEDURE [dbo].[SUBCATEGORY_IN]
AS BEGIN
insert into fact.Subcategory
select ROW_NUMBER() over (order by a.n2 asc) as ID, a.n2 from (select distinct n2 from temp.TempDataByVariable where n2 not in (select n2 from fact.Subcategory)) a
END
```

Docelowa (fakty i wymiary) struktura bazodanowa schematu dla SSIS:



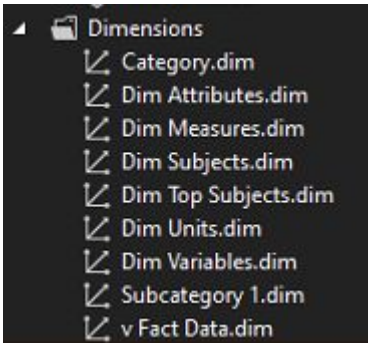
```sql
CREATE PROCEDURE [dbo].[FACTDATA_IN]
AS BEGIN
INSERT INTO FACT.FACTDATA
    (
        ID
    ,   PARENTID
    ,   MEASUREUNITID
    ,   UNITID
    ,   N1_ID
    ,   N2_ID
    ,   NAME
    ,   YEAR
    ,   VALUE
    ,   ATTRIBUTEID
    ,   AGGREGATEID
    ,   SUBJECT_ID
    ,   TOPSUBJECT_ID
    ,   DIMVARIABLES_ID
    )
SELECT
    F.ID AS ID
,   S.ID AS PARENTID
,   F.MEASUREUNITID
,   F.UNITID
,   C.ID AS N1_ID
,   SC.ID AS N2_ID
,   F.NAME
,   F.YEAR
,   F.VALUE
,   F.ATTRIBUTEID
,   F.AGGREGATEID
,   S.ID AS SUBJECT_ID
,   T.ID AS TOPSUBJECT_ID
,   DV.Id AS DIMVARIABLES_ID
FROM
    BDL.FACTDATA F
,   TEMP.DATA_TMP V
,   BDL.DimVariables DV
,   BDL.DIMSUBJECTS S
,   BDL.DIMTOPSUBJECTS T
,   FACT.CATEGORY C
,   FACT.SUBCATEGORY SC
WHERE
    F.ID=V.VARIABLEID
AND V.parentId=DV.Id
AND DV.PARENTID=S.ID
AND S.PARENTID=T.ID
AND C.N1=F.N1
AND SC.N2=F.N2
AND F.ID NOT IN (SELECT ID FROM FACT.FactData)
END
```

Baza danych SSAS

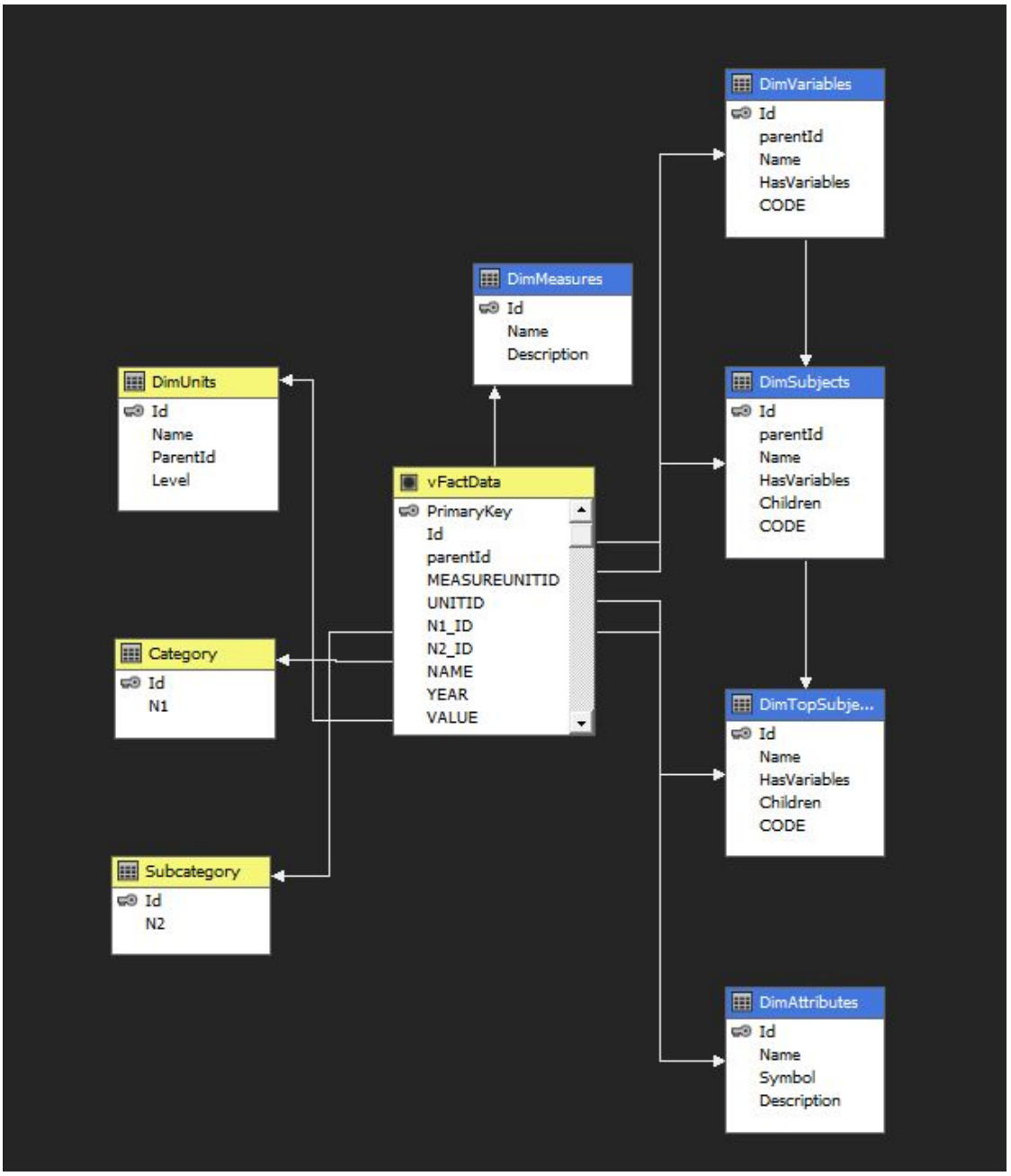
Usługa SQL Server Analysis (SSAS) jak sama nazwa wskazuje działa na bazie danych SQL Server. W projekcie wykorzystywana jest jedna kostka OLAP w schemacie gwiazdy. Na potrzeby projektu stworzono kilka wymiarów oraz widok tabeli faktów z którym łączy się usługa analityczna SQL Server.

```
CREATE view [dbo].[vFactData] as
    select *, cast(value as decimal(18,2)) as DecimalValue from fact.FactData
GO
```

Na przedstawionym poniżej wycinku ekranu widoczne są dodane do SSAS wymiary.

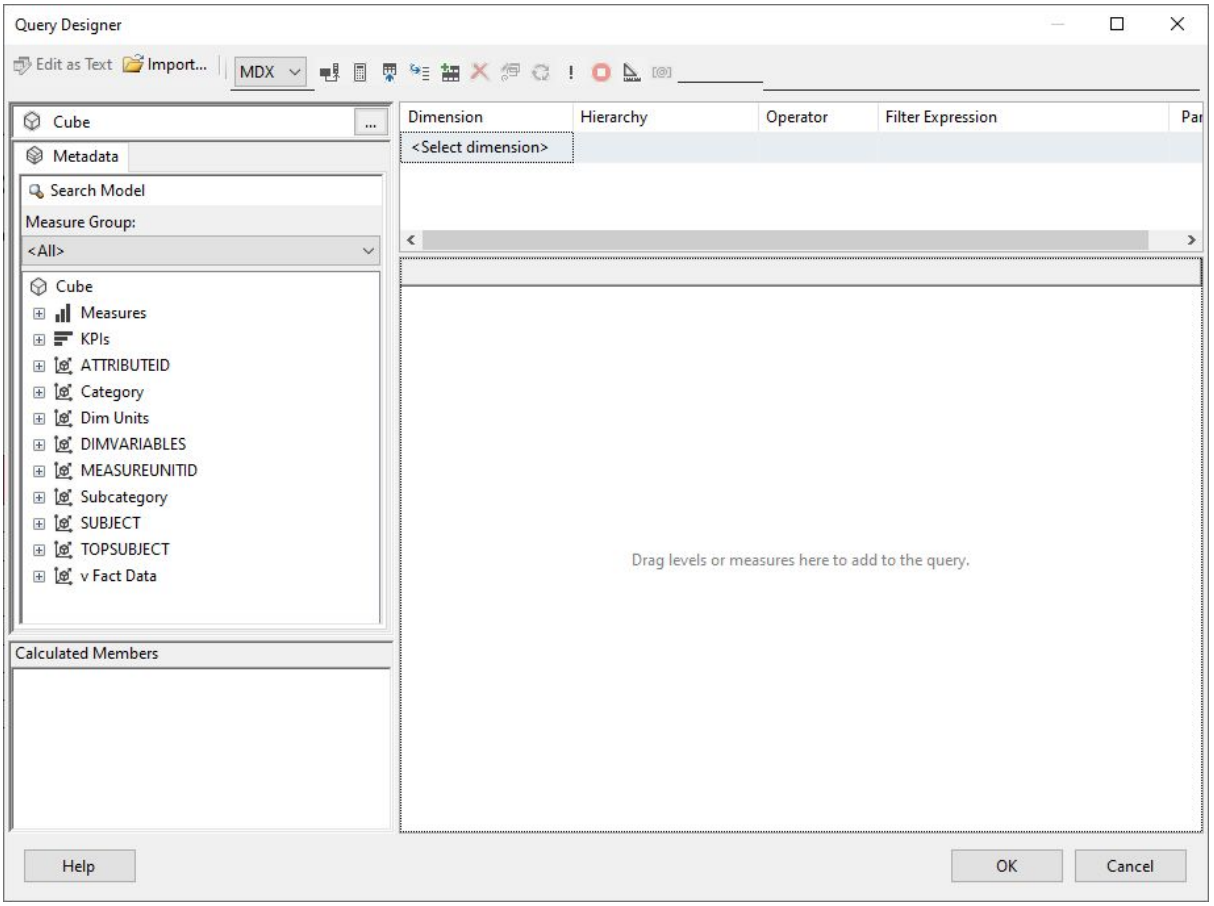


W SSAS istnieje kostka OLAP *Cube.cube.* Do kostki dodane zostały tylko te wymiary które będą potrzebne do wygenerowania raportów które zostały zamówione przez naszego fikcyjnego klienta. Kostka korzysta z widoku faktów (vFactData), wymiaru z jednostkami administracyjnymi (DimUnits), kategoriami (Category) i podkategoriami (Subcategory). Poniżej przedstawiono podgląd struktury kostki *Cube*.


DimVariables
Id
parentId
Name
HasVariables
CODE
DimMeasures
Id
Name
Description
DimUnits
Id
Name
ParentId
Level
DimSubjects
Id
parentId
Name
HasVariables
Children
CODE
vFactData
PrimaryKey
Id
parentId
MEASUREUNITID
UNITID
N1_ID
N2_ID
NAME
YEAR
VALUE
Category
Id
N1
DimTopSubje...
Id
Name
HasVariables
Children
CODE
Subcategory
Id
N2
DimAttributes
Id
Name
Symbol
Description

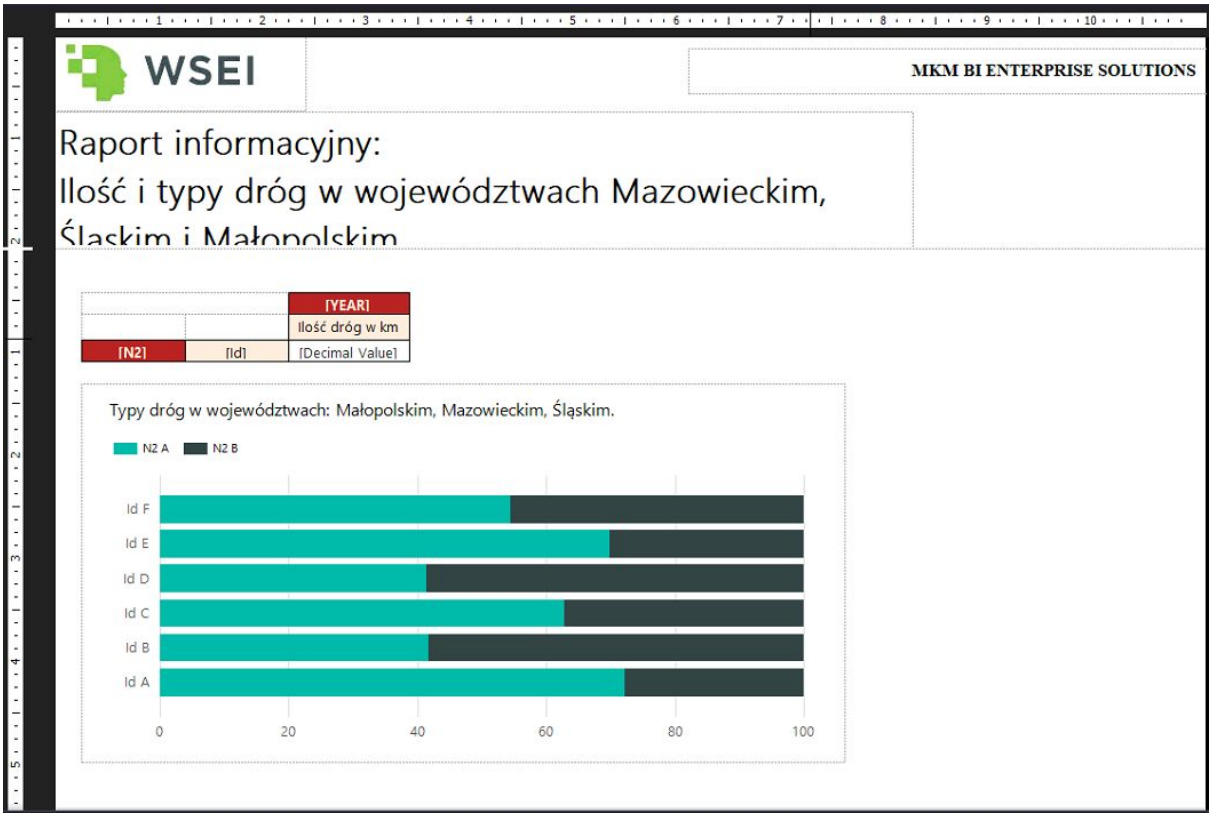
Raportowanie

Projekt wykonywany przez MKM BI Enterprises zakłada jednorazowe raporty raz do roku. Z tego powodu raporty projektowane są raz w pełni manualnie i dodane do głównej solucji nadają się do ponownego corocznego użycia. Raporty przygotowane zostały zgodnie z założeniem ‘real life scenario’, czyli raportów dla wojewody wybranego obszaru. Raporty mają na celu przedstawiać aktualną sytuacje w województwie oraz dla porównania przedstawiać dane z okolicznych lub zbliżonych parametrami województw. Przygotowane zostały trzy raporty, które związane są z zarejestrowanymi w Polsce pojazdami oraz istniejącymi drogami. Raporty przygotowano przy pomocy usługi SQL Server Reporting Services (SSRS). Przy użyciu Visual Studio z SSRS możliwe było połączenie się bezpośrednio do wcześniej utworzonej kostki i zbudowanie raportów oraz korzystanie z potrzebnych filtrów przy użyciu SSRS Query Designer dostępnej w procesie tworzenia nowego raportu który zaciąga dane z kostki analitycznej. Poniżej przedstawiono wycinek ekranu z okna Query Designer. Możemy wybrać tu interesujące nas dane, założyć interesujące nas filtry oraz zobaczyć podgląd danych zanim dane zostaną przekazane dalej.



Raporty zostały zbudowane przy użyciu dostępnej w SSRS funkcji “Design”, jest tam możliwe budowanie tabel, wykresów, wskaźników i wiele więcej. Przy naszych danych do zobrazowania zostały użyte tabelę oraz wykres. Poniżej przedstawiony jest wycinek ekranu z narzędzia do projektowania raportów, ułożona została tabela która później zapełni się

danymi, oraz dodany i skonfigurowany został wykres który również przy wyświetlaniu wypełni się danymi.



Poniższa lista prezentuje opis oraz podgląd przygotowanych na potrzebę projektu raportów.

1. **Raport dot. grup wiekowych.**
   Pierwszy raport stanowi informację o ilości pojazdów w danej grupie wiekowej w danym województwie. Wybrane zostały pojazdy z grupy wiekowej 21-25 lat (Czyli wyprodukowane przed rokiem 2000) oraz samochody które mają maksymalnie rok. Raport dotyczy dwóch sąsiadujących ze sobą województw, Małopolskiego i Śląskiego na przekroju czterech lat (2015-2018). Poniższy wycinek ekranu przedstawia stronę raportu dotyczącą woj. Śląskiego.

WSEI

MKM BI ENTERPRISE SOLUTIONS

RAPORT EKOLOGICZNY

Porównanie liczby pojazdów z dwóch grup wiekowych w województwach Małoposlkim i Śląskim.

(2016-2018)

ŚLĄSKIE

		motocykle	samochody ciężarowe	samochody osobowe
2015	21-25 lat	55885	183814	1497754
	do 1 roku	27661	81939	511059
2016	21-25 lat	54490	195987	1638227
	do 1 roku	36672	77293	578400
2017	21-25 lat	59379	208435	1730950
	do 1 roku	19386	78496	649427
2018	21-25 lat	70253	225946	1862029
	do 1 roku	17542	81997	717715

2. **Raport dot. paliw.**
   Raport przedstawia ilość pojazdów podzielone na paliwa jakimi są napędzane, raport również dotyczy kilku województw.województwa Małopolskie oraz Śląskie.
   Na potrzeby raportu wybrano dwa najbardziej popularne typy paliw, olej napędowy oraz benzyna, dodatkowo dołączono jeszcze kategorie “Pozostałe” w której zawierają się pojazdy zasilane na alternatywne źródła energii.

WSEI

MKM BI ENTERPRISE SOLUTIONS

RAPORT EKOLOGICZNY
Porównanie ilości pojazdów używających poszczególnych paliw w kilku województwach.

		autobusy				ciągniki siodłowe				samochody	
		2015	2016	2017	2018	2015	2016	2017	2018	2015	2016
MAŁOPOLSKIE	benzyna	3034	3008	3136	3117	467	467	454	461	317760	311750
	olej napędowy	67898	70573	63136	65043	129766	142688	141126	148992	1208755	1264550
	pozostałe	3635	3686	13242	13792	3866	3866	14829	15437	60090	59725
MAZOWIECKIE	benzyna	3418	3430	3437	3418	2253	2266	2278	2278	722368	710150
	olej napędowy	97734	103546	102272	105210	483898	542208	578138	633766	2359117	2456960
	pozostałe	1779	1818	7347	9318	2560	2605	22579	24563	28998	28998
POMORSKIE	benzyna	1389	1389	1395	1376	717	704	698	698	216397	210976
	olej napędowy	38643	40723	35392	36294	127187	137530	134035	143814	806528	840646
	pozostałe	5542	5606	10675	10784	7533	7520	19776	20307	102138	101882
ŚLĄSKIE	benzyna	2458	2458	2483	2490	1344	1325	1299	1306	452083	442509
	olej napędowy	66682	67872	51526	52627	199501	215872	203162	218509	1427315	1476902
	pozostałe	922	1030	19072	19187	45	45	28813	29824	3718	3866

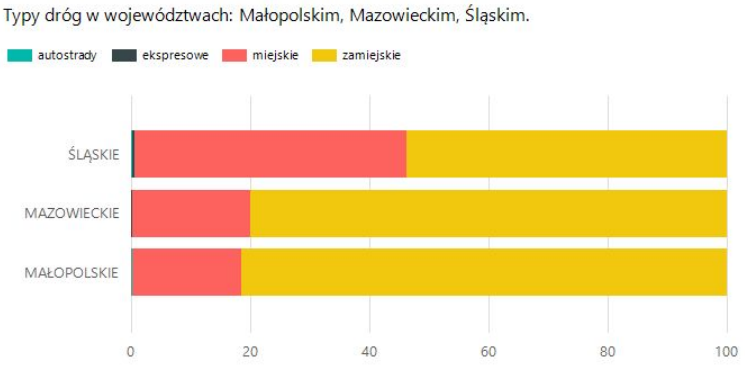
3. **Raport dot. dróg.**
   Kolejny raport przedstawia to jakie drogi występują w danym województwie, oraz ich ilość w kilometrach. Wykres zawarty w raporcie prezentuje wizualnie to jakiego typu drogi występują w danym województwie najczęściej. Do tego raportu zestawiono trzy województwa - Małopolskie, Mazowieckie i Śląskie.

WSEI

MKM BI ENTERPRISE SOLUTIONS

Raport informacyjny:
Ilość i typy dróg w województwach Mazowieckim, Śląskim i Małopolskim.

		2015	2016	2017	2018
		Ilość dróg w km	Ilość dróg w km	Ilość dróg w km	Ilość dróg w km
autostrady	MAŁOPOLSKIE	14001	14001	14001	14001
	MAZOWIECKIE	6295	6295	6295	6295
	ŚLĄSKIE	15653	15509	15509	15509
ekspresowe	MAŁOPOLSKIE	2607	2607	3091	3091
	MAZOWIECKIE	22516	22516	26775	34131
	ŚLĄSKIE	15571	15643	15761	15761
miejskie	MAŁOPOLSKIE	1110554	1127040	1154906	1158694
	MAZOWIECKIE	1803059	1827802	1865050	1896051
	ŚLĄSKIE	2466189	2491981	2489075	2465165
zamiejskie	MAŁOPOLSKIE	4907674	4969984	5189478	5153050
	MAZOWIECKIE	7302797	7437453	7631027	7773338
	ŚLĄSKIE	2901747	2908429	2926874	2937894



Opis wdrożenia produkcyjnego

Na chwilę obecną nie planuje się żadnych wdrożeń produkcyjnych projektu. Niezależnie od tego faktu, przedstawiono poniżej kroki instalacji na docelowym serwerze.

Wymagania systemowe

Projekt potrzebuje wszystkich składników, jakie były opisane wcześniej, tj. SQL Server 2017 Developer, jak i również pakiety programowe pochodne, tj. Analysis Services, Integration Services oraz Reporting Services.
Wszystkie te komponenty powinny (aczkolwiek nie muszą) znajdować się na jednym serwerze. Zaletą instalacji na jednym serwerze jest na pewno łatwość konfiguracji oraz izolacja pod kątem bezpieczeństwa.

Kroki instalacyjne

Instalacja komponentów odbywa się z użyciem solucji Visual Studio. Solucja ta nie musi znajdować się na docelowym serwerze instalacyjnym. Ważne jest, aby z poziomu komputera na którym odbywa się instalacja, był dostęp do serwera docelowego (miejsca docelowego instalacji).

1. Pobranie najnowszej wersji solucji Visual Studio z GitHub
2. Utworzenie nowej czystej bazy danych - katalogu docelowego, na docelowej instancji SQL Server
3. Należy zainstalować projekt BDL_DB w utworzonej w pkcie 2. bazie danych
4. Należy zainstalować (wdrożyć) projekt BDL_Analysis
5. Należy zainstalować (wdrożyć) projekt BDL_Integration
6. Końcowym krokiem jest instalacja (wdrożenie) projektu BDL_Reporting

Możliwe błędy

Błędy które mogą się pojawiać będą raczej standardowe i związane z konfiguracją połączeń do docelowych serwisów.

1. Projekt Analysis - należy pamiętać o dodaniu użytkownika na którym działa usługa, jako login do SQL Server
2. Pamiętać trzeba o dobrym skonfigurowaniu połączenia do Reporting Services

Ze względu na nietypowość i złożoność połączeń między serwisami zaleca się, aby wdrożenia przeprowadzali wykwalifikowani inżynierowie z doświadczeniem.

Podsumowanie

Projekt miał pokazać możliwości technologicznie pobierania danych z zewnętrznych źródeł, zapisania ich lokalnie na serwerze celem ich dalszego przetworzenia, tak aby można było zaprezentować je w formie raportu. Myślę, że fakt ten udało się zaprezentować i udowodnić w praktyce jego działanie.

Polecamy się na przyszłość, MKM BI Enterprise Solutions! :)

Spis treści