Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

WYKORZYSTANIE MS SQL I PAKIETÓW MS BI DO BUDOWY APLIKACJI

PROJEKT: GENERACJA JPK_WB

Bartłomiej Anczok

307330

Spis treści

1. WSTĘP

1.1. CZYM JEST JPK_WB?

JPK_WB zawiera informacje z wyciągów rachunków bankowych przedsiębiorców. Dzięki dostarczaniu tych raportów szybciej identyfikowane są podejrzane transakcje, a także weryfikowane przychody i koszty z obrotami na rachunku bankowym. Istnienie JPK_WB ma znacznie ograniczyć nadużycia i oszustwa podatkowe.

JPK_WB obowiązuje od lipca 2018 roku (mikro-, małe i średnie przedsiębiorstwa), zgodnie z pierwotnym założeniem.

1.2. STRUKTURA JPK_WB

Jednolity Plik Kontrolny dla wyciągu bankowego zawiera informacje o przeprowadzonych transakcjach na rachunku bankowym przedsiębiorcy, które obejmują m.in:

- dane identyfikacyjne nadawcy i odbiorcy zlecenia płatniczego - czyli nazwę, adres (o ile taki posiada), numer NIP lub REGON,
- numer rachunku nadawcy i odbiorcy,
- numer IBAN rachunku, którego dotyczy wyciąg,
- datę i czas obciążenia rachunku nadawcy oraz datę i czas wpłaty gotówkowej,
- kwotę i walutę,
- tytuł i opis zlecenia płatniczego,
- saldo rachunku przedsiębiorcy po dokonaniu zlecenia płatniczego,
- informację o numerze rachunku wirtualnego utworzonego w celu identyfikacji masowych płatności, jeżeli zlecenie płatnicze dotyczy uznania takiego rachunku.

1.3. KOGO DOTYCZY JPK_WB?

Obowiązek składania raportów z wyciągów bankowych dotyczy dużych, średnich i małych przedsiębiorstw. Duże przedsiębiorstwa sporządzają JPK_WB już od lipca 2016 roku, natomiast małe i średnie przedsiębiorstwa od lipca 2018 roku.

1.4. TRYB SKŁADANIA JPK_WB?

Przedsiębiorca powinien mieć możliwość wygenerować pliki JPK_WB na swoim rachunku bankowym. Co ważne, jeżeli przedsiębiorca posiada rachunek bankowy w zagranicznej jednostce, będzie musiał dostarczyć JPK_WB we własnym zakresie.

Wprowadzenie Jednolitego Pliku Kontrolnego dla wyciągu bankowego ma znacznie skrócić czas weryfikacji deklaracji VAT w przypadku wystąpienia zwrotu podatku VAT oraz przyspieszyć identyfikację nieuczciwych kontrahentów. Dodatkową korzyścią tworzenia plików JPK_WB ma być usprawnienie procesu kontroli, która znacznie szybciej będzie wykrywać ewentualne nieprawidłowości na rachunkach bankowych przedsiębiorców.

2. NARZĘDZIA

Do wykonania projektu postanowiłem wykorzystać:

- SQL Server
- SQL Server Management Studio
- Pakiety SSIS

Microsoft SQL Server– system zarządzania bazą danych, wspierany i rozpowszechniany przez korporację Microsoft. Jest to główny produkt bazodanowy tej firmy, który charakteryzuje się tym, iż jako język zapytań używany jest przede wszystkim Transact-SQL, który stanowi rozwinięcie standardu ANSI/ISO.

MS SQL Server jest platformą bazodanową typu klient-serwer. W stosunku do Microsoft Jet, który stosowany jest w programie MS Access, odznacza się lepszą wydajnością, niezawodnością i skalowalnością. Przede wszystkim są tu zaimplementowane wszelkie mechanizmy wpływające na bezpieczeństwo operacji (m.in. procedury wyzwalane).

Na potrzeby projektu uruchomiłem lokalny serwer. Działa on na wersji z 2019 roku.

SQL Server Management Studio - zintegrowane środowisko do zarządzania wszystkimi komponentami (baza danych, usługi analityczne, usługi raportowe itd.), wchodzącymi w skład Microsoft SQL Server. Zawiera narzędzia do konfiguracji, monitorowania i administrowania instancjami SQL Server. Umożliwia budowę zapytań i skryptów, zawiera zarówno edytor skryptów jak i narzędzia graficzne. Aplikacja po raz pierwszy pojawiła się wraz z Microsoft SQL Server 2005.

SQL Server Integration Services (SSIS) – graficzne narzędzie ETL firmy Microsoft włączone do Microsoft SQL Server od wersji 2005. SQL Server Integration Services jest składnikiem oprogramowania Microsoft SQL Server, który powstał w celu wykonywania szerokiego zakresu zadań migracji danych. Do ich tworzenia oraz zarządania nimi wykorzystałem środowisko Microsoft Visual Studio 2019.

3. OGÓLNE

3.1. WYMAGANIA FUNKCJONALNE

Numer	Opis Wyma
WF0100	Dane wczytywane sa z dówch plików txt o okreslonym formacie
WF0110	Plik pierwszy zawiera dane właściciela rachunku
WF0111	Pierwszy wiersz pliku zawiera numer IBAN
WF0112	Drugi wiersz zawiera dane wlasciciela rachunku rozdzielne średnikiem (NIP;Nazwa;[REGON])
WF0113	Trzeci wiersz zawiera dane adresowe: (kod_kraju;woj.;powiat;gmina;ulica;nr_domu;[nr.lokalu];miejscowosc;kod_pocztowy;poczta)
WF0120	Plik drugi
WF0121	Nazwa pliku musi odpowiadać numerowi IBAN z pliku pierwszego
WF0122	Pierwszy wiersz zawiera saldo początkowe
WF0123	Każdy koljny wiersz zawira dane transkacji oddzielone srednikiem (data;kontrahent;opis;kwota_operacji)
WF0124	Przy uznaniu kwota dodatnia, przy obciazeniu ujemna
WF0200	Wczytanie pliku odbuwa się przez pakiet SSIS
WF0210	Przy wczytaniu weryfikowana jest popawność struktury pliku
WF0220	Dane wczytywane sa do tabeli tymczasowej
WF0221	Tabela tymczasowa jest czyszczona na poczatku operacji
WF0300	Po wczytaniu danych do tabeli tymaczasowej uruhamiana jest trigger tworzacy tabele docelowe z danymi
WF0310	Na poczatku czyscimy tabele docelowe z wyjątkiem tabli podmitoow i adresow
WF0311	Jeśli generujemy JPK dla tego samego podmiotu co ostatnio, to nie usuwamy rekordow wyciagow (sprawdzamy na podstawie numeru IBAN oraz NIP)
WF0312	warunkiem koniecznym jest również ciaglosc salda miedzy plikami
WF0400	Generacj pliku XML zgodnego z JPK_WB odbywa się procedurą
WF0410	Procedura przyjmuje 3 argumenty date_od; date_do;miejsce_zapisu_pliku_xml
WF0420	Argumenty dat sa opcjonalne
WF0421	Wprzypadku niepodania daty_od procedura wygeneruje JPK dla wszystkich rekordow do daty_do
WF0422	Analogiczna sytuacja przy nie podaniu daty_do
WF0423	W przypadku nie podania zadnej z dat, procedura wygeneruje JPK dla wszyskich danych
WF0500	Procedura domyślnie obsługuje tylko walutę PLN

3.2. PRZYKŁAD PLIKU PODMIOT1.TXT

```
PL25109024028713951641489122
4058195881;podmiot1;137318807;0202
PL;Slaskie;Lubliniecki;Lubliniec;Lipowa;14; ;Lubliniec;42-700;Lubliniec
```

3.3. PRZYKŁAD PLIKU IBAN.TXT

```
1056.36
02.18.2022;aaaa;piwo;-58.00
02.18.2022;bbbb;bilet;-160.00
02.19.2022;eeee;burgery;-55.55
02.21.2022;cccc;ubrania;-375.75
02.22.2022;dddd;zwrot;20.00
```

3.4. URUCHOMIENIE

Przed uruchomieniem procedury generującej plik jpk, konieczne jest załadowanie danych do bazy. W tym celu należy uruchomić pakiet SSIS zajmujący się tym zadaniem.

Dla pakietu SSIS konieczne jest zdefiniowanie miejsca w którym znajduję się plik podmiot.txt (Source-podmiot.txt) oraz IBAN.txt (Script task). W przypadku drugiego pliku należy zmodyfikować zmienną do folderu z plikami: ***folderPath.***

Po załadowaniu danych, możemy przystąpić do uruchomienia procedury generującej plik jpk_wb. Tutaj również konieczne jest wykonanie kilku kroków. Po pierwsze, należy zezwolić serwerowi na uruchamiania : `xp_cmdshell.` Jest to konieczne dla poprawnego zapisu danych na dysku. W tym celu należy w SSMS wykonać polecania:

```
EXEC master.dbo.sp_configure 'show advanced options', 1
RECONFIGURE
EXEC master.dbo.sp_configure 'xp_cmdshell', 1
RECONFIGURE
```

Po tym kroku, możemy finalnie przystąpić do generacji JPK_WB. Generacja tego pliku odbywa się poprzez procedurę ***GEN_JPK.*** Przyjmuję ona 3 argumenty:

- @od – określa datę w formacie: MM-DD-YYYY od której ma zostać wygenerowany JPK_WB \
- @do – określa datę w formacie: MM-DD-YYYY do której ma zostać wygenerowany JPK_WB
- @path – określa folder zapisu pliku JPK_WB

Argumenty dat są opcjonalne. W przypadku ich nie podania, zostanie wygenerowany plik dla wszystkich danych w bazie.

Przykład uruchomienia:

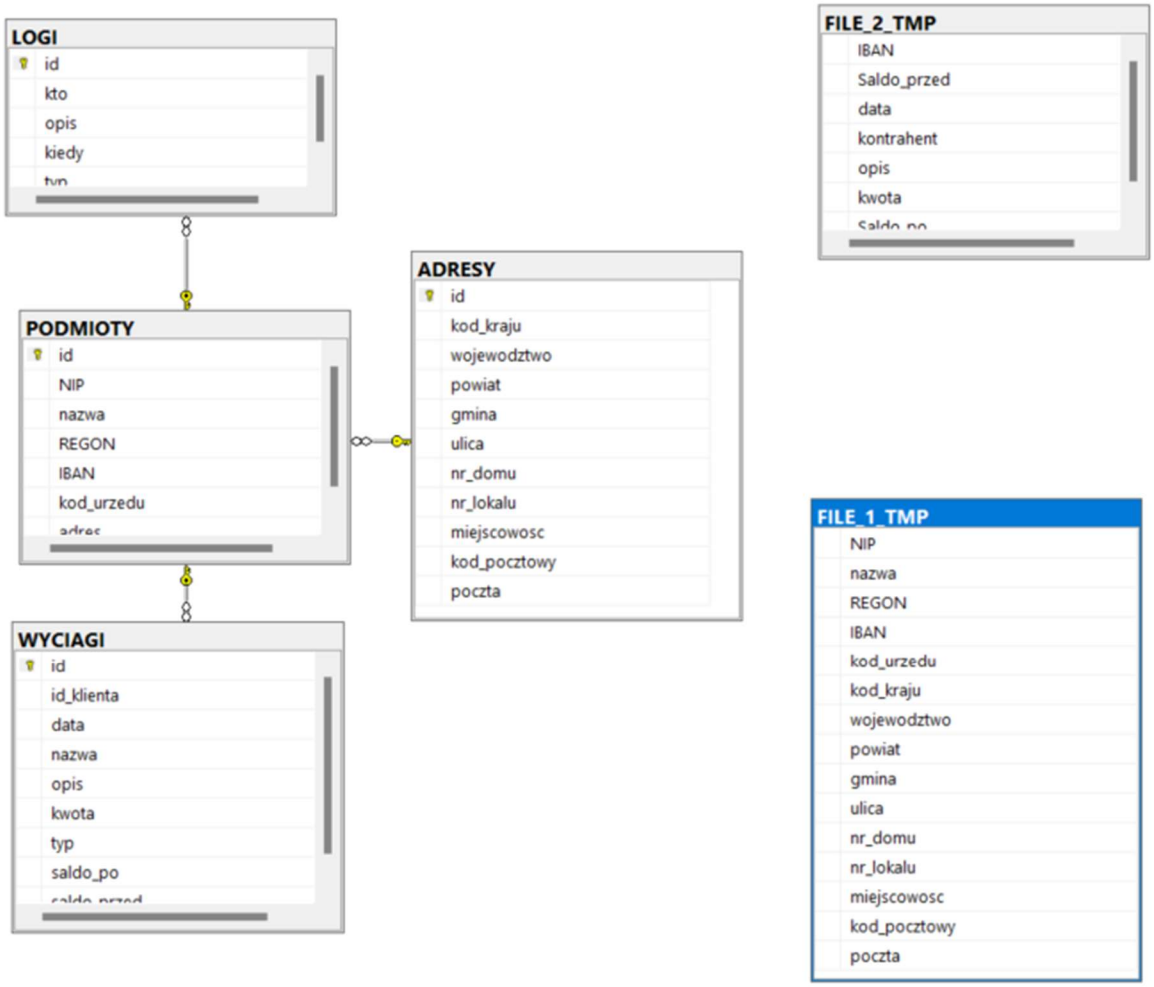
```
EXEC dbo.GEN_JPK @path='C:\tmp\'
```

3.5. DALSZY ROZWÓJ

Generacja pliku JPK_WB działa w pełni poprawnie i ciężko znaleźć inny kierunek w którym można by ten projekt dalej rozwijać. To co na pewno można dodać to obsługę większej ilości walut a nie tylko PLN jak jest obecnie.

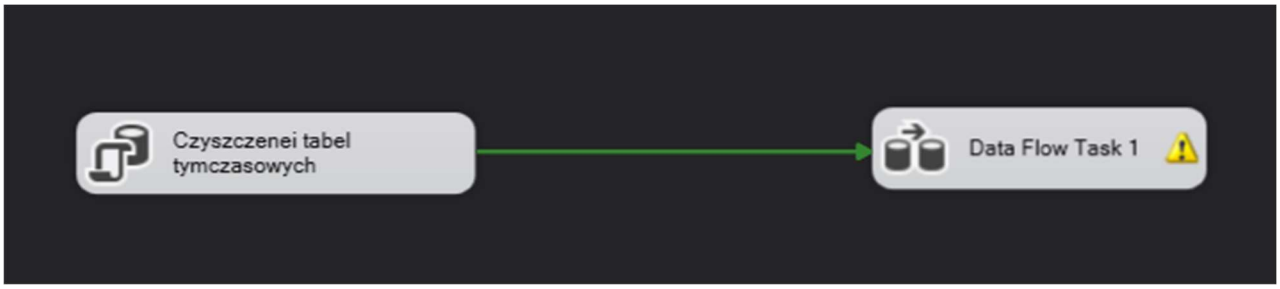
4. WYKONANIE PROJEKTU

4.1. STRUKTURA BAZY DANYCH



4.2. PAKIETY SSIS

4.2.1. CONTROL FLOW

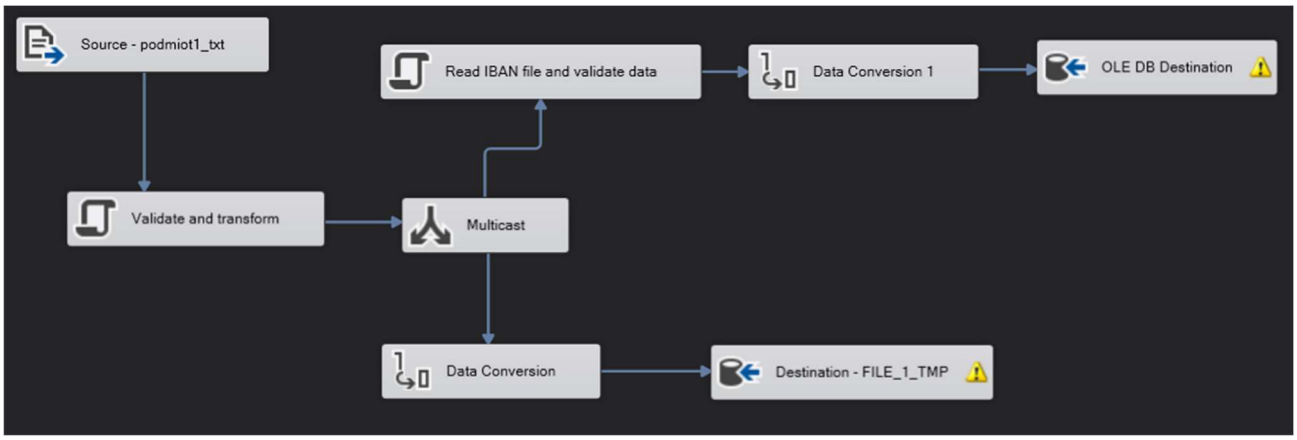


Czyszczenie tabel tymczasowych:

TRUNCATE TABLE [dbo].[FILE_1_TMP];

TRUNCATE TABLE [dbo].[FILE_2_TMP];

4.2.2. DATA FLOW



4.2.2.1. VALIDATE AND TRANSFORM:

```
int current_row = 1;
List<string> list = new List<string>();
public override void Input0_ProcessInputRow(Input0Buffer Row)
{
{
	if(!ValidateData(Row.Column0, current_row)){
		bool cancel = false;
		this.ComponentMetaData.FireError(0, "My sub", "Error", string.Empty,
0, out cancel);

		throw new Exception("Wrong file format");
	}

	if (current_row > 1)
	{
		foreach (string variableName in Row.Column0.Split(';'))
		{
			Console.WriteLine(variableName);
			list.Add(variableName);
		}

	} else
	{
		list.Add(Row.Column0);
	}

	current_row++;
	if (current_row == 4)
	{
		string[] arr = list.ToArray();

		Output0Buffer.AddRow();

		Output0Buffer.IBAN = arr[0];
		Output0Buffer.NIP = arr[1];
		Output0Buffer.Nazwa = arr[2];
		Output0Buffer.REGON = arr[3];
		Output0Buffer.Urzad = arr[4];
		Output0Buffer.Kraj = arr[5];
		Output0Buffer.Woj = arr[6];
		Output0Buffer.Powiat = arr[7];
		Output0Buffer.gmina = arr[8];
		Output0Buffer.Ulica = arr[9];
		Output0Buffer.nrdomu = arr[10];
		Output0Buffer.nrlokalu = arr[11];
		Output0Buffer.miejscowosc = arr[12];
		Output0Buffer.kodpocztowy = arr[13];
		Output0Buffer.poczta = arr[14];
	}
}
}

public Boolean ValidateData(String row, int row_number)
{
{
	Boolean check = false;
	switch (row_number)
	{
		case 1:
			string strRegex = @"(^[A-Z]{2}[0-9]{26}$)";
			check = Regex.IsMatch(row, strRegex);
			break;
		case 2:
			string strRegex1 = @"(^[0-9]{10}$)";
```

```
                    string strRegex2 = @"(^[0-9]{9}$)";
                    string strRegex3 = @"(^[0-9]{4}$)";

                    String[] tmp = row.Split(';');

                    if (tmp.Length == 4 && Regex.IsMatch(tmp[0], strRegex1)
                            && Regex.IsMatch(tmp[2], strRegex2) &&
Regex.IsMatch(tmp[3], strRegex3){
                            check = true;
                    }

                    break;
            case 3:
                    string strRegex4 = @"(^[A-Z]{2}$)";
                    string strRegex5 = @"(^[A-Z]{1}[a-z]{1,}$)";
                    string strRegex6 = @"(^(\s*|\d+)$)";
                    string strRegex7 = @"(^(\d{2}-\d{3})$)";


                    String[] tmp1 = row.Split(';');


                    if (tmp1.Length == 10 && Regex.IsMatch(tmp1[0], strRegex4) &&
Regex.IsMatch(tmp1[1], strRegex5)
                            && Regex.IsMatch(tmp1[2], strRegex5) &&
Regex.IsMatch(tmp1[3], strRegex5)
                            && Regex.IsMatch(tmp1[4], strRegex5) &&
Regex.IsMatch(tmp1[5], strRegex6)
                            && Regex.IsMatch(tmp1[6], strRegex6) &&
Regex.IsMatch(tmp1[7], strRegex5)
                            && Regex.IsMatch(tmp1[8], strRegex7) &&
Regex.IsMatch(tmp1[9], strRegex5))
                    {
                            check = true;
                    }

                    break;
        }
        return check;
}
}

}
```

4.2.2.2. READ IBAN FILE AND VALIDATE DATA:

```
int current_row = 1;
String[] lines;
List<String> rows = new List<String>();

String folderPath = "C:\\Users\\anczo\\OneDrive\\Dokumenty\\SQL Server Management
Studio\\";
public override void Input0_ProcessInputRow(Input0Buffer Row)
{
	if(current_row == 1)
	{
		String IBAN = Row.IBAN;
		lines= File.ReadAllLines(folderPath+ IBAN +".txt", Encoding.UTF8);
		double saldo_poczatkowe = 0.0;
		try
		{
			saldo_poczatkowe = double.Parse(lines[0].Trim(),
CultureInfo.InvariantCulture);
		}
		catch (FormatException e)
		{
			throw new Exception("Error in first line of file: " + IBAN +
".txt");
		}

		for (int i = 1; i < lines.Length; i++)
		{
			if (!validateRow(lines[i]))
			{
				throw new Exception("Error format in " +(i+1)+ "
line of file");
			}
			String[] tmp = lines[i].Split(';');

			String data = tmp[0];
			String kontrahent = tmp[1];
			String opis = tmp[2];
			double kwota = double.Parse(tmp[3].Trim(),
CultureInfo.InvariantCulture);

			Output0Buffer.AddRow();
			Output0Buffer.IBAN = IBAN;
			Output0Buffer.Saldoprzed = (decimal)saldo_poczatkowe;
			Output0Buffer.Data = data;
			Output0Buffer.Kontrahent = kontrahent;
			Output0Buffer.Opis = opis;
			Output0Buffer.Kwota = (decimal)kwota;
			Output0Buffer.Saldopo = (decimal)(saldo_poczatkowe + kwota);

			saldo_poczatkowe += kwota;
		}

	}



}

public Boolean validateRow(String row)
{
	{
		Boolean check = false;
		System.DateTime temp;

		string strRegex = @"(^[a-zA-Z0-9]{1,}$)";
```

```
                string strRegex1 = @"(^[-]?[0-9]{1,}.[0-9]{2}$)";
                string dateRegex = @"(^([0-9]{2}.[0-9]{2}.[0-9]{4})$)";

                string[] tmp = row.Split(';');
                bool a = Regex.IsMatch(tmp[0], dateRegex);
                bool b = Regex.IsMatch(tmp[1], strRegex);
                bool c = Regex.IsMatch(tmp[2], strRegex);
                bool d = Regex.IsMatch(tmp[3], strRegex1);
                if(tmp.Length == 4 && a && b && c && d)
                {
                        check = true;
                }
                return check;
        }
}

}
```

4.3. SQL

4.3.1. TWORZENIE TABEL

```
USE DB_JPK_WB
GO

-- tabele: podmioty (id, NIP, nazwa, REGON, adres), adresy, wyciagi

/*
  DROP TABLE WYCIAGI
  DROP TABLE LOGI
  DROP TABLE PODMIOTY
  DROP TABLE ADRESY
*/

--tabela docelowa adresy, opsije adres podmiotu
IF NOT EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'ADRESY')
    AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
  CREATE TABLE dbo.ADRESY
  ( id          int       NOT NULL IDENTITY CONSTRAINT PK_ADRESY PRIMARY
KEY
  , kod_kraju     nvarchar(2)   NOT NULL DEFAULT 'PL'
  , wojewodztwo     nvarchar(50)  NOT NULL
  , powiat        nvarchar(50)  NOT NULL
  , gmina       nvarchar(50)  NOT NULL
  , ulica       nvarchar(50)  NOT NULL
  , nr_domu       nvarchar(50)  NOT NULL
  , nr_lokalu     nvarchar(50)  NOT NULL DEFAULT ''
  , miejscowosc     nvarchar(50)  NOT NULL
  , kod_pocztowy    nvarchar(6)   NOT NULL
  , poczta        nvarchar(50)  NOT NULL
  )

END
GO

--tabela docelowa pomioty, opsije podmiot
IF NOT EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'PODMIOTY')
    AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
  CREATE TABLE dbo.PODMIOTY
  ( id      int       NOT NULL IDENTITY CONSTRAINT PK_PODMIOTY PRIMARY KEY
  , NIP     nvarchar(10)  NOT NULL UNIQUE
  , nazwa   nvarchar(100) NOT NULL
  , REGON   nvarchar(9)   NULL
  ,   IBAN    nvarchar(30)  NOT NULL UNIQUE
  , kod_urzedu  nvarchar(4)   NOT NULL
  , adres   int       NOT NULL CONSTRAINT FK_PODMIOT_ADRES REFERENCES
adresy(id)
```

```
  )

END
GO

--tabela docelowa wyciagi, opsije transakcje bankowe
IF NOT EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'WYCIAGI')
    AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
  CREATE TABLE dbo.WYCIAGI
  ( id      int       NOT NULL IDENTITY CONSTRAINT PK_WYCIAGI PRIMARY KEY
  ,   id_klienta  int       NOT NULL CONSTRAINT FK_WYCIAG_PODMIOT
REFERENCES PODMIOTY(id)
  , data    DATE      NOT NULL
  , nazwa   nvarchar(100) NOT NULL
  , opis    nvarchar(100) NOT NULL
  , kwota   DECIMAL(19,4) NOT NULL
  , typ     BIT       NOT NULL -- 1 - uznanie, 0 - obciazenie
  , saldo_po  DECIMAL(19,4) NOT NULL
  , saldo_przed DECIMAL(19,4) NOT NULL
  )

END
GO

--tabela logi, opsije hisorie logow
IF NOT EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'LOGI')
    AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
  CREATE TABLE dbo.LOGI
  ( id      int       NOT NULL IDENTITY CONSTRAINT PK_LOGI PRIMARY KEY
  , kto     int       NULL   CONSTRAINT FK_LOG_PODMIOT REFERENCES
PODMIOTY(id)
  , opis    nvarchar(100) NOT NULL
  , kiedy   DATETIME    NOT NULL DEFAULT GETDATE()
  , typ     BIT       NOT NULL -- 1 - blad, 0 - info
  )

END
GO


-- TEMP Tables
--DROP TABLE FILE_1_TMP
IF NOT EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'FILE_1_TMP')
    AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
  CREATE TABLE FILE_1_TMP (
```

```
      NIP          nvarchar(10)  NOT NULL UNIQUE
    , nazwa        nvarchar(100) NOT NULL
    , REGON        nvarchar(9)   NULL
    , IBAN         nvarchar(30)  NOT NULL UNIQUE
    , kod_urzedu   nvarchar(4)   NOT NULL
    , kod_kraju    nvarchar(2)   NOT NULL DEFAULT 'PL'
    , wojewodztwo      varchar(50)   NOT NULL
    , powiat         nvarchar(50)  NOT NULL
    , gmina        nvarchar(50)  NOT NULL
    , ulica        nvarchar(50)  NOT NULL
    , nr_domu        nvarchar(50)  NOT NULL
    , nr_lokalu      nvarchar(50)  NOT NULL DEFAULT ''
    , miejscowosc      nvarchar(50)  NOT NULL
    , kod_pocztowy     nvarchar(6)   NOT NULL
    , poczta         nvarchar(50)  NOT NULL
    )
END
GO

--Drop table FILE_2_TMP
IF NOT EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'FILE_2_TMP')
    AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
  CREATE TABLE FILE_2_TMP (
      IBAN         nvarchar(30)  NOT NULL
    , Saldo_przed      DECIMAL(19,4) NOT NULL
    , data         nvarchar(50)  NOT NULL
    , kontrahent       nvarchar(50)  NOT NULL
    ,   opis         nvarchar(250) NOT NULL
    , kwota        DECIMAL(19,4) NOT NULL
    , Saldo_po       DECIMAL(19,4) NOT NULL
    )
END
GO

--Select * FROM FILE_1_TMP
--Select * FROM FILE_2_TMP
```

4.3.2. TWORZENIE TRIGGER'OW

```
USE DB_JPK_WB
GO

-- FILE 1 trigger, uruchamia sie w trakcie dodania rekordow i laduje dane
do tabel docelowych
IF EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'TRIGGER_INSERT_FILE_1')
    AND   [type] = 'TR'
    )

BEGIN
  DROP TRIGGER TRIGGER_INSERT_FILE_1
END
GO


CREATE TRIGGER TRIGGER_INSERT_FILE_1
  ON dbo.FILE_1_TMP
  AFTER INSERT
  AS
BEGIN
  DECLARE
    @NIP         nvarchar(10)
  , @nazwa         nvarchar(50)
  , @REGON         nvarchar(9)
  , @IBAN        nvarchar(30)
  , @kod_urzedu      nvarchar(4)
  , @wojewodztwo     nvarchar(50)
  , @powiat        nvarchar(50)
  , @gmina         nvarchar(50)
  , @ulica         nvarchar(50)
  , @nr_domu       nvarchar(50)
  , @nr_lokalu       nvarchar(50)
  , @miejscowosc     nvarchar(50)
  , @kod_pocztowy    nvarchar(6)
  , @poczta        nvarchar(50);

  SELECT --pobranie danych z tabeli tymczasowej
    @NIP = NIP,
    @nazwa = nazwa,
    @REGON = REGON,
    @IBAN = IBAN,
    @kod_urzedu = kod_urzedu,
    @wojewodztwo = wojewodztwo,
    @powiat = powiat,
    @gmina = gmina,
    @ulica = ulica,
    @nr_domu = nr_domu,
    @nr_lokalu = nr_lokalu,
    @miejscowosc = miejscowosc,
    @kod_pocztowy = kod_pocztowy,
    @poczta = poczta

  FROM dbo.FILE_1_TMP
```

```
  IF NOT EXISTS --sprawdzenie czy podmiot juz wystepuje w bazie
  ( SELECT 1
    from dbo.PODMIOTY
    WHERE NIP = @NIP and IBAN = @IBAN
    )
  BEGIN
    --najpierw dodajemy adres
    INSERT INTO dbo.ADRESY
      (wojewodztwo, powiat, gmina, ulica, nr_domu, nr_lokalu, miejscowosc,
kod_pocztowy, poczta)
    Values
      (@wojewodztwo, @powiat, @gmina, @ulica, @nr_domu, @nr_lokalu,
@miejscowosc, @kod_pocztowy, @poczta)

    --klucz glowny dodanego adresu
    declare @id int
    select @id = Scope_Identity()

    --dodanie podmiotu do tabeli
    INSERT INTO dbo.PODMIOTY
      (NIP, nazwa, REGON, IBAN, kod_urzedu, adres)
    VALUES
      (@NIP, @nazwa, @REGON, @IBAN, @kod_urzedu, @id)

    --klucz glowny ostatniego dodanego podmiotu
    select @id = Scope_Identity()

    --dodanie do logow
    INSERT INTO dbo.LOGI
      (kto, opis, kiedy, typ)
    VALUES
      (@id, 'Dodanie nowego podmiotu '+@IBAN+' do bazy', SYSDATETIME(), 0)
  END
  ELSE
  BEGIN
    INSERT INTO dbo.LOGI
      (kto, opis, kiedy, typ)
    VALUES
      (@id, 'Proba dodania pomiotu ('+@IBAN+') juz wystepujacego w bazie',
SYSDATETIME(), 1)
  END


END
GO

/* sprawdzenie
SELECT * FROM LOGI
SELECT * FROM ADRESY
SELECT * FROM PODMIOTY
*/

-- FILE 2 trigger uruchamia sie po dodaniu rekordow do tabeli tymaczoswej

IF EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = N'TRIGGER_INSERT_FILE_2')
    AND   [type] = 'TR'
    )
```

```
BEGIN
  DROP TRIGGER TRIGGER_INSERT_FILE_2
END
GO


CREATE TRIGGER TRIGGER_INSERT_FILE_2
  ON dbo.FILE_2_TMP
  AFTER INSERT
  AS
BEGIN
  DECLARE
    @saldo_przed    DECIMAL(19,4)
  , @data       DATETIME
  , @kontrahent     nvarchar(50)
  , @opis       nvarchar(250)
  , @kwota        DECIMAL(19,4)
  , @typ        bit
  , @saldo_po     DECIMAL(19,4)
  , @IBAN       nvarchar(30)
  , @id         int
  , @saldo_check1   DECIMAL(19,4)
  , @saldo_check2   DECIMAL(19,4)
  , @IBAN_check1    nvarchar(30)
  , @IBAN_check2    nvarchar(30);

  --sprawdzenei czy dane z tego samego rachunku
  SELECT @IBAN_check1=IBAN FROM dbo.PODMIOTY WHERE id = (SELECT TOP 1 id
FROM dbo.WYCIAGI)
  SELECT TOP 1 @IBAN_check2=IBAN FROM dbo.FILE_2_TMP

  --sprawdzenie zgodnosci sald
  SELECT TOP 1 @saldo_check1 = saldo_przed FROM dbo.FILE_2_TMP
  SELECT TOP 1 @saldo_check2 = saldo_po FROM dbo.WYCIAGI ORDER BY id DESC

  IF NOT @saldo_check1 = @saldo_check2 or NOT @IBAN_check1 LIKE
@IBAN_check2 --sprawdzenie czy kontynuacja poprzedniego pliku
  BEGIN
    INSERT INTO LOGI VALUES((SELECT TOP 1 id FROM dbo.WYCIAGI),
'Czyszczenie tabeli wyciagi', SYSDATETIME(), 0)
    TRUNCATE TABLE dbo.WYCIAGI
  END

  -- dodanie wartosci poprzez petle
  DECLARE cursor_trigger INSENSITIVE CURSOR
  FOR
    SELECT * FROM dbo.FILE_2_TMP

  OPEN cursor_trigger;

  FETCH NEXT FROM cursor_trigger INTO @IBAN, @saldo_przed, @data,
@kontrahent, @opis, @kwota, @saldo_po
  WHILE @@FETCH_STATUS = 0
    BEGIN
      if CHARINDEX('-',@kwota) > 0 --ustalenie czy uznanie czy obciazenie
      begin
        SET @typ = 0
      end
      else
      begin
```

```
        SET @typ = 1
      end

      SELECT @id=id FROM dbo.PODMIOTY WHERE IBAN LIKE @IBAN --pobranie id
podmiotu

      INSERT INTO dbo.WYCIAGI
        (id_klienta, data, nazwa, opis, kwota, typ, saldo_po, saldo_przed)
      VALUES
        (@id, @data, @kontrahent, @opis, @kwota, @typ, @saldo_po,
@saldo_przed)



    FETCH NEXT FROM cursor_trigger INTO @IBAN, @saldo_przed, @data,
@kontrahent, @opis, @kwota, @saldo_po
    END
  CLOSE cursor_trigger
  DEALLOCATE cursor_trigger

  INSERT INTO LOGI VALUES(@id, 'Dodanie rekordow wyciagu bankowego
dla'+@IBAN, SYSDATETIME(), 0)

END


/*
SELECT * FROM dbo.PODMIOTY
SELECT * FROM dbo.ADRESY
SELECT * FROM dbo.WYCIAGI
SELECT * FROM dbo.LOGI
*/
```

4.3.3. TWORZENIE FUNKCJI

```
USE DB_JPK_WB
GO

IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'CONVERT_MONEY')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.CONVERT_MONEY () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.CONVERT_MONEY(@d DECIMAL(19,4) )
/* format kwoty dopuszczalny w plikach JPK */
RETURNS nvarchar(20)
AS
BEGIN
  RETURN RTRIM(LTRIM(STR(@d,18,2)))
END
GO


IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'CONVERT_DATE')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.CONVERT_DATE () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.CONVERT_DATE(@d datetime )
/* format daty dopuszczalny w plikach JPK */
RETURNS nchar(10)
AS
BEGIN
  RETURN CONVERT(nchar(10), @d, 120)
END
GO

IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'GET_SALDO_POCZ')
    AND   (o.xtype = 'FN')
  )
```

```
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.GET_SALDO_POCZ () returns money AS
begin return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.GET_SALDO_POCZ(@od date)
--pobranie salda poczatkowego
RETURNS nvarchar(20)
AS
BEGIN
  DECLARE @saldo DECIMAL(19,4);
  SELECT TOP 1 @saldo= saldo_przed FROM WYCIAGI WHERE data >= @od
  RETURN RTRIM(LTRIM(STR(@saldo,18,2)))
END
GO


IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'GET_SALDO_KON')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.GET_SALDO_KON () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.GET_SALDO_KON(@do date)
--pobranie salda koncowego
RETURNS nvarchar(20)
AS
BEGIN
  DECLARE @saldo DECIMAL(19,4);
  SELECT TOP 1 @saldo= saldo_po FROM WYCIAGI WHERE data <= @do ORDER BY id
DESC
  RETURN RTRIM(LTRIM(STR(@saldo,18,2)))
END
GO


IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'GET_ROW_NUM')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.GET_ROW_NUM () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END
```

```
GO

ALTER FUNCTION dbo.GET_ROW_NUM(@od date, @do date)
--pobranie ilosci rekordow w bazie
RETURNS int
AS
BEGIN
  DECLARE @num int;
  SELECT @num=COUNT(*) FROM WYCIAGI WHERE data BETWEEN @od and @do
  RETURN @num
END
GO


IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'GET_SUM_UZN')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.GET_SUM_UZN () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.GET_SUM_UZN(@od date, @do date)
--pobranie sumy uznan
RETURNS nvarchar(20)
AS
BEGIN
  DECLARE @sum DECIMAL(19,4);
  SELECT @sum=COALESCE(SUM(kwota),0) FROM WYCIAGI WHERE (typ = 1) and (data
BETWEEN @od and @do)
  RETURN RTRIM(LTRIM(STR(@sum,18,2)))
END
GO



IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'GET_SUM_OBC')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.GET_SUM_OBC () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.GET_SUM_OBC(@od date, @do date)
--pobranie sumy obciazen
```

```
RETURNS nvarchar(20)
AS
BEGIN
  DECLARE @sum DECIMAL(19,4);
  SELECT @sum=COALESCE(SUM(kwota),0) FROM WYCIAGI WHERE (typ = 0) and (data
BETWEEN @od and @do)
  RETURN RTRIM(LTRIM(REPLACE(STR(@sum,18,2),'-','')))
END
GO

IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'T_ZNAKOWY')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.T_ZNAKOWY () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.T_ZNAKOWY(@msg nvarchar(256) )
/* wyczyść pole tekstowe z wrażliwych znaków o dlugosci 256 znakow*/
RETURNS nvarchar(256)
AS
BEGIN
  IF (@msg IS NULL)  OR (RTRIM(@msg) = N'')
    RETURN N''

  SET @msg = LTRIM(RTRIM(@msg))
  /* clear potentially dangerous characters for XML within the string */
  SET @msg = REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(@msg,'\n',N'
'),N'<',N'?'),N'>','?'),N':',N'?'),N'\',N'?'')
  SET @msg = REPLACE(@msg,N'/',N'!')

  RETURN RTRIM(LEFT(@msg,255))
END
GO


IF NOT EXISTS
  ( SELECT 1
    FROM sysobjects o
    WHERE (o.name = 'NAME_TOKEN')
    AND   (o.xtype = 'FN')
  )
  BEGIN
    DECLARE @sql nvarchar(500)
    SET @sql = 'CREATE FUNCTION dbo.NAME_TOKEN () returns money AS begin
return 0 end '
    EXEC sp_sqlexec @sql
  END

GO

ALTER FUNCTION dbo.NAME_TOKEN(@msg nvarchar(256) )
/* wyczyść pole tekstowe z wrażliwych znaków o dlugosci 240 znakow*/
```

```
RETURNS nvarchar(240)
AS
BEGIN
  IF (@msg IS NULL)  OR (RTRIM(@msg) = N'')
    RETURN N''

  SET @msg = LTRIM(RTRIM(@msg))
  /* clear potentially dangerous characters for XML within the string */
  SET @msg = REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(@msg,'\n',N'
'),N'<',N'?'),N'>','?'),N':',N'?'),N'\',N'?'')
  SET @msg = REPLACE(@msg,N'/',N'!')

  RETURN RTRIM(LEFT(@msg,239))
END
GO
```

4.3.4. PROCEDURA GENERUJĄCA JPK_WB

```
USE DB_JPK_WB
GO

IF NOT EXISTS
( SELECT 1
    from sysobjects o (NOLOCK)
    WHERE (o.[name] = 'GEN_JPK')
    AND   (OBJECTPROPERTY(o.[ID],'IsProcedure')=1)
)
BEGIN
  DECLARE @stmt nvarchar(100)
  SET @stmt = 'CREATE PROCEDURE dbo.GEN_JPK AS '
  EXEC sp_sqlexec @stmt
END
GO


-- procedura generujaca plik jpk_wb
ALTER PROCEDURE dbo.GEN_JPK @od nvarchar(100)=NULL, @do nvarchar(100)=NULL,
@path nvarchar(100)
AS
  IF @od is NULL
  BEGIN
    SELECT @od = MIN(data) FROM dbo.WYCIAGI
  END

  IF @do is NULL
  BEGIN
    SELECT @do = MAX(data) FROM dbo.WYCIAGI
  END



  DECLARE @xml xml,
      @id int;

  SELECT TOP 1 @id = id_klienta FROM WYCIAGI --pobranie klucza glownego
klienta

   SET @xml = null

    ;WITH XMLNAMESPACES(N'http://jpk.mf.gov.pl/wzor/2019/09/27/09271/'
AS tns
     ,
N'http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/DefinicjeTypy
/' AS etd)

    select @xml =
    ( SELECT --naglowek
        (
        SELECT
          N'1-0'            AS [tns:KodFormularza/@wersjaSchemy]
        , N'JPK_WB'           AS [tns:KodFormularza/@kodSystemowy]
        , N'JPK_WB'                   AS [tns:KodFormularza]
        , N'1'                        AS [tns:WariantFormularza]
        , N'1'                        AS [tns:CelZlozenia]
        , GETDATE()                   AS [tns:DataWytworzeniaJPK]
```

```
        , dbo.CONVERT_DATE(@od)      AS [tns:DataOd]
        , dbo.CONVERT_DATE(@do)      AS [tns:DataDo]
        , s.kod_urzedu           AS [tns:KodUrzedu]

          FROM dbo.PODMIOTY (NOLOCK) s WHERE s.id = @id

        FOR XML PATH('tns:Naglowek'), TYPE)
    ,
        (SELECT
                (
        SELECT  --identyfikator podmiotu
          s.NIP            AS [etd:NIP]
                  , dbo.NAME_TOKEN(nazwa)   AS [etd:PelnaNazwa]
          , s.REGON          AS [etd:REGON]

                        FROM dbo.PODMIOTY (NOLOCK) s WHERE s.id = @id
                        FOR XML PATH('tns:IdentyfikatorPodmiotu'), TYPE
                )
                ,
                (
        SELECT  -- adres podmiotu
                N'PL'       AS [etd:KodKraju]
                        ,       s.Wojewodztwo       AS [etd:Wojewodztwo]
                        ,       s.Powiat            AS [etd:Powiat]
                        ,       s.gmina             AS [etd:Gmina]
                        ,       s.Ulica             AS [etd:Ulica]
                        ,       s.nr_domu       AS [etd:NrDomu]
                        ,       s.nr_lokalu     AS [etd:NrLokalu]
                        ,       s.Miejscowosc   AS [etd:Miejscowosc]
                        ,       s.kod_pocztowy    AS [etd:KodPocztowy]
            ,       s.poczta            AS [etd:Poczta]

                                FROM dbo.ADRESY (NOLOCK) s WHERE s.id =
(SELECT adres FROM dbo.PODMIOTY WHERE id = @id)

                        FOR XML PATH('tns:AdresPodmiotu'), TYPE
                )
        FOR XML PATH('tns:Podmiot1'), TYPE
        )
    ,
    ( SELECT
        (
        SELECT  --numer IBAN
          s.IBAN            AS [tns:NumerRachunku]

          FROM dbo.PODMIOTY (NOLOCK) s WHERE s.id = @id

        FOR XML PATH('tns:NumerRachunku'), TYPE)
    )
    ,
    ( SELECT
        (
        SELECT --salda
          dbo.GET_SALDO_POCZ (@od)        AS [tns:SaldoPoczatkowe]
        , dbo.GET_SALDO_KON (@do)       AS [tns:SaldoKoncowe]

        FOR XML PATH('tns:Salda'), TYPE)
    )
    ,
    ( SELECT
        (
```

```
        SELECT --historia transakcji
          RANK() OVER (ORDER BY s.id)    AS [tns:NumerWiersza]
        , dbo.CONVERT_DATE(s.data)       AS [tns:DataOperacji]
        , dbo.T_ZNAKOWY(s.nazwa)       AS [tns:NazwaPodmiotu]
        , dbo.T_ZNAKOWY(s.opis)        AS [tns:OpisOperacji]
        , dbo.CONVERT_MONEY(s.kwota)     AS [tns:KwotaOperacji]
        , dbo.CONVERT_MONEY(s.saldo_po)    AS [tns:SaldoOperacji]

          FROM WYCIAGI (NOLOCK) s WHERE data BETWEEN @od and @do
        FOR XML PATH('tns:WyciagWiersz'), TYPE)

    )
    ,
    ( SELECT
        (
        SELECT --wyciag kontrolny
          dbo.GET_ROW_NUM (@od, @do)       AS [tns:LiczbaWierszy]
        , dbo.GET_SUM_OBC (@od, @do)       AS [tns:SumaObciazen]
        , dbo.GET_SUM_UZN (@od, @do)       AS [tns:SumaUznan]

        FOR XML PATH('tns:WyciagCtrl'), TYPE)

    )
    FOR XML PATH(''), TYPE, ROOT('tns:JPK')
        )

    SET @xml.modify('declare namespace tns =
"http://jpk.mf.gov.pl/wzor/2019/09/27/09271/"; insert attribute
xsi:schemaLocation{"http://jpk.mf.gov.pl/wzor/2019/09/27/09271/
schema.xsd"} as last into (tns:JPK)[1]')


    INSERT INTO LOGI VALUES(@id, 'Wygenerowanie pliku JPK_WB',
SYSDATETIME(), 0)

    --tabela tymczasowa potrzebna do zapisu xml na dysku
    DROP TABLE IF EXISTS ##TEMP_TABLE
    SELECT @xml as xml INTO ##TEMP_TABLE

    --SELECT * FROM ##TEMP_TABLE

    DECLARE @sql nvarchar(500)
    SET @sql = 'bcp "SELECT xml FROM ##TEMP_TABLE" queryout
"'+@path+'JPK_WB.xml" -T -c -t,'

    EXEC xp_cmdshell @sql

 GO

 /*
 zmiany konfigu serwera w celu umozlienia zapisu przez xp_cmdshell
 EXEC master.dbo.sp_configure 'show advanced options', 1
RECONFIGURE
EXEC master.dbo.sp_configure 'xp_cmdshell', 1
RECONFIGURE
*/

 EXEC dbo.GEN_JPK @path='C:\tmp\'
```

4.4. WYNIK

```
<tns:JPK

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Definicje
Typy/"
  xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/
schema.xsd">
  <tns:Naglowek

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Definicje
Typy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:KodFormularza wersjaSchemy="1-0"
kodSystemowy="JPK_WB">JPK_WB</tns:KodFormularza>
    <tns:WariantFormularza>1</tns:WariantFormularza>
    <tns:CelZlozenia>1</tns:CelZlozenia>
    <tns:DataWytworzeniaJPK>2022-05-
07T15:25:17.920</tns:DataWytworzeniaJPK>
    <tns:DataOd>2022-02-10</tns:DataOd>
    <tns:DataDo>2022-02-12</tns:DataDo>
    <tns:KodUrzedu>0202</tns:KodUrzedu>
  </tns:Naglowek>
  <tns:Podmiot1

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Definicje
Typy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:IdentyfikatorPodmiotu

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Definicje
Typy/"
      xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
      <etd:NIP>4058195881</etd:NIP>
      <etd:PelnaNazwa>podmiot1</etd:PelnaNazwa>
      <etd:REGON>137318807</etd:REGON>
    </tns:IdentyfikatorPodmiotu>
    <tns:AdresPodmiotu

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Definicje
Typy/"
      xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
      <etd:KodKraju>PL</etd:KodKraju>
      <etd:Wojewodztwo>slaskie</etd:Wojewodztwo>
      <etd:Powiat>lubliniecki</etd:Powiat>
      <etd:Gmina>lubliniec</etd:Gmina>
      <etd:Ulica>lipowa</etd:Ulica>
      <etd:NrDomu>14</etd:NrDomu>
      <etd:NrLokalu/>
      <etd:Miejscowosc>lubliniec</etd:Miejscowosc>
      <etd:KodPocztowy>42-700</etd:KodPocztowy>
      <etd:Poczta>lubliniec</etd:Poczta>
    </tns:AdresPodmiotu>
  </tns:Podmiot1>
  <tns:NumerRachunku
```

```
xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:NumerRachunku>PL25109024028713951641489122</tns:NumerRachunku>
  </tns:NumerRachunku>
  <tns:Salda

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:SaldoPoczatkowe>125.00</tns:SaldoPoczatkowe>
    <tns:SaldoKoncowe>123.00</tns:SaldoKoncowe>
  </tns:Salda>
  <tns:WyciagWiersz

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:NumerWiersza>1</tns:NumerWiersza>
    <tns:DataOperacji>2022-02-10</tns:DataOperacji>
    <tns:NazwaPodmiotu>aaaa</tns:NazwaPodmiotu>
    <tns:OpisOperacji>piwo</tns:OpisOperacji>
    <tns:KwotaOperacji>38.00</tns:KwotaOperacji>
    <tns:SaldoOperacji>163.00</tns:SaldoOperacji>
  </tns:WyciagWiersz>
  <tns:WyciagWiersz

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:NumerWiersza>2</tns:NumerWiersza>
    <tns:DataOperacji>2022-02-10</tns:DataOperacji>
    <tns:NazwaPodmiotu>bbbb</tns:NazwaPodmiotu>
    <tns:OpisOperacji>sushi</tns:OpisOperacji>
    <tns:KwotaOperacji>100.00</tns:KwotaOperacji>
    <tns:SaldoOperacji>263.00</tns:SaldoOperacji>
  </tns:WyciagWiersz>
  <tns:WyciagWiersz

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:NumerWiersza>3</tns:NumerWiersza>
    <tns:DataOperacji>2022-02-11</tns:DataOperacji>
    <tns:NazwaPodmiotu>eeee</tns:NazwaPodmiotu>
    <tns:OpisOperacji>burgery</tns:OpisOperacji>
    <tns:KwotaOperacji>-22.00</tns:KwotaOperacji>
    <tns:SaldoOperacji>241.00</tns:SaldoOperacji>
  </tns:WyciagWiersz>
  <tns:WyciagWiersz

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:NumerWiersza>4</tns:NumerWiersza>
    <tns:DataOperacji>2022-02-11</tns:DataOperacji>
    <tns:NazwaPodmiotu>cccc</tns:NazwaPodmiotu>
    <tns:OpisOperacji>kaufland</tns:OpisOperacji>
    <tns:KwotaOperacji>-75.00</tns:KwotaOperacji>
    <tns:SaldoOperacji>165.00</tns:SaldoOperacji>
```

```
  </tns:WyciagWiersz>
  <tns:WyciagWiersz

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:NumerWiersza>5</tns:NumerWiersza>
    <tns:DataOperacji>2022-02-12</tns:DataOperacji>
    <tns:NazwaPodmiotu>dddd</tns:NazwaPodmiotu>
    <tns:OpisOperacji>fryzjer</tns:OpisOperacji>
    <tns:KwotaOperacji>-42.00</tns:KwotaOperacji>
    <tns:SaldoOperacji>123.00</tns:SaldoOperacji>
  </tns:WyciagWiersz>
  <tns:WyciagCtrl

xmlns:etd="http://crd.gov.pl/xml/schematy/dziedzinowe/mf/2018/08/24/eD/Defi
nicjeTypy/"
    xmlns:tns="http://jpk.mf.gov.pl/wzor/2019/09/27/09271/">
    <tns:LiczbaWierszy>5</tns:LiczbaWierszy>
    <tns:SumaObciazen>139.00</tns:SumaObciazen>
    <tns:SumaUznan>138.00</tns:SumaUznan>
  </tns:WyciagCtrl>
</tns:JPK>
```