



AKADEMIA GÓRNICZO-HUTNICZA

WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

BAZY DANYCH II

---

# Aplikacja obsługująca dane przestrzenne dwuwymiarowe

---

*Autor:*  
Maciej Kubicki

*Prowadzący:*  
mgr inż. Andrzej Lemański

27 grudnia 2016

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
1.1	Temat projektu . . . . .	2
1.2	Realizacja . . . . .	2
<b>2</b>	<b>Projekt</b>	<b>3</b>
2.1	Korzystanie z aplikacji - UDT . . . . .	3
2.2	UDT - poprawność danych . . . . .	5
2.3	Korzystanie z aplikacji - UDF . . . . .	6

# 1 Wstęp

## 1.1 Temat projektu

Opracować aplikację obsługującą dane przestrzenne dwuwymiarowe. Należy opracować własne typy UDT i funkcje wyznaczające odległość pomiędzy punktami i sprawdzającą czy punkt należy do danego obszaru.

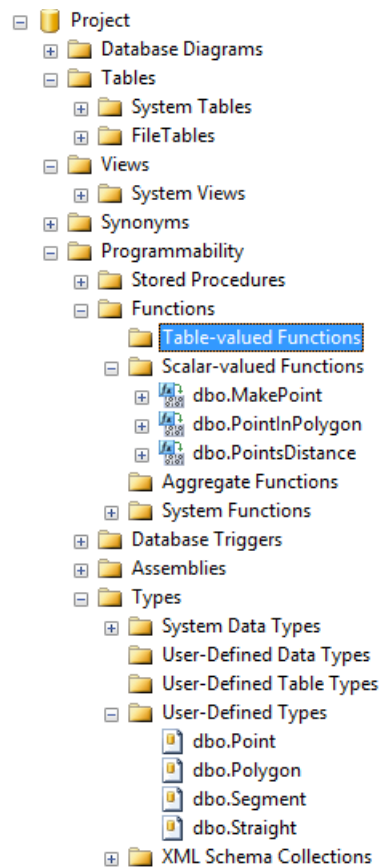
## 1.2 Realizacja

Do realizacji projektu użyłem środowiska CLR, aplikacja została stworzona przy pomocy platformy .NET z wykorzystaniem C#. Projekt został stworzony w Visual Studio 2015 i SqlServer 2014. W ramach projektu powstały typy:

- Point - reprezentujący punkt w przestrzeni dwuwymiarowej,
- Segment - reprezentujący odcinek w przestrzeni dwuwymiarowej,
- Polygon - reprezentujący wielokąt w przestrzeni dwuwymiarowej,
- Straight - reprezentujący prostą w przestrzeni dwuwymiarowej.

Powyższe typy dostarczają również zestaw funkcji działających na tych typach. Między innymi asString() dla wszystkich typów, GetX(), GetY(), MakePoint(x,y) dla Point, GetA(), GetB() dla Straight. Powstały również UDF:

- PointInPolygon(polygon, point) - funkcja sprawdzająca czy zadany punkt znajduje się wewnątrz wielokąta
- PointsDistance(point1, point2) - funkcja wyznaczająca odległość między punktami.



Rysunek 1: Projekt umieszczony na sqlserverze

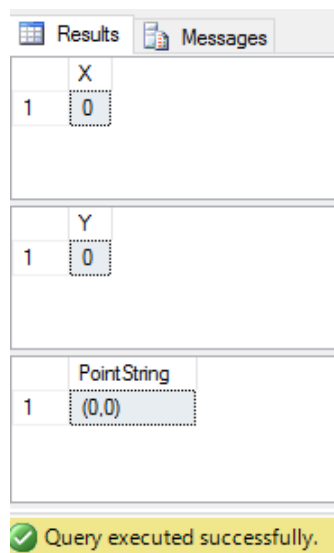
## 2 Projekt

### 2.1 Korzystanie z aplikacji - UDT

Po uruchomieniu projektu i użyciu deploy, jesteśmy w stanie korzystać z powyższej funkcjonalności na SqlServerze. Poniżej przedstawiam przykładowe wykorzystanie typu Point, który może zostać przypisany na dwa sposoby - przy pomocy funkcji MakePoint(), oraz przypisując string w formie '(x,y)'.

```
1 declare @point [dbo].[Point];
2 declare @point1 [dbo].[Point];
3 declare @point2 [dbo].[Point];
4 declare @point3 [dbo].[Point];
5 declare @point4 [dbo].[Point];
6 set @point = '(0,0)';
7 set @point1 = [dbo].[MakePoint](0,2);
8 set @point2 = [dbo].[MakePoint](2,2);
9 set @point3 = [dbo].[MakePoint](2,0);
10 set @point4 = [dbo].[MakePoint](1,4);
11 select @point.GetX() as 'X';
12 select @point.GetY() as 'Y';
13 select @point.asString() as 'PointString';
```

Oto otrzymany wynik:



The screenshot shows the 'Results' pane of SQL Server Enterprise Manager. It displays three separate result sets for the same query. The first result set is titled 'X' and contains one row with the value '0'. The second result set is titled 'Y' and contains one row with the value '0'. The third result set is titled 'PointString' and contains one row with the value '(0,0)'. At the bottom, a green status bar indicates 'Query executed successfully.'

	X
1	0

	Y
1	0

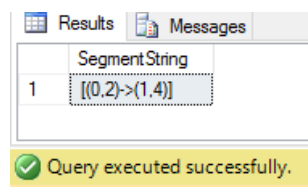
  

	PointString
1	(0,0)

Poniżej prezentuje użycie typu Segment, który definiuję przy pomocy stringu w postaci 'point1;point2'.

```
1 declare @seg [dbo].[Segment];
2 set @seg = ''+@point1.asString()+';'+@point4.asString();
3 select @seg.asString();
```

Oto otrzymany wynik:



The screenshot shows the 'Results' pane of SQL Server Enterprise Manager. It displays a single result set titled 'SegmentString' with one row containing the value '[(0,2)->(1,4)]'. At the bottom, a green status bar indicates 'Query executed successfully.'

	SegmentString
1	[(0,2)->(1,4)]

Kolejny przykład pokazuje użycie typu Straight, definiujemy go przy pomocy stringu '(a,b)'.

```
1 declare @stra [dbo].[Straight];
2 set @stra = '(1.2,4)';
3 select @stra.asString() as StraightEquation;
4 select @stra.GetA() as A;
5 select @stra.GetB() as B;
```

Oto otrzymany wynik:

Results		Messages	
StraightEquation			
1	y = 1,2x + 4		
	A		
1	1.2		
	B		
1	4		
Query executed successfully.			

Poniżej przedstawiam użycie typu Polygon, który jest definiowany przy pomocy stringu w postaci 'point1;point2;...;pointN'.

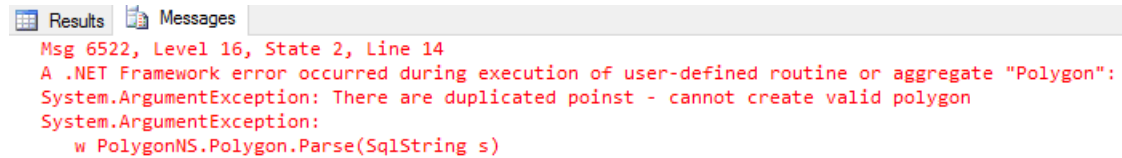
```
1 declare @poly [dbo].[Polygon]
2 set @poly=''+@point.asString()+';'+@point1.asString()+';'+@point2.
   asString()+';'+@point3.asString();
3 select @poly.NumberOfVerts() AS 'Number of Verts';
4 select @poly.asString() AS 'PolygonString';
```

Oto otrzymany wynik:

Results		Messages	
	Number of Verts		
1	4		
	PolygonString		
1	[(0,0),(0,2),(2,2),(2,0)]		
Query executed successfully.			

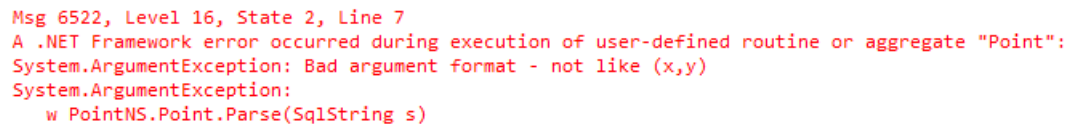
## 2.2 UDT - poprawność danych

Wszystkie typy mają walidację podanych danych przez użytkownika, przy definicji przy pomocy stringu jego format musi być odpowiedni, dodatkowo dla typu Segment i Polygon sprawdzane jest czy nie powtarzają się punkty, a dla typu Straight czy a i b nie jest 0 jednocześnie. Natomiast dla typu Polygon sprawdzane jest czy żadna krawędź się nie przecina ze sobą - czy da się utworzyć wielokąt z podanej kolejności punktów. W razie podania nie prawidłowych danych użytkownik zobaczy błąd(wyjątek). Oto przykładowe błędy:



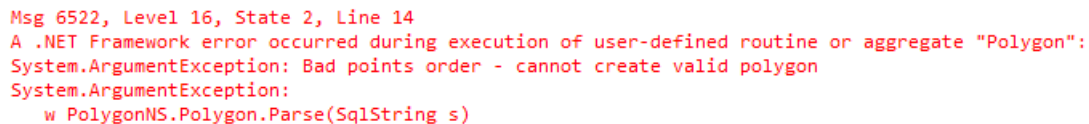
The screenshot shows the 'Messages' pane in SQL Server Enterprise Manager. It displays an error message from the .NET Framework. The message text is: 'Msg 6522, Level 16, State 2, Line 14 A .NET Framework error occurred during execution of user-defined routine or aggregate "Polygon": System.ArgumentException: There are duplicated pointst - cannot create valid polygon System.ArgumentException: w PolygonNS.Polygon.Parse(SqlString s)'. The error is related to a duplicate point in a polygon definition.

Rysunek 2: Powtórzone punkty w wielokącie



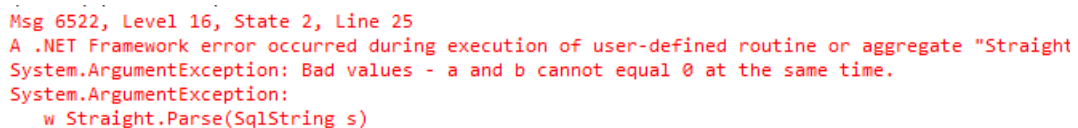
The screenshot shows the 'Messages' pane in SQL Server Enterprise Manager. It displays an error message from the .NET Framework. The message text is: 'Msg 6522, Level 16, State 2, Line 7 A .NET Framework error occurred during execution of user-defined routine or aggregate "Point": System.ArgumentException: Bad argument format - not like (x,y) System.ArgumentException: w PointNS.Point.Parse(SqlString s)'. The error is related to an incorrect argument format for a point definition.

Rysunek 3: Zły format stringu przy definicji punktu



The screenshot shows the 'Messages' pane in SQL Server Enterprise Manager. It displays an error message from the .NET Framework. The message text is: 'Msg 6522, Level 16, State 2, Line 14 A .NET Framework error occurred during execution of user-defined routine or aggregate "Polygon": System.ArgumentException: Bad points order - cannot create valid polygon System.ArgumentException: w PolygonNS.Polygon.Parse(SqlString s)'. The error is related to a bad order of points in a polygon definition.

Rysunek 4: Z podanej kolejności punktów nie da się stworzyć wielokąta



The screenshot shows the 'Messages' pane in SQL Server Enterprise Manager. It displays an error message from the .NET Framework. The message text is: 'Msg 6522, Level 16, State 2, Line 25 A .NET Framework error occurred during execution of user-defined routine or aggregate "Straight": System.ArgumentException: Bad values - a and b cannot equal 0 at the same time. System.ArgumentException: w Straight.Parse(SqlString s)'. The error is related to both 'a' and 'b' values being 0 for a straight line definition.

Rysunek 5: a i b równe 0

## 2.3 Korzystanie z aplikacji - UDF

Temat projektu zakładał stworzenie funkcji do wyznaczania odległości między dwoma punktami - prosty wzór matematyczny:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Druga funkcja miała sprawdzać czy dany punkt zawiera się w danym obszarze. Do implementacji tej funkcji posłużyłem się algorytmem zgodnie z którym półprosta utworzona z danego punktu w dowolnym kierunku jeśli punkt jest w obszarze to przecina granice tego obszaru nieparzystą ilość razy, jeśli punkt jest poza obszarem to przecina granice tego obszaru (boki wielokąta) parzystą ilość razy. W tym algorytmie należało rozważyć między innymi sytuacje kiedy punkt jest na granicy obszaru oraz jak półprosta zawiera bok wielokąta; Użycie powyższych UDF:

```
1 declare @point [dbo].[Point];
2 declare @point1 [dbo].[Point];
3 declare @point2 [dbo].[Point];
4 declare @point3 [dbo].[Point];
5 declare @point4 [dbo].[Point];
6 set @point = '(0,0)';
7 set @point1 = [dbo].[MakePoint](0,2);
8 set @point2 = [dbo].[MakePoint](2,2);
9 set @point3 = [dbo].[MakePoint](2,0);
10 set @point4 = [dbo].[MakePoint](1,4);
11 select [dbo].[PointsDistance](@point, @point4) AS 'Distance between points'
12 declare @poly [dbo].[Polygon]
13 set @poly = '' + @point.asString() + ',' + @point1.asString() + ',' + @point2.asString() + ',' + @point3.asString() + ',' + @point4.asString() + ',' + @point.asString();
14 select [dbo].[PointInPolygon](@poly, @point);
15 select [dbo].[PointInPolygon](@poly, '(2.1,0)');
16 select [dbo].[PointInPolygon](@poly, '(0,2.1)');
17 select [dbo].[PointInPolygon](@poly, '(3,2)');
18 select [dbo].[PointInPolygon](@poly, '(2,3)');
19 select [dbo].[PointInPolygon](@poly, '(-2,2.1)');
20 select [dbo].[PointInPolygon](@poly, '(1,1)');
```

Results		Messages
Distance between points		
1	4.12310562561766	
(No column name)		
1	1	
(No column name)		
1	0	
(No column name)		
1	0	
(No column name)		
1	0	
(No column name)		
1	0	
(No column name)		
1	1	

Powyższe przykład prezentuję poprawność działania funkcji. Dla funkcji PointInPolygon 1 - prawda - punkt jest w danym obszarze, 0 - fałsz - punkt jest poza danym obszarem.