

Dokumentacja Projektu

Obsługa danych przestrzennych dwuwymiarowych z wykorzystaniem własnych typów danych w MS SQL SERVER

Adam Łaba

19 czerwca 2022

1. Wstęp

Projekt polega na przygotowaniu oraz implementacji API umożliwiającego obsługę i przetwarzanie dwuwymiarowych danych przestrzennych przy wykorzystaniu CLR UDT. Wewnątrz projektu znajdują się także testy API, oraz przykładowa aplikacja konsolowa, która korzysta z API.

2. Opis funkcjonalności API

Zaprojektowane API pozwala na:

- Dodawanie punktów przestrzennych dwuwymiarowych
- Wyznaczanie odległości między dwoma punktami
- Dodawanie wielokątów dwuwymiarowych
- Obliczenia pola powierzchni wielokąta prostego
- Sprawdzenie, czy dany punkt znajduje się wewnątrz wielokąta

Zarówno punkt i wielokąt zostały zaimplementowane przy wykorzystaniu mechanizmu CLR UDT w języku C#.

Point

Jest klasą reprezentującą punkt w przestrzeni dwuwymiarowej. Posiada dwa pola:

`Double m_x`

`Double m_y`

Które reprezentują położenie punktu. Dodatkowo klasa zawiera metody i funkcje:

- `Point()` - konstruktor domyślny, ustawia pola na NaN
- `Point(double x, double y)` - konstruktor dwuargumentowy
- `string ToString()` - zwraca string w postaci "(x, y)"
- `Point Parse()` - parsuje stringa w postaci "x;y" na punkt
- `bool IsNull()` - zwraca informację, czy punkt jest Nullem
- `SqlDouble DistanceTo(Point p2)` - oblicza odległość do punktu

- `SqlDouble Distance(Point p1, Point p2)` - oblicza odległość między punktami

Oraz metody pozwalające na serializację danych.

Polygon

Reprezentuje wielokąt w przestrzeni dwuwymiarowej. Ma jedno pole:

`List<Point> m_points`

Które zawiera punkty, z których składa się wielokąt. Klasa zawiera metody i funkcje:

- `Polygon()` - konstruktor bezargumentowy
- `Polygon()` - konstruktor dwuargumentowy
- `bool IsNull()` - zwraca informację, czy wielokąt jest Nullem
- `string ToString()` - zwraca stringa w postaci "(x1, y1) - (x2, y2) -..."
- `Polygon Parse()` - parsuje stringa w postaci "x1;y1;x2;y2..." na wielokąt
- `bool onSegment(Point p1, Point p2, Point p3)` - pomocnicza funkcja, sprawdzająca czy punkt p2 znajduje się na odcinku p1 - p3
- `int orientation(Point p1, Point p2, Point p3)` - pomocnicza funkcja, znajdująca względną orientację między trzema punktami
- `bool edgesIntersect()` - pomocnicza funkcja sprawdzająca, czy dwie krawędzie się przecinają
- `bool IsSimple()` - metoda sprawdzająca, czy wielokąt jest prosty (nie zawiera żadnych przecinających się krawędzi)
- `SqlDouble Area()` - oblicza pole powierzchni wielokąta prostego
- `double CheckLeft(Point p1, Point p2, Point p3)` - pomocnicza funkcja, sprawdzająca względną pozycję trzech punktów
- `bool IsInside(Point p)` - sprawdza, czy dany punkt znajduje się wewnątrz prostokąta

oraz metody pozwalające na serializację danych.

3. Opis implementacji

Odległość między dwoma punktami jest obliczana w oparciu o wzór:

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

Współliniowość trzech punktów:

$$P = x_a(y_b - y_c) + x_b(y_c - y_a) + x_c(y_a - y_b)$$

Jeśli $P = 0$, to punkty leżą na jednej linii.

Pole wielokąta obliczane jest w oparciu o shoelace formula (algorytm sznurówki), który jest reprezentowany przez wzór:

$$P_{1, 2, \dots, n} = \frac{1}{2} \cdot \left(\begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + \dots + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix} \right)$$

4. Testy jednostkowe

Dla zdecydowanej większości metod oraz funkcji zostały napisane testy, sprawdzające ich działanie. Testy zostały podzielone na dwie klasy - testy dotyczące funkcji i metod klasy Point oraz Polygon. Mamy więc:

Point

- `YTest()` - sprawdza getter dla współrzędnej Y. Jest realizowany przez asercję `-3.3 = Point(-3, -3.3).Y`
- `XTest()` - sprawdza getter dla współrzędnej X. Jest realizowany przez asercję `2.5 = Point(2.5, 0).X`
- `NullTest()` - sprawdza Null. Jest realizowany przez dwie asercje `Double.NaN = Point().X`
`Double.NaN = Point().Y`
- `IsNullTest()` - sprawdza IsNull. Jest realizowany przez asercję `Point().IsNull = true`
- `ToStringTest()` - sprawdza metodę `ToString()`. Jest realizowany przez asercję `Point(3, 3).ToString() = "(3, 3)"`
- `DistanceToTest()` - sprawdza metodę `DistanceTo()`. Sprawdza, czy odległość między (1, 1) a (1, 5) jest równa 4.
- `DistanceTest()` - sprawdza, czy odległość między (5, 5) a (5, 10) jest równa 5
- Oraz dodatkowo testy obu konstruktorów

Polygon

- `NullTest()` - sprawdza Null
- `IsNullTest()` - sprawdza metodę `IsNull()`. Jest realizowany przez asercję `Polygon().IsNull() = false`
- `AddPointTest()` - sprawdza `AddPoint`
- `ToStringTest()` - sprawdza metodę `ToString()`
- `ParseTest()` - sprawdza funkcję `Parse()`
- `OtientationTest()` - sprawdza funkcję `orientation()`
- `onSegmentTest()` - sprawdza funkcję `onSegment()`
- `IsSimpleTest()` - sprawdza metodę `IsSimple()`

- `IsInsideTest()` - sprawdza, czy metoda informująca czy punkt znajduje się wewnątrz wielokąta działa poprawnie
- `edgesIntersectTest()` - sprawdza poprawne działanie metody sprawdzającej, czy krawędzie się przecinają
- `CheckLeftTest()` - sprawdza działanie metody `CheckLeft()`
- `AreaTest()` - sprawdza, czy obliczanie pola wielokąta działa poprawnie

Wszystkie testy przechodziły poprawnie - potwierdza to poniższy zrzut ekranu:

Test run completed Results: 22/22 passed; Item(s) checked: 0			
	Result	Test Name	Project
<input type="checkbox"/>	Passed	AreaTest	APITest
<input type="checkbox"/>	Passed	ToStringTest	APITest
<input type="checkbox"/>	Passed	IsInsideTest	APITest
<input type="checkbox"/>	Passed	IsNullTest	APITest
<input type="checkbox"/>	Passed	DistanceToTest	APITest
<input type="checkbox"/>	Passed	orientationTest	APITest
<input type="checkbox"/>	Passed	XTest	APITest
<input type="checkbox"/>	Passed	ParseTest	APITest
<input type="checkbox"/>	Passed	NullTest	APITest
<input type="checkbox"/>	Passed	PointConstructorTest1	APITest
<input type="checkbox"/>	Passed	IsSimpleTest	APITest
<input type="checkbox"/>	Passed	ParseTest	APITest
<input type="checkbox"/>	Passed	IsNullTest	APITest
<input type="checkbox"/>	Passed	NullTest	APITest
<input type="checkbox"/>	Passed	ToStringTest	APITest
<input type="checkbox"/>	Passed	onSegmentTest	APITest
<input type="checkbox"/>	Passed	PointConstructorTest	APITest
<input type="checkbox"/>	Passed	CheckLeftTest	APITest
<input type="checkbox"/>	Passed	AddPointTest	APITest
<input type="checkbox"/>	Passed	edgesIntersectTest	APITest
<input type="checkbox"/>	Passed	DistanceTest	APITest
<input type="checkbox"/>	Passed	YTest	APITest

5. Inne typy dostępne w SQL SERVER

SQL SERVER udostępnia typy:

- Point
- Polygon
- LineString
- CircularString
- CompoundCurve
- CurvePolygon
- MultiPoint
- MultiLineString
- MultiPolygon

- **GeometryCollection**

służące do reprezentacji danych przestrzennych dwuwymiarowych.

Porównując powyższe do zaimplementowanych samodzielnie typów rzuca się jedna różnica - Polygon dostępny w SQL SERVER nie wymaga istnienia instancji Point, w przeciwieństwie do Polygonu zbudowanego w oparciu o listę Pointów.

6. Podsumowanie, wnioski

Ten konkretny przypadek nie był optymalnym wykorzystaniem UDT, ponieważ typy oferujące tę samą funkcjonalność są już dostępne w SQL SERVER. Dodatkowo wykorzystywanie UDT nie jest zalecaną przez Microsoft funkcjonalnością, i powinno być wykorzystywane tylko w skrajnych przypadkach, jak np. dla danych geograficznych.

7. Literatura

<https://stackoverflow.com/>

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/>

https://en.wikipedia.org/wiki/Shoelace_formula