

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

Autor: Aleksander Kluczka

1. Opis projektu

Celem projektu jest przygotowanie API oraz jego implementacji, która umożliwi przetwarzanie danych dwuwymiarowych z wykorzystaniem własnych typów danych CLR UDT oraz metod.

2. Opis funkcjonalności API

Zaimplementowane API umożliwia następujące czynności:

- dodawanie dowolnej liczby punktów;
- dodawanie dowolnej liczby wielokątów prostych;
- usuwanie istniejących punktów;
- usuwanie istniejących wielokątów;
- obliczenie dystansu między dwoma podanymi punktami;
- obliczenie współczynników a oraz b z formuły $y = ax + b$ dla linii prostej przechodzącej przez dwa punkty podane przez użytkownika;
- sprawdzenie, czy podany punkt znajduje się na danym odcinku;
- sprawdzenie, czy podane trzy punkty są współliniowe;
- obliczenie pola danego wielokąta prostego;
- sprawdzenie liczby punktów będących wierzchołkami wielokąta prostego;
- sprawdzenie, czy podany punkt znajduje się wewnątrz danego wielokąta prostego.

Wspomniane punkty i wielokąty proste są typami danych zdefiniowanymi w języku C# i wykorzystywanymi w aplikacji oraz bazie danych. Są to dwie klasy:

2.1 Klasa Point

Jest to klasa reprezentująca punkt w przestrzeni dwuwymiarowej, która udostępnia następujące publiczne metody:

- konstruktor domyślny;

- konstruktor dwuargumentowy dla parametrów x oraz y;
- DistanceTo(Point other) - obliczanie odległości między dwoma punktami;
- CalcA(Point other) - obliczenie współczynnika nachylenia prostej przechodzącej przez dwa punkty;
- CalcB(Point other) - obliczenie punktu przecięcia z osią OY prostej przechodzącej przez dwa punkty;
- IsOnLine(Point two, Point three) - sprawdzenie czy dany punkt znajduje się na odcinku wymierzonym między dwoma innymi punktami;
- AreCollinear(Point two, Point three) - sprawdzenie czy dane trzy punkty są współliniowe.

2.2 Klasa Polygon

Jest to klasa reprezentująca wielokąt prosty znajdujący się na płaszczyźnie dwuwymiarowej. Polygon przedstawia następujące publiczne funkcje składowe:

- konstruktor domyślny;
- GetPointCount() - sprawdzenie liczby punktów należących do danego wielokąta;
- CalculateArea() - obliczenie pola danego wielokąta prostego. Jeśli wielokąt nie jest prosty, to zwracane jest 0.

2.3 Klasa Test

Dodatkowo stworzona została klasa Test przeznaczona do, jak sama nazwa wskazuje, testowania funkcjonalności API. Każda metoda prywatna znajdująca się w tej klasie odpowiada jednemu testowi. Lista testów jest następująca:

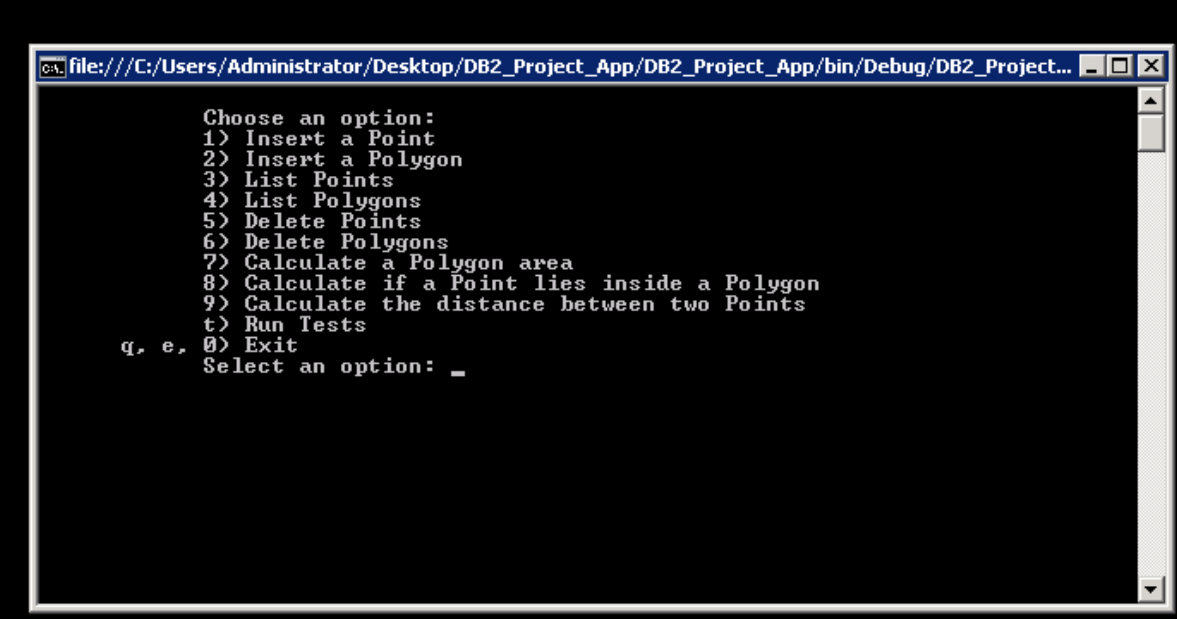
- TestPointDefaultConstructor();
- TestPointXyConstructor();
- TestPointParse();
- TestPointToString();
- TestPointDistanceTo();
- TestPointCalcA();
- TestPointCalcB();
- TestPointIsOnLine();
- TestPointAreCollinear();
- TestPolygonDefaultConstructor();
- TestPolygonNewPoint();
- TestPolygonGetPointCount();
- TestPolygonCalculateArea();
- TestPolygonIsCoordInside();
- TestPolygonIsPointInside().

Powyższe funkcje składowe wywoływane są przez publiczną metodę CallAllTests(). Interfejs użytkownika daje możliwość przeprowadzenia testów.

3. Opis interfejsu użytkownika

Funkcjonalność API jest dostępna w postaci aplikacji konsolowej napisanej w języku C#. Program łączy się z silnikiem bazy danych oraz wywołuje funkcje i procedury tam zdefiniowane. Wynik zwracany jest do aplikacji oraz wypisywany do konsoli. Implementacja bazuje na zdefiniowanych z pomocą interfejsu ADO.NET typach danych.

Po uruchomieniu programu, przed użytkownikiem pojawia się następujący interfejs:



```
file:///C:/Users/Administrator/Desktop/DB2_Project_App/DB2_Project_App/bin/Debug/DB2_Project...
Choose an option:
1> Insert a Point
2> Insert a Polygon
3> List Points
4> List Polygons
5> Delete Points
6> Delete Polygons
7> Calculate a Polygon area
8> Calculate if a Point lies inside a Polygon
9> Calculate the distance between two Points
t> Run Tests
q, e, 0> Exit
Select an option: _
```

Wylistowane zostały dostępne opcje:

- 1 - 9: wykorzystanie API bazy danych;
- t: uruchomienie testów jednostkowych;
- q, e, 0: wyjście z programu.

3.1 Uruchomienie

Żeby program działał poprawnie, użytkownik musi uruchomić program na maszynie wirtualnej udostępnionej na okres trwania zajęć laboratoryjnych. Należy uruchomić solucję Visual Studio nazwaną "DB2_Project_App" i w programie kliknąć przycisk "Start debugging" dla wersji Debug, albo "Start without debugging" dla wersji Release.

Bardzo ważne jest, aby zakończyć działanie programu poprawnie, to znaczy poprzez wybranie opcji "Exit" w menu głównym. Inne "niespodziewane" wyjście z aplikacji

może zepsuć konstrukcję tabel Point i Polygon. W takim przypadku należy ręcznie w bazie danych wywołać procedury: "EXEC DropPointsTable" oraz "EXEC DropPolygonsTable".

W ramach wyjaśnienia warto wspomnieć, że aplikacja użytkownika przygotowana jest na taką ewentualność i wykryje powyższy błąd, ale niestety środowisko Visual Studio nie pozwoli na jej uruchomienie.

4. Implementacja funkcjonalności

Niektóre funkcje składowe klas Point i Polygon stosują wzory matematyczne, które opisane zostały w niniejszym rozdziale.

4.1. Odległość między punktami

Wyrażana wzorem:

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

, gdzie zmienne x i y to koordynaty punktów 1 oraz 2.

4.2. Współczynnik kierunkowy funkcji liniowej

Obliczany za pomocą:

$$a = \frac{y_1 - y_2}{x_1 - x_2}$$

, gdzie zmienne x i y to koordynaty punktów 1 oraz 2.

4.3. Punkt przecięcia funkcji liniowej z osią OY

Jest to punkt, który można uzyskać z formuły:

$$b = y - a \cdot x$$

, gdzie x i y to koordynaty punktu leżącego na prostej, a a to jej współczynnik kierunkowy.

4.4. Współliniowość trzech punktów

Do sprawdzenia, czy punkty są współliniowe, użyto wzoru na pole trójkąta. Interesuje nas wyłącznie, czy pole to równa się zero, czy nie:

$$P_{1,2,3} = x_1 \cdot (y_2 - y_3) + x_2 \cdot (y_3 - y_1) + x_3 \cdot (y_1 - y_2)$$

Zmienne x i y to położenie punktów 1, 2 oraz 3.

4.5. Położenie punktu na odcinku

Najpierw sprawdzana jest możliwość, że punkty nie są współliniowe. Jeśli jest to prawda, zwracany jest natychmiastowo fałsz. W przeciwnym wypadku używany jest poniższy wzór logiczny:

$\min(x_1, x_3) \leq x_2 \leq \max(x_1, x_3) \text{ AND } \min(y_1, y_3) \leq y_2 \leq \max(y_1, y_3)$
, gdzie x i y to koordynaty punktów 1, 2 oraz 3.

4.6. Pole wielokąta prostego

Do obliczenia pola dowolnego wielokąta prostego wykorzystano tzw. shoelace algorithm (ang. "algorytm sznurówki"), którego wzór jest następujący:

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

, gdzie x i y to położenia punktów 1, ..., n.

5. Wyniki testów

- TestPointDefaultConstructor():
sprawdzenie, czy domyślnie punkt położony jest w (0, 0), rezultat pomyślny;
- TestPointXyConstructor()
sprawdzenie, czy konstruktor dwuargumentowy poprawnie ustawia położenie punktu w (x, y). Dla x = 5, y = -3 rezultat pomyślny;
- TestPointParse()
sprawdzenie poprawności konwersji typu String na typ Point. Dla "13,7;-9,1" rezultat pomyślny;
- TestPointToString()
sprawdzenie poprawności konwersji typu Point na typ String. Dla punktu w (0.001, -3.12) rezultat pomyślny;
- TestPointDistanceTo()
sprawdzenie, czy poprawnie liczona jest odległość między punktami. Dla punktów (3, 4) oraz (0, 0) rezultat pomyślny;
- TestPointCalcA()
sprawdzenie, czy współczynnik kierunkowy prostej obliczany jest poprawnie. Dla punktów (3, 3) oraz (11, 11) rezultat pomyślny;
- TestPointCalcB()

sprawdzenie, czy punkt przecięcia prostej z osią OY jest wyznaczany poprawnie. Dla punktów (3, 3) oraz (11, 3) rezultat pomyślny;

- `TestPointIsOnLine()`
sprawdzenie, czy dla punktu na odcinku zwracana jest prawda. Dla punktów (3, 3), (7, 7) i (9, 9) rezultat pomyślny;
- `TestPointAreCollinear()`
sprawdzenie, czy dla punktów współliniowych zwracana jest prawda. Dla punktów (3, 3), (7, 7) i (9, 9) rezultat pomyślny;
- `TestPolygonDefaultConstructor()`
sprawdzenie, czy domyślnie wielokąt ma pole równe 0 oraz nie posiada żadnych punktów. Rezultat pomyślny;
- `TestPolygonNewPoint()`
sprawdzenie, czy dodanie punktu zwiększy liczbę zwracaną przez `GetPointCount()`. Dla punktu (3.0, 7.1) rezultat pomyślny;
- `TestPolygonGetPointCount()`
sprawdzenie, czy po dodaniu kilku punktów liczba wierzchołków wielokąta jest prawidłowa. Dla trzech dodanych punktów rezultat pomyślny;
- `TestPolygonCalculateArea()`
sprawdzenie, czy pole wielokąta prostego obliczane jest prawidłowo. Dla wierzchołków (1, 1), (1, -1), (-1, -1) rezultat pomyślny;
- `TestPolygonIsCoordInside()`
sprawdzenie, czy dla położenia w środku wielokąta zwraca jest prawda. Dla wierzchołków (1, 1), (1, -1), (-1, -1) i (-1, 1) rezultat pomyślny;
- `TestPolygonIsPointInside()`
sprawdzenie, czy dla punktu w środku wielokąta zwraca jest prawda. Dla wierzchołków (1, 1), (1, -1), (-1, -1) i (-1, 1) rezultat pomyślny.

6. Typy danych przestrzennych w SQL SERVER

W ramach funkcjonalności .NET CLR, SQL SERVER udostępnia typy danych przestrzennych, spośród niektóre zostały przedstawione poniżej:

- `Polygon`;
- `Point`;
- `LineString`;
- `CircularString`;
- `CompoundCurve`;
- `CurvePolygon`;
- `MultiPoint`;
- `MultiLineString`;
- `MultiPolygon`;
- `GeometryCollection`.

Z największych różnic należy wspomnieć, że przykładowo do stworzenia instancji Polygon w SQL SERVER nie jest wymagana instancja typu Point, co ma miejsce w tym projekcie.

7. Podsumowanie

W ramach treningu, definiowanie własnych typów danych jest świetnym zajęciem. Należy jednak przytoczyć opinię Microsoft, że nie jest to optymalne rozwiązanie do codziennych zastosowań. Tej natury typy wykorzystywane są głównie do reprezentacji danych geograficznych, dlatego dla przeciętnego użytkownika nie są one potrzebne.

Kod źródłowy, zarówno aplikacji w C# jak i skrypty SQL zostały zamieszczone w archiwum zip wraz z dokumentacją. Alternatywnie pliki te znajdują się w repozytorium GitHub: https://github.com/vis4rd/db2_project_2022.

8. Literatura

<https://www.wikihow.com/Calculate-the-Area-of-a-Polygon>

<https://stackoverflow.com/>

<https://docs.microsoft.com/en-us/sql/?view=sql-server-ver16>