Krzysztof Łukaszczyk
Wydział Informatyki i Zarządzania
Wyższej Szkoły Informatyki w Łodzi

Promotor: dr hab. Adam Pelikant, prof. WSInf

ZASTOSOWANIE DANYCH SPATIAL DO OPISU ELEMENTÓW POWIERZCHNIOWYCH Z ZASTOSOWANIEM MS SQL SERVER 2008

Streszczenie – Głównym tematem artykułu jest stworzenie systemu wspierającego proces wyszukiwania obiektów płaskich na mapie oraz wprowadzania danych o takich obiektach i składowania ich po stronie serwera bazy danych. Wykorzystane zostały w tym celu wbudowane typy złożone Spatial wraz z ich metodami. W warstwie aplikacyjnej zastosowano środowisko programistyczne .NET oraz Google Maps jako narzędzie graficznej prezentacji wyników oraz wprowadzania danych.

1 Wstęp

Obecnie jesteśmy świadkami gwałtownego rozwoju systemów informatycznych, w których stosuje się metody odwzorowywania danych przestrzennych na różnorodnych mapach. Poczynając od prostych aplikacji na urządzenia mobilne pokazujące punkty na mapie, poprzez systemy nawigacyjne wyznaczające trasy, aż po rozbudowane systemy kartograficzne gromadzące w swoich bazach dane o położeniu obiektów - wszędzie tam istnieje konieczność użycia systemów informacji geograficznej (ang. Geographical Information System – GIS).

Istnieje na świecie wiele rozwiązań systemów GIS, przy czym te najpopularniejsze opierają się na zasadzie webowych usług informacyjnych, przez co użytkownik przy pomocy prostego interfejsu zwykłej przeglądarki internetowej uzyskuje dostęp do baz danych oraz zaawansowanych funkcji przestrzennych systemu GIS. Tak dynamiczny rozwój systemów tych spowodował to, że każdy liczący się na rynku producent platform bazodanowych zmuszony został do implementacji szeregu metod odwzorowujących dane przestrzenne w swoich produktach.

Różnorodność systemów GIS oraz platform bazodanowych do ich obsługi spowodowała konieczność opracowania jednego wspólnego standardu umożliwiającego wzajemną kompatybilność systemów

informatycznych. Takie standardy definiuje międzynarodowa organizacja OGC (ang. Open Geospatial Consortium) z siedzibą w Wayland Massachusetts utworzona w roku 1994. Jednym ze standardów specyfikacji opracowanych przez OGC jest między innymi zbiór metod oraz funkcjonalności niezależnych od języków programowania służących do opisywania obiektów geograficznych oraz manipulowania nimi. Także firma Microsoft kierując się wymaganiami rynku wraz ze swoim produktem SQL Server Spatial w wersji 2008 spełnia standardy OGC 1.1 dostarczając tym samym jedną z bardziej popularnych platform bazodanowych obok takich producentów jak Oracle czy PostgreSQL umożliwiającą tworzenie systemów GIS.

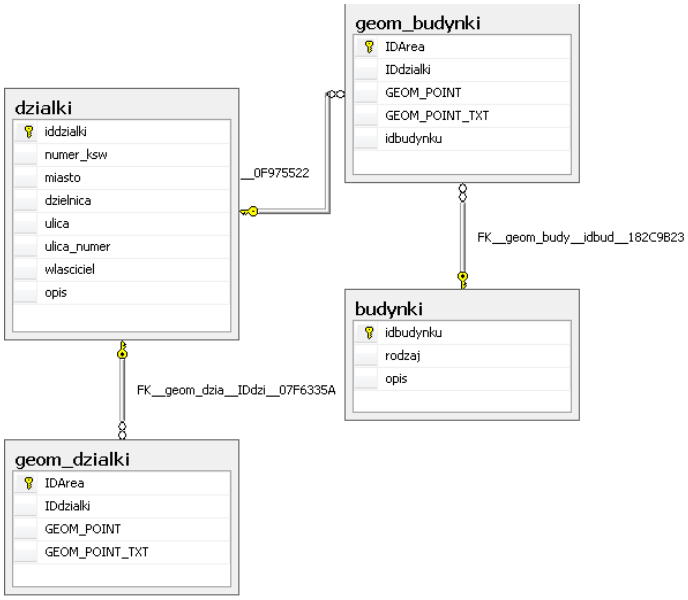
Wszystkie opisane powyżej aspekty oraz zainteresowanie praktycznym użyciem systemów GIS były podstawowym czynnikiem który zadecydował o wyborze zagadnień związanych z odwzorowywaniem danych przestrzennych jako tematu tej pracy. Wybierając produkt Microsoft SQL Server 2008 jako warstwę bazodanową zapewniono, spełnienie wszystkich współczesnych standardów OGC w dziedzinie opisywania obiektów geograficznych oraz integracje się z zastosowanym narzędziem programistycznym. Dodatkowo wybór rozwiązań Google Maps pozwolił na bardzo atrakcyjny i ciekawy sposób zdefiniowania warstwy prezentacyjnej.

W celu realizacji projektu opracowano i zrealizowano następujące elementy będące integralną częścią całego systemu:

- projekt całości systemu, w tym określenie wymagań dla systemu, wybór technologii oraz znalezienie atrakcyjnej z punktu widzenia użytkownika metody wizualnej prezentacji obiektów geometrycznych;
- zaprojektowanie struktury bazy danych w oparciu o serwer Microsoft SQL służącej do przechowywania danych zarówno danych przestrzennych, jak i danych informacyjnych dla całego systemu, w tym wybór typu obiektu geometrycznego przechowywanego w strukturach bazy danych;
- opracowanie zbioru funkcji zaimplementowanych w bazie danych niezbędnych do wykonywania operacji na obiektach geometrycznych oraz zbioru funkcji pomocniczych wykorzystywanych w prezentacji graficznej;
- opracowanie i implementacja elementów interfejsu programowania aplikacji (API) Google Maps w celu wizualizacji obiektów geometrycznych na mapie;
- zaprojektowanie warstwy prezentacji w postaci zestawu dwóch aplikacji napisanych w języku C# przy użyciu narzędzia Microsoft Visual Studio 2010 realizujących cele systemu;
- testowanie całości systemu oraz przygotowanie dokumentacji w postaci instrukcji dla użytkownika oraz administratora.

System zbudowany w oparciu o powyższe elementy posiada strukturę dwuwarstwową. Pierwszą warstwę stanowi baza danych wraz z jej funkcjami, drugą – warstwa prezentacji, która wykorzystując API Google powoduje, że system jest wizualnie atrakcyjny dla użytkownika. System Ewidencja gruntów posłużył z racji swojej specyfiki prezentacji jedynie kilku wybranych metod Spatial opisujących dane przestrzenne.

Na system Ewidencja gruntów składa się baza danych utworzona w narzędziu Microsoft SQL Server 2008 oraz dwie aplikacje zapewniające warstwę prezentacyjną systemu. Pierwsza z aplikacji służy do wizualizacji danych zgromadzonych w bazie danych SQL za pośrednictwem API Google Maps. Druga aplikacja jest aplikacją pomocniczą napisaną w celu wspomagania rejestrowania obiektów geograficznych w bazie danych. Baza danych systemu Ewidencja gruntów została w całości zaprojektowana oraz wykonana przy użyciu narzędzia Microsoft SQL Server 2008. Strukturę tablic systemu Ewidencja gruntów przedstawia rysunek nr 1 przedstawiony poniżej.



Rys. 1. Schemat relacyjny bazy danych systemu Ewidencja gruntów

System Ewidencja gruntów wykorzystuje do swego działania strukturę czterech tablic utworzonych w bazie danych. Są to tablice przechowujących dane geograficzne oraz ewidencyjne. Tablicami przechowującymi dane geograficzne są tablice: GEOM_DZIALKI oraz GEOM_BUDYNKI. Tablicami przeznaczonymi do przechowywania danych ewidencyjnych są tablice: DZIALKI oraz BUDYNKI.

Rozważając na etapie projektowania całego systemu metodę przechowywania danych geograficznych określono, że najprostszą metodą będzie przechowywanie danych każdego obiektu w postaci zbioru n-punktów o współrzędnych będących długością oraz szerokością geograficzną. Takie założenie było podyktowane głównie koniecznością dostosowania się do warstwy prezentacji wykorzystującej punkt jako podstawowy element składający się na obiekt budowany za pomocą interfejsu API Google Maps.

Tablica GEOM_DZIALKI przechowuje informacje o zbiorze punktów składających się na obiekt o nazwie działka w systemie, zaś tablica GEOM_BUDYNKI przechowuje informacje o zbiorze punktów składających się na obiekty o nazwie budynki będących częściami składowymi obiektu działka. Struktura tablicy GEOM_DZIALKI w postaci skryptu SQL wygląda następująco:

```
CREATE TABLE geom_dzialki
(IDArea INT IDENTITY(1,1) NOT NULL,
IDdzialki int,
GEOM_POINT geography,
GEOM_POINT_TXT AS GEOM_POINT.STAsText())
```

Kolumna GEOM_POINT typu geography przechowuje dane geograficzne punktów składających się na poszczególne działki w postaci binarnej, która jest trudna do interpretacji. W celu jej ułatwienia, wprowadzono ostatnią kolumnę o nazwie GEOM_POINT_TXT. Kolumna wykorzystuje w swojej definicji odwołanie AS GEOM_POINT.STAsText(), która zapewnia przy wykorzystaniu metody o tej samej nazwie, konwersję danych binarnych na format tekstowy. Kolumna została wprowadzona jedynie w celu łatwiejszej interpretacji danych zgromadzonych w tablicy, żadna z funkcji SQL lub mechanizmów prezentacji nie odwołuje się do jej wartości. Bliźniaczą względem tablicy GEOM_DZIALKI jest tablica GEOM_BUDYNKI. Tablica została zaprojektowana do przechowywania danych geograficznych opisujących budynki zlokalizowane na obszarach działek. Struktura tablicy jest analogiczna co struktura tablicy GEOM_DZIALKI. Jedyna różnica to dodatkowa kolumna IDBUDYNKU określająca rodzaj budynku. Struktura tablicy w postaci skryptu SQL wygląda następująco:

```
CREATE TABLE geom_budynki
(IDArea INT IDENTITY(1,1) NOT NULL,
IDdzialki int,
IdBudynku int,
GEOM_POINT geography,
GEOM_POINT_TXT AS GEOM_POINT.STAsText())
```

Tablica GEOM_BUDYNKI powiązana jest z pozostałymi tablicami tworzącymi system za pomocą następujących więzów integralności w postaci kluczy obcych:

```
ALTER TABLE geom_budynki ADD FOREIGN KEY (Iddzialki)
REFERENCES Dzialki(Iddzialki)
ALTER TABLE geom_budynki ADD FOREIGN KEY (Idbudynku)
REFERENCES Budynki(Idbudynku)
```

Pierwszy z kluczy zapewnia powiązanie kolumny IDDZIALKI z taka samą kolumną w tablicy GEOM_DZIALKI. W praktyce zatem zrealizowane zostało za jego pomocą powiązanie danych geometrycznych składających się na budynek z konkretnym obszarem geometrycznym stanowiącym działkę. Drugi, zapewnia powiązanie danych geometrycznych stanowiących budynek z konkretnym typem budynku umieszczonym w tablicy BUDYNKI pełniącej rolę tablicy ewidencyjnej. Pozostałe kolumny tablicy czyli GEOM_POINT i GEOM_POINT_TXT pełnią tę samą rolę co w opisanej powyżej tablicy GEOM_DZIALKI jednak odnoszą się do danych geograficznych budynków.

W systemie Ewidencja gruntów zostały zaprojektowane dwie tablice pełniące rolę tablic ewidencyjnych. Są to DZIALKI oraz BUDYNKI. Na dane ewidencyjne działki składają się informacje dotyczące np. numer księgi ewidencyjnej, dane adresowe, dane o właścicielu oraz opis. Tablica BUDYNKI przechowuje informacje o rodzaju budynku wraz z opisem.

2 Oprogramowanie po stronie serwera bazy danych

Na etapie projektowania aplikacji realizujących prezentację danych zgromadzonych w systemie Ewidencja gruntów zaistniała konieczność znacznego uproszczenia zapytań prezentujących dane ewidencyjne budynków w kontekście działki, w skład której one wchodzą. W związku z tym, utworzono widok w bazie danych który realizuję powyższą funkcjonalność. Struktura widoku V_BUDYNKI w postaci skryptu SQL wygląda następująco:

```
CREATE VIEW v_budynki
AS
SELECT gb.iddzialki, b.idbudynku, b.rodzaj FROM
geom_budynki gb
INNER JOIN budynki b
ON gb.idbudynku = b.idbudynku
GROUP BY gb.IDdzialki, b.idbudynku, b.rodzaj
```

W pierwszej fazie projektowania systemu Ewidencja gruntów założono maksymalne wykorzystanie funkcjonalności mechanizmów jakie daje sama baza danych MS SQL Server do obsługi danych przestrzennych Spatial. W związku z tym opracowano szereg funkcji SQL, które pozwalają na dokonywanie potrzebnych obliczeń i przekształceń obiektów geometrycznych oraz geograficznych na potrzeby ich prezentacji w aplikacji za pomocą interfejsu Google Maps API. Dodatkowo w trakcie projektowania samych aplikacji wyniknęła potrzeba uzupełnienia zbioru funkcji o funkcje pomocnicze dokonujące konwersji obiektów na potrzeby specyfiki warstwy prezentacyjnej. W związku z powyższym, funkcje SQL można podzielić na następujące dwie grupy:

1. Funkcje podstawowe
   - fDzialka2Polygon – odpowiada za utworzenie obiektu geometrycznego typu polygon z zestawu rekordów tablicy GEOM_DZIALKI, system przechowuje dane każdego obiektu geograficznego takiego jak działka czy budynek w postaci zbioru n-punktów typu POINT o współrzędnych będących długością oraz szerokością geograficzną przypisanych do konkretnego obiektu. Funkcja jest niezbędna z uwagi na konieczność budowania obiektów typu polygon na potrzeby dodatkowych przekształceń oraz zastosowań metod Spatial np. obliczających współrzędne środka figury. Funkcja zwraca obiekt będący obiektem geometrycznym Spatial typu polygon.
   - fDzialka2PolygonG – jest funkcją tożsama z funkcją Polygon, parametrem jej wywołania, tak jak w przypadku opisywanej wcześniej funkcji fDzialka2Polygon jest wartość typu INT będąca odpowiednikiem wartości pola IDDZIALKI w tablicy GEOM_DZIALKI. Różnicą jest natomiast zwracana wartość – w przypadku tej funkcji jest to obiekt geograficzny Spatial typu poligon; funkcja znajduje zastosowanie do tworzenia obiektów geograficznych typu poligon będących interpretacją obszaru działki, dla których będą dokonywane obliczenia takie jak powierzchnia oraz obwód.
   - fBudynek2Polygon – tworzy obiekt geometryczny typu polygon z zestawu rekordów tablicy GEOM_BUDYNKI; parametrami funkcji są dwie wartości całkowite INT odpowiadające wartościom kolumn IDDZIALKI oraz IDBUDYNKU tablicy GEOM_BUDYNKI; funkcja w połączeniu z jedną z funkcji pomocniczych znajduje zastosowanie w warstwie prezentacyjnej do obliczeń środków obiektów będących budynkami zlokalizowanymi na działkach.
   - fBudynek2PolygonG – jest funkcją tożsamą z funkcją fBudynek2Polygon; parametrami jej wywołania, tak jak w przypadku opisywanej wcześniej funkcji fBudynek2Polygon są wartości typu INT odpowiadające wartością pól IDDZIALKI i IDBUDYNKU

tablicy GEOM_BUDYNKI; różnicą jest natomiast zwracana wartość, którą jest to obiekt geograficzny Spatial typu poligon; funkcja znajduje zastosowanie do tworzenia obiektów geograficznych typu poligon będących interpretacją obszaru budynku na działce, dla którego będą dokonywane obliczenia takie jak powierzchnia oraz obwód.

- fPowierzchnia – jest funkcją wykorzystywaną do obliczania pola powierzchni dowolnego obszaru będącego obszarem geograficznym wykorzystując metodę Spatial o nazwie STArea(); funkcja zwraca policzoną wartość pola powierzchni jako wartość zmiennoprzecinkowa w jednostce metr kwadratowy; jedynym jej parametrem jest obiekt typu geography; który musi być prawidłowym obiektem geograficznym (wg. specyfikacji WTK) w przeciwnym wypadku funkcja zwróci błąd; funkcja nie wykonuje sprawdzenia prawidłowości obiektu geometrycznego podawanego w parametrze.
- fObwod – jest funkcją wykorzystywaną do obliczania obwodu dowolnego obszaru będącego obszarem geograficznym; do obliczeń wykorzystuje metodę Spatial o nazwie STLength();zwraca policzoną wartość obwodu jako liczbę zmiennoprzecinkowa w metrach; jedynym jej parametrem jest obiekt typu geography; tak jak w przypadku funkcji fPowierzchnia obiekt podany w parametrze musi być prawidłowym obiektem geograficznym (wg. specyfikacji WTK) w przeciwnym wypadku funkcja zwróci błąd; również nie wykonuje sprawdzenia prawidłowości obiektu geometrycznego podawanego w parametrze.

Praktyczną realizację funkcji przedstawmy na podstawie skryptu SQL tworzącego funkcję fDzialka2Polygon.

```
CREATE FUNCTION fDzialka2Polygon
(@IDDzialki int)
RETURNS geometry
AS
BEGIN
DECLARE @PointStr varchar(MAX);
DECLARE @result varchar(MAX);
DECLARE @Point geography
DECLARE @CoordinateStream varbinary(max)
DECLARE @i int
SET @i = 0
DECLARE GeomCursor CURSOR FOR SELECT GEOM_POINT FROM
geom_dzialki WITH (NOLOCK) WHERE IDdzialki =
@IDdzialki ;
OPEN GeomCursor;
FETCH NEXT FROM GeomCursor INTO @Point;
WHILE @@FETCH_STATUS = 0
```

```
BEGIN
IF (@i=0) BEGIN
SET @CoordinateStream =
SUBSTRING(@Point.STAsBinary(),6,16)
END
ELSE BEGIN
SET @CoordinateStream = @CoordinateStream +
SUBSTRING(@Point.STAsBinary(),6,16)
END;
SET @i = @i+1
FETCH NEXT FROM GeomCursor INTO @Point;
END;
CLOSE GeomCursor;
DEALLOCATE GeomCursor;
SET @PointStr = geometry::STGeomFromWKB
(
0x01
+ 0x03000000
+ 0x01000000
+ CONVERT(varbinary, REVERSE(CONVERT(varbinary,@i+1)))
+ @CoordinateStream +
SUBSTRING(@CoordinateStream,1,16)
,0).STAsText()
SET @result = @PointStr;
RETURN @result;
END;
```

Na przykładzie powyższego kodu omówmy poszczególne instrukcje składające się na całość funkcji. Początkowe linie kodu ciała funkcji oznaczają deklarację zmiennych na potrzeby operacji wykonywanych za pomocą dalszych instrukcji. Instrukcjami deklaracji są wszystkie instrukcje rozpoczynające się dyrektywą DECLARE, której parametrami są: nazwa zmiennej lokalnej, rozpoczynająca się obowiązkowo od znaku @ oraz drugi parametr będący jej typem danych. Dla typów danych znakowych varchar oraz binarnych varbinary określono dodatkowo w parametrze maksymalny ich rozmiar poprzez frazę max będącą reprezentacją 2^31-1 bajtów.

Kolejne instrukcje definiują i otwierają kursor dla zapytania SQL. Kursor w tym wypadku służy do zbudowania zestawu rekordów będących wartościami kolumny GEOM_POINT tablicy GEOM_DZIALKI. Na wierszach tych będą dokonywane kolejne operacje w funkcji. Deklaracja kursora wygląda następująco:

```
DECLARE GeomCursor CURSOR FOR SELECT GEOM_POINT FROM
geom_dzialki WITH (NOLOCK) WHERE IDdzialki =
@IDdzialki ;
OPEN GeomCursor;
```

Fraza WITH (NOLOCK) użyta w definicji kursora wymusza na serwerze SQL nie blokowanie rekordów tablicy GEOM_DZIALKI, na których działa kursor. Następna instrukcja:

```
FETCH NEXT FROM GeomCursor INTO @Point;
```

powoduje pobranie do uprzednio zdefiniowanej zmiennej przestrzennej typu geography o nazwie @POINT pierwszego rekordu kursora. W pętli WHILE dla wszystkich rekordów zwracanych przez kursor budowany jest ciąg znaków zdefiniowany jako zmienna lokalna typu binarnego CoordinateStream. Zmienna reprezentuje zbiór wierzchołków składających się na budowany w dalszej części funkcji obiekt przestrzenny. Przy czym, blok instrukcji warunkowej:

```
IF (@i=0) BEGIN
SET @CoordinateStream =
SUBSTRING(@Point.STAsBinary(),6,16)
END
```

odnosi się do pierwszego wierzchołka. Zmienna przestrzenna POINT opisuje współrzędne wierzchołka. Zmienna ta przedstawiona binarnie przyjmuje dokładnie 22 bajty tak jak na przykładzie poniżej:

```
SELECT TOP 1 GEOM_POINT.STAsBinary() FROM geom_dzialki
```

Wynik ma postać:

```
0x0101000000000000A356893340B3D7EEEC7AE54940
```

Na powyższym przykładzie widać wyraźnie, że zaczynając od 6 bajtu, kolejne 16 bajtów są bajtami odpowiadającymi za określanie współrzędnych punktu. Zatem budując ciąg znaków w zmiennej CoordinateStream te właśnie bajty trzeba wziąć pod uwagę. Realizowane jest to za pomocą instrukcji:

```
SUBSTRING(@Point.STAsBinary(),6,16)
```

Najważniejszym elementem funkcji fDzialka2Polygon jest instrukcja przedstawiona poniżej, pozwalająca z uprzednio utworzonego w pętli ciągu binarnego @CoordinateStream zbudować obiekt przestrzenny typu polygon.

```
SET @PointStr = geometry::STGeomFromWKB
(
0x01
```

```
+ 0x03000000
+ 0x01000000
+ CONVERT(varbinary,REVERSE(CONVERT(varbinary,@i+1)))
+ @CoordinateStream +
SUBSTRING(@CoordinateStream,1,16)
,0).STAsText()
```

Konwersja zmiennej binarnej @CoordinateStream realizowana jest za pomocą metody Spatial o nazwie STGeomFromWKB. Metoda pozwala na konwersję danych binarnych Well-Known Binary (WKB) na obiekty przestrzenne Spatial typu geography. Parametrami metody są: obiekt przestrzenny Spatial typu geography zapisany w formie binarnej, oraz tzw. SRID oznaczający Spatial reference ID.

Parametr pierwszy opisywanej metody w funkcji fDzialka2Polygon budowany jest jako połączenie zmiennej binarnej @CoordinateStream wraz z następującymi wartościami:

0x01 – oznaczający typ kodowania Big Endian;

0x03000000 – czterobajtowa wartość oznaczająca typ budowanego obiektu – w tym wypadku jest to polygon;

0x01000000 – ilość pierścieni wielokąta polygon – w tym wypadku wartość 1 oznacza jeden pierścień zewnętrzny.

W rezultacie, zmienna @PointStr przechowująca gotowy obiekt przestrzenny Spatial typu geography poddawana jest ostatniej już konwersji na postać tekstową przy zastosowaniu metody STAsText(). Ostatnie dwie instrukcje:

```
SET @result = @PointStr;
RETURN @result;
```

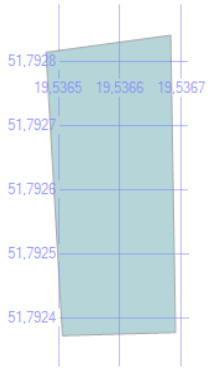
pozwalają na przygotowanie zmiennej @result zwracającej wynik funkcji. W tablicy GEOM_DZIALKI jest zdefiniowana działka będąca obszarem geograficznym składającym się z punktów będących poszczególnymi rekordami tablicy. Każdy z punktów jest obiektem geograficznym typu POINT o współrzędnych odpowiadających szerokości oraz długości geograficznej. Dane działki o IDDZIALKI =1 prezentuje zapytanie SQL poniżej:

```
SELECT iddzialki, geom_point_txt from geom_dzialki
WHERE IDdzialki=1
iddzialki geom_point_txt
1 POINT (19.536478221416473 51.792813889143325)
1 POINT (19.536505043506622 51.792372602313804)
1 POINT (19.53669011592865 51.792377579257014)
1 POINT (19.536682069301605 51.792840432574359)
```

Funkcja fDzialka2Polygon pozwala na przekształcenie zestawu powyższych rekordów w jeden obiekt typu polygon. Zapytanie SQL poniżej ilustruje zastosowanie funkcji dla opisywanego przykładu:

```
SELECT dbo.fDzialka2Polygon(1)
```

Wynikiem jest obiekt typu polygon prezentowany na ilustracji poniżej utworzony z poszczególnych punktów zapisanych jako rekordy tablicy GEOM_DZIALKI dla działki o IDDZIALKI =1, pokazany na rysunku 2.



Rys. 2. Graficzna prezentacja funkcji fDzialka2Polygon

Ponadto zdefiniowane zostały funkcje pomocnicze:

- fLatitude – jest jedną z dwóch bliźniaczych funkcji pomocniczych wykorzystywanych na potrzeby warstwy prezentacyjnej. Razem z funkcją fLongitude służy ona do odczytania z obiektu przestrzennego Spatial zdefiniowanego jako punt (point) jednej z jego współrzędnych – szerokości geograficznej. Parametrem funkcji jest obiekt geograficzny przekazany do funkcji po przekształceniu na postać tekstową w zmiennej typu VARCHAR.
- fLongitude – jest analogiczna do funkcji fLatitude, w odróżnieniu jednak od niej zwraca wartość będąca współrzędną długości geograficznej.
- fFloat2Varchar – realizuje jednocześnie dwa zadania. Pierwsze z nich to konwersja wartości zmiennoprzecinkowej float do postaci tekstowej przy uwzględnieniu założonej dokładności trzynastu znaków po przecinku. Drugie natomiast to zmiana separatora liczby zmiennoprzecinkowej z przecinka na kropkę. Utworzenie tej funkcji było konieczne ze względu na specyfikę interfejsu Google Maps API do prezentowania obiektów geograficznych. Funkcja jest przykładem funkcji dosyć sztywnej, jednak realizującej w poprawny sposób zakładane zadania. Funkcję tę można by zastąpić

całkowicie poprzez zastosowanie metod konwersji zmiennych środowiskowych dotyczących języka zaimplementowanych w aplikacji warstwy prezentacyjnej.

- fCenter – ma za zadanie znajdowanie współrzędnych środka dowolnego obiektu przestrzennego przekazanego w parametrze funkcji.

Przykładem realizacji jest skrypt tworzący funkcję fLatitude w postaci skryptu SQL:

```
CREATE FUNCTION fLatitude
(@point varchar(MAX))
RETURNS float
AS
BEGIN
DECLARE @latitude geography;
DECLARE @result float;
SET @latitude =
geography::STGeomFromText(@point,4326);
SET @result = @latitude.Lat;
RETURN @result;
END;
```

Przykład zastosowania funkcji do obliczenia wartości będącej współrzędną szerokości geograficznej ilustruje skrypt poniżej:

```
SELECT GEOM_POINT_TXT, dbo.fLongitude(geom_point_txt)
AS FLATITUDE
FROM geom_dzialki WHERE IDdzialki=1

GEOM_POINT_TXT FLATITUDE
POINT (19.536478221416473 51.792813889143325)
19,5364782214165
POINT (19.536505043506622 51.792372602313804)
19,5365050435066
POINT (19.53669011592865 51.792377579257014)
19,5366901159287
POINT (19.536682069301605 51.792840432574359)
19,5366820693016
```

Przykład prezentuje zastosowanie funkcji fLatitude obliczającej szerokość geograficzną z obiektu przestrzennego typu point przekazanego jako parametr do funkcji w formie tekstowej.
Struktura funkcji fCenter w postaci skryptu SQL wygląda następująco:

```
CREATE FUNCTION fCenter
(@polygon geometry)
```

```
RETURNS varchar(MAX)
AS
BEGIN
DECLARE @center geometry;
DECLARE @result varchar(MAX)
SET @result = @polygon.STCentroid().ToString();
RETURN @result;
END;
```

Parametrem funkcji jest dowolny obiekt przestrzenny Spatial, dla którego możliwe jest określenie współrzędnych środka przy zastosowaniu metody STCentroid(). Funkcja zwraca środek obiektu w postaci obiektu typu point przekształconego na postać tekstową. Zastosowanie funkcji ilustruje przykład poniżej:

```
SELECT dbo.fCenter(dbo.fDzialka2Polygon(1)) AS srodek

srodek
POINT (19.536589432796593 51.792604970513693)
```

3 Opis systemu Ewidencja gruntów – aplikacje

Na całość wizualną systemu składają się dwie aplikacje. Pierwsza z nich – pełni rolę przeglądarki danych zgromadzonych w systemie, druga natomiast jest aplikacją pomocniczą wspomagającą proces zasilania systemu danymi przestrzennymi. Obie aplikacje zostały wykonane przy użyciu narzędzia programistycznego MS Visual Studio 2010 z zastosowaniem języka C# w technologii tzw. grubego klienta. Wykorzystują elementy prezentacji danych przestrzennych w postaci map za pośrednictwem interfejsu Google Maps API.

Podstawową funkcją aplikacji Przeglądarka jest prezentacja danych przestrzennych zgromadzonych w bazie danych MS SQL Server 2008 przy użyciu interfejsu Google Maps API. Aplikacja wymaga do działania z racji wykorzystywanych funkcjonalności stałego dostępu do Internetu oraz połączenia z dedykowanym serwerem SQL. Rysunek 3 prezentuje interfejs użytkownika aplikacji.

Głównymi obiektami interfejsu użytkownika w aplikacji są następujące elementy:

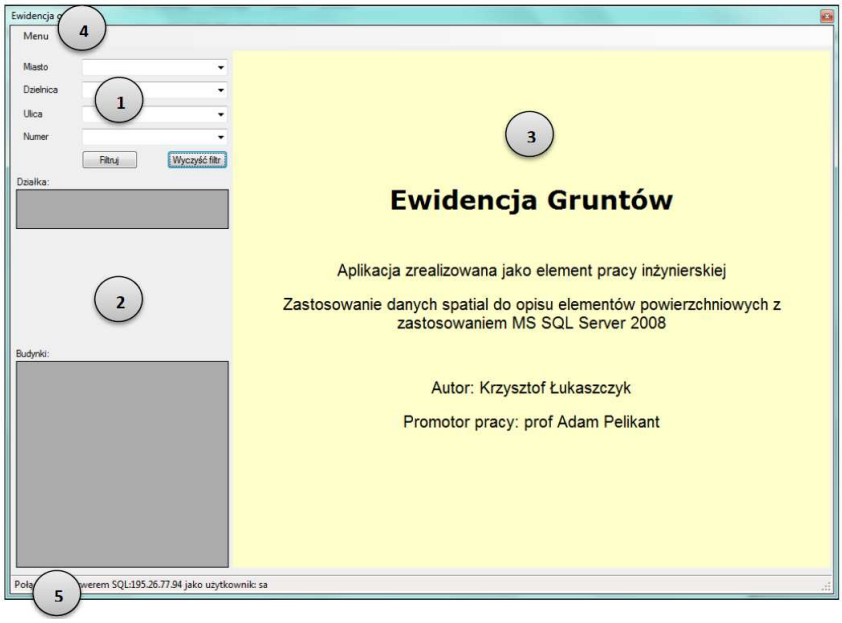
1 – zestaw pól wyboru składający się na sekcję filtrów.

2 – pola opisowe realizujące funkcjonalność kontekstowego przegląda danych ewidencyjnych.

3 – graficzna przeglądarka danych w formie map Google Maps.

4 – pasek menu

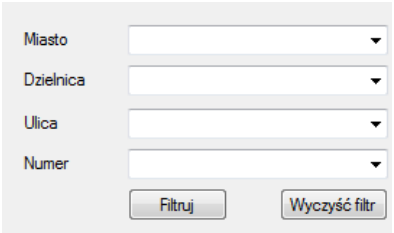
5 – dolny pasek informacyjny.



Rys. 3. Interfejs aplikacji Ewidencja gruntów – Przeglądarka

Aplikacja do prawidłowego działania wymaga stałego połączenia z siecią Internet w zakresie dostępu do strony http://maps.google.com po protokole http na standardowym porcie numer 80. Połączenie to jest wymagane na potrzeby odwzorowania danych przestrzennych zgromadzonych w bazie danych za pomocą map interfejsu Google Maps API. Takie rozwiązanie zapewnia przejrzystą formę prezentacji obszarów, dodatkowo znacznie uatrakcyjnia jego formę wizualną. Kolejnym wymaganym przez aplikację połączeniem jest połączenie z serwerem MS SQL Server 2008 udostępniającym bazę danych dla całości systemu. Parametry połączenia są w pełni konfigurowalne i zapisane w pliku konfiguracyjnym aplikacji, możliwe zaś do zmiany z poziomu użytkownika aplikacji po wyborze odpowiedniej pozycji menu. Aplikacja w momencie jej uruchomienia dokonuje automatycznie próby połączenia do serwera SQL zgodnie z danymi zapisanymi w swojej konfiguracji. Fakt niepowodzenia odnotowywany jest odpowiednim komunikatem dla użytkownika i ograniczeniem funkcjonalności aplikacji. W przypadku sukcesu, wyświetlany jest standardowy interfejs aplikacji oraz informacja o połączeniu z serwerem SQL prezentowana w dolnym pasku interfejsu aplikacji.

Sekcję filtrów aplikacji przedstawia rysunek 4:



Rys. 4. Sekcja filtrów

Sekcja składa się z czterech pól typu combobox pozwalających na wybór z bazy danych systemu konkretnego obszaru przestrzennego, składającego się na działkę. Zestaw pól działa na zasadzie wyboru kontekstowego, co oznacza że system zawęża kryteria wyszukiwania użytkownika wraz z wyborem wartości w kolejnych polach. Przy czym pierwszym kryterium jest zawsze miasto, zaś kolejnymi: dzielnica, ulica i wreszcie na końcu jej numer. Tak przyjęte założenie pozwoliło znacznie uprościć proces wyszukiwania konkretnego obszaru spośród wszystkich obszarów zgromadzonych w bazie danych. Mechanizm wyboru danych stanowiących zestaw rekordów pola combobox opiera się na połączeniu z bazą danych MS SQL Server przy użyciu metody DataSource. Przykładowo dla pola Miasto fragment kodu opisujący powyższą funkcjonalność wygląda następująco:

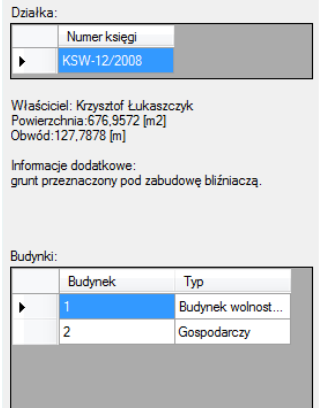
```
//miasto
ds = new DataSet();
string sqlquery = select distinct miasto from dzialki;
da = new SqlDataAdapter(sqlquery, conn);
SqlCommandBuilder cmBuilder = new
SqlCommandBuilder(da);
da.Fill(ds);
cbmiasto.DataSource = ds.Tables[0];
cbmiasto.DisplayMember = Miasto;
cbmiasto.ValueMember = Miasto;
```

Mechanizm kontekstowości dla kolejnego pola sekcji filtrów zapewnia zdarzenie typu Leave obiektu cbmiasto. Zdarzenie to realizuje funkcjonalności przedstawione na poniższym fragmencie kodu programu.

```
//miasto
if (cbmiasto.SelectedValue != null)
{
ds = new DataSet();
string sqlquery = select distinct dzielnica from
dzialki where miasto = ' +
```

```
cbmiasto.SelectedValue + ' ;
da = new SqlDataAdapter(sqlquery, conn);
SqlCommandBuilder cmBuilder = new
SqlCommandBuilder(da);
da.Fill(ds);
cbdzielnica.DataSource = ds.Tables[0];
cbdzielnica.DisplayMember = Dzielnica;
cbdzielnica.ValueMember = Dzielnica;
cbdzielnica.Enabled = true;
cbmiasto.Enabled = false;
}
```

Sekcję opisową aplikacji przedstawia rysunek 5:



Rys. 5. Sekcja pól opisowych

Sekcja składa się z dwóch pól typu gridview oraz jednego pola tekstowego typu multiline. Wszystkie te obiekty razem składają się na zespół pozwalający w zależności od kontekstu wybranej działki uzyskiwać informacje ewidencyjne na jej temat oraz budynków zlokalizowanych na jej terenie. Wszystkie informacje prezentowane w sekcji pochodzą z bazy danych, głównie z tablic ewidencyjnych systemu. Na prezentowane w sekcji informacje składają się numer księgi ewidencyjnej wybranej działki, informacje o jej właścicielu, informacje opisowe oraz dane o budynkach zlokalizowanych na terenie działki wraz z ich opisem. Dodatkowo prezentowane są informacje o powierzchni działki oraz jej obwodzie obliczane dynamicznie przy użyciu wcześniej opisywanych funkcji bazy danych fPowierzchnia i fObwod.

Najważniejsze rozwiązania programistyczne po stronie aplikacji omówmy na przykładzie funkcji fdzialkaPowierzchnia, której kod przedstawiono poniżej.

```
public float fdzialkaPowierzchnia(int IDdzialki)
{ float pow;
string sql = null;
SqlCommand sqlcmd;
sqlcmd = new SqlCommand(sql, conn);
sql = select dbo.fPowierzchnia(dbo.fDzialka2PolygonG(
+
IDdzialki + ));
SqlCommand command = new SqlCommand(sql, conn);
SqlDataReader reader5 = command.ExecuteReader();
reader5.Read();
pow = Convert.ToSingle(reader5.GetValue(0));
reader5.Close();
return pow; }
```

Funkcja pozwala w oparciu o przekazany parametr będący identyfikatorem obiektu przestrzennego reprezentującego działkę na obliczenie jego pola powierzchni. Funkcja tak jak w przypadku wielu innych zawiera standardowy zestaw instrukcji definiujących zmienne przechowujące zapytania SQL oraz budowane na ich podstawie zastawy rekordów. Jednak jej najistotniejszym elementem jest zapytanie SQL postaci:

```
sql = select dbo.fPowierzchnia(dbo.fDzialka2PolygonG(
+
IDdzialki + ));
```

wywołujące dwie funkcje z bazy danych, które razem w połączeniu pozwalają realizować funkcjonalność obliczania pola powierzchni obszaru geometrycznego w oparciu o metody Spatial. Wewnętrzna z funkcji zapytania o nazwie fDzialka2PolygonG buduje geograficzny obiekt przestrzenny typu polygon z zestawu rekordów tablicy GEOM_DZIALKI. Odbywa się to w oparciu o przekazany w parametrze funkcji identyfikator obszaru – działki. Funkcja zewnętrzna dbo.fPowierzchnia oblicza pole powierzchni obszaru będącego przestrzennym obiektem geograficznym zbudowanym przez funkcję wewnętrzną.

Przyjęto ogólną zasadę w konstrukcji całej aplikacji, aby bloki kodu odpowiedzialne za konkretne funkcjonalności grupować w funkcje. Taka metoda pozwoliła znacznie uporządkować kod aplikacji oraz w znacznym stopniu ułatwiła analizę ewentualnych błędów. Grupowanie kodu w funkcje przyczyniło się pośrednio także do utrzymania założonego porządku przy opisie całego systemu Ewidencja gruntów.

Przeglądarka map została zaprojektowana jako element zapewniający wizualizację danych o obiektach geograficznych zgromadzonych w

bazie danych MS SQL Server 2008. Zasada jej działania opiera się na wykorzystaniu interfejsu programowania aplikacji (API) Google Maps do prezentowania na mapie obszarów geograficznych. Rys. 6 prezentuje fragment aplikacji Ewidencja gruntów – przeglądarka przedstawiający obszar przestrzenny odczytany z bazy danych systemu i zaprezentowany w formie mapy Google Maps.

Rys. 6. Graficzna przeglądarka danych w formie map Google Maps

Elementem zapewniającym funkcjonalność prezentacji danych geograficznych o obiektach przestrzennych w systemie jest kontrolka WebBrowser. Umożliwia ona w oparciu o informacje odczytywane kontekstowo z bazy danych systemu oraz budowany na tej podstawie kod HTML realizować graficzną prezentację obiektów przestrzennych. Budowany dynamicznie kod HTML musi być całkowicie zgodny ze specyfikacją interfejsu Google Maps API. Aplikacja wykorzystuje na potrzeby prezentacji specyfikację interfejsu w wersji wyłącznie numer 3 (Google Maps JavaScript API V3). Całość kodu odpowiedzialnego za prezentację danych przestrzennych za pomocą map podzielona została na następujące funkcje:

- Funkcje nadrzędne - budujące całość mapy:
  - fbuildHTML().
- Funkcje podrzędne - ogólne:
  - fdefaultHTML();
  - fhaederHTML();

- fopcjemapyHTML();
- ffooterHTML().

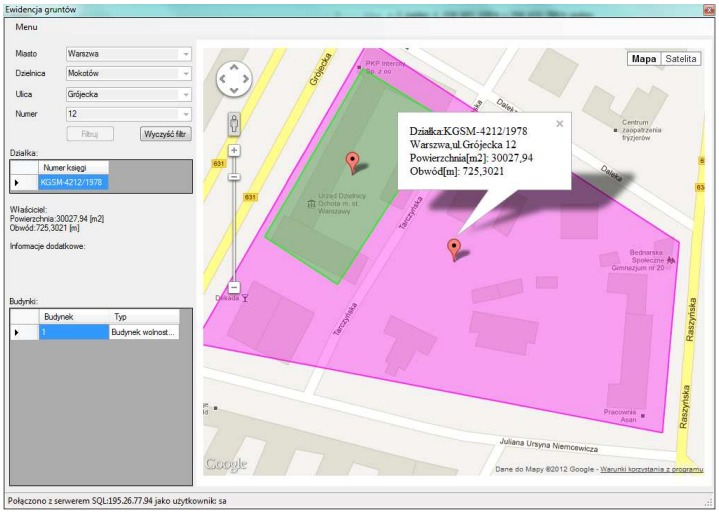
– Funkcje dotyczące działek:
  - fCenterOfDzialkaHTML;
  - fdzialkaMarkerHTML;
  - fdzialkamarkerinfoHTML;
  - fdzialkaHTML.
– Funkcje dotyczące budynków:
  - fCenterOfBudynekHTML;
  - fbudynekMarkerHTML;
  - fbudynekMarkerinfoHTML;
  - fbudynekHTML.
– Funkcje pomocnicze:
  - GetRandomColor().

Z uwagi na dosyć rozbudowane instrukcje oraz częściowe ich powielanie w każdej z funkcji omówione szczegółowo zostaną wyłącznie wybrane fragmenty. Przeanalizujmy fragment funkcji fdzialkamarkerinfoHTML, który zamieszczono poniżej.

```
1 public string fdzialkamarkerinfoHTML(int IDdzialki)
2 {
3 sHTML = null;
4 string sql = null;
5 SqlCommand sqlcmd;
6 sqlcmd = new SqlCommand(sql, conn);
7 sql = select numer_ksw, ulica, ulica_numer, miasto
from dzialki where iddzialki =  + IDdzialki + ;;
8 SqlCommand command = new SqlCommand(sql, conn);
9 SqlDataReader reader4 = command.ExecuteReader();
10 reader4.Read();
11 sHTML = sHTML + var dzialkainfo = \Dziaka: +
reader4.GetValue(0) + </br> + reader4.GetValue(3) +
,ul. + reader4.GetValue(1) +   + reader4.GetValue(2);
12 reader4.Close();
13 //powierzchnia
14 sHTML = sHTML + </br> + Powierzchnia[m2]:  +
fdzialkaPowierzchnia(IDdzialki);
15 //obwod
16 sHTML = sHTML + </br> + Obwód:  +
fdzialkaObwod(IDdzialki) + \; + \r\n;
17 sHTML = sHTML + var infowindow = new
google.maps.InfoWindow({ + \r\n;
18 sHTML = sHTML + content: dzialkainfo + \r\n;
```

```
19 sHTML = sHTML + }); + \r\n;
20 sHTML = sHTML +
google.maps.event.addListener(markerdzialka +
IDdzialki + , 'mouseover', function() { + \r\n;
21 sHTML = sHTML + infowindow.open(mapa, markerdzialka
+ IDdzialki + ); + \r\n;
22 sHTML = sHTML + }); + \r\n;
23 sHTML = sHTML +
google.maps.event.addListener(markerdzialka +
IDdzialki + , 'mouseout', function() { + \r\n;
24 sHTML = sHTML + infowindow.close(); + \r\n;
25 sHTML = sHTML + }); + \r\n;
26 return sHTML;
27 }
```

Funkcja służy do obsługi informacji wyświetlanej w postaci tzw. dymku po najechaniu wskaźnikiem myszy na marker. Informacja dotyczy obszaru przestrzennego – działki, której identyfikator IDDZIALKI przekazywany jest jako argument wywołania funkcji skutek pokazuje rysunek 8.



Rys. 7. Ilustracja działania funkcji fdzialkamarkerinfoHTML

Przypomnijmy kroki budowania prostej mapy. Etap pierwszy i drugi to etap tworzenia kodu HTML wraz z obsługą języka skryptów Java Script. Etap trzeci służy utworzeniu i wyświetleniu w przeglądarce pierwszej mapy za pomocą interfejsu Google Maps API. Etap czwarty to dodanie do mapy znacznika za pomocą klasy MARKER(). Dodajmy zatem do naszego kodu funkcjonalność wyświetlania informacji w postaci dymków

wyświetlanych po najechaniu kursorem myszy na marker. W tym celu do funkcji MAPASTART() dodajmy następujące instrukcje:

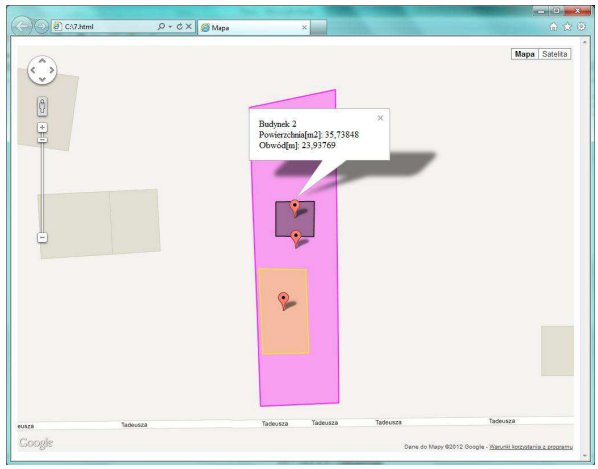
```
34 var markerinfo = Tekst wyswietlany w dymku markera;
35 var markerinfowindow = new google.maps.InfoWindow({
36 content: markerinfo
37 });
38 google.maps.event.addListener(marker, 'mouseover',
function()
39 {
40 markerinfowindow.open(mapa, marker);
41 });
42 google.maps.event.addListener(markerbudynek1,
'mouseout', function() {
43 markerinfowindow.close();
44 });
```

Klasa Marker dostarcza metody realizujące funkcjonalność wyświetlania informacji w postaci dymków skojarzonych z konkretnymi markerami. Na potrzeby przykładu omówmy poszczególne instrukcje realizujące tę funkcjonalność. Linia 34 definiuje zmienną o nazwie markerinfo przechowującą tekst wyświetlany w obszarze dymku. Linia kolejne od 35 do 37 definiują obiekt jako InfoWindow zawierający treść przechowywaną w uprzednio zdefiniowanej zmiennej markerinfo. Obiekt InfoWindow jest reprezentacją tzw. dymku na mapie. Kolejne instrukcje są definicją zdarzeń skojarzonych z naszym znacznikiem zdefiniowanym pod nazwą marker. Instrukcje zawarte w liniach od 38 do 41 definiują reakcję na zdarzenie mouseover czyli najechanie wskaźnikiem myszy na marker. Reakcją na zdarzenie jest wyświetlenie uprzednio zdefiniowanego obiektu InfoWindow. Kolejne zdarzenie to przemieszczenie kursora myszy poza obszar markera zdefiniowane zostało w liniach od 42 do 44. Reakcją na nie jest zamknięcie obiektu InfoWindow.

Złożoność opisywanej funkcji spowodowana jest specyfiką interfejsu Google Maps API, która zakłada użycie z góry zdefiniowanych znaków będących separatorami oraz wymaga całkowitej zgodności składniowej budowanego kodu. Ta specyfika przekłada się na konieczność budowy w dwóch etapach zmiennej przechowującej tablicę wielowymiarową zawierającą współrzędne wierzchołków obszaru geograficznego. Wymagane jest bowiem oddzielenie znakiem separatora, którym jest przecinek wszystkich wierszy tablicy poza ostatnim. Aby móc zrealizować to zadanie w pierwszym etapie tworzony jest zestaw rekordów służący policzeniu ilości wierszy odpowiadających ilości wierzchołków budowanego obszaru. Następnie w drugim etapie dla wszystkich wierszy poza ostatnim na końcu budowanego ciągu znaków w zmiennej sHTML dodawany jest znak separatora.

Takie rozwiązanie jest bardzo elastyczne i pozwala budować obszary składające się teoretycznie z nieskończonej ilości wierzchołków. Mogłem oczywiście przyjąć założenie że każdy z obiektów geograficznych stanowiących działkę będzie miał skończoną – stałą liczbę wierzchołków, co pozwoliłoby znacznie uprościć kod samego programu. Jednak takie rozwiązanie byłoby bardzo sztywne i ograniczałoby obszary działek wyłącznie do czworoboków.

Funkcje dotyczące budynków są bardzo zbliżone do opisywanych funkcji dotyczących działek. Tak jak one służę do budowania kodu Java Script pozwalającego na wyświetlanie na mapie wielokątów reprezentujących budynki. Ich wyróżnikiem są dwa parametry przekazywane do każdej z funkcji. Pierwszy IDDZIALKI dotyczy obszaru reprezentującego działkę, na której umieszczono budynek. Parametr drugi IDBUDYNKU jest identyfikatorem konkretnego budynku umieszczonego na obszarze działki.



Rys. 8. Mapa wygenerowana przez system Ewidencja gruntów

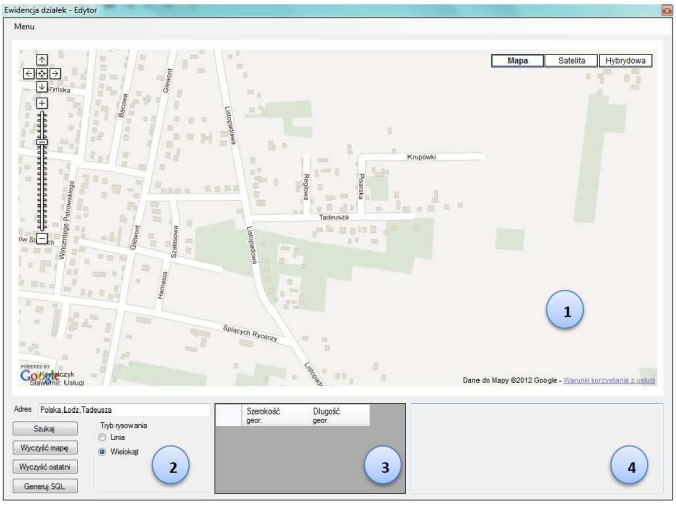
Przyjęto założenie, że w odróżnieniu od funkcji dotyczących budynków istnieje możliwość osadzenia teoretycznie nieskończonej ilości budynków na jednym obszarze działki. Jest to oczywiście założenie czysto teoretyczne. W praktyce jednak i na potrzeby demonstracji całości systemu ilość budynków na jednym obszarze działki nie przekracza dwóch. Efekt działania powyżej opisywanych funkcjonalności przedstawiony jest na rysunku 9.

Aplikacja Ewidencja gruntów – edytor powstała jako aplikacja pomocnicza wchodząca w skład całości systemu. W trakcie tworzenia całości systemu spotkałem się z problemem zasilenia przykładowymi danymi geograficznymi tablic bazy danych systemu. Problem był dosyć istotny i wymagał określenia w bardzo precyzyjny sposób szerokości i

długości geograficznej każdego z punktów stanowiących wierzchołki obszarów (działek i budynków). W efekcie finalnym powstała w pełni funkcjonalna aplikacja przy użyciu narzędzia programistycznego MS Visual Studio 2010 z zastosowaniem języka C# w technologii tzw. grubego klienta. Rysunek 10 prezentuje interfejs użytkownika aplikacji.

Głównymi elementami interfejsu użytkownika w aplikacji są następujące elementy:

- Sekcja 1 – graficzna przeglądarka danych w formie map Google Maps.
- Sekcja 2 – sekcja elementów aplikacji zapewniających interakcję z użytkownikiem.
- Sekcja 3 – przeglądarka współrzędnych wybranych na mapie punktów.
- Sekcja 4 – przeglądarka instrukcji SQL.



Rys. 9. Interfejs aplikacji Ewidencja gruntów – Edytor

Aplikacja podobnie jak poprzednia wymaga do prawidłowego działania stałego połączenia z siecią Internet w zakresie dostępu do strony http://maps.google.com. Dostęp ten realizowany jest po protokole http na standardowym porcie numer 80. Całość kodu podzielona jest na bloki tematyczne, głównie za pomocą poszczególnych funkcji Java Script. Dodatkowo, jak w przypadku poprzedniej aplikacji w kodzie można wyróżnić instrukcje języka HTML stanowiących nagłówek i stopkę całego kodu. Najważniejszymi funkcjami realizującymi funkcjonalność tworzenia i wyświetlania markerów oraz obiektów geometrycznych są:

- mapClick() – realizująca odpowiedź na zdarzenie kliknięcia przez użytkownika w punkcie umieszczenia kursora myszy na mapie.

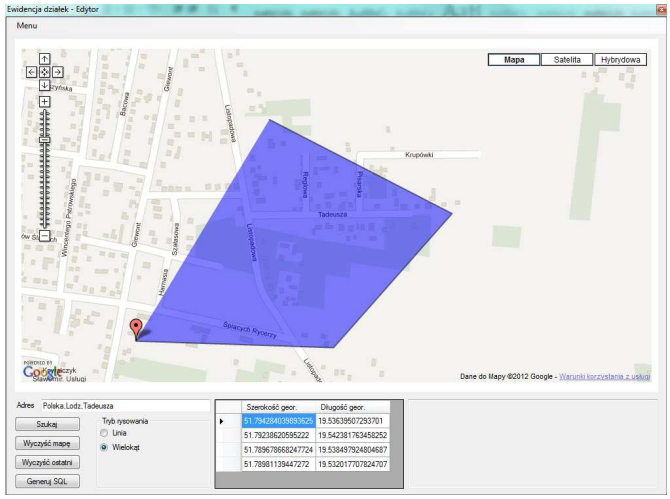
- clearMap() – funkcja wywoływana z poziomu sekcji interaktywnej, powodująca usunięcie z mapy wszystkich markerów i obiektów geometrycznych.
- toggleDrawMode() – funkcja wywoływana z poziomu sekcji interaktywnej, powodująca przełączenie trybu rysowania obiektów geometrycznych z linii na wielokąty i odwrotnie.
- deleteLastPoint() – funkcja wywoływana z poziomu sekcji interaktywnej. Umożliwia usunięcie z mapy obiektu markera oraz zmiennej tablicowej polyPoints współrzędnych geograficznych ostatniego punktu naniesionego na mapę.
- drawCoordinates() – funkcja wywoływana w funkcji mapClick() w odpowiedzi na zdarzenie kliknięcia przez użytkownika na mapie. Powoduje wstawienie obiektu klasy Marker() w punkcie umieszczenia kursora myszy. Dodatkowo funkcja realizuje funkcjonalność rysowania na mapie obiektów geometrycznych typu linie i wielokąty.
- logCoordinates() – kolejna funkcja nie wywoływana wprost przez użytkownika. Powoduje zapisanie w zmiennej tablicowej polyPoints współrzędnych geograficznych punktu należących do umieszczanego na mapie markera.
- showAddress – funkcja wywoływana z poziomu sekcji interaktywnej. Realizuje funkcjonalność wyszukiwania obszarów na mapie w oparciu o adres w formacie kraj, miasto, ulica wprowadzony przez użytkownika. Po znalezieniu poszukiwanego adresu – geokoder Google wyświetla mapę obszaru centrując ją w punkcie skojarzonym z wyszukanym adresem.

Przykład działania aplikacji rysującej obiekt geometryczny – wielokąt w oparciu o wskazane przez użytkownika na mapie punkty ilustruje rysunek 11. W sekcji o nazwie przeglądarka współrzędnych punktów prezentowane są w formie tabelarycznej wartości długości i szerokości geograficznej w/w punktów.

Jak zostało to już wspomniane wcześniej aplikacja do swojego działania wymaga stałego połączenia z serwerem bazy danych Microsoft SQL Server 2008. Parametry tego połączenia przechowywane są w pliku konfiguracyjnym o nazwie Spatial.exe.config będącym w tym samym katalogu co aplikacja. Odczyt i zapis parametrów realizowany jest przez aplikację za pośrednictwem metody ConfigurationManager.AppSettings. Plik konfiguracyjny jest typowym plikiem XML o następującej budowie:

```
1 <?xml version=1.0 encoding=utf-8 ?>
2 <configuration>
3 <configSections>
```

```
4 <sectionGroup name=userSettings
type=System.Configuration.UserSettingsGroup, System,
Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089 >
5 <section name=Spatial.Properties.Settings
type=System.Configuration.ClientSettingsSection,
System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089
allowExeDefinition=MachineToLocalUser
requirePermission=false />
6 </sectionGroup>
7 </configSections>
8 <userSettings>
9 <Spatial.Properties.Settings>
10 <setting name=suggest serializeAs=String>
11 <value>Suggest</value>
12 </setting>
13 <setting name=custom serializeAs=String>
14 <value>CustomSource</value>
15 </setting>
16 </Spatial.Properties.Settings>
17 </userSettings>
18 <appSettings>
19 <add key=Serwer value=127.0.0.1 />
20 <add key=Port value=1444 />
21 <add key=User value=sa />
22 <add key=Password value=haslo />
23 </appSettings>
24 </configuration>
```



Rys. 10. Przykład działania aplikacji Ewidencja gruntów – edytor

Oprócz standardowych tagów generowanych automatycznie przez mechanizmy aplikacji w liniach od 1 do 17, plik konfiguracyjny zawiera parametry użytkownika dotyczące połączenia do serwera bazy danych. Parametry te zawarte są w liniach od 19 do 22. Dane w pliku konfiguracyjnym są zapisywane wyłącznie przez mechanizmy aplikacji. Nie jest zalecane ręcznie zmienianie ustawień w nim za pośrednictwem edytora. Dostęp do ustawień realizowany jest za pośrednictwem funkcji Konfiguracja w menu głównym aplikacji Ewidencja gruntów – przeglądarka. Na podstawie danych zawartych w pliku konfiguracyjnym aplikacja buduje łańcuch połączeniowy ConnectionString służący do nawiązania połączenia z bazą danych. Ta funkcjonalność realizowana jest poprzez zestaw instrukcji wymienionych poniżej:

```
public static string serwer =
ConfigurationManager.AppSettings[Serwer];
public static string port =
ConfigurationManager.AppSettings[Port];
public static string user =
ConfigurationManager.AppSettings[User];
public static string password =
ConfigurationManager.AppSettings[Password];
public static string conn_str = Data Source= + serwer
+ , + port + ;Initial Catalog=spatial;User ID= + user
+ ;Password= + password + ;;
SqlConnection conn = new SqlConnection(conn_str);
conn.Open();
```

W rezultacie wykonania powyższego fragment kodu, zostaje nawiązane połączenie z bazą danych conn zgodnie z danymi zawartymi w zmiennej conn_str. W aspekcie dotyczącym bezpieczeństwa należy zauważyć sposób przechowywania hasła połączenia do bazy danych w pliku konfiguracyjnym. Dla uproszczenia całości mechanizmu konfiguracji hasło to przechowywane jest w postaci jawnego tekstu. Istnieje zatem możliwość jego odczytu przez każdą osobę mającą dostęp do pliku Spatial.exe.config.

4 Podsumowanie

Celem pracy było zaprezentowanie zastosowania metod przestrzennych Spatial do opisu elementów powierzchniowych z wykorzystaniem Microsoft SQL Server 2008. Cel ten został zrealizowany na przykładzie stworzonego na tę potrzebę systemu o nazwie Ewidencja gruntów. Głównym elementem systemu stała się aplikacja Ewidencja gruntów –

przeglądarka. Aplikacja ta pozwoliła poprzez zastosowanie jako warstwy prezentacyjnej interfejsu Google Maps API w ciekawy i atrakcyjny sposób omówić zagadnienia związane z tematem mojej pracy.

Najważniejszą częścią pracy był szczegółowy opis funkcji bazy danych, w których to zawarte zostały instrukcje dotyczące konkretnych metod przestrzennych Spatial. Funkcje te w połączeniu z warstwą prezentacyjną w postaci map Google pozwoliły w znacznym stopniu przybliżyć zagadnienia związane z obsługą i odwzorowywaniem danych przestrzennych.

Literatura

[1] *Inside Microsoft SQL Server 2008: T-SQL Programming,* Wydawnictwo MSPress

[2] *Beginning Spatial with SQL Server 2008,* Wydawnictwo Apress

[3] http://www.opengeospatial.org

[4] http://social.msdn.microsoft.com/Forums/en-US/sqlspatial/thread/0ff0e3ec-fcb3-4aaa-9fbc-dbca6186d717

[5] http://gmapsapi.com

[6] http://msdn.microsoft.com

[7] http://pl.wikipedia.org

APPLICATION OF SPATIAL DATA TO DESCRIBE THE SURFACE ELEMENTS USING MS SQL SERVER 2008

Summary – The main topic of the article is to create a system that supports the search process of flat objects on a map, and the input of such objects and the server-side database storage. For this purpose built complex Spatial types together with their methods have been used. In the application tier uses the .NET development environment and Google Maps as a graphical presentation of results and data entry.