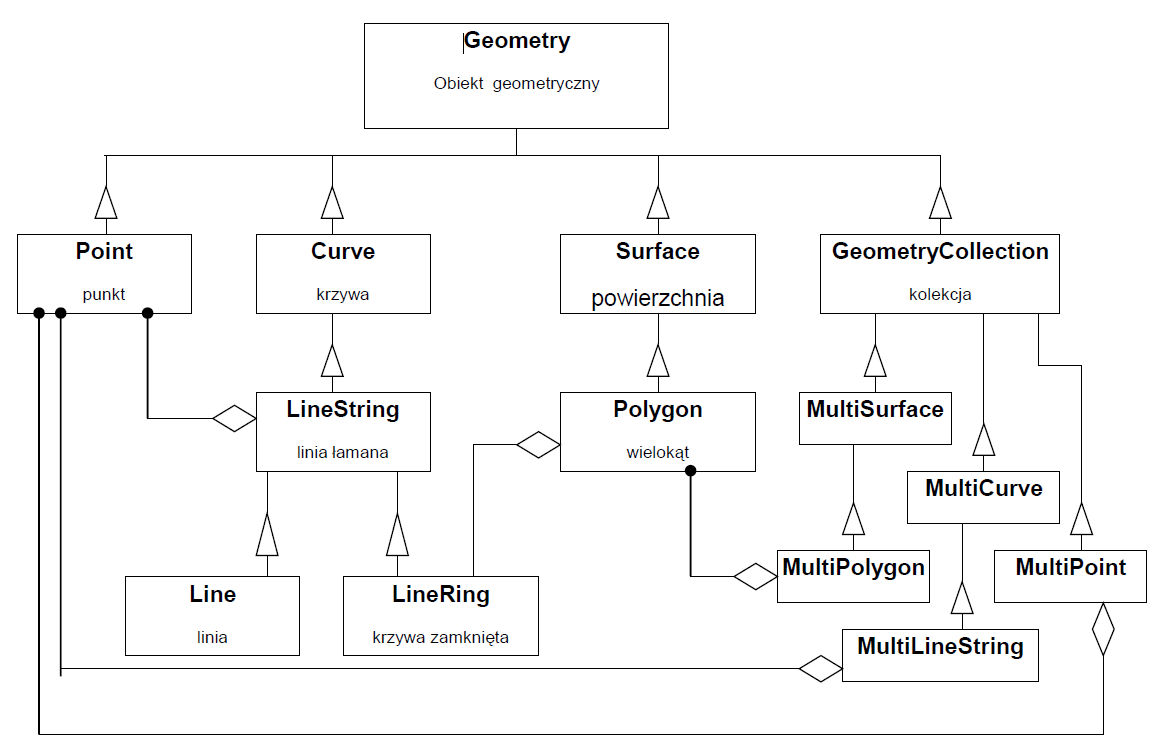
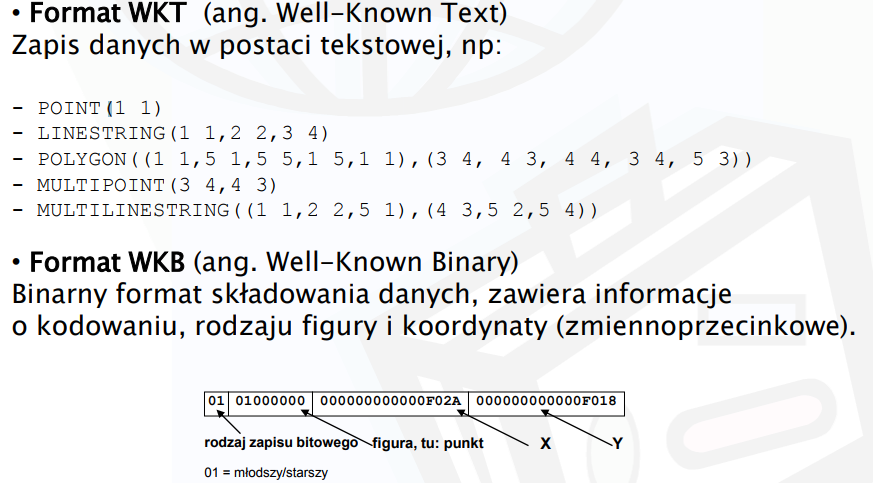
**Dane typu geography w MS SQL Server**

W ostatnich latach powstały powszechnie zaakceptowane międzynarodowe normy (ISO) i specyfikacje (OpenGIS) dotyczące systemów GIS. Wynikiem porozumienia obu organizacji (ISO i OGC) jest stan aktualny, w którym specyfikacje OpenGIS pełnią rolę specyfikacji implementacyjnych uwzględniających specyficzne wymagania konkretnego środowiska narzędzi programistycznych.

Specyfikacja standardu dla typu geometry i geography została przedstawiona na rysunku. Dla obu typów hierarchia jest identyczna. Należy jednak pamiętać, że przy operacjach wykonywanych na typach geography uwzględniania jest sferyczność kuli ziemskiej.



Dane typów geography i geometry reprezentowane są w bazach danych zwykle   
w jednym z dwóch formatów: Well-Known Text (WKT) oraz Well-Known Binary (WKB). Oba te typy zdefiniowane są w standardach dostarczanych przez OGC. Charakterystyka formatu WKT i WKB została przedstawiona niżej na rysunku.



Rys.7. Charakterystyka formatu WKT i WKB

MS SQL Server dostarcza nam w zakresie danych przestrzennych dwa typy tj. typ geometry oraz typ geography.

Pierwszy z typów tj. typ geometry wykorzystuje się do przechowywania informacji o obiektach znajdujących się w dwuwymiarowej przestrzeni. Przykładem takiej przestrzeni jest układ kartezjański, gdzie położenie każdego punktu określane jest za pomocą wartości X i Y. Tego rodzaju ograniczenie (do dwóch wymiarów) powoduje jednakże, że Ziemię musimy traktować jako powierzchnię płaską, przez co możemy przechowywać tylko dane obiektów na tyle małych, aby nie wpływała na nie krzywizna planety. Przykładowo można je zastosować do przedstawienia planów pomieszczeń w budynkach czy też położenia elementów w magazynie.

Typem danych, który pozwala na uwzględnienie elipsoidalnego charakteru budowy kuli ziemskiej jest drugi z typów tj. typ geography. Przeznaczeniem tego typu jest przechowywanie danych o względnie dużych obiektach (takich, gdzie krzywizna Ziemi ma znaczenie) oraz współrzędnych geograficznych budynków, miast czy państw. Możliwe jest to dzięki temu, że informacje o położeniu podawane są za pomocą długości i szerokości geograficznej.

Jako przykład funkcji uwzględniającej sferyczność kuli ziemskiej można wskazać funkcję STDistance, która liczy odległość między dwiema figurami. Odległość ta jest liczona zupełnie inaczej dla danych typu geometry (gdzie jest długością linii prostej łączącej obie figury), a inaczej dla geography (gdzie jest długością łuku ze względu na krzywiznę powierzchni).

W obu ww. typach danych opis położenia punktu możliwy jest tylko w dwóch wymiarach. SQL Server dostarcza nam jednak również dwie dodatkowe wartości do opisu każdego punktu – Z i M. Należy jednak zaznaczyć, że te dane nie są wykorzystywane w żadnych obliczeniach, ale pozostają jedynie do dyspozycji użytkownika, np. do przechowywania informacji o wysokości położenia punktu nad poziomem morza.

**Proste wykorzystanie typu geography**

DECLARE @GeoTable TABLE

(

id int identity(1,1),

location geography

)

--Using geography::STGeomFromText

INSERT INTO @GeoTable

SELECT geography::STGeomFromText('POINT(-122.35900 47.65129)', 4326)

--Using geography::Point

INSERT INTO @GeoTable

SELECT geography::Point(47.65100,-122.34720, 4326);

DECLARE @DistanceFromPoint geography

SET @DistanceFromPoint = geography::STGeomFromText('POINT(-122.34150 47.65234)', 4326);

SELECT id,location.Lat Lat,location.Long Long,location.STDistance(@DistanceFromPoint) Distance

FROM @GeoTable;

Kolejne przykłady definiowania za pomocą składni SQL wybranych elementów dla typu geometry:

**Point**

|  |  |
| --- | --- |
| **Skrypt** | DECLARE @g geometry;  SET @g =geometry::STGeomFromText('POINT (3.51 4.51)', 0);  //Lub  SET @g = geometry::Parse('POINT(3.51 4.51)');  select @g |
| **Wynik** | 0x00000000010C14AE47E17A140C400AD7A3703D0A1240 |
| **Wizualizacja** |  |

Tworzenie Point dla typu geometry (źródło: opracowanie własne)

**LineString**

|  |  |
| --- | --- |
| **Skrypt** | DECLARE @g2geometry;  DECLARE @g2 geometry='LINESTRING(1 1, 2 3,4 8, -6 3)';  select @g2 |
| **Wynik** | 0x00000000010C14AE47E17A140C400AD7A3703D0A1240 |
| **Wizualizacja** |  |

Tworzenie Linestring dla typu geometry (źródło: opracowanie własne)

**Polygon**

|  |  |
| --- | --- |
| **Skrypt** | DECLARE @g2 geometry;  DECLARE @g2 geometry='POLYGON((-5 -5, -5 5, 5 5, 5 -5, -5 -5),  (3 0, 6 0, 6 3, 3 3, 3 0),(3 -3,5 -3,4 -4,3 -4,3 -3))';  select @g2 |
| **Wynik** | 0x0000000001000F00000000000000000014C000000000000014C000000000000014C0000000000000144000000000000014400000000000001440000000000000144000000000000014C000000000000014C000000000000014C00000000000000840000000000000000000000000000018400000000000000000000000000000184000000000000008400000000000000840000000000000084000000000000008400000000000000000000000000000084000000000000008C0000000000000144000000000000008C0000000000000104000000000000010C0000000000000084000000000000010C0000000000000084000000000000008C00300000002000000000005000000000A00000001000000FFFFFFFF0000000003 |
| **Wizualizacja** |  |

Tworzenie Polygon dla typu geometry (źródło: opracowanie własne)

**MultiPolygon**

|  |  |
| --- | --- |
| **Skrypt** | DECLARE @g2 geometry;  DECLARE @g2 geometry='MULTIPOLYGON(((1 1, 1 -1, -1 -1, -1 1, 1 1)),  ((1 1, 3 1, 3 3, 1 3, 1 1)))';  select @g2 |
| **Wynik** | 0x0000000001040A000000000000000000F03F000000000000F03F000000000000F03F000000000000F0BF000000000000F0BF000000000000F0BF000000000000F0BF000000000000F03F000000000000F03F000000000000F03F000000000000F03F000000000000F03F0000000000000840000000000000F03F00000000000008400000000000000840000000000000F03F0000000000000840000000000000F03F000000000000F03F020000000200000000020500000003000000FFFFFFFF0000000006000000000000000003000000000100000003 |
| **Wizualizacja** |  |

Tworzenie MultiPolygon dla typu geometry (źródło: opracowanie własne)

Przykłady tworzenia wybranych obiektów dla typu geography:

**MultiPolygon**

|  |  |
| --- | --- |
| **Skrypt** | DECLARE @g geography;  SET @g =geography::STMPolyFromText('MULTIPOLYGON(((-121.358 46.653, -121.348 46.649, -121.358 46.658, -121.358 46.653)), ((-121.341 46.656, -121.341 46.661, -121.351 46.661, -121.341 46.656)))', 4326);  SELECT @g ; |
| **Wynik** | 0xE6100000010408000000DD24068195534740F4FDD478E9565EC0508D976E1253474083C0CAA145565EC04E62105839544740F4FDD478E9565EC0DD24068195534740F4FDD478E9565EC08716D9CEF7534740E7FBA9F1D2555EC0F853E3A59B544740E7FBA9F1D2555EC0F853E3A59B5447405839B4C876565EC08716D9CEF7534740E7FBA9F1D2555EC0020000000200000000020400000003000000FFFFFFFF0000000006000000000000000003000000000100000003 |
| **Wizualizacja** |  |

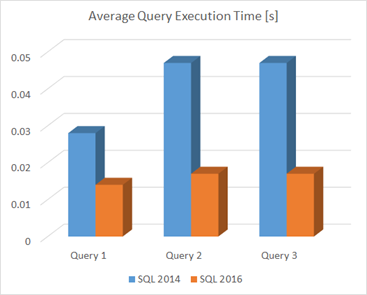
Tabela 5. Tworzenie MultiPolygon dla typu geography (źródło: opracowanie własne)

**GeometryCollection**

|  |  |
| --- | --- |
| **Skrypt** | DECLARE @g geography='GEOMETRYCOLLECTION(CIRCULARSTRING(-111.358 41.653, -111.348 41.649, -111.348 41.658, -111.358 41.658, -111.358 41.653)  ,CURVEPOLYGON(CIRCULARSTRING(-111.358 41.653, -111.348 41.649, -111.348 41.658, -111.358 41.658, -111.358 41.653)))';  SELECT @g |
| **Wynik** | 0xE610000002040A000000DD24068195D34440F4FDD478E9D65BC0508D976E12D3444083C0CAA145D65BC04E62105839D4444083C0CAA145D65BC04E62105839D44440F4FDD478E9D65BC0DD24068195D34440F4FDD478E9D65BC0DD24068195D34440F4FDD478E9D65BC0508D976E12D3444083C0CAA145D65BC04E62105839D4444083C0CAA145D65BC04E62105839D44440F4FDD478E9D65BC0DD24068195D34440F4FDD478E9D65BC0020000000200000000020500000003000000FFFFFFFF000000000700000000000000000800000000010000000A |
| **Wizualizacja** |  |

Tabela 6. Tworzenie GeometryCollection dla typu geography (źródło: opracowanie własne)

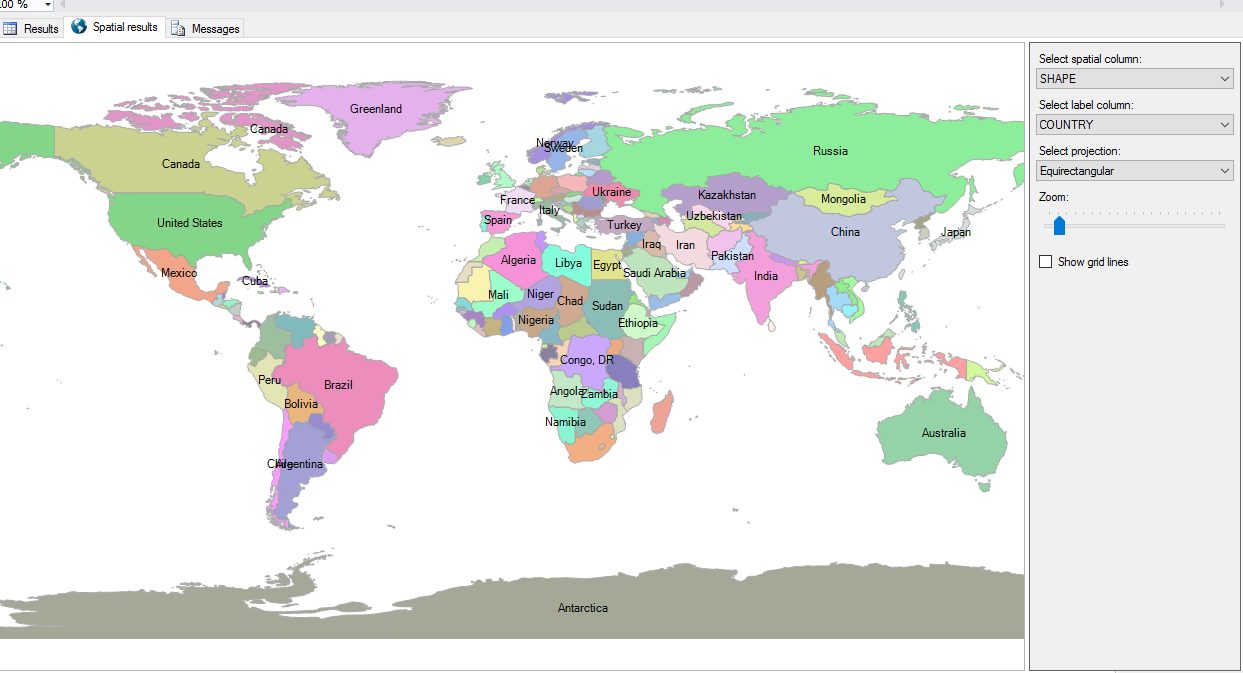
Wersja MS SQL z 2016 roku wprowadziła istotną nowość   
w postaci natywnego wsparcia typów geometry i geography (ang. spatial data). Dzięki takiemu podejściu możliwe jest uniknięcie nadmiarowych transakcji, co   
w konsekwencji prowadzi do znaczącego przyspieszenia wykonywania operacji na tych typach danych



Średni czas operacji na typach geometrycznych dla MS SQL 2014 I 2016

Poniżej przedstawiono bardziej zaawansowane zastosowanie typów geometrycznych na przykładzie odwzorowania terenów administracyjnych





Wizualizacja danych geograficznych opisujących kontury krajów

Zawartość tabeli tworzącej układ geograficzny Polski, który jest ogólnodostępny.



. Przykładowy zestaw danych geometrycznych w MS SQL Server



Wizualizacja danych geograficznych opisujących kontury województw (źródło: opracowanie własne)

**Linki pomocnicze:**

Dane shape:

* <https://freegisdata.rtwilson.com/>
* [https://gis-support.pl/baza-wiedzy/dane-do-pobrania/#](https://gis-support.pl/baza-wiedzy/dane-do-pobrania/)

Biblioteka GDAL

* <http://www.gisinternals.com/release.php>

**Przykłady poleceń import danych w powłoce shellowej bibliotece GDAL:**

**1) import do MSSQL**

ogr2ogr -lco ENCODING=CP1250 -f  **MSSQLSpatial** "MSSQL:server=localhost;database=master;trusted\_connection=yes" "C:\Users\pfigat\Desktop\panstwo\Panstwo.shp"

**2) Import do MSSQL z przeliczeniem EPSG**

ogr2ogr -lco ENCODING=CP1250 **-s\_srs epsg:2180 -t\_srs epsg:4326**  -f  MSSQLSpatial "MSSQL:server=localhost;database=master;trusted\_connection=yes" "plik.shp"

**3)Import do MSSQL z dostepem do bazy za pomocą loginu I hasla**

ogr2ogr -lco ENCODING=CP1250 -s\_srs epsg: 4258 -t\_srs epsg:4326  -f  MSSQLSpatial „MSSQL:server=localhost;database=master; **UID=sa;PWD=sa** "   
"C:\Users\pfigat\Desktop\panstwo\panstwo.shp"

**4) import Gmin z ewentualnym nadpisaniem istniejącej już bazy ( -overwrite)**

ogr2ogr -**overwrite** -lco ENCODING=CP1250 -s\_srs epsg:2180 -t\_srs epsg:4326  -overwrite -f  MSSQLSpatial "MSSQL:serveQr=nazwa\_serwera;database=nazwa\_bazy;trusted\_connection=yes" "d:\dane\gminy.shp"

**5) import z konwersją danych do typu Geography**

ogr2ogr -overwrite -lco **"GEOM\_TYPE=geography"** -s\_srs epsg:4258 -t\_srs epsg:4326 -overwrite -f MSSQLSpatial "MSSQL:server=Andromeda3\kobize\_test;database=YtFilms; UID= YtFilms;PWD= YtFilms" "C:\Users\Pawel.Figat.IOS\Desktop\Powiaty\Powiaty.shp"

**6) przekształcenie danych do formatu JSON**

ogr2ogr -f GEOJSON -s\_srs epsg:2180 -t\_srs epsg:4326  "d:\dane\gminy.json" "d:\dane\PRG\_jednostki\_administracyjne\_v15\gminy.shp"

Ciekawostki

* <https://thetruesize.com/>

Reorientacja obiektów w bazie

UPDATE precincts\_reprojected

SET geog4326= geog4326.MakeValid().ReorientObject()

WHERE geog4326.MakeValid().EnvelopeAngle() > 90;