Jacek Pawłowski
Wydział Informatyki i Zarządzania
Wyższej Szkoły Informatyki w Łodzi

Promotor: dr hab. Adam Pelikant, prof. WSInf

ZAPIS PLIKÓW GRAFIKI WEKTOROWEJ DO BAZY DANYCH MS SQL

Streszczenie – Artykuł poświęcony jest zastosowaniu serwera baz danych do przechowywania grafiki wektorowej. Oryginalnym elementem zaproponowanego rozwiązania jest przechowywanie elementów składowych tego typu danych, zamiast, co do tej pory było stosowane, zapisywaniu całego pliku w typach BLOB. Wykorzystany został złożony, obiektowy typ geometry, a jako środowisko wykorzystano MS SQL Server. Zrealizowano dekompozycję najpopularniejszego pliku grafiki wektorowej *.pdf do obiektów bazy danych. Jako weryfikację jakości przyjętych rozwiązań zrealizowano zapis danych z bazy do wyjściowego formatu. Zaproponowane rozwiązanie może zostać wykorzystane do przechowywania i wielokrotnego wykorzystywania bloków wektorowych przy tworzeniu złożonych rysunków i dokumentacji.

1 Wstęp

W związku z rozwojem systemów komputerowych producenci oprogramowania komputerowego udostępniają użytkownikom ciągle nowe programy, mające ogromne możliwości zarządzania danymi. Powoduje to przeplatanie się technologii oraz częste łączenie ich w sposób nieoczywisty, jak np.: w przypadku Serwera SQL stworzonego przez Microsoft. W SQL Server 2008 zaimplementowano typ umożliwiający zapis w bazie danych grafiki wektorowej za pomocą przestrzennych typów danych nazwany Geometry. Typy te pozwalają zapisać do pól tabel informacji o kształcie, położeniu oraz rozmiarze obiektów w przestrzeni. Microsoft stworzył także wiele funkcji umożliwiających operowanie na danych o typie Geometry. Umożliwiają one np.: obliczanie odległości pomiędzy obiektami, znalezienie środka ciężkości figury lub jej pola powierzchni. Typ Geometry został stworzony głównie z myślą o tworzeniu map.

Grafika wektorowa jest sposobem zapisu obrazów (obiektów) za pomocą współrzędnych poszczególnych elementów w przestrzeni

kartezjańskiej. Przykładowe formaty plików grafiki wektorowej to DXF, DWG, CDR lub SVG. Pliki DXF są formatem stworzonym przez firmę Autodesk i stosowanym przez oprogramowanie Autocad. Pliki te zawierają tekstowy opis wszystkich obiektów zawartych w rysunku. Microsoft dostarczył do tego specjalny typ Geometry umożliwiający przechowanie obiektów graficznych w bazie danych MS SQL Server. Dane zawarte w typie Geometry opisują obiekty zapisane w płaskiej przestrzeni dwuwymiarowej.

Celem publikacji jest opisanie sposobu tworzenia aplikacji w języku C # umożliwiającej zapis obrazów z pliku DXF do bazy danych MS SQL Server w formacie Geometry oraz późniejszy odczyt stworzonych obrazów i skonstruowanie z nich pliku DXF. Program odczyta cały plik tego typu, wybierze z niego wszystkie obiekty mające reprezentację graficzną, wykona na nich operacje matematyczne tak, aby wiernie je zapisać do odpowiednich pól tabeli. Zapis rysunku do pliku polega na wybraniu odpowiednich obiektów z tabeli, sformatowaniu ich oraz zapisie do pliku DXF.

2 Opis plików grafiki wektorowej

Pliki DXF(ang. Data Exchange Format) to pliki tekstowe zapisane w formacie ASCII, zawierające tekstową interpretacje rysunków wektorowych wykonanych w programie Autocad firmy Autodesk. Początkowo format ten służył do wymiany danych pomiędzy Autocade'em a 3D Studio. Format jednak zyskał większą popularność i jest dziś znacznie szerzej stosowany, głównie ze względu na łatwość konstruowania plików oraz na udostępnioną przez firmę Autodesk dokumentację. W stosunku do binarnego formatu stosowanego domyślnie przez Autocad'a (DWG) pliki formatu DXF zajmują większą ilość miejsca w pamięci oraz są wolniej przetwarzane.

Plik DXF składa się z par linii, przy czym w linii nieparzystej znajduje się kod opisujący znaczenie wartości w znajdującej się niżej linii parzystej. Używając tych par plik DXF jest złożony z sekcji, które składają się z kodów i wartości przez nie opisanych. Wyróżnia się następujące sekcje pliku DXF:

- HEADER – sekcja zawierająca zestaw zmiennych dotyczących rysunku np.: format wersji Autocada, w której utworzono plik, rodzaj jednostek miary(metryczne czy calowe) itd.
- CLASSES – sekcja zawierająca dane dla klas zdefiniowanych w aplikacji, których instancje występują w innych sekcjach pliku.
- TABLES – sekcja zawierająca zestaw tablic, w których opisane są standardowe elementy rysunku takie jak np.: typy linii czy warstwy.

- BLOCKS – sekcja składająca się z bloków rysunku.
- ENTITIES – sekcja zawierająca spis wszystkich obiektów mających reprezentację graficzną np.: linia, polilinia, punkt.
- OBJECT – sekcja składająca się z obiektów nieposiadających interpretacji graficznej.
- END OF FILE – oznacza koniec pliku DXF.

Przykładowy fragment pliku DXF opisujący linie:

```
LINE
  5
129
  8
LAYER1
 10
14.97
 20
24.90
 30
0.0
 11
50.0
 21
50.0
 31
0.0
```

Powyższy przykład należy interpretować następująco: kod 5 oznacza uchwyt danego obiektu i wynosi 129 (jest to wartość hexadecymalna), kod 8 to nazwa warstwy w tym przypadku „LAYER1”, kod 10 to wartość pierwszego punktu X i równa 14.97. Następnie 20 i 30 to kolejne kody oznaczające wartość Y i Z równe odpowiednio 24.90 oraz 0. Kody 11, 21, 31 oznaczają końcowe punkty linii. Kody 30 i 31 równe zero oznaczają rysunek dwuwymiarowy.

W tym opracowaniu najważniejszą sekcją jest ENTITIES, ponieważ w niej właśnie zapisane są wszystkie obiekty mające graficzne znaczenie. Poniżej przedstawiono listę zawierającą opis obiektów rozpoznawanych przez stworzoną aplikację:

- 3DFACE – powierzchnia trójwymiarowa.
- 3DSOLID – bryła trójwymiarowa.
- ARC – łuk, opisany za pomocą następujących danych:
  - Środek (kody 10,20,30) – środek okręgu, na którym łuk jest opisany.
  - Promień (kod 40) – promień okręgu, na którym łuk jest opisany.
  - Kąt początkowy łuku (kod 50).

  - Kąt końcowy łuku (kod 51).
- ATTDEF – definicja atrybutu obiektu.
- ATTRIB – atrybut obiektu.
- CIRCLE – koło, opisane w następujący sposób:
  - Środek (kody 10,20,30) – środek okręgu.
  - Promień (kod 40) – promień okręgu.
- DIMMENSION – obiekt opisuje wymiarowanie rysunku.
- ELLIPSE – obiekt opisujący elipsę.
- HATCH – wzór wypełniający powierzchnię obiektów.
- HELIX – obiekt o kształcie spirali.
- IMAGE – obrazek zapisany pliku.
- INSERT – umożliwia wstawienie obiektu
- LEADER – odnośnik
- LIGHT – obiekt reprezentujący oświetlenie obrazu.
- LINE – linia, opisana za pomocą następujących informacji:
  - Punkt początkowy (kody 10,20,30).
  - Punkt końcowy (kody 11,21,31)
- LWPOLYLINE – linia w przestrzeni dwuwymiarowej
- MESH- siatka na powierzchni obiektu.
- MLINE – obiekt składający się z wielu połączonych końcami linii prostych
- MLEADERSTYLE – zbiór stylów odnośników
- MLEADER – obiekt składający się z wielu odnośników
- MTEXT – tekst umieszczony na rysunku.
- OLEFRAME – okno zawierające obiekt OLE – służy do komunikacji i pracy z wieloma aplikacjami.
- OLE2FRAME - obiekt przechowuje obiekt typu OLE2 (obiekt rozszerzony w porównaniu do OLE o mechanizm „drag and drop”
- POINT – punkt (kody 10,20,30) oznaczające współrzędne punktu.
- POLYLINE – polilinia, opisana za pomocą przykładowych informacji:
  - Kody 10,20,30 zawsze równe zero.
  - Flaga polilinii (kod 70) oznaczająca np.:, że polilinia jest zamknięta – wartość równa 1.
  - Domyślna początkowa szerokość (opcjonalne, kod 40), wartość domyślna równa zero.
  - Domyślna końcowa szerokość (opcjonalne, kod 41), wartość domyślna równa zero.
- RAY – nieskończenie długa linia konstrukcyjna.
- REGION – zamknięty dwuwymiarowy obszar.
- SECTION – sekcja
- SEQEND – koniec sekwencji obiektów, np.: polilinii.
- SHAPE – umożliwia wstawienie obiektu

- SOLID – bryła trójwymiarowa.
- SPLINE – linia krzywa.
- SUN – obiekt reprezentujący oświetlenie słoneczne.
- SURFACE – powierzchnia.
- TABLE – tabela.
- TEXT – tekst umieszczony na rysunku.
- TOLERANCE – obiekt przechowujący informację o tolerancjach naniesionych na rysunek.
- TRACE – ślad, trasa
- UNDERLAY – podkreślenie
- VIEWPORT – widoki, używane głównie przy rysowaniu rysunków.
- VERTEX – kąt, obiekt opisujący fragmenty linii, z których składa się polilinia, opisany za pomocą następujących danych:
  - Punkt początkowy (kody 10,20,30)
  - Bulge – współczynnik łuku, wartość opisana jako: „tangens jednej czwartej kąta zawartego w segmencie łuku". Gdy ta wartość jest większa od zera łuk jest prawoskrętny, natomiast wartość ujemna oznacza łuk lewoskrętny względem punktu początkowego. Współczynnik łuku równy 1 oznacza, że łuk jest półkolem.
- XLINE – uproszczona linia prosta, zawiera tylko punkt początkowy i końcowy

3 Opis typu Geometry

Typ Geometry (przestrzenny typ danych) przechowuje informacje o położeniu, kształcie i rozmiarach obiektów znajdujących się w płaskiej przestrzeni dwuwymiarowej. Typ ten jest zaimplementowany jako element CLR w SQL Server 2008 posiada szereg metod umożliwiających operacje na obiektach oraz ich analizę. Standardem tym opiekuje się OGC – Open Geospatial Consorcium, międzynarodowa organizacja działająca charytatywnie, dostarczająca standardy geograficzno-przestrzenne. Dane w tym formacie mogą być definiowane na 3 sposoby:

- WKT (ang. Well Known Text) – jest tekstem złożonym ze znaczników służącym do definiowania obiektów w formacie Geometry. Przykładowe definicje obiektów:
  - Punkt: POINT(40 55)
  - Linia: LINESTRING(40 50, 60 75)
- WKB (ang. Well Known Binary) – dane o obiektach zapisane w formacie binarnym. Służą do przechowywania informacji w bazie danych oraz do wymiany danych pomiędzy bazami danych. Przykładowe dane zapisane w formacie binarnym:

  - 0x000000000114000000000000144000000000000014400000000000001440000002440
    są to dane zapisane jako liczby hexadecymalne, natomiast tak wygląda ten sam obiekt w formacie WKT: LINESTRING (5 5, 5 10)

- GML (ang. Geography Markup Language) – jest to format zapisu danych używających gramatyki XML. Został stworzony przez OGC, aby przyśpieszyć i ułatwić dostęp do danych typu Geometry oraz aby umożliwić dostęp do nich za pomocą Internetu. Oto przykład linii zapisanej w formacie GML:

```
<gml:LineString gml:id="p21"
   srsName="urn:ogc:def:crs:EPSG:6.6:4326">
      <gml:coordinates>45.67, 88.56 55.56,89.44
      </gml:coordinates>
</gml:LineString >
```

Jest to linia o punkcie początkowym o współrzędnych 45.67 i 88.56 oraz punkcie końcowym o współrzędnych 55.56 i 89.44.

Typy obiektów wspieranych przez SQL Server 2008:

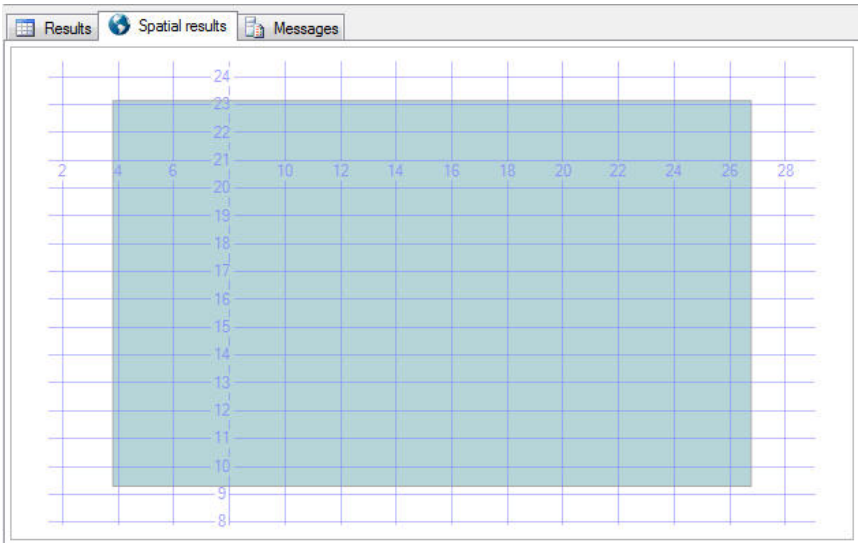
- Punkt – Point przykład definicji za pomocą WKT: POINT(50 50)
- Linia – Linestring przykład definicji za pomocą WKT: LINESTRING(40 50, 70 150);
- Wielokąt – Polygon przykład definicji za pomocą WKT: POLYGON((5 5, 5 10, 10 10, 10 5, 5 5))
- Zbiór linii – Multilinestring przykład definicji za pomocą WKT: MULTILINESTRING((1 6, 11 5, 20 0)
- Zbiór wielokątów – Multipolygon przykład definicji za pomocą WKT: MULTIPOLYGON(((5 5, 11.6 9 , -2 -2,0.7 100, 5 5))
- Zbiór punktów – Multipoint przykład definicji za pomocą WKT: MULTIPOINT(5 9.6, 11.8 2)
- Kolekcja figur geometrycznych – GeometryCollection przykład definicji za pomocą WKT:
- GEOMETRYCOLLECTION(POINT(100 76.5),LINESTRING(11 9,4 14))

W typie geometry zaimplementowano dużą liczbę metod umożliwiających operację na danych. Metody te możemy podzielić na trzy grupy:

- Metody podstawowe – jest to zestaw podstawowych metod pozwalają wykonać proste operacje na obiektach można za ich pomocą uzyskać o nich informacje np.: wymiary, typ obiektu (linia, punkt itd.) lub identyfikator obiektu.

- Metody statyczne – zwracają obiekt z teksu zapisanego w formacie WKT.
- Metody rozszerzone – umożliwiają wykonywanie zaawansowanych testów na obiektach oraz ich konwersji.

Przykładem metod podstawowych jest STArea, która zwraca pole powierzchni zadanego obiektu. Załóżmy, że mamy dany prostokąt o identyfikatorze id_obi równym 11 jak na rysunku poniżej:



Rys. 1. Prostokąt zapisany w bazie danych w formacie Geometry

Metodę wywołujemy w następujący sposób:

```
DECLARE @a GEOMETRY;
SELECT @a = GeomCol2 FROM obiekty
WHERE id_obi = 11
SELECT @a.STArea() as Pole;
```

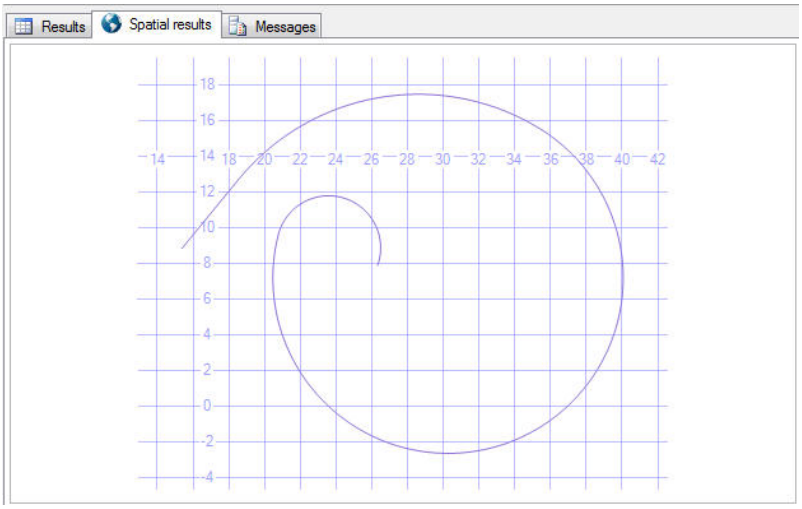
Rezultatem jej działania jest pole prostokąta równe 317,911633352599.

Przykładem metody rozszerzonej może być metoda Reduce, która zwraca obiekt uproszczony zgodnie z algorytmem Douglas-Peucker'a. Jako parametr funkcji przekazywany jest stopień aproksymacji. Wywołanie tej metody może wyglądać następująco:

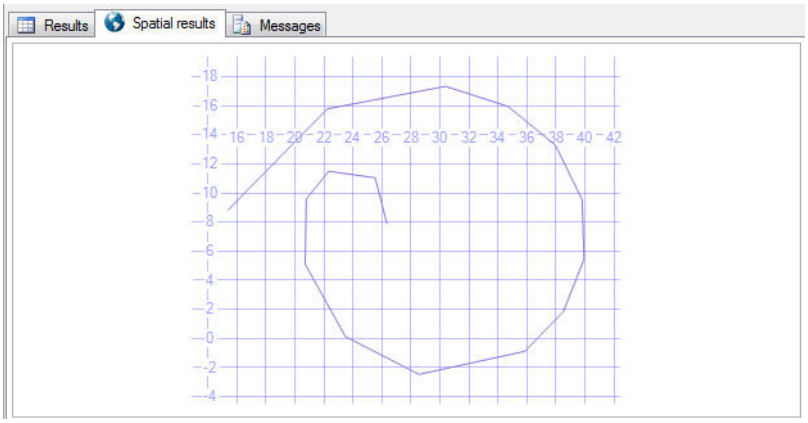
```
DECLARE @o GEOMETRY
DECLARE @g GEOMETRY
SELECT @g = GeomCol2  FROM obiekty
WHERE id_obi = 12
```

```
SELECT @o= @g.Reduce(0.75).ToString();
SELECT @o
```

Metoda ta została wywołana na obiekcie jak na rysunku 2, a rezultatem jej działania jest obiekt pokazany na rysunku 3



Rys. 2. Krzywa zapisana w bazie danych w formacie Geometry



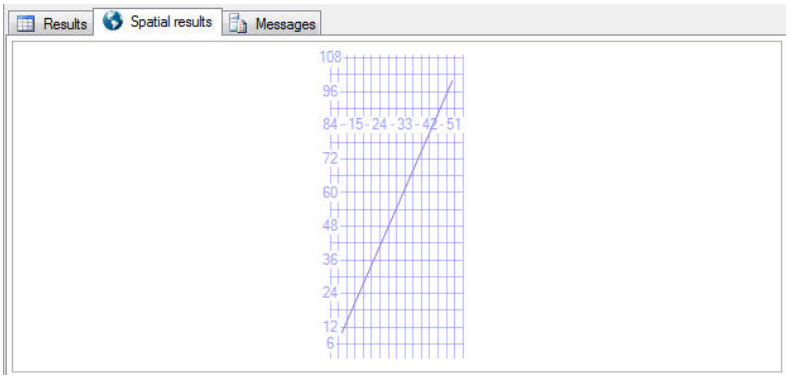
Rys. 3. Krzywa będąca rezultatem działania metody „Reduce”

Statyczna metoda STLineFromText tworzy obiekt typu geometry z tekstu przekazanego jako parametr. Oto przykładowe wywołanie tej metody:

```
DECLARE @a GEOMETRY
```

```
SET @a = Geometry::STLineFromText('Linestring(10
10, 50 100)',0)
SELECT @a
```

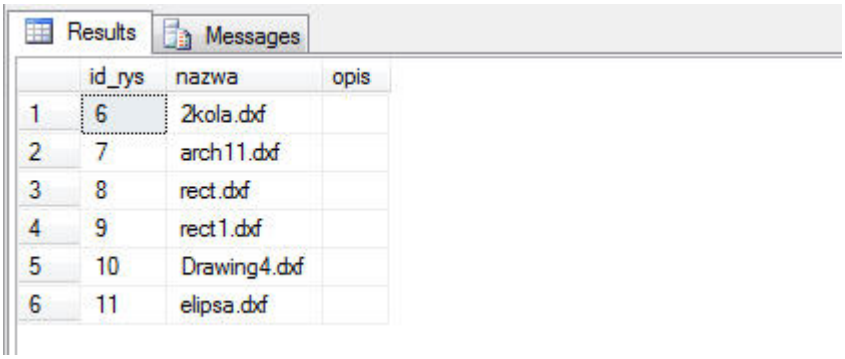
Rezultat wywołania tej metody przedstawia rysunek 4:



Rys. 4. Linia będąca rezultatem działania funkcji STLineFromText.

4 Opis bazy danych

Baza danych została oparta na serwerze baz danych stworzonym przez firmę Microsoft – SQL Server 2008. W bazie stworzone zostały 2 tabele: rysunek i obiekty. Tabela rysunek składa się z dwóch kolumn – rysunek 5. Pierwsza kolumna nazwana Id_rys jest kluczem podstawowym z nadanym atrybutem autonumerowania. Druga kolumna przechowuje nazwę rysunku i ma nadany atrybut unique, co powoduje, że nazwy rysunków nie mogą się powtarzać.



	id_rys	nazwa	opis
1	6	2kola.dxf	
2	7	arch11.dxf	
3	8	rect.dxf	
4	9	rect1.dxf	
5	10	Drawing4.dxf	
6	11	elipsa.dxf	

Rys. 5. Przykładowa zawartość tabeli „rysunek”

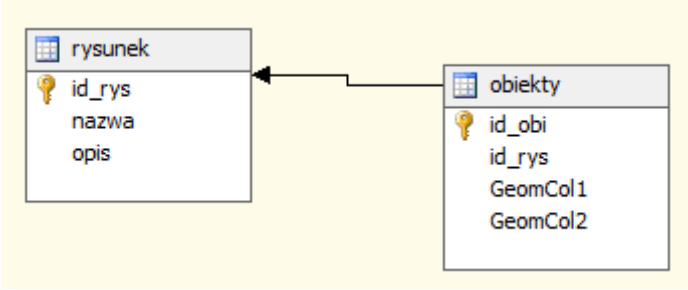
Tabela obiekty składa się z czterech kolumn. W pierwszej kolumnie o nazwie Id_rys zapisany jest numer identyfikacyjny rysunku, do którego dany obiekt należy. Kolejna kolumna opisana Id_obiektu zawiera unikalny numer obiektu z autonumerowaniem. Identyfikator obiektu nie jest używany w bazie danych, został dodany, aby umożliwić późniejszą rozbudowę bazy danych lub rozwój aplikacji. Kolejna kolumna nosi nazwę GeomCol1 i zawiera informacje o obiekcie typu geometry zapisane w postaci binarnej WKB (Well Known Binary). Ostatnia kolumna o nazwie GeomCol2 także zawiera informacje typu Geometry, ale w postaci tekstowej WKT (Well Known Text) – rysunek6.

Results | Spatial results | Messages

	id_rys	id_obi	GeomCol1	GeomCol2
1	6	6	0x0000000001046901000009EE4112D...	LINESTRING (310.990521497626 777.731648950595, 3...
2	8	10	0x0000000001040500000024BFCA06...	POLYGON ((288.617926399211 714.417961423549, 102...
3	9	11	0x0000000001040500000080427667...	POLYGON ((3.79824333982134 23.1413471316369, 26....
4	10	12	0x0000000001049203000008B53A7BD...	LINESTRING (15.3959330813175 8.83368064327421, 1...
5	6	7	0x0000000001046901000007C908383...	LINESTRING (192.188661343538 775.048896027921, 1...
6	6	8	0x0000000001046901000000334E655E...	LINESTRING (373.036222835278 413.553544461869, 3...
7	7	9	0x000000000104E400000087747368...	LINESTRING (108.760461914777 189.106001202135, 1...
8	11	13	0x0000000001048100000009FCDAACF...	LINESTRING (8.26335 18.7969, 8.25284 18.6308, 8.263...

Rys. 6. Przykładowa zawartość tabeli „obiekty”.

Pomiędzy tabelami nadana została relacja jeden do wielu, co oznacza, że do jednego unikalnego identyfikatora rysunku (Id_rys) może być przypisane nieskończenie wiele obiektów w nim zawartych – rysunek 7.

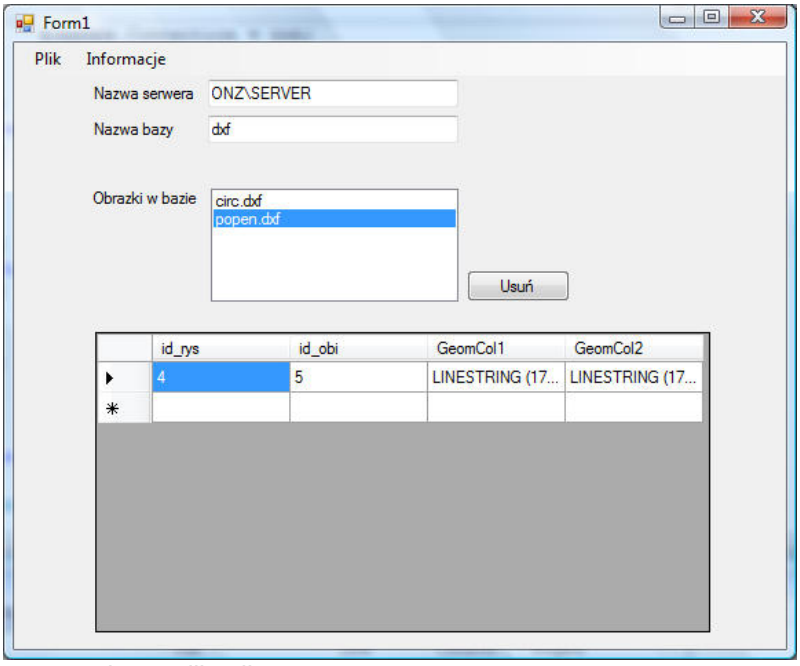


Rys. 7. Tabele i relacje między nimi.

5 Opis interfejsu graficznego aplikacji

Na interfejs graficzny aplikacji składa się jedno główne okno, które zawiera menu rozwijane, dwa pola tekstowe, pole listy, przycisk o

nazwie Usuń oraz pole tabeli - rysunek 8. Pola tekstowe służą do wpisana nazwy bazy danych oraz adresu serwera, do którego program ma się podłączyć i z którego odczytywać i zapisywać pliki. Menu rozwijane składa się z dwóch list rozwijanych: Plik oraz Informacje. W liście plik znajdują się polecenia Otwórz, Zapisz oraz Koniec. Lista Informacje zawiera pole O mnie. Kliknięcie polecenia Otwórz powoduje wyświetlenie okna dialogu pozwalającego wybrać plik, który zostanie zapisany w bazie danych. Okno dialogowe ma dodany filtr, który domyślnie powoduje wyświetlenie jedynie plików o rozszerzeniu DXF. Polecenie Zapisz powoduje otwarcie okna dialogowego oczekującego na podanie nazwy pliku, do którego zostanie zapisany zaznaczony rysunek. Polecenie Zakończ kończy działania programu. Przycisk Usuń służy do usuwania rysunków z bazy danych. Aby skasować rysunek należy zaznaczyć go w polu listy i kliknąć przycisk Usuń, w bazie danych skasowane zostaną wszystkie obiekty danego rysunku z tabeli obiekty a następnie zostanie usunięty wpisz z tabeli rysunek. Zaznaczenie nazwy rysunku w polu listy powoduje wyświetlenie wszystkich jego obiektów w tabeli DataGridView umieszczonej w oknie głównym. Tabela ma ustawiony atrybut tylko do odczytu, aby uniemożliwić zmianę danych. Mimo tego ewentualne zmiany nie zostały by zapisane, ponieważ nie ma w aplikacji metody uaktualniającej informacje w bazie danych.



Rys. 8. Okno aplikacji

6 Zapis pliku do bazy danych

Odczyt pliku zaczyna się poprzez rozwinięcie menu Plik i wybranie polecenia Open. Zostaje wtedy otwarte okno dialogowe, w którym można wybrać plik. Dla tego okna dialogowego został nadany filtr, powodujący wyświetlenie w nim tylko plików z rozszerzeniem DXF. Jeżeli okno jako rezultat swojego działania zwróci wartość DialogResult.OK zostaje wywołana metoda polacz(). Metoda ta nawiązuje połączenie z bazą danych. Kolejnym krokiem jest wywołanie metody zapisującej plik do bazy danych o nazwie wstaw_plik. Metoda ta jako parametr przyjmuje nazwę pliku a zwraca liczbę całkowita, która jest identyfikatorem rysunku w bazie danych – id_rys. Scemat przetwarzania danych podczas zapisu do bazy przedstawia rysunek 9.

Dalej program wykorzystując klasę StreamReader odczytuje otwarty plik:

```
using(StreamReader sr = new
StreamReader(openFileDialog1.OpenFile()))
```

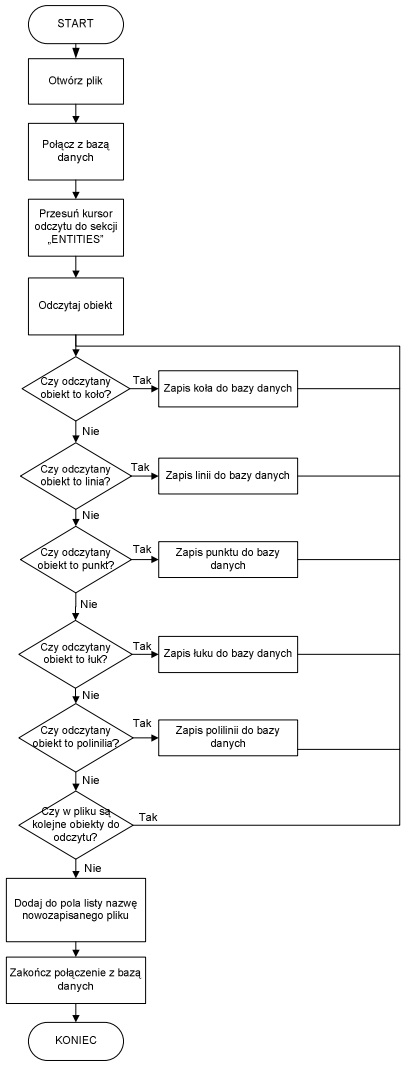
Odczyt pliku wykonywany jest linijka po linijce, aż do napotkania linijki o kodzie ENTITIES:

```
if (tekst == "ENTITIES") flaga = true;
```

Wtedy globalnie zdeklarowana zmienna logiczna o nazwie flaga zostaje ustawiona na wartość true. Oznacza to, że kursor odczytu pliku znajduje się w sekcji, w której zapisane są obiekty graficzne. Odczytywana jest kolejna linijka z pliku a jej wartość porównywana jest z rozpoznawanymi przez aplikację obiektami. Jeżeli nastąpiło trafienie program odczytuje kolejne linijki pliku oraz interpretuje ich znaczenie i wartości tak, aby możliwy był ich zapis w bazie danych.

Najbardziej skomplikowanym obiektem odczytywanym przez aplikacje jest polilinia. Do odczytu polilinii aplikacja używa globalnie zadeklarowanych kolejek typu FIFO o nazwach punkty i typ oraz dwóch zmiennych typu boolowskiego o nazwie luk i zamknięta. Kolejka punkty przechowuje kolejne, odczytane z pliku elementy polilinii natomiast kolejka typ zawiera kody opisujące elementy w kolejce punkty. Pierwszą czynnością wykonywaną po trafieniu na definicję polilinii jest sprawdzenie czy jest zamknięta, tzn. czy jej początek łączy się z końcem. Wykonywane jest to w pętli do, która odczytuje wartość przechowywaną pod kodem o numerze 70, jeżeli jego wartość jest równa jeden to polilinia jest zamknięta. Gdy kod 70 zawiera wartość równą zero, polilinia jest otwarta. Odpowiednia wartość jest przypisywana zmiennej o nazwie zamknięta i program przechodzi do

kolejnej pętli. Jest to pętla while, której warunkiem końca jest trafienie na obiekt o nazwie SEQEND oznaczający koniec sekwencji polilinii.



Rys. 9. Schemat blokowy przedstawiający odczyt danych z pliku DXF.

W pętli tej odczytywane są kolejne współrzędne oraz kody punktów opisujących daną polilinię i dodawane do odpowiednich kolejek. Po zakończeniu odczytu punktów z pliku następuje ich obróbka, w tym celu zdefiniowane zostały dwie tablice typu double o nazwach arr_p i arr_t. Do tablic skopiowane zostają kolejki punkty i kod. Operacja ta jest niezbędna, ponieważ po kolejkach nie można się poruszać do tyłu, raz zdjęta wartość zostaje z niej usunięta. Następnie sprawdzona zostaje

zawartość zmiennej zamknięta, jeżeli jest ona ustawiona na tru" oznacza, że polilinia jest zamknięta i do bazy danych zostanie zapisany wielokąt zamknięty (poligon), jeżeli polilinia jest otwarta do bazy danych zapisana zostanie multilinia.

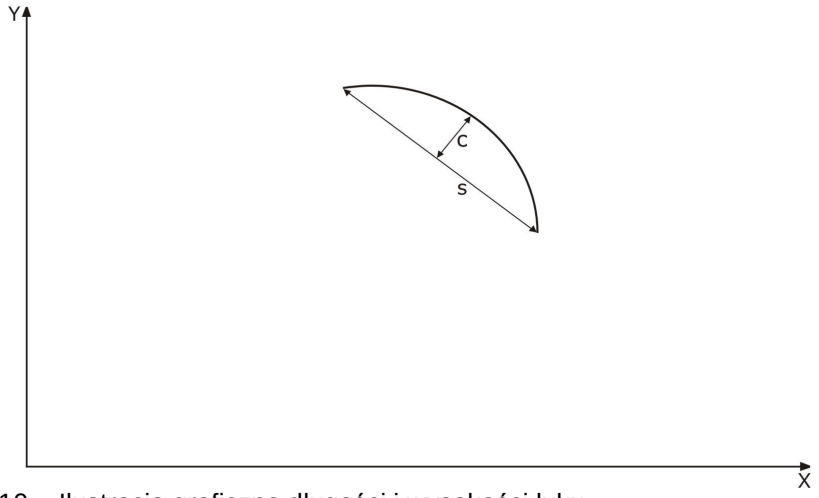
Listing 1. Fragment kodu metody zapisu polilinii

```
if (tekst == "POLYLINE" && flaga)
   {
   string odczyt;
   luk = false;
   bool zamknieta = false;
   while (sr.ReadLine().Replace(" ", "") != "8") ;
   nazwa_warstwy = sr.ReadLine().Replace(" ", "");
   do
   {
      odczyt = sr.ReadLine().Replace(" ", "");
      if (odczyt == "70")
      {
         if (sr.ReadLine().Replace(" ", "") == "1")
zamknieta = true;
      }
   } while (odczyt != "VERTEX");
   while ((odczyt = sr.ReadLine()) != "SEQEND")
   {
      if ((odczyt.Replace(" ", "")) == "10")
      {
         …
      }
      if ((odczyt.Replace(" ", "")) == "42")
      {
         …
      }
   }
double[] arr_p = new double[punkty.Count];
punkty.CopyTo(arr_p, 0);
string[] arr_t = new string[typ.Count];
typ.CopyTo(arr_t, 0);
if (zamknieta)
   {
   string k1, k2, k3 = null, k5 = null;
   k1 = "insert into obiekty( id_rys,GeomCol1) values
( " + id__rys;
   k2 = " , geometry::STGeomFromText('POLYGON((";
   …
```

```
   while (licznik < dl_kol)
   {
      …
      t_odcz = arr_t[licznik];
      p_odcz = arr_p[licznik];
      xp1 = p_odcz;
      licznik++;
      …
   }
   punkty.Clear();
   typ.Clear();
   string komlinia = k1 + k2 + k3 + k5;
      SqlCommand koml = new SqlCommand(komlinia, con);
   koml.ExecuteNonQuery();
}
```

Ponieważ oba przypadki są bardzo podobne pod względem implementacji rozważymy tylko przypadek, gdy polilinia jest zamknięta. Na początku zdeklarowane zostaje pięć zmiennych pomocniczych typu string, pierwsze dwie zostają zainicjalizowane częścią polecenia SQL odpowiedzialnego za wstawienie do bazy danych poligonu w formacie Geometry z parametrem id_rys, który zawiera numer identyfikacyjny rysunku, do którego należy opisywana polilinia. Dalej do dwóch globalnie zdeklarowanych zmiennych x0 i y0 przypisane są początkowe punkty poligonu, zostaną one dopisane jako ostatnie punkty w deklaracji poligonu zapisywanego w bazie danych (typ Geometry wymaga, aby początkowe i końcowe punktu poligonu były takie same). Następnie w pętli while odczytywane są kolejne punkty z tablic arr_p i arr_t. Jeżeli są to sekwencje kodów 10 i 20 oznacza to, że wielobok jest zbudowany z linii prostych. We współrzędnych tych linii zamieniany jest znak przecina na kropkę i kolejne punkty są dopisywane do zmiennej typu znakowego o nazwie k3, jeżeli odczytany kod będzie równy 42 oznacza to, że elementem wieloboku jest łuk. Wartość opisywana przez kod 42 definiowana jest jako współczynnik łuku i definiowany jako jedna czwarta tangensa kąta ϑ (teta) opisanego na nim.

Gdy aplikacja trafi na łuk, dopisuje jego punkt początkowy do zmiennej pomocniczej k3. Następnie obliczona jest długość i wysokość łuku. Przez wysokość łuku rozumiemy odległość środka linii łączącej końce luku i jego wierzchołka. Na rysunku poniżej wysokość jest zaznaczona jako C, natomiast długość łuku została oznaczona jako S.



Rys. 10. Ilustracja graficzna długości i wysokości łuku.

Mając obliczoną wysokość i długość możemy obliczyć promień łuku. Następnie obliczamy środek koła opisanego na łuku. Kolejnym krokiem jest określenie kierunku rysowania łuku, mówi o tym znak kąta, jeżeli jest dodatni to łuk jest prawoskrętny natomiast ujemny kąt oznacza łuk lewoskrętny. Używając tych danych oraz wzorów do obliczenia obrotu punktu o kąt aplikacja oblicza współrzędne kolejnych punktów, które zostaną połączone odcinkami. Współrzędne te są konkatenowane z uprzednio zdefiniowaną zmienną (k3) typu string i utworzą łuk. Dalej, jeżeli w tablicach zostały jakieś wartość aplikacja odczytuje je, interpretuje i ponownie dołącza do zmiennej k3, tak aby do bazy danych została zapisana cała zadana polilinia. Po odczytaniu ostatniego kodu i punktów z tablicy aplikacja łączy wszystkie zdefiniowane wcześniej zmienne pomocnicze w jedną komendę języka SQL o nazwie komlinia, tworzy instancję klasy SqlCommand o nazwie koml. W jej konstruktorze przekazywana jest zmienna komlinia oraz obiekt con zawierający bieżące połączenie z bazą danych. Następnie wywoływana jest metoda ExecuteNonQuery obiektu koml powodująca zapis wielokąta do bazy danych.

7 Zapisu danych z bazy do pliku w formacie DXF

Zapis rysunku do pliku rozpoczyna się od wybrania nazwy pliku z listy umieszczonej w oknie aplikacji, następnie otwarcia menu Plik i wybrania polecenia Zapisz. Aplikacja otwiera okno dialogowe, oczekujące podania przez użytkownika nazwy pliku oraz wybrania folderu docelowego. Domyślnym rozszerzeniem jest DXF, jednak użytkownik może podać nazwę pliku z innym rozszerzeniem. Jeżeli nazwa pliku została podana i

jej długość jest różna od zera plik zostaje utworzony. Następnie aplikacja tworzy instancję klasy StreamWrite o nazwie plik. Do pliku zapisywane są standardowe, domyślne linie definiujące wszystkie wymagane przez format DXF sekcje. Ostatnią sekcją, które jest zapisywana i pozostaje otwarta jest ENTITIES. Dalej aplikacja w pętli foreach odczytuje kolejno wszystkie rzędy z pierwszej tabeli znajdujące się w obiekcie typu dataset o nazwie ds. Obiekt ten zawiera wszystkie obiekty zapisane w bazie danych należące do zaznaczonego uprzednio na liście pliku.

Następnie inkrementowana jest zdefiniowana globalne zmienna typu całkowitego o nazwie handle przechowujące uchwyt (unikalny identyfikator obiektu w pliku DXF). Do zmiennej o nazwie geom_col przypisywany jest aktualnie odczytany rząd z tabeli. Następnie definiowana jest tablica typu string o nazwie tablica, za pomocą metody delim zmienna geom_col jest dzielona i odpowiednie jej fragmenty zostają przypisane do kolejnych elementów tablicy. Po tej operacji w zmiennej tablica element o indeksie zero zawiera nazwę obiektu w niej zawartego.

Listing 2. Realizacja zapisu obiektu do pliku DXF

```
if (tablica[0] == "POLYGON")
   {
   plik.WriteLine("POLYLINE");
   plik.WriteLine("  5");
    plik.WriteLine(handle.ToString("X"));
   plik.WriteLine("  8");
   plik.WriteLine("0");
   plik.WriteLine(" 66");
   plik.WriteLine("     1");
   plik.WriteLine(" 10");
   plik.WriteLine("0.0");
   plik.WriteLine(" 20");
   plik.WriteLine("0.0");
   plik.WriteLine(" 30");
   plik.WriteLine("0.0");
   plik.WriteLine(" 70");
   plik.WriteLine("     1");
   plik.WriteLine("  0");

   for (int i = 2; i < tablica.Length - 2; i++)
   {
      handle++;
      plik.WriteLine("VERTEX");
      plik.WriteLine("  5");
```

```
    plik.WriteLine(handle.ToString("X"));
    plik.WriteLine("  8");
    plik.WriteLine("0");
    plik.WriteLine(" 10");
    plik.WriteLine(Convert.ToString(tablica[i]));
    i++;
    plik.WriteLine(" 20");
    plik.WriteLine(Convert.ToString(tablica[i]));
    plik.WriteLine(" 30");
    plik.WriteLine("0.0");
    plik.WriteLine("  0");
  }
  handle++;
  plik.WriteLine("SEQEND");
  plik.WriteLine("  8");
  plik.WriteLine("0");
  plik.WriteLine("  5");
  plik.WriteLine(handle.ToString("X"));
  plik.WriteLine("  0");
  }
```

Najpierw aplikacja zapisuje do pliku standardowy początek opisujący linię. Najważniejszym elementem jest uchwyt (handle) inkrementowany przy każdym kolejnym obiekcie o jeden i zapisywany do pliku w formacie liczby szesnastkowej. Dalej zapisywane są poszczególne fragmenty składowe linii aż do osiągnięcia ostatniego wiersza tablicy. Jako ostatni zapisywany jest obiekt o nazwie SEQEND oznaczający koniec sekwencji, czyli zarazem zapisywanej linii. Warto zauważyć, że obiekt SEQEND także ma swój unikalny w skali pliku uchwyt.

Podobnie wygląda zapis do pliku DXF wieloboku (poligonu). Obiekt ten jest zapisywany jako polilinia. O tym, że jest to figura geometryczna posiadająca pole powierzchni mówi kod o numerze 70 i jego wartość ustawiona na 1. Program odczytujący tak zapisany plik DXF automatycznie połączy koniec i początek obiektu tworząc wielobok zamknięty.

Ostatnia operacja przy zapisywaniu danych do pliku to jego zamknięcie. Na końcu dodawany jest kod ENDSEC oznaczający zakończenie sekcji ENTITES oraz znacznik końca pliku EOF. Następnie plik jest zamykany poprzez wywołanie metody Close należącej do instancji klasy StreamWriter o nazwie plik.

8 Podsumowanie

Stworzona aplikacja w pełni wykonuje założone zadanie. Z zadanego pliku wybierane są wszystkie obiekty posiadające interpretację graficzną, wykonywane są na nich niezbędne operacje matematyczne, aby skonwertować je do formatu Geometry i ostatecznie zostają zapisane do bazy danych. Najbardziej skomplikowanym obiektem okazała się być polilinia, a właściwie jej część opisana jako bulge. Jest to wartość opisująca współczynnik łuku, czyli krzywa, która wymagała skomplikowanych przekształceń, aby zapisać ją w formacie Geometry. Kolejnymi obiektami, które sprawiły kłopoty autorowi są łuk i koło. W typie Geometry nie zawarto koła ani linii krzywej. Spowodowało to konieczność aproksymacji wszystkich elementów zawierających krzywe (łuk, koło, polilinia) za pomocą odcinków linii prostej. Proces ten spowodował znaczące różnice w typach obiektów zapisywanych i odczytywanych z bazy danych. Załóżmy, że do bazy danych zapisujemy obrazek zawierający jeden okrąg. W trakcie zapisu, zostanie on przekształcony w zbiór odcinków, tak, więc utraci ważną właściwość – pole powierzchni. Następnie, gdy spróbujemy zapisać ten sam obraz do pliku DXF zostanie także zapisany jako zbiór odcinków, nie będzie już możliwości edytowania go jako koła. Aproksymacja wymaga wielu obliczeń matematycznych, zależnych od obiektu zadanego, jednak każde obliczenia na liczbach zmiennoprzecinkowych mogą powodować błędy zaokrąglenia i obcięcia. Powoduje to zmiany w wymiarach i położeniu obiektów na płaszczyźnie. Różnice te jednak pojawiają się dopiero przy bardzo dużych rozmiarach obiektów, lub przy bardzo dużych powiększeniach. Mimo tych trudności po przeprowadzeniu wielu testów można stwierdzić, że założony cel został osiągnięty. Stworzona aplikacja wiernie odtwarza odczytywane i zapisywane obiekty. W bazie danych obiekty formatowane są poprawnie i można na nich wykonywać wszystkie metody udostępnione przez typ Geometry.

Literatura

[1] „DXF Reference” Autodesk 2011

[2] Open Geospatial Consortium:

- OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture
- OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option
- Simple Features - SQL - Binary Geometry (1.1)
- Simple Features - SQL - Normalized Geometry (1.1)
- Simple Features - SQL - Types and Functions (1.1)

[3] Harrington J., Relational Database Design and Implementation. 2009.

[4] Lobel L., Programming Microsoft SQL Server 2008. Microsoft Press, 2009.

[5] Aitchison A., Beginning Spatial with SQL Server 2008.

[6] Pedersen J., Beginning C# 2008 Databases: From Novice to Professional. 2009

[7] Troelsen A., Język C# 2008 i platforma NET 3.5. PWN

[8] Gross C., Beginning C# 2008: From Novice to Professional. 2009

[9] Brent Hall G., Spatial Database Systems: Design, Implementation and Project Management.2009

WRITING VECTOR GRAPHIC FILES TO MS SQL DATABASE

Summary – Article is devoted to the use of the database server to store vector graphics. Original element of the proposed solution is to store the components of this type of data, instead of what used to be used, save the file in a BLOB types. Object-type geometry was used, whereas an environment was MS SQL Server. Achieved decomposition of the most popular vector graphics file *. pdf to database objects. As a verification of the quality of the solutions implemented to write data from the database to the output format. The proposed solution can be used for storing and reusing blocks for creating complex vector drawings and documentation.