Geometry en geography maken Silverlight GIS

Geografische Informatiesystemen met Microsoft tools

Wim Ligtendag

Geografische Informatiesystemen (GIS) bestaan al sinds de vroege jaren '60 en GIS ontwikkelaars zijn er dus al zo'n kleine halve eeuw. Maar tot voor kort behoorden deze ontwikkelaars tot een uithoek van de IT sector, waarin men zich vooral bezighield met zaken die voor het grootste deel van de IT collega's vaag of onbekend waren. Ook bediende men zich vaak van obscure macrotalen die nergens anders gebruikt werden.

In het afgelopen decennium is deze situatie sterk veranderd. GIS ontwikkelaars zijn in de main stream van de IT terecht gekomen en bedienen zich tegenwoordig van moderne ontwikkeltalen en –omgevingen, zoals bijvoorbeeld C# en Visual Studio. Daarbij is, net als in andere sectoren, het ontwikkelen van internetapplicaties steeds belangrijker geworden.

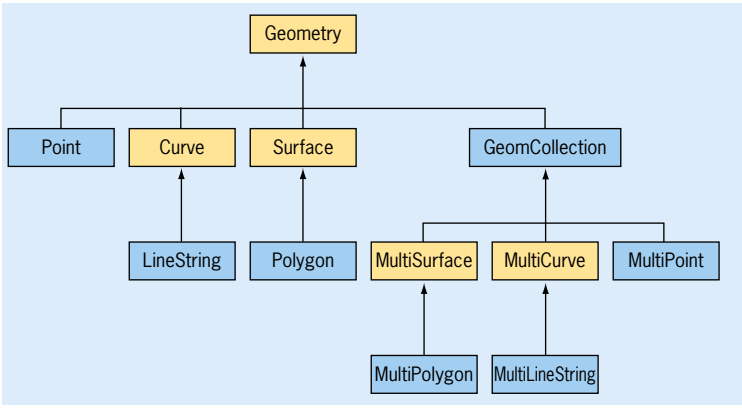
Daarnaast zijn er de laatste jaren steeds meer GIS standaarden gedefinieerd en GIS gerelateerde Frameworks en componenten beschikbaar gekomen. Ook Microsoft heeft aan deze laatste ontwikkeling bijgedragen, bijvoorbeeld door het opnemen van Spatial Data Types in SQL Server 2008 en door het beschikbaar stellen van Bing Maps. In dit artikel zal een applicatie worden beschreven waarin uitsluitend met gebruikmaking van Microsoft technologie- en ruimtelijke data wordt afgebeeld op een kaart in een internetapplicatie. Daarmee wordt geïllustreerd dat de beschikbare Microsoft tools en componenten het nu mogelijk maken om vrij geavanceerde GIS toepassingen te maken zonder dat de ontwikkelaar daarvoor zelf nog veel aanvullende componenten hoeft te maken. De werkende applicatie is te downloaden op mijn weblog: http://www.gisbelowsealevel.nl/blog.

SQL Server Spatial Types

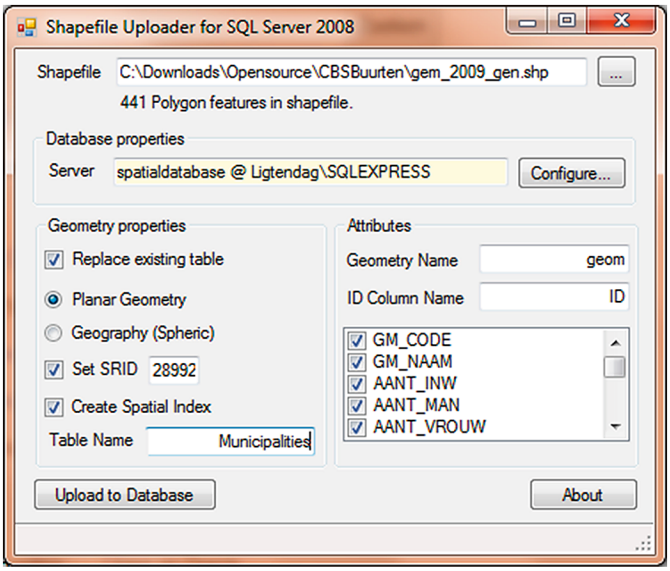
Sinds de release van SQL Server 2008 is het mogelijk om ruimtelijke entiteiten (punten, lijnen en vlakken en combinaties daarvan) rechtstreeks in een databasetabel op te slaan.
Er zijn met het oog daarop twee Spatial Types toegevoegd aan de collectie beschikbare datatypen: het geometry type en het geography type. Kort gezegd komt het erop neer dat in het geometry type data kan worden opgeslagen die geprojecteerd is op een plat vlak, terwijl het geography type data kan bevatten die ongeprojecteerd is en die dus op een bol of een ellips passen.
Laatstgenoemde data heeft coördinaten die in lengte- en breedtegraden wordt beschreven. Beide typen kunnen een zevental verschillende geometrieën bevatten (zie figuur 1). Deze geometrieën voldoen aan de Open Geospatial Consortium (OGC) Simple Features for SQL Specification, versie 1.1.0 (zie: http://www.opengeospatial.org/standards/sfs).

Ook de door de OGC gespecificeerde operaties die op deze typen kunnen moeten worden uitgevoerd (zoals bijvoorbeeld EQUALS(), INTERSECTS() en DISTANCE()), zijn door Microsoft nauwgezet geïmplementeerd.
Deze operaties kunnen via T-SQL worden uitgevoerd, maar ook vanuit .NET, want de geometry en geography typen en hun operaties zijn benaderbaar via een met SQL Server meegeleverde .NET assembly onder de naam Microsoft.SqlServer.Types.dll.

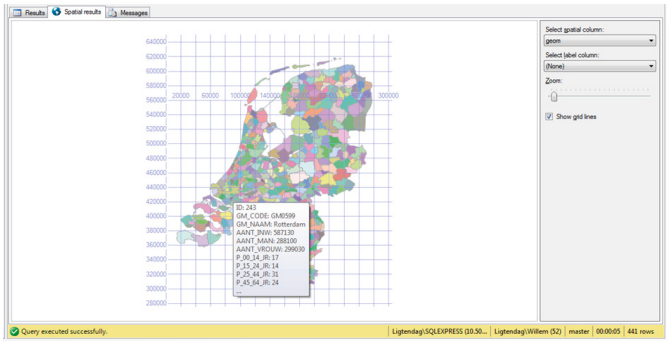
Als data provider voor de applicatie, die SqlServerAndBingInSilverlight is genoemd, is de gratis beschikbare versie SQL Server 2008 R2 Express gebruikt. Met SQL Server Management Studio is eerst een nieuwe database aangemaakt. Vervolgens is daarin een tabel met ruimtelijke data gecreëerd.
Vrij beschikbare ruimtelijke (vector)data is via het internet gemakkelijk te vinden. Ik heb hier gebruik gemaakt van de dataset zoals die te downloaden is bij het CBS (http://www.cbs.nl/nl-



FIGUUR 1 DE GEOMETRIETYPEN IN SQL SERVER 2008 (INSTANTIEABELE TYPEN IN BLAUW)



FIGUUR 2 DE SHAPE2SQL APPLICATIE.



FIGUUR 3 SPATIAL VIEWER IN SQL SERVER MANAGEMENT STUDIO

NL/menu/themas/dossiers/nederland-regionaal/publicaties/geografische-data/archief/2010/2010-wijk-en-buurtkaart-2009.htm). Deze dataset bevat een zogenaamde shapefile met gemeentegrenzen van Nederland.
Hoewel de shapefile het de facto uitwisselformaat is voor GIS data, heeft SQL Server Management Studio geen importmogelijkheid voor dit dataformaat. Maar dat is niet echt een probleem, want Morten Nielsen heeft zijn applicatie Shape2SQL beschikbaar gesteld om shapefiles te importeren in SQL Server 2008 (http://www.sharpgis.net/page/Shape2SQL.aspx; figuur 2).

Na importeren van de shapefile met de gemeenten, kan de tabel in SQL Server Management Studio bekeken worden (zie figuur 3).

De applicatie

Met de applicatie SqlServerAndBingInSilverlight kan nu vervolgens deze ruimtelijke data worden ontsloten en in een internet browser getoond. Voor het tonen van de data in de browser is Silverlight een erg handige optie, aangezien Silverlight ook Geometry typen heeft. Daardoor wordt het afbeelden van de SQL Server geometrieën in essentie gereduceerd tot het converteren van SQL Server geometrieën naar Silverlight geometrieën.
Ook het zelf maken van een MapControl, die onder meer functionaliteit als navigeren en in- en uitzoomen hoort te bevatten, wordt de ontwikkelaar door Microsoft uit handen genomen. Sinds ongeveer een jaar is de Bing Maps Silverlight Control SDK beschikbaar(http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=beb29d27-6f0c-494f-b028-1e0e3187e830; zie voor de interactive SDK: http://www.microsoft.com/maps/isdk/silverlight/). Deze SDK bevat een MapControl, die communiceert met de Bing Maps Web Services en op die manier kaarten van de hele wereld kan tonen. Net als bij de Bing Maps javascript API is het mogelijk om aan de standaard getoonde data eigen data toe te voegen en te tonen. Dat is precies wat nodig is om de Nederlandse gemeenten in de MapControl af te beelden. Om de MapControl te kunnen gebruiken, is het overigens wel nodig om een Bing Maps Developer Account aan te maken en vervolgens een Bing Maps Key aan te vragen. (http://msdn.microsoft.com/en-us/library/ff428642.aspx).
Om de Spatial Dataset uit SQL Server beschikbaar te krijgen in de Silverlight MapControl, moet uiteraard een Web Service worden gemaakt. De gemakkelijkste manier om dat te doen is via het WCF RIA Services framework, en wel door met behulp van Visual Studio 2010 een Silverlight Business Application te maken. Via een ADO.NET Entity Data Model en een Domain Service Class (for WCF RIA Services applications), waarin de betreffende Spatial Dataset wordt opgenomen, kan een WCF Web Service worden gegenereerd.
Vanuit de Silverlight applicatie kan deze Web Service vervolgens worden benaderd. Na het opvragen van de SQL Server geometry data dient deze nog te worden getransformeerd naar een voor de Silverlight MapControl begrijpelijk formaat en tenslotte aan de MapControl te worden toegevoegd.
Veel van het hierboven beschreven werk kon ik verwezenlijken via een aantal tools en Visual Studio 2010 Templates en Wizards. Maar daarbij kwam ik toch wel een aantal problemen tegen. Ik zal die hieronder kort beschrijven en oplossingen daarvoor aandragen.

> Voor het tonen van de data in de browser is Silverlight handig, omdat dit ook Geometry typen heeft.

Entity Framework en CLR UserDefined Types

Bij het aanmaken van het Entity Data Model kwam ik tot de ontdekking dat het ADO .NET Entity Framework geen CLR User Defined Types ondersteunt. Helaas behoren de SQL Server geometry en geography typen tot deze categorie. Toen ik een Entity Data Model maakte met daarin de Spatial Dataset kreeg ik de volgende waarschuwing:

"The data type 'geometry' is not supported; the column 'geom' in table 'SPATIALDATA.MDF.dbo.Municipalities' was excluded."

Gelukkig bestaat voor dit probleem een workaround. Door het maken van een Database View kan het datatype toch opgenomen worden in de vorm van een byte array:

CREATE VIEW MunicipalitiesView AS SELECT ID, [...], CAST(geom AS VARBINARY(MAX)) AS geom FROM [dbo].[Municipalities]

Het op basis van deze Database View gegenereerde Entity Data-Model kon daarna in een Domain Service class worden opgenomen. De Entity MunicipalitiesView bleek een partial class te zijn, zodat

ik de byte array Property geom daarvan ook eenvoudig in de vorm van een string Property kon exposen:

```
public partial class MunicipalitiesView
{
  [DataMember()]
  public string Geometry
  {
    get
    {
      using (MemoryStream memoryStream = new MemoryStream(geom))
      {
        using (BinaryReader binaryReader = new BinaryReader
        (memoryStream))
        {
          Microsoft.SqlServer.Types.SqlGeometry sqlGeometry = new
          Microsoft.SqlServer.Types.SqlGeometry();
          sqlGeometry.Read(binaryReader);
          return sqlGeometry.ToPolygonString();
        }
      }
    }
    set
    {
      // TODO Implement this if you want to store data back to the
      database
    }
  }
}
```

Voor het genereren van de string heb ik het standaard Microsoft SqlGeometry type uitgebreid door middel van een extension method die ik ToPolygonString() heb genoemd:

```
public static class SqlGeometryHelper
{
  public static string ToPolygonString(this SqlGeometry sqlGeometry)
  {
    StringBuilder stringBuilder = new StringBuilder();
    if (sqlGeometry != null && !sqlGeometry.IsNull)
    {
      sqlGeometry = sqlGeometry.MakeValid();
      for (int geometryIndex = 0; geometryIndex < (int)
      sqlGeometry.STNumGeometries(); geometryIndex++)
      {
        SqlGeometry subGeometry = sqlGeometry.STGeometryN
        (geometryIndex + 1);
        if (subGeometry.STGeometryType() == "Polygon" ||
        subGeometry.STGeometryType() == "MultiPolygon")
        {
          Point[] points = GetPointsFromSqlGeometry(subGeometry.
          STExteriorRing());
          Point[] projectedPoints = RDNewToWGS84(points);
          if (geometryIndex > 0)
          {
            stringBuilder.Append(";");
          }
          AddSegmentToGeometry(stringBuilder, projectedPoints);
        }
      }
    }
    return stringBuilder.ToString();
  }

  private static Point[] GetPointsFromSqlGeometry(SqlGeometry
  sqlGeometry)
  {
    Point[] points = new Point[(Int32)(sqlGeometry.STNum-
    Points())];
    for (int i = 0; i < sqlGeometry.STNumPoints(); i++)
    {
      SqlGeometry pointGeometry = sqlGeometry.STPointN(i + 1);
      points[i] = new Point(pointGeometry.STX.Value, point-
      Geometry.STY.Value);
    }
    return points;
  }

  private static void AddSegmentToGeometry(StringBuilder string-
  Builder, Point[] points)
  {
    for (int i = 0; i < points.Length; i++)
    {
      if (i > 0)
      {
        stringBuilder.Append(",");
      }
      stringBuilder.Append(String.Format("{0},{1}",
      points[i].X.ToString(CultureInfo.InvariantCulture), points[i
      ].Y.ToString(CultureInfo.InvariantCulture)));
    }
  }

  private static Point[] RDNewToWGS84(Point[] points)
  {
    Point[] transformedPoints = new Point[points.Length];
    for (int i = 0; i < transformedPoints.Length; i++)
    {
      transformedPoints[i] = RDNewToWGS84(points[i]);
    }
    return transformedPoints;
  }

  private static Point RDNewToWGS84(Point point)
  {
    double dX = (point.X - 155000) * Math.Pow(10, -5);
    double dY = (point.Y - 463000) * Math.Pow(10, -5);

    double latitudeBasePoint = (3235.65389 * dY) + (-32.58297 *
    Math.Pow(dX, 2)) +
      (-0.2475 * Math.Pow(dY, 2)) + (-0.84978 * Math.Pow(dX, 2) * dY) +
      (-0.0655 * Math.Pow(dY, 3)) + (-0.01709 * Math.Pow(dX, 2) *
      Math.Pow(dY, 2)) +
      (-0.00738 * dX) + (0.0053 * Math.Pow(dX, 4)) +
      (-0.00039 * Math.Pow(dX, 2) * Math.Pow(dY, 3)) +
      (0.00033 * Math.Pow(dX, 4) * dY) + (-0.00012 * dX * dY);

    double longitudeBasePoint = (5260.52916 * dX) + (105.94684 *
    dX * dY) +
      (2.45656 * dX * Math.Pow(dY, 2)) + (-0.81885 * Math.Pow
      (dX, 3)) +
      (0.05594 * dX * Math.Pow(dY, 3)) + (-0.05607 *
      Math.Pow(dX, 3) * dY) +
      (0.01199 * dY) + (-0.00256 * Math.Pow(dX, 3) *
      Math.Pow(dY, 2)) +
      (0.00128 * dX * Math.Pow(dY, 4)) + (0.00022 *
      Math.Pow(dY, 2)) +
      (-0.00022 * Math.Pow(dX, 2)) + (0.00026 * Math.Pow(dX, 5));

    double latitude = 52.15517440 + (latitudeBasePoint / 3600);
    double longitude = 5.38720621 + (longitudeBasePoint / 3600);

    return new Point(longitude, latitude);
  }
}
```

Deze extension method maakt gebruik van de methods van het SqlGeometry CLR type om de afzonderlijke coördinaten op te vragen en die in een reeks van x,y coördinaten te plaatsen. Daarbij vormt elke reeks een ring en is die gescheiden van een volgende reeks door een semicolon. Het spreekt vanzelf dat voor andere geometry typen dan vlakken (Polygons en MultiPolygons geheten) deze extension method moet worden uitgebreid. Dat heb ik hier achterwege gelaten.

Coördinaatstelsels

Het tweede probleem dat ik tegenkwam was dat de geometrieën van de Nederlandse gemeenten coördinaten hebben in het stelsel van de Rijksdriehoeksmeting. Daar kan Bing Maps niet mee omgaan. De MapControl verwacht coördinaten in decimale breedte- en lengtegraden. Er moet dus voor ieder punt een transformatie plaatsvinden van Rijksdriehoekstelsel coördinaten naar geografische coördinaten. Ik moest deze transformatie dus zelf eerst uit-

voeren alvorens de reeks coördinaten via de Web Service beschikbaar te stellen aan de Silverlight applicatie.
De method RDNewToWGS84() zorgt daarvoor. De in deze method uitgevoerde transformatie is niet tot op de centimeter nauwkeurig (zie daarvoor het artikel waaruit de omrekeningsformules ook zijn overgenomen: http://www.dekoepel.nl/pdf/Transformatieformules.pdf). Landmeters bijvoorbeeld kunnen er dus niets mee aanvangen, maar voor ons doel is de transformatie nauwkeurig genoeg.

Complexe Polygonen

Mijn derde probleem was gelegen in het afbeelden van complexe vlakken. Bing Maps kan standaard alleen maar simpele vlakken (Polygonen) bestaande uit één enkele ring afbeelden. Dat geldt ook voor de Silverlight MapControl. Ingewikkelder Polygonen, bestaande uit meerdere ringen met daarin eventueel ook nog uitsparingen (zogenaamde inner rings), kunnen niet worden afgebeeld.

Kaarten en luchtfoto's in Bing Maps lenen zich niet voor zeer nauwkeurige GIS toepassingen.

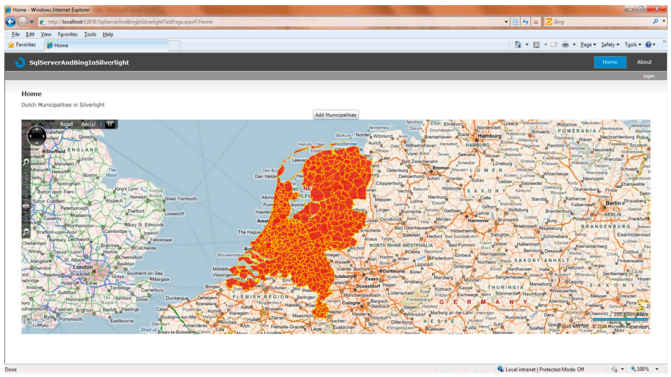
Ricky Brundritt, Microsoft MVP for Windows Live, heeft hier evenwel al een oplossing voor bedacht. Hij ontwikkelde onder meer de MapMultiPolygon class die dit probleem ondervangt (http://rbrundritt.spaces.live.com/blog/cns!E7DBA9A4-BFD458C5!1100.entry). In mijn Silverlight Client kon ik daarom met weinig code volstaan om de Gemeenten af te beelden in de Map Control:

```
private void OnClick(object sender, System.Windows.RoutedEventArgs e)
{
  MunicipalitiesDomainContext municipalitiesDomainContext = new
  MunicipalitiesDomainContext();

  EntityQuery<MunicipalitiesView> spatialQuery = from items in
  municipalitiesDomainContext.GetMunicipalitiesViewsQuery()
                        select items;

  municipalitiesDomainContext.Load(spatialQuery, LoadCompleted,
  null);
}

private void LoadCompleted(LoadOperation<MunicipalitiesView> args)
{
  foreach (MunicipalitiesView item in args.Entities)
  {
    if (IsMultiPolygon(item.Geometry))
    {
      MapMultiPolygon multiPolygon = CreateMultiPolygon(item.
      Geometry);
      map1.Children.Add(multiPolygon);
    }
    else
    {
      MapPolygon polygon = CreatePolygon(item.Geometry);
      map1.Children.Add(polygon);
    }
  }
}
….
```



FIGUUR 4 DE APPLICATIE IN WERKING

Conclusie

Een en ander leverde in vrij korte tijd een zeer acceptabele Silverlight Viewer voor Spatial Data op, gebouwd uitsluitend op basis van Microsoft technologie (figuur 4). Uiteraard is de beschreven applicatie erg rudimentair en mist er nog basale functionaliteit zoals bijvoorbeeld het zichtbaar en onzichtbaar maken van toegevoegde data en het tonen van een beschrijving van een ruimtelijke entiteit na een muisklik. Maar dat alles is met weinig moeite bij te bouwen. De tools zijn voorhanden.

Twee kanttekeningen moeten nog wel gemaakt.
In de eerste plaats moet gezegd dat de kaarten en luchtfoto's in Bing Maps zich niet lenen voor zeer nauwkeurige GIS toepassingen. Sub-meter nauwkeurigheid van de dataweergave kan niet worden gegarandeerd. Dat betekent dat dus bijvoorbeeld KLIC toepassingen (die informatie verschaffen bij o.m. graafwerkzaamheden over de precieze locatie van kabels en leidingen) niet op basis van Bing Maps kunnen worden gemaakt. Maar voor het overgrote deel van de vele soorten denkbare GIS applicaties is deze beperking niet relevant. Een dergelijke nauwkeurigheid is namelijk slechts zelden vereist.
In de tweede plaats leent de huidige MapControl zich niet voor het toevoegen van erg veel vectordata aan de kaart. Gebruik van de hier beschreven applicatie laat zien dat het afbeelden van de 441 gemeenten (met in totaal 34953 coördinaten) nog wel gaat, maar dat de performance bij het zoomen en het verschuiven van de kaart al ernstig achteruitgaat. Bij grotere hoeveelheden vectordata zal dus eerst server-side een bitmap moeten worden gegenereerd, die dan vervolgens op de MapControl kan worden afgebeeld. Ook dat is mogelijk, maar dat onderwerp valt buiten het bestek van dit artikel.

Wim Ligtendag, werkt als GIS Consultant bij De GISFabriek (http://www.gisfabriek.nl) . Hij is te bereiken via wim@gisfabriek.nl of wim@ligtendag.nl. Wim's blog is te vinden op http://www.gisbelowsealevel.nl/blog.