

پشتیبانی Spatial data از SQL Server

از SQL Server 2008 به بعد، نوع داده جدیدی به نام geography به نوع‌های قابل تعریف ستون‌ها اضافه شده‌است. در این نوع ستون‌ها می‌توان طول و عرض جغرافیایی یک نقطه را ذخیره کرد و سپس به کمک توابع توکاری از آن‌ها کوئری گرفت.

	Column Name	Data Type	Allow Nulls
🔑	Id	int	<input type="checkbox"/>
▶	Location	geography	<input checked="" type="checkbox"/>
	Name	geography	<input checked="" type="checkbox"/>
	Type	geometry	<input checked="" type="checkbox"/>
		hierarchyid	<input checked="" type="checkbox"/>
		image	<input type="checkbox"/>
		int	<input type="checkbox"/>
		money	<input type="checkbox"/>
		nchar(10)	<input type="checkbox"/>
		ntext	<input type="checkbox"/>

در اینجا نمونه‌ای از نحوه‌ی تعریف و همچنین مقدار دهی این نوع ستون‌ها را مشاهده می‌کنید:

```
CREATE TABLE [Geo](
[id] [int] IDENTITY(1,1) NOT NULL,
[Location] [geography] NULL
)

insert into Geo( Location , long, lat ) values
( geography::STGeomFromText ('POINT(-121.527200 45.712113)', 4326))
```

متد geography::STGeoFromText یک SQL CLR function است. این متد در مثال فوق، مختصات یک نقطه را دریافت کرده‌است. همچنین نیاز دارد بداند که این نقطه توسط چه نوع سیستم مختصاتی ارائه می‌شود. عدد 4326 در اینجا یک [SRID](#) یا Spatial Reference System Identifier استاندارد است. برای نمونه اطلاعات ارائه شده توسط Google و یا Bing توسط این استاندارد ارائه می‌شوند. در اینجا متدهای توکار دیگری مانند geography::STDistance برای یافتن فاصله مستقیم بین نقاط نیز ارائه شده‌اند. خروجی آن بر حسب متر است.

پشتیبانی از Spatial Data در Entity framework

پشتیبانی از نوع مخصوص geography، در EF 5 توسط نوع داده‌ای DbGeography ارائه شد. این نوع داده‌ای immutable است. به این معنا که پس از نمونه سازی، دیگر مقدار آن قابل تغییر نیست.

در اینجا برای نمونه مدلی را مشاهده می‌کنید که از نوع داده‌ای DbGeography استفاده می‌کند:

```
using System.Data.Entity.Spatial;

namespace EFGeoTests.Models
{
    public class GeoLocation
    {
        public int Id { get; set; }
        public DbGeography Location { get; set; }
        public string Name { get; set; }
        public string Type { get; set; }

        public override string ToString()
        {
            return string.Format("Name:{0}, Location:{1}", Name, Location);
        }
    }
}
```

به همراه یک Context، تا کلاس GeoLocation در معرض دید EF قرار گیرد:

```
using System;
using System.Data.Entity;
using EFGeoTests.Models;

namespace EFGeoTests.Config
{
    public class MyContext : DbContext
    {
        public DbSet<GeoLocation> GeoLocations { get; set; }

        public MyContext()
            : base("Connection1")
        {
            this.Database.Log = sql => Console.WriteLine(sql);
        }
    }
}
```

برای مقدار دهی خاصیت Location از نوع DbGeography می‌توان از متد ذیل استفاده کرد که بسیار شبیه به متد geography::STGeoFromText عمل می‌کند:

```
private static DbGeography createPoint(double longitude, double latitude, int coordinateSystemId = 4326)
{
    var text = string.Format(CultureInfo.InvariantCulture.NumberFormat, "POINT({0} {1})", longitude, latitude);
    return DbGeography.PointFromText(text, coordinateSystemId);
}
```

تهیه منبع داده‌ی جغرافیایی

برای تدارک یک مثال واقعی جغرافیایی، نیاز به اطلاعاتی دقیق داریم. این نوع اطلاعات عموماً توسط یک سری فایل مخصوص به نام [Shapefiles](#) که حاوی اطلاعات برداری جغرافیایی هستند ارائه می‌شوند. برای نمونه اطلاعات جغرافیایی به روز ایران را از آدرس ذیل می‌توانید دریافت کنید:

<http://download.geofabrik.de/asia/iran.html>

<http://download.geofabrik.de/asia/iran-latest.shp.zip>

پس از دریافت این فایل، به تعدادی فایل با پسوند های shp، shx و dbf خواهیم رسید. فایل‌های shp بیانگر فرمت اشکال ذخیره شده هستند. فایل‌های shx یک سری ایندکس بوده و فایل‌های dbf از نوع بانک اطلاعاتی dBase IV می‌باشند.

همچنین اگر فایل‌های prj را باز کنید، یک چنین اطلاعاتی در آن موجودند:

```
GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]]
```

نکته‌ی مهمی که در اینجا باید مدنظر داشت، استاندارد [GCS_WGS_1984](#) آن است. این استاندارد معادل است با استاندارد [EPSG](#) 4326. عدد 4326 آن جهت ثبت این اطلاعات در یک بانک اطلاعاتی SQL Server حائز اهمیت است (پارامتر coordinateSystemId در متد createPoint و ممکن است از هر فایلی به فایل دیگر متفاوت باشد).

خواندن فایل‌های shp در دات نت

پس از دریافت فایل‌های shp و بانک‌های اطلاعاتی مرتبط با اطلاعات جغرافیایی ایران، اکنون نوبت به پردازش این فایل‌های مخصوص با فرمت بانک اطلاعاتی فاکس پرو مانند، رسیده‌است. برای این منظور می‌توان از پروژه‌ی سورس باز ذیل استفاده کرد:

[C# Esri Shapefile Reader](#)

این پروژه در خواندن فایل‌های shp بدون نقص عمل می‌کند اما توانایی خواندن نام‌های فارسی وارد شده در این نوع بانک‌های اطلاعاتی را ندارد. برای رفع این مشکل، سورس آن را از Codeplex دریافت کنید. سپس فایل Shapefile.cs را گشوده و ابتدای خاصیت Current آن را به نحو ذیل تغییر دهید:

```
/// <summary>
/// Gets the current shape in the collection
/// </summary>
public Shape Current
{
    get
    {
        if (_disposed) throw new ObjectDisposedException("Shapefile");
        if (!_opened) throw new InvalidOperationException("Shapefile not open.");

        // get the metadata
        StringDictionary metadata = null;
        if (!RawMetadataOnly)
        {
            metadata = new StringDictionary();
            for (int i = 0; i < _dbReader.FieldCount; i++)
            {
                string value = _dbReader.GetValue(i).ToString();
                if (_dbReader.GetDataTypeName(i) == "DBTYPE_WVARCHAR")
                {
                    // برای نمایش متون فارسی نیاز است
                    value = Encoding.UTF8.GetString(Encoding.GetEncoding(720).GetBytes(value));
                }
                metadata.Add(_dbReader.GetName(i), value);
            }
        }
    }
}
```

در اینجا فقط سطر استفاده از Encoding خاصی با شماره 720 و تبدیل آن به UTF8 اضافه شده‌است. پس از آن بدون مشکل می‌توان برچسب‌های فارسی را از فایل‌های dBase IV این نوع بانک‌های اطلاعاتی استخراج کرد (اصلاح شده‌ی آن در فایل پیوست مطلب موجود است).

```
using System.Collections.Generic;
using System.Linq;
using Catfood.Shapefile;

namespace EFGeoTests
{
    public class MapPoint
    {
        public Dictionary<string, string> Metadata { set; get; }
        public double X { set; get; }
        public double Y { set; get; }
    }
}
```

```

public static class ShapeReader
{
    public static IList<MapPoint> ReadShapeFile(string path)
    {
        var results = new List<MapPoint>();

        using (var shapefile = new Shapefile(path))
        {
            foreach (var shape in shapefile)
            {
                if (shape.Type != ShapeType.Point)
                    continue;

                var shapePoint = shape as ShapePoint;
                if (shapePoint == null)
                    continue;

                var metadataNames = shape.GetMetadataNames();
                if(!metadataNames.Any())
                    continue;

                var metadata = new Dictionary<string, string>();
                foreach (var metadataName in metadataNames)
                {
                    metadata.Add(metadataName, shape.GetMetadata(metadataName));
                }

                results.Add(new MapPoint
                {
                    Metadata = metadata,
                    X = shapePoint.Point.X,
                    Y = shapePoint.Point.Y
                });
            }
        }

        return results;
    }
}

```

در کدهای فوق به کمک کتابخانه‌ی C# Esri Shapefile Reader، اطلاعات نقاط بانک اطلاعاتی shape files را خوانده و به صورت لیست‌هایی از MapPoint بازگشت می‌دهیم. نکته‌ی مهم آن، Metadata است که از هر فایلی به فایل دیگر می‌توان متفاوت باشد. به همین جهت این اطلاعات را به شکل ویژگی‌های key/value در این نوع بانک‌های اطلاعاتی ذخیره می‌کنند.

افزودن اطلاعات جغرافیایی به بانک اطلاعاتی SQL Server به کمک Entity framework

فایل places.shp را در مجموعه فایل‌هایی که در ابتدای بحث عنوان شدند، می‌توانید مشاهده کنید. قصد داریم اطلاعات نقاط آن را به مدل GeoLocation انتساب داده و سپس ذخیره کنیم:

```

var points = ShapeReader.ReadShapeFile("IranShapeFiles\\places.shp");
using (var context = new MyContext())
{
    context.Configuration.AutoDetectChangesEnabled = false;
    context.Configuration.ProxyCreationEnabled = false;
    context.Configuration.ValidateOnSaveEnabled = false;

    if (context.GeoLocations.Any())
        return;

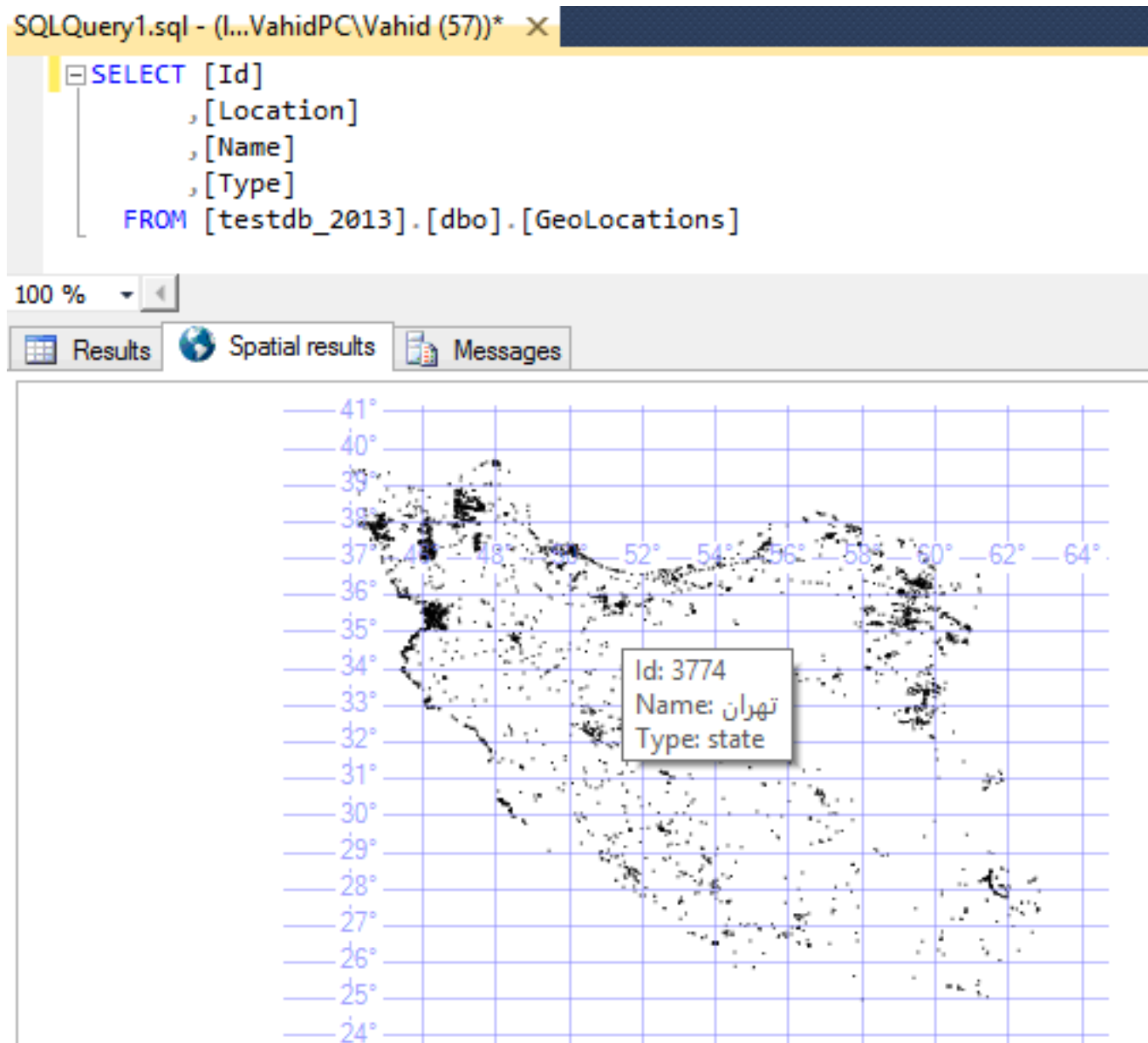
    foreach (var point in points)
    {
        context.GeoLocations.Add(new GeoLocation
        {
            Name = point.Metadata["name"],
            Type = point.Metadata["type"],
            Location = createPoint(point.X, point.Y)
        });
    }

    context.SaveChanges();
}

```

تعریف متد createPoint را که بر اساس X و Y نقاط، معادل قابل پذیرش آنرا جهت SQL Server تهیه می‌کند، در ابتدای بحث مشاهده کردید.

در فایل‌های مرتبط با places.shp، متادیتا name، معادل نام شهرهای ایران است و type آن بیانگر شهر، روستا و امثال آن می‌باشد. پس از اینکه اطلاعات مکان‌های ایران، در SQL Server ذخیره شدند، نمایش بصری آن‌ها را در management studio نیز می‌توان مشاهده کرد:



کوئری گرفتن از اطلاعات جغرافیایی

فرض کنید می‌خواهیم مکان‌هایی را با فاصله کمتر از 5 کیلومتر از تهران پیدا کنیم:

```
var tehran = createPoint(51.4179604, 35.6884243);
using (var context = new MyContext())
{
    // find any locations within 5 kilometers ordered by distance
}
```

```

var locations = context.GeoLocations
    .Where(loc => loc.Location.Distance(tehran) < 5000)
    .OrderBy(loc => loc.Location.Distance(tehran))
    .ToList();

foreach (var location in locations)
{
    Console.WriteLine(location.Name);
}

```

همانطور که پیشتر نیز عنوان شد، متد Distance بر اساس متر کار می‌کند. به همین جهت برای تعریف 5 کیلومتر به نحو فوق عمل شده‌است. همچنین نحوه‌ی مرتب سازی اطلاعات نیز بر اساس فاصله از یک مکان مشخص صورت گرفته‌است. و یا اگر بخواهیم دقیقاً بر اساس مختصات یک نقطه، مکانی را بیابیم، می‌توان از متد SpatialEquals استفاده کرد:

```

var tehran = createPoint(51.4179604, 35.6884243);
using (var context = new MyContext())
{
    // find any locations within 5 kilometers ordered by distance
    var tehranLocation = context.GeoLocations.FirstOrDefault(loc =>
loc.Location.SpatialEquals(tehran));
    if (tehranLocation != null)
    {
        Console.WriteLine(tehranLocation.Type);
    }
}

```

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

[EFGeoTests.zip](#)

نظرات خوانندگان

نویسنده: محمد باقر سیف الهی
تاریخ: ۱۰:۴۱ ۱۳۹۳/۰۳/۳۱

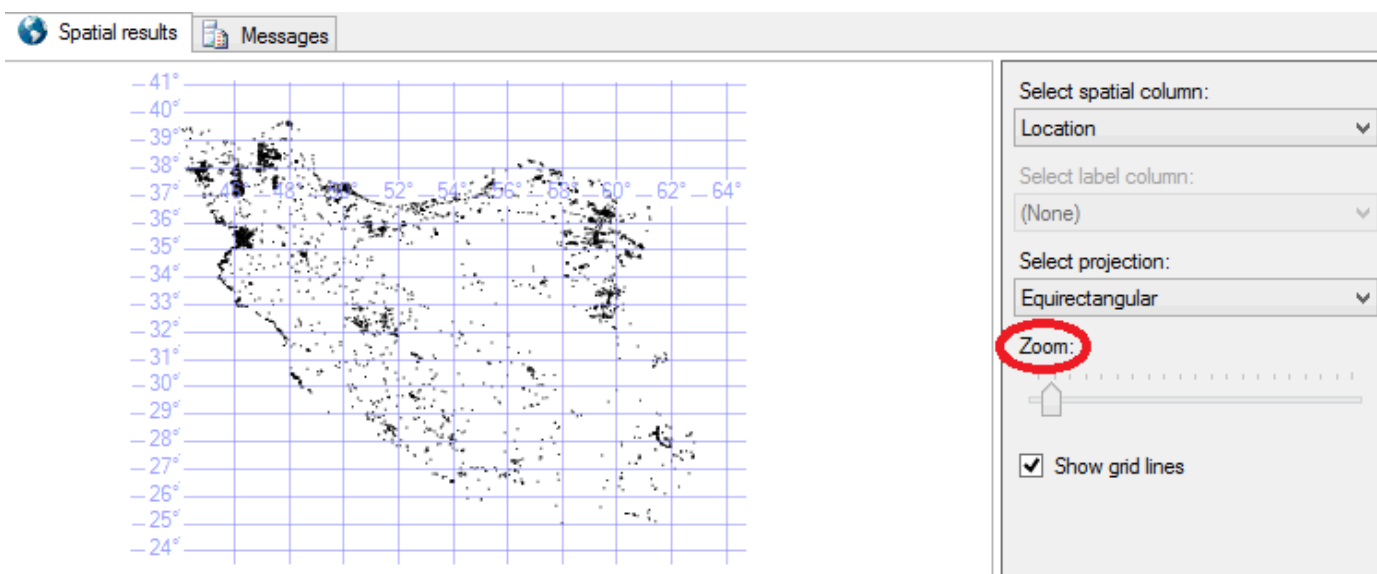
سلام

مشکلی که قبل‌تر در حین کار با فایل‌های shp برخوردیم، تغییر مقیاس نقشه و نگاشت مختصات خاص و توزیع شده ای بود که باید روی نقشه نمایش داده می‌شد. (مثلا نمایش مراکز استان‌ها روی نقشه و تغییر scale نقشه و به تبع اون، تغییر مکان مختصات) برای این کار چه راهکاری هست؟
* دسترسی به فایل‌های dbf و prj و sbn و sbx و shx وجود دارد.

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۱:۱۰ ۱۳۹۳/۰۳/۳۱

این‌ها بیشتر مسایل نمایشی است و توانمندی ابزار نمایش دهنده‌ی اطلاعات نقشه. نیازی نیست در اصل دیتابیس و اطلاعات، تغییری حاصل شود؛ چون اندازه‌ی نمایشی حتی اگر 10 برابر هم شود، در فاصله‌ی بین تهران و شیراز نهایتاً تغییری حاصل نخواهد شد و طول و عرض جغرافیایی مکان‌ها ثابت خواهند ماند. برای نمونه اگر مثال پیوست شده را اجرا کنید، خود management studio امکان تغییر اندازه‌ی نمایشی را دارد:



در برنامه‌های دات نت هم برای مثال از [SharpMap](#) می‌شود برای نمایش این نوع اطلاعات به همراه [تغییر اندازه و ابعاد](#) خودکار نقشه استفاده کرد. برای برنامه‌های وب هم [jVectorMap](#) چنین قابلیت‌هایی را دارد.

نویسنده: بهراد ایزدی
تاریخ: ۱۵:۰۰ ۱۳۹۳/۰۴/۰۲

سلام

ممنون برای این آموزش خوب

بعضی مناطق فایل shp رو ندارند و فقط دو نوع OSM براشون موجود هست راهی برای خواندن اون‌ها وجود داره ؟

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۵:۱۲ ۱۳۹۳/۰۴/۰۲

- برنامه‌ی سورس باز [ArcGIS Editor for OpenStreetMap](#) یک چنین قابلیت‌ی را دارد.
- همچنین یک سری 4 قسمتی را [در اینجا](#) می‌توانید در مورد تبدیل open street maps به داده‌های SQL Server مطالعه کنید.

نویسنده: شاهین کیاست
تاریخ: ۱۷:۰۶ ۱۳۹۳/۰۷/۱۳

- آیا ممکن هست به جای نوع داده‌ی DbGeography از نوع داده‌ی SQLGeometry استفاده کرد؟
- ویرایش
- تنها کافی است نوع Property مورد نظر را DbGeometry تعیین کرد.