عنوان: **مروری بر طراحی Schema less بانک اطلاعاتی SisoDb**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۵ ۱۵:۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, NoSQL, SisoDb

اس کیوال سرور، از سال 2005 به بعد، به صورت توکار امکان تعریف و ذخیره سازی اطلاعات schema less و یا schema free را به کمک فیلدهایی از نوع XML ارائه داده است؛ به همراه یکپارچگی آن با زبان XQuery برای تهیه کوئری‌های سریع سمت سرور. در فیلدهای XML می‌توان اطلاعات انواع و اقسام اشیاء را بدون اینکه نیازی به تعریف تک تک فیلدهای مورد نیاز، در بانک اطلاعاتی وجود داشته باشد، ذخیره کرد. یک نمونه از کاربرد چنین امکانی، نوشتن برنامه‌های «فرم ساز» است. برنامه‌هایی که کاربران آن می‌توانند فیلد اضافه و کم کرده و نهایتا اطلاعات را ذخیره و از آن‌ها کوئری بگیرند.
خوب، این فیلد کمتر بحث شده XML، فقط در اس کیوال سرور و نگارش‌های اخیر آن وجود دارد. اگر نیاز به کار با بانک‌های اطلاعاتی سبک‌تری وجود داشت چطور؟ یک راه حل عمومی برای این مساله مراجعه به روش‌های NoSQL است. یعنی بطور کلی بانک‌های اطلاعاتی رابطه‌ای کنار گذاشته شده و به یک سکوی کاری دیگر سوئیچ کرد. در این بین، SisoDb راه حل میانه‌ای را ارائه داده است. با کمک SisoDb می‌توان اطلاعات را به صورت schema less و بدون نیاز به تعریف فیلدهای متناظر آن‌ها، در انواع و اقسام بانک‌های اطلاعاتی SQL Server با فرمت JSON ذخیره و بازیابی کرد. این انواع و اقسام، شامل SQL Server CE نیز می‌شود.

دریافت و نصب SisoDb

دریافت و نصب SisoDb بسیار ساده است. به کمک package manager و امکانات NuGet، کلمه Sisodb را جستجو کنید. در بین مداخل ظاهر شده، پروایدر مورد علاقه خود را انتخاب و نصب نمائید. برای مثال اگر قصد دارید با SQL Server CE کار کنید، SisoDb.Sql1Ce4 را انتخاب و یا اگر SQL Server 2008 مدنظر شما است، SisoDb.Sql12008 را انتخاب و نصب نمائید.

ثبت و بازیابی اطلاعات به کمک SisoDb

کار با SisoDb بسیار روان است. نیازی به تعاریف نگاشت‌ها و ORM خاصی نیست. یک مثال مقدماتی آن‌را در ادامه ملاحظه می‌کنید:

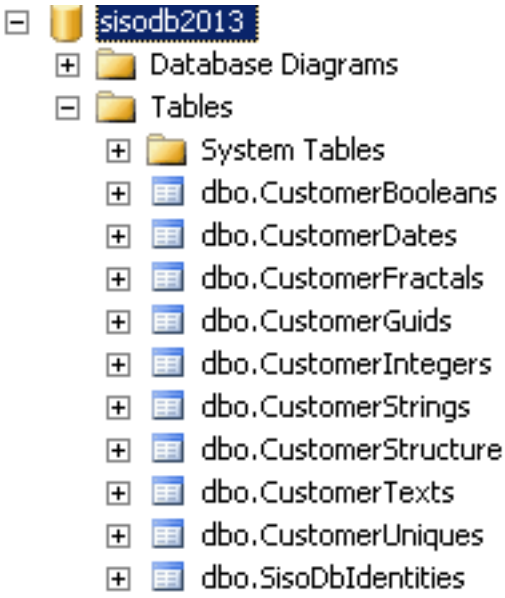
```
using SisoDb.Sql2008;

namespace SisoDbTests
{
    public class Customer
    {
        public int Id { get; set; }
        public int CustomerNo { get; set; }
        public string Name { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            /*var cnInfo = new Sql1Ce4ConnectionInfo(@"Data source=sisodb2013.sdf;");
            var db = new Sql1Ce4DbFactory().CreateDatabase(cnInfo);
            db.EnsureNewDatabase();*/

            var cnInfo = new Sql2008ConnectionInfo(@"Data Source=(local);Initial
Catalog=sisodb2013;Integrated Security = true");
            var db = new Sql2008DbFactory().CreateDatabase(cnInfo);
            db.EnsureNewDatabase();

            var customer = new Customer
            {
                CustomerNo = 20,
                Name = "Vahid"
            };
            db.UseOnceTo().Insert(customer);

            using (var session = db.BeginSession())
            {
                var info = session.Query<Customer>().Where(c => c.CustomerNo == 20).FirstOrDefault();

                var info2 = session.Query<Customer>().Where(c => c.CustomerNo == 20 &&
c.Name=="Vahid").FirstOrDefault();
```

```
            }
        }
    }
}
```

در این مثال، ابتدا اتصال به بانک اطلاعاتی برقرار شده و سپس بانک اطلاعاتی جدید تهیه می‌شود. سپس یک وهله از شیء مشتری ایجاد و ذخیره می‌گردد. در ادامه دو کوئری بر روی بانک اطلاعاتی انجام شده است.

ساختار داخلی SisoDb

SisoDb به ازای هر کلاس، حداقل 9 جدول را ایجاد می‌کند. در ادامه نحوه ذخیره سازی شیء مشتری ایجاد شده و مقادیر خواص آن‌را نیز مشاهده می‌نمائید:

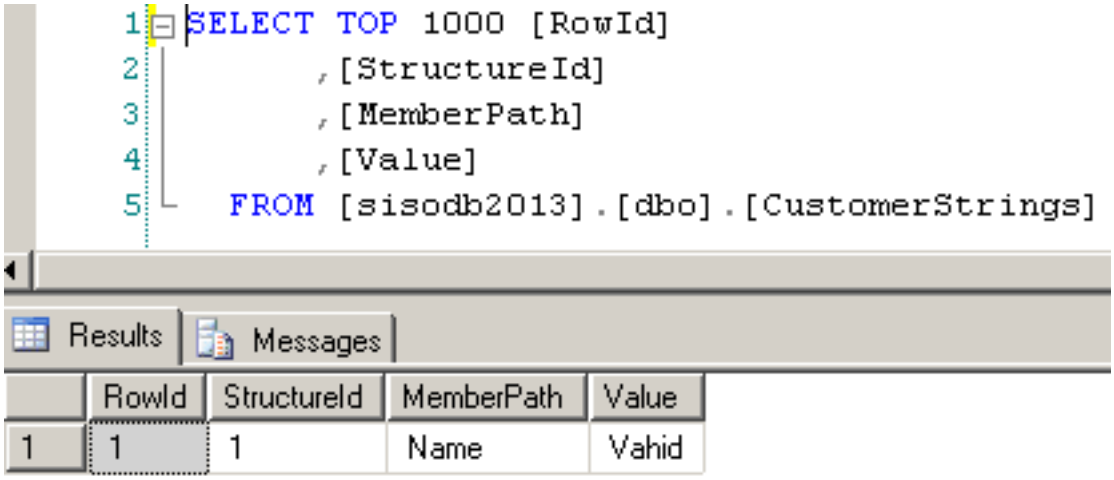



```
SELECT TOP 1000 [RowId]
      ,[StructureId]
      ,[MemberPath]
      ,[Value]
      ,[StringValue]
  FROM [sisodb2013].[dbo].[CustomerIntegers]
```

Results | Messages

	RowId	StructureId	MemberPath	Value	StringValue
1	1	1	Id	1	1
2	2	1	CustomerNo	20	20

ذخیره سازی جداگانه خواص عددی

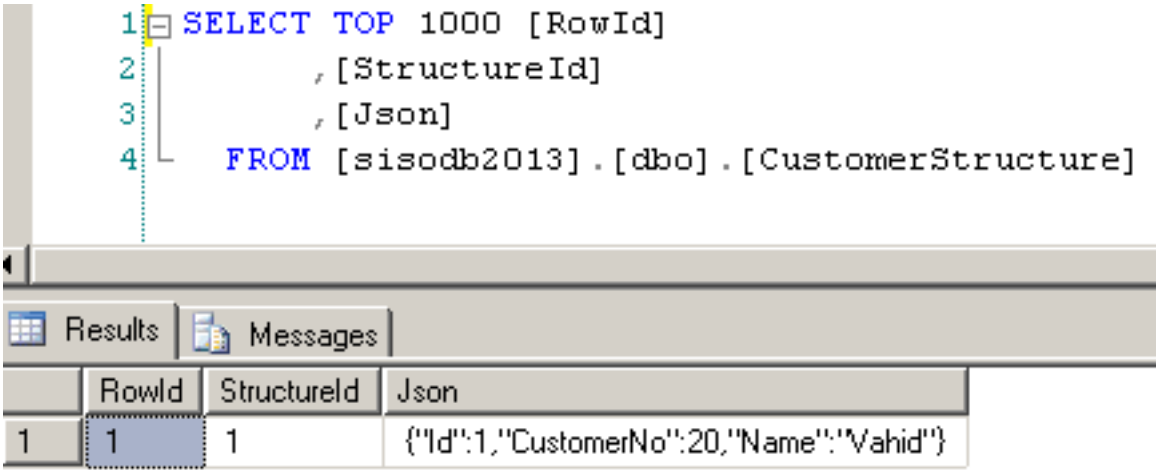


```
SELECT TOP 1000 [RowId]
      ,[StructureId]
      ,[MemberPath]
      ,[Value]
  FROM [sisodb2013].[dbo].[CustomerStrings]
```

Results | Messages

	RowId	StructureId	MemberPath	Value
1	1	1	Name	Vahid

ذخیره سازی جداگانه خواص رشته‌ای

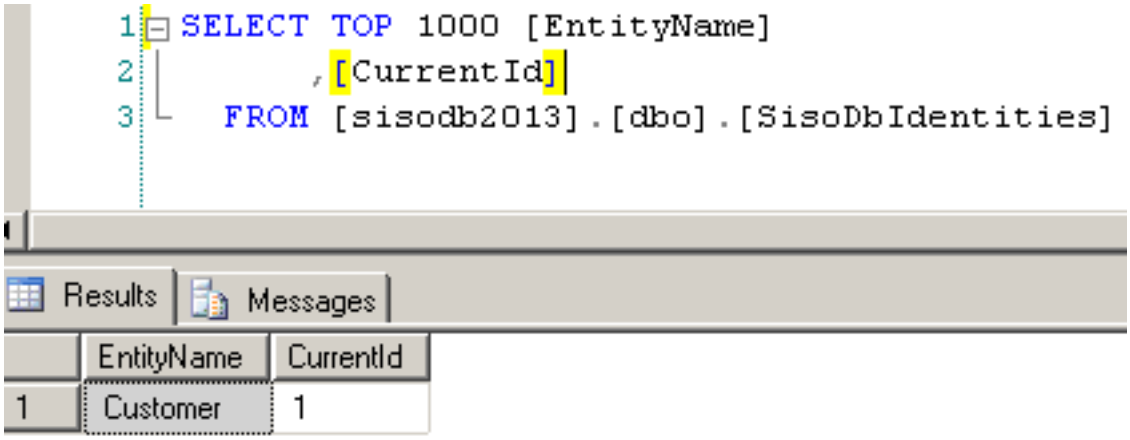


```
SELECT TOP 1000 [RowId]
      ,[StructureId]
      ,[Json]
  FROM [sisodb2013].[dbo].[CustomerStructure]
```

Results | Messages

	RowId	StructureId	Json
1	1	1	{"Id":1,"CustomerNo":20,"Name":"Vahid"}

ذخیره سازی کلی شیء مشتری با فرمت JSON به صورت یک رشته



همانطور که ملاحظه می‌کنید، یک جدول کلی SisoDbIdentities ایجاد شده است که اطلاعات نام اشیاء را در خود نگهداری می‌کند. سپس اطلاعات خواص اشیاء یکبار به صورت JSON ذخیره می‌شوند؛ با تمام اطلاعات تو در توی ذخیره شده در آن‌ها و همچنین یکبار هم هر خاصیت را به صورت یک رکورد جداگانه، بر اساس نوع کلی آن‌ها، در جداول رشته‌ای، عددی و امثال آن ذخیره می‌کند.

شاید بپرسید که چرا به همان فیلد رشته‌ای JSON اکتفاء نشده است؟ از این جهت که پردازشگر سمت بانک اطلاعاتی آن همانند فیلدهای XML در SQL Server و نگارش‌های مختلف آن وجود ندارد (برای مثال به کمک زبان T-SQL می‌توان از زبان XQuery در خود بانک اطلاعاتی، بدون نیاز به واکشی کل اطلاعات در سمت کلاینت، به صورت یکپارچه استفاده کرد). به همین جهت برای کوئری گرفتن و یا تهیه ایندکس، نیاز است این موارد جداگانه ذخیره شوند.

به این ترتیب زمانیکه کوئری تهیه می‌شود، برای مثال:

```
var info = session.Query<Customer>().Where(c => c.CustomerNo == 20).FirstOrDefault();
```

به کوئری زیر ترجمه می‌گردد:

```
SELECT DISTINCT TOP(1) (s.[StructureId]),
                       s.[Json]
                FROM   [CustomerStructure] s
                       LEFT JOIN [CustomerIntegers] mem0
                            ON  mem0.[StructureId] = s.[StructureId]
                            AND mem0.[MemberPath] = 'CustomerNo'
                WHERE  (mem0.[Value] = 20);
```

و یا کوئری ذیل:

```
var info2 = session.Query<Customer>().Where(c => c.CustomerNo == 20 &&
c.Name=="Vahid").FirstOrDefault();
```

معادل زیر را خواهد داشت:

```
SELECT DISTINCT TOP(1) (s.[StructureId]),
                       s.[Json]
                FROM   [CustomerStructure] s
                       LEFT JOIN [CustomerIntegers] mem0
                            ON  mem0.[StructureId] = s.[StructureId]
                            AND mem0.[MemberPath] = 'CustomerNo'
                       LEFT JOIN [CustomerStrings] mem1
                            ON  mem1.[StructureId] = s.[StructureId]
                            AND mem1.[MemberPath] = 'Name'
                WHERE  ((mem0.[Value] = 20) AND (mem1.[Value] = 'Vahid'));
```

در هر دو حالت از جداول کمکی تعریف شده برای تهیه کوئری استفاده کرده و نهایتا فیلد JSON اصلی را برای نگاشت نهایی به اشیاء تعریف شده در برنامه بازگشت می‌دهد.

در کل این هم یک *روش تفکر* و طراحی Schema less است که با بسیاری از بانک‌های اطلاعاتی موجود سازگاری دارد.
برای مشاهده اطلاعات بیشتری در مورد جزئیات این روش می‌توان به Wiki آن مراجعه کرد.