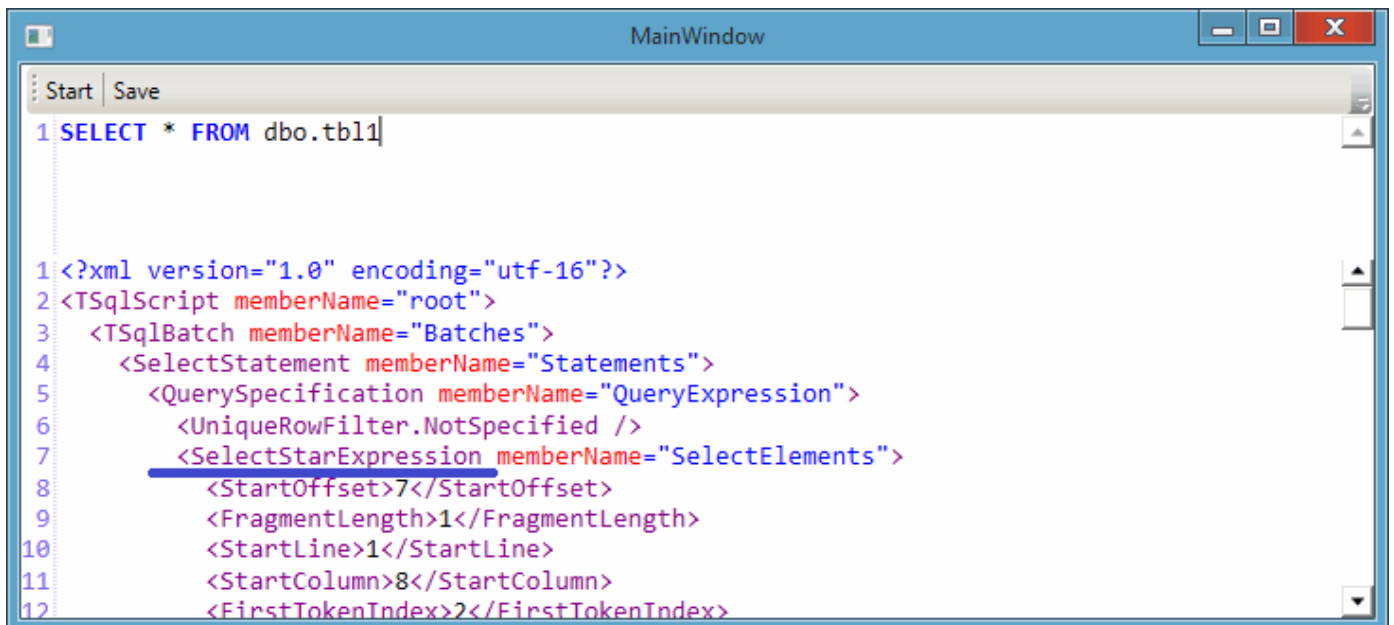


مدتی قبل مطلبی را در مورد کتابخانه‌ی ویژه SQL Server که یک T-SQL Parser تمام عیار است، [در این سایت مطالعه کردید](#). در این قسمت، همان مطلب را به نحو بهتر و ساده‌تری بازنویسی خواهیم کرد. مشکلی که در دراز مدت با SQLDom وجود خواهد داشت، مواردی مانند SelectStarExpression و CreateProcedureStatement و امثال آن هستند. این‌ها را از کجا باید تشخیص داد؟ همچنین مراحل بررسی این اجزاء، نسبتاً طولانی هستند و نیاز به یک راه حل عمومی‌تر در این زمینه وجود دارد.

راه حلی برای این مشکل در مطلب «[XML 'Visualizer' for the TransactSql.ScriptDom parse tree](#)» ارائه شده‌است. در اینجا تمام اجزای TSqlFragment توسط Reflection مورد بررسی و استخراج قرار گرفته و نهایتاً یک فایل XML از آن حاصل می‌شود. اگر نکات ذکر شده در این مقاله را تبدیل به یک برنامه با استفاده مجدد کنیم، به چنین شکلی خواهیم رسید:



این برنامه را از اینجا می‌توانید دریافت کنید:

[DomToXml.zip](#)

همانطور که در تصویر مشاهده می‌کنید، اینبار به سادگی، SelectStarExpression قابل تشخیص است و تنها کافی است در T-SQL پردازش شده، به دنبال SelectStarExpression‌ها بود. برای اینکار جهت ساده شدن آنالیز می‌توان با ارث بری از کلاس پایه TSqlFragmentVisitor شروع کرد:

```
using System;
using System.Linq;
using Microsoft.SqlServer.TransactSql.ScriptDom;

namespace DbCop
{
    public class SelectStarExpressionVisitor : TSqlFragmentVisitor
    {
        public override void ExplicitVisit(SelectStarExpression node)
```

```

    {
        Console.WriteLine(
            "`Select *\` detected @StartOffset:{0}, Line:{1}, T-SQL: {2}",
            node.StartOffset,
            node.StartLine,
            string.Join(string.Empty, node.ScriptTokenStream.Select(x => x.Text)).Trim());

        base.ExplicitVisit(node);
    }
}

```

در کلاس پایه TSqlFragmentVisitor به ازای تمام اشیاء شناخته شده‌ی ScriptDom، یک متد ExplicitVisit قابل بازنویسی در نظر گرفته شده‌است. در اینجا برای مثال نمونه‌ی SelectStarExpression آن را بازنویسی کرده‌ایم. مرحله‌ی بعد، اجرای این کلاس Visitor است:

```

public static class GenericVisitor
{
    public static void Start(string tSql, TSqlFragmentVisitor visitor)
    {
        IList<ParseError> errors;
        TSqlScript sqlFragment;
        using (var reader = new StringReader(tSql))
        {
            var parser = new TSq1120Parser(initialQuotedIdentifiers: true);
            sqlFragment = (TSqlScript)parser.Parse(reader, out errors);
        }

        if (errors != null && errors.Any())
        {
            var sb = new StringBuilder();
            foreach (var error in errors)
                sb.AppendLine(error.Message);

            throw new InvalidOperationException(sb.ToString());
        }
        sqlFragment.Accept(visitor);
    }
}

```

در اینجا متد Accept کلاس TSq1120Parser، امکان پذیرش یک Visitor را دارد. به این معنا که Parser در حال کار، هر زمانیکه در حال آنالیز قسمتی از T-SQL دریافتی بود، نتیجه را به اطلاع یکی از متدهای کلاس پایه TSqlFragmentVisitor نیز خواهد رساند. بنابراین دیگر نیازی به نوشتن حلقه و بررسی تک تک اجزای خروجی TSq1120Parser نیست. اگر نیاز به بررسی SelectStarExpression داریم، فقط کافی است Visitor آن را طراحی کنیم.

مثالی از نحوه‌ی استفاده از کلاس GenericVisitor فوق را در اینجا ملاحظه می‌کنید:

```

var tsql = @"WITH ctex AS (
SELECT * FROM sys.objects
)
SELECT * FROM ctex";
GenericVisitor.Start(tsql, new SelectStarExpressionVisitor());

```