

SQL CLR

Severýn Tomáš

Obsah

- Seznámení
- CLR vs T-SQL
- CLR vs extended stored procedures
- Nasazení
- Chuťovky

Některé použité zkratky

- UDF – user defined function
- UDT - User Define Type
- TVF – Table valued function
 - Bude vysvětleno později

Seznámení

- Integrace .net jazyků do MS SQL serveru
- Od verze 2005
- Nenahrazuje T-SQL ani XPs
 - Extended Stored procedures

CLR vs T-SQL

- 1) **Množinové operace jsou stále rychlejší v T-SQL**

MS tweakuje T-SQL a vyhledávací engine již po dlouhou dobu CLR spíše tíhne k objektovému přístupu manipulace s daty (pokud se nepoužije T-SQL jako podvrstva)

- 2) **Procedurální a rekurzivní operace rychlejší v CLR**

Maximální hloubka rek. zanoření u T-SQL je 32 (nastavitelných max 100 u 2005 serveru)

- 3) **Předchozí pravidla nelze brát jako dogma**

CLR vs T-SQL problém

- V T-SQL se píše SQL přímo do textu procedury
- V CLR je potřeba volat “klasicky” pomocí ADO.NET
- Tímto se SQL nedá zjistit v compile time

CLR vs T-SQL

- Posílání dat na klienta
 - Zatímco T-SQL SELECT rovnou posílá data klientovi
 - V CLR je třeba využít objekt SqlPipe

SqlPipe

```
CREATE PROC proc1 AS SELECT col1 FROM dbo.table1;
```

```
[Microsoft.SqlServer.Server.SqlProcedure]
```

```
public static void proc1 () {  
    using (SqlConnection conn = new  
        SqlConnection("context connection = true")) {  
        conn.Open();  
        SqlCommand cmd = new SqlCommand(  
            "SELECT col1 FROM dbo.table1", conn);  
        SqlContext.Pipe.ExecuteAndSend(cmd);  
        conn.Close();  
    }  
}
```

SqlPipe

- Takhle je to v případě, že posíláme data přímo na klienta
- Pokud je chceme zpracovat

```
SqlDataRecord r;// jeden záznam
sqlpipe.SendResultsStart(record);
while(1) {
    r.SetString("a"); r.SetDateTime(Today());
    sqlpipe.SendResultsRow(record);
}
sqlpipe.SendResultsEnd();
```

Table Valued Function

- Zjednodušeně
- Objekt, který implementuje iterátor
- Každý řádek výsledné tabulky se dostane jako klasický MoveNext()

Příklad – rozklad na prvočinitele

```
public partial class UserDefinedFunctions {
    [Microsoft.SqlServer.Server.SqlFunction(
        FillRowMethodName="RowFiller", TableDefinition="Divisor int")]
    public static IEnumerable Divisor(Int32 input) {
        if (input < 1) {
            throw new ArgumentOutOfRangeException("input",
                „Please pass in a value greater than 0“);
        }
        for (int i = 1; i < input; i++)
        { // Is i a perfect divisor of input?
            if(input%i == 0) { yield return i ; }
        }
    }
    public static void RowFiller(object row, out int Divisor) {
        Divisor = (int)row ;
    }
}
```

CLR vs XPs

- Již od MSSQL 2000 – Extended Stored procedures
- Možnost napsat funkci v C++
- Jaká je tedy výhoda CLR?

CLR vs XPs

Zabezpečení

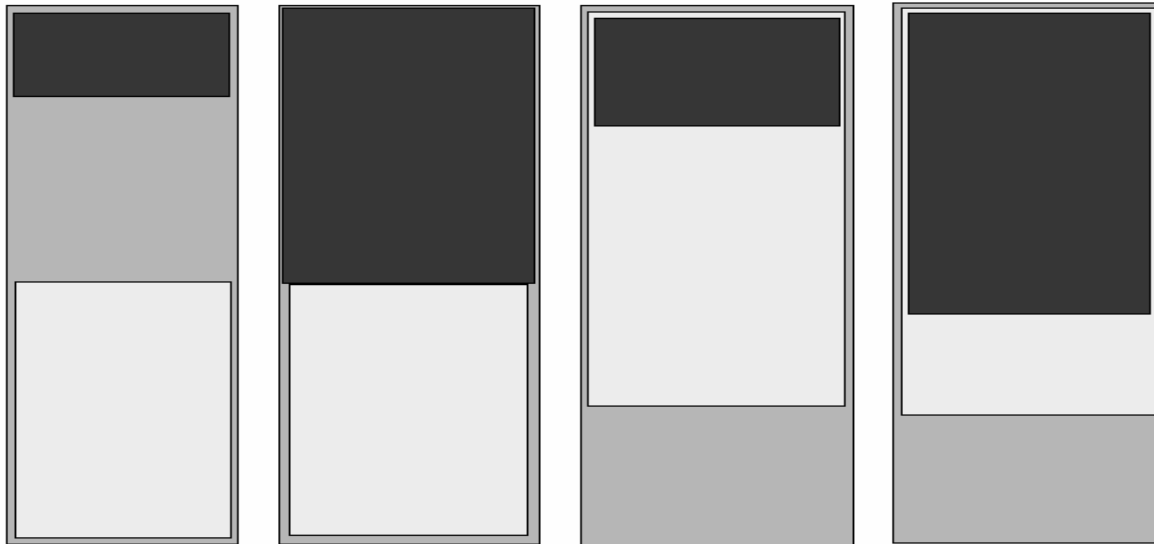
- CLR objekty se kategorizují do 3 bezpečnostních kategorií
 - SAFE – default level, nejrestriktivnější, pouze lokální zdroje
 - EXTERNAL_ACCESS – povolené některé vnější zdroje (např: souborový systém, síť, registr, proměnné prostředí...)
 - UNSAFE – nedoporučeno, může vše, stejně jako XPs

CLR vs XPs

- CLR – jakýkoliv .NET jazyk
- Managed kód
- Lepší práce s novými datovými typy varchar(max), varbinary(max) a i XML.
- CLR objekty mohou být použity jako trigger, UDF, TVF i agregační funkce
- Ale unmanaged kód bývá rychlejší...
- I tak jsou už nedoporučovány XPs

Proč jsou nedoporučovány?

- SQL server má dohled nad pamětí CLR
- Zatímco nad XPs ne



Atributy které se používají

- `SqlProcedure`
- `SqlFunction`
- `SqlTrigger`
- `SqlUserDefinedAggregate`

Transakce

- V .NET 2.0 System.Transactions
- System.Transactions.Transaction.Current.Rollback();
- Rollbackne stávající transakci
- RM – resource manager
- DTC – Distributed transaction coordinator
- Dvofázový commit
- Pokud se chci transakci vyhnout musím do connection stringu připsat „enlist=false“, by default jsou všechny operace zařazeny do nadřazené transakce

Vypočtené sloupce a indexy

- V tomto je trochu omezení
- Nové parametry atributů - BOOL
 - **IsDeterministic**
 - **IsPrecise**
 - **IsSystemVerified**
 - **SystemDataAccess**
 - **UserDataAccess**

Vypočtené sloupce a indexy

- U tsql tyto parametry ohodnotí systém
- U jazyka v CLR musí toto určit programátor pomocí parametrů
- Jenže i tak server nevěří
 - Jde oindexovat pouze zperzistentnělé sloupce
- for the sake of user's safety, Sql Server in this release REQUIRES the user to persist the computed columns (unlike the tsql case) to actually index the computed columns.

Co se mu nelíbí

- Pokud nebude funkce deterministická
 - Na stejný vstup vrátí různé výstupy
 - Dojde ke korupci indexu
- Proč nám to nevadí u perzistentních?
 - Funkce se propočítá pouze jednou, ne vždy při přístupu na index
 - Teprve uložená hodnota se oindexuje

Jak je to tedy skutečně s CLR v SQL serveru

- Rozdíl oproti „normálnímu“?
 - V tom jak nakládá s thready, pamětí a synchronizací
- Volá se přímo **Serverová API**
 - Jak pro uživatelský, tak vlastní kód
 - Pro vytvoření threadu a synchronizaci se volají přímo serverová primitiva

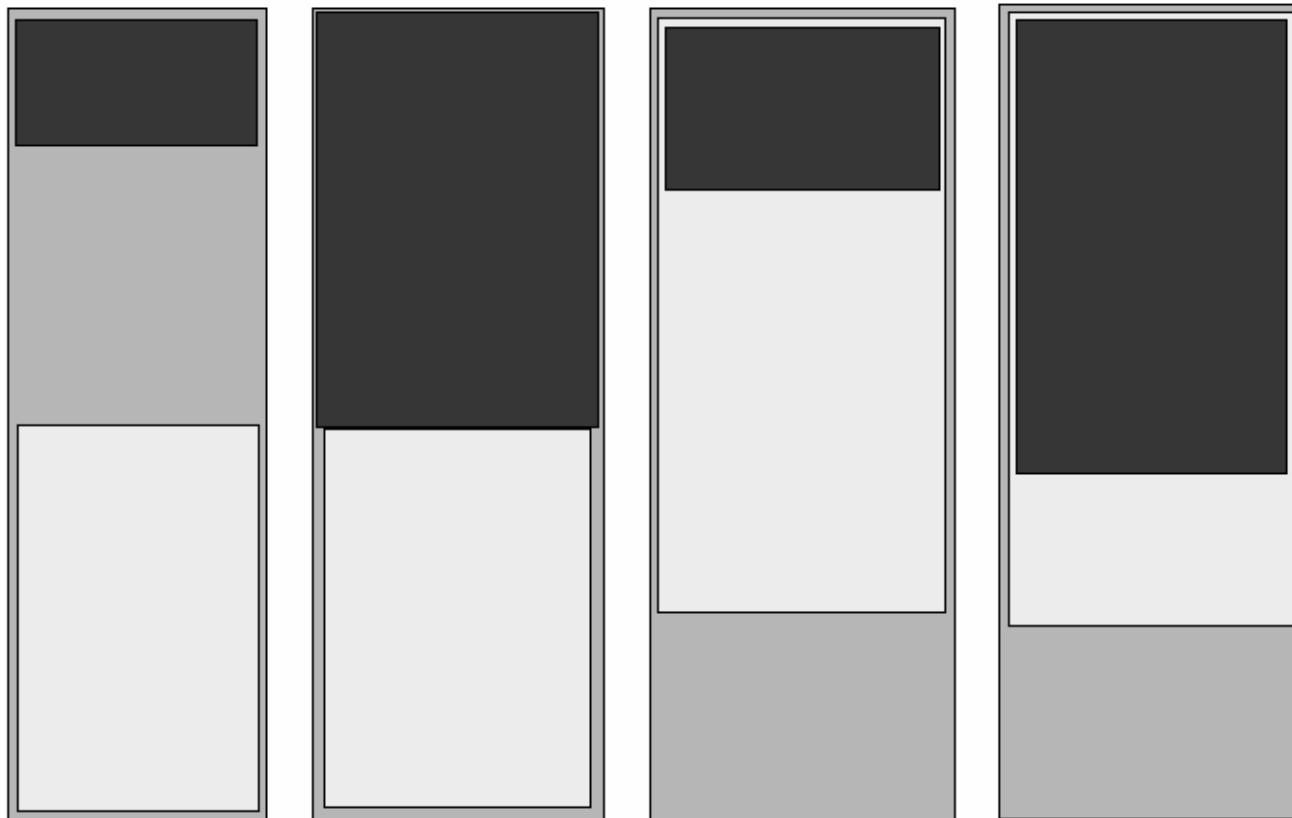
CLR uvnitř serveru

- Používání serverových mechanismů dovoluje
 - Plánovači přeplánovat proces čekající na synchronizaci
 - Například při inicializaci GC uvnitř CLR všechna přidružená vlákna čekají na GC
 - Server tedy může naplánovat jiný DB proces neběžící v CLR
 - Pokud je dlouhotrvající thread, může mu explicitně zavolat GC
 - Také server může odhalit deadlock

Alokace paměti

- Alokace probíhá pomocí serverových funkcí
 - Tím pádem se tato paměť započítává do celkově alokované paměti serverem
 - Tím je zajištěno že server a CLR spolu o paměť „nebojují“
 - Také může přímo odepřít CLR alokaci další paměti pokud sám potřebuje

Zopakování



Výkon integrovaného CLR

- Proces kompilace
 - V průběhu kompilace SQL pokud server narazí na managed proceduru, nebo funkci vytvoří blok(stub) MSIL kódu
 - Pro namarshalování parametrů z SQL do CLR
 - Zavolání procedury
 - Vrácení výsledku
 - „glue“ code

Glue code

- Snaží se o type-specific optimalizace SQL serveru
 - Nulabilita
 - Řešení výjimek
- Snaží se o co nejméně boxing/unboxing operací
 - Co nejpřesnější mapování na typy

Kompilace

- Tého blok je pomocí standardního JITteru přeložen do nativního kódu
- JITter je zavolán na úrovni metod
- Jakmile je blok zkompilován pointer ukazuje přímo na implementaci funkce
- Žádné další „invocation costs“
- Nemusí se přistupovat tak často k metadatům

Výkon UDF

- Rychlejší volací cesta než T-SQL uložené procedury
- Rychlejší manipulace se stringy, počítání a pod – z toho jak funguje CLR
- Lepší na intenzivní počítání s menším přístupem k datům
- Ale T-SQL je rychlejší na častějších operacích s Db

Výkon – UD agragate

- Je rychlejší, než agragace založená na kursorrech
- Ale pomalejší než build in agregační funkce

Výkon TVF

- Rychlejší než extended stored procedures
- Dokáže data brát postupně, čímž omezí paměťovou náročnost
 - Např Csv soubor vrácený jako relace

Pole a Kursory, textové řetězce

- Pokud T-SQL potřebuje brát data lépe vyjádřitelná v poli je CLR rychlejší ve zpracovávání a procházení polem
- Varchar se namapuje na SqlString, nebo SqlChar
 - SqlString – jako string namapuje celý řetězec do paměti
 - SqlChar – Stremovací rozhraní
 - Lepší pro LOBy

Serializace

- Uživatelské typy jsou serializovány pro potřeby ukládání na SQL serveru
- Protože jsou v podstatě nadstavbou nad skalárním systémem
- SQL server implementuje serializaci
 - Format.Native
 - Je možné si napsat vlastní
 - Nicméně kvůli výkonu je doporučována Format.Native

Normalizace

- Relační operace jako porovnávání a třídění operuje přímo nad binární reprezentací hodnoty UDT
- Je to způsobeno ukládáním normovaných(binárně setříděných) stavů UDT na disku
- Nemusí se pro porovnávání instancovat a volat metody
- Umožňuje vytvoření histogramů,indexů apod
- Normalizované UDT jsou takřka stejně výkonné jako vestavěné typy

Jak se to nasazuje

- Manual Deployment
- Build a nasazení pomocí Visual Studio

Manual Deployment

- Programátor napíše sestavu managed objektů
- Rutiny musí být static (shared ve VB)
- UDT a agregace jsou napsány jako samostatné objekty
- Vytvoří se assembly

Manual Deployment

- Nahrajeme na server
- Zavolá se rutina DDL CREATE ASSEMBLY
 - V této rutině se mohou specifikovat práva
- Vytvoří se T-SQL procedury, které volají funkce z assembly
 - Nebo se jen přiřadí jméno
- Pak se mohou volat s T-SQL navzájem

Upozornění

- Nestačí nahrát dll na server
- Server vždy operuje tu, která je v paměti od posledního Load
- Musí se zavolat explicitně
 - Nebo při restartu serveru

Nevýhody

- Dll je nahrána do celé instance serveru, nedá se použít pouze pro jednu Databázi
- Jiné formátování connection stringů v ADO uvnitř

Build, nasazení a debugging ve VS

- Ve VS 2005 přibyl nový typ projektu
 - SQL server projekt
- Po buildnutí se na připojeném serveru automaticky vytvoří obalovací funkce
- Deployment také nahraje .pdb soubory pro debugging
- Nezáleží na připojení(tabular vs http)

Debugging II

- Je možné debugovat i již nasazenou assembly
- Klasickým attach to process

IS NULL vs IsNull

- Pokud je třeba zjistit v dotazu zda není objekt NULL jsou dvě možnosti
- Query 1:
SELECT id
FROM Points
WHERE NOT (location IS NULL) -- equivalently: WHERE
location IS NOT NULL
- Query 2:
SELECT id
FROM Points
WHERE location.IsNull = 0

Query 1

- `SELECT id`
`FROM Points`
`WHERE NOT (location IS NULL) -- nebo: WHERE location`
`IS NOT NULL`
- U této metody je použito normální handling NULLu, pokud je objekt NULL, tak se nedeserializuje a rovnou se vrátí že je nebo není NULL

Query 2:

- `SELECT id`
`FROM Points`
`WHERE location.IsNull = 0`
- Tady se pro každý objekt v databázi provede deserializace, a otestuje se tento parametr
- Je to daleko pomalejší

Kdy tedy použít IsNull property?

- Pokud potřebujeme tuto hodnotu zjistit z CLR kódu
- Server potřebuje rozhodnout, že je objekt NULL, tak aby mohl používat nativní NULL handling
- Pokud použijeme na NULL objektu, vrátí se NULL, ne 1 jak by se dalo předpokládat

Další Výhody SQL 2005 s CLR

- XML – rozšířená podpora
 - Ať už ze strany serveru
 - Tak i samotný CLR má operace s XML vymyšlené
- Snadnější přenositelnost mezi platformami

Zdroje

- <http://www.clipcode.biz/stream/CLRInSQLSvr.pdf>
- <http://blogs.msdn.com/sqlclr>
- <http://msdn2.microsoft.com/en-us/library/ms131047.aspx#>
- <http://msdn2.microsoft.com/en-us/library/ms131075.aspx>
- http://searchsqlserver.techtarget.com/tip/1,289483,sid87_gci1165410,00.html?bucket=ETA
- <http://davidhayden.com/blog/dave/archive/2006/04/18/2917.aspx>
- <http://www.developer.com/net/net/article.php/3556571>
- <http://www.sqlskills.com/resources/Whitepapers/SQL%20Server%20DBA%20Guide%20to%20SQLCLR.htm>
- <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsq190/html/mandataaccess.asp>