



Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Eletrónica e
Telecomunicações e de Computadores

Mestrado em Engenharia Informática e de Computadores
Arquitetura de Sistemas Distribuídos
2013/2014
Relatório Aula Prática Nº 5

Nome	Nº de Aluno	E-mail
Rui Miranda	A32342	a32342@alunos.isel.pt
David Coelho	A21359	a21359@alunos.isel.pt
Frederico Ferreira	A7066	a7066@alunos.isel.pt

Introdução

Este trabalho prático tem como base o uso de “Managed Code” para criar objectos numa base de dados SQL Server, permitindo complementar o Transact-SQL com o uso das linguagens disponíveis em .NET Framework.

A integração de CLR pode ser usada para criar *Stored Procedures*, *Triggers*, *User Defined Functions*, *User Defined Types* e *Aggregates*, usando, nesta prática, a linguagem C#.

A integração destes novos modulos na base de dados precisa de ser “autorizada” com a activação da opção **clr enabled**.

Exercício 1

1. Crie no visual studio um projecto SqlServer, adicione uma user defined function (SQL CLRc#) com o código fornecido no ficheiro Ex1.cs.

a) Justifique o que acontece quando faz “publish” da solução. Que soluções existem para o problema? Adapte uma delas e faça “publish” da solução.

b) Justifique o que observa se na base de dados ASI tentar executar o seguinte código seguinte:

```
create table t (i int primary key, j as dbo.f() persisted)
```

Alínea a)

Implementação

Na preparação da base de dados para este exercício usámos o ficheiro batch **0.run.setupBD.bat** que em SQLCMD executa o script SQL contido no ficheiro **0.run.setupBD.sql** (esta construção permite a todos os elementos do grupo executar todos os passos isolando num ficheiro de configuração as diferenças de localização das bases de dados e nome dos servidores).

Fazendo *publish* do código fornecido obtemos o seguinte erro:

```
CREATE ASSEMBLY failed because type 'UserDefinedFunctions' in safe assembly 'p5ex1_SqlClr' has a static field 'i'. Attributes of static fields in safe assemblies must be marked readonly in Visual C#, ReadOnly in Visual Basic ....
```

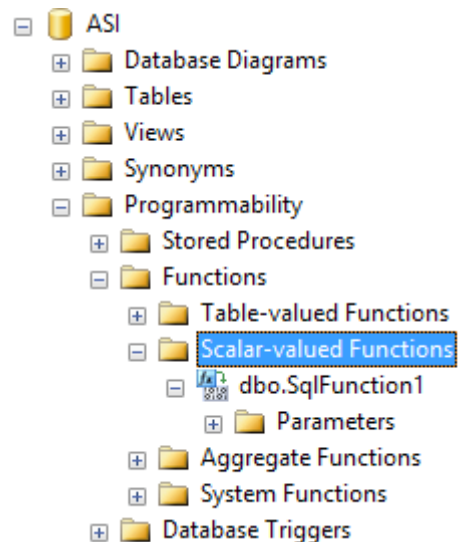
A quando do carregamento de um novo código (assembly) como uso de **Create Assembly** o *Managed Code* usa o *CAS (code access security)* para validar um conjunto de restrições que dependem do nível indicado **SAFE**, **EXTERNAL_ACCESS** e **UNSAFE**. Cada um destes níveis define um conjunto de restrições e verificações em runtime, que são incrementais e em que **SAFE** é o modo mais restritivo. Os níveis **External_Access** e **Unsafe**, precisam de privilégios adicionais ao nível do SQL Server, para o utilizador que for criar o assembly, além de um processo de assinatura dos módulo com uso de chaves assimétricas ().

Uma das restrições para os modos **Safe** e **External_access** é a impossibilidade de manutenção de “estado”, pelo que o uso de objectos **static** está restringida a atividades *readonly*.

As 2 opções seriam a create do assembly com nível **Unsafe**, ou a inclusão de **readonly** na declaração da variável **i**. A nossa opção foi pela segunda, para manter o nível **SAFE** que deverá ser usado sempre, salvo quando exista uma impossibilidade justificável e sem alternativa:

```
readonly static int i;
```

A figura do lado mostra a função criada na estrutura da BD.



Alínea b)

Implementação

A tentativa de execução de:

```
create table t (i int primary key, j as dbo.f() persisted)
```

termina com o erro:

```
Msg 4936, Level 16, State 1, Line 1
```

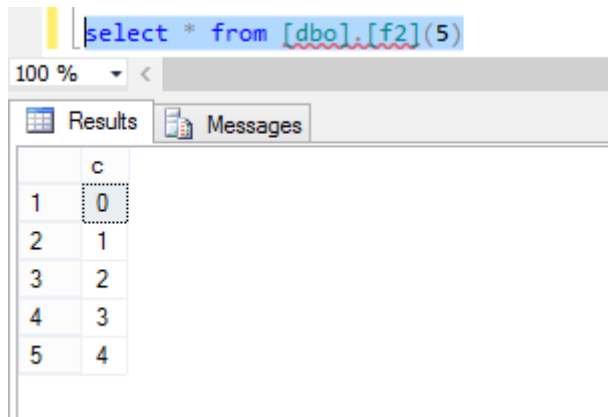
```
Computed column 'j' in table 't' cannot be persisted because the column is non-deterministic.
```

Uma vez que a função não foi (nem podia ser) classificada como determinística.

Existem diversas propriedades associadas às UDF (*User Defined Function*) que determinam como o SQL Server pode indexar colunas que usam essa função. Essas propriedades são:

- *Determinism* - uma função com os mesmos parâmetros de input e mesmo estado da base de dados retorna sempre o mesmo valor;
- *Precision* – uma função é dita precisa se não envolver operações de vírgula flutuante (onde os valores são compostos com mática e expoente e podem ter arredondamentos que alterem o resultado final);
- *Data Access* – para indicar se a UDF tem acesso (de leitura) a dados do utilizador;
- *System Data Access* – para indicar se a UDF tem acesso a metadados da base de dados local, dentro do contexto das permissões do utilizador;
- *IsSystemVerified* – indica se o determinismo e precisão são verificáveis pelo motor da base de dados e é *false* para as funções CLR.

A figura do lado mostra a execução da função criada, para gerar uma tabela com 5 elementos.



	c
1	0
2	1
3	2
4	3
5	4

Exercício 2

2. Repita a fase inicial do problema anterior, mas considerando o ficheiro Ex2.c. Justifique o que acontece quando tenta fazer “publish” do projecto. Como soluciona o problema?

Implementação

Ao tentar fazer *publish* do código Ex2.cs obtemos o seguinte erro:

Msg 10306, Level 16, State 1, Procedure f2, Line 1 The SqlFunctionAttribute of the Init method for a CLR table-valued function must set the FillRowMethodName property.

Uma *CLR Table-Valued Function* é um tipo de *Managed Code* que retorna um conjunto de valores (table) com base num **IEnumerable** ou num **IEnumerator** object, sendo necessário fornecer a constituição da tabela (com a diretiva **TableDefinition**) e uma função/ método para apresentar os valores com a estrutura indicada (a diretiva **FillRowMethodName**).

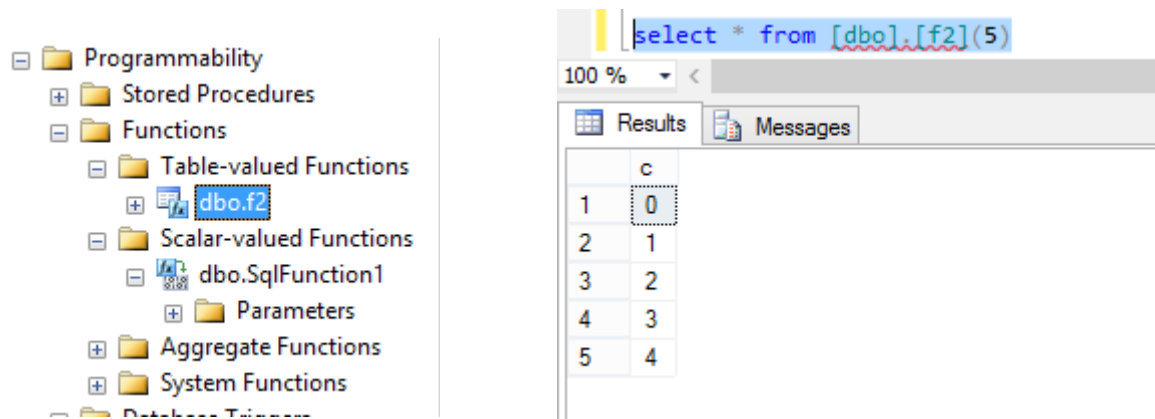
No exemplo do código acima FillRowMethodName não tinha sido fornecido, não sendo possível criar a *TVF* sem esse elemento.

Para solucionar o problema basta portanto criar o método e acrescentar a sua referência:

```
[Microsoft.SqlServer.Server.SqlFunction( TableDefinition = "c int",  
    FillRowMethodName= "trataLinha")]
```

```
.....  
public static void trataLinha(object linha, out int c)  
{  
    SqlInt32 dr = (SqlInt32)linha;  
    c = (int)dr;  
}  
.....
```

As figuras abaixo mostram o resultado da publicação e da execução função escalar criada:



Exercício 3

3. No visual studio, inclua num projecto Sql Server um gatilho com o código existente no ficheiro Ex3.c. Considere, também, o código TSQL seguinte, executado sobre a base de dados ASI:

```
create table t1 (c1 int, c2 int)
create table t2 (c1 int)
insert into t1 values(1,1)
insert into t1 values (2,2)
insert into t1 values(3,3)
update t1 set c2 = null where c1 =1 or c1 = 3
```

Utilize o debugger do Visual studio para determinar qual o erro que faz com que nunca apareçam linhas na tabela t2 como resultado da instrução update mostrada no código.

Implementação

O trigger fornecido apresentava um erro na comparação de um valor null dentro de um comando sql:

```
cmd.CommandText = "insert into t2 select c1 from inserted where c2 = null";
```

A correção foi simples e passou pela substituição pelo seguinte:

```
cmd.CommandText = "insert into t2 select c1 from inserted where c2 is null";
```

Incluir aqui algo sobre o processo de debug

Exercício 4

4. Construa um “user defined type” que permita manter informação sobre rectângulos, dadas os comprimentos dos lados (unidades sempre fixas) e que possibilite código do tipo:

```
create table tr(i int, r Rectangulo)
insert into tr values(1, '(1,3)')
insert into tr values(2, '(2,4)')
select r.l1 as l1, r.l2 as l2, r.Area as a from tr
```

Se quiser, pode usar a expressão regular:

```
@"(\\((?<11>[0-9]+)\\,(?<12>[0-9]+)\\))"
```

Implementação

Um *User Defined Type* pode ser feito com base numa estrutura ou num objecto, com atributos públicos que correspondem aos campos do tipo e tem de implementar métodos para:

- obter a versão Null do objecto;
- validar se é Null

- extrair de uma string os valores do novo tipo, com base na representação escolhida
- representar o tipo numa string (a função inversa da anterior)
- e ainda um método para serialização do objecto caso não seja usada a forma nativa

A solução está implementada no projecto **P5Ex4** dentro da diretoria Exercício4.

A figura abaixo apresenta o resultado da utilização da estrutura criada dentro de uma nova tabela.

```

create table tr(i int, r Rectangulo)

insert into tr values(1,'(1,3)')
insert into tr values(2,'(2,4)')
insert into tr values(3,null)
insert into tr values(4,'null')

select * from tr

select r.l1 as l1, r.l2 as l2, r.Area as a from tr
  
```

	i	r
1	1	0xBFF0000000000000C008000000000000C0080000000000...
2	2	0xC000000000000000C010000000000000C0200000000000...
3	3	NULL
4	4	NULL

	l1	l2	a
1	1	3	3
2	2	4	8
3	NULL	NULL	NULL

Query executed successfully at 23:15:00 | (local) (11.0 SP1)

Data Tools Operations

Publish completed successfully

Exercício 5

5. Construa um “user defined aggregate” que permita calcular o máximo das áreas dos rectângulos de uma coluna de uma tabela, como, por exemplo:

```
select dbo.MaxArea(r) from tr
```

que produzirá um valor real.

Implementação

Neste ponto da prática foi construído um *User Defined Aggregate* que permite obter o máximo de um rectângulo numa dada coluna de uma tabela.

Uma função de agregação, tem de a partir do tipo de dados base a usar implementat os seguintes métodos:

- Init() para iniciar o cálculo
- Accumulate() para inserir cada um dos valores

- Merge() para permite a junção de duas agregações na produção do agregado em causa
- Terminate() para apresentar o resultado final resultante do processo de agregação.

A implementação foi acrescentada em **P5Ex4** à definição do tipo Rectângulo usada no agregador criado. Dentro da diretoria Exercício5 o ficheiro **5.query.Aggregate.sql** foi usado no teste do agregado criado.

A figura seguinte mostra o resultado de testes de execução da função de agregação MaxArea.

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query with four statements: a SELECT from dbo.tr, and three calls to the dbo.MaxArea function with different filters. The bottom pane shows the results of these queries. The first query returns a table with 4 rows. The subsequent three queries return single-row results for the MaxArea function with filters i > 3, i < 2, and i > 3 respectively. A status bar at the bottom indicates the query executed successfully.

```

select i, r.l1, r.l2, r.Area from dbo.tr
select dbo.MaxArea( r) from tr
select dbo.MaxArea( r) from tr where i > 3
select dbo.MaxArea( r) from tr where i < 2

```

	i	(No column name)	(No column name)	(No column name)
1	1	1	3	3
2	2	2	4	8
3	3	NULL	NULL	NULL
4	4	NULL	NULL	NULL

	(No column name)
1	8

	(No column name)
1	0

	(No column name)
1	3

Query executed successfully. | FFSS (11.0 SP1) | F

Glossário

<i>Managed Code CLR integration</i>	Refere código que corre dentro do CLR (<i>common language runtime</i>) <i>CLR hosted in Microsoft SQL Server</i>
---	---