Uložené procedúry a funkcie

1 Reťazcové príkazy

String Command - Reťazcový príkaz a EXEC sp_executesql
a1) bez parametra:
a2) s parameterom:
b) parameter @i s pretypovaním:
c) Command s 3 parametrami: dopyt, typ (bez Declare), hodnota

Príklady
a1) bez parametra:

```
DECLARE @txt1 NVARCHAR(20),
            @txt2 NVARCHAR(40), -- Pozor – 30 zle.
            @txt3 NVARCHAR(50)
SET @txt1 = 'Use OsobaVztah '
-- Dodajte filter:  WHERE id<4
SET @txt2 = 'SELECT * FROM Osoba WHERE id < 4'
SET @txt3 = @txt1 + @txt2
print @txt3

EXEC sp_executesql @txt3
```

Lepšie s parameterom:
b) parameter @i s pretypovaním:

```
DECLARE @i INT;
SET @i = 4;
DECLARE @txt1 NVARCHAR(20),
            @txt2 NVARCHAR(30),
            @txt3 NVARCHAR(50)
SET @txt1 = 'Use OsobaVztah '
SET @txt2 = 'SELECT * FROM Osoba WHERE id <'
SET @txt3 = @txt1 + @txt2 + CAST(@i AS nvarchar(10))
print @txt3

EXEC sp_executesql @txt3
```

c) Command s 3 parametrami: dopyt, typ (bez Declare), hodnota (Tu treba N'... ')

```
EXECUTE             sp_executesql
                    N'USE OsobaVztah
                    SELECT * FROM Osoba
                    WHERE id < @i',    -- dopyt
                    N'@i int',         -- typ (bez Declare)
                    @i = 4;                  -- hodnota(bez SET)
```

d) **Príklad (cvič.)**: Vytvor tabulku T1 s 3 alebo 20 (!!!) stĺpcami

2 Uložená procedúra

Uložená procedúra (Stored Procedure - SP) je uložená kolekcia SQL (DDL, DML) a T-SQL príkazov s vstupno/výstupnými parametrami a vrátenou hodnotou stavu.

SP môžeme vytvoriť pre dočasné (#meno, ##meno2) alebo stále použitie.

Typy SP:
- A) systémové
- B) uživateľské
    - T-SQL
    - CLR

2A) Systémové procedúry (T-SQL)
A1) Všetky DB
A2) Všetky tabulky
A3) Všetky stlpce
A4) Výpis kódu SP

2B) Uživateľské uložené procedúry:
B1) SP bez parametrov
B2) SP s IN @parametrom
B3) SP S IN & OUT parametrami
B4) SP ako "číselný" výraz - RETURN číslo
B5) Čo používa/od čoho závisí object/SP

Výhody	**Nevýhody**
- modulárne programovanie - rýchlejší výpočet - kešovanie exekučného plánu - redukcia výmen po sieti	- limitované programovanie – cykly - portabilita – na druhý DBMS, napr. Oracle

Príkazové konštrukcie
BEGIN…END
IF…ELSE
WHILE
GOTO
RETURN, WAITFOR, BREAK, CONTINUE

3 Funkcia

Podobne funkciám iných programovacích jazykov, funkcie v MS SQL Server majú vstupné parametre a po vykonaní potrebných akcií, výpočtov vráti hodnotu, ktorá môže byť skalárna alebo množinová (tabuľka) hodnota.

Typy SP:
- A) systemové
- B) užívateľské
    a) Skalárne funkcie - 'FN'
    b) Funkcie s tabuľkovou hodnotou
      b1) štandardnými/default stĺpcami - 'IF'
      b2) definovanými stĺpcami - 'TF'

A) Systémové (štandardné) funkcie
- Štandardné
  - Systémové funkcie - SUSER_NAME(), @@ERROR, @@CURSOR_ROWS
  - T-SQL funkcie - GETDATE(), ...
    - agregačné f.
    - poradové (ranking) f.
    - f. dátumu a času
    - f. matematické
    - f. reťazcové
    - f. systémové - CASE, ISDATE, ISNULL, @@ERROR, NULLIF, COALESCE – vráti prvú nie NULL hodnotu
    - kurzorové f.
    - metadata f. - DB_ID, OBJECT_ID, COL_NAME
    - bezpečnostné f.
    - štatistické
    - funkcie na obrázky

Výhody a nevýhody – pozri SP.

4) Systémové pohľady

View sys (system) poskytuje metadata
- Databases a Files Catalog Views
    - sys.databases
- Object Catalog Views
    - sys.tables - catalog view sa dedí od:
    - sys.objects
    - sys.columns
    - sys.foreign_keys
    - sys.indexes
    - sys.stats
    - sys.views

    - sys.procedures
    - sys.triggers
- Information Schema Views
    - DB1.INFORMATION_SCHEMA.TABLES
    - DB1.INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    - DB1.INFORMATION_SCHEMA.CHECK_CONSTRAINTS
    - DB1.INFORMATION_SCHEMA.COLUMNS
    - DB1.INFORMATION_SCHEMA.VIEWS

4 Príklady

2A) Systémové procedúry (T-SQL)

a) Všetky DB:

```
EXEC sp_databases;        ------ SELECT * FROM sys.databases
```

b) Všetky tabuľky:

```
use OsobaVztah
GO
EXEC sp_tables    ---- SELECT * FROM sys.tables;


EXEC sp_tables @TABLE_OWNER = dbo
```

c) Všetky stĺpce: !!! Ctrl + T , Ctrl + D (, Ctrl + R)

```
EXEC sp_columns Osoba

EXEC sp_columns @table_name = Osoba, @column_name = 'm%'
SELECT COL_NAME(OBJECT_ID('Osoba'), 2)
```

d) Kód SP:

```
EXEC sp_helptext sp_columns -- trigger, view

---- Hlavicka:
--SELECT OBJECT_DEFINITION (OBJECT_ID(N'OsobaVztah.dbo.sp_columns'));
```

2B) Užívateľské uložené procedúry

!!! **Design Patterns** Guru's Guide to SQL Server™ Stored Procedures, XML, and HTML, By Ken Henderson

Faktorial - SP

```
USE maz
IF OBJECT_ID('dbo.sp_Faktorial') IS NOT NULL DROP PROC dbo.sp_Faktorial
GO

CREATE PROC dbo.sp_Faktorial @n decimal(38,0), @fak decimal(38,0)
OUTPUT -- or simply OUT
AS
DECLARE @n_old decimal(38,0)
```

```
      IF (@n< = 1) SET @fak = 1
      ELSE
      BEGIN
        SET @n_old = @n-1
        EXEC dbo.sp_Faktorial @n_old, @fak OUT -- Rekurzia
        SET @fak = @fak*@n
        --IF (@@ERROR<>0) RETURN(-1)
      END
RETURN(0)
GO

DECLARE @fak decimal(38,0)
EXEC dbo.sp_Faktorial 32, @fak OUT
SELECT @fak  -- 263130836933693530167218012160000000
```

3B) Užívateľské funkcie

a) Skalarne funkcie - 'FN'

```
IF OBJECT_ID('f_suma', 'FN') IS NOT NULL DROP FUNCTION
f_suma;
GO
CREATE FUNCTION f_suma( @c1 int, @c2 int ) RETURNS int
AS
BEGIN
  RETURN (@c1+@c2)
END
GO
SELECT dbo.f_suma(1, 2)  -- dbo. !!!!!!!!!!!!!!!
```

b1) Inline tabuľkové funkcie bez deklarácie stĺpcov tabuľky

```
USE OsobaVztah;
GO

IF OBJECT_ID('Hm3', 'IF') IS NOT NULL DROP FUNCTION Hm3;
GO

CREATE FUNCTION Hm3(@v int)
     RETURNS Table
AS
     RETURN (SELECT * FROM Osoba
                    WHERE vaha > @v
                )
GO

SELECT * FROM Hm3( 70);
GO
```

b2) Tabuľková funkcia s definovanými stĺpcami - 'TF'

```
IF OBJECT_ID('ftab', 'TF') IS NOT NULL DROP FUNCTION ftab;
GO

CREATE FUNCTION ftab ( @min INT, @max INT, @by int )
     RETURNS @O2 TABLE ( xxx INT )
AS
BEGIN
    WHILE @min <= @max
    BEGIN
        INSERT INTO @O2 ( xxx ) VALUES ( @min )
        SET @min = @min + @by
    END

    RETURN
END
GO
SELECT * FROM ftab ( 10, 30, 2 )
```

Vráťte hodnoty zo stredu tabuľky - vynechajte zo začiatku a konca

```
USE tempdb
GO
IF OBJECT_ID('S1') IS NOT NULL DROP table S1
GO
CREATE TABLE tempdb..S1 (x int)
Insert S1 Values (10),(20),(30),(40),(50),(60),(70)
GO
-- 4 pld.
SELECT a.x ax, b.x bx FROM tempdb..S1 a CROSS JOIN tempdb..S1 b

SELECT a.x ax FROM tempdb..S1 a CROSS JOIN tempdb..S1 b GROUP BY a.x

SELECT a.x ax, COUNT(CASE WHEN b.x <= a.x THEN 1 ELSE NULL END)
       FROM tempdb..S1 a CROSS JOIN tempdb..S1 b GROUP BY a.x
  HAVING COUNT(CASE WHEN b.x <= a.x THEN 1 ELSE NULL END) > 2
-- Po 4 pld. uz je to lahke
IF OBJECT_ID('dbo.Od_Stredu', 'IF') IS NOT NULL -- aj bez dbo
DROP FUNCTION dbo.Od_Stredu;                    -- aj bez dbo
GO
CREATE FUNCTION dbo.Od_Stredu(@bez int)          -- aj bez dbo
RETURNS TABLE
AS
RETURN(
  SELECT a.x ax FROM tempdb..S1 a CROSS JOIN tempdb..S1 b
  GROUP BY a.x
  HAVING COUNT(CASE WHEN b.x <= a.x THEN 1 ELSE NULL END) > @bez
     AND COUNT(CASE WHEN b.x >= a.x THEN 1 ELSE NULL END) > @bez
)
GO
```

```
SELECT * FROM dbo.Od_Stredu(0) ORDER BY ax
SELECT * FROM dbo.Od_Stredu(2) ORDER BY ax
```

5 Systémové pohľady = Catalog views + Information Schema Views + …

http://msdn.microsoft.com/en-us/library/ms177862(v=sql.120).aspx http://msdn.microsoft.com/en-us/library/ms189783(v=sql.120).aspx http://msdn.microsoft.com/en-us/library/ms186778(v=sql.120).aspx

Systémové pohľady zatriedené do kolekcií ako

Catalog Views, Inf.Sch.Views

vracajú rôzne metadata.

Catalog views, ako Object Catalog View *sys.xxx*, vrátia informácie ktoré používa SQL Sever DB Engine:

```
-- Pozri vyssie v casti 4 Priklady pri Systemovych procedurach za --
SELECT * FROM sys.databases
USE OsobaVztah;
-- SELECT * FROM sys.objects – Pozri nizsie
SELECT * FROM sys.tables
SELECT * FROM sys.columns where OBJECT_NAME([Object_ID]) = 'Osoba'

-- syscolumns, sys.all_columns, sys.triggers, select * from sys.procedures

SELECT t.name,c.name FROM sys.tables t
                    join sys.columns c on t.Object_ID = c.object_id
```

```
a) Dobre:
use Poliklinika
SELECT * FROM INFORMATION_SCHEMA.TABLES
b) Lepsie:
use master
SELECT * FROM Poliklinika.INFORMATION_SCHEMA.TABLES


SELECT * FROM OsobaVztah.INFORMATION_SCHEMA.TABLES
SELECT * FROM OsobaVztah.INFORMATION_SCHEMA.TABLE_CONSTRAINTS
SELECT * FROM OsobaVztah.INFORMATION_SCHEMA.CHECK_CONSTRAINTS
SELECT * FROM OsobaVztah.INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME =
'Osoba';
SELECT * FROM OsobaVztah.INFORMATION_SCHEMA.VIEWS


SELECT * FROM sys.objects
```

Typ objektu:

U = Table (user-defined)
V = View
P = SQL Stored Procedure
FN = SQL scalar function
IF = SQL inline table-valued function
TF = SQL table-valued-function
TR = SQL DML trigger

AF = Aggregate function (CLR)
C = CHECK constraint
D = DEFAULT (constraint or stand-alone)
F = FOREIGN KEY constraint
FS = Assembly (CLR) scalar-function
FT = Assembly (CLR) table-valued function
IT = Internal table
PC = Assembly (CLR) stored-procedure
PG = Plan guide
PK = PRIMARY KEY constraint
R = Rule (old-style, stand-alone)
RF = Replication-filter-procedure
S = System base table
SN = Synonym
SO = Sequence object
SQ = Service queue
TA = Assembly (CLR) DML trigger
TT = Table type
UQ = UNIQUE constraint
X = Extended stored procedure

DÚ
- Pomocou SP rieste:

```
SELECT t.name,c.name FROM sys.tables t
                    join sys.columns c on t.Object_ID = c.object_id
```