Uložené procedúry a funkcie. Systémové pohľady. SysColumns. Ošetrenie chýb.

1 Reťazcové príkazy

Jazyk T-SQL ponúka viac programovacích modulárnych konštrukcií, ako reťazcový príkaz, uložená procedúra, funkcia, trigger.

Reťazcový príkaz (String Command) https://msdn.microsoft.com/en-us/library/ms188332.aspx

```
EXEC sp_executesql @str_sql [ , , ]
```

a) Reťazcový príkaz s jedným argumentom
- a1) `str_sql` bez parametra
- a2) `str_sql` s (pretypovaným) parametrom

b) Reťazcový príkaz s viac argumentami
- b1) príkaz s 3 argumentami: `str_sql`, typ (bez Declare), hodnota
- b2) príkaz s *ľubovoľným* počtom argumentov

Príklady

a1) `str_sql` bez parametra:

```
-- USE osobaVztah
DECLARE @txt1 NVARCHAR(20),
        @txt2 NVARCHAR(40), -- Pozor – 30 nestaci, 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
```

Dalo by sa aj pomocou jednej premennej patričnej dĺžky:

```
…
SET @txt1 = 'Use OsobaVztah; SELECT * FROM Osoba WHERE id < 4'
```

Lepšie s parameterom:

a2) `str_sql` s (pretypovaným) parameter

```
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
```

b1) príkaz s 3 argumentami

Reťazec obsahuje premenné, typy a hodnoty ktorých stanovujeme parametrami:
`str_sql`, typ (bez Declare), hodnota (tu treba `N'... '`)

```
EXECUTE sp_executesql
        N'USE OsobaVztah
        SELECT * FROM Osoba
        WHERE id < @i',    -- 1 - dopyt
        N'@i int',         -- 2 - typ (bez Declare)
        @i = 4;            -- 3 - hodnota(bez SET)
```

b2) príkaz s *ľubovoľným* počtom argumentov

Posledný tretí *hodnotový* argument je možné zopakovať a typy môže zadať premenná:

4 argumenty

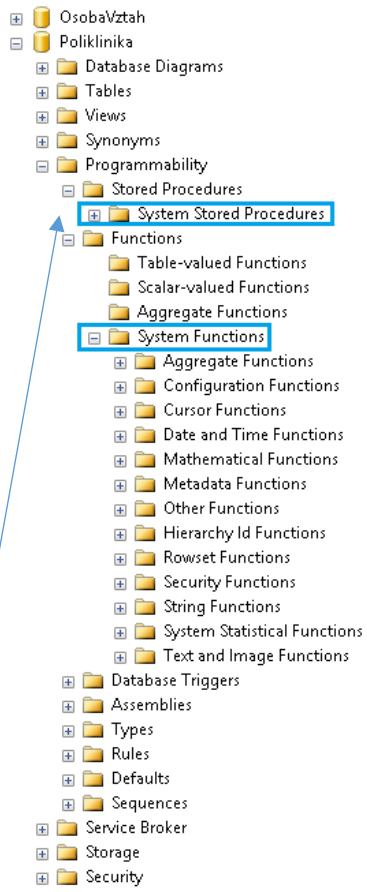
```
USE Poliklinika;
DECLARE @sql NVARCHAR(MAX), @prem NVARCHAR(MAX)
SET @sql = 'SELECT * FROM Navstevy WHERE den BETWEEN @d1 and @d2'
SET @prem = '@d1 DATETIME, @d2 DATETIME' -- !
EXEC sp_executesql @sql, @prem, @d1='1/1/2008', @d2='8/31/2008'
```

2 argumenty + 4 hodnotové argumenty

```
DECLARE @sql NVARCHAR(MAX), @prem NVARCHAR(MAX)
SET @sql = 'SELECT @i1+@i2+@i3+@i4'
SET @prem = '@i1 int, @i2 int, @i3 int, @i4 int' -- !
EXEC sp_executesql @sql, @prem, @i1='1', @i2='2', @i3='3', @i4='4' --10
```

c) **Príklad (cvič./DÚ)**: Vytvorme tabuľku T1 s 3 alebo 20 (!!!) stĺpcami pomocou reťazcového príkazu.

2 Uložená procedúra



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

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

Typy SP:
- A) systémové
- B) užívateľské
  - T-SQL
  - CLR (Common language runtime)

2A) Systémové procedúry (T-SQL)
- A1) Všetky DB `sp_databases`
- A2) Všetky tabulky `sp_tabes`
- A3) Všetky stlpce `sp_columns`
- A4) Výpis kódu SP a iné

```
Use master
Select count(*) from sys.sysobjects where xtype = 'P'
-- počet všetkých systémových SP 1382 - r. 2023
```

Pozri ďalší príklad na `sys.sysobjects` nižšie!!!

2B) Užívateľské uložené procedúry:
- B1) SP bez argumentu
- B2) SP s IN @argumentom
- B3) SP S IN & OUT argumentami
- 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 - je to prostriedok na zvýšenie bezpečnosti	- limitované programovanie – cykly - portabilita – na druhý DBMS, napr. Oracle

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

3 Funkcia

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

Typy funkcií

- A) systemové
- B) užívateľské
  - a) Skalárne funkcie - 'FN' - **scalar function**
  - b) Funkcie s tabuľkovou hodnotou
    - b1) štandardnými/default stĺpcami - 'IF' - **inline table-valued function**
    - b2) definovanými stĺpcami - 'TF' - **table-valued-function**

```
Use master
Select count(*) from sys.sysobjects where xtype = 'FN' or xtype = 'IF' or xtype
='TF'
-- počet všetkých systémových funkcií 135 - r. 2023
```

A) Systémové (štandardné) funkcie https://msdn.microsoft.com/en-us/library/ms174318.aspx

- 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 f.
  - funkcie na obrázky

Pozri odkazy z predchádzajúcej prednášky.

Výhody a nevýhody – pozri SP.

4) Systémové pohľady

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

SQL Server ukladá metadáta o jeho konfigurácii, objektoch, dátových typoch, obmedzeniach atď. v systémových tabuľkách, ktoré nie sú priamo dopytovateľné. Pre prístup k týmto metadátam SQL Server ponúka niekoľko možností.

Katalóg je miesto, kde sa okrem iného uchovávajú **schémy** (konceptuálne, interné) a všetky príslušné mapovania (koncepčné / interné). Inými slovami, **katalóg** obsahuje detailné informácie, **metadáta** týkajúce sa rôznych objektov samotného systému.

Cluster (> ~ databázový server) má **katalógy** (> ~ databázy) a katalóg má **schémy** (~tabuľky).

Podobne, **information schema views** vrátia informácie o metadátach databázy. SQL Server 2005, 2008 ... automaticky vytvorí 20 INFORMATION_SCHEMA pohľadov v každej databáze. https://www.simple-talk.com/sql/t-sql-programming/using-information-schema-views/

View sys (system) poskytuje metadata

- Databases & 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

.

```
SELECT * FROM sys.databases
SELECT * FROM Poliklinika.INFORMATION_SCHEMA.TABLES
```

- 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
  - INFORMATION_SCHEMA.ROUTINES a ďalšie.

Systémom generovaný kód na vytvorenie DB uz z existujúcej DB:

- *pravé klepnutie na konkrétnu DB – Tasks – Generate Scripts ... – iba tabulky.*

5 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'), 3) -- priezvisko
```

d) Kód SP:

```
       EXEC sp_helptext sp_columns -- trigger, view
-- NO: EXEC sp_helptext sp_executesql

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

2B) Užívateľské uložené procedúry

Uložená procedúra na faktorial – argumenty sú so zavínačom.

```
USE dbmaz
IF OBJECT_ID('dbo.sp_Faktor', 'P') IS NOT NULL DROP PROC dbo.sp_Faktor
GO

CREATE PROC dbo.sp_Faktor @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_Faktor @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_Faktor 32, @fak OUT
SELECT @fak  -- 263130836933693530167218012160000000
```

3B) Užívateľské funkcie - argumenty sú so zavínačom.

a) Skalárne 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 (a begin - end) - 'IF'
Stĺpce sa dedia cez SELECT https://www.sqlservercentral.com/articles/creating-and-using-inline-table-valued-functions

```
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
-- Parne cisla od min po max
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 nn riadkov

Na základe ĽUBOVOĽNÉHO KĽÚČA

```
USE tempdb
GO
IF OBJECT_ID('S1') IS NOT NULL DROP table S1
GO
CREATE TABLE tempdb..S1 (x int unique) -- kluc
Insert S1 Values (10),(20),(30),(40),(50),(60),(70)

-- Pohrame sa 3x s CROSS JOIN, GROUP BY a HAVING.
-- Pre kazde ax nech je bx >= ax:
SELECT a.x ax, b.x bx
   FROM tempdb..S1 a CROSS JOIN tempdb..S1 b WHERE b.x >= a.x order by ax
-- Kolko krat je ax <= bx?
SELECT a.x ax, COUNT(CASE WHEN b.x >= a.x THEN 1 ELSE NULL END) pocet
  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 -- vynechat HAVING
```

Po experimentoch už je to ľahké (vynechať zo začiatku a konca nn riadkov)

```
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(@nn 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) > @nn
     AND COUNT(CASE WHEN b.x >= a.x THEN 1 ELSE NULL END) > @nn
)
GO

SELECT * FROM dbo.Od_Stredu(0) ORDER BY ax
SELECT * FROM dbo.Od_Stredu(2) ORDER BY ax
SELECT * FROM dbo.Od_Stredu(9) ORDER BY ax
```

Efektívny kod (CTE alebo WITH tabuľky budú budúci týždeň):

```
IF OBJECT_ID('Od_Stredu_Efektivne', 'IF') IS NOT NULL
DROP FUNCTION Od_Stredu_Efektivne;
GO
CREATE FUNCTION Od_Stredu_Efektivne(@nn int)
RETURNS TABLE
AS
RETURN (
    WITH CCTE AS (
        SELECT x,
               ROW_NUMBER() OVER (ORDER BY x ASC) as rnk_asc,
               ROW_NUMBER() OVER (ORDER BY x DESC) as rnk_desc
        FROM tempdb..S1
    )
    SELECT x FROM CCTE
    WHERE rnk_asc > @nn AND rnk_desc > @nn
);
GO
SELECT * FROM Od_Stredu_Efektivne(2) ORDER BY x
```

4) 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/sys Views, INFORMATION_SCHEMA Views`
vracajú rôzne metadata.

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

```
-- Taktiez pozri vyssie priklady pri Systemovych procedurach
-- use master
SELECT * FROM sys.databases
```

```
USE OsobaVztah;
SELECT * FROM sys.tables
SELECT * FROM sys.columns where OBJECT_NAME([Object_ID]) = 'Osoba'
-- SELECT * FROM sys.objects -- Pozri nizsie

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

Ktorá tabuľka má koľko stĺpcov v danej DB:

```
USE Poliklinika
SELECT t.name tab,c.name stlp 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) Lepšie:

```
-- use master  -- alebo lubovolnu DB
SELECT * FROM Poliklinika.INFORMATION_SCHEMA.TABLES
SELECT * FROM Poliklinika.sys.tables
USE Poliklinika;  EXEC sp_tables @TABLE_OWNER = dbo

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

use poliklinika
SELECT COLUMN_NAME,* FROM INFORMATION_SCHEMA.COLUMNS
      WHERE TABLE_NAME = 'lekari'
-- <~~>
exec sp_columns [lekari]

SELECT * FROM sys.objects
```

Pozri príklad nižšie!!!

5) VYUŽITIE SysColumns

Rozdiely medzi SysColumns a sys.columns. Ich využitie.

- SysColumns patrí užívateľovi, kým sys.columns je Object Catalog View
- majú inú štruktúru stĺpcov
- SysColumns má viac riadkov

```
SELECT TOP 3 * FROM SysColumns            -- <=>
SELECT TOP 3 * FROM Master.dbo.SysColumns -- ~~
SELECT TOP 3 * FROM sys.columns           -- vacsi pocet stlpcov
```

```
SELECT count(*) FROM Master.dbo.SysColumns  -- viac ako 18 tisic
SELECT count(*) FROM SysColumns      -- asi tisíc dvesto <=> pre Polikl.
SELECT count(*) FROM sys.columns
```

Spôsoby vytvorenia súvislej postupnosti hodnôt

- Values a alias s názvom ako tabuľka

```
SELECT * FROM ( select mesprijem from poliklinika..pacienti ) xxxTab --alias-nazov
SELECT N FROM ( VALUES(-1),(-2),(-5) ) xxxTab(N) -- !!! tabulka s jenym stlpcom
```

- SysColumns a window funkcia

```
SELECT TOP 3 ROW_NUMBER() OVER (ORDER BY name) FROM SysColumns; #resp. Master.dbo.
```

	(No column name)
1	1
2	2
3	3

Využitie:

```
DECLARE @den TABLE (d DATETIME NOT NULL)
INSERT INTO @den (d)
  SELECT DATEADD(day, N, getdate())
       FROM (VALUES(-1),(-2),(-3)) a(N)
       ORDER BY N ASC
SELECT d FROM @den
```

	d
1	2021-03-03 22:54:57.743
2	2021-03-02 22:54:57.743
3	2021-03-01 22:54:57.743

```
-- <=>
DECLARE @den2 TABLE (d DATETIME NOT NULL)
INSERT INTO @den2 (d)
  SELECT TOP 3 DATEADD(day, -1 * (ROW_NUMBER() OVER (ORDER BY name)), getdate())
  FROM Master.dbo.SysColumns;
SELECT d FROM @den2
```

Dodatok

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

```
USE master;
Select xtype, count(xtype) Nb from sys.sysobjects
group by xtype
```

	xtype	Nb
1	IF	49
2	U	7
3	SQ	3
4	AF	9
5	X	132
6	FS	4
7	PC	3
8	P	1382
9	V	429
10	S	78
11	IT	8
12	TF	18
13	FN	40

```
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
```

6) Ošetrenie chýb

a1) sys.messages pre @@ERROR
a2) RAISERROR <--> PRINT:
b1) Starý spôsob: @@ERROR, @@ROWCOUNT;
b21) Nový spôsob: TRY - CATCH
b22) S užívateľskou procedúrou

```
---- a1) sys.messages pre @@ERROR
select * from sys.messages where message_id = 547

---- a2) RAISERROR <--> PRINT:
--    druhy par. severity: 0-10, 11-25:
--    treti par. state: 0-127
RAISERROR('Haha', 10, 1)
RAISERROR('Haha', 11, 1)
```

b1) Starý spôsob SQL Server 2000: @@ERROR, @@ROWCOUNT;

```
-- Uloz info o moznej chybe do premennych!
-- The DELETE statement conflicted with the REFERENCE constraint ...
USE Poliklinika
GO
DECLARE @ErrorVar INT;
DECLARE @RowCountVar INT;

DELETE FROM Lekari WHERE idL = 1;
-- Uloz hned info o moznej chybe do premennych a reaguj!
SELECT @ErrorVar = @@ERROR,
       @RowCountVar = @@ROWCOUNT;
IF (@ErrorVar <> 0)
  BEGIN
    PRINT N'Csaba';
    PRINT N' - Cislo chyby = ' + CAST(@ErrorVar AS
NVARCHAR(8));
    PRINT N' - Pocet vymazanych riadkov = ' +
CAST(@RowCountVar AS NVARCHAR(8));
  END
GO
```

b21) Nový spôsob: BEGIN TRY – BEGIN CATCH:

```
USE Poliklinika
GO

BEGIN TRY
     DELETE FROM Lekari WHERE idL = 1;
END TRY
BEGIN CATCH
      SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() AS ErrorState,
        ERROR_PROCEDURE() AS ErrorProcedure,
        ERROR_LINE() AS ErrorLine,
        ERROR_MESSAGE() AS ErrorMessage;
        RAISERROR ('Chyba ...', 11, 1)
END CATCH
```

Ďalší príklad:

```
BEGIN TRY
     SELECT GETDATE()
     SELECT 1/0 Bude_Chyba
     SELECT GETDATE() -- to sa uz nevykona
END TRY
BEGIN CATCH
```

```
        RAISERROR ('Chyba pri deleni', 11, 1)
END CATCH;
```

b22) S užívateľskou procedúrou Chyba, ktorá bude trvalo existoať v DB Poliklinika:

```
USE Poliklinika;
GO
IF OBJECT_ID ( 'Chyba', 'P' ) IS NOT NULL DROP PROCEDURE
Chyba;
GO
CREATE PROCEDURE Chyba
AS
     SELECT -- DALO BY SA AJ PRINT
          ERROR_NUMBER() AS ErrorNumber,
          ERROR_SEVERITY() AS ErrorSeverity,
          ERROR_STATE() AS ErrorState,
          ERROR_PROCEDURE() AS ErrorProcedure,
          ERROR_LINE() AS ErrorLine,
          ERROR_MESSAGE() AS ErrorMessage;
     RAISERROR ('TCs - Chyba! Pozri vysled dopytu', 11, 1)
GO

BEGIN TRY
      DELETE FROM Lekari WHERE idL = 1;
END TRY
BEGIN CATCH
        EXEC Chyba
END CATCH;
```

DÚ

- Pomocou SP rieste:

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


-- Statistics
--   https://learn.microsoft.com/en-us/sql/relational-
databases/statistics/statistics?view=sql-server-ver16
SELECT OBJECT_NAME(OBJECT_ID) as tblName,
OBJECT_ID, name,
stats_id
FROM sys.stats
WHERE OBJECT_NAME(OBJECT_ID) = 'navstevy'

SELECT OBJECT_NAME(OBJECT_ID) as tblName, *
FROM sys.stats_columns
WHERE OBJECT_NAME(OBJECT_ID) = 'lekari'
```

```sql
USE master;
DROP DATABASE OsobaVztah;
CREATE DATABASE OsobaVztah;
USE OsobaVztah;
GO
CREATE TABLE Osoba (
         id INT NOT NULL PRIMARY KEY,
         meno VARCHAR(10) NOT NULL,
         priezvisko VARCHAR(20) NOT NULL,
         rodne_priezvisko VARCHAR(20),
         dat_nar DATE NOT NULL,
         dat_smrti DATE,
         pohlavie CHAR(1) NOT NULL CHECK (pohlavie IN ('m','z')),
         vyska DEC(4,1) CHECK (vyska BETWEEN 30.0 AND 250.0),
         vaha DECIMAL(4,1),          -- To iste ako DEC(4,1)
         otec INT ,
         matka INT ,
         FOREIGN KEY (otec) REFERENCES Osoba(id),
         FOREIGN KEY (matka) REFERENCES Osoba(id)       -- cudzi kluc s menom
);
GO
CREATE TABLE Vztah
(
  id      INT  NOT NULL PRIMARY KEY,
  id_on   INT  NOT NULL,
  id_ona  INT  NOT NULL,
  od      DATETIME NOT NULL,
  do      DATETIME
);
GO
INSERT Osoba VALUES(1, 'Adam', 'Prvy', NULL, '1918.05.11', '1968.10.01', 'm', 180.0, 80.0, NULL, NULL);
INSERT Osoba VALUES(2, 'Eva', 'Prva', 'Druha', '1919.1.9', '1988.7.22', 'z', 160.0, 60.0, NULL, NULL);
INSERT Osoba VALUES(3, 'Zoly', 'Mudry', NULL, '1918.4.7', '19900923', 'm', 175.5, 75, NULL, NULL);
INSERT Osoba VALUES(4, 'NASta', 'Kovacova', 'Rostova', '1928.2.5', '1965.3.11', 'z', 155.0, 99, NULL, NULL);
INSERT INTO Osoba (id, priezvisko, meno, rodne_priezvisko, dat_nar, dat_smrti, pohlavie, vyska, vaha, otec,
matka )
   VALUES(5,'Urban' , 'Jozef', NULL, '1922.10.19', NULL, 'm', 199.5, Null, NULL, NULL); -- meno vs. priezvisko
INSERT INTO Osoba (id,meno, priezvisko, rodne_priezvisko, dat_nar, dat_smrti, pohlavie, vyska, vaha, otec, matka
)
VALUES(6, 'Maria', 'Urbanova', 'Novakova', '1937.12.8', NULL, 'z', 172.5, 57.5, 1, 2 ),
(7, 'Patrik', 'Novak', 'Novak', '1945.6.19', NULL, 'm', 182.5, 89.5, 1, 2 ),
(8, 'Patricia','Novakova', 'Haluskova', '1952.1.8', NULL, 'z', 143.5, 35, NULL, NULL),
(9, 'Michal', 'Kovac', 'Kovac', '1942.4.10', NULL, 'm', 167.0, 88, 3, 2 );
INSERT Osoba VALUES(10,'Roman', 'Kovac', 'Kovac', '1948.5.20', NULL, 'm', 179.5, 78.5, 3, 4 ), -- aj tak sa da
:)
(11,'Peter', 'Horvath', 'Horvath', '1959.7.2', '2000.12.31','m', 193.0, 110.5,NULL, NULL);
INSERT Osoba VALUES(12,'Lucia', 'Horvathova','Urbanova', '1959.1.13', NULL, 'z', 156.5, 45.5, 5, 6 );
INSERT Osoba VALUES(13,'Urban', 'Urban', 'Urban', '1957.3.31', NULL, 'm', 138.2, 24.5, 5, 6 );
INSERT Osoba VALUES(14,'DASa', 'Novakova', 'Novakova', '1970.7.17', NULL, 'z', 167.0, 55.0, 7, 8 );
INSERT Osoba VALUES(15,'Viera', 'Silna', 'Novakova', '1973.2.13', NULL, 'z', 169.5, 63.0, 7, 8 );
INSERT Osoba VALUES(16,'Vladimir','Silny', 'Silny', '1974.8.1', '2002.12.4', 'm', 175.5, 73.0, NULL, NULL);
INSERT Osoba VALUES(17,'Milena', 'Slaba', 'Slaba', '1979.9.14', NULL, 'z', 164.0, 64.0, NULL, NULL);
INSERT Osoba VALUES(18,'Jan', 'Horvath', 'Horvath', '1982.1.16', NULL, 'm', 159.5, 65.5, 11, 12 );
INSERT Osoba VALUES(19,'Zuzana', 'Silna', 'Silna', '2002.3.1', NULL, 'z', 158.5, 60.0, 16, 15 );
INSERT Osoba VALUES(20,'Zuzana', 'Slaba', 'Slaba', '1999.12.16', NULL, 'z', 171.5, 54.5, 16, 17 );
--INSERT Osoba VALUES(21,'Zuzana', 'Prava', 'Prava', '1990.11.26', NULL, 'z', 170.5, 60.5, 16, 17 );
--INSERT Osoba VALUES(22,'Zuzana', 'Lava', 'Lava', '1931.01.14', NULL, 'z', 195.5, 58.5, 16, 17 );
--INSERT Osoba VALUES(23,'Zuzana', 'Stredna', 'Stredna', '1945.04.08', NULL, 'z', 150.5, 87, 16, 17 );
GO
INSERT INTO Vztah VALUES (1,1, 2, '1937.6.1',  '1967.5.11' );
INSERT Vztah VALUES (2,3, 2, '1967.5.12', '1988.7.22' );
INSERT Vztah VALUES (3,3, 4, '1938.12.2', '1965.3.11' );
INSERT Vztah VALUES (4,5, 6, '1953.11.11', NULL       );
INSERT Vztah VALUES (5,7, 8, '1970.7.22', '1975.9.1'  );
INSERT Vztah VALUES (6,11,12,'1980.3.4',  '2000.12.31');
INSERT Vztah VALUES (7,16,15,'1997.7.31', '2002.12.4' );

SELECT * FROM Vztah;
SELECT * FROM Osoba;
```