4) Dočasné tabuľky a kurzory

1) Dočasné tabuľky http://msdn.microsoft.com/en-us/library/ms177399(SQL.90).aspx
2) Kurzory

1) Dočasné tabuľky

Ako vieme, tabuľku, ako výsledok dopytu, môžeme

- vytlačiť
- kombinovať s UNION, EXCEPT, INTERSECT alebo s ďalším dopytom
- **mať v pamäti** ako CTE alebo premennú @ tabuľku

  ```
  WITH T3(xx) AS (SELECT … ) SELECT …
  ```

  a znovupoužiť v dopytoch
- **uložiť**
  - trvalo ako VIEW (nie samotnú tabuľku, ale definujúci dopyt)

    ```
    CREATE VIEW Vmaz AS
          SELECT
    ```
  - dočasne, prechodne ako #, ## tabuľky
- sumarizovať (CUBE)

0) CREATE TABLE A(idA INT);
1) CREATE TABLE #A(idA INT); -- lokalna prechodna tabulka
2) CREATE TABLE ##A(idA INT); -- globalna prechodna tabulka
3) DECLARE @A TABLE(idA INT); -- premenna typu table

Lokálne # a globálne ## tabuľky sú vytvorené v tempdb a po ukončení sesie sa vymažú. Kým sa lokálna #A tabuľka je viazná k jednému query listu, k tej, v ktorej bola vytvorená, globálna ##A tabuľka je prístupná k všetkým quey listom.

```
USE tempdb;
GO

-- 1)
IF OBJECT_ID ('#A') IS NOT NULL DROP TABLE #A;
GO
CREATE TABLE #A(col1 INT);
GO

INSERT #A VALUES(1)
INSERT #A VALUES(2)

SELECT * FROM #A

-- 2)
IF OBJECT_ID ('##A') IS NOT NULL DROP TABLE ##A;
GO
CREATE TABLE ##A(col1 INT);
GO
```

```
INSERT ##A VALUES(10)
INSERT ##A VALUES(20)

SELECT * FROM ##A

-- 3)
DECLARE @A TABLE(col1 INT)
INSERT   @A VALUES(100)
INSERT   @A VALUES(200)

SELECT * FROM @A
DELETE FROM @A
SELECT * FROM @A

INSERT @A SELECT * FROM #A
SELECT * FROM @A
```

Zopakovanie

```
-- -1)
USE tempdb;
IF OBJECT_ID ('T') IS NOT NULL DROP TABLE T;
GO
CREATE TABLE T(col1 INT);
GO

-- 0)
INSERT T VALUES(1)
INSERT T VALUES(2), (3), (4)

-- a)
INSERT INTO T
     SELECT 7 UNION ALL
     SELECT 8

-- b) V Insert sa da pouzivat Select
INSERT T SELECT * FROM T  -- az po CREATE TABLE!!

SELECT * FROM T

-- c) Vkladaj without INSERT
IF OBJECT_ID ('#B') IS NOT NULL DROP TABLE #B;
SELECT * INTO #B FROM T -- bez CREATE TABLE!!! FROM nutne!

SELECT * FROM #B
```

3) Kurzory

Kurzor je DB prostriedok na získanie prístupu k jednotlivým **riadkom** tabuľky DB. Spracovanie tabuliek po riadkoch pomocou kurzoru môže byť problematické, lebo DB-vé optimalizačné systémy ťažko nájdu samostatne optimálnu paralelnú verziu kódu s kurzorom. Napriek tomu, DB systémy podporujú kurzory (MS SQL, Oracle viac) na základe štandardu SQL-92.

Syntax.

1) Deklarácia
DECLARE kur1 CURSOR FOR SELECT …
2) Otvorenie + použitie: získanie
3) Zatvorenie

Podrobnejsie: http://msdn.microsoft.com/en-us/library/ms180169.aspx ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_6tsql/html/5d68dac2-f91b-4342-bb4e-209ee132665f.htm

```
ISO Syntax
DECLARE cursor_name [ INSENSITIVE ] [ SCROLL ] CURSOR
     FOR select_statement
     [ FOR { READ ONLY | UPDATE [ OF column_name [ ,...n ] ] } ]
[;]

Transact-SQL Extended Syntax
DECLARE cursor_name CURSOR [ LOCAL | GLOBAL ]
     [ FORWARD_ONLY | SCROLL ]
     [ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
     [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
     [ TYPE_WARNING ]
     FOR select_statement
     [ FOR UPDATE [ OF column_name [ ,...n ] ] ]
[;]
```

Štandardne: FORWARD_ONLY => FETCH NEXT FROM kur
SCROLL => FETCH FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE

```
FETCH
          [ [ NEXT | PRIOR | FIRST | LAST
                   | ABSOLUTE { n | @nvar }
                   | RELATIVE { n | @nvar }
              ]
              FROM
          ]
{ { [ GLOBAL ] cursor_name } | @cursor_variable_name }
[ INTO @variable_name [ ,...n ] ]
```

Skalárne funkcie pre kurzory

@@CURSOR_ROWS – konštanta vráti počet riadkov v kurzore
@@FETCH_STATUS – používa sa v cykle na zistenie, či kurzor ešte obsahuje riadok

0 FETCH statement was successful
-1 FETCH statement failed or the row was beyond the result set
-2 Row fetched is missing

Príklad 1: Vráťte prvú DB zo zoznamu DB pomocou sysdatabases a **kurzora.**

```
-- 1/3) Deklaracia
DECLARE db_cursor CURSOR  FOR
        SELECT name FROM master.dbo.sysdatabases
        WHERE name NOT IN ('master','model','msdb')
-- DECLARE @jaj VARCHAR(50)

-- 2/3) Otvorenie a pouzitie, ziskanie
OPEN db_cursor
FETCH NEXT FROM db_cursor  -- INTO @jaj

-- 3/3) Zatvorenie a uvolnenie
CLOSE db_cursor
DEALLOCATE db_cursor

-------------------------------------------
```

Priklad 2: Pokracovanie – vráťte zoznam všetkých DB.

```
-- 1/3)
DECLARE db_cursor CURSOR
   FORWARD_ONLY
FOR
SELECT name FROM master.dbo.sysdatabases
                    WHERE name NOT IN ('master','model','msdb')

-- 2/3)
OPEN db_cursor
DECLARE @dbName VARCHAR(50)
FETCH NEXT FROM db_cursor INTO @dbName
print cast(@@CURSOR_ROWS as char(10))

WHILE @@FETCH_STATUS = 0
BEGIN
      print @dbName
      FETCH NEXT FROM db_cursor INTO @dbName
END

-- 3/3)
CLOSE db_cursor
DEALLOCATE db_cursor
```

Vráťte názvy všetkých databáz s prislušnými tabuľkami.

```
-- Pomocou dvoch kurzorov, pritom jeden je retazcovy prikaz.
-- Vysledok do MESSAGES!!!
DECLARE @db VARCHAR(255)
DECLARE @tab VARCHAR(255)
DECLARE @cmd NVARCHAR(500)

DECLARE kurDB CURSOR FOR
SELECT name FROM master.dbo.sysdatabases
     WHERE name NOT IN ('master', 'msdb', 'model', 'ReportServer',
'ReportServerTempDB', 'tempdb') and name != 'AdventureWorks2014'
     ORDER BY name

OPEN kurDB

FETCH NEXT FROM kurDB INTO @db
WHILE @@FETCH_STATUS = 0
BEGIN
   print ''
   print @db + ':'
   SET @cmd = 'DECLARE kurTab CURSOR FOR SELECT table_name FROM [' +
@db + '].INFORMATION_SCHEMA.TABLES'
   EXEC (@cmd)  -- vytvor kurTab!
   OPEN kurTab

   FETCH NEXT FROM kurTab INTO @tab
   WHILE @@FETCH_STATUS = 0
   BEGIN
       print   '  ' + @tab
       FETCH NEXT FROM kurTab INTO @tab
   END

   CLOSE kurTab   DEALLOCATE kurTab

   FETCH NEXT FROM kurDB INTO @db
END
CLOSE kurDB   DEALLOCATE kurDB
```

Príklady na cvičení:
- Vkladať riadky do tabuľky pomocou kurzoru:
- Vytvoriť tabuľku t1 s tromi riadkami, tabuľku t2 do ktorej prenesiete riadky pomocou kurzora
- Stoličky