

3. Týždeň

Pohľady a CTE WITH

I) VIEW – POHĽAD

II) CTE WITH

1) Úvod

2) Tranzitívny uzáver

3) Syntax rekurzívneho CTE

4) Príklady

a) postupnosti

b) faktoriál

c) OsobaVzťah

5) Ohraničenie iteračného kroku

Vieme, že výsledkom dopytu je tabuľka. Ju môžeme

1. vytlačiť

2. kombinovať s UNION, EXCEPT, INTERSECT alebo s ďalším dopytom

a ako uvidíme

3. uložiť

- trvalo ako VIEW (nie samotnú tabuľku, ale definujúci dopyt)

- dočasne/prechodne ako #, ## tabuľky

4. mať v pamäti ako

- CTE (WITH tabuľka) alebo

- premennú @ tabuľku a znovu použiť v dopytoch

5. sumarizovať (CUBE)

I) VIEW – POHĽAD <https://msdn.microsoft.com/en-us/library/ms190174.aspx>

VIEW slúži na

- rýchle získanie výsledkov komplexných dopytov z (väčšinou) viacerých tabuliek a

- skrytie citlivých údajov v tabuľke.

Po vytvorení VIEW sa naň odvoláva ako na štandardnú tabuľku. VIEW sa vytvára v rámci danej DB. VIEW, jeho kompilovaný definujúci dopyt, sa uloží spolu s DB, je súčasťou DB, podobne ako uložené procedúry alebo indexy.

VIEW je možné vytvoriť, odstrániť, upraviť s

CREATE VIEW

DROP VIEW

ALTER VIEW – nie UPDATE

Syntax vytvorenia VIEW a schematický príklad jeho použitia:

```
CREATE VIEW nazov AS resp. CREATE VIEW nazov(s1, ...,sk) AS  
  SELECT col1, ..., colk FROM ...
```

```
IF OBJECT_ID ('Vnazov', 'V') IS NOT NULL DROP VIEW Vnazov;  
GO  
CREATE VIEW Vnazov AS  
  SELECT ...          -- definujuci dopyt  
  
SELECT * FROM Vnazov  -- pouzitie VIEW
```

Ako vidíme, pri vytvorení VIEW sa zadáva jeho názov a definujúci dopyt, ale stĺpce nie je potrebné. Stĺpce VIEW, ich typ a počet, sú určené s definujúcim dopytom (príkaz SELECT ...).

Ak sa vytvorí VIEW, informácie sú uložené v *sys.views*, *sys.columns*, *sys.sql_dependencies* a text definujúceho dopytu v *sys.sql_modules*.

VIEW je virtuálna tabuľka, ktorá sa vytvorí definujúcim dopytom a sa vykoná neskôr. Preto, ak sa zmení tabuľka, použitá v definujúcom dopyte pohľadu, zmení sa aj výsledok pohľadu. Fyzickou súčasťou DB nie je samotná tabuľka VIEW, ale jeho definujúci dopyt.

Ak VIEW závisí od tabuľky alebo pohľadu, ktoré boli zrušené alebo ich štruktúra bola zmenená, potom pri jeho použití Database Engine generuje chybové hlásenie.

Pohľady spolu s uloženými procedúrami zvyšujú bezpečnosť DB. Ich pomocou sa môžeme vyhnúť, aby konkrétni užívatelia mohli priamo modifikovať DB.

Poznámky

- VIEW reaguje na zmenu stavu tabuľky a schémy.
- VIEW môže mať maximálne 1024 stĺpcov.
- VIEW je relácia, množina riadkov a ORDER BY je dovolené použiť iba spolu s TOP.
- Do VIEW vkladať riadky s INSERT môžeme až po jeho vytvorení s definujúcim dopytom (CREATE VIEW V AS SELECT ...).

```
USE tempdb  
GO
```

```
IF OBJECT_ID ('maz', 'U') IS NOT NULL DROP TABLE maz  
GO  
CREATE TABLE maz(id int, pohlavie char(1), x float)  
GO
```

```

INSERT maz VALUES(1, 'm', 10)
INSERT maz VALUES(2, 'z', 1)
INSERT maz VALUES(3, 'm', 40)
INSERT maz VALUES(4, 'm', 30)
GO
SELECT * FROM maz

```

```

IF OBJECT_ID ('Vmaz', 'V') IS NOT NULL DROP VIEW Vmaz;
GO
CREATE VIEW Vmaz // alebo CREATE VIEW Vmaz(a,b,c)
AS
SELECT TOP 2 id, pohlavie, x FROM maz
      WHERE pohlavie = 'm' order by x asc -- bez TOP NIE
GO
SELECT * FROM Vmaz

```

UPDATE tabuľky a ALTER pohľadu sa odráža vo VIEW

```

SELECT * FROM Vmaz
UPDATE maz SET x = 100.99 where id = 1
SELECT * FROM maz
SELECT * FROM Vmaz
GO

```

```

ALTER VIEW vmaz AS
SELECT id, pohlavie, x FROM maz
      WHERE id<4
GO
SELECT * FROM Vmaz
GO

```

Vkladanie (Insert) do View sa realizuje ako vkladanie do tabuľky:

```

SELECT count(x) FROM maz
INSERT INTO Vmaz VALUES(2, 'm',10)
GO
SELECT count(x) FROM maz
SELECT * FROM Vmaz
SELECT * FROM maz

```

Vkladanie (Insert) do View z dvoch tabuliek:

```

IF OBJECT_ID ('mazPr') IS NOT NULL DROP TABLE mazPr
GO
CREATE TABLE mazPr(id int, nazov varchar(10), idMaz float)
GO
INSERT mazPr VALUES(10, 'b1', 2)
INSERT mazPr VALUES(20, 'b2', 2)
INSERT mazPr VALUES(30, 'cc', 3)
GO

```

```

IF OBJECT_ID ('VmazPr', 'V') IS NOT NULL DROP VIEW VmazPr;

```

```

GO
CREATE VIEW VmazPr
AS
SELECT m.id, pohlavie, nazov FROM maz m JOIN mazPr ON
mazPr.idMaz=m.id
GO
SELECT * FROM mazPr
SELECT * FROM VmazPr

```

Nie je prípustné, aby modifikácia pohľadu ovplyvnil obsah dvoch tabuliek.

```

INSERT VmazPr VALUES(40, 'z', 'dd')
Msg 4405, Level 16, State 1, Line 3
View or function 'VmazPr' is not updatable because the modification affects multiple base tables.

```

II) CTE – WITH <https://msdn.microsoft.com/en-us/library/ms175972.aspx>

- 1) Úvod
- 2) Tranzitívny uzáver
- 3) Syntax rekurzívneho CTE
- 4) Príklady
 - a) postupnosti
 - b) faktoriál
 - c) OsobaVzťah
- 5) Ohraničenie iteračného kroku

1) Úvod

CTE (common table expression) štandardne slúži na prehľadný zápis komplexných dopytov. CTE po deklarácii sa chová ako štandardná tabuľka a je možné sa naň odvolať z doprovodného dopytu viackrát. Na rozdiel od VIEW alebo # a ## tabuliek, CTE existuje podobne @ tabuľky iba **v pamäti** počas vykonania dopytu. Pozor, po GO už nie je prístupný.

Podľa definície MS, CTE je *dočasne pomenovaná množina výsledkov*.

CTE môže sa odvolávať na seba, čo sa využíva pre **rekurziu**.

CTE môžeme použiť v príkazoch SELECT, INSERT, UPDATE, DELETE alebo CREATE VIEW.

Syntax CTE:

```

WITH
cteTab1(s1, ..., sk) AS
(
    SELECT c1, ..., ck FROM tab
)
SELECT ... FROM cteTab1

```

Ako vidíme, názvy stĺpcov môžu byť rozdielne, ale ich **počet** má byť rovnaký. **Typ** stĺpcov **s*** sa zhoduje s typom vrátených stĺpcov **c***. Za složobné slovo **WITH** môžeme definovať viac CTE tabuliek (pozri nižšie príklady).

Príklad.

Nájdite tretiu najmenšiu hodnotu bez použitia Min, porovnávacieho operátora a ROW_NUMBER() – pomocou EXCEPT.

```
-- NO - iba motivacia:  
SELECT TOP(3) x FROM maz  
WHERE x IS NOT NULL  
ORDER BY x ASC  
  
EXCEPT  
SELECT TOP(2) x FROM maz  
WHERE x IS NOT NULL  
ORDER BY x ASC
```

PRED EXCEPT nemôže byť SELECT s ORDER BY

Uvedieme tri riešenia

Obalíme dopyt pred EXCEPT

```
SELECT * FROM (  
SELECT TOP(3) x FROM maz  
WHERE x IS NOT NULL  
ORDER BY x DESC) jaj  
  
EXCEPT  
SELECT TOP(2) x FROM maz  
WHERE x IS NOT NULL  
ORDER BY x DESC
```

Výsledok zatiaľ nie je správny – posledný **ORDER BY** sa vzťahuje na celý kód **SELECT ... EXCEPT SELECT ...**, preto treba obaliť aj druhý dopyt:

```
SELECT * FROM (  
SELECT TOP(3) x FROM maz  
WHERE x IS NOT NULL  
ORDER BY x DESC) jaj  
  
EXCEPT  
SELECT * FROM (  
SELECT TOP(2) x FROM maz  
WHERE x IS NOT NULL  
ORDER BY x DESC) juj
```

2) Riešenie pomocou dvoch CTE tabuliek:

```
WITH  
T3(xx) AS
```

```

(SELECT TOP(3) x FROM maz
 WHERE x IS NOT NULL
 ORDER BY x DESC
),
T2(xx) AS
(
SELECT TOP(2) x FROM maz
 WHERE x IS NOT NULL
 ORDER BY x DESC
)
SELECT * FROM T3
EXCEPT
SELECT * FROM T2

```

3) Riešenie pomocou jednej CTE tabuľky a jedného (povedzme) VIEW:

```

IF OBJECT_ID ('V2', 'V') IS NOT NULL DROP VIEW V2;
GO
CREATE VIEW V2
AS
SELECT TOP(2) x FROM maz
 WHERE x IS NOT NULL
 ORDER BY x DESC
GO
WITH T3(xx) AS
(
SELECT TOP(3) x FROM maz
 WHERE x IS NOT NULL
 ORDER BY x DESC
)
SELECT * FROM T3
EXCEPT
SELECT * FROM V2;

```

2) Tranzitívny uzáver binárnej relácie

Binárna relácia R z množiny A do množiny B (alebo medzi dvomi množinami A, B) je podmnožina Kartezianskeho súčinu $A \times B$, teda je to kolekcia usporiadaných dvojíc prvkov A, B .

Binárna relácia R z A do B sa označuje aj ako aRb . Ak $A=B$, hovoríme, že binárna relácia R je na A .

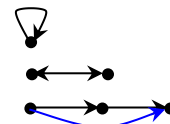
Príklad 1: Nech $X = Y = \{1,2,3\}$. Potom binárna relácia X je väčšie ako Y je zadaná množinou $\{(2,1), (3,1), (3,2)\}$.

Vlastnosti binárnych relácií R na množine A

Uvažujme **binárnu reláciu** $R \subseteq A \times A$, kde A je množina.

Definícia 1. R nazývame

- 1) **reflexívnou**, ak pre $\forall a: a \in A \Rightarrow (a,a) \in R$
- 2) **symetrickou**, ak pre $\forall a,b \in A: (a,b) \in R \Rightarrow (b,a) \in R$
- 3) **tranzitívnou**, ak pre $\forall a,b,c \in A: ((a,b) \in R \wedge (b,c) \in R) \Rightarrow (a,c) \in R$



Príklad 2:

Nech R je relácia na A definovaná takto: $(x,y) \in R \Leftrightarrow x < y$.

R nie je reflexívna: $x < x \dots$ neplatí

R nie je symetrická: $x < y \Rightarrow y < x \dots$ neplatí

R je tranzitívna: $x < y \wedge y < z \Rightarrow x < z \dots$ platí $\forall x,y,z \in A$

Príklad 3:

Nech relácia R je rovnosť na A , teda $(x,y) \in R \Leftrightarrow x = y$.

R je reflexívna: $x = x \dots$ platí $\forall x \in A$

R je symetrická: $x = y \Rightarrow y = x \dots$ platí $\forall x,y \in A$

R je tranzitívna: $x = y \wedge y = z \Rightarrow x = z \dots$ platí $\forall x,y,z \in A$

Ako sme videli, binárna relácia R na A nemusí mať niektorú z vlastností reflexivity, symetrie a tranzitivity. Ale je možné rozšíriť reláciu R , nájsť väčšiu podmnožinu z $A \times A$, ktorá obsahuje R a má požadované vlastnosti. *Binárny uzáver* je najmenšie rozšírenie. Ide o nájdenie minimálneho počtu potrebných dvojíc.

Definícia 2.

Nech R je relácia na množine A , $R \subseteq A \times A$ a $k \in \mathbb{N}$. Hovoríme, že $(x,y) \in R^k$, ak existuje postupnosť prvkov $x = x_0, x_1, \dots, x_{k-1}, x_k = y$ taká, že platí:

$$(x_0, x_1) \in R, (x_1, x_2) \in R, \dots, (x_{k-1}, x_k) \in R.$$

Potom **tranzitívnym uzáverom binárnej relácie** R nazývame reláciu

$$R^+ = R^1 \cup R^2 \cup \dots = \bigcup_{i=1}^{\infty} R^i.$$

Tranzitívny uzáver sa využíva na riešenie úloh o dosiahnuteľnosti.

<p>Príklad 4. $R = \{(1,1), (1,2), (1,3), (2,3), (3,1)\}$ Uzáver R: $(2,3)$ a $(3,1) \Rightarrow (2,1)$ $(2,1)$ a $(1,2) \Rightarrow (2,2)$ $(3,1)$ a $(1,2) \Rightarrow (3,2)$ $(3,1)$ a $(1,3) \Rightarrow (3,3)$ \Rightarrow $R^+ = \{(1,1), (1,2), (1,3), (2,3), (3,1), (2,1), (2,2), (3,2), (3,3)\}$.</p>	<p>Príklad 5.</p> <p>- tranzitívny uzáver:</p>
---	---

Nech X je množina letísk a uvažujme reláciu $x R y$:

existuje priamy let z letiska x do letiska y .

Potom táto relácia nie je tranzitívna. Jej tranzitívny uzáver R^+ je:

do letiska y je možné sa dostať z x cez niekoľko letísk.

Tranzitívnym uzáverom tabuľky *dietaRodic* je potomokPredok (všetky priame a nepriame "rodičovské" vzťahy) – je to výsledok WITH /CTE tabuľky (**pozri nižšie**):

```
WITH ...
SELECT ...
UNION ALL
SELECT ...
```

```
use OsobaVztah
select * from osoba
```

	id	meno	priezvisko	rodne_priezvisko	dat_nar	dat_smrti	pohlavie	vyska	vaha	otec	matka
1	1	Adam	Prvy	NULL	1918-05-11 00:00:00.000	1968-10-01 00:00:00.000	m	180.0	80.0	NULL	NULL
2	2	Eva	Prva	Druha	1919-01-09 00:00:00.000	1988-07-22 00:00:00.000	z	160.0	60.0	NULL	NULL
3	3	Zoly	Mudry	NULL	1918-04-07 00:00:00.000	1990-09-23 00:00:00.000	m	175.5	75.0	NULL	NULL
4	4	NASta	Kovacova	Rostova	1928-02-05 00:00:00.000	1965-03-11 00:00:00.000	z	155.0	99.0	NULL	NULL
5	5	Jozef	Urban	NULL	1922-10-19 00:00:00.000	NULL	m	199.5	NULL	NULL	NULL
6	6	Maria	Urbanova	Novakova	1937-12-08 00:00:00.000	NULL	z	172.5	57.5	1	2
7	7	Patrik	Novak	Novak	1945-06-19 00:00:00.000	NULL	m	182.5	89.5	1	2
8	8	Patricia	Novakova	Haluskova	1952-01-08 00:00:00.000	NULL	z	143.5	35.0	NULL	NULL
9	9	Michal	Kovac	Kovac	1942-04-10 00:00:00.000	NULL	m	167.0	88.0	3	2
10	10	Roman	Kovac	Kovac	1948-05-20 00:00:00.000	NULL	m	179.5	78.5	3	4
11	11	Peter	Horvath	Horvath	1959-07-02 00:00:00.000	2000-12-31 00:00:00.000	m	193.0	110.5	NULL	NULL
12	12	Lucia	Horvathova	Urbanova	1959-01-13 00:00:00.000	NULL	z	156.5	45.5	5	6
13	13	Urban	Urban	Urban	1957-03-31 00:00:00.000	NULL	m	138.2	24.5	5	6
14	14	DASa	Novakova	Novakova	1970-07-17 00:00:00.000	NULL	z	167.0	55.0	7	8
15	15	Viera	Silna	Novakova	1973-02-13 00:00:00.000	NULL	z	169.5	63.0	7	8
16	16	Vladimir	Silny	Silny	1974-08-01 00:00:00.000	2002-12-04 00:00:00.000	m	175.5	73.0	NULL	NULL
17	17	Milena	Slaba	Slaba	1979-09-14 00:00:00.000	NULL	z	164.0	64.0	NULL	NULL
18	18	Jan	Horvath	Horvath	1982-01-16 00:00:00.000	NULL	m	159.5	65.5	11	12
19	19	Zuzana	Silna	Silna	2002-03-01 00:00:00.000	NULL	z	158.5	60.0	16	15
20	20	Zuzana	Slaba	Slaba	1999-12-16 00:00:00.000	NULL	z	171.5	54.5	16	17

Vzťah potomok-predok je možné zobrazit' rôznymi spôsobmi. Uvedieme ženskú aj mužskú vetvu osôb.

id	uplnaCesta
1	Prvy Adam
2	Prva Eva
3	Mudry Zoly
4	Kovacova NASta
5	Urban Jozef
6	Prva Eva \ Urbanova Maria
7	Prva Eva \ Novak Patrik
8	Novakova Patricia
9	Prva Eva \ Kovac Michal
10	Kovacova NASta \ Kovac Roman
11	Horvath Peter
12	Prva Eva \ Urbanova Maria \ Horvathova Lucia
13	Prva Eva \ Urbanova Maria \ Urban Oto
14	Novakova Patricia \ Novakova DASa
15	Novakova Patricia \ Silna Viera
16	Silny Vladimir
17	Slaba Milena
18	Prva Eva \ Urbanova Maria \ Horvathova Lucia \ Horvath Jan
19	Novakova Patricia \ Silna Viera \ Silna Zuzana
20	Slaba Milena \ Slaba Zuzana

Ženská vetva osôb

id	uplnaCesta
1	Prvy Adam
2	Prva Eva
3	Mudry Zoly
4	Kovacova NASta
5	Urban Jozef
6	Prvy Adam \ Urbanova Maria
7	Prvy Adam \ Novak Patrik
8	Novakova Patricia
9	Mudry Zoly \ Kovac Michal
10	Mudry Zoly \ Kovac Roman
11	Horvath Peter
12	Urban Jozef \ Horvathova Lucia
13	Urban Jozef \ Urban Oto
14	Prvy Adam \ Novak Patrik \ Novakova DASa
15	Prvy Adam \ Novak Patrik \ Silna Viera
16	Silny Vladimir
17	Slaba Milena
18	Horvath Peter \ Horvath Jan
19	Silny Vladimir \ Silna Zuzana
20	Silny Vladimir \ Slaba Zuzana

Mužská vetva osôb

Príklad 1. Tranzitívny uzáver rodičovských (priamych a nepriamych) vzťahov (riešenie zatiaľ bez WITH). Najprv vytvoríme tabuľku T1 rodičov (otec, matka) a detí, a z nej postupne tabuľku T2, T3 starých a prastarých rodičov.

Počet riadkov 11+6+2=19.

Osoby s uvedenými rodičmi (otec, matka not null). Pretože osoba môže mať partnera/ku, správnejšie by bolo uviesť namiesto rodičovskej vetvy iba mužskú alebo ženskú vetvu, pozri nižšie riešenie s WITH.

```
use OsobaVztah
IF OBJECT_ID('T1', 'U') IS NOT NULL drop table T1
select Vztah=1, pre.otec, pre.matka, pre.id
    INTO T1 from osoba pre
    where pre.otec is not null or
           pre.matka is not null
select * from T1
```

Pretože 2-ka je matkou 6-ky a 6-ka je matkou 12-ky, potom 2-ka je babkou 12-ky. T2 už bude obsahovať st.rodičov.

Tu v T2 nepremenujeme pre.otec na staryotec ...

```
IF OBJECT_ID('T2', 'U') IS NOT NULL drop table T2
select Vztah=2, pre.otec, pre.matka, pre.id
    INTO T2 from T1 pot JOIN T1 pre
    on pot.otec=pre.id or pot.matka=pre.id
select * from T2
```

Bude otec či matka id-čkom?

```
on T1.otec= T2.id or T1.matka= T2.id
```

Ak pre.id z T2 má pot.matka /st.mamu z T1, potom pre.matka z T2 je prababka pre pot.id z T1:

V Tš nepremenujeme pre.otec na praotec ...

```
IF OBJECT_ID('T3', 'U') IS NOT NULL drop table T3
select Vztah=3, pre.otec, pre.matka, pre.id
    INTO T3 from T1 pot JOIN T2 pre
    on pot.otec=pre.id or pot.matka=pre.id
select * from T3
```

```
IF OBJECT_ID('T4', 'U') IS NOT NULL drop table T4
select Vztah=4, pre.otec, pre.matka, pre.id
    INTO T4 from T1 pot JOIN T3 pre
    on pot.otec=pre.id or pot.matka=pre.id
select * from T4
```

```
select * from t1 union select * from t2 union select * from t3
```

	Vztah	otec	matka	id
1	1	1	2	6
2	1	1	2	7
3	1	3	2	9
4	1	3	4	10
5	1	5	6	12
6	1	5	6	13
7	1	7	8	14
8	1	7	8	15
9	1	11	12	18
10	1	16	15	19
11	1	16	17	20

	Vztah	otec	matka	id
1	2	1	2	12
2	2	1	2	13
3	2	1	2	14
4	2	1	2	15
5	2	5	6	18
6	2	7	8	19

	Vztah	otec	matka	id
1	3	1	2	18
2	3	1	2	19

	Vztah	otec	matka	id
--	-------	------	-------	----

Príklad 2. Faktorial 4!=24 (riešenie bez WITH)

Pripomíname, že na pomenovanie stĺpca sú tri možnosti:

```
SELECT 1      -- do Results
SELECT i = 1
SELECT 1 AS i
SELECT 1 i
PRINT 1      -- do Messages
```

```
drop table T1
select i=1, f=1 INTO T1 -- ⇔ select 1 i, 1 f into T1
select * from T1
```

```
drop table T2
select i=i+1, f=f*(i+1)INTO T2 from T1
select * from T2
```

```
drop table T3
select i=i+1, f=f*(i+1)INTO T3 from T2
select * from T3
```

```
drop table T4
select i=i+1, f=f*(i+1)INTO T4 from T3
select * from T4
```

	i	f
1	1	1
1	2	2
1	3	6
1	4	24

3) Syntax rekurzívneho CTE

Vieme, že rekurzívna funkcia volá seba opakovane, iteračne.

Pre rekurzívne CTE je charakteristické nasledovné:

- po služobnom slove AS vnútri zátvorky () sú dva **SELECT** príkazy, ktoré sú spojené pomocou **UNION ALL**
- prvý **SELECT** sa vykoná iba raz
- v druhom **SELECT**e má byť určená, zakódovaná podmienka zopakovania iteračného kroku (druhého **SELECT**u), kým sa nevráti žiadny riadok
- názvy stĺpcov - (ne)rovnaké
- odvolanie sa na seba uskutočňuje buď pomocou a) stĺpcov alebo b) názvu CTE

Ukážeme príklad na použitie oboch možností odvolanie sa na seba.

4) Príklady

a) Postupnosti

Vytlačte postupnosť čísel od 1 do 6 (každý člen, číslo do nového riadku).

Uvádzame tri riešenia – posledné bude pomocou rekurzíe:

- 1) Do správ pomocou WHILE:

```
DECLARE @c INT;
SET @c = 1;

WHILE @c <= 6
BEGIN
    PRINT @c                -- => tlač ako message
    -- SELECT @c            -- => 6 SELECTOV, tabuliek ako result
    SET @c = @c + 1
END
GO
```

- 2) S použitím pomocnej tabuľky a WHILE:

```
USE tempdb;
GO

IF OBJECT_ID('maz') IS NOT NULL DROP TABLE maz
GO
```

```

CREATE TABLE maz (x int)
GO

DECLARE @c INT;
SET @c = 1;

WHILE @c <= 6
    BEGIN
        INSERT maz VALUES (@c)
        SET @c = @c + 1
    END
SELECT * FROM maz

```

3) Rekurziou na báze CTE WITH:

```

WITH post(k) AS
(
    SELECT i = 1      -- 1 AS i
    UNION ALL
    SELECT j = k + 1 FROM post
      WHERE k < 6
)
SELECT k FROM post;

```

Results	
	k
1	1
2	2
3	3
4	4
5	5
6	6

Prirodzenejší je rovnaký názov stĺpcov:

```

WITH post(k) AS
(
    SELECT k = 1
    UNION ALL
    SELECT k = k + 1
      FROM post WHERE k < 6
)
SELECT k FROM post;

```

b) Faktoriál - C#

```

static int fakt(int n) { ... }

// 1. riešenie ...
int f;
if (n <= 1) f = 1;
else      f = fakt(n - 1) * n;
return f;

// 2. riešenie ...
return ( n <= 1 ) ? 1 : ( fakt(n - 1) * n );

```

b1) Vypočítajte všetky faktoriály 1! až 6!

```
select i=i+1, f=f*(i+1) INTO T2 from T1
```

```

WITH faktorial(i, fakt)
AS
(
    SELECT i = 1, fakt = 1
    UNION ALL
    SELECT i = i + 1, fakt = fakt * (i+1)
      FROM faktorial WHERE i < 6
)
SELECT i, fakt FROM faktorial

```

	i	fakt
1	1	1
2	2	2
3	3	6
4	4	24
5	5	120
6	6	720

b2) Vypočítajte 6! (hard-coding 6)

```
WITH faktorial(i, fakt) AS
(
  SELECT i = 1, fakt = 1
  UNION ALL
  SELECT i = i + 1, fakt = fakt * (i+1)
  FROM faktorial WHERE i < 6
)
SELECT '6! = ' + CAST(fakt AS CHAR(9))
FROM faktorial
WHERE i=6;
```

(No column name)	
1	12! = 479001600

Na cvičení (bez hard-coding):

```
12! = 479001600
20! = 2432902008176640000
33! = 8683317618811886495518194401280000000
     7654321098765432109876543210987654321
```

```
int, real          2,147,483,647 => 12!    4 Bytes
bigint:           9,223,372,036,854,775,807 => 20!    8 Bytes
decimal(38,0):    10^38 - 1 => 33!    17 Bytes
```

c) OsobaVztah

Tabuľku T1 rodičov a detí sme vyššie vytvorili pomocou

```
IF OBJECT_ID('T1', 'U') IS NOT NULL drop table T1
select Vztah=1, pre.otec, pre.matka, pre.id
INTO T1 from osoba pre
where pre.otec is not null or
pre.matka is not null
select * from T1
```

a tabuľku T2 starých rodičov a vnukov pomocou

```
IF OBJECT_ID('T2', 'U') IS NOT NULL drop table T2
select Vztah=2, pre.otec, pre.matka, pot.id
INTO T2 from T1 pot JOIN T1 pre
on pot.otec=pre.id or pot.matka=pre.id
select * from T2
```

Pozrime sa teraz na riešenie s použitím WITH (zatiaľ bez rekurzie).

```
SELECT id, matka, otec FROM osoba -- 20
GO
WITH dietaRodic(idD, idR) AS -- T1
(
  SELECT d.id idD, r.id idR
  FROM Osoba d
  JOIN Osoba r ON r.id IN (d.matka, d.otec)
)
-- 1) dieta - rodic
--SELECT * FROM dietaRodic -- T1 x 2 = 22
-- 2) vnuk - stRodic
```

```

SELECT o1.idD vnuk, o2.idR stRod    -- T2 x 2 = 12
FROM dietaRodic o1
     JOIN dietaRodic o2 ON o1.idR = o2.idD;

```

Vypíšme všetky päťice

[idPot, potomokCelMeno, idPred, predokCelMeno, vzdialenosť]

Teda päťice osôb takých, že

druhá je predkom prvej (⇔ prvá je potomkom druhej),

kde potomok môže byť dieťa, vnuk/čka, pravnuk/čka atď. a predok rodič, starý rodič alebo prarodič.

```
USE OsobaVztah;
```

```
GO
```

```
WITH potomokPredok(idPot, idPre, vzdial) AS
```

```

(
  SELECT d.id , r.id , 1
     FROM Osoba d
     JOIN Osoba r ON r.id IN (d.matka, d.otec)
  UNION ALL
  SELECT d.id, r.idPre, vzdial + 1
     FROM Osoba d
     JOIN potomokPredok r ON r.idPot IN (d.matka, d.otec)
)

```

```
--SELECT * FROM potomokPredok WHERE vzdial=3 --iba id-cka
```

```

SELECT idPot, o1.priezvisko + ' ' + o1.meno Pot, idPre,
       o2.priezvisko + ' ' + o2.meno Pred, vzdial

```

```
FROM potomokPredok p
```

```
JOIN Osoba o1 ON o1.id = p.idPot
```

```
JOIN Osoba o2 ON o2.id = p.idPre
```

```
ORDER BY vzdial, 1, 3, 2, 4;
```

	idPot	Pot	idPre	Pred	vzdial
1	6	Urbanova Maria	1	Prvy Adam	1
2	6	Urbanova Maria	2	Prva Eva	1
3	7	Novak Patrik	1	Prvy Adam	1
...					
33	19	Silna Zuzana	7	Novak Patrik	2
34	19	Silna Zuzana	8	Novakova Patricia	2
35	18	Horvath Jan	1	Prvy Adam	3
36	18	Horvath Jan	2	Prva Eva	3
37	19	Silna Zuzana	1	Prvy Adam	3
38	19	Silna Zuzana	2	Prva Eva	3

Počet riadkov je dvakrát 11+6+2=19.

Pre každého človeka nájdite vetvu jeho mužských/ženských predkov podľa vyššieho vzoru.

```
WITH genCesta (id, cesta) AS
(
    SELECT id, CAST( '' AS VARCHAR(90) )
        --FROM Osoba WHERE otec IS NULL
        FROM Osoba WHERE matka IS NULL
    UNION ALL
    SELECT o1.id,
        CAST( g.cesta + o2.priezvisko + ' ' +
            o2.meno+ ' \ ' AS VARCHAR(90) )
    FROM Osoba o1, Osoba o2, genCesta g
        --WHERE g.id = o1.otec AND g.id = o2.id
    WHERE g.id = o1.matka AND g.id = o2.id
)
SELECT o.id, c.cesta + o.priezvisko + ' ' +
    o.meno uplnaCesta
FROM Osoba o, genCesta c
WHERE o.id = c.id ORDER BY 1 -- => 20
```

id	uplnaCesta
1	Prvy Adam
2	Prva Eva
3	Mudy Zoly
4	Kovacova NASTa
5	Urban Jozef
6	Prvy Adam \ Urbanova Maria
7	Prvy Adam \ Novak Patrik
8	Novakova Patricia
9	Mudy Zoly \ Kovac Michal
10	Mudy Zoly \ Kovac Roman
11	Horvath Peter
12	Urban Jozef \ Horvathova Lucia
13	Urban Jozef \ Urban Oto
14	Prvy Adam \ Novak Patrik \ Novakova DASA
15	Prvy Adam \ Novak Patrik \ Silna Viera
16	Silny Vladimir
17	Slaba Milena
18	Horvath Peter \ Horvath Jan
19	Silny Vladimir \ Silna Zuzana
20	Silny Vladimir \ Slaba Zuzana

id	uplnaCesta
1	Prvy Adam
2	Prva Eva
3	Mudy Zoly
4	Kovacova NASTa
5	Urban Jozef
6	Prva Eva \ Urbanova Maria
7	Prva Eva \ Novak Patrik
8	Novakova Patricia
9	Prva Eva \ Kovac Michal
10	Kovacova NASTa \ Kovac Roman
11	Horvath Peter
12	Prva Eva \ Urbanova Maria \ Horvathova Lucia
13	Prva Eva \ Urbanova Maria \ Urban Oto
14	Novakova Patricia \ Novakova DASA
15	Novakova Patricia \ Silna Viera
16	Silny Vladimir
17	Slaba Milena
18	Prva Eva \ Urbanova Maria \ Horvathova Lucia \ Horvath Jan
19	Novakova Patricia \ Silna Viera \ Silna Zuzana
20	Slaba Milena \ Slaba Zuzana

5) Ohraničenie iteračného kroku

OPTION (MAXRECURSION 100)
 - v rámci samoštúdia (help).

```

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;

```