1) VIEW – POHĽAD
2) CTE WITH - I

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)

1) VIEW – POHĽAD

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.

Syntax vytvorenia VIEW a príklad jeho použitia:

```
CREATE VIEW nazov(st1, ..., stk) 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. 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 a neskoršie vykoná definujúcim dopytom. 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, Database Engine generuje chybové hlásenie.

Pohľady spolu s uloženými procedúrami zvyšujú bezpečnosť DB. Ich pomocou sa môžeme vyhnúť, aby 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') 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
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

```
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

-- Zmeny v schéme + ďalšie =>cvičenie!
```

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 VmazPr

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.
```

2) CTE - WITH

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 nového 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 st* sa zhoduje s typom vrátených stĺpcov col*.

Príklad.
Najdite tretiu najmensiu hodnotu bez použitia Min, porovnavacieho operatora 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 a ani za nemoze byt SELECT s ORDER BY

-- NO ???

```
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
```

Tri riešenia:

1)

```
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 tabulky a jedneho(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;
```