Poradové a agregačné *window* funkcie _____

1) Poradové, agregačné a analytické *window* funkcie
2) Extrémy pomocou DENSE_RANK(), TOP() - Príklady
3) Spriemernené poradia
4) Kumulatívne súčty

Window (windowing or windowed) funkcie (WF) vykonávajú výpočet na množine riadkov a vrátia **jednu** agregovanú hodnotu **pre každý riadok.**

WF sú súčasťou štandardov ANSI SQL 2003. Agregačné a rangové (ranking functions) WF boli zavedené do jazyka T-SQL v roku 2005, ešte ďalšie (ORDER BY v klauzule OVER, LAG, LEAD, FIRST_VALUE a LAST_VALUE, frame a analytické funkcie) v roku 2012, kým GROUPING, GROUPING_ID a APPROX_COUNT_DISTINCT v 2019.

1) Poradové, agregačné a analytické *window* funkcie

Okno/window je množina riadkov, pre ktorú sa aplikujú window funkcie v SELECT zozmame, ako napr ROW_NUMBER, Sum.

Klauzula **OVER** určuje, ktoré riadky tvoria okno. Klauzula OVER má tri možné komponenty: PARTITION BY, ORDER BY a FRAME:
- výraz PARTITION BY rozdeľuje riadky do segmentov/okien
- výraz ORDER BY je voliteľný, je však potrebný pre niektoré typy funkcií okna, a určuje poradie, v akom sa aplikuje WF.
- rám sa používa pre niektoré špecifické typy funkcií okna.

Syntax:

```
window_function()Over(Partition by ...)
window_function()Over(Order by ...)
window_function()Over(Partition by C1 Order by C2)
```

```
OVER (
        [ <PARTITION BY clause> ]
        [ <ORDER BY clause> ]
        [ <ROW or RANGE clause> ]
    )
```

```
OVER( PARTITION BY krajID
      ORDER BY DATEPART(yy, beginDate)
      ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING )
```

PARTITION BY rozdelí výsledok do segmentov. Window funkcie sú aplikované zvlášť pre každý segment (výpočty sa začínajú odznova pre každý segment).

Window funkcia môže byť _____
a) poradová funkcia
b) agregačná funkcia
c) analitická funkcia
d) NEXT VALUE FOR funkcia.

Pred použitím *window* funkcie, ktorá vypočíta hodnotu pre každý riadok v okne, **Over** klauzula určuje _____
- segmentovanie / partition a
- zoradenie / order riadkov.

Treba poznamenať, že WF nie je dovolené priamo použiť vo WHERE klauzule (riešenie napr. CTE).

Nižšie sa vyjadríme k jednotlivým typom okrem posledného d).

1a) PrWF() OVER (ORDER BY col1)

Poradové/ranking window funkcie vrátia pre každý riadok segmentu poradie/rank. Funkcie, určujúce poradie:

- ROW_NUMBER – číselník=jednoduché očíslovanie vrátených riadkov/záznamov
- RANK – poradie
- DENSE_RANK – stlačené poradie
- NTILE – nČastí http://www.sqljunkies.com/WebLog/sqlbi/archive/2006/04/19/20482.aspx

Spriemernené poradie – nie je podporované.

Syntax Over pre poradové window funkcie:

```
PrWF() OVER ( [ PARTITION BY value_expression , ... [ n ] ]
            <ORDER BY_Clause> )
```

```
Use Poliklinika
SELECT      ROW_NUMBER()      OVER (ORDER BY poplatok) AS ciselnik,
            RANK()            OVER (ORDER BY poplatok) AS poradie,
            poplatok,
            DENSE_RANK()      OVER (ORDER BY poplatok) AS stlcPor,
            NTILE(2)          OVER (ORDER BY poplatok) AS nCasti
      FROM  Navstevy
      WHERE --poplatok IS NOT NULL AND
            poplatok IN(300, 500)
      ORDER BY poplatok

-- segmentovanie + zoradenie
select ROW_NUMBER() OVER (PARTITION BY idL
                  ORDER BY poplatok) AS ciselnik,
 idp,idL from navstevy
```

	ciselnik	poradie	poplatok	stlcPor	nCasti
1	1	1	300	1	1
2	2	1	300	1	1
3	3	3	500	2	1
4	4	3	500	2	2
5	5	3	500	2	2

Jednou výhodou funkcie ROW_NUMBER je schopnosť meniť nejedinečné riadky na jedinečné riadky, čo sa dá využiť na odstránenie duplicitných riadkov:

```
DROP TABLE if EXISTS #TabD
CREATE TABLE #TabD(cislo INT, pismo CHAR(1));
INSERT INTO #TabD(cislo, pismo)
VALUES(1,'x'),
      (2,'y'),(2,'y'),
      (3,'z'),(3,'z');
SELECT * FROM #TabD;

SELECT cislo, pismo,
   ROW_NUMBER() OVER(PARTITION BY cislo, pismo ORDER BY cislo) AS rNb
FROM #TabD;

WITH Haha AS (
   SELECT cislo, pismo,
     ROW_NUMBER() OVER(PARTITION BY cislo, pismo ORDER BY cislo) AS rNb
   FROM #TabD
)
DELETE Haha
WHERE rNb <> 1;
SELECT * FROM #TabD;
```

	cislo	pismo	rNb
1	1	x	1
2	2	y	1
3	2	y	2
4	3	z	1
5	3	z	2

	cislo	pismo
1	1	x
2	2	y
3	3	z

1b) AgWF() OVER (PARTITION BY col1)

Agregačná window funkcia priraďuje každému segmentu zodpovedajúcu hodnotu, ktorú zapíše do každého riadku zodpovedalúceho segmentu.

Agregačné window funkcie:
COUNT, SUM, AVG, MAX, MIN, ...
Syntax Over pre agregačné window funkcie:
AWF() OVER ([PARTITION BY value_expression , ... [n]])

Pridaním klauzuly OVER k agregačnej funkcii je možné sa vyhnúť pravidlám, ktoré klasicky treba dodržiavať.

```
use poliklinika
--1
SELECT poplatok, COUNT(poplatok)
FROM Navstevy
WHERE poplatok IN(300, 500)
GROUP BY poplatok
ORDER BY poplatok

--2 <=> namiesto GROUP BY poplatok pisat OVER(PARTITION BY poplatok) v Selecte

SELECT DISTINCT poplatok, COUNT(poplatok)OVER(PARTITION BY poplatok)AS pocet
FROM Navstevy
WHERE poplatok IN(300, 500)
ORDER BY poplatok
```

	poplatok	pocet
1	300	2
2	500	3

V posledných dvoch dopytoch dopíšte do SELECTu aj den. Vysvetlite výsledok!

```
--3 SQL SERVER 2014 - OK
SELECT DISTINCT poplatok, COUNT(poplatok)OVER(PARTITION BY poplatok ORDER BY
poplatok)poc
FROM Navstevy
WHERE poplatok IN(300, 500)

SELECT DISTINCT
           poplatok,
           COUNT(poplatok)OVER(PARTITION BY poplatok)AS pocet,
           SUM(poplatok)  OVER(PARTITION BY poplatok) AS suma,
           AVG(CAST(poplatok AS FLOAT))
                          OVER(PARTITION BY poplatok) AS priemer,
           MAX(poplatok)  OVER(PARTITION BY poplatok) AS maxi
FROM Navstevy
WHERE poplatok IN(300, 500)
ORDER BY poplatok
```

	poplatok	pocet	suma	priemer	maxi
1	300	2	600	300	300
2	500	3	1500	500	500

1c) AnWF() OVER (PARTITION BY col1)

Analitycká window funkcia na rozdiel od agregačnej funkcie môže vrátiť viac riadkov pre jednotlivé skupiny. AnWF môže byť využitá napr. na výpočet kĺzavých priemerov.
Analytické window funkcie: https://docs.microsoft.com/en-us/sql/t-sql/functions/analytic-functions-transact-sql
CUME_DIST, FIRST_VALUE, LAG, LAST_VALUE, LEAD, ...

Použitie CUME_DIST a FIRST_VALUE ukážeme v časti 4 a LAG na cvičení.

2) Extrémy pomocou DENSE_RANK(), TOP() - Príklady

```
-- F1) Vrátte prvých troch najmladších - ak sú distinct:
SELECT TOP(3) L.datnar FROM Lekari L
     ORDER BY L.datNar DESC
```

	datnar
1	1980-02-15 00:00:00.000
2	1970-04-02 00:00:00.000
3	1961-11-14 00:00:00.000

```
-- F2) Usporiadajte lekarov podla veku zostupne a
--     vratte aj ciselnik ~ poradie.
SELECT krstne, datNar
        , ROW_NUMBER() OVER(ORDER BY datnar DESC) AS ciselnik
        , DENSE_RANK() OVER(ORDER BY datnar DESC) AS poradie
     FROM Lekari
```

	krstne	datNar	ciselnik	poradie
1	Klara	1980-02-15 00:00:00.000	1	1
2	Zuzka	1970-04-02 00:00:00.000	2	2
3	Zoli	1961-11-14 00:00:00.000	3	3
4	Oto	1960-05-05 00:00:00.000	4	4
5	Imro	1956-11-09 00:00:00.000	5	5

```
-- F3) Najdite udaje o tretom/tej najmladsom/ej lekarovi/ke:
SELECT L.krstne, L.spec, L.datNar FROM
          (SELECT krstne, spec, datNar
              ,DENSE_RANK() OVER(ORDER BY datnar DESC) AS poradie
           FROM Lekari  -- 1980,1970,1961,1960,1956
          ) L
     WHERE L.poradie = 3 --WHERE T.poradie BETWEEN 3 AND 3
```

	krstne	spec	datNar
1	Zoli	Zubny	1961-11-14 00:00:00.000

3) Spriemernené poradia

Už poznáme window funkcie určujúce poradie/RANK:
- ROW_NUMBER - číselník=jednoduché očíslovanie vrátených riadkov/záznamov
- RANK - poradie
- DENSE_RANK - stlačené poradie
- NTILE - nČastí

ale chýba **spriemernené** poradie.

Príklad: Usporiadajte poplatky 300 a 500 z tabuľky Návštevy vzostupne, uvedte aj číslo riadkov (číselník) a vypočítajte spriemernené poradia.

```
-- Vnútorný VD vráti číselník a vonkajší dodá priemer segmentov/okien.
USE Poliklinika;
```

```
GO

SELECT poplatok, ciselnik, AVG(CAST(ciselnik AS FLOAT))
                        OVER(PARTITION BY poplatok) AS sprPoradie
FROM
(
      SELECT      poplatok,
                  ROW_NUMBER() OVER (ORDER BY poplatok) AS ciselnik
      FROM  Navstevy
      WHERE poplatok IN( 300, 500 )
) TT
```

	poplatok	ciselnik	sprPoradie
1	300	1	1,5
2	300	2	1,5
3	500	3	4
4	500	4	4
5	500	5	4

Group By:

```
SELECT poplatok, -- ciselnik,
      AVG(CAST(ciselnik AS FLOAT)) AS sprPoradie
FROM
(
      SELECT poplatok,
                  ROW_NUMBER() OVER (ORDER BY poplatok) AS ciselnik
      FROM  Navstevy
      WHERE poplatok IN( 300, 500 )
) TT
Group BY poplatok
```

	poplatok	sprPoradie
1	300	1.5
2	500	4

4) Kumulatívne súčty

Postup výpočtu kumulatívneho súčtu.

Tabuľka Sucet so stĺpcom x:

```
USE tempdb
create table Sucet(x int)
insert Sucet values(10)
insert Sucet values(20)
insert Sucet values(30)
insert Sucet values(40)
```

x	T1.x	T2.x	Sucty
10	10	10	10
20	20	10	
30	20	20	30
40	30	10	
	30	20	
	30	30	60
	40	10	
	40	20	
	40	30	
	40	40	100

```
-- OK:
SELECT T1.x T1x, T2.x kumS1,
      T1.x * CUME_DIST () OVER (PARTITION BY T1.x ORDER BY T2.x) AS kumS2,
            CUME_DIST () OVER (PARTITION BY T1.x ORDER BY T2.x) AS kumS
      FROM Sucet T1 CROSS JOIN Sucet T2
 --WHERE T1.x <= T2.x  -- NO
   WHERE T2.x <= T1.x
   ORDER BY T1x;
```

	T1x	kumS1	kumS2	kumS
1	10	10	10	1
2	20	10	10	0.5
3	20	20	20	1
4	30	10	10	0.333333333333333
5	30	20	20	0.666666666666667
6	30	30	30	1
7	40	10	10	0.25
8	40	20	20	0.5
9	40	30	30	0.75
10	40	40	40	1

```
-- pokracovanie:
  SELECT T1.x T1x, SUM(T2.x) sucty
      FROM Sucet T1 CROSS JOIN Sucet T2
      WHERE T2.x <= T1.x
      GROUP BY T1.x
  ORDER BY T1x;
```

	T1x	T2x
1	10	10
2	20	30
3	30	60
4	40	100

```
---- pokracovanie:
SELECT T1x, T.sucty FROM
(SELECT T1.x T1x, SUM(T2.x) sucty
          FROM Sucet T1 JOIN Sucet T2 ON T2.x <= T1.x
          GROUP BY T1.x
) T
WHERE T1x=40;
```

	T1x	sucty
1	40	100

```sql
-- V ktorom mesiaci zacali ordinovat jednotlivi lekari
-- v novej poliklinike
use poliklinika
SELECT  idL, den, month(den) mes,
  FIRST_VALUE(den) OVER (PARTITION BY idL order by idL,den) [1.v],
  month(FIRST_VALUE(den) OVER (PARTITION BY idL order by idL,den)) mes1v
FROM  navstevy
```

idL	den	mes	1.v	mes1v
1	2008-06-05 00:00:00.000	6	2008-06-05 00:00:00.000	6
1	2008-06-05 00:00:00.000	6	2008-06-05 00:00:00.000	6
1	2008-06-05 00:00:00.000	6	2008-06-05 00:00:00.000	6
1	2008-07-05 00:00:00.000	7	2008-06-05 00:00:00.000	6
1	2008-09-05 00:00:00.000	9	2008-06-05 00:00:00.000	6
1	2008-09-05 00:00:00.000	9	2008-06-05 00:00:00.000	6
1	2008-09-05 00:00:00.000	9	2008-06-05 00:00:00.000	6
1	2008-10-05 00:00:00.000	10	2008-06-05 00:00:00.000	6
1	2008-11-05 00:00:00.000	11	2008-06-05 00:00:00.000	6
2	2008-08-05 00:00:00.000	8	2008-08-05 00:00:00.000	8
2	2008-08-05 00:00:00.000	8	2008-08-05 00:00:00.000	8
2	2008-05-05 00:00:00.000	5	2008-08-05 00:00:00.000	8
3	2008-05-05 00:00:00.000	5	2008-05-05 00:00:00.000	5
3	2008-05-05 00:00:00.000	5	2008-05-05 00:00:00.000	5
3	2008-07-05 00:00:00.000	7	2008-05-05 00:00:00.000	5
3	2008-07-05 00:00:00.000	7	2008-05-05 00:00:00.000	5
4	2008-06-05 00:00:00.000	6	2008-06-05 00:00:00.000	6
4	2008-08-05 00:00:00.000	8	2008-06-05 00:00:00.000	6
4	2008-10-05 00:00:00.000	10	2008-06-05 00:00:00.000	6
5	2008-10-05 00:00:00.000	10	2008-10-05 00:00:00.000	10
5	2008-09-05 00:00:00.000	9	2008-10-05 00:00:00.000	10
5	2008-11-05 00:00:00.000	11	2008-10-05 00:00:00.000	10