

Poradové a agregáčn  window funkcie _____

- 1) Poradov , agregáčn  a analytick  window funkcie
- 2) Extr my pomocou `DENSE_RANK()`, `TOP()` - Pr klady
- 3) Spriemernen  poradia
- 4) Kumulat vne s čty

Window (windowing or windowed) funkcie (WF) vykonávajú v počet na mnoŹine riadkov a vr tia **jednu** agregovan  hodnotu **pre kaŹd  riadok**.

WF s  s časťou štandardov ANSI SQL 2003. Agregáčn  a rangov  (ranking functions) WF boli zaveden  do jazyka T-SQL v roku 2005, eŹte ďalšie (ORDER BY v klauzule OVER, LAG, LEAD, FIRST_VALUE a LAST_VALUE, frame a analytick  funkcie) v roku 2012, k m GROUPING, GROUPING_ID a APPROX_COUNT_DISTINCT v 2019.

1) Poradov , agregáčn  a analytick  window funkcie

Okno/window je mnoŹina riadkov, pre ktor  sa aplikuj  window funkcie v SELECT zozname, ako napr `ROW_NUMBER`, `Sum`.

Klauzula **OVER** určuje, ktor  riadky tvoria okno. Klauzula OVER m  tri moŹn  komponenty: PARTITION BY, ORDER BY a FRAME:

- v raz PARTITION BY rozdeľuje riadky do segmentov/okien
- v raz ORDER BY je voliteľn , je vŹak potrebn  pre niektor  typy funkci  okna; určuje poradie, v akom sa aplikuje WF.
- r m sa pouŹiva pre niektor  Źpecifick  typy funkci  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 rozdel  v sledok do segmentov. Window funkcie s  aplikovan  zvl Źt pre kaŹd  segment (v počty sa začínaj  odznova pre kaŹd  segment).

Window funkcia m Źe byť _____

- a) poradov  funkcia
- b) agregáčn  funkcia
- c) analitick  funkcia
- d) NEXT VALUE FOR funkcia.

Pred pouŹit m window funkcie, ktor  vypoč ta hodnotu pre kaŹd  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 <https://www.sqljunkies.com/Weblog/sqlbi/2006/06/19/20487.asp>

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 priraduje každému segmentu zodpovedajúcu hodnotu, ktorú zapíše do každého riadku zodpovedajúceho 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 dopíšete 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)
```

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

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

Analýtická window funkcia na rozdiel od agregačnej funkcie môže vrátiť viac riadkov pre jednotlivé skupiny. AnWF môže byť využitá napr. na výpočet kľzavých 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) Vráťte 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 podľa veku zostupne a
-- vráťte aj číselník ~ 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	Klára	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 údaje 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.

-- Vnútrotný 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
```

	poplatok	sprPoradie
1	300	1,5
2	500	4

Group BY poplatok

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	0.3333333333333333
5	30	20	20	0.6666666666666667
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

-- pokračovanie:

```
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

---- pokračovanie:

```
SELECT T1x, T.sucty FROM
(SELECT T1.x T1x, SUM(T2.x) sucty
```

	T1x	sucty
1	40	100

```
FROM Sucet T1 JOIN Sucet T2 ON T2.x <= T1.x
GROUP BY T1.x
```

```
) T
WHERE T1x=40;
```

```
-- V ktorom mesiaci zacali ordinovat jednotlivi lekari
-- (v novej poliklinike)
```

```
SELECT idL, CAST(den as date) 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)) m1,
-- <=>
min(month(den)) OVER (PARTITION BY idL) AS m2
FROM navstevy
```

	idL	den_	mes	1.v	m1	m2
1	1	2008-06-05	6	2008-06-05 00:00:00.000	6	6
2	1	2008-06-05	6	2008-06-05 00:00:00.000	6	6
3	1	2008-06-05	6	2008-06-05 00:00:00.000	6	6
4	1	2008-07-05	7	2008-06-05 00:00:00.000	6	6
5	1	2008-09-05	9	2008-06-05 00:00:00.000	6	6
6	1	2008-09-05	9	2008-06-05 00:00:00.000	6	6
7	1	2008-09-05	9	2008-06-05 00:00:00.000	6	6
8	1	2008-10-05	10	2008-06-05 00:00:00.000	6	6
9	1	2008-11-05	11	2008-06-05 00:00:00.000	6	6
10	2	2008-05-05	5	2008-05-05 00:00:00.000	5	5
11	2	2008-08-05	8	2008-05-05 00:00:00.000	5	5
12	2	2008-08-05	8	2008-05-05 00:00:00.000	5	5
13	3	2008-05-05	5	2008-05-05 00:00:00.000	5	5
14	3	2008-05-05	5	2008-05-05 00:00:00.000	5	5
15	3	2008-07-05	7	2008-05-05 00:00:00.000	5	5
16	3	2008-07-05	7	2008-05-05 00:00:00.000	5	5
17	4	2008-06-05	6	2008-06-05 00:00:00.000	6	6
18	4	2008-08-05	8	2008-06-05 00:00:00.000	6	6
19	4	2008-10-05	10	2008-06-05 00:00:00.000	6	6
20	5	2008-09-05	9	2008-09-05 00:00:00.000	9	9
21	5	2008-10-05	10	2008-09-05 00:00:00.000	9	9
22	5	2008-11-05	11	2008-09-05 00:00:00.000	9	9