Poradové a agregačné *window* funkcie. ROLLUP a CUBE

1) Poradové a agregačné *window* funkcie

Okno/window je množina riadkov, pre ktorú sa aplikujú window funkcie v SELECT zozname.
Over klauzula určuje http://technet.microsoft.com/en-us/library/ms189461.aspx
- segmentovanie / partition a
- zoradenie / order riadkov

pred použitím zodpovedajúcej *window* funkcie.

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).

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> )
```

```
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

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

Agregačná window funkcia priraďuje každému segmentu zodpovedajúcu hodnotu.

Agregačné window funkcie:

COUNT, SUM, AVG, MAX, MIN

Syntax Over pre agregačné window funkcie:

```
AWF() OVER ( [ PARTITION BY value_expression , ... [ n ] ] )
```

```
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

```
--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) **TOP(n)** alebo **TOP n** vráti z výsledku dopytu prvých n riadkov.

ISO SQL 2003 Standard

```
SELECT  TOP   3 * from T          -- MS SQL Server, alebo TOP(3)
SELECT  FIRST 3 * from T          -- Ingres
SELECT * FROM T LIMIT 3           -- MySQL
SELECT * from T WHERE ROWNUM <= 3 -- Oracle
```

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

```
-- F1) Vrátte prvýh 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 T2x
    FROM Sucet T1 CROSS JOIN Sucet T2
 -- WHERE T1.x <= T2.x  -- NO
    WHERE T2.x <= T1.x
ORDER BY T1x;
```

	T1x	T2x
1	10	10
2	20	10
3	20	20
4	30	10
5	30	20
6	30	30
7	40	10
8	40	20
9	40	30
10	40	40

```
-- 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

	T1x	sucty
1	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;
```

5) Group By a Datepart, With RollUp a With Cube

Mesačné sumárne poplatky v druhom polroku

```
USE Poliklinika;
GO

SELECT DATEPART(mm, n.den) mes, SUM(n.poplatok) suma
      FROM Navstevy n
      GROUP BY DATEPART(mm, n.den)
      HAVING   Month(n.den) >= 7
      ORDER BY suma
      --ORDER BY DATEPART(mm, n.den)
```

	mes	suma
1	9	400
2	11	750
3	7	1200
4	10	1400
5	8	2000

ROLLUP a **CUBE** sa využívajú pri vypočítavaní sumárnych veličín.
- ROLLUP generuje agregačné hodnoty pre hierarchické hodnoty vo vybraných stĺpcoch
- CUBE generuje agregačné hodnoty pre všetky kombinacie hodnôt vo vybraných stĺpcoch.

```
---Sumárne poplatky za 200,500 a 800 u jednotliv.špec
SELECT L.Spec, N.poplatok, sum(N.Poplatok) suma
            FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
            Where N.Poplatok IN(500,200,800)

            Group by spec, poplatok
```

	Spec	poplatok	suma
1	Ocny	200	1000
2	Ocny	500	500
3	Zubny	500	1000
4	Zubny	800	1600

```
SELECT L.Spec, N.poplatok, sum(N.Poplatok) suma
            FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
            Where N.Poplatok IN(500,200,800)
            Group by spec, poplatok
With Rollup
```

	Spec	poplatok	suma
1	Ocny	200	1000
2	Ocny	500	500
3	Ocny	NULL	1500
4	Zubny	500	1000
5	Zubny	800	1600
6	Zubny	NULL	2600
7	NULL	NULL	4100

```
SELECT CASE WHEN L.Spec IS NULL THEN 'ZVsetci'
        Else L.Spec End Spec,
       CASE WHEN N.poplatok IS NULL THEN 'SumPop'
        Else cast(N.poplatok as Varchar(10)) End Popl,
       sum(N.Poplatok) Suma
   FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
Where N.Poplatok IN(500,200,800)
Group by spec, poplatok
With Rollup
```

	Spec	Popl	Suma
1	Ocny	200	1000
2	Ocny	500	500
3	Ocny	SumPop	1500
4	Zubny	500	1000
5	Zubny	800	1600
6	Zubny	SumPop	2600
7	ZVsetci	SumPop	4100

```
SELECT CASE WHEN L.Spec IS NULL THEN 'ZVsetci'
        Else L.Spec End Spec,
       CASE WHEN N.poplatok IS NULL THEN 'SumPop'
        Else cast(N.poplatok as Varchar(10)) End Popl,
       sum(N.Poplatok) Suma
   FROM Lekari L JOIN Navstevy N ON L.idL = N.idL
Where N.Poplatok IN(500,200,800)
Group by spec, poplatok
With cube Order By Spec, Poplatok Desc
```

	Spec	Popl	Suma
1	Ocny	500	500
2	Ocny	200	1000
3	Ocny	SumPop	1500
4	Zubny	800	1600
5	Zubny	500	1000
6	Zubny	SumPop	2600
7	ZVsetci	800	1600
8	ZVsetci	500	1500
9	ZVsetci	200	1000
10	ZVsetci	SumPop	4100

Kontingenčné alebo pivot tabuľky ponúkajú viac možností.