13. FEJEZET

Részletező csoportosítások és analitikus függvények az SQL-ben

Elméleti összefoglaló

Gyakran van szükség különböző részletezettségű csoportosítások szerinti tulajdonságok lekérdezésére, „mi lenne, ha..." típusú kérdések megválaszolására, vagy rangsorok előállítására. E problémákat *elemző jellegűeknek* nevezhetnénk, nem csupán az elemzőfüggvények miatt, hanem a vizsgált adathalmaz belső szerkezetét kifinomult módon firtató jellegük miatt is. E fejezetben tehát ilyen elemző feladatokra mutatunk be eszközöket.

Miután ez a terület a magyar nyelvű szakirodalomban újdonságnak számít, ezért az elméleti összefoglalóban sok és részletes példát mutatunk be. Ezekhez hasonló feladatsort viszont már nem lett volna célszerű készíteni. Hasznosabbnak gondoltuk, hogy inkább egy „valódi" vállalati környezetben, a következő fejezetben bemutatásra kerülő DEMO vállalatra vonatkozóan fogalmazunk meg olyan feladatokat, melyek megoldásához e fejezet eszközeinek hatékonysága nyilvánvalóvá válik.

Az SQL új csoportképzési eszközei

A ROLLUP operátor

A ROLLUP operátor segítségével csoportfüggvényeket alkalmazhatunk oszlopok sorozatán egyre finomodó részcsoportképzéssel. A ROLLUP operátort a GROUP BY utasításrészben használjuk a következő szintaktika szerint:

```
GROUP BY ROLLUP (oszlop1 [, oszlop2]...)
```

A szelekciós listában kijelölt csoportműveletet a ROLLUP kulcsszó utáni, zárójelben lévő oszlopok szerint végzi a megadott oszlopsorrend szerint jobbról balra (*lásd* az alábbi példát). Az ezen oszlopokban (nem eredeti táblaadatként) NULL értéket tartalmazó sorok adják a csoportműveletet részértékeit, illetve a végértékét a csoportművelet oszlopában. E megjelenítés eredményeként a függőleges irányban eddig homogén (azonos jelentésű) listák, a ROLLUP operátor következtében ebben az irányban is strukturálttá váltak. (A csoportosítások esetén felmerülő problémákkal kapcsolatban *lásd* 14.8. feladat.)

13.1. példa

Listázzuk ki részlegenként, és ezen belül foglalkozásonként az összfizetéseket, továbbá a részlegenkénti összfizetéseket, végül a vállalati összfizetést, és ugyanezen csoportosításban a létszámokat is.

1. megoldás (Egyszerű csoportosítással)

```
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
  GROUP BY deptno, job;

SELECT deptno, SUM(sal), COUNT(*) AS Létszám
  FROM emp
  GROUP BY deptno;

SELECT SUM(sal), COUNT(*) AS Létszám
  FROM emp;
```

Eredmény

```
    DEPTNO JOB          SUM(SAL)    LÉTSZÁM
---------- --------- ---------- ----------
        10 CLERK           1300          1
        10 MANAGER         2450          1
        10 PRESIDENT       5000          1
        20 CLERK           1900          2
        20 ANALYST         6000          2
        20 MANAGER         2975          1
        30 CLERK            950          1
        30 MANAGER         2850          1
        30 SALESMAN        5600          4

    DEPTNO   SUM(SAL)    LÉTSZÁM
---------- ---------- ----------
        10       8750          3
        20      10875          5
        30       9400          6

  SUM(SAL)    LÉTSZÁM
---------- ----------
     29025         14
```

2. megoldás (Részcsoportképzéssel)

```
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
  GROUP BY ROLLUP (deptno, job);
```

Eredmény

```
    DEPTNO JOB          SUM(SAL)    LÉTSZÁM
---------- --------- ---------- ----------
        10 CLERK           1300          1
        10 MANAGER         2450          1
        10 PRESIDENT       5000          1
        10                 8750          3
        20 CLERK           1900          2
        20 ANALYST         6000          2
        20 MANAGER         2975          1
        20                10875          5
        30 CLERK            950          1
        30 MANAGER         2850          1
        30 SALESMAN        5600          4
        30                 9400          6
                          29025         14

13 sor kijelölve.
```

Megjegyzés

Látható tehát, hogy a ROLLUP operátor nélküli GROUP BY utasításrész használatakor az eredeti feladat megoldásához három utasításra volt szükség, mivel így csupán egyszerű csoportösszegeket tudunk képezni a megadott oszlopok szerinti csoportosításban.

A ROLLUP operátort használva már összesített részösszegeket is kapunk a paraméteroszlopok szerint jobbról balra haladva, míg végül az utolsó sorban megkapjuk az összesített részösszeget, vagyis a végösszeget. Ilyen módon az eredeti feladat megoldásához egyetlen utasítás elegendő volt.

13.2. példa

Listázza ki munkakörönként és azon belül részlegenként az összfizetést, az átlagfizetést, valamint a csoportlétszámot.

Megoldás

```
SELECT job                AS munkakör,
       deptno             AS részleg,
       SUM(sal)           AS részösszeg,
       ROUND(AVG(sal),0)  AS átlag,
       COUNT(*)           AS létszám
  FROM emp
  GROUP BY ROLLUP (job, deptno);
```

Eredmény

```
MUNKAKÖR     RÉSZLEG RÉSZÖSSZEG      ÁTLAG    LÉTSZÁM
--------- ---------- ---------- ---------- ----------
CLERK            10       1300       1300          1
CLERK            20       1900        950          2
CLERK            30        950        950          1
CLERK                     4150       1038          4
ANALYST          20       6000       3000          2
ANALYST                   6000       3000          2
MANAGER          10       2450       2450          1
MANAGER          20       2975       2975          1
MANAGER          30       2850       2850          1
MANAGER                   8275       2758          3
SALESMAN         30       5600       1400          4
SALESMAN                  5600       1400          4
PRESIDENT        10       5000       5000          1
PRESIDENT                 5000       5000          1
                         29025       2073         14

15 sor kijelölve.
```

13.3. példa

Oldja meg az előző példa feladatát oly módon, hogy a lista összesítéseket tartalmazó sorába megjelenjen az összesítés értelmezése.

1. megoldás (CASE kifejezéssel)

```
SELECT
    NVL(job,'MunkakörÖsszes')   AS Munkakör,
    CASE
      WHEN job is NULL
        THEN NULL
      ELSE
        SUBSTR(NVL(TO_CHAR(deptno),'RészlegÖsszes'),1,15)
    END                         AS részleg,
    SUM(sal)                    AS részösszeg,
    ROUND(AVG(sal))             AS átlag,
    COUNT(*)                    AS létszám
  FROM emp
  GROUP BY ROLLUP (job, deptno);
```

2. megoldás (DECODE függvénnyel)

```
SELECT
    NVL(job,'MunkakörÖsszes')   AS Munkakör,
```

```
    DECODE(NVL(job,'X'),
       'X',  NULL,
       SUBSTR(NVL(TO_CHAR(deptno),'RészlegÖsszes'),1,15))
                                AS részleg,
    SUM(sal)                    AS részösszeg,
    ROUND(AVG(sal))             AS átlag,
    COUNT(*)                    AS létszám
  FROM emp
  GROUP BY ROLLUP (job, deptno);
```

Eredmény (mindkét esetben)

MUNKAKÖR	RÉSZLEG	RÉSZÖSSZEG	ÁTLAG	LÉTSZÁM
CLERK	10	1300	1300	1
CLERK	20	1900	950	2
CLERK	30	950	950	1
CLERK	RészlegÖsszes	4150	1038	4
ANALYST	20	6000	3000	2
ANALYST	RészlegÖsszes	6000	3000	2
MANAGER	10	2450	2450	1
MANAGER	20	2975	2975	1
MANAGER	30	2850	2850	1
MANAGER	RészlegÖsszes	8275	2758	3
SALESMAN	30	5600	1400	4
SALESMAN	RészlegÖsszes	5600	1400	4
PRESIDENT	10	5000	5000	1
PRESIDENT	RészlegÖsszes	5000	5000	1
MunkakörÖsszes		29025	2073	14

15 sor kijelölve.

A CUBE operátor

A CUBE operátor hasonló a ROLLUP operátorhoz, de segítségével a sorok csoportjainak valamely oszlop(ok)ra vonatkozó összes csoportosítási kombináció szerinti részösszege meghatározható.

```
GROUP BY CUBE (oszlop1 [, oszlop2]...)
```

A részösszegképzést a zárójelben lévő oszlopok szerint végzi a megadott oszlopsorrend szerint jobbról balra, de az összes kiválasztási kombinációt előállítva (*lásd* a következő példát). Az üres attribútumú sorok adják a részösszegeket, illetve a végösszeget.

13.4. példa

Oldja meg a 13.2. példa feladatát oly módon, hogy a `deptno` és a `job` oszlopokra vonatkozó összes felbontási kombináció részösszegét megjelenítse. Tehát képezzen részösszeget

- munkakörönként és azon belül részlegenként (ez a `Részösszeg11`) (ezzel ekvivalens, ezért külön nem elvégzendő: részlegenként és azon belül munkakörönként),
- munkakörönként (ez a `Részösszeg10`),
- részlegenként (ez a `Részösszeg01`),
- teljes táblára (ez a `Részösszeg00`, amely egyben a `Végösszeg`).

1. megoldás (A `GROUP BY` utasításrész használatával)

```
COLUMN deptno              FORMAT 99

PROMPT Részösszeg11:
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
  GROUP BY deptno, job;

PROMPT Részösszeg10:
SELECT job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
  GROUP BY job;

PROMPT Részösszeg01:
SELECT deptno, SUM(sal), COUNT(*) AS Létszám
  FROM emp
  GROUP BY deptno;

PROMPT Részösszeg00 (Végösszeg):
SELECT SUM(sal), COUNT(*) AS Létszám
  FROM emp;

CLEAR COLUMNS
```

Eredmény

```
Részösszeg11:
DEPTNO JOB         SUM(SAL)    LÉTSZÁM
------ --------- ---------- ----------
    10 CLERK           1300          1
    10 MANAGER         2450          1
    10 PRESIDENT       5000          1
    20 CLERK           1900          2
    20 ANALYST         6000          2
    20 MANAGER         2975          1
```

```
30 CLERK              950          1
30 MANAGER           2850          1
30 SALESMAN          5600          4

Részösszeg10:
JOB          SUM(SAL)    LÉTSZÁM
---------  ----------  ----------
ANALYST          6000           2
CLERK            4150           4
MANAGER          8275           3
PRESIDENT        5000           1
SALESMAN         5600           4

Részösszeg01:
DEPTNO   SUM(SAL)    LÉTSZÁM
------ ----------  ----------
    10       8750           3
    20      10875           5
    30       9400           6

Részösszeg00 (Végösszeg):
  SUM(SAL)    LÉTSZÁM
----------  ----------
     29025          14
```

2. megoldás (A `GROUP BY` utasításrész és a `CUBE` operátor használatával)

```
SELECT deptno, job, SUM(sal), COUNT(*) AS Létszám
  FROM emp
  GROUP BY CUBE (deptno, job);
```

Eredmény

```
    DEPTNO JOB          SUM(SAL)    LÉTSZÁM
---------- ---------  ----------  ----------
                            29025          14    ⇐ Részösszeg00
           CLERK             4150           4    ⇐ Részösszeg10
           ANALYST           6000           2    ⇐ Részösszeg10
           MANAGER           8275           3    ⇐ Részösszeg10
           SALESMAN          5600           4    ⇐ Részösszeg10
           PRESIDENT         5000           1    ⇐ Részösszeg10
        10                   8750           3    ⇐ Részösszeg01
        10 CLERK             1300           1    ⇐ Részösszeg11
        10 MANAGER           2450           1    ⇐ Részösszeg11
        10 PRESIDENT         5000           1    ⇐ Részösszeg11
        20                  10875           5    ⇐ Részösszeg01
```

```
20 CLERK            1900          2    <= Részösszeg11
20 ANALYST          6000          2    <= Részösszeg11
20 MANAGER          2975          1    <= Részösszeg11
30                  9400          6    <= Részösszeg01
30 CLERK             950          1    <= Részösszeg11
30 MANAGER          2850          1    <= Részösszeg11
30 SALESMAN         5600          4    <= Részösszeg11

18 sor kijelölve.
```

Megjegyzés

- Tehát a CUBE operátor minden csoportosítási kombinációt előállít, míg a ROLLUP operátor csak az adott sorrendhez tartozókat. Ha a fenti példában a CUBE helyett ROLLUP operátort használtunk volna, akkor nem kaptuk volna meg a részösszeg10 értékeket (illetve az ezeket tartalmazó sorokat).
- Mivel ilyen módon igen nagy eredménytábla keletkezik (melyben szereplő összes csoportosítási kombinációra az esetek jelentős részében nem tartunk igényt), ezért a CUBE operátort viszonylag ritkán használjuk.

A GROUPING függvény

A GROUPING indikátorfüggvény megmutatja, hogy a lista egy sorában „ki a felelős" a paraméterek közül. Használható a ROLLUP és a CUBE operátorokkal együtt oly módon, hogy 0 értéket rendel ahhoz az oszlophoz, amelyik szerint csoportképzés történt (vagyis amelyik csoportképző oszlop). Természetesen a csoportosítás vonatkozhat több oszlopra is.

13.5. példa

Listázza a részlegenkénti foglalkozásonkénti összfizetéseket, a részlegenkénti összfizetéseket, munkakörönkénti összfizetéseket, valamint a vállalati összfizetést, úgy, hogy minden sorban 0 értéket rendeljen ahhoz az oszlophoz, amelyik szerint ott történt csoportképzés, és 1 értéket pedig ahhoz, amelyik szerint ott *nem* történt csoportképzés.

1. megoldás (A GROUPING függvény használatával)

```
COLUMN „Részleg"          FORMAT 99
SELECT deptno             AS "Részleg",
       job                AS "Foglalkozás",
       SUM(sal)           AS "Részösszeg",
       GROUPING(deptno)   AS "RészlegInd",
       GROUPING(job)      AS "FoglalkozásInd"
  FROM emp
  GROUP BY CUBE (deptno, job);
CLEAR COLUMNS
```

Eredmény

```
Részleg Foglalkoz Részösszeg RészlegInd FoglalkozásInd
------- --------- ---------- ---------- --------------
                       29025          1              1
        CLERK           4150          1              0
        ANALYST         6000          1              0
        MANAGER         8275          1              0
        SALESMAN        5600          1              0
        PRESIDENT       5000          1              0
     10                 8750          0              1
     10 CLERK           1300          0              0
     10 MANAGER         2450          0              0
     10 PRESIDENT       5000          0              0
     20                10875          0              1
     20 CLERK           1900          0              0
     20 ANALYST         6000          0              0
     20 MANAGER         2975          0              0
     30                 9400          0              1
     30 CLERK            950          0              0
     30 MANAGER         2850          0              0
     30 SALESMAN        5600          0              0

18 sor kijelölve.
```

2. megoldás (A `GROUPING` függvény és a `CASE` kifejezés használatával)

```
COLUMN „Részleg"              FORMAT 99
SELECT deptno                 AS "Részleg",
       job                    AS "Foglalkozás",
       SUM(sal)               AS "Részösszeg",
       CASE
         WHEN GROUPING(deptno)=1 THEN 'nem csoportosít'
         ELSE 'csoportosít'
       END                    AS "RészlegInd",
       CASE
         WHEN GROUPING(job)=1 THEN 'nem csoportosít'
         ELSE 'csoportosít'
       END                    AS "FoglalkozásInd"
  FROM emp
  GROUP BY CUBE (deptno,job);
CLEAR COLUMNS
```

Eredmény

```
Részleg Foglalkoz Részösszeg RészlegInd      FoglalkozásInd
------- --------- ---------- --------------- --------------
                       29025 nem csoportosít nem csoportosít
        CLERK           4150 nem csoportosít csoportosít
        ANALYST         6000 nem csoportosít csoportosít
        MANAGER         8275 nem csoportosít csoportosít
        SALESMAN        5600 nem csoportosít csoportosít
        PRESIDENT       5000 nem csoportosít csoportosít
     10                 8750 csoportosít     nem csoportosít
     10 CLERK           1300 csoportosít     csoportosít
     10 MANAGER         2450 csoportosít     csoportosít
     10 PRESIDENT       5000 csoportosít     csoportosít
     20                10875 csoportosít     nem csoportosít
     20 CLERK           1900 csoportosít     csoportosít
     20 ANALYST         6000 csoportosít     csoportosít
     20 MANAGER         2975 csoportosít     csoportosít
     30                 9400 csoportosít     nem csoportosít
     30 CLERK            950 csoportosít     csoportosít
     30 MANAGER         2850 csoportosít     csoportosít
     30 SALESMAN        5600 csoportosít     csoportosít

18 sor kijelölve.
```

Ellenőrzés

```
SELECT deptno, SUM(sal)
  FROM emp
  GROUP BY deptno;
```

Eredmény

```
    DEPTNO   SUM(SAL)
---------- ----------
        10       8750
        20      10875
        30       9400
```

Megjegyzés

- Az ellenőrzés mintapéldájában a `deptno` oszlop szerint csoportosított részösszegeket kaptuk meg, melyeket tartalmazó feladatmegoldásbeli sorokban ezért a `RészlegInd` értéke `0`, illetve `csoportosít`.
- Mivel a `GROUPING` függvény paramétere csak egyetlen oszlop, így több oszlop esetén használata nehézkes, ezért ilyenkor inkább az alább bemutatásra kerülő `GROUPING_ID` indikátorfüggvényt alkalmazzuk a csoportosítás kifejezésére.
- Megjegyezzük, hogy a fenti feladat a `CASE` kifejezés helyett a `DECODE` függvénnyel is megoldható lett volna.

A GROUPING_ID függvény

A GROUPING_ID indikátorfüggvény értéke egy pozitív egész szám, amely összhangban van a GROUPING függvénnyel, és szintén minden sorhoz társítva van. Ennek a függvénynek egy adott sorbeli értéke megadja, hogy a ROLLUP vagy CUBE operátor paraméteroszlopai közül melyik szerint *nem* történt a csoportképzés. Ennek meghatározása úgy történik, hogy a paraméteroszlopok közül a legjobboldalihoz tartozik 2-nek a 0-adik hatványa, attól balra az 1-ső hatványa stb. Így egy adott sorban e függvény értéke a csoportképzésben aktuálisan részt *nem* vevő paraméteroszlopokhoz tartozó értékek összege. Például CUBE (A, B, C) esetén, ha egy sor a B oszlop szerinti csoportosítás eredménye, akkor a GROUPING_ID függvény értéke 5, mivel a C oszlop miatt $2^0 = 1$, az A oszlop miatt pedig $2^2 = 4$. Az így keletkező függvényértékekkel oly módon lehet rendezni az eredménytábla sorait (*lásd* az alábbi példa megjegyzését), hogy a legtöbb oszlop szerinti csoportosítás (a legfinomabb felbontás) eredményei az eredménytábla elejére, míg a csoportosítás nélküli eredmények (például a végösszeg) az eredménytábla végére kerüljenek.

13.6. példa

Listázza ki a részlegenkénti foglalkozásonkénti összfizetéseket, a részlegenkénti összfizetéseket, a munkakörönkénti összfizetéseket, valamint a vállalati összfizetést, úgy, hogy egy indikátoroszlop minden sorban jelezze, hogy mely oszlop szerint történt a csoportosítás.

1. megoldás

```
COLUMN „Részleg"                  FORMAT 99
SELECT deptno                     AS "Részleg",
       job                        AS "Foglalkozás",
       SUM(sal)                   AS "Részösszeg",
       GROUPING_ID(deptno, job)   AS "Indikátor"
  FROM emp
  GROUP BY CUBE(deptno,job);
CLEAR COLUMNS
```

Eredmény

```
Részleg Foglalkoz Részösszeg  Indikátor
------- --------- ---------- ----------
                       29025          3
        CLERK           4150          2
        ANALYST         6000          2
        MANAGER         8275          2 (***)
        SALESMAN        5600          2
        PRESIDENT       5000          2
     10                 8750          1
     10 CLERK           1300          0
     10 MANAGER         2450          0
     10 PRESIDENT       5000          0
```

```
20                10875          1
20 CLERK           1900          0 (*)
20 ANALYST         6000          0
20 MANAGER         2975          0
30                 9400          1 (**)
30 CLERK            950          0
30 MANAGER         2850          0
30 SALESMAN        5600          0

18 sor kijelölve.
```

Megjegyzés

Egy oszlophoz rendelt 0 (helyiértéken figyelembe véve) itt is a csoportosítást jelzi. Ennek megfelelően a `GROUPING_ID(deptno, job)` függvény az értelmezése szerint 0+0=0 értéket ad, ha mind a `deptno`, mind a `job` szerint, 0+1=1 értéket, ha csak a `deptno` szerint és 2+0=2 értéket, ha csak a `job` szerint történt csoportképzés. Ennek megfelelően a fenti lista (*) sora a `GROUP BY deptno, job`, a (**) sora a `GROUP BY deptno`, míg a (***) sora a `GROUP BY job` csoportképzés eredményét adja a részösszegképzésben.

2. megoldás (Tekintsük a fenti listát az indikátorfüggvény szerint rendezve)

```
COLUMN "Részleg"                FORMAT 99
SELECT deptno                   AS "Részleg",
       job                      AS "Foglalkozás",
       SUM(sal)                 AS "Részösszeg",
       GROUPING_ID(deptno, job) AS "Indikátor"
  FROM emp
  GROUP BY CUBE (deptno,job)
  ORDER BY "Indikátor";
CLEAR COLUMNS
```

Eredmény

```
Részleg Foglalkoz Részösszeg  Indikátor
------- --------- ---------- ----------
     10 CLERK           1300          0
     10 MANAGER         2450          0
     10 PRESIDENT       5000          0
     20 CLERK           1900          0
     30 CLERK            950          0
     30 SALESMAN        5600          0
     30 MANAGER         2850          0
     20 MANAGER         2975          0
     20 ANALYST         6000          0
     10                 8750          1
     20                10875          1
```

```
30                  9400          1
   CLERK            4150          2
   ANALYST          6000          2
   MANAGER          8275          2
   PRESIDENT        5000          2
   SALESMAN         5600          2
                   29025          3

18 sor kijelölve.
```

Megjegyzés

Az eredmény ilyen módon rendezve áttekinthető képet ad a különböző szintű csoportok részösszegeiről. Az 1. megoldásbeli (rendezés nélküli) lista előnye viszont, hogy a hierarchiaszintek egymás közötti kapcsolatát mutatja szemléletesen.

A GROUPING SETS függvény

A GROUPING SETS a GROUP BY utasításrész további kiterjesztése. Segítségével több csoportot képezhetünk egy lekérdezésben. Ilyenkor a GROUPING SETS paramétereként megadott oszlophalmazokra (és csak azokra) végzi el a csoportosítást.

13.7. példa

Képezzen részösszegeket a fizetés (sal) oszlopon az alábbi csoportosításokban:

- mgr, deptno, job,
- deptno, job és
- mgr.

Megoldás

```
SELECT mgr, deptno, job, SUM(sal)
  FROM emp
  GROUP BY
    GROUPING SETS((mgr, deptno, job),
                  (deptno, job),
                  (mgr));
```

Eredmény

```
       MGR     DEPTNO JOB         SUM(SAL)
---------- ---------- --------- ----------
      7782         10 CLERK           1300    <= (mgr, deptno, job)
      7839         10 MANAGER         2450    <= (mgr, deptno, job)
                   10 PRESIDENT       5000    <= (mgr, deptno, job)
      7788         20 CLERK           1100    <= (mgr, deptno, job)
      7902         20 CLERK            800    <= (mgr, deptno, job)
```

```
7566        20 ANALYST      6000   <= (mgr, deptno, job)
7839        20 MANAGER      2975   <= (mgr, deptno, job)
7698        30 CLERK         950   <= (mgr, deptno, job)
7839        30 MANAGER      2850   <= (mgr, deptno, job)
7698        30 SALESMAN     5600   <= (mgr, deptno, job)
            10 CLERK        1300   <= (deptno, job)
            10 MANAGER      2450   <= (deptno, job)
            10 PRESIDENT    5000   <= (deptno, job)
            20 CLERK        1900   <= (deptno, job)
            20 ANALYST      6000   <= (deptno, job)
            20 MANAGER      2975   <= (deptno, job)
            30 CLERK         950   <= (deptno, job)
            30 MANAGER      2850   <= (deptno, job)
            30 SALESMAN     5600   <= (deptno, job)
7566                        6000   <= (mgr)
7698                        6550   <= (mgr)
7782                        1300   <= (mgr)
7788                        1100   <= (mgr)
7839                        8275   <= (mgr)
7902                         800   <= (mgr)
                            5000   <= (mgr)

26 sor kijelölve.
```

Megjegyzés
Láthatóan ez a legáttekinthetőbb módja a csoportfüggvények részértéke meghatározásának.

Az SQL analitikus függvényei

Az analitikus függvények mind az elérhető funkciókban, mind az adattáblák szerkezetének kezelésében új szemléletet jelentenek, új függvénycsaládot alkotnak. Egyrészt a gazdaságban széles körben alkalmazott kiértékelő függvények, statisztikai számítások széles skáláját tartalmazzák, másrészt pedig például az adott tulajdonság szerint csoportosított sorok által alkotott táblatartományok belsejében mozgó „ablakon" végzett részösszegképzés új lehetőségeket nyitott az adatfeldolgozás előtt. Ezek a függvények jelenleg még nincsenek a standard SQL nyelvben, de ezeké a jövő, igénylik a felhasználók, ezért szabványosítás alatt állnak.

A bemutatásra kerülő analitikus függvényeket három csoportba osztva ismertetjük. Ezek a csoportok a *rang*, a *statisztikai* és a *hisztogram* függvények csoportjai. Egyes függvények definíció szerint ugyan nem analitikusak, ám a többihez hasonló funkciójuk, vagy egyes analitikus függvényeket a használat során jól kiegészítő jellegük miatt mégis itt ismertetjük őket. Ezekre a tulajdonságukra bemutatásuk során külön ki fogunk térni.

Az analitikus függvények általános felépítése

Megadás

analitikus_függvény_neve([*paraméter*]) OVER (*analitikus_záradék*)

Értelmezés (munkatábla, paraméter)

Az analitikus függvények egy SELECT utasításban kizárólag a szelekciós listában szerepelhetnek (allekérdezésként természetesen már bárhol). Ennek megfelelően azon a táblaterületen működnek, melyeket a SELECT utasítás egyéb utasításrészei (FROM, WHERE, GROUP BY, HAVING) a számára rendelkezésre bocsátanak. Ezt a táblaterületet az analitikus függvények *munkatáblájának* nevezzük.

Egy analitikus függvény *paramétere* tetszőleges oszlopkifejezés lehet (hasonlóan a csoportfüggvényhez), mely egymással kompatibilis oszlopneveket, konstansokat, műveleteket és sorfüggvényeket tartalmazhat.

A pontosabb értelmezést az egyes részek bemutatásánál adjuk meg. (Részletes leírás az Oracle-rendszer Help funkciója segítségével kapható (magyar nyelven vázlatos ismertetést *lásd* [15].)

Megjegyzés

Az analitikus függvények bizonyos értelemben a csoportfüggvények kiterjesztésének tekinthetők, bár elég jelentős fogalmi módosulással. Tény mindenesetre, hogy ha egy analitikus függvény neve szerepel a csoportfüggvények között, akkor az (legalábbis szintaktikailag) előállítható az OVER opció elhagyásával.

AZ ANALITIKUS ZÁRADÉK

Megadás

[[*partíció-tag*] [*rendező-tag* [*ablak-tag*]]]

Értelmezés (munkaterület)

Az analitikus záradék az analitikus függvény munkatáblájának *munkaterületét* jelöli ki. Értelmezését az alábbiakban tagonként adjuk meg. A szintaktikai leírásból láthatóan az analitikus záradék teljesen el is maradhat. Ekkor a függvény értelmezési tartománya a teljes munkatábla.

Megjegyzés

Az analitikus záradék hiánya egy üres, azaz OVER() alakú OVER opciót eredményez, mely nem azonos az OVER opció hiányával. Ez utóbbi – mint feljebb már jeleztük – (formailag legalábbis) csoportfüggvénnyé alakítaná az analitikus függvényt.

A partíció-tag

Megadás

PARTITION BY *oszlopkifejezések_listája*

Értelmezés (partíció)
A partíció-tag segítségével a munkatábla munkaterületén olyan összefüggő táblatartományokat (azaz partíciókat) lehet képezni, ahol egy partíción belül az oszlopkifejezések értéke azonos.

Ez egyrészt egy rendezést jelent az oszlopkifejezés szerint. Ha az oszlopkifejezés-lista többelemű, akkor az abban megadott sorrend szerint történik elsődleges, másodlagos stb. rendezés.

Másrészt egy-egy partíció egymást követő sorai jelölik ki az analitikus függvény egy-egy értelmezési tartományát, vagyis az analítikus függvény minden partíción újra és újra kiértékelésre kerül.

Megjegyezzük, hogy a partícióképzés láthatóan nem azonos a `GROUP BY` utasításrész által végzett csoportképzéssel. A csoportfüggvények minden csoporthoz csak egyetlen értéket rendelnek, míg az analitikus függvények a partíciók minden sorához rendelnek értéket. A csoportok elemi sorait csak a csoportképző oszlopok értékei és a nem csoportképző oszlopokon kiértékelt csoportfüggvények értékei reprezentálják, maguk az elemi sorok, valamint a további oszlopok „eltűnnek".

A rendező-tag
Megadás
`ORDER BY` *oszlopkifejezések_listája* `[ ASC | DESC ]`

Értelmezés (partíciókon belüli rendezés)
A rendező-tag minden egyes partíción belül rendez a benne megadott oszlopkifejezések szerint. Sokszor azonos az analitikus függvény paramére és a rendező-tagban megadott oszlop. Ilyenkor egy partíción belüli rendezés és az analitikus függvényérték (például kumulált összeg) jól látható kapcsolatban áll egymással, más esetben ez a kapcsolat rejtettebb (*lásd* 13.15. és 13.16. példa).

Egyes analitikus függvényeknél (például a rang függvények analitikus alakjánál) hiányzik a paraméter megadása. Ekkor a függvény paramétere tulajdonképpen nem más, mint a rendező-tagban megadott oszlop (*lásd* 13.11. példa).

Az ablak-tag
Az ablak-tag kijelöli minden partíción belül azt az (úgynevezett aktuális sorhoz rögzített méretű, és a partíció megadott résztartományán folyamatosan mozgó) összefüggő táblatartományt, melynek sorain történik az analitikus függvény által kijelölt műveletvégzés (például kumulált összegzés – *lásd* 13.17. példa).

Az ablak-tag használata esetén az analitikus függvény lényegében egy olyan csoportfüggvény, melynek csoportképző attribútuma a paraméterében megadott oszlopkifejezés, hatóköre az ablak. A függvényértéket az aktuális sorhoz rendeli.

Az ablaktartomány kijelölése lehet fizikai (`ROWS`), illetve logikai (`RANGE`).

A fizikai tartományt kijelölő ablak-tag megadása
`ROWS BETWEEN` *kifejezés1* `PRECEEDING AND` *kifejezés2* `FOLLOWING`

ahol *kifejezés1* megadja azt, hogy az ablak a partícióján belül az aktuális sor (`CURRENT ROW`) előtt hány sorral kezdődjön, a *kifejezés2* pedig azt, hogy utána hány sorig tartson.

A logikai tartományt kijelölő ablak-tag megadása

```
RANGE BETWEEN [UNBOUNDED PRECEDING | CURRENT ROW] AND
              [CURRENT ROW | UNBOUNDED FOLLOWING]
```

vagy

```
RANGE [UNBOUNDED PRECEDING | CURRENT ROW]
```

ahol az egyes kulcskifejezések jelentése:

- `UNBOUNDED PRECEDING`: a partíció első sorától kezdődően,
- `UNBOUNDED FOLLOWING`: a partíció utolsó soráig,
- `CURRENT ROW`: a partíció aktuális sorától, illetve soráig.

Rang függvények

AZ ANALITIKUS RANG FÜGGVÉNYEK

Az analitikus rang függvények az `OVER` opcióval ellátott `RANK`, `DENSE_RANK` és `PERCENT_RANK` függvények. Jellegzetességük, hogy nem tartalmaznak ablak-tagot, és paraméterlistájuk üres. Tulajdonképpen a rendező tagban szereplő oszlopkifejezés a paraméterük, aszerint végeznek rangsorolást a kijelölt partíciókban. Hiányzó partíció-tag esetén a rangsorolás az egész táblára vonatkozik.

A `RANK` függvény az 1-től a partíció méretéig terjedő számtartományban sorszámozza (azaz látja el „rang"-gal) a partíció elemeit alkotó sorokat oly módon, hogy az azonos szinten levőknek ugyanazt a rangot adja, a következő szinten álló pedig annyival nagyobb rangsorbeli számot kap, ahány egyforma rangértékűt talált az előző szinten. (Tekintsük a 13.8. példát. `Martin` és `Ward` rangja egyaránt `4`, de a rangsorban a következő, `Miller` rangja már `6`. Az 5-ös rangérték tehát kimaradt.) Szemléletesen szólva, e függvény esetén sor *n* rangértéke azt jelenti, hogy `(`*n*`-1)` darab sor előzi meg a rangsorban.

A `DENSE_RANK` függvény működése hasonlít a `RANK` függvényhez, ám ez "tömöríti" a sorszámozást. Az előző példánál maradva, ez a függvény már kiosztja az 5-ös rangértéket. Szemléletesen, a függvény esetén az *n*-edik rangú sor a minősítés szempontjából az *n*-edik, bár lehet, hogy `(`*n*`-1)`-nél több sor előzi meg a rangsorban.

Mind a `RANK`, mind a `DENSE_RANK` függvény esetén megállapítható, hogy az általuk visszaadott rangszámok elvileg mást jelentenek, mint a korábban már megismert `ROWNUM` függvény (pszeudooszlop). Ez utóbbi egyáltalán nem tudott mit kezdeni az azonos minősítésű sorokkal, gyakorlatilag a tárolási sorrendtől függően adott az egyiknek alacsonyabb, a másiknak magasabb sorszámot.

A `PERCENT_RANK` függvény a rangértékeket a `[0..1]` zárt tartománybeli értékként adja meg a következőképpen: ha egy sor rangja a `RANK` függvény szerint *n*, és a partíció (mely tartalmazza) *k* sorból áll, akkor a rangja `(`*n*`-1)/(`*k*`-1)` a `PERCENT_RANK` függvény szerint. Láthatóan a legkisebb kiosztott rangérték e függvény szerint a `0`, ám a legnagyobb nem feltétlenül az `1`.

Az analitikus rang függvények használatának szintaktikája:

```
RANK() OVER ([PartíciósTag]
                ORDER BY oszlopkifejezés1 [DESC | ASC]
                        [, oszlopkifejezés2 [DESC | ASC] ]...)

DENSE_RANK() OVER ([PartíciósTag]
                    ORDER BY oszlopkifejezés1 [DESC | ASC]
                            [, oszlopkifejezés2 [DESC | ASC] ]...)

PERCENT_RANK() OVER ([PartíciósTag]
                      ORDER BY oszlopkifejezés1 [DESC | ASC]
                              [, oszlopkifejezés2 [DESC | ASC] ]...)
```

AZ AGGREGÁLÓ RANG FÜGGVÉNYEK

Az OVER opció helyett a WITHIN GROUP opcióval ellátott rang függvényeket aggregáló rang függvényeknek nevezzük. Valójában nem analitikus függvények, de a működésük mögött az analitikus rang függvények állnak. Ezek paraméterlistája már nem üres.

Egy aggregáló rang függvény a paramétereként megadott konstans kifejezés értékének helyét keresi meg az egyes partíciókban (vagy az egész munkatáblában) annak feltételezésével, hogy e konstans érték maga is az egyes partíciók (vagy az egész munkatábla) eleme. (Pontosabban azt feltételezve, hogy e konstans az egyes partíciókat alkotó sorokra vonatkozóan a rendező-tag oszlopkifejezése értéktartományának eleme.) Látható tehát, hogy e függvény a *„mi lenne, ha..."* típusú kérdések megválaszolására alkalmas, és működése során felhasználja az analitikus rangfüggvényt.

Használatuk jellegzetessége, hogy mivel nem analitikus függvények, ezért az analitikus függvények partícióstruktúráját a GROUP BY csoportosító utasításrésszel kell „pótolni". Jól szemléltetik mindezt a 13.9. és 13.10. példa, ahol az első, megadott partíciókon végez rangsorolást, az utóbbi pedig azt adja meg, hogy az egyes partíciókon belül egy hipotetikus érték milyen rangsorú lenne. A partíciókat itt egy GROUP BY utasításrész adja meg.

További jellegzetességük, hogy a fentiekből következően az aggregáló rang függvények már csak azért sem tartalmazhatnak ablak-tagot, mert nem analitikusak.

Megjegyezzük, hogy a konstans paraméterkifejezések és a rendező-tagban lévő oszlopkifejezések sorrendjének és típusának nyilván egyeznie kell.

Az aggregáló rang függvények használatának szintaktikája

```
RANK(kifejezés1 [, kifejezés2]...) WITHIN GROUP
  (ORDER BY oszlopkifejezés1 [DESC | ASC]
          [, oszlopkifejezés2 [DESC | ASC] ]...)

DENSE_RANK(kifejezés1 [, kifejezés2]...) WITHIN GROUP
  (ORDER BY oszlopkifejezés1 [DESC | ASC]
          [, oszlopkifejezés2 [DESC | ASC] ]...)
```

```
PERCENT_RANK(kifejezés1 [, kifejezés2]...) WITHIN GROUP
  (ORDER BY oszlopkifejezés1 [DESC | ASC]
        [, oszlopkifejezés2 [DESC | ASC] ]...)
```

13.8. példa

Készítsen szkript programot, mely előállítja a dolgozók fizetési rangsorát, valamint megállapítja, hogy egy, a felhasználó által megadott fizetési érték hányadik lenne a feltételezett rangsorban.

Megoldás

```
-- Szkript program az analitikus és az aggregáló
-- rang függvények szemléltetésére
COLUMN ename                FORMAT A7
COLUMN sal                  FORMAT 9999
COLUMN "SzázalékosRangsor" FORMAT 0.99

-- Analitikus
SELECT ename,
       sal,
       RANK() OVER (ORDER BY sal DESC)          AS "NormálRangsor",
       DENSE_RANK() OVER (ORDER BY sal DESC)    AS "TömörítettRangsor",
       PERCENT_RANK() OVER (ORDER BY sal DESC)  AS "SzázalékosRangsor"
  FROM emp;

-- Aggregáló
ACCEPT fiz PROMPT 'Fizetés: '
SELECT UPPER('Hányadik?')                       AS "Kérdés",
       RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
                                                AS "NormálRangsor",
       DENSE_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
                                                AS "TömörítettRangsor",
       PERCENT_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
                                                AS "SzázalékosRangsor"
  FROM emp;
CLEAR COLUMNS
```

Eredmény

```
ENAME      SAL NormálRangsor TömörítettRangsor SzázalékosRangsor
------- ----- ------------- ----------------- -----------------
KING      5000             1                 1              0.00
FORD      3000             2                 2              0.08
SCOTT     3000             2                 2              0.08
JONES     2975             4                 3              0.23
BLAKE     2850             5                 4              0.31
```

```
CLARK     2450          6          5        0.38
ALLEN     1600          7          6        0.46
TURNER    1500          8          7        0.54
MILLER    1300          9          8        0.62
MARTIN    1250         10          9        0.69
WARD      1250         10          9        0.69
ADAMS     1100         12         10        0.85
JAMES      950         13         11        0.92
SMITH      800         14         12        1.00

14 sor kijelölve.

Fizetés: 4000
régi   2:          RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
új   2:          RANK(4000) WITHIN GROUP (ORDER BY sal DESC)
régi   4:          DENSE_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
új   4:          DENSE_RANK(4000) WITHIN GROUP (ORDER BY sal DESC)
régi   6:          PERCENT_RANK(&fiz) WITHIN GROUP (ORDER BY sal DESC)
új   6:          PERCENT_RANK(4000) WITHIN GROUP (ORDER BY sal DESC)

Kérdés    NormálRangsor TömörítettRangsor SzázalékosRangsor
--------- ------------- ----------------- -----------------
HÁNYADIK?             2                 2              0.07
```

Megjegyzés.
A „hányadik?" kérdésre adott válaszok megadják az eredménytábla egyes rangsorai-ban (oszlopaiban), hogy a felhasználó által megadott fizetésű dolgozó adatainak az emp táblába való feltételezett beillesztése után e dolgozó az új rangsorokban hányadik lenne.

13.9. példa

Készítse el a dolgozók részlegenkénti rangsorát elsődlegesen a fizetés, másodlagosan a munkakör szerint rendezve.

Megoldás

```
COLUMN ename                FORMAT A7
COLUMN sal                  FORMAT 9999
COLUMN deptno               FORMAT 99

SELECT deptno, ename, sal, job,
       RANK() OVER (PARTITION BY deptno
                    ORDER BY sal DESC, job) AS „NormálRangsor",
       DENSE_RANK() OVER (PARTITION BY deptno
                          ORDER BY sal DESC, job) AS "TömörRangsor"
  FROM emp;
```

```
CLEAR COLUMNS
```

Eredmény

```
DEPTNO ENAME      SAL JOB       NormálRangsor TömörRangsor
------ -------- ----- --------- ------------- ------------
    10 KING     5000 PRESIDENT             1            1
    10 CLARK    2450 MANAGER               2            2
    10 MILLER   1300 CLERK                 3            3
    20 FORD     3000 ANALYST               1            1
    20 SCOTT    3000 ANALYST               1            1
    20 JONES    2975 MANAGER               3            2
    20 ADAMS    1100 CLERK                 4            3
    20 SMITH     800 CLERK                 5            4
    30 BLAKE    2850 MANAGER               1            1
    30 ALLEN    1600 SALESMAN              2            2
    30 TURNER   1500 SALESMAN              3            3
    30 MARTIN   1250 SALESMAN              4            4
    30 WARD     1250 SALESMAN              4            4
    30 JAMES     950 CLERK                 6            5

14 sor kijelölve.
```

13.10. példa

Állapítsa meg, hogy a felhasználó által megadott fizetésérték hányadik lenne a fizetések részlegenkénti feltételezett rangsorában.

Megoldás

```
ACCEPT fiz PROMPT 'Fizetés:'

COLUMN SUM(sal)            FORMAT 99999
COLUMN deptno              FORMAT 99

SELECT &fiz                                                AS "MegadottFizetés",
       deptno,
       SUM(sal),
       RANK(&fiz) WITHIN GROUP(ORDER BY sal)               AS "NormálRangsor",
       DENSE_RANK(&fiz) WITHIN GROUP(ORDER BY sal)         AS "TömörRangsor"
  FROM emp
  GROUP BY deptno;

CLEAR COLUMNS
```

Eredmény

```
Fizetés:1500
régi   1: SELECT &fiz                                              AS
"MegadottFizetés",
új   1: SELECT 1500                                              AS
"MegadottFizetés",
régi   4:        RANK(&fiz) WITHIN GROUP(ORDER BY sal)          AS
"NormálRangsor",
új   4:        RANK(1500) WITHIN GROUP(ORDER BY sal)          AS
"NormálRangsor",
régi   5:        DENSE_RANK(&fiz) WITHIN GROUP(ORDER BY sal)  AS
"TömörRangsor"
új   5:        DENSE_RANK(1500) WITHIN GROUP(ORDER BY sal)  AS
"TömörRangsor"

MegadottFizetés DEPTNO SUM(SAL) NormálRangsor TömörRangsor
--------------- ------ -------- ------------- ------------
           1500     10     8750             2            2
           1500     20    10875             3            3
           1500     30     9400             4            3
```

Megjegyzés

- Az eredménytábla rangsor oszlopaiban szereplő értékek azt adják meg, hogy a felhasználó által megadott fizetésű dolgozó adatainak az `emp` táblába való feltételezett beillesztése után ez a dolgozó az egyes részlegek új rangsoraiban hányadik lenne.
- E feladat megoldásának helyességét az előző példa eredménytáblája alapján ellenőrizhetjük.

13.11. példa

Határozza meg a `Dallas`-i és a `New York`-i dolgozók fizetési sorrendjét.

1. megoldás

```
BREAK ON deptno  ON loc
SELECT
    e.deptno,
    loc,
    ename,
    sal,
    DENSE_RANK() OVER(PARTITION BY e.deptno ORDER BY sal DESC)
                 AS sorrend
  FROM emp e,
       dept d
```

```
    WHERE e.deptno = d.deptno  AND
          UPPER(loc) IN ('DALLAS', 'NEW YORK')
    ORDER BY d.deptno, ename;
  CLEAR BREAK
```

2. megoldás

```
  BREAK ON deptno  ON loc
  SELECT
      e.deptno,
      loc,
      ename,
      sal,
      DENSE_RANK() OVER(PARTITION BY e.deptno, ename ORDER BY sal DESC)
                   AS sorrend
    FROM emp e,
         dept d
    WHERE e.deptno = d.deptno  AND
          UPPER(loc) IN ('DALLAS', 'NEW YORK');
  CLEAR BREAK
```

Eredmény (mindkét megoldás esetén)

```
    DEPTNO LOC           ENAME             SAL    SORREND
---------- ------------- ---------- ---------- ----------
        10 NEW YORK      CLARK            2450          2
                         KING             5000          1
                         MILLER           1300          3
        20 DALLAS        ADAMS            1100          3
                         FORD             3000          1
                         JONES            2975          2
                         SCOTT            3000          1
                         SMITH             800          4

8 sor kijelölve.
```

13.12. példa

Határozza meg előbb részlegenként és azon belül munkakörönként, majd munkakörönként és azon belül részlegenként a dolgozók fizetési sorrendjét.

Megoldás

```
  BREAK ON deptno  ON job
  SELECT
      deptno,
      job,
```

```
    ename,
    sal,
    RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
                                                AS NormálRangsor,
    DENSE_RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
                                                AS TömörRangsor
  FROM emp;

BREAK ON job  ON deptno
SELECT
    job,
    deptno,
    ename,
    sal,
    RANK() OVER (PARTITION BY job, deptno ORDER BY sal DESC)
                                                AS NormálRangsor,
    DENSE_RANK() OVER (PARTITION BY job, deptno ORDER BY sal DESC)
                                                AS TömörRangsor
  FROM emp;
CLEAR BREAK
```

Eredmény

DEPTNO	JOB	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
10	CLERK	MILLER	1300	1	1
	MANAGER	CLARK	2450	1	1
	PRESIDENT	KING	5000	1	1
20	ANALYST	FORD	3000	1	1
		SCOTT	3000	1	1
	CLERK	ADAMS	1100	1	1
		SMITH	800	2	2
	MANAGER	JONES	2975	1	1
30	CLERK	JAMES	950	1	1
	MANAGER	BLAKE	2850	1	1
	SALESMAN	ALLEN	1600	1	1
		TURNER	1500	2	2
		MARTIN	1250	3	3
		WARD	1250	3	3

14 sor kijelölve.

JOB	DEPTNO	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
ANALYST	20	FORD	3000	1	1
		SCOTT	3000	1	1

```
CLERK          10 MILLER          1300          1          1
               20 ADAMS           1100          1          1
                  SMITH            800          2          2
               30 JAMES            950          1          1
MANAGER        10 CLARK           2450          1          1
               20 JONES           2975          1          1
               30 BLAKE           2850          1          1
PRESIDENT      10 KING            5000          1          1
SALESMAN       30 ALLEN           1600          1          1
                  TURNER          1500          2          2
                  MARTIN          1250          3          3
                  WARD            1250          3          3

14 sor kijelölve.
```

Megjegyzés

Láthatóan a kétféle particionálás (éppen azért, mert azonos részletezettségű felosztást eredményezett) az egyes dolgozókat ugyanazokkal hozta közös rangsorba és mindkét particionálásban azonos ranggal.

13.13. példa

Határozza meg előbb részlegenként és azon belül munkakörönként, majd csak részlegenként a dolgozók fizetési sorrendjét. A lista legyen elsődlegesen a részlegazonosító, másodlagosan a munkakör, harmadlagosan a dolgozók neve szerint rendezve.

Megoldás

```
BREAK ON deptno  ON job
SELECT
    deptno,
    job,
    ename,
    sal,
    RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
                                              AS NormálRangsor,
    DENSE_RANK() OVER (PARTITION BY deptno, job ORDER BY sal DESC)
                                              AS TömörRangsor
  FROM emp
  ORDER BY deptno, job, ename;

SELECT
    deptno,
    job,
```

```
    ename,
    sal,
    RANK() OVER (PARTITION BY deptno ORDER BY sal DESC)
                                                AS NormálRangsor,
    DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal DESC)
                                                AS TömörRangsor
  FROM emp
  ORDER BY deptno, job, ename;
CLEAR BREAK
```

Eredmény

DEPTNO	JOB	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
10	CLERK	MILLER	1300	1	1
	MANAGER	CLARK	2450	1	1
	PRESIDENT	KING	5000	1	1
20	ANALYST	FORD	3000	1	1
		SCOTT	3000	1	1
	CLERK	ADAMS	1100	1	1
		SMITH	800	2	2
	MANAGER	JONES	2975	1	1
30	CLERK	JAMES	950	1	1
	MANAGER	BLAKE	2850	1	1
	SALESMAN	ALLEN	1600	1	1
		MARTIN	1250	3	3
		TURNER	1500	2	2
		WARD	1250	3	3

14 sor kijelölve.

DEPTNO	JOB	ENAME	SAL	NORMÁLRANGSOR	TÖMÖRRANGSOR
10	CLERK	MILLER	1300	3	3
	MANAGER	CLARK	2450	2	2
	PRESIDENT	KING	5000	1	1
20	ANALYST	FORD	3000	1	1
		SCOTT	3000	1	1
	CLERK	ADAMS	1100	4	3
		SMITH	800	5	4
	MANAGER	JONES	2975	3	2
30	CLERK	JAMES	950	6	5
	MANAGER	BLAKE	2850	1	1
	SALESMAN	ALLEN	1600	2	2
		MARTIN	1250	4	4

```
TURNER          1500          3          3
WARD            1250          4          4

14 sor kijelölve.
```

Megjegyzés

- Tekintsük először a rangsorértékeket. Látható, hogy a két lista – bár felépítésük azonos, mégis különböző rangsorértékeket rendel az egyes dolgozókhoz. Ennek oka a particionálás különböző részletezettségében keresendő. A gyakorlat számára feltehetően a második lista a használhatóbb. A tanulság az, hogy analitikus függvények használatánál adott részletezettségű lista előállításához elegendő kisebb részletezettségű particionálást alkalmazni, mivel előfordulhat, hogy a túlzott részletezettségű particionálás már semmitmondó lesz.
- Figyeljünk fel a rendezésre is. Mindkét listát azonos rendező-utasításrésszel láttuk el, ennek megfelelően azonos sorrendű sorokat kaptunk (itt most csak az első négy oszlopot tekintjük). Ha azonban összehasonlítjuk e példa első listáját az előző példa első listájával, akkor azt láthatjuk, hogy azok között csak a részleg-munkakör partíciókon belüli nevek sorrendjében van különbség, noha az első példa megoldásában nem is alkalmaztunk rendező utasításrészt. Ennek oka az (amint az analitikus függvények partíció-tagjának ismertetésénél már utaltunk rá), hogy a particionáló oszlopkifejezések szerint már történik egy rendezés. Feltehető a kérdés, vajon nem lett volna-e egyszerűbb, ha a név szerinti rendezést az első listát generáló utasításban nem a rendező utasításrészbe tesszük, hanem az analitikus függvény rendező-tagjába `ORDER BY ename,sal...`, vagy `ORDER BY sal,ename...` módon. Csakhogy akkor a rangsor meghatározása már nem csupán a fizetésre, hanem a név–fizetés, vagy a fizetés–név párosra történt volna!

13.14. példa

Állítsa elő a dolgozók fizetésének részlegenkénti, majd a teljes vállalatra vonatkozó halmozott összegeit. (Az ilyen összegképzést kumulált összegnek is nevezik.)

Megoldás

```
COLUMN ename                FORMAT A7
COLUMN sal                  FORMAT 9999
COLUMN deptno               FORMAT 99

PROMPT Részlegösszegek kiíratása dolgozónként:
SELECT ename, emp.deptno, sal, RészlegÖsszeg
  FROM emp,
       (SELECT deptno,
               SUM(sal) RészlegÖsszeg
          FROM emp
          GROUP BY deptno)  Részleg
```

```
  WHERE emp.deptno = Részleg.deptno
  ORDER BY deptno, sal;

PROMPT Részlegösszegek kiíratása gyűjtősorba:
SELECT deptno, ename, SUM(sal)
  FROM emp
  GROUP BY ROLLUP (deptno, ename)
  ORDER by deptno, SUM(sal);

PROMPT Halmozott részlegösszegek kiíratása:
SELECT deptno,
       ename,
       sal,
       RANK() OVER (PARTITION BY deptno ORDER BY sal ASC)
                  AS "NormálRangsor",
       DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal ASC)
                  AS "TömörRangsor",
       SUM(SAL) OVER (PARTITION BY deptno ORDER BY sal ASC)
                  AS "RészlegreHalmozottÖsszeg"
  FROM emp;

PROMPT Halmozott összegek kiíratása a teljes vállalatra:
SELECT ename,
       sal,
       RANK() OVER (ORDER BY sal ASC)
                  AS "NormálRangsor",
       DENSE_RANK() OVER (ORDER BY sal ASC)
                  AS "TömörRangsor",
       SUM(SAL) OVER (ORDER BY sal ASC)
                  AS "HalmozottÖsszeg"
  FROM emp;

CLEAR COLUMNS
```

Eredmény

```
Részlegösszegek kiíratása dolgozónként:
ENAME   DEPTNO   SAL RÉSZLEGÖSSZEG
------- ------- ----- -------------
MILLER       10  1300          8750
CLARK        10  2450          8750
KING         10  5000          8750
SMITH        20   800         10875
ADAMS        20  1100         10875
JONES        20  2975         10875
```

```
SCOTT        20  3000          10875
FORD         20  3000          10875
JAMES        30   950           9400
MARTIN       30  1250           9400
WARD         30  1250           9400
TURNER       30  1500           9400
ALLEN        30  1600           9400
BLAKE        30  2850           9400

14 sor kijelölve.

Részlegösszegek kiíratása gyűjtősorba:
DEPTNO ENAME      SUM(SAL)
------ ------- ----------
    10 MILLER          1300
    10 CLARK           2450
    10 KING            5000
    10                 8750
    20 SMITH            800
    20 ADAMS           1100
    20 JONES           2975
    20 FORD            3000
    20 SCOTT           3000
    20                10875
    30 JAMES            950
    30 WARD            1250
    30 MARTIN          1250
    30 TURNER          1500
    30 ALLEN           1600
    30 BLAKE           2850
    30                 9400
                      29025

18 sor kijelölve.

Halmozott részlegösszegek kiíratása:
DEPTNO ENAME      SAL NormálRangsor TömörRangsor RészlegreHalmozottÖsszeg
------ ------- ----- ------------- ------------ ------------------------
    10 MILLER   1300             1            1                     1300
    10 CLARK    2450             2            2                     3750
    10 KING     5000             3            3                     8750
    20 SMITH     800             1            1                      800
    20 ADAMS    1100             2            2                     1900
    20 JONES    2975             3            3                     4875
```

```
20 FORD      3000          4          4         10875
20 SCOTT     3000          4          4         10875
30 JAMES      950          1          1           950
30 MARTIN    1250          2          2          3450
30 WARD      1250          2          2          3450
30 TURNER    1500          4          3          4950
30 ALLEN     1600          5          4          6550
30 BLAKE     2850          6          5          9400

14 sor kijelölve.

Halmozott összegek kiíratása a teljes vállalatra:
ENAME      SAL NormálRangsor TömörRangsor HalmozottÖsszeg
------- ----- ------------- ------------ ---------------
SMITH      800             1            1             800
JAMES      950             2            2            1750
ADAMS     1100             3            3            2850
MARTIN    1250             4            4            5350
WARD      1250             4            4            5350
MILLER    1300             6            5            6650
TURNER    1500             7            6            8150
ALLEN     1600             8            7            9750
CLARK     2450             9            8           12200
BLAKE     2850            10            9           15050
JONES     2975            11           10           18025
FORD      3000            12           11           24025
SCOTT     3000            12           11           24025
KING      5000            14           12           29025

14 sor kijelölve.
```

Statisztikai függvények

A hagyományos statisztikai függvényeknek (SUM, AVG, STDDEV, VARIANCE, MIN, MAX és COUNT) szintén létezik analitikus alakja, melyeket a következő szintaktika szerint használhatunk:

függvénynév([DISTINCT | ALL] *paraméter*) OVER (*analitikus_záradék*)

ahol láthatóan újat az analitikus függvények eddigi definíciójához képest a DISTINCT és az ALL opciók megjelenése jelent. E kulcsszavak a korábbi (szelekciós listabeli, illetve a csoportfüggvényeknél említett) jelentésükhöz hasonló módon azt eredményezik, hogy a függvény csak a munkatábla különböző, illetve minden paraméterértékét veszik figyelembe a függvényérték generálásakor (az ALL az alapértelmezett opció).

13.15. példa

Határozza meg a fizetésük szerint növekvően rendezett dolgozók fizetésének folyamatos halmozott összegét, az első sortól az utolsóig. (Ezt kumulált összegnek is nevezzük.)

Megoldás

```
SELECT empno, ename, sal,
       SUM(sal) OVER
         (ORDER BY sal
            RANGE UNBOUNDED PRECEDING)
              AS "Kumulált",
       SUM(sal) OVER
         (ORDER BY sal
            RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
              AS "Ugyanaz"
  FROM emp;
```

Eredmény

EMPNO	ENAME	SAL	Kumulált	Ugyanaz
7369	SMITH	800	800	800
7900	JAMES	950	1750	1750
7876	ADAMS	1100	2850	2850
7654	MARTIN	1250	5350	5350
7521	WARD	1250	5350	5350
7934	MILLER	1300	6650	6650
7844	TURNER	1500	8150	8150
7499	ALLEN	1600	9750	9750
7782	CLARK	2450	12200	12200
7698	BLAKE	2850	15050	15050
7566	JONES	2975	18025	18025
7902	FORD	3000	24025	24025
7788	SCOTT	3000	24025	24025
7839	KING	5000	29025	29025

14 sor kijelölve.

Megjegyzés

- Az eredmény első sorában az első dolgozó fizetése (mint halmozott összeg), a másodikban az első és a második fizetés összege és így tovább. Az utolsó sorban tehát az összes dolgozó összfizetése található.
- Az azonos értékeket egyszerre adja össze, ha közvetlenül egymás mellett állnak (*lásd* WARD és MARTIN sorait).

13.16. példa

Határozza meg a névsor szerint növekvően rendezett dolgozók fizetéseinek részlegükön belüli halmozott (kumulált) összegét.

Megoldás

```
SELECT deptno, ename, sal,
       SUM(sal) OVER
         (PARTITION BY deptno  ORDER BY ename
            RANGE UNBOUNDED PRECEDING)
              AS "KumuláltÖsszeg",
       SUM(sal) OVER
         (PARTITION BY deptno  ORDER BY ename
            RANGE BETWEEN UNBOUNDED PRECEDING  AND  CURRENT ROW)
              AS "Ugyanaz"
  FROM emp;
```

Eredmény

```
    DEPTNO ENAME             SAL KumuláltÖsszeg    Ugyanaz
---------- ---------- ---------- -------------- ----------
        10 CLARK            2450           2450       2450
        10 KING             5000           7450       7450
        10 MILLER           1300           8750       8750
        20 ADAMS            1100           1100       1100
        20 FORD             3000           4100       4100
        20 JONES            2975           7075       7075
        20 SCOTT            3000          10075      10075
        20 SMITH             800          10875      10875
        30 ALLEN            1600           1600       1600
        30 BLAKE            2850           4450       4450
        30 JAMES             950           5400       5400
        30 MARTIN           1250           6650       6650
        30 TURNER           1500           8150       8150
        30 WARD             1250           9400       9400

14 sor kijelölve.
```

13.17. példa

Határozza meg a fizetésük szerint rendezett dolgozókhoz az olyan mozgó „ablakbeli" halmozott összeget, mely végigmegy e rendezett tábla teljes tartományán, és az aktuális sor előtti két sort és az aktuális utáni egy sort tartalmazza (mozgó ablakban képzett összeg). A lista tartalmazza a kumulált összeget is a dolgozók fenti sorrendjében.

Megoldás

```
SELECT ename, sal,
       SUM(sal) OVER
         (ORDER BY sal ASC
            ROWS BETWEEN 2 PRECEDING AND 1 FOLLOWING)
              AS "AblakÖsszeg",
       SUM(sal) OVER
         (ORDER BY sal ASC)
              AS "KumuláltÖsszeg"
  FROM emp;
```

Eredmény

```
ENAME             SAL AblakÖsszeg KumuláltÖsszeg
---------- ---------- ----------- ---------------
SMITH             800        1750             800
JAMES             950        2850            1750
ADAMS            1100        4100            2850
MARTIN           1250        4550            5350
WARD             1250        4900            5350
MILLER           1300        5300            6650
TURNER           1500        5650            8150
ALLEN            1600        6850            9750
CLARK            2450        8400           12200
BLAKE            2850        9875           15050
JONES            2975       11275           18025
FORD             3000       11825           24025
SCOTT            3000       13975           24025
KING             5000       11000           29025

14 sor kijelölve.
```

Megjegyzés

Tekintsük aktuális sorként (`CURRENT ROW`) például a `Jones` sorát, fizetése `2975`. Az ablakot úgy adtuk meg a lekérdezésben, hogy az aktuális sor előtt kettő, utána pedig egy sor tartozzon bele. Az aktuális sor előtt lévő két fizetés; `Clark`-é `2450` és `Blake`-é `2850`, valamint az utána következő, ez `Ford`-é, mely `3000`. Összeadva e fizetésértékeket (`2975 + 2450 + 2850 + 3000 = 11275`), valóban az aktuális sor `AblakÖsszeg`-beli értéket kaptuk.

RATIO_TO_REPORT függvény

A `RATIO_TO_REPORT` egy speciális statisztikai függvény, kiszámítja az érték és a csoportösszeg közötti az arányt. A csoport itt is lehet az egész tábla. Használata a következő szintaktika szerint történhet:

```
RATIO_TO_REPORT( kifejezés ) OVER ( PartícióTag )
```

ahol tehát az analitikus záradékból csak a partíció-tag szerepelhet.

13.18. példa

Listázza ki a dolgozók azonosítóját, nevét, fizetését, valamint azt az értéket, mely megadja, milyen arányban részesülnek az összfizetésből.

1. megoldás

```
SELECT empno, ename, sal,
       RATIO_TO_REPORT(sal) OVER()
         AS részesedés
   FROM emp ORDER BY sal desc;
```

Eredmény

EMPNO	ENAME	SAL	RÉSZESEDÉS
7839	KING	5000	.172265289
7788	SCOTT	3000	.103359173
7902	FORD	3000	.103359173
7566	JONES	2975	.102497847
7698	BLAKE	2850	.098191214
7782	CLARK	2450	.084409991
7499	ALLEN	1600	.055124892
7844	TURNER	1500	.051679587
7934	MILLER	1300	.044788975
7521	WARD	1250	.043066322
7654	MARTIN	1250	.043066322
7876	ADAMS	1100	.037898363
7900	JAMES	950	.032730405
7369	SMITH	800	.027562446

2. megoldás (Listázás formátum megadásával)

```
SELECT empno, ename, sal,
       TO_CHAR(RATIO_TO_REPORT(sal) OVER(),'9990.999')
         AS részarány
  FROM emp
  ORDER BY sal DESC;
```

Eredmény

EMPNO	ENAME	SAL	RÉSZARÁNY
7839	KING	5000	0.172
7788	SCOTT	3000	0.103

```
      7902 FORD            3000      0.103
      7566 JONES           2975      0.102
      7698 BLAKE           2850      0.098
      7782 CLARK           2450      0.084
      7499 ALLEN           1600      0.055
      7844 TURNER          1500      0.052
      7934 MILLER          1300      0.045
      7521 WARD            1250      0.043
      7654 MARTIN          1250      0.043
      7876 ADAMS           1100      0.038
      7900 JAMES            950      0.033
      7369 SMITH            800      0.028
```

13.19. példa

Listázza ki a dolgozók munkakörét, nevét, fizetését, valamint azt az értéket, mely megadja, milyen arányban részesülnek munkakörük összfizetéséből.

Megoldás

```
SELECT job, ename, sal,
       TO_CHAR(RATIO_TO_REPORT(sal) OVER
         (PARTITION BY job),'99999990.999')
            AS "munkakörarány"
  FROM emp;
```

Eredmény

```
JOB       ENAME      SAL munkakörarány
--------- ---------- ---------- -------------
ANALYST   SCOTT            3000         0.500
ANALYST   FORD             3000         0.500
CLERK     SMITH             800         0.193
CLERK     ADAMS            1100         0.265
CLERK     MILLER           1300         0.313
CLERK     JAMES             950         0.229
MANAGER   JONES            2975         0.360
MANAGER   CLARK            2450         0.296
MANAGER   BLAKE            2850         0.344
PRESIDENT KING             5000         1.000
SALESMAN  ALLEN            1600         0.286
SALESMAN  MARTIN           1250         0.223
SALESMAN  TURNER           1500         0.268
SALESMAN  WARD             1250         0.223

14 sor kijelölve.
```

SPECIÁLIS SZÉLSŐÉRTÉK-FÜGGVÉNYEK

A FIRST_VALUE, illetve LAST_VALUE szélsőértékek meghatározására alkalmas analitikus függvények. Tulajdonképpen speciális statisztikai függvényeknek tekinthetjük őket, melyek képesek

- nem feltétlenül numerikus értékeken is szélsőértékek meghatározására (erre még a MIN és MAX is képes), továbbá
- valamely oszlop szerint rendezett ablaktartomány valamely (esetleg másik) oszlopa első és utolsó elemének meghatározására.

Ezeket a függvényeket a következő szintaktika szerint használhatjuk:

függvénynév(*kifejezés*) OVER (*analitikus utasításrész*)

Ez a két analitikus függvény nem ágyazható be más analitikus függvények kifejezés részébe.

13.20. példa

Listázza ki dolgozónként a nevüket, a fizetésüket, a részlegazonosítójukat, valamint részlegük legkisebb és legnagyobb fizetését, elsődlegesen a részlegazonosító, másodlagosan a fizetés szerint rendezve.

1. megoldás (A korábbi ismereteink alapján, inline nézettel)

```
SELECT ename, sal, al.*
  FROM emp,
       (SELECT deptno,
               MIN(sal)  AS legkisebb,
               MAX(sal)  AS legnagyobb
          FROM emp
          GROUP BY deptno)  al
  WHERE emp.deptno = al.deptno
  ORDER BY al.deptno, sal;
```

2. megoldás (A FIRST_VALUE és LAST_VALUE függvényekkel)

```
SELECT ename, sal, deptno,
       FIRST_VALUE(sal) OVER
         (PARTITION BY deptno
            ORDER BY sal ASC
            RANGE BETWEEN UNBOUNDED PRECEDING AND
                          UNBOUNDED FOLLOWING)
                               AS legkisebb,
       LAST_VALUE(sal) OVER
         (PARTITION BY deptno
            ORDER BY sal ASC
            RANGE BETWEEN UNBOUNDED PRECEDING AND
```

```
                        UNBOUNDED FOLLOWING)
                            AS legnagyobb
  FROM emp;
```

Eredmény (mindkét esetben)

```
ENAME             SAL     DEPTNO  LEGKISEBB LEGNAGYOBB
---------- ---------- ---------- ---------- ----------
MILLER            1300         10       1300       5000
CLARK             2450         10       1300       5000
KING              5000         10       1300       5000
SMITH              800         20        800       3000
ADAMS             1100         20        800       3000
JONES             2975         20        800       3000
SCOTT             3000         20        800       3000
FORD              3000         20        800       3000
JAMES              950         30        950       2850
WARD              1250         30        950       2850
MARTIN            1250         30        950       2850
TURNER            1500         30        950       2850
ALLEN             1600         30        950       2850
BLAKE             2850         30        950       2850

14 sor kijelölve.
```

Megjegyzés

Ha a fenti megoldásban fordított irányban rendezünk (`ASC` helyett `DESC` kulcsszót írva), akkor természetesen a `FIRST_VALUE`, illetve a `LAST_VALUE` függvények a legnagyobb, illetve legkisebb értékeket adják.

13.21. példa

Listázza ki dolgozónként a nevüket, a fizetésüket, a részlegazonosítójukat, valamint részlegüknek az ábécé szerinti első és utolsó dolgozója *nevét* elsődlegesen a részlegazonosító, másodlagosan a név szerint rendezve.

1. megoldás (A korábbi ismereteink alapján, inline nézettel)

```
SELECT ename, sal, a1.*
  FROM emp,
       (SELECT deptno,
               MIN(ename)  AS legkisebb,
               MAX(ename)  AS legnagyobb
          FROM emp
          GROUP BY deptno)  a1
  WHERE emp.deptno = a1.deptno
  ORDER BY a1.deptno, ename;
```

2. megoldás (A FIRST_VALUE és LAST_VALUE függvényekkel)

```
SELECT ename, sal, deptno,
       FIRST_VALUE(ename) OVER
        (PARTITION BY deptno
           ORDER BY ename ASC
           RANGE BETWEEN UNBOUNDED PRECEDING AND
                         UNBOUNDED FOLLOWING)
                              AS Első,
      LAST_VALUE(ename) OVER
        (PARTITION BY deptno
           ORDER BY ename ASC
           RANGE BETWEEN UNBOUNDED PRECEDING AND
                         UNBOUNDED FOLLOWING)
                              AS Utolsó
  FROM emp;
```

Eredmény (mindkét esetben)

```
ENAME             SAL     DEPTNO ELSŐ       UTOLSÓ
---------- ---------- ---------- ---------- ----------
CLARK            2450         10 CLARK      MILLER
KING             5000         10 CLARK      MILLER
MILLER           1300         10 CLARK      MILLER
ADAMS            1100         20 ADAMS      SMITH
FORD             3000         20 ADAMS      SMITH
JONES            2975         20 ADAMS      SMITH
SCOTT            3000         20 ADAMS      SMITH
SMITH             800         20 ADAMS      SMITH
ALLEN            1600         30 ALLEN      WARD
BLAKE            2850         30 ALLEN      WARD
JAMES             950         30 ALLEN      WARD
MARTIN           1250         30 ALLEN      WARD
TURNER           1500         30 ALLEN      WARD
WARD             1250         30 ALLEN      WARD

14 sor kijelölve.
```

13.22. példa

Listázza ki dolgozónként a nevüket, a fizetésüket, a részlegazonosítójukat, valamint részlegüknek az ábécé szerinti első és utolsó dolgozója *fizetését* elsődlegesen a részlegazonosító, másodlagosan a név szerint rendezve.

Megoldás

```
SELECT ename, sal, deptno,
       FIRST_VALUE(sal) OVER
        (PARTITION BY deptno
           ORDER BY ename ASC
           RANGE BETWEEN UNBOUNDED PRECEDING AND
                         UNBOUNDED FOLLOWING)
                               AS Első,
      LAST_VALUE(sal) OVER
        (PARTITION BY deptno
           ORDER BY ename ASC
           RANGE BETWEEN UNBOUNDED PRECEDING AND
                         UNBOUNDED FOLLOWING)
                               AS Utolsó
  FROM emp;
```

Eredmény

ENAME	SAL	DEPTNO	ELSŐ	UTOLSÓ
CLARK	2450	10	2450	1300
KING	5000	10	2450	1300
MILLER	1300	10	2450	1300
ADAMS	1100	20	1100	800
FORD	3000	20	1100	800
JONES	2975	20	1100	800
SCOTT	3000	20	1100	800
SMITH	800	20	1100	800
ALLEN	1600	30	1600	1250
BLAKE	2850	30	1600	1250
JAMES	950	30	1600	1250
MARTIN	1250	30	1600	1250
TURNER	1500	30	1600	1250
WARD	1250	30	1600	1250

14 sor kijelölve.

Megjegyzés

Ennek a feladatnak a hagyományos eszközökkel való megoldása már igen bonyolult lett volna.

Hisztogramfüggvények

WIDTH_BUCKET FÜGGVÉNY

E függvény segítségével „azonos szélességű" hisztogram készíthető, azaz meghatározza, hogy azonos méretű értéktartományokban az oszlopkifejezés értékei hogyan oszlanak meg (vagyis egy-egy érték melyik tartományba kerül). Használatának szintaktikája:

```
WIDTH_BUCKET(oszlopkifejezés, alsóhatár, felsőhatár, felosztások száma)
```

ahol a függvény minden sorhoz meghatározza annak az értéktartománynak a sorszámát, melybe e sor *oszlopkifejezése* tartozik. Az egyes értéktartományok az *oszlopkifejezés alsó- és felsőhatárai* között azonos méretűek, sorszámuk értékük szerint növekvő. Számukat a *felosztások száma* határozza meg. Ha az *oszlopkifejezés* értéktartománya beleesik az *alsó- és felsőhatárok* közötti tartományba, akkor e szám a *felosztások száma*, de létezhet nulladik és (*felosztások száma* +1)-edik tartomány is, melyekkel az alul- és túlcsordulások kezelhetők.

Megjegyezzük, hogy e függvény tulajdonképpen nem is analitikus függvény (hiányzik az OVER opció), hanem csoportfüggvény. A hisztogramfüggvények körében bemutatott másik függvény, az NTILE (*lásd* alább) azonban már az, hasonló funkciójuk miatt célszerű őket együtt tárgyalni.

13.23. példa

Listázza ki a dolgozók nevét, fizetését és a fizetési kategóriáját, ahol ez utóbbit úgy kapja meg, hogy három egyenlő részre osztja a 800..5001 értéktartományt.

Megoldás

```
SELECT ename, sal,
       WIDTH_BUCKET(sal,800,5001,3)  AS Kategória
  FROM emp
 ORDER BY Kategória;
```

Eredmény

```
ENAME             SAL  KATEGÓRIA
---------- ---------- ----------
MARTIN            1250          1
ALLEN             1600          1
JAMES              950          1
SMITH              800          1
ADAMS             1100          1
WARD              1250          1
TURNER            1500          1
MILLER            1300          1
BLAKE             2850          2
CLARK             2450          2
```

```
JONES           2975          2
SCOTT           3000          2
FORD            3000          2
KING            5000          3

14 sor kijelölve.
```

13.24. példa

Oldja meg az előző feladatot, ám most az 1000..4500 tartomány 3 egyenlő részre osztásával.

Megoldás

```
SELECT ename, sal,
       WIDTH_BUCKET(sal,1000,4500,3)  AS Kategória
  FROM emp
  ORDER BY Kategória;
```

Eredmény

```
ENAME             SAL  KATEGÓRIA
---------- ---------- ----------
SMITH             800          0
JAMES             950          0
ALLEN            1600          1
WARD             1250          1
ADAMS            1100          1
TURNER           1500          1
MARTIN           1250          1
MILLER           1300          1
JONES            2975          2
BLAKE            2850          2
CLARK            2450          2
FORD             3000          2
SCOTT            3000          2
KING             5000          4

14 sor kijelölve.
```

Megjegyzés

Láthatóan a `Smith` és a `James` lefelé, `King` pedig felfelé „lóg ki" a kijelölt (1000..4500) tartományból.

NTILE FÜGGVÉNY

Az NTILE függvény a WIDTH_BUCKET függvényhez hasonlóan hisztogram készítésére használható, ám ez a hisztogram „azonos magasságú, különböző szélességű" (vagyis a különböző szélességű értéktartományok azonos számú sort tartalmaznak). Használata a következő szintaktika szerint történhet:

```
NTILE(kifejezés) OVER ( [PartícióTag] RendezőTag )
```

ahol a függvény hatására két esemény következik be. Egyrészt történik egy rendezés a rendező tag szerint, másrészt az így rendezett lista sorait a kifejezés értékének megfelelő számú (lehetőleg azonos mennyiségű sort tartalmazó) résztartományra bontja, és ezeket 1-től kezdődő sorszámmal látja el. A függvény minden sorhoz hozzárendeli az ezt a sort tartalmazó résztartomány sorszámát.

13.25. példa

Listázza ki a fizetésük szerint rendezett dolgozók nevét, fizetését és fizetési kategóriáját, ahol ez utóbbit az összes dolgozó fizetési tartományának három részre osztásával úgy kapja meg, hogy minden részbe közel azonos számú dolgozó kerüljön. Az egyes kategóriák sorszáma a fizetések szerint legyen növekvő.

Megoldás

```
SELECT ename, sal,
       NTILE(3) OVER (ORDER BY sal)  AS Kategória
  FROM emp;
```

Eredmény

```
ENAME             SAL  KATEGÓRIA
---------- ---------- ----------
SMITH             800          1
JAMES             950          1
ADAMS            1100          1
MARTIN           1250          1
WARD             1250          1
MILLER           1300          2
TURNER           1500          2
ALLEN            1600          2
CLARK            2450          2
BLAKE            2850          2
JONES            2975          3
FORD             3000          3
SCOTT            3000          3
KING             5000          3

14 sor kijelölve.
```

13.26. példa

Listázza ki a fizetésük szerint az előző feladatbeli módon kategorizált dolgozók nevét, részlegazonosítóját, fizetését és fizetési kategóriáját elsődlegesen a részlegazonosító, másodlagosan pedig a kategória szerint rendezve.

Megoldás

```
SELECT *
  FROM (SELECT ename, deptno, sal,
               NTILE(3) OVER (ORDER BY sal)  AS Kategória
          FROM emp)
  ORDER BY deptno, Kategória;
```

Eredmény

```
ENAME          DEPTNO        SAL  KATEGÓRIA
---------- ---------- ---------- ----------
MILLER             10       1300          2
CLARK              10       2450          2
KING               10       5000          3
SMITH              20        800          1
ADAMS              20       1100          1
JONES              20       2975          3
FORD               20       3000          3
SCOTT              20       3000          3
JAMES              30        950          1
MARTIN             30       1250          1
WARD               30       1250          1
TURNER             30       1500          2
BLAKE              30       2850          2
ALLEN              30       1600          2

14 sor kijelölve.
```

13.27. példa

Kategorizáljuk a dolgozókat a 13.25. példabelihez hasonló módon, de most a kategorizálási tartomány ne az összes, hanem csak az azonos munkakörű dolgozók fizetése által legyen kijelölve. A lista a munkakört, a nevet, a fizetést és a kategóriát tartalmazza.

Megoldás

```
SELECT job, ename, sal,
       NTILE(3) OVER (PARTITION BY job ORDER BY sal)  AS Kategória
  FROM emp;
```

Eredmény

```
JOB       ENAME      SAL  KATEGÓRIA
--------- ---------- ---------- ----------
ANALYST   FORD         3000          1
ANALYST   SCOTT        3000          2
CLERK     SMITH         800          1
CLERK     JAMES         950          1
CLERK     ADAMS        1100          2
CLERK     MILLER       1300          3
MANAGER   CLARK        2450          1
MANAGER   BLAKE        2850          2
MANAGER   JONES        2975          3
PRESIDENT KING         5000          1
SALESMAN  MARTIN       1250          1
SALESMAN  WARD         1250          1
SALESMAN  TURNER       1500          2
SALESMAN  ALLEN        1600          3

14 sor kijelölve.
```

Megjegyzés

A megadott három kategóriához minden munkakörben három csoportot hozott létre, kivéve azokat a munkaköröket, melyekben nem volt legalább három dolgozó.