9. Funcții SQL specifice pentru rapoarte

Cuprins

9. Funcții SQL specifice pentru grupare și clasare

- Clauza *GROUP BY* permite gruparea înregistrărilor selectate după valorile expresiilor precizate în aceasta. Pentru fiecare grup, va fi întoarsă o singură linie de informație.
- Clauza *GROUP BY* poate produce grupări super-agregat utilizând extensiile *CUBE* sau *ROLLUP*.
- Extensia *ROLLUP* grupează înregistrările selectate pe baza valorilor primelor *n*, *n*-1, ..., 0 expresii din clauza *GROUP BY* şi întoarce o singură linie pentru fiecare grup.
  - *ROLLUP* creează grupări prin deplasarea într-o singură direcție, de la dreapta la stânga, de-a lungul listei de coloane specificate în clauza *GROUP BY*. Apoi, se aplică funcția agregat acestor grupări. Dacă sunt specificate *n* expresii în operatorul *ROLLUP*, numărul de grupări generate va fi *n*+1.
  - *GROUP BY ROLLUP* (*expr_1*, *expr_2*, …, *expr_n*) generează *n+1* tipuri de linii, corespunzătoare următoarelor grupări:

    ```
    GROUP BY (expr_1, expr_2, …, expr_n-1, expr_n)
    GROUP BY (expr_1, expr_2, …, expr_n-1)
    …
    GROUP BY (expr_1, expr_2)
    GROUP BY (expr_1)
    GROUP BY ()
    ```

    – corespunzător absenței clauzei GROUP BY şi deci, calculului funcțiilor grup din cerere pentru întreg tabelul.
- *CUBE* grupează liniile selectate pe baza valorilor tuturor combinațiilor posibile ale expresiilor specificate şi întoarce câte o linie totalizatoare pentru fiecare grup.
  - Acest operator este folosit pentru a produce mulțimi de rezultate care sunt utilizate în rapoarte. În vreme ce *ROLLUP* produce subtotalurile doar pentru o parte dintre combinațiile posibile, operatorul *CUBE* produce subtotaluri pentru toate combinațiile posibile de grupări specificate în clauza *GROUP BY*, precum şi un total general.
  - Dacă există *n* coloane sau expresii în clauza *GROUP BY*, vor exista 2^n combinații posibile superagregat.

9.1. Funcții de grupare

- Atunci când în clauza *GROUP BY* sunt utilizate extensiile *ROLLUP* și *CUBE*, apar două probleme.
  - În primul rând, cum putem determina, prin cod, care dintre înregistrările întoarse ca rezultat sunt subtotaluri și cum putem determina nivelul de agregare pentru un anumit subtotal.
    - Destul de des se întâmplă să fie nevoie de subtotaluri în calcule cum ar fi „procentaj din total“, deci este necesară determinarea înregistrărilor care sunt subtotaluri.
  - În al doilea rând, ce se întâmplă dacă rezultatele întoarse conțin atât valori *null* stocate în tabele, cât și valori *null* întoarse de operatorii *ROLLUP* sau *CUBE*? Cum facem diferența între ele?

9.1.1. Funcția GROUPING

- Această funcție poate rezolva cele două probleme enunțate anterior.
- Folosind o singură coloană ca argument, funcția *GROUPING* întoarce:
  - valoarea 1 dacă valoarea *null* este generată de operatorii *ROLLUP* sau *CUBE*;
  - valoarea 0 pentru celelalte valori (inclusiv valoarea *null*) stocate în tabele.

Exemplul 9.1

În acest exemplu funcția *GROUPING* este utilizată pentru a crea un set de coloane utile pentru identificarea subtotalurilor.

```
SELECT categorie_1 raion, luna, oras,
       SUM(pret_unitar_vanzare*cantitate) valoare,
       GROUPING(categorie_1) as GC,
       GROUPING(luna) AS GL,
       GROUPING(oras) AS GO
FROM   vanzari v, regiuni r, produse p, timp t
WHERE  v.regiune_id=r.id_regiune
AND    v.produs_id=p.id_produs
AND    v.timp_id=t.id_timp
AND    oras IN ('Bucuresti', 'Bacau')
AND    luna IN (3, 4)
AND    an = 2007
GROUP BY ROLLUP(categorie_1, luna, oras);
```

```
RAION            LUNA    ORAS          VALOARE         GC         GL         GO
-------------------------------------------------------------------------------
IT                  3    Bucuresti      273.32          0          0          0
IT                  3                   273.32          0          0          1
IT                  4    Bucuresti     3741.09          0          0          0
IT                  4                  3741.09          0          0          1
IT                                     4014.41          0          1          1
Papetarie           3    Bacau          242.05          0          0          0
Papetarie           3    Bucuresti    22342.68          0          0          0
Papetarie           3                 22584.73          0          0          1
Papetarie           4    Bucuresti      458.24          0          0          0
Papetarie           4                   458.24          0          0          1
Papetarie                              23042.97          0          1          1
                                       27057.38          1          1          1
```

Un program poate identifica cu ușurință înregistrarea detaliată după secvența „0 0 0“, subtotalul pentru primul nivel prin secvența „0 0 1“, pentru al doilea nivel prin „0 1 1“, iar totalul general prin secvența „1 1 1“.

Exemplul 9.2

Pentru reprezentare mai bună a rezultatelor se poate utiliza funcția *DECODE*.

```
SELECT DECODE(GROUPING(categorie_1),1,'Toate categoriile',
              categorie_1) AS raion,
       DECODE(GROUPING(oras),1,'Toate orasele',oras)
              AS oras,
       SUM(pret_unitar_vanzare*cantitate) valoare
FROM   vanzari v, regiuni r, produse p
WHERE  v.regiune_id=r.id_regiune
AND    v.produs_id=p.id_produs
AND oras IN ('Bucuresti', 'Bacau')
GROUP BY CUBE(categorie_1,oras);
```

```
RAION                ORAS                VALOARE
-----------------------------------------------------
IT                   Bucuresti            6504.9
IT                   Toate orasele        6504.9
Papetarie            Bacau                 242.05
Papetarie            Bucuresti           23368.47
Papetarie            Toate orasele       23610.52
Toate categoriile    Bacau                 242.05
Toate categoriile    Bucuresti           29873.37
Toate categoriile    Toate orasele       30115.42
```

Exemplul 9.3

Funcția *GROUPING* nu este utilă doar pentru a identifica valorilor *null,* ci și pentru sortarea subtotalurilor și filtrarea rezultatelor. În exemplul următor rezultatul obținut constă dintr-o parte a subtotalurilor generate de operatorul *CUBE*. Clauza *HAVING* setează constrângeri pentru coloanele care folosesc funcția *GROUPING*.

```
SELECT categorie_1 raion, oras,
       SUM(pret_unitar_vanzare*cantitate) valoare,
       GROUPING(categorie_1) as GC,
       GROUPING(luna) AS GL,
       GROUPING(oras) AS GO
FROM   vanzari v, regiuni r, produse p, timp t
WHERE  v.regiune_id=r.id_regiune
AND    v.produs_id=p.id_produs
AND    v.timp_id=t.id_timp
AND    oras IN ('Bucuresti', 'Bacau')
AND    luna IN (3,4)
AND    an = 2007
GROUP BY CUBE(categorie_1,luna, oras)
HAVING  (GROUPING(categorie_1)=1 AND
         GROUPING(luna)= 1 AND
         GROUPING(oras)=1)
         OR
         (GROUPING(categorie_1)=1 AND GROUPING(luna)= 1)
         OR
         (GROUPING(oras)=1 AND GROUPING(luna)= 1);
```

```
RAION          ORAS            VALOARE         GC         GL         GO
-------------------------------------------------------------------------
IT                            10338.17          0          1          1
Papetarie                     72097.22          0          1          1
               Bacau            672.61          1          1          0
               Bucuresti      81762.78          1          1          0
                              82435.39          1          1          1
```

9.1.2. Funcția GROUPING_ID

- Pentru a găsi nivelul de agregare pentru o anumită înregistrare, o interogare trebuie să obțină informații prin intermediul funcției *GROUPING* pentru fiecare coloană a clauzei *GROUP BY*.
  - De exemplu, o clauză *GROUP BY* cu 4 expresii ar trebui analizată împreună cu 4 funcții *GROUPING*. Aceasta nu este o soluție utilă, deoarece numărul de coloane necesar interogării se mărește.

- Pentru a rezolva această problemă se poate utiliza funcția *GROUPING_ID*.
  - Această funcție întoarce un singur număr care reprezintă nivelul de agregare. Funcția *GROUPING_ID* ia fiecare secvența de 1 și 0 care ar fi generate dacă am folosi funcții *GROUPING* pentru coloanele implicate în grupare și le concatenează, formând un vector. Acesta este tratat ca un număr binar, iar numărul echivalent în baza 10 este întors de funcția *GROUPING_ID*.
  - De exemplu, dacă este utilizată gruparea *CUBE*(a, b), valorile posibile sunt următoarele:

Nivel de agregare	Vector de biți	GROUPING_ID
a, b	0 0	0
a	0 1	1
b	1 0	2
Total general	1 1	3

Exemplul 9.4

```
SELECT categorie_1 raion, categorie_2 grupa,
       SUM(pret_unitar_vanzare*cantitate) valoare,
       GROUPING_ID(categorie_1, categorie_2) grouping_id
FROM   vanzari v,  produse p, timp t
WHERE  v.produs_id=p.id_produs
AND    v.timp_id=t.id_timp
GROUP BY CUBE (categorie_1, categorie_2)
ORDER BY categorie_1;
```

```
RAION        GRUPA                                VALOARE  GROUPING_ID
-------------------------------------------------------------------------
IT           Accesorii IT                          4526.58           0
IT           Audio-Video                           9105.15           0
IT           Computere și Laptop                   2492.04           0
IT           Consumabile                           2645.59           0
IT           Foto-Video                            3168.98           0
IT           Periferice                            1918.08           0
IT           Tehnica de birou                      3041.72           0
IT                                                26898.14           1
Papetarie    Accesorii pentru birou               13056.66           0
Papetarie    Ambalare și marcare, diverse          1169.84           0
Papetarie    Articole din hartie                    3749.7           0
Papetarie    Articole protocol și igiena           4529.94           0
Papetarie    Comunicare și prezentare              4945.19           0
Papetarie    Hartie de copiator                 2 6636.78           0
Papetarie    Instrumente de scris și corectura     7742.66           0
Papetarie    Organizare și arhivare               22803.83           0
```

```
Papetarie     Scolare                              229.37          0
Papetarie                                        94863.97          1
              Accesorii IT                        4526.58          2
              Accesorii pentru birou             13056.66          2
              Ambalare și marcare, diverse        1169.84          2
              Articole din hartie                13749.7           2
              Articole protocol și igiena         4529.94          2
              Audio-Video                         9105.15          2
              Computere și Laptop                 2492.04          2
              Comunicare și prezentare            4945.19          2
              Consumabile                         2645.59          2
              Foto-Video                          3168.98          2
              Hartie de copiator                 26636.78          2
              Instrumente de scris și corectura   7742.66          2
              Organizare și arhivare             22803.83          2
              Periferice                          1918.08          2
              Scolare                              229.37          2
              Tehnica de birou                    3041.72          2
                                                121762.11          3
```

9.1.3. Funcția GROUP_ID

- Deși extensiile clauzei *GROUP BY* sunt puternice și flexibile, acestea pot întoarce rezultate complexe care să includă și grupări dublate.
- Funcția *GROUP_ID* permite identificarea acestor duplicate. Dacă sunt mai multe seturi de înregistrări calculate pentru un nivel dat, funcția *GROUP_ID* atribuie valoarea 0 pentru toate înregistrările din primul grup. Toate celelalte grupuri vor avea alocate valori mai mari decât 0, pornind de la 1.

Exemplul 9.5

Următorul exemplu conține o interogare care generează agregări duplicate.

```
SELECT categorie_1 raion, categorie_2 grupa,
       SUM(pret_unitar_vanzare*cantitate) valoare,
       GROUPING_ID(categorie_1,categorie_2) grouping_id,
       GROUP_ID() AS group_id
FROM   vanzari v, produse p, timp t
WHERE  v.produs_id=p.id_produs
AND    v.timp_id=t.id_timp
GROUP BY GROUPING SETS (categorie_1,
                        ROLLUP(categorie_1, categorie_2))
ORDER BY categorie_1, categorie_2;
```

```
RAION        GRUPA                        VALOARE  GROUPING_ID   GROUP_ID
-------------------------------------------------------------------------
IT           Accesorii IT                  4526.58           0          0
IT           Audio-Video                   9105.15           0          0
```

```
IT            Computere și Laptop            2492.04          0          0
IT            Consumabile                    2645.59          0          0
IT            Foto-Video                     3168.98          0          0
IT            Periferice                     1918.08          0          0
IT            Tehnica de birou               3041.72          0          0
IT                                          26898.14          1          0
IT                                          26898.14          1          1
Papetarie     Accesorii pentru birou        13056.66          0          0
Papetarie     Ambalare și marcare, diverse   1169.84          0          0
Papetarie     Articole din hartie           13749.7           0          0
Papetarie     Articole protocol și igiena    4529.94          0          0
Papetarie     Comunicare și prezentare       4945.19          0          0
Papetarie     Hartie de copiator            26636.78          0          0
Papetarie     Instrumente de scris           7742.66          0          0
Papetarie     Organizare și arhivare        22803.83          0          0
Papetarie     Scolare                         229.37          0          0
Papetarie                                   94863.97          1          0
Papetarie                                   94863.97          1          1
                                           121762.11          3          0
```

Această interogare generează următoarele grupări:

- (categorie_1, categorie_2);
- (categorie_1);
- (categorie_1);
- (), corespunzătoare totalului general.

Se observă că gruparea după *categorie_1* se repetă de două ori. Utilizând funcția *GROUP_ID* se pot filtra grupările duplicate din rezultat.

Exemplul 9.6

```
SELECT categorie_1 raion, categorie_2 grupa,
       SUM(pret_unitar_vanzare*cantitate) valoare,
       GROUPING_ID(categorie_1,categorie_2) AS grouping_id
FROM   vanzari v, produse p, timp t
WHERE  v.produs_id=p.id_produs
AND    v.timp_id=t.id_timp
GROUP BY GROUPING SETS (categorie_1,
                        ROLLUP(categorie_1, categorie_2))
HAVING   GROUP_ID() =0
ORDER BY categorie_1,categorie_2;
```

```
RAION         GRUPA                          VALOARE    GROUPING_ID
-------------------------------------------------------------------
IT            Accesorii IT                    4526.58           0
IT            Audio-Video                     9105.15           0
IT            Computere și Laptop             2492.04           0
IT            Consumabile                     2645.59           0
IT            Foto-Video                      3168.98           0
```

```
IT              Periferice                      1918.08          0
IT              Tehnica de birou                3041.72          0
IT                                             26898.14          1
Papetarie       Accesorii pentru birou         13056.66          0
Papetarie       Ambalare și marcare, diverse    1169.84          0
Papetarie       Articole din hartie            13749.7           0
Papetarie       Articole protocol și igiena     4529.94          0
Papetarie       Comunicare și prezentare        4945.19          0
Papetarie       Hartie de copiator             26636.78          0
Papetarie       Instrumente de scris            7742.66          0
Papetarie       Organizare și arhivare         22803.83          0
Papetarie       Scolare                          229.37          0
Papetarie                                      94863.97          1
                                              121762.11          3
```

9.2. Funcții de clasare

- O funcție de clasare calculează numărul de ordine al unei înregistrări prin comparație cu celelalte înregistrări din mulțimea de date.

9.2.1. Funcțiile RANK și DENSE_RANK

- Funcțiile *RANK* și *DENSE_RANK* permit aflarea pozițiilor elementelor dintr-un grup de date. Aceste funcții determină pozițiile într-un clasament.
- Sintaxa funcțiilor *RANK* și *DENSE_RANK* este următoarea:

```
RANK() OVER(
      [ PARTITION BY <expr1> [, ...]]
        ORDER BY <expr2> [ASC | DESC]
         [NULLS FIRST | NULLS LAST] [, ...]
     )
DENSE_RANK() OVER(
      [ PARTITION BY <expr1> [, ...]]
        ORDER BY <expr2> [ASC | DESC]
         [NULLS FIRST | NULLS LAST] [, ...]
     )
```

- Diferența dintre funcția *RANK* și funcția *DENSE_RANK* constă în faptul că funcția *DENSE_RANK* nu lasă spații între secvențele ordonate.
  - Aceasta înseamnă că dacă s-ar ordona locurile într-o competiție folosind funcția *DENSE_RANK* și ar fi trei persoane clasate pe locul întâi, atunci următoarea persoană ar fi clasată pe locul 2. Funcția *RANK* ar avea ca rezultat cele trei persoane pe locul întâi, dar următoarea persoană ar fi clasată pe locul 4.

- Câteva dintre caracteristicile importante ale funcțiilor *RANK* și *DENSE_RANK* sunt date în continuare.
  - Ordonarea crescătoare este setată în mod implicit.
  - Clauza *PARTITION BY* determină divizarea mulțimii rezultat a interogării în grupurile pe care va opera funcția. Funcția se reinițializează ori de câte ori grupul se schimbă. De fapt, valoarea expresiilor clauzei *PARTITION BY* definesc limitele de reinițializare.
  - Dacă clauza *PARTITION BY* lipsește, atunci se va calcula ordinea peste întreaga mulțime rezultat.
  - Expresia *expr1* poate fi orice expresie validă care implică referințe de coloane, constante, agregate sau expresii ce invocă aceste elemente.
  - Clauzele *NULLS FIRST* și *NULLS LAST* stabilesc poziționarea valorilor *null* în secvența ordonată. Ordinea secvenței se realizează comparând valori *null* cu valori non-*null*. Dacă secvența este ordonată crescător, atunci clauza *NULLS FIRST* implică faptul că valoarea *null* este mai mică decât orice valoare non-*null*, iar clauza *NULLS LAST* implică faptul că valoarea *null* este mai mare decât orice valoare non-*null*.
  - Dacă ambele clauze *NULLS FIRST* și *NULLS LAST* sunt omise, atunci ordonarea valorilor *null* depinde de opțiunile *ASC* și *DESC*. Valorile *null* sunt considerate mai mari decât orice alte valori.

Exemplul 9.7

Următorul exemplu ilustrează cum se schimbă modul de ordonare/clasare, folosind opțiunea *ASC*/*DESC*.

```
SELECT categorie_2 grupa,
       SUM(pret_unitar_vanzare*cantitate) valoare,
       DENSE_RANK() OVER (ORDER BY
         SUM(pret_unitar_vanzare*cantitate) ASC) asc_rank,
       DENSE_RANK() OVER (ORDER BY
         SUM(pret_unitar_vanzare*cantitate) DESC) desc_rank
FROM  vanzari v, timp t, produse p
WHERE v.produs_id=p.id_produs
AND   v.timp_id=t.id_timp
AND   categorie_1='IT'
GROUP BY categorie_2
  ORDER BY valoare;
```

```
GRUPA                VALOARE    ASC_RANK  DESC_RANK
-------------------- ---------- ---------- ----------
Periferice             1918.08          1          7
Computere și Laptop    2492.04          2          6
Consumabile            2645.59          3          5
Tehnica de birou       3041.72          4          4
Foto-Video             3168.98          5          3
Accesorii IT           4526.58          6          2
Audio-Video            9105.15          7          1
```

Exemplul 9.8

În cazul sortării după mai multe expresii, funcțiile de clasare trebuie să determine ordinea pentru valorile egale din cadrul mulțimii. De exemplu, avem o interogare care determină valoarea vânzările pe raioane, pentru lunile martie și aprilie ale anului 2007. Funcția *TRUNC* este utilizată numai cu scopul de a crea valori egale pentru această interogare.

```
SELECT categorie_1 raion, luna,
      TRUNC(SUM(pret_unitar_vanzare*cantitate),-3) AS valoare,
      SUM(cantitate) AS nr_prod,
      DENSE_RANK() OVER (ORDER BY
          SUM(pret_unitar_vanzare*cantitate) DESC,
          SUM(cantitate) DESC) AS dense_rank
FROM  vanzari v, timp t, produse p
WHERE v.produs_id=p.id_produs
AND   v.timp_id=t.id_timp
AND   luna IN (3, 4)
AND   an = 2007
GROUP BY categorie_1,luna
ORDER BY valoare DESC, nr_prod DESC;
```

```
RAION          LUNA    VALOARE     NR_PROD   DENSE_RANK
-------------- ------  ---------  ---------  ----------
Papetarie       3       6000000     3350768    1
Papetarie       4       6000000     1612606    2
IT              4       3000000     235833     3
IT              3       3000000     221973     4
```

Exemplul 9.9

Diferența dintre funcțiile *RANK* și *DENSE_RANK* este ilustrată în exemplu următor.

```
SELECT categorie_1 raion, luna,
  TRUNC(SUM(pret_unitar_vanzare*cantitate),-6) AS valoare,
  SUM(cantitate) AS nr_prod,
  RANK() OVER (ORDER BY
        TRUNC(SUM(pret_unitar_vanzare*cantitate),-6) DESC)
        AS rank,
```

```
  DENSE_RANK() OVER (ORDER BY
        TRUNC(SUM(pret_unitar_vanzare*cantitate),-6) DESC)
        AS dense_rank
FROM  vanzari v, timp t, produse p
WHERE v.produs_id=p.id_produs
AND   v.timp_id=t.id_timp
AND   luna IN (3, 4)
AND   an = 2007
GROUP BY categorie_1,luna
ORDER BY valoare DESC, nr_prod DESC;
```

```
RAION          LUNA    VALOARE     NR_PROD   RANK  DENSE_RANK
-------------- ------  ---------  ---------  ----- ----------
Papetarie       3       6000000     3350768    1    1
Papetarie       4       6000000     1612606    1    1
IT              4       3000000     235833     3    2
IT              3       3000000     221973     3    2
```

Observam că în cazul funcție *DENSE_RANK*, cea mai mare valoare din lista ordonată este reprezentată de numărul de valori distincte din mulțimea de date.

Exemplul 9.10

Pentru a ordona produsele dintr-o categorie, în funcție de valoarea vânzărilor, se poate utiliza următoarea interogare:

```
SELECT categorie_5 categorie, denumire,
       SUM(pret_unitar_vanzare*cantitate) VALOARE,
       DENSE_RANK() OVER (PARTITION BY categorie_5
          ORDER BY SUM(pret_unitar_vanzare*cantitate) DESC)
          AS RANK
FROM  vanzari v, produse p
WHERE v.produs_id=p.id_produs
AND   categorie_5 IN ('CD-R', 'CD-RW')
GROUP BY categorie_5, denumire
ORDER BY categorie_5, valoare DESC;
```

```
CATEGORIE DENUMIRE                                          VALOARE       RANK
-------------------------------------------------------------------------------
CD-R     CD-R Verbatim 40x 800MB 90 MIN 1 buc/jewel          485.6          1
CD-R     CD-R Sony 48x 700MB 80 MIN 1 buc/slim               400.86         2
CD-R     CD-R Samsung 52x 700MB 80 MIN 1 buc/slim            153.6          3
CD-RW    CD-RW Omega 24x 700MB 80 MIN 1 buc/slim             572.5          1
CD-RW    CD-RW Traxdata 12x 70MB 80 MIN 1 buc/slim           246.6          2
CD-RW    CD-RW Samsung 210x 700MB 80 MIN 1 buc/slim           38.25         3
```

- O comandă *SELECT* poate conține mai multe funcții de clasare, fiecare partiționând datele în grupuri diferite.

Exemplul 9.11

Următoarea interogare ordonează produsele în funcție de vânzările din fiecare lună (*rank_luna*) și din fiecare categorie aleasă (*rank_categ*).

```
SELECT categorie_5 categorie, luna, denumire,
       SUM(pret_unitar_vanzare*cantitate) VALOARE,
       DENSE_RANK() OVER (PARTITION BY luna
         ORDER BY SUM(pret_unitar_vanzare*cantitate) DESC)
         AS RANK_luna,
       DENSE_RANK() OVER (PARTITION BY categorie_5
         ORDER BY SUM(pret_unitar_vanzare*cantitate) DESC)
         AS RANK_categ
FROM  vanzari v, produse p, timp t
WHERE v.produs_id=p.id_produs
AND   t.id_timp=v.timp_id
AND   an=2007
AND   categorie_5 IN ('CD-R', 'CD-RW')
GROUP BY categorie_5, luna, denumire;
```

```
CATEGORIE     LUNA   DENUMIRE                VALOARE  RANK_LUNA RANK_CATEG
-------------------------------------------------------------------------
CD-R             3   CD-R Verbatim 40x          468           2           1
CD-R             3   CD-R Sony 48x            400.86          3           2
CD-R             3   CD-R Samsung 52x          153.6          5           3
CD-R             7   CD-R Verbatim 40x          17.6          1           4
CD-RW            3   CD-RW Omega 24x           572.5          1           1
CD-RW            3   CD-RW Traxdata            246.6          4           2
CD-RW            3   CD-RW Samsung 210x         38.25         6           3
```

În acest exemplu, coloana *rank_luna* determină un clasament al produselor din punct de vedere al vânzărilor pentru luna martie cu pozițiile 1, 2, 3, 4, 5 și 6, iar pentru luna iulie poziția 1 (fiind un singur produs vândut în această lună, *CD-R Verbatim 40x*).

- Funcțiile de clasare se pot reinițializa și în funcție de grupările generate de operatorii *CUBE*, *ROLLUP* sau *GROUPING SETS*.

Exemplul 9.12

```
SELECT categorie_1 raion, luna,
       SUM(pret_unitar_vanzare*cantitate) VALOARE,
       GROUPING_ID(categorie_1, luna) grouping_id,
```

```
        DENSE_RANK() OVER (PARTITION BY
           GROUPING_ID(categorie_1, luna)
           ORDER BY SUM(pret_unitar_vanzare*cantitate) DESC)
           AS RANK_PER_GRUP
FROM  vanzari v, produse p, timp t
WHERE v.produs_id=p.id_produs
AND   v.timp_id=t.id_timp
AND   luna IN (3, 4, 5)
AND an = 2007
GROUP BY CUBE (categorie_1, luna);
```

```
RAION                LUNA   VALOARE GROUPING_ID  RANK_PER_GRUP
-------------------------------------------------------------
Papetarie            3      85702.33           0             1
IT                   4      10903.13           0             2
IT                   3      10205.74           0             3
IT                   5       4782.76           0             4
Papetarie            4       4024.22           0             5
Papetarie            5        229.31           0             6
Papetarie                   89955.86           1             1
IT                          25891.63           1             2
                     3      95908.07           2             1
                     4      14927.35           2             2
                     5       5012.07           2             3
                           115847.49           3             1
```

- Primele *n* poziții se pot obține incluzând funcția *RANK* într-o subinterogare și aplicând o condiție filtru în afara interogării.

Exemplul 9.13

Următoarea interogare obține primele cinci orașe cu vânzările cele mai mari în anul 2007:

```
SELECT *
FROM
   (SELECT  oras,
            SUM(pret_unitar_vanzare*cantitate) VALOARE,
            DENSE_RANK() OVER (ORDER BY
              SUM(pret_unitar_vanzare*cantitate) DESC)
              AS oras_RANK
    FROM    vanzari v, timp t, regiuni r
    WHERE   v.timp_id=t.id_timp
    AND     v.regiune_id=r.id_regiune
    AND     an = 2007
    GROUP BY oras)
WHERE oras_RANK <= 5;
```

```
ORAS                           VALOARE  ORAS_RANK
------------------------------ ---------- ----------
Bucuresti                       84820.82          1
Ploiesti                        11219.88          2
Pitesti                          5467.93          3
Bacau                             4271.1          4
Iasi                             3104.89          5
```

9.2.2. Funcția CUME_DIST

- Funcția *CUME_DIST* calculează poziția procentuală a unei valori relativ la o mulțime de valori.
  - Calculează distribuția cumulativă a unei valori relativ la o mulțime de valori.
- Pentru a calcula rezultatul întors de funcția *CUME_DIST* pentru o valoare *x* dintr-o mulțime *S* de dimensiune *n*, este utilizată următoarea formulă:

```
CUME_DIST(x) = (numărul valorilor din S care sunt
                înaintea lui x, inclusiv x)/ n
```

- Sintaxa funcției *CUME_DIST* este următoarea:

```
CUME_DIST() OVER
  ([PARTITION BY <expr1> [, ...]]
   ORDER BY <expr2> [ASC|DESC]
       [NULLS FIRST | NULLS LAST] [, ...])
```

- Ordonarea crescătoare este setată implicit. În acest caz, cea mai mică valoare va determina cel mai mic procent. Valorile *NULL* sunt luate în considerare atât pentru calculul numărătorului, cât și pentru calculul numitorului.
- Rezultatul întors de funcție este în intervalul (0,1]. Valoarea 1 este atinsă pentru cea mai mare valoare din partiție, în cazul ordonării ascendente, respectiv pentru cea mai mică valoare din partiție în cazul ordonării descendente.

Exemplul 9.14

```
SELECT cod, luna,
       SUM(cantitate*pret_unitar_vanzare) VALOARE,
       CUME_DIST() OVER
        (PARTITION BY cod
         ORDER BY SUM(cantitate*pret_unitar_vanzare))
       AS CUME_DIST
FROM   vanzari v, produse p, timp t, plati pl
WHERE  v.produs_id=p.id_produs
AND    v.plata_id=pl.id_plata
```

```
AND    v.timp_id=t.id_timp
AND    luna IN (3, 4, 5, 6, 7)
AND    an = 2007
GROUP BY cod, luna;

COD           LUNA    VALOARE    CUME_DIST
-----------------------------------------
DP               7     1910.03          0.2
DP               5     3330.29          0.4
DP               6     3437.81          0.6
DP               4    13846.35          0.8
DP               3    95131.83          1
Num              7       77.21          0.2
Num              6      224.36          0.4
Num              3      776.24          0.6
Num              4      1081            0.8
Num              5     1681.78          1
```

Interpretarea rezultatelor:

- Linia 1 = 1/5 = 0.2
- Linia 2 = 2/5 = 0.4
- Linia 3 = 3/5 = 0.6
- Linia 4 = 4/5 = 0.8
- Linia 5 = 5/5 =1

De exemplu, pentru linia 3 se poate obține următoarea interpretare:

- *60% dintre vânzările realizate prin dispoziție de plată* (*DP*) *au valoarea mai mică sau egală cu vânzările realizate prin dispoziție de plată* (*DP*) *în luna iunie.*

9.2.3. Funcția PERCENT_RANK

- Funcția *PERCENT_RANK* este similară funcției *CUME_DIST*, numai că aceasta utilizează pozițiile liniilor dintr-o partiție în loc de valorile lor.
  - Determină percentila unei valori relativ la o mulțime de valori.
- Această funcție întoarce poziția procentuală a unei linii în cadrul unei partiții, calculată prin următoarea formulă:

```
(poziția liniei în partiție -1) /
(numărul liniilor din partiție -1)
```

- Funcția *PERCENT_RANK* întoarce valori cuprinse între 0 și 1. Linia care are rangul 1 va avea valoarea *PERCENT_RANK* egală cu 0, iar ultima linie din partiție va avea valoarea *PERCENT_RANK* egală cu 1.

- Sintaxa acestei funcții este următoarea:

```
PERCENT_RANK() OVER
  ([PARTITION BY <expr1> [, ...]]
   ORDER BY <expr2> [ASC|DESC]
        [NULLS FIRST | NULLS LAST] [, ...])
```

Exemplul 9.15

În următorul exemplu se calculează poziția procentuală a valorilor vânzărilor lunare în cadrul vânzările realizate pentru fiecare tip de plată.

```
SELECT  cod, luna,
        SUM(cantitate*pret_unitar_vanzare) VALOARE,
        PERCENT_RANK() OVER (PARTITION BY cod ORDER BY
          SUM(cantitate*pret_unitar_vanzare))
          AS PERCENT_RANK
FROM   vanzari v, produse p, timp t, plati pl
WHERE  v.produs_id=p.id_produs
AND    v.plata_id=pl.id_plata
AND    v.timp_id=t.id_timp
AND    luna IN (3, 4, 5, 6, 7)
AND    an = 2007
GROUP BY cod, luna;
```

```
COD   LUNA   VALOARE    PERCENT_RANK
---- -----  ---------- ------------
DP       7    2827.53            0
DP       5    3841.41         0.25
DP       6    3964.33          0.5
DP       4   14712.69         0.75
DP       3   78258.01            1
Num      6      22               0
Num      7      77.21         0.25
Num      4    1385.64          0.5
Num      3    1830.85         0.75
Num      5    1962.98            1
```

Interpretarea rezultatului:

- Prima partiție este corespunzătoare valorii *DP* și conține primele 5 linii.
- Valorile funcției *PERCENT_RANK* pentru această partiție sunt:
  - Linia 1 = (1-1)/(5-1) = 0 => cea mai mică valoare din mulțime
  - Linia 2 = (2-1)(5-1) = 1/4 = 0.25 => valoarea se află pe percentila 25 => este mai mare decât 25% dintre valorile mulțimii și doar 75% dintre valori pot fi mai mari decât ea

- Linia 3 = (3-1)/(5-1) = 2/4 = 0.5 => valoarea se află pe percentila 50 => este mai mare decât 50% dintre valorile mulțimii și doar 50% dintre valori pot fi mai mari decât ea
- Linia 4 = (4-1)/(5-1) = 3/4 = 0.75 => valoarea se află pe percentila 75 => este mai mare decât 75% dintre valorile mulțimii și doar 25% dintre valori pot fi mai mari decât ea
- Linia 5 = (5-1)/(5-1) = 4/4 = 1 => cea mai mare valoare din mulțime

9.2.4. Funcția NTILE

- Funcția *NTILE* împarte o partiție ordonată într-un număr specificat de grupuri, numite segmente și atribuie fiecărei linii din partiție numărul corespunzător segmentului său. Această funcție permite utilizatorului să împartă o mulțime de date în grupuri egale ca dimensiune.
- Segmentele sunt calculate astfel încât fiecărui segment să-i fie atribuit același număr de linii sau cu cel mult o linie în plus față de celelalte segmente. De exemplu, dacă am avea 100 de linii într-o partiție și am vrea să o împărțim în 4 segmente, atunci fiecărui segment îi vor fi atribuite câte 25 de linii.
- Dacă numărul de linii dintr-o partiție nu este divizibil cu numărul segmentelor, atunci liniile suplimentare vor fi distribuite câte una pe segment. De exemplu, dacă am avea 103 linii într-o partiție cu o funcție *NTILE*(5), primele 21 de linii vor fi atribuite primului segment, următoarele 21 de linii vor fi atribuite celui de al doilea segment, următoarele 21 de linii vor fi atribuite celui de al treilea segment, următoarele 20 de linii vor fi atribuite celui de al patrulea segment, iar ultimele 20 vor fi atribuite celui de-al cincilea segment.
- Funcția *NTILE* are sintaxa următoare:

```
NTILE(N) OVER
  ([PARTITION BY <expr1> [, ...]]
    ORDER BY <expr2> [ASC|DESC]
      [NULLS FIRST | NULLS LAST] [, ...])
```

Exemplul 9.16

În exemplul următor se împarte totalul vânzărilor lunare în 4 segmente:

```
SELECT  luna, SUM(cantitate*pret_unitar_vanzare) VALOARE,
        NTILE(4) OVER (ORDER BY
```

```
           SUM(cantitate*pret_unitar_vanzare)) AS NTILE4
FROM    vanzari v, produse p, timp t
WHERE   v.produs_id=p.id_produs
AND     v.timp_id=t.id_timp
AND     luna IN (3, 4, 5, 6, 7)
AND     an = 2007
GROUP BY luna
ORDER BY valoare;

 LUNA  VALOARE      NTILE4
------ ---------   --------
   2     265.21          1
   7    1987.24          1
   6    3662.17          2
   5    5012.07          2
   4   14927.35          3
   3   95908.07          4
```

Valorile egale pot fi distribuite în segmente adiacente. Pentru a obține rezultate deterministe, trebuie să se utilizeze ordonarea după o cheie unică.

9.2.5. Funcția ROW_NUMBER

- Funcția *ROW_NUMBER* atribuie un număr unic (secvențial, pornind de la 1) fiecărei linii dintr-o partiție. Această funcție are sintaxa următoare:

```
ROW_NUMBER() OVER
  ([PARTITION BY <expr1> [, ...]]
    ORDER BY <expr2> [ASC|DESC]
      [NULLS FIRST | NULLS LAST] [, ...])
```

Exemplul 9.17

```
SELECT  categorie_1 raion, luna,
        SUM(cantitate*pret_unitar_vanzare) VALOARE,
        ROW_NUMBER() OVER (PARTITION BY categorie_1
          ORDER BY SUM(cantitate*pret_unitar_vanzare) DESC)
          AS ROW_NR
FROM    vanzari v, produse p, timp t
WHERE   v.produs_id=p.id_produs
AND     v.timp_id=t.id_timp
AND     luna IN (3, 4, 5, 6, 7)
AND     an = 2007
GROUP BY categorie_1, luna;

RAION               LUNA   VALOARE      ROW_NR
--------------- ---------- ---------- ----------
IT                       4   10903.13          1
```

IT	3	10205.74	2
IT	5	4782.76	3
IT	7	625.1	4
IT	2	265.21	5
IT	6	116.2	6
Papetarie	3	85702.33	1
Papetarie	4	4024.22	2
Papetarie	6	3545.97	3
Papetarie	7	1362.14	4
Papetarie	5	229.31	5

Bibliografie

1. Connolly T.M., Begg C.E., *Database Systems*: *A Practical Approach to Design, Implementation and Management*, 5th edition, Pearson Education, 2005
2. Dollinger R., Andron L., *Baze de date și gestiunea tranzacțiilor*, Editura Albastră, Cluj-Napoca, 2004
3. Inmon W.H., *Building the Data Warehouse*, 4th Edition, Wiley, 2005
4. Kimball R., *The Data Warehouse Toolkit*, 3rd Edition, Wiley, 2013
5. Kimball R., Ross M., Thornthwaite W., Mundy J., Becker B., *The Data Warehouse Lifecycle Toolkit*, 2nd Edition Wiley, 2008
6. Oracle and/or its affiliates, *Oracle Database Concepts*, 1993, 2024
7. Oracle and/or its affiliates, *Oracle Database Administrator's Guide*, 2001, 2025
8. Oracle and/or its affiliates, *Oracle Database Performance Tuning Guide*, 2013, 2025
9. Oracle and/or its affiliates, *Oracle Database SQL Language Reference*, 1996, 2025
10. Oracle and/or its affiliates, *Oracle Database PL/SQL Language Reference*, 1996, 2025
11. Oracle and/or its affiliates, *Oracle Database: SQL Tuning Workshop*, 2010, 2025
12. Oracle and/or its affiliates, *Oracle OLAP Customizing Analytic Workspace Manager*, 2006, 2019
13. Oracle and/or its affiliates, *Oracle OLAP DML Reference*, 1994, 2019
14. Oracle and/or its affiliates, *Oracle OLAP User's Guide*, 2003, 2019
15. Oracle and/or its affiliates, *Oracle Warehouse Builder Concepts*, 2000, 2021
16. Oracle and/or its affiliates, *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*, 2000, 2021
17. Oracle and/or its affiliates, *Oracle Database Data Warehousing Guide*, 2001, 2021
18. Oracle University, *Oracle Database*: *PL/SQL Fundamentals*, *Student Guide*, 2009, 2025
19. Poe V., Klauer P., Brobst S., *Building A Data Warehouse for Decision Support*, 2nd Edition, Prentice Hall; 1997
20. Popescu I., Alecu A., Velcescu L., Florea (Mihai) G., *Programare avansată în Oracle9i*, Ed. Tehnică, 2004