Funciones analíticas RANK, DENSE_RANK, FIRST y LAST

by admin - sábado, octubre 03, 2020

https://dbandtech.com/funciones-analiticas-rank-dense_rank-first-y-last/

Este artículo ofrece una descripción general de las funciones analíticas **RANK, DENSE_RANK, FIRST y LAST**. Si es nuevo en las funciones analíticas, probablemente debería leer primero esta introducción a las funciones analíticas.

Preparar

Los ejemplos de este artículo requieren la siguiente tabla.

```
--DROP TABLE emp PURGE;

CREATE TABLE emp (
  empno    NUMBER(4) CONSTRAINT pk_emp PRIMARY KEY,
  ename    VARCHAR2(10),
  job      VARCHAR2(9),
  mgr      NUMBER(4),
  hiredate DATE,
  sal      NUMBER(7,2),
  comm     NUMBER(7,2),
  deptno   NUMBER(2)
);
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,to_date('17-12-1980'
,'dd-mm-yyyy'),800,NULL,20);
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,to_date('20-2-198
1','dd-mm-yyyy'),1600,300,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,to_date('22-2-1981
','dd-mm-yyyy'),1250,500,30);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,to_date('2-4-1981'
,'dd-mm-yyyy'),2975,NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,to_date('28-9-19
81','dd-mm-yyyy'),1250,1400,30);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,to_date('1-5-1981'
,'dd-mm-yyyy'),2850,NULL,30);
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,to_date('9-6-1981'
,'dd-mm-yyyy'),2450,NULL,10);
```

```
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,to_date('13-JUL-87
','dd-mm-rr')-85,3000,NULL,20);
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,to_date('17-11-19
81','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,to_date('8-9-198
1','dd-mm-yyyy'),1500,0,30);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,to_date('13-JUL-87',
 'dd-mm-rr')-51,1100,NULL,20);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,to_date('3-12-1981',
'dd-mm-yyyy'),950,NULL,30);
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,to_date('3-12-1981'
,'dd-mm-yyyy'),3000,NULL,20);
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,to_date('23-1-1982'
,'dd-mm-yyyy'),1300,NULL,10);
COMMIT;
```

RANK

La descripción básica de la **función analítica RANK** se muestra a continuación.

```
RANK() OVER ([ query_partition_clause ] order_by_clause)
```

Supongamos que queremos asignar un orden secuencial, o rango, a las personas dentro de un departamento según el salario, podríamos usar la función **RANK** de esta manera.

```
SELECT empno,
       deptno,
       sal,
       RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myrank
FROM   emp;
```

EMPNO	DEPTNO	SAL	MYRANK
7934	10	1300	1
7782	10	2450	2
7839	10	5000	3
7369	20	800	1
7876	20	1100	2
7566	20	2975	3
7788	20	3000	4
7902	20	3000	4
7900	30	950	1

```
      7654          30       1250           2
      7521          30       1250           2
      7844          30       1500           4
      7499          30       1600           5
      7698          30       2850           6

SQL>
```

Lo que vemos aquí es donde dos personas tienen el mismo salario y se les asigna el mismo rango. Cuando varias filas comparten el mismo rango, el siguiente rango en la secuencia no es consecutivo. Esto es como una medalla olímpica en el sentido de que si dos personas comparten el oro, no hay medalla de plata, etc.

El hecho de que podamos clasificar las filas en el departamento significa que podemos hacer una consulta **Top-N** por departamento. El siguiente ejemplo asigna el rango en la vista en línea, luego usa ese rango para restringir las filas a los 2 empleados (peor pagados) inferiores en cada departamento.

```
SELECT *
FROM   (SELECT empno,
               deptno,
               sal,
               RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myran
k
        FROM   emp)
WHERE  myrank <= 2;
EMPNO          DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7934          10       1300          1
      7782          10       2450          2
      7369          20        800          1
      7876          20       1100          2
      7900          30        950          1
      7521          30       1250          2
      7654          30       1250          2
SQL>
```

DENSE_RANK

La descripción básica de la **función analítica DENSE_RANK** se muestra a continuación.

```
DENSE_RANK() OVER([ query_partition_clause ] order_by_clause)
```

La función **DENSE_RANK** actúa como la función RANK excepto que asigna rangos consecutivos, por lo que esto no es como una medalla olímpica.

```
SELECT empno,
       deptno,
       sal,
       DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal) AS myrank
FROM   emp;

     EMPNO     DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7934         10       1300          1
      7782         10       2450          2
      7839         10       5000          3
      7369         20        800          1
      7876         20       1100          2
      7566         20       2975          3
      7788         20       3000          4
      7902         20       3000          4
      7900         30        950          1
      7654         30       1250          2
      7521         30       1250          2
      7844         30       1500          3
      7499         30       1600          4
      7698         30       2850          5

SQL>
```

Al igual que con la función analítica **RANK**, podemos hacer una **consulta Top-N** por departamento. El siguiente ejemplo asigna el rango denso en la vista en línea, luego usa ese rango para restringir las filas a los 2 mejores empleados (mejor pagados) en cada departamento.

```
SELECT *
FROM   (SELECT empno,
               deptno,
               sal,
               DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal DES
C) AS myrank
        FROM   emp)
WHERE  myrank <= 2;
EMPNO          DEPTNO        SAL     MYRANK
---------- ---------- ---------- ----------
      7839         10       5000          1
      7782         10       2450          2
```

```
      7788         20       3000          1
      7902         20       3000          1
      7566         20       2975          2
      7698         30       2850          1
      7499         30       1600          2
SQL>
```

FIRST and LAST

Probablemente debería evitar FIRST y LAST en favor de las funciones analíticas FIRST_VALUE y LAST_VALUE, que siguen la **sintaxis normal de la función analítica.** Siempre los uso en lugar de **FIRST y LAST** ahora.

Las **funciones FIRST y LAST** se pueden utilizar para devolver el primer o el último valor de una secuencia ordenada. Digamos que queremos mostrar el salario de cada empleado, junto con el más bajo y el más alto dentro de su departamento, podemos usar algo como.

```
SELECT empno,
       deptno,
       sal,
       MIN(sal) KEEP (DENSE_RANK FIRST ORDER BY sal) OVER (PARTITION B
Y deptno) AS lowest,
       MAX(sal) KEEP (DENSE_RANK LAST ORDER BY sal) OVER (PARTITION BY
 deptno) AS highest
FROM   emp
ORDER BY deptno, sal;

     EMPNO     DEPTNO        SAL     LOWEST    HIGHEST
---------- ---------- ---------- ---------- ----------
      7934         10       1300       1300       5000
      7782         10       2450       1300       5000
      7839         10       5000       1300       5000
      7369         20        800        800       3000
      7876         20       1100        800       3000
      7566         20       2975        800       3000
      7788         20       3000        800       3000
      7902         20       3000        800       3000
      7900         30        950        950       2850
      7654         30       1250        950       2850
      7521         30       1250        950       2850
      7844         30       1500        950       2850
      7499         30       1600        950       2850
      7698         30       2850        950       2850
```

```
SQL>
```

Las funciones MIN y MAX son casi irrelevantes aquí, ya que es FIRST, LAST y KEEP las que seleccionan la fila cuyo valor se utilizará. Podemos demostrar esto usando MIN para el valor alto y bajo.

```
SELECT empno,
       deptno,
       sal,
       MIN(sal) KEEP (DENSE_RANK FIRST ORDER BY sal) OVER (PARTITION B
Y deptno) AS lowest,
       MIN(sal) KEEP (DENSE_RANK LAST ORDER BY sal) OVER (PARTITION BY
 deptno) AS highest
FROM   emp
ORDER BY deptno, sal;

     EMPNO     DEPTNO        SAL     LOWEST    HIGHEST
---------- ---------- ---------- ---------- ----------
      7934         10       1300       1300       5000
      7782         10       2450       1300       5000
      7839         10       5000       1300       5000
      7369         20        800        800       3000
      7876         20       1100        800       3000
      7566         20       2975        800       3000
      7788         20       3000        800       3000
      7902         20       3000        800       3000
      7900         30        950        950       2850
      7654         30       1250        950       2850
      7521         30       1250        950       2850
      7844         30       1500        950       2850
      7499         30       1600        950       2850
      7698         30       2850        950       2850

SQL>
```

Obtenemos el mismo resultado.

También podríamos lograr el mismo resultado usando FIRST_VALUE y LAST_VALUE, o MIN y MAX como funciones analíticas básicas. En la práctica, no utilizo **FIRST y LAST** con mucha frecuencia.
