Funciones de ventana en SQL (*SQL Window Functions*)

Una mini-introducción

Francisco J. Moreno A.

Una función de ventana en SQL permite obtener un determinado valor (un valor calculado a partir de un conjunto de filas) para la fila actual (*the current row*). Ese conjunto de filas define la ventana de filas para la fila actual.

Suponga que una consulta en SQL **está imprimiendo los resultados** y en este instante la consulta se encuentra imprimiendo la fila mostrada en azul abajo (*the current row*). Para esa fila se ejecuta la función de ventana y se obtiene un valor el cual se imprime junto a la fila actual.

- Fila ya impresa … valor obtenido con la función de ventana.
- Fila ya impresa … valor obtenido con la función de ventana.
- Fila ya impresa … valor obtenido con la función de ventana.
- **Fila actual** (que se está imprimiendo)…valor a obtener mediante la función de ventana: se aplica la función de ventana a un conjunto de filas y se obtiene el valor que será impreso junto con la fila actual.
- Siguientes filas a imprimir…

A continuación, se muestran ejemplos concretos.

```
DROP TABLE sale;
CREATE TABLE sale(
code NUMBER(8) PRIMARY KEY,
year NUMBER(4) CHECK(year > 1990 AND year < 2021) NOT NULL,
country VARCHAR(10) NOT NULL,
product VARCHAR(15) NOT NULL,
value NUMBER(8) NOT NULL,
UNIQUE(year, country, product)
);

INSERT INTO sale VALUES(1,  2000,  'USA ',        'TV ',        1500);
INSERT INTO sale VALUES(2,  2000,  'Finland ',    'Phone ',     100);
INSERT INTO sale VALUES(3,  2001,  'Finland ',    'Phone ',     10);
INSERT INTO sale VALUES(4,  2000,  'USA ',        'Calculator ',  75);
INSERT INTO sale VALUES(5,  2000,  'India ',      'Calculator ',  75);
INSERT INTO sale VALUES(6,  2000,  'USA ',        'Computer ',  1200);
INSERT INTO sale VALUES(7,  2002,  'India ',      'Calculator ',  75);
INSERT INTO sale VALUES(8,  2002,  'USA ',        'Computer ',  1500);
INSERT INTO sale VALUES(9,  2001,  'USA ',        'Calculator ',  50);
INSERT INTO sale VALUES(10, 2001,  'USA ',        'Computer ',  1500);
INSERT INTO sale VALUES(11, 2001,  'India ',      'Computer ',  1200);
```

Ejemplo 1.

```
SELECT code,
    SUM(value) OVER() AS totval
FROM sale;
```

CODE	TOTVAL
1	7285
2	7285
3	7285
4	7285
5	7285
6	7285
7	7285
8	7285
9	7285
10	7285
11	7285

Acá se tiene que cada fila sale acompañada de la suma de todos los valores de la tabla. Así, la ventana de filas para cada fila es, en este ejemplo, toda la tabla. La ventana se define mediante la cláusula **OVER**. En este ejemplo la ventana de filas para cada fila es toda la tabla ya que los paréntesis de **OVER** están vacíos.

Ejemplo 2a.

```
SELECT code, country,
    SUM(value) OVER(PARTITION BY country) AS totval
FROM sale;
```

CODE	COUNTRY	TOTVAL
2	Finland	110
3	Finland	110
5	India	1350
7	India	1350
11	India	1350
10	USA	5825
9	USA	5825
8	USA	5825
4	USA	5825
1	USA	5825
6	USA	5825

Acá se tiene que cada fila sale acompañada de la suma de todos los valores correspondientes a su país. Así, la ventana de filas para cada fila son todas las filas de la tabla que tienen el mismo país que la fila actual.

Nótese la diferencia con un **GROUP BY**, si se tiene la consulta:

Ejemplo 2b.

```
SELECT country,
    SUM(value) AS totval
FROM sale
```

```
GROUP BY country;
```

```
COUNTRY        TOTVAL
---------- ----------
USA              5825
Finland           110
India            1350
```

Acá se obtiene para CADA PAÍS la suma de sus valores. Así en esta consulta la función **SUM** es una función de grupo, en la consulta anterior es una función de ventana.

Un aspecto desafortunado de las funciones de ventana es que no se pueden usar en el **WHERE** de la consulta en la que están definidas. Así, lo siguiente **genera un error**:

Ejemplo 2c.

```
SELECT code, country,
    SUM(value) OVER(PARTITION BY country) AS totval
FROM sale
WHERE totval > 1000;
```

Tampoco funciona:

Ejemplo 2d.

```
SELECT code, country,
    SUM(value) OVER(PARTITION BY country) AS totval
FROM sale
WHERE SUM(value) OVER(PARTITION BY country) > 1000;
```

Lo anterior se puede solucionar mediante un **SELECT** externo (es decir, mediante una *inline view*) donde se hace la restricción (o sea, el **WHERE**):

Ejemplo 2e.

```
SELECT *
FROM (SELECT code, country,
            SUM(value) OVER(PARTITION BY country) AS totval
        FROM sale)
WHERE totval > 1000;
```

CODE	COUNTRY	TOTVAL
5	India	1350
7	India	1350
11	India	1350
10	USA	5825
9	USA	5825
8	USA	5825
4	USA	5825
1	USA	5825
6	USA	5825

Ejemplo 3.

Un ejemplo similar al anterior, pero con varias funciones de ventana:

```
SELECT code, country, year,
    SUM(value) OVER(PARTITION BY country) AS totvalco,
    COUNT(*) OVER(PARTITION BY country) AS contco,
    SUM(value) OVER(PARTITION BY year) AS totvalye,
    COUNT(*) OVER(PARTITION BY year) AS contye
FROM sale;
```

CODE	COUNTRY	YEAR	TOTVALCO	CONTCO	TOTVALYE	CONTYE
3	Finland	2001	110	2	2760	4
2	Finland	2000	110	2	2950	5
11	India	2001	1350	3	2760	4
7	India	2002	1350	3	1575	2
5	India	2000	1350	3	2950	5
8	USA	2002	5825	6	1575	2
10	USA	2001	5825	6	2760	4
9	USA	2001	5825	6	2760	4
4	USA	2000	5825	6	2950	5
1	USA	2000	5825	6	2950	5
6	USA	2000	5825	6	2950	5

- **La cláusula ORDER BY en una función de ventana**

En la cláusula **OVER** también se puede especificar un orden para las filas que conforman la ventana. Esto permite calcular acumulados con respecto a la fila actual. Veamos un ejemplo concreto.

Ejemplo 4.

```
SELECT code,
    SUM(value) OVER(ORDER BY code) AS totval
FROM sale;
```

CODE	TOTVAL
1	1500
2	1600
3	1610
4	1685
5	1760
6	2960
7	3035
8	4535
9	4585
10	6085
11	7285

Así se tiene que para la fila con code = 1, que es la fila que tiene el menor code en la tabla, se obtiene que la suma de value es 1500. Ahora, para la fila con code = 2, se obtiene que la suma de value es 1500 + 100, ya que se suman los values de las filas con code = 1 y con code = 2. Y así sucesivamente para las otras filas. Este cálculo en inglés se denomina un *running total*.

Ejemplo 5.

Se pueden usar simultáneamente las dos cláusulas **PARTITION BY** y **ORDER BY**. Por ejemplo:

```
SELECT code, year,
    SUM(value) OVER(PARTITION BY YEAR ORDER BY code) AS totval
FROM sale;
```

CODE	YEAR	TOTVAL
1	2000	1500
2	2000	1600
4	2000	1675
5	2000	1750
6	2000	2950
3	2001	10
9	2001	60
10	2001	1560
11	2001	2760
7	2002	75
8	2002	1575

Así se tiene que para cada fila la ventana está formada por todas las filas que tienen su mismo año.
Y se obtiene su suma de value (un *running total*) según el orden de la fila dado por code. Por ejemplo, para el año 2001 la fila con menor code es la que tiene code = 3, entonces para esa fila se obtiene una suma = 10. Continuando con el año 2001, la fila con el siguiente menor code es la que tiene code = 9. Entonces para ella se obtiene una suma de 10 + 50 = 60.

- **Las funciones de ventana de enumeración**

Una pregunta frecuente es ¿cómo enumerar las filas de una consulta? Para ello SQL ofrece varias opciones. Veamos las funciones de enumeración: **ROW_NUMBER**, **RANK** y **DENSE_RANK**.

Ejemplo 6a.

```
SELECT code, country,
   ROW_NUMBER() OVER (ORDER BY country) AS numord
FROM sale;
```

```
      CODE COUNTRY        NUMORD
---------- ---------- ----------
         2 Finland             1
         3 Finland             2
         5 India               3
         7 India               4
        11 India               5
        10 USA                 6
         9 USA                 7
         8 USA                 8
         4 USA                 9
         1 USA                10
         6 USA                11
```

Acá se ordenan las filas por country y se comienza la enumeración desde 1. Si dos filas tienen el mismo country, su orden es arbitrario, a menos que se agregue un criterio de ordenamiento adicional, por ejemplo:

Ejemplo 6b.

```
SELECT code, country,
   ROW_NUMBER() OVER (ORDER BY country, code) AS numord
FROM sale;
```

```
      CODE COUNTRY        NUMORD
---------- ---------- ----------
         2 Finland             1
         3 Finland             2
         5 India               3
         7 India               4
        11 India               5
         1 USA                 6
         4 USA                 7
         6 USA                 8
         8 USA                 9
         9 USA                10
        10 USA                11
```

También se puede lograr una enumeración local (es decir, enumerar para cada country), así:

Ejemplo 6c.

```
SELECT code, country,
   ROW_NUMBER() OVER (PARTITION BY country ORDER BY country, code) AS numord
FROM sale;
```

```
      CODE COUNTRY         NUMORD
---------- ---------- ----------
         2 Finland             1
         3 Finland             2
         5 India               1
         7 India               2
        11 India               3
         1 USA                 1
         4 USA                 2
         6 USA                 3
         8 USA                 4
         9 USA                 5
        10 USA                 6
```

Ahora veamos una enumeración con RANK.

Ejemplo 6d.

```
SELECT code, country,
   RANK() OVER (ORDER BY country) AS numord
FROM sale;
```

```
      CODE COUNTRY         NUMORD
---------- ---------- ----------
         2 Finland             1
         3 Finland             1
         5 India               3
         7 India               3
        11 India               3
        10 USA                 6
         9 USA                 6
         8 USA                 6
         4 USA                 6
         1 USA                 6
         6 USA                 6
```

Acá se puede ver que para el caso de Finland (que es el country menor según el orden) hay **dos** filas entonces a ambas se les asigna un 1. Luego vienen las filas de la India, entonces se les asigna la posición 3 (porque hubo **dos** filas de Finland, por ello la numeración para las filas de la India es 3) y así sucesivamente.
La diferencia con DENSE_RANK se muestra a continuación:

Ejemplo 6e.

```
SELECT code, country,
    DENSE_RANK() OVER (ORDER BY country) AS numord
FROM sale;
```

```
      CODE COUNTRY        NUMORD
---------- ---------- ----------
         2 Finland             1
         3 Finland             1
         5 India               2
         7 India               2
        11 India               2
        10 USA                 3
         9 USA                 3
         8 USA                 3
         4 USA                 3
         1 USA                 3
         6 USA                 3
```

Así, dependiendo de las necesidades, se puede elegir la función de enumeración apropiada.

- **Las funciones LAG y LEAD.**

Estas funciones permiten acceder a valores de atributos de filas anteriores (LAG) o posteriores (LEAD) con respecto a la fila actual.

Ejemplo 6.

```
SELECT code,
    LAG(code, 1) OVER(ORDER BY code) AS prevcode,
    LAG(value, 1) OVER(ORDER BY code) AS prevval,
    LAG(value, 2) OVER(ORDER BY code) AS prevval2,
    LEAD(code, 1) OVER(ORDER BY code) AS nextcode,
    LEAD(value, 1) OVER(ORDER BY code) AS nextval,
    LEAD(value, 2) OVER(ORDER BY code) AS nextval2
FROM sale;
```

CODE	PREVCODE	PREVVAL	PREVVAL2	NEXTCODE	NEXTVAL	NEXTVAL2
1				2	100	10
2	1	1500		3	10	75
3	2	100	1500	4	75	75
4	3	10	100	5	75	1200
5	4	75	10	6	1200	75
6	5	75	75	7	75	1500
7	6	1200	75	8	1500	50
8	7	75	1200	9	50	1500
9	8	1500	75	10	1500	1200
10	9	50	1500	11	1200	
11	10	1500	50			

Recordar que las “celdas” que aparecen sin valor (**en blanco**) representan **nulos**.

Estas funciones reciben dos parámetros: el primero corresponde al atributo del cual se desea obtener su valor y el segundo el desplazamiento (cuantas filas hacia atrás (LAG) o cuantas filas hacia adelante (LEAD) con respecto a la fila actual). Si el desplazamiento accede a una fila que no existe, los valores retornados son NULL, por ejemplo, no existe fila anterior (LAG) a la fila con code = 1 ni existe fila posterior (LEAD) a la fila con code = 11.

Hasta acá (todo lo anterior) hace parte del tema para el examen final

---

Nota (lo correspondiente a esta nota no entra en el examen). Por otro lado, aunque **no se evaluará lo siguiente,** tenga en cuenta que existen otras funciones de ventana. Además, las funciones de ventana tienen muchas opciones que vale la pena explorar.

Referencia
Algunos fragmentos de este documento fueron modificados y adaptados de
https://dev.mysql.com/doc/refman/8.0/en/window-functions-usage.html

Las pruebas las hice en Oracle 21.
Un saludo,
Pacho.