SQL V

Base de Datos 2021

Window Functions

CONSULTAS EN SQL - FUNCIONES VENTANA

```
[WINDOW_FUNCTION] (expression)
OVER (
  [PARTITION BY expressions, ...]
  [ORDER BY expressions, ...]
)
```

- Las **funciones ventana** permiten hacer cálculos a lo largo de un conjunto de filas relacionadas a la fila actual.
- Una función de ventana realiza operaciones *similares* a la agregación.
- Sin embargo, adiciona la información agregada a cada fila correspondiente, en lugar de devolver una única fila por grupo, como sí las funciones de agregación.

FUNCIONES VENTANA - TIPOS DE FUNCIONES

AGREGACIÓN

- AVG
- COUNT
- MIN/MAX
- SUM

RANKING

- RANK
- DENSE_RANK
- ROW_NUMBER

VALOR

- FIRST_VALUE
- LAST_VALUE
- NTH_VALUE
- LAG
- LEAD

Algunas otros ejemplos: https://mariadb.com/kb/en/window-functions-overview/

```
SELECT
                RANK() OVER (PARTITION BY department ORDER BY salary DESC)
                        AS dept_ranking,
                department,
                employee_id,
                full_name,
                salary
FROM employee;
```

dept_ranking bigint	department text	employee_id integer	full_name text	salary numeric
1	ACCOUNTS	105	Mareen Bisset	1200.00
2	ACCOUNTS	106	Airton Graue	1100.00
1	IT	104	Milton Kowarsky	1800.00
2	IT	101	Sean Moldy	1500.00
1	SALES	102	Peter Dugan	2000.00
2	SALES	103	Lilian Penn	1700.00
3	SALES	100	Mary Johns	1000.00

```
SELECT  employee_id,
             full_name,
             department,
             salary,
             salary / MAX(salary) OVER (PARTITION BY department ORDER BY salary DESC)
                       AS salary_metric
FROM employee
ORDER BY 5;
```

employee_id integer	full_name text	department text	salary numeric	salary_metric numeric
100	Mary Johns	SALES	1000.00	0.50
101	Sean Moldy	IT	1500.00	0.83
103	Lilian Penn	SALES	1700.00	0.85
106	Airton Graue	ACCOUNTS	1100.00	0.92
104	Milton Kowarsky	IT	1800.00	1.00
105	Mareen Bisset	ACCOUNTS	1200.00	1.00
102	Peter Dugan	SALES	2000.00	1.00

Ejemplo 1: Agregación y Ranking

Se tiene la siguiente tabla **sales**:

year	country	product	profit
2000	Finland	Computer	1500
2000	Finland	Phone	100
2000	India	Calculator	75
2000	India	Computer	1200
2000	USA	Calculator	75
2000	USA	Computer	1500
2001	Finland	Phone	10
2001	USA	Calculator	50
2001	USA	Computer	1500
2001	USA	TV	150

Se desea encontrar el ratio de aporte de cada producto con respecto a la ganancia total de ese año del país.

Además, se desea explicitar el ranking de la ganancia de cada producto con respecto a la ganancia total histórica por país.

La tabla debe estar ordenada por:

- País ascendente
- Ranking ascendente

Ratio de Aporte con Agregaciones

La primera parte del problema se puede resolver con agregaciones y joins:

```
WITH
    `total_profit_per_year` AS (
        SELECT `year`, `country`, SUM(profit) as `sum_profit`
        FROM `sales` GROUP BY `year`, `country`
    )
SELECT `year`, `country`, `product`, `profit`,
       `profit`/`sum_profit` as `ratio`
FROM `sales` INNER JOIN `total_profit_per_year` USING(`year`, `country`);
```

Problema: esta query es muy verbosa

Ratio de Aporte con Funciones de Ventana

Con funciones de ventana, se puede escribir una query equivalente menos verbosa:

```
SELECT `year`, `country`, `product`, `profit`,
       `profit`/SUM(`profit`) OVER (PARTITION BY `country`, `year`) AS `ratio`
FROM `sales;
```

El partition-by de una ventana de agregación está en *biyección* con el group-by de una agregación.

Luego, es posible computar sobre una ventana vacía, i.e., `OVER ()`, la cual se corresponde a una agregación sin group-by

```
+------+---------+-----------+--------+--------+
| year | country | product   | profit | ratio  |
+------+---------+-----------+--------+--------+
| 2000 | Finland | Computer  | 1500   | 0.9375 |
| 2000 | Finland | Phone     | 100    | 0.0625 |
| 2000 | India   | Calculator| 75     | 0.0588 |
| 2000 | India   | Computer  | 1200   | 0.9412 |
| 2000 | USA     | Calculator| 75     | 0.0476 |
| 2000 | USA     | Computer  | 1500   | 0.9524 |
| 2001 | Finland | Phone     | 10     | 1.0000 |
| 2001 | USA     | Calculator| 50     | 0.0294 |
| 2001 | USA     | Computer  | 1500   | 0.8824 |
| 2001 | USA     | TV        | 150    | 0.0882 |
+------+---------+-----------+--------+--------+
```

Agregando el Ranking

La función de ranking no es de agregación, y solo puede utilizarse como función de ventana:

```
SELECT `year`, `country`, `product`, `profit`,
       `profit`/SUM(`profit`) OVER (PARTITION BY `year`, `country`) AS `ratio`,
       RANK() OVER (PARTITION BY `country` ORDER BY `profit` DESC) as `ranking`
FROM `sales`
ORDER BY `country` ASC, `ranking` ASC;
```

El ranking se relaciona estrechamente con el order-by de la ventana. Si la ventana no lleva order-by, todas las filas van a estar en empate.

La resolución de un empate depende de la función de ranking utilizada.

Agregando el Ranking

```
SELECT `year`, `country`, `product`, `profit`,
       `profit`/SUM(`profit`) OVER (PARTITION BY `year`, `country`) AS `ratio`,
       RANK() OVER (PARTITION BY `country` ORDER BY `profit` DESC) as `ranking`
FROM `sales`
ORDER BY `country` ASC, `ranking` ASC;
```

year	country	product	profit	ratio	ranking
2000	Finland	Computer	1500	0.9375	1
2000	Finland	Phone	100	0.0625	2
2001	Finland	Phone	10	1.0000	3
2000	India	Computer	1200	0.9412	1
2000	India	Calculator	75	0.0588	2
2000	USA	Computer	1500	0.9524	1
2001	USA	Computer	1500	0.8824	1
2001	USA	TV	150	0.0882	3
2000	USA	Calculator	75	0.0476	4
2001	USA	Calculator	50	0.0294	5

Ventanas Nombradas

Las ventanas también pueden ser declaradas con un nombre, el cual podrá ser referenciado después de la keyword OVER:

```
SELECT `year`, `country`, `product`, `profit`,
       `profit`/SUM(`profit`) OVER `W_annual` AS `ratio`,
       RANK() OVER `W_historic` as `ranking`
FROM `sales`
WINDOW `W_annual` AS (PARTITION BY `year`, `country`),
       `W_historic` AS (PARTITION BY `country`)
ORDER BY `country` ASC, `ranking` ASC;
```

Ejemplo 2: Ranking y Valor

Se tiene la siguiente tabla **race**:

time	runner
0:03:15	a
0:07:56	a
0:11:39	a
0:16:03	a
0:03:32	b
0:07:12	b
0:12:01	b
0:15:43	b
0:02:58	c
0:07:34	c
0:11:51	c
0:16:17	c

Dicha tabla muestra los tiempos en los cuales los corredores **a**, **b** y **c** completaron cada una de las cuatro vueltas de una carrera.

Se desea mostrar el tiempo que demoró cada corredor en realizar cada vuelta.

La tabla debe estar ordenada por:

- Corredor ascendente
- Vuelta ascendente

Computando con valores anteriores

Las funciones de ventana también permiten acceder a valores de filas anteriores y posteriores:

```
SELECT `runner`,
       RANK() OVER `W` AS `lap`,
       TIMEDIFF(`time`, LAG(`time`, 1, "00:00:00") OVER `W`) AS `lap_time`
FROM `race`
WINDOW `W` AS (PARTITION BY `runner` ORDER BY `time` ASC)
ORDER BY `runner` ASC, `lap` ASC;
```

El segundo parámetro de LAG indica cuántas filas hacia atrás debe mirar, y el tercer parámetro indica el valor por defecto si no existe la fila que se está buscando.

El order-by de la ventana es crucial para elegir la fila correcta.

También existe la función de ventana LEAD, la cual mira hacia adelante.

Computando con valores anteriores

```
SELECT `runner`,
       RANK() OVER `W` AS `lap`,
       TIMEDIFF(`time`, LAG(`time`, 1, "00:00:00") OVER `W`) AS `lap_time`
FROM `race`
WINDOW `W` AS (PARTITION BY `runner` ORDER BY `time` ASC)
ORDER BY `runner` ASC, `lap` ASC;
```

runner	lap	lap_time
a	1	0:03:15
a	2	0:04:41
a	3	0:03:43
a	4	0:04:24
b	1	0:03:32
b	2	0:03:40
b	3	0:04:49
b	4	0:03:42
c	1	0:02:58
c	2	0:04:36
c	3	0:04:17
c	4	0:04:26

Restricción de las Funciones de Ventana

Una función de ventana solo puede ocurrir en el **select** y en el **order-by** de una query.

Para poder filtrar con un predicado sobre el valor computado por una función de ventana, es necesario utilizar una subquery.

Restricción de las Funciones de Ventana

Suponer que, en el ejemplo anterior, se desean encontrar las vueltas de cada corredor que hayan durado entre 3 y 4 minutos. Luego, se puede escribir la siguiente query:

```
WITH
    `t` AS (
        SELECT `runner`,
               RANK() OVER `W` AS `lap`,
               TIMEDIFF(`time`, LAG(`time`, 1, "00:00:00") OVER `W`) AS `lap_time`
        FROM `race`
        WINDOW `W` AS (PARTITION BY `runner` ORDER BY `time` ASC)
        ORDER BY `runner` ASC, `lap` ASC
    )
SELECT `runner`, `lap`
FROM `t`
WHERE `lap_time` BETWEEN '00:03:00' AND '00:04:00';
```