SQL 窗口函数速查表

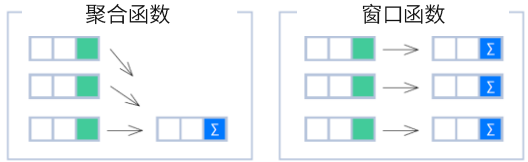
不剪发的Tony 老师

窗口函数（分析函数）

基于一个滑动窗口，也就是与当前行相关的一组数据行计算出一个结果。

聚合函数与窗口函数

聚合函数将一组数据汇总成一个结果，窗口函数为每一行数据计算出一个结果。



窗口函数

```
SELECT column1, column2,
  window_function OVER (
    PARTITION BY ...
    ORDER BY ...
    frame_clause) AS column_alias
FROM table_name;
```

命名窗口

```
SELECT column1, column2,
  window_function OVER window_name
FROM table_name
WINDOW window_name AS (
  PARTITION BY ...
  ORDER BY ...
  frame_clause);
```

分区（PARTITION BY）、排序（ORDER BY）以及窗口大小（frame_clause）都是可选项，MySQL、PostgreSQL、SQLite 支持命名窗口，Oracle、SQL Server 不支持。

SQL 子句逻辑执行顺序

1. FROM、JOIN
2. WHERE
3. GROUP BY
4. aggregate function
5. HAVING
6. window function
7. SELECT
8. DISTINCT
9. UNION、INTERSECT、EXCEPT、MINUS
10. ORDER BY
11. OFFSET
12. LIMIT、FETCH、TOP

窗口函数可以用于 SELECT 列表或者 ORDER BY 子句中，但是不能出现在其他子句中。

PARTITION BY

如果指定了分区，将会针对每个分区单独进行分析；否则，所有数据作为一个整体进行分析。

month	city	sold
1	Rome	200
2	Paris	500
1	London	100
1	Paris	300
2	Rome	300
2	London	400
3	Rome	400

PARTITION BY city

month	city	sold	sum
1	Paris	300	800
2	Paris	500	800
1	Rome	200	900
2	Rome	300	900
3	Rome	400	900
1	London	100	500
2	London	400	500

ORDER BY

指定分区内的排序方式，与 ORDER BY 子句的作用类似。排序选项通常用于数据的分类排名。

sold	city	month
200	Rome	1
500	Paris	2
100	London	1
300	Paris	1
300	Rome	2
400	London	2
400	Rome	3

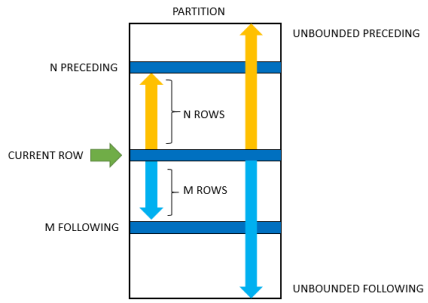
PARTITION BY city ORDER BY month

sold	city	month
300	Paris	1
500	Paris	2
200	Rome	1
300	Rome	2
400	Rome	3
100	London	1
400	London	2

窗口大小

基于当前行指定一个滑动的窗口，窗口总是位于分区范围之内。指定了窗口之后，分析函数不再基于分区进行计算，而是基于窗口内的数据进行计算。

```
ROWS | RANGE | GROUPS BETWEEN frame_start AND frame_end
```



frame_start:

- UNBOUNDED PRECEDING
- N PRECEDING
- CURRENT ROW

frame_end:

- CURRENT ROW
- N FOLLOWING
- UNBOUNDED FOLLOWING

ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING

	city	sold	month
	Paris	300	1
	Rome	200	1
	Paris	500	2
current row	Rome	100	4
	Paris	200	4
	Paris	300	5
	Rome	200	5
	London	200	5
	London	100	6
	Rome	300	6

1 row before the current row and 1 row after the current row

RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING

	city	sold	month
	Paris	300	1
	Rome	200	1
	Paris	500	2
current row	Rome	100	4
	Paris	200	4
	Paris	300	5
	Rome	200	5
	London	200	5
	London	100	6
	Rome	300	6

values in the range between 3 and 5 ORDER BY must contain a single expression

GROUPS BETWEEN 1 PRECEDING AND 1 FOLLOWING

	city	sold	month
	Paris	300	1
	Rome	200	1
	Paris	500	2
current row	Rome	100	4
	Paris	200	4
	Paris	300	5
	Rome	200	5
	London	200	5
	London	100	6
	Rome	300	6

1 group before the current row and 1 group after the current row regardless of the value

只有 PostgreSQL、SQLite 支持 GROUPS 选项。

SQL 窗口函数速查表

不剪发的Tony 老师

常用聚合函数列表

排名窗口函数
- RANK
- DENSE_RANK
- PERCENT_RANK
- ROW_NUMBER
- NTILE
- CUME_DIST

取值窗口函数
- LAG
- LEAD
- FIRST_VALUE
- LAST_VALUE
- NTH_VALUE

聚合窗口函数
- AVG
- SUM
- COUNT
- MAX
- MIN

SQL Server 不支持 NTH_VALUE 函数。

默认窗口
如果指定了 ORDER BY，默认 RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
否则，默认为 ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

排名窗口函数
- RANK()，计算每行数据在其分区中的名次；如果存在名次相同的数据，后续的排名将会产生跳跃。
- DENSE_RANK()，计算每行数据在其分区中的名次；存在名次相同的数据，后续的排名不会跳跃。
- PERCENT_RANK()，与 RANK() 相同，但是以百分比的形式显示每行数据的名次，取值范围 [0-1]。
- ROW_NUMBER()，为分区中的每行数据分配一个唯一序列号，从 1 开始分配。
- NTILE()， 将分区内的数据分为 N 等份，为每行数据计算其所在的位置。
- CUME_DIST()， 计算每行数据在其分区内的累积分布， 也就是排在该行数据之前的所有数据所占的比率，取值范围 (0-1]。

city	price	row_number	rank	dense_rank
		over(order by price)		
Paris	7	1	1	1
Rome	7	2	1	1
London	8.5	3	3	2
Berlin	8.5	4	3	2
Moscow	9	5	5	3
Madrid	10	6	6	4
Oslo	10	7	6	4

cume_dist() OVER(ORDER BY sold)

city	sold	cume_dist
Paris	100	0.2
Berlin	150	0.4
Rome	200	0.8
Moscow	200	0.8
London	300	1

80% of values are less than or equal to this one

percent_rank() OVER(ORDER BY sold)

city	sold	percent_rank
Paris	100	0
Berlin	150	0.25
Rome	200	0.5
Moscow	200	0.5
London	300	1

without this row 50% of values are less than this row's value

ntile(3)

city	sold	
Rome	100	1
Paris	100	1
London	200	1
Moscow	200	2
Berlin	200	2
Madrid	300	2
Oslo	300	3
Dublin	300	3

排名窗口函数不支持动态的窗口大小（frame_clause），而是以整个分区（PARTITION BY）作为分析的窗口。

取值窗口函数
- LAG(expr, offset, default)， 返回分区中当前行之前第 offset 行对应的 expr 。 offset 和 default 可选，默认值分别为 1 和 NULL。
- LEAD(expr, offset, default)， 返回分区中当前行之后第 offset 行的对应expr 。 offset 和 default 可选，默认值分别为 1 和 NULL。
- FIRST_VALUE(expr)， 返回窗口内第一行对应的 expr 。
- LAST_VALUE(expr)， 返回窗口内最后一行对应的 expr 。
- NTH_VALUE(expr, n)， 返回窗口内第 n 行对应的 expr 。

first_value(sold) OVER (PARTITION BY city ORDER BY month)

city	month	sold	first_value
Paris	1	500	500
Paris	2	300	500
Paris	3	400	500
Rome	2	200	200
Rome	3	300	200
Rome	4	500	200

last_value(sold) OVER (PARTITION BY city ORDER BY month **RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING**)

city	month	sold	last_value
Paris	1	500	400
Paris	2	300	400
Paris	3	400	400
Rome	2	200	500
Rome	3	300	500
Rome	4	500	500

nth_value(sold, 2) OVER (PARTITION BY city ORDER BY month)

city	month	sold	nth_value
Paris	1	500	300
Paris	2	300	300
Paris	3	400	300
Rome	2	200	300
Rome	3	300	300
Rome	4	500	300
Rome	5	300	300
London	1	100	NULL

lag(sold) OVER(ORDER BY month)

order by month	month	sold	
	1	500	NULL
	2	300	500
	3	400	300
	4	100	400
	5	500	100

lead(sold) OVER(ORDER BY month)

order by month	month	sold	
	1	500	300
	2	300	400
	3	400	100
	4	100	500
	5	500	NULL

lag(sold, 2, 0) OVER(ORDER BY month)

order by month	month	sold		
	1	500	0	offset=2
	2	300	0	
	3	400	500	
	4	100	300	
	5	500	400	

lead(sold, 2, 0) OVER(ORDER BY month)

order by month	month	sold		
	1	500	400	
	2	300	100	
	3	400	500	
	4	100	0	offset=2
	5	500	0	

LAG 和 LEAD 函数不支持动态的窗口大小（frame_clause），而是以整个分区（PARTITION BY）作为分析的窗口。