睿创神脑
RUICHUANGSHENNAO

PolarDB
PostgreSQL

PostgreSQL
窗口函数

睿创神脑
RUICHUANGSHENNAO

PolarDB PostgreSQL

Objectives

- 窗口函数如何定义
- 专用窗口函数的种类
- 掌握常用的窗口函数
- 熟练使用聚合函数作为窗口函数
- 窗口函数的框架来计算移动平均

睿创神脑
RUICHUANGSHENNAO

“窗口”的由来

PolarDB PostgreSQL

- 窗口函数也称为 OLAP 函数。为了让大家快速形成直观印象，才起了这样一个容易理解的名称。
- 通过 PARTITION BY 分组后的记录集合称为“窗口”。
- 从词语意思的角度考虑，可能“组”比“窗口”更合适一些，但是在SQL中，“组”更多的是用来特指使用 GROUP BY 分割后的记录集合，因此，为了避免混淆，使用PARTITION BY 时称为窗口。
- 注意：可以不指定 PARTITION BY ，会将这个表当成一个“大窗口”。

睿创神脑
RUICHUANGSHENNAO

窗口函数应用场景

PolarDB PostgreSQL

应用场景：

（1）用于分区排序

（2）动态Group By

（3）Top N

（4）累计计算

（5）层次查询

睿创神脑
RUICHUANGSHENNAO

窗口函数的种类

PolarDB PostgreSQL

- 窗口函数大体可以分为以下两种：
  1、能够作为窗口函数的聚合函数（ SUM 、 AVG 、 COUNT 、 MAX 、 MIN ）。
  2、RANK 、 DENSE _ RANK 、 ROW _ NUMBER 等专用窗口函数。

上面第一种应用中将聚合函数书写在语法的“< 窗口函数 >”中，就能够当作窗口函数来使用了。聚合函数根据使用语法的不同，可以在聚合函数和窗口函数之间进行转换。

上面第二种应用中的函数是标准 SQL 定义的 OLAP 专用函数，这里将其统称为“专用窗口函数”。从这些函数的名称可以很容易看出其 OLAP 的用途。

睿创神脑
RUICHUANGSHENNAO

专用窗口函数

PolarDB PostgreSQL

- RANK 函数
  计算排序时，如果存在相同位次的记录，则会跳过之后的位次。
  比如：有 3 条记录排在第 1 位时：1 位、1 位、1 位、4 位……

- ROW_NUMBER 函数
  赋予唯一的连续位次。
  比如：有 3 条记录排在第 1 位时：1 位、2 位、3 位、4 位……

- DENSE_RANK 函数
  同样是计算排序，即使存在相同位次的记录，也不会跳过之后的位次。
  比如：有 3 条记录排在第 1 位时：1 位、1 位、1 位、2 位……

专用窗口函数-RANK()函数

--示例：

```
select ename,job,sal,
    rank() over (PARTITION BY job
                 ORDER BY  sal) as rankin
from emp;
```

- PARTITION BY 能够设定分组和排序的对象范围。本例中，为了按照工作进行分组和排序，我们指定了job。
- ORDER BY 能够指定按照哪一列、何种顺序进行排序。为了按照工资的升序进行排列，我们指定了sal 。

ename	job	sal	rankin
SCOTT	ANALYST	2000	1
FORD	ANALYST	3000	2
SMITH	CLERK	800	1
JAMES	CLERK	950	2
ADAMS	CLERK	1100	3
MILLER	CLERK	1300	4
CLARK	MANAGER	2450	1
BLAKE	MANAGER	2850	2
JONES	MANAGER	2975	3
KING	PRESIDENT	5000	1
MARTIN	SALESMAN	1250	1
WARD	SALESMAN	1250	1
TURNER	SALESMAN	1500	3
ALLEN	SALESMAN	1600	4

分组并排序

排序列

存在相同位次的记录，则会跳过之后的位次

睿创神脑
RUICHUANGSHENNAO

专用窗口函数-DENSE_RANK()函数

PolarDB PostgreSQL

--示例

```
select ename,job,sal,
DENSE_RANK() over (PARTITION by job
                   ORDER BY SAL ) as dense_rankin
from emp;
```

ename	job	sal	dense_rankin
SCOTT	ANALYST	2000	1
FORD	ANALYST	3000	2
SMITH	CLERK	800	1
JAMES	CLERK	950	2
ADAMS	CLERK	1100	3
MILLER	CLERK	1300	4
CLARK	MANAGER	2450	1
BLAKE	MANAGER	2850	2
JONES	MANAGER	2975	3
KING	PRESIDENT	5000	1
MARTIN	SALESMAN	1250	1
WARD	SALESMAN	1250	1
TURNER	SALESMAN	1500	2
ALLEN	SALESMAN	1600	3

分组列

排序列

相同位次的记录，也不会跳过之后的位次

睿创神脑
RUICHUANGSHENNAO

专用窗口函数- ROW_NUMBER 函数

PolarDB PostgreSQL

示例：

```
select ename,job,sal,
ROW_NUMBER() over (PARTITION BY job
                   ORDER BY SAL ) as unique_rankin
from emp;
```

ename	job	sal	unique_rankin
SCOTT	ANALYST	2000	1
FORD	ANALYST	3000	2
SMITH	CLERK	800	1
JAMES	CLERK	950	2
ADAMS	CLERK	1100	3
MILLER	CLERK	1300	4
CLARK	MANAGER	2450	1
BLAKE	MANAGER	2850	2
JONES	MANAGER	2975	3
KING	PRESIDENT	5000	1
MARTIN	SALESMAN	1250	1
WARD	SALESMAN	1250	2
TURNER	SALESMAN	1500	3
ALLEN	SALESMAN	1600	4

分组列

排序列

存在相同位次的记录，也不会跳过之后的位次

睿创神脑
RUICHUANGSHENNAO

专用窗口函数使用技巧

PolarDB PostgreSQL

- 使用 RANK 或 ROW_ NUMBER 时无需任何参数，只需要像 RANK ()或者 ROW_ NUMBER() 这样保持括号中为空就可以了。这也是专用窗口函数通常的使用方式。

```
select ename,job,sal,
    RANK() OVER (PARTITION BY job
                 ORDER BY sal) as rankin,
    DENSE_RANK() OVER (PARTITION BY job
                       ORDER BY sal) as dense_rank,
    ROW_NUMBER() OVER (PARTITION BY job
                       ORDER BY sal) as row_rankin
from emp;
```

睿创神脑
RUICHUANGSHENNAO

PolarDB
PostgreSQL

窗口函数的适用范围

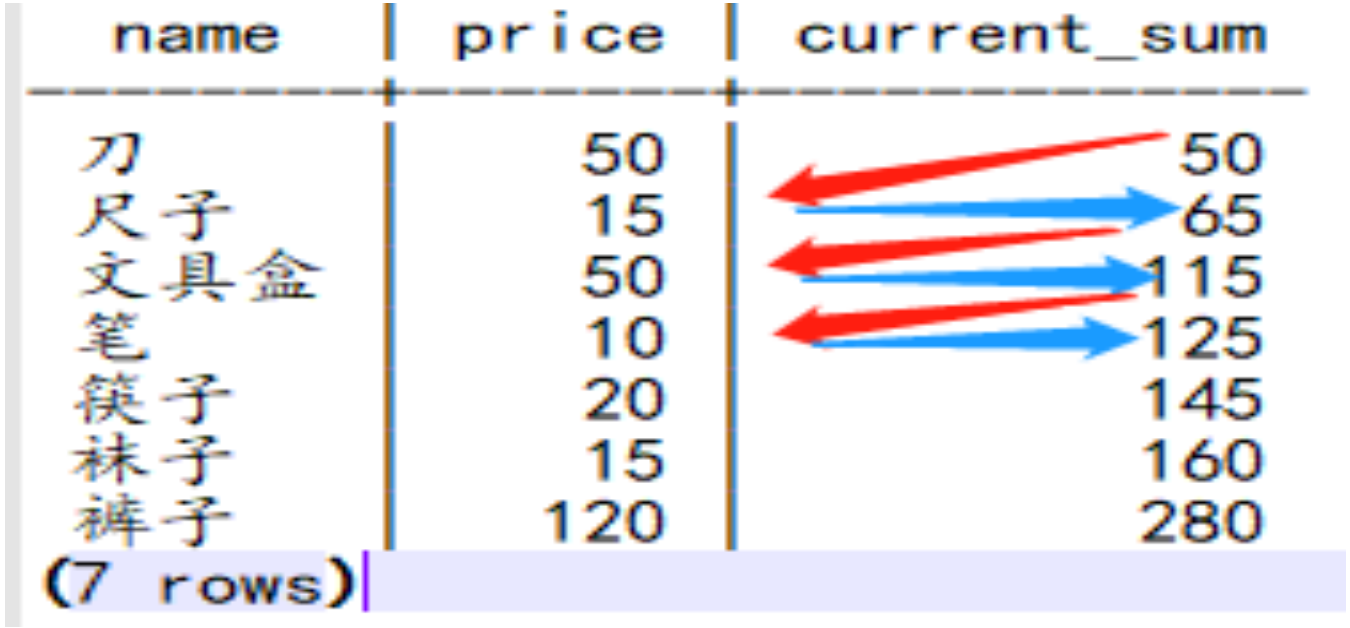
- 使用窗口函数的位置却有非常大的限制。更确切地说，窗口函数只能书写在一个特定的位置。
  这个位置就是 SELECT 子句之中。反过来说，就是这类函数不能在WHERE 子句或者 GROUP BY 子句中使用。

- 为什么窗口函数只能在 SELECT 子句中使用呢?
  在 DBMS内部，窗口函数是对 WHERE 子句或者 GROUP BY 子句处理后的“结果”进行的操作。大家仔细想一想就会明白，在得到用户想要的结果之前，即使进行了排序处理，结果也是错误的。在得到排序结果之后，如果通过 WHERE 子句中的条件除去了某些记录，或者使用 GROUP BY 子句进行了汇总处理，那好不容易得到的排序结果也无法使用了。

作为窗口函数使用的聚合函数

----计算price值的累计结果

```
select name,price,
        SUM(price) over (order by name) as current_sum
from product;
```



name	price	current_sum
刀	50	50
尺子	15	65
文具盒	50	115
笔	10	125
筷子	20	145
袜子	15	160
裤子	120	280

(7 rows)

睿创神脑
RUICHUANGSHENNAO

作为窗口函数使用的聚合函数

PolarDB PostgreSQL

----计算SAL值的累计结果

```
select ename,sal,
        SUM(sal) over (ORDER BY ename) as current_sum
from emp;
```

ename	sal	current_sum
ADAMS	1100	1100
ALLEN	1600	2700
BLAKE	2850	5550
CLARK	2450	8000
FORD	3000	11000
JAMES	950	11950
JONES	2975	14925
KING	5000	19925
MARTIN	1250	21175
MILLER	1300	22475
SCOTT	2000	24475
SMITH	800	25275
TURNER	1500	26775
WARD	1250	28025

睿创神脑
RUICHUANGSHENNAO

作为窗口函数使用的聚合函数

PolarDB PostgreSQL

- 所有的聚合函数都能用作窗口函数，其语法和专用窗口函数完全相同。
- 使用 SUM 函数时，并不像 RANK 或者 ROW _ NUMBER 那样括号中的内容为空，而是和之前我们学过的一样，需要在括号内指定作为汇总对象的列。

睿创神脑
RUICHUANGSHENNAO

PolarDB PostgreSQL

指定框架（汇总范围）

```
select name,price,
avg (price) over (order by name rows 2 preceding) as moving_avg
from product;
```

name	price	moving_avg	
刀	50	50.0000000000000000	---(50)/1
尺子	15	32.5000000000000000	---(50+15)/2
文具盒	50	38.3333333333333333	---(50+15+50)/3
笔	10	25.0000000000000000	---(15+50+10)/3
筷子	20	26.6666666666666667	---(50+10+20)/3
袜子	15	15.0000000000000000	---(10+20+15)/3
裤子	120	51.6666666666666667	---(20+15+120)/3

这里我们使用了 ROWS （“行”）和 PRECEDING （“之前”）两个关键字，将框架指定为“截止到之前 ~ 行”，因此“ ROWS 2 PRECEDING ”就是将框架指定为“截止到之前 2 行”，也就是将作为汇总对象的记录限定为如下的“最靠近的 3 行”。
最靠近的3行=自身（当前记录）+ 之前第1行的记录 + 之前第2行的记录

睿创神脑
RUICHUANGSHENNAO

计算移动平均

PolarDB PostgreSQL

- 由于框架是根据当前记录来确定的，因此和固定的窗口不同，其范围会随着当前记录的变化而变化。

- 这样的统计方法称为移动平均（moving average）。由于这种方法在希望实时把握“最近状态”时非常方便，因此常常会应用在对股市趋势的实时跟踪当中。

- 使用关键字 FOLLOWING （“之后”）替换 PRECEDING ，就可以指定“截止到之后~行”作为框架了。

睿创神脑
RUICHUANGSHENNAO

计算移动平均—同时指定前后行

PolarDB PostgreSQL

```
select name,price,
avg (price) over (order by name rows between 1 preceding and 1 following) as
moving_avg
from product;
```

name	price	moving_avg	
刀	50	32.5000000000000000	---(50+15)/2
尺子	15	38.3333333333333333	---(50+15+50)/3
文具盒	50	25.0000000000000000	---(15+50+10)/3
笔	10	26.6666666666666667	---(50+10+20)/3
筷子	20	15.0000000000000000	---(10+20+15)/3
袜子	15	51.6666666666666667	---(20+15+120)/3
裤子	120	67.5000000000000000	---(15+120)/2

睿创神脑
RUICHUANGSHENNAO

PolarDB PostgreSQL

两个order by

- OVER 子句中的 ORDER BY 只是用来决定窗口函数按照什么样的顺序进行计算的，对结果的排列顺序并没有影响。在 SELECT 语句的最后，使用 ORDER BY子句进行指定按照 ranking 列进行排列，结果才会顺序显示，但是如果使用了，会打乱原本窗口函数出来的显示结果。

- 有些 DBMS（PG） 也可以按照窗口函数的 ORDER BY 子句所指定的顺序对结果进行排序。

- 在一条 SELECT 语句中使用两次 ORDER BY 会有点别扭，但是尽管这两个 ORDER BY 看上去是相同的，但其实它们的功能却完全不同。

睿创神脑
RUICHUANGSHENNAO

PolarDB
PostgreSQL

总结

- 专用窗口函数
  rank()
  row_number()
  dense_ranking()。

- 将聚合函数作为窗口函数使用---需要带参数

- 框架的用法---计算移动平均

睿创神脑
RUICHUANGSHENNAO

PolarDB
PostgreSQL

练习

- 1、设计一张表用来存储高中生每门课程的每场考试成绩，包括 学号,姓名,年级,班级,科目,成绩,考试场次.
- 2、写入一些测试数据
- 3、使用窗口函数查询最近一次考试，每个人班级每科排名、总分排名，年级每科排名、总分排名，与班级第一名的科目分差和总分分差，与年级第一名的科目分差和总分分差.

睿创神脑
RUICHUANGSHENNAO

PolarDB PostgreSQL