postgreSQL 窗口函数总结

窗口函数说明

1、我们都知道在 SQL 中有一类函数叫做聚合函数,例如 sum()、avg()、max()等等,这类函数可以将多行数据按照规则聚集为一行,一般来讲聚集后的行数是要少于聚集前的行数的,但是有时我们想要既显示聚集前的数据,又要显示聚集后的数据,这时我们便引入了窗口函数。
2、在所有的 SQL 处理中,窗口函数都是最后一步执行,而且仅位于 Order by 字句之前。
3、Partition By 子句可以称为查询分区子句,非常类似于 Group By,都是将数据按照边界值分组,而 Over 之前的函数在每一个分组之内进行,如果超出了分组,则函数会重新计算。
4、order by 子句会让输入的数据强制排序。Order By 子句对于诸如 row_number()，lead()，LAG()等函数是必须的，因为如果数据无序，这些函数的结果就没有任何意义。因此如果有了 Order By 子句，则 count()，min()等计算出来的结果就没有任何意义。
5、如果只使用 partition by 子句,未指定 order by 的话,我们的聚合是分组内的聚合。
6、当同一个 select 查询中存在多个窗口函数时,他们相互之间是没有影响的。

row_number/rank/dense_rank 的区别

这三个窗口函数的使用场景非常多,区别分别为:
1、row_number()从 1 开始，按照顺序，生成分组内记录的序列,row_number()的值不会存在重复,当排序的值相同时,按照表中记录的顺序进行排列
2、rank() 生成数据项在分组中的排名，排名相等会在名次中留下空位
3、dense_rank() 生成数据项在分组中的排名，排名相等会在名次中不会留下空位

注意：
rank 和 dense_rank 的区别在于排名相等时会不会留下空位。

窗口函数语句

<窗口函数>
OVER ([PARTITION BY <列清单>]
ORDER BY <排序用列清单>)

over:窗口函数关键字
partition by:对结果集进行分组
order by:设定结果集的分组数据排序

聚合函数:聚合函数（SUM、AVG、COUNT、MAX、MIN）

内置函数:rank、dense_rank、row_number、percent_rank、grouping sets、first_value、last_value、nth_value 等专用窗口函

1 准备数据

1.1 创建测试的表 test1

```
create table test1(
department varchar(50),
number       numeric,
wages        numeric
);
```

1.2 插入数据到 test1 表中

```
insert into   test1 values
('发展部','8','6000'),
('发展部','10','5200'),
('销售部','1','5000'),
('销售部','3','4800'),
('发展部','7','4200'),
('销售部','4','4800'),
('发展部','9','4500'),
('私立部','5','3500'),
('私立部','2','3900'),
('发展部','11','5200');
```

2 rank over 窗口函数使用

rank():返回行号,对比值重复时行号重复并间断， 即返回 1,2,2,4...

2.1 按照分区查看每行的个数

```
select *,rank() over(partition by department) cn from test1;
```

信息 结果1

department	number	wages	cn
发展部	8	6000	1
发展部	10	5200	1
发展部	7	4200	1
发展部	9	4500	1
发展部	11	5200	1
私立部	2	3900	1
私立部	5	3500	1
销售部	4	4800	1
销售部	1	5000	1
销售部	3	4800	1

2.2 按照分区和排序查看每行的数据

select *,rank() over(partition by department order by wages desc) cn from test1;

信息 结果1

department	number	wages	cn
发展部	8	6000	1
发展部	10	5200	1
发展部	7	4200	1
发展部	9	4500	1
发展部	11	5200	1
私立部	2	3900	1
私立部	5	3500	1
销售部	4	4800	1
销售部	1	5000	1
销售部	3	4800	1

2.3 查看每个部门最高的数据

select * from (
select *,rank() over(partition by department order by wages desc) cn from test1)
tn where cn=1;

信息 结果1

department	number	wages	cn
发展部	8	6000	1
私立部	2	3900	1
销售部	3	4800	1

3 row_number over 窗口函数的使用

row_number():返回行号,对比值重复时行号不重复不间断,即返回 1,2,3,4,5....,不返回 1,2,2,4...

3.1 显示数据的行号

3.1.1 顺序显示行号

select *,row_number() over() cn from test1

信息 结果1

department	number	wages	cn
发展部	8	6000	1
发展部	10	5200	2
销售部	1	5000	3
销售部	3	4800	4
发展部	7	4200	5
销售部	4	4800	6
发展部	9	4500	7
私立部	5	3500	8
私立部	2	3900	9
发展部	11	5200	10

3.1.2 获取一段内的数据

select *,row_number() over() cn from test1 limit 4 OFFSET 2

信息 结果1

department	number	wages	cn
销售部	1	5000	3
销售部	3	4800	4
发展部	7	4200	5
销售部	4	4800	6

3.2 显示分区的个数

select *,row_number() over(partition by department) cn from test1

信息 结果1

department	number	wages	cn
发展部	8	6000	1
发展部	10	5200	2
发展部	7	4200	3
发展部	9	4500	4
发展部	11	5200	5
私立部	2	3900	1
私立部	5	3500	2
销售部	4	4800	1
销售部	1	5000	2
销售部	3	4800	3

3.3 按照 department 分组 wages 排序显示数据

select *,row_number() over(partition by department order by wages desc) cn from test1

信息 结果1

department	number	wages	cn
发展部	8	6000	1
发展部	10	5200	2
发展部	11	5200	3
发展部	9	4500	4
发展部	7	4200	5
私立部	2	3900	1
私立部	5	3500	2
销售部	1	5000	1
销售部	4	4800	2
销售部	3	4800	3

3.4 查看每个部门的最高的数据

select * from (select *,row_number() over(partition by department order by wages desc) cn from test1)
tn where cn =1;

信息 结果1

department	number	wages	cn
发展部	8	6000	1
私立部	2	3900	1
销售部	3	4800	1

4 dense_rank 窗口函数使用

4.1 rank 与 dense_rank 的区别

rank()：返回行号,对比值重复时行号重复并间断， 即返回 1,2,2,4...
dense_rank():返回行号,对比值重复时行号重复但不间断， 即返回 1,2,2,3

注意他两的区别

4.2 dense_rank 窗口函数的显示

select *,dense_rank() over(partition by department order by wages desc) cn from test1;

信息 结果1

department	number	wages	cn
发展部	8	6000	1
发展部	10	5200	2
发展部	11	5200	2
发展部	9	4500	3
发展部	7	4200	4
私立部	2	3900	1
私立部	5	3500	2
销售部	3	4800	1
销售部	1	5000	2
销售部	4	4800	3

4.3 rank 窗口函数的显示

select *,rank() over(partition by department order by wages desc) cn from test1;

信息 结果1

department	number	wages	cn
发展部	8	6000	1
发展部	10	5200	2
发展部	11	5200	2
发展部	9	4500	4
发展部	7	4200	5
私立部	2	3900	1
私立部	5	3500	2
销售部	3	4800	1
销售部	1	5000	2
销售部	4	4800	3

5 rank/row_number/dense_rank 比较

rank():返回行号,对比值重复时行号重复并间断， 即返回 1,2,2,4...

row_number():返回行号,对比值重复时行号不重复不间断,即返回 1,2,3,4,5....,不返回1,2,2,4...
dense_rank():返回行号,对比值重复时行号重复但不间断， 即返回 1,2,2,3

```
select  department,number,wages,
-- 值同排名相同，同时不保留被占用的排名序号，即总排名号不连续
rank() over(partition by department order by wages desc) as rnl,
-- 值同，排名相同，保留下一个的排名序列号，即总排名连续
dense_rank() over(partition by department order by wages desc) as rn2,
-- 强制按列的结果排序，更像行号。
row_number() over(partition by department order by wages desc) as rn3
from test1;
```

信息 结果1

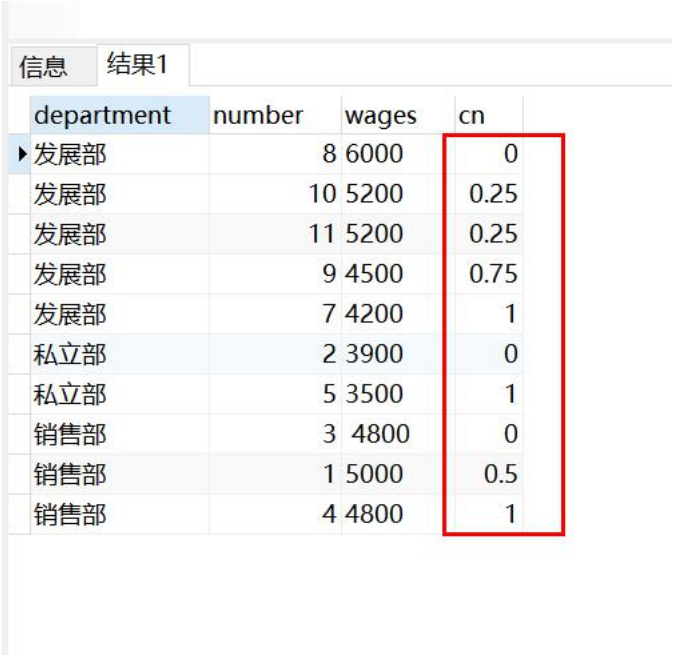
department	number	wages	rnl	rn2	rn3
发展部	8	6000	1	1	1
发展部	10	5200	2	2	2
发展部	11	5200	2	2	3
发展部	9	4500	4	3	4
发展部	7	4200	5	4	5
私立部	2	3900	1	1	1
私立部	5	3500	2	2	2
销售部	1	5000	1	1	1
销售部	4	4800	2	2	2
销售部	3	4800	2	2	3

6 percent_rank 窗口函数的使用

percent_rank()：从当前开始,计算在分组中的比例 (行号-1)*(1/(总记录数-1))

6.1 计算分组中的比例

```
select *,percent_rank() over(partition by department order by wages desc) cn from test1;
```



信息 结果1

department	number	wages	cn
发展部	8	6000	0
发展部	10	5200	0.25
发展部	11	5200	0.25
发展部	9	4500	0.75
发展部	7	4200	1
私立部	2	3900	0
私立部	5	3500	1
销售部	3	4800	0
销售部	1	5000	0.5
销售部	4	4800	1

7 grouping sets 函数的使用

7.1 先按照 wages 分组再按照 department 进行分组

在以下结果中可以看出 wages 有相同的显示了 null 值,如果想做唯一数据去掉该条件即可

```
select department,wages,count(1) from test1 group by grouping
sets(department,(department,wages)) order by department;
```

58

信息 结果1

department	wages	count
发展部	4200	1
发展部	4500	1
发展部	5200	2
发展部	6000	1
发展部	(Null)	5
私立部	3500	1
私立部	3900	1
私立部	(Null)	2
销售部	4800	1
销售部	5000	1
销售部	4800	1
销售部	(Null)	3

8 聚合函数+窗口函数使用

8.1 查看一个部门的个数

select department,number,wages,count(*) over() from test1 where department = '发展部';

信息 结果1

department	number	wages	count
发展部	8	6000	5
发展部	10	5200	5
发展部	7	4200	5
发展部	9	4500	5
发展部	11	5200	5

8.2 统计每个部门的 wages 之和

select department,number,wages,sum(wages) over(partition by department) from test1;

信息 结果1

department	number	wages	sum
发展部	8	6000	25100
发展部	10	5200	25100
发展部	7	4200	25100
发展部	9	4500	25100
发展部	11	5200	25100
私立部	2	3900	7400
私立部	5	3500	7400
销售部	4	4800	14600
销售部	1	5000	14600
销售部	3	4800	14600

8.3 按照排序统计每个部门的 wages 之和

select department,number,wages,sum(wages) over(partition by department ORDER BY wages desc) from test1;

信息 结果1

department	number	wages	sum
发展部	8	6000	6000
发展部	10	5200	16400
发展部	11	5200	16400
发展部	9	4500	20900
发展部	7	4200	25100
私立部	2	3900	3900
私立部	5	3500	7400
销售部	1	5000	5000
销售部	4	4800	14600
销售部	3	4800	14600

8.4 按照分组和排序统计数据

select department,number,wages,
sum(wages) over() sum1,
sum(wages) over (order by department) sum2,
sum(wages) over (partition by department) sum3,
sum(wages) over (partition by department order by wages desc) sum4

from test1
order by department desc;

信息 结果1

department	number	wages	sum1	sum2	sum3	sum4
销售部	1	5000	47100	47100	14600	5000
销售部	3	4800	47100	47100	14600	14600
销售部	4	4800	47100	47100	14600	14600
私立部	2	3900	47100	32500	7400	3900
私立部	5	3500	47100	32500	7400	7400
发展部	8	6000	47100	25100	25100	6000
发展部	10	5200	47100	25100	25100	16400
发展部	9	4500	47100	25100	25100	20900
发展部	11	5200	47100	25100	25100	16400
发展部	7	4200	47100	25100	25100	25100

8.5 window 子句使用

8.5.1 windom 子句的说明

我们在上面已经通过使用 partition by 子句将数据进行了分组的处理,如果我们想要更细粒度的划分,我们就要引入 window 子句了。

window 子句:
- preceding（preceding）：往前
- following(following)：往后
- current row(current row)：当前行
- unbounded(unbounded)：起点
- unbounded preceding 表示从前面的起点
- unbounded following：表示到后面的终点

8.5.2 执行的 SQL 语句

```
select department,number,wages,
--所有行相加
sum(wages) over() as sum1,
-- 统计按照 department 组内的 sum
sum(wages) over(partition by department) as sum2,
-- 统计按照 department 分组 wages 排序 sum
sum(wages) over(partition by department order by wages) as sum3,
-- 表示从前面的起点到当前的行统计 sum
```

sum(wages) over(partition by department order by wages rows between unbounded preceding and current row) as sum4,
-- 表示往前 1 行到当前的行的统计
sum(wages) over(partition by department order by wages rows between 1 preceding and current row) as sum5,
-- 表示往前 1 行到往后 1 行的统计
sum(wages) over(partition by department order by wages rows between 1 preceding and 1 following)as sum6,
-- 表示当前的行到后面的重点统计
sum(wages) over(partition by department order by wages rows between current row and unbounded following) as sum7
from test1;

信息 结果1

department	number	wages	sum1	sum2	sum3	sum4	sum5	sum6	sum7
发展部	7	4200	47100	25100	4200	4200	4200	8700	25100
发展部	9	4500	47100	25100	8700	8700	8700	13900	20900
发展部	10	5200	47100	25100	19100	13900	9700	14900	16400
发展部	11	5200	47100	25100	19100	19100	10400	16400	11200
发展部	8	6000	47100	25100	25100	25100	11200	11200	6000
私立部	5	3500	47100	7400	3500	3500	3500	7400	7400
私立部	2	3900	47100	7400	7400	7400	7400	7400	3900
销售部	4	4800	47100	14600	9600	4800	4800	9600	14600
销售部	3	4800	47100	14600	9600	9600	9600	14600	9800
销售部	1	5000	47100	14600	14600	14600	9800	9800	5000

8.6 窗口函数中的序列函数

8.6.1 序列函数的说明

常用的序列函数有下面几个:
ntile(ntile)
ntile(n),用于将分组数据按照顺序切分成 n 片，返回当前切片值
ntile 不支持 rows between，
比如 ntile(2) over(partition by cost order by name rows between 3 preceding and current row)

8.6.2 执行的语句

select department,number,wages,
-- 全局数据进行分割
ntile(3) over() as sample1,
-- 按照分组,将数据今个

```
ntile(3) over(partition by department) sample2,
-- 按照排序对数据进行分割
ntile(3) over(order by department) sample3,
-- 按照分组和排序进行数据分割
ntile(3) over(partition by department order by wages) sample4
from test1 order by department desc;
```

信息 结果1

department	number	wages	sample1	sample2	sample3	sample4
销售部	4	4800	2	1	3	1
销售部	1	5000	1	2	3	3
销售部	3	4800	1	3	3	2
私立部	5	3500	3	2	2	1
私立部	2	3900	3	1	2	2
发展部	7	4200	2	2	1	1
发展部	9	4500	2	2	1	1
发展部	11	5200	3	3	2	2
发展部	10	5200	1	1	1	2
发展部	8	6000	1	1	1	3

9 first_value\last_value 使用

9.1 first_value 和 last_value 说明

first_value 取分组内排序后,截止到当前行,第一个值
last_value 取分组内排序后,截止到当前行,最后一个值,如果有重复值获取获取最后一个

以下函数在 greenplum 才可使用
nth_value 用来取结果集每一个分组的指定行数的字段值。(如果不存在返回 null)

9.2 执行的 SQL

```
select department,number,wages,
first_value(number) over(partition by department order by wages desc)as f1,
last_value(number) over(partition by department order by wages desc) as f2
from test1;
```

信息 结果1

department	number	wages	f1	f2
发展部	8	6000	8	8
发展部	10	5200	8	11
发展部	11	5200	8	11
发展部	9	4500	8	9
发展部	7	4200	8	7
私立部	2	3900	2	2
私立部	5	3500	2	5
销售部	1	5000	1	1
销售部	4	4800	1	3
销售部	3	4800	1	3