OVER函数介绍 - CSDN博客

开窗函数指定了分析函数工作的数据窗口大小，这个数据窗口大小可能会随着行的变化而变化，举例如下：

1：over后的写法：

over（order by salary） 按照salary排序进行累计，order by是个默认的开窗函数
over（partition by deptno）按照部门分区
over（partition by deptno order by salary）

2：开窗的窗口范围：

over（order by salary range between 5 preceding and 5 following）：窗口范围为当前行数据幅度减5加5后的范围内的。

举例：

--sum(s)over(order by s range between 2 preceding and 2 following)表示加2或2的范围内的求和

```
 select name,class,s, sum(s)over(order by s range between 2 preceding and 2 following) mm from t2
adf         3         45         45   --45加2减2即43到47，但是s在这个范围内只有45
asdf        3         55         55
cfe         2         74         74
3dd         3         78         158 --78在76到80范围内有78，80，求和得158
fda         1         80         158
gds         2         92         92
ffd         1         95         190
dss         1         95         190
ddd         3         99         198
gf           3          99          198
```

over（order by salaryrowsbetween 5 preceding and 5 following）：窗口范围为当前行前后各移动5行。

举例：

--sum(s)over(order by s rows between 2 preceding and 2 following)表示在上下两行之间的范围内

```
select name,class,s, sum(s)over(order by s rows between 2 preceding and 2 following) mm from t2
adf         3         45         174   （45+55+74=174）
asdf        3         55         252    （45+55+74+78=252）
cfe         2         74         332     （74+55+45+78+80=332）
3dd         3         78         379     （78+74+55+80+92=379）
fda         1         80         419
gds         2         92         440
ffd         1         95         461
dss         1         95         480
ddd         3         99         388
gf          3         99         293
```

over（order by salary range between unbounded preceding and unbounded following）或者
over（order by salary rows between unbounded preceding and unbounded following）：窗口不做限制

3、与over函数结合的几个函数介绍

row_number()over()、rank()over()和dense_rank()over()函数的使用

下面以班级成绩表t2来说明其应用

t2表信息如下：

```
cfe     2     74
dss     1     95
ffd     1     95
fda     1     80
gds     2     92
gf      3     99
ddd     3     99
adf     3     45
asdf    3     55
3dd     3     78
```

```
select * from
    (
    select name,class,s,rank()over(partition by class order by s desc) mm from t2
    )
    where mm=1;
```

得到的结果是:

```
dss     1     95     1
ffd     1     95     1
gds     2     92     1
gf      3     99     1
ddd     3     99     1
```

注意：

1.在求第一名成绩的时候，不能用row_number()，因为如果同班有两个并列第一，row_number()只返回一个结果;

```
select * from
    (
    select name,class,s,row_number()over(partition by class order by s desc) mm from t2
    )
    where mm=1;
```

```
1     95     1  --95有两名但是只显示一个
2     92     1
3     99     1 --99有两名但也只显示一个
```

2.rank()和dense_rank()可以将所有的都查找出来：

如上可以看到采用rank可以将并列第一名的都查找出来；

rank()和dense_rank()区别：

--rank()是跳跃排序，有两个第二名时接下来就是第四名；

```
select name,class,s,rank()over(partition by class order by s desc) mm from t2
dss     1     95     1
ffd     1     95     1
fda     1     80     3 --直接就跳到了第三
gds     2     92     1
cfe     2     74     2
gf      3     99     1
ddd     3     99     1
3dd     3     78     3
asdf    3     55     4
adf     3     45     5
```

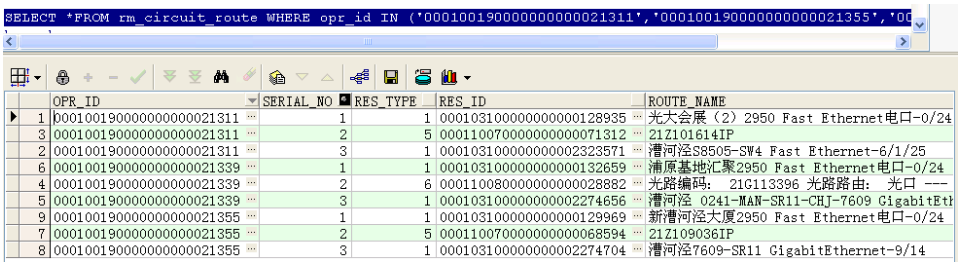
--dense_rank()1是连续排序，有两个第二名时仍然跟着第三名

```
select name,class,s,dense_rank()over(partition by class order by s desc) mm from t2
dss         1         95         1
ffd         1         95         1
fda         1         80         2 --连续排序（仍为2）
gds         2         92         1
cfe         2         74         2
gf          3         99         1
ddd         3         99         1
3dd         3         78         2
asdf        3         55         3
adf         3         45         4
```

--sum()over（）的使用

```
select name,class,s, sum(s)over(partition by class order by s desc) mm from t2 --根据班级进行分数求和
dss         1         95         190  --由于两个95都是第一名，所以累加时是两个第一名的相加
ffd         1         95         190
fda         1         80         270  --第一名加上第二名的
gds         2         92         92
cfe         2         74         166
gf          3         99         198
ddd         3         99         198
3dd         3         78         276
asdf        3         55         331
adf         3         45         376
```

first_value() over()和last_value() over()的使用



SELECT *FROM rm_circuit_route WHERE opr_id IN ('000100190000000000021311','000100190000000000021355','00

	OPR_ID	SERIAL_NO	RES_TYPE	RES_ID	ROUTE_NAME
1	00010019000000000021311	1	1	00010310000000000128935	光大会展（2）2950 Fast Ethernet电口-0/24
3	00010019000000000021311	2	5	00011007000000000071312	21Z101614IP
2	00010019000000000021311	3	1	00010310000000002323571	漕河泾S8505-SW4 Fast Ethernet-6/1/25
6	00010019000000000021339	1	1	00010310000000000132659	浦原基地汇聚2950 Fast Ethernet电口-0/24
4	00010019000000000021339	2	6	00011008000000000028882	光路编码: 21G113396 光路路由: 光口 ---
5	00010019000000000021339	3	1	00010310000000002274656	漕河泾 0241-MAN-SR11-CHJ-7609 GigabitEth
9	00010019000000000021355	1	1	00010310000000000129969	新漕河泾大厦2950 Fast Ethernet电口-0/24
7	00010019000000000021355	2	5	00011007000000000068594	21Z109036IP
8	00010019000000000021355	3	1	00010310000000002274704	漕河泾7609-SR11 GigabitEthernet-9/14

--找出这三条电路每条电路的第一条记录类型和最后一条记录类型

```
SELECT opr_id,res_type,
       first_value(res_type) over(PARTITION BY opr_id ORDER BY res_type) low,
       last_value(res_type) over(PARTITION BY opr_id ORDER BY res_type rows BETWEEN unbounded
preceding AND unbounded following) high
  FROM rm_circuit_route
WHERE opr_id IN ('000100190000000000021311','000100190000000000021355','000100190000000000021339')
 ORDER BY opr_id;
```

	OPR_ID	RES_TYPE	LOW	HIGH
1	00010019000000000021311	1	1	5
2	00010019000000000021311	1	1	5
3	00010019000000000021311	5	1	5
4	00010019000000000021339	1	1	6
5	00010019000000000021339	1	1	6
6	00010019000000000021339	6	1	6
7	00010019000000000021355	1	1	5
8	00010019000000000021355	1	1	5
9	00010019000000000021355	5	1	5

注：*rows BETWEEN unbounded preceding AND unbounded following* 的使用

--取last_value时不使用*rows BETWEEN unbounded preceding AND unbounded following的结果*

```
SELECT opr_id,res_type,
       first_value(res_type) over(PARTITION BY opr_id ORDER BY res_type) low,
       last_value(res_type) over(PARTITION BY opr_id ORDER BY res_type) high
  FROM rm_circuit_route
 WHERE opr_id IN ('00010019000000000021311','00010019000000000021355','00010019000000000021339')
 ORDER BY opr_id;
```

如下图可以看到，如果不使用

rows BETWEEN unbounded preceding AND unbounded following，取出的last_value由于与res_type进行进行排列，因此取出的电路的最后一行记录的类型就不是按照电路的范围提取了，而是以res_type为范围进行提取了。

	OPR_ID	RES_TYPE	LOW	HIGH
1	00010019000000000021311	1	1	1
2	00010019000000000021311	1	1	1
3	00010019000000000021311	5	1	5
4	00010019000000000021339	1	1	1
5	00010019000000000021339	1	1	1
6	00010019000000000021339	6	1	6
7	00010019000000000021355	1	1	1
8	00010019000000000021355	1	1	1
9	00010019000000000021355	5	1	5

在first_value和last_value中ignore nulls的使用

数据如下：

```
SELECT * FROM rm_circuit_route WHERE opr_id='00012559000000000273366';
```

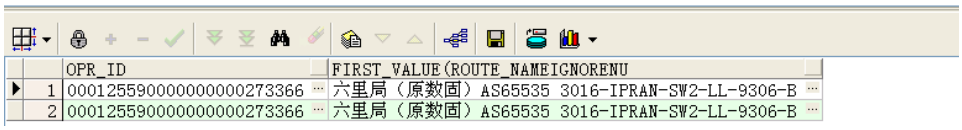
	OPR_ID	SERIAL_NO		RES_I	ROUTE_NAME
1	00012559000000000273366	1	5	00011	
2	00012559000000000273366	2	1	00010	六里局（原数固）AS65535 3016-IPRAN-SW2-LL-9306-B

取出该电路的第一条记录，加上*ignore nulls后，如果第一条是判断的那个字段是空的，则默认取下一条，结果如下所示：*

```
SELECT  opr_id,
       first_value(route_name ignore nulls) OVER(order by opr_id)
  from rm_circuit_route
WHERE opr_id='00012559000000000273366';
```

	OPR_ID	FIRST_VALUE(ROUTE_NAMEIGNORENU
1	00012559000000000273366	六里局（原数固）AS65535 3016-IPRAN-SW2-LL-9306-B
2	00012559000000000273366	六里局（原数固）AS65535 3016-IPRAN-SW2-LL-9306-B

```
SELECT  opr_id,
       first_value(route_name ignore nulls)
       OVER(order by ROWNUM rows BETWEEN unbounded preceding AND unbounded following)
  from rm_circuit_route
WHERE opr_id='000125590000000000273366';
```



	OPR_ID	FIRST_VALUE(ROUTE_NAMEIGNORENU
1	000125590000000000273366	六里局（原数固）AS65535 3016-IPRAN-SW2-LL-9306-B
2	000125590000000000273366	六里局（原数固）AS65535 3016-IPRAN-SW2-LL-9306-B

--lag() over()函数用法（取出前n行数据）

lag(expresstion,,)

```
with a as
(select 1 id,'a' name from dual
 union
 select 2 id,'b' name from dual
 union
 select 3 id,'c' name from dual
 union
 select 4 id,'d' name from dual
 union
 select 5 id,'e' name from dual
)
select id,name,lag(id,1,'')over(order by name) from a;
```

--lead() over()函数用法（取出后N行数据）

lead(expresstion,,)

```
with a as

(select 1 id,'a' name from dual

 union

 select 2 id,'b' name from dual

 union

 select 3 id,'c' name from dual

 union

 select 4 id,'d' name from dual

 union

 select 5 id,'e' name from dual

)

select id,name,lead(id,1,'')over(order by name) from a;
```

--ratio_to_report(a)函数用法 Ratio_to_report() 括号中就是分子，over() 括

号中就是分母

```
with a as (select 1 a from dual
           union all
select 1 a from dual
           union  all
select 1 a from dual
           union all
select 2 a from dual
           union all
select 3 a from dual
           union all
select 4 a from dual
           union all
select 4 a from dual
           union all
select 5 a from dual
           )
select a, ratio_to_report(a)over(partition by a) b from a
order by a;
with a as (select 1 a from dual
           union all
select 1 a from dual
           union  all
select 1 a from dual
```

```
          union all
select 2 a from dual
          union all
select 3 a from dual
          union all
select 4 a from dual
          union all
select 4 a from dual
          union all
select 5 a from dual
          )
select a, ratio_to_report(a)over() b from a --分母缺省就是整个占比
order by a;
with a as (select 1 a from dual
          union all
select 1 a from dual
          union  all
select 1 a from dual
          union all
select 2 a from dual
          union all
select 3 a from dual
          union all
select 4 a from dual
```

```
            union all

select 4 a from dual

            union all

select 5 a from dual

            )

select a, ratio_to_report(a)over() b from a

group by a order by a;--分组后的占比
```

percent_rank用法

计算方法：所在组排名序号-1除以该组所有的行数-1，如下所示自己计算的pr1与通过percent_rank函数得到的值是一样的：

```
SELECT a.deptno,
       a.ename,
       a.sal,
       a.r,
       b.n,
       (a.r-1)/(n-1) pr1,
       percent_rank() over(PARTITION BY a.deptno ORDER BY a.sal) pr2
  FROM (SELECT deptno,
               ename,
               sal,
               rank() over(PARTITION BY deptno ORDER BY sal) r --计算出在组中的排名序号
          FROM emp
         ORDER BY deptno, sal) a,
       (SELECT deptno, COUNT(1) n FROM emp GROUP BY deptno) b --按部门计算每个部门的所有成员数
 WHERE a.deptno = b.deptno;
```

	DEPTNO	ENAME	SAL	R	N	PR1	PR2
1	10	MILLER	1300.00	1	3	0	0
2	10	CLARK	2450.00	2	3	0.5	0.5
3	10	KING	5000.00	3	3	1	1
4	20	SMITH	800.00	1	5	0	0
5	20	ADAMS	1100.00	2	5	0.25	0.25
6	20	JONES	2975.00	3	5	0.5	0.5
7	20	SCOTT	3000.00	4	5	0.75	0.75
8	20	FORD	3000.00	4	5	0.75	0.75
9	30	JAMES	950.00	1	6	0	0
10	30	MARTIN	1250.00	2	6	0.2	0.2
11	30	WARD	1250.00	2	6	0.2	0.2
12	30	TURNER	1500.00	4	6	0.6	0.6
13	30	ALLEN	1600.00	5	6	0.8	0.8
14	30	BLAKE	2850.00	6	6	1	1

cume_dist函数

计算方法：所在组排名序号除以该组所有的行数，但是如果存在并列情况，则需加上并列的个数-1，
如下所示自己计算的pr1与通过percent_rank函数得到的值是一样的：

```
SELECT a.deptno,
       a.ename,
       a.sal,
       a.r,
       b.n,
       c.rn,
       (a.r + c.rn - 1) / n pr1,
       cume_dist() over(PARTITION BY a.deptno ORDER BY a.sal) pr2
  FROM (SELECT deptno,
               ename,
               sal,
               rank() over(PARTITION BY deptno ORDER BY sal) r
          FROM emp
         ORDER BY deptno, sal) a,
       (SELECT deptno, COUNT(1) n FROM emp GROUP BY deptno) b,
       (SELECT deptno, r, COUNT(1) rn,sal
          FROM (SELECT deptno,sal,
                       rank() over(PARTITION BY deptno ORDER BY sal) r
                  FROM emp)
         GROUP BY deptno, r,sal
         ORDER BY deptno) c --c表就是为了得到每个部门员工工资的一样的个数
 WHERE a.deptno = b.deptno
   AND a.deptno = c.deptno(+)
   AND a.sal = c.sal;
```

DEPTNO	ENAME	SAL	R	N	RN	PR1	PR2
10	MILLER	1300.00	1	3	1	0.333333333333333	0.333333333333333
10	CLARK	2450.00	2	3	1	0.666666666666667	0.666666666666667
10	KING	5000.00	3	3	1	1	1
20	SMITH	800.00	1	5	1	0.2	0.2
20	ADAMS	1100.00	2	5	1	0.4	0.4
20	JONES	2975.00	3	5	1	0.6	0.6
20	FORD	3000.00	4	5	2	1	1
20	SCOTT	3000.00	4	5	2	1	1
30	JAMES	950.00	1	6	1	0.166666666666667	0.166666666666667
30	WARD	1250.00	2	6	2	0.5	0.5
30	MARTIN	1250.00	2	6	2	0.5	0.5
30	TURNER	1500.00	4	6	1	0.666666666666667	0.666666666666667
30	ALLEN	1600.00	5	6	1	0.833333333333333	0.833333333333333
30	BLAKE	2850.00	6	6	1	1	1

percentile_cont函数

含义：输入一个百分比（该百分比就是按照percent_rank函数计算的值），返回该百分比位置的平均值
如下，输入百分比为0.7，因为0.7介于0.6和0.8之间，因此返回的结果就是0.6对应的sal的1500加上0.8对应的sal的1600平均

```
SELECT ename,
       sal,
       deptno,
       percentile_cont(0.7) within GROUP(ORDER BY sal) over(PARTITION BY deptno) "Percentile_Cont",
       percent_rank() over(PARTITION BY deptno ORDER BY sal) "Percent_Rank"
  FROM emp
 WHERE deptno IN (30, 60);
```

	ENAME	SAL	DEPTNO	Percentile_Cont	Percent_Rank
1	JAMES	950.00	30	1550	0
2	WARD	1250.00	30	1550	0.2
3	MARTIN	1250.00	30	1550	0.2
4	TURNER	1500.00	30	1550	0.6
5	ALLEN	1600.00	30	1550	0.8
6	BLAKE	2850.00	30	1550	1

若输入的百分比为0.6，则直接0.6对应的sal值，即1500

```
SELECT ename,
       sal,
       deptno,
       percentile_cont(0.6) within GROUP(ORDER BY sal) over(PARTITION BY deptno) "Percentile_Cont",
       percent_rank() over(PARTITION BY deptno ORDER BY sal) "Percent_Rank"
  FROM emp
 WHERE deptno IN (30, 60);
```

	ENAME	SAL	DEPTNO	Percentile_Cont	Percent_Rank
1	JAMES	950.00	30	1500	0
2	WARD	1250.00	30	1500	0.2
3	MARTIN	1250.00	30	1500	0.2
4	TURNER	1500.00	30	1500	0.6
5	ALLEN	1600.00	30	1500	0.8
6	BLAKE	2850.00	30	1500	1

PERCENTILE_DISC函数

功能描述：返回一个与输入的分布百分比值相对应的数据值，分布百分比的计算方法见函数CUME_DIST，如果没有正好对应的数据值，就取大于该分布值的下一个值。

注意：本函数与PERCENTILE_CONT的区别在找不到对应的分布值时返回的替代值的计算方法不同

SAMPLE：下例中0.7的分布值在部门30中没有对应的Cume_Dist值，所以就取下一个分布值0.83333333所对应的SALARY来替代

```
SELECT ename,
       sal,
       deptno,
       percentile_disc(0.7) within GROUP(ORDER BY sal) over(PARTITION BY deptno) "Percentile_Disc",
       cume_dist() over(PARTITION BY deptno ORDER BY sal) "Cume_Dist"
  FROM emp
 WHERE deptno IN (30, 60);
```

	ENAME	SAL	DEPTNO	Percentile_Disc	Cume_Dist
1	JAMES	950.00	30	1600	0.166666666666667
2	WARD	1250.00	30	1600	0.5
3	MARTIN	1250.00	30	1600	0.5
4	TURNER	1500.00	30	1600	0.666666666666667
5	ALLEN	1600.00	30	1600	0.833333333333333
6	BLAKE	2850.00	30	1600	1