

Window Functions

CSC 256, SQL Programming

Window functions

- A window function performs a calculation on a subset of rows related to the current row.
- Common window functions
 - `row_number`: number of the current row within its partition
 - `rank`: rank of the current row based on ordering
 - `dense_rank`: rank of the current row without gaps

Window Function Syntax

- Basic syntax:

```
window_func(arg1, arg2, ...) OVER (  
  [PARTITION BY expression]  
  [ORDER BY expression [ASC | DESC]  
  )
```

- PARTITION BY divides rows into multiple groups that the window function is applied to. If left out, the partition is the whole table.
- ORDER BY specifies the order of the rows in each partition

Example Table

id	c1	c2
1	A	2
2	B	5
3	C	5
4	A	7
5	C	3
6	B	5
7	A	2

Window Function Example

```
select
  id,
  c2,
  row_number() over (order by c2),
  rank() over (order by c2),
  dense_rank() over (order by c2)
from t;
```

id	c2	row_number	rank	dense_rank
7	2	1	1	1
1	2	2	1	1
5	3	3	3	2
2	5	4	4	3
3	5	5	4	3
6	5	6	4	3
4	7	7	7	4

WINDOW

- The window can be defined with a WINDOW clause.
- Previous example

```
select
  id,
  c2,
  row_number() over w,
  rank() over w,
  dense_rank() over w
from t
window w as (order by c2);
```

Aggregate Window Functions

- Aggregate functions are computed over the window.

```
select id, c2, c1,  
       avg(c2) over ()  
from t;
```

id	c2	c1	avg
1	2	A	4.1428571428571429
2	5	B	4.1428571428571429
3	5	C	4.1428571428571429
4	7	A	4.1428571428571429
5	3	C	4.1428571428571429
6	5	B	4.1428571428571429
7	2	A	4.1428571428571429

PARTITION Example

- PARTITION groups records into window frames based on an expression.

```
select id, c2, c1,  
       avg(c2) over (partition by c1)  
from t;
```

id	c2	c1	avg
4	7	A	3.6666666666666667
1	2	A	3.6666666666666667
7	2	A	3.6666666666666667
6	5	B	5.0000000000000000
2	5	B	5.0000000000000000
5	3	C	4.0000000000000000
3	5	C	4.0000000000000000

Window Frame Syntax

- The frame clause is an optional third part of the window that specifies the set of rows included in the window frame.

- Syntax:

```
{ RANGE | ROWS } frame_start |
```

```
{ RANGE | ROWS } BETWEEN frame_start AND frame_end
```

- Values for frame_start and frame_end:

- UNBOUND PRECEDING
- value PRECEDING
- CURRENT ROW
- value FOLLOWING
- UNBOUNDED FOLLOWING

Window Frame Example

- Note the duplicate values:

```
select id, c2, c1,  
       sum(c2) over (partition by c1 order by c2)  
from t;
```

id	c2	c1	sum
1	2	A	4
7	2	A	4
4	7	A	11
6	5	B	10
2	5	B	10
5	3	C	3
3	5	C	8

Window Frame Example (continued)

- We can get a cumulative sum by adding a window frame:

```
select id, c2, c1,  
       sum(c2) over (partition by c1 order by c2  
                    rows unbounded preceding)  
from t;
```

id	c2	c1	sum
1	2	A	2
7	2	A	4
4	7	A	11
6	5	B	5
2	5	B	10
5	3	C	3
3	5	C	8

Additional Example

- Box filter smoothing – average over a sliding window:

```
select id, c2, c1,  
       sum(c2) over (order by id  
                     rows between 2 preceding and current row)  
from t;
```

id	c2	c1	sum
1	2	A	2
2	5	B	7
3	5	C	12
4	7	A	17
5	3	C	15
6	5	B	15
7	2	A	10

LAG and LEAD

- The LAG and LEAD functions return a value that is offset based on the current window before or after the current row respectively
- Syntax:

`[lag|lead](value, offset, default_value)`

- The default offset is 1
- The default default_value is NULL

LAG Example

```
select id, c2, c1,  
       lag(c2) over ()  
from t;
```

id	c2	c1	lag
1	2	A	
2	5	B	2
3	5	C	5
4	7	A	5
5	3	C	7
6	5	B	3
7	2	A	5

LEAD Example

```
select id, c2, c1,  
       lead(c2, 2, 0) over ()  
from t;
```

id	c2	c1	lead
1	2	A	5
2	5	B	7
3	5	C	3
4	7	A	5
5	3	C	2
6	5	B	0
7	2	A	0