

# Шпаргалка по оконным функциям SQL

## Список Оконных Функций

### Агрегатные функции

- `avg()`
- `count()`
- `max()`
- `min()`
- `sum()`

### Функции Ранжирования

- `row_number()`
- `rank()`
- `dense_rank()`

### Функции Распределения

- `percent_rank()`
- `cume_dist()`

### Аналитические Функции

- `lead()`
- `lag()`
- `ntile()`
- `first_value()`
- `last_value()`
- `nth_value()`

## АГРЕГАТНЫЕ ФУНКЦИИ

- `avg(expr)` – среднее значение для строк в пределах окна
- `count(expr)` – количество значений для строк в пределах окна
- `max(expr)` – максимальное значение в пределах окна
- `min(expr)` – минимальное значение в пределах окна
- `sum(expr)` – сумма значений в пределах окна

**ORDER BY и Оконная Рамка:** агрегатные функции не требуют применения оператора ORDER BY. Они работают с компонентами окна (ROWS, RANGE, GROUPS).

## ФУНКЦИИ РАНЖИРОВАНИЯ

- `row_number()` – уникальный номер в каждом разделе с последовательной нумерацией строк
- `rank()` – ранжирование раздела и входящих значений, может содержать пробелы
- `dense_rank()` – ранжирование раздела и входящих значений, не содержит пробелы

city	price	row_number	rank	dense_rank
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

**ORDER BY и Оконная Рамка:** для функций `rank()` и `dense_rank()` требуется оператор ORDER BY, для `row_number()` этот оператор не обязателен. Функции ранжирования не работают с компонентами окна (ROWS, RANGE, GROUPS).

## АНАЛИТИЧЕСКИЕ ФУНКЦИИ

- `lead(expr, offset, default)` – функция для смещения `n` строк после текущей строки; параметры `offset` и `default` являются дополнительными; по умолчанию: `offset = 1`, `default = NULL`
- `lag(expr, offset, default)` – функция для смещения `n` строк перед текущей строкой; параметры `offset` и `default` являются дополнительными; по умолчанию: `offset = 1`, `default = NULL`

`lag(sold) OVER(ORDER BY month)`

order by month	month	sold	lag(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	lead(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	lag(sold, 2, 0)
↓	1	500	0
	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	lead(sold, 2, 0)
↓	1	500	400
	2	300	100
	3	400	500
	4	100	0
	5	500	0

- `ntile(n)` – делит строки в разделе на `n` групп с одинаковым количеством строк; присваивает каждому ряду его группу

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

**ORDER BY и Оконная Рамка:** для функций `ntile()`, `lead()`, и `lag()` требуется оператор ORDER BY. Они не работают с компонентами окна (ROWS, RANGE, GROUPS).

## ФУНКЦИИ РАСПРЕДЕЛЕНИЯ

- `percent_rank()` – ранжирование строк в процентах, значение в интервале от [0, 1]:  $(\text{ранк} - 1) / (\text{общее кол. во рядов} - 1)$
- `cume_dist()` – кумулятивное распределение значения в группе; количество строк со значениями меньше/ равными значению текущей строки, деленное на общее число строк; значение в интервале (0, 1]

`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% значений меньше или равны этому значению

`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

← без этой строки 50% значений меньше или равны значению этой строки

**ORDER BY и Оконная Рамка:** для функций распределения требуется оператор ORDER BY. Они не работают с компонентами окна (ROWS, RANGE, GROUPS).

- `first_value(expr)` – значение первой строки в пределах окна
- `last_value(expr)` – значение последней строки в пределах окна

`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 DIAПАЗОН МЕЖДУ ПРЕДЫДУЩЕЙ И ПОСЛЕДУЮЩЕЙ СТРОКОЙ БЕЗ ОГРАНИЧЕНИЙ)`

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

Заметка: обычно ДИАПАЗОН МЕЖДУ ПРЕДЫДУЩИХ БЕЗ ОГРАНИЧЕНИЙ и ПОСЛЕДУЮЩИХ БЕЗ ОГРАНИЧЕНИЙ используются с функцией `last_value()`. Со стандартной оконной рамкой для ORDER BY, ДИАПАЗОНА С ПРЕДЫДУЩИМИ СТРОКАМИ БЕЗ ОГРАНИЧЕНИЙ, функция `last_value()` возвращает значения для текущей строки

- `nth_value(expr, n)` – значение для `n`-ной строки в пределах окна, где `n` – целое число

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

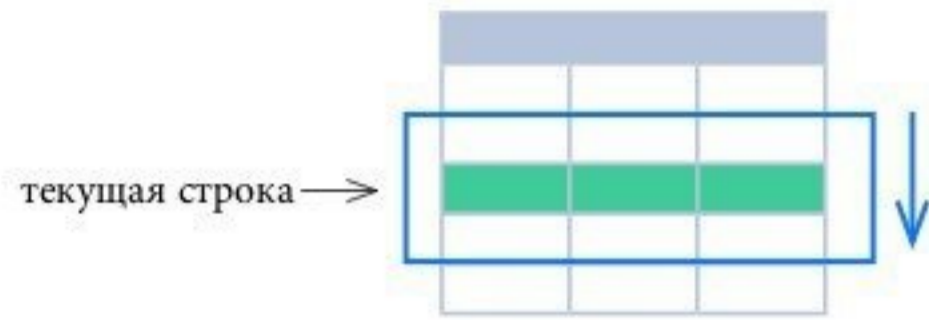
city	month	sold	nth_value
Paris	1	500	300
Paris	2	300	300
Paris	3	400	300
Rome	2	200	300
Rome	3	300	300
Rome	4	500	300
Rome	5	300	300
London	1	100	NULL

**ORDER BY и Оконная Рамка:** для функций `first_value()`, `last_value()`, и `nth_value()` оператор ORDER BY не требуется. Они работают с компонентами окна (ROWS, RANGE, GROUPS).

# Оконные функции в SQL

## ОКОННЫЕ ФУНКЦИИ

осуществляют вычисления с помощью подвижного окна, в котором набор строк тем или иным образом относится к текущей строке.



## АГРЕГАТНЫЕ ФУНКЦИИ VS ОКОННЫЕ ФУНКЦИИ

в отличие от агрегатных функций, оконные функции не преобразовывают все строки в одну



## PARTITION BY

делят строки на группы, называемые разделами, к которым применяется оконная функция

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

Стандартное Разделение: без оператора PARTITION BY вся таблица является разделом.

## ORDER BY

определяет порядок строк в каждом разделе, к которому применяется оконная функция

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

Стандартная сортировка: без оператора ORDER BY порядок строк в каждом разделе произвольный

## СИНТАКСИС

```
SELECT city, month,
       sum(sold) OVER (
         PARTITION BY city
         ORDER BY month
         RANGE UNBOUNDED PRECEDING) total
FROM sales;
```

```
SELECT <column_1>, <column_2>,
       <window_function>() OVER (
         PARTITION BY <...>
         ORDER BY <...>
         <window_frame>) <window_column_alias>
FROM <table_name>;
```

## Пример Окна

```
SELECT country, city,
       rank() OVER country_sold_avg
FROM sales
WHERE month BETWEEN 1 AND 6
GROUP BY country, city
HAVING sum(sold) > 10000
WINDOW country_sold_avg AS (
  PARTITION BY country
  ORDER BY avg(sold) DESC)
ORDER BY country, city;
```

```
SELECT <column_1>, <column_2>,
       <window_function>() OVER <window_name>
FROM <table_name>
WHERE <...>
GROUP BY <...>
HAVING <...>
WINDOW <window_name> AS (
  PARTITION BY <...>
  ORDER BY <...>
  <window_frame>)
ORDER BY <...>;
```

Операторы PARTITION BY, ORDER BY и обозначение рамки окна являются произвольными в данных примерах.

## КОМАНДЫ SQL В ЛОГИЧЕСКОМ ПОРЯДКЕ

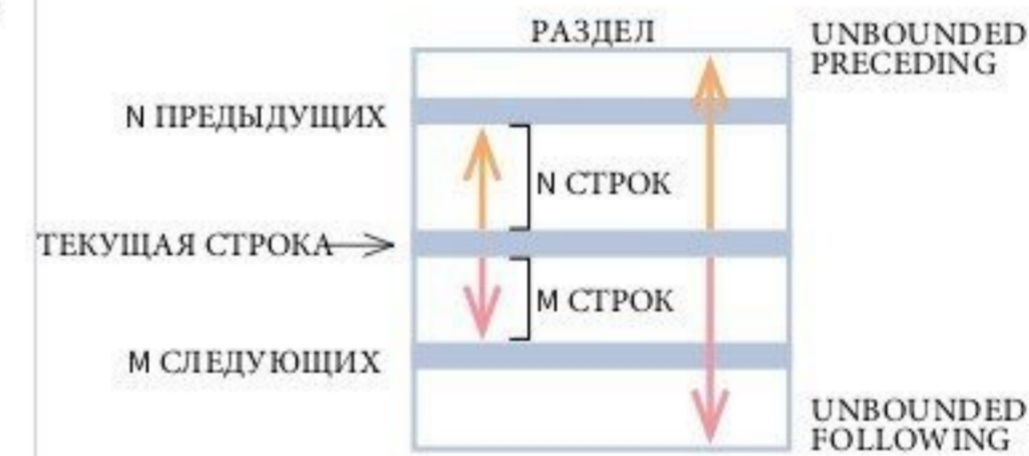
1. FROM, JOIN
2. WHERE
3. GROUP BY
4. агрегатные функции
5. HAVING
6. оконные функции
7. SELECT
8. DISTINCT
9. UNION/INTERSECT/EXCEPT
10. ORDER BY
11. OFFSET
12. LIMIT/FETCH/TOP

Вы можете использовать оконные функции с операторами SELECT и ORDER BY. Однако с операторами FROM, WHERE, GROUP BY и HAVING они не работают.

## ОКОННАЯ РАМКА

это набор строк, которые тем или иным образом относятся к текущей строке. Для каждого раздела она вычисляется отдельно

СТРОКИ (ROWS) | ДИАПАЗОН (RANGE) | ГРУППЫ (GROUPS) между нижней и верхней границей



Границы рамки могут быть следующими:

- UNBOUNDED PRECEDING
- n PRECEDING
- CURRENT ROW
- n FOLLOWING
- UNBOUNDED FOLLOWING

Нижняя граница должна быть указана ПЕРЕД верхней границей

СТРОКИ (ROWS BETWEEN) МЕЖДУ 1 ПРЕДЫДУЩЕЙ И 1 СЛЕДУЮЩЕЙ

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

1 строка перед текущей и 1 строка после нее

ДИАПАЗОН (RANGE BETWEEN) МЕЖДУ 1 ПРЕДЫДУЩИМ И 1 СЛЕДУЮЩИМ ДИАПАЗОНОМ

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

значения в диапазоне от 3 до 5, сортировка с помощью ORDER BY

ГРУППЫ (GROUPS BETWEEN) МЕЖДУ 1 ПРЕДЫДУЩЕЙ И 1 СЛЕДУЮЩЕЙ

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

1 группа перед текущей строкой и 1 группа после нее, независимо от значения

На 2020 год ГРУППЫ поддерживаются только в PostgreSQL версии 11 и выше

## СОКРАЩЕНИЯ

Сокращение	Что означает
UNBOUNDED PRECEDING	ПРЕДЫДУЩИЕ СТРОКИ БЕЗ ОГРАНИЧЕНИЙ
n PRECEDING	СТРОКИ МЕЖДУ ТЕКУЩЕЙ И n ПРЕДЫДУЩИХ
CURRENT ROW	ТОЛЬКО ТЕКУЩАЯ СТРОКА
n FOLLOWING	СТРОКИ МЕЖДУ ТЕКУЩЕЙ И n СЛЕДУЮЩИХ
UNBOUNDED FOLLOWING	СЛЕДУЮЩИЕ СТРОКИ БЕЗ ОГРАНИЧЕНИЙ

## СТАНДАРТНАЯ РАМКА

Если применен оператор ORDER BY, то рамка находится в диапазоне МЕЖДУ ПРЕДЫДУЩЕЙ БЕЗ ОГРАНИЧЕНИЙ И ТЕКУЩЕЙ СТРОКОЙ.

Без оператора ORDER BY рамка находится между ПРЕДЫДУЩЕЙ И ПОСЛЕДУЮЩЕЙ СТРОКОЙ БЕЗ ОГРАНИЧЕНИЙ