

Базы данных и SQL

Урок 5
SQL – оконные функции





План курса

1

Инструменты для работы с базами данных

2

SQL – создание объектов, изменение данных, логические операторы

3

SQL – выборка данных, сортировка, агрегатные функции

4

SQL – объединение таблиц union, соединение - join, подзапросы

5

SQL – оконные функции

6

Будущий урок

7

Будущий урок

8

Будущий урок

9

Будущий урок

10

Будущий урок

11

Будущий урок

12






Будущий урок



Содержание урока



Что будет на уроке сегодня

-  Групповые функции без сортировки
-  Использование сортировки в оконной функции
-  Функции ранжирования, получения значений из соседних строк
-  Функции смещения и аналитические функции
-  Представления (view) – использование, операции с ними



Оконные функции SQL: таблица для работы

```
1 CREATE TABLE sales(  
2     sales_employee VARCHAR(50) NOT NULL,  
3     fiscal_year INT NOT NULL,  
4     sale DECIMAL(14,2) NOT NULL,  
5     PRIMARY KEY(sales_employee,fiscal_year)  
6 );  
7  
8 INSERT INTO sales(sales_employee,fiscal_year,sale)  
9 VALUES('Bob',2016,100),  
10        ('Bob',2017,150),  
11        ('Bob',2018,200),  
12        ('Alice',2016,150),  
13        ('Alice',2017,100),  
14        ('Alice',2018,200),  
15        ('John',2016,200),  
16        ('John',2017,150),  
17        ('John',2018,250);  
18  
19 SELECT * FROM sales;
```



Вспомним агрегатные функции:

```
1 SELECT SUM(sale)
2 FROM sales;
```

Общий объем продаж по финансовым годам:

```
1 SELECT fiscal_year, SUM(sale)
2 FROM sales
3 GROUP BY fiscal_year;
```



Пример агрегатной функции

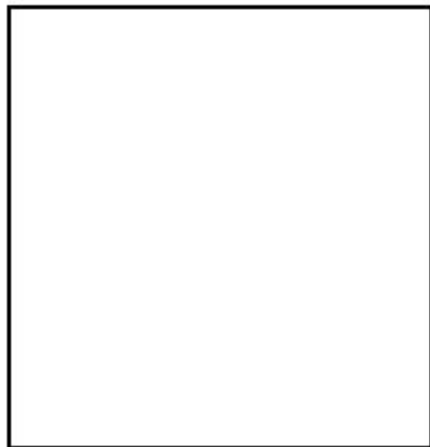
```
1 SELECT
2     fiscal_year,
3     SUM(sale)
4 FROM
5     sales
6 GROUP BY
7     fiscal_year;
```

	fiscal_year	sales_employee	sale	total_sales
▶	2016	Alice	150.00	450.00
	2016	Bob	100.00	450.00
	2016	John	200.00	450.00
	2017	Alice	100.00	400.00
	2017	Bob	150.00	400.00
	2017	John	150.00	400.00
	2018	Alice	200.00	650.00
	2018	Bob	200.00	650.00
	2018	John	250.00	650.00



Оконные функции SQL

Обычный запрос



Запрос с оконной функцией





Синтаксис оконной функции

```
1 SELECT
2 Название функции (столбец для вычислений)
3 OVER
4 (
5     PARTITION BY столбец для группировки
6     ORDER BY столбец для сортировки
7     ROWS или RANGE выражение для ограничения строк в пределах группы
8 )
```



Табличка для примеров:

Date	Medium	Conversions
10.05.2020	cpa	1
10.05.2020	cpc	2
10.05.2020	organic	1
11.05.2020	cpa	1
11.05.2020	cpc	3
11.05.2020	organic	2
11.05.2020	direct	1
12.05.2020	cpc	1
12.05.2020	organic	2



OVER(): просуммируем столбец «Conversions»:

```
1 SELECT
2   Date,
3   Medium,
4   Conversions,
5   SUM(Conversions) OVER() AS 'Sum'
6 FROM Orders;
```



OVER(): просуммируем столбец «Conversions»:

	Date	Medium	Conversions	Sum	
	10.05.2020	cpa	1	14	
	10.05.2020	cpc	2	14	
	10.05.2020	organic	1	14	
	11.05.2020	cpa	1	14	
	11.05.2020	cpc	3	14	
	11.05.2020	organic	2	14	
	11.05.2020	direct	1	14	
	12.05.2020	cpc	1	14	
	12.05.2020	organic	2	14	



PARTITION BY

```
1 SELECT
2   Date,
3   Medium,
4   Conversions,
5   SUM(Conversions) OVER(PARTITION BY Date) AS 'Sum'
6 FROM Order;
```



PARTITION BY

	Date	Medium	Conversions	Sum	
	10.05.2020	cpa	1	4	
	10.05.2020	cpc	2	4	
	10.05.2020	organic	1	4	
	11.05.2020	cpa	1	7	
	11.05.2020	cpc	3	7	
	11.05.2020	organic	2	7	
	11.05.2020	direct	1	7	
	12.05.2020	cpc	1	3	
	12.05.2020	organic	2	3	



ORDER BY

```
1 SELECT
2   Date,
3   Medium,
4   Conversions,
5   SUM(Conversions) OVER(PARTITION BY Date ORDER BY Medium) AS 'Sum'
6 FROM Orders;
```



PARTITION BY

	Date	Medium	Conversions	Sum	
	10.05.2020	cpa	1	1	
	10.05.2020	cpc	2	3	
	10.05.2020	organic	1	4	
	11.05.2020	cpa	1	1	
	11.05.2020	cpc	3	4	
	11.05.2020	direct	1	5	
	11.05.2020	organic	2	7	
	12.05.2020	cpc	1	1	
	12.05.2020	organic	2	3	



ROWS и RANGE

UNBOUNDED PRECEDING — указывает, что окно начинается с первой строки группы;

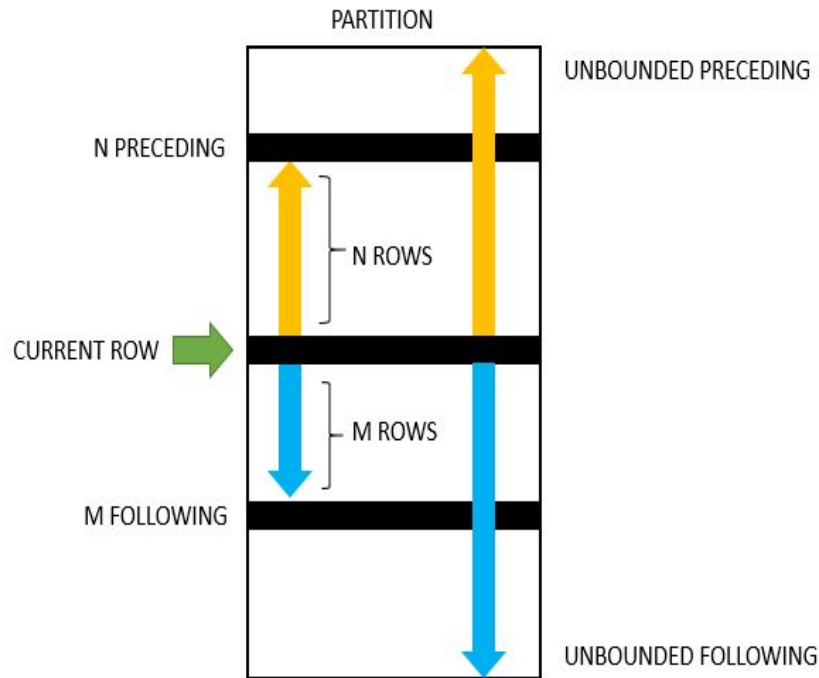
UNBOUNDED FOLLOWING — с помощью данной инструкции можно указать, что окно заканчивается н последней строке группы;

CURRENT ROW — инструкция указывает, что окно начинается или заканчивается на текущей строке;

BETWEEN «граница окна» AND «граница окна» — указывает нижнюю и верхнюю границу окна;

«Значение» **PRECEDING** — определяет число строк перед текущей строкой (не допускается в предложении RANGE).;

«Значение» **FOLLOWING** — определяет число строк после текущей строки (не допускается в предложении RANGE).





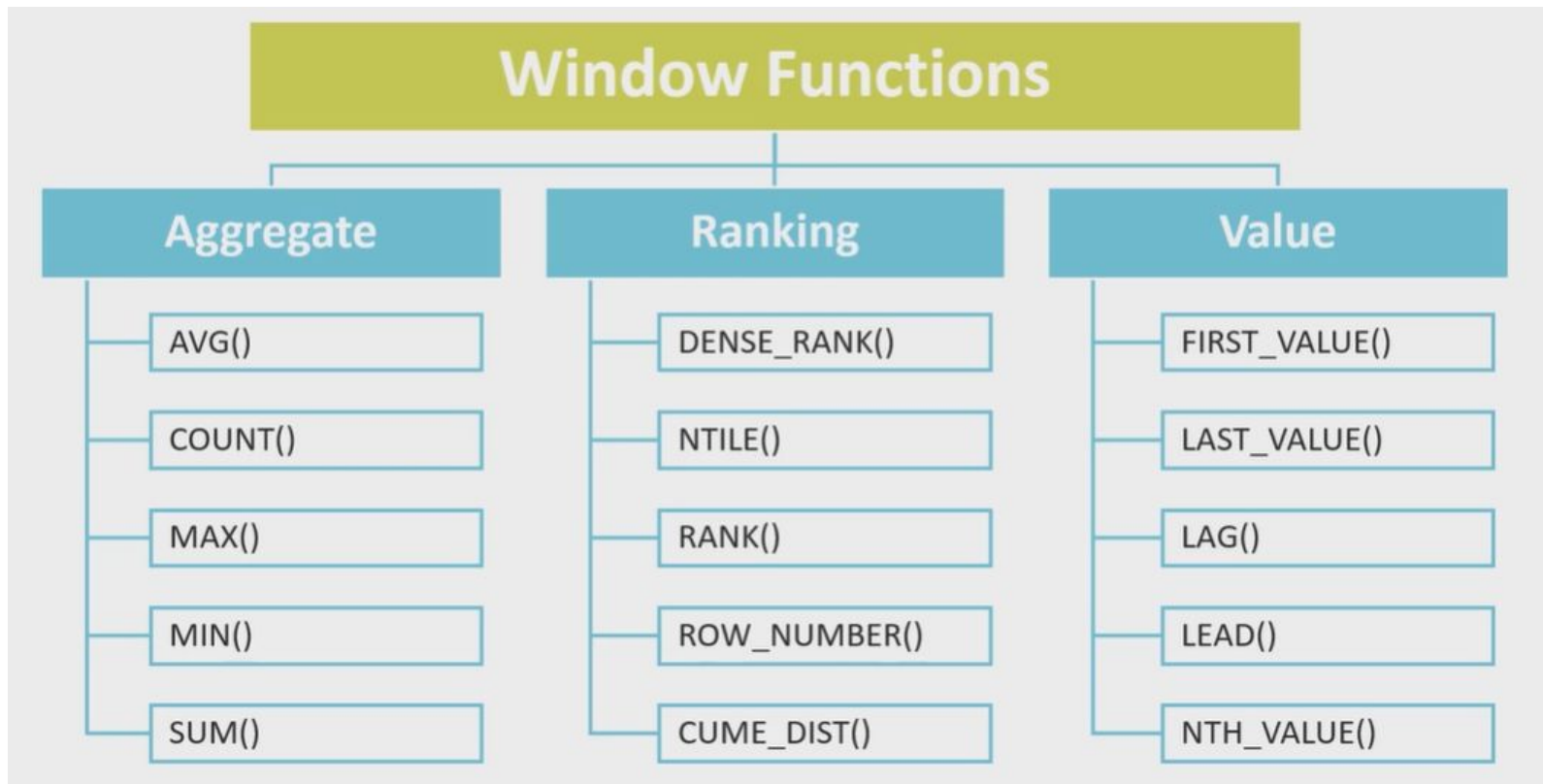
ROWS и RANGE: пример

```
1 SELECT
2   Date,
3   Medium,
4   Conversions,
5   SUM(Conversions) OVER(PARTITION BY Date ORDER BY Conversions ROWS BETWEEN
   CURRENT ROW AND 1 FOLLOWING) AS 'Sum'
6 FROM Orders;
```



ROWS и RANGE: пример

	Date	Medium	Conversions	Sum	
	10.05.2020	cpa	1	2	= 1+1
	10.05.2020	organic	1	3	= 1+2
	10.05.2020	cpc	2	2	= 2
	11.05.2020	cpa	1	2	= 1+1
	11.05.2020	direct	1	3	= 1+2
	11.05.2020	organic	2	5	= 2+3
	11.05.2020	cpc	3	3	= 3
	12.05.2020	cpc	1	3	= 1+2
	12.05.2020	organic	2	2	= 2





Агрегатные функции

SUM — возвращает сумму значений в столбце;

COUNT — вычисляет количество значений в столбце (значения NULL не учитываются);

AVG — определяет среднее значение в столбце;

MAX — определяет максимальное значение в столбце;

MIN — определяет минимальное значение в столбце.



Пример использования агрегатных функций с оконной конструкцией OVER:

```
1 SELECT
2   Date,
3   Medium,
4   Conversions,
5   SUM(Conversions) OVER(PARTITION BY Date) AS 'Sum' ,
6   COUNT(Conversions) OVER(PARTITION BY Date) AS 'Count' ,
7   AVG(Conversions) OVER(PARTITION BY Date) AS 'Avg' ,
8   MAX(Conversions) OVER(PARTITION BY Date) AS 'Max' ,
9   MIN(Conversions) OVER(PARTITION BY Date) AS 'Min'
10 FROM Orders
```



Пример использования агрегатных функций с оконной конструкцией OVER:

	Date	Medium	Conversions	Sum	Count	Avg	Max	
	10.05.2020	cpa	1	4	3	1,3	2	
	10.05.2020	cpc	2	4	3	1,3	2	
	10.05.2020	organic	1	4	3	1,3	2	
	11.05.2020	cpa	1	7	4	1,75	3	
	11.05.2020	cpc	3	7	4	1,75	3	
	11.05.2020	organic	2	7	4	1,75	3	
	11.05.2020	direct	1	7	4	1,75	3	
	12.05.2020	cpc	1	3	2	1,5	2	
	12.05.2020	organic	2	3	2	1,5	2	



Ранжирующие функции

ROW_NUMBER — функция возвращает номер строки и используется для нумерации;

RANK — функция возвращает ранг каждой строки. В данном случае значения уже анализируются и, в случае нахождения одинаковых, возвращает одинаковый ранг с пропуском следующего значения;

DENSE_RANK — функция возвращает ранг каждой строки. Но в отличие от функции RANK, она для одинаковых значений возвращает ранг, не пропуская следующий;

NTILE — это функция, которая позволяет определить к какой группе относится текущая строка. Количество групп задается в скобках.



Ранжирующие функции

```
1 SELECT
2   Date,
3   Medium,
4   Conversions,
5   ROW_NUMBER()
6   OVER(PARTITION BY Date ORDER BY Conversions) AS 'Row_number',
7   RANK()
8   OVER(PARTITION BY Date ORDER BY Conversions) AS 'Rank',
9   DENSE_RANK()
10  OVER(PARTITION BY Date ORDER BY Conversions) AS 'Dense_Rank',
11  NTILE(3)
12  OVER(PARTITION BY Date ORDER BY Conversions) AS 'Ntile'
13 FROM Orders
```



Ранжирующие функции

	Date	Medium	Conversions	Row_number	Rank	Dense_Rank	Ntile	
	10.05.2020	cpa	1	1	1	1	1	
	10.05.2020	organic	1	2	1	1	2	
	10.05.2020	cpc	2	3	3	2	3	
	11.05.2020	cpa	1	1	1	1	1	
	11.05.2020	direct	1	2	1	1	1	
	11.05.2020	organic	2	3	3	2	2	
	11.05.2020	cpc	3	4	4	3	3	
	12.05.2020	cpc	1	1	1	1	1	
	12.05.2020	organic	2	2	2	2	2	



Функции смещения

LAG или **LEAD** – функция LAG обращается к данным из предыдущей строки окна, а LEAD к данным из следующей строки. Функцию можно использовать для того, чтобы сравнивать текущее значение строки с предыдущим или следующим. Имеет три параметра: столбец, значение которого необходимо вернуть, количество строк для смещения (по умолчанию 1), значение, которое необходимо вернуть если после смещения возвращается значение NULL;

FIRST_VALUE или **LAST_VALUE** — с помощью функции можно получить первое и последнее значение в окне. В качестве параметра принимает столбец, значение которого необходимо вернуть.



Функции смещения

```
1 SELECT
2   Date,
3   Medium,
4   Conversions,
5   LAG(Conversions) OVER(PARTITION BY Date ORDER BY Date) AS 'Lag' ,
6   LEAD(Conversions) OVER(PARTITION BY Date ORDER BY Date) AS 'Lead' ,
7   FIRST_VALUE(Conversions) OVER(PARTITION BY Date ORDER BY Date) AS 'First_Value',
8   LAST_VALUE(Conversions) OVER(PARTITION BY Date ORDER BY Date) AS 'Last_Value'
9 FROM Orders;
```



Функции смещения

	Date	Medium	Conversions	Lag	Lead	First_Value	Last_Value	
	10.05.2020	сра	1	NULL	2	1	1	
	10.05.2020	срс	2	1	1	1	1	
	10.05.2020	organic	1	2	NULL	1	1	
	11.05.2020	сра	1	NULL	3	1	1	
	11.05.2020	срс	3	1	2	1	1	
	11.05.2020	organic	2	3	1	1	1	
	11.05.2020	direct	1	2	NULL	1	1	
	12.05.2020	срс	1	NULL	2	1	2	
	12.05.2020	organic	2	1	NULL	1	2	



В чем заключается главное отличие оконных функций от функций агрегации с группировкой?

Применение функции агрегации и команды GROUP BY

Имя	Предмет	Оценка
Петя	матем	3
Петя	рус	4
Петя	физ	5
Петя	история	4
Маша	матем	4
Маша	рус	3
Маша	физ	5
Маша	история	3



Имя	Средняя оценка
Петя	4
Маша	3,75

В чем заключается главное отличие оконных функций от функций агрегации с группировкой?

Применение Оконной функции

Имя	Предмет	Оценка
Петя	матем	3
Петя	рус	4
Петя	физ	5
Петя	история	4
Маша	матем	4
Маша	рус	3
Маша	физ	5
Маша	история	3



Имя	Предмет	Оценка	Средняя оценка
Петя	матем	3	4
Петя	рус	4	4
Петя	физ	5	4
Петя	история	4	4
Маша	матем	4	3,75
Маша	рус	3	3,75
Маша	физ	5	3,75
Маша	история	3	3,75

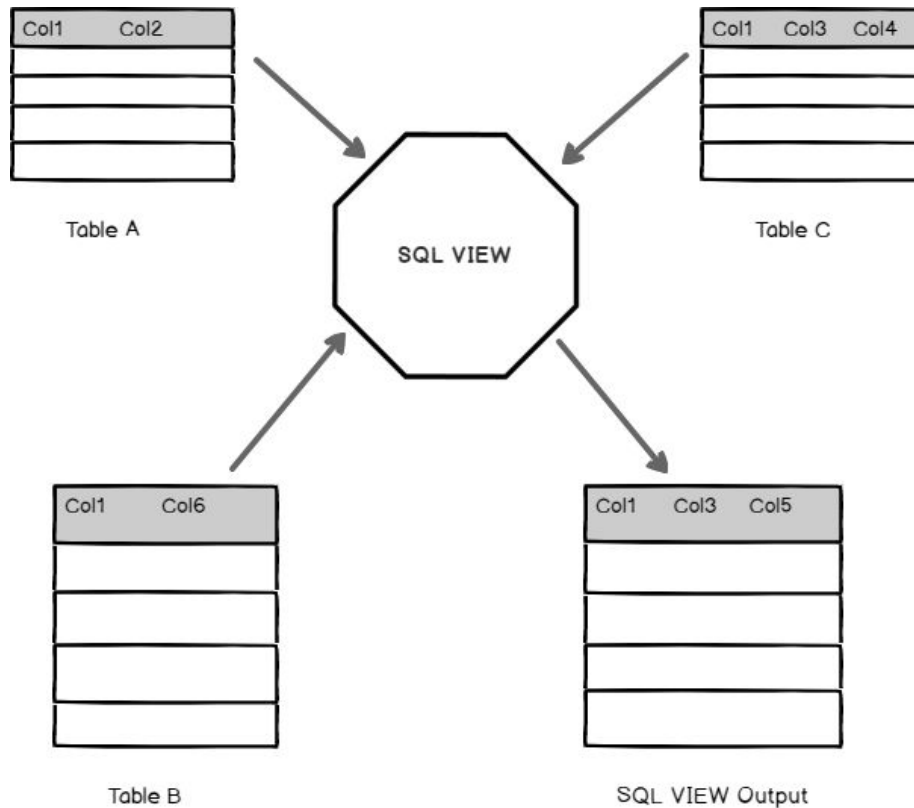


Порядок расчета оконных функций в SQL запросе

SELECT	list of columns, window functions
FROM	table / joint tables / subquery
WHERE	filtering clause
GROUP BY	list of columns
HAVING	aggregation filtering clause
ORDER BY	list of columns / window functions



Представления





Представления

```
1 CREATE [OR REPLACE] VIEW view_name AS
2
3 SELECT columns
4
5 FROM tables
6
7 [WHERE conditions];
```

OR REPLACE - необязательный. Если вы не укажете этот атрибут и VIEW уже существует, оператор CREATE VIEW вернет ошибку.

view_name - имя VIEW, которое вы хотите создать в MySQL.

WHERE conditions - необязательный. Условия, которые должны быть выполнены для записей, которые должны быть включены в VIEW.



Представления

```
1 CREATE VIEW Londonstaff
2     AS SELECT *
3     FROM Salespeople
4     WHERE city = 'London';
```

```
1 SELECT *
2 FROM Londonstaff;
3
```



Представления

```
1 SELECT *  
2 FROM Londonstaff;  
3
```

SQL Execution Log			
SELECT * FROM Londonstaff;			
snum	sname	city	comm
1001	Peel	London	0.1200
1004	Motika	London	0.1100



Представления

Customer_id	Customer_name	Contact_no	Email	Purchased_amount	City
184	Ravi Kumar	9887463893	ravi@gmail.com	8000.00	Kolkata
987	Vinay Das	9839878678	vinay@yahoo.in	12000.00	Delhi
452	K.Amarnath	7598759387	amar@gmail.com	15000.00	Kolkata
874	Abhinash Desai	7675878798	desai@gmail.com	5000.00	Mumbai



```
1 CREATE VIEW customer_archive AS
2 SELECT customer_id, customer_name, contact_no, purchased_amount, city
3 FROM customer
4 WHERE purchased_amount > 10000;
```

Customer_id	Customer_name	Contact_no	Purchased_amount	City
987	Vinay Das	9839878678	12000.00	Delhi
452	K.Amarnath	7598759387	15000.00	Kolkata



Операции с представлениями: удаление и объединение

DROP: представление/виртуальную таблицу можно удалить с помощью команды DROP VIEW. Если мы хотим удалить таблицу customer_archive.

```
1 DROP VIEW customer_archive;
```

Объединение:

```
1 CREATE VIEW view-name AS
2 SELECT column1, column2, column3, ...
3 FROM table_name1 INNER JOIN table_name2
4 ON table_name1.column = table_name2.column;
```



Операции с представлениями: изменение

```
1 ALTER VIEW view_name AS
2   SELECT columns
3   FROM table
4   WHERE conditions;
```

```
1 ALTER VIEW hardware_suppliers AS
2   SELECT supplier_id, supplier_name, address, city
3   FROM suppliers
4   WHERE category_type = 'Hardware';
```




Задача:
создайте представление, которое показывало бы всех заказчиков, имеющих самые высокие рейтинги. Привязке к таблице нет, колонки таблицы воображаемые








Ответ на задачу:

```
1 CREATE VIEW Highratings
2     AS SELECT *
3     FROM Customers
4     WHERE rating =
5         (SELECT MAX (rating)
6          FROM Customers);
```



Итоги занятия:

-  Вспомнили групповые функции без сортировки
-  Научились пользоваться сортировкой в оконной функции
-  Узнали про функции ранжирования, получения значений из соседних строк
-  Функции смещения и аналитические функции
-  Представления (view) – использование, операции с ними



Спасибо за внимание!