Базы данных и SQL

Семинар 5.

Вопросы?

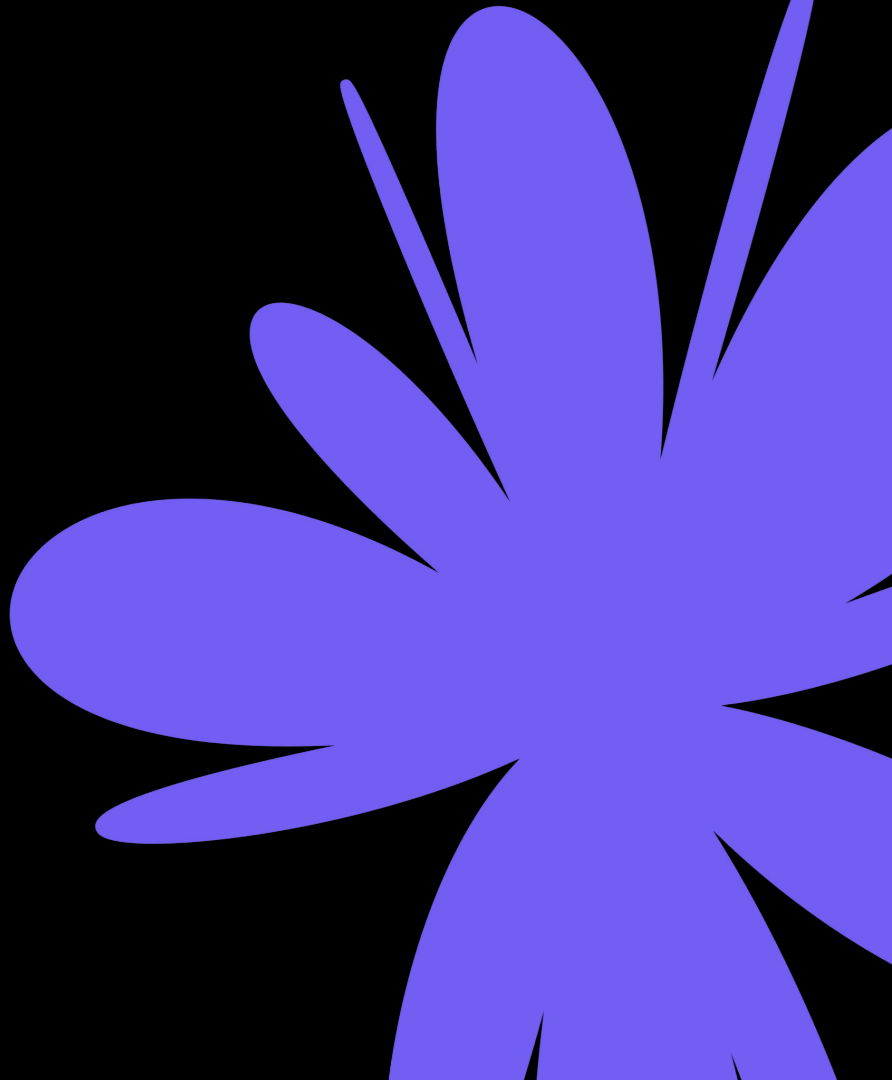
Вопросы?

Вопросы?

План на сегодня:

- ➔ Quiz!
- ➔ Рекурсивные CTE
- ➔ Перерыв
- ➔ Оконные функции
- ➔ Домашнее задание

Quiz!



Для создания новой виртуальной таблицы, которая базируется на результатах сделанного ранее SQL запроса, используется команда:

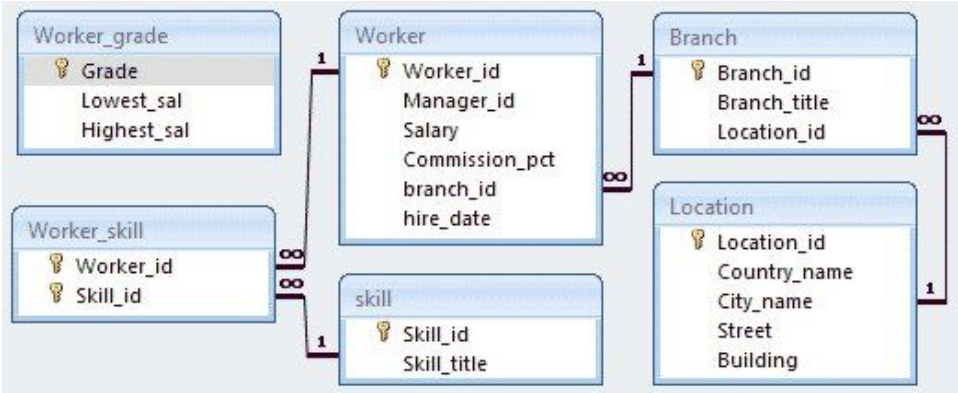
1. CREATE VIRTUAL TABLE
2. CREATE VIEW
3. ALTER VIEW

Для создания новой виртуальной таблицы, которая базируется на результатах сделанного ранее SQL запроса, используется команда:

1. CREATE VIRTUAL TABLE
2. CREATE VIEW
3. ALTER VIEW

Для создания представления, в которое должны попасть только имена сотрудников, работающих в отделе Research, используется запрос:

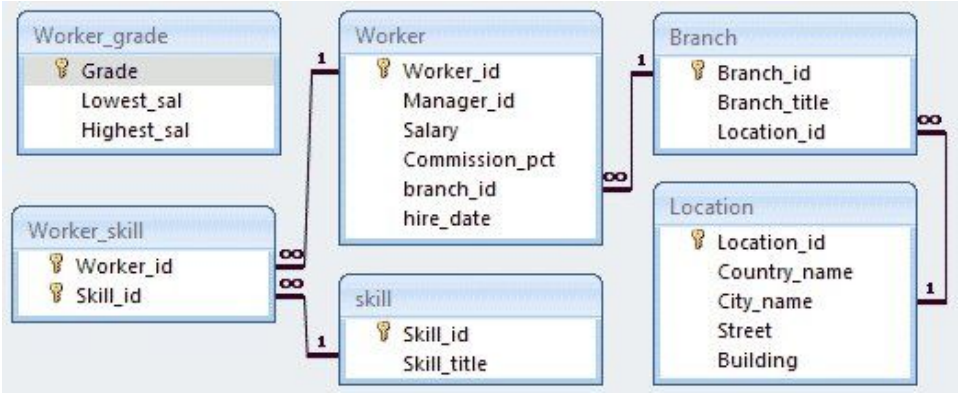
CREATE _____
SELECT Worker_name FROM Worker w, Branch b
WHERE w.Branch_id = b.Branch_id AND Branch_title LIKE ‘Research’



1. VIEW AS
2. view1 AS
3. VIEW view1
4. VIEW view1 AS SUBQUERY
5. VIEW view1 AS

Для создания представления, в которое должны попасть только имена сотрудников, работающих в отделе Research, используется запрос:

CREATE _____
SELECT Worker_name FROM Worker w, Branch b
WHERE w.Branch_id = b.Branch_id AND Branch_title LIKE ‘Research’

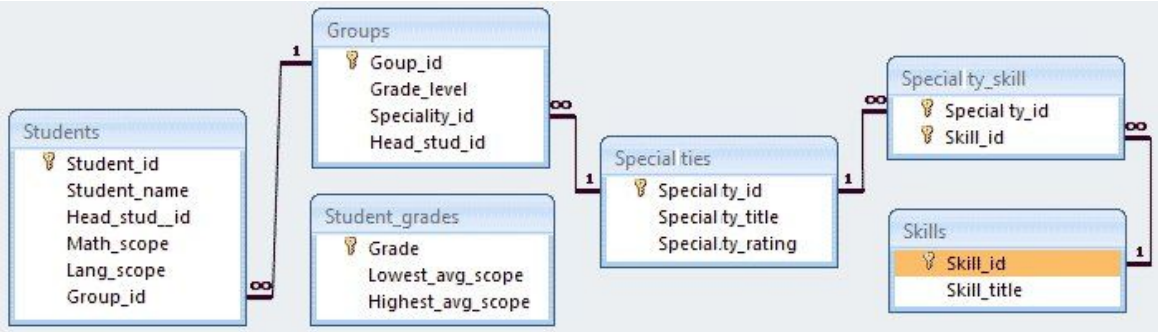


1. VIEW AS
2. view1 AS
3. VIEW view1
4. VIEW view1 AS SUBQUERY
5. VIEW view1 AS

Для создания представления, в которое должны попасть только имена студентов второго курса, используется запрос:

CREATE VIEW view 1

AS.....

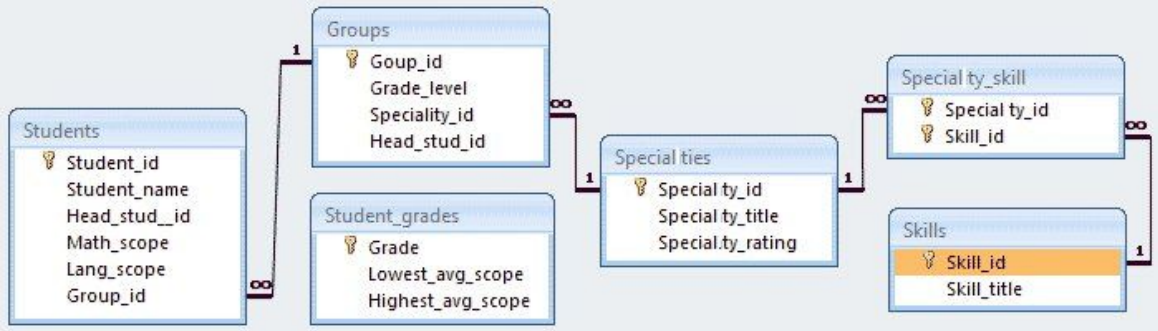


1. (SELECT Student_name FROM Students JOIN Groups ON Students.Group_id = Groups.Group_id) WITH CHECK OPTION Grade_level = 2
2. SELECT Student_name FROM Students, Groups WHERE Students.Group_id = Groups.Group_id AND Grade_level = 2
3. (SELECT Student_name FROM Students JOIN Groups ON Students.Group_id = Groups.Group_id AND Grade_level = 2)
4. WITH CHECK OPTION Grade_level=2 (SELECT Student_name FROM Students JOIN Groups ON Students.Group_id = Groups.Group_id)

Для создания представления, в которое должны попасть только имена студентов второго курса, используется запрос:

CREATE VIEW view 1

AS.....



1. (SELECT Student_name FROM Students JOIN Groups ON Students.Group_id = Groups.Group_id) WITH CHECK OPTION Grade_level = 2
2. SELECT Student_name FROM Students, Groups WHERE Students.Group_id = Groups.Group_id AND Grade_level = 2
3. (SELECT Student_name FROM Students JOIN Groups ON Students.Group_id = Groups.Group_id AND Grade_level = 2)
4. WITH CHECK OPTION Grade_level=2 (SELECT Student_name FROM Students JOIN Groups ON Students.Group_id = Groups.Group_id)

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

1. При использовании агрегирующих функций предложение GROUP BY сокращает количество строк в запросе с помощью их группировки, а при использовании оконных функций количество строк в запросе не уменьшается по сравнении с исходной таблицей.
2. Никакого различия нет
3. При использовании агрегирующих функций предложение GROUP BY НЕ сокращает количество строк в запросе с помощью их группировки, а при использовании оконных функций количество строк в запросе не уменьшается по сравнении с исходной таблицей.

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

1. При использовании агрегирующих функций предложение GROUP BY сокращает количество строк в запросе с помощью их группировки, а при использовании оконных функций количество строк в запросе не уменьшается по сравнении с исходной таблицей.
2. Никакого различия нет
3. При использовании агрегирующих функций предложение GROUP BY НЕ сокращает количество строк в запросе с помощью их группировки, а при использовании оконных функций количество строк в запросе не уменьшается по сравнении с исходной таблицей.

Оконные функции делятся на:

1. Агрегатные функции
2. Ранжирующие функции
3. Встроенные
4. Функции смещения
5. Аналитические функции

Оконные функции делятся на:

1. Агрегатные функции
2. Ранжирующие функции
3. Встроенные
4. Функции смещения
5. Аналитические функции

15мин

CTE (Common Table Expressions)

Что это такое?

До версии 8.0:

Производные таблицы (Derived Tables)

SELECT ... FROM (subquery) AS derived, t1, ...

Начиная с 8.0, также доступны:

Обобщенные табличные выражения (Common Table Expressions)

WITH cte AS (subquery) SELECT ... FROM cte, t1 ...

15мин

Производные таблицы:

```
SELECT dt.a
        FROM t1 LEFT JOIN
                ((SELECT ... FROM ...) AS dt JOIN t2 ON ...) ON ...
```

- ... сначала видим dt.a
- ... что такое dt ?
- ... приходится искать вглубь

Обобщенные табличные выражения:

```
WITH dt AS (SELECT ... FROM ...)
SELECT dt.a
FROM t1 LEFT JOIN (dt JOIN t2 ON ... ) ON ...
```

Табличные выражения по сравнению с производными таблицами

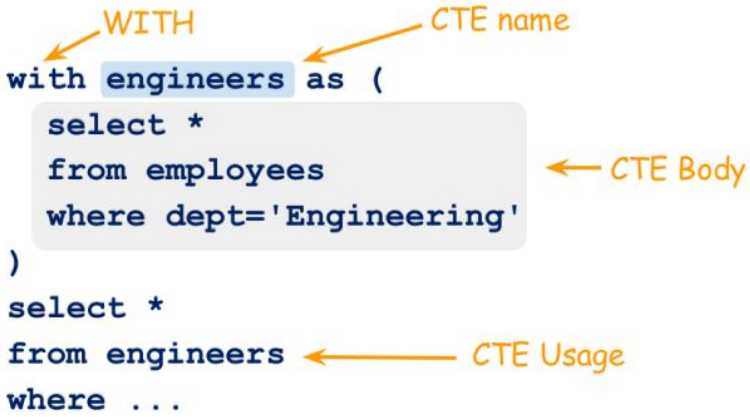
15мин

- Проще читаются
- Проще выстраивать в цепочки
- Можно ссылаться много раз

CTE

15мин

```
WITH <cte_name> (<colums>) AS
(
  <cte_query>
)
  <main_query>
```



15мин

Рекурсивные CTE

CTE является рекурсивным, если его подзапрос ссылается на его собственное имя. Если планируется использовать рекурсивный CTE то в запрос должен быть включен параметр RECURSIVE.

```
WITH RECURSIVE <cte_name> (<colums>) AS
(
 <base_case_query>
 UNION ALL
 <recursive_step_query> -- invoke the CTE
here!
)
<main_query>
```

```
WITH RECURSIVE sequence (n) AS
(
  SELECT 0
  UNION ALL
  SELECT n + 1
  FROM sequence
  WHERE n + 1 <= 10
)
SELECT n
FROM sequence;
```

Рекурсивные CTE

15мин

```
WITH RECURSIVE cte AS
    (SELECT ... FROM table_name1 WHERE ... # начальный подзапрос
    UNION
    SELECT ... FROM cte, table_name2 WHERE ... ) # рекурсивный подзапрос
SELECT ... FROM cte; # внешний запрос
```

Результат вычисляется путем итераций

– **Шаг 0:** результат дает начальный подзапрос

– **Шаг N+1**: выполняется начальная и рекурсивная части, в качестве значения cte при выполнении рекурсивной части используется результат выполнения шага N

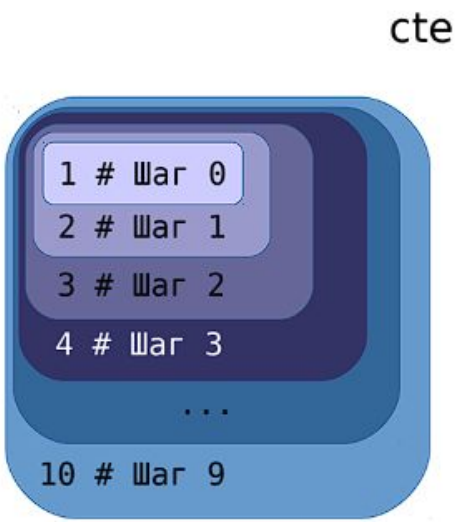
– Останавливаем вычисление, когда результат очередного шага совпадает с результатом прошлого шага

– К результату итераций применяется внешний запрос

Пример: генерация набора от 1 до 10

15мин

```
WITH RECURSIVE cte AS
(
        SELECT 1 AS a
        UNION ALL
        SELECT a + 1 FROM cte
        WHERE a < 10
)
SELECT * FROM cte;
```



15мин

Задача

Ссылка на материалы для работы:
https://drive.google.com/file/d/1J5UCDn8hksQmNFogOIqfgZ8eRKy2PA6q/view?usp=sharing

1. Используя CTE, выведите всех пользователей из таблицы users_profile

2. Используя CTE, подсчитайте количество активных пользователей . Задайте псевдоним результирующему окну. Пример:

Total Active Users
2

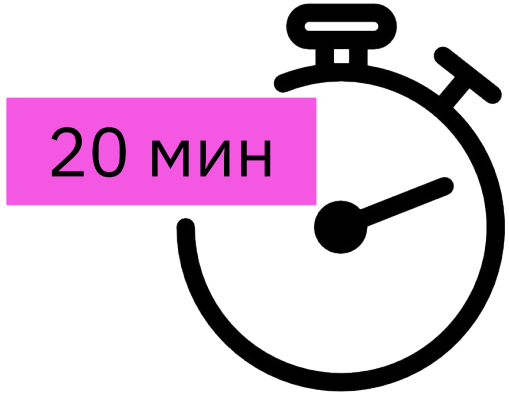
3. С помощью CTE реализуйте таблицу квадр… 1 до 10:
(пример для чисел от 1 до 3)

a	result
1	1
2	4
3	9

Ваши вопросы?

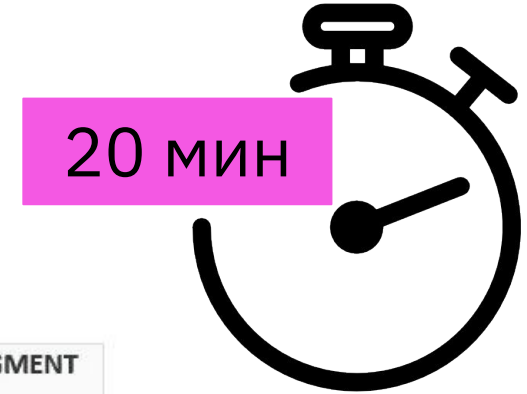
Перерыв

Оконные функции



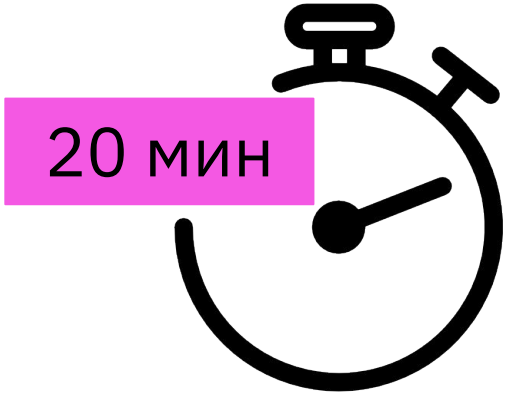
SELECT
Название функции (столбец для вычислений)
OVER (
PARTITION BY столбец для группировки
ORDER BY столбец для сортировки
ROWS или RANGE выражение для ограничения строк в пределах группы
)

Таблица для работы



TB	ID_CLIENT	ID_DOG	OSZ (Задолженность)	PROCENT_RATE	RATING	SEGMENT
A	1	111	100	6	10	SREDN
A	1	222	150	6	10	SREDN
A	2	333	50	9	15	MMB
B	1	444	200	7	10	SREDN
B	3	555	1000	5	16	CIB
B	4	666	500	10	20	CIB
B	4	777	10	12	17	MMB
C	5	888	20	11	21	MMB
C	5	999	200	9	13	SREDN

Задача:



Собрать дэшборд, в котором содержится информация о максимальной задолженности в каждом банке, а также средний размер процентной ставки в каждом банке в зависимости от сегмента и количество договоров всего всем банкам

Задача:

20 мин

TB	ID_CLIENT	ID_DOG	OSZ	PROCENT_RATE	RATING	SEGMENT	Максимальная задолженность в разбивке по банкам	Средняя процентная ставка в разрезе банка и сегмента	Всего договоров во всех банках
A	2	333	50	9	15	MMB	150	9	9
A	1	111	100	6	10	SREDN	150	6	9
A	1	222	150	6	10	SREDN	150	6	9
B	3	555	1000	5	16	CIB	1000	7.5	9
B	4	666	500	10	20	CIB	1000	7.5	9
B	4	777	10	12	17	MMB	1000	12	9
B	1	444	200	7	10	SREDN	1000	7	9
C	5	888	20	11	21	MMB	200	11	9
C	5	999	200	9	13	SREDN	200	9	9

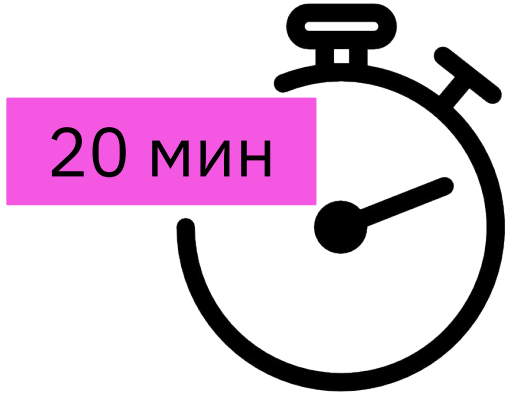
Проранжируем таблицу по убыванию количества ревизий:

20 мин

tb	dep	count_revisions	row_number	rank	dense_rank	ntile
D	Rozn	120	1	1	1	1
F	Rozn	111	2	2	2	1
A	Corp	100	3	3	3	1
D	Corp	95	4	4	4	1
A	IT	95	5	4	4	1
D	IT	85	6	6	5	1
E	IT	80	7	7	6	2
E	Rozn	72	8	8	7	2
E	Corp	70	9	9	8	2
B	Corp	70	10	9	8	2
F	Corp	66	11	11	9	2
B	Rozn	65	12	12	10	2
C	IT	63	13	13	11	3
B	IT	58	14	14	12	3
A	Rozn	47	15	15	13	3
C	Corp	42	16	16	14	3
C	Rozn	40	17	17	15	3
F	IT	33	18	18	16	3

TB	DEP	Count_Revisions
A	Corp	100
A	Rozn	47
A	IT	86
B	Corp	70
B	Rozn	65
B	IT	58
C	Corp	42
C	Rozn	40
C	IT	63
D	Corp	95
D	Rozn	120
D	IT	85
E	Corp	70
E	Rozn	72
E	IT	80
F	Corp	66
F	Rozn	111
F	IT	33

Задача:

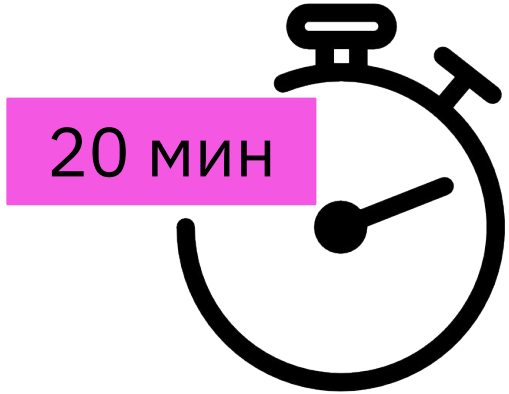


Найти второй отдел во всех банках по количеству ревизий.

```
SELECT  MAX(count_revisions) ms
FROM Table_Rev
WHERE count_revisions!=(SELECT MAX(count_revisions)
 FROM Table_Rev)
```

Но если речь идет не про второй отдел, а про третий?
Уже сложнее. Именно поэтому, попробуйте воспользоваться оконной функцией

Задача




```
With T_R as
(
SELECT * , DENSE_RANK() OVER(PARTITION BY tb
ORDER BY count_revisions) ds
FROM Table_Rev
)
SELECT tb,dep,count_revisions
FROM T_R
WHERE ds=1
```

TB	DEP	Count_Revisions
A	Rozn	47
B	IT	58
C	Rozn	40
D	IT	85
E	Corp	70
F	IT	33

Оконные функции смещения

LAG — смещение назад.

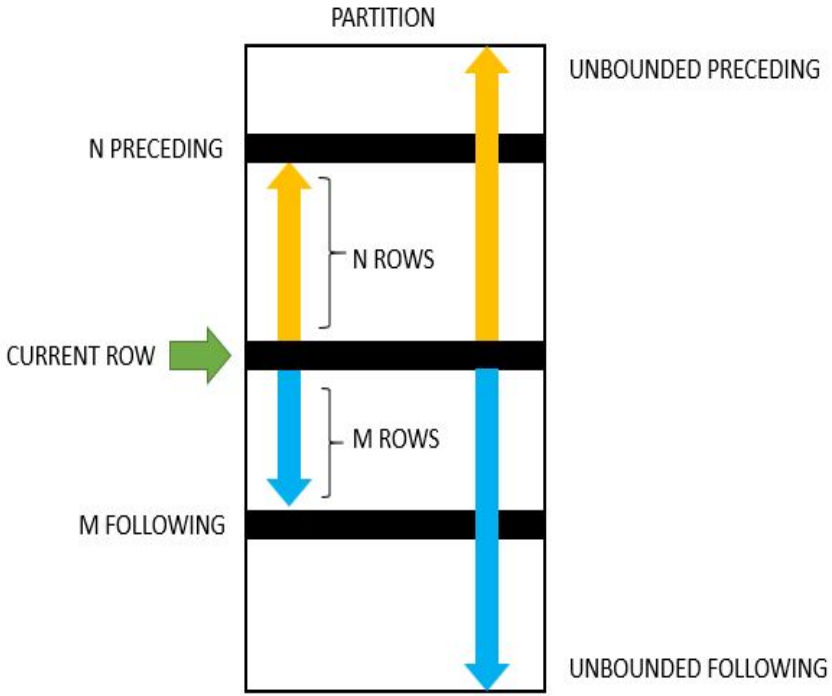
LEAD — смещение вперед.

FIRST_VALUE — найти первое значение набора данных.

LAST_VALUE — найти последнее значение набора данных.

LAG и **LEAD** имеют следующие аргументы:

- Столбец, значение которого необходимо вернуть
- На сколько строк выполнить смешение (дефолт =1)
- Что вставить, если вернулся NULL

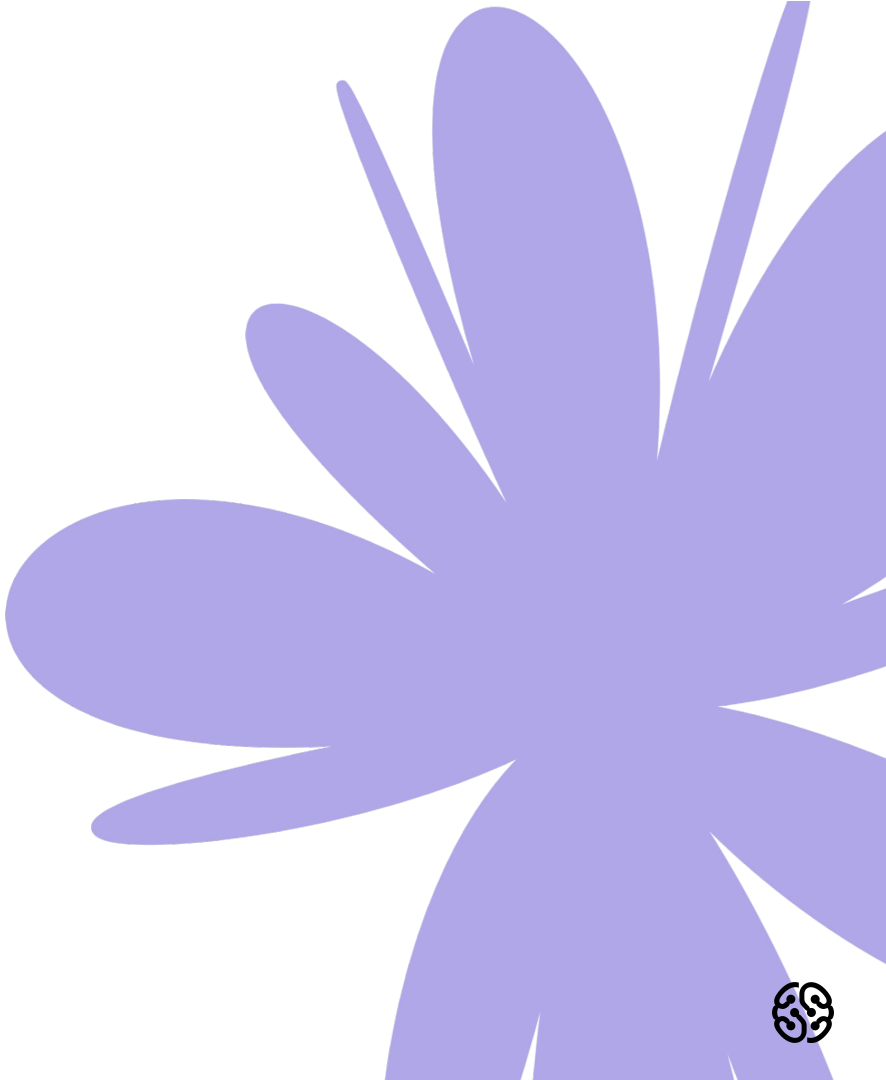


Оконные функции смещения

id_task	event	date_event
1	Open	2020-02-01
1	To_1_Line	2020-02-02
1	To_2_Line	2020-02-03
1	Successful	2020-02-04
1	Close	2020-02-05
2	Open	2020-03-01
2	To_1_Line	2020-03-02
2	Denied	2020-03-03
3	Open	2020-04-01
3	To_1_Line	2020-04-02
3	To_2_Line	2020-04-03

id_task	event	date_event	next_event	next_date
1	Open	2020-02-01	To_1_Line	2020-02-02
1	To_1_Line	2020-02-02	To_2_Line	2020-02-03
1	To_2_Line	2020-02-03	Successful	2020-02-04
1	Successful	2020-02-04	Close	2020-02-05
1	Close	2020-02-05	end	2099-01-01
2	Open	2020-03-01	To_1_Line	2020-03-02
2	To_1_Line	2020-03-02	Denied	2020-03-03
2	Denied	2020-03-03	end	2099-01-01
3	Open	2020-04-01	To_1_Line	2020-04-02
3	To_1_Line	2020-04-02	To_2_Line	2020-04-03
3	To_2_Line	2020-04-03	end	2099-01-01

Ваши вопросы?



Домашнее задание

1. Создайте представление, в которое попадут автомобили стоимостью до 25 000 долларов

2. Изменить в существующем представлении порог для стоимости: пусть цена будет до 30 000 долларов (используя оператор ALTER VIEW)

3. Создайте представление, в котором будут только автомобили марки “Шкода” и “Ауди”

```
mysql> SELECT * FROM Cars;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
```

Домашнее задание

Вывести название и цену для всех анализов, которые продавались 5 февраля 2020 и всю следующую неделю.

Есть таблица анализов Analysis:
an_id — ID анализа;
an_name — название анализа;
an_cost — себестоимость анализа;
an_price — розничная цена анализа;
an_group — группа анализов.
Есть таблица групп анализов Groups:
gr_id — ID группы;
gr_name — название группы;
gr_temp — температурный режим хранения.
Есть таблица заказов Orders:
ord_id — ID заказа;
ord_datetime — дата и время заказа;
ord_an — ID анализа.

Домашнее задание

Добавьте новый столбец под названием «время до следующей станции». Чтобы получить это значение, мы вычитаем время станций для пар смежных станций. Мы можем вычислить это значение без использования оконной функции SQL, но это может быть очень сложно. Проще это сделать с помощью оконной функции LEAD . Эта функция сравнивает значения из одной строки со следующей строкой, чтобы получить результат. В этом случае функция сравнивает значения в столбце «время» для станции со станцией сразу после нее.

train_id integer	station character varying(20)	station_time time without time zone	time_to_next_station interval
110	San Francisco	10:00:00	00:54:00
110	Redwood City	10:54:00	00:08:00
110	Palo Alto	11:02:00	01:33:00
110	San Jose	12:35:00	
120	San Francisco	11:00:00	01:49:00
120	Palo Alto	12:49:00	00:41:00
120	San Jose	13:30:00	

Рефлексия

Был урок полезен вам?

Узнали вы что-то новое?

Что было сложно?

Спасибо за внимание