SQL: оконные функции, транзакции

Running total mileage visual		
Day	**Miles Driving**	**Running Total**
Jan. 1	60	
Jan. 2	80	
Jan. 3	10	
Jan. 4	150	

Оконные функции

```
SELECT
        Day,
        MilesDriving,
        SUM(MilesDriving) OVER(ORDER BY Day) AS RunningTotal
FROM table;
```

Вот пример, показывающий, как сравнить зарплату каждого сотрудника со средней зарплатой его отдела:

```
SELECT depname, empno, salary, avg(salary) OVER (PARTITION BY depname)
  FROM empsalary;
```

```
  depname   | empno | salary |          avg
------------+-------+--------+-----------------------
 develop    |    11 |   5200 | 5020.0000000000000000
 develop    |     7 |   4200 | 5020.0000000000000000
 develop    |     9 |   4500 | 5020.0000000000000000
 develop    |     8 |   6000 | 5020.0000000000000000
 develop    |    10 |   5200 | 5020.0000000000000000
 personnel  |     5 |   3500 | 3700.0000000000000000
 personnel  |     2 |   3900 | 3700.0000000000000000
 sales      |     3 |   4800 | 4866.6666666666666667
 sales      |     1 |   5000 | 4866.6666666666666667
 sales      |     4 |   4800 | 4866.6666666666666667
(10 rows)
```

Оконные функции

Running Average Example		
Day	**Daily Revenue**	**3 Day Average**
1	39	
2	528	
3	39	
4	86	
5	86	
6	351	

Оконные функции

```
SELECT
        Day
        DailyRevenue,
AVG(Daily Revenue) OVER(ORDER BY Day ROWS 2 PRECEDING) AS 3DayAverage
FROM table;
```

Оконные функции

- Хотим посчитать среднее время между заказами для каждого пользователя
- Как это сделать?

Оконные функции

- Хотим посчитать среднее время между заказами для каждого пользователя
- Как это сделать?
- Нужно для каждой строчки находить предыдущую

Оконные функции

```
select
   user_id,
   order_datetime,
   lag(order_datetime) as prev_order_datetime over (partition by user_id order by order_datetime)
from orders
```

Ещё про оконные функции

- https://postgrespro.ru/docs/postgrespro/9.5/tutorial-window
- https://postgrespro.ru/docs/postgresql/9.6/functions-window

NULL

- Null не равен ничему (даже себе)
- Проверка на null: my_value is null / my_value is not null
- По умолчанию все колонки допускают null
- Чтобы не допускать null’ов, нужно указывать это в схеме:
- **CREATE TABLE** Persons (
  PersonID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
  );

CASE WHEN

```
SELECT title,
      length,
      CASE
         WHEN length> 0
             AND length <= 50 THEN 'Short'
         WHEN length > 50
             AND length <= 120 THEN 'Medium'
         WHEN length> 120 THEN 'Long'
      END as duration
FROM film
ORDER BY title;
```

CASE WHEN

```
SELECT
        SUM (CASE WHEN rental_rate = 0.99 THEN 1 ELSE 0 END) AS "Economy",
        SUM (CASE WHEN rental_rate = 2.99 THEN 1 ELSE 0 END) AS "Mass",
        SUM (CASE WHEN rental_rate = 4.99 THEN 1 ELSE 0 END ) AS "Premium"
FROM
        film;
```

CASE WHEN

```
SELECT title,
      rating,
      CASE rating
         WHEN 'G' THEN 'General Audiences'
         WHEN 'PG' THEN 'Parental Guidance Suggested'
         WHEN 'PG-13' THEN 'Parents Strongly Cautioned'
         WHEN 'R' THEN 'Restricted'
         WHEN 'NC-17' THEN 'Adults Only'
      END rating_description
FROM film
ORDER BY title;
```

Условные выражения

- GREATEST(1, 2, 3) = 3
- LEAST(10, -2, 10, null) = -2
- COALESCE(value1, value2)

Преобразования между типами

- cast(1 as varchar) -> ‘1’
- cast('1sdfsdf' as int) -> ERROR: invalid input syntax for type integer: "1sdfsdf"
- В postgresql: value::new_type

Транзакции

Транзакция — группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще, и тогда она не должна произвести никакого эффекта.

То же, но по-русски

Транзакция - последовательность операций над данными, имеющая начало и конец.

То же, но по-русски

Транзакция - последовательность операций над данными, имеющая начало и конец.

Например, перевод денег с одного счета на другой:

```
update accounts set balance = balance – 1000 where id = 1;
update accounts set balance = balance + 1000 where id = 2;
```

Begin - commit

```
begin;


update accounts set balance = balance – 1000 where
id = 1;
update accounts set balance = balance + 1000 where
id = 2;


commit;
```

А как отменить это всё?

```
begin;

update accounts set balance = balance – 1000 where id = 1;

select balance from accounts where id = 1 --получаем -50

...
```

А как отменить это всё?

```
begin;

update accounts set balance = balance – 1000 where id = 1;

select balance from accounts where id = 1 --получаем -50

rollback;
```

Транзакции - главное

- Транзакции атомарны – либо выполнится всё, либо не выполнится ничего
- Транзакции независимы друг от друга
- Одиночная команда – тоже транзакция
- Транзакцию можно отменить, не завершая, и вам за это ничего не будет
- А ещё можно отменить только часть транзакции (см. `savepoint / rollback to`)
- Избегайте долгих транзакций

Параллельные транзакции