https://aristov.tech

Аристов Евгений

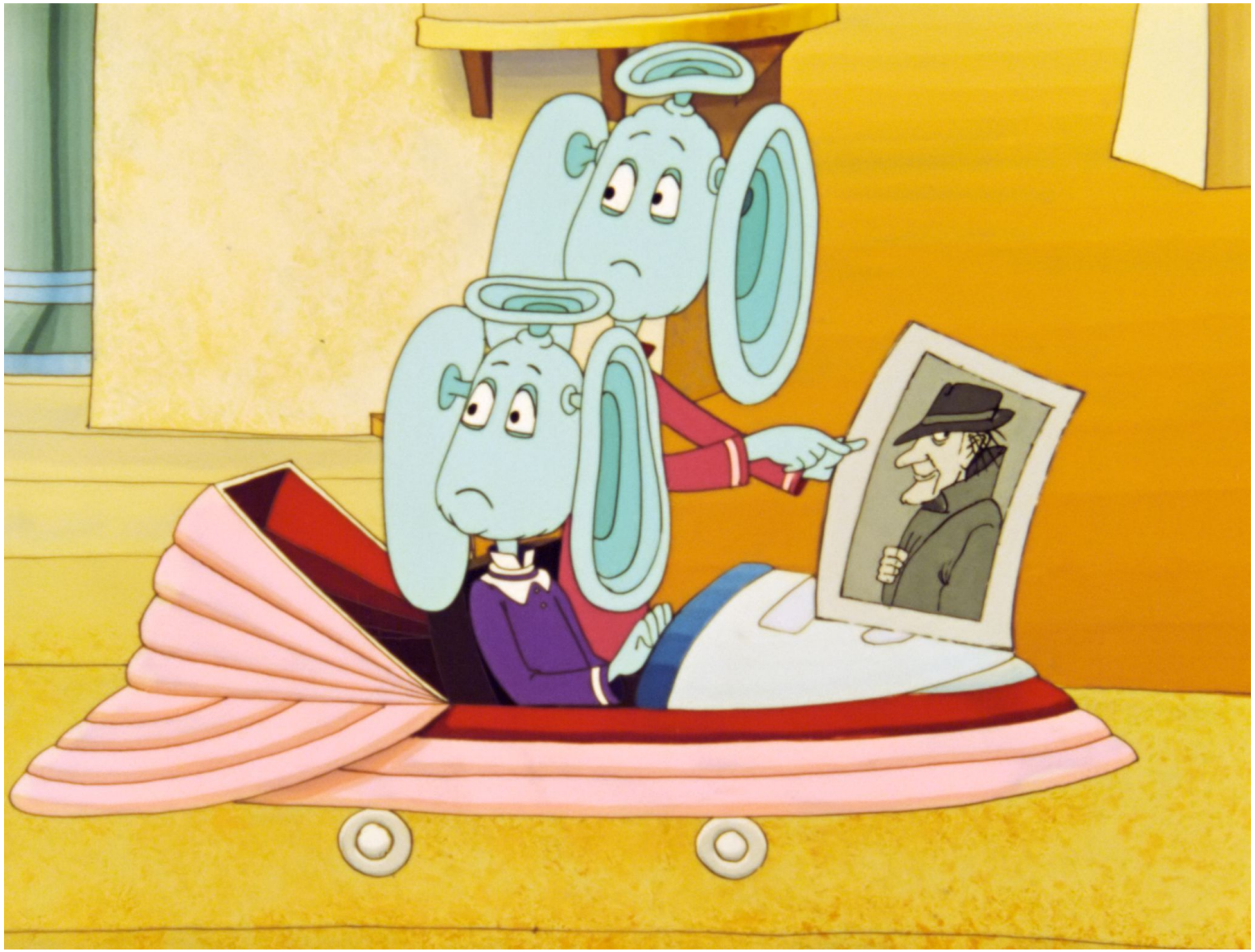
# PL/pgSQL в PostgreSQL

## за 31 занятие


## Функции для работы с массивами, в т.ч. многомерными

https://aristov.tech

https://aristov.tech



https://aristov.tech

https://aristov.tech

**Аристов Евгений Николаевич**

https://aristov.tech

Founder & CEO [aristov.tech](aristov.tech)
25 лет занимаюсь разработкой БД и ПО

Архитектор высоконагруженных баз данных и инфраструктуры

Спроектировал и разработал более ста проектов для финансового сектора, сетевых магазинов, фитнес-центров, отелей.

Сейчас решаю актуальные для бизнеса задачи: аудит и оптимизация БД и инфраструктуры, миграция на PostgreSQL, обучение сотрудников.

Автор более 10 практических курсов по PostgreSQL, MySQL, Mongo и др..

Автор книг по PostgreSQL. Новинка **PostgreSQL 16: лучшие практики оптимизации**

https://aristov.tech

# Правила вебинара

Задаем вопрос в чат

Вопросы вижу, отвечу в момент логической паузы

Если есть вопрос голосом - поставьте знак ? в чат

Если остались вопросы, можно их задать на следующем занятии или в комментариях к записи

https://aristov.tech

https://aristov.tech

# Маршрут вебинара

Создание массива и доступ к элементам

Функции для работы с массивами

Операции с многомерными массивами

https://aristov.tech

https://aristov.tech

# Массивы

https://aristov.tech

https://aristov.tech

# Массивы

Это набор однотипных данных. Определение и доступ к элементам заключаются в квадратные скобки. Нумерация элементов идёт с 1.

Примеры одномерных массивов:

```
DECLARE
    -- Простые массивы
    numbers INTEGER[];
    names TEXT[];
    flags BOOLEAN[];

    -- С инициализацией
    prices NUMERIC[] := ARRAY[10.5, 20.3, 15.7];
    cities TEXT[] := ARRAY['Москва', 'СПб', 'Казань'];
```

https://aristov.tech

https://aristov.tech

# Массивы. Сложные типы

```
DECLARE
    -- Массивы дат
    dates DATE[] := ARRAY['2024-01-01', '2024-01-02'];

    -- Массивы JSON
    json_data JSON[] := ARRAY['{"name": "John"}', '{"name": "Jane"}'];

    -- Массивы UUID
    uuids UUID[] := ARRAY[
        '550e8400-e29b-41d4-a716-446655440000'::uuid,
        'f47ac10b-58cc-4372-a567-0e02b2c3d479'::uuid
    ];
```

https://aristov.tech

https://aristov.tech

# Массивы. Доступ

```
DECLARE
    fruits TEXT[] := ARRAY['яблоко', 'банан', 'апельсин'];
    first_fruit TEXT;
    last_fruit TEXT;
    sliced_fruits TEXT[];
BEGIN
    -- Доступ к элементам (индексация с 1)
    first_fruit := fruits[1];
    last_fruit := fruits[array_length(fruits, 1)];

    -- Изменение элементов
    fruits[2] := 'груша';

    -- Срез массива
    sliced_fruits := fruits[1:2];  -- ['яблоко', 'груша']
```

https://aristov.tech

https://aristov.tech

# Функции для работы с массивами

https://aristov.tech

https://aristov.tech

# Массивы. Функции

```
-- Добавление элементов
numbers := array_append(numbers, 4);        -- [1, 2, 3, 4]
numbers := array_prepend(0, numbers);       -- [0, 1, 2, 3, 4]
numbers := array_cat(numbers, ARRAY[5, 6]); -- [0, 1, 2, 3, 4, 5, 6]

-- Удаление элементов
numbers := array_remove(numbers, 3);        -- Удалить все 3
numbers := array_remove(numbers, 999);      -- Ничего не изменится, так как массив меньшей длины

-- Фильтрация (через unnest)
SELECT ARRAY(SELECT unnest(numbers) WHERE unnest % 2 = 0)
INTO filtered_numbers;
```

https://aristov.tech

https://aristov.tech

# Массивы. Доп.функции

```
-- Длина массива
result := array_length(arr, 1)::text; RETURN NEXT;

-- Поиск элемента
searched_index := array_position(arr, 8);
result := COALESCE(searched_index::text, 'не найден');

-- Содержит ли элемент
result := (5 = ANY(arr))::text;

-- Пересечение массивов
result := array_to_string(arr && ARRAY[3,7], ','); – не сработает, посмотрим на практике
```

https://aristov.tech

https://aristov.tech

# Массивы. Доп.функции

```
-- Сортировка
 sorted_arr := ARRAY(SELECT unnest(arr) ORDER BY 1);
 result := array_to_string(sorted_arr, ',');


 -- Реверс
 reversed_arr := ARRAY(SELECT unnest(arr) ORDER BY ordinality DESC FROM unnest(arr) WITH ORDINALITY); – аналогично нерабочий код
 result := array_to_string(reversed_arr, ',');
```

https://aristov.tech

https://aristov.tech

# Многомерные массивы

https://aristov.tech

https://aristov.tech

# Многомерные массивы

Примеры двухмерных массивов:

```
DECLARE
    numbers INTEGER[][];
    names TEXT[][];


    -- доступ
i = numbers[1][1]
```

Трехмерных:

```
    numbers INTEGER[][][];
i = numbers[1][1][1]
```

Четырёх и тд аналогично

https://aristov.tech

https://aristov.tech

# Цикл по элементам многомерного массива FOR

Цикл FOREACH очень похож на FOR. Отличие в том, что вместо перебора строк SQL-запроса происходит перебор элементов массива. Синтаксис цикла FOREACH:

```
[ <<метка>> ]
FOREACH цель [ SLICE размерность_массива] IN ARRAY выражение LOOP
   операторы
END LOOP [ метка ];
```

Без указания SLICE, или если SLICE равен 0, цикл выполняется по всем элементам массива, полученного из *выражения*. Переменной *цель* последовательно присваивается каждый элемент массива и для него выполняется тело цикла.

https://aristov.tech

https://aristov.tech

# Итоги

Массивы в PL/pgSQL — это мощный инструмент для:

- **Пакетной обработки** данных
- **Динамических запросов** с множественными условиями
- **Временного хранения** наборов значений
- **Упрощения сложной логики**

Особенности:

- аффектит на производительность, особенно удаление элемента
- прямой доступ без проверки value := my_array[100];  -- может быть ошибка, если элемента нет или тип не совпадает
- по факту дорого аннестить и собирать обратно - лучше напрямую с массивами работать

https://aristov.tech

https://aristov.tech

# Практика

https://aristov.tech

https://aristov.tech

# Итоги

https://aristov.tech

https://aristov.tech

# Итоги

Остались ли вопросы?

Увидимся на следующем занятии

https://aristov.tech

https://aristov.tech

# Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений

https://aristov.tech

Все материалы защищены авторским правом. Использование полностью или частично разрешено только с письменного разрешения Аристов Е.Н. (с)