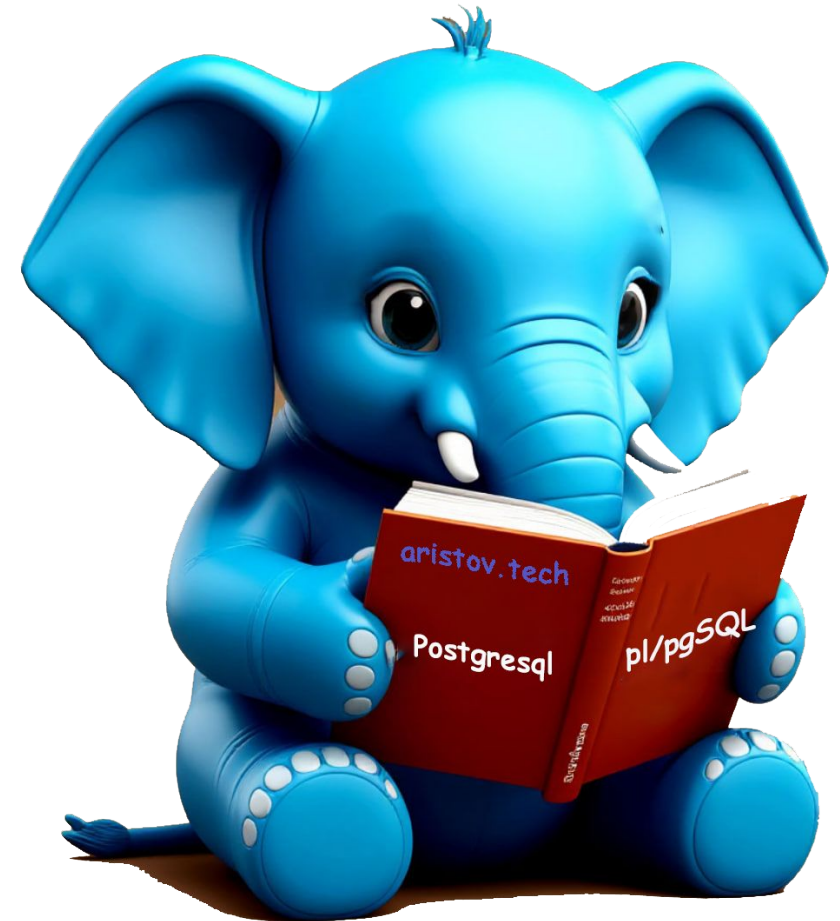


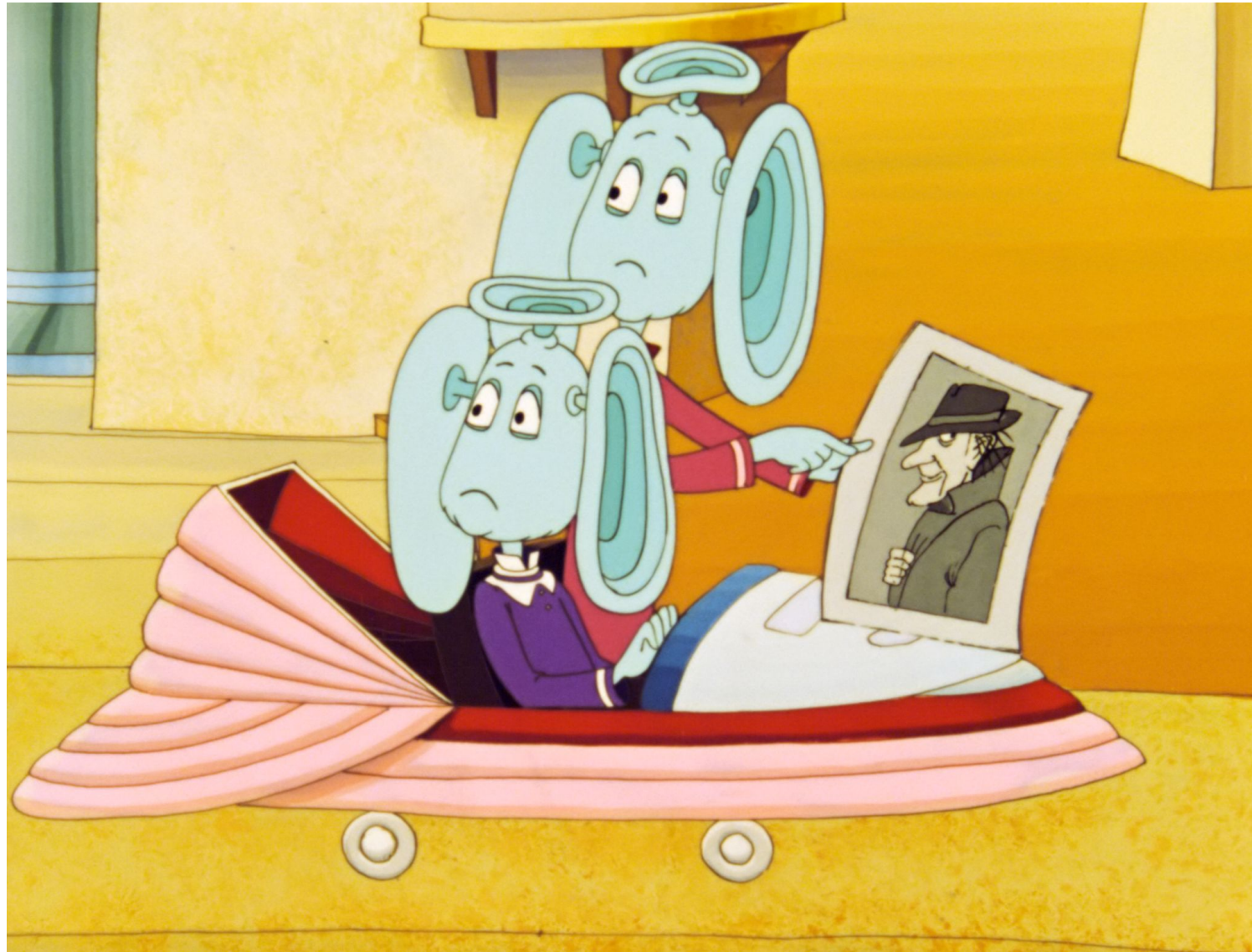
Аристов Евгений

PL/pgSQL в PostgreSQL

за 31 занятие

Использование кортежей





**Аристов
Евгений
Николаевич**



<https://aristov.tech>

Founder & CEO aristov.tech

<https://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](#)

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

Маршрут вебинара

Назначение кортежей

Преимущества и недостатки

Бест практис

Кортежи

Кортежи

ROWTYPE в PostgreSQL позволяет работать со строками таблиц и составными типами как с едиными объектами.

DECLARE

emp_record **employees**%**ROWTYPE**; -- Объявление переменной типа строки таблицы

Кортежи. Преимущества

Автоматическая адаптация к схеме:

При изменении таблицы не нужно менять код функций

```
ALTER TABLE employees ADD COLUMN email TEXT;
```

Функция продолжит работать без изменений!

Переменная `emp_record` автоматически включает новое поле

Кортежи. Преимущества

Удобство работы с целыми строками:

```
DECLARE
```

```
    original_emp employees%ROWTYPE;
```

Модифицируем нужные поля

```
    original_emp.name := new_name;
```

```
    original_emp.id := DEFAULT;
```

```
    INSERT INTO employees VALUES (original_emp.*) RETURNING id INTO new_emp_id;
```

```
    RETURN new_emp_id;
```

Кортежи. Преимущества

Сравнение целых строк:

```
current_emp employees%ROWTYPE;
```

```
cached_emp employees%ROWTYPE;
```

-- Текущее состояние

```
SELECT * INTO current_emp FROM employees WHERE id = emp_id;
```

-- Предположим, что у нас есть кэш (например, в другой таблице)

```
SELECT * INTO cached_emp FROM employees_cache WHERE id = emp_id;
```

-- Сравниваем всю строку целиком

```
IF current_emp IS DISTINCT FROM cached_emp THEN ....
```

Кортежи. Недостатки

Производительность при частичном использовании:

НЕЭФФЕКТИВНО: выбираем все поля, но используем только одно

```
CREATE OR REPLACE FUNCTION get_employee_name(emp_id INTEGER)
```

```
RETURNS TEXT AS $$
```

```
DECLARE
```

```
    emp employees%ROWTYPE; -- Выбираем ВСЕ поля
```

```
BEGIN
```

```
    SELECT * INTO emp FROM employees WHERE id = emp_id;
```

```
    RETURN emp.name; -- Используем только одно поле!
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

Кортежи. Недостатки

Отсутствие строгой типизации при изменениях:

Проблема: если удалить поле из таблицы

```
ALTER TABLE employees DROP COLUMN salary;
```

Функция сломается в RUNTIME, а не при компиляции

Ошибка: column "salary" of relation "employees" does not exist

Кортежи. Недостатки

Сложность с NULL значениями:

```
DECLARE
```

```
    emp employees%ROWTYPE;
```

```
BEGIN
```

```
-- Инициализация всей строки как NULL
```

```
emp := NULL;
```

```
-- Попытка доступа к полю вызовет ERROR
```

```
-- RAISE NOTICE 'Name: %', emp.name; -- ОШИБКА!
```

Кортежи. Лучшие практики

Используйте ROWTYPE для:

- ❖ Триггерных функций (NEW/OLD)
- ❖ Операций со всей строкой
- ❖ Прототипирования

Избегайте ROWTYPE для:

- ❖ Функций, использующих 1-2 поля
- ❖ High-performance кода
- ❖ Часто изменяемых таблиц

Практика

Итоги

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений