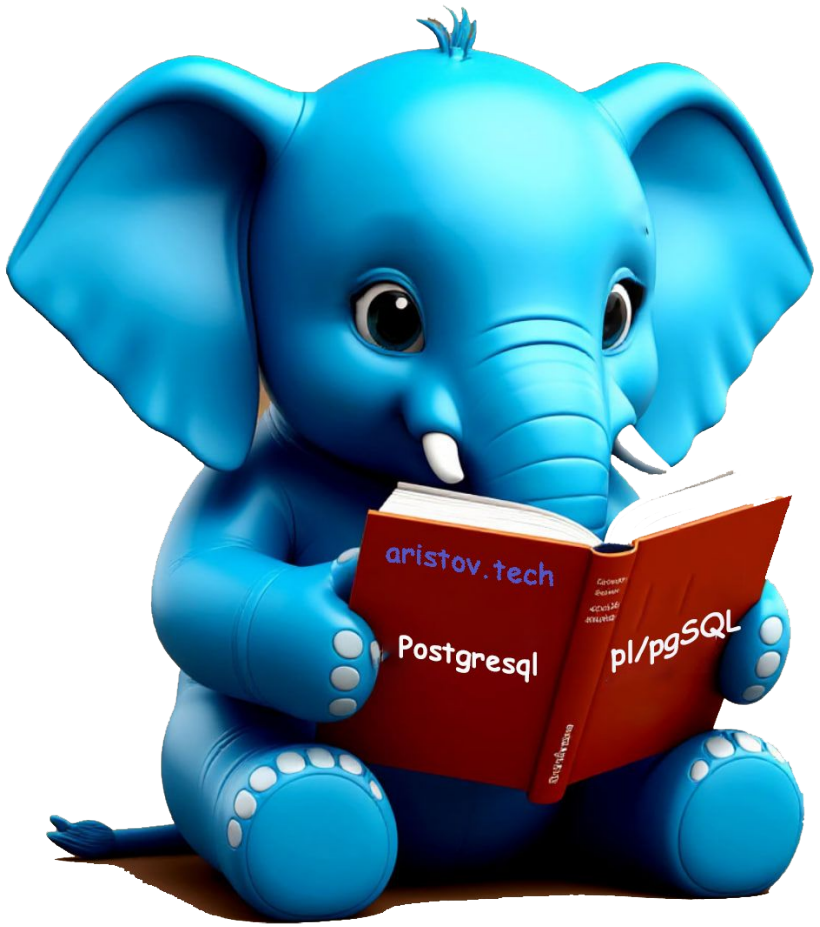
Аристов Евгений

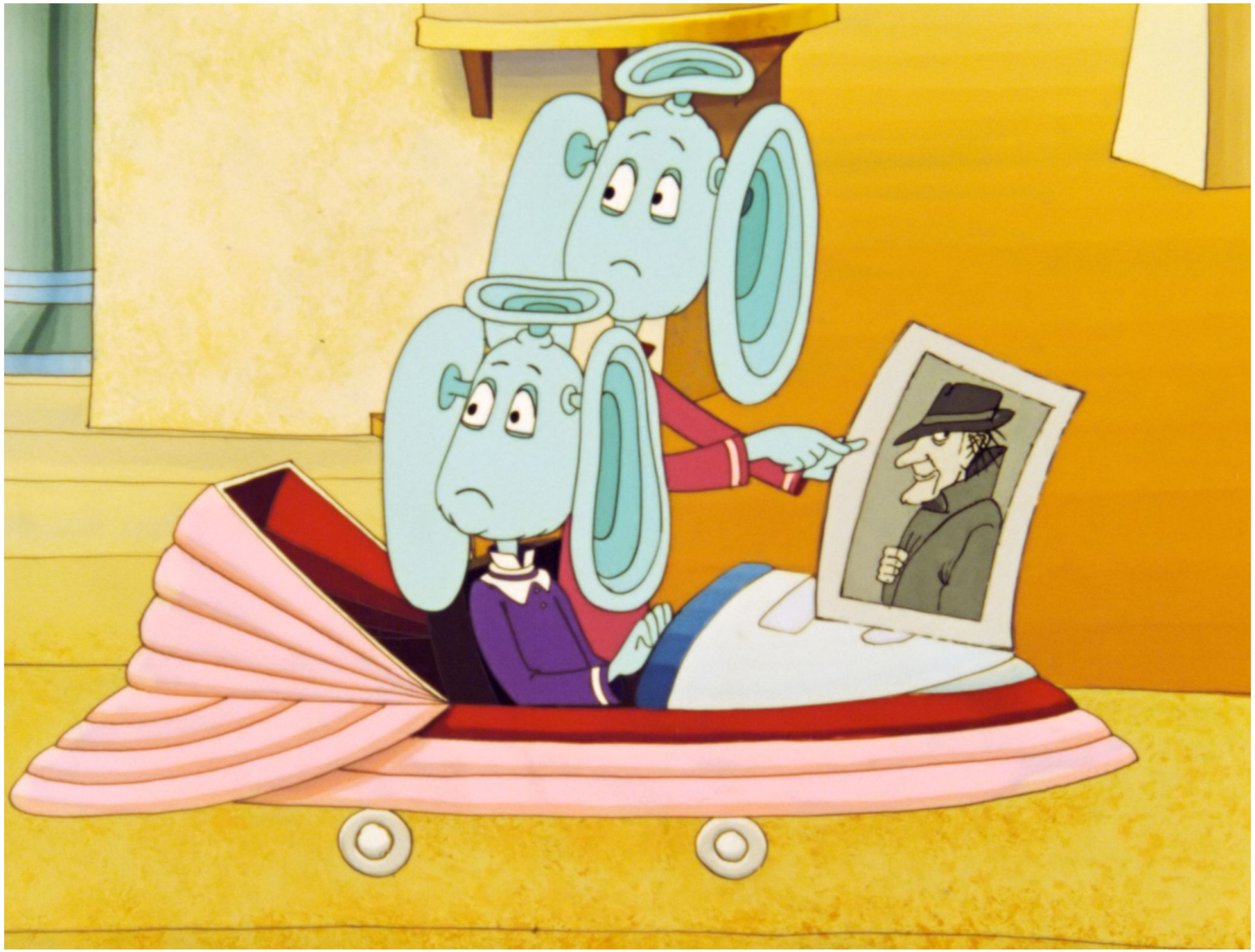
PL/pgSQL в PostgreSQL

за 31 занятие



Различие SQL, PL/pgSQL, PL/Python

https://aristov.tech



Аристов Евгений Николаевич

Founder & CEO 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

https://aristov.tech

Маршрут вебинара

Назначение других языков

Отличия языков

Особенности синтаксиса PL/pgSQL

Примеры использования PL/Python

https://aristov.tech

Язык PL/PgSQL

https://aristov.tech

Язык PL/PgSQL

Несмотря на то, что мы можем писать функции на чистом SQL, в Постгресе был придуман PL/pgSQL.

Основные цели загружаемого процедурного языка:

- может выполнять сложные вычисления,
- используется для создания функций и триггеров,
- добавляет управляющие структуры к языку SQL,
- наследует все пользовательские типы, функции и операторы,
- прост в использовании.

Функции PL/pgSQL могут использоваться везде, где допустимы встроенные функции. Например, можно создать функции со сложными вычислениями и условной логикой, а затем использовать их при определении операторов или в индексных выражениях.

В версии PostgreSQL 9.0 и выше, PL/pgSQL устанавливается по умолчанию

Язык PL/PgSQL

PL/pgSQL позволяет сгруппировать блок вычислений и последовательность запросов *внутри* сервера базы данных, таким образом, мы получаем силу процедурного языка и простоту использования SQL при значительной экономии накладных расходов на клиент-серверное взаимодействие.

Важные особенности:

- Расширяется функционал простого SQL
- Доступ к дополнительному, в т.ч. внешнему функционалу
- Расширенные варианты возврата значений из функций
- Промежуточные ненужные результаты не передаются между сервером и клиентом
- Есть возможность избежать многочисленных разборов одного запроса
- Медленнее простого SQL

В результате это приводит к значительному увеличению производительности по сравнению с приложением, которое не использует хранимых функций.

Кроме того, PL/pgSQL позволяет использовать все типы данных, операторы и функции SQL.

Особенности PL/PgSQL

```
CREATE FUNCTION sales(total real) RETURNS real AS $$
BEGIN
   RETURN (4 лекция) total * 2;
END;
$$ LANGUAGE plpgsql;
```

https://aristov.tech

Caveats

Важно не путать использование BEGIN/END для группировки операторов в PL/pgSQL с одноимёнными SQL-командами для управления транзакциями. BEGIN/END в PL/pgSQL служат только для группировки предложений; они не начинают и не заканчивают транзакции.

PL/Python

PL/Python

PL/Python — это процедурное расширение для PostgreSQL, которое позволяет писать функции, триггеры и процедуры на языке Python. Его использование оправдано в specific scenarios, где встроенные возможности SQL и PL/pgSQL недостаточны.

Ключевые причины использования PL/Python:

1. Доступ к богатой экосистеме Python

- **Библиотеки для Data Science:** Интеграция с pandas, numpy, scikit-learn для сложной аналитики и ML прямо внутри БД.
- **Работа с данными:** Использование json, xml, yaml парсеров для обработки сложных структур.
- **Сетевые возможности:** Вызов API (requests), парсинг веб-страниц (BeautifulSoup), работа с сетевыми протоколами.
- **Системные вызовы:** Взаимодействие с ОС (например, работа с файловой системой).

https://aristov.tech

Ключевые причины использования PL/Python:

2. Сложные вычисления и алгоритмы

- **Машинное обучение:** Обучение и применение моделей напрямую в БД (например, классификация данных в реальном времени).
- **Статистический анализ:** Расчёты, которые сложно реализовать на SQL (например, регрессия, кластеризация).
- **Криптография:** Хеширование, шифрование с использованием библиотек типа cryptography.

Ключевые причины использования PL/Python:

3. Интеграция с внешними системами

- **Веб-сервисы:** Автоматическая отправка данных в CRM, ERP или другие системы через API.
- **Работа с очередями:** Отправка задач в RabbitMQ, Kafka или Redis.
- **Нотификации:** Отправка email (smtplib) или сообщений в Slack/Telegram через вебхуки.

https://aristov.tech

Ключевые причины использования PL/Python:

4. Преобразование данных

- **Парсинг сложных форматов:** Обработка нестандартных текстовых или бинарных данных.
- **Конвертация данных:** Например, преобразование изображений в текстовое описание с помощью OCR (Tesseract).
- **Генерация контента:** Создание отчетов в PDF, Excel или HTML прямо внутри БД.

https://aristov.tech

Ключевые причины использования PL/Python:

5. Автоматизация административных задач

- **Кастомный мониторинг:** Проверка состояния БД и отправка алертов.
- **Резервное копирование:** Интеграция с облачными хранилищами (S3, Google Cloud Storage).
- **Динамическое управление:** Изменение конфигураций БД на основе условий.

Примеры использования PL/Python:

Вызов внешнего API

```
CREATE FUNCTION get_currency_rate(currency VARCHAR)
RETURNS JSON
AS $$
    import requests
    response = requests.get(f'https://api.exchangerate.host/latest?base={currency}')
    return response.json()
$$ LANGUAGE plpython3u;
```

https://aristov.tech

Примеры использования PL/Python:

Работа с файловой системой

```
CREATE FUNCTION read_logs(log_path VARCHAR)
RETURNS TEXT[]
AS $$
    with open(log_path, 'r') as f:
        return f.readlines()
$$ LANGUAGE plpython3u;
```

https://aristov.tech

Примеры использования PL/Python:

Классификация текста с помощью ML

```
CREATE FUNCTION classify_text(text TEXT)
RETURNS VARCHAR
AS $$
    import pickle
    model = pickle.load(open('/models/text_classifier.pkl', 'rb'))
    return model.predict([text])[0]
$$ LANGUAGE plpython3u;
```

https://aristov.tech

Ограничения и риски

- **Безопасность:**
  - Используйте только **plpython3u** (untrusted), если нет полного доверия к пользователям.
  - Риск выполнения опасного кода (например, системных вызовов).
- **Производительность:**
  - Запуск Python-кода требует больше ресурсов, чем нативный SQL.
  - Не подходит для высоконагруженных OLTP-операций.
- **Зависимости:**
  - Необходимость установки Python-библиотек на сервере БД.
  - Совместимость версий Python и библиотек.
- **Переносимость:**
  - Привязка к PostgreSQL и конкретной версии Python.

https://aristov.tech

Практика

Итоги

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений