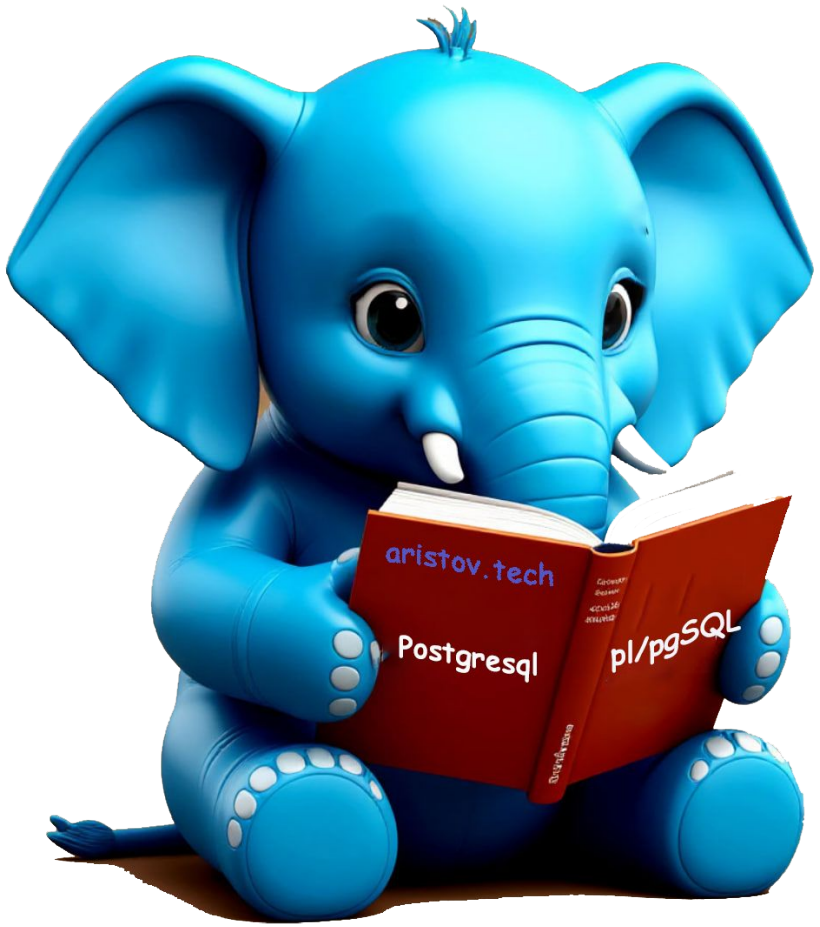
Аристов Евгений

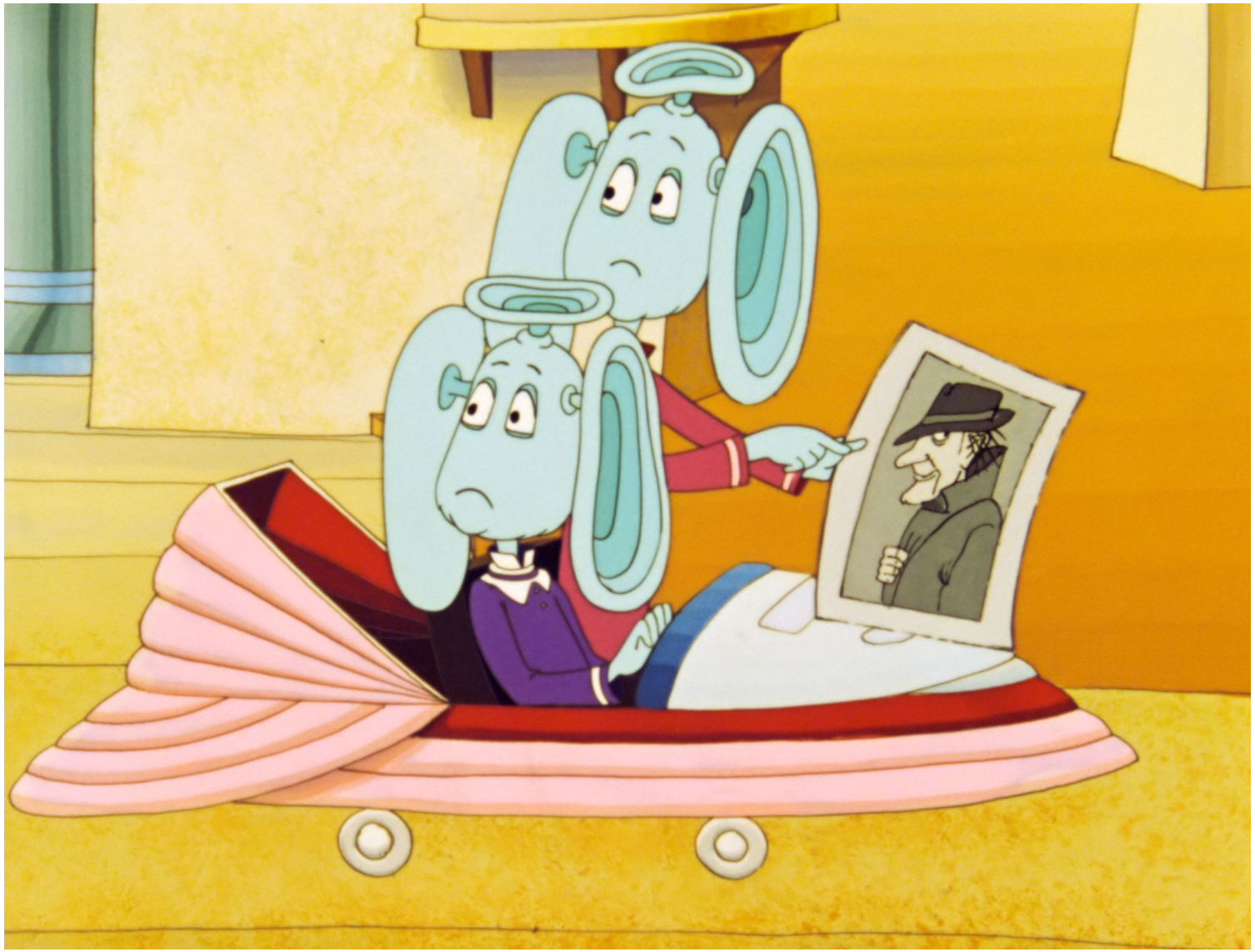
PL/pgSQL в PostgreSQL

за 31 занятие



Динамический SQL

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

Причины использования динамического SQL

Варианты защиты от SQL инъекций

Подготовленные запросы

https://aristov.tech

Динамический SQL

https://aristov.tech

Динамический SQL

Часто требуется динамически формировать команды внутри функций на PL/pgSQL, то есть такие команды, в которых при каждом выполнении могут использоваться разные таблицы или типы данных. Обычно PL/pgSQL кеширует планы выполнения процедур, но в случае с динамическими командами это не будет работать.

https://www.postgresql.org/docs/current/plpgsql-implementation.html#PLPGSQL-PLAN-CACHING

Для исполнения динамических команд предусмотрен оператор EXECUTE:

EXECUTE *строка* [INTO [STRICT] *цель*] [USING *выражение* [, ...]];

где *строка* это выражение, формирующее текст SQL команды, которую нужно выполнить. Необязательная *цель* — это переменная или разделённый запятыми список простых переменных и полей записи/кортежа, куда будут помещены результаты команды. Необязательные выражения в USING формируют значения, которые будут вставлены в команду.

Динамический SQL

В сформированном тексте команды замена имён переменных PL/pgSQL на их значения проводиться не будет.

Все необходимые значения переменных должны быть вставлены в командную строку при её построении, либо нужно использовать параметры, как описано ниже.

Обратите внимание, что нет никакого плана кеширования для команд, выполняемых с помощью EXECUTE. Вместо этого план создаётся каждый раз при выполнении.

То есть, строка команды может динамически создаваться внутри функции для выполнения действий с различными таблицами и столбцами.

Динамический SQL

В тексте команды можно использовать значения параметров, ссылки на параметры обозначаются как $1, $2 и т. д. Эти символы указывают на значения, находящиеся в предложении USING. Такой метод зачастую предпочтительнее, чем вставка значений в команду в виде текста: он позволяет исключить во время исполнения дополнительные расходы на преобразования значений в текст и обратно, и **не открывает возможности для SQL-инъекций**, не требуя применять экранирование или кавычки для спецсимволов.

SQL injection для начинающих. Часть 1

Пример:

```
EXECUTE 'SELECT count(*) FROM mytable WHERE inserted_by = $1 AND inserted <= $2'
INTO c
USING checked_user, checked_date;
```

Динамический SQL

При работе с динамическими командами часто приходится иметь дело с экранированием одинарных кавычек, чтобы не допустить SQL инъекций.

Можно напрямую вызывать функции заключения в кавычки:

```
EXECUTE 'UPDATE tbl SET '
|| quote_ident(colname)
|| ' = '
|| quote_literal(newvalue)
|| ' WHERE key = '
|| quote_literal(keyvalue);
```

https://aristov.tech

Динамический SQL. Подготовленный запрос

Мы можем заранее создать запрос.

Постгрес его оптимально подготовит и закеширует план выполнения - не нужно будет каждый раз его строить. В дальнейшем мы можем использовать этот подготовленный запрос и передавать туда параметры, не изменяя структуры запроса. Если в запросе изменится набор полей и тд, нужно будет создавать новый подготовленный запрос:

```
PREPARE запрос AS имя
EXECUTE имя(параметры)
```

https://www.postgresql.org/docs/current/sql-prepare.html

Подготовленный запрос. Примеры

```
PREPARE fooplan (int, text, bool, numeric) AS
    INSERT INTO foo VALUES($1, $2, $3, $4);
EXECUTE fooplan(1, 'Hunter Valley', 't', 200.00);

PREPARE usrrptplan (int) AS
    SELECT * FROM users u, logs l WHERE u.usrid=$1 AND u.usrid=l.usrid
    AND l.date = $2;
EXECUTE usrrptplan(1, current_date);
```

https://aristov.tech

Итоги

Помним про опасность SQL инъекций и методы защиты от них

Подготовленные запросы - прямой путь для оптимизации производительности

Практика

Итоги

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений