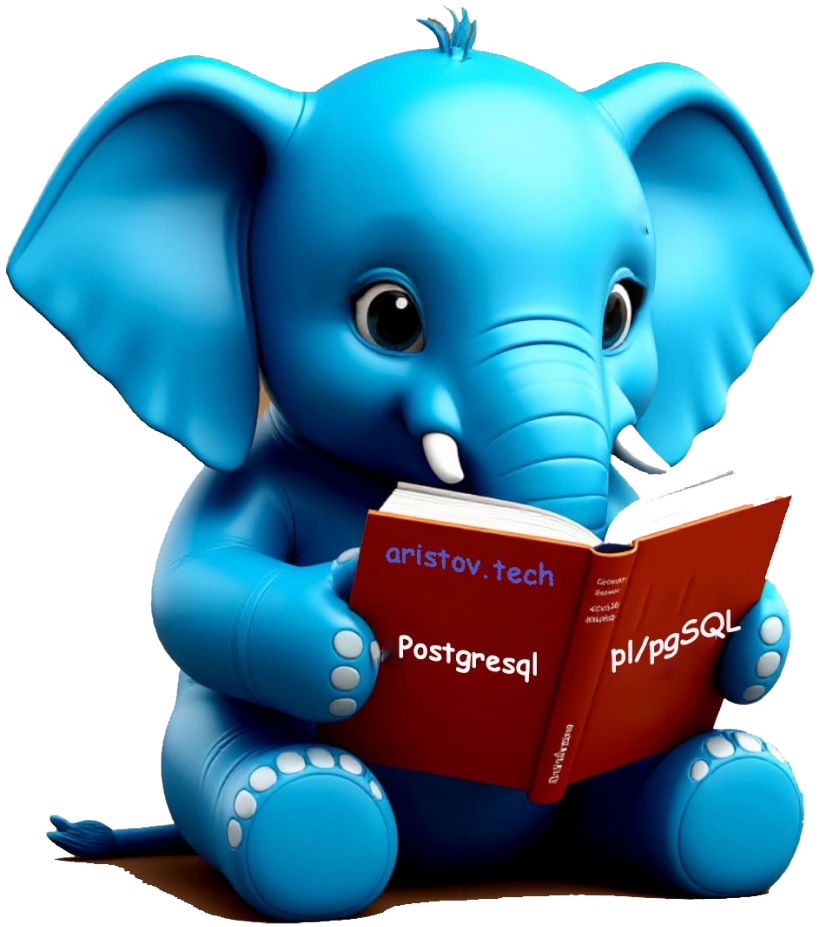
Аристов Евгений

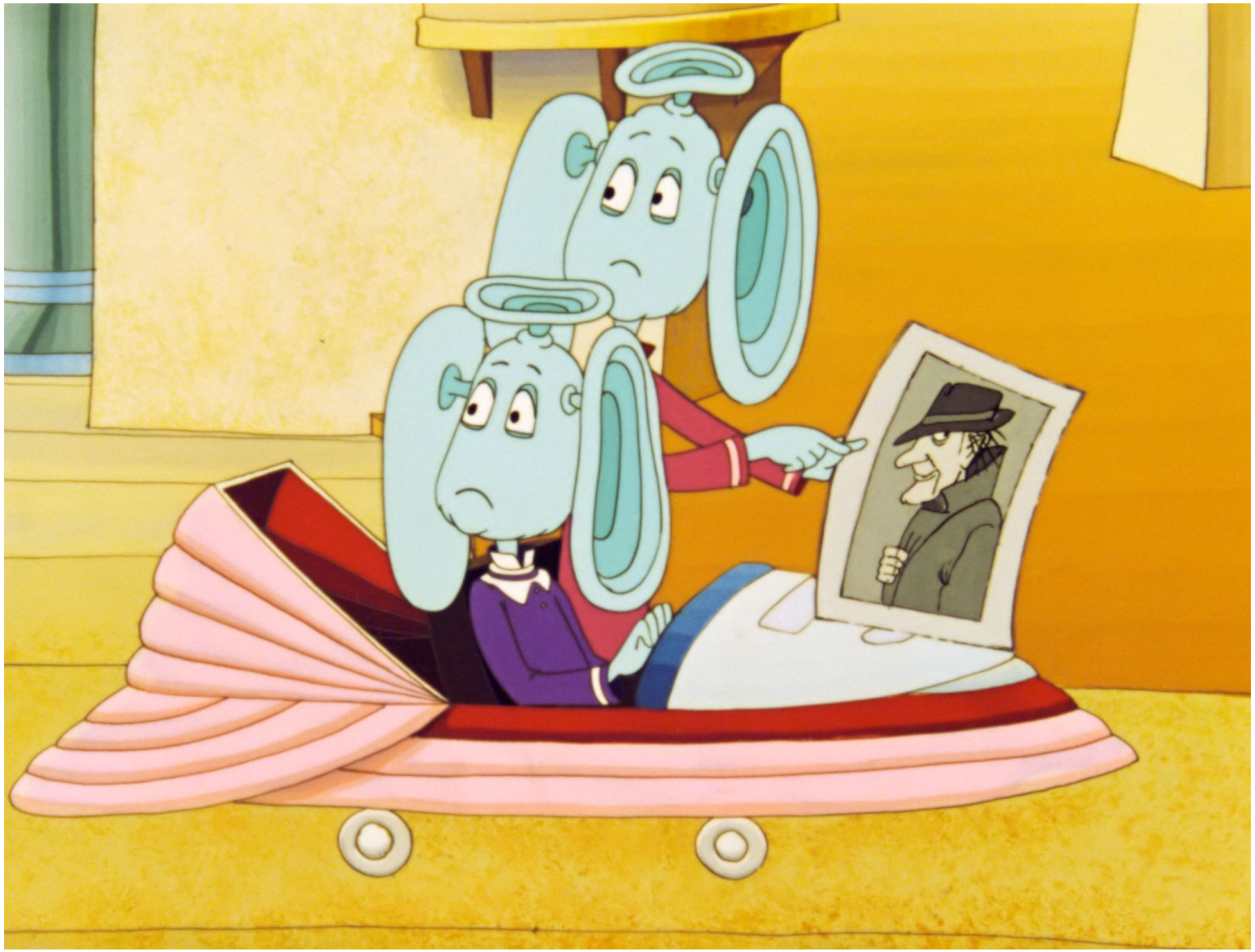
PL/pgSQL в PostgreSQL

за 31 занятие



DML триггеры

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

Принцип работы

Триггерная функция

Варианты триггеров

Условный триггер

Кейсы использования

https://aristov.tech

Виды триггеров, продакшн кейсы и особенности использования

https://aristov.tech

Триггерные функции

В стандартных RDBMS триггер - функция срабатывающая при выполнении определенных условий.

В PL/pgSQL пошли дальше - можно создавать триггерные функции, которые будут вызываться триггером при изменениях данных или событиях в базе данных.

Соответственно можно на разные события вызывать одну и ту же функцию, что в целом превращает триггеры в модульную конструкцию.

Триггерная функция создаётся командой CREATE FUNCTION, при этом у функции не должно быть аргументов, а **типом возвращаемого значения** должен быть **trigger** (для триггеров, срабатывающих при изменениях данных - **DML**) или **event_trigger** (для триггеров, срабатывающих при событиях в базе - **DDL**).

Для триггеров автоматически определяются специальные локальные переменные с именами вида TG_имя, описывающие условие, повлёкшее вызов триггера.

https://www.postgresql.org/docs/current/plpgsql-trigger.html

https://aristov.tech

Создание триггера

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

CREATE [OR REPLACE] [CONSTRAINT] TRIGGER *name* { BEFORE | AFTER | INSTEAD OF } { *event* [OR ...] }

ON *table_name*

[FROM *referenced_table_name*]

[NOT DEFERRABLE | [DEFERRABLE] [INITIALLY IMMEDIATE | INITIALLY DEFERRED]]

[REFERENCING { { OLD | NEW } TABLE [AS] *transition_relation_name* } [...]]

[FOR [EACH] { ROW | STATEMENT }]

[WHEN (*condition*)]

EXECUTE { FUNCTION | PROCEDURE } *function_name* (*arguments*)

where *event* can be one of:

INSERT

UPDATE [OF *column_name* [, ...]]

DELETE

TRUNCATE

Триггеры при изменении данных

Триггерная функция должна объявляться **без аргументов**, даже если ожидается, что она будет получать аргументы, заданные в команде CREATE TRIGGER — такие аргументы передаются через TG_ARGV, как описано ниже.

NEW

Тип данных RECORD. Переменная содержит новую строку базы данных для команд **INSERT/UPDATE** в триггерах уровня строки. В триггерах уровня оператора и для команды DELETE эта переменная имеет значение null.

OLD

Тип данных RECORD. Переменная содержит старую строку базы данных для команд **UPDATE/DELETE** в триггерах уровня строки. В триггерах уровня оператора и для команды INSERT эта переменная имеет значение null.

https://aristov.tech

Передаваемые параметры

TG_NAME

Тип данных name. Переменная содержит имя сработавшего триггера.

TG_WHEN

Тип данных text. Строка, содержащая BEFORE, AFTER или INSTEAD OF, в зависимости от определения триггера.

TG_LEVEL

Тип данных text. Строка, содержащая ROW или STATEMENT, в зависимости от определения триггера. При указании STATEMENT нужно указать временную таблицу, содержащую все результирующие промежуточные строки

TG_OP

Тип данных text. Строка, содержащая INSERT, UPDATE, DELETE или TRUNCATE, в зависимости от того, для какой операции сработал триггер.

TG_RELID

Тип данных oid. OID таблицы, для которой сработал триггер.

Передаваемые параметры

TG_TABLE_NAME

Тип данных name. Имя таблицы, для которой сработал триггер.

TG_TABLE_SCHEMA

Тип данных name. Имя схемы, содержащей таблицу, для которой сработал триггер.

TG_NARGS

Тип данных integer. Число аргументов в команде CREATE TRIGGER, которые передаются в триггерную функцию.

TG_ARGV[]

Тип данных массив text. Аргументы от оператора CREATE TRIGGER. Индекс массива начинается с 0. Для недопустимых значений индекса (< 0 или >= tg_nargs) возвращается NULL.

https://aristov.tech

Возвращаемые значения

Триггерная функция должна вернуть либо NULL, либо запись/строку, соответствующую структуре таблице, для которой сработал триггер.

Если BEFORE триггер уровня строки возвращает **NULL**, то все дальнейшие действия с этой строкой **прекращаются** (т. е. не срабатывают последующие триггеры, команда INSERT/UPDATE/DELETE для этой строки не выполняется). Если возвращается не NULL, то дальнейшая обработка продолжается именно с этой строкой. Возвращение строки отличной от начальной NEW, изменяет строку, которая будет вставлена или изменена.

Если в триггерной функции нужно выполнить некоторые действия и не менять саму строку, то нужно возвратить переменную NEW (или её эквивалент). Для того чтобы изменить сохраняемую строку, можно поменять отдельные значения в переменной NEW и затем её вернуть. Либо создать и вернуть полностью новую переменную.

В случае строчного триггера BEFORE для команды DELETE само возвращаемое значение не имеет прямого эффекта, но оно должно быть отличным от NULL, чтобы не прерывать обработку строки. Обратите внимание, что переменная NEW всегда NULL в триггерах на DELETE, поэтому возвращать её не имеет смысла. Традиционной практикой для триггеров DELETE является возврат переменной OLD.

Условный триггер

```
CREATE TRIGGER conditional_trigger
    AFTER UPDATE ON large_table
    FOR EACH ROW
    WHEN (OLD.important_column IS DISTINCT FROM NEW.important_column)
    EXECUTE FUNCTION important_changes_only();
```

Продакшн кейсы

- для сложных бизнес проверок/интеграций (опасно ходить в другие системы - можем сами себя заддосить)
- для довнесения изменений - дата записи, пользователь и тд
- для аудита в PL/pgSQL
- для ведения таблицы итогов
- поддержания целостности денормализованных данных

Пример на практике:

1. Напишем простую табличку магазин (товар, остаток, цена)
2. Простую табличку с продажами из этого магазина (товар, колво, сумма)
3. Сделаем триггер, чтобы при продаже товара остатки уменьшались
4. Сделаем триггер, чтобы сумма автоматом проставлялась при продаже

Особенности:

- Может быть несколько триггеров на 1 таблицу - выполняются по типу действия + имя
- Может быть один триггер на несколько таблиц - если имя релейшна, для которого он вызывается
- *Посмотрим на падение производительности при использовании триггеров.*

Практика

Итоги

Итоги

Используйте триггеры осторожно, так как они могут значительно влиять на производительность и усложнять отладку!

Остались ли вопросы?

Увидимся на следующем занятии

Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений