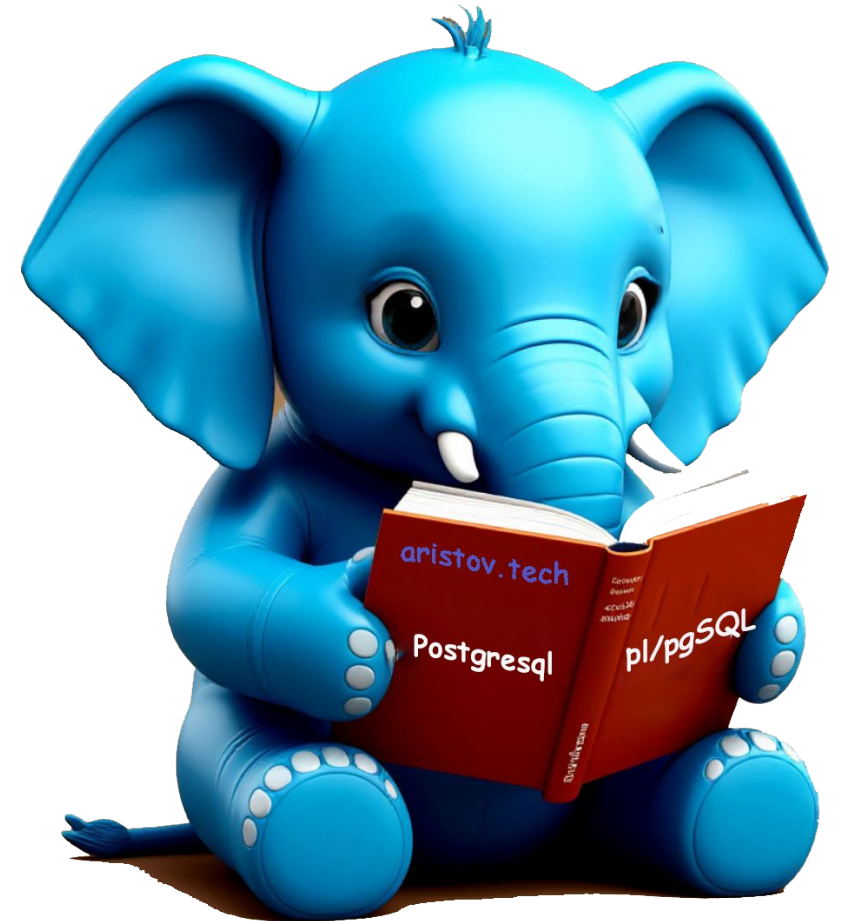


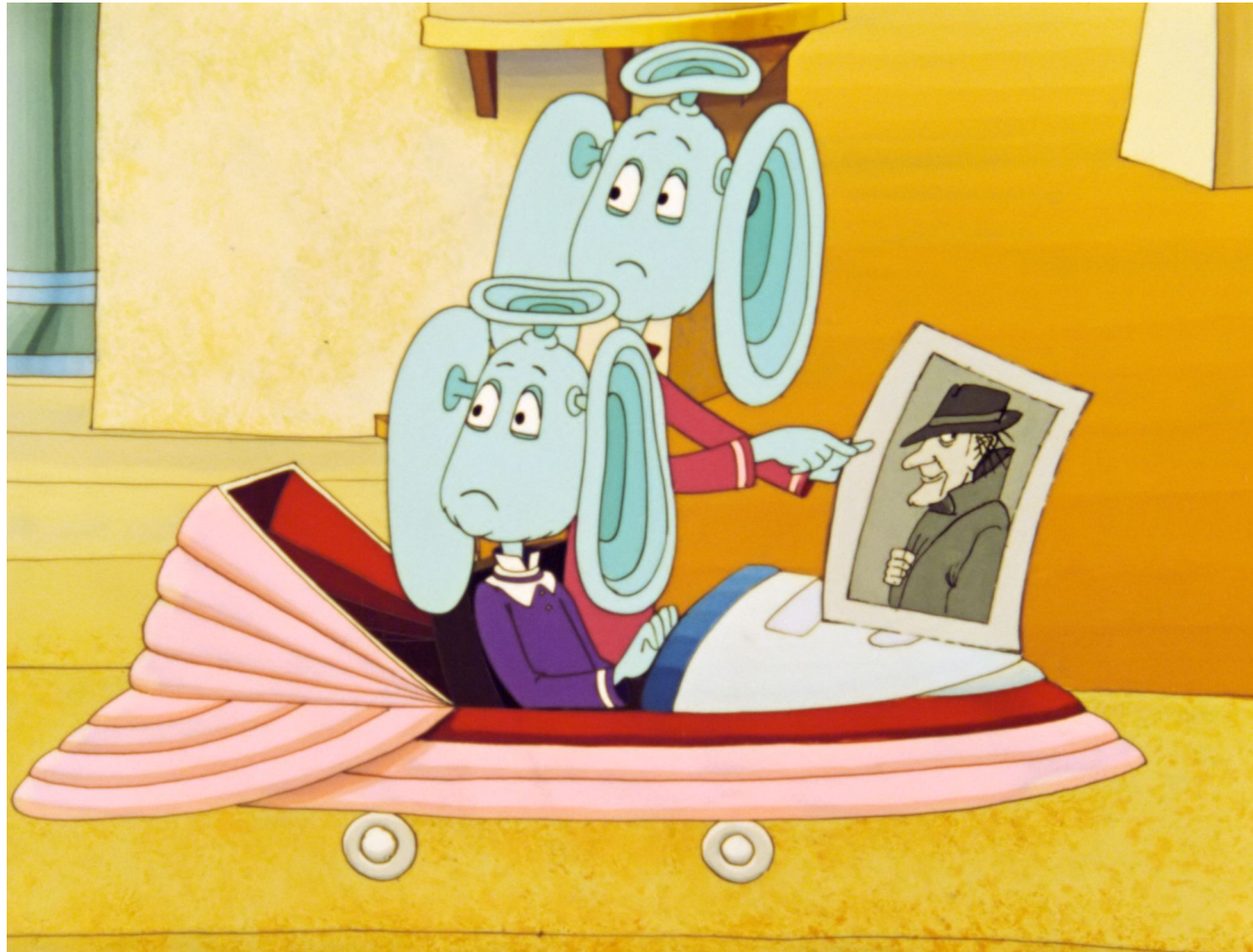
Аристов Евгений

PL/pgSQL в PostgreSQL

за 31 занятие

Операторы ветвления if & case





**Аристов
Евгений
Николаевич**



<https://aristov.tech>

<https://aristov.tech>

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

Маршрут вебинара

Основные управляющие конструкции

Операторы ветвления и их варианты

Язык PL/PGSQL: основные операторы, управляющие конструкции

Основные управляющие конструкции

Наиболее полезная и важная часть PL/pgSQL.

С их помощью можно очень гибко и эффективно манипулировать данными в PostgreSQL.

<https://www.postgresql.org/docs/current/plpgsql-control-structures.html>

Сегодня мы рассмотрим:

- ❖ Ветвления
- ❖ Циклы (24 тема)
- ❖ Исключения (25 тема)

Условные операторы

Операторы IF и CASE позволяют выполнять команды в зависимости от определённых условий. PL/pgSQL поддерживает три формы IF:

- ❖ **IF ... THEN ... END IF**
- ❖ **IF ... THEN ... ELSE ... END IF**
- ❖ **IF ... THEN ... ELSIF ... THEN ... ELSE ... END IF**

и две формы CASE:

- ❖ **CASE ... WHEN ... THEN ... ELSE ... END CASE**
- ❖ **CASE WHEN ... THEN ... ELSE ... END CASE**

IF THEN

IF-THEN это простейшая форма условного оператора.

IF логическое-выражение **THEN**
операторы
END IF;

Операторы между THEN и END IF выполняются, если условие (*логическое выражение*) истинно. В противном случае они опускаются.

Пример:

```
IF user_id <> 0 THEN  
  UPDATE users SET user_email = email WHERE id_user = user_id;  
END IF;
```

IF THEN ELSE

IF-THEN-ELSE добавляет к IF-THEN возможность указать альтернативный набор операторов, которые будут выполнены, если условие не истинно (в том числе, если условие NULL).

```
IF логическое-выражение THEN  
    операторы  
ELSE  
    операторы  
END IF;
```

Обратите внимание, что выполнение команд пойдет только по 1 пути!!!

Если одно из условий будет истинно, дальнейшие сравнения и операторы будут проигнорированы.

IF THEN ELSE. Пример

IF-THEN-ELSE добавляет к IF-THEN возможность указать альтернативный набор операторов, которые будут выполнены, если условие не истинно (в том числе, если условие NULL).

```
IF a=1 THEN
  t := 'a=1'
ELSE IF a=2 THEN
  t := 'a=2'
ELSE
  t := 'a not 1 or 2'
END IF;
```

Обратите внимание, что выполнение команд пойдет только по 1 пути!!!

Если одно из условий будет истинно, дальнейшие сравнения и операторы будут проигнорированы.

IF THEN ELSIF

IF-THEN-ELSIF обеспечивает удобный способ проверки нескольких вариантов по очереди.

В целом это больше синтаксический сахар.

```
IF логическое-выражение THEN
  операторы
[ELSIF логическое-выражение THEN операторы [ELSIF логическое-выражение THEN
операторы ...]]
[ELSE операторы]
END IF;
```

Условия в IF последовательно проверяются до тех пор, пока не будет найдено первое истинное. После этого операторы, относящиеся к этому условию, выполняются, и управление переходит к следующей после END IF команде. (Все последующие условия не проверяются.) Если ни одно из условий IF не является истинным, то выполняется блок ELSE (если присутствует).

IF THEN ELSIF Пример

```
IF n = 0 THEN
    result := 'zero';
ELSIF n > 0 THEN
    result := 'positive';
ELSIF number < 0 THEN
    result := 'negative';
ELSE
    -- остаётся только один вариант: number имеет значение NULL
    result := 'NULL';
END IF;
```

CASE

Простая форма CASE реализует условное выполнение на основе сравнения операндов.

```
CASE выражение-поиска  
  WHEN выражение [, выражение [...]] THEN  
    операторы  
  [WHEN выражение [, выражение [...]] THEN операторы ...]  
  [ELSE операторы]  
END CASE;
```

Выражение-поиска вычисляется (один раз) и последовательно сравнивается с каждым *выражением* в условиях WHEN. Если совпадение найдено, то выполняются соответствующие *операторы* и управление переходит к следующей после END CASE команде. (Все последующие *выражения* WHEN не проверяются.)

Если совпадение не было найдено, то выполняются *операторы* в ELSE. Но если ELSE нет, то вызывается исключение CASE_NOT_FOUND.

CASE. Пример

```
CASE x  
  WHEN 1, 2 THEN  
    msg := 'один или два';  
  ELSE  
    msg := 'значение, отличное от один или два';  
END CASE;
```


CASE с перебором условий

```
CASE  
  WHEN логическое-выражение THEN  
    операторы  
  [WHEN логическое-выражение THEN операторы ...]  
  [ELSE операторы]  
END CASE;
```

Эта форма CASE реализует условное выполнение, основываясь на истинности логических условий. Каждое *логическое-выражение* в предложении WHEN вычисляется по порядку до тех пор, пока не будет найдено истинное. Затем выполняются соответствующие *операторы* и управление переходит к следующей после END CASE команде. (Все последующие выражения WHEN не проверяются)

Если ни одно из условий не окажется истинным, то выполняются *операторы* в ELSE. Но если ELSE нет, то вызывается исключение CASE_NOT_FOUND.

CASE с перебором условий. Пример

```
CASE  
  WHEN x BETWEEN 0 AND 10 THEN  
    msg := 'значение в диапазоне между 0 и 10';  
  WHEN x BETWEEN 11 AND 20 THEN  
    msg := 'значение в диапазоне между 11 и 20';  
END CASE;
```

Эта форма CASE полностью эквивалента IF-THEN-ELSIF, за исключением того, что при невыполнении всех условий и отсутствии ELSE, IF-THEN-ELSIF ничего не делает, а CASE вызывает ошибку.

CASE в режиме SQL

Если использовать язык SQL или исполняемый запрос внутри PL/pgSQL, то прямо внутри SELECT также можно использовать CASE (IF нет к сожалению), только синтаксис будет немного отличаться:

```
CREATE FUNCTION maximum(a integer, b integer) RETURNS integer AS $$  
    SELECT CASE  
        WHEN a > b  
        THEN a  
        ELSE b  
    END;  
$$ LANGUAGE SQL;
```

Практика

Итоги

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений