



PostgreSQL

# Программирование на стороне сервера



PostgreSQL

# Основные операторы PL/pgSQL



PostgreSQL

# Основные операторы PL/pgSQL

- Оператор IF
- Оператор CASE
- Оператор NULL
- Оператор PERFORM

# Условный оператор IF

- Для перехода к операторам в разделе **THEN** условие **IF (ELSIF)** должно быть истинно
  - в т.ч. не неизвестно (не **UNKNOWN**)
- В случае отсутствия разделов ELSIF и/или ELSE исключение не генерируется

```
IF условие_1 THEN
    инструкции_1;
[ELSIF условие_2 THEN
    инструкции_2;]
[ELSE
    инструкции_else;]
END IF;
```



# Пример

```
do language plpgsql
$block_name$
DECLARE
    v_salary "Production"."Products".unitprice%type ;
    v_rang char(1);
begin

    SELECT unitprice INTO STRICT v_salary
    FROM "Production"."Products"
    WHERE productid =1;

    IF v_salary > 2500::money THEN
        v_rang := 'A';
    ELSIF v_salary > 1000::money THEN
        v_rang := 'B';
    ELSE
        v_rang := 'C';
    END IF;

    RAISE NOTICE 'Цена %, ранг %', v_salary, v_rang ;
end
$block_name$;
```

Вывод

Цена \$18.00, ранг C



# Оператор CASE

- Аналогичен оператору IF, но генерирует исключение, если ни одно из условий не выполняется и отсутствует раздел ELSE

Синтаксис простого оператора

```
CASE выражение
  WHEN значение_1.1
    [, значение_1.2]... THEN
    инструкции_1;
  [WHEN значение_2.1
    [, значение_2.2] THEN
    инструкции_2;]...
  [ELSE
    инструкции_else;]
END CASE;
```

Синтаксис поискового оператора

```
CASE
  WHEN выражение_1 THEN
    инструкции_1;
  [WHEN выражение_2 THEN
    инструкции_2;]...
  [ELSE
    инструкции_else;]
END CASE;
```



# Пример

```
do language plpgsql
$block_name$
DECLARE
    v_salary numeric = 200 ;
    v_rang char(1) := 'A' ;
begin

    CASE v_rang
        WHEN 'A', 'B' THEN
            v_salary := v_salary + 500 ;
        WHEN 'C' THEN
            v_salary := v_salary + 1500 ;
        ELSE
            v_salary := v_salary + 1000 ;
    END CASE;

    RAISE notice 'Цена %, ранг %', v_salary, v_rang ;
end
$block_name$;
```

Вывод ✕

Цена 700, ранг A

# CASE: оператор PL/pgSQL и SQL-выражение

ОПЕРАТОР	ВЫРАЖЕНИЕ
Выполняет действия	Возвращает результат
Каждой паре <b>WHEN-THEN</b> может соответствовать несколько инструкций	Каждая пара <b>WHEN-THEN</b> определяет только <u>одно</u> выражение
В конце – <b>END CASE</b>	В конце - <b>END</b>
Если ни одно из условий не выполняется и отсутствует раздел <b>ELSE</b> – генерирует <u>исключение</u>	Если ни одно из условий не выполняется и отсутствует раздел <b>ELSE</b> – возвращает <u>NULL</u>
<pre>CASE rang   WHEN 'A' THEN     v_salary := v_salary + 500;     RAISE NOTICE 'Ok';   ELSE     RAISE NOTICE 'Not ok'; END CASE;</pre>	<pre>v_salary := CASE v_rang               WHEN 'A' THEN                 v_salary + 500               ELSE v_salary             END;</pre>





# Таблицы истинности

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	
TRUE	FALSE
FALSE	TRUE
NULL	NULL

# Оператор NULL

- Оператор NULL применяется в местах, где компилятор не должен ничего делать
- В PL/pgSQL этот оператор в коде допустимо опускать

```
CASE v_rang
  WHEN 'A' THEN
    v_salary := v_salary + 500;
    RAISE NOTICE 'Ok';
  ELSE
    NULL;
END CASE;
```

```
CASE v_rang
  WHEN 'A' THEN
    v_salary := v_salary + 500;
    RAISE NOTICE 'Ok';
  ELSE
    END CASE;
```



# Оператор PERFORM

- Равнозначен SELECT, но без возвращения результата
- После выполнения в специальную переменную **FOUND** логического типа записывается:
  - **TRUE** – если предыдущий запрос PERFORM вернул > 0 строк
  - **FALSE** – 0 строк

```
DO $$  
BEGIN  
    PERFORM * FROM "HR"."Employees";  
    IF FOUND THEN  
        RAISE NOTICE 'Таблица не пуста!';  
    ELSE  
        RAISE NOTICE 'В таблице пока ничего нет...';  
    END IF;  
END $$
```

