

???? ?????????????? ????????????

? ?????????? ?? PL/SQL, ??????

????? ??? ??????? ??? ?????????? ?

Postgres?

В PostgreSQL для написания процедур и функций используется процедурный язык **PL/pgSQL** (Procedural Language/PostgreSQL Structured Query Language).

???????? (Functions)

Функции в PostgreSQL могут возвращать значение (или набор строк) и могут использоваться в SQL-запросах (в секциях `SELECT`, `WHERE`, `JOIN`).

Синтаксис создания простой функции:

SQL

```
CREATE OR REPLACE FUNCTION calculate_bonus(emp_id INT, bonus_percentage NUMERIC)
RETURNS NUMERIC AS $$
DECLARE
    current_salary NUMERIC;
    calculated_bonus NUMERIC;
BEGIN
    -- Получаем текущую зарплату сотрудника
    SELECT salary INTO current_salary
    FROM employees
    WHERE id = emp_id;

    -- Если сотрудник не найден, возвращаем 0
    IF current_salary IS NULL THEN
        RETURN 0;
    END IF;

    calculated_bonus := current_salary * (bonus_percentage / 100);
    RETURN calculated_bonus;
```

```
END;  
$$ LANGUAGE plpgsql;
```

Вызов функции:

SQL

```
SELECT calculate_bonus(5, 10.5);
```

???????? (Procedures)

Процедуры появились начиная с PostgreSQL 11. В отличие от функций, они **не возвращают значение** через `RETURN` (но могут возвращать данные через параметры `INOUT` или `OUT`), однако они **поддерживают транзакции** (`COMMIT`, `ROLLBACK`), в то время как функции — нет.

Синтаксис создания процедуры:

SQL

```
CREATE OR REPLACE PROCEDURE transfer_funds(  
    sender_account INT,  
    receiver_account INT,  
    amount NUMERIC  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    -- Снимаем средства с одного счета  
    UPDATE accounts  
    SET balance = balance - amount  
    WHERE id = sender_account;  
  
    -- Зачисляем средства на другой счет  
    UPDATE accounts  
    SET balance = balance + amount  
    WHERE id = receiver_account;  
  
    -- Фиксируем транзакцию  
    COMMIT;  
END;  
$;
```

Вызов процедуры (используется команда `CALL`):

SQL



```
CALL transfer_funds(1, 2, 500.00);
```

????????? ?????????????? ?????????????? PL/pgSQL

- **Условия (IF / THEN / ELSE):**

SQL

```
IF v_count > 10 THEN
    -- код
ELSIF v_count > 0 THEN
    -- код
ELSE
    -- код
END IF;
```

- **Циклы (LOOP, WHILE, FOR):**

SQL

```
-- Цикл с условием
WHILE v_counter < 10 LOOP
    v_counter := v_counter + 1;
END LOOP;

-- Обход результатов запроса
FOR r_row IN SELECT * FROM employees LOOP
    RAISE NOTICE 'Employee: %', r_row.name;
END LOOP;
```

- **Обработка ошибок (EXCEPTION):**

SQL

```
BEGIN
    -- рискованный код
EXCEPTION
    WHEN division_by_zero THEN
```

```
RAISE NOTICE 'Деление на ноль!';  
WHEN OTHERS THEN  
    RAISE NOTICE 'Произошла непредвиденная ошибка: %', SQLERRM;  
END;
```

Главное отличие при выборе между ними: если вам нужно делать выборку данных внутри `SELECT` — пишите **функцию**. Если вам нужно выполнить набор команд модификации данных с управлением транзакциями — используйте **процедуру**.

Какой именно сценарий или задачу вы хотите реализовать с помощью PL/pgSQL? Могу написать конкретный пример под ваши требования.

??? ?????????????? ?????? PL/SQL ? Oracle ? Postgres?

Сравнение процедурных расширений Oracle (**PL/SQL**) и PostgreSQL (**PL/pgSQL**) сводится к сопоставлению зрелого коммерческого стандарта и мощного открытого аналога. Несмотря на внешнюю схожесть (оба языка вдохновлены Ada и используют блоки `DECLARE`, `BEGIN`, `EXCEPTION`), между ними есть глубокие архитектурные и синтаксические различия.

????????? ??????????

Характеристика	Oracle (PL/SQL)	PostgreSQL (PL/pgSQL)
Интеграция с СУБД	Язык является родным и доминирующим для СУБД. На нем написаны даже многие внутренние системные пакеты.	Язык — один из множества поддерживаемых (<code>plpython</code> , <code>plperl</code> , <code>pltcl</code>), хотя и самый популярный для написания логики в БД.
Пакеты (Packages)	Есть . Позволяют группировать типы, функции и процедуры, а также имеют состояние сессии (общие переменные внутри пакета).	Нет в чистом виде (хотя можно эмулировать через схемы). Переменные уровня сессии требуют сторонних расширений или таблиц.
Типы данных и коллекции	Огромный выбор встроенных сложных типов: <code>TABLE</code> (ассоциативные массивы), <code>VARRAY</code> , <code>NESTED TABLE</code> , продвинутая работа с записями (<code>ROWTYPE</code>).	Коллекции ограничены базовыми массивами и таблицами типов (строки таблиц, <code>RECORD</code>), работа с ними менее гибкая.

Характеристика	Oracle (PL/SQL)	PostgreSQL (PL/pgSQL)
Обработка SQL (курсоры)	Поддерживает неявные курсоры с мощными атрибутами (<code>%FOUND</code> , <code>%NOTFOUND</code> , <code>%ROWCOUNT</code>), а также продвинутые конструкции <code>BULK COLLECT</code> и <code>FORALL</code> для массовой обработки (оптимизация контекст-переключений).	Курсоры есть (<code>FOR</code> циклы по выборкам), но массовые операции (<code>BULK</code>) реализуются иначе, часто через конструкции вроде <code>ANY</code> или временные таблицы.
Управление транзакциями	Поддерживает автономные транзакции (<code>PRAGMA AUTONOMOUS_TRANSACTION</code>), позволяя фиксировать изменения в логах независимо от основной транзакции.	Обычные функции не могут управлять транзакциями. Процедуры могут выполнять <code>COMMIT</code> или <code>ROLLBACK</code> , но автономных транзакций «из коробки» до сих пор нет (требуется костыли через внешние вызовы или <code>dblink</code>).
Обработка ошибок	Большой встроенный набор predefined исключений (<code>NO_DATA_FOUND</code> , <code>TOO_MANY_ROWS</code> и др.).	Исключения обрабатываются через блок <code>EXCEPTION</code> , но кодов ошибок и имен меньше, часто приходится проверять коды ошибок через <code>SQLSTATE</code> .
Объектно-ориентированные возможности	Поддерживает ООП-концепции: типы объектов с методами, наследование объектов в базе данных.	Отсутствуют ООП-возможности на уровне языка процедур.

??????? ?????????????? ???????

1. Объявление переменных и присваивание:

- **Oracle:** Часто используется оператор присваивания `:=`, но также активно применяется `DEFAULT` или прямая инициализация. Широко развита работа с атрибутами `%TYPE` и `%ROWTYPE`.
- **PostgreSQL:** Синтаксис очень похож (`:=`), но синтаксис функций требует оборачивания тела в текстовые кавычки (`$$ ... $$`) и явного указания языка `LANGUAGE plpgsql`.

2. Функция выборки одной строки (когда строка не найдена):

- **Oracle:** Если запрос в блоке PL/SQL возвращает 0 строк, автоматически возбуждается исключение `NO_DATA_FOUND`.

- **PostgreSQL:** При использовании `SELECT ... INTO` в PL/pgSQL переменные просто становятся `NULL`, исключение не генерируется (если не настроено строгое поведение).

3. Создание пакетов (отсутствие в Postgres):

- **Oracle:**

SQL

```
CREATE OR REPLACE PACKAGE my_pack AS
    procedure do_something;
END my_pack;
```

- **PostgreSQL:** Пакеты создать нельзя. Функции и процедуры создаются по отдельности, а для изоляции и структурирования кода используются разные **схемы** (schemas).

Какой конкретный паттерн из Oracle вы пытаетесь перенести в PostgreSQL? Возможно, для него есть стандартный и элегантный обходной путь.

????? ??? ????? ????? ?????????????????? ?
postgres?

PostgreSQL обладает уникальной архитектурой расширяемости (благодаря системе обработчиков языков — *Language Handlers*), которая позволяет писать функции, триггеры и процедуры не только на PL/pgSQL, но и на множестве других языков программирования.

Все языки делятся на две категории: входящие в стандартный комплект и сторонние расширения.

1. ????? ?? ?????????????? ??????????????

Эти языки поставляются вместе с базовым сервером PostgreSQL, их нужно только активировать в конкретной базе данных командой `CREATE EXTENSION extension_name;` (например, `CREATE EXTENSION plpython3u;`).

- **PL/Python (`plpython3u`)**

Один из самых популярных альтернативных языков. Позволяет использовать всю мощь Python для работы с данными, математикой, парсинга или вызова внешних API прямо из базы.

SQL

```
CREATE FUNCTION get_max(a int, b int) RETURNS int AS $$  
    return max(a, b)  
$$ LANGUAGE plpython3u;
```

- **PL/Perl (plperl)** Исторически один из первых языков в Postgres. Удобен для сложных манипуляций со строками, регулярных выражений и задач системного администрирования.
- **PL/Tcl (pltcl)** Язык Tcl (Tool Command Language), исторически встроенный в экосистему PostgreSQL. Реже используется в новых проектах, но поддерживается «из коробки».

2. ?????????? ? ????????????? ?????? (?????????????)

Эти языки разрабатываются сторонним сообществом, устанавливаются отдельно, но глубоко интегрируются в СУБД:

- **PL/v8 (JavaScript / V8)**

Позволяет писать функции на **JavaScript**, используя движок V8 от Google. Идеально подходит, если нужно обрабатывать JSON-документы (которых в Postgres множество) с помощью привычного JS-кода или переиспользовать бизнес-логику из Node.js/фронтенда на стороне базы данных.

- **PL/Java**

Дает возможность писать код на **Java**, используя стандартные интерфейсы JDBC. Востребовано в энтерпрайз-среде, где вся логика монолита завязана на Java.

- **PL/R**

Интеграция со статистическим языком **R**. Позволяет выполнять сложные математические вычисления, статистический анализ и строить модели машинного обучения прямо внутри SQL-запросов.

- **PL/Lua**

Сверхбыстрый и легкий скриптовый язык **Lua**, встроенный в PostgreSQL. Часто применяется там, где критична максимальная производительность скриптов в базе.

- **PL/sh**

Позволяет выполнять **системные shell-скрипты** (Bash и др.) прямо из базы данных (например, запускать утилиты ОС по триггеру). Это крайне опасный с точки зрения безопасности язык, поэтому он используется только в изолированных контурах.

????? ? ??? ????????????????

- Для стандартной бизнес-логики, CRUD-операций и высокой скорости работы без лишних зависимостей выбирайте **PL/pgSQL**.
- Для работы с нейросетями, сложной аналитики, парсинга или интеграции с внешними библиотеками экосистемы Python незаменим **PL/Python**.
- Для интенсивной работы с JSON и быстрой обработки данных через знакомый синтаксис — **PL/v8 (JS)**.

Revision #2

Created 2026-09-28 03:48:33 UTC by Nikolay

Updated 2026-09-28 03:54:39 UTC by Nikolay