

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

Postgres ? C#

- [Как вызываются процедуры и функции на PL/Python из C#?](#)
- [Как передавать и парсить JSON/JSONB из C# в PL/Python функцию через Npgsql?](#)

??? ?????????? ?????????? ? ????????? ?? PL/Python ?? C#?

Вызов процедур и функций PostgreSQL из C# (начиная с C# 7 и выше) выполняется через стандартную библиотеку **Npgsql** (официальный ADO.NET провайдер для PostgreSQL).

С точки зрения приложения C# **абсолютно не имеет значения, на чем написана функция/процедура** (на PL/pgSQL, PL/Python, PL/v8 или C). СУБД предоставляет единый интерфейс вызова.

????? ????????, ?????????????? ?????????? ??? ?????? ??????

Для вызова функций рекомендуется использовать обычные SQL-запросы `SELECT * FROM function_name(@param)`. Это универсальный подход, работающий как для скалярных значений, так и для табличных результатов (`RETURNS TABLE`).

?????? 1: ?????? ????????, ?????????????? ?????? ?????? (`RETURNS TABLE`)

Предположим, у нас есть функция PL/Python из предыдущего примера:

SQL



```
CREATE OR REPLACE FUNCTION generate_user_report(min_age int)
RETURNS TABLE(user_id int, full_name text, status text) AS $$
    # PL/Python логика
$$ LANGUAGE plpython3u;
```

Код C# для вызова функции и чтения данных:

```
using System;
using System.Threading.Tasks;
using Npgsql;

public class PostgresExample
{
```

```

private const string ConnectionString =
"Host=localhost;Database=mydb;Username=postgres;Password=secret";

public async Task ReadUserReportAsync(int minAge)
{
    await using var conn = new NpgsqlConnection(ConnectionString);
    await conn.OpenAsync();

    // Обращаемся к функции как к обычной таблице
    const string sql = "SELECT user_id, full_name, status FROM
generate_user_report(@minAge)";

    await using var cmd = new NpgsqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("minAge", minAge);

    await using var reader = await cmd.ExecuteReaderAsync();

    while (await reader.ReadAsync())
    {
        // Используем кортежи (Tuples), доступные начиная с C# 7
        var (userId, fullName, status) = (
            reader.GetInt32(0),
            reader.GetString(1),
            reader.GetString(2)
        );

        Console.WriteLine($"ID: {userId}, Имя: {fullName}, Статус: {status}");
    }
}
}

```

?????? 2: ?????? ?????????? ?????????? (????????????? ?????? ??????????)

C#
■ ■

```

public async Task<decimal> GetCartTotalAsync(string cartJson)
{
    await using var conn = new NpgsqlConnection(ConnectionString);
    await conn.OpenAsync();

```

```

const string sql = "SELECT calculate_cart_total(@cartJson)";

await using var cmd = new NpgsqlCommand(sql, conn);
cmd.Parameters.AddWithValue("cartJson", cartJson);

// ExecuteScalarAsync возвращает первое значение первой строки
var result = await cmd.ExecuteScalarAsync();
return Convert.ToDecimal(result);
}

```

????? ?????????? (Procedure)

Процедуры в PostgreSQL вызывают с помощью команды `CALL`. В Npgsql это можно сделать двумя способами: через явный `CALL` или через `CommandType.StoredProcedure`.

??????? ???????? ???????????? ?????? `CALL`

Предположим, у нас есть процедура PL/Python:

SQL

■ ■

```

CREATE OR REPLACE PROCEDURE cleanup_old_logs(days_old int) AS $$
    # PL/Python логика
$$ LANGUAGE plpython3u;

```

Код C#:

```

public async Task CleanupLogsAsync(int daysOld)
{
    await using var conn = new NpgsqlConnection(ConnectionString);
    await conn.OpenAsync();

    // Явный вызов процедуры
    const string sql = "CALL cleanup_old_logs(@daysOld)";

    await using var cmd = new NpgsqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("daysOld", daysOld);
}

```

```
// Используем ExecuteNonQueryAsync, так как процедура не возвращает табличный результат
await cmd.ExecuteNonQuery();
}
```

??????????? ?????? ? ?????????? (ARRAY) ?? C# 7

Если ваша функция PL/Python принимает или возвращает массив PostgreSQL (например, `integer[]` или `text[]`), Npgsql автоматически сопоставляет их с массивами C# (`int[]`, `string[]`).

Пример:

```
public async Task ProcessIdsAsync(int[] ids)
{
    await using var conn = new NpgsqlConnection(ConnectionString);
    await conn.OpenAsync();

    const string sql = "SELECT process_array_function(@ids)";

    await using var cmd = new NpgsqlCommand(sql, conn);
    // Npgsql автоматически приведёт int[] C# к integer[] в Postgres
    cmd.Parameters.AddWithValue("ids", ids);

    await cmd.ExecuteNonQuery();
}
```

??? ?????????? ? ?????????? JSON/JSONB ?? C# ? PL/Python ?????????? ?????? Npgsql?

Передача данных в формате **JSON / JSONB** из C# в PL/Python — один из наиболее эффективных подходов: C# сериализует объект в JSON, Npgsql передает его в PostgreSQL как native-тип `jsonb`, а PL/Python автоматически или с помощью стандартного модуля `json` превращает его в родной словарь (`dict`) или список (`list`).

1. ?????????? ?????????? ? PostgreSQL (PL/Python)

Функция принимает тип `jsonb` (или `json`). В PL/Python аргументы типа `json/jsonb` передаются в виде обычных **строк**, поэтому внутри их нужно распарсить с помощью стандартного модуля `json`.

SQL

```
CREATE OR REPLACE FUNCTION process_user_data(payload jsonb)
RETURNS TABLE(user_id int, status text, is_valid boolean) AS $$
    import json

    # PL/Python получает json/jsonb как строку
    data = json.loads(payload)

    users = data.get("users", [])
    result = []

    for user in users:
        u_id = user.get("id")
        email = user.get("email", "")

        # Простая проверка/обработка
        is_valid = "@" in email
        status = "Active" if user.get("age", 0) >= 18 else "Pending"
```

```

        result.append({
            "user_id": u_id,
            "status": status,
            "is_valid": is_valid
        })

    return result

$$ LANGUAGE plpython3u;

```

2. ?????? ?? C# ? ????????? Npgsql ? System.Text.Json

В современных версиях Npgsql передача JSON выполняется через встроенную поддержку `System.Text.Json` или явное указание типа `NpgsqlDbType.Jsonb`.

???????? 1: ??????????????? `System.Text.Json` (?????????????????)

C#
■ ■

```

using System;
using System.Collections.Generic;
using System.Text.Json;
using System.Threading.Tasks;
using Npgsql;
using NpgsqlTypes;

public class JsonPostgresExample
{
    private const string ConnectionString =
        "Host=localhost;Database=mydb;Username=postgres;Password=secret";

    public async Task ProcessDataAsync()
    {
        // 1. Формируем C# объект для передачи
        var requestData = new
        {
            users = new[]
            {
                new { id = 101, email = "alex@example.com", age = 25 },
                new { id = 102, email = "invalid_email", age = 16 }
            }
        }
    }
}

```

```

};

// 2. Сериализуем объект в строку JSON
string jsonPayload = JsonSerializer.Serialize(requestData);

await using var conn = new NpgsqlConnection(ConnectionString);
await conn.OpenAsync();

const string sql = "SELECT user_id, status, is_valid FROM
process_user_data(@payload)";

await using var cmd = new NpgsqlCommand(sql, conn);

// 3. Добавляем параметр с явным указанием NpgsqlDbType.Jsonb
cmd.Parameters.Add(new NpgsqlParameter("payload", NpgsqlDbType.Jsonb)
{
    Value = jsonPayload
});

// 4. Читаем результат
await using var reader = await cmd.ExecuteReaderAsync();
while (await reader.ReadAsync())
{
    var userId = reader.GetInt32(0);
    var status = reader.GetString(1);
    var isValid = reader.GetBoolean(2);

    Console.WriteLine($"ID: {userId} | Статус: {status} | Валиден: {isValid}");
}
}
}

```

3. ?????????? ??????????: ?????????? JSONB ?? PL/Python ? C#

Если функция PL/Python должна вернуть сложный JSON-объект обратно в C#:

????????? ? PostgreSQL:

SQL
■ ■

```

CREATE OR REPLACE FUNCTION get_system_stats()
RETURNS jsonb AS $$
    import json

    stats = {
        "server_time": "2026-09-12",
        "metrics": {
            "cpu_usage": 42.5,
            "active_connections": 12
        },
        "tags": ["prod", "primary"]
    }

    # Возвращаем JSON-строку
    return json.dumps(stats)
$$ LANGUAGE plpython3u;

```

??? ?????? ? C#:

C#
■ ■

```

public async Task GetStatsAsync()
{
    await using var conn = new NpgsqlConnection(ConnectionString);
    await conn.OpenAsync();

    const string sql = "SELECT get_system_stats()";
    await using var cmd = new NpgsqlCommand(sql, conn);

    var rawJson = (string)await cmd.ExecuteScalarAsync();

    // Десериализуем JSON в C# объект
    using var doc = JsonDocument.Parse(rawJson);
    var cpu = doc.RootElement
        .GetProperty("metrics")
        .GetProperty("cpu_usage")
        .GetDouble();

    Console.WriteLine($"CPU Usage: {cpu}%");
}

```

????????? ???????? ??? ?????? ? JSON/JSONB:

1. **Тип параметра в С#:** При передаче строкового JSON всегда указывайте `NpgsqlDbType.Jsonb` (или `NpgsqlDbType.Json`), иначе Npgsql передаст строку как обычный `text`, и Postgres выдаст ошибку несоответствия типов.
2. **Сериализация внутри PL/Python:**
 - При **приеме** `jsonb` в Python аргумент является строкой `str` — применяйте `json.loads(arg)`.
 - При **возврате** `jsonb` отправляйте строку — применяйте `json.dumps(obj)`.