

??? ?????????? ?????????? ? ????????? ?? 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();
}

```

Revision #1

Created 2026-09-28 06:16:27 UTC by Nikolay

Updated 2026-09-28 06:18:52 UTC by Nikolay