



Программирование серверных
процедур на языках PL/Perl,
PL/Python, PL/v8

Иван Панченко

Что есть?

В коробке

- ◆ PL/PgSQL
- ◆ PL/Perl
- ◆ PL/Python
- ◆ PL/Tcl



Что есть?

Вне коробки

- ◆ PL/v8
- ◆ PL/Java
- ◆ PL/Lua
- ◆ PL/R
- ◆ PL/bash
- ◆ PL/php
- ◆



Просто ли создать новый PL/@#\$?

CREATE [OR REPLACE] [TRUSTED] [PROCEDURAL]

LANGUAGE имя

HANDLER обработчик_вызова_функции

[INLINE обработчик_инлайн_блока]

[VALIDATOR функция_проверки_кода_функции]

◆ Эти функции можно писать только на C

<https://postgrespro.ru/docs/postgrespro/10/plhandler>

<https://postgrespro.ru/docs/postgrespro/9.6/sql-createlanguage>

Почему пишут на PL/PgSQL

Потому же, почему на SQL:

- ◆ Почти декларативный
- ◆ Весьма ограниченный
- ◆ Известный тем же людям

Какой самый быстрый процедурный язык?

- ◆ C
- ◆ SQL

Доверять или не доверять?

TRUSTED	UNTRUSTED
<p>Любой юзер может создать функцию. Нельзя работать с I/O, в т.ч. с сетью</p>	<p>Создает функцию только суперьюзер. Можно всё, что можно.</p>
<p>plpgsql plperl plv8</p>	<p>plperlu plpython2u plpython3u</p>

- ◆ Интерпретатор создается при первом использовании
- ◆ PL/PerlU и PL/Perl – разные экземпляры интерпретатора
- ◆ Параметры:

```
plperl.on_init = "use Data::Dumper; ....."
```

```
plperl.on_plperl_init = "....."
```

```
plperl.on_plperlu_init = "....."
```

```
plperl.use_strict = true;
```

Особенности PL/Python

- ◆ Интерпретатор создается при первом использовании
- ◆ PL/Python2 и PL/Python3 вместе использовать нельзя
- ◆ Указать, что делать при инициализации, нельзя
- ◆ Однострочники делать неудобно
- ◆ SD – статический словарь; GD - глобальный словарь

Особенности PL/v8

- ◆ Автоматический маппинг JSON (JSONB)
- ◆ Возможность определять window functions
- ◆ Возможность делать подтранзакции
- ◆ Упрощенный вызов других функций PL/v8
- ◆ `plv8.start_proc=my_start_func`

(имя PLv8-функции, производящей инициализацию)

Подробнее

см <https://rymc.io/2016/03/22/a-deep-dive-into-plv8/>

Установка PL/Perl, PL/Python

Из исходников:

```
./configure --with-perl --with-python  
[PYTHON=/usr/bin/python3]
```

Из пакетов:

```
sudo apt install postgresql-plpython3-10
```

Потом:

```
CREATE LANGUAGE plperl;
```

```
CREATE LANGUAGE plperlu;
```

```
CREATE LANGUAGE plpythonu;
```

```
CREATE LANGUAGE plpython3u;
```

Вариант 1) . Из пакета

```
sudo apt install postgresql-9.5-plv8
```

Вариант 2) . Собрать

```
wget https://github.com/plv8/plv8/archive/v2.1.0.tar.gz
```

```
apt install curl python; tar xzf v2.1.0.tar.gz
```

```
cd plv8-v2.1.0 && \
```

```
    make PG_CONFIG=/usr/local/pgsql/bin/pg_config static -j 4 &&\
```

```
    make PG_CONFIG=/usr/local/pgsql/bin/pg_config install
```

Неприятности: i386; static; 2 Gb !

```
CREATE EXTENSION plv8;
```

Hello world PL/Perl

```
DO $$  
    elog(NOTICE, "Hello World");  
$$ LANGUAGE plperl;
```

NOTICE: Hello World

DO

Hello world PL/Python

```
DO $$  
    plpy.notice('Hello World', hint="Будь здоров",  
                detail="В деталях");  
$$ LANGUAGE plpythonu;
```

NOTICE: Hello World

DETAIL: В деталях

HINT: Будь здоров

DO

Hello world PL/v8

```
DO $$  
    plv8.elog(NOTICE, 'Hello World');  
$$ LANGUAGE plv8;
```

NOTICE: Hello World

DO

```
select count(*) from pg_class;
```



```
DO LANGUAGE plpgsql $$ DECLARE a int;
```

```
BEGIN SELECT count(*) INTO a FROM pg_class; END; $$; DO
```



```
DO LANGUAGE plperl $$
```

```
spi_exec_query('select count(*) from pg_class'); $$;
```



```
select count(*) from pg_class;
```

◆ Time: 0.375 ms

```
DO LANGUAGE plpgsql $$ DECLARE a int;
```

```
BEGIN SELECT count(*) INTO a FROM pg_class; END; $$; DO
```

◆ Time: 0.615 ms

```
DO LANGUAGE plperl $$
```

```
spi_exec_query('select count(*) from pg_class'); $$;
```

◆ Time: 0.638 ms


```
DO LANGUAGE plpythonu $$
```

```
    plpy.execute('select count(*) from pg_class'); $$;
```



```
DO LANGUAGE plv8 $$
```

```
    plv8.execute('select count(*) from pg_class'); $$;
```



```
DO LANGUAGE plpythonu $$
```

```
    plpy.execute('select count(*) from pg_class'); $$;
```

◆ Time: 0.632 ms

```
DO LANGUAGE plv8 $$
```

```
    plv8.execute('select count(*) from pg_class'); $$;
```

◆

```
DO LANGUAGE plpythonu $$
```

```
    plpy.execute('select count(*) from pg_class'); $$;
```

◆ Time: 0.632 ms

```
DO LANGUAGE plv8 $$
```

```
    plv8.execute('select count(*) from pg_class'); $$;
```

◆ Time: 0.586 ms

```
DO LANGUAGE plpythonu $$
```

```
    plpy.execute('select count(*) from pg_class'); $$;
```

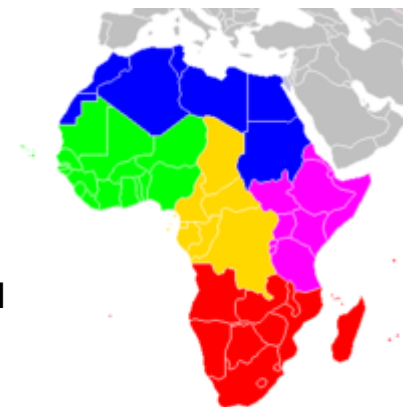
◆ Time: 0.632 ms

```
DO LANGUAGE plv8 $$
```

```
    plv8.execute('select count(*) from pg_class'); $$;
```

◆ Time: 0.586 ms

На самом деле, это просто ноль. Поэтому значения столь близки



Производительность?

```
DO LANGUAGE plpgsql $$  
  
DECLARE a int; i int;  
  
    BEGIN  FOR i IN 0..1000 LOOP  
        select count(*) INTO a FROM pg_class;  
  
    END LOOP;  
  
END; $$;
```

◆ 129.748 ms

```
DO LANGUAGE plperl $$ for(0..1000) {  
    spi_exec_query('select count(*) from pg_class');  
}  
  
$$;
```

◆ Time: 122.381 ms

Производительность?

```
DO LANGUAGE plpythonu $$  
  for i in range (0,1000) :  
    plpy.execute('select count(*) from pg_class');  
$$;
```

◆ Time: 119.615 ms

```
DO LANGUAGE plv8 $$  
  for(var i=0;i<=1000;i++)  
    plv8.execute('select count(*) from pg_class');  
$$;
```

◆ Time: 129.748 ms

Производительность!

```
DO LANGUAGE plperl $$  
  
my $h=spi_prepare('select count(*) from pg_class');  
  
for(0..1000) { spi_exec_prepared($h); }  
  
    spi_freepplan($h); $$;
```



```
DO LANGUAGE plpythonu $$  
  
h = plpy.prepare('select count(*) from pg_class');  
  
for i in range (0,1000) :  
  
    plpy.execute(h);  
  
$$;
```

Производительность!

```
DO LANGUAGE plperl $$  
my $h=spi_prepare('select count(*) from pg_class');  
for(0..1000) { spi_exec_prepared($h); }  
  
    spi_freepplan($h); $$;
```

◆ Time: 75.746 ms

```
DO LANGUAGE plpythonu $$  
h = plpy.prepare('select count(*) from pg_class');  
for i in range (0,1000) :  
    plpy.execute(h);  
  
$$;
```

◆ Time: 76.462 ms

Производительность!

```
DO LANGUAGE plv8 $$  
    var h=plv8.prepare('select count(*) from pg_class');  
    for(var i=0;i<=1000;i++) h.execute();  
$$;
```



Производительность!

```
DO LANGUAGE plv8 $$  
    var h=plv8.prepare('select count(*) from pg_class');  
    for(var i=0;i<=1000;i++) h.execute();  
$$;
```

◆ Time: 82.110 ms

Вычисления?

```
DO LANGUAGE plpgsql $$ DECLARE a bigint;
BEGIN a=0;
      FOR i IN 0..1000000 LOOP a=a+i*i::bigint;
END LOOP; END; $$;
```

◆ Time: 319.937 ms

```
DO LANGUAGE plperl $$
      my $a=0; for my $i (0..1000000) {
          $a+=$i*$i; }
      $$;
```



Вычисления?

```
DO LANGUAGE plpgsql $$ DECLARE a bigint;
BEGIN a=0;
      FOR i IN 0..1000000 LOOP a=a+i*i::bigint;
END LOOP; END; $$;
```

◆ Time: 319.937 ms

```
DO LANGUAGE plperl $$
      my $a=0; for my $i (0..1000000) {
          $a+=$i*$i; }
      $$;
```

◆ Time: 108.669 ms

```
DO LANGUAGE plpythonu $$  
  a = 0;  
  for i in range (0,1000000) :  
    a = a + i * i;  
  
$$;
```

◆ Time: 98.586 ms

Вычисления?

```
DO LANGUAGE plv8 $$ var a=0;  
    for (var i=0; i<=1000000; i++) a+=i*i ;  
$$;
```



Вычисления?

```
DO LANGUAGE plv8 $$ var a=0;  
    for (var i=0; i<=1000000; i++) a+=i*i ;  
$$;
```

◆ Time: 5.511 ms

Не забываем о точности ?

```
DO LANGUAGE plv8 $$ plv8.eelog(WARNING,  
  parseInt(33333383333312755033)) $$;
```

```
WARNING: 33333383333312754000
```

- ◆ В Javascript целое представляется в форме float.
- ◆ Поэтому в предыдущих примерах результат получается быстро,
- ◆ но не точно:

333333833333127550

вместо 333333833333500000

$$\Sigma = n*(n+1)*(2n+1)/6$$

Вызовы функций?

```
CREATE OR REPLACE FUNCTION sq(i bigint) RETURNS bigint
  LANGUAGE plpgsql as $$
  BEGIN RETURN i^2;
  END $$;
```

```
DO LANGUAGE plpgsql $$ DECLARE a bigint;
BEGIN a=0; FOR i IN 0..1000000 LOOP a=a+sq(i);
END LOOP; END; $$;
```

◆ Time: 1349.423 ms

```
DO LANGUAGE plperl $$ sub sq { return $_[0]* $_[0]; };
  my $a=0; for my $i (0..1000000) { $a+=sq($i); } $$;
```

◆ Time: 253.281 ms

Вызовы функций?

```
DO LANGUAGE plpythonu $$  
  
def sq(i) :  
    return i*i  
  
a=0  
  
for i in range (0,1000000) :  
    a = a + sq(i)  
  
$$;
```

◆ Time: 253.281 ms

Вызовы функций?

```
DO LANGUAGE plv8 $$  
var a=0; function sq(i) { return i*i; }  
for(var i=0;i<=1000000;i++) a+=sq(i) ;  
$$;
```

- ◆ Time: 8.543 ms
- ◆ Результат неточный!

◆ Хорошо:

```
CREATE OR REPLACE function cr() RETURNS int LANGUAGE
plperl AS $$
return spi_exec_query('select count(*) from pg_class')
    ->{rows}->[0]->{count};
$$;
```

◆ Плохо:

```
CREATE OR REPLACE function cr2() RETURNS int
LANGUAGE plperl as $$
    my $h = spi_prepare('select count(*) from pg_class');
    return spi_exec_prepared($h)->{rows}->[0]->{count};
$$;
```

◆ Хорошо:

```
spi_freeplan($h)
```

◆ Хорошо:

```
CREATE OR REPLACE function cr3() RETURNS int LANGUAGE
plpythonu as $$
return plpy.execute('select count(*) from
pg_class')[0]['count']$$;
```

◆ Хорошо:

```
h = plpy.prepare('select count(*) from pg_class')
return plpy.execute(h)[0]['count']
```

◆ Хорошо:

```
CREATE OR REPLACE FUNCTION crq() RETURNS int LANGUAGE plv8
    return plv8.execute('select count(*) from
pg_class')[0].count;
$$;
```

◆ Плохо:

```
h = plv8.prepare('select count(*) from pg_class');
return h.execute()[0].count;
```

◆ Хорошо:

```
h.free();
```

◆ В каком виде они попадают в Perl?

```
CREATE OR REPLACE FUNCTION crq(a int, b bytea,  
    c int[], d json ) RETURNS void LANGUAGE plperl AS  
$$ warn Dumper(@_) $$;
```

◆ Результат

```
$VAR1 = '1';  
$VAR2 = '\\x123456';  
$VAR3 = bless( {'array' => [ '1\\', '2', '3'],  
                'typeoid' => 1007  
                }, 'PostgreSQL::InServer::ARRAY' );  
$VAR4 = '{"a":"b"}';
```


Как же быть с JSON'ом в PL/Perl?

```
my $json = JSON::XS::decode_json(Encode::encode_utf8($text));
```

```
my $ret = Encode::encode_json(JSON::XS::encode_json($json));
```

неудобно и затратно

◆ В каком виде они попадают в Python?

```
CREATE OR REPLACE FUNCTION pdump(a int, b bytea,  
    c int[], d json ) RETURNS void LANGUAGE plpythonu AS  
  
$$ plpy.warning(a,b,c,d) $$;
```

◆ Результат

```
(1, '\x124V', # это лучше чем в PL/Perl  
[1, 2, 3],    # это тоже  
'{"a":"b"}')
```

◆ Начиная с Pg 10 : Поддержка многомерных массивов
(в Perl были и раньше)

```
[[1,2], [3,4]]
```

Параметры: TRANSFORM

- ◆ В PostgreSQL 9.5 появилось преобразование типов параметров

```
CREATE TRANSFORM FOR store LANGUAGE plperl (  
    FROM SQL WITH FUNCTION hstore_to_plperl(internal),  
    TO SQL WITH FUNCTION plperl_to_hstore(internal)  
);
```

- ◆ Пока есть только для hstore (plperl / plpython) и ltree (plpython)
- ◆ В Postgres Professional сделали TRANSFORM для JSONB
- ◆ Будет в PostgreSQL 11

<https://commitfest.postgresql.org/16/1335/>

<https://commitfest.postgresql.org/16/1400/>

Параметры в PL/v8

- ◆ В каком виде они попадают в Javascript?

```
CREATE OR REPLACE FUNCTION jdump(a int, b bytea,  
    c int[], d json ) RETURNS void LANGUAGE plv8 AS  
  
$$ plv8.elog(WARNING, a, b, c, d) $$;
```

- ◆ Результат

```
1  
18,52,86 // массив байтов  
1,2,3    // массив чисел  
[object Object] // JSON !! супер
```



Спасибо за внимание!

Контакты:

info@postgrespro.ru

<https://www.postgrespro.ru/>