
Артур Закиров

# Использование pg_variables в качестве временных таблиц

# Что такое pg_variables

- Расширение для PostgreSQL:

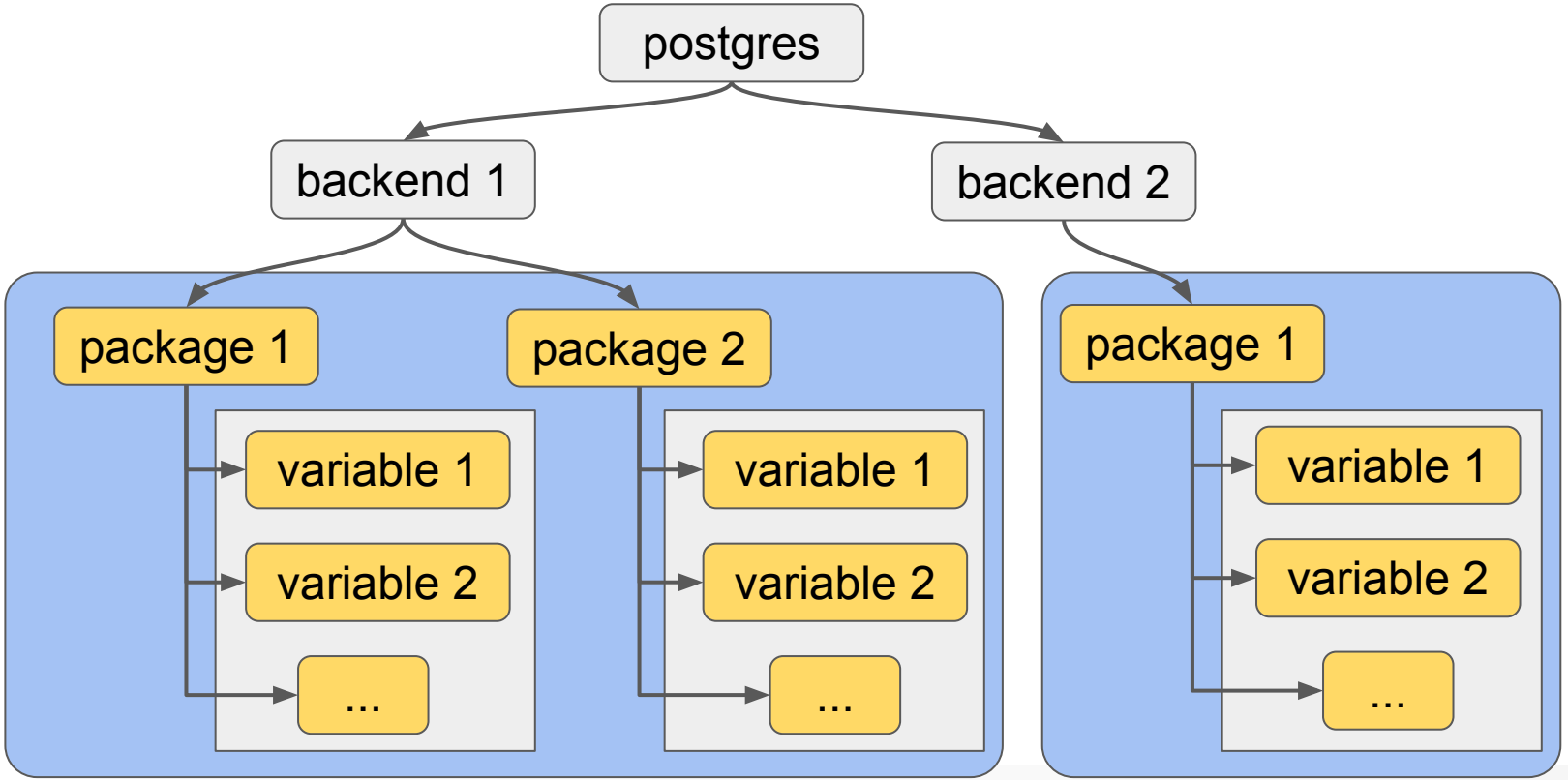
https://github.com/postgrespro/pg_variables

- Создание и использование **сессионных** переменных
- Переменные внутри сессии глобальные
- В 2018 году была добавлена поддержка транзакций
- Переменные по умолчанию не транзакционные

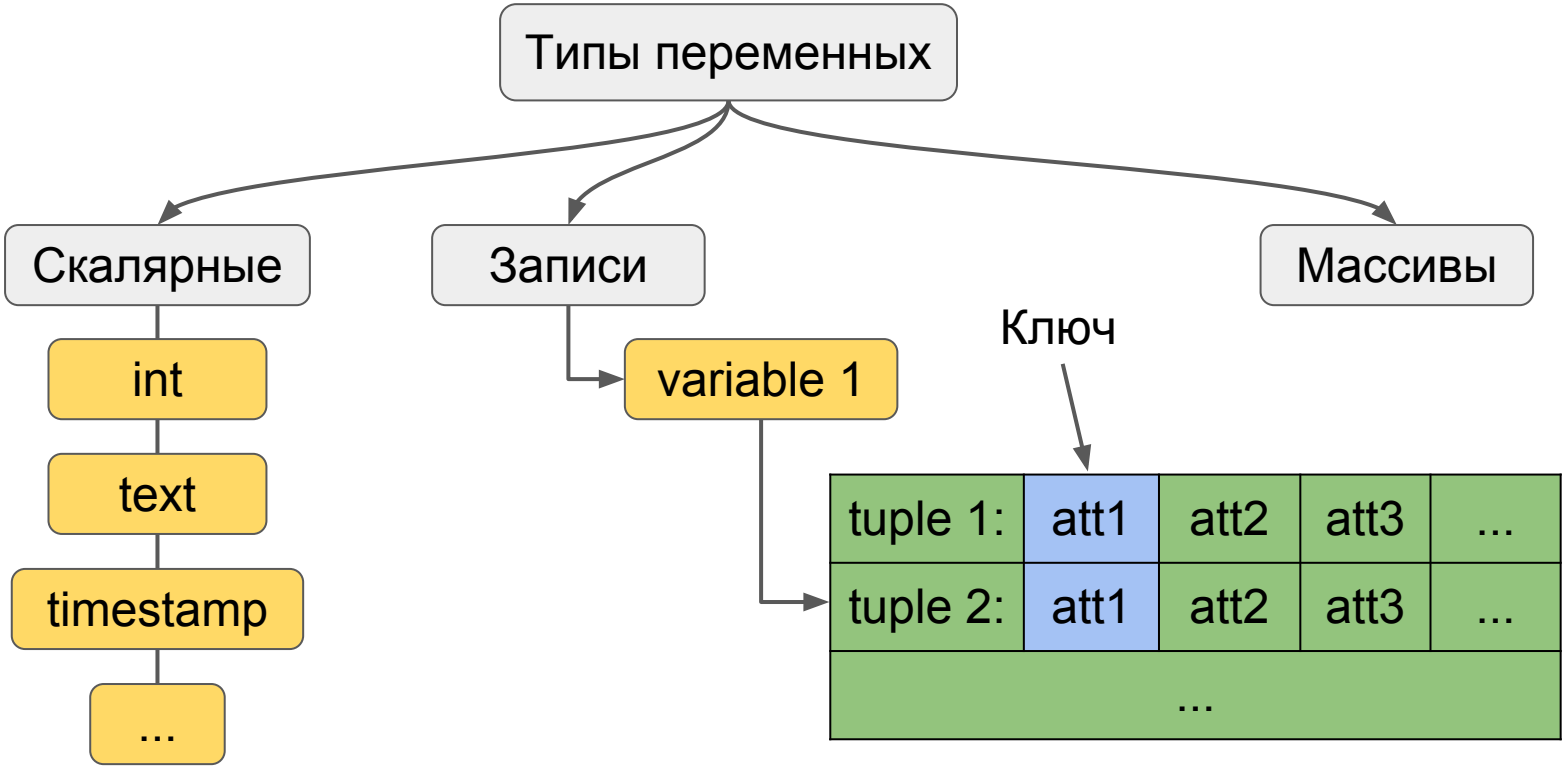
Есть патч (Pavel Stehule, schema variables):

https://commitfest.postgresql.org/21/1608/

# Переменные в pg_variables



# Переменные в pg_variables



# Функции pg_variables

```
select * from package1.variable1; -- нет SQL-интерфейса
```

Работа с переменными с помощью функций:

- pgv_set(package, name, value, is_transactional)
- pgv_get(package, name, var_type, strict)
- pgv_insert(package, name, record, is_transactional)
- pgv_update(package, name, record)
- pgv_delete(package, name, key)
- pgv_select(package, name)
- pgv_select(package, name, key)
- pgv_select(package, name, keys_array)

# Временные таблицы

- Информация о временных таблицах добавляется в системный каталог
- Данные записываются на диск
- Нежурналируемые
- Собирается статистика для планировщика
- Можно создавать индексы
- Autovacuum не выполняет очистку временных таблиц

# Временные таблицы

Минусы:

- При длительных сессиях и обновлениях таблицы каталог может “распухать”
- При длительных сессиях может случиться переполнение счетчика транзакций

```
ERROR:  database is not accepting commands to avoid wraparound
data loss in database "postgres"
HINT:  Stop the postmaster and vacuum that database in
single-user mode.
```

- Нельзя создавать временные таблицы на реплике

# Переполнение счетчика транзакций

| pg_class | |
|---|---|
| **relname** | **relfrozenxid** |
| test_perm | 255764846 |
| test_temp | 568 |
| pg_statistic | 255764846 |
| ... | |

| pg_database | |
|---|---|
| **datname** | **datfrozenxid** |
| postgres | 568 |
| template0 | 200711590 |
| template1 | 201034181 |

min(relfrozenxid)

# Переполнение счетчика транзакций

Решение проблемы:

```
$ pg_ctl stop -D data
$ postgres --single -D data postgres
backend> drop table test_temp1;
backend> drop table test_temp2;
backend> vacuum;
```

# SELECT: pg_variables

Демо база:

https://edu.postgrespro.ru/bookings.pdf

```
=# select pgv_insert('bookings', 'tickets', tickets)
          from tickets;
3518,016 ms
2949857 rows

=# select pgv_select('bookings', 'tickets',
                      '0005432001526'::char(13));
0,501 ms
```

# SELECT: временная таблица

```
=# create temp table tickets_tmp
                     as select * from tickets;
1974,033 ms
2949857 rows

=# select * from tickets_tmp
            where ticket_no = '0005432001526'::char(13);
489,573 ms
```

# SELECT: временная таблица

```
=# create temp table tickets_tmp_idx (
                     ticket_no char(13) primary key,
                     book_ref char(6) not null,
                     passenger_id varchar(20) not null,
                     passenger_name text not null,
                     contact_data jsonb);

=# insert into tickets_tmp_idx select * from tickets;
11214,471 ms

=# select * from tickets_tmp_idx
             where ticket_no = '0005432001526'::char(13);
0,628 ms
```

# SELECT: сравнение

```
=# do
  $$begin
    for i in 1..100000 loop
      perform ...;
    end loop;
  end$$;
```

**447,003 ms (pg_variables)**

**vs**

**1394,275 ms (таблица)**

# SELECT c LIMIT: pg_variables

```
=# select pgv_select('bookings', 'tickets')
          limit 1000 offset 1000000;
313,485 ms

Limit(... rows=1000 ...)
   -> ProjectSet(... rows=1001000 ...)
       ->  Result (... rows=1 ...)
 Planning Time: 0.090 ms
 Execution Time: 365.199 ms
```

# SELECT с LIMIT: временная таблица

```
=# select * from tickets_tmp_idx
             limit 1000 offset 1000000;
99,822 ms

Limit(... rows=1000 ...)
   Buffers: local hit=16762
   ->  Seq Scan on tickets_tmp_idx(... rows=1001000 ...)
       Buffers: local hit=16762
 Planning Time: 0.147 ms
 Execution Time: 174.220 ms

=# alter system set temp_buffers to ...;
```

# JOIN: pg_variables

```
=# select pgv_insert('bookings', 'flights_v',
          row(flight_id, scheduled_departure,
              departure_city, arrival_city))
          from flights_v;
```

**377,881 ms**

**214867 rows**

# JOIN: таблица и pg_variables

```
=# select f.scheduled_departure, f.departure_city,
          f.arrival_city, tf.fare_conditions, tf.amount
   from ticket_flights tf,
   lateral pgv_select('bookings', 'flights_v', tf.flight_id)
           as f(flight_id int,
                scheduled_departure timestamptz,
                departure_city text,
                arrival_city text)
   where tf.ticket_no = '0005432661915'
   order by f.scheduled_departure;
1,509 ms
6 rows
```

# JOIN: таблицы

```
=# select f.scheduled_departure, f.departure_city,
          f.arrival_city, tf.fare_conditions, tf.amount
   from ticket_flights tf
   join flights_v f on tf.flight_id = f.flight_id
   where tf.ticket_no = '0005432661915'
   order by f.scheduled_departure;
3,716 ms
6 rows
```

# JOIN: сравнение

```
=# do
  $$begin
    for i in 1..100000 loop
      perform ...;
    end loop;
  end$$;
```

**3694,762 ms (pg_variables)**

**vs**

**8790,846 ms (таблица)**

# IN: pg_variables

```
=# select pgv_insert('bookings', 'airports',
          row(airport_code, airport_name, city))
          from airports;
4,339 ms
104 rows
=# select a.a_code, a.a_name, a.city
          from pgv_select('bookings', 'airports',
               (select array_agg(distinct departure_airport)
                from flights where status = 'Cancelled'))
               as a(a_code char(3), a_name text, city text)
          order by a.city, a.a_code;
53,602 ms
103 rows
```

# IN: таблица

```
=# select a.airport_code, a.airport_name, a.city
          from airports a
          where a.airport_code in
                (select distinct departure_airport
                 from flights where status = 'Cancelled')
          order by a.city, a.airport_code;
```

**32,923 ms**
**103 rows**

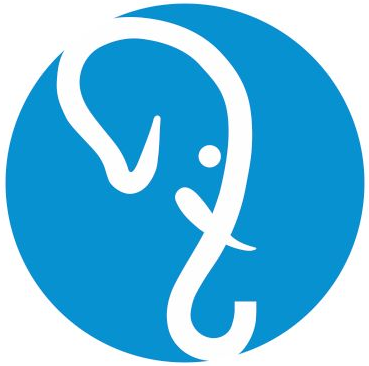
# Ссылки

pg_variables:

https://github.com/postgrespro/pg_variables

Демо база:

https://edu.postgrespro.ru/bookings.pdf



**Postgres Professional**

www.postgrespro.ru

+7 (495) 150-06-91

info@postgrespro.ru