PostgreSQL-Consulting

data egret





# Что нового в PostgreSQL в плане мониторинга

Alexey Lesovsky
alexey.lesovsky@dataegret.com

# Кто докладчик?

PostgreSQL DBA, системный администратор

Data Egret — PostgreSQL консалтинг и поддержка

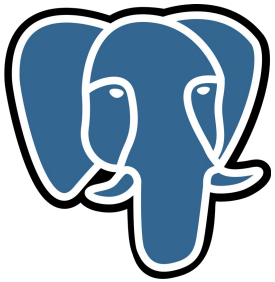
Люблю мониторинги, статистику, визуализации


cacti


Zabbix


Nagios®


Prometheus



PgCluu

PgHero

PGObserver

PGWATCH

Grafana

## The Statistics Collector

https://www.postgresql.org/docs/current/monitoring-stats.html

PostgreSQL's *statistics collector* is a subsystem that supports collection and reporting of information about server activity. Presently, the collector can count accesses to tables and indexes in both disk-block and individual-row terms. It also tracks the total number of rows in each table, and information about vacuum and analyze actions for each table. It can also count calls to user-defined functions and the total time spent in each one.

PostgreSQL also supports reporting dynamic information about exactly what is going on in the system right now, such as the exact command currently being executed by other server processes, and which other connections exist in the system. This facility is independent of the collector process.

### 27.2.1. Statistics Collection Configuration

Since collection of statistics adds some overhead to query execution, the system can be configured to collect or not collect information. This is controlled by configuration parameters that are normally set in `postgresql.conf`. (See **Chapter 19** for details about setting configuration parameters.)

The parameter **track_activities** enables monitoring of the current command being executed by any server process.

The parameter **track_counts** controls whether statistics are collected about table and index accesses.

The parameter **track_functions** enables tracking of usage of user-defined functions.

The parameter **track_io_timing** enables monitoring of block read and write times.

Normally these parameters are set in `postgresql.conf` so that they apply to all server processes, but it is possible to turn them on or off in individual sessions using the **SET** command. (To prevent ordinary users from hiding their activity from the administrator, only superusers are allowed to change these parameters with `SET`.)

The statistics collector transmits the collected information to other PostgreSQL processes through temporary files. These files are stored in the directory named by the **stats_temp_directory** parameter, `pg_stat_tmp` by default. For better performance, `stats_temp_directory` can be pointed at a RAM-based file system, decreasing physical I/O requirements. When the server shuts down cleanly, a permanent copy of the statistics data is stored in the `pg_stat` subdirectory, so that statistics can be retained across server restarts. When recovery is performed at server start (e.g., after immediate shutdown, server crash, and point-in-time recovery), all statistics counters are reset.

### 27.2.2. Viewing Statistics

Several predefined views, listed in **Table 27.1**, are available to show the current state of the system. There are also several other views, listed in **Table 27.2**, available to show the results of statistics collection. Alternatively, one can build custom views using the underlying statistics functions, as discussed in **Section 27.2.20**.

When using the statistics to monitor collected data, it is important to realize that the information does not update instantaneously. Each individual server process transmits new statistical counts to the collector just before going idle; so a query or transaction still in progress does not affect the displayed totals. Also, the collector itself emits a new report at most once per `PGSTAT_STAT_INTERVAL` milliseconds (500 ms unless altered while building the server). So the displayed information lags behind actual activity. However, current-query information collected by `track_activities` is always up-to-date.

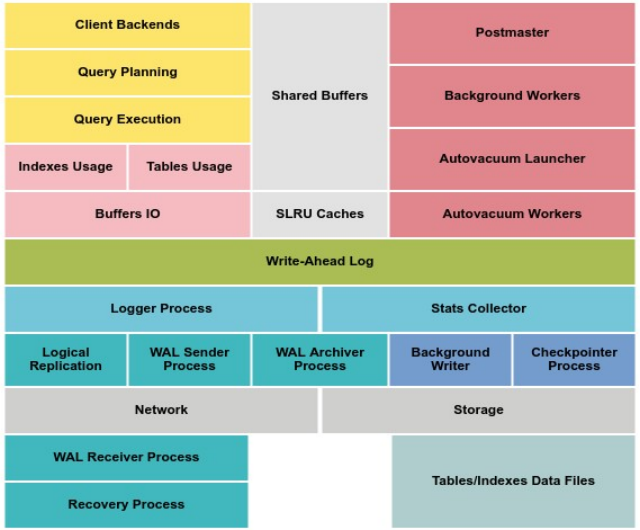
Another important point is that when a server process is asked to display any of these statistics, it first fetches the most recent report emitted by the collector process and then continues to use this snapshot for all statistical views and functions until the end of its current transaction. So the statistics will show static information as long as you continue the current transaction. Similarly, information about the current queries of all sessions is collected when any such information is first requested within a transaction, and the same information will be displayed throughout the transaction. This is a feature, not a bug, because it allows you to perform several queries on the statistics and correlate the results without worrying that the numbers are changing underneath you. But if you want to see new results with each query, be sure to do the queries outside any transaction block. Alternatively, you can invoke `pg_stat_clear_snapshot()`, which will discard the current transaction's statistics snapshot (if any). The next use of statistical information will cause a new snapshot to be fetched.

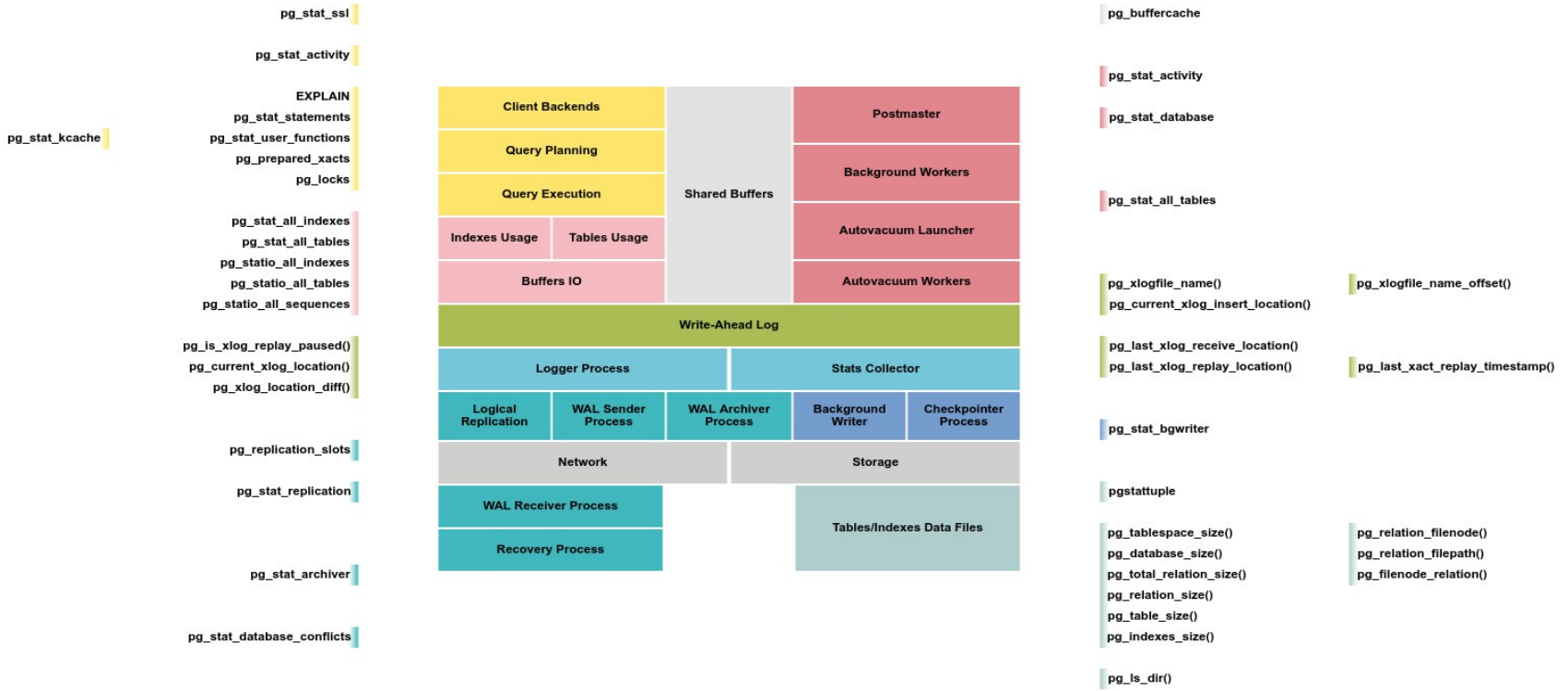
A transaction can also see its own statistics (as yet untransmitted to the collector) in the views `pg_stat_xact_all_tables`, `pg_stat_xact_sys_tables`, `pg_stat_xact_user_tables`, and `pg_stat_xact_user_functions`. These numbers do not act as stated above; instead they update continuously throughout the transaction.

Some of the information in the dynamic statistics views shown in **Table 27.1** is security restricted. Ordinary users can only see all the information about their own sessions (sessions belonging to a role that they are a member of). In rows about other sessions, many columns will be null. Note, however, that the existence of a session and its general properties such as its sessions user and database are visible to all users. Superusers and members of the built-in role `pg_read_all_stats` (see also **Section 21.5**) can see all the information about all sessions.

**Table 27.1. Dynamic Statistics Views**

| View Name | Description |
|---|---|
| `pg_stat_activity` | One row per server process, showing information related to the current activity of that process, such as state and current query. See `pg_stat_activity` for details. |
| `pg_stat_replication` | One row per WAL sender process, showing statistics about replication to that sender's connected standby server. See `pg_stat_replication` for details. |
| `pg_stat_wal_receiver` | Only one row, showing statistics about the WAL receiver from that receiver's connected server. See `pg_stat_wal_receiver` for details. |
| `pg_stat_subscription` | At least one row per subscription, showing information about the subscription workers. See `pg_stat_subscription` for details. |
| `pg_stat_ssl` | One row per connection (regular and replication), showing information about SSL used on this connection. See `pg_stat_ssl` for details. |
| `pg_stat_gssapi` | One row per connection (regular and replication), showing information about GSSAPI authentication and encryption used on this connection. See `pg_stat_gssapi` for details. |


Client Backends
Query Planning
Query Execution
Indexes Usage
Tables Usage
Buffers IO
Shared Buffers
SLRU Caches
Postmaster
Background Workers
Autovacuum Launcher
Autovacuum Workers
Write-Ahead Log
Logger Process
Stats Collector
Logical Replication
WAL Sender Process
WAL Archiver Process
Background Writer
Checkpointer Process
Network
Storage
WAL Receiver Process
Recovery Process
Tables/Indexes Data Files




https://pgstats.dev


https://pgstats.dev

# 1 / 7

**Две минуты**
**Турецкий! (с)**

*Snatch, 2001*

pg_stat_activity — недостаточно

9.6 pg_stat_progress_vacuum

12 pg_stat_progress_create_index

12 pg_stat_progress_cluster

# pg_stat_progress_analyze, 13

```
View "pg_stat_progress_analyze"
--------------------------------
 pid
 datid
 datname
 relid
 phase
 sample_blks_total
 sample_blks_scanned
 ext_stats_total
 ext_stats_computed
 child_tables_total
 child_tables_done
 current_child_table_relid
```

# pg_stat_progress_analyze, 13

+ этап операции

```
View "pg_stat_progress_analyze"
-------------------------------
 pid
 datid
 datname
 relid
 phase
 sample_blks_total
 sample_blks_scanned
 ext_stats_total
 ext_stats_computed
 child_tables_total
 child_tables_done
 current_child_table_relid
```

# pg_stat_progress_analyze, 13

+ этап операции

+ размер сэмпла и сколько обработано

```
View "pg_stat_progress_analyze"
-------------------------------
 pid
 datid
 datname
 relid
 phase
 sample_blks_total
 sample_blks_scanned
 ext_stats_total
 ext_stats_computed
 child_tables_total
 child_tables_done
 current_child_table_relid
```

# pg_stat_progress_analyze, 13

+ этап операции

+ размер сэмпла и сколько обработано

+ поддержка партиционированных таблиц

```
View "pg_stat_progress_analyze"
-------------------------------
 pid
 datid
 datname
 relid
 phase
 sample_blks_total
 sample_blks_scanned
 ext_stats_total
 ext_stats_computed
 child_tables_total
 child_tables_done
 current_child_table_relid
```

# pg_stat_progress_analyze, 13

+ этап операции
+ размер сэмпла и сколько обработано
+ поддержка партиционированных таблиц
+ соединение с pg_stat_activity, pg_locks

```
View "pg_stat_progress_analyze"
--------------------------------
 pid
 datid
 datname
 relid
 phase
 sample_blks_total
 sample_blks_scanned
 ext_stats_total
 ext_stats_computed
 child_tables_total
 child_tables_done
 current_child_table_relid
```

# pg_stat_progress_basebackup, 13

```
View pg_stat_progress_basebackup
--------------------------------
 pid
 phase
 backup_total
 backup_streamed
 tablespaces_total
 tablespaces_streamed
```

# pg_stat_progress_basebackup, 13

+ этап операции

```
View pg_stat_progress_basebackup
--------------------------------
 pid
 phase
 backup_total
 backup_streamed
 tablespaces_total
 tablespaces_streamed
```

# pg_stat_progress_basebackup, 13

+ этап операции

+ полный размер и сколько уже отправлено

```
View pg_stat_progress_basebackup
--------------------------------

 pid
 phase
 backup_total
 backup_streamed
 tablespaces_total
 tablespaces_streamed
```

# pg_stat_progress_basebackup, 13

+ этап операции

+ полный размер и сколько уже отправлено

+ соединение с pg_stat_activity, pg_locks?

```
View pg_stat_progress_basebackup
--------------------------------
 pid
 phase
 backup_total
 backup_streamed
 tablespaces_total
 tablespaces_streamed
```

# pg_stat_progress_copy, 14

```
View pg_stat_progress_copy
--------------------------
 pid
 datid
 datname
 relid
 command
 type
 bytes_processed
 bytes_total
 tuples_processed
 tuples_excluded
```

# pg_stat_progress_copy, 14

+ детали COPY команды

```
View pg_stat_progress_copy
--------------------------
 pid
 datid
 datname
 relid
 command
 type
 bytes_processed
 bytes_total
 tuples_processed
 tuples_excluded
```

# pg_stat_progress_copy, 14

+ детали COPY команды

+ сколько обработано байт и строк

```
View pg_stat_progress_copy
--------------------------
 pid
 datid
 datname
 relid
 command
 type
 bytes_processed
 bytes_total
 tuples_processed
 tuples_excluded
```

# pg_stat_progress_copy, 14

+ детали COPY команды

+ сколько обработано байт и строк

+ соединение с pg_stat_activity, pg_locks

```
View pg_stat_progress_copy
--------------------------
 pid
 datid
 datname
 relid
 command
 type
 bytes_processed
 bytes_total
 tuples_processed
 tuples_excluded
```

CHECKPOINT

CREATE ..., ALTER ..., DROP ...

REFRESH ...

SELECT ..., INSERT ..., UPDATE ..., DELETE ...

# 2/ 7

## Тик так, мистер Уик, тик так

*John Wick: Chapter 3 Parabellum, 2019*

pg_stat_activity

- backend_start
- xact_start
- query_start
- state_change

pg_stat_activity

- backend_start
- xact_start
- query_start
- state_change → state

pg_stat_activity

- backend_start
- xact_start → state → idle
- query_start → active
- state_change → idle in transaction
- → idle in transaction (aborted)
- → fastpath function call

Сколько времени подключения проводят в состояниях?

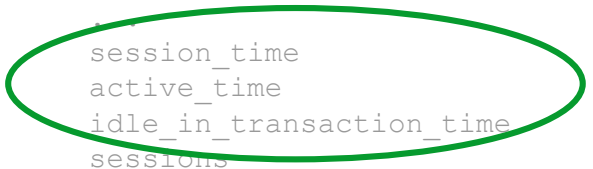
# pg_stat_database, 14

```
View pg_stat_database
-------------------------
 ...
 session_time
 active_time
 idle_in_transaction_time
 sessions
 sessions_abandoned
 sessions_fatal
 sessions_killed
 ...
```

# pg_stat_database, 14

Время проведенное в сессиях

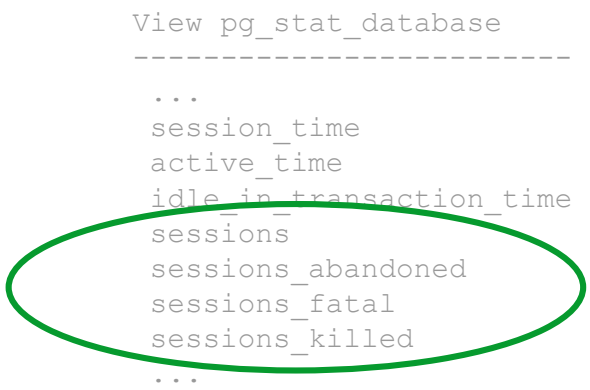
```
View pg_stat_database
-------------------------
 ...
 session_time
 active_time
 idle_in_transaction_time
 sessions
 sessions_abandoned
 sessions_fatal
 sessions_killed
 ...
```



# pg_stat_database, 14

Время проведенное в сессиях

Статусы завершения сессий



| datname | backends_total | session_total | active_total | idle_xact_total | session,ms | active,ms | idle_xact,ms | sessions | abandoned | fatal | killed |
|---|---|---|---|---|---|---|---|---|---|---|---|
| lesovsky | 1 | 166:09:30 | 00:20:37 | 00:00:00 | 1012.78 | 11.84 | 0.00 | 0 | 0 | 0 | 0 |
| pgbench | 0 | 122:11:18 | 10:28:28 | 00:16:57 | 0.00 | 0.00 | 0.00 | 0 | 0 | 0 | 0 |
| postgres | 0 | 05:25:01 | 00:42:53 | 00:00:00 | 0.00 | 0.00 | 0 | 0 | 0 | 0 | 0 |
| flight_v2 | 0 | 01:01:21 | 00:06:23 | 00:00:00 | 0.00 | 0.00 | 0.00 | 0 | 0 | 0 | 0 |
| flight | 1 | 00:32:47 | 00:06:14 | 00:00:00 | 1692.82 | 322.24 | 0.00 | 698 | 0 | 0 | 0 |
| env_signatures | 0 | 00:22:31 | 00:20:05 | 00:00:06 | 0.00 | 0.00 | 0.00 | 0 | 0 | 0 | 0 |
| pgscv_fixtures | 0 | 00:22:29 | 00:17:00 | 00:00:00 | 0.00 | 0.00 | 0 | 0 | 0 | 0 | 0 |
| template0 | 0 | 00:00:00 | 00:00:00 | 00:00:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 00:00:00 | 00:00:00 | 00:00:00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| template1 | 0 | 00:00:00 | 00:00:00 | 00:00:00 | 0.00 | 0.00 | 0 | 0 | 0 | 0 | 0 |

# 3/7

**Nobody panics when things go according to plan. Even if the plan is horrifying!**

*The Dark Knight. 2008*

EXPLAIN и auto_exaplain — выборочное планирование

- Ручной запуск (EXPLAIN)
- Поиск и анализ журналов

Время планирования в pg_stat_statements (13)

- time переменовали в exec_time
- добавили plan_time
- pg_stat_statements.track_planning -- default: off

Отметка уровня выполнения (14)

- pg_stat_statements.track: all vs top
- track = all — двойной подсчет в агрегатах
- pg_stat_statements.toplevel

Track rows for utility commands (14)

- REFRESH, CREATE TABLE AS, SELECT INTO, FETCH

pg_stat_statements_info (14) — dealloc, stats_reset

Track WAL usage stats (13)

# 4/7

**У меня, все ходы записаны! (с)**

*12 стульев, 1976*

Объем записи от запросов? pg_stat_statements

Объем записи от фоновых служб? pg_stat_bgwriter

Объем записи WAL — ?

Объем записи от запросов? pg_stat_statements

Объем записи от фоновых служб? pg_stat_bgwriter

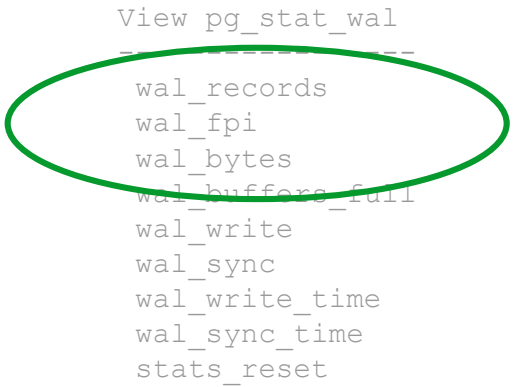
Объем записи WAL — SELECT pg_current_wal_lsn() - '0/0'::pg_lsn;

# pg_stat_wal, 14

```
View pg_stat_wal
-----------------
 wal_records
 wal_fpi
 wal_bytes
 wal_buffers_full
 wal_write
 wal_sync
 wal_write_time
 wal_sync_time
 stats_reset
```

# pg_stat_wal, 14

Количественные характеристики



# pg_stat_wal, 14

Количественные характеристики

Нехватка wal_buffers

```
View pg_stat_wal
-----------------
 wal_records
 wal_fpi
 wal_bytes
 wal_buffers_full
 wal_write
 wal_sync
 wal_write_time
 wal_sync_time
 stats_reset
```

# pg_stat_wal, 14

Количественные характеристики

Нехватка wal_buffers

Затраченное время

```
View pg_stat_wal
-----------------
 wal_records
 wal_fpi
 wal_bytes
 wal_buffers_full
 wal_write
 wal_sync
 wal_write_time
 wal_sync_time
 stats_reset
```

Трекинг WAL статистики, 13:

- EXPLAIN, auto_explain
- autovacuum
- pg_stat_statements

# 5 / 7

**One does not simply (c)**

*The Lord of the Rings: The Fellowship of the Ring, 2001*

Комплексный учет памяти: RES, VIRT, SHM, HUGEPAGES и т.п.

top, htop могут, и часто вводят в заблуждение

Можно использовать данные из /proc/$PID/

https://blog.anarazel.de/2020/10/07/measuring-the-memory-overhead-of-a-postgres-connection/

# pg_backend_memory_contexts, 14

```
View pg_backend_memory_contexts
-------------------------------
 name
 ident
 parent
 level
 total_bytes
 total_nblocks
 free_bytes
 free_chunks
 used_bytes
```

# pg_backend_memory_contexts, 14

Детали об участке памяти

```
View pg_backend_memory_contexts
-------------------------------
 name
 ident
 parent
 level
 total_bytes
 total_nblocks
 free_bytes
 free_chunks
 used_bytes
```

# pg_backend_memory_contexts, 14

Детали об участке памяти

Статистика total, free, used

```
View pg_backend_memory_contexts
-------------------------------
 name
 ident
 parent
 level
 total_bytes
 total_nblocks
 free_bytes
 free_chunks
 used_bytes
```

# pg_backend_memory_contexts, 14

Детали об участке памяти

Статистика total, free, used

pg_log_backend_memory_contexts(integer)

# 6/7

– Dance with me.

– You don't dance.

– It was just my cover.

– Was sloth your cover, too?

*Mr. & Mrs. Smith, 2005*

pg_stat_activity

pg_stat_statements

pg_stat_activity.queryid

pg_stat_statements.queryid

compute_query_id, 14

```
select a.*, s.* from pg_stat_activity a inner join pg_stat_statements s on (a.datid = s.dbid AND a.usesysid = s.userid AND a.query_id = s.queryid)
where a.pid = 1001291;

-[ RECORD 1 ]-------+------------------------------------------------------------
datid               | 16413
datname             | pgbench
pid                 | 1001291
leader_pid          |
usesysid            | 10
usename             | postgres
application_name    | pgbench
client_addr         |
client_hostname     |
client_port         | -1
backend_start       | 2021-05-22 10:15:57.299468+05
xact_start          | 2021-05-22 10:16:25.566151+05
query_start         | 2021-05-22 10:16:25.566623+05
state_change        | 2021-05-22 10:16:25.566763+05
wait_event_type     |
wait_event          |
state               | idle in transaction
backend_xid         | 237577
backend_xmin        |
query_id            | 2517686606037258902
query               | SELECT abalance FROM pgbench_accounts WHERE aid = 1715456;
backend_type        | client backend
userid              | 10
dbid                | 16413
toplevel            | t
queryid             | 2517686606037258902
query               | SELECT abalance FROM pgbench_accounts WHERE aid = $1
plans               | 0
total_plan_time     | 0
min_plan_time       | 0
max_plan_time       | 0
mean_plan_time      | 0
stddev_plan_time    | 0
calls               | 209439
total_exec_time     | 4251.98569499987
min_exec_time       | 0.005414
max_exec_time       | 0.435581
mean_exec_time      | 0.020301785698938563
stddev_exec_time    | 0.005889254053319066
rows                | 209439
shared_blks_hit     | 884097
```

pg_stat_activity.leader_pid (13)

```
lesovsky@matanuii:~  $ pgcenter profile -P 1050881 -W -U postgres pgbench
LOG: Profiling process 1050881 with 100ms sampling
------ ------------ ----------------------------
% time      seconds wait_event                    query: select count(*) from pgbench_accounts ;
------ ------------ ----------------------------
 90.91    61.090423 IO.DataFileRead
  7.74     5.203075 Running
  1.34     0.902708 IPC.BtreePage
------ ------------ ----------------------------
100.00    67.196206 including workers
          67.196206
^Cgot interrupt
lesovsky@matanuii:~  $ pgcenter profile -P 1050881 -U postgres pgbench
LOG: Profiling process 1050881 with 100ms sampling
------ ------------ ----------------------------
% time      seconds wait_event                    query: select count(*) from pgbench_accounts ;
------ ------------ ----------------------------
 90.00   176.637457 IO.DataFileRead
  9.73    19.102891 Running
  0.20     0.401478 IPC.BtreePage
  0.06     0.121435 LWLock.BufferIO
------ ------------ ----------------------------
100.00   196.263261 including workers
          65.421091
^Cgot interrupt
lesovsky@matanuii:~  $
```

*Inception, 2010*

## pg_shmem_allocations — распределение памяти в shared buffers (13)

pg_shmem_allocations — распределение памяти в shared buffers (13)

pg_stat_slru — использование SLRU кэшей (13)

pg_shmem_allocations — распределение памяти в shared buffers (13)

pg_stat_slru — использование SLRU кэшей (13)

pg_stat_replication_slots — статистика слотов репликации (14)

pg_shmem_allocations — распределение памяти в shared buffers (13)

pg_stat_slru — использование SLRU кэшей (13)

pg_stat_replication_slots — статистика слотов репликации (14)

pg_locks.waitstart (14)

pg_shmem_allocations — распределение памяти в shared buffers (13)

pg_stat_slru — использование SLRU кэшей (13)

pg_stat_replication_slots — статистика слотов репликации (14)

pg_locks.waitstart (14)

Время выполнения IO при логировании auto-vacuum/analyze (14)

THE END

pg_stat_actvity, pg_stat_progress_*

pg_stat_statements


https://pgstats.dev

# pg_stat_activity

Related items: Client Backends, Query Execution, Background Workers, Autovacuum Launcher, Autovacuum Workers

The pg_stat_activity view will have one row per server process, showing information related to the current activity of that process.

## TIPS'N'TRICKS

- Use 'now() - xact_start' for get human-readable age of transactions. Same approach can be used with 'backend_start', 'query_start' and 'state_change'.
- Use 'coalesce(usename, backend_type)', because system processes like autovacuum workers have NULL in 'usename'.

## RESOURCES

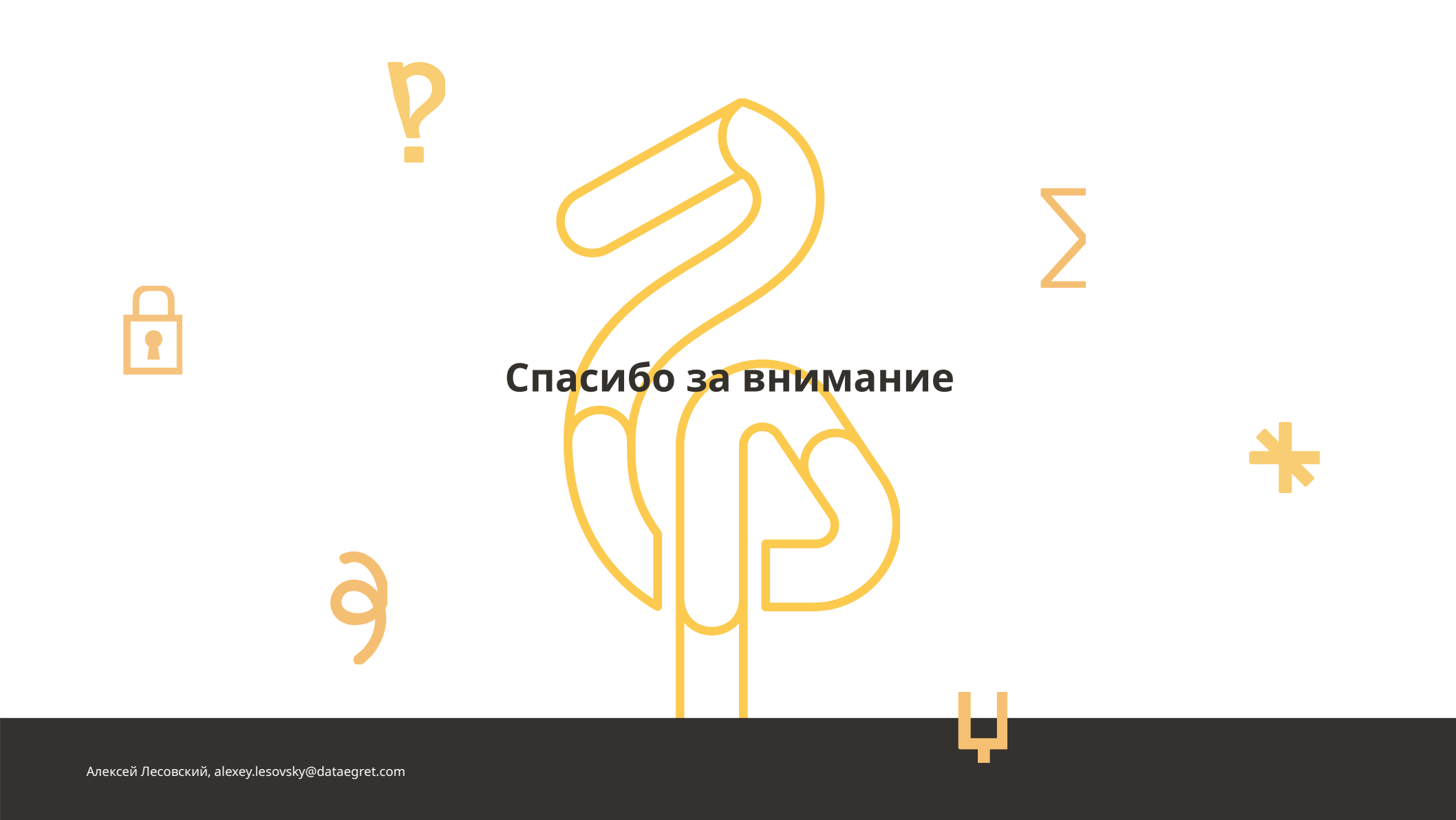
- Official documentation
- Summary activity. Brief information about connected clients.
- Long activity. Current activity longer than 100ms.

## CHANGES

- 14: Add 'query_id' to report query ID of recent query executed by backend.
- 13: Add 'leader_pid' to report a parallel worker's leader process.
- 10: Add 'backend_type' column which shows auxiliary processes, background workers, and wal sender processes.
- 10: Allow to show the SQL query being executed by parallel workers.
- 9.6: Replace the 'waiting' column with 'wait_event_type' and 'wait_event'.

## COLUMNS DESCRIPTION

| name | type | description |
|---|---|---|
| datid | oid | OID of the database this backend is connected to. |
| datname | name | Name of the database this backend is connected to. |
| pid | integer | Process ID of this backend. |
| leader_pid | integer | Process ID of the parallel group leader, if this process is a parallel query worker. NULL if this process is a parallel group leader or does not participate in parallel query. **Added in Postgres 13.** |
| usesysid | oid | OID of the user logged into this backend. |
| usename | name | Name of the user logged into this backend. |
| application_name | text | Name of the application that is connected to this backend. |
| client_addr | inet | IP address of the client connected to this backend. If this field is NULL, it indicates either that the client is connected via a Unix socket on the server machine or that this is an internal process such as autovacuum. |
| client_hostname | text | Host name of the connected client, as reported by a reverse DNS lookup of client_addr. This field will only be non-NULL for IP connections, and only when 'log_hostname' is enabled. |
| client_port | integer | TCP port number that the client is using for communication with this backend, or -1 if a Unix socket is used. If this field is NULL, it indicates that this is an internal server process. |
| backend_start | timestamp with time zone | Time when this process was started. For client backends, this is the time the client connected to the server. |
| xact_start | timestamp with time zone | Time when this process' current transaction was started, or NULL if no transaction is active. If the current query is the first of its transaction, this column is equal to the 'query_start' column. |
| query_start | timestamp with time zone | Time when the currently active query was started, or if 'state' is not active, when the last query was started. |
| state_change | timestamp with time zone | Time when the state was last changed. |
| waiting | boolean | True if this backend is currently waiting on a lock. **Removed in Postgres 9.6.** |
| wait_event_type | text | The type of event for which the backend is waiting, if any; otherwise NULL. See Wait Event Types table. **Added in Postgres 9.6.** |
| wait_event | text | Wait event name if backend is currently waiting, otherwise NULL. See the group of Wait Events tables. **Added in Postgres 9.6.** |
| | | Current overall state of this backend. Possible values are: * active: The backend is executing a query. * idle: The backend is waiting for a new client command. * idle in transaction: The backend is in a transaction, but is not |



# Спасибо за внимание

Алексей Лесовский, alexey.lesovsky@dataegret.com