Ваши эксперты по PostgreSQL

data egret

Инструменты от Data Egret для облегчения администрирования баз данных. Что и в каких случаях применять?

Александр Никитин. DBA DataEgret. 2021-07-08

https://github.com/dataegret/pg-utils

Скриптов много, но что и в каких случаях запускать?

Как интерпретировать результаты их выполнения?

4 top_databases.sql

```
       name        |  owner   |  size
-------------------+----------+---------
 db1               | user1    | 3408 GB
 db2               | user2    | 827 MB
 postgres          | postgres | 29 MB
```

Применение: Узнать, какие базы данных занимают больше всего места.

5 top_tables.sql

```
 nspname | relname     | type |  size   | idxsize |  total
---------+-------------+------+---------+---------+---------
 public  | tbl1        | r    | 467 GB  | 615 GB  | 1082 GB
 public  | tbl2        | r    | 100 GB  | 170 GB  | 269 GB
 public  | tbl1_i_idx  | i    | 132 GB  | 0 bytes | 132 GB
```

Применение: узнать какие объекты занимают больше всего места в БД.

6 table_write_activity.sql

```
 table      | size  |   tblsp    | seq_scan  |   idx_scan    | n_tup_ins  | n_tup_upd  | n_tup_del  |   total    | hot_rate | fillfactor
------------+-------+------------+-----------+---------------+------------+------------+------------+------------+----------+------------
public.tbl1 | 96 MB | pg_default |   8000784 |     196879723 | 8418672871 |  171580771 |          0 | 8664741784 |    56.59 |
public.tbl2 | 32 GB | pg_default |       807 |  36731890581 |   41344045 | 1907381516 |          0 | 3855732138 |     0.02 |
```

hot_rate – процент HOT-апдейтов от всех апдейтов

Применение: 50 таблиц, в которых изменения максимальны. Поиск индексов, которые ломают HOT-апдейты.

7 table_index_write_activity.sql

```
 table       | size    | tblsp |   tblsp    | seq_scan  | idx_scan   | n_tup_ins  | n_tup_upd  | n_tup_del  |   total    | hot_rate | fillfactor
-------------+---------+-------+------------+-----------+------------+------------+------------+------------+------------+----------+------------
 public.tbl1 | 96 MB   |       | pg_default |   8000904 |  196882926 | 8418673156 |  171583689 |          0 | 8493162256 |    56.59 |
 public.tbl3 | 145 MB  |       | pg_default |      2561 |   20905668 | 1783251086 |   90844537 | 1783850846 | 3657564409 |     0.42 |
```

hot_rate – процент HOT-апдейтов от всех апдейтов

Применение: 50 таблиц, в которых идёт максимальное изменение индексов, кандидаты на поиск раздутых индексов.

8 table_bloat_approax.sql, table_bloat.sql

```
 nspname | relname | total_size | toast_size | table_waste_percent | table_waste | total_waste_percent | total_waste
---------+---------+------------+------------+---------------------+-------------+---------------------+-------------
 public  | tbl1    | 20 GB      | 0 bytes    |                 1.7 | 353 MB      |                 1.7 | 353 MB
```

Примечание: если не задать маску внутри файла, то будут обработаны все доступные таблицы, после чего 20 самых раздутых будут выведены.

Применение: подсчитывает примерный блоат таблиц, с разной степенью точности, нужно расширение pgstattuple.

9 index_bloat.sql

```
 schema | table | table_size | index_name      | index_size | index_scans | waste_percent |  waste
--------+-------+------------+-----------------+------------+-------------+---------------+---------
 public | tbl1  | 99 MB      | tbl1_i_idx      | 65 MB      |   201335805 |           0.0 | 0 bytes
```

Примечание: если не задать маску внутри файла, то будут обработаны все доступные индексы, после чего 20 самых раздутых будут выведены.

Применение: подсчитывает примерный блоат индекса, нужно расширение pgstattuple.

10 set_default_grants.sql

- role_ro, role_rw

Применение: Переопределяет привилегии по-умолчанию.

11 check_missing_grants.sql

```
   Type    |             Name              | Access privileges
-----------+-------------------------------+-------------------
 sequence  | public.posts_id_seq           |
 table     | public.t3                     |
 table     | public.t4                     |
 table     | public.tbl_count              |
```

Применение: проверка – всем ли объектам выданы права через role_ro и role_rw

12 set_missing_grants.sql

```
NEW GRANT
-------------------------------------------------------------------------------
GRANT SELECT ON TABLE "public"."foo" TO role_ro;
GRANT SELECT ON TABLE "public"."t1" TO role_ro;
GRANT SELECT ON TABLE "public"."tbl_count" TO role_ro;
GRANT SELECT ON TABLE "public"."t2" TO role_ro;
GRANT SELECT,USAGE ON SEQUENCE "public"."t1_id_seq" TO role_rw;
GRANT USAGE ON SCHEMA "public" TO role_rw;
GRANT USAGE ON SCHEMA "public" TO role_ro;
```

Применение: генерирует команды, которые устанавливают права на объекты в соответствии со схемой с двумя типами пользователей.

13 seq_scan_tables.sql

table	n_live_tup	seq_scan	seq_tup_read	write_activity	index_count	idx_scan	idx_tup_fetch
public.tbl1	93115027	1832083	19832014402108	485570465	12	6231691575	5877305963
public.tbl2	16070317	4916650	9947360358855	106709207	12	113611104616	186209407714

Применение: находит 20 таблиц, из которых наиболее интенсивно идёт извлечение записей последовательным сканированием. Повод взглянуть на них пристальнее.

14 redundant_indexes.sql

```
-[ RECORD 1 ]--------+-------------------------------------------------------------
indrelid             | tbl1
main_index           | CREATE INDEX tbl1_i_j_idx ON public.tbl1 USING btree (i, j)
redundant_index      | CREATE INDEX tbl1_i_idx ON public.tbl1 USING btree (i)
redundant_index_size | 8303 MB
```

Применение: лишние индексы тормозят работу системы, необходимо тратить ресурсы на их поддержку и хранение, запуск скрипта поможет выявить избыточные индексы.

15 query_stat_cpu_time.sql, query_stat_io_time.sql, query_stat_rows.sql, query_stat_time.sql

time_%	iotime_%	cputime_%	total_time	avg_time	avg_io_time	calls	call_%	rows	row_%	query
49.14	87.74	49.00	18366870.62	520.717	3.302	35272297	94.79	330293978	99.41	other
40.73	0.00	40.88	15225081.09	5564722.62	0.00	2736	0.01	6086	0.00	SELECT a.field1, ...

Примечание: Для работы этих запросов нужно расширение pg_stat_statement и track_io_timings = ‘on’.

Применение: сортируем запросы по количеству потраченных ресурсов, возвращённых записей. Соответственно, вы можете узнать какие именно запросы потребляют больше всего интересующих вас ресурсов.

16 query_stat_counts.sql

time_%	total_time	avg_time	calls	call_%	rows	row_%	query
74.99	17635741.34	871.388	20238674	40.03	435479980	93.15	other
0.02	5567.31	1.194	4663871	9.22	0	0.00	BEGIN
0.02	4567.85	1.024	4462007	8.82	0	0.00	COMMIT
0.10	23276.30	10.321	2255225	4.46	2255225	0.48	select nextval ('seqEntities')

Применение: пересекается с предыдущими запросами, но упор сделан на количество вызовов того или иного запроса.

17 low_used_indexes.sql

```
table       | indexrelname | idx_scan | write_activity |  seq_scan  | n_live_tup |  size
------------+--------------+----------+----------------+------------+------------+---------
public.tbl1 | tbl1_i_idx   |        0 |     3667037622 |       2561 |     249718 | 84 MB
public.tbl2 | tbl2_i_idx   |  1495476 |     3667037622 |       2561 |     249718 | 79 MB
```

Применение: позволяет выявить малоиспользуемые индексы. К примеру, индексы созданные "про запас".

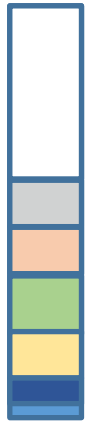
18 locktree.sql

```
  ts_age  | change_age | datname | usename | client_addr |  pid  | state  | lvl | blocked | query
----------+------------+---------+---------+-------------+-------+--------+-----+---------+-------------------------------------------------------------------
--------------------
 00:00:01 | 00:00:00   | db1     | user1   | 127.0.0.1   | 10487 | idletx |   0 |       3 |  UPDATE "tbl1" SET "state" = 'failed', "updated_at" = '2021-
04-20 14:59:44.9086
 00:00:00 | 00:00:00   | db1     | user1   | 127.0.0.1   | 10485 | active |   1 |       1 |  . UPDATE "tbl1" SET "state" = 'failed', "updated_at" = '2021-
04-20 14:59:45.5484
 00:00:00 | 00:00:00   | db1     | user1   | 127.0.0.1   | 10546 | active |   2 |       0 |  . . UPDATE "tbl1" SET "state" = 'failed', "updated_at" =
'2021-04-20 14:59:45.4221
 00:00:00 | 00:00:00   | db1     | user1   | 127.0.0.1   | 10503 | active |   1 |       0 |  . UPDATE "tbl1" SET "state" = 'failed', "updated_at" = '2021-
04-20 14:59:45.6364
```

Применение: показывает дерево блокировок, кто что блокирует.

19 index_candidate_to_partial.sql

```
-[ RECORD 1 ]-----+-------------------------------------------------------------
table             | tbl1
column            | state
index             | tbl1_state_idx
null_frac         | 0
most_common_freqs | {0.560013,0.405883,0.03201,0.00172667,0.000103333}
most_common_vals  | {RETURNED,SHIPPED,CANCELLED,PROCESSABLE,NOT_ENOUGH}
index_size        | 9884 MB
index_def         | CREATE INDEX tbl1_state_idx ON public.tbl1 USING btree (state)
```



Примечание: top5 самых частых значений составляют суммарно как минимум 95% строк от всех not null значений.

Применение: показывает индексы, которые целесообразно сделать частичными.

20 indexes_with_nulls.sql

```
 table | index         | field | stanullfrac | indexsize |    indexdef
-------+---------------+-------+-------------+-----------+-------------------------------------------------------------------------------
 tbl1  | tbl1_i_idx    | i     |    0.989833 | 7843 MB   | CREATE INDEX tbl1_i_idx ON public.tbl1 USING btree (i)
 tbl1  | tbl1_j_idx    | j     |    0.728597 | 7790 MB   | CREATE INDEX tbl1_j_idx ON public.tbl1 USING btree (j)
 tbl2  | tbl2_m_n_idx  | m     |    0.513933 | 4303 MB   | CREATE INDEX tbl2_m_n_idx ON public.tbl2 USING btree (m, n)
 tbl3  | tbl3_f_idx    | f     |    0.606783 | 4154 MB   | CREATE INDEX tbl3_f_idx ON public.tbl3 USING btree (f) WHERE (f IS NOT NULL)
```

Применение: показывает индексы, построенные по полям, в которых большой процент NULL значений.

21 check_uniq_indexes.sql

```
   schema    |                  relname
-------------+-------------------------------------------
 public      | tbl1
 public      | tbl2
```

Применение: показывает таблицы, которые не имеют уникального ключа, это может быть важным, например для логической репликации.

22 check_strange_fk.sql

```
 ?column?     | typname | ?column?     | typname
--------------+---------+--------------+---------
 tbl1.column1 | int4    | tbl2.column1 | int8
 tbl1.column2 | text    | tbl2.column2 | varchar
```

Применение: показывает несовпадение типов данных между полями по которым построено ограничение FK.

23 db_activity9.6.sql

```
    ts_age      |        state        |    query_age    |   change_age    | datname | pid  |  usename | waiting | client_addr | client_port |  query
----------------+---------------------+-----------------+-----------------+---------+------+----------+---------+-------------+-------------+------------
--------------------
 00:00:00.240443 | active              | 00:00:00.2386   | 00:00:00.238598 | db1     | 6799 | user1    | f       |             |          -1 | SELECT i
from tbl1
 00:00:00.146824 | idle in transaction | 00:00:00.001841 | 00:00:00.001774 | db2     | 3771 | user2    | f       | 10.0.0.13   |       50766 | SELECT j
from tbl2 where i>=100
```

Примечание: один из наиболее часто используемых скриптов.

Применение: Благодаря запросу можно довольно быстро понять что именно происходит в системе, какие процессы сейчас протекают, как долго работает тот или иной запрос и т.д.

24 query_stat_total.sql и query_stat_total_13.sql

```
total time:     00:28:28 (IO: 25.11%)
total queries:  1,157,197 (unique: 569)
report for all databases, version 0.9.5 @ PostgreSQL 13.2
tracking top 10000 queries, utilities off, logging 250ms+ queries

================================================================================================================
pos:1    total time: 00:10:18 (36.2%, CPU: 48.3%, IO: 0.0%)      calls: 1,513 (0.13%)    avg_time: 408.69ms (IO: 0.0%)
user: user1    db: db1     rows: 1,513 (0.00%)     query:
DELETE FROM "tbl1" WHERE "id" = $1
================================================================================================================
pos:2    total time: 00:02:29 (8.7%, CPU: 11.6%, IO: 0.1%)       calls: 236 (0.02%)      avg_time: 632.54ms (IO: 0.3%)
user: user2    db: db2     rows: 1,612 (0.00%)     query:
SELECT field1, field2 FROM tbl2 ...
```

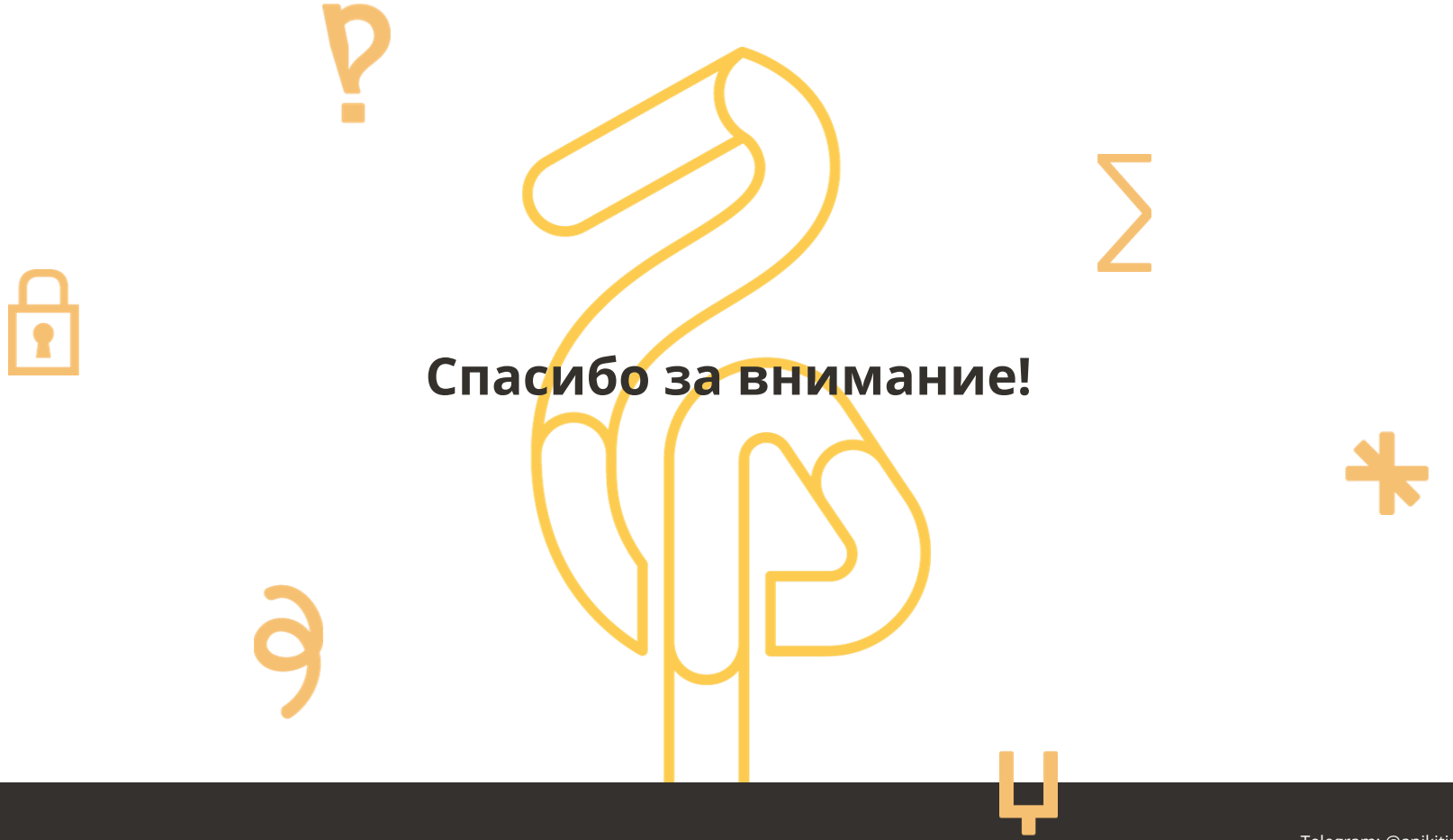
Примечание: Для работы этих запросов нужно расширение pg_stat_statement и track_io_timings = ‘on’.

Применение: при условии скидывания статистики по pg_stat_statement получаем ежедневный срез потребления ресурсов.

Общие выводы

- Скриптов много, но это не должно вас отпугивать.
- Каждый из них был создан для решения той или иной задачи

- Не бойтесь экспериментировать с инструментами (на тестовых машинах, конечно ☺)!



Спасибо за внимание!

Александр Никитин

Telegram: @anikitindba