

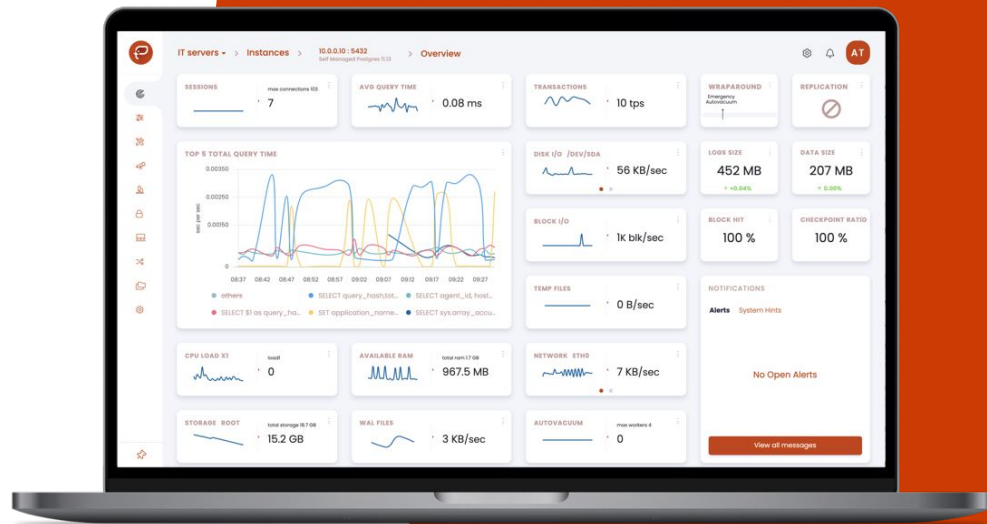


Лаборатории Тантор

Исследования, Производство,
Инновации

Autovacuum.
Вредные советы.

Вадим Яценко





По мотивам проблем компании N. Disclaimer

- NDA;
- Все совпадения и лица случайны;
- Ошибки проектирования и обслуживания банальны и встречаются часто;
- Никакого критиканства, только факты;
- Техническая команда клиента - высококвалифицированные специалисты (без сарказма).





Компания N. Как выглядит бизнес?

- Продукт - облачное хранилище файлов
- Количество файлов не ограничено
- Метаданные, индексы, пользователи и другая информация хранятся в БД PostgreSQL
- Сотни клиентов по всему миру



Azure



Distributed Cloud File System

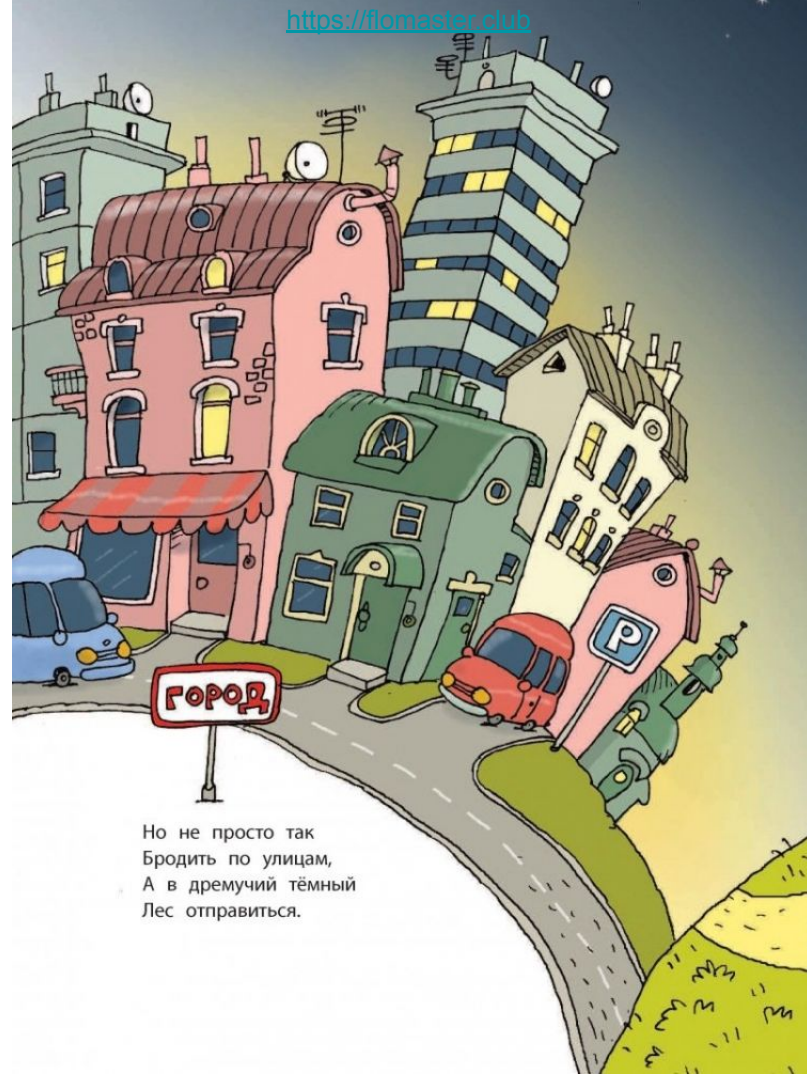
День 1. Звонок

- ~ 4 дня down time;
- PostgreSQL 12.2, БД ~3.5 TB, RAM 128 GB;
- По всем признакам проблема с переполнением счетчика транзакций (wraparound)
- Техническая команда пыталась решить проблему своими силами (прочитали весь Stackoverflow);
- VACUUM FREEZE не помогает.



День 1. Подключаемся

- Физического доступа к серверам клиента нет
- 2 сервера PostgreSQL: primary и standby;
- Оба сервера еще не в single user mode, но уже на грани;
- Репликация уже отключена;
- Backup + PITR есть;
- Checksum-ы выключены.



День 1. Лечим по фотографии через Zoom



- Разрешаем в `pg_hba.conf` только локальное подключение;
- Ищем relation с самым старым XID: `age(pg_class.relFrozenxid):`

table	xid_age	mxid_age	tx_before_wraparound_vacuum	size	last_autovacuum
files	2146483647	106234998	-1146483647	2065 GB	2021-11-29 15:51:47.259058-06
pg_toast.pg_toast_16601	59040376	32596452	667403548	8192 bytes	2022-01-04 11:494673-06
pg_toast.pg_toast_25187618	347924588	19702521	380297479	8192 bytes	2021-12-30 17:53:23.446814-06

- Возраст XID близок к максимально возможному
- `autovacuum_freeze_max_age = 1000000000 !!!`
- Последний `autovacuum` больше месяца назад

Вредные советы. № 1

Если ты нашел конфиг,
В интернете классный,
Применяй на всех БД,
Это не опасно!

Будут базы без сомненья
Функционировать прекрасно,
И зачем конфиг менять,
Каждый раз напрасно?!





День 1. Запускаем VACUUM FREEZE

- Меняем настройки:
 - autovacuum = off (to prevent wraparound все равно запустится)
 - maintenance_work_mem = 100GB
 - vacuum_cost_delay = 0
 - vacuum_cost_limit = 10000

```
(postgres@[local]:5432) [postgres] > VACUUM FREEZE files;  
> ERROR: MultiXactID 97 does not longer exist - apparent wraparound
```




День 1. Самое время посмотреть логи

```
MultiXactID NNN does not longer exist - apparent wraparound  
MultiXactID NNN has not been created yet - apparent wraparound
```

[BUG #15142: ERROR: MultiXactId nnnn has not been created yet -- apparent wraparound in v9.5](#)

```
FWIW this seems to be some sort of data corruption, where the XID got  
overwritten by a bogus value in some way. The bigger question is how  
many other such cases are there.
```

```
regards
```

```
--
```

```
Tomas Vondra
```

```
http://www.2ndQuadrant.com
```

```
PostgreSQL Development, 24x7 Support, Remote DBA, Training & Services
```

Вредные советы. № 2

Будь бесстрашным и отважным,
Не включай проверку сумм!
Блоки битые? Не страшно!
Если есть standby надежный!

Мониторинг, логи, парсинг -
это сложно, долго, нудно,
Только время занимает,
Кто их будет проверять?!





День 1. Ищем иголку в стоге сена

План №1 - решение “в лоб”:

- Пробуем найти все id, где xmax = 97;
- Удаляем строки с помощью обычной команды DELETE.

```
(postgres@[local]:5432) [postgres] >
select xmax, id from files where xmax = 97 limit 1;
 xmax | id
-----+---
(0 rows)
```



День 1. Ищем иголку в стоге сена

План № 2:

- `gdb` - клиент не готов устанавливать на prod сервер;
- Пробуем прочитать построчно из курсора с помощью `FETCH_COUNT`;
- Запрос должен “упасть” на первом поврежденном блоке;
- Таблица > 2TB, индексы около 1 TB, поэтому читать будем долго (см. вредный совет №3).

```
postgres=# \set FETCH_COUNT 1
postgres=# \pset pager off
Pager usage is off.
postgres=# SELECT ctid, id FROM files;
```



День 1. Ищем иголку в стоге сена

...через несколько часов:

```
(38800364, 8) | 8378010208
(38800364, 12) | 8243997611
(38800364, 13) | 8378010226
(38800364, 15) | 8123582815
(38800364, 17) | 9287295631
(38800364, 18) | 9347356909
(38800364, 20) | 9287294134
(38800364, 41) | 9347396533
(38800364, 42) | 9347356899
(38800364, 43) | 9347356906
(38800365, 5) | 8154702765
(38800365, 8) | 820448651
(38800365, 9) | 8243997606
(38800365, 11) | 8413155097
(38800365, 17) | 8187238529
(38800365, 19) | 8154703080
(38800365, 21) | 8154703877
ERROR: MultiXactId 97 does no longer exist -- apparent wraparound
Time: 7.482 ms
(postgres@[local]:5432) [postgres] >
```

Блок 38800366 кандидат на анализ



День 1. Ищем иголку в стоге сена

- Смотрим pageinspect-ом. Функция heap_page_items позволяет получить информацию об указателях и версиях строк.

```
postgres=# CREATE EXTENSION pageinspect;
CREATE EXTENSION
postgres=# select t_xmin, t_xmax, t_ctid FROM
heap_page_items(get_raw_page('files',38800366)) WHERE t_xmax = 97;
 t_xmin  | t_xmax  | t_ctid
-----+-----+-----
1260286 | 97      | (0,1824242)
```

- Новая версия строки не найдена



День 1. Глубока ли кроличья нора?

- Что параллельно удалось выяснить:
 - Блоки повреждены в том числе и на standby-сервере;
 - Есть backup 3 дневной давности + WAL-ы;
 - Восстановление backup-а около 12 часов, блоки тоже повреждены;
- Вопросы, на которые хотелось бы иметь ответы, перед восстановлением:
 - Сколько блоков повреждено;
 - Как блоки “распределены” по таблице;
 - Какую стратегию выбрать:
 - pg_filedump + копирование данных + восстановление
 - поиск и “зануление” битых блоков

Вредные советы. № 3

Если в базу много пишут,
И огромные таблицы,
То, конечно, это круто,
Ты ведь делаешь high-load!

И зачем кому-то надо
Разделять их на кусочки,
Если есть ограничение,
Аж в десятки терабайт!





День 1. dd нас спасет

- Исходные данные таблицы:
 - Таблица без индексов ~ 1 TB;
 - Toast пуст;
 - Каждый файл таблицы размером в 1 GB;
- Ищем расположение таблицы в директории /base:

```
(postgres@[local]:5432) [postgres] > SELECT
pg_relation_filepath('files');
pg_relation_filepath
-----
base/16401/16601
```



День 1. dd нас спасет

- Конвертируем номер блока в номер файла:

```
(38800366 * 8192) / (1024^3) = 296.0233002
```

←----- base/16401/16601.296

- Проверяем с помощью pg_filedump:

```
pg_filedump -i ./base/16401/16601.296 | grep "XMAX: 97"
```

```
XMIN: 0 XMAX: 97 CID|XVAC: 0
```

- Рассчитываем номер блока в файле:

```
(0.0233002 * 1024^3) / 8192 = 3054.0038144
```

←----- Блок 3054

- Останавливаем сервер PostgreSQL и бэкапим файл /base/16401/16601.296



День 1. dd нас спасет

- “Зануляем” весь блок:

```
dd if=/dev/zero of=$PG_DATA/base/16401/16601.296 bs=8192 seek=3054  
count=1 conv=notrunc
```

- Проверяем с помощью pg_filedump:

```
pg_filedump -i ./base/16401/16601.296 | grep "XMAX: 97"
```

- Проверяем, с помощью pageinspect:

```
postgres=# select t_xmin, t_xmax, t_ctid FROM  
heap_page_items(get_raw_page('files',38800366)) WHERE t_xmax = 97;  
 t_xmin | t_xmax | t_ctid  
-----+-----+-----  
(0 rows)
```

День 1. Итоги

- Проблема в неработающем **AUTOVACUUM** из-за битых блоков в самой большой таблице БД (~2 TB);
- Как минимум 2 типа ошибок, связанных с этой проблемой. Количество битых блоков не известно;
- Нашли 1 битый блок с $x_{max} = 97$ и его “занулили”;
- Запустили поиск по следующему $x_{max} = 56$.
- Идем спать....





День 2. Ищем следующий блок с gdb

Первая сессия:

- `select pg_backend_pid();`
 - `break errfinish`
`cont`
 - `bt full`
`cont`
-

Вторая сессия:

- `gdb -p PID`
- `select xmax, id from`
`files where xmax=56`
`limit 1;`

В stack ищем:

- `page` - абсолютный номер блока в таблице
- `bi_lo` - номер блока в файле
- `input_message` - запрос, приведший к падению



День 2. Смотрим stack gdb

```
Breakpoint 1, errfinish (dummy=0) at elog.c:411
411  elog.c: No such file or directory.
(gdb) bt full
#0  errfinish (dummy=0) at elog.c:411
```

```
tup = {t_len = 88, t_self = {ip_blkid = {bi_hi = 592, bi_lo = 3055}, ip_posid = 2}, t_tableOid = 16601,
      t_data = 0x7f5025a80c18)
ndeleted = 0
priorXmax = <optimized out>
htup = 0x7f5025a80c18
maxoff = 68
offnum = 2
```

Блок 3055

```
#9  0x00000000004bdfc7 in heapgettup_pagemode (scan=scan@entry=0x2108ef8, dir=<optimized out>, nkeys=0, key=0x0) at heapam.c:1065
tuple = 0x2108f48
backward = false
```

page = 38800367

base/16401/16601.296

```
finished = false
---Type <return> to continue, or q <return> to quit---
dp = 0x7f5025702c80 ""
lines = <optimized out>
lineindex = 0
lineoff = <optimized out>
linesleft = 0
lpp = <optimized out>
#10 0x00000000004bf01e in heap_getnextslot (sscan=0x2108ef8, direction=<optimized out>, slot=0x210c020) at heapam.c:1358
scan = 0x2108ef8
#11 0x000000000063bf2e in table_scan_getnextslot (slot=0x210c020, direction=ForwardScanDirection, sscan=<optimized out>)
at ../../../../src/include/access/tableam.h:889
No locals.
```

input_message = {data = 0x2008390 "select xmax, id from files where xmax=56 limit 1;", len = 50, maxlen = 1024,

cursor = 50}

```
opt = <optimized out>
status = <optimized out>
userDoption = <optimized out>
listen_addr_saved = true
i = <optimized out>
output_config_variable = <optimized out>
__func__ = "PostmasterMain"
#28 0x0000000000484dd3 in main (argc=3, argv=0x2002dd0) at main.c:228
```



День 2. И снова dd нас спасет

- “Зануляем” весь блок:

```
dd if=/dev/zero of=$PG_DATA/base/16401/16601.296 bs=8192 seek=3055  
count=1 conv=notrunc
```

- Проверяем с помощью pg_filedump:

```
pg_filedump -i ./base/16401/16601.296 | grep "XMAX: 56"
```

- Проверяем, с помощью pageinspect:

```
postgres=# select t_xmin, t_xmax, t_ctid FROM  
heap_page_items(get_raw_page('files',38800367)) WHERE t_xmax = 56;  
 t_xmin | t_xmax | t_ctid  
-----+-----+-----  
(0 rows)
```

День 2. MultiXactID has not been created yet

- Запускаем gdb;
- Найден следующий блок с номером ... 3056;

- Зануляем:

```
dd if=/dev/zero of=$PG_DATA/base/16401/16601.296 bs=8192 seek=3056  
count=1 conv=notrunc
```

- Проверяем с помощью pg_filedump:

```
pg_filedump -i ./base/16401/16601.296 | grep "XMAX: 1614807043"
```

- Проверяем, с помощью pageinspect;

День 2. Ходим по кругу

- Запускаем VACUUM FREEZE по таблице;
- Найден битый блок;
- Зануляем с помощью `dd`;
- Проверяем с помощью `pg_filedump`;
- Проверяем, с помощью `pageinspect`;
- Ищем новый битый блок.



День 2. Итоги

- Найдены еще 4 битых блока [3055 - 3058](#)
- Снова запускаем VACUUM FREEZE по таблице;
- Идем спать ...



День 3. Виден свет в конце тунеля

- VACUUM FREEZE по таблице завершился без ошибок;
- Запускаем VACUUM FREEZE по всей БД;
- Меняем настройки autovacuum-а для всего инстанса и отдельно для самой большой таблицы (гораздо агрессивнее);
- Понижаем autovacuum_freeze_max_age = 500000000 (!!!);
- Включаем checksum-ы;
- Прodelываем все тоже самое с Standby сервером;

-  [Пишем большой документ с рекомендациями.](#)

Спустя почти 6 месяцев...



День 1. Дежавю

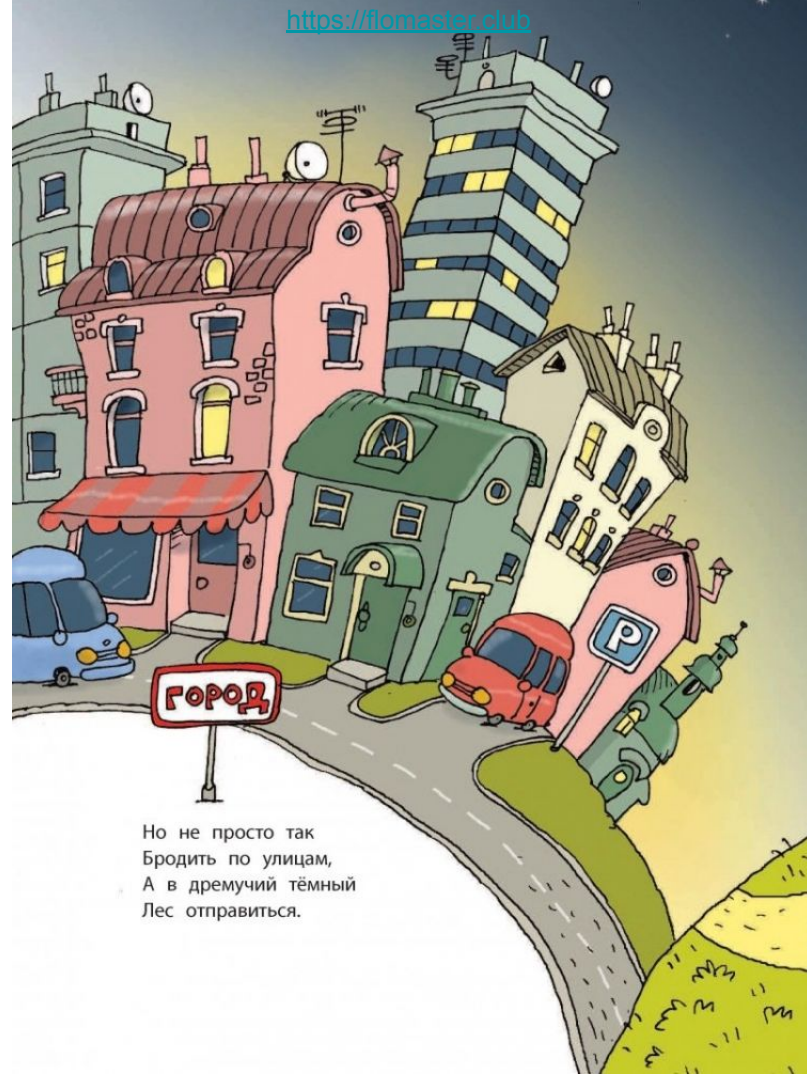
- Другой клиент нашего клиента;
- ~ 2 дня down time;
- PostgreSQL 12.2, БД ~3.5 TB, RAM 128 GB;
- По всем признакам проблема с переполнением счетчика транзакций (wraparound)
- Техническая команда пыталась решить проблему своими силами (прочитали весь Stackoverflow);
- VACUUM FREEZE не помогает.

Нет приятнее заняться,
Чем в носу поковырять.
Всем ужасно интересно,
Что там спрятано внутри.
А кому смотреть противно,
Тот пускай и не глядит.
Мы же в нос к нему не лезем,
Пусть и он не пристаёт.



День 1. Подключаемся

- Физического доступа к серверам клиента нет
- 2 сервера PostgreSQL: primary и standby;
- Оба сервера уже в single user mode;
- Репликация не отключена;
- Backup + PITR есть;
- Checksum-ы выключены.





День 1. Что то новенькое

- Проверяем репликационный слот - ОК;
- Ищем relation с самым старым XID: `age(pg_class.relFrozenxid):`

pg_temp.pg_temp_42.pg_temp_275833	2146483648	1023109592	-1146483648	8192 bytes	
pg_temp_42.parentalist	2146483648	1023109592	-1146483648	16 kB	
filen	1013774761	461264332	-7764332	2117 us	2023-05-22 04:17:13.335169-04

- Временная схема `pg_temp` !!!
- Возраст XID максимальный
- `autovacuum_freeze_max_age = 1000000000` !!!
- Autovacuum не приходил

Вредные советы. № 4

Если вам рекомендуют,
Или делают советы,
То не вздумайте, конечно,
Поспешить их выполнять!

Лучший способ убедиться,
Что советуют плохое -
Это ничего не делать,
Ведь работает и так!





День 1. Временные объекты

Проверяем сколько их в каталоге:

```
-----  
1: ?column? = "DROP TABLE pg_temp_76.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_76.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_174.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_174.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_177.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_177.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_169.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_169.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_211.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_211.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_213.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_213.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_210.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_210.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_216.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_216.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_217.parentslist;"      (typeid = 25, len = -1, typmod = -1, byval = f)  
-----  
1: ?column? = "DROP TABLE pg_temp_217.parentslist_guid_device_guid_version_key;"      (typeid = 25, len = -1, typmod = -1, byval = f)
```



День 1. Проблемы с temp tables

- Application использует временные таблицы для формирования list в хранимых функциях;
- Сессии персистентные -> временные объекты не удаляются;
- В явном виде временные таблицы не удаляются application;
- В одной из временных таблиц “залип” самый старый XID;
- После перехода single user mode, таблицы не удалились;
- VACUUM FREEZE не может обслужить другие таблицы.

Вредные советы. № 4

Надо временных объектов,
Создавать как можно больше,
И держать к БД коннекты,
Персистентные всегда!

Эти правила друзья.
Очень громко повторяйте,
Их тому, кто пишет код,
В базе у себя.





День 1. Прогоняем VACUUM FREEZE

После выполнения VACUUM FREEZE проверяем самый старый XID:

```
backend> WITH max_age AS (SELECT 2000000000 as max_old_xid, setting AS autovacuum_freeze_max_age FROM pg_c
max_old_xid::int, m.autovacuum_freeze_max_age::int, age(d.datfrozenxid) AS oldest_current_xid FROM pg_cata
t_current_xid, max(ROUND(100*(oldest_current_xid/max_old_xid::float))) AS percent_towards_wraparound, max(
OM per_database_stats;
  1: oldest_current_xid (typeid = 23, len = 4, typmod = -1, byval = t)
  2: percent_towards_wraparound (typeid = 701, len = 8, typmod = -1, byval = t)
  3: percent_towards_emergency_autovac (typeid = 701, len = 8, typmod = -1, byval = t)
----
  1: oldest_current_xid = "1451724364" (typeid = 23, len = 4, typmod = -1, byval = t)
  2: percent_towards_wraparound = "73" (typeid = 701, len = 8, typmod = -1, byval = t)
  3: percent_towards_emergency_autovac = "145" (typeid = 701, len = 8, typmod = -1, byval = t)
----
```

Возраст XID в БД изменился на 1451724364 (73%)



День 1. Проверяем XID по DB

```
backend> SELECT datname, age(datfrozenxid), current_setting('autovacuum_freeze_max_age') FROM pg_database ORDER BY 2 DESC;
 1: datname      (typeid = 19, len = 64, typmod = -1, byval = f)
 2: age (typeid = 23, len = 4, typmod = -1, byval = t)
 3: current_setting (typeid = 25, len = -1, typmod = -1, byval = f)
----
 1: datname = "template1"      (typeid = 19, len = 64, typmod = -1, byval = f)
 2: age = "1451724364" (typeid = 23, len = 4, typmod = -1, byval = t)
 3: current_setting = "1000000000" (typeid = 25, len = -1, typmod = -1, byval = f)
----
 1: datname = "template0"      (typeid = 19, len = 64, typmod = -1, byval = f)
 2: age = "1451724364" (typeid = 23, len = 4, typmod = -1, byval = t)
 3: current_setting = "1000000000" (typeid = 25, len = -1, typmod = -1, byval = f)
----
 1: datname = "postgres"      (typeid = 19, len = 64, typmod = -1, byval = f)
 2: age = "0" (typeid = 23, len = 4, typmod = -1, byval = t)
 3: current_setting = "1000000000" (typeid = 25, len = -1, typmod = -1, byval = f)
----
```

Самый старый XID с возрастом 1451724364 (73%) в БД template1 и template0 !!!



День 1. Обслуживаем template1

```
(postgres@[local]:5432) [template1] > SELECT datname, age(datfrozenxid), c
current_setting('autovacuum_freeze_max_age') FROM pg_database ORDER BY 2 DE
SC;
 datname | age | current_setting
-----+-----+-----
 postgres |  0 | 1000000000
 template1 |  0 | 1000000000
 template0 |  0 | 1000000000
(3 rows)

Time: 2.889 ms
(postgres@[local]:5432) [template1] >
```

Возраст XID изменился на 0

Спустя 3 недели ...



День 1. Уже даже не дежавю

- Другой клиент нашего клиента;
- ~ 2 дня down time;
- PostgreSQL 12.2, БД ~11 TB, RAM 128 GB;
- По всем признакам проблема с переполнением счетчика транзакций (wraparound)
- Техническая команда пыталась решить проблему своими силами (прочитали весь Stackoverflow);
- VACUUM FREEZE не помогает.

Нет приятнее занятыя,
Чем в носу поковырять.
Всем ужасно интересно,
Что там спрятано внутри.
А кому смотреть противно,
Тот пускай и не глядит.
Мы же в нос к нему не лезем,
Пусть и он не пристаёт.



Продолжение следует ...



Вредные советы. № 5

Вот проблема за проблемой.
Каждый раз одно и то же!
Может нужно базу данных,
Вам скорее заменить?!

Взять чего-то современной.
Модно, стильно, молодежно.
Все в хранить “key-value”-виде,
Или лучше в файл писать.



Вместо **заклучения**

Добавление 64-bit XIDs в PostgreSQL core?

или

Будущее за Table Access Methods с 64-bit XIDs?



Спасибо!

info@tantorlabs.ru

+7 495 787 51 78

tantorlabs.ru

