PG BootCamp Russia 2025 Ekaterinburg
PGBootCamp.ru

Xid, отладка, 2 фиксá.

Разработка и отладка системы 64-битной идентификации транзакций

Евгений Воропаев, старший разработчик «Тантор Лабс»

Немного обо мне

- Кандидат физ.-мат. наук (физика конденсированного состояния вещества).
- Занимаюсь разработкой Postgres с 2023 г.
- Вклад в сообщество:
  - Xid64 для PG18dev
  - Bootstrap SLRU (рефакторинг, Ready for committer)
- Хобби: альпинизм, гитара, IT-стартапы.

Автореферат диссертации

Xid64 для PG18dev

Профессионально занимаюсь разработкой ПАК с 2006 года.

- 2006-2007 - охранно-пожарные системы
- 2006 -2010 - комплексы оптической спектрометрии
- **2010 - 2015 - противоугонные системы “Microlock”**
- 2014 - 2018 - АСУТП и РК
- 2018 - 2020 - интернет вещей в освещении
- **2020 - 2021 - автоматический спортивный хронометража “Торок”**
- 2021 - 2022 - контроллеры доступа “Apollo”
- 2022 - 2023 - система виртуализации “Брест”
- с 2023 - разработка СУБД Postgres/Tantor

План на следующие полчаса

О чём поговорим. Теория.

1. Термины
2. 32-битная система идентификации транзакций
3. 64-битная система идентификации транзакций

Чему научимся. Практика.

1. «Пришла беда откуда ЖДАЛИ»
   Включение расширенных тестов при репликации
2. «Ну, всё понятно! Но что конкретно?»
   Проблемы консистентности страниц при PRUNE/FREZE
3. «Сапожник без сапог?»
   Отладка без отладчика
4. «Вскрытие покажет!»
   Анализ багов
5. «Staying alive!»
   Ремонт

Термины

Системы идентификации транзакций

Счётчик транзакций

Идентификатор транзакций

64-битный идентификатор транзакций

Xid64

Xid32

TransactionID

32-битный идентификатор транзакций

FullTransactionID

ShortTransactionID

Термины

Система идентификации транзакций

это набор
- алгоритмов,
- типов переменных
- и самих переменных,

реализующий
- уникальную идентификацию транзакций
- и функционал определения порядка следования транзакций во времени.

Счётчик транзакций

алгоритм, реализующий функционал получения следующего свободного идентификатора транзакций.

Идентификатор транзакции

число, однозначно идентифицирующее транзакцию. Также этот термин используем для переменной, хранящей это число.

Идентификатор транзакции

В оригинальном Postgres используются два типа идентификаторов транзакций:

TransactionID
идентификатор транзакции, 32 - битный

FullTransactionID
полный идентификатор транзакции, 64-битный

Какой тип применяется при определении порядка следования транзакций?

Какой тип используется в кортежах?

Что инкрементируется в счётчике транзакций?

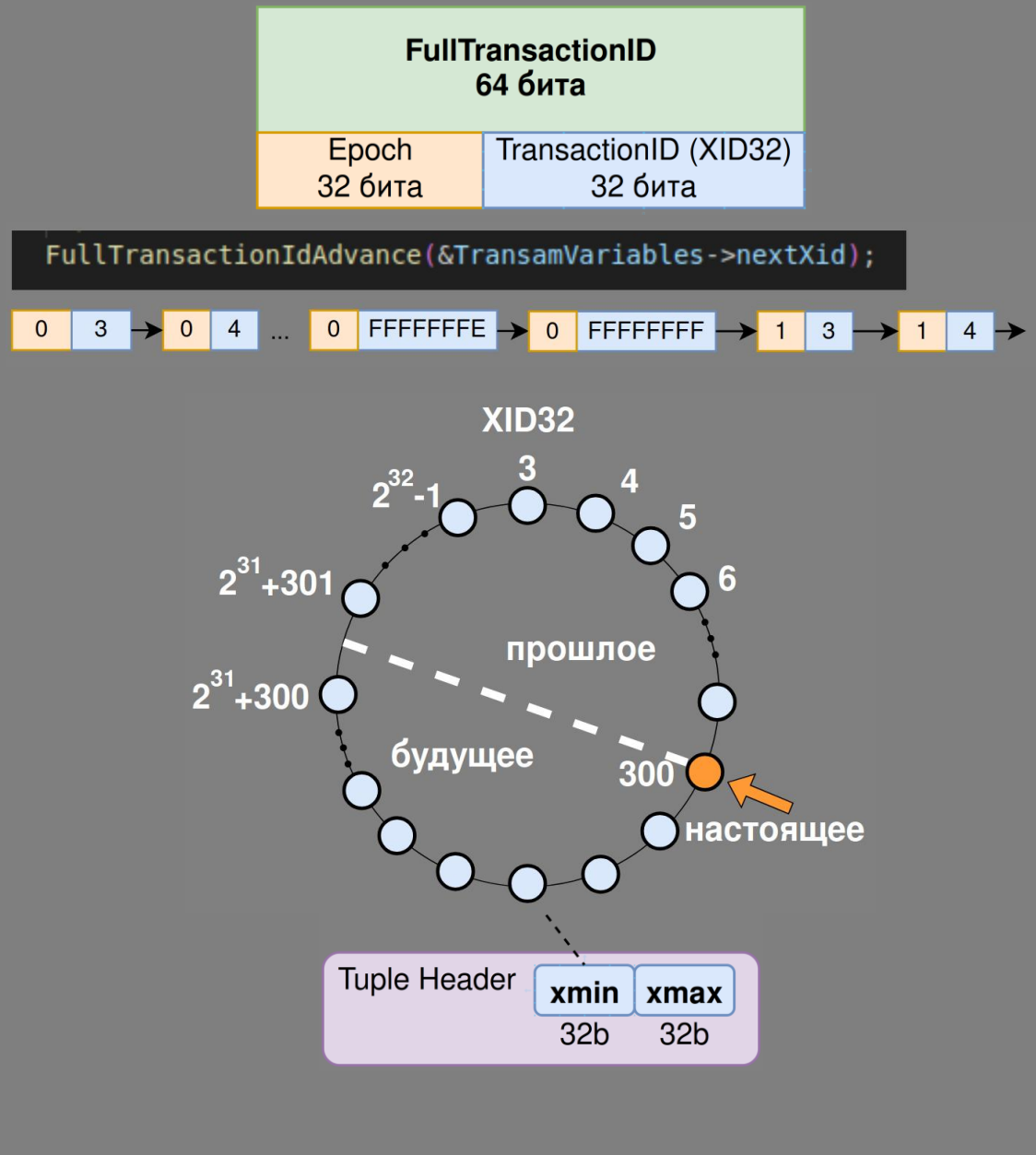
Счётчик транзакций **VS** Идентификатор транзакции

TransactionID (XID32)
идентификатор транзакции. 32-битный

FullTransactionID
полный идентификатор транзакции
64-битный

Состав счётчика транзакций
1) переменная-счётчик (64b)
2) вспомогательные переменные
3) алгоритм инкремента

Где хранится/используется Xid32?
- в самой транзакции
- в кортежах!



Wraparound

Причины:

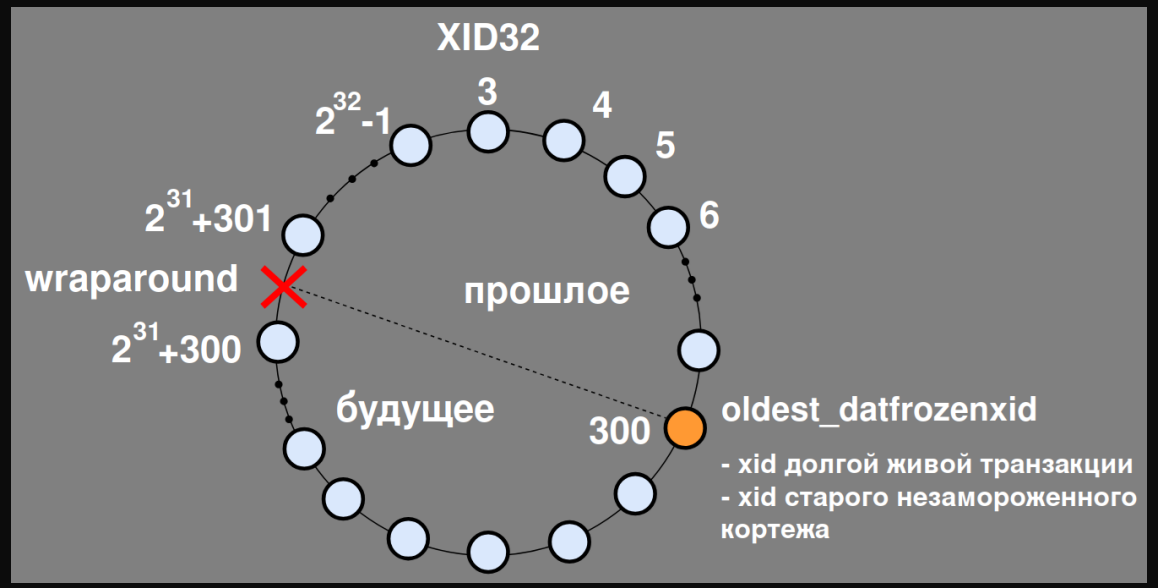
- долгие транзакции
- необработанные VACUUM-ом кортежи

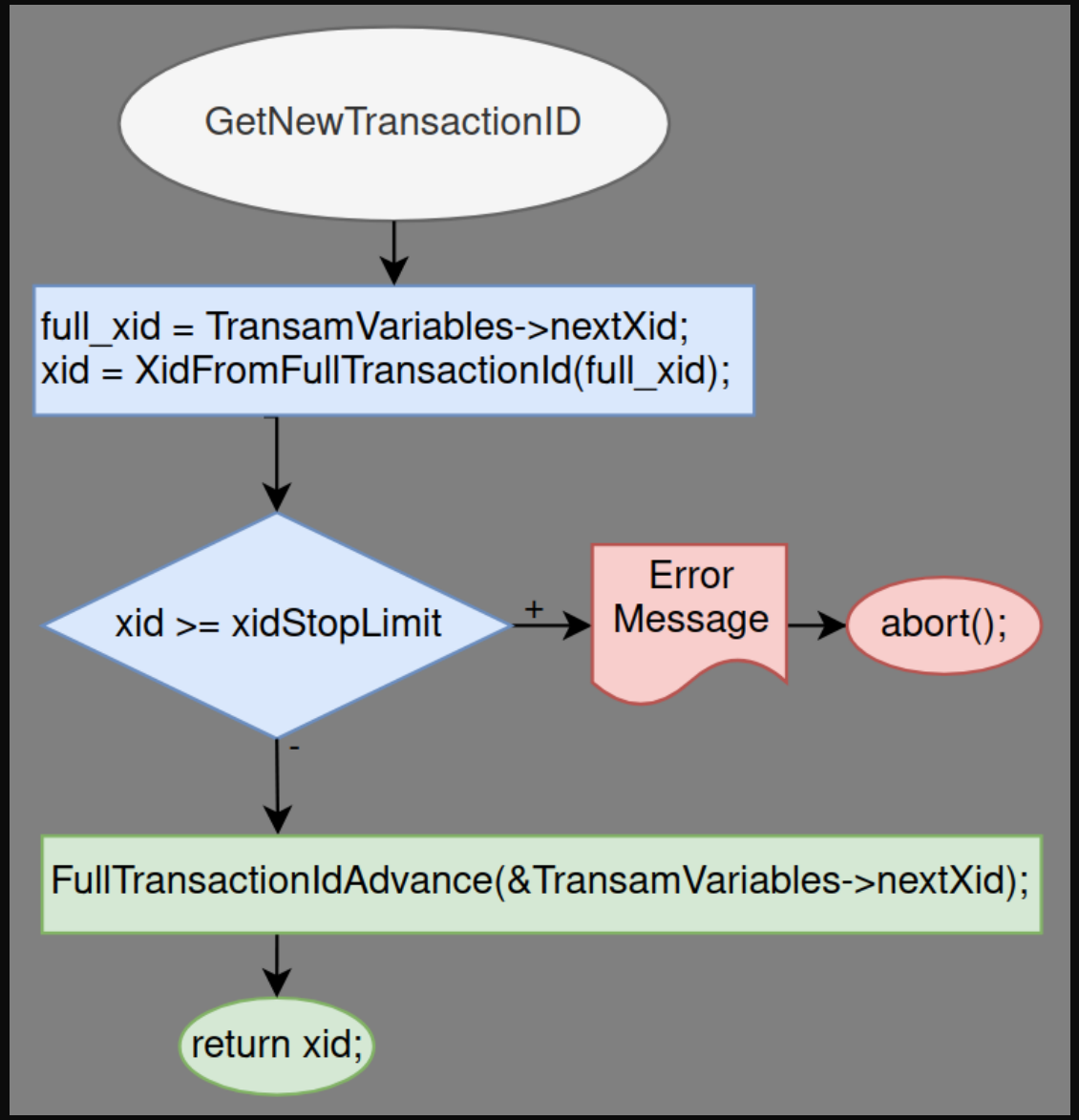
Итог - остановка кластера.

Оценка периодичности проблемы:

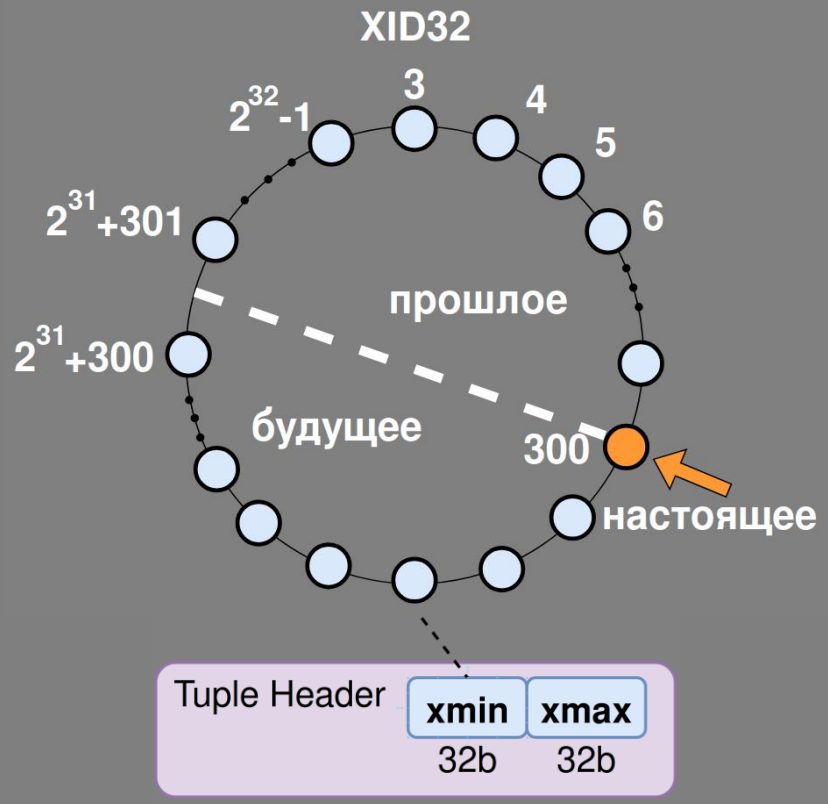
Xid32: $2^{32}-3$ = 4 294 967 293 идентификаторов транзакций

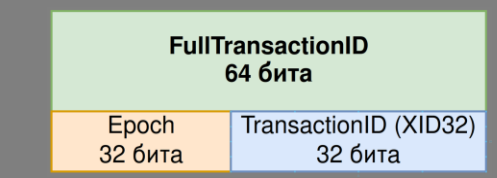
20 000 транз/с => 2,5 сут => Wraparound



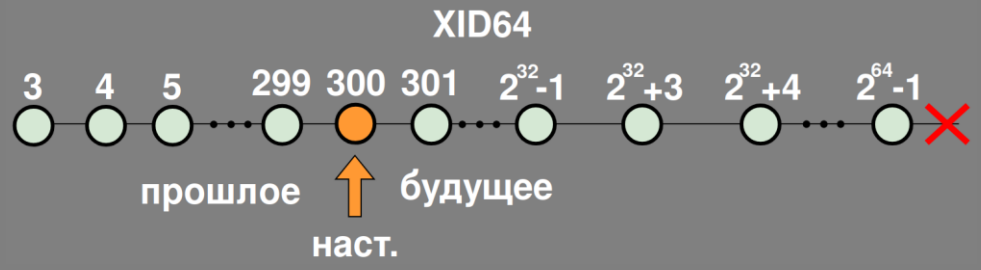


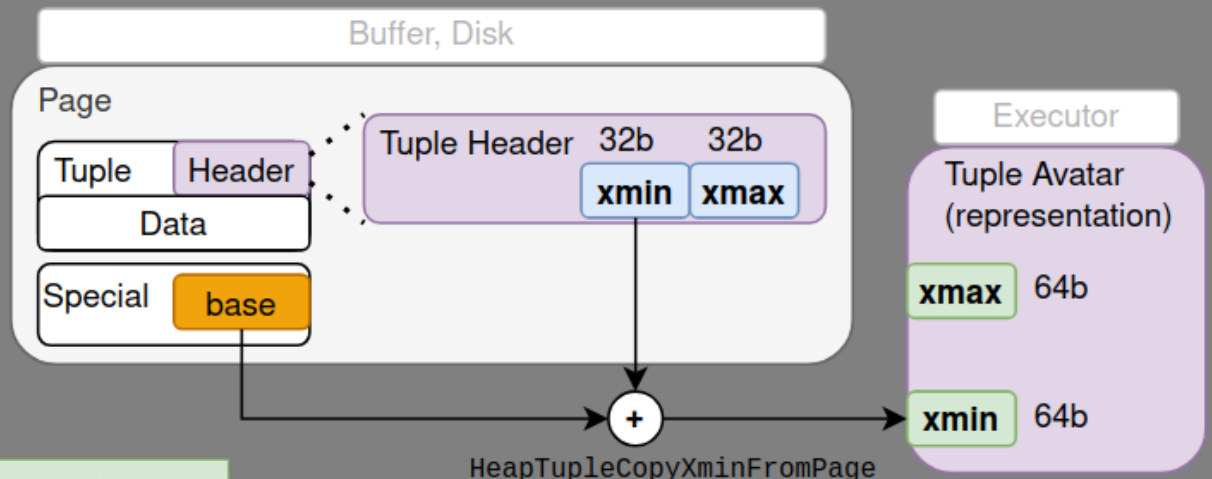
Xid32



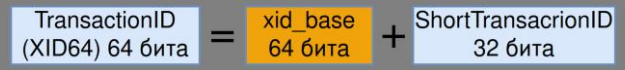


Xid64






```
HeapTupleCopyXminFromPage
HeapTupleCopyXmaxFromPage

HeapTupleHeaderStoreXmax
HeapTupleHeaderStoreXmin
```



Доработка патча xid64 для текущего мастера

Дано:

1) патч версии 52, совместимый с master-веткой где-то между PG16 и PG17.

2) master-ветка версии PG18dev

3) фрагменты, вынесенные в отдельные патчи GUC64, MXIDOFF64 и др.

Задача:

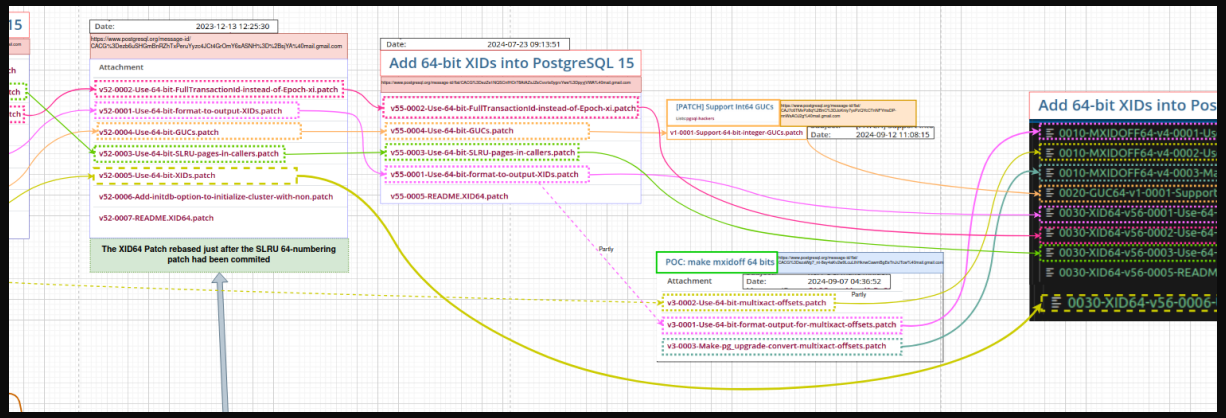
реализовать XID64 для текущего PG18dev

Сложности (формальные):

- запутанное генеалогическое древо патча;
- существенные изменения кодовой базы PG.

Сложности (функциональные):

- **изменение WAL-записей prune и freeze;**
- баги версии 52;
- и мн. др.

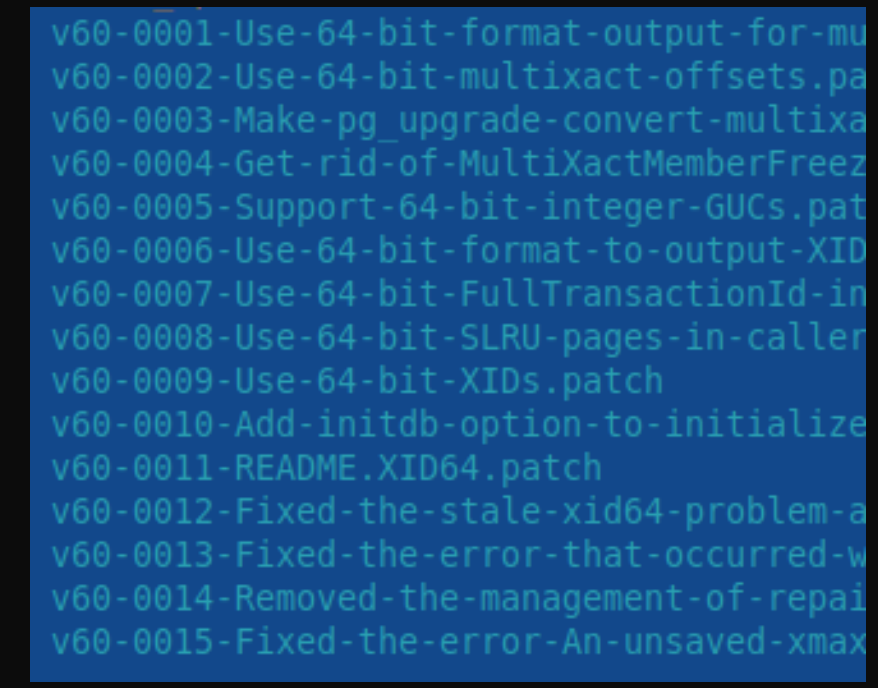


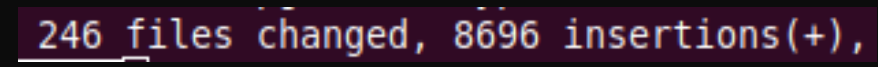

```
evoro:lin-ev:~/tantor/dv/pg18xid64PS(ps/pg18xid64)$ git log
commit f83d709760d84253b539ee454ce9ce1ceb9ad9ac
Author: Heikki Linnakangas <heikki.linnakangas@iki.fi>
Date:   Mon Mar 25 14:59:58 2024 +0200

    Merge prune, freeze and vacuum WAL record formats
```

```
evoro:lin-ev:~/tantor/dv/pg18xid64PS(ps/pg18xid64)$ git
commit 6dbb490261a6170a3fc3e326c6983ad63e795047
Author: Heikki Linnakangas <heikki.linnakangas@iki.fi>
Date:   Wed Apr 3 19:32:28 2024 +0300

    Combine freezing and pruning steps in VACUUM
```

Доработка патча xid64 для текущего мастера

1. пошагово применяем патч за патчем (11 патчей)
2. большие патчи разделяем на мелкие фрагменты и также пошагово применяем (>200 файлов)
   - устраняем формальные несоответствия кода
   - сложные изменения базы пытаемся проанализировать и адаптировать вдумчиво (= закладываем новые баги)
3. собираем, устраняем ошибки сборки
4. тестируем
5. если есть ошибки в тестах, отлаживаем, устраняем ошибки, возвращаемся в пункт 3
6. **видим сообщение “All test successful!”**
7. **не видим сообщение “Error”**
8. **Радуемся!** 😀
9. **Вспоминаем, что не включили “wal_check_consistency”** 😳😟😟





Готовимся к практике

На вашем компьютере НАСТРОЕНО окружение для сборки и тестирования текущего master (PG18dev)

https://github.com/e-voro/pg18xid64v60_ekat_practice

git clone https://github.com/e-voro/pg18xid64v60_ekat_practice

Практика. Материалы

Из жизни страниц

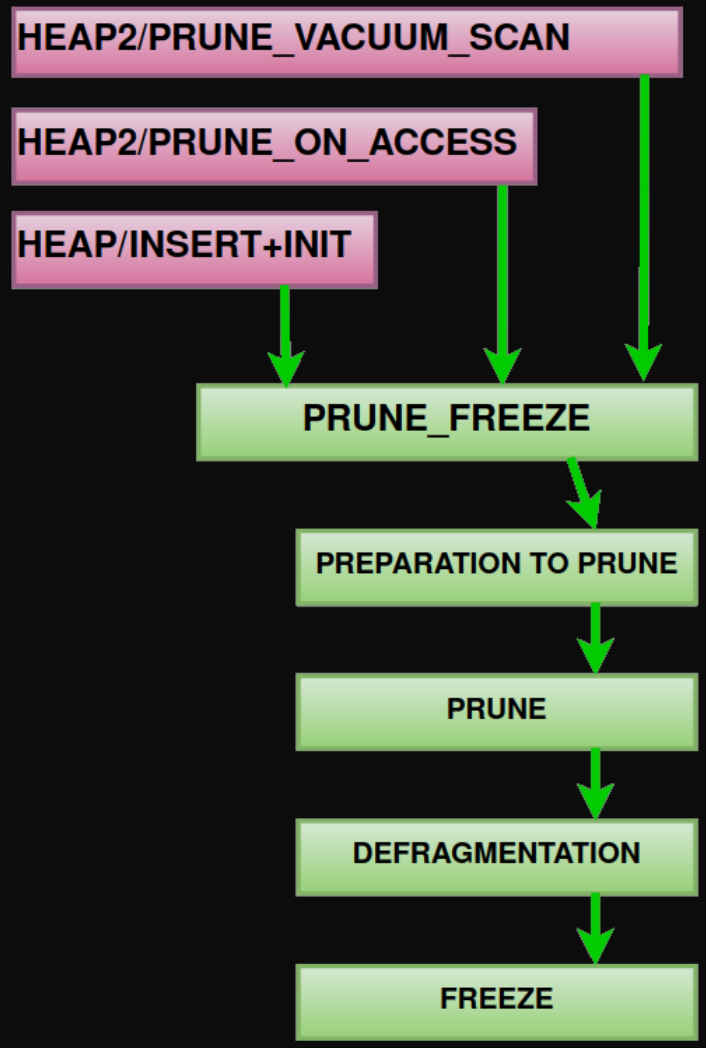
Розовым - операции уровня Resource Manager

Зелёным - низкоуровневые операции

PRUNE - умерщвление, удаление, перенаправление кортежей

DEFRAGMENTATAION - восстановление фрагментации страницы

FREEZE - замораживание кортежей



Практика. Начало

cd pg18xid64v60_ekat_practice

git checkout ABC

Имеем патч Xid64, доработанный для PG18dev, содержащий исправления ряда багов.

Запускаем тест 027_stream_regress

./tst0.sh

“All test successful!” 😀

```
evoro:lin-ev:~/tantor/dv/pg18xid64v60_ekat_practice(pg18xid64v60_ekat_practice)$ ./tst0.sh
============================================================
BALOO: STARTED TESTING. NO WAL consistency checking.
============================================================
++ make -C src/test/recovery/ check PROVE_TESTS=t/027_stream_regress.pl
...
# +++ tap check in src/test/recovery +++
t/027_stream_regress.pl .. ok
All tests successful.
Files=1, Tests=9, 42 wallclock secs ( 0.01 usr  0.00 sys +  2.07 cusr  1.53 csys =  3.61 CPU)
Result: PASS
...
============================================================
BALOO: FINISHED TESTING. NO WAL consistency checking.
============================================================
```

Практика. Баг 0.

Запускаем тот же тест 027_stream_regress, но с проверкой консистентности.

./tst1.sh

```
make -C "src/test/recovery/" \
      check PROVE_TESTS="t/027_stream_regress.pl" \
      PG_TEST_EXTRA="wal_consistency_checking"
```

Ctrl+C

Был бы Postgres – приключения на него найдутся!

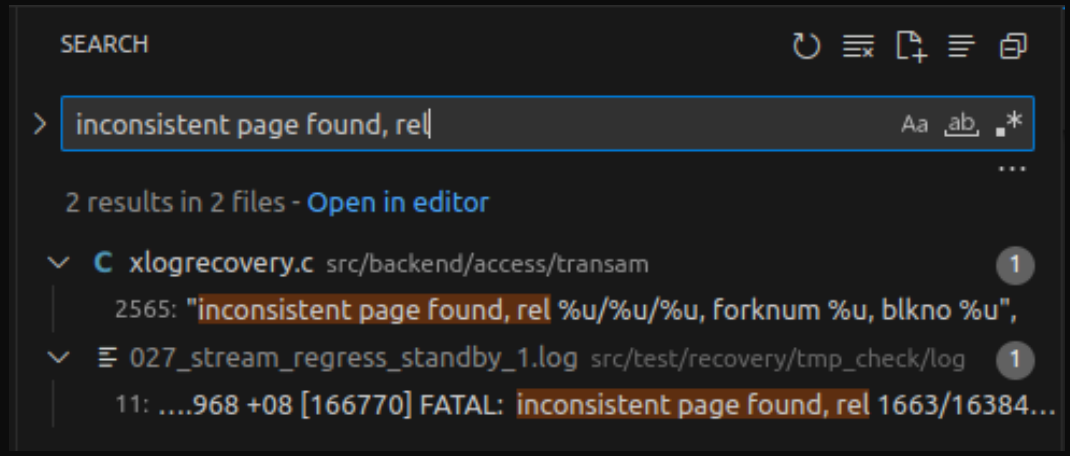
Имеем **неконсистентное состояние страницы на стороне реплики**.

```
LOG:  started streaming WAL from primary at 0/3000000 on timeline 1
FATAL:  inconsistent page found, rel 1663/16384/1259, forknum 0, blkno 0
CONTEXT:  WAL redo at 0/3490DF0 for Heap/INSERT+INIT: off: 0, flags: 0x00;
LOG:  startup process (PID 166770) exited with exit code 1
```

Баг 0. Поиск источника сообщения

Текстовым поиском выясняем место в коде, откуда мы получили сообщение об ошибке.

Это функция **verifyBackUpConsistency**



Откуда звук?

src > backend > access > transam > C xlogrecovery.c > verifyBackupPageConsistency(XLogReaderState *)

```
2470    {
2484        {
2561            /* Time to compare the primary and replay images. */
2562            if (memcmp(replay_image_masked, primary_image_masked, BLCKSZ) != 0)
2563            {
2564                elog(FATAL,
2565                     "inconsistent page found, rel %u/%u/%u, forknum %u, blkno %u",
2566                     rlocator.spcOid, rlocator.dbOid, rlocator.relNumber,
2567                     forknum, blkno);
2568            }
2569        }
2570    }
```

Баг 0. Логгеры

primary_image ≠replay_image

Но что именно не равно?

Добавляем логгеры.

git checkout DEF

Запускаем сборку и тест.

./bld.sh

./tst1.sh

Останавливаем тест.

Ctrl+C

Извлекаем лог страницы из логов standby-сервера.

./pagelog_standb.sh

Вскрытие покажет!

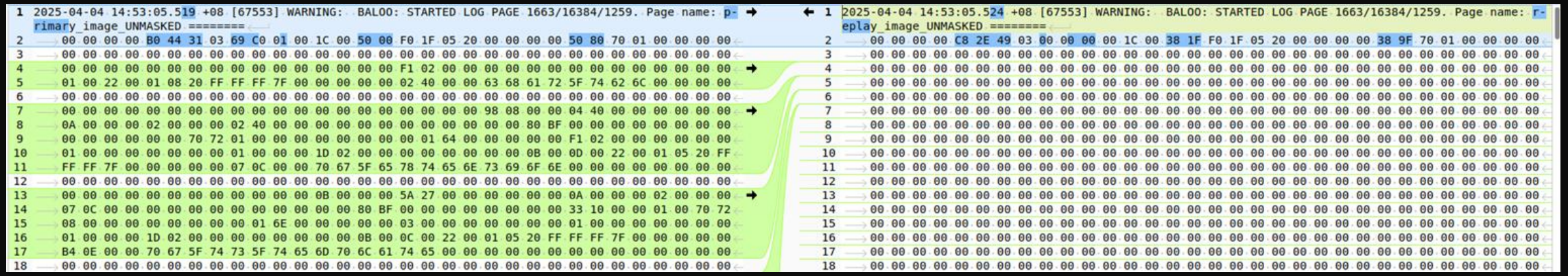
```
static inline void baloo_log_page_X(RelFileLocator locator, Page page, const char* page_name)
{
#define BALOO_X_SPC 1663U
#define BALOO_X_DB  16384U
#define BALOO_X_REL 1259U

    if (locator.spcOid == BALOO_X_SPC
        && locator.dbOid == BALOO_X_DB
        && locator.relNumber == BALOO_X_REL)
    {
        StringInfo str;
        str = makeStringInfo();
        appendStringInfo(str,
            "BALOO: STARTED LOG PAGE %u/%u/%u. Page name: %s ========\n",
            BALOO_X_SPC, BALOO_X_DB, BALOO_X_REL, page_name);

        baloo_log_page(str, page);
        baloo_log_backtrace(str);

        appendStringInfo(str,
            "BALOO: FINISHED LOG PAGE %u/%u/%u. Page name: %s ========",
            BALOO_X_SPC, BALOO_X_DB, BALOO_X_REL, page_name);

        elog(WARNING, "%s", str->data);

        destroyStringInfo(str);
    }
}
```

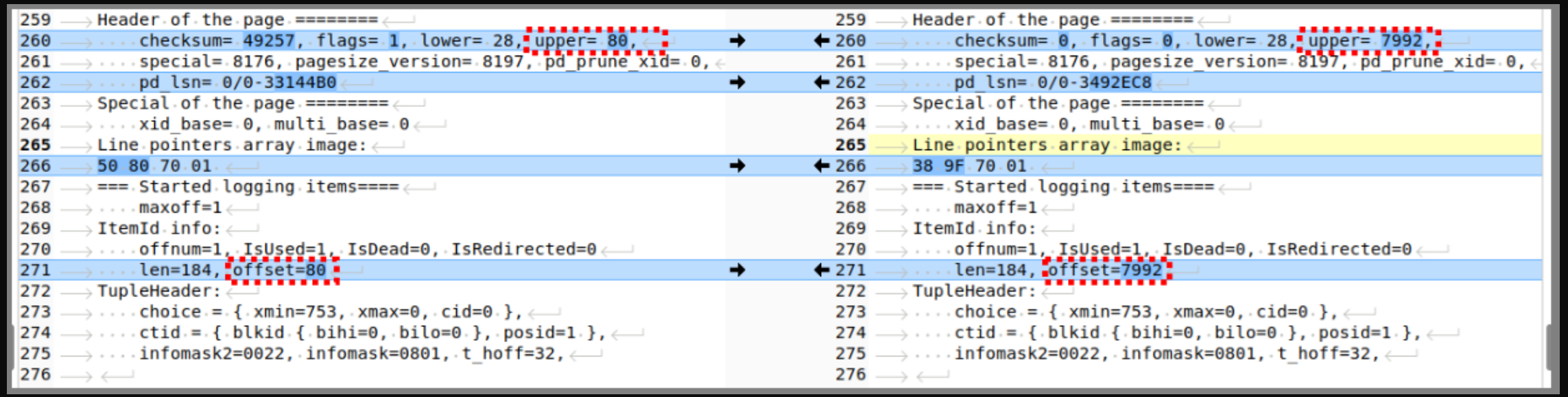
```
evoro:lin-ev:~/tantor/dv/pg18xid64v60_ekat_practice[774d1d398a6]$ ./pagelog_standb.sh
=======================================================
BALOO: STARTED EXTACTING PAGES FROM PRIMARY LOGs
=======================================================
++ ./pagelogextractor.pl src/test/recovery/tmp_check/log/027_stream_regress_standby_1.log
Extract a page N=1 with pagename=primary_image_UNMASKED. File: ./pagelog/20250405-163357138-N00001_primary_image_UNMASKED
Extract a page N=2 with pagename=replay_image_UNMASKED. File: ./pagelog/20250405-163357139-N00002_replay_image_UNMASKED
Extract a page N=3 with pagename=primary_image_MASKED. File: ./pagelog/20250405-163357141-N00003_primary_image_MASKED
Extract a page N=4 with pagename=replay_image_MASKED. File: ./pagelog/20250405-163357143-N00004_replay_image_MASKED
Extract a fatality message N=5. File: ./pagelog/20250405-163357143-N00005_FATAL
++ set +x
=======================================================
BALOO: FINISHED EXTACTING PAGES FROM PRIMARY LOGs
-------------------------------------------------------
```

Баг 0. Сравниваем страницы

meld pagelog/1primary_image_UNMASKED pagelog/2replay_image_UNMASKED

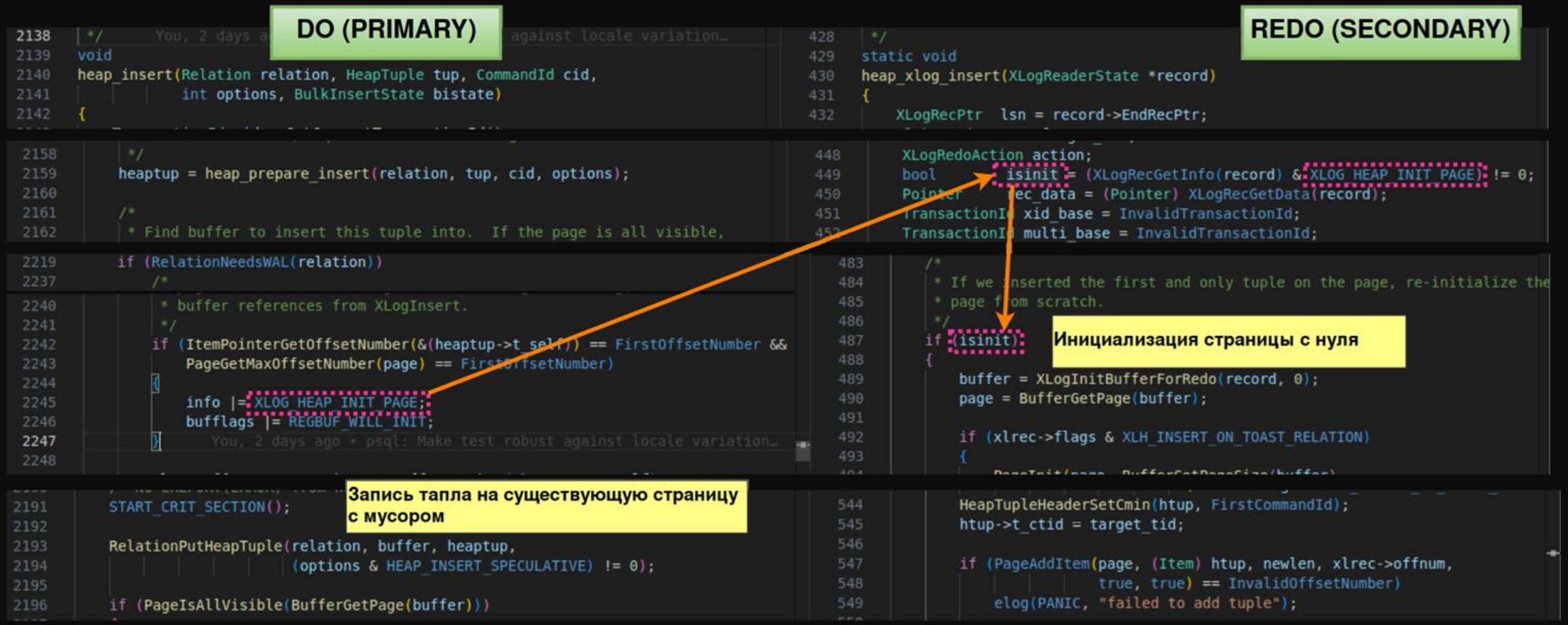
Вскрытие покажет!





Баг 0. Сравниваем код функций, производящих страницы

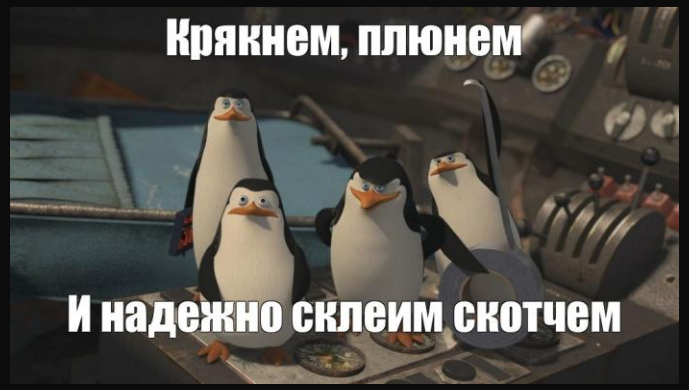
Вскрытие покажет!



Баг 0. Заплатка

Добавляем логику, определяющую, была ли страница чистой до вставки тапла.

git checkout GHI



AFTER THE FIX

```
2139  void
2140  heap_insert(Relation relation, HeapTuple tup, CommandId cid,
2141              int options, BulkInsertState bistate)
2142  {
2143      TransactionId xid = GetCurrentTransactionId();
2144      HeapTuple   heaptup;
2145      Buffer      buffer;
2146      Buffer      vmbuffer = InvalidBuffer;
2147      bool        all_visible_cleared = false;
2148      PageHeader  pageheader;
2149      bool        were_not_tuples;      You, 2 hours ago • GHI. Bug0 is fixed. A conc
2150
2196      were_not_tuples = pageheader->pd_special == (uint32) pageheader->pd_upper;
2197
2242          /*
2243           * If this is the single and first tuple on page, we can reinit the
2244           * page instead of restoring the whole thing.  Set flag, and hide
2245           * buffer references from XLogInsert.
2246           */
2247          if (ItemPointerGetOffsetNumber(&(heaptup->t_self)) == FirstOffsetNumber &&
2248              PageGetMaxOffsetNumber(page) == FirstOffsetNumber && were_not_tuples)
2249          {
2250              info |= XLOG_HEAP_INIT_PAGE;
2251              bufflags |= REGBUF_WILL_INIT;
2252          }
```

Баг 1. Это prune

git checkout GHI
./bld.sh

Запускаем тест.
./tst1.sh

Останавливаем тест.
Ctrl+C

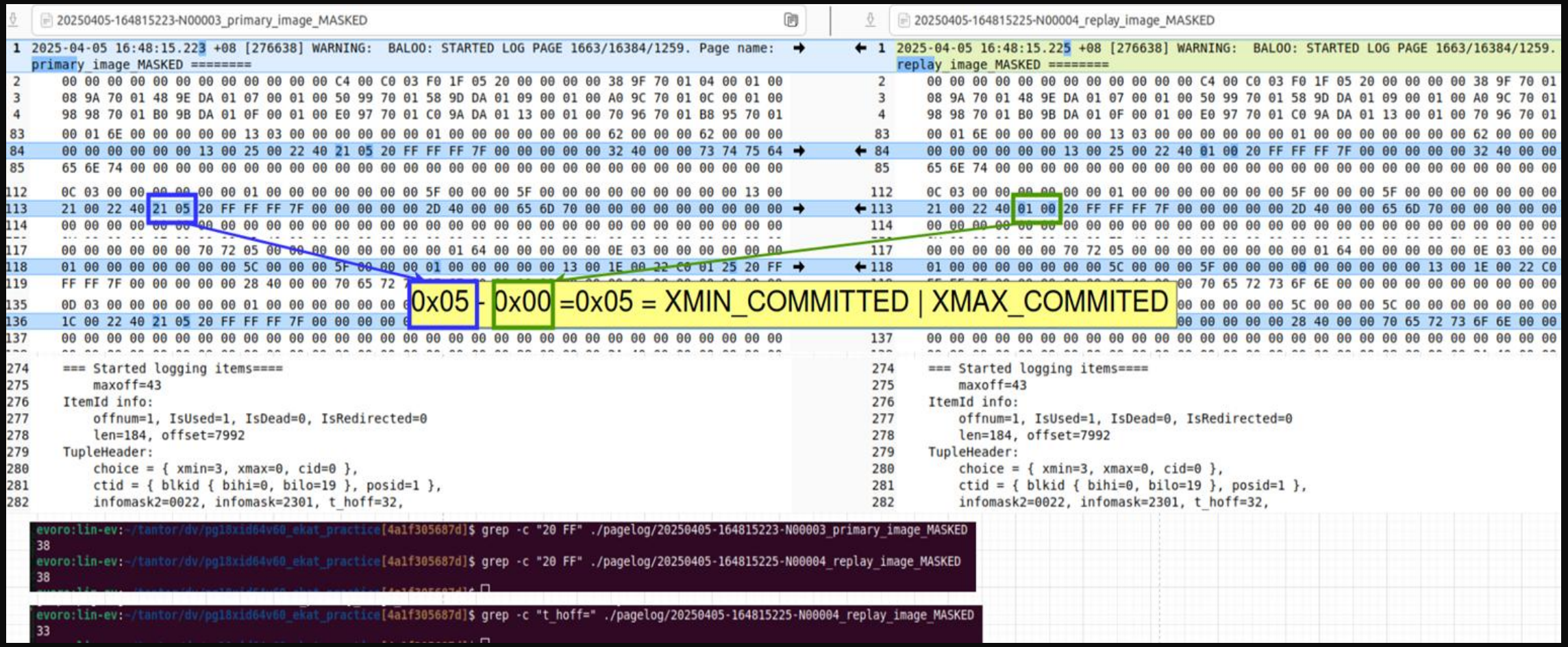
Извлекаем лог страницы из логов standby-сервера.
./pagelog_standb.sh

```
2025-04-05 16:48:15.225 +08 [276638] FATAL:  inconsistent page found, rel
1663/16384/1259, forknum 0, blkno 19
2025-04-05 16:48:15.225 +08 [276638] CONTEXT:  WAL redo at 0/7E2A378 for
Heap2/PRUNE_ON_ACCESS: snapshotConflictHorizon: 782, isCatalogRel: F,
nplans: 0, nredirected: 4, ndead: 0, nunused: 1, redirected: [16->19,
25->30, 29->33, 34->37], unused: [28]; blkref #0: rel 1663/16384/1259, blk
19 FPW
```

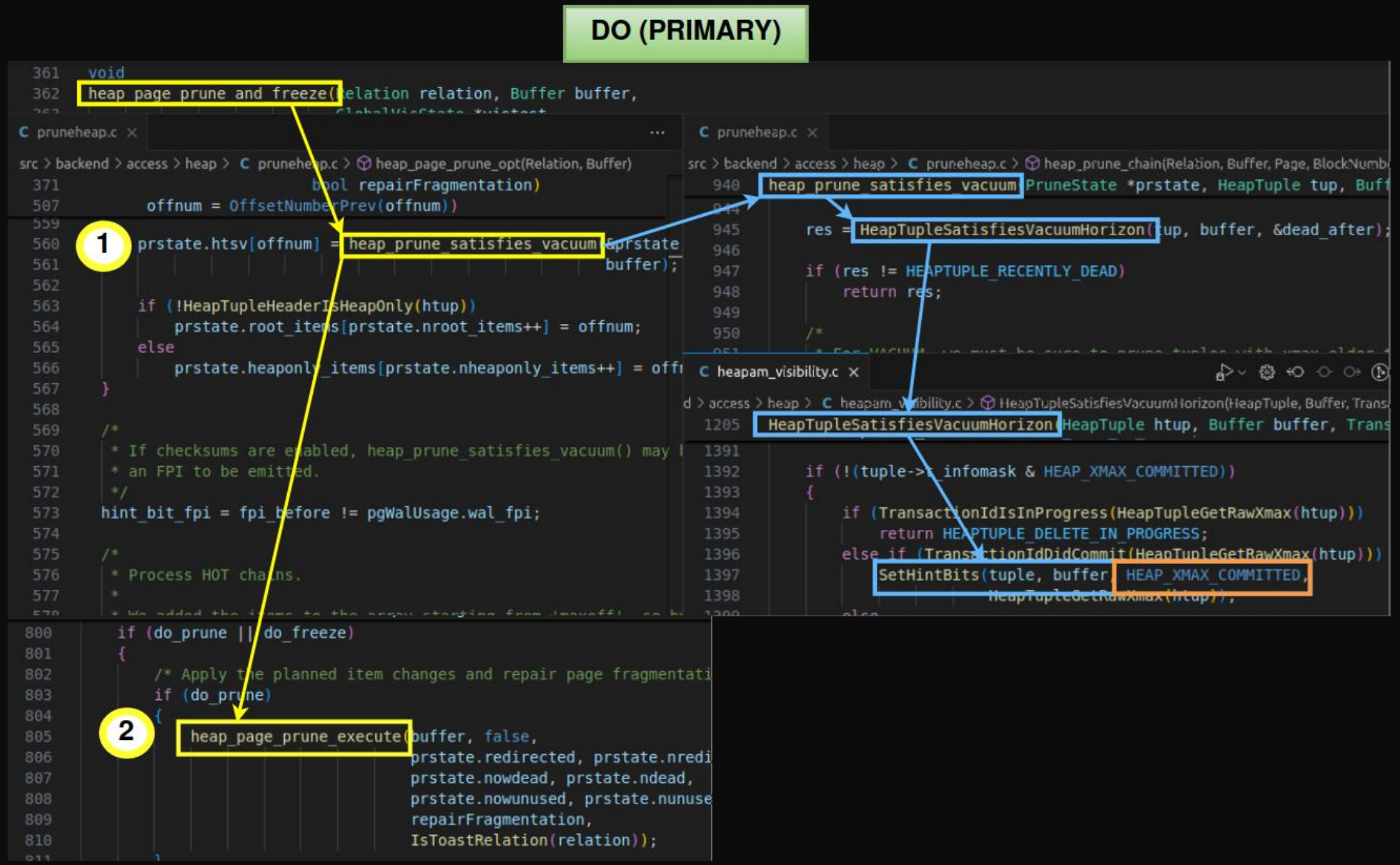
```
evoro:lin-ev:~/tantor/dv/pg18xid64PS(ps/pg18xid64)$ git log
commit f83d709760d84253b539ee454ce9ce1ceb9ad9ac
Author: Heikki Linnakangas <heikki.linnakangas@iki.fi>
Date:   Mon Mar 25 14:59:58 2024 +0200

    Merge prune, freeze and vacuum WAL record formats
```

Это prune!
Ну, всё понятно!
Но что конкретно?

Баг 1. «Вскрытие покажет». Анализ



Баг 1. Prune на стороне DO



Баг 1. Prune на стороне REDO

REDO (SECONDARY)

ckend > access > heap > C heapam_xlog.c > heap_xlog_prune_freeze(XLogReaderState *)

```
heap_xlog_prune_freeze(XLogReaderState *record)
    if (action == BLK_NEEDS_REDO)

        heap_xlog_deserialize_prune_and_freeze(dataptr, xlrec.flags,
                                               &nplans, &plans, &frz_offsets,
                                               &nredirected, &redirected,
                                               &ndead, &nowdead,
                                               &nunused, &nowunused);

        /*
         * Update all line pointers per the record, and repair fragmentation
         * if needed.
         */
        if (nredirected > 0 || ndead > 0 || nunused > 0)
            heap_page_prune_execute(buffer,
                                    (xlrec.flags & XLHP_CLEANUP_LOCK) == 0,
                                    redirected, nredirected,
                                    nowdead, ndead,
                                    nowunused, nunused,
                                    (xlrec.flags & XLHP_REPAIR_FRAGMENTATION) != 0,
                                    (xlrec.flags & XLHP_ON_TOAST_RELATION) != 0);
```

2

Баг 1. На что повлиял патч?

Xid64 - патч

pruneheap.c

```
1592  heap_page_prune_execute(Buffer buffer, bool lp_truncate_only,
1593                          OffsetNumber *redirected, int nredirected,
1594                          OffsetNumber *nowdead, int ndead,
1595                          OffsetNumber *nowunused, int nunused,
1596                          bool repairFragmentation,
1597                          bool is_toast)
1598  {
```

pruneheap.c

src > backend > access > heap > C pruneheap.c > heap_page_prune_execute(Buffer, bool, OffsetNur

```
1597                          bool is_toast)
1739      else
1741          /*
1745      if (repairFragmentation)
1746          PageRepairFragmentation(page, is_toast);
1747
1748      /*
1749       * Now that the page has been modified, assert that redirec
1750       * still point to valid targets.
1751       */
1752      page_verify_redirects(page);
1753  }       You, 3 days ago • psql: Make test robust against locale
1754  }
```

PG18dev

pruneheap.c 5

src > backend > access > heap > C pruneheap.c > heap_page_prune_execute(Buffer, bool, OffsetNu

```
1550  /*
1559   */
1560  void
1561  heap_page_prune_execute(Buffer buffer, bool lp_truncate_only,
1562                          OffsetNumber *redirected, int nredirected,
1563                          OffsetNumber *nowdead, int ndead,
1564                          OffsetNumber *nowunused, int nunused)
1565  {
```

pruneheap.c 5

src > backend > access > heap > C pruneheap.c > ...

```
1564                          OffsetNumber *nowunused, int nunused)
1706      else
1707      {
1708          /*
1709           * Finally, repair any fragmentation, and update the page's
1710           * about whether it has free pointers.
1711           */
1712          PageRepairFragmentation(page);
1713
1714          /*
1715           * Now that the page has been modified, assert that redirec
1716           * still point to valid targets.
```

Баг 2.

git checkout JKL
./bld.sh

Запускаем тест.
./tst1.sh

```
2025-04-05 19:26:02.102 +08 [297359] FATAL:  inconsistent page found, rel 1663/16384/2611,
forknum 0, blkno 0
2025-04-05 19:26:02.102 +08 [297359] CONTEXT:  WAL redo at 0/7D0E3758 for Heap2/
PRUNE_VACUUM_SCAN: snapshotConflictHorizon: 6905, isCatalogRel: F, nplans: 1, nredirected: 0,
ndead: 0, nunused: 0, plans: [{ xmax: 0, infomask: 2816, infomask2: 4, ntuples: 33, offsets:
[1, 2, 3, 4, 5, 6, 66, 67, 69, 70, 71, 72, 75, 76, 77, 78, 104, 105, 106, 107, 108, 109, 194,
195, 196, 197, 198, 199, 200, 202, 203, 260, 261] }], dead: [204, 205, 211, 212, 240, 241];
blkref #0: rel 1663/16384/2611 blk 0 FPW
```

```
evoro:lin-ev:~/tantor/dv/pg18xid64PS(ps/pg18xid64)$ git log
commit f83d709760d84253b539ee454ce9ce1ceb9ad9ac
Author: Heikki Linnakangas <heikki.linnakangas@iki.fi>
Date:   Mon Mar 25 14:59:58 2024 +0200

    Merge prune, freeze and vacuum WAL record formats
```

Баг 2.
Адаптируем логи

git checkout PQR
./bld.sh

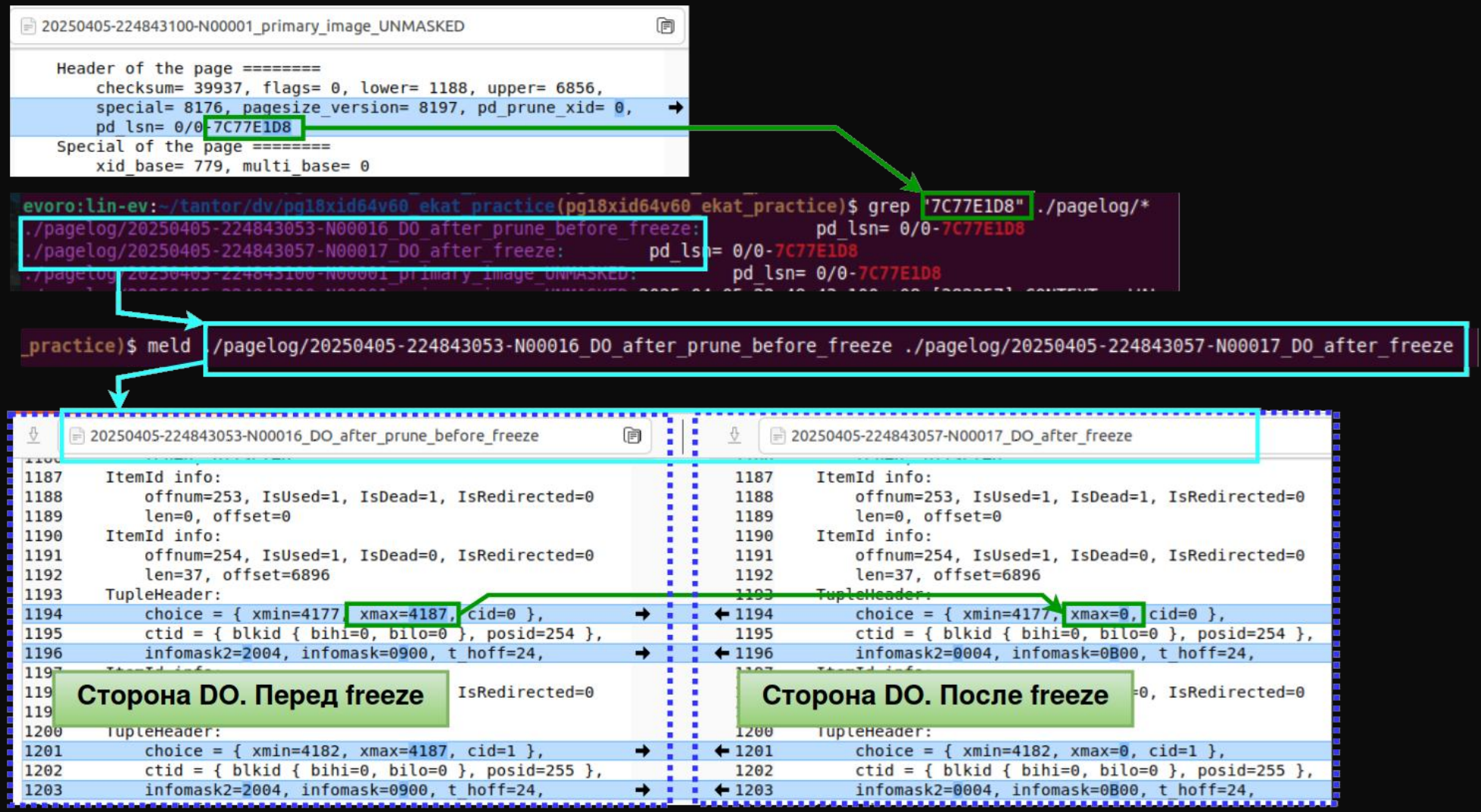
Запускаем тест.
./tst1.sh

Извлекаем лог страницы из логов standby-сервера.
./pagelog_standb.sh

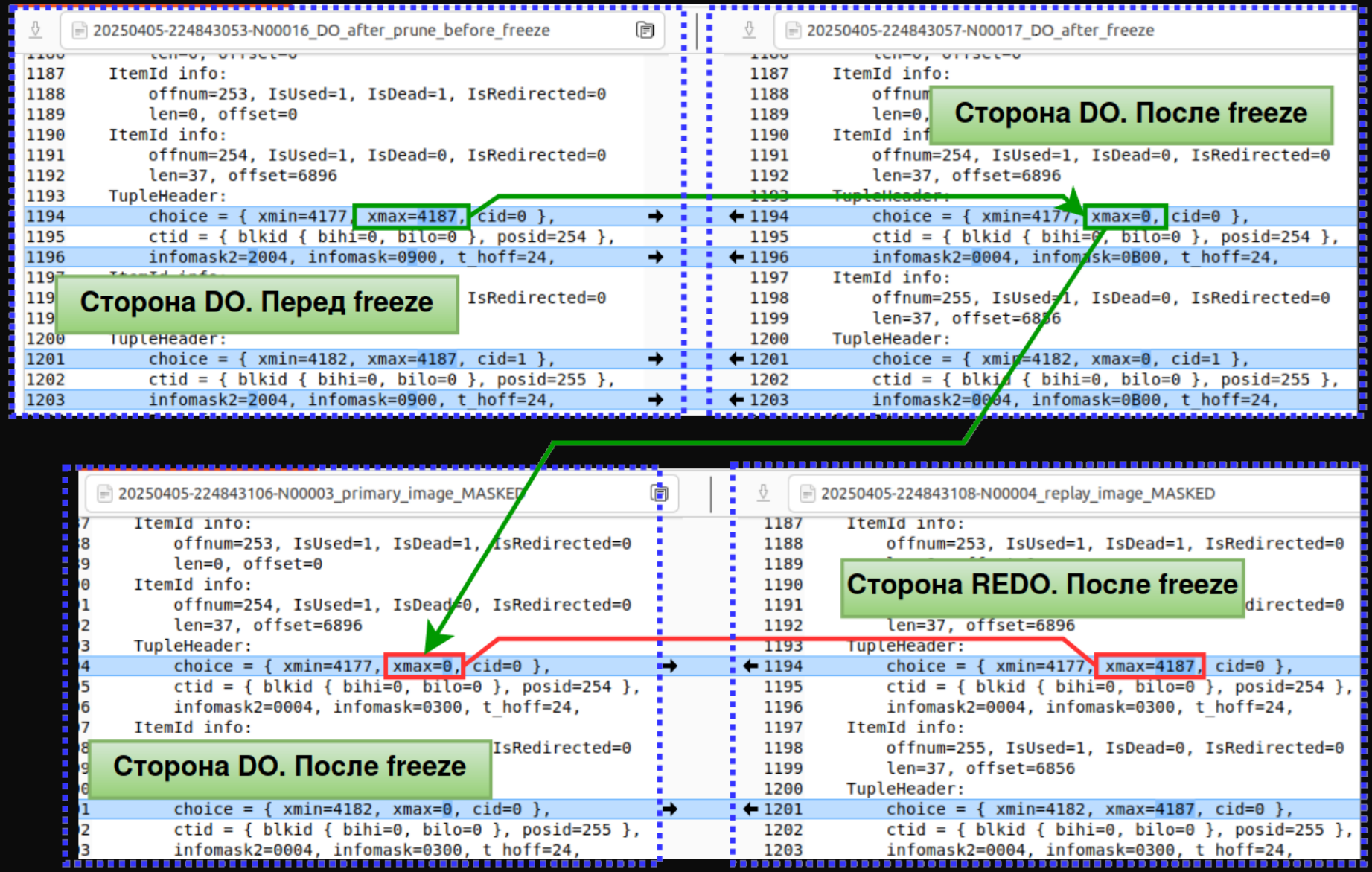
```
2025-04-05 19:26:02.102 +08 [297359] FATAL:  inconsistent page found, rel 1663/16384/2611,
forknum 0, blkno 0
2025-04-05 19:26:02.102 +08 [297359] CONTEXT:  WAL redo at 0/7D0E3758 for Heap2/
PRUNE_VACUUM_SCAN: snapshotConflictHorizon: 6905, isCatalogRel: F, nplans: 1, nredirected: 0,
ndead: 6, nunused: 0, plans: [{ xmax: 0, infomask: 2816, infomask2: 4, ntuples: 33, offsets:
[1, 2, 3, 4, 5, 6, 66, 67, 69, 70, 71, 72, 75, 76, 77, 78, 104, 105, 106, 107, 108, 109, 194,
195, 196, 197, 198, 199, 200, 202, 203, 260, 261] }], dead: [204, 205, 211, 212, 240, 241];
blkref #0: rel 1663/16384/2611 blk 0 FPW
```

```
evoro:lin-ev:~/tantor/dv/pg18xid64v60_ekat_practice(pg18xid64v60_ek
diff --git a/src/include/access/htup_details.h b/src/include/access
index 4f0e1408316..eac73462cb4 100644
--- a/src/include/access/htup_details.h
+++ b/src/include/access/htup_details.h
@@ -1384,7 +1384,7 @@ static inline void baloo_log_page_X(RelFileLo
 {
 #define BALOO_X_SPC     1663U
 #define BALOO_X_DB      16384U
-#define BALOO_X_REL 1259U
+#define BALOO_X_REL 2611U

        if (locator.spcOid == BALOO_X_SPC
                        && locator.dbOid == BALOO_X_DB
```

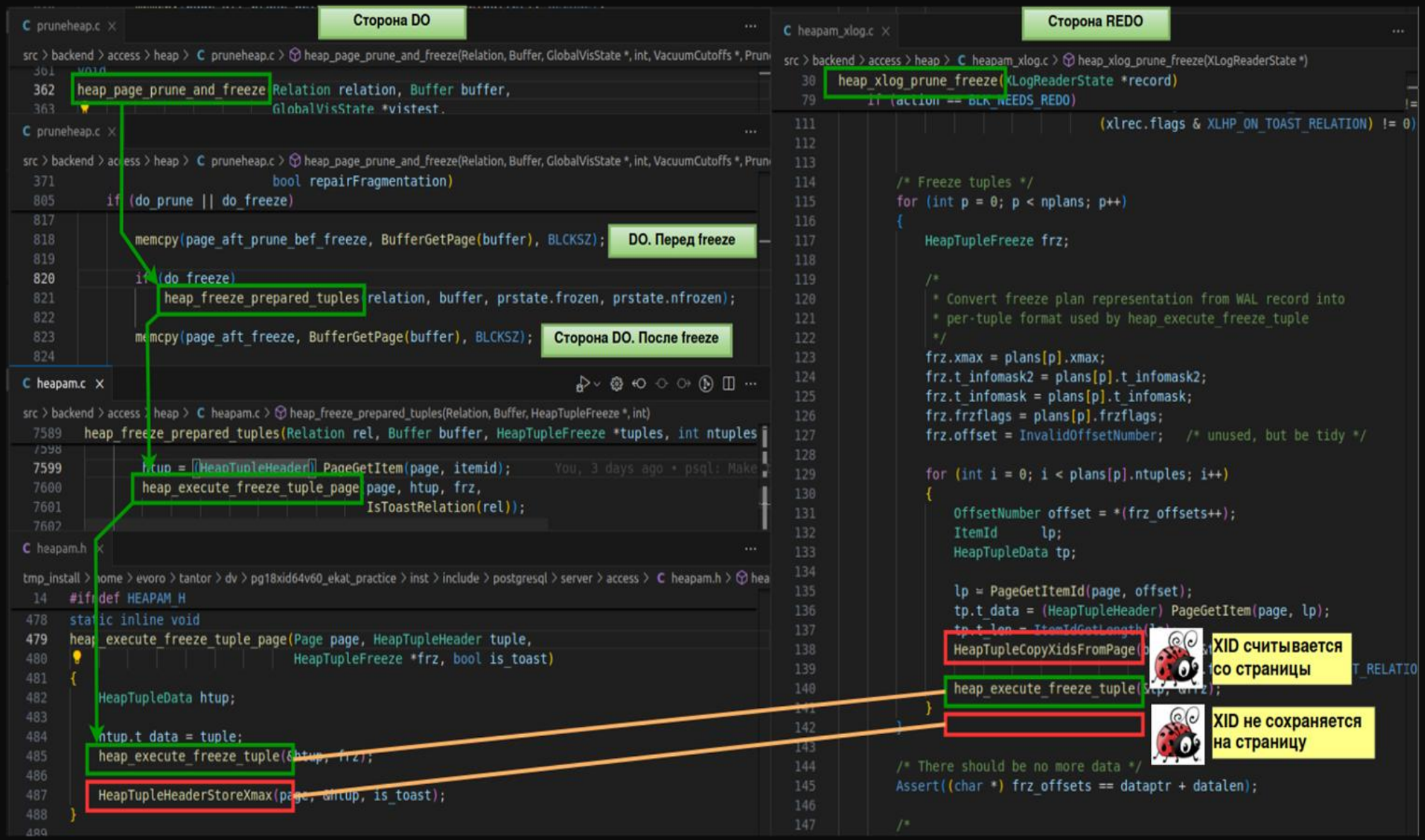
Баг 2. Поиск образа страницы.



Баг 2.
Анализ.


20250405-224843053-N00016_DO_after_prune_before_freeze
20250405-224843057-N00017_DO_after_freeze
Сторона DO. Перед freeze
Сторона DO. После freeze
20250405-224843106-N00003_primary_image_MASKED
20250405-224843108-N00004_replay_image_MASKED
Сторона DO. После freeze
Сторона REDO. После freeze

Баг 2. Анализ.


Сторона DO
Сторона REDO
DO. Перед freeze
Сторона DO. После freeze
XID считывается со страницы
XID не сохраняется на страницу

Баг 2. Ремонт.
“Staying alive!”

git checkout VWX
./bld.sh

Запускаем тест.
./tst1.sh

```
# +++ tap check in src/test/recovery +++
t/027_stream_regress.pl .. ok
All tests successful.
Files=1, Tests=9, 215 wallclock secs ( 0.02 usr  0.00 sys +
Result: PASS
make: Leaving directory '/home/evoro/tantor/dv/pg18xid64v60
++ set +x
================================================================
BALOO: FINISHED TESTING. WITH WAL consistency checking.
================================================================
```

Партнёры и соучастники

(в доработке патча xid64 для PG17, PG18dev)

Евгений Воропаев (Тантор)
Адаптация кода. Доработки.
Ремонт основной массы багов. Доклад.

Иван Кушнаренко (Тантор)
Fixed the "stale xid64" problem at the heapam_tuple_satisfies_snapshot function.

Эдуард Степанов (Тантор)
Fixed the error that occurred during parsing an XLog record in the DecodeUpdate function when the XLOG_HEAPINITPAGE flag is set.

Сергей Соловьёв (Тантор)
Participation in fixing the bug of the XMAX_COMMITTED hint bit.

Олег Гуров (Тантор)
Руководство, техническое наставничество, волшебный пендель.

Андрей Бородин (Yandex)
Техническое наставничество.

tantor

PG BootCamp Russia 2025 Ekaterinburg
PGBootCamp.ru

Спасибо за внимание!

www.tantorlabs.ru