Журналирование
Журнал предзаписи

Авторы: Егор Рогов, Павел Лузанов, Илья Баштанов

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Журнал упреждающей записи (WAL)

Логическое и физическое устройство журнала

Процесс упреждающей записи и восстановление

Журнал

Основная задача

возможность восстановления согласованности данных после сбоя

Механизм

при изменении данных действие также записывается в журнал

журнальная запись попадает на диск раньше измененных данных

восстановление после сбоя — повторное выполнение потерянных операций с помощью журнальных записей

Основная причина существования журнала — необходимость восстановления согласованности данных в случае сбоя, при котором теряется содержимое оперативной памяти, в частности, буферный кеш.

Журнал обеспечивает выполнение свойства долговечности (буква «D» из набора свойств транзакций ACID).

Одновременно с изменением данных в странице буферного кеша в журнале создается запись, содержащая информацию, достаточную для повторения этой операции. Журнальная запись в обязательном порядке попадает на диск (или другое энергонезависимое устройство) до того, как туда попадет измененная страница — отсюда и название: «журнал предзаписи», «write-ahead log».

В случае сбоя можно прочитать журнал и при необходимости повторить те операции, которые уже были выполнены, но результат которых не успел попасть на диск.

https://postgrespro.ru/docs/postgresql/13/wal-intro

Что защищено журналом

Изменение любых страниц в буферном кеше

- в том числе страницы таблиц и индексов
- кроме нежурналируемых и временных таблиц

Фиксация и отмена транзакций — буферы CLOG

Файловые операции

- создание и удаление файлов
- создание и удаление каталогов

Журналировать нужно все операции, при выполнении которых возможна ситуация, что при сбое изменения (или часть изменений) не дойдут до диска.

В частности, в журнал записываются следующие действия:

- изменение страниц в буферном кеше (как правило, это страницы таблиц и индексов) — так как измененная страница попадает на диск не сразу;

- фиксация и отмена транзакций — точно так же, изменение статуса происходит в буфере CLOG и попадает на диск не сразу;

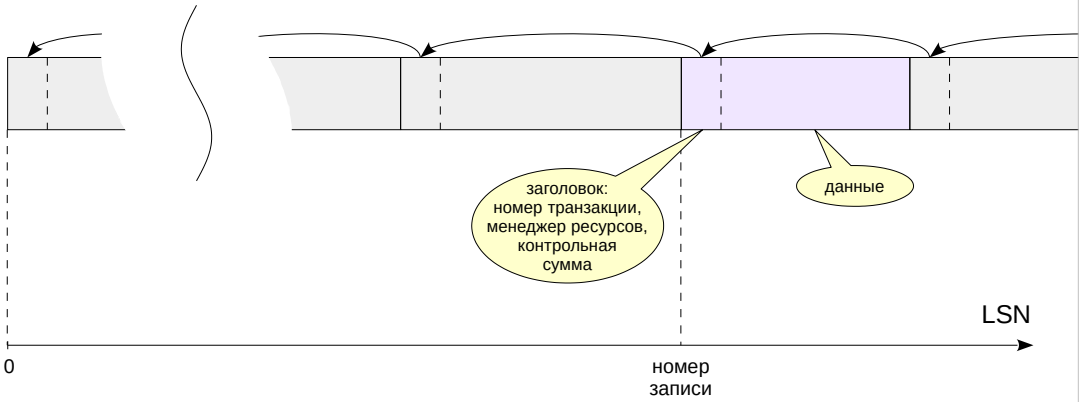
- файловые операции (создание и удаление файлов и каталогов, например, создание файлов при создании таблицы) — так как эти операции должны происходить синхронно с изменением данных.

При этом в журнал не записываются:

- операции с нежурналируемыми таблицами — их название говорит само за себя;

- операции с временными таблицами — они существуют не дольше, чем создавший их сеанс, и поэтому не нуждаются в восстановлении.

Логическое устройство



последовательность записей
номер записи — 64-битный LSN (log sequence number)
специальный тип pg_lsn

5

Логически журнал можно представить себе как последовательность записей различной длины. Каждая запись содержит данные о некоторой операции, предваренные заголовком. В заголовке, в числе прочего, указаны:

- номер транзакции, к которой относится запись;

- менеджер ресурсов — компонент системы, ответственный за данную запись;

- контрольная сумма (CRC).

Сами данные могут иметь разный смысл. Менеджер ресурсов «понимает», как интерпретировать данные в своей записи. Есть отдельные менеджеры для таблиц, для каждого типа индекса, для статуса транзакций и т. п. Например, данные могут представлять собой некоторый фрагмент страницы, который надо записать поверх ее содержимого с определенным смещением.

Для того чтобы сослаться на определенную запись, используется тип данных pg_lsn (LSN = log sequence number) — 64-битное число, представляющее собой байтовое смещение до записи относительно начала журнала.

https://postgrespro.ru/docs/postgresql/13/datatype-pg-lsn

Логическое устройство журнала

Список менеджеров ресурсов можно получить утилитой pg_waldump:

```
student$ /usr/lib/postgresql/13/bin/pg_waldump -r list
```

```
XLOG
Transaction
Storage
CLOG
Database
Tablespace
MultiXact
RelMap
Standby
Heap2
Heap
Btree
Hash
Gin
Gist
Sequence
SPGist
BRIN
CommitTs
ReplicationOrigin
Generic
LogicalMessage
```

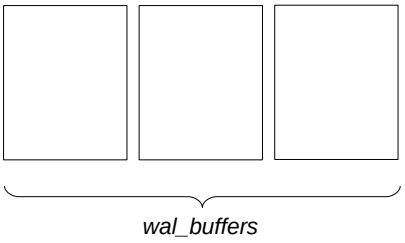
LSN выводится как два 32-битных числа в шестнадцатеричной системе через косую черту.

Текущая позиция в журнале:

```
=> SELECT pg_current_wal_insert_lsn();
```

```
 pg_current_wal_insert_lsn
---------------------------
 0/BC0E9D0
(1 row)
```

Физическое устройство



В памяти

кольцевой буферный кеш

wal_buffers = –1

1/32 *shared_buffers*

На диске

файлы (сегменты) по 16 МБ

На диске журнал хранится в виде файлов (сегментов) в каталоге PGDATA/pg_wal. Каждый файл по умолчанию занимает 16 МБ. Размер сегмента может быть задан при инициализации кластера.

Журнальные записи попадают в текущий файл; когда он заполняется — начинает использоваться следующий.

В оперативной памяти для журнала выделены специальные буферы. Размер кеша задается параметром *wal_buffers* (значение по умолчанию подразумевает автоматическую настройку: выделяется 1/32 часть буферного кеша).

Журнальный кеш устроен наподобие буферного кеша, но работает преимущественно в режиме кольцевого буфера: записи добавляются в «голову» буфера, а записываются на диск с «хвоста».

Физическое устройство журнала

Все журнальные файлы (сегменты) находятся в каталоге /var/lib/postgresql/13/main/pg_wal/, а начиная с PostgreSQL 10 их также показывает специальная функция:

```
=> SELECT * FROM pg_ls_waldir() LIMIT 10;
```

```
           name           |   size   |      modification
--------------------------+----------+------------------------
 000000010000000000000000E | 16777216 | 2024-01-16 10:52:51+03
 000000010000000000000000F | 16777216 | 2024-01-16 10:52:52+03
 000000010000000000000000C | 16777216 | 2024-01-16 10:52:52+03
 000000010000000000000000B | 16777216 | 2024-01-16 10:53:15+03
 000000010000000000000000D | 16777216 | 2024-01-16 10:52:51+03
(5 rows)
```

Имена файлов составлены из трех чисел. Первое — номер линии времени (используется при восстановлении из архива), а два следующих — старшие разряды LSN.

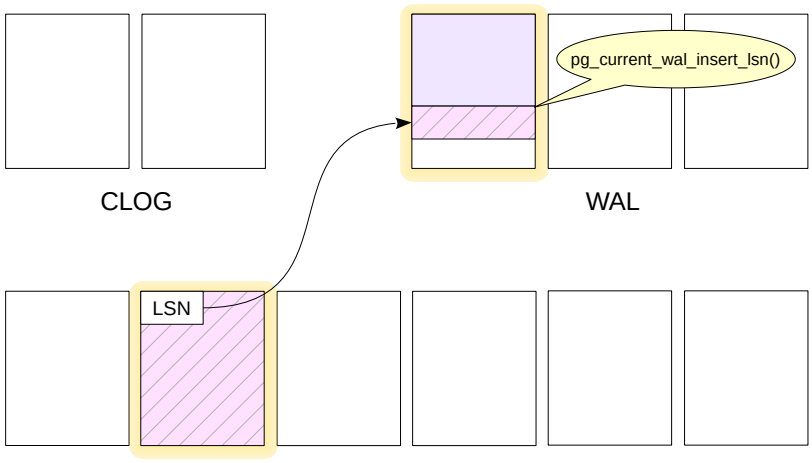
Размер файлов можно задать при инициализации кластера, по умолчанию — 16 Мбайт.

Текущая позиция находится в этом файле:

```
=> SELECT pg_walfile_name('0/BC0E9D0');
```

```
     pg_walfile_name
--------------------------
 000000010000000000000000B
(1 row)
```

Упреждающая запись



Проиллюстрируем сказанное выше про упреждающую запись.
На слайде показаны три важные области общей памяти экземпляра:

- буферный кеш (размером *shared_buffers*),

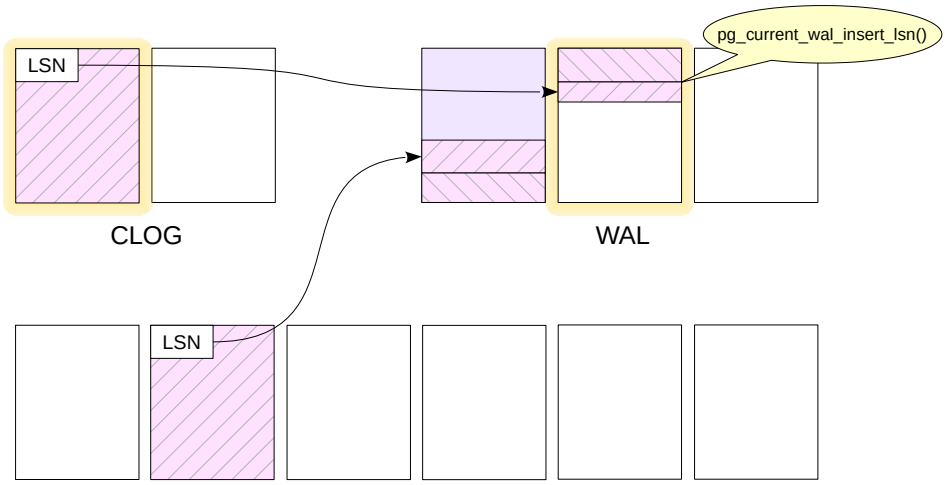
- только что рассмотренный журнальный кеш WAL (размером *wal_buffers*),

- кеш состояния транзакций, называемый также CLOG (размером 128 страниц).

При изменении страницы данных в буферном кеше формируется журнальная запись. Она помещается в страницу журнала, а ссылка на запись (если быть точным, ее LSN + длина, то есть LSN следующей записи) записывается в специальное поле LSN в заголовке страницы данных.

Позицию для записи можно узнать с помощью функции pg_current_wal_insert_lsn().

Упреждающая запись

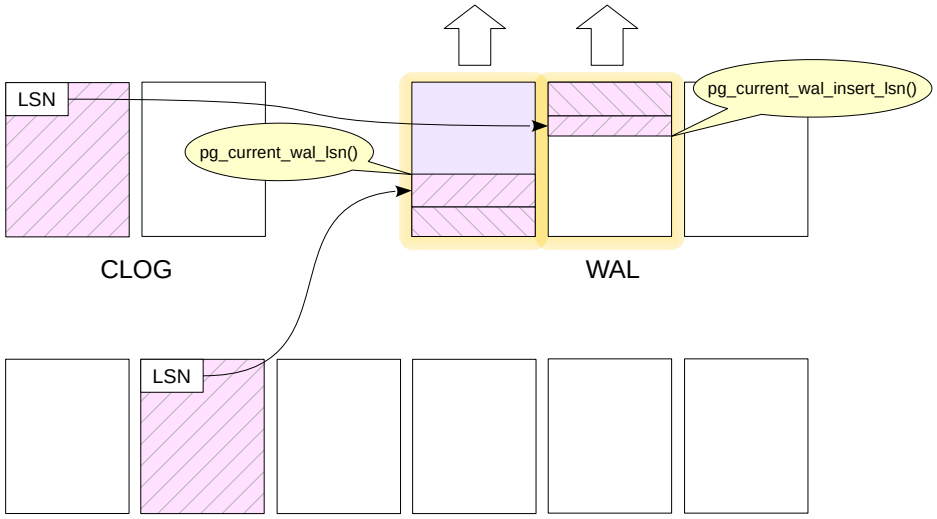


Допустим, далее происходит фиксация транзакции. Для этого формируется журнальная запись, меняется бит состояния на странице CLOG и ссылка на эту запись проставляется в поле LSN измененной страницы.

При вставке указатель pg_current_wal_insert_lsn сдвигается вперед.

Заметим, что между записями, относящимися к одной транзакции, могут попасть записи других транзакций, относящихся к любой БД. Журнал — общий для всего кластера.

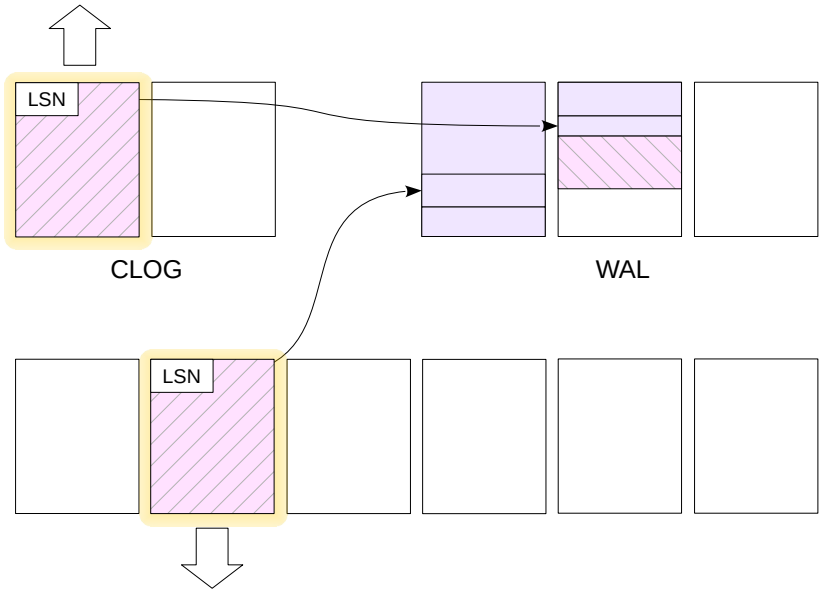
Упреждающая запись



Далее в какой-то момент (в какой именно — будет рассмотрено в теме «Настройка журнала») журнальные записи, которые еще не попали на диск, должны на него попасть.

Функция pg_current_wal_lsn() показывает последнюю запись, уже дошедшую до диска.

Упреждающая запись



Только после того, как на диск попали журнальные записи, могут быть записаны и сами измененные страницы. Порядок контролируется с учетом LSN последнего изменения страницы и текущего состояния pg_current_wal_lsn. При этом работа продолжается, в журнал будут попадать новые и новые записи. Главное, чтобы запись с LSN последнего изменения страницы была на диске.

Если окажется, что страница данных должна быть записана (например, она вытесняется из буферного кеша), а журнальная запись еще не попала на диск, журнальные буферы сбрасываются принудительно.

Упреждающая запись

Создадим небольшую таблицу:

```
=> CREATE DATABASE wal_log;
```

```
CREATE DATABASE
```

```
=> \c wal_log
```

```
You are now connected to database "wal_log" as user "student".
```

```
=> CREATE TABLE t(id integer);
```

```
CREATE TABLE
```

```
=> INSERT INTO t VALUES (1);
```

```
INSERT 0 1
```

Мы будем заглядывать в заголовок табличной страницы. Для этого понадобится расширение:

```
=> CREATE EXTENSION pageinspect;
```

```
CREATE EXTENSION
```

Начнем транзакцию.

```
=> BEGIN;
```

```
BEGIN
```

Текущая позиция и текущий сегмент журнала:

```
=> SELECT pg_current_wal_insert_lsn(), pg_walfile_name(pg_current_wal_insert_lsn());
```

```
 pg_current_wal_insert_lsn |     pg_walfile_name
---------------------------+--------------------------
 0/BD2B8E8                 | 000000010000000000000000B
(1 row)
```

Изменим строку в таблице:

```
=> UPDATE t SET id = id + 1;
```

```
UPDATE 1
```

Позиция в журнале изменилась:

```
=> SELECT pg_current_wal_insert_lsn();
```

```
 pg_current_wal_insert_lsn
---------------------------
 0/BD2B930
(1 row)
```

Этот же номер LSN (или меньший, если в журнал попали дополнительные записи) мы найдем и в заголовке измененной страницы:

```
=> SELECT lsn FROM page_header(get_raw_page('t',0));
```

```
    lsn
-----------
 0/BD2B930
(1 row)
```

Завершим транзакцию.

```
=> COMMIT;
```

```
COMMIT
```

Позиция в журнале снова изменилась:

```
=> SELECT pg_current_wal_insert_lsn();
```

```
 pg_current_wal_insert_lsn
---------------------------
 0/BD2B958
(1 row)
```

Размер журнальных записей (в байтах), соответствующих нашей транзакции, можно узнать вычитанием одной позиции из другой:

```
=> SELECT '0/BD2B958'::pg_lsn - '0/BD2B8E8'::pg_lsn;
```

```
 ?column?
----------
      112
(1 row)
```

Безусловно, в журнал попадает информация обо всех действиях во всем кластере, но в данном случае мы рассчитываем на то, что в системе ничего не происходит.

Теперь воспользуемся утилитой pg_waldump, чтобы посмотреть содержимое журнала.

Утилита может работать и с диапазоном LSN (как в этом примере), и выбрать записи для указанной транзакции. Запускать ее следует от имени пользователя ОС postgres, так как ей требуется доступ к журнальным файлам на диске.

```
postgres$ /usr/lib/postgresql/13/bin/pg_waldump -p /var/lib/postgresql/13/main/pg_wal -s 0/BD2B8E8 -e 0/BD2B958 000000010000000000000000B
```

```
rmgr: Heap        len (rec/tot):     69/    69, tx:        650, lsn: 0/0BD2B8E8, prev 0/0BD2B8A0, desc: HOT_UPDATE off 1 xmax 650 flags 0x41 ; new off 2 xmax 0, blkref #0: rel 1663/166
rmgr: Transaction len (rec/tot):     34/    34, tx:        650, lsn: 0/0BD2B930, prev 0/0BD2B8E8, desc: COMMIT 2024-01-16 10:53:19.613367 MSK
```

Мы видим заголовки журнальных записей:

- операция HOT_UPDATE, относящаяся к странице, которую мы смотрели (rel+blk),
- операция COMMIT с указанием времени.

Восстановление

Алгоритм (упрощенный)

при старте сервера после сбоя
(состояние кластера в `pg_control` отличается от «shut down»):

1. для каждой журнальной записи:
   1.1. определить страницу, к которой относится эта запись
   1.2. применить запись, если ее LSN больше, чем LSN страницы
2. перезаписать нежурналируемые таблицы init-файлами

Если в работе сервера произошел сбой, то при последующем запуске процесс startup (запускаемый postmaster-ом в самом начале работы) обнаружит это, посмотрев в файл pg_control и увидев статус, отличный от «shut down». Тогда автоматически будет выполнено восстановление.

Процесс startup будет последовательно читать журнал и применять записи к страницам, если в этом есть необходимость (что можно проверить, сравнив LSN страницы на диске с LSN журнальной записи). Изменение страниц происходит в буферном кеше, как при обычной работе — для этого postmaster запускает необходимые фоновые процессы.

Аналогично записи применяются и к файлам: например, если запись говорит о том, что файл должен существовать, а его нет — файл создается.

В конце процесса все нежурналируемые таблицы перезаписываются с помощью образов в init-файлах.

Приведенный алгоритм является упрощенным. В частности, ничего не говорится о том, с какого места надо начинать чтение журнальных записей (это будет рассмотрено в теме «Контрольная точка»).

Итоги

Использование буферов в оперативной памяти
приводит к необходимости журналирования

Журнал содержит информацию,
позволяющую повторно выполнить операции после сбоя
и восстановить согласованность

Журнал всегда записывается на диск до того,
как записываются измененные страницы данных

Практика

1. Создайте таблицу с первичным ключом и добавьте в нее несколько строк. Сколько байт занимают сгенерированные журнальные записи?
2. Чем можно объяснить довольно большое число? Посмотрите заголовки этих журнальных записей утилитой pg_waldump и проверьте свои предположения.
3. Измените добавленные в таблицу строки. Снова измените строки, но не фиксируйте транзакцию. Сымитируйте сбой, прервав процесс postmaster.

   Запустите сервер и убедитесь, что зафиксированные изменения не пропали, а незафиксированная транзакция оборвана. Найдите информацию о восстановлении после сбоя в журнале сообщений сервера.

3. Воспользуйтесь командой

$ sudo kill -9 *номер-процесса*

Номер процесса находится в файле postmaster.pid в каталоге PGDATA сервера.

1. Размер журнальных записей

```
=> CREATE DATABASE wal_log;
```

```
CREATE DATABASE
```

```
=> \c wal_log
```

```
You are now connected to database "wal_log" as user "student".
```

Запомним начальную позицию в журнале:

```
=> SELECT pg_current_wal_insert_lsn();
```

```
 pg_current_wal_insert_lsn
---------------------------
 0/3A4D36B8
(1 row)
```

Создадим таблицу и добавим строки:

```
=> CREATE TABLE t(
  id integer PRIMARY KEY,
  s text
);
```

```
CREATE TABLE
```

```
=> INSERT INTO t VALUES (1, 'A'), (2, 'B'), (3, 'C');
```

```
INSERT 0 3
```

Запомним конечную позицию:

```
=> SELECT pg_current_wal_insert_lsn();
```

```
 pg_current_wal_insert_lsn
---------------------------
 0/3A4F2340
(1 row)
```

Размер журнальных записей:

```
=> SELECT '0/3A4F2340'::pg_lsn - '0/3A4D36B8'::pg_lsn;
```

```
 ?column?
----------
   126088
(1 row)
```

2. Состав журнальных записей

Журнальный файл:

```
=> SELECT pg_walfile_name('0/3A4D36B8');
```

```
     pg_walfile_name
--------------------------
 00000001000000000000003A
(1 row)
```

Смотрим записи:

```
postgres$ /usr/lib/postgresql/13/bin/pg_waldump -p /var/lib/postgresql/13/main/pg_wal -s 0/3A4D36B8 -e 0/3A4F2340 00000001000000000000003A
```

```
rmgr: Storage     len (rec/tot):     42/    42, tx:          0, lsn: 0/3A4D36B8, prev 0/3A4D2EA0, desc: CREATE base/25322/25323
rmgr: Heap        len (rec/tot):     54/  1518, tx:     128892, lsn: 0/3A4D36E8, prev 0/3A4D36B8, desc: INSERT off 8 flags 0x01, blkref #0: rel 1663/25322/1247 blk 9 FPW
rmgr: Btree       len (rec/tot):     53/  1013, tx:     128892, lsn: 0/3A4D3CD8, prev 0/3A4D36E8, desc: INSERT_LEAF off 46, blkref #0: rel 1663/25322/2703 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  2021, tx:     128892, lsn: 0/3A4D40E8, prev 0/3A4D3CD8, desc: INSERT_LEAF off 23, blkref #0: rel 1663/25322/2704 blk 2 FPW
rmgr: Heap        len (rec/tot):     54/  6798, tx:     128892, lsn: 0/3A4D48D0, prev 0/3A4D40E8, desc: INSERT off 112 flags 0x01, blkref #0: rel 1663/25322/2608 blk 56 FPW
rmgr: Btree       len (rec/tot):     53/  4621, tx:     128892, lsn: 0/3A4D6378, prev 0/3A4D48D0, desc: INSERT_LEAF off 162, blkref #0: rel 1663/25322/2673 blk 24 FPW
rmgr: Btree       len (rec/tot):     53/  8009, tx:     128892, lsn: 0/3A4D7588, prev 0/3A4D6378, desc: INSERT_LEAF off 283, blkref #0: rel 1663/25322/2674 blk 39 FPW
rmgr: Heap        len (rec/tot):    207/   207, tx:     128892, lsn: 0/3A4D94F0, prev 0/3A4D7588, desc: INSERT off 9 flags 0x00, blkref #0: rel 1663/25322/1247 blk 9
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4D95C0, prev 0/3A4D94F0, desc: INSERT_LEAF off 46, blkref #0: rel 1663/25322/2703 blk 2
rmgr: Btree       len (rec/tot):     53/  5873, tx:     128892, lsn: 0/3A4D9600, prev 0/3A4D95C0, desc: INSERT_LEAF off 65, blkref #0: rel 1663/25322/2704 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4DAD10, prev 0/3A4D9600, desc: INSERT off 113 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4DAD60, prev 0/3A4DAD10, desc: INSERT_LEAF off 162, blkref #0: rel 1663/25322/2673 blk 24
rmgr: Btree       len (rec/tot):     53/  2185, tx:     128892, lsn: 0/3A4DADA8, prev 0/3A4DAD60, desc: INSERT_LEAF off 75, blkref #0: rel 1663/25322/2674 blk 41 FPW
rmgr: Heap        len (rec/tot):     54/   874, tx:     128892, lsn: 0/3A4DB638, prev 0/3A4DADA8, desc: INSERT off 2 flags 0x01, blkref #0: rel 1663/25322/1259 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/  2213, tx:     128892, lsn: 0/3A4DB9A8, prev 0/3A4DB638, desc: INSERT_LEAF off 106, blkref #0: rel 1663/25322/2662 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  3845, tx:     128892, lsn: 0/3A4DC268, prev 0/3A4DB9A8, desc: INSERT_LEAF off 87, blkref #0: rel 1663/25322/2663 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  1013, tx:     128892, lsn: 0/3A4DD170, prev 0/3A4DC268, desc: INSERT_LEAF off 46, blkref #0: rel 1663/25322/3455 blk 4 FPW
rmgr: Heap        len (rec/tot):     54/  7498, tx:     128892, lsn: 0/3A4DD568, prev 0/3A4DD170, desc: INSERT off 31 flags 0x01, blkref #0: rel 1663/25322/1249 blk 16 FPW
rmgr: Btree       len (rec/tot):     53/  6665, tx:     128892, lsn: 0/3A4DF2D0, prev 0/3A4DD568, desc: INSERT_LEAF off 179, blkref #0: rel 1663/25322/2658 blk 14 FPW
rmgr: Btree       len (rec/tot):     53/  5973, tx:     128892, lsn: 0/3A4E0CF8, prev 0/3A4DF2D0, desc: INSERT_LEAF off 294, blkref #0: rel 1663/25322/2659 blk 9 FPW
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4E2468, prev 0/3A4E0CF8, desc: INSERT off 32 flags 0x00, blkref #0: rel 1663/25322/1249 blk 16
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E2518, prev 0/3A4E2468, desc: INSERT_LEAF off 180, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E2558, prev 0/3A4E2518, desc: INSERT_LEAF off 295, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4E2598, prev 0/3A4E2558, desc: INSERT off 33 flags 0x00, blkref #0: rel 1663/25322/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4E2648, prev 0/3A4E2598, desc: INSERT_LEAF off 179, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E2690, prev 0/3A4E2648, desc: INSERT_LEAF off 294, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4E26D0, prev 0/3A4E2690, desc: INSERT off 34 flags 0x00, blkref #0: rel 1663/25322/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4E2780, prev 0/3A4E26D0, desc: INSERT_LEAF off 182, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E27C8, prev 0/3A4E2780, desc: INSERT_LEAF off 294, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4E2808, prev 0/3A4E27C8, desc: INSERT off 36 flags 0x00, blkref #0: rel 1663/25322/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4E28B8, prev 0/3A4E2808, desc: INSERT_LEAF off 179, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E2900, prev 0/3A4E28B8, desc: INSERT_LEAF off 294, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4E2940, prev 0/3A4E2900, desc: INSERT off 38 flags 0x00, blkref #0: rel 1663/25322/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4E29F0, prev 0/3A4E2940, desc: INSERT_LEAF off 183, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E2A38, prev 0/3A4E29F0, desc: INSERT_LEAF off 294, blkref #0: rel 1663/25322/2659 blk 9
rmgr: XLOG        len (rec/tot):     49/  8241, tx:     128892, lsn: 0/3A4E2A78, prev 0/3A4E2A38, desc: FPI_FOR_HINT , blkref #0: rel 1663/25322/1249 fork fsm blk 2 FPW
rmgr: Heap        len (rec/tot):     54/  1558, tx:     128892, lsn: 0/3A4E4AC8, prev 0/3A4E2A78, desc: INSERT off 10 flags 0x01, blkref #0: rel 1663/25322/1249 blk 52 FPW
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4E50E0, prev 0/3A4E4AC8, desc: INSERT_LEAF off 179, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E5128, prev 0/3A4E50E0, desc: INSERT_LEAF off 294, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4E5168, prev 0/3A4E5128, desc: INSERT off 11 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4E5218, prev 0/3A4E5168, desc: INSERT_LEAF off 184, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E5260, prev 0/3A4E5218, desc: INSERT_LEAF off 294, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4E52A0, prev 0/3A4E5260, desc: INSERT off 114 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     53/  1849, tx:     128892, lsn: 0/3A4E52F0, prev 0/3A4E52A0, desc: INSERT_LEAF off 57, blkref #0: rel 1663/25322/2673 blk 33 FPW
rmgr: Btree       len (rec/tot):     53/  6105, tx:     128892, lsn: 0/3A4E5A30, prev 0/3A4E52F0, desc: INSERT_LEAF off 4, blkref #0: rel 1663/25322/2674 blk 37 FPW
rmgr: Heap        len (rec/tot):     54/  3730, tx:     128892, lsn: 0/3A4E7228, prev 0/3A4E5A30, desc: INSERT off 56 flags 0x00, blkref #0: rel 1664/0/1214 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/  1661, tx:     128892, lsn: 0/3A4E80D8, prev 0/3A4E7228, desc: INSERT_LEAF off 56, blkref #0: rel 1664/0/1232 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/  1213, tx:     128892, lsn: 0/3A4E8758, prev 0/3A4E80D8, desc: INSERT_LEAF off 56, blkref #0: rel 1664/0/1233 blk 1 FPW
rmgr: Standby     len (rec/tot):     42/    42, tx:     128892, lsn: 0/3A4E8C18, prev 0/3A4E8758, desc: LOCK xid 128892 db 25322 rel 25323
rmgr: Storage     len (rec/tot):     42/    42, tx:     128892, lsn: 0/3A4E8C48, prev 0/3A4E8C18, desc: CREATE base/25322/25326
rmgr: Heap        len (rec/tot):    207/   207, tx:     128892, lsn: 0/3A4E8C78, prev 0/3A4E8C48, desc: INSERT off 10 flags 0x00, blkref #0: rel 1663/25322/1247 blk 9
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4E8D48, prev 0/3A4E8C78, desc: INSERT_LEAF off 48, blkref #0: rel 1663/25322/2703 blk 2
rmgr: Btree       len (rec/tot):     53/  6113, tx:     128892, lsn: 0/3A4E8D88, prev 0/3A4E8D48, desc: INSERT_LEAF off 124, blkref #0: rel 1663/25322/2704 blk 4 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4EA588, prev 0/3A4E8D88, desc: INSERT off 115 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EA5D8, prev 0/3A4EA588, desc: INSERT_LEAF off 164, blkref #0: rel 1663/25322/2673 blk 24
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EA620, prev 0/3A4EA5D8, desc: INSERT_LEAF off 284, blkref #0: rel 1663/25322/2674 blk 39
rmgr: Heap        len (rec/tot):    203/   203, tx:     128892, lsn: 0/3A4EA668, prev 0/3A4EA620, desc: INSERT off 3 flags 0x00, blkref #0: rel 1663/25322/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EA738, prev 0/3A4EA668, desc: INSERT_LEAF off 107, blkref #0: rel 1663/25322/2662 blk 2
rmgr: Btree       len (rec/tot):     53/  3173, tx:     128892, lsn: 0/3A4EA778, prev 0/3A4EA738, desc: INSERT_LEAF off 70, blkref #0: rel 1663/25322/2663 blk 5 FPW
```

```
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EB3E0, prev 0/3A4EA778, desc: INSERT_LEAF off 47, blkref #0: rel 1663/25322/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EB420, prev 0/3A4EB3E0, desc: INSERT off 12 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EB4D0, prev 0/3A4EB420, desc: INSERT_LEAF off 187, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EB518, prev 0/3A4EB4D0, desc: INSERT_LEAF off 302, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EB558, prev 0/3A4EB518, desc: INSERT off 13 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EB608, prev 0/3A4EB558, desc: INSERT_LEAF off 188, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EB650, prev 0/3A4EB608, desc: INSERT_LEAF off 303, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EB690, prev 0/3A4EB650, desc: INSERT off 14 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EB740, prev 0/3A4EB690, desc: INSERT_LEAF off 187, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EB788, prev 0/3A4EB740, desc: INSERT_LEAF off 304, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EB7C8, prev 0/3A4EB788, desc: INSERT off 15 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EB878, prev 0/3A4EB7C8, desc: INSERT_LEAF off 190, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EB8C0, prev 0/3A4EB878, desc: INSERT_LEAF off 302, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EB900, prev 0/3A4EB8C0, desc: INSERT off 16 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EB9B0, prev 0/3A4EB900, desc: INSERT_LEAF off 191, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EB9F8, prev 0/3A4EB9B0, desc: INSERT_LEAF off 302, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EBA38, prev 0/3A4EB9F8, desc: INSERT off 17 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EBAE8, prev 0/3A4EBA38, desc: INSERT_LEAF off 190, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EBB30, prev 0/3A4EBAE8, desc: INSERT_LEAF off 302, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EBB70, prev 0/3A4EBB30, desc: INSERT off 18 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EBC20, prev 0/3A4EBB70, desc: INSERT_LEAF off 192, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EBC68, prev 0/3A4EBC20, desc: INSERT_LEAF off 302, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EBCA8, prev 0/3A4EBC68, desc: INSERT off 19 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EBD58, prev 0/3A4EBCA8, desc: INSERT_LEAF off 190, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EBDA0, prev 0/3A4EBD58, desc: INSERT_LEAF off 302, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EBDE0, prev 0/3A4EBDA0, desc: INSERT off 20 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EBE90, prev 0/3A4EBDE0, desc: INSERT_LEAF off 193, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EBED8, prev 0/3A4EBE90, desc: INSERT_LEAF off 302, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Storage     len (rec/tot):     42/    42, tx:     128892, lsn: 0/3A4EBF18, prev 0/3A4EBED8, desc: CREATE base/25322/25328
rmgr: Standby     len (rec/tot):     42/    42, tx:     128892, lsn: 0/3A4EBF48, prev 0/3A4EBF18, desc: LOCK xid 128892 db 25322 rel 25328
rmgr: Heap        len (rec/tot):    203/   203, tx:     128892, lsn: 0/3A4EBF78, prev 0/3A4EBF48, desc: INSERT off 4 flags 0x00, blkref #0: rel 1663/25322/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EC060, prev 0/3A4EBF78, desc: INSERT_LEAF off 108, blkref #0: rel 1663/25322/2662 blk 2
rmgr: Btree       len (rec/tot):     88/    88, tx:     128892, lsn: 0/3A4EC0A0, prev 0/3A4EC060, desc: INSERT_LEAF off 71, blkref #0: rel 1663/25322/2663 blk 5
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EC0F8, prev 0/3A4EC0A0, desc: INSERT_LEAF off 48, blkref #0: rel 1663/25322/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EC138, prev 0/3A4EC0F8, desc: INSERT off 21 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EC1E8, prev 0/3A4EC138, desc: INSERT_LEAF off 196, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EC230, prev 0/3A4EC1E8, desc: INSERT_LEAF off 311, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EC270, prev 0/3A4EC230, desc: INSERT off 22 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EC320, prev 0/3A4EC270, desc: INSERT_LEAF off 197, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EC368, prev 0/3A4EC320, desc: INSERT_LEAF off 312, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):     54/  3590, tx:     128892, lsn: 0/3A4EC3A8, prev 0/3A4EC368, desc: INSERT off 20 flags 0x01, blkref #0: rel 1663/25322/2610 blk 3 FPW
rmgr: Btree       len (rec/tot):     53/  3193, tx:     128892, lsn: 0/3A4ED1B0, prev 0/3A4EC3A8, desc: INSERT_LEAF off 155, blkref #0: rel 1663/25322/2678 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/  3193, tx:     128892, lsn: 0/3A4EDE30, prev 0/3A4ED1B0, desc: INSERT_LEAF off 155, blkref #0: rel 1663/25322/2679 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4EEAC8, prev 0/3A4EDE30, desc: INSERT off 116 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EEB18, prev 0/3A4EEAC8, desc: INSERT_LEAF off 58, blkref #0: rel 1663/25322/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EEB60, prev 0/3A4EEB18, desc: INSERT_LEAF off 285, blkref #0: rel 1663/25322/2674 blk 39
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4EEBA8, prev 0/3A4EEB60, desc: INSERT off 117 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EEBF8, prev 0/3A4EEBA8, desc: INSERT_LEAF off 59, blkref #0: rel 1663/25322/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EEC40, prev 0/3A4EEBF8, desc: INSERT_LEAF off 286, blkref #0: rel 1663/25322/2674 blk 39
rmgr: XLOG        len (rec/tot):     49/   137, tx:     128892, lsn: 0/3A4EEC88, prev 0/3A4EEC40, desc: FPI , blkref #0: rel 1663/25322/25328 blk 0 FPW
rmgr: Heap        len (rec/tot):    188/   188, tx:     128892, lsn: 0/3A4EED18, prev 0/3A4EEC88, desc: INPLACE off 3, blkref #0: rel 1663/25322/1259 blk 0
rmgr: Heap        len (rec/tot):    188/   188, tx:     128892, lsn: 0/3A4EEDD8, prev 0/3A4EED18, desc: INPLACE off 4, blkref #0: rel 1663/25322/1259 blk 0
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4EEE98, prev 0/3A4EEDD8, desc: HOT_UPDATE off 2 xmax 128892 flags 0x60 ; new off 5 xmax 0, blkref #0: rel 1663/
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4EEEE8, prev 0/3A4EEE98, desc: INSERT off 118 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EEF38, prev 0/3A4EEEE8, desc: INSERT_LEAF off 58, blkref #0: rel 1663/25322/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EEF80, prev 0/3A4EEF38, desc: INSERT_LEAF off 284, blkref #0: rel 1663/25322/2674 blk 39
rmgr: Storage     len (rec/tot):     42/    42, tx:     128892, lsn: 0/3A4EEFC8, prev 0/3A4EEF80, desc: CREATE base/25322/25329
rmgr: Standby     len (rec/tot):     42/    42, tx:     128892, lsn: 0/3A4EEFF8, prev 0/3A4EEFC8, desc: LOCK xid 128892 db 25322 rel 25329
rmgr: Heap        len (rec/tot):    203/   203, tx:     128892, lsn: 0/3A4EF028, prev 0/3A4EEFF8, desc: INSERT off 6 flags 0x00, blkref #0: rel 1663/25322/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EF0F8, prev 0/3A4EF028, desc: INSERT_LEAF off 109, blkref #0: rel 1663/25322/2662 blk 2
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EF138, prev 0/3A4EF0F8, desc: INSERT_LEAF off 88, blkref #0: rel 1663/25322/2663 blk 2
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EF180, prev 0/3A4EF138, desc: INSERT_LEAF off 49, blkref #0: rel 1663/25322/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:     128892, lsn: 0/3A4EF1C0, prev 0/3A4EF180, desc: INSERT off 23 flags 0x00, blkref #0: rel 1663/25322/1249 blk 52
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EF270, prev 0/3A4EF1C0, desc: INSERT_LEAF off 198, blkref #0: rel 1663/25322/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EF2B0, prev 0/3A4EF270, desc: INSERT_LEAF off 313, blkref #0: rel 1663/25322/2659 blk 9
rmgr: Heap        len (rec/tot):    197/   197, tx:     128892, lsn: 0/3A4EF2F0, prev 0/3A4EF2B0, desc: INSERT off 21 flags 0x00, blkref #0: rel 1663/25322/2610 blk 3
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EF3B8, prev 0/3A4EF2F0, desc: INSERT_LEAF off 155, blkref #0: rel 1663/25322/2678 blk 1
rmgr: Btree       len (rec/tot):     64/    64, tx:     128892, lsn: 0/3A4EF3F8, prev 0/3A4EF3B8, desc: INSERT_LEAF off 156, blkref #0: rel 1663/25322/2679 blk 1
rmgr: Heap        len (rec/tot):     54/  1826, tx:     128892, lsn: 0/3A4EF438, prev 0/3A4EF3F8, desc: INSERT off 3 flags 0x01, blkref #0: rel 1663/25322/2606 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:     128892, lsn: 0/3A4EFB60, prev 0/3A4EF438, desc: INSERT_LEAF off 3, blkref #0: rel 1663/25322/2579 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   209, tx:     128892, lsn: 0/3A4EFC00, prev 0/3A4EFB60, desc: INSERT_LEAF off 2, blkref #0: rel 1663/25322/2664 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   209, tx:     128892, lsn: 0/3A4EFCD8, prev 0/3A4EFC00, desc: INSERT_LEAF off 3, blkref #0: rel 1663/25322/2665 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:     128892, lsn: 0/3A4EFDB0, prev 0/3A4EFCD8, desc: INSERT_LEAF off 1, blkref #0: rel 1663/25322/2666 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:     128892, lsn: 0/3A4EFE50, prev 0/3A4EFDB0, desc: INSERT_LEAF off 3, blkref #0: rel 1663/25322/2667 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4EFEF0, prev 0/3A4EFE50, desc: INSERT off 119 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EFF40, prev 0/3A4EFEF0, desc: INSERT_LEAF off 63, blkref #0: rel 1663/25322/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4EFF88, prev 0/3A4EFF40, desc: INSERT_LEAF off 285, blkref #0: rel 1663/25322/2674 blk 39
rmgr: Heap        len (rec/tot):     80/    80, tx:     128892, lsn: 0/3A4EFFD0, prev 0/3A4EFF88, desc: INSERT off 120 flags 0x00, blkref #0: rel 1663/25322/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     128892, lsn: 0/3A4F0038, prev 0/3A4EFFD0, desc: INSERT_LEAF off 61, blkref #0: rel 1663/25322/2673 blk 33
rmgr: Btree       len (rec/tot):     53/  6413, tx:     128892, lsn: 0/3A4F0080, prev 0/3A4F0038, desc: INSERT_LEAF off 226, blkref #0: rel 1663/25322/2674 blk 29 FPW
rmgr: XLOG        len (rec/tot):     49/   137, tx:     128892, lsn: 0/3A4F1990, prev 0/3A4F0080, desc: FPI , blkref #0: rel 1663/25322/25329 blk 0 FPW
rmgr: Heap        len (rec/tot):    188/   188, tx:     128892, lsn: 0/3A4F1A20, prev 0/3A4F1990, desc: INPLACE off 5, blkref #0: rel 1663/25322/1259 blk 0
rmgr: Heap        len (rec/tot):    188/   188, tx:     128892, lsn: 0/3A4F1AE0, prev 0/3A4F1A20, desc: INPLACE off 6, blkref #0: rel 1663/25322/1259 blk 0
rmgr: Transaction len (rec/tot):   1397/  1397, tx:     128892, lsn: 0/3A4F1BA0, prev 0/3A4F1AE0, desc: COMMIT 2024-01-16 11:15:32.770012 MSK; inval msgs: catcache 50 catcache 49 catca
rmgr: Heap        len (rec/tot):     61/    61, tx:     128893, lsn: 0/3A4F2130, prev 0/3A4F1BA0, desc: INSERT+INIT off 1 flags 0x00, blkref #0: rel 1663/25322/25323 blk 0
rmgr: Btree       len (rec/tot):    102/   102, tx:     128893, lsn: 0/3A4F2170, prev 0/3A4F2130, desc: NEWROOT lev 0, blkref #0: rel 1663/25322/25329 blk 1, blkref #2: rel 1663/25322/
rmgr: Btree       len (rec/tot):     64/    64, tx:     128893, lsn: 0/3A4F21D8, prev 0/3A4F2170, desc: INSERT_LEAF off 1, blkref #0: rel 1663/25322/25329 blk 1
rmgr: Heap        len (rec/tot):     61/    61, tx:     128893, lsn: 0/3A4F2218, prev 0/3A4F21D8, desc: INSERT off 2 flags 0x00, blkref #0: rel 1663/25322/25323 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     128893, lsn: 0/3A4F2258, prev 0/3A4F2218, desc: INSERT_LEAF off 2, blkref #0: rel 1663/25322/25329 blk 1
rmgr: Heap        len (rec/tot):     61/    61, tx:     128893, lsn: 0/3A4F2298, prev 0/3A4F2258, desc: INSERT off 3 flags 0x00, blkref #0: rel 1663/25322/25323 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     128893, lsn: 0/3A4F22D8, prev 0/3A4F2298, desc: INSERT_LEAF off 3, blkref #0: rel 1663/25322/25329 blk 1
rmgr: Transaction len (rec/tot):     34/    34, tx:     128893, lsn: 0/3A4F2318, prev 0/3A4F22D8, desc: COMMIT 2024-01-16 11:15:32.796388 MSK
```

Вначале (до первой операции COMMIT) происходит активная работа с таблицами и индексами системного каталога. За счет этого размер записей и получился существенно больше, чем в демонстрации.

3. Восстановление после сбоя

Обновляем строки:

```
=> UPDATE t SET s = 'FOO';
```

```
UPDATE 3
```

```
=> BEGIN;
```

```
BEGIN
```

```
=> UPDATE t SET s = 'BAR'; -- не фиксируем транзакцию
```

```
UPDATE 3
```

Прерываем основной серверный процесс.

```
student$ sudo head -n 1 /var/lib/postgresql/13/main/postmaster.pid
```

```
42119
```

```
student$ sudo kill -QUIT 42119
```

```
student$ sudo pg_ctlcluster 13 main status
```

```
pg_ctl: no server running
```

Запускаем сервер.

```
student$ sudo pg_ctlcluster 13 main start
```

Проверяем изменения:

```
student$ psql wal_log
```

```
=> SELECT * FROM t;

 id |  s
----+-----
  1 | FOO
  2 | FOO
  3 | FOO
(3 rows)
```

Журнал сообщений:

```
postgres$ tail -n 6 /var/log/postgresql/postgresql-13-main.log

2024-01-16 11:15:34.215 MSK [43007] LOG:  database system was interrupted; last known up at 2024-01-16 11:15:32 MSK
2024-01-16 11:15:42.868 MSK [43007] LOG:  database system was not properly shut down; automatic recovery in progress
2024-01-16 11:15:42.888 MSK [43007] LOG:  redo starts at 0/3A4D2D98
2024-01-16 11:15:42.892 MSK [43007] LOG:  invalid record length at 0/3A4F2440: wanted 24, got 0
2024-01-16 11:15:42.892 MSK [43007] LOG:  redo done at 0/3A4F2418
2024-01-16 11:15:43.240 MSK [43006] LOG:  database system is ready to accept connections
```