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;

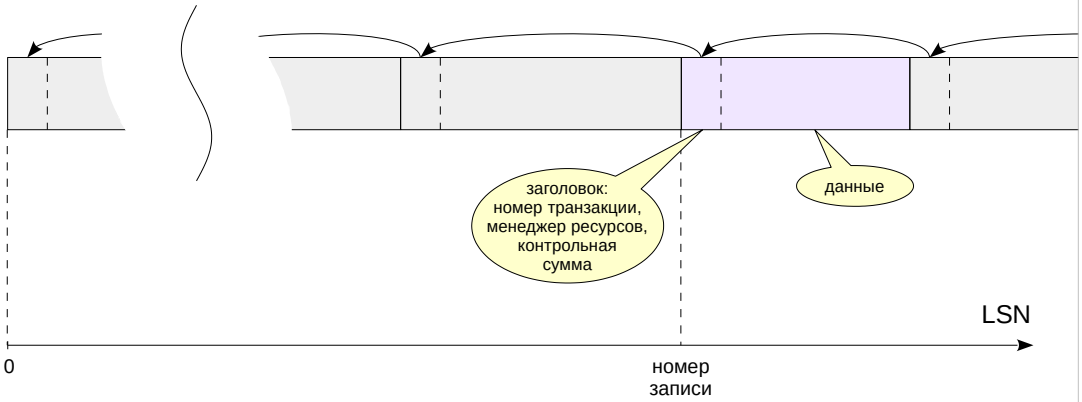
- файловые операции (создание и удаление файлов и каталогов, например, создание файлов при создании таблицы) — так как эти операции должны происходить синхронно с изменением данных.

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
```

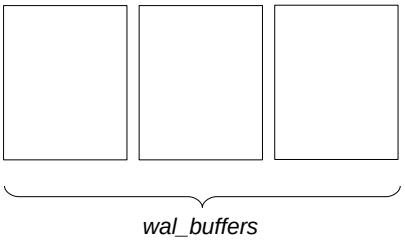
LSN выводится как два 32-битных числа в шестнадцатеричной системе через косую черту.

Текущая позиция в журнале:

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

```
 pg_current_wal_insert_lsn
---------------------------
 1/52950E08
(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
--------------------------+----------+------------------------
 000000010000000100000053 | 16777216 | 2023-01-19 13:43:23+03
 000000010000000100000052 | 16777216 | 2023-01-19 14:30:50+03
 000000010000000100000059 | 16777216 | 2023-01-19 13:43:14+03
 000000010000000100000055 | 16777216 | 2023-01-19 13:43:18+03
 000000010000000100000056 | 16777216 | 2023-01-19 13:43:26+03
 000000010000000100000054 | 16777216 | 2023-01-19 13:43:18+03
 00000001000000010000005A | 16777216 | 2023-01-19 14:30:30+03
 000000010000000100000058 | 16777216 | 2023-01-19 13:43:19+03
 000000010000000100000057 | 16777216 | 2023-01-19 13:43:22+03
(9 rows)
```

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

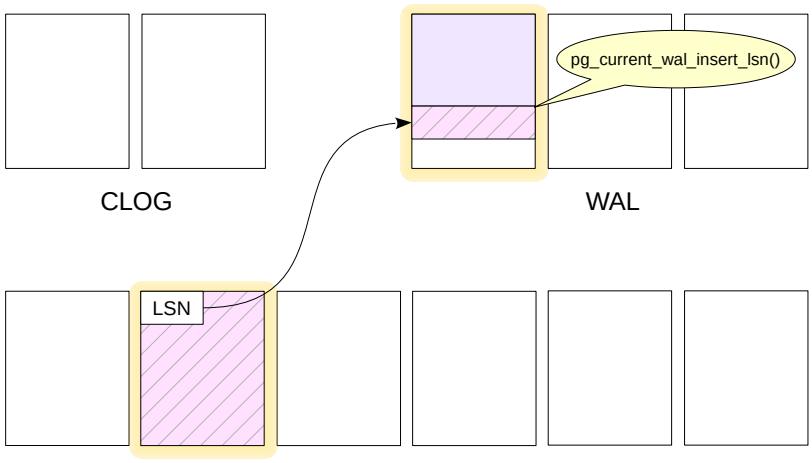
Размер файлов можно задать при инициализации кластера, по умолчанию — 16 Мбайт.

Текущая позиция находится в этом файле:

```
=> SELECT pg_walfile_name('1/52950E08');
```

```
     pg_walfile_name
--------------------------
 000000010000000100000052
(1 row)
```

Упреждающая запись



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

- буферный кеш (размером *shared_buffers*),

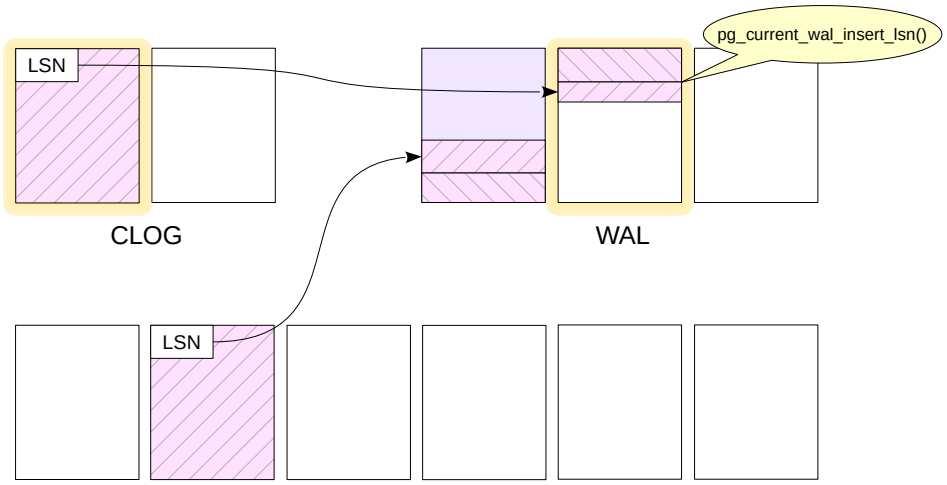
- только что рассмотренный журнальный кеш WAL (размером *wal_buffers*),

- 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

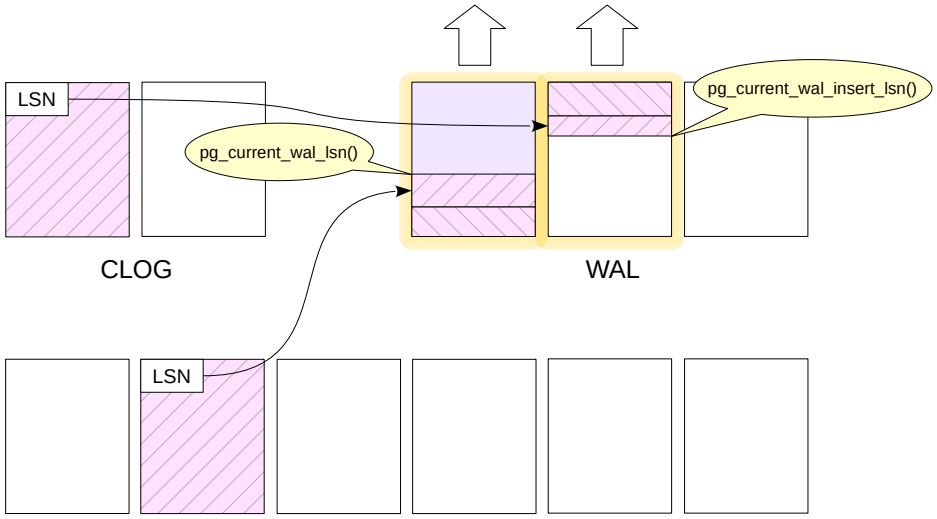


Допустим, далее происходит фиксация транзакции. Для этого формируется журнальная запись, меняется бит состояния на странице CLOG и ссылка на эту запись проставляется в поле LSN измененной страницы.

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.

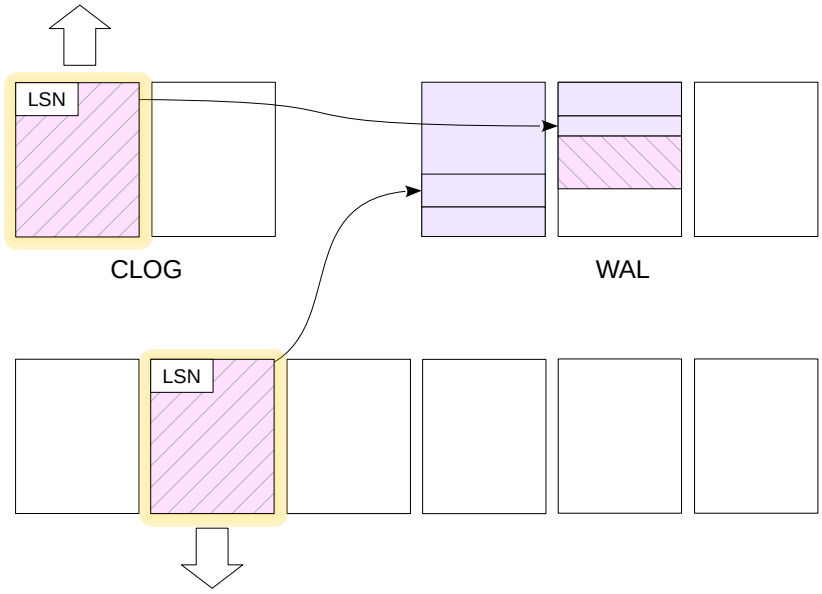
Упреждающая запись



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
---------------------------+--------------------------
 1/52A5CF00                | 000000010000000100000052
(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
---------------------------
 1/52A5CF48
(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
------------
 1/52A5CF48
(1 row)
```

Завершим транзакцию.

```
=> COMMIT;
```

```
COMMIT
```

Позиция в журнале снова изменилась:

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

```
 pg_current_wal_insert_lsn
---------------------------
 1/52A5CF70
(1 row)
```

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

```
=> SELECT '1/52A5CF70'::pg_lsn - '1/52A5CF00'::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 1/52A5CF00 -e 1/52A5CF70 000000010000000100000052
```

```
rmgr: Heap        len (rec/tot):     69/    69, tx:    1204413, lsn: 1/52A5CF00, prev 1/52A5CEB8, desc: HOT_UPDATE off 1 xmax 1204413 flags 0x41 ; new off 2 xmax 0, blkref #0: rel 1663
rmgr: Transaction len (rec/tot):     34/    34, tx:    1204413, lsn: 1/52A5CF48, prev 1/52A5CF00, desc: COMMIT 2023-01-19 14:30:51.432596 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
---------------------------
 1/7DBFB248
(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
---------------------------
 1/7DC1A058
(1 row)
```

Размер журнальных записей:

```
=> SELECT '1/7DC1A058'::pg_lsn - '1/7DBFB248'::pg_lsn;

 ?column?
----------
   126480
(1 row)
```

2. Состав журнальных записей

Журнальный файл:

```
=> SELECT pg_walfile_name('1/7DBFB248');

     pg_walfile_name
--------------------------
 00000001000000010000007D
(1 row)
```

Смотрим записи:

```
postgres$ /usr/lib/postgresql/13/bin/pg_waldump -p /var/lib/postgresql/13/main/pg_wal -s 1/7DBFB248 -e 1/7DC1A058 00000001000000010000007D

rmgr: Storage     len (rec/tot):     42/    42, tx:          0, lsn: 1/7DBFB248, prev 1/7DBFAA00, desc: CREATE base/111927/111928
rmgr: Heap        len (rec/tot):     54/  1518, tx:    1359337, lsn: 1/7DBFB278, prev 1/7DBFB248, desc: INSERT off 8 flags 0x01, blkref #0: rel 1663/111927/1247 blk 9 FPW
rmgr: Btree       len (rec/tot):     53/  1013, tx:    1359337, lsn: 1/7DBFB868, prev 1/7DBFB278, desc: INSERT_LEAF off 46, blkref #0: rel 1663/111927/2703 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  2021, tx:    1359337, lsn: 1/7DBFBC60, prev 1/7DBFB868, desc: INSERT_LEAF off 23, blkref #0: rel 1663/111927/2704 blk 2 FPW
rmgr: Heap        len (rec/tot):     54/  6798, tx:    1359337, lsn: 1/7DBFC460, prev 1/7DBFBC60, desc: INSERT off 112 flags 0x01, blkref #0: rel 1663/111927/2608 blk 56 FPW
rmgr: Btree       len (rec/tot):     53/  4621, tx:    1359337, lsn: 1/7DBFDEF0, prev 1/7DBFC460, desc: INSERT_LEAF off 162, blkref #0: rel 1663/111927/2673 blk 24 FPW
rmgr: Btree       len (rec/tot):     53/  8009, tx:    1359337, lsn: 1/7DBFF118, prev 1/7DBFDEF0, desc: INSERT_LEAF off 283, blkref #0: rel 1663/111927/2674 blk 39 FPW
rmgr: Heap        len (rec/tot):    207/   207, tx:    1359337, lsn: 1/7DC01080, prev 1/7DBFF118, desc: INSERT off 9 flags 0x00, blkref #0: rel 1663/111927/1247 blk 9
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC01150, prev 1/7DC01080, desc: INSERT_LEAF off 46, blkref #0: rel 1663/111927/2703 blk 2
rmgr: Btree       len (rec/tot):     53/  5873, tx:    1359337, lsn: 1/7DC01190, prev 1/7DC01150, desc: INSERT_LEAF off 65, blkref #0: rel 1663/111927/2704 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC028A0, prev 1/7DC01190, desc: INSERT off 113 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC028F0, prev 1/7DC028A0, desc: INSERT_LEAF off 162, blkref #0: rel 1663/111927/2673 blk 24
rmgr: Btree       len (rec/tot):     53/  2185, tx:    1359337, lsn: 1/7DC02938, prev 1/7DC028F0, desc: INSERT_LEAF off 75, blkref #0: rel 1663/111927/2674 blk 41 FPW
rmgr: Heap        len (rec/tot):     54/   874, tx:    1359337, lsn: 1/7DC031C8, prev 1/7DC02938, desc: INSERT off 2 flags 0x01, blkref #0: rel 1663/111927/1259 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/  2213, tx:    1359337, lsn: 1/7DC03538, prev 1/7DC031C8, desc: INSERT_LEAF off 106, blkref #0: rel 1663/111927/2662 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  3845, tx:    1359337, lsn: 1/7DC03DE0, prev 1/7DC03538, desc: INSERT_LEAF off 87, blkref #0: rel 1663/111927/2663 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  1013, tx:    1359337, lsn: 1/7DC04D00, prev 1/7DC03DE0, desc: INSERT_LEAF off 46, blkref #0: rel 1663/111927/3455 blk 4 FPW
rmgr: Heap        len (rec/tot):     54/  7498, tx:    1359337, lsn: 1/7DC050F8, prev 1/7DC04D00, desc: INSERT off 31 flags 0x01, blkref #0: rel 1663/111927/1249 blk 16 FPW
rmgr: Btree       len (rec/tot):     53/  6665, tx:    1359337, lsn: 1/7DC06E60, prev 1/7DC050F8, desc: INSERT_LEAF off 179, blkref #0: rel 1663/111927/2658 blk 14 FPW
rmgr: Btree       len (rec/tot):     53/  5973, tx:    1359337, lsn: 1/7DC08888, prev 1/7DC06E60, desc: INSERT_LEAF off 294, blkref #0: rel 1663/111927/2659 blk 9 FPW
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC09FE0, prev 1/7DC08888, desc: INSERT off 32 flags 0x00, blkref #0: rel 1663/111927/1249 blk 16
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0A0A8, prev 1/7DC09FE0, desc: INSERT_LEAF off 180, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0A0E8, prev 1/7DC0A0A8, desc: INSERT_LEAF off 295, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC0A128, prev 1/7DC0A0E8, desc: INSERT off 33 flags 0x00, blkref #0: rel 1663/111927/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC0A1D8, prev 1/7DC0A128, desc: INSERT_LEAF off 179, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0A220, prev 1/7DC0A1D8, desc: INSERT_LEAF off 294, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC0A260, prev 1/7DC0A220, desc: INSERT off 34 flags 0x00, blkref #0: rel 1663/111927/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC0A310, prev 1/7DC0A260, desc: INSERT_LEAF off 182, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0A358, prev 1/7DC0A310, desc: INSERT_LEAF off 294, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC0A398, prev 1/7DC0A358, desc: INSERT off 36 flags 0x00, blkref #0: rel 1663/111927/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC0A448, prev 1/7DC0A398, desc: INSERT_LEAF off 179, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0A490, prev 1/7DC0A448, desc: INSERT_LEAF off 294, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC0A4D0, prev 1/7DC0A490, desc: INSERT off 38 flags 0x00, blkref #0: rel 1663/111927/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC0A580, prev 1/7DC0A4D0, desc: INSERT_LEAF off 183, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0A5C8, prev 1/7DC0A580, desc: INSERT_LEAF off 294, blkref #0: rel 1663/111927/2659 blk 9
rmgr: XLOG        len (rec/tot):     49/  8241, tx:    1359337, lsn: 1/7DC0A608, prev 1/7DC0A5C8, desc: FPI_FOR_HINT , blkref #0: rel 1663/111927/1249 fork fsm blk 2 FPW
rmgr: Heap        len (rec/tot):     54/  1558, tx:    1359337, lsn: 1/7DC0C658, prev 1/7DC0A608, desc: INSERT off 10 flags 0x01, blkref #0: rel 1663/111927/1249 blk 52 FPW
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC0CC70, prev 1/7DC0C658, desc: INSERT_LEAF off 179, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0CCB8, prev 1/7DC0CC70, desc: INSERT_LEAF off 294, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC0CCF8, prev 1/7DC0CCB8, desc: INSERT off 11 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC0CDA8, prev 1/7DC0CCF8, desc: INSERT_LEAF off 184, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC0CDF0, prev 1/7DC0CDA8, desc: INSERT_LEAF off 294, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC0CE30, prev 1/7DC0CDF0, desc: INSERT off 114 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     53/  1849, tx:    1359337, lsn: 1/7DC0CE80, prev 1/7DC0CE30, desc: INSERT_LEAF off 57, blkref #0: rel 1663/111927/2673 blk 33 FPW
rmgr: Btree       len (rec/tot):     53/  6105, tx:    1359337, lsn: 1/7DC0D5C0, prev 1/7DC0CE80, desc: INSERT_LEAF off 4, blkref #0: rel 1663/111927/2674 blk 37 FPW
rmgr: Heap        len (rec/tot):     54/  4118, tx:    1359337, lsn: 1/7DC0EDB8, prev 1/7DC0D5C0, desc: INSERT off 56 flags 0x00, blkref #0: rel 1664/0/1214 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/  1661, tx:    1359337, lsn: 1/7DC0FDD0, prev 1/7DC0EDB8, desc: INSERT_LEAF off 56, blkref #0: rel 1664/0/1232 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/  1213, tx:    1359337, lsn: 1/7DC10468, prev 1/7DC0FDD0, desc: INSERT_LEAF off 56, blkref #0: rel 1664/0/1233 blk 1 FPW
rmgr: Standby     len (rec/tot):     42/    42, tx:    1359337, lsn: 1/7DC10928, prev 1/7DC10468, desc: LOCK xid 1359337 db 111927 rel 111928
rmgr: Storage     len (rec/tot):     42/    42, tx:    1359337, lsn: 1/7DC10958, prev 1/7DC10928, desc: CREATE base/111927/111931
rmgr: Heap        len (rec/tot):    207/   207, tx:    1359337, lsn: 1/7DC10988, prev 1/7DC10958, desc: INSERT off 10 flags 0x00, blkref #0: rel 1663/111927/1247 blk 9
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC10A58, prev 1/7DC10988, desc: INSERT_LEAF off 48, blkref #0: rel 1663/111927/2703 blk 2
rmgr: Btree       len (rec/tot):     53/  6113, tx:    1359337, lsn: 1/7DC10A98, prev 1/7DC10A58, desc: INSERT_LEAF off 111, blkref #0: rel 1663/111927/2704 blk 4 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC12298, prev 1/7DC10A98, desc: INSERT off 115 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC122E8, prev 1/7DC12298, desc: INSERT_LEAF off 164, blkref #0: rel 1663/111927/2673 blk 24
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC12330, prev 1/7DC122E8, desc: INSERT_LEAF off 284, blkref #0: rel 1663/111927/2674 blk 39
rmgr: Heap        len (rec/tot):    203/   203, tx:    1359337, lsn: 1/7DC12378, prev 1/7DC12330, desc: INSERT off 3 flags 0x00, blkref #0: rel 1663/111927/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC12448, prev 1/7DC12378, desc: INSERT_LEAF off 107, blkref #0: rel 1663/111927/2662 blk 2
rmgr: Btree       len (rec/tot):     53/  3173, tx:    1359337, lsn: 1/7DC12488, prev 1/7DC12448, desc: INSERT_LEAF off 44, blkref #0: rel 1663/111927/2663 blk 5 FPW
```

```
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC130F0, prev 1/7DC12488, desc: INSERT_LEAF off 47, blkref #0: rel 1663/111927/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13130, prev 1/7DC130F0, desc: INSERT off 12 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC131E0, prev 1/7DC13130, desc: INSERT_LEAF off 187, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13228, prev 1/7DC131E0, desc: INSERT_LEAF off 302, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13268, prev 1/7DC13228, desc: INSERT off 13 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC13318, prev 1/7DC13268, desc: INSERT_LEAF off 188, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13360, prev 1/7DC13318, desc: INSERT_LEAF off 303, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC133A0, prev 1/7DC13360, desc: INSERT off 14 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC13450, prev 1/7DC133A0, desc: INSERT_LEAF off 187, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13498, prev 1/7DC13450, desc: INSERT_LEAF off 304, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC134D8, prev 1/7DC13498, desc: INSERT off 15 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC13588, prev 1/7DC134D8, desc: INSERT_LEAF off 190, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC135D0, prev 1/7DC13588, desc: INSERT_LEAF off 302, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13610, prev 1/7DC135D0, desc: INSERT off 16 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC136C0, prev 1/7DC13610, desc: INSERT_LEAF off 191, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13708, prev 1/7DC136C0, desc: INSERT_LEAF off 302, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13748, prev 1/7DC13708, desc: INSERT off 17 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC137F8, prev 1/7DC13748, desc: INSERT_LEAF off 190, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13840, prev 1/7DC137F8, desc: INSERT_LEAF off 302, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13880, prev 1/7DC13840, desc: INSERT off 18 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC13930, prev 1/7DC13880, desc: INSERT_LEAF off 192, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13978, prev 1/7DC13930, desc: INSERT_LEAF off 302, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC139B8, prev 1/7DC13978, desc: INSERT off 19 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC13A68, prev 1/7DC139B8, desc: INSERT_LEAF off 190, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13AB0, prev 1/7DC13A68, desc: INSERT_LEAF off 302, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13AF0, prev 1/7DC13AB0, desc: INSERT off 20 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC13BA0, prev 1/7DC13AF0, desc: INSERT_LEAF off 193, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13BE8, prev 1/7DC13BA0, desc: INSERT_LEAF off 302, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Storage     len (rec/tot):     42/    42, tx:    1359337, lsn: 1/7DC13C28, prev 1/7DC13BE8, desc: CREATE base/111927/111933
rmgr: Standby     len (rec/tot):     42/    42, tx:    1359337, lsn: 1/7DC13C58, prev 1/7DC13C28, desc: LOCK xid 1359337 db 111927 rel 111933
rmgr: Heap        len (rec/tot):    203/   203, tx:    1359337, lsn: 1/7DC13C88, prev 1/7DC13C58, desc: INSERT off 4 flags 0x00, blkref #0: rel 1663/111927/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13D58, prev 1/7DC13C88, desc: INSERT_LEAF off 108, blkref #0: rel 1663/111927/2662 blk 2
rmgr: Btree       len (rec/tot):     88/    88, tx:    1359337, lsn: 1/7DC13D98, prev 1/7DC13D58, desc: INSERT_LEAF off 45, blkref #0: rel 1663/111927/2663 blk 5
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13DF0, prev 1/7DC13D98, desc: INSERT_LEAF off 48, blkref #0: rel 1663/111927/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13E30, prev 1/7DC13DF0, desc: INSERT off 21 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC13EE0, prev 1/7DC13E30, desc: INSERT_LEAF off 196, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC13F28, prev 1/7DC13EE0, desc: INSERT_LEAF off 311, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC13F68, prev 1/7DC13F28, desc: INSERT off 22 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC14030, prev 1/7DC13F68, desc: INSERT_LEAF off 197, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC14078, prev 1/7DC14030, desc: INSERT_LEAF off 312, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):     54/  3590, tx:    1359337, lsn: 1/7DC140B8, prev 1/7DC14078, desc: INSERT off 20 flags 0x01, blkref #0: rel 1663/111927/2610 blk 3 FPW
rmgr: Btree       len (rec/tot):     53/  3193, tx:    1359337, lsn: 1/7DC14EC0, prev 1/7DC140B8, desc: INSERT_LEAF off 155, blkref #0: rel 1663/111927/2678 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/  3193, tx:    1359337, lsn: 1/7DC15B40, prev 1/7DC14EC0, desc: INSERT_LEAF off 155, blkref #0: rel 1663/111927/2679 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC167D8, prev 1/7DC15B40, desc: INSERT off 116 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC16828, prev 1/7DC167D8, desc: INSERT_LEAF off 58, blkref #0: rel 1663/111927/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC16870, prev 1/7DC16828, desc: INSERT_LEAF off 285, blkref #0: rel 1663/111927/2674 blk 39
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC168B8, prev 1/7DC16870, desc: INSERT off 117 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC16908, prev 1/7DC168B8, desc: INSERT_LEAF off 59, blkref #0: rel 1663/111927/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC16950, prev 1/7DC16908, desc: INSERT_LEAF off 286, blkref #0: rel 1663/111927/2674 blk 39
rmgr: XLOG        len (rec/tot):     49/   137, tx:    1359337, lsn: 1/7DC16998, prev 1/7DC16950, desc: FPI , blkref #0: rel 1663/111927/111933 blk 0 FPW
rmgr: Heap        len (rec/tot):    188/   188, tx:    1359337, lsn: 1/7DC16A28, prev 1/7DC16998, desc: INPLACE off 3, blkref #0: rel 1663/111927/1259 blk 0
rmgr: Heap        len (rec/tot):    188/   188, tx:    1359337, lsn: 1/7DC16AE8, prev 1/7DC16A28, desc: INPLACE off 4, blkref #0: rel 1663/111927/1259 blk 0
rmgr: Heap        len (rec/tot):     81/    81, tx:    1359337, lsn: 1/7DC16BA8, prev 1/7DC16AE8, desc: HOT_UPDATE off 2 xmax 1359337 flags 0x60 ; new off 5 xmax 0, blkref #0: rel 1663
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC16C00, prev 1/7DC16BA8, desc: INSERT off 118 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC16C50, prev 1/7DC16C00, desc: INSERT_LEAF off 58, blkref #0: rel 1663/111927/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC16C98, prev 1/7DC16C50, desc: INSERT_LEAF off 284, blkref #0: rel 1663/111927/2674 blk 39
rmgr: Storage     len (rec/tot):     42/    42, tx:    1359337, lsn: 1/7DC16CE0, prev 1/7DC16C98, desc: CREATE base/111927/111934
rmgr: Standby     len (rec/tot):     42/    42, tx:    1359337, lsn: 1/7DC16D10, prev 1/7DC16CE0, desc: LOCK xid 1359337 db 111927 rel 111934
rmgr: Heap        len (rec/tot):    203/   203, tx:    1359337, lsn: 1/7DC16D40, prev 1/7DC16D10, desc: INSERT off 6 flags 0x00, blkref #0: rel 1663/111927/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC16E10, prev 1/7DC16D40, desc: INSERT_LEAF off 109, blkref #0: rel 1663/111927/2662 blk 2
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC16E50, prev 1/7DC16E10, desc: INSERT_LEAF off 88, blkref #0: rel 1663/111927/2663 blk 2
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC16E98, prev 1/7DC16E50, desc: INSERT_LEAF off 49, blkref #0: rel 1663/111927/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:    1359337, lsn: 1/7DC16ED8, prev 1/7DC16E98, desc: INSERT off 23 flags 0x00, blkref #0: rel 1663/111927/1249 blk 52
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC16F88, prev 1/7DC16ED8, desc: INSERT_LEAF off 198, blkref #0: rel 1663/111927/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC16FC8, prev 1/7DC16F88, desc: INSERT_LEAF off 313, blkref #0: rel 1663/111927/2659 blk 9
rmgr: Heap        len (rec/tot):    197/   197, tx:    1359337, lsn: 1/7DC17008, prev 1/7DC16FC8, desc: INSERT off 21 flags 0x00, blkref #0: rel 1663/111927/2610 blk 3
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC170D0, prev 1/7DC17008, desc: INSERT_LEAF off 155, blkref #0: rel 1663/111927/2678 blk 1
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359337, lsn: 1/7DC17110, prev 1/7DC170D0, desc: INSERT_LEAF off 156, blkref #0: rel 1663/111927/2679 blk 1
rmgr: Heap        len (rec/tot):     54/  1826, tx:    1359337, lsn: 1/7DC17150, prev 1/7DC17110, desc: INSERT off 3 flags 0x01, blkref #0: rel 1663/111927/2606 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:    1359337, lsn: 1/7DC17878, prev 1/7DC17150, desc: INSERT_LEAF off 3, blkref #0: rel 1663/111927/2579 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   209, tx:    1359337, lsn: 1/7DC17918, prev 1/7DC17878, desc: INSERT_LEAF off 2, blkref #0: rel 1663/111927/2664 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   209, tx:    1359337, lsn: 1/7DC179F0, prev 1/7DC17918, desc: INSERT_LEAF off 3, blkref #0: rel 1663/111927/2665 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:    1359337, lsn: 1/7DC17AC8, prev 1/7DC179F0, desc: INSERT_LEAF off 1, blkref #0: rel 1663/111927/2666 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:    1359337, lsn: 1/7DC17B68, prev 1/7DC17AC8, desc: INSERT_LEAF off 3, blkref #0: rel 1663/111927/2667 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC17C08, prev 1/7DC17B68, desc: INSERT off 119 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC17C58, prev 1/7DC17C08, desc: INSERT_LEAF off 63, blkref #0: rel 1663/111927/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC17CA0, prev 1/7DC17C58, desc: INSERT_LEAF off 285, blkref #0: rel 1663/111927/2674 blk 39
rmgr: Heap        len (rec/tot):     80/    80, tx:    1359337, lsn: 1/7DC17CE8, prev 1/7DC17CA0, desc: INSERT off 120 flags 0x00, blkref #0: rel 1663/111927/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:    1359337, lsn: 1/7DC17D38, prev 1/7DC17CE8, desc: INSERT_LEAF off 61, blkref #0: rel 1663/111927/2673 blk 33
rmgr: Btree       len (rec/tot):     53/  6413, tx:    1359337, lsn: 1/7DC17D80, prev 1/7DC17D38, desc: INSERT_LEAF off 226, blkref #0: rel 1663/111927/2674 blk 29 FPW
rmgr: XLOG        len (rec/tot):     49/   137, tx:    1359337, lsn: 1/7DC196A8, prev 1/7DC17D80, desc: FPI , blkref #0: rel 1663/111927/111934 blk 0 FPW
rmgr: Heap        len (rec/tot):    188/   188, tx:    1359337, lsn: 1/7DC19738, prev 1/7DC196A8, desc: INPLACE off 5, blkref #0: rel 1663/111927/1259 blk 0
rmgr: Heap        len (rec/tot):    188/   188, tx:    1359337, lsn: 1/7DC197F8, prev 1/7DC19738, desc: INPLACE off 6, blkref #0: rel 1663/111927/1259 blk 0
rmgr: Transaction len (rec/tot):   1397/  1397, tx:    1359337, lsn: 1/7DC198B8, prev 1/7DC197F8, desc: COMMIT 2023-01-19 14:40:12.116518 MSK; inval msgs: catcache 50 catcache 49 catca
rmgr: Heap        len (rec/tot):     61/    61, tx:    1359338, lsn: 1/7DC19E30, prev 1/7DC198B8, desc: INSERT+INIT off 1 flags 0x00, blkref #0: rel 1663/111927/111928 blk 0
rmgr: Btree       len (rec/tot):    102/   102, tx:    1359338, lsn: 1/7DC19E70, prev 1/7DC19E30, desc: NEWROOT lev 0, blkref #0: rel 1663/111927/111934 blk 1, blkref #2: rel 1663/1119
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359338, lsn: 1/7DC19ED8, prev 1/7DC19E70, desc: INSERT_LEAF off 1, blkref #0: rel 1663/111927/111934 blk 1
rmgr: Heap        len (rec/tot):     61/    61, tx:    1359338, lsn: 1/7DC19F18, prev 1/7DC19ED8, desc: INSERT off 2 flags 0x00, blkref #0: rel 1663/111927/111928 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359338, lsn: 1/7DC19F58, prev 1/7DC19F18, desc: INSERT_LEAF off 2, blkref #0: rel 1663/111927/111934 blk 1
rmgr: Heap        len (rec/tot):     61/    61, tx:    1359338, lsn: 1/7DC19F98, prev 1/7DC19F58, desc: INSERT off 3 flags 0x00, blkref #0: rel 1663/111927/111928 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:    1359338, lsn: 1/7DC19FD8, prev 1/7DC19F98, desc: INSERT_LEAF off 3, blkref #0: rel 1663/111927/111934 blk 1
rmgr: Transaction len (rec/tot):     34/    34, tx:    1359338, lsn: 1/7DC1A030, prev 1/7DC19FD8, desc: COMMIT 2023-01-19 14:40:12.141432 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
```

```
158580
```

```
student$ sudo kill -9 158580
```

```
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

2023-01-19 14:40:13.829 MSK [159471] LOG:  database system was interrupted; last known up at 2023-01-19 14:40:11 MSK
2023-01-19 14:40:14.552 MSK [159471] LOG:  database system was not properly shut down; automatic recovery in progress
2023-01-19 14:40:14.556 MSK [159471] LOG:  redo starts at 1/7DBFA8F8
2023-01-19 14:40:14.558 MSK [159471] LOG:  invalid record length at 1/7DC1A158: wanted 24, got 0
2023-01-19 14:40:14.558 MSK [159471] LOG:  redo done at 1/7DC1A130
2023-01-19 14:40:14.608 MSK [159470] LOG:  database system is ready to accept connections
```