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;

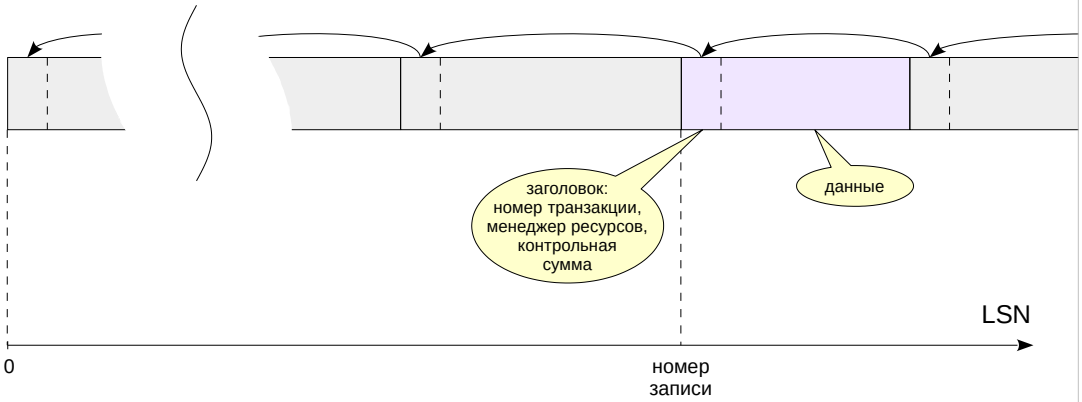
- файловые операции (создание и удаление файлов и каталогов, например, создание файлов при создании таблицы) — так как эти операции должны происходить синхронно с изменением данных.

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
```

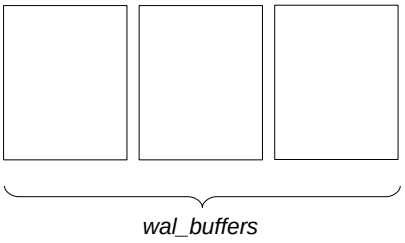
LSN выводится как два 32-битных числа в шестнадцатеричной системе через косую черту.

Текущая позиция в журнале:

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

```
 pg_current_wal_insert_lsn
---------------------------
 0/91816A28
(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
--------------------------+----------+------------------------
 000000010000000000000092 | 16777216 | 2022-12-16 19:14:20+03
 000000010000000000000095 | 16777216 | 2022-12-16 19:14:24+03
 000000010000000000000098 | 16777216 | 2022-12-16 19:14:18+03
 000000010000000000000096 | 16777216 | 2022-12-16 19:14:19+03
 000000010000000000000097 | 16777216 | 2022-12-16 19:14:23+03
 000000010000000000000093 | 16777216 | 2022-12-16 19:14:21+03
 000000010000000000000091 | 16777216 | 2022-12-16 19:45:00+03
 000000010000000000000094 | 16777216 | 2022-12-16 19:14:18+03
(8 rows)
```

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

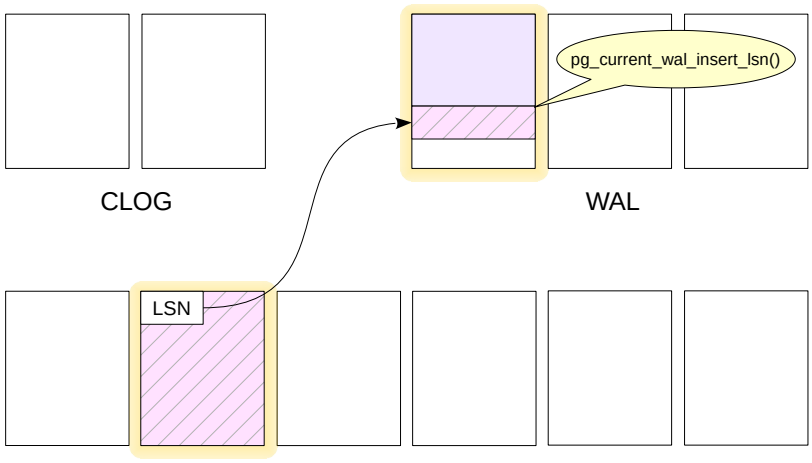
Размер файлов можно задать при инициализации кластера, а в дальнейшем изменить утилитой pg_resetwal. Размер по умолчанию — 16 Мбайт.

Текущая позиция находится в этом файле:

```
=> SELECT pg_walfile_name('0/91816A28');
```

```
     pg_walfile_name
--------------------------
 000000010000000000000091
(1 row)
```

Упреждающая запись



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

- буферный кеш (размером *shared_buffers*),

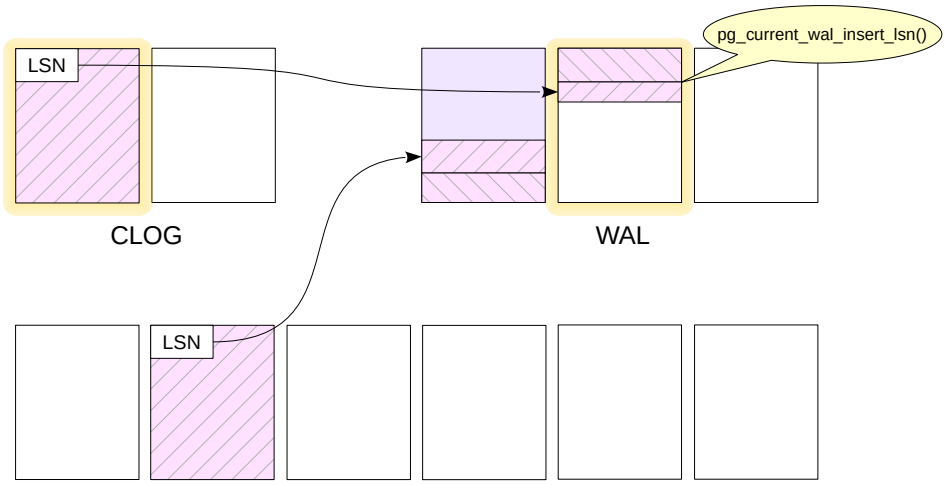
- только что рассмотренный журнальный кеш WAL (размером *wal_buffers*),

- 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

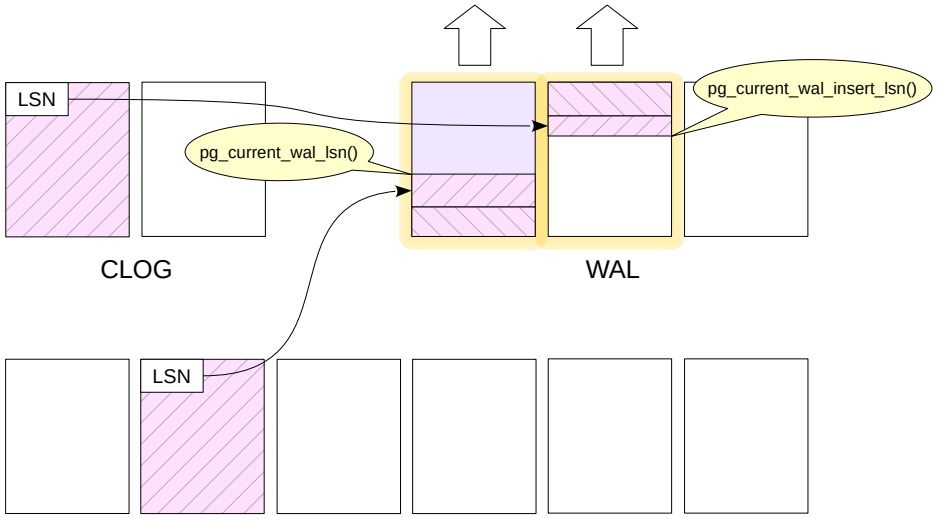


Допустим, далее происходит фиксация транзакции. Для этого формируется журнальная запись, меняется бит состояния на странице CLOG и ссылка на эту запись проставляется в поле LSN измененной страницы.

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.

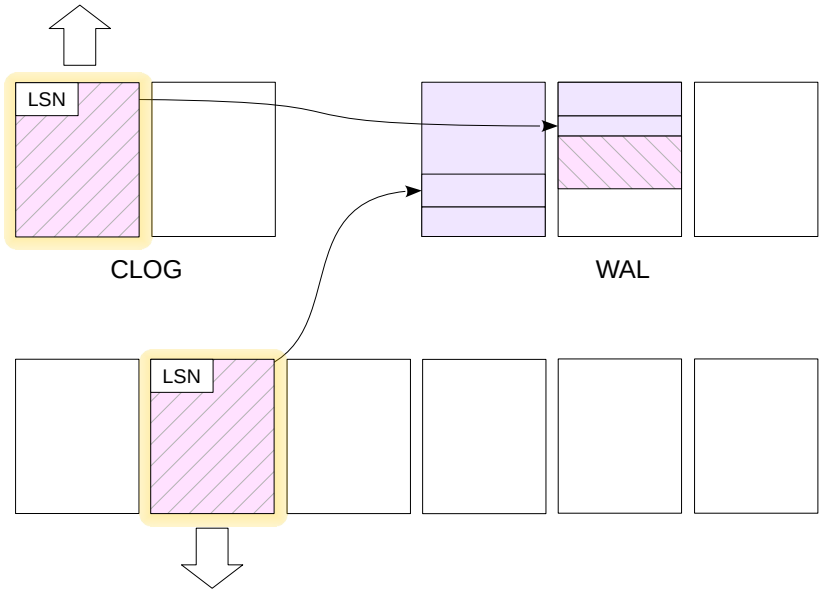
Упреждающая запись



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
---------------------------+--------------------------
 0/91922870                | 000000010000000000000091
(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
---------------------------
 0/919228B8
(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
------------
 0/919228B8
(1 row)
```

Завершим транзакцию.

```
=> COMMIT;
```

```
COMMIT
```

Позиция в журнале снова изменилась:

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

```
 pg_current_wal_insert_lsn
---------------------------
 0/919228E0
(1 row)
```

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

```
=> SELECT '0/919228E0'::pg_lsn - '0/91922870'::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 0/91922870 -e 0/919228E0 000000010000000000000091
```

```
rmgr: Heap        len (rec/tot):     69/    69, tx:     485459, lsn: 0/91922870, prev 0/91922828, desc: HOT_UPDATE off 1 xmax 485459 flags 0x41 ; new off 2 xmax 0, blkref #0: rel 1663/
rmgr: Transaction len (rec/tot):     34/    34, tx:     485459, lsn: 0/919228B8, prev 0/91922870, desc: COMMIT 2022-12-16 19:45:01.282838 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/BCACF558
(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/BCAED730
(1 row)
```

Размер журнальных записей:

```
=> SELECT '0/BCAED730'::pg_lsn - '0/BCACF558'::pg_lsn;

 ?column?
----------
   123352
(1 row)
```

2. Состав журнальных записей

Журнальный файл:

```
=> SELECT pg_walfile_name('0/BCACF558');

     pg_walfile_name
--------------------------
 000000010000000000000BC
(1 row)
```

Смотрим записи:

```
postgres$ /usr/lib/postgresql/13/bin/pg_waldump -p /var/lib/postgresql/13/main/pg_wal -s 0/BCACF558 -e 0/BCAED730 000000010000000000000BC

rmgr: Storage     len (rec/tot):     42/    42, tx:          0, lsn: 0/BCACF558, prev 0/BCACED10, desc: CREATE base/60090/60091
rmgr: Heap        len (rec/tot):     54/  1518, tx:     640315, lsn: 0/BCACF588, prev 0/BCACF558, desc: INSERT off 8 flags 0x01, blkref #0: rel 1663/60090/1247 blk 9 FPW
rmgr: Btree       len (rec/tot):     53/  1013, tx:     640315, lsn: 0/BCACFB78, prev 0/BCACF588, desc: INSERT_LEAF off 46, blkref #0: rel 1663/60090/2703 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  2021, tx:     640315, lsn: 0/BCACFF70, prev 0/BCACFB78, desc: INSERT_LEAF off 23, blkref #0: rel 1663/60090/2704 blk 2 FPW
rmgr: Heap        len (rec/tot):     54/  6798, tx:     640315, lsn: 0/BCAD0770, prev 0/BCACFF70, desc: INSERT off 112 flags 0x01, blkref #0: rel 1663/60090/2608 blk 56 FPW
rmgr: Btree       len (rec/tot):     53/  4621, tx:     640315, lsn: 0/BCAD2218, prev 0/BCAD0770, desc: INSERT_LEAF off 162, blkref #0: rel 1663/60090/2673 blk 24 FPW
rmgr: Btree       len (rec/tot):     53/  8009, tx:     640315, lsn: 0/BCAD3428, prev 0/BCAD2218, desc: INSERT_LEAF off 283, blkref #0: rel 1663/60090/2674 blk 39 FPW
rmgr: Heap        len (rec/tot):    207/   207, tx:     640315, lsn: 0/BCAD5390, prev 0/BCAD3428, desc: INSERT off 9 flags 0x00, blkref #0: rel 1663/60090/1247 blk 9
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAD5460, prev 0/BCAD5390, desc: INSERT_LEAF off 46, blkref #0: rel 1663/60090/2703 blk 2
rmgr: Btree       len (rec/tot):     53/  5873, tx:     640315, lsn: 0/BCAD54A0, prev 0/BCAD5460, desc: INSERT_LEAF off 65, blkref #0: rel 1663/60090/2704 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAD6BB0, prev 0/BCAD54A0, desc: INSERT off 113 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAD6C00, prev 0/BCAD6BB0, desc: INSERT_LEAF off 162, blkref #0: rel 1663/60090/2673 blk 24
rmgr: Btree       len (rec/tot):     53/  2185, tx:     640315, lsn: 0/BCAD6C48, prev 0/BCAD6C00, desc: INSERT_LEAF off 75, blkref #0: rel 1663/60090/2674 blk 41 FPW
rmgr: Heap        len (rec/tot):     54/   874, tx:     640315, lsn: 0/BCAD74D8, prev 0/BCAD6C48, desc: INSERT off 2 flags 0x01, blkref #0: rel 1663/60090/1259 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/  2213, tx:     640315, lsn: 0/BCAD7848, prev 0/BCAD74D8, desc: INSERT_LEAF off 106, blkref #0: rel 1663/60090/2662 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  3845, tx:     640315, lsn: 0/BCAD8108, prev 0/BCAD7848, desc: INSERT_LEAF off 87, blkref #0: rel 1663/60090/2663 blk 2 FPW
rmgr: Btree       len (rec/tot):     53/  1013, tx:     640315, lsn: 0/BCAD9010, prev 0/BCAD8108, desc: INSERT_LEAF off 46, blkref #0: rel 1663/60090/3455 blk 4 FPW
rmgr: Heap        len (rec/tot):     54/  7498, tx:     640315, lsn: 0/BCAD9408, prev 0/BCAD9010, desc: INSERT off 31 flags 0x01, blkref #0: rel 1663/60090/1249 blk 16 FPW
rmgr: Btree       len (rec/tot):     53/  6665, tx:     640315, lsn: 0/BCADB170, prev 0/BCAD9408, desc: INSERT_LEAF off 179, blkref #0: rel 1663/60090/2658 blk 14 FPW
rmgr: Btree       len (rec/tot):     53/  5973, tx:     640315, lsn: 0/BCADCB98, prev 0/BCADB170, desc: INSERT_LEAF off 294, blkref #0: rel 1663/60090/2659 blk 9 FPW
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCADE308, prev 0/BCADCB98, desc: INSERT off 32 flags 0x00, blkref #0: rel 1663/60090/1249 blk 16
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCADE3B8, prev 0/BCADE308, desc: INSERT_LEAF off 180, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCADE3F8, prev 0/BCADE3B8, desc: INSERT_LEAF off 295, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCADE438, prev 0/BCADE3F8, desc: INSERT off 33 flags 0x00, blkref #0: rel 1663/60090/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCADE4E8, prev 0/BCADE438, desc: INSERT_LEAF off 179, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCADE530, prev 0/BCADE4E8, desc: INSERT_LEAF off 294, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCADE570, prev 0/BCADE530, desc: INSERT off 34 flags 0x00, blkref #0: rel 1663/60090/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCADE620, prev 0/BCADE570, desc: INSERT_LEAF off 182, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCADE668, prev 0/BCADE620, desc: INSERT_LEAF off 294, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCADE6A8, prev 0/BCADE668, desc: INSERT off 36 flags 0x00, blkref #0: rel 1663/60090/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCADE758, prev 0/BCADE6A8, desc: INSERT_LEAF off 179, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCADE7A0, prev 0/BCADE758, desc: INSERT_LEAF off 294, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCADE7E0, prev 0/BCADE7A0, desc: INSERT off 38 flags 0x00, blkref #0: rel 1663/60090/1249 blk 16
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCADE890, prev 0/BCADE7E0, desc: INSERT_LEAF off 183, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCADE8D8, prev 0/BCADE890, desc: INSERT_LEAF off 294, blkref #0: rel 1663/60090/2659 blk 9
rmgr: XLOG        len (rec/tot):     49/  8241, tx:     640315, lsn: 0/BCADE918, prev 0/BCADE8D8, desc: FPI_FOR_HINT , blkref #0: rel 1663/60090/1249 fork fsm blk 2 FPW
rmgr: Heap        len (rec/tot):     54/  1558, tx:     640315, lsn: 0/BCAE0968, prev 0/BCADE918, desc: INSERT off 10 flags 0x01, blkref #0: rel 1663/60090/1249 blk 52 FPW
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE0F80, prev 0/BCAE0968, desc: INSERT_LEAF off 179, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE0FC8, prev 0/BCAE0F80, desc: INSERT_LEAF off 294, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE1008, prev 0/BCAE0FC8, desc: INSERT off 11 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE10B8, prev 0/BCAE1008, desc: INSERT_LEAF off 184, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE1100, prev 0/BCAE10B8, desc: INSERT_LEAF off 294, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAE1140, prev 0/BCAE1100, desc: INSERT off 114 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     53/  1849, tx:     640315, lsn: 0/BCAE1190, prev 0/BCAE1140, desc: INSERT_LEAF off 57, blkref #0: rel 1663/60090/2673 blk 33 FPW
rmgr: Btree       len (rec/tot):     53/  6105, tx:     640315, lsn: 0/BCAE18D0, prev 0/BCAE1190, desc: INSERT_LEAF off 4, blkref #0: rel 1663/60090/2674 blk 37 FPW
rmgr: Heap        len (rec/tot):     54/  4118, tx:     640315, lsn: 0/BCAE30C8, prev 0/BCAE18D0, desc: INSERT off 56 flags 0x00, blkref #0: rel 1664/0/1214 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/  1661, tx:     640315, lsn: 0/BCAE40F8, prev 0/BCAE30C8, desc: INSERT_LEAF off 56, blkref #0: rel 1664/0/1232 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/  1213, tx:     640315, lsn: 0/BCAE4778, prev 0/BCAE40F8, desc: INSERT_LEAF off 56, blkref #0: rel 1664/0/1233 blk 1 FPW
rmgr: Standby     len (rec/tot):     42/    42, tx:     640315, lsn: 0/BCAE4C38, prev 0/BCAE4778, desc: LOCK xid 640315 db 60090 rel 60091
rmgr: Storage     len (rec/tot):     42/    42, tx:     640315, lsn: 0/BCAE4C68, prev 0/BCAE4C38, desc: CREATE base/60090/60094
rmgr: Heap        len (rec/tot):    207/   207, tx:     640315, lsn: 0/BCAE4C98, prev 0/BCAE4C68, desc: INSERT off 10 flags 0x00, blkref #0: rel 1663/60090/1247 blk 9
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE4D68, prev 0/BCAE4C98, desc: INSERT_LEAF off 48, blkref #0: rel 1663/60090/2703 blk 2
rmgr: Btree       len (rec/tot):     53/  6113, tx:     640315, lsn: 0/BCAE4DA8, prev 0/BCAE4D68, desc: INSERT_LEAF off 147, blkref #0: rel 1663/60090/2704 blk 4 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAE65A8, prev 0/BCAE4DA8, desc: INSERT off 115 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE65F8, prev 0/BCAE65A8, desc: INSERT_LEAF off 164, blkref #0: rel 1663/60090/2673 blk 24
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE6640, prev 0/BCAE65F8, desc: INSERT_LEAF off 284, blkref #0: rel 1663/60090/2674 blk 39
rmgr: Heap        len (rec/tot):    203/   203, tx:     640315, lsn: 0/BCAE6688, prev 0/BCAE6640, desc: INSERT off 3 flags 0x00, blkref #0: rel 1663/60090/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE6758, prev 0/BCAE6688, desc: INSERT_LEAF off 107, blkref #0: rel 1663/60090/2662 blk 2
rmgr: Btree       len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAE6798, prev 0/BCAE6758, desc: INSERT_LEAF off 39, blkref #0: rel 1663/60090/2663 blk 2
```

```
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE67E8, prev 0/BCAE6798, desc: INSERT_LEAF off 47, blkref #0: rel 1663/60090/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE6828, prev 0/BCAE67E8, desc: INSERT off 12 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE68D8, prev 0/BCAE6828, desc: INSERT_LEAF off 187, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE6920, prev 0/BCAE68D8, desc: INSERT_LEAF off 302, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE6960, prev 0/BCAE6920, desc: INSERT off 13 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE6A10, prev 0/BCAE6960, desc: INSERT_LEAF off 188, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE6A58, prev 0/BCAE6A10, desc: INSERT_LEAF off 303, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE6A98, prev 0/BCAE6A58, desc: INSERT off 14 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE6B48, prev 0/BCAE6A98, desc: INSERT_LEAF off 187, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE6B90, prev 0/BCAE6B48, desc: INSERT_LEAF off 304, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE6BD0, prev 0/BCAE6B90, desc: INSERT off 15 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE6C80, prev 0/BCAE6BD0, desc: INSERT_LEAF off 190, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE6CC8, prev 0/BCAE6C80, desc: INSERT_LEAF off 302, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE6D08, prev 0/BCAE6CC8, desc: INSERT off 16 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE6DB8, prev 0/BCAE6D08, desc: INSERT_LEAF off 191, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE6E00, prev 0/BCAE6DB8, desc: INSERT_LEAF off 302, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE6E40, prev 0/BCAE6E00, desc: INSERT off 17 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE6EF0, prev 0/BCAE6E40, desc: INSERT_LEAF off 190, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE6F38, prev 0/BCAE6EF0, desc: INSERT_LEAF off 302, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE6F78, prev 0/BCAE6F38, desc: INSERT off 18 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE7028, prev 0/BCAE6F78, desc: INSERT_LEAF off 192, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE7070, prev 0/BCAE7028, desc: INSERT_LEAF off 302, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE70B0, prev 0/BCAE7070, desc: INSERT off 19 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE7160, prev 0/BCAE70B0, desc: INSERT_LEAF off 190, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE71A8, prev 0/BCAE7160, desc: INSERT_LEAF off 302, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE71E8, prev 0/BCAE71A8, desc: INSERT off 20 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE7298, prev 0/BCAE71E8, desc: INSERT_LEAF off 193, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE72E0, prev 0/BCAE7298, desc: INSERT_LEAF off 302, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Storage     len (rec/tot):     42/    42, tx:     640315, lsn: 0/BCAE7320, prev 0/BCAE72E0, desc: CREATE base/60090/60096
rmgr: Standby     len (rec/tot):     42/    42, tx:     640315, lsn: 0/BCAE7350, prev 0/BCAE7320, desc: LOCK xid 640315 db 60090 rel 60096
rmgr: Heap        len (rec/tot):    203/   203, tx:     640315, lsn: 0/BCAE7380, prev 0/BCAE7350, desc: INSERT off 4 flags 0x00, blkref #0: rel 1663/60090/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE7450, prev 0/BCAE7380, desc: INSERT_LEAF off 108, blkref #0: rel 1663/60090/2662 blk 2
rmgr: Btree       len (rec/tot):     88/    88, tx:     640315, lsn: 0/BCAE7490, prev 0/BCAE7450, desc: INSERT_LEAF off 40, blkref #0: rel 1663/60090/2663 blk 2
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE74E8, prev 0/BCAE7490, desc: INSERT_LEAF off 48, blkref #0: rel 1663/60090/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE7528, prev 0/BCAE74E8, desc: INSERT off 21 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE75D8, prev 0/BCAE7528, desc: INSERT_LEAF off 196, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE7620, prev 0/BCAE75D8, desc: INSERT_LEAF off 311, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAE7660, prev 0/BCAE7620, desc: INSERT off 22 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE7710, prev 0/BCAE7660, desc: INSERT_LEAF off 197, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAE7758, prev 0/BCAE7710, desc: INSERT_LEAF off 312, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):     54/  3590, tx:     640315, lsn: 0/BCAE7798, prev 0/BCAE7758, desc: INSERT off 20 flags 0x01, blkref #0: rel 1663/60090/2610 blk 3 FPW
rmgr: Btree       len (rec/tot):     53/  3193, tx:     640315, lsn: 0/BCAE85B8, prev 0/BCAE7798, desc: INSERT_LEAF off 155, blkref #0: rel 1663/60090/2678 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/  3193, tx:     640315, lsn: 0/BCAE9238, prev 0/BCAE85B8, desc: INSERT_LEAF off 155, blkref #0: rel 1663/60090/2679 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAE9EB8, prev 0/BCAE9238, desc: INSERT off 116 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE9F08, prev 0/BCAE9EB8, desc: INSERT_LEAF off 58, blkref #0: rel 1663/60090/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE9F50, prev 0/BCAE9F08, desc: INSERT_LEAF off 285, blkref #0: rel 1663/60090/2674 blk 39
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAE9F98, prev 0/BCAE9F50, desc: INSERT off 117 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAE9FE8, prev 0/BCAE9F98, desc: INSERT_LEAF off 59, blkref #0: rel 1663/60090/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAEA048, prev 0/BCAE9FE8, desc: INSERT_LEAF off 286, blkref #0: rel 1663/60090/2674 blk 39
rmgr: XLOG        len (rec/tot):     49/   137, tx:     640315, lsn: 0/BCAEA090, prev 0/BCAEA048, desc: FPI , blkref #0: rel 1663/60090/60096 blk 0 FPW
rmgr: Heap        len (rec/tot):    188/   188, tx:     640315, lsn: 0/BCAEA120, prev 0/BCAEA090, desc: INPLACE off 3, blkref #0: rel 1663/60090/1259 blk 0
rmgr: Heap        len (rec/tot):    188/   188, tx:     640315, lsn: 0/BCAEA1E0, prev 0/BCAEA120, desc: INPLACE off 4, blkref #0: rel 1663/60090/1259 blk 0
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAEA2A0, prev 0/BCAEA1E0, desc: HOT_UPDATE off 2 xmax 640315 flags 0x60 ; new off 5 xmax 0, blkref #0: rel 1663/
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAEA2F0, prev 0/BCAEA2A0, desc: INSERT off 118 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAEA340, prev 0/BCAEA2F0, desc: INSERT_LEAF off 58, blkref #0: rel 1663/60090/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAEA388, prev 0/BCAEA340, desc: INSERT_LEAF off 284, blkref #0: rel 1663/60090/2674 blk 39
rmgr: Storage     len (rec/tot):     42/    42, tx:     640315, lsn: 0/BCAEA3D0, prev 0/BCAEA388, desc: CREATE base/60090/60097
rmgr: Standby     len (rec/tot):     42/    42, tx:     640315, lsn: 0/BCAEA400, prev 0/BCAEA3D0, desc: LOCK xid 640315 db 60090 rel 60097
rmgr: Heap        len (rec/tot):    203/   203, tx:     640315, lsn: 0/BCAEA430, prev 0/BCAEA400, desc: INSERT off 6 flags 0x00, blkref #0: rel 1663/60090/1259 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAEA500, prev 0/BCAEA430, desc: INSERT_LEAF off 109, blkref #0: rel 1663/60090/2662 blk 2
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAEA540, prev 0/BCAEA500, desc: INSERT_LEAF off 90, blkref #0: rel 1663/60090/2663 blk 2
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAEA588, prev 0/BCAEA540, desc: INSERT_LEAF off 49, blkref #0: rel 1663/60090/3455 blk 4
rmgr: Heap        len (rec/tot):    175/   175, tx:     640315, lsn: 0/BCAEA5C8, prev 0/BCAEA588, desc: INSERT off 23 flags 0x00, blkref #0: rel 1663/60090/1249 blk 52
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAEA678, prev 0/BCAEA5C8, desc: INSERT_LEAF off 198, blkref #0: rel 1663/60090/2658 blk 14
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAEA6B8, prev 0/BCAEA678, desc: INSERT_LEAF off 313, blkref #0: rel 1663/60090/2659 blk 9
rmgr: Heap        len (rec/tot):    197/   197, tx:     640315, lsn: 0/BCAEA6F8, prev 0/BCAEA6B8, desc: INSERT off 21 flags 0x00, blkref #0: rel 1663/60090/2610 blk 3
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAEA7C0, prev 0/BCAEA6F8, desc: INSERT_LEAF off 155, blkref #0: rel 1663/60090/2678 blk 1
rmgr: Btree       len (rec/tot):     64/    64, tx:     640315, lsn: 0/BCAEA800, prev 0/BCAEA7C0, desc: INSERT_LEAF off 156, blkref #0: rel 1663/60090/2679 blk 1
rmgr: Heap        len (rec/tot):     54/  1826, tx:     640315, lsn: 0/BCAEA840, prev 0/BCAEA800, desc: INSERT off 3 flags 0x01, blkref #0: rel 1663/60090/2606 blk 0 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:     640315, lsn: 0/BCAEAF68, prev 0/BCAEA840, desc: INSERT_LEAF off 3, blkref #0: rel 1663/60090/2579 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   209, tx:     640315, lsn: 0/BCAEB008, prev 0/BCAEAF68, desc: INSERT_LEAF off 2, blkref #0: rel 1663/60090/2664 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   209, tx:     640315, lsn: 0/BCAEB0E0, prev 0/BCAEB008, desc: INSERT_LEAF off 3, blkref #0: rel 1663/60090/2665 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:     640315, lsn: 0/BCAEB1B8, prev 0/BCAEB0E0, desc: INSERT_LEAF off 1, blkref #0: rel 1663/60090/2666 blk 1 FPW
rmgr: Btree       len (rec/tot):     53/   153, tx:     640315, lsn: 0/BCAEB258, prev 0/BCAEB1B8, desc: INSERT_LEAF off 3, blkref #0: rel 1663/60090/2667 blk 1 FPW
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAEB2F8, prev 0/BCAEB258, desc: INSERT off 119 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAEB348, prev 0/BCAEB2F8, desc: INSERT_LEAF off 63, blkref #0: rel 1663/60090/2673 blk 33
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAEB390, prev 0/BCAEB348, desc: INSERT_LEAF off 285, blkref #0: rel 1663/60090/2674 blk 39
rmgr: Heap        len (rec/tot):     80/    80, tx:     640315, lsn: 0/BCAEB3D8, prev 0/BCAEB390, desc: INSERT off 120 flags 0x00, blkref #0: rel 1663/60090/2608 blk 56
rmgr: Btree       len (rec/tot):     72/    72, tx:     640315, lsn: 0/BCAEB428, prev 0/BCAEB3D8, desc: INSERT_LEAF off 61, blkref #0: rel 1663/60090/2673 blk 33
rmgr: Btree       len (rec/tot):     53/  6413, tx:     640315, lsn: 0/BCAEB470, prev 0/BCAEB428, desc: INSERT_LEAF off 226, blkref #0: rel 1663/60090/2674 blk 29 FPW
rmgr: XLOG        len (rec/tot):     49/   137, tx:     640315, lsn: 0/BCAECD98, prev 0/BCAEB470, desc: FPI , blkref #0: rel 1663/60090/60097 blk 0 FPW
rmgr: Heap        len (rec/tot):    188/   188, tx:     640315, lsn: 0/BCAECE28, prev 0/BCAECD98, desc: INPLACE off 5, blkref #0: rel 1663/60090/1259 blk 0
rmgr: Heap        len (rec/tot):    188/   188, tx:     640315, lsn: 0/BCAECEE8, prev 0/BCAECE28, desc: INPLACE off 6, blkref #0: rel 1663/60090/1259 blk 0
rmgr: Transaction len (rec/tot):   1397/  1397, tx:     640315, lsn: 0/BCAECFA8, prev 0/BCAECEE8, desc: COMMIT 2022-12-16 19:54:19.279291 MSK; inval msgs: catcache 50 catcache 49 catca
rmgr: Heap        len (rec/tot):     61/    61, tx:     640316, lsn: 0/BCAED520, prev 0/BCAECFA8, desc: INSERT+INIT off 1 flags 0x00, blkref #0: rel 1663/60090/60091 blk 0
rmgr: Btree       len (rec/tot):    102/   102, tx:     640316, lsn: 0/BCAED560, prev 0/BCAED520, desc: NEWROOT lev 0, blkref #0: rel 1663/60090/60097 blk 1, blkref #2: rel 1663/60090/
rmgr: Btree       len (rec/tot):     64/    64, tx:     640316, lsn: 0/BCAED5C8, prev 0/BCAED560, desc: INSERT_LEAF off 1, blkref #0: rel 1663/60090/60097 blk 1
rmgr: Heap        len (rec/tot):     61/    61, tx:     640316, lsn: 0/BCAED608, prev 0/BCAED5C8, desc: INSERT off 2 flags 0x00, blkref #0: rel 1663/60090/60091 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     640316, lsn: 0/BCAED648, prev 0/BCAED608, desc: INSERT_LEAF off 2, blkref #0: rel 1663/60090/60097 blk 1
rmgr: Heap        len (rec/tot):     61/    61, tx:     640316, lsn: 0/BCAED688, prev 0/BCAED648, desc: INSERT off 3 flags 0x00, blkref #0: rel 1663/60090/60091 blk 0
rmgr: Btree       len (rec/tot):     64/    64, tx:     640316, lsn: 0/BCAED6C8, prev 0/BCAED688, desc: INSERT_LEAF off 3, blkref #0: rel 1663/60090/60097 blk 1
rmgr: Transaction len (rec/tot):     34/    34, tx:     640316, lsn: 0/BCAED708, prev 0/BCAED6C8, desc: COMMIT 2022-12-16 19:54:19.305687 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
```

```
120849
```

```
student$ sudo kill -9 120849
```

```
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

2022-12-16 19:54:21.198 MSK [121737] LOG:  database system was interrupted; last known up at 2022-12-16 19:54:19 MSK
2022-12-16 19:54:21.645 MSK [121737] LOG:  database system was not properly shut down; automatic recovery in progress
2022-12-16 19:54:21.647 MSK [121737] LOG:  redo starts at 0/BCACEC08
2022-12-16 19:54:21.648 MSK [121737] LOG:  invalid record length at 0/BCAED830: wanted 24, got 0
2022-12-16 19:54:21.648 MSK [121737] LOG:  redo done at 0/BCAED808
2022-12-16 19:54:21.677 MSK [121736] LOG:  database system is ready to accept connections
```