



PROFESSIONAL  
Posigres

Упреждающий  
журнал



Журнал упреждающей записи (WAL) и его устройство

Процесс упреждающей записи

Надежность записи

Синхронный и асинхронный режимы

Уровни журнала

Настройки

## 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  
процесс упреждающей записи wal writer

при сбое происходит повторное выполнение операций из журнала,  
если это необходимо

## Другие применения

непрерывное архивирование, восстановление на произвольный момент  
физическая репликация и горячий резерв  
логическая репликация

# Что защищено журналом

Изменение любых страниц в буферном кэше

в том числе страницы таблиц и индексов

кроме нежурналируемых и временных таблиц

кроме hash-индексов

Фиксация и отмена транзакций (CLOG)

Файловые операции

создание и удаление файлов

создание и удаление каталогов

## Логическое

последовательность записей

записи защищены CRC

указатель на запись — LSN (log sequence number)

специальный тип `pg_lsn`

## В памяти

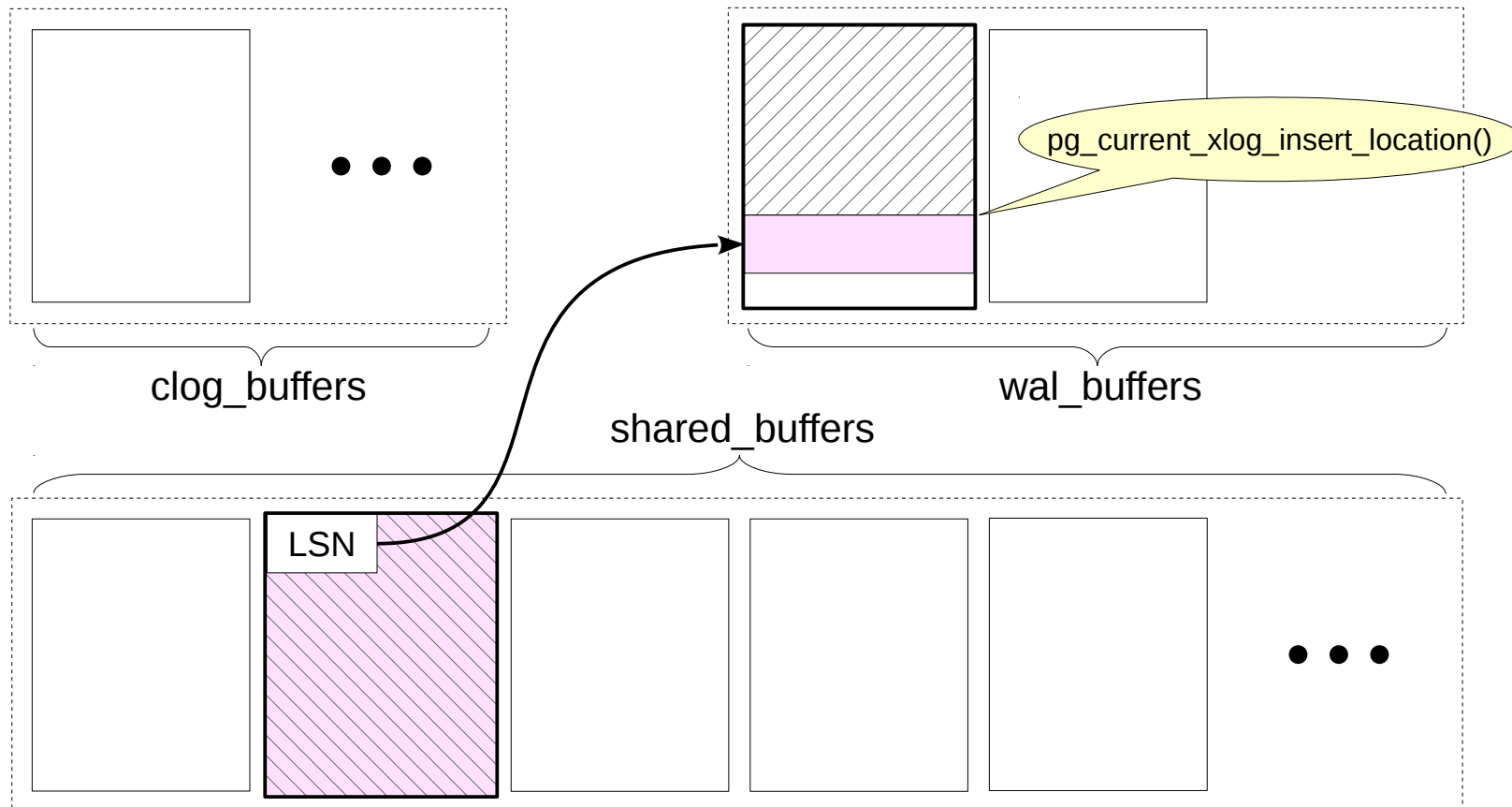
страницы

кольцевой буферный кэш (*wal\_buffers*)

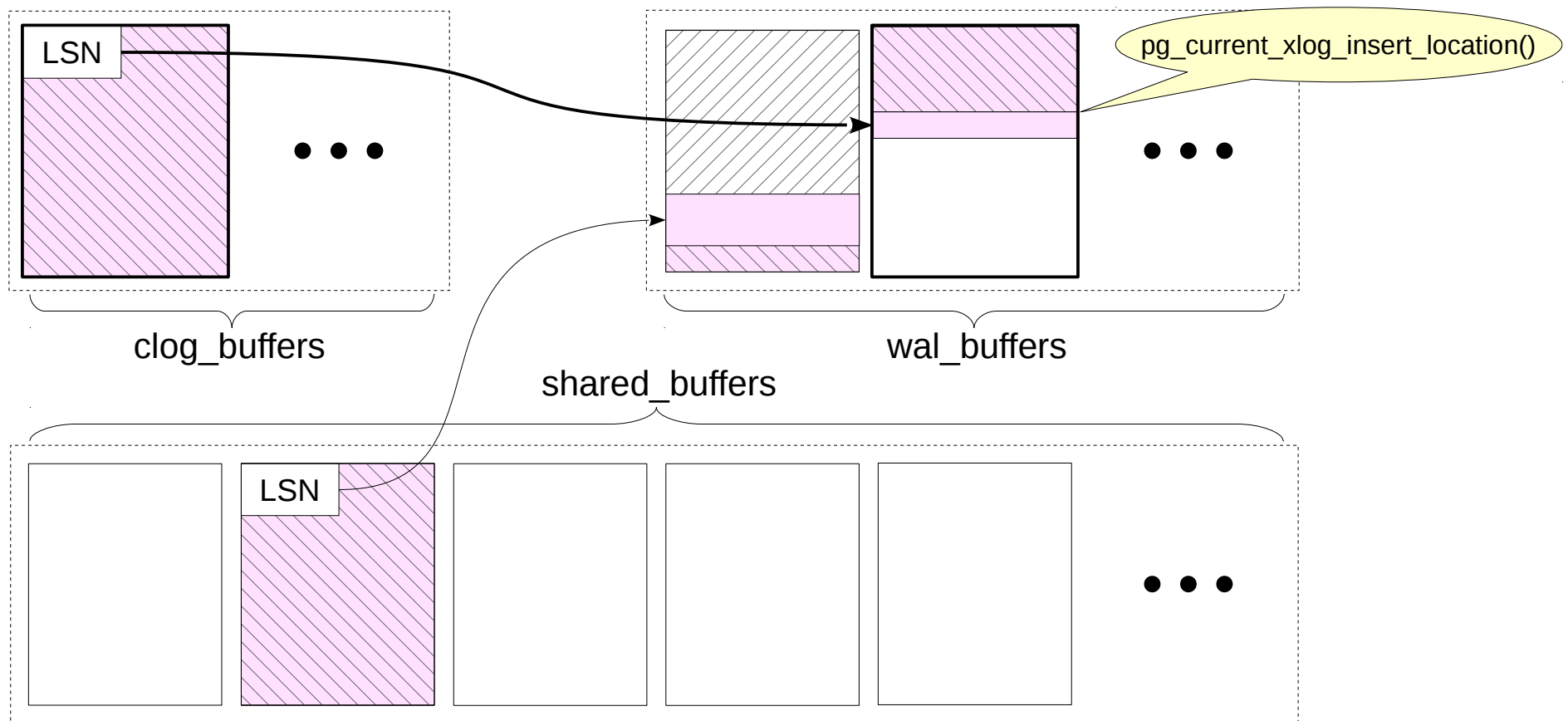
вытеснение

блокировки

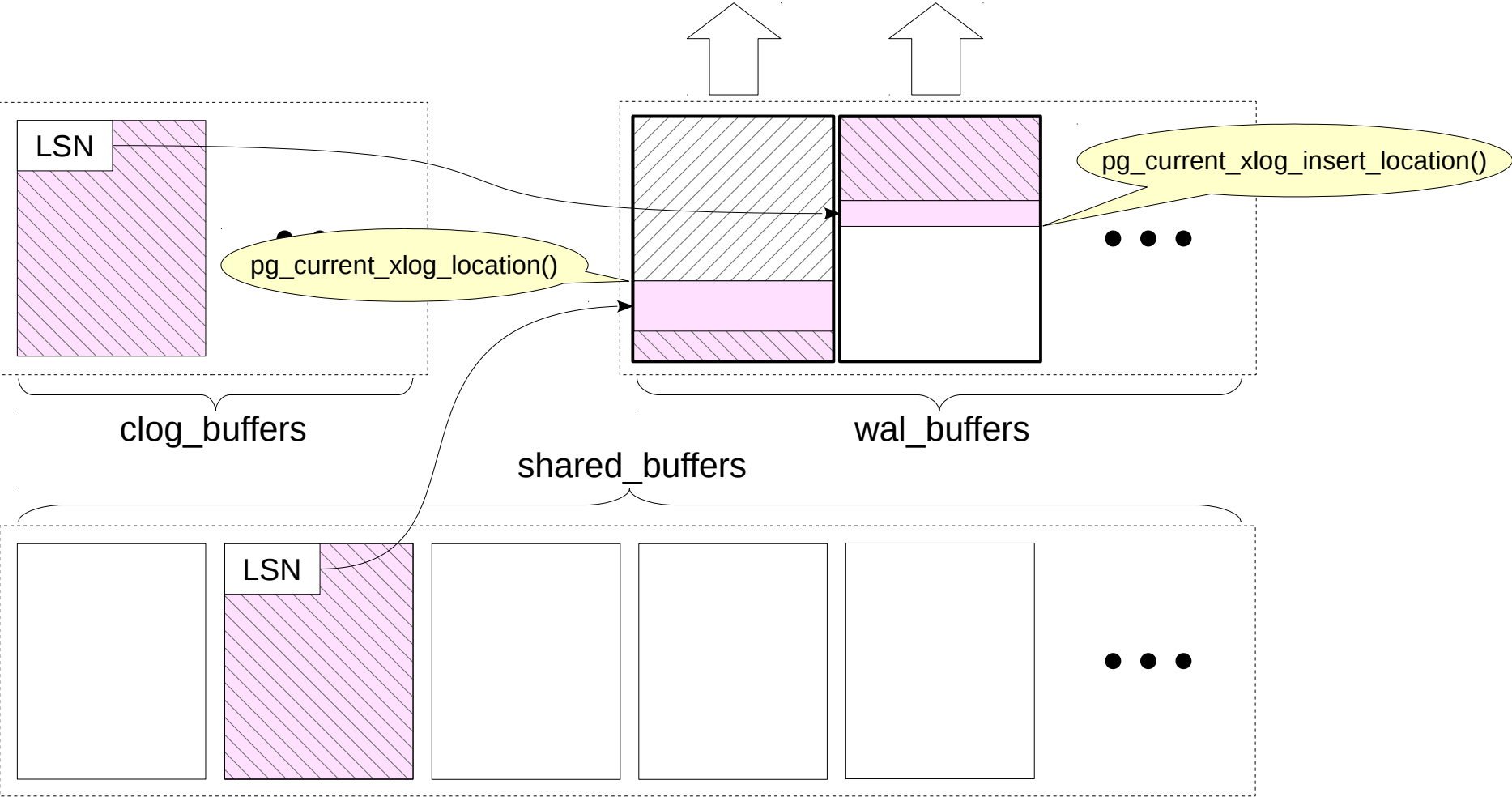
# Пример



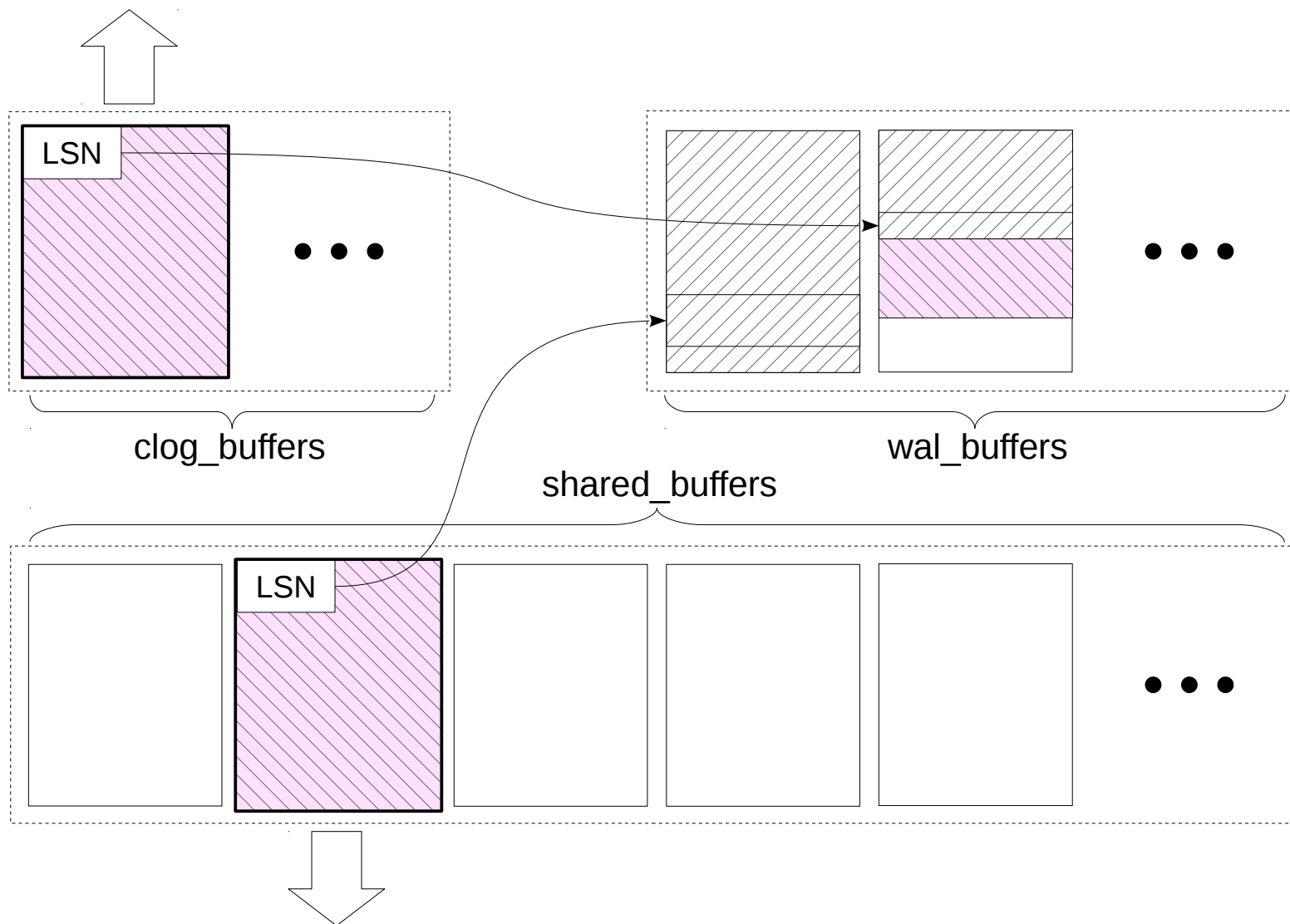
# Пример



# Пример



# Пример



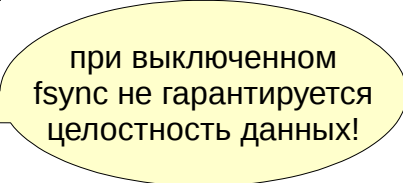
## Кэширование

данные должны дойти до энергонезависимого хранилища через кэши операционной системы, контроллера, диска  
настраивается способ синхронизации ОС;  
для выбора лучшего способа есть утилита `pg_test_fsync`,  
но в любом случае — дорогая операция  
журнал следует размещать на отдельной файловой системе  
если контроллер использует кэш, должна быть батарейка  
кэш диска на запись как правило следует отключать

## Настройки

`fsync = on`

`wal_sync_method`




при выключенном  
`fsync` не гарантируется  
целостность данных!

## Повреждение данных

записи журнала защищены CRC

можно включить контрольные суммы страниц  
(только при инициализации кластера: `initdb -k`)

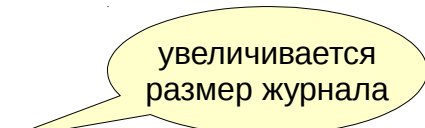


накладные  
расходы

## Атомарность записи страниц

запись образа страницы в журнал  
при первом изменении после контрольной точки

некоторые файловые системы могут гарантировать атомарность



увеличивается  
размер журнала

## Настройки

`full_page_writes` = on

`wal_compression` = off (сжатие образа страницы, 9.5)

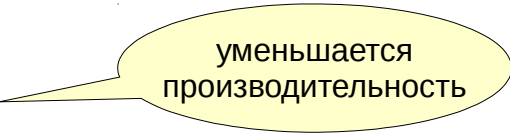
`wal_log_hints` = off (on подразумевается при контрольных суммах)

## Алгоритм

при фиксации изменений, если включен *synchronous\_commit*, а также перед записью страницы, если запись LSN еще не на диске: ждет *commit\_delay*, если активно не менее *commit\_siblings* транзакций записывает журнал до необходимого LSN

## Свойства

гарантируется долговечность



уменьшается  
производительность

## Настройки

*synchronous\_commit* = on  
*commit\_delay* = 0  
*commit\_siblings* = 5

## Алгоритм

циклы записи через *wal\_writer\_delay*

записывает только целиком заполненные страницы,  
но если новых полных страниц нет, записывает последнюю до конца

## Свойства

зафиксированные изменения могут пропасть ( $3 \times wal\_writer\_delay$ )

целостность данных тем не менее гарантируется

## Настройки

`synchronous_commit` — можно устанавливать в транзакции

`wal_writer_delay = 200ms`

Журнал удобно использовать для разных задач, если дополнить информацию

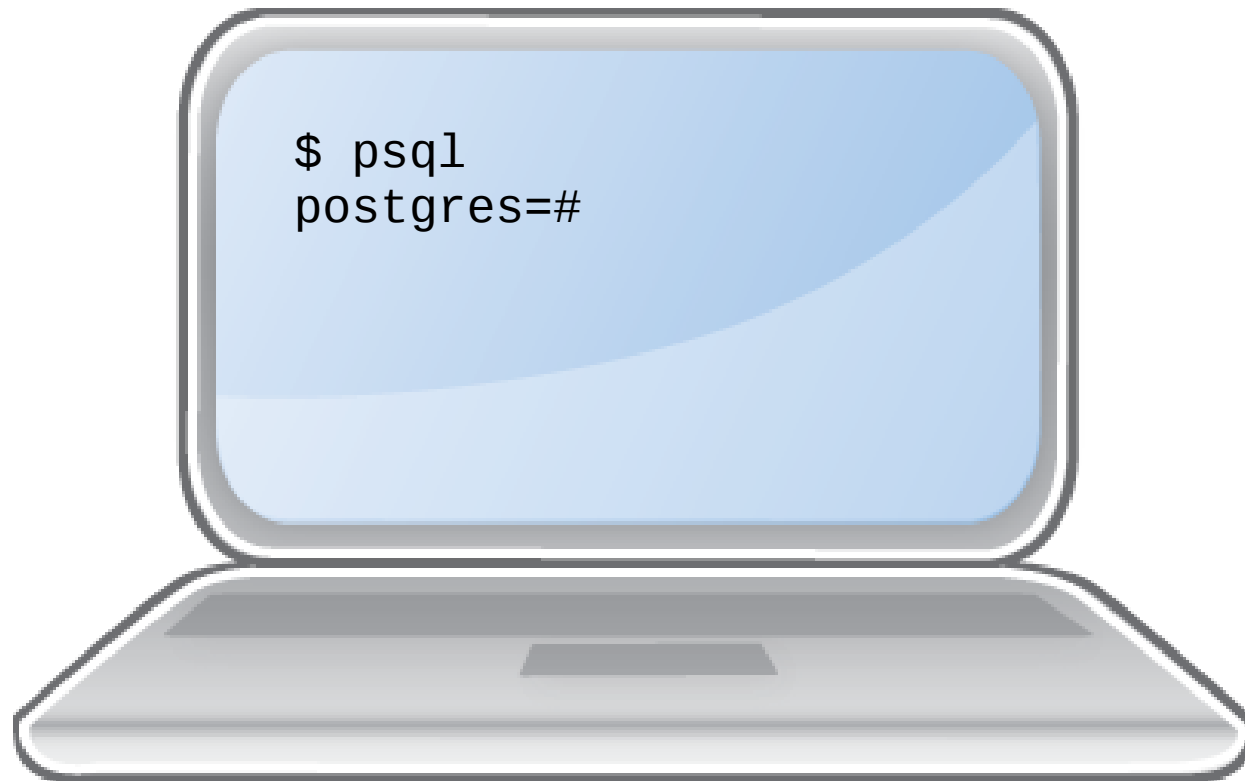
<i>уровень</i>	<i>задача</i>	<i>дополнительная информация</i>
minimal	восстановление после сбоя	—
archive	непрерывное архивирование	операции массовой обработки данных
hot_standby	горячий резерв	статус транзакций
logical	логическая репликация	реконструкция операций

## Настройка

`wal_level = minimal`

чем выше уровень, тем больше размер журнала

# Демонстрация



Журнал позволяет восстановить согласованность данных после сбоев, а также используется для других задач

Запись централизована в процессе wal writer

Настройка требует поиска баланса

между надежностью, размером и производительностью

1. Создайте базу DB8 и в ней таблицу с несколькими строками.
2. Установите параметры:
  - а) `full_page_writes = on`, `wal_compression = off`.
3. Выполните контрольную точку.
4. Обновите строку таблицы.
5. Проверьте, какие изменения были записаны в журнал при выполнении п. 4 и вычислите их размер в байтах.
6. Повторите пп. 3–5 с параметрами:
  - б) `full_page_writes = on`, `wal_compression = on`,
  - в) `full_page_writes = off`.
7. Сравните результаты.



### **Авторские права**

Курс «Администрирование PostgreSQL 9.5. Расширенный курс»

© Postgres Professional, 2016 год.

Авторы: Егор Рогов, Павел Лузанов

### **Использование материалов курса**

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

### **Обратная связь**

Отзывы, замечания и предложения направляйте по адресу:

[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

### **Отказ от ответственности**

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Журнал упреждающей записи (WAL) и его устройство

Процесс упреждающей записи

Надежность записи

Синхронный и асинхронный режимы

Уровни журнала

Настройки

## 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  
процесс упреждающей записи wal writer  
при сбое происходит повторное выполнение операций из журнала,  
если это необходимо

## Другие применения

непрерывное архивирование, восстановление на произвольный момент  
физическая репликация и горячий резерв  
логическая репликация

Основная причина существования журнала — восстановление согласованности данных в случае сбоя. Тем самым обеспечивается выполнение свойства долговечности (из набора свойств транзакций АСИД).

Одновременно с изменением данных в журнале создается запись, содержащая информацию, достаточную для повторения действий. Журнальная запись в обязательном порядке попадает на диск (или другое энергонезависимое устройство) до того, как туда попадают измененные данные. Журнала записывает либо тот процесс, которому это нужно для продолжения работы, либо специальный процесс упреждающей записи — wal writer.

В случае сбоя можно прочитать журнал и при необходимости повторно выполнить те действия, которые должны были произойти (подробно этот процесс рассматривается в теме «Контрольная точка»).

Кроме того, журнал удобно использовать и для других задач. Например, поток журнальных записей можно записывать в архив, чтобы получить возможность восстановления из резервной копии на произвольный момент. Можно передавать поток на другой сервер и «проигрывать» его там — так реализуется физическая репликация и горячий резерв (это рассматривается в четырех темах про репликацию). Кроме физической репликации можно реализовать и логическую, выбирая из журнала необходимую информацию об операциях.

<http://www.postgresql.org/docs/9.5/static/wal.html>

## Изменение любых страниц в буферном кэше

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

## Фиксация и отмена транзакций (CLOG)

## Файловые операции

создание и удаление файлов  
создание и удаление каталогов

Журналировать нужно все операции, при выполнении которых возможно рассогласование данных на диске.

В журнал записываются следующие действия:

- изменение страниц в буферном кэше (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 (и временными) таблицами — название говорит само за себя;
- операции с hash-индексами — не реализовано, так как hash-индексы не используются в качестве индексов (а служат для сопоставления функций хэширования различным типам данных).

## Логическое

- последовательность записей
- записи защищены CRC
- указатель на запись — LSN (log sequence number)
- специальный тип `pg_lsn`

## В памяти

- страницы
- кольцевой буферный кэш (`wal_buffers`)
- вытеснение
- блокировки

Логически журнал можно представить себе как последовательность записей различной длины. Каждая запись содержит данные о некоторой операции, а также служебную информацию и контрольную сумму.

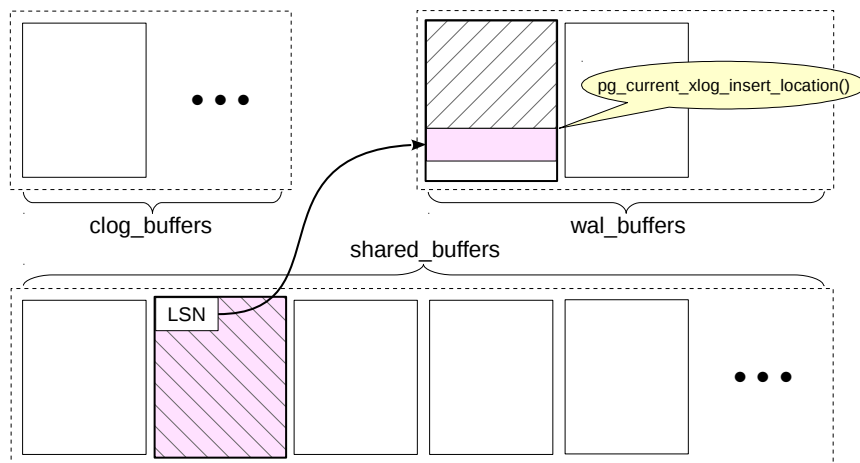
Для того, чтобы сослаться на определенную запись, используется тип данных `pg_lsn` (LSN = log sequence number) — 64-битное число.

<http://www.postgresql.org/docs/9.5/static/datatype-pg-lsn.html>

В оперативной памяти для журнала выделены специальные буферы. Размер кэша задается параметром `wal_buffers` (значение по умолчанию подразумевает автоматическую настройку). Журнальный кэш устроен наподобие буферного кэша (рассмотрен в теме «Буферный кэш») — имеется алгоритм вытеснения, кэш защищен необходимыми блокировками.

Важное отличие состоит в том, что журнальный кэш работает преимущественно в режиме кольцевого буфера: записи добавляются в «голову» буфера, а записываются на диск с «хвоста».

Необходимость в чтении старых страниц с диска возникает только при восстановлении после сбоя.

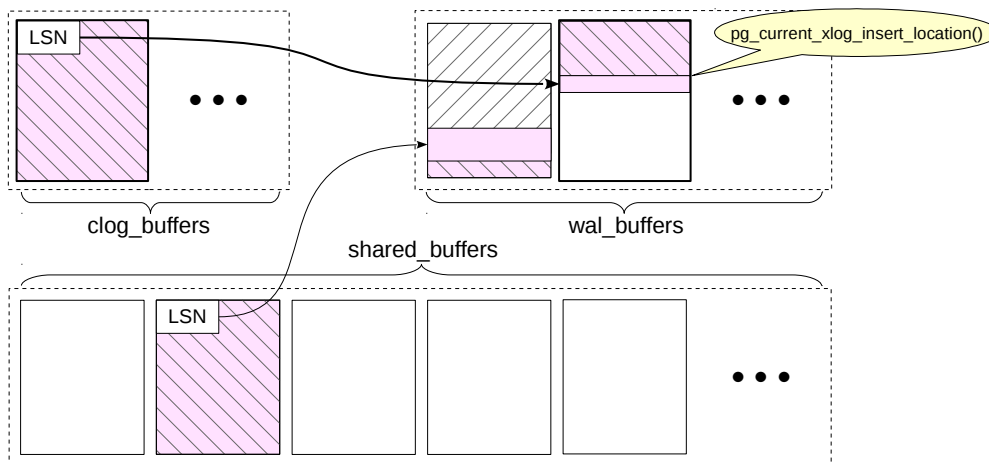


Проиллюстрируем сказанное выше. На слайде показаны три важные области общей памяти экземпляра:

- буферный кэш (размером `shared_buffers`),
- только что рассмотренный журнальный кэш, называемый также XLOG (размером `wal_buffers`)
- кэш состояния транзакций, называемый также CLOG (размером `clog_buffers`).

При изменении данных на странице в буферном кэше, создается журнальная запись. Она помещается на страницу журнала, а ссылка на запись помещается в специальное поле LSN в заголовке страницы данных.

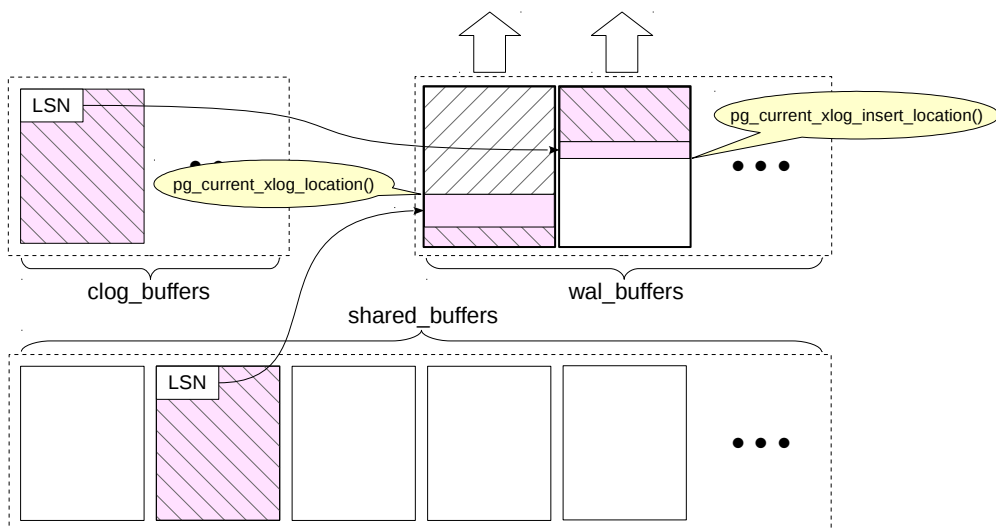
Позицию для записи можно посмотреть с помощью функции `pg_current_xlog_insert_location()`.



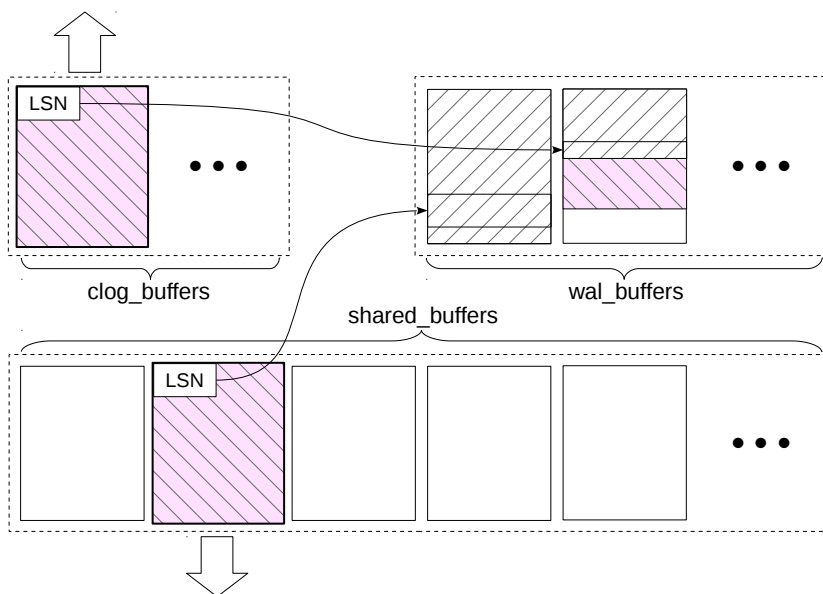
Далее происходит фиксация транзакции. Для этого создается журнальная запись, меняется бит состояния на странице CLOG и ссылка на запись проставляется в поле LSN.

При вставке указатель `pg_current_xlog_insert_location` сдвигается вперед.

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



Далее в какой-то момент (в какой именно — будет рассмотрено ниже) должна произойти запись журнала. На диск попадают данные (все имеющиеся или только часть из них) с того места, где запись остановилась в прошлый раз. Функция `pg_current_xlog_location()` показывает последнюю запись, уже дошедшую до диска.



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 состояние `pg_current_xlog_location`.

При этом работа продолжается, в журнал будут попадать новые и новые записи. Главное, чтобы запись с LSN последнего изменения страницы была на диске.

## Кэширование

данные должны дойти до энергонезависимого хранилища через кэши операционной системы, контроллера, диска  
настраивается способ синхронизации ОС;  
для выбора лучшего способа есть утилита `pg_test_fsync`, но в любом случае — дорогая операция  
журнал следует размещать на отдельной файловой системе  
если контроллер использует кэш, должна быть батарейка  
кэш диска на запись как правило следует отключать

## Настройки

`fsync = on`  
`wal_sync_method`

при выключенном `fsync` не гарантируется целостность данных!

Есть несколько моментов, которые влияют на надежность.

Во-первых, это всевозможные кэши, находящиеся на пути данных к энергонезависимому хранилищу (такому, как пластина жесткого диска). Существует кэш операционной системы, аппаратные кэши имеются у контроллеров и самих дисков. Поскольку Постгрес работает с дисками через вызовы операционной системы, он полагается на то, что вызов `fsync` (или другого метода, настраивается параметром `wal_sync_method`) гарантирует попадание данных на диск. Чтобы это действительно гарантировать, контроллер должен иметь батарейку (чтобы в случае отключения питания успеть записать данные из кэша); кэш записи на самом диске как правило следует отключать.

Поскольку синхронизация — дорогая операция, имеет смысл записывать журнал на отдельный физический носитель. Это можно сделать, примонтировав файловую систему на место каталога `$PGDATA/pg_xlog`.

Синхронизацию можно и отменить (параметр `fsync`), но в этом случае целостность данных не будет гарантирована. Единственный разумный вариант отключения этого параметра — временное увеличение производительности в случае, если данные можно восстановить (например, при начальной миграции).

<http://www.postgresql.org/docs/9.5/static/wal-reliability.html>

## Повреждение данных

записи журнала защищены CRC  
можно включить контрольные суммы страниц  
(только при инициализации кластера: `initdb -k`)

накладные  
расходы

## Атомарность записи страниц

запись образа страницы в журнал  
при первом изменении после контрольной точки  
некоторые файловые системы могут гарантировать атомарность

увеличивается  
размер журнала

## Настройки

`full_page_writes` = on  
`wal_compression` = off (сжатие образа страницы, 9.5)  
`wal_log_hints` = off (on подразумевается при контрольных суммах)

11

Во-вторых, данные могут быть повреждены из-за аппаратного сбоя носителя.

Чтобы вовремя обнаружить проблему, журнальные записи всегда снабжаются контрольными суммами.

Страницы данных также можно защитить контрольными суммами при инициализации кластера. Это приведет, однако, к дополнительным накладным расходам на вычисление сумм и их контроль. Кроме того, при включенных контрольных суммах в журнал всегда попадает даже такая «несущественная» информация, как биты подсказок (были рассмотрены в теме «Страницы и версии строк»), которую иначе можно отключить параметром `wal_log_hints`.

В-третьих, есть проблема атомарности записи. Поскольку страница занимает 8 КБ или больше, а на низком уровне запись, как правило, идет блоками по 512 байт, при сбое питания может записаться только часть страницы.

Для защиты Постгрес позволяет записывать образ страницы в журнал при первом ее изменении после контрольной точки, что, конечно, увеличивает размер журнала. Если это создает проблемы с вводом-выводом, размер можно уменьшить за счет сжатия (параметр `wal_compression`, появился в 9.5), что в свою очередь увеличит нагрузку на процессор.

Если используемая операционная система позволяет гарантировать атомарность другими средствами, параметр `full_page_writes` можно отключить.

## Алгоритм

при фиксации изменений, если включен *synchronous\_commit*, а также перед записью страницы, если запись LSN еще не на диске: ждет *commit\_delay*, если активно не менее *commit\_siblings* транзакций записывает журнал до необходимого LSN

## Свойства

гарантируется долговечность

уменьшается  
производительность

## Настройки

```
synchronous_commit = on  
commit_delay       = 0  
commit_siblings    = 5
```

Запись журнала происходит в одном из двух режимов: синхронном (когда продолжение работы невозможно без записи журнала до определенной отметки) и асинхронном (когда журнал записывается частями в фоне).

Синхронный режим работает в двух случаях:

- при фиксации изменений, если включен параметр *synchronous\_commit*;
- при записи страницы данных, если соответствующая журнальная запись еще не на диске.

Поскольку синхронизация выполняется долго, выгодно синхронизировать сразу как можно больше данных. Для этого процесс, записывающий журнал, делает паузу, определяемую параметром *commit\_delay*, но только в том случае, если имеется не менее *commit\_siblings* активных транзакций — расчет на то, что за время ожидания часть транзакций успеют создать новые журнальные записи.

Затем процесс выполняет запись журнала до необходимого LSN (или несколько больше, если за время ожидания добавились новые записи).

При синхронной записи гарантируется долговечность — если транзакция зафиксирована, то *commit* не вернет управление до тех пор, пока журнальная запись не окажется на диске. Обратная сторона состоит в том, что синхронная запись уменьшает производительность.

## Алгоритм

циклы записи через `wal_writer_delay`  
записывает только целиком заполненные страницы,  
но если новых полных страниц нет, записывает последнюю до конца

## Свойства

зафиксированные изменения могут пропасть ( $3 \times wal\_writer\_delay$ )  
целостность данных тем не менее гарантируется

## Настройки

`synchronous_commit` — можно устанавливать в транзакции  
`wal_writer_delay = 200ms`

При асинхронной записи процесс `wal writer` работает частями с интервалом в `wal_writer_delay`. В каждом цикле записи он выполняет следующее:

- если с прошлого раза была целиком заполнена одна или несколько журнальных страниц, записываются только полностью заполненные страницы (или часть таких страниц; вспомним, что журнальный кэш представляет собой кольцевой буфер — записывается только непрерывная последовательность страниц). При большом потоке изменений это позволяет работать эффективнее, не синхронизируя одну и ту же страницу несколько раз.

- если же заполненные страницы не появились, записываются изменение на текущей странице журнала.

Асинхронная запись эффективнее синхронной — фиксация изменений не ждет записи. Однако надежность уменьшается: зафиксированные данные могут пропасть в случае сбоя, если между фиксацией и сбоем прошло менее  $3 \times wal\_writer\_delay$  времени.

Не стоит забывать, что параметр `synchronous_commit` можно устанавливать в рамках транзакции. Если в приложении есть транзакции, надежностью которых можно пожертвовать, это можно использовать для увеличения производительности.

<http://www.postgresql.org/docs/9.5/static/wal-async-commit.html>

Журнал удобно использовать для разных задач, если дополнить информацию

<i>уровень</i>	<i>задача</i>	<i>дополнительная информация</i>
minimal	восстановление после сбоя	—
archive	непрерывное архивирование	операции массовой обработки данных
hot_standby	горячий резерв	статус транзакций
logical	логическая репликация	реконструкция операций

## Настройка

wal\_level = minimal

чем выше уровень, тем больше размер журнала

Кроме основной задачи — восстановления согласованности после сбоя — журнал удобно использовать и для других целей. Однако для экономии места на уровне `minimal` (по умолчанию) журнал содержит только минимально необходимый объем данных для восстановления.

Поэтому журнал нужно дополнять:

- `archive`

Для организации непрерывного архивирования в журнал должны попадать операции массовой обработки данных (такие, как `create table as select`; `create index`; `cluster`; `copy` для таблиц, созданных или очищенных в текущей транзакции). В минимальном режиме этого не происходит, а долговечность достигается за счет немедленной записи данных на диск.

- `hot_standby`

Для организации горячего резерва в журнал должна попадать информация о статусе транзакций.

В 9.6 уровень `archive` будет совмещен с уровнем `hot_standby`, поскольку разница в объеме между ними не существенна.

- `logical`

Для логической репликации в журнал должна попадать информация, на основе которой можно определить выполняемую операцию (в частности, при изменении строки надо иметь значения полей, идентифицирующих эту строку).



Журнал позволяет восстановить согласованность данных после сбоев, а также используется для других задач

Запись централизована в процессе wal writer

Настройка требует поиска баланса между надежностью, размером и производительностью

1. Создайте базу DB8 и в ней таблицу с несколькими строками.
2. Установите параметры:
  - а) `full_page_writes = on`, `wal_compression = off`.
3. Выполните контрольную точку.
4. Обновите строку таблицы.
5. Проверьте, какие изменения были записаны в журнал при выполнении п. 4 и вычислите их размер в байтах.
6. Повторите пп. 3–5 с параметрами:
  - б) `full_page_writes = on`, `wal_compression = on`,
  - в) `full_page_writes = off`.
7. Сравните результаты.

Чтобы определить размер журнальной записи, посмотрите значение функции `pg_current_xlog_location()` до и после действий, и вычислите разницу между этими значениями.

Сами журнальные записи можно посмотреть с помощью утилиты `pg_xlogdump`.