



# Переключение на реплику



Переключение на реплику

Проблемы непрерывного архивирования и потоковый архив

Возвращение в строй старого мастера

## Плановое переключение

останов основного сервера для технических работ без прерывания обслуживания

ручной режим

## Аварийное переключение

переход на реплику из-за сбоя основного сервера

ручной режим,

но в принципе можно автоматизировать с помощью дополнительного кластерного ПО

## Триггерный файл

восстановление завершается при появлении триггерного файла

*recovery.conf*

`trigger_file` — задает имя файла

## «Продвижение»

восстановление завершается при получении команды

`pg_ctl promote`

## Удаление *recovery.conf*

удаление файла *recovery.conf* и перезапуск сервера

## Применение записей WAL

если остались полученные, но еще не примененные записи

## Переход на новую ветвь времени

выполняется все время при окончании восстановления

## Архивация частичного сегмента из старой ветви (и дальнейшая архивация новых сегментов)

если настроено непрерывное архивирование

## Реплика становится самостоятельным сервером

запускаются недостающие процессы (wal writer, autovacuum...),  
принимаются запросы на запись

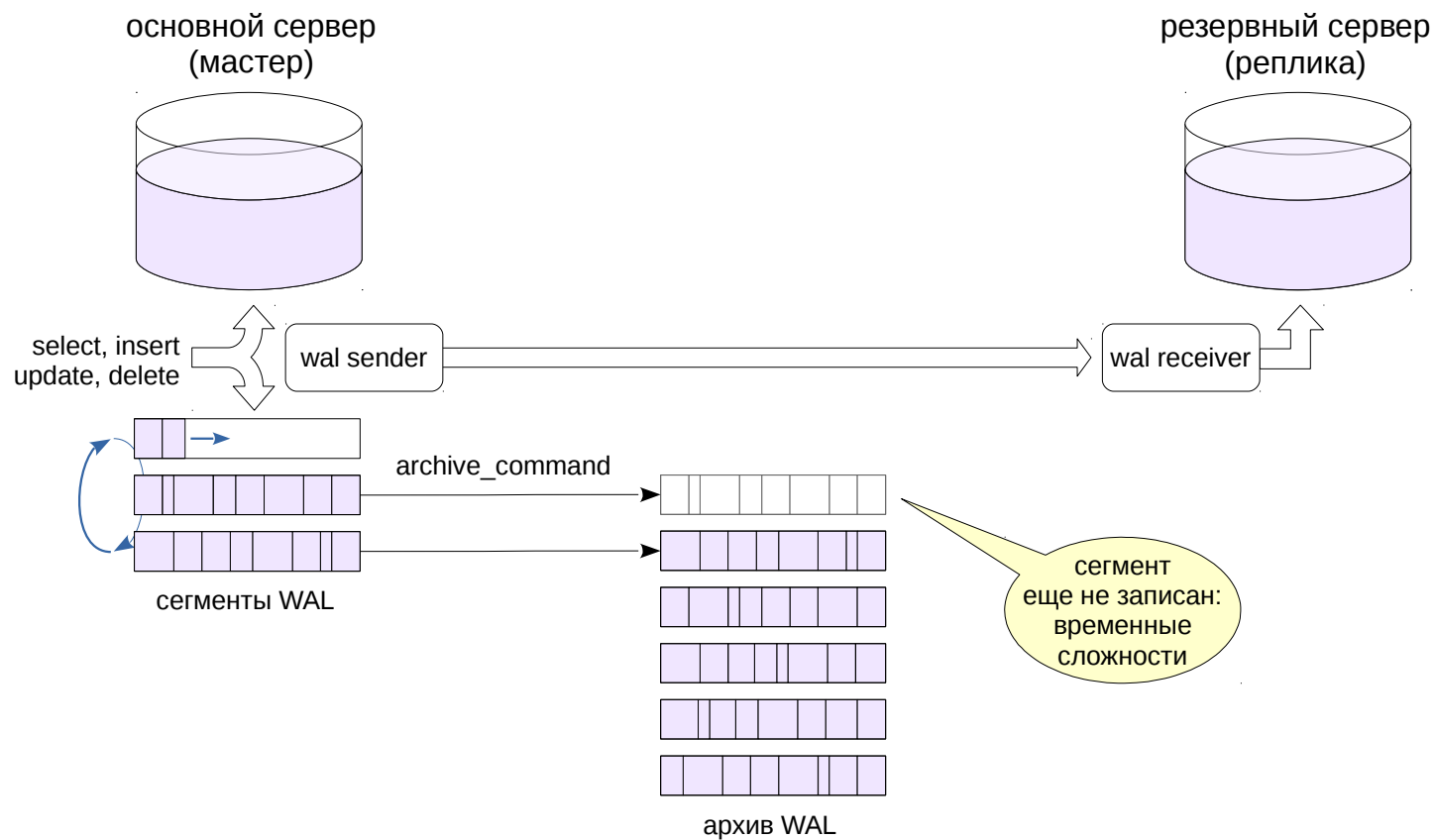
## Проблемы непрерывного архивирования (в контексте потоковой репликации)

нет гарантии, что при отказе мастера все сегменты (кроме текущего) уже записаны в архив; бывшая реплика не станет повторять попытки — реплика ничего не знает про архивирование на мастере

нет гарантии, что при отказе мастера реплика успела получить все записи WAL, уже записанные в архив

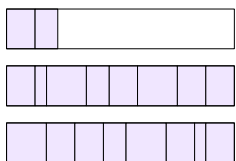
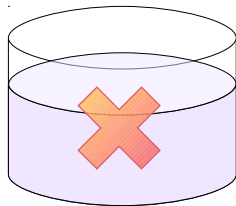
— данные поступают в два источника по-разному:  
на реплику постоянно, но с потенциальной задержкой;  
в архив периодически, но как правило быстро

# Проблемы архива (пример)



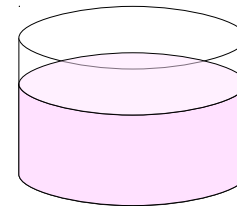
# Проблемы архива (пример)

бывший мастер



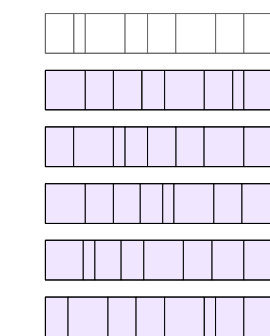
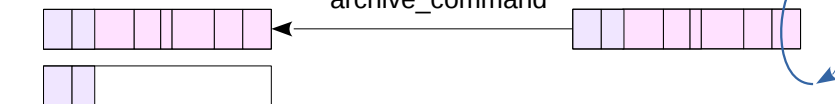
сегменты WAL

основной сервер  
(бывшая реплика)



select, insert  
update, delete

archive\_command



сегмент  
потерян

архив WAL

## Вариант 1: другая настройка архивирования (9.5)

в архив пишет не только мастер, но и реплика  
(на реплике работает процесс archiver)

команда *archive\_command* должна корректно работать при  
одновременной записи одного и того же файла двумя серверами  
частичные сегменты отличаются от полных префиксом *.partial*

*postgresql.conf*

`archive_mode = always`

## Вариант 2: потоковый архив

утилита `pg_receivexlog`

`pg_receivexlog -D каталог -h узел -p порт -U роль -S слот`

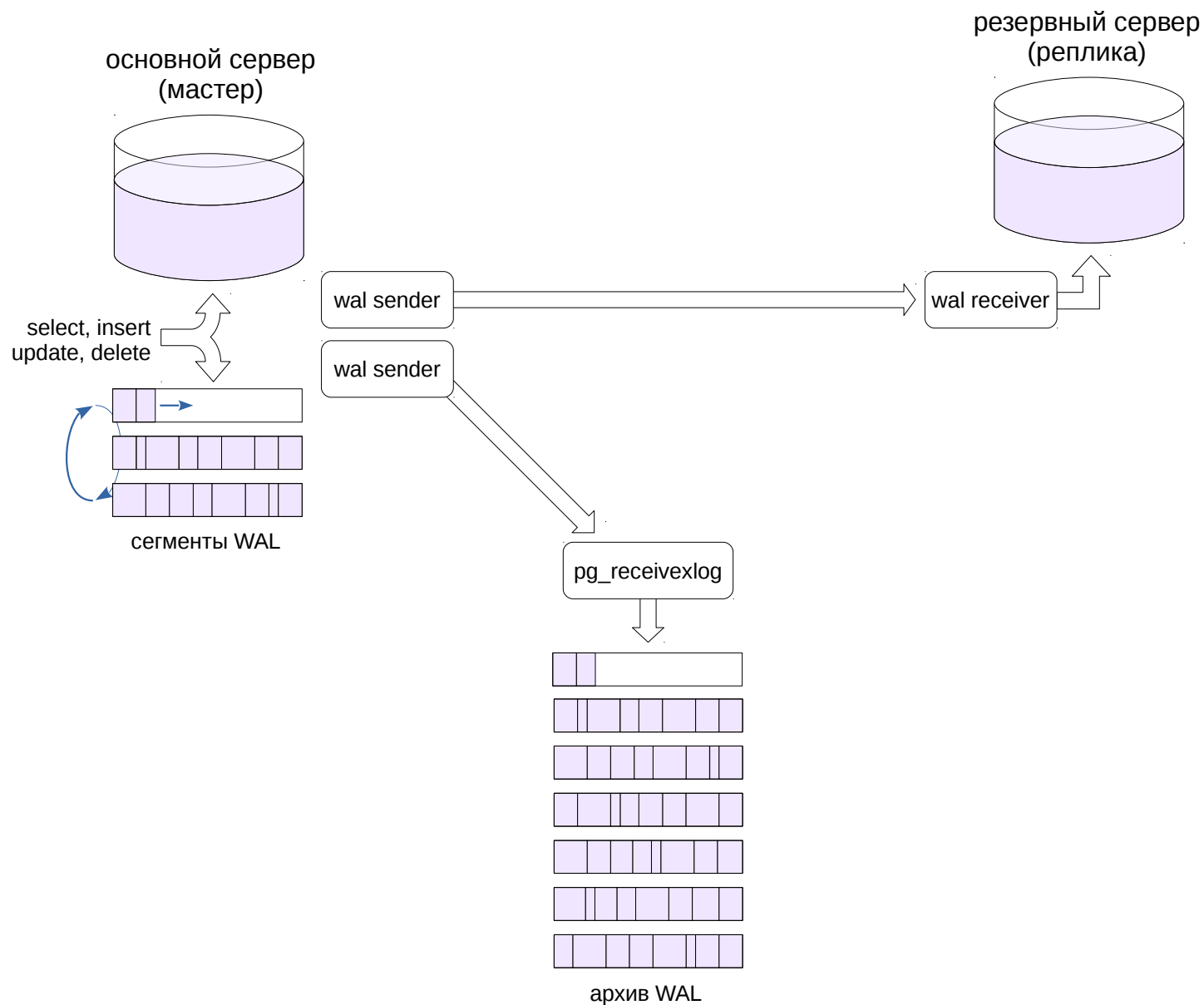
подключается по протоколу репликации (можно использовать слот) и записывает поток записей WAL в файлы-сегменты

стартовая позиция — начало сегмента, следующего за последним заполненным сегментом, найденным в каталоге,

или начало текущего сегмента сервера, если каталог пустой

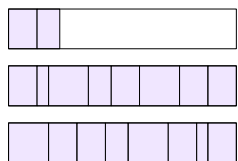
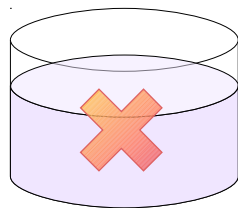
при переходе на новый мастер надо перенастроить параметры

# ПОТОКОВЫЙ архив



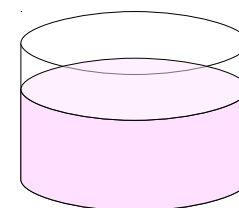
# Потоковый архив

бывший мастер



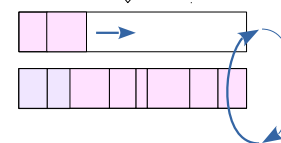
сегменты WAL

основной сервер  
(бывшая реплика)

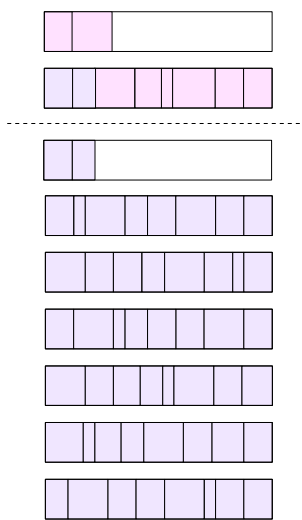


wal sender

select, insert  
update, delete



pg\_receivexlog



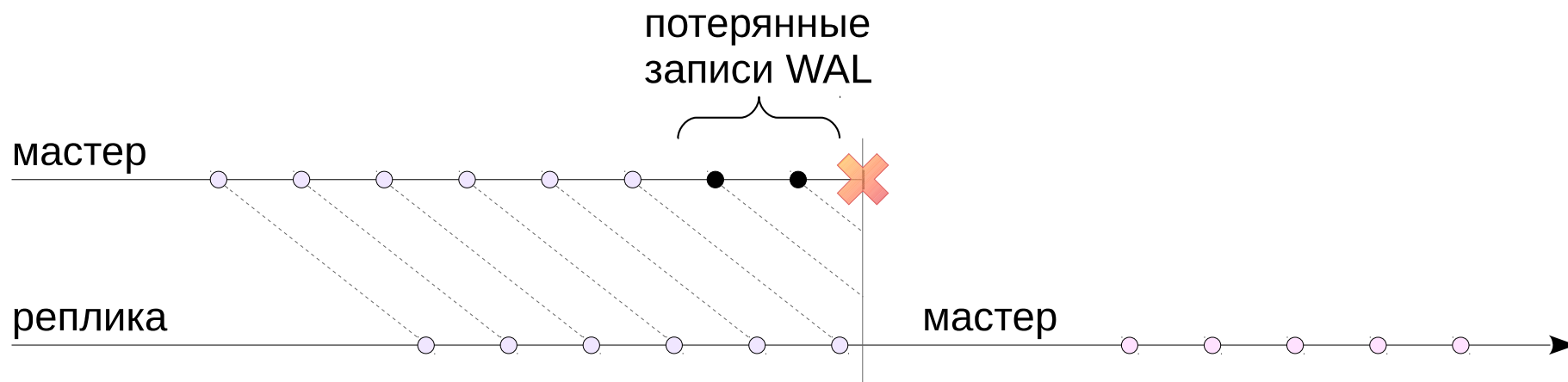
архив WAL

## Вариант 0: подключение бывшего мастера к новому

в режиме восстановления в надежде, что он получит необходимые записи WAL и станет новой репликой

— не работает!

проблема в записях, не успевших попасть на реплику из-за задержки



## Вариант 1: восстановление «с нуля» из резервной копии

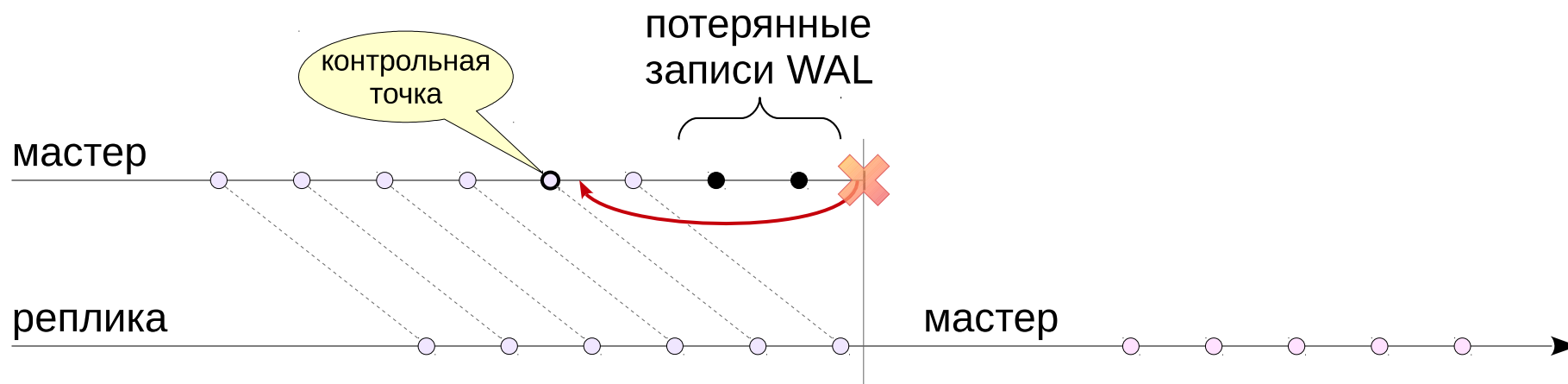
на месте бывшего мастера разворачивается абсолютно новая реплика

процесс занимает много времени

можно ускорить rsync, если с момента сбоя прошло немного времени  
(но все равно долго для больших баз данных)

## Вариант 2: утилита pg\_rewind (9.5)

«откатывает» потерянные записи WAL, заменяя соответствующие страницы на диске страницами с нового мастера копирует с нового мастера все служебные файлы дальше работает обычный режим восстановления



## Особенности и ограничения pg\_rewind

все необходимые журнальные файлы целевого сервера должны сохраниться в pg\_xlog

у целевого сервера должны быть включены контрольные суммы или параметр `wal_log_hints = on`

целевой сервер должен быть остановлен через контрольную точку (данные на диске должны быть в согласованном состоянии)

ветвь времени сервера-источника не должна меняться, а ветвь времени целевого сервера должна увеличиться

# Демонстрация



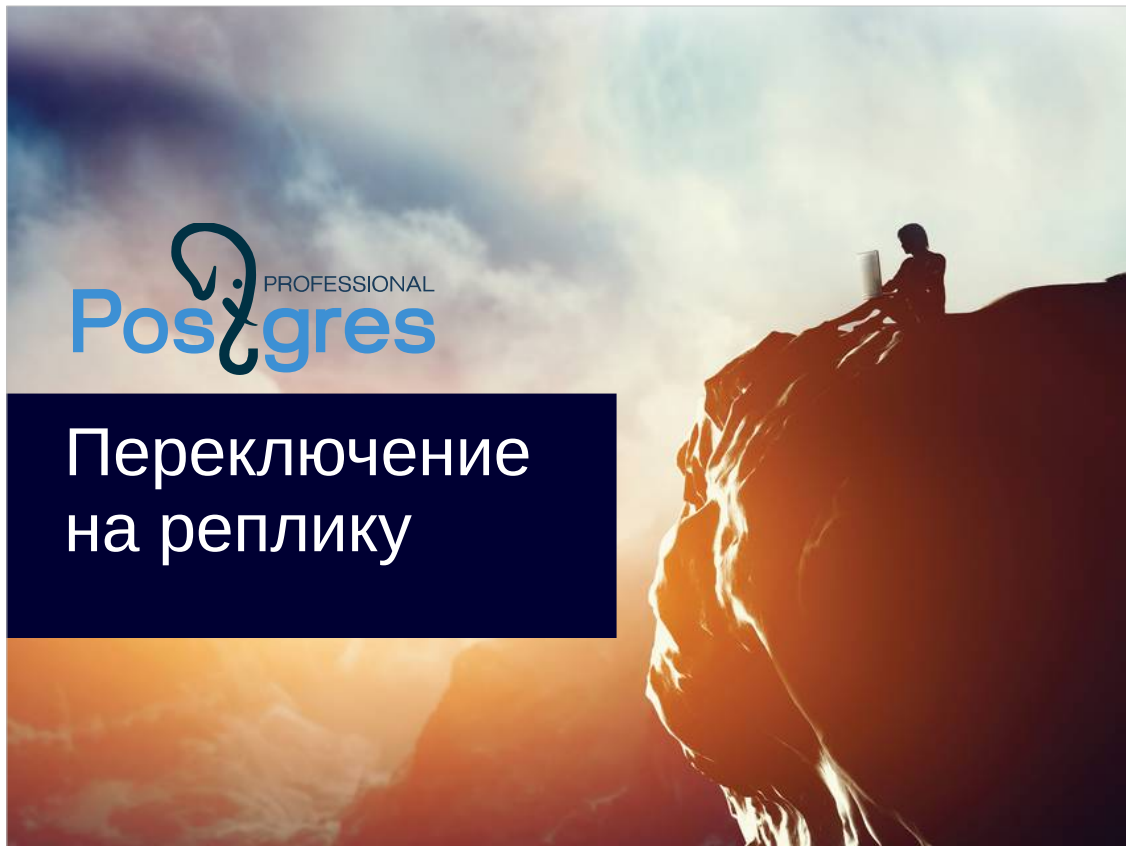
Переключение используется как в штатных, так и в нештатных ситуациях

После переключения бывший мастер надо вернуть в строй

Обе процедуры должны быть заранее отработаны

Особое внимание надо уделить архиву WAL, если он используется для восстановления из резервных копий

1. Выполните необходимую настройку мастера и реплики для потоковой репликации с использованием слота, без непрерывного архивирования.  
В файл `recovery.conf` реплики добавьте параметр для триггерного файла.
2. Имитируйте сбой основного сервера.
3. Сделайте реплику основным сервером с помощью триггерного файла.
4. Верните в строй бывший основной сервер, выполнив резервную копию с нового мастера и настроив необходимые параметры.
5. Убедитесь, что репликация работает и использует слот.



### **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.

Переключение на реплику

Проблемы непрерывного архивирования и потоковый архив

Возвращение в строй старого мастера

## Плановое переключение

останов основного сервера для технических работ без прерывания обслуживания  
ручной режим

## Аварийное переключение

переход на реплику из-за сбоя основного сервера  
ручной режим,  
но в принципе можно автоматизировать с помощью дополнительного кластерного ПО

Причины перехода на резервный сервер бывают разные. Это может быть необходимость проведения технических работ на основном сервере — тогда переход выполняется в удобное время в штатном режиме. А может быть сбой основного сервера, и в таком случае переходить на резервный сервер нужно как можно быстрее, чтобы не прерывать обслуживание пользователей.

Даже в случае сбоя переход осуществляется вручную, так как PostgreSQL не имеет встроенного кластерного программного обеспечения (которое должно следить на состоянием серверов и инициировать переход).

## Триггерный файл

восстановление завершается при появлении триггерного файла

*recovery.conf*

`trigger_file` — задает имя файла

## «Продвижение»

восстановление завершается при получении команды

`pg_ctl promote`

## Удаление *recovery.conf*

удаление файла *recovery.conf* и перезапуск сервера

В процессе репликации резервный сервер находится в режиме постоянного восстановления: он читает записи WAL из архива, локального каталога `pg_xlog` и из потокового интерфейса. Чтобы перейти на резервный сервер, надо разорвать цикл восстановления.

Для этого есть несколько способов.

Во-первых, триггерный файл, имя которого определяется в параметре `trigger_file` файла *recovery.conf*. При появлении в системе такого файла цикл прерывается.

Во-вторых, можно выполнить команду `pg_ctl promote`.

Наконец, можно просто удалить файл *recovery.conf* и перезапустить сервер. Это не вполне «честный» способ, поскольку в этом случае реплика не поймет, что восстановление завершено, и не перейдет на новую ветвь времени. Дальше мы будем рассматривать только два первых варианта.

## Применение записей WAL

если остались полученные, но еще не примененные записи

## Переход на новую ветвь времени

выполняется все время при окончании восстановления

## Архивация частичного сегмента из старой ветви (и дальнейшая архивация новых сегментов)

если настроено непрерывное архивирование

## Реплика становится самостоятельным сервером

запускаются недостающие процессы (wal writer, autovacuum...),  
принимаются запросы на запись

Получив тот или другой сигнал, сервер завершает восстановление и переходит в обычный режим работы.

Предварительно он применяет уже полученные записи WAL, которые еще не были применены.

Затем сервер переходит на новую ветку времени, как всегда происходит при окончании восстановления.

Если настроено архивирование, то сервер сбрасывает не заполненный до конца (частичный) сегмент WAL, относящийся к старой ветви времени, и записывает поступающие записи в новый сегмент, относящийся к новой ветви времени (начало которого повторяет старый частичный сегмент). При этом на сервере запускаются все недостающие процессы, которые не требовалось резервному серверу (такие, как wal writer и autovacuum launcher), и останавливается процесс startup process.

## Проблемы непрерывного архивирования (в контексте потоковой репликации)

нет гарантии, что при отказе мастера все сегменты (кроме текущего) уже записаны в архив; бывшая реплика не станет повторять попытки  
— реплика ничего не знает про архивирование на мастере

нет гарантии, что при отказе мастера реплика успела получить все записи WAL, уже записанные в архив

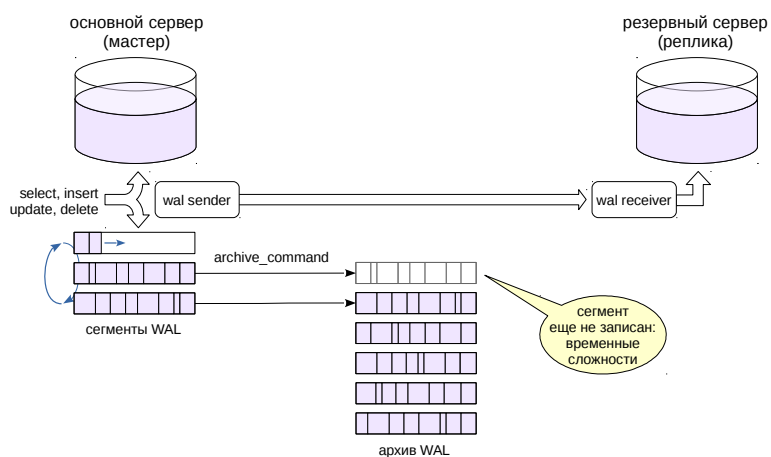
— данные поступают в два источника по-разному:  
на реплику постоянно, но с потенциальной задержкой;  
в архив периодически, но как правило быстро

Архив журналов упреждающей записи, наполняемый с помощью механизма непрерывного архивирования, имеет ряд проблем, обнаруживаемых в контексте потоковой репликации. Если происходит отказ мастера и реплика занимает его место, возможны ситуации, приводящие к потере архива:

- При отказе мастера не все сегменты были записаны в архив (например, могли возникнуть временные проблемы и `archive_command` возвращала ошибку). Когда бывшая реплика займет место мастера, она не запишет недостающие сегменты в архив, хотя они у нее есть. В результате получаем неполный архив.

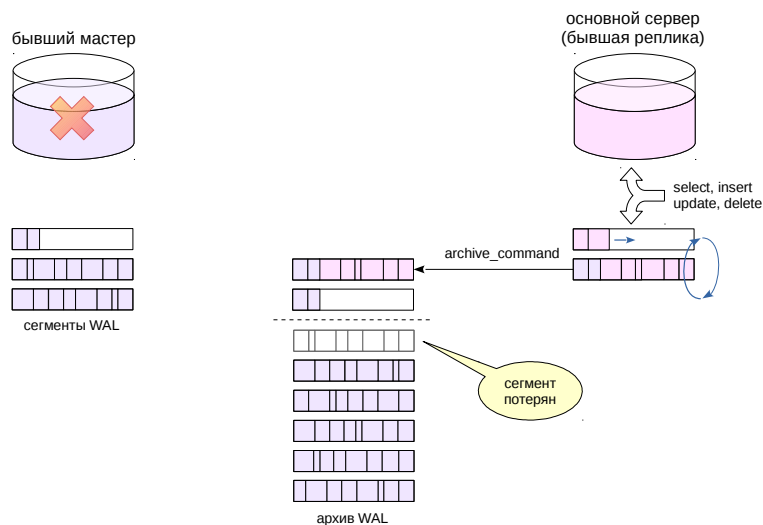
- Реплика может отставать от мастера. Если мастер непосредственно перед отказом успел записать в архив заполненный сегмент, то на реплике этот сегмент может остаться частичным. Когда реплика займет место мастера, она попытается скопировать частичный сегмент в архив, но `archive_command` не перезапишет уже существующий файл. В результате получим некорректные данные в архиве.

# Проблемы архива (пример)



На рисунке приведен пример потери, возникающий, когда на момент перехода на реплику не все сегменты были сброшены в архив.

# Проблемы архива (пример)



«Дыра» в архивных файлах приводит к тому, что с помощью архива невозможно будет восстановиться из резервной копии, сделанной до появления «дыры».

## Вариант 1: другая настройка архивирования (9.5)

в архив пишет не только мастер, но и реплика  
(на реплике работает процесс archiver)

команда *archive\_command* должна корректно работать при  
одновременной записи одного и того же файла двумя серверами  
частичные сегменты отличаются от полных префиксом *.partial*

*postgresql.conf*

```
archive_mode = always
```

Необходимые исправления в непрерывном архивировании сделаны в версии 9.5.

Во-первых, введено новое значение параметра `archive_mode: always`. При этом значении на реплике запускается процесс `archiver`, который записывает сегменты в архив наравне с мастером. Таким образом, один и тот же файл будет записан два раза: и мастером, и репликой. Это накладывает на команду `archive_command` серьезные требования:

- она *должна* перезаписывать существующий файл, но только если содержимое отличается;

- она *должна* поддерживать возможность параллельного вызова.

Во-вторых, частичные сегменты имеют префикс `.partial`, что позволяет отличать их от заполненных сегментов и не учитывать при восстановлении. Частичный сегмент записывается репликой при повышении до мастера и переходе на новую ветвь времени.

## Вариант 2: потоковый архив

утилита `pg_receivexlog`

`pg_receivexlog -D каталог -h узел -p порт -U роль -S слот`

подключается по протоколу репликации (можно использовать слот) и записывает поток записей WAL в файлы-сегменты

стартовая позиция — начало сегмента, следующего за последним заполненным сегментом, найденным в каталоге, или начало текущего сегмента сервера, если каталог пустой  
при переходе на новый мастер надо перенастроить параметры

Другое решение — использование утилиты `pg_receivexlog` для записи сегментов в архив по протоколу потоковой репликации.

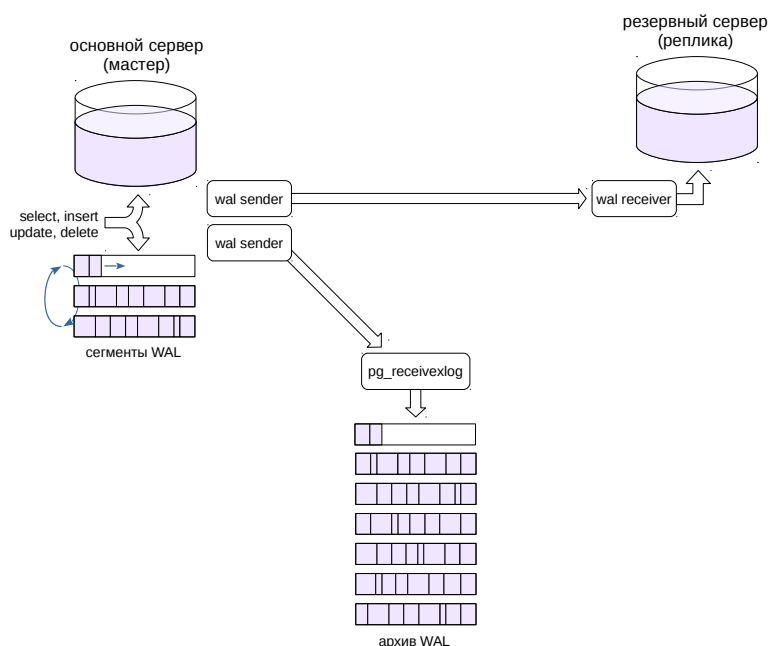
Обычно утилита запускается на отдельном «архивном» сервере и подключается к мастеру с параметрами, указанными в ключах. Утилита может (и должна) использовать слот репликации, чтобы гарантированно не потерять записи.

Утилита формирует файлы аналогично тому, как это делает сервер, и записывает их в указанный каталог. Еще не заполненные сегменты записываются с префиксом `.partial`.

Архивирование всегда начинается с начала сегмента, следующего за последним уже полностью заполненным сегментом, который присутствует в архиве. Если архив пуст (первый запуск), архивирование начинается с начала текущего сегмента.

При переходе на новый основной сервер утилиту требуется остановить и запустить заново с соответствующими параметрами.

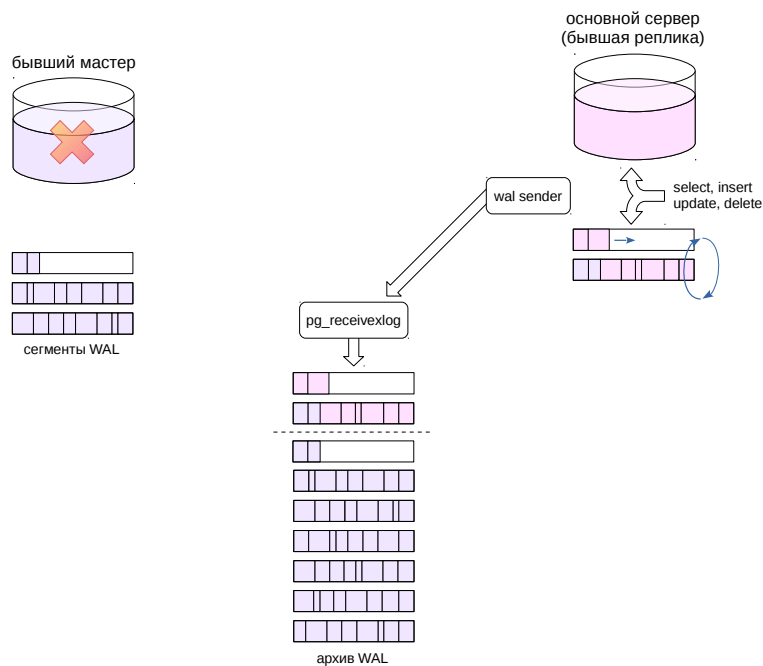
Требуется учесть, что сама по себе утилита не запускается автоматически (как сервис) и не демонизируется.



Утилита `pg_receivexlog` подключается к мастеру по протоколу потоковой репликации аналогично тому, как это делает процесс `wal receiver`. Подключение обрабатывается отдельным процессом `wal sender` (это необходимо учесть при установке параметра `max_wal_senders`).

Утилита записывает данные, не дожидаясь получения всего сегмента.

# Потоковый архив



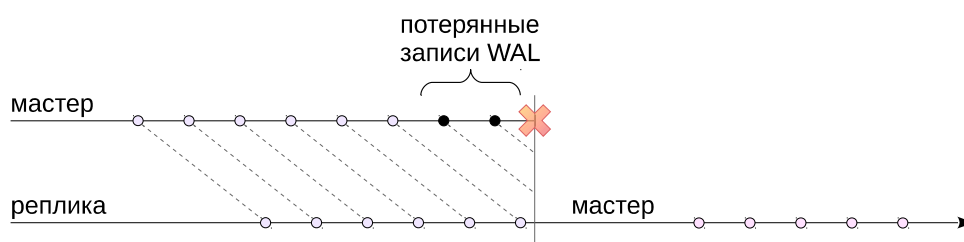
При переключении на новый основной сервер в архиве остается частичный файл `.partial` (он не будет учитываться при восстановлении), соответствующий старой ветви времени. Утилита начинает записывать файлы, соответствующие новой ветви времени.

## Вариант 0: подключение бывшего мастера к новому

в режиме восстановления в надежде, что он получит необходимые записи WAL и станет новой репликой

— не работает!

проблема в записях, не успевших попасть на реплику из-за задержки



13

Если переход на реплику произошел из-за аппаратуры (необходима замена дисков или сервера) или операционной системы (необходима переустановка ОС), то единственный вариант состоит в изготовлении абсолютно новой реплики на новом сервере.

Если же переход был штатным, нужен способ быстро вернуть старый мастер в строй (теперь уже в качестве реплики).

К сожалению, простой способ, который первым приходит в голову — просто подключить старый мастер к новому по репликационному протоколу — не работает. Причина в том, что из-за задержек в репликации часть записей WAL, попавших в журнал упреждающей записи (а может быть и на дисковые страницы), могла не дойти до реплики. Если на старом мастере остались такие записи, то применение записей с нового мастера приведет к повреждению базы данных.

## Вариант 1: восстановление «с нуля» из резервной копии

на месте бывшего мастера разворачивается абсолютно новая реплика  
процесс занимает много времени

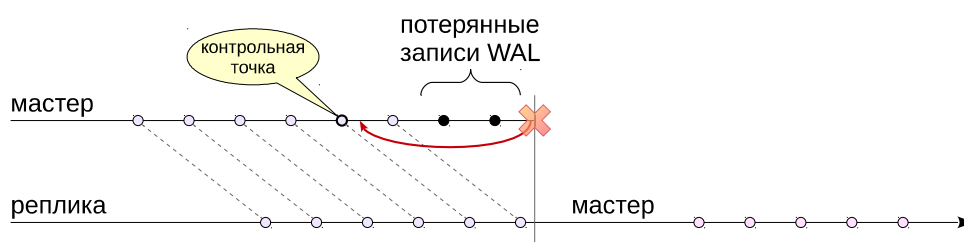
можно ускорить rsync, если с момента сбоя прошло немного времени  
(но все равно долго для больших баз данных)

Всегда есть вариант создать абсолютно новую реплику путем изготовления базовой резервной копии. Однако для больших баз данных этот процесс может занимать много времени.

Вариант такого подхода — не использовать утилиту `pg_basebackup`, а сделать копию вручную с использованием `rsync`. Если выполнять копирование сразу после перехода на реплику, процесс должен занять существенно меньше времени.

### Вариант 2: утилита `pg_rewind` (9.5)

«откатывает» потерянные записи WAL, заменяя соответствующие страницы на диске страницами с нового мастера копирует с нового мастера все служебные файлы дальше работает обычный режим восстановления



15

Еще более быстрый вариант состоит в использовании утилиты `pg_rewind`. Она доступна, начиная с версии 9.5, а для версий 9.3 и 9.4 существует в виде расширения.

Утилита определяет записи WAL, которые не дошли до реплики (вплоть до ближайшей контрольной точки), и находит страницы, затронутые этими записями.

Найденные страницы (которых должно быть немного) заменяются страницами с нового мастера. Кроме того, утилита копирует с сервера-источника (нового мастера) все служебные файлы.

Дальше применяются все необходимые записи WAL с нового мастера. Фактически, это выполняет уже не утилита, а обычный процесс восстановления после запуска сервера. Чтобы восстановление началось с нужного момента, утилита создает управляющий файл `backup_label`.

### Особенности и ограничения pg\_rewind

все необходимые журнальные файлы целевого сервера должны сохраниться в pg\_xlog

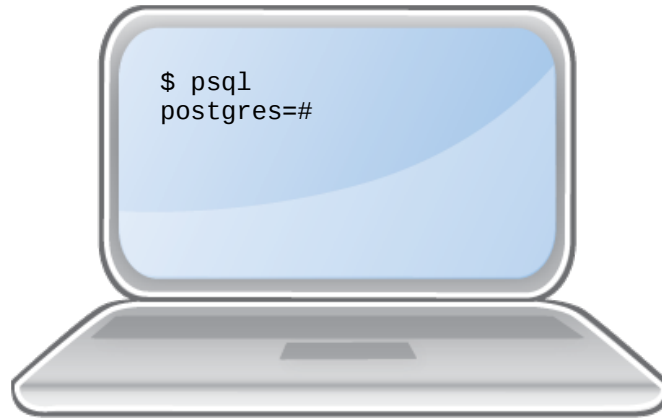
у целевого сервера должны быть включены контрольные суммы или параметр wal\_log\_hints = on

целевой сервер должен быть остановлен через контрольную точку (данные на диске должны быть в согласованном состоянии)

ветвь времени сервера-источника не должна меняться, а ветвь времени целевого сервера должна увеличиться

Необходимые условия для работы утилиты:

- Все сегменты WAL от текущего момента до контрольной точки должны находиться в каталоге pg\_xlog. Это не проблема, если целевой сервер (старый мастер) был остановлен перед переключением на реплику, или хотя бы вскоре после этого.
- Первое изменение данных после контрольной точки должно вызывать запись в WAL полной страницы. Параметра full\_page\_writes = on недостаточно, поскольку он не учитывает «незначительные» изменения страниц (hint bits). Требуется, чтобы либо кластер был инициализирован с контрольными суммами страниц, либо нужно устанавливать параметр wal\_log\_hints = on.
- Сервер должен быть остановлен аккуратно, с выполнением контрольной точки, так, чтобы на диске были согласованные данные. Поэтому такой способ не подходит для аварийного переключения.
- Сервер-источник (старая реплика) должен был стать мастером путем повышения с увеличением номера ветви времени. При этом целевой сервер (старый мастер) не должен изменять ветвь времени. Это ограничение будет снято в версии 9.6.



Переключение используется как в штатных,  
так и в нештатных ситуациях

После переключения бывший мастер надо вернуть в строй

Обе процедуры должны быть заранее отработаны

Особое внимание надо уделить архиву WAL, если он  
используется для восстановления из резервных копий

1. Выполните необходимую настройку мастера и реплики для потоковой репликации с использованием слота, без непрерывного архивирования.  
В файл `recovery.conf` реплики добавьте параметр для триггерного файла.
2. Имитируйте сбой основного сервера.
3. Сделайте реплику основным сервером с помощью триггерного файла.
4. Верните в строй бывший основной сервер, выполнив резервную копию с нового мастера и настроив необходимые параметры.
5. Убедитесь, что репликация работает и использует слот.