Авторы: Егор Рогов, Павел Лузанов

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

Темы

Файловый архив — непрерывная архивация

Потоковый архив — утилита pg_receivewal

Восстановление с использованием архива

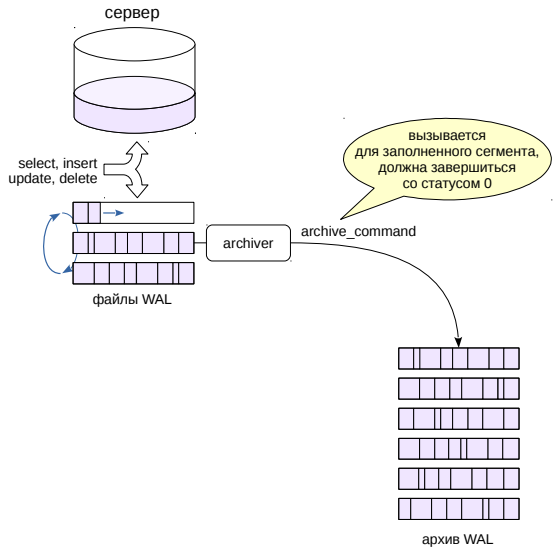
Непрерывная архивация

Файлы журнала архивируются по мере заполнения

+ вся настройка внутри СУБД

− журнальные записи попадают в архив с задержкой

Непрерывная архивация



archive_mode = on
archive_command
archive_timeout

Раз у нас есть базовая резервная копия и журнал предзаписи, то, добавляя каким-то образом к копии все новые журнальные файлы, генерируемые сервером, мы можем восстановить систему не только на момент копирования файловой системы, но и вообще на произвольный момент времени.

И такая возможность есть. Но журнальные записи добавляются не к самой резервной копии, а в отдельный «архив».

Отправление журнальных файлов в архив реализуется фоновым процессом archiver, который включается параметром archive_mode = on.

Для копирования определяется произвольная команда shell в параметре archive_command. Она вызывается при заполнении очередного сегмента WAL. Если команда завершается с нулевым статусом, то считается, что сегмент успешно помещен в архив и может быть удален с сервера. При ненулевом статусе этот сегмент (и следующие за ним) не будут удаляться, а сервер будет периодически повторять команду архивирования, пока не получит 0.

Параметр archive_timeout позволяет указать максимальное время переключения на новый сегмент WAL — это позволяет при невысокой активности сервера сохранять тем не менее журналы не реже, чем хотелось бы (иными словами, потерять данные максимум за указанное время). Переключение на новый сегмент можно выполнить и вручную с помощью функции pg_switch_wal().

https://postgrespro.ru/docs/postgresql/10/continuous-archiving

archive_command

Команда ОС, архивирующая заполненный сегмент WAL

- копирует файл `%p` в архив под именем `%f`,
  сам архив может быть организован любым образом
- должна завершаться со статусом 0 только при успехе
  (пустая команда не считается успешной и приостанавливает архивацию)
- не должна перезаписывать уже существующие файлы
- должна гарантировать запись в энергонезависимую память
- удобно реализовать нужную логику в собственном скрипте
- удобно включить коллектор сообщений (logging_collector = on)
  и получать диагностику в журнале сообщений сервера

Команда архивирования должна скопировать указанный файл в некий архив. Архив может быть организован любым образом. Например, это может быть файловая система на отдельном сервере.

Команда обязана завершаться с нулевым статусом только в случае успеха.

Команда не должна перезаписывать уже существующие файлы, так как это скорее всего означает какую-то ошибку и перезапись файла может погубить архив.

Для гарантии надежности команда должна обеспечить попадание файла в энергонезависимую память: иначе при сбое архива можно потерять сегмент, если сервер PostgreSQL успеет его удалить.

Удобно включить сбор сообщений с помощью процесса logging_collector, так как в этом случае сообщения об ошибках при выполнения archive_command будут попадать в журнал сервера.

Если подразумевается сложная логика, то ее удобно записать в скрипт и использовать имя скрипта в качестве команды копирования.

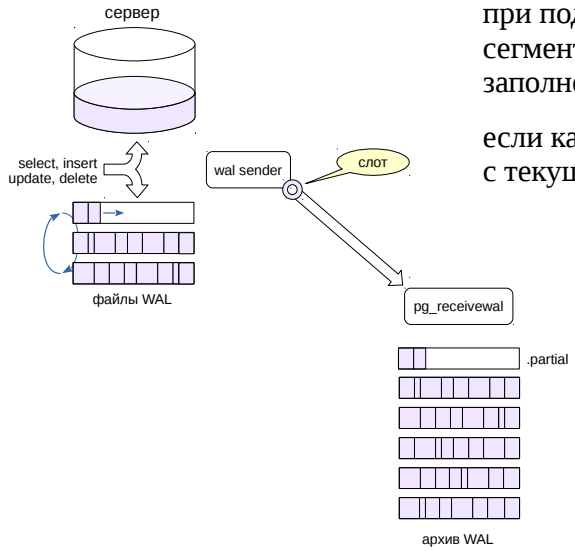
https://postgrespro.ru/docs/postgresql/10/runtime-config-wal.html#GUC-ARCHIVE-COMMAND

Потоковый архив

Архивация по протоколу репликации

+ журнальные записи попадают в архив сразу же
− требуется отдельная (хотя и штатная) утилита

pg_receivewal



при подключении читает WAL с начала сегмента, следующего за последним заполненным сегментом в каталоге

если каталог пустой, начинает с текущего сегмента сервера

Можно организовать архив иным способом, с помощью протокола репликации. Для этого используется утилита pg_receivewal.

Обычно утилита запускается на отдельном сервере и подключается к серверу 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 сам сервер, и записывает их в указанный каталог. Еще не до конца заполненные сегменты отличаются 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. Вместе со слотом это гарантирует отсутствие пропусков в архиве, даже если утилита отключалась на некоторое время.

Требуется учесть, что сама по себе утилита не запускается автоматически (как сервис) и не демонизируется. Она потребует дополнительного мониторинга и (в случае репликации) действией по переключению на другой сервер.

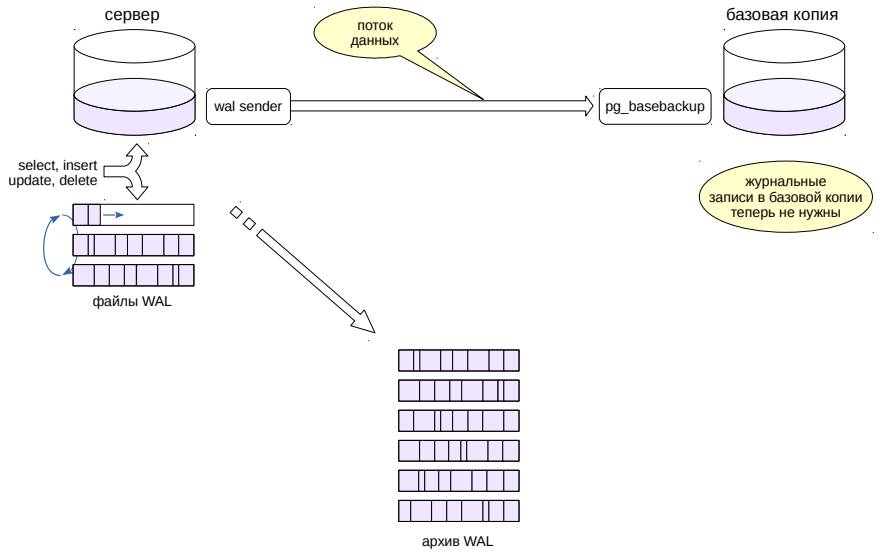
Восстановление

Резервное копирование при наличии архива

Управление восстановлением

Ветви времени

Базовая копия



Если мы располагаем настроенным архивом журнала предзаписи, то в базовой резервной копии файлы журнала становятся не обязательны — ведь их при восстановлении можно получить из архива. Поэтому pg_basebackup можно запускать с ключом --wal-method=none.

(Но и никакого вреда от журнальных файлов, кроме занимаемого объема, внутри базовой копии, кроме занимаемого объема, тоже нет. Более того, они позволят восстановить систему в ситуации, если архив окажется недоступен.)

Восстановление

tablespace_map

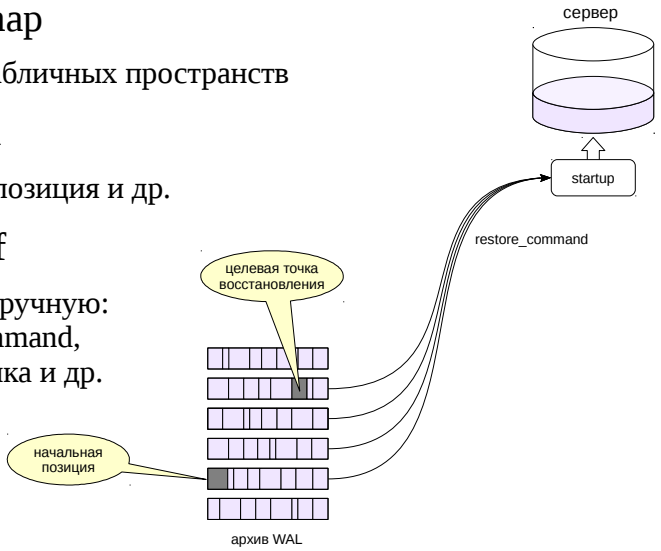
пути для табличных пространств

backup_label

начальная позиция и др.

recovery.conf

создается вручную:
restore_command,
целевая точка и др.



Процесс восстановления немного усложняется. Им управляют три файла.

Файл **tablespace_map** создается при создании базовой резервной копии и содержит пути для табличных пространств. Если символические ссылки в pg_tblspc отсутствуют (это верно для формата tar), то они будут созданы при старте сервера на основании информации в tablespace_map.

С этим файлом мы уже встречались в теме «Базовая резервная копия».

Файл метки **backup_label** также всегда создается при создании базовой резервной копии и содержит название, время создания копии и — самое важное — название сегмента WAL и позицию в нем, с которой надо начинать восстановление.

На самом деле файл метки создается, даже если архив не используется. Просто в таком случае он не играет никакой важной роли.

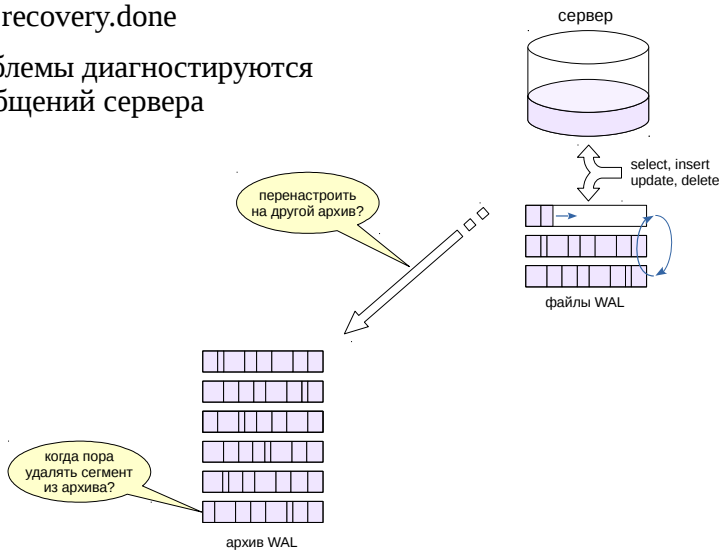
Наконец, файл **recovery.conf,** который создается вручную. Его формат соответствует другим конфигурационным файлам, например, postgresql.conf. В нем можно указать команду восстановления restore_command, целевую точку и некоторую другую информацию (подробнее рассматривается дальше).

Если recovery.conf не создан, то, с точки зрения системы, она просто выполняет восстановление после сбоя. Если же создан — то PostgreSQL понимает, что происходит управляемое пользователем восстановление из резервной копии.

Восстановление

recovery.conf → recovery.done

возможные проблемы диагностируются
по журналу сообщений сервера



После того, как восстановление завершено, процесс startup завершается, postmaster запускает остальные служебные процессы, необходимые для работы экземпляра, и сервер начинает работать в обычном режиме. Файл recovery.conf переименовывается в recovery.done.

Возможна ситуация, когда запущенный сервер не стартует. В этом случае в операционной системе будут отсутствовать процессы (и не будет файла postmaster.pid), а recovery.conf не будет переименован. Причину ошибки можно узнать из журнала сообщений сервера (например, в файле recovery.conf была допущена ошибка). После исправления причины ошибки сервер следует запустить еще раз.

Важный момент: если на сервере-источнике было настроено непрерывное архивирование, то на резервном сервере его надо либо отключить, либо перенаправить в другой архив.

Заметим, что pg_basebackup не ведет каталога сделанных резервных копий. Поэтому при удалении некоторых резервных копий нет штатного автоматического способа очищать архив от файлов, которые уже никогда не пригодятся для восстановления. Чтобы очистить архив, нужно просмотреть все имеющиеся копии и среди всех backup_label найти минимальный номер сегмента WAL. Тогда все файлы с номерами, меньшими найденного, могут быть безопасно удалены.

Настройки recovery.conf

Команда восстановления

restore_command — команда, «обратная» archive_command:
копирует файл `%f` из архива в `%p`

должна завершаться со статусом 0 только при успехе

Восстановление до определенной точки (PITR)

по умолчанию проигрываются все журнальные записи из архива

recovery_target = 'immediate'	только восстановить согласованность
recovery_target_name = '*имя*'	до именованной точки, созданной pg_create_restore_point('*имя*')
recovery_target_time = '*время*'	до указанного времени
recovery_target_xid = '*xid*'	до указанной транзакции
recovery_target_inclusive = on|off	включать ли указанную точку

В минимальном варианте, в файле recovery.conf достаточно одного параметра **restore_command,** который определяет команду, «обратную» команде копирование archive_command. Она должна скопировать файл в обратном направлении, и тоже должна завершаться со статусом 0 только при успехе. Это очень важно, так как в процессе восстановления сервер может выполнять команду для файлов, которых не окажется в архиве — это не ошибка, но команда должна завершиться с ненулевым статусом.

В таком минимальном варианте базы данных будут восстановлены максимально близко к моменту сбоя. Однако процесс восстановления можно остановить и в любой другой момент, указав **целевую точку.**

Параметр recovery_target = 'immediate' остановит восстановление, как только будет достигнута согласованность. Фактически, это эквивалентно восстановлению из базовой копии без архива.

Параметр recovery_target_name позволяет указать именованную точку восстановления, созданную ранее с помощью функции pg_create_restore_point(). Это полезно, если заранее известно, что может потребоваться восстановление.

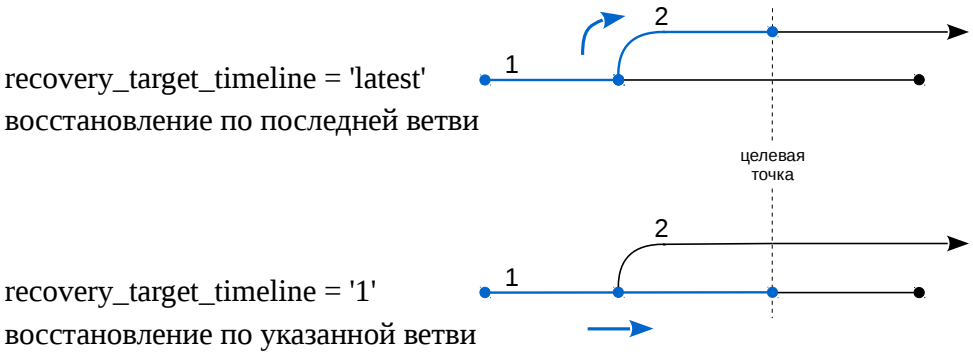
Параметр recovery_target_time позволяет указать произвольное время (timestamp), а параметр recovery_target_xid — произвольную транзакцию. При этом с помощью параметра recovery_target_inclusive можно включить или исключить саму указанную точку.

Ветви времени

Порядковый номер, +1 при каждом восстановлении

номер *N* ветви времени входит в имя сегмента WAL

история сохраняется в файле pg_wal/*N*.history и архивируется



После восстановления в прошлом сервер будет генерировать новые сегменты WAL, которые будут пересекаться с сегментами «из прошлой жизни». Чтобы не потерять сегменты и, вместе с ними, возможность восстановления на другой момент, PostgreSQL вводит понятие ветви времени. После каждого восстановления (с использованием recovery.conf) номер ветви времени увеличивается, и этот номер является частью номера сегментов WAL.

По умолчанию восстановление происходит в текущей ветви времени. Выбрать другую ветвь можно с помощью параметра recovery_target_timeline.

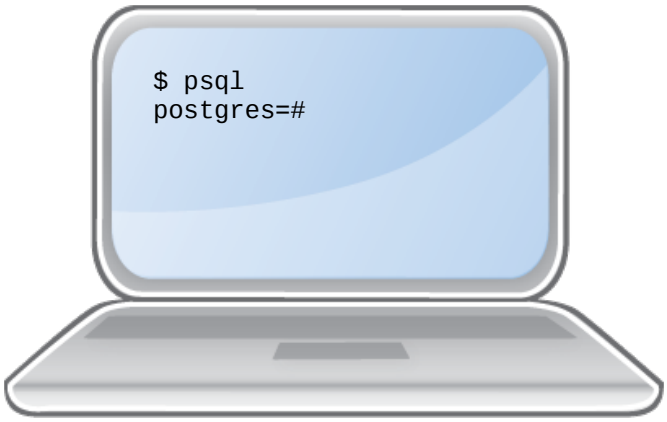
На рисунках приведен пример ветвления. Первая ветвь имеет номер 1. В некоторый момент произошло восстановление в прошлом и началась ветвь 2.

Если теперь выполнить восстановление с указанием параметра recovery_target_timeline = 'latest', то, дойдя то точки ветвления, восстановление продолжится во второй (более поздней) ветви.

Если же указан параметр recovery_target_timeline = '1', то восстановление продолжится по первой (указанной) ветви.

Информацию о точках ветвления PostgreSQL берет из файлов истории pg_wal/*N*.history. Поэтому их никогда не следует удалять из архива.

Демонстрация



Итоги

Физическое резервирование —
базовые резервные копии и архив журнала предзаписи

Хорошо подходит

- для текущего периодического резервирования
- для восстановления после сбоя с минимальной потерей данных
- для восстановления на произвольный момент времени
- для резервирования данных большого объема

Плохо подходит

- для длительного хранения

Не подходит

- для миграции на другую платформу

Практика

1. На первом сервере создайте базу данных и в ней таблицу с какими-нибудь данными.
2. Настройте непрерывное архивирование.
3. Сделайте базовую резервную копию кластера с помощью pg_basebackup, без файлов журнала.
4. Вставьте еще несколько строк в таблицу и убедитесь, что текущий сегмент WAL попал в архив.
5. Восстановите второй сервер из резервной копии, указав в recovery.conf одну только команду восстановления. Проверьте, что в таблице восстановились все строки.
6. Остановите второй сервер и восстановите его повторно из той же резервной копии, на этот раз указав целевую точку восстановления immediate. Проверьте таблицу.