

Резервное копирование Обзор



Авторские права

© Postgres Professional, 2017 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

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.

Логическое резервное копирование

Физическое резервное копирование

Что такое логическое резервное копирование

Копия таблицы

Копия базы данных

Копия кластера

Команды SQL для восстановления данных с нуля

- + можно сделать копию отдельного объекта или базы
- + можно восстановиться на кластере другой основной версии
- + можно восстановиться на другой архитектуре
- невысокая скорость

Существует два вида резервирования: логическое и физическое.

Логическое резервирование — это набор команд SQL, восстанавливающая кластер (или базу данных, или отдельный объект) с нуля.

Такая копия представляет собой по сути обычный текстовый файл, что дает известную гибкость. Например, можно сделать копию только тех объектов, которые нужны; можно отредактировать файл, изменив имена или типы данных и т. п.

Кроме того, команды SQL можно выполнить на другой версии СУБД (при наличии совместимости на уровне команд) или на другой архитектуре (то есть не требуется двоичная совместимость).

Однако для большой базы этот механизм неэффективен, поскольку выполнение команд займет много времени. К тому же восстановить систему из такой резервной копии можно только на тот момент, в который она была сделана.

<https://postgrespro.ru/docs/postgresql/10/backup-dump>

Резервное копирование

вывод таблицы или результатов запроса в файл, на консоль или в программу

Восстановление

добавление строк из файла или с консоли к существующей таблице

Серверный вариант

команда SQL COPY

файл должен быть доступен
пользователю postgres
на сервере

Клиентский вариант

команда psql \COPY

файл должен быть доступен
запустившему psql
на клиенте

Если требуется сохранить только содержимое одной таблицы, можно воспользоваться командой COPY.

Команда позволяет записать таблицу (или результат произвольного запроса) либо в файл, либо на консоль, либо на вход произвольной программе. При этом можно указать ряд параметров, таких как формат (текстовый, csv или двоичный), разделитель полей, текстовое представление null и др.

Другой вариант команды, наоборот, считывает из файла или из консоли строки с полями и записывает их в таблицу. Таблица при этом не очищается, новые строки добавляются к уже существующим.

Команда COPY работает существенно быстрее, чем аналогичные команды INSERT — клиенту не нужно много раз обращаться к серверу, а серверу не нужно много раз анализировать команды.

<https://postgrespro.ru/docs/postgresql/10/sql-copy>

В psql существует клиентский вариант команды COPY с аналогичным синтаксисом. В отличие от серверного варианта COPY, который является командой SQL, клиентский вариант — это команда psql.

Указание имени файла в команде SQL соответствует файлу на сервере БД. У пользователя, под которым работает PostgreSQL (обычно postgres), должен быть доступ к этому файлу. В клиентском варианте обращение к файлу происходит на клиенте, а на сервер передается только содержимое.

<https://postgrespro.ru/docs/postgresql/10/app-psql>

Резервное копирование

выдает на консоль или в файл либо SQL-скрипт,
либо архив в специальном формате с оглавлением
поддерживает параллельное выполнение
позволяет ограничить набор выгружаемых объектов
(таблицы, схемы, только DML или только DDL и т. п.)

Восстановление

SQL-скрипт — `psql`
формат с оглавлением — `pg_restore`
(позволяет ограничить набор объектов при восстановлении)
поддерживает параллельное выполнение
новая база должна быть создана из шаблона `template0`
заранее должны быть созданы роли и табличные пространства
после восстановления имеет смысл выполнить сбор статистики

Для создания полноценной резервной копии базы данных используется утилита `pg_dump`. В зависимости от указанных параметров, результатом работы является либо SQL-скрипт, содержащий команды, создающие выбранные объекты, либо файл в специальном формате с оглавлением.

Чтобы восстановить объекты из SQL-скрипта, достаточно прогнать его через `psql`.

<https://postgrespro.ru/docs/postgresql/10/app-pgdump>

Для восстановления резервной копии в специальном формате требуется другая утилита — `pg_restore`. Она читает файл и преобразует его в обычные команды `psql`. Преимущество в том, что набор объектов можно ограничить не при создании резервной копии, а уже при восстановлении. Кроме того, создание резервной копии в спец. формате и восстановление из неё может выполняться параллельно.

<https://postgrespro.ru/docs/postgresql/10/app-pgrestore>

Базу данных для восстановления надо создавать из шаблона `template0`, так как все изменения, сделанные в `template1`, также попадут в резервную копию. Кроме того, заранее должны быть созданы необходимые роли и табличные пространства, поскольку эти объекты не относятся ко всему кластеру. После восстановления базы имеет смысл выполнить команду `ANALYZE`, которая соберет статистику.

Резервирование

- сохраняет весь кластер, включая роли и табличные пространства
- выдает на консоль или в файл SQL-скрипт
- параллельное выполнение не поддерживается, но можно выгрузить только глобальные объекты и воспользоваться pg_dump

Восстановление

- с помощью psql

Чтобы создать резервную копию всего кластера, включая роли и табличные пространства, можно воспользоваться утилитой pg_dumpall.

Поскольку pg_dumpall требуется доступ ко всем объектам всех БД, имеет смысл запускать ее от имени суперпользователя. Утилита по очереди подключается к каждой БД кластера и выгружает информацию с помощью pg_dump. Кроме того, она сохраняет и данные, относящиеся к кластеру в целом.

Результатом работы pg_dumpall является скрипт для psql. Другие форматы не поддерживаются. Это означает, что pg_dumpall не поддерживает параллельную выгрузку данных, что может оказаться проблемой при больших объемах данных. В таком случае можно воспользоваться ключом --globals-only, чтобы выгрузить только роли и табличные пространства, а сами базы данных выгрузить с помощью pg_dump.

<https://postgrespro.ru/docs/postgresql/10/app-pg-dumpall>

Что такое физическое резервное копирование

Холодное и горячее резервное копирование

Протокол репликации

Автономные резервные копии

Непрерывная архивация журналов предзаписи

Используется механизм восстановления после сбоя:
копия данных и журналы предзаписи

- + скорость восстановления
- + можно восстановить кластер на определенный момент времени
- нельзя восстановить отдельную базу данных, только весь кластер
- восстановление только на той же основной версии и архитектуре

Физическое резервирование использует механизм восстановления после сбоев. Для этого требуются:

- копия файлов кластера (базовая резервная копия),
- набор журналов предзаписи, необходимых для восстановления согласованности.

Если файловая система уже согласована (копия снималась при корректно остановленном сервере), то журналы не требуются.

Однако наличие архива журналов позволяет из базовой резервной копии получить состояние кластера на любой момент времени. Таким образом можно иметь резервную копию практически на момент сбоя (либо сознательно восстановить систему на некоторый момент в прошлом).

Высокая скорость восстановления и возможность создавать копию «на лету», не выключая сервер, делает физическое резервирование основным инструментом периодического резервного копирования.

<https://postgrespro.ru/docs/postgresql/10/backup-file>

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

Горячо или холодно?

	Холодное резервирование		Горячее резервирование
Файловая система копируется при...	выключенном сервере	неаккуратно выключенном сервере	работающем сервере 
Журналы предварительной записи...	не нужны	нужны с последней контрольной точки 	нужны за время копирования ФС 

Физическое резервирование так или иначе предполагает создание копии файловой системы.

Если копия создается при выключенном сервере, она называется «холодной». Такая копия либо содержит согласованные данные (если сервер был выключен аккуратно), либо содержит все необходимые для восстановления журналы (например, если используется снимок данных средствами операционной системы). Это упрощает восстановление, но требует останова сервера.

Если копия создается при работающем сервере (что требует определенных действий — просто так копировать файлы нельзя), она называется «горячей». В этом случае процедура сложнее, но позволяет обойтись без останова.

При горячем резервировании копия ФС будет содержать несогласованные данные. Однако механизм восстановления после сбоев можно успешно применить и к восстановлению из резервной копии. Для этого потребуются журналы предзаписи как минимум за время копирования файлов.

Автономная копия содержит и файлы данных, и WAL

Резервное копирование — утилита `pg_basebackup`

- подключается к серверу по протоколу репликации
- выполняет контрольную точку
- переключается на следующий сегмент WAL
- копирует файловую систему в указанный каталог
- переключается на следующий сегмент WAL
- сохраняет все сегменты WAL, сгенерированные за время копирования

Восстановление

- разворачиваем созданную автономную копию
- запускаем сервер

11

Для создания горячей резервной копии существует утилита `pg_basebackup`.

В начале утилита выполняет контрольную точку и переключает журнал на новый файл. Затем делается копия файловой системы кластера, после чего журнал снова переключается на новый файл.

Все файлы WAL за время между переключениями также копируются в резервную копию — такая копия называется автономной, поскольку содержит в себе все необходимое для восстановления.

Для восстановления достаточно развернуть резервную копию и запустить сервер. При необходимости он выполнит восстановление согласованности с помощью имеющихся файлов WAL и будет готов к работе.

<https://postgrespro.ru/docs/postgresql/10/app-pgbasebackup>

Протокол

- получение потока журнальных записей
- команды управления резервным копированием и репликацией

Обслуживается процессом `wal_sender`

Параметр `wal_level = replica`

Слот репликации

- серверный объект для получения журнальных записей
- помнит, какая запись была считана последней
- сегмент WAL не удаляется, пока он полностью не прочитан через слот

Чтобы сохранить все необходимые для восстановления файлы WAL, сгенерированные сервером за время копирования файлов, утилита подключается к серверу по специальному протоколу репликации. Несмотря на название, это протокол используется не только для репликации (о которой пойдет речь в следующей теме), но и для резервного копирования. Протокол позволяет получать поток журнальных записей параллельно с копированием файлом.

Чтобы сервер не удалил необходимые файлы WAL преждевременно, может использоваться слот репликации.

Для того, чтобы подключение было возможно, необходим ряд настроек.

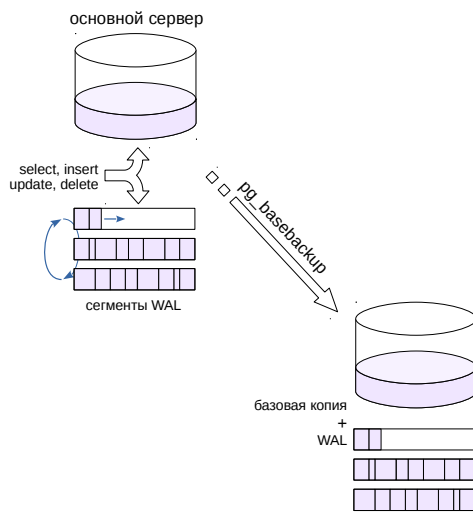
Во-первых, роль должна обладать атрибутом `REPLICATION` (или быть суперпользователем). Кроме того, этой роли должно быть выдано разрешение в конфигурационном файле `pg_hba.conf`.

Во-вторых, параметр `max_wal_senders` должен быть установлен в достаточно большое значение. Этот параметр ограничивает число одновременно работающих процессов `wal_sender`, обслуживающим подключения по протоколу репликации.

В-третьих, параметр `wal_level`, определяющий количество информации в журнале, должен быть установлен в значение `replica`.

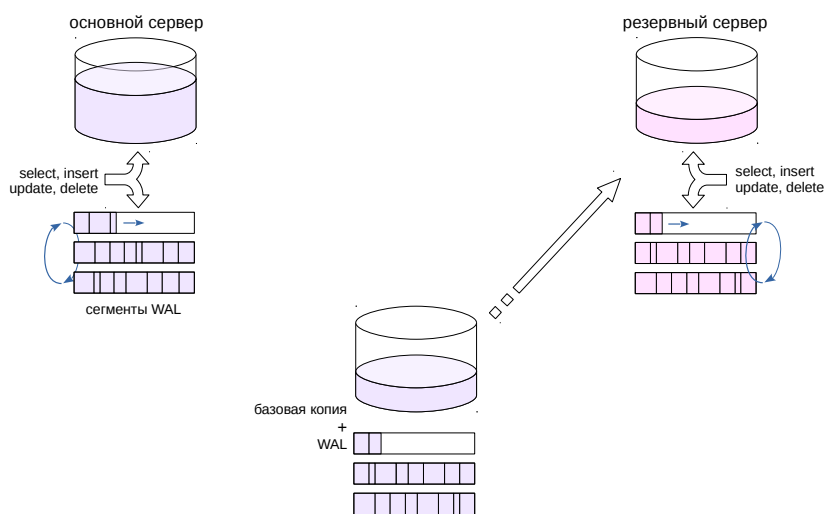
Начиная с версии 10, настройки по умолчанию уже включают все эти требования (при локальном подключении).

<https://postgrespro.ru/docs/postgresql/10/protocol-replication>



На этом рисунке слева представлен основной сервер. Он обрабатывает поступающие запросы; при этом формируются записи WAL и изменяется состояние баз данных (сначала в буферном кэше, потом на диске). Сегменты WAL циклически перезаписываются (точнее, удаляются старые сегменты, так как имена файлов уникальны).

Внизу рисунка (а в реальной жизни — обычно на другом сервере) изображена созданная резервная копия — базовая копия данных и набор файлов WAL.



При восстановлении базовая резервная копия, включающая необходимые файлы WAL, разворачивается, например, на другом сервере (справа на рисунке).

После старта сервера он восстанавливает согласованность и приступает к работе. Восстановление происходит на тот момент времени, на который была сделана резервная копия. Разумеется, за прошедшее время основной сервер может уйти далеко вперед.

Файловый архив

сегменты WAL копируются в архив по мере заполнения
механизм работает под управлением сервера
неизбежны задержки попадания данных в архив

Потоковый архив

в архив постоянно записывается поток журнальных записей
требуются внешние средства
задержки минимальны

Дальнейшее развитие идеи горячей резервной копии: раз у нас есть копия файловой системы и журналы упреждающей записи, то, постоянно сохраняя новые журналы, мы сможем восстановить систему не только на момент копирования файлов, но и вообще на произвольный момент.

Для этого есть два способа. Первый состоит в том, чтобы не просто перезаписывать файлы WAL, а предварительно откладывая их куда-то в архив. Это можно реализовать специальными настройками сервера. К сожалению, при таком варианте файл WAL не попадет в архив, пока сервер не переключится на запись в другой файл.

Второй способ — постоянно читать журнальные записи по протоколу репликации и записывать их в тот же архив. При таком варианте задержки будут минимальны, но потребуются настроить отдельную от сервера утилиту для получения данных из потока.

Процесс archiver

Параметры

`archive_mode = on`

`archive_command` команда shell для копирования сегмента WAL в отдельное хранилище

`archive_timeout` максимальное время для переключения на новый сегмент WAL

Алгоритм

при заполнении сегмента WAL вызывается команда `archive_command`

если команда завершается со статусом 0, сегмент удаляется

если команда возвращает не 0 (в частности, если команда не задана), сегмент остается до тех пор, пока попытка не будет успешной

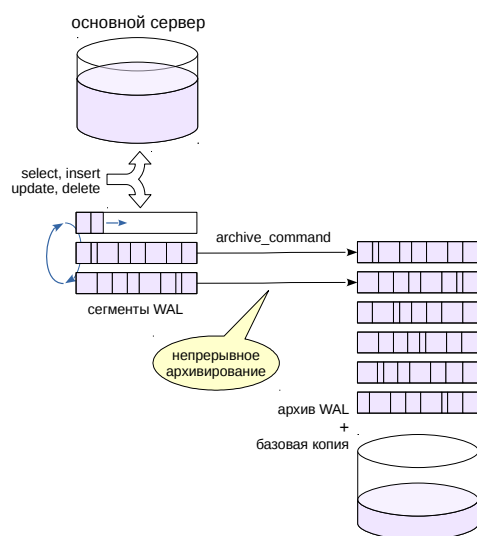
16

Файловый архив реализуется фоновым процессом archiver.

PostgreSQL позволяет определить для копирования произвольную команду shell в параметре `archive_command`. Сам механизм включается параметром `archive_mode = on`. Параметр `archive_timeout` позволяет указать максимальное время переключения на новый сегмент WAL — это позволяет при невысокой активности сервера сохранять журналы не реже, чем указанное значение. Переключение на новый сегмент можно выполнить и вручную с помощью функции `pg_switch_log()`.

Общий алгоритм таков. При заполнении очередного сегмента WAL вызывается функция копирования. Если она завершается с нулевым статусом, то сегмент может быть удален. Если же нет, то сегмент (и следующие за ним) не будет удален, а сервер будет периодически пытаться выполнить команду, пока не получит 0.

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



На этом рисунке изображен основной сервер с настроенным непрерывным архивированием: заполненные сегменты WAL копируются в отдельный архив с помощью команды в параметре `archive_command`. Обычно такой архив расположен на другом сервере. Там же создается базовая резервная копия (или несколько копий, сделанных в разные моменты времени).

Утилита `pg_receivewal`

подключается по протоколу репликации (можно использовать слот)
и направляет 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

в отличие от файлового архива, записи пишутся постоянно

при переходе на новый сервер надо перенастраивать параметры

Другое решение — использование утилиты `pg_receivewal` для записи сегментов в архив по протоколу потоковой репликации (до версии 10 эта утилита называлась `pg_receivexlog`).

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.

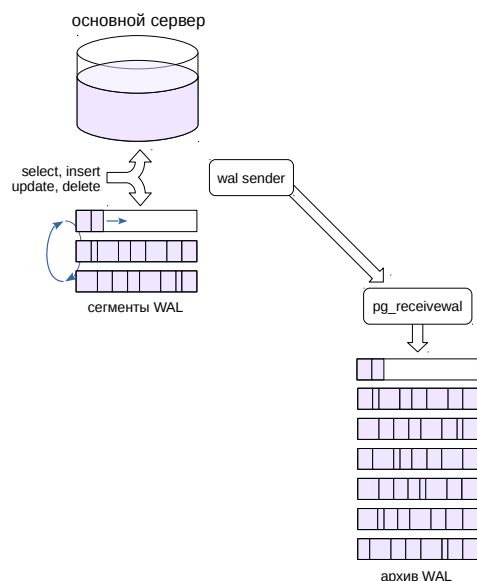
Утилита формирует файлы аналогично тому, как это делает сервер, и записывает их в указанный каталог. Еще не заполненные сегменты записываются с префиксом `.partial`.

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 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.

<https://postgrespro.ru/docs/postgresql/10/app-pgreceivewal>



Утилита `pg_receivewal` подключается к серверу по протоколу потоковой репликации. Подключение обрабатывается отдельным процессом `wal sender` (это необходимо учесть при установке параметра `max_wal_senders`).

Утилита записывает данные, не дожидаясь получения всего сегмента.

Настроенное непрерывное архивирование журналов

Резервное копирование — pg_basebackup

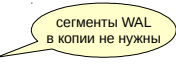
подключается к серверу по протоколу репликации

выполняет контрольную точку

переключается на следующий сегмент WAL

копирует файловую систему в указанный каталог

переключается на следующий сегмент WAL



сегменты WAL
в копии не нужны

Восстановление

разворачиваем резервную копию

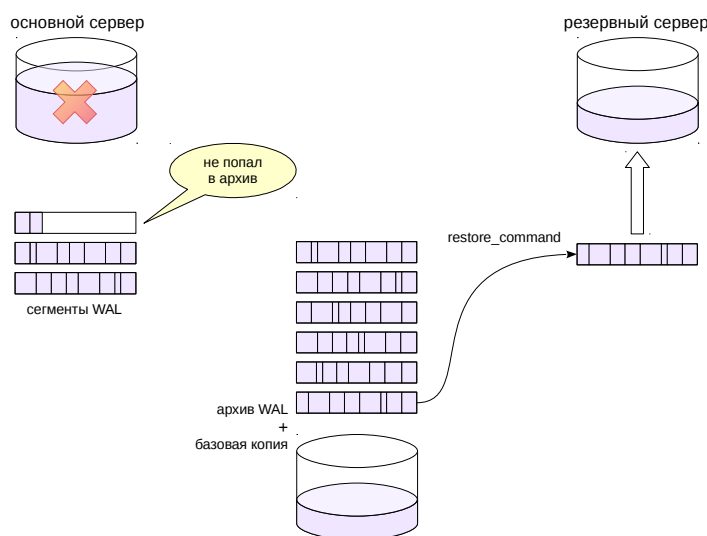
создаем управляющий файл `$PGDATA/recovery.conf`

(чтение WAL из архива, указание целевой точки восстановления)

запускаем сервер

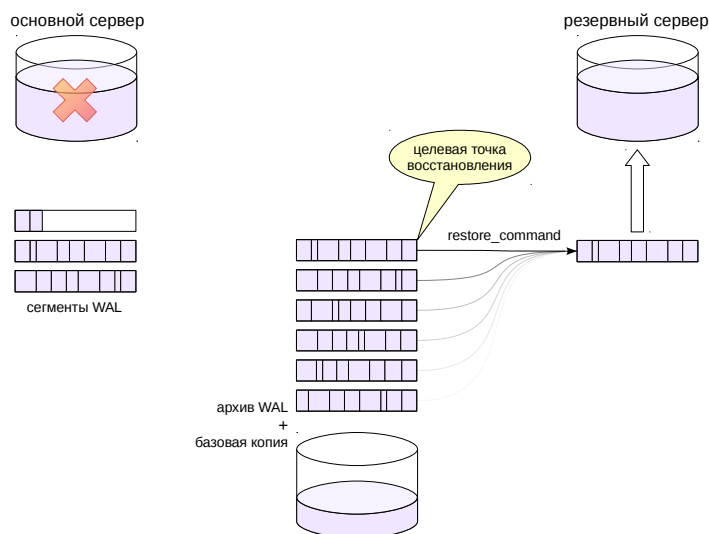
Для создания резервной копии при настроенном непрерывном архивировании используется та же утилита `pg_basebackup`, только с другим набором параметров. Единственное отличие состоит в том, что в резервную копию не сохраняются файлы WAL, поскольку они уже есть в архиве.

Восстановление в таком случае выполняется более сложно. Помимо разворачивания базовой резервной копии, требуется создать файл `$PGDATA/recovery.conf`, управляющий восстановлением. В этом файле указывается команда `restore_command` (обратная `archive_command` — она копирует нужные файлы из архива в каталог сервера) и целевая точка восстановления.



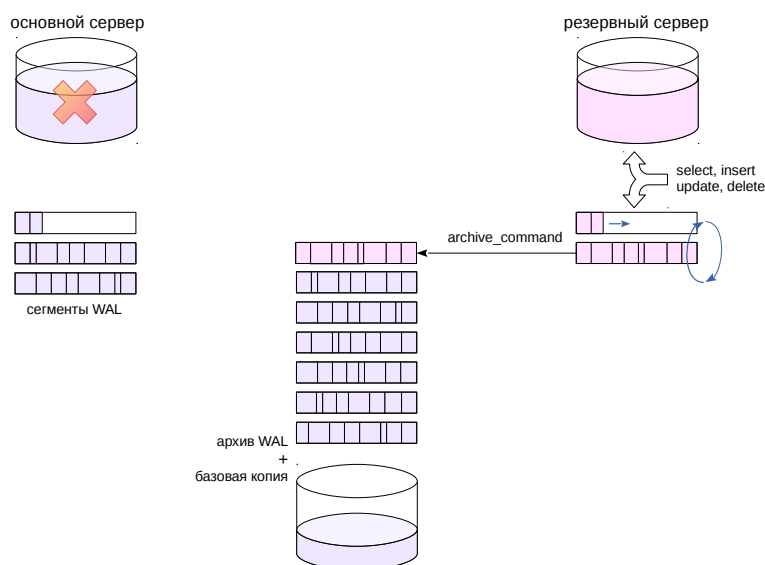
Процедура восстановления (например, при выходе сервера из строя основного сервера), выполняется следующим образом. На другом (или на том же) сервере разворачивается базовая резервная копия и создается файл `recovery.conf`. Сервер запускается и начинает чтение и применение сегментов WAL из архива, используя `restore_command`.

Следует обратить внимание, что последний незаполненный сегмент WAL основного сервера не попадет в архив, если используется файловый архив. Однако сегмент можно вручную подложить резервному серверу в каталог `pg_wal` (до версии 10 — `pg_xlog`), если есть такая возможность. Конечно, таких сегментов может оказаться и несколько, но только как следствие какого-то сбоя при архивировании.



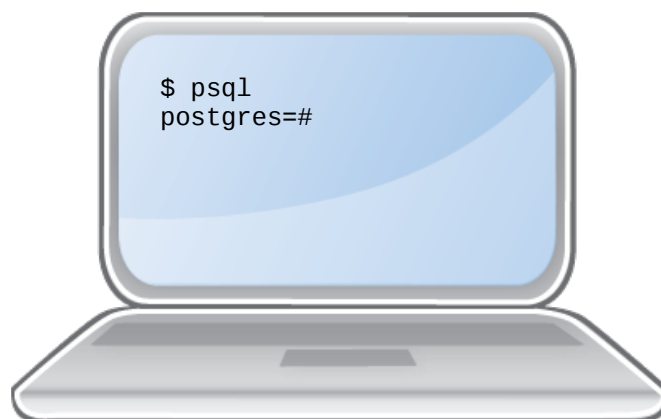
Прочитав все сегменты WAL из архива (а после этого — все дополнительные сегменты WAL из каталога `pg_wal`), резервный сервер тем самым «догоняет» состояние баз данных до актуального. Максимально возможная потеря — незаполненный сегмент WAL, не попавший в архив, если его по каким-то причинам невозможно скопировать.

Указание в файле `recovery.conf` целевой точки восстановления позволяет остановить применение журнальных записей в желаемый момент.



После этого резервный сервер переходит в обычный режим работы, принимает запросы, записывает сегменты WAL в архив и так далее, выступая в качестве нового полноценного основного сервера.

Если развернутый сервер предполагается использовать вместо основного, то его имеет смысл располагать на таком же, или, по крайней мере, на сравнимом по характеристикам аппаратуры сервере, чтобы избежать снижения производительности.



Логическая резервная копия —
команды SQL для восстановления состояния объектов

команда `copy`, утилиты `pg_dump` и `pg_dumpall`

Физическая резервная копия —
копия файлов кластера + набор файлов WAL

утилита `pg_basebackup`

Архив журнальных файлов

файловый или потоковый

позволяет восстановить систему на произвольный момент времени

1. Создайте базу данных и таблицу в ней с несколькими строками.
2. Сделайте логическую копию базы данных с помощью утилиты `pg_dump`.
3. Удалите базу данных и восстановите ее из сделанной копии.
4. Сделайте автономную физическую резервную копию кластера с помощью утилиты `pg_basebackup`.
5. Измените таблицу.
6. Восстановите новый кластер из сделанной резервной копии и проверьте, что база данных не содержит более поздних изменений.