

Репликация

Логическая репликация



Авторские права

© Postgres Professional, 2018 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 непрямым, 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.

Отличия логической репликации от физической

Поставщики и подписчики

Логическое декодирование и слоты логической репликации

Конфликты и их разрешение

Выполнение триггеров на подписчике

Физическая

- мастер-реплика: поток данных только в одну сторону
- трансляция потока журнальных записей или файлов журнала
- требуется двоичная совместимость серверов
- возможна репликация только всего кластера

Логическая

- публикация-подписчики: у сервера нет выделенной роли
- трансляция изменений табличных строк
- необходим уровень журнала `logical`
- требуется совместимость на уровне протокола
- возможна выборочная репликация отдельных таблиц

Как мы помним, при физической репликации серверы имеют назначенные роли: мастер и реплика. Мастер передает на реплику журнальные записи (в виде файлов или потока записей); реплика применяет эти записи к своим файлам данных. Применение происходит чисто механически, без «понимания смысла» изменений, поэтому важна двоичная совместимость между серверами (одинаковые платформы и основные версии PostgreSQL). Поскольку журнал общий для всего кластера, то и реплицировать можно только кластер целиком.

При логической репликации на одном сервере создается публикация, другие серверы могут на нее подписаться. У сервера нет выделенной роли: один и тот же сервер может как публиковать изменения, так и подписываться на другие (или даже свои) подписки. Подписчику передается информация об изменениях строк в таблицах в платформо-независимом виде; двоичная совместимость не требуется. Для работы логической репликации в журнале публикующего сервера необходима дополнительная информация (параметр `wal_level = logical`). Логическая репликация позволяет транслировать не все изменения, а только касающиеся определенных таблиц.

Логическая репликация доступна, начиная с версии 10; более ранние версии должны были использовать расширение `pglogical`, либо организовывать репликацию с помощью триггеров.

Публикация

- объект базы данных
- выдает изменения данных построчно
- изменения в порядке фиксации транзакций

Подписчики

- подписываются, получают и применяют изменения
- таблицы и столбцы сопоставляются по полным именам
- поддерживается «бесшовная» синхронизация данных
- при создании подписки
- могут возникать конфликты с локальными данными

Логическая репликация использует модель «публикация-подписчики».

На одном сервере создается *публикация*, которая может включать ряд таблиц одной базы данных. Для репликации из нескольких баз данных потребуется создать несколько публикаций.

Публикация включает в себя изменения, происходящие с таблицами: эти изменения передаются на уровне строк («в таблице такой-то такая-то строка изменилась таким-то образом»).

Изменения выдаются не сразу, а только при фиксации транзакции.

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

Другие серверы могут создавать *подписки* на публикации, получать и применять изменения.

Применение изменений всегда происходит построчно. Хотя каждое отдельное изменение не требует накладных расходов на разбор и планирование запроса, массовые изменения из-за этого будут выполняться медленнее.

Таблицы идентифицируются по полным именам (включая схему), столбцы также идентифицируются по именам. Это позволяет подписчику использовать отличающуюся схему данных (например, иметь в таблице дополнительные столбцы).

Есть возможность автоматической начальной синхронизации содержимого таблиц при создании подписки.

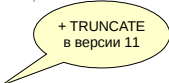
<https://postgrespro.ru/docs/postgresql/10/logical-replication-subscription>

Реплицируются не все изменения

только команды INSERT, UPDATE, DELETE

только базовые таблицы

(не реплицируются последовательности, материализованные представления, секционированные таблицы)



+ TRUNCATE
в версии 11

Подписку можно создать только на основном сервере

не работает на физических репликах

Циклы в репликации не обрабатываются

нельзя реплицировать одну и ту же таблицу
с одного сервера на другой и обратно

Реплицируются только изменения базовых таблиц, вызванные командами INSERT, UPDATE и DELETE. В частности, не реплицируются команды DDL, что означает необходимость для подписчика предварительно самостоятельно создать все необходимые таблицы. Также не реплицируются другие объекты, объединяемые термином relation — последовательности, материализованные представления, секционированные таблицы.

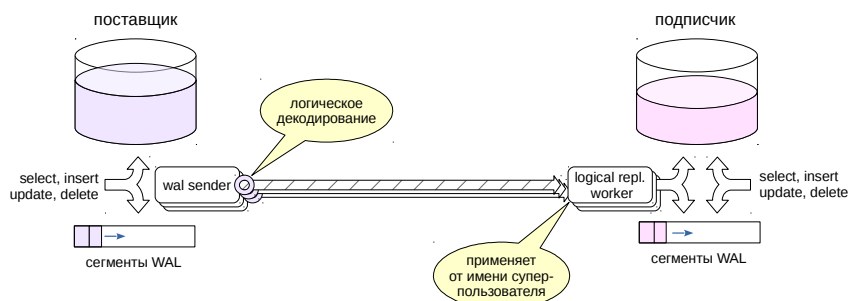
Если используется физическая репликация, то публикацию можно создать только на основном сервере.

Нет возможности организовать репликацию одной и той же таблицы между двумя серверами: изменения, сделанные на первом сервере, применяются вторым и тут же снова пересылаются первому — который скорее всего не сможет их применить из-за нарушения ограничений целостности.

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

Многие из этих ограничений будут устранены в следующих выпусках PostgreSQL.

Схема работы



max_wal_senders
max_replication_slots

max_logical_replication_workers
max_worker_processes

6

Данные об изменениях таблиц передаются подписчику тем же процессом wal sender, что и при обычной потоковой репликации. Так же, как и при потоковой репликации, этот процесс читает журнал предзаписи, но не просто транслирует прочитанные записи, а предварительно декодирует их. В отличие от физической репликации, в обязательном порядке используется слот логической репликации.

На подписчике информацию принимает фоновый процесс logical replication worker и применяет ее. В это же время сервер-подписчик принимает обычные запросы и на чтение, и на запись.

Обратите внимание, что на публикующем сервере может быть запущено много процессов wal sender — по одному на каждого подписчика каждой публикации. Настройки *max_wal_senders* и *max_replication_slots* должны быть выставлены в достаточное число.

На сервере-подписчике необходимо установить параметры *max_logical_replication_workers* (для процессов, принимающих изменения по подписке) и в целом *max_worker_processes* (как минимум на единицу больше, так как есть еще процесс logical replication launcher, но вообще этот пул используется и на другие нужды).

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

Переупорядочивающий буфер

wal sender читает журнальные записи и накапливает их в буфере, раскладывая по транзакциям
буфер в локальной памяти; при необходимости сбрасывается на диск

Модуль вывода

получает накопленные записи при фиксации транзакции
декодирует записи, формируя сообщения об операциях над табличными строками в платформно-независимом формате
фильтрует сообщения, на которые подписан получатель

Слот логической репликации

гарантирует, что подписчик не пропустит изменения

Полезно представлять внутреннее устройство логической репликации. Журнальные записи читаются процессом wal sender и раскладываются по отдельным транзакциям в специальном буфере в оперативной памяти. Это делается для того, чтобы при фиксации транзакции можно было взять все изменения, сделанные именно этой транзакцией, и передать их подписчику. При превышении определенного порога буфер начинает сбрасываться на диск (в каталог PGDATA/pg_replslots).

Заметим, что при наличии нескольких подписчиков и, следовательно, нескольких процессов wal sender, *каждый* из этих процессов будет самостоятельно читать WAL: буфер, упорядочивающий записи, находится в *локальной* памяти каждого процесса wal sender.

Когда транзакция фиксируется, ее изменения передаются *модулю вывода*, который *декодирует* их и представляет в платформно-независимом (текстовом) формате. Процесс wal sender передает эти декодированные сообщения подписчику (если он на них подписан) через *слот логической репликации*. Этот слот похож на обычный репликационный слот, но к нему привязан *модуль вывода*.

Для декодирования необходимо знать структуру таблиц на момент формирования журнальной записи. Текущего состояния системного каталога недостаточно, так как в ходе транзакции у таблицы, например, может измениться число столбцов. Для этого в журнал на уровне logical дополнительно записываются исторические снимки данных системного каталога (похожие на обычные снимки данных MVCC).

<https://postgrespro.ru/docs/postgresql/10/logicaldecoding>

Режимы идентификации для изменения и удаления

`ALTER TABLE ... REPLICA IDENTITY ...`

- а) столбцы первичного ключа (по умолчанию)
- б) столбцы указанного уникального индекса с ограничением NOT NULL
- в) все столбцы
- г) без идентификации (по умолчанию для системного каталога)

Конфликты — нарушение ограничений целостности

репликация приостанавливается до устранения конфликта
требуется вручную исправить данные на подписчике

Вставка новых строк на подписчике происходит достаточно просто.

Интереснее обстоит дело при изменениях и удалениях — в этом случае надо как-то идентифицировать старую версию строки. По умолчанию для этого используются столбцы первичного ключа, но при определении таблицы можно указать и другие способы: использовать уникальный индекс или использовать все столбцы. В первом случае для поиска строки будет использоваться соответствующий индекс, во втором — полное сканирование таблицы (что крайне неэффективно для больших таблиц).

Можно вообще отказаться от поддержки репликации для некоторых таблиц (по умолчанию — таблицы системного каталога).

Поскольку таблицы на публикующем сервере и на подписчике могут изменяться независимо друг от друга, при вставке новых версий строк возможно возникновение конфликта — нарушение ограничения целостности. В этом случае процесс применения записей приостанавливается до тех пор, пока конфликт не будет разрешен. Автоматического разрешения пока не существует; нужно вручную исправить данные на подписчике так, чтобы устранить конфликт.

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



Логическая репликация: модель «публикация-подписчики»

Передаются изменения табличных строк

Возможна выборочная репликация отдельных таблиц

Не требуется двоичная совместимость серверов

Пока доступны только базовые возможности,
но функционал активно развивается

1. Создайте две базы данных на одном сервере.
2. В первой базе данных создайте таблицу с первичным ключем и добавьте в нее несколько строк.
3. Перенесите определение созданной таблицы во вторую базу данных с помощью логической резервной копии.
4. Настройте логическую репликацию таблицы из первой базы данных во вторую.
5. Проверьте работу репликации.
6. Удалите подписку.

3. Воспользуйтесь утилитой `pg_dump` с ключом `--schema-only`.

4. Если попробовать выполнить это обычным образом, команда создания подписки «повиснет» из-за того, что она должна дожидаться завершения активных транзакций на публикующем сервере, то есть и самой себя в том числе. В таком случае необходимо заранее создать слот логической репликации, как описано в документации: <https://postgrespro.ru/docs/postgresql/10/sql-createsubscription>