



ОНЛАЙН-ОБРАЗОВАНИЕ

# Онлайн-образование




**Не забыть включить запись!**







# Меня хорошо видно && слышно?

Ставьте  , если все хорошо  
Напишите в чат, если есть проблемы



# Тема: Виды и устройство репликации в PostgreSQL. Практика применения.



Коробков Виктор



# Правила вебинара



Активно участвуем



Задаем вопрос в чат или голосом



Off-topic обсуждаем в группе Telegram



Вопросы вижу в чате, могу ответить не сразу

# Цели вебинара:

**1** Настройка репликации

**2** Выбор оптимального плана репликации

# Смысл | зачем вам это уметь

**1** Обеспечить высокую доступность

**2** Организовать масштабируемость

# Маршрут вебинара

Задачи репликации



Физическая репликация



Логическая репликация



Примеры применения



# Задачи репликации

**Репликация** - процесс синхронизации нескольких копий одного объекта (например, нескольких кластеров PostgreSQL на разных серверах).

Решает задачи:

- ***отказоустойчивости;***
- ***масштабируемости.***

# Зачем нужна репликация ???

1. Высокая доступность. Бэкап это хорошо, но нужно время на его развертывание.



# Зачем нужна репликация ???

1. Высокая доступность. Бэкап это хорошо, но нужно время на его развертывание.
2. Бэкап лучше делать с реплики, а не мастера.

# Зачем нужна репликация ???

1. Высокая доступность. Бэкап это хорошо, но нужно время на его развертывание.
2. Бэкап лучше делать с реплики, а не мастера.
3. Что делать, когда закончились физические ядра и память у сервера? Горизонтально масштабировать!!!



# Зачем нужна репликация ???

1. Высокая доступность. Бэкап это хорошо, но нужно время на его развертывание.
2. Бэкап лучше делать с реплики, а не мастера.
3. Что делать, когда закончились физические ядра и память у сервера? Горизонтально масштабировать!!!
4. Геораспределение нагрузки

# Виды репликации

1. **Физическая** репликация - описание изменений на уровне файлов. Побайтовая копия данных.
2. **Логическая** репликация - изменения данных в терминах строк таблиц. Более высокий уровень, чем файлы данных.

[Памятка евангелиста PostgreSQL: репликаны против репликации](#)



The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. The diagram consists of numerous small dots connected by thin white lines, forming a complex web that suggests digital connectivity or data flow. The text 'Физическая репликация' is centered within this blue band in a large, white, sans-serif font.

# Физическая репликация



# Физическая репликация

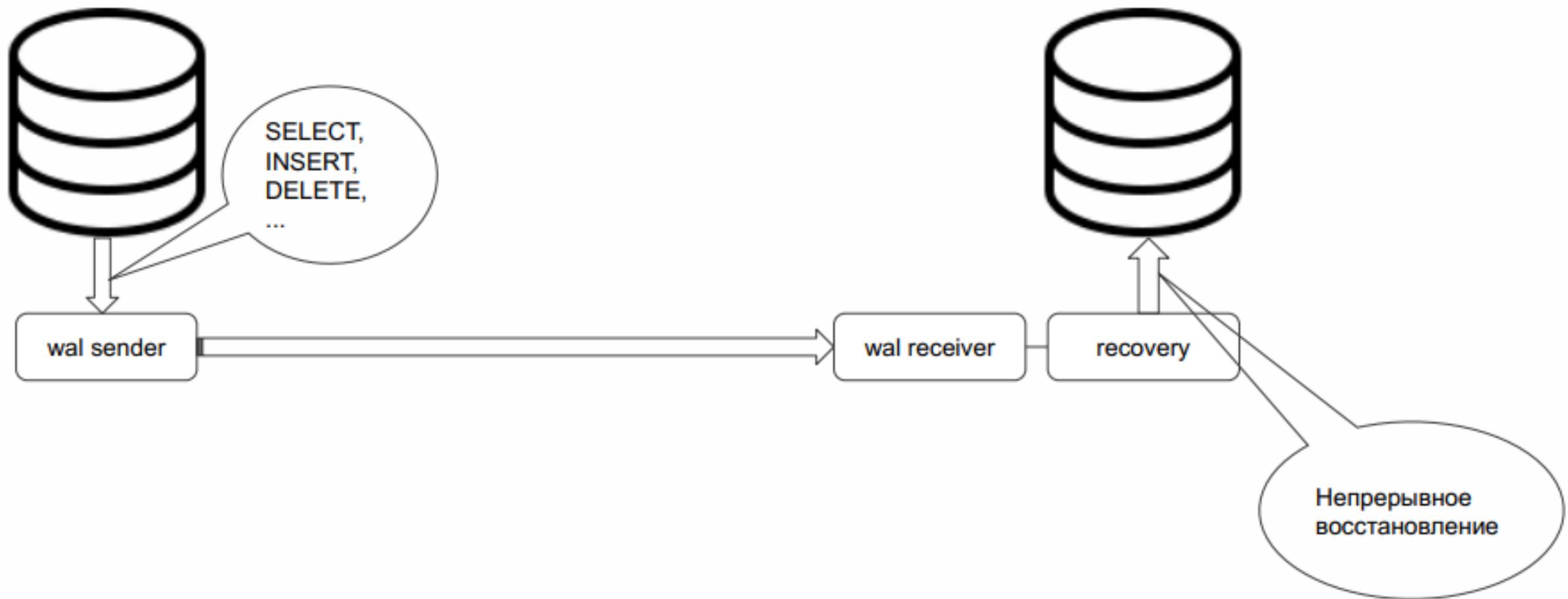
1. Записи WAL передаются на реплику и применяются:
  - поток данных только в одну сторону
  - реплицируется кластер целиком, выборочная реплика невозможна
2. Реплика — точная копия мастера:
  - одна и та же основная версия сервера
  - полностью совместимые архитектуры и платформы
3. Реплика доступна только для чтения.

# Использование физической репликации

1. Горячий резерв для высокой доступности
2. Балансировка OLTP-нагрузки
3. Реплика для отчетов
4. Несколько реплик и каскадная репликация
5. Отложенная репликация



# Использование физической репликации



# Возможности реплики

## Допускаются

- запросы на чтение данных (select, copy to, курсоры)
- установка параметров сервера (set, reset)
- управление транзакциями (begin, commit, rollback...)
- создание резервной копии (pg\_basebackup)

# Возможности реплики

## Не допускаются

- любые изменения (insert, update, delete, truncate, nextval...)
- блокировки, предполагающие изменение (select for update...)
- команды DDL (create, drop...), в том числе создание временных таблиц
- команды сопровождения (vacuum, analyze, reindex...)
- управление доступом (grant, revoke...)
- не срабатывают триггеры и пользовательские блокировки

## Vacuum?

- параметр hot\_standby\_feedback



# Процессы

Master	Replica
Wal sender	Wal receiver
Archiver	Archiver (archive_mode = always)
Checkpointner	Checkpointner
Stats collector	Stats collector
Wal writer	
Autovacuum	

# Виды физической репликации

1. Асинхронная репликация - последовательное выполнение транзакций на всех репликах.

Хорошая пропускная способность.

Появляется время отклика в течении которого отдельные реплики могут быть фактически неидентичными.

`synchronous_commit = off`

# Виды физической репликации

2. Синхронная репликация - коммит на реплике должен быть доступен до того, как он будет доступен на мастере.

**Плюсы** - в момент отказа получают абсолютно идентичные мастер и реплику. Могут потеряться последние транзакции которые еще не были зафиксированы, но мастер и реплика в любом случае идентичны.

**Минусы** - Проблемы с производительностью. Primary ждет ответа от реплики об удачном коммите.

В случаях возникновения проблем с репликой Primary останавливается и не может работать дальше до тех пор пока Реплика не будет восстановлена, либо до тех пор пока эта реплика не будет удалена из списка синхронных реплик на Primary сервере.



# Синхронная репликация

`synchronous_standby_names` = (список реплик)

`synchronous_commit` = on | remote\_write | remote\_apply

`remote_write` - дожидается только ответа о том, что информация дошла до реплики, но не факт, что произошла запись на диск

on - подтверждает, что произошла запись на диск в WAL файл

`remote_apply` - дает подтверждение тому, что запись применена в базе

[Потоковая репликация в PostgreSQL и пример фейловера](#)

# Куда смотреть

## На Мастере:

Процесс можно отслеживать в представлении ***pg\_current\_wal\_lsn()***, остальные — представление ***pg\_stat\_replication***. Реплика передает мастеру статус репликации при каждой записи на диск, но как минимум раз в ***wal\_receiver\_status\_interval*** секунд (по умолчанию — 10 секунд).

Если используется слот репликации, то информацию о нем можно получить из представления ***pg\_replication\_slots***.

## На Реплике:

В представлении ***pg\_stat\_wal\_receiver*** и с помощью функций ***pg\_last\_wal\_receive\_lsn()*** и ***pg\_last\_wal\_replay\_lsn()***.

# Интересные параметры

**max\_worker\_processes** - должно быть таким же или большим, чем на Мастер сервере. Иначе нельзя будет делать запросы на Реплику.

**wal\_level** - replica или logical

**archive\_mode** - on|off

**wal\_receiver\_status\_interval** – частота передачи сигнала от реплики мастеру (по умолчанию 10 с.)

**wal\_retrieve\_retry\_interval** – время ожидания реплики, прежде чем повторить попытку получить WAL (5 с.)

**recovery\_min\_apply\_delay** – время задержки на восстановление

**max\_replication\_slots** – количество слотов



# Физическая репликация - ПРАКТИКА

Начиная с версии 10, все необходимые настройки уже присутствуют по умолчанию:

- `wal_level = replica;`
- `max_wal_senders`
- разрешение на подключение в `pg_hba.conf` по протоколу репликации.

Создадим 2 кластер

```
$ pg_createcluster -d /var/lib/postgresql/14/main2 14 main2
```

Удалим оттуда файлы

```
$ rm -rf /var/lib/postgresql/14/main2
```

Сделаем бэкап нашей БД. Ключ `-R` создаст заготовку управляющего файла `recovery.conf` (запуск на вторичном сервере, если другой хост то `-h`)

```
$ pg_basebackup -p 5432 -R -D /var/lib/postgresql/14/main2
```

Зададим другой порт (до версии 10)

```
-- $ echo 'port = 5433' >> /var/lib/postgresql/14/main2/postgresql.auto.conf
```

Добавим параметр горячего резерва, чтобы реплика принимала запросы на чтение (до версии 10)

```
--$ echo 'hot_standby = on' >> /var/lib/postgresql/14/main2/postgresql.auto.conf
```

Стартуем кластер

```
$ pg_ctlcluster 14 main2 start
```

Смотрим как стартовал

```
$ pg_lsclusters
```

# Физическая репликация - ПРАКТИКА

Перевод реплики в состояние мастера

- **\$ pg\_ctlcluster 14 main2 promote**

*Что произойдет?*

# Физическая репликация - ПРАКТИКА

Перевод реплики в состояние мастера

- **\$ pg\_ctlcluster 14 main2 promote**

Получим 2 разных независимых сервера



# Повышение Реплики до Мастера

## Причины

- плановое переключение (switchover):
- останов основного сервера для проведения технических работ
- аварийное переключение (failover):
- переход на реплику из-за сбоя основного сервера

## Процедура

- убедиться, что мастер остановлен
- переключение вручную
- автоматическое переключение отсутствует

# Подключение старого Мастера

## **Простое подключение бывшего мастера — не работает**

- проблема потери записей WAL, не попавших на реплику из-за задержки

## **Восстановление «с нуля» из резервной копии**

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



The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on the middle section of the image is a semi-transparent blue band. Within this band, a network diagram is visible, consisting of numerous small blue dots connected by thin, light-blue lines, creating a web-like pattern. Centered within this band is the main title text.

# Логическая репликация



# Логическая репликация

- поставщик-подписчик: поток данных возможен в обе стороны
- информация о строках (уровень журнала logical)
- требуется совместимость на уровне протокола
- возможна выборочная репликация отдельных таблиц



# Логическая репликация

- Встроенная логическая репликация доступна с 10 версии PostgreSQL. Для более ранних версий аналогичный функционал доступен в расширении **pg\_logical**.
- Для передачи логических изменений (на уровне строк) используется протокол репликации. Для работы такой репликации требуется установка уровня журнала **logical**.
- Другой способ организации логической репликации состоит в использовании триггеров для перехвата изменений, помещения этой информации в очередь событий и передача ее на другой сервер. Такой способ, однако, менее эффективен, и уходит в прошлое (**Slony-I**).

# Отличия от физической репликации

- По сути тут нет понятий Мастер и Реплика
- На сервере создается публикация, на которую другие серверы могут подписываться
- Подписчику же передается информация об изменениях строк в таблицах: двоичная совместимость не требуется.

# Логическая репликация

## Публикующий сервер

- выдает изменения данных построчно в порядке их фиксации (реплицируются команды INSERT, UPDATE, DELETE), в 11 версии добавили TRUNCATE
- возможна начальная синхронизация
- всегда используется слот логической репликации
- DDL не передаются, то есть таблицы-приемники на стороне подписчика надо создавать вручную
- применение изменений происходит без выполнения команд SQL и связанных с этим накладных расходов на разбор и планирование, что уменьшает нагрузку на подписчика
- параметр **wal\_level = logical**

# Логическая репликация

## Подписчики

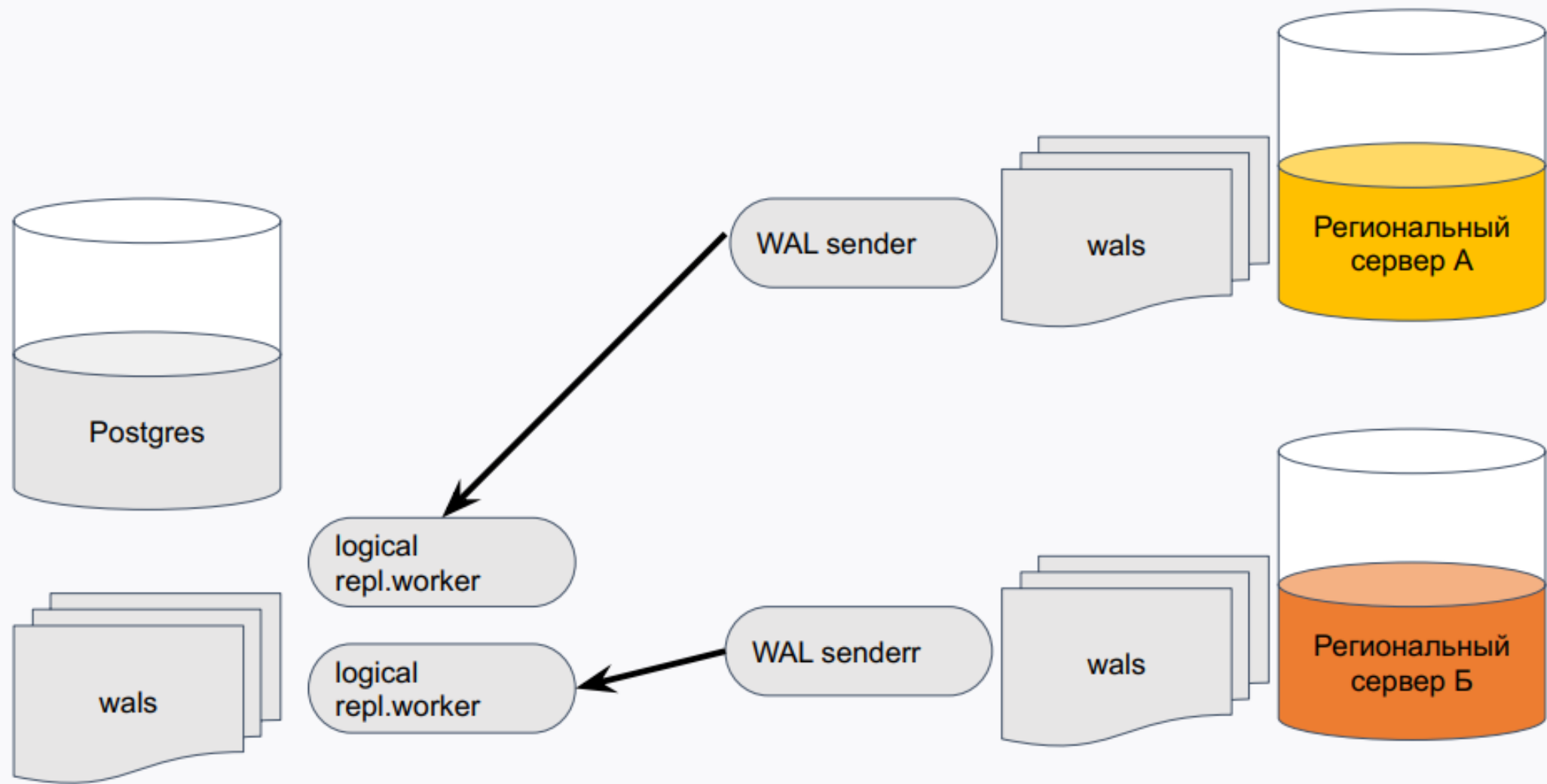
- получают и применяют изменения
- без разбора, трансформаций и планирования — сразу выполнение
- возможны конфликты с локальными данными



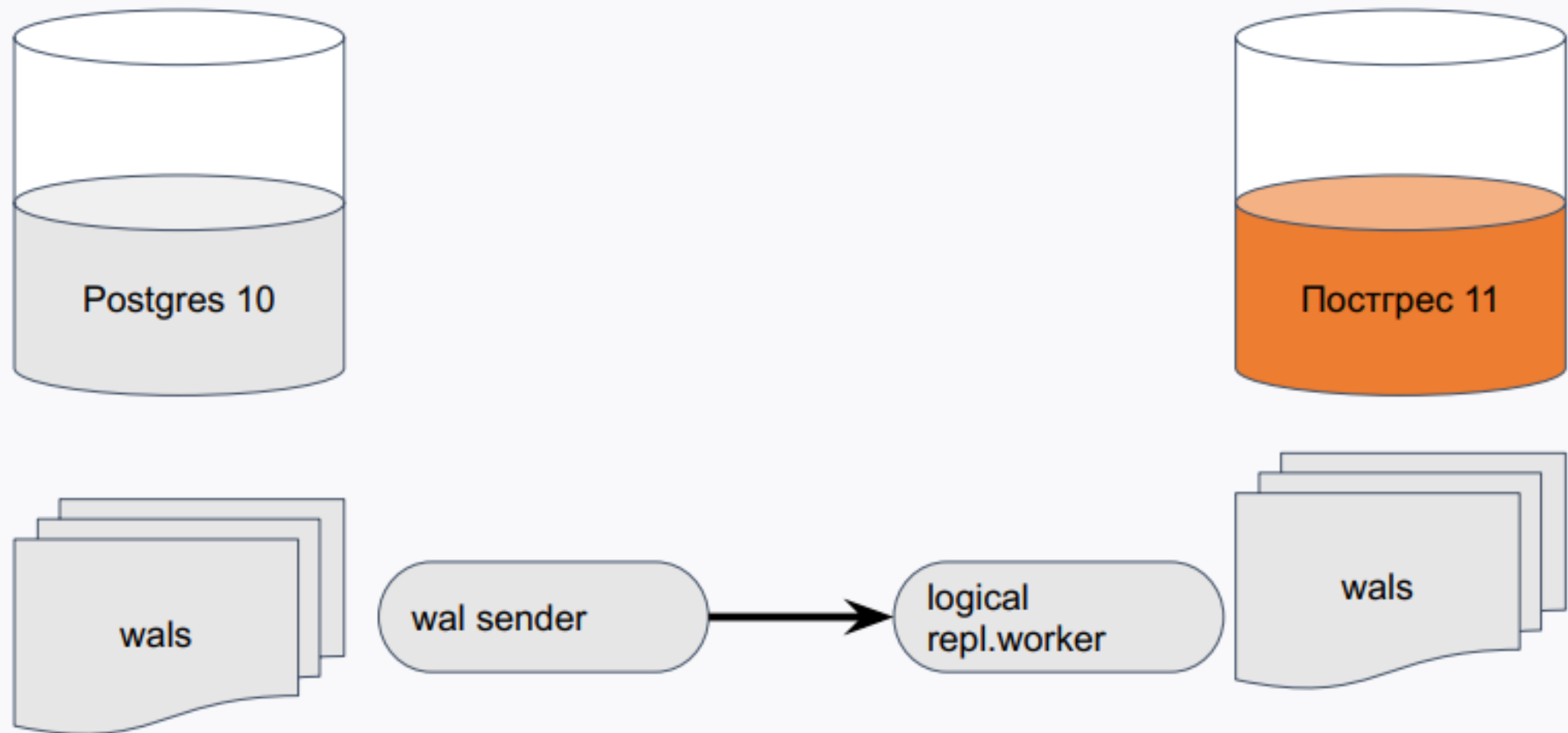
# Назначение логической репликации

- Консолидация и общие справочники
- Обновление основной версии сервера
- Мастер-мастер (в будущем)

# Объединение данных



# Обновление версий



## В новых версиях

С 15 версии можно фильтровать записи и колонки.

С 16 версии стало возможным логическую репликацию настроить таким образом, что можно подписаться на публикацию на физической реплике.



# Логическая репликация - ПРАКТИКА

Используем два сервера и настроим логическую репликацию. Для этого нам понадобится дополнительная информация в журнале.

```
ALTER SYSTEM SET wal_level = logical;
```

Рестартуем кластер

```
$ sudo pg_ctlcluster 14 main restart
```

На первом сервере создаем публикацию:

```
\c db_name
```

```
CREATE TABLE test(i int);
```

```
CREATE PUBLICATION test_pub FOR TABLE test;
```

```
\dRp+
```

```
\password
```

# Логическая репликация - ПРАКТИКА

создадим подписку на втором экземпляре

**\c db\_name**

**CREATE TABLE test(i int);**

**CREATE SUBSCRIPTION test\_sub  
CONNECTION 'host=localhost port=5432 user=postgres  
password=test dbname=replica' PUBLICATION test\_pub WITH  
(copy\_data = false);**

**\dRs**

# Логическая репликация - ПРАКТИКА

состояние подписки

```
SELECT * FROM pg_stat_subscription \gx
```

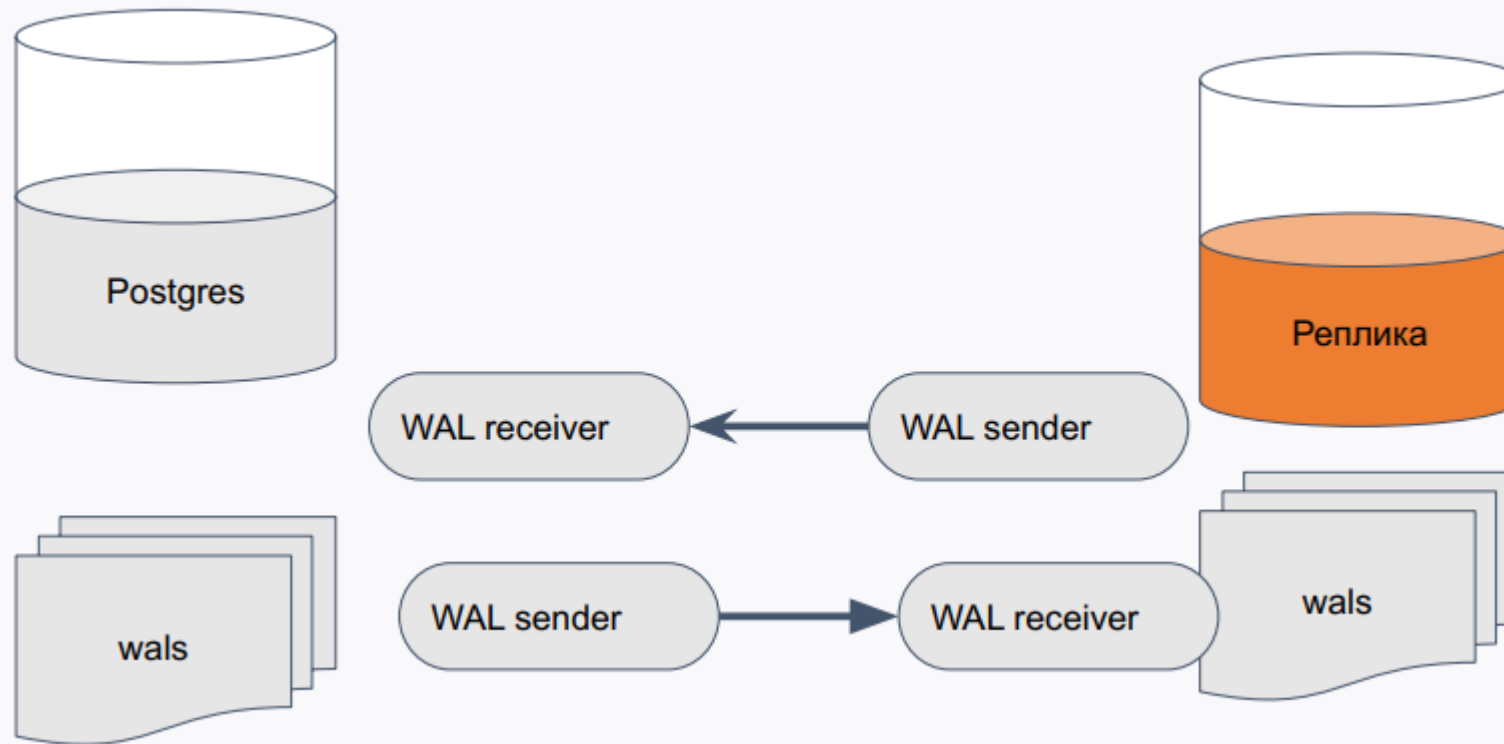
удаление публикации и подписки

```
drop publication test_pub;  
drop subscription test_sub;
```

найти проблему можно в логах

```
$ tail /var/log/postgresql/postgresql-14-main2.log
```

# Мастер мастер репликация



- Любой сервер - полная копия доступная для записи
- Высока вероятность конфликтов
- Сложность в администрировании
- Сложность восстановления



# Мастер мастер репликация

В ванильном PostgreSQL нельзя одновременно создать на таблицу подписку и публикацию на одном инстансе и зеркально сделать публикацию с подпиской на другом инстансе.

Запрещено создание подписки, порождающей цикл.



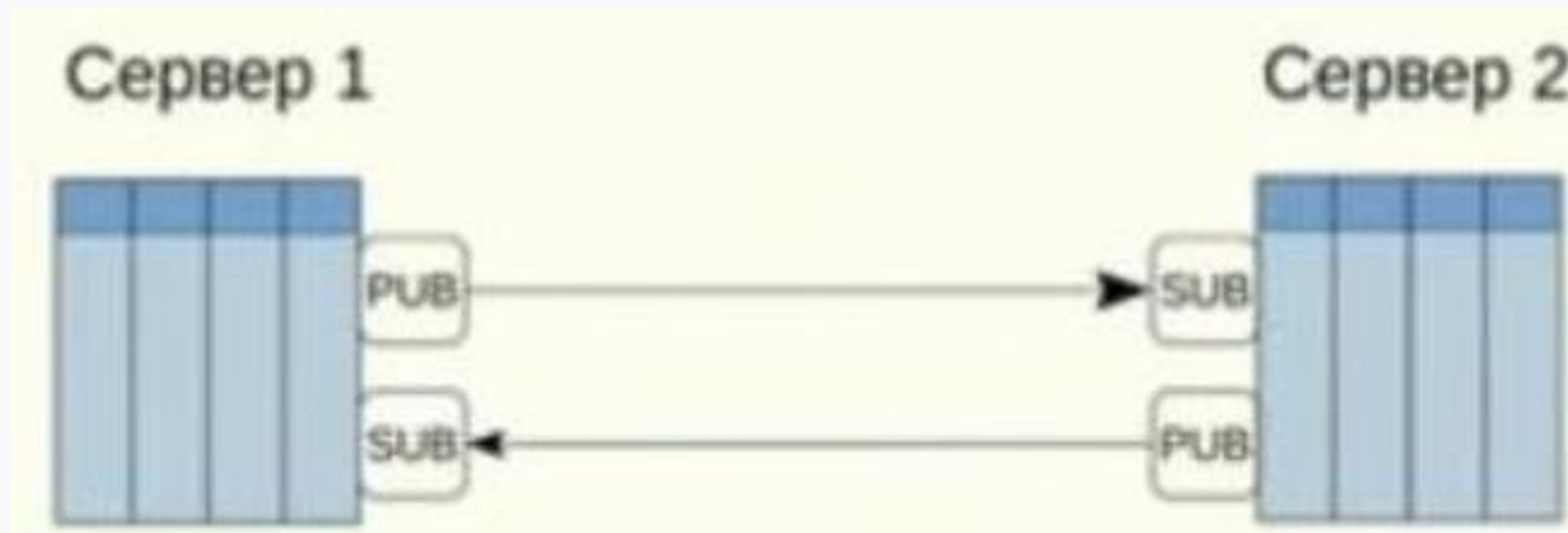
# Мастер мастер репликация

В ванильном PostgreSQL нельзя **БЫЛО ДО** 16-й версии.

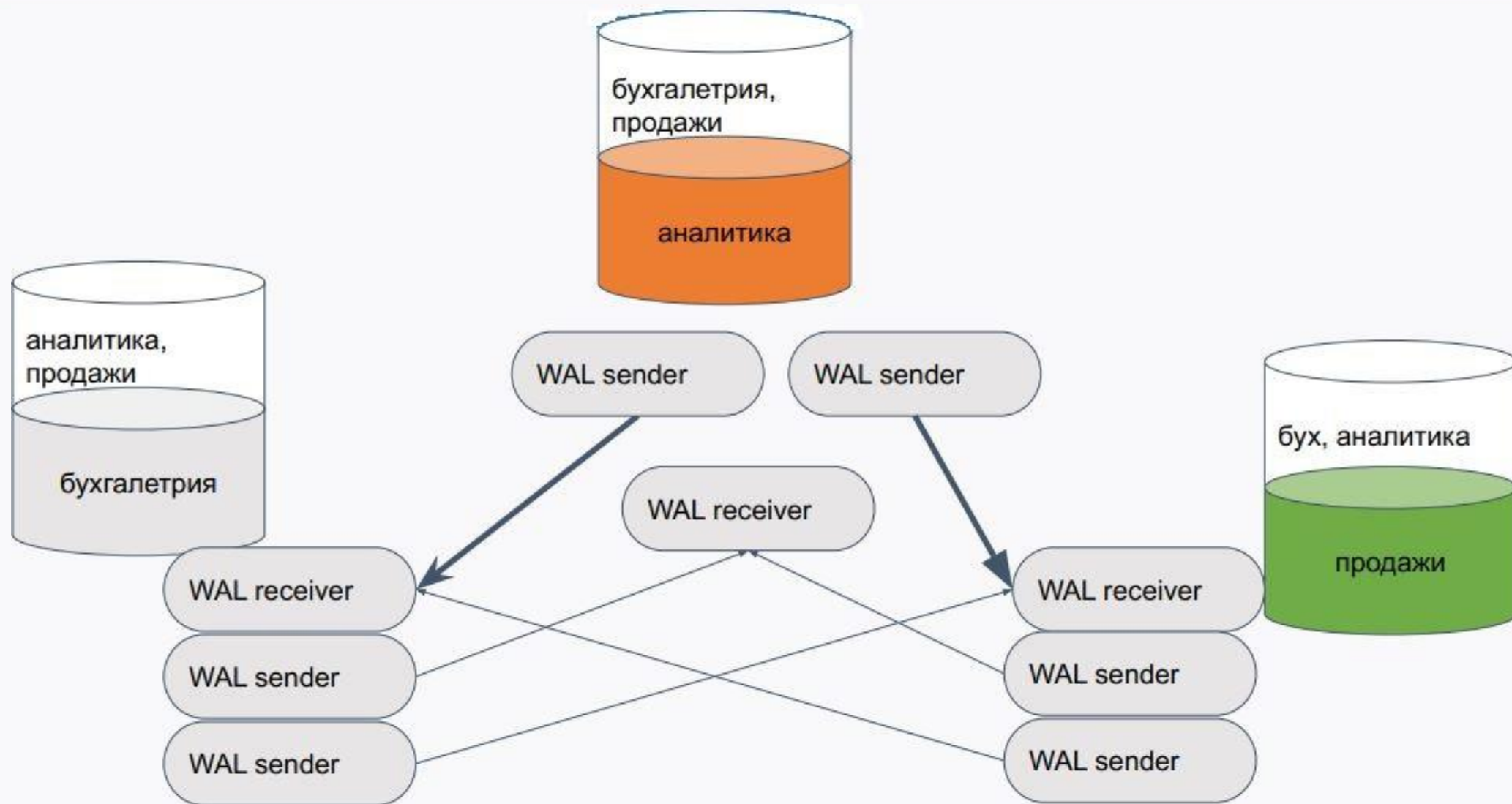
Теперь для создания этого достаточно указать опцию  
origin = none.

```
CREATE SUBSCRIPTION test_sub
```

```
CONNECTION 'host=localhost port=5432 user=postgres password=test  
dbname=replica' PUBLICATION test_pub WITH (copy_data = false, origin =  
none);
```



# Мастер мастер репликация: Percona eXtraDB Cluster



# Мастер мастер репликация

- 1) MultiMaster от компании Postgres Pro  
(расширение для СУБД Postgres Pro Enterprise)
- 2) Postgres Plus xDB Replication Server with MultiMaster Replication (MMR) от компании EnterpriseDB
- 3) Компания 2ndQuadrant (сейчас уже часть компании EnterpriseDB)
- 4) Bucardo
- ...



# Краткий итог

**Механизм репликации основан на передаче журнальных записей на реплику и их применении**

- трансляция потока записей или файлов WAL

**Физическая репликация создает точную копию всего кластера**

- однонаправленная
- требует двоичной совместимости

**Логическая репликация передает изменения строк отдельных таблиц**

- разнонаправленная
- совместимость на уровне протокола



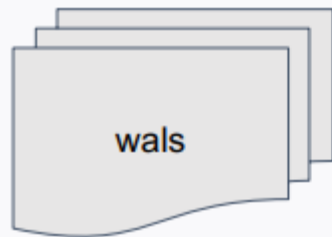
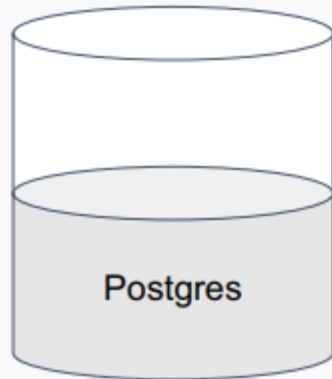
The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent network pattern consisting of numerous small dots connected by thin, light-blue lines, creating a web-like structure across the center of the slide.

# Примеры применения



# Горячий резерв for high availability

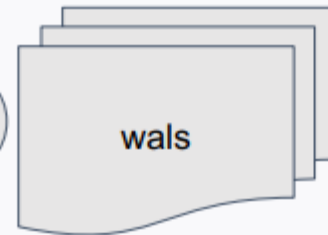
`synchronous_commit = on`



WAL sender



WAL receiver



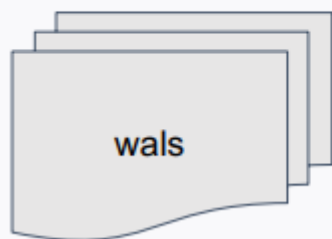
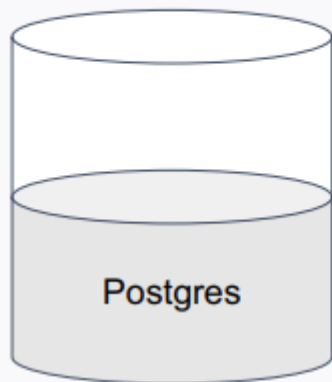
`hot_standby_feedback = off`

`max_standby_streaming_delay=t`

- синхронная репликация, реплика должна максимально соответствовать мастеру
- запросы к реплике возможны, но не приоритетны

# Балансировка OLTP

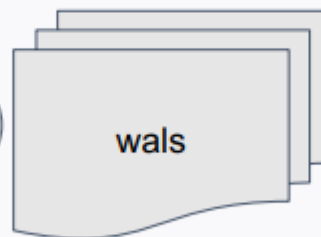
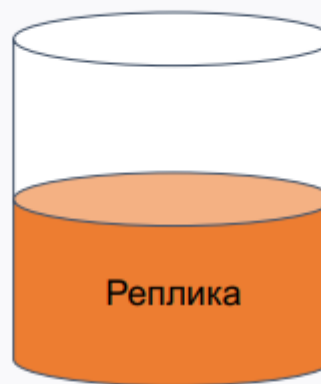
`synchronous_commit = off`



WAL sender



WAL receiver



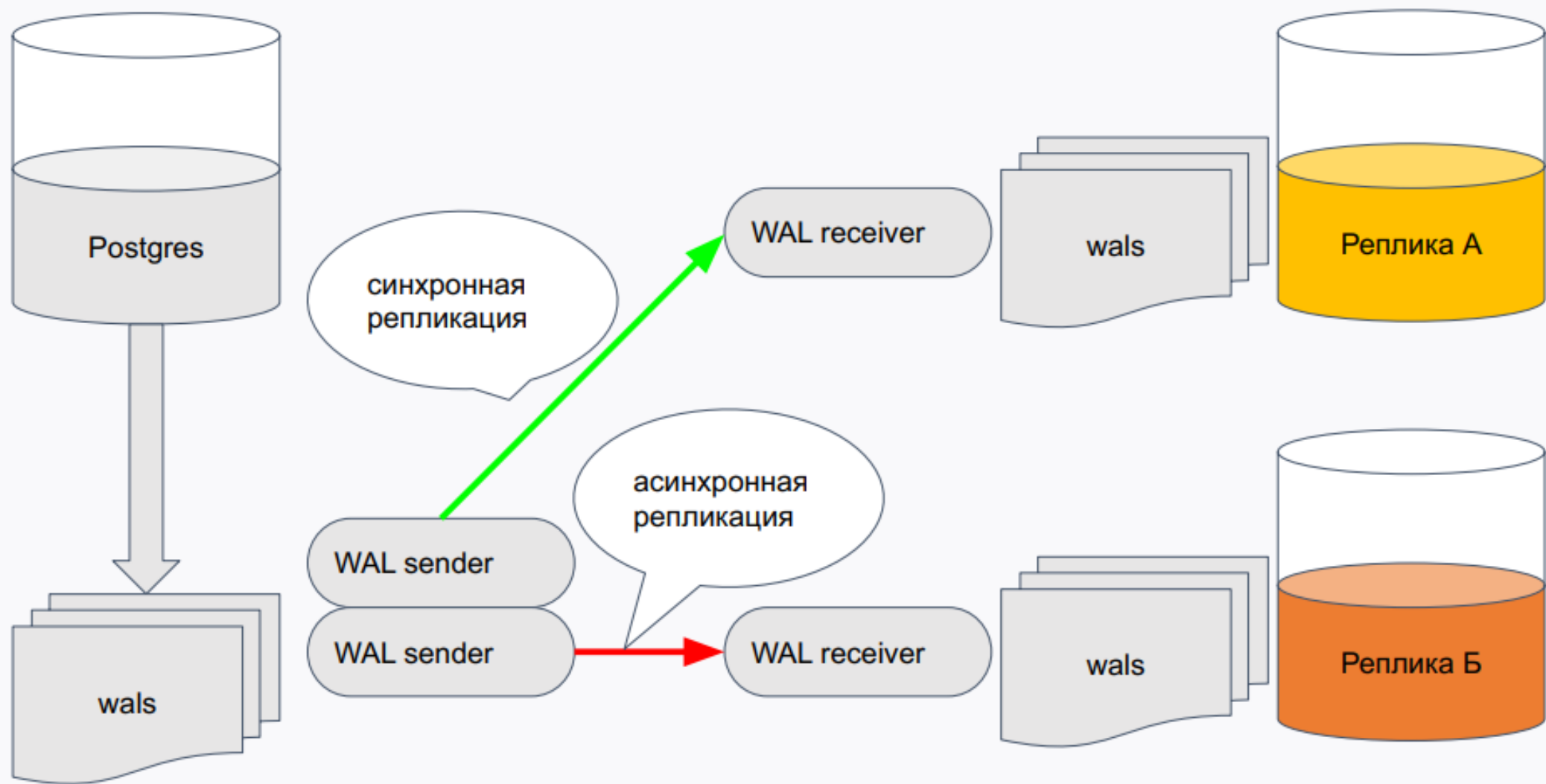
`hot_standby_feedback = on`

`max_standby_streaming_delay=T`

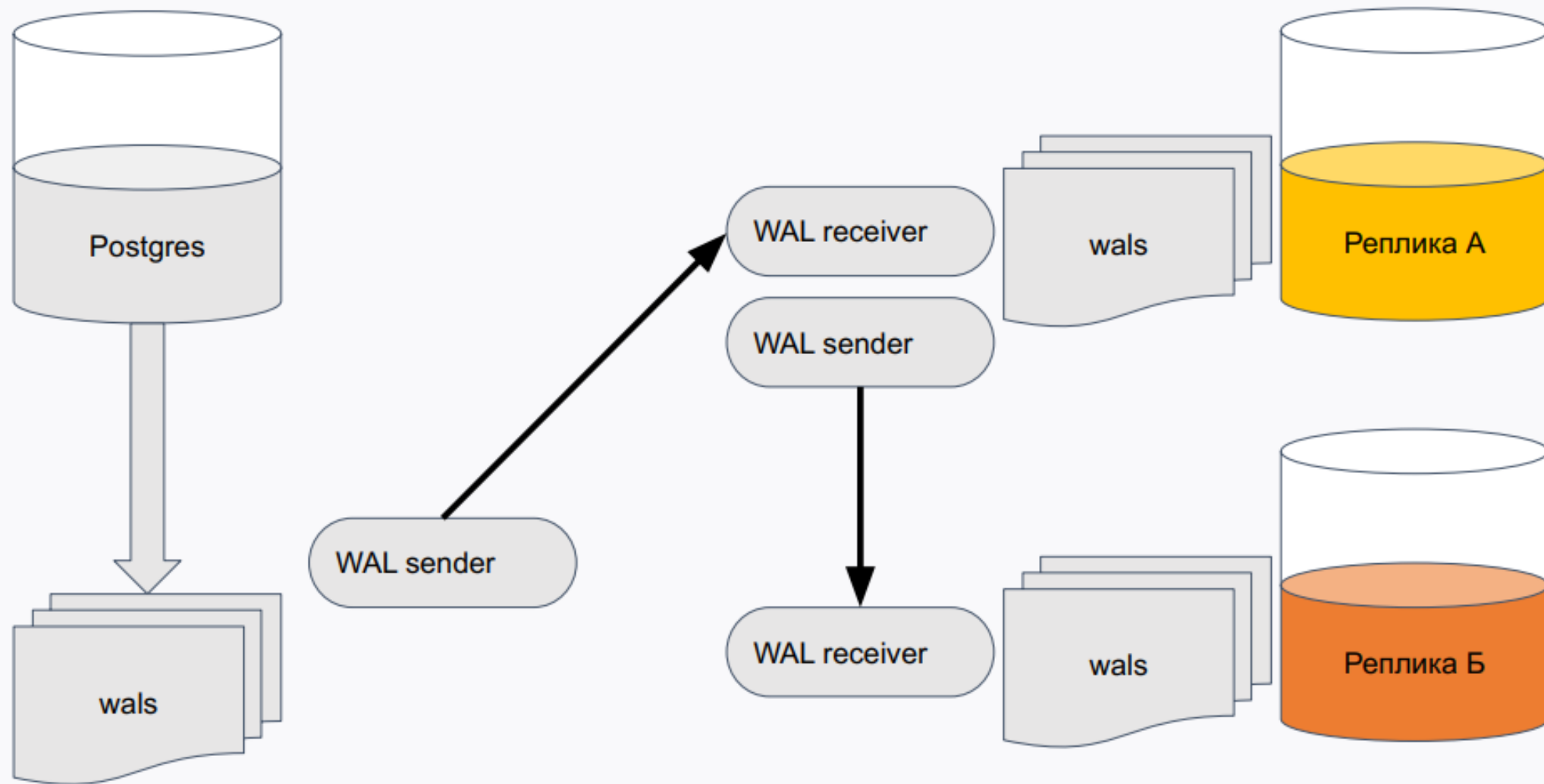
- много коротких запросов
- запросы на реплике должны обрабатываться
- долгие запросы повлияют на мастер, но в идеале их не должно быть



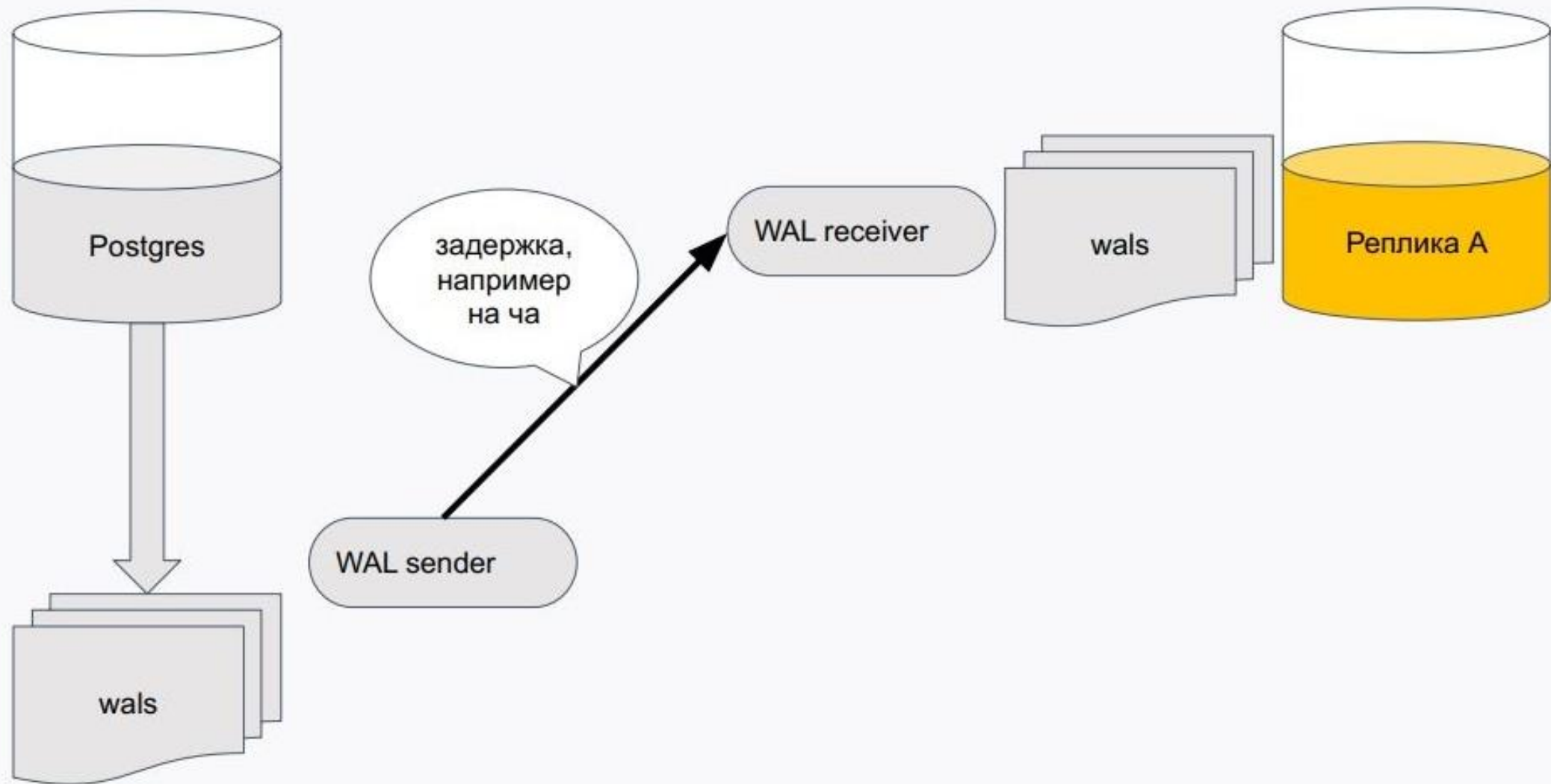
# Горизонтальное масштабирование с НА



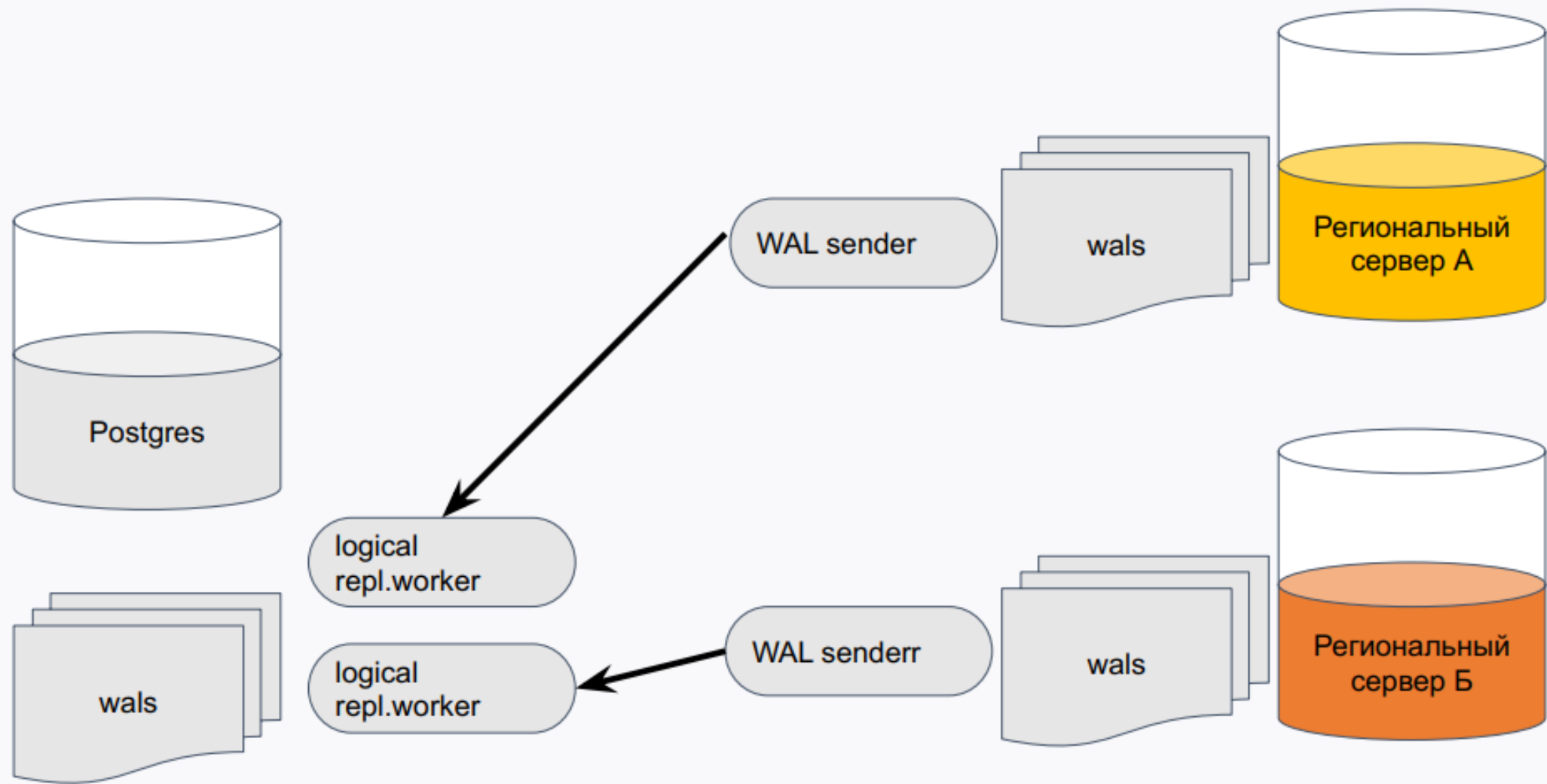
# Каскадная репликация



# Time machine

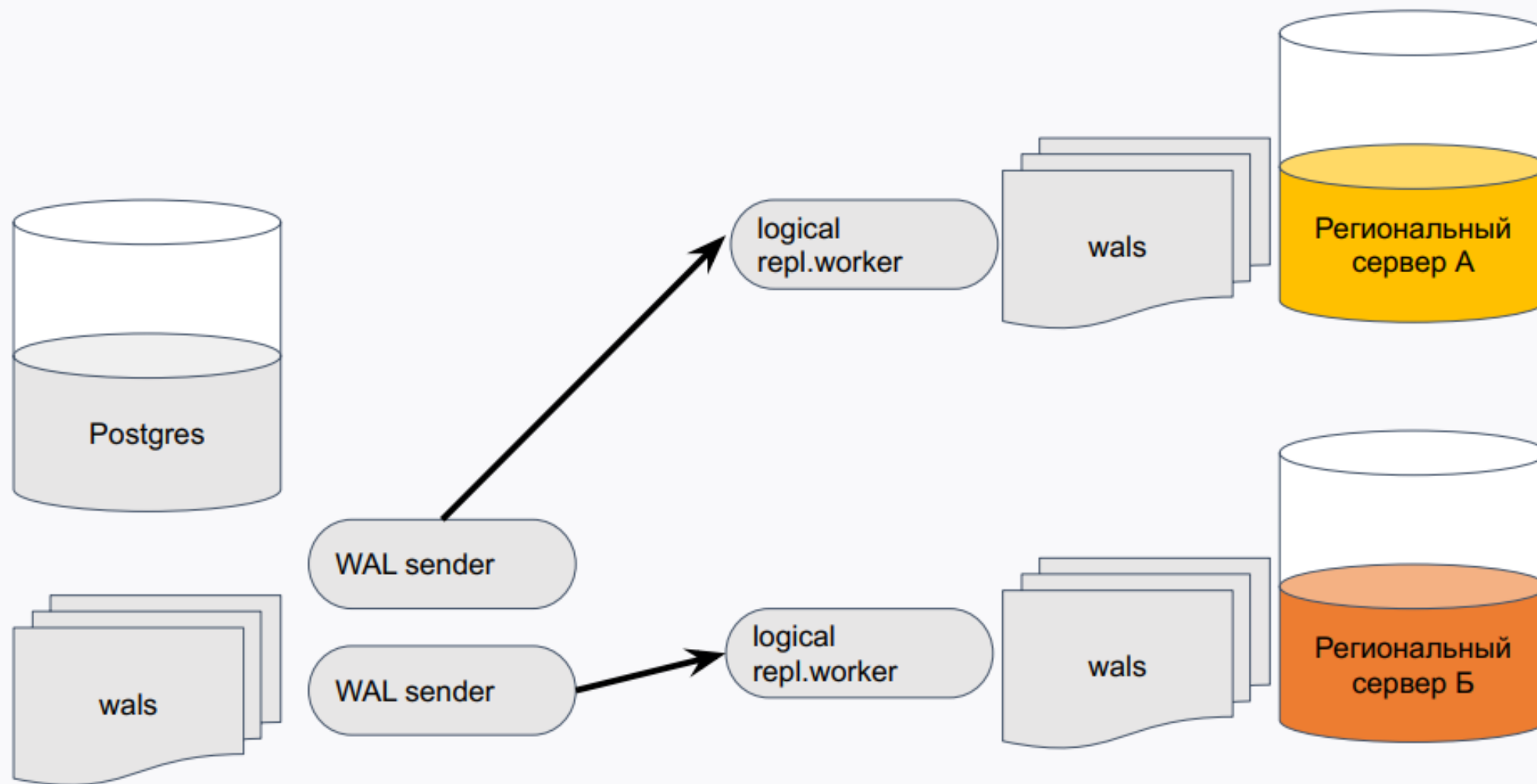


# Объединение данных

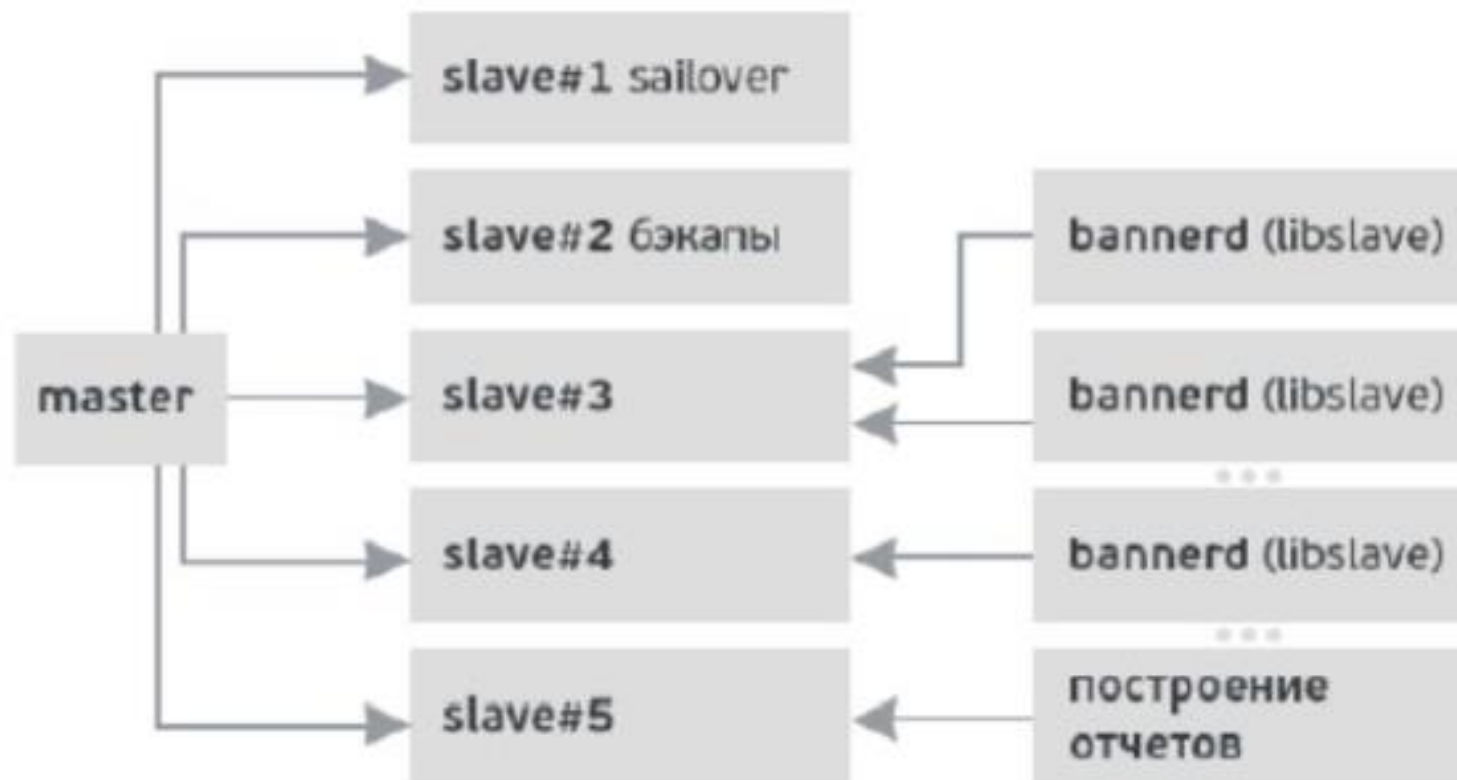




# Рассылка справочников



# Как в мейл.ру



\*Структура проекта Mail.Ru Target

# Рефлексия



1. Какие варианты репликации вы запомнили ?
2. Какая модель репликации больше понравилась ?
3. В чем разница между синхронной и асинхронной репликацией ?

# ДЗ

На 1 VM создаем таблицы test для записи, test2 для запросов на чтение. Создаем публикацию таблицы test и подписываемся на публикацию таблицы test2 с VM №2.

На 2 VM создаем таблицы test2 для записи, test для запросов на чтение. Создаем публикацию таблицы test2 и подписываемся на публикацию таблицы test1 с VM №1.

3 VM использовать как реплику для чтения и бэкапов (подписаться на таблицы из VM №1 и №2 ).  
Небольшое описание, того, что получилось.

\* реализовать горячее реплицирование для высокой доступности на 4VM. Источником должна выступать VM №3. Написать с какими проблемами столкнулись.



The background of the entire image is an aerial photograph of a city with many skyscrapers, overlaid with a semi-transparent blue layer. A network of thin, light-blue lines connects various points across the blue area, creating a digital or technological aesthetic.

Заполните, пожалуйста,  
опрос о занятии по ссылке в чате



Спасибо за внимание!  
Приходите на следующие вебинары



Коробков Виктор