OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

SQL и реляционные СУБД PostgreSQL в облаках

Аристов Евгений

telegram @AEugene

https://aristov.tech

Преподаватель

Евгений Аристов

Более 20 лет занимаюсь разработкой ПО на Java/Spring, C#, PHP.
Архитектор баз данных PostgreSQL, MongoDB, MSSQL, Oracle, MySQL, MariaDB, Couchbase и др.

Деплой БД как on-premise Google Cloud Platform, AWS, Azure, Yandex Cloud, так и Kubernetes, DBaaS, MultiCloud.

За это время было спроектировано и разработано более ста проектов для сетей магазинов, фитнес-центров, отелей, финансовом секторе.

Огромный опыт построения и эксплуатации систем виртуализации VmWare ESXi, Hyper-V.

Автор книг по PostgreSQL. Новинка "PostgreSQL 14. Оптимизация, Kubernetes, кластера, облака."

Правила вебинара

Активно участвуем

Задаем вопрос в чат

Вопросы вижу в чате, могу ответить не сразу

Если остались вопросы/офтопик в канал Slack

Маршрут вебинара

Цели вебинара | После занятия вы сможете

1 Представлять сложность выбора той или иной технологии

2 Примерно понимать как работать с облачными технологиями

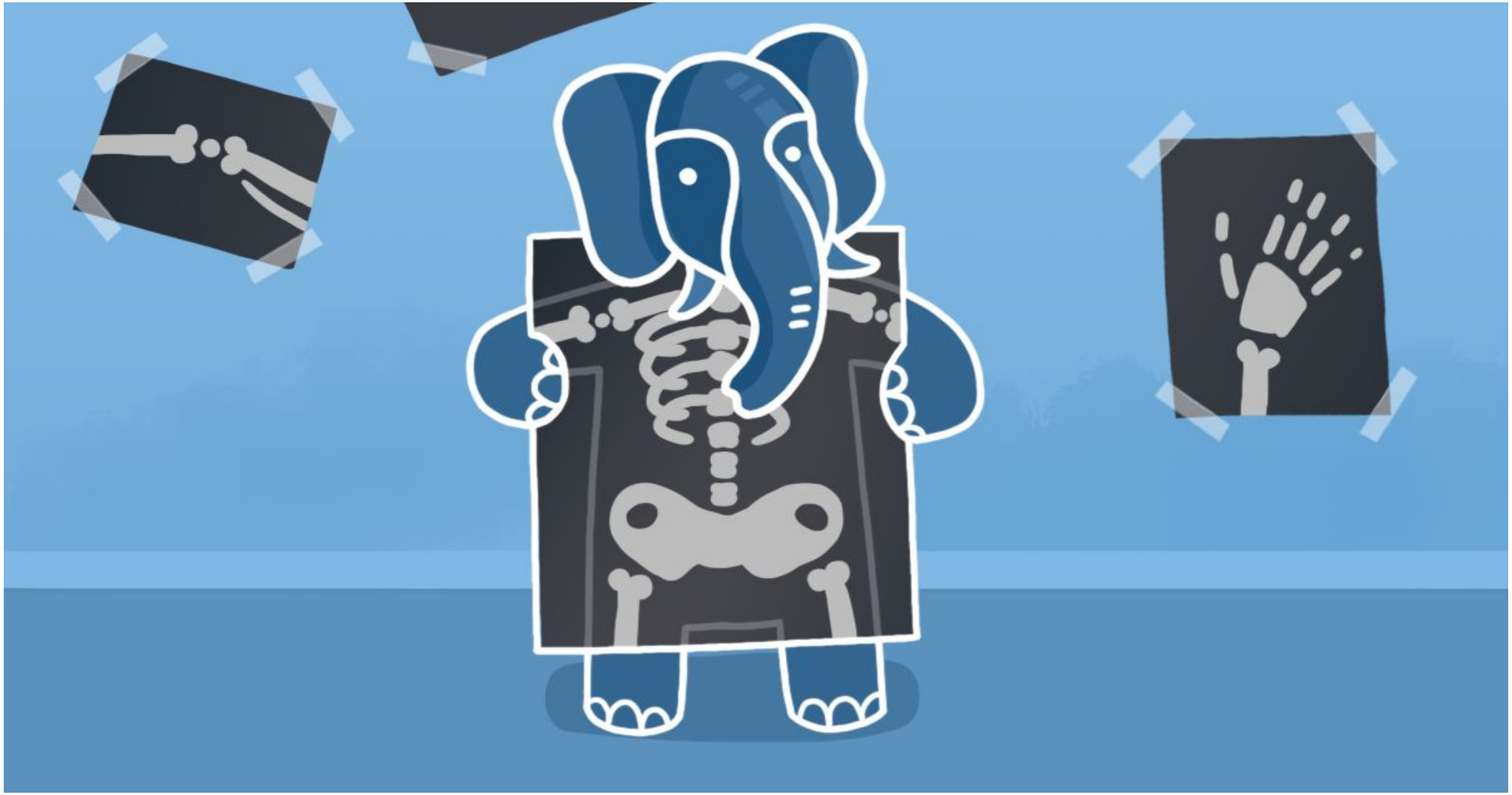
3 Рассказать про ACID и транзакции

Смысл | Зачем вам это уметь, в результате:

1 Осознанно развивать востребованные навыки

PostgreSQL

Постгрес



Проблемы

Проблемы

Что у нас есть на данный момент в большинстве проектов.
90% что был такой сценарий:

- У нас есть специалисты по Java/Ruby/etc
- ORM работает из коробки
- DBA нет или нет на это времени
- Постгрес выбрали, так как самый популярный, ну и не Оракл же
- Пока объемы были небольшие, все работало как часы
- Когда проект стал высоконагруженным начались проблемы:
  - коннектов не хватает
  - памяти не хватает
  - tps уперся в потолок в пик нагрузок

Стандартный путь решения

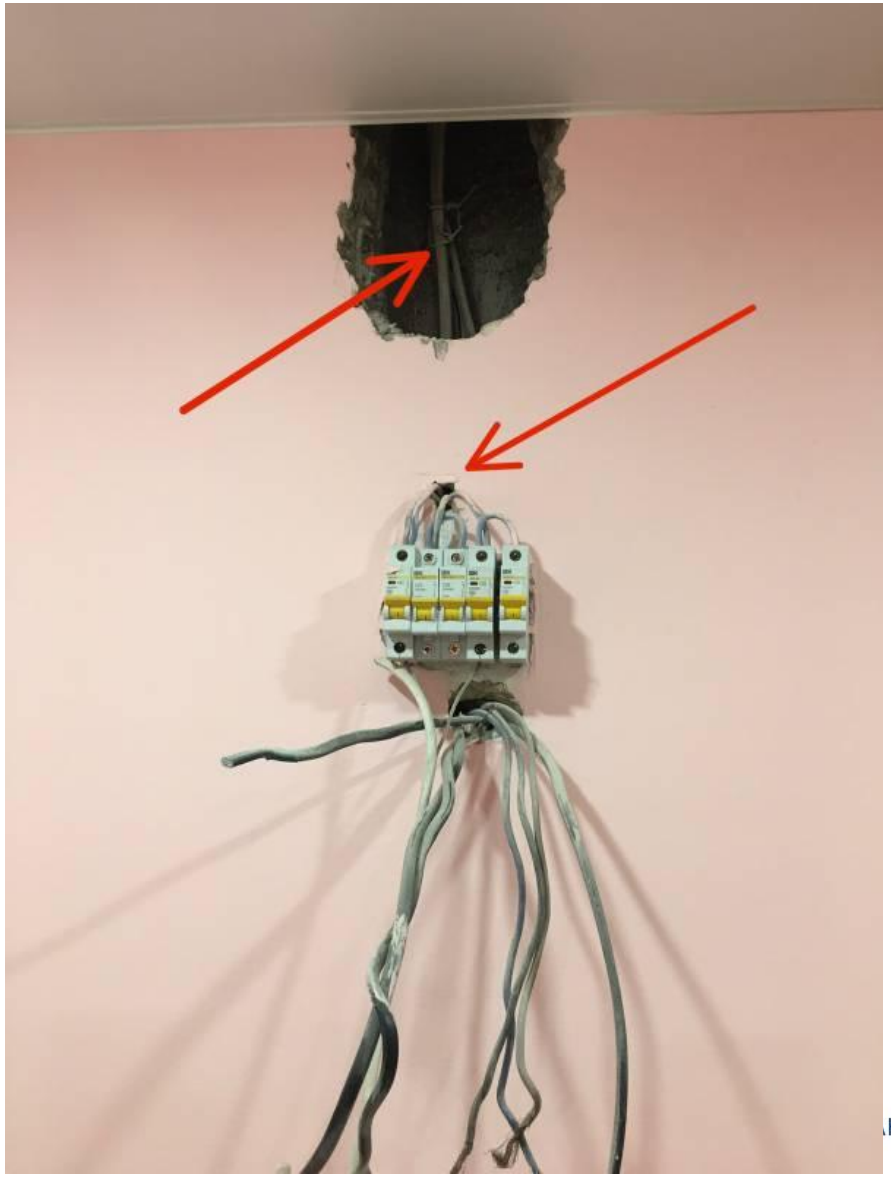
Увеличить мощность инстансов

Минусы:

- пределы всегда есть
- не всегда это оптимально
- стоимость увеличения инстансов в определенный момент превышает стоимость привлечения стороннего DBA

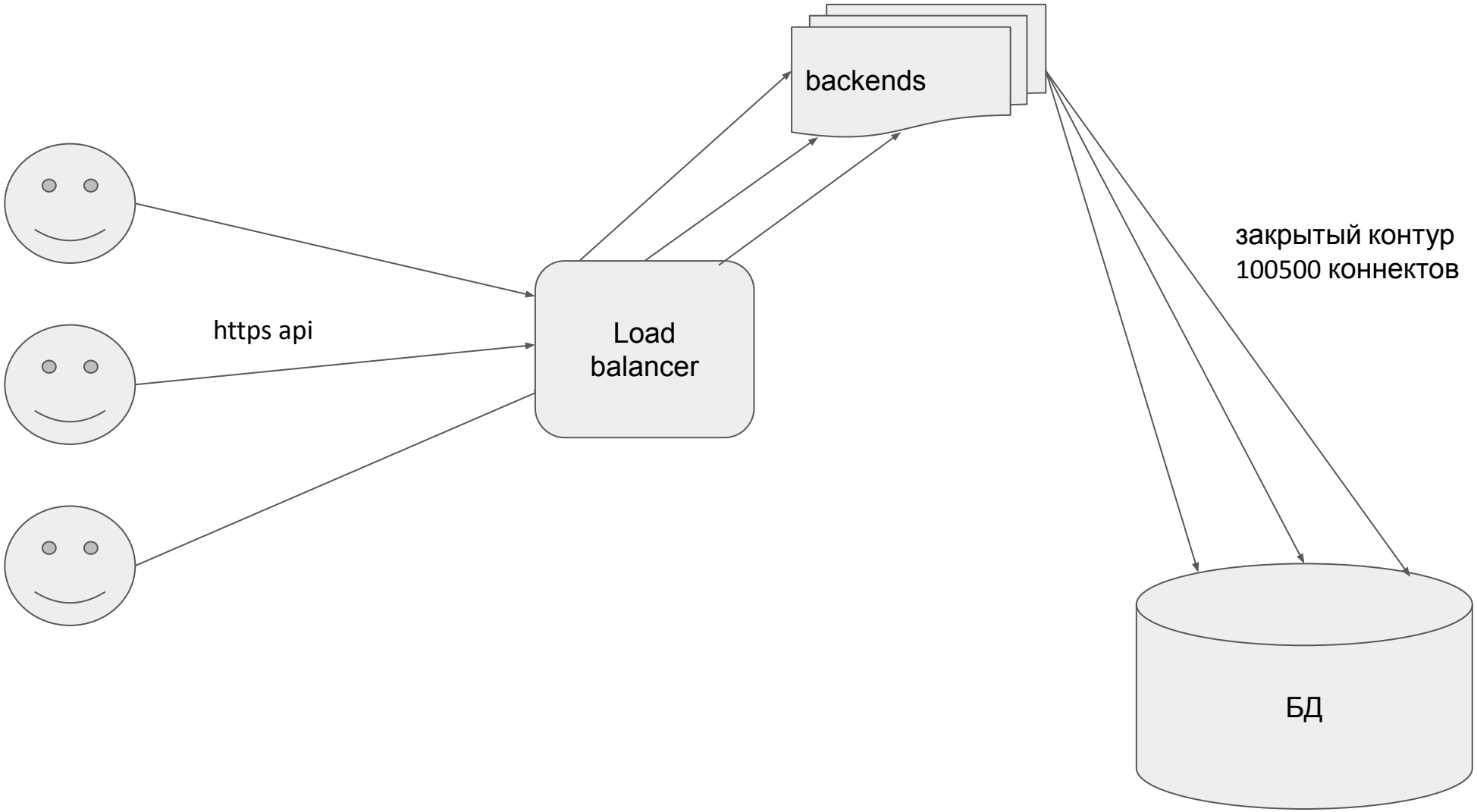
Стандартный путь решения

Иногда пытаются решить своими силами или нанимают первого попавшегося ДБА Получается вот так:

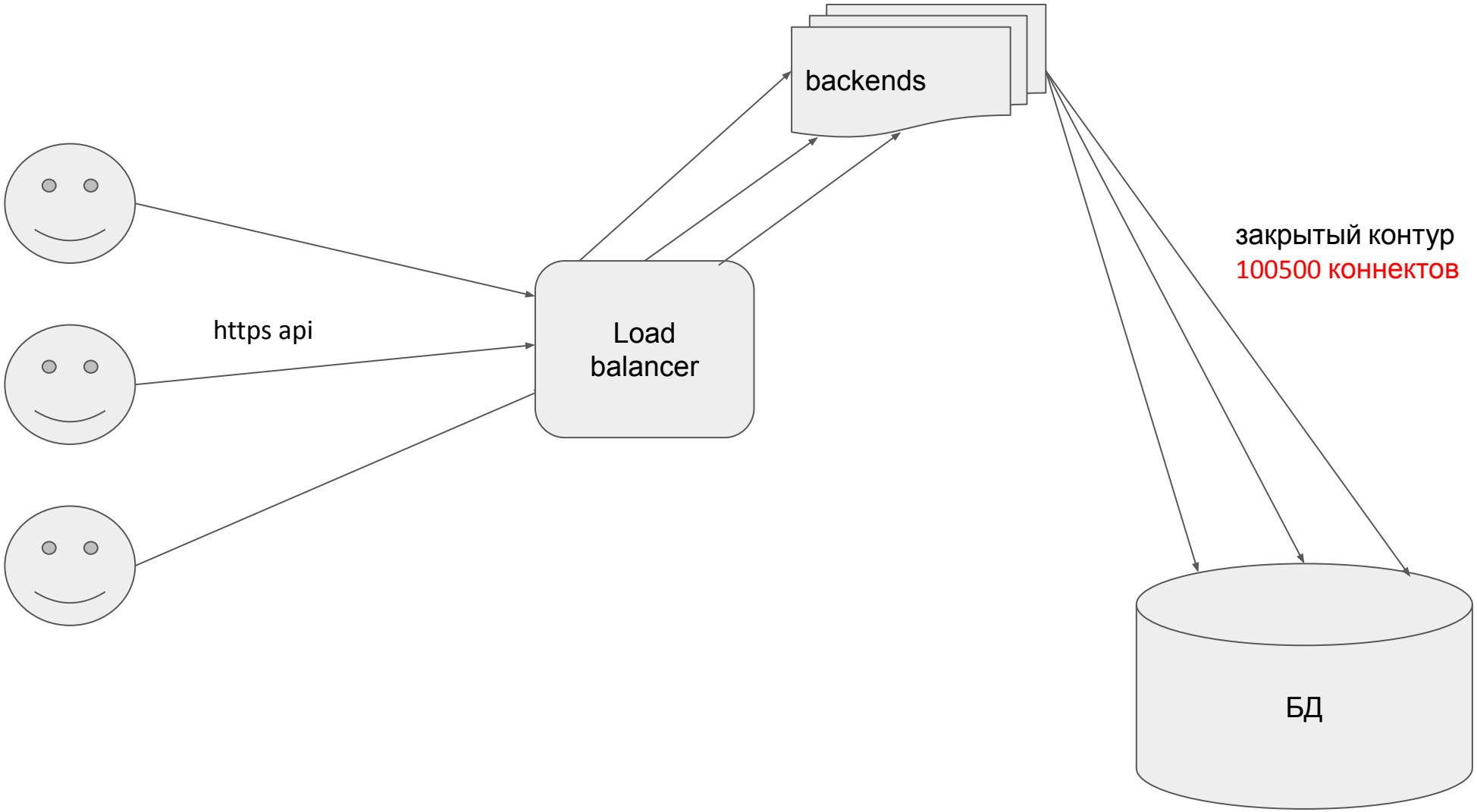


Текущая конфигурация

Текущая конфигурация

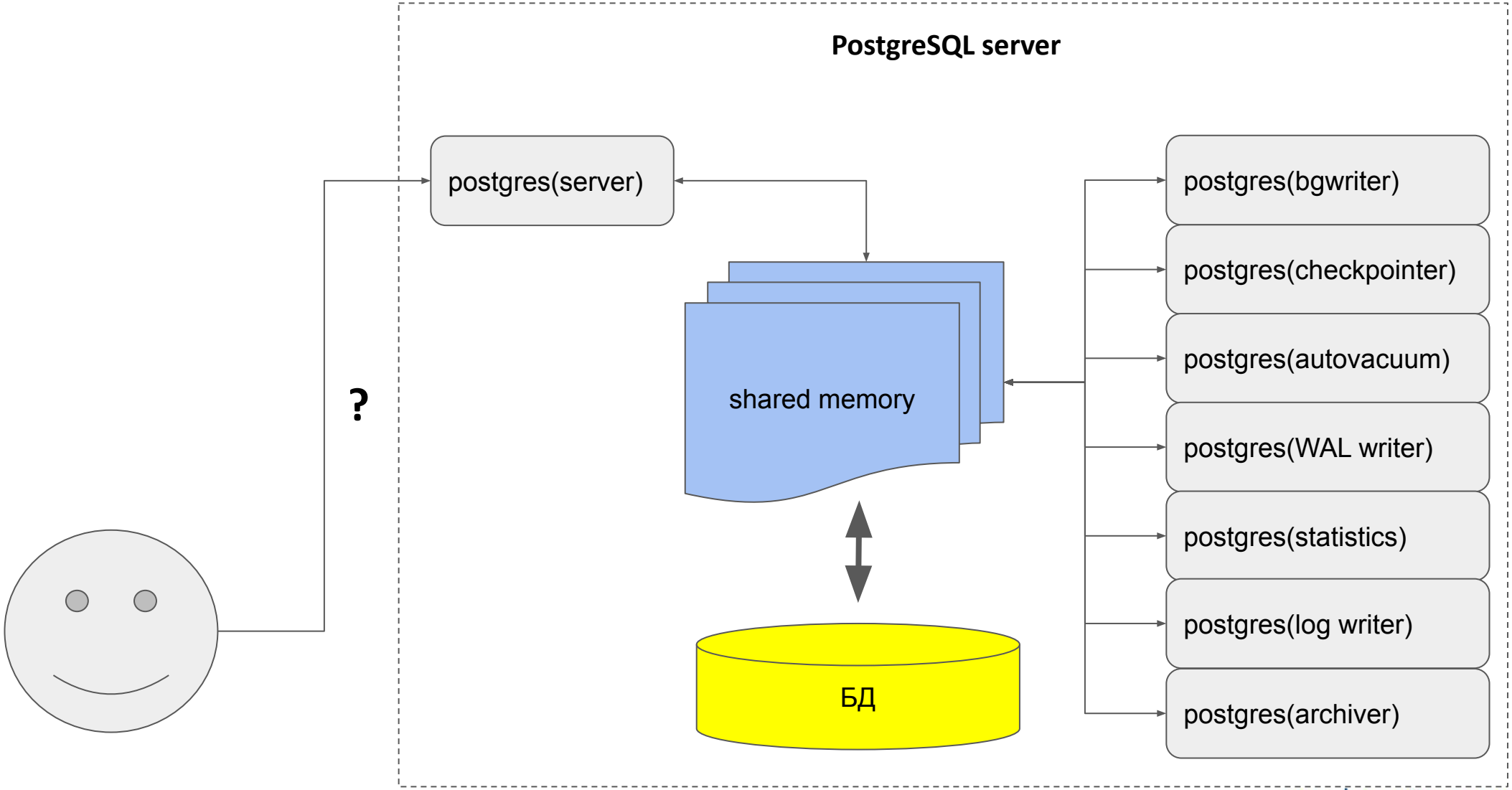


Чем плохо много коннектов?

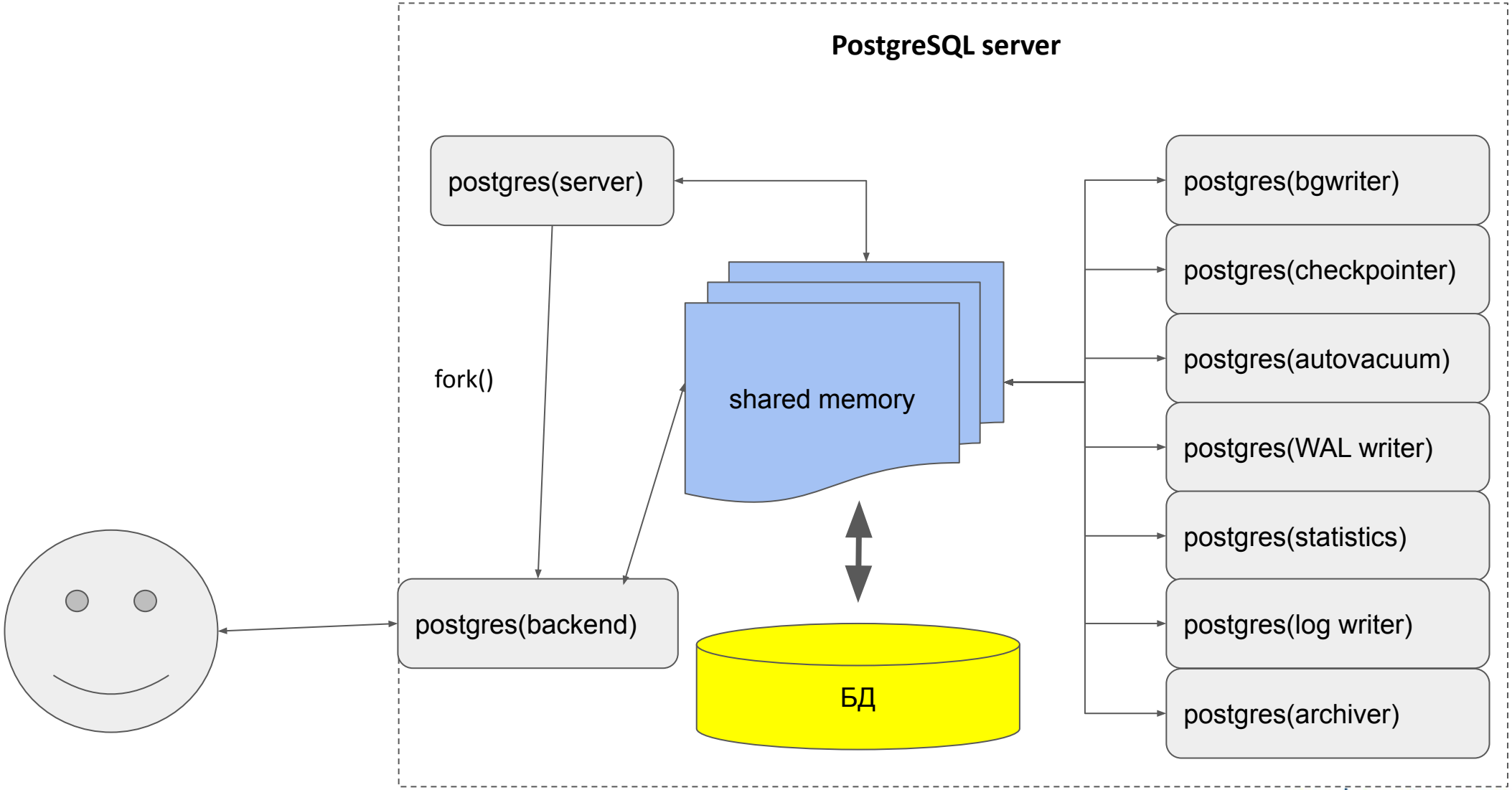


Серверные процессы и память

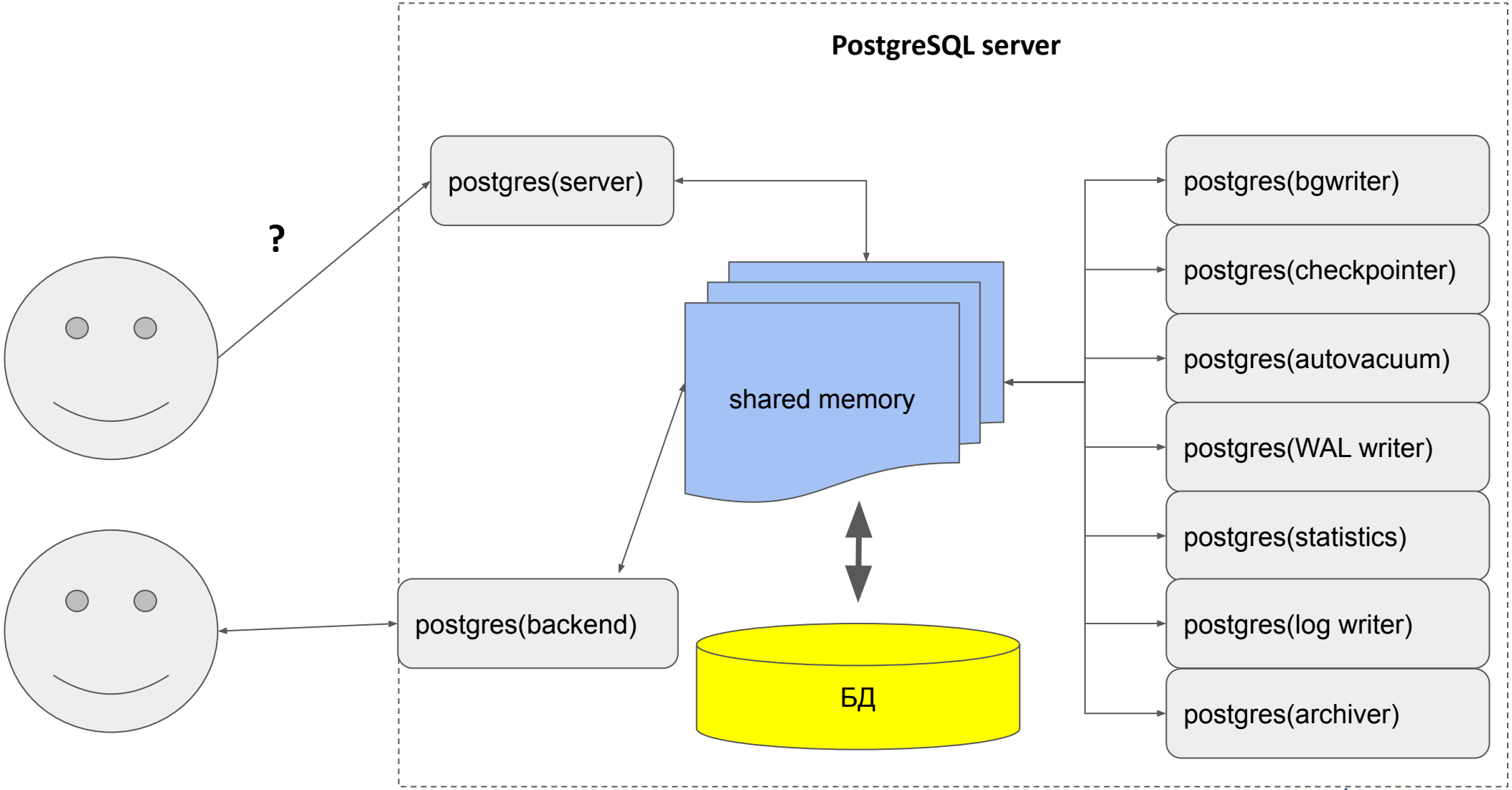
Серверные процессы и память



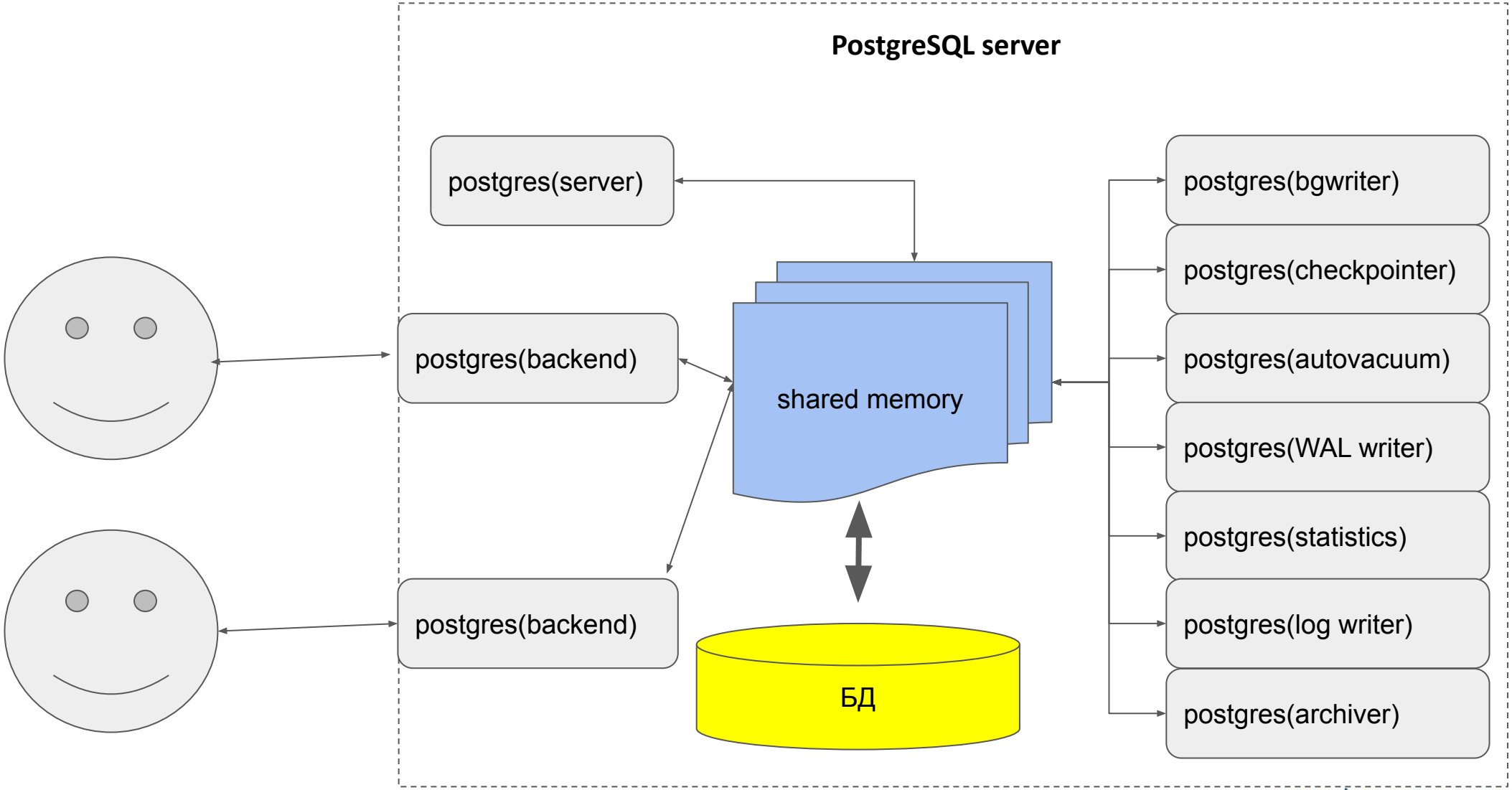
Серверные процессы и память



Серверные процессы и память



Серверные процессы и память



Кроме это выделяется память для каждой сессии

- принадлежит backend процессу
- **work_mem (4 MB)**

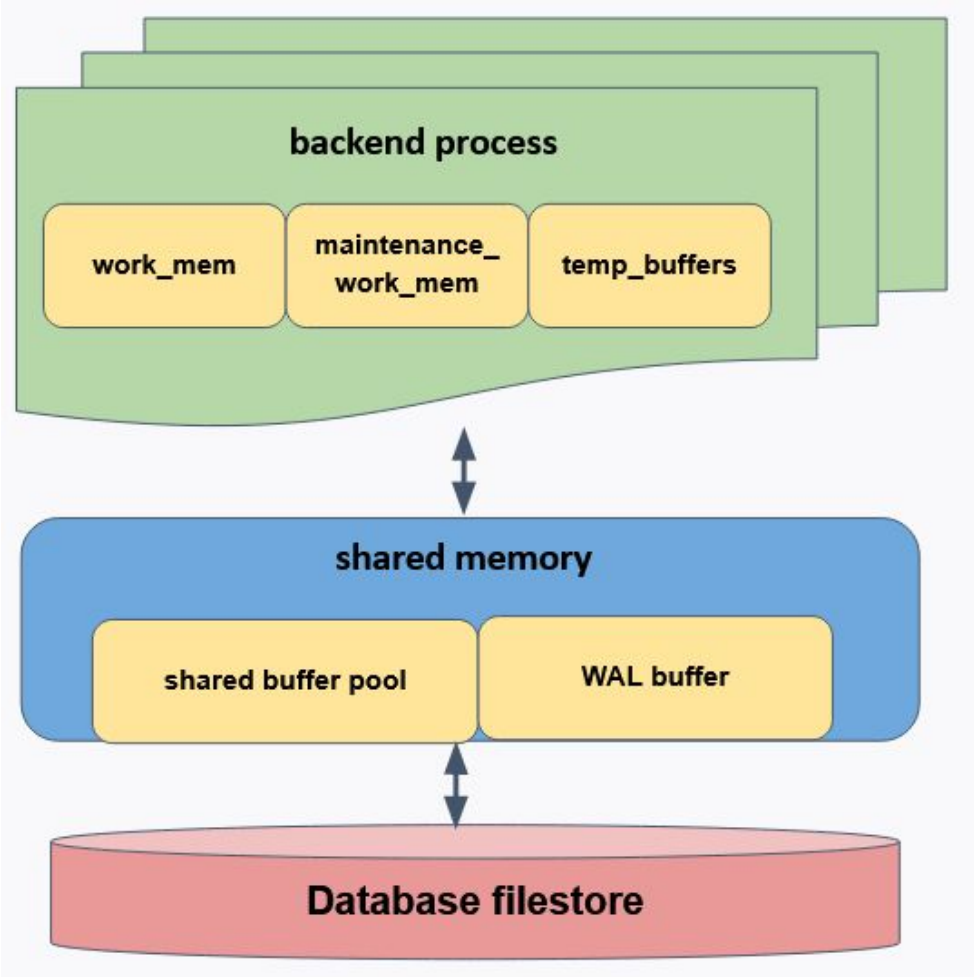
  эта память используется на этапе выполнения запроса для сортировок строк, например ORDER BY и DISTINCT
- **maintenance_work_mem (64MB)**

  используется служебными операциями типа VACUUM и REINDEX

  выделяется только при использовании команд обслуживания в сессии
- **temp_buffers (8 MB)**

используется на этапе выполнения для хранения временных таблиц

http://www.interdb.jp/pg/pgsql02.html



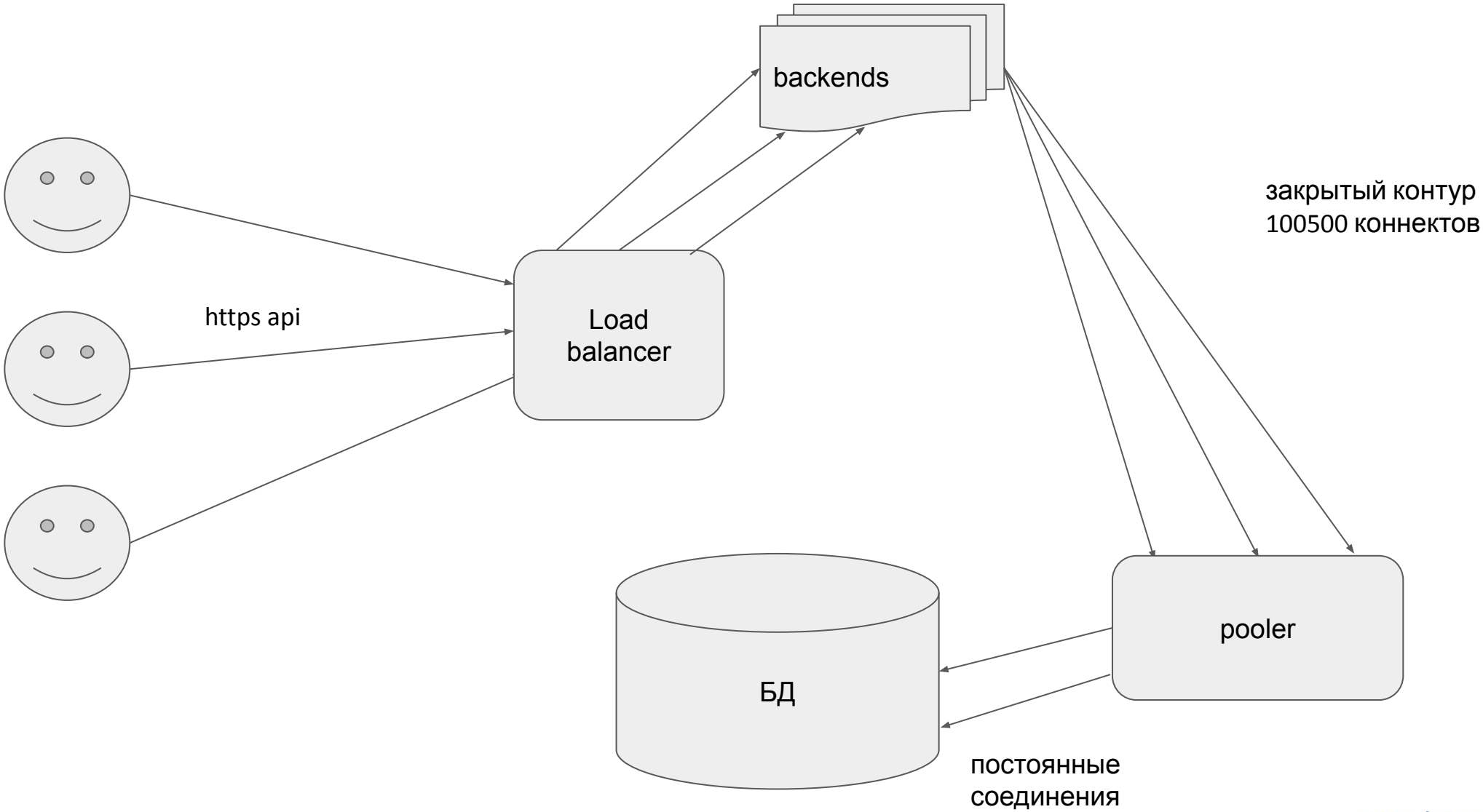
Что не так с настройками?

Памяти у инстанса 8 Gb (периодически приходил OOM killer)

```
max_connections = 1000          # (change requires restart)
shared_buffers = 16GB           # min 128kB
work_mem = 16MB                 # min 64kB
maintenance_work_mem = 256MB    # min 1MB
```

Пулеры

Добавляем пул коннектор



Классические варианты

Pgpool II

pg_bouncer + haproxy

Odyssey

ну или использовать готовые кластера

Классические варианты

Pgpool II

- используется встроенный механизм репликации
- есть пул соединений
- балансировщик нагрузки
- высокая доступность (наблюдатель с виртуальным IP, автоматическое переключение мастера)

https://severalnines.com/database-blog/guide-pgpool-postgresql-part-one

Классические варианты

PgBouncer

PgBouncer - пул соединений:

- легковесный - 2 Кб на соединение
- можно выбрать тип соединения: на сессию, транзакцию или каждую операцию
- онлайн-реконфигурация без сброса подключений

Неплохо бы еще добавить HAPROXY для балансинга нагрузки

https://www.percona.com/blog/2018/10/02/scaling-postgresql-using-connection-poolers-and-load-balancers-for-an-enterprise-grade-environment/

Сравнение

https://scalegrid.io/blog/postgresql-connection-pooling-part-4-pgbouncer-vs-pgpool/

Снова проблема

А куда балансировать нагрузку?

У нас ведь 1 БД...

Снова проблема

А куда балансировать нагрузку?

У нас ведь 1 БД...

Ответ - используем репликацию и лоад балансер и отправляем запросы на запись на PRIMARY сервер, на чтение на SECONDARY сервера

Варианты репликации

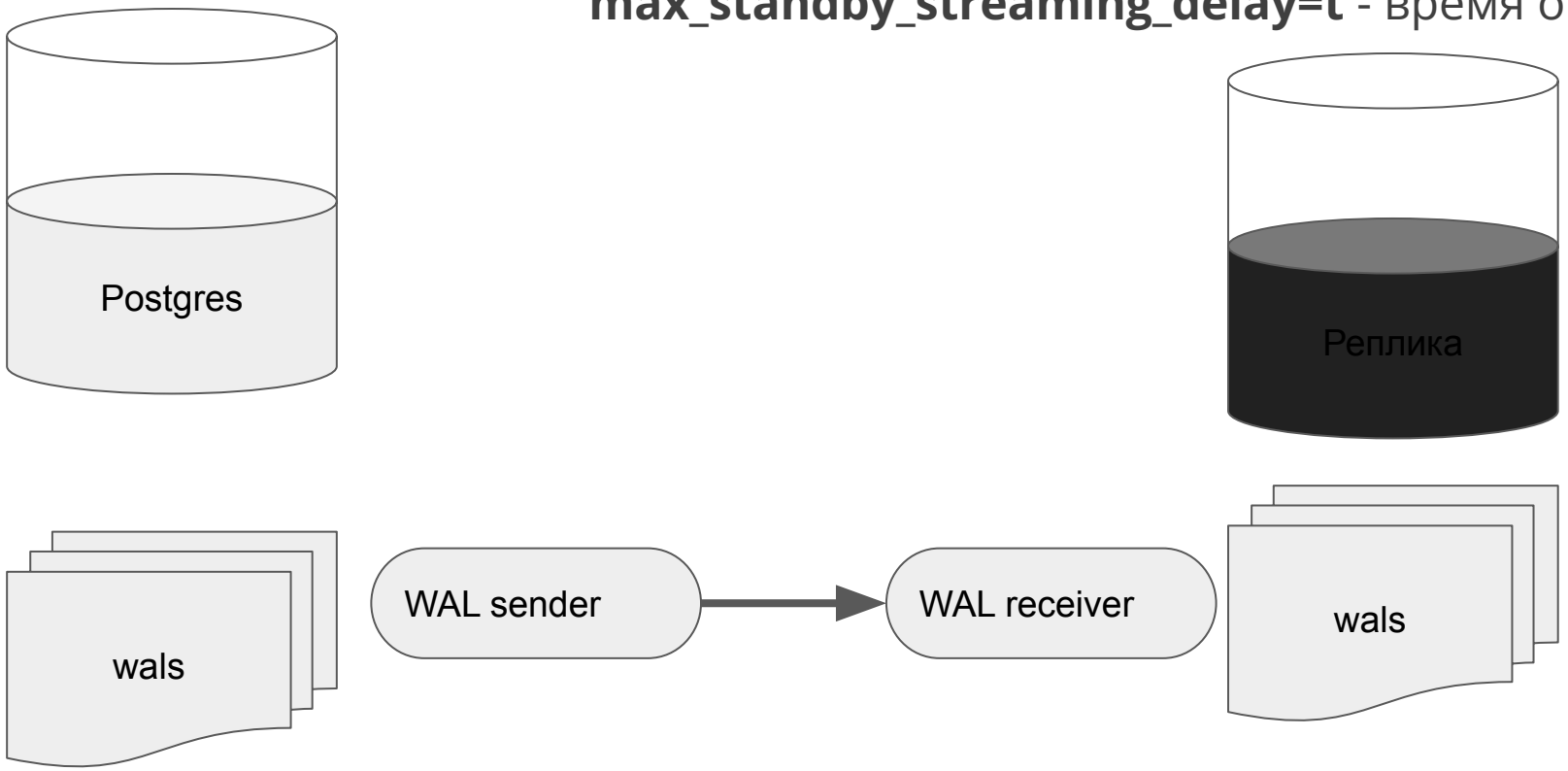
Горячий резерв for high availability

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

synchronous_commit = on

hot_standby_feedback = off - не сообщаем о транзакциях мастеру

max_standby_streaming_delay=t - время отметы конфл.транз.



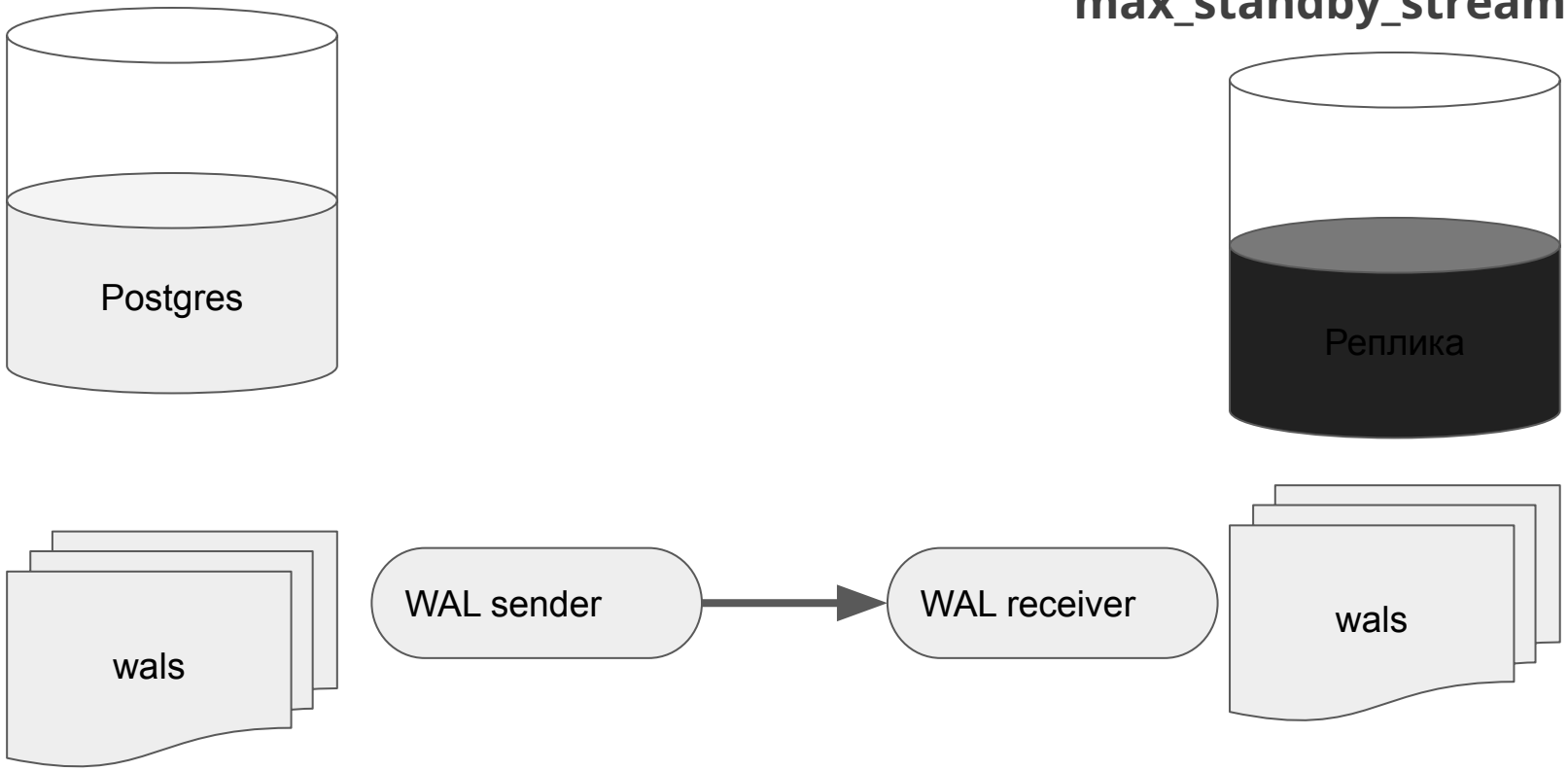
Балансировка OLTP

- много коротких запросов
- запросы на реплике должны отрабатываться
- долгие запросы повлияют на мастер, но у нас их не должно быть

synchronous_commit = off

hot_standby_feedback = on

max_standby_streaming_delay=T



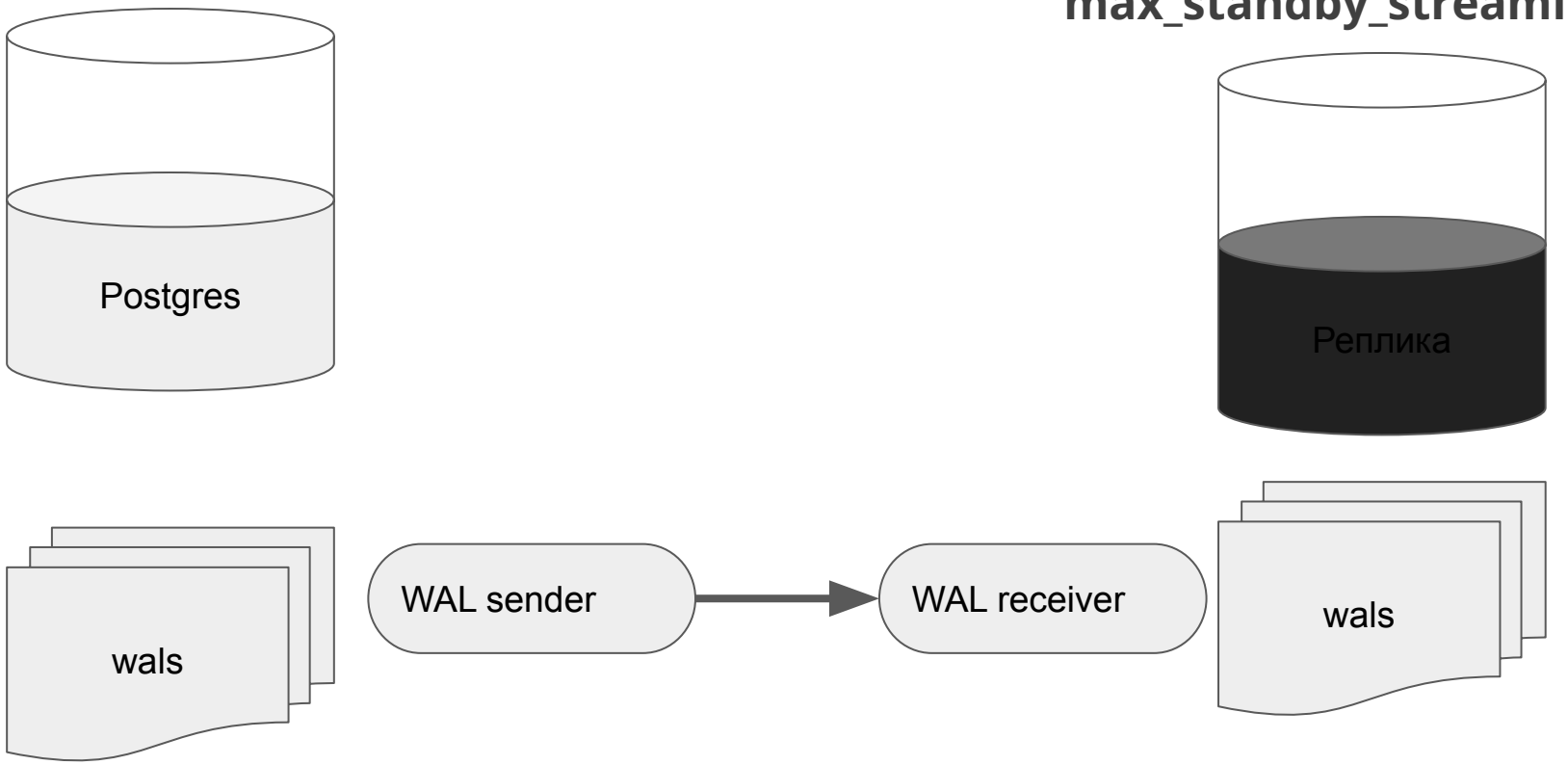
Реплика для отчетов

- запросы на реплике должны отрабатывать, в том числе и долгие. Самые актуальные данные не нужны
- мастер не должен испытывать от этого неудобства

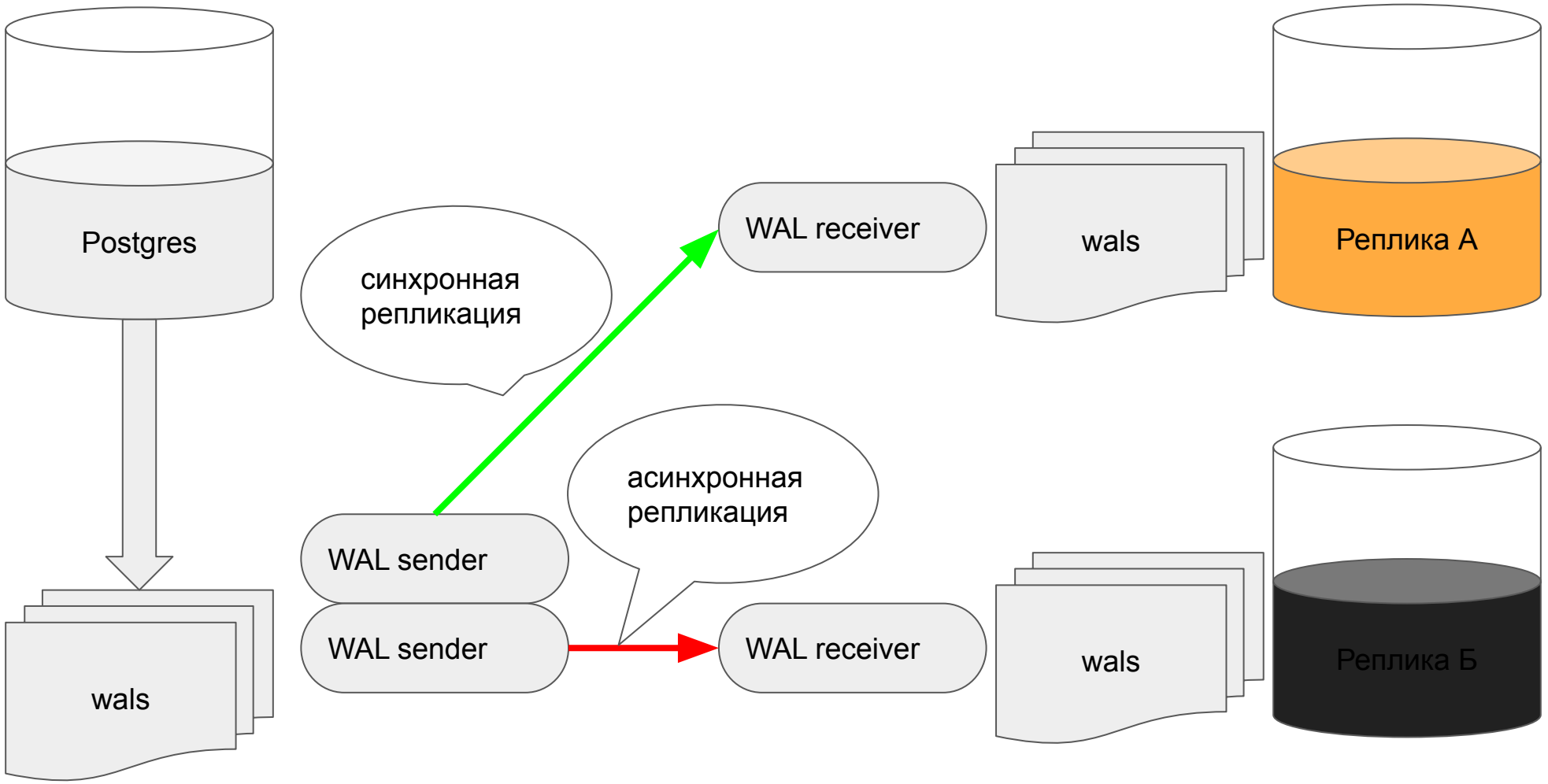
synchronous_commit = off

hot_standby_feedback = off

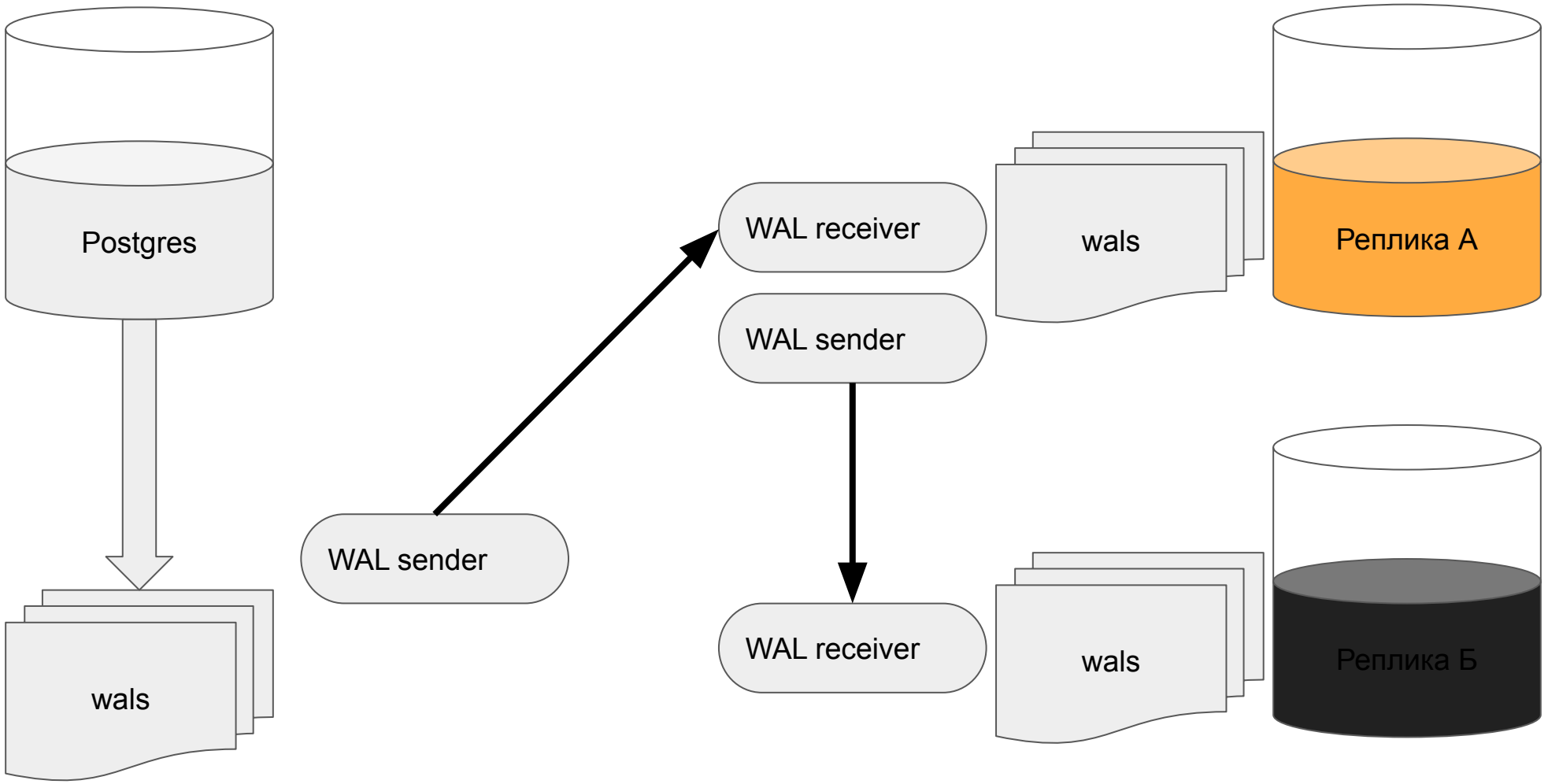
max_standby_streaming_delay=-1



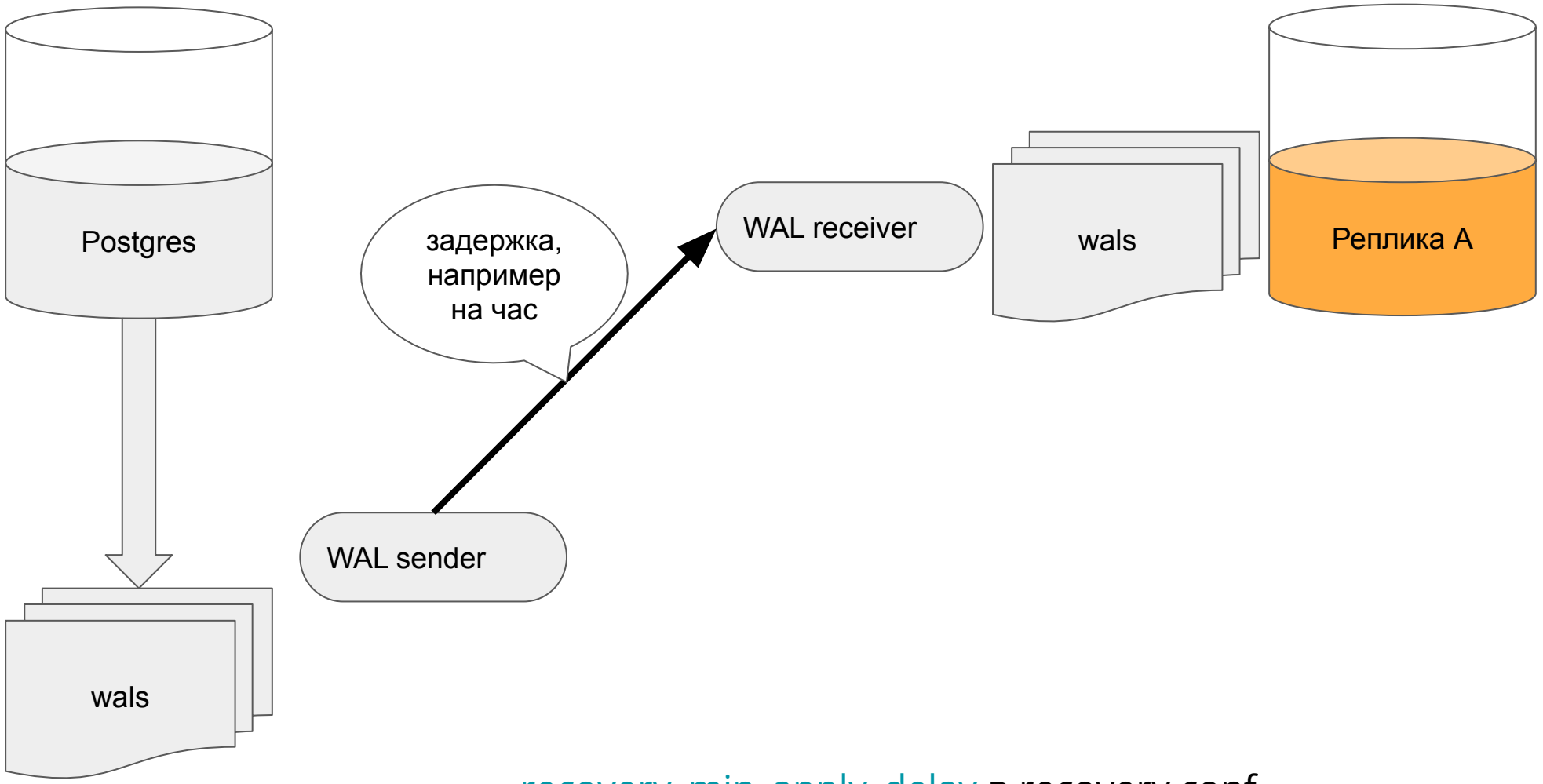
Горизонтальное масштабирование с HA



Каскадная репликация

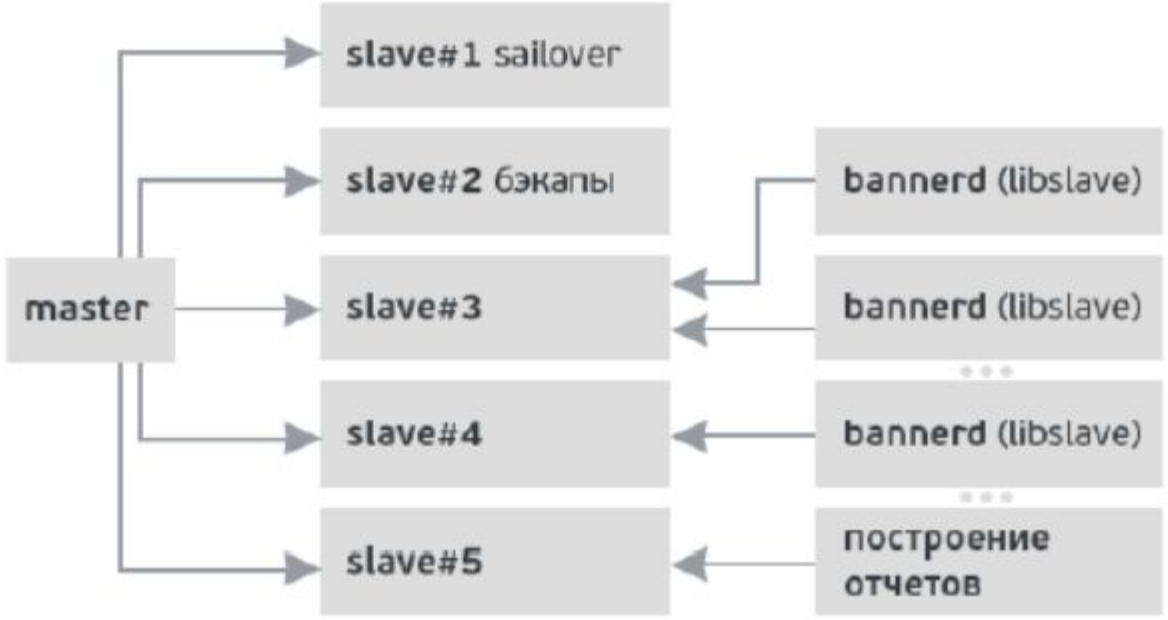


Time machine



recovery_min_apply_delay в recovery.conf

Как в мейл.ру



*Структура проекта Mail.Ru Target

Снова проблема

Вроде все хорошо, но..

кто будет переключать SECONDARY в PRIMARY при проблемах?

Решение 0

Вроде все хорошо, но..

кто будет переключать SECONDARY в PRIMARY при проблемах?

PgPool II

Решение 1

Вроде все хорошо, но..

кто будет переключать SECONDARY в PRIMARY при проблемах?

Pg_auto_failover:

- HA
- отказоустойчивость
- синхронный и асинхронный режим
- открытый исходный код

https://github.com/citusdata/pg_auto_failover

Решение 2

Вроде все хорошо, но..

кто будет переключать SECONDARY в PRIMARY при проблемах?

Repmgr:

- открытый исходный код
- автоматическое переключение основного сервера при падении
- очень простая установка и масштабирование

https://github.com/EnterpriseDB/repmgr

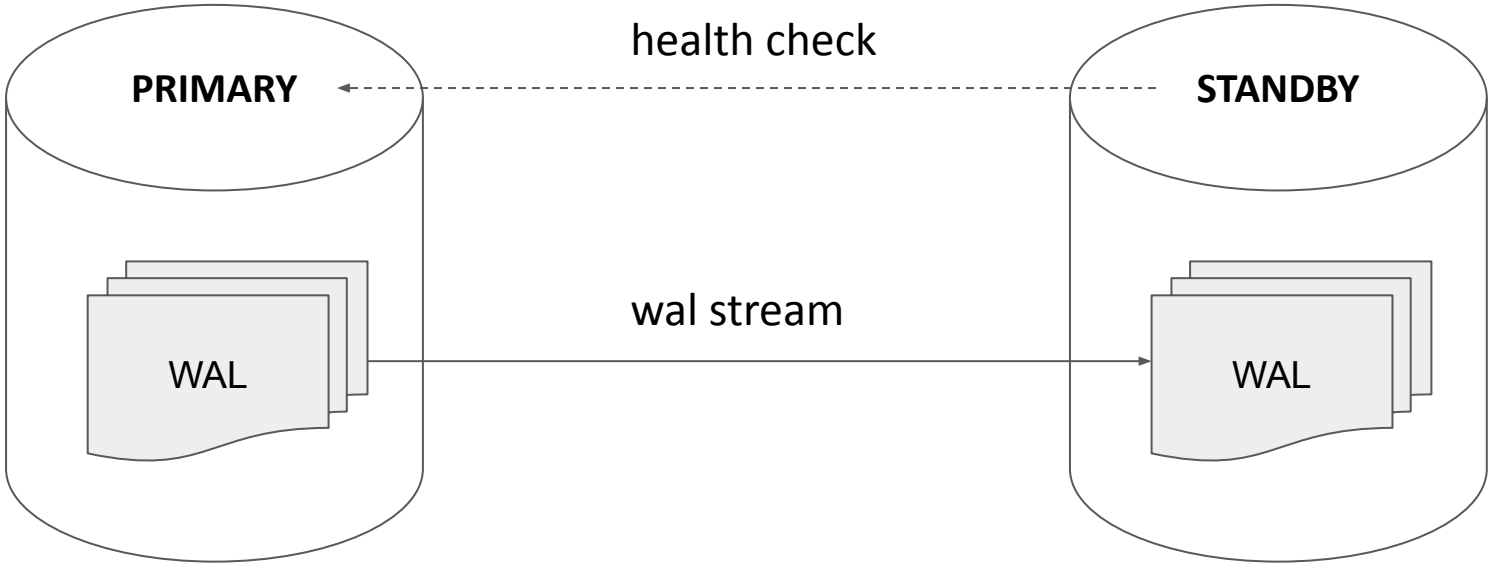
Решение X

Вроде все хорошо, но..

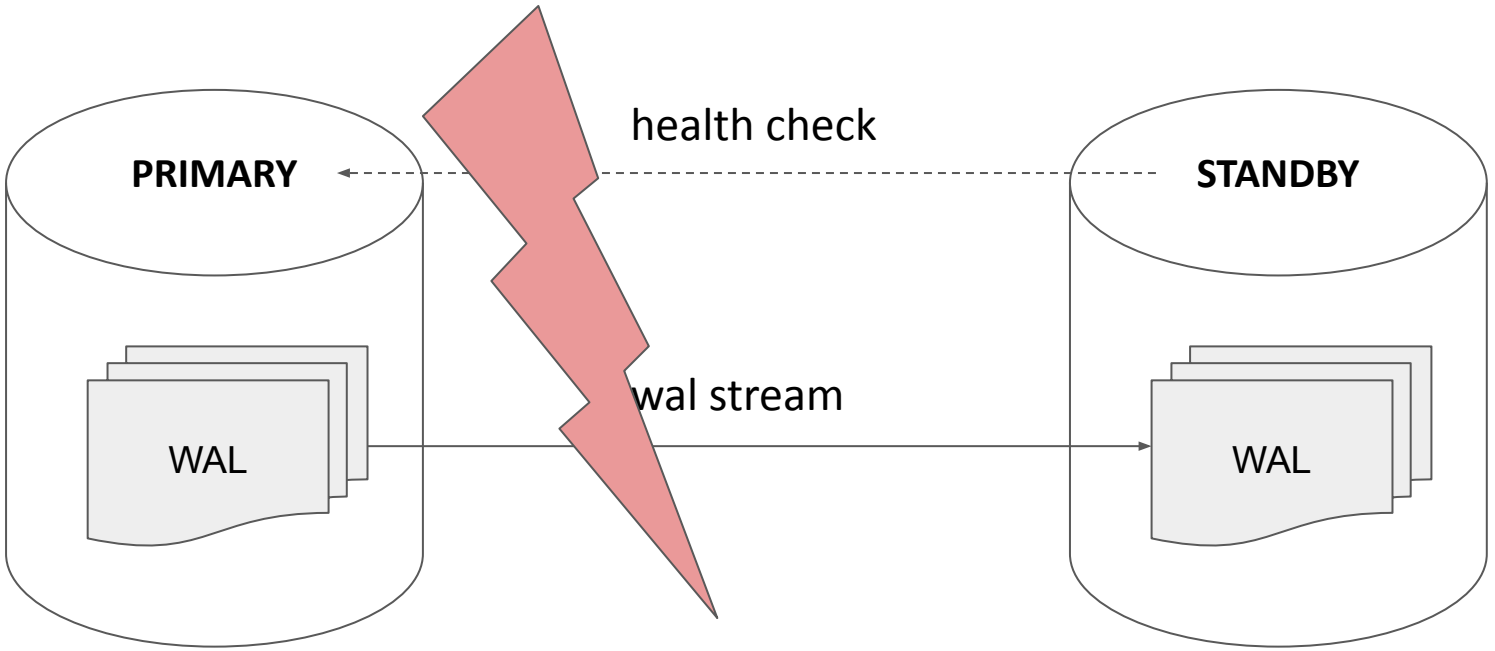
кто будет переключать SECONDARY в PRIMARY при проблемах?

А мы ничего не забыли?

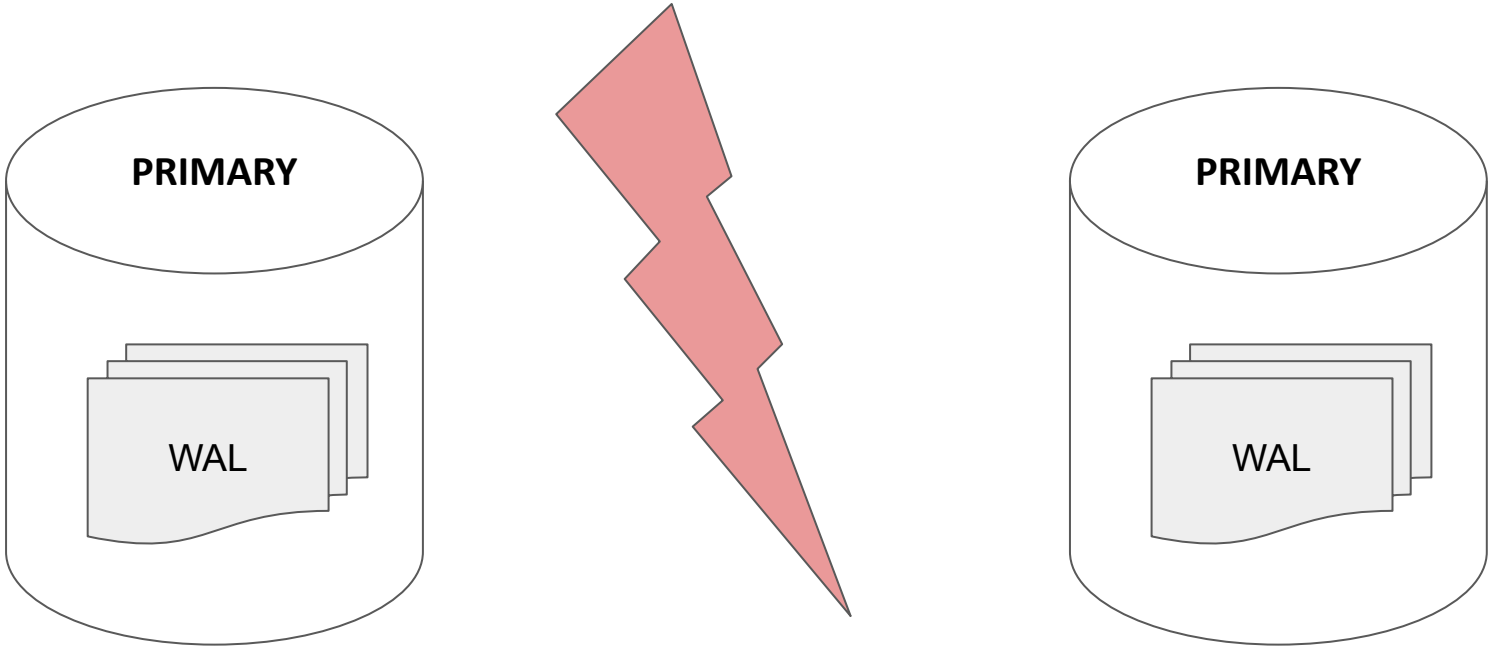
Кластер из 2 нод



Что произойдет при обрыве сети?

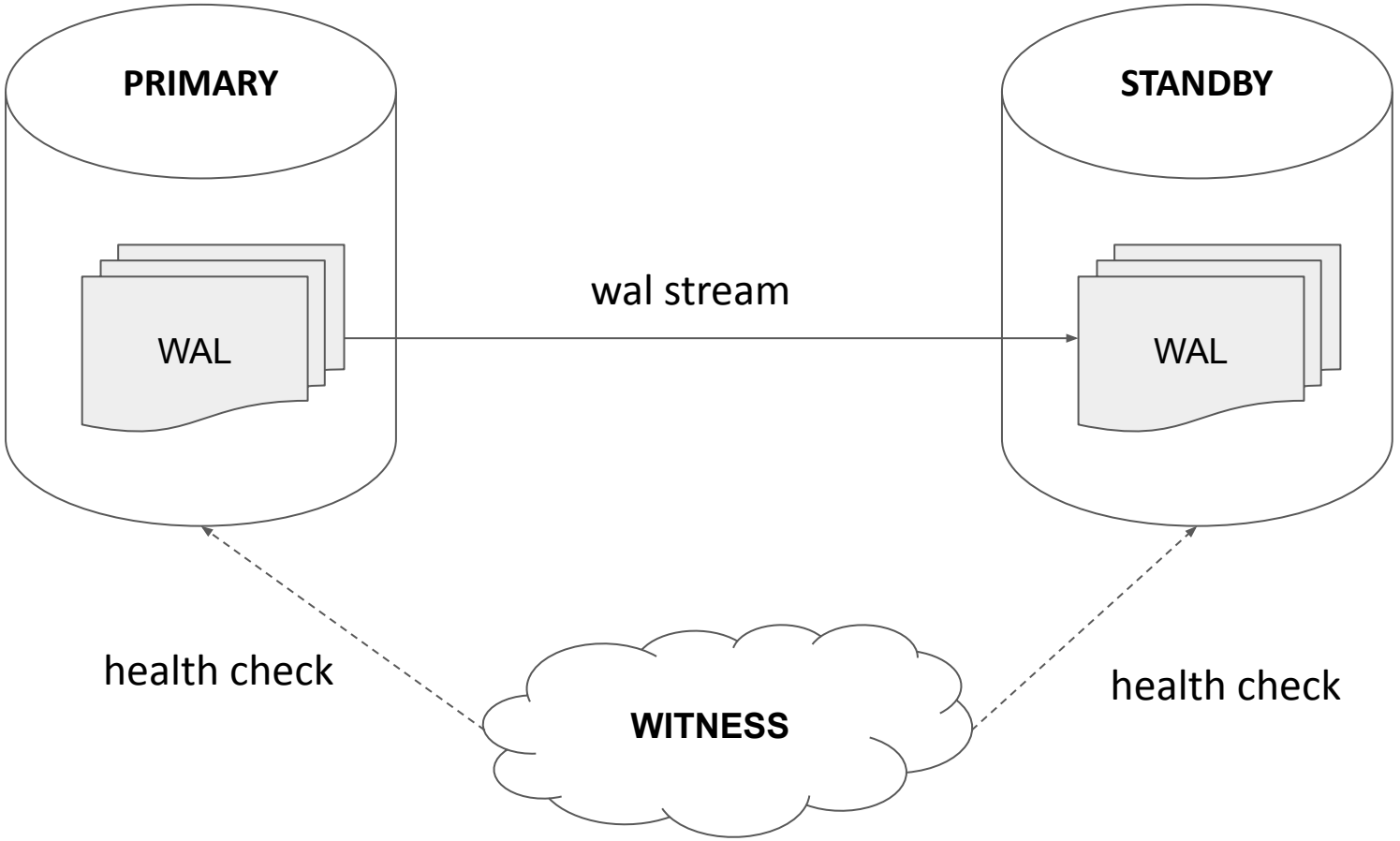


Правильно! Splitbrain



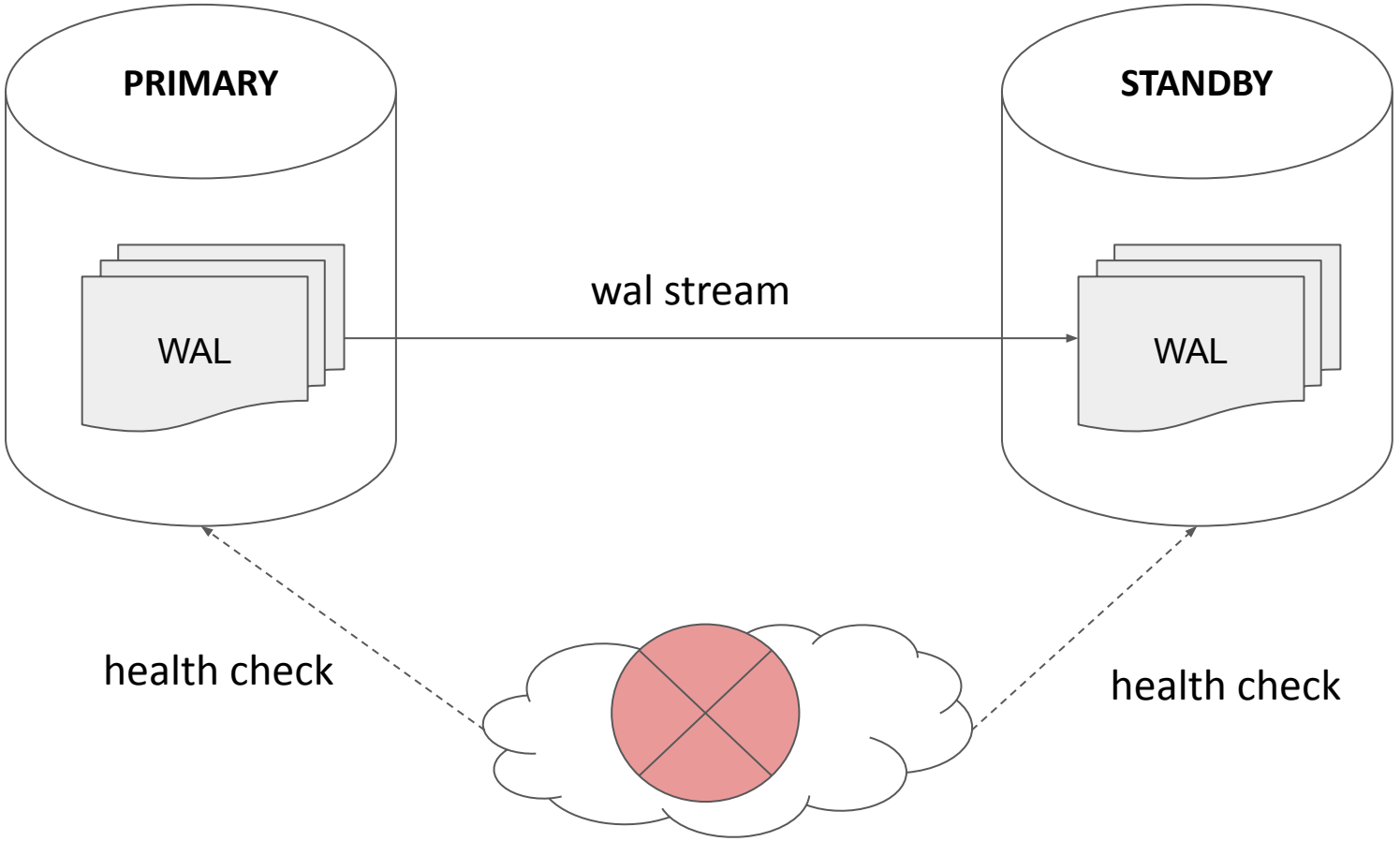
Кластер из 2 нод с Witness

Что может пойти не так?



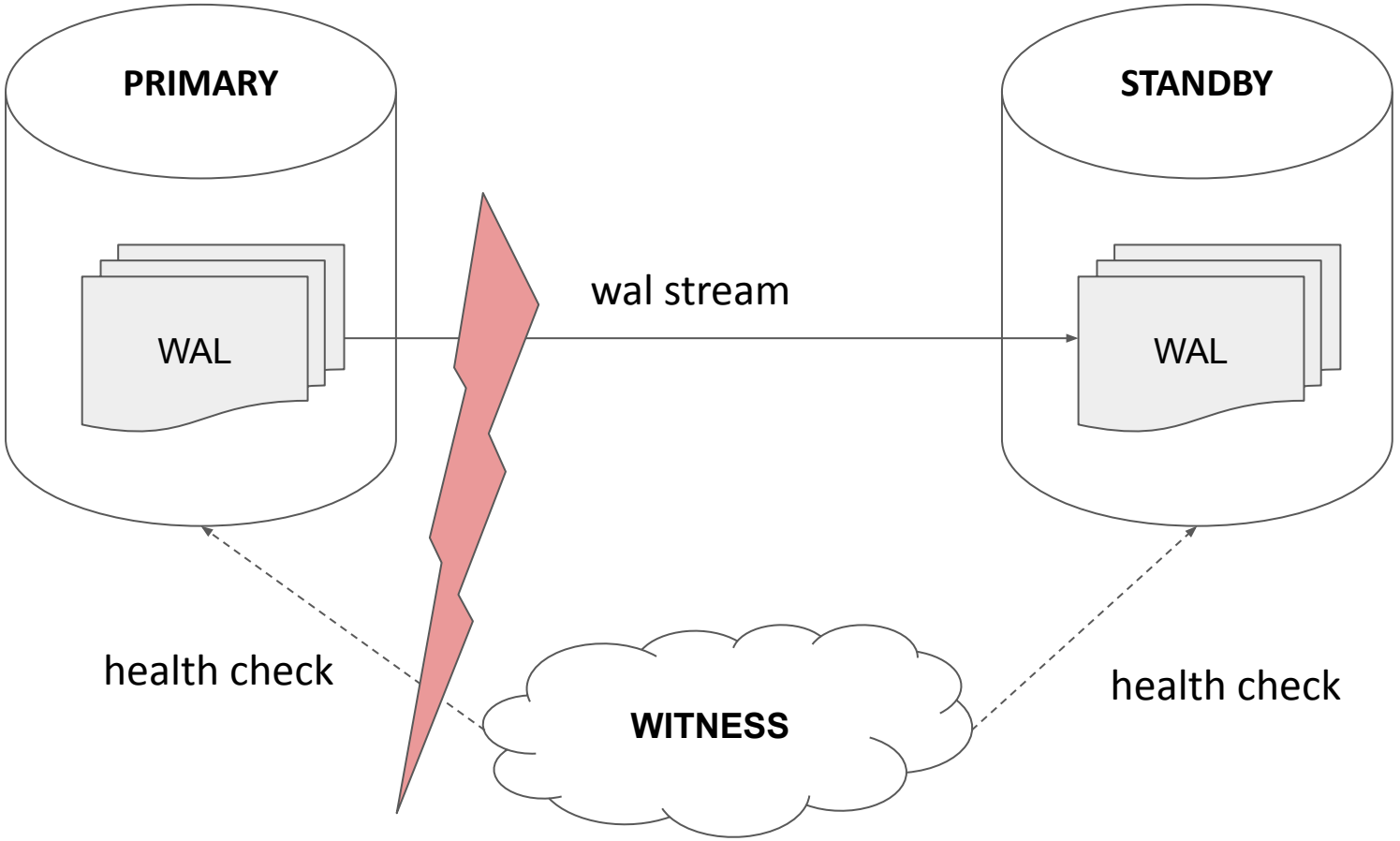
Кластер из 2 нод с Witness

Умрет наблюдатель



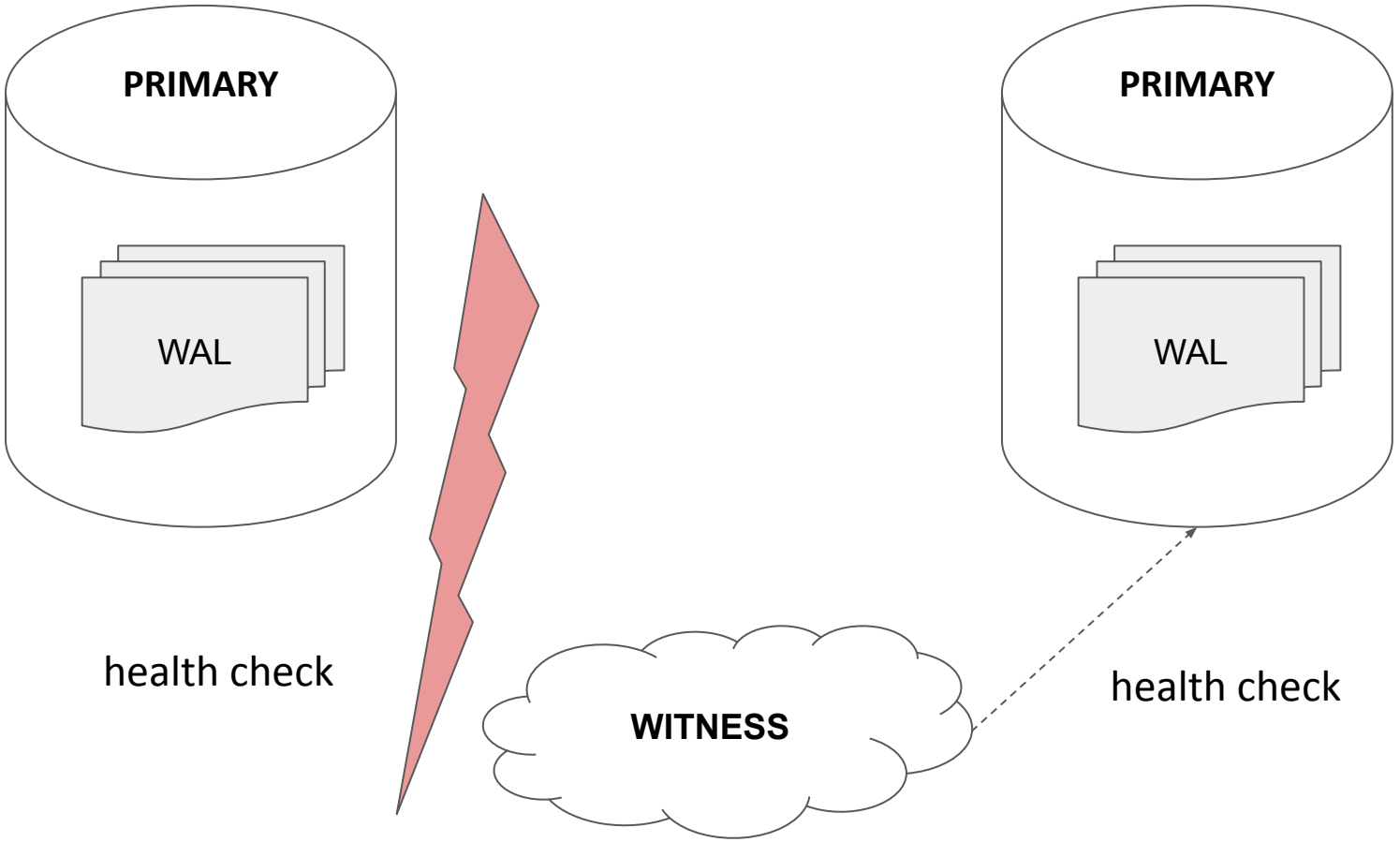
Кластер из 2 нод с Witness

Обрыв соединения с основной нодой



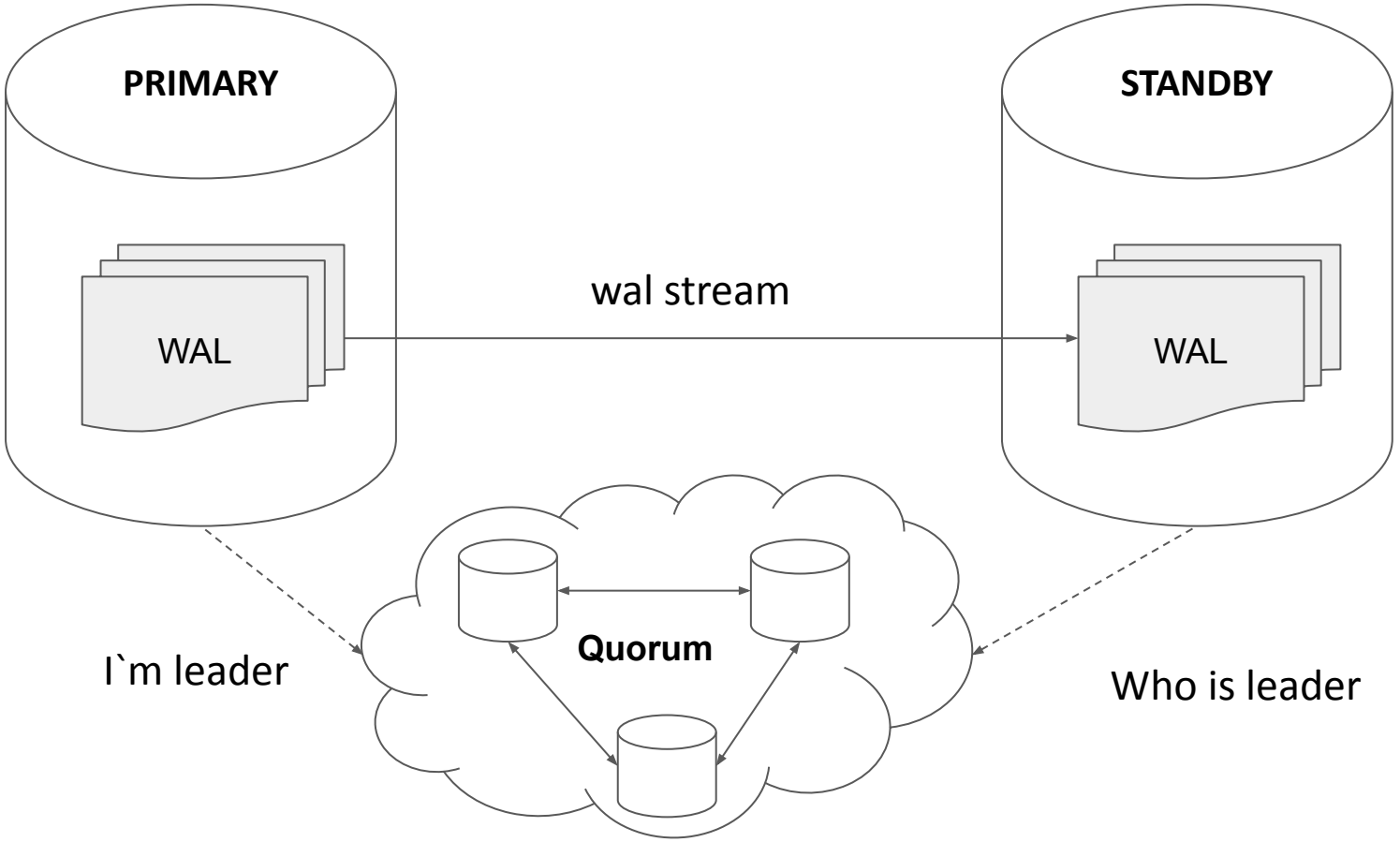
Кластер из 2 нод с Witness

И опять splitbrain

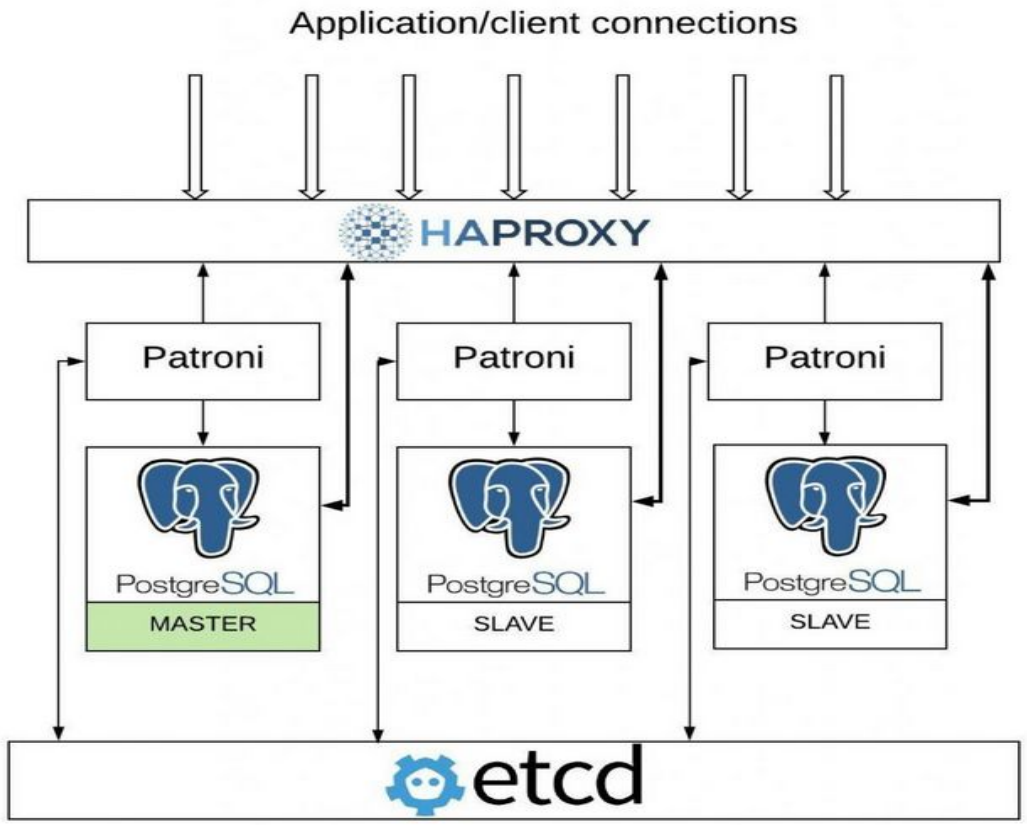


Кластер из 2 нод с ETCD/CONSUL/ZOOKEEPER

Одно из решений (Patroni)



Patroni



можно еще добавить pgbouncer, keepalived+2 HAPROXY для HA

Еще варианты кластеров

HA:

- Stolon
- Slony
- ClusterControl
- Kubegres

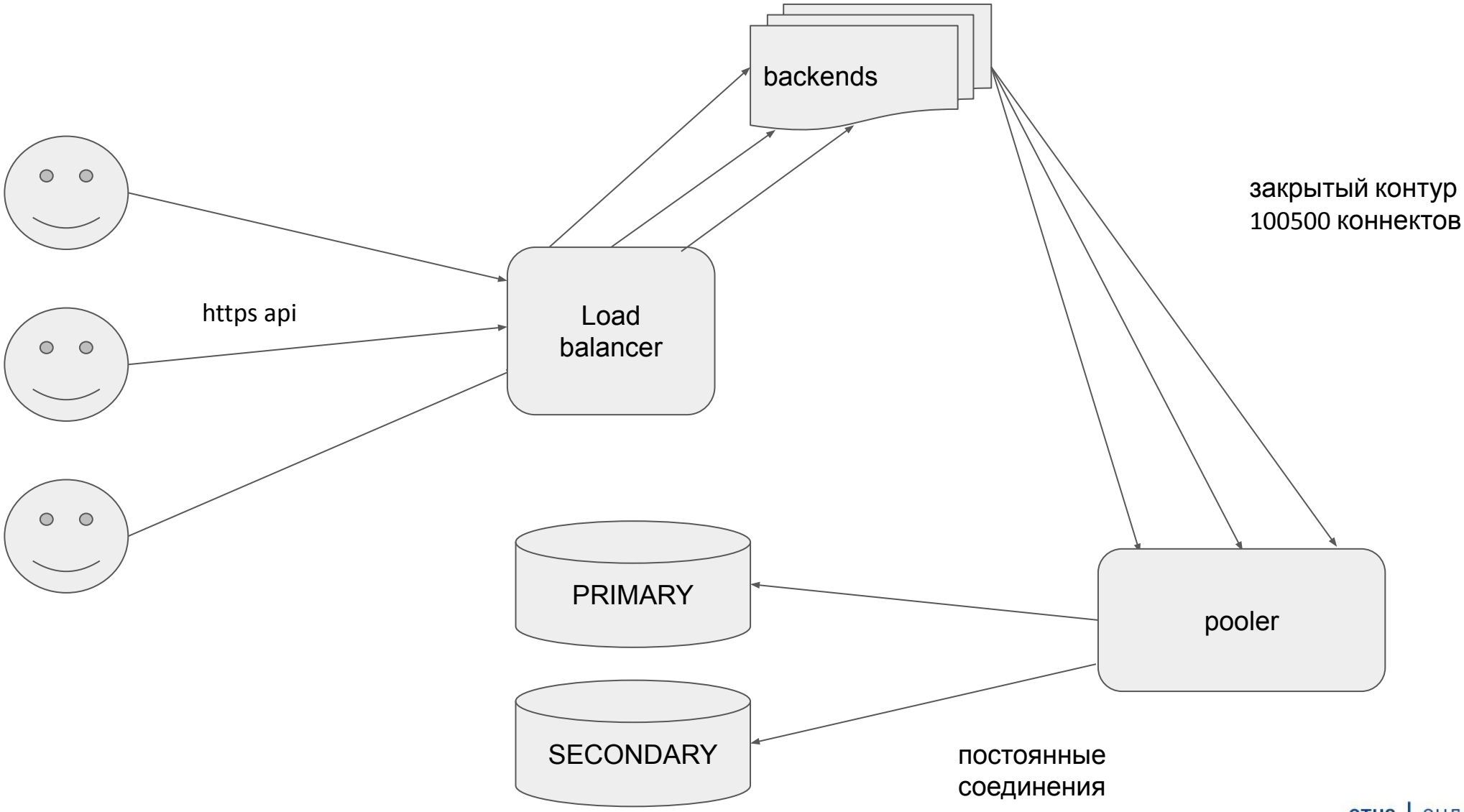
Параллельные кластера:

- Postgres-BDR
- CitusData
- Bucardo
- CockroachDB
- Yogabyte
- Greenplum

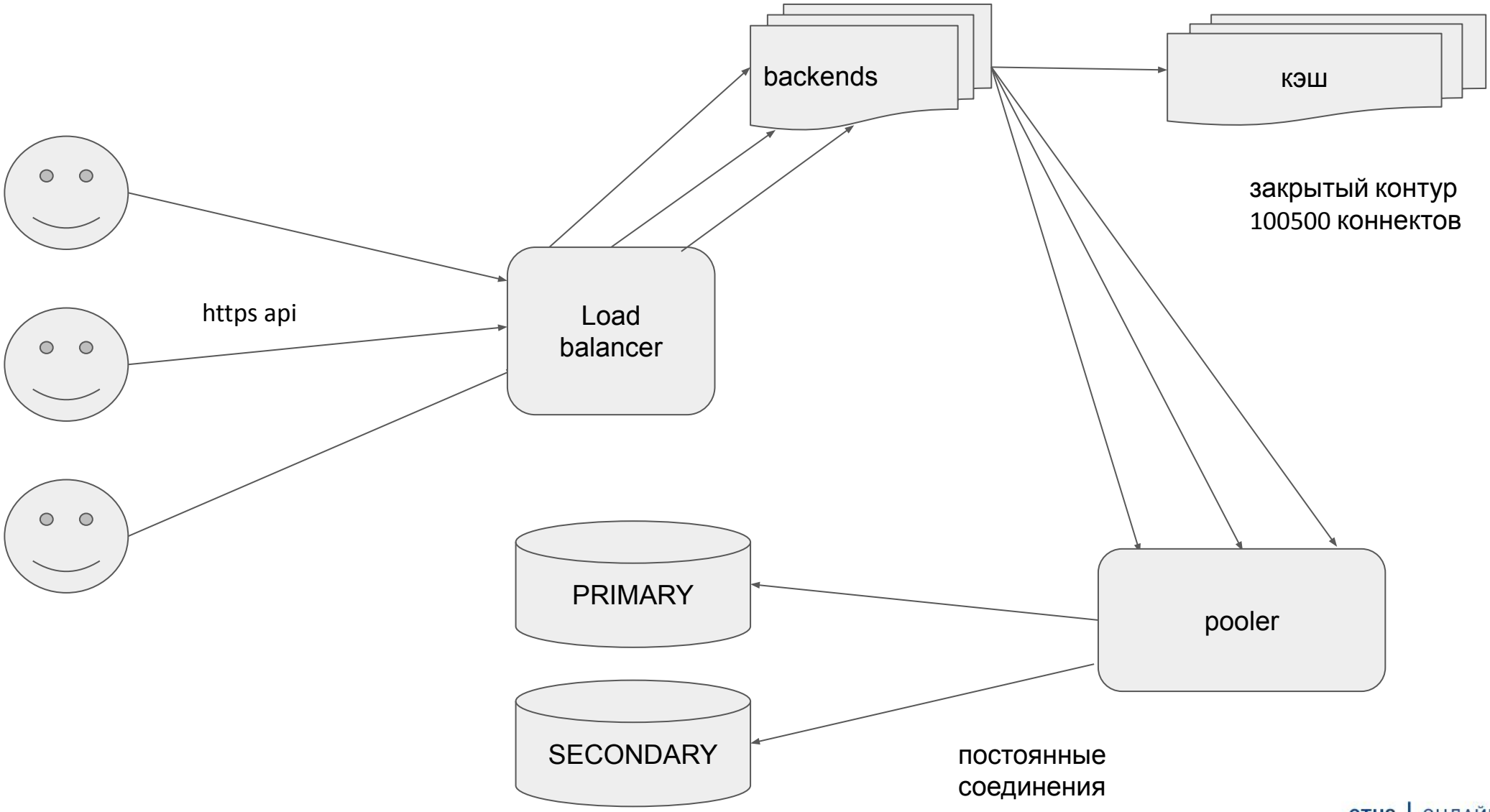
А уж сколько облачных решений...

Добавили реплики. Что дальше

Ну теперь то все хорошо?



Добавим кэш


backends
кэш
закрытый контур
100500 коннектов
https api
Load
balancer
PRIMARY
SECONDARY
pooler
постоянные
соединения

Какие проблемы при этом есть?

- **Если делать локальный кэш, то лоад балансер должен запросы с 1 фронта всегда отправлять на нужный бэкенд**
- **Если делать общий tarantool/redis (не совсем кэш), то сетевые задержки**

И общие проблемы:

- **TTL**
- **инвалидация кэша**

Как думаете, на этом все?)

минитест

https://forms.gle/nrzsMpQWiUKMqf5RA

2-3 минуты

Варианты облаков

Варианты облаков

Google https://cloud.google.com/

AWS https://aws.amazon.com/ru/

ЯндексОблако https://cloud.yandex.ru/

VKCloud https://mcs.mail.ru/

SberCloud https://sbercloud.ru/ru

Selectel https://selectel.ru/services/cloud/servers/

ACID

ACID

Транзакционные системы, OLTP - **O**n**l**ine **T**ransaction **P**rocessing

ACID

- **Atomicity** — Атомарность
- **Consistency** — Согласованность
- **Isolation** — Изолированность
- **Durability** — Долговечность

транзация (**transaction**)

- называется множество операций, выполняемое приложением
- которое переводит базу данных из одного корректного состояния в другое корректное состояние (согласованность)
- при условии, что транзакция выполнена полностью (атомарность)
- и без помех со стороны других транзакций (изолированность)

Уровни изоляции транзакций

Стандарт SQL допускает четыре уровня изоляции, которые определяются в терминах аномалий, которые допускаются при конкурентном выполнении транзакций на этом уровне:

● «Грязное» чтение (dirty read). Транзакция T1 может читать строки измененные, но еще не зафиксированные, транзакцией T2. Отмена изменений (ROLLBACK) в T2 приведет к тому, что T1 прочитает данные, которых никогда не существовало.

● Неповторяющееся чтение (non-repeatable read). После того, как транзакция T1 прочитала строку, транзакция T2 изменила или удалила эту строку и зафиксировала изменения (COMMIT). При повторном чтении этой же строки транзакция T1 видит, что строка изменена или удалена.

● Фантомное чтение (phantom read). Транзакция T1 прочитала набор строк по некоторому условию. Затем транзакция T2 добавила строки, также удовлетворяющие этому условию. Если транзакция T1 повторит запрос, она получит другую выборку строк.

● Аномалия сериализации - Результат успешной фиксации группы транзакций оказывается несогласованным при всевозможных вариантах исполнения этих транзакций по очереди.

Уровни изоляции транзакций

	«грязное» чтение	неповторяю-щееся чтение	фантомное чтение	аномалия сериализации
Read Uncommitted	Допускается, но не в PG	да	да	да
Read Committed	-	да	да	да
Repeatable Read	-	-	Допускается, но не в PG	да
Serializable	-	-	-	-

на всех уровнях не допускается потеря зафиксированных изменений

https://habr.com/ru/company/otus/blog/501294/
https://postgrespro.ru/docs/postgrespro/13/transaction-iso

Уровни изоляции транзакций

Аномалия сериализации

class | value

-------+-------

1 | 10

1 | 20

2 | 100

2 | 200

сериализуемая транзакция A вычисляет:

SELECT SUM(value) FROM mytab WHERE class = 1; добавляем запись с суммой и class 2
сериализуемая транзакция B вычисляет:
SELECT SUM(value) FROM mytab WHERE class = 2; добавляем запись с суммой и class 1
что произойдет при попытке фиксации изменений?

Уровни изоляции транзакций

ОШИБКА: не удалось сериализовать доступ из-за зависимостей чтения/записи между транзакциями

причем если бы одна из транзакций была бы repeatable read, все бы успешно закоммитилось

Случай из жизни

Случай из жизни

У нас тут надысь клиенты пишут АААААА у нас прод не работает спасите помогите. Созваниваемся, начинаем разбираться - нескольких микросервисов нет. Спрашиваем где, говорят, они жрали дофига. Говорим, окей, давайте вот так и так задеплойте. Заработало. Спрашивают, а где данные???? Смотрим, а они и сторадж почистили, чтобы место не занимал. Спрашиваем, а где ж, ..., бекапящие сторадж кронджобы??? Говорят, что выключили, так как бекапы бесполезно тратили деньги их компании

%)

Результаты опроса

https://docs.google.com/forms/d/1PrU_ox2kyZehdHuJFQ4LaELPvBdVKnqDRsIF6fDNGlc/edit#responses

ДЗ

Домашнее задание

- создать новый проект в Google Cloud Platform, Яндекс облако или на любых ВМ, например postgres2023-<yyyymmdd>, где yyyymmdd год, месяц и день вашего рождения (имя проекта должно быть уникально на уровне GCP)

- далее создать инстанс виртуальной машины Compute Engine с дефолтными параметрами - 1-2 ядра, 2-4Гб памяти, любой линукс, на курсе Ubuntu 100%

- добавить свой ssh ключ в GCE metadata

- зайти удаленным ssh (первая сессия), не забывайте про ssh-add

- поставить PostgreSQL из пакетов apt install

- зайти вторым ssh (вторая сессия)

- запустить везде psql из под пользователя postgres

- выключить auto commit

- сделать в первой сессии новую таблицу и наполнить ее данными

```
create table persons(id serial, first_name text, second_name text);
insert into persons(first_name, second_name) values('ivan', 'ivanov');
insert into persons(first_name, second_name) values('petr', 'petrov');
commit;
```

- посмотреть текущий уровень изоляции: show transaction isolation level

- начать новую транзакцию в обоих сессиях с дефолтным (не меняя) уровнем изоляции

- в первой сессии добавить новую запись

```
insert into persons(first_name, second_name) values('sergey', 'sergeev');
```

- сделать select * from persons во второй сессии

- видите ли вы новую запись и если да то почему?

- завершить первую транзакцию - commit;

Домашнее задание

- сделать select * from persons во второй сессии
- видите ли вы новую запись и если да то почему?
- завершите транзакцию во второй сессии
- начать новые но уже repeatable read транзакции - set transaction isolation level repeatable read;
- в первой сессии добавить новую запись

  insert into persons(first_name, second_name) values('sveta', 'svetova');
- сделать select * from persons во второй сессии
- видите ли вы новую запись и если да то почему?
- завершить первую транзакцию - commit;
- сделать select * from persons во второй сессии
- видите ли вы новую запись и если да то почему?
- завершить вторую транзакцию
- сделать select * from persons во второй сессии
- видите ли вы новую запись и если да то почему?

ДЗ сдаем в виде миниотчета в markdown в гите

Рефлексия

Рефлексия

Как вам занятие?

Остались ли вопросы?

Заполните, пожалуйста,
опрос о занятии по ссылке в чате
https://otus.ru/polls/63764/

Спасибо за внимание!
Приходите на следующие вебинары

Аристов Евгений